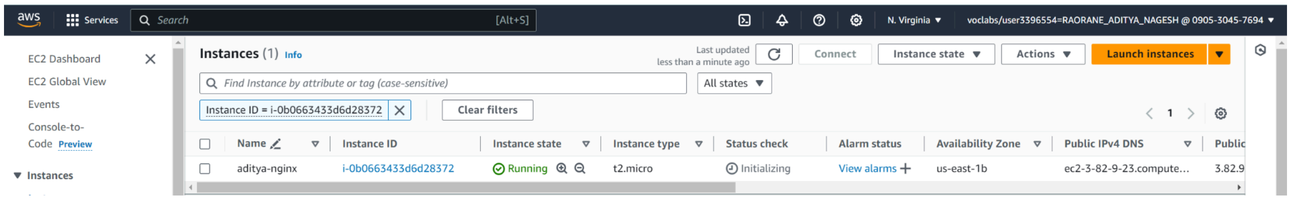
Name: Aditya Nagesh Raorane Class: D15C/ Batch B Roll No: 44

EXPERIMENT No.4

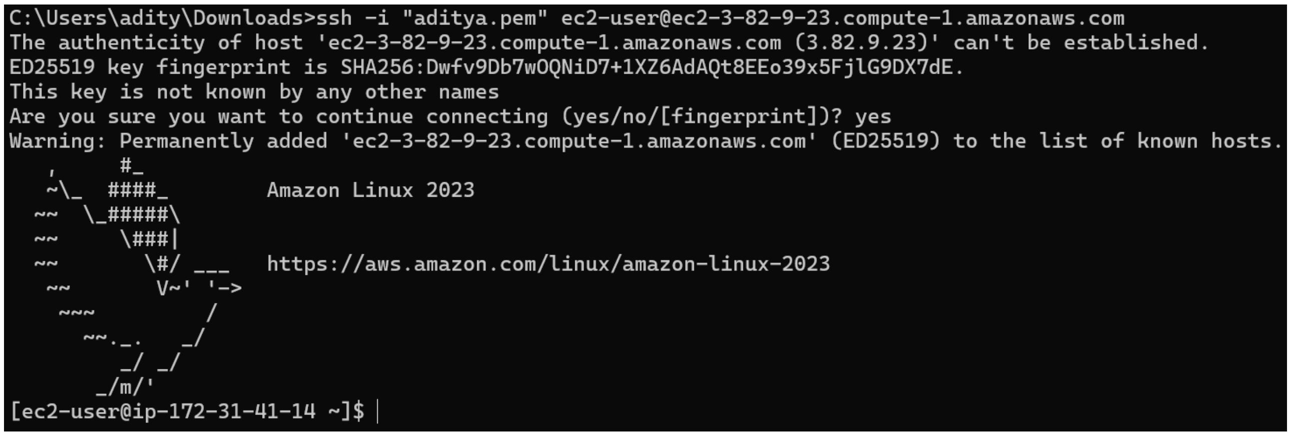
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

1. Select **Amazon linux** as **OS** image and create an AWS **EC2 instance** named **aditya-nignx**.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
☐	aditya-nginx	i-0b0663433d6d28372	Running	t2.micro	Initializing	View alarms +	us-east-1b	ec2-3-82-9-23.compute...	3.82.9

2. Make ssh connection in terminal

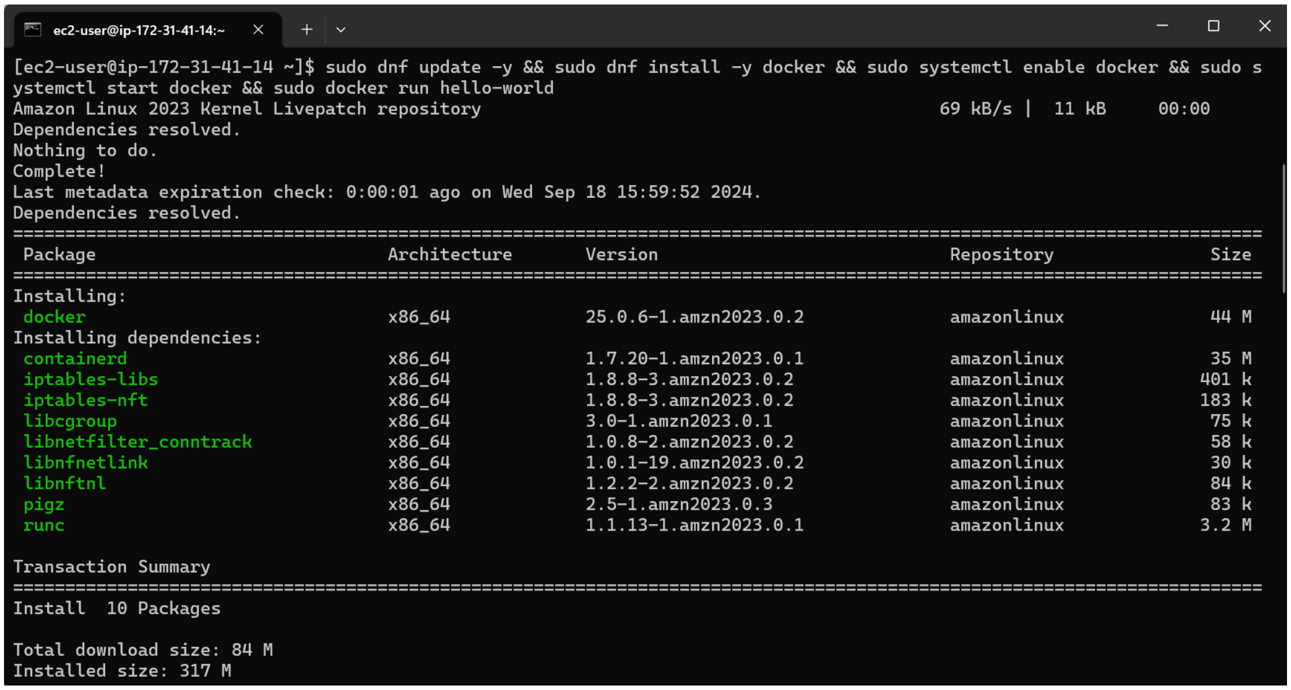


```
C:\Users\adity\Downloads>ssh -i "aditya.pem" ec2-user@ec2-3-82-9-23.compute-1.amazonaws.com
The authenticity of host 'ec2-3-82-9-23.compute-1.amazonaws.com (3.82.9.23)' can't be established.
ED25519 key fingerprint is SHA256:Dwfv9Db7wOQNiD7+1XZ6AdAQt8EEo39x5FjlG9DX7dE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-82-9-23.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-41-14 ~]$
```

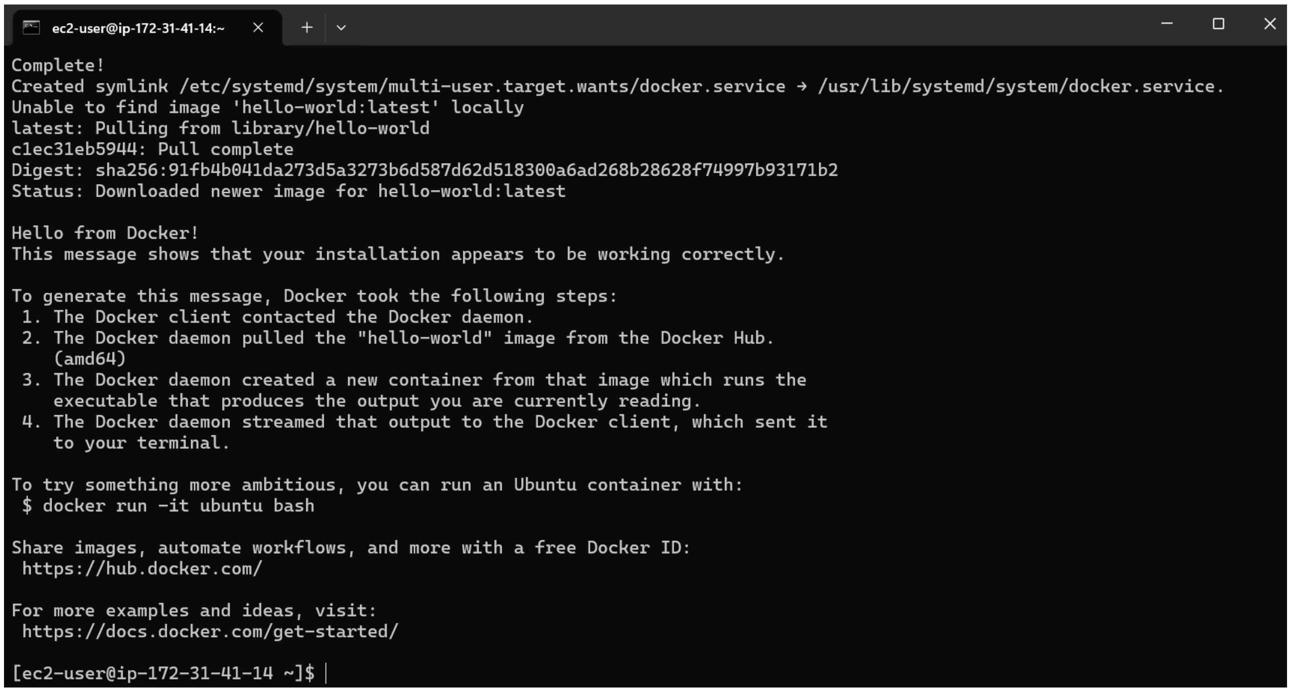
Name: Aditya Nagesh Raorane Class: D15C/ Batch B Roll No: 44

