Kubernetes Assignment

1. Kubernetes Installation and Worker Node Join

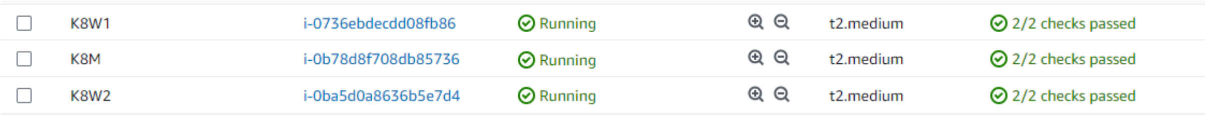
Before installing and setting up Kubernetes, we need to have below pre-requisites.

Prerequisites:

- Launch 3 EC2 Instance with at least t2.medium configuration, one EC2 will be for Master (Control Plane) and other two will be for Worker Nodes.

Pre-requisites are done as below:

- Launching 3 EC2 Instance with at least t2.medium configuration, one EC2 will be for Master (Control Plane) and other two will be for Worker Nodes.



□	K8W1	i-0736ebdecdd08fb86	✓ Running	🔍 🔍	t2.medium	✓ 2/2 checks passed
□	K8M	i-0b78d8f708db85736	✓ Running	🔍 🔍	t2.medium	✓ 2/2 checks passed
□	K8W2	i-0ba5d0a8636b5e7d4	✓ Running	🔍 🔍	t2.medium	✓ 2/2 checks passed

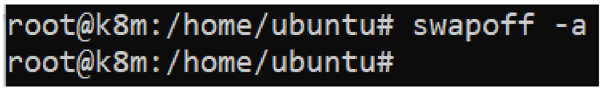
Installation Steps are divided into two parts

- ➢ Setting up containerd
- ➢ Installation steps for Kubernetes.

Installation Steps are started as below:

- **Setting up containerd on Master (K8M) EC2 instance**

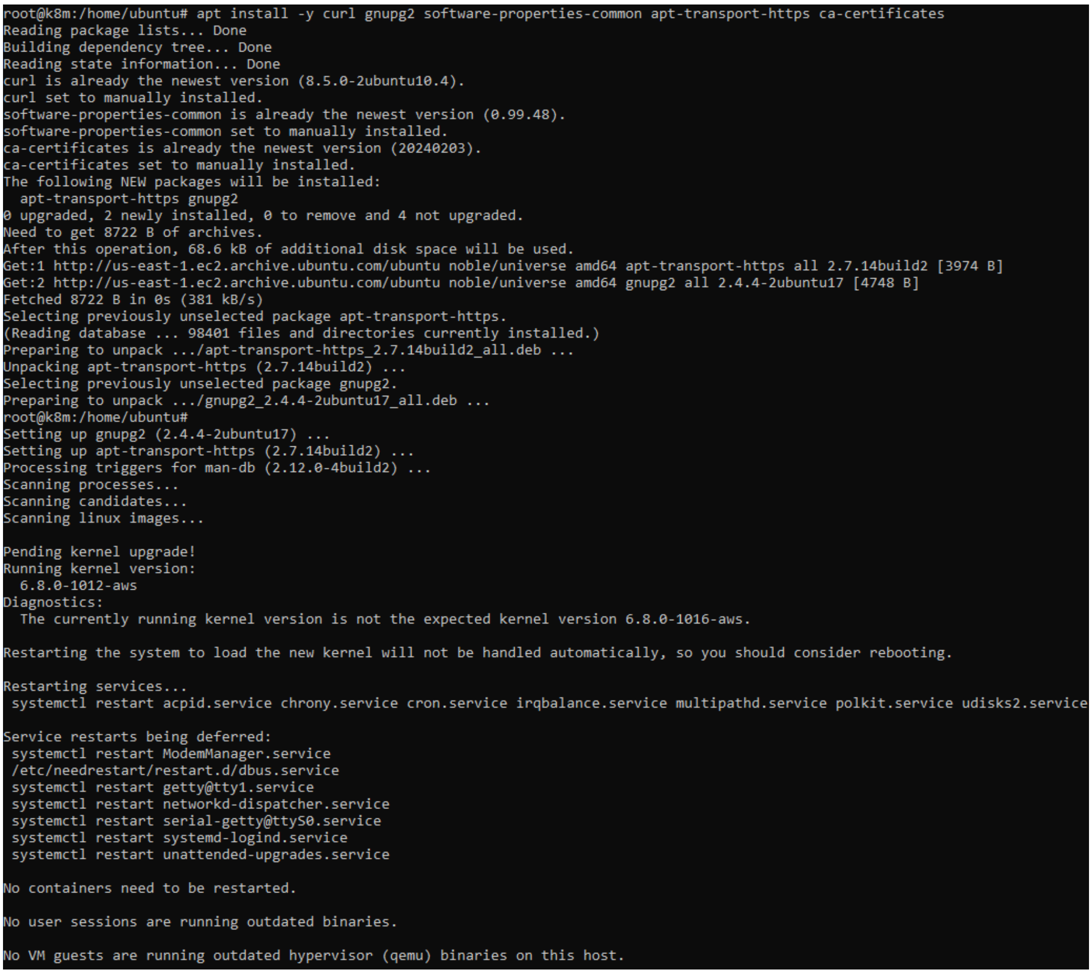
swapoff -a (This command is used to disable all swap space on the system)



```
root@k8m:/home/ubuntu# swapoff -a
root@k8m:/home/ubuntu#
```

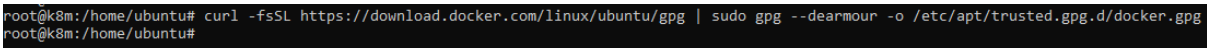
Swap space is an area on the disk that is used when the system's RAM is fully utilized. By turning off swap, you're instructing the system to stop using swap and rely solely on physical RAM.

apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates (Command to install gnu package and transport https)



```
root@k8m:/home/ubuntu# apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (8.5.0-2ubuntu10.4).
curl set to manually installed.
software-properties-common is already the newest version (0.99.48).
software-properties-common set to manually installed.
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
The following NEW packages will be installed:
  apt-transport-https gnupg2
0 upgraded, 2 newly installed, 0 to remove and 4 not upgraded.
Need to get 8722 B of archives.
After this operation, 68.6 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 apt-transport-https all 2.7.14build2 [3974 B]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 gnupg2 all 2.4.4-2ubuntu17 [4748 B]
Fetched 8722 B in 0s (381 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 98401 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.7.14build2_all.deb ...
Unpacking apt-transport-https (2.7.14build2) ...
Selecting previously unselected package gnupg2.
Preparing to unpack .../gnupg2_2.4.4-2ubuntu17_all.deb ...
root@k8m:/home/ubuntu#
Setting up gnupg2 (2.4.4-2ubuntu17) ...
Setting up apt-transport-https (2.7.14build2) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.8.0-1012-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-1016-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...
 systemctl restart acpid.service chrony.service cron.service irqbalance.service multipathd.service polkit.service udisks2.service

Service restarts being deferred:
 systemctl restart ModemManager.service
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart serial-getty@ttyS0.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service

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.
```

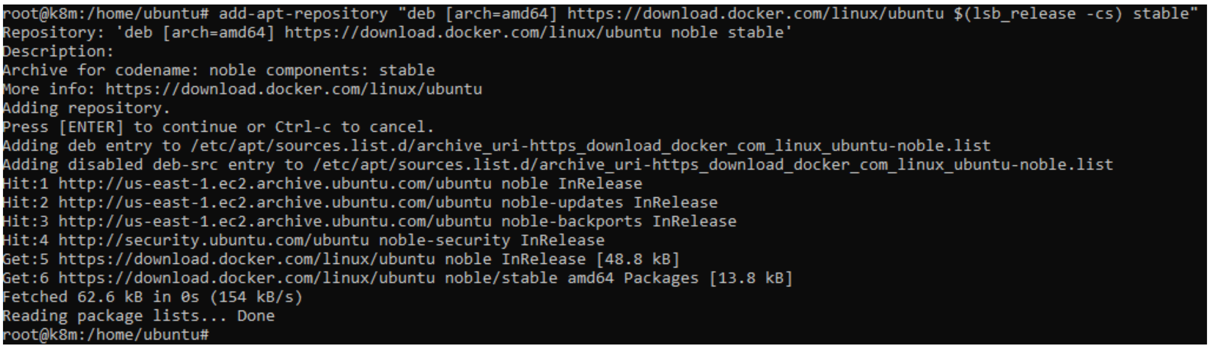
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg (Command sequence is used to add Docker's official GPG key to your system's list of trusted keys.)



```
root@k8m:/home/ubuntu# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg
root@k8m:/home/ubuntu#
```

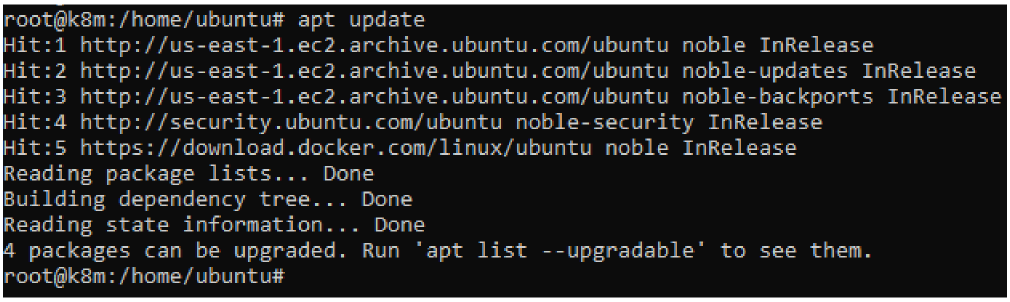
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" (Command adds Docker's APT repository to your system)

Press Enter to Add it with user input



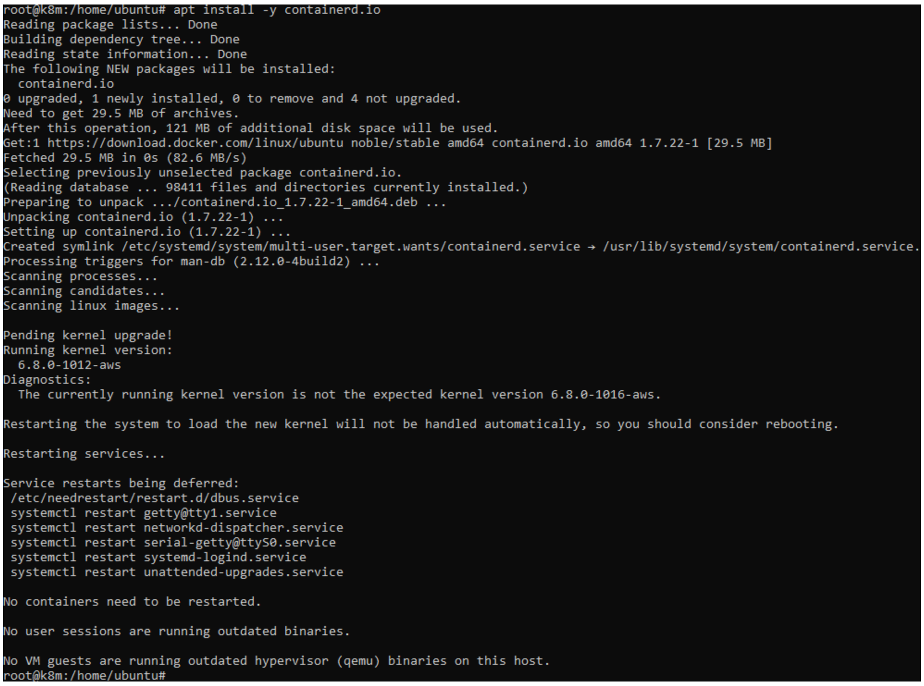
```
root@k8m:/home/ubuntu# add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [13.8 kB]
Fetched 62.6 kB in 0s (154 kB/s)
Reading package lists... Done
root@k8m:/home/ubuntu#
```

apt update (command updates the local package database)



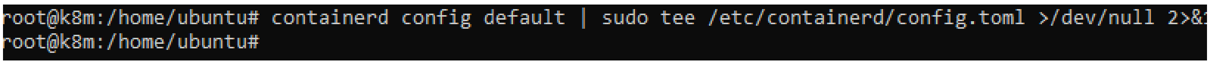
```
root@k8m:/home/ubuntu# apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
4 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@k8m:/home/ubuntu#
```

apt install -y containerd.io (Installs the containerd package)



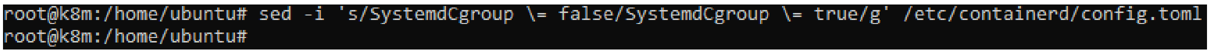
```
root@k8m:/home/ubuntu# apt install -y containerd.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  containerd.io
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 29.5 MB of archives.
After this operation, 121 MB of additional disk space will be used.
Get:1 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.22-1 [29.5 MB]
Fetched 29.5 MB in 0s (82.6 MB/s)
Selecting previously unselected package containerd.io.
(Reading database ... 98411 files and directories currently installed.)
Preparing to unpack .../containerd.io_1.7.22-1_amd64.deb ...
Unpacking containerd.io (1.7.22-1) ...
Setting up containerd.io (1.7.22-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.8.0-1012-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-1016-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...

Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart serial-getty@ttyS0.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service

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@k8m:/home/ubuntu#
```

containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1 (Command is used to generate and save a default configuration file for containerd.)



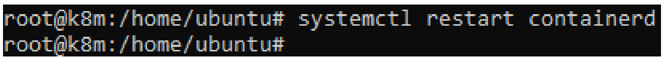
```
root@k8m:/home/ubuntu# containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
root@k8m:/home/ubuntu#
```

sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml (Command is used to modify the containerd configuration file.)



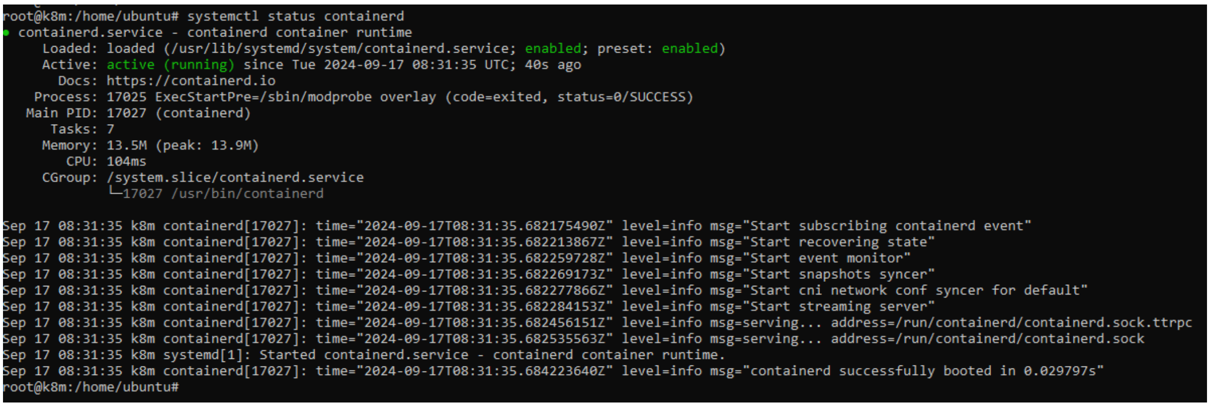
```
root@k8m:/home/ubuntu# sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml
root@k8m:/home/ubuntu#
```

systemctl restart containerd (Command restarts the containerd service)



```
root@k8m:/home/ubuntu# systemctl restart containerd
root@k8m:/home/ubuntu#
```

systemctl status containerd (Command checks the status of containerd service)



```
root@k8m:/home/ubuntu# systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-09-17 08:31:35 UTC; 40s ago
       Docs: https://containerd.io
    Process: 17025 ExecStartPre=/sbin/modprobe overlay (code=exited, status=0/SUCCESS)
   Main PID: 17027 (containerd)
      Tasks: 7
     Memory: 13.5M (peak: 13.9M)
        CPU: 104ms
     CGroup: /system.slice/containerd.service
             └─17027 /usr/bin/containerd

Sep 17 08:31:35 k8m containerd[17027]: time="2024-09-17T08:31:35.682175490Z" level=info msg="Start subscribing containerd event"
Sep 17 08:31:35 k8m containerd[17027]: time="2024-09-17T08:31:35.682213867Z" level=info msg="Start recovering state"
Sep 17 08:31:35 k8m containerd[17027]: time="2024-09-17T08:31:35.682259728Z" level=info msg="Start event monitor"
Sep 17 08:31:35 k8m containerd[17027]: time="2024-09-17T08:31:35.682269173Z" level=info msg="Start snapshots syncer"
Sep 17 08:31:35 k8m containerd[17027]: time="2024-09-17T08:31:35.682277866Z" level=info msg="Start cni network conf syncer for default"
Sep 17 08:31:35 k8m containerd[17027]: time="2024-09-17T08:31:35.682284153Z" level=info msg="Start streaming server"
Sep 17 08:31:35 k8m containerd[17027]: time="2024-09-17T08:31:35.682456151Z" level=info msg=serving... address=/run/containerd/containerd.sock.ttrpc
Sep 17 08:31:35 k8m containerd[17027]: time="2024-09-17T08:31:35.682535563Z" level=info msg=serving... address=/run/containerd/containerd.sock
Sep 17 08:31:35 k8m systemd[1]: Started containerd.service - containerd container runtime.
Sep 17 08:31:35 k8m containerd[17027]: time="2024-09-17T08:31:35.684223640Z" level=info msg="containerd successfully booted in 0.029797s"
root@k8m:/home/ubuntu#
```

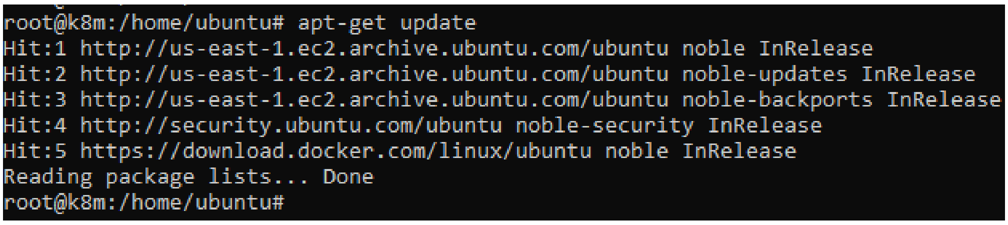
- **Installation steps for setting up Kubernetes Control plane (K8M).**

We need to visit the below website to get the installation steps for latest version and older version also. We are installing the latest version i.e. v1.31

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

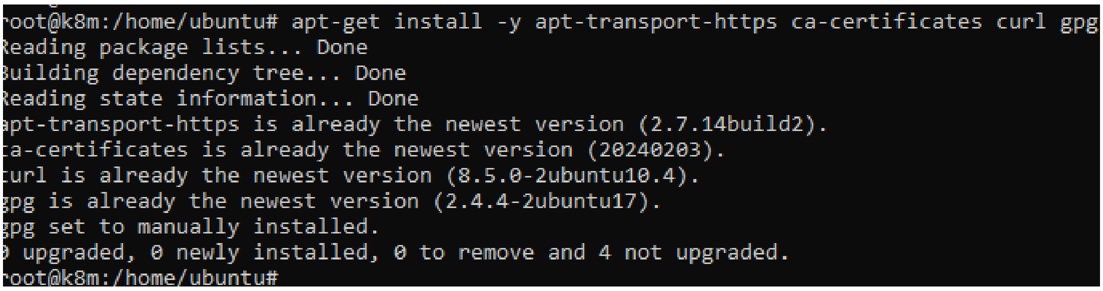
Installation Steps on Control Plane (K8M) are as below:

apt-get update (Command updates the local package database)



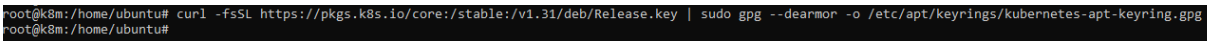
```
root@k8m:/home/ubuntu# apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
root@k8m:/home/ubuntu#
```

apt-get install -y apt-transport-https ca-certificates curl gpg (Already installed before installing containerId)



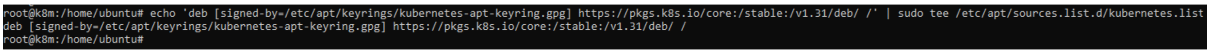
```
oot@k8m:/home/ubuntu# apt-get install -y apt-transport-https ca-certificates curl gpg
eading package lists... Done
uilding dependency tree... Done
eading state information... Done
pt-transport-https is already the newest version (2.7.14build2).
a-certificates is already the newest version (20240203).
url is already the newest version (8.5.0-2ubuntu10.4).
pg is already the newest version (2.4.4-2ubuntu17).
pg set to manually installed.
 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
oot@k8m:/home/ubuntu#
```

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
(Command fetches the key for the repository)



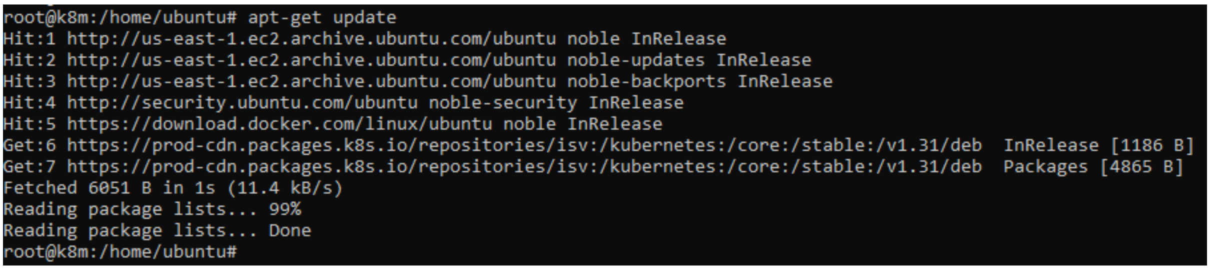
```
root@k8m:/home/ubuntu# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
root@k8m:/home/ubuntu#
```

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list (Command overwrites any existing configuration in /etc/apt/sources.list.d/kubernetes.list)



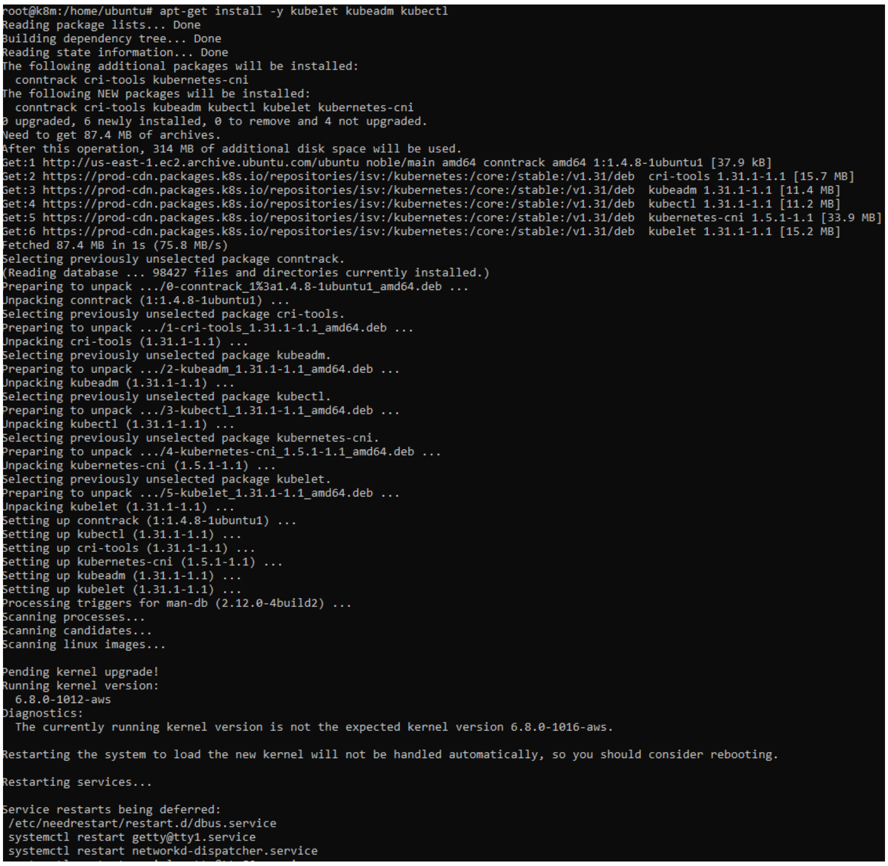
```
root@k8m:/home/ubuntu# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
root@k8m:/home/ubuntu#
```

apt-get update (Updates the package index)



