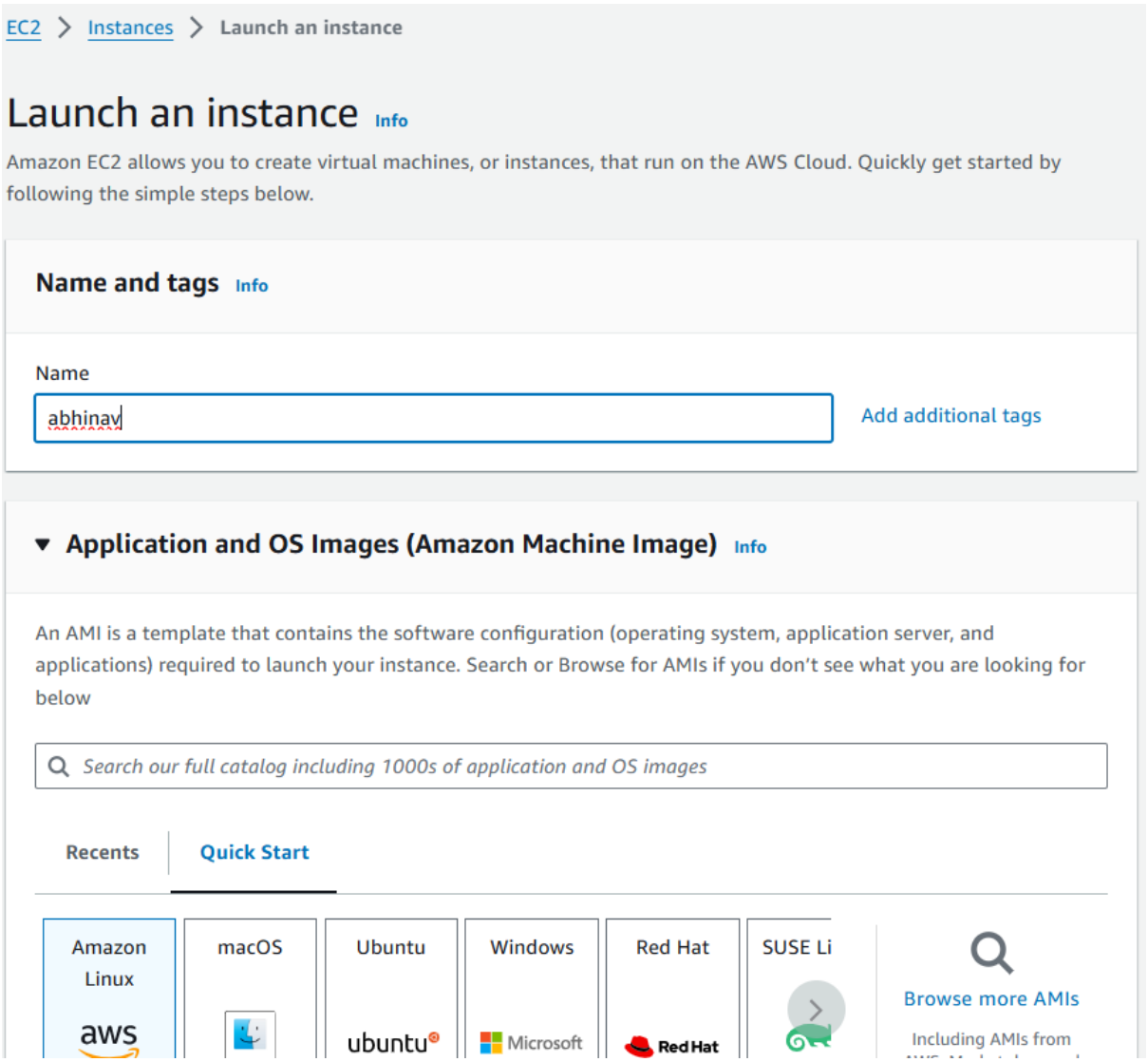
Name: Abhinav Swaminathan **D15C** **Roll No: 01**

