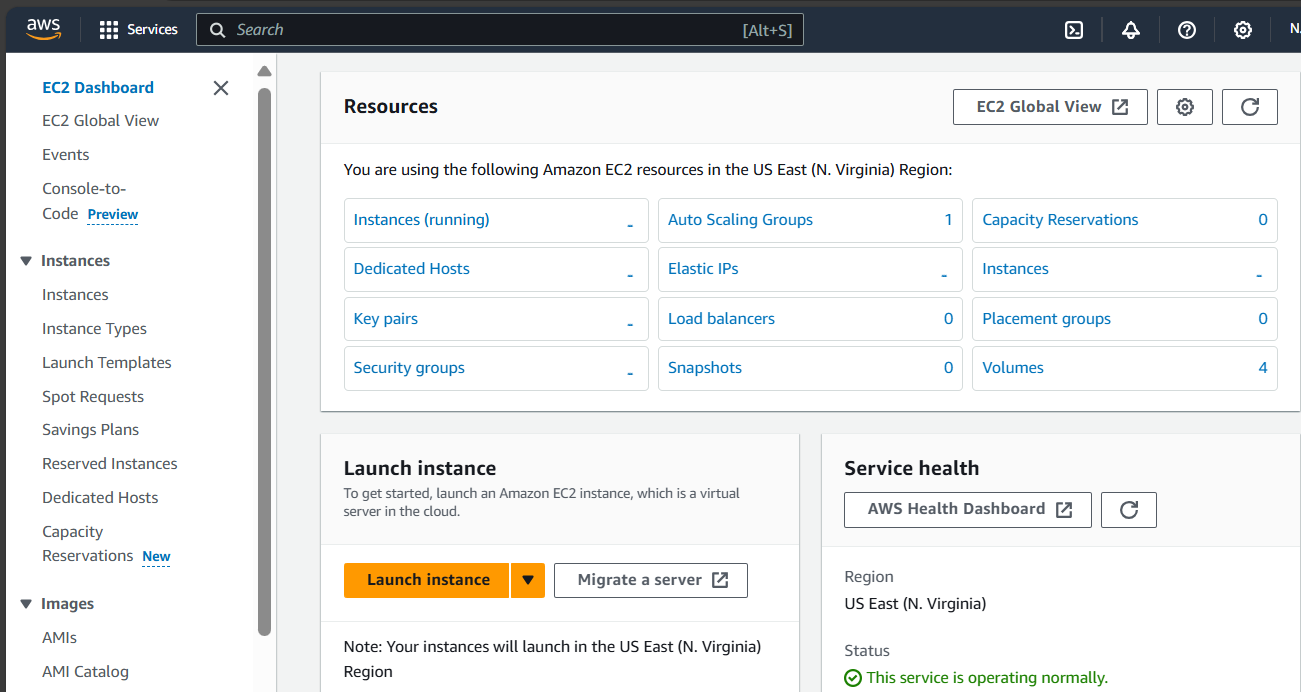
**Experiment 4**

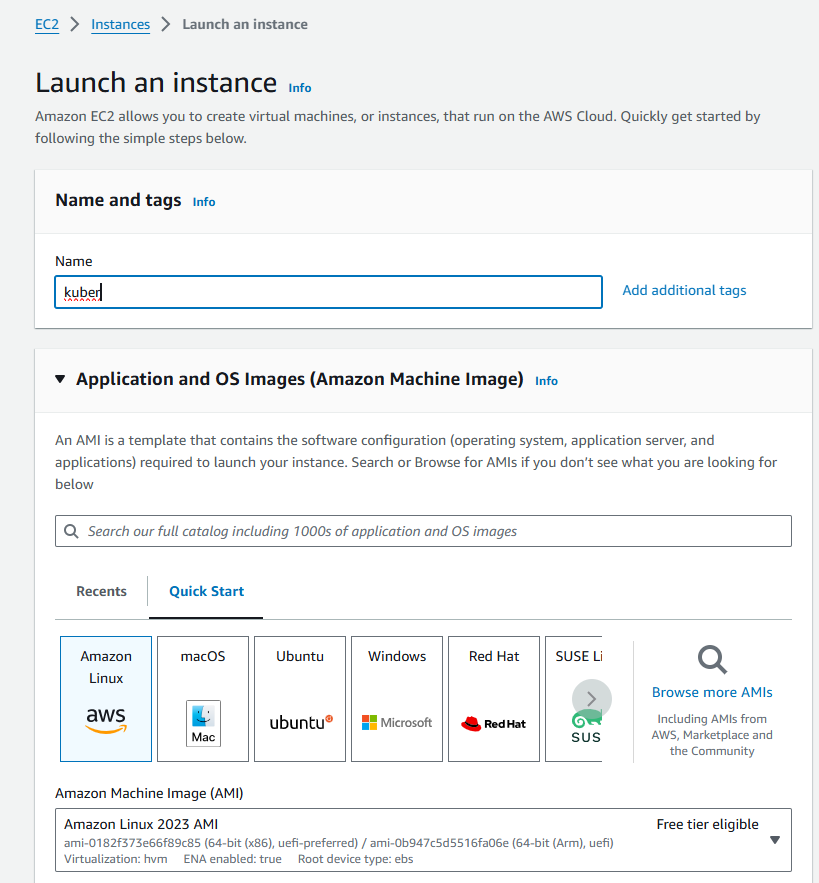
**Aim:** To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and

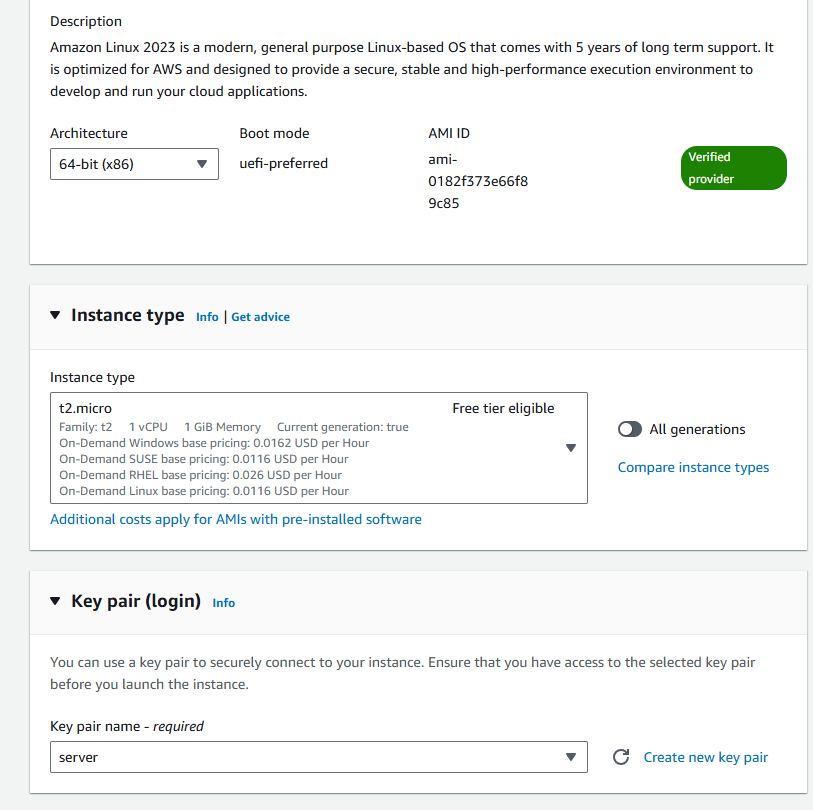
deploy Your First Kubernetes Application.

