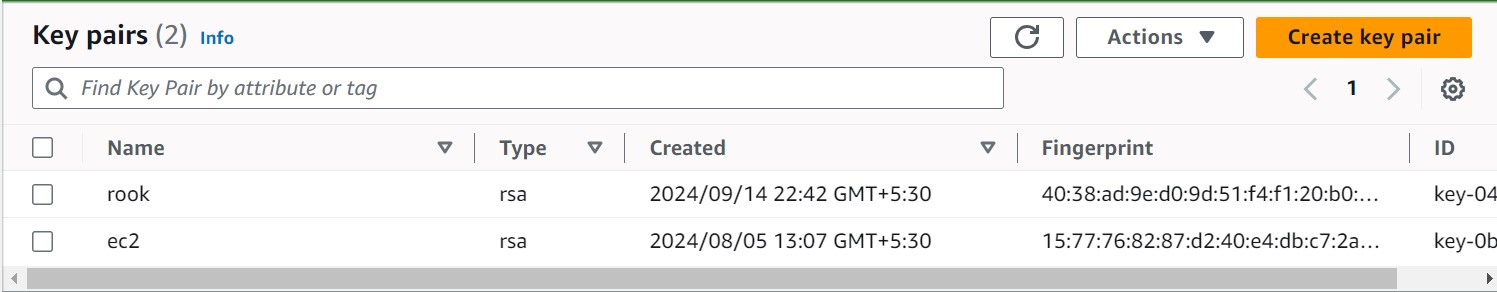
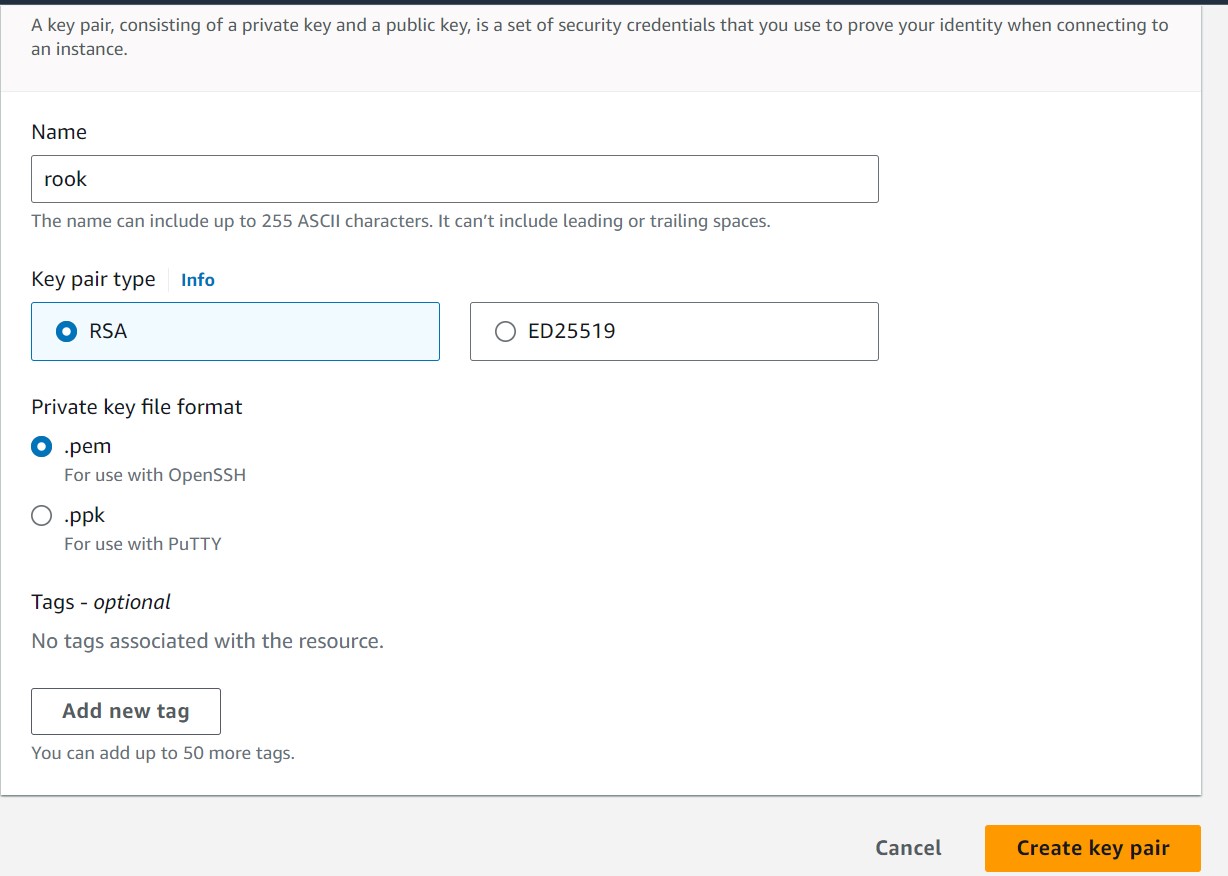
**Aim:**