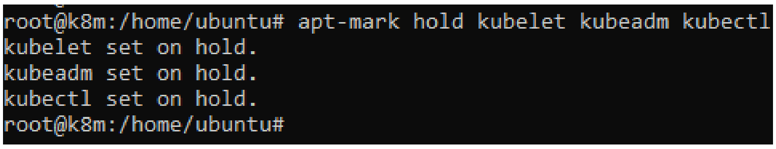
```
root@k8m:/home/ubuntu# apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (11.4 kB/s)
Reading package lists... 99%
Reading package lists... Done
root@k8m:/home/ubuntu#
```

apt-get install -y kubelet kubeadm kubectl (Command to install kubelet, kubeadm and kubectl)



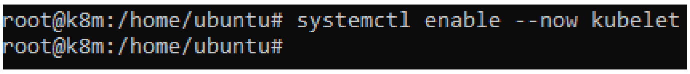
```
root@k8m:/home/ubuntu# apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 4 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubelet 1.31.1-1.1 [15.2 MB]
Fetched 87.4 MB in 1s (75.8 MB/s)
Selecting previously unselected package conntrack.
(Reading database ... 98427 files and directories currently installed.)
Preparing to unpack .../0-conntrack_1%3a1.4.8-1ubuntu1_amd64.deb ...
Unpacking conntrack (1:1.4.8-1ubuntu1) ...
Selecting previously unselected package cri-tools.
Preparing to unpack .../1-cri-tools_1.31.1-1.1_amd64.deb ...
Unpacking cri-tools (1.31.1-1.1) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../2-kubeadm_1.31.1-1.1_amd64.deb ...
Unpacking kubeadm (1.31.1-1.1) ...
Selecting previously unselected package kubectl.
Preparing to unpack .../3-kubectl_1.31.1-1.1_amd64.deb ...
Unpacking kubectl (1.31.1-1.1) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../4-kubernetes-cni_1.5.1-1.1_amd64.deb ...
Unpacking kubernetes-cni (1.5.1-1.1) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.31.1-1.1_amd64.deb ...
Unpacking kubelet (1.31.1-1.1) ...
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Setting up kubectl (1.31.1-1.1) ...
Setting up cri-tools (1.31.1-1.1) ...
Setting up kubernetes-cni (1.5.1-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.8.0-1012-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-1016-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...

Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
```

apt-mark hold kubelet kubeadm kubectl (Command prevents a specific package from being updated due to compatibility issues or other reasons. It ensures that the package remains at its current version until explicitly unheld.)



```
root@k8m:/home/ubuntu# apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@k8m:/home/ubuntu#
```

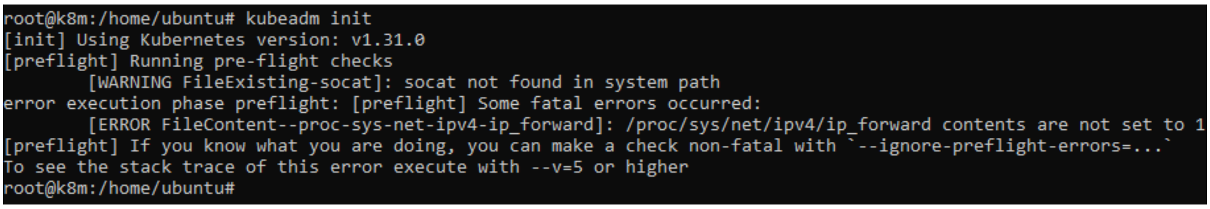
systemctl enable --now kubelet (Enable the kubelet service before running kubeadm)



```
root@k8m:/home/ubuntu# systemctl enable --now kubelet
root@k8m:/home/ubuntu#
```

kubeadm init --pod-network-cidr=10.244.0.0/16 (Command Initializes the Kubernetes setup)

It gives the below error

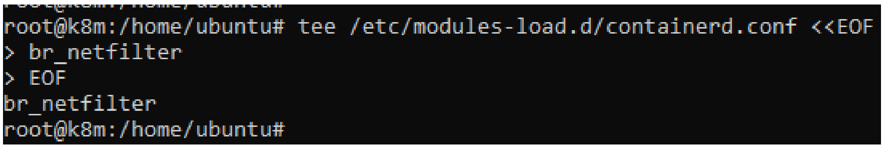


```
root@k8m:/home/ubuntu# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
root@k8m:/home/ubuntu#
```

We need to perform the below troubleshooting to initialize the kubeadm by freeing up ram and reinitialise iptables settings

- Configure the Kernel Module ‘br_netfilter’ in the containerd configuration file.

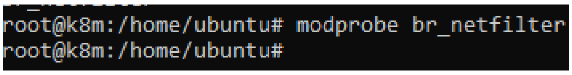
  tee /etc/modules-load.d/containerd.conf <<EOF
  br_netfilter
  EOF



```
root@k8m:/home/ubuntu# tee /etc/modules-load.d/containerd.conf <<EOF
> br_netfilter
> EOF
br_netfilter
root@k8m:/home/ubuntu#
```

- Load the br_netfilter modules into the running Linux kernel.

  modprobe br_netfilter

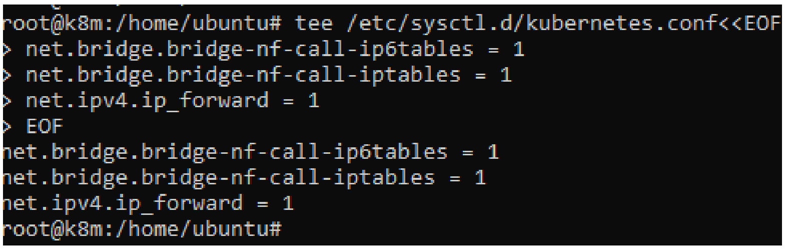


```
root@k8m:/home/ubuntu# modprobe br_netfilter
root@k8m:/home/ubuntu#
```

- Update Iptables Settings.

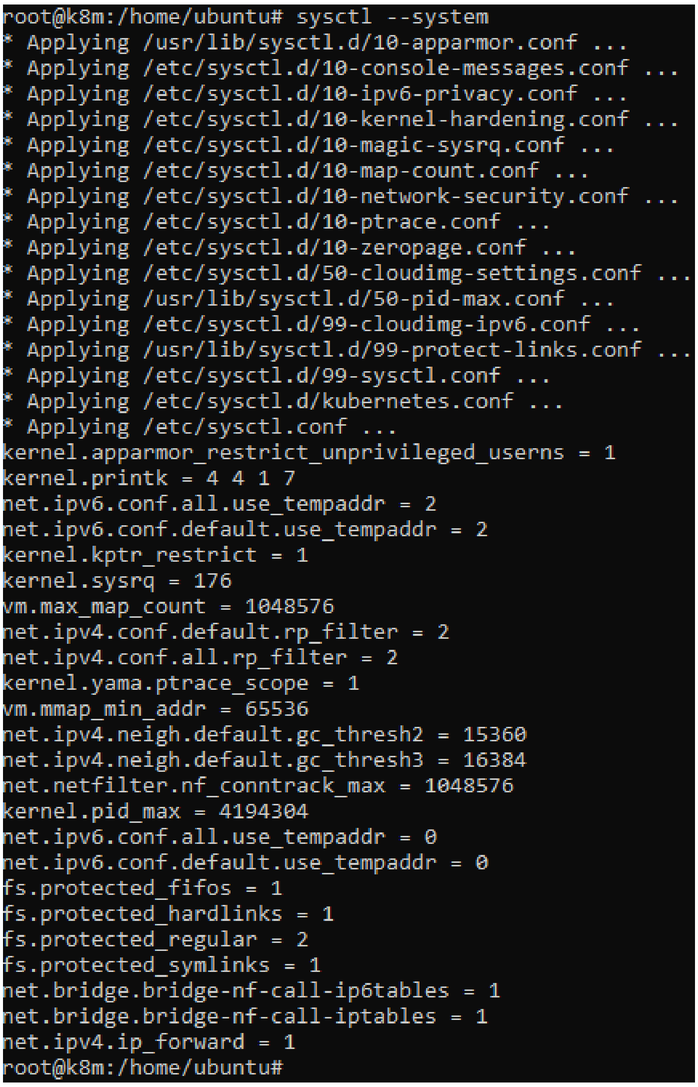
  Note: To ensure packets are properly processed by IP tables during filtering and port forwarding, set the **net.bridge.bridge-nf-call-iptables to ‘1’** in your sysctl configuration file. Otherwise, you may encounter the following error: **[ERROR FileContent–proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1.** To avoid this, execute the following command.

```
tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```



```
root@k8m:/home/ubuntu# tee /etc/sysctl.d/kubernetes.conf<<EOF
> net.bridge.bridge-nf-call-ip6tables = 1
> net.bridge.bridge-nf-call-iptables = 1
> net.ipv4.ip_forward = 1
> EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
root@k8m:/home/ubuntu#
```

`sysctl --system` (Command applies kernel settings without reboot)

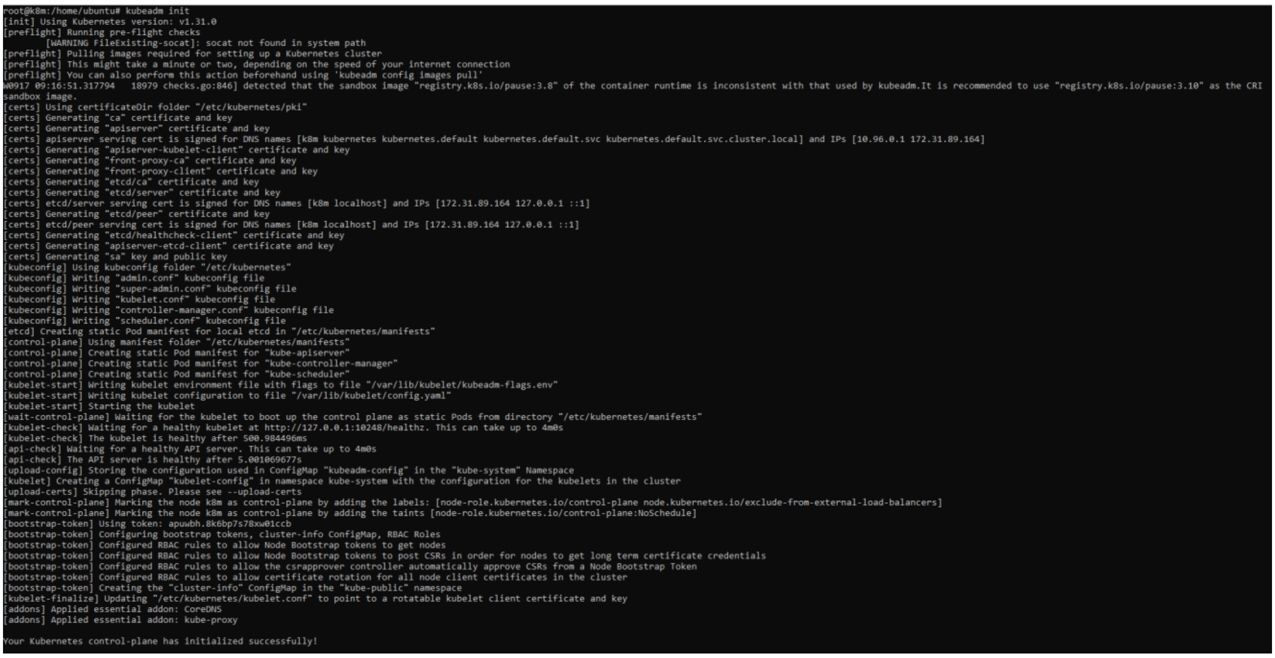


```
root@k8m:/home/ubuntu# sysctl --system
* Applying /usr/lib/sysctl.d/10-apparmor.conf ...
* Applying /etc/sysctl.d/10-console-messages.conf ...
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
* Applying /etc/sysctl.d/10-map-count.conf ...
* Applying /etc/sysctl.d/10-network-security.conf ...
* Applying /etc/sysctl.d/10-ptrace.conf ...
* Applying /etc/sysctl.d/10-zeropage.conf ...
* Applying /etc/sysctl.d/50-cloudimg-settings.conf ...
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
* Applying /etc/sysctl.d/99-cloudimg-ipv6.conf ...
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/kubernetes.conf ...
* Applying /etc/sysctl.conf ...
kernel.apparmor_restrict_unprivileged_userns = 1
kernel.printk = 4 4 1 7
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
kernel.kptr_restrict = 1
kernel.sysrq = 176
vm.max_map_count = 1048576
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.all.rp_filter = 2
kernel.yama.ptrace_scope = 1
vm.mmap_min_addr = 65536
net.ipv4.neigh.default.gc_thresh2 = 15360
net.ipv4.neigh.default.gc_thresh3 = 16384
net.netfilter.nf_conntrack_max = 1048576
kernel.pid_max = 4194304
net.ipv6.conf.all.use_tempaddr = 0
net.ipv6.conf.default.use_tempaddr = 0
fs.protected_fifos = 1
fs.protected_hardlinks = 1
fs.protected_regular = 2
fs.protected_symlinks = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
root@k8m:/home/ubuntu#
```

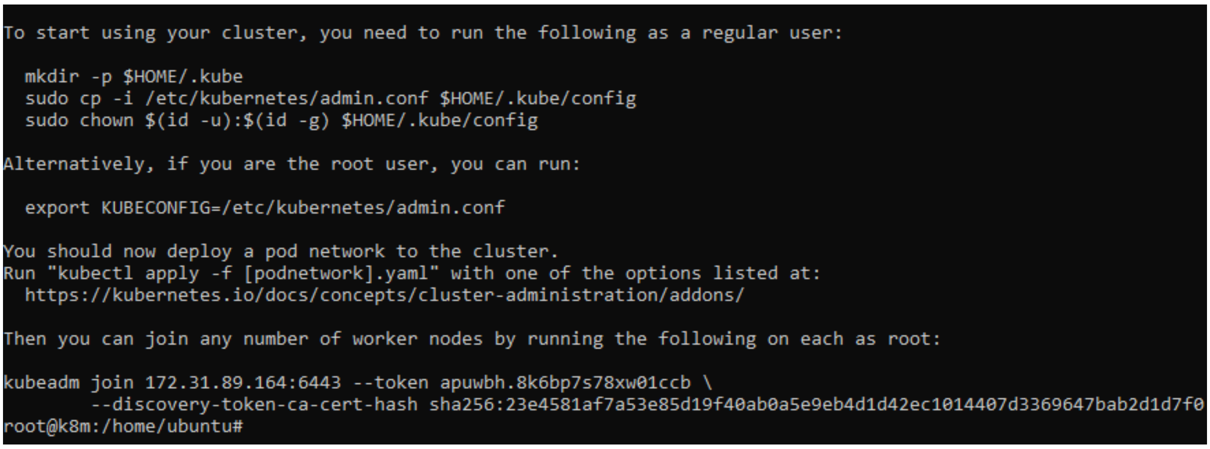
Once you've verified and potentially adjusted the configuration, proceed with reinitializing the Kubernetes cluster. we initialized the kubeadm again and installation was successful.

kubeadm init --pod-network-cidr=10.244.0.0/16

(Command Initializes the Kubernetes setup)



```
root@k8m:/home/ubuntu# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
[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 beforehand using 'kubeadm config images pull'
W0917 09:16:51.317794   18979 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" 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 [k8m kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.89.164]
[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 [k8m localhost] and IPs [172.31.89.164 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [k8m localhost] and IPs [172.31.89.164 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[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"
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 500.984496ms
[api-check] Waiting for a healthy API server. This can take up to 4m0s
[api-check] The API server is healthy after 5.001069677s
[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 k8m 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 k8m as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: apuwbh.8k6bp7s78xw01ccb
[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!
```

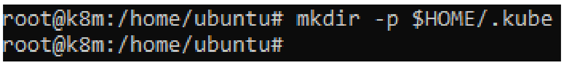


```
To start using your cluster, you need to run the following as a regular user:

  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/

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

kubeadm join 172.31.89.164:6443 --token apuwbh.8k6bp7s78xw01ccb \
        --discovery-token-ca-cert-hash sha256:23e4581af7a53e85d19f40ab0a5e9eb4d1d42ec1014407d3369647bab2d1d7f0
root@k8m:/home/ubuntu#
```

We can also see the above screenshot that the token has been generated to join the Worker Nodes to control plane. We need to run the command on both the worker nodes.

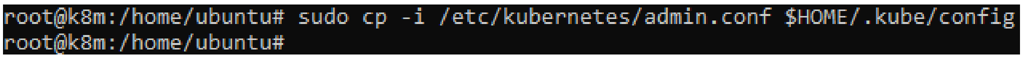
Since Kubernetes control-plane has initialized successfully, we need to run the below commands to start using the cluster.

mkdir -p $HOME/.kube



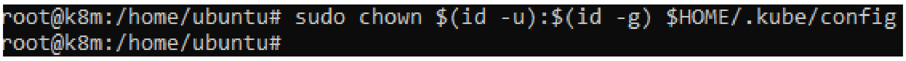
```
root@k8m:/home/ubuntu# mkdir -p $HOME/.kube
root@k8m:/home/ubuntu#
```

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



```
root@k8m:/home/ubuntu# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@k8m:/home/ubuntu#
```

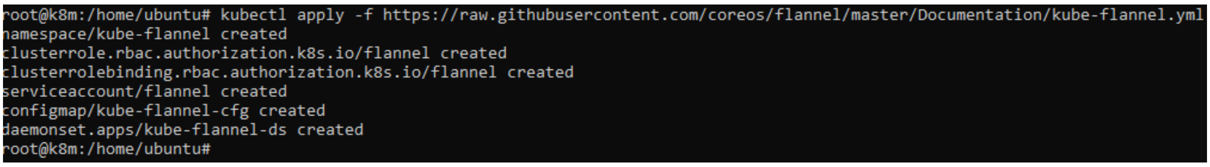
sudo chown $(id -u):$(id -g) $HOME/.kube/config



```
root@k8m:/home/ubuntu# sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@k8m:/home/ubuntu#
```

We also need to ready our cluster, for that we need to assign a flannel network using the kube-flannel.yml file by running the below command.

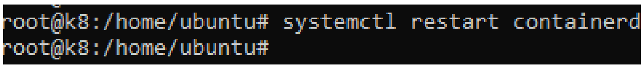
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml



```
root@k8m:/home/ubuntu# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
root@k8m:/home/ubuntu#
```

Restart the containerd service

systemctl restart containerd

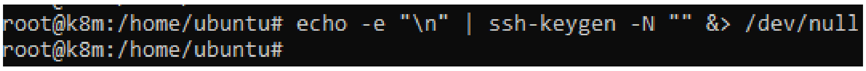


```
root@k8:/home/ubuntu# systemctl restart containerd
root@k8:/home/ubuntu#
```

Now we need to setup password less SSH between Worker Node and Control Plane.

- Command runs SSH-Keygen without prompting anything

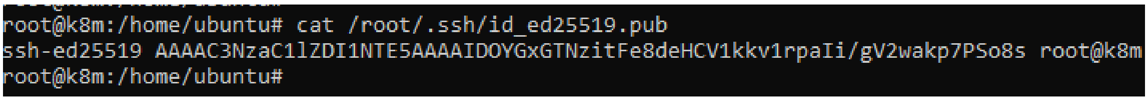
  echo -e "\n" | ssh-keygen -N "" &> /dev/null



```
root@k8m:/home/ubuntu# echo -e "\n" | ssh-keygen -N "" &> /dev/null
root@k8m:/home/ubuntu#
```

- Command to check whether the pub file is created in the below location along with the required contents.

  cat /root/.ssh/id_ed25519.pub



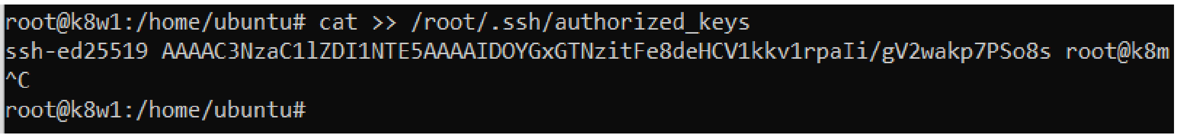
```
root@k8m:/home/ubuntu# cat /root/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDOYGxGTNzitFe8deHCV1kkv1rpaIi/gV2wakp7PSo8s root@k8m
root@k8m:/home/ubuntu#
```

Its shows that the public is created.

- Now copy the contents over the Worker Node 1 and Worker Node 2

Worker Node 1

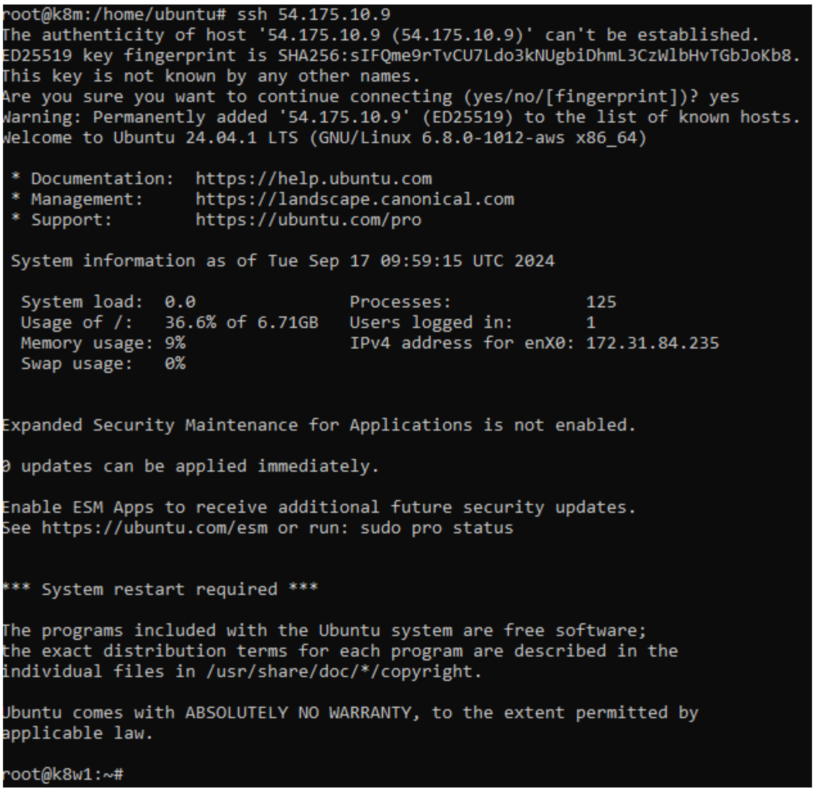
cat >> /root/.ssh/authorized_keys



```
root@k8w1:/home/ubuntu# cat >> /root/.ssh/authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDOYGxGTNzitFe8deHCV1kkv1rpaIi/gV2wakp7PSo8s root@k8m
^C
root@k8w1:/home/ubuntu#
```

Verify SSH from Control Plane

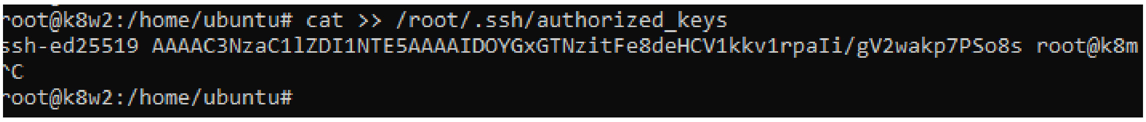
ssh Public IP of Worker 1



```
root@k8m:/home/ubuntu# ssh 54.175.10.9
The authenticity of host '54.175.10.9 (54.175.10.9)' can't be established.
ED25519 key fingerprint is SHA256:sIFQme9rTvCU7Ldo3kNUgbiDhmL3CzWlbHvTGbJoKb8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.175.10.9' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Tue Sep 17 09:59:15 UTC 2024

