**Name:** Abhay Gupta

**Roll No:** 18

**Class:** D15B

**Experiment 4**

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

**Theory:**

**Kubernetes**, originally developed by Google, is an open-source container orchestration platform. It automates the deployment, scaling, and management of containerized applications, ensuring high availability and fault tolerance. Kubernetes is now the industry standard for container orchestration and is governed by the **Cloud Native Computing Foundation (CNCF)**, with contributions from major cloud and software providers like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

**Kubernetes Deployment:** Is a resource in Kubernetes that provides declarative updates for Pods and ReplicaSets. With a Deployment, you can define how many replicas of a pod should run, roll out new versions of an application, and roll back to previous versions if necessary. It ensures that the desired number of pod replicas are running at all times.

**Necessary Requirements:**

● **EC2 Instance:** The experiment required launching a t2.medium EC2 instance with 2 CPUs, as Kubernetes demands sufficient resources for effective functioning.

**● Minimum Requirements:**

○ **Instance Type:** t2.medium

○ **CPUs:** 2

○ **Memory:** Adequate for container orchestration.

This ensured that the Kubernetes cluster had the necessary resources to function smoothly.

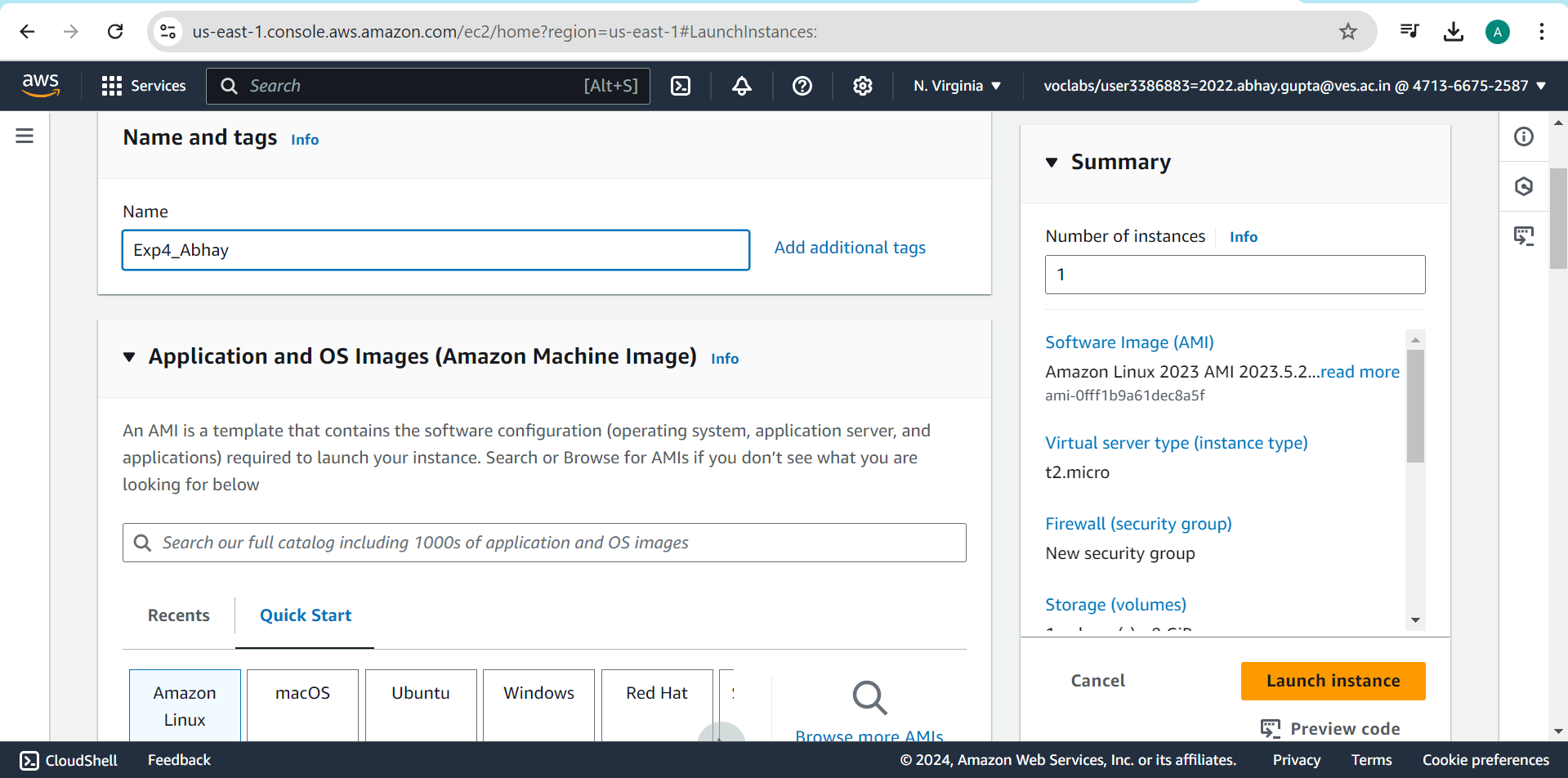
Note:

