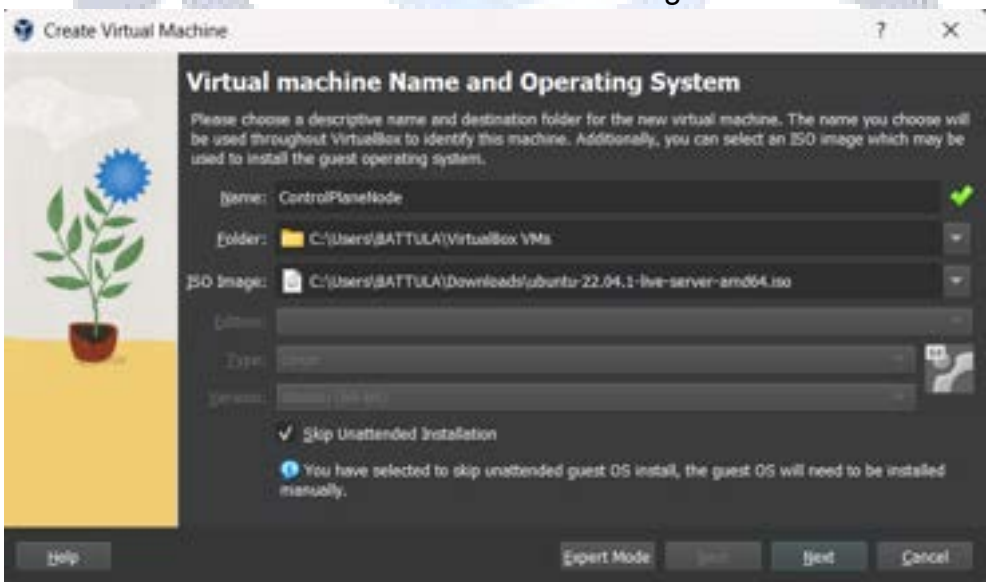
# Kubernetes multi-node setup using Kubeadm

Let us set up a kubernetes cluster on multi-node i.e., one node as control plane and another node as compute plane using the same CRI(containerd) and Kubeadm as we used in single-node setup.

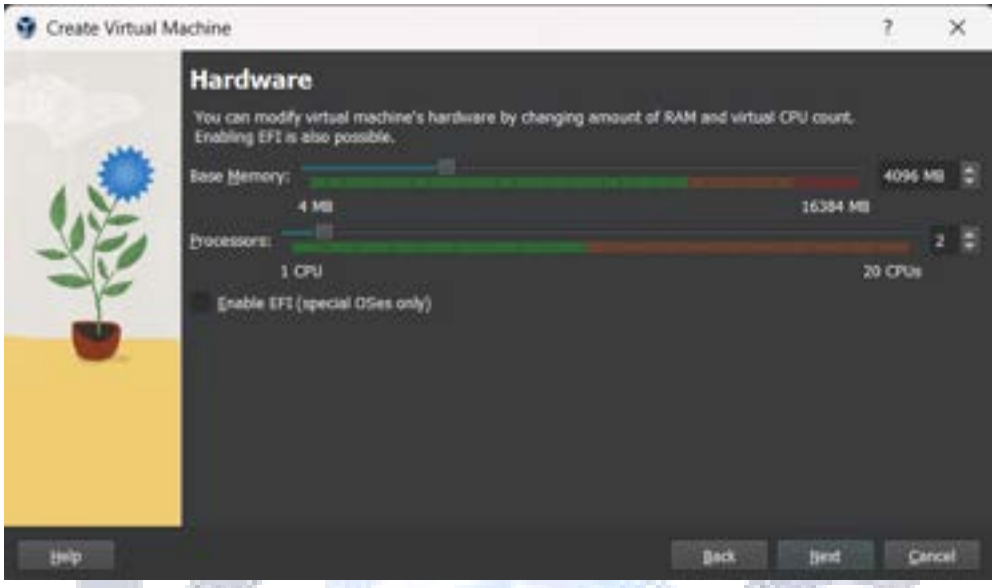
To set up a compute plane node, we can clone the existing control plane node and modify it or create one from scratch like we did in single-node setup and attach it to the control plane node.

Let us first setup a control plane node:

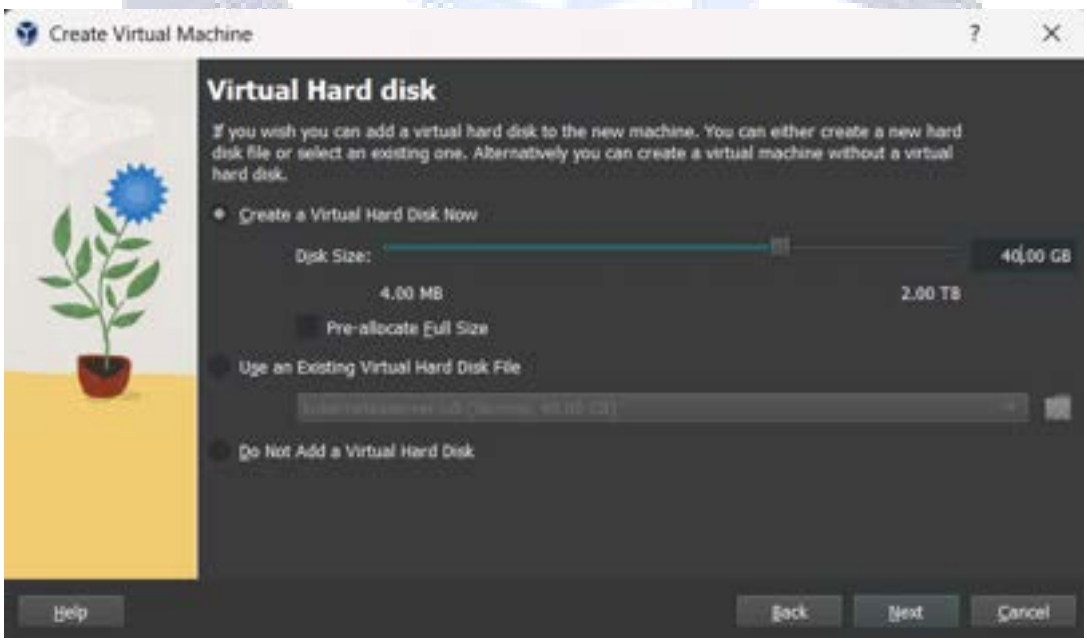
1. Let us first create a VM with ubuntu os as iso image.



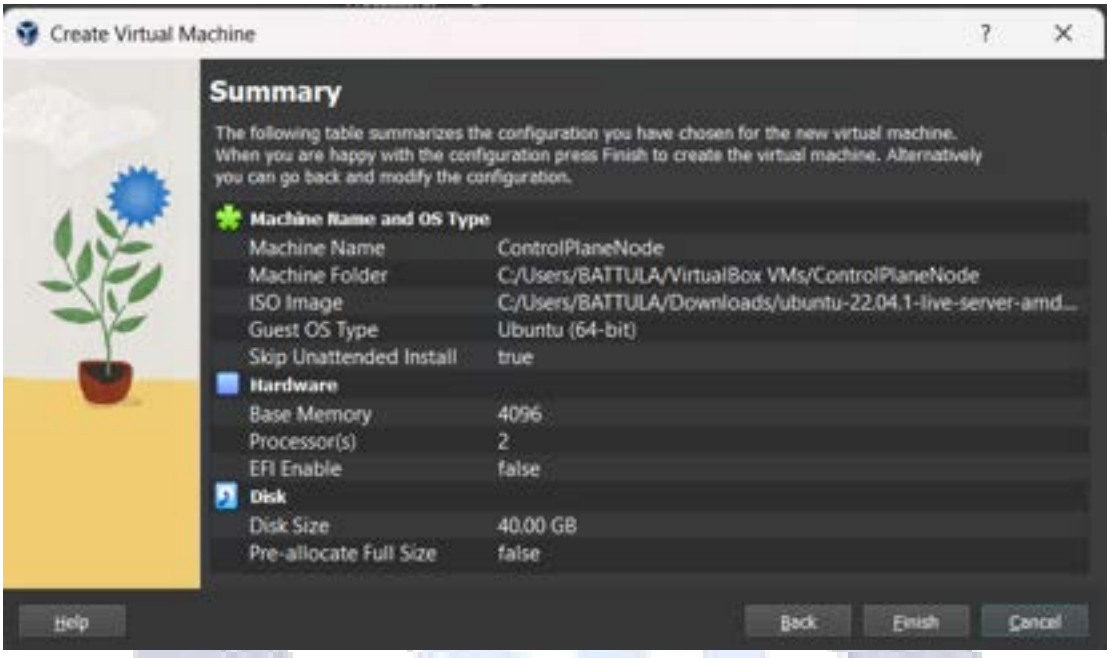

2. Now let us go through the hardware configuration that we are going to assign for the VM. We are using 4GB RAM and 2 cores of CPU.



