**1.docker**

**云端Ubuntu:**

**1.kubelete**

**2.cloudcore**

**3.kuboard**

**安装教程：**[**https://www.kuboard.cn/install/v3/install-in-k8s.html**](https://www.kuboard.cn/install/v3/install-in-k8s.html)

**访问地址（同一局域网下）：**[**http://192.168.31.100:30080/**](http://192.168.31.100:30080/)

**用户名：admin**

**密码：Kuboard123**

**边缘端Ubuntu:**

**1.edgecore**

**2. EdgeXFoundry**

**3. edgex Dashboard**

**访问地址（同一局域网下）：http://192.168.31.101:4000/zh/#/dashboard**

[**EdgeX Console**](http://192.168.31.101:4000/zh/#/dashboard)

**DS18B20温度传感器：**

**参考：**

[**树莓派4B+ubuntu20.04读取ds18b20温度传感器数据-CSDN博客**](https://blog.csdn.net/qq_25231683/article/details/120018287)[**树莓派4B-Python-控制DS18B20（温度传感器）\_micro python操作ds18b20-CSDN博客**](https://blog.csdn.net/qq_46476163/article/details/116534840)

[**https://www.bilibili.com/video/BV1sE411u7vd**](https://www.bilibili.com/video/BV1sE411u7vd)

[**EdgeX Foundry--MQTT设备服务（一） - 简书 (jianshu.com)**](https://www.jianshu.com/p/ffb0822fb755)

[**EdgeX Foundry--MQTT设备服务（二） - 简书 (jianshu.com)**](https://www.jianshu.com/p/af515094244b)

[**MQTT - EdgeX Foundry 文档 --- MQTT - EdgeX Foundry Documentation**](https://docs.edgexfoundry.org/3.0/examples/Ch-ExamplesAddingMQTTDevice/)

**1)改内存卡下的usercfg.txt文件**

**添加如下内容：**

**#ds18b20**

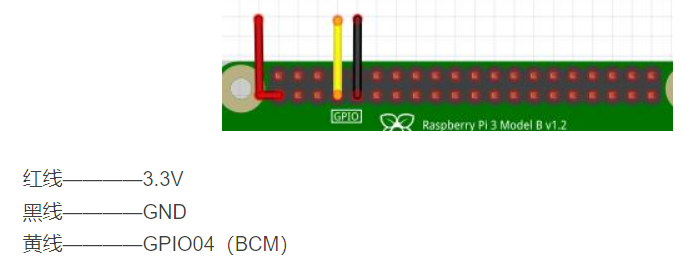
**dtoverlay=w1-gpio-pullup,gpiopin=4**

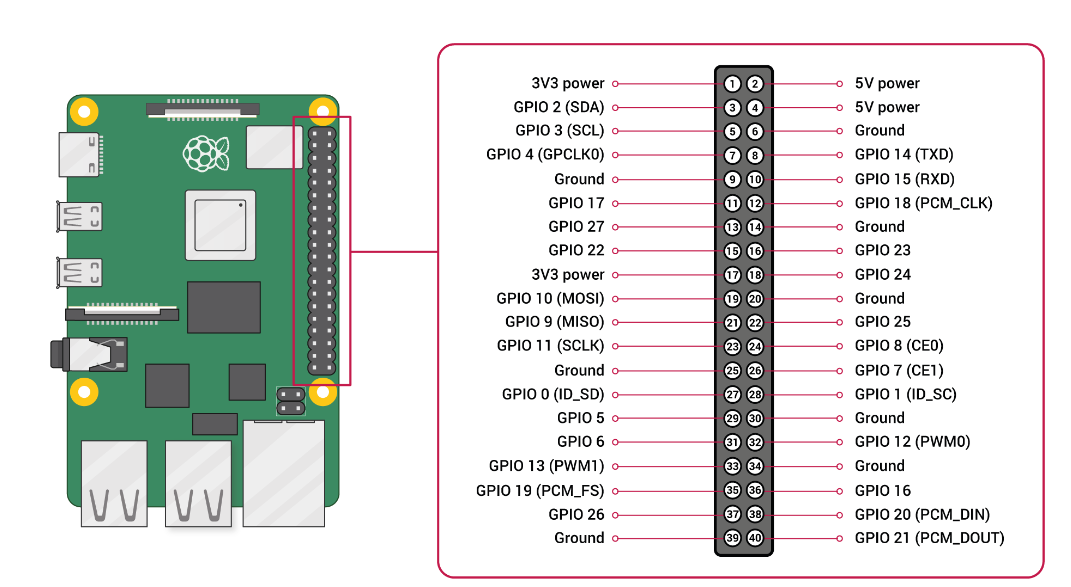
**2)设备连接**

**红线————3.3V**

**黑线————GND**

**黄线————GPIO04（BCM）**

****

****

**3) 检验设备连接**

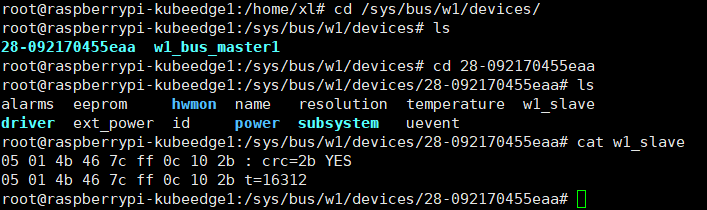
**cd /sys/bus/w1/devices/**

**1s**

**一般显示的地址为 28-xxxxxxxxxxxx**

**cd 28-092170455eaa**

**cat w1\_slave**

****

**最后的 “ t= 16312 ” 为当前的温度（摄氏度） X 1000 后的结果，所以除以1000才是真正的温度（16.31℃）。**

**4) python代码**

**终端运行python程序（无参数）：**

[**mock-device-driver/mock-device-for-mqtt.py at master · SHILIANG17671467654/mock-device-driver · GitHub**](https://github.com/SHILIANG17671467654/mock-device-driver/blob/master/mock-device-for-mqtt.py)**（数据上传、接受命令）**

