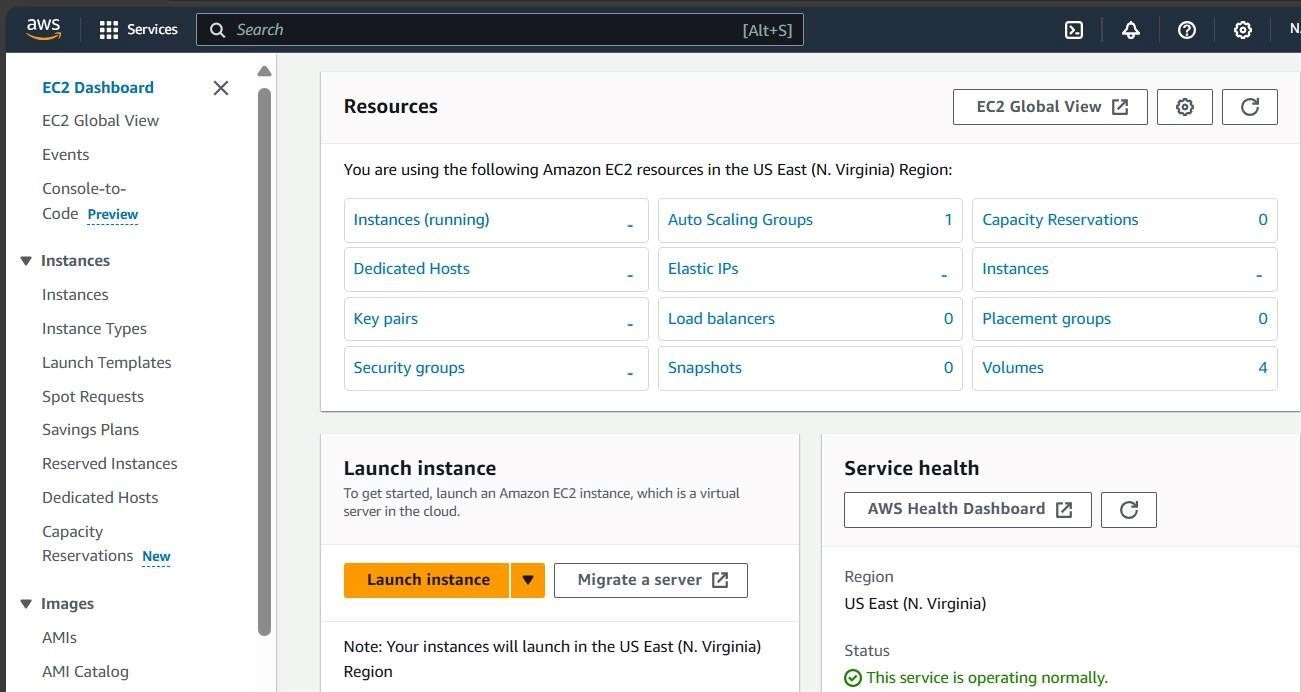
**EXPERIMENT NO. 3**

**Aim**: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud.

