**Name – Nandan Shailesh Kasat**

**Kubernetes Task 1**

Q1. Write a note on Kubernetes Architecture. Explain about each component of Kubernetes cluster.

Ans:

Kubernetes follows a client-server architecture. It’s designed as a set of loosely coupled and highly modular components, which makes the system flexible, extensible, and easy to manage. A master and worker nodes constitute a Kubernetes cluster.

Overview of the Kubernetes Architecture:

Control Manager

Kube Scheduler

Kube-Proxy

Kube

Api - Server

Kubelet

Container Engine

etcd

WORKER NODE

MASTER NODE

**Master Node:** The master node is responsible for the management of the Kubernetes cluster. It is the entry point for all administrative tasks.

The master node manages worker nodes and pods in the cluster. The components of the master node include:**API Server (kube-apiserver):** Acts as the frontend for the Kubernetes control plane.

It exposes the Kubernetes API and is the main management point of the entire cluster.

**Controller Manager (kube-controller-manager):** Runs controllers, which are background threads that handle routine tasks in the cluster.

**Scheduler (kube-scheduler):** Distributes work or containers across multiple nodes. It looks for newly created pods and assigns them to nodes.

**etcd:** A consistent and highly-available key value store used as Kubernetes’ backing store for all cluster data.

**Worker Nodes:** Worker nodes are the machines (VMs, physical servers, etc.) that run your applications and cloud workflows.

The components of a worker node include:**Kubelet:** An agent that runs on each node in the cluster. It makes sure that containers are running in a pod1.

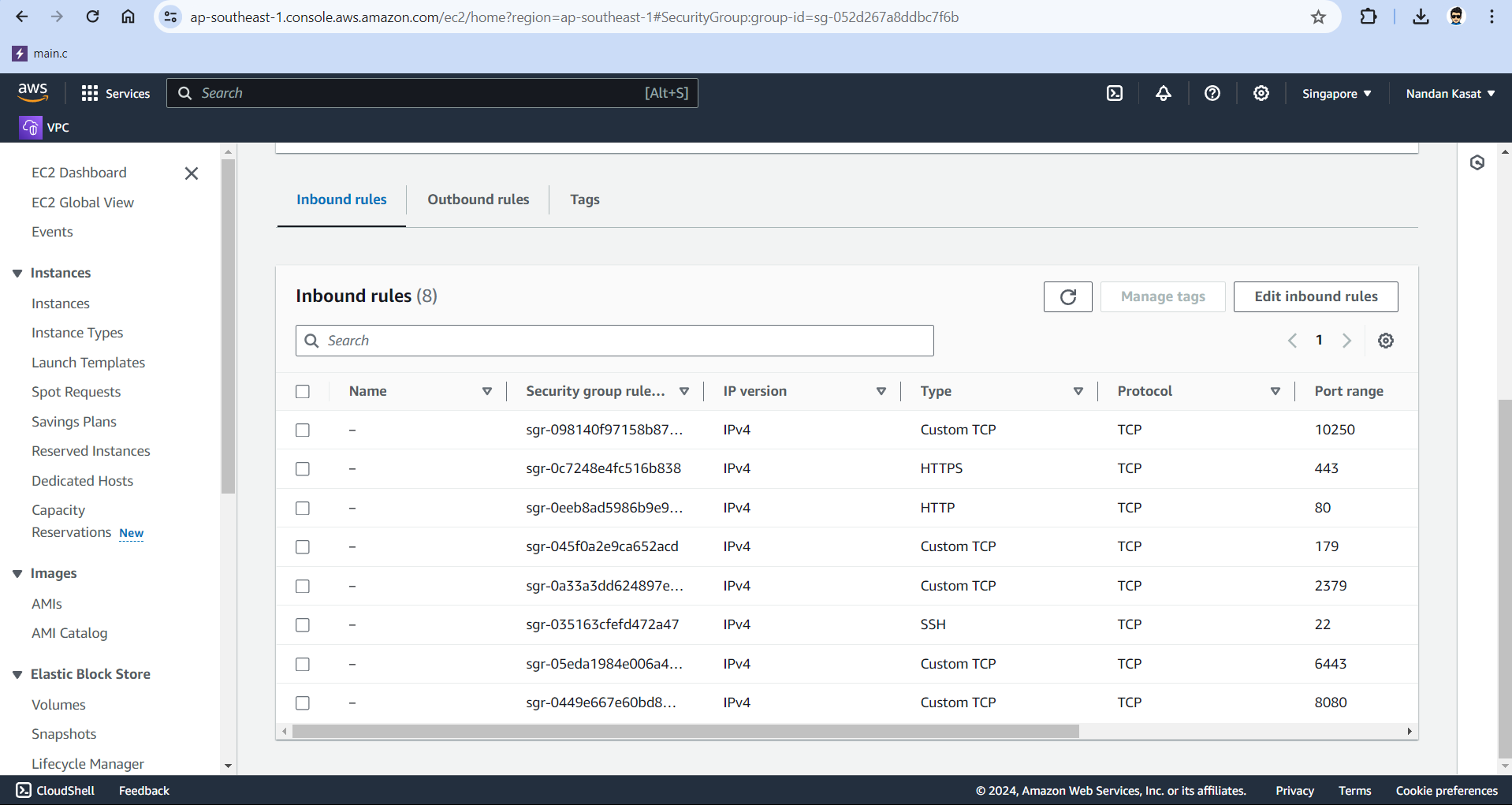
**Kube Proxy (kube-proxy):** A network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.