**python3 ds18b20\_mqtt.py**

**# 07.树莓派python读取温度传感器DS18B20数据**

**#!/usr/bin/python3**

**import os**

**import time**

**# import RPi.GPIO as GPIO**

**# import time**

**import json**

**import random**

**from paho.mqtt import client as mqtt\_client**

**# DS18B20 设备文件地址**

**device\_file ='/sys/bus/w1/devices/28-092170455eaa/w1\_slave'**

**# 边缘端地址**

**broker = '192.168.31.101'**

**port = 1883**

**'''**

**# kubeedge 相关mqtt主题**

**#topic = "$hw/events/device/temperature/twin/update"**

**#topic0 = "DataTopic"**

**#topic0 = "incoming/data/humidity\_temperature/humidity"**

**#topic1 = "incoming/data/humidity\_temperature/humidity"**

**'''**

**# edgex中温度设备订阅的mqtt主题**

**topic2 = "incoming/data/temperature\_device\_mqtt/temperature" # ?????**

**temperature = 0**

**humidity = 0**

**# 随机生成 client ID**

**client\_id = f'python-mqtt-{random.randint(0, 1000)}'**

**# cat 文件**

**def read\_temp\_raw():**

**f = open(device\_file,'r')**

**lines = f.readlines()**

**f.close()**

**return lines**

**# 从文件中获得温度数据**

**def read\_temp():**

**lines = read\_temp\_raw()**

**while lines[0].strip()[-3:] != 'YES':**

**# print('--------------Eeeor----------------')**

**time.sleep(0.2)**

**lines = read\_temp\_raw()**

**# print('--------------Success----------------')**

**equals\_pos = lines[1].find('t=')**

**if equals\_pos != -1:**

**temp\_string = lines[1][equals\_pos+2:]**

**temp\_c = float(temp\_string)/1000.0**

**return temp\_c**

**# 连接 mqtt broker**

**def connect\_mqtt():**

**def on\_connect(client, userdata, flags, rc):**

**if rc == 0:**

**print("Connected to MQTT Broker!")**

**else:**

**print("Failed to connect, return code %d\n", rc)**

**client = mqtt\_client.Client(client\_id)**

**client.on\_connect = on\_connect**

**client.connect(broker, port)**

**return client**

**# mqtt pub消息**

**def publish(client, msg, topic):**

**print("发送的消息:",msg)**

**result = client.publish(topic, msg)**

**# result: [0, 1]**

**status = result[0]**

**if status == 0:**

**print(f"Send `{msg}` to topic `{topic}`")**

**else:**

**print(f"Failed to send message to topic {topic}") # \*\*\*\*\*\***

**# msg\_count += 1**

**'''**

**# 测试温度数据读取**

**if \_\_name\_\_=="\_\_main\_\_"：**

**while True:**

**print('temp C = %f'%read\_temp())**

**time.sleep(1)**

**'''**

**# 连接mqtt broker，循环调用**

**client = connect\_mqtt()**

**client.loop\_start()**

**# 向 edgex 发送数据**

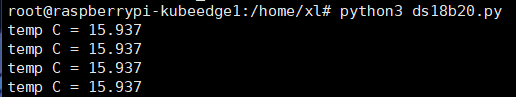
**while True:**

**time.sleep(1) # f = 1hz**

**temperature = round(read\_temp(), 3)**

**if(temperature !=0):**

**publish(client, temperature, topic2)**

****

**生成自己需要的edgex服务：**

[**温度传感器->Edgex->KubeEdge - SiYuan - 思源笔记 v2.10.14 (goho.co)**](http://58091ab615.goho.co/stage/build/desktop/?r=pbacpkb)

[**Getting Started using Docker - EdgeX Foundry Documentation**](https://docs.edgexfoundry.org/3.0/getting-started/Ch-GettingStartedDockerUsers/)

**git clone https://github.com/edgexfoundry/edgex-compose.git**

**cd /home/xl/EdgeX/edgex-compose/compose-builder**

**git branch**

**git checkout Minnesota**

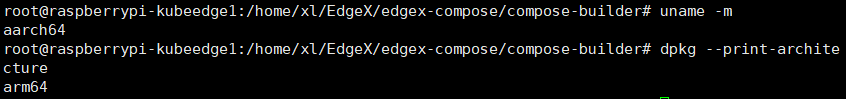
**make gen no-secty arm64 ds-mqtt mqtt-broker**

**查看系统架构：**

[**x86-64、amd64、arm、aarch64 都是些什么？-CSDN博客**](https://blog.csdn.net/qq_24433609/article/details/125991550)

[**linux如何查看系统架构？（查看系统架构命令）（armv7l）\_linux 3.4.39 cst 2019 armv7l gnu/linux-CSDN博客**](https://blog.csdn.net/Dontla/article/details/120625069)

**dpkg --print-architecture 或 uname -m**

****

**edgex添加设备（UI方式）：**

[**http://192.168.31.101:8500/ui/dc1/services**](http://192.168.31.101:8500/ui/dc1/services)

**需要重新启动edgex：**

**# 进入compose-bulider 目录**

**cd edgex-compose/compose-builder**

**# 启动edgex**

**docker compose up -d**

**树莓派备份：**

**方式一：全盘备份**

**sudo apt install dc3dd**

**sudo dc3dd if=/dev/mmcblk0 of=/dev/sda**

**方式二：压缩备份**

**# 教程：**[**https://www.bilibili.com/video/BV1B64y1d78p/**](https://www.bilibili.com/video/BV1B64y1d78p/)

**#** [**https://www.bilibili.com/video/BV14P411j7x6/**](https://www.bilibili.com/video/BV14P411j7x6/)

[**树莓派系统备份及还原 - 水水滴答 - 博客园 (cnblogs.com)**](https://www.cnblogs.com/havain/p/15785046.html#:~:text=1%E3%80%81%E5%AE%89%E8%A3%85gparted%EF%BC%8C%E6%89%93%E5%BC%80%E8%99%9A%E6%8B%9F%E6%9C%BA%EF%BC%8C%E5%9C%A8%E7%BB%88%E7%AB%AF%E8%BE%93%E5%85%A5%20sudo%20apt%20install%20gparted,2%E3%80%81%E7%B3%BB%E7%BB%9F%E5%8E%8B%E7%BC%A9%20%E5%9C%A8gparted%E8%BD%AF%E4%BB%B6%E4%B8%AD%EF%BC%8C%E5%8E%8B%E7%BC%A9SD%E5%8D%A1%E5%86%85%E5%AD%98%E8%87%B3%E6%A0%91%E8%8E%93%E6%B4%BE%E7%B3%BB%E7%BB%9F%E5%BD%93%E5%89%8D%E5%8D%A0%E7%94%A8%E7%9A%84%E5%A4%A7%E5%B0%8F%EF%BC%8C%E5%A6%82%E4%B8%8B%E5%9B%BE%20%E6%B3%A8%E6%84%8F%EF%BC%9ASD%E5%8D%A1%E6%8F%92%E4%B8%8A%E5%90%8E%EF%BC%8C%E8%A6%81%E5%85%88%E9%80%89%E5%AE%9A%E5%AF%B9%E5%BA%94%E7%9A%84%E7%9B%98%E7%AC%A6%EF%BC%8C%E6%9C%80%E5%BC%80%E5%A7%8Bwritable%E6%98%AF%E9%94%81%E5%AE%9A%E7%9A%84%EF%BC%8C%E9%9C%80%E8%A6%81%E5%85%88umount%E6%89%8D%E8%83%BD%E8%BF%9B%E8%A1%8C%E7%A9%BA%E9%97%B4%E5%8E%8B%E7%BC%A9%E3%80%82%203%E3%80%81%E7%B3%BB%E7%BB%9F%E5%A4%87%E4%BB%BD%20%E6%8C%89%E4%B8%8A%E8%BF%B0%E6%93%8D%E4%BD%9C%E5%AE%8C%E6%88%90%E5%90%8E%EF%BC%8C%E5%BC%B9%E5%87%BASD%E5%8D%A1%EF%BC%8C%E7%84%B6%E5%90%8E%E5%88%87%E6%8D%A2%E5%88%B0windows%E4%B8%8B%E8%BF%9B%E8%A1%8C%E5%A4%87%E4%BB%BD%E3%80%82)

