

智能边缘平台

# 用户指南

文档版本 01  
发布日期 2023-11-23



版权所有 © 华为技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## 商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## 注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

# 安全声明

## 漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：

<https://www.huawei.com/cn/psirt/vul-response-process>

如企业客户须获取漏洞信息，请参见如下网址：

<https://securitybulletin.huawei.com/enterprise/cn/security-advisory>

# 目录

<b>1 使用 IEF 构建边缘计算</b>	<b>1</b>
<b>2 服务实例</b>	<b>3</b>
<b>3 专业版操作指南</b>	<b>6</b>
3.1 节点管理	6
3.1.1 边缘节点概述	6
3.1.2 配置边缘节点环境	7
3.1.3 注册边缘节点	10
3.1.4 纳管边缘节点	13
3.1.5 升级边缘节点	14
3.1.6 日志、监控和告警	16
3.1.7 安装并配置 GPU 驱动	20
3.1.8 边缘核心软件 EdgeCore 配置管理	21
3.1.9 删除边缘节点	22
3.2 终端设备管理	23
3.2.1 终端设备与设备孪生	23
3.2.2 设备模板	24
3.2.3 终端设备	26
3.2.4 终端设备绑定到边缘节点	26
3.2.5 设备孪生工作原理	27
3.2.6 设备数据上云	29
3.2.7 使用证书进行安全认证	36
3.2.8 MQTT Topic	44
3.2.8.1 设备孪生变更	44
3.2.8.2 设备孪生 delta	46
3.2.8.3 设备成员变更	47
3.2.8.4 设备属性变更	48
3.2.8.5 设备成员获取	49
3.2.8.6 设备成员获取结果	49
3.2.8.7 设备孪生获取	50
3.2.8.8 设备孪生获取结果	51
3.2.8.9 设备孪生更新	52
3.2.8.10 设备孪生更新结果	53

3.2.8.11 请求加密数据.....	54
3.2.8.12 获取加密数据.....	54
3.2.8.13 添加告警.....	55
3.2.8.14 清除告警.....	59
3.2.8.15 自定义 Topic.....	59
3.3 容器应用管理.....	60
3.3.1 容器应用.....	60
3.3.2 应用模板.....	68
3.3.3 配置项.....	72
3.3.4 密钥.....	73
3.3.5 加密数据.....	75
3.3.6 健康检查配置说明.....	76
3.4 边云消息.....	77
3.4.1 边云消息概述.....	78
3.4.2 云端下发消息到边缘节点.....	81
3.4.3 边缘节点上报消息到云端.....	85
3.4.4 系统订阅.....	88
3.5 批量管理.....	92
3.5.1 批量节点注册.....	92
3.5.2 批量节点升级.....	96
3.5.3 批量应用部署.....	98
3.5.4 批量应用升级.....	105
3.6 审计.....	106
3.6.1 支持云审计的关键操作.....	106
3.6.2 如何查看审计日志.....	108
3.7 权限管理.....	109
3.7.1 创建用户并授权使用 IEF.....	109
3.7.2 自定义策略.....	110
3.7.3 IEF 资源.....	111
3.7.4 IEF 请求条件.....	111
<b>4 铂金版操作指南.....</b>	<b>114</b>
4.1 节点管理.....	114
4.1.1 边缘节点概述.....	114
4.1.2 配置边缘节点环境.....	115
4.1.3 注册边缘节点.....	118
4.1.4 纳管边缘节点.....	121
4.1.5 边缘节点组.....	122
4.1.6 升级边缘节点.....	126
4.1.7 日志、监控和告警.....	127
4.1.8 安装并配置 GPU 驱动.....	131
4.1.9 边缘核心软件 EdgeCore 配置管理.....	132
4.1.10 删除边缘节点.....	133

4.2 终端设备管理.....	134
4.2.1 终端设备与设备孪生.....	134
4.2.2 设备模板.....	135
4.2.3 终端设备.....	136
4.2.4 终端设备绑定到边缘节点.....	137
4.2.5 设备孪生工作原理.....	138
4.2.6 设备数据上云.....	140
4.2.7 使用证书进行安全认证.....	147
4.2.8 MQTT Topic.....	155
4.2.8.1 设备孪生变更.....	156
4.2.8.2 设备孪生 delta.....	157
4.2.8.3 设备成员变更.....	158
4.2.8.4 设备属性变更.....	159
4.2.8.5 设备成员获取.....	160
4.2.8.6 设备成员获取结果.....	160
4.2.8.7 设备孪生获取.....	161
4.2.8.8 设备孪生获取结果.....	162
4.2.8.9 设备孪生更新.....	163
4.2.8.10 设备孪生更新结果.....	164
4.2.8.11 请求加密数据.....	165
4.2.8.12 获取加密数据.....	166
4.2.8.13 添加告警.....	166
4.2.8.14 清除告警.....	170
4.2.8.15 自定义 Topic.....	171
4.3 容器应用管理.....	172
4.3.1 容器应用.....	172
4.3.2 应用模板.....	181
4.3.3 亲和与反亲和调度.....	186
4.3.4 配置项.....	189
4.3.5 密钥.....	190
4.3.6 加密数据.....	192
4.3.7 健康检查配置说明.....	193
4.4 应用网格.....	194
4.4.1 操作场景.....	195
4.4.2 服务.....	195
4.4.3 网关.....	198
4.5 边云消息.....	201
4.5.1 边云消息概述.....	201
4.5.2 云端下发消息到边缘节点.....	203
4.5.3 边缘节点上报消息到云端.....	207
4.5.4 系统订阅.....	210
4.6 批量管理.....	215

---

4.6.1 批量节点注册.....	215
4.6.2 批量节点升级.....	219
4.6.3 批量应用部署.....	221
4.6.4 批量应用升级.....	227
4.7 审计.....	228
4.7.1 支持云审计的关键操作.....	228
4.7.2 如何查看审计日志.....	231
4.8 权限管理.....	232
4.8.1 创建用户并授权使用 IEF.....	232
4.8.2 自定义策略.....	234
4.8.3 系统委托说明.....	234

# 1 使用 IEF 构建边缘计算

智能边缘平台（Intelligent EdgeFabric）通过纳管用户的边缘节点，提供将云上应用延伸到边缘的能力，联动边缘和云端的数据，同时，在云端提供统一的边缘节点/应用监控、日志采集等运维能力，为企业提供完整的边缘计算解决方案。

## 使用 IEF 构建边缘计算

使用IEF构建边缘计算的步骤如**图1-1**所示。

### 1. 纳管边缘节点，绑定终端设备。

使用IEF构建边缘计算首先需要将边缘节点纳入IEF的管理（通过在边缘节点安装边缘节点软件），并将终端设备绑定到边缘节点，做完这些后您就可以通过IEF往边缘节点部署应用。

纳管边缘节点、绑定终端设备的详细内容将在**节点管理**和**终端设备管理**中介绍。

### 2. 开发应用并制作镜像，上传到**容器镜像服务**（SWR）。

这个步骤是针对实际业务场景开发应用，开发完成后制作成容器镜像，并上传到SWR上，这样后面IEF下发应用后，边缘节点就可以从SWR中拉取应用镜像。

虽然这里将开发应用放在了**步骤 1**之后，但这两个步骤之间并没有明确的先后顺序，您也可以先开发应用，然后再纳管边缘节点、绑定终端设备。

### 3. 部署应用。

边缘节点纳管、应用开发完后，就可以通过IEF将应用部署到边缘节点，运行您的实际业务。

应用运行后，就能通过AOM对应用进行监控和告警，提供运维便利性。

部署应用的详细内容将在**容器应用管理**中详细介绍。

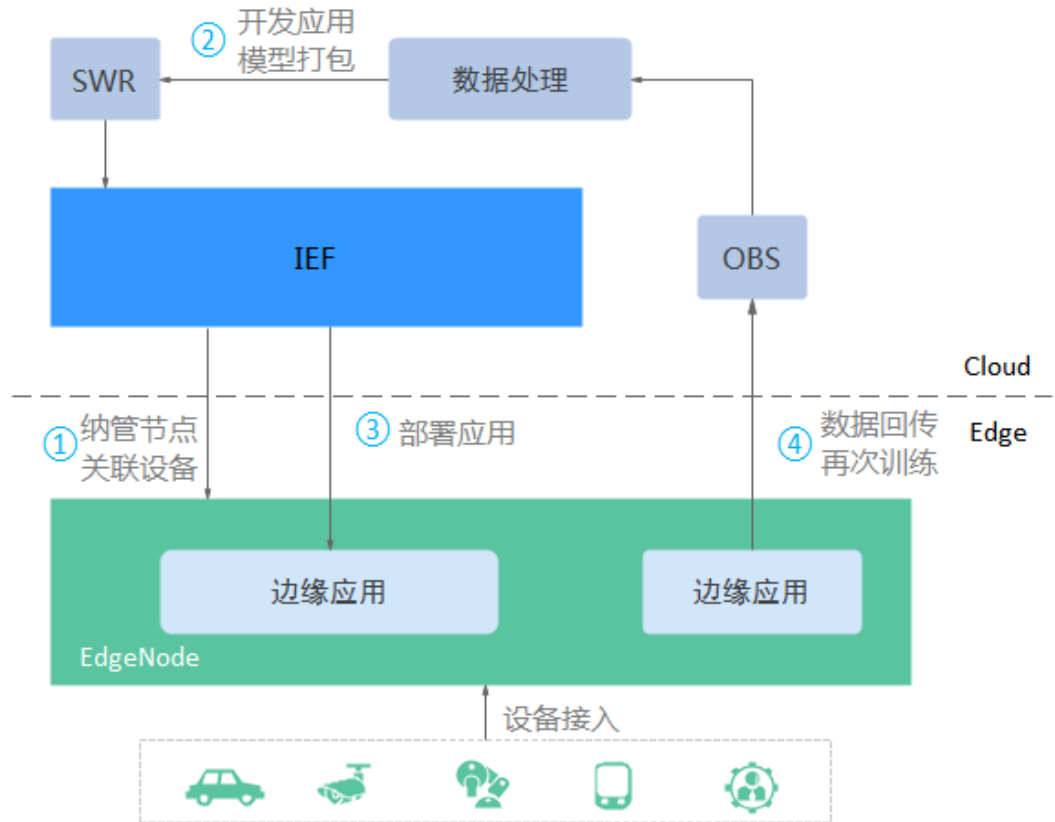
### 4. （可选）回传数据到云上做进一步处理，根据处理结果更新应用。

这个步骤与IEF的使用本身没有强相关，但是这是一个常见的根据数据改进应用的方法，您可以根据自身需求选择是否操作。

例如在边缘运行人脸识别应用，定期将采集到的数据上传到OBS中，然后将这些数据加入训练集，训练出新的、更好的人脸识别模型，使用新的模型打包成新的镜像，再更新到边缘节点的应用中。



图 1-1 构建边缘计算



# 2 服务实例

服务实例是指IEF上用于管理边缘节点、下发应用的管理集群，IEF当前支持两种服务实例。

- **专业版服务实例（默认）**：所有用户共享管理集群。支持节点管理、设备管理、容器应用管理、批量作业管理和边云消息等功能。
- **铂金版服务实例**：需要单独创建，单个用户独占管理集群。支持管理大规模节点，性能更高。在专业版基础上，增加支持节点组、应用网格等功能。

具体差异请参见[表2-1](#)。

表 2-1 版本功能说明

功能特性	说明	专业版	铂金版
边缘节点管理	支持注册、纳管边缘节点	√	√
终端设备管理	支持注册终端设备，支持绑定终端设备到边缘节点	√	√
容器应用管理	支持下发容器应用到边缘节点	√	√
边云消息路由	提供边云消息通道，支持边云消息转发	√	√
多网络接入支持	支持Internet、VPN和专线接入	√	√
监控运维	支持监控运维	√	√
批量作业	支持批量创建容器应用、批量更新容器应用、批量注册边缘节点、批量升级边缘节点	√	√
边缘节点组	支持创建边缘节点组，将具备相同属性（如硬件架构）的多个边缘节点组成一个边缘节点组，以便统一化管理	×	√

功能特性	说明	专业版	铂金版
多实例支持	支持多个容器应用实例	×	√
独享集群	支持独享管理面集群	×	√
应用网格	支持服务发现；应用流量治理，包括负载均衡等多种治理能力	×	√
插件管理	支持插件管理能力。	√	√
Kubernetes原生接口开放	支持通过kubectl操作服务实例对应的Kubernetes集群	×	√

## 约束与限制

不支持服务实例就地扩容，即超出服务实例配额的边缘节点/应用只能通过新建服务实例进行管理，请提前结合自身业务合理规划并创建对应规模服务实例。

## 创建铂金版服务实例

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“总览”，单击页面右上角的“创建铂金版服务实例”。

**步骤3** 配置参数。

- **区域**：选择服务实例所在区域。不同的区域之间服务实例不互通，建议您选择最靠近您业务的区域，这样可以减少网络时延、提高访问速度。
- **实例名称**：填写实例名称。
- **终端节点服务白名单**：输入需要添加至白名单的用户domainID。
- **边云接入方式**：当前支持“互联网接入”和“专线接入”。专线连接IEF的具体方法请参见[通过专线或VPN连接IEF](#)。
- **边缘节点规模**：选择服务实例能管理的边缘节点规模。当前支持选择50、200、1000节点。
- **接入带宽**：接入方式为“互联网接入”时，根据边缘节点规模，分别对应为5Mbit/s、10Mbit/s、30Mbit/s。“专线接入”的带宽由专线决定。
- **高级设置**：多可用区部署，即铂金版服务实例部署在多个可用区，支持多可用区容灾，但是对于集群性能有所损耗。

**步骤4** 单击“下一步”。

**步骤5** 确认订单详情，单击“创建”。

服务实例创建需要20-30分钟，您可以在服务实例列表的“详情”中查看状态。

在“总览”页面可以查看到铂金版服务实例的信息。

图 2-1 查看实例信息



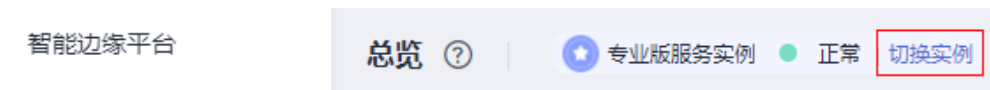
----结束

## 切换实例

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“总览”，单击页面右侧“切换实例”，如下图所示。

图 2-2 切换实例



**步骤3** 在需要切换的实例下，单击“选择实例”。

----结束

# 3 专业版操作指南

---

## 3.1 节点管理

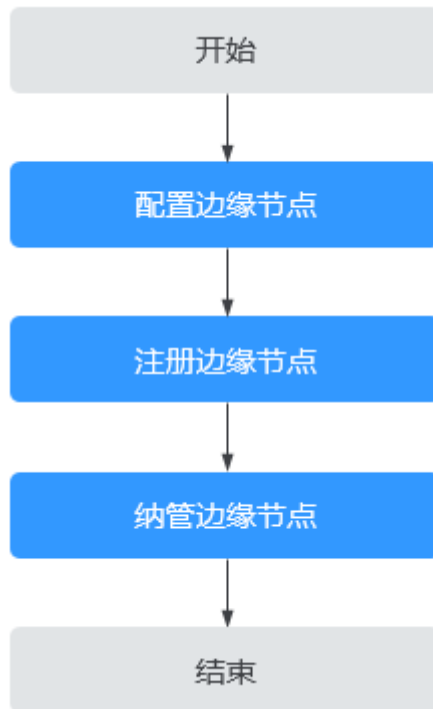
### 3.1.1 边缘节点概述

边缘节点是您自己的边缘计算机，用于运行边缘应用，处理您的数据，并安全、便捷地和云端应用进行协同。您可以通过智能边缘平台部署系统应用来延伸云服务能力到边缘节点，或者通过部署您自己的应用来构建您自己的边缘计算能力。

为了使智能边缘平台能够管理您的边缘节点，您需要做如下步骤：

1. 准备边缘节点，边缘节点需要满足一定的规格要求，具体请参见[配置边缘节点环境](#)。
2. 在IEF中创建边缘节点，获取边缘节点的安装工具和配置文件，具体请参见[注册边缘节点](#)。
3. 使用上一步获取的安装工具和配置文件纳管边缘节点，具体请参见[纳管边缘节点](#)。

图 3-1 使用流程



### 3.1.2 配置边缘节点环境

#### 边缘节点规格要求

边缘节点既可以是物理机，也可以是虚拟机。边缘节点需要满足表3-1的规格要求。

表 3-1 边缘节点要求

项目	规格
OS	<p>操作系统语言必须切换至英文。</p> <ul style="list-style-type: none"><li>x86_64架构 Ubuntu LTS (Xenial Xerus)、Ubuntu LTS (Bionic Beaver)、CentOS、EulerOS、RHEL、银河麒麟、中兴新支点、中标麒麟、openEuler、uos (Unity Operating System)、ol (Oracle Linux)、hce (Huawei Cloud Euler)、openEuler 23.09 Edge</li><li>armv7i (arm32) 架构 Raspbian GNU/Linux (stretch)</li><li>aarch64 (arm64) 架构 Ubuntu LTS (Bionic Beaver)、CentOS、EulerOS、openEuler、uos (Unity Operating System)、ol (Oracle Linux)、hce (Huawei Cloud Euler)、openEuler 23.09 Edge</li></ul> <p><b>说明</b> 推荐使用面向边缘计算场景的openEuler 23.09 Edge操作系统。</p>

项目	规格
内存	边缘软件开销约128MB，为保证业务的正常运行，建议边缘节点的内存大于256MB。
CPU	>= 1核
硬盘	>= 1GB
GPU（可选）	同一个边缘节点上的GPU型号必须相同。 <b>说明</b> 当前支持Nvidia Tesla系列P4、P40、T4等型号GPU。 含有GPU硬件的机器，作为边缘节点的时候可以不使用GPU。 如果边缘节点使用GPU，您需要在纳管前安装GPU驱动。 目前只有使用x86架构的GPU节点才能纳管到IEF中使用。
NPU（可选）	华为昇腾AI加速处理器。 <b>说明</b> 当前支持集成了华为昇腾处理器的边缘节点，如Atlas 300推理卡、Atlas 800推理服务器。同时支持昇腾310P、昇腾310B、昇腾310P共享模式和虚拟化切分的NPU规格。 如果边缘节点使用NPU，请确保边缘节点已安装驱动（目前昇腾310仅支持1.3.x.x和1.32.x.x的固件版本，例如1.3.2.B893，可用npusmi info命令查看固件版本）（NPU驱动需不小于22.0.4版本，进入驱动所在路径如“/usr/local/Ascend/driver”，执行cat version.info命令查看）。如果没有安装驱动，请联系设备厂商获取支持。
容器引擎	Docker版本必须高于17.06。使用高于或等于1.23版本的docker时，需设置docker cgroupfs版本为1，不支持docker HTTP API v2。 （请勿使用18.09.0版本Docker，该版本存在严重bug，详见 <a href="https://github.com/docker/for-linux/issues/543">https://github.com/docker/for-linux/issues/543</a> ；如果已使用此版本，请尽快升级。） <b>须知</b> Docker安装完成后，请将Docker进程配置为开机启动，避免系统重启后Docker进程未启动引起的系统异常。 Docker Cgroup Driver必须设置为cgroupfs。详细配置方法请参考 <a href="#">在边缘节点安装Docker后，如何设置Docker Cgroup Driver?</a> 。
glibc	版本必须高于2.17。
端口使用	边缘节点需要使用8883端口，8883端口用于边缘节点内置MQTT broker监听端口，请确保该端口能够正常使用。
时间同步	边缘节点时间需要与UTC标准时间保持一致，否则会导致边缘节点的监控数据、日志上传出现偏差。您可以选择合适的NTP服务器进行时间同步，从而保持时间一致。详细配置方法请参见 <a href="#">如何同步NTP服务器?</a> 。

## 配置边缘节点环境

**步骤1** 以具备sudo权限的用户登录边缘节点。

## 步骤2 GPU驱动配置。

如果边缘节点使用GPU，您需要安装并配置GPU驱动，详细方法请参见[安装并配置GPU驱动](#)。

## 步骤3 NPU驱动配置。

如果边缘节点使用华为昇腾 AI加速处理器，请确保已安装对应驱动。

## 步骤4 在边缘节点上安装Docker并检查Docker状态。

Docker版本必须高于17.06，推荐使用18.06.3版本。请勿使用18.09.0版本Docker，该版本存在严重bug，如果已使用此版本，请尽快升级。

Docker安装完成后，可以执行**docker -v**命令检查Docker是否安装正常，如果回显如下则说明安装正常。

```
# docker -v
Docker version 19.03.12, build 48a66213fee
```

## 步骤5 配置边缘节点防火墙规则。

检查边缘节点防火墙状态。

```
systemctl status firewalld
firewall-cmd --state
```

回显中，not running表示关闭，running表示开启。

如果防火墙开启，您需要打开8883端口，或关闭防火墙。

- 打开8883端口。

```
firewall-cmd --add-port=8883/tcp --permanent
systemctl restart firewalld
```
- 关闭防火墙。

```
systemctl disable firewalld
systemctl stop firewalld
```

----结束

## 安全提示

为提升主机安全性，建议您对边缘节点进行操作系统加固，可参考：

1. 设置所有OS系统口令（包括管理员和普通用户）、数据库账号口令、应用（WEB）系统管理账号口令为强口令，密码12位以上。
2. 应用程序不以管理员权限账号运行，应用程序（如Web）不使用数据库管理员权限账号与数据库交互。设置安全组，仅向公网开放必要端口，业务WEB控制台端口、局域网内部通信端口避免暴露在公网。关闭高危端口（如SSH端口），或采取限制允许访问端口的源IP、使用VPN/堡垒机建立的运维通道等措施消减风险。
3. 业务数据定期异地备份，避免黑客入侵主机造成数据丢失。
4. 定期检测系统和软件中的安全漏洞，及时更新系统安全补丁，将软件版本升级到官方最新版本。
5. 建议从官方渠道下载安装软件，对非官方渠道下载的软件，建议使用杀毒软件扫描后再运行。

如果使用的是华为云 ECS，可参考：

1. 将主机登录方式设置为密钥登录。



2. 使用华为云官方提供的[企业主机安全服务](#)深度防御。

### 3.1.3 注册边缘节点

注册边缘节点就是在IEF设定边缘节点的配置，并获取边缘节点配置文件和安装程序。

#### 约束与限制

- 同一节点上的NPU，只支持同一种切分规格。
- 暂不支持已纳管的节点，通过升级NPU插件到2.1.0版本的方式支持虚拟化切分。用户需要将边缘节点进行卸载重装，手动切分NPU后，再纳管至IEF。
- NPU驱动需大于22.0版本，进入驱动所在路径（如“/usr/local/Ascend/driver”），执行cat version.info命令查看。
- 选择共享模式的情况下，需要在NPU卡上设置所有芯片的容器共享模式。

#### 注册边缘节点

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“边缘资源 > 边缘节点”，单击页面右上角的“注册边缘节点”。

**步骤3** 配置边缘节点基本信息。

图 3-2 边缘节点基本信息（1）

The screenshot shows the 'Register Edge Node' configuration page in the IEF console. It includes the following elements:

- Service Instance:** A radio button selected for 'Professional Service Instance'.
- Name:** A text input field with the placeholder '请输入边缘节点名称'.
- Description:** A text area with the placeholder '选填, 请输入边缘节点描述' and a character count '0/255'.
- Tags:** Two input fields for '请输入标签名' and '请输入标签值', with a note '还可以创建20个标签'.
- AI Acceleration Card:** A section with a tip: '如需部署昇腾应用或者GPU应用, 请根据选择切换AI加速卡类型 (了解边缘节点规格要求)'. Below are three tabs: '不启用' (selected), '昇腾AI加速卡', and 'Nvidia GPU'.
- Binding Device:** A table with columns '设备', '设备与节点的关系', '备注', and '操作'. A '+ 绑定设备' button is below the table.
- Whether to use Docker:** A dropdown menu set to '是'.
- Listening Address:** A section with two tabs: '网卡' (selected) and 'IP'. Below are two rows of listening addresses: 'tts:// lo : 8883' and 'tts:// docker0 : 8883', each with a delete icon. A '+ 新增监听地址' button is at the bottom.

- **名称:** 边缘节点的名称。允许中文、英文字母、数字、中划线、下划线，最小长度1，最大长度64。
- **描述:** 选填，请输入边缘节点描述信息。
- **标签:** 标签可用于对资源进行标记，方便分类管理。
- **AI加速卡**

昇腾AI加速卡：支持华为昇腾处理器（即NPU）的边缘节点。如果使用华为昇腾310、310B等芯片，请选择“昇腾AI加速卡”，然后选择NPU类型。昇腾AI加速卡支持的NPU类型，如下表3-2。

表 3-2 NPU 类型说明

类型	描述
昇腾310	昇腾310芯片
昇腾310B	昇腾310B芯片

Nvidia GPU：如果您的边缘节点搭载了Nvidia GPU显卡，请选择“Nvidia GPU”。

不启用：边缘节点未使用AI加速卡时选择。

#### 📖 说明

如果边缘节点上没有搭载Nvidia GPU显卡，而这里选择了启用“Nvidia GPU”，则纳管边缘节点会失败。

如果边缘节点使用GPU，您需要在纳管前安装并配置GPU驱动，详细方法请参见[安装并配置GPU驱动](#)。

- **绑定设备**：为边缘节点绑定终端设备，如果您还没有注册终端设备，请参见[终端设备管理](#)注册终端设备。终端设备在注册边缘节点后仍然可以绑定。
- **是否启用docker**：启用后可以支持部署容器应用。
- **监听地址**：

边缘节点内置的MQTT broker的监听地址，用于发送和接收边云消息。边云消息的使用请参见[边云消息概述](#)。

默认监听lo（localhost）和docker0两个本地网卡，您可以通过指定网卡名或IP地址设置需要监听的网卡，还可以增加其他需要监听的网卡或IP地址。

图 3-3 边缘节点基本信息（2）

📌 应用运维管理服务AOM每月赠送500M免费日志存储额度，超出部分将按需收费 [\(了解计费详情\)](#)

系统日志    应用日志

请合理设置日志文件大小和滚动数量避免过多占用节点存储。

日志文件大小(MB) ?

滚动日志周期 ?

滚动日志数量 ?

是否开启云端日志

当前支持配置边缘节点的系统日志和应用日志。

- **系统日志**：边缘节点上IEF软件（如edge-core、edge-logger和edge-monitor等）产生的日志。
- **应用日志**：边缘节点上部署的应用所产生的日志。

系统日志和应用日志需要配置如下几个参数：

- **日志文件大小**：日志文件大小限制，单位MB，默认50，取值范围10-1000。某个日志文件如果达到大小限制，则会转储。
  - 系统日志保存在边缘节点“/var/IEF/sys/log/”目录下，然后转储到AOM。
  - 应用日志会将容器的标准输出和挂载到边缘节点“/var/IEF/app/log”的日志转储到AOM。
- **滚动日志周期**：日志转储周期，可选项：每天、每周、每月、每年。日志文件大小和滚动日志周期是同时生效的，满足任何一个条件都会进行日志转储。
- **滚动日志数量**：日志文件转储个数，默认5，取值范围1-10。边缘节点保存的转储日志数量如果达到限制，则会删除最老的那个转储文件。
- **是否开启云端日志**：  
您可以根据需要控制是否上传日志到AOM服务，开启之后您可以在AOM中查看日志，具体请参见[在AOM查看日志](#)。

**步骤4** 勾选“我已经阅读并同意《华为云服务等级协议》”，单击页面右下角的“注册”，节点注册可选“通过证书注册”和“通过token注册”两种方式。

通过证书注册。下载配置文件和边缘节点安装工具，在后续[纳管边缘节点](#)时将用到这些。

图 3-4 下载配置文件和边缘核心软件

### 请下载软件并在边缘节点完成以下步骤

以下操作将节点连接到智能边缘平台。您必须现在下载配置文件，稍后将无法找回。



1. 根据页面提示，单击“下载 边缘节点名称.tar.gz 配置文件”下载配置文件。
2. 根据您边缘节点的CPU架构选择边缘节点安装工具，单击“下载EdgeCore Installer”。

**步骤5** 在右下角勾选“我已完成下载”，并单击“完成”。

您可以看到边缘节点的状态为“未纳管”，这是因为还未安装[注册边缘节点](#)下载的边缘节点安装工具，请参见[纳管边缘节点](#)纳管节点。

图 3-5 未纳管的边缘节点

<input type="checkbox"/>	名称/ID	状态	主机名/网络	应用实例(正常/全部)	创建时间	边缘侧软件版本	节点标签	节点类型	操作
<input type="checkbox"/>	ief-node 18c37006-8bbb-4a9a-9e5	未纳管 安装指南	--	0/0	2021/07/19 18:39:55 GMT...	--	--	自建节点	删除   更多 ▾

----结束

## 后续操作

完成注册后，您需要对边缘节点进行纳管，具体请参见[纳管边缘节点](#)。

### 3.1.4 纳管边缘节点

纳管边缘节点就是在实际的边缘节点上使用[注册边缘节点](#)中下载的安装程序和配置文件，安装边缘核心软件EdgeCore，这样边缘节点就能与IEF连接，纳入IEF管理。

边缘节点初次纳管时，智能边缘平台自动安装最新版本的边缘核心软件EdgeCore。例如，当前IEF边缘软件有2.51.0、2.52.0、2.53.0三个版本，当您初次纳管节点时，IEF将推送最新版本2.53.0给您的边缘节点进行安装。

#### 说明

在IEF上注册的边缘节点与实际的边缘节点机器是一一对应的关系，一个边缘节点的安装工具和配置文件只能安装在一台实际的边缘节点上。

## 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[配置边缘节点环境](#)。
- 已经注册好节点并获取到节点配置文件和安装工具，具体请参见[注册边缘节点](#)。

## 纳管边缘节点

**步骤1** 以具备sudo权限的用户登录边缘节点。

**步骤2** 将[注册边缘节点](#)下载的边缘节点安装工具和配置文件上传到边缘节点指定目录，例如“/home”目录，并进入该目录。

**步骤3** 执行如下命令，解压缩安装工具到“/opt”文件夹。

```
sudo tar -zxvf edge-installer_1.0.0_x86_64.tar.gz -C /opt
```

`edge-installer_1.0.0_x86_64.tar.gz`请替换为[注册边缘节点](#)下载的安装工具。

**步骤4** 执行如下命令，解压缩配置文件到“opt/IEF/Cert”目录。如果纳管的是“通过token注册”的边缘节点，请跳过此步骤。

```
sudo mkdir -p /opt/IEF/Cert; sudo tar -zxvf 边缘节点名称.tar.gz -C /opt/IEF/Cert
```

`边缘节点名称.tar.gz`请替换为[注册边缘节点](#)下载的配置文件的名称。

**步骤5** 执行如下命令，纳管边缘节点。

- 通过证书注册

```
cd /opt/edge-installer; sudo ./installer -op=install
```

- 通过token注册

```
cd /opt/edge-installer; sudo ./installer -op=install -identifier=token注册的凭证
```

*token*注册的凭证请替换为[注册边缘节点](#)保存的安装凭证*identifier*字段。

**步骤6** 验证边缘节点是否纳管成功。

1. 登录IEF管理控制台。
2. 选择左侧导航栏的“边缘资源 > 边缘节点”。

3. 查看边缘节点的状态。当前状态为“运行中”表示纳管成功。

图 3-6 查看边缘节点状态

<input type="checkbox"/>	名称/ID	状态	主机名/网络	应用实例(正常/全部)	创建时间	边缘侧软件版本
<input type="checkbox"/>	ief-node 7092ad14-adee-4a09-b969-1	运行中	eth0:192.168.0.230	0/0	2021/07/20 09:33:26 GMT+08:00	2.52.0 <a href="#">可升级</a>

----结束

### 须知

纳管后请勿删除边缘节点的“/opt”目录，否则需要重新注册边缘节点并纳管。

## 3.1.5 升级边缘节点

### 背景信息

边缘节点上安装的EdgeCore软件支持升级，IEF会不定期发布新版本，您可以根据需求升级边缘节点。

### 版本支持策略

IEF只负责维护发布周期一年内的边缘节点软件版本，建议您的边缘节点每年至少升级一次。

### 版本升级规范

边缘节点升级时，智能边缘平台自动选择最新版本的EdgeCore在边缘节点进行升级。

例如，当前EdgeCore有2.22.0、2.23.0、2.24.0三个版本，而您的边缘节点上EdgeCore版本为2.12.0，当您升级边缘节点时，IEF将推送最新版本2.24.0给您的边缘节点进行升级。

### 注意事项

- 为了让您的边缘节点应用更稳定可靠的运行，IEF不会主动升级您的边缘节点上的EdgeCore，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。
- 处于维护周期中的版本升级，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。
- 处于维护周期外的版本升级，可能会因为容器重启引起业务的短暂中断。
- 请勿在节点升级过程中变更节点配置，比如重启Docker、安装卸载GPU/NPU驱动、OS内核升级、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

- 步骤1 登录边缘节点，配置防火墙规则。

检查边缘节点防火墙状态。

```
systemctl status firewalld  
firewall-cmd --state
```

回显中，not running表示关闭，running表示开启。

如果防火墙开启，您需要打开8883端口，或关闭防火墙。

- 打开8883端口。  
firewall-cmd --add-port=8883/tcp --permanent  
systemctl restart firewalld
- 关闭防火墙。  
systemctl disable firewalld  
systemctl stop firewalld

**步骤2** 登录IEF管理控制台。

**步骤3** 选择左侧导航栏的“边缘资源 > 边缘节点”。

**步骤4** 在“边缘侧软件版本”列查看是否可以升级。

仅处于“运行中”状态的边缘节点才可以升级。

- 如果显示可升级，表示可以升级。
- 如果没有显示，请查看边缘节点是否处于“运行中”状态。如果边缘节点处于“运行中”且无显示，则说明当前边缘节点EdgeCore是最新版本。

**图 3-7** 查看边缘节点是否可以升级

名称/ID	状态	主机名/网络	应用实例(正常/全部)	创建时间	边缘侧软件版本	节点标签	节点类型
ief-node 7092ad14-adee-4a09-b969-1505bbdecef5	运行中	eth0.192.168.0.230	0/0	2021/07/20 09:33:26 GMT+08:00	2.52.0 可升级	--	自建节点

**步骤5** 单击“更多 > 升级”。

**图 3-8** 升级边缘节点

名称/ID	状态	主机名/网络	应用实例(正常/全部)	创建时间	边缘侧软件版本	节点标签	节点类型	操作
ief-node 7092ad14-adee-4a09-b969-1505bbdecef5	运行中	eth0.192.168.0.230	0/0	2021/07/20 09:33:26 GMT+08:00	2.52.0 可升级	--	自建节点	删除 更多 启用 停用 查看监控 升级

**步骤6** 单击节点名称进入节点详情页面，可以查看详细升级记录。

图 3-9 升级记录



----结束

## 3.1.6 日志、监控和告警

### 日志说明

边缘节点会上传系统日志和应用日志，您需要在IEF控制台上打开日志开关。

- **系统日志**：边缘节点上IEF软件（如edge-core、edge-logger和edge-monitor等）产生的日志。
- **应用日志**：边缘节点上部署的应用所产生的日志。
  - 边缘节点会上传“/var/IEF/app/log”目录的日志，您可以在创建应用时将容器中目录挂载到“/var/IEF/app/log/{appName}”下，具体挂载方法请参见 **hostPath：将主机某个目录挂载到容器中**。在AOM中可以按{appName}分类查看到应用的日志。
  - 边缘节点会上传容器日志，日志组件会上传“{{DOCKER\_ROOT\_DIR}}/containers/{containerID}/{containerID}-json.log”文件的内容，DOCKER\_ROOT\_DIR可以通过**docker info**命令查询到，containerID就是容器ID。

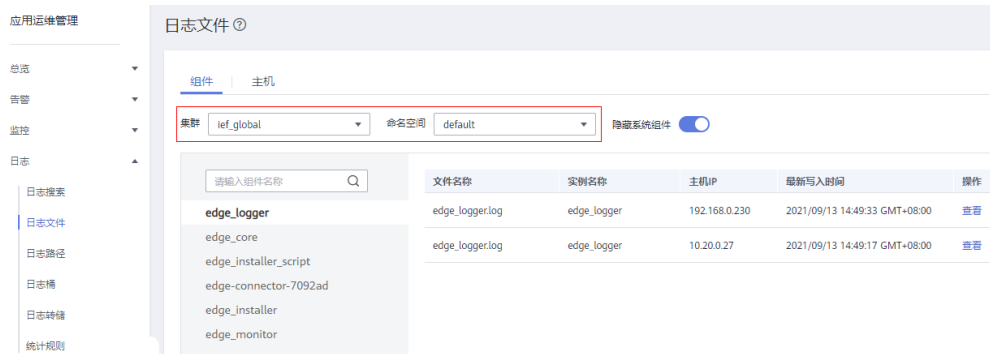
### 在 AOM 查看日志

**步骤1** 登录AOM管理控制台。

**步骤2** 在左侧导航栏选择“日志 > 日志文件”，单击“组件”页签。

**步骤3** 选择集群“ief\_global”和命名空间“default”。

图 3-10 选择集群和命名空间



步骤4 搜索应用名称，单击日志文件右侧的“查看”，即可查看详细日志。

----结束

## 在 AOM 中查看节点监控信息

您可以在AOM查看节点监控信息。

步骤1 登录AOM管理控制台。

步骤2 选择监控的节点。

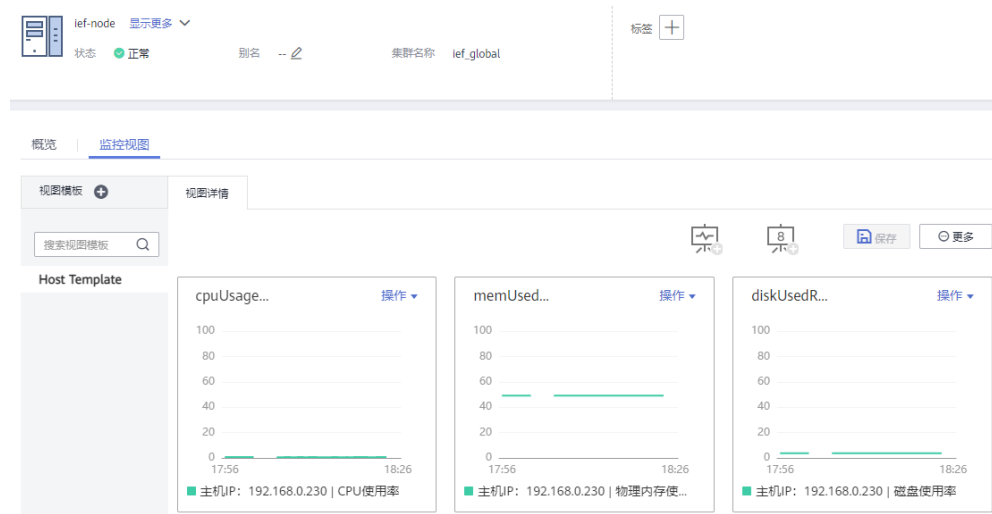
图 3-11 选择监控节点



步骤3 单击节点名称，在“监控视图”页签下，您可以查看节点的资源使用情况，如CPU、内存的使用率等。



图 3-12 查看监控信息



----结束

## 在 AOM 中查看容器监控信息

AOM中可以查看边缘节点上容器应用的监控信息。

**步骤1** 登录AOM管理控制台。

**步骤2** 选择要监控的容器工作负载。

图 3-13 选择工作负载

The screenshot shows the '容器监控' (Container Monitoring) page. It features a table with columns for '工作负载' (Workload), '名称' (Name), '状态' (Status), '所属集群' (Cluster), '命名空间' (Namespace), '物理内存容量(MB)' (Physical Memory Capacity), 'CPU使用率(%)' (CPU Usage), and '内存使用率(%)' (Memory Usage). The table lists several workloads, all with a status of '正常' (Normal). The first row shows a workload named 'y-name' in the 'ief\_global' cluster, namespace 'default', with a physical memory capacity of '-MB', CPU usage of '-%', and memory usage of '-%'. The table also includes filter options for status, cluster, namespace, and a '添加筛选条件' (Add Filter) button.

**步骤3** 单击工作负载名称，进入详情页面，在“监控视图”页签下，您可以设置容器的监控指标，如CPU、内存的使用率等。

图 3-14 查看监控信息



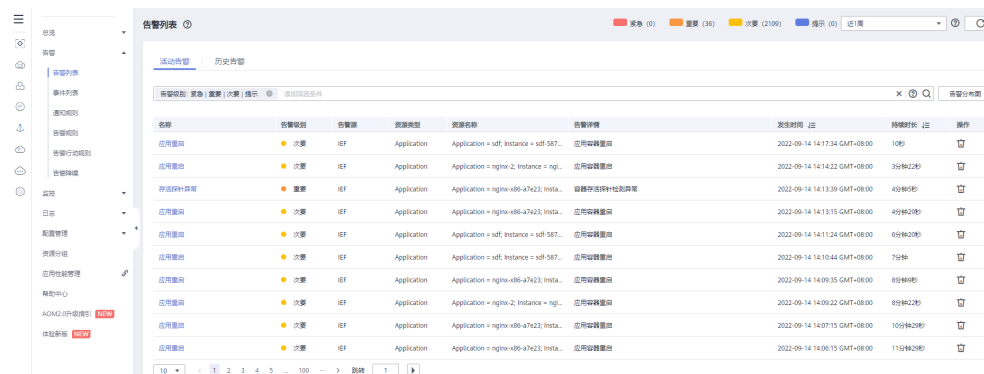
----结束

## IEF 预置的告警

IEF为每个边缘节点预置了7个告警规则，这7类告警会自动上报到AOM。

告警名称	触发条件	清除条件	告警等级
容器引擎异常	边缘节点配置Docker使能时，查询Docker信息失败	Docker正常运行，EdgeCore能够获取到Docker信息	紧急
存活探针异常	应用配置存活探针，探针检测到异常	容器探针检测成功	重要
申请GPU资源失败	部署GPU应用，申请GPU资源失败	成功申请到GPU资源	紧急
获取GPU信息失败	边缘节点配置GPU使能时，查询GPU信息失败	成功查询到GPU信息	紧急
AK/SK无效	EdgeHub连续10次分发临时AK/SK，检测到过期或者状态异常	EdgeHub成功分发临时AK/SK	重要
应用重启	应用容器异常重启	无需清除	次要
容器绑定网卡异常	容器绑定的网卡发生异常	容器绑定的网卡状态正常	紧急

图 3-15 查看告警



名称	告警级别	告警源	资源类型	资源名称	告警详情	发生时间	持续时间	操作
应用健康	次要	IEF	Application	Application = sdf; Instance = sdf-587...	应用健康异常	2022-09-14 14:17:34 GMT+08:00	10秒	
应用健康	次要	IEF	Application	Application = nginx-2; Instance = ngi...	应用健康异常	2022-09-14 14:14:22 GMT+08:00	29分22秒	
应用健康	次要	IEF	Application	Application = nginx-a86-a7a23; Insta...	应用健康异常	2022-09-14 14:13:39 GMT+08:00	49分58秒	
应用健康	次要	IEF	Application	Application = nginx-a86-a7a23; Insta...	应用健康异常	2022-09-14 14:13:15 GMT+08:00	49分20秒	
应用健康	次要	IEF	Application	Application = sdf; Instance = sdf-587...	应用健康异常	2022-09-14 14:11:24 GMT+08:00	69分20秒	
应用健康	次要	IEF	Application	Application = sdf; Instance = sdf-587...	应用健康异常	2022-09-14 14:10:44 GMT+08:00	79秒	
应用健康	次要	IEF	Application	Application = nginx-a86-a7a23; Insta...	应用健康异常	2022-09-14 14:09:35 GMT+08:00	89分08秒	
应用健康	次要	IEF	Application	Application = nginx-2; Instance = ngi...	应用健康异常	2022-09-14 14:09:22 GMT+08:00	89分02秒	
应用健康	次要	IEF	Application	Application = nginx-a86-a7a23; Insta...	应用健康异常	2022-09-14 14:07:15 GMT+08:00	100分20秒	
应用健康	次要	IEF	Application	Application = nginx-a86-a7a23; Insta...	应用健康异常	2022-09-14 14:06:15 GMT+08:00	11分02秒	

## 在 AOM 中设置告警

您可以在AOM中创建告警规则来监控边缘节点上的各项指标，请参考[创建阈值规则](#)进行设置。

## 上报自定义告警到 AOM

IEF支持从边缘节点上报自定义告警到AOM，使用MQTT客户端发布告警信息到MQTT broker，IEF会将告警自动上报到AOM。

具体请参见[添加告警](#)和[清除告警](#)。

## 3.1.7 安装并配置 GPU 驱动

### 背景信息

对于使用GPU的边缘节点，在纳管边缘节点前，需要安装并配置GPU驱动。

IEF当前支持Nvidia Tesla系列P4、P40、T4等型号GPU，支持CUDA Toolkit 8.0至10.0版本对应的驱动。

### 操作步骤

#### 步骤1 安装GPU驱动。

1. 下载GPU驱动，推荐驱动链接：

[https://www.nvidia.com/content/DriverDownload-March2009/confirmation.php?url=/tesla/440.33.01/NVIDIA-Linux-x86\\_64-440.33.01.run&lang=us&type=Tesla](https://www.nvidia.com/content/DriverDownload-March2009/confirmation.php?url=/tesla/440.33.01/NVIDIA-Linux-x86_64-440.33.01.run&lang=us&type=Tesla)

2. 执行如下安装驱动命令。

```
bash NVIDIA-Linux-x86_64-440.33.01.run
```

3. 执行如下命令检查GPU驱动安装状态。

```
nvidia-smi
```

#### 步骤2 以root用户登录边缘节点。

#### 步骤3 执行如下命令。

```
nvidia-modprobe -c0 -u
```

**步骤4** 创建文件夹。

```
mkdir -p /var/IEF/nvidia/drivers /var/IEF/nvidia/bin /var/IEF/nvidia/lib64
```

**步骤5** 拷贝驱动文件。

- 对于CentOS，依次执行如下命令拷贝驱动文件：

```
cp /lib/modules/{当前环境内核版本号}/kernel/drivers/video/nvi* /var/IEF/nvidia/drivers/
```

```
cp /usr/bin/nvidia-* /var/IEF/nvidia/bin/
```

```
cp -rd /usr/lib64/libcuda* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/libEG* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/libGL* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/libnv* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/libOpen* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/libvdpau_nvidia* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/vdpau /var/IEF/nvidia/lib64/
```

- 对于Ubuntu，依次执行如下命令拷贝驱动文件：

```
cp /lib/modules/{当前环境内核版本号}/kernel/drivers/video/nvi* /var/IEF/nvidia/drivers/
```

```
cp /usr/bin/nvidia-* /var/IEF/nvidia/bin/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libcuda* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libEG* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libGL* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libnv* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libOpen* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libvdpau_nvidia* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/vdpau /var/IEF/nvidia/lib64/
```

其中，当前环境内核版本号可以使用`uname -r`命令查看获取，如下所示，请替换为实际取值。

```
# uname -r  
3.10.0-514.e17.x86_64
```

**步骤6** 执行以下命令修改目录权限。

```
chmod -R 755 /var/IEF
```

----结束

## 3.1.8 边缘核心软件 EdgeCore 配置管理

### 操作场景

IEF边缘软件支持对EdgeCore配置参数进行管理，通过该功能您可以对边缘核心软件EdgeCore进行深度配置。

## 操作步骤

**步骤1** 在边缘节点上执行如下命令修改EdgeCore配置，并保存。

```
vi /opt/IEF/Edge-core/conf/edge.yaml
```

支持配置的参数如下表所示：

表 3-3 参数说明

组件	参数	说明	取值
edge-core	interface-name	网卡名称	默认：eth0
	internal-server	内置mqtt broker监听地址	tls://lo:8883,tls://docker0:8883
	image-gc-high-threshold	触发镜像垃圾回收的磁盘使用率百分比	默认：80
	image-gc-low-threshold	镜像垃圾回收试图释放资源后达到的磁盘使用率百分比	默认：40
	swr-url	拉取镜像的代理地址	默认：""

**步骤2** 更改完配置之后，重启EdgeCore。

```
systemctl restart edgecore
```

----结束

### 3.1.9 删除边缘节点

#### 前提条件

删除边缘节点前，需要先解绑终端设备、删除边缘节点上的应用和证书。

#### 操作步骤

**步骤1** 以可运行sudo命令的用户登录边缘节点。

**步骤2** 执行以下命令卸载已纳管节点上的软件和配置文件。

```
cd /opt/edge-installer; sudo ./installer -op=uninstall
```

**步骤3** 登录IEF管理控制台。

**步骤4** 选择左侧导航栏的“边缘资源 > 边缘节点”。

**步骤5** 单击待删除边缘节点右侧的“更多 > 删除”。

**步骤6** 根据提示完成删除操作。

----结束

## 3.2 终端设备管理

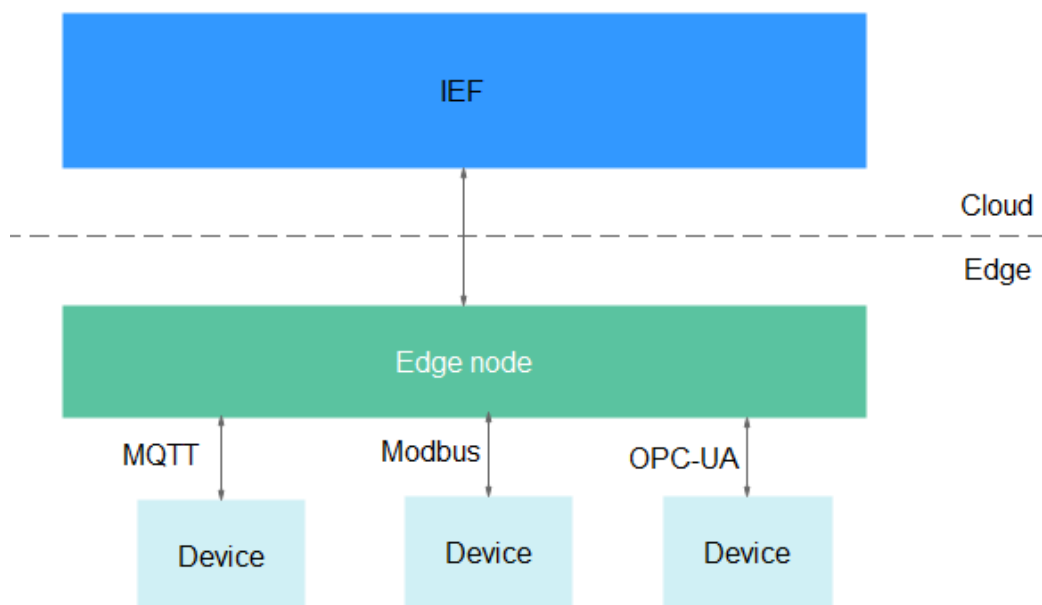
### 3.2.1 终端设备与设备孪生

#### 终端设备

终端设备可以小到传感器、控制器，大到智能摄像机或数控机床。

终端设备可以连接到边缘节点，终端设备支持通过MQTT协议接入。终端设备接入后，可以在IEF中对终端设备进行统一管理。

图 3-16 终端设备管理



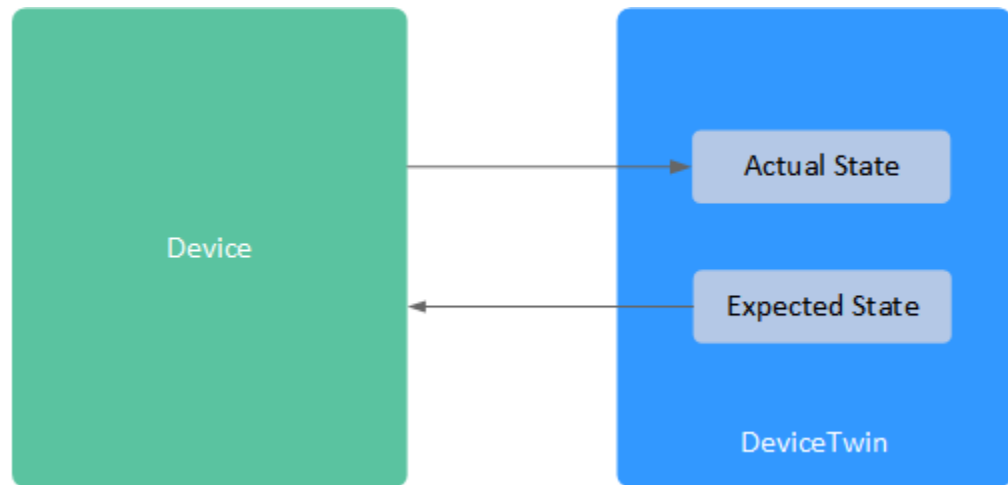
#### 设备孪生（DeviceTwin）

终端设备通常包含两类数据：

- 一是不会改变的元数据，包括序列号、资产标识符、Mac地址等描述设备信息的数据。这种数据也可以称为终端设备的静态属性或**设备属性**。
- 另一类是终端设备的动态数据，包括特定背景下的终端设备专有实时数据，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Expected State（期望的状态）进行状态更新，此外终端设备实时反馈自身的Actual State（真实的状态），设备孪生同时记录终端设备的Actual State和Expected State。这种方式也使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。

