TASK-01 KUBERNETES

Date: 20/05/24

Q.1 Write a note on Kubernetes Architecture. Explain about each component of Kubernetes cluster.

Kubernetes Architecture:S

Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Its architecture consists of several components that work together to create and manage a distributed system of containers.

Key Components of Kubernetes Cluster:

I. Master Node:

- The control plane of the Kubernetes cluster.
- Manages and orchestrates the cluster's various components.
- Consists of several components:

1. **API Server:** Exposes Kubernetes API, which clients (like kubectl) use to interact with the cluster.
2. **Scheduler:** Assigns nodes to newly created pods based on resource requirements and other constraints.
3. **Controller Manager:** Monitors the cluster's state and performs tasks such as node and pod management.
4. **etcd :** A distributed key-value store that stores cluster state and configuration data.

II. Worker Node :

- Runs the containers that form the application workload.
- Consists of several components:

1. **Kubelet:** Agent that runs on each node and communicates with the Kubernetes API server. It manages the node and its containers.
2. **Kube Proxy:** Maintains network rules on nodes. It handles routing of traffic to appropriate containers.
3. **Container Runtime:** Software responsible for running containers (e.g., Docker, containerd).

III. Pods:

- A pod is the smallest deployable unit in Kubernetes.
- It represents a single instance of a running process in your cluster.
- Pods can contain one or more containers that are tightly coupled and share resources, such as networking and storage.
- They are ephemeral by nature, meaning they can be created, destroyed, and replaced dynamically.

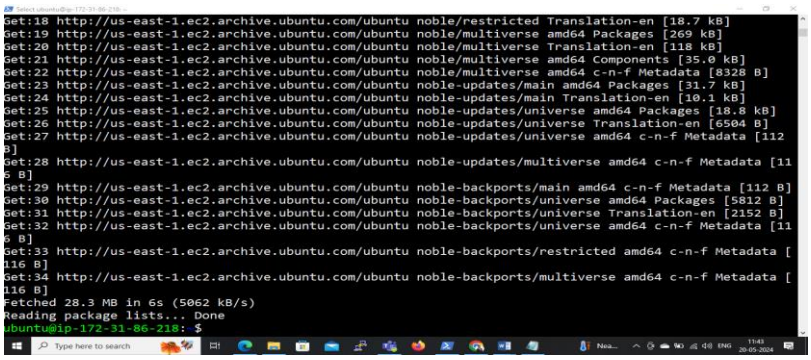
Q.2 Prepare a documentation on Kubernetes setup on Ubuntu. With screenshot of each command.

- Launch 2 Instances.
- For the instance of **Master Node** choose instance type as **t2.small**
- For the instance of **Worker Node** choose instance type as **t2.micro**

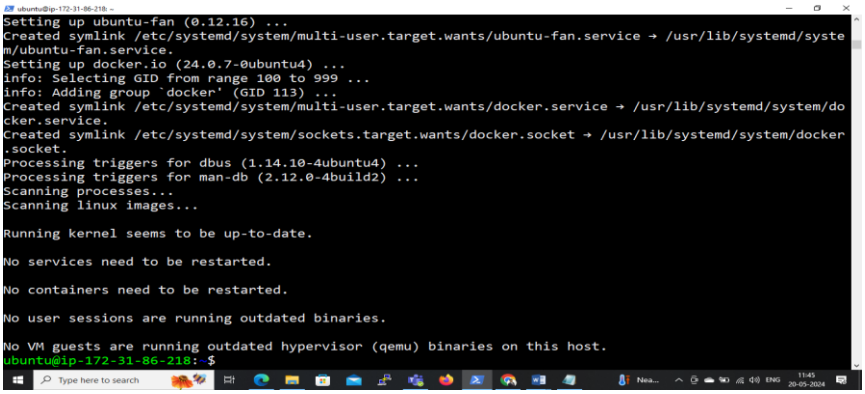
(**In Security Group of Master Node open port: 22, 443, 80, 8080, 179, 2379, 10250, 6443)**

On Master node & Worker node:

