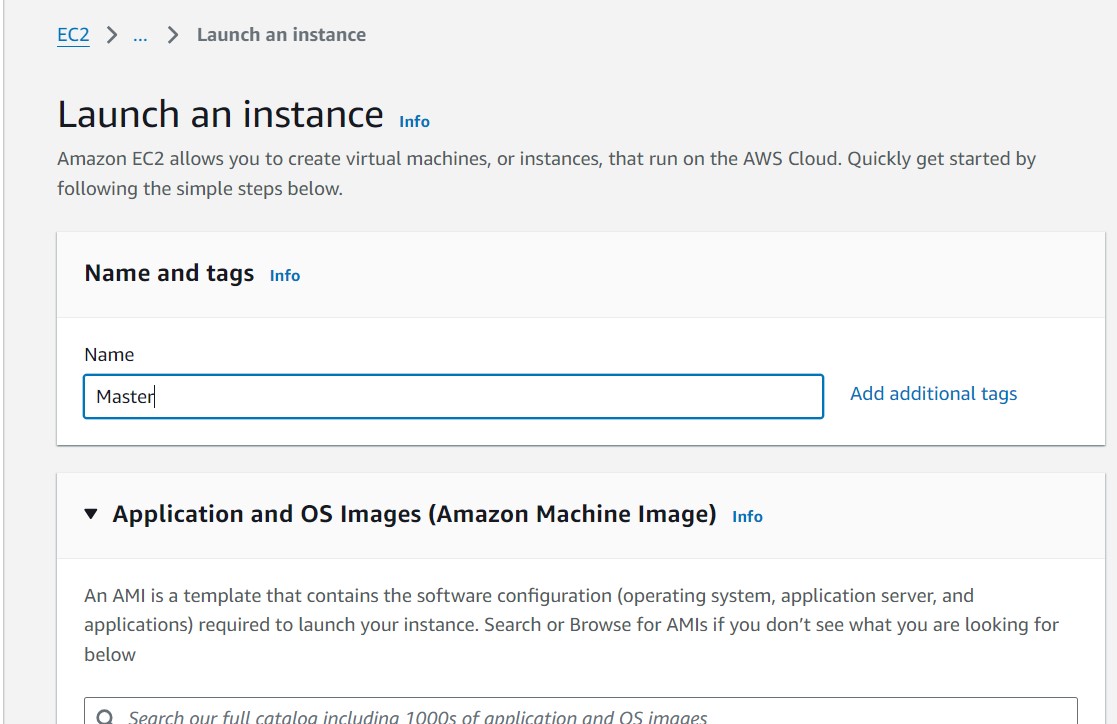
**CaseStudy**

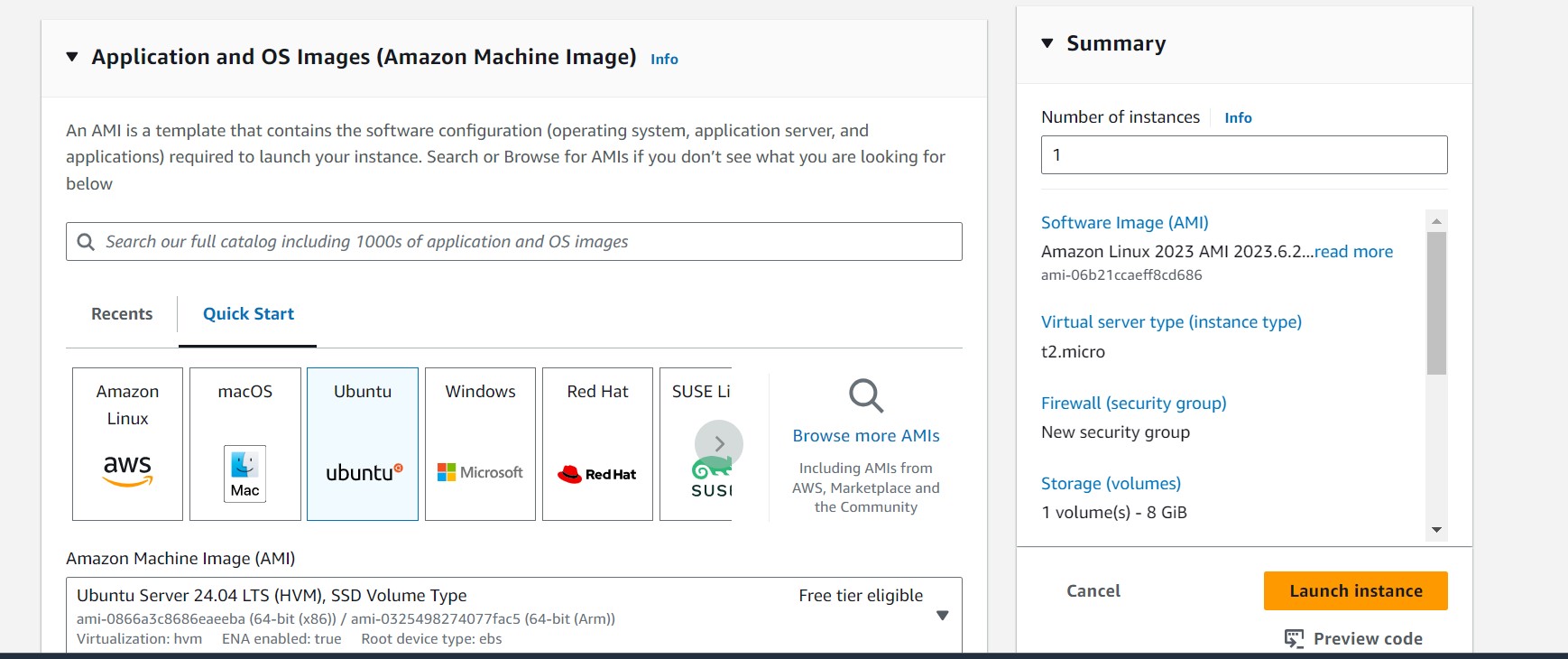
**Kubernetes Deployment**

A Kubernetes Deployment is used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application. Deployments can scale the number of replica pods, enable the rollout of updated code in a controlled manner, or roll back to an earlier deployment version if necessary.