图 3-17 DeviceTwin



在IEF中可以创建终端设备，并能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属性和孪生信息。边缘节点上的应用程序可在边缘节点获取终端设备属性、设备孪生信息、以及修改终端设备孪生期望值和真实值。同时IEF负责同步云、边的孪生信息，当有冲突时，将以边缘侧的修改为主。

详细的终端设备状态边云协同机制请参见[设备孪生工作原理](#)。

## 使用流程

使用IEF管理和控制终端设备，通常使用的步骤如下：

1. 定义终端设备模板（包含设备属性、孪生属性）。
2. 使用模板创建设备。  
您也可以不使用模板，直接创建设备。
3. 将终端设备关联到边缘节点。
4. 在IEF中管理和控制终端设备，监测终端设备状态。

### 3.2.2 设备模板

边缘计算场景下，通常有数量庞大的终端设备，在IEF中可以将一类终端设备定义成统一的模板，这样可以通过模板创建终端设备。比如您可以创建一个自定义的终端设备模板“Camera”，创建终端设备时可以使用该模板的所有属性，而不必为每一个类似的终端设备设置同样的属性。

#### 创建设备模板

- 步骤1** 登录IEF管理控制台。
- 步骤2** 选择左侧导航栏“边缘资源 > 终端设备”，单击页面右上角的“创建设备模板”。
- 步骤3** 填写模板名称，选择访问协议，设置设备属性、孪生属性、设备标签和描述。

图 3-18 创建设备模板

服务实例  专业版服务实例

\*名称

访问协议  MQTT

描述

0/255

---

模板属性

**注意：**设备属性会明文展示所输入信息，请不要填入敏感信息，如涉及敏感信息，请先加密，请防止信息泄露。

属性名	类型	属性值	是否可选	操作
+ 新增属性				

孪生属性

**注意：**设备孪生属性会明文展示所输入信息，请不要填入敏感信息，如涉及敏感信息，请先加密，请防止信息泄露。

属性名	类型	属性值	操作
+ 新增孪生属性			

标签

还可以创建20个标签。

- **名称：**终端设备模板的名称。
- **访问协议：**IEF支持MQTT协议。
- **描述：**输入对终端设备的描述信息。
- **模板属性：**  
属性是键值对形式，请输入属性名和属性值，并选择类型。  
通常将不会改变的元数据，例如序列号、资产标识符、Mac地址之类的信息设置为模板属性。
- **孪生属性：**  
通常将终端设备的动态数据，例如特定背景下的设备专有实时数据，例如灯的开、关状态等设置为孪生属性。
  - MQTT协议：MQTT协议的孪生属性是键值对形式，请输入属性名和属性值，并选择类型。

#### 须知

IEF不提供任何加解密工具，对您配置的设备属性值不感知，如果设备属性值设置为加解密文，需要您自行解密。

- **标签：**标签用于为终端设备分类，标签可以帮助您更加快速的搜索到想要的终端设备。您可以为终端设备设置标签，方便分类管理。

**步骤4** 单击“创建”，即创建设备模板成功，返回到设备模板页面。

----结束



### 3.2.3 终端设备

终端设备可以连接到边缘节点，终端设备支持通过MQTT协议接入。终端设备接入后，可以在IEF中对终端设备进行统一管理。

#### 注册终端设备

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏“边缘资源 > 终端设备”，单击页面右上角的“注册终端设备”。

**步骤3** 填写设备参数。

- **名称**：终端设备的名称。
- **ID**：您可以选择自定义设备ID，或者让IEF自动为您生成ID。如果需要自定义设备ID，请勾选“自定义设备ID”并填写设备ID。
- **访问协议**：IEF支持MQTT协议。
- **描述**：输入对终端设备的描述信息。
- **设备配置**：选择已经创建的设备模板，为设备自动添加属性。模板中定义了设备属性、孪生属性、设备标签信息。您也可以不使用模板，直接手动为设备添加属性和标签，其含义与模板中定义一致，具体请参见[设备模板](#)。

**步骤4** 单击“注册”，即注册终端设备成功，返回到终端设备列表页面。

----结束

#### 后续操作

您可以将终端设备添加到边缘节点中，具体请参见[终端设备绑定到边缘节点](#)。

### 3.2.4 终端设备绑定到边缘节点

一个边缘节点可以绑定多个终端设备，但一个终端设备只可以被绑定于一个边缘节点。通过绑定终端设备到固定的边缘节点，您可以在边缘节点部署相应的应用，实现管理终端设备和监控终端设备状态等功能。

#### 绑定边缘节点

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“边缘资源 > 终端设备”。

**步骤3** 在终端设备所在行右侧单击“绑定节点”。

**步骤4** 填写终端设备与节点的关系，选择要绑定的边缘节点，单击“确定”。

----结束

#### 从边缘节点绑定终端设备

您还可以在边缘节点上操作绑定终端设备。

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“边缘资源 > 边缘节点”，单击对应节点进入边缘节点详情页。

**步骤3** 选择“设备”页签，单击“绑定设备”。

**步骤4** 在弹出的窗口中勾选需要绑定的设备，并填写终端设备与节点的关系，单击“确定”。

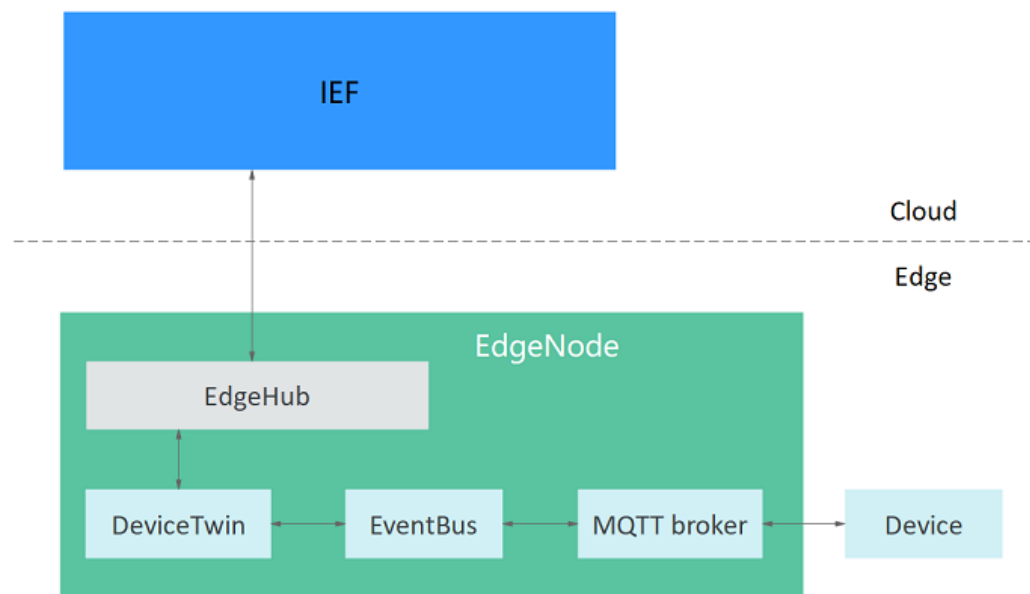
----结束

### 3.2.5 设备孪生工作原理

边缘节点纳管后，会在边缘节点上安装Edge Agent，其中终端设备管理相关组件如下所示。

- EdgeHub：WebSocket客户端，包括同步云端资源更新、报告边缘节点和终端设备信息到云端等功能。
- DeviceTwin：设备孪生，负责存储终端设备状态并将设备状态同步到云端。
- EventBus：与MQTT服务器交互的客户端，为其他组件提供订阅和发布消息的功能。
- MQTT broker：MQTT服务器。

图 3-19 终端设备管理



终端设备、边缘节点、IEF通信的过程中，设备孪生（DeviceTwin）起到了一个非常重要的作用，设备孪生保持设备的动态数据，包括特定背景下的设备专有实时数据，例如灯的开、关状态。

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Expected State（期望的状态）进行状态更新，此外终端设备实时反馈自身的Actual State（真实的状态），设备孪生同时记录设备的Actual State和Expected State。这种方式也使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。

设备孪生的参考示例如下。

```
{  
  "device": {
```

```
"id": "989e4fc8-9f24-44d7-9f9c-a0bd3bfb1949",  
"description": "my home light",  
"name": "light",  
"state": "online",  
"twin": {  
  "powerstatus": {  
    "expected": {  
      "value": "ON"  
    },  
    "actual": {  
      "value": "OFF"  
    },  
    "metadata": {  
      "type": "string"  
    }  
  }  
}
```

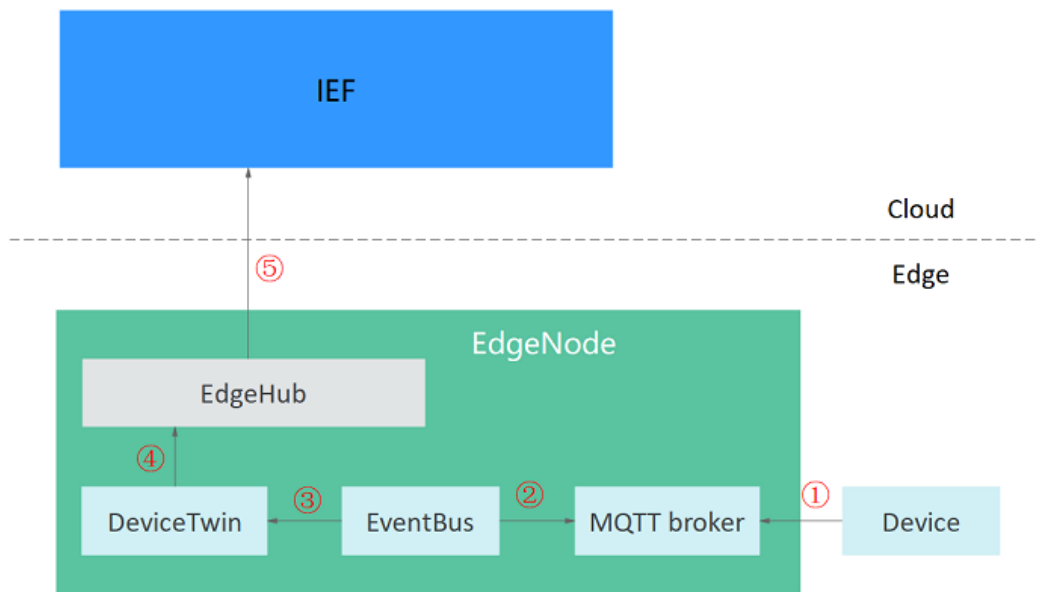
这个名为“light”的设备定义一个名为“powerstatus”的孪生属性，期望值（expected）是ON，而实际值（actual）是OFF。

下面看下这个终端设备与边缘节点和IEF通信的过程。

### 终端设备上报实际状态到云端

终端设备上报实际状态到云端的过程如图3-20所示。

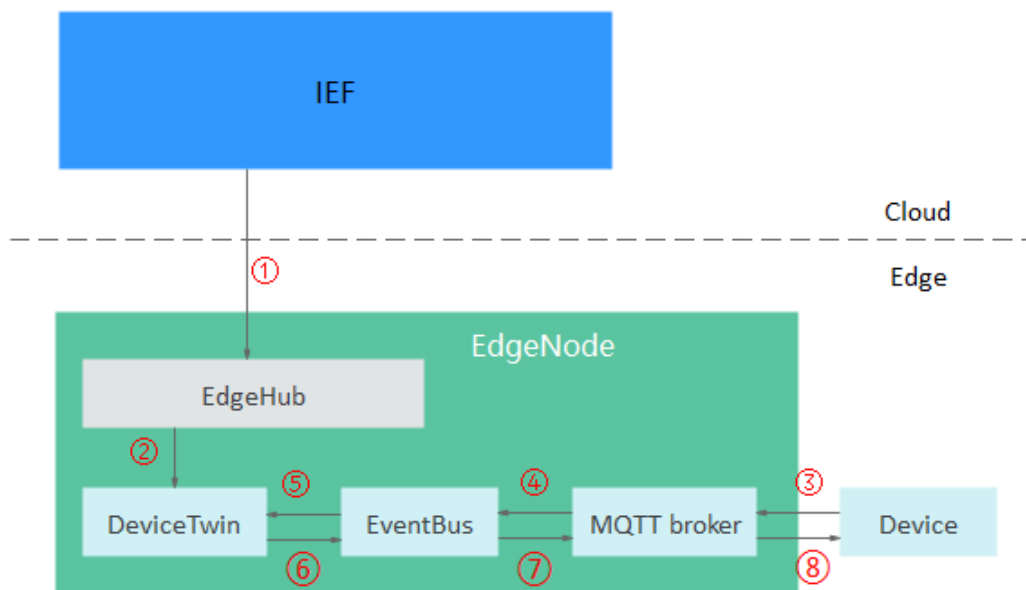
图 3-20 设备上报状态



1. 终端设备将实际状态（Actual State）实时上报给MQTT broker。
2. EventBus从MQTT broker收到订阅消息，消息内容包含终端设备的实际状态。
3. EventBus把终端设备实际状态发送给DeviceTwin，DeviceTwin在边缘节点存储终端设备实际状态。
4. DeviceTwin同步实际状态给WebSocket客户端EdgeHub。
5. EdgeHub发送消息给IEF。

## 云端修改孪生属性控制终端设备状态

图 3-21 修改终端设备状态



1. 在IEF中修改终端设备的孪生属性，IEF将终端设备期望状态（Expected State）发送给边缘节点的EdgeHub。
2. EdgeHub发送终端设备期望状态消息到DeviceTwin，DeviceTwin在边缘节点存储终端设备期望状态。
3. 终端设备实时发消息给MQTT broker查询终端设备期望状态。
4. EventBus接收到从MQTT broker发过来的消息。
5. EventBus根据消息去查询终端设备期望状态。
6. DeviceTwin反馈当前终端设备期望状态给EventBus。
7. EventBus发送设终端备期望状态的结果给MQTT broker。
8. 终端设备从MQTT broker收到订阅消息，根据期望状态调整实际状态。

### 3.2.6 设备数据上云

#### MQTT broker

终端设备可以通过MQTT协议与IEF云端进行通信，您也可以通过发送/订阅消息控制终端设备。

边缘节点上有一个**内置MQTT broker**，内置MQTT broker使用8883端口与终端设备通信，与内置MQTT broker通信需要经过安全认证，具体请参见[使用证书进行安全认证](#)。

另外，边缘节点还支持与**外置MQTT broker**通信，即在边缘节点上安装一个MQTT broker（如开源的Mosquitto，默认使用1883端口通信）。

#### 说明

如使用外置MQTT broker，请注意需要保证外置MQTT broker通信的端口能正常使用。

## MQTT Topic

终端设备与边缘节点、IEF的通信都是通过给MQTT broker中转消息实现的，在MQTT broker中，默认提供如表3-4所示的Topic（消息主题），上报状态、控制终端设备状态都是通过发送/订阅消息实现的。

应用程序编写完后，可以通过应用部署功能，将应用从IEF中部署到边缘节点，详情请参见[容器应用管理](#)。

表 3-4 IEF 提供的默认 Topic

名称	使用类型	Topic	说明
设备孪生变更	订阅	\$hw/events/device/{device_id}/twin/update/document	设备孪生更新文档，当孪生变化时，反映孪生变化前、变化后的区别。
设备孪生delta	订阅	\$hw/events/device/{device_id}/twin/update/delta	设备孪生delta事件，当孪生变化时，反映期望值与真实值不一致的孪生信息。
设备成员变更	订阅	\$hw/events/node/{node_id}/membership/updated	绑定终端设备关系变化。
设备属性变更	订阅	\$hw/events/device/{device_id}/updated	终端设备属性更新。
设备成员获取	发布	\$hw/events/node/{node_id}/membership/get	绑定终端设备关系获取。
设备成员获取结果	订阅	\$hw/events/node/{node_id}/membership/get/result	绑定终端设备关系获取结果。
设备孪生获取	发布	\$hw/events/device/{device_id}/twin/get	设备孪生获取。
设备孪生获取结果	订阅	\$hw/events/device/{device_id}/twin/get/result	设备孪生获取结果。
设备孪生更新	发布	\$hw/events/device/{device_id}/twin/update	设备孪生更新。
设备孪生更新结果	订阅	\$hw/events/device/{device_id}/twin/update/result	设备孪生更新结果。
请求加密数据	发布	\$hw/{project_id}/encryptdatas/{encryptdata_name}/properties/{properties_name}/decrypt	发布获取加密数据请求。

名称	使用类型	Topic	说明
获取加密数据	订阅	\$hw/{project_id}/ encryptdatas/ {encryptdata_name}/ properties/ {properties_name}/ plaintext	订阅获取加密数据。
添加告警	发布	\$hw/alarm/{appname}/add	向AOM发送告警。
清除告警	发布	\$hw/alarm/{appname}/ clear	清除AOM中告警。
自定义Topic	发布	{project_id}/nodes/ {node_id}/user/ {custom_topic}	自定义Topic，Topic根据您的需要自行定义。 您可以将终端设备数据发送到边缘节点MQTT broker的自定义Topic中，IEF会将这些数据转发到DIS通道或APIG后端地址。数据转发到DIS通道或者APIG后端地址后，您可以提取这些数据，并对数据进行处理分析。

接下来将介绍如何在边缘侧获取终端设备信息，接收云上的控制消息，以及如何将终端设备数据上报到云端。MQTT收发消息的示例代码请参见[Go语言代码样例](#)和[Java语言代码样例](#)。

## 获取节点关联的终端设备成员

**步骤1** 向[设备成员获取](#)发送获取终端设备成员消息的请求。

**Topic:** \$hw/events/node/{node\_id}/membership/get

**Payload:** {"event\_id":"自定义ID"}

示例如下：

```
$hw/events/node/{node_id}/membership/get
{"event_id":""}
```

**步骤2** 向[设备成员获取结果](#)订阅终端设备成员的返回结果。

**Topic:** \$hw/events/node/{node\_id}/membership/get/result

返回结果示例如下：

```
{
  "event_id": "",
  "timestamp": 1554986455386,
  "devices": [
    {
      "id": "2144773f-13f1-43f5-af07-51991d4fd064",
      "name": "equipmentA",
      "state": "unknown",
    }
  ]
}
```

```
    "attributes": {
      "name": {
        "value": "a",
        "optional": true,
        "metadata": {
          "type": "string"
        }
      }
    }
  }
}
```

----结束

## 获取设备孪生

**步骤1** 向[设备孪生获取](#)发送请求获取设备孪生。

**Topic:** \$hw/events/device/{device\_id}/twin/get

**Payload:** {"event\_id":"自定义id"}

**步骤2** 向[设备孪生获取结果](#)订阅设备孪生的返回结果。

**Topic:** \$hw/events/device/{dvice\_id}/twin/get/result

获取的结果如下:

```
{
  "event_id": "",
  "timestamp": 1554988425592,
  "twin": {
    "humidity": {
      "expected": {
        "value": "0",
        "metadata": {
          "timestamp": 1554988419529
        }
      },
      "optional": true,
      "metadata": {
        "type": "int"
      }
    },
    "temperature": {
      "expected": {
        "value": "0",
        "metadata": {
          "timestamp": 1554988419529
        }
      },
      "optional": true,
      "metadata": {
        "type": "int"
      }
    }
  }
}
```

----结束

## 监听设备孪生事件

通过[获取节点关联的终端设备成员](#)和[获取设备孪生](#)以后，即可获取到节点绑定的终端设备ID，随后即可监听该终端设备的事件。

在云端更新设备孪生属性，设置期望值从而达到控制边侧终端设备的目的。  
例如某个设备有两个孪生属性，humidity和temperature。

图 3-22 孪生属性

属性名	类型	期望值	设置时间
humidity	int	9	2020/07/13 09:40:38 GMT+08:00
temperature	int	0	2020/07/13 09:40:58 GMT+08:00

在“设备孪生”页签中编辑孪生属性，将humidity由9改为10。属性修改完成以后，在端侧可收到两个事件：“设备孪生变更事件”和“设备孪生delta事件”。

- **设备孪生变更事件**：包含变更前和变更后的设备孪生信息详情。
- **设备孪生delta事件**：包含设备孪生的详情信息以及设备孪生属性期望值与实际值不一致的delta部分。

订阅这两个Topic，就可以收到变更设备孪生的消息。

**步骤1** 订阅**设备孪生变更**。

**Topic**: \$hw/events/device/{device\_id}/twin/update/document

在边侧收到变更消息如下：

```
{
  "event_id": "0f921313-4074-46a2-96f6-aac610721059",
  "timestamp": 1555313685831,
  "twin": {
    "humidity": {
      "last": {
        "expected": {
          "value": "9",
          "metadata": {
            "timestamp": 1555313665978
          }
        }
      },
      "optional": true,
      "metadata": {
        "type": "int"
      }
    },
    "current": {
      "expected": {
        "value": "10",
        "metadata": {
          "timestamp": 1555313685831
        }
      },
      "optional": true,
      "metadata": {
        "type": "int"
      }
    }
  },
  "temperature": {
    "last": {
```



```
    "expected": {
      "value": "0",
      "metadata": {
        "timestamp": 1555313665978
      }
    },
    "actual": {
      "value": "2",
      "metadata": {
        "timestamp": 1555299457284
      }
    },
    "optional": true,
    "metadata": {
      "type": "int"
    }
  },
  "current": {
    "expected": {
      "value": "0",
      "metadata": {
        "timestamp": 1555313685831
      }
    },
    "actual": {
      "value": "2",
      "metadata": {
        "timestamp": 1555299457284
      }
    },
    "optional": true,
    "metadata": {
      "type": "int"
    }
  }
}
```

## 步骤2 订阅设备孪生delta。

**Topic:** \$hw/events/device/{device\_id}/twin/update/delta

在边侧收到变更消息如下:

```
{
  "event_id": "60fb5baf-d4ad-47b0-a21e-8b57b52d0978",
  "timestamp": 1555313685837,
  "twin": {
    "humidity": {
      "expected": {
        "value": "10",
        "metadata": {
          "timestamp": 1555313685831
        }
      }
    },
    "optional": true,
    "metadata": {
      "type": "int"
    }
  },
  "temperature": {
    "expected": {
      "value": "0",
      "metadata": {
        "timestamp": 1555313685831
      }
    },
    "actual": {
      "value": "2",
      "metadata": {
```

```

        "timestamp": 1555299457284
      }
    },
    "optional": true,
    "metadata": {
      "type": "int"
    }
  }
},
"delta": {
  "humidity": "10",
  "temperature": "0"
}
}

```

----结束

## 上报设备属性实际值

**步骤1** 发布终端设备孪生更新事件。

**Topic:** \$hw/events/device/{device\_id}/twin/update

**Payload:** {"event\_id":"","timestamp":0,"twin":{"属性":{"actual":{"value":"设备实际值"}}}}

示例如下:

```

{
  "event_id": "",
  "timestamp": 0,
  "twin": {
    "temperature": {
      "actual": {
        "value": "2"
      }
    }
  }
}

```

发布后，在云端可以观察到设备孪生的实际值发生了相应的变化，如[图3-23](#)所示。

**图 3-23** 孪生属性值发生变化

属性名	类型	期望值	设置时间	实际值
humidity	int	9	2020/07/13 09:40:38 GMT+08:00	
temperature	int	0	2020/07/13 09:46:53 GMT+08:00	2

**步骤2** 在边缘侧订阅[设备孪生更新结果](#)，能收到设备孪生更新事件的结果。

**Topic:** \$hw/events/device/{device\_id}/twin/update/result

更新的结果如下所示。

```

{
  "event_id": "",
  "timestamp": 1554992093859,
  "twin": {
    "temperature": {
      "actual": {

```

```
        "value": "2",
        "metadata": {
            "timestamp": 1554992093859
        }
    },
    "optional": true,
    "metadata": {
        "type": "int"
    }
}
}
```

----结束

## 3.2.7 使用证书进行安全认证

### 操作场景

内置MQTT broker默认开启端口进行TLS（Transport Layer Security）安全认证，客户端必须带上证书才能访问MQTT broker。

终端设备和应用可以通过在对应节点详情页创建的证书进行安全认证。

### 约束与限制

- 证书与边缘节点绑定，在一个边缘节点下申请的证书只能用来访问该边缘节点的MQTT broker，如果访问其他边缘节点的MQTT broker，会导致认证失败。
- 一个边缘节点最多只能申请10份证书。
- 证书的有效期为5年。
- MQTT使用限制

表 3-5 MQTT 使用限制

描述	限制
支持的MQTT协议版本	3.1.1
与标准MQTT协议的区别	<ul style="list-style-type: none"><li>• 支持QoS 0</li><li>• 支持Topic自定义</li><li>• 不支持QoS 1和QoS 2</li><li>• 不支持will、retain msg</li></ul>
MQTTS支持的安全等级	采用TCP通道基础 + TLS协议（TLSV1.2 版本）

### 申请证书

#### 📖 说明

证书有效期为5年。

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“边缘资源 > 边缘节点”。

**步骤3** 单击边缘节点名称，进入边缘节点详情。

**步骤4** 选择“证书”页签，单击“添加证书”。

**步骤5** 输入证书名称，单击“确定”。

证书添加成功后会自动下载，请妥善保管证书。

图 3-24 添加证书

**添加证书** ×

**i** 证书添加成功后，系统将自动下载证书文件

\* 名称

描述

0/255

**确定** **取消**

----结束

## 使用证书

证书用于终端设备与MQTT broker通信时鉴权。

下面是[Go语言代码样例](#)和[Java语言代码样例](#)，演示了如何使用证书做鉴权。

### 📖 说明

1. 客户端不需要校验服务端证书，单向认证即可。
2. 内置MQTT broker默认开启8883端口。
3. 样例中的Go语言MQTT Client引用了github.com/eclipse/paho.mqtt.golang开源库。
4. 客户端需要处理断连事件，实现掉线重连机制，提高连接可靠性。

## Go 语言代码样例

```
package main

import (
    "crypto/tls"
    "crypto/x509"
    "fmt"
```

```
"math/rand"
"sync"
"time"

MQTT "github.com/eclipse/paho.mqtt.golang"
)

func main() {
    subClient := InitMqttClient(onSubConnectionLost)
    pubClient := InitMqttClient(onPubConnectionLost)

    wait := sync.WaitGroup{}
    wait.Add(1)

    go func() {
        for {
            time.Sleep(1*time.Second)
            pubClient.Publish("topic", 0, false, "hello world")
        }
    }()

    subClient.Subscribe("topic", 0, onReceived)

    wait.Wait()
}

func InitMqttClient(onConnectionLost MQTT.ConnectionLostHandler) MQTT.Client {
    pool := x509.NewCertPool()
    cert, err := tls.LoadX509KeyPair("/tmp/example_cert.crt", "/tmp/example_cert.key")
    if err != nil {
        panic(err)
    }

    tlsConfig := &tls.Config{
        RootCAs: pool,
        Certificates: []tls.Certificate{cert},
        // 单向认证, client不校验服务端证书
        InsecureSkipVerify: true,
    }
    // 使用tls或者ssl协议, 连接8883端口
    opts := MQTT.NewClientOptions().AddBroker("tls://127.0.0.1:8883").SetClientID(fmt.Sprintf("%f", rand.Float64()))
    opts.SetTLSConfig(tlsConfig)
    opts.OnConnect = onConnect
    opts.AutoReconnect = false
    // 回调函数, 客户端与服务端断连后立刻被触发
    opts.OnConnectionLost = onConnectionLost
    client := MQTT.NewClient(opts)
    loopConnect(client)
    return client
}

func onReceived(client MQTT.Client, message MQTT.Message) {
    fmt.Printf("Receive topic: %s, payload: %s \n", message.Topic(), string(message.Payload()))
}

// sub客户端与服务端断连后, 触发重连机制
func onSubConnectionLost(client MQTT.Client, err error) {
    fmt.Println("on sub connect lost, try to reconnect")
    loopConnect(client)
    client.Subscribe("topic", 0, onReceived)
}

// pub客户端与服务端断连后, 触发重连机制
func onPubConnectionLost(client MQTT.Client, err error) {
    fmt.Println("on pub connect lost, try to reconnect")
    loopConnect(client)
}
```

```
func onConnect(client MQTT.Client) {
    fmt.Println("on connect")
}

func loopConnect(client MQTT.Client) {
    for {
        token := client.Connect()
        if rs, err := CheckClientToken(token); !rs {
            fmt.Printf("connect error: %s\n", err.Error())
        } else {
            break
        }
        time.Sleep(1 * time.Second)
    }
}

func CheckClientToken(token MQTT.Token) (bool, error) {
    if token.Wait() && token.Error() != nil {
        return false, token.Error()
    }
    return true, nil
}
```

## Java 语言代码样例

MqttClientDemo.java文件:

```
/*
Description: MQTT消息收发JAVA demo。需要先创建边缘节点并下载获取客户端证书
*/

package com.example.demo;

import javax.net.ssl.SSLSocketFactory;

import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

/*
* MQTT Demo 展示客户端连接边缘节点broker进行消息的收发，连接进行SSL安全认证，demo演示包括如下：
* 1、MQTT接收客户端，接收MQTT消息
* 2、MQTT发送客户端，发送MQTT消息
*/

public class MqttClientDemo {
    private static int QOS_TYPE = 2;
    //MQTT服务器地址
    private static final String MQTT_HOST = "ssl://x.x.x.x:8883";
    //MQTT发送客户端id
    private static final String MQTT_PUB_CLIENT_ID = "pub_client_1";
    //MQTT接收客户端id
    private static final String MQTT_SUB_CLIENT_ID = "sub_client_1";
    //MQTT通道订阅主题topic
    private static final String TOPIC = "/hello";
    //MQTT客户端连接SSL证书配置路径
    public static final String CLIENT_CERT_FILE_PATH = "example_cert.crt";
    public static final String CLIENT_KEY_FILE_PATH = "example_cert.key";
    //MQTT客户端连接超时时间（秒）
    public static final int TIME_OUT_INTERVAL = 10;
    //MQTT客户端发送心跳间隔（秒）
    public static final int HEART_TIME_INTERVAL = 20;
    //MQTT客户端断线重试间隔（毫秒）
    public static final int RECONNECT_INTERVAL = 10000;
    //MQTT客户端发送消息间隔（毫秒）
    public static final int PUBLISH_MSG_INTERVAL = 3000;
```

```
//MQTT client客户端
private MqttClient mqttClient;
//MQTT client连接MQTT的客户端ID，一般以客户端唯一标识符表示
private String clientId;
//MQTT client连接配置项
private MqttConnectOptions connOpts;
//初始化MQTT客户端未订阅任何topic
private boolean isSubscribe = false;

public MqttClientDemo(String id) throws MqttException {
    setClientId(id);
    initMqttClient();
    initCallback();
    initConnectOptions();
    connectMqtt();
}

/*****
 * 发送消息
 * @param message 待发送的消息
 * @throws MqttException
 *****/
public void publishMessage(String message) throws MqttException {
    MqttMessage mqttMessage = new MqttMessage(message.getBytes());
    mqttMessage.setQos(QOS_TYPE);
    mqttMessage.setRetained(false);
    mqttClient.publish(TOPIC, mqttMessage);
    System.out.println(String.format("MQTT Client[%s] publish message[%s]", clientId, message));
}

/*****
 * 订阅topic
 * @throws MqttException
 *****/
public void subscribeTopic() throws MqttException {
    int[] Qos = {QOS_TYPE};
    String[] topics = {TOPIC};
    mqttClient.subscribe(topics, Qos);
    isSubscribe = true;
}

/*****
 * 启动线程定时发送MQTT消息
 * @throws MqttException
 *****/
public void startPublishMessage() {
    new Thread() {
        @Override
        public void run() {
            while (true) {
                try {
                    Thread.sleep(PUBLISH_MSG_INTERVAL);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                try {
                    publishMessage("hello world!");
                } catch (MqttException e) {
                    System.out.println(String.format("MQTT client[%s] publish message error,errorMsg[%s]",
clientId, e.getMessage()));
                }
            }
        }
    }.start();
}

/*****
 * 初始化MQTT客户端
```

```
* @throws MqttException 连接异常
***** /
private void initMqttClient() throws MqttException {
    MemoryPersistence persistence = new MemoryPersistence();
    mqttClient = new MqttClient(MQTT_HOST, clientId, persistence);
}

/*****
* 初始化连接配置
* @throws MqttException 连接异常
***** /
private void initConnectOptions() {
    connOpts = new MqttConnectOptions();
    // 设置是否清空session, 这里如果设置为false表示服务器会保留客户端的连接记录, 这里设置为true表示
    每次连接到服务器都以新的身份连接
    connOpts.setCleanSession(true);
    connOpts.setHttpsHostnameVerificationEnabled(false);
    // 设置超时时间, 单位为秒
    connOpts.setConnectionTimeout(TIME_OUT_INTERVAL);
    // 设置会话心跳时间, 单位为秒, 服务器会每隔1.5*20秒的时间向客户端发送个消息判断客户端是否在线,
    但这个方法并没有重连的机制
    connOpts.setKeepAliveInterval(HEART_TIME_INTERVAL);
    SSLSocketFactory factory = null;
    try {
        factory = SslUtil.getSocketFactory(CLIENT_CERT_FILE_PATH, CLIENT_KEY_FILE_PATH);
    } catch (Exception e) {
        e.printStackTrace();
    }
    // TLS连接配置
    connOpts.setSocketFactory(factory);
}

/*****
* 发起连接MQTT connect请求
* @throws MqttException 连接异常
***** /
private void connectMqtt() throws MqttException {
    mqttClient.connect(connOpts);
    System.out.println(String.format("MQTT client[%s] is connected,the connctOptions: \n%s", clientId,
connOpts.toString()));
}

/*****
* 设置回调接口
* @throws MqttException 连接异常
***** /
private void initCallback() {
    mqttClient.setCallback(new MqttMessageCallback());
}

private void setClientId(String id) {
    clientId = id;
}

/*****
* MQTT Client重连函数, 调用连接函数并判断是否订阅过Topic, 如果订阅过topic则重新订阅topic
* @throws MqttException
***** /
private void rconnectMqtt() throws MqttException {
    connectMqtt();
    if (isSubscribe) {
        subscribeTopic();
    }
}

/*****
* MQTT client 订阅topic后, MQTT 通道有数据, 则通过该回调接口接收消息
* @version V1.0
***** /
```



```
private class MqttMessageCallback implements MqttCallback {

    @Override
    public void connectionLost(Throwable cause) {
        System.out.println(String.format("MQTT Client[%s] connect lost,Retry in 10 seconds,info[%s]",
clientId, cause.getMessage()));
        while (!mqttClient.isConnected()) {
            try {
                Thread.sleep(RECONNECT_INTERVAL);
                System.out.println(String.format("MQTT Client[%s] reconnect ....", clientId));
                reconnectMqtt();
            } catch (Exception e) {
                continue;
            }
        }
    }

    @Override
    public void messageArrived(String topic, MqttMessage mqttMessage) {
        String message = new String(mqttMessage.getPayload());
        System.out.println(String.format("MQTT Client[%s] receive message[%s] from topic[%s]", clientId,
message, topic));
    }

    @Override
    public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {

    }
}

public static void main(String[] args) throws MqttException {
    try {
        //订阅MQTT通道
        MqttClientDemo mqttsubClientDemo = new
MqttClientDemo(MqttClientDemo.MQTT_SUB_CLIENT_ID);
        mqttsubClientDemo.subscribeTopic();
        //往MQTT通道发送: hello world
        MqttClientDemo mqttpubClientDemo = new
MqttClientDemo(MqttClientDemo.MQTT_PUB_CLIENT_ID);
        mqttpubClientDemo.startPublishMessage();
    } catch (MqttException e) {
        System.out.println(String.format("program start error,errorMessage[%s]", e.getMessage()));
    }
}
}
```

### SslUtil.java文件:

```
/*
Description: SSL工具类, 加载client ssl证书配置, 忽略服务器证书校验
*/

package com.example.demo;

import java.io.ByteArrayInputStream;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyPair;
import java.security.KeyStore;
import java.security.Security;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;

import javax.net.ssl.KeyManagerFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocketFactory;
```

```
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.openssl.PEMReader;
import org.bouncycastle.openssl.PasswordFinder;

public class SslUtil {

    /*****
     * 验证并获取SSLSocketFactory
     *****/
    public static SSLSocketFactory getSocketFactory(final String crtFile, final String keyFile) throws Exception
    {
        Security.addProvider(new BouncyCastleProvider());

        // 1、加载客户端证书
        PEMReader reader_client =
            new PEMReader(new InputStreamReader(new
            ByteArrayInputStream(Files.readAllBytes(Paths.get(crtFile))));
            X509Certificate cert = (X509Certificate) reader_client.readObject();
            reader_client.close();

        // 2、加载客户端key
        reader_client = new PEMReader(
            new InputStreamReader(new ByteArrayInputStream(Files.readAllBytes(Paths.get(keyFile)))),
            new PasswordFinder() {
                @Override
                public char[] getPassword() {
                    return null;
                }
            }
        );

        // 3、发送客户端密钥和证书到服务器进行身份验证
        KeyStore ks = KeyStore.getInstance(KeyStore.getDefaultType());
        ks.load(null, null);
        ks.setCertificateEntry("certificate", cert);
        ks.setKeyEntry("private-key", ((KeyPair) reader_client.readObject()).getPrivate(), "".toCharArray(), new
        Certificate[] {cert});
        KeyManagerFactory kmf =
        KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
        kmf.init(ks, "".toCharArray());

        // 4、创建socket factory
        SSLContext context = SSLContext.getInstance("TLSv1.2");
        TrustManager[] tms = new TrustManager[1];
        TrustManager miTM = new TrustAllManager();
        tms[0] = miTM;
        context.init(kmf.getKeyManagers(), tms, null);

        reader_client.close();

        return context.getSocketFactory();
    }

    /*****
     * 忽略服务端证书校验
     *****/
    static class TrustAllManager implements TrustManager, X509TrustManager {
        @Override
        public X509Certificate[] getAcceptedIssuers() {
            return null;
        }

        @Override
        public void checkServerTrusted(X509Certificate[] certs, String authType)
```

```
        throws CertificateException {
    }

    public boolean isServerTrusted(X509Certificate[] certs) {
        return true;
    }

    public boolean isClientTrusted(X509Certificate[] certs) {
        return true;
    }

    @Override
    public void checkClientTrusted(X509Certificate[] certs, String authType)
        throws CertificateException {
    }
}
}
```

pom.xml文件:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>mqtt.example</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>7</source>
                    <target>7</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.eclipse.paho/org.eclipse.paho.client.mqttv3 -->
        <dependency>
            <groupId>org.eclipse.paho</groupId>
            <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
            <version>1.2.1</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.bouncycastle/bcprov-jdk16 -->
        <dependency>
            <groupId>org.bouncycastle</groupId>
            <artifactId>bcprov-jdk16</artifactId>
            <version>1.45</version>
        </dependency>
    </dependencies>
</project>
```

## 3.2.8 MQTT Topic

### 3.2.8.1 设备孪生变更

设备孪生更新文档，当孪生变化时，反映孪生变化前、变化后的区别。

## Topic

\$hw/events/device/{device\_id}/twin/update/document

参数	类型	说明
device_id	String	终端设备ID

## 使用方式

使用MQTT客户端订阅该Topic。

## 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
twin	Object	设备孪生变更信息集合，每个孪生以key/value形式存在。value中包含变化前last、变更后current的孪生信息，孪生信息中包含是否可选、孪生metadata包含类型信息、孪生期望状态包含期望值和更新时间、孪生真实状态包含真实值和更新时间等。

## 示例

终端设备绑定到边缘节点时可收到如下消息。

```
$hw/events/device/{device_id}/twin/update/document
{
  "event_id": "",
  "timestamp": 1557314742122,
  "twin": {
    "state": {
      "last": null,
      "current": {
        "expected": {
          "value": "running",
          "metadata": {
            "timestamp": 1557314742122
          }
        }
      },
      "optional": true,
      "metadata": {
        "type": "string"
      }
    }
  }
}
```

### 3.2.8.2 设备孪生 delta

设备孪生delta事件，当孪生变化时，反映期望值与真实值不一致的孪生信息。

#### Topic

`$hw/events/device/{device_id}/twin/update/delta`

参数	类型	说明
device_id	String	终端设备ID

#### 使用方式

使用MQTT客户端订阅该Topic。

#### 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
twin	Object	设备孪生变更信息集合，每个孪生以key/value形式存在。value中包含是否可选、孪生metadata包含类型信息、孪生期望状态包含期望值和更新时间、孪生真实状态包含真实值和更新时间等。
delta	Map	包含设备孪生期望值与真实值不同的孪生名称和期望值。

#### 示例

终端设备绑定到边缘节点时可收到如下消息。

```
$hw/events/device/{device_id}/twin/update/delta
{
  "event_id":"b9625811-f34f-4252-bee9-98185e7e1ec7",
  "timestamp":1557314742131,
  "twin":{
    "state":{
      "expected":{
        "value":"running",
        "metadata":{
          "timestamp":1557314742122
        }
      },
      "optional":true,
      "metadata":{
        "type":"string"
      }
    }
  },
  "delta":{
```

```
"state":"running"
  }
}
```

### 3.2.8.3 设备成员变更

绑定终端设备关系变化。

#### Topic

\$hw/events/node/{node\_id}/membership/updated

参数	类型	说明
node_id	String	节点ID

#### 使用方式

使用MQTT客户端订阅该Topic。

#### 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
added_devices	Array	终端设备信息集合，每个终端设备包含设备的id、name、attributes、twin等信息。
removed_devices	Array	终端设备信息集合，每个终端设备包含设备的id、name、attributes、twin等信息。

#### 示例

终端设备绑定到边缘节点时可收到如下消息。

```
$hw/events/node/{node_id}/membership/updated
{
  "event_id":"04a975ab-fd51-49be-85f5-5967e994f640",
  "timestamp":1557314742136,
  "added_devices":[
    {
      "id":"ab39361a-6fc0-4c94-b919-72b1e08ca690",
      "name":"IEF-device",
      "state":"unknown",
      "attributes":{
        "address":{
          "value":"xxx",
          "optional":true,
          "metadata":{
            "type":"string"
          }
        }
      }
    }
  ]
}
```

```

    },
    "twin":{
      "state":{
        "expected":{
          "value":"running",
          "metadata":{
            "timestamp":1557314434570
          }
        },
        "optional":true,
        "metadata":{
          "type":"string"
        }
      }
    }
  },
  "removed_devices":null
}

```

### 3.2.8.4 设备属性变更

终端设备属性更新。

#### Topic

\$hw/events/device/{device\_id}/updated

参数	类型	说明
device_id	String	终端设备ID

#### 使用方式

使用MQTT客户端订阅该Topic。

#### 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生时间戳
attributes	Object	终端设备属性变更信息集合，key/value形式存在。key为属性名称，value包含属性值、是否可选或属性metadata包含类型信息等。

#### 示例

终端设备绑定到边缘节点时可收到如下消息。

```

$hw/events/device/{device_id}/updated
{
  "event_id": "",
  "timestamp": 1557314742136,

```

```
"attributes":{
  "address":{
    "value":"xxx",
    "optional":true,
    "metadata":{
      "type":"string"
    }
  }
}
```

### 3.2.8.5 设备成员获取

发布获取终端设备成员信息请求。

#### Topic

\$hw/events/node/{node\_id}/membership/get

参数	类型	说明
node_id	String	节点ID

#### 使用方式

使用MQTT客户端发布该Topic，与[设备成员获取结果](#)配对使用。

#### 参数说明

参数	类型	说明
event_id	String	事件ID，由用户自定义。

#### 示例

```
$hw/events/node/3fbb5b8d-32db-4271-a34f-a013e021b6ce/membership/get
{
  "event_id":"bc876bc-345d-4050-86a8-319a5b13cc10"
}
```

### 3.2.8.6 设备成员获取结果

订阅终端设备成员信息获取结果。

#### Topic

\$hw/events/node/{node\_id}/membership/get/result

参数	类型	说明
node_id	String	节点ID



## 使用方式

使用MQTT客户端订阅该Topic，与[设备成员获取](#)配对使用。

## 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
devices	Array	终端设备信息集合，每个设备包含设备的id、name、attributes等信息。attributes信息key/value形式存在。key为属性名称，value包含属性值、是否可选或属性metadata包含类型信息等。

## 示例

```
$hw/events/node/3fbb5b8d-32db-4271-a34f-a013e021b6ce/membership/get/result
{
  "event_id":"bc876bc-345d-4050-86a8-319a5b13cc10",
  "timestamp":1557317193524,
  "devices":[
    {
      "id":"ab39361a-6fc0-4c94-b919-72b1e08ca690",
      "name":"IEF-device",
      "state":"unknown",
      "attributes":{
        "address":{
          "value":"longgang",
          "optional":true,
          "metadata":{
            "type":"string"
          }
        }
      }
    }
  ]
}
```

### 3.2.8.7 设备孪生获取

发布获取设备孪生请求。

## Topic

```
$hw/events/device/{device_id}/twin/get
```

参数	类型	说明
device_id	String	终端设备ID

## 使用方式

使用MQTT客户端发布该Topic，与[设备孪生获取结果](#)成对使用。

## 参数说明

参数	类型	说明
event_id	String	事件ID

## 示例

```
$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/get  
{  
  "event_id":"123456"  
}
```

### 3.2.8.8 设备孪生获取结果

订阅设备孪生获取结果。

## Topic

\$hw/events/device/{device\_id}/twin/get/result

参数	类型	说明
device_id	String	设备ID

## 使用方式

使用MQTT客户端订阅该Topic，与[设备孪生获取](#)配对使用。

## 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
twin	Object	终端设备孪生信息集合，每个孪生以key/value形式存在。value中包含是否可选、孪生metadata包含类型信息、孪生期望状态包含期望值和更新时间、孪生真实状态包含真实值和更新时间等。

## 示例

```
$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/get/result
```

```
{
  "event_id":"123456",
  "timestamp":1557317510926,
  "twin":{
    "state":{
      "expected":{
        "value":"stop",
        "metadata":{
          "timestamp":1557316778931
        }
      }
    },
    "optional":true,
    "metadata":{
      "type":"string"
    }
  }
}
```

### 3.2.8.9 设备孪生更新

发布设备孪生更新信息。

#### Topic

\$hw/events/device/{device\_id}/twin/update

参数	类型	说明
device_id	String	终端设备ID

#### 使用方式

使用MQTT客户端发布该Topic，与[设备孪生更新结果](#)配对使用。

#### 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
twin	Object	需要修改的设备孪生的信息集合，key/value形式存在。key为需要修改的孪生的名称，value包含需要修改期望状态的期望值或真实状态的真实值。

#### 示例

```
$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/update
{
  "event_id":"123457",
  "twin":{
    "state":{
      "actual":{
        "value":"stop"
      }
    }
  }
}
```

```
}
  }
}
}
```

### 3.2.8.10 设备孪生更新结果

订阅设备孪生更新结果。

#### Topic

\$hw/events/device/{device\_id}/twin/update/result

参数	类型	说明
device_id	String	终端设备ID

#### 使用方式

使用MQTT客户端订阅该Topic，与[设备孪生更新](#)配对使用。

#### 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
twin	Object	设备孪生更新信息集合，每个孪生以key/value形式存在。value中包含是否可选、孪生metadata包含类型信息、孪生期望状态包含期望值和更新时间或孪生真实状态包含真实值和更新时间等。

#### 示例

\$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/update/result

```
{
  "event_id":"123457",
  "timestamp":1557317614026,
  "twin":{
    "state":{
      "actual":{
        "value":"stop",
        "metadata":{
          "timestamp":1557317614026
        }
      },
      "optional":true,
      "metadata":{
        "type":"string"
      }
    }
  }
}
```

### 3.2.8.11 请求加密数据

发布获取加密数据请求。

#### Topic

`$hw/{project_id}/encryptdatas/{encryptdata_name}/properties/{properties_name}/decrypt`

参数	类型	说明
project_id	String	项目ID。获取方式请参见 <a href="#">获取项目ID</a> 。
encryptdata_name	String	加密数据集名称。
properties_name	String	加密数据的“键名”。

#### 使用方式

使用MQTT客户端发布该Topic，与[获取加密数据](#)配对使用。

请求时必须使用证书进行安全认证，认证方法请参见[使用证书进行安全认证](#)。

#### 参数说明

无

#### 示例

发布空消息即可。

### 3.2.8.12 获取加密数据

订阅获取加密数据。

#### Topic

`$hw/{project_id}/encryptdatas/{encryptdata_name}/properties/{properties_name}/plaintext`

参数	类型	说明
project_id	String	项目ID。获取方式请参见 <a href="#">获取项目ID</a> 。
encryptdata_name	String	加密数据集名称。
properties_name	String	加密数据的“键名”。

## 使用方式

使用MQTT客户端订阅该Topic，与[请求加密数据](#)配对使用。

请求时必须使用证书进行安全认证，认证方法请参见[使用证书进行安全认证](#)。

## 参数说明

参数	类型	说明
plain_text	String	加密数据的明文值。
ret_message	String	错误信息。

## 示例

当请求正确时，收到消息如下。

