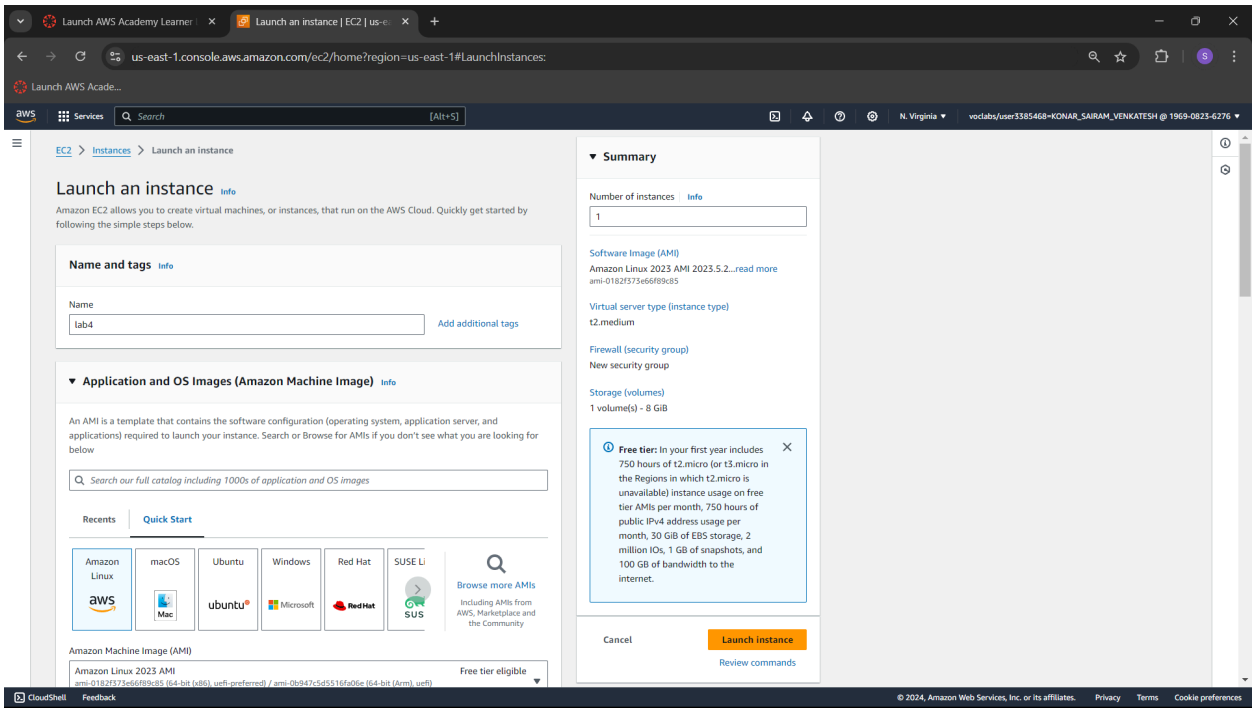
Sairam Konar D15C 27 2024-25

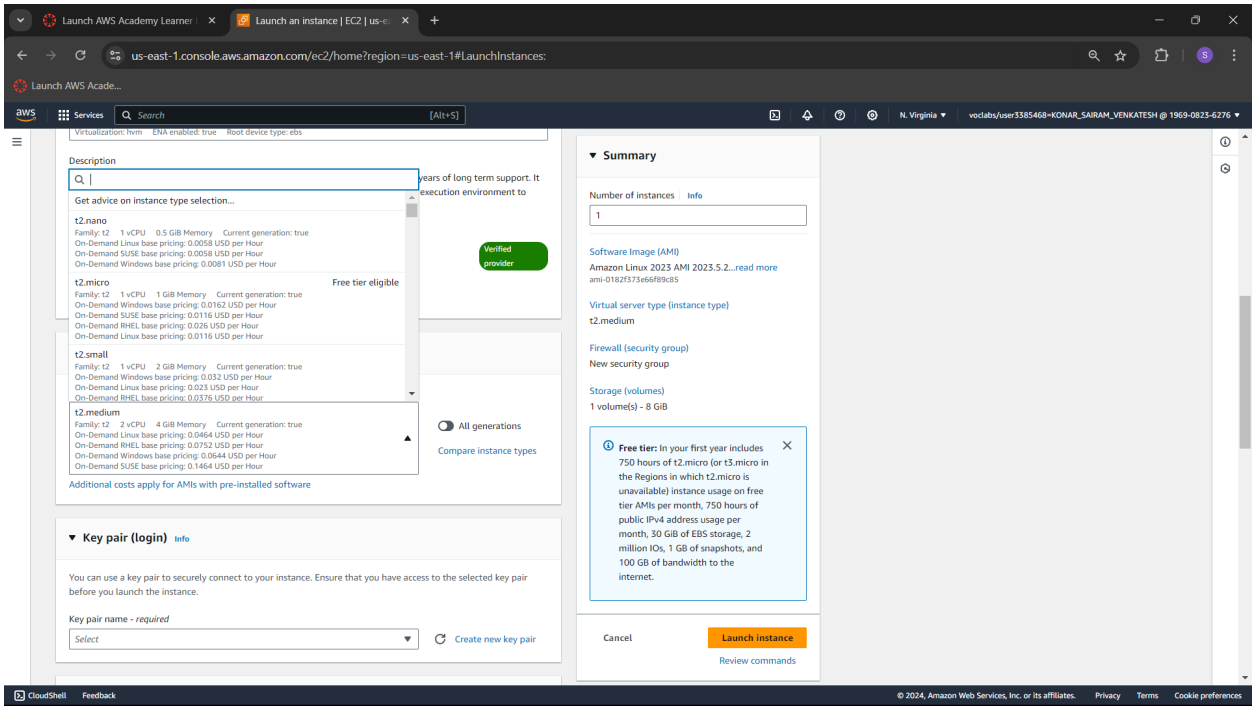
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Step 