T11 PRASAD AROTE 06

Lab Assignment 10

AIM: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms

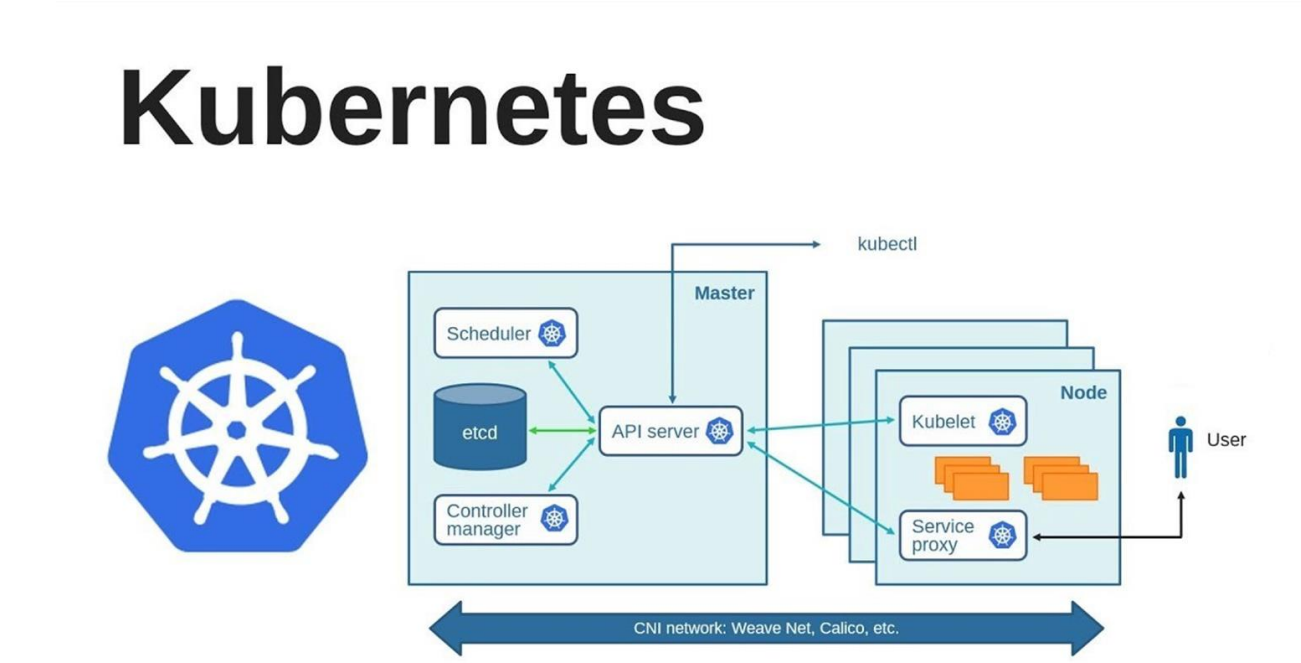
LO2. To deploy single and multiple container applications and manage application deployments with rollouts in Kubernetes.

