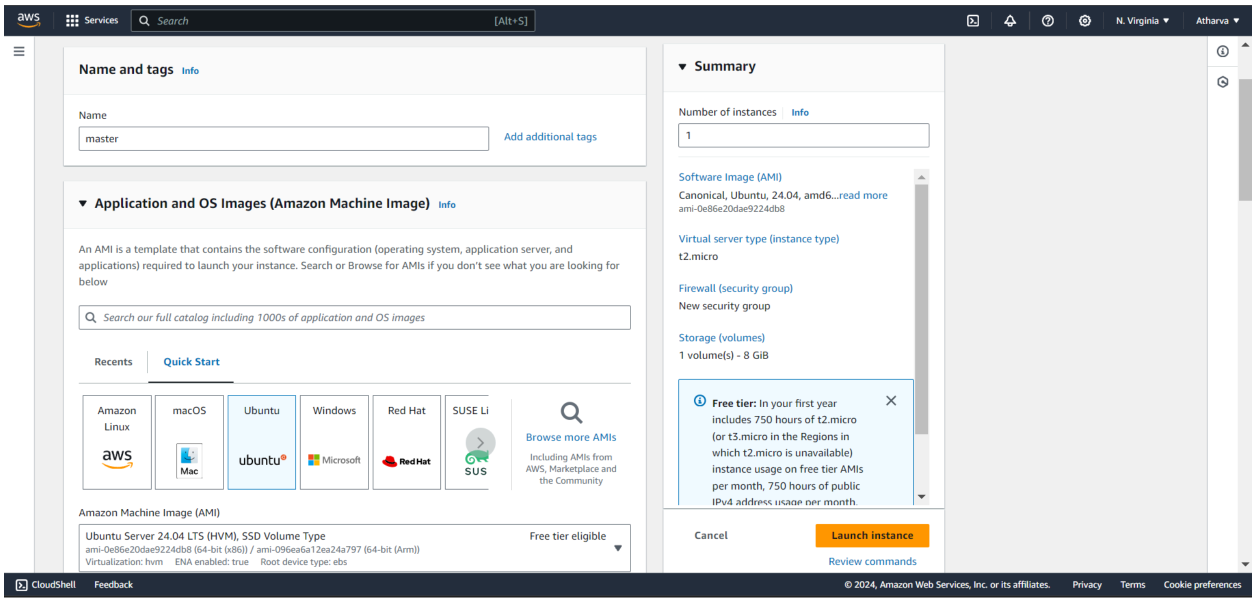
Jai Navani **Experiment 3**
D15A 31
AIM: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Step 1: Prerequisites
1.1 Create 3 EC2 instances,one for the master node and two for the worker nodes.


Name and tags

Name: master

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

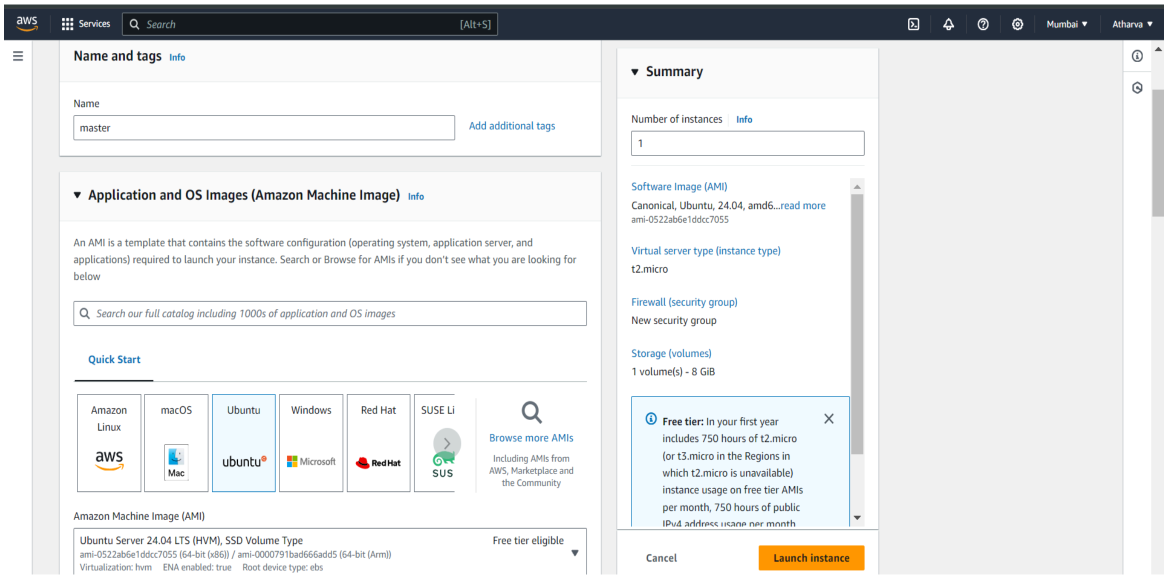
Amazon Machine Image (AMI)
Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-0e86e20dae9224db8 (64-bit (x86)) / ami-096ea6a12ea24a797 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Summary

