Create Kubernetes cluster using Kubeadm on Ubuntu 22.04 LTS

by Lokeshkumar

In this Article we are going to learn How to Create Kubernetes cluster using Kubeadm on Ubuntu 22.04 LTS and Join Worker Node to the Cluster.

Prerequisites:

- 2 or 3 Ubuntu 20.04 LTS System with Minimal Installation
- Minimum 2 or more CPU, 3 GB RAM.
- Disable SWAP on All node
- SSH Access with sudo privileges

Table of Contents

Firewall Ports/Inbound Traffic Ports for Kubernetes Cluster

S.No	Protocol	Direction	Port Range	Purpose	Used By
1	TCP	Inbound	6443*	Kubernetes API server	All
2	TCP	Inbound	2379-2380	etcd server client API	kube-apiserver,etcd
3	TCP	Inbound	10250	Kubelet API	Self, Control plane
4	TCP	Inbound	10251	kube-scheduler	Self
5	TCP	Inbound	10252	kube-controller-manager	Self

Master node :

You can clone the repository for reference.

```
git clone https://github.com/techiescamp/kubeadm-scripts
```

```
root@ip-1-0-0-73:~# git clone https://github.com/techiescamp/kubeadm-scripts
Cloning into 'kubeadm-scripts'...
remote: Enumerating objects: 286, done.
remote: Counting objects: 100% (120/120), done.
remote: Compressing objects: 100% (63/63), done.
remote: Total 286 (delta 76), reused 81 (delta 52), pack-reused 166
Receiving objects: 100% (286/286), 85.74 KiB | 4.76 MiB/s, done.
Resolving deltas: 100% (110/110), done.
```

Step #1:IPtables to see bridged traffic

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

```
sudo modprobe overlay
```

```
sudo modprobe br_netfilter
```

```
root@ip-1-0-0-73:~# sudo modprobe overlay
root@ip-1-0-0-73:~# sudo modprobe br_netfilter
```

```
# 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@ip-1-0-0-73:~# 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
root@ip-1-0-0-73:~#
```

Apply sysctl params without reboot

sudo sysctl –system

```
root@ip-1-0-0-73:~# 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 ...
net.ipv6.conf.all.use_tempaddr = 0
net.ipv6.conf.default.use_tempaddr = 0
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
fs.protected_fifos = 1
fs.protected_hardlinks = 1
```

```
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
kernel.pid_max = 4194304
* Applying /etc/sysctl.d/99-cloudimg-ipv6.conf ...
net.ipv6.conf.all.use_tempaddr = 0
net.ipv6.conf.default.use_tempaddr = 0
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
fs.protected_fifos = 1
fs.protected_hardlinks = 1
fs.protected_regular = 2
fs.protected_symlinks = 1
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/k8s.conf ...
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
* Applying /etc/sysctl.conf ...
root@ip-1-0-0-73:~#
```

Step #2:Disable swap on all the Nodes

For kubeadm to work properly, you need to disable swap on all the nodes using the following command.

```
sudo swapoff -a

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

```
root@ip-1-0-0-73:~# sudo swapoff -a
(crontab -l 2>/dev/null; echo "@reboot /sbin/swapoff -a") | crontab - || true
root@ip-1-0-0-73:~#
```

Step #3:Install CRI-O Runtime On All The Nodes

Create the .conf file to load the modules at bootup

```
cat <<EOF | sudo tee /etc/modules-load.d/crio.conf

overlay

br_netfilter

EOF
```

```
root@ip-1-0-0-73:~# cat <<EOF | sudo tee /etc/modules-load.d/crio.conf
overlay
br_netfilter
EOF
overlay
br_netfilter
root@ip-1-0-0-73:~#
```

Set up required sysctl params, these persist across reboots.

cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip_forward = 1

net.bridge.bridge-nf-call-ip6tables = 1

EOF

```
root@ip-1-0-0-73:~# cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
root@ip-1-0-0-73:~#
```

Execute the following commands to enable overlayFS & VxLan pod communication.

sudo modprobe overlay

sudo modprobe br_netfilter

```
root@ip-1-0-0-73:~# sudo modprobe overlay
root@ip-1-0-0-73:~# sudo modprobe br_netfilter
root@ip-1-0-0-73:~#
```

Set up required sysctl params, these persist across reboots.

```
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf

net.bridge.bridge-nf-call-iptables  = 1

net.ipv4.ip_forward                 = 1

net.bridge.bridge-nf-call-ip6tables = 1

EOF
```

```
root@ip-1-0-0-73:~# cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
root@ip-1-0-0-73:~#
```

Reload the parameters.

