# **Lab x: Multi Master K8s Cluster with kubeadm**

## **Pre-requisite**

* 7 servers are needed
* They need to be in the same network
* Network need to have access to the internet

1. The first machine needed is the machine on which the HAProxy load balancer will be installed. We will assign the IP 10.10.10.93 to this machine
2. We also need three Kubernetes master nodes. These machines will have the IPs 10.10.10.90, 10.10.10.91, and 10.10.10.92
3. Finally, we will also have three Kubernetes worker nodes with the IPs 10.10.10.100, 10.10.10.101, and 10.10.10.102

## **Installing kubectl**

1. Download the binary

$wget <https://storage.googleapis.com/kubernetes-release/release/v1.15.0/bin/linux/amd64/kubectl>

1. Add the execution permission to the binary

$chmod +x kubectl

1. Move the binary to /usr/local/bin

$sudo mv kubectl /usr/local/bin

1. Verify the installation

$ kubectl version

## **Installing the HAProxy load balancer**

1. As we will deploy three Kubernetes master nodes, we need to deploy an HAPRoxy load balancer in front of them to distribute the traffic.
2. SSH to the 10.10.10.93 Ubuntu machine.Update the machine.

$ sudo apt-get update

$ sudo apt-get upgrade

1. Install HAProxy and Configure HAProxy to load balance the traffic between the three Kubernetes master nodes.

$ sudo apt-get install haproxy

$ sudo vim /etc/haproxy/haproxy.cfg

global

...

default

...

frontend kubernetes

bind 10.10.10.93:6443

option tcplog

mode tcp

default\_backend kubernetes-master-nodes

backend kubernetes-master-nodes

mode tcp

balance roundrobin

option tcp-check

server k8s-master-0 10.10.10.90:6443 check fall 3 rise 2

server k8s-master-1 10.10.10.91:6443 check fall 3 rise 2

server k8s-master-2 10.10.10.92:6443 check fall 3 rise 2

1. Restart HAProxy

$ sudo systemctl restart haproxy

## **Preparing the nodes for kubeadm**

1. Preparing the 10.10.10.90/91/92/100/101/102 machine. Performing below steps on all systems
2. Installing Docker latest version

$ sudo -s

# curl -fsSL https://get.docker.com -o get-docker.sh

# sh get-docker.sh

# usermod -aG docker your-user

1. Installing kubeadm, kublet, and kubectl. Add the Google repository key.

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

1. Add the Google repository

# vim /etc/apt/sources.list.d/kubernetes.list

deb http://apt.kubernetes.io kubernetes-xenial main

1. Update the list of packages and install kubelet, kubeadm and kubectl

# apt-get update

# apt-get install kubelet kubeadm kubectl

1. Disable the swap

# swapoff -a

# sed -i '/ swap / s/^/#/' /etc/fstab

## **Initializing the master nodes**

1. Initializing the Master node 10.10.10.90
2. SSH to the 10.10.10.90 machine and Create the configuration file for kubeadm

$ vim config.yaml

apiVersion: kubeadm.k8s.io/v1alpha3

kind: ClusterConfiguration

kubernetesVersion: stable

apiServerCertSANs:

- 10.10.10.93

controlPlaneEndpoint: "10.10.10.93:6443"

etcd:

external:

endpoints:

- https://10.10.10.90:2379

- https://10.10.10.91:2379

- https://10.10.10.92:2379

caFile: /etc/etcd/ca.pem

certFile: /etc/etcd/kubernetes.pem

keyFile: /etc/etcd/kubernetes-key.pem

networking:

podSubnet: 10.30.0.0/24

apiServerExtraArgs:

apiserver-count: "3"

1. Initialize the machine as a master node

$ sudo kubeadm init --config=config.yaml

1. Copy the certificates to the two other masters

$ sudo scp -r /etc/kubernetes/pki ubuntu@10.10.10.91:~

$ sudo scp -r /etc/kubernetes/pki [ubuntu@10.10.10.92:~](mailto:ubuntu@10.10.10.92:~)

1. Initializing the 2nd master node 10.10.10.91. SSH to the 10.10.10.91 machine.Remove the apiserver.crt and apiserver.key.