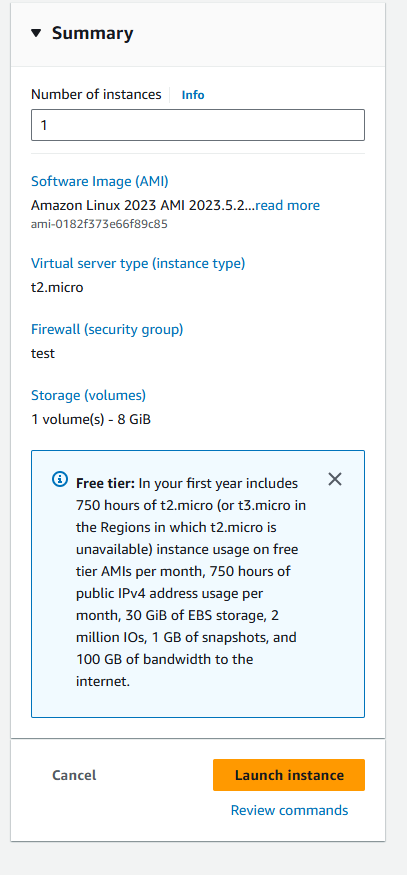
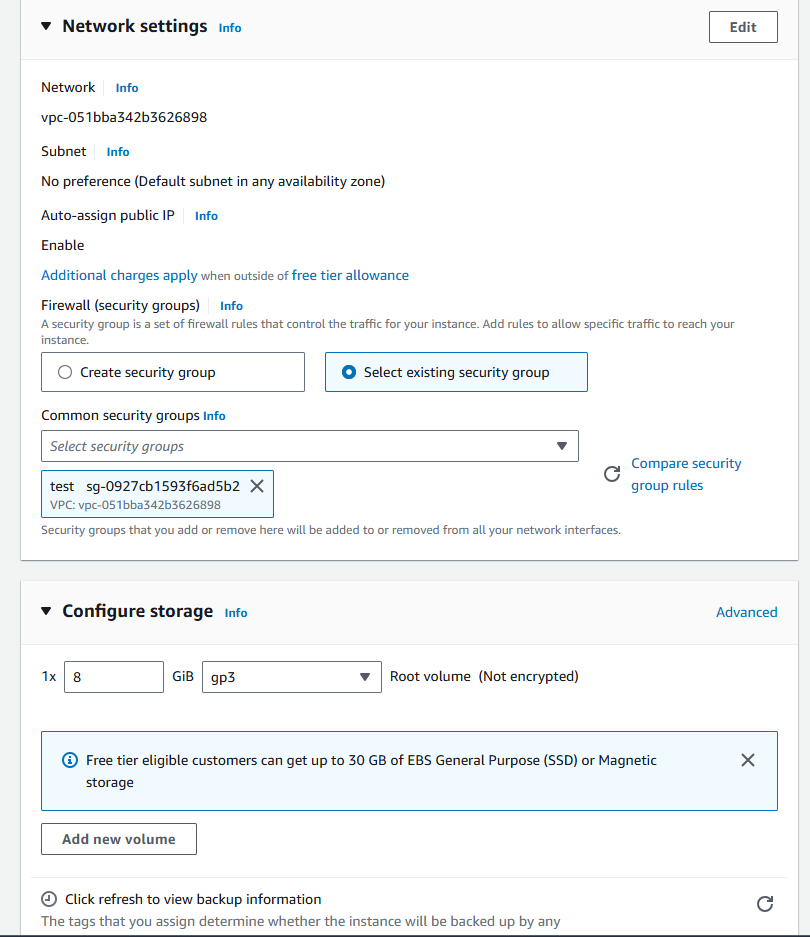
**Procedure:**

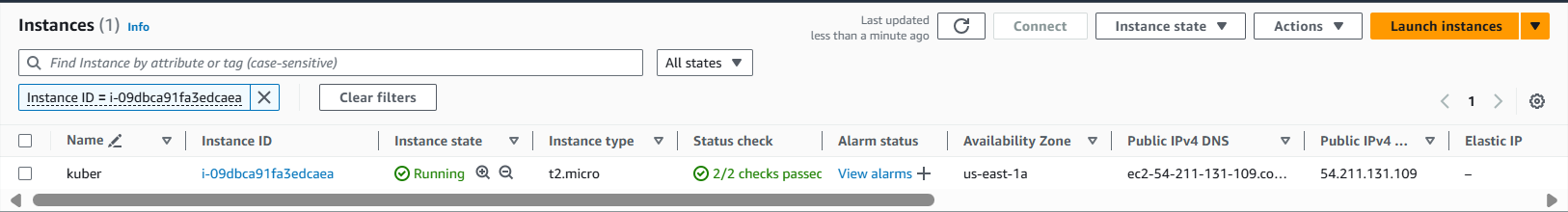
1. **Creation Of EC-2 instance**

* Create an EC2 AWS Linux instance on AWS .also edit the Security Group Inbound Rules to allow SSH. then select the t2.micro instance type