```
sudo sysctl –system
```

```
root@ip-1-0-0-73:~# 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 #4:Install Kubeadm & Kubelet & Kubectl on all Nodes

Install the required dependencies

Update your system packages:

```
sudo apt-get update
```

```
Get:20 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packag
es [41.6 kB]
Get:21 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-
en [9768 B]
Get:22 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f
Metadata [476 B]
Get:23 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [
41.7 kB]
Get:24 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [
10.5 kB]
Get:25 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Meta
data [388 B]
Get:26 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-
f Metadata [116 B]
Get:27 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packag
es [24.3 kB]
Get:28 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-
en [16.4 kB]
Get:29 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f
Metadata [644 B]
Get:30 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-
f Metadata [116 B]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [802 kB]
Get:32 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [169 kB]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [11.3 kB]
Get:34 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [882 kB]
Get:35 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [142 kB]
Get:36 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [536
 B]
Get:37 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [785 kB]
Get:38 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [143 kB]
Get:39 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [16.7
kB]
Get:40 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [36.5 kB]
Get:41 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [7060 B]
Get:42 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [260
 B]
Fetched 27.4 MB in 5s (5846 kB/s)
Reading package lists... Done
root@ip-1-0-0-73:~#
```

Install apt-transport-https curl

sudo apt-get install -y apt-transport-https curl

```
root@ip-1-0-0-73:~# sudo apt-get install -y apt-transport-https curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcurl4
The following NEW packages will be installed:
  apt-transport-https
The following packages will be upgraded:
  curl libcurl4
2 upgraded, 1 newly installed, 0 to remove and 125 not upgraded.
Need to get 486 kB of archives.
After this operation, 169 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-trans
port-https all 2.4.10 [1510 B]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.
81.0-1ubuntu1.13 [194 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libcurl4 amd6
4 7.81.0-1ubuntu1.13 [290 kB]
Fetched 486 kB in 0s (17.9 MB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 64295 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.4.10_all.deb ...
Unpacking apt-transport-https (2.4.10) ...
Preparing to unpack .../curl_7.81.0-1ubuntu1.13_amd64.deb ...
Unpacking curl (7.81.0-1ubuntu1.13) over (7.81.0-1ubuntu1.10) ...
Preparing to unpack .../libcurl4_7.81.0-1ubuntu1.13_amd64.deb ...
Unpacking libcurl4:amd64 (7.81.0-1ubuntu1.13) over (7.81.0-1ubuntu1.10) ...
Setting up apt-transport-https (2.4.10) ...
Setting up libcurl4:amd64 (7.81.0-1ubuntu1.13) ...
Setting up curl (7.81.0-1ubuntu1.13) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.
```

Add gpg keys

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

```
root@ip-1-0-0-73:~# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt
-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8
)).
OK
root@ip-1-0-0-73:~#
```

sudo vi /etc/apt/sources.list.d/kubernetes.list

Add this below lines in this file

deb https://apt.kubernetes.io/ kubernetes-xenial main

```
deb https://apt.kubernetes.io/ kubernetes-xenial main
~
~
~
~
~
~
~
~
~
~
~
```

Lets install kubelet kubeadm kubectl

sudo apt-get update

```
root@ip-1-0-0-73:~# sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8993 B]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packages [69.9 kB]
Fetched 78.9 kB in 2s (51.7 kB/s)
Reading package lists... Done
W: https://apt.kubernetes.io/dists/kubernetes-xenial/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in ap
```

sudo apt-get install -y kubelet kubeadm kubectl

```
root@ip-1-0-0-73:~# sudo apt-get install -y kubelet kubeadm kubectl
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 125 not upgraded.
Need to get 87.1 MB of archives.
After this operation, 336 MB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 conntrack amd64 1:1.4.6-2build2 [33.5 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 ebtables amd64 2.0.11-4build2 [84.9 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 socat amd64 1.7.4.1-3ubuntu4 [349 kB]
Get:4 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 cri-tools amd64 1.26.0-00 [18.9 MB]
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubernetes-cni amd64 1.2.0-00 [27.6 MB]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubelet amd64 1.28.2-00 [19.5 MB]
Get:7 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubectl amd64 1.28.2-00 [10.3 MB]
Get:8 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubeadm amd64 1.28.2-00 [10.3 MB]
Fetched 87.1 MB in 6s (14.1 MB/s)
Selecting previously unselected package conntrack.
(Reading database ... 64299 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_1.26.0-00_amd64.deb ...
Unpacking cri-tools (1.26.0-00) ...
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_1.2.0-00_amd64.deb ...
Unpacking kubernetes-cni (1.2.0-00) ...
Selecting previously unselected package socat.
Preparing to unpack .../4-socat_1.7.4.1-3ubuntu4_amd64.deb ...
Unpacking socat (1.7.4.1-3ubuntu4) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.28.2-00_amd64.deb ...
Unpacking kubelet (1.28.2-00) ...
Selecting previously unselected package kubectl.
```

```
Unpacking cri-tools (1.26.0-00) ...
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_1.2.0-00_amd64.deb ...
Unpacking kubernetes-cni (1.2.0-00) ...
Selecting previously unselected package socat.
Preparing to unpack .../4-socat_1.7.4.1-3ubuntu4_amd64.deb ...
Unpacking socat (1.7.4.1-3ubuntu4) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.28.2-00_amd64.deb ...
Unpacking kubelet (1.28.2-00) ...
Selecting previously unselected package kubectl.
Preparing to unpack .../6-kubectl_1.28.2-00_amd64.deb ...
Unpacking kubectl (1.28.2-00) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../7-kubeadm_1.28.2-00_amd64.deb ...
Unpacking kubeadm (1.28.2-00) ...
Setting up conntrack (1:1.4.6-2build2) ...
Setting up kubectl (1.28.2-00) ...
Setting up ebtables (2.0.11-4build2) ...
Setting up socat (1.7.4.1-3ubuntu4) ...
Setting up cri-tools (1.26.0-00) ...
Setting up kubernetes-cni (1.2.0-00) ...
Setting up kubelet (1.28.2-00) ...
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /lib/systemd/system/kubelet.service.
Setting up kubeadm (1.28.2-00) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-1-0-0-73:~#
```

