KubeEdge + Sedna 安装部署(By AdaYang)

说在前面

1. k8s 只需要安装在 master 节点上，其他的节点都不用
2. kubeedge的运行前提是master上必须有k8s
3. docker只是用来发布容器pods的
4. **calico只需要安装在master上**，它是节点通信的插件，如果没有这个，master上安装kubeedge的coredns会报错。但是，节点上又不需要安装这个，因为kubeedge针对这个做了自己的通信机制
5. 一些插件比如calico、edgemesh、sedna、metric-service还有kuborad等，都是通过yaml文件启动的，所以实际要下载的是**k8s的控制工具kubeadm和kubeedge的控制工具keadm**。然后提前准备好刚才的yaml文件，启动k8s和kubeedge后，直接根据yaml文件容器创建
6. namespace可以看作不同的虚拟项目，service是指定的任务目标文件，pods是根据service或者其他yaml文件创建的具体容器
7. 一个物理节点上可以有很多个pods，pods是可操作的最小单位，一个service可以设置很多pods，一个service可以包含很多物理节点
8. 一个pods可以看作一个根据docker镜像创建的实例
9. 如果是主机容器创建任务，要设置dnsPolicy（很重要）
10. 拉去 docker 镜像的时候，**一定要先去确认架构是否支持**

Master 前期准备

关闭防火墙

```bash
ufw disable
```

开启 ipv 4 转发 配置 iptables 参数

将桥接的 `IPv4/IPv6` 流量传递到 iptables 的链

```bash
modprobe br_netfilter
#方法1
cat >> /etc/sysctl.conf << EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
sysctl -p   #执行此命令生效配置
```

禁用 swap 分区

```bash
swapoff -a                                          #临时关闭
sed -ri 's/.*swap.*/#&/' /etc/fstab                 #永久关闭
```

设置主机名

不同虚拟机上设置

```bash
hostnamectl set-hostname master
bash

hostnamectl set-hostname node
bash

hostnamectl set-hostname node
bash
```

设置 DNS

```bash
#不同主机上使用ifconfig查看ip地址
sudo vim /etc/hosts

#根据自己的ip添加
#ip 主机名
192.168.247.128 master
192.168.247.129 node-1
```

安装 docker 并配置

```bash
sudo apt-get install docker.io -y

sudo systemctl start docker
sudo systemctl enable docker

docker --version

# 如果是 ARM64 架构
# 参考 https://blog.csdn.net/qq_34253926/article/details/121629068
```

一定要配置的

```bash
sudo vim /etc/docker/daemon.json

#master添加:
{
  "registry-mirrors": ["https://b9pmyelo.mirror.aliyuncs.com"],
  "exec-opts": ["native.cgroupdriver=systemd"]
}

#node添加:
{
  "registry-mirrors": ["https://b9pmyelo.mirror.aliyuncs.com"],
  "exec-opts": ["native.cgroupdriver=cgroupfs"]
}

#都要执行
sudo systemctl daemon-reload
sudo systemctl restart docker

#查看修改后的docker Cgroup的参数
docker info | grep Cgroup
```

Master 节点安装 kubernetes

添加阿里源

```bash
curl -s https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | sudo
apt-key add -
cat > /etc/apt/sources.list.d/kubernetes.list <<EOF
deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main
EOF
apt-get update
```

安装相关组件

需要在每台机器上安装以下的软件包： - `kubeadm`：用来初始化集群的指令。 - `kubelet`：在集群中的每个节点上用来启动 Pod 和容器等。 - `kubectl`：用来与集群通信的命令行工具。

```bash
#查看软件版本
apt-cache madison kubeadm

#安装kubeadm、kubelet、kubectl
sudo apt-get update
sudo apt-get install -y kubelet=1.22.2-00 kubeadm=1.22.2-00
kubectl=1.22.2-00
#锁定版本
sudo apt-mark hold kubelet kubeadm kubectl
```

```bash
kubeadm init --pod-network-cidr 10.244.0.0/16 \
--image-repository registry.cn-hangzhou.aliyuncs.com/google_containers
```

安装 calico

官方解释：因为在整个 kubernetes 集群里，pod 都是分布在不同的主机上的，为了实现这些 pod 的跨主机通信所以我们必须要安装 CNI 网络插件，这里选择 calico 网络。

步骤1：在 master 上下载配置 calico 网络的 yaml。

```bash
#注意对应版本，v1.22和v3.20
wget https://docs.projectcalico.org/v3.20/manifests/calico.yaml --no-check-certificate
```

步骤2：修改 calico.yaml 里的 pod 网段。

```bash
#把calico.yaml里pod所在网段改成kubeadm init时选项--pod-network-cidr所指定的网段，
#直接用vim编辑打开此文件查找192，按如下标记进行修改：

# no effect. This should fall within `--cluster-cidr`.
# - name: CALICO_IPV4POOL_CIDR
#   value: "192.168.0.0/16"
# Disable file logging so `kubectl logs` works.
- name: CALICO_DISABLE_FILE_LOGGING
  value: "true"

#把两个#及#后面的空格去掉，并把192.168.0.0/16改成10.244.0.0/16

# no effect. This should fall within `--cluster-cidr`.
- name: CALICO_IPV4POOL_CIDR
  value: "10.244.0.0/16"
# Disable file logging so `kubectl logs` works.
- name: CALICO_DISABLE_FILE_LOGGING
  value: "true"
```

步骤3：提前下载所需要的镜像。

```bash
#查看此文件用哪些镜像：
grep image calico.yaml

#image: calico/cni:v3.20.6
#image: calico/cni:v3.20.6
#image: calico/pod2daemon-flexvol:v3.20.6
#image: calico/node:v3.20.6
#image: calico/kube-controllers:v3.20.6
```

在 master 节点中下载镜像

```shell
#换成自己的版本
for i in calico/cni:v3.20.6 calico/pod2daemon-flexvol:v3.20.6 calico/node:v3.20.6 calico/kube-controllers:v3.20.6 ; do docker pull $i ; done
```

安装可视化工具 kuboard

```bash
#master上
wget https://kuboard.cn/install-script/kuboard.yaml
#所有节点下载镜像
docker pull eipwork/kuboard:latest
```

Kube-shell

kube-shell (https://link.zhihu.com/?target=https%3A//github.com/cloudnativelabs/kube-shell) (推荐，kubectl 编辑的小工具)

```bash
#安装
pip install kube-shell

#启动
kube-shell

#退出
exit
```

安装 KubeEdge

安装版本：v1.9.2

安装keadm

进入官网版本链接：Release KubeEdge v1.9.2 release · kubeedge/kubeedge (github.com) (https://github.com/kubeedge/kubeedge/releases/tag/v1.9.2)

右键复制链接地址(注意架构)，节点中输入：

```bash
#查看自己的架构
uname -u

#下载对应版本以及架构
https://github.com/kubeedge/kubeedge/releases/download/v1.9.2/keadm-v1.9.2-linux-amd64.tar.gz
#解压
tar zxvf keadm-v1.9.2-linux-amd64.tar.gz
#添加执行权限
chmod +x keadm-v1.9.2-linux-amd64/keadm/keadm
#移动目录
mv keadm-v1.9.2-linux-amd64/keadm/keadm /usr/local/bin/
```

安装 metrics-server(后续主要用于HPA)

- 用于追踪边缘节点日志
- 官网安装 (会出错, 拉取镜像超时问题):

```bash
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

Info

```
root@master:~/software/yaml# kubectl get pods -A
NAMESPACE     NAME                                       READY   STATUS             RESTARTS   AGE
kube-system   calico-kube-controllers-594649bd75-phr8s   1/1     Running            0          4m14s
kube-system   calico-node-tc86v                          1/1     Running            0          4m14s
kube-system   coredns-7d89d9b6b8-pv9x8                   1/1     Running            0          3h22m
kube-system   coredns-7d89d9b6b8-qxz6j                   1/1     Running            0          3h22m
kube-system   etcd-master                                1/1     Running            0          3h22m
kube-system   kube-apiserver-master                      1/1     Running            0          3h22m
kube-system   kube-controller-manager-master             1/1     Running            0          3h22m
kube-system   kube-proxy-7k8cz                           1/1     Running            0          3h22m
kube-system   kube-scheduler-master                      1/1     Running            0          3h22m
kube-system   kuboard-74c645f5df-kh7ks                   1/1     Running            0          3m25s
kube-system   metrics-server-5cc9cfcd68-zmwc5            0/1     ImagePullBackOff   0          25m
kube-system   metrics-server-665b9cb7ff-r7d5g            0/1     ImagePullBackOff   0          25m
```

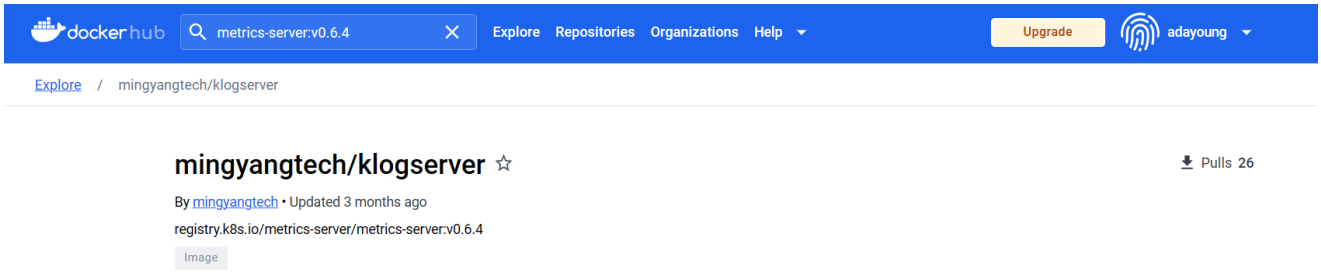
- 本地安装

```bash
# 先下载官方提供的yaml
wget https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

修改 yaml 的内容，先看官方的版本，然后去 docker hub 找对应的镜像，**并且添加“- --kubelet-insecure-tls”**

```
128   template:
129     metadata:
130       labels:
131         k8s-app: metrics-server
132     spec:
133       containers:
134       - args:
135         - --cert-dir=/tmp
136         - --secure-port=4443
137         - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
138         - --kubelet-use-node-status-port
139         - --metric-resolution=15s
140         image: registry.k8s.io/metrics-server/metrics-server:v0.6.4
141         imagePullPolicy: IfNotPresent
142         livenessProbe:
143           failureThreshold: 3
144           httpGet:
```

去 docker hub 找镜像



修改 `components.yaml` 文件

```
Vim
1       spec:
2         containers:
3         - args:
4           - --cert-dir=/tmp
5           - --secure-port=4443
6           - --kubelet-preferred-address-
7   types=InternalIP,ExternalIP,Hostname
8           - --kubelet-use-node-status-port
9           - --metric-resolution=15s
10          - --kubelet-insecure-tls
11          image: mingyangtech/klogserver:v0.6.4
```

```
    containers:
    - args:
      - --cert-dir=/tmp
      - --secure-port=4443
      - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
      - --kubelet-use-node-status-port
      - --metric-resolution=15s
      - --kubelet-insecure-tls
      image: mingyangtech/klogserver:v0.6.4
      imagePullPolicy: IfNotPresent
      livenessProbe:
```

然后手动 pull 镜像

```bash
docker pull mingyangtech/klogserver:v0.6.4
```

Hint

一定要写版本号，否则会报错

```
root@master:~/software/yaml# docker pull docker.io/mingyangtech/klogserver
Using default tag: latest
Error response from daemon: manifest for mingyangtech/klogserver:latest not found: manifest unknown: manif
est unknown
```

EdgeMesh

kubeedge的通讯组件，把github源码下载下来

```bash
git clone https://github.com/kubeedge/edgemesh.git
```

Sedna

官方下载（会超时）

```bash
curl
https://raw.githubusercontent.com/kubeedge/sedna/main/scripts/installatio
n/install.sh | SEDNA_ACTION=create bash -
```

超时报错

```
root@master:~/software/install_kubeedge# curl https://raw.githubusercontent.com/kubeedge/sedna/main/script
s/installation/install.sh | SEDNA_ACTION=create bash -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  9591  100  9591    0     0   3927      0  0:00:02  0:00:02 --:--:--  3925
No resources found in default namespace.
downloading sedna.io_datasets.yaml into /tmp/tmp.kdLHDmZ6c5.sedna/build/crds
downloading sedna.io_federatedlearningjobs.yaml into /tmp/tmp.kdLHDmZ6c5.sedna/build/crds
timeout to download sedna.io_federatedlearningjobs.yamll after 60 seconds...
```

本地安装

- 下载 `install.sh` 文件（不要保存到 build 中，可以永久使用）

```bash
wget https://raw.githubusercontent.com/kubeedge/sedna/main/scripts/installation/install.sh
```

- 相关的修改（不要保存到 build 中，可以永久使用）

```bash
#改名字
mv install.sh offline-install.sh

#修改文件中的一下内容（sh的语法）
1.第28行：TMP_DIR='/opt/sedna'
2.第34行：删除  trap "rm -rf '%TMP_DIR'" EXIT
3.第415行：删除  prepare
```

> ⚠ **Warning**
>
> 记得看 `install.sh` 中 sedna 的版本，去 docker hub 中看镜像是否适用于你的架构!! (主要针对 arm)

- 下载 sedna 的官网 github（不要保存到 build 中，可以永久使用）
  官方地址：http://github.com/kubeedge/sed (https://link.zhihu.com/?target=http%3A//github.com/kubeedge/sedna/tree/main/build)

```bash
git clone https://github.com/kubeedge/sedna.git
```

- 创建安装目录（和上面的路径要一样），并且复制文件

```bash
mkdir /opt/sedna
#把下载的sedna文件中的build目录复制到该地址
mv ./sedna /opt/sedna/
```

Node 节点前期准备

关闭防火墙

```bash
ufw disable
setenforce 0  #临时关闭
```

开启 ipv4转发，配置 iptables 参数

将桥接的 IPv4/IPv6 流量传递到 iptables 的链

```bash
modprobe br_netfilter
#方法1
cat >> /etc/sysctl.conf << EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
sysctl -p   #执行此命令生效配置
```

禁用 swap 分区

```bash
swapoff -a                                          #临时关闭
sed -ri 's/.*swap.*/#&/' /etc/fstab                 #永久关闭
```

设置主机名

在不同的虚拟机上执行

```text
hostnamectl set-hostname master
bash

hostnamectl set-hostname node
bash

```

安装 docker

```bash
sudo apt-get install docker.io -y

sudo systemctl start docker
sudo systemctl enable docker

docker --version

# 如果是 ARM64 架构
# 参考 https://blog.csdn.net/qq_34253926/article/details/121629068
```

一定要配置的

```bash
sudo vim /etc/docker/daemon.json

#master添加:
{
  "registry-mirrors": ["https://b9pmyelo.mirror.aliyuncs.com"],
  "exec-opts": ["native.cgroupdriver=systemd"]
}

#node添加:
{
  "registry-mirrors": ["https://b9pmyelo.mirror.aliyuncs.com"],
  "exec-opts": ["native.cgroupdriver=cgroupfs"]
}

#都要执行
sudo systemctl daemon-reload
sudo systemctl restart docker

#查看修改后的docker Cgroup的参数
docker info | grep Cgroup
```

安装 KubeEdge

- 安装版本：v1.9.2

安装keadm

进入官网版本链接：Release KubeEdge v1.9.2 release · kubeedge/kubeedge (github.com) (https://github.com/kubeedge/kubeedge/releases/tag/v1.9.2)

右键复制链接地址(注意架构），节点中输入：

```bash
#查看自己的架构
uname -a

#下载对应版本以及架构
wget https://github.com/kubeedge/kubeedge/releases/download/v1.9.2/keadm-v1.9.2-linux-arm64.tar.gz
#解压
tar zxvf keadm-v1.9.2-linux-arm64.tar.gz
#添加执行权限
chmod +x keadm-v1.9.2-linux-arm64/keadm/keadm
#移动目录
mv keadm-v1.9.2-linux-arm64/keadm/keadm /usr/local/bin/
```

运行

K8s + KubeEdge

Kubernetes 启动

删除从前的信息（如果是第一次可跳过）

```bash
swapoff -a
kubeadm reset
systemctl daemon-reload
systemctl restart docker kubelet
rm -rf $HOME/.kube/config

rm -f /etc/kubernetes/kubelet.conf    #删除k8s配置文件
rm -f /etc/kubernetes/pki/ca.crt    #删除K8S证书

iptables -F && iptables -t nat -F && iptables -t mangle -F && iptables -X
```

初始化 k8s 集群

Bash Bash

```bash
kubeadm init --pod-network-cidr 10.244.0.0/16 \
--image-repository registry.cn-hangzhou.aliyuncs.com/google_containers

#上一步创建成功后，会提示：主节点执行提供的代码
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

#如果只有一个点，需要容纳污点
kubectl taint nodes --all node-role.kubernetes.io/master node-
role.kubernetes.io/master-
```

可能的问题

```bash
(未理解原因)使用下面的命令可能会报错:
kubeadm init \
  --apiserver-advertise-address=192.168.247.128 \
  --image-repository registry.aliyuncs.com/google_containers \
  --kubernetes-version v1.22.10 \
  --service-cidr=10.96.0.0/12 \
  --pod-network-cidr=10.244.0.0/16 \
  --ignore-preflight-errors=all
(最后加上  --v=5  会输出一些日志)

报错内容如下:

[kubelet-check] Initial timeout of 40s passed.

        Unfortunately, an error has occurred:
                timed out waiting for the condition

        This error is likely caused by:
                - The kubelet is not running
                - The kubelet is unhealthy due to a misconfiguration of
the node in some way (required cgroups disabled)

        If you are on a systemd-powered system, you can try to
troubleshoot the error with the following commands:
                - 'systemctl status kubelet'
                - 'journalctl -xeu kubelet'

        Additionally, a control plane component may have crashed or
exited when started by the container runtime.
        To troubleshoot, list all containers using your preferred
container runtimes CLI.

        Here is one example how you may list all Kubernetes containers
running in docker:
                - 'docker ps -a | grep kube | grep -v pause'
                Once you have found the failing container, you can
inspect its logs with:
                - 'docker logs CONTAINERID'

error execution phase wait-control-plane: couldn't initialize a
Kubernetes cluster
To see the stack trace of this error execute with --v=5 or higher
```

启动 calico 和 kuboard

```bash
#进入这两个文件的目录，一定要执行，不然coredns无法初始化
kubectl apply -f calico.yaml
kubectl apply -f kuboard.yaml

#检查启动状态，需要都running，除了calico可以不用
kubectl get pods -A

#获取 kuboard  token
echo $(kubectl -n kube-system get secret $(kubectl -n kube-system get secret | grep kuboard-user | awk '{print $1}') -o go-template='{{.data.token}}' | base64 -d)
```

KubeEdge 启动

Master

重启（如果第一次可以跳过）

```bash
keadm reset
```

启动 cloudcore

通过keadm启动

```bash
#注意修改--advertise-address的ip地址
keadm init --advertise-address=114.212.81.11 --kubeedge-version=1.9.2


#打开转发路由
kubectl get cm tunnelport -n kubeedge -o yaml
#找到10350 或者10351
#set the rule of trans, 设置自己的端口
CLOUDCOREIPS=xxx.xxx.xxx.xxx

# dport的内容对应tunnelport
iptables -t nat -A OUTPUT -p tcp --dport 10351 -j DNAT --to $CLOUDCOREIPS:10003
```

Edge 节点加入时可能会自动部署 calico-node 和 kube-proxy，kube-proxy 会部署成功（但是 edgecore 的 log 会提示不应该部署 kube-proxy），calico 会初始化失败。为了避免上述情况，做如下操作：

```bash
kubectl get daemonset -n kube-system | grep -v NAME | awk '{print $1}'
|xargs -n 1 kubectl patch daemonset -n kube-system --type='json' -
p='[{"op":"replace","path":"/spec/template/spec/affinity","value":
{"nodeAffinity":{"requireDuringSchedulingIgnoredDuringExecution":
{"nodeSelectorTerms":[{"matchExpressions":[{"key":"node-
role.kubernetes.io/edge","operator":"DoesNotExist"}]}]}}}}]'

$ kubectl edit daemonset -n kube-system kube-proxy

# 添加以下内容
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: node-role.kubernetes.io/edge
                operator: DoesNotExist

# calico-node添加相同内容
$ kubectl edit daemonset -n kube-system calico-node
```

```yaml
spec:
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: kube-proxy
  template:
    metadata:
      creationTimestamp: null
      labels:
        k8s-app: kube-proxy
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: node-role.kubernetes.io/edge
                operator: DoesNotExist
```

