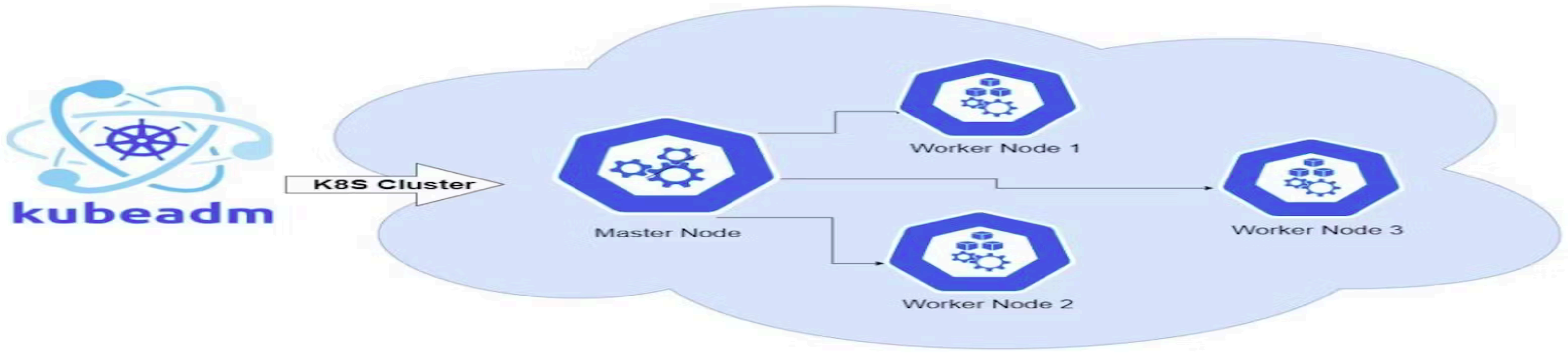
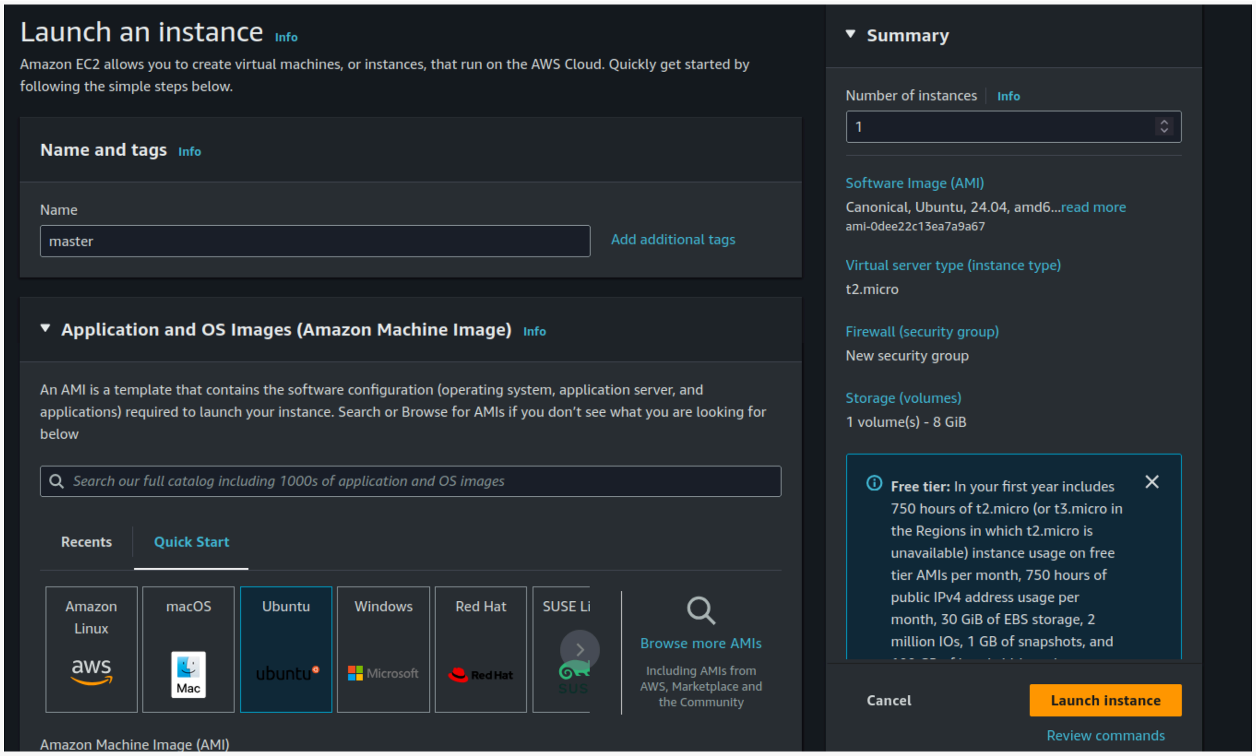


Task : Kubeadm Installation and Configuration.

What is Kubeamd ?

Kubeadm is a tool designed to simplify the process of setting up and managing Kubernetes clusters. It provides a straightforward way to bootstrap a Kubernetes control plane and join nodes to the cluster, ensuring proper configuration and compatibility.

Step 1: Create and Launch a Master Node Instance.



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

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

Recents
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)

Summary

Number of instances Info: 1

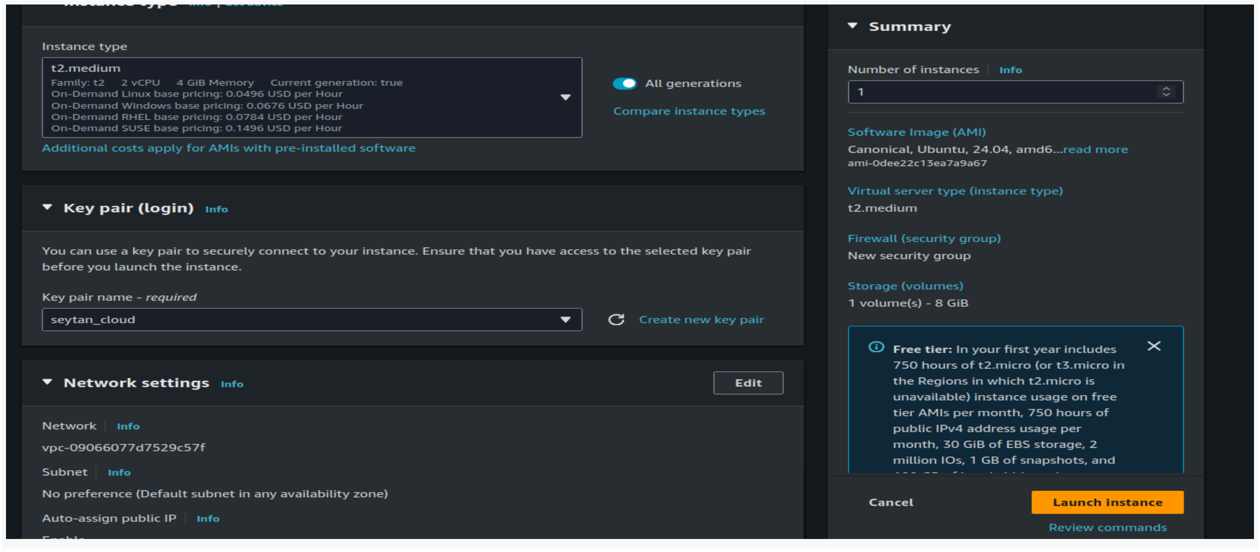
Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more
ami-0dee22c13ea7a9a67