sudo apt-mark hold kubelet kubeadm kubectl

```
root@ip-1-0-0-73:~# sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@ip-1-0-0-73:~#
```

ls

```
root@ip-1-0-0-73:~# ls
kubeadm-scripts  snap
root@ip-1-0-0-73:~# cd kubeadm-scripts/
root@ip-1-0-0-73:~/kubeadm-scripts# ls
README.md  Vagrantfile  manifests  scripts  terraform
root@ip-1-0-0-73:~/kubeadm-scripts# cd scripts/
root@ip-1-0-0-73:~/kubeadm-scripts/scripts# ls
common.sh  master.sh
```

Run the ./common.sh file (kubeadm-scripts/scripts) in this location on both nodes:

```
sudo ./common.sh
```

```
root@ip-1-0-0-73:~/kubeadm-scripts# ls
README.md  Vagrantfile  manifests  scripts  terraform
root@ip-1-0-0-73:~/kubeadm-scripts# cd scripts/
root@ip-1-0-0-73:~/kubeadm-scripts/scripts# ls
common.sh  master.sh
root@ip-1-0-0-73:~/kubeadm-scripts/scripts# sudo ./common.sh
+ KUBERNETES_VERSION=1.28.1-00
+ sudo swapoff -a
+ crontab -l
+ crontab -
+ echo '@reboot /sbin/swapoff -a'
+ sudo apt-get update -y
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages
[984 kB]
Hit:6 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Fetched 1212 kB in 1s (1200 kB/s)
```

```
* Applying /etc/sysctl.d/k8s.conf ...
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
* Applying /etc/sysctl.conf ...
+ cat
+ sudo tee /etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list
deb https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/xUbuntu_2
2.04/ /
+ cat
+ sudo tee /etc/apt/sources.list.d/devel:kubic:libcontainers:stable:cri-o:1.28.list
deb http://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable:/cri-o:/1.
28/xUbuntu_22.04/ /
+ curl -L https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable:cri-o:
1.28/xUbuntu_22.04/Release.key
+ sudo apt-key --keyring /etc/apt/trusted.gpg.d/libcontainers.gpg add -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0Warning: apt-k
ey is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
100   393  100   393    0     0    596      0 --:--:-- --:--:-- --:--:--   596
100   394  100   394    0     0    469      0 --:--:-- --:--:-- --:--:--   469
100   395  100   395    0     0    388      0  0:00:01  0:00:01 --:--:--   388
100   396  100   396    0     0    329      0  0:00:01  0:00:01 --:--:--     0
100   397  100   397    0     0    288      0  0:00:01  0:00:01 --:--:--   288
100  1093  100  1093    0     0    698      0  0:00:01  0:00:01 --:--:--   698
OK
+ curl -L https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/xUb
untu_22.04/Release.key
+ sudo apt-key --keyring /etc/apt/trusted.gpg.d/libcontainers.gpg add -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0Warning: apt-k
ey is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
100  1093  100  1093    0     0   1691      0 --:--:-- --:--:-- --:--:--  1691
OK
+ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
```

lastly you see like this

```
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
+ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://dl.k8s.io/apt/doc/apt-key.gpg
+ echo 'deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main'
+ sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main
+ sudo apt-get update -y
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:6 http://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable:/cri-o:/1.28/xUbuntu_22.04  InRelease [1632 B]
Hit:4 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Get:7 https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/xUbuntu_22.04  InRelease [1639 B]
Fetched 3271 B in 1s (3372 B/s)
Reading package lists... Done
+ sudo apt-get install -y kubelet=1.28.1-00 kubectl=1.28.1-00 kubeadm=1.28.1-00
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following held packages will be changed:
  kubeadm kubectl kubelet
The following packages will be DOWNGRADED:
  kubeadm kubectl kubelet
0 upgraded, 0 newly installed, 3 downgraded, 0 to remove and 124 not upgraded.
E: Packages were downgraded and -y was used without --allow-downgrades.
root@ip-1-0-0-73:~/kubeadm-scripts/scripts#
```