- sudo apt-get update –y

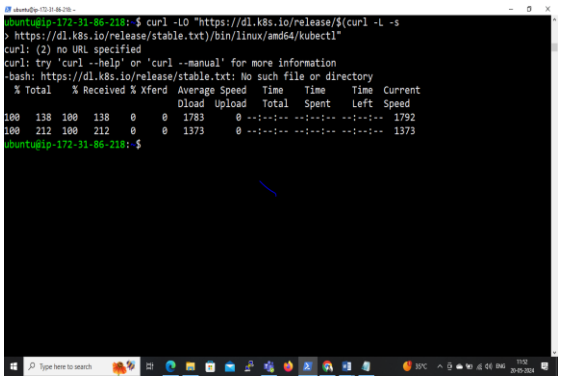


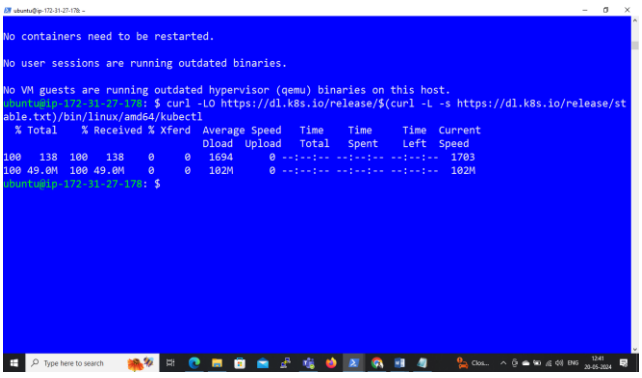
- sudo apt-get install docker.io –y



 # Download the latest release with the command

curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

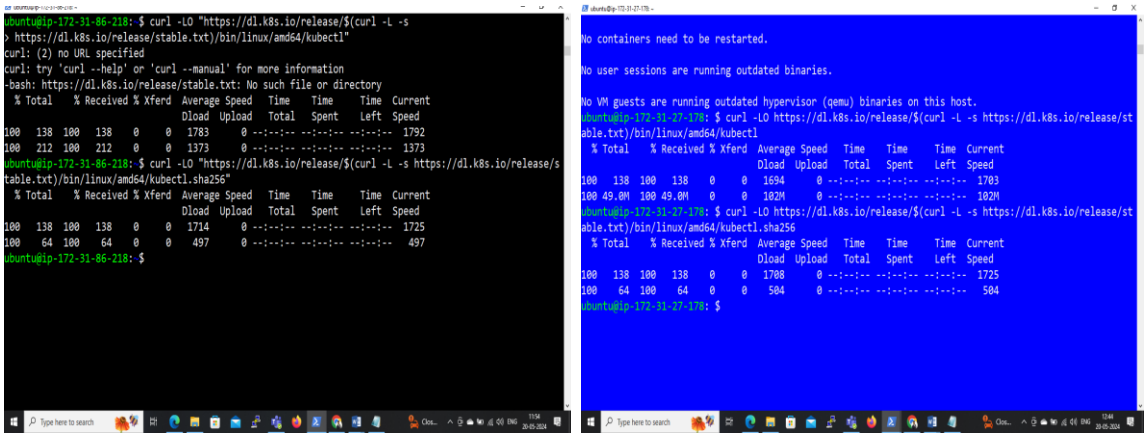




 # Validate the binary (optional)

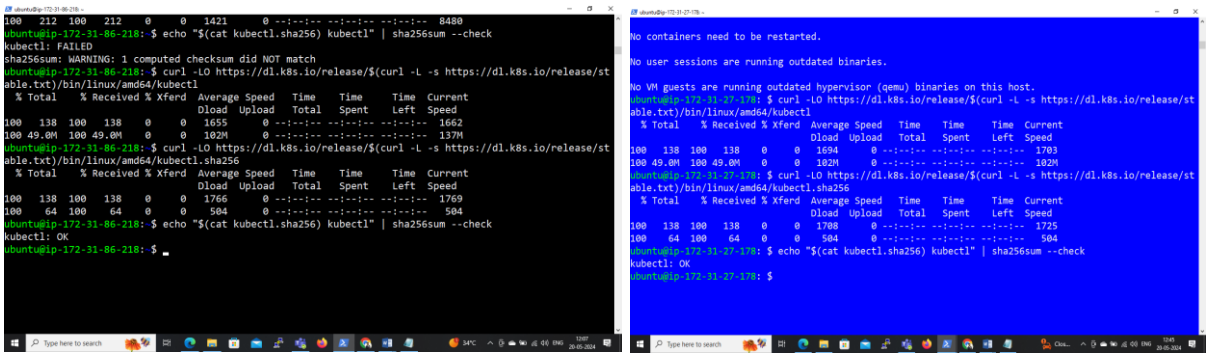
 # Download the kubectl checksum file:

curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256



Validate the kubectl binary against the checksum file:

- echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

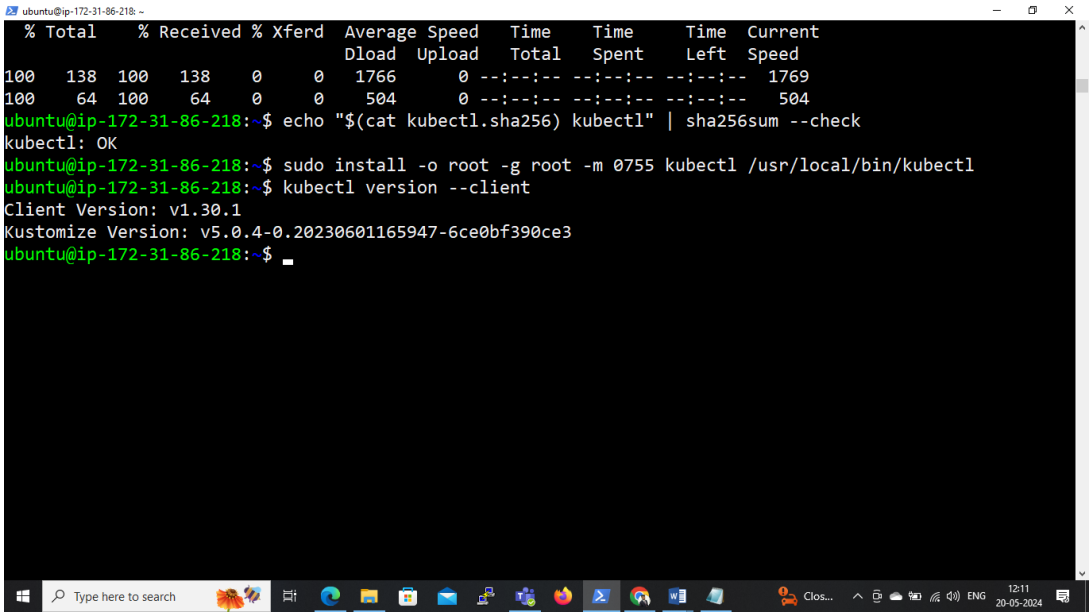


Install kubectl

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

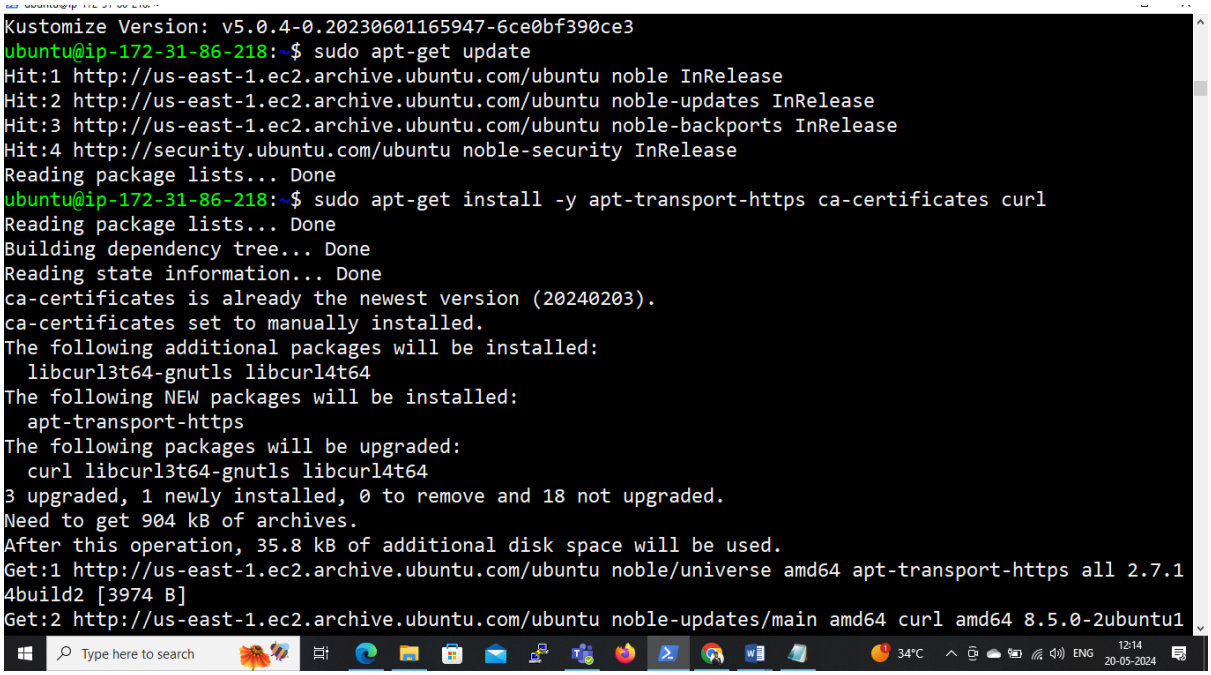
Test to ensure the version you installed is up-to-date:

- kubectl version –client



 # Update the apt package index and install packages needed to use the Kubernetes apt repository: sudo apt-get update

- sudo apt-get install -y apt-transport-https ca-certificates curl



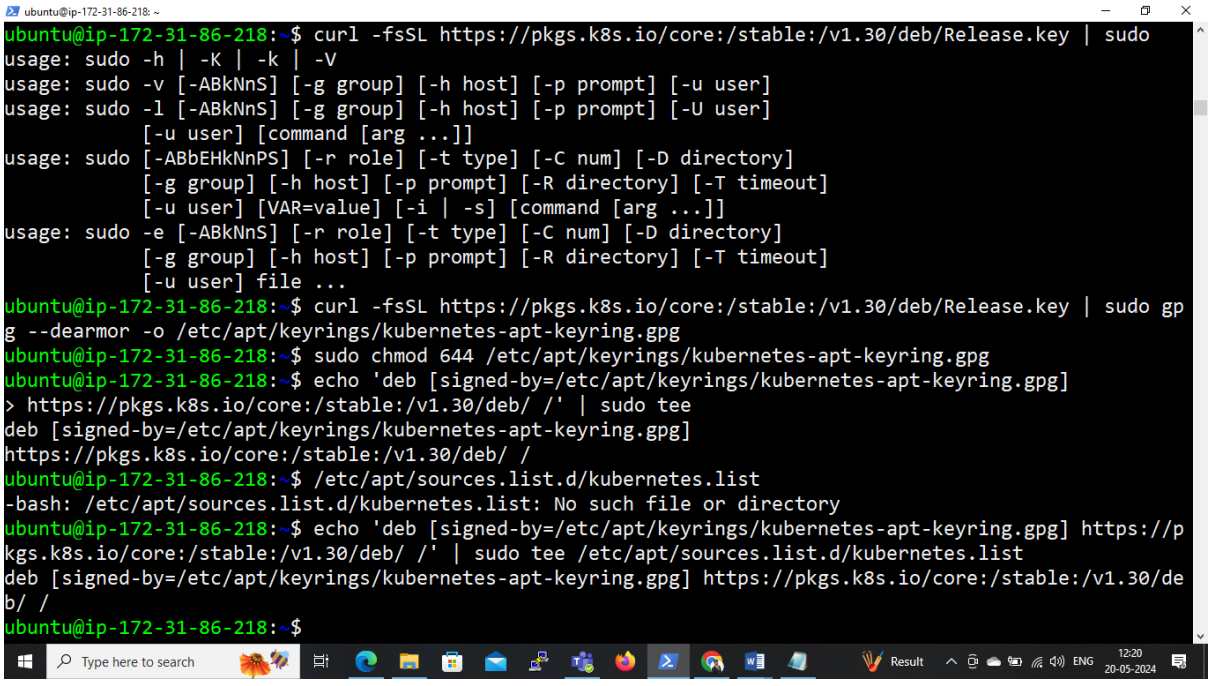
 # Download the public signing key for the Kubernetes package repositories. The same signing key is used for all repositories so you can disregard the version in the URL:

- curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
- sudo chmod 644 /etc/apt/keyrings/kubernetes-apt-keyring.gpg

 # Add the appropriate Kubernetes apt repository. If you want to use Kubernetes version different than v1.30, replace v1.30 with the desired minor version in the command below:

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

deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /

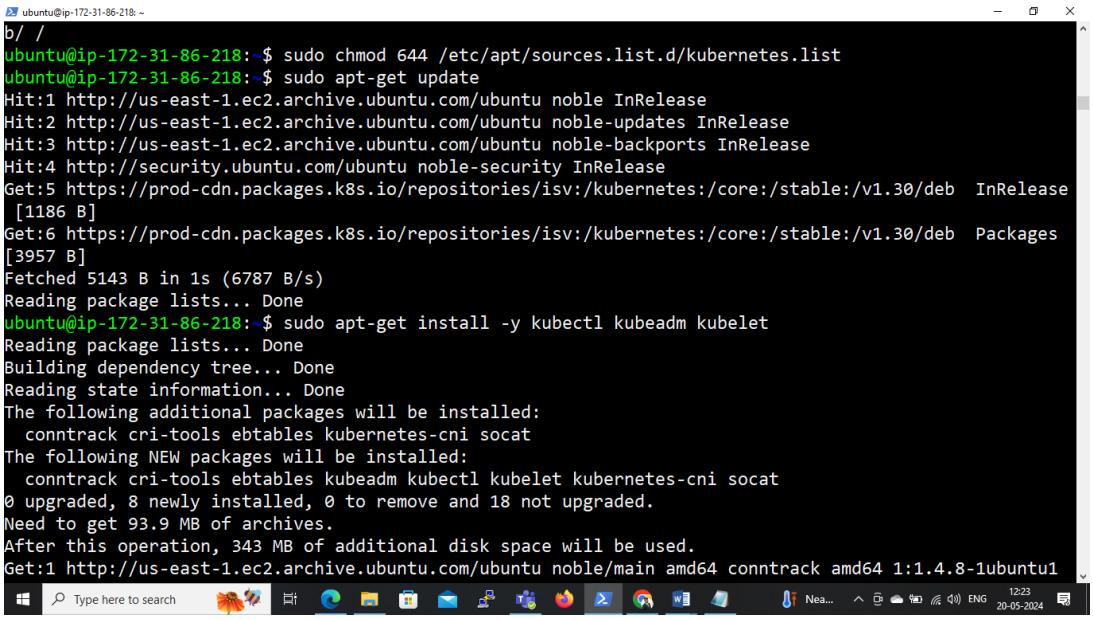


- sudo chmod 644 /etc/apt/sources.list.d/kubernetes.list

 # Update apt package index, then install kubectl, kubeadm and kubelet:

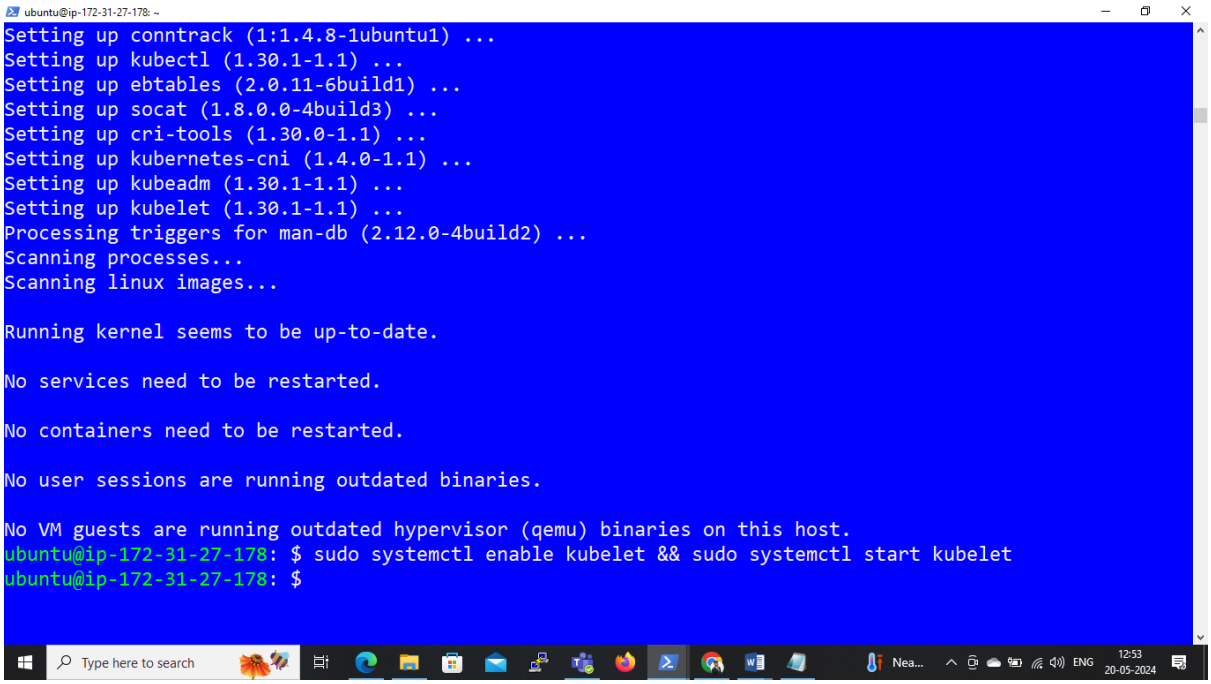