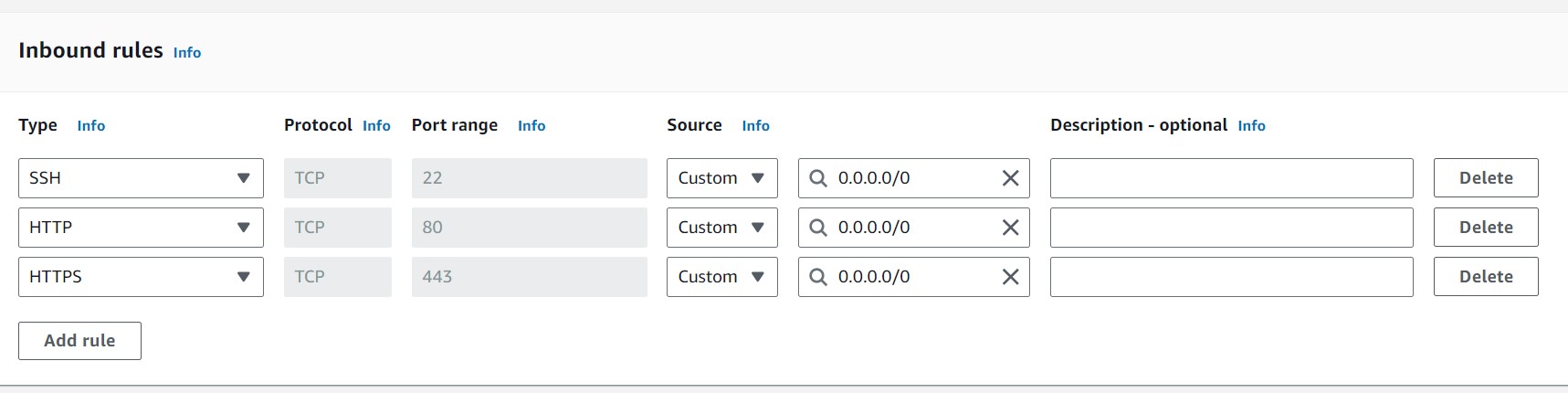
**Steps:**

1. Create an EC2 Ubuntu Instance on AWS.

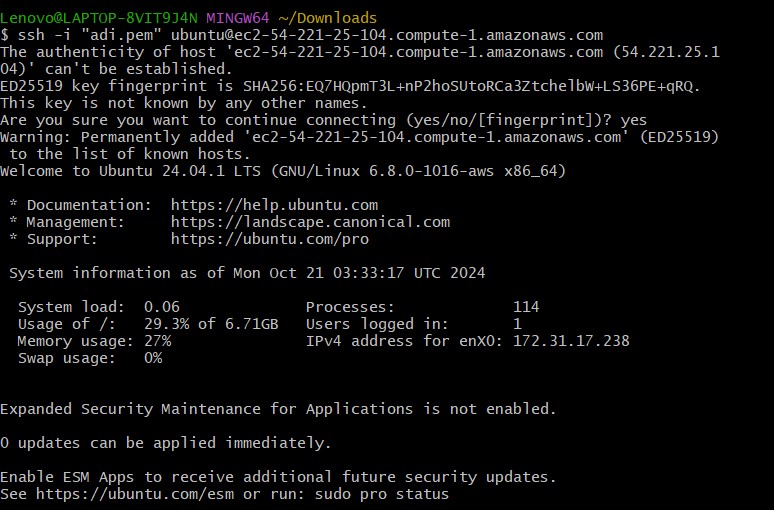




1. Edit the Security Group Inbound Rules to allow SSH

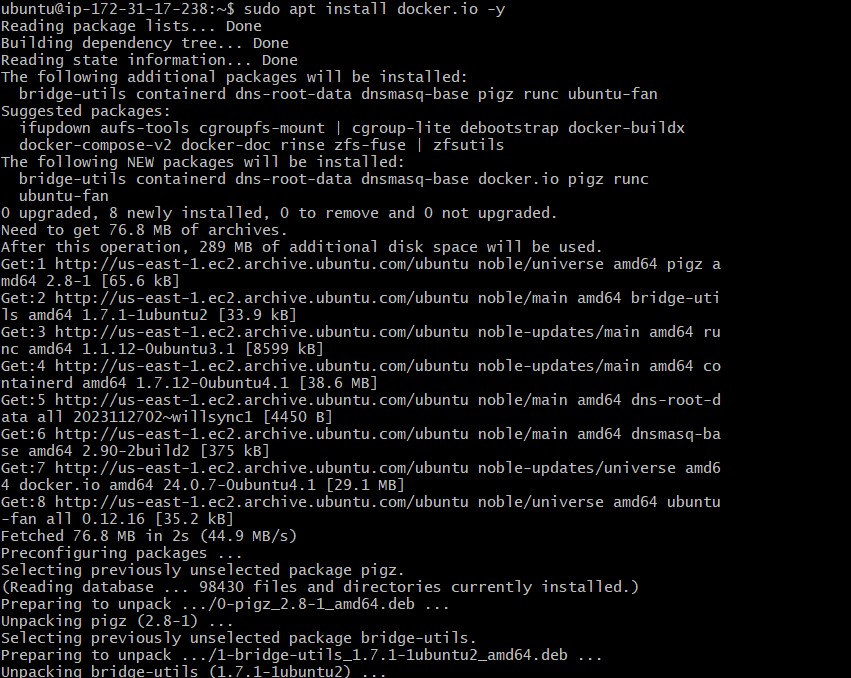


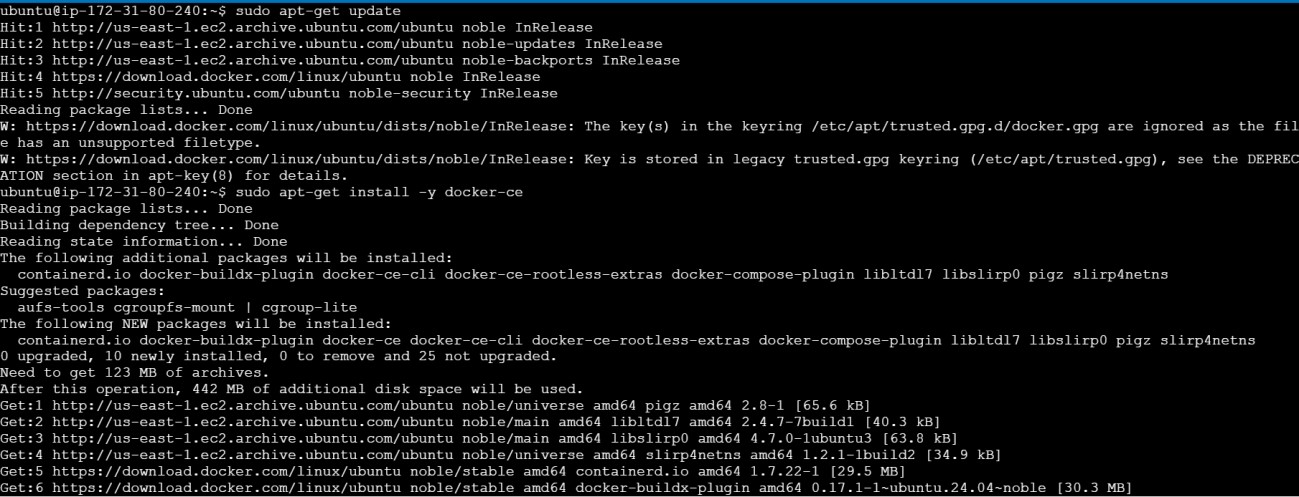