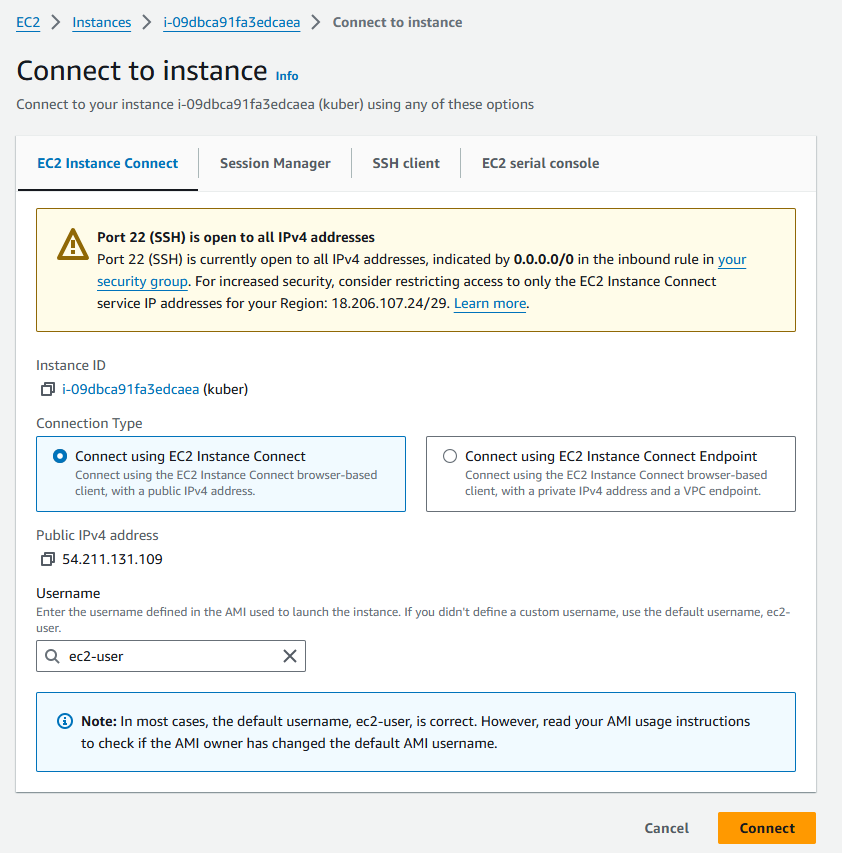








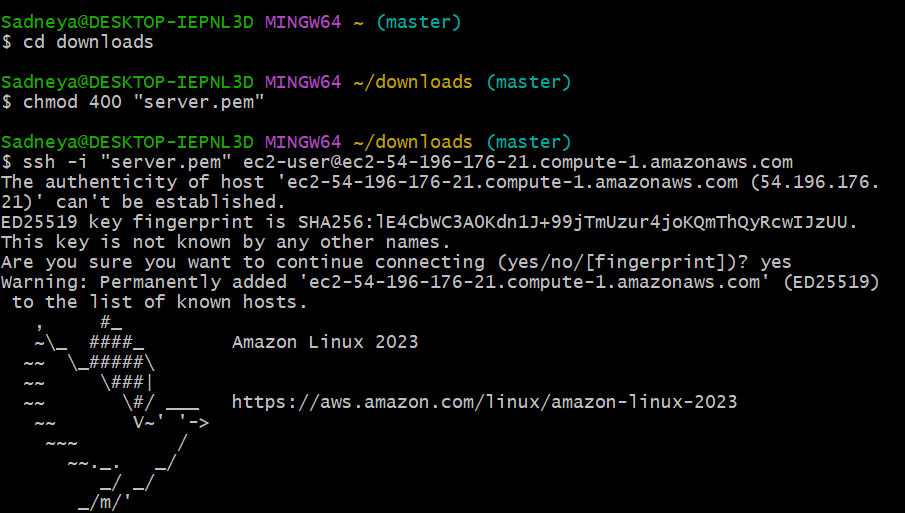
* Thus Kuber named -instance gets created.Then click on Id of that instance then click on connect button you will se this:



* Then go into SSH client where you will get this command

Chmod 400 “keyname.pem”

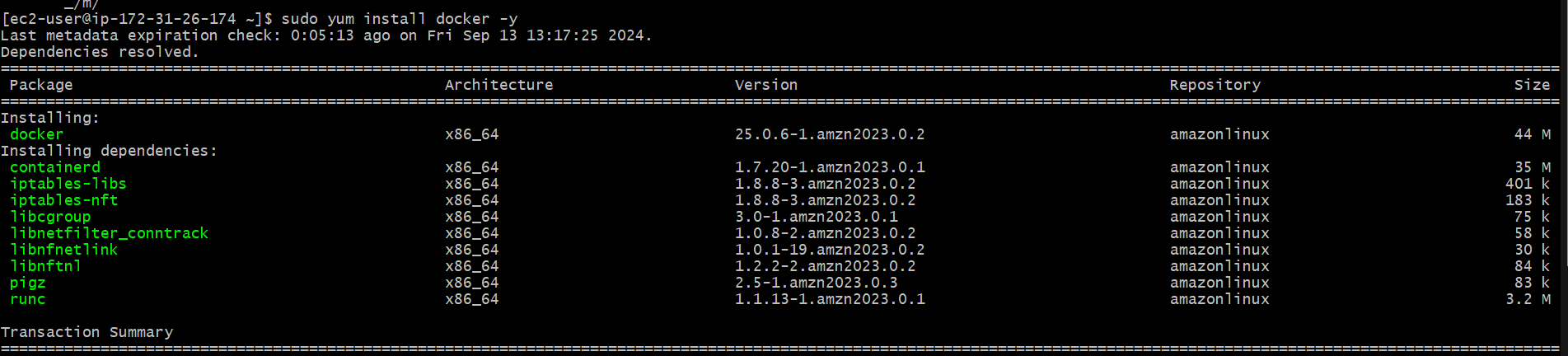
ssh -i <keyname>.pem ubuntu@<public\_ip\_address> copy it and then connect it and run the following command for establishing connection.(I have entered this command on git bash where i entered in downloads where server.pem is stored then as the key is not accessible hence we need to change its mode using chmod 400 “key name.pem”. Then use the given command for making connections).