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, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and



Instance type

t2.medium
Family: t2   2 vCPU   4 GiB Memory   Current generation: true
On-Demand Linux base pricing: 0.0496 USD per Hour
On-Demand Windows base pricing: 0.0676 USD per Hour
On-Demand RHEL base pricing: 0.0784 USD per Hour
On-Demand SUSE base pricing: 0.1496 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

seytan_cloud

Create new key pair

Network settings Info

Edit

Network | Info

vpc-09066077d7529c57f

Subnet | Info

No preference (Default subnet in any availability zone)

Auto-assign public IP | Info

Summary

Number of instances | Info

1

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd6...read more
ami-0dee22c13ea7a9a67

Virtual server type (instance type)

t2.medium

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

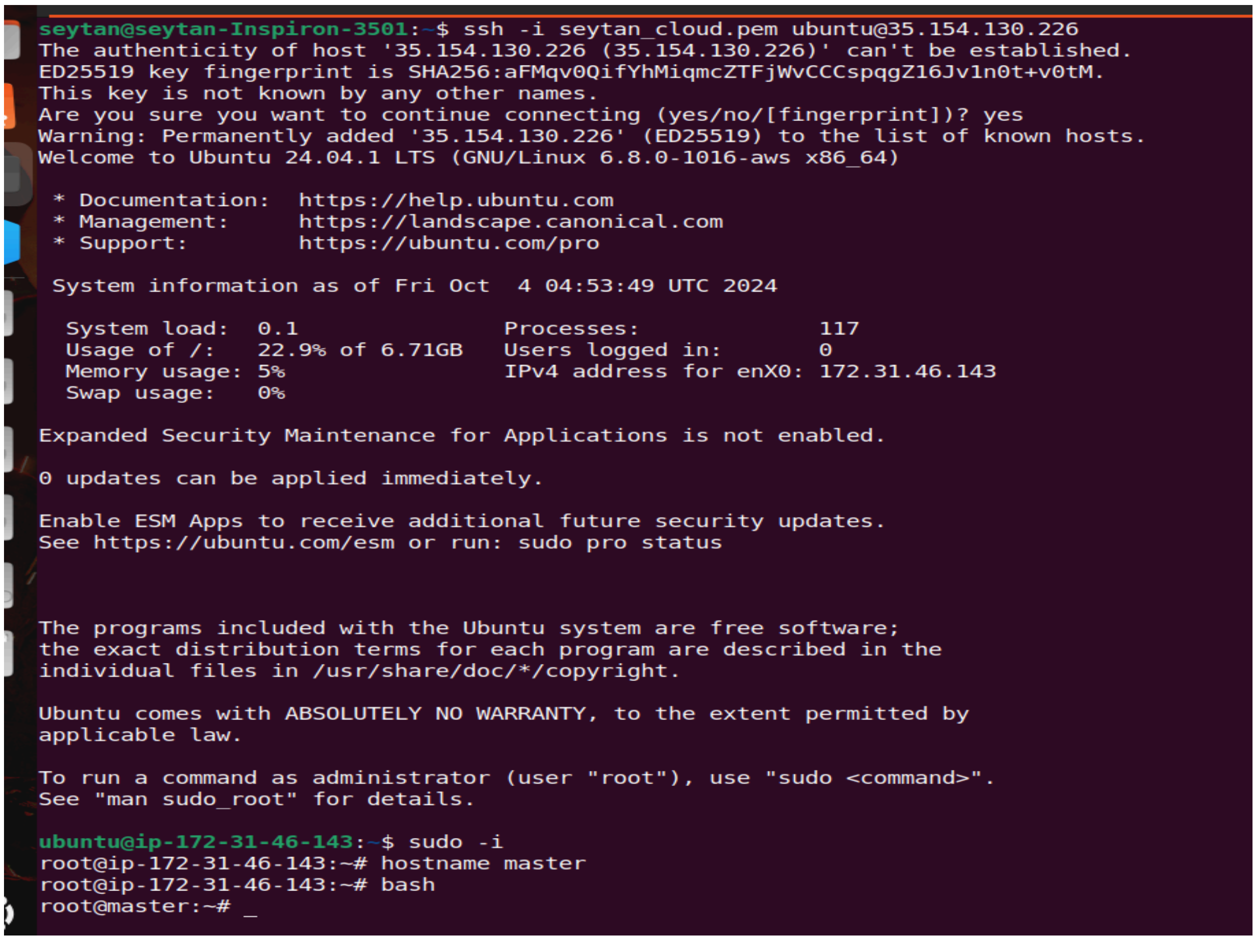
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and

Cancel

Launch instance

Review commands

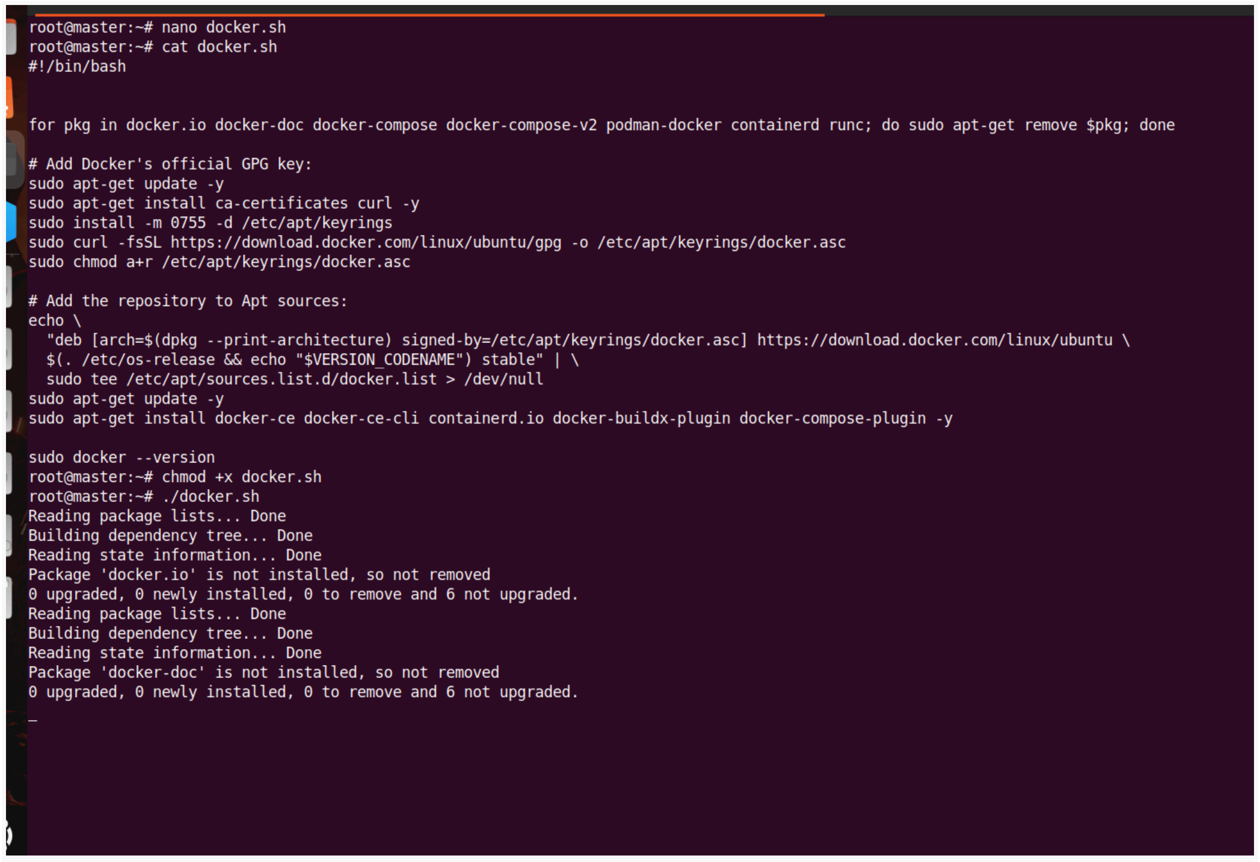
Step 2: Take ssh of it and rename hostname to master using following commands :
sudo -i
hostname master
bash



```
seytan@seytan-Inspiron-3501:~$ ssh -i seytan_cloud.pem ubuntu@35.154.130.226
The authenticity of host '35.154.130.226 (35.154.130.226)' can't be established.
ED25519 key fingerprint is SHA256:aFMqv0QifYhMiqmcZTFjWvCCCspqgZ16Jv1n0t+v0tM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '35.154.130.226' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

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

 System information as of Fri Oct  4 04:53:49 UTC 2024

  System load:  0.1               Processes:             117
  Usage of /:   22.9% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.46.143
  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 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-172-31-46-143:~$ sudo -i
root@ip-172-31-46-143:~# hostname master
root@ip-172-31-46-143:~# bash
root@master:~# _
```