- sudo apt-get update

- sudo apt-get install -y kubectl kubeadm kubelet



 # Start the kublet service and enable it:

- sudo systemctl enable kubelet && sudo systemctl start kubelet



(Above all commands run on both – Master and Worker nodes)

On Master:

 # Initialize kubeadm

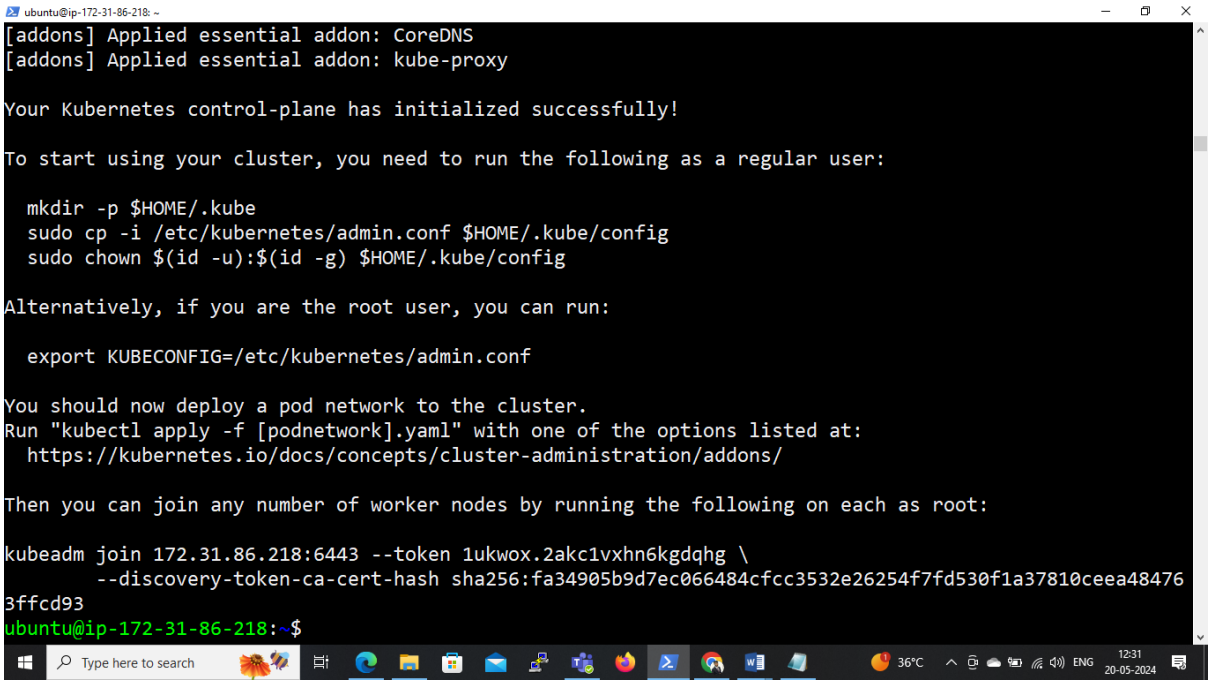
- sudo kubeadm init --ignore-preflight-errors=all

 # After running this command we get the kube api token which shows below:

kubeadm join 172.31.86.218:6443 --token 1ukwox.2akc1vxhn6kgdqhg \

--discovery-token-ca-cert-hash sha256:fa34905b9d7ec066484cfcc3532e26254f7fd530f1a37810ceea484763ffcd93