Number of instances: 1
Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more
ami-0e86e20dae9224db8
Virtual server type (instance type): t2.micro
Firewall (security group): New security group
Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month.

1.2 Proceed with the following settings and create a new key pair as follows(use the same key pair for all the three nodes)



Name and tags Info

Name

master

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Li

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-0522ab6e1ddcc7055 (64-bit (x86)) / ami-0000791bad666add5 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Summary

Number of instances Info

1

Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0522ab6e1ddcc7055

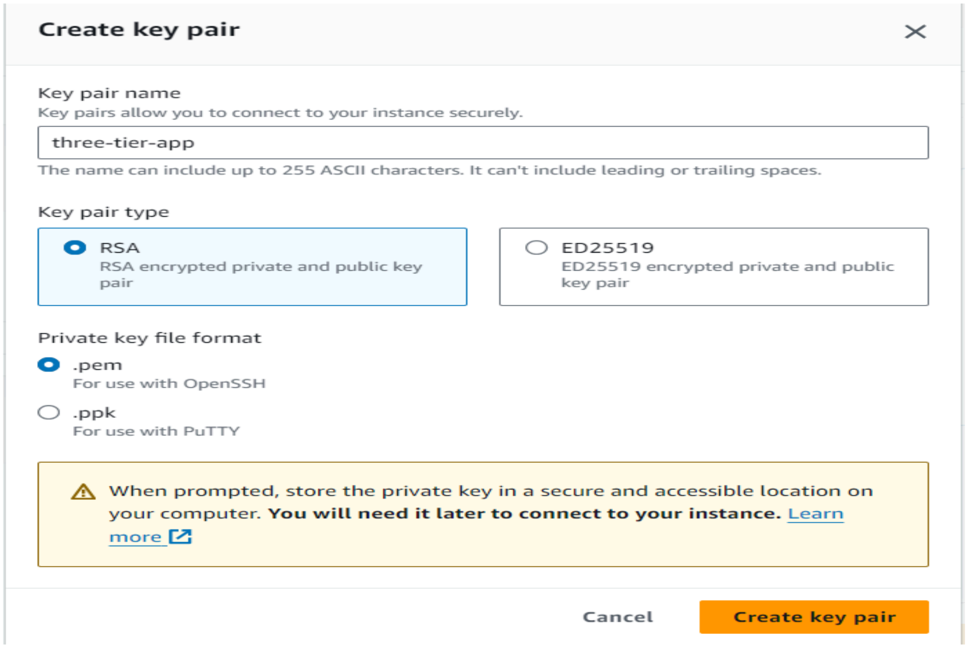
Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month

Cancel | Launch instance



Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.