**# SD A: 系统**

**# SD B: U盘**

**(1) U盘挂载**

**# Linux 上挂载和弹出U盘都和 Windows 不同，不能直接识别出来，需要手动挂载，拔出U盘也需要手动取消U盘挂载。**

[**树莓派无法识别U盘，详细步骤配置支持exFAT文件系统 - Labno3**](https://www.labno3.com/2021/08/14/raspberry-pi-exfat-adding-support-for-exfat-file-system/)

[**树莓派挂载U盘-CSDN博客**](https://blog.csdn.net/lengyuefeng212/article/details/114160824)

**sudo apt update**

**sudo apt-get install exfat-utils**

**sudo apt-get install exfat-fuse**

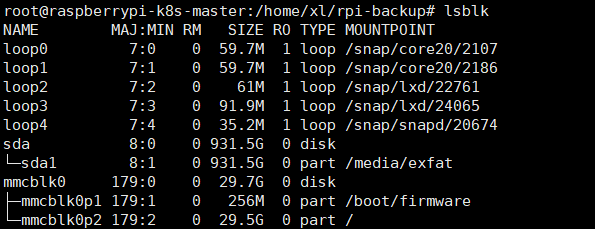
**sudo apt install dc3dd**

**mkdir /media/exfat**

**sudo mount -t exfat /dev/sda1 /media/exfat**

**#查看usb挂载在哪里**

**lsblk**

****

**# 查看硬盘使用**

**df -h**

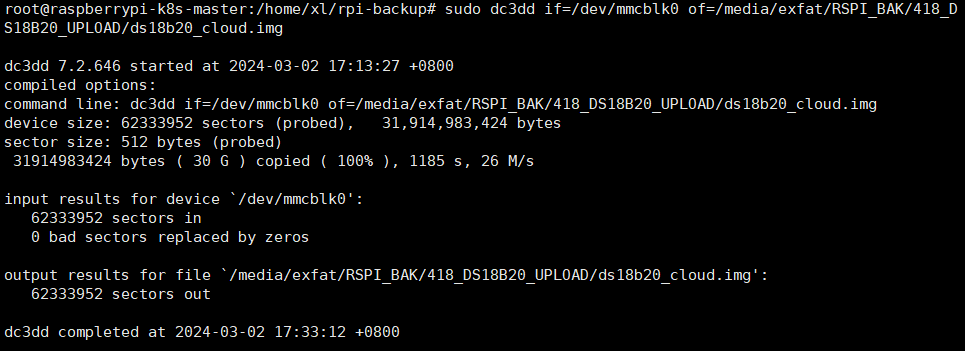
**# 查看运存使用**

**free -h**

**(2)备份成镜像**

**sudo dc3dd if=/dev/mmcblk0 of=【USB挂载路径】/myimg.img**

**# sudo dc3dd if=/dev/mmcblk0 of=/media/exfat/RSPI\_BAK/418\_DS18B20\_UPLOAD/ds18b20\_edge.img**

****

**(3)裁剪镜像**

**git clone https://github.com/Drewsif/PiShrink/**

**chmod +x PiShrink/pishrink.sh**

**sudo mv PiShrink/pishrink.sh /usr/local/bin**

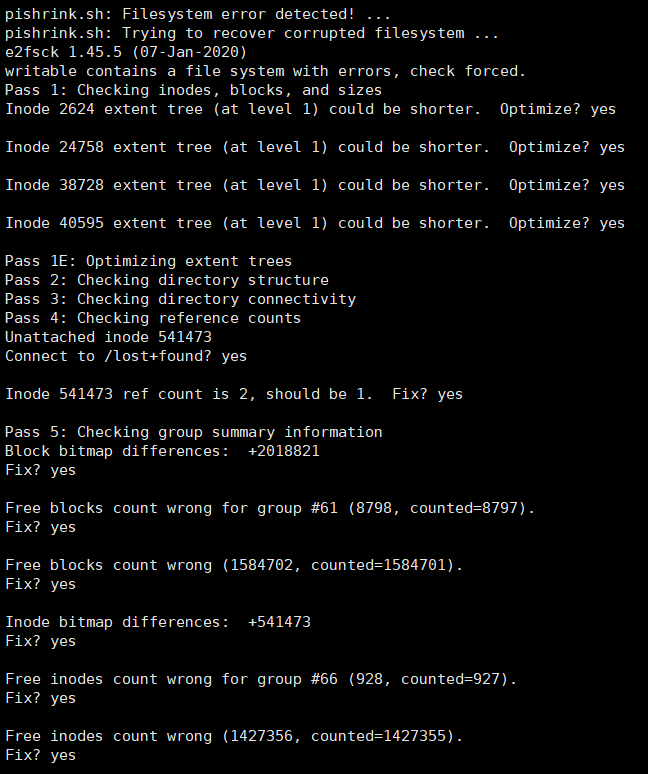
**sudo pishrink.sh -z 【USB挂载路径】/myimg.img**

**# sudo pishrink.sh -z /media/exfat/RSPI\_BAK/418\_DS18B20\_UPLOAD/ds18b20\_edge.img**

**(4)卸载挂载**

**sudo umount /media/exfat/**

**若直接拔下U盘，可能会出现以下问题：**

****

**遇到的问题：**

**虚拟机输入密码进不了桌面：**

sudo chown -R xl1 /home/xl1

sudo reboot

[Ubuntu 重复循环登录解决办法\_desk反复跳出-CSDN博客](https://blog.csdn.net/qq_52147555/article/details/133958222)

[如何修复 Ubuntu 登录循环 - 让技术更轻松 --- How to Fix the Ubuntu Login Loop - Make Tech Easier](https://www.maketecheasier.com/fix-ubuntu-login-loop/)

**go 环境变量设置：**

~/.bashrc //用户

/etc/profile //所有用户

source ~/.bashrc

[【Ubuntu】Ubuntu设置和查看环境变量\_ubuntu如何查看系统环境变量-CSDN博客](https://blog.csdn.net/white_idiot/article/details/78253004)

[Go 语言环境安装 | 菜鸟教程 (runoob.com)](https://www.runoob.com/go/go-environment.html)

**nodejs npm版本要求**： >16 >8

npm ERR! notsup Required: {"node":">=14.18.3 <17","npm":">=7.21.0 <9"}

npm ERR! notsup Actual: {"npm":"10.5.0","node":"v20.12.1"}

----------nodejs v16.19.1-----npm 8.19.3---------------

$ cd node

$ sudo ./configure

$ sudo make

$ sudo make install

1111 1101 1101 1001

0000 0010 0010 0110

**edge节点的pod csi-node-driver 一直creating**

calico-csi：docker.io/calico/csi:v3.25.2

csi-node-driver-registrar：docker.io/calico/node-driver-registrar:v3.25.2

**metrics-server镜像：**

registry.aliyuncs.com/google\_containers/metrics-server:v0.4.1

docker tag registry.aliyuncs.com/google\_containers/metrics-server:v0.4.0 k8s.gcr.io/metrics-server/metrics-server:v0.4.0