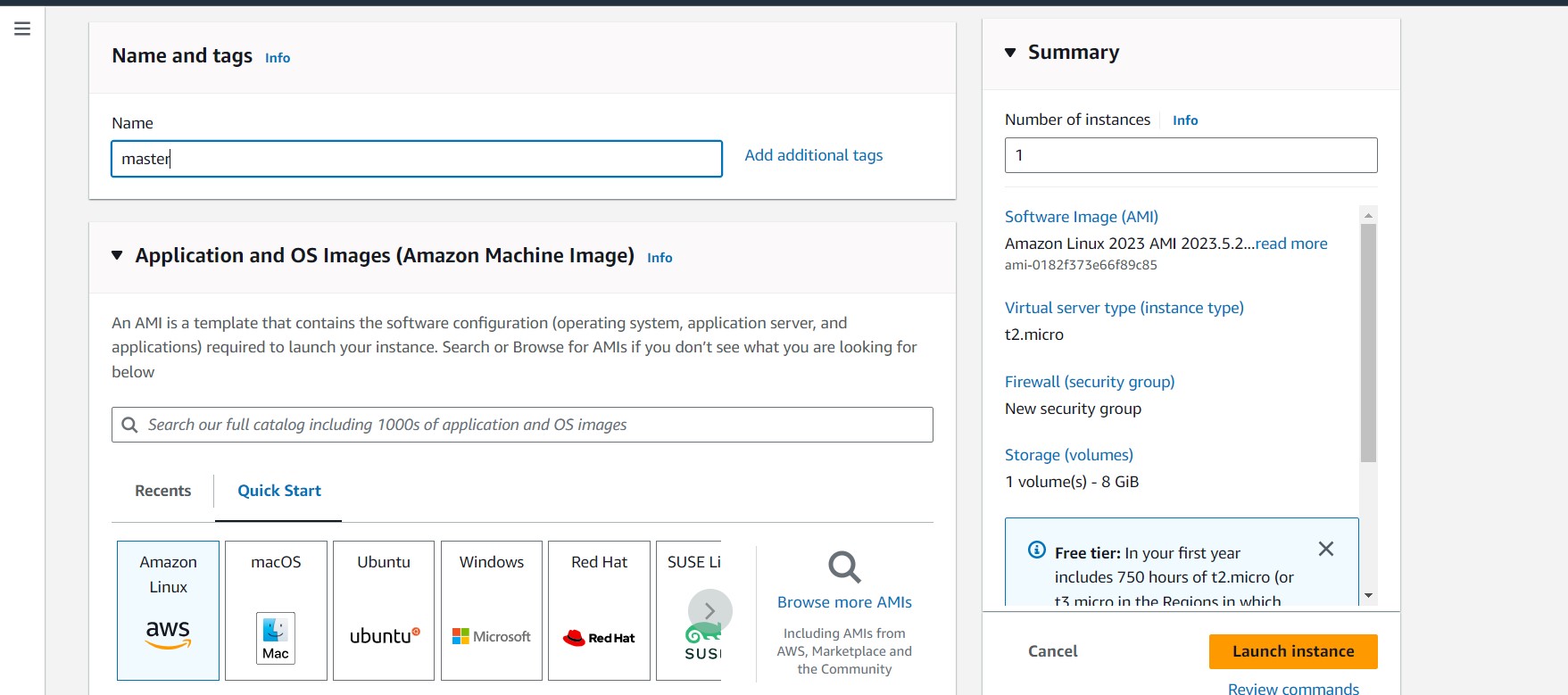
**Procedure:**

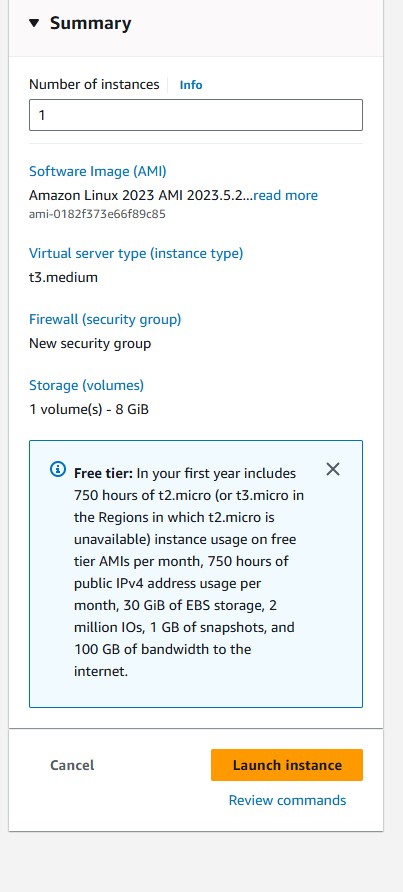