1. SSH into the machine **ssh -i <keyname>.pem ubuntu@<public\_ip\_address>**

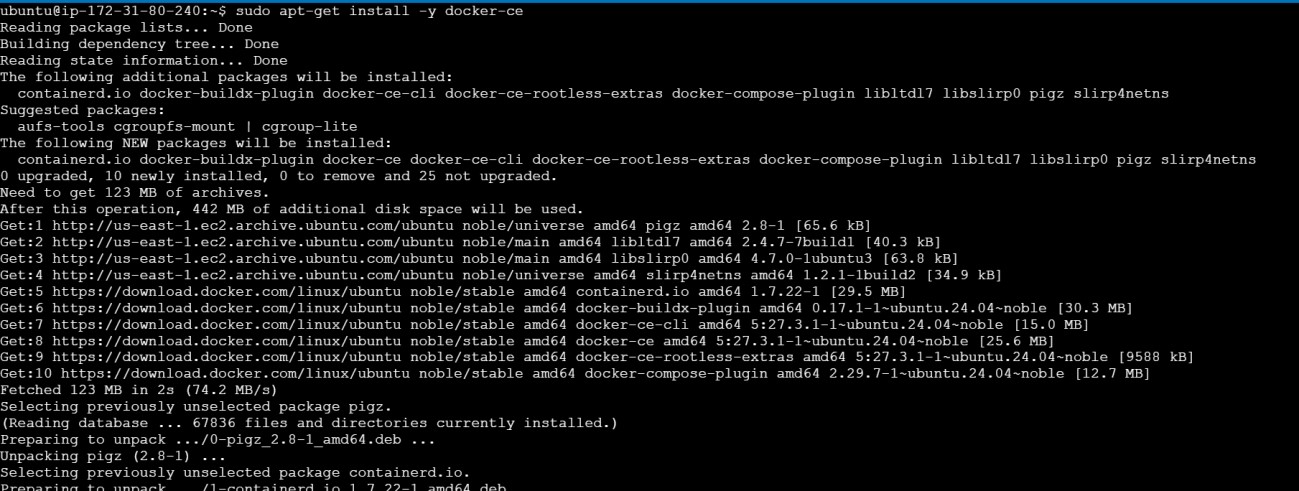


**Step 4: Run the below commands to install and setup Docker. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null sudo add-apt-repository "deb**