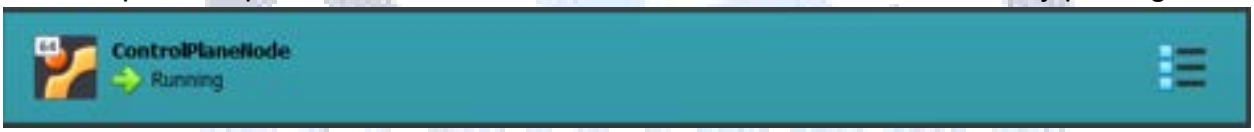

3. For storage, it is optimal to assign any convenient storage(40-50GB in our case) for our VM.



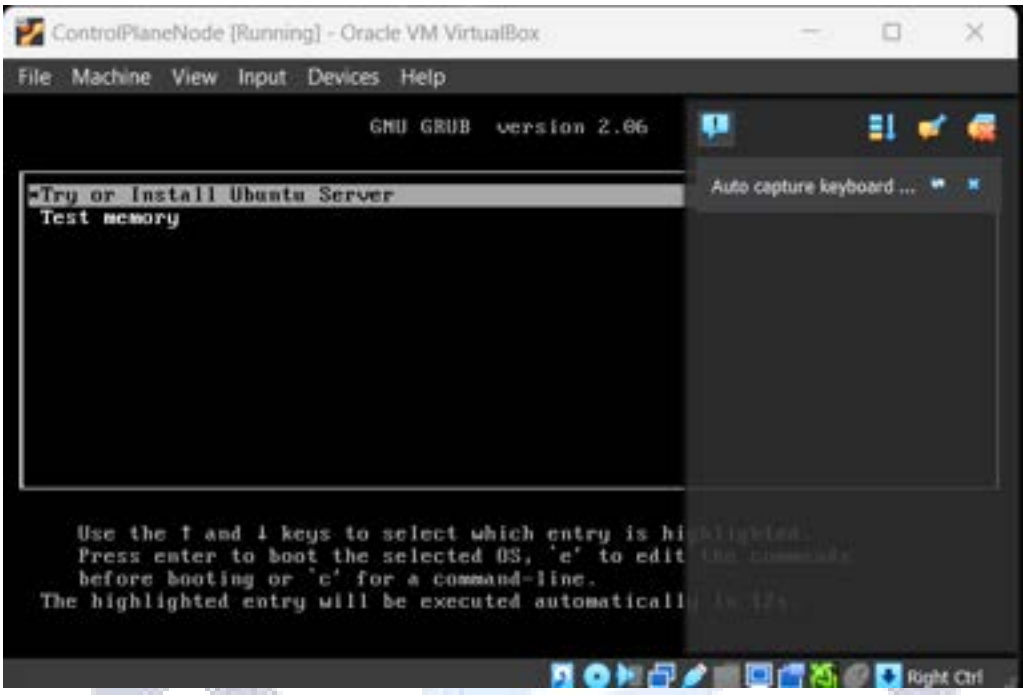

4. Here we can see the summary of the configurations we have selected.



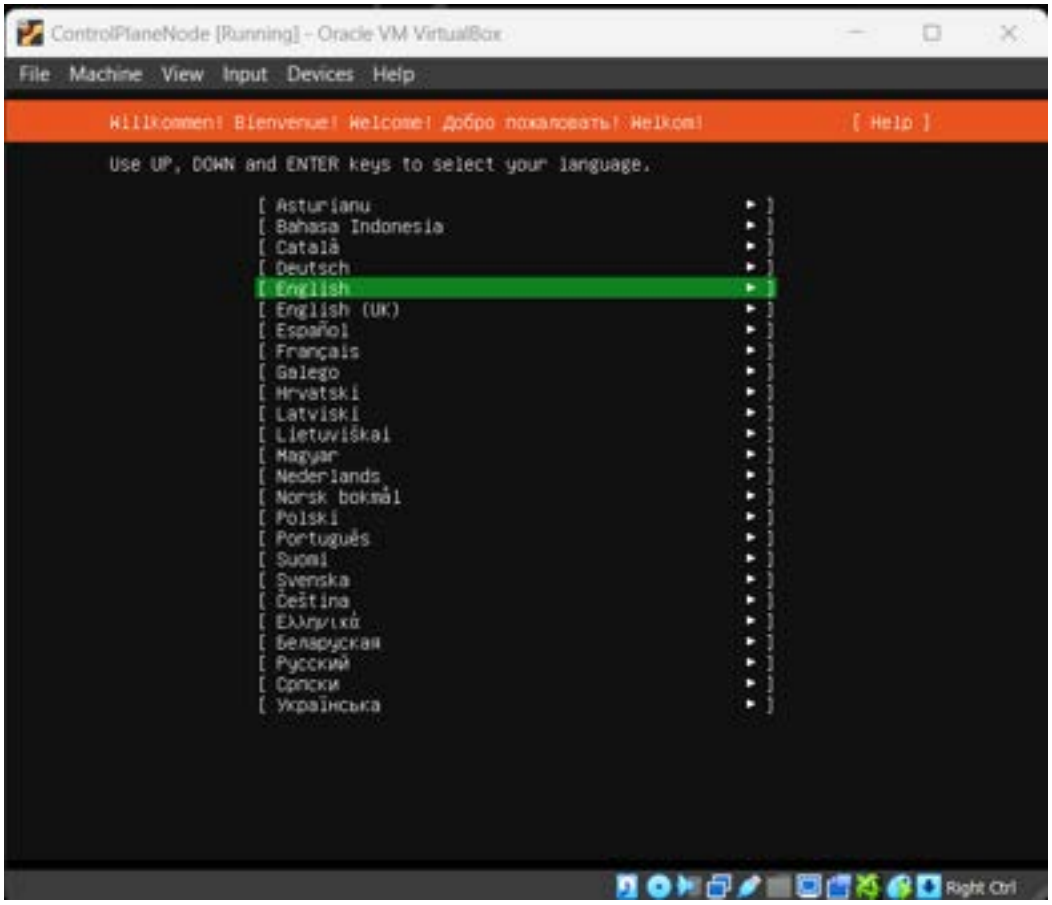

5. Upon completion, we shall start the VM and install all the necessary packages.



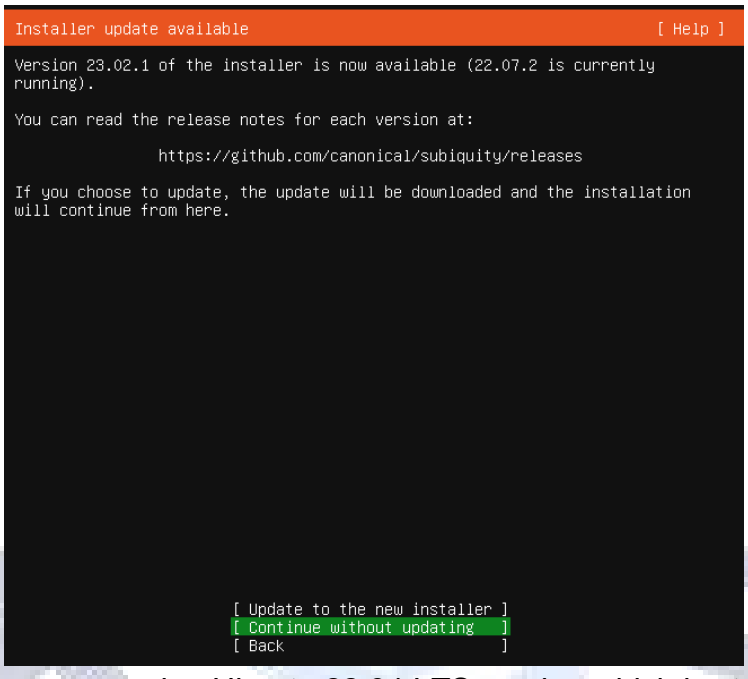

6. We can just proceed with this step.




7. After some time, it completes the boot up process and we can then start our installation process. We will now be prompted to select the language we wish to proceed with.



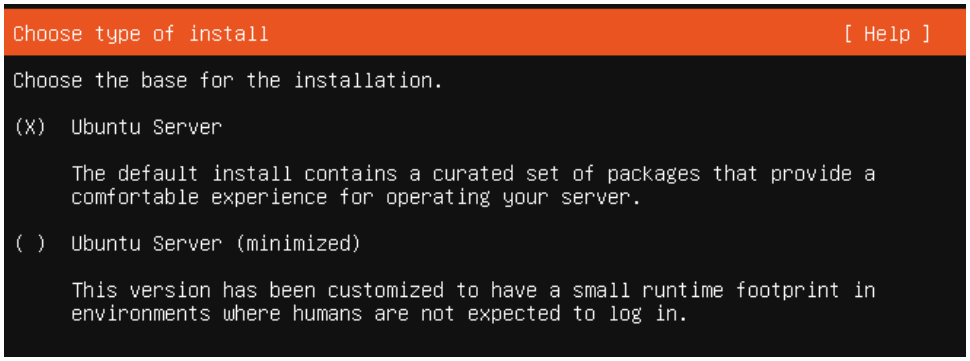

8. Whenever a new version of the iso image is available, the VM gives us an option to whether to continue with the existing version or update to the latest version.



