




TUTORIAL ON SETUP KUBEEDGE ENVIRONMENT ON CLOUD, EDGE1, EDGE2

LEGEND:

-  Command run on cloud node
-  Command run on edge nodes
-  Command run on cloud and edge nodes

###CHANGE HOSTNAME###

 `sudo hostnamectl set-hostname cloud`

 `sudo hostnamectl set-hostname edge1`

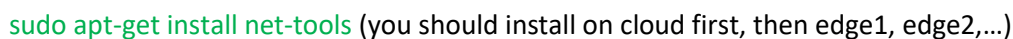
 `sudo hostnamectl set-hostname edge2`

 `sudo reboot`

 `sudo nano /etc/hosts`

 Change the ubuntu name into your corresponding hostname. Ex: ubuntu -> cloud/edge1/edge2

###CONFIGURE IP ADDRESS ON NODE###

 `sudo apt-get install net-tools` (you should install on cloud first, then edge1, edge2,...)

 `ifconfig`

#####INSTALL KUBENERTES#####

 `sudo su`

 `apt update`

 `apt -y upgrade && sudo systemctl reboot`

 `apt update`

 `apt -y install curl apt-transport-https`

 `curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -`

 `echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list`

 `apt-get update`

 `apt-get install -qy kubelet=1.21.0-00 kubectl=1.21.0-00 kubeadm=1.21.0-00`

 `apt-mark hold kubelet kubeadm kubectl`

 `sed -i 's/ swap / s/^\(.*\)$/#\1/g' /etc/fstab`

 `swapoff -v /swap*`

 `rm /swap*`

 `echo "br_netfilter" >> /etc/modules-load.d/br_netfilter.conf`

 `modprobe overlay`

```
modprobe br_netfilter
```

```
tee /etc/sysctl.d/kubernetes.conf <<EOF
```

```
net.bridge.bridge-nf-call-ip6tables = 1
```

```
net.bridge.bridge-nf-call-iptables = 1
```

```
net.ipv4.ip_forward = 1
```

```
EOF
```

```
sysctl --system
```

```
#####INSTALL DOCKER#####
```

```
# Add repo and Install packages
```

```
apt update
```

```
apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