**Pods (Container Engine):** The smallest and simplest unit in the Kubernetes object model that you create or deploy.

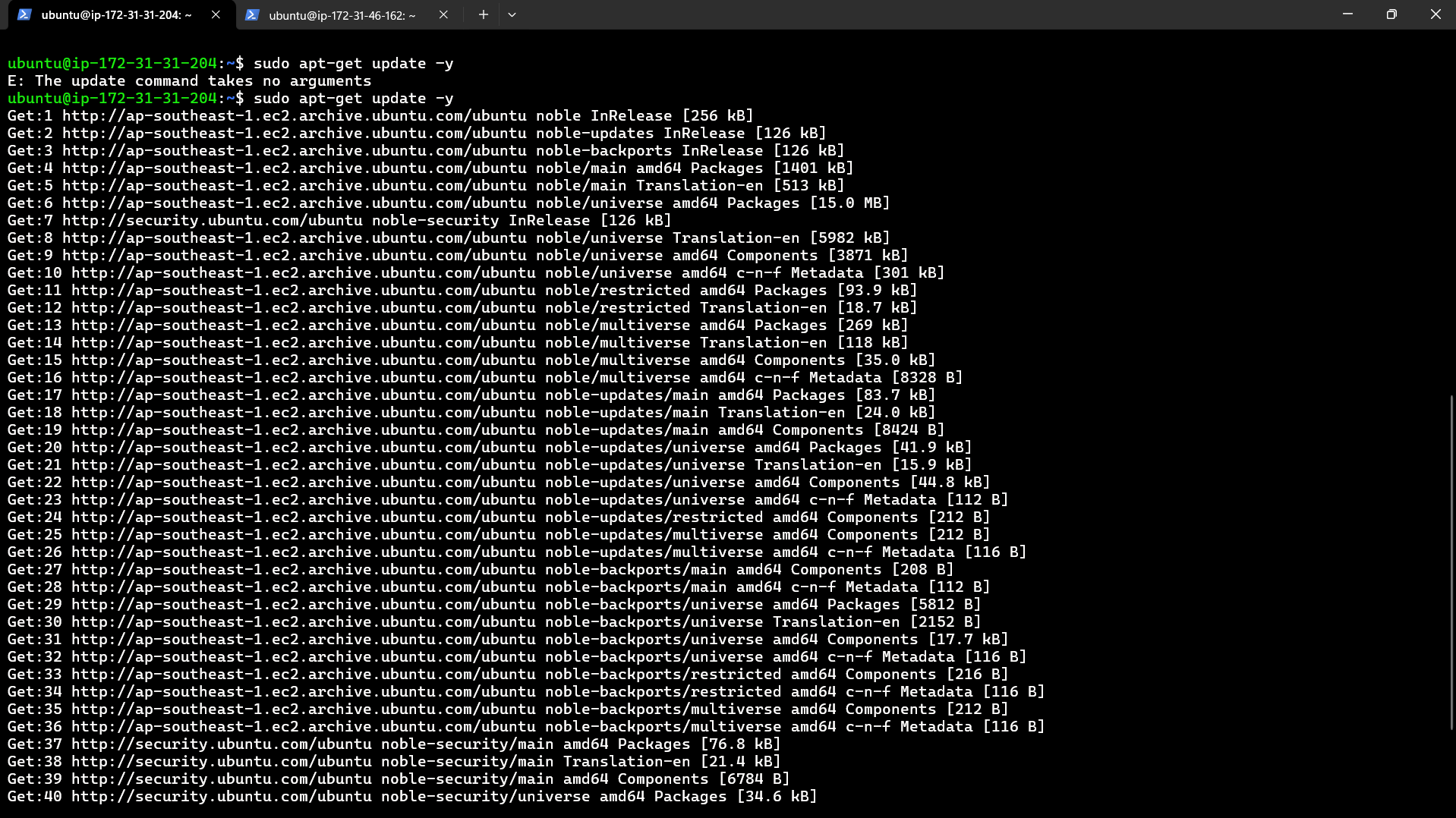
Q2. Prepare a documentation on Kubernetes setup on Ubuntu. With screenshot of each command.

On Master:

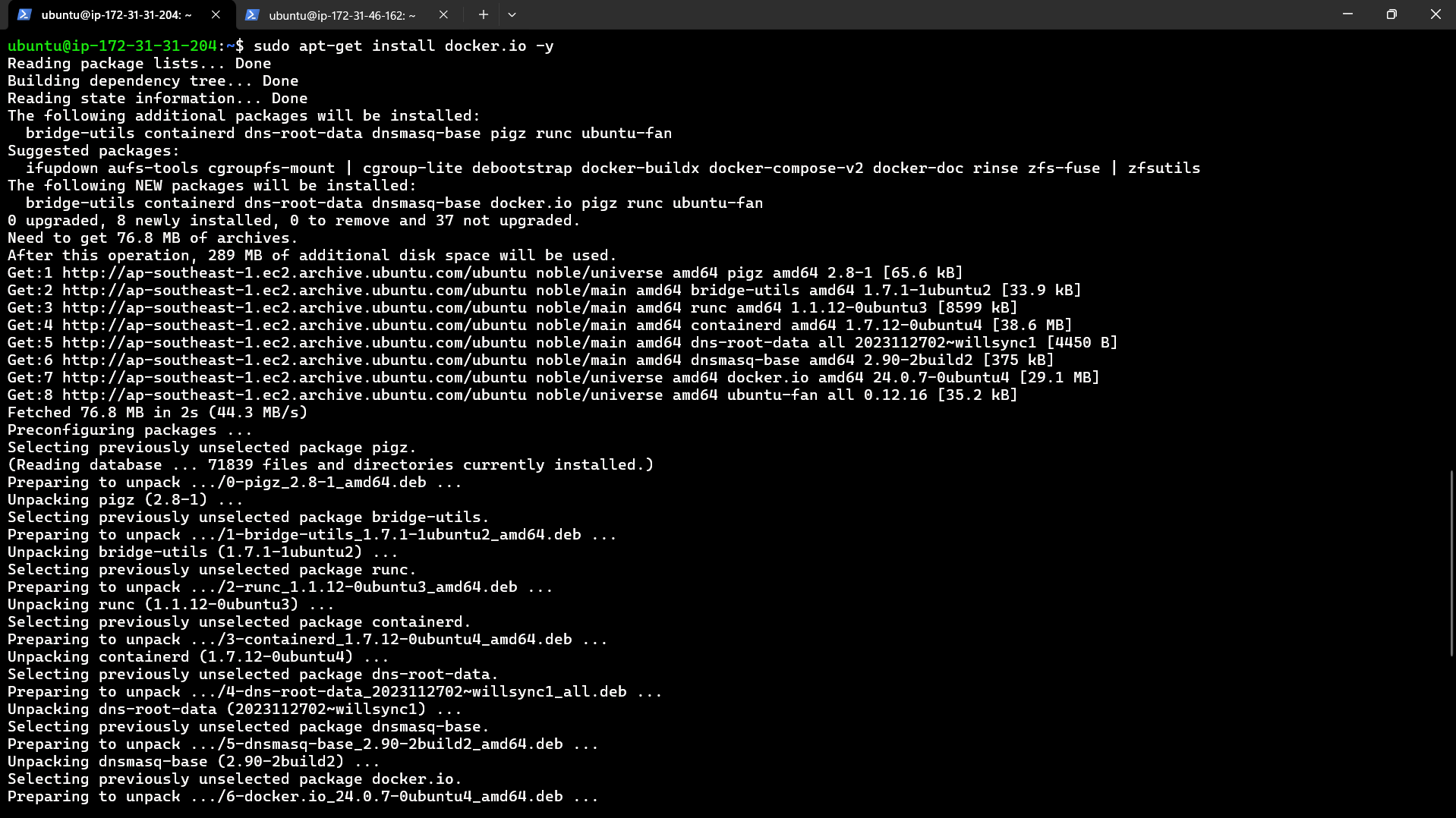
Add Rules 8080, 179, 6443, 10250, 2379 to Security Group of Master Node instance (kube\_master).



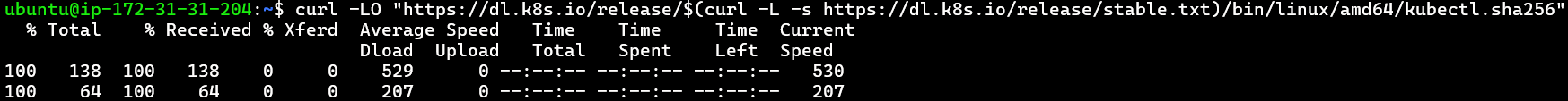
1. sudo apt-get update –y



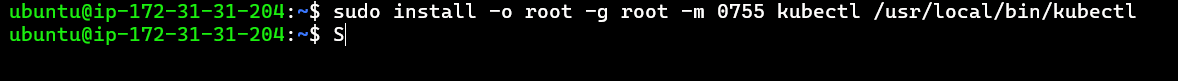
1. sudo apt-get install docker.io –y



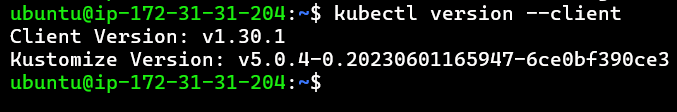
3 curl -LO “[https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256](https://dl.k8s.io/release/$(curl%20-L%20-s%20%20%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256)”



1. sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl



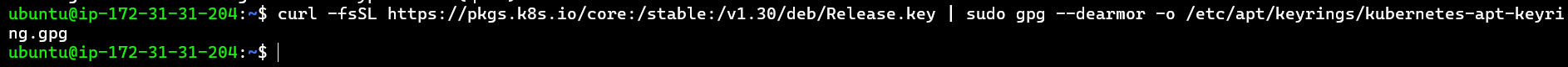
1. kubectl version –client



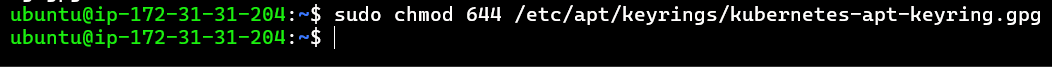
1. sudo apt-get update
2. sudo apt-get install -y apt-transport-https ca-certificates curl



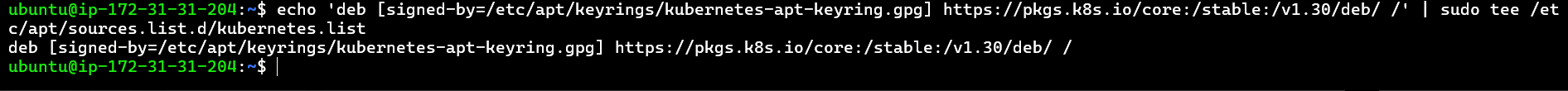
1. curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg



1. sudo chmod 644 /etc/apt/keyrings/kubernetes-apt-keyring.gpg



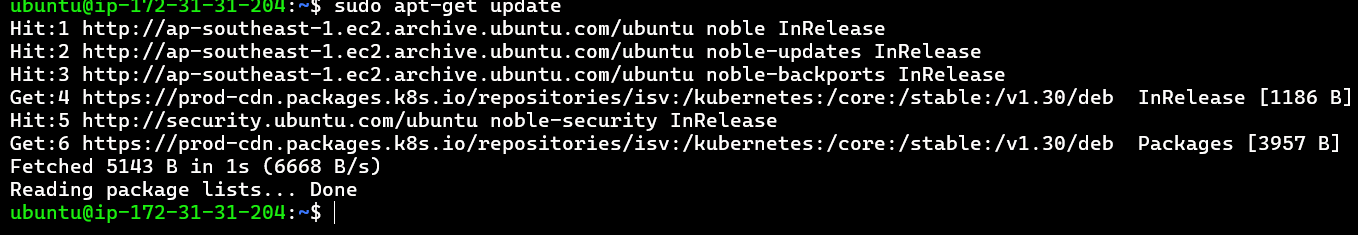
1. echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list



