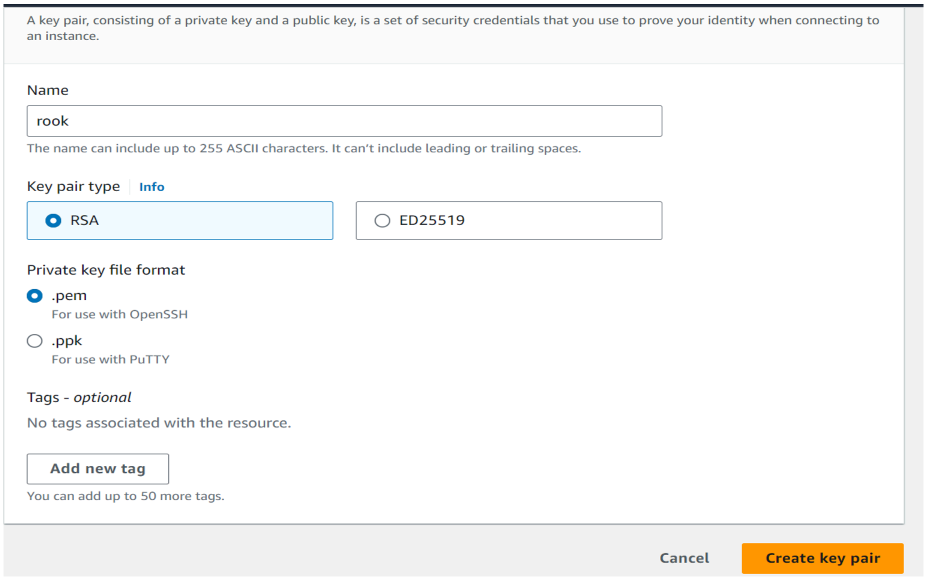
Aim:
To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Steps:

1. Create a key pair.



A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

rook

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

Key pair type Info

● RSA ○ ED25519

Private key file format

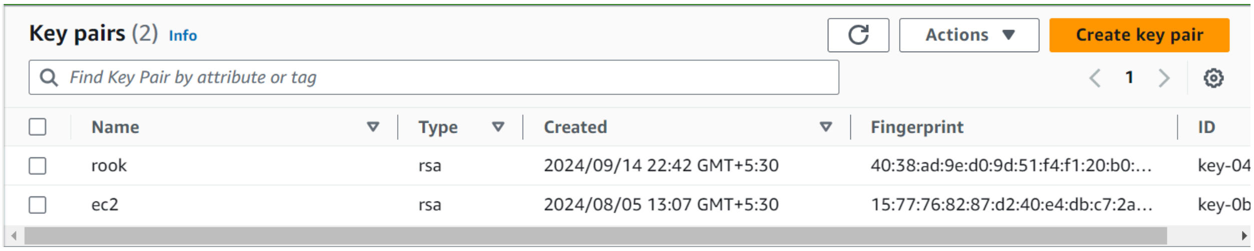
● .pem
For use with OpenSSH

○ .ppk
For use with PuTTY

Tags - *optional*

No tags associated with the resource.

You can add up to 50 more tags.

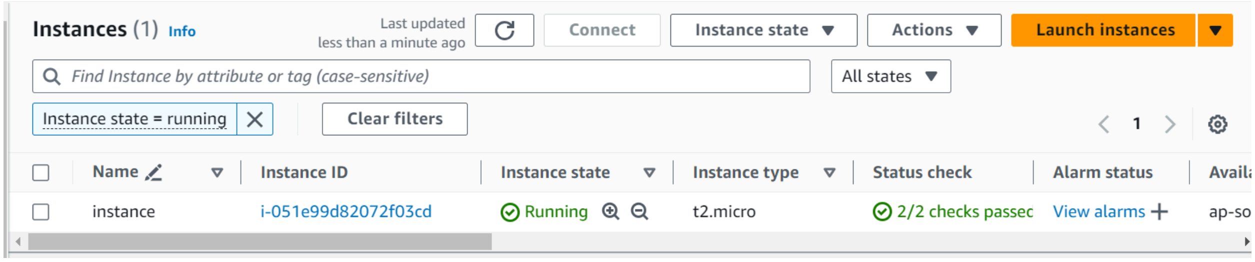


Key pairs (2) Info

□	Name	Type	Created	Fingerprint	ID
□	rook	rsa	2024/09/14 22:42 GMT+5:30	40:38:ad:9e:d0:9d:51:f4:f1:20:b0:...	key-04
□	ec2	rsa	2024/08/05 13:07 GMT+5:30	15:77:76:82:87:d2:40:e4:db:c7:2a...	key-0b

The .pem file will be downloaded on your machine and will be required in the further steps.

2. Now we will create an EC2 Ubuntu instance. Select the key pair which you just created while creating this instance.

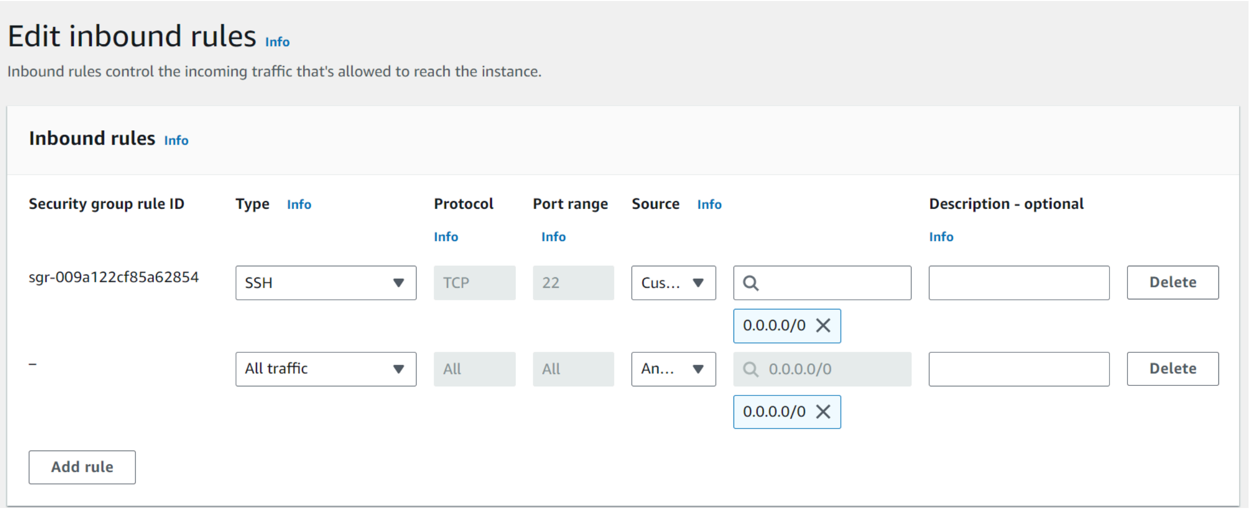


Instances (1) Info

Last updated
less than a minute ago

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Avail
□	instance	i-051e99d82072f03cd	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	ap-so