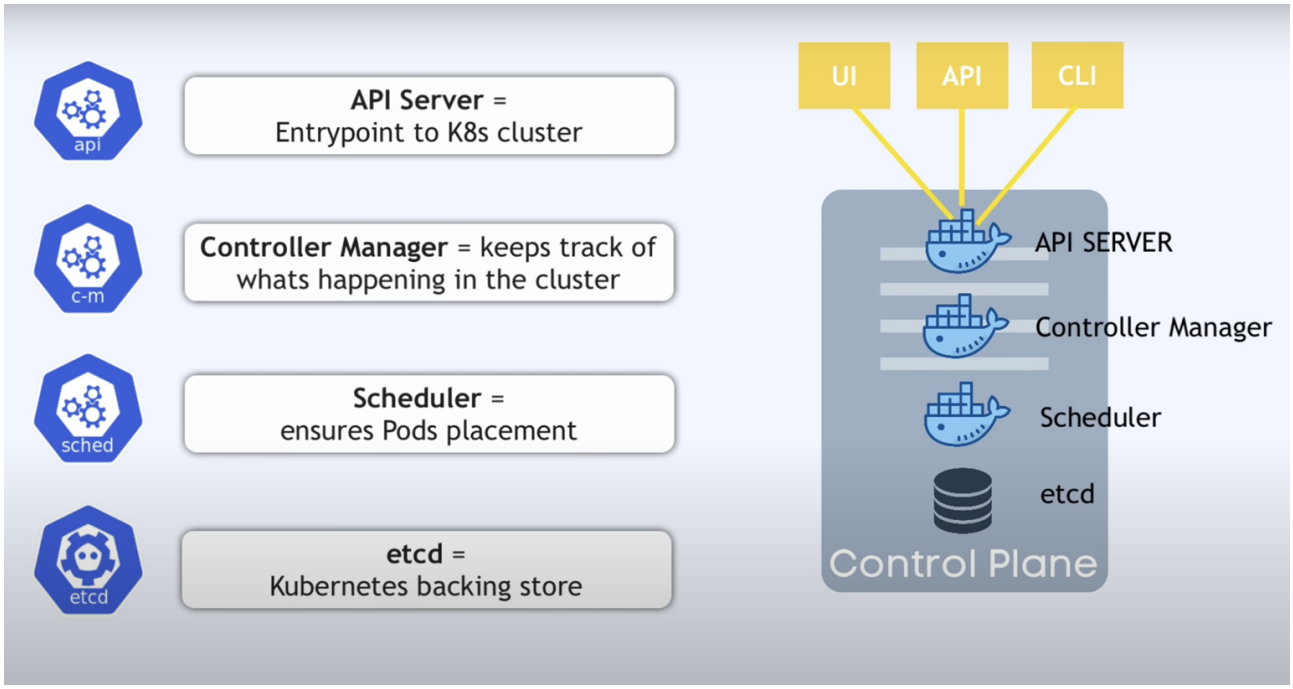
THEORY:

Kubernetes is an open-source container management tool that automates container deployment, scaling & load balancing.

It schedules, runs, and manages isolated containers that are running on virtual/physical/cloud machines.
All top cloud providers support Kubernetes.
One popular name for Kubernetes is K8s.

ARCHITECTURE





Working with Kubernetes

- We create a Manifest (.yml) file
- Apply those to cluster (to master) to bring it into the desired state.
- POD runs on a node, which is controlled by the master.

● Role of Master Node

- Kubernetes cluster contains containers running or Bare Metal / VM instances/cloud instances/ all mix.
- Kubernetes designates one or more of these as masters and all others as workers.
- The master is now going to run a set of K8s processes. These processes will ensure the smooth functioning of the cluster. These processes are called the ‘Control Plane.
- Can be Multi-Master for high availability.
- Master runs control plane to run cluster smoothly.

● Components of Control Plane

■ Kube-api-server → (For all communications)

- This api-server interacts directly with the user (i.e we apply .yml or .json manifest to kube-api-server)
- This kube-api-server is meant to scale automatically as per load.
- Kube-api-server is the front end of the control plane.

■ etcd

- Stores metadata and status of the cluster.
- etcd is a consistent and high-available store (key-value-store)

- Source of touch for cluster state (info about the state of the cluster)

→ etcd has the following features

1. Fully Replicated → The entire state is available on every node in the cluster.
2. Secure → Implements automatic TLS with optional client-certificate authentication.
3. Fast → Benchmarked at 10,000 writes per second.

■ Kube-schedular (action)

- When users request the creation & management of Pods, Kube-scheduler is going to take action on these requests.
- Handles POD creation and Management.
- Kube-scheduler match/assign any node to create and run pods.
- A scheduler watches for newly created pods that have no node assigned. For every pod that the scheduler discovers, the scheduler becomes responsible for finding the best node for that pod to run.
- The scheduler gets the information for hardware configuration from configuration files and schedules the Pods on nodes accordingly.

■ Controller-Manager

- Make sure the actual state of the cluster matches the desired state.

→ Two possible choices for controller manager—

1. If K8s is on the cloud, then it will be a cloud controller manager.
2. If K8s is on non-cloud, then it will be kube-controller-manager.

Components on the master that runs the controller

Node Controller → For checking the cloud provider to determine if a node has been detected in the cloud after it stops responding.

Route-Controller → Responsible for setting up a network, and routes on your cloud.

Service-Controller → Responsible for load Balancers on your cloud against services of type Load Balancer.