启动 metric-service

方便跟踪日志

```bash
#进入自己的下载目录
kubectl apply -f components.yaml
#line 144
#image: xin053/metrics-server:v0.6.2

# 检验是否部署成功
$ kubectl top nodes
```

`kubectl logs/exec`

Enable Kubectl logs/exec to debug pods on the edge | KubeEdge (https://kubeedge.io/docs/advanced/debug/)

```bash
export CLOUDCOREIPS="114.212.81.11"

echo $CLOUDCOREIPS

cd /etc/kubeedge/
cp $GOPATH/src/github.com/kubeedge/kubeedge/build/tools/certgen.sh /etc/kubeedge/

bash certgen.sh stream
```

设置 iptables 规则（理论上启动 cloudcore 就设置好了）

```bash
iptables -t nat -A OUTPUT -p tcp --dport 10350 -j DNAT --to $CLOUDCOREIPS:10003
```

更新配置

- 修改：`/etc/kubeedge/config/cloudcore.yaml`

```
cloudStream:
        enable: true
        streamPort: 10003
        tlsStreamCAFile: /etc/kubeedge/ca/streamCA.crt
        tlsStreamCertFile: /etc/kubeedge/certs/stream.crt
        tlsStreamPrivateKeyFile: /etc/kubeedge/certs/stream.key
        tlsTunnelCAFile: /etc/kubeedge/ca/rootCA.crt
        tlsTunnelCertFile: /etc/kubeedge/certs/server.crt
        tlsTunnelPrivateKeyFile: /etc/kubeedge/certs/server.key
        tunnelPort: 10004
```

- 修改：`/etc/kubeedge/config/edgecore.yaml`

```
edgeStream:
        enable: true
        handshakeTimeout: 30
        readDeadline: 15
        server: 192.168.0.139:10004
        tlsTunnelCAFile: /etc/kubeedge/ca/rootCA.crt
        tlsTunnelCertFile: /etc/kubeedge/certs/server.crt
        tlsTunnelPrivateKeyFile: /etc/kubeedge/certs/server.key
        writeDeadline: 15
```

重启

- 在云端：

```
sudo systemctl restart cloudcore.service
```

- 在边缘侧：

```
sudo systemctl restart edgecore.service
```

Node

启动 edgecore

消除之前的信息（如果第一次，请跳过）

```shell
#重启
keadm reset

#重新加入时，报错存在文件夹，直接删除
rm -rf /etc/kubeedge

#docker容器占用(通常是mqtt)
docker ps -a
docker stop mqtt
docker rm mqtt
```

启动

```shell
#注意这里是要加入master的IP地址，token是master上获取的
keadm join --cloudcore-ipport=114.212.81.11:10000 --kubeedge-version=1.9.2
--
token=621812b3b27f180cceb29a707bb49146abcc8ea04bda810d53d8a0e3fc4c023f.eyJ
hbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MDQyNTIzNjR9.oqFu_1ETGlOhTEr
R8VBkkipDUMQ6tYCzGRLjQQ8Uxog
```

如果一直显示下载问题，可以手动下载对应的文件：

```
root@edge2:/etc/kubeedge# ls
edgecore.service  kubeedge-v1.9.2-linux-arm64  kubeedge-v1.9.2-linux-arm64.tar.gz
root@edge2:/etc/kubeedge#
```

如果 check 时候提醒 check 未通过，询问是否删除源文件重新下载，填 n，否则它又要重新去下载。

重要

```shell
#加入后，先查看信息
journalctl -u edgecore.service -f
```

如果修改了 daemonset 的话此时 `docker ps` 只会有 mqtt

```
root@node:~/software# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED              STATUS              P
ORTS                                          NAMES
da924d66d072   eclipse-mosquitto:1.6.15       "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.
0.0.0:1883->1883/tcp, :::1883->1883/tcp       mqtt
```

可能出现的问题

```shell
execute keadm command failed:  failed to exec 'bash -c sudo ln
/etc/kubeedge/edgecore.service /etc/systemd/system/edgecore.service &&
sudo systemctl daemon-reload && sudo systemctl enable edgecore && sudo
systemctl start edgecore', err: ln: failed to create hard link
'/etc/systemd/system/edgecore.service': File exists
, err: exit status 1
```

在尝试创建符号链接时，目标路径已经存在，因此无法创建。这通常是因为 `edgecore.service` 已经存在于 `/etc/systemd/system/` 目录中

```shell
sudo rm /etc/systemd/system/edgecore.service
```

`kubectl` 安装

1. 下载 kubectl

```shell
curl -LO https://storage.googleapis.com/kubernetes-
release/release/v1.17.0/bin/linux/amd64/kubectl
```

2. 使 kubectl 二进制可执行文件

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

3. 将二进制文件移到 PATH中

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

4. 测试安装版本

```shell
kubectl version --client
```

5. 将 master 节点下/etc/kubernetes/admin. Conf 复制到 edge 节点，配置环境变量

```shell
vim /etc/profile
export KUBECONFIG=/etc/kubernetes/admin.conf【修改地址】
source /etc/profile
```

启动 EdgeMesh

前期准备

步骤1: 去除 K8s master 节点的污点

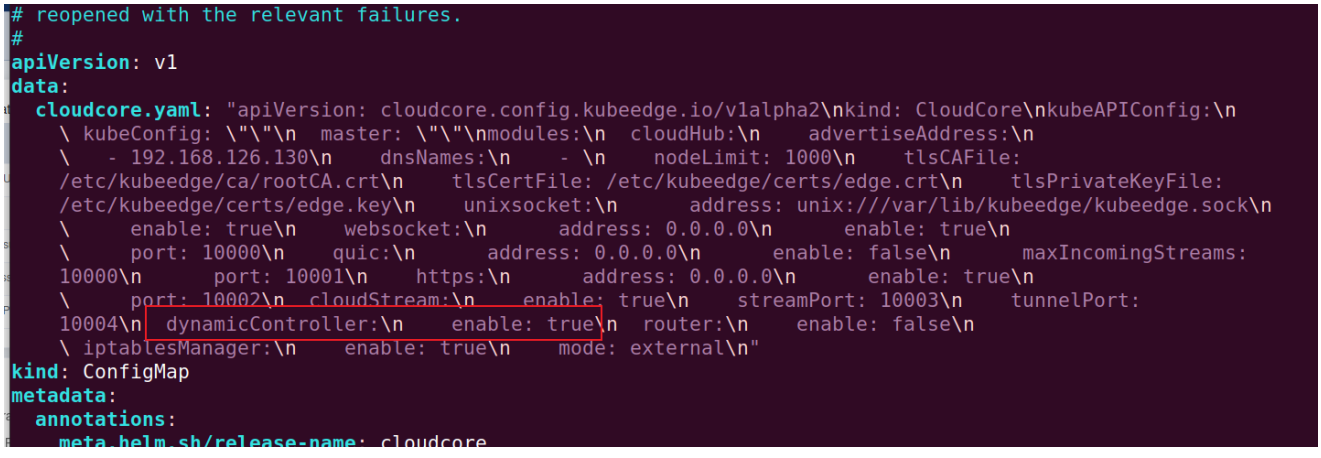
```shell
$ kubectl taint nodes --all node-role.kubernetes.io/master-
```

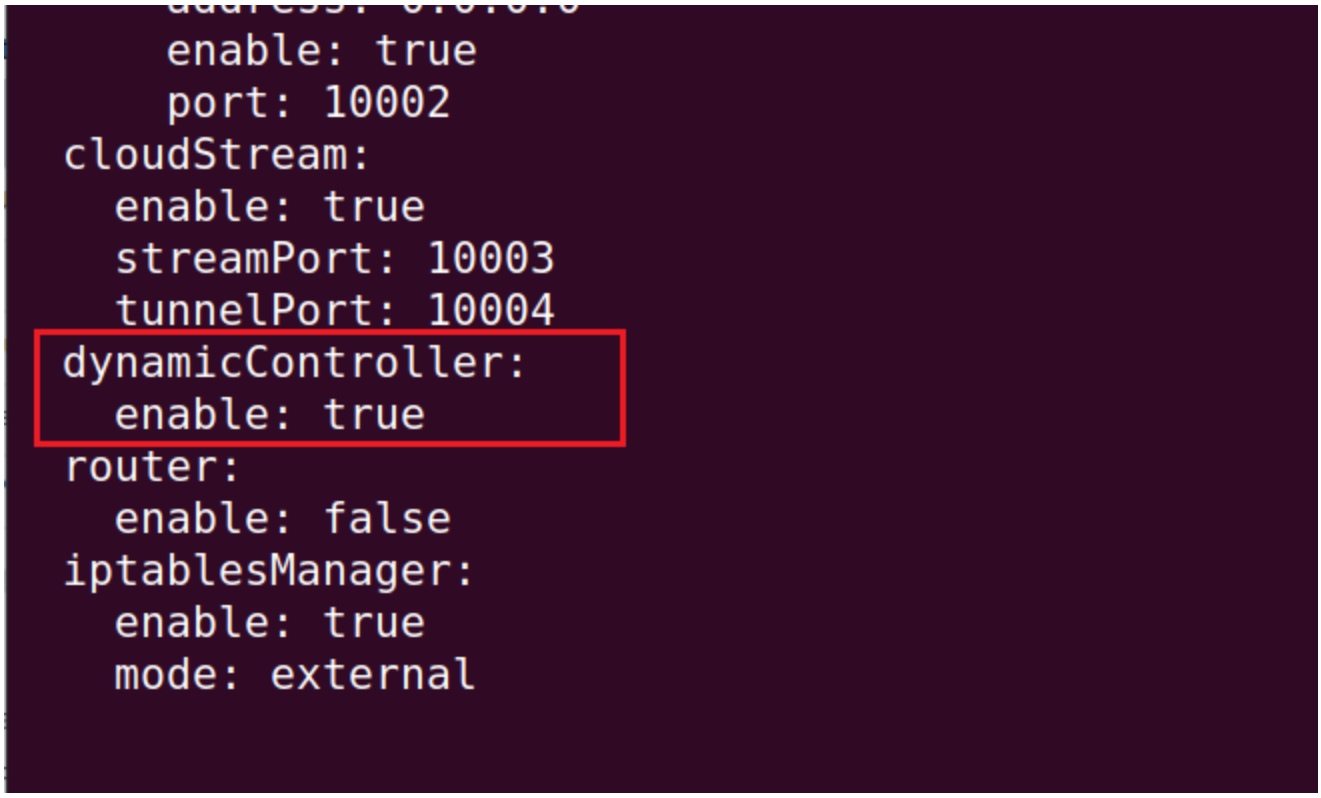
步骤2: 给 Kubernetes API 服务添加过滤标签

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

- **步骤3**: 启用 KubeEdge 的边缘 Kube-API 端点服务**important**
- 在云端，开启 dynamicController 模块，配置完成后，需要重启 cloudcore(**1.10+ 版本，cloudcore 以容器方式运行**)

```shell
#keadm安装的
kubectl edit cm -n kubeedge cloudcore
#修改
modules:
  ...
  dynamicController:
    enable: true
...

#执行完后,检查一下
kubectl describe cm -n kubeedge cloudcore

#如果不放心，直接去kuboard在kubeedge上把cloudcore删除掉，然后会根据新的模板创建新的容器
```





KubeEdge 1.9.2 版本： 修改配置，`vim /etc/kubeedge/config/cloudcore.yaml`，重启 cloudcore 后才生效。重启命令：`systemctl restart cloudcore`

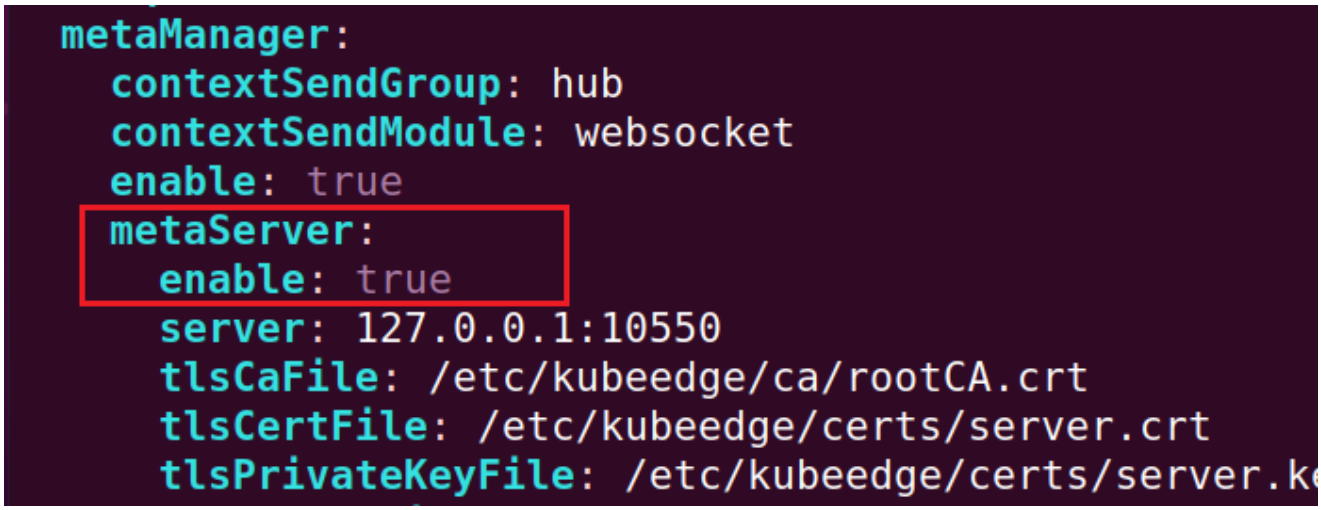
SHELL

```
modules:
  ..
  cloudStream:
    enable: true
    streamPort: 10003
  ..
  dynamicController:
    enable: true
```

在边缘节点，打开 metaServer 模块（如果KubeEdge < 1.8.0，还需关闭旧版 edgeMesh 模块），配置完成后，需要重启 edgecore

SHELL

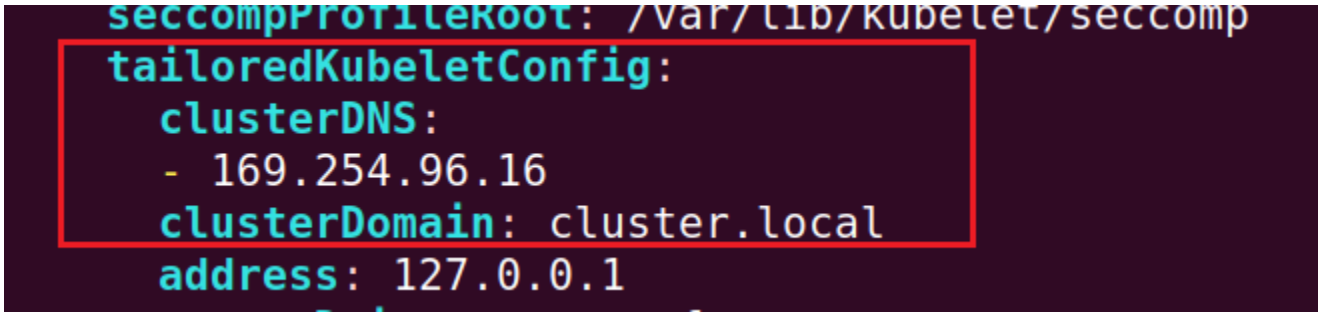
```
$ vim /etc/kubeedge/config/edgecore.yaml
modules:
  ...
  edgeMesh:
    enable: false
  ...
  metaManager:
    metaServer:
      enable: true
...
```



在边缘节点配置 clusterDNS 和 clusterDomain，配置完成后，需要重启 edgecore。

SHELL

```
# 1.12+以下操作
$ vim /etc/kubeedge/config/edgecore.yaml
modules:
  ...
  edged:
    ...
    tailoredKubeletConfig:
      ...
      clusterDNS:
      - 169.254.96.16
      clusterDomain: cluster.local
...

# 1.12-以下操作
$ vim /etc/kubeedge/config/edgecore.yaml
modules:
  ...
  edged:
    clusterDNS: 169.254.96.16
    clusterDomain: cluster.local

#重启
systemctl restart edgecore.service
#检查状况
journalctl -u edgecore.service -f
#确定正常运行
```



添加的内容在 tailoredKubeletConfig 下第一二位才生效，不理解但是不在这个位置就不生效了 即一定是图上这样的顺序

步骤4: 最后，在边缘节点，测试边缘 Kube-API 端点功能是否正常

```shell
# 边缘节点上
curl 127.0.0.1:10550/api/v1/services
```


```
root@node:/joint_inference# curl 127.0.0.1:10550/api/v1/services
{"apiVersion":"v1","items":[{"apiVersion":"v1","kind":"Service","metadata":{"creationTimestamp":"2023-11-05T12:22:13Z","
abels":{"jointinference.sedna.io/name":"helmet-detection-inference-example","jointinference.sedna.io/uid":"71b21fab-533e
4649-9cd7-f05c2383213f","jointinference.sedna.io/worker-type":"cloud"},"managedFields":[{"apiVersion":"v1","fieldsType":
FieldsV1","fieldsV1":{"f:metadata":{"f:labels":{".":{},"f:jointinference.sedna.io/name":{},"f:jointinference.sedna.io/ui
":{},"f:jointinference.sedna.io/worker-type":{}},"f:ownerReferences":{".":{},"k:{\"uid\":\"71b21fab-533e-4649-9cd7-f05c2
83213f\"}":{}}},"f:spec":{"f:internalTrafficPolicy":{},"f:ports":{".":{},"k:{\"port\":5000,\"protocol\":\"TCP\"}":{".":{
,"f:name":{},"f:port":{},"f:protocol":{},"f:targetPort":{}}},"f:selector":{},"f:sessionAffinity":{},"f:type":{}}},"manag
r":"sedna-gm","operation":"Update","time":"2023-11-05T12:22:13Z"}],"name":"helmet-detection-inference-example-cloud","na
espace":"default","ownerReferences":[{"apiVersion":"sedna.io/v1alpha1","blockOwnerDeletion":true,"controller":true,"kind
:"JointInference","name":"helmet-detection-inference-example","uid":"71b21fab-533e-4649-9cd7-f05c2383213f"}],"resourceVe
sion":"8438","uid":"a5524e56-3422-46ad-96fc-5d2a7501f96e"},"spec":{"clusterIP":"10.100.196.188","clusterIPs":["10.100.19
.188"],"internalTrafficPolicy":"Cluster","ipFamilies":["IPv4"],"ipFamilyPolicy":"SingleStack","ports":[{"name":"tcp-0","
ort":5000,"protocol":"TCP","targetPort":5000}],"selector":{"jointinference.sedna.io/name":"helmet-detection-inference-ex
mple","jointinference.sedna.io/uid":"71b21fab-533e-4649-9cd7-f05c2383213f","jointinference.sedna.io/worker-type":"cloud"
,"sessionAffinity":"None","type":"ClusterIP"},"status":{"loadBalancer":{}}},{"apiVersion":"v1","kind":"Service","metadat
":{"creationTimestamp":"2023-11-05T10:46:33Z","labels":{"component":"apiserver","provider":"kubernetes","service.edgemes
.kubeedge.io/service-proxy-name":""},"managedFields":[{"apiVersion":"v1","fieldsType":"FieldsV1","fieldsV1":{"f:metadata
:{"f:labels":{".":{},"f:component":{},"f:provider":{}}},"f:spec":{"f:clusterIP":{},"f:internalTrafficPolicy":{},"f:ipFam
lyPolicy":{},"f:ports":{".":{},"k:{\"port\":443,\"protocol\":\"TCP\"}":{".":{},"f:name":{},"f:port":{},"f:protocol":{},"
:targetPort":{}}},"f:sessionAffinity":{},"f:type":{}}},"manager":"kube-apiserver","operation":"Update","time":"2023-11-0
T10:46:33Z"},{"apiVersion":"v1","fieldsType":"FieldsV1","fieldsV1":{"f:metadata":{"f:labels":{"f:service.edgemesh.kubeed
```


Note

如果返回值是空列表，或者响应时长很久（接近 10 s）才拿到返回值，说明你的配置可能有误，请仔细检查。

可能的问题

```
 TLSStreamPrivateKeyFile: Invalid value: "/etc/kubeedge/certs/stream.key":
TLSStreamPrivateKeyFile not exist
12月 14 23:02:23 cloud.kubeedge cloudcore[196229]:   TLSStreamCertFile:
Invalid value: "/etc/kubeedge/certs/stream.crt": TLSStreamCertFile not
exist
12月 14 23:02:23 cloud.kubeedge cloudcore[196229]:   TLSStreamCAFile:
Invalid value: "/etc/kubeedge/ca/streamCA.crt": TLSStreamCAFile not exist
12月 14 23:02:23 cloud.kubeedge cloudcore[196229]: ]
```