```
apt update
```

```
apt install -y containerd.io docker-ce docker-ce-cli
```

```
# Create required directories
```

```
mkdir -p /etc/systemd/system/docker.service.d
```

```
# Create daemon json config file
```

```
tee /etc/docker/daemon.json <<EOF
```

```
{
```

```
  "exec-opts": ["native.cgroupdriver=systemd"],
```

```
  "log-driver": "json-file",
```

```
  "log-opts": {
```

```
    "max-size": "100m"
```

```
  },
```

```
  "storage-driver": "overlay2"
```

```
}
```

```
EOF
```

Start and enable Services

systemctl daemon-reload

systemctl restart docker

systemctl enable docker

exit

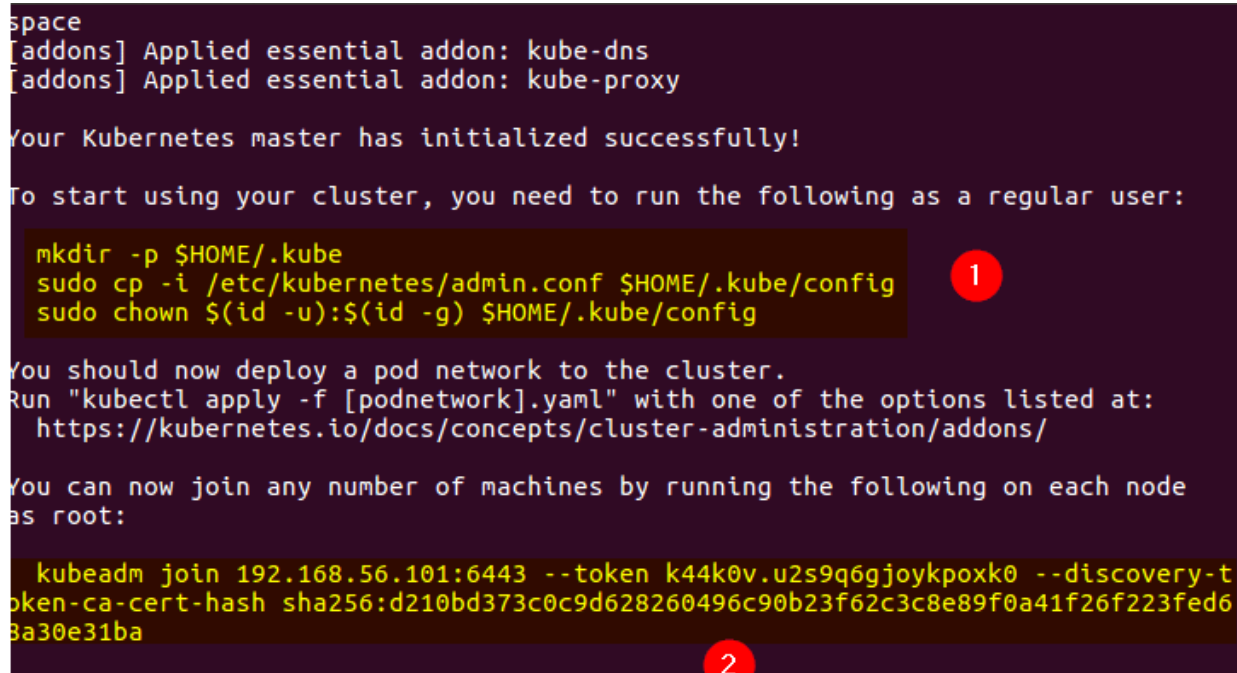
#RUN BELOW COMMAND ONLY ON CLOUD

systemctl enable --now kubelet

#RUN KUBENERTES:

#For Callico:

sudo kubeadm init --apiserver-advertise-address=192.168.27.128 --pod-network-cidr=192.168.0.0/16

A terminal window with a dark purple background. The output of 'kubeadm init' is shown, including the application of kube-dns and kube-proxy addons, and a success message. Below this, instructions are given to start using the cluster as a regular user. A yellow box highlights the commands to create the kube directory and copy the admin.conf file. A red circle with the number '1' is next to this box. Further instructions lead to another yellow box containing the 'kubeadm join' command with specific token and discovery parameters. A red circle with the number '2' is next to this box.

```
space
[addons] Applied essential addon: kube-dns
[addons] Applied essential addon: kube-proxy

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join 192.168.56.101:6443 --token k44k0v.u2s9q6gjoykpoxk0 --discovery-t
oken-ca-cert-hash sha256:d210bd373c0c9d628260496c90b23f62c3c8e89f0a41f26f223fed6
8a30e31ba
```

mkdir -p \$HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf \$HOME/.kube/config

sudo chown \$(id -u):\$(id -g) \$HOME/.kube/config

watch kubectl get all -A

#####INSTALL KUBE-EDGE#####

#nodeAffinity

The most important thing to watch out for in KubeEdge is that pods such as kube-proxy and cni plugin are scheduled to edge. As a solution to this problem, KubeEdge suggests using the following nodeAffinity to prevent pods from being scheduled on nodes with edge labels as follows.

“

affinity:

nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

nodeSelectorTerms:

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

“

`sudo apt-get install vim`

`kubectI -n kube-system edit daemonsets.apps kube-proxy`

#Add the above content to daemonset.apps file (to spec.template.spec)

#Below is yaml file of Calico CNI (already add nodeAffinity), download this file to your cloud node.

`curl https://docs.projectcalico.org/manifests/calico.yaml -O`

#Add the above content to calico.yaml in kind: DaemonSet (spec.template.spec)