three-tier-app

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

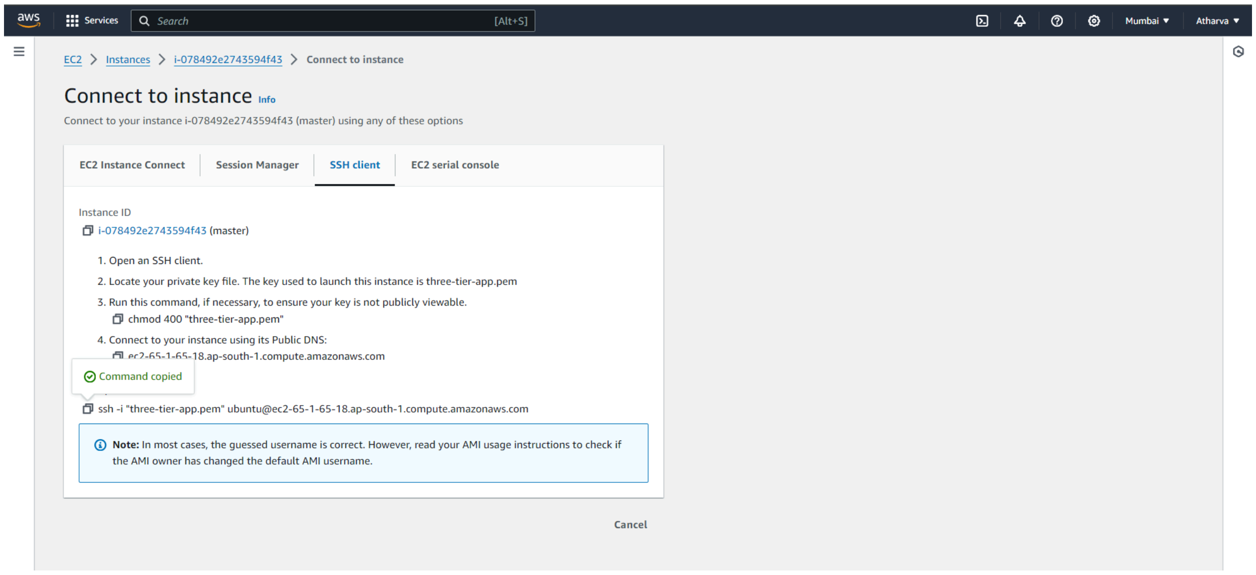
[✓ RSA — RSA encrypted private and public key pair]
ED25519 — ED25519 encrypted private and public key pair

Private key file format

[✓ .pem — For use with OpenSSH]
.ppk — For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

Cancel | Create key pair



Connect to instance Info

Connect to your instance i-078492e2743594f43 (master) using any of these options

Instance ID

i-078492e2743594f43 (master)

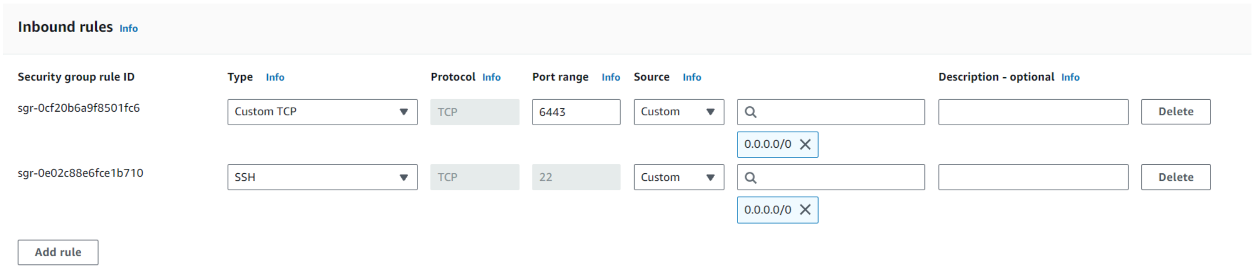
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is three-tier-app.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   `chmod 400 "three-tier-app.pem"`
4. Connect to your instance using its Public DNS:
   ec2-65-1-65-18.ap-south-1.compute.amazonaws.com

Command copied

```
ssh -i "three-tier-app.pem" ubuntu@ec2-65-1-65-18.ap-south-1.compute.amazonaws.com
```