Volume-Controller → For creating, attaching, and mounting volumes and interacting with the cloud provider to orchestrate volume.

■ Nodes (Kubelet and Container Engine)

- Node is going to run 3 important pieces of software/process.

Kubelet

- The agent running on the node.
- Listens to Kubernetes master (eg- Pod creation request).
- Use port 10255.
- Send success/Fail reports to master.

Container Engine

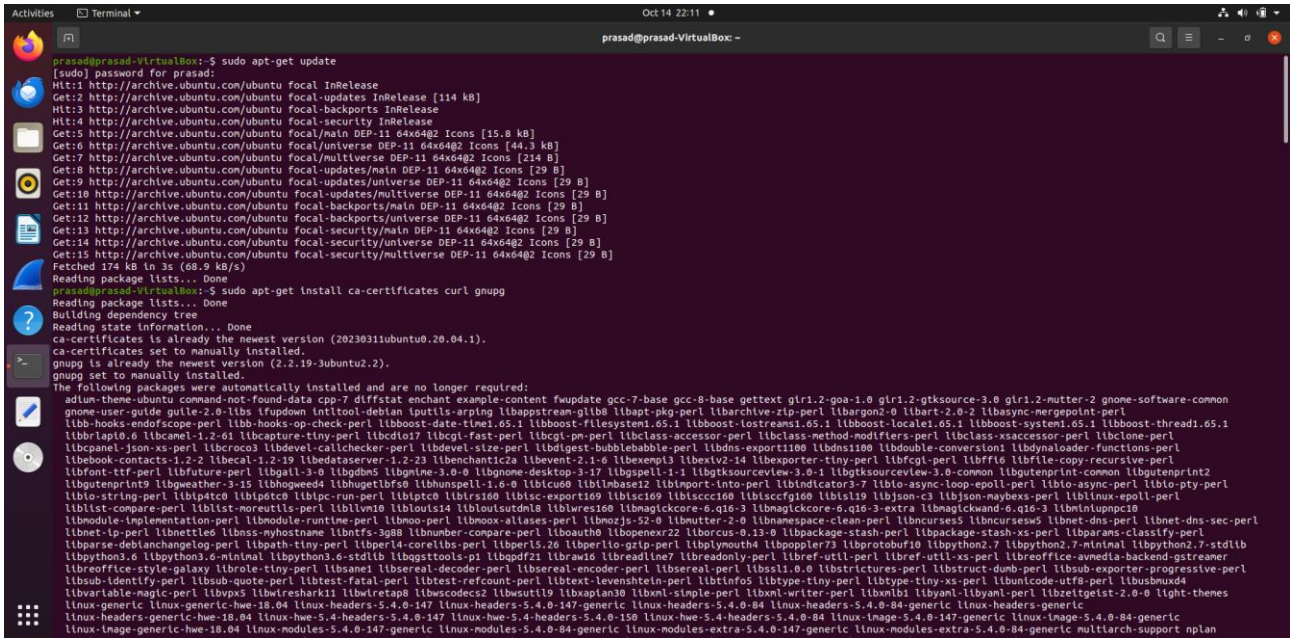
- Works with kubelet
- Pulling images
- Start/Stop Containers
- Exposing containers on ports specified in the manifest.

