Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 16

Aim : To learn to explore, scale, and update an application with Kubernetes

Scenario : In continuation, light and understanding of the previous practical done by you, where you learned to create a cluster, and deploy and debug an application- continue using the same application, or create a new application and hence perform the following tasks.

Tasks, Codes and Screenshots containing commands and output :

1. Learn about a Service in Kubernetes, and understand how labels and LabelSelector objects relate to a Service.

A Service in Kubernetes is a resource that enables load balancing and service discovery for a set of pods. Labels are key-value pairs assigned to pods, and a LabelSelector is used by a Service to determine which pods to include in the service. Labels and LabelSelectors are crucial for defining the pods associated with a Service, ensuring traffic is correctly directed to them.

2. Expose an application outside a Kubernetes cluster using a Service.

a) **knote.yaml** :

```
apiVersion: v1
kind: Service
metadata:
name: knote
spec:
selector:
app: knote
ports:
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 16

```yaml
- name: http
protocol: TCP
port: 3000
type: NodePort
---
apiVersion: apps/v1
kind: Deployment
metadata:
name: knote
spec:
replicas: 1
selector:
matchLabels:
app: knote
template:
metadata:
labels:
app: knote
spec:
containers:
- name: app
image: learnk8s/knote-js:1.0.0
ports:
- containerPort: 3000
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 16

```
env:
- name: MONGO_URL
value: mongodb://mongo:27017/dev
imagePullPolicy: Always
```

b) mongo.yaml :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: mongo-pvc
spec:
accessModes:
- ReadWriteOnce
resources:
requests:
storage: 256Mi
---
apiVersion: v1
kind: Service
metadata:
name: mongo
spec:
selector:
app: mongo
```

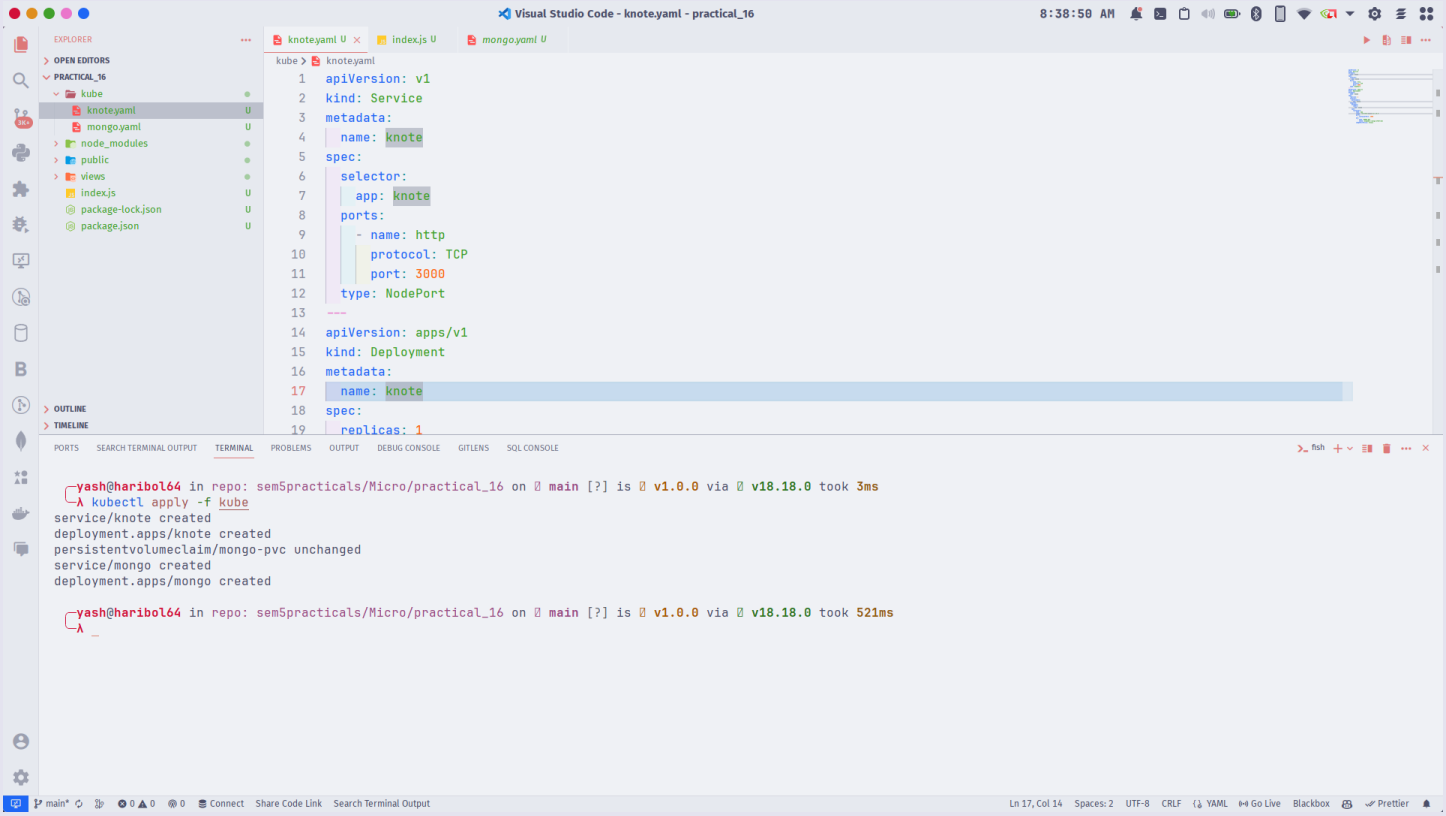
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 16