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

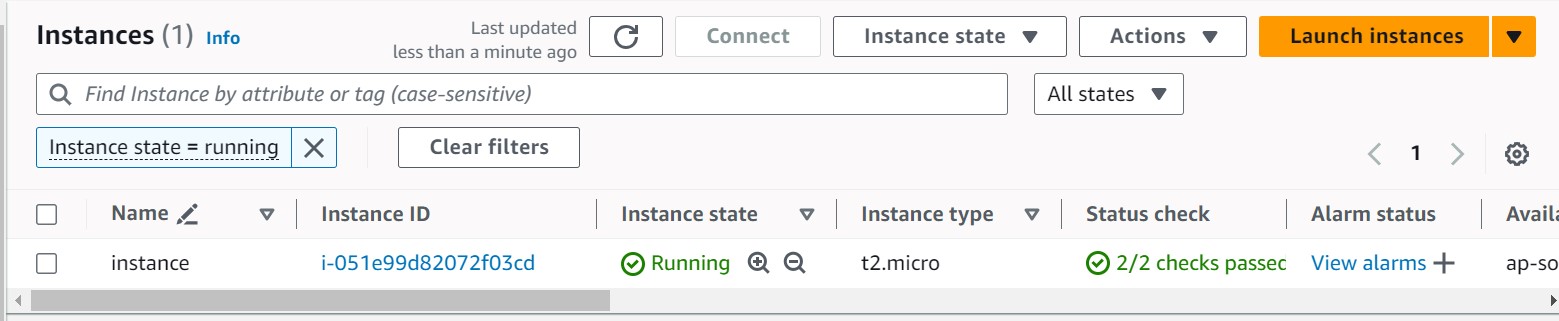
Steps:

1. Create a key pair.

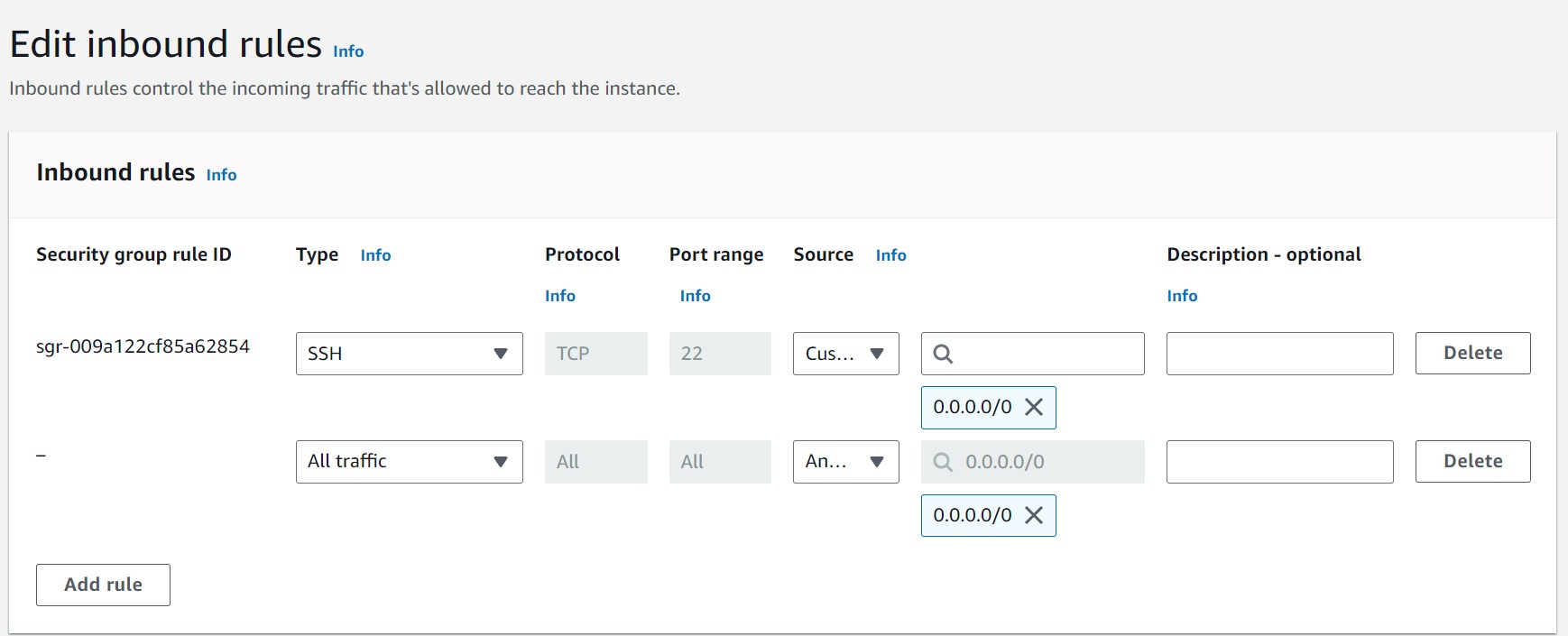


The .pem file will be downloaded on your machine and will be required in the further steps.

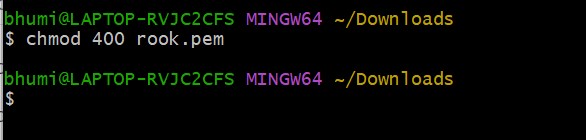
1. Now we will create an EC2 Ubuntu instance. Select the key pair which you just created while creating this instance.