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

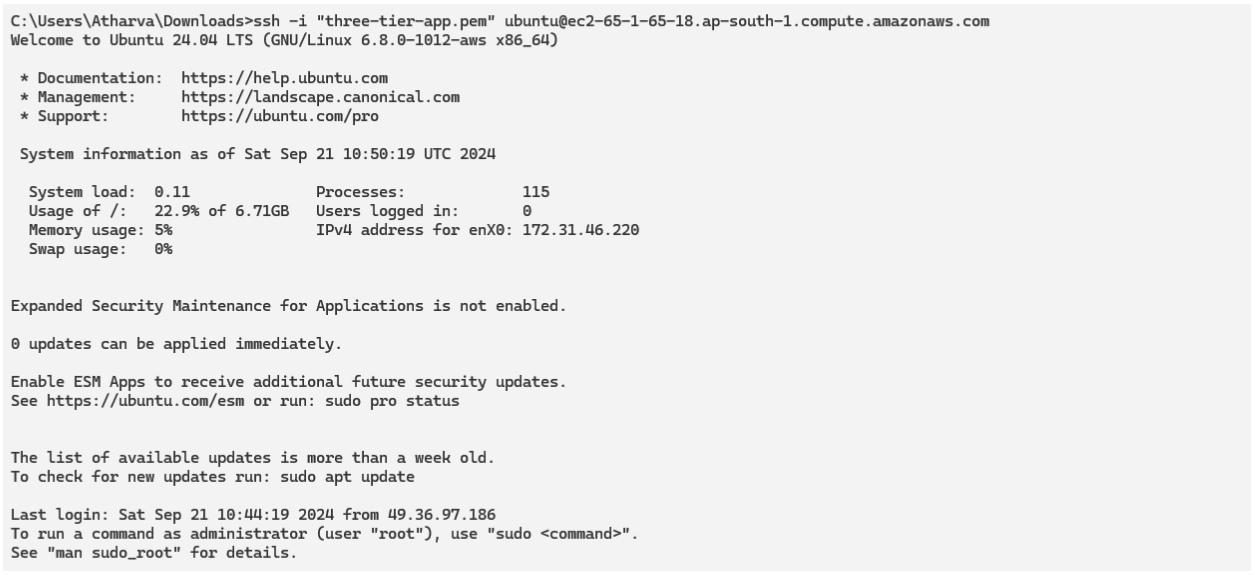
1.3 Add port 6443 in each security group



Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-0cf20b6a9f8501fc6	Custom TCP	TCP	6443	Custom	0.0.0.0/0	
sgr-0e02c88e6fce1b710	SSH	TCP	22	Custom	0.0.0.0/0	

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

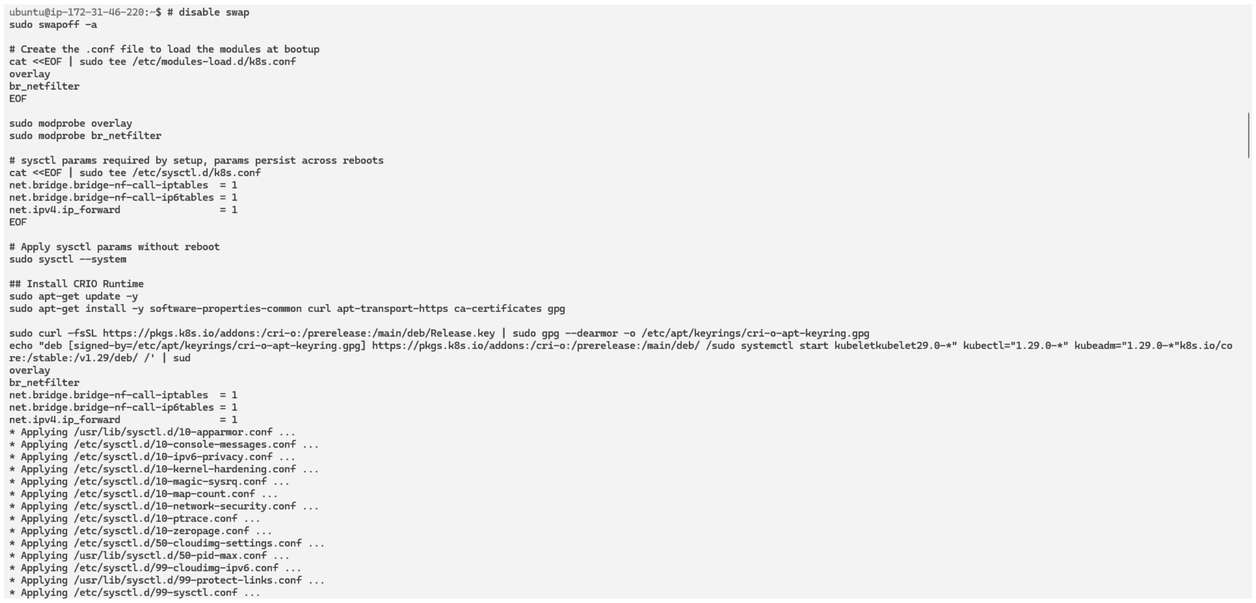


```
C:\Users\Atharva\Downloads>ssh -i "three-tier-app.pem" ubuntu@ec2-65-1-65-18.ap-south-1.compute.amazonaws.com
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Sat Sep 21 10:50:19 UTC 2024

  System load:  0.11              Processes:             115
  Usage of /:   22.9% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.46.220
  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

Last login: Sat Sep 21 10:44:19 2024 from 49.36.97.186
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Step 2: Run the following commands on both the master and worker nodes to prepare them for kubeadm.