AWS Personal Account is preferred but we can also perform it on AWS Academy(adding some ignores in the command if any error occurs in below as the below experiment is performed on Personal Account .).

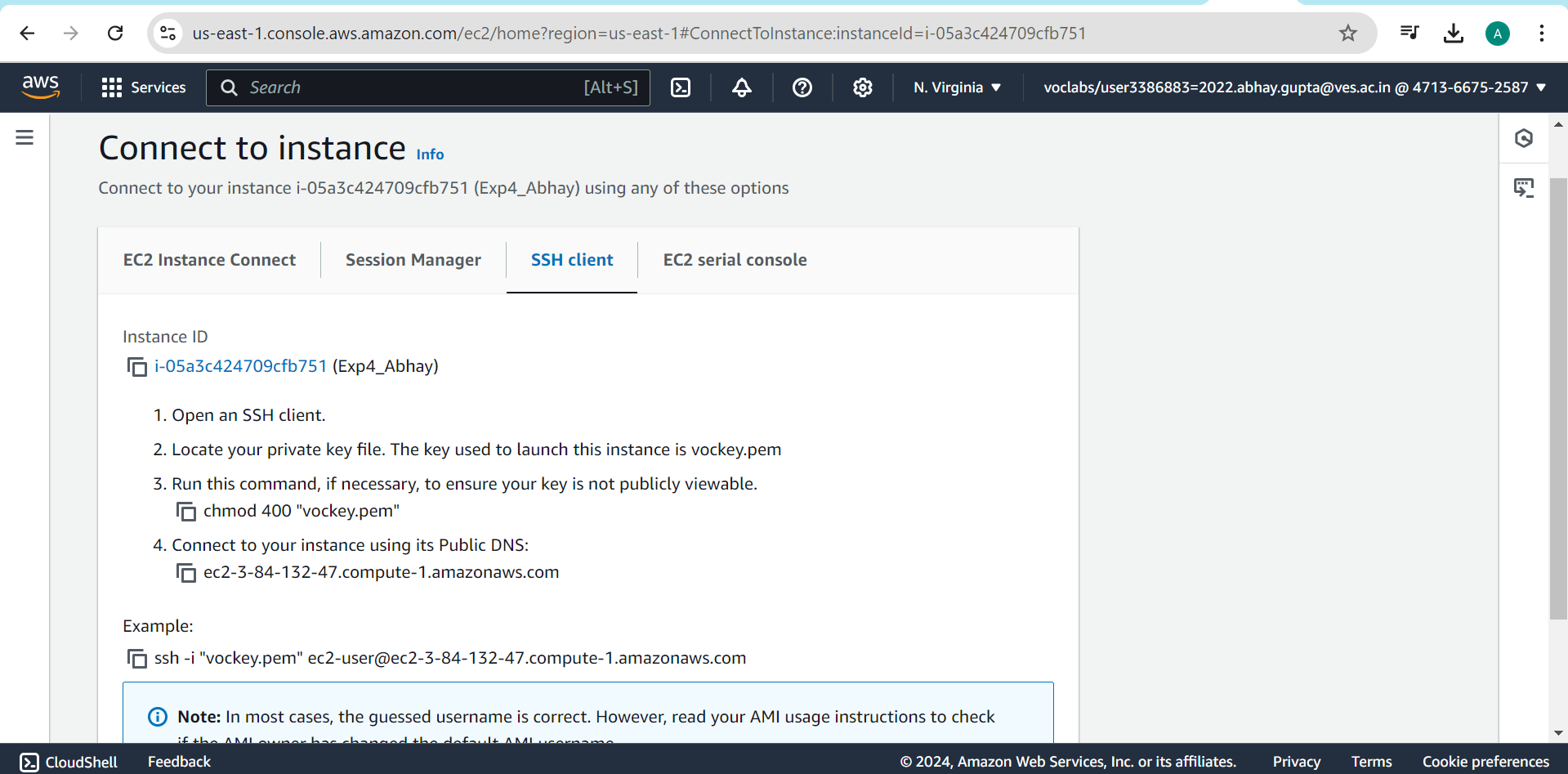
If You are using AWS Academy Account Errors you will face in kubeadm init command so you have to add some ignores with this command.