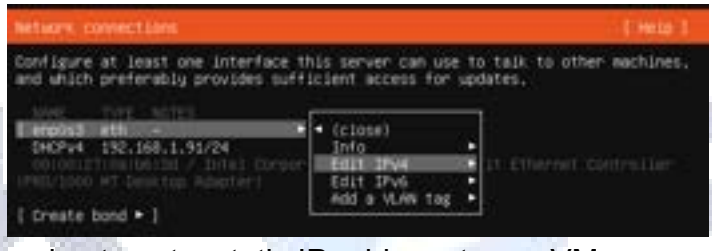

In our case, we are using Ubuntu 22.04 LTS version which is stable and bugs free. So, we are good to go with it.

9. Now we can see an option to select our keyboard language. Select the optimum language and continue to the next step.

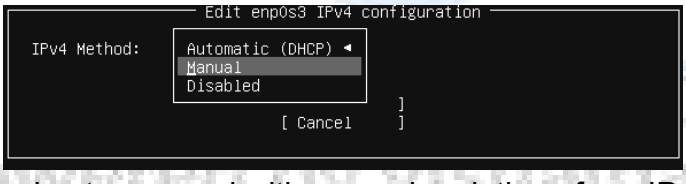

10. In this step, we are now asked whether we would like to install the default Ubuntu base packages or whether to install a minimized version of Ubuntu.



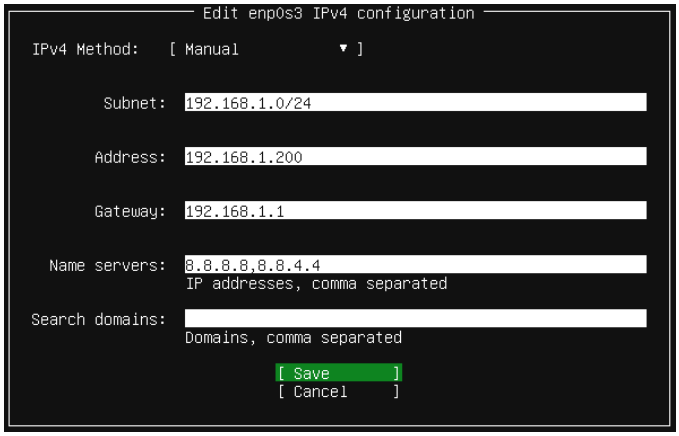

11. Next, we come across network settings where we can change our VM's network IP address.




Here we are going to set a static IP address to our VM as normally the VM's IP address might change every other time we boot up our VM.

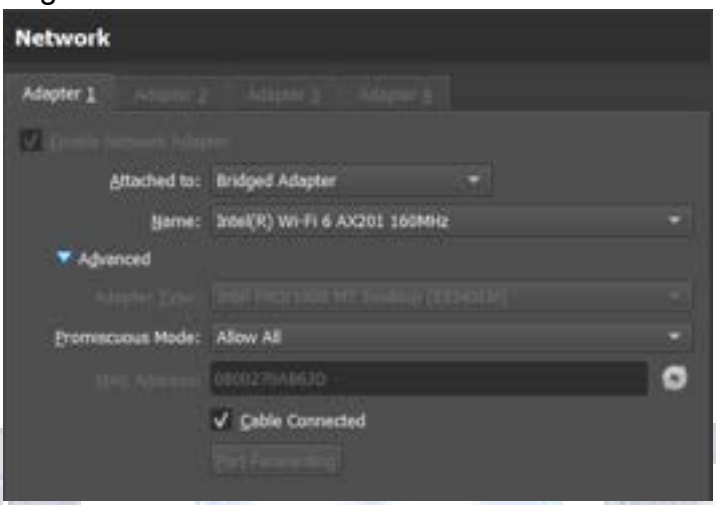



Here we are going to proceed with manual updation of our IP address.

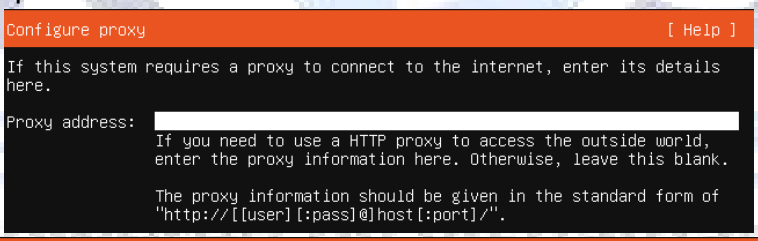


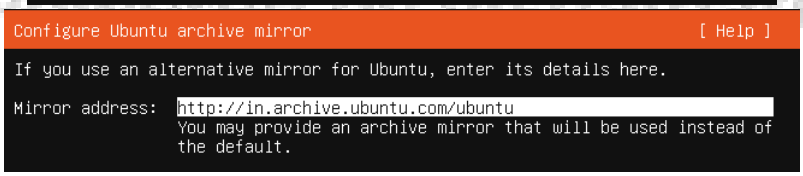

Here we are using google's default nameservers(8.8.8.8 and 8.8.4.4).

***Note***: Before that, we need to change the network type in VM settings from NAT to Bridged Adapter. This step can be performed right after assigning the hardware configurations to the VM.