**[arch=amd64]** [**https://download.docker.com/linux/ubuntu**](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**

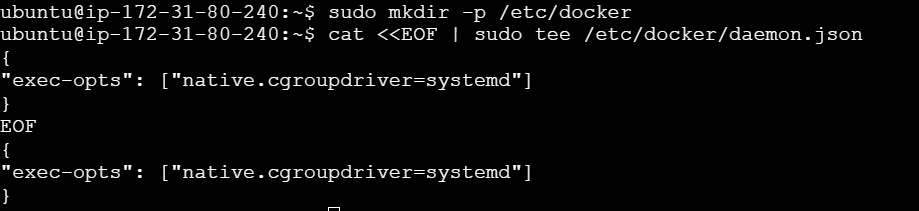
**cat <<EOF | sudo tee /etc/docker/daemon.json**

**{**

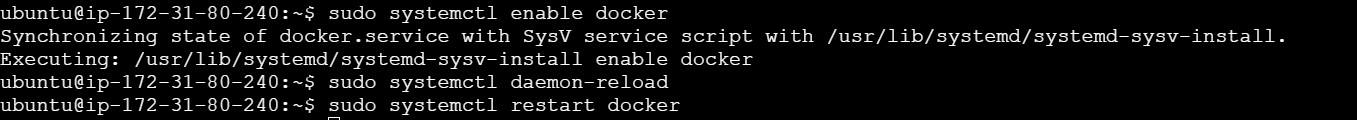
**"exec-opts": ["native.cgroupdriver=systemd"]**

**}**

**EOF**



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



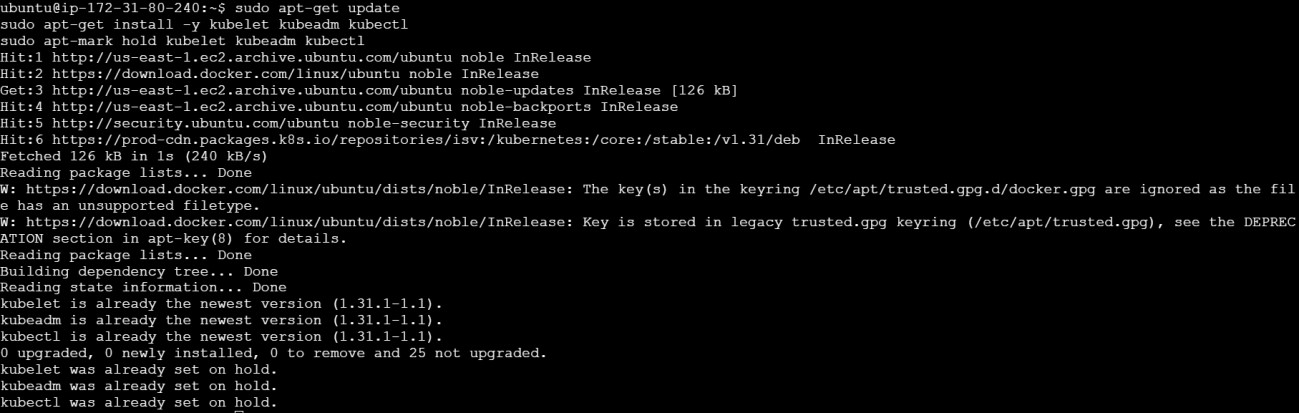
**Step 5: Run the below command to install Kubernets. curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o**