3. Install Docker

- sudo dnf update -y && sudo dnf install -y docker && sudo systemctl enable docker && sudo systemctl start docker && sudo docker run hello-world



```
[ec2-user@ip-172-31-41-14 ~]$ sudo dnf update -y && sudo dnf install -y docker && sudo systemctl enable docker && sudo s
ystemctl start docker && sudo docker run hello-world
Amazon Linux 2023 Kernel Livepatch repository                               69 kB/s |  11 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:00:01 ago on Wed Sep 18 15:59:52 2024.
Dependencies resolved.
========================================================================================================================
 Package                            Architecture       Version                            Repository               Size
========================================================================================================================
Installing:
 docker                             x86_64             25.0.6-1.amzn2023.0.2              amazonlinux              44 M
Installing dependencies:
 containerd                         x86_64             1.7.20-1.amzn2023.0.1              amazonlinux              35 M
 iptables-libs                      x86_64             1.8.8-3.amzn2023.0.2               amazonlinux             401 k
 iptables-nft                       x86_64             1.8.8-3.amzn2023.0.2               amazonlinux             183 k
 libcgroup                          x86_64             3.0-1.amzn2023.0.1                 amazonlinux              75 k
 libnetfilter_conntrack             x86_64             1.0.8-2.amzn2023.0.2               amazonlinux              58 k
 libnfnetlink                       x86_64             1.0.1-19.amzn2023.0.2              amazonlinux              30 k
 libnftnl                           x86_64             1.2.2-2.amzn2023.0.2               amazonlinux              84 k
 pigz                               x86_64             2.5-1.amzn2023.0.3                 amazonlinux              83 k
 runc                               x86_64             1.1.13-1.amzn2023.0.1              amazonlinux             3.2 M

Transaction Summary
========================================================================================================================
Install  10 Packages

Total download size: 84 M
Installed size: 317 M
```

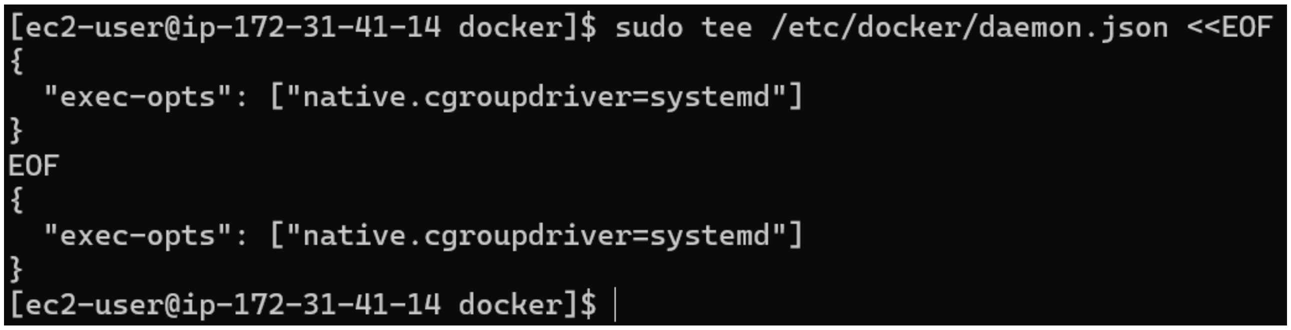


```
Complete!
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:91fb4b041da273d5a3273b6d587d62d518300a6ad268b28628f74997b93171b2
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

[ec2-user@ip-172-31-41-14 ~]$
```

Then, configure cgroup in a daemon.json file.This allows kubernetes to manage host more efficiently.

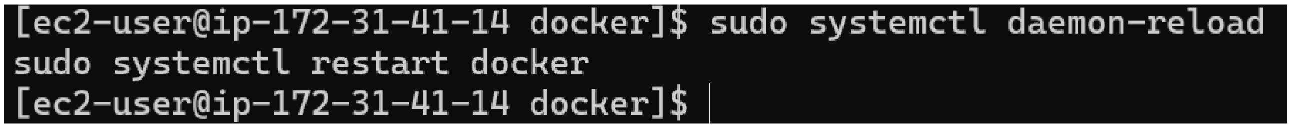
- cd /etc/docker

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



```
[ec2-user@ip-172-31-41-14 docker]$ sudo tee /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
[ec2-user@ip-172-31-41-14 docker]$
```

- sudo systemctl daemon-reload
- sudo systemctl restart docker



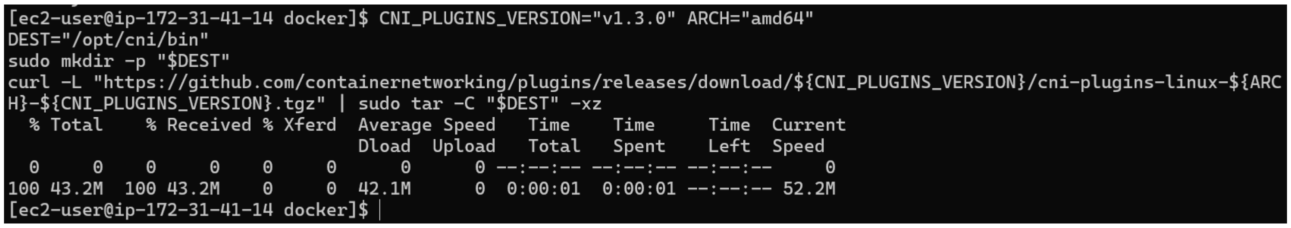
```
[ec2-user@ip-172-31-41-14 docker]$ sudo systemctl daemon-reload
sudo systemctl restart docker
[ec2-user@ip-172-31-41-14 docker]$
```

4. Install Kubernetes

Note: I'm directly installing binary package you may install from package repository of your distribution

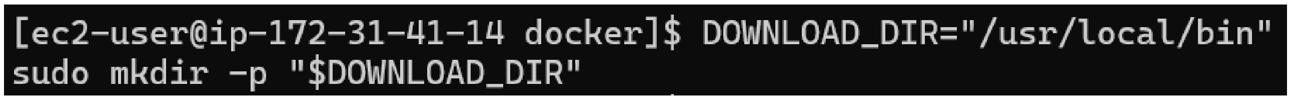
Install CNI plugins (required for most pod network):

CNI_PLUGINS_VERSION="v1.3.0" ARCH="amd64"
DEST="/opt/cni/bin"
sudo mkdir -p "$DEST"
curl -L "https://github.com/containernetworking/plugins/releases/download/${CNI_PLUGINS_VERSION}/cni-plugins-linux-${ARCH}-${CNI_PLUGINS_VERSION}.tgz" | sudo tar -C "$DEST" -xz



```
[ec2-user@ip-172-31-41-14 docker]$ CNI_PLUGINS_VERSION="v1.3.0" ARCH="amd64"
DEST="/opt/cni/bin"
sudo mkdir -p "$DEST"
curl -L "https://github.com/containernetworking/plugins/releases/download/${CNI_PLUGINS_VERSION}/cni-plugins-linux-${ARC
H}-${CNI_PLUGINS_VERSION}.tgz" | sudo tar -C "$DEST" -xz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 43.2M  100 43.2M    0     0  42.1M      0  0:00:01  0:00:01 --:--:-- 52.2M
[ec2-user@ip-172-31-41-14 docker]$ 
```

Define the directory to download command files:
DOWNLOAD_DIR="/usr/local/bin"
sudo mkdir -p "$DOWNLOAD_DIR"

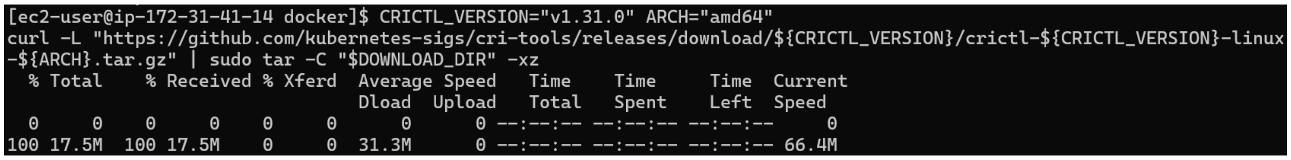


```
[ec2-user@ip-172-31-41-14 docker]$ DOWNLOAD_DIR="/usr/local/bin"
sudo mkdir -p "$DOWNLOAD_DIR"
```