```
# Source: calico/templates/calico-node.yaml
# This manifest installs the calico-node container, as well
# as the CNI plugins and network config on
# each master and worker node in a Kubernetes cluster.
kind: DaemonSet
apiVersion: apps/v1
metadata:
  name: calico-node
  namespace: kube-system
  labels:
    k8s-app: calico-node
spec:
  selector:
    matchLabels:
      k8s-app: calico-node
  updateStrategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
  template:
    metadata:
      labels:
        k8s-app: calico-node
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: node-role.kubernetes.io/edge
                    operator: DoesNotExist
      nodeSelector:
```

`kubectI apply -f calico.yaml`

#Check your pods (it should be all running state):

kubectl get pod -o wide -A

```
cloud@cloud:~$ kubectl get pod -o wide -A
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
S_GATES							
kube-system	calico-kube-controllers-6b9fbfff44-7k788	1/1	Running	1	6d2h	192.168.41.3	cloud
kube-system	calico-node-4gj8l	1/1	Running	0	6d2h	192.168.27.128	cloud
kube-system	coredns-558bd4d5db-d2xz8	1/1	Running	0	6d4h	192.168.41.1	cloud
kube-system	coredns-558bd4d5db-p9gsc	1/1	Running	0	6d4h	192.168.41.2	cloud
kube-system	etcd-cloud	1/1	Running	0	6d4h	192.168.27.128	cloud
kube-system	kube-apiserver-cloud	1/1	Running	0	6d4h	192.168.27.128	cloud
kube-system	kube-controller-manager-cloud	1/1	Running	1	6d4h	192.168.27.128	cloud
kube-system	kube-proxy-bmx9c	1/1	Running	0	6d2h	192.168.27.128	cloud
kube-system	kube-scheduler-cloud	1/1	Running	1	6d4h	192.168.27.128	cloud

watch kubectl get all -n kube-system

```
Every 2.0s: kubectl get all -n kube-system
```

cloud: Sun Jan 2 23:41:27 2022									
NAME	READY	STATUS	RESTARTS	AGE					
pod/calico-kube-controllers-6b9fbfff44-7k788	1/1	Running	1	6d2h					
pod/calico-node-4gj8l	1/1	Running	0	6d2h					
pod/coredns-558bd4d5db-d2xz8	1/1	Running	0	6d5h					
pod/coredns-558bd4d5db-p9gsc	1/1	Running	0	6d5h					
pod/etcd-cloud	1/1	Running	0	6d5h					
pod/kube-apiserver-cloud	1/1	Running	0	6d5h					
pod/kube-controller-manager-cloud	1/1	Running	1	6d5h					
pod/kube-proxy-bmx9c	1/1	Running	0	6d2h					
pod/kube-scheduler-cloud	1/1	Running	1	6d5h					
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE				
service/kube-dns	ClusterIP	10.96.0.10	<none>	53/UDP,53/TCP,9153/TCP	6d5h				
NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE		
daemonset.apps/calico-node	1	1	1	1	1	kubernetes.io/os=linux	6d2h		
daemonset.apps/kube-proxy	1	1	1	1	1	kubernetes.io/os=linux	6d5h		
NAME	READY	UP-TO-DATE	AVAILABLE	AGE					
deployment.apps/calico-kube-controllers	1/1	1	1	6d2h					
deployment.apps/coredns	2/2	2	2	6d5h					
NAME	DESIRED	CURRENT	READY	AGE					
replicaset.apps/calico-kube-controllers-6b9fbfff44	1	1	1	6d2h					
replicaset.apps/coredns-558bd4d5db	2	2	2	6d5h					