Step 3: Update the master node by running → sudo apt-get update

Step 4: Create a docker.sh file for installing the docker .



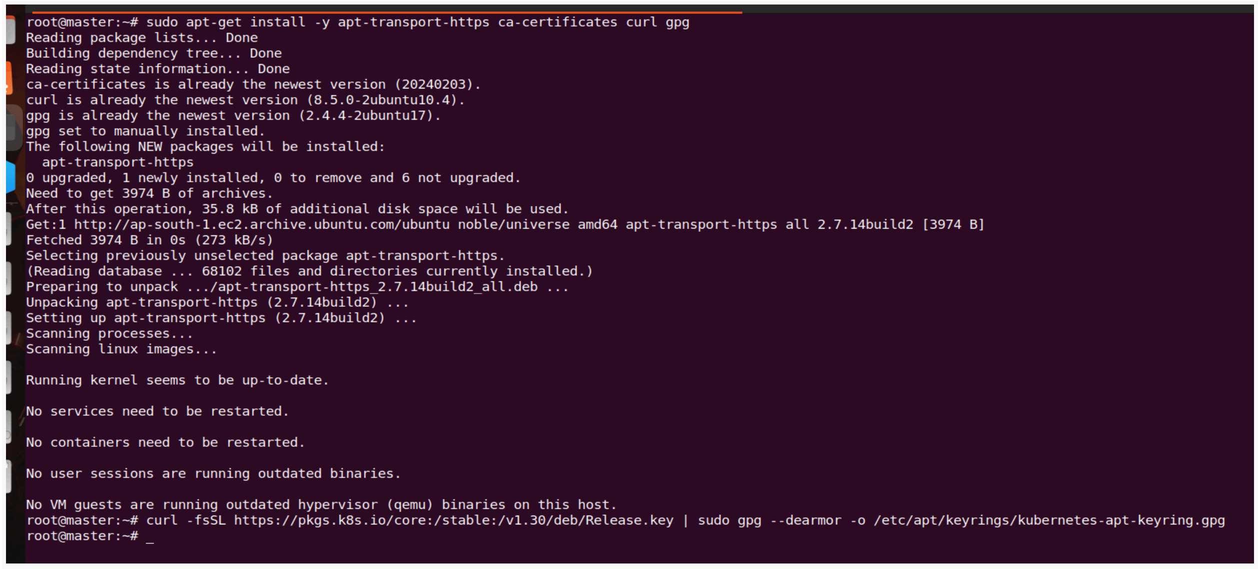
```
root@master:~# nano docker.sh
root@master:~# cat docker.sh
#!/bin/bash


for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done

# Add Docker's official GPG key:
sudo apt-get update -y
sudo apt-get install ca-certificates curl -y
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update -y
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

sudo docker --version
root@master:~# chmod +x docker.sh
root@master:~# ./docker.sh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker.io' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker-doc' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
_
```

After that give execution permission
chmod +x docker.sh
And run the file ./docker.sh

Step 5: Now we need to install the kubeadm so follow the below instructions.

- **sudo 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**

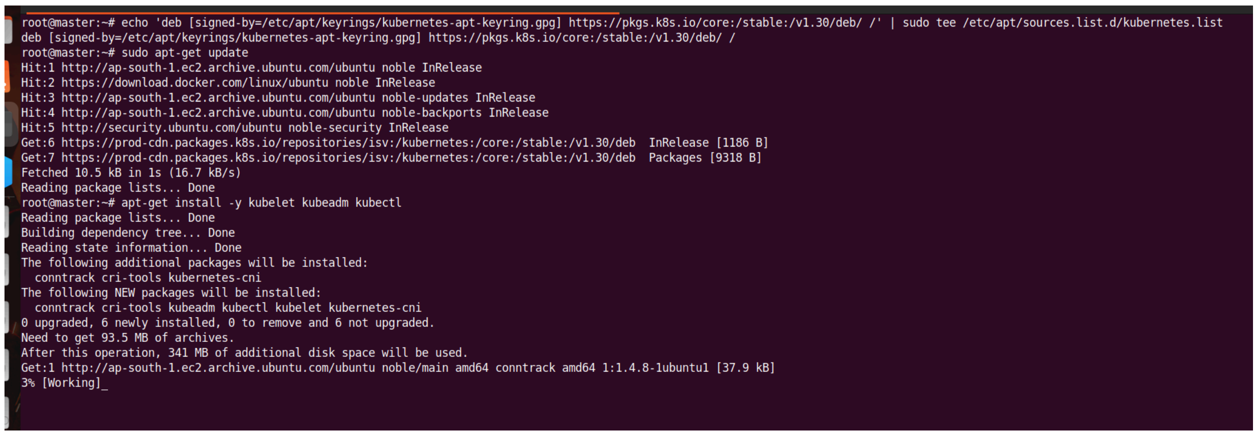


```
root@master:~# sudo apt-get install -y apt-transport-https ca-certificates curl gpg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
curl is already the newest version (8.5.0-2ubuntu10.4).
gpg is already the newest version (2.4.4-2ubuntu17).
gpg set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 3974 B of archives.
After this operation, 35.8 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 apt-transport-https all 2.7.14build2 [3974 B]
Fetched 3974 B in 0s (273 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 68102 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.7.14build2_all.deb ...
Unpacking apt-transport-https (2.7.14build2) ...
Setting up apt-transport-https (2.7.14build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@master:~# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
root@master:~# _
```

- **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**

- **sudo apt-get update**

- **sudo apt-get install -y kubelet kubeadm kubectl**



```
root@master:~# 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
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
root@master:~# sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  Packages [9318 B]
Fetched 10.5 kB in 1s (16.7 kB/s)
Reading package lists... Done
root@master:~# 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 6 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://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
3% [Working]_
```

Step 5: Add essential port in the security group.