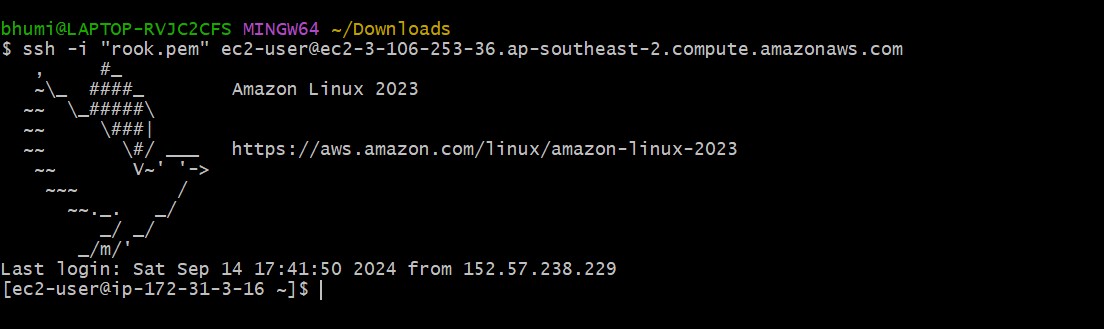
1. Now edit the inbound rules to allow ssh.



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

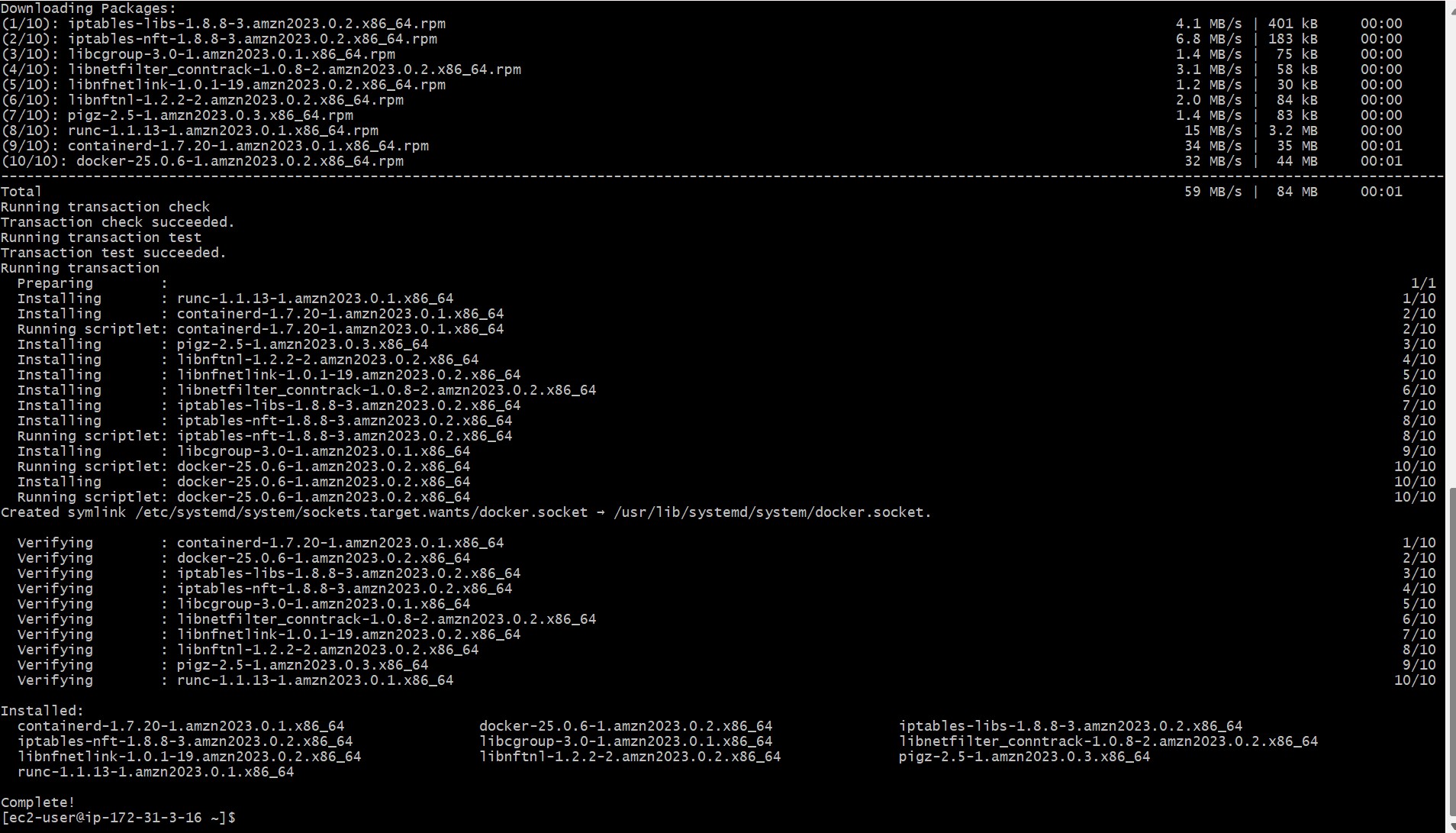
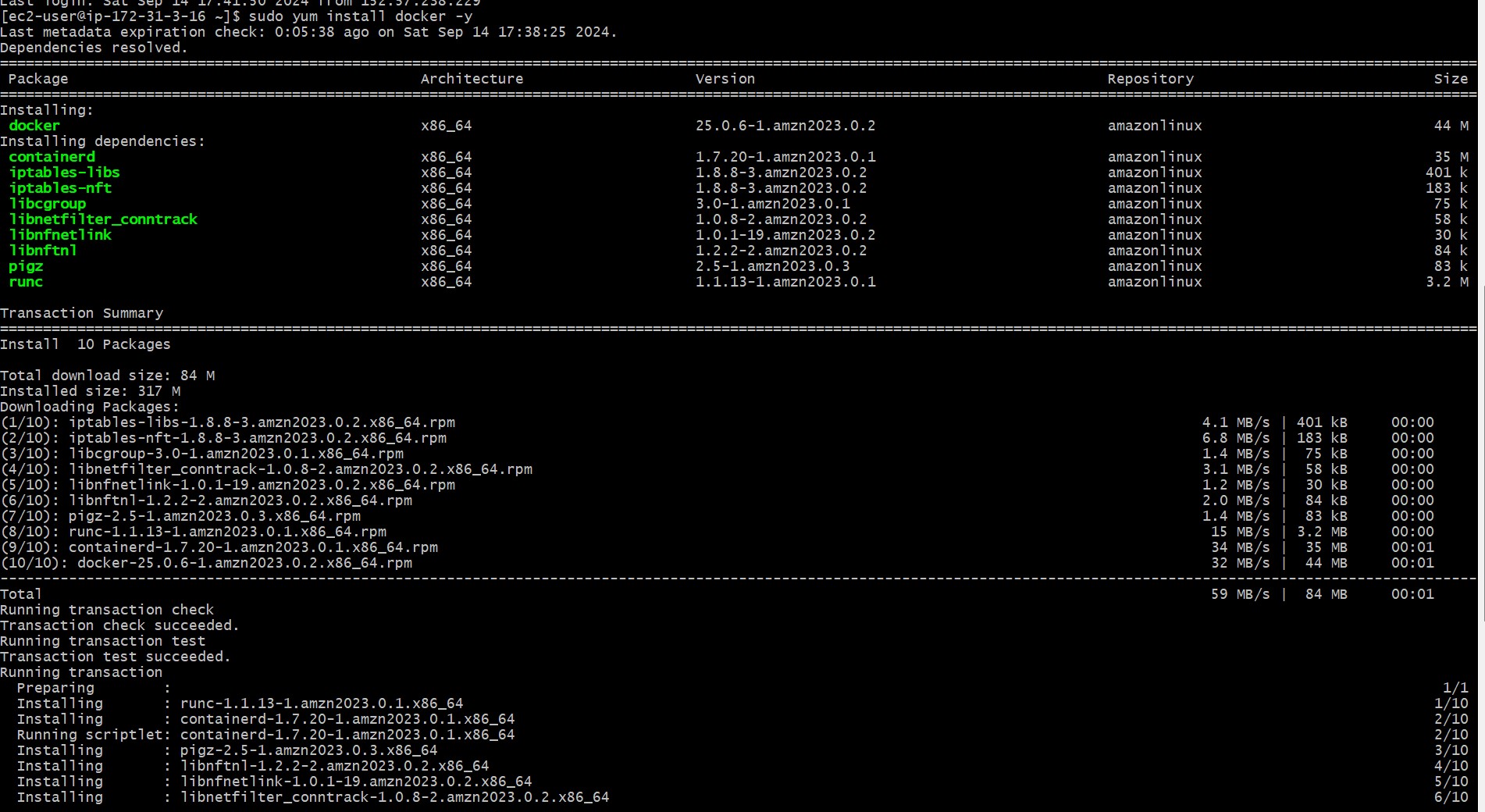


