Install Kubernetes master node and worker node in local VM (kodekloud)

Use VM
k8s-m
k8s-w1

Set hostname

```
hostname new-hostname     =set host name  ex: hostname rajiv
hostname                  =see the hostname
sudo su                   =need to logout and login to get the effect
```

Edit host file of k8-master

```
127.0.1.1 master
172.31.24.22 master.example.com master
172.31.30.46 worker1.example.com worker1
```

Edit host file of k8-worker1

```
127.0.1.1 worker1
172.31.24.22 master.example.com master
172.31.30.46 worker1.example.com worker1
```

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

step 1: Do this both master and worker node (need this setup for prerequisite)

```
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
```

```
lsmod | grep br_netfilter
lsmod | grep overlay
```

```
sysctl net.bridge.bridge-nf-call-iptables net.bridge.bridge-nf-call-ip6tables
net.ipv4.ip_forward
```

Step 2: Install docker

Install docker

Docker install in ubuntu
https://docs.docker.com/engine/install/ubuntu/

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" 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
```

```
#sudo systemctl status docker
#sudo systemctl start docker
#sudo systemctl enable docker
#sudo systemctl stop docker
```

or
only install containerd

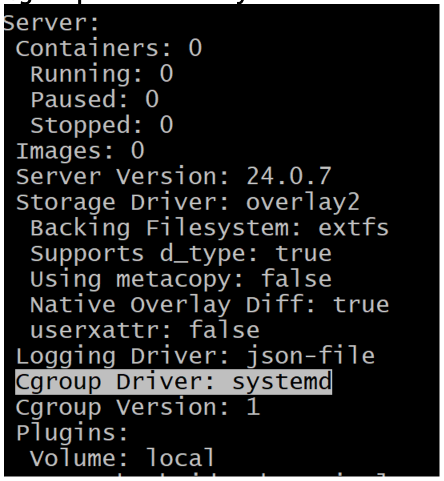
Step 3: containerd use system

#docker info

```
# vi /etc/docker/daemon.json
{
        "exec-opts": ["native.cgroupdriver=systemd"]
}
~
# systemctl daemon-reload
# systemctl restart docker
# docker info
```

```
Make sure

Cgroup Driver: systemd
```



```
Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 24.0.7
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: systemd
 Cgroup Version: 1
 Plugins:
  Volume: local
```

https://kubernetes.io/docs/setup/production-environment/container-runtimes/#containerd

Delete all content in contaierd config file add only this following 3 lines
# sudo vi /etc/containerd/config.toml

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
SystemdCgroup = true
```

# sudo systemctl restart containerd

Step 4: Installing kubeadm, kubelet and kubectl

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

```
sudo apt-get update
# apt-transport-https may be a dummy package; if so, you can skip that package
sudo apt-get install -y apt-transport-https ca-certificates curl
```

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg -
-dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | 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
```

Step 5: Disable swap and also disable firewall

Disable Swap
# sudo swapoff -a
#sudo vi /etc/fstab
delete the swap file then save and exit

Disable firewall
# sudo ufw disable

If its worker node then done for master node need to run the 6 step and for joining the worker node to master node need to run the step 8.

Step 6: run the below cluster command in master node only

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

```
# kubeadm init --pod-network-cidr=10.244.0.0/16
```

The 192.168.30.10 is the master node ip

After install successfully we get the following message at the bottom

```
Your Kubernetes control-plane has initialized successfully!