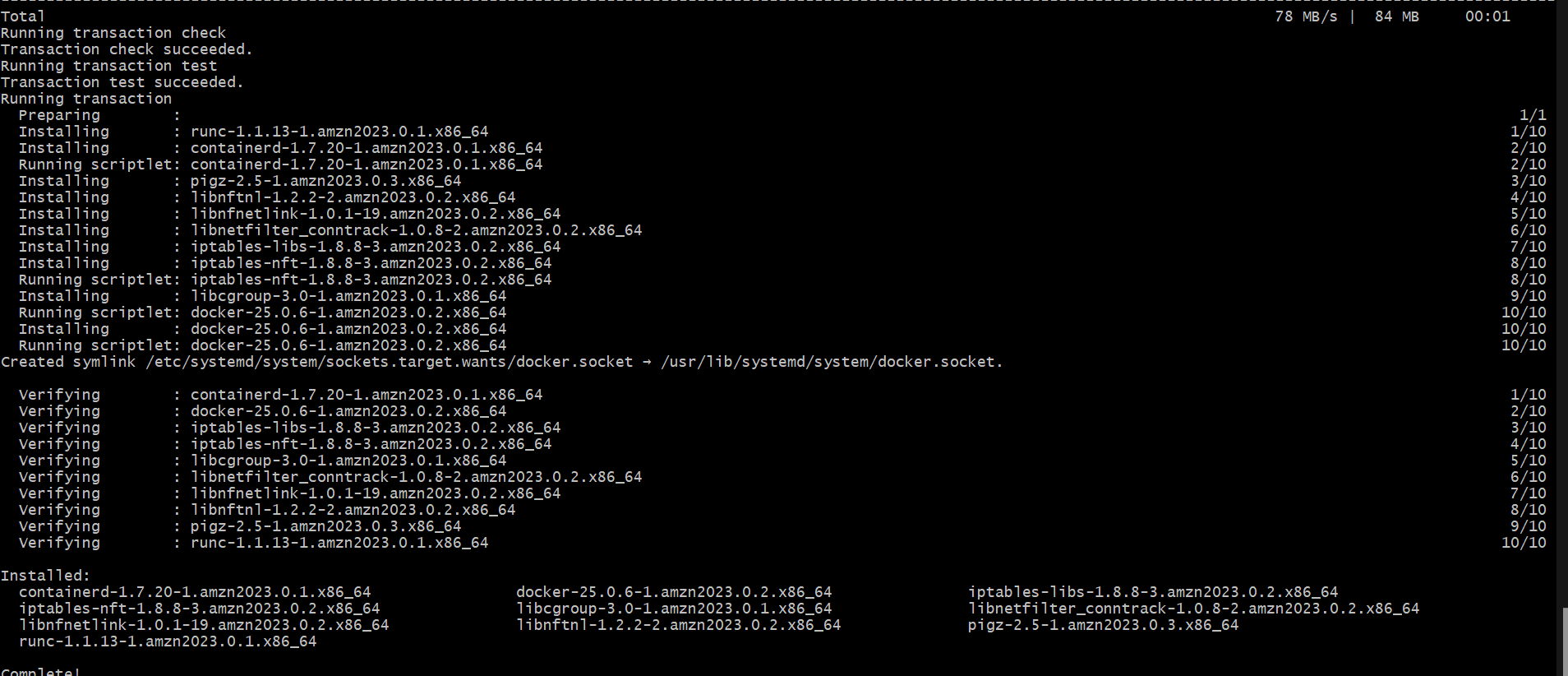


**2. Installation of Docker**

1. . For installation of Docker into the machines run the following command:

sudo yum install docker -y





* Then, configure cgroup in a daemon.json file by using following commands

cd /etc/docker

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

{

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

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

"log-opts": {

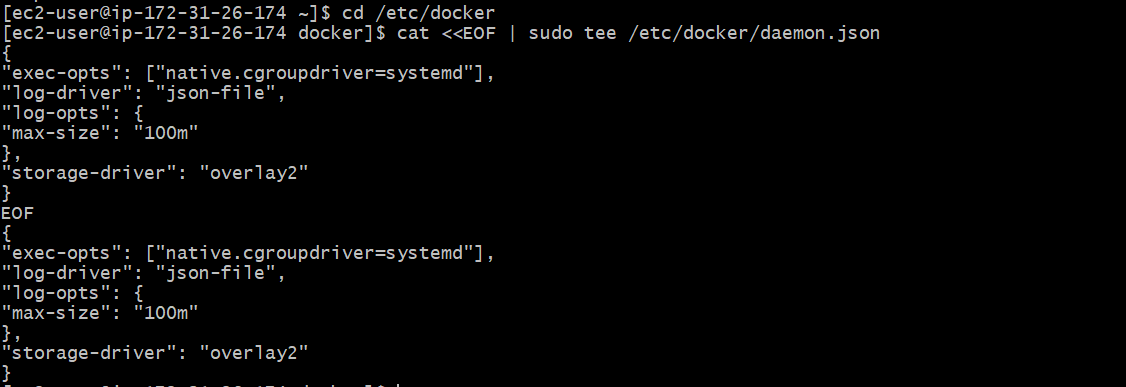
"max-size": "100m"

},

"storage-driver": "overlay2"

}

EOF

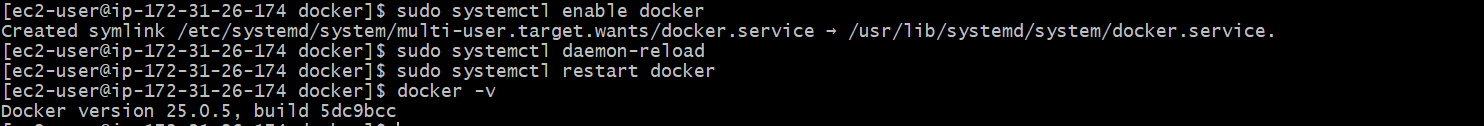


* Then after this run the following command to enable and start docker and also to load the daemon.json file.

sudo systemctl enable docker

sudo systemctl daemon-reload

sudo systemctl restart docker



* docker -v



**3. Then Install Kubernetes with the following command**.

* SELinux needs to be disable before configuring kubelet thus run the following command

sudo setenforce 0

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

* Here We are adding kubernetes using the repository whose command is given below.

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/

enabled=1

gpgcheck=1

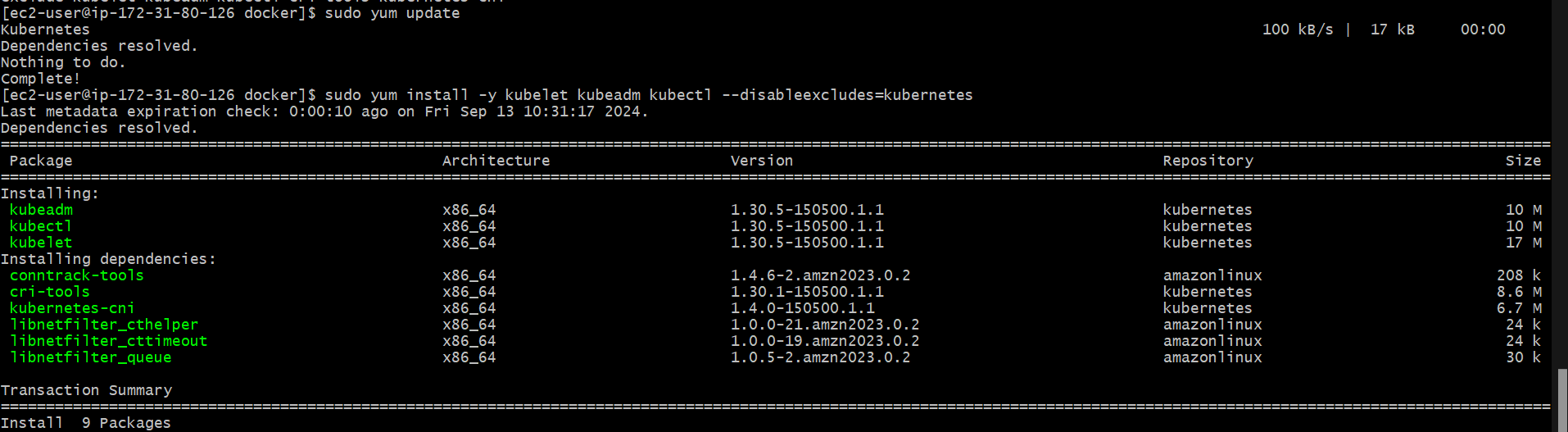
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key