1. Now use this command on the terminal: ssh -i <keyname>.pem ubuntu@ and replace
   * Keyname with the name of your key pair, in our case test1.
   * As we are using amazon Linux instead of ubuntu we will have ec2-user • Replace public ip address with its value. Go to your instance and scroll down and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



1. Docker installation:

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



1. Then to configure cgroup in a daemon json file we will run cd /etc/docker cat <<EOF | sudo tee /etc/docker/daemon.json

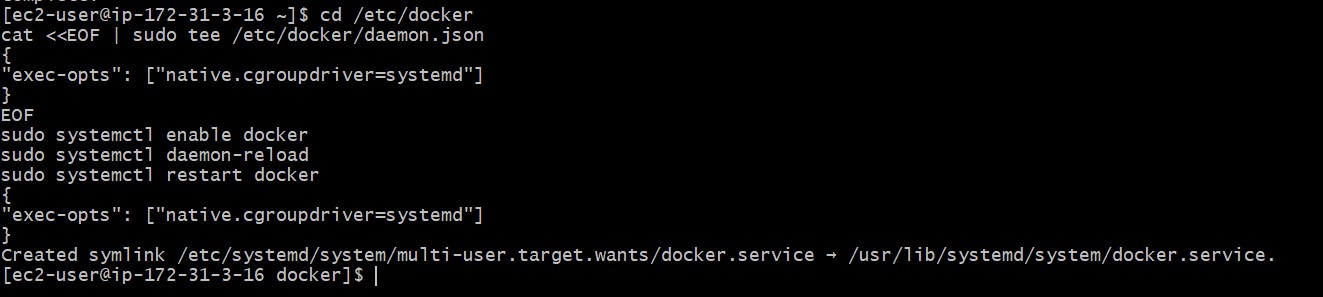
{

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

}

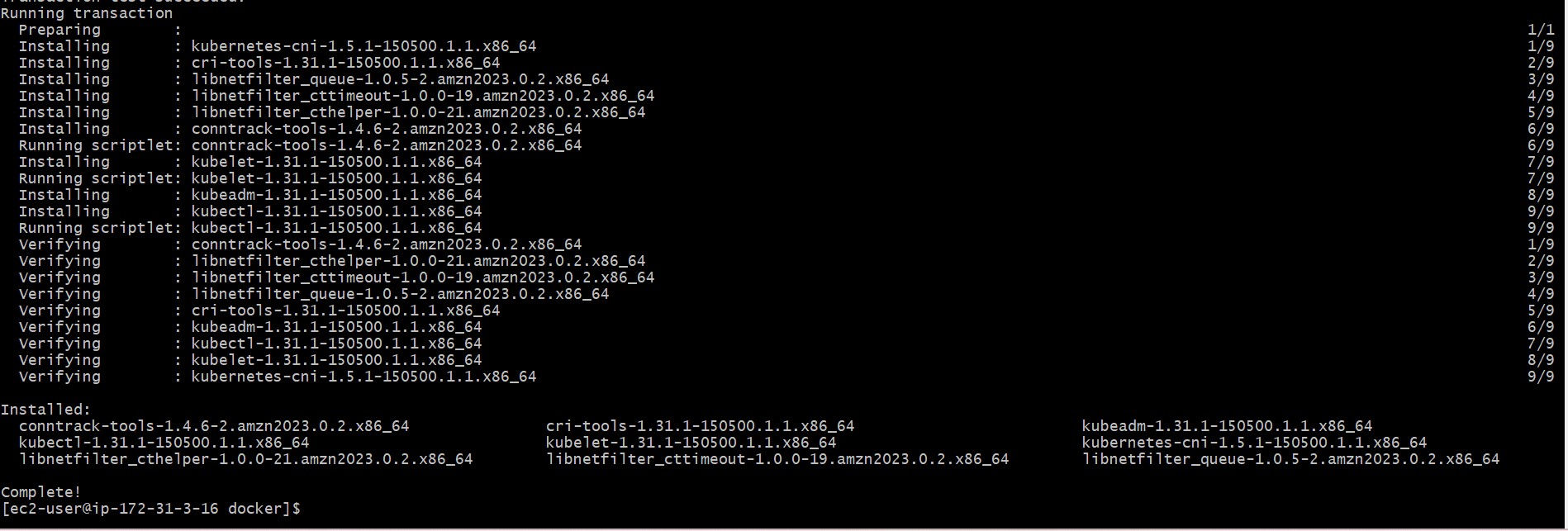
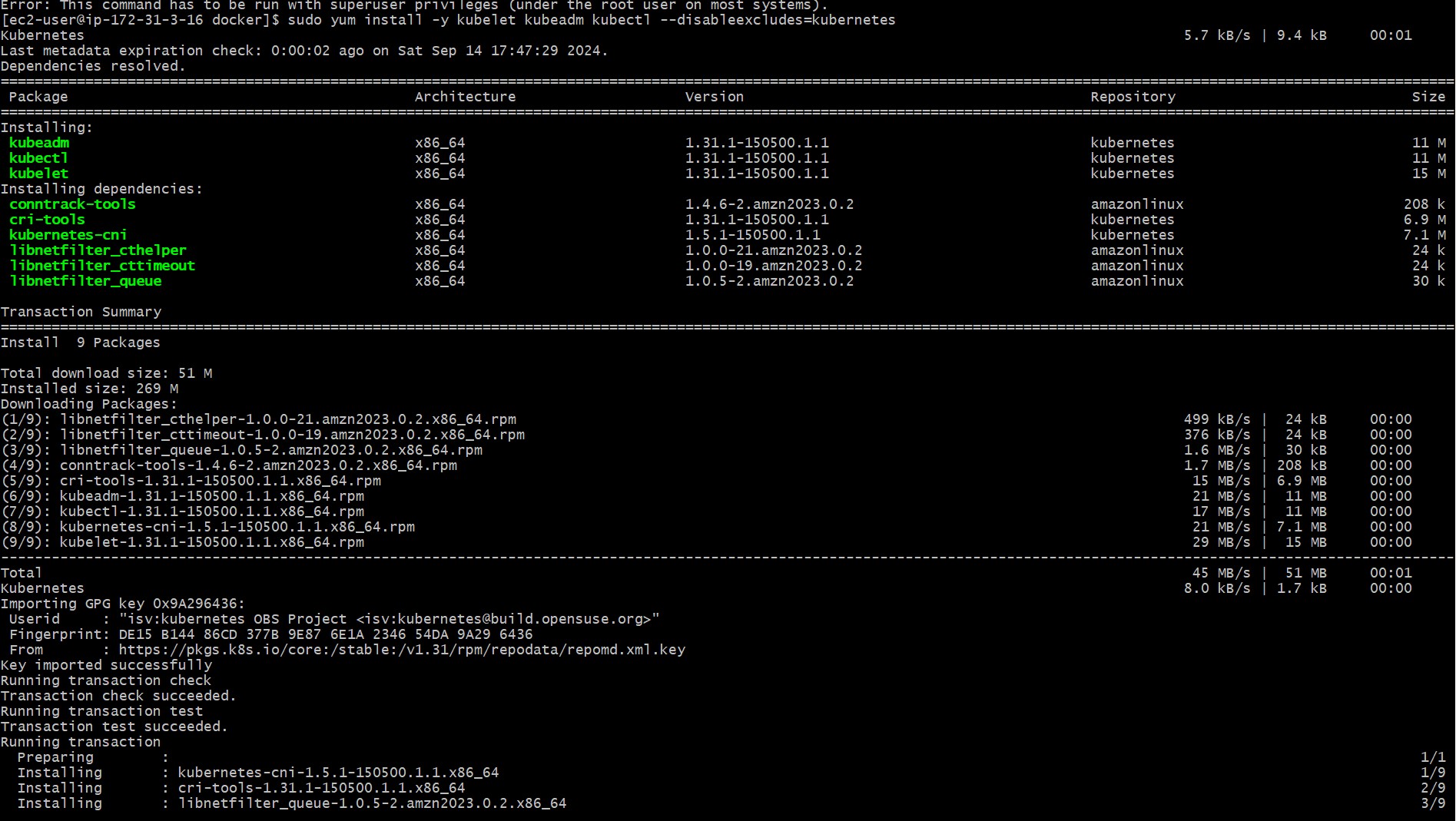
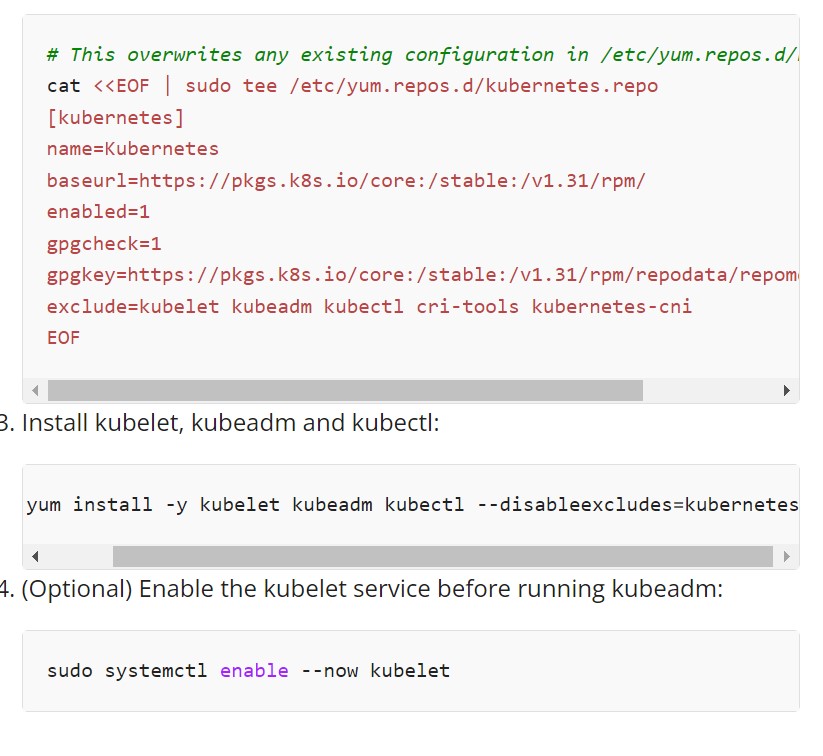
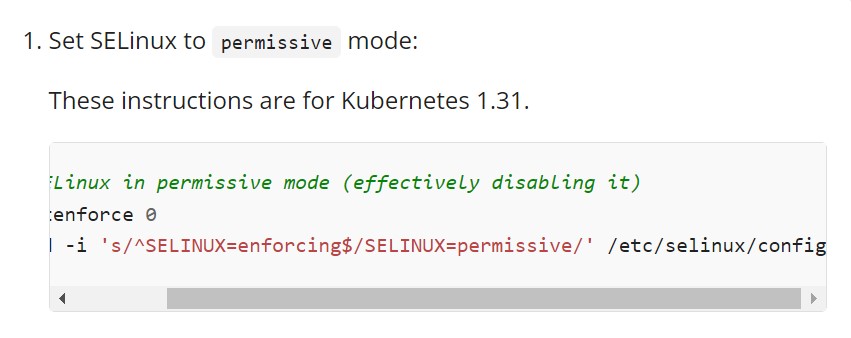
EOF