```
{
  "ret_code": 200,
  "plain_text": "xxxxxxxxxx"
}
```

当请求错误时，收到消息如下。

```
{
  "ret_code": 400,
  "ret_message": "xxxxxxxxxx"
}
```

### 3.2.8.13 添加告警

向AOM发送告警。

## Topic

\$hw/alarm/{appname}/add

参数	类型	说明
appname	String	应用名称，任意字符串即可。

## 使用方式

使用MQTT客户端发布该Topic。

## 参数说明

参数	类型	说明
alarmName	String	告警名称，需要支持中英文两个版本，格式为“中文##English”。

参数	类型	说明
alarmId	String	告警ID，需要保证为唯一值。可以参考 <a href="#">alarmId生成方法</a> 。
detailedInformation	String	告警描述，需要支持中英文两个版本，格式为“中文##English”。
url	String	根因分析跳转入口，如果没有则填空。
source	String	告警源，只能是由大小写字母组成的字符串。
cleared	Boolean	清除告警标示。 <ul style="list-style-type: none"> <li>• true：表示上报的告警已经清除。</li> <li>• false：表示需要手动清除。</li> </ul>
policyID	String	告警的规则ID，阈值规则填写ruleId，没有则填写空。
objectInstance	String	定位信息，如果没传此字段，则该字段默认取alarmId的值。
perceivedSeverity	Integer	告警级别 <ul style="list-style-type: none"> <li>• 1：紧急</li> <li>• 2：重要</li> <li>• 3：次要</li> <li>• 4：提示</li> </ul>
resourceId	Object	告警信息，具体请参见 <a href="#">表3-6</a> 。
neType	String	产生告警的资源类型 <ul style="list-style-type: none"> <li>• Application（应用）</li> <li>• DB（数据库）</li> <li>• Host（主机）</li> </ul>
eventType	Integer	告警类型 <ul style="list-style-type: none"> <li>• 21：动态阈值告警</li> <li>• 22：批量阈值告警</li> <li>• 23：阈值告警</li> <li>• 24：系统类告警</li> <li>• 25：新增/删除探针</li> <li>• 26：agent安装类告警</li> <li>• 27：配额超限类告警</li> </ul>
probableCause	String	可能原因
proposedRepairActions	String	修复建议

表 3-6 resourceId

参数	类型	说明
namespace	String	资源类型，有如下几种指标 <ul style="list-style-type: none"> <li>● PAAS.CONTAINER表示容器指标</li> <li>● PAAS.NODE表示节点指标</li> </ul>
dimension	Object	<p>维度信息，用于跟监控上报的节点应用信息关联起来。具体请参见表3-7。</p> <p>目前涉及到告警主要是节点和应用两种类型，所以只需要关注节点和应用这两种维度。</p> <p>应用维度可以细分为服务、实例、容器和进程，这四个维度可以选择一个或多个上报。</p> <ul style="list-style-type: none"> <li>● 服务维度需要上报：clusterId、nameSpace和serviceID</li> <li>● 实例维度需要上报：podID和podName</li> <li>● 容器维度需要上报：containerID和containerName</li> <li>● 进程维度需要上报：processID和processName</li> </ul>

表 3-7 Dimension

参数	类型	说明
clusterId	String	项目ID，获取方式请参见 <a href="#">获取项目ID</a> 。
nameSpace	String	默认为default。
nodeIP	String	节点IP。
serviceID	String	<p>服务ID。</p> <ul style="list-style-type: none"> <li>● 容器应用为{projectId}_{hostid}_{appName}的md5值。</li> <li>● 进程应用为{projectId}_{hostid}_{进程名}_{进程pid}的md5值。</li> <li>● processID定义方式： md5({projectId})_{hostid}_md5({进程名})_{进程pid}。 md5(projectId) 表示求projectId的md5值， md5({进程名})表示求进程名的md5值</li> </ul> <p>这些维度信息需要与监控上报的维度信息一致，不一致会导致无法关联到对应的资源。</p>
podID	String	实例ID。
podName	String	实例名称。
containerID	String	容器ID。



参数	类型	说明
containerName	String	容器名称。
processID	String	进程ID。
processName	String	进程名称。
Application	String	应用名称。

## alarmId 生成方法

### 📖 说明

alarmId可以不使用本方法生成，只要保证唯一就可以。

使用{projectId}\_{产生告警的服务名}\_{维度信息}\_{指标名称}\_{告警类型}\_{规则信息}生成md5值。

其中：

- **projectId**：项目ID，获取方式请参见[获取项目ID](#)。
- **维度信息**：
  - 节点信息：{clusterId}\_{namespace}\_{ip}
  - 容器信息：  
{clusterId}\_{namespace}\_{appName}\_{podName}\_{containerId}
  - 应用信息：{clusterId}\_{namespace}\_{appName}
- **告警类型**：
  - 21：动态阈值告警
  - 22：批量阈值告警
  - 23：阈值告警
  - 24：系统类告警
  - 25：新增/删除探针
  - 26：agent安装类告警
- **规则信息**：阈值规则的告警使用阈值规则名称；业务自己触发的告警（没有规则）则填写NA。动态阈值的则填写policyId。

## 示例

- **节点告警**

```
{
  "alarmName": "测试##test",
  "alarmId": "73ccbccce05de74f9d3dda42f6ecfe20",
  "detailedInformation": "测试##test",
  "url": "",
  "source": "IEF",
  "cleared": false,
  "policyID": "",
  "perceivedSeverity": 4,
  "resourceId": {
```

```
"namespace": "PAAS.NODE",
"dimension": {
  "clusterId": "e277befa37a64ed1aa25b522e686bc28",
  "nameSpace": "default",
  "nodeIP": "192.168.0.164"
}
},
"neType": "Host",
"eventType": 23
}
```

- 应用告警

```
{
  "alarmName": "应用重启test##Application restart",
  "alarmId": "b09076ff565c59d4da0db0c9223781",
  "detailedInformation": "应用重启 test##Application restart test",
  "url": "",
  "source": "IEF",
  "cleared": false,
  "policyID": "",
  "perceivedSeverity": 3,
  "resourceId": {
    "namespace": "PAAS.CONTAINER",
    "dimension": {
      "containerName": "container-e991acd3-864c-4038-8a90-e042eebab496",
      "containerID": "70b385315c8ac507b3de7dfe1258932cea0b53a850b7d030ce7ed0a55c47877c",
      "podID": "0e9ce4fd-b732-11e9-8a30-fa163e9b3546",
      "podName": "hxxkapp1-7898f5bd4b-2lj8z"
    }
  },
  "neType": "Application",
  "eventType": 23
}
```

### 3.2.8.14 清除告警

清除AOM中告警。消息体与[添加告警](#)保持一致即可，即清除告警与添加告警只有Topic不同，其余可以相同。

#### Topic

\$hw/alarm/{appname}/clear

参数	类型	说明
appname	String	应用名称，任意字符串即可。

#### 使用方式

使用MQTT客户端发布该Topic。

#### 参数说明

与[添加告警](#)的参数一致。

### 3.2.8.15 自定义 Topic

IEF支持自定义Topic，Topic根据您的需要自行定义。

您可以将终端设备数据发送到边缘节点MQTT broker的自定义Topic中，IEF会将这些数据转发到DIS通道或APIG后端地址。数据转发到DIS通道或者APIG后端地址后，您可以提取这些数据，并对数据进行处理分析。

使用自定义Topic需要在IEF创建消息路由，并在此处定义Topic，IEF根据消息路由转发Topic中的数据，创建消息路由的具体步骤请参见[边云消息概述](#)。

## Topic

```
{project_id}/nodes/{node_id}/user/{custom_topic}
```

## 使用方式

使用MQTT客户端发布该Topic。

## 参数说明

IEF转发自定义Topic内容是透传，可以转发任意内容。

“{custom\_topic}”支持通配符“#”和“+”，可以将多条符合通配规则的消息进行统一转发。

“#”是一个匹配主题中任意层次数的通配符，多层通配符可以表示大于等于0的层次。“+”是单层通配符，只匹配主题的一层。

对于符合通配规则的Topic消息进行最小范围匹配后进行路由消息转发。如消息Topic为123/aaa/567和123/bbb/567，可以配置通配转发规则123+/567进行统一转发。

## 示例

创建路由规则后，您可以往自定义Topic中发送消息（数据），IEF会将消息转发到指定的端点。在IEF中还会记录转发成功次数，如下图所示。

图 3-25 转发成功次数

消息路由名	源端点	目的端点	状态	转发消息数
test-rule-5238	SystemREST 云端	SystemEventBus 边缘	启用	共0条
--	/asdf/bbb5196	/test4205		成功:0 失败:0

## 3.3 容器应用管理

### 3.3.1 容器应用

IEF支持下发容器应用到边缘节点，您可以下发自定义边缘应用。本节主要介绍如何创建自定义边缘应用。

#### 约束与限制

- 边缘节点磁盘占用超过70%时，会启动镜像回收机制回收容器镜像占用的磁盘空间，此时部署容器应用会导致容器启动变慢，请在部署容器应用前规划好边缘节点磁盘空间。
- 创建容器应用时，边缘节点会从容器镜像服务拉取镜像，如果镜像超大且边缘节点下载带宽较小，容器镜像没有拉取完成，从而导致控制台上容器应用显示创建

失败。虽然应用创建失败，但容器镜像拉取不会中断，等容器镜像拉取成功后，容器应用的状态会刷新为创建成功。此情况下也可以先将容器镜像拉取到边缘节点，然后再创建容器应用。

- 容器镜像的架构必须与节点架构一致，比如节点为x86，那容器镜像的架构也必须是x86。

## 创建边缘应用

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“边缘应用 > 容器应用”，单击页面右上角“创建容器应用”。

**步骤3** 填写基本信息。

- **名称**：容器应用的名称。
- **实例数量**：容器应用的实例数量。一个容器应用只能拥有一个实例。
- **配置方式**
  - 自定义配置：即从零开始配置容器应用，具体请参见[步骤4-步骤6](#)。
  - 应用模板配置：选择一个已经定义好的应用模板，可以在模板的基础上进行修改，使用应用模板能够帮助您省去重复的工作量。模板的定义与[步骤4-步骤6](#)需要的配置相同，创建模板的方法请参见[应用模板](#)。
- **部署描述**：容器应用描述。
- **标签**：标签可用于对资源进行标记，方便分类管理。

图 3-26 基本信息

服务实例  专业版服务实例

创建方式 自定义边缘应用

\* 名称

实例数量 1

配置方式  自定义配置  应用模板配置

部署描述

0/255

标签

还可以创建20个标签。

**步骤4** 配置容器。

选择需要部署的镜像，单击“使用镜像”。

- **我的镜像**：展示了您在[容器镜像服务](#)中创建的所有镜像。

- **他人共享**：展示了其他用户共享的镜像，共享镜像是在SWR中操作的，具体请参见[共享私有镜像](#)。

选择镜像后，您可以配置容器的规格。

- **镜像版本**：请选择需要部署的镜像版本。

#### 须知

在生产环境中部署容器时，应避免使用latest版本。因为这会导致难以确定正在运行的镜像版本，并且难以正确回滚。

- **容器规格**：据需要选择容器CPU、内存、昇腾AI加速卡和GPU的配额。
- **昇腾AI加速卡**容器应用选择的AI加速卡配置与实际部署的边缘节点配置的AI加速卡必须一致，否则会创建应用失败，详见[注册边缘节点时AI加速卡配置](#)。

#### 说明

虚拟化切分后的NPU类型，一个容器只能挂载一个虚拟化NPU，只有当该容器退出后，该虚拟化NPU才能分配给其他容器使用。

昇腾AI加速卡支持的NPU类型，如下表。

表 3-8 NPU 类型说明

类型	描述
昇腾310	昇腾310芯片
昇腾310B	昇腾310B芯片

图 3-27 容器配置

The screenshot shows a configuration form for a container. It includes the following fields and options:

- 镜像名称**: wordpress (with a [更换镜像](#) link)
- \* 镜像版本**: php7.3 (dropdown menu)
- \* 容器名称**: container-047e662c (text input)
- 容器规格**:
  - CPU配额**:
    - 申请: 0.25 Core
    - 限制: 0.25 Core
  - 内存配额**:
    - 申请: 512.00 MiB
    - 限制: 512.00 MiB
- AI加速卡**:
  - 不启用
  - 昇腾AI加速卡
  - Nvidia GPU

您还可以对容器进行如下高级配置。

- **运行命令**

容器镜像拥有存储镜像信息的相关元数据，如果不设置生命周期命令和参数，容器运行时运行镜像制作时提供的默认的命令和参数，Dockerfile这两个字段为“Entrypoint”和“CMD”。

如果在创建容器应用时填写了容器的运行命令和参数，将会覆盖镜像构建时的默认命令"Entrypoint"、"CMD"，规则如下：

表 3-9 容器如何执行命令和参数

镜像 Entrypoint	镜像CMD	容器运行命令	容器运行参数	最终执行
[touch]	[/root/test]	未设置	未设置	[touch /root/test]
[touch]	[/root/test]	[mkdir]	未设置	[mkdir]
[touch]	[/root/test]	未设置	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

图 3-28 运行命令



- 运行命令  
输入可执行的命令，例如/run/start。  
若可执行命令有多个，多个命令之间用空格进行分隔。若命令本身带空格，则需要加引号（" "）。

**说明**

多命令时，运行命令建议用/bin/sh或其他shell，其他全部命令作为参数来传入。

- 运行参数  
输入控制容器运行命令的参数，例如--port=8080。  
若参数有多个，多个参数以换行分隔。
- 安全选项
  - 可以通过开启“特权选项”，使容器拥有root权限，可以访问主机上的设备（如GPU、FPGA）。
  - 指定运行用户  
IEF默认不改变容器运行的用户，以构建镜像时定义的运行用户来运行。  
打开开关后，下方出现运行用户的输入框，可输入范围为0~65534的整数。指定了运行用户后，应用将以该运行用户来运行。如果镜像的操作系统中没有该用户ID，将导致容器应用启动失败。

- **环境变量配置**

容器运行环境中设定的变量。可以在应用部署后修改，为应用提供极大的灵活性。当前支持手动添加、密钥导入、配置项导入和变量引用方式。

- 手动添加支持自定义变量名称和变量值。
- 使用密钥导入，环境变量名称可自定义输入，环境变量值支持引用密钥的属性值，密钥创建方法请参见[密钥](#)。
- 使用配置项导入，环境变量名称可自定义输入，环境变量值支持引用配置项的属性值，配置项创建方法请参见[配置项](#)。
- 变量引用支持引用“hostIP”，即边缘节点的IP地址。

#### 📖 说明

IEF不会对用户输入的环境变量进行加密。如果用户配置的环境变量涉及敏感信息，用户需要自行加密后再填入，并在应用中自己完成解密过程。

IEF也不提供任何加解密工具，如果您需要设置加密密文，可以使用其他加解密工具。

- **数据存储**

您可以通过定义本地卷，将边缘节点本地存储目录挂载到容器中，以实现数据文件的持久化存储。

当前支持如下四种类型的本地卷。

- **hostPath**：将主机某个目录挂载到容器中。hostPath是一种持久化存储，应用删除后hostPath里面的内容依然存在于边缘节点本地硬盘目录中，如果后续重新创建应用，挂载后依然可以读取到之前写入的内容。

您可以将应用日志目录挂载到主机的“/var/IEF/app/log/{appName}”目录，“{appName}”是应用名，边缘节点会将“/var/IEF/app/log/{appName}”目录下后缀为log和trace的文件上传到AOM。

挂载目录为容器内应用的日志路径，如Nginx应用默认的日志路径为/var/log/nginx；权限需配置为“读写”。

图 3-29 日志卷挂载

本地卷名称	类型	挂载目录	权限	操作
log	hostPath	/var/IEF/app/log/nginx	/var/log/nginx	读写 删除

- **emptyDir**：一种简单的空目录，主要用于临时存储，支持在硬盘或内存中创建。emptyDir挂载后就是一个空目录，应用程序可以在里面读写文件，emptyDir的生命周期与应用相同，应用删除后emptyDir的数据也同时删除掉。
- **configMap**：存储应用所需配置信息的资源类型，创建方法请参见[配置项](#)。
- **secret**：密钥是一种用于存储应用所需的认证信息、证书、密钥等敏感信息的资源类型，创建方法请参见[密钥](#)。

#### 须知

- 请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，应用创建失败。
- 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。

- **健康检查**

健康检查是指容器运行过程中根据用户需要定时检查容器健康状况或是容器中负载的健康状况。

- 应用存活探针：应用存活探针用于探测容器是否正常工作，不正常则重启实例。当前支持发送HTTP请求和执行命令检查，检测容器响应是否正常。
- 应用业务探针：应用业务探针用于探测业务是否就绪，如果业务还未就绪，就不会将流量转发到当前实例。

详细的配置说明请参见[健康检查配置说明](#)。

**步骤5** 单击“下一步”，进行部署配置。

选择部署对象，指定一个边缘节点。

**图 3-30** 指定边缘节点



您还可以对容器进行如下高级配置。

- **重启策略**

- **总是重启**：当应用容器退出时，无论是正常退出还是异常退出，系统都会重新拉起应用容器。
- **失败时重启**：当应用容器异常退出时，系统会重新拉起应用容器，正常退出时，则不再拉起应用容器。
- **不重启**：当应用容器退出时，无论是正常退出还是异常退出，系统都不再重新拉起应用容器。

---

**须知**

只有选择了“总是重启”，容器应用才能在创建后更新升级和修改访问配置。

- **Host PID**

启用时容器与边缘节点宿主机共享PID命名空间，这样在容器或边缘节点上能够进行互相操作，比如在容器中启停边缘节点的进程、在边缘节点启停容器的进程。

启用此选项要求边缘节点软件版本大于等于2.8.0。

**步骤6** 单击“下一步”，进行访问配置。

容器访问支持端口映射和主机网络两种方式。

---

**注意**

当部署在同一个边缘节点的容器端口冲突时，容器会启动失败。

---



## • 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。

端口映射可以选择主机网卡，请注意端口映射不支持选择IPv6地址类型的网卡。

图 3-31 端口映射



## • 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。

**步骤7** 单击“下一步”，确认容器应用的规格，确认无误后勾选“我已经阅读并同意《华为云服务等级协议》”，单击“创建”。

---结束

## 查看应用运维信息

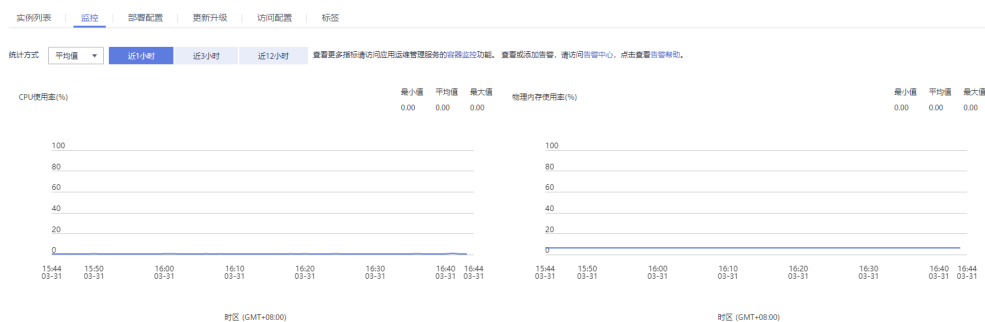
应用部署后，您可以在IEF控制台查看应用的CPU、内存等信息。您还可以通过访问应用运维管理（AOM）的“容器监控”功能，查看更多监控指标，也可以在“告警”功能查看或添加告警。

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏“边缘应用 > 容器应用”，单击容器应用名称。

**步骤3** 选择“监控”页签即可查看应用监控信息。

图 3-32 应用监控信息



---结束

## 更新升级应用

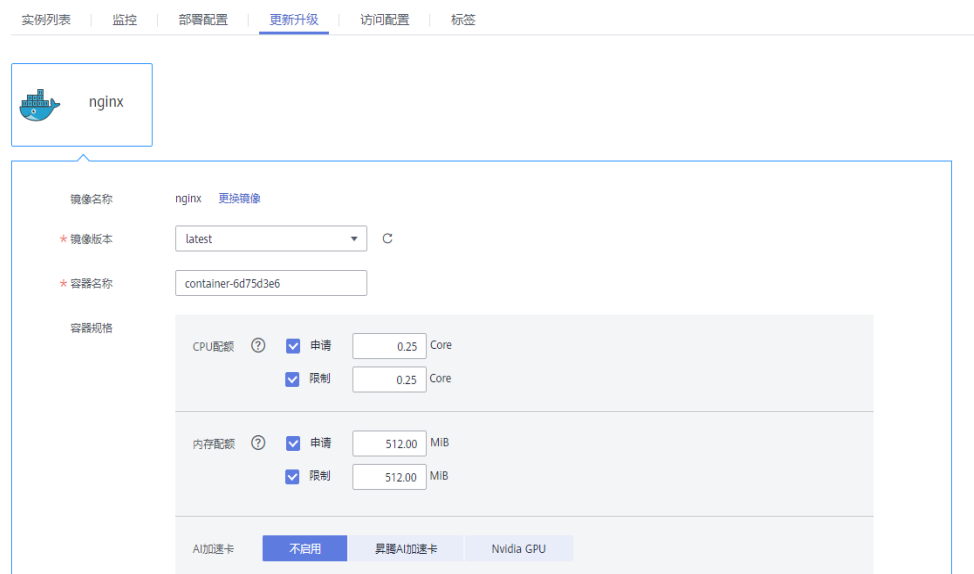
应用部署后，您可以更新升级应用，应用升级采用滚动升级的方式，即先创建新应用实例，然后删除老的应用实例。

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏“边缘应用 > 容器应用”，单击容器应用名称。

**步骤3** 选择“更新升级”页签，修改容器相关配置。具体配置参数与**步骤4**中一致。

图 3-33 更新升级



**步骤4** 配置修改后，单击“提交”。

----结束

## 修改访问配置

应用部署后，您可以修改应用的访问配置。

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏“边缘应用 > 容器应用”，单击容器应用名称。

**步骤3** 选择“访问配置”页签，修改容器相关配置。具体配置参数与**步骤6**中一致。

图 3-34 修改访问配置



**步骤4** 配置修改后，单击“提交”。

---结束

## 3.3.2 应用模板

容器应用模板用于定义用户的边缘应用，用户需要指定容器应用的容器镜像、配置信息、磁盘挂载信息以及资源占用信息。

应用需要基于镜像创建，用户首先需要先制作镜像并上传至镜像仓库。

### 创建应用模板

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏“边缘应用 > 应用模板”，单击页面右上角“创建应用模板”。

**步骤3** 填写基本信息。

- **名称（必填）**：应用模板名称。
- **版本（必填）**：版本号。
- **别名**：应用模板名称以外的一种名称。
- **架构**：选择应用支持的架构。
- **描述**：模板的描述信息。
- **标签**：标签可用于对应用模板进行标记，方便分类管理。此处的标签只用于标识应用模板，可以在搜索时使用标签过滤应用模板。

**步骤4** 单击“下一步”，添加容器。

1. 选择需要部署的镜像，单击“使用镜像”。
  - **我的镜像**：展示了您在容器镜像服务中创建的所有镜像。
  - **他人共享**：展示了其他用户共享的镜像。
2. 单击“下一步”，配置容器规格。

图 3-35 镜像信息

The screenshot shows the 'Add Container' step of the application template configuration. The interface includes a progress bar with three steps: 1. Basic Information, 2. Add Container (active), and 3. Application Configuration. The main form area contains the following fields and options:

- 镜像名称** (Image Name): nginx, with a '修改镜像' (Modify Image) link.
- 镜像版本** (Image Version): latest\_metadata (dropdown menu).
- 容器规格** (Container Specifications):
  - CPU配额** (CPU Quota):
    - 申请** (Request): 0.25 Core
    - 限制** (Limit): 1.00 Core
  - 内存配额** (Memory Quota):
    - 申请** (Request): 512.00 MiB
    - 限制** (Limit): 512.00 MiB
- At the bottom, there are three buttons: '不启用' (Not Enabled), '昇腾AI加速卡' (Ascend AI Accelerator Card), and 'Nvidia GPU'.

- **镜像版本**：请选择需要部署的镜像版本。
- **容器规格**：根据需要选择容器CPU、内存、昇腾AI加速卡和GPU的配额。

- **昇腾AI加速卡**：容器应用选择的AI加速卡配置与实际部署的边缘节点配置的AI加速卡必须一致，否则会创建应用失败，详见[注册边缘节点时AI加速卡配置](#)。

**说明**

虚拟化切分后的NPU类型，一个容器只能挂载一个虚拟化NPU，只有当该容器退出后，该虚拟化NPU才能分配给其他容器使用。

昇腾AI加速卡支持的NPU类型，如下表。

**表 3-10** NPU 类型说明

类型	描述
昇腾310	昇腾310芯片
昇腾310B	昇腾310B芯片

**步骤5** 单击“下一步”，配置应用信息。

- **启动命令**

容器镜像拥有存储镜像信息的相关元数据，如果不设置生命周期命令和参数，容器运行时运行镜像制作时提供的默认命令和参数，Dockerfile中这两个字段为“Entrypoint”和“CMD”。

如果在创建容器应用时填写了容器的运行命令和参数，将会覆盖镜像构建时的默认命令“Entrypoint”、“CMD”，规则如下：

**表 3-11** 容器如何执行命令和参数

镜像 Entrypoint	镜像CMD	容器运行命令	容器运行参数	最终执行
[touch]	[/root/test]	未设置	未设置	[touch /root/test]
[touch]	[/root/test]	[mkdir]	未设置	[mkdir]
[touch]	[/root/test]	未设置	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

**图 3-36** 启动命令



### - 运行命令

输入可执行的命令，例如`/run/start`。

若可执行命令有多个，多个命令之间用空格进行分隔。若命令本身带空格，则需要加引号（`""`）。

#### 📖 说明

多命令时，运行命令建议用`/bin/sh`或其他shell，其他全部命令作为参数来传入。

### - 运行参数

输入控制容器运行命令的参数，例如`--port=8080`。

若参数有多个，多个参数以换行分隔。

## ● 环境变量

容器运行环境中设定的变量。可以在部署应用时修改。

单击“添加环境变量”，输入变量名称，变量值支持手动添加、密钥导入、配置项导入和变量引用方式。

- 手动添加支持自定义变量名称和变量值。
- 使用密钥导入，环境变量名称可自定义输入，环境变量值支持引用密钥的属性值，密钥创建方法请参见[密钥](#)。
- 使用配置项导入，环境变量名称可自定义输入，环境变量值支持引用配置项的属性值，配置项创建方法请参见[配置项](#)。
- 变量引用支持引用“hostIP”，即边缘节点的IP地址。

#### 📖 说明

IEF不会对用户输入的环境变量进行加密。如果用户配置的环境变量涉及敏感信息，用户需要自行加密后再填入，并在应用中自己完成解密过程。

IEF也不提供任何加解密工具，如果您需要设置加密密文，可以使用其他平台的加解密工具。

## ● 卷

卷是指容器运行过程中使用的存储卷，当前支持如下四种类型。

- `hostPath`：将主机某个目录挂载到容器中。`hostPath`是一种持久化存储，应用删除后`hostPath`里面的内容依然存在于边缘节点本地硬盘目录中，如果后续重新创建应用，挂载后依然可以读取到之前写入的内容。

您可以将应用日志目录挂载到主机的“`/var/IEF/app/log/{appName}`”目录，“`{appName}`”是应用名，边缘节点会将“`/var/IEF/app/log/{appName}`”目录下后缀为`log`和`trace`的文件上传到AOM。

图 3-37 日志卷挂载

本地卷名称	类型	挂载目录	权限	操作
log	hostPath	/var/IEF/app/log/ap	/home/log	只读 删除

- `emptyDir`：一种简单的空目录，主要用于临时存储，支持在硬盘或内存中创建。`emptyDir`挂载后就是一个空目录，应用程序可以在里面读写文件，`emptyDir`的生命周期与应用相同，应用删除后`emptyDir`的数据也同时删除掉。
- `configMap`：存储应用所需配置信息的资源类型，创建方法请参见[配置项](#)。
- `secret`：密钥是一种用于存储应用所需的认证信息、证书、密钥等敏感信息的资源类型，创建方法请参见[密钥](#)。

**须知**

- 请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，应用创建失败。
- 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。

**选项****特权选项**

可以通过开启“特权选项”，使容器拥有root权限，可以访问主机上的设备（如GPU、FPGA）。

**指定运行用户**

IEF默认不改变容器运行的用户，以构建镜像时定义的运行用户来运行。

打开开关后，下方出现运行用户的输入框，可输入范围为0~65534的整数。指定了运行用户后，应用将以该运行用户来运行。如果镜像的操作系统中没有该用户ID，将导致容器应用启动失败。

**重启策略**

总是重启：当应用容器退出时，无论是正常退出还是异常退出，系统都会重新拉起应用容器。

失败时重启：当应用容器异常退出时，系统会重新拉起应用容器，正常退出时，则不再拉起应用容器。

不重启：当应用容器退出时，无论是正常退出还是异常退出，系统都不再重新拉起应用容器。

**容器网络**

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。

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。

**健康检查**

- 应用存活探针：应用存活探针用于探测容器是否正常工作，不正常则重启实例。当前支持发送HTTP请求和执行命令检查，通过检测容器响应是否正常。
- 应用业务探针：应用业务探针用于探测业务是否就绪，如果业务还未就绪，就不会将流量转发到当前实例。

详细的配置说明请参见[健康检查配置说明](#)。

**步骤6** 配置完成后，单击“创建”。

----结束

## 创建新的应用模板版本

边缘应用模板支持创建多个应用版本，方便您管理边缘应用。

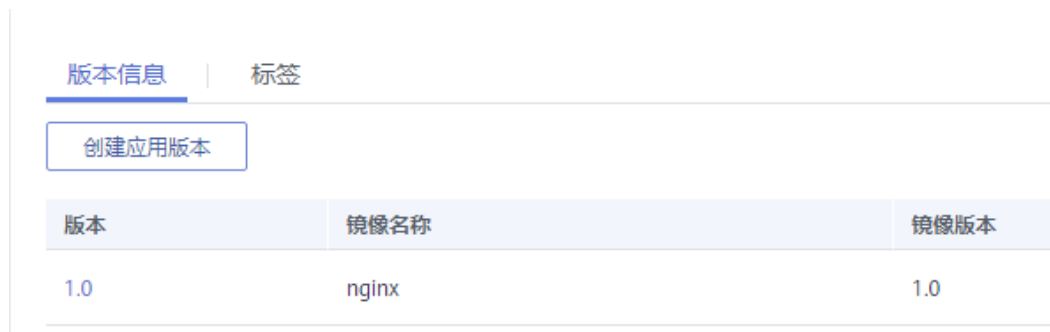
**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“边缘应用 > 应用模板”。

**步骤3** 单击需要增加新版本的应用模板，进入“应用模板详情”页面。

**步骤4** 单击页面左下角的“创建应用版本”。

图 3-38 创建应用版本



**步骤5** 填入应用版本，单击“下一步”，填写模板详细信息，具体请参照[创建应用模板](#)，完成新版本的应用模板创建。

----结束

### 3.3.3 配置项

配置项（ConfigMap）是一种用于存储工作负载所需配置信息的资源类型，配置项允许您将配置文件从容器镜像中解耦，从而增强容器工作负载的可移植性。

配置项价值如下：

- 使用配置项功能可以帮您管理不同环境、不同业务的配置。
- 方便您部署相同工作负载的不同环境，配置文件支持多版本，方便您进行更新和回滚工作负载。
- 方便您快速将您的配置以文件的形式导入到容器中。

#### 创建配置项

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏“边缘应用 > 应用配置”，单击页面右上角“创建配置项”。

**步骤3** 填写配置数据。

图 3-39 配置项

\* 配置项名称 test-config

配置数据

注意：配置项会明文展示所输入信息，请不要填入敏感信息，如涉及敏感信息，请先加密，请防止信息泄露。

属性名	属性值	操作
key	value	删除

添加配置项数据

描述 请输入描述 0/255

- **配置项名称**：输入配置项名称。
- **配置数据**：配置数据是键值对形式，请输入属性名和属性值。

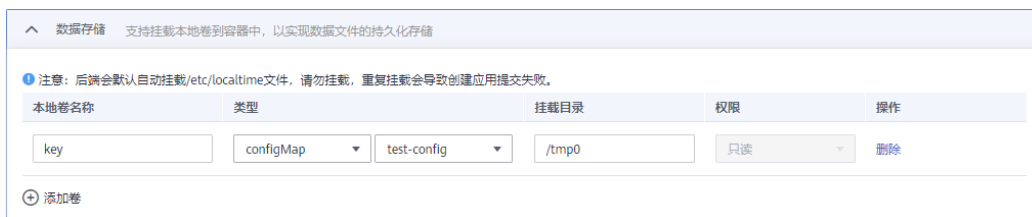
**步骤4** 单击“创建”，即创建配置项成功，返回到配置项列表页面。

----结束

## 配置项的使用

您可以在创建容器应用高级配置中选择数据存储时使用ConfigMap。

图 3-40 使用 ConfigMap



ConfigMap挂载到容器后，会根据ConfigMap的内容在挂载目录下创建文件，每条配置数据（属性名-属性值）为一个文件，其中属性名即文件的名称，属性值为文件的内容。例如ConfigMap的属性名为“key”，属性值为“value”，这条ConfigMap挂载到/tmp0目录下，那么挂载成功后，在/tmp0目录下就存在一个名为“key”的文件，其内容为“value”。

### 3.3.4 密钥

密钥是一种用于存储应用所需的认证信息、证书、密钥等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。例如，在“数据存储”中挂载secret类型的卷，使其成为容器中的文件；或者在“环境变量”中加载，使其成为容器中的环境变量。

#### 须知

密钥的使用可能涉及用户敏感信息，用户输入敏感信息前，需要自行对敏感信息进行加密，并在应用中对加密的数据进行解密后方可使用，请用户注意。

## 创建密钥

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏“边缘应用 > 应用配置”，单击页面右上角“创建密钥”。

**步骤3** 填写密钥信息。



图 3-41 创建密钥

服务实例 专业版服务实例

\* 密钥名称

密钥类型 Opaque

密钥数据 注意：填入信息应先进行加密再进行Base64编码，并保证应用能对其进行解密。

* 属性名	属性值	操作
<input type="text"/>	<input type="text"/>	<span>删除</span>

添加密钥数据

描述

0/255

- **密钥名称**：密钥的名称。
- **密钥类型**：当前支持创建“Opaque”类型的密钥。“Opaque”类型的数据是一个键值对形式，要求属性值为“Base64”编码格式。Base64编码方法请参见[如何进行Base64编码](#)。
- **密钥数据**：密钥数据是键值对形式，请输入属性名和属性值，其中属性值必须为Base64编码格式的字符串。
- **描述**：输入密钥的描述信息。

**步骤4** 单击“创建”，即创建密钥成功，返回到密钥列表页面。

----结束

## 如何进行 Base64 编码

对字符串进行Base64加密，可以直接使用`echo -n 要编码的内容 | base64`命令即可，示例如下：

```
root@ubuntu:~# echo -n "example value" | base64
ZXhhbXBsZSB2YWx1ZQ==
```

## 密钥的使用

您可以在创建容器应用高级配置中选择数据存储时使用Secret。

图 3-42 使用密钥

数据存储 支持挂载本地卷到容器中，以实现数据文件的持久化存储

注意：后端会默认自动挂载/etc/ localtime文件，请勿挂载，重复挂载会导致创建应用提交失败。

本地卷名称	类型	挂载目录	权限	操作
<input type="text" value="key"/>	<span>secret</span>	<input type="text" value="test-secret"/>	<input type="text" value="/tmp0"/>	<span>只读</span> <span>删除</span>

添加卷

Secret挂载到容器后，会根据Secret的内容在挂载目录下创建文件，每条密钥数据（属性名-属性值）为一个文件，其中属性名即文件的名称，属性值为文件的内容。例如Secret的属性名为“key”，属性值为“ZXhhbXBsZSB2YWx1ZQ==”，这条Secret挂载到/tmp0目录下，那么挂载成功后，在/tmp0录下就存在一个名为“key”的文件，其内容为“example value”。

### 3.3.5 加密数据

加密数据是一种用于对敏感信息进行加密的资源类型，可以帮助您对敏感数据进行加密保存，边缘应用可以通过访问边缘MQTT Server获取明文数据。

#### 创建加密数据

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏“边缘应用 > 应用配置”，单击页面右上角“创建加密数据”，填写相关参数。

图 3-43 创建加密数据

服务实例 + 专业版服务实例

\* 加密数据集名称

\* 加密数据项

**注意：**加密数据的数据项至少有一项，会明文展示所输入信息，请不要填入敏感信息，如涉及敏感信息，请先加密，请防止信息泄露。

键名	值	操作
key	.....	删除
	.....	

+ 添加加密数据

描述

0/255

加密数据以键值对形式存储，其中值需要填写两次。每个加密数据集可以添加多条加密数据。

**步骤3** 单击“创建”。

创建后可以查看数据，但出于数据隐私保护，暂不支持查看明文数据。且您可以编辑或删除加密数据。

图 3-44 加密数据

加密数据项 | 绑定节点

+ 添加 **注意：**加密数据项至少保留一项

键名	值	操作
key	.....	<span>编辑</span> <span>删除</span>

----结束

## 绑定加密数据到边缘节点

您可以在多处绑定加密数据到边缘节点。加密数据可以绑定到非运行中状态的边缘节点，当边缘节点恢复到运行中状态时会自动同步已绑定的加密数据到边缘节点。

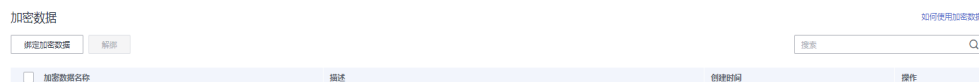
- 在加密数据详情页，您可以选择“绑定节点 > 绑定边缘节点”。

图 3-45 绑定边缘节点



- 在边缘节点详情页选择“配置”，在加密数据部分单击“绑定加密数据”。

图 3-46 绑定加密数据



## 加密数据的使用

加密数据绑定到边缘节点后，在边缘节点上使用MQTT客户端就可以获取到加密数据。

请求时必须使用证书进行安全认证，认证方法请参见[使用证书进行安全认证](#)。

**步骤1** 订阅[获取加密数据](#)。

**Topic:** \$hw/{project\_id}/encryptdatas/{encryptdata\_name}/properties/{properties\_name}/plaintext

**步骤2** 发布[请求加密数据](#)。

**Topic:** \$hw/{project\_id}/encryptdatas/{encryptdata\_name}/properties/{properties\_name}/decrypt

请求发布后，[步骤1](#)中就能收到解密后的明文数据。

----结束

### 3.3.6 健康检查配置说明

健康检查是指容器运行过程中根据用户需要定时检查容器健康状况或是容器中负载的健康状况。

- 应用存活探针：应用存活探针用于探测容器是否正常工作，不正常则重启实例。当前支持发送HTTP请求和执行命令检查，检测容器响应是否正常。
- 应用业务探针：应用业务探针用于探测业务是否就绪，如果业务还未就绪，就不会将流量转发到当前实例。

IEF支持HTTP请求检查和执行命令检查两种方式。

## HTTP 请求检查

向容器发送HTTP GET请求，如果探针收到2xx或3xx，说明容器是健康的。

例如下图这个配置，IEF会在容器启动10秒（延迟时间）后，发送HTTP GET请求到“http://{实例IP}/healthz:8080”，如果在2秒（超时时间）内没有响应则视为检查失败；如果请求响应的状态码为2xx或3xx，则说明容器是健康的。

### 说明

这里无需填写主机地址，默认直接使用实例的IP（即往容器发送请求），除非您有特殊需求。

图 3-47 HTTP 请求检查

检查方式	<input type="radio"/> 不配置	<input checked="" type="radio"/> HTTP请求检查	<input type="radio"/> 执行命令检查
路径	<input type="text" value="/healthz"/>		
端口	<input type="text" value="8080"/>		
主机地址	<input type="text" value="请求的主机地址, 默认为实例IP"/>		
协议	<input checked="" type="radio"/> HTTP	<input type="radio"/> HTTPS	
延迟时间 / 秒	<input type="text" value="10"/>		
超时时间 / 秒	<input type="text" value="2"/>		

## 执行命令检查

探针执行容器中的命令并检查命令退出的状态码，如果状态码为0则说明健康。

例如下图中这个配置，IEF会在容器启动10秒（延迟时间）后，在容器中执行 **cat /tmp/healthy**命令，如果在2秒（超时时间）内没有响应，则视为检查失败；如果命令成功执行并返回0，则说明容器是健康的。

图 3-48 执行命令检查

检查方式	<input type="radio"/> 不配置	<input type="radio"/> HTTP请求检查	<input checked="" type="radio"/> 执行命令检查
执行命令	<input type="text" value="cat /tmp/healthy"/>		
延迟时间 / 秒	<input type="text" value="10"/>		
超时时间 / 秒	<input type="text" value="2"/>		

## 3.4 边云消息

## 3.4.1 边云消息概述

IEF提供了边云消息路由功能，您可以配置消息路由，IEF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，可以是终端设备、云服务等。
- 消息路由：消息转发的路径。

### 消息端点

IEF提供如下默认消息端点：

- **SystemEventBus**：边缘节点上的MQTT，代表节点通信，可以作为源端点向云上发数据，也可以作为目的端点，接收云上消息。端点资源为边缘节点MQTT Topic。
- **SystemREST**：云端的REST网关接口，可以作为源端点，向边缘侧发送REST请求。端点资源为REST请求的路径。

您还可以创建如下消息端点：

- **Service Bus**：边缘节点上的事务请求处理端点，可以作为目的端点，处理文件上传请求。端点资源为REST请求的路径。
- **DIS**：数据接入服务，可以作为目的端点，接收由IEF转发的数据。端点资源为在DIS服务中创建的DIS通道。
- **APIG**：API网关服务，可以作为目的端点，接收由IEF转发的数据。端点资源为在API网关服务中创建的API地址。

### 消息路由

目前支持如下几种消息转发路径：

- SystemREST到Service Bus：通过调用云端的REST Gateway接口，获取边缘节点上的文件服务。
- SystemREST到SystemEventBus：通过调用云端的REST Gateway接口，向边缘节点中的SystemEventBus（MQTT broker）发送消息。
- SystemEventBus到DIS/APIG服务：您可以将终端设备数据发送到边缘节点SystemEventBus（MQTT broker）的自定义Topic中，IEF会将这些数据转发到DIS通道或APIG后端地址。数据转发到DIS通道或者APIG后端地址后，您可以提取这些数据，并对数据进行处理分析。这条路径需要在创建消息路由时自定义MQTT Topic，自定义Topic的详细说明请参见[自定义Topic](#)。

使用消息路由功能，您只需要先[创建消息端点](#)，然后[创建消息路由](#)。

### 创建消息端点

- 步骤1** 登录IEF管理控制台。
- 步骤2** 选择左侧导航栏“边云消息 > 消息端点”。
- 步骤3** 单击页面右上角“创建消息端点”，填写相关参数。

图 3-49 创建消息端点



- **消息端点类型:** 选择类型, 当前支持DIS、APIG和Service Bus。
- **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的端点需要填写, 范围为1-65535。

**步骤4** 单击“确定”, 即创建消息端点成功, 返回到消息端点列表页面。

----结束

## 创建消息路由

- 步骤1** 登录IEF管理控制台。
- 步骤2** 选择左侧导航栏“边云消息 > 消息路由”。
- 步骤3** 单击页面右上角“创建消息路由”。
- 步骤4** 填写相关参数。

图 3-50 创建消息路由

服务实例  专业版服务实例

\* 消息路由名称

\* 源端点

\* 源端点资源

\* 目的端点

\* 目的端点资源

描述

0/255

- **消息路由名称**：输入消息路由名称。

---

 **注意**

消息路由和[系统订阅](#)是同一种资源，命名不能冲突。

- **源端点**：选择源端点，如SystemREST。
- **源端点资源**：输入源端点资源。
- **目的端点**：选择目的端点，如SystemEventBus。
- **目的端点资源**：输入目的端点资源。

**步骤5** 单击“创建”，即创建规则成功，返回到规则列表页面。

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。

----**结束**

## 消息路由停用/启用

- 停用消息路由：在指定消息路由右侧单击“停用”。  
消息路由停用后，该路由规则不再生效，系统不会再对发送到源端点指定资源的消息进行转发处理。
- 启用消息路由：在指定已停用消息路由右侧单击“启用”。  
消息路由启用后，该路由规则生效，系统会对发送到源端点指定资源的消息进行转发处理。

## 3.4.2 云端下发消息到边缘节点

### 操作场景

IEF支持从云端SystemREST下发消息到边缘节点的SystemEventBus（MQTT broker）或服务Bus两种场景，本章节是以前者为例。

通过调用开放在公网的IEF云端的REST Gateway接口，结合节点ID和自定义Topic，向边缘节点中的SystemEventBus发送消息。您的终端设备可以通过订阅对应的自定义Topic进行消息接收，实现自定义Topic的从云到边的消息下发。

使用方法主要分如下三个步骤。

1. [创建消息路由](#)
2. [发送消息](#)
3. [接收消息](#)

### 创建消息路由

SystemREST和SystemEventBus为系统默认端点，无需创建。其中SystemREST代表该Region下IEF云端REST Gateway接口。SystemEventBus代表边缘节点的MQTT broker。

- 步骤1** 登录IEF管理控制台。
- 步骤2** 选择左侧导航栏“边云消息 > 消息路由”。
- 步骤3** 单击页面右上角“创建消息路由”。
- 步骤4** 填写相关参数。



图 3-51 创建消息路由

* 消息路由名称	<input type="text" value="SystemREST2SystemEventBus"/>
* 源端点	<input type="text" value="SystemREST"/>
* 源端点资源	<input type="text" value="/"/>
* 目的端点	<input type="text" value="SystemEventBus"/>
* 目的端点资源	<input type="text" value="/"/>
描述	<div style="border: 1px solid #ccc; padding: 5px; min-height: 100px;">请输入描述</div>

0/255

- **消息路由名称：**输入消息路由名称，如SystemREST2SystemEventBus。

 **注意**

消息路由和[系统订阅](#)是同一种资源，命名不能冲突。

- **源端点：**选择SystemREST。
- **源端点资源：**填写符合URL路径的以“/”开头的字符串，可以使用{}作为入参进行层级模糊匹配，如/aaa/{any-str}/bbb可以匹配/aaa/ccc/bbb或者/aaa/ddd/bbb。整个源端点资源为调用REST接口时的匹配字段。
- **目的端点：**选择SystemEventBus。
- **目的端点资源：**目的端点资源为消息转发至MQTT时对应的Topic前缀。

 **说明**

源端点资源和目的端点资源都填写为“/”时，IEF会对所有发送到REST接口的请求转发到对应节点的MQTT中，且消息的Topic与请求URL一致。

**步骤5** 单击“创建”。

在“消息路由”页面可以看到已创建的路由。

图 3-52 消息路由

消息路由...	源端点	目的端点	状态	转发消息数
<input type="checkbox"/>	SystemREST2Syst... SystemREST 云端	SystemEventBus 边缘	启用	共0条 成功:0 失败: 0

----结束

## 发送消息

从云端SystemREST下发到边缘节点的SystemEventBus，您首先需要获取SystemREST的Endpoint，然后你需要构造请求，将消息通过路由发送至SystemEventBus。

**步骤1** 获取SystemREST的Endpoint。

1. 登录IEF管理控制台。
2. 选择左侧导航栏“边云消息 > 消息端点”。在SystemREST所在行“消息端点属性”列，public的取值即为SystemREST的Endpoint，如下图。

图 3-53 SystemREST 的 Endpoint

消息端点名称	类型	位置	消息端点属性
<input type="checkbox"/> SystemEventBus	eventbus 系统端点	边缘	
<input type="checkbox"/> SystemREST	rest 系统端点	云端	public = restep.ief2.cn-north-4.myhuaweicloud.com

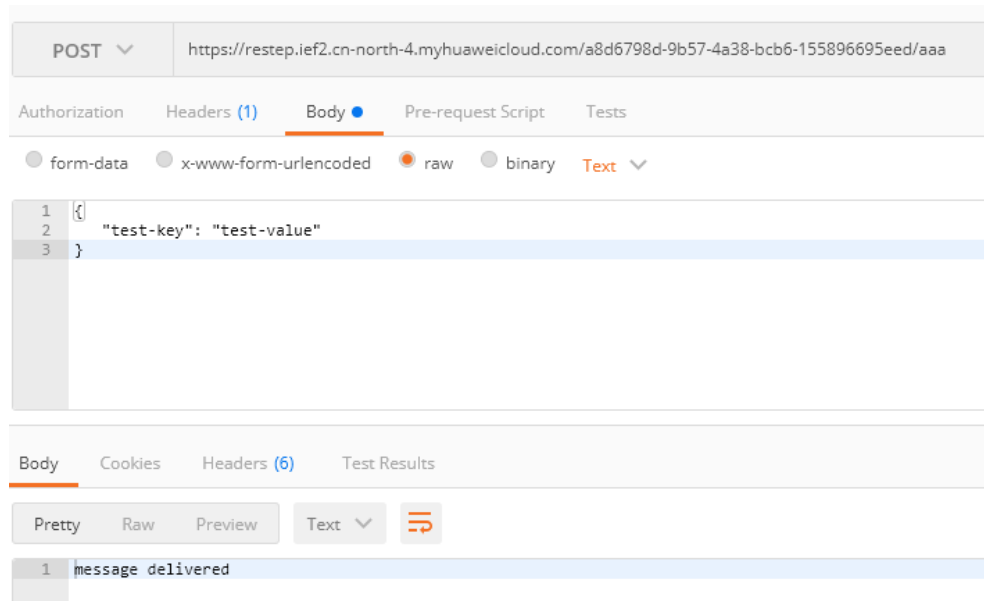
**步骤2** 构造请求，发送消息。

往SystemREST发送消息，需构造HTTPS请求，具体信息如下。

- Method: POST
- URL: **https://{rest\_endpoint}/{node\_id}/{topic}**，rest\_endpoint即**步骤1**中获取的Endpoint，node\_id为边缘节点ID，topic为消息的主题，即**创建消息路由**中定义的源端点资源。
- Body: 发送的消息内容，由用户自定义。
- Header: X-Auth-Token，取值为该用户在IAM获取的有效Token，Token获取方法请参见**用户Token**。

例如使用Postman发送如下消息。

图 3-54 发送消息



----结束

## 接收消息

**步骤1** 登录边缘节点。

**步骤2** 使用MQTT客户端接收消息。

当前边缘节点MQTT支持两种模式。

- 一种是内置的MQTT broker（使用8883端口），需要使用节点证书认证，然后订阅对应Topic接收消息，具体使用方式请参见[使用证书进行安全认证](#)。
- 另外一种是使用外置的MQTT broker（使用1883端口），需要先安装第三方MQTT broker，然后订阅对应Topic接收消息。

这里介绍使用外置的MQTT broker的方式，外置的MQTT broker需要先安装MQTT broker，例如安装Mosquitto，步骤如下。

- 对于Ubuntu操作系统，可以使用如下命令安装mosquitto：

```
apt-get install mosquitto
systemctl start mosquitto
systemctl enable mosquitto
```
- 对于CentOS操作系统，使用如下命令安装mosquitto：

```
yum install epel-release
yum install mosquitto
systemctl start mosquitto
systemctl enable mosquitto
```

安装完成后，使用订阅命令订阅，订阅后如果有消息发送，就会收到消息，如下所示。其中#表示订阅任何主题，可以替换为指定的主题，如/aaa、/bbb等。

```
[root@ief-node ~]# mosquitto_sub -t '#' -d
Client mosq-m02iwjsp4j2ISMw6rw sending CONNECT
Client mosq-m02iwjsp4j2ISMw6rw received CONNACK (0)
Client mosq-m02iwjsp4j2ISMw6rw sending SUBSCRIBE (Mid: 1, Topic: #, QoS: 0, Options: 0x00)
Client mosq-m02iwjsp4j2ISMw6rw received SUBACK
Subscribed (mid: 1): 0
Client mosq-m02iwjsp4j2ISMw6rw received PUBLISH (d0, q0, rQ, mQ, '/aaa', ... (31 bytes))
```

