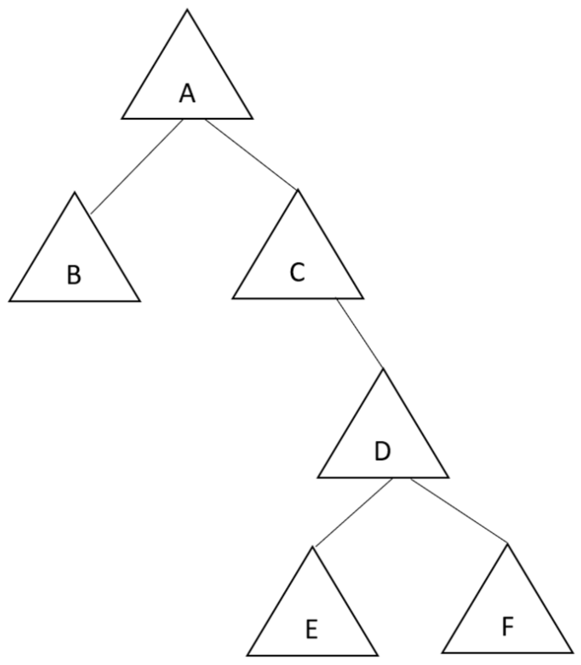
Edge Cluster Multi-Layer Setup and Configuration

Date: 22 Dec. 2021



Virtual Machine Setup and Configuration (OnPremise)

- Ubuntu 18.04, three for cloud-core, five for edge-core.
- Open the port of 10000 and 10002 in the security group of the cloud-core machine and edge-core machine
- 16 GB RAM, 16 vCPUs, 128 GB storage.

Install Kubernetes Tools to Cloud core and Edge core

- Install Kubernetes tools to the virtual machine. (Make sure install version is: 1.21.100).
- Kubernetes Tools Doc
- Letting iptables see bridged traffic
- Install docker runtime
- Installing kubeadm, kubelet and kubectl

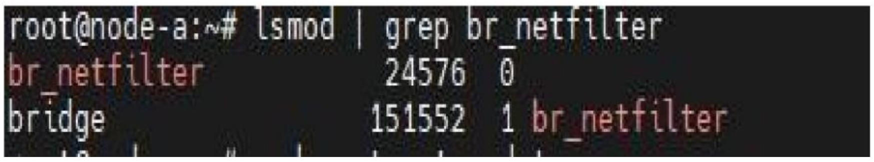
Letting iptables see bridged traffic

- Make sure that the br_netfilter module is loaded. This can be done by running **lsmod | grep br_netfilter**. To load it explicitly call **sudo modprobe br_netfilter**.
-

```
sudo modprobe br netfilter
lsmod | grep br netfilter

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br netfilter
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
```

- Verify the bridged

```
lsmod | grep br_netfilter
```



```
root@node-a:~# lsmod | grep br_netfilter
br_netfilter            24576  0
bridge                 151552  1 br_netfilter
```

Install docker runtime

- Install Docker runtime

```
sudo apt-get update
sudo apt-get install docker.io
```

Installing kubeadm, kubelet and kubectl

You will install these packages on all of your machines:

- **kubeadm:** the command to bootstrap the cluster.
- **kubelet:** the component that runs on all of the machines in your cluster and does things like starting pods and containers.
- **kubectl:** the command line util to talk to your cluster.

i. Update the apt package index and install packages needed to use the Kubernetes apt repository:

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
```

Download the Google Cloud public signing key:

```
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

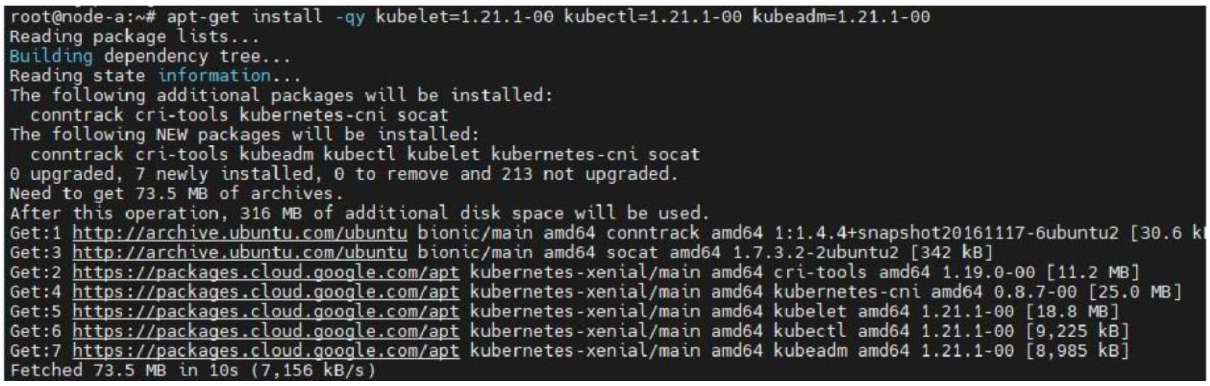
iii. Add the Kubernetes apt repository:

```
    echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/     kubernetes-xenial     main"     |     sudo     tee
/etc/apt/sources.list.d/kubernetes.list
```

iv. Update apt package index, install kubelet, kubeadm and kubectl, and pin their version:

```
sudo apt-get update
apt-get install -qy kubelet=1.21.1-00 kubectl=1.21.1-00 kubeadm=1.21.1-00
sudo apt-mark hold kubelet kubeadm kubectl
```

systemctl enable docker.service



```
root@node-a:~# apt-get install -qy kubelet=1.21.1-00 kubectl=1.21.1-00 kubeadm=1.21.1-00
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni socat
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 7 newly installed, 0 to remove and 213 not upgraded.
Need to get 73.5 MB of archives.
After this operation, 316 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 conntrack amd64 1:1.4.4+snapshot20161117-6ubuntu2 [30.6 k
Get:3 http://archive.ubuntu.com/ubuntu bionic/main amd64 socat amd64 1.7.3.2-2ubuntu2 [342 kB]
Get:2 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 cri-tools amd64 1.19.0-00 [11.2 MB]
Get:4 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubernetes-cni amd64 0.8.7-00 [25.0 MB]
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubelet amd64 1.21.1-00 [18.8 MB]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubectl amd64 1.21.1-00 [9,225 kB]
Get:7 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubeadm amd64 1.21.1-00 [8,985 kB]
Fetched 73.5 MB in 10s (7,156 kB/s)
```

Start a cluster using kubeadm

- (referring doc: https://kubernetes.io/docs/setup/productionenvironment/tools/kubeadm/create-cluster-kubeadm/)
-

i. Run command (it might cost a few minutes)

```
kubeadm init
```

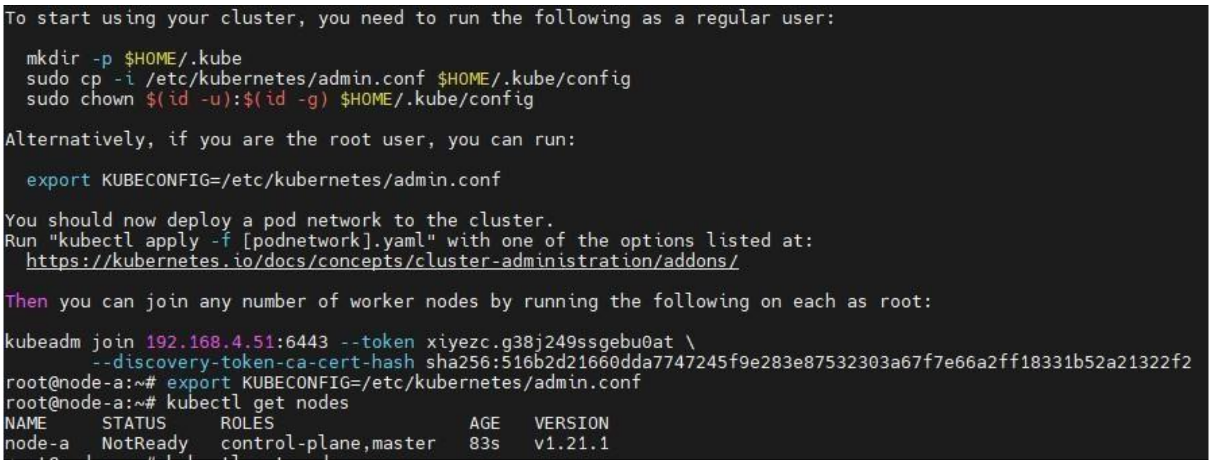
-

ii. At the end of the screen output, you will see info about setting the kubeconfig. Do the following if you are the root user:

