Edge Cluster Multi-Layer Setup and Configuration

Date: 15 Dec. 2021

Abstract

The purpose of this document is to how to setup and configuration Cloud core and Edge core , and describe the each step to create virtual machine, setup the port number, install kubernets, GoLang, and so on. Running Cloud core and Edge core and deployed mission and task to Edge node. Improve the Edge computing. This Cloud and Edge design is derived from cloud end to edge end for Edge System Functional Description and the Setup Requirements Specification. The intended user of this program is the edge computing user.

1. Virtual Machine Setup (create Cloud core and Edge core virutal machine, and setup port),
2. Fornax Installation and Configuration (Install all the kubernetes compnents: kubectl, kubadm, bubelet),
3. GoLang Installlation and Configuration (Install all GoLang component and load the Fornax source code),
4. Generate Machine Security certification, and deployed to virtual machine,
5. Install CRD file in Cloud core.
6. Run cloud-core and edge-core and deployed mission and verify the mission.

1.1. Virtual Machine Setup and Configuration (We use AWS for example)

- 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
- Go to doc and follow up instruction to setup: Virtual Machine Setup and Configuration Virtual Machine Setup and Configuration
- After done, you can continue to 1.2.

1.2. Install Kubernetes Tools to Cloud core and Edge core

- Install kubernetes tools to virtual machine.(Make sure install version is: 1.21.1-00).
- Kubernetes Tools Doc
- Letting iptables see bridged traffic
- Install docker runtime
- Installing kubeadm, kubelet and kubectl

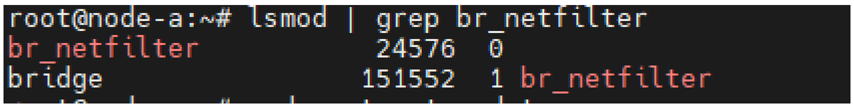
1.2.1. Letting iptables see bridged traffic

• 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
```

1.2.2. Install docker runtime

- Install Docker runtime

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

1.2.3. 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
```

    ii. 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
```

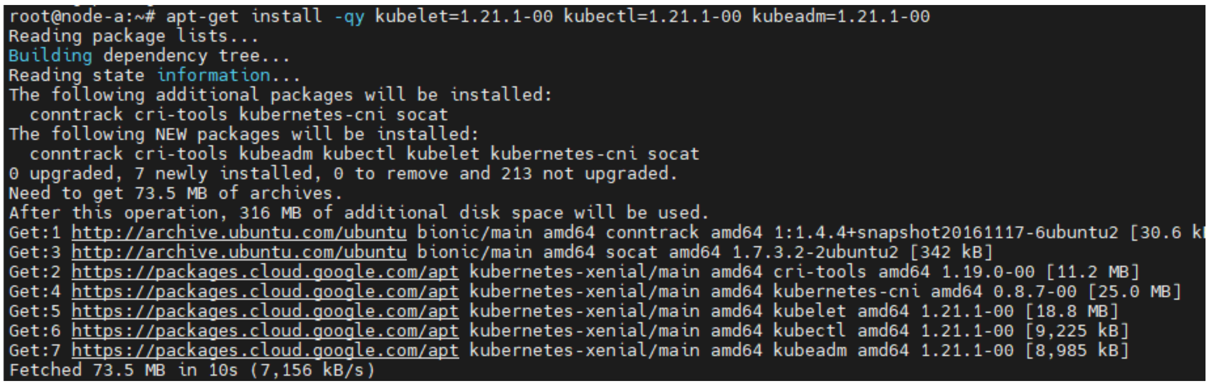
    iii. Add the Kubernetes apt repository:

```
• 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
```

- Next, run the command to enable docker service 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)
```