1: Set Up EC2 Instances.

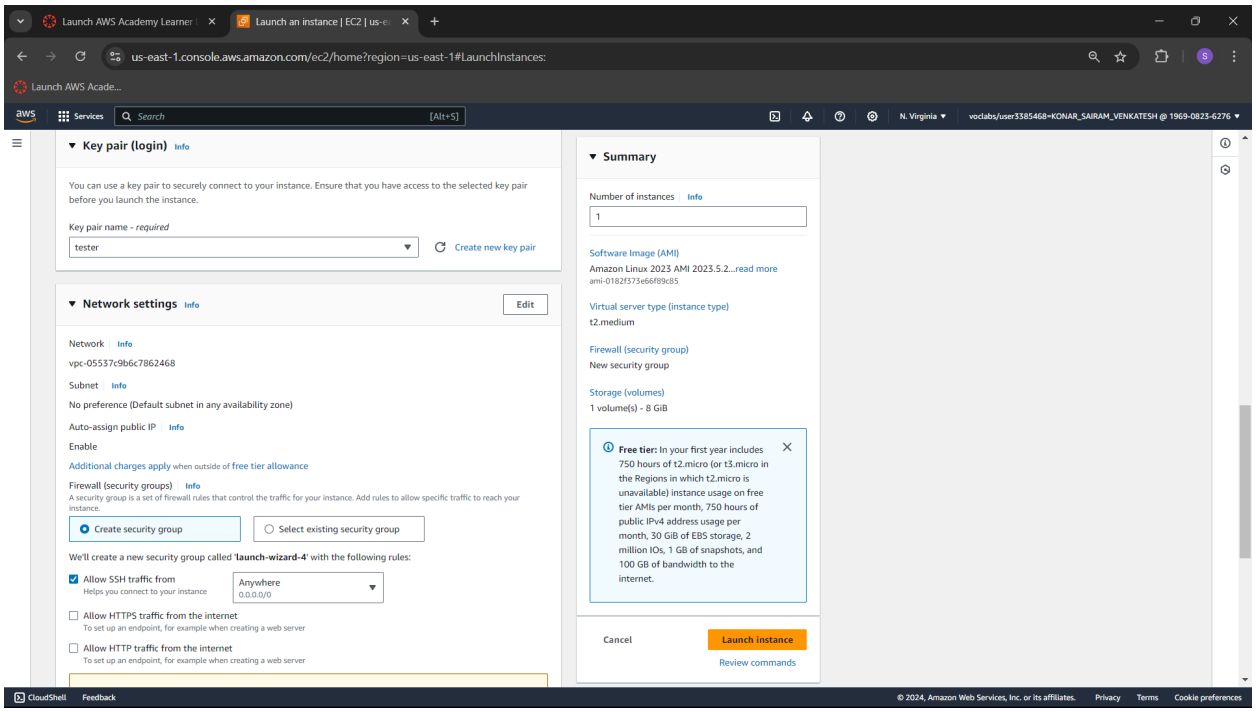
1) Setup 1 EC2 Instance
   Select Amazon Linux as the OS image.



