**Multi Master k8-cluster setup  
Kubernetes v1.25.4 installation – Ubuntu**

**\*Disable Firewall**

ufw disable

**\*Disable swap**

swapoff -a; sed -i '/swap/d' /etc/fstab

**\*Update sysctl settings for Kubernetes networking for communicating pods between the cluster**

cat >>/etc/sysctl.d/kubernetes.conf<<EOF

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

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

EOF

sysctl --system

echo 1 >>/proc/sys/net/ipv4/ip\_forward

useradd -m -d /home/engteam engteam

sudo chown engteam:engteam /home/engteam

Add engteam as sudo user

**--------------------------------------changes to be done with engteam user----------------------------------**

sudo modprobe overlay

sudo modprobe br\_netfilter

**K8 Cluster setup follow the below steps:**

**Install docker.**

Create a file with the below command,

**vi docker.sh**

**Add the below contents in this file**,

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install -y docker-ce=5:20.10.17~3-0~ubuntu-focal

sudo service docker start

sudo chmod 666 /var/run/docker.sock

sudo systemctl daemon-reload

sudo systemctl restart docker

to check version

apt-cache policy docker-ce

Now execute the below commands to run the above file and install docker, run this below cmd on both master and worker nodes

chmod +x docker.sh

./docker.sh

Then add daemon.json file in /etc/docker

{

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

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

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}

sudo systemctl enable docker

sudo systemctl daemon-reload

sudo systemctl restart docker  
-----------------------------------------------------------------------------------------------------------------------------  
Run changes with root user

usermod -aG docker engteam

**2. Install Kubernetes**

With the root user

Create a file with the below command, run this below cmd on both master and worker nodes

vi kubernetes.sh

Add the below contents in this file,

sudo apt-get update && sudo apt-get install -y apt-transport-https curl

curl -s [https://packages.cloud.google.com/apt/doc/apt-key.gpg](https://packages.cloud.google.com/apt/doc/apt-key.gpg#Cases/dv/_blank) | sudo apt-key add -

cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list

deb [https://apt.kubernetes.io/](https://apt.kubernetes.io/#Cases/dv/_blank) kubernetes-xenial main

EOF

sudo apt-get update

sudo apt-get install -y kubelet=1.23.2-00 kubeadm=1.23.2-00 kubectl=1.23.2-00

sudo apt-mark hold kubelet kubeadm kubectl

Now execute the below commands to run the above file and install Kubernetes, run this below cmd on both master and worker nodes

chmod +x kubernetes.sh

./kubernetes.sh

**3. Initialize the cluster with the below command, only on master**

With the root user

kubeadm init --control-plane-endpoint="172.23.24.249:8443" --upload-certs --apiserver-advertise-address=172.23.24.145 --pod-network-cidr=10.32.0.0/12

-contorol-plane-endpoint ip is virtual ip for multimatser purpose

Change the private IP accordingly.

When the cluster is initialized Successfully it will print the kubeadm join command. Note that command as you will have to execute it in the worker node to connect it to the master later.

Now, execute the below commands in the master one-by-one,

exit

AFTER INITAlISATION RUN BOTH COMMANDS ON MASTER AND WORKER

Your Kubernetes control-plane 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

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/

You can now join any number of the control-plane node running the following command on each as root:

kubeadm join 172.23.21.230:8443 --token f0a7pu.ebw96s1tyhuvcw48 \

--discovery-token-ca-cert-hash sha256:e84c9bcc53ba2a198509e7646e2b0b5c30f189f29c0724ecfd153b51f98b6d32 \

--control-plane --certificate-key 9e6c826b1e88efe267442733c86474aa94af6001c15875509147c3fe081226e6 --apiserver-advertise-address=172.23.21.63

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!

As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use

"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

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

kubeadm join 172.23.21.230:8443 --token f0a7pu.ebw96s1tyhuvcw48 \

--discovery-token-ca-cert-hash sha256:e84c9bcc53ba2a198509e7646e2b0b5c30f189f29c0724ecfd153b51f98b6d32

**4. Network setup only run on master (POD Network policy)**

kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml

kubectl get nodes

**5. KEEPALIVED install and create required conf file for keepalived on both master nodes**

**With the engteam user**

sudo apt-get install keepalived

After install keepalived package we create keepalived.conf file

sudo vi /etc/keepalived/keepalived.conf

engteam@Prodk8sMaster2:~$ cat /etc/keepalived/keepalived.conf

global\_defs {

router\_id LVS\_DEVEL

}

vrrp\_script check\_apiserver {

script "/etc/keepalived/check\_apiserver.sh"