**Step 1:** Log in to your AWS Academy/personal account and launch a new Ec2 Instance. Select Ubuntu as AMI and t2.medium as Instance Type, create a key of type RSA with .pem extension, and move the downloaded key to the new folder.

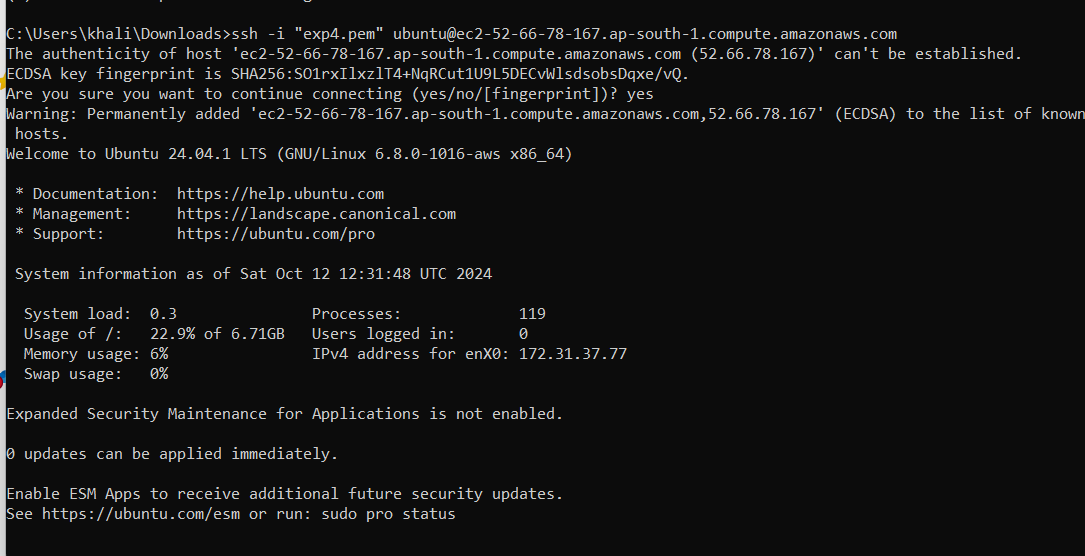
Note: A minimum of 2 CPUs are required so Please select t2.medium and do not forget to stop the instance after the experiment because it is not available in the free tier.



**Step 2:** After creating the instance click on Connect the instance and navigate to SSH Client



**Step 3:** Now open the folder in the terminal where our .pem key is stored and paste the Example command (starting with ssh -i …..) in the terminal.