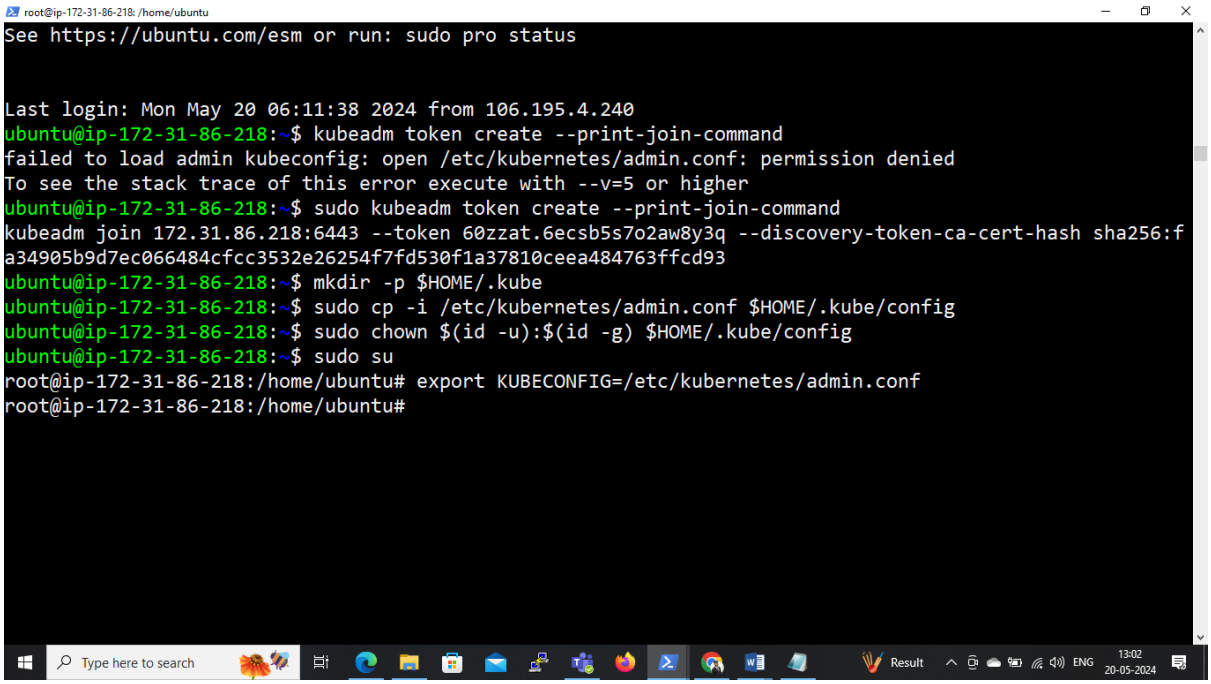
Save this token somewhere in our machine



Master node:-

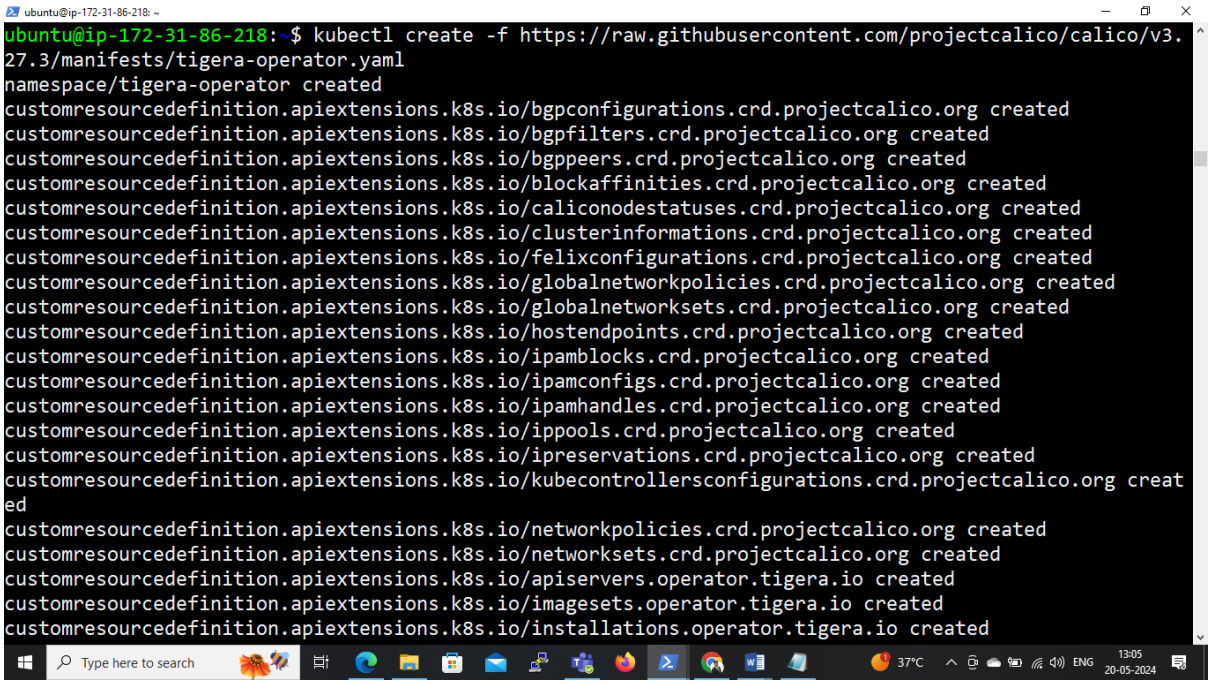
- mkdir -p $HOME/.kube
- sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
- sudo chown $(id -u):$(id -g) $HOME/.kube/config