**/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**

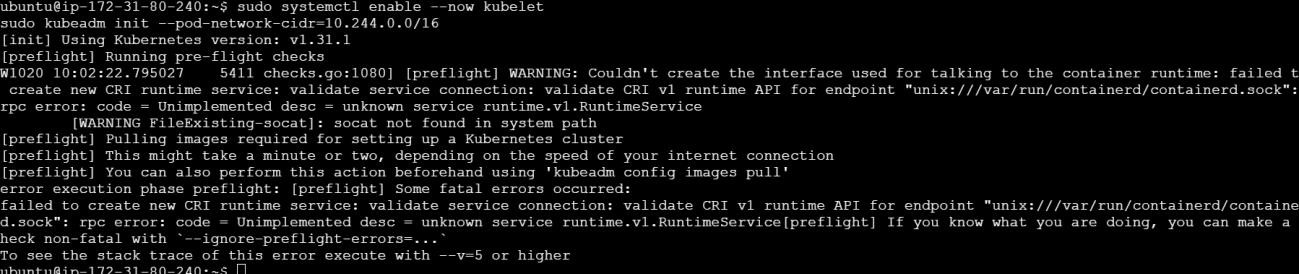


# sudo apt-get update sudo apt-get install -y kubelet kubeadm kubectl sudo apt-mark hold kubelet kubeadm kubectl

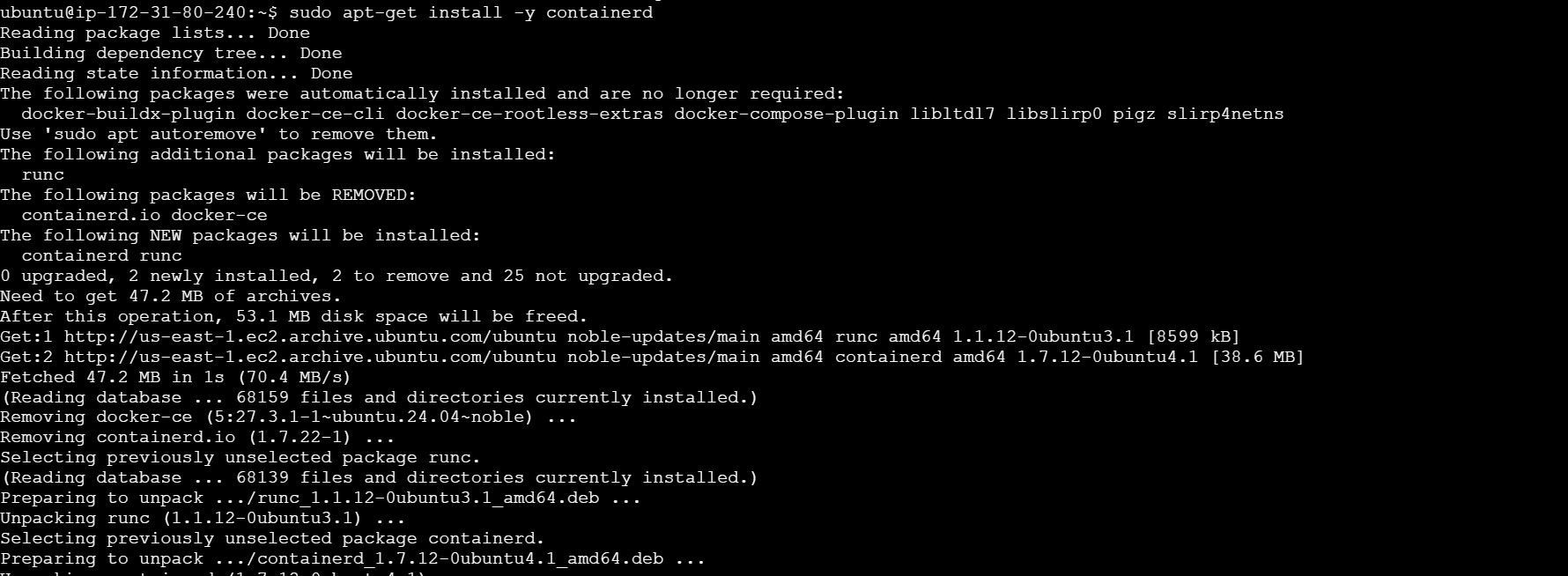


**sudo systemctl enable --now kubelet**

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



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

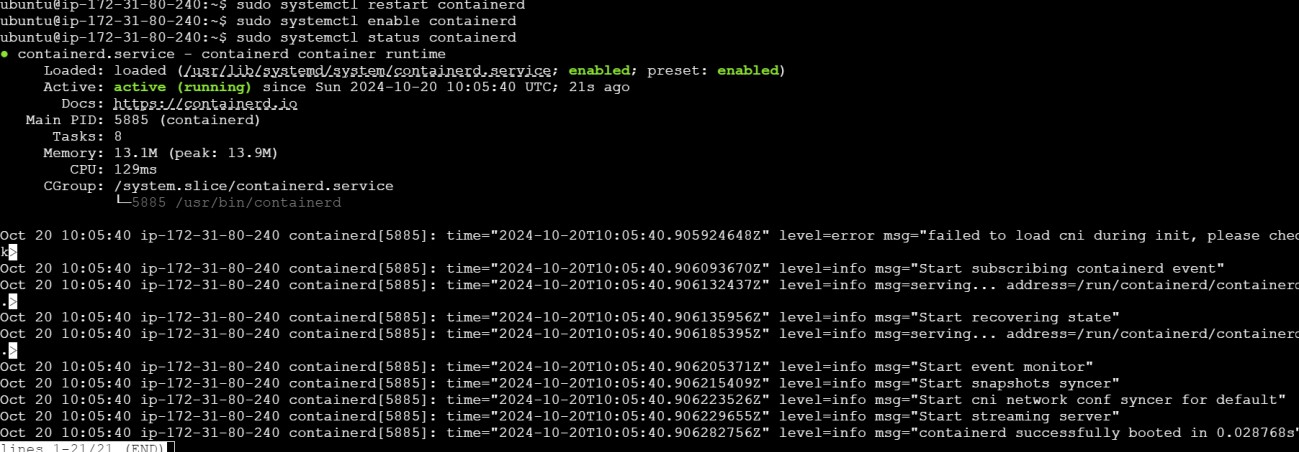


**sudo mkdir -p /etc/containerd**

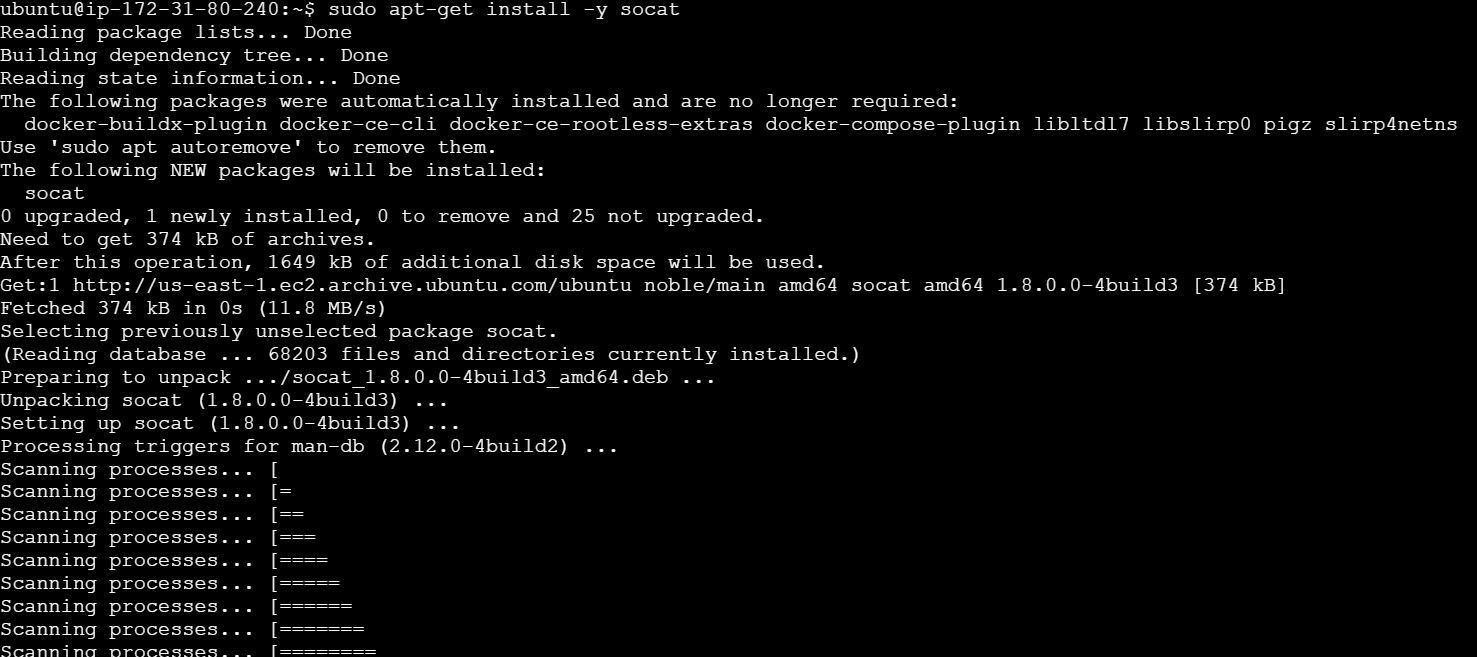