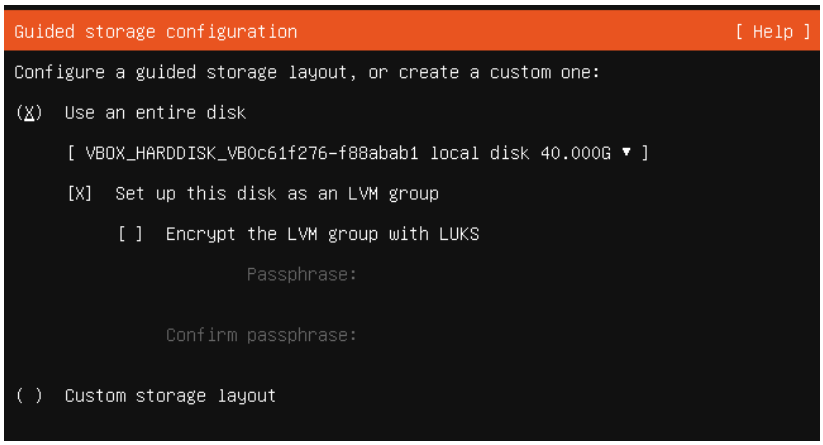



12. Configuring proxy and mirror address are not the things we are going to use, so we shall skip them.

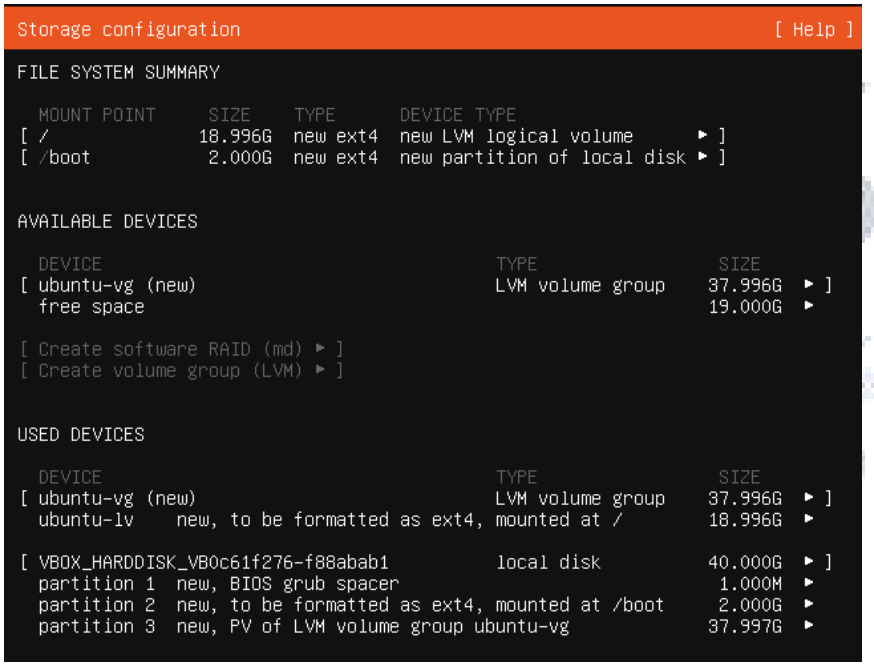


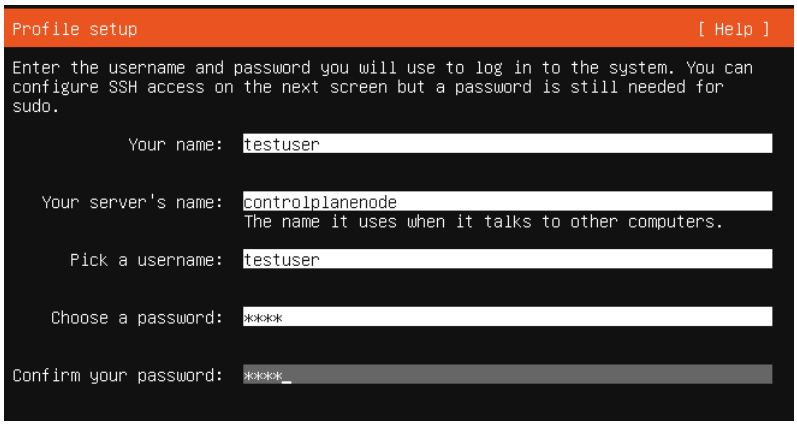

13. We can modify the storage type, but here we shall proceed with the default LVM partition by the VM.



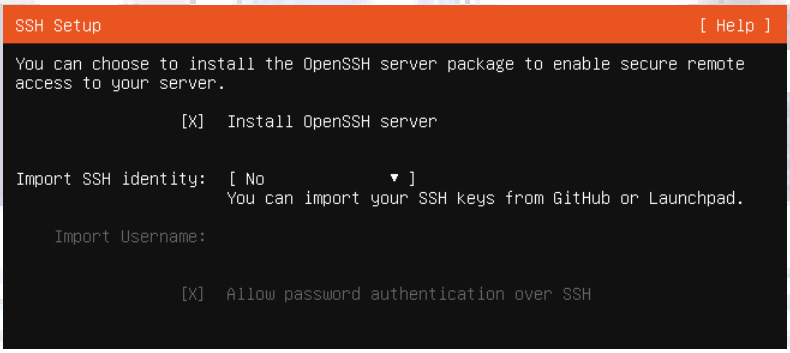

14. Here we can see the configuration details.



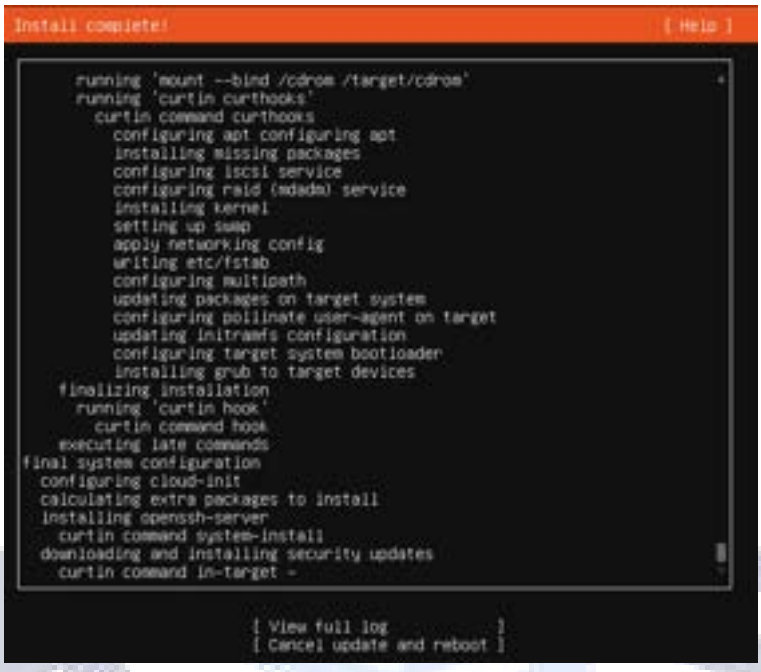

15. Now we can fill up the details for profile setup.



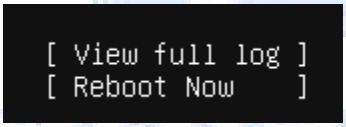

16. Now this is an important step as we can communicate with the VM from remote servers or from CMD, mobaXterm or git, etc,. So, we shall turn on the “Install OpenSSH server” option.



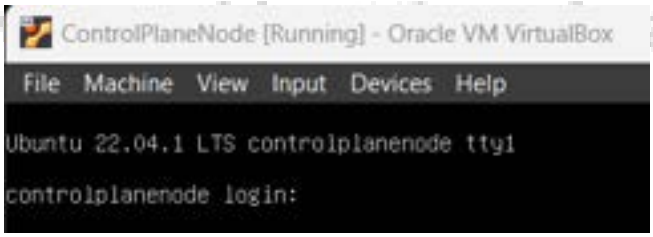

17. We can skip this step as we can install all these packages separately later based on our requirements.
18. Now the installation process starts and we can also see the log files if we want to.




19. After completion we can see this option, we proceed with the reboot now option.



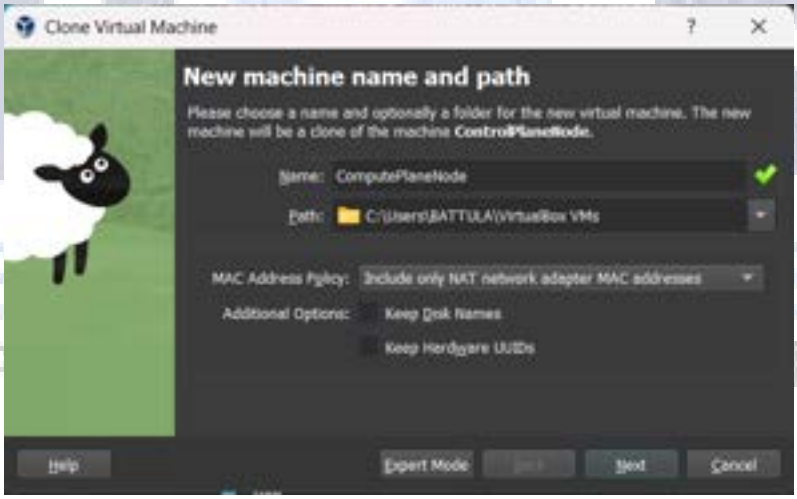

20. Upon successful installation, we can see that the VM is open successfully and it is completely functional.



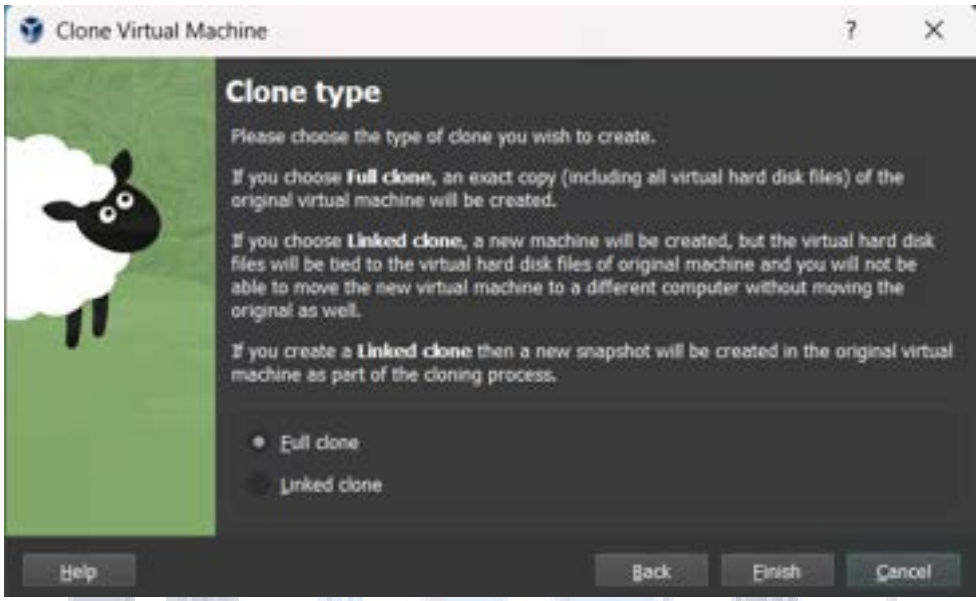

Now in-order to create a compute plane node, we can completely follow the above steps and create a new VM. Instead we can simply clone the VM we just created and modify it.