  System load:  0.0               Processes:             125
  Usage of /:   36.6% of 6.71GB   Users logged in:       1
  Memory usage: 9%                IPv4 address for enX0: 172.31.84.235
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@k8w1:~#
```

Worker Node 2

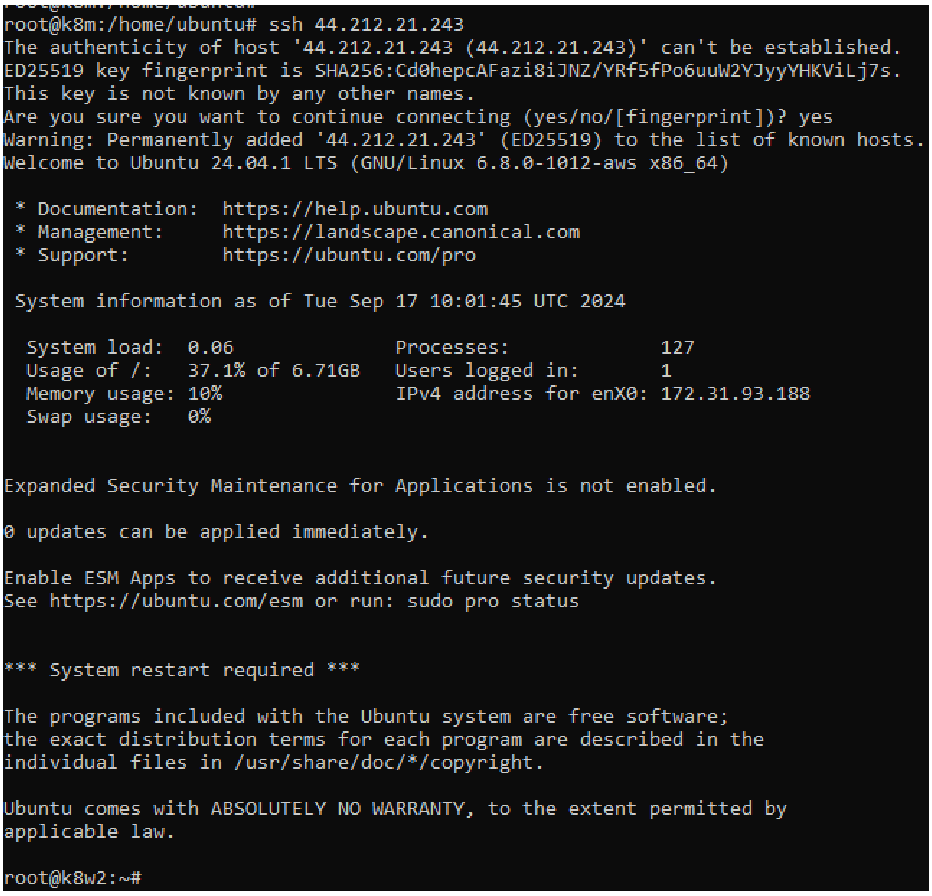
cat >> /root/.ssh/authorized_keys



```
root@k8w2:/home/ubuntu# cat >> /root/.ssh/authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDOYGxGTNzitFe8deHCV1kkv1rpaIi/gV2wakp7PSo8s root@k8m
^C
root@k8w2:/home/ubuntu#
```

Verify SSH from Control Plane

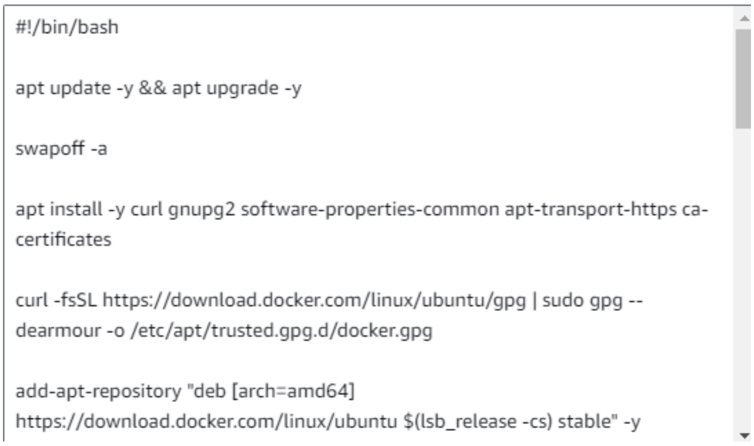
ssh Public IP of Worker 2



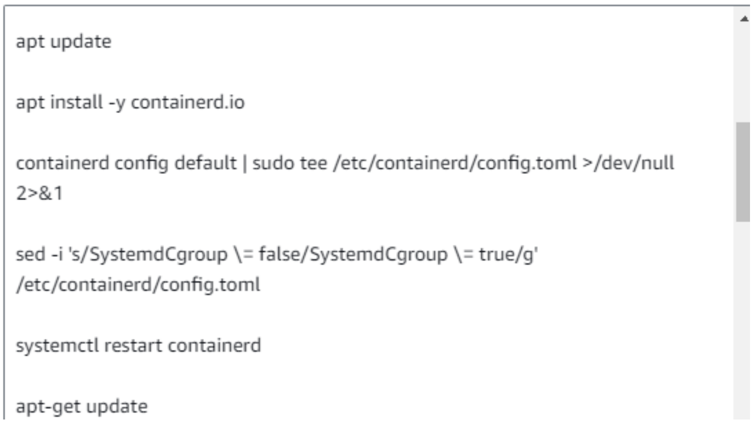
```
root@k8m:/home/ubuntu# ssh 44.212.21.243
The authenticity of host '44.212.21.243 (44.212.21.243)' can't be established.
ED25519 key fingerprint is SHA256:Cd0hepcAFazi8iJNZ/YRf5fPo6uuW2YJyyYHKViLj7s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '44.212.21.243' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Tue Sep 17 10:01:45 UTC 2024

  System load:  0.06              Processes:             127
  Usage of /:   37.1% of 6.71GB   Users logged in:       1
  Memory usage: 10%               IPv4 address for enX0: 172.31.93.188
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@k8w2:~#
```

Installation Steps on Worker Node 1 and Worker 2 are as below:

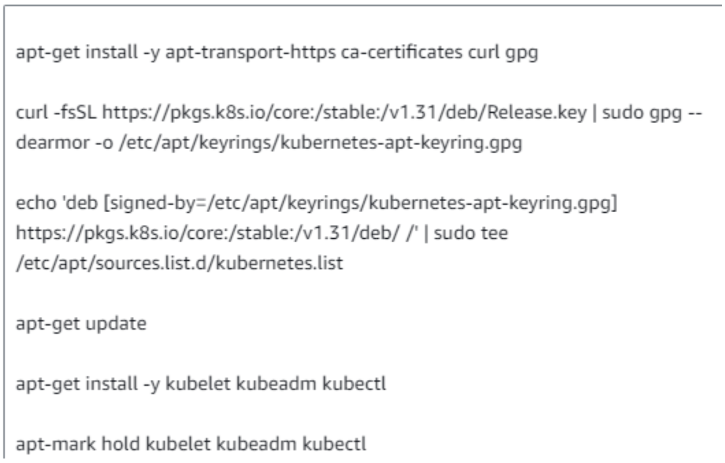
On Worker Node 1 and Worker Node 2, we have performed the installation of containerd and Kubernetes setup before “kubeadm init” step by putting all the steps performed above in the user data of EC2 instance.



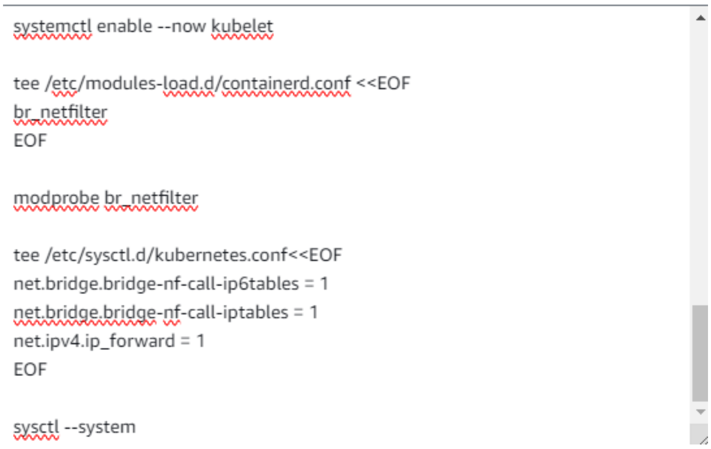
```
#!/bin/bash

apt update -y && apt upgrade -y

swapoff -a

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

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmour -o /etc/apt/trusted.gpg.d/docker.gpg

add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" -y
```



```
apt update

apt install -y containerd.io

containerd config default | sudo tee /etc/containerd/config.toml >/dev/null
2>&1

sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g'
/etc/containerd/config.toml

systemctl restart containerd

apt-get update
```



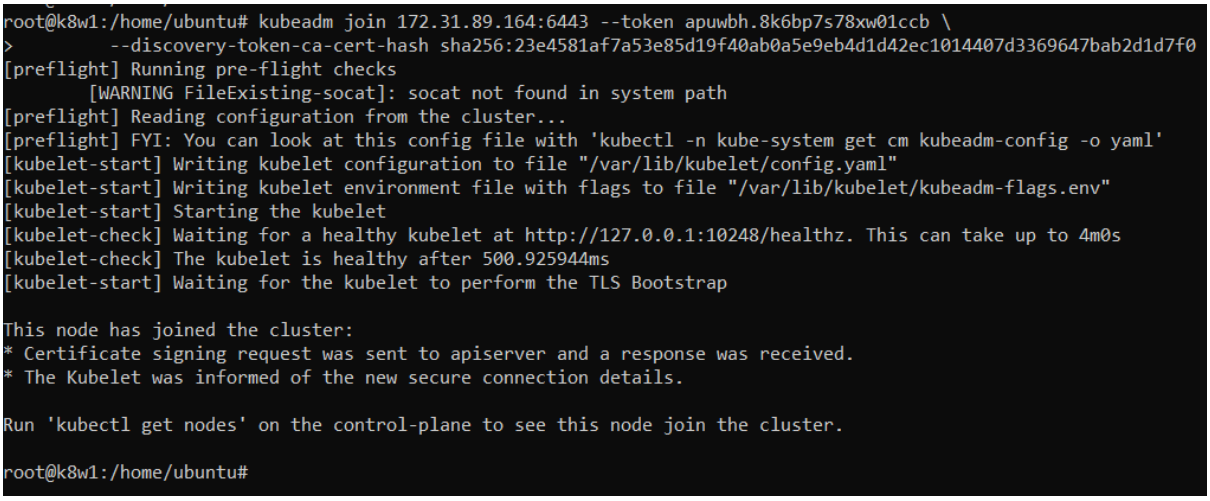
```
apt-get install -y apt-transport-https ca-certificates curl gpg

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --
dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list

apt-get update

apt-get install -y kubelet kubeadm kubectl

apt-mark hold kubelet kubeadm kubectl
```



```
systemctl enable --now kubelet

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

modprobe br_netfilter

tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF

sysctl --system
```

Now we will perform the step of joining the worker nodes to cluster using the token from Control plane (K8M)

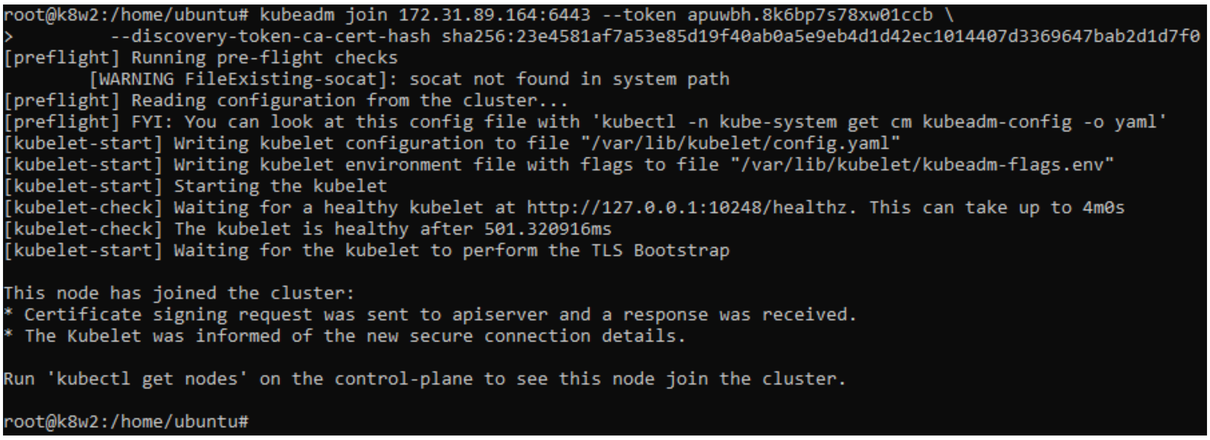
Worker Node 1



```
root@k8w1:/home/ubuntu# kubeadm join 172.31.89.164:6443 --token apuwbh.8k6bp7s78xw01ccb \
>         --discovery-token-ca-cert-hash sha256:23e4581af7a53e85d19f40ab0a5e9eb4d1d42ec1014407d3369647bab2d1d7f0
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
[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-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 500.925944ms
[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@k8w1:/home/ubuntu#
```

Successfully joined the Cluster.

Worker Node 2



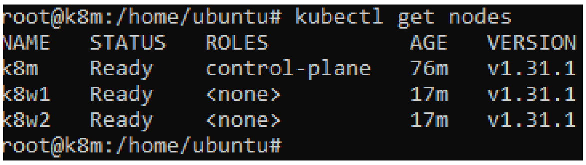
```
root@k8w2:/home/ubuntu# kubeadm join 172.31.89.164:6443 --token apuwbh.8k6bp7s78xw01ccb \
>         --discovery-token-ca-cert-hash sha256:23e4581af7a53e85d19f40ab0a5e9eb4d1d42ec1014407d3369647bab2d1d7f0
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
[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-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.320916ms
[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@k8w2:/home/ubuntu#
```

Successfully joined the Cluster.

Verify the Nodes are there.

On Control Plane, run the below command

kubectl get nodes



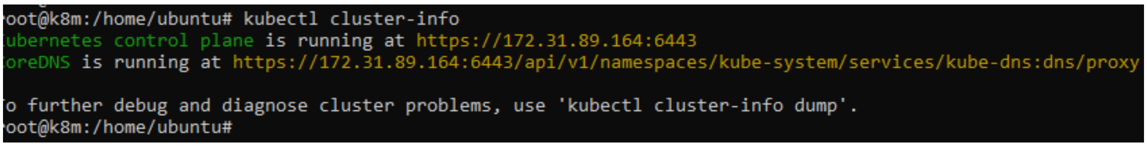
```
root@k8m:/home/ubuntu# kubectl get nodes
NAME    STATUS   ROLES           AGE   VERSION
k8m     Ready    control-plane   76m   v1.31.1
k8w1    Ready    <none>          17m   v1.31.1
k8w2    Ready    <none>          17m   v1.31.1
root@k8m:/home/ubuntu#
```

Kubernetes setup with worker node joined to cluster is completed.

2) Kubernetes Commands

➢ View Cluster Info

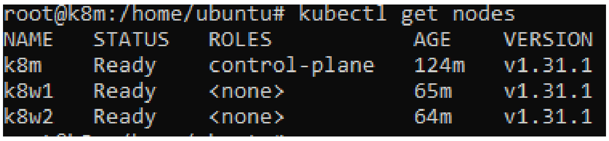
kubectl cluster-info



```
oot@k8m:/home/ubuntu# kubectl cluster-info
ubernetes control plane is running at https://172.31.89.164:6443
oreDNS is running at https://172.31.89.164:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

o further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
oot@k8m:/home/ubuntu#
```

➢ List All Nodes

kubectl get nodes

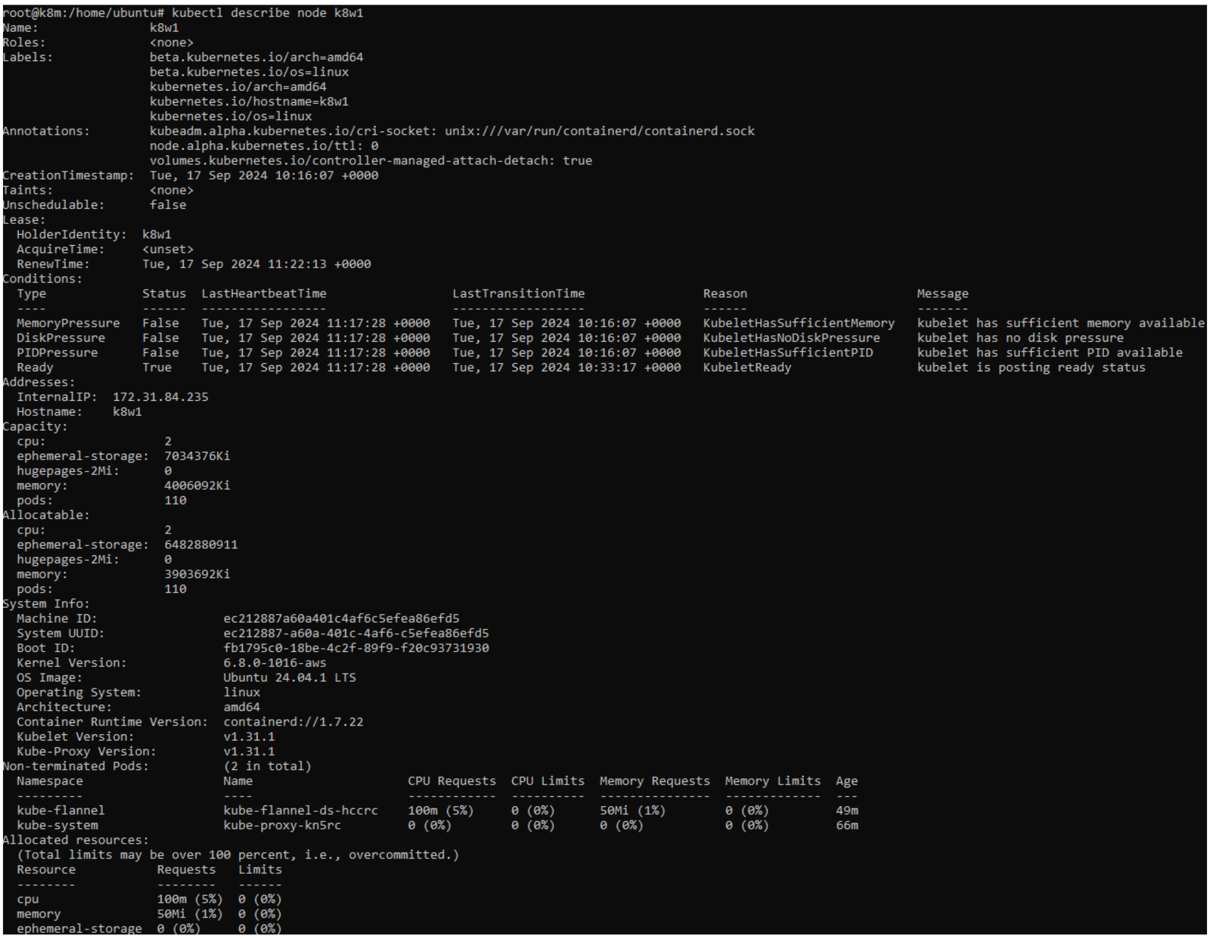


```
root@k8m:/home/ubuntu# kubectl get nodes
NAME    STATUS   ROLES           AGE    VERSION
k8m     Ready    control-plane   124m   v1.31.1
k8w1    Ready    <none>          65m    v1.31.1
k8w2    Ready    <none>          64m    v1.31.1
```

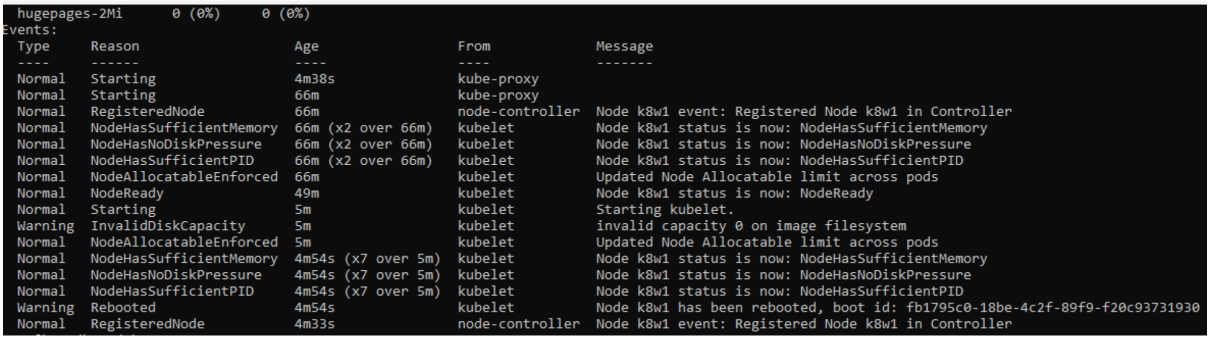
➢ Describe a Node

kubectl describe node <node-name>

kubectl describe node k8w1



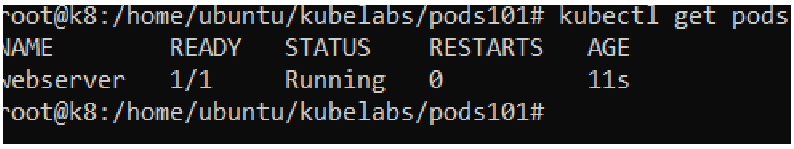
```
root@k8m:/home/ubuntu# kubectl describe node k8w1
Name:               k8w1
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=k8w1
                    kubernetes.io/os=linux
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: unix:///var/run/containerd/containerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Tue, 17 Sep 2024 10:16:07 +0000
Taints:             <none>
Unschedulable:      false
Lease:
  HolderIdentity:  k8w1
  AcquireTime:     <unset>
  RenewTime:       Tue, 17 Sep 2024 11:22:13 +0000
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----             ------  -----------------                 ------------------                ------                       -------
  MemoryPressure   False   Tue, 17 Sep 2024 11:17:28 +0000   Tue, 17 Sep 2024 10:16:07 +0000   KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure     False   Tue, 17 Sep 2024 11:17:28 +0000   Tue, 17 Sep 2024 10:16:07 +0000   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure      False   Tue, 17 Sep 2024 11:17:28 +0000   Tue, 17 Sep 2024 10:16:07 +0000   KubeletHasSufficientPID      kubelet has sufficient PID available
  Ready            True    Tue, 17 Sep 2024 11:17:28 +0000   Tue, 17 Sep 2024 10:33:17 +0000   KubeletReady                 kubelet is posting ready status
Addresses:
  InternalIP:  172.31.84.235
  Hostname:    k8w1
Capacity:
  cpu:                2
  ephemeral-storage:  7034376Ki
  hugepages-2Mi:      0
  memory:             4006092Ki
  pods:               110
Allocatable:
  cpu:                2
  ephemeral-storage:  6482880911
  hugepages-2Mi:      0
  memory:             3903692Ki
  pods:               110
System Info:
  Machine ID:                 ec212887a60a401c4af6c5efea86efd5
  System UUID:                ec212887-a60a-401c-4af6-c5efea86efd5
  Boot ID:                    fb1795c0-18be-4c2f-89f9-f20c93731930
  Kernel Version:             6.8.0-1016-aws
  OS Image:                   Ubuntu 24.04.1 LTS
  Operating System:           linux
  Architecture:               amd64
  Container Runtime Version:  containerd://1.7.22
  Kubelet Version:            v1.31.1
  Kube-Proxy Version:         v1.31.1
Non-terminated Pods:          (2 in total)
  Namespace                   Name                     CPU Requests  CPU Limits  Memory Requests  Memory Limits  Age
  ---------                   ----                     ------------  ----------  ---------------  -------------  ---
  kube-flannel                kube-flannel-ds-hccrc    100m (5%)     0 (0%)      50Mi (1%)        0 (0%)         49m
  kube-system                 kube-proxy-kn5rc         0 (0%)        0 (0%)      0 (0%)           0 (0%)         66m
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests   Limits
  --------           --------   ------
  cpu                100m (5%)  0 (0%)
  memory             50Mi (1%)  0 (0%)
  ephemeral-storage  0 (0%)     0 (0%)
```