3. Now edit the inbound rules to allow ssh.



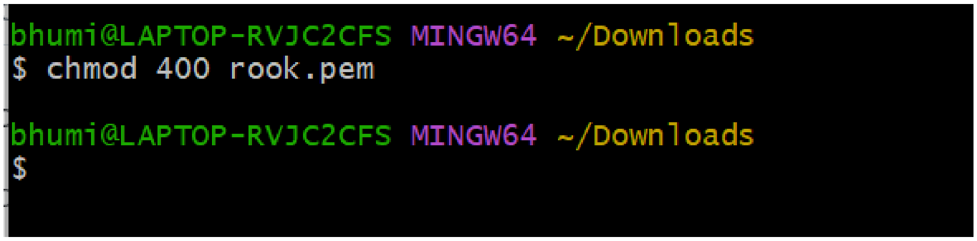
Edit inbound rules Info

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

Inbound rules Info

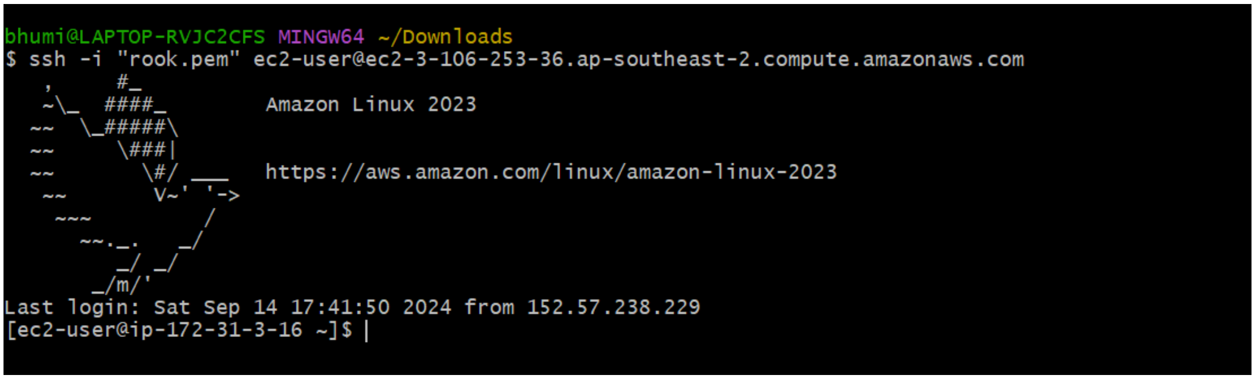
Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-009a122cf85a62854	SSH ▼	TCP	22	Cus... ▼	🔍 0.0.0.0/0 ✕		Delete
–	All traffic ▼	All	All	An... ▼	🔍 0.0.0.0/0 0.0.0.0/0 ✕		Delete

4. Open git bash and go to the directory where pem file is located and use chmod to provide permissions.



```
bhumi@LAPTOP-RVJC2CFS MINGW64 ~/Downloads
$ chmod 400 rook.pem

bhumi@LAPTOP-RVJC2CFS MINGW64 ~/Downloads
$
```

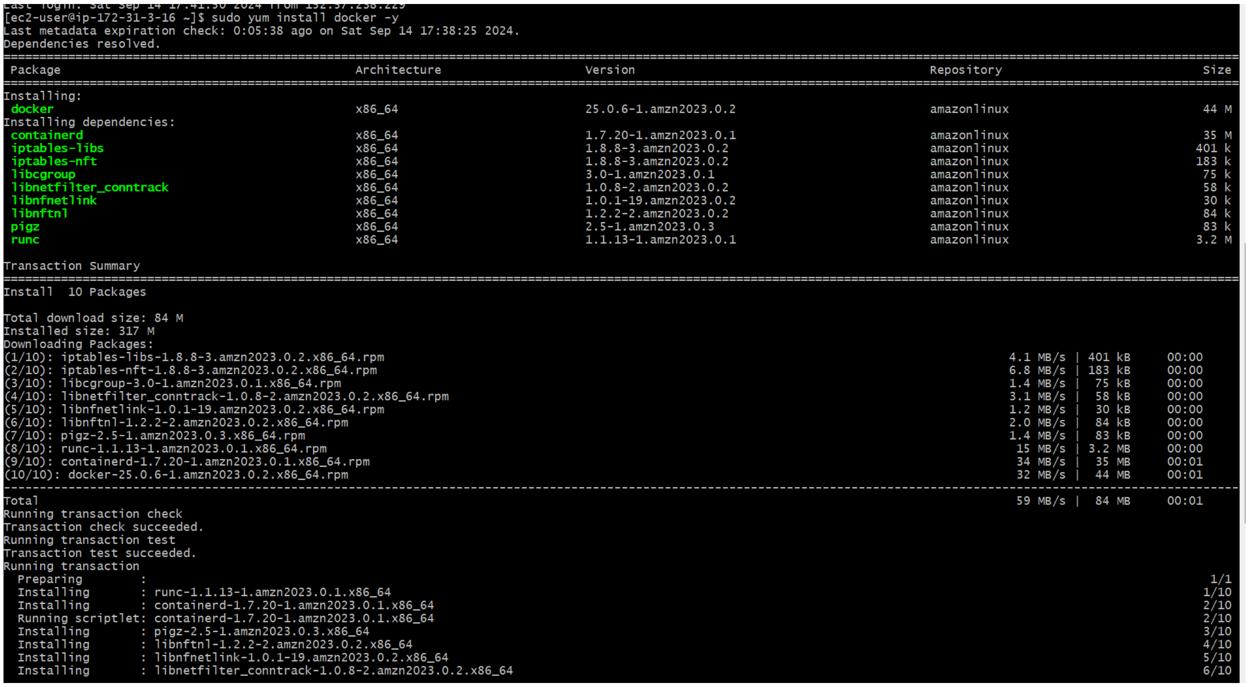
5. Now use this command on the terminal: ssh -i <keyname>.pem ubuntu@ and replace
   • Keyname with the name of your key pair, in our case test1.
   • As we are using amazon Linux instead of ubuntu we will have ec2-user
   • Replace public ip address with its value. Go to your instance and scroll down and you will find the public ip address there.