Optionally install crictl (required for interaction with the Container Runtime Interface (CRI), optional for kubeadm):

CRICTL_VERSION="v1.31.0"
ARCH="amd64"

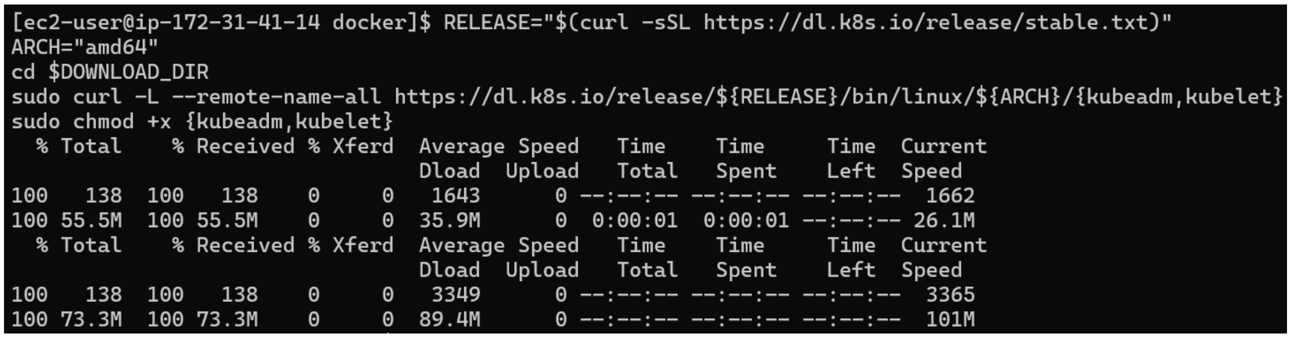
curl -L
"https://github.com/kubernetes-sigs/cri-tools/releases/download/${CRICTL_VERSION}/
c rictl-${CRICTL_VERSION}-linux-${ARCH}.tar.gz" | sudo tar -C $DOWNLOAD_DIR -xz



```
[ec2-user@ip-172-31-41-14 docker]$ CRICTL_VERSION="v1.31.0" ARCH="amd64"
curl -L "https://github.com/kubernetes-sigs/cri-tools/releases/download/${CRICTL_VERSION}/crictl-${CRICTL_VERSION}-linux
-${ARCH}.tar.gz" | sudo tar -C "$DOWNLOAD_DIR" -xz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 17.5M  100 17.5M    0     0  31.3M      0 --:--:-- --:--:-- --:--:-- 66.4M
```

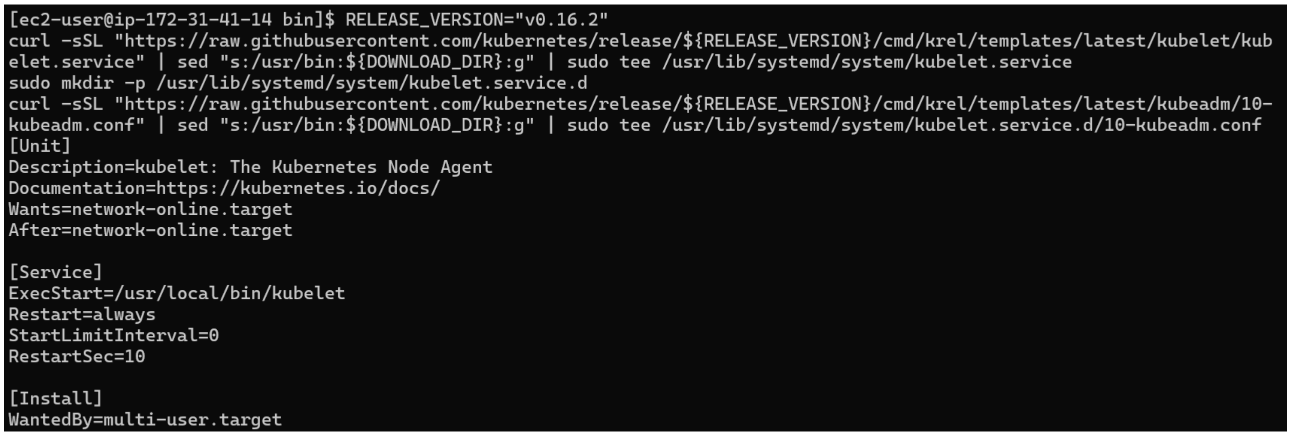
Install `kubeadm`, `kubelet` and add a `kubelet` systemd service:

RELEASE="$(curl -sSL
https://dl.k8s.io/release/stable.txt)" ARCH="amd64"
cd $DOWNLOAD_DIR
sudo curl -L --remote-name-all
https://dl.k8s.io/release/${RELEASE}/bin/linux/${ARCH}/{kubeadm,kubelet
} sudo chmod +x {kubeadm,kubelet}

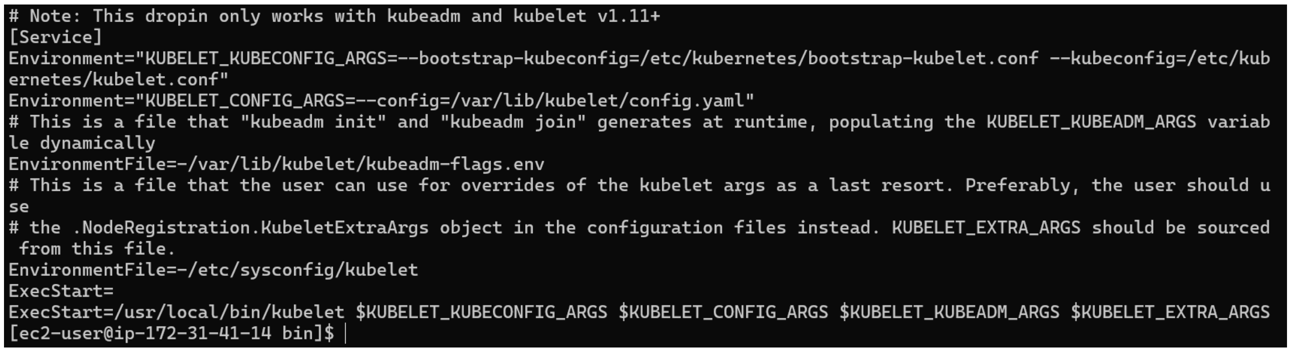


```
[ec2-user@ip-172-31-41-14 docker]$ RELEASE="$(curl -sSL https://dl.k8s.io/release/stable.txt)"
ARCH="amd64"
cd $DOWNLOAD_DIR
sudo curl -L --remote-name-all https://dl.k8s.io/release/${RELEASE}/bin/linux/${ARCH}/{kubeadm,kubelet}
sudo chmod +x {kubeadm,kubelet}
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0   1643      0 --:--:-- --:--:-- --:--:--  1662
100 55.5M  100 55.5M    0     0  35.9M      0  0:00:01  0:00:01 --:--:-- 26.1M
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0   3349      0 --:--:-- --:--:-- --:--:--  3365
100 73.3M  100 73.3M    0     0  89.4M      0 --:--:-- --:--:-- --:--:--  101M
```

RELEASE_VERSION="v0.16.2"
curl -sSL
"https://raw.githubusercontent.com/kubernetes/release/${RELEASE_VERSION}/cmd/kr
el/templates/latest/kubelet/kubelet.service" | sed "s:/usr/bin:${DOWNLOAD_DIR}:g" |
sudo tee /usr/lib/systemd/system/kubelet.service
sudo mkdir -p /usr/lib/systemd/system/kubelet.service.d
curl -sSL
"https://raw.githubusercontent.com/kubernetes/release/${RELEASE_VERSION}/cmd/kr
el/templates/latest/kubeadm/10-kubeadm.conf" | sed
"s:/usr/bin:${DOWNLOAD_DIR}:g"
| sudo tee /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf



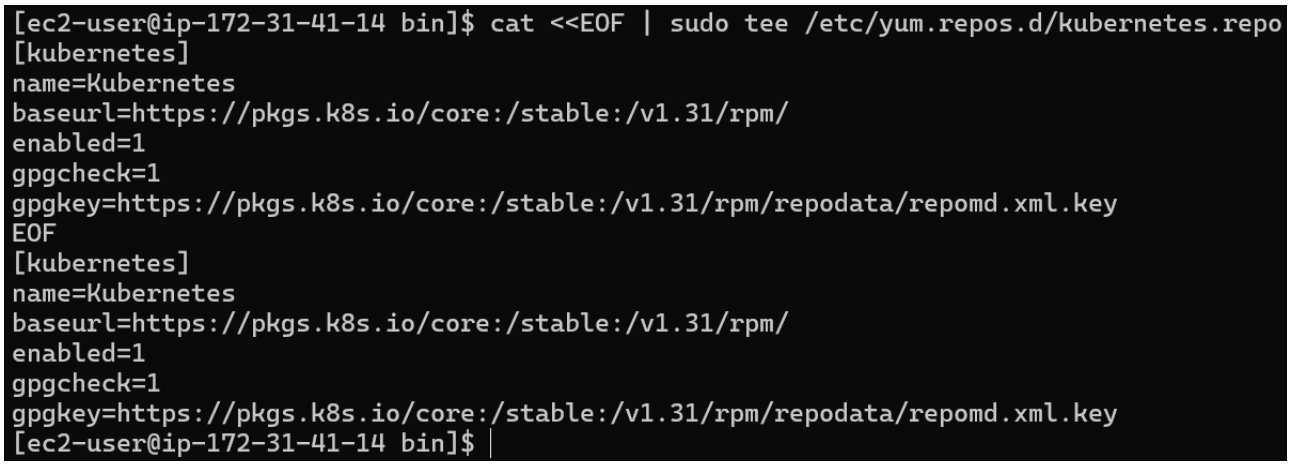
```
[ec2-user@ip-172-31-41-14 bin]$ RELEASE_VERSION="v0.16.2"
curl -sSL "https://raw.githubusercontent.com/kubernetes/release/${RELEASE_VERSION}/cmd/krel/templates/latest/kubelet/kub
elet.service" | sed "s:/usr/bin:${DOWNLOAD_DIR}:g" | sudo tee /usr/lib/systemd/system/kubelet.service
sudo mkdir -p /usr/lib/systemd/system/kubelet.service.d
curl -sSL "https://raw.githubusercontent.com/kubernetes/release/${RELEASE_VERSION}/cmd/krel/templates/latest/kubeadm/10-
kubeadm.conf" | sed "s:/usr/bin:${DOWNLOAD_DIR}:g" | sudo tee /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf
[Unit]
Description=kubelet: The Kubernetes Node Agent
Documentation=https://kubernetes.io/docs/
Wants=network-online.target
After=network-online.target

[Service]
ExecStart=/usr/local/bin/kubelet
Restart=always
StartLimitInterval=0
RestartSec=10

[Install]
WantedBy=multi-user.target
```



```
# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kub
ernetes/kubelet.conf"
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, populating the KUBELET_KUBEADM_ARGS variab
le dynamically
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a last resort. Preferably, the user should u
se
# the .NodeRegistration.KubeletExtraArgs object in the configuration files instead. KUBELET_EXTRA_ARGS should be sourced
 from this file.
EnvironmentFile=-/etc/sysconfig/kubelet
ExecStart=
ExecStart=/usr/local/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS
[ec2-user@ip-172-31-41-14 bin]$
```

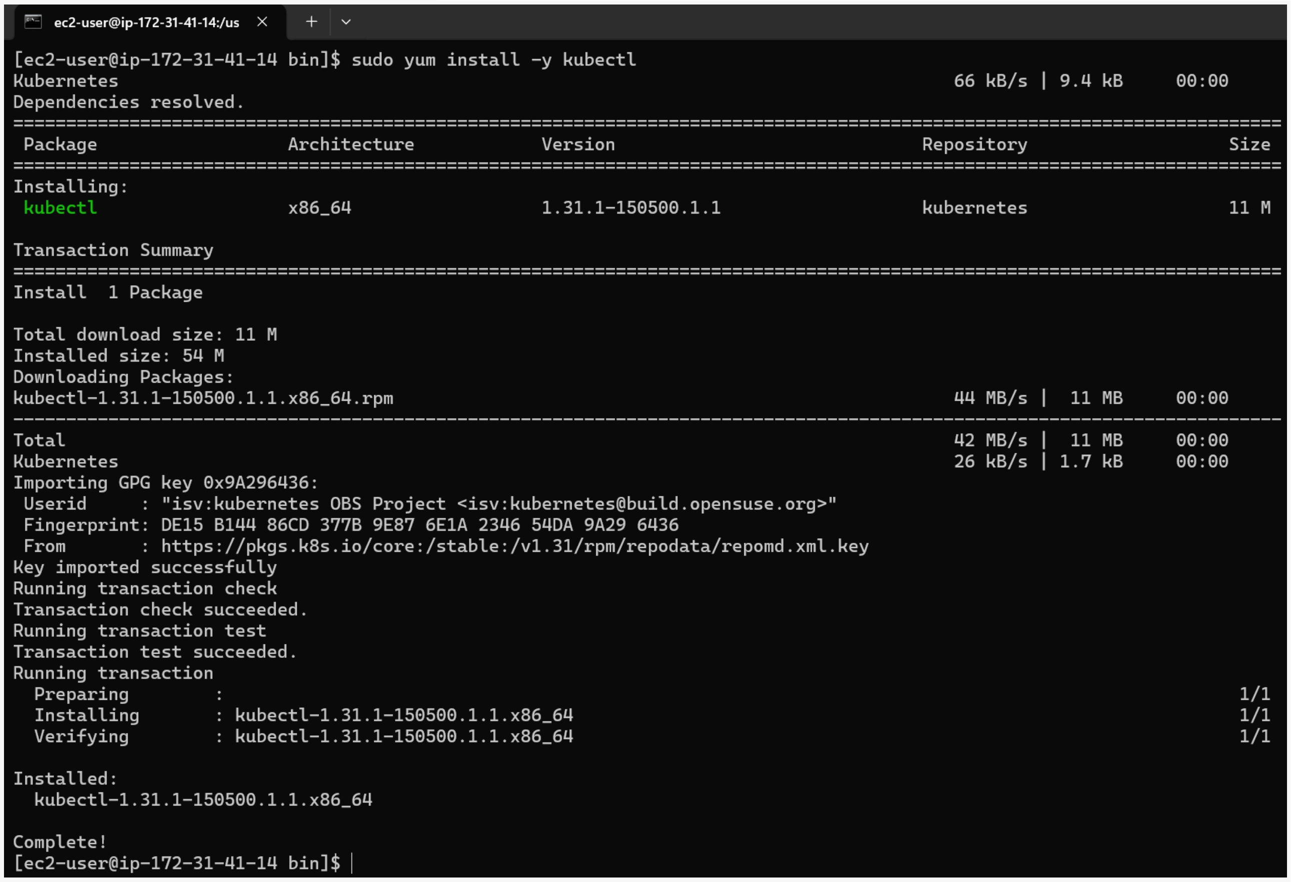
Now we need to install kubectl

```
Set up repository:

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repo
md.xml.key
EOF
```

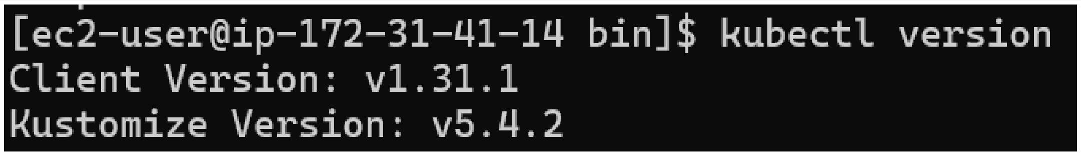


```
[ec2-user@ip-172-31-41-14 bin]$ cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
[ec2-user@ip-172-31-41-14 bin]$
```

`sudo yum install -y kubectl`



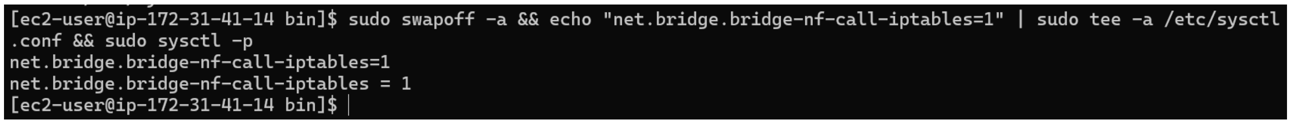
```
[ec2-user@ip-172-31-41-14 bin]$ sudo yum install -y kubectl
Kubernetes                                                                      66 kB/s | 9.4 kB     00:00
Dependencies resolved.
================================================================================================================
 Package                 Architecture            Version                            Repository             Size
================================================================================================================
Installing:
 kubectl                 x86_64                  1.31.1-150500.1.1                  kubernetes             11 M

Transaction Summary
================================================================================================================
Install  1 Package

Total download size: 11 M
Installed size: 54 M
Downloading Packages:
kubectl-1.31.1-150500.1.1.x86_64.rpm                                            44 MB/s |  11 MB     00:00
----------------------------------------------------------------------------------------------------------------
Total                                                                           42 MB/s |  11 MB     00:00
Kubernetes                                                                      26 kB/s | 1.7 kB     00:00
Importing GPG key 0x9A296436:
 Userid     : "isv:kubernetes OBS Project <isv:kubernetes@build.opensuse.org>"
 Fingerprint: DE15 B144 86CD 377B 9E87 6E1A 2346 54DA 9A29 6436
 From       : https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                        1/1
  Installing       : kubectl-1.31.1-150500.1.1.x86_64                                                       1/1
  Verifying        : kubectl-1.31.1-150500.1.1.x86_64                                                       1/1

Installed:
  kubectl-1.31.1-150500.1.1.x86_64

Complete!
[ec2-user@ip-172-31-41-14 bin]$
```



```
[ec2-user@ip-172-31-41-14 bin]$ kubectl version
Client Version: v1.31.1
Kustomize Version: v5.4.2
```

```
We have installed successfully installed  kubernetes
```