- sudo su
- export KUBECONFIG=/etc/kubernetes/admin.conf
- exit



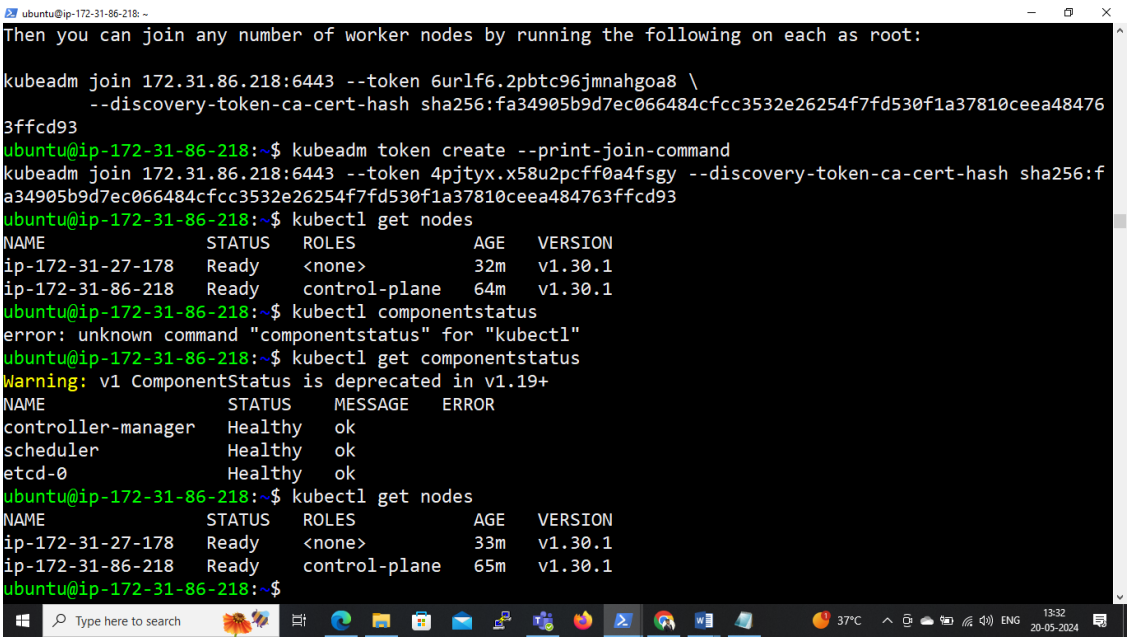
 # To install Calico on Master

- kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.27.3/manifests/tigera-operator.yaml
- kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.27.3/manifests/custom-resources.yaml



On master node check status & get nodes:

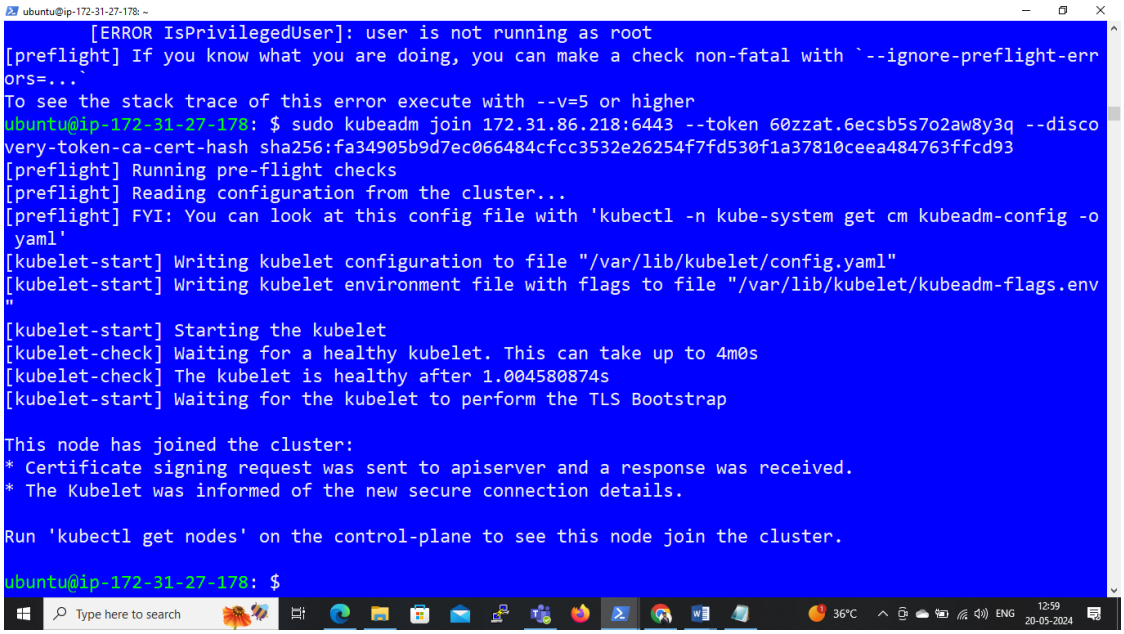
- kubectl get componentstatus
- kubectl get nodes



On Worker:

sudo kubeadm join 172.31.86.218:6443 --token 60zzat.6ecsb5s7o2aw8y3q --disco

very-token-ca-cert-hash sha256:fa34905b9d7ec066484cfcc3532e26254f7fd530f1a37810ceea484763ffcd93