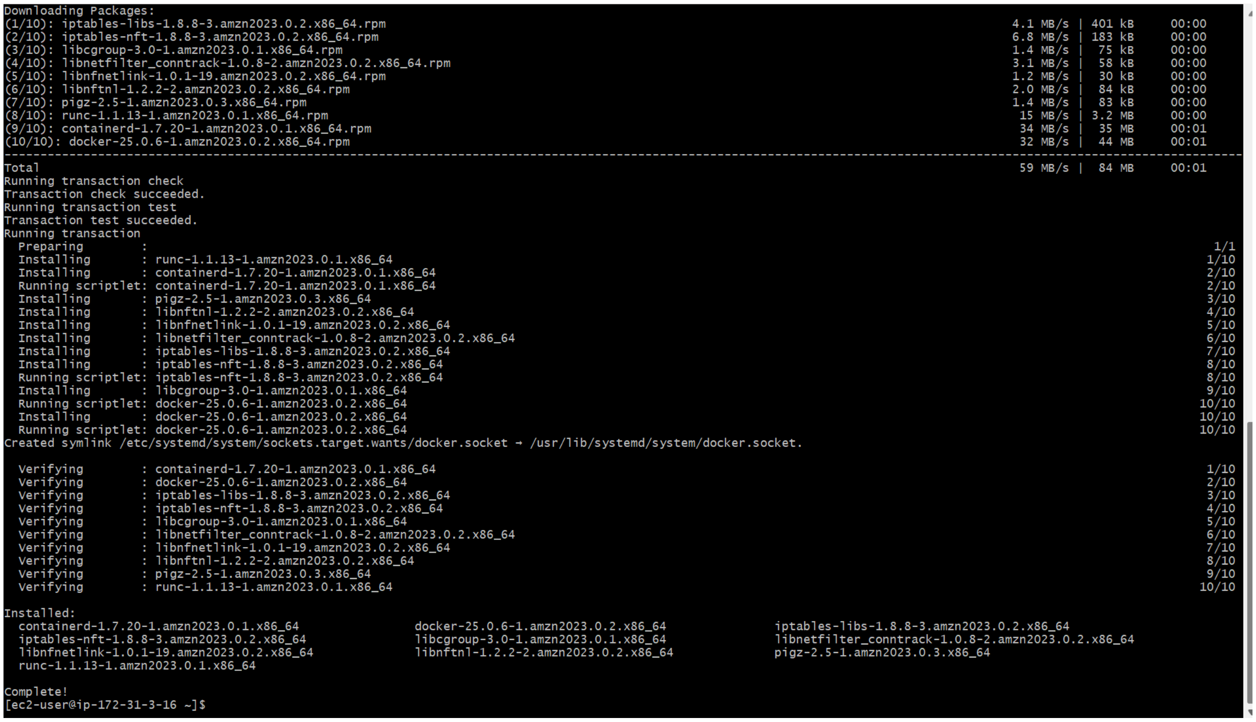
```
bhumi@LAPTOP-RVJC2CFS MINGW64 ~/Downloads
$ ssh -i "rook.pem" ec2-user@ec2-3-106-253-36.ap-southeast-2.compute.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Sep 14 17:41:50 2024 from 152.57.238.229
[ec2-user@ip-172-31-3-16 ~]$ |
```

6. Docker installation:

We will be installing docker by using “sudo yum install docker -y”