**Step 4:** Run the below commands to install and setup Docker.

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee**

**/etc/apt/trusted.gpg.d/docker.gpg > /dev/null**

**sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"**

**sudo apt-get update**

**sudo apt-get install -y docker-ce**

**sudo mkdir -p /etc/docker**

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

**{**

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

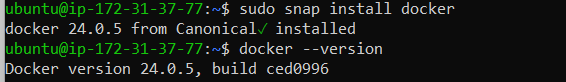
**}**

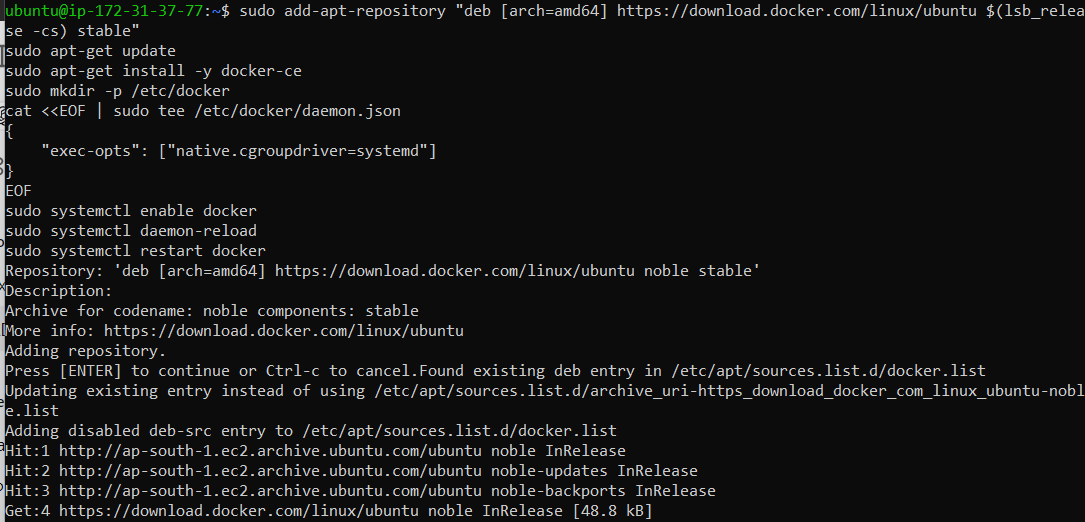
**EOF**

**sudo systemctl enable docker**

**sudo systemctl daemon-reload**

**sudo systemctl restart docker**

****

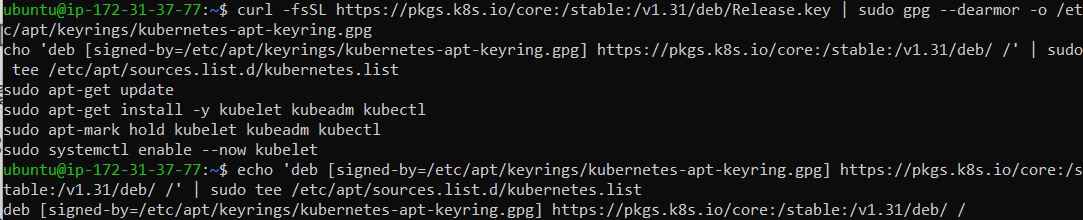
****

**Step 5:** Run the below command to install Kubernets.

**curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg**

**echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]**

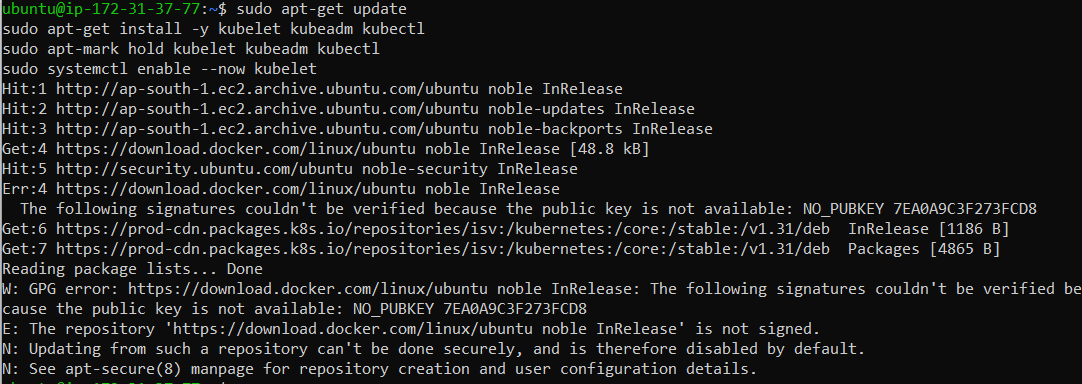
**https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list**

