Edge Cluster Multi-Layer Setup and Configuration

Date: 17 Dec. 2021

1. Virtual Machine Setup and Configuration (On-Premise)

- Ubuntu 18.04, one for cloud-core, two 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.

2. 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

2.1. 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
```

2.2. Install docker runtime

- Install Docker runtime

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

2.3. 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)
```

2.4. 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
```

3.1. 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
```

- Instal 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, the kubecofig file of cluster B to the machine of cluster C.

- Copy the kubeconfig files of cluster A, B, and C 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. (**See 1.3.3. for detail**) 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 /etc/kubeedge/config -p
mkdir /root/.kube/config -p

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

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, and C.

- For first line mostly use "export KUBECONFIG=/etc/kubernetes/admin.conf".

```
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. (**See 1.3.3. for detail**) 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
```

3. 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. (**See 1.3.3. for detail**) 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_B_kubeconfig_file]
```

Run Fornax Cluster in Machine B, C

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# nohup _output/local/bin/cloudcore > cloudcore.logs 2>&1 &
[1] 19887
root@node-a:~/go/src/github.com/kubeedge# tail -f cloudcore.logs
I1217 10:58:58.830704   19887 core.go:24] Starting module edgecontroller
I1217 10:58:58.830860   19887 upstream.go:123] start upstream controller
I1217 10:58:58.831186   19887 downstream.go:446] start downstream controller
I1217 10:58:58.831617   19887 core.go:24] Starting module devicecontroller
I1217 10:58:58.838268   19887 downstream.go:873] Start downstream devicecontroller
I1217 10:58:58.959060   19887 signcerts.go:100] Succeed to creating token
I1217 10:58:58.959204   19887 server.go:44] start unix domain socket server
I1217 10:58:58.959498   19887 uds.go:71] listening on: //var/lib/kubeedge/kubeedge.sock
I1217 10:58:58.960439   19887 server.go:64] Starting cloudhub websocket server
I1217 10:59:00.840681   19887 upstream.go:63] Start upstream devicecontroller
```

In machine B. (Notes: If we have C machine, we need also run "cloudcore" 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
```

Run cloudcore in machine B (if you have machine C) (Step 3)

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

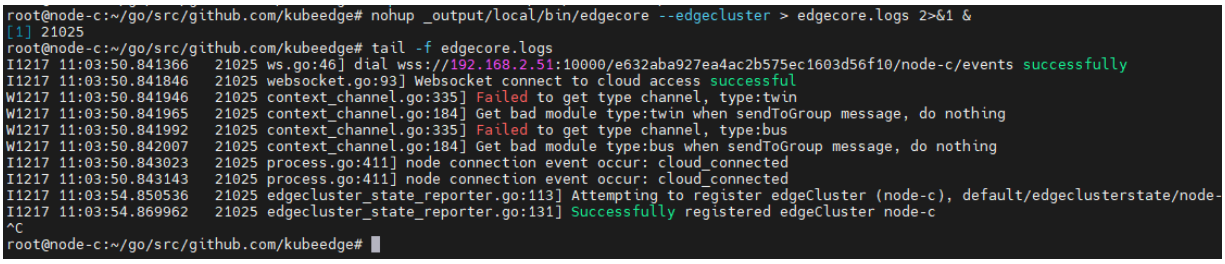
```
root@node-b:~/go/src/github.com/kubeedge# nohup _output/local/bin/edgecore --edgecluster > edgecore.logs 2>&1 &
[3] 27746
root@node-b:~/go/src/github.com/kubeedge# tail -f edgecore.logs
I1217 11:02:39.052517   27746 ws.go:46] dial wss://192.168.2.50:10000/e632aba927ea4ac2b575ec1603d56f10/node-b/events successfully
I1217 11:02:39.052874   27746 websocket.go:93] Websocket connect to cloud access successful
W1217 11:02:39.052972   27746 context_channel.go:335] Failed to get type channel, type:twin
W1217 11:02:39.052990   27746 context_channel.go:184] Get bad module type:twin when sendToGroup message, do nothing
W1217 11:02:39.053012   27746 context_channel.go:335] Failed to get type channel, type:bus
W1217 11:02:39.053024   27746 context_channel.go:184] Get bad module type:bus when sendToGroup message, do nothing
I1217 11:02:39.053172   27746 process.go:411] node connection event occur: cloud_connected
I1217 11:02:39.053278   27746 process.go:411] node connection event occur: cloud_connected
I1217 11:02:39.843323   27746 edgecluster_state_reporter.go:113] Attempting to register edgeCluster (node-b), default/edgeclusterstate/node-b
I1217 11:02:39.855113   27746 edgecluster_state_reporter.go:131] Successfully registered edgeCluster node-b
ls
^C
root@node-b:~/go/src/github.com/kubeedge# ps -ef | grep cloudcore
root     22206  2001  0 11:00 pts/0    00:00:00 _output/local/bin/cloudcore
root     29425  2001  0 11:03 pts/0    00:00:00 grep --color=auto cloudcore
root@node-b:~/go/src/github.com/kubeedge# tail -f cloudcore.logs
I1217 11:00:54.188099   22206 core.go:24] Starting module synccontroller
I1217 11:00:54.188164   22206 core.go:24] Starting module missionstatepruner
I1217 11:00:54.189584   22206 upstream.go:123] start upstream controller
I1217 11:00:54.189929   22206 downstream.go:446] start downstream controller
I1217 11:00:54.190062   22206 downstream.go:873] Start downstream devicecontroller
I1217 11:00:54.320827   22206 signcerts.go:100] Succeed to creating token
I1217 11:00:54.320948   22206 server.go:44] start unix domain socket server
I1217 11:00:54.321224   22206 uds.go:71] listening on: //var/lib/kubeedge/kubeedge.sock
I1217 11:00:54.322046   22206 server.go:64] Starting cloudhub websocket server
I1217 11:00:56.190485   22206 upstream.go:63] Start upstream devicecontroller
^C
```

In machine C. (only run edgecore)

- 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
```



In machine A, do following command in Second Command Window.

```
kubectl apply -f tests/edgecluster/data/missions/deployment-to-all.yaml
```