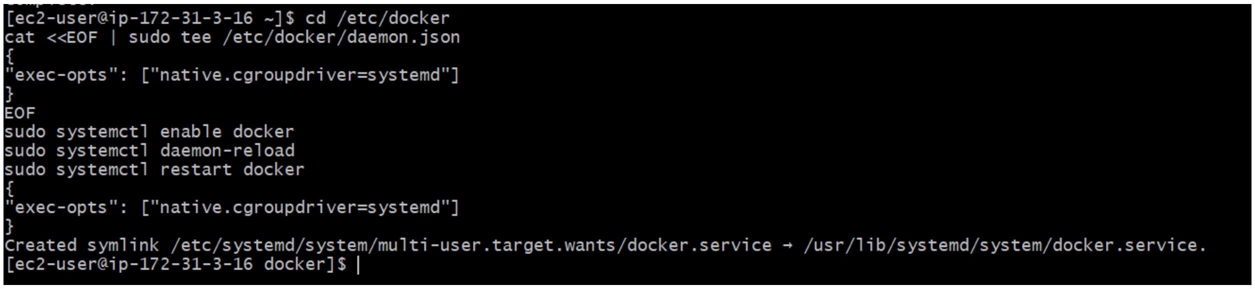
```
Last login: Sat Sep 14 17:41:50 2024 from 152.57.238.229
[ec2-user@ip-172-31-3-16 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:05:38 ago on Sat Sep 14 17:38:25 2024.
Dependencies resolved.
==================================================================================================================================
 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
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm                                              4.1 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm                                               6.8 MB/s | 183 kB     00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm                                                    1.4 MB/s |  75 kB     00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm                                     3.1 MB/s |  58 kB     00:00
(5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64.rpm                                              1.2 MB/s |  30 kB     00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm                                                   2.0 MB/s |  84 kB     00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm                                                         1.4 MB/s |  83 kB     00:00
(8/10): runc-1.1.13-1.amzn2023.0.1.x86_64.rpm                                                       15 MB/s | 3.2 MB     00:00
(9/10): containerd-1.7.20-1.amzn2023.0.1.x86_64.rpm                                                 34 MB/s |  35 MB     00:01
(10/10): docker-25.0.6-1.amzn2023.0.2.x86_64.rpm                                                    32 MB/s |  44 MB     00:01
----------------------------------------------------------------------------------------------------------------------------------
Total                                                                                               59 MB/s |  84 MB     00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                          1/1
  Installing       : runc-1.1.13-1.amzn2023.0.1.x86_64                                                                       1/10
  Installing       : containerd-1.7.20-1.amzn2023.0.1.x86_64                                                                 2/10
  Running scriptlet: containerd-1.7.20-1.amzn2023.0.1.x86_64                                                                 2/10
  Installing       : pigz-2.5-1.amzn2023.0.3.x86_64                                                                          3/10
  Installing       : libnftnl-1.2.2-2.amzn2023.0.2.x86_64                                                                    4/10
  Installing       : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64                                                               5/10
  Installing       : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64                                                      6/10
```



```
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm                                              4.1 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm                                               6.8 MB/s | 183 kB     00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm                                                    1.4 MB/s |  75 kB     00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm                                     3.1 MB/s |  58 kB     00:00
(5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64.rpm                                              1.2 MB/s |  30 kB     00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm                                                   2.0 MB/s |  84 kB     00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm                                                         1.4 MB/s |  83 kB     00:00
(8/10): runc-1.1.13-1.amzn2023.0.1.x86_64.rpm                                                       15 MB/s | 3.2 MB     00:00
(9/10): containerd-1.7.20-1.amzn2023.0.1.x86_64.rpm                                                 34 MB/s |  35 MB     00:01
(10/10): docker-25.0.6-1.amzn2023.0.2.x86_64.rpm                                                    32 MB/s |  44 MB     00:01
----------------------------------------------------------------------------------------------------------------------------------
Total                                                                                               59 MB/s |  84 MB     00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                          1/1
  Installing       : runc-1.1.13-1.amzn2023.0.1.x86_64                                                                       1/10
  Installing       : containerd-1.7.20-1.amzn2023.0.1.x86_64                                                                 2/10
  Running scriptlet: containerd-1.7.20-1.amzn2023.0.1.x86_64                                                                 2/10
  Installing       : pigz-2.5-1.amzn2023.0.3.x86_64                                                                          3/10
  Installing       : libnftnl-1.2.2-2.amzn2023.0.2.x86_64                                                                    4/10
  Installing       : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64                                                               5/10
  Installing       : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64                                                      6/10
  Installing       : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64                                                               7/10
  Installing       : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64                                                                8/10
  Running scriptlet: iptables-nft-1.8.8-3.amzn2023.0.2.x86_64                                                                8/10
  Installing       : libcgroup-3.0-1.amzn2023.0.1.x86_64                                                                     9/10
  Running scriptlet: docker-25.0.6-1.amzn2023.0.2.x86_64                                                                    10/10
  Installing       : docker-25.0.6-1.amzn2023.0.2.x86_64                                                                    10/10
  Running scriptlet: docker-25.0.6-1.amzn2023.0.2.x86_64                                                                    10/10
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.

  Verifying        : containerd-1.7.20-1.amzn2023.0.1.x86_64                                                                 1/10
  Verifying        : docker-25.0.6-1.amzn2023.0.2.x86_64                                                                     2/10
  Verifying        : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64                                                               3/10
  Verifying        : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64                                                                4/10
  Verifying        : libcgroup-3.0-1.amzn2023.0.1.x86_64                                                                     5/10
  Verifying        : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64                                                      6/10
  Verifying        : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64                                                               7/10
  Verifying        : libnftnl-1.2.2-2.amzn2023.0.2.x86_64                                                                    8/10
  Verifying        : pigz-2.5-1.amzn2023.0.3.x86_64                                                                          9/10
  Verifying        : runc-1.1.13-1.amzn2023.0.1.x86_64                                                                      10/10

Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64             docker-25.0.6-1.amzn2023.0.2.x86_64             iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64            libcgroup-3.0-1.amzn2023.0.1.x86_64             libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64           libnftnl-1.2.2-2.amzn2023.0.2.x86_64            pigz-2.5-1.amzn2023.0.3.x86_64
  runc-1.1.13-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-3-16 ~]$
```

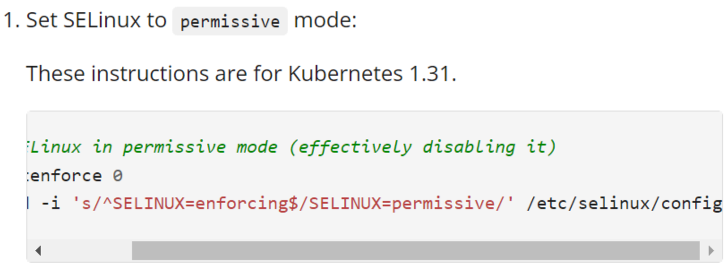
7. Then to configure cgroup in a daemon json file we will run
   cd /etc/docker
   cat <<EOF | sudo tee /etc/docker/daemon.json
   {
   "exec-opts": ["native.cgroupdriver=systemd"]

```
}
EOF
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```



```
[ec2-user@ip-172-31-3-16 ~]$ cd /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-3-16 docker]$ |
```

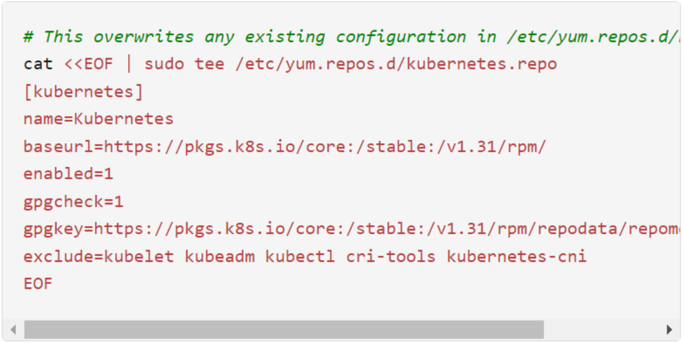
8. Kubernetes installation:
   Search kubeadm installation on your browser and scroll down and select red hat-based distributions.



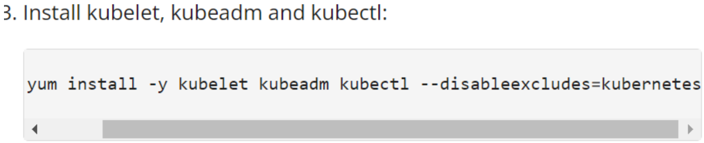
1. Set SELinux to `permissive` mode:

These instructions are for Kubernetes 1.31.

