EXPERIMENT NO. 3

NAME : PRANAV POL CLASS : D15A ROLL NO. : 42

Aim : To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Theory :

Kubernetes (often abbreviated as K8s) is a powerful open-source platform for managing containerized applications across multiple hosts. It provides an abstraction layer for deploying and scaling applications, ensuring high availability and fault tolerance. The **Kubernetes Cluster** architecture consists of several key components divided into two major categories:

1. **Master Node Components**
2. **Worker Node Components**

1. Master Node Components

The **Master Node** is responsible for managing the entire Kubernetes cluster. It acts as the brain of the cluster, orchestrating container deployments, scaling, and communication between the nodes.

a. API Server

- Acts as the front-end for the Kubernetes control plane.
- All internal and external communication with the cluster is through the API server.
- It validates and processes RESTful requests.

b. Scheduler

- Responsible for distributing workload across the cluster.
- Determines which worker node will host a newly created pod based on resource availability, policy constraints, and other factors.

c. Controller Manager

- Ensures that the desired state of the cluster matches the actual state.
- Examples of controllers include the **Node Controller**, **Replication Controller**, **Endpoint Controller**, and **Service Controller**.

d. etcd

- A consistent and highly-available key-value store used as Kubernetes' backing store.
- Stores the entire configuration and state of the Kubernetes cluster.
- etcd must be backed up regularly as it's critical to the integrity of the cluster.

e. Cloud Controller Manager (optional)

- Integrates cloud-specific APIs into Kubernetes.
- Handles cloud-related services like load balancing, managing cloud storage, and node lifecycle events in cloud platforms like AWS, GCP, Azure, etc.

2. Worker Node Components

The **Worker Nodes** are where your applications run. Each worker node communicates with the master node and executes the commands given.

a. Kubelet

- The primary agent that runs on each worker node.
- Ensures that the containers described in a pod are running and healthy.
- Communicates with the API server and ensures that the desired state is met on the node.

b. Kube-proxy

- Manages the networking for the worker node.
- Ensures that traffic is correctly routed to and from the pods running on the worker node.
- Facilitates communication between services within the cluster.

c. Container Runtime

- Responsible for pulling container images and running the containers.
- Common runtimes include Docker, containerd, or CRI-O.

d. Pods

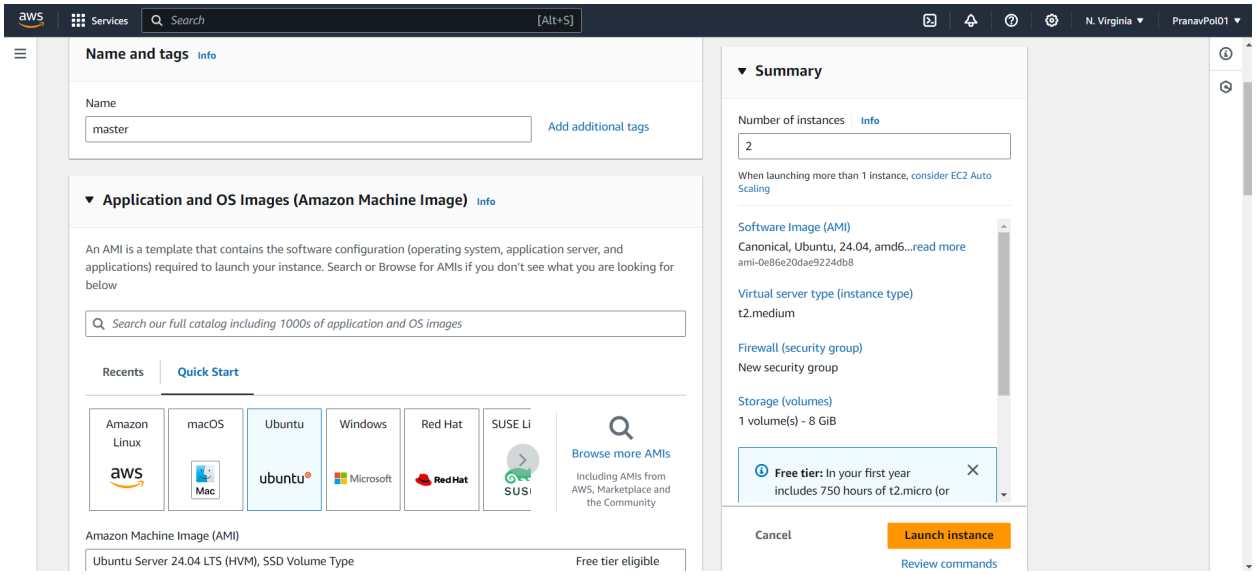
- The smallest deployable unit in Kubernetes.