**update 错误：**

W: GPG 错误：http://mirrors.aliyun.com/ubuntu bionic InRelease: 由于没有公钥，无法验证下列签名： NO\_PUBKEY 3B4FE6ACC0B21F32

E: 仓库 “http://mirrors.aliyun.com/ubuntu bionic InRelease” 没有数字签名。

N: 无法安全地用该源进行更新，所以默认禁用该源。

N: 参见 apt-secure(8) 手册以了解仓库创建和用户配置方面的细节。

**key:**

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 40976EAF437D05B5

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 3B4FE6ACC0B21F32

解压：[Linux命令之解压缩：tar、zip、rar 命令\_linux解压命令-CSDN博客](https://blog.csdn.net/afei__/article/details/82619843)

**1. 加载节点 Metrics 信息失败**

**错误原因：Error: Request failed with status code 404 请检查 kube-system 名称空间下 metrics-server 的日志信息。**

**2.设备数据上传到云端失败（2月目标）**

**注意：使用kubeedge自带的MQTT服务器上传数据**

**错误：能收集到数据，但不能上传**

**3.** **MQTTBrokerInfo/Host 中改成边缘端IP有什么用？**

**就是MQTTBroker的IP地址，与docker-compose文件的服务配置相对应？？？**

**4. 1883端口、1884端口**

**1883端口一般基于无账号密码的服务。1884一般对应需要账号密码的加密服务。**

**5.edgex的数据流怎么获得python发出的数据？（在哪儿订阅的主题）**

**python数据发送的topic是datatopic（incoming/data/#），device-mqtt服务自动订阅**

**commantopic是edgex发送命令的topic**

****

**6. edgex的ui能读值------表明edgex收到了（设备）的数据？**

**7.kuiper没有输出**

**$hw/events/device/temperature/twin/update**

**GET状态：http://192.168.31.101:59720/rules/111/status**

**8.查日志**

**journalctl --unit edgecore.service -n 20**

[**【K8S系列】如何高效查看 k8s日志\_k8s查看pod日志-CSDN博客**](https://blog.csdn.net/weixin_36755535/article/details/131283024)

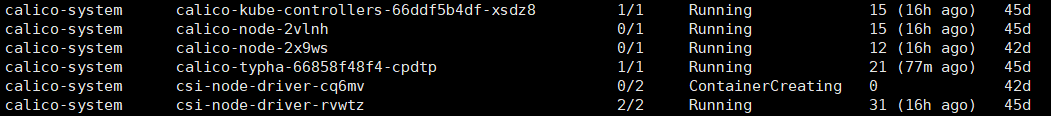
**ubuntu22无网络、无图标**

<https://blog.csdn.net/qq_36347513/article/details/126573588>

[1 node(s) had taint {node.kubernetes.io/disk-pressure: }, that the pod didn‘t tolerate\_0/126 nodes are available: 1 node(s) had taint ino-CSDN博客](https://blog.csdn.net/weixin_43114954/article/details/119155118)

**err:**

**1.calico网络节点pod运行失败**

****

**2.**

**UI for kubernetes**

**Kubectl：官方自带的kubernetes 命令行客户端工具，最原始也是最好用的工具，**

**Kuboard：基于 Kubernetes 的微服务管理界面。目的是帮助用户快速在 Kubernetes 上落地微服务。**

**Kubernetes-Dashboard：官方的WEB UI，可以通过Dashboard查看集群信息，对容器服务进行故障管理、资源管理等。例如，您可以使用部署向导来扩展部署，启动滚动更新，重新启动Pod或部署新应用程序。**

* **向 Kubernetes 集群部署容器化应用**
* **诊断容器化应用的问题**
* **管理集群的资源**
* **查看集群上所运行的应用程序**
* **创建、修改Kubernetes 上的资源（例如 Deployment、Job、DaemonSet等）**
* **展示集群上发生的错误**

**例如：您可以伸缩一个 Deployment、执行滚动更新、重启一个 Pod 或部署一个新的应用程序**

**Kuboard 的定位和 Dashboard 是相似的，**

**主要的区别 在于：**

* **Kuboard 关注微服务参考架构的视角对界面进行组织，参考** [**Kuboard 简介**](https://kuboard.cn/overview/share-coder.html)
* **Kuboard 中，不需要手工编写 YAML 文件，进一步降低 K8S 使用难度，提高便捷性**
* **Kuboard 可以导出整个微服务架构的部署信息，并在新的名称空间/集群导入配置信息**
* **Kuboard 的一个发展方向是，提供内建的** [**监控套件**](https://kuboard.cn/guide/example/monitor.html)**（目前的全局监控套件成熟度比较高）**

**kuboard比较官方，关注微服务参考架构，也比较成熟，难以改动**

**kubernetes dashboard 安装 问题：**

[**Kubernetes入门(二)——Dashboard 安装 - bjehp - 博客园 (cnblogs.com)**](https://www.cnblogs.com/ljhbjehp/p/13567354.html)

**自拉镜像：**

[**查看kubernets上的image信息 - PanPan003 - 博客园 (cnblogs.com)**](https://www.cnblogs.com/panpanwelcome/p/10138864.html)

**# 查看pods所使用的image**

**kubectl describe pods $podsname -n $namespace**

**Image: kubernetesui/metrics-scraper:v1.0.1**

**docker pull kubernetesui/metrics-scraper:v1.0.1**

**Image: kubernetesui/dashboard:v2.0.0-beta5**

**docker pull registry.cn-hangzhou.aliyuncs.com/google\_containers/dashboard:v2.0.0-beta5**

**Deleted Images:**

**untagged: k8s.gcr.io/kube-proxy:v1.23.0**

**untagged: k8s.gcr.io/coredns/coredns:v1.8.6**

**untagged: k8s.gcr.io/etcd:3.5.1-0**

**untagged: k8s.gcr.io/kube-controller-manager:v1.23.0**

**untagged: k8s.gcr.io/pause:3.6**

**untagged: k8s.gcr.io/kube-apiserver:v1.23.0**

**untagged: k8s.gcr.io/kube-scheduler:v1.23.0**

**The connection to the server 192.168.31.100:6443 was refused - did you specify the right host or port?**

**解决：可能是相关镜像untag或没有该镜像**

**kubernetes-dashboard 创建的服务**

**namespace/kubernetes-dashboard unchanged**

**serviceaccount/kubernetes-dashboard unchanged**

**service/kubernetes-dashboard configured**

**secret/kubernetes-dashboard-certs unchanged**

**secret/kubernetes-dashboard-csrf unchanged**

**secret/kubernetes-dashboard-key-holder unchanged**

**configmap/kubernetes-dashboard-settings unchanged**

**role.rbac.authorization.k8s.io/kubernetes-dashboard unchanged**

**clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard unchanged**

**rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard unchanged**

**clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard unchanged**

**deployment.apps/kubernetes-dashboard unchanged**

**service/dashboard-metrics-scraper unchanged**

**deployment.apps/dashboard-metrics-scraper unchanged**

**kubernetes-dashboard 卸载：**

**kubectl delete -f auth.yaml**

**kubectl delete -f v2.0.0-beta5.yaml**

**kubectl config view --minify -o jsonpath='{.clusters[?(@.name==\"kubernetes\")].cluster.server}**

**资源查询：**

**【1本身】metrics-server-5f8db484f9-b9hqr 0/1 Running 23 (18m ago) 52d**

**Error: unable to load configmap based request-header-client-ca-file: Get "https://10.96.0.1:443/api/v1/namespaces/kube-system/configmaps/extension-apiserver-authentication": dial tcp 10.96.0.1:443: i/o timeout**