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



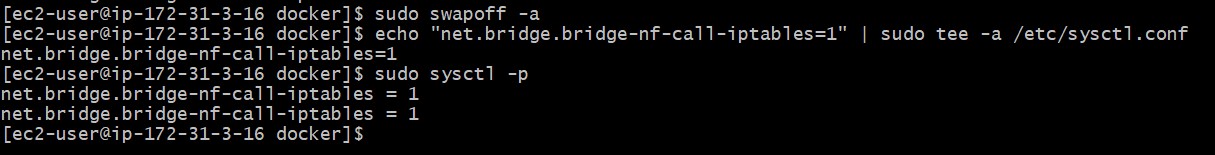
1. Kubernetes installation:

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



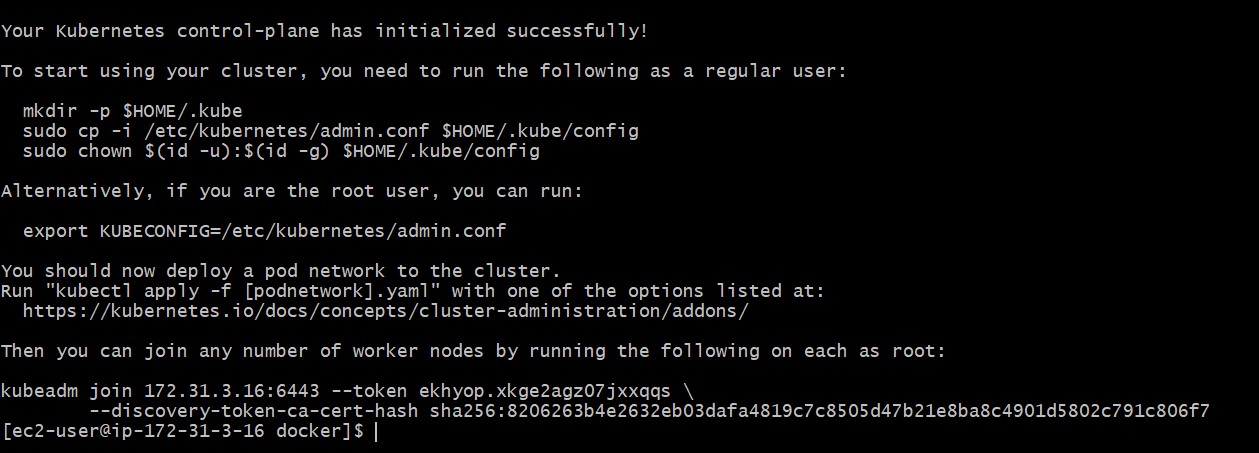
1. After installing Kubernetes, we need to configure internet options to allow bridging.

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

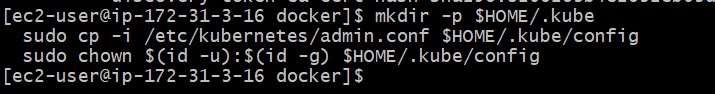


10.Initializing kubecluster:

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

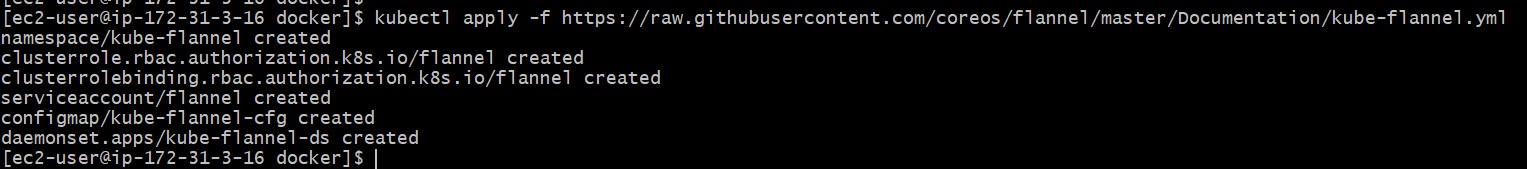


11. The mkdir command that is generated after initialization has to be copy pasted in the terminal.

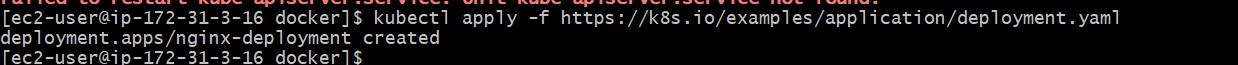


12.Then, add a common networking plugin called flannel:

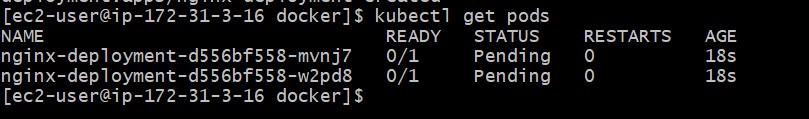
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/ kube-flannel.yml