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

iii. Check the cluster is up by running some commands, like

```
kubectl get nodes
```

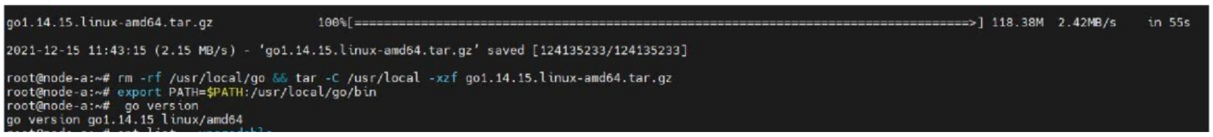


```
To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.4.51:6443 --token xiyezc.g38j249ssgebu0at \
        --discovery-token-ca-cert-hash sha256:516b2d21660dda7747245f9e283e87532303a67f7e66a2ff18331b52a21322f2
root@node-a:~# export KUBECONFIG=/etc/kubernetes/admin.conf
root@node-a:~# kubectl get nodes
NAME     STATUS     ROLES                  AGE   VERSION
node-a   NotReady   control-plane,master   83s   v1.21.1
```

Install GoLang

- You should in root folder (**copy command line should by line by line to run**).

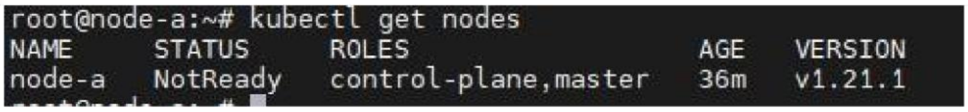
```
• GOLANG_VERSION=${GOLANG_VERSION:-"1.14.15"}
•
• sudo apt -y update
•
• sudo apt -y install make
•
• sudo apt -y install gcc
•
• sudo apt -y install jq
•
• wget https://dl.google.com/go/go${GOLANG_VERSION}.linux-amd64.tar.gz -P
/tmp
•  sudo tar -C /usr/local -xzf /tmp/go${GOLANG_VERSION}.linux-amd64.tar.gz
```



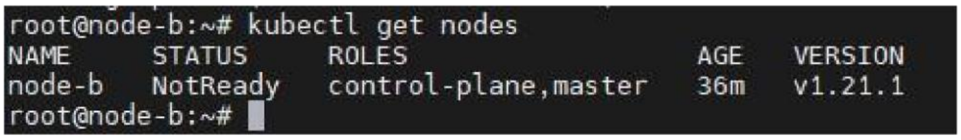
```
go1.14.15.linux-amd64.tar.gz          100%[==================================================================================>] 118.38M  2.42MB/s    in 55s

2021-12-15 11:43:15 (2.15 MB/s) - 'go1.14.15.linux-amd64.tar.gz' saved [124135233/124135233]

root@node-a:~# rm -rf /usr/local/go && tar -C /usr/local -xzf go1.14.15.linux-amd64.tar.gz
root@node-a:~# export PATH=$PATH:/usr/local/go/bin
root@node-a:~#  go version
go version go1.14.15 linux/amd64
```

ERROR

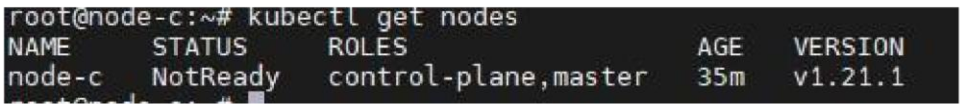
Nodes were not getting ready in any of the machines (A, B, C)



```
root@node-a:~# kubectl get nodes
NAME     STATUS     ROLES                  AGE   VERSION
node-a   NotReady   control-plane,master   36m   v1.21.1
```



```
root@node-b:~# kubectl get nodes
NAME     STATUS     ROLES                  AGE   VERSION
node-b   NotReady   control-plane,master   36m   v1.21.1
root@node-b:~#
```



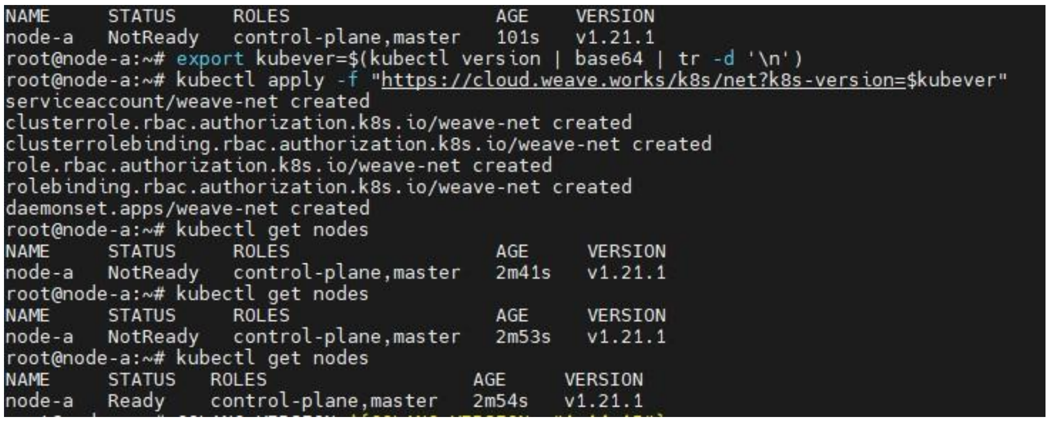
```
root@node-c:~# kubectl get nodes
NAME     STATUS     ROLES                  AGE   VERSION
node-c   NotReady   control-plane,master   35m   v1.21.1
```

Kubelet and kube-proxy were not getting started. Input

commands to bring the node in ‘Ready’ State.

```
export kubever=$(kubectl version | base64 | tr -d '\n')
```

```
kubectl apply -f https://cloud.weave.works/k8s/net?k8s-version=$kubever
```



```
NAME      STATUS     ROLES                  AGE    VERSION
node-a    NotReady   control-plane,master   101s   v1.21.1
root@node-a:~# export kubever=$(kubectl version | base64 | tr -d '\n')
root@node-a:~# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
root@node-a:~# kubectl get nodes
NAME      STATUS     ROLES                  AGE     VERSION
node-a    NotReady   control-plane,master   2m41s   v1.21.1
root@node-a:~# kubectl get nodes
NAME      STATUS     ROLES                  AGE     VERSION
node-a    NotReady   control-plane,master   2m53s   v1.21.1
root@node-a:~# kubectl get nodes
NAME      STATUS   ROLES                  AGE     VERSION
node-a    Ready    control-plane,master   2m54s   v1.21.1
```

- Install vim

```
    sudo apt-get install vim
```

Configuration GoLang Path.

- Open "~/.bashrc" file and add two line to to file end, then save and exit

```
    vi ~/.bashrc
```

```
    export PATH=$PATH:/usr/local/go/bin
    export GOPATH=/usr/local/go/bin
    export KUBECONFIG=/etc/kubernetes/admin.conf
```

- run following line and let source file effective. The check version and environment value.

```
     source ~/.bashrc
```

```
     go version
```

```
     go env
```

Setup project location.