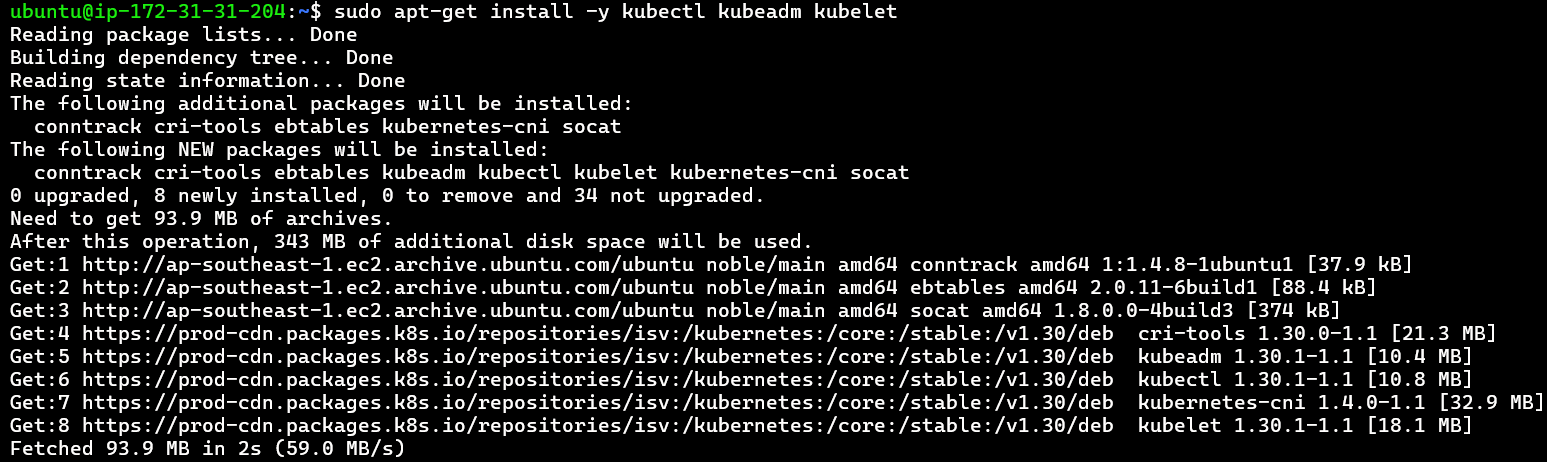
1. sudo chmod 644 /etc/apt/sources.list.d/kubernetes.list



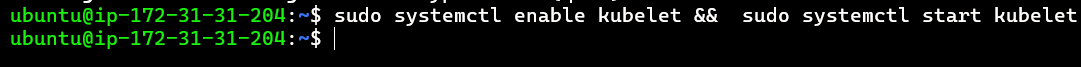
1. sudo apt-get update



1. sudo apt-get install -y kubectl kubeadm kubelet

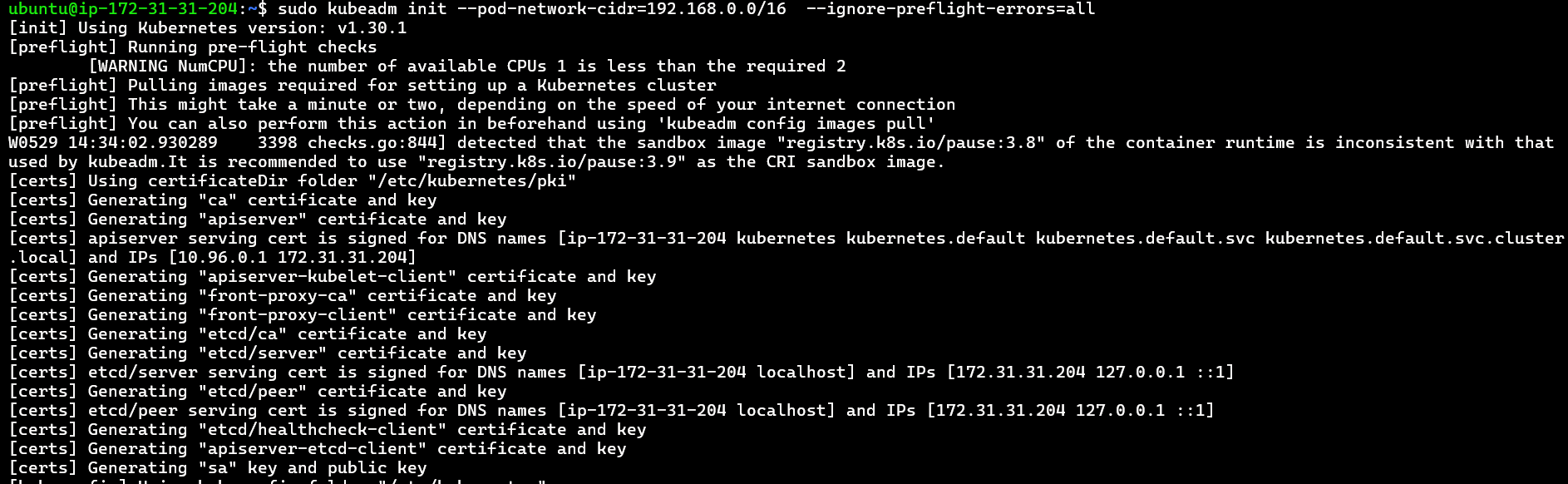


1. sudo systemctl enable kubelet && sudo systemctl start kubelet



///// 16 sudo kubeadm init --pod-network-cidr=192.168.0.0/16

17 sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --ignore-preflight-errors=all



