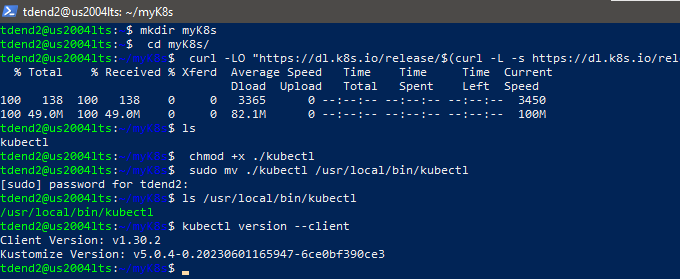
**Building a Kubernetes cluster**

**a. Section 1.3 Installing Minikube**

Installed kubectl

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



Installing minikube:

Minikube is a tool that makes it easy to run Kubernetes locally. Minikube runs a single-node

Kubernetes cluster inside Virtual Machine (VM) for users looking to try out Kubernetes or

develop with it day-to-day.

Minikube supports the following Kubernetes features:

• DNS

• NodePorts

• ConfigMaps and Secrets

• Dashboards

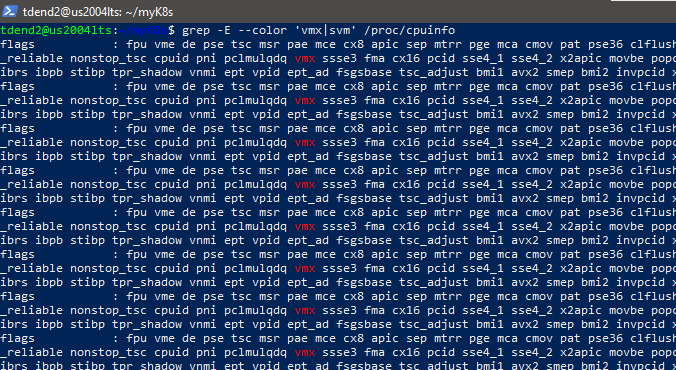
• Container Runtime: Docker, CRI-O, and containerd

• Enabling CNI (Container Network Interface)

• Ingress

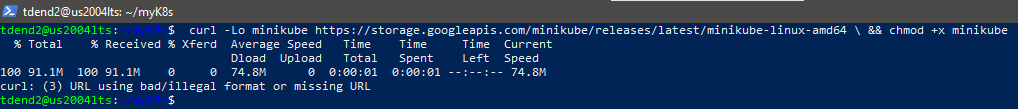
To check if virtualization is supported on Linux, ran the following command and verified that the

output is non-empty: *grep -E --color 'vmx|svm' /proc/cpuinfo*



Downloaded the binary and also make it executable.

curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 \ && chmod +x minikube



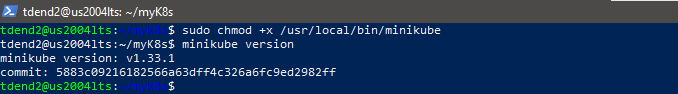


Moved the binary into local bin directory

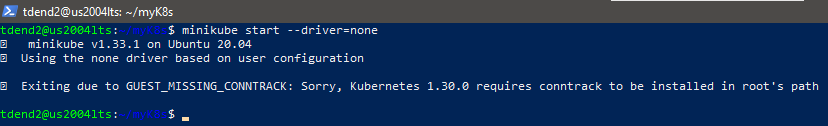




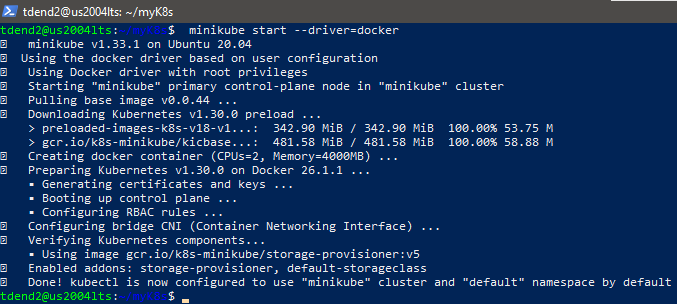
Checked the version of the minikube.