```
  hugepages-2Mi      0 (0%)     0 (0%)
Events:
  Type     Reason                   Age                 From             Message
  ----     ------                   ----                ----             -------
  Normal   Starting                 4m38s               kube-proxy
  Normal   Starting                 66m                 kube-proxy
  Normal   RegisteredNode           66m                 node-controller  Node k8w1 event: Registered Node k8w1 in Controller
  Normal   NodeHasSufficientMemory  66m (x2 over 66m)   kubelet          Node k8w1 status is now: NodeHasSufficientMemory
  Normal   NodeHasNoDiskPressure    66m (x2 over 66m)   kubelet          Node k8w1 status is now: NodeHasNoDiskPressure
  Normal   NodeHasSufficientPID     66m (x2 over 66m)   kubelet          Node k8w1 status is now: NodeHasSufficientPID
  Normal   NodeAllocatableEnforced  66m                 kubelet          Updated Node Allocatable limit across pods
  Normal   NodeReady                49m                 kubelet          Node k8w1 status is now: NodeReady
  Normal   Starting                 5m                  kubelet          Starting kubelet.
  Warning  InvalidDiskCapacity      5m                  kubelet          invalid capacity 0 on image filesystem
  Normal   NodeAllocatableEnforced  5m                  kubelet          Updated Node Allocatable limit across pods
  Normal   NodeHasSufficientMemory  4m54s (x7 over 5m)  kubelet          Node k8w1 status is now: NodeHasSufficientMemory
  Normal   NodeHasNoDiskPressure    4m54s (x7 over 5m)  kubelet          Node k8w1 status is now: NodeHasNoDiskPressure
  Normal   NodeHasSufficientPID     4m54s (x7 over 5m)  kubelet          Node k8w1 status is now: NodeHasSufficientPID
  Warning  Rebooted                 4m54s               kubelet          Node k8w1 has been rebooted, boot id: fb1795c0-18be-4c2f-89f9-f20c93731930
  Normal   RegisteredNode           4m33s               node-controller  Node k8w1 event: Registered Node k8w1 in Controller
```

➢ Get Pod Information

kubectl get pods

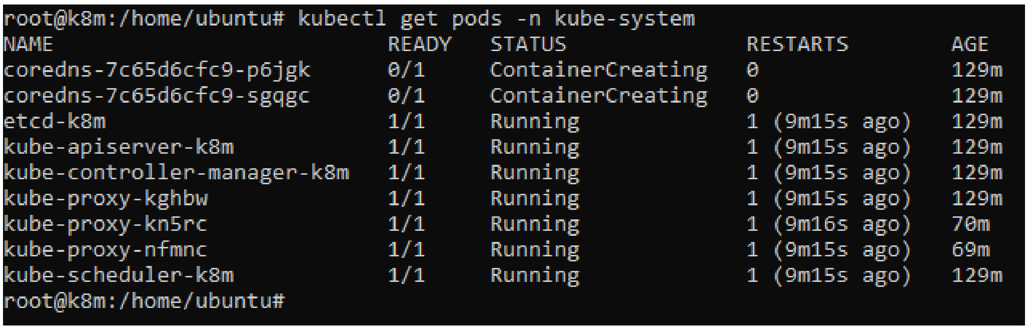


```
oot@k8:/home/ubuntu/kubelabs/pods101# kubectl get pods
NAME          READY    STATUS     RESTARTS    AGE
ebserver    1/1      Running    0           11s
oot@k8:/home/ubuntu/kubelabs/pods101#
```

➢ Get Pods in a Specific Namespace

kubectl get pods -n <namespace>

kubectl get pods -n kube-system

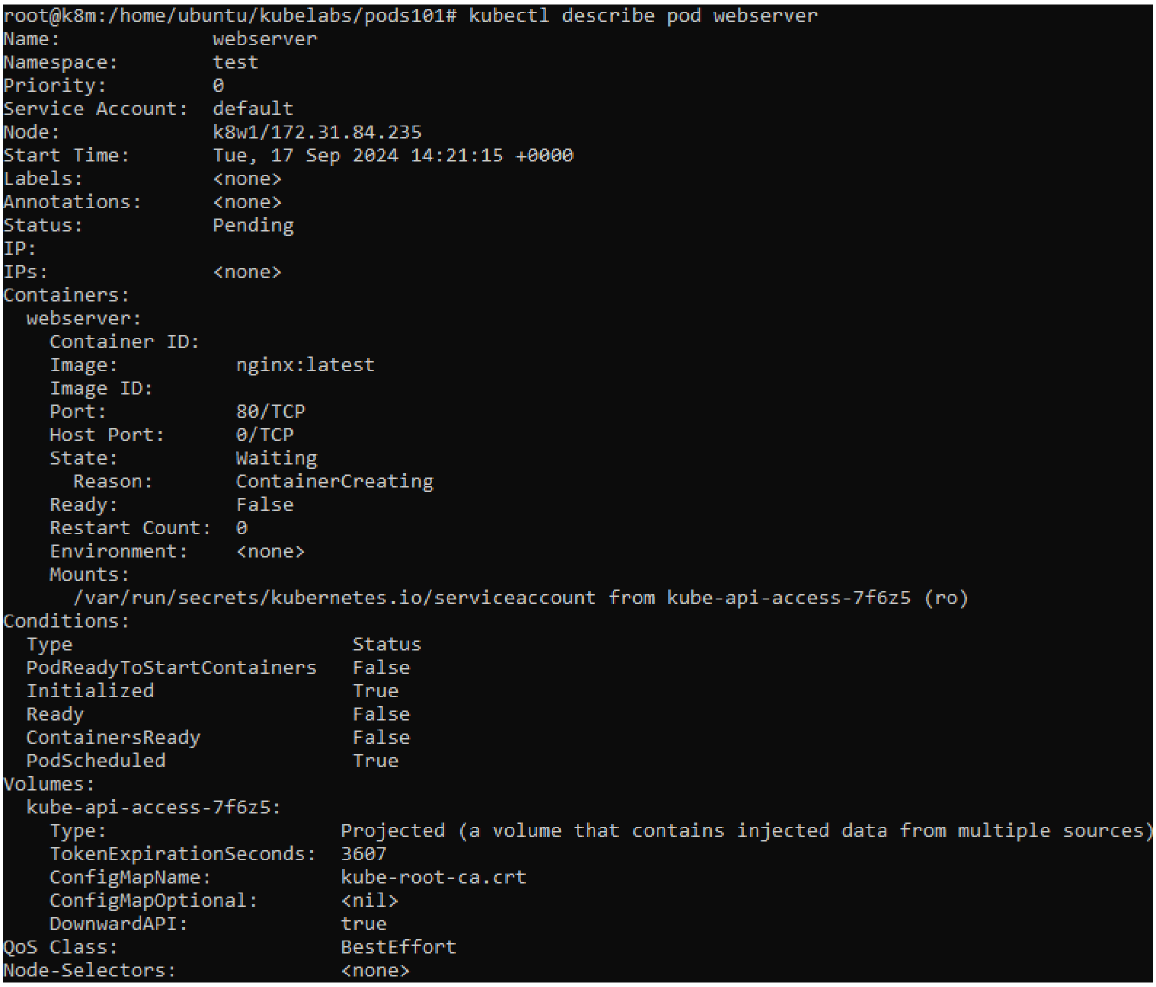


```
root@k8m:/home/ubuntu# kubectl get pods -n kube-system
NAME                              READY   STATUS              RESTARTS        AGE
coredns-7c65d6cfc9-p6jgk          0/1     ContainerCreating   0               129m
coredns-7c65d6cfc9-sgqgc          0/1     ContainerCreating   0               129m
etcd-k8m                          1/1     Running             1 (9m15s ago)   129m
kube-apiserver-k8m                1/1     Running             1 (9m15s ago)   129m
kube-controller-manager-k8m       1/1     Running             1 (9m15s ago)   129m
kube-proxy-kghbw                  1/1     Running             1 (9m15s ago)   129m
kube-proxy-kn5rc                  1/1     Running             1 (9m16s ago)   70m
kube-proxy-nfmnc                  1/1     Running             1 (9m15s ago)   69m
kube-scheduler-k8m                1/1     Running             1 (9m15s ago)   129m
root@k8m:/home/ubuntu#
```

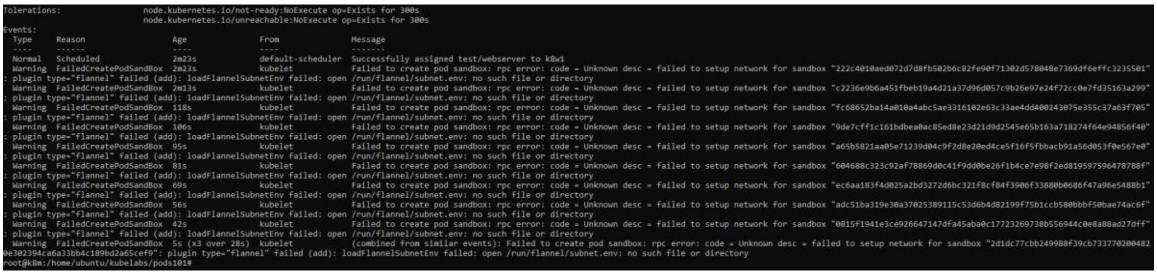
➢ Describe a Pod

kubectl describe pod <pod-name>

kubectl describe pod webserver



```
root@k8m:/home/ubuntu/kubelabs/pods101# kubectl describe pod webserver
Name:             webserver
Namespace:        test
Priority:         0
Service Account:  default
Node:             k8w1/172.31.84.235
Start Time:       Tue, 17 Sep 2024 14:21:15 +0000
Labels:           <none>
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Containers:
  webserver:
    Container ID:
    Image:          nginx:latest
    Image ID:
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-7f6z5 (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   False
  Initialized                 True
  Ready                       False
  ContainersReady             False
  PodScheduled                True
Volumes:
  kube-api-access-7f6z5:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
```

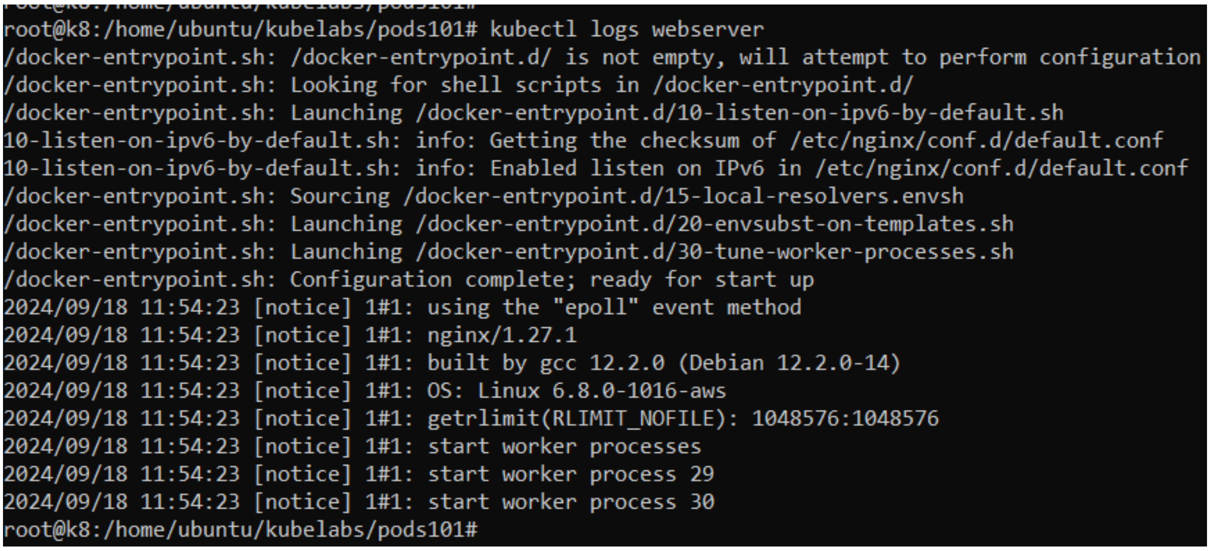


```
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason                  Age                From               Message
  ----     ------                  ----               ----               -------
  Normal   Scheduled               2m23s              default-scheduler  Successfully assigned test/webserver to k8w1
  Warning  FailedCreatePodSandBox  2m23s              kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup network for sandbox "222c4010aed072d7d8fb502b6c82fe90f71302d578048e7369df6effc3235501"
: plugin type="flannel" failed (add): loadFlannelSubnetEnv failed: open /run/flannel/subnet.env: no such file or directory
  Warning  FailedCreatePodSandBox  2m13s              kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup network for sandbox "c2236e9b6a451fbeb19a4d21a37d96d057c9b26e97e24f72cc0e7fd35163a299"
: plugin type="flannel" failed (add): loadFlannelSubnetEnv failed: open /run/flannel/subnet.env: no such file or directory
  Warning  FailedCreatePodSandBox  118s               kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup network for sandbox "fc68652ba14a010a4abc5ae3316102e63c33ae4dd400243075e355c37a63f705"
: plugin type="flannel" failed (add): loadFlannelSubnetEnv failed: open /run/flannel/subnet.env: no such file or directory
  Warning  FailedCreatePodSandBox  106s               kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup network for sandbox "9de7cff1c161bdbea0ac85ed8e23d21d9d2545e65b163a718274f64e94856f40"
: plugin type="flannel" failed (add): loadFlannelSubnetEnv failed: open /run/flannel/subnet.env: no such file or directory
  Warning  FailedCreatePodSandBox  95s                kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup network for sandbox "a65b5821aa05e71239d04c9f2d8e20ed4ce5f16f5fbbacb91a56d053f0e567e0"
: plugin type="flannel" failed (add): loadFlannelSubnetEnv failed: open /run/flannel/subnet.env: no such file or directory
  Warning  FailedCreatePodSandBox  81s                kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup network for sandbox "604688c323c92af78869d0c41f9dd0be26f1b4ce7e98f2ed819597596478788f"
: plugin type="flannel" failed (add): loadFlannelSubnetEnv failed: open /run/flannel/subnet.env: no such file or directory
  Warning  FailedCreatePodSandBox  69s                kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup network for sandbox "ec6aa183f4d025a2bd3272d6bc321f8cf84f3906f33880b0686f47a96e5488b1"
: plugin type="flannel" failed (add): loadFlannelSubnetEnv failed: open /run/flannel/subnet.env: no such file or directory
  Warning  FailedCreatePodSandBox  56s                kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup network for sandbox "adc51ba319e30a37025389115c53d6b4d82199f75b1ccb580bbbf50bae74ac6f"
: plugin type="flannel" failed (add): loadFlannelSubnetEnv failed: open /run/flannel/subnet.env: no such file or directory
  Warning  FailedCreatePodSandBox  42s                kubelet            Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup network for sandbox "0815f1941e3ce926647147dfa45aba0c17723269738b556944c0e8a88ad27dff"
: plugin type="flannel" failed (add): loadFlannelSubnetEnv failed: open /run/flannel/subnet.env: no such file or directory
  Warning  FailedCreatePodSandBox  5s (x3 over 28s)   kubelet            (combined from similar events): Failed to create pod sandbox: rpc error: code = Unknown desc = failed to setup network for sandbox "2d1dc77cbb249988f39cb73377020482
0e302394ca6a33bb4c189bd2a65cef9": plugin type="flannel" failed (add): loadFlannelSubnetEnv failed: open /run/flannel/subnet.env: no such file or directory
root@k8m:/home/ubuntu/kubelabs/pods101#
```

➢ View Pod Logs

kubectl logs <pod-name>

kubectl logs webserver

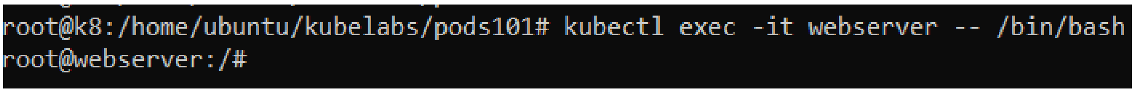


```
root@k8:/home/ubuntu/kubelabs/pods101#
root@k8:/home/ubuntu/kubelabs/pods101# kubectl logs webserver
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/09/18 11:54:23 [notice] 1#1: using the "epoll" event method
2024/09/18 11:54:23 [notice] 1#1: nginx/1.27.1
2024/09/18 11:54:23 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/09/18 11:54:23 [notice] 1#1: OS: Linux 6.8.0-1016-aws
2024/09/18 11:54:23 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/09/18 11:54:23 [notice] 1#1: start worker processes
2024/09/18 11:54:23 [notice] 1#1: start worker process 29
2024/09/18 11:54:23 [notice] 1#1: start worker process 30
root@k8:/home/ubuntu/kubelabs/pods101#
```

- **Execute Command in a Pod**

kubectl exec -it <pod-name> -- /bin/

kubectl exec -it webserver -- /bin/bash

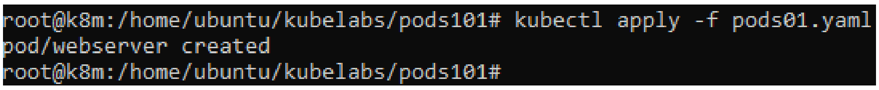


```
root@k8:/home/ubuntu/kubelabs/pods101# kubectl exec -it webserver -- /bin/bash
root@webserver:/#
```

- **Create Resources from a YAML File**

kubectl apply -f <file.yaml>

kubectl apply -f pods01.yaml

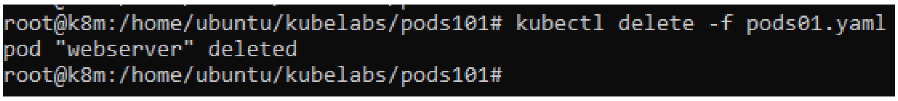


```
root@k8m:/home/ubuntu/kubelabs/pods101# kubectl apply -f pods01.yaml
pod/webserver created
root@k8m:/home/ubuntu/kubelabs/pods101#
```

Delete Resources from a YAML File

kubectl delete -f <file.yaml>

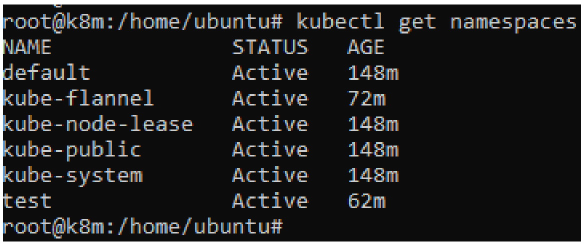
kubectl delete -f pods01.yaml



```
root@k8m:/home/ubuntu/kubelabs/pods101# kubectl delete -f pods01.yaml
pod "webserver" deleted
root@k8m:/home/ubuntu/kubelabs/pods101#
```

Get Namespaces

kubectl get namespaces

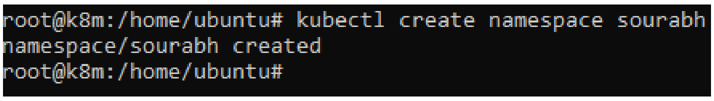


```
root@k8m:/home/ubuntu# kubectl get namespaces
NAME              STATUS   AGE
default           Active   148m
kube-flannel      Active   72m
kube-node-lease   Active   148m
kube-public       Active   148m
kube-system       Active   148m
test              Active   62m
root@k8m:/home/ubuntu#
```

Create a Namespace

kubectl create namespace <namespace-name>

kubectl create namespace sourabh

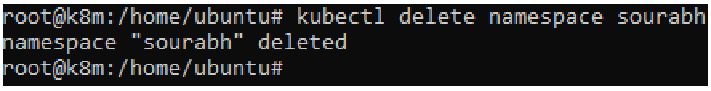


```
root@k8m:/home/ubuntu# kubectl create namespace sourabh
namespace/sourabh created
root@k8m:/home/ubuntu#
```

Delete a Namespace

kubectl delete namespace <namespace-name>

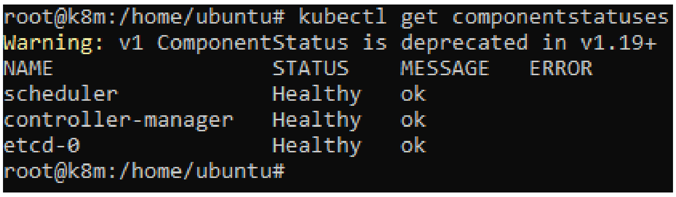
kubectl delete namespace sourabh



```
root@k8m:/home/ubuntu# kubectl delete namespace sourabh
namespace "sourabh" deleted
root@k8m:/home/ubuntu#
```

Check Cluster Health

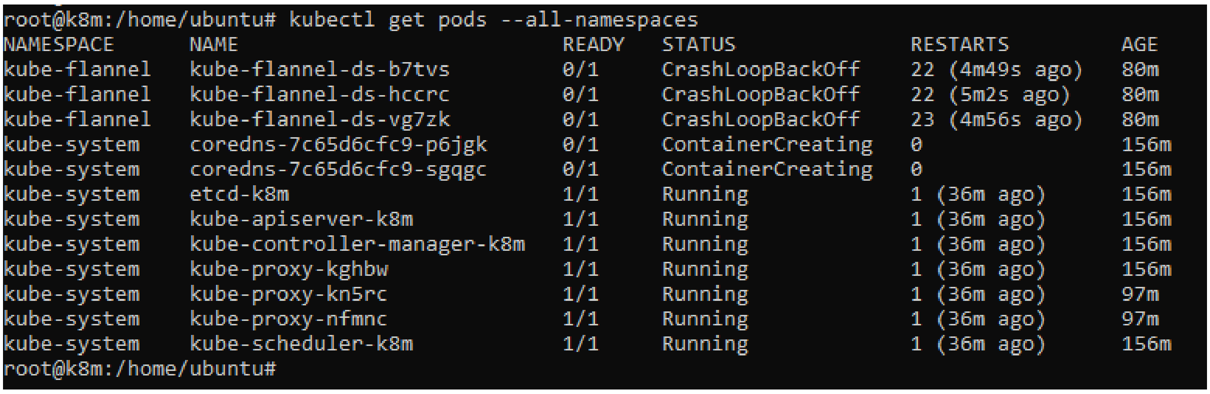
kubectl get componentstatuses



```
root@k8m:/home/ubuntu# kubectl get componentstatuses
Warning: v1 ComponentStatus is deprecated in v1.19+
NAME                 STATUS    MESSAGE   ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-0               Healthy   ok
root@k8m:/home/ubuntu#
```

Display pod information for all namespaces

kubectl get pods –all-namespaces

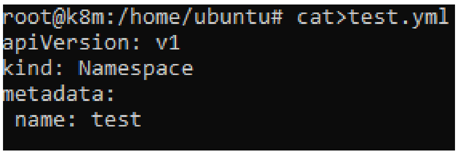


```
root@k8m:/home/ubuntu# kubectl get pods --all-namespaces
NAMESPACE      NAME                             READY   STATUS              RESTARTS          AGE
kube-flannel   kube-flannel-ds-b7tvs            0/1     CrashLoopBackOff    22 (4m49s ago)    80m
kube-flannel   kube-flannel-ds-hccrc            0/1     CrashLoopBackOff    22 (5m2s ago)     80m
kube-flannel   kube-flannel-ds-vg7zk            0/1     CrashLoopBackOff    23 (4m56s ago)    80m
kube-system    coredns-7c65d6cfc9-p6jgk         0/1     ContainerCreating   0                 156m
kube-system    coredns-7c65d6cfc9-sgqgc         0/1     ContainerCreating   0                 156m
kube-system    etcd-k8m                         1/1     Running             1 (36m ago)       156m
kube-system    kube-apiserver-k8m               1/1     Running             1 (36m ago)       156m
kube-system    kube-controller-manager-k8m      1/1     Running             1 (36m ago)       156m
kube-system    kube-proxy-kghbw                 1/1     Running             1 (36m ago)       156m
kube-system    kube-proxy-kn5rc                 1/1     Running             1 (36m ago)       97m
kube-system    kube-proxy-nfmnc                 1/1     Running             1 (36m ago)       97m
kube-system    kube-scheduler-k8m               1/1     Running             1 (36m ago)       156m
root@k8m:/home/ubuntu#
```

3) Creating a Namespace and changing default namespace to your created namespace.

- Create the yml file for creating the namespace.

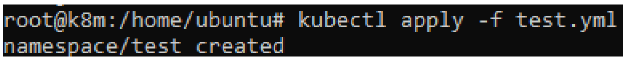
  cat>test.yml
  apiVersion: v1
  kind: Namespace
  metadata:
   name: test



```
root@k8m:/home/ubuntu# cat>test.yml
apiVersion: v1
kind: Namespace
metadata:
 name: test
```

- Now we will create the name space using the below command

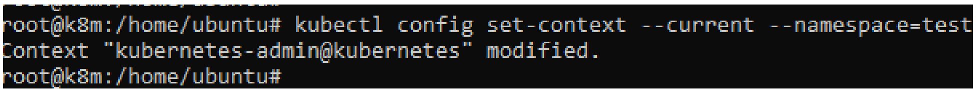
  kubectl apply -f test.yml (we can also use create in place of apply, its same as run and pull in docker)



```
root@k8m:/home/ubuntu# kubectl apply -f test.yml
namespace/test created
```

- Switch Namespaces (Changing the default namespace to our created namespace test) by using the below command.

  kubectl config set-context --current --namespace=test

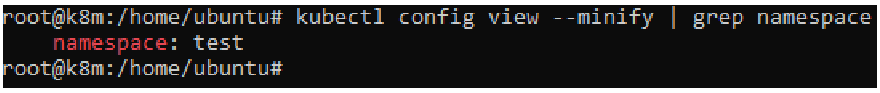


```
root@k8m:/home/ubuntu# kubectl config set-context --current --namespace=test
Context "kubernetes-admin@kubernetes" modified.
root@k8m:/home/ubuntu#
```

We can see that modification are done successfully.

- To check the current namespace, we will run the below command

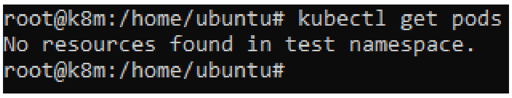
```
kubectl config view --minify | grep namespace
```



```
root@k8m:/home/ubuntu# kubectl config view --minify | grep namespace
    namespace: test
root@k8m:/home/ubuntu#
```

- Verify that default namespace has changed to our namespace.