**sudo containerd config default | sudo tee /etc/containerd/config.toml**



**sudo systemctl restart containerd sudo systemctl enable containerd sudo systemctl status containerd**

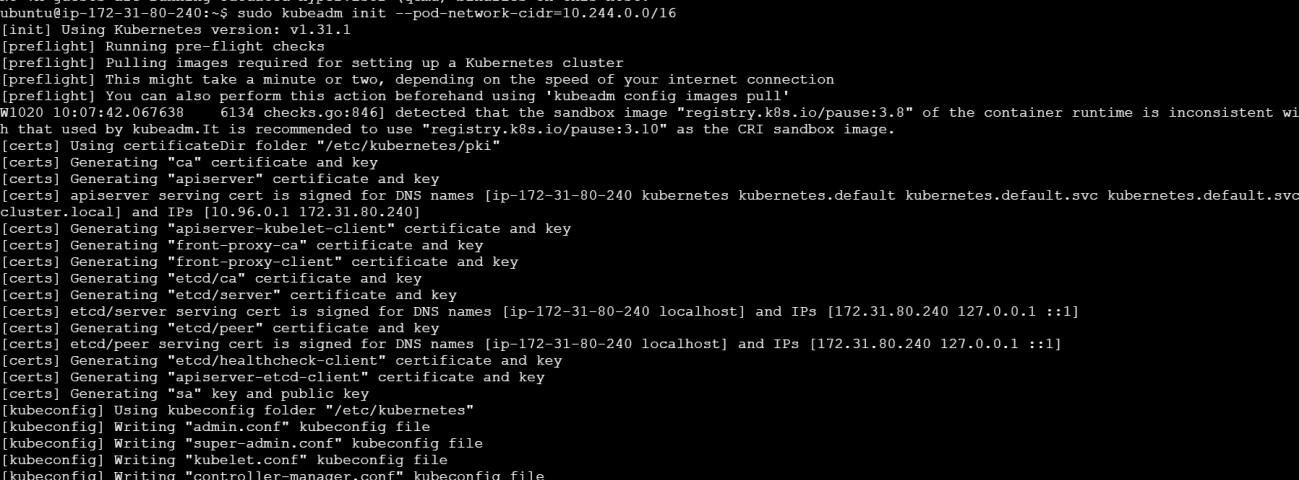


**sudo apt-get install -y socat**



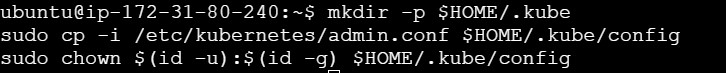
**Step 6: Initialize the Kubecluster**

**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**

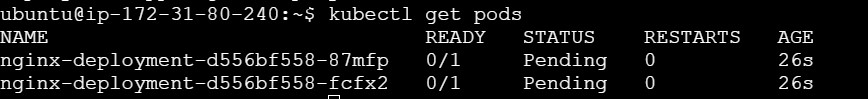
**https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml**