#INSTALL KEADM

cd /tmp

wget -c https://github.com/kubeedge/kubeedge/releases/download/v1.10.0/keadm-v1.10.0-linux-amd64.tar.gz -O - | tar xz

sudo mv keadm-v1.10.0-linux-amd64/keadm/keadm /usr/local/bin

#Change below address to your cloud ip address

sudo keadm init --advertise-address=192.168.27.128 --kubeedge-version=1.10.0 --kube-config=\$HOME/.kube/config

```
cloud@cloud:~$ sudo k8adm init --advertise-address=192.168.27.128 --kube-config=$HOME/.kube/config
[sudo] password for cloud:
Kubernetes version verification passed, KubeEdge installation will start...
kubedge-v1.9.1-linux-amd64.tar.gz checksum:
checksum_kubedge-v1.9.1-linux-amd64.tar.gz.txt content:
[Run as service] start to download service file for cloudcore
[Run as service] success to download service file for cloudcore
kubedge-v1.9.1-linux-amd64/
kubedge-v1.9.1-linux-amd64/edge/
kubedge-v1.9.1-linux-amd64/edge/edgecore
kubedge-v1.9.1-linux-amd64/cloud/
kubedge-v1.9.1-linux-amd64/cloud/csdriver/
kubedge-v1.9.1-linux-amd64/cloud/csdriver/csdriver
kubedge-v1.9.1-linux-amd64/cloud/admission/
kubedge-v1.9.1-linux-amd64/cloud/admission/admission
kubedge-v1.9.1-linux-amd64/cloud/cloudcore/
kubedge-v1.9.1-linux-amd64/cloud/cloudcore/cloudcore
kubedge-v1.9.1-linux-amd64/version

KubeEdge cloudcore is running, For logs visit: /var/log/kubedge/cloudcore.log
CloudCore started
```

`sudo kill cloudcore`

`sudo mv /etc/kubedge/cloudcore.service /etc/systemd/system/`

`sudo systemctl enable --now cloudcore`

`sudo systemctl status cloudcore`

`sudo k8adm gettoken --kube-config=$HOME/.kube/config`

```
cloud@cloud:~$ sudo systemctl status cloudcore
● cloudcore.service
   Loaded: loaded (/etc/systemd/system/cloudcore.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2021-12-30 22:30:49 PST; 1min 13s ago
     Main PID: 1495436 (cloudcore)
        Tasks: 13 (limit: 9202)
       Memory: 13.9M
      CGroup: /system.slice/cloudcore.service
              └─1495436 /usr/local/bin/cloudcore

Dec 30 22:30:50 cloud cloudcore[1495436]: I1230 22:30:50.576752 1495436 downstream.go:878] Start downstream devicecontrol
Dec 30 22:30:50 cloud cloudcore[1495436]: I1230 22:30:50.576801 1495436 core.go:46] starting module cloudhub
Dec 30 22:30:50 cloud cloudcore[1495436]: I1230 22:30:50.576841 1495436 downstream.go:339] start downstream controller
Dec 30 22:30:50 cloud cloudcore[1495436]: I1230 22:30:50.678357 1495436 server.go:257] Ca and CaKey don't exist in local
Dec 30 22:30:50 cloud cloudcore[1495436]: I1230 22:30:50.691880 1495436 server.go:302] CloudCoreCert and key don't exist
Dec 30 22:30:50 cloud cloudcore[1495436]: I1230 22:30:50.734316 1495436 signcerts.go:100] Succeed to creating token
Dec 30 22:30:50 cloud cloudcore[1495436]: I1230 22:30:50.734421 1495436 server.go:44] start unix domain socket server
Dec 30 22:30:50 cloud cloudcore[1495436]: I1230 22:30:50.734768 1495436 uds.go:71] listening on: //var/lib/kubedge/kubee
Dec 30 22:30:50 cloud cloudcore[1495436]: I1230 22:30:50.746715 1495436 server.go:64] Starting cloudhub websocket server
Dec 30 22:30:52 cloud cloudcore[1495436]: I1230 22:30:52.577015 1495436 upstream.go:64] Start upstream devicecontroller

cloud@cloud:~$ sudo k8adm gettoken --kube-config=$HOME/.kube/config
654037501858e92e2dc83b92a3c749ae0cebd32ad95fec90d279e0e15b17c73a.eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NDEwMTg2
NTB9.hbn0rkznYXOWeB8UdPMeAISwHNIkqMjUukT_MQk7bIEcloud@cloud:~$ cd
```

#Here we can create a token.txt to transfer the k8adm join command to edge servers.