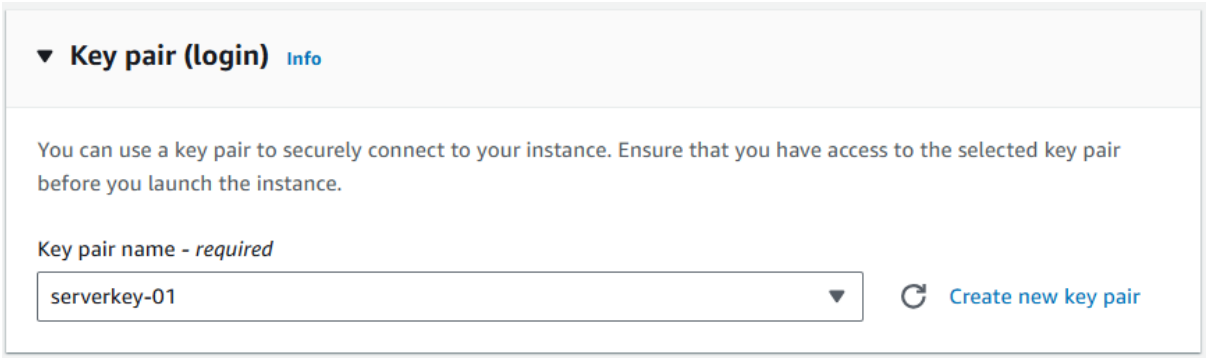
Experiment No. 4

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

Steps:

1. **Create an EC2 Linux Instances on AWS.**





Network settings Info

Edit

Network Info

vpc-0404d393731afabc3

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

(•) Create security group

( ) Select existing security group

We'll create a new security group called '**launch-wizard-4**' with the following rules:

[x] Allow SSH traffic from
Helps you connect to your instance

Anywhere
0.0.0.0/0

[x] Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

[x] Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

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

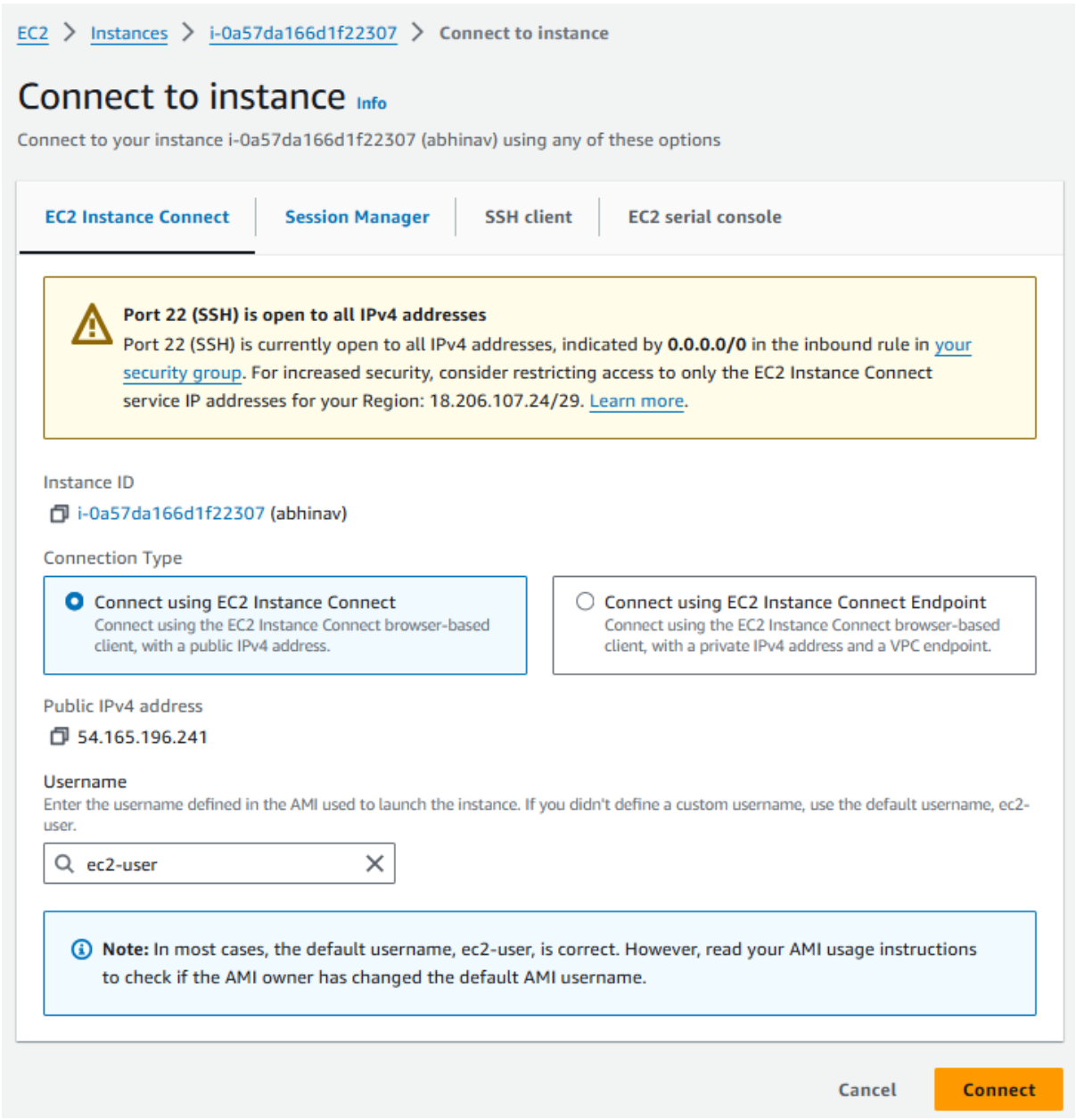
Instances (1) Info

Last updated less than a minute ago | Connect | Instance state | Actions | Launch instances

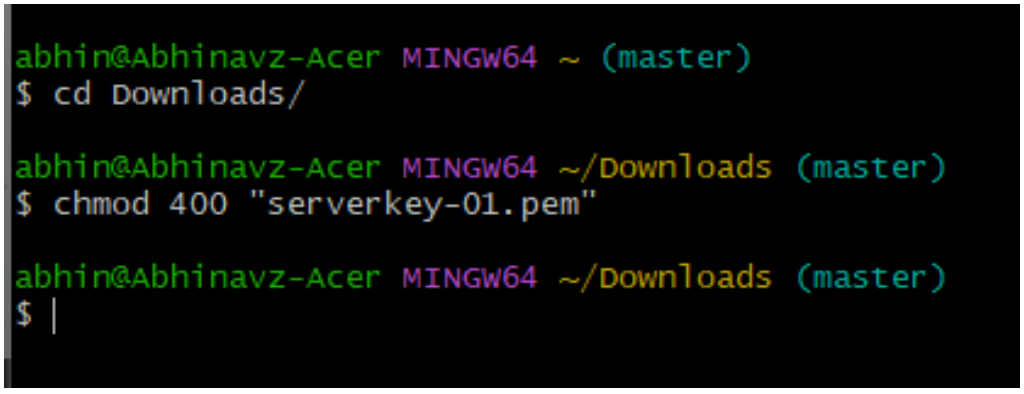
Find Instance by attribute or tag (case-sensitive) | Running | 1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
abhinav	i-0a57da166d1f22307	Running	t2.micro	2/2 checks passed	View alarms	us-east-1c	ec2-54-165-196-241.co...	54.165.196

2. **Then click on Id of that instance then click on connect**



3. **SSH into the created machine instance**
   - Give permissions to the current user to the downloaded pem file using - **chmod 400 <security_filename.pem>**