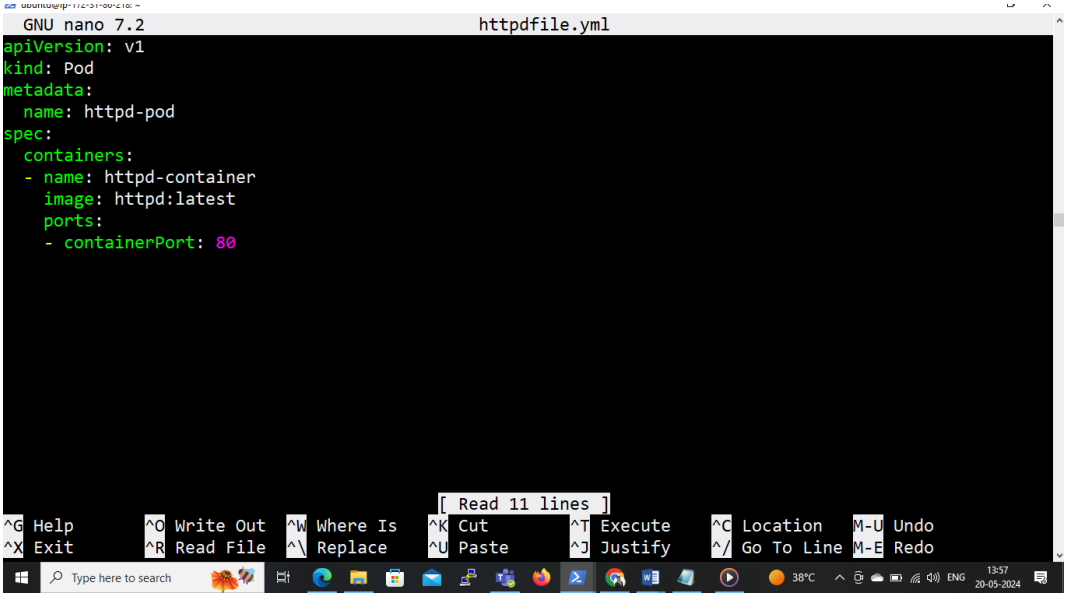


Q.3 Write a manifest file to create an httpd container in Pod, create pod using that manifest file. Also go inside that httpd container and create own pages and try to access those pages from inside the container.

On Master node :

 # Create a file ,

- **nano httpdfile.yml**

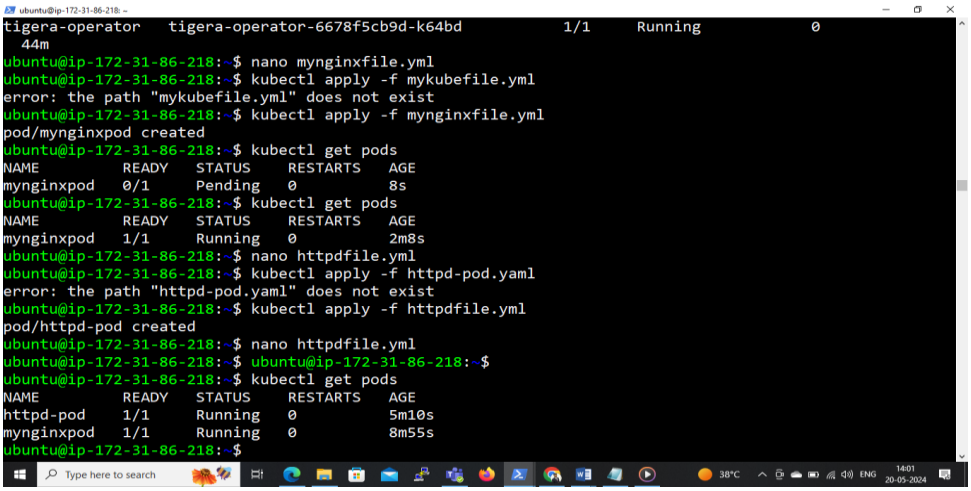


 # To create the Pod using this manifest file, we can run the following command:

- kubectl apply –f httpdfile.yml

 # Command to check the pods

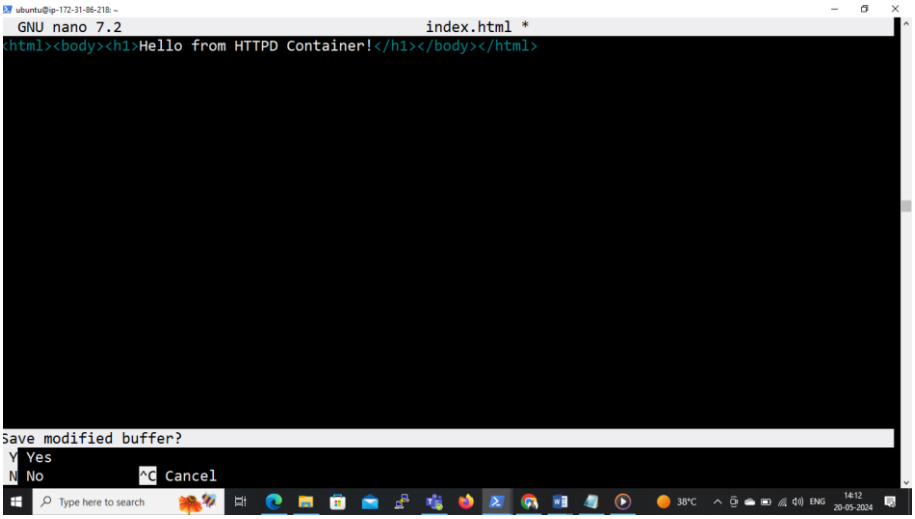
- kubectl get nodes



 # Once the Pod is running, We can enter in container & create HTML page.

- kubectl exec -it httpd-pod /bin/bash
- cd /usr/local/apache2/htdocs

- apt update
- apt install nano
- nano index.html



 # To access these pages from inside the container, we can use following command:

- apt install curl
- curl loalhost