Now you need to change master.sh file

```
sudo nano master.sh

PUBLIC_IP_ACCESS="false"

False replace with true

PUBLIC_IP_ACCESS="true"

By default I would be PUBLIC_IP_ACCESS="true" only but once we need verify that's it
```

After using sudo ./master.sh the master / control plane will generate the token as mentioned below

the same token has to use in every node join with master then the communication will be establish in between control plane and cluster

```
GNU nano 6.2                                master.sh

set -euxo pipefail

# If you need public access to API server using the servers Public IP adress, change PUBLIC>

PUBLIC_IP_ACCESS="true"
NODENAME=$(hostname -s)
POD_CIDR="192.168.0.0/16"

# Pull required images

sudo kubeadm config images pull

# Initialize kubeadm based on PUBLIC_IP_ACCESS

if [[ "$PUBLIC_IP_ACCESS" == "false" ]]; then

    MASTER_PRIVATE_IP=$(ip addr show eth0 | awk '/inet / {print $2}' | cut -d/ -f1)
    sudo kubeadm init --apiserver-advertise-address="$MASTER_PRIVATE_IP" --apiserver-cert-e>

elif [[ "$PUBLIC_IP_ACCESS" == "true" ]]; then

    MASTER_PUBLIC_IP=$(curl ifconfig.me && echo "")
    sudo kubeadm init --control-plane-endpoint="$MASTER_PUBLIC_IP" --apiserver-cert-extra-s>

else
    echo "Error: MASTER_PUBLIC_IP has an invalid value: $PUBLIC_IP_ACCESS"
    exit 1
fi

# Configure kubeconfig

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

# Install Claico Network Plugin Network

^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line
```

Now run the master.sh file

```
sudo ./master.sh
```

```
root@ip-1-0-0-73:~/kubeadm-scripts/scripts# sudo ./master.sh
+ PUBLIC_IP_ACCESS=true
++ hostname -s
+ NODENAME=ip-1-0-0-73
+ POD_CIDR=192.168.0.0/16
+ sudo kubeadm config images pull
```

```
+ POD_CIDR=192.168.0.0/16

+ sudo kubeadm config images pull

[config/images] Pulled registry.k8s.io/kube-apiserver:v1.28.2

[config/images] Pulled registry.k8s.io/kube-controller-manager:v1.28.2

[config/images] Pulled registry.k8s.io/kube-scheduler:v1.28.2

[config/images] Pulled registry.k8s.io/kube-proxy:v1.28.2

[config/images] Pulled registry.k8s.io/pause:3.9

[config/images] Pulled registry.k8s.io/etcd:3.5.9-0

[config/images] Pulled registry.k8s.io/coredns/coredns:v1.10.1

+ [[ true == \f\a\l\s\e ]]

+ [[ true == \t\r\u\e ]]

++ curl ifconfig.me

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current

                                 Dload  Upload   Total   Spent    Left  Speed
```

```
100    13  100    13    0     0     48      0 --:--:-- --:--:-- --:--:--    48

++ echo ''

+ MASTER_PUBLIC_IP=43.205.242.73

+ sudo kubeadm init --control-plane-endpoint=43.205.242.73 --apiserver-cert-extra-sans=43.205.242.73 --pod-network-cidr=192.168.0.0/16 --node-name ip-1-0-0-73 --ignore-preflight-errors Swap

[init] Using Kubernetes version: v1.28.2

[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 [ip-1-0-0-73 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 1.0.0.73 43.205.242.73]

[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 [ip-1-0-0-73 localhost] and IPs [1.0.0.73
127.0.0.1 ::1]

[certs] Generating "etcd/peer" certificate and key

[certs] etcd/peer serving cert is signed for DNS names [ip-1-0-0-73 localhost] and IPs [1.0.0.73
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 "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"

[kubelet-start] Starting the kubelet

[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s

[apiclient] All control plane components are healthy after 6.506746 seconds

[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace

[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster

[upload-certs] Skipping phase. Please see --upload-certs

[mark-control-plane] Marking the node ip-1-0-0-73 as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]
```

```
[mark-control-plane] Marking the node ip-1-0-0-73 as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]

[bootstrap-token] Using token: k7pcqe.rw7k3dik9mifkm4x

[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles

[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes

[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials

[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token

[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster

[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace

[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key

[addons] Applied essential addon: CoreDNS

[addons] Applied essential addon: kube-proxy



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 43.205.242.73:6443 --token k7pcqe.rw7k3dik9mifkm4x \
        --discovery-token-ca-cert-hash
sha256:f42bbb0341f5717ce53dc2a12ee753ec15d2bd02c80462bfa29187baa8394750 \
        --control-plane
```