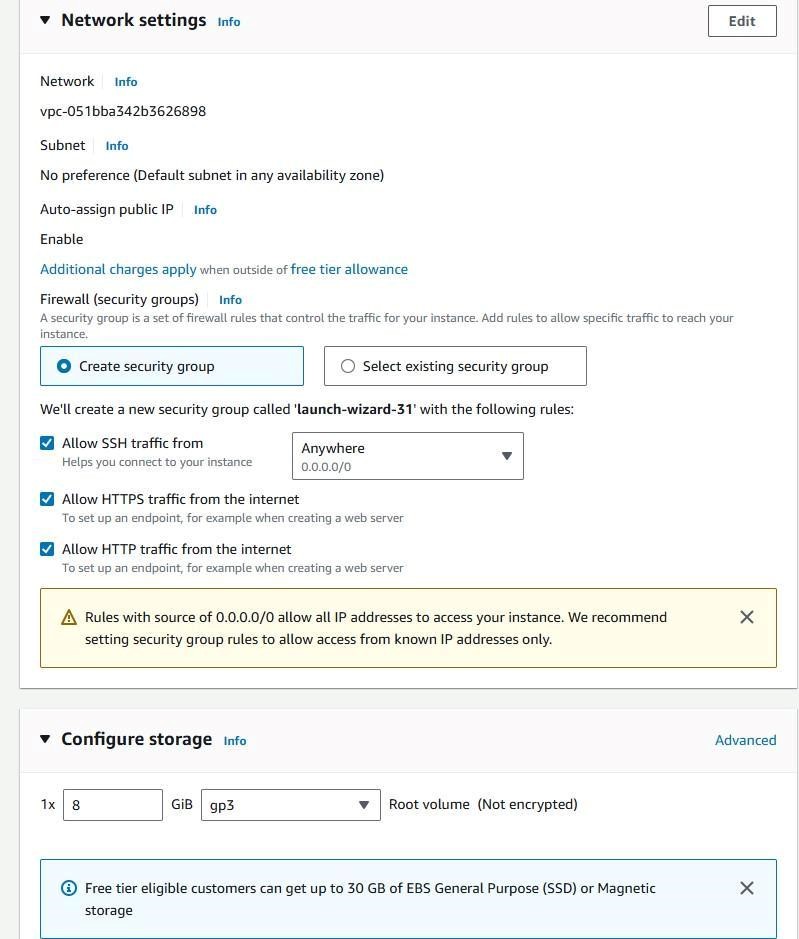
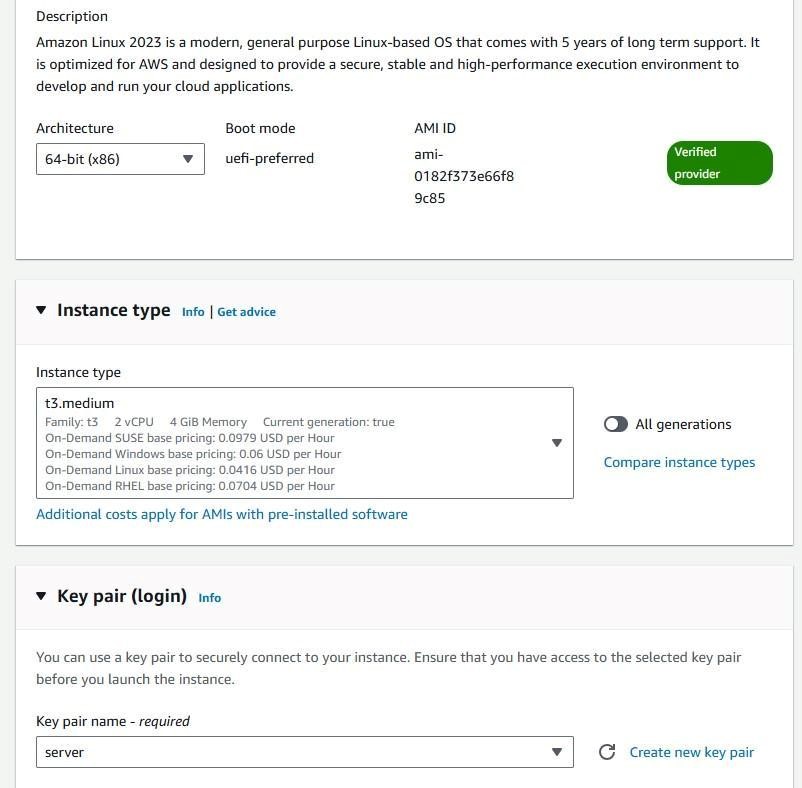
**1. Creation Of Instance**