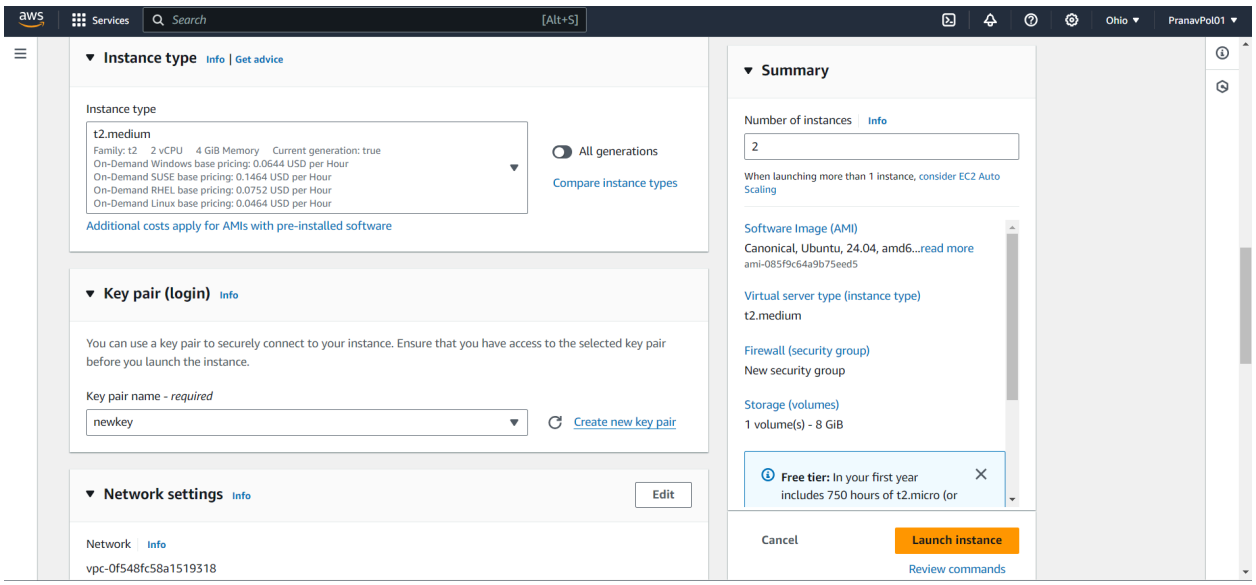
- A pod encapsulates one or more containers and their shared storage, network, and configuration options.

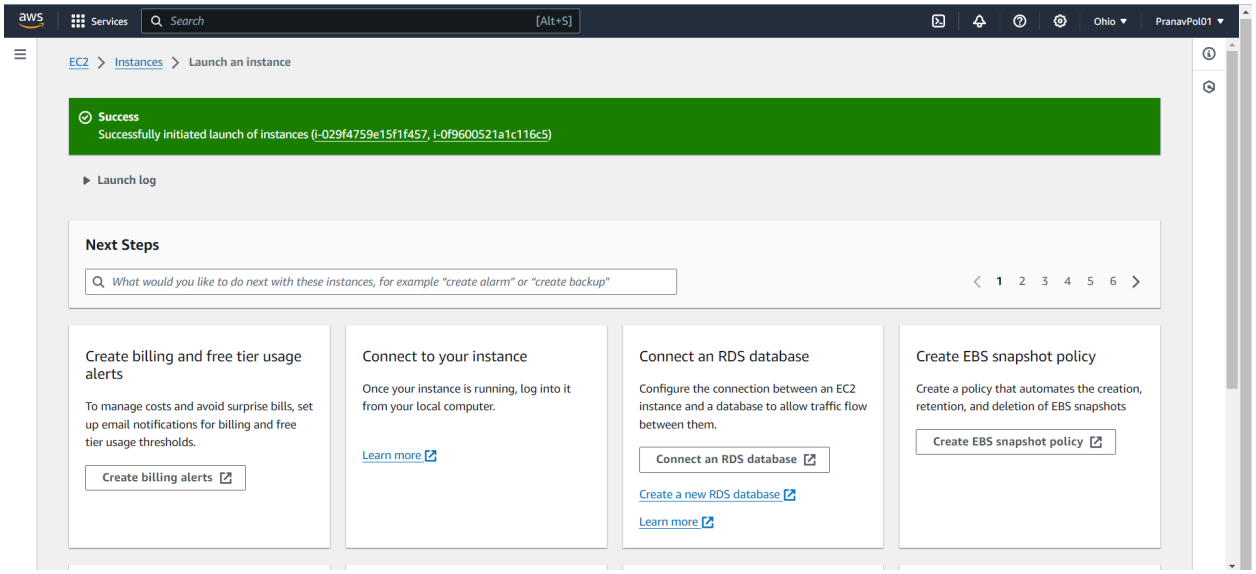
Implementation :