18 mkdir -p $HOME/.kube



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

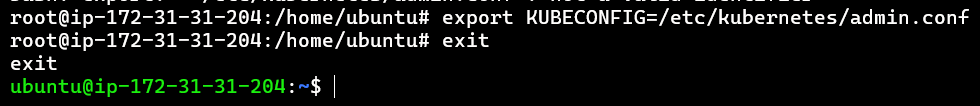


20 sudo chown $(id -u):$(id -g) $HOME/.kube/config



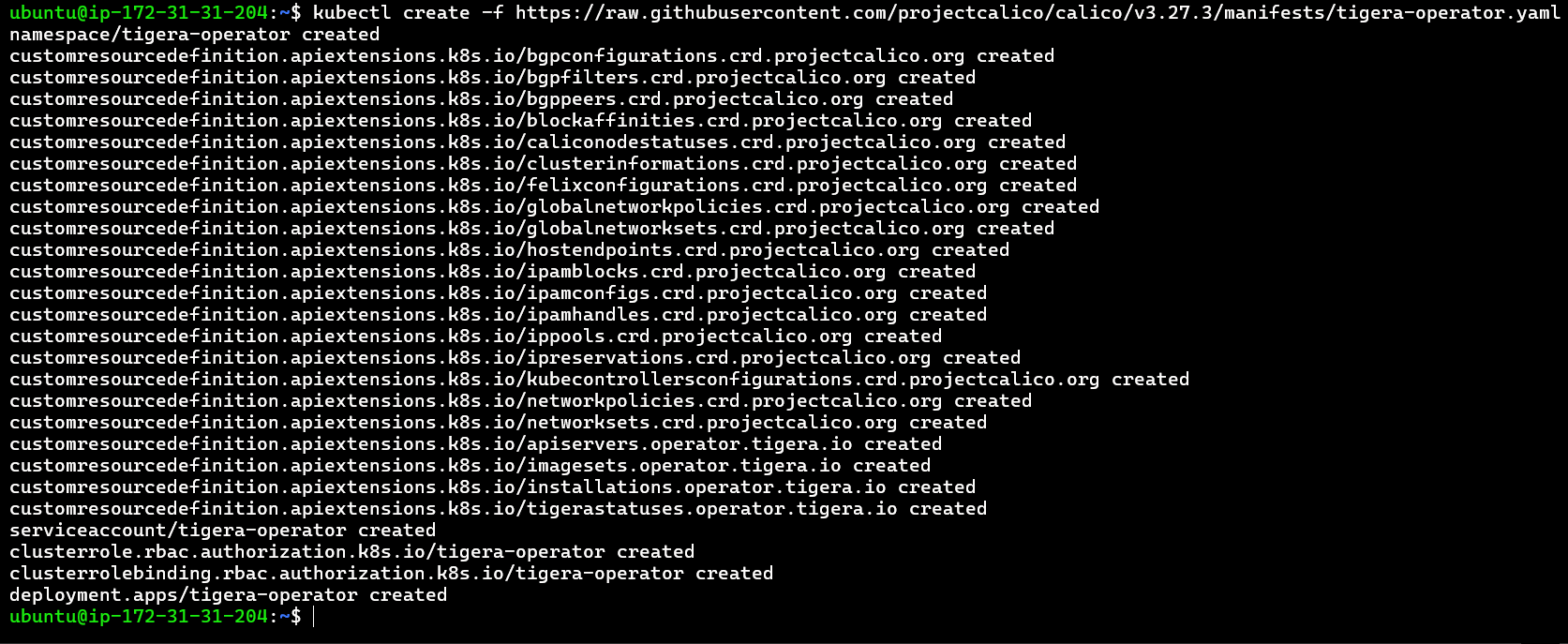
21 sudo su

export KUBECONFIG=/etc/kubernetes/admin.conf



Exit

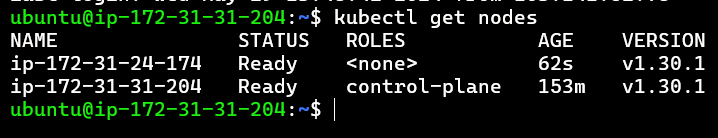
22 kubectl create -f <https://raw.githubusercontent.com/projectcalico/calico/v3.27.3/manifests/tigera-operator.yaml>



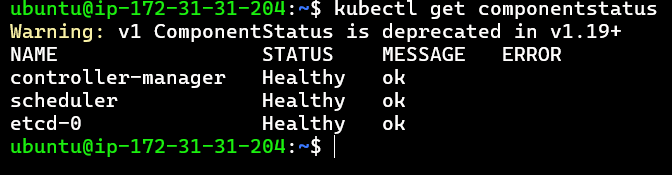
23 kubectl create -f <https://raw.githubusercontent.com/projectcalico/calico/v3.27.3/manifests/custom-resources.yaml>