$ rm ~/pki/apiserver.\*

1. 3- Move the certificates to the /etc/kubernetes directory

$ sudo mv ~/pki /etc/kubernetes/

1. Create the configuration file for kubeadm

$ vim config.yaml

apiVersion: kubeadm.k8s.io/v1alpha3

kind: ClusterConfiguration

kubernetesVersion: stable

apiServerCertSANs:

- 10.10.10.93

controlPlaneEndpoint: "10.10.10.93:6443"

etcd:

external:

endpoints:

- https://10.10.10.90:2379

- https://10.10.10.91:2379

- https://10.10.10.92:2379

caFile: /etc/etcd/ca.pem

certFile: /etc/etcd/kubernetes.pem

keyFile: /etc/etcd/kubernetes-key.pem

networking:

podSubnet: 10.30.0.0/24

apiServerExtraArgs:

apiserver-count: "3"

1. Initialize the machine as a master node

$ sudo kubeadm init --config=config.yaml

1. Initializing the 3rd master node 10.10.10.92 1- SSH to the 10.10.10.92 machine.Remove the apiserver.crt and apiserver.key

$ rm ~/pki/apiserver.\*

1. Move the certificates to the /etc/kubernetes directory

$ sudo mv ~/pki /etc/kubernetes/

1. Create the configuration file for kubeadm

$ vim config.yaml

apiVersion: kubeadm.k8s.io/v1alpha3

kind: ClusterConfiguration

kubernetesVersion: stable

apiServerCertSANs:

- 10.10.10.93

controlPlaneEndpoint: "10.10.10.93:6443"

etcd:

external:

endpoints:

- https://10.10.10.90:2379

- https://10.10.10.91:2379

- https://10.10.10.92:2379

caFile: /etc/etcd/ca.pem

certFile: /etc/etcd/kubernetes.pem

keyFile: /etc/etcd/kubernetes-key.pem

networking:

podSubnet: 10.30.0.0/24

apiServerExtraArgs:

apiserver-count: "3"

1. Initialize the machine as a master node

$ sudo kubeadm init --config=config.yaml

## **Joining the worker nodes to the cluster**

1. Execute the “kubeadm join” command that you copied from the last step of the initialization of the masters.

$ sudo kubeadm join 10.10.40.93:6443 --token [your\_token] --discovery-token-ca-cert-hash sha256:[your\_token\_ca\_cert\_hash]

1. Run same command on worker node 101 and 102

## **Verifying the cluster status**

$ sudo kubectl --kubeconfig /etc/kubernetes/admin.conf get nodes

## **Configuring kubectl on the client machine**

1. SSH to one of the master node. 10.10.10.90 and sdd permissions to the admin.conf file

$ sudo chmod +r /etc/kubernetes/admin.conf

1. From the client machine, copy the configuration file

$ scp ubuntu@10.10.10.90:/etc/kubernetes/admin.conf .

1. Create and configure the kubectl configuration directory

$ mkdir ~/.kube

$ mv admin.conf ~/.kube/config

$ chmod 600 ~/.kube/config

1. Go back to the SSH session on the master and change back the permissions of the configuration file

$ sudo chmod 600 /etc/kubernetes/admin.conf

1. check that you can access the Kubernetes API from the client machine

$ kubectl get nodes

## **Deploying the overlay network**

1. We are going to use Calico as the overlay network. You can also use static route or another overlay network tool like Weavenet or Flannel. Deploy the overlay network pods from the client machine.

$kubectl apply -f <https://docs.projectcalico.org/v3.7/manifests/calico.yaml>

1. Check that the nodes are in Ready state and also check that the pods are deployed properly

$ kubectl get nodes

$ kubectl get pods -n kube-system