Step 1:Pre-requisites

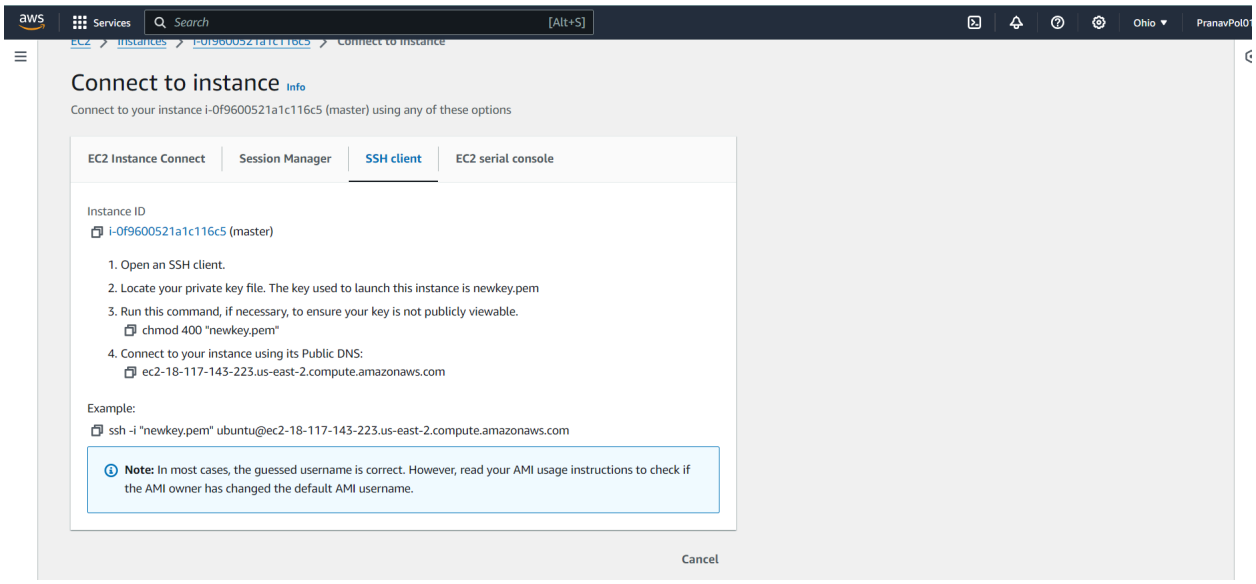
1.1. Create 2 EC2 instances,one for the master node and one for the worker nodes.