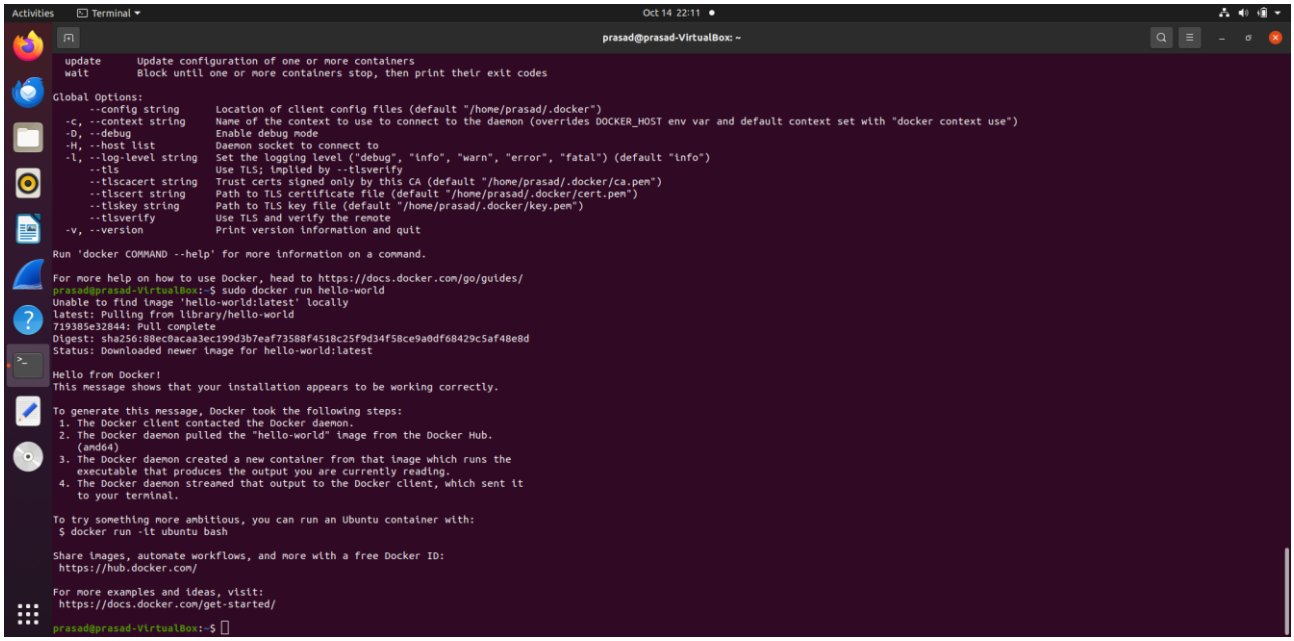
Kube-Proxy

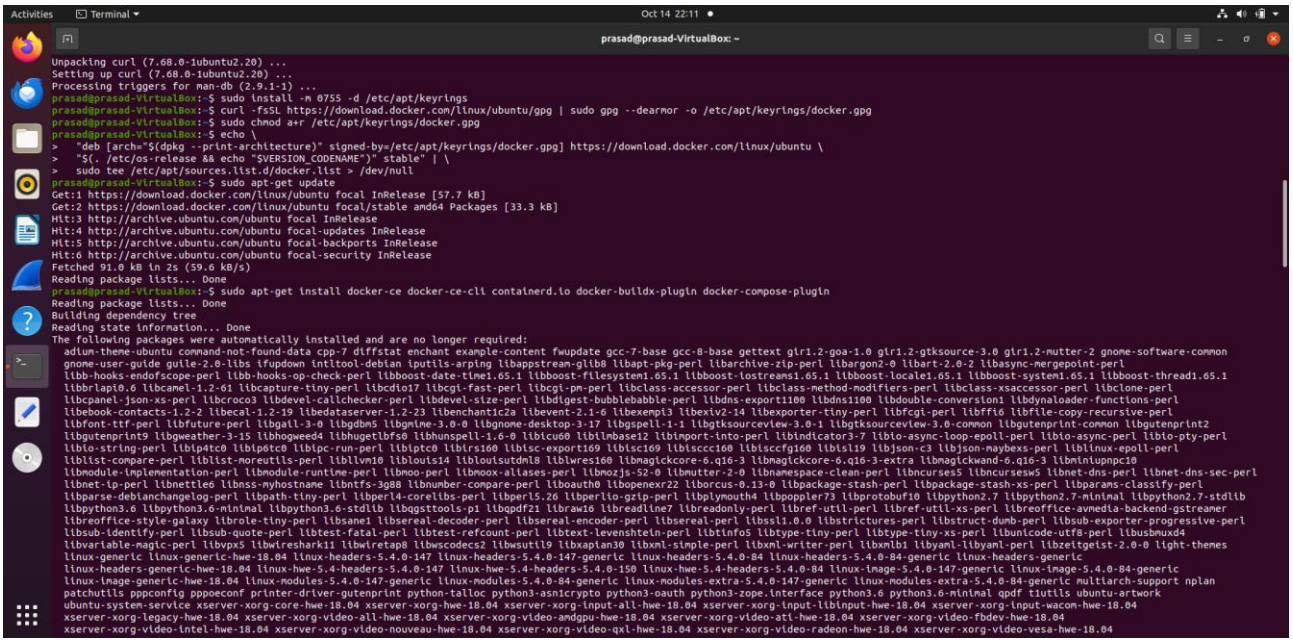
- Assign IP to each pod.
- It is required to assign IP addresses to Pods (dynamic)
- Kube-proxy runs on each node & this makes sure that each pod will get its unique IP Address.
- These 3 components collectively consist of ‘node’.