Port 6443:

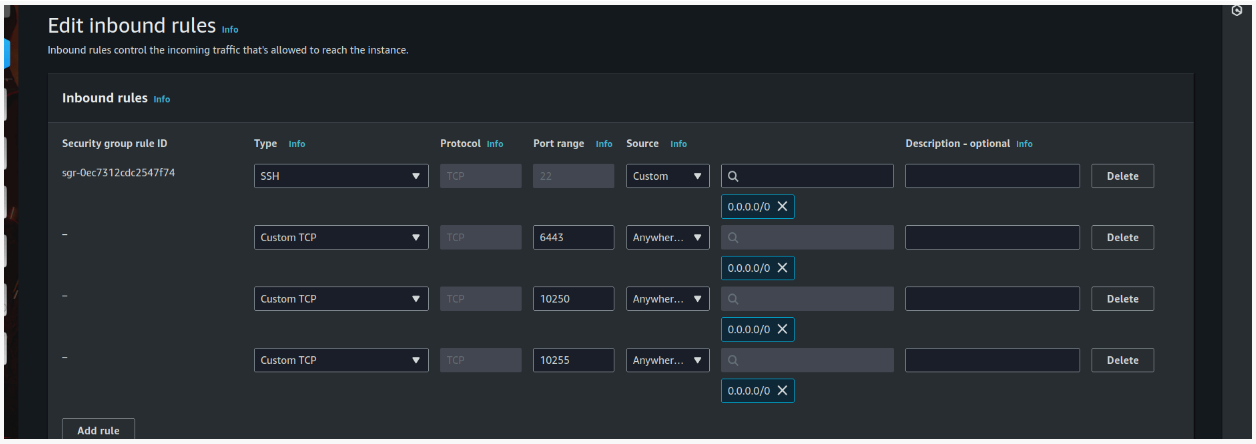
- This is the default port for the Kubernetes API server. It is used for communication between the API server and clients, such as `kubectl`, as well as between different components of the Kubernetes cluster.

Port 10250:

- This port is used by the Kubelet, which is the primary agent that runs on each node in a Kubernetes cluster. It allows for communication between the Kubelet and the Kubernetes API server, enabling the server to retrieve metrics and manage pods on the node.

Port 10255:

- This port is used by the Kubelet for read-only access to the metrics endpoint. It provides information about the status of the node and its running pods, though it is less commonly used due to security concerns since it exposes sensitive data without authentication.


Edit inbound rules Info
Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-0ec7312cdc2547f74	SSH	TCP	22	Custom	0.0.0.0/0		Delete
–	Custom TCP	TCP	6443	Anywher...	0.0.0.0/0		Delete
–	Custom TCP	TCP	10250	Anywher...	0.0.0.0/0		Delete
–	Custom TCP	TCP	10255	Anywher...	0.0.0.0/0		Delete

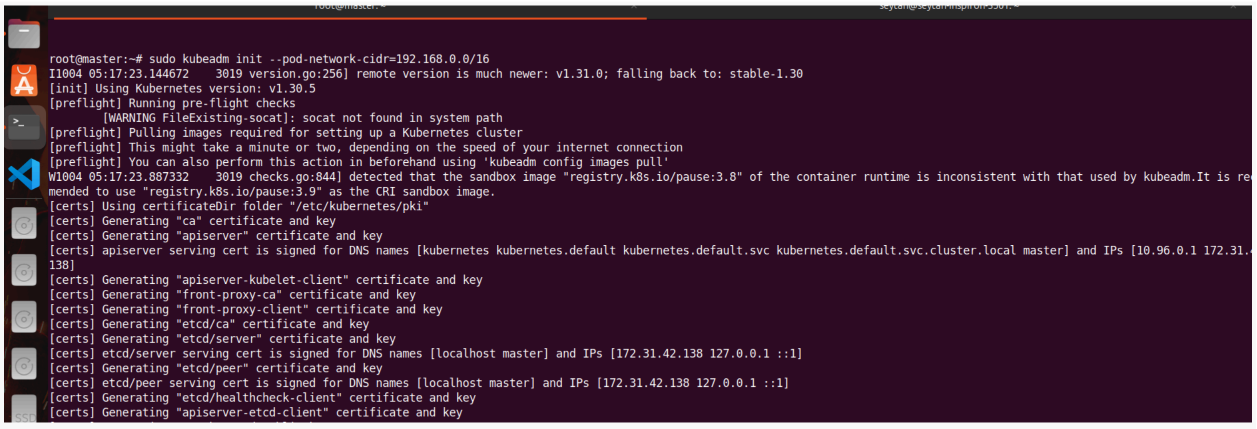
Add rule

Step 6:Run the Following command to initialize the kubeadm

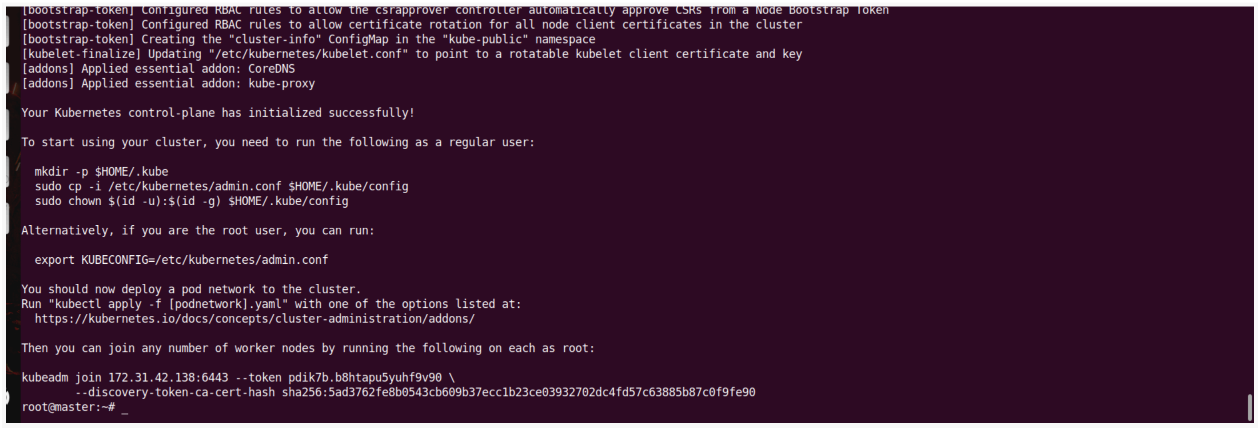
- **sudo kubeadm init --pod-network-cidr=192.168.0.0/16**

The command `sudo kubeadm init --pod-network-cidr=192.168.0.0/16` is used to initialize a Kubernetes cluster with Kubeadm. Here's a brief explanation:

1. **`sudo kubeadm init`: This part of the command runs Kubeadm with elevated privileges to initialize the control plane for the Kubernetes cluster.**
2. **`--pod-network-cidr=192.168.0.0/16`: This flag specifies the CIDR block for the pod network. By setting this, you define the IP address range that will be used for the pods in your cluster, allowing for proper networking configuration. This particular CIDR block is commonly used with network plugins like Calico.**



```
root@master:~# sudo kubeadm init --pod-network-cidr=192.168.0.0/16
I1004 05:17:23.144672    3019 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'
W1004 05:17:23.887332    3019 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 re
mended to use "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 [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local master] and IPs [10.96.0.1 172.31.
138]
[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 [localhost master] and IPs [172.31.42.138 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master] and IPs [172.31.42.138 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
```



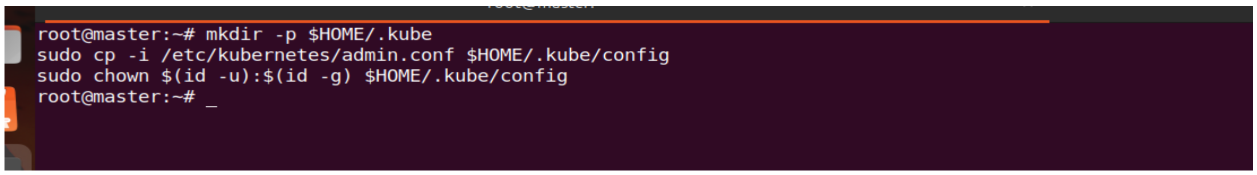
```
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap 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 172.31.42.138:6443 --token pdik7b.b8htapu5yuhf9v90 \
        --discovery-token-ca-cert-hash sha256:5ad3762fe8b0543cb609b37ecc1b23ce03932702dc4fd57c63885b87c0f9fe90
root@master:~# _
```

Step 7: Now run the next commands to start using cluster.