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

```
kubeadm join 43.205.242.73:6443 --token k7pcqe.rw7k3dik9mifkm4x \
        --discovery-token-ca-cert-hash
sha256:f42bbb0341f5717ce53dc2a12ee753ec15d2bd02c80462bfa29187baa8394750
+ mkdir -p /root/.kube
+ sudo cp -i /etc/kubernetes/admin.conf /root/.kube/config
++ id -u
++ id -g
+ sudo chown 0:0 /root/.kube/config
+ kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/tigera-operator.yaml
namespace/tigera-operator created
```

```
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.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

customresourcedefinition.apiextensions.k8s.io/apiservers.operator.tigera.io created

customresourcedefinition.apiextensions.k8s.io/imagesets.operator.tigera.io created

customresourcedefinition.apiextensions.k8s.io/installations.operator.tigera.io created

customresourcedefinition.apiextensions.k8s.io/tigerastatuses.operator.tigera.io created

serviceaccount/tigera-operator created

clusterrole.rbac.authorization.k8s.io/tigera-operator created

clusterrolebinding.rbac.authorization.k8s.io/tigera-operator created

deployment.apps/tigera-operator created

+ curl https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/custom-
resources.yaml -O

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current

                                 Dload  Upload   Total   Spent    Left  Speed

100   824  100   824    0     0   1943      0 --:--:-- --:--:-- --:--:--  1947

+ kubectl create -f custom-resources.yaml

installation.operator.tigera.io/default created
```

```
apiserver.operator.tigera.io/default created

root@ip-1-0-0-73:~/kubeadm-scripts/scripts#
```

Use the following commands from the output to create the kubeconfig in master so that you can use kubectl to interact with cluster API

```
mkdir -p $HOME/.kube

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

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

```
2. 43.205.242.73
lient certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

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 43.205.242.73:6443 --token k7pcqe.rw7k3dik9mifkm4x \
        --discovery-token-ca-cert-hash sha256:f42bbb0341f5717ce53dc2a12ee753ec15d2bd02c80462
bfa29187baa8394750 \
        --control-plane

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

kubeadm join 43.205.242.73:6443 --token k7pcqe.rw7k3dik9mifkm4x \
        --discovery-token-ca-cert-hash sha256:f42bbb0341f5717ce53dc2a12ee753ec15d2bd02c80462
bfa29187baa8394750
+ mkdir -p /root/.kube
+ sudo cp -i /etc/kubernetes/admin.conf /root/.kube/config
++ id -u
++ id -g
+ sudo chown 0:0 /root/.kube/config
+ kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests
/tigera-operator.yaml
namespace/tigera-operator created
```

```
root@ip-1-0-0-73:~# ls
kubeadm-scripts  snap
root@ip-1-0-0-73:~# cd kubeadm-scripts
root@ip-1-0-0-73:~/kubeadm-scripts# ls
README.md  Vagrantfile  manifests  scripts  terraform
root@ip-1-0-0-73:~/kubeadm-scripts# cd scripts/
root@ip-1-0-0-73:~/kubeadm-scripts/scripts# ls
common.sh  custom-resources.yaml  master.sh
root@ip-1-0-0-73:~/kubeadm-scripts/scripts# mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
cp: overwrite '/root/.kube/config'? ^C
root@ip-1-0-0-73:~/kubeadm-scripts/scripts# ls
common.sh  custom-resources.yaml  master.sh
root@ip-1-0-0-73:~/kubeadm-scripts/scripts# mkdir -p $HOME/.kube
root@ip-1-0-0-73:~/kubeadm-scripts/scripts# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kub
e/config
cp: overwrite '/root/.kube/config'?
root@ip-1-0-0-73:~/kubeadm-scripts/scripts# sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@ip-1-0-0-73:~/kubeadm-scripts/scripts#
```

Master node setup completed

Referred by :

https://www.fosstechnix.com/kubernetes-cluster-using-kubeadm-on-ubuntu-22/

Kubernetes Worker node setup:

Prerequisites:

- 2 or 3 Ubuntu 20.04 LTS System with Minimal Installation
- Minimum 2 or more CPU, 3 GB RAM.
- Disable SWAP on All node
- SSH Access with sudo privileges

Table of Contents

Firewall Ports/Inbound Traffic Ports for Kubernetes Cluster

Worker node(s) Ports

S.No	Protocol	Direction	Port Range	Purpose	Used By
1	TCP	Inbound	10250	Kubelet API	Self, Control plane
2	TCP	Inbound	30000-32767	NodePort Services	All

You can clone the repository for reference.

git clone https://github.com/techiescamp/kubeadm-scripts

