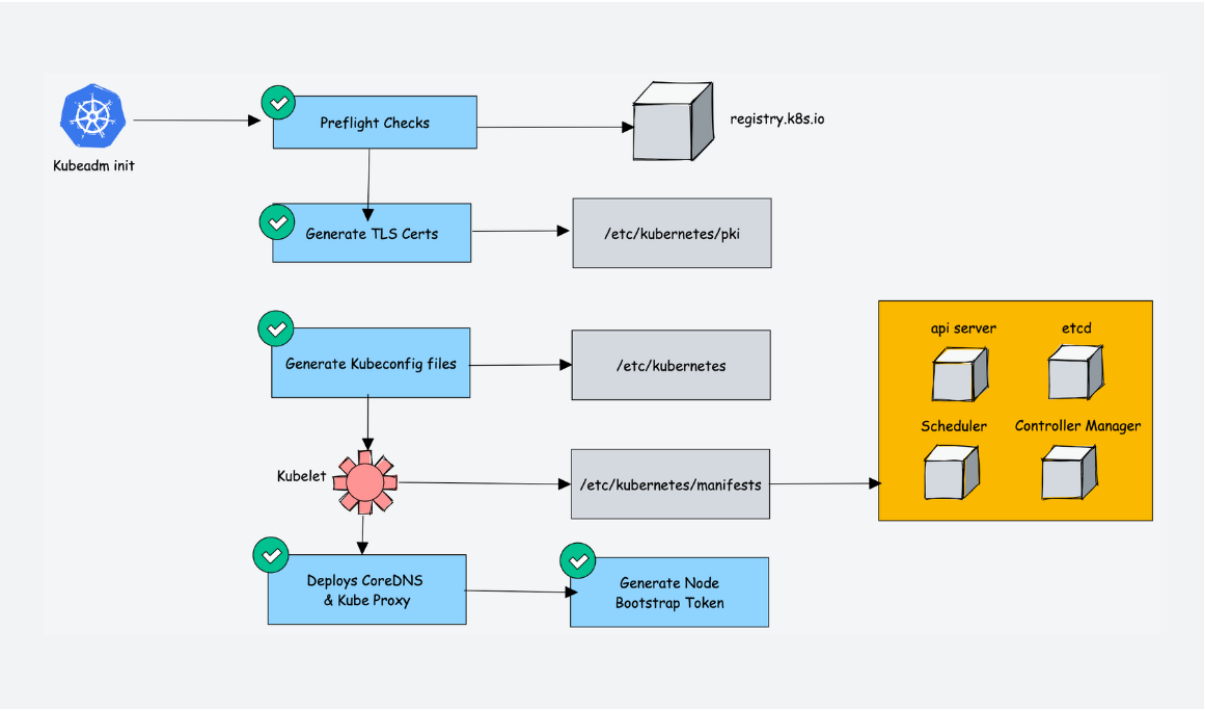
Tuesday, 2 April 2024

Setup Kubernetes Cluster Using Kubeadm



Kubeadm Setup Prerequisites

1. Minimum two **Ubuntu nodes** [One master and one worker node]. You can have more worker nodes as per your requirement.

2. The master node should have a minimum of **2 vCPU and 2GB RAM**.

3. For the worker nodes, a minimum of 1vCPU and 2 GB RAM is recommended.

4. **10.X.X.X/X** network range with static IPs for master and worker nodes. We will be using the **192.x.x.x** series as the pod network range that will be used by the Calico network plugin. Make sure the Node IP range and pod IP range don't overlap.

Kubeadm Port Requirements

Control-plane node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services**	All

STEP 1: ENABLE IPTABLES BRIDGED TRAFFIC ON ALL THE NODES

Execute the following commands on **all the nodes** for IPtables to see bridged traffic.

#cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay

br_netfilter

EOF

```
ubuntu@ip-172-31-32-138:~$ sudo -i
root@ip-172-31-32-138:~# hostnamectl set-hostname master
root@ip-172-31-32-138:~# bash
root@master:~# cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
overlay
br_netfilter
```

#sudo modprobe overlay

#sudo modprobe br_netfilter

```
root@master:~# sudo modprobe overlay
sudo modprobe br_netfilter
root@master:~# 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
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
```

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

```
root@master:~# sudo modprobe overlay
sudo modprobe br_netfilter
root@master:~# 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
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
```

Apply sysctl params without reboot

#sudo sysctl —system

```
root@master:~# sudo sysctl --system
* Applying /etc/sysctl.d/10-console-messages.conf ...
kernel.printk = 4 4 1 7
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
kernel.kptr_restrict = 1
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
kernel.sysrq = 176
* Applying /etc/sysctl.d/10-network-security.conf ...
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.all.rp_filter = 2
* Applying /etc/sysctl.d/10-ptrace.conf ...
kernel.yama.ptrace_scope = 1
* Applying /etc/sysctl.d/10-zeropage.conf ...
vm.mmap_min_addr = 65536
* Applying /usr/lib/sysctl.d/50-default.conf ...
kernel.core_uses_pid = 1
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.default.accept_source_route = 0
sysctl: setting key "net.ipv4.conf.all.accept_source_route": Invalid argument
net.ipv4.conf.default.promote_secondaries = 1
sysctl: setting key "net.ipv4.conf.all.promote_secondaries": Invalid argument
net.ipv4.ping_group_range = 0 2147483647
net.core.default_qdisc = fq_codel
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
fs.protected_regular = 1
fs.protected_fifos = 1
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
kernel.pid_max = 4194304
* Applying /etc/sysctl.d/99-cloudimg-ipv6.conf ...
```

STEP 2: DISABLE SWAP ON ALL THE NODES

#sudo swapoff -a

(crontab -l 2>/dev/null; echo "@reboot /sbin/swapoff -a") | crontab - || true

```
root@master:~# sudo swapoff -a
(crontab -l 2>/dev/null; echo "@reboot /sbin/swapoff -a") | crontab - || true
```

STEP 3: INSTALL CRI-O RUNTIME ON ALL THE NODES

Execute the following commands **on all the nodes** to install required dependencies and the latest version of CRIO.

#sudo apt-get update -y

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

```
root@master:~# sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates
Hit:1 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:5 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:6 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:7 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:8 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:9 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:10 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1519 kB]
Get:11 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [293 kB]
Get:12 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1644 kB]
Get:13 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:14 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [274 kB]
Get:15 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1060 kB]
Get:16 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [241 kB]
Get:17 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [22.1 kB]
Get:18 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [49.6 kB]
Get:19 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [12.0 kB]
Get:20 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [472 B]
Get:21 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [67.1 kB]
Get:22 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [11.0 kB]
Get:23 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [388 B]
Get:24 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:25 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [28.4 kB]
Get:26 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.2 kB]
Get:27 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [644 B]
Get:28 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:29 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1303 kB]
Get:30 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [233 kB]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1616 kB]
Get:32 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [271 kB]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [852 kB]
Get:34 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [163 kB]
Get:35 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [16.8 kB]
Get:36 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [37.1 kB]
Get:37 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [7476 B]
Get:38 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [260 B]
```

#curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key |

gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg

echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" |

tee /etc/apt/sources.list.d/cri-o.list

```
root@master:~# curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key |
    gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /"
|
    tee /etc/apt/sources.list.d/cri-o.list
deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /
```

#sudo apt-get update -y

#sudo apt-get install -y cri-o

#sudo systemctl daemon-reload

#sudo systemctl enable crio —now

#sudo systemctl start crio.service

```
root@master:~# sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service
```

Install crictl.

crictl, a CLI utility to interact with the containers created by the container runtime.

#VERSION="v1.28.0"

wget https://github.com/kubernetes-sigs/cri-tools/releases/download/$VERSION/crictl-$VERSION-linux-amd64.tar.gz