```
{  
  "test-key": "test-value"  
}
```

----结束

### 3.4.3 边缘节点上报消息到云端

#### 操作场景

IEF支持从边缘节点上报消息到云端。

您可以将消息发送到边缘节点SystemEventBus（MQTT broker）的自定义Topic中，IEF会将这些数据转发到DIS通道或APIG后端地址。数据转发到DIS通道或者APIG后端地址后，您可以提取这些数据，并对数据进行处理分析。

本章节使用DIS端点作为示例，APIG端点的使用方法类似，主要分如下几个步骤。

1. [创建消息端点](#)
2. [购买DIS接入通道](#)
3. [创建消息路由](#)
4. [发送消息](#)

#### 创建消息端点

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏“边云消息 > 消息端点”。

**步骤3** 单击页面右上角“创建消息端点”，选择DIS，填写消息端点名称。

图 3-55 创建消息端点



**步骤4** 单击“确定”。

----结束

#### 购买 DIS 接入通道

往DIS发消息需要购买DIS接入通道。

**步骤1** 登录数据接入服务DIS控制台。

**步骤2** 单击右侧“购买接入通道”，填写对应参数，如下图所示。

图 3-56 购买接入通道

\* 计费模式

\* 区域   
不同区域的资源之间内网不互通。请选择靠近您客户的区域，可以降低网络时延、提高访问速度。

\* 通道名称   
可使用自动生成的由前缀“dis-”加4位随机字符或数字组成的名称，例如：dis-HvB1，也可自定义。

\* 通道类型   ?

\* 分区数量 ?  1   您最多可使用49个分区。申请扩大配额  
选择的规格为：普通通道 | 1 个分区 | 通道理论容量：1 MB/秒（接入）；2 MB/秒（读取）

\* 生命周期（小时）  24

\* 源数据类型    ?

\* 自动扩容 ?

企业项目 ?

高级配置

**步骤3** 单击“立即购买”，确认产品规格无误后，单击“提交”。

----结束

## 创建消息路由

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏“边云消息 > 消息路由”。

**步骤3** 单击页面右上角“创建消息路由”。

**步骤4** 填写相关参数，如下图所示。

图 3-57 创建消息路由

服务实例 专业版服务实例

\* 消息路由名称

\* 源端点

\* 源端点资源

**注意：**用户可以向边缘节点的Eventbus中的自定义Topic发布数据，并设置消息路由将发布在该Topic中的数据上传至DIS（数据接入服务）或API网关服务

选择节点

e78acc02d9d141eda5ca8e88fa35f6f8/nodes/0/user/

\* 目的端点

\* 目的端点资源

描述

0/255

- **消息路由名称：**输入消息路由名称。

**注意**

消息路由和**系统订阅**是同一种资源，命名不能冲突。

- **源端点：**选择“SystemEventBus”。
- **源端点资源：**选择“自定义topic”，选择发送消息的边缘节点，填写topic名称。
- **目的端点：**选择**创建消息端点**创建的端点。
- **目的端点资源：**选择**购买DIS接入通道**中购买的通道。

**说明**

- 请记录此处的Topic，如图中红框所示。创建成功后，也可以在消息路由列表中“源端点”列查看。
- 自定义Topic后，需将完整的Topic（如图中红框所示）用于消息发送。

**步骤5** 单击“创建”。

----结束

## 发送消息

在边缘节点使用MQTT客户端发送消息。

此处需要放到**创建消息路由**中指定的Topic，如下图所示使用mosquitto\_pub发送。

```
[root@ief-node ~]# mosquitto_pub -t '05e1aef9040010e22fcc009adecb056/nodes/7092ad14-adee-4a09-b969-1505bbdecef5/user/aaa' -d -m '{ "edgemsg": "msgToCloud" }'
```

```
Client mosq-p5LouPQIW2gx0JPKRF sending CONNECT
Client mosq-p5LouPQIW2gx0JPKRF received CONNACK (0)
Client mosq-p5LouPQIW2gx0JPKRF sending PUBLISH (d0, q0, r0, m1, '05e1aef9040010e22fcc009adecb056/nodes/7092ad14-adee-4a09-b969-1505bbdecef5/user/aaa', ... (26 bytes))
Client mosq-p5LouPQIW2gx0JPKRF sending DISCONNECT
```

消息发送后，您可以在消息路由处看到已经成功转发一条消息，如下图所示。

图 3-58 转发消息数

消息路由...	源端点	目的端点	状态	转发消息数
<input type="checkbox"/> example2dis	<input type="checkbox"/> SystemEventBus <span>边缘</span>	<input type="checkbox"/> dis_example <span>云端</span>	<input checked="" type="checkbox"/> 启用	共1条
--	05e1aef9040010e22fcc009adecb056/nodes/7092a...	dis-ief		<span style="border: 1px solid red; padding: 2px;">成功:1</span> 失败: 0

您可以在DIS界面看到有消息流入，如下图所示。

图 3-59 DIS 数据监控



## 获取数据

数据转发到DIS通道后，您可以提取这些数据，并对数据进行处理分析。DIS的数据获取方法请参见[从DIS获取数据](#)。

## 3.4.4 系统订阅

### 操作场景

IEF提供系统订阅，您可以订阅IEF资源的变更事件，当资源创建、更新、删除时，IEF会发送消息到您指定的APIG端点，以便您及时感知资源的变化。

系统订阅是边云消息的一种特定实现，IEF发送特定资源的事件消息到指定Topic，并调用APIG的API，用户可通过API调用感知资源的变化。

### 支持订阅的事件

当前支持订阅如下事件。

表 3-12 支持订阅的事件

系统事件	Topic	资源类型	操作
实例创建	\$hw/events/instance/+/created	应用实例 ( instance )	created
实例更新	\$hw/events/instance/+/updated	应用实例 ( instance )	updated
实例删除	\$hw/events/instance/+/deleted	应用实例 ( instance )	deleted
应用删除	\$hw/events/deployment/+/deleted	容器应用 ( deployment )	deleted
应用创建	\$hw/events/deployment/+/created	容器应用 ( deployment )	created
应用更新	\$hw/events/deployment/+/updated	容器应用 ( deployment )	updated
节点上线	\$hw/events/edgeNode/+/online	边缘节点 ( edgeNode )	online
节点离线	\$hw/events/edgeNode/+/offline	边缘节点 ( edgeNode )	offline

## 系统订阅流程

系统订阅流程如下：

1. 在APIG创建供IEF调用的API。
2. 在IEF中创建APIG端点。
3. 在IEF中创建系统订阅。

## 创建 API

在APIG中创建API，具体请参见[APIG 快速入门](#)。

该API用于给IEF调用，发送系统事件消息。

## 创建 APIG 端点

- 步骤1** 登录IEF管理控制台。
- 步骤2** 选择左侧导航栏“边云消息 > 消息端点”。
- 步骤3** 单击页面右上角“创建消息端点”，填写相关参数。



图 3-60 创建消息端点



- **消息端点类型**：选择APIG。
- **消息端点名称**：输入消息端点名称。

**步骤4** 单击“确定”，即创建消息端点成功，返回到消息端点列表页面。

----结束

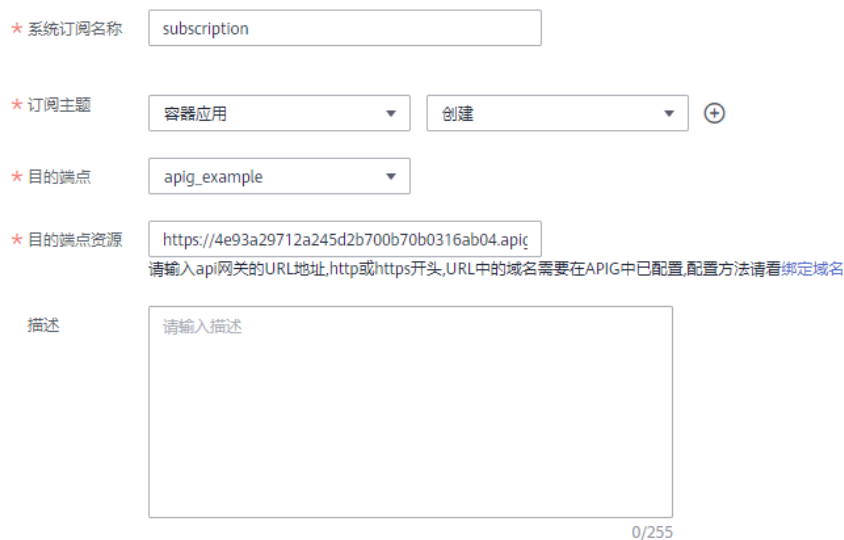
## 创建系统订阅

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏“边云消息 > 系统订阅”。

**步骤3** 单击页面右上角“创建系统订阅”，填写相关参数。

图 3-61 创建系统订阅



- **系统订阅名称**：输入系统订阅名称。

**注意**

系统订阅和消息路由是同一种资源，命名不能冲突。

- **订阅主题**：订阅事件的主题和操作，如创建容器应用，当前支持订阅的事件如表 3-12。
- **目的端点**：创建APIG端点中创建的端点。
- **目的端点资源**：创建API中创建的API资源。

**步骤4** 单击“创建”，完成系统订阅创建。

---结束

## 订阅后说明

创建系统订阅后，当有系统事件发生，在IEF控制台系统订阅列表中，会记录消息转发的次数。

图 3-62 转发消息数

系统订...	订阅主题	目的端点	状态	转发消息数
<input type="checkbox"/>	subscription	apig_example <span>云镜</span>	<input checked="" type="checkbox"/> 启用	共0条 成功:0 失败:0
<input type="checkbox"/>	deployment/created	https://4e93a29712a245d2b700b70b0316ab0...		

同时IEF会调用APIG中注册的API，请求消息体如下所示。请求消息体采用标准的CloudEvents格式，CloudEvents详细信息请参见[这里](#)。

```
{
  "data": {
    "event_type": "instance",
    "operation": "created",
    "timestamp": 134567677,
    "topic": "$hw/events/deployment+/created",
    "name": "xxxx",
    "attributes": {"ID": "x"}
  },
  "datacontenttype": "application/json",
  "source": "sysevents",
  "id": "xxxx",
  "time": "2020-11-5 xxx"
}
```

- **data**：系统事件的数据。
  - event\_type：资源类型，String类型。
  - operation：资源的操作类型，String类型。
  - topic：消息发送的Topic，String类型。
  - timestamp：事件产生的时间戳，Uint64类型。
  - name：资源名称，String类型。
  - attributes：资源的属性，删除资源时消息中无此参数，Object类型。
- datacontenttype：系统事件数据内容的格式，String类型。
- source：系统事件的来源，String类型。
- id：系统事件ID，String类型。

- time：系统事件产生时间，String类型。

当前支持的资源类型、操作和Topic如表3-12所示。

## 3.5 批量管理

### 3.5.1 批量节点注册

#### 使用场景

如果您需要对大量相同类型的边缘节点进行管理、更新与维护，这时使用[注册边缘节点](#)的方法逐一纳管边缘节点会有些繁琐。

IEF提供批量节点注册功能，使得相同类型的边缘节点能够预安装软件，在上电联网后能自动纳管到IEF中。批量节点注册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。

#### 批量节点注册流程

批量纳管边缘节点的流程如下：

1. 准备边缘节点，边缘节点需要满足一定的规格要求，请参见[配置边缘节点环境](#)在节点上完成环境配置。
2. [创建批量节点注册作业](#)，获取配置文件和注册软件，并预安装到边缘节点上，具体请参见[批量纳管边缘节点](#)。如果在启动脚本配置注册命令，设备上电联网后会自动纳管成为IEF的边缘节点。

#### 创建批量节点注册作业

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“批量管理 > 节点注册”，单击页面右上角的“批量节点注册”。

**步骤3** 配置基本信息。其中带“\*”标志的参数为必填参数。

服务实例  专业版服务实例

\* 名称

描述

0/255

标签

还可以创建20个标签。

属性名	属性值	操作
<input type="button" value="新增属性"/>		

- **名称**: 批量节点注册作业的名称。
- **描述**: 作业的描述信息。
- **标签**: 标签可用于对资源进行标记, 方便分类管理。
- **属性**: 属性是键值对形式, 请输入属性名和属性值。

**步骤4** 单击页面右下角的“创建”, 节点注册可选“通过证书注册”和“通过token注册”两种方式。

- **方式一**: 通过证书注册。下载配置文件、EdgeCore Register和EdgeCore Installer。

#### 须知

为保障节点安全, 您现在必须下载配置文件和工具, 稍后将无法找回。

### 请下载软件并在边缘节点完成以下步骤

以下操作将节点连接到智能边缘平台。您必须现在下载配置文件, 稍后将无法找回。



- 根据页面提示, 单击“下载cert\_1.tar.gz配置文件”下载配置文件。
  - 根据您边缘节点的CPU架构选择EdgeCore Register和EdgeCore Installer, 单击“下载EdgeCore Register”和“下载EdgeCore Installer”。
- **方式二**: 通过token注册。下载EdgeCore Register和EdgeCore Installer, 并保存安装凭证。

图 3-63 下载 EdgeCore Register 和 EdgeCore Installer

### 请下载软件并在边缘节点完成以下步骤

以下操作将节点连接到智能边缘平台。

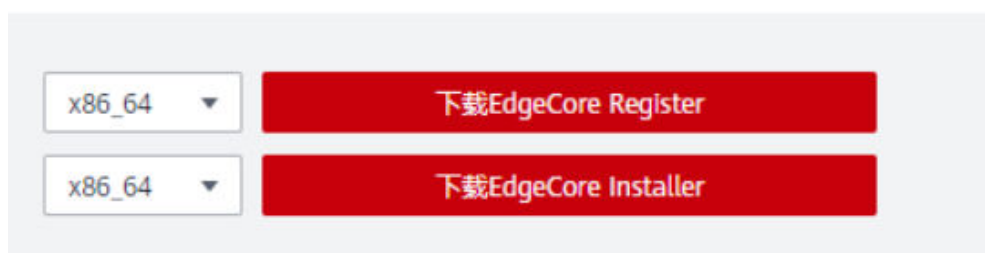


图 3-64 安装凭证

```
4) 执行安装命令

cd /opt/edge-register; ./register --mode=identifier --identifier=eyJ1cmwiOiJodHRwczovLzgy
OTlkZjUwLWYyN2EtNDhkZi05NzhmLTBkM2U5YzU0ZjVhYy5jb206MTAwMDIiLCJhdXRoljoiUW1WaGNpQmxlVXBvWWtkamFVOXBtbE5WZWtreFRtbEpf
5jb206MTAwMDIiLCJhdXRoljoiUW1WaGNpQmxlVXBvWWtkamFVOXBtbE5WZWtreFRtbEpf
MGx1VWpWalEwazTV3R3V0ZaRfNqa3VaWGxLYkdWSVFXbFBhbEUwVFdwSk1rMVVhek5P
UkZGelNXMXNhMGxxYjJsUFJHczFUWHBvYlUxcVjYUk5SMVpZVFZN01FNVVUWGhNVkdjeF
```

- a. 根据您边缘节点的CPU架构选择EdgeCore Register和EdgeCore Installer，单击“下载 EdgeCore Register”和“下载EdgeCore Installer”。
- b. 保存执行安装命令中的 *identifier* 字段。

**步骤5** 将下载的文件上传至边缘节点，稍后参见[批量纳管边缘节点](#)操作指南纳管边缘节点。

----结束

## 批量纳管边缘节点

按照批量节点注册作业完成页的操作指南，在边缘节点执行纳管操作。

**步骤1** 以具备sudo权限的用户登录边缘节点。

**步骤2** 上传[创建批量节点注册作业：步骤4](#)下载好的配置文件和工具到边缘节点。

**步骤3** 执行如下命令解压配置文件、EdgeCore Register和EdgeCore Installer。

```
sudo tar -zxvf edge-installer_1.0.10_x86_64.tar.gz -C /opt
```

```
sudo tar -zxvf edge-register_2.0.10_x86_64.tar.gz -C /opt
```

```
sudo tar -zxvf cert_batchNodes.tar.gz -C /opt/edge-register/ ( “通过token注册” 时不需要执行此命令 )
```

*edge-installer\_1.0.10\_x86\_64.tar.gz*、*edge-register\_2.0.10\_x86\_64.tar.gz*和 *cert\_batchNodes.tar.gz*请替换为[创建批量节点注册作业](#)下载的文件名。

**步骤4** (可选) 配置节点预置信息文件。

批量纳管边缘节点会采用默认配置进行节点纳管，其中：

- 节点名称为节点hostname和mac地址的组合 “hostname\_mac”。
- GPU和NPU默认不启用。
- Docker默认启用。

如需自定义配置节点信息，可以参考以下操作配置节点信息文件。

```
cd /opt/edge-register
```

```
vi nodeinfo
```

在nodeinfo文件中填入节点的配置信息，其格式如下所示（下面样例罗列了所有可配置的参数，可以根据场景需求配置其中的参数值，对于不需要自定义配置的参数可以删除示例中对应的参数配置）：

包括节点名称、描述、是否启用GPU、NPU、NPU类型、属性以及日志配置等，具体参数解释请参见[API参考 > 注册边缘节点](#)。

```
{
  "name": "nodename",
  "description": "",
  "enable_gpu": false,
  "enable_npu": true,
  "npu_type": "****",
  "enable_docker": true,
  "attributes": [
    {
      "key": "key1",
      "value": "value1"
    }
  ],
  "log_configs": [
    {
      "component": "app",
      "type": "local",
      "level": "debug",
      "size": 100,
      "rotate_num": 5,
      "rotate_period": "daily"
    }
  ],
  "device_infos": [
    {
      "device_ids": ["15696983-5ee6-43b4-9653-5d8512813dcc"],
      "relation": "camera",
      "comment": "devicedescription"
    }
  ],
  "mqtt_config": {
    "enable_mqtt": false,
    "mqttps": []
  },
  "tags": [
    {
      "key": "name",
      "value": "value"
    }
  ]
}
```

**步骤5** 执行如下命令，纳管边缘节点。

- 通过证书注册  
**cd /opt/edge-register; ./register --mode=cert**
- 通过token注册  
**cd /opt/edge-register; ./register --mode=identifier --identifier=*token****注册的凭证*  
*token*注册的凭证请替换为[创建批量节点注册作业](#)时保存的安装凭证*identifier*字段。

执行完成后，可以根据[查看纳管的节点](#)查看节点是否注册成功。

您也可以配置上电启动脚本，在边缘节点上电时自动执行注册命令，在上电脚本中添加注册命令：**cd /opt/edge-register; ./register --mode=cert**

如果注册命令的执行结果返回值为0，则表示注册成功。

----结束

## 查看纳管的节点

批量纳管的节点，可以在控制台查看。

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“批量管理 > 节点注册”，在右侧单击作业名称。

**步骤3** 在“节点”页签下，可以查看到已经纳管的节点。

图 3-65 查看批量纳管的节点



----结束

## 3.5.2 批量节点升级

### 使用场景

在某些场景下，您可能需要对大量边缘节点的EdgeCore软件进行管理、更新维护，IEF提供批量节点升级功能，即批量升级边缘节点。

#### 说明

边缘节点需要与IEF正常通信才可以升级成功。

### 注意事项

- 为了让您的边缘节点应用更稳定可靠的运行，IEF不会主动升级您的边缘节点上的EdgeCore，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。
- 处于维护周期中的版本升级，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。
- 处于维护周期外的版本升级，可能会因为容器重启引起业务的短暂中断。
- 请勿在节点升级过程中变更节点配置，比如重启Docker、安装卸载GPU/NPU驱动、OS内核升级、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

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“批量管理 > 节点升级”，单击页面右上角的“批量节点升级”。

**步骤3** 配置批量节点升级作业基本信息。

**图 3-66** 批量节点升级

\* 名称

描述  0/255

标签

升级配置

升级对象

- **名称**：创建的节点升级作业名称。
- **描述**：节点升级作业描述。
- **标签**：节点升级作业的标签
- **升级对象**：需要升级的节点。  
单击“选择边缘节点”，选择需要升级的节点。  
您还可以单击页面右上角的“标签搜索”，输入标签名和标签值，单击“搜索”，筛选出指定标签的节点。然后勾选需要升级的边缘节点，单击“确定”。

**步骤4** 单击“下一步”，确认升级信息，确认无误后单击“创建”。

----结束

## 状态说明

批量节点升级作业有以下八种状态。

- **排队中**：作业等待执行
- **执行中**：作业处于执行状态
- **成功**：全部任务执行成功



- **部分成功**: 部分任务执行成功
- **失败**: 全部任务执行失败
- **停止中**: 作业处于停止中
- **已停止**: 作业已停止
- **更新超时**: 作业排队和执行时间超过10分钟仍未完成

批量作业执行过程中可以停止，停止后可以继续。

如果批量作业执行失败、部分成功或更新超时，可以重试执行作业，将未执行成功的作业再次执行一遍。

图 3-67 重试



作业名称/ID	IEF	状态	作业执行状态	创建时间	更新时间	描述	操作
id01	a318f564-203a-4e88-ac50-20a498e167f8	部分成功	总次数 2 成功数 1 失败数 1 未执行数 0	2022/09/20 16:14:39 GMT+08:00	2022/09/20 16:14:56 GMT+08:00	...	停止 详情 <b>重试</b>
new-test2	5a53f956-8402-432c-84a8-009740984a04	成功	总次数 2 成功数 2 失败数 0 未执行数 0	2022/08/23 10:18:32 GMT+08:00	2022/08/23 10:18:53 GMT+08:00	...	停止 详情 删除

## 3.5.3 批量应用部署

### 使用场景

在某些场景下，您可能需要对大量的边缘节点部署相同的应用，IEF提供批量应用部署功能。

#### 说明

- 节点架构必须全部是同一架构的节点，比如全部为x86或arm。
- 容器镜像的架构必须与节点架构一致，比如节点为x86，那容器镜像的架构也必须是x86。

### 操作步骤

- 步骤1** 登录IEF管理控制台。
- 步骤2** 选择左侧导航栏的“批量管理 > 应用部署”，单击页面右上角的“批量应用部署”。
- 步骤3** 配置应用部署作业基本信息。

图 3-68 创建批量应用部署作业

- **名称**：创建的应用部署作业名称。
- **描述**：应用部署作业描述。
- **标签**：应用部署作业的标签。
- **部署对象**：需要部署容器应用的边缘节点。  
单击“选择边缘节点”，勾选需要选择的边缘节点。

您还可以单击页面右上角的“标签搜索”，输入标签名和标签值，单击“搜索”，筛选出指定标签的边缘节点。然后勾选边缘节点，单击“确定”。

图 3-69 选择标签

名称/ID	状态	主机名/网维	创建时间
ecs-ief-ubuntu_fa-16-3e-28-07-a0 209ba62d-2588-4476-aec1-6617e93bad14	运行中	ecs-ief-ubuntu 192.168.0.204	2021/08/05 14:40:17 GMT+08:00

- **重启策略**

- 总是重启：当应用容器退出时，无论是正常退出还是异常退出，系统都会重新拉起应用容器。
- 失败时重启：当应用容器异常退出时，系统会重新拉起应用容器，正常退出时，则不再拉起应用容器。
- 不重启：当应用容器退出时，无论是正常退出还是异常退出，系统都不再重新拉起应用容器。

### 📖 说明

只有“总是重启”的容器应用，才能在创建后更新升级。

- **Host PID**：容器与宿主机共享PID命名空间，启用此选项要求边缘节点软件版本大于等于2.8.0。

**步骤4** 单击“下一步”，配置容器信息。

图 3-70 容器信息

服务实例  专业版服务实例

创建方式 自定义边缘应用

名称前缀

实例数量 1

配置方式  自定义配置  应用模板配置

部署描述  0/255

标签  继承边缘节点共性标签

还可以创建20个标签。

- **名称前缀**：应用部署的名称前缀。IEF会根据前缀生成完整的名称。
- **实例数量**：应用部署的实例数量，专业版只有1个实例。
- **配置方式**
  - 自定义配置：即从零开始配置容器应用，具体请参见**步骤5**。
  - 应用模板配置：选择一个已经定义好的应用模板，可以在模板的基础上进行修改，使用应用模板能够帮助您省去重复的工作量。模板的定义与**步骤5**需要的配置相同，创建模板的方法请参见**应用模板**。
- **部署描述**：输入容器应用描述。
- **标签**：容器应用标签，可勾选“继承边缘节点共性标签”。

**步骤5** 配置容器。

选择需要部署的镜像，单击“使用镜像”。

- **我的镜像**：展示了您在**容器镜像服务**中创建的所有镜像。
- **他人共享**：展示了其他用户共享的镜像，共享镜像是在SWR中操作的，具体请参见**共享私有镜像**。

选择镜像后，您可以配置容器的规格。

- **镜像版本**：请选择需要部署的镜像版本。

#### 须知

在生产环境中部署容器时，应避免使用latest版本。因为这会导致难以确定正在运行的镜像版本，并且难以正确回滚。

- **容器规格**：据需要选择容器CPU、内存、昇腾AI加速卡和GPU的配额。
- **昇腾AI加速卡容器应用**选择的AI加速卡配置与实际部署的边缘节点配置的AI加速卡必须一致，否则会创建应用失败，详见[注册边缘节点时AI加速卡配置](#)。

#### 说明

虚拟化切分后的NPU类型，一个容器只能挂载一个虚拟化NPU，只有当该容器退出后，该虚拟化NPU才能分配给其他容器使用。

昇腾AI加速卡支持的NPU类型，如下表。

表 3-13 NPU 类型说明

类型	描述
昇腾310	昇腾310芯片
昇腾310B	昇腾310B芯片

图 3-71 容器配置

镜像名称: wordpress [更换镜像](#)

\* 镜像版本: php7.3

\* 容器名称: container-047e662c

容器规格

CPU配额  申请: 0.25 Core  限制: 0.25 Core

内存配额  申请: 512.00 MiB  限制: 512.00 MiB

AI加速卡:

您还可以对容器进行如下高级配置。

- **运行命令**

容器镜像拥有存储镜像信息的相关元数据，如果不设置生命周期命令和参数，容器运行时运行镜像制作时提供的默认的命令和参数，Dockerfile这两个字段为“Entrypoint”和“CMD”。

如果在创建容器应用时填写了容器的运行命令和参数，将会覆盖镜像构建时的默认命令“Entrypoint”、“CMD”，规则如下：

表 3-14 容器如何执行命令和参数

镜像 Entrypoint	镜像CMD	容器运行命令	容器运行参数	最终执行
[touch]	[/root/test]	未设置	未设置	[touch /root/test]
[touch]	[/root/test]	[mkdir]	未设置	[mkdir]
[touch]	[/root/test]	未设置	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

图 3-72 运行命令



- 运行命令

输入可执行的命令，例如/run/start。

若可执行命令有多个，多个命令之间用空格进行分隔。若命令本身带空格，则需要加引号（" "）。

**说明**

多命令时，运行命令建议用/bin/sh或其他shell，其他全部命令作为参数来传入。

- 运行参数

输入控制容器运行命令的参数，例如--port=8080。

若参数有多个，多个参数以换行分隔。

• 安全选项

- 可以通过开启“特权选项”，使容器拥有root权限，可以访问主机上的设备（如GPU、FPGA）。

- 指定运行用户

IEF默认不改变容器运行的用户，以构建镜像时定义的运行用户来运行。

打开开关后，下方出现运行用户的输入框，可输入范围为0~65534的整数。

指定了运行用户后，应用将以该运行用户来运行。如果镜像的操作系统中没有该用户ID，将导致容器应用启动失败。

• 环境变量配置

容器运行环境中设定的变量。可以在应用部署后修改，为应用提供极大的灵活性。当前支持手动添加、密钥导入、配置项导入和变量引用方式。

- 手动添加支持自定义变量名称和变量值。
- 使用密钥导入，环境变量名称可自定义输入，环境变量值支持引用密钥的属性值，密钥创建方法请参见[密钥](#)。
- 使用配置项导入，环境变量名称可自定义输入，环境变量值支持引用配置项的属性值，配置项创建方法请参见[配置项](#)。
- 变量引用支持引用“hostIP”，即边缘节点的IP地址。

### 📖 说明

IEF不会对用户输入的环境变量进行加密。如果用户配置的环境变量涉及敏感信息，用户需要自行加密后再填入，并在应用中自己完成解密过程。

IEF也不提供任何加解密工具，如果您需要设置加密密文，可以使用其他加解密工具。

### ● 数据存储

您可以通过定义本地卷，将边缘节点本地存储目录挂载到容器中，以实现数据文件的持久化存储。

当前支持如下四种类型的本地卷。

- hostPath：将主机某个目录挂载到容器中。hostPath是一种持久化存储，应用删除后hostPath里面的内容依然存在于边缘节点本地硬盘目录中，如果后续重新创建应用，挂载后依然可以读取到之前写入的内容。

您可以将应用日志目录挂载到主机的“/var/IEF/app/log/{appName}”目录，“{appName}”是应用名，边缘节点会将“/var/IEF/app/log/{appName}”目录下后缀为log和trace的文件上传到AOM。

挂载目录为容器内应用的日志路径，如nginx应用默认的日志路径为/var/log/nginx；权限需配置为“读写”。

图 3-73 日志卷挂载

本地卷名称	类型	挂载目录	权限	操作
log	hostPath	/var/IEF/app/log/nginx	/var/log/nginx	读写 删除

- emptyDir：一种简单的空目录，主要用于临时存储，支持在硬盘或内存中创建。emptyDir挂载后就是一个空目录，应用程序可以在里面读写文件，emptyDir的生命周期与应用相同，应用删除后emptyDir的数据也同时删除掉。
- configMap：存储应用所需配置信息的资源类型，创建方法请参见[配置项](#)。
- secret：密钥是一种用于存储应用所需的认证信息、证书、密钥等敏感信息的资源类型，创建方法请参见[密钥](#)。

### 须知

- 请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，应用创建失败。
- 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。

### ● 健康检查

健康检查是指容器运行过程中根据用户需要定时检查容器健康状况或是容器中负载的健康状况。

- 应用存活探针：应用存活探针用于探测容器是否正常工作，不正常则重启实例。当前支持发送HTTP请求和执行命令检查，检测容器响应是否正常。
- 应用业务探针：应用业务探针用于探测业务是否就绪，如果业务还未就绪，就不会将流量转发到当前实例。

详细的配置说明请参见[健康检查配置说明](#)。

**步骤6** 单击“下一步”，进行访问配置。

容器访问支持主机网络和端口映射两种方式。

- **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。

端口映射可以选择主机网卡，请注意端口映射不支持选择IPv6地址类型的网卡。

**图 3-74** 端口映射



- **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。

**步骤7** 单击“下一步”，确认容器应用的规格，确认无误后单击“创建”。

----结束

## 状态说明

批量应用部署作业有以下八种状态。

- **排队中**：作业等待执行
- **执行中**：作业处于执行状态
- **成功**：全部任务执行成功
- **部分成功**：部分任务执行成功
- **失败**：全部任务执行失败
- **停止中**：作业处于停止中
- **已停止**：作业已停止
- **更新超时**：作业排队和执行时间超过10分钟仍未完成

批量作业执行过程中可以停止，停止后可以继续。

如果批量作业执行失败、部分成功或更新超时，可以重试执行作业，将未执行成功的作业再次执行一遍。

## 3.5.4 批量应用升级

### 说明

容器镜像的架构必须与节点架构一致，比如节点为x86，那容器镜像的架构也必须是x86。

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“批量管理 > 应用升级”，单击页面右上角的“批量应用升级”。

**步骤3** 配置应用升级作业基本信息。

图 3-75 批量应用升级

The screenshot shows a web form for configuring a batch application upgrade. It has three main input sections: '名称' (Name) with a text box, '描述' (Description) with a larger text area, and '标签' (Tags) with two text boxes. A note at the bottom indicates that selected applications will use the same configuration. A '升级对象' (Upgrade Object) button is located at the bottom left.

- **名称**：创建的应用升级作业名称。
- **描述**：应用升级作业描述。
- **标签**：应用升级作业的标签。
- **升级对象**：需要升级的容器应用。

单击“请选择应用部署”，选择需要升级的容器应用。

您还可以单击页面右上角的“标签搜索”，输入标签名和标签值，单击“搜索”，筛选出指定标签的容器应用。然后勾选需要升级的容器应用，单击“确定”。

**步骤4** 单击“下一步”，配置容器，此处的配置与**步骤5**相同。

**步骤5** 单击“下一步”，进行访问配置，此处的配置与**步骤6**相同。

**步骤6** 单击“下一步”，确认容器应用的规格，确认无误后单击“创建”。

----结束

## 状态说明

批量应用升级作业有以下八种状态。

- **排队中**：作业等待执行
- **执行中**：作业处于执行状态
- **成功**：全部任务执行成功
- **部分成功**：部分任务执行成功



- **失败**: 全部任务执行失败
- **停止中**: 作业处于停止中
- **已停止**: 作业已停止
- **更新超时**: 作业排队和执行时间超过10分钟仍未完成

批量作业执行过程中可以停止，停止后可以继续。

如果批量作业执行失败、部分成功或更新超时，可以重试执行作业，将未执行成功的作业再次执行一遍。

## 3.6 审计

### 3.6.1 支持云审计的关键操作

#### 操作场景

云审计服务（Cloud Trace Service, CTS），是华为云安全解决方案中专业的日志审计服务，提供对各种云资源操作记录的收集、存储和查询功能，可用于支撑安全分析、合规审计、资源跟踪和问题定位等常见应用场景。

通过云审计服务，您可以记录与IEF相关的操作事件，便于日后的查询、审计和回溯。

#### 支持审计的关键操作列表

表 3-15 云审计服务支持的 IEF 操作列表

操作名称	资源类型	事件名称
注册边缘节点	node	createEdgeNode
更新边缘节点	node	updateEdgeNode
删除边缘节点	node	deleteEdgeNode
创建设备模板	deviceTemplate	createDeviceTemplate
更新设备模板	deviceTemplate	updateDeviceTemplate
删除设备模板	deviceTemplate	deleteDeviceTemplate
注册设备	device	createDevice
更新设备	device	updateDevice
删除设备	device	deleteDevice
部署Edge-Connector	device	deployConnector
更新设备孪生	device	updateDeviceTwin
更新设备访问配置	device	updateDeviceAccessConfig
创建应用模板	application	createApplication

操作名称	资源类型	事件名称
更新应用模板	application	updateApplication
删除应用模板	application	deleteApplication
创建应用模板版本	appVersion	createAppVersion
更新应用模板版本	appVersion	updateAppVersion
删除应用模板版本	appVersion	deleteAppVersion
创建配置项	configmap	createConfigMap
更新配置项	configmap	updateConfigMap
删除配置项	configmap	deleteConfigMap
创建容器应用	deployment	createDeployment
更新容器应用	deployment	updateDeployment
删除容器应用	deployment	deleteDeployment
查询容器应用列表	deployment	getDeploymentList
查询容器应用	deployment	getDeployment
创建端点	endpoint	createEndpoint
删除端点	endpoint	deleteEndpoint
添加节点证书	node	AddNodeCert
删除节点证书	node	deleteNodeCert
升级固件	nodeFirmware	UpgradeNodeFirmware
查询应用实例列表	Pods	getPods
查询实例	Pod	getPod
创建节点注册作业	product	createProduct
删除节点注册作业	product	deleteProduct
创建节点升级作业	batchJob	createJob
停止节点升级作业	batchJob	pauseJob
删除节点升级作业	batchJob	deleteJob
查询配额	quota	getQuota
更新配额	quota	updateQuota
创建规则	rule	createRule
删除规则	rule	deleteRule
更新规则	rule	updateRule

操作名称	资源类型	事件名称
创建密钥	secret	createSecret
更新密钥	secret	updateSecret
删除密钥	secret	deleteSecret
过滤标签	Tags	filterTags
批量添加删除标签	Tags	batchAddDeleteTags
添加标签	Tags	addTag
删除标签	Tags	deleteTag
创建系统订阅	Systemevent	createSystemevent
删除系统订阅	Systemevent	DeleteSystemevent
启用系统订阅	Systemevent	startSystemevent
停用系统订阅	Systemevent	stopSystemevent

## 3.6.2 如何查看审计日志

### 操作场景

开启了云审计服务（CTS）后，系统开始记录IEF相关的操作。CTS会保存最近1周的操作记录。

本小节介绍如何在CTS管理控制台查看最近1周的操作记录。

### 操作步骤

**步骤1** 登录CTS管理控制台。


**步骤2** 选择左侧导航栏的“事件列表”，进入事件列表页面。

**步骤3** 事件记录了云资源的操作详情，设置筛选条件，单击“查询”。

当前事件列表支持四个维度的组合查询，详细信息如下：

- 事件类型、事件来源、资源类型和筛选类型。  
在下拉框中选择查询条件。其中，“事件类型”选择“管理事件”，“事件来源”选择“IEF”。  
其中，
  - 筛选类型选择“按资源ID”时，还需手动输入某个具体的资源ID，目前仅支持全字匹配模式的查询。
  - 筛选类型选择“按资源名称”时，还需选择或手动输入某个具体的资源名称。
- 操作用户：在下拉框中选择某一具体的操作用户。
- 事件级别：可选项为“所有事件级别”、“Normal”、“Warning”、“Incident”，只可选择其中一项。

- 时间范围：可选项为“最近1小时”、“最近1天”、“最近1周”和“自定义时间段”，本示例选择“最近1周”。

**步骤4** 在需要查看的事件左侧，单击  图标展开该事件的详细信息。

**步骤5** 在需要查看的事件右侧，单击“查看事件”，弹出一个窗口，显示了该操作事件结构的详细信息。

----结束

## 3.7 权限管理

### 3.7.1 创建用户并授权使用 IEF

如果您需要对您所拥有的智能边缘平台（IEF）进行精细的权限管理，您可以使用[统一身份认证服务](#)（Identity and Access Management，简称IAM），通过IAM，您可以：

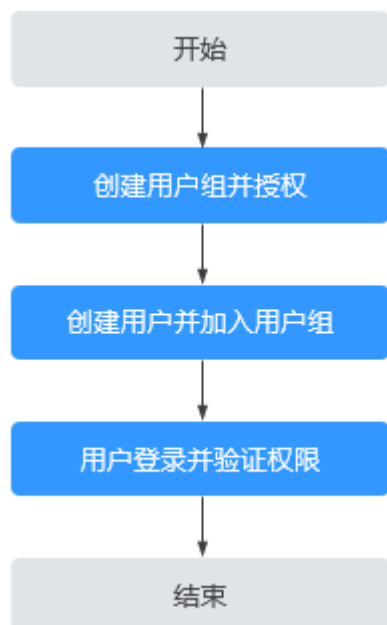
将IEF资源委托给更专业、高效的其他云账号或者云服务，这些账号或者云服务可以根据权限进行代运维。

如果云账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用IEF服务的其它功能。

本章节为您介绍对用户授权的方法，操作流程如[图3-76](#)所示。

#### 示例流程

图 3-76 给用户授权 IEF 权限流程



#### 1. 创建用户组并授权

在IAM控制台创建用户组，并授予IEF只读权限“IEF ReadOnlyAccess”。为用户组授权时，作用范围选择“区域级项目”，然后根据以下原则设置：

- 在个别区域授权：选择指定的一个或多个项目，例如“cn-north-4 [华北-北京四]”。注意：此场景选择“所有项目”时，授权将不生效。
- 在所有区域授权：选择“所有项目”。

图 3-77 在个别区域授权

在以下作用范围

全局服务  
访问全局级服务时，不需要切换区域。授权后在全局区域生效。如对象存储服务（OBS）。 [了解更多...](#)

区域级项目  
访问项目级服务时，需要切换至被授权项目。授权后只在授权区域生效。如需在所有区域都生效，请选择所有项目。如弹性云服务器（ECS）。 [了解更多...](#)

cn-north-4 [华北-北京四]

图 3-78 在所有区域授权

在以下作用范围

全局服务  
访问全局级服务时，不需要切换区域。授权后在全局区域生效。如对象存储服务（OBS）。 [了解更多...](#)

区域级项目  
访问项目级服务时，需要切换至被授权项目。授权后只在授权区域生效。如需在所有区域都生效，请选择所有项目。如弹性云服务器（ECS）。 [了解更多...](#)

所有项目 (包括未来在所有区域下创建的项目)

## 2. 创建用户并加入用户组

在IAM控制台创建用户，并将其加入[1.创建用户组并授权](#)中创建的用户组。

## 3. 用户登录并验证权限。

新创建的用户登录控制台，验证IEF服务的管理员权限。

## 3.7.2 自定义策略

IEF可以创建自定义策略。

目前华为云支持以下两种方式创建自定义策略：

- 可视化视图创建自定义策略：无需了解策略语法，按可视化视图导航栏选择云服务、操作、资源、条件等策略内容，可自动生成策略。
- JSON视图创建自定义策略：可以在选择策略模板后，根据具体需求编辑策略内容；也可以直接在编辑框内编写JSON格式的策略内容。

具体创建步骤请参见：[创建自定义策略](#)。本章为您介绍常用的IEF自定义策略样例。

### IEF 自定义策略样例

授权用户创建、更新应用和应用模板的权限。

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ief:deployment:create",
        "ief:appVersion:update",
        "ief:deployment:update",
        "ief:application:create"
      ],
      "Condition": {
        "StringEquals": {
          "ief:AssumeUserName": [
            "test"
          ]
        }
      }
    }
  ]
}
```

```

    }
  },
  "Resource": [
    "ief:*:deployment:*",
    "ief:*:appVersion:*",
    "ief:*:application:*"
  ]
}
]
}
}

```

### 3.7.3 IEF 资源

资源是服务中存在的对象。在IEF中，资源包括：product、node、group、deployment、batchjob、application、appVersion、IEFInstance，您可以在创建自定义策略时，通过指定资源路径来选择特定资源。

表 3-16 IEF 的指定资源与对应路径

指定资源	资源名称	资源路径
product	节点注册作业	IEF:*:product:
node	边缘节点	IEF:*:node:
group	边缘节点组	IEF:*:group:
deployment	应用部署	IEF:*:deployment:
batchjob	批量作业	IEF:*:batchjob:
application	应用模板	IEF:*:application:
appVersion	应用模板版本	IEF:*:appVersion:
IEFInstance	IEF实例	IEF:*:IEFInstance:

### 3.7.4 IEF 请求条件

您可以在创建自定义策略时，通过添加“请求条件”（Condition元素）来控制策略何时生效。请求条件包括条件键和运算符，条件键表示策略语句的Condition元素，分为全局级条件键和服务级条件键。**全局级条件键**（前缀为g:）适用于所有操作，**服务级条件键**（前缀为服务缩写，如ief）仅适用于对应服务的操作。运算符与条件键一起使用，构成完整的条件判断语句。

IEF通过IAM预置了一组条件键，例如，您可以先使用ief:AssumeUserName条件键检查使用者的用户名称，然后再允许执行操作。下表显示了适用于IEF服务特定的条件键。

表 3-17 IEF 请求条件

IEF条件键	运算符	描述
ief:AssumeUserName	StringEndWithAnyOfIfExists StringStartWithAnyOfIfExists StringEndWithIfExists StringStartWithIfExists StringNotLikeAnyOfIfExists StringLikeAnyOfIfExists StringNotEqualsIgnoreCaseAnyOfIfExists StringEqualsIgnoreCaseAnyOfIfExists StringNotEqualsAnyOfIfExists StringEqualsAnyOfIfExists StringNotLikeIfExists StringLikeIfExists StringNotEqualsIgnoreCaseIfExists StringEqualsIgnoreCaseIfExists StringNotEqualsIfExists StringEqualsIfExists IsNullOrEmpty StringEndWithAnyOf StringStartWithAnyOf StringEndWith StringStartWith StringNotLikeAnyOf StringLikeAnyOf StringNotEqualsIgnoreCaseAnyOf StringEqualsIgnoreCaseAnyOf StringNotEqualsAnyOf StringEqualsAnyOf StringNotLike StringLike StringNotEqualsIgnoreCase StringEqualsIgnoreCase StringNotEquals StringEquals	匹配用户名称

## 示例

当用户名为test的时候，才可以使用该策略。

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ief:deployment:create",
        "ief:appVersion:update",
        "ief:deployment:update",
        "ief:application:create"
      ],
      "Condition": {
        "StringEquals": {
          "ief:AssumeUserName": [
            "test"
          ]
        }
      },
      "Resource": [
        "ief:*:deployment:*",
        "ief:*:appVersion:*",
        "ief:*:application:*"
      ]
    }
  ]
}
```



# 4 铂金版操作指南

---

## 4.1 节点管理

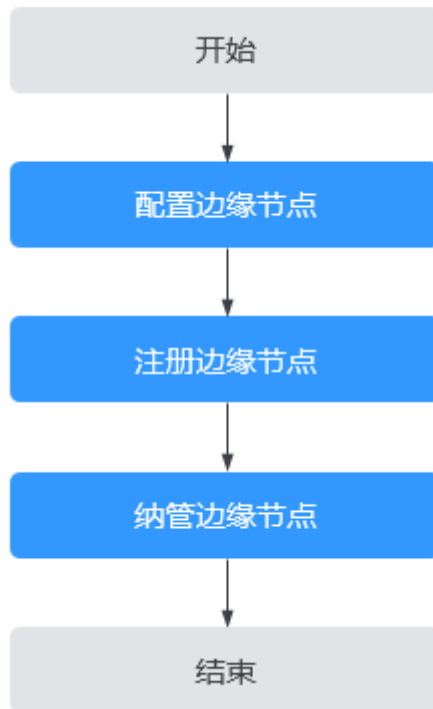
### 4.1.1 边缘节点概述

边缘节点是您自己的边缘计算机，用于运行边缘应用，处理您的数据，并安全、便捷地和云端应用进行协同。您可以通过智能边缘平台部署系统应用来延伸云服务能力到边缘节点，或者通过部署您自己的应用来构建您自己的边缘计算能力。

为了使智能边缘平台能够管理您的边缘节点，您需要做如下步骤：

1. 准备边缘节点，边缘节点需要满足一定的规格要求，具体请参见[配置边缘节点环境](#)。
2. 在IEF中创建边缘节点，获取边缘节点的安装工具和配置文件，具体请参见[注册边缘节点](#)。
3. 使用上一步获取的安装工具和配置文件纳管边缘节点，具体请参见[纳管边缘节点](#)。

图 4-1 使用流程



## 4.1.2 配置边缘节点环境

### 边缘节点规格要求

边缘节点既可以是物理机，也可以是虚拟机。边缘节点需要满足表4-1的规格要求。

表 4-1 边缘节点要求

项目	规格
OS	<p>操作系统语言必须切换至<b>英文</b>。</p> <ul style="list-style-type: none"> <li>x86_64架构 Ubuntu LTS (Xenial Xerus)、Ubuntu LTS (Bionic Beaver)、CentOS、EulerOS、RHEL、银河麒麟、中兴新支点、中标麒麟、openEuler、uos (Unity Operating System)、ol (Oracle Linux)、hce (Huawei Cloud Euler)、openEuler 23.09 Edge</li> <li>armv7i ( arm32 ) 架构 Raspbian GNU/Linux (stretch)</li> <li>aarch64 ( arm64 ) 架构 Ubuntu LTS (Bionic Beaver)、CentOS、EulerOS、openEuler、uos (Unity Operating System)、ol (Oracle Linux)、hce (Huawei Cloud Euler)、openEuler 23.09 Edge</li> </ul> <p><b>说明</b> 推荐使用面向边缘计算场景的openEuler 23.09 Edge操作系统。</p>

项目	规格
内存	边缘软件开销约128MB，为保证业务的正常运行，建议边缘节点的内存大于256MB。
CPU	>= 1核
硬盘	>= 1GB
GPU（可选）	同一个边缘节点上的GPU型号必须相同。 <b>说明</b> 当前支持Nvidia Tesla系列P4、P40、T4等型号GPU。 含有GPU硬件的机器，作为边缘节点的时候可以不使用GPU。 如果边缘节点使用GPU，您需要在纳管前安装GPU驱动。 目前只有使用x86架构的GPU节点才能纳管到IEF中使用。
NPU（可选）	华为昇腾AI加速处理器。 <b>说明</b> 当前支持集成了华为昇腾处理器的边缘节点，如Atlas 300推理卡、Atlas 800推理服务器。同时支持昇腾310P、昇腾310B、昇腾310P共享模式和虚拟化切分的NPU规格。 如果边缘节点使用NPU，请确保边缘节点已安装驱动（目前昇腾310仅支持1.3.x.x和1.32.x.x的固件版本，例如1.3.2.B893，可用npusmi info命令查看固件版本）（NPU驱动需不小于22.0.4版本，进入驱动所在路径如“/usr/local/Ascend/driver”，执行cat version.info命令查看）。如果没有安装驱动，请联系设备厂商获取支持。
容器引擎	Docker版本必须高于17.06。使用高于或等于1.23版本的docker时，需设置docker cgroupfs版本为1，不支持docker HTTP API v2。 （请勿使用18.09.0版本Docker，该版本存在严重bug，详见 <a href="https://github.com/docker/for-linux/issues/543">https://github.com/docker/for-linux/issues/543</a> ；如果已使用此版本，请尽快升级。） <b>须知</b> Docker安装完成后，请将Docker进程配置为开机启动，避免系统重启后Docker进程未启动引起的系统异常。 Docker Cgroup Driver必须设置为cgroupfs。详细配置方法请参考在 <a href="#">边缘节点安装Docker后，如何设置Docker Cgroup Driver?</a> 。
glibc	版本必须高于2.17。
端口使用	边缘节点需要使用8883端口，8883端口用于边缘节点内置MQTT broker监听端口，请确保该端口能够正常使用。
时间同步	边缘节点时间需要与UTC标准时间保持一致，否则会导致边缘节点的监控数据、日志上传出现偏差。您可以选择合适的NTP服务器进行时间同步，从而保持时间一致。详细配置方法请参见 <a href="#">如何同步NTP服务器?</a> 。

## 配置边缘节点环境

**步骤1** 以具备sudo权限的用户登录边缘节点。

**步骤2 GPU驱动配置。**

如果边缘节点使用GPU，您需要安装并配置GPU驱动，详细方法请参见[安装并配置GPU驱动](#)。

**步骤3 NPU驱动配置。**

如果边缘节点使用华为昇腾 AI加速处理器，请确保已安装对应驱动。

**步骤4 在边缘节点上安装Docker并检查Docker状态。**

Docker版本必须高于17.06，推荐使用18.06.3版本。请勿使用18.09.0版本Docker，该版本存在严重bug，如果已使用此版本，请尽快升级。

Docker安装完成后，可以执行**docker -v**命令检查Docker是否安装正常，如果回显如下则说明安装正常。

```
# docker -v
Docker version 19.03.12, build 48a66213fee
```

**步骤5 配置边缘节点防火墙规则。**

检查边缘节点防火墙状态。

```
systemctl status firewalld
firewall-cmd --state
```

回显中，not running表示关闭，running表示开启。

如果防火墙开启，您需要打开8883端口，或关闭防火墙。

- 打开8883端口。

```
firewall-cmd --add-port=8883/tcp --permanent
systemctl restart firewalld
```
- 关闭防火墙。