INSTALLATION:

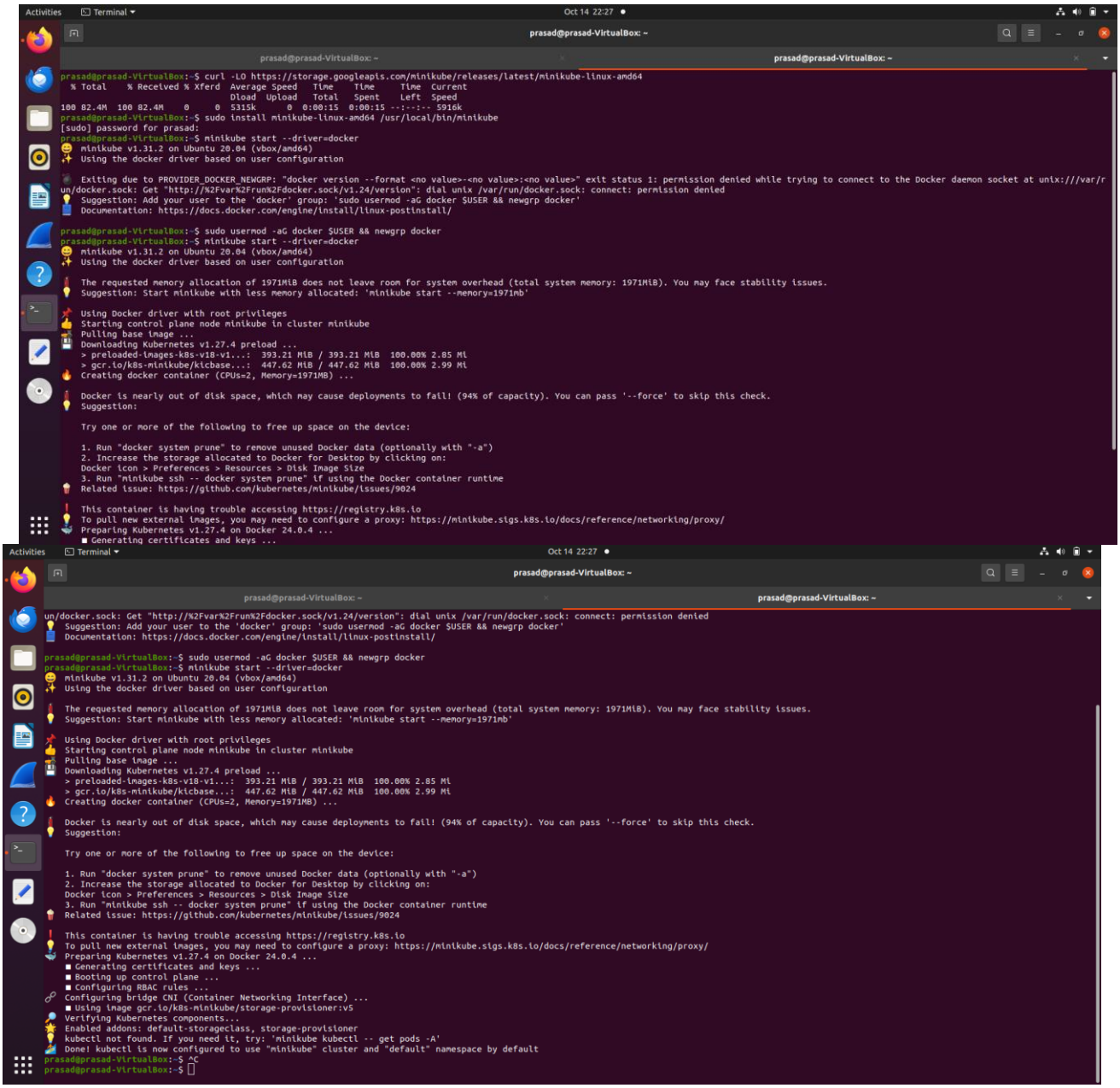
1. Install Docker



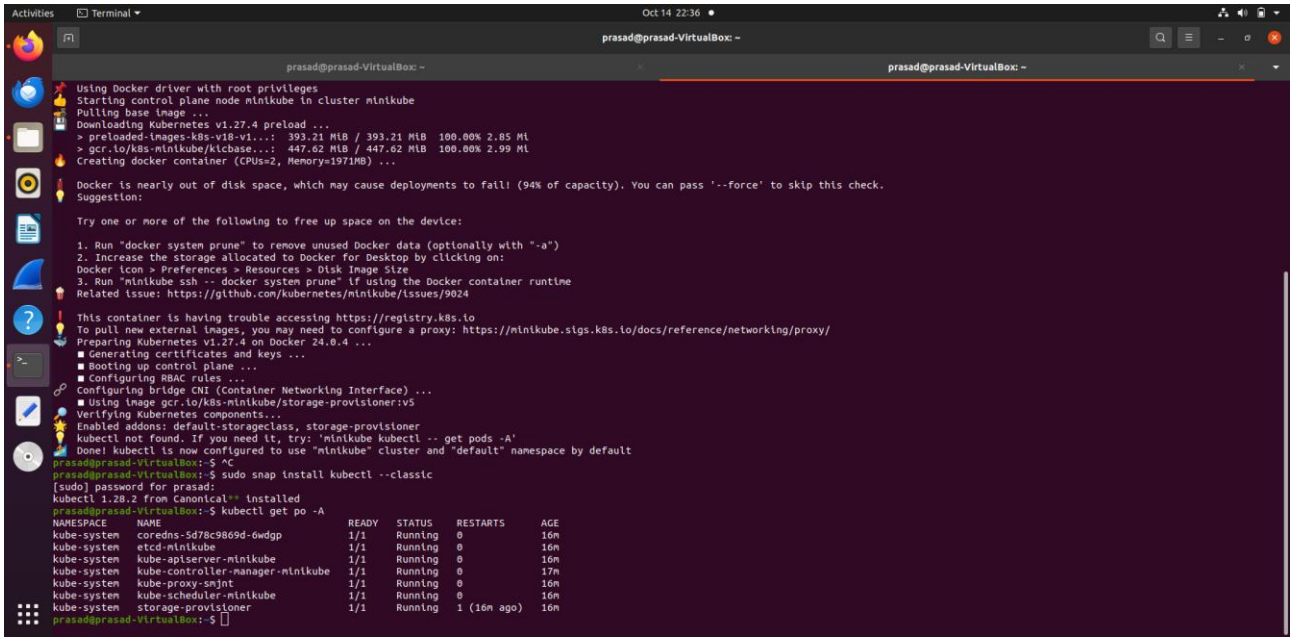