sudo tar zxvf crictl-$VERSION-linux-amd64.tar.gz -C /usr/local/bin

rm -f crictl-$VERSION-linux-amd64.tar.gz

```
ubuntu@master:~$ VERSION="v1.28.0"
wget https://github.com/kubernetes-sigs/cri-tools/releases/download/$VERSION/crictl-$VERSION-linux-amd64.tar.gz
sudo tar zxvf crictl-$VERSION-linux-amd64.tar.gz -C /usr/local/bin
rm -f crictl-$VERSION-linux-amd64.tar.gz
--2024-04-02 06:50:02--  https://github.com/kubernetes-sigs/cri-tools/releases/download/v1.28.0/crictl-v1.28.0-linux-amd64
.tar.gz
Resolving github.com (github.com)... 20.27.177.113
Connecting to github.com (github.com)|20.27.177.113|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/80172100/c07e7dde-d128-43aa-b1ec-08
484b098c32?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAVCODYLSA53PQK4ZA%2F20240402%2Fus-east-1%2Fs3%2Faws4_reque
st&X-Amz-Date=20240402T065002Z&X-Amz-Expires=300&X-Amz-Signature=c25273ddc7a1963cadb4515c16ddd17ecfabd52371d57f06da09a572f
918445f&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=80172100&response-content-disposition=attachment%3B%20filenam
e%3Dcrictl-v1.28.0-linux-amd64.tar.gz&response-content-type=application%2Foctet-stream [following]
--2024-04-02 06:50:02--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/80172100/c07e7dde-d1
28-43aa-b1ec-08484b098c32?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAVCODYLSA53PQK4ZA%2F20240402%2Fus-east-1%2F
s3%2Faws4_request&X-Amz-Date=20240402T065002Z&X-Amz-Expires=300&X-Amz-Signature=c25273ddc7a1963cadb4515c16ddd17ecfabd52371
d57f06da09a572f918445f&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=80172100&response-content-disposition=attachme
nt%3B%20filename%3Dcrictl-v1.28.0-linux-amd64.tar.gz&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.1
33, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23868098 (23M) [application/octet-stream]
Saving to: 'crictl-v1.28.0-linux-amd64.tar.gz'

crictl-v1.28.0-linu 100%[===================>]  22.76M   143MB/s    in 0.2s

2024-04-02 06:50:03 (143 MB/s) - 'crictl-v1.28.0-linux-amd64.tar.gz' saved [23868098/23868098]

crictl
```

STEP 4: INSTALL KUBEADM & KUBELET & KUBECTL ON ALL NODES

Download the GPG key for the Kubernetes APT repository **on all the nodes.**

#KUBERNETES_VERSION=1.29

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v$KUBERNETES_VERSION/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:/v$KUBERNETES_VERSION/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@master:~$ KUBERNETES_VERSION=1.29

sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v$KUBERNETES_VERSION/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyring
s/kubernetes-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v$KUBERNETES_VERSION/
deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /
```

#sudo apt-get update -y

```
ubuntu@master:~$ sudo apt-get update -y
Hit:1 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease:/main/deb  InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  InRelease [1186 B]
Hit:6 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  Packages [6511 B]
Fetched 7697 B in 1s (12.7 kB/s)
Reading package lists... Done
```

You can use the following commands to find the latest versions

#apt-cache madison kubeadm | tac

```
ubuntu@master:~$ apt-cache madison kubeadm | tac
   kubeadm | 1.29.0-1.1 | https://pkgs.k8s.io/core:/stable:/v1.29/deb  Packages
   kubeadm | 1.29.1-1.1 | https://pkgs.k8s.io/core:/stable:/v1.29/deb  Packages
   kubeadm | 1.29.2-1.1 | https://pkgs.k8s.io/core:/stable:/v1.29/deb  Packages
   kubeadm | 1.29.3-1.1 | https://pkgs.k8s.io/core:/stable:/v1.29/deb  Packages
```

Specify the version as shown below. Here I am using 1.29.0-1.1

#sudo apt-get install -y kubelet=1.29.0-1.1 kubectl=1.29.0-1.1 kubeadm=1.29.0-1.1