查看 `/etc/kubeedge` 下是否有 `certgen.sh` 并且 `bash certgen.sh stream`

补充

为了在云端可以 logs 边端的 pod 的信息，需要修改 /etc/kubeedge/config/edgecore.yaml 文件，将 `edgeStream` 下 `enable` 改为 true

```
edgeStream:
  enable: true
  handshakeTimeout: 30
  readDeadline: 15
  server: 114.212.81.11:10004
  tlsTunnelCAFile: /etc/kubeedge/ca/rootCA.crt
  tlsTunnelCertFile: /etc/kubeedge/certs/server.crt
  tlsTunnelPrivateKeyFile: /etc/kubeedge/certs/server.key
  writeDeadline: 15
edged:
  clusterDNS: 169.254.96.16
  clusterDomain: cluster.local
  cgroupDriver: cgroupfs
  cgroupRoot: ""
  cgroupsPerQOS: true
  clusterDNS: ""
```

安装

- **步骤1**: 获取 EdgeMesh

```
#或者自己提前下载好
git clone https://github.com/kubeedge/edgemesh.git
cd edgemesh
```

- **步骤2**: 安装 CRDs

```shell
kubectl apply -f build/crds/istio/
```

- **步骤 3**: 部署 edgemesh-agent

  `resources` 中的内容需要进行修改，具体看 Edgemesh log 报错中解决方案

```shell
kubectl apply -f build/agent/resources/
```

> **Cite**
>
> 快速上手 | EdgeMesh (https://edgemesh.netlify.app/zh/guide/#%E6%89%8B%E5%8A%A8%E5%AE%89%E8%A3%85) 全网最全EdgeMesh Q&A手册 - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/585749690)

- **步骤4**: 检验部署结果

```shell
kubectl get all -n kubeedge -o wide
```

> **Warning**
>
> 记得查看所有 `edgemesh-agent` 的信息，排查所有 `fail` 和 `warning` (warning 尽量解决)

运行正常截图

云端 agent——

```
root@cloud:/home/guest/yby/kubeedge/demo# kubectl logs edgemesh-agent-lmdkq -n kubeedge
I1119 16:50:53.405315       1 server.go:56] Version: v1.14.0-dirty
I1119 16:50:53.405478       1 server.go:92] [1] Prepare agent to run
I1119 16:50:53.405860       1 netif.go:96] bridge device edgemesh0 already exists
I1119 16:50:53.406003       1 server.go:96] edgemesh-agent running on CloudMode
I1119 16:50:53.406018       1 server.go:99] [2] New clients
W1119 16:50:53.406038       1 client_config.go:617] Neither --kubeconfig nor --master was specified.  Using the inClusterConfig.  This might not work.
I1119 16:50:53.504784       1 server.go:106] [3] Register beehive modules
W1119 16:50:53.504828       1 module.go:37] Module EdgeDNS is disabled, do not register
I1119 16:50:53.506473       1 server.go:68] Using userspace Proxier.
I1119 16:50:55.208554       1 module.go:34] Module EdgeProxy registered successfully
I1119 16:50:55.214538       1 util.go:273] Listening to docker0 is meaningless, skip it.
I1119 16:50:55.214560       1 util.go:273] Listening to flannel.1 is meaningless, skip it.
I1119 16:50:55.214567       1 util.go:273] Listening to cni0 is meaningless, skip it.
I1119 16:50:55.214574       1 util.go:273] Listening to br-c48d8747b91b is meaningless, skip it.
I1119 16:50:55.214582       1 util.go:273] Listening to edgemesh0 is meaningless, skip it.
I1119 16:50:55.214589       1 util.go:273] Listening to kube-ipvs0 is meaningless, skip it.
I1119 16:50:55.214596       1 util.go:273] Listening to tunl0 is meaningless, skip it.
I1119 16:50:55.607911       1 module.go:181] I'm {12D3KooWP5uK9af8vGaf9oiukDSVPqWG2Rm7dxA5wCvJ11dPSQ9y: [/ip4/127.0.0.1/tcp/20006 /ip4/114.212.81.11/tcp/20006 /ip4/114.212.81.11/tcp/20006]}
I1119 16:50:55.608008       1 module.go:190] Run as a relay node
I1119 16:50:55.608135       1 module.go:203] Bootstrapping the DHT
I1119 16:50:56.106566       1 tunnel.go:80] Starting MDNS discovery service
I1119 16:50:56.106614       1 tunnel.go:93] Starting DHT discovery service
I1119 16:50:56.106784       1 module.go:34] Module EdgeTunnel registered successfully
I1119 16:50:56.106809       1 server.go:112] [4] Cache beehive modules
I1119 16:50:56.106850       1 tunnel.go:493] Starting relay finder
I1119 16:50:56.106857       1 server.go:119] [5] Start all modules
I1119 16:50:56.107005       1 core.go:24] Starting module EdgeProxy
I1119 16:50:56.107166       1 core.go:24] Starting module EdgeTunnel
I1119 16:50:56.107346       1 config.go:135] "Starting endpoints config controller"
I1119 16:50:56.107358       1 config.go:317] "Starting service config controller"
I1119 16:50:56.107405       1 shared_informer.go:240] Waiting for caches to sync for endpoints config
I1119 16:50:56.107405       1 shared_informer.go:240] Waiting for caches to sync for service config
I1119 16:50:56.204568       1 loadbalancer.go:239] "Starting loadBalancer destinationRule controller"
I1119 16:50:56.204591       1 shared_informer.go:240] Waiting for caches to sync for loadBalancer destinationRule
I1119 16:50:56.307505       1 shared_informer.go:247] Caches are synced for endpoints config
I1119 16:50:56.307590       1 shared_informer.go:247] Caches are synced for service config
I1119 16:50:56.404735       1 shared_informer.go:247] Caches are synced for loadBalancer destinationRule
I1119 16:51:21.919273       1 loadbalancer.go:720] Dial legacy network between coredns-7f89b7bc75-hm4dc - {udp cloud.kubeedge 10.244.111.17:53}
I1119 16:51:21.920318       1 tunnel.go:264] Could not find peer edge1.kubeedge in cache, auto generate peer info: {12D3KooWFz1dKY8L3JC8wAY6sJ5MswvPEGKysPCfcaGxFmeH7wkz: []}
I1119 16:51:21.920495       1 tunnel.go:278] New stream between peer edge1.kubeedge: {12D3KooWFz1dKY8L3JC8wAY6sJ5MswvPEGKysPCfcaGxFmeH7wkz: [/ip4/114.212.81.11/tcp/20006/p2p/12D3KooWP5uK9af8vGaf9oiukDSVPqWG2Rm7dxA5wCvJ11dPSQ9y/p2p-circuit]} success
I1119 16:51:21.922740       1 loadbalancer.go:732] Dial libp2p network between hostname-edge-84cb45ccf4-5xbtk - {tcp edge1.kubeedge 172.17.0.3:9376}
I1119 16:51:55.997441       1 loadbalancer.go:720] Dial legacy network between coredns-7f89b7bc75-dwkzb - {udp cloud.kubeedge 10.244.111.16:53}
```

云端和边端不在同一子网记得设置 relayNode

边端 agent——

```
root@cloud:/home/guest/yby/kubeedge/demo# kubectl logs edgemesh-agent-qclz4 -n kubeedge
I1119 16:50:53.062817       1 server.go:56] Version: v1.14.0-dirty
I1119 16:50:53.062943       1 server.go:92] [1] Prepare agent to run
I1119 16:50:53.064487       1 netif.go:96] bridge device edgemesh0 already exists
I1119 16:50:53.064717       1 server.go:96] edgemesh-agent running on EdgeMode
I1119 16:50:53.064795       1 server.go:99] [2] New clients
I1119 16:50:53.066623       1 server.go:106] [3] Register beehive modules
I1119 16:50:53.067654       1 module.go:34] Module EdgeDNS registered successfully
I1119 16:50:53.068192       1 server.go:68] Using userspace Proxier.
I1119 16:50:53.857964       1 module.go:34] Module EdgeProxy registered successfully
I1119 16:50:53.958057       1 util.go:273] Listening to docker0 is meaningless, skip it.
I1119 16:50:53.958559       1 util.go:273] Listening to flannel.1 is meaningless, skip it.
I1119 16:50:53.958939       1 util.go:273] Listening to edgemesh0 is meaningless, skip it.
I1119 16:50:54.160415       1 module.go:181] I'm {12D3KooWFz1dKY8L3JC8wAY6sJ5MswvPEGKysPCfcaGxFmeH7wkz: [/ip4/127.0.0.1/tcp/20006 /ip4/192.168.1.2/tcp/20006]}
I1119 16:50:54.160883       1 module.go:203] Bootstrapping the DHT
I1119 16:50:54.161206       1 tunnel.go:435] [Bootstrap] bootstrapping to 12D3KooWP5uK9af8vGaf9oiukDSVPqWG2Rm7dxA5wCvJ11dPSQ9y
E1119 16:50:54.359013       1 tunnel.go:439] [Bootstrap] failed to bootstrap with {12D3KooWP5uK9af8vGaf9oiukDSVPqWG2Rm7dxA5wCvJ11dPSQ9y: [/ip4/114.212.81.11/tcp/20006]}: failed to dial 12D3KooWP5uK9af8vGaf9oiukDSVPqWG2Rm7dxA5wCvJ11dPSQ9y:
  * [/ip4/114.212.81.11/tcp/20006] dial tcp4 114.212.81.11:20006: connect: connection refused
E1119 16:50:54.359510       1 tunnel.go:450] [Bootstrap] Not all bootstrapDail connected, continue bootstrapDail...
I1119 16:51:04.359848       1 tunnel.go:435] [Bootstrap] bootstrapping to 12D3KooWP5uK9af8vGaf9oiukDSVPqWG2Rm7dxA5wCvJ11dPSQ9y
I1119 16:51:04.465588       1 tunnel.go:445] [Bootstrap] success bootstrapped with {12D3KooWP5uK9af8vGaf9oiukDSVPqWG2Rm7dxA5wCvJ11dPSQ9y: [/ip4/114.212.81.11/tcp/20006]}
I1119 16:51:04.466777       1 tunnel.go:80] Starting MDNS discovery service
I1119 16:51:04.467374       1 tunnel.go:93] Starting DHT discovery service
I1119 16:51:04.467556       1 module.go:34] Module EdgeTunnel registered successfully
I1119 16:51:04.467577       1 server.go:112] [4] Cache beehive modules
I1119 16:51:04.467608       1 server.go:119] [5] Start all modules
I1119 16:51:04.467824       1 core.go:24] Starting module EdgeTunnel
I1119 16:51:04.467951       1 core.go:24] Starting module EdgeDNS
I1119 16:51:04.468011       1 core.go:24] Starting module EdgeProxy
I1119 16:51:04.468250       1 tunnel.go:493] Starting relay finder
I1119 16:51:04.468671       1 dns.go:38] Runs CoreDNS v1.8.0 as a local dns
I1119 16:51:04.473168       1 config.go:317] "Starting service config controller"
I1119 16:51:04.473216       1 shared_informer.go:240] Waiting for caches to sync for service config
I1119 16:51:04.473293       1 config.go:135] "Starting endpoints config controller"
I1119 16:51:04.473308       1 shared_informer.go:240] Waiting for caches to sync for endpoints config
I1119 16:51:04.559317       1 loadbalancer.go:239] "Starting loadBalancer destinationRule controller"
I1119 16:51:04.559359       1 shared_informer.go:240] Waiting for caches to sync for loadBalancer destinationRule
I1119 16:51:04.565261       1 tunnel.go:126] [MDNS] Discovery found peer: {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/ip4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]}
I1119 16:51:04.660357       1 shared_informer.go:247] Caches are synced for loadBalancer destinationRule
I1119 16:51:04.758510       1 shared_informer.go:247] Caches are synced for endpoints config
I1119 16:51:04.758581       1 shared_informer.go:247] Caches are synced for service config
I1119 16:51:04.759018       1 log.go:184] [INFO] plugin/reload: Running configuration MD5 = 4ba233d14e39c5d57b4b7aa950136711
.:53 on 169.254.96.16
I1119 16:51:04.791041       1 log.go:184] [INFO] 169.254.96.16:45633 - 14210 "HINFO IN 3648482678301049553.5333225444095395442. udp 57 false 512" NXDOMAIN qr,rd,ra 57 0.031587923s
I1119 16:51:05.266780       1 tunnel.go:205] Discovery service got a new stream from {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/ip4/192.168.1.4/tcp/20006]}
I1119 16:51:05.266843       1 tunnel.go:205] Discovery service got a new stream from {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/ip4/192.168.1.4/tcp/20006]}
I1119 16:51:05.267081       1 tunnel.go:234] [DHT] Discovery from edge2 : {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/ip4/192.168.1.4/tcp/20006]}
I1119 16:51:05.267240       1 tunnel.go:234] [MDNS] Discovery from edge2 : {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/ip4/192.168.1.4/tcp/20006]}
I1119 16:51:05.267566       1 tunnel.go:156] [MDNS] New stream between peer {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/ip4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]}
```

Sedna

安装

SHELL

```
#直接进入之前准备好的文件目录运行
SEDNA_ACTION=create bash - offline-install.sh

# 卸载
SEDNA_ACTION=delete bash - offline-install.sh
```

正常运行情况

如图展示正常运行情况：`lc-edge`：

```
root@cloud:/home/guest/yby/test# kubectl get pods -n sedna -owide
NAME                  READY   STATUS    RESTARTS   AGE   IP              NODE             NOMINATED NODE   READINESS GATES
gm-5bb9c898d6-tlbqh   1/1     Running   0          13s   10.244.111.55   cloud.kubeedge   <none>           <none>
kb-6b7897c89-xdwlv    1/1     Running   0          13s   10.244.111.54   cloud.kubeedge   <none>           <none>
lc-6rc65              1/1     Running   0          12s   192.168.1.2     edge1.kubeedge   <none>           <none>
lc-xx9w7              1/1     Running   0          12s   192.168.1.4     edge2            <none>           <none>
lc-zznkh              1/1     Running   0          12s   114.212.81.11   cloud.kubeedge   <none>           <none>
root@cloud:/home/guest/yby/test# kubectl logs lc-6rc65 -n sedna
I1120 03:36:54.260967       1 server.go:120] manager dataset is started
I1120 03:36:54.261053       1 server.go:120] manager model is started
I1120 03:36:54.261061       1 websocket.go:176] client starts to connect global manager(address: gm.sedna:9000)
I1120 03:36:54.261099       1 server.go:120] manager jointinferenceservice is started
I1120 03:36:54.261128       1 server.go:120] manager federatedlearningjob is started
I1120 03:36:54.261151       1 server.go:120] manager incrementallearningjob is started
I1120 03:36:54.261168       1 server.go:120] manager lifelonglearningjob is started
I1120 03:36:54.261386       1 server.go:140] server binds port 9100 successfully
E1120 03:36:54.285924       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: unexpected EOF
I1120 03:36:59.294292       1 websocket.go:187] websocket connects global manager(address: gm.sedna:9000) successful
root@cloud:/home/guest/yby/test# kubectl logs lc-xx9w7 -n sedna
I1120 03:36:54.251836       1 server.go:120] manager dataset is started
I1120 03:36:54.251958       1 server.go:120] manager model is started
I1120 03:36:54.251993       1 server.go:120] manager jointinferenceservice is started
I1120 03:36:54.252038       1 server.go:120] manager federatedlearningjob is started
I1120 03:36:54.252058       1 server.go:120] manager incrementallearningjob is started
I1120 03:36:54.252078       1 server.go:120] manager lifelonglearningjob is started
I1120 03:36:54.252297       1 server.go:140] server binds port 9100 successfully
I1120 03:36:54.252745       1 websocket.go:176] client starts to connect global manager(address: gm.sedna:9000)
root@cloud:/home/guest/yby/test# kubectl logs -n sedna lc-zznkh
I1120 03:36:54.223544       1 websocket.go:176] client starts to connect global manager(address: gm.sedna:9000)
I1120 03:36:54.224481       1 server.go:120] manager dataset is started
I1120 03:36:54.224839       1 server.go:120] manager model is started
I1120 03:36:54.224895       1 server.go:120] manager jointinferenceservice is started
I1120 03:36:54.224934       1 server.go:120] manager federatedlearningjob is started
I1120 03:36:54.224954       1 server.go:120] manager incrementallearningjob is started
I1120 03:36:54.224968       1 server.go:120] manager lifelonglearningjob is started
I1120 03:36:54.225126       1 server.go:140] server binds port 9100 successfully
E1120 03:36:54.227205       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: unexpected EOF
I1120 03:36:59.230176       1 websocket.go:187] websocket connects global manager(address: gm.sedna:9000) successful
```

`gm`

```
I1120 03:04:32.195545       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:04:32.195771       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:04:32.208604       1 worker.go:138] pod helmet-detection-inference-example-cloud-gbrdd is created successfully for sedna.io/v1alpha1, Kind=JointInference default/helmet-detectio
n-inference-example
I1120 03:04:32.216953       1 worker.go:182] Service helmet-detection-inference-example-cloud is created successfully for JointInference default/helmet-detection-inference-example
I1120 03:04:32.224007       1 worker.go:138] pod helmet-detection-inference-example-edge-w8l65 is created successfully for sedna.io/v1alpha1, Kind=JointInference default/helmet-detection
-inference-example
I1120 03:04:32.235849       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:04:32.235989       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:05:25.308864       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:05:25.309031       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:06:19.580882       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:06:19.581109       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:07:13.852845       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:07:13.853102       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:08:08.124739       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:08:08.125024       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:09:02.396719       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:09:02.396962       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:09:56.668433       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:09:56.668647       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:10:50.940429       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:10:50.940648       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:11:45.212041       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:11:45.212217       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:12:39.484151       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:12:39.484419       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:12:55.201722       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:12:55.201922       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:13:29.926016       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:13:29.926161       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:13:29.935630       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:13:29.935732       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:13:33.756097       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:13:33.756355       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:14:28.028129       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:14:28.028416       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:14:57.556033       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:14:57.556175       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:15:03.428943       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:15:03.429115       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:15:03.462206       1 worker.go:138] pod helmet-detection-inference-example-cloud-t8lpz is created successfully for sedna.io/v1alpha1, Kind=JointInference default/helmet-detectio
n-inference-example
I1120 03:15:03.490007       1 worker.go:182] Service helmet-detection-inference-example-cloud is created successfully for JointInference default/helmet-detection-inference-example
I1120 03:15:03.502067       1 worker.go:138] pod helmet-detection-inference-example-edge-g8rqt is created successfully for sedna.io/v1alpha1, Kind=JointInference default/helmet-detection
-inference-example
I1120 03:15:03.518628       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:15:03.518747       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
I1120 03:15:22.299897       1 context.go:89] sending jointinferenceservice default/helmet-detection-inference-example to node(edge2)
I1120 03:15:22.300115       1 context.go:184] write key jointinferenceservice/default/helmet-detection-inference-example to node edge2 successfully
```