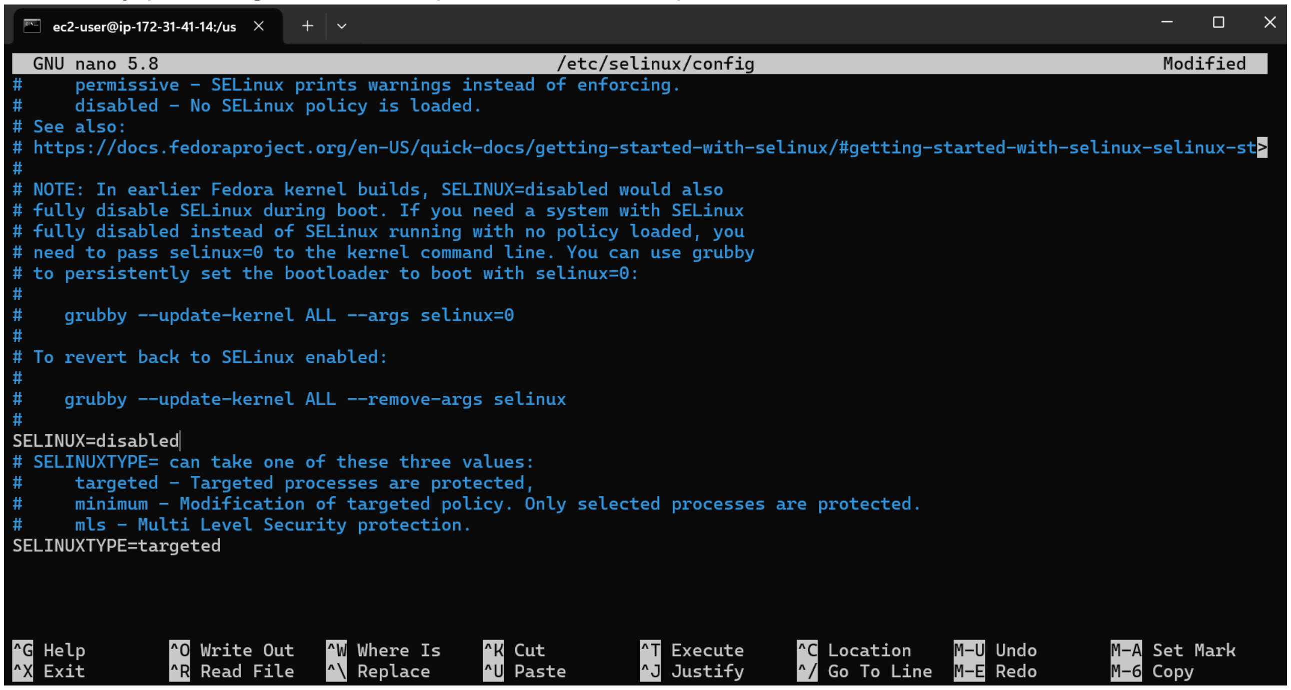
```
After installing Kubernetes, we need to configure internet
options to allow bridging.
sudo swapoff -a
&& echo
"net.bridge.bridg
e-nf-call-iptable
s=1" | sudo tee
-a
/etc/sysctl.conf
&& sudo sysctl -p
```



```
[ec2-user@ip-172-31-41-14 bin]$ sudo swapoff -a && echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl
.conf && sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-41-14 bin]$ |
```

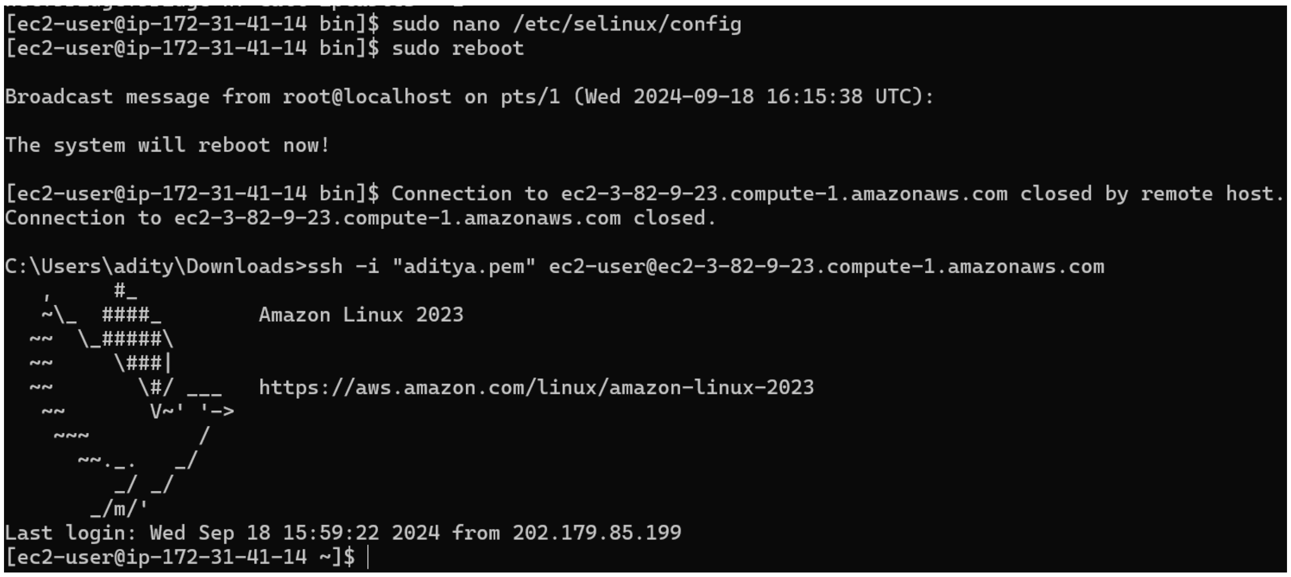
Disable SELINUX

Type **sudo nano /etc/selinux/config** and set the value of **SELINUX=disabled** instead of **SELINUX=permissive**
Save the file by pressing ctrl+o then press enter then press ctrl+x



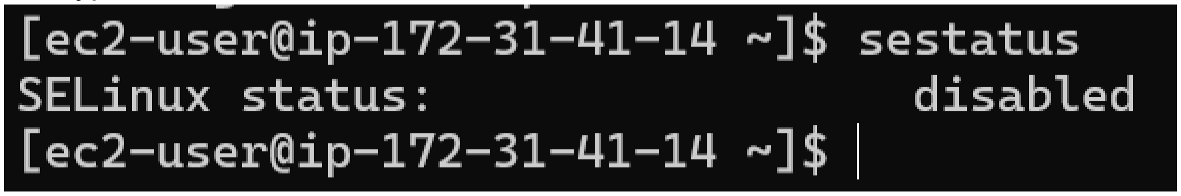
```
  GNU nano 5.8                              /etc/selinux/config                                      Modified
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
# See also:
# https://docs.fedoraproject.org/en-US/quick-docs/getting-started-with-selinux/#getting-started-with-selinux-selinux-st>
#
# NOTE: In earlier Fedora kernel builds, SELINUX=disabled would also
# fully disable SELinux during boot. If you need a system with SELinux
# fully disabled instead of SELinux running with no policy loaded, you
# need to pass selinux=0 to the kernel command line. You can use grubby
# to persistently set the bootloader to boot with selinux=0:
#
#    grubby --update-kernel ALL --args selinux=0
#
# To revert back to SELinux enabled:
#
#    grubby --update-kernel ALL --remove-args selinux
#
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted



^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

Then reboot the system using **sudo reboot**
After rebooting we need to make ssh connection with machine after it gets disconnected



```
[ec2-user@ip-172-31-41-14 bin]$ sudo nano /etc/selinux/config
[ec2-user@ip-172-31-41-14 bin]$ sudo reboot

Broadcast message from root@localhost on pts/1 (Wed 2024-09-18 16:15:38 UTC):

The system will reboot now!

[ec2-user@ip-172-31-41-14 bin]$ Connection to ec2-3-82-9-23.compute-1.amazonaws.com closed by remote host.
Connection to ec2-3-82-9-23.compute-1.amazonaws.com closed.