1. We can select the following clone option:

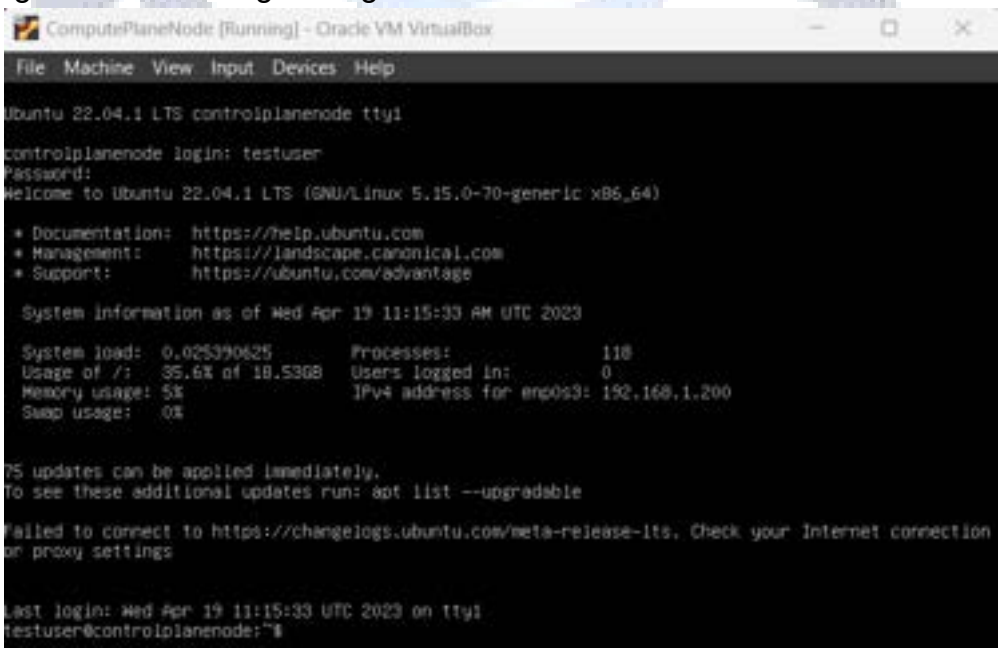

2. Now this pop-up appears, where we name our clone server.




3. Now, we go with the full clone option.




4. Now if we launch the clone VM, we can see that all the details are the same as the original VM including the login details.




5. There are two complexities we have over here, first of all we have to change the hostname of the VM and the other is to change the IP address.

```
root@controlplanenode:~# hostnamectl set-hostname computeplanenode
root@controlplanenode:~#
root@controlplanenode:~# _
```

In order to view the changes, we have to reauthenticate into the VM.

```
Ubuntu 22.04.1 LTS computeplanenode tty1

computeplanenode login: testuser
Password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-70-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Apr 19 11:20:08 AM UTC 2023

  System load:  0.03515625         Processes:               115
  Usage of /:   35.6% of 18.53GB   Users logged in:         0
  Memory usage: 5%                 IPv4 address for enp0s3: 192.168.1.200
  Swap usage:   0%


75 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection
or proxy settings


Last login: Wed Apr 19 11:16:10 UTC 2023 on tty1
testuser@computeplanenode:~$
```

As we can see, the changes have been reflected effectively.

6. We can see that the IP address is also the same as before. So, let us change the IP address and give a static IP address to the VM.

```
testuser@computeplanenode:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:ab:8e:1a brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.200/24 brd 192.168.1.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feab:8e1a/64 scope link
       valid_lft forever preferred_lft forever
testuser@computeplanenode:~$ _
```

7. In order to change the IP address and options, we have to modify the netplan config file.

```
  GNU nano 6.2                    /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp0s3:
      addresses:
      - 192.168.1.200/24
      gateway4: 192.168.1.1
      nameservers:
        addresses:
        - 8.8.8.8
        - 8.8.4.4
        search: []
  version: 2
```

8. The few changes that we are going to make here are just the addresses tab and instead of gateway4 variable, we are going to use routes variable, as gateway4 variable has been deprecated.

```
addresses:
- 192.168.1.201/24
routes:
  - to: default
    via: 192.168.1.1
```

9. Now, we apply the changes and then the changes will be reflected.

```
root@computeplanenode:~# netplan apply
root@computeplanenode:~#
root@computeplanenode:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:ab:8e:1a brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.201/24 brd 192.168.1.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feab:8e1a/64 scope link
       valid_lft forever preferred_lft forever
root@computeplanenode:~#
```

Now that both the VMs are up and ready, we can now install the common packages for both the VMs first and then custom install the packages based on the requirements.

As we are using the VMs for setting up a kubernetes cluster, we shall disable swap memory which is recommended to be done by kubernetes.

To disable swap memory, we have to modify the fstab file under /etc/fstab. Just comment out the last line in the file and save changes.

```
  GNU nano 6.2                                  /etc/fstab *
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/ubuntu-vg/ubuntu-lv during curtin installation
/dev/disk/by-id/dm-uuid-LVM-TWjHdOVTvA2MEMMU4QGlGU0jYnxVCk90yWOKz4kJqAoFbOf1T9rGetPjlzMosX4U / ext4
# /boot was on /dev/sda2 during curtin installation
/dev/disk/by-uuid/145bd5bf-3915-4c87-b008-f4dacf2de22b /boot ext4 defaults 0 1
#/swap.img      none    swap    sw      0       0
```

```
root@computeplanenode:~# free -h
               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       191Mi       3.4Gi       1.0Mi       217Mi       3.4Gi
Swap:          3.8Gi          0B       3.8Gi
```

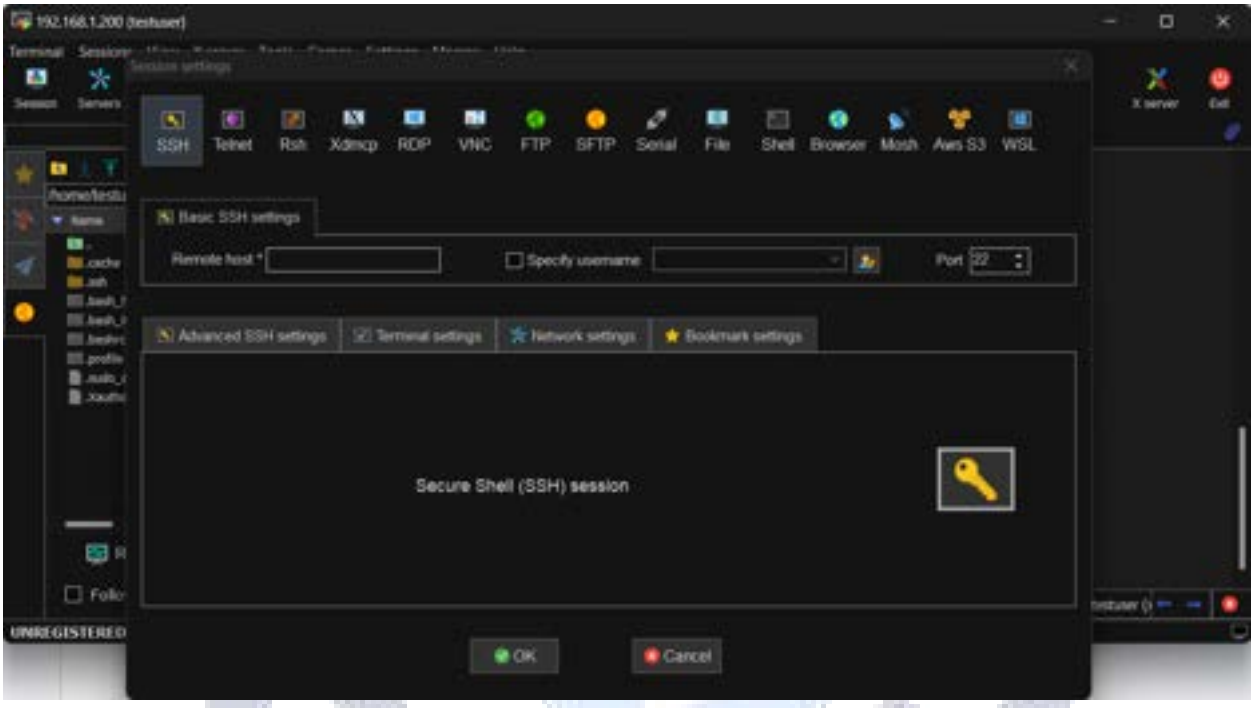
Now, we can use swapoff command to turn off swap memory.

```
root@computeplanenode:~# swapoff -a
root@computeplanenode:~#
root@computeplanenode:~# free -h
               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       190Mi       3.4Gi       1.0Mi       217Mi       3.4Gi
Swap:             0B          0B          0B
root@computeplanenode:~#
root@computeplanenode:~# _
```

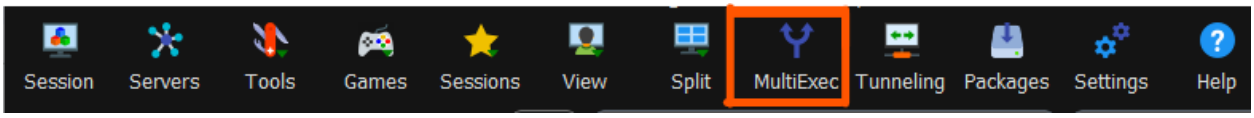
Now we can see that the swap memory is totally turned off. This is the same process for control plane node.

```
root@controlplanenode:~# free -h
               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       180Mi       3.4Gi       1.0Mi       211Mi       3.4Gi
Swap:             0B          0B          0B
root@controlplanenode:~#
```

Hereon, we can follow the same steps to install the CRI which is containerd in this case.
***Note***: We can use a tool called MobaXterm in which we have an option to multi-execute. Which means that we can make changes in multiple VMs at the same time.
We can open a session in MobaXterm by:

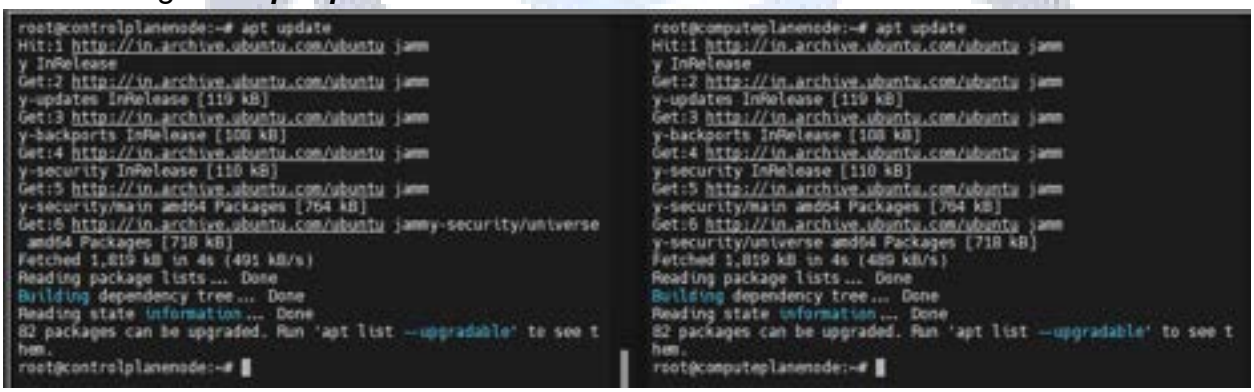



Now, upon opening both the nodes, we can use the MultiExec option which stands for multi execution.




In general, it looks like this:


1. As the VMs are newly created, as a rule of thumb we need to update the VMs using the ***"apt update"*** command.




In case there are any upgradable files, we can use the **"*apt upgrade*"** command.

Upon the completion of these updates, we can proceed to the further step, where we are going to install the gpg keys and required packages to set up the containerd CRI which is common for both the nodes.

2. Now on all nodes, we need to update the containerd config files, set up required sysctl params:

```
root@controlplanenode:~# cat > /etc/modules-load.d/containerd.con
f <<EOF
overlay
br_netfilter
EOF
root@controlplanenode:~# modprobe overlay
modprobe br_netfilter
root@controlplanenode:~#
root@controlplanenode:~# cat > /etc/sysctl.d/99-kubernetes-cri.co
nf <<EOF
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
root@controlplanenode:~#
```

```
root@computeplanenode:~# cat > /etc/modules-load.d/containerd.con
f <<EOF
overlay
br_netfilter
EOF
root@computeplanenode:~# modprobe overlay
modprobe br_netfilter
root@computeplanenode:~#
root@computeplanenode:~# cat > /etc/sysctl.d/99-kubernetes-cri.co
nf <<EOF
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
root@computeplanenode:~#
```

3. Now, upon the modification, we need to use the command ***"sysctl –system"***.

```
root@controlplanenode:~# 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": Inva
lid argument
net.ipv4.conf.default.promote_secondaries = 1
sysctl: setting key "net.ipv4.conf.all.promote_secondaries": Inva
lid 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
```

```
root@computeplanenode:~# 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": Inva
lid argument
net.ipv4.conf.default.promote_secondaries = 1
sysctl: setting key "net.ipv4.conf.all.promote_secondaries": Inva
lid 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
```

4. Now, after the sysctl parameters setup, we need to install the containerd package and requirements from docker official website.

```
root@controlplanenode:~# apt-get update && apt-get install -y apt
-transport-https ca-certificates curl software-properties-common
Hit:1 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Ign:2 https://download.docker.com/linux/debian jammy InRelease
Get:3 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
 [119 kB]
Err:4 https://download.docker.com/linux/debian jammy Release
  404  Not Found [IP: 18.155.49.90 443]
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelea
se [108 kB]
Hit:6 http://in.archive.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists ... Done
E: The repository 'https://download.docker.com/linux/debian jammy
 Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is
 therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user con
figuration details.
root@controlplanenode:~#
root@controlplanenode:~#
root@controlplanenode:~# curl -fsSL https://download.docker.com/l
inux/ubuntu/gpg | apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.g
pg.d instead (see apt-key(8)).
OK
root@controlplanenode:~#
root@controlplanenode:~#
root@controlplanenode:~# add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/u
```

```
root@computeplanenode:~# apt-get update && apt-get install -y apt
-transport-https ca-certificates curl software-properties-common
Ign:1 https://download.docker.com/linux/debian jammy InRelease
Err:2 https://download.docker.com/linux/debian jammy Release
  404  Not Found [IP: 18.155.49.114 443]
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
 [119 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelea
se [108 kB]
Hit:6 http://in.archive.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists ... Done
E: The repository 'https://download.docker.com/linux/debian jammy
 Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is
 therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user con
figuration details.
root@computeplanenode:~#
root@computeplanenode:~#
root@computeplanenode:~# curl -fsSL https://download.docker.com/l
inux/ubuntu/gpg | apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.g
pg.d instead (see apt-key(8)).
OK
root@computeplanenode:~#
root@computeplanenode:~#
root@computeplanenode:~# add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/u
```

5. Docker's official GPG key:

```
root@controlplanenode:~# curl -fsSL https://download.docker.com/l
inux/ubuntu/gpg | apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.g
pg.d instead (see apt-key(8)).
OK
root@controlplanenode:~#
root@controlplanenode:~#
```

```
root@computeplanenode:~# curl -fsSL https://download.docker.com/l
inux/ubuntu/gpg | apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.g
pg.d instead (see apt-key(8)).
OK
root@computeplanenode:~#
root@computeplanenode:~#
```

6. Adding docker's apt repository

```
root@controlplanenode:~# curl -fsSL https://download.docker.com/l
inux/ubuntu/gpg | apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.g
pg.d instead (see apt-key(8)).
OK
root@controlplanenode:~#
root@controlplanenode:~# add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/u
buntu jammy stable'
Description:
Archive for codename: jammy components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Found existing deb entry in /etc/apt/sources.list.d/archive_uri-h
ttps_download_docker_com_linux_ubuntu-jammy.list
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_dow
nload_docker_com_linux_ubuntu-jammy.list
Found existing deb-src entry in /etc/apt/sources.list.d/archive_u
ri-https_download_docker_com_linux_ubuntu-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_
uri-https_download_docker_com_linux_ubuntu-jammy.list
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [4
8.9 kB]
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Ign:3 https://download.docker.com/linux/debian jammy InRelease
Err:4 https://download.docker.com/linux/debian jammy Release
  404  Not Found [IP: 18.155.49.14 443]
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
 [119 kB]
```

```
root@computeplanenode:~# curl -fsSL https://download.docker.com/l
inux/ubuntu/gpg | apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.g
pg.d instead (see apt-key(8)).
OK
root@computeplanenode:~#
root@computeplanenode:~# add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/u
buntu jammy stable'
Description:
Archive for codename: jammy components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Found existing deb entry in /etc/apt/sources.list.d/archive_uri-h
ttps_download_docker_com_linux_ubuntu-jammy.list
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_dow
nload_docker_com_linux_ubuntu-jammy.list
Found existing deb-src entry in /etc/apt/sources.list.d/archive_u
ri-https_download_docker_com_linux_ubuntu-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_
uri-https_download_docker_com_linux_ubuntu-jammy.list
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [4
8.9 kB]
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Ign:3 https://download.docker.com/linux/debian jammy InRelease
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
 [119 kB]
Err:5 https://download.docker.com/linux/debian jammy Release
  404  Not Found [IP: 18.155.49.14 443]
```

7. Finally we can install the containerd.io package

```
root@controlplanenode:~# apt-get update && apt-get install -y con
tainerd.io
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Ign:2 https://download.docker.com/linux/debian jammy InRelease
Err:3 https://download.docker.com/linux/debian jammy Release
  404  Not Found [IP: 18.155.49.114 443]
Hit:4 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
 [119 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelea
se [109 kB]
Hit:7 http://in.archive.ubuntu.com/ubuntu jammy-security InReleas
e
Reading package lists ... Done
W: https://download.docker.com/linux/ubuntu/dists/jammy/InRelease
: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.g
pg), see the DEPRECATION section in apt-key(8) for details.
E: The repository 'https://download.docker.com/linux/debian jammy
 Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is
 therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user con
figuration details.
root@controlplanenode:~#
root@controlplanenode:~#
```

```
root@computeplanenode:~# apt-get update && apt-get install -y con
tainerd.io
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Ign:2 https://download.docker.com/linux/debian jammy InRelease
Err:3 https://download.docker.com/linux/debian jammy Release
  404  Not Found [IP: 18.155.49.90 443]
Hit:4 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
 [119 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelea
se [109 kB]
Hit:7 http://in.archive.ubuntu.com/ubuntu jammy-security InReleas
e
Reading package lists ... Done
W: https://download.docker.com/linux/ubuntu/dists/jammy/InRelease
: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.g
pg), see the DEPRECATION section in apt-key(8) for details.
E: The repository 'https://download.docker.com/linux/debian jammy
 Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is
 therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user con
figuration details.
root@computeplanenode:~#
root@computeplanenode:~#
```