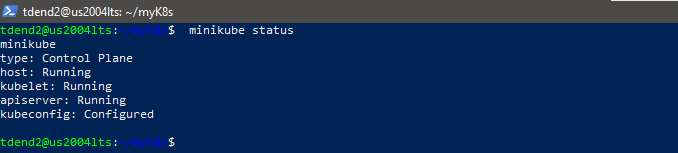
Ran the following command to start up a local Kubernetes cluster. We may use a virtual machine for nodes like the docker-machine in the previous exercise. However, we will use the ‘docker’ or ‘none’ driver to speed up starting containers and save some resources by using Docker containers instead of virtual machines. ‘none’ driver requires ‘root’.



we use a newer ‘docker’ driver:



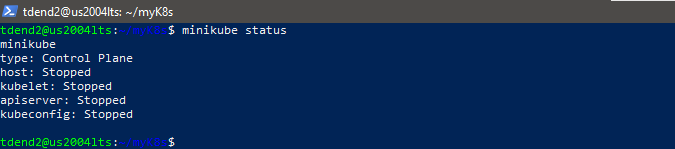
minikube status:



Stop cluster and recheck the status of your cluster:minikube stop



minikube status:

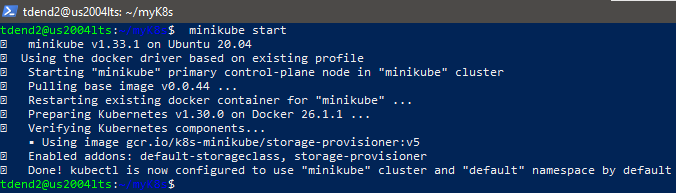


—----------------------------------------------------------------------------------------------------------------------------

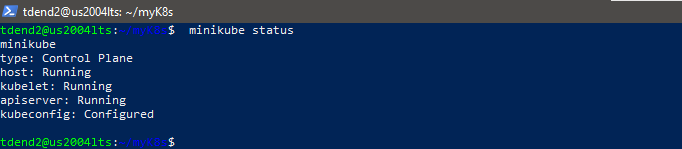
**b. Section 1.4 Installing Kubernetes with Minikube**

Started the cluster. They remember that we will use the ‘docker’ driver. Checked the status of the cluster.

*minikube start*

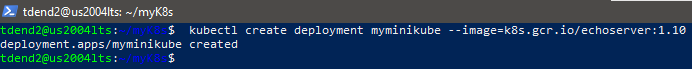


Cluster status:Minikube status



Interacted with cluster using kubectl. Let’s create a Kubernetes Deployment using an existing image named ‘echoserver’, a simple HTTP server, and expose it on port 8080 using --port.

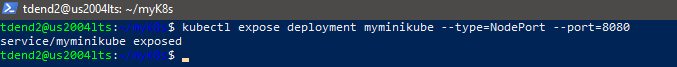
*kubectl create deployment myminikube --image=*[*k8s.gcr.io/echoserver:1.10*](http://k8s.gcr.io/echoserver:1.10)



To access the myminikube Deployment, expose it as a Service:

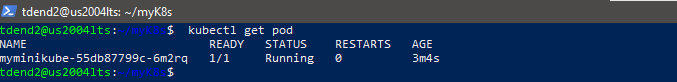
*kubectl expose deployment myminikube --type=NodePort -- port=8080*

The option --type=NodePort specifies the type of the Service.

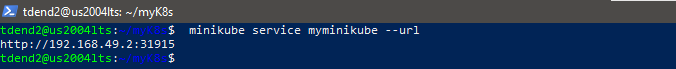
**

The myminikube Pod is now launched, but we must wait until the Pod is up before accessing it via the exposed Service.