目前存在问题：连接具有偶然性?

```
root@cloud:/home/guest/yby/test# kubectl logs lc-c6n56 -n sedna
I1120 06:14:37.795430       1 websocket.go:176] client starts to connect global manager(address: gm.sedna:9000)
I1120 06:14:37.796235       1 server.go:120] manager dataset is started
I1120 06:14:37.796316       1 server.go:120] manager model is started
I1120 06:14:37.796422       1 server.go:120] manager jointinferenceservice is started
I1120 06:14:37.796479       1 server.go:120] manager federatedlearningjob is started
I1120 06:14:37.796500       1 server.go:120] manager incrementallearningjob is started
I1120 06:14:37.796519       1 server.go:120] manager lifelonglearningjob is started
I1120 06:14:37.796761       1 server.go:140] server binds port 9100 successfully
E1120 06:14:41.688917       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:59296->10.107.87.64:9000: read: connect
ion reset by peer
E1120 06:14:50.583093       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:59310->10.107.87.64:9000: read: connect
ion reset by peer
E1120 06:14:59.477002       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:59316->10.107.87.64:9000: read: connect
ion reset by peer
E1120 06:15:08.360409       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:59330->10.107.87.64:9000: read: connect
ion reset by peer
E1120 06:15:58.361041       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: dial tcp 10.99.206.217:9000: i/o timeout
E1120 06:16:03.361193       1 websocket.go:200] max retry count reached when connecting global manager(address: gm.sedna:9000)
I1120 06:16:03.361238       1 websocket.go:176] client starts to connect global manager(address: gm.sedna:9000)
E1120 06:16:48.361614       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: dial tcp 10.99.206.217:9000: i/o timeout
E1120 06:16:57.183128       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:42128->10.99.206.217:9000: read: connec
tion reset by peer
E1120 06:17:05.982603       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:42140->10.99.206.217:9000: read: connec
tion reset by peer
E1120 06:17:14.857423       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:42142->10.99.206.217:9000: read: connec
tion reset by peer
E1120 06:17:23.677812       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:42150->10.99.206.217:9000: read: connec
tion reset by peer
E1120 06:17:28.678133       1 websocket.go:200] max retry count reached when connecting global manager(address: gm.sedna:9000)
I1120 06:17:28.678203       1 websocket.go:176] client starts to connect global manager(address: gm.sedna:9000)
E1120 06:17:32.677803       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:42160->10.99.206.217:9000: read: connec
tion reset by peer
E1120 06:17:41.577052       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:42166->10.99.206.217:9000: read: connec
tion reset by peer
E1120 06:17:50.494374       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:42182->10.99.206.217:9000: read: connec
tion reset by peer
E1120 06:17:59.282041       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:42188->10.99.206.217:9000: read: connec
tion reset by peer
E1120 06:18:08.184128       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:42200->10.99.206.217:9000: read: connec
tion reset by peer
E1120 06:18:13.184324       1 websocket.go:200] max retry count reached when connecting global manager(address: gm.sedna:9000)
I1120 06:18:13.184395       1 websocket.go:176] client starts to connect global manager(address: gm.sedna:9000)
E1120 06:18:17.087979       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:42214->10.99.206.217:9000: read: connec
tion reset by peer
E1120 06:18:25.976963       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read tcp 192.168.1.4:42216->10.99.206.217:9000: read: connec
tion reset by peer
I1120 06:18:31.080876       1 websocket.go:187] websocket connects global manager(address: gm.sedna:9000) successful
root@cloud:/home/guest/yby/test#
```

Demo

跑官方给的 Joint inference Using Joint Inference Service in Helmet Detection Scenario — Sedna 0.4.1 documentation (https://sedna.readthedocs.io/en/latest/examples/joint_inference/helmet_detection_inference/README.html)

准备模型

- 边缘节点准备小模型

```shell
mkdir -p /data/little-model
cd /data/little-model
wget https://kubeedge.obs.cn-north-1.myhuaweicloud.com/examples/helmet-detection-inference/little-model.tar.gz
tar -zxvf little-model.tar.gz
```

- 云准备大模型

```shell
mkdir -p /data/big-model
cd /data/big-model
wget https://kubeedge.obs.cn-north-1.myhuaweicloud.com/examples/helmet-detection-inference/big-model.tar.gz
tar -zxvf big-model.tar.gz
```

拉取镜像

注意看镜像适用的架构

```shell
# 边
docker pull kubeedge/sedna-example-joint-inference-helmet-detection-little:v0.3.0

# 云
docker pull kubeedge/sedna-example-joint-inference-helmet-detection-big:v0.3.0
```

部署服务

- 为云准备大模型

```shell
kubectl create -f - <<EOF
apiVersion: sedna.io/v1alpha1
kind:  Model
metadata:
  name: helmet-detection-inference-big-model
  namespace: default
spec:
  url: "/data/big-model/yolov3_darknet.pb"
  format: "pb"
EOF
```

- 为边准备小模型

```shell
kubectl create -f - <<EOF
apiVersion: sedna.io/v1alpha1
kind: Model
metadata:
  name: helmet-detection-inference-little-model
  namespace: default
spec:
  url: "/data/little-model/yolov3_resnet18.pb"
  format: "pb"
EOF
```

- 边端创建输出目录
  记得提前创建，否则会启 pod 时会一直 pending, edgecore 会有报错内容，大致是 volumemounts 失败

```shell
mkdir -p /joint_inference/output
```

- 准备 Service 的 yaml 文件

YAML

```
CLOUD_NODE="cloud-node-name"
EDGE_NODE="edge-node-name"


kubectl create -f - <<EOF
apiVersion: sedna.io/v1alpha1
kind: JointInferenceService
metadata:
  name: helmet-detection-inference-example
  namespace: default
spec:
  edgeWorker:
    model:
      name: "helmet-detection-inference-little-model"
    hardExampleMining:
      name: "IBT"
      parameters:
        - key: "threshold_img"
          value: "0.9"
        - key: "threshold_box"
          value: "0.9"
    template:
      spec:
        nodeName: $EDGE_NODE
        hostNetwork: true
        dnsPolicy: ClusterFirstWithHostNet
        containers:
        - image: kubeedge/sedna-example-joint-inference-helmet-detection-
little:v0.3.0
          imagePullPolicy: IfNotPresent
          name:  little-model
          env:  # user defined environments
          - name: input_shape
            value: "416,736"
          - name: "video_url"
            value: "rtsp://localhost/video"
          - name: "all_examples_inference_output"
            value: "/data/output"
          - name: "hard_example_cloud_inference_output"
            value: "/data/hard_example_cloud_inference_output"
          - name: "hard_example_edge_inference_output"
            value: "/data/hard_example_edge_inference_output"
          resources:  # user defined resources
            requests:
              memory: 64M
```

```
              cpu: 100m
            limits:
              memory: 2Gi
          volumeMounts:
            - name: outputdir
              mountPath: /data/
        volumes:   # user defined volumes
          - name: outputdir
            hostPath:
              # user must create the directory in host
              path: /joint_inference/output
              type: Directory

  cloudWorker:
    model:
      name: "helmet-detection-inference-big-model"
    template:
      spec:
        nodeName: $CLOUD_NODE
        hostNetwork: true
        dnsPolicy: ClusterFirstWithHostNet
        containers:
          - image: kubeedge/sedna-example-joint-inference-helmet-
detection-big:v0.3.0
            name:  big-model
            imagePullPolicy: IfNotPresent
            env:  # user defined environments
              - name: "input_shape"
                value: "544,544"
            resources:  # user defined resources
              requests:
                memory: 2Gi
EOF
```

Tip

1. 删除重新部署的话除了 `delete -f` 上面的 yaml 文件，还要手动删除一些 pod 和 **svc**。否则可能会没有效果，此时 edgecore、lc 可能会有相关报错
2. 如果删清了还是没有对应的 pod，注意查看 model 和 service 的 metadata 下的 name 是否对应...

模拟视频流（边）

- step1: install the open source video streaming server EasyDarwin (https://github.com/EasyDarwin/EasyDarwin/tree/dev) .
- step2: start EasyDarwin server.

- step3: download video (https://kubeedge.obs.cn-north-1.myhuaweicloud.com/examples/helmet-detection-inference/video.tar.gz) .
- step 4: push a video stream to the url (e.g., `rtsp://localhost/video`) that the inference service can connect.

Cite

`arm` 架构下的 easydarwin Ubuntu1604交叉编译全志T7开发板ARM版Easydarwin - 食铁兽 - FFmpeg/OpenCV/Qt/Presagis/VAPSXT/VegaPrime/STAGE/TerraVista/Ondulus/Creator/HeliSIM/FlightSIM/CDB/V5D/VxWorks/Wayland/破解入门技术分享 (feater.top) (https://feater.top/linux/cross-compile-easydarwin-under-ubuntu-t7-arm)

SHELL

```
wget
https://github.com/EasyDarwin/EasyDarwin/releases/download/v8.1.0/EasyDarwin-linux-8.1.0-1901141151.tar.gz
tar -zxvf EasyDarwin-linux-8.1.0-1901141151.tar.gz
cd EasyDarwin-linux-8.1.0-1901141151
./start.sh

mkdir -p /data/video
cd /data/video
wget https://kubeedge.obs.cn-north-1.myhuaweicloud.com/examples/helmet-detection-inference/video.tar.gz
tar -zxvf video.tar.gz

ffmpeg -re -i /data/video/video.mp4 -vcodec libx264 -f rtsp
rtsp://localhost/video
```

```
root@node:/joint_inference# kubectl get pods
NAME                                              READY   STATUS    RESTARTS   AGE
helmet-detection-inference-example-cloud-nrpjc    1/1     Running   0          61m
helmet-detection-inference-example-edge-r5pcc     1/1     Running   0          61m
root@node:/joint_inference#
```

Tip

如果 pod 出现 `CrashLoopBackOff` 的状态的话可借鉴 7 张图解 CrashLoopBackOff，如何发现问题并解决它？ - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/558957039)

查看结果

查看/joint_inference/output（即上述创建的输出目录）

```
root@node:/joint_inference/output# ll
总用量 20
drwxr-xr-x 5 root root 4096 11月   5 19:29 ./
drwxr-xr-x 3 root root 4096 11月   5 19:24 ../
drwxr-xr-x 2 root root 4096 11月   5 21:05 hard_example_cloud_inference_output/
drwxr-xr-x 2 root root 4096 11月   5 21:05 hard_example_edge_inference_output/
drwxr-xr-x 2 root root 4096 11月   5 21:05 output/
root@node:/joint_inference/output#
```

云端 pod 的日志如下，说明云边通信成功：

```
[2023-11-20 07:57:34,983] base.py(58) [INFO] - Start sedna server over http://10.244.111.6:5000
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://10.244.111.6:5000 (Press CTRL+C to quit)
INFO:     114.212.81.11:48376 - "POST /sedna/predict HTTP/1.1" 200 OK
INFO:     114.212.81.11:39500 - "POST /sedna/predict HTTP/1.1" 200 OK
INFO:     114.212.81.11:47564 - "POST /sedna/predict HTTP/1.1" 200 OK
```

GPU 支持

前期尝试

k8s 使用 GPU 需要 nvidia-device-plugin 插件支持

```shell
kubectl create -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.7.3/nvidia-device-plugin.yml
```

但是会发现问题：

```
root@cloud:/home/guest/yby/kubeedge/gpu# kubectl logs nvidia-device-plugin-daemonset-rcx2x -n kube-system
exec /usr/bin/nvidia-device-plugin: exec format error
root@cloud:/home/guest/yby/kubeedge/gpu# kubectl get pods --all-namespaces -owide | grep nvidia
kube-system   nvidia-device-plugin-daemonset-5m74z       0/1     Error     2          51s   172.17.0.2      edge2           <none>           <none>
kube-system   nvidia-device-plugin-daemonset-9pq7q       0/1     Error     2          51s   172.17.0.2      edge3           <none>           <none>
kube-system   nvidia-device-plugin-daemonset-c8z2b       1/1     Running   0          51s   10.244.111.23   cloud.kubeedge  <none>           <none>
kube-system   nvidia-device-plugin-daemonset-rcx2x       0/1     Error     2          51s   172.17.0.2      edge1           <none>           <none>
```

这个问题也遇见了很多次了，大概就是架构问题导致无法运行，需要重新编译 docker

```shell
git clone -b 1.0.0-beta6 https://github.com/nvidia/k8s-device-plugin.git

cd k8s-device-plugin

wget https://labs.windriver.com/downloads/0001-arm64-add-support-for-arm64-architectures.patch

wget https://labs.windriver.com/downloads/0002-nvidia-Add-support-for-tegra-boards.patch

wget https://labs.windriver.com/downloads/0003-main-Add-support-for-tegra-boards.patch

git am 000*.patch

应用: arm64: add support for arm64 architectures
应用: nvidia: Add support for tegra boards
应用: main: Add support for tegra boards

docker buildx  build --platform linux/arm64 -t adayoung/k8s-device-plugin:v0.1 -f docker/arm64/Dockerfile.ubuntu16.04 .
```

出现问题:

```
root@cloud:/home/guest/yby/kubeedge/multiedge/test-yaml/gpu# kubectl get pods -A
NAMESPACE     NAME                                        READY   STATUS              RESTARTS   AGE
kube-system   calico-kube-controllers-577f77cb5c-v2n6z    1/1     Running             16         47d
kube-system   calico-node-9ntpv                           1/1     Running             3          47d
kube-system   coredns-7f89b7bc75-dwkzb                    1/1     Running             3          47d
kube-system   coredns-7f89b7bc75-hm4dc                    1/1     Running             3          47d
kube-system   etcd-cloud.kubeedge                         1/1     Running             4          47d
kube-system   kube-apiserver-cloud.kubeedge               1/1     Running             4          47d
kube-system   kube-controller-manager-cloud.kubeedge      1/1     Running             5          47d
kube-system   kube-proxy-b5pq9                            1/1     Running             4          19d
kube-system   kube-scheduler-cloud.kubeedge               1/1     Running             4          47d
kube-system   metrics-server-6fb454cdb7-llnvw             1/1     Running             3          21d
kube-system   nvidia-device-plugin-daemonset-edge-wfzmt   0/1     ContainerCannotRun  6          15m
kubeedge      edgemesh-agent-7g4lz                        1/1     Running             4          15d
kubeedge      edgemesh-agent-xqh79                        1/1     Running             1          19d
kubeedge      edgemesh-agent-xwn65                        1/1     Running             2          23h
kubeedge      edgemesh-agent-z7ct7                        1/1     Running             0          23h
root@cloud:/home/guest/yby/kubeedge/multiedge/test-yaml/gpu#
```

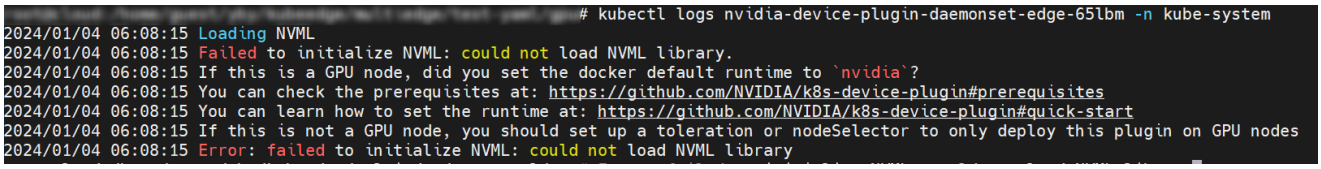
```
7531a55d5d2446ada1e5e8eba212b8
      Exit Code:  127
      Started:    Wed, 03 Jan 2024 17:08:56 +0800
      Finished:   Wed, 03 Jan 2024 17:08:56 +0800
    Last State:   Terminated
      Reason:     ContainerCannotRun
      Message:    failed to create task for container: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: error during container init: error running hook #0: error running hook: exit status 1, stdout: src: /usr/lib/libvisionworks_sfm.so, src_lnk: libvisionworks_sfm.so.0.90, dst: /data/docker/overlay2/248bc260a9329d263e3d6e74d67531a55d5d2446ada1e5e8eba212b8b44c665f/merged/usr/lib/libvisionworks_sfm.so, dst_lnk: libvisionworks_sfm.so.0.90
src: /usr/lib/libvisionworks_sfm.so.0.90, src_lnk: libvisionworks_sfm.so.0.90.4, dst: /data/docker/overlay2/248bc260a9329d263e3d6e74d67531a55d5d2446ada1e5e8eba212b8b44c665f/merged/usr/lib/libvisionworks_sfm.so.0.90, dst_lnk: libvisionworks_sfm.so.0.90.4
src: /usr/lib/libvisionworks.so, src_lnk: libvisionworks.so.1.6, dst: /data/docker/overlay2/248bc260a9329d263e3d6e74d67531a55d5d2446ada1e5e8eba212b8b44c665f/merged/usr/lib/libvisionworks.so, dst_lnk: libvisionworks.so.1.6
src: /usr/lib/libvisionworks_tracking.so, src_lnk: libvisionworks_tracking.so.0.88, dst: /data/docker/overlay2/248bc260a9329d263e3d6e74d67531a55d5d2446ada1e5e8eba212b8b44c665f/merged/usr/lib/libvisionworks_tracking.so, dst_lnk: libvisionworks_tracking.so.0.88
```

```
, stderr: nvidia-container-cli: ldcache error: open failed: /sbin/ldconfig.real: no such file or directory: unknown.
ERRO[0000] error waiting for container:
```

此问题的解决：只需要更新动态库配置文件和将它报错的路径链接过去

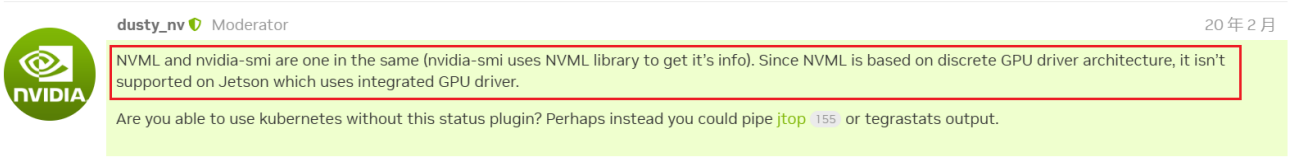
```shell
root@edge2:/software# ldconfig
root@edge2:/software# whereis /sbin/ldconfig
ldconfig: /sbin/ldconfig
root@edge2:/software# ln -s /sbin/ldconfig /sbin/ldconfig.real
root@edge2:/software# sudo systemctl daemon-reload
root@edge2:/software# sudo systemctl restart docker
```

然而问题只会一个接一个的出现：




```shell
root@cloud:/home/guest/yby/kubeedge/multiedge/test-yaml/gpu# kubectl logs
nvidia-device-plugin-daemonset-edge-65lbm -n kube-system
2024/01/04 06:08:15 Loading NVML
2024/01/04 06:08:15 Failed to initialize NVML: could not load NVML
library.
2024/01/04 06:08:15 If this is a GPU node, did you set the docker default
runtime to `nvidia`?
2024/01/04 06:08:15 You can check the prerequisites at:
https://github.com/NVIDIA/k8s-device-plugin#prerequisites
2024/01/04 06:08:15 You can learn how to set the runtime at:
https://github.com/NVIDIA/k8s-device-plugin#quick-start
2024/01/04 06:08:15 If this is not a GPU node, you should set up a
toleration or nodeSelector to only deploy this plugin on GPU nodes
2024/01/04 06:08:15 Error: failed to initialize NVML: could not load NVML
library
```

> **Quote**
>
> nvidia-smi installation on Jetson TX2 - Jetson & Embedded Systems / Jetson TX2 - NVIDIA Developer Forums (https://forums.developer.nvidia.com/t/nvidia-smi-installation-on-jetson-tx2/111495)



官方回复 jetson 系列不能用 NVML... 但是这个插件开发者兼容了 jetson 板子!