- Ssh using –
  ssh -i (keyname).pem (username)@(public ipv4 dns address)
  where keyname is name of the key you created. (server-01.pem). Other details can be found on the Instance dashboard.

```
abhin@Abhinavz-Acer MINGW64 ~/Downloads (master)
$ ssh -i "serverkey-01.pem" ec2-user@ec2-54-165-196-241.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-90-103 ~]$
```

4. Installation Of Docke007

- **sudo yum install docker -y**

```
[ec2-user@ip-172-31-90-103 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:13:41 ago on Sat Sep 14 03:42:27 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     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_ 2.2 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_6 2.5 MB/s | 183 kB     00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm 1.3 MB/s |  75 kB     00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023 1.3 MB/s |  58 kB     00:00
(5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_ 938 kB/s |  30 kB     00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rp 1.6 MB/s |  84 kB     00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm      1.7 MB/s |  83 kB     00:00
(8/10): runc-1.1.13-1.amzn2023.0.1.x86_64.rpm    21 MB/s | 3.2 MB     00:00
(9/10): containerd-1.7.20-1.amzn2023.0.1.x86_64  31 MB/s |  35 MB     00:01
(10/10): docker-25.0.6-1.amzn2023.0.2.x86_64.rp  28 MB/s |  44 MB     00:01
--------------------------------------------------------------------------------
Total                                            52 MB/s |  84 MB     00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  Installing       : runc-1.1.13-1.amzn2023.0.1.x86_64
  Installing       : containerd-1.7.20-1.amzn2023.0.1.x86_64
  Running scriptlet: containerd-1.7.20-1.amzn2023.0.1.x86_64
  Installing       : pigz-2.5-1.amzn2023.0.3.x86_64
  Installing       : libnftnl-1.2.2-2.amzn2023.0.2.x86_64
  Installing       : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64
  Installing       : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  Installing       : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  Installing       : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
```

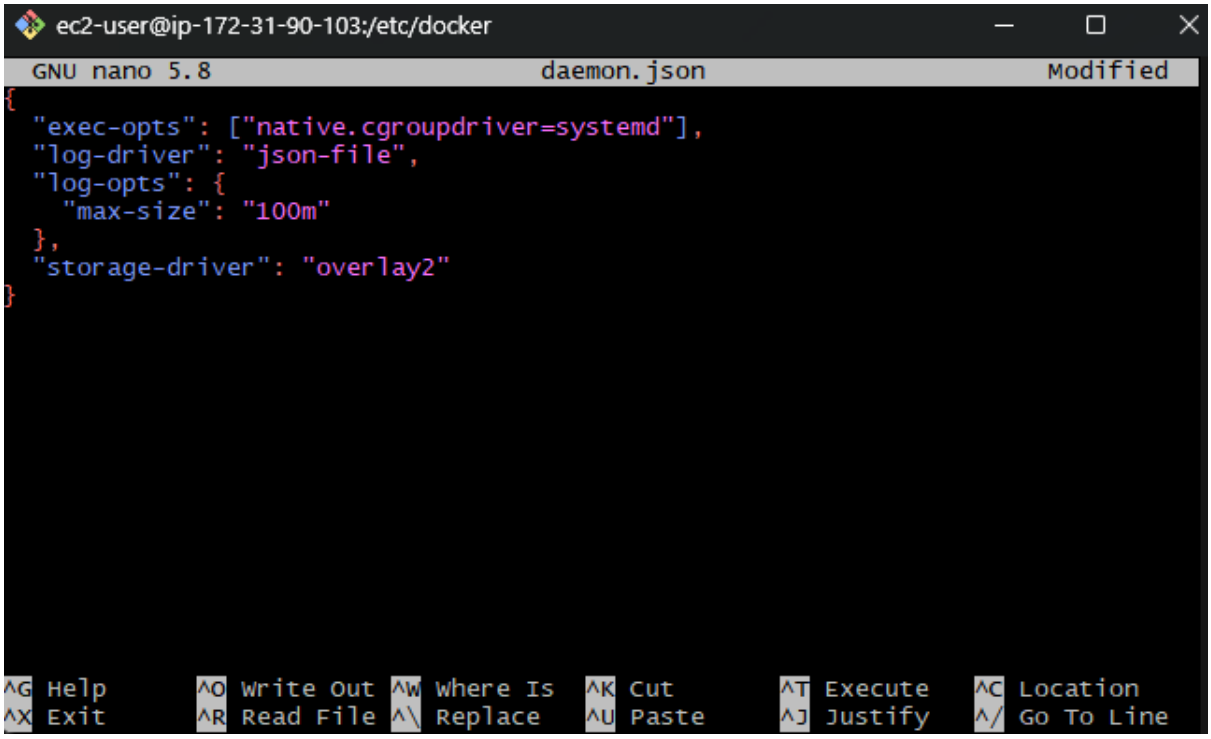
```
Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64                docker-25.0.6-1.amzn2023.0.2.x86_64
  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64     libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-90-103 ~]$
```