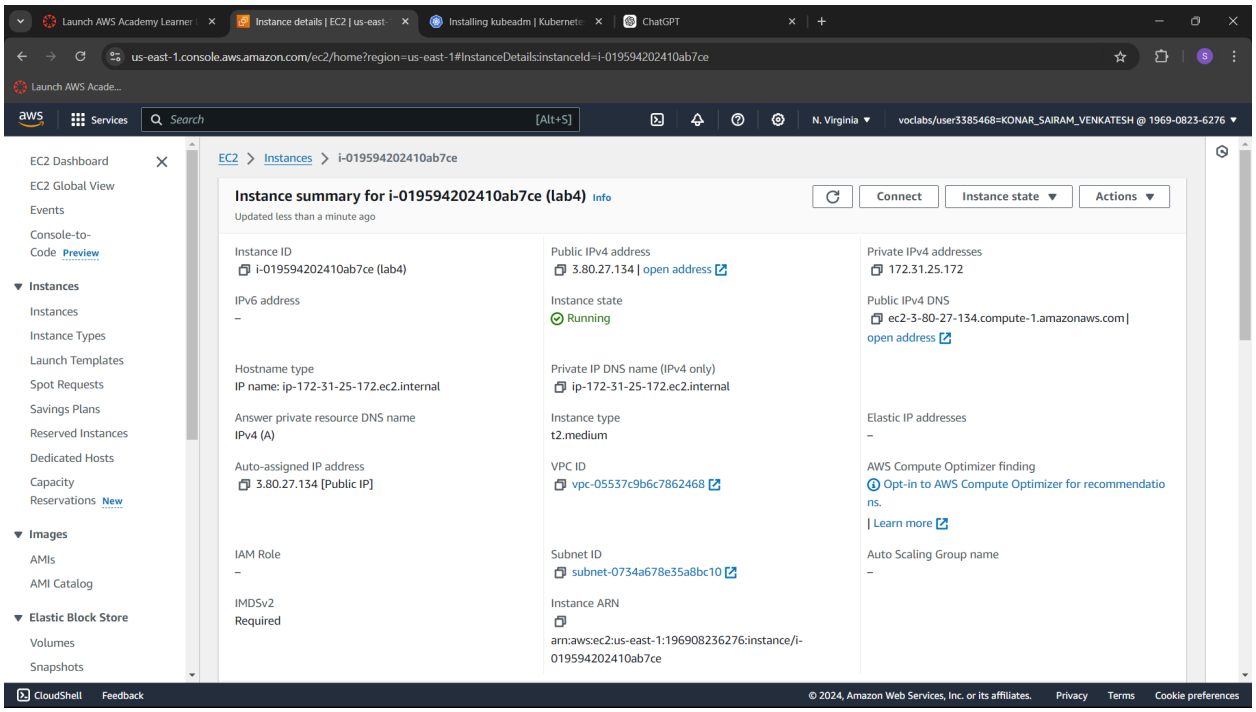
IMPORTANT: The default instance type and free one provided by AWS is t2.micro, which provides only 1CPU and 1 GiB of memory. For running Kubernetes, a minimum of 2 CPUs and 2GiB of RAM is required, hence change **t2.micro** to **t2.medium.**