```
ubuntu@master:~$ sudo apt-get install -y kubelet=1.29.0-1.1 kubectl=1.29.0-1.1 kubeadm=1.29.0-1.1
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools ebtables kubernetes-cni socat
The following NEW packages will be installed:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 39 not upgraded.
Need to get 92.4 MB of archives.
After this operation, 346 MB of additional disk space will be used.
Get:2 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 conntrack amd64 1:1.4.6-2build2 [33.5 kB]
Get:1 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  cri-tools 1.29.0-1.1 [20.1 M
B]
Get:4 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 ebtables amd64 2.0.11-4build2 [84.9 kB]
Get:6 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 socat amd64 1.7.4.1-3ubuntu4 [349 kB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  kubernetes-cni 1.3.0-1.1 [31
.4 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  kubelet 1.29.0-1.1 [19.8 MB]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  kubectl 1.29.0-1.1 [10.5 MB]
Get:8 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  kubeadm 1.29.0-1.1 [10.1 MB]
Fetched 92.4 MB in 3s (29.0 MB/s)
Selecting previously unselected package conntrack.
(Reading database ... 65311 files and directories currently installed.)
Preparing to unpack .../0-conntrack_1%3a1.4.6-2build2_amd64.deb ...
Unpacking conntrack (1:1.4.6-2build2) ...
Selecting previously unselected package cri-tools.
Preparing to unpack .../1-cri-tools_1.29.0-1.1_amd64.deb ...
Unpacking cri-tools (1.29.0-1.1) ...
Selecting previously unselected package ebtables.
Preparing to unpack .../2-ebtables_2.0.11-4build2_amd64.deb ...
Unpacking ebtables (2.0.11-4build2) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../3-kubernetes-cni_1.3.0-1.1_amd64.deb ...
Unpacking kubernetes-cni (1.3.0-1.1) ...
Selecting previously unselected package socat.
```

install the latest version

#sudo apt-get install -y kubelet kubeadm kubectl

Add hold to the packages to prevent upgrades.

#sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@master:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

Add the node IP to KUBELET_EXTRA_ARGS.

#sudo apt-get install -y jq

local_ip="$(ip --json addr show eth0 | jq -r '.[0].addr_info[] | select(.family == "inet") | .local')"

cat > /etc/default/kubelet << EOF

KUBELET_EXTRA_ARGS=--node-ip=$local_ip

EOF

```
ubuntu@master:~$ sudo apt-get install -y jq
local_ip="$(ip --json addr show eth0 | jq -r '.[0].addr_info[] | select(.family == "inet") | .local')"
cat > /etc/default/kubelet << EOF
KUBELET_EXTRA_ARGS=--node-ip=$local_ip
EOF
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libjq1 libonig5
The following NEW packages will be installed:
  jq libjq1 libonig5
0 upgraded, 3 newly installed, 0 to remove and 39 not upgraded.
Need to get 357 kB of archives.
After this operation, 1087 kB of additional disk space will be used.
Get:1 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libonig5 amd64 6.9.7.1-2build1 [172 kB]
Get:2 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libjq1 amd64 1.6-2.1ubuntu3 [133 kB]
Get:3 http://ap-northeast-3.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 jq amd64 1.6-2.1ubuntu3 [52.5 kB]
Fetched 357 kB in 0s (1752 kB/s)
Selecting previously unselected package libonig5:amd64.
(Reading database ... 65429 files and directories currently installed.)
Preparing to unpack .../libonig5_6.9.7.1-2build1_amd64.deb ...
Unpacking libonig5:amd64 (6.9.7.1-2build1) ...
Selecting previously unselected package libjq1:amd64.
Preparing to unpack .../libjq1_1.6-2.1ubuntu3_amd64.deb ...
Unpacking libjq1:amd64 (1.6-2.1ubuntu3) ...
Selecting previously unselected package jq.
Preparing to unpack .../jq_1.6-2.1ubuntu3_amd64.deb ...
Unpacking jq (1.6-2.1ubuntu3) ...
Setting up libonig5:amd64 (6.9.7.1-2build1) ...
Setting up libjq1:amd64 (1.6-2.1ubuntu3) ...
Setting up jq (1.6-2.1ubuntu3) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.6) ...
Scanning processes...
Scanning linux images...
```

Same configuration on both the instance till now.

STEP 5: INITIALIZE KUBEADM ON MASTER NODE TO SETUP CONTROL PLANE

1. Master Node with Private IP

Replace 10.0.0.10 with the IP of your master node.

#IPADDR="10.0.0.10"

NODENAME=$(hostname -s)

POD_CIDR="192.168.0.0/16"

#sudo kubeadm init **--apiserver-advertise-address**=$IPADDR --apiserver-cert-extra-sans=$IPADDR --pod-network-cidr=$POD_CIDR --node-name $NODENAME --ignore-preflight-errors Swap

2. Master Node With Public IP

#IPADDR=$(curl ifconfig.me && echo "")

NODENAME=$(hostname -s)

POD_CIDR="192.168.0.0/16"