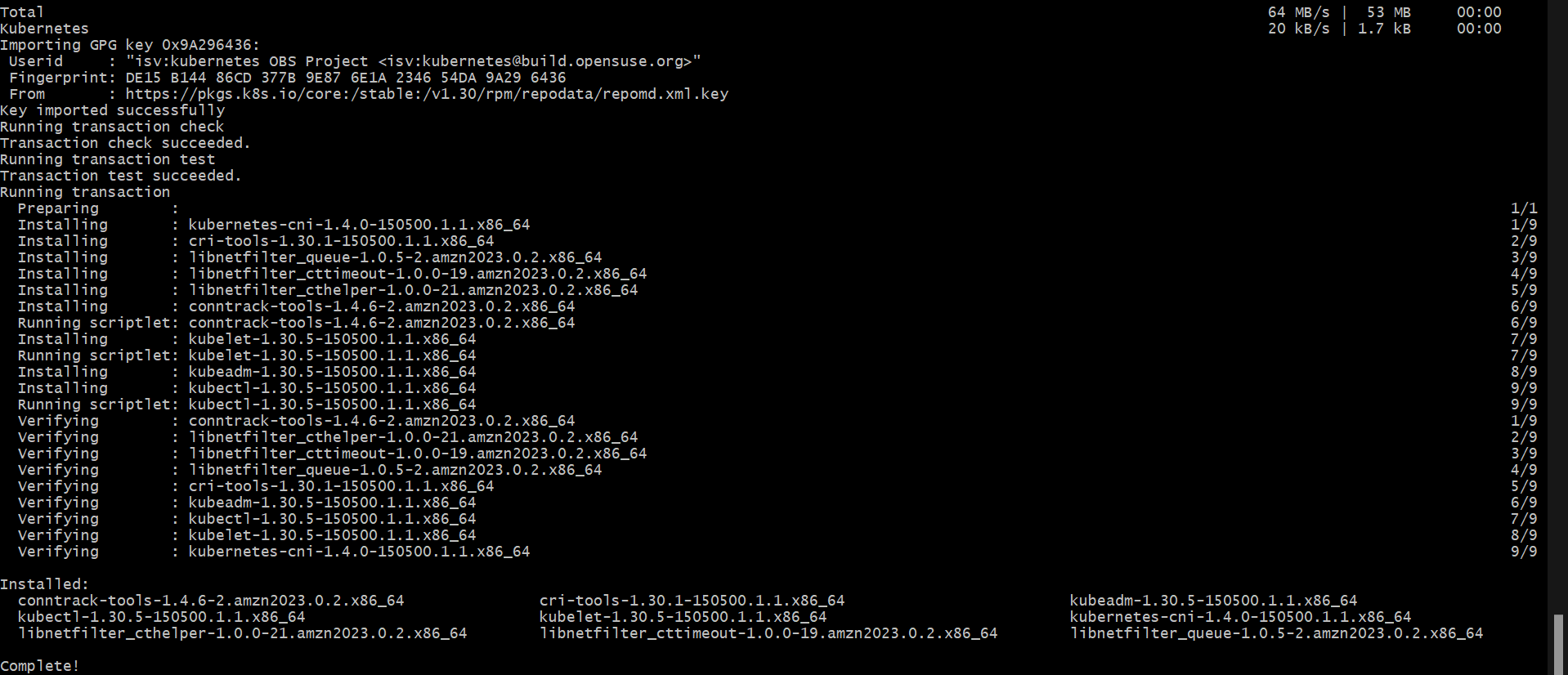
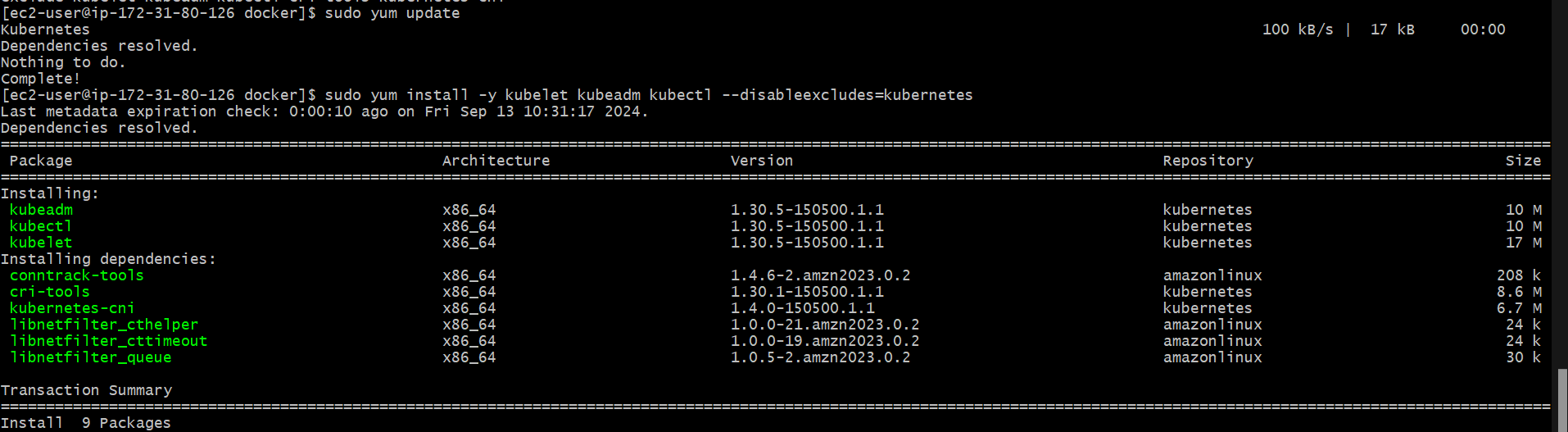
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

EOF



* After that Run following command to make the updation and also to install kubelet ,kubeadm, kubectl: sudo yum update



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

* After installing Kubernetes, we need to configure internet options to allow bridging.

