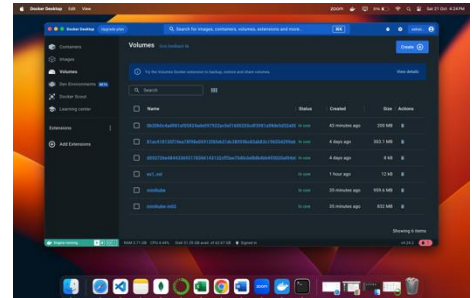


EXERCISE 1 – Images and Containers

Step 1:

```
(base) aakashsukre@Aakashs-MacBook-Air Project %  
docker volume create ex1_vol
```



The ex1_vol Docker volume is created by this command. In Docker containers, data is preserved via volumes.

Step 2:

```
(base) aakashsukre@Aakashs-MacBook-Air Project %  
docker run -it --name sender -v ex1_vol:/data ubuntu:22.04
```

docker run	Starts the container.
-it	This is used for running the container in interactive mode.
--name sender	This will give the name sender to the container.
-v ex1_vol:/data	Mounts the ex1_vol to the container data.
ubuntu:22.04	This is the image where we use.

Step 3:

```
root@44c3b29cc5e2:/data#  
echo "Hello, Welcome to Cloud Computing Project">Aakash_Sukre.txt  
exit
```

This is the command where we created a Aakash_Sukre.txt file with the content.

Step 4:

```
(base) aakashsukre@Aakashs-MacBook-Air Project %  
docker run -it --name receiver -v ex1_vol:/data:ro ubuntu:22.04
```

docker run	Starts the container.
-it	This is used for running the container in interactive mode.
--name <i>receiver</i>	This will gives the name receiver to the container.
-v ex1_vol:/data:ro	Mounts the ex1_vol to the container data with read only.
ubuntu:22.04	This is the image where we use.

Step 5:

```
root@134f01169904:/# cd /data  
  
root@134f01169904:/data# cat Aakash_Sukre.txt  
  
Hello, Welcome to Cloud Computing Project
```

Here, I used the cat command to display the content in the given file.

Summary:

The above commands give the knowledge about How to create the docker volumes, running the container in the interactive mode and using the mounted volume to share the content/data between the two containers. The receiver container reads the data from file which was created by the sender container.

EXERCISE 2 – Docker File

METHOD 1:

Step 1: Create Docker File

Docker file with the following tasks:

Ubuntu base image

FROM ubuntu

Copy install.sh file - container

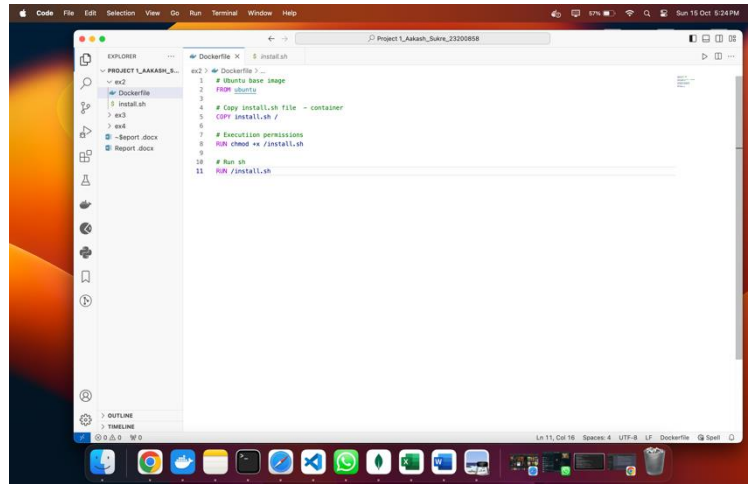
COPY install.sh /

Execution permissions

RUN chmod +x /install.sh

Run sh

RUN /install.sh



1. I have created the file Dockerfile.txt in the project directory.

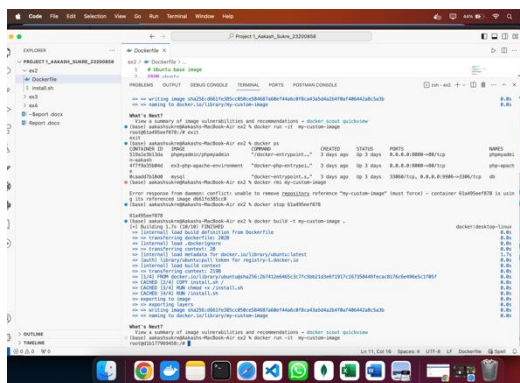
- Take the ubuntu as base image.
- Copy the install.sh file to the ex2 directory using **COPY** command.
- Give the execution permission to the install.sh shell file using **chmod +x**
- Run the install.sh file using **RUN** command.
- Save this install.sh file in the same directory where you created that Dockerfile.

2. **docker build -t my-custom-image .**

This command is used to build a custom docker image.

3. **docker run -it my-custom-image**

This command is used to run the created docker image.



Above command launch new container from the image and attached systems terminal to docker's terminal as indicated by -
root@d1b177969458:/#

Here we are in a container's my-custom-image terminal.

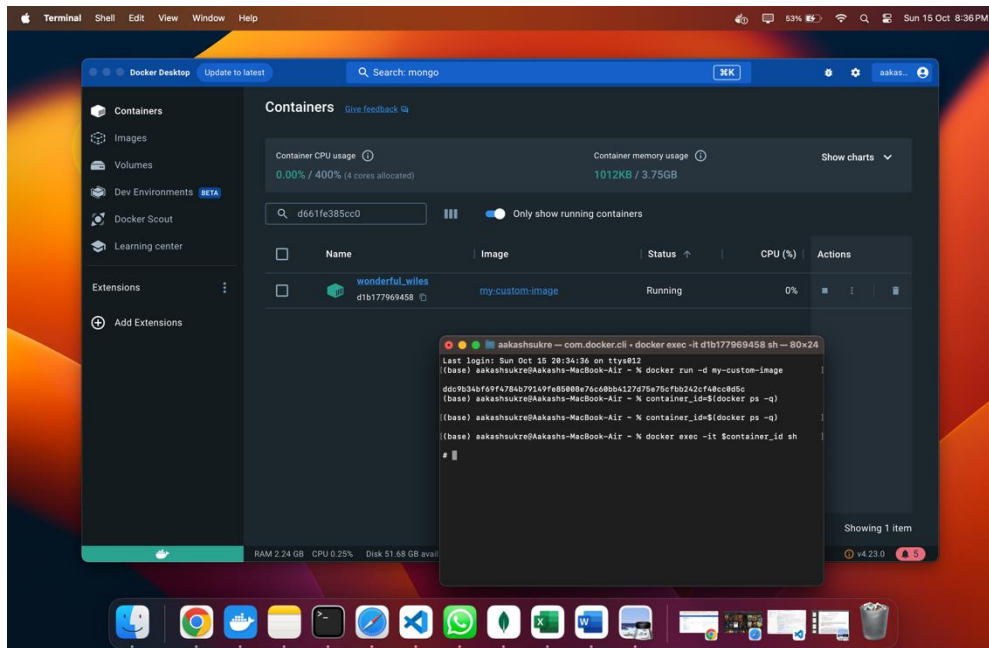
METHOD 2: using docker exec

Step 1:

```
docker run -d my-custom-image
```

step2:

```
container_id=$(docker ps -q)
```



Method 1

When you want to create a new container and start interacting with it, Method 1 (using `docker run -it`) is better. It's an easy approach to start a container and connect to its terminal.

Method 2

When you need to execute additional commands inside a running container, Method 2 (using `docker exec`) is helpful. When you didn't start a container with `-it` at the beginning, it's quite useful for attaching to an already-running one.