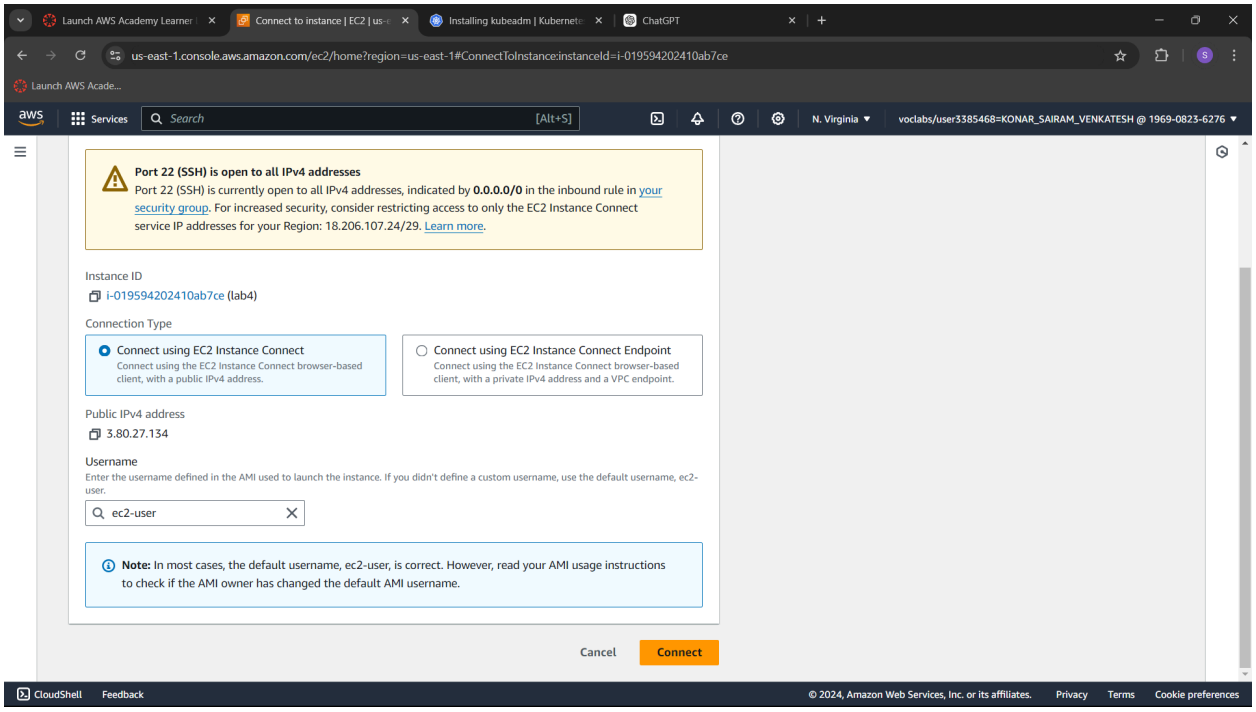
2) Select a key pair, it may be the default (vockey provided by AWS Academy) or you may create one. Click on create.



3) Once created, go back to the instances page. Click on the instance id. Then, click on connect.



4) Maintain the default options and click on connect.