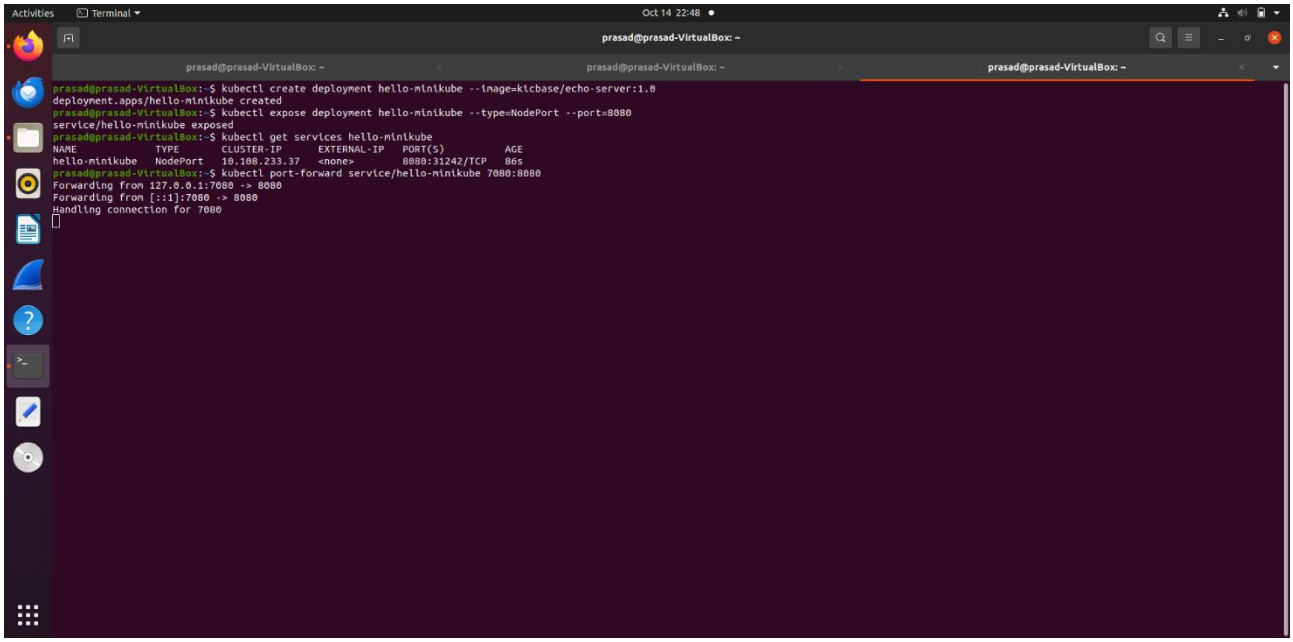
2. Install minikube using following commands