```
kubectl get pods
```



```
root@k8m:/home/ubuntu# kubectl get pods
No resources found in test namespace.
root@k8m:/home/ubuntu#
```

- Namespace has been created and assigned as default namespace.

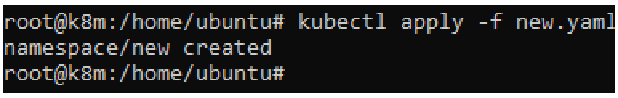
4) Create a namespace and allocate resources to that namespace.

- Create the yaml file for creating the namespace.

```
cat>new.yaml
apiVersion: v1
kind: Namespace
metadata:
 name: new
```

- Creating the namespace using the below command.

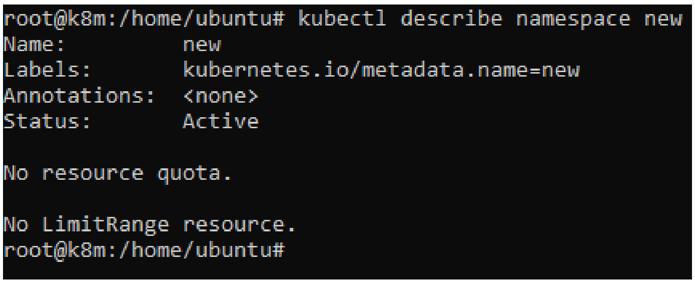
```
kubectl apply -f new.yaml
```



```
root@k8m:/home/ubuntu# kubectl apply -f new.yaml
namespace/new created
root@k8m:/home/ubuntu#
```

- Now we will verify if there is any quota or limit assigned wrt to resources on the created namespace “new”

  kubectl describe namespace new



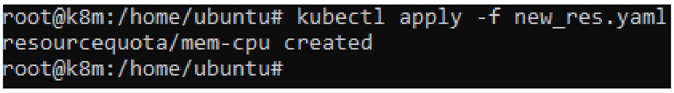
```
root@k8m:/home/ubuntu# kubectl describe namespace new
Name:         new
Labels:       kubernetes.io/metadata.name=new
Annotations:  <none>
Status:       Active

No resource quota.

No LimitRange resource.
root@k8m:/home/ubuntu#
```

  Nothing is assigned as of now

- Defining resources to the existing created namespace “new”.

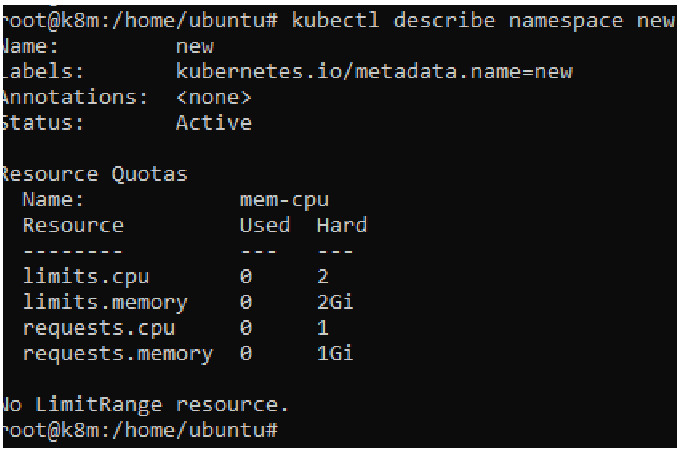
```
cat > new_res.yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: mem-cpu
  namespace: new
spec:
 hard:
  requests.cpu: "1"
  requests.memory: 1Gi
  limits.cpu: "2"
  limits.memory: 2Gi
```

- Creating the resource quota and applying it using the below command.
  kubectl apply -f new_res.yaml



```
root@k8m:/home/ubuntu# kubectl apply -f new_res.yaml
resourcequota/mem-cpu created
root@k8m:/home/ubuntu#
```

- Verify whether the quota and limits are assigned to the respected namespace “new” kubectl describe namespace new



```
root@k8m:/home/ubuntu# kubectl describe namespace new
Name:         new
Labels:       kubernetes.io/metadata.name=new
Annotations:  <none>
Status:       Active

Resource Quotas
  Name:            mem-cpu
  Resource         Used  Hard
  --------         ---   ---
  limits.cpu       0     2
  limits.memory    0     2Gi
  requests.cpu     0     1
  requests.memory  0     1Gi

No LimitRange resource.
root@k8m:/home/ubuntu#
```

➢ The resource quota and limits have been assigned successfully.

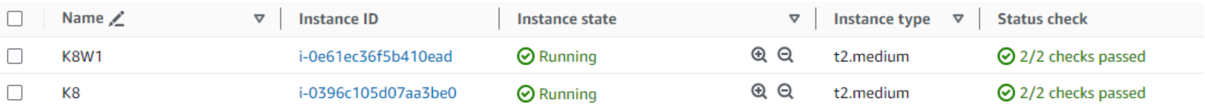
5) Install an old version of Kubernetes and upgrade it to latest version.

We will be performing upgrade of Kubernetes from v1.30 to latest v1.31

To perform the installation of v1.30 we will need the following prerequisites

Pre-requisites

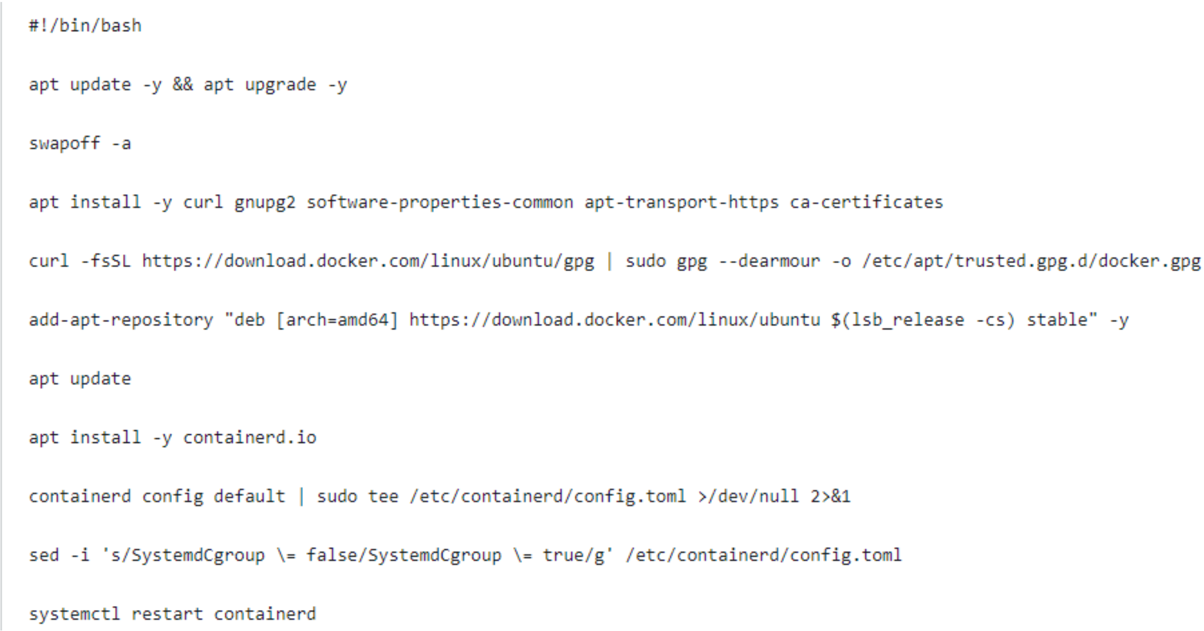
- **Launch 2 EC2 instance with at least t2.medium configuration. One is Control plane and other is Worker Node.**



□	Name	Instance ID	Instance state	Instance type	Status check
□	K8W1	i-0e61ec36f5b410ead	✓ Running	t2.medium	✓ 2/2 checks passed
□	K8	i-0396c105d07aa3be0	✓ Running	t2.medium	✓ 2/2 checks passed

- **Setting up containerd on Master Node (Control Plane K8) and perform Kubernetes Installation manually.**

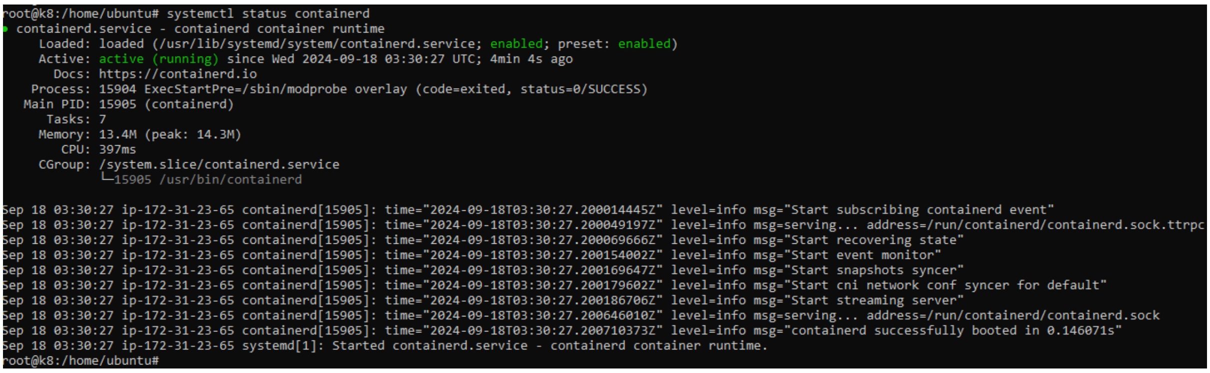
We will setup container id by putting the below commands in user data of EC2 for control plane server.



```
#!/bin/bash

apt update -y && apt upgrade -y

swapoff -a

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

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg

add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" -y

apt update

apt install -y containerd.io

containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1

sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

systemctl restart containerd
```

After the above steps are done, we will verify the containerd status on the EC2 instance.

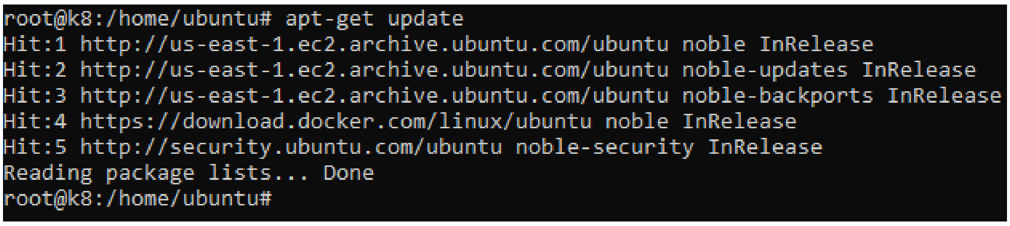
systemctl status containerd



```
root@k8:/home/ubuntu# systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Wed 2024-09-18 03:30:27 UTC; 4min 4s ago
       Docs: https://containerd.io
    Process: 15904 ExecStartPre=/sbin/modprobe overlay (code=exited, status=0/SUCCESS)
   Main PID: 15905 (containerd)
      Tasks: 7
     Memory: 13.4M (peak: 14.3M)
        CPU: 397ms
     CGroup: /system.slice/containerd.service
             └─15905 /usr/bin/containerd

Sep 18 03:30:27 ip-172-31-23-65 containerd[15905]: time="2024-09-18T03:30:27.200014445Z" level=info msg="Start subscribing containerd event"
Sep 18 03:30:27 ip-172-31-23-65 containerd[15905]: time="2024-09-18T03:30:27.200049197Z" level=info msg=serving... address=/run/containerd/containerd.sock.ttrpc
Sep 18 03:30:27 ip-172-31-23-65 containerd[15905]: time="2024-09-18T03:30:27.200069666Z" level=info msg="Start recovering state"
Sep 18 03:30:27 ip-172-31-23-65 containerd[15905]: time="2024-09-18T03:30:27.200154002Z" level=info msg="Start event monitor"
Sep 18 03:30:27 ip-172-31-23-65 containerd[15905]: time="2024-09-18T03:30:27.200169647Z" level=info msg="Start snapshots syncer"
Sep 18 03:30:27 ip-172-31-23-65 containerd[15905]: time="2024-09-18T03:30:27.200179602Z" level=info msg="Start cni network conf syncer for default"
Sep 18 03:30:27 ip-172-31-23-65 containerd[15905]: time="2024-09-18T03:30:27.200186706Z" level=info msg="Start streaming server"
Sep 18 03:30:27 ip-172-31-23-65 containerd[15905]: time="2024-09-18T03:30:27.200646010Z" level=info msg=serving... address=/run/containerd/containerd.sock
Sep 18 03:30:27 ip-172-31-23-65 containerd[15905]: time="2024-09-18T03:30:27.200710373Z" level=info msg="containerd successfully booted in 0.146071s"
Sep 18 03:30:27 ip-172-31-23-65 systemd[1]: Started containerd.service - containerd container runtime.
root@k8:/home/ubuntu#
```

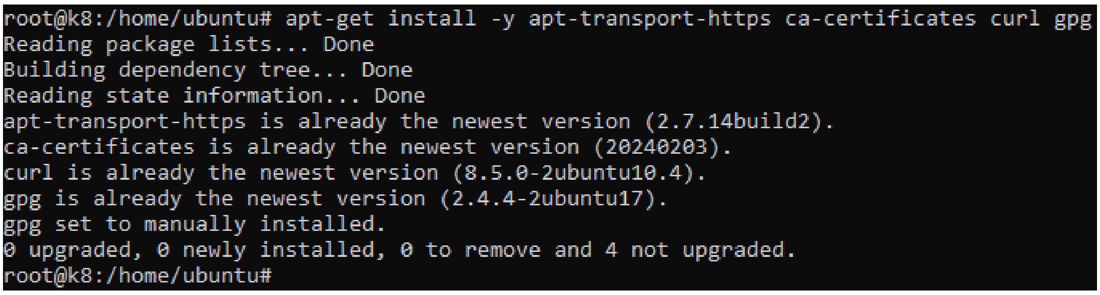
Now we will proceed with the installation of Kubernetes v1.30

apt-get update (Command updates the local package database)



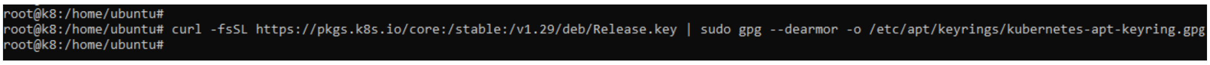
```
root@k8:/home/ubuntu# apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
root@k8:/home/ubuntu#
```

apt-get install -y apt-transport-https ca-certificates curl gpg (Already installed before installing containerId)



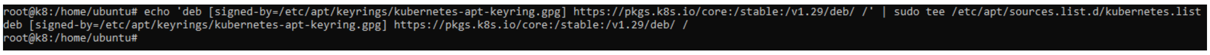
```
root@k8:/home/ubuntu# apt-get install -y apt-transport-https ca-certificates curl gpg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apt-transport-https is already the newest version (2.7.14build2).
ca-certificates is already the newest version (20240203).
curl is already the newest version (8.5.0-2ubuntu10.4).
gpg is already the newest version (2.4.4-2ubuntu17).
gpg set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
root@k8:/home/ubuntu#
```

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg (Command fetches the key for the repository)



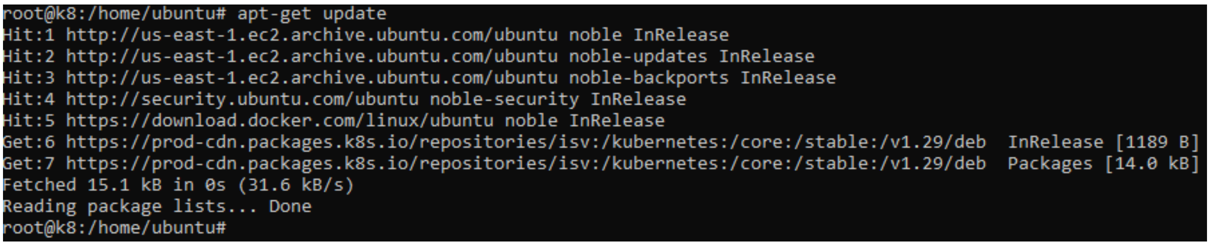
```
root@k8:/home/ubuntu#
root@k8:/home/ubuntu# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
root@k8:/home/ubuntu#
```

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list (Command overwrites any existing configuration in /etc/apt/sources.list.d/kubernetes.list)



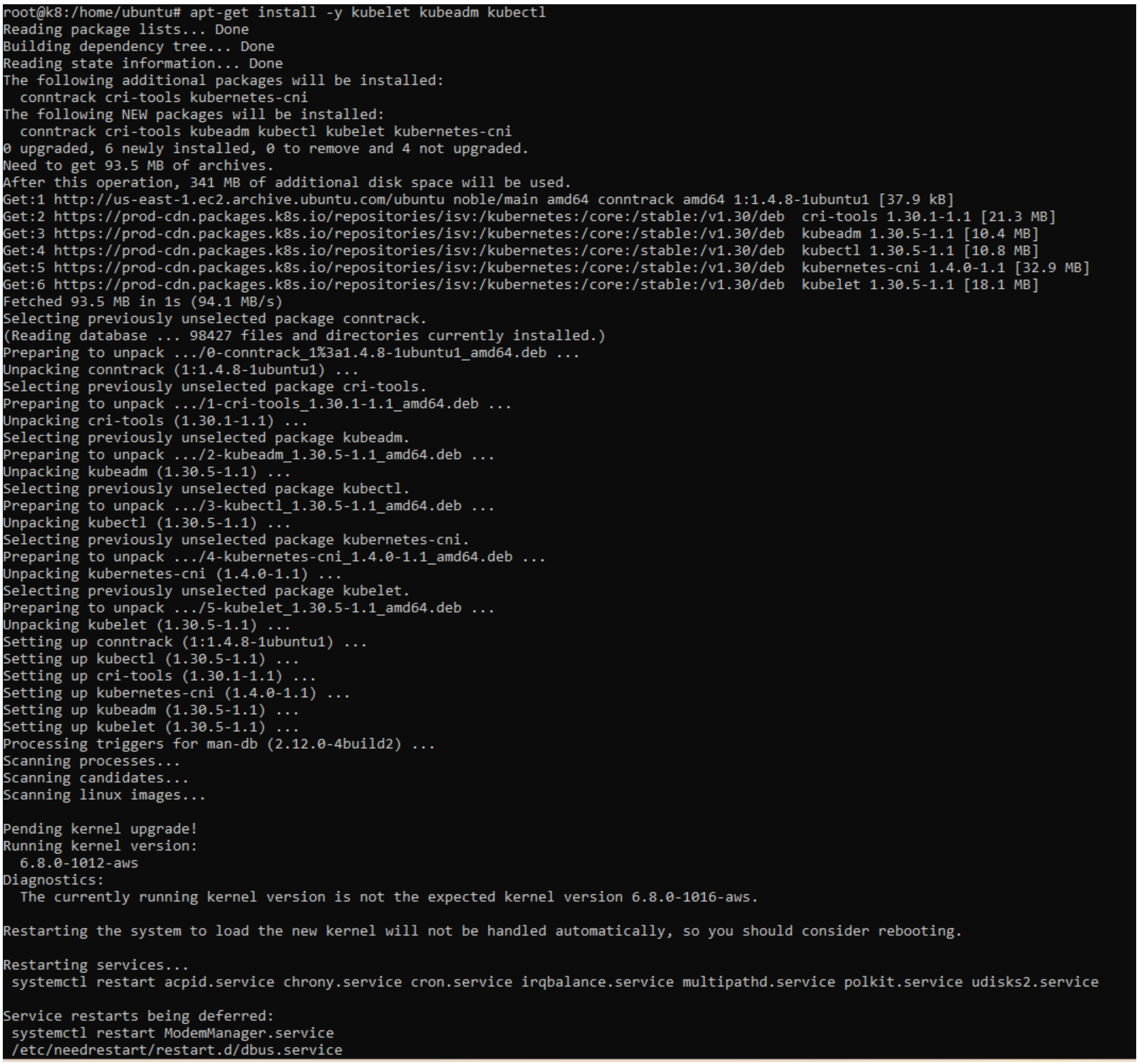
```
root@k8:/home/ubuntu# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /
root@k8:/home/ubuntu#
```

apt-get update (Updates the package index)



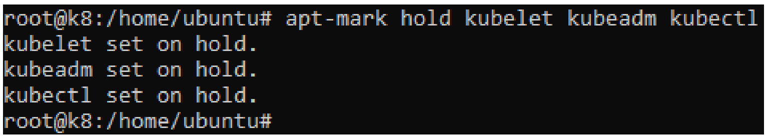
```
root@k8:/home/ubuntu# apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  InRelease [1189 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  Packages [14.0 kB]
Fetched 15.1 kB in 0s (31.6 kB/s)
Reading package lists... Done
root@k8:/home/ubuntu#
```

apt-get install -y kubelet kubeadm kubectl (Command to install kubelet, kubeadm and kubectl)



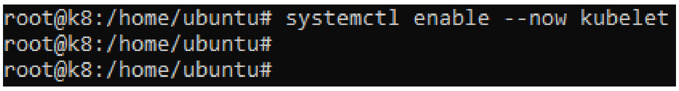
```
root@k8:/home/ubuntu# apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 4 not upgraded.
Need to get 93.5 MB of archives.
After this operation, 341 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  cri-tools 1.30.1-1.1 [21.3 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  kubeadm 1.30.5-1.1 [10.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  kubectl 1.30.5-1.1 [10.8 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  kubernetes-cni 1.4.0-1.1 [32.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  kubelet 1.30.5-1.1 [18.1 MB]
Fetched 93.5 MB in 1s (94.1 MB/s)
Selecting previously unselected package conntrack.
(Reading database ... 98427 files and directories currently installed.)
Preparing to unpack .../0-conntrack_1%3a1.4.8-1ubuntu1_amd64.deb ...
Unpacking conntrack (1:1.4.8-1ubuntu1) ...
Selecting previously unselected package cri-tools.
Preparing to unpack .../1-cri-tools_1.30.1-1.1_amd64.deb ...
Unpacking cri-tools (1.30.1-1.1) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../2-kubeadm_1.30.5-1.1_amd64.deb ...
Unpacking kubeadm (1.30.5-1.1) ...
Selecting previously unselected package kubectl.
Preparing to unpack .../3-kubectl_1.30.5-1.1_amd64.deb ...
Unpacking kubectl (1.30.5-1.1) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../4-kubernetes-cni_1.4.0-1.1_amd64.deb ...
Unpacking kubernetes-cni (1.4.0-1.1) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.30.5-1.1_amd64.deb ...
Unpacking kubelet (1.30.5-1.1) ...
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Setting up kubectl (1.30.5-1.1) ...
Setting up cri-tools (1.30.1-1.1) ...
Setting up kubernetes-cni (1.4.0-1.1) ...
Setting up kubeadm (1.30.5-1.1) ...
Setting up kubelet (1.30.5-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.8.0-1012-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-1016-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...
 systemctl restart acpid.service chrony.service cron.service irqbalance.service multipathd.service polkit.service udisks2.service

Service restarts being deferred:
 systemctl restart ModemManager.service
 /etc/needrestart/restart.d/dbus.service
```

apt-mark hold kubelet kubeadm kubectl (Command prevents a specific package from being updated due to compatibility issues or other reasons. It ensures that the package remains at its current version until explicitly unheld.)