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

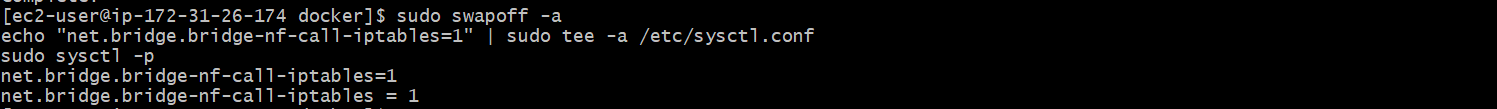
* After installing Kubernetes, we need to configure internet options to allow bridging.

sudo swapoff -a

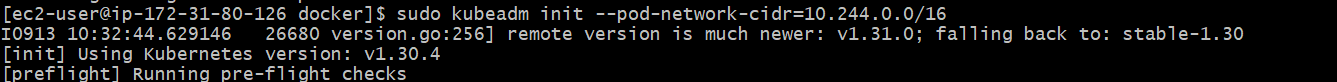
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a

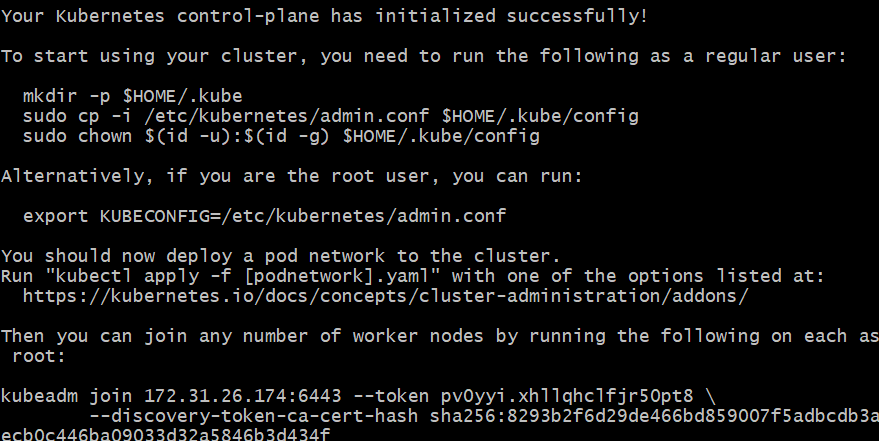
/etc/sysctl.conf

sudo sysctl -p



**4. Initialize the Kubecluster**

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



* copy the token and save for future use .

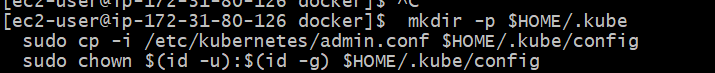
kubeadm join 172.31.26.174:6443 --token pv0yyi.xhllqhclfjr50pt8 \--discovery-token-ca-cert-hash sha256:8293b2f6d29de466bd859007f5adbcdb3aecb0c446ba09033d32a5846b3d434f

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

mkdir -p $HOME/.kube

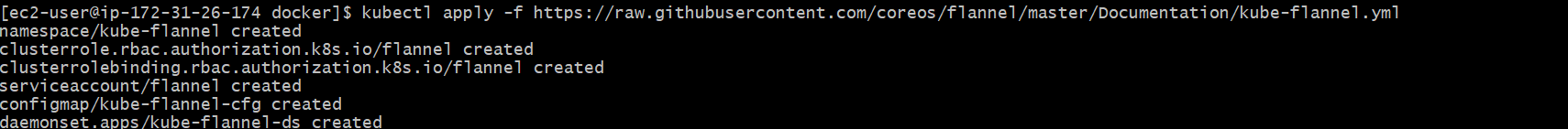
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