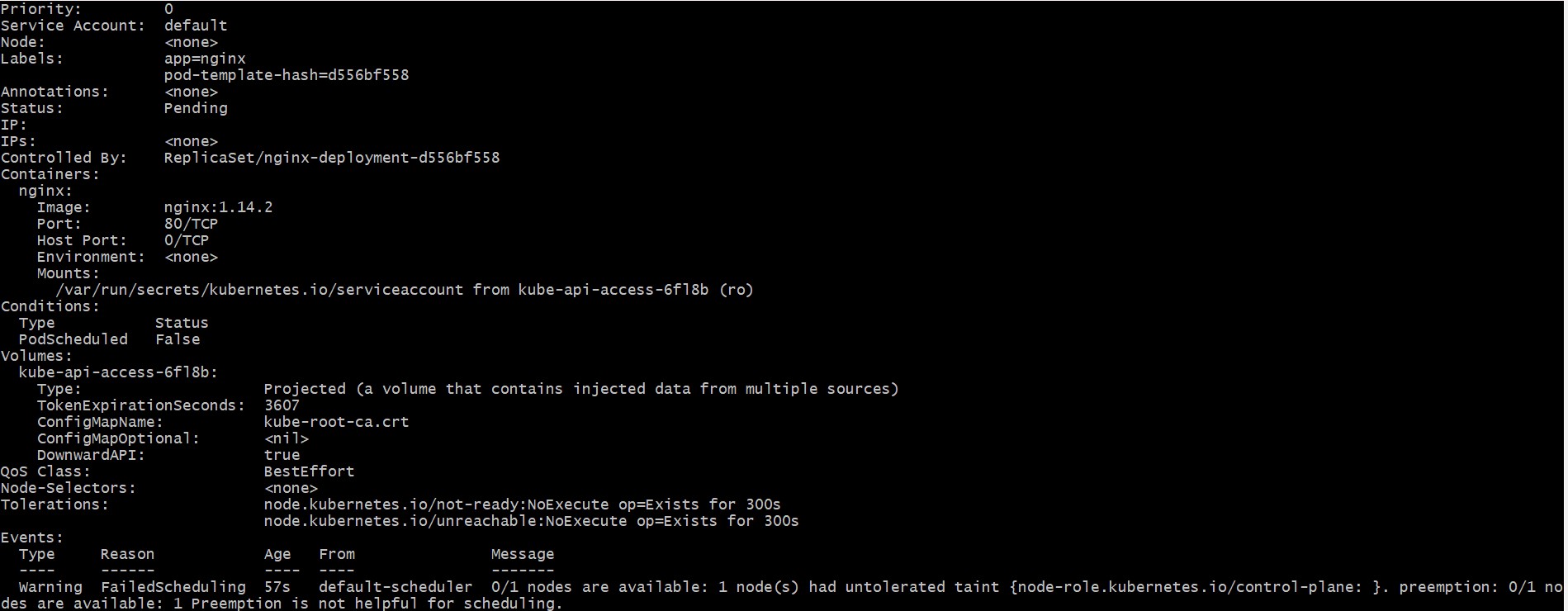
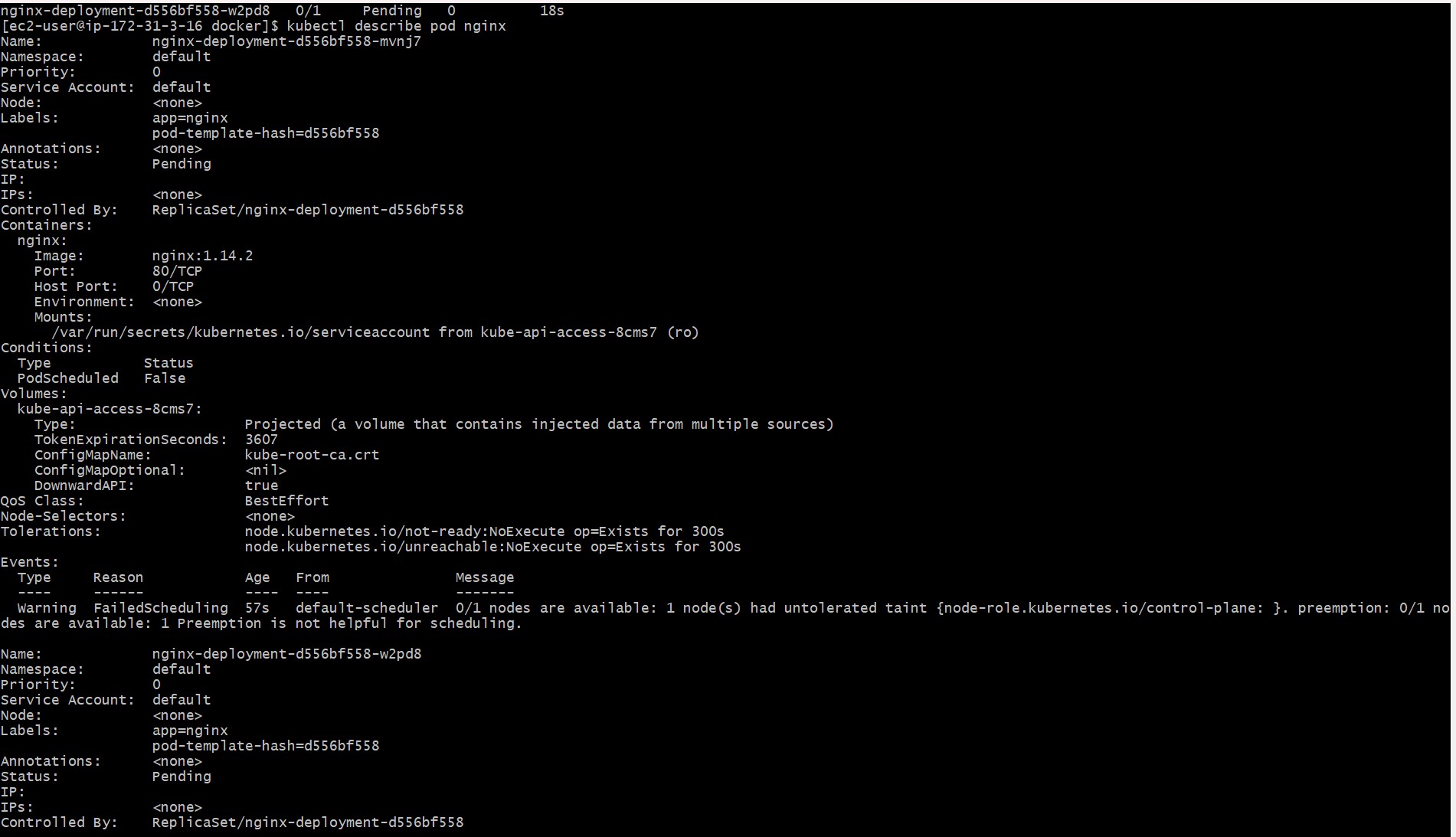
13.Now that the cluster is up and running, we can deploy our nginx server on this cluster. Apply this deployment file using this command to create a deployment kubectl apply -f https://k8s.io/examples/application/deployment.yaml



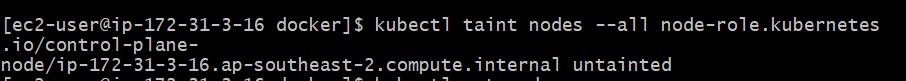
14.Use kubectl get pods to check if the pod is working correctly.



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



Use the below command to remove taints.



16.Check the pod status.



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



1. 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](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.