**Setup Kubernetes Cluster in Ubuntu 22.04 using Kubeadm**

**Steps Involved:**

Upgrade Ubuntu, assign Static IP and Set Hostnames

Disable Swap

Add kernel settings

Allow Port and Disable firewall

Install Docker

Install Kubernetes Tools

Initialize the master node

Installing Pod Network using Calico network

Join Worker Nodes

Verify Cluster Status

To set up a Kubernetes 3-node cluster using kubeadm in Ubuntu 22, you'll need to perform the following steps on each of the three nodes:

**Step 1: Set Hostnames**

Install below tools in fresh Ubuntu

# apt install net-tools openssh-server curl nano

After login to all 3 Ubuntu server - kubemaster, kubeworker-1 and kubeworker-2

Then update and upgrade

# apt update

# apt upgrade -y

# reboot

Assign Static IP

# nmtui



**Edit /etc/hosts file to map server IP with hostname**

# nano /etc/hosts

192.168.100.7 kmaster.iiml.local kmaster

192.168.100.8 kworker1.iiml.local kworker1

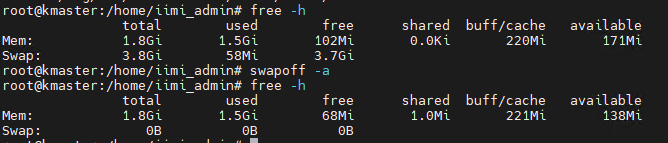
192.168.100.9 kworker2.iiml.local kworker2



**Step 2: Disable Swap**

# swapoff -a

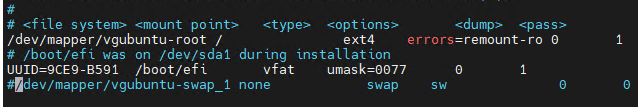
# free -h



Comment swap partition in /etc/fstab

# nano /etc/fstab

# mount -a



**Step 3: Add kernel settings**

Load following modules in all the nodes

# sudo tee /etc/modules-load.d/containerd.conf<<EOF

overlay

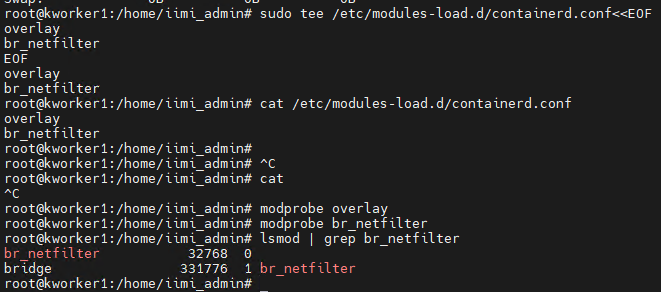
br\_netfilter

EOF

# modprobe overlay

# modprobe br\_netfilter

# lsmod | grep br\_netfilter



**Set the following 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

**Step 4: Add Rule and Disable Firewall**

Allow necessary ports in firewall

sudo ufw allow 22/tcp

sudo ufw allow 6443/tcp

sudo ufw allow 2379:2380/tcp

sudo ufw allow 10250:10255/tcp

sudo ufw status

sudo ufw disable

# ufw disbale

# ufw status

**Step 5: Install Docker**

1. Update the package list:

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

2. Install Docker:

# sudo apt install docker.io

3. Start and enable Docker:

# sudo systemctl start docker

# sudo systemctl enable docker

**Step 6: Install Kubernetes tools**

1. Add the Kubernetes repository key:

# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

2. Add the Kubernetes repository:

# sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

# echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

3. Update the package list:

# sudo apt update

4. Install Kubernetes components:

# sudo apt install -y kubelet kubeadm kubectl

5. Hold the Kubernetes packages at the current version to prevent them from being upgraded:

# sudo apt-mark hold kubelet kubeadm kubectl

**Step 7: Initialize the Master Node:**

1. On the master node, initialize the cluster :

# sudo kubeadm config images pull

# sudo kubeadm init --pod-network-cidr=172.17.0.0/16

or#

# sudo kubeadm init --pod-network-cidr=172.17.0.0/16 --control-plane-endpoint=kmaster.iiml.local

2. After the initialization is complete, follow the instructions provided by kubeadm to set up the kubeconfig file and join other nodes to the cluster.

1. Configure kubectl for the master node

Create the .kube directory:

# mkdir -p $HOME/.kube

Copy the kubeconfig file:

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

Set the ownership of the kubeconfig file:

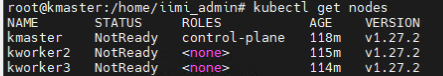
# sudo chown $(id -u):$(id -g) $HOME/.kube/config

Check cluster info

# kubectl cluster-info

# kubectl get nodes

Output:



As we can see nodes status is ‘NotReady’, so to make it active. We must install CNI

**Step 8: Install a Pod network add-on:**

1. Install a Pod network add-on to enable communication between Pods across nodes. Calico is one such option:

# curl https://raw.githubusercontent.com/projectcalico/calico/v3.25.1/manifests/calico.yaml -O

If you are using pod CIDR 192.168.0.0/16, make sure you uncomment the CALICO\_IPV4POOL\_CIDR variable in the manifest and set it to the same value as your chosen pod CIDR.

# 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.

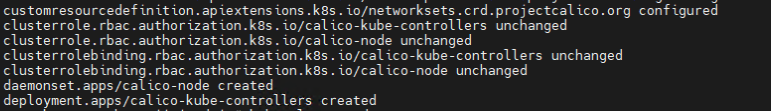
- name: CALICO\_DISABLE\_FILE\_LOGGING

value: "true"

Due to the large size of the CRD bundle, below kubectl apply might exceed request limits. Instead, use kubectl create or kubectl replace

# kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.1/manifests/calico.yaml

Output:



**Step 9: Join worker nodes to the cluster:**

1. On each worker node, run the join command provided by kubeadm when you initialized the master node.

# kubeadm join 192.168.100.7:6443 --token o397tz.ulkott25ntw1wm0o \

--discovery-token-ca-cert-hash sha256:c1ae3346c087cf940308b14490313ee31dda08be6c3c4fd8022b80a11ec9d5cf

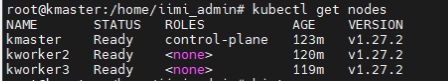
**Step 10: Verify cluster status:**

1. On the master node, check the status of the cluster and ensure that all nodes are ready:

# kubectl get pods -n kube-system

# kubectl get nodes

You should see all three nodes with a status of "Ready."



If connection refused try below

# systemctl restart docker

# systemctl restart kubelet