- **mkdir -p $HOME/.kube**

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

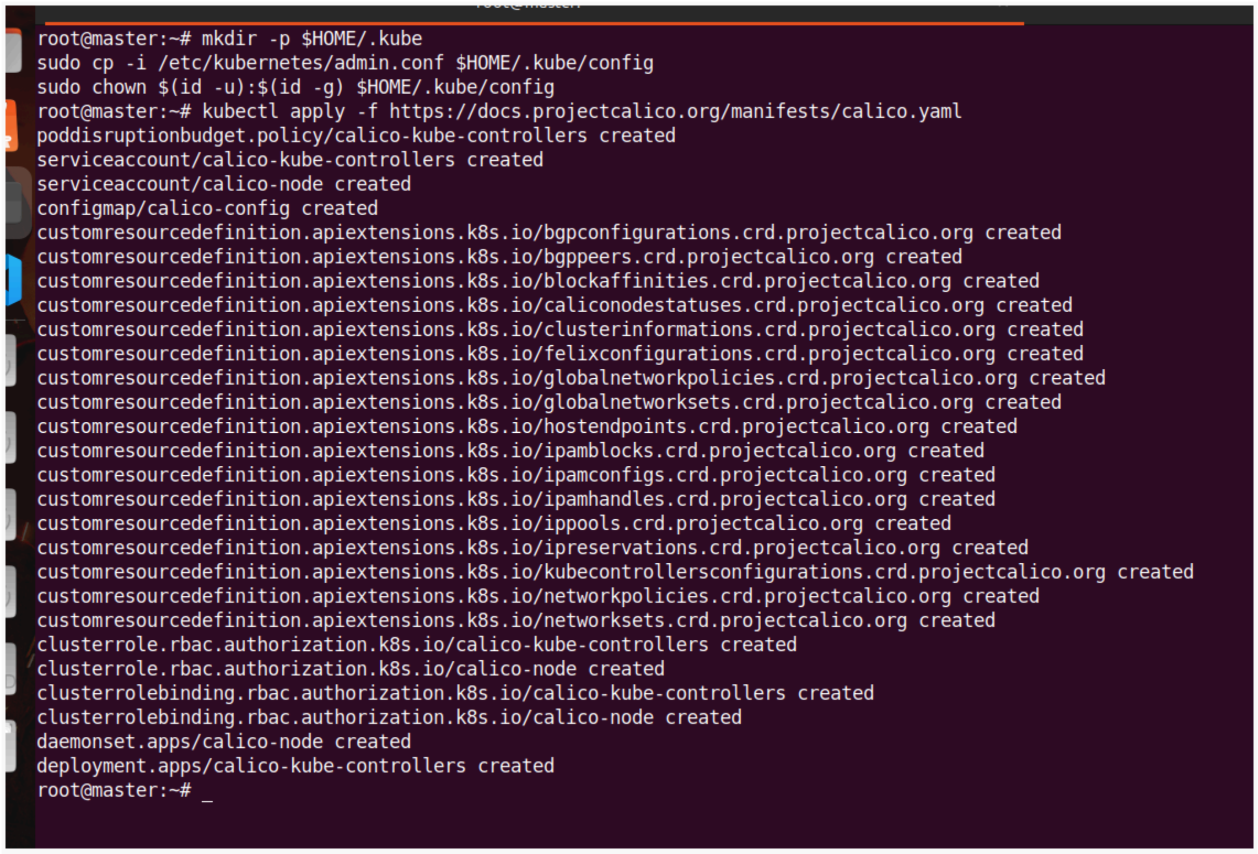
- **sudo chown $(id -u):$(id -g) $HOME/.kube/config**



```
root@master:~# mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@master:~# _
```

- **`kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml`**

The command `kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml` deploys Calico, a networking solution for Kubernetes, providing essential networking features and security policies. It simplifies the installation process by directly applying the manifest to configure the networking setup in your cluster.



```
root@master:~# mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@master:~# kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
root@master:~# _
```

What is Master and worker node ?

→Master Node:

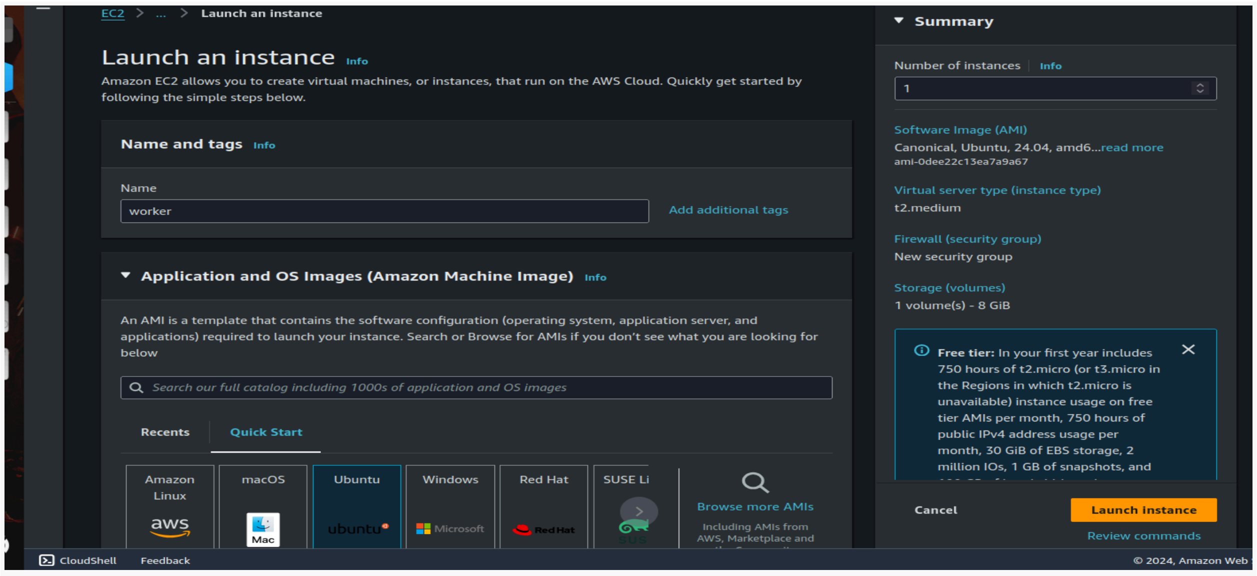
The **master node** manages the Kubernetes control plane, orchestrating cluster operations, handling API requests, and scheduling pods to ensure the desired state of the cluster.

Worker Node:

The **worker node** runs application workloads by hosting pods and communicates with the master node through agents like Kubelet, ensuring that applications are running as expected.