```
root@k8:/home/ubuntu# apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@k8:/home/ubuntu#
```

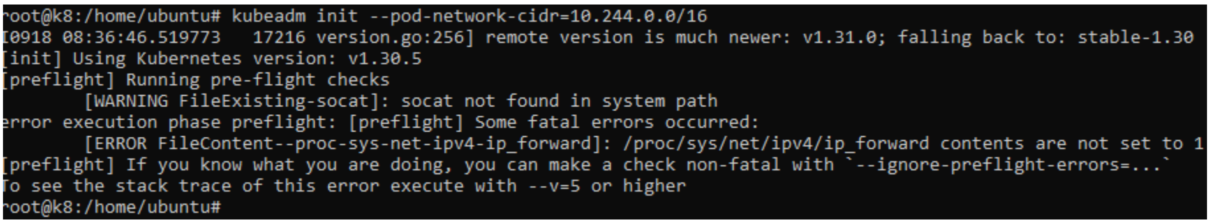
systemctl enable --now kubelet (Enable the kubelet service before running kubeadm)



```
root@k8:/home/ubuntu# systemctl enable --now kubelet
root@k8:/home/ubuntu#
root@k8:/home/ubuntu#
```

kubeadm init --pod-network-cidr=10.244.0.0/16 (Command Initializes the Kubernetes setup)

It gives the below error

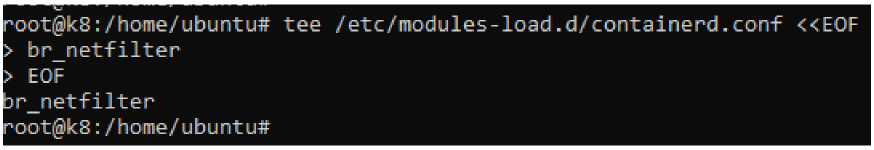


```
root@k8:/home/ubuntu# kubeadm init --pod-network-cidr=10.244.0.0/16
I0918 08:36:46.519773   17216 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.5
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
root@k8:/home/ubuntu#
```

We need to perform the below troubleshooting to initialize the kubeadm by freeing up ram and reinitialise iptables settings

- Configure the Kernel Module ‘br_netfilter’ in the containerd configuration file.

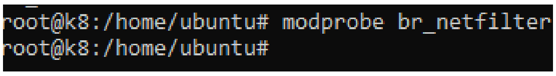
  tee /etc/modules-load.d/containerd.conf <<EOF
  br_netfilter
  EOF



```
root@k8:/home/ubuntu# tee /etc/modules-load.d/containerd.conf <<EOF
> br_netfilter
> EOF
br_netfilter
root@k8:/home/ubuntu#
```

- Load the br_netfilter modules into the running Linux kernel.

modprobe br_netfilter

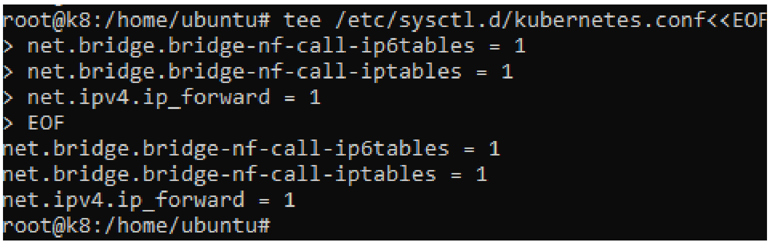


```
root@k8:/home/ubuntu# modprobe br_netfilter
root@k8:/home/ubuntu#
```

- Update Iptables Settings.

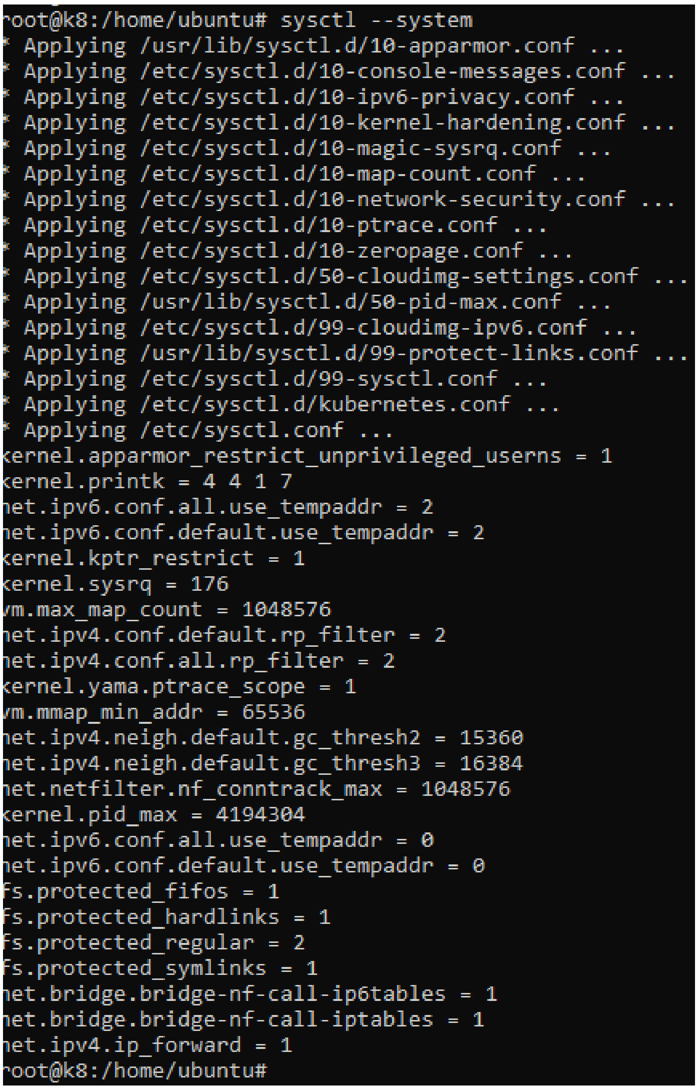
Note: To ensure packets are properly processed by IP tables during filtering and port forwarding, set the **net.bridge.bridge-nf-call-iptables to '1'** in your sysctl configuration file. Otherwise, you may encounter the following error: **[ERROR FileContent–proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1.** To avoid this, execute the following command.

```
tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```



```
root@k8:/home/ubuntu# tee /etc/sysctl.d/kubernetes.conf<<EOF
> net.bridge.bridge-nf-call-ip6tables = 1
> net.bridge.bridge-nf-call-iptables = 1
> net.ipv4.ip_forward = 1
> EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
root@k8:/home/ubuntu#
```

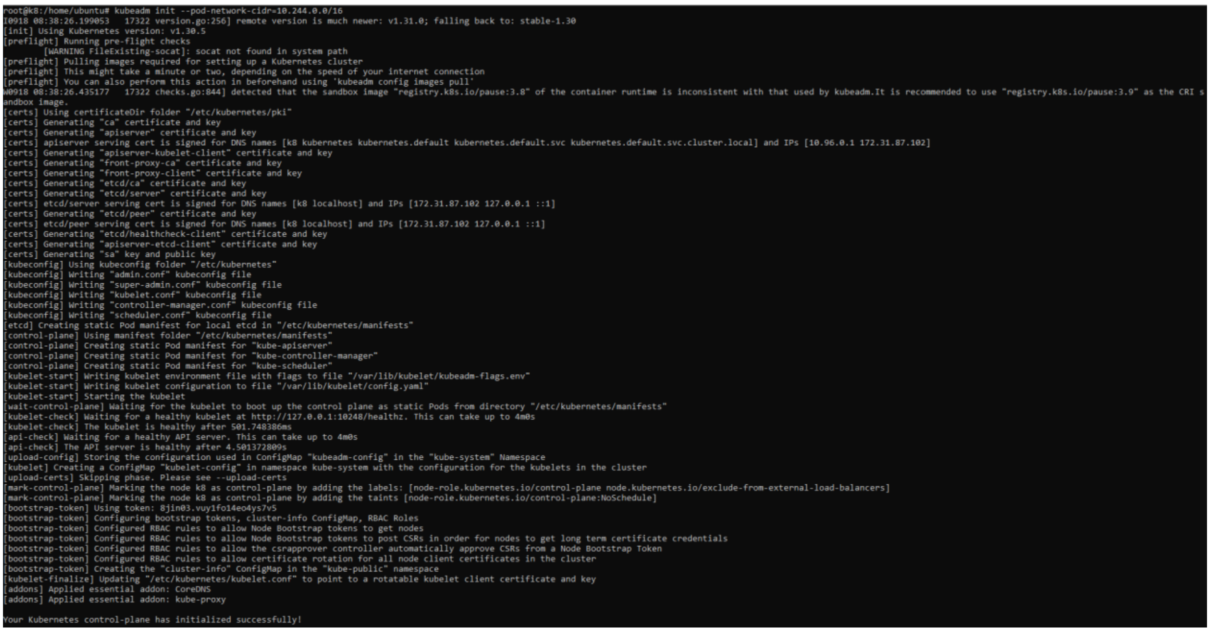
sysctl --system (Command applies kernel settings without reboot)



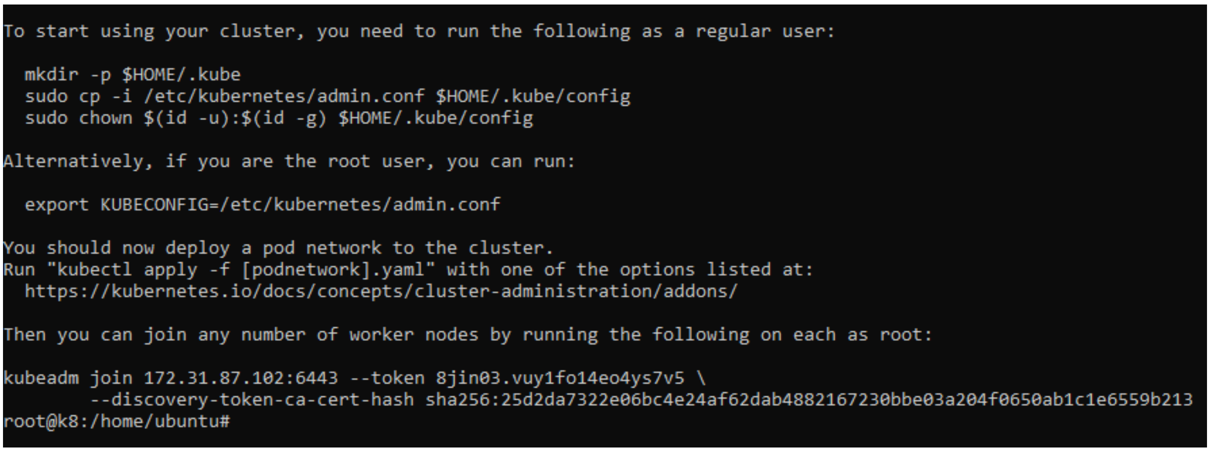
```
root@k8:/home/ubuntu# sysctl --system
* Applying /usr/lib/sysctl.d/10-apparmor.conf ...
* Applying /etc/sysctl.d/10-console-messages.conf ...
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
* Applying /etc/sysctl.d/10-map-count.conf ...
* Applying /etc/sysctl.d/10-network-security.conf ...
* Applying /etc/sysctl.d/10-ptrace.conf ...
* Applying /etc/sysctl.d/10-zeropage.conf ...
* Applying /etc/sysctl.d/50-cloudimg-settings.conf ...
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
* Applying /etc/sysctl.d/99-cloudimg-ipv6.conf ...
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/kubernetes.conf ...
* Applying /etc/sysctl.conf ...
kernel.apparmor_restrict_unprivileged_userns = 1
kernel.printk = 4 4 1 7
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
kernel.kptr_restrict = 1
kernel.sysrq = 176
vm.max_map_count = 1048576
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.all.rp_filter = 2
kernel.yama.ptrace_scope = 1
vm.mmap_min_addr = 65536
net.ipv4.neigh.default.gc_thresh2 = 15360
net.ipv4.neigh.default.gc_thresh3 = 16384
net.netfilter.nf_conntrack_max = 1048576
kernel.pid_max = 4194304
net.ipv6.conf.all.use_tempaddr = 0
net.ipv6.conf.default.use_tempaddr = 0
fs.protected_fifos = 1
fs.protected_hardlinks = 1
fs.protected_regular = 2
fs.protected_symlinks = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
root@k8:/home/ubuntu#
```

Once you've verified and potentially adjusted the configuration, proceed with reinitializing the Kubernetes cluster. we initialized the kubeadm again and installation was successful.

kubeadm init --pod-network-cidr=10.244.0.0/16 (Command Initializes the Kubernetes setup)



```
root@k8:/home/ubuntu# kubeadm init --pod-network-cidr=10.244.0.0/16
I0918 08:38:26.199053   17322 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.5
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
[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'
W0918 08:38:26.435177   17322 checks.go:844] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.9" as the CRI s
andbox 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 [k8 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.87.102]
[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 [k8 localhost] and IPs [172.31.87.102 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [k8 localhost] and IPs [172.31.87.102 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[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"
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.748386ms
[api-check] Waiting for a healthy API server. This can take up to 4m0s
[api-check] The API server is healthy after 4.501372809s
[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 k8 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 k8 as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: 8jin03.vuy1fo14eo4ys7v5
[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!
```

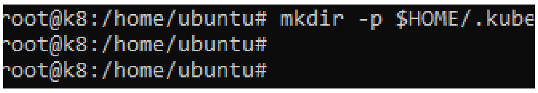


```
To start using your cluster, you need to run the following as a regular user:

  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/

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

kubeadm join 172.31.87.102:6443 --token 8jin03.vuy1fo14eo4ys7v5 \
        --discovery-token-ca-cert-hash sha256:25d2da7322e06bc4e24af62dab4882167230bbe03a204f0650ab1c1e6559b213
root@k8:/home/ubuntu#
```

We can also see the above screenshot that the token has been generated to join the Worker Nodes to control plane. We need to run the command on both the worker nodes.

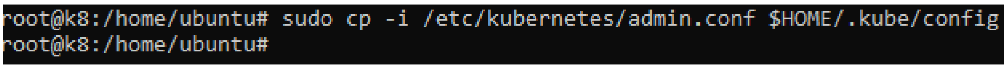
Since Kubernetes control-plane has initialized successfully, we need to run the below commands to start using the cluster.

mkdir -p $HOME/.kube



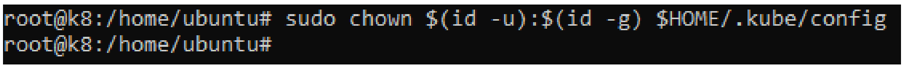
```
root@k8:/home/ubuntu# mkdir -p $HOME/.kube
root@k8:/home/ubuntu#
root@k8:/home/ubuntu#
```

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



```
root@k8:/home/ubuntu# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@k8:/home/ubuntu#
```

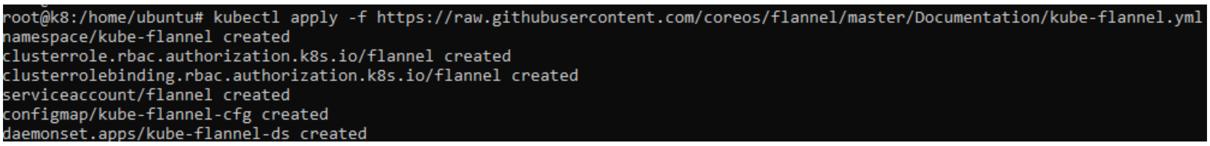
sudo chown $(id -u):$(id -g) $HOME/.kube/config



```
root@k8:/home/ubuntu# sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@k8:/home/ubuntu#
```

We also need to ready our cluster, for that we need to assign a flannel network using the kube-flannel.yml file by running the below command.

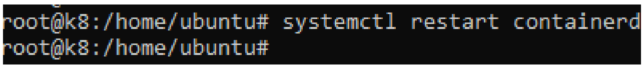
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml



```
root@k8:/home/ubuntu# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Restart the containerd service

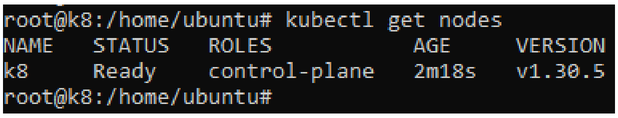
systemctl restart containerd



```
root@k8:/home/ubuntu# systemctl restart containerd
root@k8:/home/ubuntu#
```

Now we will whether our Control plane node is ready on the version 1.30

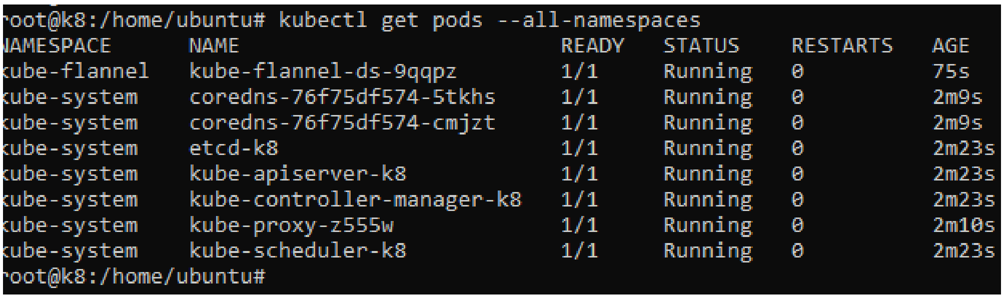
kubectl get nodes



```
root@k8:/home/ubuntu# kubectl get nodes
NAME   STATUS   ROLES           AGE     VERSION
k8     Ready    control-plane   2m18s   v1.30.5
root@k8:/home/ubuntu#
```

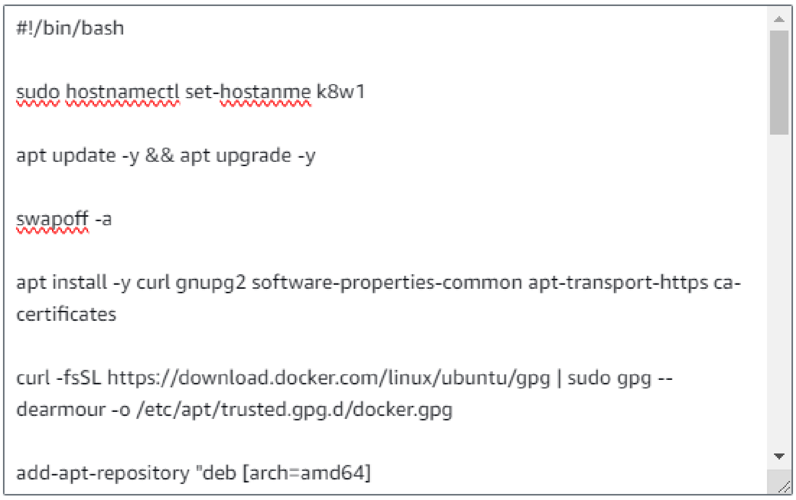
We will verify all the pods also in all namespaces are running fine or not

kubectl get pods --all-namespaces

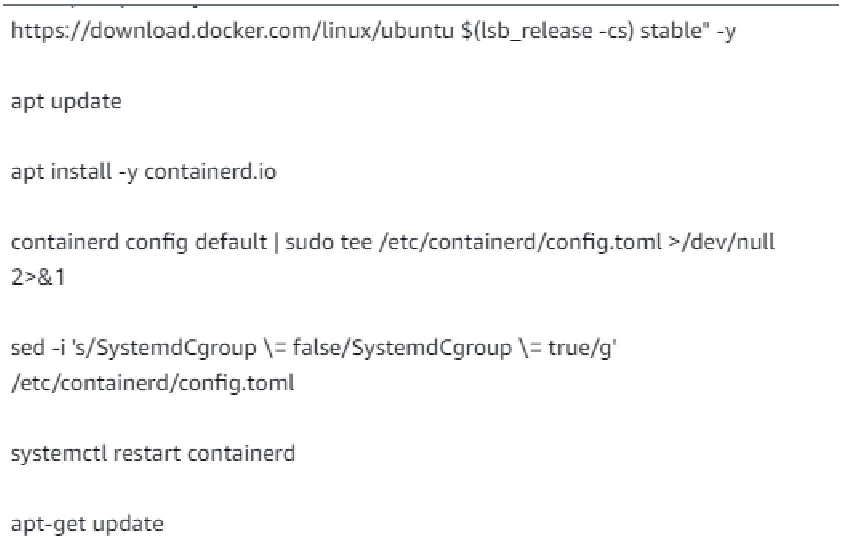


```
root@k8:/home/ubuntu# kubectl get pods --all-namespaces
NAMESPACE      NAME                             READY   STATUS    RESTARTS   AGE
kube-flannel   kube-flannel-ds-9qqpz            1/1     Running   0          75s
kube-system    coredns-76f75df574-5tkhs         1/1     Running   0          2m9s
kube-system    coredns-76f75df574-cmjzt         1/1     Running   0          2m9s
kube-system    etcd-k8                          1/1     Running   0          2m23s
kube-system    kube-apiserver-k8                1/1     Running   0          2m23s
kube-system    kube-controller-manager-k8       1/1     Running   0          2m23s
kube-system    kube-proxy-z555w                 1/1     Running   0          2m10s
kube-system    kube-scheduler-k8                1/1     Running   0          2m23s
root@k8:/home/ubuntu#
```

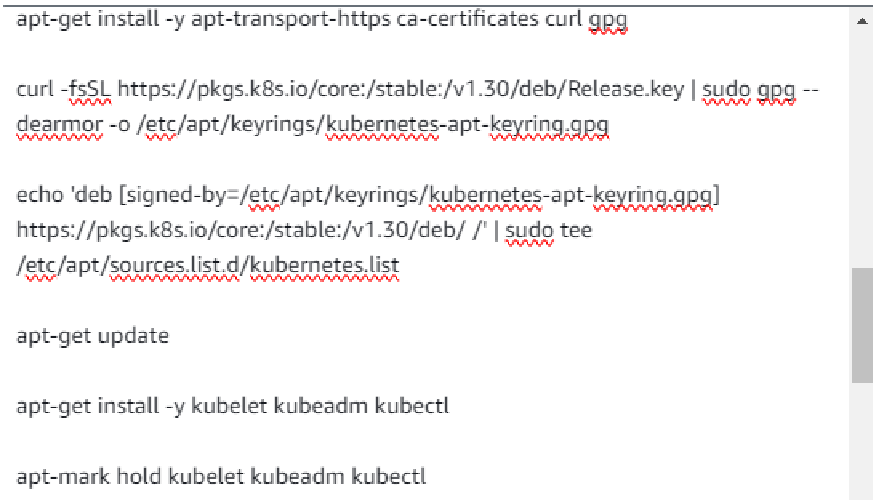
Now we will configure the worker node with the version v1.30 putting all the required commands in user data in the EC2 instance for Worker 1.



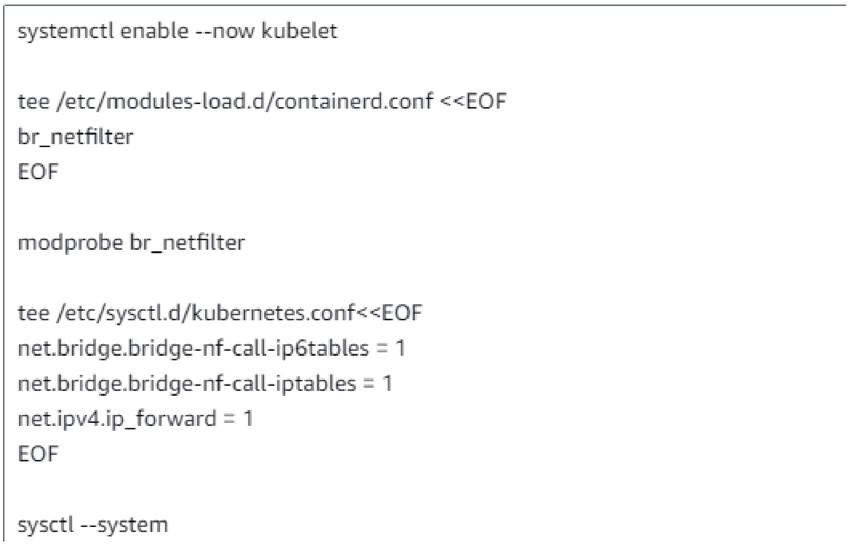
```
#!/bin/bash

sudo hostnamectl set-hostanme k8w1

apt update -y && apt upgrade -y

swapoff -a

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

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg

add-apt-repository "deb [arch=amd64]
```



```
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" -y

apt update

apt install -y containerd.io

containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1

sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

systemctl restart containerd

apt-get update
```



```
apt-get install -y apt-transport-https ca-certificates curl gpg

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

apt-get update

apt-get install -y kubelet kubeadm kubectl

apt-mark hold kubelet kubeadm kubectl
```



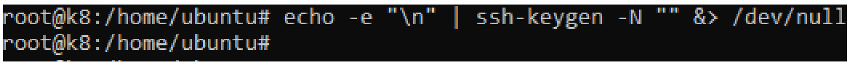
```
systemctl enable --now kubelet