> **Quote**
>
> Add support for arm64 and Jetson boards. (!20) · 合并请求 · nvidia / kubernetes / device-plugin · GitLab (https://gitlab.com/nvidia/kubernetes/device-plugin/-/merge_requests/20) NVIDIA/k8s-device-plugin at v0.13.0 (github.com) (https://github.com/NVIDIA/k8s-device-plugin/tree/v0.13.0)

部署

运行下面命令即可:

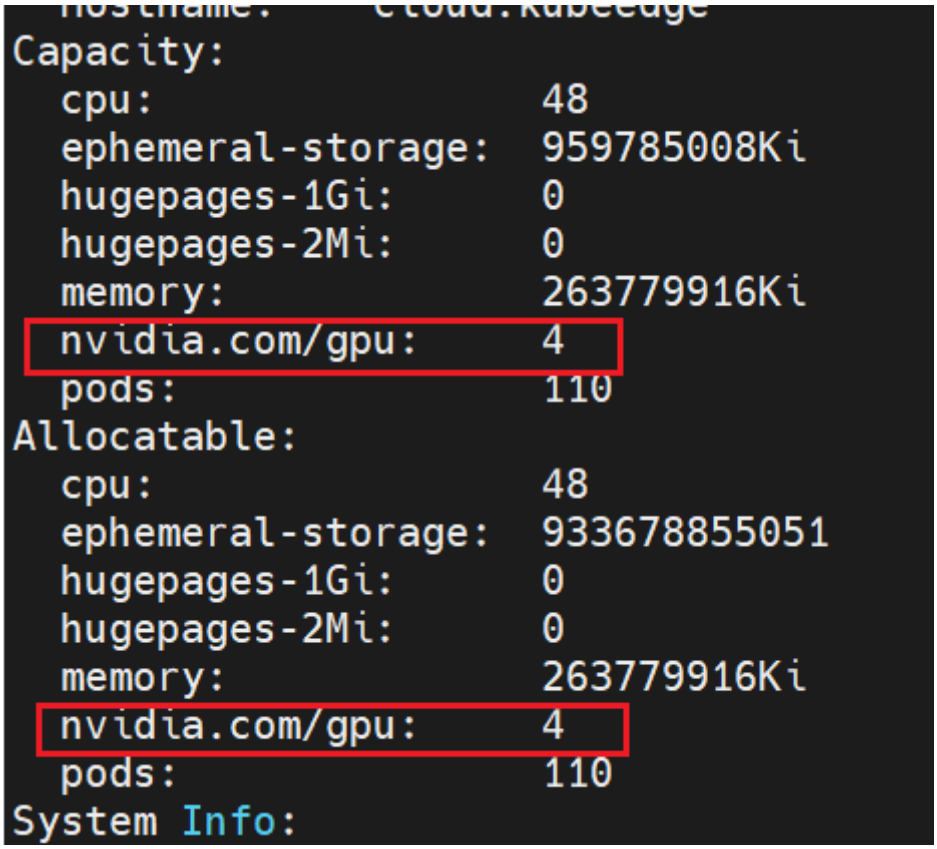
```shell
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.13.0/nvidia-device-plugin.yml
```

> **Hint**
>
> 在边端 `vim /etc/kubeedge/config/edgecore.yaml` 将修改使得 `devicePluginEnabled: true`。本身这个插件是提供给 `k8s` 使用的，本地节点采用的 kubelet 进行管理，但是 kubeedge 采用 edgecore，打开 devicePluginEnabled 才可以

正常运行

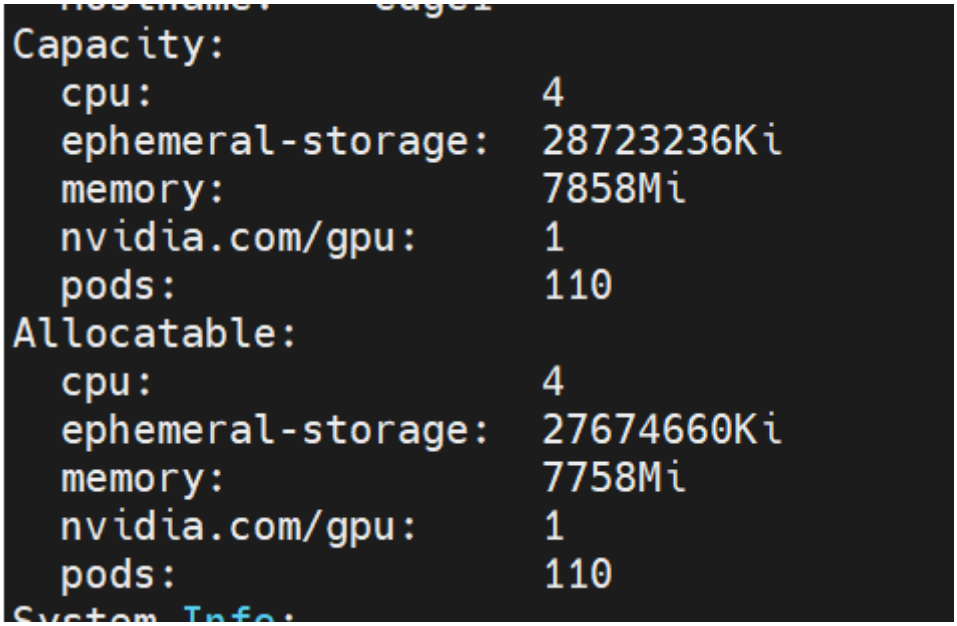
- **云端**:

```
Running with config:
{
  "version": "v1",
  "flags": {
    "migStrategy": "none",
    "failOnInitError": false,
    "nvidiaDriverRoot": "/",
    "gdsEnabled": false,
    "mofedEnabled": false,
    "plugin": {
      "passDeviceSpecs": false,
      "deviceListStrategy": "envvar",
      "deviceIDStrategy": "uuid"
    }
  },
  "resources": {
    "gpus": [
      {
        "pattern": "*",
        "name": "nvidia.com/gpu"
      }
    ]
  },
  "sharing": {
    "timeSlicing": {}
  }
}
2024/01/04 06:53:28 Retreiving plugins.
2024/01/04 06:53:28 Detected NVML platform: found NVML library
2024/01/04 06:53:28 Detected non-Tegra platform: /sys/devices/soc0/family file not found
2024/01/04 06:53:28 Starting GRPC server for 'nvidia.com/gpu'
2024/01/04 06:53:28 Starting to serve 'nvidia.com/gpu' on /var/lib/kubelet/device-plugins/nvidia-gpu.sock
2024/01/04 06:53:28 Registered device plugin for 'nvidia.com/gpu' with Kubelet
```



- 边端（jetson）

```
Running with config:
{
  "version": "v1",
  "flags": {
    "migStrategy": "none",
    "failOnInitError": false,
    "nvidiaDriverRoot": "/",
    "gdsEnabled": false,
    "mofedEnabled": false,
    "plugin": {
      "passDeviceSpecs": false,
      "deviceListStrategy": "envvar",
      "deviceIDStrategy": "uuid"
    }
  },
  "resources": {
    "gpus": [
      {
        "pattern": "*",
        "name": "nvidia.com/gpu"
      }
    ]
  },
  "sharing": {
    "timeSlicing": {}
  }
}
2024/01/04 07:15:37 Retreiving plugins.
2024/01/04 07:15:37 Detected non-NVML platform: could not load NVML: libnvidia-ml.so.1: cannot open shared object file: No such file or directory
2024/01/04 07:15:37 Detected Tegra platform: /etc/nv_tegra_release found
2024/01/04 07:15:37 Starting GRPC server for 'nvidia.com/gpu'
2024/01/04 07:15:37 Starting to serve 'nvidia.com/gpu' on /var/lib/kubelet/device-plugins/nvidia-gpu.sock
2024/01/04 07:15:37 Registered device plugin for 'nvidia.com/gpu' with Kubelet
```



SHELL

```
kubectl run -i -t nvidia --image=jitteam/devicequery
```

```
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA Tegra X2"
  CUDA Driver Version / Runtime Version          10.2 / 10.0
  CUDA Capability Major/Minor version number:    6.2
  Total amount of global memory:                 7858 MBytes (8239902720 bytes)
  ( 2) Multiprocessors, (128) CUDA Cores/MP:     256 CUDA Cores
  GPU Max Clock rate:                            1300 MHz (1.30 GHz)
  Memory Clock rate:                             1300 Mhz
  Memory Bus Width:                              128-bit
  L2 Cache Size:                                 524288 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 32768
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            Yes
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >
```

可能出现的问题

是 GPU 节点但是找不到 gpu 资源

主要针对的是服务器的情况，可以使用 `nvidia-smi` 查看显卡情况。

> **Quote**
>
> Installing the NVIDIA Container Toolkit — NVIDIA Container Toolkit 1.14.3 documentation (https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html#configuration)

按上述内容进行配置，然后还需要 `vim /etc/docker/daemon.json`，添加 default-runtime。按照 quote 的内容设置后会有“runtimes”但是 default-runtime 不会设置，可能会导致找不到 GPU 资源

```
{
    "exec-opts": [
        "native.cgroupdriver=systemd"
    ],
    "registry-mirrors": [
        "https://b9pmyelo.mirror.aliyuncs.com"
    ],
    "default-runtime": "nvidia",
    "runtimes": {
        "nvidia": {
            "args": [],
            "path": "nvidia-container-runtime"
        }
    }
}
```

jeston 系列板子问题

理论上 `k8s-device-plugin` 已经支持了 tegra 即 jetson 系列板子，会在查看 GPU 之前判断是否是 tegra 架构，如果是则采用 tegra 下查看 GPU 的方式（原因在 GPU 支持里 quote 过了），但是很奇怪的是明明是 tegra 的架构却没有检测到：

SHELL

```
2024/01/04 07:43:58 Retreiving plugins.
2024/01/04 07:43:58 Detected non-NVML platform: could not load NVML:
libnvidia-ml.so.1: cannot open shared object file: No such file or
directory
2024/01/04 07:43:58 Detected non-Tegra platform: /sys/devices/soc0/family
file not found
2024/01/04 07:43:58 Incompatible platform detected
2024/01/04 07:43:58 If this is a GPU node, did you configure the NVIDIA
Container Toolkit?
2024/01/04 07:43:58 You can check the prerequisites at:
https://github.com/NVIDIA/k8s-device-plugin#prerequisites
2024/01/04 07:43:58 You can learn how to set the runtime at:
https://github.com/NVIDIA/k8s-device-plugin#quick-start
2024/01/04 07:43:58 If this is not a GPU node, you should set up a
toleration or nodeSelector to only deploy this plugin on GPU nodes
2024/01/04 07:43:58 No devices found. Waiting indefinitely.
```

```
2024/01/04 07:43:58 Retreiving plugins.
2024/01/04 07:43:58 Detected non-NVML platform: could not load NVML: libnvidia-ml.so.1: cannot open shared object file: No such file or directory
2024/01/04 07:43:58 Detected non-Tegra platform: /sys/devices/soc0/family file not found
2024/01/04 07:43:58 Incompatible platform detected
2024/01/04 07:43:58 If this is a GPU node, did you configure the NVIDIA Container Toolkit?
2024/01/04 07:43:58 You can check the prerequisites at: https://github.com/NVIDIA/k8s-device-plugin#prerequisites
2024/01/04 07:43:58 You can learn how to set the runtime at: https://github.com/NVIDIA/k8s-device-plugin#quick-start
2024/01/04 07:43:58 If this is not a GPU node, you should set up a toleration or nodeSelector to only deploy this plugin on GPU nodes
2024/01/04 07:43:58 No devices found. Waiting indefinitely.
```

SHELL

```
$ dpkg -l '*nvidia*'
```

```
nvidia@edge2:~$ dpkg -l '*nvidia*'
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                                  Version                 Architecture            Description
+++-=====================================-=======================-=======================-=============================================
un  libgldispatch0-nvidia                 <none>                  <none>                  (no description available)
ii  libnvidia-container-tools             1.7.0-1                 arm64                   NVIDIA container runtime library (command-line t
ii  libnvidia-container0:arm64            0.10.0+jetpack          arm64                   NVIDIA container runtime library
ii  libnvidia-container1:arm64            1.7.0-1                 arm64                   NVIDIA container runtime library
un  nvidia-304                            <none>                  <none>                  (no description available)
un  nvidia-340                            <none>                  <none>                  (no description available)
un  nvidia-384                            <none>                  <none>                  (no description available)
un  nvidia-common                         <none>                  <none>                  (no description available)
ii  nvidia-container-csv-cuda             10.2.460-1              arm64                   Jetpack CUDA CSV file
ii  nvidia-container-csv-cudnn            8.2.1.32-1+cuda10.2     arm64                   Jetpack CUDNN CSV file
ii  nvidia-container-csv-tensorrt         8.2                     arm64                   Jetpack TensorRT CSV file
ii  nvidia-container-csv-visionworks      1.6.0.501               arm64                   Jetpack VisionWorks CSV file
ii  nvidia-container-runtime              3.7.0-1                 all                     NVIDIA container runtime
un  nvidia-container-runtime-hook         <none>                  <none>                  (no description available)
ii  nvidia-container-toolkit              1.7.0-1                 arm64                   NVIDIA container runtime hook
un  nvidia-cuda-dev                       <none>                  <none>                  (no description available)
ii  nvidia-l4t-3d-core                    32.7.4-20230608211515   arm64                   NVIDIA GL EGL Package
ii  nvidia-l4t-apt-source                 32.7.4-20230608211515   arm64                   NVIDIA L4T apt source list debian package
ii  nvidia-l4t-bootloader                 32.7.4-20230608211515   arm64                   NVIDIA Bootloader Package
ii  nvidia-l4t-camera                     32.7.4-20230608211515   arm64                   NVIDIA Camera Package
un  nvidia-l4t-ccp-t186ref                <none>                  <none>                  (no description available)
ii  nvidia-l4t-configs                    32.7.4-20230608211515   arm64                   NVIDIA configs debian package
ii  nvidia-l4t-core                       32.7.4-20230608211515   arm64                   NVIDIA Core Package
ii  nvidia-l4t-cuda                       32.7.4-20230608211515   arm64                   NVIDIA CUDA Package
ii  nvidia-l4t-firmware                   32.7.4-20230608211515   arm64                   NVIDIA Firmware Package
ii  nvidia-l4t-gputools                   32.7.4-20230608211515   arm64                   NVIDIA dgpu helper Package
ii  nvidia-l4t-graphics-demos             32.7.4-20230608211515   arm64                   NVIDIA graphics demo applications
ii  nvidia-l4t-gstreamer                  32.7.4-20230608211515   arm64                   NVIDIA GST Application files
ii  nvidia-l4t-init                       32.7.4-20230608211515   arm64                   NVIDIA Init debian package
ii  nvidia-l4t-initrd                     32.7.4-20230608211515   arm64                   NVIDIA initrd debian package
```

elezar commented on Feb 8, 2023 — Member

Note that looking at the initial logs that you provided you may have been using `v1.7.0` of the NVIDIA Container Toolkit. This is quite an old version and we greatly improved our support for Tegra-based systems with the `v1.10.0` release. It should also be noted that in order to use the GPU Device Plugin on Tegra-based systems (specifically targetting the integrated GPUs) at least `v1.11.0` of the NVIDIA Container Toolkit is required.

There are no Tegra-specific changes in the `v1.12.0` release, so using the `v1.11.0` release should be sufficient in this case.

Quote

Plug in does not detect Tegra device Jetson Nano · Issue #377 · NVIDIA/k8s-device-plugin (github.com) (https://github.com/NVIDIA/k8s-device-plugin/issues/377)

Note that looking at the initial logs that you provided you may have been using **`v1.7.0`** of the NVIDIA Container Toolkit. This is quite an old version and we greatly improved our support for Tegra-based systems with the **`v1.10.0`** release. It should also be noted that in order to use the GPU Device Plugin on Tegra-based systems (specifically targetting the integrated GPUs) at least **`v1.11.0`** of the NVIDIA Container Toolkit is required.

> There are no Tegra-specific changes in the `v1.12.0` release, so using the `v1.11.0` release should be sufficient in this case.

那么应该需要升级**NVIDIA Container Toolkit**

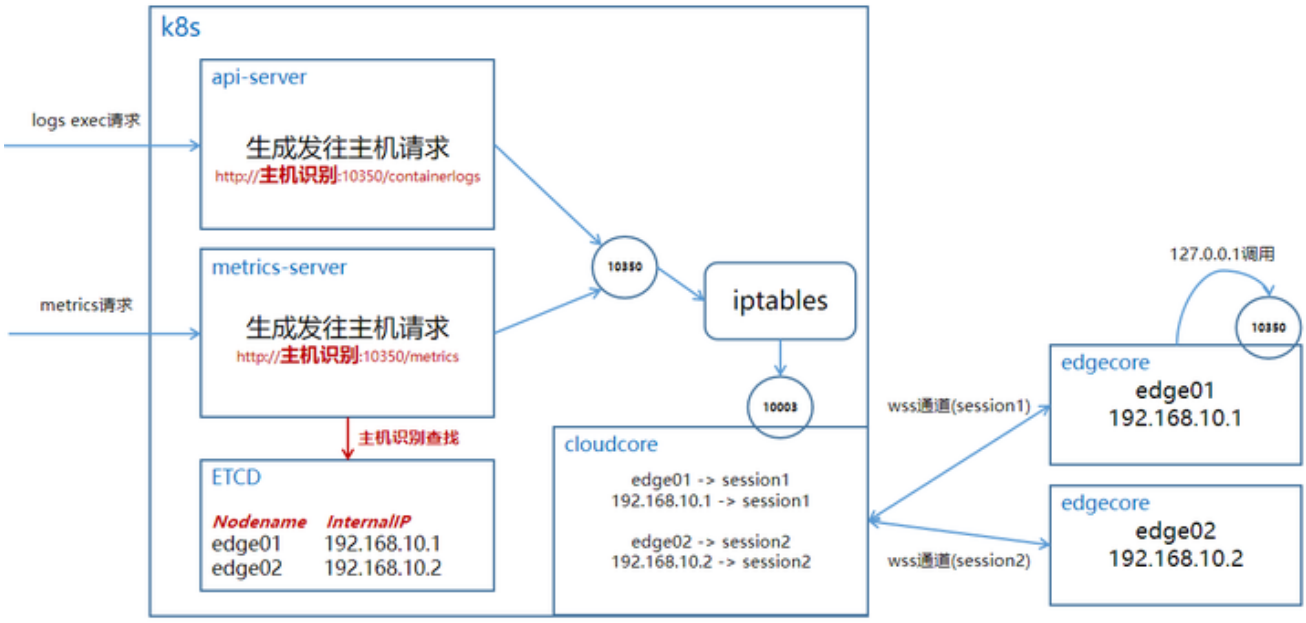
```shell
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
  && curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-container-toolkit.list | \
    sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' | \
    sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list

sudo apt-get update

sudo apt-get install -y nvidia-container-toolkit
```

可能出现的问题

`kubectl logs <pod-name>` 超时

```
root@cloud:/home/guest/yby/kubeedge/demo# kubectl logs joint-inference-example-edge-pnprm
Error from server: Get "https://192.168.1.4:10350/containerLogs/default/joint-inference-example-edge-pnprm/little-model": dial tcp 192.168.1.4:10350: i/o timeout
```

原因：借鉴 Kubernetes 边缘节点抓不到监控指标？试试这个方法！ - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/379962934)



可以发现报错访问的端口就是 10350，而在 kubeedge 中 10350 应该会进行转发，所以应该是 cloudcore 的设置问题。

解决：Enable Kubectl logs/exec to debug pods on the edge | KubeEdge (https://kubeedge.io/docs/advanced/debug/) 根据这个链接设置即可。

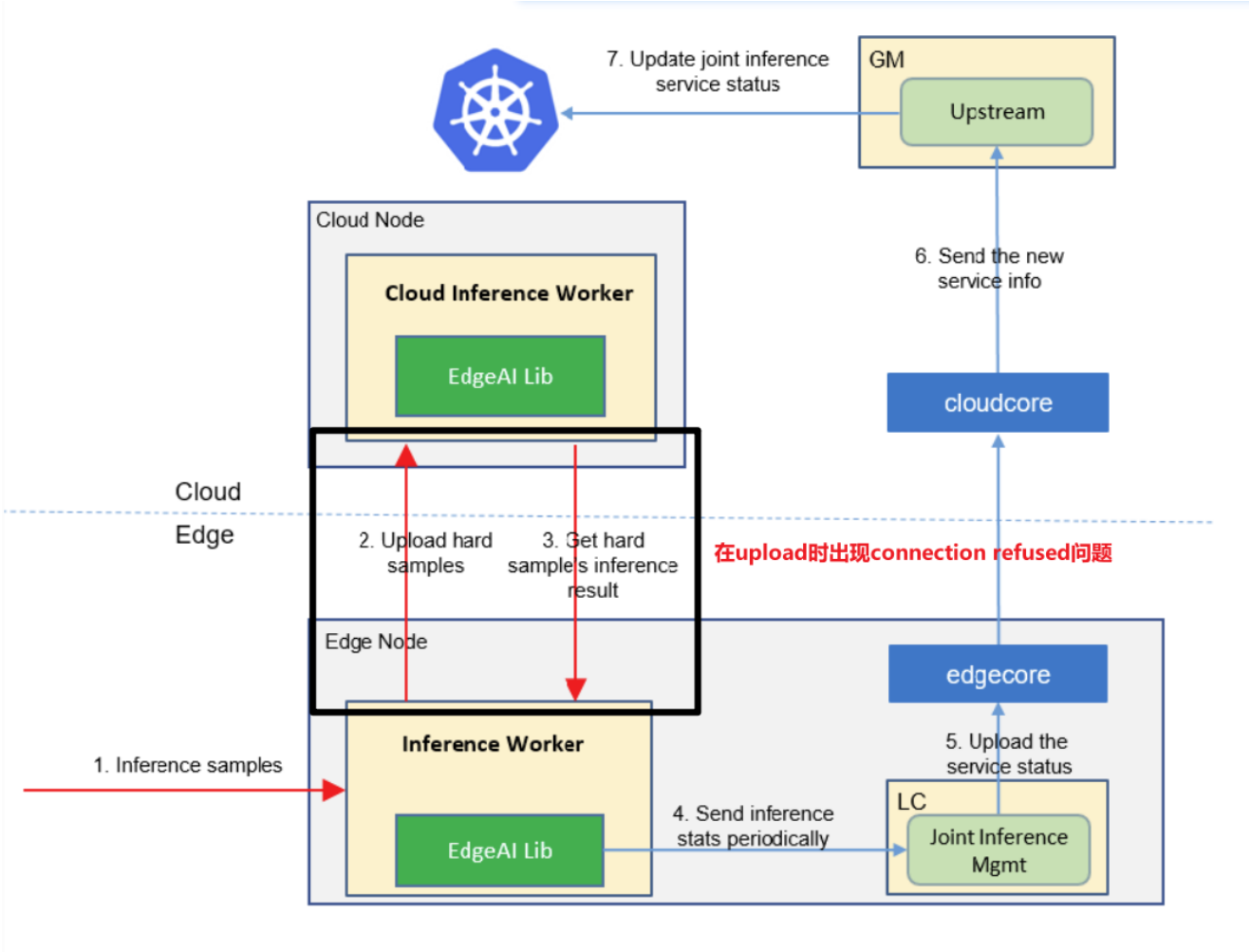
`kubectl logs <pod-name>` 卡住

可能的原因：之前 `kubectl logs` 时未结束就 ctrl+c 结束了导致后续卡住 解决：重启 edgecore `systemctl restart edgecore.service`

