Installing Kubernetes on Ubuntu 22.04

Steps to be performed on both Master and Worker Nodes

Step1 to Step-5 are common steps

Step 1) Set hostname and add entries in the hosts file

Login to **both master node and worker nodes**, add Name,IP to /etc/hosts files

Add the following entries in /etc/hosts file on each node

```
192.168.1.4 master-node
192.168.1.5 worker-node1
192.168.1.6 worker-node2
```

Step 2) Disable swap & add kernel settings

```
sudo swapoff -a
```

```
sudo tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF
```

```
sudo modprobe overlay
```

```
sudo modprobe br_netfilter
```

```
sudo tee /etc/sysctl.d/kubernetes.conf <<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

Reload system:

```
sudo sysctl --system
```

Step 3) Install containerd run time

```
sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates
```

`sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates`

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/dock
```

`sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg`

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu  $(lsb_release -cs) stable"
```

`sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu  $(lsb_release -cs) stable"`

Pree ENTER to continue

```
sudo apt update
sudo apt install -y containerd.io
```

```
containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
```

```
sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml
```

`sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml`

```
sudo systemctl restart containerd
sudo systemctl enable containerd
```

Step 4) Add apt repository for Kubernetes

```
curl -fsSL https://dl.k8s.io/apt/doc/apt-key.gpg | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-archive-key
```

`curl -fsSL https://dl.k8s.io/apt/doc/apt-key.gpg | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-archive-keyring.gpg`

```
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xeni
```

`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`

Step 5) Install Kubernetes components Kubectl, kubeadm & kubelet

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

Perform the following only on Master Node

Step 6A) Initialize Kubernetes cluster with Kubeadm command

Now, we are all set to initialize Kubernetes cluster. Run the following Kubeadm command from the master node only.

```
sudo kubeadm init --control-plane-endpoint=master
#where master is my host name in the above

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Step 6B) Install Calico Pod Network Add-on

Run following kubectl command to install Calico network plugin from the master node:

```
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml
```

`kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml`

Perform on all worker nodes

Join both the worker nodes to the cluster by just copy pasting on screen instructions.

```
sudo kubeadm join master:6443 --token vt4ua6.wcma2y8pl4menxh2 \
   --discovery-token-ca-cert-hash sha256:0494aa7fc6ced8f8e7b20137ec0c5d2699dc5f8e616656932ff9173c94962a36
```

Check the nodes status from master node using kubectl command,

```
kubectl get nodes
```

As we can see nodes status is ‘NotReady’, so to make it active. We must install CNI (Container Network Interface) or network add-on plugins like Calico, Flannel and Weave-net.

Verify the status of pods in kube-system namespace,

```
kubectl get pods -n kube-system
```

Perfect, check the nodes status as well.

```
kubectl get nodes
```

Great, above confirms that nodes are active node.

Now, we can say that our Kubernetes cluster is functional.