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

modprobe br_netfilter

tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF

sysctl --system
```

Same steps have been performed successfully in Worker 1 till kubeadm init

Now we will setup password less SSH between Control plane and Worker Node.

- Command runs SSH-Keygen without prompting anything

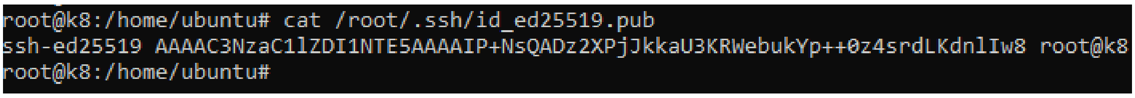
  echo -e "\n" | ssh-keygen -N "" &> /dev/null



```
root@k8:/home/ubuntu# echo -e "\n" | ssh-keygen -N "" &> /dev/null
root@k8:/home/ubuntu#
```

- Command to check whether the pub file is created in the below location along with the required contents.

  cat /root/.ssh/id_ed25519.pub



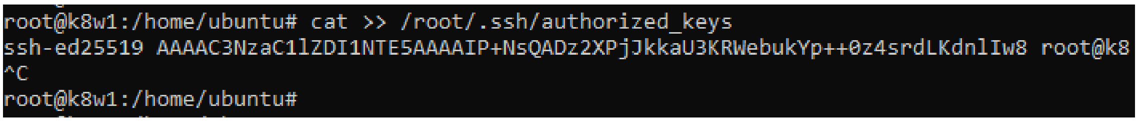
```
root@k8:/home/ubuntu# cat /root/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP+NsQADz2XPjJkkaU3KRWebukYp++0z4srdLKdnlIw8 root@k8
root@k8:/home/ubuntu#
```

Its shows that the public is created.

- Now copy the contents over the Worker Node 1 and Worker Node 2

Worker Node 1

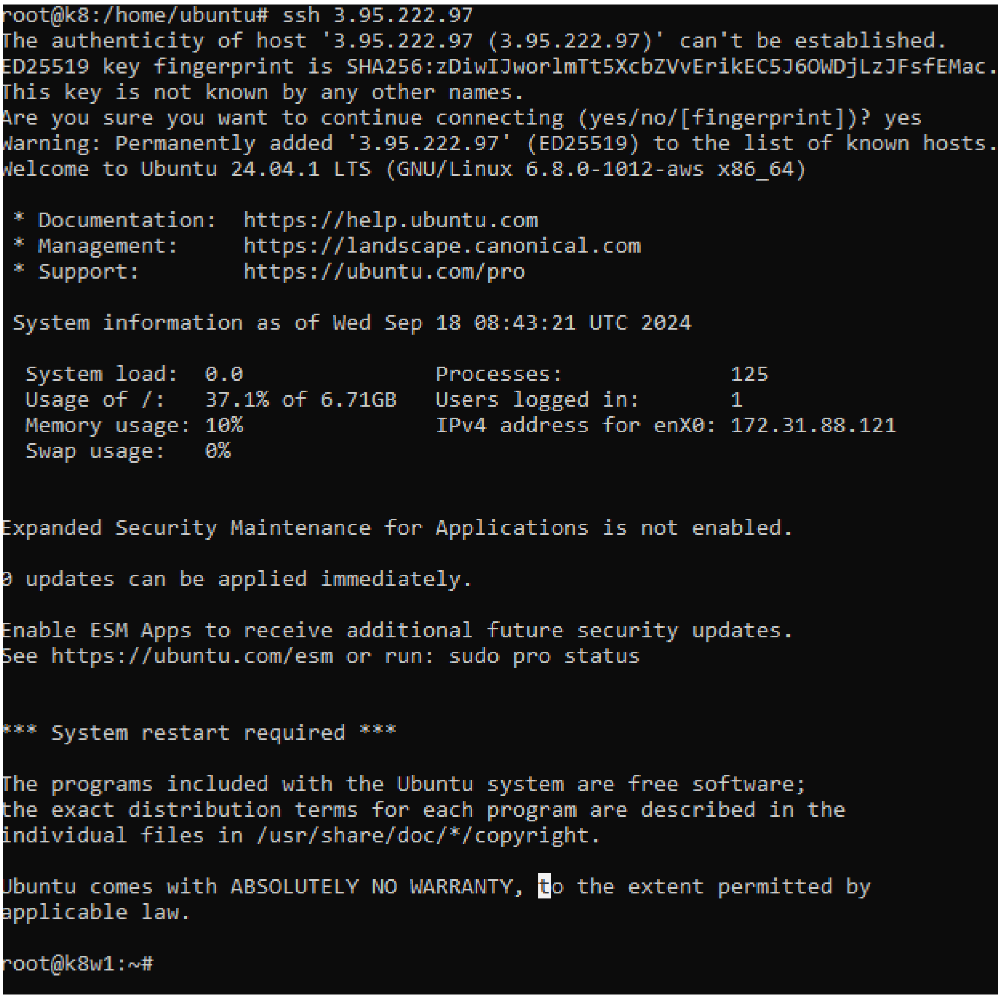
cat >> /root/.ssh/authorized_keys



```
root@k8w1:/home/ubuntu# cat >> /root/.ssh/authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP+NsQADz2XPjJkkaU3KRWebukYp++0z4srdLKdnlIw8 root@k8
^C
root@k8w1:/home/ubuntu#
```

Verify SSH from Control Plane

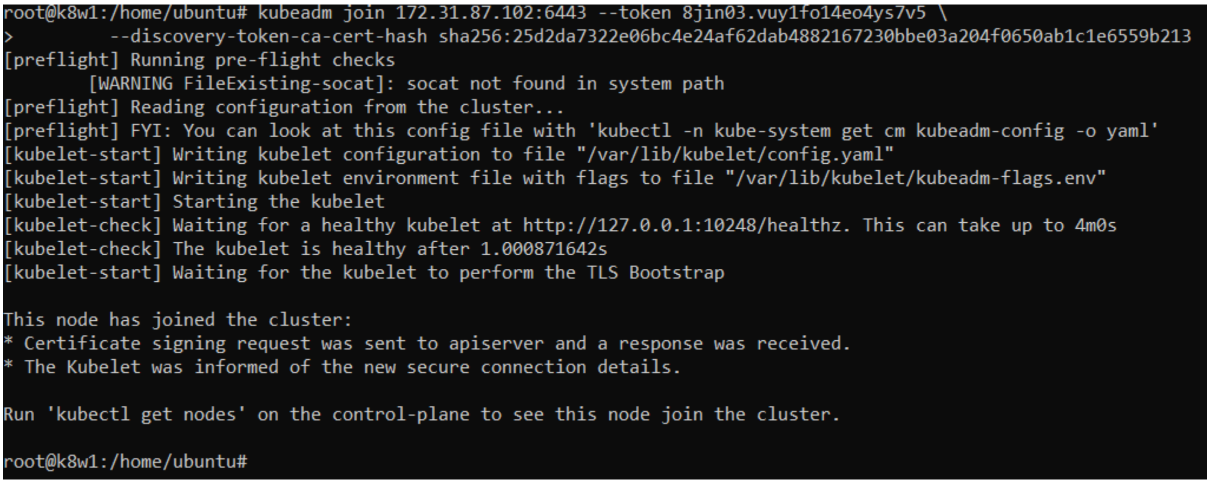
ssh Public IP of Worker 1



```
root@k8:/home/ubuntu# ssh 3.95.222.97
The authenticity of host '3.95.222.97 (3.95.222.97)' can't be established.
ED25519 key fingerprint is SHA256:zDiwIJworlmTt5XcbZVvErikEC5J6OWDjLzJFsfEMac.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.95.222.97' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Wed Sep 18 08:43:21 UTC 2024

  System load:  0.0               Processes:             125
  Usage of /:   37.1% of 6.71GB   Users logged in:       1
  Memory usage: 10%               IPv4 address for enX0: 172.31.88.121
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@k8w1:~#
```

Now we will perform the step of joining the worker nodes to cluster using the token from Control plane (K8)

Worker Node 1

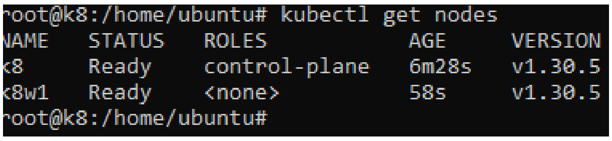


```
root@k8w1:/home/ubuntu# kubeadm join 172.31.87.102:6443 --token 8jin03.vuy1fo14eo4ys7v5 \
>         --discovery-token-ca-cert-hash sha256:25d2da7322e06bc4e24af62dab4882167230bbe03a204f0650ab1c1e6559b213
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
[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-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.000871642s
[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@k8w1:/home/ubuntu#
```

Successfully joined the Cluster.

Verify that the worker node joined the cluster.

kubectl get nodes



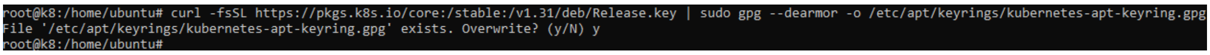
```
root@k8:/home/ubuntu# kubectl get nodes
NAME   STATUS   ROLES           AGE     VERSION
k8     Ready    control-plane   6m28s   v1.30.5
k8w1   Ready    <none>          58s     v1.30.5
root@k8:/home/ubuntu#
```

Now we will start the upgrade process from v1.30 to v1.31

Upgrading Control Plane Node

We will overwrite the key string by replacing it with the version we want to upgrade in our case we will take key string for v1.31 so that we can get the packages of that version

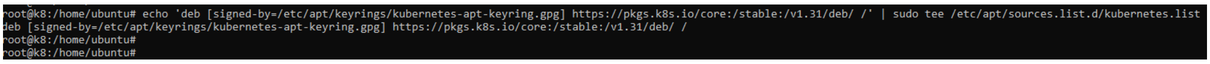
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg



```
root@k8:/home/ubuntu# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
File '/etc/apt/keyrings/kubernetes-apt-keyring.gpg' exists. Overwrite? (y/N) y
root@k8:/home/ubuntu#
```

Now we will overwrite the existing configuration with that of version 1.31

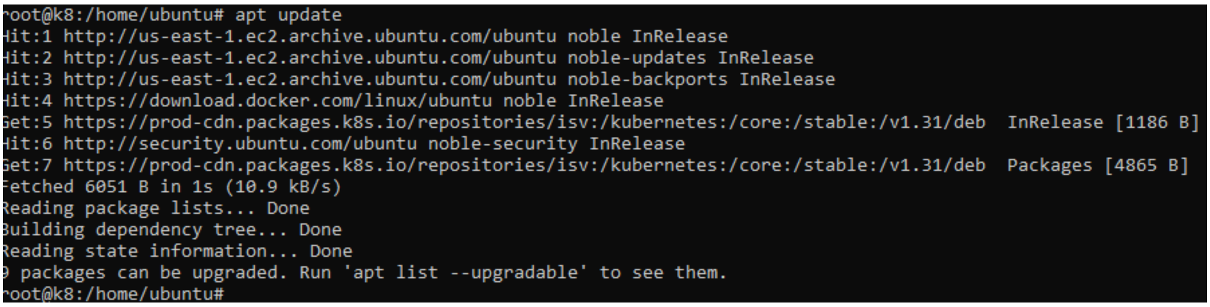
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list



```
root@k8:/home/ubuntu# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
root@k8:/home/ubuntu#
root@k8:/home/ubuntu#
```

Then we need to update the package index to v 1.31

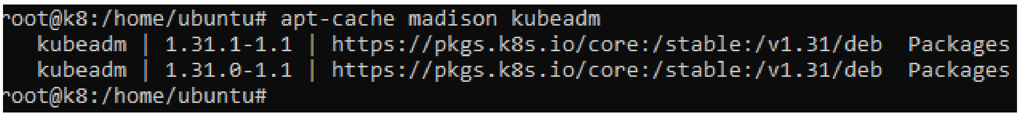
apt-get update



```
root@k8:/home/ubuntu# apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (10.9 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
9 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@k8:/home/ubuntu#
```

We can check for the available package versions on which we can update.

apt-cache madison kubeadm



```
root@k8:/home/ubuntu# apt-cache madison kubeadm
   kubeadm | 1.31.1-1.1 | https://pkgs.k8s.io/core:/stable:/v1.31/deb  Packages
   kubeadm | 1.31.0-1.1 | https://pkgs.k8s.io/core:/stable:/v1.31/deb  Packages
root@k8:/home/ubuntu#
```

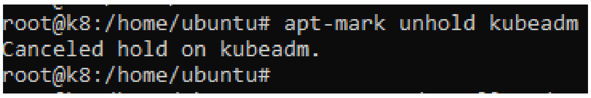
These are components we need to upgraded for Kubernetes Control plane.

- Kubeadm
- Cluster
- Kubectl and kubelet

We are upgrading the kubeadm on control plane by following the below steps

Now we need to remove the hold on the kubeadm, so that we can upgrade it to the latest version v 1.31

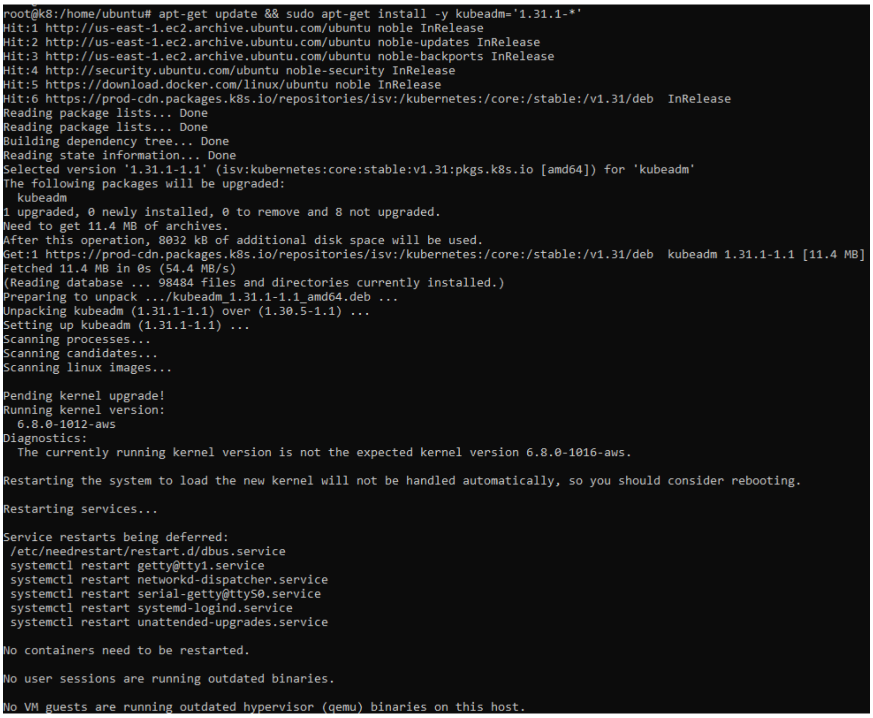
apt-mark unhold kubeadm



```
root@k8:/home/ubuntu# apt-mark unhold kubeadm
Canceled hold on kubeadm.
root@k8:/home/ubuntu#
```

Now we will install the kubeadm package for v 1.31

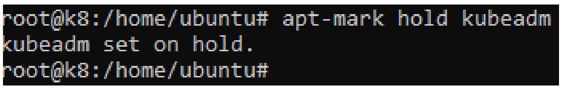
apt-get update && sudo apt-get install -y kubeadm='1.31.1-*'



```
root@k8:/home/ubuntu# apt-get update && sudo apt-get install -y kubeadm='1.31.1-*'
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Selected version '1.31.1-1.1' (isv:kubernetes:core:stable:v1.31:pkgs.k8s.io [amd64]) for 'kubeadm'
The following packages will be upgraded:
  kubeadm
1 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
Need to get 11.4 MB of archives.
After this operation, 8032 kB of additional disk space will be used.
Get:1 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Fetched 11.4 MB in 0s (54.4 MB/s)
(Reading database ... 98484 files and directories currently installed.)
Preparing to unpack .../kubeadm_1.31.1-1.1_amd64.deb ...
Unpacking kubeadm (1.31.1-1.1) over (1.30.5-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.8.0-1012-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-1016-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...

Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart serial-getty@ttyS0.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service

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.
```

Now we need to put back the hold on the kubeadm as version update is done.

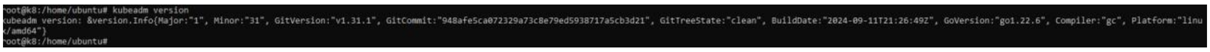
apt-mark hold kubeadm



```
root@k8:/home/ubuntu# apt-mark hold kubeadm
kubeadm set on hold.
root@k8:/home/ubuntu#
```

We will verify the kubeadm version

kubeadm version

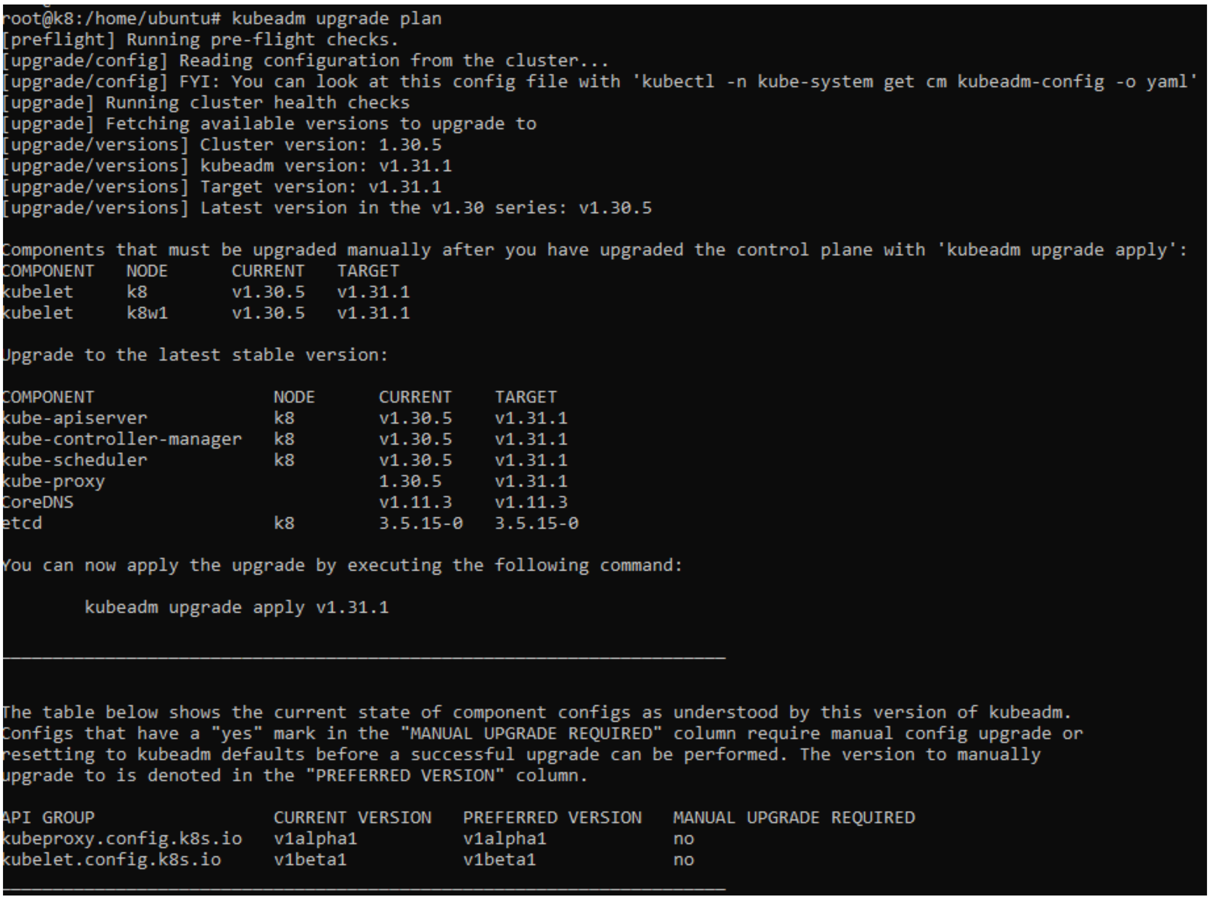


```
root@k8:/home/ubuntu# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"31", GitVersion:"v1.31.1", GitCommit:"948afe5ca072329a73c8e79ed5938717a5cb3d21", GitTreeState:"clean", BuildDate:"2024-09-11T21:26:49Z", GoVersion:"go1.22.6", Compiler:"gc", Platform:"linu
x/amd64"}
root@k8:/home/ubuntu#
```

We can see kubeadm is upgraded to v 1.31

Now we will upgrade the cluster and system pods by following the below steps

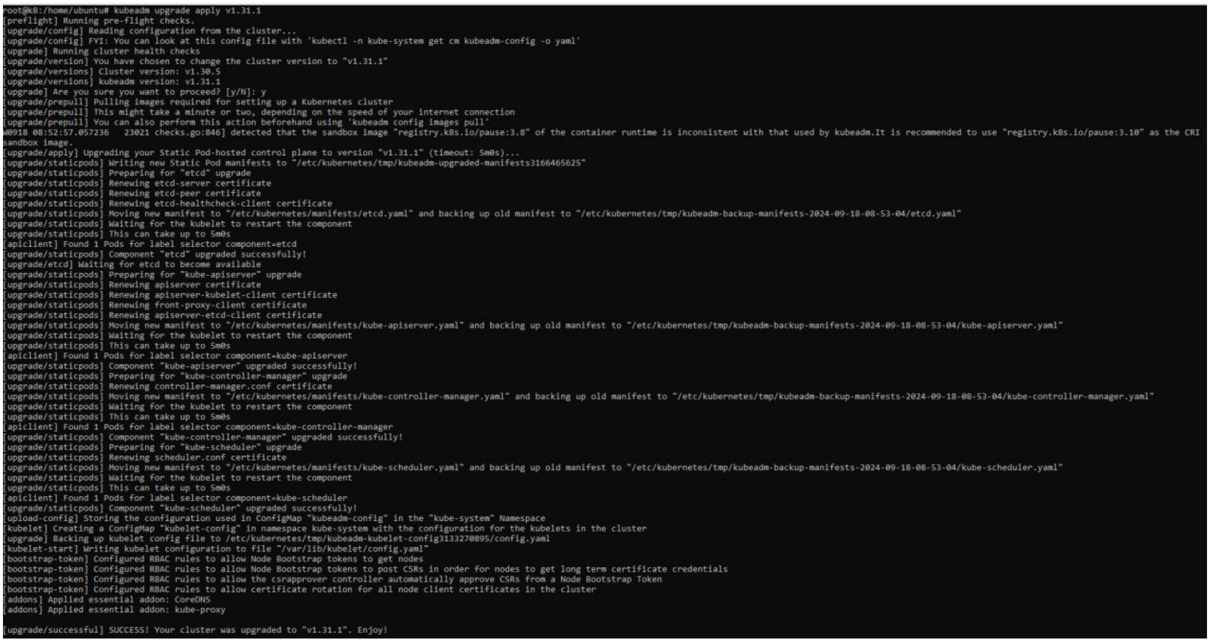
kubeadm upgrade plan (Command gives the layout of what is current version of the components and on which version they will be upgraded)



```
root@k8:/home/ubuntu# kubeadm upgrade plan
[preflight] Running pre-flight checks.
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[upgrade] Running cluster health checks
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: 1.30.5
[upgrade/versions] kubeadm version: v1.31.1
[upgrade/versions] Target version: v1.31.1
[upgrade/versions] Latest version in the v1.30 series: v1.30.5

Components that must be upgraded manually after you have upgraded the control plane with 'kubeadm upgrade apply':
COMPONENT   NODE      CURRENT    TARGET
kubelet     k8        v1.30.5    v1.31.1
kubelet     k8w1      v1.30.5    v1.31.1

Upgrade to the latest stable version:

COMPONENT                 NODE      CURRENT    TARGET
kube-apiserver            k8        v1.30.5    v1.31.1
kube-controller-manager   k8        v1.30.5    v1.31.1
kube-scheduler            k8        v1.30.5    v1.31.1
kube-proxy                          1.30.5     v1.31.1
CoreDNS                             v1.11.3    v1.11.3
etcd                      k8        3.5.15-0   3.5.15-0

You can now apply the upgrade by executing the following command:

        kubeadm upgrade apply v1.31.1

_____________________________________________________________________


The table below shows the current state of component configs as understood by this version of kubeadm.
Configs that have a "yes" mark in the "MANUAL UPGRADE REQUIRED" column require manual config upgrade or
resetting to kubeadm defaults before a successful upgrade can be performed. The version to manually
upgrade to is denoted in the "PREFERRED VERSION" column.