**本身查不到，边缘端打开可以查到，**

**root@raspberrypi-k8s-master:/home/xl# kubectl top nodes**

**NAME CPU(cores) CPU% MEMORY(bytes) MEMORY%**

**raspberrypi-kubeedge1 277m 6% 1302Mi 16%**

**raspberrypi-k8s-master <unknown> <unknown> <unknown> <unknown>**

**但是边缘端不工作时**

**root@raspberrypi-k8s-master:/home/xl# kubectl top nodes**

**Error from server (ServiceUnavailable): the server is currently unable to handle the request (get nodes.metrics.k8s.io)**

**root@raspberrypi-k8s-master:/home/xl# kubectl top pods**

**Error from server (ServiceUnavailable): the server is currently unable to handle the request (get pods.metrics.k8s.io)**

**【2 board】**

**取消注释：**

**- --apiserver-host=http://my-address:port**

**pod board可以run，但metric仍然crash，**

**pod dashboard-metrics 显示错误：**

**{"level":"info","msg":"Kubernetes host: https://10.96.0.1:443","time":"2024-03-10T08:18:04Z"}**

**{"level":"fatal","msg":"Unable to initialize database tables: Binary was compiled with 'CGO\_ENABLED=0', go-sqlite3 requires cgo to work. This is a stub","time":"2024-03-10T08:18:04Z"}**

**可能与arm结构以及sqlite数据库架构有关，**

[**pod Lifecycle 及 pod 异常状态处理\_keep的技术博客\_51CTO博客**](https://blog.51cto.com/keep11/5899592)

**pod处于 Evicted 状态**

**实例是因为节点资源不足被驱逐，但是这些实例并没有被自动清理**

**kubectl get pods | grep Evicted | awk '{print $1}' | xargs kubectl delete pod**

**kubectl get pods -n** kubeedge **| grep Evicted | awk '{print $1}' | xargs kubectl delete pod -n kubeedge**

**kubectl get pods -A | grep Evicted | awk '{print $1}' | xargs kubectl delete pod -A**

**删除非running**

**kubectl get pods -A | grep -v Running | awk '{print $2}' | xargs kubectl delete pod -n kubesphere-system**

**kubectl get pods -A | grep Terminating | awk '{print $1}' | xargs kubectl delete pod -n kuboard --force --grace-period=0**

**kubectl get pods -A | grep -v Running | awk '{print $2}' | xargs kubectl delete pod -n kubesphere-system**

[**K8s无法删除状态为terminating的pod解决方法\_k8s terminating-CSDN博客**](https://blog.csdn.net/lisongyue123/article/details/118966921)

**pod卡在terminating，delete很长时间无法删除：**

**（1)强制删除**

**kubectl delete pod xxx -n xxx --force --grace-period=0**

**kubectl delete pod calico-apiserver-547976699c-85v56 -n calico-apiserver --force --grace-period=0**

**（2)如果强制删除还不行，设置finalizers为空**

**（如果一个容器已经在运行，这时需要对一些容器属性进行修改，又不想删除容器，或不方便通过replace的方式进行更新。kubernetes还提供了一种在容器运行时，直接对容器进行修改的方式，就是patch命令。）**

**kubectl patch pod xxx -n xxx -p '{"metadata":{"finalizers":null}}'**

**docker.io/calico/node-driver-registrar:v3.24.5**

**docker.io/calico/csi:v3.24.**

[**K8S故障检查-Pod处于ContainerCreating状态 - 知乎 (zhihu.com)**](https://zhuanlan.zhihu.com/p/639994067)

**kubernetes安装Dashboard**

[**【kubernetes系列之安装Dashboard】\_kubernetes dashboard安装-CSDN博客**](https://blog.csdn.net/xuan_lu/article/details/123141368)

[**Kubernetes Dashboard 安装与使用-阿里云开发者社区 (aliyun.com)**](https://developer.aliyun.com/article/745086)

[**部署和访问 Kubernetes 仪表板（Dashboard） | Kubernetes**](https://kubernetes.io/zh-cn/docs/tasks/access-application-cluster/web-ui-dashboard/)

[**安装Kubernetes Dashboard | Kuboard**](https://kuboard.cn/install/install-k8s-dashboard.html#%E5%AE%89%E8%A3%85)

**curl下载文件：**

**curl -O https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml**

**可能因为版本兼容、更新等问题，版本v2.7.0pod直接就可以running**

**访问地址：**

**端口可通过kubectl get svc --all-namespaces 查询**

[**https://192.168.31.100:31225/**](https://192.168.31.100:31225/)

**kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{print $1}')**

**eyJhbGciOiJSUzI1NiIsImtpZCI6IlBVcWVfcEQ1cmFNWkVHUG1KQzZ1ejZOYW1aME1FeXFJZ05wUlUwZ1R4TmsifQ..T3NxwLO9Vf3cCs2jnnNNQUlpy5COZVeX7oQThR642qr3a1sYMa4yn3Mq-Tvsb-cO-MPFFD7hWUP\_MbgQPas5u0LBdjax-j97sIgpG3kG6Mu-Z-MYE5vJ1KLGVcljwkr9b9\_1\_5UdD7zaVQXMZl96JcoAXEAm4kgLuuiklVayTNkx8sfSx2JtIXnP8cHTwYz2zPmvt9MDGVXjoW0k8epbRyuh96EoMn7wEUf5Bw9ufA-2EE9LDpIcmD0V70GUa\_5V0Rf41LnzLX7ZMMz3jRCgQTUAysa7gfpDII6SBum3n8jsJOQe9ulS6Rlp2Im3g0nRA883XdWQTUADnOgovRYMmg**