```
systemctl disable firewalld
systemctl stop firewalld
```

----**结束**

## 安全提示

为提升主机安全性，建议您对边缘节点进行操作系统加固，可参考：

1. 设置所有OS系统密码（包括管理员和普通用户）、数据库账号密码、应用（WEB）系统管理账号密码为强密码，密码12位以上。
2. 应用程序不以管理员权限账号运行，应用程序（如Web）不使用数据库管理员权限账号与数据库交互。设置安全组，仅向公网开放必要端口，业务WEB控制台端口、局域网内部通信端口避免暴露在公网。关闭高危端口（如SSH端口），或采取限制允许访问端口的源IP、使用VPN/堡垒机建立的运维通道等措施消减风险。
3. 业务数据定期异地备份，避免黑客入侵主机造成数据丢失。
4. 定期检测系统和软件中的安全漏洞，及时更新系统安全补丁，将软件版本升级到官方最新版本。
5. 建议从官方渠道下载安装软件，对非官方渠道下载的软件，建议使用杀毒软件扫描后再运行。

如果使用的是华为云 ECS，可参考：

1. 将主机登录方式设置为密钥登录。

2. 使用华为云官方提供的[企业主机安全服务](#)深度防御。

### 4.1.3 注册边缘节点

注册边缘节点就是在IEF设定边缘节点的配置，并获取边缘节点配置文件和安装程序。

#### 约束与限制

- 同一节点上的NPU，只支持同一种切分规格。
- 暂不支持已纳管的节点，通过升级NPU插件到2.1.0版本的方式支持虚拟化切分。用户需要将边缘节点进行卸载重装，手动切分NPU后，再纳管至IEF。
- NPU驱动需大于22.0版本，进入驱动所在路径（如“/usr/local/Ascend/driver”），执行cat version.info命令查看。
- 选择共享模式的情况下，需要在NPU卡上设置所有芯片的容器共享模式。
- D310P共享模式必须使用完整的卡，不支持虚拟化切分。
- 健康芯片数量为物理芯片数，如节点使用了切分或者共享模式，则展示的是切分后或者共享后的芯片数目。

#### 注册边缘节点

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“边缘资源 > 边缘节点”，单击页面右上角的“注册边缘节点”。

**步骤3** 配置边缘节点基本信息。

图 4-2 边缘节点基本信息（1）

服务实例 platinum

名称

描述  0/255

标签    
还可以创建20个标签。

如需部署昇腾应用或者GPU应用, 请根据选择切换AI加速卡类型。 ([了解边缘节点规格要求](#))

AI加速卡  不启用  昇腾AI加速卡  Nvidia GPU

设备	设备与节点的关系	备注	操作
<input type="button" value="绑定设备"/>			

是否启用docker  是  否

监听地址  网卡  IP

tls://	<input type="text" value="to"/>	:	<input type="text" value="8883"/>	<input type="button" value="删除"/>
tls://	<input type="text" value="docker0"/>	:	<input type="text" value="8883"/>	<input type="button" value="删除"/>
<input type="button" value="新增监听地址"/>				

- **名称**: 边缘节点的名称。允许中文、英文字母、数字、中划线、下划线，最小长度1，最大长度64。

- **描述**: 选填, 请输入边缘节点描述信息。
- **标签**: 标签可用于对资源进行标记, 方便分类管理。
- **AI加速卡**  
昇腾AI加速卡: 支持华为昇腾处理器(即NPU)的边缘节点, 支持昇腾310P和昇腾310P芯片虚拟化切分的NPU规格。如果使用华为昇腾310、310B、910、310P、310P-共享模式、310P-1c等芯片, 请选择“昇腾AI加速卡”, 然后选择NPU类型。  
AI加速卡支持的NPU类型, 如下表4-2。

表 4-2 NPU 类型说明

类型	描述
昇腾310	昇腾310芯片
昇腾310B	昇腾310B芯片
昇腾910	昇腾910芯片
昇腾310P	昇腾310P芯片
昇腾310P-共享模式	昇腾310P芯片支持多容器共享
昇腾310P-1C	昇腾310P芯片按照模板名vir01虚拟化切分后的类型
昇腾310P-2C	昇腾310P芯片按照模板名vir02虚拟化切分后的类型
昇腾310P-2C.1CPU	昇腾310P芯片按照模板名vir02_1c虚拟化切分后的类型
昇腾310P-4C	昇腾310P芯片按照模板名vir04虚拟化切分后的类型。
昇腾310P-4C.3CPU	昇腾310P芯片按照模板名vir04_3c虚拟化切分后的类型

Nvidia GPU: 如果您的边缘节点搭载了Nvidia GPU显卡, 请选择“Nvidia GPU”。

不启用: 边缘节点未使用AI加速卡时选择。

#### 📖 说明

如果边缘节点上没有搭载Nvidia GPU显卡, 而这里选择了启用“Nvidia GPU”, 则纳管边缘节点会失败。

如果边缘节点使用GPU, 您需要在纳管前安装并配置GPU驱动, 详细方法请参见[安装并配置GPU驱动](#)。

- **绑定设备**: 为边缘节点绑定终端设备, 如果您还没有注册终端设备, 请参见[终端设备管理](#)注册终端设备。终端设备在注册边缘节点后仍然可以绑定。
- **是否启用docker**: 启用后可以支持部署容器应用。
- **监听地址**:

边缘节点内置的MQTT broker的监听地址，用于发送和接收边云消息。边云消息的使用请参见[设备孪生工作原理](#)和[边云消息概述](#)。

默认监听lo（localhost）和docker0两个本地网卡，您可以通过指定网卡名或IP地址设置需要监听的网卡，还可以增加其他需要监听的网卡或IP地址。

图 4-3 边缘节点基本信息（2）

i 应用运维管理服务AOM每月赠送500M免费日志存储额度，超出部分将按需收费 [\(了解计费详情\)](#)

---

系统日志 应用日志

请合理设置日志文件大小和滚动数量避免过多占用节点存储。

日志文件大小(MB) ?

滚动日志周期 ?

滚动日志数量 ?

是否开启云端日志

当前支持配置边缘节点的系统日志和应用日志。

- **系统日志**：边缘节点上IEF软件（如edge-core、edge-logger和edge-monitor等）产生的日志。
- **应用日志**：边缘节点上部署的应用所产生的日志。

系统日志和应用日志需要配置如下几个参数：

- **日志文件大小**：日志文件大小限制，单位MB，默认50，取值范围10-1000。某个日志文件如果达到大小限制，则会转储。
  - 系统日志保存在边缘节点“/var/IEF/sys/log/”目录下，然后转储到AOM。
  - 应用日志会将容器的标准输出和挂载到边缘节点“/var/IEF/app/log”的日志转储到AOM。
- **滚动日志周期**：日志转储周期，可选项：每天、每周、每月、每年。日志文件大小和滚动日志周期是同时生效的，满足任何一个条件都会进行日志转储。
- **滚动日志数量**：日志文件转储个数，默认5，取值范围1-10。边缘节点保存的转储日志数量如果达到限制，则会删除最老的那个转储文件。
- **是否开启云端日志**：  
您可以根据需要控制是否上传日志到AOM服务，开启之后您可以在AOM中查看日志，具体请参见[在AOM查看日志](#)。

**步骤4** 勾选“我已经阅读并同意《华为云服务等级协议》”，单击页面右下角的“注册”。

通过证书注册。下载配置文件和边缘节点安装工具，在后续[纳管边缘节点](#)时将用到这些。

图 4-4 下载配置文件和边缘核心软件



1. 根据页面提示，单击“下载 边缘节点名称.tar.gz 配置文件”下载配置文件。
2. 根据您边缘节点的CPU架构选择边缘节点安装工具，单击“下载EdgeCore Installer”。

**步骤5** 在右下角勾选“我已完成下载”，并单击“完成”。

您可以看到边缘节点的状态为“未纳管”，这是因为还未安装[注册边缘节点](#)下载的边缘节点安装工具，请参见[纳管边缘节点](#)纳管节点。

图 4-5 未纳管的边缘节点

<input type="checkbox"/>	名称/ID	状态	主机名/网络	应用实例(正常/全部)	创建时间	边缘侧软件版本	节点标签	节点类型	操作
<input type="checkbox"/>	ief-node 18c37006-8bbb-4a9a-9e5	未纳管 安装指南	--	0/0	2021/07/19 18:39:55 GMT...	--	--	自建节点	删除 更多 ▾

----结束

## 后续操作

完成注册后，您需要对边缘节点进行纳管，具体请参见[纳管边缘节点](#)。

### 4.1.4 纳管边缘节点

纳管边缘节点就是在实际的边缘节点上使用[注册边缘节点](#)中下载的安装程序和配置文件，安装边缘核心软件EdgeCore，这样边缘节点就能与IEF连接，纳入IEF管理。

边缘节点初次纳管时，智能边缘平台自动安装最新版本的边缘核心软件EdgeCore。例如，当前IEF边缘软件有2.51.0、2.52.0、2.53.0三个版本，当您初次纳管节点时，IEF将推送最新版本2.53.0给您的边缘节点进行安装。

#### 📖 说明

在IEF上注册的边缘节点与实际的边缘节点机器是一一对应的关系，一个边缘节点的安装工具和配置文件只能安装在一台实际的边缘节点上。

## 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[配置边缘节点环境](#)。
- 已经注册好节点并获取到节点配置文件和安装工具，具体请参见[注册边缘节点](#)。



## 纳管边缘节点

**步骤1** 以具备sudo权限的用户登录边缘节点。

**步骤2** 将[注册边缘节点](#)下载的边缘节点安装工具和配置文件上传到边缘节点指定目录，例如“/home”目录，并进入该目录。

**步骤3** 执行如下命令，解压缩安装工具到“/opt”文件夹。

```
sudo tar -zxvf edge-installer_1.0.0_x86_64.tar.gz -C /opt
```

`edge-installer_1.0.0_x86_64.tar.gz`请替换为[注册边缘节点](#)下载的安装工具。

**步骤4** 执行如下命令，解压缩配置文件到“opt/IEF/Cert”目录。如果纳管的是“通过token注册”的边缘节点，请跳过此步骤。

```
sudo mkdir -p /opt/IEF/Cert; sudo tar -zxvf 边缘节点名称.tar.gz -C /opt/IEF/Cert
```

`边缘节点名称.tar.gz`请替换为[注册边缘节点](#)下载的配置文件。

**步骤5** 执行如下命令，纳管边缘节点。

- 通过证书注册

```
cd /opt/edge-installer; sudo ./installer -op=install
```

**步骤6** 验证边缘节点是否纳管成功。

1. 登录IEF管理控制台，在“总览”页面切换实例为铂金版。
2. 选择左侧导航栏的“边缘资源 > 边缘节点”。
3. 查看边缘节点的状态。当前状态为“运行中”表示纳管成功。

图 4-6 查看边缘节点状态

名称/ID	状态	主机名/网络	应用实例(正常/全部)	创建时间	边缘侧软件版本
ief-node 7092ad14-adee-4a09-b969-1	运行中	eth0:192.168.0.230	0/0	2021/07/20 09:33:26 GMT+08:00	2.52.0 可升级

----结束

### 须知

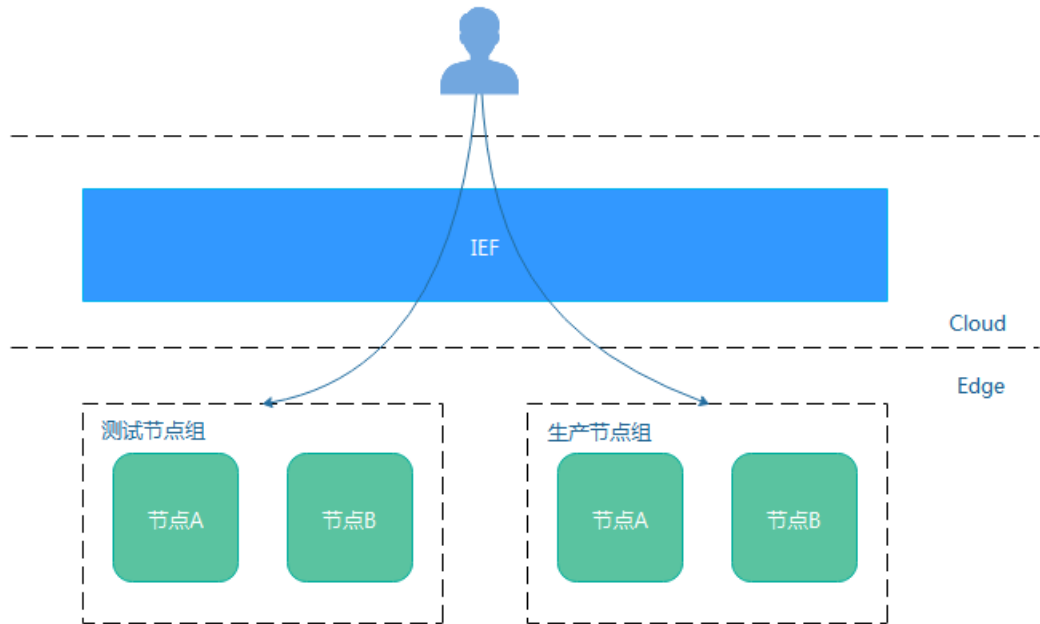
纳管后请勿删除边缘节点的“/opt”目录，否则需要重新注册边缘节点并纳管。

## 4.1.5 边缘节点组

边缘节点组可以将节点进行分组，并将边缘应用部署在节点组内，节点组有如下功能：

- 节点绑定/解绑节点组
- 根据节点组分配安全证书，用于访问节点组内节点及指定节点组部署的应用
- 应用指定节点组自动调度（根据节点组内资源使用量进行调度）
- 节点故障，应用重调度到其他正常节点
- 应用和节点的亲和/反亲和调度规则指定

图 4-7 节点组



## 创建边缘节点组

- 步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。
- 步骤2** 选择左侧导航栏的“边缘资源 > 边缘节点组”，单击页面右上角的“创建边缘节点组”。
- 步骤3** 配置节点组名称、描述，选择添加需要绑定到节点组的边缘节点，单击“创建”。  
节点组创建后也可以动态绑定/解绑节点。

图 4-8 创建边缘节点组

The screenshot shows a web interface for creating an edge node group. The title bar reads '创建边缘节点组'. The form contains the following elements:

- 服务实例**: cloud
- \* 节点组名称**: te
- 描述**: 请输入描述 (0/255)
- 标签**: 请输入标签名, 请输入标签值 (还可以创建20个标签。)
- 节点组类型**: 普通节点组 (selected), IEC 节点组
- 边缘节点**: 选择边缘节点

---结束

## 指定节点组部署应用

支持指定节点组部署应用，应用根据节点组内节点的资源选择优节点进行部署。部署应用具体请参见[容器应用](#)。

图 4-9 创建容器应用



- **故障策略**: 指应用实例所在的边缘节点不可用时, 是否将应用实例重新调度, 迁移到边缘节点组内的其他可用节点。

#### 📖 说明

节点组内不可用节点比例大于 0.55 时, 将停止自动迁移。

- **迁移时间窗**: 故障时间多长才触发迁移。

## 节点组证书

在节点组中部署的应用可能调度到节点组内任意节点, 终端设备可能访问节点组内任意节点, 从边缘节点处创建的证书不能满足应用访问节点 MQTT broker 的需要, 所以需要使用节点组证书。

#### 📖 说明

证书有效期为5年。

单击节点组名称, 进入详情页面。在“证书”页签下添加证书。

图 4-10 创建节点组证书



## 4.1.6 升级边缘节点

### 背景信息

边缘节点上安装的EdgeCore软件支持升级，IEF会不定期发布新版本，您可以根据需求升级边缘节点。

### 版本支持策略

IEF只负责维护发布周期一年内的边缘节点软件版本，建议您的边缘节点每年至少升级一次。

### 版本升级规范

边缘节点升级时，智能边缘平台自动选择最新版本的EdgeCore在边缘节点进行升级。

例如，当前EdgeCore有2.22.0、2.23.0、2.24.0三个版本，而您的边缘节点上EdgeCore版本为2.12.0，当您升级边缘节点时，IEF将推送最新版本2.24.0给您的边缘节点进行升级。

### 注意事项

- 为了让您的边缘节点应用更稳定可靠的运行，IEF不会主动升级您的边缘节点上的EdgeCore，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。
- 处于维护周期中的版本升级，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。
- 处于维护周期外的版本升级，可能会因为容器重启引起业务的短暂中断。
- 请勿在节点升级过程中变更节点配置，比如重启Docker、安装卸载GPU/NPU驱动、OS内核升级、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

**步骤1** 登录边缘节点，配置防火墙规则。

检查边缘节点防火墙状态。

```
systemctl status firewalld  
firewall-cmd --state
```

回显中，not running表示关闭，running表示开启。

如果防火墙开启，您需要打开8883端口，或关闭防火墙。

- 打开8883端口。  
firewall-cmd --add-port=8883/tcp --permanent  
systemctl restart firewalld
- 关闭防火墙。  
systemctl disable firewalld  
systemctl stop firewalld

**步骤2** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤3** 选择左侧导航栏的“边缘资源 > 边缘节点”。

**步骤4** 在“边缘侧软件版本”列查看是否可以升级。

仅处于“运行中”状态的边缘节点才可以升级。

- 如果显示可升级，表示可以升级。
- 如果没有显示，请查看边缘节点是否处于“运行中”状态。如果边缘节点处于“运行中”且无显示，则说明当前边缘节点EdgeCore是最新版本。

图 4-11 查看边缘节点是否可以升级



名称/ID	状态	主机名/网络	应用实例(正常/全部)	创建时间	边缘侧软件版本	节点标签	节点类型
ief-node 7092ad14-adee-4a09-b969-1505bbdecef5	运行中	eth0.192.168.0.230	0/0	2021/07/20 09:33:26 GMT+08:00	2.52.0 可升级	--	自建节点

步骤5 单击“更多 > 升级”。

图 4-12 升级边缘节点



名称/ID	状态	主机名/网络	应用实例(正常/全部)	创建时间	边缘侧软件版本	节点标签	节点类型	操作
ief-node 7092ad14-adee-4a09-b969-1505bbdecef5	运行中	eth0.192.168.0.230	0/0	2021/07/20 09:33:26 GMT+08:00	2.52.0 可升级	--	自建节点	删除 更多 应用 停用 查看监控 升级

步骤6 单击节点名称进入节点详情页面，可以查看详细升级记录。

图 4-13 升级记录



ief-node ● 运行中  
ID: 7092ad14-adee-4a09-b969-1505bbdecef5

状态描述 运行中

升级记录 安装成功 [查看](#)

节点描述 -- [编辑](#)

日志 [查看日志](#) [?](#)

----结束

## 4.1.7 日志、监控和告警

### 日志说明

边缘节点会上传系统日志和应用日志，您需要在IEF控制台上打开日志开关。

- **系统日志**：边缘节点上IEF软件（如edge-core、edge-logger和edge-monitor等）产生的日志。
- **应用日志**：边缘节点上部署的应用所产生的日志。
  - 边缘节点会上传“/var/IEF/app/log”目录的日志，您可以在创建应用时将容器中目录挂载到“/var/IEF/app/log/{appName}”下，具体挂载方法请参见

- **hostPath**: 将主机某个目录挂载到容器中。在AOM中可以按{appName}分类查看到应用的日志。
- 边缘节点会上传容器日志，日志组件会上传“{{DOCKER\_ROOT\_DIR}}/containers/{containerID}/{containerID}-json.log”文件的内容，DOCKER\_ROOT\_DIR可以通过**docker info**命令查询到，containerID就是容器ID。

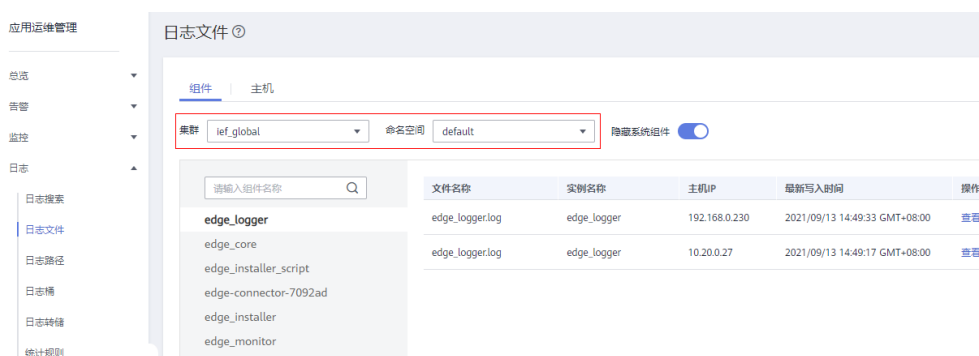
## 在 AOM 查看日志

**步骤1** 登录AOM管理控制台。

**步骤2** 在左侧导航栏选择“日志 > 日志文件”，单击“组件”页签。

**步骤3** 选择集群“ief\_global”和命名空间“default”。

图 4-14 选择集群和命名空间



**步骤4** 搜索应用名称，单击日志文件右侧的“查看”，即可查看详细日志。

----结束

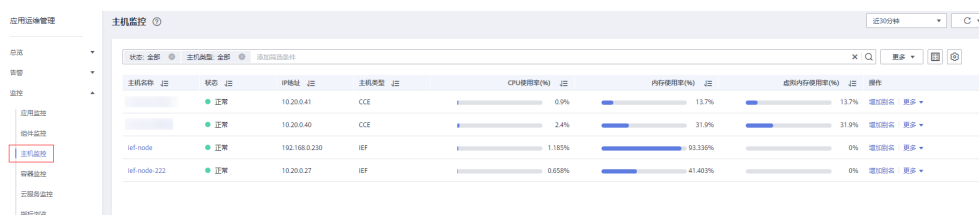
## 在 AOM 中查看节点监控信息

您可以在AOM查看节点监控信息。

**步骤1** 登录AOM管理控制台。

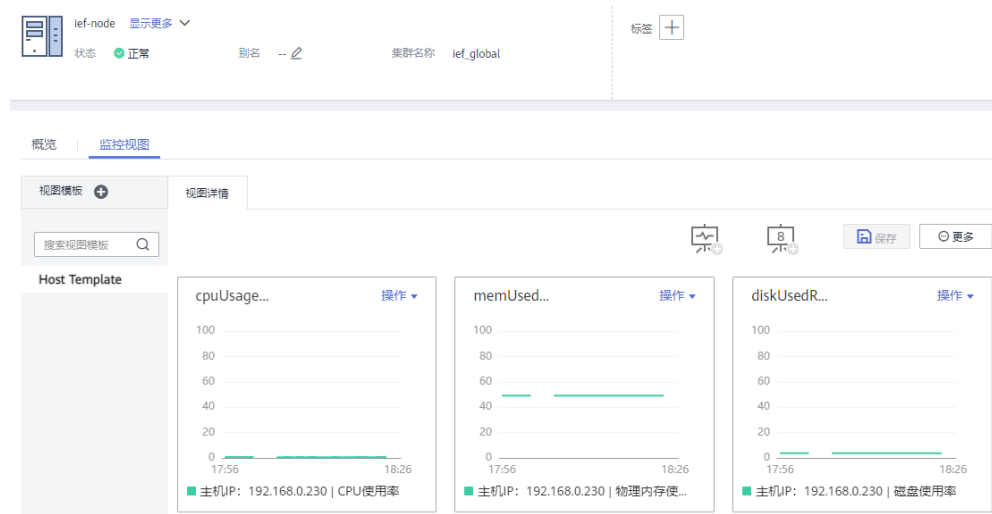
**步骤2** 选择监控的节点。

图 4-15 选择监控节点



**步骤3** 单击节点名称，在“监控视图”页签下，您可以查看节点的资源使用情况，如CPU、内存的使用率等。

图 4-16 查看监控信息



----结束

## 在 AOM 中查看容器监控信息

AOM中可以查看边缘节点上容器应用的监控信息。

**步骤1** 登录AOM管理控制台。

**步骤2** 选择要监控的容器工作负载。

图 4-17 选择工作负载

The screenshot shows the '容器监控' (Container Monitoring) page. It features a table with columns for '工作负载' (Workload), '名称' (Name), '状态' (Status), '所属集群' (Cluster), '命名空间' (Namespace), '物理内存容量(MB)' (Physical Memory Capacity), 'CPU使用率(%)' (CPU Usage), and '内存使用率(%)' (Memory Usage). The table lists several workloads, including 'a-cc', 'a-sc', 'a33', and 'b-cc', all with a status of '正常' (Normal). The table also includes search and filter icons.

**步骤3** 单击工作负载名称，进入详情页面，在“监控视图”页签下，您可以设置容器的监控指标，如CPU、内存的使用率等。



图 4-18 查看监控信息



----结束

## IEF 预置的告警

IEF为每个边缘节点预置了7个告警规则，这7类告警会自动上报到AOM。

告警名称	触发条件	清除条件	告警等级
容器引擎异常	边缘节点配置Docker使能时，查询Docker信息失败	Docker正常运行，EdgeCore能够获取到Docker信息	紧急
存活探针异常	应用配置存活探针，探针检测到异常	容器探针检测成功	重要
申请GPU资源失败	部署GPU应用，申请GPU资源失败	成功申请到GPU资源	紧急
获取GPU信息失败	边缘节点配置GPU使能时，查询GPU信息失败	成功查询到GPU信息	紧急
AK/SK无效	EdgeHub连续10次分发临时AK/SK，检测到过期或者状态异常	EdgeHub成功分发临时AK/SK	重要
应用重启	应用容器异常重启	无需清除	次要
容器绑定网卡异常	容器绑定的网卡发生异常	容器绑定的网卡状态正常	紧急

图 4-19 查看告警

名称	告警级别	告警源	资源类型	资源名称	告警详情	发生时间 (注)	持续时间 (注)	操作
应用健康	次要	IEF	Application	Application = sdf, Instance = sdf-587...	应用健康异常	2022-09-14 14:17:34 GMT+08:00	10秒	
应用健康	次要	IEF	Application	Application = nginx-2, Instance = nginx...	应用健康异常	2022-09-14 14:14:22 GMT+08:00	29分22秒	
应用健康	次要	IEF	Application	Application = nginx-2, Instance = nginx...	应用健康异常	2022-09-14 14:13:39 GMT+08:00	49分58秒	
应用健康	次要	IEF	Application	Application = nginx-886-a7a23, Insta...	应用健康异常	2022-09-14 14:13:15 GMT+08:00	49分20秒	
应用健康	次要	IEF	Application	Application = sdf, Instance = sdf-587...	应用健康异常	2022-09-14 14:11:24 GMT+08:00	69分20秒	
应用健康	次要	IEF	Application	Application = sdf, Instance = sdf-587...	应用健康异常	2022-09-14 14:10:44 GMT+08:00	79秒	
应用健康	次要	IEF	Application	Application = nginx-886-a7a23, Insta...	应用健康异常	2022-09-14 14:09:35 GMT+08:00	89分08秒	
应用健康	次要	IEF	Application	Application = nginx-2, Instance = nginx...	应用健康异常	2022-09-14 14:09:22 GMT+08:00	89分22秒	
应用健康	次要	IEF	Application	Application = nginx-886-a7a23, Insta...	应用健康异常	2022-09-14 14:07:15 GMT+08:00	100分20秒	
应用健康	次要	IEF	Application	Application = nginx-886-a7a23, Insta...	应用健康异常	2022-09-14 14:06:15 GMT+08:00	11分29秒	

## 在 AOM 中设置告警

您可以在AOM中创建告警规则来监控边缘节点上的各项指标，请参考[创建阈值规则](#)进行设置。

## 上报自定义告警到 AOM

IEF支持从边缘节点上报自定义告警到AOM，使用MQTT客户端发布告警信息到MQTT broker，IEF会将告警自动上报到AOM。

具体请参见[添加告警](#)和[清除告警](#)。

## 4.1.8 安装并配置 GPU 驱动

### 背景信息

对于使用GPU的边缘节点，在纳管边缘节点前，需要安装并配置GPU驱动。

IEF当前支持Nvidia Tesla系列P4、P40、T4等型号GPU，支持CUDA Toolkit 8.0至10.0版本对应的驱动。

### 操作步骤

#### 步骤1 安装GPU驱动。

1. 下载GPU驱动，推荐驱动链接：

[https://www.nvidia.com/content/DriverDownload-March2009/confirmation.php?url=/tesla/440.33.01/NVIDIA-Linux-x86\\_64-440.33.01.run&lang=us&type=Tesla](https://www.nvidia.com/content/DriverDownload-March2009/confirmation.php?url=/tesla/440.33.01/NVIDIA-Linux-x86_64-440.33.01.run&lang=us&type=Tesla)

2. 执行如下安装驱动命令。

```
bash NVIDIA-Linux-x86_64-440.33.01.run
```

3. 执行如下命令检查GPU驱动安装状态。

```
nvidia-smi
```

#### 步骤2 以root用户登录边缘节点。

#### 步骤3 执行如下命令。

```
nvidia-modprobe -c0 -u
```

**步骤4** 创建文件夹。

```
mkdir -p /var/IEF/nvidia/drivers /var/IEF/nvidia/bin /var/IEF/nvidia/lib64
```

**步骤5** 拷贝驱动文件。

- 对于CentOS，依次执行如下命令拷贝驱动文件：

```
cp /lib/modules/{当前环境内核版本号}/kernel/drivers/video/nvi* /var/IEF/nvidia/drivers/
```

```
cp /usr/bin/nvidia-* /var/IEF/nvidia/bin/
```

```
cp -rd /usr/lib64/libcuda* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/libEG* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/libGL* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/libnv* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/libOpen* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/libvdpau_nvidia* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib64/vdpau /var/IEF/nvidia/lib64/
```

- 对于Ubuntu，依次执行如下命令拷贝驱动文件：

```
cp /lib/modules/{当前环境内核版本号}/kernel/drivers/video/nvi* /var/IEF/nvidia/drivers/
```

```
cp /usr/bin/nvidia-* /var/IEF/nvidia/bin/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libcuda* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libEG* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libGL* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libnv* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libOpen* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/libvdpau_nvidia* /var/IEF/nvidia/lib64/
```

```
cp -rd /usr/lib/x86_64-linux-gnu/vdpau /var/IEF/nvidia/lib64/
```

其中，当前环境内核版本号可以使用`uname -r`命令查看获取，如下所示，请替换为实际取值。

```
# uname -r  
3.10.0-514.e17.x86_64
```

**步骤6** 执行以下命令修改目录权限。

```
chmod -R 755 /var/IEF
```

----结束

## 4.1.9 边缘核心软件 EdgeCore 配置管理

### 操作场景

IEF边缘软件支持对EdgeCore配置参数进行管理，通过该功能您可以对边缘核心软件EdgeCore进行深度配置。

## 操作步骤

**步骤1** 在边缘节点上执行如下命令修改EdgeCore配置，并保存。

```
vi /opt/IEF/Edge-core/conf/edge.yaml
```

支持配置的参数如下表所示：

表 4-3 参数说明

组件	参数	说明	取值
edge-core	interface-name	网卡名称	默认：eth0
	internal-server	内置mqtt broker监听地址	tls://lo:8883,tls://docker0:8883
	mage-gc-high-threshold	触发镜像垃圾回收的磁盘使用率百分比	默认：80
	image-gc-low-threshold	镜像垃圾回收试图释放资源后达到的磁盘使用率百分比	默认：40
	swr-url	拉取镜像的代理地址	默认：""

**步骤2** 更改完配置之后，重启EdgeCore。

```
systemctl restart edgecore
```

----结束

### 4.1.10 删除边缘节点

#### 前提条件

删除边缘节点前，需要先解绑终端设备、删除边缘节点上的应用和证书。

#### 操作步骤

**步骤1** 以可运行sudo命令的用户登录边缘节点。

**步骤2** 执行以下命令卸载已纳管节点上的软件和配置文件。

```
cd /opt/edge-installer; sudo ./installer -op=uninstall
```

**步骤3** 登录IEF管理控制台。

**步骤4** 选择左侧导航栏的“边缘资源 > 边缘节点”。

**步骤5** 单击待删除边缘节点右侧的“更多 > 删除”。

**步骤6** 根据提示完成删除操作。

----结束

## 4.2 终端设备管理

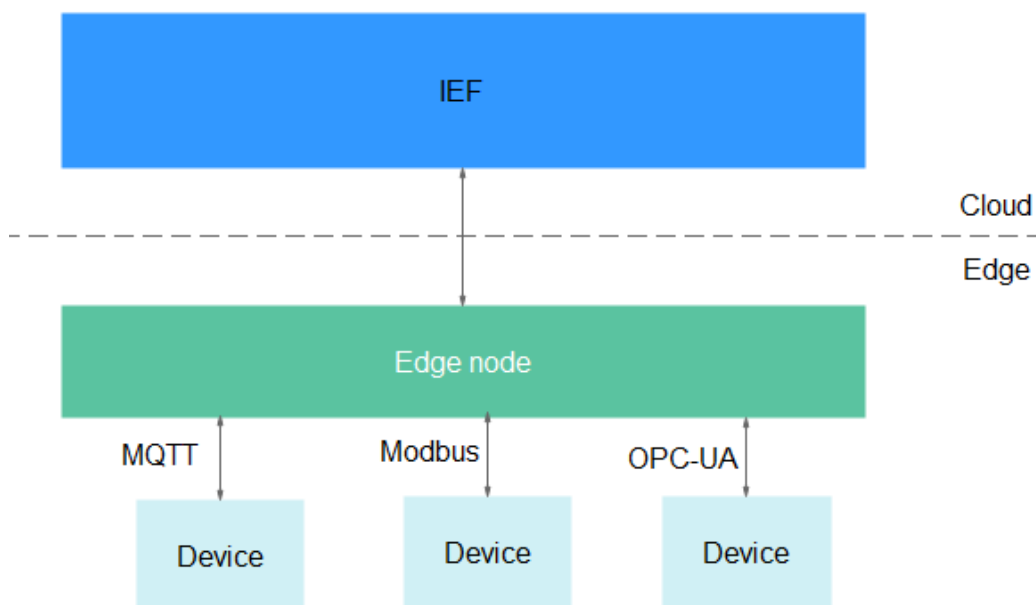
### 4.2.1 终端设备与设备孪生

#### 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。

终端设备可以连接到边缘节点，终端设备支持通过MQTT协议接入。终端设备接入后，可以在IEF中对终端设备进行统一管理。

图 4-20 终端设备管理



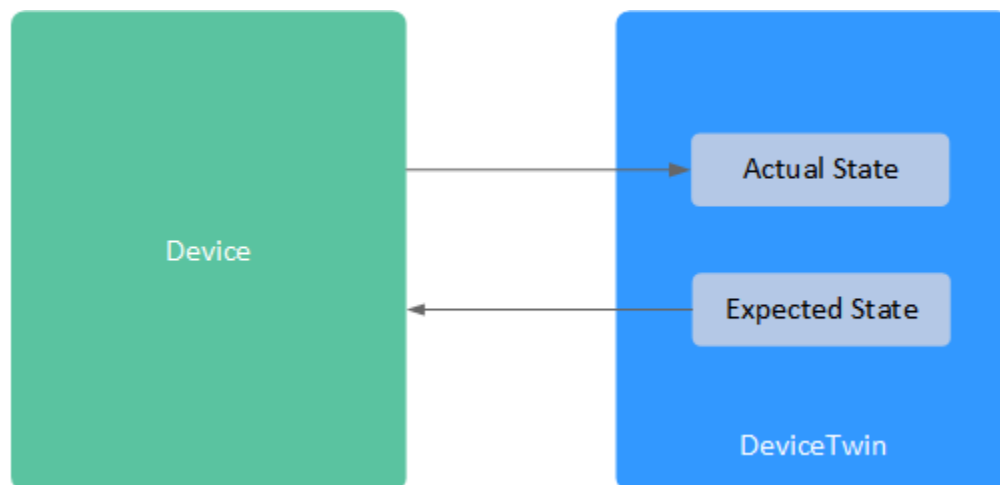
#### 设备孪生（DeviceTwin）

终端设备通常包含两类数据：

- 一是不会改变的元数据，包括序列号、资产标识符、Mac地址等描述设备信息的数据。这种数据也可以称为终端设备的静态属性或**设备属性**。
- 另一类是终端设备的动态数据，包括特定背景下的终端设备专有实时数据，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Expected State（期望的状态）进行状态更新，此外终端设备实时反馈自身的Actual State（真实的状态），设备孪生同时记录终端设备的Actual State和Expected State。这种方式也使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。

图 4-21 DeviceTwin



在IEF中可以创建终端设备，并能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属性和孪生信息。边缘节点上的应用程序可在边缘节点获取终端设备属性、设备孪生信息、以及修改终端设备孪生期望值和真实值。同时IEF负责同步云、边的孪生信息，当有冲突时，将以边缘侧的修改为主。

详细的终端设备状态边云协同机制请参见[设备孪生工作原理](#)。

## 使用流程

使用IEF管理和控制终端设备，通常使用的步骤如下：

1. 定义终端设备模板（包含设备属性、孪生属性）。
2. 使用模板创建设备。  
您也可以不使用模板，直接创建设备。
3. 将终端设备关联到边缘节点。
4. 在IEF中管理和控制终端设备，监测终端设备状态。

### 4.2.2 设备模板

边缘计算场景下，通常有数量庞大的终端设备，在IEF中可以将一类终端设备定义成统一的模板，这样可以通过模板创建终端设备。比如您可以创建一个自定义的终端设备模板“Camera”，创建终端设备时可以使用该模板的所有属性，而不必为每一个类似的终端设备设置同样的属性。

IEF支持根据访问协议定义模板。

#### 创建设备模板

- 步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。
- 步骤2** 选择左侧导航栏“边缘资源 > 终端设备”，单击页面右上角的“创建设备模板”。
- 步骤3** 填写模板名称，选择访问协议，设置设备属性、孪生属性、设备标签和描述。

图 4-22 创建设备模板

服务实例 wx-reliable

\* 名称

访问协议  MQTT  modbus

描述

0/255

---

模板属性

**注意：**设备属性会明文展示所输入信息，请不要填入敏感信息，如涉及敏感信息，请先加密，请防止信息泄露。

属性名	类型	属性值	是否可选	操作
+ 新增属性				

孪生属性

**注意：**设备孪生属性会明文展示所输入信息，请不要填入敏感信息，如涉及敏感信息，请先加密，请防止信息泄露。

属性名	类型	属性值	操作
+ 新增孪生属性			

- **名称：**终端设备模板的名称。
- **访问协议：**IEF支持MQTT协议。
- **模板属性：**  
属性是键值对形式，请输入属性名和属性值，并选择类型。  
通常将不会改变的元数据，例如序列号、资产标识符、Mac地址之类的信息设置为模板属性。
- **孪生属性：**  
通常将终端设备的动态数据，例如特定背景下的设备专有实时数据，例如灯的开、关状态等设置为孪生属性。
  - MQTT协议：MQTT协议的孪生属性是键值对形式，请输入属性名和属性值，并选择类型。

#### 须知

IEF不提供任何加解密工具，对您配置的设备属性值不感知，如果设备属性值设置为加解密文，需要您自行解密。

- **标签：**标签用于为终端设备分类，标签可以帮助您更加快速的搜索到想要的终端设备。您可以为终端设备设置标签，方便分类管理。

**步骤4** 单击“创建”，即创建设备模板成功，返回到设备模板页面。

----结束

## 4.2.3 终端设备

终端设备可以连接到边缘节点，终端设备支持通过MQTT协议接入。终端设备接入后，可以在IEF中对终端设备进行统一管理。

## 注册终端设备

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边缘资源 > 终端设备”，单击页面右上角的“注册终端设备”。

**步骤3** 填写设备参数。

- **名称**：终端设备的名称。
- **ID**：您可以选择自定义设备ID，或者让IEF自动为您生成ID。如果需要自定义设备ID，请勾选“自定义设备ID”并填写设备ID。
- **访问协议**：MQTT。
- **设备配置**：选择已经创建的设备模板，为设备自动添加属性。模板中定义了设备属性、孪生属性、设备标签信息。您也可以不使用模板，直接手动为设备添加属性和标签，其含义与模板中定义一致，具体请参见[设备模板](#)。

注意：设备模板的访问协议必须和注册终端设备时选择的协议一致，才能在此处被选择到。

**步骤4** 单击“注册”，即注册终端设备成功，返回到终端设备列表页面。

---结束

## 后续操作

您可以将终端设备添加到边缘节点中，具体请参见[终端设备绑定到边缘节点](#)。

### 4.2.4 终端设备绑定到边缘节点

一个边缘节点可以绑定多个终端设备，但一个终端设备只可以被绑定于一个边缘节点。通过绑定终端设备到固定的边缘节点，您可以在边缘节点部署相应的应用，实现管理终端设备和监控终端设备状态等功能。

## 绑定边缘节点

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“边缘资源 > 终端设备”。

**步骤3** 在终端设备所在行右侧单击“绑定节点”。

**步骤4** 填写终端设备与节点的关系，选择要绑定的边缘节点，单击“确定”。

---结束

## 从边缘节点绑定终端设备

您还可以在边缘节点上操作绑定终端设备。

**步骤1** 登录IEF管理控制台。

**步骤2** 选择左侧导航栏的“边缘资源 > 边缘节点”，单击对应节点进入边缘节点详情页。

**步骤3** 选择“设备”页签，单击“绑定设备”。

**步骤4** 在弹出的窗口中勾选需要绑定的设备，并填写终端设备与节点的关系，单击“确定”。



**说明**

节点和设备具有依赖关系，只有安装了Modbus插件的节点才能绑定Modbus协议的终端设备。

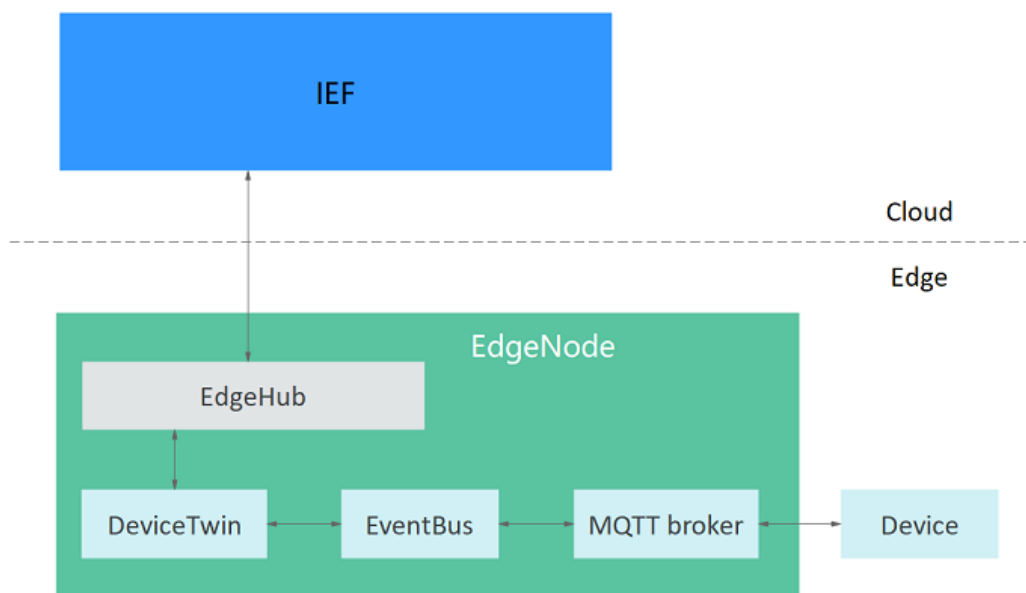
----结束

## 4.2.5 设备孪生工作原理

边缘节点纳管后，会在边缘节点上安装Edge Agent，其中终端设备管理相关组件如下所示。

- EdgeHub：WebSocket客户端，包括同步云端资源更新、报告边缘节点和终端设备信息到云端等功能。
- DeviceTwin：设备孪生，负责存储终端设备状态并将设备状态同步到云端。
- EventBus：与MQTT服务器交互的客户端，为其他组件提供订阅和发布消息的功能。
- MQTT broker：MQTT服务器。

图 4-23 终端设备管理



终端设备、边缘节点、IEF通信的过程中，设备孪生（DeviceTwin）起到了一个非常重要的作用，设备孪生保持设备的动态数据，包括特定背景下的设备专有实时数据，例如灯的开、关状态。

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Expected State（期望的状态）进行状态更新，此外终端设备实时反馈自身的Actual State（真实的状态），设备孪生同时记录设备的Actual State和Expected State。这种方式也使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。

设备孪生的参考示例如下。

```
{
  "device": {
    "id": "989e4fc8-9f24-44d7-9f9c-a0bd3bfb1949",
    "description": "my home light",
```

```
{
  "name": "light",
  "state": "online",
  "twin": {
    "powerstatus": {
      "expected": {
        "value": "ON"
      },
      "actual": {
        "value": "OFF"
      },
      "metadata": {
        "type": "string"
      }
    }
  }
}
```

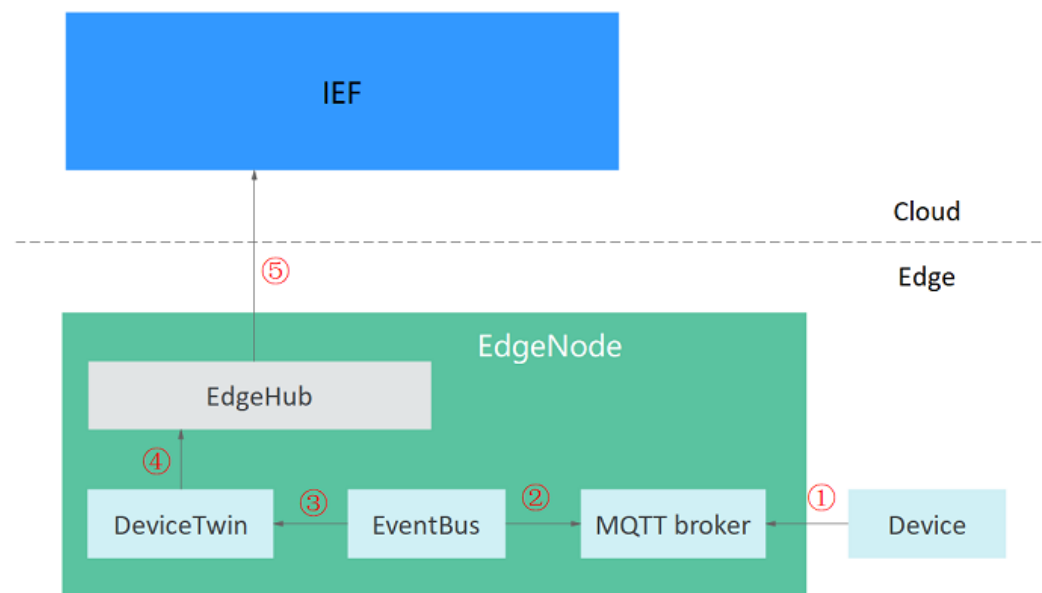
这个名为“light”的设备定义一个名为“powerstatus”的孪生属性，期望值（expected）是ON，而实际值（actual）是OFF。

下面看下这个终端设备与边缘节点和IEF通信的过程。

## 终端设备上报实际状态到云端

终端设备上报实际状态到云端的过程如图4-24所示。

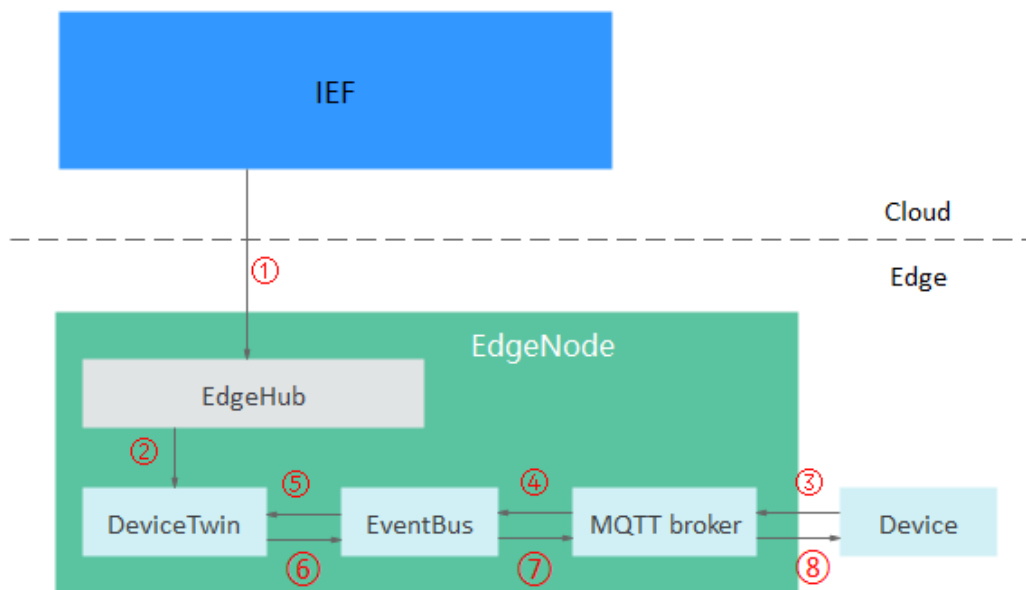
图 4-24 设备上报状态



1. 终端设备将实际状态（Actual State）实时上报给MQTT broker。
2. EventBus从MQTT broker收到订阅消息，消息内容包含终端设备的实际状态。
3. EventBus把终端设备实际状态发送给DeviceTwin，DeviceTwin在边缘节点存储终端设备实际状态。
4. DeviceTwin同步实际状态给WebSocket客户端EdgeHub。
5. EdgeHub发送消息给IEF。

## 云端修改孪生属性控制终端设备状态

图 4-25 修改终端设备状态



1. 在IEF中修改终端设备的孪生属性，IEF将终端设备期望状态（Expected State）发送给边缘节点的EdgeHub。
2. EdgeHub发送终端设备期望状态消息到DeviceTwin，DeviceTwin在边缘节点存储终端设备期望状态。
3. 终端设备实时发消息给MQTT broker查询终端设备期望状态。
4. EventBus接收到从MQTT broker发过来的消息。
5. EventBus根据消息去查询终端设备期望状态。
6. DeviceTwin反馈当前终端设备期望状态给EventBus。
7. EventBus发送设终端备期望状态的结果给MQTT broker。
8. 终端设备从MQTT broker收到订阅消息，根据期望状态调整实际状态。

### 4.2.6 设备数据上云

#### MQTT broker

终端设备可以通过MQTT协议与IEF云端进行通信，您也可以通过发送/订阅消息控制终端设备。

边缘节点上有一个**内置MQTT broker**，内置MQTT broker使用8883端口与终端设备通信，与内置MQTT broker通信需要经过安全认证，具体请参见[使用证书进行安全认证](#)。

另外，边缘节点还支持与**外置MQTT broker**通信，即在边缘节点上安装一个MQTT broker（如开源的Mosquitto，默认使用1883端口通信）。

#### 说明

如使用外置MQTT broker，请注意需要保证外置MQTT broker通信的端口能正常使用。

## MQTT Topic

终端设备与边缘节点、IEF的通信都是通过给MQTT broker中转消息实现的，在MQTT broker中，默认提供如表4-4所示的Topic（消息主题），上报状态、控制终端设备状态都是通过发送/订阅消息实现的。

应用程序编写完后，可以通过应用部署功能，将应用从IEF中部署到边缘节点，详情请参见[容器应用管理](#)。

表 4-4 IEF 提供的默认 Topic

名称	使用类型	Topic	说明
设备孪生变更	订阅	\$hw/events/device/{device_id}/twin/update/document	设备孪生更新文档，当孪生变化时，反映孪生变化前、变化后的区别。
设备孪生delta	订阅	\$hw/events/device/{device_id}/twin/update/delta	设备孪生delta事件，当孪生变化时，反映期望值与真实值不一致的孪生信息。
设备成员变更	订阅	\$hw/events/node/{node_id}/membership/updated	绑定终端设备关系变化。
设备属性变更	订阅	\$hw/events/device/{device_id}/updated	终端设备属性更新。
设备成员获取	发布	\$hw/events/node/{node_id}/membership/get	绑定终端设备关系获取。
设备成员获取结果	订阅	\$hw/events/node/{node_id}/membership/get/result	绑定终端设备关系获取结果。
设备孪生获取	发布	\$hw/events/device/{device_id}/twin/get	设备孪生获取。
设备孪生获取结果	订阅	\$hw/events/device/{device_id}/twin/get/result	设备孪生获取结果。
设备孪生更新	发布	\$hw/events/device/{device_id}/twin/update	设备孪生更新。
设备孪生更新结果	订阅	\$hw/events/device/{device_id}/twin/update/result	设备孪生更新结果。
请求加密数据	发布	\$hw/{project_id}/encryptdatas/{encryptdata_name}/properties/{properties_name}/decrypt	发布获取加密数据请求。

名称	使用类型	Topic	说明
获取加密数据	订阅	\$hw/{project_id}/ encryptdatas/ {encryptdata_name}/ properties/ {properties_name}/ plaintext	订阅获取加密数据。
添加告警	发布	\$hw/alarm/{appname}/add	向AOM发送告警。
清除告警	发布	\$hw/alarm/{appname}/ clear	清除AOM中告警。
自定义Topic	发布	{project_id}/nodes/ {node_id}/user/ {custom_topic}	自定义Topic，Topic根据您的需要自行定义。 您可以将终端设备数据发送到边缘节点MQTT broker的自定义Topic中，IEF会将这些数据转发到DIS通道或APIG后端地址。数据转发到DIS通道或者APIG后端地址后，您可以提取这些数据，并对数据进行处理分析。

接下来将介绍如何在边缘侧获取终端设备信息，接收云上的控制消息，以及如何将终端设备数据上报到云端。MQTT收发消息的示例代码请参见[Go语言代码样例](#)和[Java语言代码样例](#)。

## 获取节点关联的终端设备成员

**步骤1** 向[设备成员获取](#)发送获取终端设备成员消息的请求。

**Topic:** \$hw/events/node/{node\_id}/membership/get

**Payload:** {"event\_id":"自定义ID"}

示例如下：