Summary:

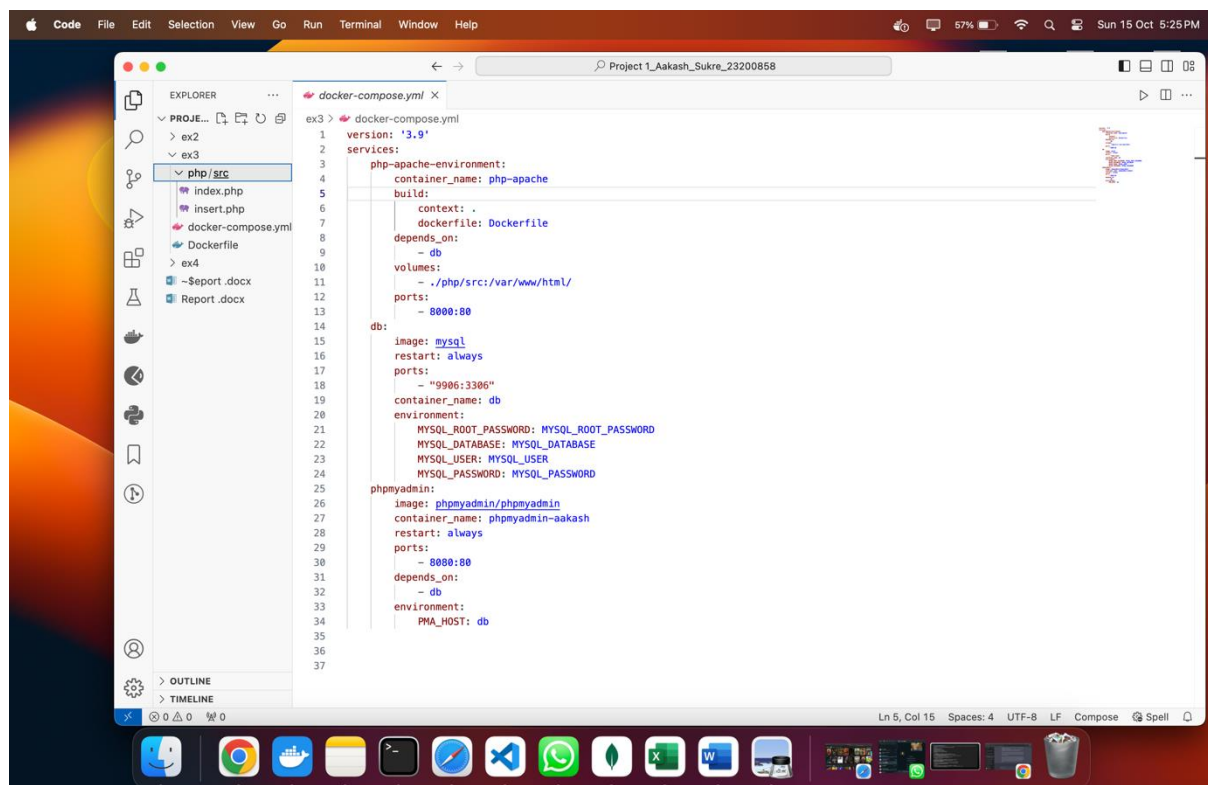
We can use these methods of container for various purposes such as Software Testing, Run Specific application in a controlled environment, Debugging and many more.

EXERCISE 3 – Dockerize a web application

Define MySQL Server.

Step 1:

1. Change and update the docker-compose.yml file in the ex3 directory.
2. Add a new MySQL server service definition in this file.
3. Update the phpMyAdmin in the docker-compose.yml file.



Let's understand the explanation of the docker-compose.yml file code.

```
db:
    # Name of Service.

    image: mysql
        # Service should use the MySQL image from Public repository.

    restart: always
        # restart the MySQL if it exits.

    ports:
        - "9906:3306"
            # Transform this port to host port on the container.

    container_name: db
    environment:
        MYSQL_ROOT_PASSWORD: MYSQL_ROOT_PASSWORD
        MYSQL_DATABASE: MYSQL_DATABASE
        MYSQL_USER: MYSQL_USER
        MYSQL_PASSWORD: MYSQL_PASSWORD

        # Set the Root Password, database, user and password.

phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: phpmyadmin-aakash
    restart: always
        # Restart the PHP Admin.

    ports:
        - 8080:80
            # Transform this port on host to port on the container.

    depends_on:
        - db
            # Specify the dependency on db Service. Ensure that Database is
            # started before the PHP Admin.

    environment:
        PMA_HOST: db
            # Sets the environment variable to the db for -
            # PhpMyAdmin make connection to the MySQL database.
```