****

**sudo apt-get update**

**sudo apt-get install -y kubelet kubeadm kubectl**

**sudo apt-mark hold kubelet kubeadm kubectl**

****

**sudo systemctl enable --now kubelet**

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

****

**Now We have got an error.**

**So we have to perform some additional commands as follow.**

**sudo apt-get install -y containerd**

**sudo mkdir -p /etc/containerd**

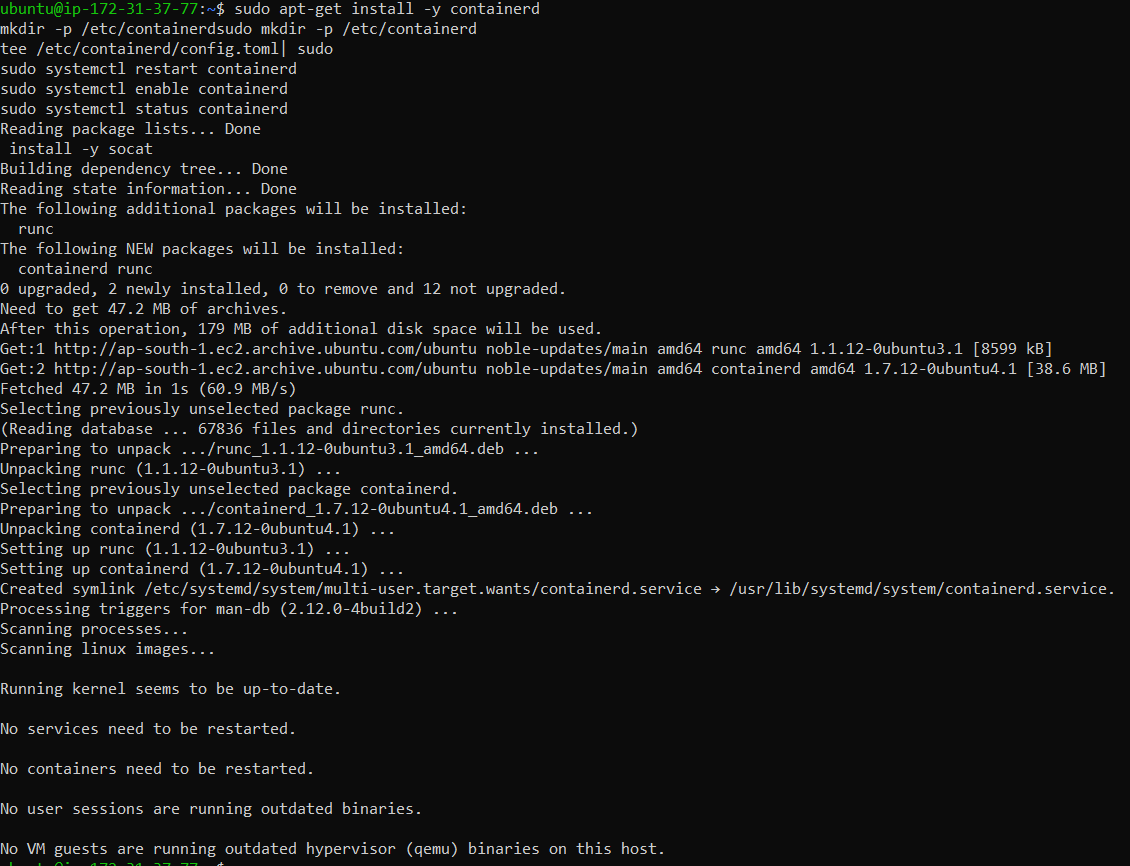
**sudo containerd config default | sudo tee /etc/containerd/config.toml**