```
ubuntu@master:~$ IPADDR=$(curl ifconfig.me && echo "")
NODENAME=$(hostname -s)
POD_CIDR="192.168.0.0/16"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    12  100    12    0     0     67      0 --:--:-- --:--:-- --:--:--    67
```

#sudo kubeadm init **--control-plane-endpoint**=$IPADDR --apiserver-cert-extra-sans=$IPADDR --pod-network-cidr=$POD_CIDR --node-name $NODENAME --ignore-preflight-errors Swap

```
ubuntu@master:~$ sudo kubeadm init --control-plane-endpoint=$IPADDR  --apiserver-cert-extra-sans=$IPADDR  --pod-network-cidr=$POD_CIDR --node-name $NODENAME --ignore-preflight-er
rors Swap
[init] Using Kubernetes version: v1.29.3
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local master] and IPs [10.96.0.1 172.3
1.32.138 15.168.20.92]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master] and IPs [172.31.32.138 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master] and IPs [172.31.32.138 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
```

```
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/

You can now join any number of control-plane nodes by copying certificate authorities
and service account keys on each node and then running the following as root:

  kubeadm join 15.168.20.92:6443 --token rfq20y.fwo0wrwxeg83fce9 \
        --discovery-token-ca-cert-hash sha256:34d245b39f7250cedf691855725605889b56ca4d1d00f1aaa10a1aec0e4b4e00 \
        --control-plane

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 15.168.20.92:6443 --token rfq20y.fwo0wrwxeg83fce9 \
        --discovery-token-ca-cert-hash sha256:34d245b39f7250cedf691855725605889b56ca4d1d00f1aaa10a1aec0e4b4e00
```

mkdir -p $HOME/.kube

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

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

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

Now, verify the kubeconfig by executing the following kubectl command to list all the pods in the kube-system namespace.

#kubectl get po -n kube-system

```
ubuntu@master:~$ kubectl get po -n kube-system
NAME                              READY   STATUS    RESTARTS   AGE
coredns-76f75df574-4v78l          1/1     Running   0          3m16s
coredns-76f75df574-vk9wv          1/1     Running   0          3m16s
etcd-master                       1/1     Running   0          3m30s
kube-apiserver-master             1/1     Running   0          3m30s
kube-controller-manager-master    1/1     Running   0          3m32s
kube-proxy-ftjn8                  1/1     Running   0          3m16s
kube-scheduler-master             1/1     Running   0          3m30s
```

You verify all the cluster component health statuses using the following command.

#kubectl get —raw='/readyz?verbose'

```
ubuntu@master:~$ kubectl get --raw='/readyz?verbose'
[+]ping ok
[+]log ok
[+]etcd ok
[+]etcd-readiness ok
[+]informer-sync ok
[+]poststarthook/start-kube-apiserver-admission-initializer ok
[+]poststarthook/generic-apiserver-start-informers ok
[+]poststarthook/priority-and-fairness-config-consumer ok
[+]poststarthook/priority-and-fairness-filter ok
[+]poststarthook/storage-object-count-tracker-hook ok
[+]poststarthook/start-apiextensions-informers ok
[+]poststarthook/start-apiextensions-controllers ok
[+]poststarthook/crd-informer-synced ok
[+]poststarthook/start-service-ip-repair-controllers ok
[+]poststarthook/rbac/bootstrap-roles ok
[+]poststarthook/scheduling/bootstrap-system-priority-classes ok
[+]poststarthook/priority-and-fairness-config-producer ok
[+]poststarthook/start-system-namespaces-controller ok
[+]poststarthook/bootstrap-controller ok
[+]poststarthook/start-cluster-authentication-info-controller ok
[+]poststarthook/start-kube-apiserver-identity-lease-controller ok
[+]poststarthook/start-kube-apiserver-identity-lease-garbage-collector ok
[+]poststarthook/start-legacy-token-tracking-controller ok
[+]poststarthook/aggregator-reload-proxy-client-cert ok
[+]poststarthook/start-kube-aggregator-informers ok
[+]poststarthook/apiservice-registration-controller ok
[+]poststarthook/apiservice-status-available-controller ok
[+]poststarthook/kube-apiserver-autoregistration ok
[+]autoregister-completion ok
[+]poststarthook/apiservice-openapi-controller ok
[+]poststarthook/apiservice-openapiv3-controller ok
[+]poststarthook/apiservice-discovery-controller ok
[+]shutdown ok
readyz check passed
```

You can get the cluster info using the following command.

#kubectl cluster-info

```
ubuntu@master:~$ kubectl cluster-info
Kubernetes control plane is running at https://15.168.20.92:6443
CoreDNS is running at https://15.168.20.92:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

STEP 6: JOIN WORKER NODES TO KUBERNETES MASTER NODE

Run this command on master node copy output.