C:\Users\adity\Downloads>ssh -i "aditya.pem" ec2-user@ec2-3-82-9-23.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Sep 18 15:59:22 2024 from 202.179.85.199
[ec2-user@ip-172-31-41-14 ~]$
```

Now if we type command **sestatus,** then it show disabled

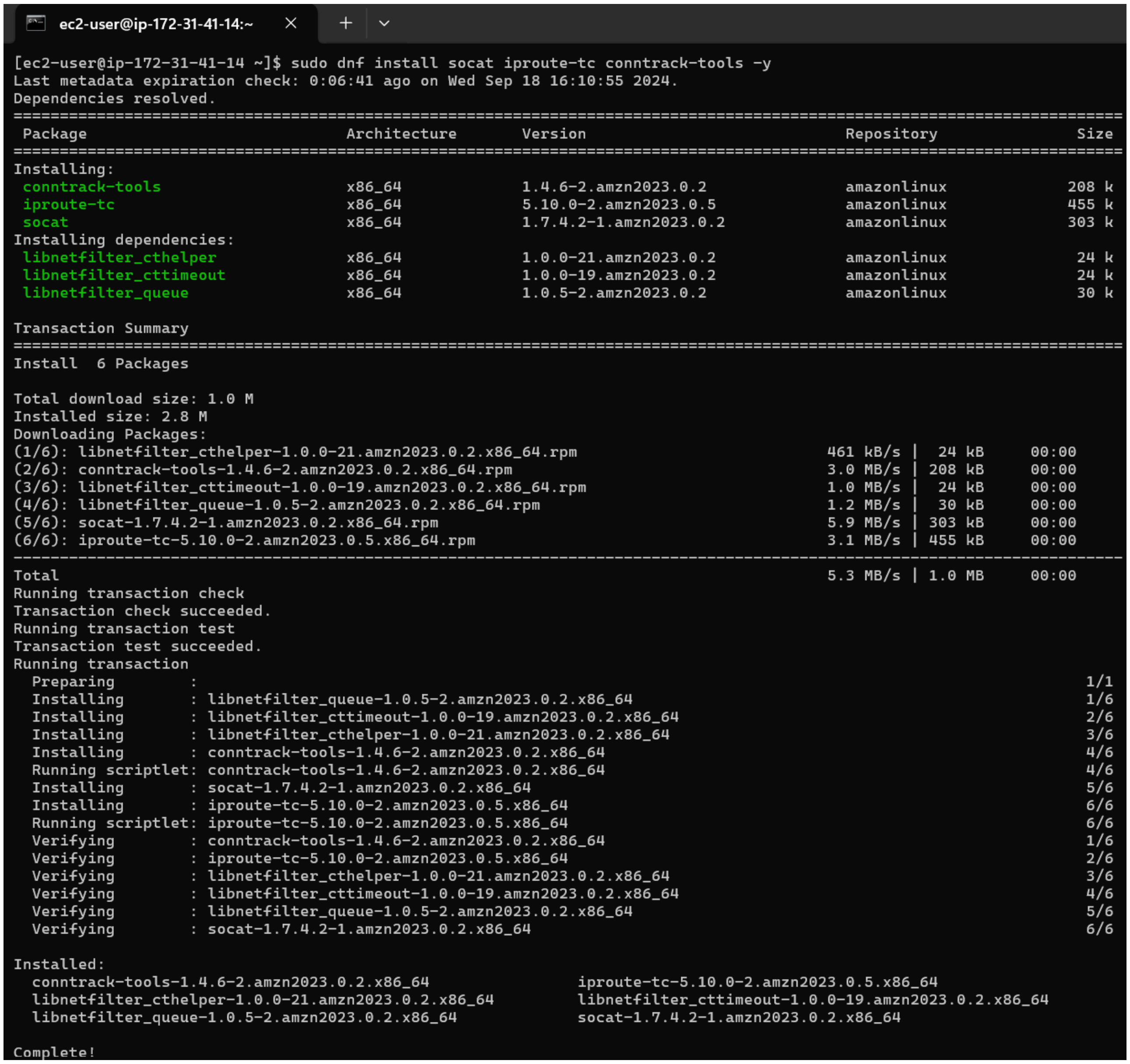


```
[ec2-user@ip-172-31-41-14 ~]$ sestatus
SELinux status:                 disabled
[ec2-user@ip-172-31-41-14 ~]$
```

5. Initialize the Kubecluster

Install packages socat and iproute-tc and conntrack to avoid prelight errors

- **sudo dnf install socat iproute-tc conntrack-tools -y**



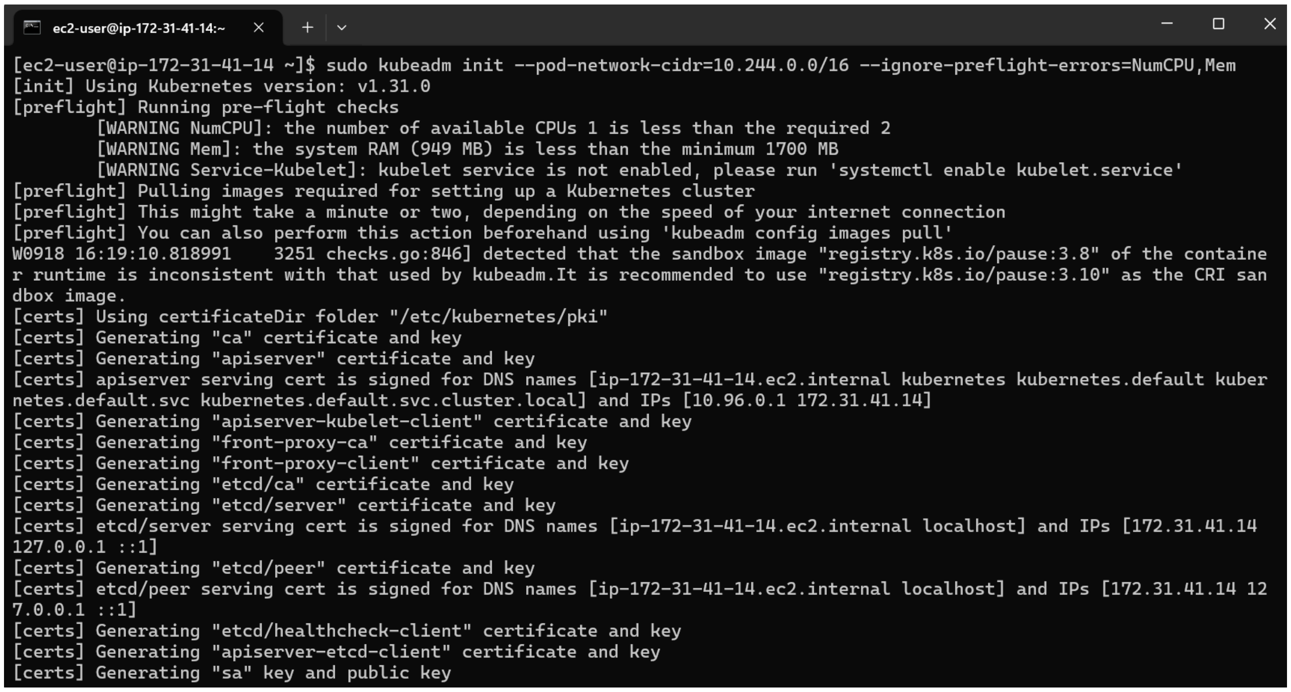
```
ec2-user@ip-172-31-41-14:~
[ec2-user@ip-172-31-41-14 ~]$ sudo dnf install socat iproute-tc conntrack-tools -y
Last metadata expiration check: 0:06:41 ago on Wed Sep 18 16:10:55 2024.
Dependencies resolved.
========================================================================================================================
 Package                            Architecture       Version                            Repository               Size
========================================================================================================================
Installing:
 conntrack-tools                    x86_64             1.4.6-2.amzn2023.0.2               amazonlinux             208 k
 iproute-tc                         x86_64             5.10.0-2.amzn2023.0.5              amazonlinux             455 k
 socat                              x86_64             1.7.4.2-1.amzn2023.0.2             amazonlinux             303 k
Installing dependencies:
 libnetfilter_cthelper              x86_64             1.0.0-21.amzn2023.0.2              amazonlinux              24 k
 libnetfilter_cttimeout             x86_64             1.0.0-19.amzn2023.0.2              amazonlinux              24 k
 libnetfilter_queue                 x86_64             1.0.5-2.amzn2023.0.2               amazonlinux              30 k

Transaction Summary
========================================================================================================================
Install  6 Packages

