#Launch virtual machines with 2gb ram, 2 cpu minimum for master node and 1gb ram, 2 cpu for slave nodes in virtual box. I am using Ubuntu 20.04.5 LTS

 # Install docker on both master and slave vm:

sudo apt-get update

sudo apt-get install ca-certificates curl gnupg lsb-release

sudo mkdir -m 0755 -p /etc/apt/keyrings

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

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

#You can confirm docker installed with “docker version”

 # Install cri-dockerd on both master and slave vm. This adapter provides a shim for Docker Engine that lets you control Docker via the Kubernetes Container Runtime Interface.

git clone https://github.com/Mirantis/cri-dockerd.git

###Install GO###

wget https://storage.googleapis.com/golang/getgo/installer_linux

sudo chmod +x ./installer_linux

./installer_linux

source ~/.bash_profile

You can confirm go installed with “go version” command

cd cri-dockerd

mkdir bin

go build -o bin/cri-dockerd #This build will take some time

mkdir -p /usr/local/bin

sudo install -o root -g root -m 0755 bin/cri-dockerd /usr/local/bin/cri-dockerd

sudo cp -a packaging/systemd/* /etc/systemd/system

sudo sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.service

sudo systemctl daemon-reload

sudo systemctl enable cri-docker.service

sudo systemctl enable --now cri-docker.socket

 # Installing kubeadm, kubelet and kubectl (on both master and slave)

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
```

```
sudo curl -fsSLo /etc/apt/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

```
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

```
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

 # Forwarding IPv4 and letting iptables see bridged traffic (only on Master vm)

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# sysctl params required by setup, params persist across reboots
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system
```

#Run the below command only on master node (192.168.29.46 is master node ip, please use your master vm ip. Also use same pod network cider given below):

sudo kubeadm init --apiserver-advertise-address=192.168.29.46 --pod-network-cidr=10.244.0.0/16 --cri-socket=unix:///var/run/cri-dockerd.sock

mkdir -p $HOME/.kube

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

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

 # A token with join command will be displayed, copy and paste that in slave node to join the cluster. At the end "--cri-socket= unix:///var/run/cri-dockerd.sock" is added as I was getting Found multiple CRI endpoints on the host error

sudo kubeadm join 192.168.29.46:6443 --token 8te8jc.dtic56eqnqwufo3w --discovery-token-ca-cert-hash sha256:6fff1209a717cb7d1928e1027827d9a7e186c7fb4f161044d9dfac67ce9000f5 --cri-socket=unix:///var/run/cri-dockerd.sock (# run the token generated by your master vm & wait for it to complete)

 # Run the below command only on master node:

wget https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml

vi kube-flannel.yml

```
    containers:
    - args:
      - --ip-masq
      - --kube-subnet-mgr
      - --iface=enp0s8
```

 # In args add --iface=enp0s8 , here enp0s8 is master node ethernet adaptor. You can check your master node ethernet adaptor with ifconfig/ip add show command. After that save this file :wq

kubectl apply -f kube-flannel.yml **(# run on master node)**

#Master node output:

```
MINGW64:/e/Projects/kubeadm/kubemaster
vagrant@kubemaster:~$ kubectl get nodes
NAME         STATUS   ROLES           AGE     VERSION
kubemaster   Ready    control-plane   8m53s   v1.26.1
kubenode1    Ready    <none>          2m1s    v1.26.1
kubenode2    Ready    <none>          87s     v1.26.1
vagrant@kubemaster:~$ kubectl get pods -A
NAMESPACE      NAME                                 READY   STATUS    RESTARTS        AGE
kube-flannel   kube-flannel-ds-dnf6m                1/1     Running   0               4m45s
kube-flannel   kube-flannel-ds-gfdhr                1/1     Running   0               90s
kube-flannel   kube-flannel-ds-k7pzk                1/1     Running   0               2m4s
kube-system    coredns-787d4945fb-25krj             1/1     Running   0               8m7s
kube-system    coredns-787d4945fb-fcf7p             1/1     Running   0               8m7s
kube-system    etcd-kubemaster                      1/1     Running   0               8m33s
kube-system    kube-apiserver-kubemaster            1/1     Running   0               8m33s
kube-system    kube-controller-manager-kubemaster   1/1     Running   1 (3m45s ago)   8m33s
kube-system    kube-proxy-gkfxb                     1/1     Running   0               8m7s
kube-system    kube-proxy-rxbh2                     1/1     Running   0               2m4s
kube-system    kube-proxy-vxmpx                     1/1     Running   0               90s
kube-system    kube-scheduler-kubemaster            1/1     Running   2 (96s ago)     8m33s
vagrant@kubemaster:~$
```