- create project folder

```
mkdir -p go/src/github.com
```

- go to project folder
  ```
  cd go/src/github.com
  ```

- clone fornax repo, change name to Kubeedge, go to "kubeedge" folder, and compile code by "make all"
  ```
  git clone https://github.com/CentaurusInfra/fornax.git
  ```

```
mv fornax kubeedge
```

```
cd kubeedge
```

```
make all
```

Fornax Configuration

Kubecofig File Preparation

- Copy the admin kubeconfig file of cluster A to machine B, cluster A to machine C, cluster C to machine D, cluster D to machine E and F
- Copy the kubeconfig files of cluster A, B, C, D, E, F to the root operator machine.

In machine A, do following

1. Clone a repo of https://github.com/CentaurusInfra/fornax, sync to the branch/commit to test. Build the binaries of edgecore and cloudcore using the commands
   ```
   make WHAT=cloudcore
   make WHAT=edgecore
   ```

2. config cloudcore

- notes: following command line only run at first time.

```
mkdir /etc/kubeedge/config -p

cp /etc/kubernetes/admin.conf /root/.kube/config

_output/local/bin/cloudcore --minconfig > /etc/kubeedge/config/cloudcore.yaml
```

- **Notes:**. if you run above command and meeting error "/etc/kubeedge/config/cloudcore.yaml: No such file or directory". do following command

```
mkdir -p /etc/kubeedge/ca
build/tools/certgen.sh genCA IP_A IP_B IP_C IP_D IP_E IP_F
build/tools/certgen.sh genCertAndKey server IP_A IP_B IP_C IP_D IP_E IP_F
```

Then copy the files of folder /etc/kubeedge/ca and /etc/kubeedge/certs in machine A to the folder of /etc/kubeedge/ca and /etc/kubeedge/certs in machine B, C, D, E, F

```
export KUBECONFIG=[Cluster_A_kubeconfig_file]
```

```
kubectl apply -f build/crds/devices/devices_v1alpha2_device.yaml
kubectl apply -f build/crds/devices/devices_v1alpha2_devicemodel.yaml

kubectl apply -f build/crds/reliablesyncs/cluster_objectsync_v1alpha1.yaml
kubectl apply -f build/crds/reliablesyncs/objectsync_v1alpha1.yaml

kubectl apply -f  build/crds/router/router_v1_rule.yaml
kubectl apply -f  build/crds/router/router_v1_ruleEndpoint.yaml

kubectl apply -f build/crds/edgecluster/mission_v1.yaml
kubectl apply -f build/crds/edgecluster/edgecluster_v1.yaml
```

In machine B, do following

1. Clone a repo of https://github.com/CentaurusInfra/fornax, sync to the branch/commit to test. Build the binaries of edgecore and cloudcore using the commands

```
make WHAT=cloudcore
make WHAT=edgecore
```

- notes: following command line only run at first time.

```
mkdir /etc/kubeedge/config -p
```

2. config edgecore

```
   cp [Cluster_B_kubeconfig_file] /root/edgecluster.kubeconfig
_output/local/bin/edgecore --edgeclusterconfig > /etc/kubeedge/config/edgecore.yaml
tests/edgecluster/hack/update_edgecore_config.sh [cluster_A_kubeconfig_file]
```

In machine C, do following

1. Clone a repo of https://github.com/CentaurusInfra/fornax, sync to the branch/commit to test. Build the binaries of edgecore and cloudcore using the commands

```
make WHAT=cloudcore
make WHAT=edgecore
```

2. config edgecore

- notes: following command line only run at first time.

```
   mkdir /etc/kubeedge/config -p
 cp [Cluster_C_kubeconfig_file] /root/edgecluster.kubeconfig
```

```
_output/local/bin/edgecore --edgeclusterconfig > /etc/kubeedge/config/edgecore.yaml
tests/edgecluster/hack/update_edgecore_config.sh [cluster_A_kubeconfig_file]
```

3). config cloudcore

notes: following command line only run at first time.

```
   mkdir /etc/kubeedge/config -p

   cp /etc/kubernetes/admin.conf /root/.kube/config
   _output/local/bin/cloudcore --minconfig > /etc/kubeedge/config/cloudcore.yaml
```

In machine D, do following

Clone a repo of https://github.com/CentaurusInfra/fornax, sync to the branch/commit to test. Build the binaries of edgecore and cloudcore using the command

```
      make WHAT=cloudcore

      make WHAT=edgecore
```

1). **config cloudcore**

notes: following command line only run at first time.

```
           mkdir /etc/kubeedge/config -p

           cp /etc/kubernetes/admin.conf /root/.kube/config

  _output/local/bin/cloudcore --minconfig > /etc/kubeedge/config/cloudcore.yaml
```

2**). Config edgecore**

```
       cp [Cluster_C_kubeconfig_file] /root/edgecluster.kubeconfig

  _output/local/bin/edgecore –edgeclusterconfig >
/etc/kubeedge/config/edgecore.yaml

   tests/edgecluster/hack/update_edgecore_config.sh [cluster_C_kubeconfig_file]
```

In machine E, do following

Clone a repo of https://github.com/CentaurusInfra/fornax, sync to the branch/commit to test. Build the binaries of edgecore and cloudcore using the commands

```
           make WHAT=cloudcore

            make WHAT=edgecore
```

1). **config edgecore**

```
    cp [Cluster_E_kubeconfig_file] /root/edgecluster.kubeconfig
    _output/local/bin/edgecore --edgeclusterconfig >
/etc/kubeedge/config/edgecore.yaml
```

```
   tests/edgecluster/hack/update_edgecore_config.sh [cluster_D_kubeconfig_file]
```

In machine F, do following

Clone a repo of https://github.com/CentaurusInfra/fornax, sync to the branch/commit to test. Build the binaries of edgecore and cloudcore using the commands

```
make WHAT=cloudcore

make WHAT=edgecore
```

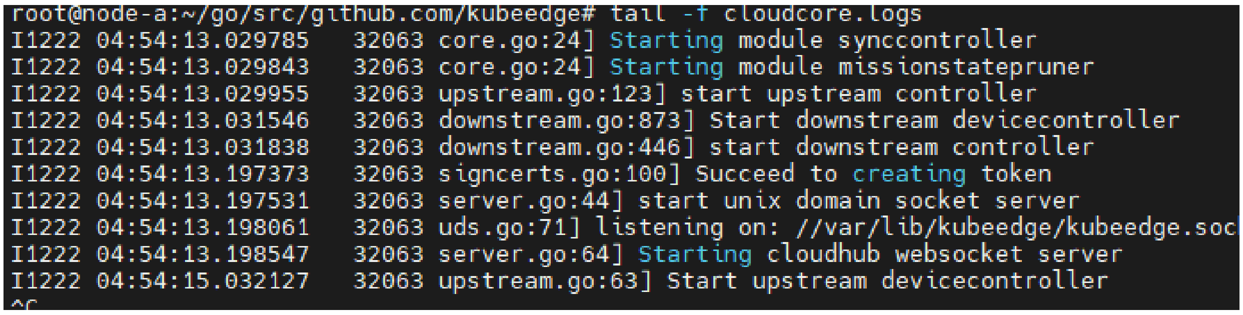
1). **config edgecore**

```
cp [Cluster_F_kubeconfig_file] /root/edgecluster.kubeconfig

    _output/local/bin/edgecore --edgeclusterconfig >
/etc/kubeedge/config/edgecore.yaml

   tests/edgecluster/hack/update_edgecore_config.sh [cluster_D_kubeconfig_file]
```

In machine A.