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 192.168.30.10:6443 --token im3yse.cfw1f4jyx1wqlsn3 \
      --discovery-token-ca-cert-hash sha256:f12831724f0fd2a4df061c9dfd779810af4d3292ac02ad8
```

Now run the 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
```

Now we can run the command

```
#kubectl get pods
```

it will now show any pod and show a message no resource found.

step 7: set the network we use weave net

Set the network

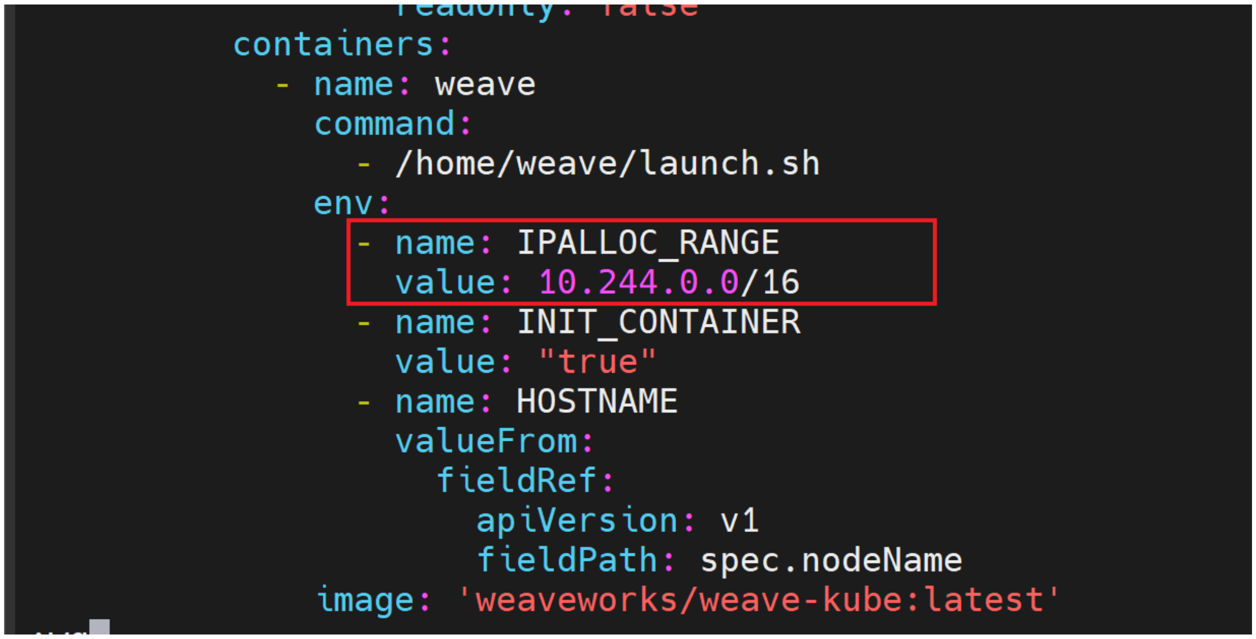
https://kubernetes.io/docs/concepts/cluster-administration/addons/

we use weave net

https://www.weave.works/docs/net/latest/kubernetes/kube-addon/

```
#wget https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml
#ll
#sudo vi weave-daemonset-k8s.yaml
```

now edit the file and the line as shown in the below image



```
      containers:
        - name: weave
          command:
            - /home/weave/launch.sh
          env:
            - name: IPALLOC_RANGE
              value: 10.244.0.0/16
            - name: INIT_CONTAINER
              value: "true"
            - name: HOSTNAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: spec.nodeName
          image: 'weaveworks/weave-kube:latest'
```

# kubectl apply -f weave-daemonset-k8s.yaml
# kubectl get pods -A

OR
#kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml

Step 8: join worker node to master node

Now go to the worker node and run the following command:

# sudo kubeadm join 192.168.30.10:6443 --token im3yse.cfw1f4jyx1wqlsn3 --discovery-token-ca-cert-hash sha256:f12831724f0fd2a4df061c9dfd779810af4d3292ac02ad85bf7e84bb8c08ed07

Step 9:

Go to the master node
#kubectl run nginx –image=nginx
#kubectl get pods
#kubectl delete pods nginx

Token related command

kubeadm token list	See all token list
kubeadm token create	Create a token
kubeadm token delete token-name	Delete a token
kubeadm token create --print-join-command	worker nodes join command