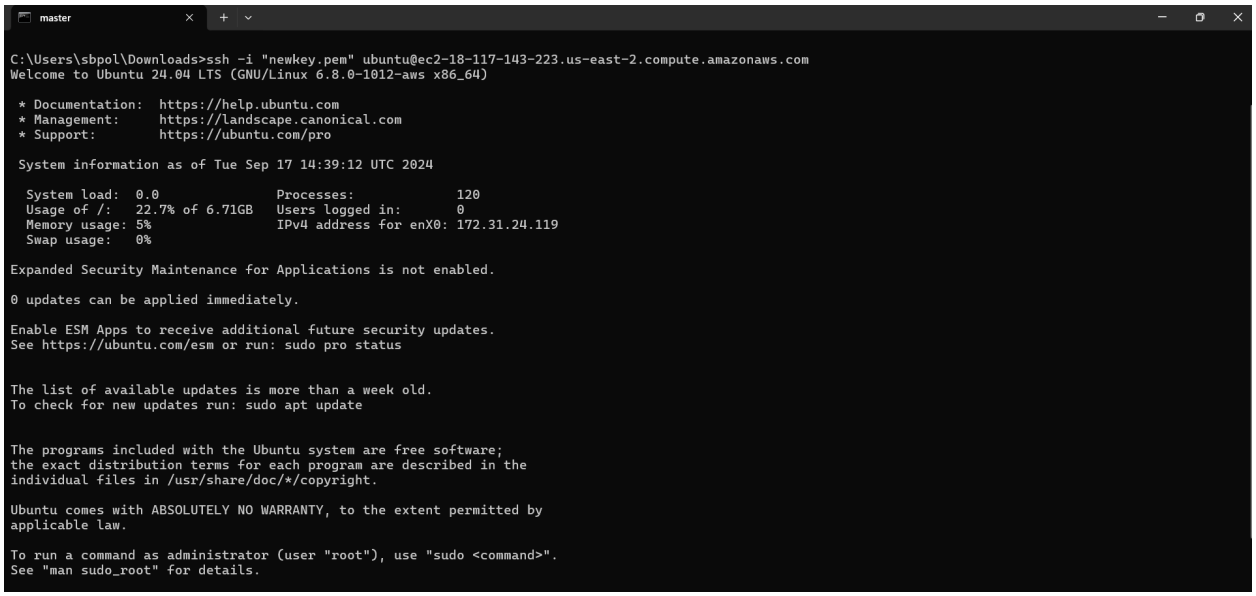


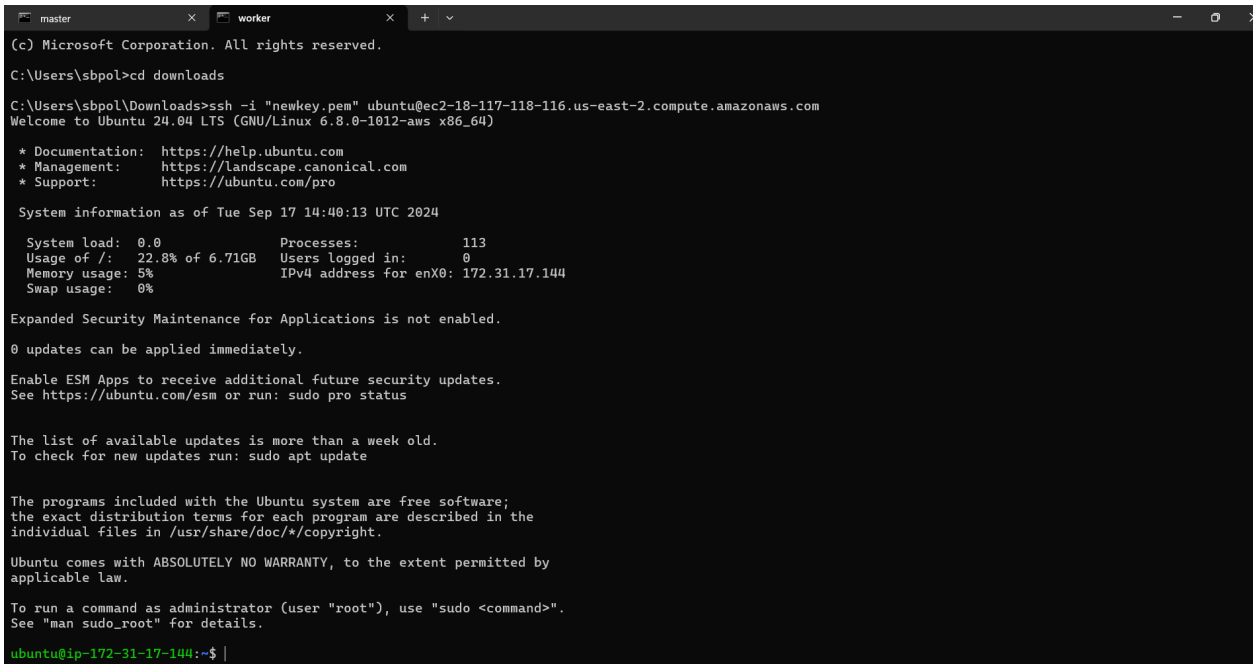




1.2 . After the instances have been created,copy the text given in the example part of each of the three instances into git bash.