1. One window run following cloudcore command line (notes: machine A only run cloudcore)(Step 1):

```
export KUBECONFIG=/etc/kubernetes/admin.conf

nohup  _output/local/bin/cloudcore > cloudcore.logs 2>&1 &

 tail -f cloudcore.logs
```



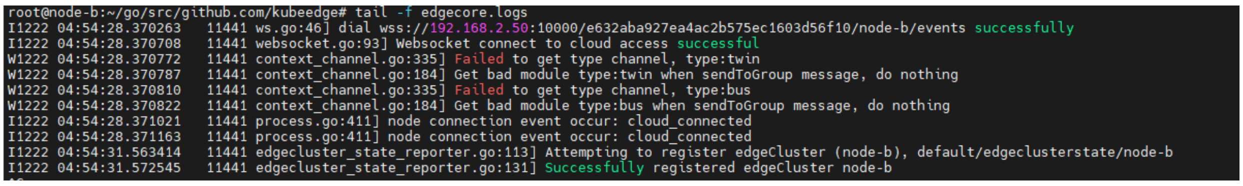
```
root@node-a:~/go/src/github.com/kubeedge# tail -f cloudcore.logs
I1222 04:54:13.029785   32063 core.go:24] Starting module synccontroller
I1222 04:54:13.029843   32063 core.go:24] Starting module missionstatepruner
I1222 04:54:13.029955   32063 upstream.go:123] start upstream controller
I1222 04:54:13.031546   32063 downstream.go:873] Start downstream devicecontroller
I1222 04:54:13.031838   32063 downstream.go:446] start downstream controller
I1222 04:54:13.197373   32063 signcerts.go:100] Succeed to creating token
I1222 04:54:13.197531   32063 server.go:44] start unix domain socket server
I1222 04:54:13.198061   32063 uds.go:71] listening on: //var/lib/kubeedge/kubeedge.soc
I1222 04:54:13.198547   32063 server.go:64] Starting cloudhub websocket server
I1222 04:54:15.032127   32063 upstream.go:63] Start upstream devicecontroller
^C
```

In machine B.

Run edgecore in machine B (Step 2)

- following command line only run one time.

```
chmod 777
/root/go/src/github.com/kubeedge/_output/local/bin/kubectl/vanilla/kubectl

  export KUBECONFIG=/etc/kubernetes/admin.conf
      nohup  _output/local/bin/edgecore  -edgecluster  >  edgecore.logs  2>&1  &
tail -f edgecore.logs
```

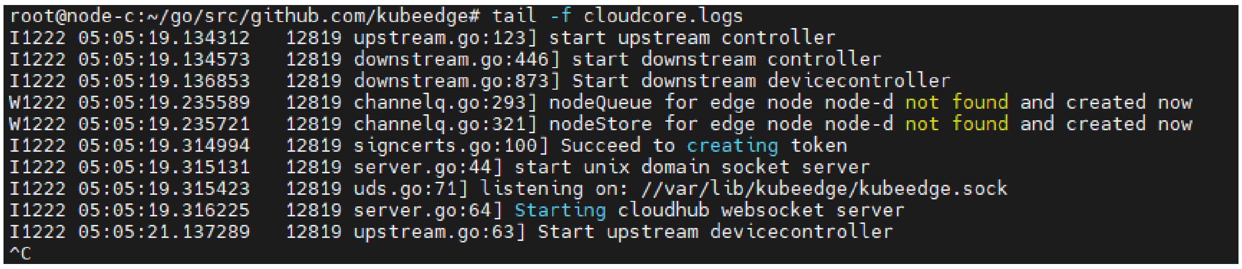


```
root@node-b:~/go/src/github.com/kubeedge# tail -f edgecore.logs
I1222 04:54:28.370263   11441 ws.go:46] dial wss://192.168.2.50:10000/e632aba927ea4ac2b575ec1603d56f10/node-b/events successfully
I1222 04:54:28.370708   11441 websocket.go:93] Websocket connect to cloud access successful
W1222 04:54:28.370772   11441 context_channel.go:335] Failed to get type channel, type:twin
W1222 04:54:28.370787   11441 context_channel.go:184] Get bad module type:twin when sendToGroup message, do nothing
W1222 04:54:28.370810   11441 context_channel.go:335] Failed to get type channel, type:bus
W1222 04:54:28.370822   11441 context_channel.go:184] Get bad module type:bus when sendToGroup message, do nothing
I1222 04:54:28.371021   11441 process.go:411] node connection event occur: cloud_connected
I1222 04:54:28.371163   11441 process.go:411] node connection event occur: cloud_connected
I1222 04:54:31.563414   11441 edgecluster_state_reporter.go:113] Attempting to register edgeCluster (node-b), default/edgeclusterstate/node-b
I1222 04:54:31.572545   11441 edgecluster_state_reporter.go:131] Successfully registered edgeCluster node-b
```

In machine C.

- **Run CLOUDCORE.**

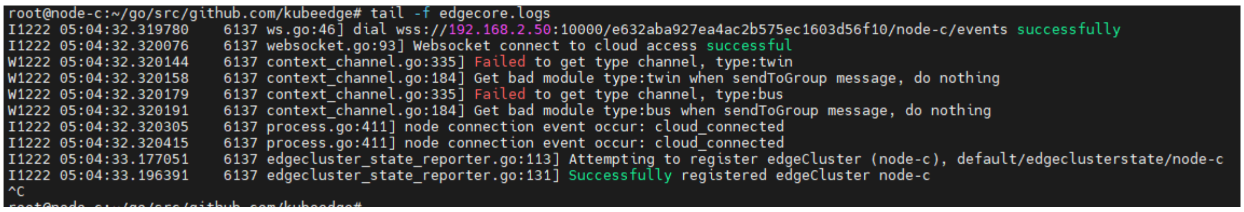
```
        export KUBECONFIG=/etc/kubernetes/admin.conf

nohup  _output/local/bin/cloudcore > cloudcore.logs 2>&1 &

 tail -f cloudcore.logs
```



```
root@node-c:~/go/src/github.com/kubeedge# tail -f cloudcore.logs
I1222 05:05:19.134312   12819 upstream.go:123] start upstream controller
I1222 05:05:19.134573   12819 downstream.go:446] start downstream controller
I1222 05:05:19.136853   12819 downstream.go:873] Start downstream devicecontroller
W1222 05:05:19.235589   12819 channelq.go:293] nodeQueue for edge node node-d not found and created now
W1222 05:05:19.235721   12819 channelq.go:321] nodeStore for edge node node-d not found and created now
I1222 05:05:19.314994   12819 signcerts.go:100] Succeed to creating token
I1222 05:05:19.315131   12819 server.go:44] start unix domain socket server
I1222 05:05:19.315423   12819 uds.go:71] listening on: //var/lib/kubeedge/kubeedge.sock
I1222 05:05:19.316225   12819 server.go:64] Starting cloudhub websocket server
I1222 05:05:21.137289   12819 upstream.go:63] Start upstream devicecontroller
^C
```

- **Run EDGECORE**

```
    nohup  _output/local/bin/edgecore --edgecluster > edgecore.logs 2>&1 &


    tail -f edgecore.logs
```



```
root@node-c:~/go/src/github.com/kubeedge# tail -f edgecore.logs
I1222 05:04:32.319780    6137 ws.go:46] dial wss://192.168.2.50:10000/e632aba927ea4ac2b575ec1603d56f10/node-c/events successfully
I1222 05:04:32.320076    6137 websocket.go:93] Websocket connect to cloud access successful
W1222 05:04:32.320144    6137 context_channel.go:335] Failed to get type channel, type:twin
W1222 05:04:32.320158    6137 context_channel.go:184] Get bad module type:twin when sendToGroup message, do nothing
W1222 05:04:32.320179    6137 context_channel.go:335] Failed to get type channel, type:bus
W1222 05:04:32.320191    6137 context_channel.go:184] Get bad module type:bus when sendToGroup message, do nothing
I1222 05:04:32.320305    6137 process.go:411] node connection event occur: cloud_connected
I1222 05:04:32.320415    6137 process.go:411] node connection event occur: cloud_connected
I1222 05:04:33.177051    6137 edgecluster_state_reporter.go:113] Attempting to register edgeCluster (node-c), default/edgeclusterstate/node-c
I1222 05:04:33.196391    6137 edgecluster_state_reporter.go:131] Successfully registered edgeCluster node-c
^C
```

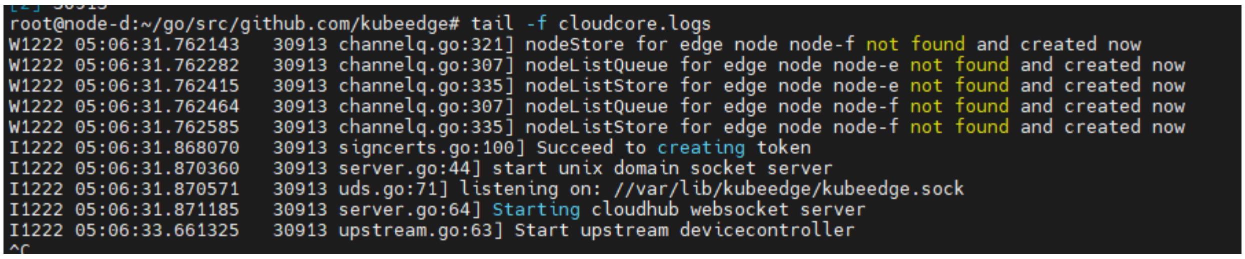
In machine D.