Total download size: 1.0 M
Installed size: 2.8 M
Downloading Packages:
(1/6): libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64.rpm                        461 kB/s |  24 kB     00:00
(2/6): conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64.rpm                               3.0 MB/s | 208 kB     00:00
(3/6): libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64.rpm                       1.0 MB/s |  24 kB     00:00
(4/6): libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64.rpm                            1.2 MB/s |  30 kB     00:00
(5/6): socat-1.7.4.2-1.amzn2023.0.2.x86_64.rpm                                       5.9 MB/s | 303 kB     00:00
(6/6): iproute-tc-5.10.0-2.amzn2023.0.5.x86_64.rpm                                   3.1 MB/s | 455 kB     00:00
------------------------------------------------------------------------------------------------------------------------
Total                                                                                5.3 MB/s | 1.0 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                1/1
  Installing       : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                                                 1/6
  Installing       : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64                                            2/6
  Installing       : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                                             3/6
  Installing       : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                    4/6
  Running scriptlet: conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                    4/6
  Installing       : socat-1.7.4.2-1.amzn2023.0.2.x86_64                                                            5/6
  Installing       : iproute-tc-5.10.0-2.amzn2023.0.5.x86_64                                                        6/6
  Running scriptlet: iproute-tc-5.10.0-2.amzn2023.0.5.x86_64                                                        6/6
  Verifying        : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                    1/6
  Verifying        : iproute-tc-5.10.0-2.amzn2023.0.5.x86_64                                                        2/6
  Verifying        : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                                             3/6
  Verifying        : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64                                            4/6
  Verifying        : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                                                 5/6
  Verifying        : socat-1.7.4.2-1.amzn2023.0.2.x86_64                                                            6/6

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                 iproute-tc-5.10.0-2.amzn2023.0.5.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64          libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64              socat-1.7.4.2-1.amzn2023.0.2.x86_64

Complete!
```

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

  --ignore-preflight-errors=NumCPU,Mem

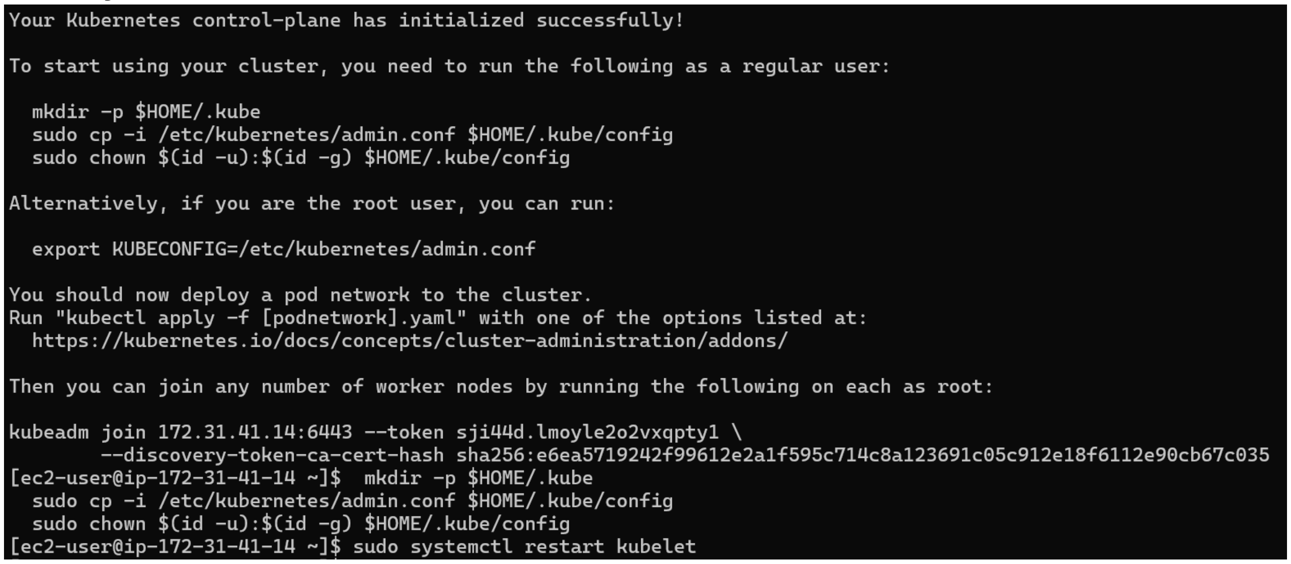
Copy the mkdir and chown commands from the top and execute them

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



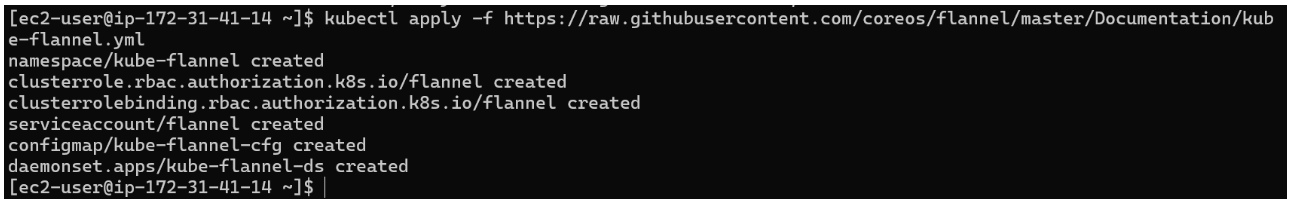
```
[ec2-user@ip-172-31-41-14 ~]$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU,Mem
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
[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'
W0918 16:19:10.818991    3251 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the containe
r runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI san
dbox 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-172-31-41-14.ec2.internal kubernetes kubernetes.default kuber
netes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.41.14]
[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-172-31-41-14.ec2.internal localhost] and IPs [172.31.41.14
127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-41-14.ec2.internal localhost] and IPs [172.31.41.14 12
7.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
```

- **sudo systemctl restart kubelet**



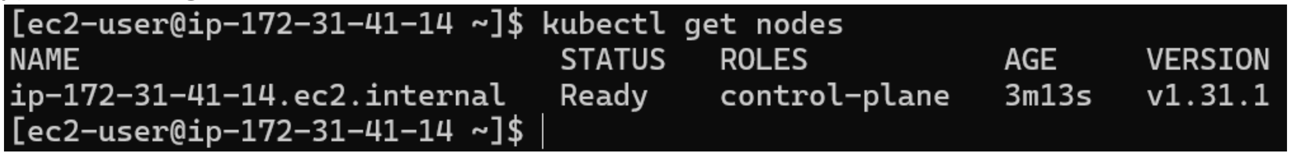
```
Your Kubernetes control-plane has initialized successfully!

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 172.31.41.14:6443 --token sji44d.lmoyle2o2vxqpty1 \
        --discovery-token-ca-cert-hash sha256:e6ea5719242f99612e2a1f595c714c8a123691c05c912e18f6112e90cb67c035
[ec2-user@ip-172-31-41-14 ~]$  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-41-14 ~]$ sudo systemctl restart kubelet
```

- Then, add a common networking plugin called flannel as mentioned in the code.
  kubectl apply -f
  https://raw.githubusercontent.com/coreos/flannel/master/Documentation/ku be-fla nnel.yml



```
[ec2-user@ip-172-31-41-14 ~]$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kub
e-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
[ec2-user@ip-172-31-41-14 ~]$
```

Now type **kubectl get nodes**

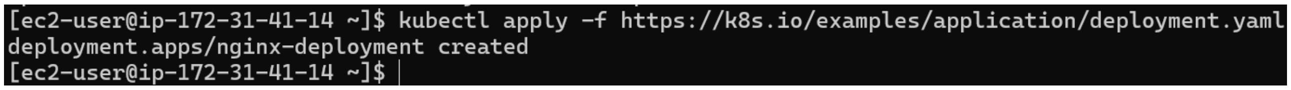