Verified if the Pod is up and running

**

Got the URL of the exposed Service to view the Service details:



To be able to see the output in your browser on your laptop.We may need to use SSH Port Forwarding.

SSH port forwarding is a mechanism in SSH for tunneling application ports from the client machine to the server machine or vice versa.

Opened another NEW terminal and run the ssh command with some options like the below: (Note: Recommended you use two SSH terminals simultaneously: one for normal command-line connection, the other for SSH port forwarding.)

• SSH port forwarding option: -L

• Your laptop’s local port: 31915

• Your cluster’s IP and its port inside your host/docker VM: 192.168.49.2:31915

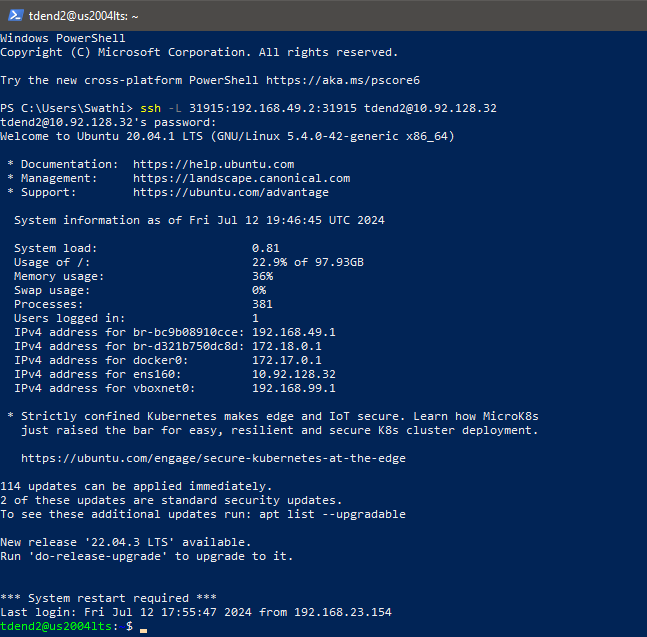
• Your host/docker VM’s login and IP: tdend2@10.92.128.46

LeeMBP15 is my laptop.

ssh -L 31915:192.168.49.2:31915 tdend2@10.92.128.32

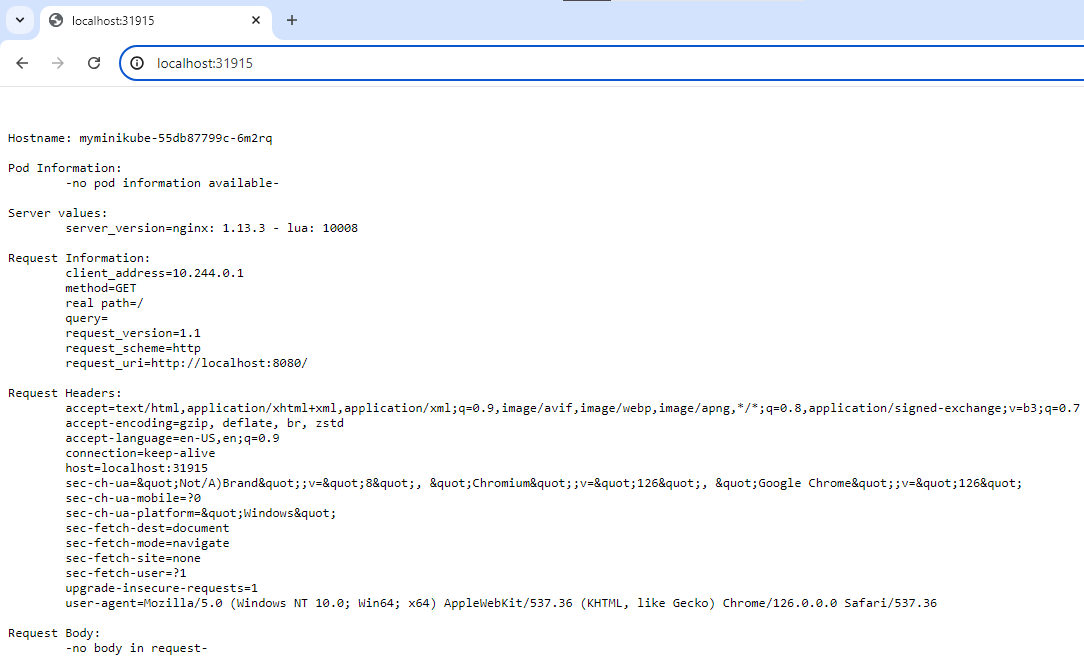
We need to keep the SSH terminal while using your browser.