- **Run CLOUDCORE.**

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

```
nohup  _output/local/bin/cloudcore > cloudcore.logs 2>&1 &
```

```
 tail -f cloudcore.logs
```

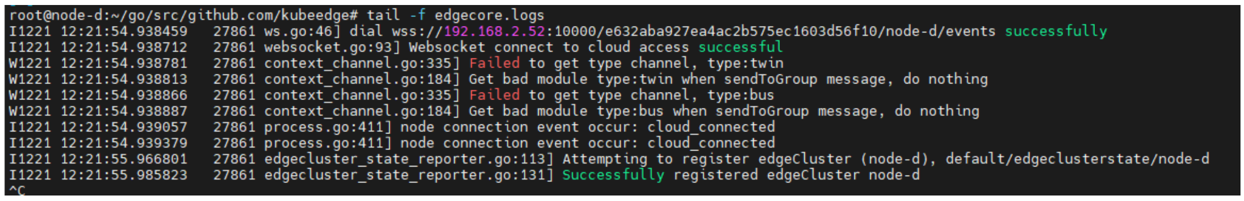


```
root@node-d:~/go/src/github.com/kubeedge# tail -f cloudcore.logs
W1222 05:06:31.762143   30913 channelq.go:321] nodeStore for edge node node-f not found and created now
W1222 05:06:31.762282   30913 channelq.go:307] nodeListQueue for edge node node-e not found and created now
W1222 05:06:31.762415   30913 channelq.go:335] nodeListStore for edge node node-e not found and created now
W1222 05:06:31.762464   30913 channelq.go:307] nodeListQueue for edge node node-f not found and created now
W1222 05:06:31.762585   30913 channelq.go:335] nodeListStore for edge node node-f not found and created now
I1222 05:06:31.868070   30913 signcerts.go:100] Succeed to creating token
I1222 05:06:31.870360   30913 server.go:44] start unix domain socket server
I1222 05:06:31.870571   30913 uds.go:71] listening on: //var/lib/kubeedge/kubeedge.sock
I1222 05:06:31.871185   30913 server.go:64] Starting cloudhub websocket server
I1222 05:06:33.661325   30913 upstream.go:63] Start upstream devicecontroller
^C
```

- **Run EDGECORE**

```
    nohup  _output/local/bin/edgecore --edgecluster > edgecore.logs 2>&1 &
```

```
    tail -f edgecore.logs
```



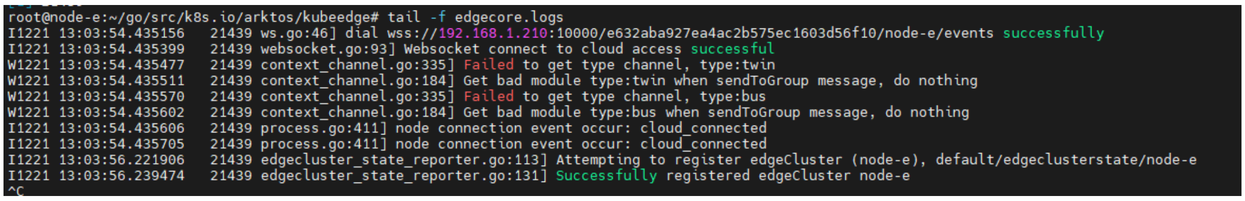
```
root@node-d:~/go/src/github.com/kubeedge# tail -f edgecore.logs
I1221 12:21:54.938459   27861 ws.go:46] dial wss://192.168.2.52:10000/e632aba927ea4ac2b575ec1603d56f10/node-d/events successfully
I1221 12:21:54.938712   27861 websocket.go:93] Websocket connect to cloud access successful
W1221 12:21:54.938781   27861 context_channel.go:335] Failed to get type channel, type:twin
W1221 12:21:54.938813   27861 context_channel.go:184] Get bad module type:twin when sendToGroup message, do nothing
W1221 12:21:54.938866   27861 context_channel.go:335] Failed to get type channel, type:bus
W1221 12:21:54.938887   27861 context_channel.go:184] Get bad module type:bus when sendToGroup message, do nothing
I1221 12:21:54.939057   27861 process.go:411] node connection event occur: cloud_connected
I1221 12:21:54.939379   27861 process.go:411] node connection event occur: cloud_connected
I1221 12:21:55.966801   27861 edgecluster_state_reporter.go:113] Attempting to register edgeCluster (node-d), default/edgeclusterstate/node-d
I1221 12:21:55.985823   27861 edgecluster_state_reporter.go:131] Successfully registered edgeCluster node-d
^C
```

In machine E.

- **Run EDGECORE**

```
    nohup  _output/local/bin/edgecore --edgecluster > edgecore.logs 2>&1 &
```

```
    tail -f edgecore.logs
```

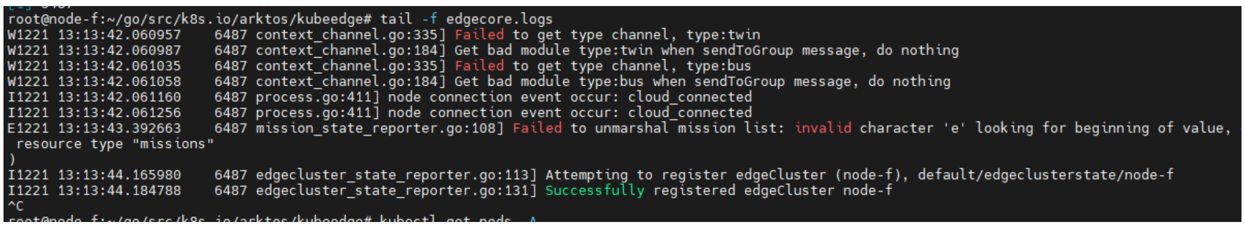


```
root@node-e:~/go/src/k8s.io/arktos/kubeedge# tail -f edgecore.logs
I1221 13:03:54.435156   21439 ws.go:46] dial wss://192.168.1.210:10000/e632aba927ea4ac2b575ec1603d56f10/node-e/events successfully
I1221 13:03:54.435399   21439 websocket.go:93] Websocket connect to cloud access successful
W1221 13:03:54.435477   21439 context_channel.go:335] Failed to get type channel, type:twin
W1221 13:03:54.435511   21439 context_channel.go:184] Get bad module type:twin when sendToGroup message, do nothing
W1221 13:03:54.435570   21439 context_channel.go:335] Failed to get type channel, type:bus
W1221 13:03:54.435602   21439 context_channel.go:184] Get bad module type:bus when sendToGroup message, do nothing
I1221 13:03:54.435606   21439 process.go:411] node connection event occur: cloud_connected
I1221 13:03:54.435705   21439 process.go:411] node connection event occur: cloud_connected
I1221 13:03:56.221906   21439 edgecluster_state_reporter.go:113] Attempting to register edgeCluster (node-e), default/edgeclusterstate/node-e
I1221 13:03:56.239474   21439 edgecluster_state_reporter.go:131] Successfully registered edgeCluster node-e
^C
```