```
root@ip-1-0-0-243:/home/ubuntu# git clone https://github.com/techiescamp/kubeadm-scripts
Cloning into 'kubeadm-scripts'...
remote: Enumerating objects: 286, done.
remote: Counting objects: 100% (120/120), done.
remote: Compressing objects: 100% (63/63), done.
remote: Total 286 (delta 76), reused 81 (delta 52), pack-reused 166
Receiving objects: 100% (286/286), 85.74 KiB | 7.79 MiB/s, done.
Resolving deltas: 100% (110/110), done.
```

Step #1:IPtables to see bridged traffic

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
```

```
root@ip-1-0-0-243:/home/ubuntu# cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
overlay
br_netfilter
root@ip-1-0-0-243:/home/ubuntu#
```

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

```
root@ip-1-0-0-243:/home/ubuntu# sudo modprobe overlay
root@ip-1-0-0-243:/home/ubuntu# sudo modprobe br_netfilter
root@ip-1-0-0-243:/home/ubuntu#
```

Apply sysctl params without reboot sudo sysctl –system

sudo sysctl --system

```
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 ...
net.ipv6.conf.all.use_tempaddr = 0
net.ipv6.conf.default.use_tempaddr = 0
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
fs.protected_fifos = 1
fs.protected_hardlinks = 1
fs.protected_regular = 2
fs.protected_symlinks = 1
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.conf ...
root@ip-1-0-0-243:/home/ubuntu#
```

Step #2:Disable swap on all the Nodes

For kubeadm to work properly, you need to disable swap on all the nodes using the following command.

```
sudo swapoff -a

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

```
root@ip-1-0-0-243:/home/ubuntu# sudo swapof
f -a
(crontab -l 2>/dev/null; echo "@reboot /sbi
n/swapoff -a") | crontab - || true
root@ip-1-0-0-243:/home/ubuntu#
```

Step #3:Install CRI-O Runtime On All The Nodes

Create the .conf file to load the modules at bootup

```
cat <<EOF | sudo tee /etc/modules-load.d/crio.conf

overlay

br_netfilter

EOF
```

```
root@ip-1-0-0-243:/home/ubuntu# cat <<EOF | sudo tee /etc/modules-load.d/crio.conf
overlay
br_netfilter
EOF
overlay
br_netfilter
root@ip-1-0-0-243:/home/ubuntu#
```

```
# Set up required sysctl params, these persist across reboots.

cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip_forward = 1

net.bridge.bridge-nf-call-ip6tables = 1

EOF
```

```
root@ip-1-0-0-243:/home/ubuntu# cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
root@ip-1-0-0-243:/home/ubuntu#
```

Execute the following commands to enable overlayFS & VxLan pod communication.

```
sudo modprobe overlay


sudo modprobe br_netfilter
```

```
root@ip-1-0-0-243:/home/ubuntu# sudo modprobe overlay
root@ip-1-0-0-243:/home/ubuntu# sudo modprobe br_netfilter
root@ip-1-0-0-243:/home/ubuntu#
```

Set up required sysctl params, these persist across reboots.

```
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf

net.bridge.bridge-nf-call-iptables  = 1

net.ipv4.ip_forward                 = 1

net.bridge.bridge-nf-call-ip6tables = 1


EOF
```

```
root@ip-1-0-0-243:/home/ubuntu# cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
root@ip-1-0-0-243:/home/ubuntu#
```

Reload the parameters.