In another SSH terminal:



Accessed the webpage in your browser with localhost and the port number

In browser : <http://localhost:31915/>



Hostname: myminikube-55db87799c-6m2rq

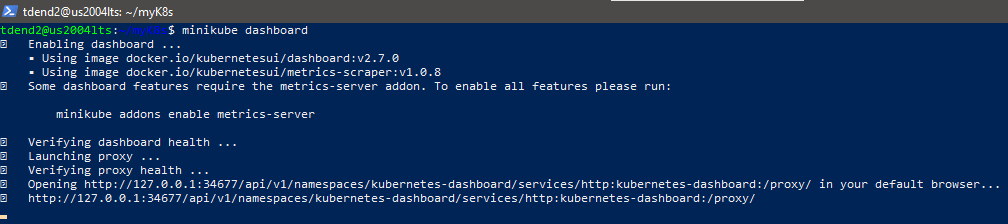
—----------------------------------------------------------------------------------------------------------------------------

**c. Section 1.5 Kubernetes Dashboard**

Creating Kubernetes Dashboard, a web-based Kubernetes user interface.

We can use Dashboard to deploy containerized applications to a Kubernetes cluster, troubleshoot containerized application, and manage the cluster resources. We can use Dashboard to get an overview of applications running on cluster, as well as for creating or modifying individual Kubernetes resources (such as Deployments, Jobs, DaemonSets, etc.). For example, we can scale a Deployment, initiate a rolling update, restart a pod or deploy new applications using a deploy wizard.

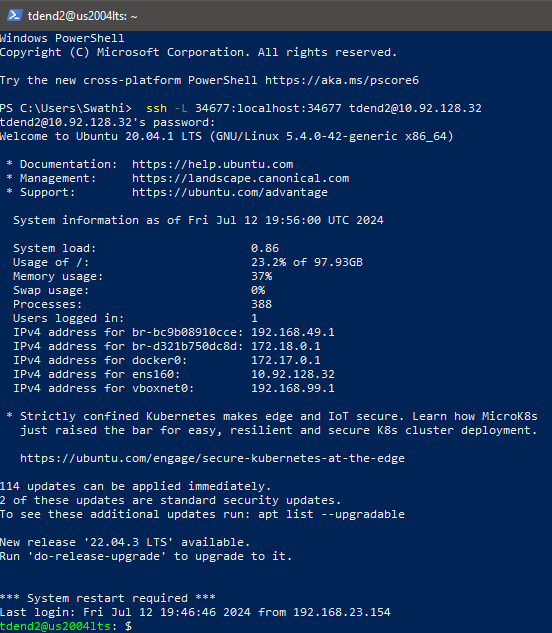
The dashboard also provides information on the state of Kubernetes resources in your cluster and any errors that may have occurred. See <https://kubernetes.io/docs/tasks/access-applicationcluster/web-ui-dashboard/>