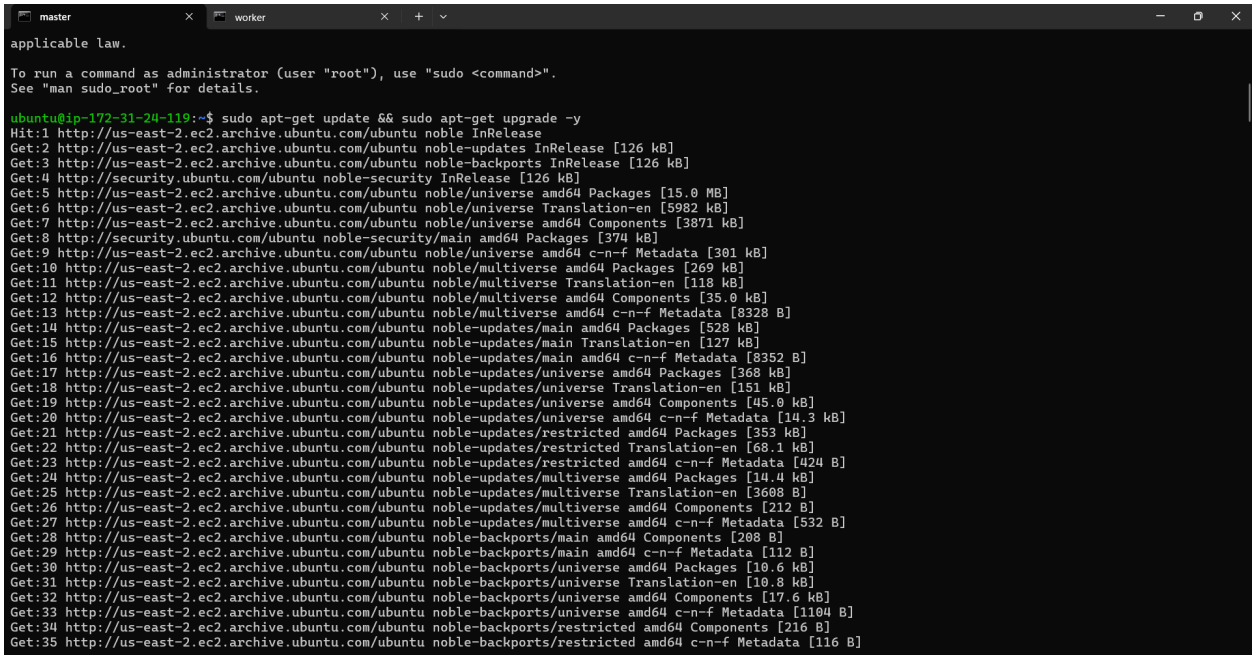



master
C:\Users\sbpol\Downloads>ssh -i "newkey.pem" ubuntu@ec2-18-117-143-223.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
System information as of Tue Sep 17 14:39:12 UTC 2024
System load: 0.0 Processes: 120
Usage of /: 22.7% of 6.71GB Users logged in: 0
Memory usage: 5% IPv4 address for enX0: 172.31.24.119
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
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
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.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.


master
worker
(c) Microsoft Corporation. All rights reserved.
C:\Users\sbpol>cd downloads
C:\Users\sbpol\Downloads>ssh -i "newkey.pem" ubuntu@ec2-18-117-118-116.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 24.04 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 14:40:13 UTC 2024
System load: 0.0 Processes: 113
Usage of /: 22.8% of 6.71GB Users logged in: 0
Memory usage: 5% IPv4 address for enX0: 172.31.17.144
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
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
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.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-172-31-17-144:~$

Step 2:Prepare Nodes

2.1. Update the package manager on all nodes:
Sudo apt-get update &7 sudo apt-get update -y



2.2. Disable Swap (Kubernetes requires swap to be off):

sudo swapoff -a

sudo sed -i '/ swap / s/^/#/' /etc/fstab

2.3. Load necessary kernel modules for networking and iptables: cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay
br_netfilt
er EOF
sudo modprobe overlay
sudo modprobe
br_netfilter

```
ubuntu@ip-172-31-24-119:~$ cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter
overlay
br_netfilter
```