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

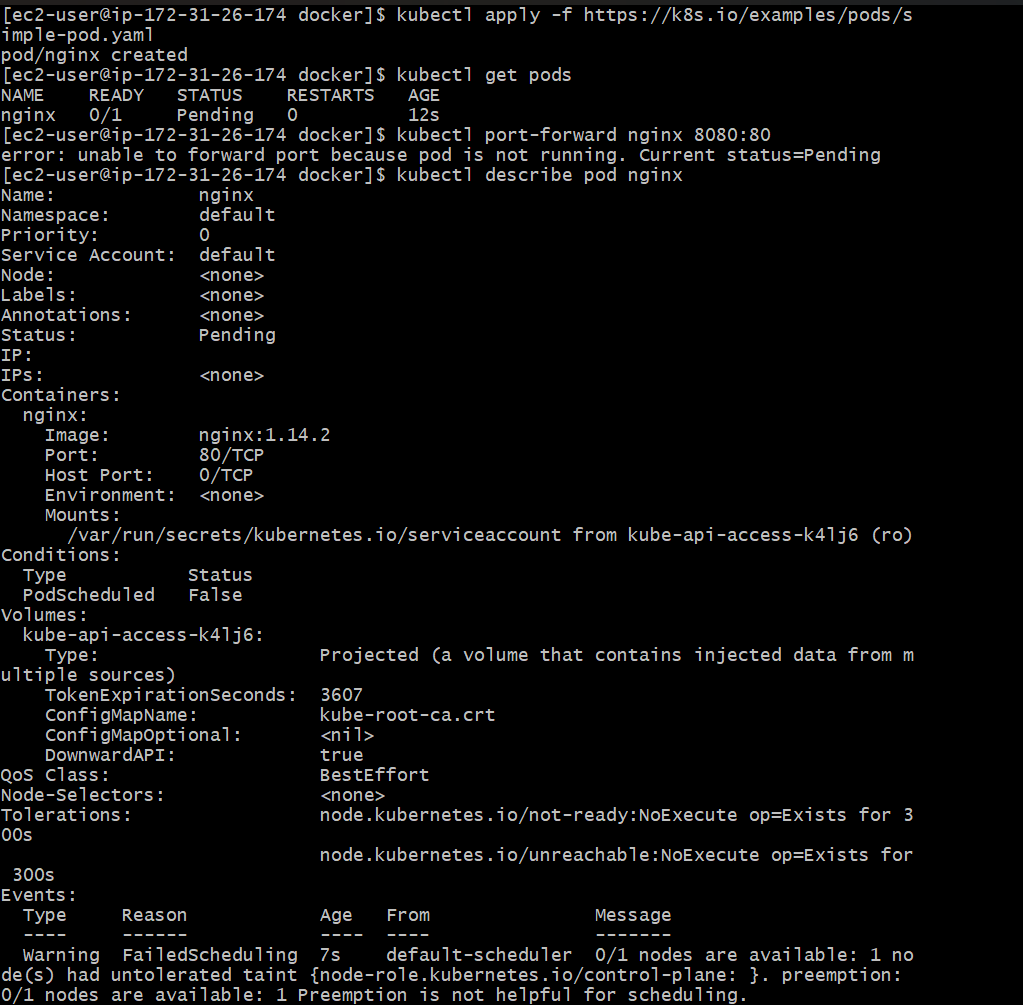
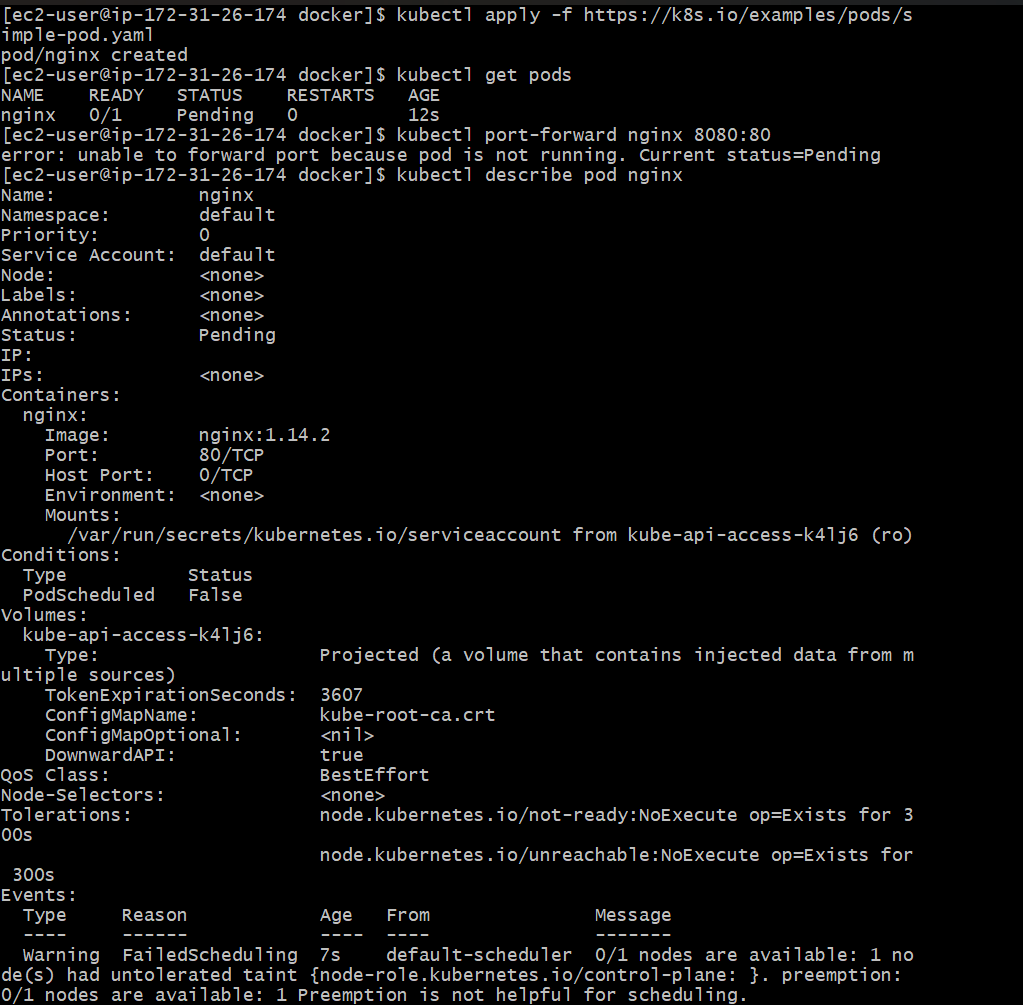


* Then, add a common networking plugin called flannel as mentioned in the code.

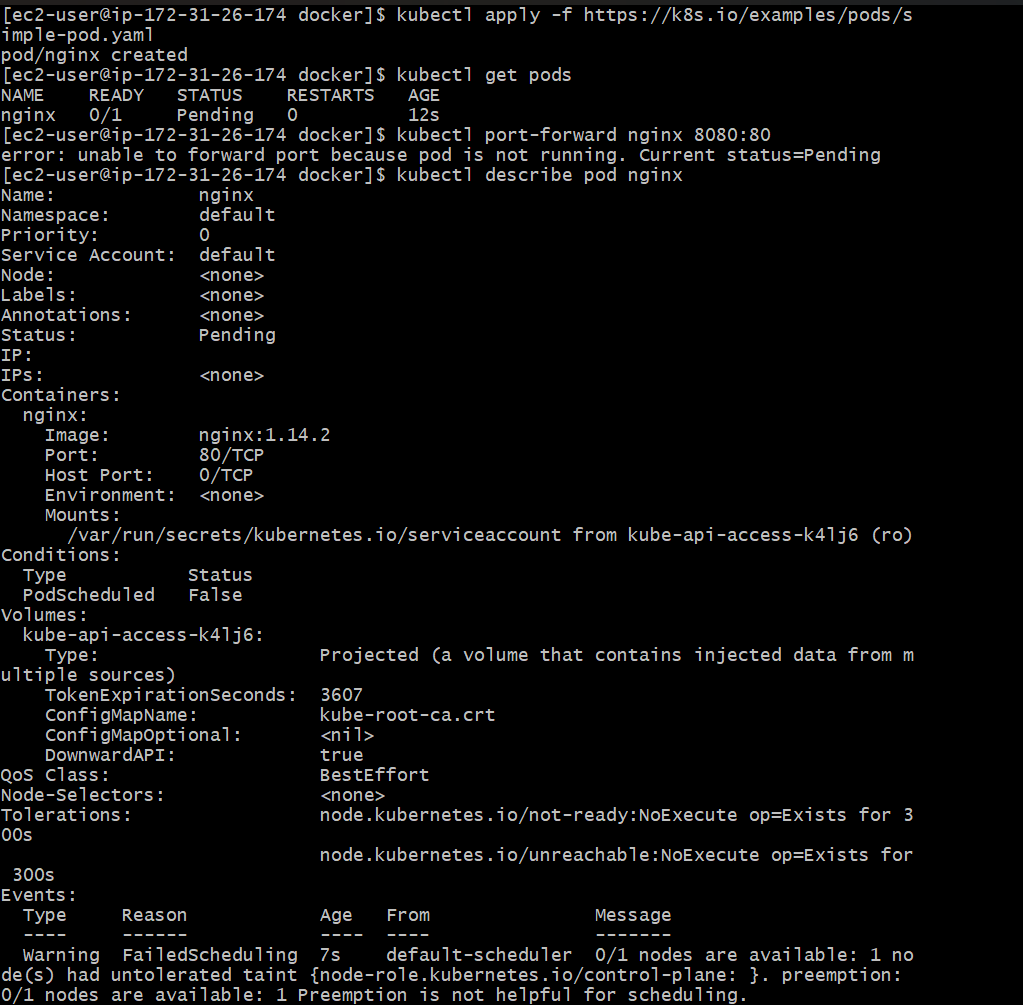
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