#kubeadm token create —print-join-command

```
ubuntu@master:~$ kubeadm token create --print-join-command
kubeadm join 15.168.20.92:6443 --token ptjd52.0xomampnxx0uaqks --discovery-token-ca-cert-hash sha256:34d245b39f7250cedf691855725605889b56ca4d1d00f1aaa10a1aec0e4b4e00
```

Run the same output with #sudo in worker node.

#kubeadm token create --print-join-command

kubeadm join 15.168.20.92:6443 --token s67m72.s7tq7q29jp97pdug --discovery-token-ca-cert-hash sha256:34d245b39f7250cedf691855725605889b56ca4d1d00f1aaa10a1aec0e4b4e00

```
root@worker:~# kubeadm token create --print-join-command
kubeadm join 15.168.20.92:6443 --token s67m72.s7tq7q29jp97pdug --discovery-token-ca-cert-hash sha256:34d245b39f7250cedf691855725605889b56ca4d1d00f1aaa10a1aec0e4b4e00
failed to load admin kubeconfig: open /root/.kube/config: no such file or directory
To see the stack trace of this error execute with --v=5 or higher
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Now execute the **kubectl command from the master node** to check if the node is added to the master.

#kubectl get nodes

```
ubuntu@master:~$ kubectl get nodes
NAME      STATUS   ROLES           AGE    VERSION
master    Ready    control-plane   26m    v1.29.3
worker    Ready    <none>          119s   v1.29.3
```

STEP 7: INSTALL CALICO NETWORK PLUGIN FOR POD NETWORKING

Kubeadm does not configure any network plugin. You need to install a network plugin of your choice for kubernetes pod networking and enable network policy.

I am using the Calico network plugin for this setup.

#kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

```
ubuntu@master:~$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
```

you will see calico pods and running CoreDNS pods.

#kubectl get po -n kube-system

```
ubuntu@master:~$ kubectl get po -n kube-system
NAME                                      READY   STATUS    RESTARTS   AGE
calico-kube-controllers-658d97c59c-hnq89  1/1     Running   0          30s
calico-node-lkr9j                         0/1     Running   0          30s
calico-node-q7swc                         0/1     Running   0          30s
coredns-76f75df574-4v78l                  1/1     Running   0          30m
coredns-76f75df574-vk9wv                  1/1     Running   0          30m
etcd-master                               1/1     Running   0          30m
kube-apiserver-master                     1/1     Running   0          30m
kube-controller-manager-master            1/1     Running   0          30m
kube-proxy-ftjn8                          1/1     Running   0          30m
kube-proxy-nbd8j                          1/1     Running   0          6m32s
kube-scheduler-master                     1/1     Running   0          30m
```

STEP 8: SETUP KUBERNETES METRICS SERVER

#kubectl top nodes

```
ubuntu@master:~$ kubectl top nodes
error: Metrics API not available
```

To install the metrics server, execute the following metric server manifest file. It deploys metrics server version v0.6.2.

#kubectl apply -f https://raw.githubusercontent.com/techiescamp/kubeadm-scripts/main/manifests/metrics-server.yaml

```
ubuntu@master:~$ kubectl apply -f https://raw.githubusercontent.com/techiescamp/kubeadm-scripts/main/manifests/metrics-server.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
```

STEP 9: DEPLOY A SAMPLE NGINX APPLICATION

```
#cat <<EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
EOF
```

```
ubuntu@master:~$ cat <<EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
EOF
deployment.apps/nginx-deployment created
```

Expose the Nginx deployment on a **NodePort 32000**

```
#cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 32000
EOF
```

```
ubuntu@master:~$ cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 32000
EOF
service/nginx-service created
```

#kubectl get pods

```
ubuntu@master:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-7c79c4bf97-l85bc   1/1     Running   0          20s
nginx-deployment-7c79c4bf97-pssth   1/1     Running   0          20s
```

Hit public IP:portnumber