2.1. Configure sysctl settings for Kubernetes networking: cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

sudo sysctl --system

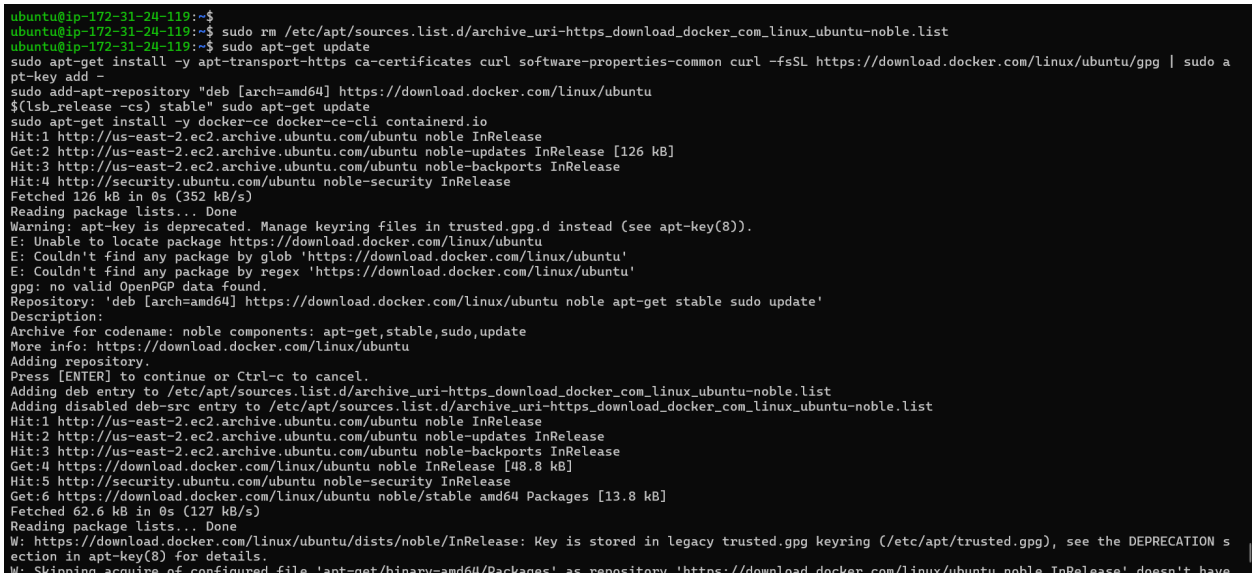
```
ubuntu@ip-172-31-24-119:~$ cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
* 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/k8s.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
```

Step 3: Install Docker

Kubernetes uses container runtimes like Docker. Install Docker on all nodes.
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
software-properties-common curl -fsSL
https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu

$(lsb_release -cs) stable" sudo apt-get update sudo apt-get install -y docker-ce
docker-ce-cli containerd.io



Configure Docker for Kubernetes:

```
cat <<EOF | sudo tee /etc/docker/daemon.json
{
 "exec-opts":
 ["native.cgroupdriver=systemd"],
 "log-driver": "json-file",

 "log-opts": {

  "max-size": "100m"
 },
 "storage-driver": "overlay2"
}
```

EOF

sudo systemctl restart docker

```
ubuntu@ip-172-31-24-119:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"], "log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
sudo systemctl restart docker

{
"exec-opts": ["native.cgroupdriver=systemd"], "log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
ubuntu@ip-172-31-24-119:~$
```

Step 4: Install kubeadm, kubelet, kubectl

Install Kubernetes tools on all nodes.

4.1. Add Kubernetes APT repository:

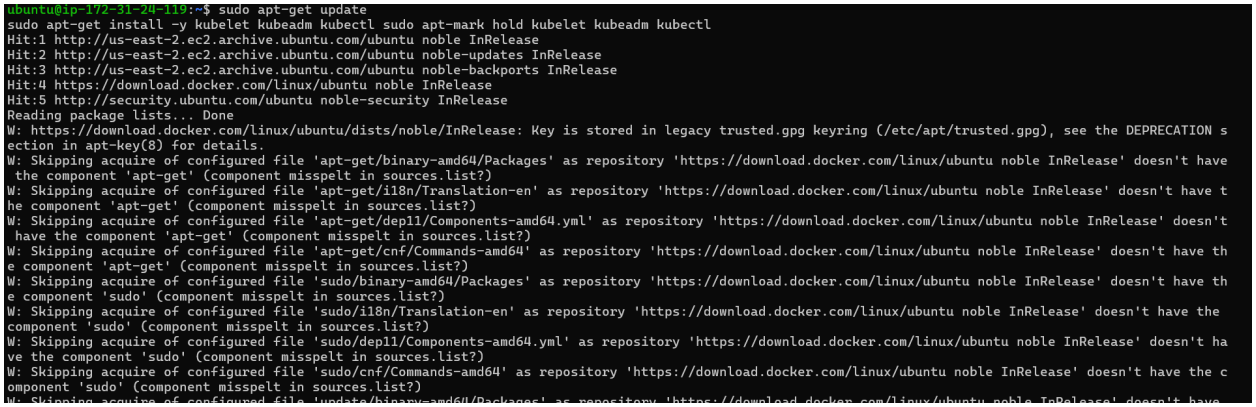
sudo curl -fsSLo
/usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee

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

```
ubuntu@ip-172-31-17-144:~$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main
```

4.2. Install kubeadm, kubelet, and
kubectl: sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl



4.3 Install crio runtime
Using CRI-O can help streamline your container management processes, improve the security of your applications, and simplify the overall management of your infrastructure.