```
sudo sysctl --system
```

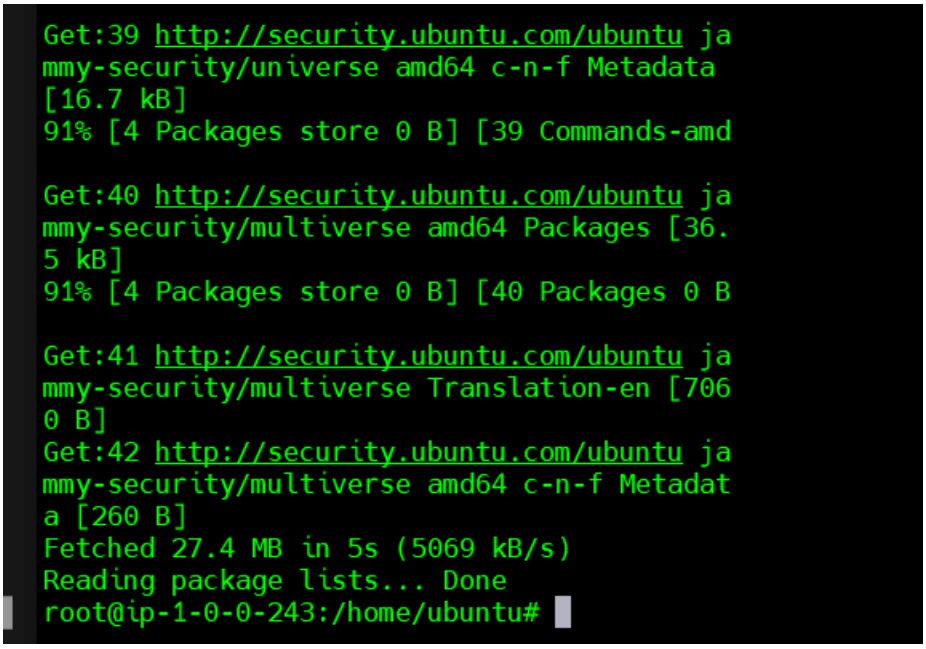
```
fs.protected_symlinks = 1
fs.protected_regular = 1
fs.protected_fifos = 1
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
kernel.pid_max = 4194304
* Applying /etc/sysctl.d/99-cloudimg-ipv6.conf ...
net.ipv6.conf.all.use_tempaddr = 0
net.ipv6.conf.default.use_tempaddr = 0
* Applying /etc/sysctl.d/99-kubernetes-cri.conf ...
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
fs.protected_fifos = 1
fs.protected_hardlinks = 1
fs.protected_regular = 2
fs.protected_symlinks = 1
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.conf ...
root@ip-1-0-0-243:/home/ubuntu#
```

Step #4:Install Kubeadm & Kubelet & Kubectl on all Nodes

Install the required dependencies

Update your system packages:

```
sudo apt-get update
```



```
Get:39 http://security.ubuntu.com/ubuntu ja
mmy-security/universe amd64 c-n-f Metadata
[16.7 kB]
91% [4 Packages store 0 B] [39 Commands-amd

Get:40 http://security.ubuntu.com/ubuntu ja
mmy-security/multiverse amd64 Packages [36.
5 kB]
91% [4 Packages store 0 B] [40 Packages 0 B

Get:41 http://security.ubuntu.com/ubuntu ja
mmy-security/multiverse Translation-en [706
0 B]
Get:42 http://security.ubuntu.com/ubuntu ja
mmy-security/multiverse amd64 c-n-f Metadat
a [260 B]
Fetched 27.4 MB in 5s (5069 kB/s)
Reading package lists... Done
root@ip-1-0-0-243:/home/ubuntu#
```

Install apt-transport-https curl

```
sudo apt-get install -y apt-transport-https curl
```

```
Preparing to unpack .../libcurl4_7.81.0-1ubuntu1.13_amd64.deb ...
Unpacking libcurl4:amd64 (7.81.0-1ubuntu1.13) over (7.81.0-1ubuntu1.10) ...
Setting up apt-transport-https (2.4.10) ...
Setting up libcurl4:amd64 (7.81.0-1ubuntu1.13) ...
Setting up curl (7.81.0-1ubuntu1.13) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-1-0-0-243:/home/ubuntu#
```

Add gpg keys

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

```
root@ip-1-0-0-243:/home/ubuntu# curl -s https://packages.cloud.google.com/apt/doc/apt-key.
gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key
(8)).
OK
root@ip-1-0-0-243:/home/ubuntu#
```

sudo vi /etc/apt/sources.list.d/kubernetes.list

```
sudo vi /etc/apt/sources.list.d/kubernetes.list
~
~
```

Add this below lines in this file

deb https://apt.kubernetes.io/ kubernetes-xenial main

Lets install kubelet kubeadm kubectl

sudo apt-get update

```
root@ip-1-0-0-243:/home/ubuntu# curl -s https://packages.cloud.google.com/apt/doc/apt-key.
gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key
(8)).
OK
root@ip-1-0-0-243:/home/ubuntu# sudo vi /etc/apt/sources.list.d/kubernetes.list
root@ip-1-0-0-243:/home/ubuntu# sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:4 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8993 B]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packages [69.9 kB
]
Fetched 306 kB in 1s (225 kB/s)
Reading package lists... Done
W: https://apt.kubernetes.io/dists/kubernetes-xenial/InRelease: Key is stored in legacy tr
usted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for de
tails.
root@ip-1-0-0-243:/home/ubuntu#
```

sudo apt-get install -y kubelet kubeadm kubectl

```
Setting up socat (1.7.4.1-3ubuntu4) ...
Setting up cri-tools (1.26.0-00) ...
Setting up kubernetes-cni (1.2.0-00) ...
Setting up kubelet (1.28.2-00) ...
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /lib/systemd
/system/kubelet.service.
Setting up kubeadm (1.28.2-00) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

sudo apt-mark hold kubelet kubeadm kubectl

```
root@ip-1-0-0-243:/home/ubuntu# sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@ip-1-0-0-243:/home/ubuntu#
```