Step 2: Installation of Docker

1) Use command

‘sudo su’

This allows you to act as the root user of the terminal

```
[ec2-user@ip-172-31-25-172 ~]$ sudo su
[root@ip-172-31-25-172 ec2-user]#
```

2) We can install docker using YUM(Yellowdog Updater, Modified). Use the command

‘yum install docker -y’

```
[root@ip-172-31-25-172 ec2-user]# yum install docker -y
Last metadata expiration check: 0:37:57 ago on Fri Sep 13 17:56:13 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
```

```
  Running scriptlet: docker-25.0.6-1.amzn2023.0.2.x86_64
  Installing       : docker-25.0.6-1.amzn2023.0.2.x86_64
  Running scriptlet: docker-25.0.6-1.amzn2023.0.2.x86_64
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
  Verifying        : docker-25.0.6-1.amzn2023.0.2.x86_64
  Verifying        : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  Verifying        : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
  Verifying        : libcgroup-3.0-1.amzn2023.0.1.x86_64
  Verifying        : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  Verifying        : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64
  Verifying        : libnftnl-1.2.2-2.amzn2023.0.2.x86_64
  Verifying        : pigz-2.5-1.amzn2023.0.3.x86_64
  Verifying        : runc-1.1.13-1.amzn2023.0.1.x86_64

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.a
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64             libcgroup-3.0-1.amzn2023.0.1.x86_64              libnetfilter_conntrack-
  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
  runc-1.1.13-1.amzn2023.0.1.x86_64

Complete!
[root@ip-172-31-25-172 ec2-user]#
```

3) Now, configure a daemon.json file by using the following chain of commands.

- cd /etc/docker
- cat <<EOF | sudo tee /etc/docker/daemon.json

  {

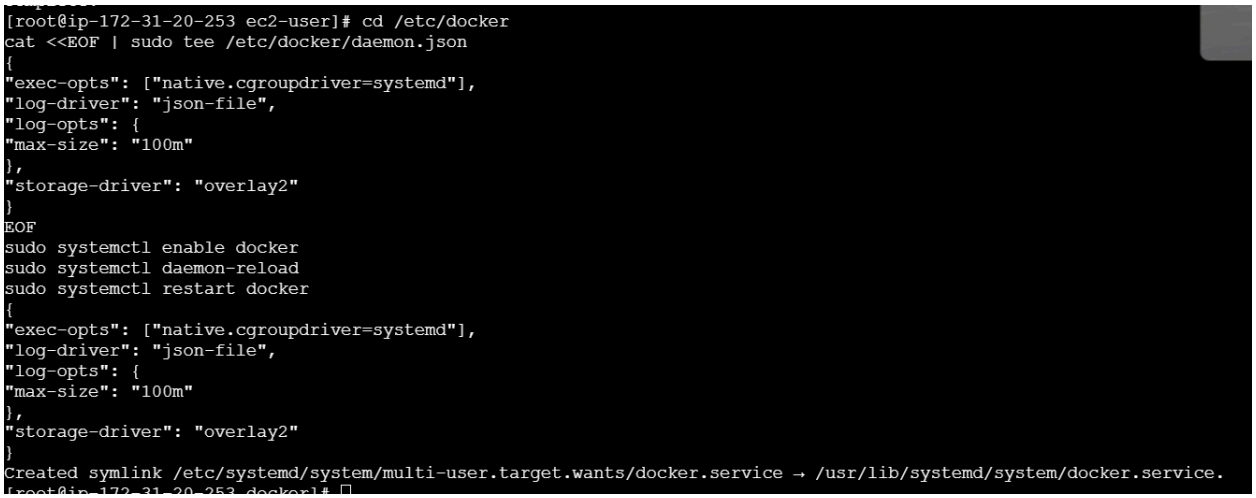
  "exec-opts": ["native.cgroupdriver=systemd"],