interval 3

weight -2

fall 10

rise 2

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens160

virtual\_router\_id 51

priority 100

authentication {

auth\_type PASS

auth\_pass 42

}

virtual\_ipaddress {

172.23.21.230

}

track\_script {

check\_apiserver

}

}

sudo systemctl daemon-reload

sudo systemctl restart keepalived

**6. HAPROXY – install HA Proxy on both master nodes**

sudo apt-get install haproxy

sudo cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg\_bkp

update the required changes in haproxy.cfg file

sudo vi /etc/haproxy/haproxy.cfg

engteam@Prodk8sMaster2:~$ cat /etc/haproxy/haproxy.cfg

# /etc/haproxy/haproxy.cfg

#---------------------------------------------------------------------

# Global settings

#---------------------------------------------------------------------

global

log /dev/log local0

log /dev/log local1 notice

daemon

#---------------------------------------------------------------------

# common defaults that all the 'listen' and 'backend' sections will

# use if not designated in their block

#---------------------------------------------------------------------

defaults

mode http

log global

option httplog

option dontlognull

option http-server-close

option forwardfor except 127.0.0.0/8

option redispatch

retries 1

timeout http-request 10s

timeout queue 20s

timeout connect 5s

timeout client 20s

timeout server 20s

timeout http-keep-alive 10s

timeout check 10s

#---------------------------------------------------------------------

# apiserver frontend which proxys to the control plane nodes

#---------------------------------------------------------------------

frontend apiserver

bind \*:8443

mode tcp

option tcplog

default\_backend apiserver

#---------------------------------------------------------------------

# round robin balancing for apiserver

#---------------------------------------------------------------------

backend apiserver

option httpchk GET /healthz

http-check expect status 200

mode tcp

option ssl-hello-chk

balance roundrobin

server prodmaster1 172.23.21.63:6443 check fall 3 rise 2

server prodmaster2 172.23.21.64:6443 check fall 3 rise 2

# [...]

sudo systemctl daemon-reload

sudo systemctl restart haproxy

HA Proxy installation process on master node:

After the execution if error occurs

sed -i "s/cgroupDriver: systemd/cgroupDriver: cgroupfs/g" /var/lib/kubelet/config.yaml

systemctl daemon-reload

systemctl restart kubelet

**Container run time error execute the below command**

rm /etc/containerd/config.toml

systemctl restart containerd

Please update below configuration haproxy.cfg file on both master nodes

----------------------------------------------------------------------------- #---------------------------------------------------------------------

# apiserver frontend which proxys to the control plane nodes

#---------------------------------------------------------------------

frontend apiserver

bind \*:8443

mode tcp

option tcplog

default\_backend apiserver

#---------------------------------------------------------------------

# round robin balancing for apiserver

#---------------------------------------------------------------------

backend apiserver

option httpchk GET /healthz

http-check expect status 200

mode tcp

option ssl-hello-chk

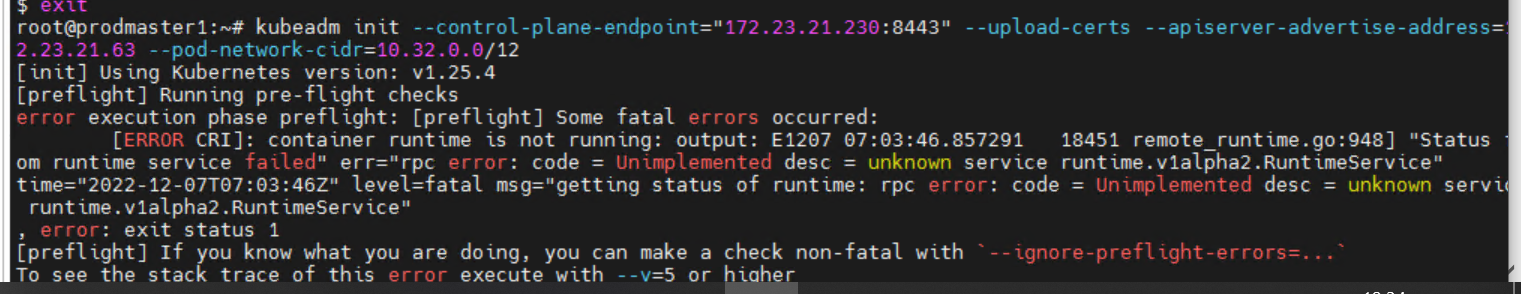
balance roundrobin

server Prodk8sMaster 172.23.21.64:6443 check fall 3 rise 2

server Prodk8sMaster2 172.23.21.63:6443 check fall 3 rise 2

# [...]

-----------------------------------------------------------------------------



Container run time error execute the below command

rm /etc/containerd/config.toml

systemctl restart containerd

Your Kubernetes control-plane 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

Run above 3 commands on both master nodes with the engteam user

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/

You can now join any number of the control-plane node running the following command on each as root:

kubeadm join 172.23.21.230:8443 --token mv8nuu.kr586fq5zv617jso \

--discovery-token-ca-cert-hash sha256:e71c7026ce8d92643f296992c077bbc9b5787f12b224bc5eccd4afabe6a42a47 \

--control-plane --certificate-key 09b18cf27231b6268660f8e86db8c92e66dc6b4a4e6664dcf5857f311f34cc7f

Add --apiserver-advertise-address=172.23.21.64 in the master joining command

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!

As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use

"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

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

kubeadm join 172.23.21.230:8443 --token mv8nuu.kr586fq5zv617jso \

--discovery-token-ca-cert-hash sha256:e71c7026ce8d92643f296992c077bbc9b5787f12b224bc5eccd4afabe6a42a47

To create token for worker node

kubeadm token create --print-join-command

For control plane certificate key generation to add another master

After kubeadm reset perform below Command for total cleanup

The reset process does not clean CNI configuration. To do so, you must remove /etc/cni/net.d

kubeadm init phase upload-certs --upload-certs