In machine F.

```
 nohup  _output/local/bin/edgecore --edgecluster > edgecore.logs 2>&1 &
```

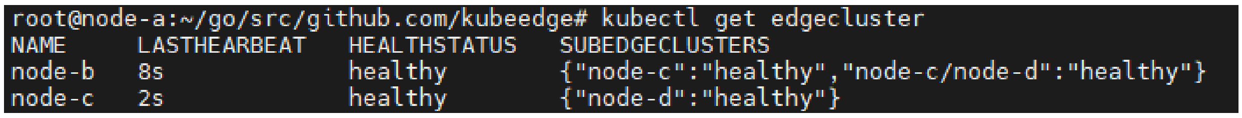
```
    tail -f edgecore.logs
```



```
root@node-f:~/go/src/k8s.io/arktos/kubeedge# tail -f edgecore.logs
W1221 13:13:42.060957    6487 context_channel.go:335] Failed to get type channel, type:twin
W1221 13:13:42.060987    6487 context_channel.go:184] Get bad module type:twin when sendToGroup message, do nothing
W1221 13:13:42.061035    6487 context_channel.go:335] Failed to get type channel, type:bus
W1221 13:13:42.061058    6487 context_channel.go:184] Get bad module type:bus when sendToGroup message, do nothing
I1221 13:13:42.061160    6487 process.go:411] node connection event occur: cloud_connected
I1221 13:13:42.061256    6487 process.go:411] node connection event occur: cloud_connected
E1221 13:13:43.392663    6487 mission_state_reporter.go:108] Failed to unmarshal mission list: invalid character 'e' looking for beginning of value,
 resource type "missions"
)
I1221 13:13:44.165980    6487 edgecluster_state_reporter.go:113] Attempting to register edgeCluster (node-f), default/edgeclusterstate/node-f
I1221 13:13:44.184788    6487 edgecluster_state_reporter.go:131] Successfully registered edgeCluster node-f
^C
```

Now In machine A, Check edgecluster:

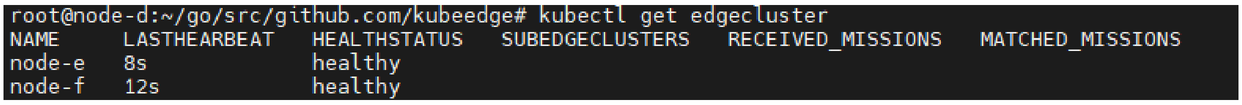
kubectl get edgecluster



```
root@node-a:~/go/src/github.com/kubeedge# kubectl get edgecluster
NAME     LASTHEARBEAT   HEALTHSTATUS   SUBEDGECLUSTERS
node-b   8s             healthy        {"node-c":"healthy","node-c/node-d":"healthy"}
node-c   2s             healthy        {"node-d":"healthy"}
```

Now In machine D, Check edgecluster:

kubectl get edgecluster

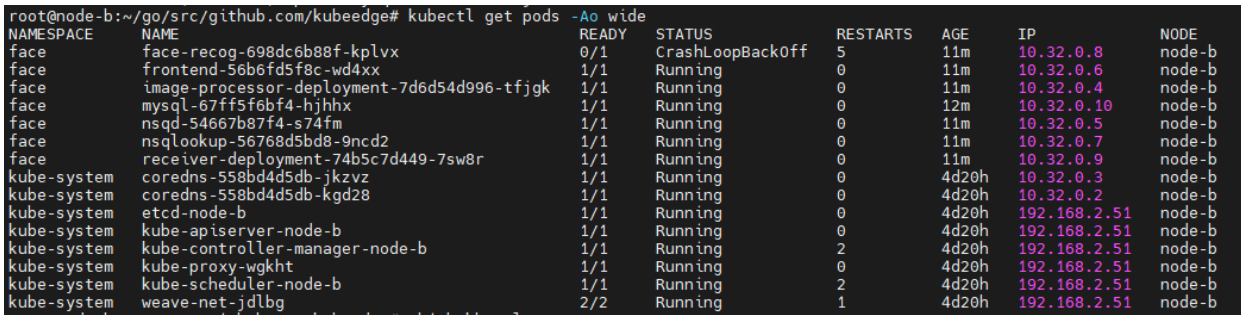


```
root@node-d:~/go/src/github.com/kubeedge# kubectl get edgecluster
NAME     LASTHEARBEAT   HEALTHSTATUS   SUBEDGECLUSTERS   RECEIVED_MISSIONS   MATCHED_MISSIONS
node-e   8s             healthy
node-f   12s            healthy
```

Now In machine B, C, D, E, F, check pods status:

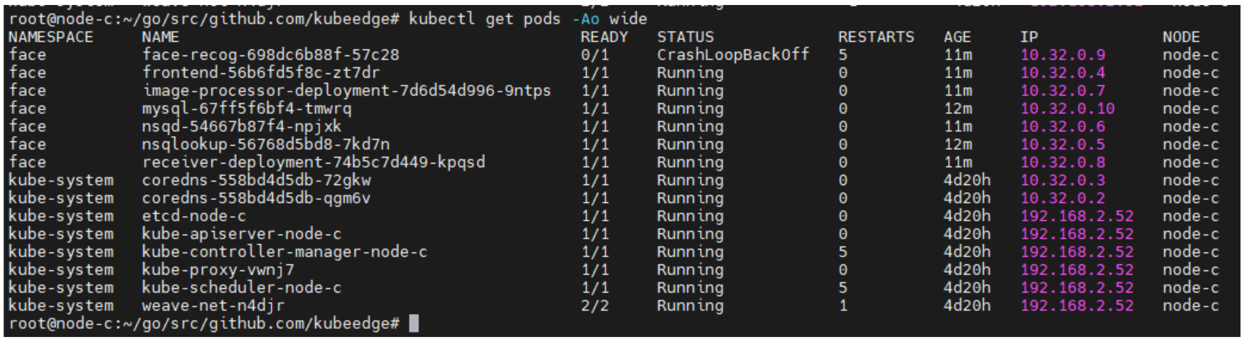
kubectl get pods -Ao wide

Machine B



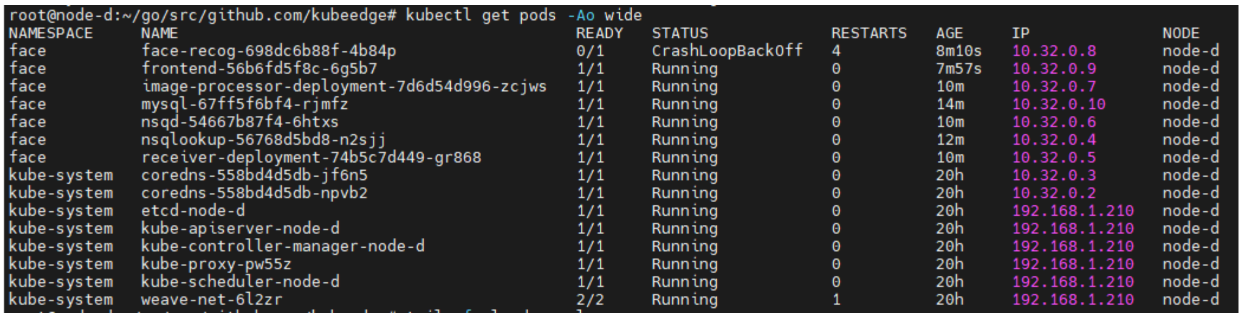
```
root@node-b:~/go/src/github.com/kubeedge# kubectl get pods -Ao wide
NAMESPACE     NAME                                          READY   STATUS             RESTARTS   AGE    IP             NODE
face          face-recog-698dc6b88f-kplvx                   0/1     CrashLoopBackOff   5          11m    10.32.0.8      node-b
face          frontend-56b6fd5f8c-wd4xx                     1/1     Running            0          11m    10.32.0.6      node-b
face          image-processor-deployment-7d6d54d996-tfjgk   1/1     Running            0          11m    10.32.0.4      node-b
face          mysql-67ff5f6bf4-hjhhx                        1/1     Running            0          12m    10.32.0.10     node-b
face          nsqd-54667b87f4-s74fm                         1/1     Running            0          11m    10.32.0.5      node-b
face          nsqlookup-56768d5bd8-9ncd2                    1/1     Running            0          11m    10.32.0.7      node-b
face          receiver-deployment-74b5c7d449-7sw8r          1/1     Running            0          11m    10.32.0.9      node-b
kube-system   coredns-558bd4d5db-jkzvz                      1/1     Running            0          4d20h  10.32.0.3      node-b
kube-system   coredns-558bd4d5db-kgd28                      1/1     Running            0          4d20h  10.32.0.2      node-b
kube-system   etcd-node-b                                   1/1     Running            0          4d20h  192.168.2.51   node-b
kube-system   kube-apiserver-node-b                         1/1     Running            0          4d20h  192.168.2.51   node-b
kube-system   kube-controller-manager-node-b                1/1     Running            2          4d20h  192.168.2.51   node-b
kube-system   kube-proxy-wgkht                              1/1     Running            0          4d20h  192.168.2.51   node-b
kube-system   kube-scheduler-node-b                         1/1     Running            2          4d20h  192.168.2.51   node-b
kube-system   weave-net-jdlbg                               2/2     Running            1          4d20h  192.168.2.51   node-b
```