**NAMESPACE NAME READY STATUS RESTARTS AGE**

**calico-apiserver calico-apiserver-547976699c-jnp6h 1/1 Running 6 (140m ago) 26h**

**calico-apiserver calico-apiserver-547976699c-jxbps 1/1 Running 24 (140m ago) 55d**

**calico-system calico-kube-controllers-66ddf5b4df-xsdz8 1/1 Running 24 (140m ago) 55d**

**calico-system calico-node-2vlnh 0/1 Running 24 (140m ago) 55d**

**calico-system calico-node-2x9ws 0/1 Running 18 (138m ago) 53d**

**calico-system calico-typha-66858f48f4-cpdtp 1/1 Running 34 (138m ago) 55d**

**calico-system csi-node-driver-cq6mv 0/2 ContainerCreating 0 53d**

**calico-system csi-node-driver-rvwtz 2/2 Running 49 (140m ago) 55d**

**kube-system coredns-64897985d-7jhzk 1/1 Running 24 (140m ago) 55d**

**kube-system coredns-64897985d-tv72z 1/1 Running 24 (140m ago) 55d**

**kube-system etcd-raspberrypi-k8s-master 1/1 Running 67 (140m ago) 55d**

**kube-system kube-apiserver-raspberrypi-k8s-master 1/1 Running 47 (140m ago) 52d**

**kube-system kube-controller-manager-raspberrypi-k8s-master 1/1 Running 54 (140m ago) 55d**

**kube-system kube-proxy-gkpdf 1/1 Running 20 (138m ago) 53d**

**kube-system kube-proxy-whd77 1/1 Running 25 (140m ago) 55d**

**kube-system kube-scheduler-raspberrypi-k8s-master 1/1 Running 53 (65m ago) 55d**

**kube-system metrics-server-5f8db484f9-b9hqr 0/1 Running 25 (137m ago) 52d**

**kubeedge cloudcore-5b5d64d755-hz86q 0/1 Error 16 53d**

**kubeedge cloudcore-5b5d64d755-zch57 1/1 Running 6 (140m ago) 26h**

**kubernetes-dashboard dashboard-metrics-scraper-6f669b9c9b-bqtkd 1/1 Running 0 67m**

**kubernetes-dashboard kubernetes-dashboard-758765f476-2c4bb 1/1 Running 0 67m**

**kuboard kuboard-agent-2-8799659db-26nwc 1/1 Running 53 (133m ago) 53d**

**kuboard kuboard-agent-857d885c74-svdtn 1/1 Running 57 (132m ago) 53d**

**kuboard kuboard-etcd-g8h97 1/1 Running 6 (140m ago) 26h**

**kuboard kuboard-v3-56b4b954c9-vls94 1/1 Running 7 (140m ago) 26h**

**tigera-operator tigera-operator-75c746649f-pfs7t 0/1 CrashLoopBackOff 30 (134m ago) 26h**

**logs of tigera-operator-75c746649f-pfs7t:**

**invalid configuration: no configuration has been provided, try setting KUBERNETES\_MASTER environment variable**

[**No configuration has been provided, try setting KUBERNETES\_MASTER environment variable\_invalid configuration: no configuration has been p-CSDN博客**](https://blog.csdn.net/wozaizhe56/article/details/127416667)

[**无法访问Kubernetes集群：无效配置：未提供配置，请尝试设置KUBERNETES\_MASTER环境变量· Issue #1234 · terraform-aws-modules/terraform-aws-eks --- Kubernetes cluster unreachable: invalid configuration: no configuration has been provided, try setting KUBERNETES\_MASTER environment variable · Issue #1234 · terraform-aws-modules/terraform-aws-eks (github.com)**](https://github.com/terraform-aws-modules/terraform-aws-eks/issues/1234)

**export KUBE\_CONFIG\_PATH=~/.kube/config**

**前端环境：**

**1.node.js**

[**Ubuntu安装Node.js（亲测，最新，详细）\_乌班图虚拟机解压安装node-CSDN博客**](https://blog.csdn.net/zwslovexyj/article/details/120650976)

**2.VsCode开发工具**

**k8s部署edgex**

[**edgex-examples/deployment/helm at v3.0 · edgexfoundry/edgex-examples (github.com)**](https://github.com/edgexfoundry/edgex-examples/tree/v3.0/deployment/helm)

[**系列文章（七）丨Kubernetes上的EdgeX Foundry (qq.com)**](https://mp.weixin.qq.com/s/ECdEkc9QdkVScn4Lvl_JJA)

https://mp.weixin.qq.com/s/KDzLG3X8o2\_BcM9da\_rZ5w

<https://GitHub.com/WormOn/edgecomputing/tree/master/end>

**架构**

[详解边缘计算系统逻辑架构：云、边、端协同 (qq.com)](https://mp.weixin.qq.com/s/cLcUNff3bcILTJ6u4wZjlw)

**云边端详解—安装、原理、源码**

[「连载」边缘计算（三十八）03-21：边缘部分源码分析（源码分析篇） (qq.com)](https://mp.weixin.qq.com/s/zD1KdilHbx5Yntvv5DoW0Q)

**1.modbus**

[**Modbus - EdgeX Foundry文档 --- Modbus - EdgeX Foundry Documentation**](https://docs.edgexfoundry.org/3.0/examples/Ch-ExamplesAddingModbusDevice/)

[**详解Modbus通信协议---清晰易懂-CSDN博客**](https://blog.csdn.net/as480133937/article/details/123197782)

[**大神带你秒懂Modbus通信协议-CSDN博客**](https://blog.csdn.net/tiandiren111/article/details/118347661)

**用在工业上的简单协议**

**大致分为以下几种:**

* **Modbus-RTU**
* **Modbus-ASCII**
* **Modbus-TCP**

**主从方式通信，有问即答**

**2.opcua**

[**OPC UA详细介绍 - 知乎 (zhihu.com)**](https://zhuanlan.zhihu.com/p/430243728)

**1. mqtt**

device.config.yaml deviceList、protocols、autoEvents

device.profile.yml value、cmd

docker-compose.yml make gen ds-mqtt mqtt-broker no-secty ui

docker-compose.override.yml path to congig/profile

**上报**

incoming/data/my-custom-device/values

**订阅**

"command/my-custom-device/#"

**收到get请求**

0 /1 /2cmd /3method/4uuid

command/my-custom-device/randnum/get/293d7a00-66e1-4374-ace0-07520103c95f

**回复**"randnum"

command/response/#

**收到set请求**

command/my-custom-device/testmessage/set/293d7a00-66e1-4374-ace0-07520103c95f

**回复**"message"

command/response/#

**"command/response/"+ uuid**

**SET 命令**

curl http://localhost:59882/api/v3/device/name/my-ds18b20-01/message \

-H "Content-Type:application/json" -X PUT \

-d '{"message":"Hello!"}'

不能用空格 ！！符号

**GET 命令（api一样）**

PUT请求

json\_pp命令会接收这个JSON输出，并将其格式化以便更易于阅读。

curl http://localhost:59882/api/v3/device/name/my-custom-device/message | json\_pp

收到的结果：。。。。。。

"id":"13164041-2e6c-4454-9bc3-8e8987e85311", UUID

"origin":1660298227470009014, 时间戳

**autoEvents**

查询 core-data 的读取 API

curl http://localhost:59880/api/v3/reading/resourceName/message | json\_pp

* **name = device name**
* **cmd = deviceResource name**
* **method = get or set**
* **cmd = device reading**