1.2.4. Start a cluster using kubeadm

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

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

```
kubeadm init
```

-

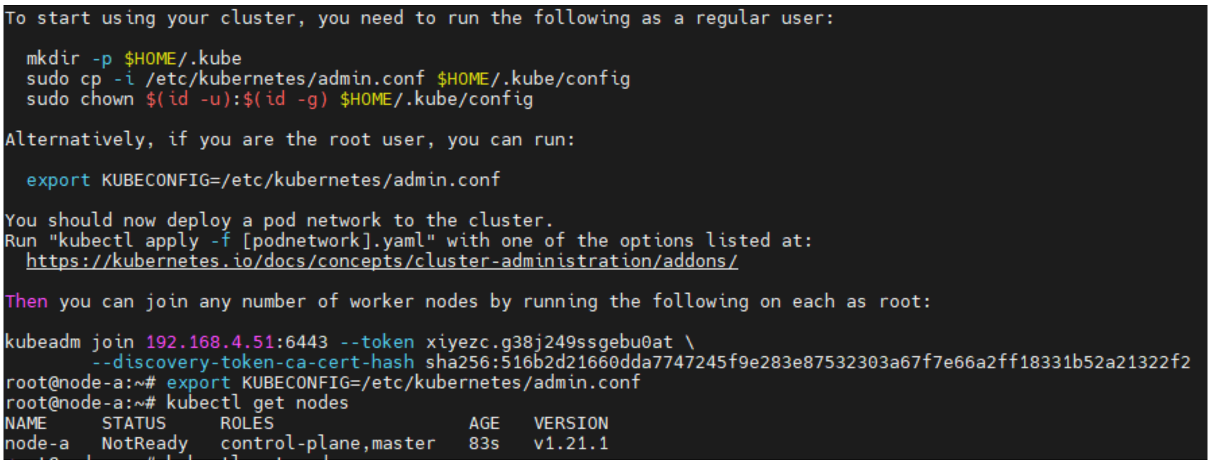
ii. At the end of the screen output, you will see info about setting the kubeconfig. Do the following if you are the root user:

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

-

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

```
kubectl get nodes
```

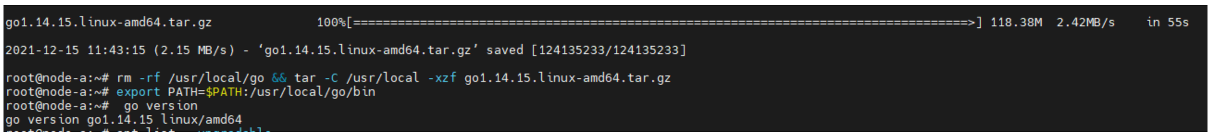


```
To start using your cluster, you need to run the following as a regular user:

  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
```

1.3.1. Install GoLang

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

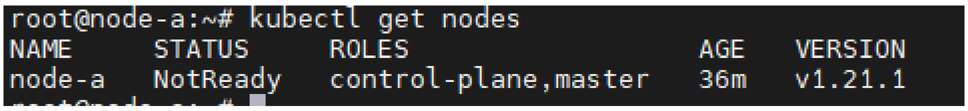
```
GOLANG_VERSION=${GOLANG_VERSION:-"1.14.15"}

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
```



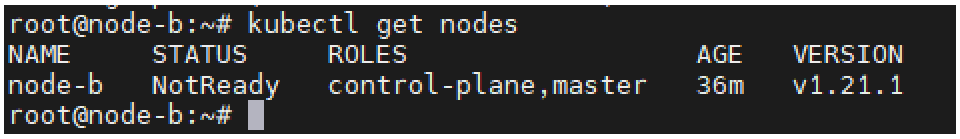
```
go1.14.15.linux-amd64.tar.gz           100%[==========================================================================>] 118.38M  2.42MB/s    in 55s

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

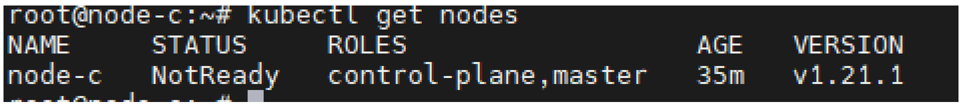
Nodes were not getting ready in any of the machines (A, B, C)



```
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
```