- Configure cgroup in a daemon.json
  (this can be done by creating the file and using **nano** text editor)

```
{
        "exec-opts": ["native.cgroupdriver=systemd"],
        "log-driver": "json-file",
        "log-opts": {
        "max-size": "100m"
        },
        "storage-driver": "overlay2"
}
```

```
[ec2-user@ip-172-31-90-103 docker]$ sudo nano daemon.json
[ec2-user@ip-172-31-90-103 docker]$
```



- Enable and start docker and also load the daemon.json
  sudo systemctl enable docker
  sudo systemctl daemon-reload
  sudo systemctl restart docker

```
[ec2-user@ip-172-31-90-103 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /us
r/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-90-103 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-90-103 docker]$
sudo systemctl restart docker
[ec2-user@ip-172-31-90-103 docker]$
```

- Check if docker is installed

```
[ec2-user@ip-172-31-90-103 docker]$ docker --version
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-90-103 docker]$
```

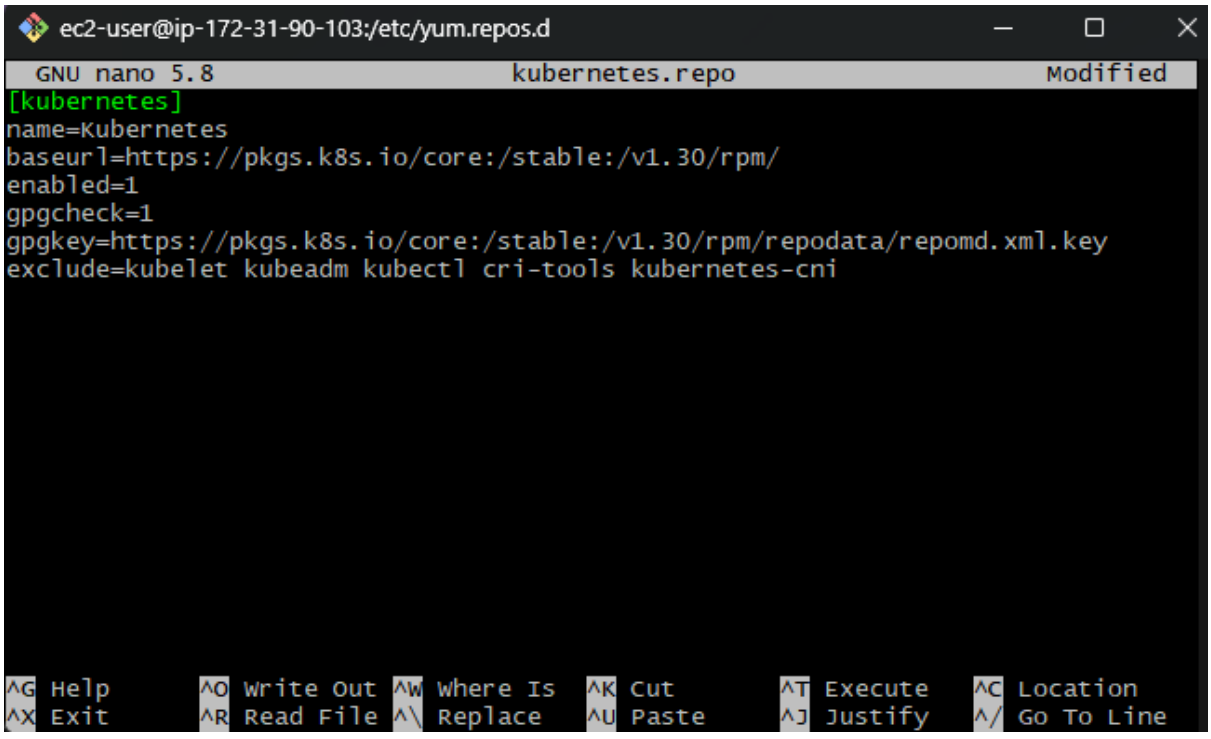
5. Install Kubernetes

- SELinux needs to be disabled before configuring kubelet
  sudo setenforce 0
  sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-90-103 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-90-103 docker]$
```

- Add Kubernetes using the repo
  (this is done by creating **kubernetes.repo** file in **/etc/yum.repos.d** and configuring it using **nano** editor)
  [kubernetes]
  name=Kubernetes
  baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
  enabled=1
  gpgcheck=1
  gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
  exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

```
[ec2-user@ip-172-31-90-103 docker]$ cd /etc/yum.repos.d/
[ec2-user@ip-172-31-90-103 yum.repos.d]$ ls
amazonlinux.repo  kernel-livepatch.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo nano kubernetes.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$ ls
amazonlinux.repo  kernel-livepatch.repo  kubernetes.repo
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```