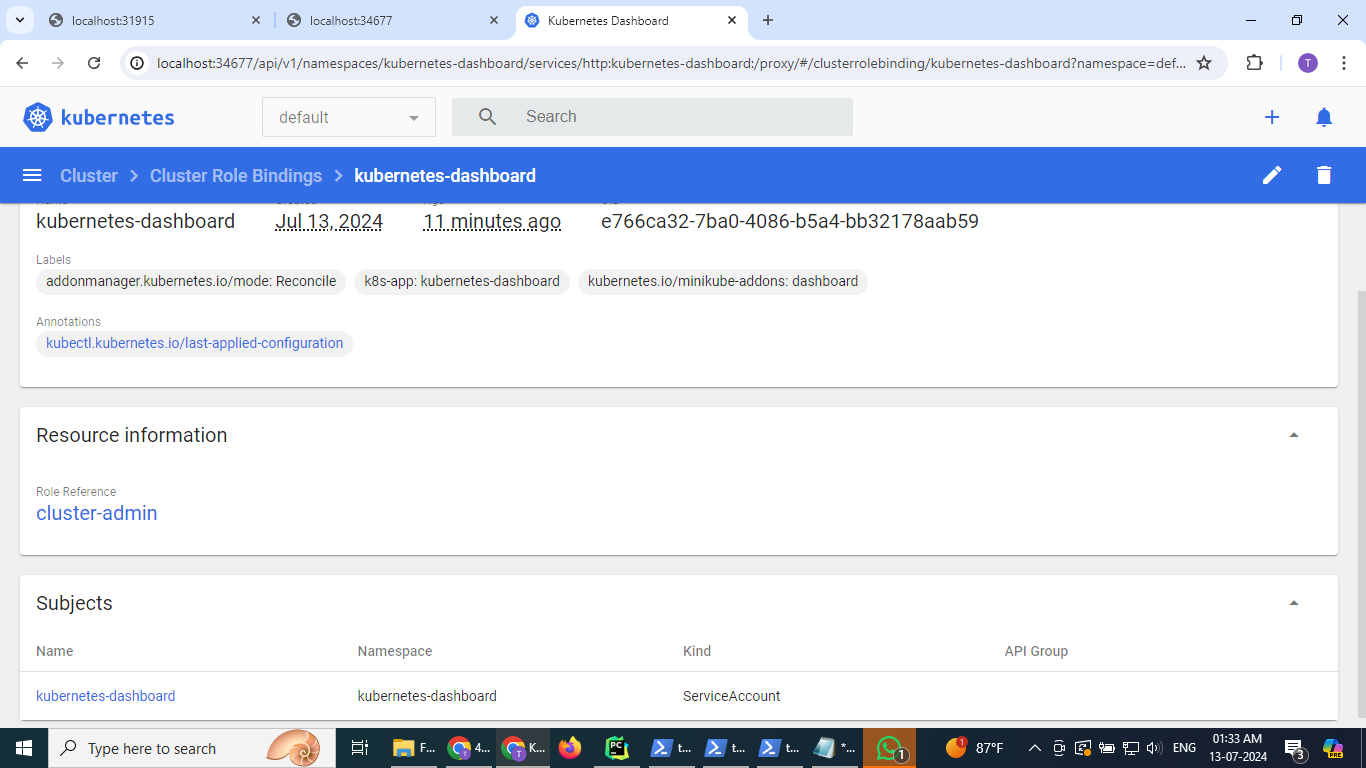
Note: Do not cancel the process by pressing ‘Ctrl + c’

To access the K8s Dashboard, we need the SSH port forwarding. Opened another NEW terminal and ran the ssh command with some options like the below:

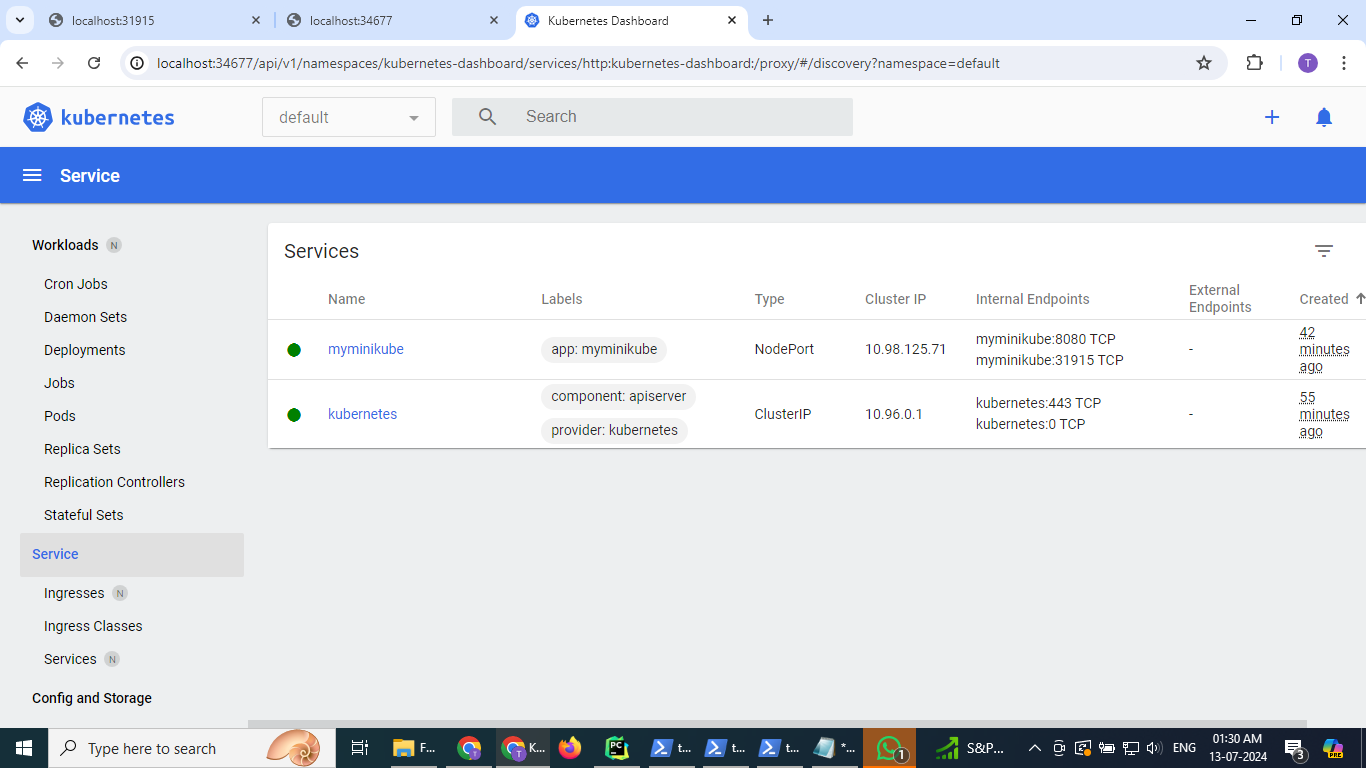
*ssh -L 34677:localhost:34677 tdend2@10.92.128.32*