8. Now we can configure containerd. First, we create a repository,

```
mkdir -p /etc/containerd
```

```
root@controlplanenode:~# containerd config default > /etc/contain
erd/config.toml
root@controlplanenode:~#
root@controlplanenode:~# sed -i 's/SystemdCgroup = false/SystemdC
group = true/g' /etc/containerd/config.toml
root@controlplanenode:~#
root@controlplanenode:~#
```

```
root@computeplanenode:~# containerd config default > /etc/contain
erd/config.toml
root@computeplanenode:~#
root@computeplanenode:~# sed -i 's/SystemdCgroup = false/SystemdC
group = true/g' /etc/containerd/config.toml
root@computeplanenode:~#
root@computeplanenode:~#
```

9. Now we restart containerd

```
root@controlplanenode:~# systemctl restart containerd
root@controlplanenode:~#
```

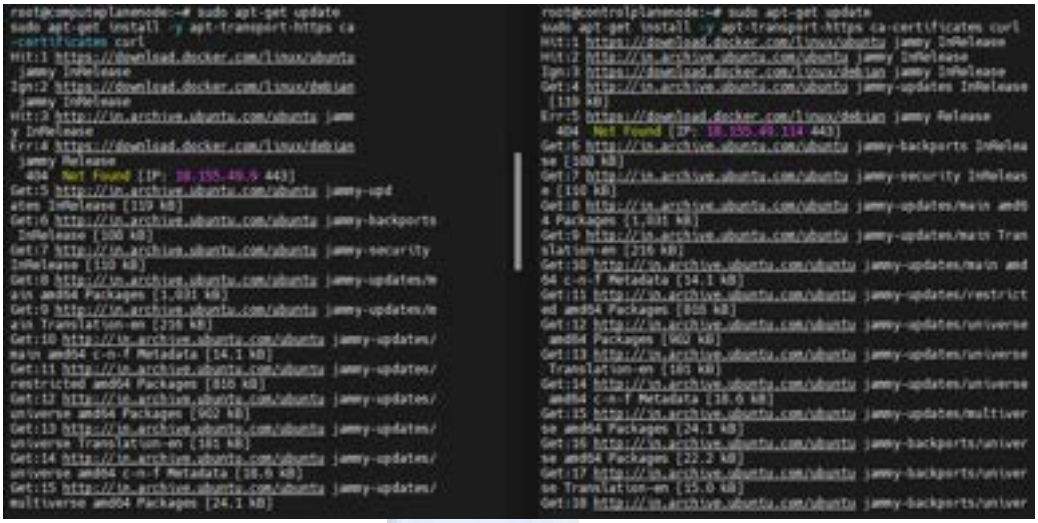
```
root@computeplanenode:~# systemctl restart containerd
root@computeplanenode:~#
```

10. To execute crictl CLI commands, ensure we create a configuration file as mentioned below.

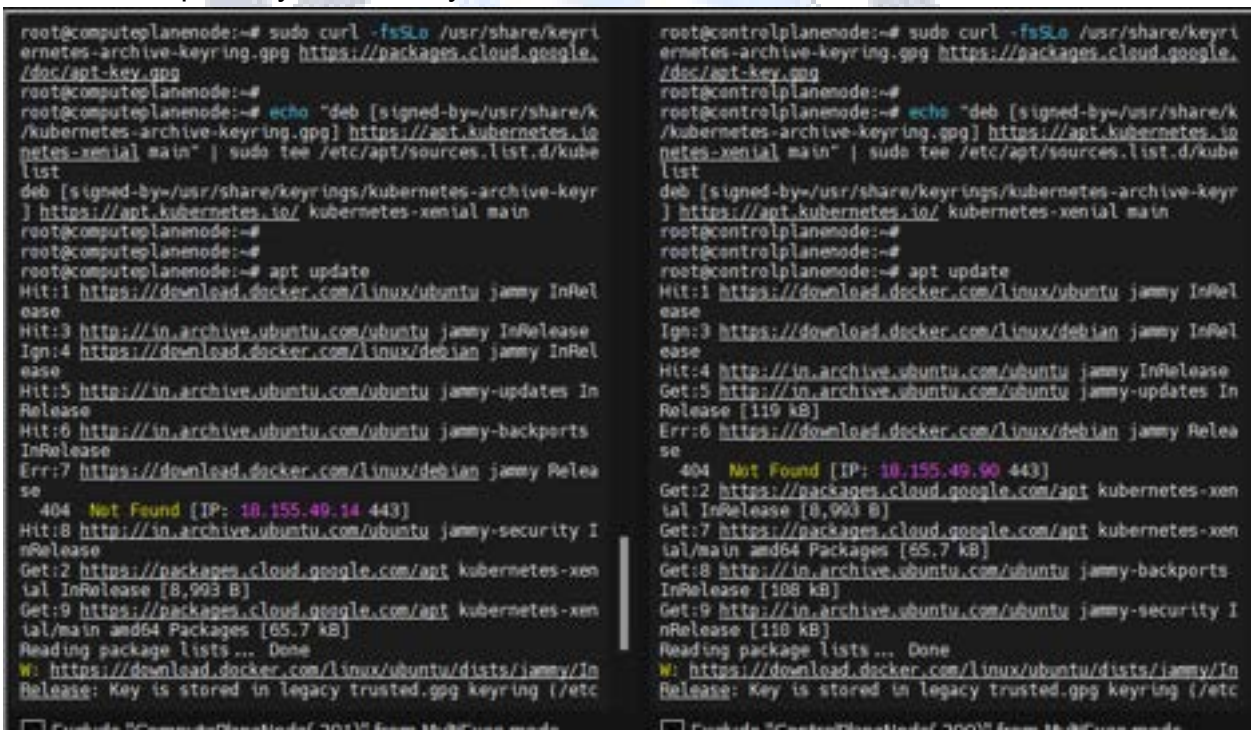
```
root@controlplanenode:~# cat > /etc/crictl.yaml <<EOF
runtime-endpoint: unix:///run/containerd/containerd.sock
image-endpoint: unix:///run/containerd/containerd.sock
timeout: 2
> EOF
root@controlplanenode:~#
root@controlplanenode:~# cat /etc/crictl.yaml
runtime-endpoint: unix:///run/containerd/containerd.sock
image-endpoint: unix:///run/containerd/containerd.sock
timeout: 2
root@controlplanenode:~#
```

```
root@computeplanenode:~# cat > /etc/crictl.yaml <<EOF
runtime-endpoint: unix:///run/containerd/containerd.sock
image-endpoint: unix:///run/containerd/containerd.sock
timeout: 2
> EOF
root@computeplanenode:~#
root@computeplanenode:~# cat /etc/crictl.yaml
runtime-endpoint: unix:///run/containerd/containerd.sock
image-endpoint: unix:///run/containerd/containerd.sock
timeout: 2
root@computeplanenode:~#
```

11. Now, after the above process is done, we can now install some kubernetes packages



12. Now, we are downloading the gpg key for the kubernetes repository and creating the repository in our filesystem.



13. Install the necessary kubernetes packages

```
root@computeplanenode:~# apt-get install -y kubelet kubead
m kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and ar
e no longer required:
  libflashrom1 libftdi1-2
Use 'apt autoremove' to remove them.
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 5 not upgra
ded.
Need to get 85.9 MB of archives.
After this operation, 328 MB of additional disk space will
 be used.
```

```
root@controlplanenode:~# apt-get install -y kubelet kubead
m kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and ar
e no longer required:
  libflashrom1 libftdi1-2
Use 'apt autoremove' to remove them.
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 5 not upgra
ded.
Need to get 85.9 MB of archives.
After this operation, 328 MB of additional disk space will
 be used.
```

14. To ensure that we don't face issues any further, we use hold updates.

```
root@computeplanenode:~# apt-mark hold kubelet kubeadm kub
ectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@computeplanenode:~#
```

```
root@controlplanenode:~# apt-mark hold kubelet kubeadm kub
ectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@controlplanenode:~#
```

15. From this point, we need to follow different procedures to complete the setup. So, we shall exit the multi-exec mode.