```
# disable swap
sudo swapoff -a

# Create the .conf file to load the modules at bootup
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# sysctl params required by setup, params persist across reboots
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system
```

```
## Install CRIO Runtime
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"

# 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

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-172-31-46-220:~$ # disable swap
sudo swapoff -a

# Create the .conf file to load the modules at bootup
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# sysctl params required by setup, params persist across reboots
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system

## Install CRIO Runtime
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 systemctl start kubeletkubelet29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"k8s.io/co
re:/stable:/v1.29/deb/ /' | sud
overlay
br_netfilter
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 ...
```

Step3: Run the above command only on master node

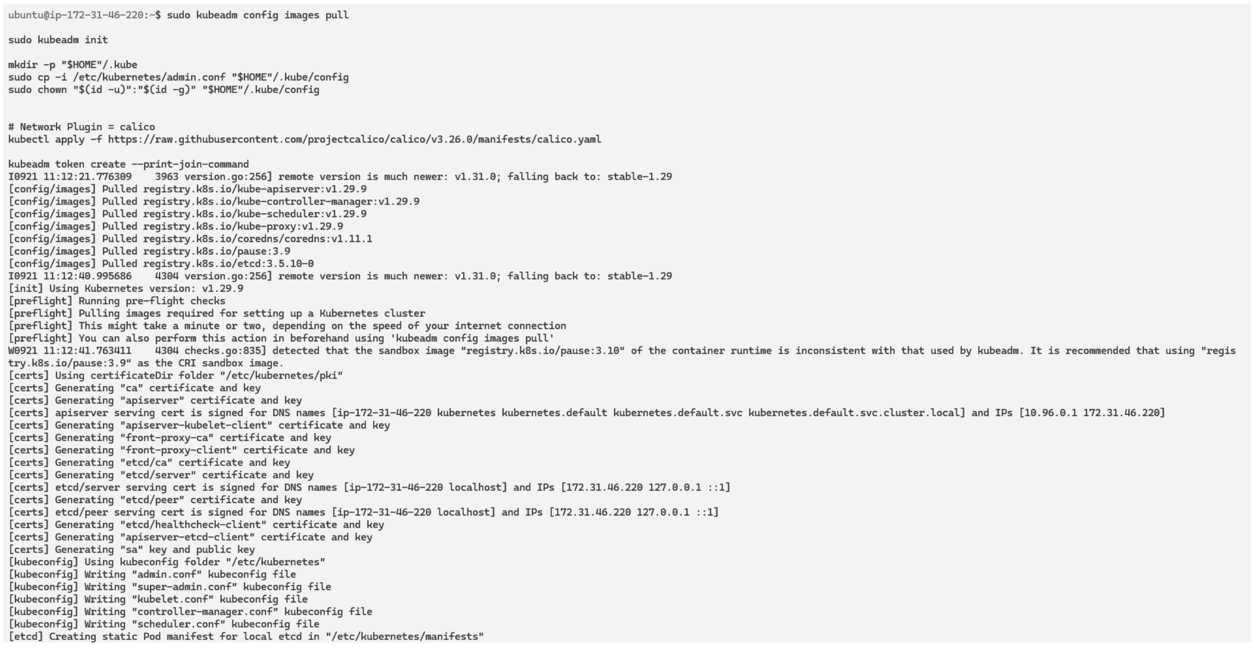
sudo kubeadm config images pull

sudo kubeadm init

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

# Network Plugin = calico
kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

kubeadm token create --print-join-command

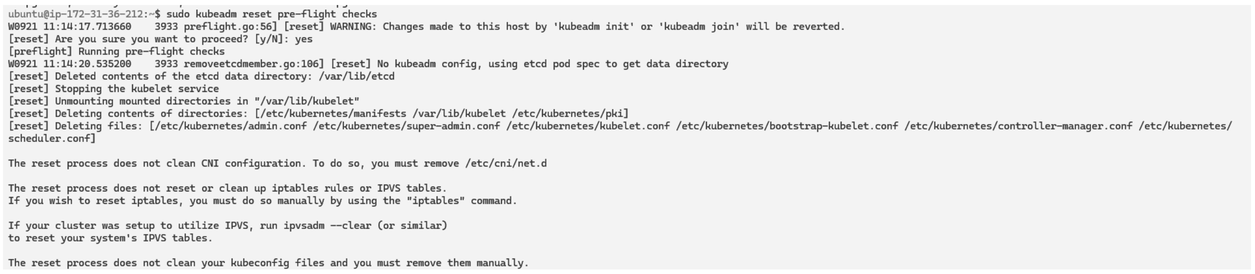