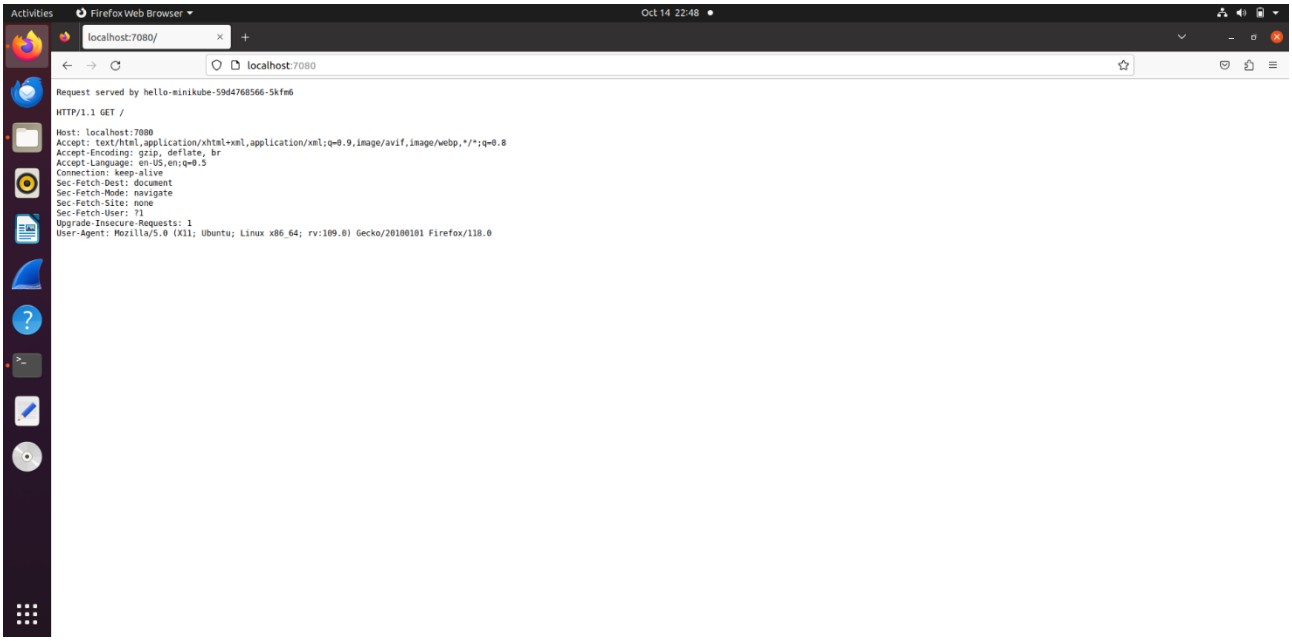


3. Install kubectl



4. Create a sample deployment.





CONCLUSION:

Here we studied Kubernetes cluster architecture in detail. Also we installed Kubernetes in ubuntu machine and created a sample deployment.