24 kubectl get nodes

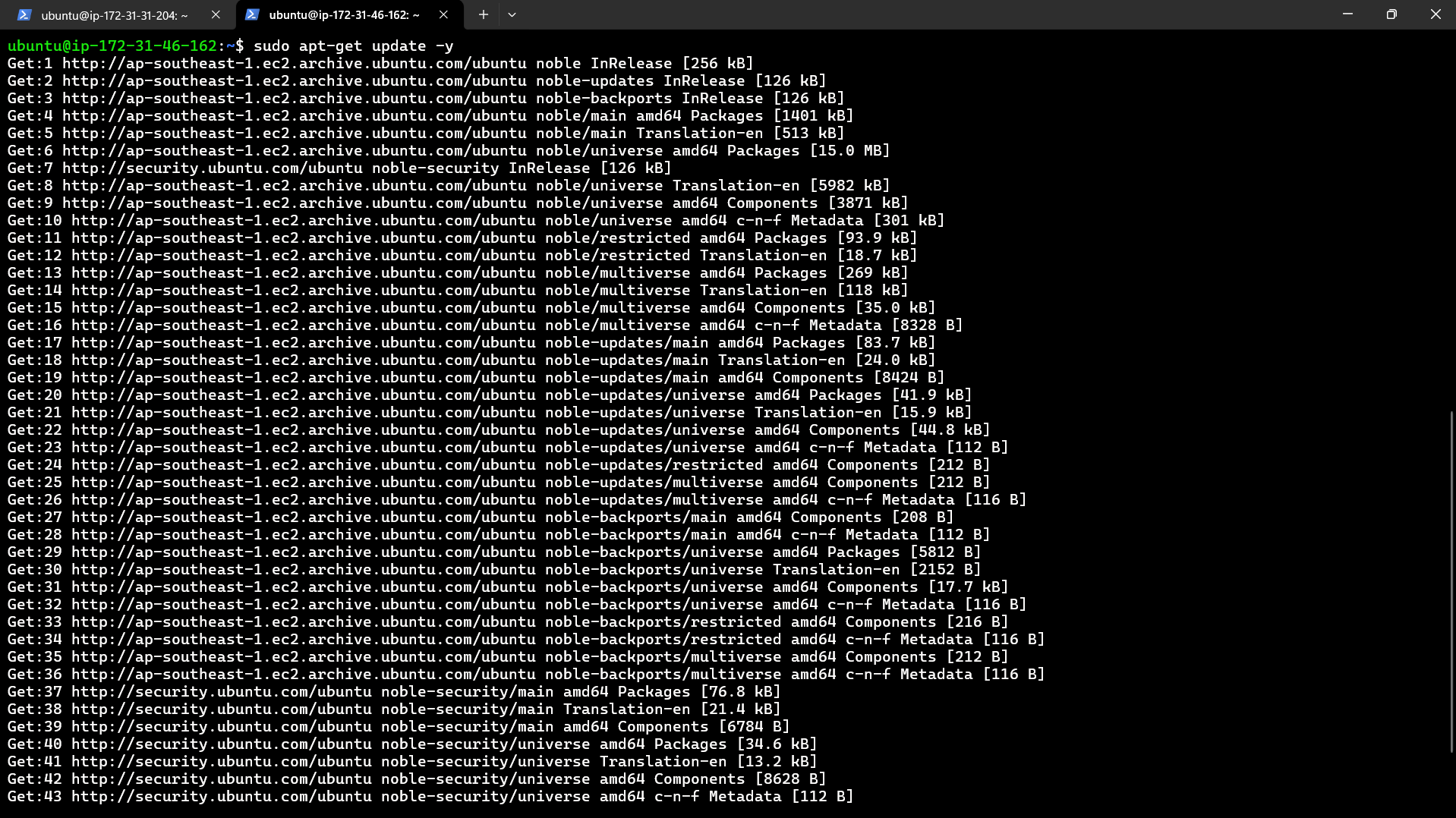


25 kubectl get componentstatus

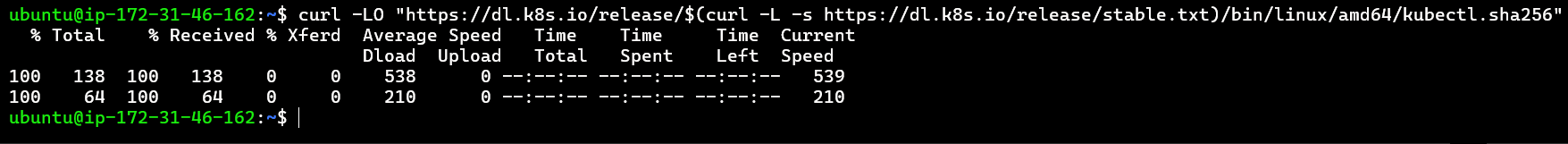


On Worker (kube\_worker):

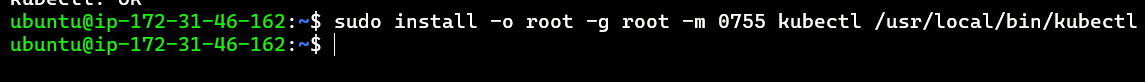
1. sudo apt-get update –y



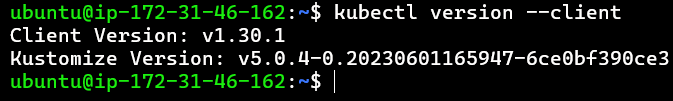
2 curl -LO “[https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256](https://dl.k8s.io/release/$(curl%20-L%20-s%20%20%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256)”