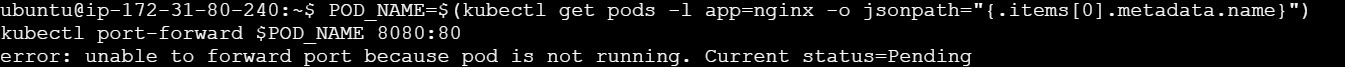
**kubectl apply -f** [**https://k8s.io/examples/application/deployment.yaml**](https://k8s.io/examples/application/deployment.yaml)



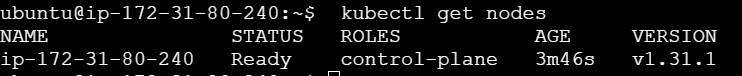
# kubectl get pods



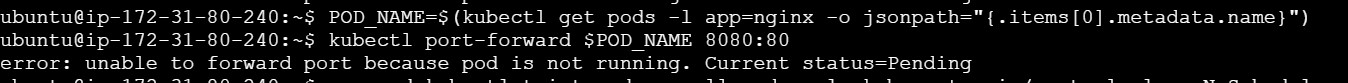
# POD\_NAME =$(kubectl get pods -l app=nginx -o jsonpath = “{.item[0].metadata.name}”)



# kubectl get nodes



**POD\_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}") kubectl port-forward $POD\_NAME 8080:80**



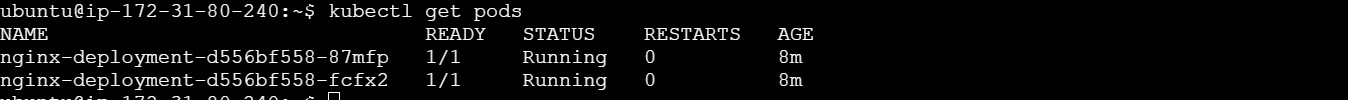
**command kubectl tain tnodes--all node-role.kubernetes.io/control-plane-node/ip-172-3120-171 untainted**



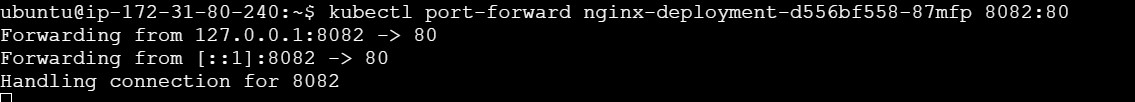
**kubectl get nodes**



# kubectl get pods



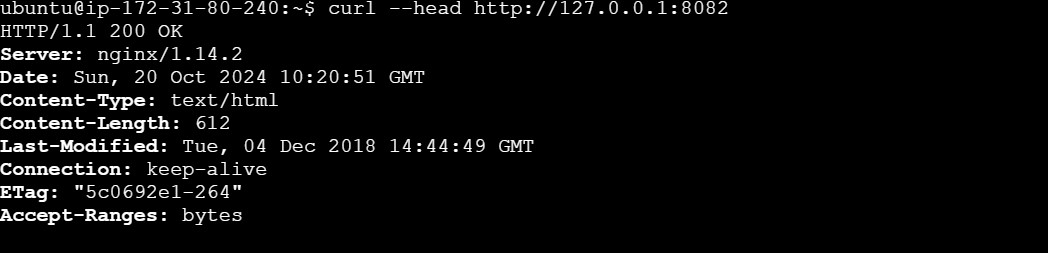
**POD\_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}") kubectl port-forward $POD\_NAME 8082:80**