Machine C



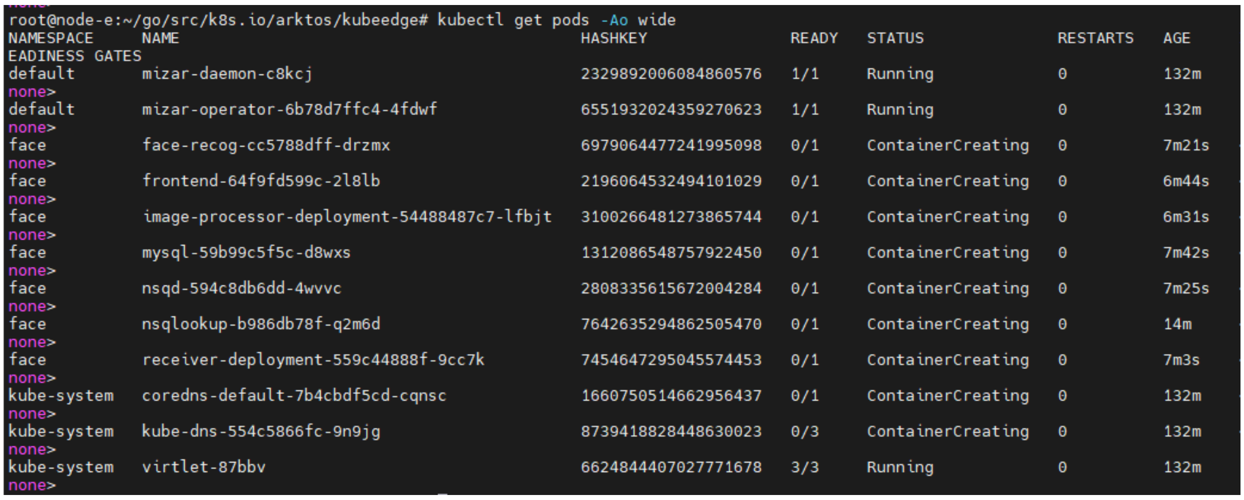
```
root@node-c:~/go/src/github.com/kubeedge# kubectl get pods -Ao wide
NAMESPACE     NAME                                          READY   STATUS             RESTARTS   AGE    IP             NODE
face          face-recog-698dc6b88f-57c28                   0/1     CrashLoopBackOff   5          11m    10.32.0.9      node-c
face          frontend-56b6fd5f8c-zt7dr                     1/1     Running            0          11m    10.32.0.4      node-c
face          image-processor-deployment-7d6d54d996-9ntps   1/1     Running            0          11m    10.32.0.7      node-c
face          mysql-67ff5f6bf4-tmwrq                        1/1     Running            0          12m    10.32.0.10     node-c
face          nsqd-54667b87f4-npjxk                         1/1     Running            0          11m    10.32.0.6      node-c
face          nsqlookup-56768d5bd8-7kd7n                    1/1     Running            0          12m    10.32.0.5      node-c
face          receiver-deployment-74b5c7d449-kpqsd          1/1     Running            0          11m    10.32.0.8      node-c
kube-system   coredns-558bd4d5db-72gkw                      1/1     Running            0          4d20h  10.32.0.3      node-c
kube-system   coredns-558bd4d5db-qgm6v                      1/1     Running            0          4d20h  10.32.0.2      node-c
kube-system   etcd-node-c                                   1/1     Running            0          4d20h  192.168.2.52   node-c
kube-system   kube-apiserver-node-c                         1/1     Running            0          4d20h  192.168.2.52   node-c
kube-system   kube-controller-manager-node-c                1/1     Running            5          4d20h  192.168.2.52   node-c
kube-system   kube-proxy-vwnj7                              1/1     Running            0          4d20h  192.168.2.52   node-c
kube-system   kube-scheduler-node-c                         1/1     Running            5          4d20h  192.168.2.52   node-c
kube-system   weave-net-n4djr                               2/2     Running            1          4d20h  192.168.2.52   node-c
root@node-c:~/go/src/github.com/kubeedge#
```

Machine D



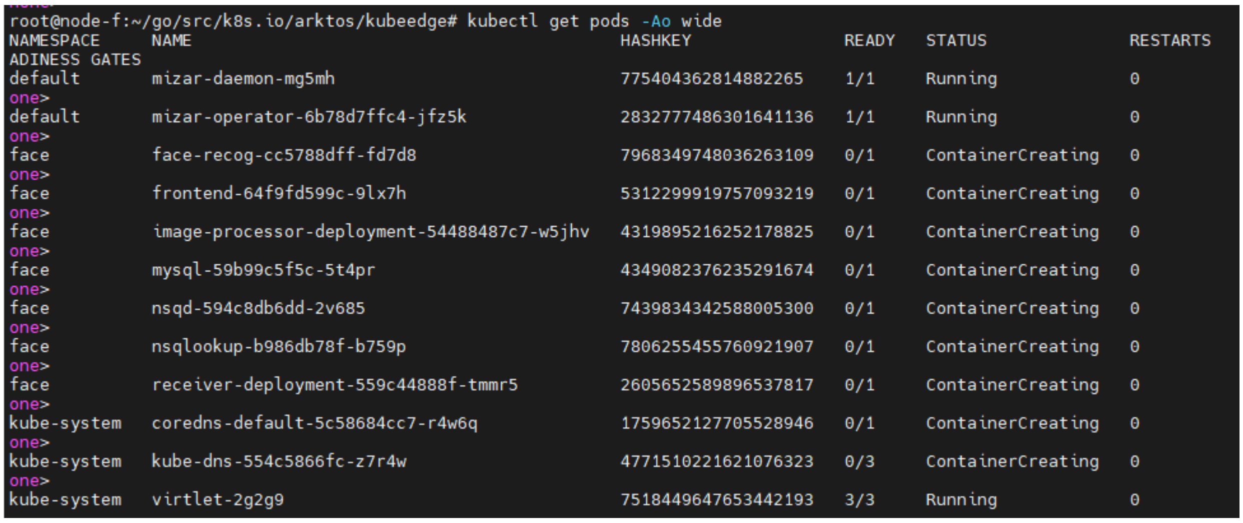
```
root@node-d:~/go/src/github.com/kubeedge# kubectl get pods -Ao wide
NAMESPACE     NAME                                          READY   STATUS             RESTARTS   AGE     IP              NODE
face          face-recog-698dc6b88f-4b84p                   0/1     CrashLoopBackOff   4          8m10s   10.32.0.8       node-d
face          frontend-56b6fd5f8c-6g5b7                     1/1     Running            0          7m57s   10.32.0.9       node-d
face          image-processor-deployment-7d6d54d996-zcjws   1/1     Running            0          10m     10.32.0.7       node-d
face          mysql-67ff5f6bf4-rjmfz                        1/1     Running            0          14m     10.32.0.10      node-d
face          nsqd-54667b87f4-6htxs                         1/1     Running            0          10m     10.32.0.6       node-d
face          nsqlookup-56768d5bd8-n2sjj                    1/1     Running            0          12m     10.32.0.4       node-d
face          receiver-deployment-74b5c7d449-gr868          1/1     Running            0          10m     10.32.0.5       node-d
kube-system   coredns-558bd4d5db-jf6n5                      1/1     Running            0          20h     10.32.0.3       node-d
kube-system   coredns-558bd4d5db-npvb2                      1/1     Running            0          20h     10.32.0.2       node-d
kube-system   etcd-node-d                                   1/1     Running            0          20h     192.168.1.210   node-d
kube-system   kube-apiserver-node-d                         1/1     Running            0          20h     192.168.1.210   node-d
kube-system   kube-controller-manager-node-d                1/1     Running            0          20h     192.168.1.210   node-d
kube-system   kube-proxy-pw55z                              1/1     Running            0          20h     192.168.1.210   node-d
kube-system   kube-scheduler-node-d                         1/1     Running            0          20h     192.168.1.210   node-d
kube-system   weave-net-6l2zr                               2/2     Running            1          20h     192.168.1.210   node-d
```

