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

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.7  netmask 255.255.255.0  broadcast 192.168.100.255
        ether 00:15:5d:85:50:2c  txqueuelen 1000  (Ethernet)
```

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

```
192.168.100.7   kmaster.iiml.local kmaster
192.168.100.8   kworker1.iiml.local kworker1
192.168.100.9   kowerker2.iiml.local kworker2
```

Step 2: Disable Swap

swapoff -a
free -h

```
root@kmaster:/home/iimi_admin# free -h
               total        used        free      shared  buff/cache   available
Mem:           1.8Gi       1.5Gi       102Mi       0.0Ki       220Mi       171Mi
Swap:          3.8Gi        58Mi       3.7Gi
root@kmaster:/home/iimi_admin# swapoff -a
root@kmaster:/home/iimi_admin# free -h
               total        used        free      shared  buff/cache   available
Mem:           1.8Gi       1.5Gi        68Mi       1.0Mi       221Mi       138Mi
Swap:             0B          0B          0B
```

Comment swap partition in /etc/fstab
nano /etc/fstab
mount -a

```
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
/dev/mapper/vgubuntu-root /               ext4    errors=remount-ro 0       1
# /boot/efi was on /dev/sda1 during installation
UUID=9CE9-B591  /boot/efi       vfat    umask=0077      0       1
#/dev/mapper/vgubuntu-swap_1 none            swap    sw              0       0
```

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

```
root@kworker1:/home/iimi_admin# sudo tee /etc/modules-load.d/containerd.conf<<EOF
overlay
br_netfilter
EOF
overlay
br_netfilter
root@kworker1:/home/iimi_admin# cat /etc/modules-load.d/containerd.conf
overlay
br_netfilter
root@kworker1:/home/iimi_admin#
root@kworker1:/home/iimi_admin# ^C
root@kworker1:/home/iimi_admin# cat
^C
root@kworker1:/home/iimi_admin# modprobe overlay
root@kworker1:/home/iimi_admin# modprobe br_netfilter
root@kworker1:/home/iimi_admin# lsmod | grep br_netfilter
br_netfilter           32768  0
bridge                331776  1 br_netfilter
root@kworker1:/home/iimi_admin#
```

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.

3. 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:

```
root@kmaster:/home/iimi_admin# kubectl get nodes
NAME       STATUS     ROLES           AGE    VERSION
kmaster    NotReady   control-plane   118m   v1.27.2
kworker2   NotReady   <none>          115m   v1.27.2
kworker3   NotReady   <none>          114m   v1.27.2
```

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:

```
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org configured
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers unchanged
clusterrole.rbac.authorization.k8s.io/calico-node unchanged
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers unchanged
clusterrolebinding.rbac.authorization.k8s.io/calico-node unchanged
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
```

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."

```
root@kmaster:/home/iimi_admin# kubectl get nodes
NAME       STATUS   ROLES           AGE    VERSION
kmaster    Ready    control-plane   123m   v1.27.2
kworker2   Ready    <none>          120m   v1.27.2
kworker3   Ready    <none>          119m   v1.27.2
```

If connection refused try below

```
# systemctl restart docker
# systemctl restart kubelet
```