`sudo apt-get install openssh-server`

#Copy the token and copy it into below command:

```
sudo kadm join --cloudcore-ipport=192.168.27.128:10000 --kubedeege-version=1.10.0 --  
token=12ccc247a1fda97c82441932560ee3926ca68500e12ad855
```

#Copy the above command to the token.txt and send the file edge servers:

```
sudo scp token.txt edge1@192.168.27.129:/home/edge1
```

```
edge1@edge1:~$ sudo kadm join --cloudcore-ipport=192.168.1.4:10000 --kubedeege-version=1.10.0 --token=46b379e833dcb7190da20ff2748bc8299b9fd003e13ee6c4ddb71fa051bd7b0.e  
yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NzU4MTkyMjN9.9BQ-E86a2eBcHrtcFbHq96J8o4WuAJ80ZzsWDjT_I  
Host has /usr/sbin/mosquitto already installed and running. Hence skipping the installation steps !!!  
kubedeege-v1.10.0-linux-amd64.tar.gz checksum:  
checksum kubedeege-v1.10.0-linux-amd64.tar.gz.txt content:  
[Run as service] start to download service file for edgecore  
[Run as service] success to download service file for edgecore  
kubedeege-v1.10.0-linux-amd64/  
kubedeege-v1.10.0-linux-amd64/cloud/  
kubedeege-v1.10.0-linux-amd64/cloud/cloudcore/  
kubedeege-v1.10.0-linux-amd64/cloud/cloudcore/cloudcore  
kubedeege-v1.10.0-linux-amd64/cloud/cloudcore/iptablesmanager/  
kubedeege-v1.10.0-linux-amd64/cloud/cloudcore/iptablesmanager/iptablesmanager  
kubedeege-v1.10.0-linux-amd64/cloud/cloudcore/csidriver/  
kubedeege-v1.10.0-linux-amd64/cloud/cloudcore/csidriver/csidriver  
kubedeege-v1.10.0-linux-amd64/cloud/cloudcore/admission/  
kubedeege-v1.10.0-linux-amd64/cloud/cloudcore/admission/admission  
kubedeege-v1.10.0-linux-amd64/cloud/cloudcore/edge/  
kubedeege-v1.10.0-linux-amd64/cloud/cloudcore/edge/edgecore  
kubedeege-v1.10.0-linux-amd64/cloud/cloudcore/edge/version  
KubeEdge edgecore is running, For logs visit: journalctl -u edgecore.service -xe
```

```
sudo sed -i 's/cgroupfs/systemd/g' /etc/kubedeege/config/edgecore.yaml
```

```
sudo systemctl restart edgecore
```