```
:Linux in permissive mode (effectively disabling it)
:enforce 0
| -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

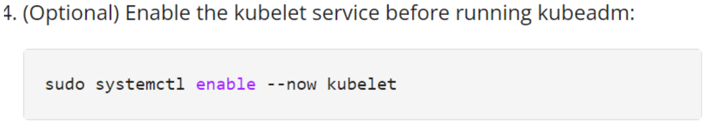


```
# This overwrites any existing configuration in /etc/yum.repos.d/
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repom
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```



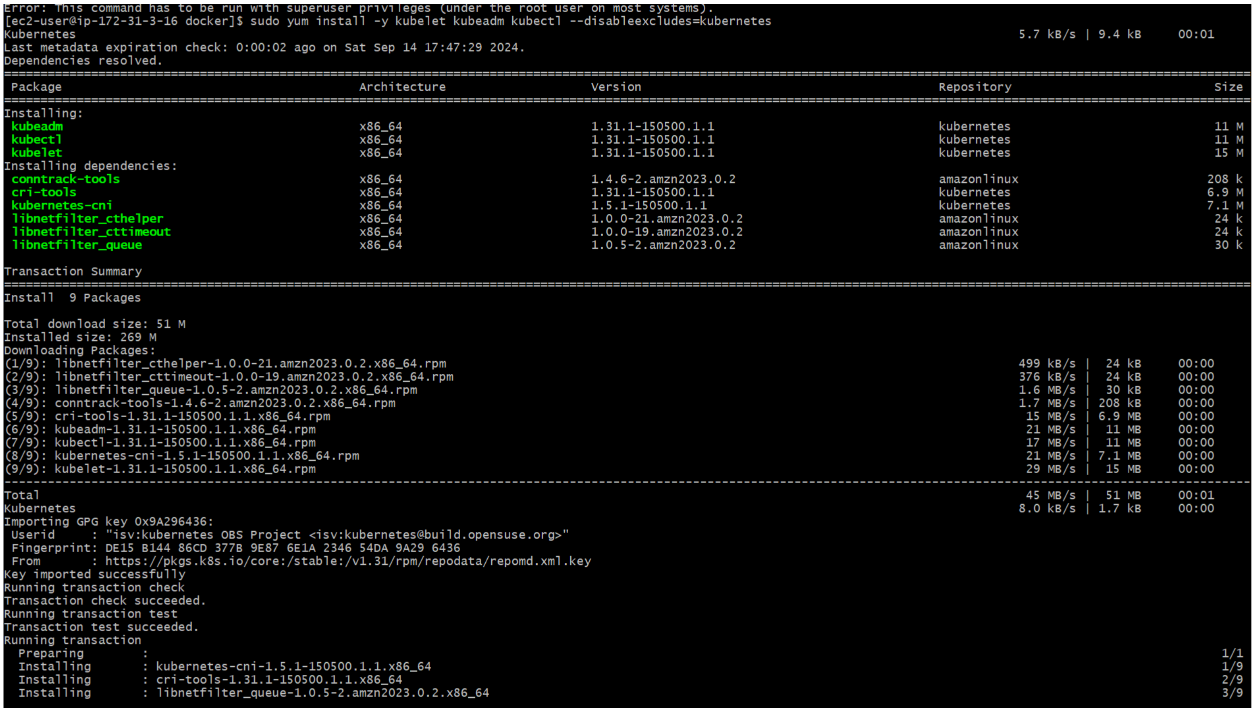
3. Install kubelet, kubeadm and kubectl:

```
yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
```

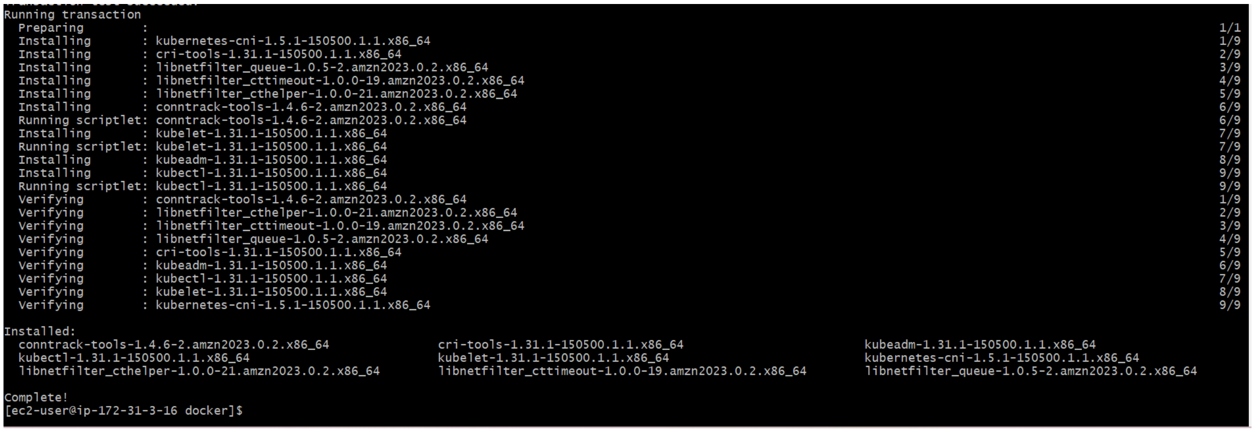


4. (Optional) Enable the kubelet service before running kubeadm:

```
sudo systemctl enable --now kubelet
```

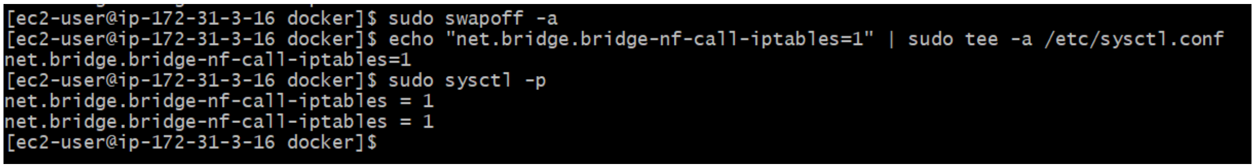