```
$hw/events/node/{node_id}/membership/get
{"event_id":""}
```

**步骤2** 向[设备成员获取结果](#)订阅终端设备成员的返回结果。

**Topic:** \$hw/events/node/{node\_id}/membership/get/result

返回结果示例如下：

```
{
  "event_id": "",
  "timestamp": 1554986455386,
  "devices": [
    {
      "id": "2144773f-13f1-43f5-af07-51991d4fd064",
      "name": "equipmentA",
      "state": "unknown",
    }
  ]
}
```

```
    "attributes": {
      "name": {
        "value": "a",
        "optional": true,
        "metadata": {
          "type": "string"
        }
      }
    }
  }
}
```

----结束

## 获取设备孪生

**步骤1** 向[设备孪生获取](#)发送请求获取设备孪生。

**Topic:** \$hw/events/device/{device\_id}/twin/get

**Payload:** {"event\_id":"自定义id"}

**步骤2** 向[设备孪生获取结果](#)订阅设备孪生的返回结果。

**Topic:** \$hw/events/device/{device\_id}/twin/get/result

获取的结果如下:

```
{
  "event_id": "",
  "timestamp": 1554988425592,
  "twin": {
    "humidity": {
      "expected": {
        "value": "0",
        "metadata": {
          "timestamp": 1554988419529
        }
      },
      "optional": true,
      "metadata": {
        "type": "int"
      }
    },
    "temperature": {
      "expected": {
        "value": "0",
        "metadata": {
          "timestamp": 1554988419529
        }
      },
      "optional": true,
      "metadata": {
        "type": "int"
      }
    }
  }
}
```

----结束

## 监听设备孪生事件

通过[获取节点关联的终端设备成员](#)和[获取设备孪生](#)以后，即可获取到节点绑定的终端设备ID，随后即可监听该终端设备的事件。

在云端更新设备孪生属性，设置期望值从而达到控制边侧终端设备的目的。

例如某个设备有两个孪生属性，humidity和temperature。

图 4-26 孪生属性

概览   <u>设备孪生</u>   标签			
+ 添加孪生属性			
属性名	类型	期望值	设置时间
humidity	int	9	2020/07/13 09:40:38 GMT+08:00
temperature	int	0	2020/07/13 09:40:58 GMT+08:00

在“设备孪生”页签中编辑孪生属性，将humidity由9改为10。属性修改完成以后，在端侧可收到两个事件：“设备孪生变更事件”和“设备孪生delta事件”。

- **设备孪生变更事件**：包含变更前和变更后的设备孪生信息详情。
- **设备孪生delta事件**：包含设备孪生的详情信息以及设备孪生属性期望值与实际值不一致的delta部分。

订阅这两个Topic，就可以收到变更设备孪生的消息。

**步骤1** 订阅**设备孪生变更**。

**Topic**: \$hw/events/device/{device\_id}/twin/update/document

在边侧收到变更消息如下：

```
{
  "event_id": "0f921313-4074-46a2-96f6-aac610721059",
  "timestamp": 1555313685831,
  "twin": {
    "humidity": {
      "last": {
        "expected": {
          "value": "9",
          "metadata": {
            "timestamp": 1555313665978
          }
        }
      },
      "optional": true,
      "metadata": {
        "type": "int"
      }
    },
    "current": {
      "expected": {
        "value": "10",
        "metadata": {
          "timestamp": 1555313685831
        }
      },
      "optional": true,
      "metadata": {
        "type": "int"
      }
    }
  },
  "temperature": {
    "last": {
```

```
    "expected": {
      "value": "0",
      "metadata": {
        "timestamp": 1555313665978
      }
    },
    "actual": {
      "value": "2",
      "metadata": {
        "timestamp": 1555299457284
      }
    },
    "optional": true,
    "metadata": {
      "type": "int"
    }
  },
  "current": {
    "expected": {
      "value": "0",
      "metadata": {
        "timestamp": 1555313685831
      }
    },
    "actual": {
      "value": "2",
      "metadata": {
        "timestamp": 1555299457284
      }
    },
    "optional": true,
    "metadata": {
      "type": "int"
    }
  }
}
```

## 步骤2 订阅设备孪生delta。

**Topic:** \$hw/events/device/{device\_id}/twin/update/delta

在边侧收到变更消息如下:

```
{
  "event_id": "60fb5baf-d4ad-47b0-a21e-8b57b52d0978",
  "timestamp": 1555313685837,
  "twin": {
    "humidity": {
      "expected": {
        "value": "10",
        "metadata": {
          "timestamp": 1555313685831
        }
      }
    },
    "optional": true,
    "metadata": {
      "type": "int"
    }
  },
  "temperature": {
    "expected": {
      "value": "0",
      "metadata": {
        "timestamp": 1555313685831
      }
    },
    "actual": {
      "value": "2",
      "metadata": {
```



```
        "timestamp": 1555299457284
      }
    },
    "optional": true,
    "metadata": {
      "type": "int"
    }
  }
},
"delta": {
  "humidity": "10",
  "temperature": "0"
}
}
```

----结束

## 上报设备属性实际值

**步骤1** 发布终端设备孪生更新事件。

**Topic:** \$hw/events/device/{device\_id}/twin/update

**Payload:** {"event\_id":"","timestamp":0,"twin":{"属性":{"actual":{"value":"设备实际值"}}}}

示例如下:

```
{
  "event_id": "",
  "timestamp": 0,
  "twin": {
    "temperature": {
      "actual": {
        "value": "2"
      }
    }
  }
}
```

发布后，在云端可以观察到设备孪生的实际值发生了相应的变化，如[图4-27](#)所示。

**图 4-27** 孪生属性值发生变化



The screenshot shows a web interface with three tabs: 'Overview', 'Device Twin', and 'Tags'. The 'Device Twin' tab is active. Below the tabs is a button labeled 'Add Device Twin Attribute'. Below the button is a table with the following data:

属性名	类型	期望值	设置时间	实际值
humidity	int	9	2020/07/13 09:40:38 GMT+08:00	
temperature	int	0	2020/07/13 09:46:53 GMT+08:00	2

**步骤2** 在边缘侧订阅[设备孪生更新结果](#)，能收到设备孪生更新事件的结果。

**Topic:** \$hw/events/device/{device\_id}/twin/update/result

更新的结果如下所示。

```
{
  "event_id": "",
  "timestamp": 1554992093859,
  "twin": {
    "temperature": {
      "actual": {
```

```
        "value": "2",
        "metadata": {
            "timestamp": 1554992093859
        }
    },
    "optional": true,
    "metadata": {
        "type": "int"
    }
}
}
```

----结束

## 4.2.7 使用证书进行安全认证

### 操作场景

内置MQTT broker默认开启端口进行TLS（Transport Layer Security）安全认证，客户端必须带上证书才能访问MQTT broker。

终端设备和应用可以通过在对应节点详情页创建的证书进行安全认证。

#### 说明

在节点组中部署的应用可能调度到节点组内任意节点，终端设备可能访问节点组内任意节点，从边缘节点处创建的证书不能满足应用访问节点MQTT broker的需要，所以需要使用节点组证书。节点组证书创建方法请参见[节点组证书](#)。

### 约束与限制

- 证书与边缘节点绑定，在一个边缘节点下申请的证书只能用来访问该边缘节点的MQTT broker，如果访问其他边缘节点的MQTT broker，会导致认证失败。
- 一个边缘节点最多只能申请10份证书。
- 证书的有效期为5年。
- MQTT使用限制

表 4-5 MQTT 使用限制

描述	限制
支持的MQTT协议版本	3.1.1
与标准MQTT协议的区别	<ul style="list-style-type: none"><li>• 支持QoS 0</li><li>• 支持Topic自定义</li><li>• 不支持QoS 1和QoS 2</li><li>• 不支持will、retain msg</li></ul>
MQTT支持的安全等级	采用TCP通道基础 + TLS协议（TLSV1.2 版本）

## 申请证书

### 📖 说明

证书有效期为5年。

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“边缘资源 > 边缘节点”。

**步骤3** 单击边缘节点名称，进入边缘节点详情。

**步骤4** 选择“证书”页签，单击“添加证书”。

**步骤5** 输入证书名称，单击“确定”。

证书添加成功后会自动下载，请妥善保管证书。

图 4-28 添加证书

添加证书

证书添加成功后，系统将自动下载证书文件

\* 名称 请输入名称

描述 请输入描述 0/255

确定 取消

----结束

## 使用证书

证书用于终端设备与MQTT broker通信时鉴权。

下面是[Go语言代码样例](#)和[Java语言代码样例](#)，演示了如何使用证书做鉴权。

### 📖 说明

1. 客户端不需要校验服务端证书，单向认证即可。
2. 内置MQTT broker默认开启8883端口。
3. 样例中的Go语言MQTT Client引用了github.com/eclipse/paho.mqtt.golang开源库。
4. 客户端需要处理断连事件，实现掉线重连机制，提高连接可靠性。

## Go 语言代码样例

```
package main

import (
    "crypto/tls"
    "crypto/x509"
    "fmt"
    "math/rand"
    "sync"
    "time"

    MQTT "github.com/eclipse/paho.mqtt.golang"
)

func main() {
    subClient := InitMqttClient(onSubConnectionLost)
    pubClient := InitMqttClient(onPubConnectionLost)

    wait := sync.WaitGroup{}
    wait.Add(1)

    go func() {
        for {
            time.Sleep(1*time.Second)
            pubClient.Publish("topic", 0, false, "hello world")
        }
    }()

    subClient.Subscribe("topic", 0, onReceived)

    wait.Wait()
}

func InitMqttClient(onConnectionLost MQTT.ConnectionLostHandler) MQTT.Client {
    pool := x509.NewCertPool()
    cert, err := tls.LoadX509KeyPair("/tmp/example_cert.crt", "/tmp/example_cert.key")
    if err != nil {
        panic(err)
    }

    tlsConfig := &tls.Config{
        RootCAs: pool,
        Certificates: []tls.Certificate{cert},
        // 单向认证, client不校验服务端证书
        InsecureSkipVerify: true,
    }
    // 使用tls或者ssl协议, 连接8883端口
    opts := MQTT.NewClientOptions().AddBroker("tls://127.0.0.1:8883").SetClientID(fmt.Sprintf("%f", rand.Float64()))
    opts.SetTLSConfig(tlsConfig)
    opts.OnConnect = onConnect
    opts.AutoReconnect = false
    // 回调函数, 客户端与服务端断连后立刻被触发
    opts.OnConnectionLost = onConnectionLost
    client := MQTT.NewClient(opts)
    loopConnect(client)
    return client
}

func onReceived(client MQTT.Client, message MQTT.Message) {
    fmt.Printf("Receive topic: %s, payload: %s \n", message.Topic(), string(message.Payload()))
}

// sub客户端与服务端断连后, 触发重连机制
func onSubConnectionLost(client MQTT.Client, err error) {
    fmt.Println("on sub connect lost, try to reconnect")
    loopConnect(client)
    client.Subscribe("topic", 0, onReceived)
}
```

```
// pub客户端与服务端断连后，触发重连机制
func onPubConnectionLost(client MQTT.Client, err error) {
    fmt.Println("on pub connect lost, try to reconnect")
    loopConnect(client)
}

func onConnect(client MQTT.Client) {
    fmt.Println("on connect")
}

func loopConnect(client MQTT.Client) {
    for {
        token := client.Connect()
        if rs, err := CheckClientToken(token); !rs {
            fmt.Printf("connect error: %s\n", err.Error())
        } else {
            break
        }
        time.Sleep(1 * time.Second)
    }
}

func CheckClientToken(token MQTT.Token) (bool, error) {
    if token.Wait() && token.Error() != nil {
        return false, token.Error()
    }
    return true, nil
}
```

## Java 语言代码样例

MqttClientDemo.java文件:

```
/*
Description: MQTT消息收发JAVA demo。需要先创建边缘节点并下载获取客户端证书
*/

package com.example.demo;

import javax.net.ssl.SSLSocketFactory;

import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

/*
* MQTT Demo 展示客户端连接边缘节点broker进行消息的收发，连接进行SSL安全认证，demo演示包括如下：
* 1、MQTT接收客户端，接收MQTT消息
* 2、MQTT发送客户端，发送MQTT消息
*/

public class MqttClientDemo {
    private static int QOS_TYPE = 2;
    //MQTT服务器地址
    private static final String MQTT_HOST = "ssl://x.x.x.x:8883";
    //MQTT发送客户端id
    private static final String MQTT_PUB_CLIENT_ID = "pub_client_1";
    //MQTT接收客户端id
    private static final String MQTT_SUB_CLIENT_ID = "sub_client_1";
    //MQTT通道订阅主题topic
    private static final String TOPIC = "/hello";
    //MQTT客户端连接SSL证书配置路径
    public static final String CLIENT_CERT_FILE_PATH = "example_cert.crt";
    public static final String CLIENT_KEY_FILE_PATH = "example_cert.key";
    //MQTT客户端连接超时时间（秒）
}
```

```
public static final int TIME_OUT_INTERVAL = 10;
//MQTT客户端发送心跳间隔（秒）
public static final int HEART_TIME_INTERVAL = 20;
//MQTT客户端断线重试间隔（毫秒）
public static final int RECONNECT_INTERVAL = 10000;
//MQTT客户端发送消息间隔（毫秒）
public static final int PUBLISH_MSG_INTERVAL = 3000;

//MQTT client客户端
private MqttClient mqttClient;
//MQTT client连接MQTT的客户端ID，一般以客户端唯一标识符表示
private String clientId;
//MQTT client连接配置项
private MqttConnectOptions connOpts;
//初始化MQTT客户端未订阅任何topic
private boolean isSubscribe = false;

public MqttClientDemo(String id) throws MqttException {
    setClientId(id);
    initMqttClient();
    initCallback();
    initConnectOptions();
    connectMqtt();
}

/*****
 * 发送消息
 * @param message 待发送的消息
 * @throws MqttException
 *****/
public void publishMessage(String message) throws MqttException {
    MqttMessage mqttMessage = new MqttMessage(message.getBytes());
    mqttMessage.setQos(QOS_TYPE);
    mqttMessage.setRetained(false);
    mqttClient.publish(TOPIC, mqttMessage);
    System.out.println(String.format("MQTT Client[%s] publish message[%s]", clientId, message));
}

/*****
 * 订阅topic
 * @throws MqttException
 *****/
public void subscribeTopic() throws MqttException {
    int[] Qos = {QOS_TYPE};
    String[] topics = {TOPIC};
    mqttClient.subscribe(topics, Qos);
    isSubscribe = true;
}

/*****
 * 启动线程定时发送MQTT消息
 * @throws MqttException
 *****/
public void startPublishMessage() {
    new Thread() {
        @Override
        public void run() {
            while (true) {
                try {
                    Thread.sleep(PUBLISH_MSG_INTERVAL);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                try {
                    publishMessage("hello world!");
                } catch (MqttException e) {
                    System.out.println(String.format("MQTT client[%s] publish message error,errorMsg[%s]",
                    clientId, e.getMessage()));
                }
            }
        }
    }
}
```

```
    }  
    }  
    }.start();  
}  
  
/*****  
 * 初始化MQTT客户端  
 * @throws MqttException 连接异常  
 *****/  
private void initMqttClient() throws MqttException {  
    MemoryPersistence persistence = new MemoryPersistence();  
    mqttClient = new MqttClient(MQTT_HOST, clientId, persistence);  
}  
  
/*****  
 * 初始化连接配置  
 * @throws MqttException 连接异常  
 *****/  
private void initConnectOptions() {  
    connOpts = new MqttConnectOptions();  
    // 设置是否清空session，这里如果设置为false表示服务器会保留客户端的连接记录，这里设置为true表示  
    每次连接到服务器都以新的身份连接  
    connOpts.setCleanSession(true);  
    connOpts.setHttpsHostnameVerificationEnabled(false);  
    // 设置超时时间，单位为秒  
    connOpts.setConnectionTimeout(TIME_OUT_INTERVAL);  
    // 设置会话心跳时间，单位为秒，服务器会每隔1.5*20秒的时间向客户端发送个消息判断客户端是否在线，  
    但这个方法并没有重连的机制  
    connOpts.setKeepAliveInterval(HEART_TIME_INTERVAL);  
    SSLSocketFactory factory = null;  
    try {  
        factory = SslUtil.getSocketFactory(CLIENT_CERT_FILE_PATH, CLIENT_KEY_FILE_PATH);  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    // TLS连接配置  
    connOpts.setSocketFactory(factory);  
}  
  
/*****  
 * 发起连接MQTT connect请求  
 * @throws MqttException 连接异常  
 *****/  
private void connectMqtt() throws MqttException {  
    mqttClient.connect(connOpts);  
    System.out.println(String.format("MQTT client[%s] is connected,the connctOptions: \n%s", clientId,  
connOpts.toString()));  
}  
  
/*****  
 * 设置回调接口  
 * @throws MqttException 连接异常  
 *****/  
private void initCallback() {  
    mqttClient.setCallback(new MqttMessageCallback());  
}  
  
private void setClientId(String id) {  
    clientId = id;  
}  
  
/*****  
 * MQTT Client重连函数，调用连接函数并判断是否订阅过Topic，如果订阅过topic则重新订阅topic  
 * @throws MqttException  
 *****/  
private void rconnectMqtt() throws MqttException {  
    connectMqtt();  
    if (isSubscribe) {  
        subscribeTopic();  
    }  
}
```

```
    }
}

/*****
 * MQTT client 订阅topic后，MQTT 通道有数据，则通过该回调接口接收消息
 * @version V1.0
 *****/
private class MqttMessageCallback implements MqttCallback {

    @Override
    public void connectionLost(Throwable cause) {
        System.out.println(String.format("MQTT Client[%s] connect lost,Retry in 10 seconds,info[%s]",
clientId, cause.getMessage()));
        while (!mqttClient.isConnected()) {
            try {
                Thread.sleep(RECONNECT_INTERVAL);
                System.out.println(String.format("MQTT Client[%s] reconnect ....", clientId));
                reconnectMqtt();
            } catch (Exception e) {
                continue;
            }
        }
    }

    @Override
    public void messageArrived(String topic, MqttMessage mqttMessage) {
        String message = new String(mqttMessage.getPayload());
        System.out.println(String.format("MQTT Client[%s] receive message[%s] from topic[%s]", clientId,
message, topic));
    }

    @Override
    public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {
    }
}

public static void main(String[] args) throws MqttException {
    try {
        //订阅MQTT通道
        MqttClientDemo mqttsubClientDemo = new
MqttClientDemo(MqttClientDemo.MQTT_SUB_CLIENT_ID);
        mqttsubClientDemo.subscribeTopic();
        //往MQTT通道发送：hello world
        MqttClientDemo mqttpubClientDemo = new
MqttClientDemo(MqttClientDemo.MQTT_PUB_CLIENT_ID);
        mqttpubClientDemo.startPublishMessage();
    } catch (MqttException e) {
        System.out.println(String.format("program start error,errorMessage[%s]", e.getMessage()));
    }
}
}
```

#### SslUtil.java文件：

```
/*****
Description: SSL工具类，加载client ssl证书配置，忽略服务器证书校验
 *****/

package com.example.demo;

import java.io.ByteArrayInputStream;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyPair;
import java.security.KeyStore;
import java.security.Security;
```



```
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;

import javax.net.ssl.KeyManagerFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.openssl.PEMReader;
import org.bouncycastle.openssl.PasswordFinder;

public class SslUtil {

    /*****
     * 验证并获取SSLSocketFactory
     *****/
    public static SSLSocketFactory getSocketFactory(final String crtFile, final String keyFile) throws Exception
    {
        Security.addProvider(new BouncyCastleProvider());

        // 1、加载客户端证书
        PEMReader reader_client =
            new PEMReader(new InputStreamReader(new
        ByteArrayInputStream(Files.readAllBytes(Paths.get(crtFile))));
        X509Certificate cert = (X509Certificate) reader_client.readObject();
        reader_client.close();

        // 2、加载客户端key
        reader_client = new PEMReader(
            new InputStreamReader(new ByteArrayInputStream(Files.readAllBytes(Paths.get(keyFile))),
            new PasswordFinder() {
                @Override
                public char[] getPassword() {
                    return null;
                }
            }
        );

        // 3、发送客户端密钥和证书到服务器进行身份验证
        KeyStore ks = KeyStore.getInstance(KeyStore.getDefaultType());
        ks.load(null, null);
        ks.setCertificateEntry("certificate", cert);
        ks.setKeyEntry("private-key", ((KeyPair) reader_client.readObject()).getPrivate(), "".toCharArray(), new
        Certificate[]{cert});
        KeyManagerFactory kmf =
        KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
        kmf.init(ks, "".toCharArray());

        // 4、创建socket factory
        SSLContext context = SSLContext.getInstance("TLSv1.2");
        TrustManager[] tms = new TrustManager[1];
        TrustManager miTM = new TrustAllManager();
        tms[0] = miTM;
        context.init(kmf.getKeyManagers(), tms, null);

        reader_client.close();

        return context.getSocketFactory();
    }

    /*****
     * 忽略服务端证书校验
     *****/
    static class TrustAllManager implements TrustManager, X509TrustManager {
```

```
@Override
public X509Certificate[] getAcceptedIssuers() {
    return null;
}

@Override
public void checkServerTrusted(X509Certificate[] certs, String authType)
    throws CertificateException {
}

public boolean isServerTrusted(X509Certificate[] certs) {
    return true;
}

public boolean isClientTrusted(X509Certificate[] certs) {
    return true;
}

@Override
public void checkClientTrusted(X509Certificate[] certs, String authType)
    throws CertificateException {
}
}
```

pom.xml文件:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>mqtt.example</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>7</source>
                    <target>7</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.eclipse.paho/org.eclipse.paho.client.mqttv3 -->
        <dependency>
            <groupId>org.eclipse.paho</groupId>
            <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
            <version>1.2.1</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.bouncycastle/bcprov-jdk16 -->
        <dependency>
            <groupId>org.bouncycastle</groupId>
            <artifactId>bcprov-jdk16</artifactId>
            <version>1.45</version>
        </dependency>
    </dependencies>
</project>
```

## 4.2.8 MQTT Topic

### 4.2.8.1 设备孪生变更

设备孪生更新文档，当孪生变化时，反映孪生变化前、变化后的区别。

#### Topic

\$hw/events/device/{device\_id}/twin/update/document

参数	类型	说明
device_id	String	终端设备ID

#### 使用方式

使用MQTT客户端订阅该Topic。

#### 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
twin	Object	设备孪生变更信息集合，每个孪生以key/value形式存在。value中包含变化前last、变更后current的孪生信息，孪生信息中包含是否可选、孪生metadata包含类型信息、孪生期望状态包含期望值和更新时间、孪生真实状态包含真实值和更新时间等。

#### 示例

终端设备绑定到边缘节点时可收到如下消息。

```
$hw/events/device/{device_id}/twin/update/document
{
  "event_id": "",
  "timestamp": 1557314742122,
  "twin": {
    "state": {
      "last": null,
      "current": {
        "expected": {
          "value": "running",
          "metadata": {
            "timestamp": 1557314742122
          }
        }
      }
    },
    "optional": true,
    "metadata": {
      "type": "string"
    }
  }
}
```

```
}  
}
```

### 4.2.8.2 设备孪生 delta

设备孪生delta事件，当孪生变化时，反映期望值与真实值不一致的孪生信息。

#### Topic

\$hw/events/device/{device\_id}/twin/update/delta

参数	类型	说明
device_id	String	终端设备ID

#### 使用方式

使用MQTT客户端订阅该Topic。

#### 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
twin	Object	设备孪生变更信息集合，每个孪生以key/value形式存在。value中包含是否可选、孪生metadata包含类型信息、孪生期望状态包含期望值和更新时间、孪生真实状态包含真实值和更新时间等。
delta	Map	包含设备孪生期望值与真实值不同的孪生名称和期望值。

#### 示例

终端设备绑定到边缘节点时可收到如下消息。

```
$hw/events/device/{device_id}/twin/update/delta  
  
{  
  "event_id":"b9625811-f34f-4252-bee9-98185e7e1ec7",  
  "timestamp":1557314742131,  
  "twin":{  
    "state":{  
      "expected":{  
        "value":"running",  
        "metadata":{  
          "timestamp":1557314742122  
        }  
      }  
    },  
    "optional":true,  
    "metadata":{  
      "type":"string"  
    }  
  }  
}
```

```

    }
  },
  "delta":{
    "state":"running"
  }
}

```

### 4.2.8.3 设备成员变更

绑定终端设备关系变化。

#### Topic

\$hw/events/node/{node\_id}/membership/updated

参数	类型	说明
node_id	String	节点ID

#### 使用方式

使用MQTT客户端订阅该Topic。

#### 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
added_devices	Array	终端设备信息集合，每个终端设备包含设备的id、name、attributes、twin等信息。
removed_devices	Array	终端设备信息集合，每个终端设备包含设备的id、name、attributes、twin等信息。

#### 示例

终端设备绑定到边缘节点时可收到如下消息。

```

$hw/events/node/{node_id}/membership/updated
{
  "event_id":"04a975ab-fd51-49be-85f5-5967e994f640",
  "timestamp":1557314742136,
  "added_devices":[
    {
      "id":"ab39361a-6fc0-4c94-b919-72b1e08ca690",
      "name":"IEF-device",
      "state":"unknown",
      "attributes":{
        "address":{
          "value":"shenzhen",
          "optional":true,

```

```

        "metadata":{
            "type":"string"
        }
    },
    "twin":{
        "state":{
            "expected":{
                "value":"running",
                "metadata":{
                    "timestamp":1557314434570
                }
            },
            "optional":true,
            "metadata":{
                "type":"string"
            }
        }
    }
},
"removed_devices":null
}
    
```

#### 4.2.8.4 设备属性变更

终端设备属性更新。

#### Topic

\$hw/events/device/{device\_id}/updated

参数	类型	说明
device_id	String	终端设备ID

#### 使用方式

使用MQTT客户端订阅该Topic。

#### 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生时间戳
attributes	Object	终端设备属性变更信息集合，key/value形式存在。key为属性名称，value包含属性值、是否可选或属性metadata包含类型信息等。

#### 示例

终端设备绑定到边缘节点时可收到如下消息。

```
$hw/events/device/{device_id}/updated
{
  "event_id": "",
  "timestamp": 1557314742136,
  "attributes": {
    "address": {
      "value": "shenzhen",
      "optional": true,
      "metadata": {
        "type": "string"
      }
    }
  }
}
```

#### 4.2.8.5 设备成员获取

发布获取终端设备成员信息请求。

#### Topic

`$hw/events/node/{node_id}/membership/get`

参数	类型	说明
node_id	String	节点ID

#### 使用方式

使用MQTT客户端发布该Topic，与[设备成员获取结果](#)配对使用。

#### 参数说明

参数	类型	说明
event_id	String	事件ID，由用户自定义。

#### 示例

```
$hw/events/node/3fbb5b8d-32db-4271-a34f-a013e021b6ce/membership/get
{
  "event_id": "bc876bc-345d-4050-86a8-319a5b13cc10"
}
```

#### 4.2.8.6 设备成员获取结果

订阅终端设备成员信息获取结果。

#### Topic

`$hw/events/node/{node_id}/membership/get/result`

参数	类型	说明
node_id	String	节点ID

## 使用方式

使用MQTT客户端订阅该Topic，与[设备成员获取](#)配对使用。

## 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
devices	Array	终端设备信息集合，每个设备包含设备的id、name、attributes等信息。attributes信息key/value形式存在。key为属性名称，value包含属性值、是否可选或属性metadata包含类型信息等。

## 示例

```
$hw/events/node/3fbb5b8d-32db-4271-a34f-a013e021b6ce/membership/get/result
{
  "event_id":"bc876bc-345d-4050-86a8-319a5b13cc10",
  "timestamp":1557317193524,
  "devices":[
    {
      "id":"ab39361a-6fc0-4c94-b919-72b1e08ca690",
      "name":"IEF-device",
      "state":"unknown",
      "attributes":{
        "address":{
          "value":"longgang",
          "optional":true,
          "metadata":{
            "type":"string"
          }
        }
      }
    }
  ]
}
```

### 4.2.8.7 设备孪生获取

发布获取设备孪生请求。

## Topic

```
$hw/events/device/{device_id}/twin/get
```



参数	类型	说明
device_id	String	终端设备ID

## 使用方式

使用MQTT客户端发布该Topic，与[设备孪生获取结果](#)成对使用。

## 参数说明

参数	类型	说明
event_id	String	事件ID

## 示例

```
$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/get  
{  
  "event_id":"123456"  
}
```

### 4.2.8.8 设备孪生获取结果

订阅设备孪生获取结果。

## Topic

```
$hw/events/device/{device_id}/twin/get/result
```

参数	类型	说明
device_id	String	设备ID

## 使用方式

使用MQTT客户端订阅该Topic，与[设备孪生获取](#)配对使用。

## 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳

参数	类型	说明
twin	Object	终端设备孪生信息集合，每个孪生以key/value形式存在。value中包含是否可选、孪生metadata包含类型信息、孪生期望状态包含期望值和更新时间、孪生真实状态包含真实值和更新时间等。

## 示例

```
$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/get/result
{
  "event_id":"123456",
  "timestamp":1557317510926,
  "twin":{
    "state":{
      "expected":{
        "value":"stop",
        "metadata":{
          "timestamp":1557316778931
        }
      },
      "optional":true,
      "metadata":{
        "type":"string"
      }
    }
  }
}
```

### 4.2.8.9 设备孪生更新

发布设备孪生更新信息。

## Topic

```
$hw/events/device/{device_id}/twin/update
```

参数	类型	说明
device_id	String	终端设备ID

## 使用方式

使用MQTT客户端发布该Topic，与[设备孪生更新结果](#)配对使用。

## 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳

参数	类型	说明
twin	Object	需要修改的设备孪生的信息集合，key/value形式存在。key为需要修改的孪生的名称，value包含需要修改期望状态的期望值或真实状态的真实值。

## 示例

```
$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/update
{
  "event_id":"123457",
  "twin":{
    "state":{
      "actual":{
        "value":"stop"
      }
    }
  }
}
```

### 4.2.8.10 设备孪生更新结果

订阅设备孪生更新结果。

## Topic

\$hw/events/device/{device\_id}/twin/update/result

参数	类型	说明
device_id	String	终端设备ID

## 使用方式

使用MQTT客户端订阅该Topic，与[设备孪生更新](#)配对使用。

## 参数说明

参数	类型	说明
event_id	String	事件ID
timestamp	Int64	事件发生事件戳
twin	Object	设备孪生更新信息集合，每个孪生以key/value形式存在。value中包含是否可选、孪生metadata包含类型信息、孪生期望状态包含期望值和更新时间或孪生真实状态包含真实值和更新时间等。

## 示例

```
$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/update/result
{
  "event_id": "123457",
  "timestamp": 1557317614026,
  "twin": {
    "state": {
      "actual": {
        "value": "stop",
        "metadata": {
          "timestamp": 1557317614026
        }
      },
      "optional": true,
      "metadata": {
        "type": "string"
      }
    }
  }
}
```

### 4.2.8.11 请求加密数据

发布获取加密数据请求。

## Topic

\$hw/{project\_id}/encryptdatas/{encryptdata\_name}/properties/{properties\_name}/  
decrypt

参数	类型	说明
project_id	String	项目ID。获取方式请参见 <a href="#">获取项目ID</a> 。
encryptdata_name	String	加密数据集名称。
properties_name	String	加密数据的“键名”。

## 使用方式

使用MQTT客户端发布该Topic，与[获取加密数据](#)配对使用。

请求时必须使用证书进行安全认证，认证方法请参见[使用证书进行安全认证](#)。

## 参数说明

无

## 示例

发布空消息即可。

### 4.2.8.12 获取加密数据

订阅获取加密数据。

#### Topic

`$hw/{project_id}/encryptdatas/{encryptdata_name}/properties/{properties_name}/plaintext`

参数	类型	说明
project_id	String	项目ID。获取方式请参见 <a href="#">获取项目ID</a> 。
encryptdata_name	String	加密数据集名称。
properties_name	String	加密数据的“键名”。

#### 使用方式

使用MQTT客户端订阅该Topic，与[请求加密数据](#)配对使用。

请求时必须使用证书进行安全认证，认证方法请参见[使用证书进行安全认证](#)。

#### 参数说明

参数	类型	说明
plain_text	String	加密数据的明文值。
ret_message	String	错误信息。

#### 示例

当请求正确时，收到消息如下。

```
{
  "ret_code": 200,
  "plain_text": "xxxxxxxxxx"
}
```

当请求错误时，收到消息如下。

```
{
  "ret_code": 400,
  "ret_message": "xxxxxxxxxx"
}
```

### 4.2.8.13 添加告警

向AOM发送告警。

## Topic

\$hw/alarm/{appname}/add

参数	类型	说明
appname	String	应用名称，任意字符串即可。

## 使用方式

使用MQTT客户端发布该Topic。

## 参数说明

参数	类型	说明
alarmName	String	告警名称，需要支持中英文两个版本，格式为“中文##English”。
alarmId	String	告警ID，需要保证为唯一值。可以参考 <a href="#">alarmId生成方法</a> 。
detailedInformation	String	告警描述，需要支持中英文两个版本，格式为“中文##English”。
url	String	根因分析跳转入口，如果没有则填空。
source	String	告警源，只能是由大小写字母组成的字符串。
cleared	Boolean	清除告警标示。 <ul style="list-style-type: none"><li>• true：表示上报的告警已经清除。</li><li>• false：表示需要手动清除。</li></ul>
policyID	String	告警的规则ID，阈值规则填写ruleId，没有则填写空。
objectInstance	String	定位信息，如果没传此字段，则该字段默认取alarmId的值。
perceivedSeverity	Integer	告警级别 <ul style="list-style-type: none"><li>• 1：紧急</li><li>• 2：重要</li><li>• 3：次要</li><li>• 4：提示</li></ul>
resourceId	Object	告警信息，具体请参见 <a href="#">表4-6</a> 。
resourceType	String	产生告警的资源类型 <ul style="list-style-type: none"><li>• Application（应用）</li><li>• DB（数据库）</li><li>• Host（主机）</li></ul>

参数	类型	说明
eventType	Integer	告警类型 <ul style="list-style-type: none"> <li>• 21: 动态阈值告警</li> <li>• 22: 批量阈值告警</li> <li>• 23: 阈值告警</li> <li>• 24: 系统类告警</li> <li>• 25: 新增/删除探针</li> <li>• 26: agent安装类告警</li> <li>• 27: 配额超限类告警</li> </ul>
probableCause	String	可能原因
proposedRepairActions	String	修复建议

表 4-6 resourceId

参数	类型	说明
namespace	String	资源类型，有如下几种指标 <ul style="list-style-type: none"> <li>• PAAS.CONTAINER表示容器指标</li> <li>• PAAS.NODE表示节点指标</li> </ul>
dimension	Object	<p>维度信息，用于跟监控上报的节点应用信息关联起来。具体请参见<a href="#">表4-7</a>。</p> <p>目前涉及到告警主要是节点和应用两种类型，所以只需要关注节点和应用这两种维度。</p> <p>应用维度可以细分为服务、实例、容器和进程，这四个维度可以选择一个或多个上报。</p> <ul style="list-style-type: none"> <li>• 服务维度需要上报：clusterId、nameSpace和serviceId</li> <li>• 实例维度需要上报：podId和podName</li> <li>• 容器维度需要上报：containerId和containerName</li> <li>• 进程维度需要上报：processId和processName</li> </ul>

表 4-7 Dimension

参数	类型	说明
clusterId	String	项目ID，获取方式请参见 <a href="#">获取项目ID</a> 。
nameSpace	String	默认为default。

参数	类型	说明
nodeIP	String	节点IP。
serviceID	String	<p>服务ID。</p> <ul style="list-style-type: none"> <li>容器应用为{projectId}_{hostid}_{appName}的md5值。</li> <li>进程应用为{projectId}_{hostid}_{进程名}_{进程pid}的md5值。</li> <li>processID定义方式： md5({projectId})_{hostid}_md5({进程名})_{进程pid}。 md5(projectId) 表示求projectId的md5值， md5({进程名})表示求进程名的md5值</li> </ul> <p>这些维度信息需要与监控上报的维度信息一致，不一致会导致无法关联到对应的资源。</p>
podID	String	实例ID。
podName	String	实例名称。
containerID	String	容器ID。
containerName	String	容器名称。
processID	String	进程ID。
processName	String	进程名称。
Application	String	应用名称。

## alarmId 生成方法

### 📖 说明

alarmId可以不使用本方法生成，只要保证唯一就可以。

使用{projectId}\_{产生告警的服务名}\_{维度信息}\_{指标名称}\_{告警类型}\_{规则信息}生成md5值。

其中：

- **projectId**：项目ID，获取方式请参见[获取项目ID](#)。
- **维度信息**：
  - 节点信息：{clusterId}\_{namespace}\_{ip}
  - 容器信息：  
{clusterId}\_{namespace}\_{appName}\_{podName}\_{containerId}
  - 应用信息：{clusterId}\_{namespace}\_{appName}
- **告警类型**：



- 21: 动态阈值告警
  - 22: 批量阈值告警
  - 23: 阈值告警
  - 24: 系统类告警
  - 25: 新增/删除探针
  - 26: agent安装类告警
- **规则信息**: 阈值规则的告警使用阈值规则名称; 业务自己触发的告警(没有规则)则填写NA。动态阈值的则填写policyId。

## 示例

- **节点告警**

```
{
  "alarmName": "测试##test",
  "alarmId": "73ccbccce05de74f9d3dda42f6ecfe20",
  "detailedInformation": "测试##test",
  "url": "",
  "source": "IEF",
  "cleared": false,
  "policyID": "",
  "perceivedSeverity": 4,
  "resourceId": {
    "namespace": "PAAS.NODE",
    "dimension": {
      "clusterId": "e277befa37a64ed1aa25b522e686bc28",
      "nameSpace": "default",
      "nodeIP": "192.168.0.164"
    }
  }
},
"neType": "Host",
"eventType": 23
}
```

- **应用告警**

```
{
  "alarmName": "应用重启test##Application restart",
  "alarmId": "b09076ff565c59d4da0db0c9223781",
  "detailedInformation": "应用重启 test##Application restart test",
  "url": "",
  "source": "IEF",
  "cleared": false,
  "policyID": "",
  "perceivedSeverity": 3,
  "resourceId": {
    "namespace": "PAAS.CONTAINER",
    "dimension": {
      "containerName": "container-e991acd3-864c-4038-8a90-e042eebab496",
      "containerID": "70b385315c8ac507b3de7dfe1258932cea0b53a850b7d030ce7ed0a55c47877c",
      "podID": "0e9ce4fd-b732-11e9-8a30-fa163e9b3546",
      "podName": "hxxkapp1-7898f5bd4b-2lj8z"
    }
  }
},
"neType": "Application",
"eventType": 23
}
```

### 4.2.8.14 清除告警

清除AOM中告警。消息体与**添加告警**保持一致即可, 即清除告警与添加告警只有Topic不同, 其余可以相同。

## Topic

\$hw/alarm/{appname}/clear

参数	类型	说明
appname	String	应用名称，任意字符串即可。

## 使用方式

使用MQTT客户端发布该Topic。

## 参数说明

与[添加告警](#)的参数一致。

### 4.2.8.15 自定义 Topic

IEF支持自定义Topic，Topic根据您的需要自行定义。

使用自定义Topic需要在IEF创建消息路由，并在此处定义Topic，IEF根据消息路由转发Topic中的数据，创建消息路由的具体步骤请参见[边云消息概述](#)。

## Topic

{project\_id}/nodes/{node\_id}/user/{custom\_topic}

## 使用方式

使用MQTT客户端发布该Topic。

## 参数说明

IEF转发自定义Topic内容是透传，可以转发任意内容。

“{custom\_topic}”支持通配符“#”和“+”，可以将多条符合通配规则的消息进行统一转发。

“#”是一个匹配主题中任意层次数的通配符，多层通配符可以表示大于等于0的层次。“+”是单层通配符，只匹配主题的一层。

对于符合通配规则的Topic消息进行最小范围匹配后进行路由消息转发。如消息Topic为123/aaa/567和123/bbb/567，可以配置通配转发规则123+/567进行统一转发。

## 示例

创建路由规则后，您可以往自定义Topic中发送消息（数据），IEF会将消息转发到指定的端点。在IEF中还会记录转发成功次数，如下图所示。

图 4-29 转发成功次数

消息路由名	源端点	目的端点	状态	转发消息数
test-rule-5238	SystemREST (云端)	SystemEventBus (边缘)	禁用	共0条 成功: 0 失败: 0

## 4.3 容器应用管理

### 4.3.1 容器应用

IEF支持下发容器应用到边缘节点，本节主要介绍如何创建自定义边缘应用。

#### 约束与限制

- 边缘节点磁盘占用超过70%时，会启动镜像回收机制回收容器镜像占用的磁盘空间，此时部署容器应用会导致容器启动变慢，请在部署容器应用前规划好边缘节点磁盘空间。
- 创建容器应用时，边缘节点会从容器镜像服务拉取镜像，如果镜像超大且边缘节点下载带宽较小，容器镜像没有拉取完成，从而导致控制台上容器应用显示创建失败。虽然应用创建失败，但容器镜像拉取不会中断，等容器镜像拉取成功后，容器应用的状态会刷新为创建成功。此情况下也可以先将容器镜像拉取到边缘节点，然后再创建容器应用。
- 容器镜像的架构必须与节点架构一致，比如节点为x86，那容器镜像的架构也必须是x86。
- 铂金版可将应用实例数量修改为零。

#### 创建边缘应用

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“边缘应用 > 容器应用”，单击页面右上角“创建容器应用”。

**步骤3** 填写基本信息。

- **名称**：容器应用的名称。
- **实例数量**：容器应用的实例数量。一个容器应用可以拥有多个实例。
- **配置方式**
  - 自定义配置：即从零开始配置容器应用，具体请参见[步骤4-步骤6](#)。
  - 应用模板配置：选择一个已经定义好的应用模板，可以在模板的基础上进行修改，使用应用模板能够帮助您省去重复的工作量。模板的定义与[步骤4-步骤6](#)需要的配置相同，创建模板的方法请参见[应用模板](#)。
- **标签**

标签可用于对资源进行标记，方便分类管理。

图 4-30 基本信息

服务实例 platinum

创建方式 自定义边缘应用

\* 名称

实例数量

配置方式  自定义配置  应用模板配置

部署描述

0/255

标签

还可以创建20个标签。

#### 步骤4 配置容器。

选择需要部署的镜像，单击“使用镜像”。

- 我的镜像：展示了您在[容器镜像服务](#)中创建的所有镜像。
- 他人共享：展示了其他用户共享的镜像，共享镜像是在SWR中操作的，具体请参见[共享私有镜像](#)。

选择镜像后，您可以配置容器的规格。

- **镜像版本**：请选择需要部署的镜像版本。

#### 须知

在生产环境中部署容器时，应避免使用latest版本。因为这会导致难以确定正在运行的镜像版本，并且难以正确回滚。

- **容器规格**：据需要选择容器CPU、内存、昇腾AI加速卡和GPU的配额。
- **昇腾AI加速卡**：容器应用选择的AI加速卡配置与实际部署的边缘节点配置的AI加速卡必须一致，否则会创建应用失败，详见[注册边缘节点](#)。

#### 说明

虚拟化切分后的NPU类型，一个容器只能挂载一个虚拟化NPU，只有当该容器退出后，该虚拟化NPU才能分配给其他容器使用。

AI加速卡支持的NPU类型，如下表。

表 4-8 NPU 类型说明

类型	描述
昇腾310	昇腾310芯片
昇腾310B	昇腾310B芯片
昇腾910	昇腾910芯片
昇腾310P	昇腾310P芯片
昇腾310P-共享模式	昇腾310P芯片支持多容器共享
昇腾310P-1C	昇腾310P芯片按照模板名vir01虚拟化切分后的类型
昇腾310P-2C	昇腾310P芯片按照模板名vir02虚拟化切分后的类型
昇腾310P-2C.1CPU	昇腾310P芯片按照模板名vir02_1c虚拟化切分后的类型
昇腾310P-4C	昇腾310P芯片按照模板名vir04虚拟化切分后的类型。
昇腾310P-4C.3CPU	昇腾310P芯片按照模板名vir04_3c虚拟化切分后的类型

图 4-31 容器配置

The screenshot shows a configuration interface for a container. It includes the following fields and options:

- 镜像名称 (Image Name):** nginx
- 镜像版本 (Image Version):** v3
- 容器名称 (Container Name):** container-66d8f335
- 容器规格 (Container Specification):**
  - CPU配额 (CPU Quota):**
    - 申请 (Request): 0.25 Core
    - 限制 (Limit): 0.25 Core
  - 内存配额 (Memory Quota):**
    - 申请 (Request): 512.00 MIB
    - 限制 (Limit): 512.00 MIB
- AI加速卡 (AI Accelerator):** 不启用 (Disabled), 昇腾AI加速卡 (Ascend AI Accelerator), Nvidia GPU

您还可以对容器进行如下高级配置。

- 运行命令**

容器镜像拥有存储镜像信息的相关元数据，如果不设置生命周期命令和参数，容器运行时运行镜像制作时提供的默认的命令和参数，Dockerfile这两个字段为“Entrypoint”和“CMD”。

如果在创建容器应用时填写了容器的运行命令和参数，将会覆盖镜像构建时的默认命令“Entrypoint”、“CMD”，规则如下：

表 4-9 容器如何执行命令和参数

镜像 Entrypoint	镜像CMD	容器运行命令	容器运行参数	最终执行
[touch]	[/root/test]	未设置	未设置	[touch /root/test]
[touch]	[/root/test]	[mkdir]	未设置	[mkdir]
[touch]	[/root/test]	未设置	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

图 4-32 运行命令



- 运行命令

输入可执行的命令，例如/run/start。

若可执行命令有多个，多个命令之间用空格进行分隔。若命令本身带空格，则需要加引号（" "）。

**说明**

多命令时，运行命令建议用/bin/sh或其他shell，其他全部命令作为参数来传入。

- 运行参数

输入控制容器运行命令的参数，例如--port=8080。

若参数有多个，多个参数以换行分隔。

• 安全选项

- 可以通过开启“特权选项”，使容器拥有root权限，可以访问主机上的设备（如GPU、FPGA）。

- 指定运行用户

IEF默认不改变容器运行的用户，以构建镜像时定义的运行用户来运行。

打开开关后，下方出现运行用户的输入框，可输入范围为0~65534的整数。

指定了运行用户后，应用将以该运行用户来运行。如果镜像的操作系统中没有该用户ID，将导致容器应用启动失败。

• 环境变量配置

容器运行环境中设定的变量。可以在应用部署后修改，为应用提供极大的灵活性。当前支持手动添加、密钥导入、配置项导入和变量引用方式。

- 手动添加支持自定义变量名称和变量值。
- 使用密钥导入，环境变量名称可自定义输入，环境变量值支持引用密钥的属性值，密钥创建方法请参见[密钥](#)。
- 使用配置项导入，环境变量名称可自定义输入，环境变量值支持引用配置项的属性值，配置项创建方法请参见[配置项](#)。
- 变量引用支持引用“hostIP”，即边缘节点的IP地址。

### 📖 说明

IEF不会对用户输入的环境变量进行加密。如果用户配置的环境变量涉及敏感信息，用户需要自行加密后再填入，并在应用中自己完成解密过程。

IEF也不提供任何加解密工具，如果您需要设置加密密文，可以使用其他加解密工具。

### ● 数据存储

您可以通过定义本地卷，将边缘节点本地存储目录挂载到容器中，以实现数据文件的持久化存储。

当前支持如下四种类型的本地卷。

- **hostPath**：将主机某个目录挂载到容器中。hostPath是一种持久化存储，应用删除后hostPath里面的内容依然存在于边缘节点本地硬盘目录中，如果后续重新创建应用，挂载后依然可以读取到之前写入的内容。

您可以将应用日志目录挂载到主机的“/var/IEF/app/log/{appName}”目录，“{appName}”是应用名，边缘节点会将“/var/IEF/app/log/{appName}”目录下后缀为log和trace的文件上传到AOM。

挂载目录为容器内应用的日志路径，如nginx应用默认的日志路径为/var/log/nginx；权限需配置为“读写”。

图 4-33 日志卷挂载

本地卷名称	类型	挂载目录	权限	操作
log	hostPath	/var/IEF/app/log/ngli	/var/log/nginx	读写 删除

- **emptyDir**：一种简单的空目录，主要用于临时存储，支持在硬盘或内存中创建。emptyDir挂载后就是一个空目录，应用程序可以在里面读写文件，emptyDir的生命周期与应用相同，应用删除后emptyDir的数据也同时删除掉。
- **configMap**：存储应用所需配置信息的资源类型，创建方法请参见[配置项](#)。
- **secret**：密钥是一种用于存储应用所需的认证信息、证书、密钥等敏感信息的资源类型，创建方法请参见[密钥](#)。

### 须知

- 请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，应用创建失败。
- 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。

### ● 健康检查

健康检查是指容器运行过程中根据用户需要定时检查容器健康状况或是容器中负载的健康状况。

- 应用存活探针：应用存活探针用于探测容器是否正常工作，不正常则重启实例。当前支持发送HTTP请求和执行命令检查，检测容器响应是否正常。
- 应用业务探针：应用业务探针用于探测业务是否就绪，如果业务还未就绪，就不会将流量转发到当前实例。

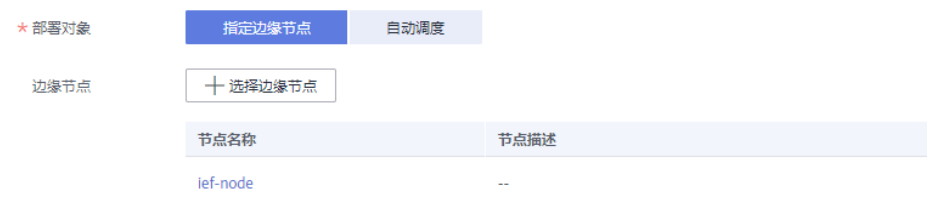
详细的配置说明请参见[健康检查配置说明](#)。

**步骤5** 单击“下一步”，进行部署配置。

选择部署对象，当前支持两种方式。

- **指定边缘节点**

**图 4-34 指定边缘节点**



- **自动调度**

容器应用将在边缘节点组内根据资源用量自动调度。您还可以设置故障策略，当应用实例所在的边缘节点不可用时，是否将应用实例重新调度，迁移到边缘节点组内的其他可用节点。

**说明**

节点组内不可用节点比例大于 0.55 时，将停止自动迁移。  
您还可以设置调度策略，详细信息请参见[亲和与反亲和调度](#)。

**图 4-35 自动调度**



**迁移时间窗**：参数表示负载在具有指定污点的节点上可以保持不被驱逐的最长时间。用户未配置迁移时间窗时，默认的迁移时间是300s，在这故障时间窗内保持节点仍可以被调度。

您还可以对容器进行如下高级配置。



- **重启策略**

- 总是重启：当应用容器退出时，无论是正常退出还是异常退出，系统都会重新拉起应用容器。  
当使用节点组时，重启策略为“总是重启”。
- 失败时重启：当应用容器异常退出时，系统会重新拉起应用容器，正常退出时，则不再拉起应用容器。
- 不重启：当应用容器退出时，无论是正常退出还是异常退出，系统都不再重新拉起应用容器。