Step 8: Now Create and Launch the worker instance and take its ssh.

Also rename it to worker by hostname command.



EC2 > ... > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: worker

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

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Li | Browse more AMIs

Including AMIs from AWS, Marketplace and

Summary

Number of instances Info: 1

Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more ami-0dee22c13ea7a9a67

Virtual server type (instance type): t2.medium

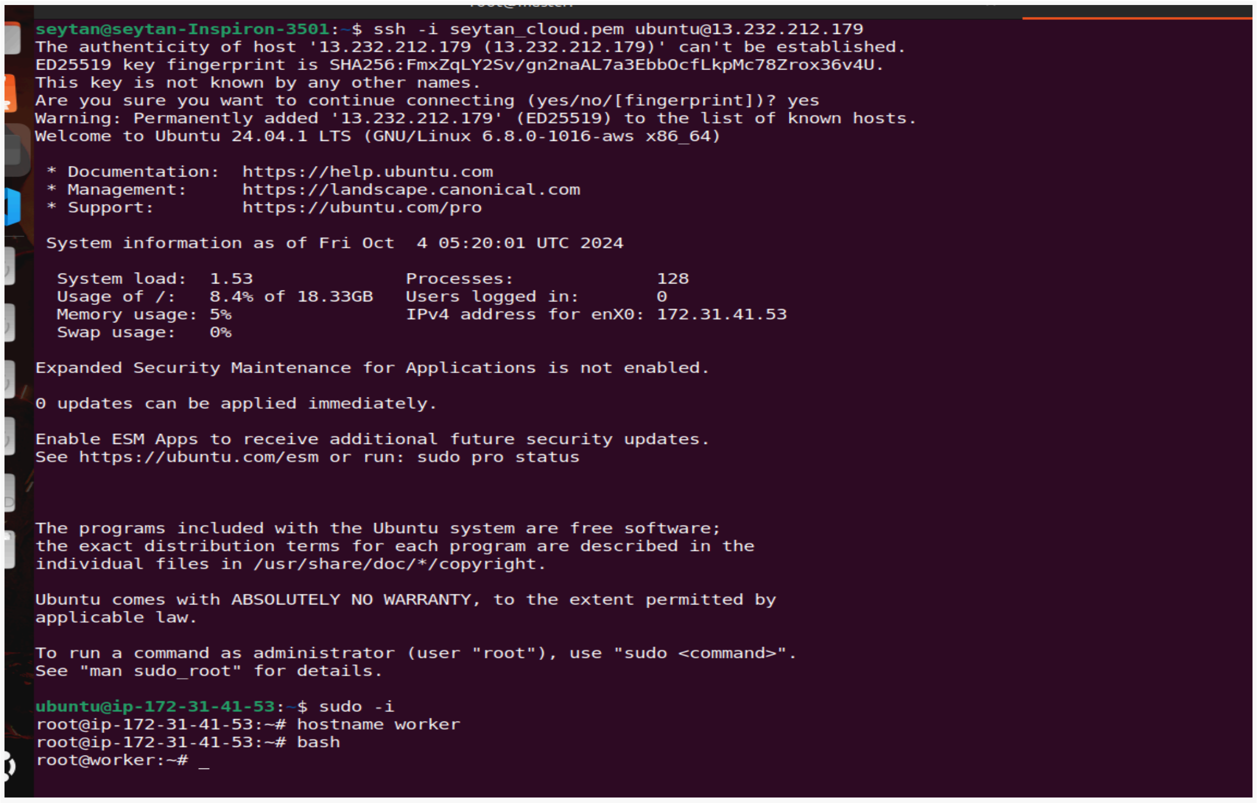
Firewall (security group): New security group

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, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and

Cancel | Launch instance | Review commands

CloudShell Feedback © 2024, Amazon Web



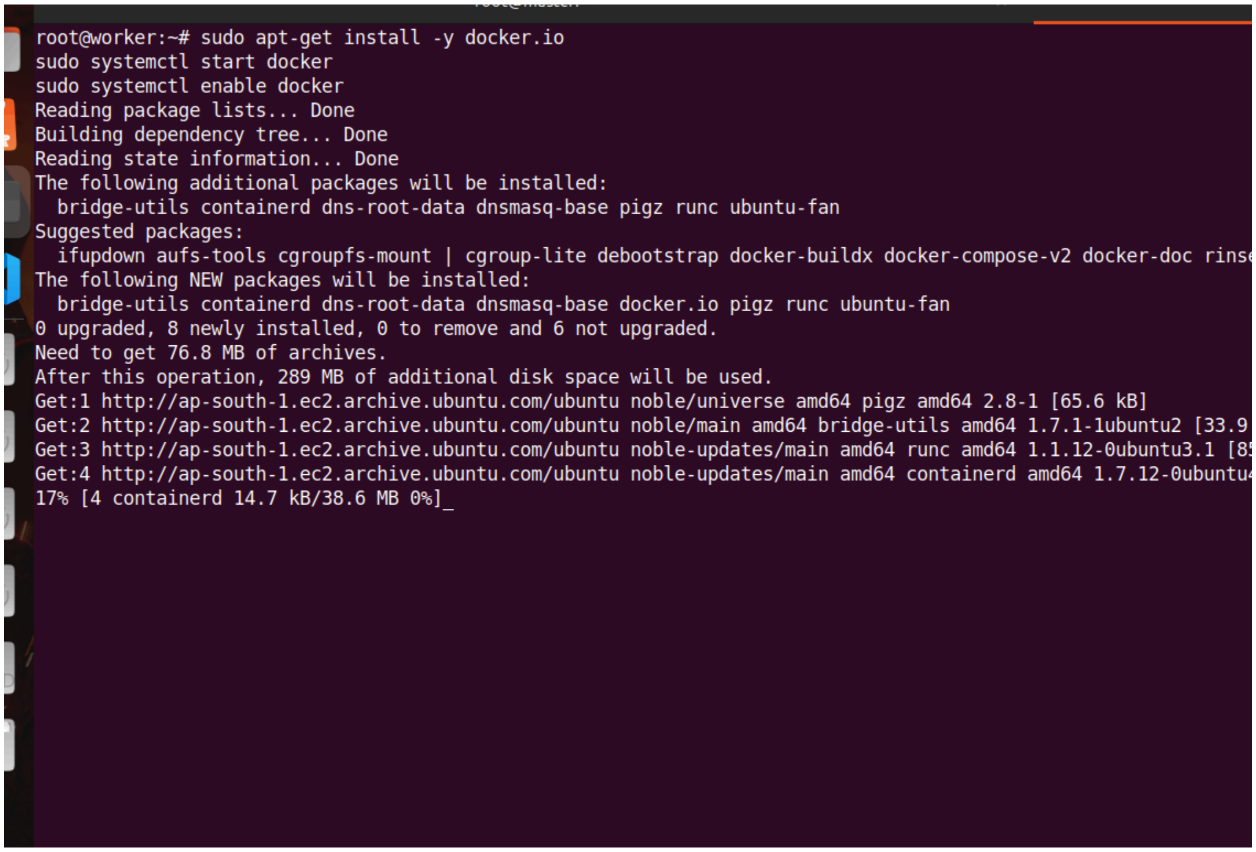
```
seytan@seytan-Inspiron-3501:~$ ssh -i seytan_cloud.pem ubuntu@13.232.212.179
The authenticity of host '13.232.212.179 (13.232.212.179)' can't be established.
ED25519 key fingerprint is SHA256:FmxZqLY2Sv/gn2naAL7a3EbbOcfLkpMc78Zrox36v4U.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.232.212.179' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

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

 System information as of Fri Oct  4 05:20:01 UTC 2024

  System load:  1.53               Processes:             128
  Usage of /:   8.4% of 18.33GB    Users logged in:       0
  Memory usage: 5%                 IPv4 address for enX0: 172.31.41.53
  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 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-172-31-41-53:~$ sudo -i
root@ip-172-31-41-53:~# hostname worker
root@ip-172-31-41-53:~# bash
root@worker:~# _
```