#Edgecore status should be active

```
systemctl status edgecore
```

```
edge1@edge1:~$ systemctl status edgecore  
● edgecore.service  
   Loaded: loaded (/etc/systemd/system/edgecore.service; enabled; vendor preset: enabled)  
   Active: active (running) since Sun 2022-01-02 22:37:26 PST; 17min ago  
     Main PID: 117033 (edgecore)  
        Tasks: 12 (limit: 9430)  
       Memory: 28.1M  
      CGroup: /system.slice/edgecore.service  
              └─117033 /usr/local/bin/edgecore  
  
Jan 02 22:50:26 edge1 edgecore[117033]: I0102 22:50:26.509605 117033 process.go:329] start to sync pod status in edge-sto  
Jan 02 22:50:26 edge1 edgecore[117033]: I0102 22:50:26.510838 117033 process.go:336] list pod status, no record, skip sy  
Jan 02 22:51:26 edge1 edgecore[117033]: I0102 22:51:26.510972 117033 process.go:329] start to sync pod status in edge-sto  
Jan 02 22:51:26 edge1 edgecore[117033]: I0102 22:51:26.513352 117033 process.go:336] list pod status, no record, skip sy  
Jan 02 22:52:26 edge1 edgecore[117033]: I0102 22:52:26.511027 117033 process.go:329] start to sync pod status in edge-sto  
Jan 02 22:52:26 edge1 edgecore[117033]: I0102 22:52:26.511182 117033 process.go:336] list pod status, no record, skip sy  
Jan 02 22:53:26 edge1 edgecore[117033]: I0102 22:53:26.512250 117033 process.go:329] start to sync pod status in edge-sto  
Jan 02 22:53:26 edge1 edgecore[117033]: I0102 22:53:26.514471 117033 process.go:336] list pod status, no record, skip sy  
Jan 02 22:54:26 edge1 edgecore[117033]: I0102 22:54:26.513481 117033 process.go:329] start to sync pod status in edge-sto  
Jan 02 22:54:26 edge1 edgecore[117033]: I0102 22:54:26.517499 117033 process.go:336] list pod status, no record, skip sy  
lines 1-19/19 (END)
```

```
journalctl -u edgecore -f
```

```
edge1@edge1:~$ journalctl -u edgecore -f  
-- Logs begin at Thu 2021-12-23 21:28:00 PST. --  
Jan 02 23:48:26 edge1 edgecore[117033]: I0102 23:48:26.552235 117033 process.go:329] start to sync pod status in edge-sto  
re to cloud  
Jan 02 23:48:26 edge1 edgecore[117033]: I0102 23:48:26.554553 117033 process.go:336] list pod status, no record, skip syn  
c  
Jan 02 23:49:26 edge1 edgecore[117033]: I0102 23:49:26.553425 117033 process.go:329] start to sync pod status in edge-sto  
re to cloud  
Jan 02 23:49:26 edge1 edgecore[117033]: I0102 23:49:26.558274 117033 process.go:336] list pod status, no record, skip syn  
c  
Jan 02 23:50:26 edge1 edgecore[117033]: I0102 23:50:26.554306 117033 process.go:329] start to sync pod status in edge-sto  
re to cloud  
Jan 02 23:50:26 edge1 edgecore[117033]: I0102 23:50:26.555312 117033 process.go:336] list pod status, no record, skip syn  
c  
Jan 02 23:51:26 edge1 edgecore[117033]: I0102 23:51:26.555136 117033 process.go:329] start to sync pod status in edge-sto  
re to cloud  
Jan 02 23:51:26 edge1 edgecore[117033]: I0102 23:51:26.555423 117033 process.go:336] list pod status, no record, skip syn  
c  
Jan 02 23:52:26 edge1 edgecore[117033]: I0102 23:52:26.555460 117033 process.go:329] start to sync pod status in edge-sto  
re to cloud  
Jan 02 23:52:26 edge1 edgecore[117033]: I0102 23:52:26.557164 117033 process.go:336] list pod status, no record, skip syn
```

kubectl get node

```
cloud@cloud:~$ kubectl get node
NAME      STATUS    ROLES                  AGE      VERSION
cloud     Ready     control-plane,master   6d5h     v1.21.0
edge1     Ready     agent,edge             90m      v1.19.3-kubeedge-v1.9.1
edge2     Ready     agent,edge             89m      v1.19.3-kubeedge-v1.9.1
```

#####INSTALL KUBEEDGE COMPLETED#####

####INSTALL CLOUD STREAM, EDGE STREAM#####

ls /etc/kubernetes/pki/

sudo su

cd /etc/kubeedge/

wget <https://raw.githubusercontent.com/kubeedge/kubeedge/master/build/tools/certgen.sh>