Sedna 云边通信不成功

```
root@cloud:/home/guest/yby/kubeedge/demo# kubectl logs joint-inference-example-edge-zg2hr | tail -n 10
[2023-11-14 02:46:57,776] client.py(62) [WARNING] - Connection refused in request http://joint-inference-example-cloud.default:5000/sedna/predict
[2023-11-14 02:47:02,131] client.py(62) [WARNING] - Connection refused in request http://joint-inference-example-cloud.default:5000/sedna/predict
[2023-11-14 02:47:06,467] client.py(62) [WARNING] - Connection refused in request http://joint-inference-example-cloud.default:5000/sedna/predict
[2023-11-14 02:47:10,823] client.py(62) [WARNING] - Connection refused in request http://joint-inference-example-cloud.default:5000/sedna/predict
[2023-11-14 02:47:15,187] client.py(62) [WARNING] - Connection refused in request http://joint-inference-example-cloud.default:5000/sedna/predict
[2023-11-14 02:47:15,188] joint_inference.py(262) [ERROR] - get cloud result error: RetryError[<Future at 0x7f56e11910 state=finished returned NoneType>]
[2023-11-14 02:47:24,487] client.py(62) [WARNING] - Connection refused in request http://joint-inference-example-cloud.default:5000/sedna/predict
[2023-11-14 02:47:28,952] client.py(62) [WARNING] - Connection refused in request http://joint-inference-example-cloud.default:5000/sedna/predict
[2023-11-14 02:47:33,312] client.py(62) [WARNING] - Connection refused in request http://joint-inference-example-cloud.default:5000/sedna/predict
[2023-11-14 02:47:37,824] client.py(62) [WARNING] - Connection refused in request http://joint-inference-example-cloud.default:5000/sedna/predict
```

`Connection refused`: 连接被拒绝。通常是连接还没建立，client 正在发 SYN 包请求建立连接，但到了 server 之后发现端口没监听，就返回 RST 包，然后应用层就报错连接被拒绝。



排查

根据报错，首先推测是 DNS 解析问题，利用 `kubectl exec` 进入后发现确实如此

```
nvidia@ubuntu:/joint_inference/output$ kubectl get svc
kNAME                          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
joint-inference-example-cloud   ClusterIP   10.97.193.34   <none>        5000/TCP   38h
kubernetes                     ClusterIP   10.96.0.1      <none>        443/TCP    25d
nvidia@ubuntu:/joint_inference/output$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
joint-inference-example-cloud-wcsls   1/1     Running   2          38h
joint-inference-example-edge-zg2hr    1/1     Running   0          38h
nvidia@ubuntu:/joint_inference/output$ kubectl exec -it joint-inference-example-edge-zg2hr sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
# wget -o- joint-inference-example-cloud.default:5000
--2023-11-15 03:21:07--  http://joint-inference-example-cloud.default:5000/
Resolving joint-inference-example-cloud.default (joint-inference-example-cloud.default)... failed: Name or service not known.
wget: unable to resolve host address 'joint-inference-example-cloud.default'
#
#
#
# wget -o- 10.97.193.34:5000
--2023-11-15 03:21:24--  http://10.97.193.34:5000/
Connecting to 10.97.193.34:5000... connected.
HTTP request sent, awaiting response...
```

查看 kube-proxy 和 coredns kube-proxy

```
root@cloud:~# kubectl get pods -n kube-system
NAME                                   READY   STATUS    RESTARTS   AGE
coredns-fcfffbc44-cvwsk                1/1     Running   0
2d17h
coredns-fcfffbc44-pm7jf                1/1     Running   0
2d17h
etcd-cloud.kubeedge                    1/1     Running   1          27d
kube-apiserver-cloud.kubeedge          1/1     Running   1          27d
kube-controller-manager-cloud.kubeedge 1/1     Running   4          27d
kube-proxy-jmh5f                       1/1     Running   0
2d17h
kube-scheduler-cloud.kubeedge          1/1     Running   3          27d
root@cloud:~# kubectl logs kube-proxy-jmh5f -n kube-system
I1114 07:42:21.104417       1 node.go:172] Successfully retrieved node IP:
114.212.81.11
I1114 07:42:21.104521       1 server_others.go:142] kube-proxy node IP is
an IPv4 address (114.212.81.11), assume IPv4 operation
I1114 07:42:21.137933       1 server_others.go:258] Using ipvs Proxier.
W1114 07:42:21.138501       1 proxier.go:445] IPVS scheduler not
specified, use rr by default
I1114 07:42:21.138933       1 server.go:650] Version: v1.20.0
I1114 07:42:21.139515       1 conntrack.go:52] Setting nf_conntrack_max to
1572864
I1114 07:42:21.139872       1 config.go:224] Starting endpoint slice
config controller
I1114 07:42:21.139895       1 shared_informer.go:240] Waiting for caches
to sync for endpoint slice config
I1114 07:42:21.139899       1 config.go:315] Starting service config
controller
I1114 07:42:21.139906       1 shared_informer.go:240] Waiting for caches
to sync for service config
I1114 07:42:21.240038       1 shared_informer.go:247] Caches are synced
for service config
I1114 07:42:21.240048       1 shared_informer.go:247] Caches are synced
for endpoint slice config
E1115 02:39:31.707943       1 proxier.go:1613] Failed to execute iptables-
restore: exit status 1 (iptables-restore: line 18 failed
)
Rules:
*nat
:KUBE-SERVICES - [0:0]
:KUBE-POSTROUTING - [0:0]
:KUBE-FIREWALL - [0:0]
:KUBE-NODE-PORT - [0:0]
:KUBE-LOAD-BALANCER - [0:0]
```

```
:KUBE-MARK-MASQ - [0:0]
-A KUBE-POSTROUTING -m comment --comment "Kubernetes endpoints dst
ip:port, source ip for solving hairpin purpose" -m set --match-set KUBE-
LOOP-BACK dst,dst,src -j MASQUERADE
-A KUBE-SERVICES -m comment --comment "Kubernetes service cluster ip +
port for masquerade purpose" -m set --match-set KUBE-CLUSTER-IP dst,dst !
-s 10.244.0.0/16 -j KUBE-MARK-MASQ
-A KUBE-SERVICES -m addrtype --dst-type LOCAL -j KUBE-NODE-PORT
-A KUBE-LOAD-BALANCER -j KUBE-MARK-MASQ
-A KUBE-FIREWALL -j KUBE-MARK-DROP
-A KUBE-SERVICES -m set --match-set KUBE-CLUSTER-IP dst,dst -j ACCEPT
-A KUBE-POSTROUTING -m mark ! --mark 0x00004000/0x00004000 -j RETURN
-A KUBE-POSTROUTING -j MARK --xor-mark 0x00004000
-A KUBE-POSTROUTING -m comment --comment "kubernetes service traffic
requiring SNAT" -j MASQUERADE --random-fully
-A KUBE-MARK-MASQ -j MARK --or-mark 0x00004000
COMMIT
*filter
:KUBE-FORWARD - [0:0]
-A KUBE-FORWARD -m comment --comment "kubernetes forwarding rules" -m mark
--mark 0x00004000/0x00004000 -j ACCEPT
-A KUBE-FORWARD -m comment --comment "kubernetes forwarding conntrack pod
source rule" -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A KUBE-FORWARD -m comment --comment "kubernetes forwarding conntrack pod
destination rule" -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
COMMIT
W1115 15:11:08.869075       1 reflector.go:436] k8s.io/client-
go/informers/factory.go:134: watch of *v1.Service ended with: an error on
the server ("unable to decode an event from the watch stream: http2:
client connection lost") has prevented the request from succeeding
W1115 15:11:08.869093       1 reflector.go:436] k8s.io/client-
go/informers/factory.go:134: watch of *v1beta1.EndpointSlice ended with:
an error on the server ("unable to decode an event from the watch stream:
http2: client connection lost") has prevented the request from succeeding
W1115 15:12:54.705854       1 reflector.go:436] k8s.io/client-
go/informers/factory.go:134: watch of *v1.Service ended with: an error on
the server ("unable to decode an event from the watch stream: http2:
client connection lost") has prevented the request from succeeding
W1115 15:12:54.718106       1 reflector.go:436] k8s.io/client-
go/informers/factory.go:134: watch of *v1beta1.EndpointSlice ended with:
an error on the server ("unable to decode an event from the watch stream:
http2: client connection lost") has prevented the request from succeeding
```

```
kube-scheduler-cloud.kubeedge            1/1     Running   3          27d
root@cloud:~# kubectl logs kube-proxy-jmh5f -n kube-system
I1114 07:42:21.104417       1 node.go:172] Successfully retrieved node IP: 114.212.81.11
I1114 07:42:21.104521       1 server_others.go:142] kube-proxy node IP is an IPv4 address (114.212.81.11), assume IPv4 operation
I1114 07:42:21.137933       1 server_others.go:258] Using ipvs Proxier.
W1114 07:42:21.138501       1 proxier.go:445] IPVS scheduler not specified, use rr by default
I1114 07:42:21.138933       1 server.go:650] Version: v1.20.0
I1114 07:42:21.139515       1 conntrack.go:52] Setting nf_conntrack_max to 1572864
I1114 07:42:21.139872       1 config.go:224] Starting endpoint slice config controller
I1114 07:42:21.139895       1 shared_informer.go:240] Waiting for caches to sync for endpoint slice config
I1114 07:42:21.139899       1 config.go:315] Starting service config controller
I1114 07:42:21.139906       1 shared_informer.go:240] Waiting for caches to sync for service config
I1114 07:42:21.240038       1 shared_informer.go:247] Caches are synced for service config
I1114 07:42:21.240048       1 shared_informer.go:247] Caches are synced for endpoint slice config
E1115 02:39:31.707943       1 proxier.go:1613] Failed to execute iptables-restore: exit status 1 (iptables-restore: line 18 failed
)
Rules:
*nat
:KUBE-SERVICES - [0:0]
:KUBE-POSTROUTING - [0:0]
:KUBE-FIREWALL - [0:0]
:KUBE-NODE-PORT - [0:0]
:KUBE-LOAD-BALANCER - [0:0]
:KUBE-MARK-MASQ - [0:0]
-A KUBE-POSTROUTING -m comment --comment "Kubernetes endpoints dst ip:port, source ip for solving hairpin purpose" -m set --match-set KUBE-LOOP-BACK dst,dst,src -j MASQUERADE
-A KUBE-SERVICES -m comment --comment "Kubernetes service cluster ip + port for masquerade purpose" -m set --match-set KUBE-CLUSTER-IP dst,dst ! -s 10.244.0.0/16 -j KUBE-MARK-MASQ
-A KUBE-SERVICES -m addrtype --dst-type LOCAL -j KUBE-NODE-PORT
-A KUBE-LOAD-BALANCER -j KUBE-MARK-MASQ
-A KUBE-FIREWALL -j KUBE-MARK-DROP
-A KUBE-SERVICES -m set --match-set KUBE-CLUSTER-IP dst,dst -j ACCEPT
-A KUBE-POSTROUTING -m mark ! --mark 0x00004000/0x00004000 -j RETURN
-A KUBE-POSTROUTING -j MARK --xor-mark 0x00004000
-A KUBE-POSTROUTING -m comment --comment "kubernetes service traffic requiring SNAT" -j MASQUERADE --random-fully
-A KUBE-MARK-MASQ -j MARK --or-mark 0x00004000
COMMIT
*filter
:KUBE-FORWARD - [0:0]
-A KUBE-FORWARD -m comment --comment "kubernetes forwarding rules" -m mark --mark 0x00004000/0x00004000 -j ACCEPT
-A KUBE-FORWARD -m comment --comment "kubernetes forwarding conntrack pod source rule" -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A KUBE-FORWARD -m comment --comment "kubernetes forwarding conntrack pod destination rule" -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
COMMIT
W1115 15:11:08.869075       1 reflector.go:436] k8s.io/client-go/informers/factory.go:134: watch of *v1.Service ended with: an error on the server ("unable to decode an event from the w
atch stream: http2: client connection lost") has prevented the request from succeeding
W1115 15:11:08.869093       1 reflector.go:436] k8s.io/client-go/informers/factory.go:134: watch of *v1beta1.EndpointSlice ended with: an error on the server ("unable to decode an event
 from the watch stream: http2: client connection lost") has prevented the request from succeeding
W1115 15:12:54.705854       1 reflector.go:436] k8s.io/client-go/informers/factory.go:134: watch of *v1.Service ended with: an error on the server ("unable to decode an event from the w
atch stream: http2: client connection lost") has prevented the request from succeeding
W1115 15:12:54.718106       1 reflector.go:436] k8s.io/client-go/informers/factory.go:134: watch of *v1beta1.EndpointSlice ended with: an error on the server ("unable to decode an event
 from the watch stream: http2: client connection lost") has prevented the request from succeeding
root@cloud:~#
```

coredns

SHELL

```
[INFO] plugin/reload: Running configuration MD5 =
db32ca3650231d74073ff4cf814959a7
CoreDNS-1.7.0
linux/amd64, go1.14.4, f59c03d
E1114 14:06:39.779265       1 reflector.go:382] pkg/mod/k8s.io/client-
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Namespace:
Get "https://10.96.0.1:443/api/v1/namespaces?
allowWatchBookmarks=true&resourceVersion=2829994&timeout=6m36s&timeoutSeco
nds=396&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:39.779268       1 reflector.go:382] pkg/mod/k8s.io/client-
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Service: Get
"https://10.96.0.1:443/api/v1/services?
allowWatchBookmarks=true&resourceVersion=2829894&timeout=5m41s&timeoutSeco
nds=341&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:39.779268       1 reflector.go:382] pkg/mod/k8s.io/client-
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Endpoints:
Get "https://10.96.0.1:443/api/v1/endpoints?
allowWatchBookmarks=true&resourceVersion=2829895&timeout=7m41s&timeoutSeco
nds=461&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:40.800932       1 reflector.go:382] pkg/mod/k8s.io/client-
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Namespace:
Get "https://10.96.0.1:443/api/v1/namespaces?
allowWatchBookmarks=true&resourceVersion=2829994&timeout=7m51s&timeoutSeco
nds=471&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:40.801741       1 reflector.go:382] pkg/mod/k8s.io/client-
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Service: Get
"https://10.96.0.1:443/api/v1/services?
allowWatchBookmarks=true&resourceVersion=2829894&timeout=7m33s&timeoutSeco
nds=453&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:40.802698       1 reflector.go:382] pkg/mod/k8s.io/client-
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Endpoints:
Get "https://10.96.0.1:443/api/v1/endpoints?
allowWatchBookmarks=true&resourceVersion=2829895&timeout=8m25s&timeoutSeco
nds=505&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:41.801663       1 reflector.go:382] pkg/mod/k8s.io/client-
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Namespace:
Get "https://10.96.0.1:443/api/v1/namespaces?
allowWatchBookmarks=true&resourceVersion=2829994&timeout=8m15s&timeoutSeco
nds=495&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:41.802358       1 reflector.go:382] pkg/mod/k8s.io/client-
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Service: Get
"https://10.96.0.1:443/api/v1/services?
allowWatchBookmarks=true&resourceVersion=2829894&timeout=7m37s&timeoutSeco
nds=457&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:41.803410       1 reflector.go:382] pkg/mod/k8s.io/client-
```

```
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Endpoints:
Get "https://10.96.0.1:443/api/v1/endpoints?
allowWatchBookmarks=true&resourceVersion=2829895&timeout=8m16s&timeoutSeco
nds=496&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:42.802131       1 reflector.go:382] pkg/mod/k8s.io/client-
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Namespace:
Get "https://10.96.0.1:443/api/v1/namespaces?
allowWatchBookmarks=true&resourceVersion=2829994&timeout=8m34s&timeoutSeco
nds=514&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:42.803073       1 reflector.go:382] pkg/mod/k8s.io/client-
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Service: Get
"https://10.96.0.1:443/api/v1/services?
allowWatchBookmarks=true&resourceVersion=2829894&timeout=8m10s&timeoutSeco
nds=490&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:42.804206       1 reflector.go:382] pkg/mod/k8s.io/client-
go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Endpoints:
Get "https://10.96.0.1:443/api/v1/endpoints?
allowWatchBookmarks=true&resourceVersion=2829895&timeout=5m3s&timeoutSecon
ds=303&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
```

```
root@cloud:~# kubectl logs coredns-fcfffbc44-cvwsk -n kube-system
.:53
[INFO] plugin/reload: Running configuration MD5 = db32ca3650231d74073ff4cf814959a7
CoreDNS-1.7.0
linux/amd64, go1.14.4, f59c03d
E1114 14:06:39.779265       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Namespace: Get "https://10.96.0.1:443/api/v1/namespaces?allowWatchBookmarks=true&resourceVersion=2829994&timeout=6m36s&timeoutSeconds=396&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:39.779268       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Service: Get "https://10.96.0.1:443/api/v1/services?allowWatchBookmarks=true&resourceVersion=2829894&timeout=5m41s&timeoutSeconds=341&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:39.779268       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Endpoints: Get "https://10.96.0.1:443/api/v1/endpoints?allowWatchBookmarks=true&resourceVersion=2829895&timeout=7m41s&timeoutSeconds=461&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:40.800932       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Namespace: Get "https://10.96.0.1:443/api/v1/namespaces?allowWatchBookmarks=true&resourceVersion=2829994&timeout=7m51s&timeoutSeconds=471&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:40.801741       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Service: Get "https://10.96.0.1:443/api/v1/services?allowWatchBookmarks=true&resourceVersion=2829894&timeout=7m33s&timeoutSeconds=453&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:40.802698       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Endpoints: Get "https://10.96.0.1:443/api/v1/endpoints?allowWatchBookmarks=true&resourceVersion=2829895&timeout=8m25s&timeoutSeconds=505&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:41.801663       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Namespace: Get "https://10.96.0.1:443/api/v1/namespaces?allowWatchBookmarks=true&resourceVersion=2829994&timeout=8m15s&timeoutSeconds=495&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:41.802358       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Service: Get "https://10.96.0.1:443/api/v1/services?allowWatchBookmarks=true&resourceVersion=2829894&timeout=7m37s&timeoutSeconds=457&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:41.803410       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Endpoints: Get "https://10.96.0.1:443/api/v1/endpoints?allowWatchBookmarks=true&resourceVersion=2829895&timeout=8m16s&timeoutSeconds=496&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:42.802131       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Namespace: Get "https://10.96.0.1:443/api/v1/namespaces?allowWatchBookmarks=true&resourceVersion=2829994&timeout=8m34s&timeoutSeconds=514&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:42.803073       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Service: Get "https://10.96.0.1:443/api/v1/services?allowWatchBookmarks=true&resourceVersion=2829894&timeout=8m10s&timeoutSeconds=490&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
E1114 14:06:42.804206       1 reflector.go:382] pkg/mod/k8s.io/client-go@v0.18.3/tools/cache/reflector.go:125: Failed to watch *v1.Endpoints: Get "https://10.96.0.1:443/api/v1/endpoints?allowWatchBookmarks=true&resourceVersion=2829895&timeout=5m3s&timeoutSeconds=303&watch=true": dial tcp 10.96.0.1:443: connect: connection refused
root@cloud:~#
```