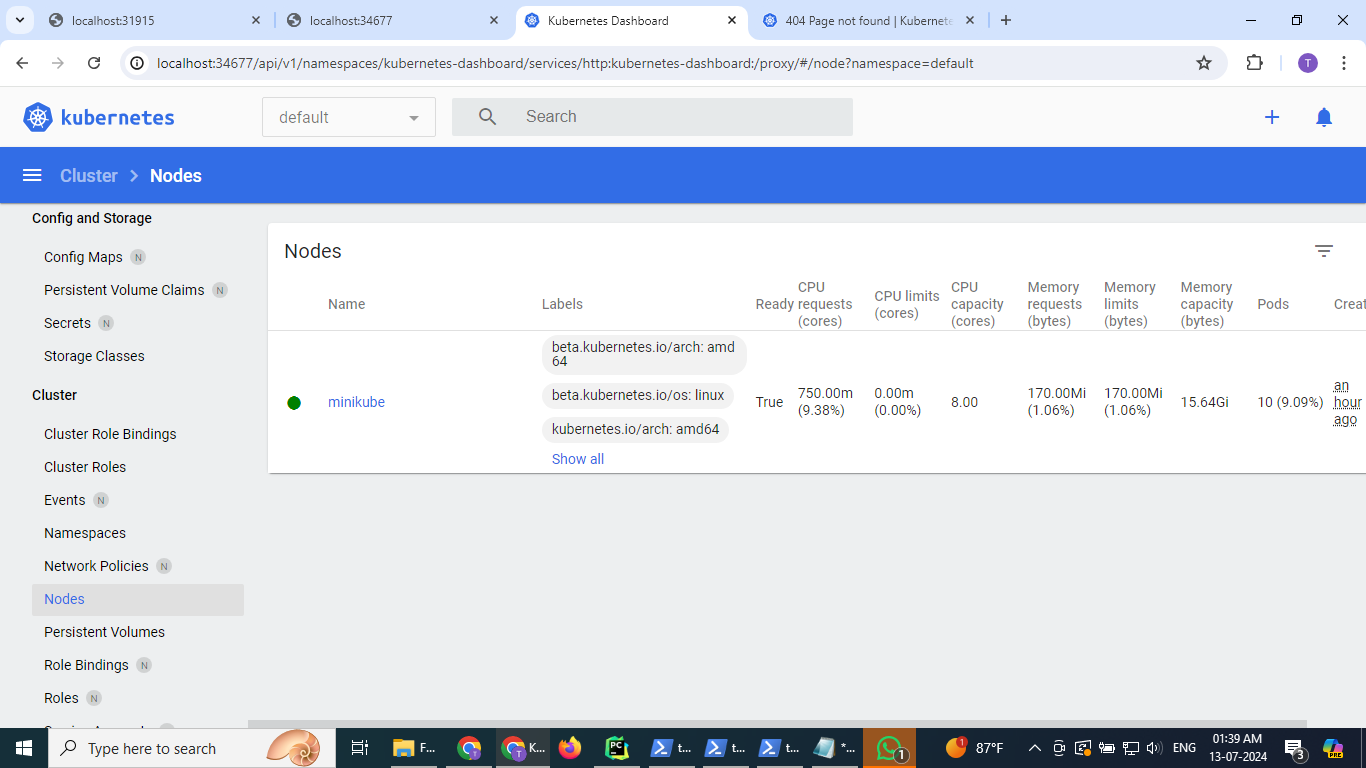
*Accessed the webpage in your browser. Note: Used my port number http://localhost:39071/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetesdashboard:/proxy/#/overview?namespace=default See screenshots on the next page. The interface design may differ from version to version.*



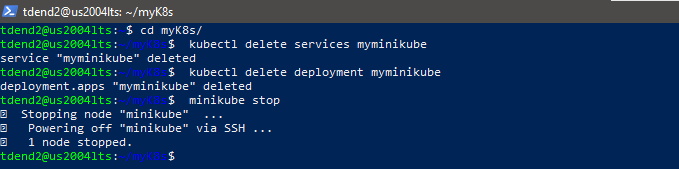
Kubernetes service, myminikube service are running and minikube node is running in the cluster as shown below and as seen from the kubernetes dashboard:



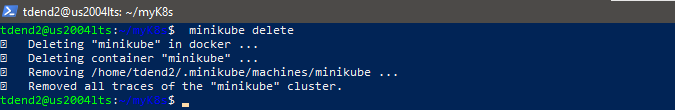
Nodes in cluster:



Deleted the ‘myminikube’ Service, deleted the ‘myminikube’ deployment,stopped cluster:



Deleted cluster:



=============================THE END==============================