- Update packages list using **sudo yum update**

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo yum update
Kubernetes                                      125 kB/s |  17 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- Install kubelet kubeadm kubectl

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo yum install -y kubelet kubeadm kub
ectl --disableexcludes=kubernetes
Last metadata expiration check: 0:00:42 ago on Sat Sep 14 04:08:20 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                 Repository     Size
================================================================================
Installing:
 kubeadm                 x86_64   1.30.5-150500.1.1       kubernetes     10 M
 kubectl                 x86_64   1.30.5-150500.1.1       kubernetes     10 M
 kubelet                 x86_64   1.30.5-150500.1.1       kubernetes     17 M
Installing dependencies:
 conntrack-tools         x86_64   1.4.6-2.amzn2023.0.2    amazonlinux   208 k
 cri-tools               x86_64   1.30.1-150500.1.1       kubernetes    8.6 M
 kubernetes-cni          x86_64   1.4.0-150500.1.1        kubernetes    6.7 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: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cttimeout-1.0.0-19.amzn2023 448 kB/s |  24 kB     00:00
(2/9): libnetfilter_cthelper-1.0.0-21.amzn2023. 409 kB/s |  24 kB     00:00
(3/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2. 1.5 MB/s |  30 kB     00:00
(4/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86 1.8 MB/s | 208 kB     00:00
(5/9): cri-tools-1.30.1-150500.1.1.x86_64.rpm    28 MB/s | 8.6 MB     00:00
(6/9): kubectl-1.30.5-150500.1.1.x86_64.rpm      23 MB/s |  10 MB     00:00
(7/9): kubeadm-1.30.5-150500.1.1.x86_64.rpm      18 MB/s |  10 MB     00:00
(8/9): kubelet-1.30.5-150500.1.1.x86_64.rpm      37 MB/s |  17 MB     00:00
(9/9): kubernetes-cni-1.4.0-150500.1.1.x86_64.r  20 MB/s | 6.7 MB     00:00
--------------------------------------------------------------------------------
Total                                            56 MB/s |  53 MB     00:00
```

