**Setup Kubernetes Cluster using kubeadm, calico and rbac in Ubuntu**

**Prerequisite Setup**

**#Setup**

# 1. 3 VMs Ubuntu 16.04.5 or 18.04.1.0, 1 master, 2 nodes.

# 2. Static IPs on individual VMs

# 3. /etc/hosts hosts file includes name to IP mappings for VMs

# 4. Swap is disabled

# 5. Take snapshots prior to installations, this way you can install

# and revert to snapshot if needed

#Disable swap, swapoff then edit your fstab removing any entry for swap partitions

#You can recover the space with fdisk. You may want to reboot to ensure your config is ok.

swapoff -a

vi /etc/fstab

#Add Google's apt repository gpg key

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

#Add the Kubernetes apt repository

sudo bash -c 'cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

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

EOF'

#Update the package list and use apt-cache to inspect versions available in the repository

sudo apt-get update

apt-cache policy kubelet | head -n 20

apt-cache policy docker.io | head -n 20

#Install the required packages, if needed we can request a specific version

sudo apt-get install -y docker.io kubelet kubeadm kubectl

sudo apt-mark hold docker.io kubelet kubeadm kubectl

#Check the status of our kubelet and our container runtime, docker.

#The kubelet will enter a crashloop until it's joined.

sudo systemctl status kubelet.service

sudo systemctl status docker.service

#Ensure both are set to start when the system starts up.

sudo systemctl enable kubelet.service

sudo systemctl enable docker.service

**Master Node Setup aka Kubernetes Control Panel Setup**

#Only on the master, download the yaml files for the pod network

wget https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/rbac-kdd.yaml

wget https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/kubernetes-datastore/calico-networking/1.7/calico.yaml

#Look inside calico.yaml and find the network range, adjust if needed.

vi calico.yaml

#Create our kubernetes cluster, specifying a pod network range matching that in calico.yaml!

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

#Configure our account on the master to have admin access to the API server from a non-privileged account.

mkdir -p $HOME/.kube

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

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

#Download yaml files for your pod network

kubectl apply -f rbac-kdd.yaml

kubectl apply -f calico.yaml

#Look for the all the system pods and calico pod to change to Running.

#The DNS pod won't start until the Pod network is deployed and Running.

kubectl get pods --all-namespaces

#Gives you output over time, rather than repainting the screen on each iteration.

kubectl get pods --all-namespaces --watch

#All system pods should be Running

kubectl get pods --all-namespaces

#Get a list of our current nodes, just the master.

kubectl get nodes

#Check out the systemd unit, and examine 10-kubeadm.conf

#Remeber the kubelet starts static pod manifests, and thus the core cluster pods

sudo systemctl status kubelet.service

#check out the directory where the kubeconfig files live

ls /etc/kubernetes

#let's check out the manifests on the master

ls /etc/kubernetes/manifests

#And look more closely at API server and etcd's manifest.

sudo more /etc/kubernetes/manifests/etcd.yaml

sudo more /etc/kubernetes/manifests/kube-apiserver.yaml

**Worker Node Setup**

#For this demo ssh into c1-node1

ssh aen@c1-node1

#Disable swap, swapoff then edit your fstab removing any entry for swap partitions

#You can recover the space with fdisk. You may want to reboot to ensure your config is ok.

swapoff -a

vi /etc/fstab

#Add the Google's apt repository gpg key

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

#Add the kuberentes apt repository

sudo bash -c 'cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

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

EOF'

#Update the package list

sudo apt-get update

apt-cache policy kubelet | head -n 20

#Install the required packages, if needed we can request a specific version

sudo apt-get install -y docker.io kubelet kubeadm kubectl

sudo apt-mark hold docker.io kubelet kubeadm kubectl

#Check the status of our kubelet and our container runtime, docker.

#The kubelet will enter a crashloop until it's joined

sudo systemctl status kubelet.service

sudo systemctl status docker.service

#Ensure both are set to start when the system starts up.

sudo systemctl enable kubelet.service

sudo systemctl enable docker.service

#If you didn't keep the output, on the master, you can get the token.

kubeadm token list

#If you need to generate a new token, perhaps the old one timed out/expired.

kubeadm token create

#On the master, you can find the ca cert hash.

openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | openssl dgst -sha256 -hex | sed 's/^.\* //'

#Using the master (API Server) IP address or name, the token and the cert has, let's join this Node to our cluster.

sudo kubeadm join 172.16.94.10:6443 \

--token 9woi9e.gmuuxnbzd8anltdg \

--discovery-token-ca-cert-hash sha256:f9cb1e56fecaf9989b5e882f54bb4a27d56e1e92ef9d56ef19a6634b507d76a9

#Back on master, this will say NotReady until the networking pod is created on the new node. Has to schedule the pod, then pull the container.

kubectl get nodes

#On the master, watch for the calico pod and the kube-proxy to change to Running on the newly added nodes.

kubectl get pods --all-namespaces --watch

#Still on the master, look for this added node's status as ready.

kubectl get nodes

#GO BACK TO THE TOP AND DO THE SAME FOR c1-node2.

#Just SSH into c1-node2 and run the commands again.

ssh aen@c1-node2

#You can skip the token re-creation if you have one that's still valid.

88\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

apt-get update

2 apt-get unzip

3 apt-get install unzip

4 sudo ./aws/istall

5 sudo ./aws/install

6 curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

7 curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

8 unzip awscliv2.zip

9 sudo ./aws/install

10 aws

11 aws configure

12 aws s3 ls

13 curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

14 curl -LO "https://dl.k8s.io/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"

15 echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

16 sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

17 kubectl version --client

18 kubectl get pods

19 aws eks update-kubeconfig --name Demo\_Cluster --region ap-south-1

20 cat ~./kube/config

21 cat ~/.kube/config

22 kubectl get pods

23 aws configure

24 kubectl get pods

25 kubectl get pods -a

26 kubectl get pods -A

27 history