---

**须知**

只有选择了“总是重启”，容器应用才能在创建后更新升级，修改实例数量和访问配置。

- **Host PID**

启用时容器与边缘节点宿主机共享PID命名空间，这样在容器或边缘节点上能够进行互相操作，比如在容器中启停边缘节点的进程、在边缘节点启停容器的进程。

启用此选项要求边缘节点软件版本大于等于2.8.0。

**步骤6** 单击“下一步”，进行访问配置。

容器访问支持端口映射和主机网络两种方式。

---

**注意**

当部署在同一个边缘节点的容器端口冲突时，容器会启动失败。

- **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。
- 端口映射可以选择主机网卡，请注意端口映射不支持选择IPv6地址类型的网卡。
- 端口映射中的主机端口可以选择指定和自动分配。选择自动分配可以避免多实例的端口冲突导致容器启动失败。

**说明**

选择自动分配时，请输入合适且充足的端口范围，避免端口冲突。

- **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。

- **服务**

点击“添加服务”，填写服务名称，访问端口，容器端口选择端口映射中的容器端口，协议支持HTTP和TCP两种。

图 4-36 添加服务

容器网络

网络类型 **端口映射** 主机网络

端口映射

容器名称	容器端口	主机端口	操作
container-4c...	80	列表选择 自动选择网卡 指定 8080	删除

添加端口映射

边缘DNS

---

服务 (Service)

服务名称 **添加服务**

服务名称

访问端口	容器端口	协议	操作
1-65535		HTTP	删除

添加端口

如果容器访问配置为端口映射，不添加端口的情况下是不能创建服务的。

服务定义了更加精细化的访问方案，您也可以在应用创建完成后再创建服务，具体请参见[创建服务](#)。

**步骤7** 单击“下一步”，确认容器应用的规格，确认无误后单击“创建”。

----结束

## 查看应用运维信息

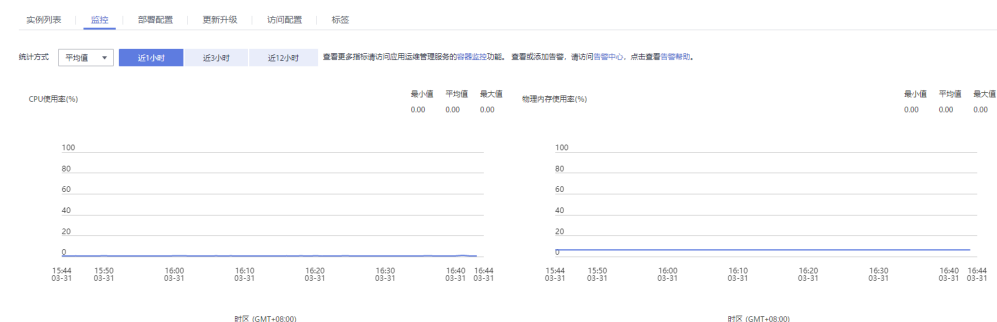
应用部署后，您可以在IEF控制台查看应用的CPU、内存等信息。您还可以通过访问应用运维管理（AOM）的“容器监控”功能，查看更多监控指标，在“告警”功能查看或添加告警。

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边缘应用 > 容器应用”，单击容器应用名称。

**步骤3** 选择“监控”页签即可查看应用监控信息。

图 4-37 应用监控信息



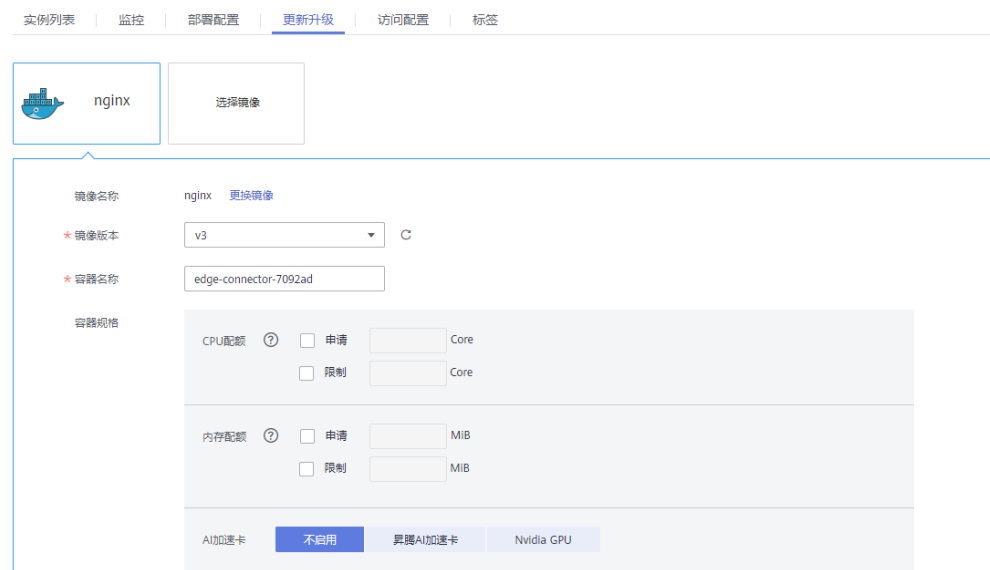
----结束

## 更新升级应用

应用部署后，您可以更新升级应用，应用升级采用滚动升级的方式，即先创建新应用实例，然后删除老的应用实例。

- 步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。
- 步骤2** 选择左侧导航栏“边缘应用 > 容器应用”，单击容器应用名称。
- 步骤3** 选择“更新升级”页签，修改容器相关配置。具体配置参数与**步骤4**中一致。

图 4-38 更新升级



- 步骤4** 配置修改后，单击“提交”。

----结束

## 修改访问配置

应用部署后，您可以修改应用的访问配置。

- 步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。
- 步骤2** 选择左侧导航栏“边缘应用 > 容器应用”，单击容器应用名称。
- 步骤3** 选择“访问配置”页签，修改容器相关配置。具体配置参数与**步骤6**中一致。

图 4-39 修改访问配置



**步骤4** 配置修改后，单击“提交”。

----结束

## 4.3.2 应用模板

容器应用模板用于定义用户的边缘应用，用户需要指定容器应用的容器镜像、配置信息、磁盘挂载信息以及资源占用信息。

应用需要基于镜像创建，用户首先需要先制作镜像并上传至镜像仓库。

### 创建应用模板

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边缘应用 > 应用模板”，单击页面右上角“创建应用模板”。

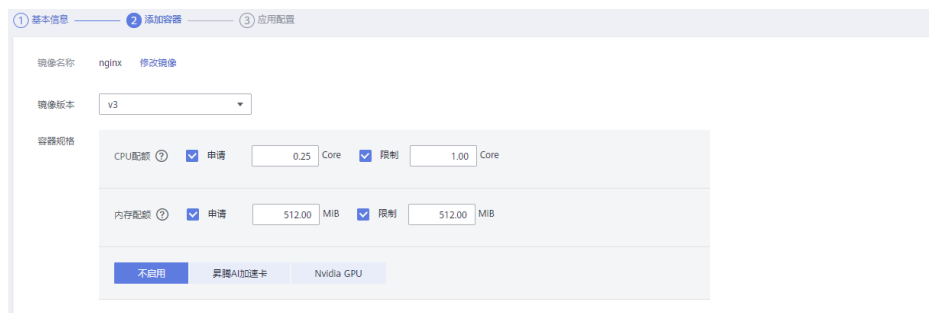
**步骤3** 填写基本信息。

- **名称（必填）**：应用模板名称。
- **版本（必填）**：版本号。
- **别名**：应用模板名称以外的一种名称。
- **架构**：选择应用支持的架构。
- **描述**：模板的描述信息。
- **标签**：标签可用于对应用模板进行标记，方便分类管理。此处的标签只用于标识应用模板，可以在搜索时使用标签过滤应用模板。

**步骤4** 单击“下一步”，添加容器。

1. 选择需要部署的镜像，单击“使用镜像”。
  - 我的镜像：展示了您在容器镜像服务中创建的所有镜像。
  - 他人共享：展示了其他用户共享的镜像。
2. 单击“下一步”，配置容器规格。

图 4-40 镜像信息



- **镜像版本**：请选择需要部署的镜像版本。
- **容器规格**：据需要选择容器CPU、内存、昇腾AI加速卡和GPU的配额。
- **昇腾AI加速卡**

容器应用选择的AI加速卡配置与实际部署的边缘节点配置的AI加速卡必须一致，否则会创建应用失败，详见[注册边缘节点](#)。

 说明

虚拟化切分后的NPU类型，一个容器只能挂载一个虚拟化NPU，只有当该容器退出后，该虚拟化NPU才能分配给其他容器使用。

昇腾AI加速卡支持的NPU类型，如下表。

表 4-10 NPU 类型说明

类型	描述
昇腾310	昇腾310芯片
昇腾310B	昇腾310B芯片
昇腾910	昇腾910芯片
昇腾310P	昇腾310P芯片
昇腾310P-共享模式	昇腾310P芯片支持多容器共享
昇腾310P-1C	昇腾310P芯片按照模板名vir01虚拟化切分后的类型
昇腾310P-2C	昇腾310P芯片按照模板名vir02虚拟化切分后的类型
昇腾310P-2C.1CPU	昇腾310P芯片按照模板名vir02_1c虚拟化切分后的类型
昇腾310P-4C	昇腾310P芯片按照模板名vir04虚拟化切分后的类型。
昇腾310P-4C.3CPU	昇腾310P芯片按照模板名vir04_3c虚拟化切分后的类型

**步骤5** 单击“下一步”，配置应用信息。

- **启动命令**

容器镜像拥有存储镜像信息的相关元数据，如果不设置生命周期命令和参数，容器运行时运行镜像制作时提供的默认命令和参数，Dockerfile中这两个字段为“Entrypoint”和“CMD”。

如果在创建容器应用时填写了容器的运行命令和参数，将会覆盖镜像构建时的默认命令“Entrypoint”、“CMD”，规则如下：

表 4-11 容器如何执行命令和参数

镜像 Entrypoint	镜像CMD	容器运行命令	容器运行参数	最终执行
[touch]	[/root/test]	未设置	未设置	[touch /root/test]
[touch]	[/root/test]	[mkdir]	未设置	[mkdir]

镜像 Entrypoint	镜像CMD	容器运行命令	容器运行参数	最终执行
[touch]	[/root/test]	未设置	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

图 4-41 启动命令



### 运行命令

输入可执行的命令，例如/run/start。

若可执行命令有多个，多个命令之间用空格进行分隔。若命令本身带空格，则需要加引号（" "）。

#### 说明

多命令时，运行命令建议用/bin/sh或其他的shell，其他全部命令作为参数来传入。

### 运行参数

输入控制容器运行命令的参数，例如--port=8080。

若参数有多个，多个参数以换行分隔。

### 环境变量

容器运行环境中设定的变量。可以在部署应用时修改。

单击“添加环境变量”，输入变量名称，变量值支持手动添加、密钥导入、配置项导入和变量引用方式。

- 手动添加支持自定义变量名称和变量值。
- 使用密钥导入，环境变量名称可自定义输入，环境变量值支持引用密钥的属性值，密钥创建方法请参见[密钥](#)。
- 使用配置项导入，环境变量名称可自定义输入，环境变量值支持引用配置项的属性值，配置项创建方法请参见[配置项](#)。
- 变量引用支持引用“hostIP”，即边缘节点的IP地址。

#### 说明

IEF不会对用户输入的环境变量进行加密。如果用户配置的环境变量涉及敏感信息，用户需要自行加密后再填入，并在应用中自己完成解密过程。

IEF也不提供任何加解密工具，如果您需要设置加密密文，可以使用其他平台的加解密工具。

- **卷**

卷是指容器运行过程中使用的存储卷，当前支持如下四种类型。

- **hostPath**: 将主机某个目录挂载到容器中。hostPath是一种持久化存储，应用删除后hostPath里面的内容依然存在于边缘节点本地硬盘目录中，如果后续重新创建应用，挂载后依然可以读取到之前写入的内容。

您可以将应用日志目录挂载到主机的“/var/IEF/app/log/{appName}”目录，“{appName}”是应用名，边缘节点会将“/var/IEF/app/log/{appName}”目录下后缀为log和trace的文件上传到AOM。

图 4-42 日志卷挂载

本地卷名称	类型	挂载目录	权限	操作
log	hostPath	/var/IEF/app/log/ap	/home/log	只读 删除

- **emptyDir**: 一种简单的空目录，主要用于临时存储，支持在硬盘或内存中创建。emptyDir挂载后就是一个空目录，应用程序可以在里面读写文件，emptyDir的生命周期与应用相同，应用删除后emptyDir的数据也同时删除掉。
- **configMap**: 存储应用所需配置信息的资源类型，创建方法请参见[配置项](#)。
- **secret**: 密钥是一种用于存储应用所需的认证信息、证书、密钥等敏感信息的资源类型，创建方法请参见[密钥](#)。

### 须知

- 请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，应用创建失败。
- 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。

- **选项**

- **特权选项**

可以通过开启“特权选项”，使容器拥有root权限，可以访问主机上的设备（如GPU、FPGA）。

- **指定运行用户**

IEF默认不改变容器运行的用户，以构建镜像时定义的运行用户来运行。

打开开关后，下方出现运行用户的输入框，可输入范围为0~65534的整数。指定了运行用户后，应用将以该运行用户来运行。如果镜像的操作系统中没有该用户ID，将导致容器应用启动失败。

- **重启策略**

**总是重启**: 当应用容器退出时，无论是正常退出还是异常退出，系统都会重新拉起应用容器。

**失败时重启**: 当应用容器异常退出时，系统会重新拉起应用容器，正常退出时，则不再拉起应用容器。

**不重启**: 当应用容器退出时，无论是正常退出还是异常退出，系统都不再重新拉起应用容器。

### - 容器网络

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。

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。

### ● 健康检查

- 应用存活探针：应用存活探针用于探测容器是否正常工作，不正常则重启实例。当前支持发送HTTP请求和执行命令检查，通过检测容器响应是否正常。
- 应用业务探针：应用业务探针用于探测业务是否就绪，如果业务还未就绪，就不会将流量转发到当前实例。

详细的配置说明请参见[健康检查配置说明](#)。

**步骤6** 配置完成后，单击“创建”。

----结束

## 创建新的应用模板版本

边缘应用模板支持创建多个应用版本，方便您管理边缘应用。

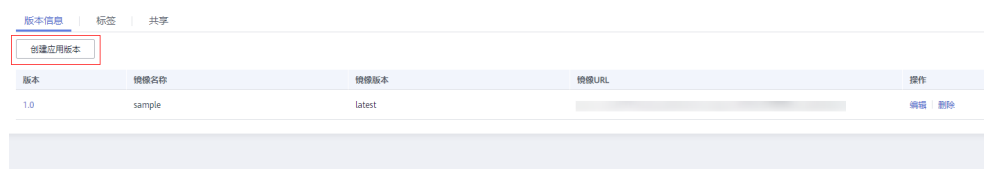
**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“边缘应用 > 应用模板”。

**步骤3** 单击需要增加新版本的应用模板，进入“应用模板详情”页面。

**步骤4** 单击页面左下角的“创建应用版本”。

图 4-43 创建应用版本



**步骤5** 填入应用版本，单击“下一步”，填写模板详细信息，具体请参照[创建应用模板](#)，完成新版本的应用模板创建。

----结束

## 共享模板

边缘应用模板可以共享给同区域的其他服务实例使用。

### 📖 说明

v2 及以上铂金版实例版本和旧版本之间无法共享应用模板。

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

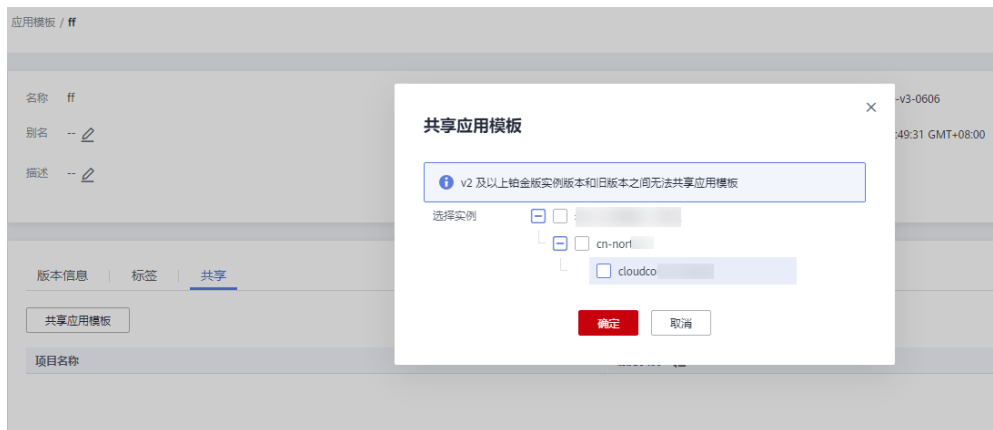
**步骤2** 选择左侧导航栏的“边缘应用 > 应用模板”。



**步骤3** 单击需要共享的应用模板，进入“应用模板详情”页面。

**步骤4** 选择“共享”页签，单击“共享应用模板”，勾选服务实例。

**步骤5** 单击“确定”。



---结束

### 4.3.3 亲和与反亲和调度

#### 亲和与反亲和调度策略

在创建容器应用时，可以设置亲和/反亲和调度策略，例如将某类应用部署到某些特定的节点、不同应用部署到不同的节点等等。

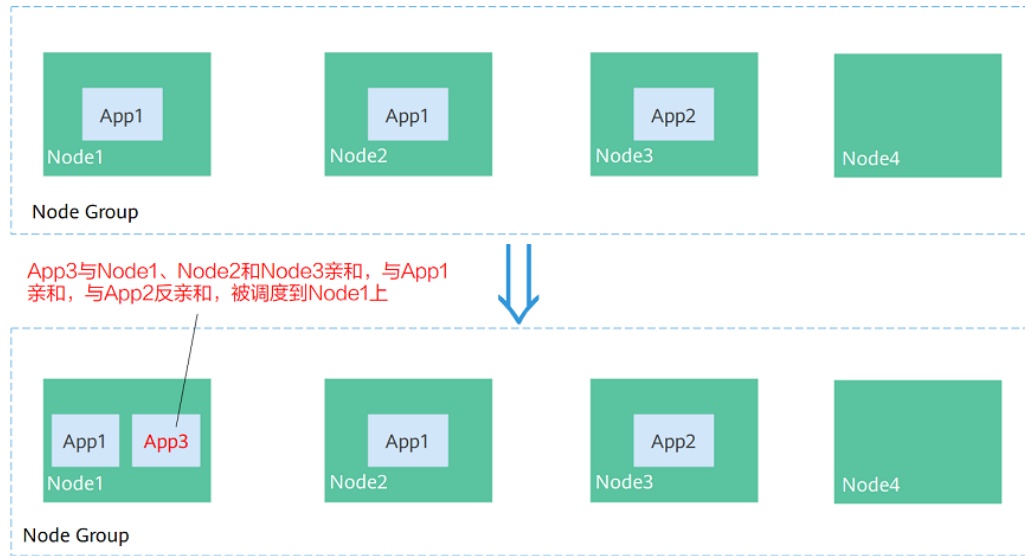
IEF当前支持简易的调度策略，具体如下。

- 亲和性
  - 容器应用与节点亲和：当容器应用与节点亲和时，容器应用只会调度到亲和的这些节点中。在设置节点的亲和与反亲和时，如果设置了与某些节点亲和，那就没有必要再设置与其他节点反亲和，因为与这些节点亲和了，必然不会调度到其他节点。
  - 容器应用与容器应用亲和：当容器应用A与容器应用B亲和时，容器应用A只会调度到容器应用B所在的节点。
- 反亲和性
  - 容器应用与节点反亲和：当容器应用与节点反亲和时，容器应用不会调度到反亲和的这些节点中。
  - 容器应用与容器应用反亲和：当容器应用A与容器应用B反亲和时，容器应用A不会调度到容器应用B所在的节点。

#### 亲和反亲和策略示例

您可以同时设置多个亲和反亲和对象。例如当前节点组有4个节点，Node1、Node2、Node3和Node4，容器应用App1的实例运行在Node1和Node2上，容器应用App2的实例运行在Node3上。这时您需要创建一个容器应用App3，容器应用App3与节点Node1、Node2、Node3亲和，与容器应用App1亲和，与容器应用App2反亲和，那容器应用App3只会被调度到Node1、Node2上。下图示例中，App3只有一个实例，被调度到了Node1上。

图 4-44 App3 被调度到 Node1 上



## 添加亲和对象

在“创建容器应用 > 部署配置”时，在选择了边缘节点组后，可以在调度策略中配置亲和调度策略。

**步骤1** 单击“添加亲和对象”。

图 4-45 添加亲和对象



**步骤2** 在弹出的对话框中，选择亲和的节点或容器应用，然后单击“确定”。

图 4-46 选择亲和对象



----结束

## 添加反亲和对象

在“创建容器应用 > 部署配置”时，在选择了边缘节点组后，可以在调度策略中配置反亲和和调度策略。

**步骤1** 单击“添加反亲和对象”。

图 4-47 添加反亲和对象



**步骤2** 在弹出的对话框中，选择反亲和的节点或容器应用，然后单击“确定”。

图 4-48 选择反亲和对象



----结束

### 4.3.4 配置项

配置项（ConfigMap）是一种用于存储工作负载所需配置信息的资源类型，配置项允许您将配置文件从容器镜像中解耦，从而增强容器工作负载的可移植性。

配置项价值如下：

- 使用配置项功能可以帮您管理不同环境、不同业务的配置。
- 方便您部署相同工作负载的不同环境，配置文件支持多版本，方便您进行更新和回滚工作负载。
- 方便您快速将您的配置以文件的形式导入到容器中。

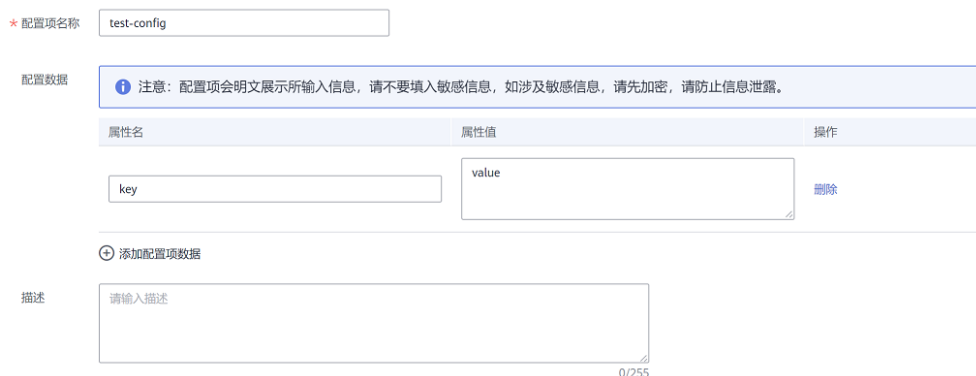
### 创建配置项

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边缘应用 > 应用配置”，单击页面右上角“创建配置项”。

**步骤3** 填写配置数据。

图 4-49 配置项



- **配置项名称：**输入配置项名称。

- **配置数据**：配置数据是键值对形式，请输入属性名和属性值。

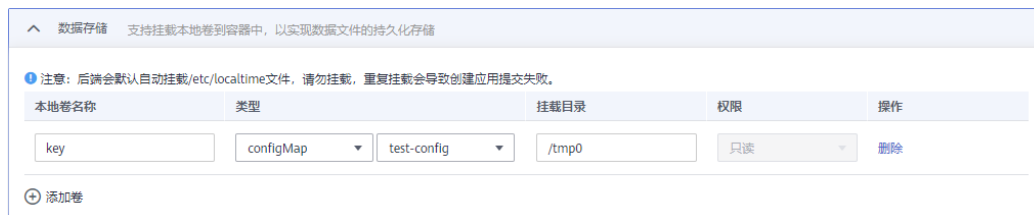
**步骤4** 单击“创建”，即创建配置项成功，返回到配置项列表页面。

----结束

## 配置项的使用

您可以在创建容器应用高级配置中选择数据存储时使用ConfigMap。

图 4-50 使用 ConfigMap



ConfigMap挂载到容器后，会根据ConfigMap的内容在挂载目录下创建文件，每条配置数据（属性名-属性值）为一个文件，其中属性名即文件的名称，属性值为文件的内容。例如ConfigMap的属性名为“key”，属性值为“value”，这条ConfigMap挂载到/tmp0目录下，那么挂载成功后，在/tmp0目录下就存在一个名为“key”的文件，其内容为“value”。

### 4.3.5 密钥

密钥是一种用于存储应用所需的认证信息、证书、密钥等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。例如，在“数据存储”中挂载secret类型的卷，使其成为容器中的文件；或者在“环境变量”中加载，使其成为容器中的环境变量。

#### 须知

密钥的使用可能涉及用户敏感信息，用户输入敏感信息前，需要自行对敏感信息进行加密，并在应用中对加密的数据进行解密后方可使用，请用户注意。

## 创建密钥

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边缘应用 > 应用配置”，单击页面右上角“创建密钥”。

**步骤3** 填写密钥信息。

图 4-51 创建密钥

服务实例 platinum

\* 密钥名称 test-secret

密钥类型 Opaque

密钥数据

注意：填入信息应先进行加密再进行Base64编码，并确保应用能对其进行解密。

* 属性名	属性值	操作
<input type="text"/>	<input type="text"/>	删除

+ 添加密钥数据

描述 请输入描述

0/255

- **密钥名称**：密钥的名称。
- **密钥类型**：当前支持创建“Opaque”类型的密钥。“Opaque”类型的数据是一个键值对形式，要求属性值为“Base64”编码格式。Base64编码方法请参见[如何进行Base64编码](#)。
- **密钥数据**：密钥数据是键值对形式，请输入属性名和属性值，其中属性值必须为Base64编码格式的字符串。

**步骤4** 单击“创建”，即创建密钥成功，返回到密钥列表页面。

----结束

## 如何进行 Base64 编码

对字符串进行Base64加密，可以直接使用`echo -n 要编码的内容 | base64`命令即可，示例如下：

```
root@ubuntu:~# echo -n "example value" | base64
ZXhhbXBsZSB2YX1ZQ==
```

## 密钥的使用

您可以在创建容器应用高级配置中选择数据存储时使用Secret。

图 4-52 使用密钥

数据存储服务 支持挂载本地卷到容器中，以实现数据文件的持久化存储

注意：后端会默认自动挂载/etc/localtime文件，请勿挂载，重复挂载会导致创建应用提交失败。

本地卷名称	类型	挂载目录	权限	操作
key	secret	test-secret	/tmp0	只读 删除

+ 添加卷

Secret挂载到容器后，会根据Secret的内容在挂载目录下创建文件，每条密钥数据（属性名-属性值）为一个文件，其中属性名即文件的名称，属性值为文件的内容。例如

Secret的属性名为“key”，属性值为“ZXhhbXBsZSB2YWx1ZQ==”，这条Secret挂载到/tmp0目录下，那么挂载成功后，在/tmp0录下就存在一个名为“key”的文件，其内容为“example value”。

## 4.3.6 加密数据

加密数据是一种用于对敏感信息进行加密的资源类型，可以帮助您对敏感数据进行加密保存，边缘应用可以通过访问边缘MQTT Server获取明文数据。

### 创建加密数据

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边缘应用 > 应用配置”，单击页面右上角“创建加密数据”，填写相关参数。

图 4-53 创建加密数据

服务实例

\* 加密数据集名称 data

\* 加密数据项

注意：加密数据的数据项至少有一项，会明文展示所输入信息，请不要填入敏感信息，如涉及敏感信息，请先加密，请防止信息泄露。

键名	值	操作
key	.....	删除
	.....	

添加加密数据

描述 请输入描述

0/255

加密数据以键值对形式存储，其中值需要填写两次。每个加密数据集可以添加多条加密数据。

**步骤3** 单击“创建”。

创建后可以查看数据，但出于数据隐私保护，暂不支持查看明文数据。且您可以编辑或删除加密数据。

图 4-54 加密数据

加密数据集 | 绑定节点

添加 注意：加密数据项至少保留一项

键名	值	操作
key	.....	编辑 删除

----结束

## 绑定加密数据到边缘节点

您可以在多处绑定加密数据到边缘节点。加密数据可以绑定到非运行中状态的边缘节点，当边缘节点恢复到运行中状态时会自动同步已绑定的加密数据到边缘节点。

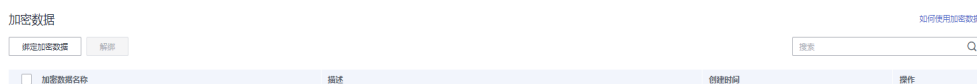
- 在加密数据详情页，您可以选择“绑定节点 > 绑定边缘节点”。

图 4-55 绑定边缘节点



- 在边缘节点详情页选择“配置”，在加密数据部分单击“绑定加密数据”。

图 4-56 绑定加密数据



## 加密数据的使用

加密数据绑定到边缘节点后，在边缘节点上使用MQTT客户端就可以获取到加密数据。

请求时必须使用证书进行安全认证，认证方法请参见[使用证书进行安全认证](#)。

**步骤1** 订阅[获取加密数据](#)。

**Topic:** \$hw/{project\_id}/encryptdatas/{encryptdata\_name}/properties/{properties\_name}/plaintext

**步骤2** 发布[请求加密数据](#)。

**Topic:** \$hw/{project\_id}/encryptdatas/{encryptdata\_name}/properties/{properties\_name}/decrypt

请求发布后，[步骤1](#)中就能收到解密后的明文数据。

----结束

### 4.3.7 健康检查配置说明

健康检查是指容器运行过程中根据用户需要定时检查容器健康状况或是容器中负载的健康状况。

- 应用存活探针：应用存活探针用于探测容器是否正常工作，不正常则重启实例。当前支持发送HTTP请求和执行命令检查，检测容器响应是否正常。
- 应用业务探针：应用业务探针用于探测业务是否就绪，如果业务还未就绪，就不会将流量转发到当前实例。

IEF支持HTTP请求检查和执行命令检查两种方式。



## HTTP 请求检查

向容器发送HTTP GET请求，如果探针收到2xx或3xx，说明容器是健康的。

例如下图这个配置，IEF会在容器启动10秒（延迟时间）后，发送HTTP GET请求到“http://{实例IP}/healthz:8080”，如果在2秒（超时时间）内没有响应则视为检查失败；如果请求响应的状态码为2xx或3xx，则说明容器是健康的。

### 说明

这里无需填写主机地址，默认直接使用实例的IP（即往容器发送请求），除非您有特殊需求。

图 4-57 HTTP 请求检查

检查方式	<input type="radio"/> 不配置	<input checked="" type="radio"/> HTTP请求检查	<input type="radio"/> 执行命令检查
路径	<input type="text" value="/healthz"/>		
端口	<input type="text" value="8080"/>		
主机地址	<input type="text" value="请求的主机地址, 默认为实例IP"/>		
协议	<input checked="" type="radio"/> HTTP	<input type="radio"/> HTTPS	
延迟时间 / 秒	<input type="text" value="10"/>		
超时时间 / 秒	<input type="text" value="2"/>		

## 执行命令检查

探针执行容器中的命令并检查命令退出的状态码，如果状态码为0则说明健康。

例如下图中这个配置，IEF会在容器启动10秒（延迟时间）后，在容器中执行 **cat /tmp/healthy**命令，如果在2秒（超时时间）内没有响应，则视为检查失败；如果命令成功执行并返回0，则说明容器是健康的。

图 4-58 执行命令检查

检查方式	<input type="radio"/> 不配置	<input type="radio"/> HTTP请求检查	<input checked="" type="radio"/> 执行命令检查
执行命令	<input type="text" value="cat /tmp/healthy"/>		
延迟时间 / 秒	<input type="text" value="10"/>		
超时时间 / 秒	<input type="text" value="2"/>		

## 4.4 应用网络

## 4.4.1 操作场景

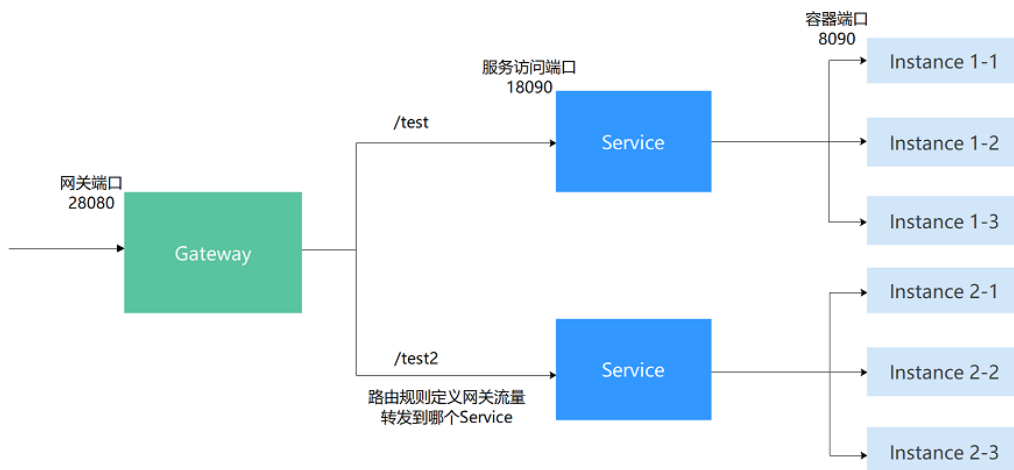
应用网格提供非侵入式的微服务治理解决方案，支持完整的生命周期管理和流量治理，支持负载均衡等多种治理能力。

应用网格中有如下概念：

- 服务：服务定义了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。
- 网关：网关可以将通过IEF部署的边缘应用暴露给外部应用访问，并能通过路由规则转发流量。

如下图所示，访问网关应用所在边缘节点的IP:网关端口/URI，就能将流量转发到对应的服务，再由服务转发到对应的后端应用实例。

图 4-59 网关流量转发路径示意图



IEF 基于 KubeEdge 生态构建，将云原生容器化应用程序编排能力延伸到了边缘。但是，在边缘计算场景下，网络拓扑较为复杂，不同区域中的边缘节点往往网络不互通，并且应用之间流量的互通是业务的首要需求，IEF 兼容社区应用网格的能力，可以提供服务和网关元数据的下发。用户使用时，需要配合在边缘节点部署社区 EdgeMesh 应用，可参考：<https://github.com/kubeedge/edgемesh/>。

## 4.4.2 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，从而实现节点上应用间的相互访问。

### 创建服务

#### 📖 说明

对于铂金版V3版本服务实例，创建的服务确保可以访问的前提是：发起访问的容器应用需要开启边缘DNS，同时必须在发起访问的节点上和被访问的节点上安装edgемesh插件。

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“应用网格 > 服务列表”，单击页面右上角“创建服务”。

**步骤3** 填写信息。

- **服务名称**：填写服务名称。
- **绑定应用**：选择服务绑定的应用。
- **端口配置**
  - 访问端口：访问容器应用时使用的端口。
  - 容器端口：容器应用实际监听的端口。
  - 协议：访问负载的通信协议，可选择HTTP或TCP。

**图 4-60** 创建服务

服务名称

服务名称, 只允许英文小写字母、数字、中划线, 最大长度64, 英文小写字母开头, 数字或小写字母结尾

绑定应用 ccccc-0dd2c [更换](#)  
主机网络

端口配置

访问端口	容器端口	协议	操作
<input type="text" value="18090"/>	<input type="text" value="8090"/>	HTTP	删除

[+ 添加端口](#)

**步骤4** 单击“创建”。

创建完成后，可以在服务列表中查看服务对应的内部访问域名。

**图 4-61** 查看服务内部访问域名和访问端口

<input type="checkbox"/> 服务名称	内部访问域名	关联应用	访问端口 -> 容器端口/协议	更新时间	操作
<input type="checkbox"/> test	<a href="#">test.e78acc02d9d141eda5cabe88fa356f8.svc.cluster.local</a>	nginx	8080->80/HTTP	2022/11/14 15:42:15 GMT+08:00	<a href="#">更新</a> <a href="#">流量策略</a> <a href="#">删除</a>

----结束

## 更新服务

服务支持更新端口配置，包括访问端口、容器端口和协议。

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

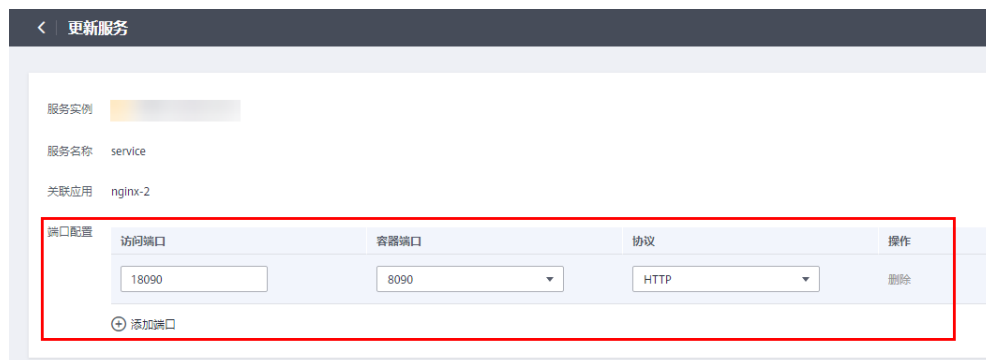
**步骤2** 选择左侧导航栏的“应用网格 > 服务列表”，单击服务所在行的“更新”，如下图所示。

**图 4-62** 更新服务

<input type="checkbox"/> 服务名称	内部访问域名	关联应用	访问端口 -> 容器端口/协议	更新时间	操作
<input type="checkbox"/> service	<a href="#">service.e78acc02d9d141eda5cabe88fa356f8.svc.cluster.local</a>	nginx-2	18090->8090/HTTP	2022/09/13 11:36:59 GMT+08:00	<a href="#">更新</a> <a href="#">流量策略</a> <a href="#">删除</a>

**步骤3** 服务仅支持更新**端口配置**。

图 4-63 更新服务的端口配置



步骤4 单击“更新”。

----结束

## 配置流量策略

服务支持配置流量策略，即当后端存在多个应用实例时，配置服务流量转发到应用实例的策略。

步骤1 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

步骤2 选择左侧导航栏的“应用网格 > 服务列表”，单击服务所在行的“流量策略”，如下图所示。

图 4-64 流量策略

服务名称	内部应用域名	关联应用	访问端口 -> 容器端口/协议	操作
service	service-e78acc0209d141eda5ca9e88fa35f6f8.svc.cluster.local	ccccc-0d42c	18090->8090/HTTP	更新 流量策略 删除
service-example	service-example-e78acc0209d141eda5ca9e88fa35f6f8.svc.cluster.local	ief-zb-nginx	8888->80/HTTP	更新 流量策略 删除

步骤3 在弹出的窗口中选择流量策略。

当前支持如下几种流量策略。

- 负载均衡算法：支持轮询和随机两种方式转发。
- 会话保持：支持基于HTTP头部的Cookie、User-Agent和自定义三种方式转发。

图 4-65 选择流量策略



### 📖 说明

使用HTTP协议的服务可以配置负载均衡算法和会话保持，但是使用TCP协议的服务目前只支持配置负载均衡算法，不能配置会话保持。

**步骤4** 单击“确定”。

----结束

## 删除服务

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“应用网格 > 服务列表”，单击服务所在行的“删除”，或者勾选要删除的服务单击服务名称上方的“删除”，如下图所示。

图 4-66 删除服务



**步骤3** 在弹出的窗口中的输入框输入“DELETE”，单击“是”，确认删除。

----结束

## 4.4.3 网关

应用网格提供了网关管理功能，根据路由规则就能将流量转发到对应的服务，再由服务转发到对应的后端应用实例。

## 创建网关

### 📖 说明

对于铂金版V3版本服务实例，创建的网关确保可以访问的前提是：必须在网关应用所在的节点上安装edgemesg-gateway插件。

服务网关在微服务实践中可以做到统一接入、流量管控、安全防护、业务隔离等功能，在IEF创建网关的操作步骤如下。

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“应用网格 > 服务网关”，单击页面右上角“创建网关”。

**步骤3** 填写网关信息。

- **网关名称**：填写网关名称。
- **绑定应用**：选择网关应用。

### 📖 说明

仅铂金版V2版本服务实例需绑定应用。对于铂金版V3版本服务实例，如果提供网关服务的边缘节点已安装IEF提供的edgemesg-gateway插件，可忽略该选项。

- **端口配置**

端口：网关使用哪个端口访问后端应用。

协议：网关使用哪种协议访问后端应用。目前支持HTTP和HTTPS协议，若设置成HTTPS协议，则还需要配置“协议版本”、“证书密钥”、“加密套件”三个参数；选择HTTP协议，只需继续配置下面的域名列表。

协议版本：选择协议版本。

证书密钥：添加SSL证书。公钥文件是公开的，密钥文件是通过申请证书获取的，因此公钥、密钥文件需用户自己提供。

### 须知

配置https协议需要的SSL证书密钥，只能在此处添加。

加密套件：选择SSL证书中包含的加密套件和客户端支持的加密套件。

域名列表：允许访问后端哪些域名，支持正则匹配，比如“\*”代表所有域名。

- **http路由**

- URI匹配规则：支持前缀匹配、精确匹配和正则匹配。
- URI：填写URI。
- 目标服务：选择规则匹配哪个服务，即该URI的流量转发到哪个服务。
- 服务访问端口：选择服务访问应用的端口。

图 4-67 网关配置

The screenshot shows a configuration form for a gateway. It includes the following elements:

- 网关名称** (Gateway Name): A text input field containing "test".
- 绑定应用** (Bind Application): A dropdown menu with "选择应用" (Select Application) and a help icon.
- 端口配置** (Port Configuration): A sub-form containing:
  - 端口** (Port): A text input field containing "28090".
  - 协议** (Protocol): A dropdown menu with "HTTP" selected.
  - 域名列表** (Domain List): A text input field containing "\*" and a plus icon.
- http路由** (HTTP Routing): A section with a note: "注意：URI匹配规则将按照列表顺序依次执行，添加规则时请注意规则顺序，避免转发错误。" (Note: URI matching rules will be executed in order of the list. Pay attention to the rule order when adding rules to avoid forwarding errors.) Below the note is a table:

URI匹配规则	URI	目标服务	服务访问端口	操作
前缀匹配	/test	test-service0001	80->80/HTTP	删除
- At the bottom, there is a plus icon and the text "添加http路由" (Add HTTP Routing).

**步骤4** 单击“确定”。

----结束

## 更新网关

服务网关支持更新端口配置以及http路由规则。

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“应用网格 > 服务网关”，单击网关所在行的“更新”，如下图所示。

图 4-68 更新网关

网关名称	关联应用	端口个数	更新时间	操作
jacke-gw	jacke-easyapp	1	2022/11/22 16:08:49 GMT+08:00	<b>更新</b> 配置路由 删除

**步骤3** 只有网关名称和绑定应用不支持更改，其他**网关信息**均可在此步骤更新。

图 4-69 更新网关信息

服务实例 

\* 网关名称 wx-gateway001

绑定应用 

端口配置

* 端口	8888	* 协议	HTTP
* 域名列表	* <input type="text"/>		

\* http路由

注意：URI匹配规则将按照列表顺序依次执行，添加规则时请注意规则顺序，避免转发错误。

URI匹配规则	URI	目标服务	服务访问端口	操作
前缀匹配	/	test-service0001	80->80/HTTP	删除

添加http路由

**步骤4** 单击“更新”。

----结束

## 配置路由

您可以给已创建好的网关添加多个路由，配置多个转发策略。

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“应用网格 > 服务网关”，单击网关所在行的“配置路由”，如下图所示。

**步骤3** 在弹出的窗口中可**配置http路由**，单击“是”，更新虚拟服务的成功。

图 4-70 配置路由

http路由

注意：URI匹配规则将按照列表顺序依次执行，添加规则时请注意规则顺序，避免转发错误。

URI匹配规则	URI	目标服务	服务访问端口	操作
前缀匹配	/test	service	18090->8090/HTTP	删除

添加http路由

是 否

----结束

## 删除网关

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“应用网格 > 服务网关”，单击网关所在行的“删除”，或者勾选要删除的网关单击网关名称上方的“删除”，如下图所示。

图 4-71 删除网关



步骤3 在弹出的窗口中的输入框输入“DELETE”，单击“是”，确认删除。

----结束

## 4.5 边云消息

### 4.5.1 边云消息概述

IEF提供了边云消息路由功能，您可以配置消息路由，IEF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，可以是终端设备、云服务等。
- 消息路由：消息转发的路径。

#### 消息端点

IEF提供如下默认消息端点：

- **SystemEventBus**：边缘节点上的MQTT，代表节点通信，可以作为源端点向云上发送数据，也可以作为目的端点，接收云上消息。端点资源为边缘节点MQTT Topic。
- **SystemREST**：云端的REST网关接口，可以作为源端点，向边缘侧发送REST请求。端点资源为REST请求的路径。

您还可以创建如下消息端点：

- **Service Bus**：边缘节点上的事务请求处理端点，可以作为目的端点，处理文件上传请求。端点资源为REST请求的路径。
- **DIS**：数据接入服务，可以作为目的端点，接收由IEF转发的数据。端点资源为在DIS服务中创建的DIS通道。
- **APIG**：API网关服务，可以作为目的端点，接收由IEF转发的数据。端点资源为在API网关服务中创建的API地址。

#### 消息路由

目前支持如下几种消息转发路径：

- **SystemREST到Service Bus**：通过调用云端的REST Gateway接口，获取边缘节点上的文件服务。
- **SystemREST到SystemEventBus**：通过调用云端的REST Gateway接口，向边缘节点中的SystemEventBus（MQTT broker）发送消息。



- SystemEventBus到DIS/APIG服务：您可以将终端设备数据发送到边缘节点 SystemEventBus（MQTT broker）的自定义Topic中，IEF会将这些数据转发到DIS通道或APIG后端地址。数据转发到DIS通道或者APIG后端地址后，您可以提取这些数据，并对数据进行处理分析。这条路径需要在创建消息路由时自定义MQTT Topic，自定义Topic的详细说明请参见[自定义Topic](#)。

使用消息路由功能，您只需要先[创建消息端点](#)，然后[创建消息路由](#)。

## 创建消息端点

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边云消息 > 消息端点”。

**步骤3** 单击页面右上角“创建消息端点”，填写相关参数。

图 4-72 创建消息端点



- **消息端点类型:** 选择类型，当前支持DIS、APIG和Service Bus。
- **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的端点需要填写，范围为1-65535。

**步骤4** 单击“确定”，即创建消息端点成功，返回到消息端点列表页面。

----结束

## 创建消息路由

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边云消息 > 消息路由”。

**步骤3** 单击页面右上角“创建消息路由”。

**步骤4** 填写相关参数。

- **消息路由名称:** 输入消息路由名称。

 **注意**

消息路由和**系统订阅**是同一种资源，命名不能冲突。

- **源端点**：选择源端点，SystemREST或SystemEventBus。
- **源端点资源**：输入源端点资源。
  - **选择SystemREST源端点时**  
填写Rest路径，如/abc/ab
  - **选择SystemEventBus源端点时**  
自定义topic：选择节点并输入一个topic  
设备数据上传通路：选择节点，选择该节点上绑定的MQTT协议的终端设备。

 **说明**

选择设备时，只允许选择MQTT协议的终端设备。

- **目的端点**：选择目的端点，如SystemEventBus。
- **目的端点资源**：输入目的端点资源。

**步骤5** 单击“创建”，即创建规则成功，返回到规则列表页面。

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。

---结束

## 消息路由停用/启用

- **停用消息路由**：在指定消息路由右侧单击“停用”。  
消息路由停用后，该路由规则不再生效，系统不会再对发送到源端点指定资源的消息进行转发处理。
- **启用消息路由**：在指定已停用消息路由右侧单击“启用”。  
消息路由启用后，该路由规则生效，系统会对发送到源端点指定资源的消息进行转发处理。

## 4.5.2 云端下发消息到边缘节点

### 操作场景

IEF支持从云端SystemREST下发消息到边缘节点的SystemEventBus（MQTT broker）。

通过调用开放在公网的IEF云端的REST Gateway接口，结合节点ID和自定义Topic，向边缘节点中的SystemEventBus发送消息。您的终端设备可以通过订阅对应的自定义Topic进行消息接收，实现自定义Topic的从云到边的消息下发。

使用方法主要分如下三个步骤。

1. **创建消息路由**
2. **发送消息**

### 3. 接收消息

#### 创建消息路由

SystemREST和SystemEventBus为系统默认端点，无需创建。其中SystemREST代表该Region下IEF云端REST Gateway接口。SystemEventBus代表边缘节点的MQTT broker。

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边云消息 > 消息路由”。

**步骤3** 单击页面右上角“创建消息路由”。

**步骤4** 填写相关参数。

图 4-73 创建消息路由

* 消息路由名称	<input type="text" value="SystemREST2SystemEventBus"/>
* 源端点	<input type="text" value="SystemREST"/>
* 源端点资源	<input type="text" value="/"/>
* 目的端点	<input type="text" value="SystemEventBus"/>
* 目的端点资源	<input type="text" value="/"/>
描述	<input type="text" value="请输入描述"/>

0/255

- **消息路由名称**：输入消息路由名称，如SystemREST2SystemEventBus。

#### ⚠ 注意

消息路由和[系统订阅](#)是同一种资源，命名不能冲突。

- **源端点**：选择SystemREST。
- **源端点资源**：填写符合URL路径的以“/”开头的字符串，可以使用{}作为入参进行层级模糊匹配，如aaa/{any-str}/bbb可以匹配/aaa/cc/bbb或者/aaa/ddd/bbb。整个源端点资源为调用REST接口时的匹配字段。

- **目的端点**：选择SystemEventBus。
- **目的端点资源**：目的端点资源为消息转发至MQTT时对应的Topic前缀。

#### 📖 说明

源端点资源和目的端点资源都填写为“/”时，IEF会对所有发送到REST接口的请求转发到对应节点的MQTT中，且消息的Topic与请求URL一致。

**步骤5** 单击“创建”。

在“消息路由”页面可以看到已创建的路由。

图 4-74 消息路由

消息路由...	源端点	目的端点	状态	转发消息数	
<input type="checkbox"/>	SystemREST2Syst...	SystemREST 云端	SystemEventBus 边缘	启用	共0条
<input type="checkbox"/>	/	/		成功:0	失败: 0

----结束

## 发送消息

从云端SystemREST下发到边缘节点的SystemEventBus，您首先需要获取SystemREST的Endpoint，然后你需要构造请求，将消息通过路由发送至SystemEventBus。

**步骤1** 获取SystemREST的Endpoint。

1. 登录IEF管理控制台，在“总览”页面切换实例为铂金版。
2. 选择左侧导航栏“边云消息 > 消息端点”。在SystemREST所在行“消息端点属性”列，public的取值即为SystemREST的Endpoint，如下图。

图 4-75 SystemREST 的 Endpoint

消息端点名称	类型	位置	消息端点属性
<input type="checkbox"/> SystemEventBus	eventbus 系统端点	边缘	
<input type="checkbox"/> SystemREST	rest 系统端点	云端	public = restep.ief2.cn-north-4.myhuaweicloud.com

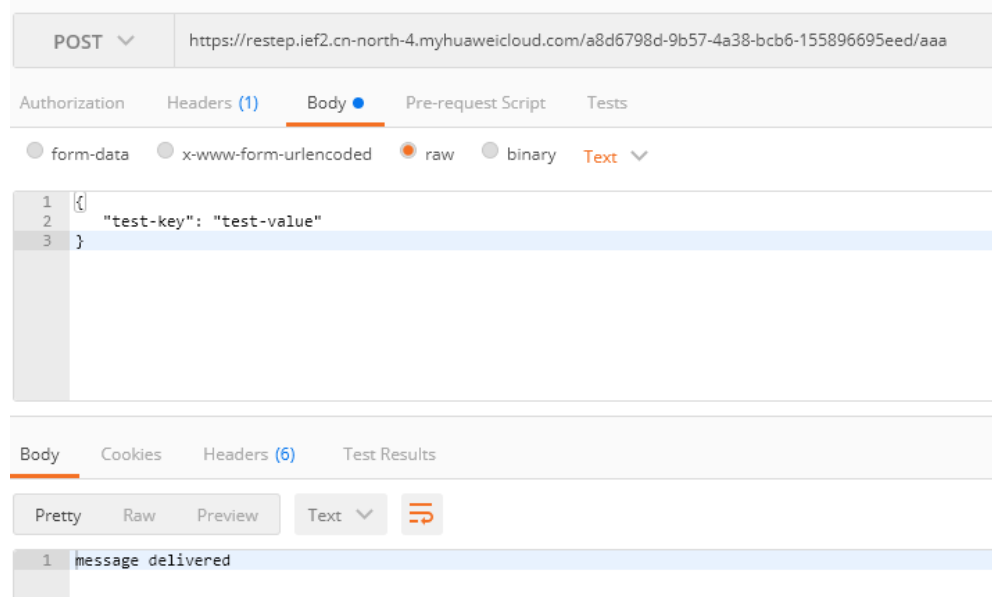
**步骤2** 构造请求，发送消息。

往SystemREST发送消息，需构造HTTPS请求，具体信息如下。

- Method: POST
- URL: **https://{rest\_endpoint}/{node\_id}/{topic}**，rest\_endpoint即**步骤1**中获取的Endpoint，node\_id为边缘节点ID，topic为消息的主题，即**创建消息路由**中定义的源端点资源。
- Body: 发送的消息内容，由用户自定义。
- Header: X-Auth-Token，取值为该用户在IAM获取的有效Token，Token获取方法请参见**用户Token**。

例如使用Postman发送如下消息。

图 4-76 发送消息



----结束

## 接收消息

**步骤1** 登录边缘节点。

**步骤2** 使用MQTT客户端接收消息。

当前边缘节点MQTT支持两种模式。

- 一种是内置的MQTT broker（使用8883端口），需要使用节点证书认证，然后订阅对应Topic接收消息，具体使用方式请参见[使用证书进行安全认证](#)。
- 另外一种是使用外置的MQTT broker（使用1883端口），需要先安装第三方MQTT broker，然后订阅对应Topic接收消息。

这里介绍使用外置的MQTT broker的方式，外置的MQTT broker需要先安装MQTT broker，例如安装Mosquitto，步骤如下。

- 对于Ubuntu操作系统，可以使用如下命令安装mosquitto：

```
apt-get install mosquitto
systemctl start mosquitto
systemctl enable mosquitto
```
- 对于CentOS操作系统，使用如下命令安装mosquitto：

```
yum install epel-release
yum install mosquitto
systemctl start mosquitto
systemctl enable mosquitto
```

安装完成后，使用订阅命令订阅，订阅后如果有消息发送，就会收到消息，如下所示。其中#表示订阅任何主题，可以替换为指定的主题，如/aaa、/bbb等。

```
[root@ief-node ~]# mosquitto_sub -t '#' -d
Client mosq-m02iwjsp4j2ISMw6rw sending CONNECT
Client mosq-m02iwjsp4j2ISMw6rw received CONNACK (0)
Client mosq-m02iwjsp4j2ISMw6rw sending SUBSCRIBE (Mid: 1, Topic: #, QoS: 0, Options: 0x00)
Client mosq-m02iwjsp4j2ISMw6rw received SUBACK
Subscribed (mid: 1): 0
Client mosq-m02iwjsp4j2ISMw6rw received PUBLISH (d0, q0, rQ, mQ, '/aaa', ... (31 bytes))
```

```
{  
  "test-key": "test-value"  
}
```

----结束

## 4.5.3 边缘节点上报消息到云端

### 操作场景

IEF支持从边缘节点上报消息到云端。

您可以将消息发送到边缘节点SystemEventBus（MQTT broker）的自定义Topic中，IEF会将这些数据转发到DIS通道或APIG后端地址。数据转发到DIS通道或者APIG后端地址后，您可以提取这些数据，并对数据进行处理分析。

本章节使用DIS端点作为示例，APIG端点的使用方法类似，主要分如下几个步骤。

1. [创建消息端点](#)
2. [购买DIS接入通道](#)
3. [创建消息路由](#)
4. [发送消息](#)

### 创建消息端点

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边云消息 > 消息端点”。

**步骤3** 单击页面右上角“创建消息端点”，选择DIS，填写消息端点名称。

图 4-77 创建消息端点



**步骤4** 单击“确定”。

----结束

### 购买 DIS 接入通道

往DIS发消息需要购买DIS接入通道。

**步骤1** 登录数据接入服务DIS控制台。

**步骤2** 单击右侧“购买接入通道”，填写对应参数，如下图所示。

图 4-78 购买接入通道

\* 计费模式

\* 区域   
不同区域的资源之间内网不互通。请选择靠近您客户的区域，可以降低网络时延、提高访问速度。

\* 通道名称   
可使用自动生成的由前缀“dis-”加4位随机字符或数字组成的名称，例如：dis-HvB1，也可自定义。

\* 通道类型   ?