```yaml
ports:
- port: 27017
targetPort: 27017
---
apiVersion: apps/v1
kind: Deployment
metadata:
name: mongo
spec:
selector:
matchLabels:
app: mongo
template:
metadata:
labels:
app: mongo
spec:
containers:
- name: mongo
image: mongo:3.6.17-xenial
ports:
- containerPort: 27017
volumeMounts:
- name: storage
```

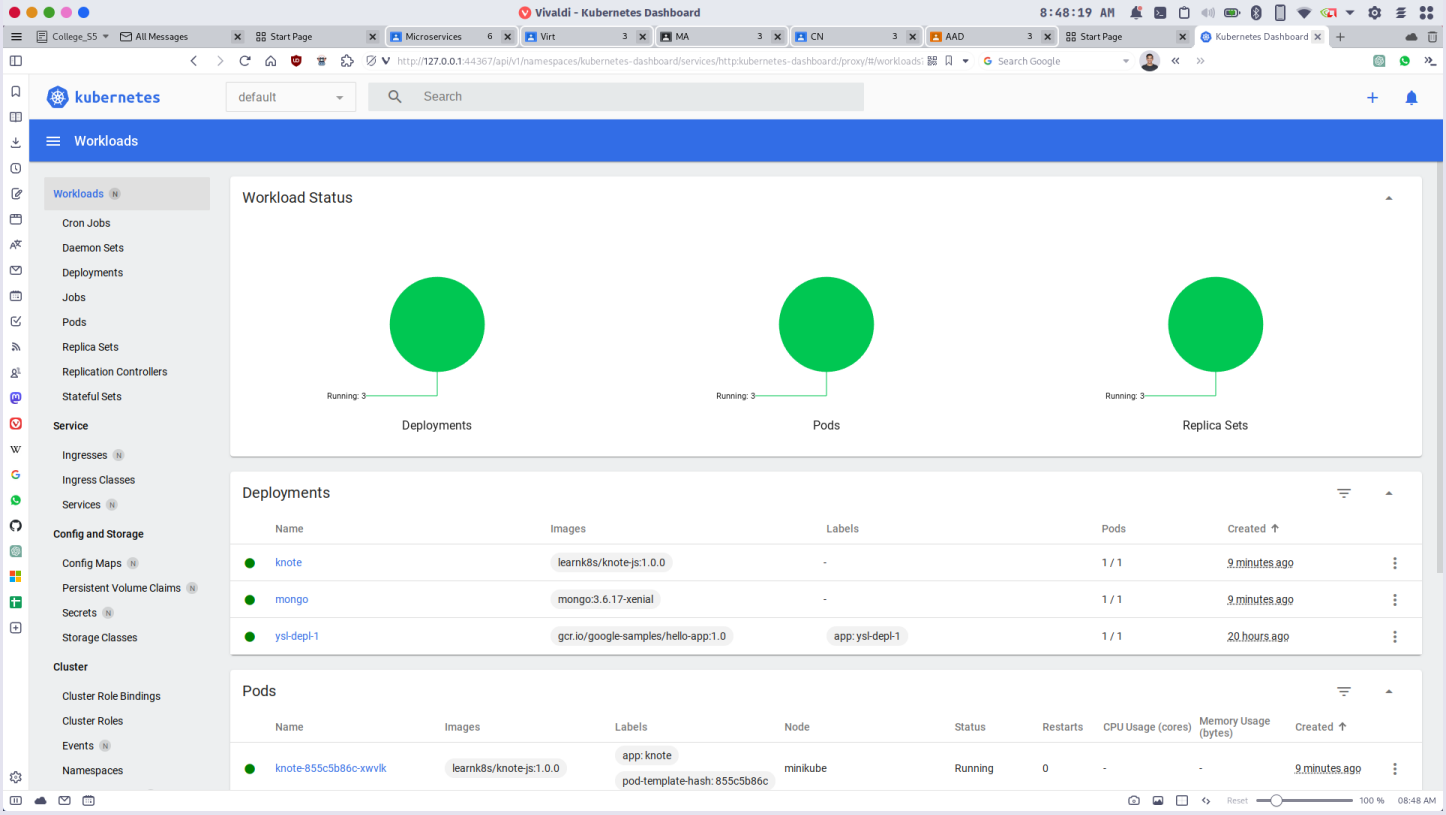
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 16

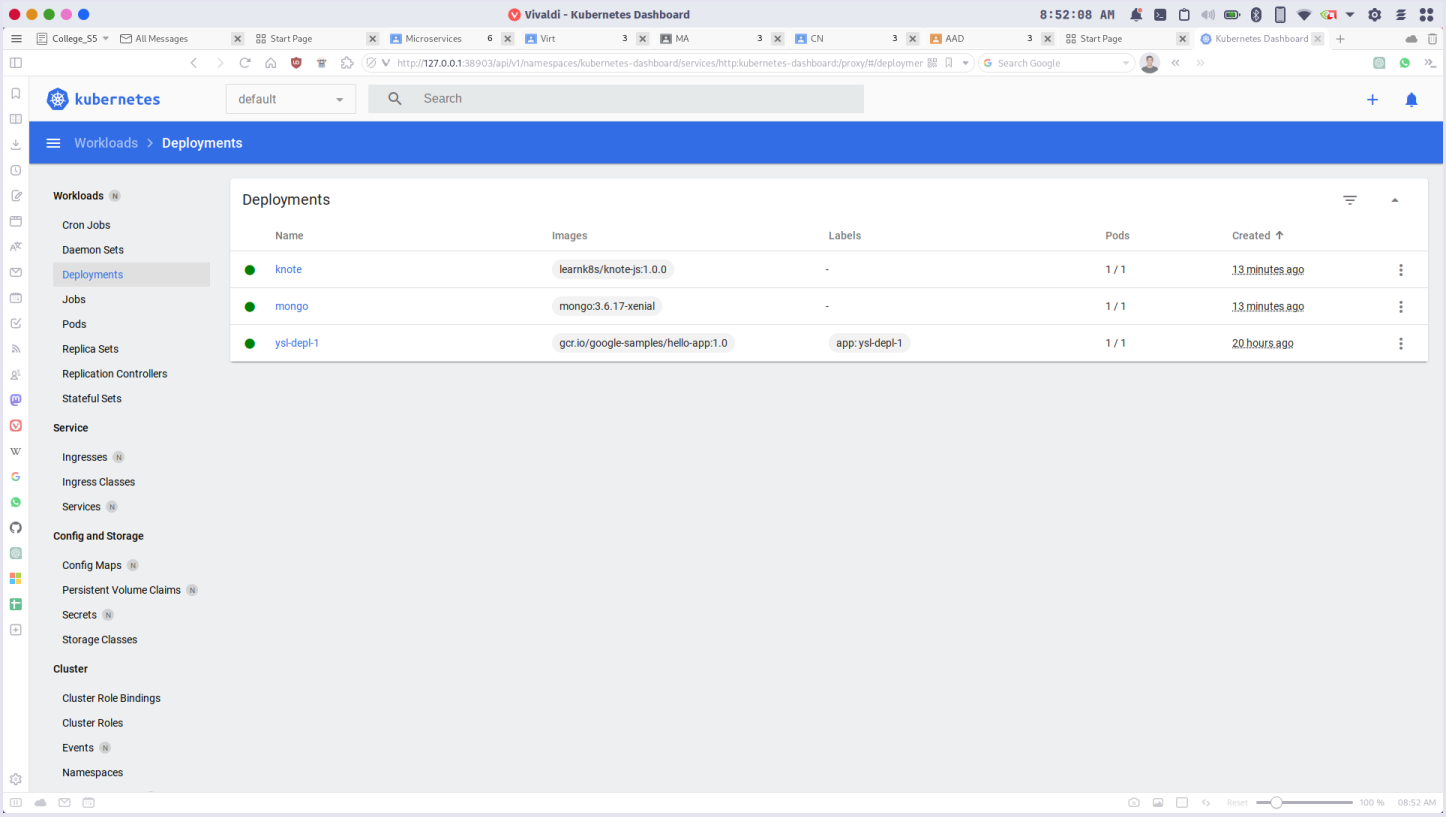
```yaml
mountPath: /data/db