**2. modbus设备接入步骤**

<https://sourceforge.net/p/modbuspal/discussion/899955/thread/72cf35ee/cd1f/attachment/ModbusPal.jar> 下载ModbusPal

sudo apt install librxtx-java

sudo java -jar ModbusPal.jar

**device-modbus:**

DEVICE\_DEVICESDIR: /home/xl2/custom-config/temperature\_modbus/

DEVICE\_PROFILESDIR: /home/xl2/custom-config/temperature\_modbus/

volumes:

- /path/to/custom-config:/home/xl2/custom-config/temperature\_modbus/

**npm**

# 删除包

rimraf node\_modules

npm uninstall -g nodemon --save-dev

npm uninstall

**摄像头：**

增加设备：

curl -X POST -H 'Content-Type: application/json' \

http://localhost:59881/api/v3/device \

-d '[

{

"apiVersion" : "v3",

"device": {

"name":"Camera001",

"serviceName": "device-onvif-camera",

"profileName": "onvif-camera",

"description": "My test camera",

"adminState": "UNLOCKED",

"operatingState": "UP",

"protocols": {

"Onvif": {

"Address": "192.168.31.101",

"Port": "8081",

"MACAddress": "d8:3a:dd:92:f1:13",

"FriendlyName":"Default Camera"

},

"CustomMetadata": {

"Location":"raspi camera"

}

}

}

}

]'

更新密文存储中的证书：

curl --data '{

"apiVersion" : "v3",

"secretName": "creds001",

"secretData":[

{

"key":"username",

"value":" "

},

{

"key":"password",

"value":" "

},

{

"key":"mode",

"value":"usernametoken"

}

]

}' -X POST "http://localhost:59984/api/v3/secret"

映射mac和secret：

curl --data 'd8:3a:dd:92:f1:13' -X PUT "http://localhost:8500/v1/kv/edgex/v3/device-onvif-camera/AppCustom/CredentialsMap/creds001"

视频流地址：

streamURI: rtsp://192.168.31.101:8554/h264

查数据：

http://192.168.31.101:59882/api/v3/device/name/Camera001/SnapshotUri

[Could not initialize SDL - No available video device (Did you set the DISPLAY variable?)-CSDN博客](https://blog.csdn.net/qq_22948593/article/details/109740372)

镜像网站：[烂泥：docker.io、gcr.io、quay.io镜像加速(20200413更新)-烂泥行天下 (ilanni.com)](https://www.ilanni.com/?p=14534)

dashboard：

pnpm dev --apiserver=https://192.168.31.100:6443

kubectl create serviceaccount curl-user -n kube-system

kubectl create clusterrolebinding curl-user-binding --clusterrole=cluster-admin --serviceaccount=kube-system:curl-user -n kube-system

kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep curl-user | awk '{print $1}')

eyJhbGciOiJSUzI1NiIsImtpZCI6IlBVcWVfcEQ1cmFNWkVHUG1KQzZ1ejZOYW1aME1FeXFJZ05wUlUwZ1R4TmsifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJjdXJsLXVzZXItdG9rZW4tNHY4ZnIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiY3VybC11c2VyIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQudWlkIjoiYjkyNzkyNTctNjA0NS00NmIzLWIzOTktM2I5MDAzODc5NDU0Iiwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50Omt1YmUtc3lzdGVtOmN1cmwtdXNlciJ9.Ji5FlP3BPH-pUl4Veb4OtFbU3qhge8MDgOV5t5U3w5k82GdQpeITjzp-B8kqUeAbTLDbQOixSNVQTy1rNcJHB6IsZPxWSGN3XTvQakG1gG5Uka28VQSExgbuK7KW8E\_GndvBLQUKYGS\_TVoj\_4G5Cn\_y8qT5\_A5sFGoDlYqmDO2wiRgGNNGjvsgSBSlLHXYXbT7RTP15zPDKCXJPEsq\_01kwBqLJN2x1jCcoPM0cnhap3Gs0FcW\_DMABLmHCQTvYTjxEjsYf-pEkfeeIPfu\_KVign97C-OY53gKgrM0FKZWLtlUkPlhqbSa9Vl6wg1IxWM8jWHLiDJixnpPMt7cHog

kubeege api 操作：

kubectl proxy --port=8080 &

curl http://localhost:8080/api/ '{"versions": ["v1"]}'

1、namespacelist

curl http://localhost:8080/api/v1/namespaces

{

"kind": "NamespaceList",

"apiVersion": "v1",

"metadata": {

"resourceVersion": "240726"

},

"items": [

{

"metadata": {

"name": "calico-apiserver",

"uid": "4774da05-b0a5-4166-b7de-919ca6472cc6",

"resourceVersion": "4747",

"creationTimestamp": "2024-04-23T08:04:00Z",

"labels": {

"kubernetes.io/metadata.name": "calico-apiserver",

"name": "calico-apiserver",

"pod-security.kubernetes.io/enforce": "privileged",

"pod-security.kubernetes.io/enforce-version": "latest"

},

"ownerReferences": [

{

"apiVersion": "operator.tigera.io/v1",

"kind": "APIServer",

"name": "default",

"uid": "f6283e71-72f8-4002-a857-d8d6237dc505",

"controller": true,

"blockOwnerDeletion": true

}

],

"managedFields": [

{

"manager": "operator",

"operation": "Update",

"apiVersion": "v1",

"time": "2024-04-23T08:04:00Z",

"fieldsType": "FieldsV1",

"fieldsV1": {

"f:metadata": {

"f:labels": {

".": {},

"f:kubernetes.io/metadata.name": {},

"f:name": {},

"f:pod-security.kubernetes.io/enforce": {},

"f:pod-security.kubernetes.io/enforce-version": {}

},

"f:ownerReferences": {

".": {},

"k:{\"uid\":\"f6283e71-72f8-4002-a857-d8d6237dc505\"}": {}

}

}

}

}

]

},

"spec": {

"finalizers": [

"kubernetes"

]

},

"status": {

"phase": "Active"

}

},

2.查看设备

curl <http://localhost:8080/apis/devices.kubeedge.io/v1alpha2/namespaces/default/devices/>

kubeedge-API

get 数据

<https://127.0.0.1:1215/api/v1/device/id/random-instance-01/random-float>

成功：

http://localhost:8080/api/v1/device/id/sensor-tag-instance-01/temperature

post 加设备 写configmap

https://127.0.0.1:1215/api/v1/callback/device

put 写数据

https://127.0.0.1:1215/api/v1/device/id/random-instance-01?random-int=1

del 删除设备

https://127.0.0.1:1215/api/v1/callback/device/id/random-instance-01

node detail

curl http://localhost:8080//api/v1/nodes/kubeedge1

instance

curl http://localhost:8080/apis/devices.kubeedge.io/v1alpha2/namespaces/default/devices/sensor-tag-instance-01

~~http://169.254.219.194:8000/apis/devices.kubeedge.io/v1alpha2/devices/sensor-tag-instance-01~~