Step 9: Install docker on worker node by the command

→

- **sudo apt-get update**

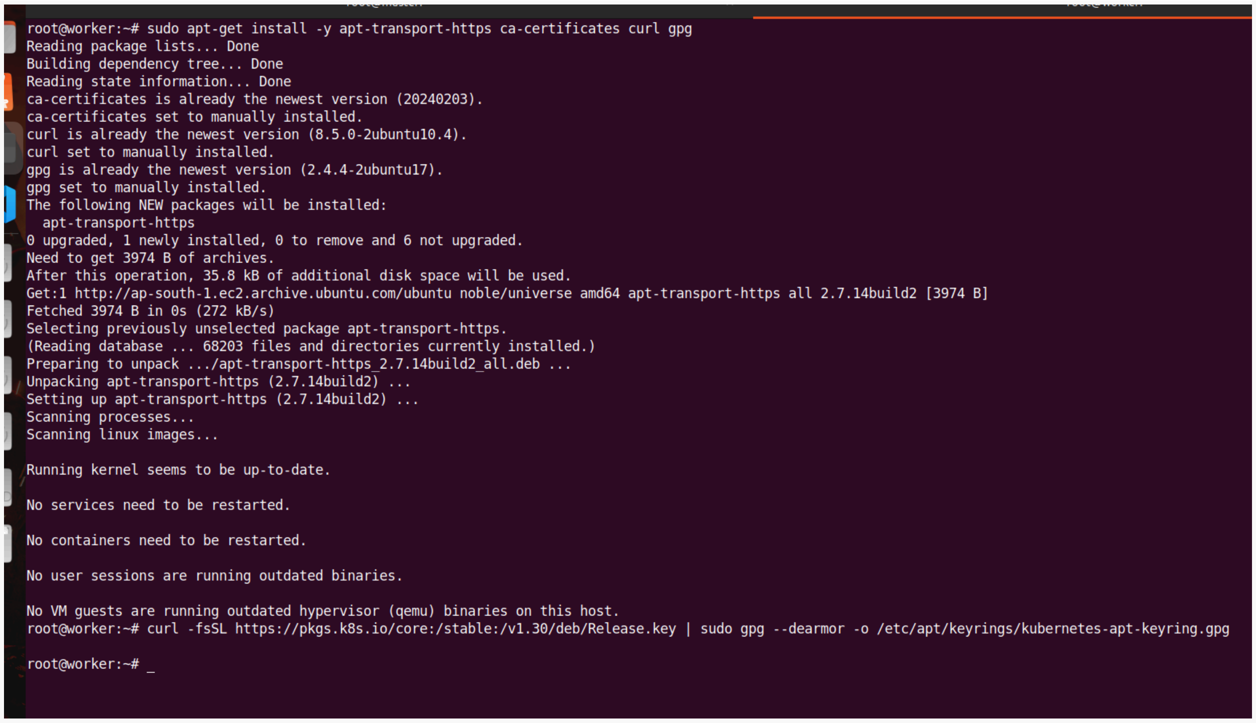
- **sudo apt-get install docker.io**



```
root@worker:~# sudo apt-get install -y docker.io
sudo systemctl start docker
sudo systemctl enable docker
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 6 not upgraded.
Need to get 76.8 MB of archives.
After this operation, 289 MB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu
17% [4 containerd 14.7 kB/38.6 MB 0%]_
```

Step 10: Install and setup kubeadm in worker.

- **sudo 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**

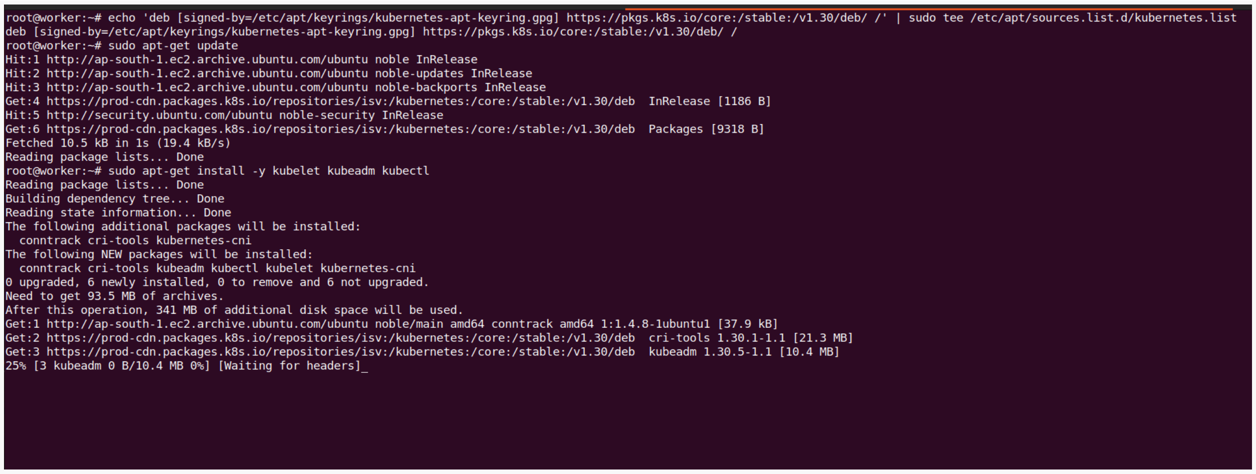


```
root@worker:~# sudo apt-get install -y apt-transport-https ca-certificates curl gpg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.4).
curl 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
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 3974 B of archives.
After this operation, 35.8 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 apt-transport-https all 2.7.14build2 [3974 B]
Fetched 3974 B in 0s (272 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 68203 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.7.14build2_all.deb ...
Unpacking apt-transport-https (2.7.14build2) ...
Setting up apt-transport-https (2.7.14build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@worker:~# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

root@worker:~# _
```

- **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**

- **sudo apt-get update**

- **sudo apt-get install -y kubelet kubeadm kubectl**

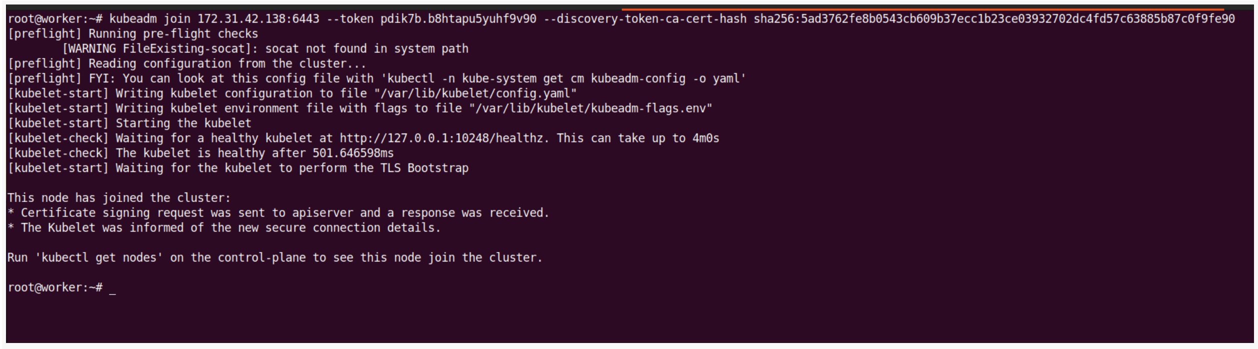


```
root@worker:~# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
root@worker:~# sudo apt-get update
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
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  InRelease [1186 B]
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  Packages [9318 B]
Fetched 10.5 kB in 1s (19.4 kB/s)
Reading package lists... Done
root@worker:~# sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools 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 6 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://ap-south-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]
25% [3 kubeadm 0 B/10.4 MB 0%] [Waiting for headers]_
```