  "log-driver": "json-file",

  "log-opts": {

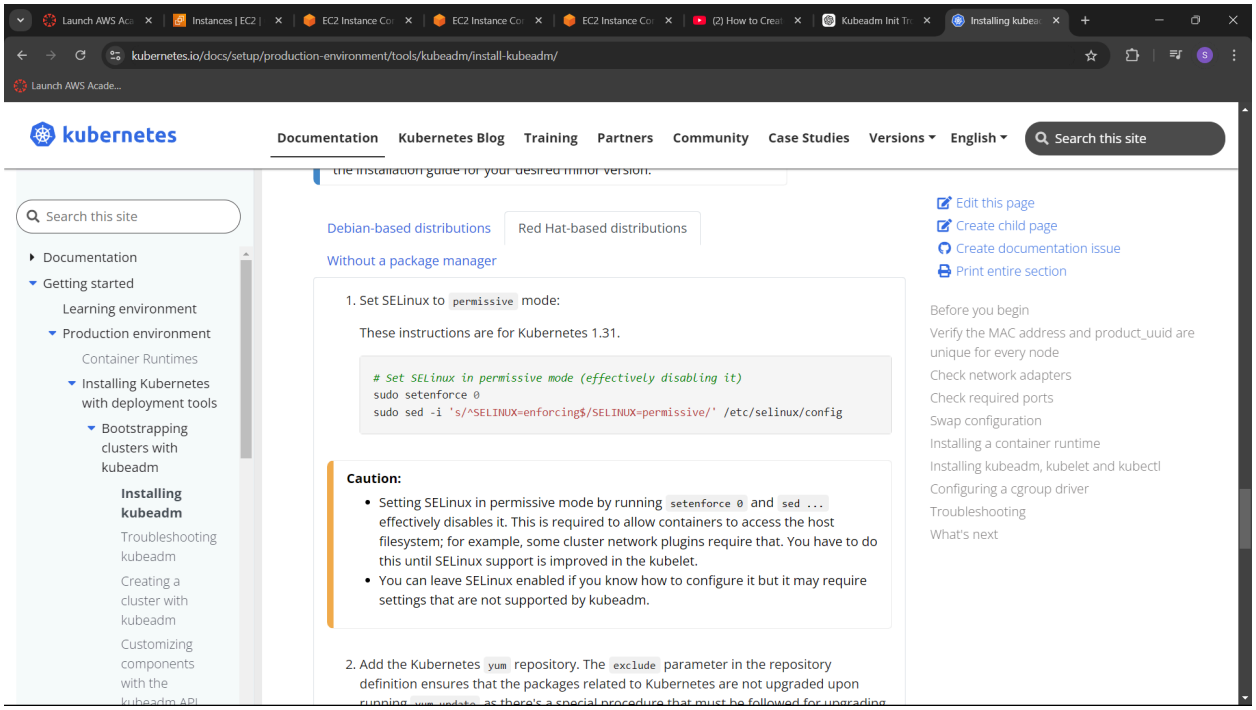
```
  "max-size": "100m"
 },
 "storage-driver": "overlay2"
}
EOF
```

- sudo systemctl enable docker
- sudo systemctl daemon-reload
- sudo systemctl restart docker



Step 3: Installing Kubernetes

1) For installing kubernetes, we will be using kubeadm, a framework used for creating kubernetes clusters using command line.
   https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
   The following will be visibe when you visit the website.



2) Select **red hat-based distributions** as amazon linux is based on red hat.

sudo setenforce 0
→ sets SELinux to permissive mode

sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
→ edits the SELinux configuration file (/etc/selinux/config) to make the change persistent across reboots. If not used, SELinux reverts to enforcing mode after reboot.

Setting SELinux to permissive mode during Kubernetes installation prevents permission-related issues with container runtimes and components that may not function correctly under SELinux's enforcing policies.