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

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

sudo apt-get update -y
sudo apt-get install -y cri-o

sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service

echo "CRI runtime installed successfully"

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

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

sudo apt-get update -y
sudo apt-get install -y cri-o

sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service

echo "CRI runtime installed successfully"
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
software-properties-common is already the newest version (0.99.48).
software-properties-common set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.4).
curl set to manually installed.
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
gpg is already the newest version (2.4.4-2ubuntu17).
gpg set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
```

4.4 Add Kubernetes APT repository and install required packages
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/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:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-44-131:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyri
ng.gpg
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/kubern
etes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /
```

4.5 Install kubelet kubectl and kubeadm
sudo apt-get update -y
sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"
sudo apt-get update -y
sudo apt-get install -y jq

sudo systemctl enable --now kubelet
sudo systemctl start kubelet

```
ubuntu@ip-172-31-44-131:~$ sudo apt-get update -y
sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"
sudo apt-get update -y
sudo apt-get install -y jq
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease:/main/deb  InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security 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 1s (23.9 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Selected version '1.29.0-1.1' (isv:kubernetes:core:stable:v1.29:pkgs.k8s.io [amd64]) for 'kubelet'
Selected version '1.29.0-1.1' (isv:kubernetes:core:stable:v1.29:pkgs.k8s.io [amd64]) for 'kubectl'
Selected version '1.29.0-1.1' (isv:kubernetes:core:stable:v1.29:pkgs.k8s.io [amd64]) for 'kubeadm'
The following additional packages will be installed:
  conntrack cri-tools ebtables kubernetes-cni socat
The following NEW packages will be installed:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 7 not upgraded.
```

```
ubuntu@ip-172-31-44-131:~$
sudo systemctl enable --now kubelet
sudo systemctl start kubelet
```

Step 5 : Execute only on master node

5.1 : kudeadm config and init

```
ubuntu@ip-172-31-44-131:~$ sudo kubeadm config images pull
I0917 16:12:12.827368   12932 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.29
[config/images] Pulled registry.k8s.io/kube-apiserver:v1.29.9
[config/images] Pulled registry.k8s.io/kube-controller-manager:v1.29.9
[config/images] Pulled registry.k8s.io/kube-scheduler:v1.29.9
[config/images] Pulled registry.k8s.io/kube-proxy:v1.29.9
[config/images] Pulled registry.k8s.io/coredns/coredns:v1.11.1
[config/images] Pulled registry.k8s.io/pause:3.9
[config/images] Pulled registry.k8s.io/etcd:3.5.10-0
```

```
ubuntu@ip-172-31-44-131:~$ sudo kubeadm init
I0917 16:12:56.929652   13290 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.29
[init] Using Kubernetes version: v1.29.9
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
W0917 16:12:57.776009   13290 checks.go:835] detected that the sandbox image "registry.k8s.io/pause:3.10" of the container runtime is inconsistent with that
 used by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-44-131 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster
.local] and IPs [10.96.0.1 172.31.44.131]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-44-131 localhost] and IPs [172.31.44.131 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-44-131 localhost] and IPs [172.31.44.131 127.0.0.1 ::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"
```

5.2 : install config from github

network plugin = calico

```
ubuntu@ip-172-31-44-131:~$ kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
```

5.3 Create token of kubeadm

```
ubuntu@ip-172-31-44-131:~$ kubeadm token create --print-join-command
kubeadm join 172.31.44.131:6443 --token zixj91.n5y95uhn5xyydmps --discovery-token-ca-cert-hash sha256:f48119eb49e6f94e34ef69d388d3b955a7b380d34a887e32494385
67341d9bcb
```