```
Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  cri-tools-1.30.1-150500.1.1.x86_64
  kubeadm-1.30.5-150500.1.1.x86_64
  kubectl-1.30.5-150500.1.1.x86_64
  kubelet-1.30.5-150500.1.1.x86_64
  kubernetes-cni-1.4.0-150500.1.1.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-172-31-90-103 yum.repos.d]$
```

- 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-172-31-90-103 yum.repos.d]$ . sudo swapoff -a
-bash: sudo: No such file or directory
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo swapoff -a
[ec2-user@ip-172-31-90-103 yum.repos.d]$ echo "net.bridge.bridge-nf-call-iptable
s=1" | sudo tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

6. **Initialize the Kubecluster**

   sudo kubeadm init --podnetwork-cidr=10.244.0.0/16

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo kubeadm init --pod-network-cidr=10
.244.0.0/16 --ignore-preflight-errors=all
I0914 04:12:17.448521   27990 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the requir
ed 2
        [WARNING Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 's
ystemctl 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 inte
rnet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config
 images pull'
W0914 04:12:17.711154   27990 checks.go:844] detected that the sandbox image "re
gistry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used
 by kubeadm.It is recommended to use "registry.k8s.io/pause:3.9" as the CRI sand
box image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
```

```
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 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

- Save the token

```
kubeadm join 172.31.90.103:6443 --token 0zk8w3.xyegkydsy42vfscm \
        --discovery-token-ca-cert-hash
sha256:31c672892b19dcb869fc46362d189234128f5bfc302bd41ae8c6078c56173f00
```

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

mkdir -p $HOME/.kube

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ mkdir -p $HOME/.kube
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ sudo cp -i /etc/kubernetes/admin.conf $
HOME/.kube/config
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$
sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-90-103 yum.repos.d]$
```

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

```
[ec2-user@ip-172-31-90-103 yum.repos.d]$ kubectl apply -f https://raw.githubuser
content.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-172-31-90-103 yum.repos.d]$
```

7. Deploy nginx server

- Apply deployment using this following command:
  kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml

```
[ec2-user@ip-172-31-90-103 docker]$ kubectl apply -f https://k8s.io/examples/pods/s
imple-pod.yaml
pod/nginx created
```

- use **kubectl get nodes** to check whether the pod gets created or not

```
[ec2-user@ip-172-31-90-103 docker]$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          12s
```

- To convert state from pending to running use following command:
  kubectl describe pod nginx (This command will help to describe the pods it gives reason for failure as it shows the untolerated taints which need to be untainted.)

```
[ec2-user@ip-172-31-90-103 docker]$ kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           <none>
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
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-k4lj6 (ro)
    Type:                     Projected (a volume that contains injected data from m
ultiple 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 3
00s
                              node.kubernetes.io/unreachable:NoExecute op=Exists for
 300s
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  7s    default-scheduler  0/1 nodes are available: 1 no
de(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption:
0/1 nodes are available: 1 Preemption is not helpful for scheduling.
```

- check pod status

```
[ec2-user@ip-172-31-90-103 ~]$ kubectl get pods
NAME    READY   STATUS    RESTARTS      AGE
nginx   1/1     Running   1 (6s ago)    90s
```

- mention the port you want to host

```
[ec2-user@ip-172-31-90-103 ~]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

8. 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** If the response is 200 OK and you can see the Nginx server name, your deployment was successful. We have successfully deployed our Nginx server on our EC2 instance.

```
[ec2-user@172-31-90-103 ~]$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Linux)
Date: Sat, 14 Sep 2024 12:31:53 GMT
Content-Type: text/html
Content-Length: 612
Connection: keep-alive
```

Conclusion:

An AWS EC2 Linux instance was set up, and Docker and Kubernetes were installed. Kubernetes was initialized successfully, and the required commands were executed. Flannel was installed as a networking plugin. Although there was an initial error with the Nginx deployment, it was eventually deployed successfully using the `simple-pod.yml` file and accessed via localhost on port 8080.