Run the following commands:

- sudo setenforce 0
- sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```
[root@ip-172-31-25-172 docker]# sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

- 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
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF

This comamnd is a repository script to create a kubernetes repository

```
[root@ip-172-31-25-172 docker]# 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
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
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
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[root@ip-172-31-25-172 docker]#
```

- yum repolist

This command shows the repositories created on the machine.

```
[root@ip-172-31-21-124 ec2-user]# yum repolist
repo id                                                                repo name
amazonlinux                                                            Amazon Linux 2023 repository
kernel-livepatch                                                       Amazon Linux 2023 Kernel Livepatch repository
kubernetes                                                             Kubernetes
```

Next step is to install kubelet, kubeadm, kubectl

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

```
[root@ip-172-31-25-172 docker]# sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Kubernetes                                                                                    67 kB/s | 9.4 kB     00:00
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                                 264 kB/s |  24 kB     00:00
(2/9): libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64.rpm                                241 kB/s |  24 kB     00:00
```

```
  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!
[root@ip-172-31-25-172 docker]#
```

Now, we need to enable the kubelet service. Run the command

- sudo systemctl enable --now kubelet

```
[root@ip-172-31-25-172 docker]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
```

- sudo swapoff -a
- echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
- sudo sysctl -p

Use these commands to establish a network bridge.

```
[root@ip-172-31-25-172 docker]# 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
[root@ip-172-31-25-172 docker]#
```

3) Firstly, we need to initialize kubernetes. For this, run the command:
    - sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU,Mem

```
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 172.31.25.172:6443 --token 47aiu9.pf1dju3zjdidpkpq \
        --discovery-token-ca-cert-hash sha256:bfb41befc2b82c1d7d7d476d008cbd229715ee92a70ee50adc11bb6a279ed781
[root@ip-172-31-25-172 docker]#
```

4) From the output, we receive the following commands:

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

   Run these commands.

```
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
```

```
[root@ip-172-31-21-124 ec2-user]#   mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

5) Add a common networking plugin ‘flannel’ using this command
   - kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
[root@ip-172-31-25-172 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
[root@ip-172-31-25-172 docker]#
[root@ip-172-31-25-172 docker]#
```

Step 3: Deploy nginx server

1) Now that the cluster is set, apply the deployment file of nginx using this command
   - kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml

```
[root@ip-172-31-28-78 docker]# kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml
pod/nginx created
[root@ip-172-31-28-78 docker]#
```

2) Use the command
   - kubectl get pods

   To get the list of pods in the cluster.

```
[root@ip-172-31-28-78 docker]# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          23s
```

This output shows that the pod is in a ‘PENDING’ state, to change it to ‘RUNNING’ state, run the following commands.

- kubectl describe pod nginx: Provides details about your pod
  This command is used to get details about the pod and potential issues with the pod

```
[root@ip-172-31-27-25 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:
```

```
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  10s   default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-r
: 0/1 nodes are available: 1 Preemption is not helpful for scheduling.
[root@ip-172-31-27-25 docker]#
```

3) From this output, we get to know that the node has some untolerated taint. To remove this, use
   - kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-

```
[root@ip-172-31-23-234 docker]# kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-23-234.ec2.internal untainted
```

4) Now, we check the status of the pod by running ‘kubectl get pods’ again

```
[root@ip-172-31-27-25 docker]# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          2m23s
[root@ip-172-31-27-25 docker]#
```

5) Now, change the port to which you want to host your server on using command
    - kubectl port-forward nginx <port number you want to host on>:80

```
[root@ip-172-31-23-234 docker]# kubectl port-forward nginx-deployment-77d8468669-s77nc 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

6) To check whether the deployment was successful, run the command
    - curl --head http://127.0.0.1:<port number given by you>

   If the terminal returns a status code of 200, it means that the deployment is successful.

Conclusion:

In this experiment, we have learned how to deploy an nginx server to a kubernetes cluster. We also learned how to tackle any intolerable taints that tend to give issues while deploying the server. We also learned how to set the port on which you want to host the server.