```
[ec2-user@ip-172-31-41-14 ~]$ kubectl get nodes
NAME                          STATUS   ROLES           AGE     VERSION
ip-172-31-41-14.ec2.internal  Ready    control-plane   3m13s   v1.31.1
[ec2-user@ip-172-31-41-14 ~]$
```

Note: If any time of get error of connection refused just restart the kubelet service (sudo systemctl restart kubelet)

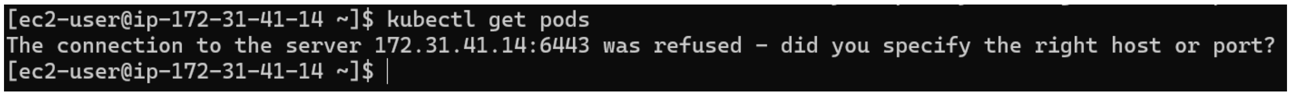
Now that the cluster is up and running, we can deploy our nginx server on this cluster.
Apply this deployment file using this command to create a deployment

- **kubectl apply -f https://k8s.io/examples/application/deployment.yaml**



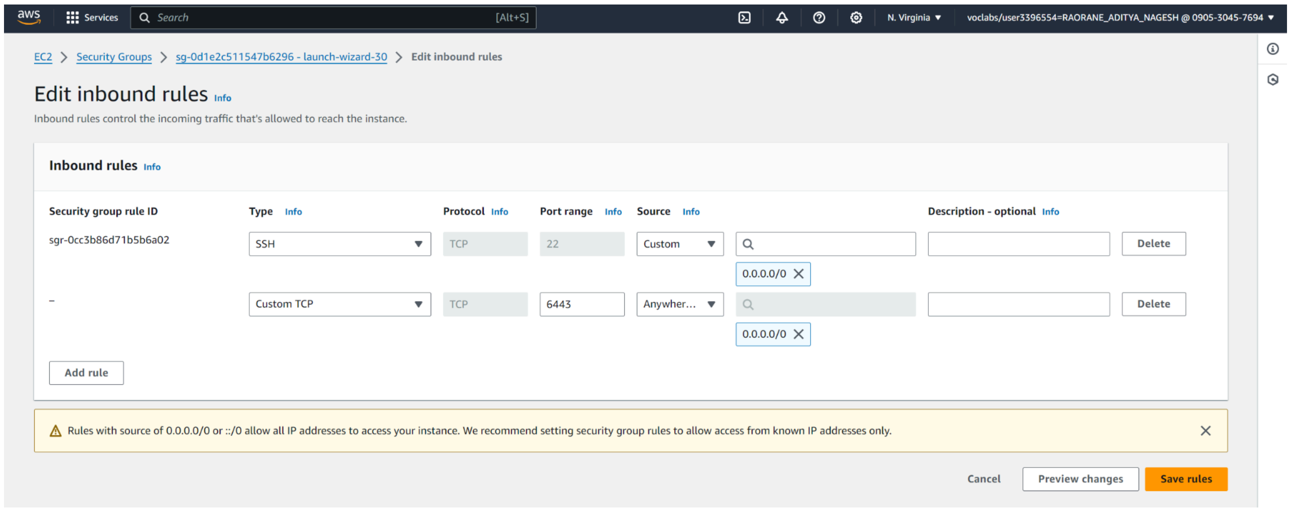
```
[ec2-user@ip-172-31-41-14 ~]$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
[ec2-user@ip-172-31-41-14 ~]$
```

Use ‘kubectl get pods’ to verify if the deployment was properly created and the pod is working correctly.



```
[ec2-user@ip-172-31-41-14 ~]$ kubectl get pods
The connection to the server 172.31.41.14:6443 was refused - did you specify the right host or port?
[ec2-user@ip-172-31-41-14 ~]$
```

Add an inbound rule under SSH security groups which will allow the traffic for a **custom TCP port** with port number **6443** with setting the source as **anywhere from IPv4.**



EC2 > Security Groups > sg-0d1e2c511547b6296 - launch-wizard-30 > Edit inbound rules

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-0cc3b86d71b5b6a02	SSH	TCP	22	Custom	0.0.0.0/0		Delete
–	Custom TCP	TCP	6443	Anywher...	0.0.0.0/0		Delete

Add rule

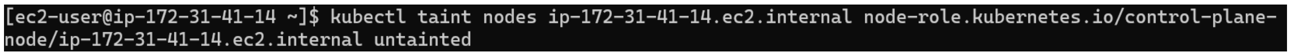
Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

As we can see our pods are in pending state
On checking logs to we came to know the pods are in tainted state (using command **kubectl describe pod nginx-deployment-d556bf558-7zthh**)

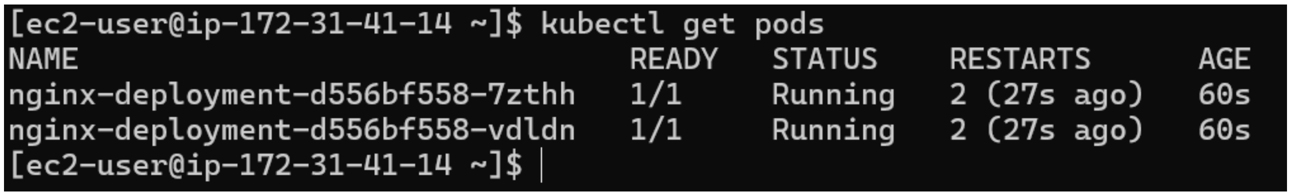
To make pods untainted
Type **kubectl get nodes** to see the name of the node.

Then type command **kubectl taint nodes <NODE_NAME> - -all**
In my case it is as follows:
kubectl taint nodes ip-172-31-41-14.ec2.internal node-role.kubernetes.io/control-plane-

After executing the above command, check again the status of pods if still pending then restart kubelet, wait for 1-2 minutes and check again.



```
[ec2-user@ip-172-31-41-14 ~]$ kubectl taint nodes ip-172-31-41-14.ec2.internal node-role.kubernetes.io/control-plane-
node/ip-172-31-41-14.ec2.internal untainted
```



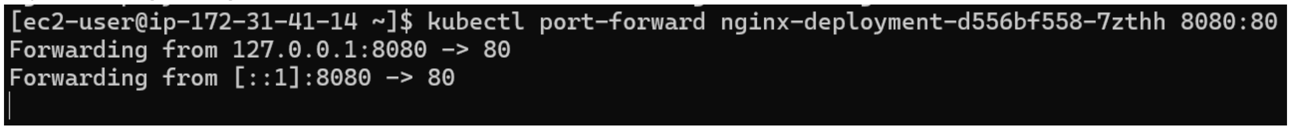
```
[ec2-user@ip-172-31-41-14 ~]$ kubectl get pods
NAME                               READY   STATUS    RESTARTS      AGE
nginx-deployment-d556bf558-7zthh   1/1     Running   2 (27s ago)   60s
nginx-deployment-d556bf558-vdldn   1/1     Running   2 (27s ago)   60s
[ec2-user@ip-172-31-41-14 ~]$
```

As we can see our pods are running

- Lastly, port forward the deployment to your localhost so that you can view it. **kubectl port-forward <POD_NAME> 8080:80**

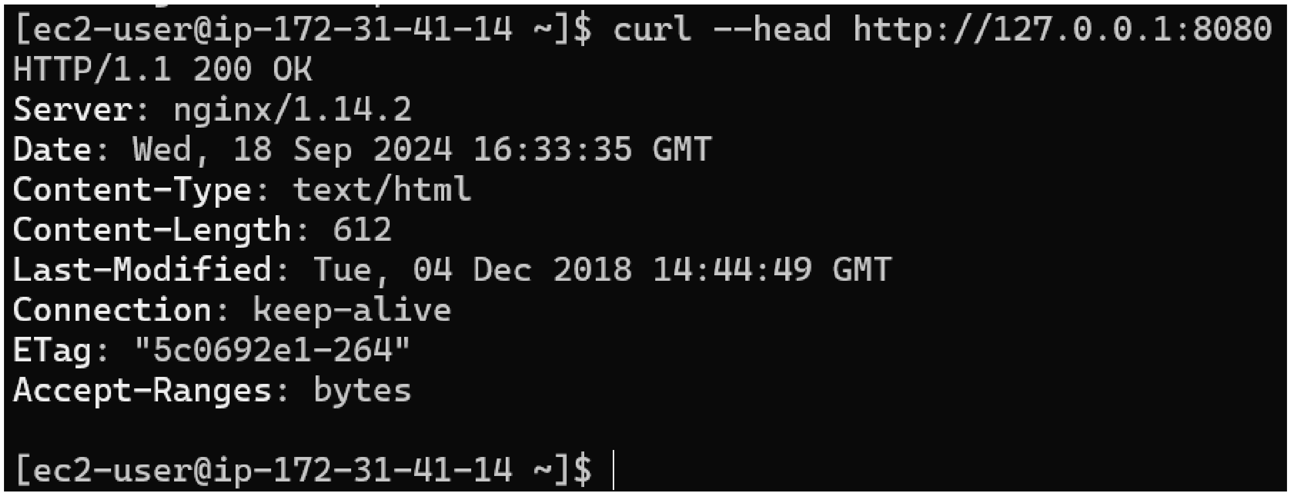
In my case : **kubectl port-forward nginx-deployment-d556bf558-7zthh 8080:80**

Note: if you are getting connection refused error then restart kubelet.



```
[ec2-user@ip-172-31-41-14 ~]$ kubectl port-forward nginx-deployment-d556bf558-7zthh 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

As port forwarding is active so we cannot type other commands.
Open new terminal window and make ssh connection to same machine
And type command **curl --head** http://127.0.0.1:8080



```
[ec2-user@ip-172-31-41-14 ~]$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Wed, 18 Sep 2024 16:33:35 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes

[ec2-user@ip-172-31-41-14 ~]$
```

Response status 200 (OK) indicates that our nginx server is running successfully on kubernetes

Conclusion: We began by installing and configuring Docker and Kubernetes, encountering some initial issues with the Kubernetes API server, which were resolved by restarting the `kubelet` service. The pods didn't start at first due to taints on the nodes, which we removed to allow normal pod scheduling. After resolving these errors, we successfully deployed Nginx server pods and configured them to be accessible via port forwarding. Additionally, we configured the SSH security group by adding an inbound rule to permit traffic on TCP port 6443 from any IPv4 address.