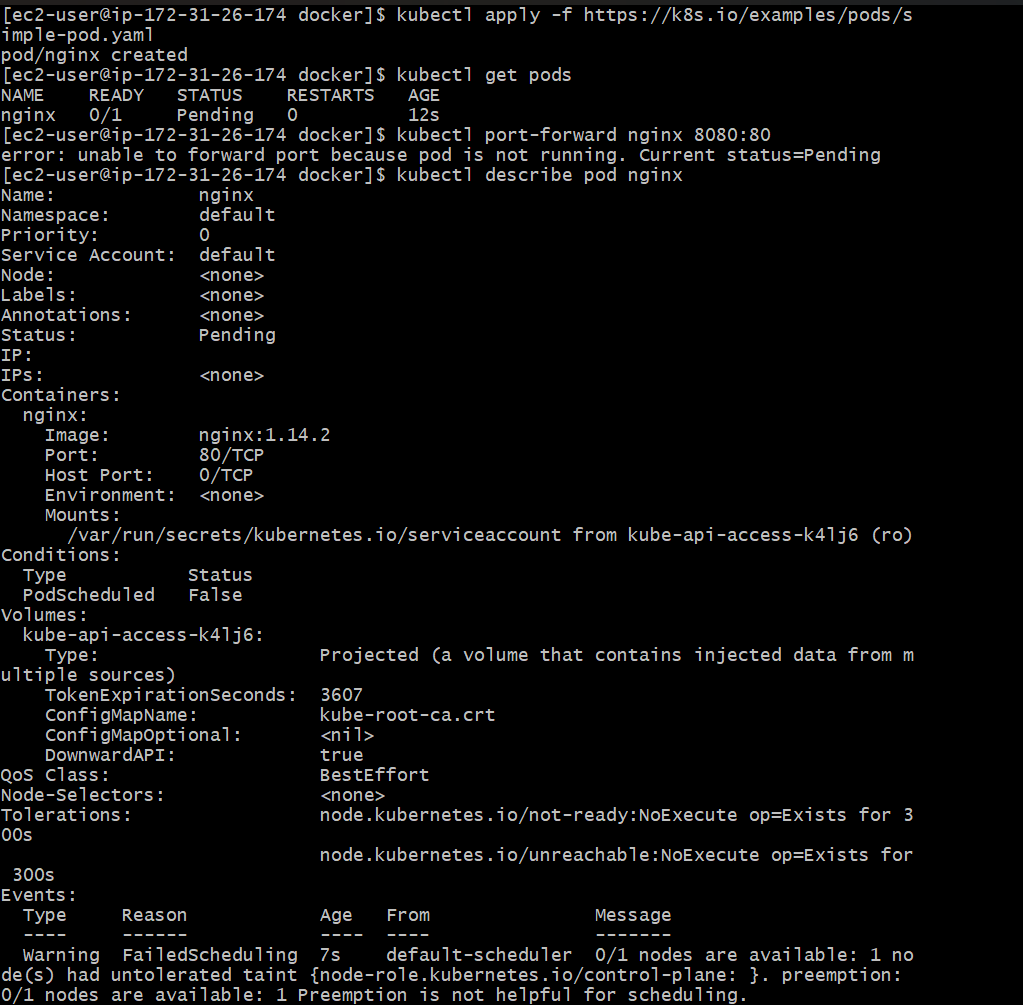


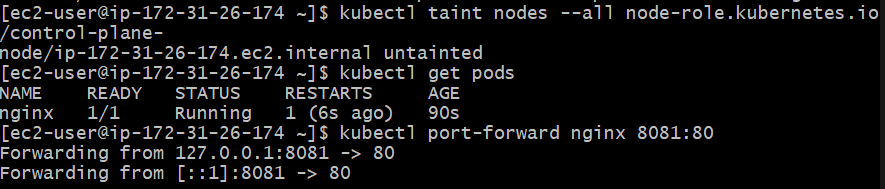
5. Now that the cluster is up and running,we can deploy our nginx server on this cluster. Apply deployment using this following command:

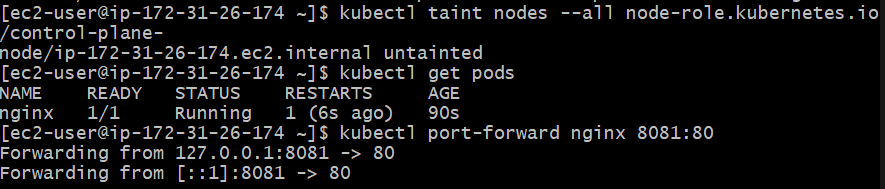
kubectl apply -f https://k8s.io/examples/pods/simple-pod.yamlThen use kubectl get nodes to check whether the pod gets created or not.

To convert state from pending to running use following command:

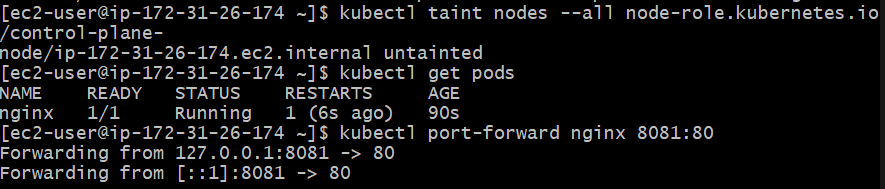
kubectl describe pod nginx This command will help to describe the pods it gives reason for failure as it shows the untolerated taints which need to be untainted.



6. Now check pod status is is running

7.Lastly, mention the port you want to host. Here i have used localhost 8081 then check it.

kubectl port-forward nginx 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.

**Conclusion:**Firstly I created an EC2 AWS Linux instance successfully.then installed docker and kubernetes successfully.then initialized kubernetes which given me token and chown and mkdir command. Then I execute mkdir and chown the command successfully. Then I installed a networking plugin called flannel successfully. Then I tried to deploy nginx which initially gave an error. Then I deployed (simple-pod.yml ) nginx successfully and also checked by using the get pods command.then hosted it on localhost 8081 ie http://localhost:8081 successfully.