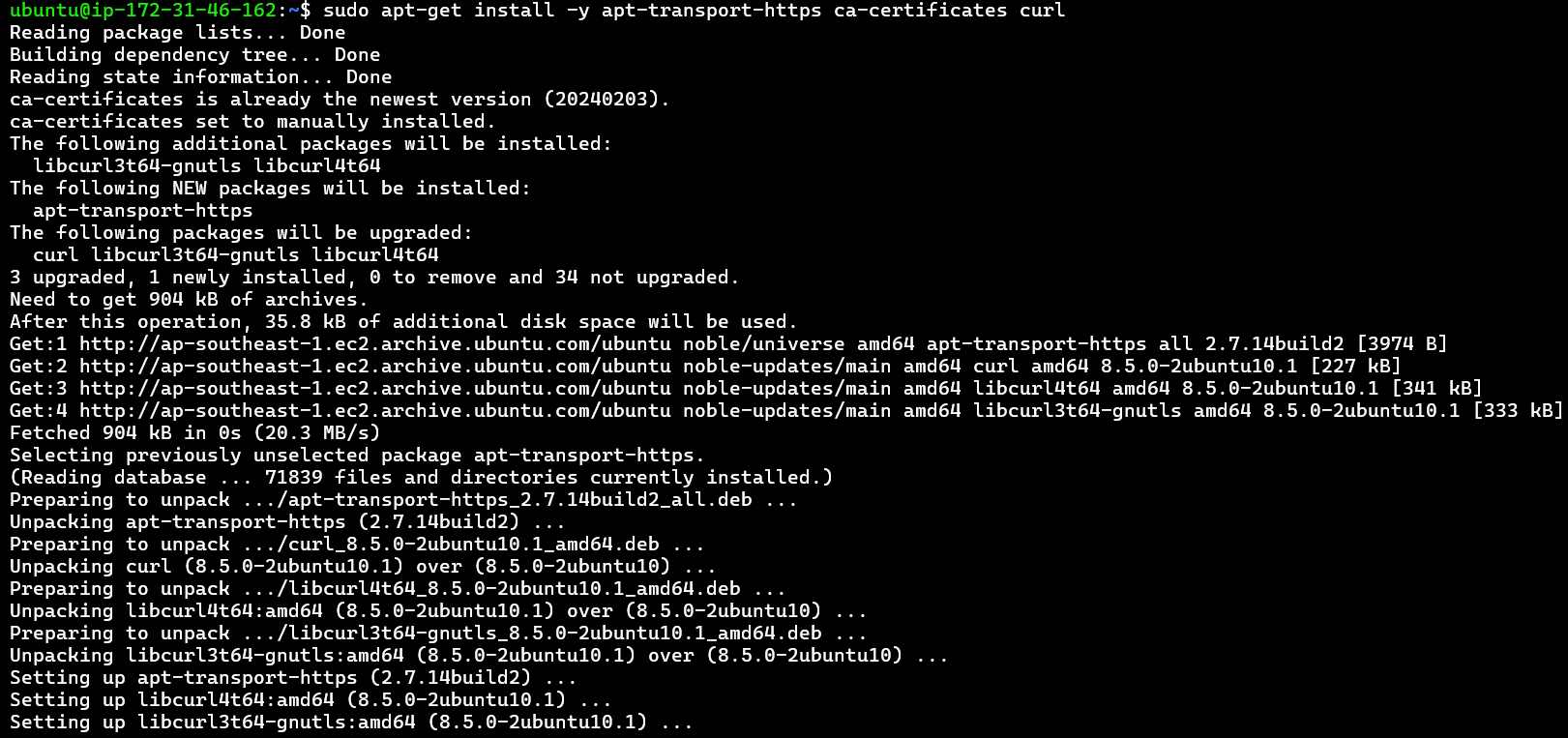
1. sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl



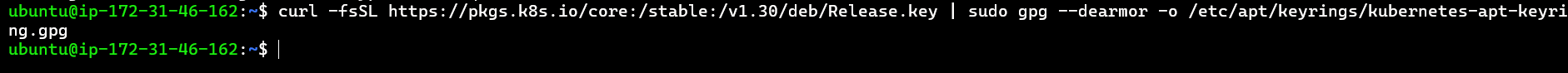
1. kubectl version –client



1. sudo apt-get update
2. sudo apt-get install -y apt-transport-https ca-certificates curl



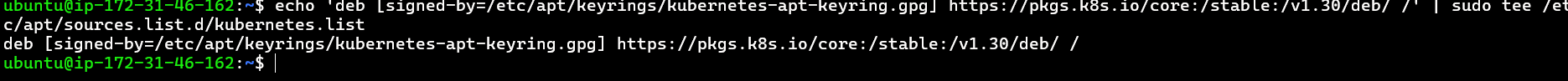
1. curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg



1. sudo chmod 644 /etc/apt/keyrings/kubernetes-apt-keyring.gpg



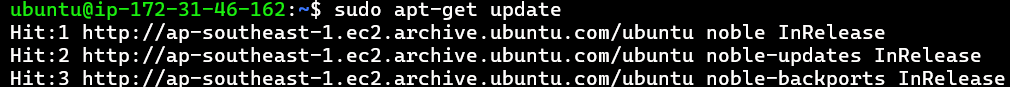
1. echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list



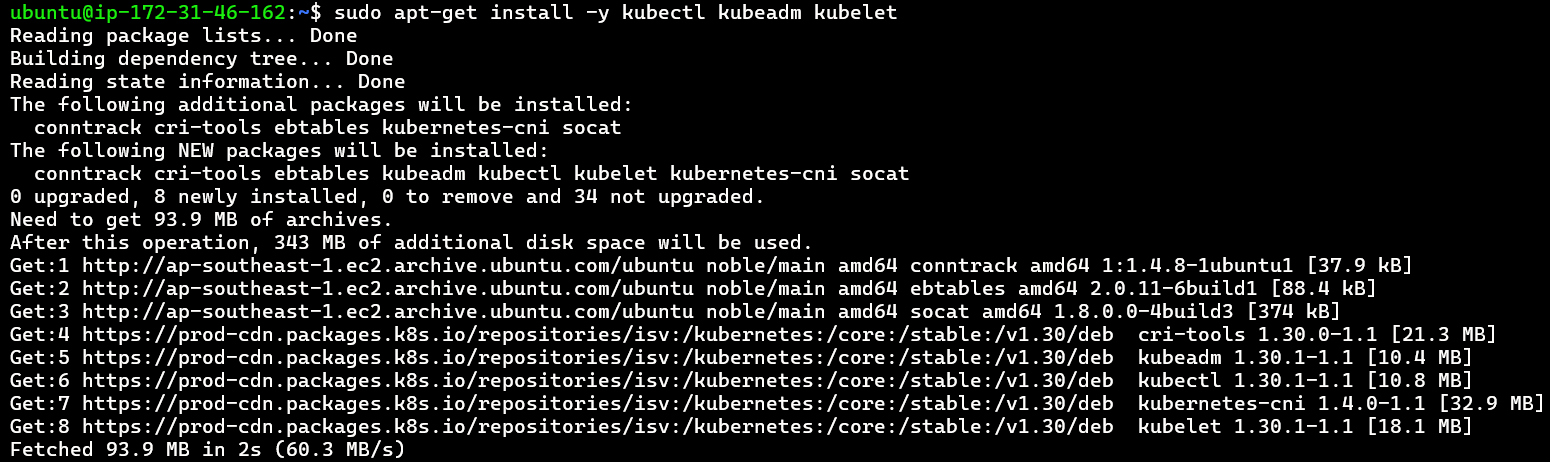
1. sudo chmod 644 /etc/apt/sources.list.d/kubernetes.list