●

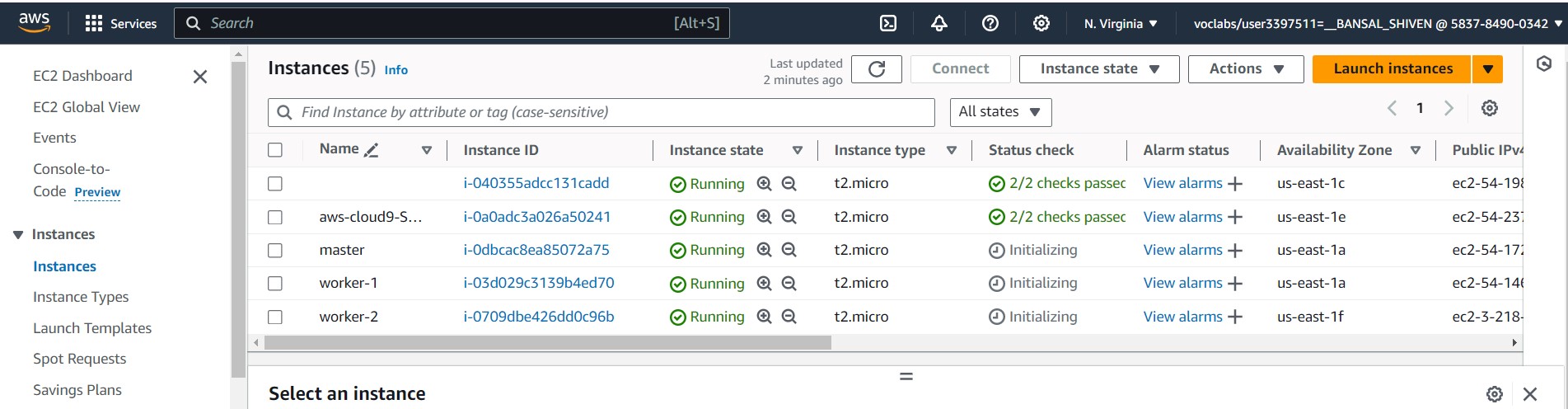


Search EC-2 instance. Then create three EC-2 instances and choose Amazon Linux as OS and also allow ssh traffic from anywhere.





To efficiently run kubernetes cluster select instance type of at least t3.medium as kubernetes recommends at least 2 vCPU to run smoothly on it.

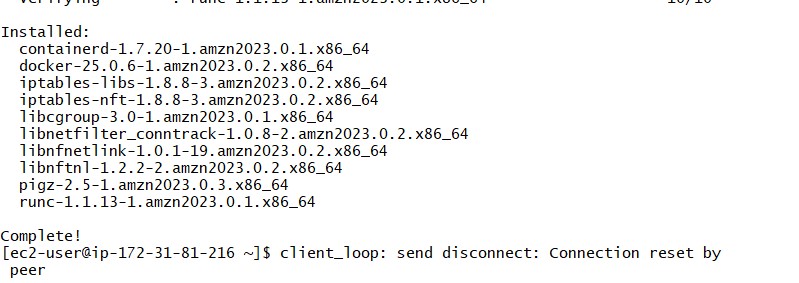
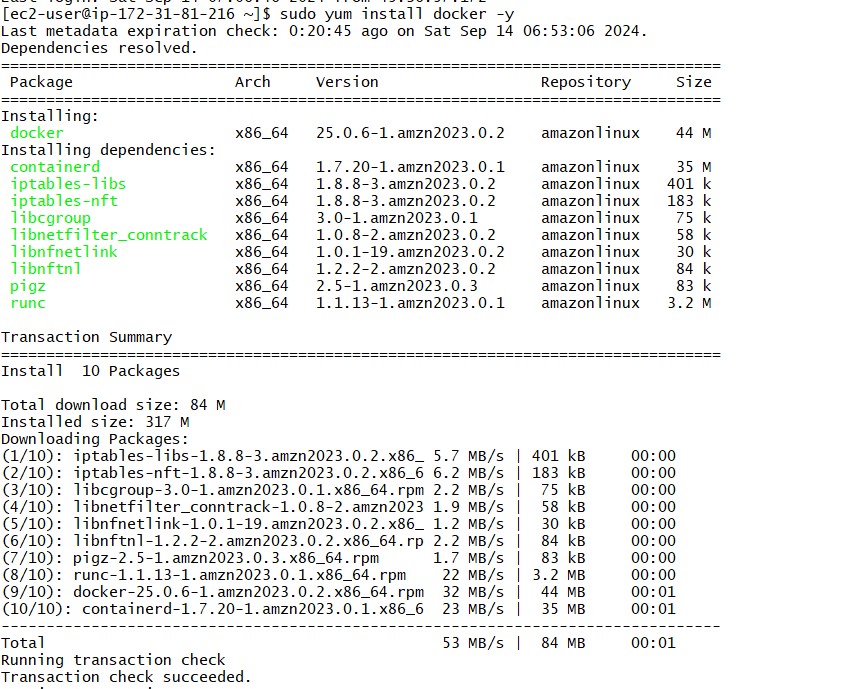


