HA cluster deployment on ubuntu 22\_04 and configuration

YouTube reference video : <https://www.youtube.com/watch?v=iwlNCePWiw4>

Title : Setting up an on-premise Kubernetes cluster (Ubuntu 22.04)

Note: On master and slave node execute same commands from 1 to 12 points.

1. Deploy ubuntu 22.0.4

Three node, 1 master and 2 worker nodes.

2) Update and upgrade ubuntu OS

sudo apt update && sudo apt upgrade -y

3) Configure host file

Master node

sudo hostnamectl set-hostname " k8s-master-IP36.blr.local"

Worker 1

sudo hostnamectl set-hostname "k8s-worker1-IP37.blr.local"

Worker 2

sudo hostnamectl set-hostname "k8s-worker2-IP38.blr.local"

4) install nao editor and update /etc/hosts file with masternode, workrsmode details:

sudo apt install -y nano

sudo nano /etc/hosts

for example:

# K8S cluster info

10.133.149.36 k8s-master-IP36.blr.local k8s-master-IP36

10.133.149.37 K8s-workernode1-IP37.blr.local K8s-workernode1-IP37

10.133.149.38 K8s-workernode2-IP38.blr.local K8s-workernode2-IP38

5) Must disable swap on all nodes in order for the Kubernetes to work.

sudo swapoff -a

free -h

6) Disable swap on startup

sudo nano /etc/fstab

comment the /swap.img line in file

#/swap.img none swap sw 0 0

sudo mount -a

free -h

7) Load the following kernel modules on all the nodes:

sudo tee /etc/modules-load.d/containerd.conf <<EOF

overlay

br\_netfilter

EOF

sudo modprobe overlay

sudo modprobe br\_netfilter

8) Set the following Kernel parameters for Kubernetes.

sudo 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

9) reload all the changes

sudo sysctl --system

10) Set container runtime

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

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg

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

sudo apt update

sudo apt install -y containerd.io

11) using system d and C GROUP

containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1

sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

sudo systemctl restart containerd

sudo systemctl enable containerd

12) Install components like kubectl, kudeadm

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

sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

sudo apt update

sudo apt install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

Note: On master and slave node execute same command from 1 to 12 points.

=======

Only on master node:

13) Configure kubernetes cluster

Note: This commad should be executed on master node

sudo kubeadm init --pod-network-cidr=192.168.0.0/16

After succesffuly execution of this command.

it given some execution command to execute on master and token number to execute on slave node to join with master node.

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 10.133.149.36:6443 --token izqf78.2klbkf5akpoh0tng \

--discovery-token-ca-cert-hash sha256:3edf63fc9a8976252e57cadb97ede5df1c84401a17415a640bd1d03b58744d92

execute this command in master node:

mkdir -p $HOME/.kube

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

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

14)

On slave node execute this command to join with master node

===========================================================

kubeadm join 10.133.149.36:6443 --token izqf78.2klbkf5akpoh0tng \

--discovery-token-ca-cert-hash sha256:3edf63fc9a8976252e57cadb97ede5df1c84401a17415a640bd1d03b58744d92

Note: Above token and command we got from the output of (when creating kubernetes cluster step no 13)

==========

15) Verify executing following command to verify on master nodes(control plane)

kubectl cluster-info

testingworld@k8s-master-IP36:~$ kubectl cluster-info

Kubernetes control plane is running at https://10.133.149.36:6443

CoreDNS is running at https://10.133.149.36:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

kubectl get nodes

for example:

testingworld@k8s-master-IP36:~$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

k8s-master-ip36.blr.local NotReady control-plane 7m45s v1.28.2

k8s-workernode1-ip37.blr.local NotReady <none> 30s v1.28.2

k8s-workernode2-ip38.blr.local NotReady <none> 5s v1.28.2

Execute this command on only master node.

16) To activate nodes from not-ready to ready state, we need to install container networking pulgins called callico flannel or weavenet.

we are using callico plugins

curl https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml -O

ls

sudo nano calico.yaml

search for CALICO\_IPV4POOL\_CIDR

uncomment the folowing and change the ip we used during cluster creation.

- name: CALICO\_IPV4POOL\_CIDR

value: "192.168.0.0/16"

- name: CALICO\_DISABLE\_FILE\_LOGGING

value: 'true'

Save and close file.

17) Apply callico network interface on master node

kubectl apply -f calico.yaml

18) Verify pod status of namespace kube-system on master node

kubectl get pods -n kube-system

testingworld@k8s-master-IP36:~$ kubectl get pods -n kube-system

NAME READY STATUS RESTARTS AGE

calico-kube-controllers-658d97c59c-d2m65 1/1 Running 0 2m37s

calico-node-bwqr9 1/1 Running 0 2m37s

calico-node-jx6n7 1/1 Running 0 2m37s

calico-node-l5tkl 1/1 Running 0 2m37s

coredns-5dd5756b68-26h4t 1/1 Running 0 29m

coredns-5dd5756b68-bb7gr 1/1 Running 0 29m

etcd-k8s-master-ip36.blr.local 1/1 Running 0 29m

kube-apiserver-k8s-master-ip36.blr.local 1/1 Running 0 29m

kube-controller-manager-k8s-master-ip36.blr.local 1/1 Running 0 29m

kube-proxy-dnnft 1/1 Running 0 29m

kube-proxy-mhxcs 1/1 Running 0 22m

kube-proxy-mlthg 1/1 Running 0 22m

kube-scheduler-k8s-master-ip36.blr.local 1/1 Running 0 29m

19) check nodes status in master node

kubectl get nodes

testingworld@k8s-master-IP36:~$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

k8s-master-ip36.blr.local Ready control-plane 31m v1.28.2

k8s-workernode1-ip37.blr.local Ready <none> 23m v1.28.2

k8s-workernode2-ip38.blr.local Ready <none> 23m v1.28.2

20) What if in case we lost the token, we can generate the new token by executing below command on master node.

kubeadm token create --print-join-command

testingworld@k8s-master-IP36:~$ kubeadm token create --print-join-command

kubeadm join 10.133.149.36:6443 --token 128b7h.7teia7u1xsg3td8d --discovery-token-ca-cert-hash sha256:3edf63fc9a8976252e57cadb97ede5df1c84401a17415a640bd1d03b58744d92

testingworld@k8s-master-IP36:~$

21) to list the token generated on master node, execute below command on master node

kubeadm token list

testingworld@k8s-master-IP36:~$ kubeadm token list

TOKEN TTL EXPIRES USAGES DESCRIPTION EXTRA GROUPS

128b7h.7teia7u1xsg3td8d 23h 2024-02-19T15:01:25Z authentication,signing <none> system:bootstrappers:kubeadm:default-node-token

izqf78.2klbkf5akpoh0tng 23h 2024-02-19T14:16:12Z authentication,signing The default bootstrap token generated by 'kubeadm init'. system:bootstrappers:kubeadm:default-node-token

testingworld@k8s-master-IP36:~$