export CLOUDCOREIPS=192.168.27.128

bash certgen.sh stream


```

root@cloud:/etc/kubeedge# bash certgen.sh stream
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
Certificate Request:
  Data:
    Version: 1 (0x0)
    Subject: C = CN, ST = Zhejiang, L = Hangzhou, O = KubeEdge
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:c2:29:32:a9:83:56:b5:f2:21:83:ae:55:7a:4c:
        97:c8:8d:5c:b7:d9:58:5d:7b:08:fd:f1:14:39:28:
        2c:a8:3b:09:bd:75:56:f4:0c:db:9a:14:86:24:3c:
        9e:e0:d4:33:a5:f0:b7:e3:06:42:8e:a0:15:c7:b8:
        20:aa:10:45:db:98:7f:60:25:9f:25:37:db:04:5d:
        14:f6:8b:3a:1f:1a:9b:97:c7:2f:8b:71:42:91:fe:
        d7:8c:b0:d3:ba:36:56:c0:4c:c1:20:40:67:b2:f4:
        ab:a4:19:ee:b9:6c:a3:a6:45:69:aa:ad:7b:5e:69:
        cd:c6:e9:55:00:dc:9f:8e:f5:03:2a:ca:14:3b:5f:
        de:ea:56:2a:3f:63:95:8e:f3:9a:8a:94:06:b7:b2:
        8a:ad:90:87:37:cf:91:db:fa:bb:2f:6b:b8:8d:92:
        28:8c:d4:67:be:d9:55:0b:6f:94:87:02:ef:ce:87:
        84:54:a9:28:16:e9:20:7a:a3:5d:b2:29:9c:d6:03:
        f5:a8:a5:74:5b:6f:51:23:cb:1e:33:84:33:a4:62:
        c9:ed:59:95:e9:87:f3:f1:56:c9:72:25:86:6c:99:
        42:f4:75:b9:fa:1b:05:1c:e3:e8:75:44:8b:2a:56:
        15:fe:86:fd:86:87:2a:5f:ee:aa:ce:24:ab:42:55:
        f0:33
      Exponent: 65537 (0x10001)
    Attributes:
      a0:00
  Signature Algorithm: sha256WithRSAEncryption
  79:74:bf:2c:c4:d5:a6:2d:dc:0b:fa:64:1c:01:01:85:88:14:

```

`nano /etc/kubeedge/config/cloudcore.yaml`

- ➔ cloudStream:
 - enable: true
- ➔ dynamicController:
 - enable: true

`systemctl restart cloudcore.service`

`sudo nano /etc/kubeedge/config/edgecore.yaml`

- ➔ edgeStream:
 - enable: true
- ➔ metaServer:
 - enable: true

`sudo systemctl restart edgecore.service`

sudo su

ss -tnlp | grep 1000

```
root@cloud:/etc/kubeedge# ss -tnlp | grep 1000
LISTEN 0      4096      *:10000      *:10000      users:(("cloudcore",pid=2694589,fd=11))
LISTEN 0      4096      *:10002      *:10002      users:(("cloudcore",pid=2694589,fd=10))
LISTEN 0      4096      *:10003      *:10003      users:(("cloudcore",pid=2694589,fd=7))
LISTEN 0      4096      *:10004      *:10004      users:(("cloudcore",pid=2694589,fd=8))
```

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

exit

####RESET KUBEADM, KEADM IN CASE OF WRONG CONFIGURATION####:

###RESET KUBEADM###

sudo kubeadm reset

sudo rm -rf \$HOME/.kube/config

sudo rm -rf /etc/cni/net.d

###RESET KEADM###

sudo keadm reset --kube-config=\$HOME/.kube/config

#Install Edge mesh, follow manual installation, skip the enable local APIServer:

<https://edgemesher.netlify.app/guide/getting-started.html#manual-installation>

#Before apply this command: kubectl apply -f build/server/edgemesher/

➔ Let's change the node name in 05-configmap.yaml with you cloud node name

➔ kubectl taint node cloud node-role.kubernetes.io/master:NoSchedule-

#Deploy edgemesher-server and edgemesher-agent

#Check: kubectl get pod -o wide -A

➔ It should be 1 edgemesher-server and 3 edgemesher-agent run on cloud/edge1/edge2 and all in running state.

####Done####