Mohammad Saeed Pourmazar

https://github.com/MohammadSaeedPourmazar

https://gitlab.com/MohammadSaeedPourmazar

https://medium.com/@MohammadSaeedPourmazar

https://dev.to/MohammadSaeedPourmazar

https://www.youtube.com/@MohammadSaeedPourmazar

https://www.instagram.com/MohammadSaeedPourmazar

https://www.facebook.com/MohammadSaeedPourmazar

https://www.linkedin.com/in/MohammadSaeedPourmazar/

https://orcid.org/0009-0008-9383-419X

Install Kubernetes On Ubuntu (Master-Node)

**** Ensure Docker is installed on your system ****

1. Update System:

First: update your system packages to the latest version:

```
sudo apt update
sudo apt upgrade -y
sudo reboot
```

2. Set Hostname:

Set your VM's hostname to Master.Node:

```
sudo hostnamectl set-hostname Master.Node
```

3. Update /etc/hosts:

Edit the /etc/hosts file to include the VM's IP and hostname:

```
sudo nano /etc/hosts
```

*Add the following line * don't forget Worker.Nodes if you have * :*

```
IP Address   Master.Node
IP Address   Worker.Node
```

4. Disable Swap:

Kubernetes requires swap to be disabled:

```
sudo swapoff -a

sudo nano /etc/fstab
```

** Comment out any swap entries (add # in front) **

```
sudo mount -a
```

The sudo mount -a command is used to mount all file systems listed in /etc/fstab that are not currently mounted. This is useful when you've made changes to /etc/fstab and want to apply them without rebooting.

5. Load Kernel Modules:

Load the necessary kernel modules for Kubernetes networking:

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

sudo modprobe overlay
sudo modprobe br_netfilter
```

6. Set Kernel Parameters:

Configure the kernel parameters for Kubernetes:

```
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

sudo sysctl --system
```

7. Install Docker (Containerd):

Install containerd for container runtime:

```
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 --dearmor -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 apt update

sudo apt install -y containerd.io
```

Configure containerd:

```
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 systemctl restart containerd

sudo systemctl enable containerd
```

8. Install Kubernetes:

*** Install kubeadm, kubelet, and kubectl (The Latest) : ***

```
sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl gpg

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.32/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:/v1.32/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
```

** Installing Specific Versions of Kubernetes Components: **

To install a specific version of kubeadm, kubelet, and kubectl (e.g., v1.30.0) on a Debian-based system, follow these steps:

1. Add the Correct Kubernetes Repository:

```
sudo apt-get update
sudo apt-get install -y curl
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo tee /etc/apt/keyrings/kubernetes-apt-keyring.asc
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.asc] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

2. Update the package list again:

```
sudo apt-get update
```

3. Install specific versions of kubelet, kubeadm, and kubectl:

```
sudo apt-get install -y kubelet=1.30.0-1.1 kubeadm=1.30.0-1.1 kubectl=1.30.0-1.1
```

4. Prevent these packages from being automatically updated:

```
sudo apt-mark hold kubelet kubeadm kubectl
```

9. Initialize Kubernetes Cluster:

Initialize the Kubernetes cluster on the master node (your VM):

** Use the VM's IP address directly or Setup DNS resolution for the hostname **

** Use the IP Address Directly **

```
sudo kubeadm init --pod-network-cidr=10.10.0.0/16 --control-plane-endpoint=(Master.Node Or IP Address)
```

10. Configure Kubernetes CLI:

Set up kubectl for your user:

```
mkdir -p $HOME/.kube

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

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

11. Retrieve And Save the Token:

Run the following command on the master node to get the join token:

```
kubeadm token create --print-join-command
```

** Save kubeadm join code for joining worker nodes **

Examle :

```
kubeadm join 13.59.105.80:6443 --token dw3d0h.dekblj8ekzho8eng \
        --discovery-token-ca-cert-hash
sha256:dea038ba36a20bf330f393ee1195130d5b3b6d2c831244a5c8f7260be29b83d1
```

12. Install Pod Network (Calico):

Install the Calico network plugin:

```
curl https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml -O
```

```
sudo nano calico.yaml
```

** Open the file you just downloaded and find the section* CALICO_IPV4POOL_CIDR ***

CALICO_IPV4POOL_CIDR

** We correct the IP range to match the CIDR of the pod network in the command sudo kubeadm init. In my example, 10.10.0.0/16the file after editing has the following form **

```
# The default IPv4 pool to create on startup if none exists. Pod IPs will be
# chosen from this range. Changing this value after installation will have
# no effect. This should fall within `--cluster-cidr`.
- name: CALICO_IPV4POOL_CIDR
  value: '10.10.0.0/16'
# Disable file logging so `kubectl logs` works.
- name: CALICO_DISABLE_FILE_LOGGING
  value: 'true'
```

```
kubectl apply -f calico.yaml
```

13. Reseting Kubernetes Setup and Synchronize Time Using NTP:

```
sudo apt-get install ntp
sudo apt-get install ntpdate
sudo ntpdate ntp.ubuntu.com
```

14. Verify Cluster Status:

Verify that the cluster is up and running:

```
kubectl get pods -n kube-system
kubectl cluster-info
kubectl get nodes
```

** The output should show your node as Ready. **