Run the ./common.sh file (kubeadm-scripts/scripts) in this location on both nodes:

sudo ./common.sh

```
root@ip-1-0-0-243:/home/ubuntu/kubeadm-scripts/scripts# sudo ./common.sh
```

```
root@ip-1-0-0-243:/home/ubuntu/kubeadm-scripts/scripts# sudo ./common.sh
+ KUBERNETES_VERSION=1.28.1-00
+ sudo swapoff -a
+ crontab -l
+ crontab -
+ echo '@reboot /sbin/swapoff -a'
+ sudo apt-get update -y
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:4 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8993 B]
Fetched 8993 B in 1s (10.2 kB/s)
Reading package lists... Done
W: https://apt.kubernetes.io/dists/kubernetes-xenial/InRelease: Key is stored in legacy tr
usted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for de
tails.
+ OS=xUbuntu_22.04
+ VERSION=1.28
+ cat
```

```
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:6 http://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable:/cri-o
:/1.28/xUbuntu_22.04  InRelease [1632 B]
Hit:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Get:7 https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/xUbun
tu_22.04  InRelease [1639 B]
Fetched 3271 B in 1s (3425 B/s)
Reading package lists... Done
+ sudo apt-get install -y kubelet=1.28.1-00 kubectl=1.28.1-00 kubeadm=1.28.1-00
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following held packages will be changed:
  kubeadm kubectl kubelet
The following packages will be DOWNGRADED:
  kubeadm kubectl kubelet
0 upgraded, 0 newly installed, 3 downgraded, 0 to remove and 124 not upgraded.
E: Packages were downgraded and -y was used without --allow-downgrades.
root@ip-1-0-0-243:/home/ubuntu/kubeadm-scripts/scripts#
```

Now you need to change master.sh file

sudo nano master.sh

```
root@ip-1-0-0-243:/home/ubuntu/kubeadm-scripts/scripts# sudo nano master.sh
```

PUBLIC_IP_ACCESS="false"

False replace with true

PUBLIC_IP_ACCESS="true"

```
  GNU nano 6.2                                  master.sh
#!/bin/bash
#
# Setup for Control Plane (Master) servers

set -euxo pipefail

# If you need public access to API server using the servers Public IP adress, change PUBL>

PUBLIC_IP_ACCESS="true"
NODENAME=$(hostname -s)
POD_CIDR="192.168.0.0/16"

# Pull required images

sudo kubeadm config images pull

                                    [ Read 46 lines ]
^G Help      ^O Write Out   ^W Where Is    ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File   ^\ Replace     ^U Paste     ^J Justify   ^/ Go To Line
```

note : should not run master.sh file in worker nodes , it is for only master node

After setting up and installation check the nodes in Master as below mentioned command

kubectl get node or kubectl get all

```
root@ip-1-0-0-73:~/kubeadm-scripts# kubectl get node
NAME          STATUS   ROLES           AGE   VERSION
ip-1-0-0-73   Ready    control-plane   122m  v1.28.2
root@ip-1-0-0-73:~/kubeadm-scripts#
```

Join node into master by using token which generated by master

```
root@ip-10-0-2-137:~#
kubeadm join 52.91.228.50:6443 --token hgcrnr.jgffem9m32v3mkgt \
        --discovery-token-ca-cert-hash sha256:cd5dd4d1bf721dcf6210200c7bf26abca18f684b680871030397a62a6266248a
[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.
```

note : same token should use for all the nodes

```
kubectl get po -n kube-system
```

```
ubuntu@ip-10-0-2-85:~$ kubectl get po -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
coredns-5dd5756b68-drqzs                   1/1     Running   0          22m
coredns-5dd5756b68-pglk6                   1/1     Running   0          22m
etcd-ip-10-0-2-85                          1/1     Running   0          22m
kube-apiserver-ip-10-0-2-85                1/1     Running   0          22m
kube-controller-manager-ip-10-0-2-85       1/1     Running   0          22m
kube-proxy-nwpck                           1/1     Running   0          22m
kube-scheduler-ip-10-0-2-85                1/1     Running   0          22m
ubuntu@ip-10-0-2-85:~$
```

In master check the nodes status with below command

```
kubectl get nodes
```

```
ubuntu@ip-10-0-2-85:~$ kubectl get nodes
NAME            STATUS   ROLES           AGE     VERSION
ip-10-0-2-137   Ready    <none>          10m     v1.28.2
ip-10-0-2-85    Ready    control-plane   4h30m   v1.28.2
ubuntu@ip-10-0-2-85:~$
```