```
ubuntu@ip-172-31-46-220:~$ sudo kubeadm config images pull

sudo kubeadm init

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


# Network Plugin = calico
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

kubeadm token create --print-join-command
I0921 11:12:21.776309    3963 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
I0921 11:12:40.995686    4304 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'
W0921 11:12:41.763411    4304 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 "regis
try.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-172-31-46-220 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.46.220]
[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-172-31-46-220 localhost] and IPs [172.31.46.220 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-46-220 localhost] and IPs [172.31.46.220 127.0.0.1 ::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"
```

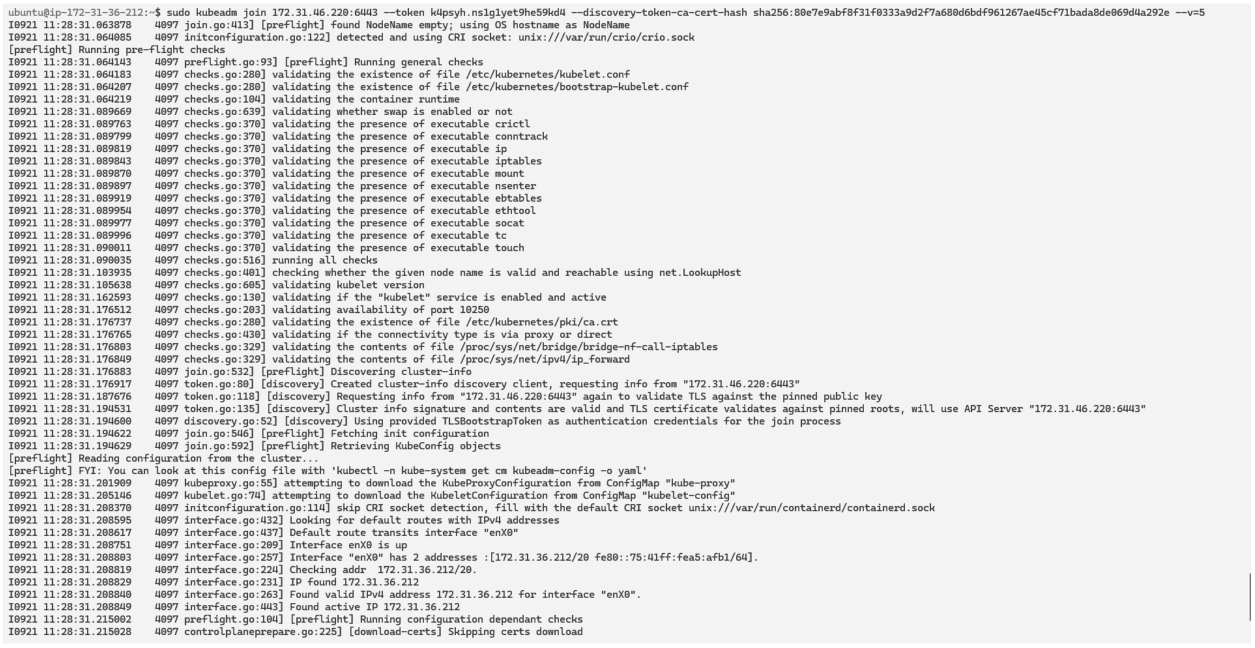
You will get kubeadm token, Copy it.

Step 4: Run the above command only on worker nodes

sudo kubeadm reset pre-flight checks
sudo your-token --v=5



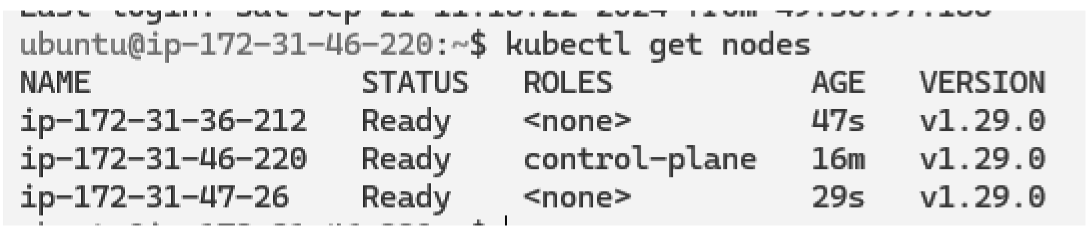
```
ubuntu@ip-172-31-36-212:~$ sudo kubeadm reset pre-flight checks
W0921 11:14:17.713660    3933 preflight.go:56] [reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join' will be reverted.
[reset] Are you sure you want to proceed? [y/N]: yes
[preflight] Running pre-flight checks
W0921 11:14:20.535200    3933 removeetcdmember.go:106] [reset] No kubeadm config, using etcd pod spec to get data directory
[reset] Deleted contents of the etcd data directory: /var/lib/etcd
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of directories: [/etc/kubernetes/manifests /var/lib/kubelet /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/super-admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.conf /etc/kubernetes/controller-manager.conf /etc/kubernetes/
scheduler.conf]