* Then for making connection through SSH into all 3 machines each in separate terminal Use this following command:

ssh -i <keyname>.pem ubuntu@<public\_ip\_address> where keyname is name of the key you created here i created key server.pem and use public IP address.(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).



* 1. **Installation Of Docker on three machines**
* For installation of Docker into all three 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"

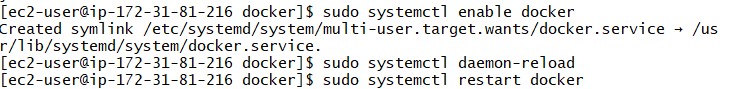
}

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



* Then check the version of docker installed. docker -v



* 1. **Installation Of Kubernetes on three machines**
* 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=Kube

rnetes

baseurl=h

tt

ps://pkgs.k8s.io/core:/stable:/v1.30/rpm/

enabled=1

gpgcheck=1

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

ta/repomd.xml.key

excl

ude=kubelet

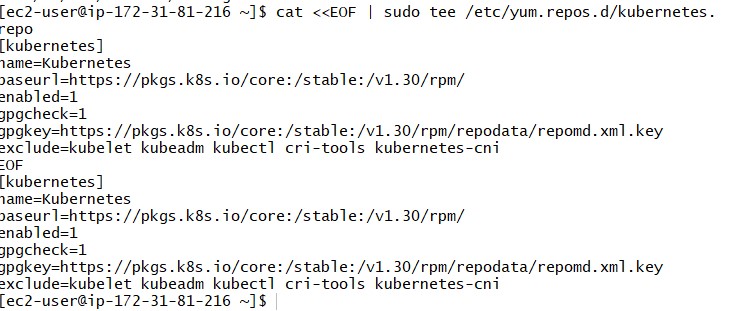
kubeadm

kubectl

cri-tools

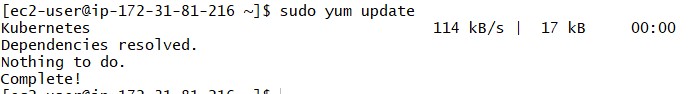
kubernetes-cni

EOF

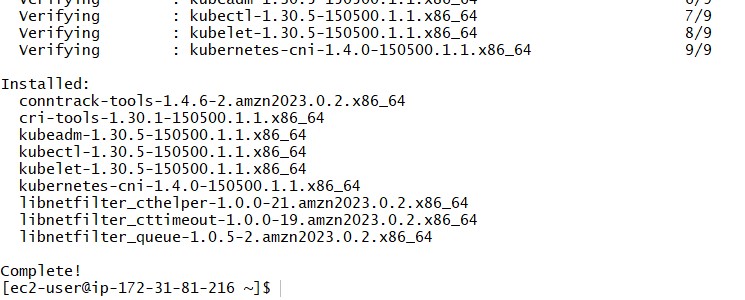
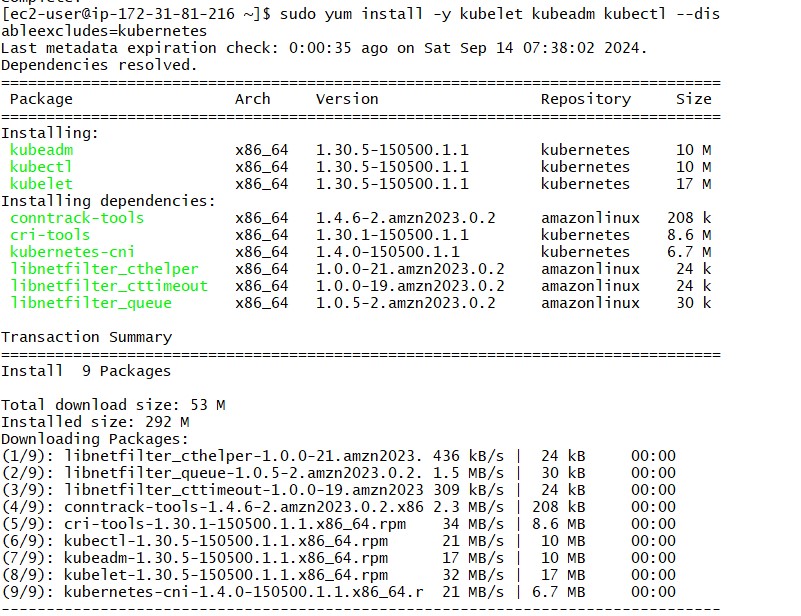