## On Control Plane Node:

1. To bootstrap the control-plane node, we have to use the **"kubeadm init"** command.

```
root@controlplanenode:~# kubeadm init --apiserver-advertise-address=192.168.1.200 --cri-socket=/run/containerd/containerd.so
ck --pod-network-cidr=10.244.0.0/16
W0424 09:48:11.337276    5605 initconfiguration.go:120] Usage of CRI endpoints without URL scheme is deprecated and can caus
e kubelet errors in the future. Automatically prepending scheme "unix" to the "criSocket" with value "/run/containerd/contai
nerd.sock". Please update your configuration!
[init] Using Kubernetes version: v1.27.1
[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'
W0424 09:48:12.317912    5605 images.go:80] could not find officially supported version of etcd for Kubernetes v1.27.1, fall
ing back to the nearest etcd version (3.5.7-0)
W0424 09:48:30.480414    5605 checks.go:835] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container ru
ntime is inconsistent with that used by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CRI sandbox
 image.
[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 [controlplanenode kubernetes kubernetes.default kubernetes.default.sv
c kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.1.200]
[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 [controlplanenode localhost] and IPs [192.168.1.200 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [controlplanenode localhost] and IPs [192.168.1.200 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
```

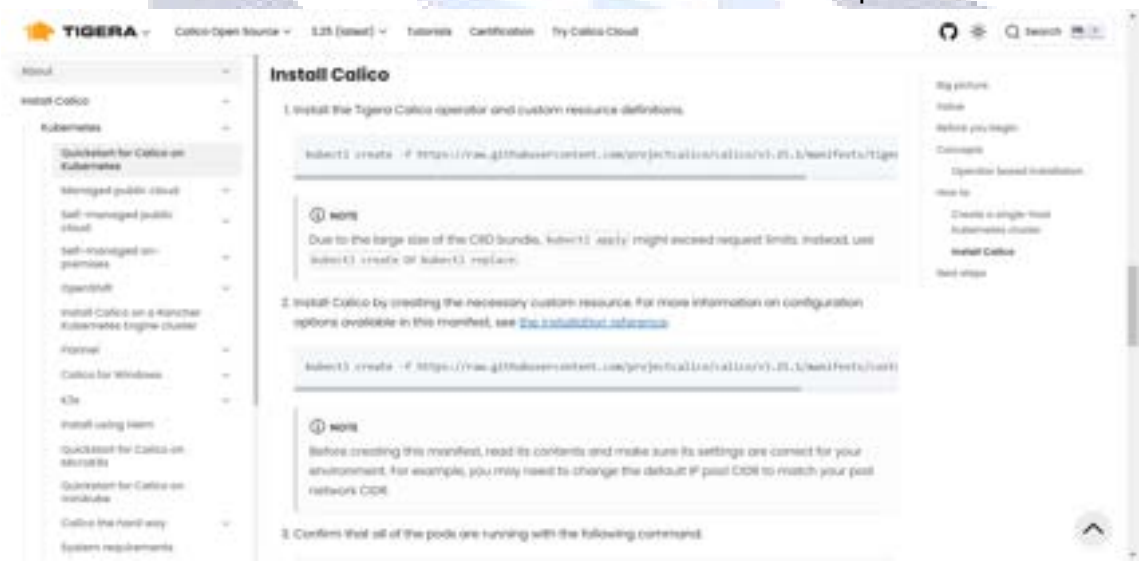
2. Upon completing the bootstrapping, we can see the instructions where we are prompted to create a directory for the config files.

```
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/

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

kubeadm join 192.168.1.200:6443 --token hu8spp.9danf5qutvwc7seu \
        --discovery-token-ca-cert-hash sha256:fd3ead50f357910cbadcdfb4283b480d0c99b40a6edc9b47579c47dd60ec12c5
root@controlplanenode:~#
```

3. So, let us create a directory and copy the config file into it.

```
root@controlplanenode:~# mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@controlplanenode:~#
root@controlplanenode:~# ls -la
total 44
drwx------  6 root root 4096 Apr 24 09:52 .
drwxr-xr-x 19 root root 4096 Apr 19 10:49 ..
-rw-------  1 root root 4568 Apr 19 13:26 .bash_history
-rw-r--r--  1 root root 3106 Oct 15  2021 .bashrc
drwxr-xr-x  2 root root 4096 Apr 24 09:52 .kube
-rw-------  1 root root   20 Apr 19 12:23 .lesshst
drwxr-xr-x  3 root root 4096 Apr 19 11:44 .local
-rw-r--r--  1 root root  161 Jul  9  2019 .profile
drwx------  3 root root 4096 Apr 19 10:58 snap
drwx------  2 root root 4096 Apr 19 10:58 .ssh
-rw-r--r--  1 root root    0 Apr 19 12:17 .sudo_as_admin_successful
root@controlplanenode:~#
root@controlplanenode:~# ls -l .kube/
total 8
-rw------- 1 root root 5641 Apr 24 09:52 config
```

4. Now, we need a container network interface. Here, we are using calico CNI. We can access the calico documentation from the net to set up the CNI.




5. Firstly, we are going to install the calico operator

```
root@controlplanenode:~# kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.1/manifests/tigera-operator.yaml

namespace/tigera-operator 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
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
root@controlplanenode:~#
```

6. Now, we need to create and apply a custom resources file.

```
root@controlplanenode:~# wget https://raw.githubusercontent.com/projectcalico/calico/v3.25.1/manifests/custom-resources.yaml

--2023-04-24 09:59:41--  https://raw.githubusercontent.com/projectcalico/calico/v3.25.1/manifests/custom-resources.yaml
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 827 [text/plain]
Saving to: 'custom-resources.yaml'

custom-resources.yaml          100%[===================================================>]     827  --.-KB/s    in 0s

2023-04-24 09:59:41 (99.1 MB/s) - 'custom-resources.yaml' saved [827/827]

root@controlplanenode:~#
```

In this file, we need to change the cidr to the range that we have specified during the bootstrapping.

```
  GNU nano 6.2                                custom-resources.yaml
# This section includes base Calico installation configuration.
# For more information, see: https://projectcalico.docs.tigera.io/master/reference/installation/api#operator.tigera.io/v1.I
apiVersion: operator.tigera.io/v1
kind: Installation
metadata:
  name: default
spec:
  # Configures Calico networking.
  calicoNetwork:
    # Note: The ipPools section cannot be modified post-install.
    ipPools:
    - blockSize: 26
      cidr: 192.168.0.0/16
      encapsulation: VXLANCrossSubnet
      natOutgoing: Enabled
      nodeSelector: all()

---

# This section configures the Calico API server.
# For more information, see: https://projectcalico.docs.tigera.io/master/reference/installation/api#operator.tigera.io/v1.A
apiVersion: operator.tigera.io/v1
kind: APIServer
metadata:
  name: default
spec: {}
```

We change the cidr to:

```
- blockSize: 26
  cidr: 10.244.0.0/16
  encapsulation: VXLANCrossSubnet
```

After changing and saving the custom-resources file, we need to apply the changes.

```
root@controlplanenode:~# kubectl apply -f custom-resources.yaml
installation.operator.tigera.io/default created
apiserver.operator.tigera.io/default created
root@controlplanenode:~#
```

7. Now that we have completed the control-plane setup, we have to connect the compute-plane nodes to the control-plane node.

```
root@controlplanenode:~# kubectl get po -A
NAMESPACE         NAME                                       READY   STATUS                  RESTARTS   AGE
calico-system     calico-kube-controllers-789dc4c76b-rf5td   0/1     Pending                 0          3m29s
calico-system     calico-node-mrw79                          0/1     Init:ImagePullBackOff   0          3m30s
calico-system     calico-typha-849686b8f6-lj2xj              0/1     ImagePullBackOff        0          3m30s
calico-system     csi-node-driver-tvrvd                      0/2     ContainerCreating       0          3m30s
kube-system       coredns-5d78c9869d-q2m7t                   0/1     Pending                 0          23m
kube-system       coredns-5d78c9869d-vbczj                   0/1     Pending                 0          23m
kube-system       etcd-controlplanenode                      1/1     Running                 0          24m
kube-system       kube-apiserver-controlplanenode            1/1     Running                 0          24m
kube-system       kube-controller-manager-controlplanenode   1/1     Running                 0          24m
kube-system       kube-proxy-m59q2                           1/1     Running                 0          23m
kube-system       kube-scheduler-controlplanenode            1/1     Running                 0          24m
tigera-operator   tigera-operator-549d4f9bdb-kwd42           1/1     Running                 0          14m
root@controlplanenode:~#
```

## On compute-plane node:

We shall use this command from the set of instructions we got after bootstrapping.

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

kubeadm join 192.168.1.200:6443 --token hu8spp.9danf5qutvwc7seu \
        --discovery-token-ca-cert-hash sha256:fd3ead50f357910cbadcdfb4283b480d0c99b40a6edc9b47579c47dd60ec12c5
```

We need to enter this command in the compute-plane node that we like to join to the control-plane node.

```
4. ControlPlaneNode(.200)    3. ComputePlaneNode(.201)
root@computeplanenode:~# kubeadm join 192.168.1.200:6443 --token hu8spp.9danf5qutvwc7seu \
        --discovery-token-ca-cert-hash sha256:fd3ead50f357910cbadcdfb4283b480d0c99b40a6edc9b47579c47dd60ec12c5
[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.

root@computeplanenode:~#
```

Now, we can see the list of nodes on the control-plane node.

```
4. ControlPlaneNode(.200)    3. ComputePlaneNode(.201)
root@controlplanenode:~# kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
computeplanenode   Ready    <none>          2m17s   v1.27.1
controlplanenode   Ready    control-plane   28m     v1.27.1
root@controlplanenode:~#
root@controlplanenode:~#
```

As we can see, we have successfully set up a multi-node kubernetes cluster using kubeadm.