# Kubernetes

# AWS EC2 Instance

# Open an SSH client.

# Locate your private key file (key.pem)

# # chmod 400 key.pem

# Now you will be able to SSH using your Public DNS/Public IP

# ssh -i "key.pem" [ec2-user@ec2-xx-xx-xx-xx.ap-south-1.compute.amazonaws.com](mailto:ec2-user@ec2-xx-xx-xx-xx.ap-south-1.compute.amazonaws.com)

# NB:- If SSH connectivity does not work, see changing the chmod to 600 (chmod 600 singaporekeypair.pem)

# [ec2-user@ip-xx-xx-xx-xx ~]$ sudo su

# # yum install -y git

# Steps to Check out from git repository

# # mkdir gitrepo

# # cd gitrepo

# # git init

# # git config --global user.email "nevin.cleetus@email.com"

# # git config --global user.name "nevin-cleetus"

# # git clone https://github.com/nevin-cleetus/k8.git

# Ensure

# 1. Internet is working

# 2. Nobody should be connected to vpn or any other proxy.

# 3. Disable firewall if enabled.

# 

# 

# 

# Preparing Servers

# login as: vagrant

# sudo su

# Disable the Selinux

# setenforce 0

sed -i 's/SELINUX=permissive/SELINUX=disabled/' /etc/sysconfig/selinux

systemctl disable firewalld

systemctl stop firewalld

swapoff -a

sed -i 's/^.\*swap/#&/' /etc/fstab

iptables -P FORWARD ACCEPT

vi /etc/sysctl.d/k8s.conf

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

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

net.ipv4.ip\_forward = 1

vm.swappiness=0

sysctl --system

Master /etc/hosts

# wget https://download.docker.com/linux/static/stable/x86\_64/docker-19.03.0.tgz

# tar -zxvf docker-19.03.0.tgz

cd docker

cp \* /usr/local/bin

# **wget** https://download.docker.com/linux/static/stable/x86\_64/docker-19.03.0.tgz

# # tar -zxvf docker-19.03.0.tgz

# cd docker

# cp \* /usr/local/bin

# Change the content of /etc/systemd/system/docker.service to the following

[Unit]

Description=Docker Application Container Engine

Documentation=http://docs.docker.io

[Service]

ExecStart=/usr/local/bin/dockerd

ExecStartPost=/sbin/iptables -I FORWARD -s 0.0.0.0/0 -j ACCEPT

ExecReload=/bin/kill -s HUP $MAINPID

Restart=on-failure

RestartSec=5

LimitNOFILE=infinity

LimitNPROC=infinity

LimitCORE=infinity

Delegate=yes

KillMode=process

[Install]

WantedBy=multi-user.target

# systemctl daemon-reload

# systemctl enable docker

# systemctl restart docker

vi /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

yum makecache fast

yum install -y -q kubeadm kubelet kubectl

vi /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf

Environment="KUBELET\_CGROUP\_ARGS=--cgroup-driver=cgroupfs"

systemctl daemon-reload

systemctl enable kubelet

Download Cloudflare's PKI and TLS toolkit

# curl -o /usr/local/bin/cfssl https://pkg.cfssl.org/R1.2/cfssl\_linux-amd64

# curl -o /usr/local/bin/cfssljson https://pkg.cfssl.org/R1.2/cfssljson\_linux-amd64

# chmod +x /usr/local/bin/cfssl\*

Create a certificates folder

kubeadm-config.yaml

apiVersion: kubeadm.k8s.io/v1alpha1

kind: MasterConfiguration

api:

advertiseAddress: 172.42.42.151

etcd:

endpoints:

- https:// 172.42.42.151:2379

- https:// 172.42.42.152:2379

caFile: /etc/etcd/ssl/ca.pem

certFile: /etc/etcd/ssl/etcd.pem

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

networking:

podSubnet: 192.168.0.0/16

apiServerCertSANs:

- 172.42.42.151

- 172.42.42.152

- 172.42.42.160

apiServerExtraArgs:

endpoint-reconciler-type: lease

Copy pki files to all other master nodes

# scp -rp /etc/kubernetes/pki 172.42.42.151:/etc/kubernetes/

# scp -rp /etc/kubernetes/pki 172.42.42.152:/etc/kubernetes/

# [ec2-user@ip-xx-xx-xx-xx ~]$

# $ sudo yum update -y

# $ sudo yum install -y docker

# Expected output

# Installed:

# docker.x86…

# Dependency Installed:

# Containerd libcgroup.x86\_64

# runc.x86\_64 pigz.x86\_64

# Complete!

# Start and Enable the docker service

# $ sudo service docker start

# $ sudo service docker status

# Make sure the service is Active: active (running)

# Only If docker group is not available. To verify run cat /etc/group | grep -i docker

# $ sudo groupadd docker

# $ sudo usermod -a -G docker ec2-user

# $ sudo systemctl enable docker

# Run the following command to activate the changes to groups:

# $ newgrp docker

# $ docker run hello-world

# 

# Expected output: Hello world message from the application.

# Kubernetes Installation

# Below steps need to be executed in both Kubernetes master and worker nodes

$ login as: ec2-user

$ sudo su

NB:- Please make sure the “gpgkey” is not split into two lines (Copy paste the below entries in a notepad and confirm before saving it). If should come in the same line. Also please make sure no special characters get copied while pasting it.