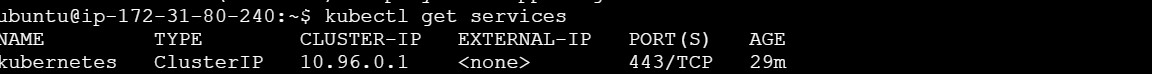
**Step 8: Verify your deployment**

**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://127.0.0.1:8082**](http://127.0.0.1:8082/)



# kubectl get services



# kubectl create deployment nginx –image=nginx



# kubectl get deployments

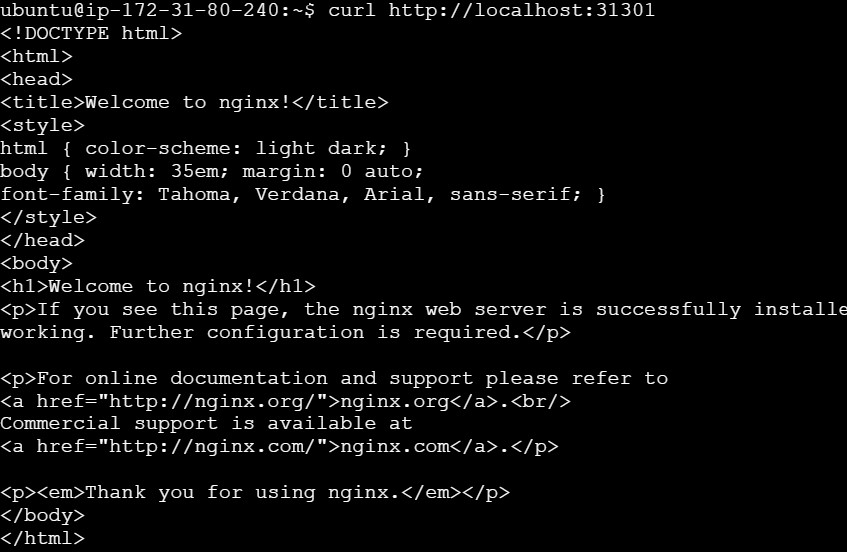


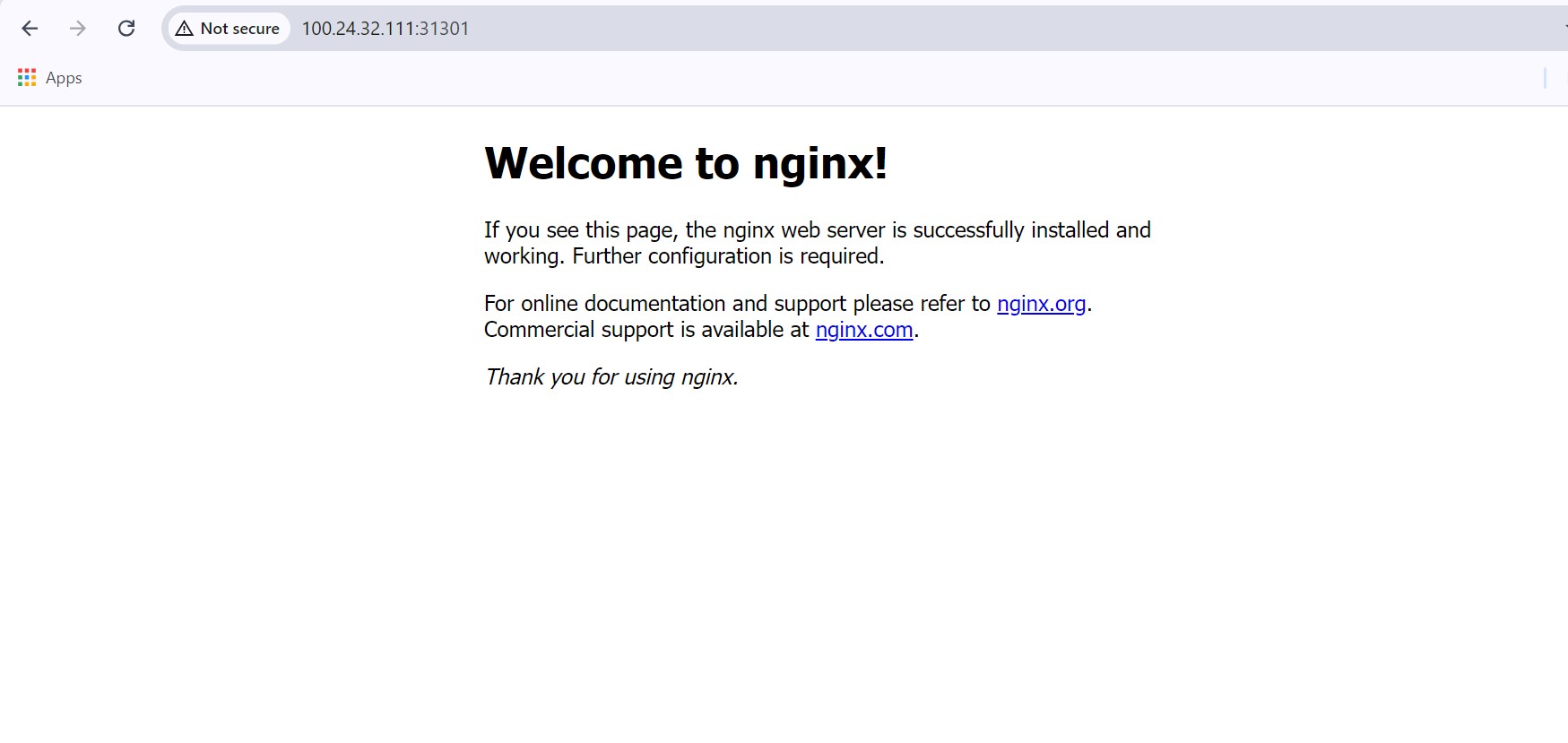
# kubectl expose deployment nginx --type=NodePort --port=80



**Nginx server is running successfully on the EC2 instance, and it's accessible locally via localhost on port 31801.**

**curl** [**http://127.0.0.1/31801**](http://127.0.0.1/31801)





**Conclusion:**

In this experiment, we successfully set up Kubernetes and Docker on an AWS EC2 Ubuntu instance, configured the necessary settings, and initialized a Kubernetes cluster. We deployed an Nginx server using a Kubernetes Deployment and implemented the Flannel networking plugin for pod communication. By checking the pod status and forwarding ports, we were able to access the Nginx server locally. The successful `200 OK` response from the `curl` command confirmed that the deployment was functioning correctly. This setup highlighted key Kubernetes operations, such as cluster management, application deployment, and verification, demonstrating the effectiveness of Kubernetes in orchestrating containerized applications efficiently.