*model add 属性*

Request error 422: http://169.254.219.194:8000/apis/devices.kubeedge.io/v1alpha2/namespaces/default/devicemodels

When creating an object, a validation error occurred.

curl http://localhost:8080/apis/devices.kubeedge.io/v1alpha2/namespaces/default/devicemodels/h-model

设备model需求：

属性的属性访问模式可以是 ReadWrite 或 ReadOnly 之一。不支持其他访问模式。

属性类型只能是 string 或 int64。不支持其他类型。

目前仅支持blueteeth、modbus 和 opcua 协议的物业访客。不支持其他协议访问者。

**helm 部署edgex：**

helm3.7.1安装

[Kubernetes K8S之Helm部署、使用与示例 - 踏歌行666 - 博客园 (cnblogs.com)](https://www.cnblogs.com/zhanglianghhh/p/14165995.html)

wget https://get.helm.sh/helm-v3.7.1-linux-amd64.tar.gz

tar xf helm-v3.7.1-linux-amd64.tar.gz

cp -a linux-amd64/helm /usr/bin/helm

git clone https://github.com/edgexfoundry/edgex-examples.git

git tag

git checkout v3.0.0

git branch

必要安装包：

helm upgrade --install ingress-nginx ingress-nginx \

--repo https://kubernetes.github.io/ingress-nginx \

--namespace ingress-nginx --create-namespace

没网，用yaml文件

[解决国内k8s的ingress-nginx镜像无法正常pull拉取问题\_ingress-nginx imagepullbackoff-CSDN博客](https://blog.csdn.net/weixin_43988498/article/details/122792536)

[安装MetalLB — Cloud Atlas beta 文档 (cloud-atlas.readthedocs.io)](https://cloud-atlas.readthedocs.io/zh-cn/latest/kubernetes/network/metallb/install_metallb.html)

规则：：

mqtt

{

  "id": "001",

  "sql": "SELECT temperature, ping, message, json, meta(sourceName) AS sourceName FROM events WHERE meta(deviceName)= \"my-ds18b20-01\"",

  "actions": [

    {

      "mqtt": {

        "server": "tcp://192.168.31.101:1884",

        "topic": "$hw/events/device/my-ds18b20-01/twin/update",

        "clientId": "demo\_001",

        "sendSingle": **true**,

        "dataTemplate": "{\"event\_id\": \"\", \"timestamp\":0, \"twin\":{\"*{{.sourceName}}*\":{\"actual\":{\"value\":\"*{{if eq .sourceName \"temperature\"}}{{.temperature}}{{else if eq .sourceName \"ping\"}}{{.ping}}{{else if eq .sourceName \"message\"}}{{.message}}{{else}}{{.json}}{{end}}*\", \"metadata\":{\"type\":\"Updated\"}}}}}"

      }

    },

    {

      "log":{}

    }

  ]

}

modbus

{

  "id": "002",

  "sql": "SELECT Temperature, ThermostatL, ThermostatH, AlarmMode, meta(sourceName) AS sourceName FROM events WHERE meta(deviceName)= \"sensor-tag01\"",

  "actions": [

    {

      "mqtt": {

        "server": "tcp://192.168.31.101:1884",

        "topic": "$hw/events/device/sensor-tag01/twin/update",

        "clientId": "demo\_002",

        "sendSingle": **true**,

        "dataTemplate": "{\"event\_id\": \"\", \"timestamp\":0, \"twin\":{\"*{{.sourceName}}*\":{\"actual\":{\"value\":\"*{{if eq .sourceName \"Temperature\"}}{{.Temperature}}{{else if eq .sourceName \"ThermostatL\"}}{{.ThermostatL}}{{else if eq .sourceName \"ThermostatH\"}}{{.ThermostatH}}{{else}}{{.AlarmMode}}{{end}}*\", \"metadata\":{\"type\":\"Updated\"}}}}}"

      }

    },

    {

      "log":{}

    }

  ]

}

curl http://localhost:59882/api/v3/device/name/sensor-tag01/ThermostatL -H "Content-Type:application/json" -X PUT -d '{"ThermostatL":"15"}'

http://192.168.31.101:59882/api/v3/device/name/Camera001/SnapshotUri



参考文献

[云计算研究综述：发展、挑战与未来 (baidu.com)](https://cloud.baidu.com/article/1937418)

[什么是云原生？这回终于有人讲明白了 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/150190166)

[从云计算到云原生：从概念到落地 - 廖雪峰的官方网站 (liaoxuefeng.com)](https://www.liaoxuefeng.com/article/1472071259389984)

[云原生的定义 · Kubernetes 中文指南——云原生应用架构实战手册 (jimmysong.io)](https://jimmysong.io/kubernetes-handbook/cloud-native/cloud-native-definition.html)

[Kubernetes已进入成熟期，而云原生时代则刚刚开始-腾讯云开发者社区-腾讯云 (tencent.com)](https://cloud.tencent.com/developer/article/1636396)

[Kubernetes（K8s）：容器编排的未来是什么？-阿里云开发者社区 (aliyun.com)](https://developer.aliyun.com/article/1377503)

[\*中兴通讯技术146net (zte.com.cn)](https://www.zte.com.cn/content/dam/zte-site/res-www-zte-com-cn/mediares/magazine/publication/com_cn/article/201903/cn201903001.pdf)

<https://www.spiceworks.com/tech/edge-computing/articles/best-edge-computing-platforms/>

[1]梁家越,刘斌,刘芳.边缘计算开源平台现状分析[J].中兴通讯技术,2019,25(03):8-14.

[边缘计算平台对比：主流选择与未来趋势 (baidu.com)](https://cloud.baidu.com/article/409677)

[边缘计算 · Kubernetes 中文指南——云原生应用架构实战手册 (jimmysong.io)](https://jimmysong.io/kubernetes-handbook/usecases/edge-computing.html)

[[原创]边缘计算开源方案对比 - wsjhk - 博客园 (cnblogs.com)](https://www.cnblogs.com/wsjhk/p/12103998.html)

[边缘计算云原生开源方案选型比较 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/353429279)

[边缘计算 · Kubernetes 中文指南——云原生应用架构实战手册 (jimmysong.io)](https://jimmysong.io/kubernetes-handbook/usecases/edge-computing.html)

[Kubernetes K8S 基本概述、设计架构和设计理念-腾讯云开发者社区-腾讯云 (tencent.com)](https://cloud.tencent.com/developer/article/1718383)

[概述 | Kubernetes](https://kubernetes.io/zh-cn/docs/concepts/overview/)

[KubeEdge简介，优势，应用场景及特点\_kubeedge应用-CSDN博客](https://blog.csdn.net/an1090239782/article/details/121017965)

[EdgeX Foundry 关于/综述 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/682322513)

curl http://localhost:59882/api/v3/device/name/sensor-tag01/ThermostatL -H "Content-Type:application/json" -X PUT -d '{"ThermostatL":"15"}'

curl http://localhost:59882/api/v3/device/name/sensor-tag01/ThermostatH -H "Content-Type:application/json" -X PUT -d '{"ThermostatH":"100"}'