* After that Run following command to make the updation and also to install kubelet

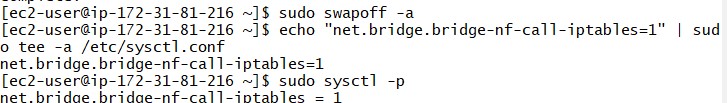
,kubeadm, kubectl: sudo yum update



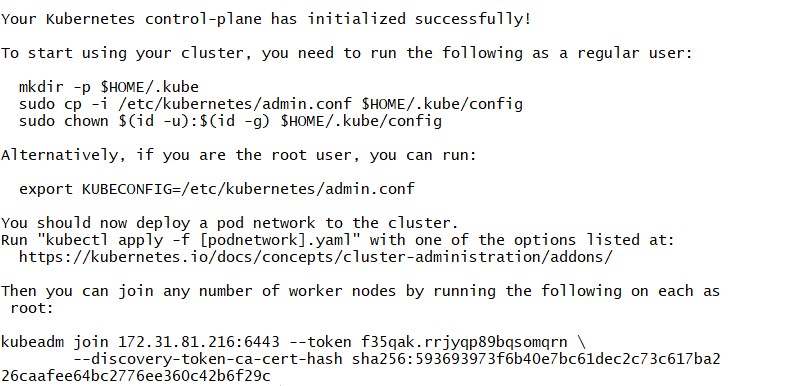
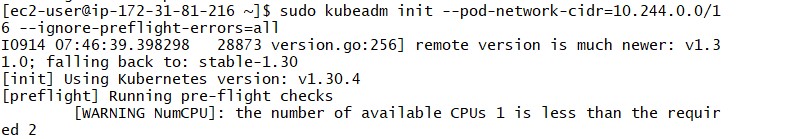
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes



* 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



* 1. **Perform this ONLY on the Master machine**
* Initialize kubernetes by typing below command sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all



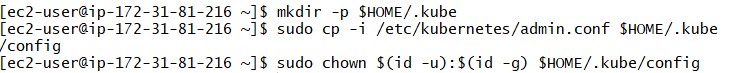
* So after initialization you will get token at the end for joining master and worker. Like here I got this :(save this token as it is required later.Then you can join any number of worker nodes by running the following on each as root.)

kubeadm join 172.31.81.216:6443 --token f35qak.rrjyqp89bqsomqrn \

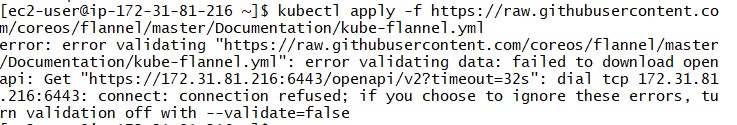
--discovery-token-ca-cert-hash sha256:593693973f6b40e7bc61dec2c73c617ba226caafee64bc2776ee360c42b6f29c

* Also,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 flammel file as mentioned in the code. kubectl apply -f [https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.ym l](https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml)



This step gives an error

**Conclusion:**

In this experiment, we successfully set up a Kubernetes cluster across three Amazon Linux EC2 instances, each equipped with Kubernetes components. The master node was initialized using kubeadm, and pod networking was configured with the Flannel network plugin. Worker nodes were integrated into the cluster through the join command generated during the master node's initialization. The process provided a comprehensive understanding of Kubernetes cluster setup on EC2. However, there was a noticeable delay in the worker nodes connecting to the master node, which may have been caused by network connectivity issues or configuration discrepancies.