（懒得一个个找原因，并且理论上不该由 kube-proxy 来处理这个转发，直接选择 reset）（后知后觉，好像把 iptables 清了重设就好了）

Edgemesh log 报错

问题描述

1. **节点情况**:

```
(base) hx@cloud:/opt/sedna/build/lc$ kubectl get nodes -owide
NAME             STATUS   ROLES                  AGE   VERSION                INTERNAL-IP     EXTERNAL-IP   OS-IMAGE
KERNEL-VERSION      CONTAINER-RUNTIME
cloud.kubeedge   Ready    control-plane,master   28h   v1.20.0                114.212.81.11   <none>        Ubuntu 20.04.6 LTS
5.15.0-52-generic   docker://19.3.12
edge1.kubeedge   Ready    agent,edge             15h   v1.21.4-kubeedge-v1.9.2   192.168.1.2     <none>        Ubuntu 18.04.6 LTS
4.9.337-tegra       docker://24.0.2
edge2            Ready    agent,edge             26h   v1.21.4-kubeedge-v1.9.2   192.168.1.4     <none>        Ubuntu 18.04.6 LTS
4.9.337-tegra       docker://24.0.2
(base) hx@cloud:/opt/sedna/build/lc$
```

2. **EdgeMesh 日志**
   边缘节点:

```
I1118 13:34:27.258080       1 tunnel.go:205] Discovery service got a new stream from {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopW
bHxZDt: [/ip4/192.168.1.4/tcp/20006]}
I1118 13:34:27.258236       1 tunnel.go:234] [MDNS] Discovery from edge2 : {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/
ip4/192.168.1.4/tcp/20006]}
I1118 13:34:27.258263       1 tunnel.go:126] [MDNS] Discovery found peer: {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/i
p4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]}
I1118 13:34:27.258402       1 tunnel.go:156] [MDNS] New stream between peer {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [
/ip4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]} success
I1118 13:34:27.259605       1 tunnel.go:192] [MDNS] Discovery to edge2 : {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/ip
4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]}
I1118 14:28:27.258095       1 tunnel.go:126] [MDNS] Discovery found peer: {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/i
p4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]}
I1118 14:28:27.258366       1 tunnel.go:156] [MDNS] New stream between peer {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [
/ip4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]} success
I1118 14:28:27.258481       1 tunnel.go:205] Discovery service got a new stream from {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopW
bHxZDt: [/ip4/192.168.1.4/tcp/20006]}
I1118 14:28:27.258587       1 tunnel.go:234] [MDNS] Discovery from edge2 : {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/
ip4/192.168.1.4/tcp/20006]}
I1118 14:28:27.259842       1 tunnel.go:192] [MDNS] Discovery to edge2 : {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/ip
4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]}
I1118 15:22:27.284503       1 tunnel.go:126] [MDNS] Discovery found peer: {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/i
p4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]}
I1118 15:22:27.284681       1 tunnel.go:156] [MDNS] New stream between peer {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [
/ip4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]} success
I1118 15:22:27.358142       1 tunnel.go:205] Discovery service got a new stream from {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopW
bHxZDt: [/ip4/192.168.1.4/tcp/20006]}
I1118 15:22:27.358310       1 tunnel.go:234] [MDNS] Discovery from edge2 : {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/
ip4/192.168.1.4/tcp/20006]}
I1118 15:22:27.358499       1 tunnel.go:192] [MDNS] Discovery to edge2 : {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [/ip
4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]}
I1118 15:31:14.187225       1 tunnel.go:278] New stream between peer edge2: {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt: [
/ip4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]} success
I1118 15:31:14.189304       1 loadbalancer.go:732] Dial libp2p network between hostname-edge-84cb45ccf4-9ttnj - {tcp edge2 172.17.0.
2:9376}
guest@cloud:~/yby/test/edgemesh$
```

服务器：

```
E1118 15:19:08.064927       1 loadbalancer.go:686] "Dial failed" err="get proxy stream from edge1.kubeedge error: new stream between
 edge1.kubeedge: {12D3KooWFz1dKY8L3JC8wAY6sJ5MswvPEGKysPCfcaGxFmeH7wkz: []} err: failed to find any peer in table"
E1118 15:19:08.315583       1 loadbalancer.go:686] "Dial failed" err="get proxy stream from edge1.kubeedge error: new stream between
 edge1.kubeedge: {12D3KooWFz1dKY8L3JC8wAY6sJ5MswvPEGKysPCfcaGxFmeH7wkz: []} err: failed to find any peer in table"
E1118 15:19:08.816454       1 loadbalancer.go:686] "Dial failed" err="get proxy stream from edge1.kubeedge error: new stream between
 edge1.kubeedge: {12D3KooWFz1dKY8L3JC8wAY6sJ5MswvPEGKysPCfcaGxFmeH7wkz: []} err: failed to find any peer in table"
E1118 15:19:09.817629       1 loadbalancer.go:686] "Dial failed" err="get proxy stream from edge1.kubeedge error: new stream between
 edge1.kubeedge: {12D3KooWFz1dKY8L3JC8wAY6sJ5MswvPEGKysPCfcaGxFmeH7wkz: []} err: failed to find any peer in table"
E1118 15:19:11.818403       1 proxysocket.go:98] "Failed to connect to balancer" err="failed to connect to an endpoint"
guest@cloud:~/yby/test/edgemesh$
```

EdgeMesh 监听的端口是 20006，**发现服务器并没有监听 20006 端口**：

```
(base) hx@cloud:/opt/sedna/build/lc$ lsof -i:20006
(base) hx@cloud:/opt/sedna/build/lc$
```

边端正常运行情况：

```
root@edge2:~# lsof -i:20006
COMMAND     PID USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
edgemesh- 27579 root   15u  IPv4 24870497      0t0  TCP localhost:20006 (LISTEN)
^C
```

排查

先复习一下**定位模型**，确定**被访问节点**上的 edgemesh-agent(右)容器是否存在、是否处于正常运行中。

这个情况是非常经常出现的，因为master节点一般都有污点，会驱逐其他pod，进而导致edgemesh-agent部署不上去。这种情况可以通过去除节点污点，使edgemesh-agent部署上去解决。

如果访问节点和被访问节点的edgemesh-agent都正常启动了，但是还报这个错误，可能是因为访问节点和被访问节点没有互相发现导致，请这样排查：

1. 首先每个节点上的edgemesh-agent都具有peer ID，比如

```shell
edge2:
I'm {12D3KooWPpY4GqqNF3sLC397fMz5ZZfxmtMTNa1gLYFopWbHxZDt:
[/ip4/127.0.0.1/tcp/20006 /ip4/192.168.1.4/tcp/20006]}

edge1.kubeedge:
I'm {12D3KooWFz1dKY8L3JC8wAY6sJ5MswvPEGKysPCfcaGxFmeH7wkz:
[/ip4/127.0.0.1/tcp/20006 /ip4/192.168.1.2/tcp/20006]}

注意:
a. peer ID是根据节点名称哈希出来的，相同的节点名称会哈希出相同的peer ID
b. 另外，节点名称不是服务器名称，是k8s node name，请用kubectl get nodes查看
```

2.如果访问节点和被访问节点处于同一个局域网内（**所有节点应该具备内网IP（10.0.0.0/8、172.16.0.0/12、192.168.0.0/16**），请看全网最全EdgeMesh Q&A手册 - 知乎 (zhihu.com) (https://zhuanlan.zhihu.com/p/585749690) **问题十二**同一个局域网内edgemesh-agent互相发现对方时的日志是 `[MDNS] Discovery found peer: <被访问端peer ID: [被访问端IP列表(可能会包含中继节点IP)]>`

3.如果访问节点和被访问节点跨子网，这时候应该看看 relayNodes 设置的正不正确，为什么中继节点没办法协助两个节点交换 peer 信息。详细材料请阅读：KubeEdge EdgeMesh 高可用架构详解 (https://link.zhihu.com/?target=https%3A//mp.weixin.qq.com/s/4whnkMM9oOaWRsI1ICsvSA) 。跨子网的 edgemesh-agent 互相发现对方时的日志是 `[DHT] Discovery found peer: <被访问端peer ID: [被访问端IP列表(可能会包含中继节点IP)]>`（适用于我的情况）

解决

在部署 edgemesh 进行 `kubectl apply -f build/agent/resources/` 操作时，修改 04-configmap，添加 relayNode（根本原因在于，不符合 ^d3939d，所以需要添加 relayNode）

> ⚠ **Warning**
>
> 请根据你的 K8s 集群设置 build/agent/resources/04-configmap.yaml 的 relayNodes，并重新生成 PSK 密码。

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: edgemesh-agent-cfg
  namespace: kubeedge
  labels:
    k8s-app: kubeedge
    kubeedge: edgemesh-agent
data:
  edgemesh-agent.yaml: |
    # For more detailed configuration, please refer to: http
    modules:
      edgeProxy:
        enable: true
      edgeTunnel:
        enable: true
        relayNodes:
        - nodeName: cloud.kubeedge
          advertiseAddress:
          - 114.212.81.11
        #- nodeName: <your relay node name2>
        #  advertiseAddress:
        #  - 2.2.2.2
        #  - 3.3.3.3
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: edgemesh-agent-psk
  namespace: kubeedge
  labels:
    k8s-app: kubeedge
    kubeedge: edgemesh-agent
data:
  # Generated by `openssl rand -base64 32`
  # NOTE: Don't use this psk, please regenerate it!!! Please
```

Sedna lc 报错

`lc` 报错内容

```
E1118 09:56:22.935783       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: dial tc
p: lookup gm.sedna on 127.0.0.53:53: no such host
E1118 09:56:27.939866       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: dial tc
p: lookup gm.sedna on 127.0.0.53:53: no such host
E1118 09:56:32.940078       1 websocket.go:200] max retry count reached when connecting global manager(address: gm.sedna:9000)
I1118 09:56:32.940164       1 websocket.go:176] client starts to connect global manager(address: gm.sedna:9000)
E1118 09:56:32.943992       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: dial tc
p: lookup gm.sedna on 127.0.0.53:53: no such host
```

原因：域名解析问题。

解决

临时解决方案：(理论上它会提示不要手动修改...)

在宿主机上修改 `/etc/resolv.conf`

```shell
nameserver 169.254.96.16
```

```
nvidia@edge2:~/yby/demo/video$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
# 127.0.0.53 is the systemd-resolved stub resolver.
# run "systemd-resolve --status" to see details about the actual nameservers.

nameserver 169.254.96.16
nameserver 127.0.0.53
search default.svc.cluster.local svc.cluster.local cluster.local nju.edu.cn
```

仍存在的问题

```
kube-shell> kubectl logs lc-qxbq2 -n sedna
I1206 11:02:57.244365       1 server.go:123] manager dataset is started
I1206 11:02:57.244458       1 server.go:123] manager model is started
I1206 11:02:57.244495       1 websocket.go:176] client starts to connect global manager(address: gm.sedna:9000)
I1206 11:02:57.244523       1 server.go:123] manager jointinferenceservice is started
I1206 11:02:57.244552       1 server.go:123] manager jointmultiedgeservice is started
I1206 11:02:57.244682       1 server.go:123] manager federatedlearningjob is started
I1206 11:02:57.244734       1 server.go:123] manager incrementallearningjob is started
I1206 11:02:57.244754       1 server.go:123] manager lifelonglearningjob is started
I1206 11:02:57.244907       1 server.go:140] server binds port 9100 successfully
E1206 11:03:01.180747       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53786->10.99.234.104:9000: read: connection reset by peer
E1206 11:03:10.114467       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53798->10.99.234.104:9000: read: connection reset by peer
E1206 11:03:18.982035       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53812->10.99.234.104:9000: read: connection reset by peer
E1206 11:03:28.228995       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53822->10.99.234.104:9000: read: connection reset by peer
E1206 11:03:37.082521       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53832->10.99.234.104:9000: read: connection reset by peer
E1206 11:03:42.083081       1 websocket.go:200] max retry count reached when connecting global manager(address: gm.sedna:9000)
I1206 11:03:42.083170       1 websocket.go:176] client starts to connect global manager(address: gm.sedna:9000)
E1206 11:03:45.908156       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53840->10.99.234.104:9000: read: connection reset by peer
E1206 11:03:54.688428       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53848->10.99.234.104:9000: read: connection reset by peer
E1206 11:04:03.507325       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53850->10.99.234.104:9000: read: connection reset by peer
```

在 $edge2$ 上连接到 gm 存在偶然性，最终虽然一定能连上，但是会尝试很多次。

```
cp 192.168.1.4:53910->10.99.234.104:9000: read: connection reset by peer
E1206 11:05:22.913920       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53924->10.99.234.104:9000: read: connection reset by peer
E1206 11:05:31.908078       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53930->10.99.234.104:9000: read: connection reset by peer
E1206 11:05:40.782733       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53932->10.99.234.104:9000: read: connection reset by peer
E1206 11:05:49.583078       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53934->10.99.234.104:9000: read: connection reset by peer
E1206 11:05:54.583280       1 websocket.go:200] max retry count reached when connecting global manager(address: gm.sedna:9000)
I1206 11:05:54.583338       1 websocket.go:176] client starts to connect global manager(address: gm.sedna:9000)
E1206 11:05:58.362803       1 websocket.go:192] client tries to connect global manager(address: gm.sedna:9000) failed, error: read t
cp 192.168.1.4:53938->10.99.234.104:9000: read: connection reset by peer
I1206 11:06:03.371120       1 websocket.go:187] websocket connects global manager(address: gm.sedna:9000) successful
```

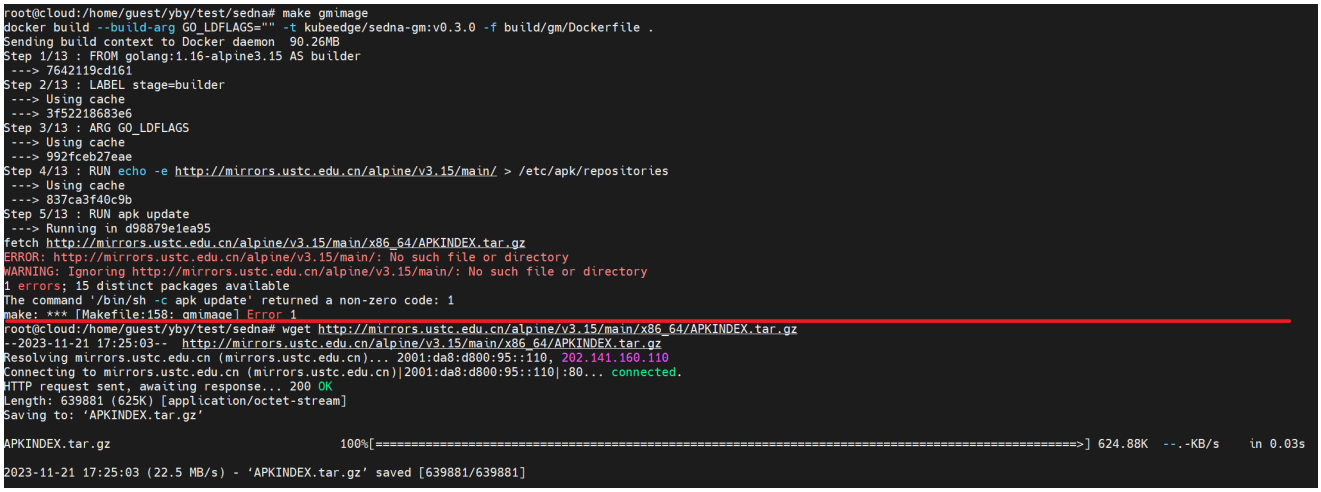
在 kube-shell 下修改了命名空间

命令行下修改命名空间：

```shell
kubectl config set-context $(kubectl config current-context) --
namespace=default
```

build gm 报错

问题描述




```shell
fetch http://mirrors.ustc.edu.cn/alpine/v3.15/main/x86_64/APKINDEX.tar.gz
ERROR: http://mirrors.ustc.edu.cn/alpine/v3.15/main/: No such file or
directory
WARNING: Ignoring http://mirrors.ustc.edu.cn/alpine/v3.15/main/: No such
file or directory
1 errors; 15 distinct packages available
The command '/bin/sh -c apk update' returned a non-zero code: 1
make: *** [Makefile:158: gmimage] Error 1
```

解决

由于在宿主机上直接 wget 链接是可以下载的，推测是 docker 内网络问题，于是指定网络。`make gmimage` 实际对应的命令是 `docker build --build-arg GO_LDFLAGS="" --network=host -t kubeedge/sedna-gm:v0.3.0 -f build/gm/Dockerfile .` 最后再添加上 `--network=host`

即最终的命令为：

```shell
docker build --build-arg GO_LDFLAGS="" --network=host -t kubeedge/sedna-
gm:v0.3.0 -f build/gm/Dockerfile .
```

官方提供的镜像只适合于 x86 架构

解决

使用**交叉编译**：

安装 buildx

`buildx` 是 Docker 官方提供的一个构建工具，它可以帮助用户快速、高效地构建 Docker 镜像，并支持多种平台的构建。使用 `buildx`，用户可以在单个命令中构建多种架构的镜像，例如 x86 和 ARM 架构，而无需手动操作多个构建命令。此外，`buildx` 还支持 Dockerfile 的多阶段构建和缓存，这可以大大提高镜像构建的效率和速度。

如果需要手动安装，可以从 GitHub 发布页面下载 (https://link.zhihu.com/?target=https%3A//github.com/docker/buildx/releases/tag/v0.10.4) 对应平台的最新二进制文件，重命名为 `docker-buildx`，然后将其放到 Docker 插件目录下（Linux/Mac 系统为 `$HOME/.docker/cli-plugins`，Windows 系统为 `%USERPROFILE%\.docker\cli-plugins`）。 给插件增加可执行权限 `chmod +x ~/.docker/cli-plugins/docker-buildx`

```shell
# 我在虚拟机上进行的交叉编译，是x86架构
wget https://github.com/docker/buildx/releases/download/v0.10.4/buildx-v0.10.4.darwin-amd64

# 重命名
mv buildx-v0.10.4.darwin-amd64 docker-buildx

# 放到对应目录
mv docker-buildx $HOME/.docker/cli-plugins

# 添加权限
chmod +x ~/.docker/cli-plugins/docker-buildx

# 验证可用
docker buildx version
```

创建 mybuilder

要使用 `buildx` 构建跨平台镜像，我们需要先创建一个 `builder`，可以翻译为「构建器」。

```shell
$ docker buildx create --name mybuilder --use
mybuilder

$ docker buildx ls
NAME/NODE    DRIVER/ENDPOINT             STATUS  BUILDKIT
PLATFORMS
mybuilder *  docker-container
  mybuilder0 unix:///var/run/docker.sock running v0.12.3
linux/amd64, linux/amd64/v2, linux/amd64/v3, linux/amd64/v4, linux/386
default      docker
  default    default                     running v0.11.6+0a15675913b7
linux/amd64, linux/amd64/v2, linux/amd64/v3, linux/amd64/v4, linux/386
```

其中 * 所在的即为正在使用的

启动 mybuilder

如果 mybuilder 的状态为 `inactive` 或者 `stopped` 需要手动进行启动（上面的操作中--use 会自动启动）

```shell
$ docker buildx inspect --bootstrap mybuilder
[+] Building 16.8s (1/1) FINISHED
 => [internal] booting buildkit
16.8s
 => => pulling image moby/buildkit:buildx-stable-1
16.1s
 => => creating container buildx_buildkit_mybuilder0
0.7s
Name:   mybuilder
Driver: docker-container

Nodes:
Name:      mybuilder0
Endpoint:  unix:///var/run/docker.sock
Status:    running
Buildkit:  v0.12.3
Platforms: linux/amd64, linux/amd64/v2, linux/amd64/v3, linux/amd64/v4,
linux/386
```

构建镜像

以 sedna 的 lc 为例，lc 运行在云端和边端，其中云端是 x 86 架构，边端是 arm 架构