Kubernetes packages are not available in the default CentOS 7 & RHEL 7 repositories, Use below command to configure its Kubernetes package repositories.

$ cat <<EOF > /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=1

repo\_gpgcheck=0

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

EOF

Ensure net.bridge.bridge-nf-call-iptables is set to 1 in your sysctl config to make sure the packets traversing the bridge are sent to iptables for processing

$ cat <<EOF > /etc/sysctl.d/k8s.conf

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

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

EOF

$ sysctl --system

Expected Output

\* Applying /etc/sysctl.d/00-defaults.conf ...

\* Applying /etc/sysctl.d/k8s.conf ...

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

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

\* Applying /etc/sysctl.conf ...

$ setenforce 0

setenforce: SELinux is disabled

$ yum install -y kubelet kubeadm kubectl

Installed:

kubeadm.x86\_64 0:1.18.0-0 kubectl.x86\_64 0:1.18.0-0 kubelet.x86\_64 0:1.18.0-0

$ systemctl enable kubelet && systemctl start kubelet

Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /usr/lib/systemd/system/kubelet.service.

# Kubernetes Master Node

# Initialize the Kubernetes master using the below command. This indicate the PODS are going get the IP addresses in the range out of the CIDR 192.168.0.0/16.

$ sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --ignore-preflight-errors=NumCPU

Your Kubernetes control-plane has initialized successfully!

……

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

**kubeadm join 172.31.41.123:6443 --token 7oy7kg.g2u9wmhewgxyl0zn \**

**--discovery-token-ca-cert-hash sha256:9e94b12d9391d5afa78bf32beece20ae129f8cff8d3a81b2085ff718a9274879**

[root@ip-172-31-38-38 ~]$ exit

[ec2-user@ip-172-31-38-38 ~]$ mkdir -p $HOME/.kube

[ec2-user @ip-172-31-38-38 ~]$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[ec2-user @ip-172-31-38-38 ~]$ sudo chown $(id -u):$(id -g) $HOME/.kube/config

# Kubernetes Worker Node

# [Don’t Copy paste and execute the below command on the worker node. Instead use the join statement which you received while executing init from the Master node above]

# kubeadm join xxx. xxx. xxx. xxx:6443 --token 7oy7kg.g2u9wmhewgxyl0zn \

--discovery-token-ca-cert-hash sha256:9e94b12d9391d5afa78bf32beece20ae129f8cff8d3a81b2085ff718a9274879

# Kubernetes Master Node

kubeadm token create --print-join-command

This node has joined the cluster:

\* Certificate signing request was sent to apiserver and a response was received.

\* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

# Kubernetes Master Node

[ec2-user@...]# kubectl cluster-info

Kubernetes master is running at https://172.31.41.123:6443

KubeDNS is running at <https://172.31.41.123:6443/api/v1/namespaces/kube-system/services/kube> -dns:dns/proxy

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

[ec2-user@...]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-X-X-X-X.ap-south-1.compute.internal NotReady <none> 2m51s v1.18.0

ip-X-X-X-X.ap-south-1.compute.internal NotReady master 7m19s v1.18.0

[ec2-user @...]# kubectl get nodes

[ec2-user @ ip- kubernetes]# cd gitrepo/k8/cni/

**Install Calico CNI plugin [Only on the Master node]**

[ec2-user @ip- kubernetes] kubectl apply -f etcd.yaml

[ec2-user @ip- kubernetes] kubectl apply -f rbac.yaml

[ec2-user @ip- kubernetes] kubectl apply -f calico.yaml

Wait for a few minutes and the node status should be changed to ‘Ready’

[root@ip- kubernetes] kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip- xxxxxxx Ready master 15m v1.18.0

ip-xxxxxxx Ready <none> 10m v1.18.0

If you want to run Kubectl from worker nodes

[root @ip- kubernetes] mkdir -p $HOME/.kube

[root @ip- kubernetes] export KUBECONFIG=/etc/kubernetes/kubelet.conf

[root @ip- kubernetes] kubectl get nodes

**Un Install Kubernetes**

sudo docker rm `docker ps -a -q`

sudo docker rmi `docker images -q`

sudo kubeadm reset

sudo yum remove kubeadm kubectl kubelet kubernetes-cni kube\*

sudo yum autoremove

sudo rm -rf ~/.kube

kubectl delete node --all

$ for service in kube-apiserver kube-controller-manager kubectl kubelet kube-proxy kube-scheduler; do

systemctl stop $service

done

$ yum -y remove kubernetes #if it's registered as a service