Machine E



```
root@node-e:~/go/src/k8s.io/arktos/kubeedge# kubectl get pods -Ao wide
NAMESPACE     NAME                                         HASHKEY               READY   STATUS              RESTARTS   AGE
EADINESS GATES
default       mizar-daemon-c8kcj                           2329892006084860576   1/1     Running             0          132m
none>
default       mizar-operator-6b78d7ffc4-4fdwf              6551932024359270623   1/1     Running             0          132m
none>
face          face-recog-cc5788dff-drzmx                   6979064477241995098   0/1     ContainerCreating   0          7m21s
none>
face          frontend-64f9fd599c-2l8lb                    2196064532494101029   0/1     ContainerCreating   0          6m44s
none>
face          image-processor-deployment-54488487c7-lfbjt  3100266481273865744   0/1     ContainerCreating   0          6m31s
none>
face          mysql-59b99c5f5c-d8wxs                       1312086548757922450   0/1     ContainerCreating   0          7m42s
none>
face          nsqd-594c8db6dd-4wvvc                        2808335615672004284   0/1     ContainerCreating   0          7m25s
none>
face          nsqlookup-b986db78f-q2m6d                    7642635294862505470   0/1     ContainerCreating   0          14m
none>
face          receiver-deployment-559c44888f-9cc7k         7454647295045574453   0/1     ContainerCreating   0          7m3s
none>
kube-system   coredns-default-7b4cbdf5cd-cqnsc             1660750514662956437   0/1     ContainerCreating   0          132m
none>
kube-system   kube-dns-554c5866fc-9n9jg                    8739418828448630023   0/3     ContainerCreating   0          132m
none>
kube-system   virtlet-87bbv                                6624844407027771678   3/3     Running             0          132m
none>
```

Machine F

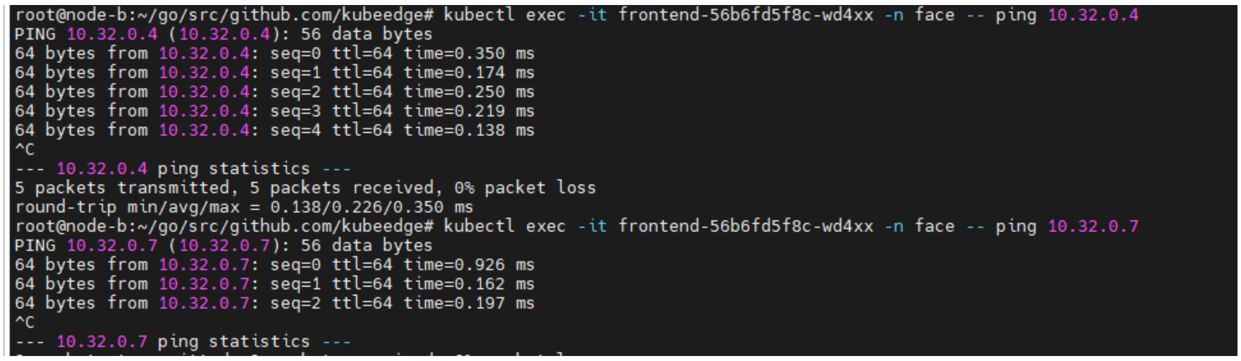


```
root@node-f:~/go/src/k8s.io/arktos/kubeedge# kubectl get pods -Ao wide
NAMESPACE     NAME                                         HASHKEY               READY   STATUS              RESTARTS
ADINESS GATES
default       mizar-daemon-mg5mh                           775404362814882265    1/1     Running             0
one>
default       mizar-operator-6b78d7ffc4-jfz5k              2832777486301641136   1/1     Running             0
one>
face          face-recog-cc5788dff-fd7d8                   7968349748036263109   0/1     ContainerCreating   0
one>
face          frontend-64f9fd599c-9lx7h                    5312299919757093219   0/1     ContainerCreating   0
one>
face          image-processor-deployment-54488487c7-w5jhv  4319895216252178825   0/1     ContainerCreating   0
one>
face          mysql-59b99c5f5c-5t4pr                       4349082376235291674   0/1     ContainerCreating   0
one>
face          nsqd-594c8db6dd-2v685                        7439834342588005300   0/1     ContainerCreating   0
one>
face          nsqlookup-b986db78f-b759p                    7806255455760921907   0/1     ContainerCreating   0
one>
face          receiver-deployment-559c44888f-tmmr5         2605652589896537817   0/1     ContainerCreating   0
one>
kube-system   coredns-default-5c58684cc7-r4w6q             1759652127705528946   0/1     ContainerCreating   0
one>
kube-system   kube-dns-554c5866fc-z7r4w                    4771510221621076323   0/3     ContainerCreating   0
one>
kube-system   virtlet-2g2g9                                7518449647653442193   3/3     Running             0
```

Ping **Machine B pod to Machine C** pod:

```
kubectl exec -it frontend-56b6fd5f8c-wd4xx -n face -- ping 10.32.0.4
```

```
kubectl exec -it frontend-56b6fd5f8c-wd4xx -n face -- ping 10.32.0.7
```



```
root@node-b:~/go/src/github.com/kubeedge# kubectl exec -it frontend-56b6fd5f8c-wd4xx -n face -- ping 10.32.0.4
PING 10.32.0.4 (10.32.0.4): 56 data bytes
64 bytes from 10.32.0.4: seq=0 ttl=64 time=0.350 ms
64 bytes from 10.32.0.4: seq=1 ttl=64 time=0.174 ms
64 bytes from 10.32.0.4: seq=2 ttl=64 time=0.250 ms
64 bytes from 10.32.0.4: seq=3 ttl=64 time=0.219 ms
64 bytes from 10.32.0.4: seq=4 ttl=64 time=0.138 ms
^C
--- 10.32.0.4 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.138/0.226/0.350 ms
root@node-b:~/go/src/github.com/kubeedge# kubectl exec -it frontend-56b6fd5f8c-wd4xx -n face -- ping 10.32.0.7
PING 10.32.0.7 (10.32.0.7): 56 data bytes
64 bytes from 10.32.0.7: seq=0 ttl=64 time=0.926 ms
64 bytes from 10.32.0.7: seq=1 ttl=64 time=0.162 ms
64 bytes from 10.32.0.7: seq=2 ttl=64 time=0.197 ms
^C
--- 10.32.0.7 ping statistics ---
```

Machine E and F (Arktos with Mizar CNI cluster)pods are getting stuck in ContainerCreating state.