The reset process does not clean CNI configuration. To do so, you must remove /etc/cni/net.d

The reset process does not reset or clean up iptables rules or IPVS tables.
If you wish to reset iptables, you must do so manually by using the "iptables" command.

If your cluster was setup to utilize IPVS, run ipvsadm --clear (or similar)
to reset your system's IPVS tables.

The reset process does not clean your kubeconfig files and you must remove them manually.
```



```
ubuntu@ip-172-31-36-212:~$ sudo kubeadm join 172.31.46.220:6443 --token k4psyh.ns1g1yet9he59kd4 --discovery-token-ca-cert-hash sha256:80e7e9abf8f31f0333a9d2f7a680d6bdf961267ae45cf71bada8de069d4a292e --v=5
I0921 11:28:31.063878    4097 join.go:413] [preflight] found NodeName empty; using OS hostname as NodeName
I0921 11:28:31.064085    4097 initconfiguration.go:122] detected and using CRI socket: unix:///var/run/crio/crio.sock
[preflight] Running pre-flight checks
I0921 11:28:31.064143    4097 preflight.go:93] [preflight] Running general checks
I0921 11:28:31.064183    4097 checks.go:280] validating the existence of file /etc/kubernetes/kubelet.conf
I0921 11:28:31.064207    4097 checks.go:280] validating the existence of file /etc/kubernetes/bootstrap-kubelet.conf
I0921 11:28:31.064219    4097 checks.go:104] validating the container runtime
I0921 11:28:31.089669    4097 checks.go:639] validating whether swap is enabled or not
I0921 11:28:31.089763    4097 checks.go:370] validating the presence of executable crictl
I0921 11:28:31.089799    4097 checks.go:370] validating the presence of executable conntrack
I0921 11:28:31.089819    4097 checks.go:370] validating the presence of executable ip
I0921 11:28:31.089843    4097 checks.go:370] validating the presence of executable iptables
I0921 11:28:31.089870    4097 checks.go:370] validating the presence of executable mount
I0921 11:28:31.089897    4097 checks.go:370] validating the presence of executable nsenter
I0921 11:28:31.089919    4097 checks.go:370] validating the presence of executable ebtables
I0921 11:28:31.089954    4097 checks.go:370] validating the presence of executable ethtool
I0921 11:28:31.089977    4097 checks.go:370] validating the presence of executable socat
I0921 11:28:31.089996    4097 checks.go:370] validating the presence of executable tc
I0921 11:28:31.090011    4097 checks.go:370] validating the presence of executable touch
I0921 11:28:31.090035    4097 checks.go:516] running all checks
I0921 11:28:31.103935    4097 checks.go:401] checking whether the given node name is valid and reachable using net.LookupHost
I0921 11:28:31.105638    4097 checks.go:605] validating kubelet version
I0921 11:28:31.162593    4097 checks.go:130] validating if the "kubelet" service is enabled and active
I0921 11:28:31.176512    4097 checks.go:203] validating availability of port 10250