**sudo systemctl restart containerd**

**sudo systemctl enable containerd**

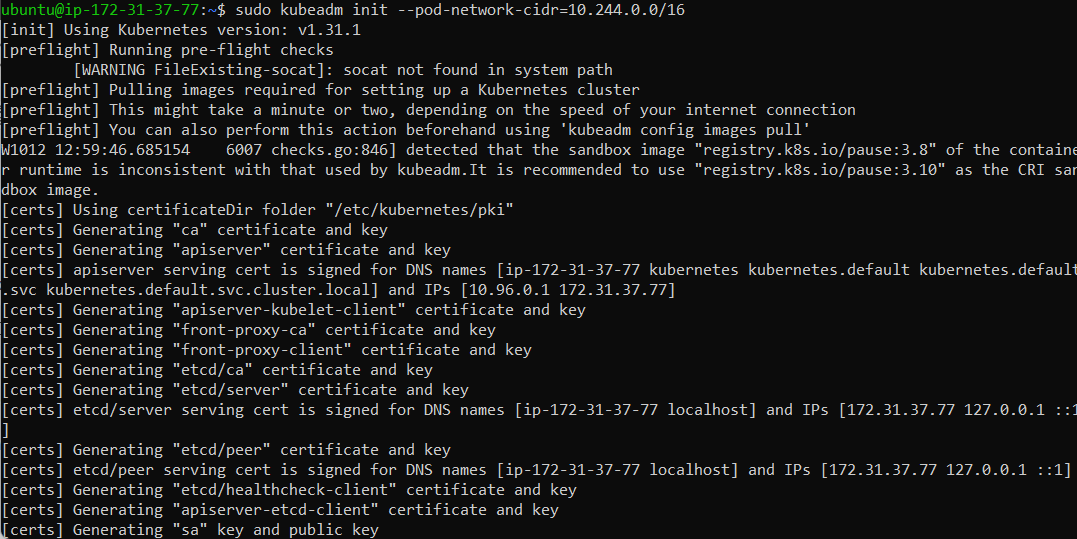
**sudo systemctl status containerd**

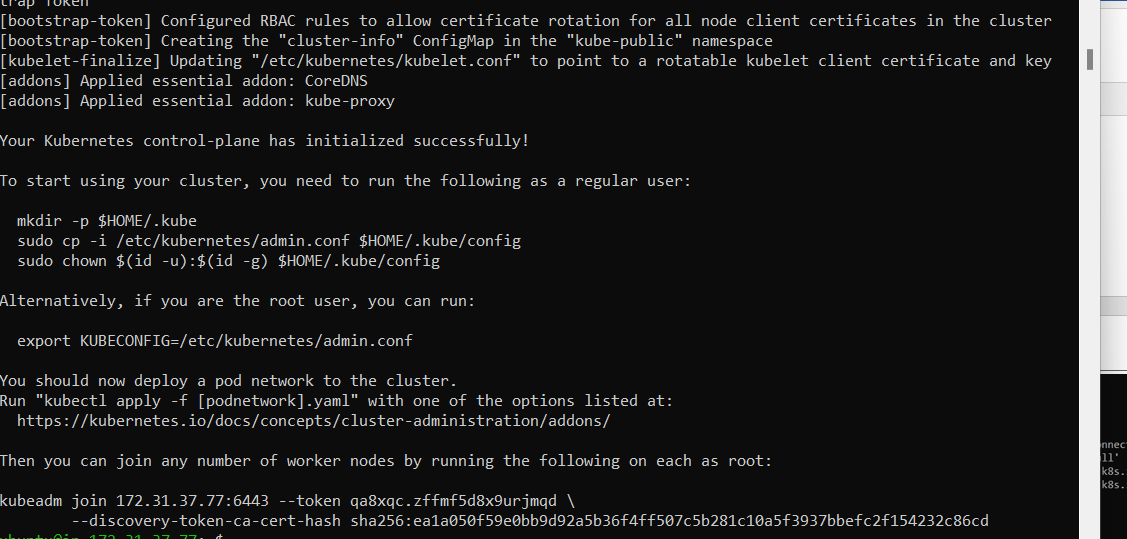
**sudo apt-get install -y socat**

****

**Step 6:** Initialize the Kubecluster

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

****

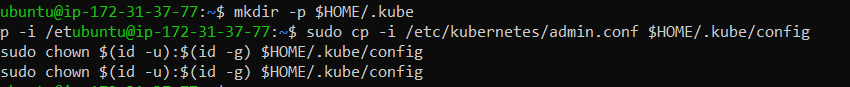
****

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

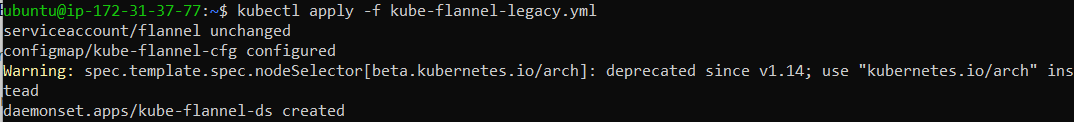
**mkdir -p $HOME/.kube**

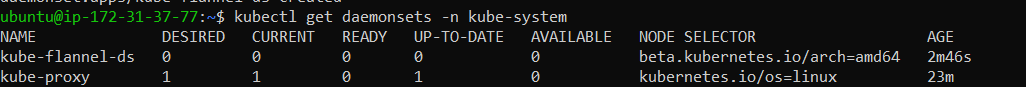
**sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config**

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

****

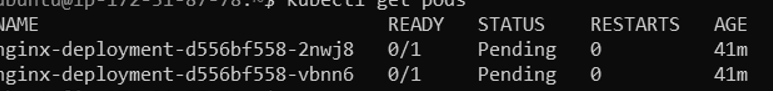
**Add a common networking plugin called flannel as mentioned in the code. kubectl apply -f**

**https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml **

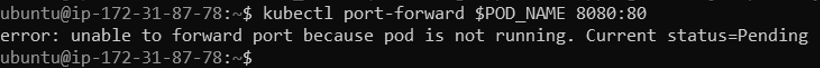
****

**Step 7: 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 get pods**

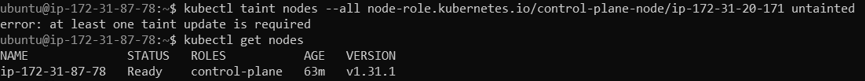
**POD\_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")**

**kubectl port-forward $POD\_NAME 8080:80**

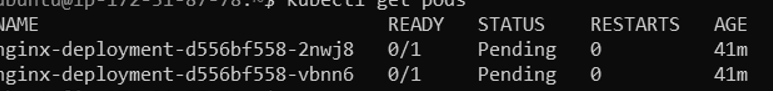