\* 分区数量 ?  1   您最多可使用49个分区。申请扩大配额  
选择的规格为：普通通道 | 1 个分区 | 通道理论容量：1 MB/秒（接入）；2 MB/秒（读取）

\* 生命周期（小时）  24

\* 源数据类型    ?

\* 自动扩容 ?

企业项目 ?

高级配置

**步骤3** 单击“立即购买”，确认产品规格无误后，单击“提交”。

----结束

## 创建消息路由

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边云消息 > 消息路由”。

**步骤3** 单击页面右上角“创建消息路由”。

**步骤4** 填写相关参数，如下图所示。

图 4-79 创建消息路由

服务实例 controller-v2-8011-1

\* 消息路由名称

\* 源端点

\* 源端点资源

**注意：** 用户可向边缘节点的Eventbus中的自定义Topic发布数据，并设置消息路由将发布在该Topic中的数据上传至OIS（数据接入服务）或API网关服务

选择节点   
e78acc02d9d141eda5ca8e88fa35f6f8/nodes/0/user/

\* 目的端点

\* 目的端点资源  C

描述

0/255

- **消息路由名称：** 输入消息路由名称。

**注意**

消息路由和[系统订阅](#)是同一种资源，命名不能冲突。

- **源端点：** 选择“SystemEventBus”。
- **源端点资源：** 选择“自定义topic”，选择发送消息的边缘节点，填写topic名称。
- **目的端点：** 选择[创建消息端点](#)创建的端点。
- **目的端点资源：** 选择[购买DIS接入通道](#)中购买的通道。

#### 说明

- 请记录此处的Topic，如图中红框所示。创建成功后，也可以在消息路由列表中“源端点”列查看。
- 自定义Topic后，需将完整的Topic（如图中红框所示）用于消息发送。

**步骤5** 单击“创建”。

----结束

## 发送消息

在边缘节点使用MQTT客户端发送消息。

此处需要放到[创建消息路由](#)中指定的Topic，如下图所示使用mosquitto\_pub发送。

```
[root@ief-node ~]# mosquitto_pub -t '05e1aef9040010e22fcc009adecb056/nodes/7092ad14-adee-4a09-b969-1505bbdecef5/user/aaa' -d -m '{ "edgemsg": "msgToCloud" }'
```

```
Client mosq-p5LouPQIW2gx0JPKRF sending CONNECT
Client mosq-p5LouPQIW2gx0JPKRF received CONNACK (0)
Client mosq-p5LouPQIW2gx0JPKRF sending PUBLISH (d0, q0, r0, m1, '05e1aef9040010e22fcc009adecb056/nodes/7092ad14-adee-4a09-b969-1505bbdecef5/user/aaa', ... (26 bytes))
Client mosq-p5LouPQIW2gx0JPKRF sending DISCONNECT
```



消息发送后，您可以在消息路由处看到已经成功转发一条消息，如下图所示。

图 4-80 转发消息数

消息路由...	源端点	目的端点	状态	转发消息数
<input type="checkbox"/>	example2dis	SystemEventBus 边缘	dis_example 云端	共1条
<input type="checkbox"/>	05e1aef9040010e22fcc009adecb056/nodes/7092a...	dis-ief	<span>启用</span>	成功: 1 失败: 0

您可以在DIS界面看到有消息流入，如下图所示。

图 4-81 DIS 数据监控



## 获取数据

数据转发到DIS通道后，您可以提取这些数据，并对数据进行处理分析。DIS的数据获取方法请参见[从DIS获取数据](#)。

## 4.5.4 系统订阅

### 操作场景

IEF提供系统订阅，您可以订阅IEF资源的变更事件，当资源创建、更新、删除时，IEF会发送消息到您指定的APIG端点，以便您及时感知资源的变化。

系统订阅是边云消息的一种特定实现，IEF发送特定资源的事件消息到指定Topic，并调用APIG的API，用户可通过API调用感知资源的变化。

### 支持订阅的事件

当前支持订阅如下事件。

表 4-12 支持订阅的事件

系统事件	Topic	资源类型	操作
实例创建	\$hw/events/instance/+/created	应用实例 ( instance )	created
实例更新	\$hw/events/instance/+/updated	应用实例 ( instance )	updated
实例删除	\$hw/events/instance/+/deleted	应用实例 ( instance )	deleted
应用删除	\$hw/events/deployment/+/deleted	容器应用 ( deployment )	deleted
应用创建	\$hw/events/deployment/+/created	容器应用 ( deployment )	created
应用更新	\$hw/events/deployment/+/updated	容器应用 ( deployment )	updated
节点创建	\$hw/events/edgeNode/+/created	边缘节点 ( edgeNode )	created
节点更新	\$hw/events/edgeNode/+/updated	边缘节点 ( edgeNode )	updated
节点删除	\$hw/events/edgeNode/+/deleted	边缘节点 ( edgeNode )	deleted
节点上线	\$hw/events/edgeNode/+/online	边缘节点 ( edgeNode )	online
节点离线	\$hw/events/edgeNode/+/offline	边缘节点 ( edgeNode )	offline
设备创建	\$hw/events/device/+/created	终端设备 ( device )	created
设备更新	\$hw/events/device/+/updated	终端设备 ( device )	updated
设备删除	\$hw/events/device/+/deleted	终端设备 ( device )	deleted

## 系统订阅流程

系统订阅流程如下：

1. 在APIG创建供IEF调用的API。
2. 在IEF中创建APIG端点。
3. 在IEF中创建系统订阅。

## 创建 API

在APIG中创建API，具体请参见[APIG 快速入门](#)。

该API用于给IEF调用，发送系统事件消息。

## 创建 APIG 端点

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边云消息 > 消息端点”。

**步骤3** 单击页面右上角“创建消息端点”，填写相关参数。

图 4-82 创建消息端点



- **消息端点类型**：选择APIG。
- **消息端点名称**：输入消息端点名称。

**步骤4** 单击“确定”，即创建消息端点成功，返回到消息端点列表页面。

----结束

## 创建系统订阅

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏“边云消息 > 系统订阅”。

**步骤3** 单击页面右上角“创建系统订阅”，填写相关参数。

图 4-83 创建系统订阅

\* 系统订阅名称

\* 订阅主题

\* 目的端点

\* 目的端点资源   
请输入api网关的URL地址,http或https开头,URL中的域名需要在APIG中已配置,配置方法请看[绑定域名](#)

描述

0/255

- **系统订阅名称：**输入系统订阅名称。



**注意**

系统订阅和[消息路由](#)是同一种资源，命名不能冲突。

- **订阅主题：**订阅事件的主题和操作，如创建容器应用，当前支持订阅的事件如[表 4-12](#)。
- **目的端点：**[创建APIG端点](#)中创建的端点。
- **目的端点资源：**[创建API](#)中创建的API资源。

**步骤4** 单击“创建”，完成系统订阅创建。

----结束

## 订阅后说明

创建系统订阅后，当有系统事件发生，在IEF控制台系统订阅列表中，会记录消息转发的次数。

图 4-84 转发消息数

<input type="checkbox"/>	系统订...	订阅主题	目的端点	状态	转发消息数
<input type="checkbox"/>	subscription	deployment/created	apig_example <a href="#">云镜</a> https://4e93a29712a245d2b700b70b0316ab0...	启用	共0条 成功:0 失败: 0

同时IEF会调用APIG中注册的API，请求消息体如下所示。请求消息体采用标准的CloudEvents格式，CloudEvents详细信息请参见[这里](#)。

```
{
  "data": {
    "event_type": "instance",
```

```

"operation": "created",
"timestamp": 134567677,
"topic": "$hw/events/deployment/+/created",
"name": "xxx",
"attributes": {"ID": "x"}
},
"datacontenttype": "application/json",
"source": "sysevents",
"id": "xxx",
"time": "2020-11-5 xxx"
}
    
```

- data: 系统事件的数据。
  - event\_type: 资源类型, String类型。
  - operation: 资源的操作类型, String类型。
  - topic: 消息发送的Topic, String类型。
  - timestamp: 事件产生的时间戳, Uint64类型。
  - name: 资源名称, String类型。
  - attributes: 资源的属性, 删除资源时消息中无此参数, Object类型。
- datacontenttype: 系统事件数据内容的格式, String类型。
- source: 系统事件的来源, String类型。
- id: 系统事件ID, String类型。
- time: 系统事件产生时间, String类型。

当前支持的资源类型、操作和Topic如下表所示。

表 4-13 支持订阅的事件

系统事件	Topic	资源类型	操作
实例创建	\$hw/events/instance/+/created	应用实例 ( instance )	created
实例更新	\$hw/events/instance/+/updated	应用实例 ( instance )	updated
实例删除	\$hw/events/instance/+/deleted	应用实例 ( instance )	deleted
应用删除	\$hw/events/deployment/+/deleted	容器应用 ( deployment )	deleted
应用创建	\$hw/events/deployment/+/created	容器应用 ( deployment )	created
应用更新	\$hw/events/deployment/+/updated	容器应用 ( deployment )	updated
节点创建	\$hw/events/edgeNode/+/created	边缘节点 ( edgeNode )	created
节点更新	\$hw/events/edgeNode/+/updated	边缘节点 ( edgeNode )	updated

系统事件	Topic	资源类型	操作
节点删除	\$hw/events/edgeNode/+/deleted	边缘节点 ( edgeNode )	deleted
节点上线	\$hw/events/edgeNode/+/online	边缘节点 ( edgeNode )	online
节点离线	\$hw/events/edgeNode/+/offline	边缘节点 ( edgeNode )	offline
设备创建	\$hw/events/device/+/created	终端设备 ( device )	created
设备更新	\$hw/events/device/+/updated	终端设备 ( device )	updated
设备删除	\$hw/events/device/+/deleted	终端设备 ( device )	deleted

## 4.6 批量管理

### 4.6.1 批量节点注册

#### 使用场景

如果您需要对大量相同类型的边缘节点进行管理、更新与维护，这时使用[注册边缘节点](#)的方法逐一纳管边缘节点会有些繁琐。

IEF提供批量节点注册功能，使得相同类型的边缘节点能够预安装软件，在上电联网后能自动纳管到IEF中。批量节点注册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。

#### 批量节点注册流程

批量纳管边缘节点的流程如下：

1. 准备边缘节点，边缘节点需要满足一定的规格要求，请参见[配置边缘节点环境](#)在节点上完成环境配置。
2. [创建批量节点注册作业](#)，获取配置文件和注册软件，并预安装到边缘节点上，具体请参见[批量纳管边缘节点](#)。如果在启动脚本配置注册命令，设备上电联网后会自动纳管成为IEF的边缘节点。

#### 创建批量节点注册作业

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“批量管理 > 节点注册”，单击页面右上角的“批量节点注册”。

**步骤3** 配置基本信息。其中带“\*”标志的参数为必填参数。

- **名称**：批量节点注册作业的名称。
- **描述**：作业的描述信息。
- **标签**：标签可用于对资源进行标记，方便分类管理。
- **属性**：属性是键值对形式，请输入属性名和属性值。

**步骤4** 单击页面右下角的“创建”，节点注册可选“通过证书注册”和“通过token注册”两种方式。

- 方式一：通过证书注册。下载配置文件、EdgeCore Register和EdgeCore Installer。

#### 须知

为保障节点安全，您现在必须下载配置文件和工具，稍后将无法找回。

图 4-85 下载配置文件、EdgeCore Register 和 EdgeCore Installer

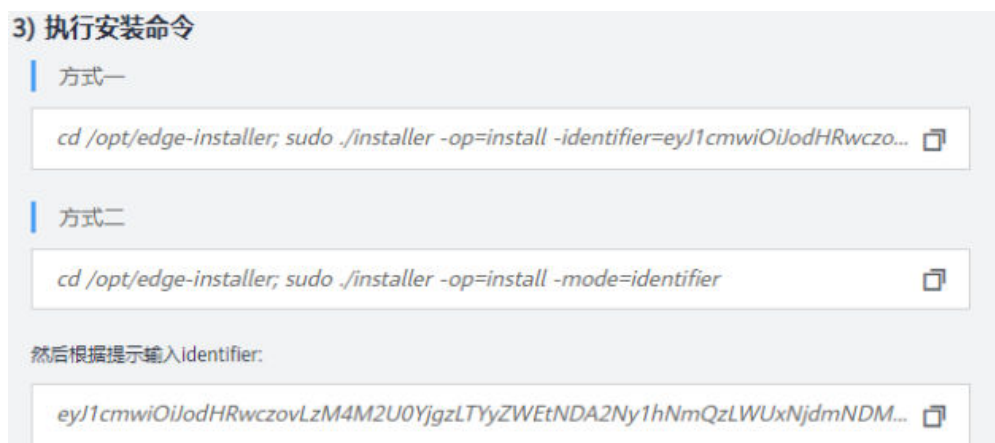


- a. 根据页面提示，单击“下载cert\_1.tar.gz配置文件”下载配置文件。
  - b. 根据您边缘节点的CPU架构选择EdgeCore Register和EdgeCore Installer，单击“下载EdgeCore Register”和“下载EdgeCore Installer”。
- 方式二：通过token注册。下载边缘节点安装工具并保存安装凭证。

图 4-86 下载边缘核心软件



图 4-87 安装命令



- a. 根据您边缘节点的CPU架构选择边缘节点安装工具，单击“下载EdgeCore Installer”。
- b. 保存界面中的安装命令。安装命令有两种执行方式，您可以任选其中的一种方式保存即可。

**步骤5** 将下载的文件上传至边缘节点，稍后参见[批量纳管边缘节点](#)操作指南纳管边缘节点。

----结束

## 批量纳管边缘节点

按照批量节点注册作业完成页的操作指南，在边缘节点执行纳管操作。

**步骤1** 以具备sudo权限的用户登录边缘节点。

**步骤2** 执行如下命令解压配置文件、EdgeCore Register和EdgeCore Installer。

```
sudo tar -zxvf edge-installer_1.0.10_x86_64.tar.gz -C /opt
```

```
sudo tar -zxvf edge-register_2.0.10_x86_64.tar.gz -C /opt
```

```
sudo tar -zxvf cert_batchNodes.tar.gz -C /opt/edge-register/（“通过token注册”时不需要执行此命令）
```

*edge-installer\_1.0.10\_x86\_64.tar.gz*、*edge-register\_2.0.10\_x86\_64.tar.gz*和*cert\_batchNodes.tar.gz*请替换为[创建批量节点注册作业](#)下载的文件名。

**步骤3** （可选）配置节点预置信息文件。



批量纳管边缘节点会采用默认配置进行节点纳管，其中

- 节点名称为节点hostname和mac地址的组合“hostname\_mac”
- GPU和NPU默认不启用
- Docker默认启用

如需自定义配置节点信息，可以参考以下操作配置节点信息文件。

```
cd /opt/edge-register
```

```
vi nodeinfo
```

在nodeinfo文件中填入节点的配置信息，其格式如下所示（下面样例罗列了所有可配置的参数，可以根据场景需求配置其中的参数值，对于不需要自定义配置的参数可以删除示例中对应的参数配置）：

包括节点名称、描述、是否启用GPU、NPU、NPU类型、属性以及日志配置等，具体参数解释请参见[API参考 > 注册边缘节点](#)。

```
{
  "name": "nodename",
  "description": "",
  "enable_gpu": false,
  "enable_npu": true,
  "npu_type": "****",
  "enable_docker": true,
  "attributes": [
    {
      "key": "key1",
      "value": "value1"
    }
  ],
  "log_configs": [
    {
      "component": "app",
      "type": "local",
      "level": "debug",
      "size": 100,
      "rotate_num": 5,
      "rotate_period": "daily"
    }
  ],
  "device_infos": [
    {
      "device_ids": ["15696983-5ee6-43b4-9653-5d8512813dcc"],
      "relation": "camera",
      "comment": "devicedescription"
    }
  ],
  "mqtt_config": {
    "enable_mqtt": false,
    "mqttps": []
  },
  "tags": [
    {
      "key": "name",
      "value": "value"
    }
  ]
}
```

**步骤4** 执行如下命令，纳管边缘节点。

- 通过证书注册  

```
cd /opt/edge-register; ./register --mode=cert
```

- 通过token注册

```
cd /opt/edge-register; ./register --mode=identifier --identifier=token注册的凭证
```

*token*注册的凭证请替换为[创建批量节点注册作业](#)时保存的安装凭证identifier字段。

执行完成后，可以根据[查看纳管的节点](#)查看节点是否注册成功。

您也可以配置上电启动脚本，在边缘节点上电时自动执行注册命令，在上电脚本中添加注册命令：`cd /opt/edge-register; ./register --mode=cert`

如果注册命令的执行结果返回值为0，则表示注册成功。

----结束

## 查看纳管的节点

批量纳管的节点，可以在控制台查看。

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“批量管理 > 节点注册”，在右侧单击作业名称。

**步骤3** 在“节点”页签下，可以查看到已经纳管的节点。

图 4-88 查看批量纳管的节点



----结束

## 4.6.2 批量节点升级

### 使用场景

在某些场景下，您可能需要对大量边缘节点的EdgeCore软件进行管理、更新维护，IEF提供批量节点升级功能，即批量升级边缘节点。

#### 📖 说明

边缘节点需要与IEF正常通信才可以升级成功。

## 注意事项

- 为了让您的边缘节点应用更稳定可靠的运行，IEF不会主动升级您的边缘节点上的EdgeCore，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。
- 处于维护周期中的版本升级，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。
- 处于维护周期外的版本升级，可能会因为容器重启引起业务的短暂中断。
- 请勿在节点升级过程中变更节点配置，比如重启Docker、安装卸载GPU/NPU驱动、OS内核升级、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

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“批量管理 > 节点升级”，单击页面右上角的“批量节点升级”。

**步骤3** 配置批量节点升级作业基本信息。

图 4-89 批量节点升级

The screenshot shows a web form for configuring a batch node upgrade. It is divided into two main sections: 'Basic Information' and 'Upgrade Configuration'.  
In the 'Basic Information' section:  
- There is a required field for 'Name' (名称) with a red asterisk and a placeholder '请输入作业名称'.  
- There is a 'Description' (描述) field with a placeholder '选填, 请输入作业描述' and a character count '0/255' at the bottom right.  
- There are two 'Tags' (标签) fields: one for the tag name (placeholder '请输入标签名') and one for the tag value (placeholder '请输入标签值').  
The 'Upgrade Configuration' section (升级配置) is separated by a horizontal line and contains:  
- An 'Upgrade Object' (升级对象) field with a placeholder '选择边缘节点'.

- **名称**: 创建的节点升级作业名称。
- **描述**: 节点升级作业描述。
- **标签**: 节点升级作业的标签

- **升级对象**：需要升级的节点。  
单击“选择边缘节点”，选择需要升级的节点。  
您还可以单击页面右上角的“标签搜索”，输入标签名和标签值，单击“搜索”，筛选出指定标签的节点。然后勾选需要升级的边缘节点，单击“确定”。

**步骤4** 单击“下一步”，确认升级信息，确认无误后单击“创建”。

----结束

## 状态说明

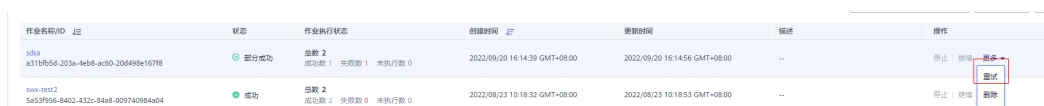
批量节点升级作业有以下八种状态。

- **排队中**：作业等待执行
- **执行中**：作业处于执行状态
- **成功**：全部任务执行成功
- **部分成功**：部分任务执行成功
- **失败**：全部任务执行失败
- **停止中**：作业处于停止中
- **已停止**：作业已停止
- **更新超时**：作业排队和执行时间超过10分钟仍未完成

批量作业执行过程中可以停止，停止后可以继续。

如果批量作业执行失败、部分成功或更新超时，可以重试执行作业，将未执行成功的作业再次执行一遍。

图 4-90 重试



作业名称/ID	状态	作业执行状态	创建时间	更新时间	描述	操作
nginx a31bf54-203a-4e8b-ac60-206498e167f8	部分成功	总次数 2 成功数 1 失败数 1 未执行数 0	2022/09/20 16:14:39 GMT+08:00	2022/09/20 16:14:56 GMT+08:00	--	停止 详情 重试
nginx 54c3f956-8402-432c-84a8-009740984a04	成功	总次数 2 成功数 2 失败数 0 未执行数 0	2022/08/23 10:18:32 GMT+08:00	2022/08/23 10:18:53 GMT+08:00	--	停止 详情 删除

## 4.6.3 批量应用部署

### 使用场景

在某些场景下，您可能需要对大量的边缘节点部署相同的应用，IEF提供批量应用部署功能。

#### 说明

- 节点架构必须全部是同一架构的节点，比如全部为x86或arm。
- 容器镜像的架构必须与节点架构一致，比如节点为x86，那容器镜像的架构也必须是x86。

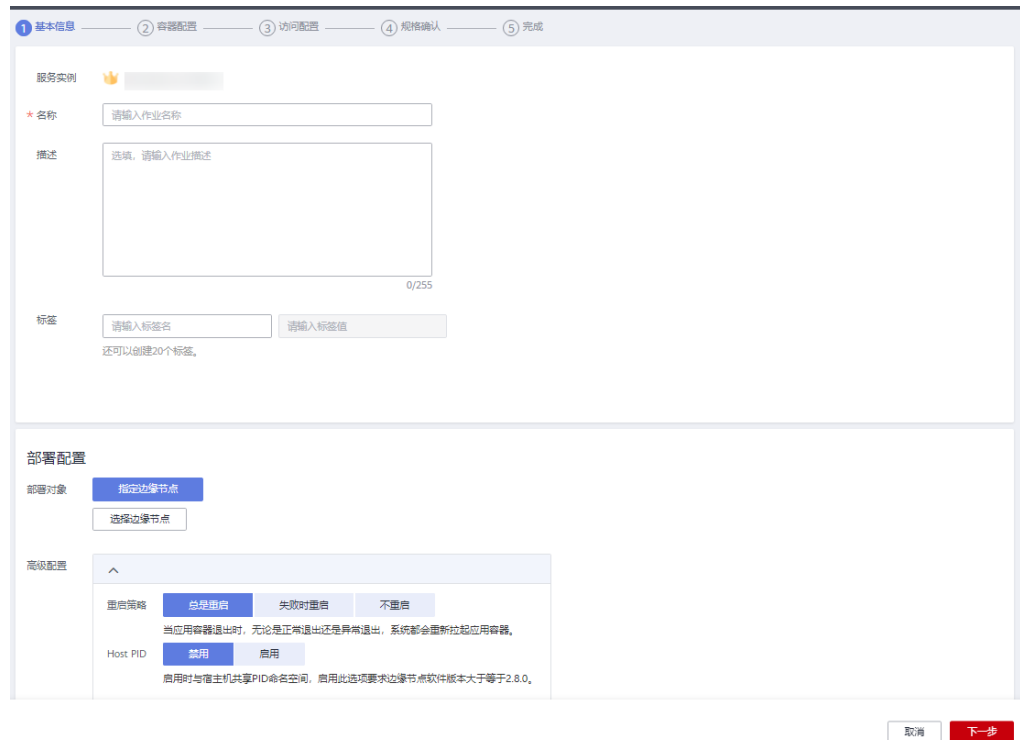
### 操作步骤

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“批量管理 > 应用部署”，单击页面右上角的“批量应用部署”。

**步骤3 配置应用部署作业基本信息。**

**图 4-91 创建批量应用部署作业**



- **名称：**创建的应用部署作业名称。
- **描述：**应用部署作业描述。
- **标签：**应用部署作业的标签。
- **部署对象：**需要部署容器应用的边缘节点。

单击“选择边缘节点”，勾选需要选择的边缘节点。

您还可以单击页面右上角的“标签搜索”，输入标签名和标签值，单击“搜索”，筛选出指定标签的边缘节点。然后勾选边缘节点，单击“确定”。

**图 4-92 选择标签**



- **重启策略**
  - 总是重启：当应用容器退出时，无论是正常退出还是异常退出，系统都会重新拉起应用容器。
  - 失败时重启：当应用容器异常退出时，系统会重新拉起应用容器，正常退出时，则不再拉起应用容器。
  - 不重启：当应用容器退出时，无论是正常退出还是异常退出，系统都不再重新拉起应用容器。
- **Host PID**：容器与宿主机共享PID命名空间，启用此选项要求边缘节点软件版本大于等于2.8.0。

**步骤4** 单击“下一步”，配置容器信息。

- **名称前缀**：应用部署的名称前缀。IEF会根据前缀生成完整的名称。
- **实例数量**：应用部署的实例数量。
- **配置方式**
  - 自定义配置：即从零开始配置容器应用，具体请参见**步骤5**。
  - 应用模板配置：选择一个已经定义好的应用模板，可以在模板的基础上进行修改，使用应用模板能够帮助您省去重复的工作量。模板的定义与**步骤5**需要的配置相同，创建模板的方法请参见**应用模板**。
- **部署描述**：输入容器应用描述。
- **标签**：容器应用标签，可勾选“继承边缘节点共性标签”。

**步骤5** 配置容器。

选择需要部署的镜像，单击“使用镜像”。

- **我的镜像**：展示了您在**容器镜像服务**中创建的所有镜像。
- **他人共享**：展示了其他用户共享的镜像，共享镜像是在SWR中操作的，具体请参见**共享私有镜像**。

选择镜像后，您可以配置容器的规格。

- **镜像版本**：请选择需要部署的镜像版本。

#### 须知

在生产环境中部署容器时，应避免使用latest版本。因为这会导致难以确定正在运行的镜像版本，并且难以正确回滚。

- **容器规格**：据需要选择容器CPU、内存、昇腾AI加速卡和GPU的配额。
- **昇腾AI加速卡**：容器应用选择的AI加速卡配置与实际部署的边缘节点配置的AI加速卡必须一致，否则会创建应用失败，详见**注册边缘节点**。

#### 说明

虚拟化切分后的NPU类型，一个容器只能挂载一个虚拟化NPU，只有当该容器退出后，该虚拟化NPU才能分配给其他容器使用。

昇腾AI加速卡支持的NPU类型，如下表。

表 4-14 NPU 类型说明

类型	描述
昇腾310	昇腾310芯片
昇腾310B	昇腾310B芯片

图 4-93 容器配置

The screenshot shows a container configuration interface. At the top, the image name is 'nginx' with a '更换镜像' (Change Image) link. Below it, the image version is 'v3' and the container name is 'container-66d8f335'. The '容器规格' (Container Specification) section includes CPU and memory settings. For CPU, '申请' (Request) and '限制' (Limit) are both set to 0.25 Core. For memory, '申请' and '限制' are both set to 512.00 MiB. At the bottom, there are options for 'AI加速卡' (AI Accelerator Card), with '不启用' (Not Enabled) selected, and '昇腾AI加速卡' (Ascend AI Accelerator Card) and 'Nvidia GPU' as alternatives.

您还可以对容器进行如下高级配置。

- **运行命令**

容器镜像拥有存储镜像信息的相关元数据，如果不设置生命周期命令和参数，容器运行时运行镜像制作时提供的默认的命令和参数，Dockerfile这两个字段为“Entrypoint”和“CMD”。

如果在创建容器应用时填写了容器的运行命令和参数，将会覆盖镜像构建时的默认命令“Entrypoint”、“CMD”，规则如下：

表 4-15 容器如何执行命令和参数

镜像 Entrypoint	镜像CMD	容器运行命令	容器运行参数	最终执行
[touch]	[/root/test]	未设置	未设置	[touch /root/test]
[touch]	[/root/test]	[mkdir]	未设置	[mkdir]
[touch]	[/root/test]	未设置	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

图 4-94 运行命令



#### - 运行命令

输入可执行的命令，例如`/run/start`。

若可执行命令有多个，多个命令之间用空格进行分隔。若命令本身带空格，则需要加引号（`"`）。

#### 📖 说明

多命令时，运行命令建议用`/bin/sh`或其他shell，其他全部命令作为参数来传入。

#### - 运行参数

输入控制容器运行命令的参数，例如`--port=8080`。

若参数有多个，多个参数以换行分隔。

### • 安全选项

- 可以通过开启“特权选项”，使容器拥有root权限，可以访问主机上的设备（如GPU、FPGA）。

- 指定运行用户

IEF默认不改变容器运行的用户，以构建镜像时定义的运行用户来运行。

打开开关后，下方出现运行用户的输入框，可输入范围为0~65534的整数。

指定了运行用户后，应用将以该运行用户来运行。如果镜像的操作系统中没有该用户ID，将导致容器应用启动失败。

### • 环境变量配置

容器运行环境中设定的变量。可以在应用部署后修改，为应用提供极大的灵活性。当前支持手动添加、密钥导入、配置项导入和变量引用方式。

- 手动添加支持自定义变量名称和变量值。

- 使用密钥导入，环境变量名称可自定义输入，环境变量值支持引用密钥的属性值，密钥创建方法请参见[密钥](#)。

- 使用配置项导入，环境变量名称可自定义输入，环境变量值支持引用配置项的属性值，配置项创建方法请参见[配置项](#)。

- 变量引用支持引用“hostIP”，即边缘节点的IP地址。

#### 📖 说明

IEF不会对用户输入的环境变量进行加密。如果用户配置的环境变量涉及敏感信息，用户需要自行加密后再填入，并在应用中自己完成解密过程。

IEF也不提供任何加解密工具，如果您需要设置加密密文，可以使用其他加解密工具。

### • 数据存储

您可以通过定义本地卷，将边缘节点本地存储目录挂载到容器中，以实现数据文件的持久化存储。



当前支持如下四种类型的本地卷。

- **hostPath**: 将主机某个目录挂载到容器中。hostPath是一种持久化存储，应用删除后hostPath里面的内容依然存在于边缘节点本地硬盘目录中，如果后续重新创建应用，挂载后依然可以读取到之前写入的内容。

您可以将应用日志目录挂载到主机的“/var/IEF/app/log/{appName}”目录，“{appName}”是应用名，边缘节点会将“/var/IEF/app/log/{appName}”目录下后缀为log和trace的文件上传到AOM。

图 4-95 日志卷挂载

本地卷名称	类型	挂载目录	权限	操作
log	hostPath	/var/IEF/app/log/nginx	/var/log/nginx	读写 删除

- **emptyDir**: 一种简单的空目录，主要用于临时存储，支持在硬盘或内存中创建。emptyDir挂载后就是一个空目录，应用程序可以在里面读写文件，emptyDir的生命周期与应用相同，应用删除后emptyDir的数据也同时删除掉。
- **configMap**: 存储应用所需配置信息的资源类型，创建方法请参见[配置项](#)。
- **secret**: 密钥是一种用于存储应用所需的认证信息、证书、密钥等敏感信息的资源类型，创建方法请参见[密钥](#)。

#### 须知

- 请不要挂载在系统目录下，如“/”、“/var/run”等，会导致容器异常。建议挂载在空目录下，若目录不为空，请确保目录下无影响容器启动的文件，否则文件会被替换，导致容器启动异常，应用创建失败。
- 挂载高危目录的情况下，建议使用低权限账号启动，否则可能会造成宿主机高危文件被破坏。

#### 健康检查

健康检查是指容器运行过程中根据用户需要定时检查容器健康状况或是容器中负载的健康状况。

- **应用存活探针**: 应用存活探针用于探测容器是否正常工作，不正常则重启实例。当前支持发送HTTP请求和执行命令检查，检测容器响应是否正常。
- **应用业务探针**: 应用业务探针用于探测业务是否就绪，如果业务还未就绪，就不会将流量转发到当前实例。

详细的配置说明请参见[健康检查配置说明](#)。

**步骤6** 单击“下一步”，进行访问配置。

容器访问支持主机网络和端口映射两种方式。

#### 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。

#### 说明

端口映射中的主机端口选择自动分配时，请输入合适且充足的端口范围，避免端口冲突。

端口映射可以选择主机网卡，请注意端口映射不支持选择IPv6地址类型的网卡。

- **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。

**步骤7** 单击“下一步”，确认容器应用的规格，确认无误后单击“创建”。

----结束

## 状态说明

批量应用部署作业有以下八种状态。

- **排队中**：作业等待执行
- **执行中**：作业处于执行状态
- **成功**：全部任务执行成功
- **部分成功**：部分任务执行成功
- **失败**：全部任务执行失败
- **停止中**：作业处于停止中
- **已停止**：作业已停止
- **更新超时**：作业排队和执行时间超过10分钟仍未完成

批量作业执行过程中可以停止，停止后可以继续。

如果批量作业执行失败、部分成功或更新超时，可以重试执行作业，将未执行成功的作业再次执行一遍。

## 4.6.4 批量应用升级

### 📖 说明

容器镜像的架构必须与节点架构一致，比如节点为x86，那容器镜像的架构也必须是x86。

**步骤1** 登录IEF管理控制台，在“总览”页面切换实例为铂金版。

**步骤2** 选择左侧导航栏的“批量管理 > 应用升级”，单击页面右上角的“批量应用升级”。

**步骤3** 配置应用升级作业基本信息。

图 4-96 批量应用升级

The screenshot shows a configuration form for 'Batch Application Upgrade'. It contains the following elements:

- Name:** A text input field with the placeholder '请输入作业名称'.
- Description:** A text area with the placeholder '选项, 请输入作业描述' and a character count '0/255'.
- Tags:** Two text input fields with placeholders '请输入标签名' and '请输入标签值'. Below them is the text '还可以创建20个标签.'
- Note:** A blue box with a note icon and text: '注意: 以下选中的应用升级将会采用同一套配置进行升级'.
- Upgrade Object:** A dropdown menu with the placeholder '请选择应用部署'.

- **名称**: 创建的应用升级作业名称。
- **描述**: 应用升级作业描述。
- **标签**: 应用升级作业的标签。
- **升级对象**: 需要升级的容器应用。

单击“请选择应用部署”，选择需要升级的容器应用。

您还可以单击页面右上角的“标签搜索”，输入标签名和标签值，单击“搜索”，筛选出指定标签的容器应用。然后勾选需要升级的容器应用，单击“确定”。

**步骤4** 单击“下一步”，配置容器，此处的配置与**步骤5**相同。

**步骤5** 单击“下一步”，进行访问配置，此处的配置与**步骤6**相同。

**步骤6** 单击“下一步”，确认容器应用的规格，确认无误后单击“创建”。

----结束

## 状态说明

批量应用升级作业有以下八种状态。

- **排队中**: 作业等待执行
- **执行中**: 作业处于执行状态
- **成功**: 全部任务执行成功
- **部分成功**: 部分任务执行成功
- **失败**: 全部任务执行失败
- **停止中**: 作业处于停止中
- **已停止**: 作业已停止
- **更新超时**: 作业排队和执行时间超过10分钟仍未完成

批量作业执行过程中可以停止，停止后可以继续。

如果批量作业执行失败、部分成功或更新超时，可以重试执行作业，将未执行成功的作业再次执行一遍。

## 4.7 审计

### 4.7.1 支持云审计的关键操作

#### 操作场景

云审计服务（Cloud Trace Service, CTS），是华为云安全解决方案中专业的日志审计服务，提供对各种云资源操作记录的收集、存储和查询功能，可用于支撑安全分析、合规审计、资源跟踪和问题定位等常见应用场景。

通过云审计服务，您可以记录与IEF相关的操作事件，便于日后的查询、审计和回溯。

## 支持审计的关键操作列表

表 4-16 云审计服务支持的 IEF 操作列表

操作名称	资源类型	事件名称
注册边缘节点	node	createEdgeNode
更新边缘节点	node	updateEdgeNode
删除边缘节点	node	deleteEdgeNode
创建边缘节点组	group	createEdgeGroup
更新边缘节点组	group	updateEdgeGroup
删除边缘节点组	group	deleteEdgeGroup
创建设备模板	deviceTemplate	createDeviceTemplate
更新设备模板	deviceTemplate	updateDeviceTemplate
删除设备模板	deviceTemplate	deleteDeviceTemplate
注册设备	device	createDevice
更新设备	device	updateDevice
删除设备	device	deleteDevice
部署Edge-Connector	device	deployConnector
更新设备孪生	device	updateDeviceTwin
更新设备访问配置	device	updateDeviceAccessConfig
创建应用模板	application	createApplication
更新应用模板	application	updateApplication
删除应用模板	application	deleteApplication
创建应用模板版本	appVersion	createAppVersion
更新应用模板版本	appVersion	updateAppVersion
删除应用模板版本	appVersion	deleteAppVersion
创建配置项	configmap	createConfigMap
更新配置项	configmap	updateConfigMap
删除配置项	configmap	deleteConfigMap
创建容器应用	deployment	createDeployment
更新容器应用	deployment	updateDeployment
删除容器应用	deployment	deleteDeployment

操作名称	资源类型	事件名称
查询容器应用列表	deployment	getDeploymentList
查询容器应用	deployment	getDeployment
创建端点	endpoint	createEndpoint
删除端点	endpoint	deleteEndpoint
添加节点证书	node	AddNodeCert
删除节点证书	node	deleteNodeCert
升级固件	nodeFirmware	UpgradeNodeFirmware
查询应用实例列表	Pods	getPods
查询实例	Pod	getPod
创建节点注册作业	product	createProduct
删除节点注册作业	product	deleteProduct
创建节点升级作业/创建应用部署作业/创建应用升级作业	batchJob	createJob
停止节点升级作业/停止应用部署作业/停止应用升级作业	batchJob	pauseJob
删除节点升级作业/删除应用部署作业/删除应用升级作业	batchJob	deleteJob
恢复节点升级作业/恢复应用部署作业/恢复应用升级作业	batchJob	restoreJob
重试节点升级作业/重试应用部署作业/重试应用升级作业	batchJob	retryJob
查询配额	quota	getQuota
更新配额	quota	updateQuota
创建规则	rule	createRule
删除规则	rule	deleteRule
更新规则	rule	updateRule
创建密钥	secret	createSecret
更新密钥	secret	updateSecret
删除密钥	secret	deleteSecret
过滤标签	Tags	filterTags
批量添加删除标签	Tags	batchAddDeleteTags
添加标签	Tags	addTag

操作名称	资源类型	事件名称
删除标签	Tags	deleteTag
绑定加密数据	encryptdata	bindEncryptdata
创建加密数据	encryptdata	createEncryptData
删除加密数据	encryptdata	deleteEncryptData
解绑加密数据	encryptdata	unbindEncryptdata
更新加密数据	encryptdata	updateEncryptData
创建网关	gateway	createGateway
创建虚拟服务	gateway	createVirtualService
删除网关	gateway	deleteGateway
删除虚拟服务	gateway	deleteVirtualService
更新网关	gateway	updateGateway
更新虚拟服务	gateway	updateVirtualService
创建系统订阅	Systemevent	createSystemevent
删除系统订阅	Systemevent	DeleteSystemevent
启用系统订阅	Systemevent	startSystemevent
停用系统订阅	Systemevent	stopSystemevent

## 4.7.2 如何查看审计日志

### 操作场景

开启了云审计服务（CTS）后，系统开始记录IEF相关的操作。CTS会保存最近1周的操作记录。

本小节介绍如何在CTS管理控制台查看最近1周的操作记录。

### 操作步骤

**步骤1** 登录CTS管理控制台。


**步骤2** 选择左侧导航栏的“事件列表”，进入事件列表页面。

**步骤3** 事件记录了云资源的操作详情，设置筛选条件，单击“查询”。

当前事件列表支持四个维度的组合查询，详细信息如下：

- 事件类型、事件来源、资源类型和筛选类型。  
在下拉框中选择查询条件。其中，“事件类型”选择“管理事件”，“事件来源”选择“IEF”。  
其中，

- 筛选类型选择“按资源ID”时，还需手动输入某个具体的资源ID，目前仅支持全字匹配模式的查询。
- 筛选类型选择“按资源名称”时，还需选择或手动输入某个具体的资源名称。
- 操作用户：在下拉框中选择某一具体的操作用户。
- 事件级别：可选项为“所有事件级别”、“Normal”、“Warning”、“Incident”，只可选择其中一项。
- 时间范围：可选项为“最近1小时”、“最近1天”、“最近1周”和“自定义时间段”，本示例选择“最近1周”。

**步骤4** 在需要查看的事件左侧，单击  图标展开该事件的详细信息。

**步骤5** 在需要查看的事件右侧，单击“查看事件”，弹出一个窗口，显示了该操作事件结构的详细信息。

----结束

## 4.8 权限管理

### 4.8.1 创建用户并授权使用 IEF

如果您需要对您所拥有的智能边缘平台（IEF）进行精细的权限管理，您可以使用[统一身份认证服务](#)（Identity and Access Management，简称IAM），通过IAM，您可以：

将IEF资源委托给更专业、高效的其他云账号或者云服务，这些账号或者云服务可以根据权限进行代运维。

如果云账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用IEF服务的其它功能。

本章节为您介绍对用户授权的方法，操作流程如[图4-97](#)所示。

#### 说明

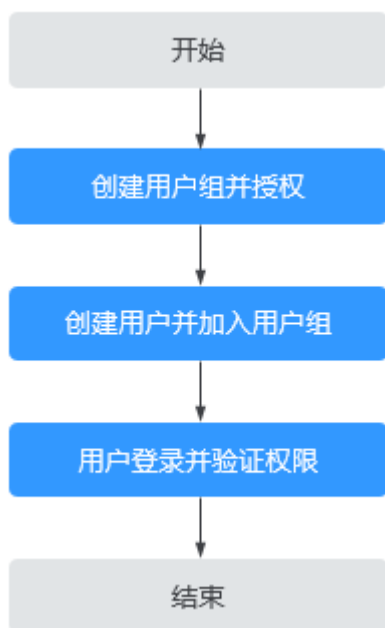
当前 IEF 只支持操作级权限管理，不支持资源和请求条件。

### 前提条件

给用户组授权之前，请您了解用户组可以添加的IEF权限，并结合实际需求进行选择，IEF支持的系统策略，请参见。若您需要对除IEF之外的其它服务授权，IAM支持服务的所有策略请参见。

## 示例流程

图 4-97 给用户授权 IEF 权限流程



### 1. 创建用户组并授权。

在IAM控制台创建用户组，并授予IEF只读权限“IEF ReadOnlyAccess”。为用户组授权时，作用范围选择“区域级项目”，然后根据以下原则设置：

- 在个别区域授权：选择指定的一个或多个项目，例如“cn-north-4 [华北-北京四]”。注意：此场景选择“所有项目”时，授权将不生效。
- 在所有区域授权：选择“所有项目”。

图 4-98 在个别区域授权



图 4-99 在所有区域授权



### 2. 创建用户并加入用户组。

在IAM控制台创建用户，并将其加入**1.创建用户组并授权**中创建的用户组。

### 3. 用户登录并验证权限。

新创建的用户登录控制台，切换至授权区域，验证权限：



- 在“服务列表”中选择智能边缘平台，进入IEF主界面，选择左侧导航栏的“边缘资源 > 边缘节点”，单击页面右上角的“注册边缘节点”，如果无法注册边缘节点，表示“IEF ReadOnlyAccess”已生效。
- 在“服务列表”中选择除智能边缘平台外（假设当前策略仅包含IEF ReadOnlyAccess）的任一服务，若提示权限不足，表示“IEF ReadOnlyAccess”已生效。

## 4.8.2 自定义策略

IEF可以创建自定义策略。

目前华为云支持以下两种方式创建自定义策略：

- 可视化视图创建自定义策略：无需了解策略语法，按可视化视图导航栏选择云服务、操作、资源、条件等策略内容，可自动生成策略。
- JSON视图创建自定义策略：可以在选择策略模板后，根据具体需求编辑策略内容；也可以直接在编辑框内编写JSON格式的策略内容。

具体创建步骤请参见：[创建自定义策略](#)。本章为您介绍常用的IEF自定义策略样例。

### IEF 自定义策略样例

授权用户创建、更新应用和应用模板的权限。

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ief:deployment:create",
        "ief:appVersion:update",
        "ief:deployment:update",
        "ief:application:create"
      ],
      "Condition": {
        "StringEquals": {
          "ief:AssumeUserName": [
            "test"
          ]
        }
      }
    }
  ],
  "Resource": [
    "ief:*:deployment:*",
    "ief:*:appVersion:*",
    "ief:*:application:*"
  ]
}
```

## 4.8.3 系统委托说明

由于IEF在运行中对镜像、存储、数据以及监控等各类云服务资源都存在依赖关系，因此当您首次登录IEF控制台时，IEF将自动请求获取当前区域下的云资源权限，从而更好地为您提供服务。服务权限包括：

- AOM服务  
IEF支持通过应用运维管理（AOM）收集边缘节点和应用容器的性能指标、日志（用户可选），帮助用户实时监控边缘节点和应用的性能、快速定位可能出现的问题，因此需要获得访问应用运维管理服务的权限

- SWR服务  
IEF 支持通过容器镜像服务（SWR）管理并下载用户自定义容器镜像，帮助用户在边缘节点上部署容器应用，因此需要获得访问容器镜像服务的权限
- OBS服务  
IEF 支持通过对象存储服务（OBS）访问您创建的边缘函数，帮助用户在边缘节点、设备上部署并运行函数，因此需要获得访问对象存储服务的权限
- DIS服务  
IEF 支持通过将边缘节点、设备上的数据发送到用户的数据接入服务（DIS）的通道中，因此需要获得访问数据接入服务的权限。

当您同意授权后，IEF将在IAM中自动创建账号委托，将账号内的其他资源操作权限委托给华为云IEF服务进行操作。关于资源委托详情，您可参考[委托](#)进行了解。

IEF自动创建的委托如下：

- [ief\\_admin\\_trust委托说明](#)
- [ief\\_edge\\_trust委托说明](#)

## ief\_admin\_trust 委托说明

ief\_admin\_trust委托具有Tenant Administrator权限。Tenant Administrator拥有除IAM管理外的全部云服务管理员权限，用于对IEF所依赖的其他云服务资源进行调用，且该授权仅在当前区域生效。

如果您在多个区域中使用IEF服务，则需在每个区域中分别申请云资源权限。您可前往“IAM控制台 > 委托”页签，单击“ief\_admin\_trust”查看各区域的授权记录。

### 说明

由于IEF对其他云服务有许多依赖，如果没有Tenant Administrator权限，可能会因为某个服务权限不足而影响IEF功能的正常使用。因此在使用IEF服务期间，请不要自行删除或者修改“ief\_admin\_trust”委托。

## ief\_edge\_trust 委托说明

ief\_edge\_trust委托没有Tenant Administrator系统角色，只包含IEF需要的云服务资源操作权限，用于边缘节点上报监控、告警、日志。

若当前ief\_edge\_trust委托的权限与IEF期望的权限不同时，控制台会提示权限变化，需要您重新授权。

以下场景中，可能会出现ief\_edge\_trust委托重新授权：

- IEF组件依赖的权限可能会随版本变动而发生变化。例如新增组件需要依赖新的权限，IEF将会更新期望的权限列表，此时需要您重新为ief\_edge\_trust委托授权。
- 当您手动修改了ief\_edge\_trust委托的权限时，该委托中拥有的权限与IEF期望的权限不相同，此时也会出现重新授权的提示。若您重新进行授权，该委托中手动修改的权限可能会失效。