```
Error: This command has to be run with superuser privileges (under the root user on most systems).
[ec2-user@ip-172-31-3-16 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Kubernetes                                                                          5.7 kB/s | 9.4 kB     00:01
Last metadata expiration check: 0:00:02 ago on Sat Sep 14 17:47:29 2024.
Dependencies resolved.
==================================================================================================================
 Package                       Architecture       Version                         Repository                 Size
==================================================================================================================
Installing:
 kubeadm                       x86_64             1.31.1-150500.1.1               kubernetes                 11 M
 kubectl                       x86_64             1.31.1-150500.1.1               kubernetes                 11 M
 kubelet                       x86_64             1.31.1-150500.1.1               kubernetes                 15 M
Installing dependencies:
 conntrack-tools               x86_64             1.4.6-2.amzn2023.0.2            amazonlinux               208 k
 cri-tools                     x86_64             1.31.1-150500.1.1               kubernetes                6.9 M
 kubernetes-cni                x86_64             1.5.1-150500.1.1                kubernetes                7.1 M
 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  9 Packages

Total download size: 51 M
Installed size: 269 M
Downloading Packages:
(1/9): libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64.rpm                       499 kB/s |  24 kB     00:00
(2/9): libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64.rpm                      376 kB/s |  24 kB     00:00
(3/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64.rpm                           1.6 MB/s |  30 kB     00:00
(4/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64.rpm                              1.7 MB/s | 208 kB     00:00
(5/9): cri-tools-1.31.1-150500.1.1.x86_64.rpm                                        15 MB/s | 6.9 MB     00:00
(6/9): kubeadm-1.31.1-150500.1.1.x86_64.rpm                                          21 MB/s |  11 MB     00:00
(7/9): kubectl-1.31.1-150500.1.1.x86_64.rpm                                          17 MB/s |  11 MB     00:00
(8/9): kubernetes-cni-1.5.1-150500.1.1.x86_64.rpm                                    21 MB/s | 7.1 MB     00:00
(9/9): kubelet-1.31.1-150500.1.1.x86_64.rpm                                          29 MB/s |  15 MB     00:00
------------------------------------------------------------------------------------------------------------------
Total                                                                                45 MB/s |  51 MB     00:01
Kubernetes                                                                          8.0 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       : kubernetes-cni-1.5.1-150500.1.1.x86_64                                                   1/9
  Installing       : cri-tools-1.31.1-150500.1.1.x86_64                                                       2/9
  Installing       : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                                           3/9
```



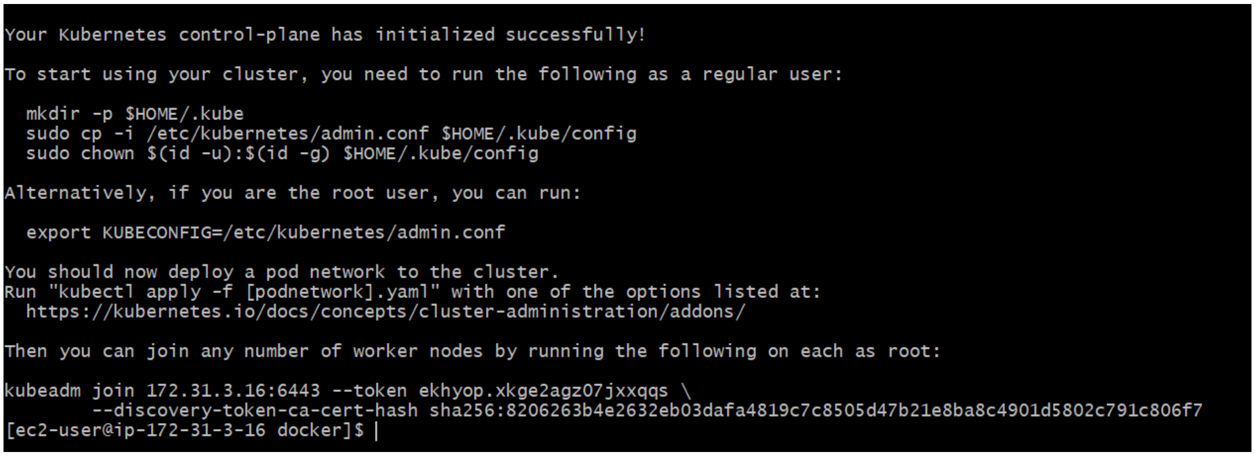
```
Running transaction
  Preparing        :                                                                                          1/1
  Installing       : kubernetes-cni-1.5.1-150500.1.1.x86_64                                                   1/9
  Installing       : cri-tools-1.31.1-150500.1.1.x86_64                                                       2/9
  Installing       : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                                           3/9
  Installing       : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64                                      4/9
  Installing       : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                                       5/9
  Installing       : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                              6/9
  Running scriptlet: conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                              6/9
  Installing       : kubelet-1.31.1-150500.1.1.x86_64                                                         7/9
  Running scriptlet: kubelet-1.31.1-150500.1.1.x86_64                                                         7/9
  Installing       : kubeadm-1.31.1-150500.1.1.x86_64                                                         8/9
  Installing       : kubectl-1.31.1-150500.1.1.x86_64                                                         9/9
  Running scriptlet: kubectl-1.31.1-150500.1.1.x86_64                                                         9/9
  Verifying        : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                              1/9
  Verifying        : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                                       2/9
  Verifying        : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64                                      3/9
  Verifying        : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                                           4/9
  Verifying        : cri-tools-1.31.1-150500.1.1.x86_64                                                       5/9
  Verifying        : kubeadm-1.31.1-150500.1.1.x86_64                                                         6/9
  Verifying        : kubectl-1.31.1-150500.1.1.x86_64                                                         7/9
  Verifying        : kubelet-1.31.1-150500.1.1.x86_64                                                         8/9
  Verifying        : kubernetes-cni-1.5.1-150500.1.1.x86_64                                                   9/9

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64          cri-tools-1.31.1-150500.1.1.x86_64                  kubeadm-1.31.1-150500.1.1.x86_64
  kubectl-1.31.1-150500.1.1.x86_64                     kubelet-1.31.1-150500.1.1.x86_64                    kubernetes-cni-1.5.1-150500.1.1.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