I0921 11:28:31.176737    4097 checks.go:280] validating the existence of file /etc/kubernetes/pki/ca.crt
I0921 11:28:31.176765    4097 checks.go:430] validating if the connectivity type is via proxy or direct
I0921 11:28:31.176803    4097 checks.go:329] validating the contents of file /proc/sys/net/bridge/bridge-nf-call-iptables
I0921 11:28:31.176849    4097 checks.go:329] validating the contents of file /proc/sys/net/ipv4/ip_forward
I0921 11:28:31.176883    4097 join.go:532] [preflight] Discovering cluster-info
I0921 11:28:31.176917    4097 token.go:80] [discovery] Created cluster-info discovery client, requesting info from "172.31.46.220:6443"
I0921 11:28:31.187676    4097 token.go:118] [discovery] Requesting info from "172.31.46.220:6443" again to validate TLS against the pinned public key
I0921 11:28:31.194531    4097 token.go:135] [discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will use API Server "172.31.46.220:6443"
I0921 11:28:31.194600    4097 discovery.go:52] [discovery] Using provided TLSBootstrapToken as authentication credentials for the join process
I0921 11:28:31.194622    4097 join.go:546] [preflight] Fetching init configuration
I0921 11:28:31.194629    4097 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'
I0921 11:28:31.201909    4097 kubeproxy.go:55] attempting to download the KubeProxyConfiguration from ConfigMap "kube-proxy"
I0921 11:28:31.205146    4097 kubelet.go:74] attempting to download the KubeletConfiguration from ConfigMap "kubelet-config"
I0921 11:28:31.208370    4097 initconfiguration.go:114] skip CRI socket detection, fill with the default CRI socket unix:///var/run/containerd/containerd.sock
I0921 11:28:31.208595    4097 interface.go:432] Looking for default routes with IPv4 addresses
I0921 11:28:31.208617    4097 interface.go:437] Default route transits interface "enX0"
I0921 11:28:31.208751    4097 interface.go:209] Interface enX0 is up
I0921 11:28:31.208803    4097 interface.go:257] Interface "enX0" has 2 addresses :[172.31.36.212/20 fe80::75:41ff:fea5:afb1/64].
I0921 11:28:31.208819    4097 interface.go:224] Checking addr  172.31.36.212/20.
I0921 11:28:31.208829    4097 interface.go:231] IP found 172.31.36.212
I0921 11:28:31.208840    4097 interface.go:263] Found valid IPv4 address 172.31.36.212 for interface "enX0".
I0921 11:28:31.208849    4097 interface.go:443] Found active IP 172.31.36.212
I0921 11:28:31.215002    4097 preflight.go:104] [preflight] Running configuration dependant checks
I0921 11:28:31.215028    4097 controlplaneprepare.go:225] [download-certs] Skipping certs download
```

Step5: Run the given command to verify cluster creation

kubectl get nodes



```
ubuntu@ip-172-31-46-220:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE   VERSION
ip-172-31-36-212   Ready    <none>          47s   v1.29.0
ip-172-31-46-220   Ready    control-plane   16m   v1.29.0
ip-172-31-47-26    Ready    <none>          29s   v1.29.0
```