Step 6 : Execute on ALL of your Worker Node's

6.1 : perform pre- flight checks and Paste the join command you got from the master node and append --v=5 at the end

```
ubuntu@ip-172-31-40-114:~$ sudo kubeadm join 172.31.44.131:6443 --token zixj91.n5y95uhn5xyydmps --discovery-token-ca-cert-hash sha256:f48119eb49e6f94e34ef69d388d3b955a7b380d34a88
67341d9bcb --v=5
I0917 16:36:16.052568   13354 join.go:413] [preflight] found NodeName empty; using OS hostname as NodeName
I0917 16:36:16.052700   13354 initconfiguration.go:122] detected and using CRI socket: unix:///var/run/crio/crio.sock
[preflight] Running pre-flight checks
I0917 16:36:16.052757   13354 preflight.go:93] [preflight] Running general checks
I0917 16:36:16.052787   13354 checks.go:280] validating the existence of file /etc/kubernetes/kubelet.conf
I0917 16:36:16.052793   13354 checks.go:280] validating the existence of file /etc/kubernetes/bootstrap-kubelet.conf
I0917 16:36:16.052800   13354 checks.go:104] validating the container runtime
I0917 16:36:16.074072   13354 checks.go:639] validating whether swap is enabled or not
I0917 16:36:16.074157   13354 checks.go:370] validating the presence of executable crictl
I0917 16:36:16.074180   13354 checks.go:370] validating the presence of executable conntrack
I0917 16:36:16.074191   13354 checks.go:370] validating the presence of executable ip
I0917 16:36:16.074204   13354 checks.go:370] validating the presence of executable iptables
I0917 16:36:16.074216   13354 checks.go:370] validating the presence of executable mount
I0917 16:36:16.074229   13354 checks.go:370] validating the presence of executable nsenter
I0917 16:36:16.074260   13354 checks.go:370] validating the presence of executable ebtables
I0917 16:36:16.074291   13354 checks.go:370] validating the presence of executable ethtool
I0917 16:36:16.074305   13354 checks.go:370] validating the presence of executable socat
I0917 16:36:16.074328   13354 checks.go:370] validating the presence of executable tc
I0917 16:36:16.074343   13354 checks.go:370] validating the presence of executable touch
I0917 16:36:16.074386   13354 checks.go:516] running all checks
I0917 16:36:16.084808   13354 checks.go:401] checking whether the given node name is valid and reachable using net.LookupHost
I0917 16:36:16.086249   13354 checks.go:605] validating kubelet version
I0917 16:36:16.133477   13354 checks.go:130] validating if the "kubelet" service is enabled and active
I0917 16:36:16.145208   13354 checks.go:203] validating availability of port 10250
I0917 16:36:16.145467   13354 checks.go:280] validating the existence of file /etc/kubernetes/pki/ca.crt
I0917 16:36:16.145489   13354 checks.go:430] validating if the connectivity type is via proxy or direct
I0917 16:36:16.145528   13354 checks.go:329] validating the contents of file /proc/sys/net/bridge/bridge-nf-call-iptables
I0917 16:36:16.145575   13354 checks.go:329] validating the contents of file /proc/sys/net/ipv4/ip_forward
I0917 16:36:16.145610   13354 join.go:532] [preflight] Discovering cluster-info
I0917 16:36:16.145642   13354 token.go:80] [discovery] Created cluster-info discovery client, requesting info from "172.31.44.131:6443"
I0917 16:36:16.155196   13354 token.go:118] [discovery] Requesting info from "172.31.44.131:6443" again to validate TLS against the pinned public key
I0917 16:36:16.162805   13354 token.go:135] [discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will use API Server "172
:6443"
I0917 16:36:16.162817   13354 discovery.go:52] [discovery] Using provided TLSBootstrapToken as authentication credentials for the join process
I0917 16:36:16.162825   13354 join.go:546] [preflight] Fetching init configuration
I0917 16:36:16.162830   13354 join.go:592] [preflight] Retrieving KubeConfig objects
[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'
I0917 16:36:16.171694   13354 kubeproxy.go:55] attempting to download the KubeProxyConfiguration from ConfigMap "kube-proxy"
I0917 16:36:16.174741   13354 kubelet.go:74] attempting to download the KubeletConfiguration from ConfigMap "kubelet-config"
I0917 16:36:16.178638   13354 initconfiguration.go:114] skip CRI socket detection, fill with the default CRI socket unix:///var/run/containerd/containerd.sock
I0917 16:36:16.178801   13354 interface.go:432] Looking for default routes with IPv4 addresses
I0917 16:36:16.178819   13354 interface.go:437] Default route transits interface "enX0"
```

Step 7 : Verify cluster connection on master node

```
ubuntu@ip-172-31-44-131:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE   VERSION
ip-172-31-40-114   Ready    <none>          26s   v1.29.0
ip-172-31-44-131   Ready    control-plane   23m   v1.29.0
```