API GROUP                 CURRENT VERSION   PREFERRED VERSION   MANUAL UPGRADE REQUIRED
kubeproxy.config.k8s.io   v1alpha1          v1alpha1            no
kubelet.config.k8s.io     v1beta1           v1beta1             no
_____________________________________________________________________
```

Now we have applied the changes in the plan by using the below command.

kubeadm upgrade apply v1.31.1



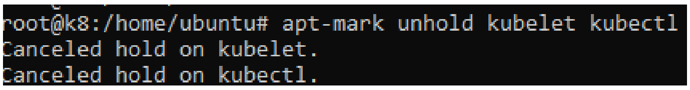
```
root@k8:/home/ubuntu# kubeadm upgrade apply v1.31.1
[preflight] Running pre-flight checks.
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[upgrade] Running cluster health checks
[upgrade/version] You have chosen to change the cluster version to "v1.31.1"
[upgrade/versions] Cluster version: v1.30.5
[upgrade/versions] kubeadm version: v1.31.1
[upgrade] Are you sure you want to proceed? [y/N]: y
[upgrade/prepull] Pulling images required for setting up a Kubernetes cluster
[upgrade/prepull] This might take a minute or two, depending on the speed of your internet connection
[upgrade/prepull] You can also perform this action beforehand using 'kubeadm config images pull'
W0918 08:52:57.057236   23021 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI
sandbox image.
[upgrade/apply] Upgrading your Static Pod-hosted control plane to version "v1.31.1" (timeout: 5m0s)...
[upgrade/staticpods] Writing new Static Pod manifests to "/etc/kubernetes/tmp/kubeadm-upgraded-manifests3166465625"
[upgrade/staticpods] Preparing for "etcd" upgrade
[upgrade/staticpods] Renewing etcd-server certificate
[upgrade/staticpods] Renewing etcd-peer certificate
[upgrade/staticpods] Renewing etcd-healthcheck-client certificate
[upgrade/staticpods] Moving new manifest to "/etc/kubernetes/manifests/etcd.yaml" and backing up old manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2024-09-18-08-53-04/etcd.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This can take up to 5m0s
[apiclient] Found 1 Pods for label selector component=etcd
[upgrade/staticpods] Component "etcd" upgraded successfully!
[upgrade/etcd] Waiting for etcd to become available
[upgrade/staticpods] Preparing for "kube-apiserver" upgrade
[upgrade/staticpods] Renewing apiserver certificate
[upgrade/staticpods] Renewing apiserver-kubelet-client certificate
[upgrade/staticpods] Renewing front-proxy-client certificate
[upgrade/staticpods] Renewing apiserver-etcd-client certificate
[upgrade/staticpods] Moving new manifest to "/etc/kubernetes/manifests/kube-apiserver.yaml" and backing up old manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2024-09-18-08-53-04/kube-apiserver.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This can take up to 5m0s
[apiclient] Found 1 Pods for label selector component=kube-apiserver
[upgrade/staticpods] Component "kube-apiserver" upgraded successfully!
[upgrade/staticpods] Preparing for "kube-controller-manager" upgrade
[upgrade/staticpods] Renewing controller-manager.conf certificate
[upgrade/staticpods] Moving new manifest to "/etc/kubernetes/manifests/kube-controller-manager.yaml" and backing up old manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2024-09-18-08-53-04/kube-controller-manager.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This can take up to 5m0s
[apiclient] Found 1 Pods for label selector component=kube-controller-manager
[upgrade/staticpods] Component "kube-controller-manager" upgraded successfully!
[upgrade/staticpods] Preparing for "kube-scheduler" upgrade
[upgrade/staticpods] Renewing scheduler.conf certificate
[upgrade/staticpods] Moving new manifest to "/etc/kubernetes/manifests/kube-scheduler.yaml" and backing up old manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2024-09-18-08-53-04/kube-scheduler.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This can take up to 5m0s
[apiclient] Found 1 Pods for label selector component=kube-scheduler
[upgrade/staticpods] Component "kube-scheduler" upgraded successfully!
[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
[upgrade] Backing up kubelet config file to /etc/kubernetes/tmp/kubeadm-kubelet-config3133270895/config.yaml
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[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
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.31.1". Enjoy!
```

This command is successfully executed.

Now we need to upgrade Kubectl and kubelet by following the below steps

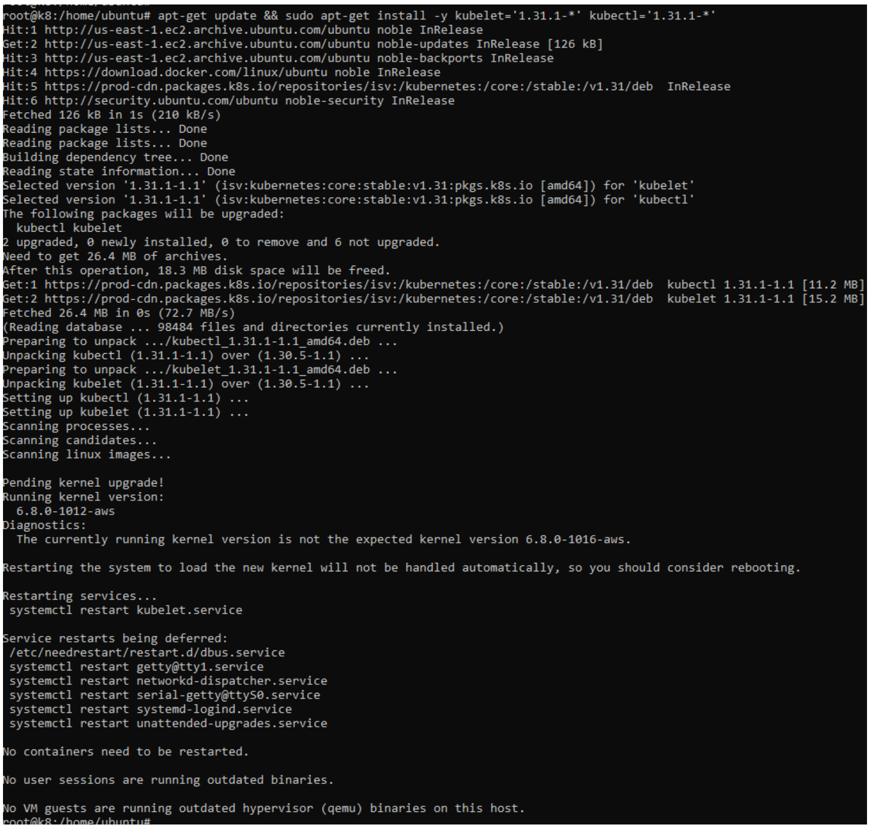
Now we need to remove the hold on the kubelet and kubectl, so that we can upgrade it to the latest version v 1.31

apt-mark unhold kubelet kubectl



```
root@k8:/home/ubuntu# apt-mark unhold kubelet kubectl
Canceled hold on kubelet.
Canceled hold on kubectl.
```

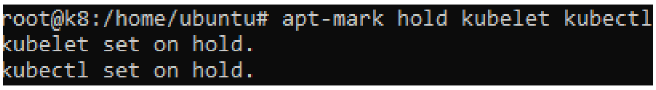
Now we will install the kubectl and kubelet package for v 1.31



```
root@k8:/home/ubuntu# apt-get update && sudo apt-get install -y kubelet='1.31.1-*' kubectl='1.31.1-*'
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 126 kB in 1s (210 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Selected version '1.31.1-1.1' (isv:kubernetes:core:stable:v1.31:pkgs.k8s.io [amd64]) for 'kubelet'
Selected version '1.31.1-1.1' (isv:kubernetes:core:stable:v1.31:pkgs.k8s.io [amd64]) for 'kubectl'
The following packages will be upgraded:
  kubectl kubelet
2 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
Need to get 26.4 MB of archives.
After this operation, 18.3 MB disk space will be freed.
Get:1 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubelet 1.31.1-1.1 [15.2 MB]
Fetched 26.4 MB in 0s (72.7 MB/s)
(Reading database ... 98484 files and directories currently installed.)
Preparing to unpack .../kubectl_1.31.1-1.1_amd64.deb ...
Unpacking kubectl (1.31.1-1.1) over (1.30.5-1.1) ...
Preparing to unpack .../kubelet_1.31.1-1.1_amd64.deb ...
Unpacking kubelet (1.31.1-1.1) over (1.30.5-1.1) ...
Setting up kubectl (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.8.0-1012-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-1016-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...
 systemctl restart kubelet.service

Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart serial-getty@ttyS0.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service

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@k8:/home/ubuntu#
```

Now we need to put back the hold on the kubeadm as version update is done.

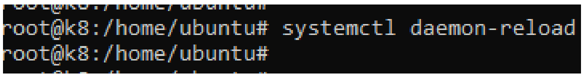
apt-mark hold kubelet kubectl



```
root@k8:/home/ubuntu# apt-mark hold kubelet kubectl
kubelet set on hold.
kubectl set on hold.
```

We will reload the daemon

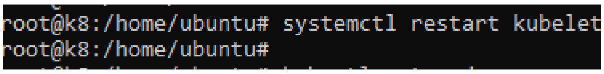
systemctl daemon-reload



```
root@k8:/home/ubuntu# systemctl daemon-reload
root@k8:/home/ubuntu#
```

We will restart the kubelet service on control plane.

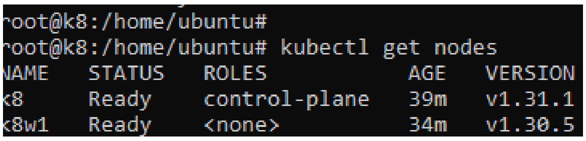
systemctl restart kubelet



```
root@k8:/home/ubuntu# systemctl restart kubelet
root@k8:/home/ubuntu#
```

We will verify that upgrade has been done on the Control plane successfully.

kubectl get nodes



```
root@k8:/home/ubuntu#
root@k8:/home/ubuntu# kubectl get nodes
NAME   STATUS   ROLES           AGE   VERSION
k8     Ready    control-plane   39m   v1.31.1
k8w1   Ready    <none>          34m   v1.30.5
```

We can see that the Control plane has been successfully upgraded to v 1.31.

Now we need to upgrade the worker node to the v 1.31

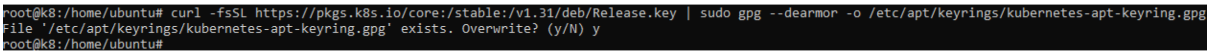
These are components we need to upgraded for Kubernetes

- Kubeadm
- Kubectl and kubelet

We need to follow the below steps on Worker Node

We will overwrite the key string by replacing it with the version we want to upgrade in our case we will take key string for v1.31 so that we can get the packages of that version

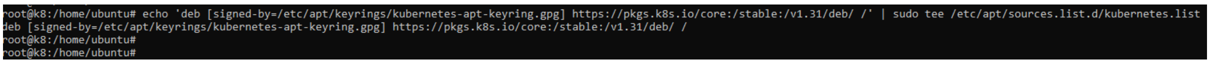
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg



```
root@k8:/home/ubuntu# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
File '/etc/apt/keyrings/kubernetes-apt-keyring.gpg' exists. Overwrite? (y/N) y
root@k8:/home/ubuntu#
```

Now we will overwrite the existing configuration with that of version 1.31

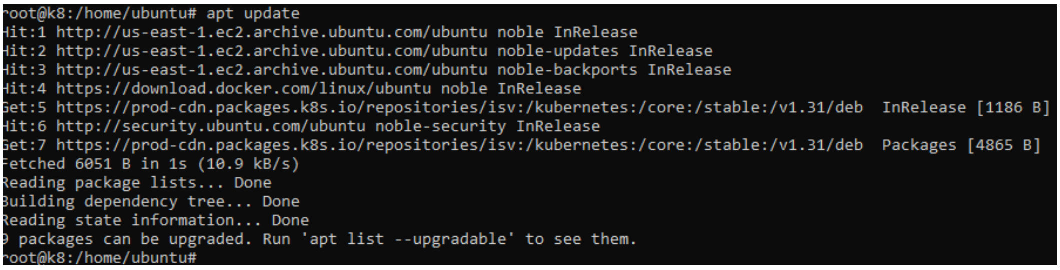
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list



```
root@k8:/home/ubuntu# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
root@k8:/home/ubuntu#
root@k8:/home/ubuntu#
```

Then we need to update the package index to v 1.31

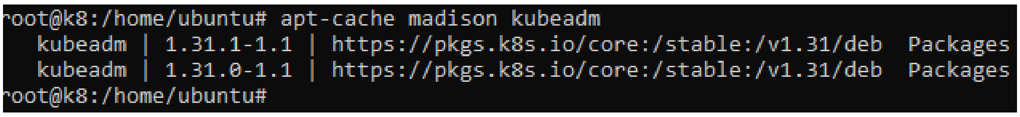
apt-get update



```
root@k8:/home/ubuntu# apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (10.9 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
9 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@k8:/home/ubuntu#
```

We can check for the available package versions on which we can update.

apt-cache madison kubeadm



```
root@k8:/home/ubuntu# apt-cache madison kubeadm
   kubeadm | 1.31.1-1.1 | https://pkgs.k8s.io/core:/stable:/v1.31/deb  Packages
   kubeadm | 1.31.0-1.1 | https://pkgs.k8s.io/core:/stable:/v1.31/deb  Packages
root@k8:/home/ubuntu#
```

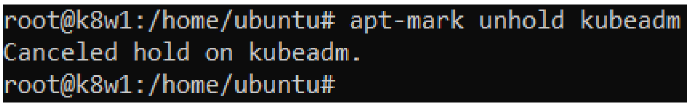
These are components we need to upgraded for Kubernetes Control plane.

- Kubeadm
- Kubectl and kubelet

We are upgrading the kubeadm on Worker Node by following the below steps

Now we need to remove the hold on the kubeadm, so that we can upgrade it to the latest version v 1.31

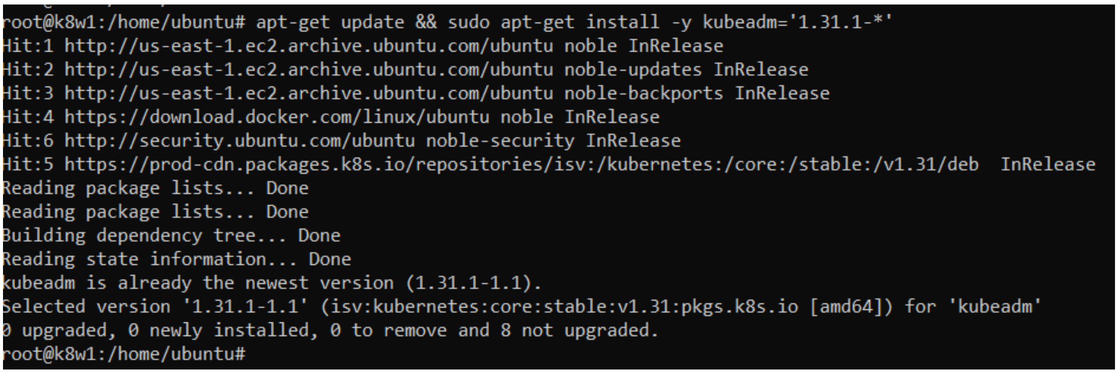
apt-mark unhold kubeadm



```
root@k8w1:/home/ubuntu# apt-mark unhold kubeadm
Canceled hold on kubeadm.
root@k8w1:/home/ubuntu#
```

Now we will install the kubeadm package for v 1.31

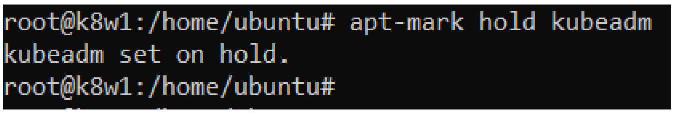
apt-get update && sudo apt-get install -y kubeadm='1.31.1-*'



```
root@k8w1:/home/ubuntu# apt-get update && sudo apt-get install -y kubeadm='1.31.1-*'
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
kubeadm is already the newest version (1.31.1-1.1).
Selected version '1.31.1-1.1' (isv:kubernetes:core:stable:v1.31:pkgs.k8s.io [amd64]) for 'kubeadm'
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
root@k8w1:/home/ubuntu#
```

Now we need to put back the hold on the kubeadm as version update is done.

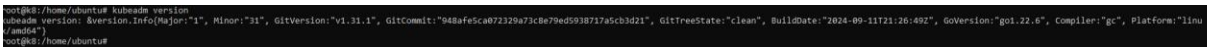
apt-mark hold kubeadm



```
root@k8w1:/home/ubuntu# apt-mark hold kubeadm
kubeadm set on hold.
root@k8w1:/home/ubuntu#
```

We will verify the kubeadm version

kubeadm version



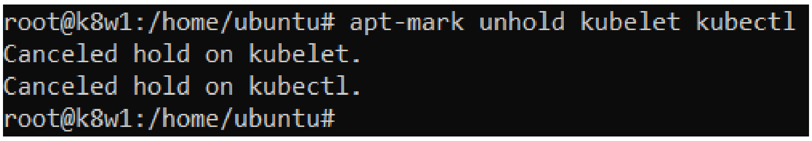
```
root@k8:/home/ubuntu# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"31", GitVersion:"v1.31.1", GitCommit:"948afe5ca072329a73c8e79ed5938717a5cb3d21", GitTreeState:"clean", BuildDate:"2024-09-11T21:26:49Z", GoVersion:"go1.22.6", Compiler:"gc", Platform:"linu
x/amd64"}
root@k8:/home/ubuntu#
```

We can see kubeadm is upgraded to v 1.31

Now we need to upgrade Kubectl and kubelet by following the below steps

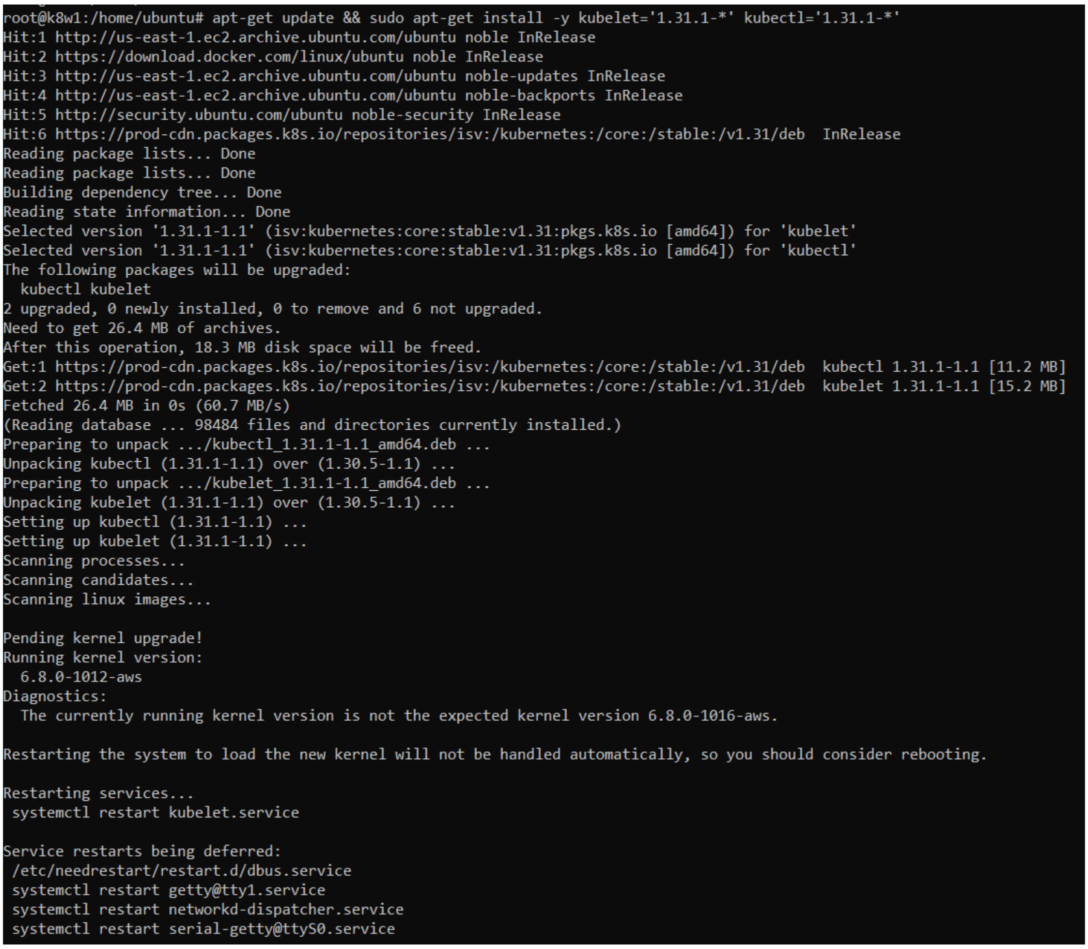
Now we need to remove the hold on the kubelet and kubectl, so that we can upgrade it to the latest version v 1.31

apt-mark unhold kubelet kubectl



```
root@k8w1:/home/ubuntu# apt-mark unhold kubelet kubectl
Canceled hold on kubelet.
Canceled hold on kubectl.
root@k8w1:/home/ubuntu#
```

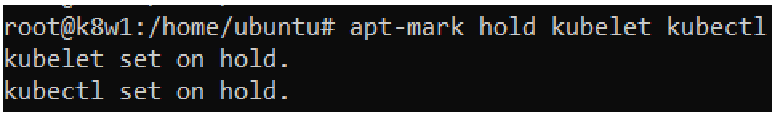
Now we will install the kubectl and kubelet package for v 1.31



```
root@k8w1:/home/ubuntu# apt-get update && sudo apt-get install -y kubelet='1.31.1-*' kubectl='1.31.1-*'
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Selected version '1.31.1-1.1' (isv:kubernetes:core:stable:v1.31:pkgs.k8s.io [amd64]) for 'kubelet'
Selected version '1.31.1-1.1' (isv:kubernetes:core:stable:v1.31:pkgs.k8s.io [amd64]) for 'kubectl'
The following packages will be upgraded:
  kubectl kubelet
2 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
Need to get 26.4 MB of archives.
After this operation, 18.3 MB disk space will be freed.
Get:1 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubelet 1.31.1-1.1 [15.2 MB]
Fetched 26.4 MB in 0s (60.7 MB/s)
(Reading database ... 98484 files and directories currently installed.)
Preparing to unpack .../kubectl_1.31.1-1.1_amd64.deb ...
Unpacking kubectl (1.31.1-1.1) over (1.30.5-1.1) ...
Preparing to unpack .../kubelet_1.31.1-1.1_amd64.deb ...
Unpacking kubelet (1.31.1-1.1) over (1.30.5-1.1) ...
Setting up kubectl (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.8.0-1012-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-1016-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...
 systemctl restart kubelet.service

Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart serial-getty@ttyS0.service
```

Now we need to put back the hold on the kubeadm as version update is done.

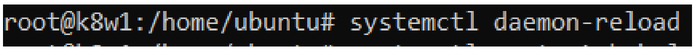
apt-mark hold kubelet kubectl



```
root@k8w1:/home/ubuntu# apt-mark hold kubelet kubectl
kubelet set on hold.
kubectl set on hold.
```

We will reload the daemon

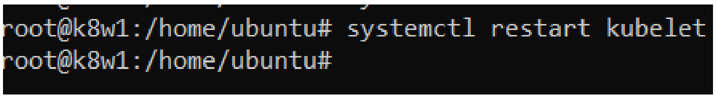
systemctl daemon-reload



```
root@k8w1:/home/ubuntu# systemctl daemon-reload
```

We will restart the kubelet service on control plane.

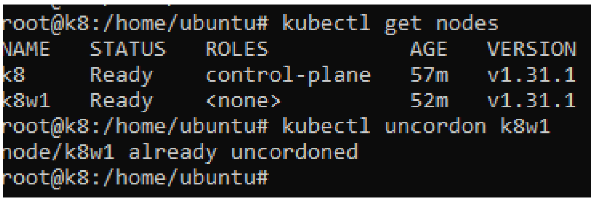
systemctl restart kubelet



```
root@k8w1:/home/ubuntu# systemctl restart kubelet
root@k8w1:/home/ubuntu#
```

We will verify that upgrade has been done from the Control plane successfully.

kubectl get nodes



```
root@k8:/home/ubuntu# kubectl get nodes
NAME   STATUS   ROLES           AGE   VERSION
k8     Ready    control-plane   57m   v1.31.1
k8w1   Ready    <none>          52m   v1.31.1
root@k8:/home/ubuntu# kubectl uncordon k8w1
node/k8w1 already uncordoned
root@k8:/home/ubuntu#
```

We can see that the Control plane and worker node also have been successfully upgraded to v 1.31.