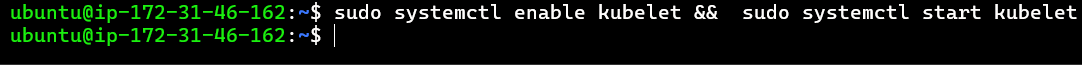
1. sudo apt-get update



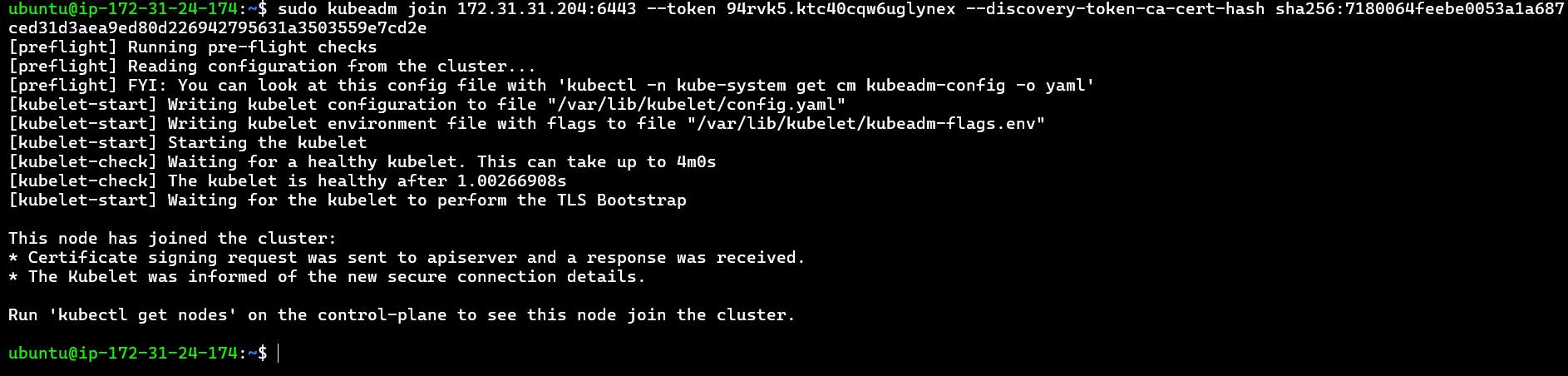
1. sudo apt-get install -y kubectl kubeadm kubelet



1. sudo systemctl enable kubelet && sudo systemctl start kubelet



1. sudo kubeadm join 172.31.31.204:6443 --token 94rvk5.ktc40cqw6uglynex --discovery-token-ca-cert-hash sha256:7180064feebe0053a1a687ced31d3aea9ed80d226942795631a3503559e7cd2e

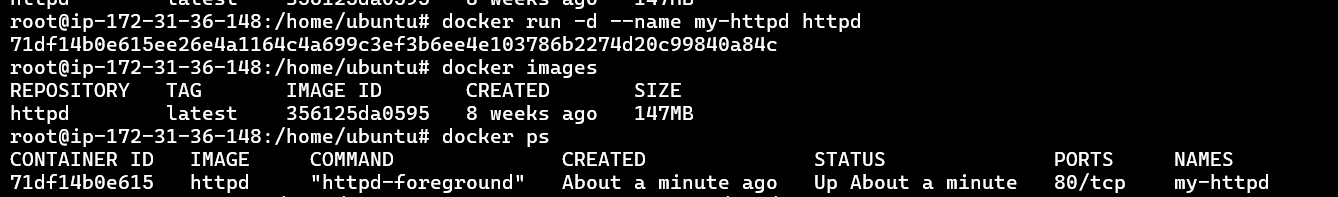


Q3. Write a manifest file to create an httpd container in Pod, create pod using that manifest file. Also go inside that httpd container and create own pages and try to access those pages from inside the container.

Ans:

Step 1: **Create a httpd images.**

**Step 2: Run httpd image:**



Step 3: **Go inside the httpd container and create your own pages:**

**docker exec -it 71df14b0e615 /bin/bash**

****

Step 4: Navigate to the htdocs directory



Step 6: Create HTML pages

root@71df14b0e615:/usr/local/apache2/htdocs# echo "<h1>Hello, World</h1>" > hello.html

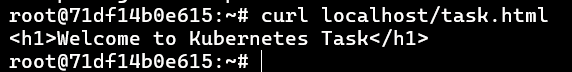
root@71df14b0e615:/usr/local/apache2/htdocs# echo "<h1>Welcome to Kubernetes Task</h1>" > task.html

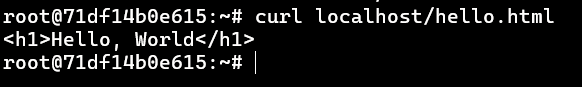


Step 7: **Access the pages from inside the container:**

root@71df14b0e615:~# curl localhost/task.html

<h1>Welcome to Kubernetes Task</h1>

****

****