```shell
$ docker buildx build --platform linux/arm64,linux/amd64 --build-arg
GO_LDFLAGS="" -t adayoung/sedna-lc:v0.3.3 -f build/lc/Dockerfile . --push
```

大面积 Evicted（disk pressure）

原因

- node 上的 kubelet 负责采集资源占用数据，并和预先设置的 threshold 值进行比较，如果超过 threshold 值，kubelet 会杀掉一些 Pod 来回收相关资源，K8sg官网解读kubernetes配置资源不足处理 (https://links.jianshu.com/go?to=https%3A%2F%2Fkubernetes.io%2Fdocs%2Ftasks%2Fadminister-cluster%2Fout-of-resource%2F)
- 默认启动时，node 的可用空间低于15%的时候，该节点上讲会执行 eviction 操作，由于磁盘已经达到了85%,在怎么驱逐也无法正常启动就会一直重启，Pod 状态也是 pending 中

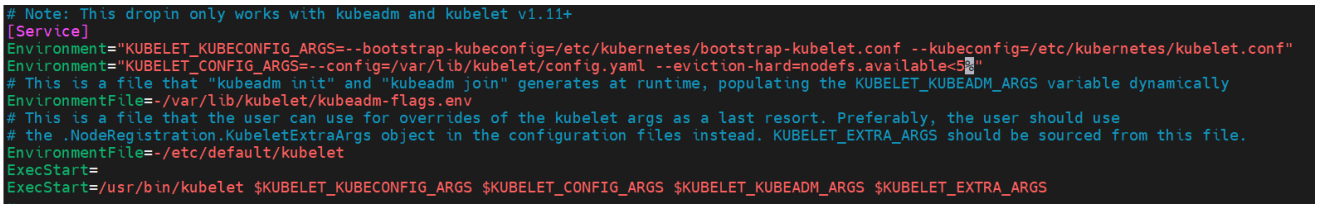
临时解决方法

- 修改配置文件增加传参数,添加此配置项`--eviction-hard=nodefs.available<5%`

```shell
root@cloud:/usr/lib/systemd/system# systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor
preset: enabled)
    Drop-In: /etc/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: active (running) since Fri 2023-12-15 09:17:50 CST; 5min ago
       Docs: https://kubernetes.io/docs/home/
   Main PID: 1070209 (kubelet)
      Tasks: 59 (limit: 309024)
     Memory: 67.2M
     CGroup: /system.slice/kubelet.service
```

可以看到配置文件目录所在位置是`/etc/systemd/system/kubelet.service.d`，配置文件是`10-kubeadm.conf`

```shell
vim /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-
kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --
kubeconfig=/etc/kubernetes/kubelet.conf"
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml --
eviction-hard=nodefs.available<5%"

最后添加上--eviction-hard=nodefs.available<5%
```

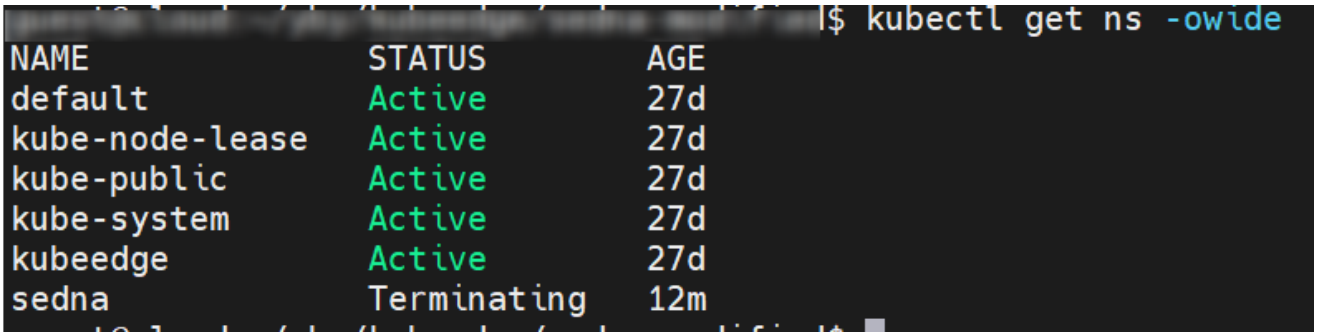


然后重启 kubelet

```shell
systemctl daemon-reload
systemctl  restart kubelet
```

会发现可以正常部署了(但是只是急救一下，磁盘空间感觉还是得清理下)

删除命名空间卡在 terminating



理论上一直等待应该是可以的(但是我等了半个钟也没成功啊!!) **方法一**，但是没啥用，依旧卡住

```shell
kubectl delete ns sedna --force --grace-period=0
```

方法二:

SHELL

开启一个代理终端
$ kubectl proxy
Starting to serve on 127.0.0.1:8001

再开启一个操作终端
将test namespace的配置文件输出保存
$ kubectl get ns sedna -o json > sedna.json
删除spec及status部分的内容还有metadata字段后的","号，切记!
剩下内容大致如下
guest@cloud:~/yby$ cat sedna.json

```
{
    "apiVersion": "v1",
    "kind": "Namespace",
    "metadata": {
        "creationTimestamp": "2023-12-14T09:12:13Z",
        "deletionTimestamp": "2023-12-14T09:15:25Z",
        "managedFields": [
            {
                "apiVersion": "v1",
                "fieldsType": "FieldsV1",
                "fieldsV1": {
                    "f:status": {
                        "f:phase": {}
                    }
                },
                "manager": "kubectl-create",
                "operation": "Update",
                "time": "2023-12-14T09:12:13Z"
            },
            {
                "apiVersion": "v1",
                "fieldsType": "FieldsV1",
                "fieldsV1": {
                    "f:status": {
                        "f:conditions": {
                            ".": {},
                            "k:{\"type\":\"NamespaceContentRemaining\"}":
{
                                ".": {},
                                "f:lastTransitionTime": {},
                                "f:message": {},
                                "f:reason": {},
                                "f:status": {},
                                "f:type": {}
                            },
```

```
                            "k:
{\"type\":\"NamespaceDeletionContentFailure\"}": {
                                ".": {},
                                "f:lastTransitionTime": {},
                                "f:message": {},
                                "f:reason": {},
                                "f:status": {},
                                "f:type": {}
                            },
                            "k:
{\"type\":\"NamespaceDeletionDiscoveryFailure\"}": {
                                ".": {},
                                "f:lastTransitionTime": {},
                                "f:message": {},
                                "f:reason": {},
                                "f:status": {},
                                "f:type": {}
                            },
                            "k:
{\"type\":\"NamespaceDeletionGroupVersionParsingFailure\"}": {
                                ".": {},
                                "f:lastTransitionTime": {},
                                "f:message": {},
                                "f:reason": {},
                                "f:status": {},
                                "f:type": {}
                            },
                            "k:
{\"type\":\"NamespaceFinalizersRemaining\"}": {
                                ".": {},
                                "f:lastTransitionTime": {},
                                "f:message": {},
                                "f:reason": {},
                                "f:status": {},
                                "f:type": {}
                            }
                        }
                    }
                },
                "manager": "kube-controller-manager",
                "operation": "Update",
                "time": "2023-12-14T09:15:30Z"
            }
        ],
        "name": "sedna",
        "resourceVersion": "3351515",
```

```
        "uid": "99cb8afb-a4c1-45e6-960d-ff1b4894773d"
    }
}
```

调接口删除

```
# curl -k -H "Content-Type: application/json" -X PUT --data-binary
@sedna.json http://127.0.0.1:8001/api/v1/namespaces/sedna/finalize
{
  "kind": "Namespace",
  "apiVersion": "v1",
  "metadata": {
    "name": "sedna",
    "uid": "99cb8afb-a4c1-45e6-960d-ff1b4894773d",
    "resourceVersion": "3351515",
    "creationTimestamp": "2023-12-14T09:12:13Z",
    "deletionTimestamp": "2023-12-14T09:15:25Z",
    "managedFields": [
      {
        "manager": "curl",
        "operation": "Update",
        "apiVersion": "v1",
        "time": "2023-12-14T09:42:38Z",
        "fieldsType": "FieldsV1",
        "fieldsV1": {"f:status":{"f:phase":{}}}
      }
    ]
  },
  "spec": {

  },
  "status": {
    "phase": "Terminating",
    "conditions": [
      {
        "type": "NamespaceDeletionDiscoveryFailure",
        "status": "True",
        "lastTransitionTime": "2023-12-14T09:15:30Z",
        "reason": "DiscoveryFailed",
        "message": "Discovery failed for some groups, 1 failing: unable to
retrieve the complete list of server APIs: metrics.k8s.io/v1beta1: the
server is currently unable to handle the request"
      },
      {
        "type": "NamespaceDeletionGroupVersionParsingFailure",
        "status": "False",
```

```
          "lastTransitionTime": "2023-12-14T09:15:30Z",
          "reason": "ParsedGroupVersions",
          "message": "All legacy kube types successfully parsed"
        },
        {
          "type": "NamespaceDeletionContentFailure",
          "status": "False",
          "lastTransitionTime": "2023-12-14T09:15:30Z",
          "reason": "ContentDeleted",
          "message": "All content successfully deleted, may be waiting on
  finalization"
        },
        {
          "type": "NamespaceContentRemaining",
          "status": "False",
          "lastTransitionTime": "2023-12-14T09:15:30Z",
          "reason": "ContentRemoved",
          "message": "All content successfully removed"
        },
        {
          "type": "NamespaceFinalizersRemaining",
          "status": "False",
          "lastTransitionTime": "2023-12-14T09:15:30Z",
          "reason": "ContentHasNoFinalizers",
          "message": "All content-preserving finalizers finished"
        }
      ]
    }
  }
```


强制删除 pod 之后部署不成功

问题描述

因为现在 edge 的 pod 是通过创建 deployment 由 deployment 进行创建，但是通过 `kubectl delete deploy <deploy-name>` 删除 deployment 后，pod 一直卡在了 terminating 状态，于是采用了 `kubectl delete pod edgeworker-deployment-7g5hs-58dffc5cd7-b77wz --force --grace-period=0` 命令进行了删除。 然后发现重新部署时候发现 assigned to edge 的 pod 都卡在 pending 状态。

解决

因为--force 是不会实际终止运行的，所以本身原来的 docker 可能还在运行，现在的做法是手动去对应的边缘节点上删除对应的容器（包括 pause，关于 pause 可以看这篇文章大白话 K8S（03）：从 Pause 容器理解 Pod 的本质 - 知乎 (zhihu.com)

(https://zhuanlan.zhihu.com/p/464712164) ），然后重启 edgecore: `systemctl restart edgecore.service`

```
507  docker ps
508  docker stop dd94221eea1f a802efa5f2e6
509  docker rm dd94221eea1f a802efa5f2e6
510  clear
511  docker ps
[redacted]
[redacted]
514  journalctl -u edgecore.service -f
515  systemctl restart edgecore
516  journalctl -u edgecore.service -f
```

应该就可以解决这个问题了。

HPA 相关

报错

我的 hpa 命令：

```shell
kubectl autoscale deployment hpa-demo --cpu-percent=10 --min=1 --max=10
```

发现 warning：（有这个 warning 就不能自动扩容了）

```shell
 HPA was unable to compute the replica count: failed to get cpu
utilization:
      missing request for cpu"}]'
```

```
guest@cloud:~/yby/kubeedge/demo/hpa$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
hpa-demo-7848d4b86f-8qmjc   1/1     Running   0          10m
guest@cloud:~/yby/kubeedge/demo/hpa$ kubectl get hpa
NAME       REFERENCE              TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
hpa-demo   Deployment/hpa-demo    <unknown>/10%   1         10        1          4m50s
guest@cloud:~/yby/kubeedge/demo/hpa$ kubectl describe hpa/hpa-demo
Name:                                                  hpa-demo
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Wed, 13 Dec 2023 15:54:37 +0800
Reference:                                             Deployment/hpa-demo
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  <unknown> / 10%
Min replicas:                                          1
Max replicas:                                          10
Deployment pods:                                       1 current / 0 desired
Conditions:
  Type           Status  Reason                   Message
  ----           ------  ------                   -------
  AbleToScale    True    SucceededGetScale        the HPA controller was able to get the target's current scale
  ScalingActive  False   FailedGetResourceMetric  the HPA was unable to compute the replica count: failed to get cpu utilization: missing request for cpu
Events:
  Type     Reason                        Age                  From                       Message
  ----     ------                        ----                 ----                       -------
  Warning  FailedComputeMetricsReplicas  113s (x12 over 4m43s)  horizontal-pod-autoscaler  invalid metrics (1 invalid out of 1), first error is: failed to get cpu utilizatio
n: missing request for cpu
  Warning  FailedGetResourceMetric       98s (x13 over 4m43s)   horizontal-pod-autoscaler  failed to get cpu utilization: missing request for cpu
```

回去看部署的 yaml 文件：

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hpa-demo
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

根据提示应该在部署的时候设置 cpu 的 request（因为 autoscale 的标准设置为 cpu ）

重新部署完 `hpa-demo.yaml`，删除原来的 hpa 后重新输入命令

```shell
kubectl autoscale deployment hpa-demo --cpu-percent=10 --min=1 --max=10

kubectl describe hpa hpa-demo
```

发现新的问题：

```shell
Events:
  Type     Reason                        Age   From
Message
  ----     ------                        ----  ----
-------
  Warning  FailedGetResourceMetric       8s    horizontal-pod-autoscaler
failed to get cpu utilization: unable to get metrics for resource cpu: no
metrics returned from resource metrics API
  Warning  FailedComputeMetricsReplicas  8s    horizontal-pod-autoscaler
invalid metrics (1 invalid out of 1), first error is: failed to get cpu
utilization: unable to get metrics for resource cpu: no metrics returned
from resource metrics API
```

```
Conditions:
  Type           Status  Reason                  Message
  ----           ------  ------                  -------
  AbleToScale    True    SucceededGetScale       the HPA controller was able to get the target's current scale
  ScalingActive  False   FailedGetResourceMetric  the HPA was unable to compute the replica count: failed to get cpu utilization: unable to get metrics for resource cpu: no
metrics returned from resource metrics API
Events:
  Type     Reason                        Age   From                       Message
  ----     ------                        ----  ----                       -------
  Warning  FailedGetResourceMetric       8s    horizontal-pod-autoscaler  failed to get cpu utilization: unable to get metrics for resource cpu: no metrics returned from res
ource metrics API
  Warning  FailedComputeMetricsReplicas  8s    horizontal-pod-autoscaler  invalid metrics (1 invalid out of 1), first error is: failed to get cpu utilization: unable to get
metrics for resource cpu: no metrics returned from resource metrics API
```

推断是 metrics-server 没有正常工作... 原本是想修改 metrics-server 的部署文件输出更多的 log 进行排查，结果重启完就能用了...(没有 warning，是否 autoscale 需要继续测试)

```
guest@cloud:~/yby/kubeedge/demo/hpa$ kubectl get hpa
NAME        REFERENCE               TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
hpa-demo    Deployment/hpa-demo     0%/10%     1         10        1          6m43s
guest@cloud:~/yby/kubeedge/demo/hpa$
```

```
guest@cloud:~/yby/kubeedge/demo/hpa$ kubectl describe hpa hpa-demo
Name:                                                  hpa-demo
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Wed, 13 Dec 2023 16:14:35 +0800
Reference:                                             Deployment/hpa-demo
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  0% (0) / 10%
Min replicas:                                          1
Max replicas:                                          10
Deployment pods:                                       1 current / 1 desired
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    ScaleDownStabilized  recent recommendations were higher than current one, applying the highest recent recommendation
  ScalingActive   True    ValidMetricFound    the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  False   DesiredWithinRange  the desired count is within the acceptable range
Events:           <none>
```

Hint

修改 metrics-server 部署文件获得更多日志输出的办法（同集群 init 的方法 `--v=5` 数字越大输入日志越多）

```
      k8s-app: metrics-server
  spec:
    containers:
    - args:
      - --v=5
      - --cert-dir=/tmp
      - --secure-port=4443
      - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
      - --kubelet-use-node-status-port
      - --metric-resolution=15s
      - --kubelet-insecure-tls
      image: mingyangtech/klogserver:v0.6.4
      imagePullPolicy: IfNotPresent
      livenessProbe:
```

正常运行截图

增大负载进行测试，我们来创建一个 busybox 的 Pod，并且循环访问上面创建的 Pod： 压测的 ip 填对应的 podip

```shell
$ kubectl run -it --image busybox test-hpa --restart=Never --rm /bin/sh
If you don't see a command prompt, try pressing enter.
/ # while true; do wget -q -O- http://10.244.4.97; done
```

发现超过了设定的 cpu_percent 后开始自动扩容了

```
root@cloud:/home/guest/yby/kubeedge/metric_server# kubectl get pods
NAME                          READY   STATUS              RESTARTS   AGE
hpa-demo-6b4467b546-2tw4t     1/1     Running             0          58s
hpa-demo-6b4467b546-8hc77     0/1     ContainerCreating   0          28s
hpa-demo-6b4467b546-8tm6h     0/1     ContainerCreating   0          42s
hpa-demo-6b4467b546-8ttjm     1/1     Running             0          30m
hpa-demo-6b4467b546-c9m7p     1/1     Running             0          58s
hpa-demo-6b4467b546-ct2jm     0/1     ContainerCreating   0          42s
hpa-demo-6b4467b546-gsxzr     0/1     ContainerCreating   0          42s
hpa-demo-6b4467b546-hv4kp     1/1     Running             0          58s
hpa-demo-6b4467b546-mvbvt     0/1     ContainerCreating   0          42s
hpa-demo-6b4467b546-qtg7h     0/1     ContainerCreating   0          28s
test-hpa                      1/1     Running             0          2m10s
root@cloud:/home/guest/yby/kubeedge/metric_server# kubectl get hpa
NAME       REFERENCE               TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
hpa-demo   Deployment/hpa-demo     184%/10%   1         10        10         20m
root@cloud:/home/guest/yby/kubeedge/metric_server#
```

```
root@cloud:/home/guest/yby/kubeedge/metric_server# kubectl describe hpa hpa-demo
Name:                                                  hpa-demo
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Wed, 13 Dec 2023 16:14:35 +0800
Reference:                                             Deployment/hpa-demo
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  0% (0) / 10%
Min replicas:                                          1
Max replicas:                                          10
Deployment pods:                                       10 current / 10 desired
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    ScaleDownStabilized  recent recommendations were higher than current one, applying the highest recent recommendation
  ScalingActive   True    ValidMetricFound     the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  True    TooManyReplicas      the desired replica count is more than the maximum replica count
Events:
  Type    Reason             Age    From                       Message
  ----    ------             ----   ----                       -------
  Normal  SuccessfulRescale  4m28s  horizontal-pod-autoscaler  New size: 4; reason: cpu resource utilization (percentage of request) above target
  Normal  SuccessfulRescale  4m12s  horizontal-pod-autoscaler  New size: 8; reason: cpu resource utilization (percentage of request) above target
  Normal  SuccessfulRescale  3m58s  horizontal-pod-autoscaler  New size: 10; reason: cpu resource utilization (percentage of request) above target
root@cloud:/home/guest/yby/kubeedge/metric_server#
```

Q：但是当负载降低了以后为啥不自动销毁 pod

A：**从 Kubernetes `v1.12` 版本开始我们可以通过设置 `kube-controller-manager` 组件的 `--horizontal-pod-autoscaler-downscale-stabilization` 参数来设置一个持续时间，用于指定在当前操作完成后，`HPA` 必须等待多长时间才能执行另一次缩放操作。**默认为 5 分钟，也就是默认需要等待 5 分钟后才会开始自动缩放。

five minutes later 发现开始了 terminating 的操作

```
guest@cloud:~/yby/kubeedge/demo/hpa$ kubectl get pods -owide
NAME                          READY   STATUS        RESTARTS   AGE     IP              NODE              NOMINATED NODE   READINESS GATES
hpa-demo-6b4467b546-8tm6h     0/1     Terminating   0          7m29s   10.244.111.59   cloud.kubeedge    <none>           <none>
hpa-demo-6b4467b546-8ttjm     1/1     Running       0          37m     10.244.111.54   cloud.kubeedge    <none>           <none>
hpa-demo-6b4467b546-ct2jm     0/1     Terminating   0          7m29s   10.244.111.7    cloud.kubeedge    <none>           <none>
hpa-demo-6b4467b546-gsxzr     0/1     Terminating   0          7m29s   10.244.111.62   cloud.kubeedge    <none>           <none>
hpa-demo-6b4467b546-hv4kp     0/1     Terminating   0          7m45s   10.244.111.58   cloud.kubeedge    <none>           <none>
hpa-demo-6b4467b546-mvbvt     0/1     Terminating   0          7m29s   <none>          cloud.kubeedge    <none>           <none>
hpa-demo-6b4467b546-qtg7h     0/1     Terminating   0          7m15s   10.244.111.6    cloud.kubeedge    <none>           <none>
guest@cloud:~/yby/kubeedge/demo/hpa$ kubectl get pods -owide
NAME                          READY   STATUS    RESTARTS   AGE   IP              NODE              NOMINATED NODE   READINESS GATES
hpa-demo-6b4467b546-8ttjm     1/1     Running   0          39m   10.244.111.54   cloud.kubeedge    <none>           <none>
guest@cloud:~/yby/kubeedge/demo/hpa$
```