Complete!
[ec2-user@ip-172-31-3-16 docker]$
```

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



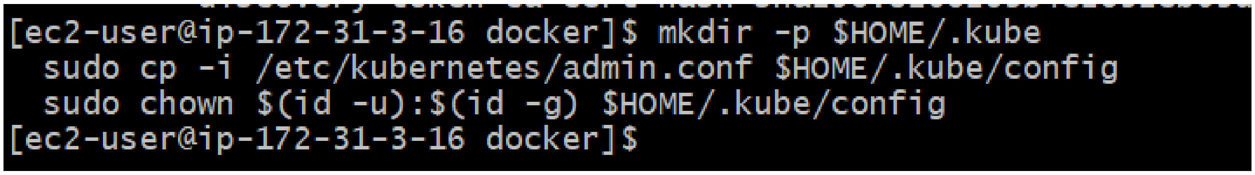
```
[ec2-user@ip-172-31-3-16 docker]$ sudo swapoff -a
[ec2-user@ip-172-31-3-16 docker]$ echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[ec2-user@ip-172-31-3-16 docker]$ sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-3-16 docker]$
```

10. Initializing kubecluster:
    sudo kubeadm init --pod-network-cidr=10.244.0.0/16



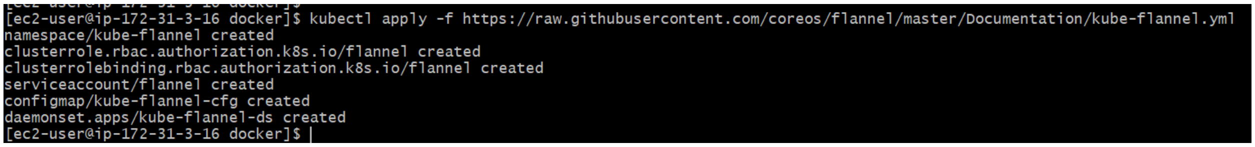
```
Your Kubernetes control-plane has initialized successfully!

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 172.31.3.16:6443 --token ekhyop.xkge2agz07jxxqqs \
        --discovery-token-ca-cert-hash sha256:8206263b4e2632eb03dafa4819c7c8505d47b21e8ba8c4901d5802c791c806f7
[ec2-user@ip-172-31-3-16 docker]$ |
```

11. The mkdir command that is generated after initialization has to be copy pasted in the terminal.



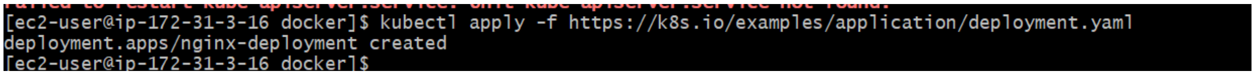
```
[ec2-user@ip-172-31-3-16 docker]$ mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-3-16 docker]$
```

12. Then, add a common networking plugin called flannel:
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/
kube-flannel.yml



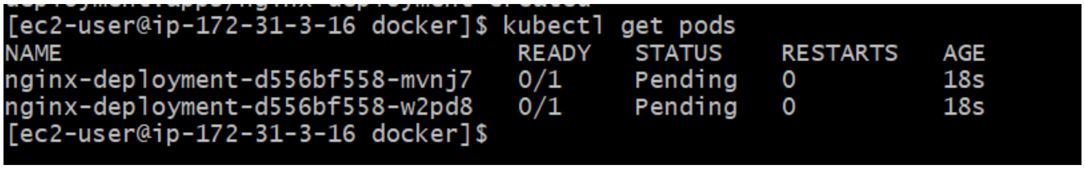
```
[ec2-user@ip-172-31-3-16 docker]$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
[ec2-user@ip-172-31-3-16 docker]$ |
```

13. 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



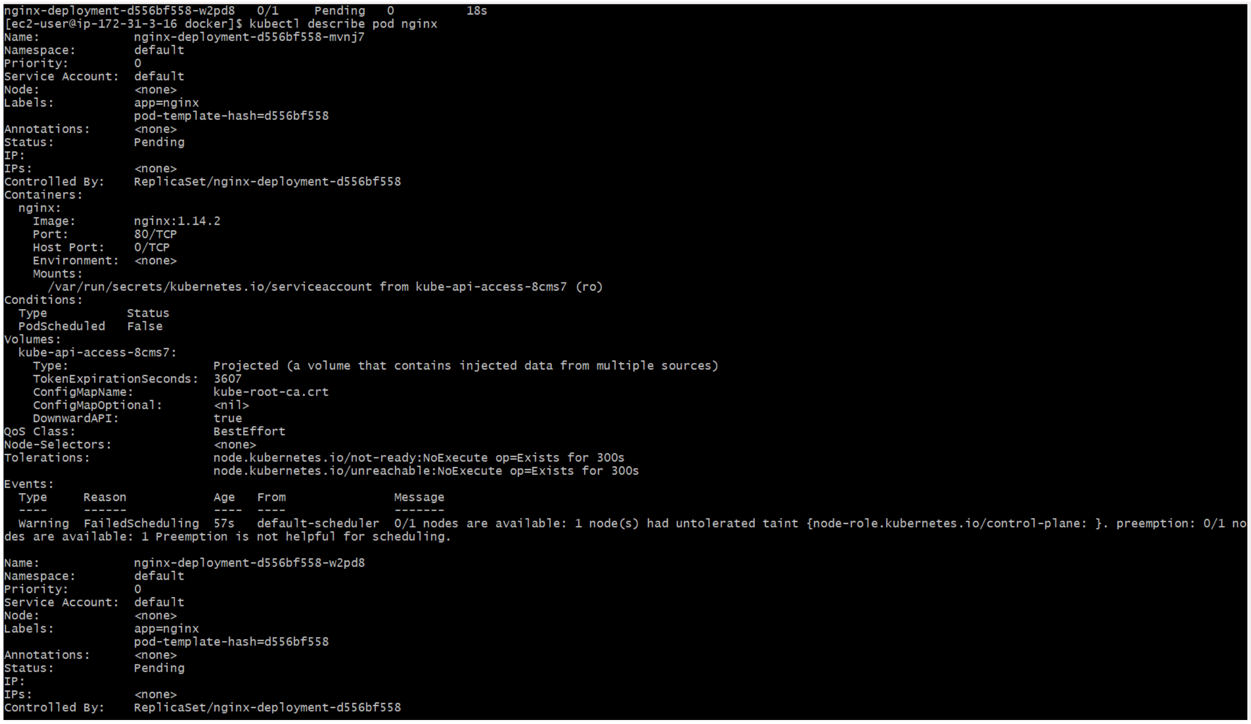
```
[ec2-user@ip-172-31-3-16 docker]$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
[ec2-user@ip-172-31-3-16 docker]$
```

14. Use kubectl get pods to check if the pod is working correctly.

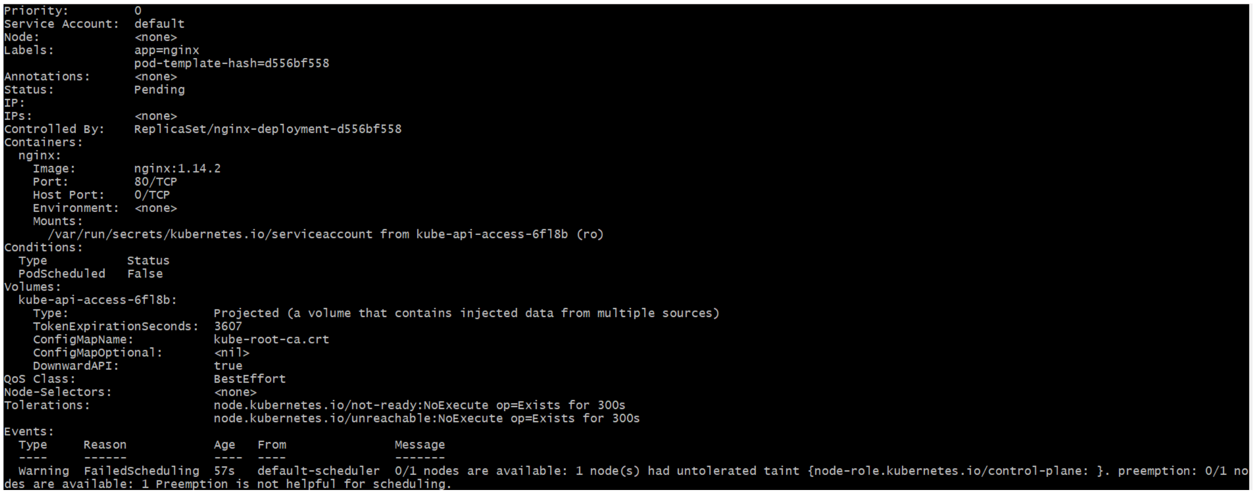