**Note : We have faced an error as pod status is pending so make it running run below commands then again run above 2 commands.**

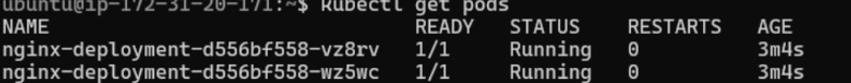
**kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted kubectl get nodes**



**kubectl get pods**

**POD\_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")**

**kubectl port-forward $POD\_NAME 8080:80**

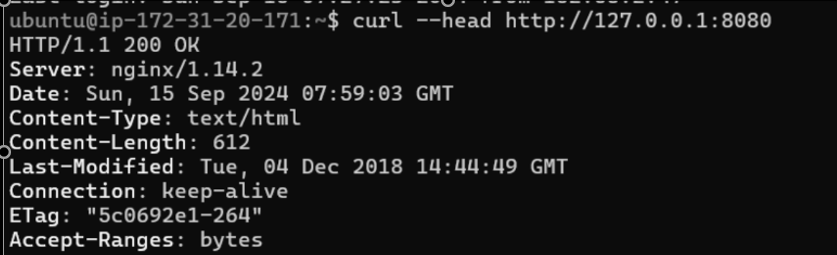


**Step 8:** Verify your deployment

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:**

In this experiment, we successfully installed Kubernetes on an EC2 instance and deployed an Nginx server using Kubectl commands. During the process, we encountered two main errors: the Kubernetes pod was initially in a pending state, which was resolved by removing the control-plane taint using kubectl taint nodes --all, and we also faced an issue with the missing containerd runtime, which was fixed by installing and starting containerd. We used a **t2.medium EC2 instance with 2 CPUs** to meet the necessary resource requirements for the Kubernetes setup and deployment.