volumes:

- name: storage

persistentVolumeClaim:

claimName: mongo-pvc
```

c) Now, create deployments of these two files inside **kube** folder using **kubectl** command and wait till all deployments, services, pods are fully created and in working state. It can be checked graphically using **minikube dashboard.**

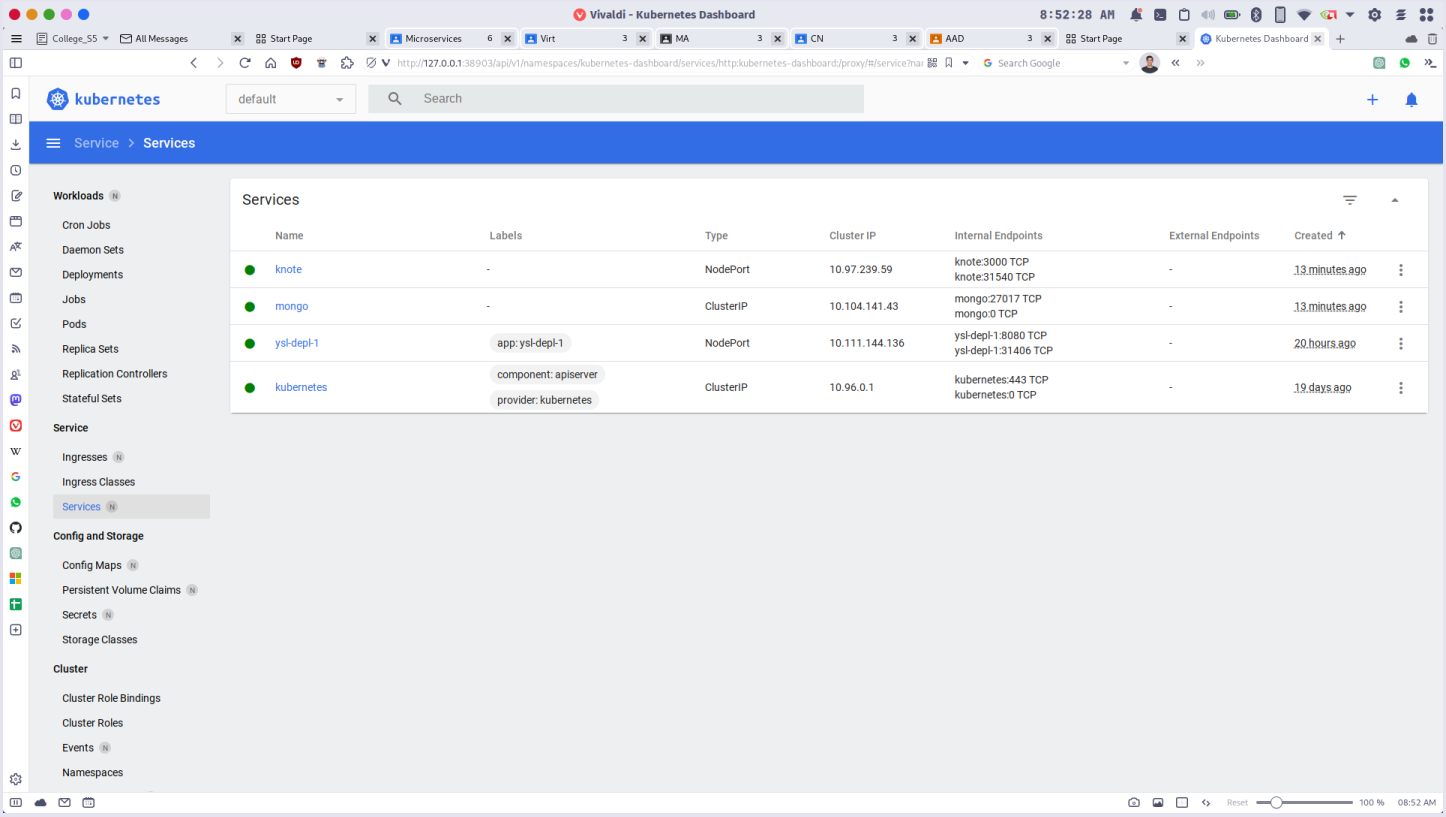


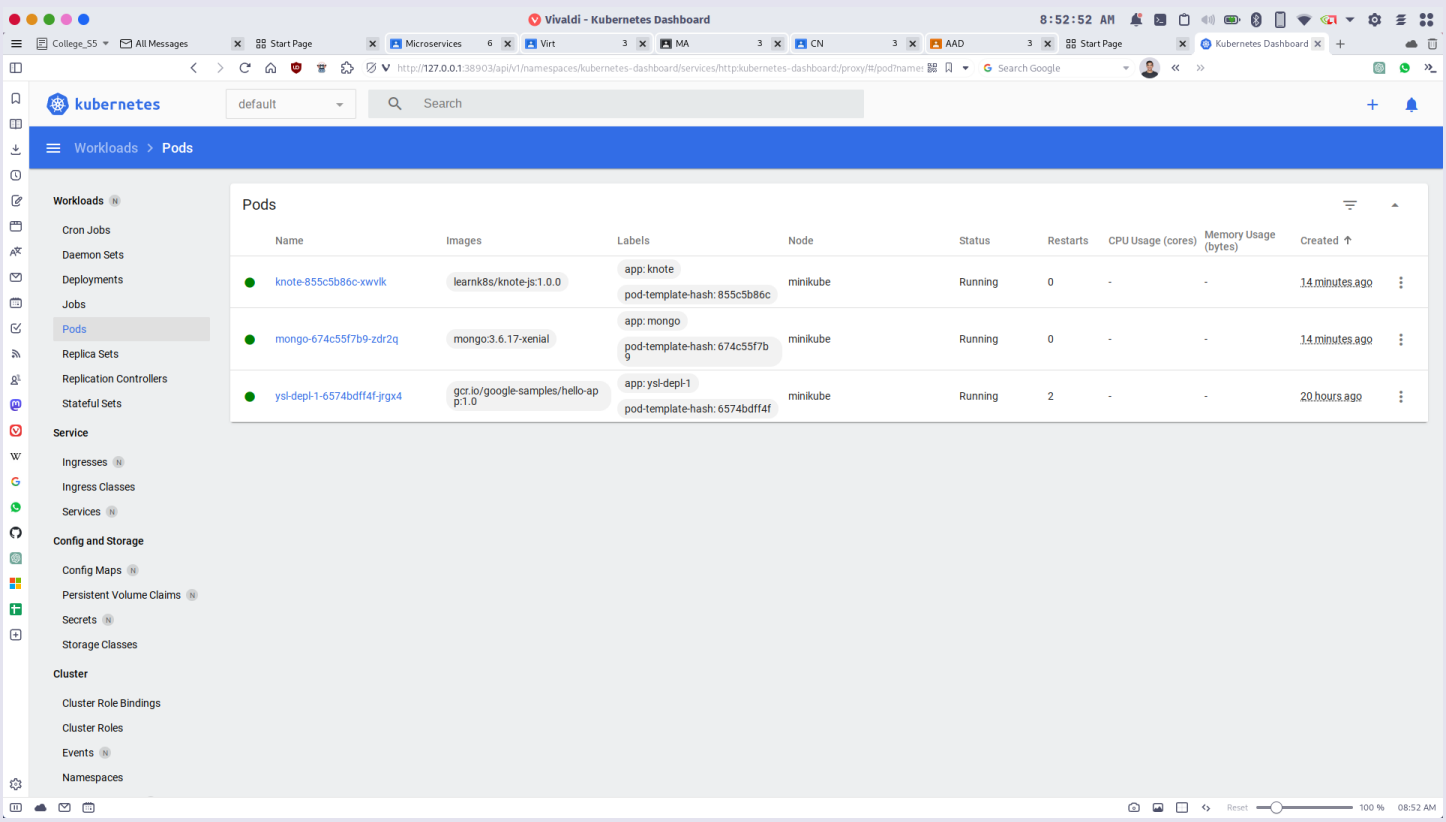
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 16





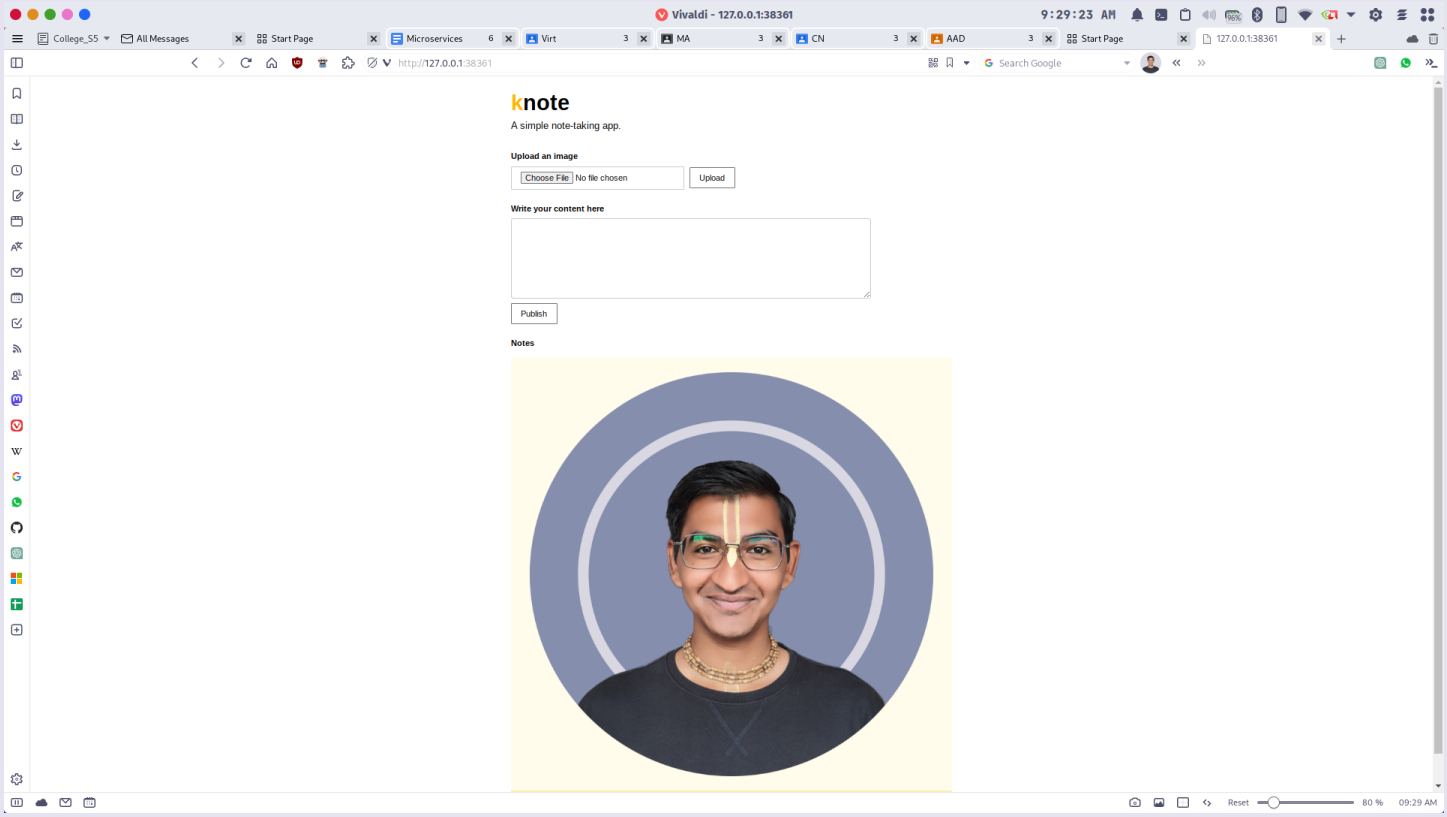
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 16





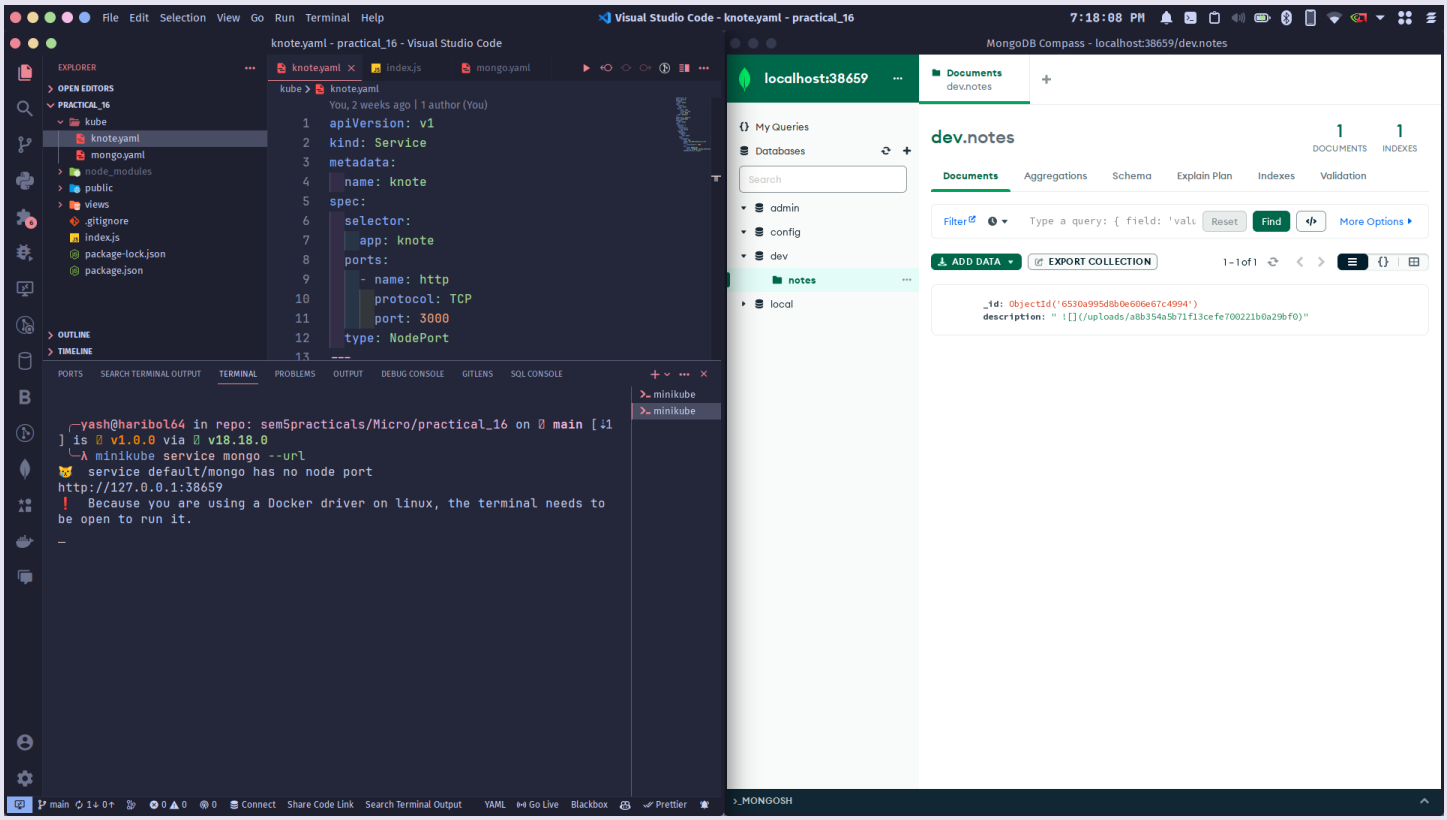
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 16

d) Run **minikube service knote --url** and on the URL given in command output, upload an image file.



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 16

e) Check the database connectivity and existence of the uploaded file via **minikube service mongo --url**



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Practical 16

3. Scale an app using kubectl.

The application can be OBJ OBJ OBJ OBJ OBJ scaled by editing replicas also. For example, to increase the replicas of the deployment knote, use command : **kubectl scale --replicas=3 deployment/knote**