Step 11: Now run this command on master node kubeadm token create --print-join-command

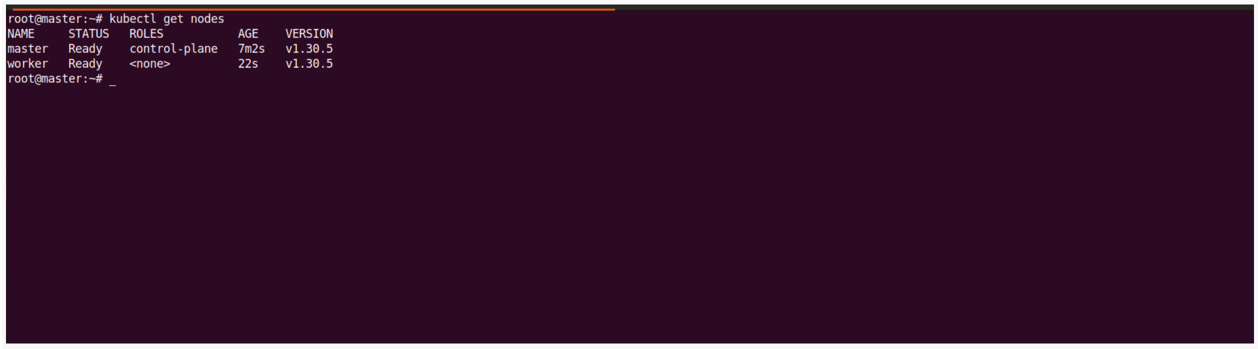
This will return a join command which we have run on worker node .

sudo kubeadm join 172.31.44.95:6443 --token 0047wt.3vi1fu7l1h1cn2dd --discovery-token-ca-cert-hash sha256:54f3c74207d7b8abded0633c28da7767fc2719863211e86789a9c72ae66b360b



```
root@worker:~# kubeadm join 172.31.42.138:6443 --token pdik7b.b8htapu5yuhf9v90 --discovery-token-ca-cert-hash sha256:5ad3762fe8b0543cb609b37ecc1b23ce03932702dc4fd57c63885b87c0f9fe90
[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 501.646598ms
[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@worker:~# _
```

Step 12: Run the kubectl get nodes on master node and see the output.

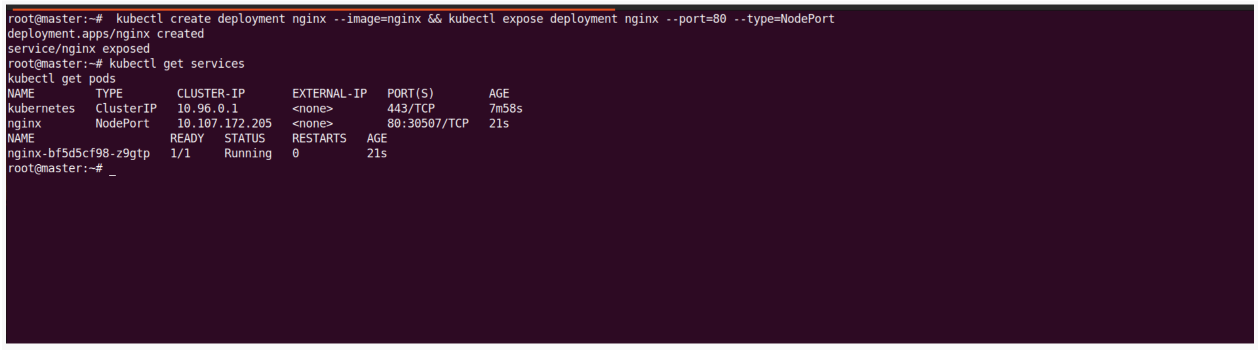


```
root@master:~# kubectl get nodes
NAME     STATUS   ROLES           AGE    VERSION
master   Ready    control-plane   7m2s   v1.30.5
worker   Ready    <none>          22s    v1.30.5
root@master:~# _
```

Step 13: Now lets test our cluster by running the nginx deployment.

- **kubectl create deployment nginx --image=nginx**

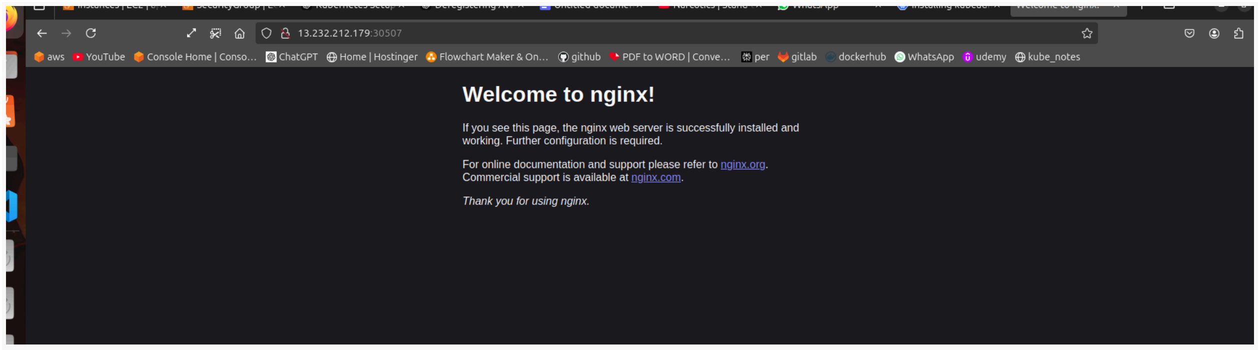
- **kubectl expose deployment nginx --port=80 --type=NodePort**



```
root@master:~#  kubectl create deployment nginx --image=nginx && kubectl expose deployment nginx --port=80 --type=NodePort
deployment.apps/nginx created
service/nginx exposed
root@master:~# kubectl get services
kubectl get pods
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP        7m58s
nginx        NodePort    10.107.172.205   <none>        80:30507/TCP   21s
NAME                     READY   STATUS    RESTARTS   AGE
nginx-bf5d5cf98-z9gtp   1/1     Running   0          21s
root@master:~# _
```

Step 14: Add the port assigned to the nginx which is 30507 in security group of the instances.

After this paste the workernode_ip:nodeport in the web browser to see the output .



13.232.212.179:30507

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Conclusion:

In this task, we successfully set up a Kubernetes cluster using Kubeadm, initializing the master node and configuring the pod network. We deployed the necessary network plugin for efficient communication between pods and verified the installation by deploying application workloads on the worker nodes. This foundational setup enables the management of containerized applications at scale, facilitating seamless orchestration and enhancing overall operational efficiency.