```
[ec2-user@ip-172-31-3-16 docker]$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-mvnj7   0/1     Pending   0          18s
nginx-deployment-d556bf558-w2pd8   0/1     Pending   0          18s
[ec2-user@ip-172-31-3-16 docker]$
```

15. To change status from pending to running use the following command: kubectl describe pod nginx.

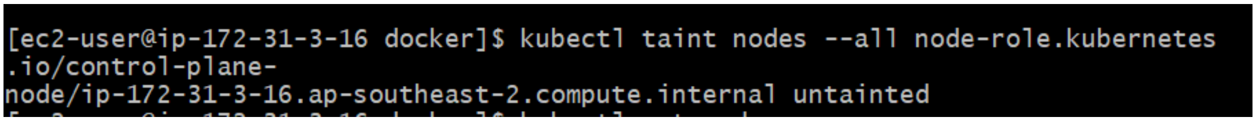


```
nginx-deployment-d556bf558-w2pd8   0/1     Pending   0          18s
[ec2-user@ip-172-31-3-16 docker]$ kubectl describe pod nginx
Name:             nginx-deployment-d556bf558-mvnj7
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           app=nginx
                  pod-template-hash=d556bf558
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Controlled By:    ReplicaSet/nginx-deployment-d556bf558
Containers:
  nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-8cms7 (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-8cms7:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  57s   default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption: 0/1 no
des are available: 1 Preemption is not helpful for scheduling.

Name:             nginx-deployment-d556bf558-w2pd8
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           app=nginx
                  pod-template-hash=d556bf558
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Controlled By:    ReplicaSet/nginx-deployment-d556bf558
```



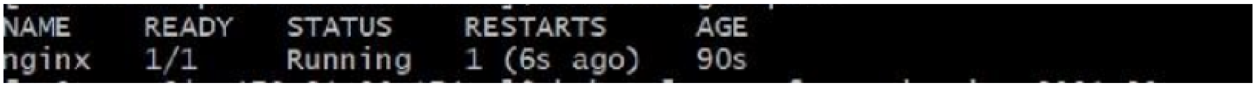
```
Priority:         0
Service Account:  default
Node:             <none>
Labels:           app=nginx
                  pod-template-hash=d556bf558
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Controlled By:    ReplicaSet/nginx-deployment-d556bf558
Containers:
  nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-6fl8b (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-6fl8b:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  57s   default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption: 0/1 no
des are available: 1 Preemption is not helpful for scheduling.
```

Use the below command to remove taints.



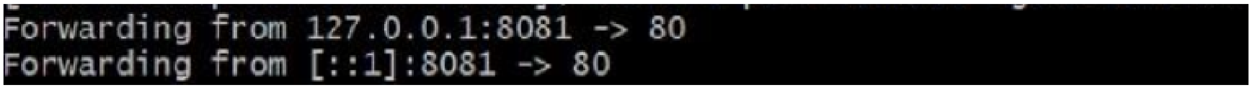
```
[ec2-user@ip-172-31-3-16 docker]$ kubectl taint nodes --all node-role.kubernetes
.io/control-plane-
node/ip-172-31-3-16.ap-southeast-2.compute.internal untainted
```

16. Check the pod status.



```
NAME    READY   STATUS    RESTARTS     AGE
nginx   1/1     Running   1 (6s ago)   90s
```

17. port forward the deployment to your localhost so that you can view it



```
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

18. Verify your deployment Open up a new terminal and ssh to your EC2 instance. Then, use this curl command to check if the Nginx server is running.
    curl --head http://127.0.0.1:8080

Conclusion: In this experiment, we launched an EC2 instance and configured SSH access by updating the inbound rules. Next, we installed Docker and Kubernetes, and adjusted network settings to enable bridging. After completing the setup, we installed the Flannel networking plugin to ensure proper communication within the cluster. Once the cluster was up and running, we successfully deployed an NGINX server and verified its deployment.