Step 2:

(base) aakashsukre@Aakashs-MacBook-Air ex3 % docker-compose up

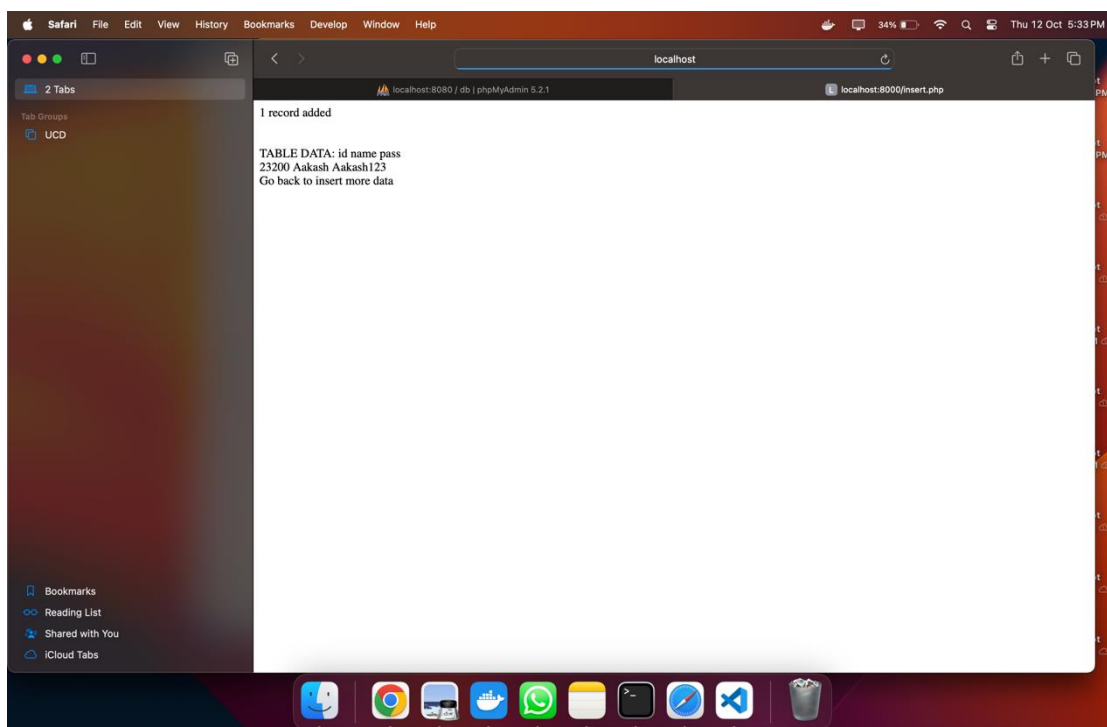
This command creates the php folder inside the ex3 directory.

Step3:

As the instruction provided in the assignment - I have added the index.php and insert.php file into the ex3/php/src folder which were created by the above step2 command.

Step4:

Insert data into the database.



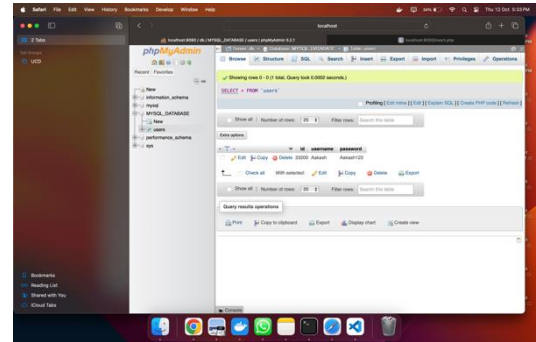
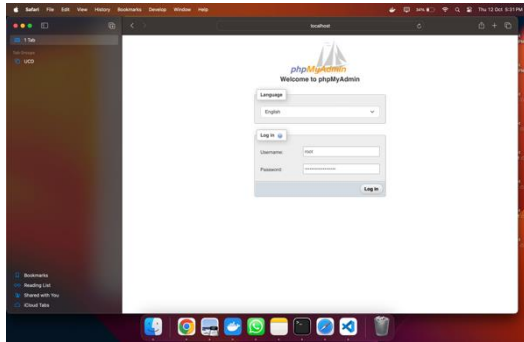
Step 5:

http://localhost:8080/index.php?route=/sql&db=MYSQL_DATABASE&table=users&pos=0

Check the data which was inserted by the above step by using the given:

Username = root

Password = MYSQL_ROOT_PASSWORD



Summary:

I have defined the MySQL server and changed the PHP Admin service in Docker-Compose file with these above changes. The PHP Admin service is set up to operate with the MySQL database, whereas the MySQL service uses the MySQL image and provides the required environment variables.

EXERCISE 4 – Kubernetes – Scale up your application.

Step 1:

I have installed the minikube.

Step 2:

1. Copy the file which were given in the assignment in to the ex4 directory.
2. `minikube start --nodes 2`

It will execute and create two nodes in the Docker Container for testing purpose :- `hello-deployment.yaml`

- ### 3. `cat hello-deployment.yaml`

This command shows the data of the file.

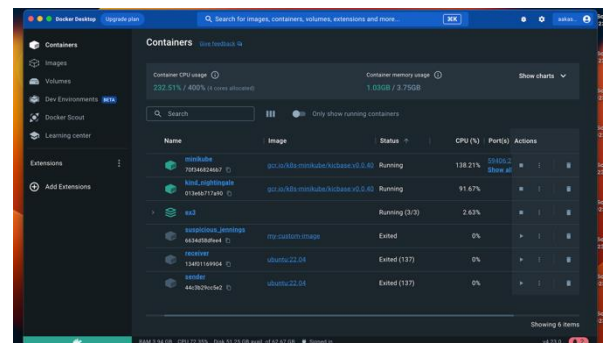
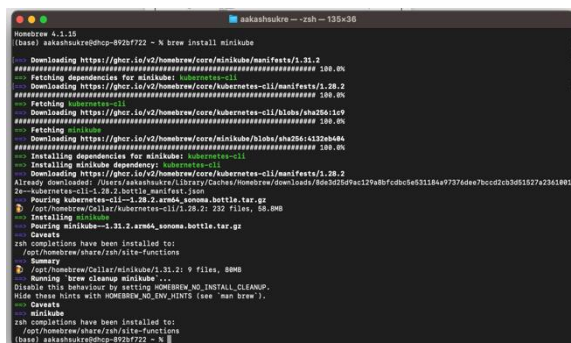
1. **kind** - Type of Kubernetes resource i.e. Deployment.
2. **spec.replicas** - How many identical pods the deployment for managing the load balance.
3. **spec.strategy** – define update strategy for replacing old pods with new.
4. **spec.template.spec.affinity** – Allow for advanced pod scheduling based on the attributes of node.
5. **spec.template.spec.containers** – define the container and their settings, including the container image, variables environment and ports, to be run within the pods created by the deployment.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 100%
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions: [ { key: app, operator: In, values: [hello] } ]
              topologyKey: "kubernetes.io/hostname"
  containers:
    - name: hello-from
      image: pbitty/hello-from:latest
      ports:
        - name: http
          containerPort: 80
      terminationGracePeriodSeconds: 1
```

Step 3:

```
kubectl apply -f hello-deployment.yaml
```

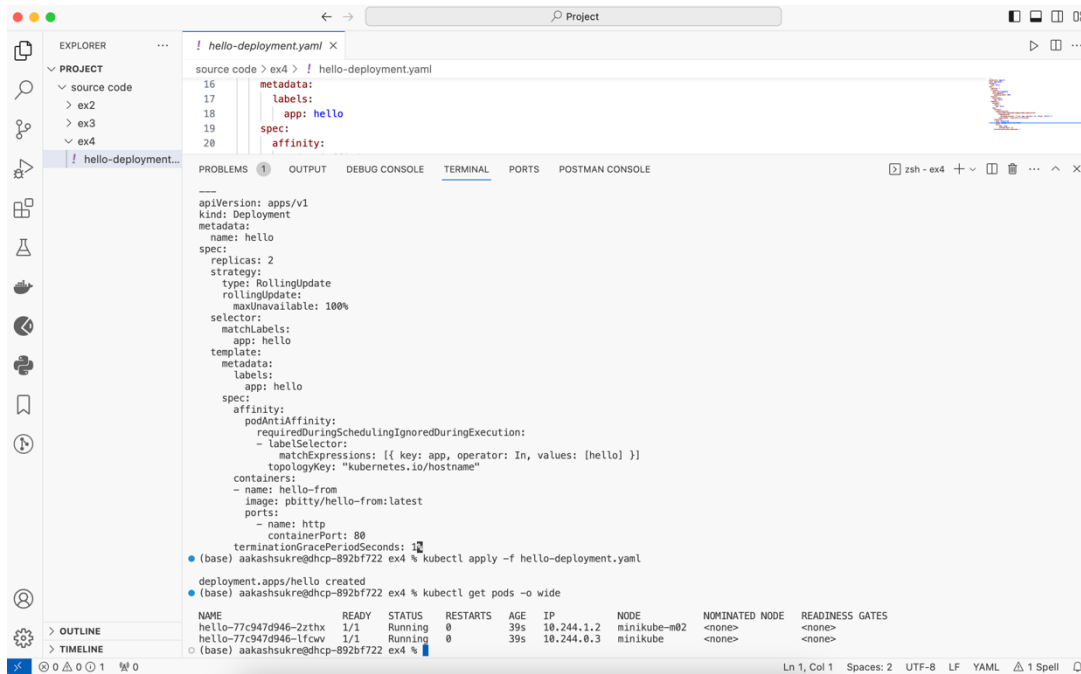
This command is used to create/update the specified configuration in the resources in Kubernetes cluster based on the hello-deployment.yaml file. This is used to tell the Kubernetes what you want to run in your cluster.



Step 4:

kubectl get deployments

This above command is used to see the running deployment.



The screenshot shows a code editor with a file named `hello-deployment.yaml` open. The file contains a Kubernetes Deployment manifest. The terminal output shows the command `kubectl apply -f hello-deployment.yaml` being executed, followed by the command `kubectl get pods -o wide`. The output of the second command shows two pods running on two different nodes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maximumUnavailable: 100%
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: app, operator: In, values: [hello]
              topologyKey: "kubernetes.io/hostname"
      containers:
        - name: hello-from
          image: pbitty/hello-from:latest
          ports:
            - name: http
              containerPort: 80
          terminationGracePeriodSeconds: 10
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
hello-77c947d946-2ztbx	1/1	Running	0	39s	10.244.1.2	minikube-m02	<none>	<none>
hello-77c947d946-1fcwv	1/1	Running	0	39s	10.244.0.3	minikube	<none>	<none>

Step 5:

kubectl get nodes

This command is used to get the information about the nodes in the Kubernetes Cluster. This will show the list of nodes with their status.

kubectl get pods -o wide

This command is used to get the information about the pods in the Kubernetes Cluster. This will show the list of pods with their details in a wide format.

Seen that two pods are running into two different nodes

Here we seen that -

Two nodes created. – 1) minikube 2) minikube-m02

Two pods created, and those pods are running into two different to balance the nodes.

Step 6:

We want to change that and assign the two pods to one specific node.

Following steps to perform this:

- A. Delete the previous deployment and make sure no other pod is running.

```
kubectl delete deployment hello
```

Using this command, I have deleted the previous deployment.

- B. You should associate the preferred node (i.e, the one that we will assign both pods) with a label. You can do so using the command:

```
kubectl label nodes minikube disktype=ssd
```

“Here I used this command for minikube to assign the label SSD.”

- C. "hello-deployment_updated.yaml"

```
kubectl apply -f hello-deployment-updated.yaml
```

This is used to tell the Kubernetes what you want to run in your cluster.

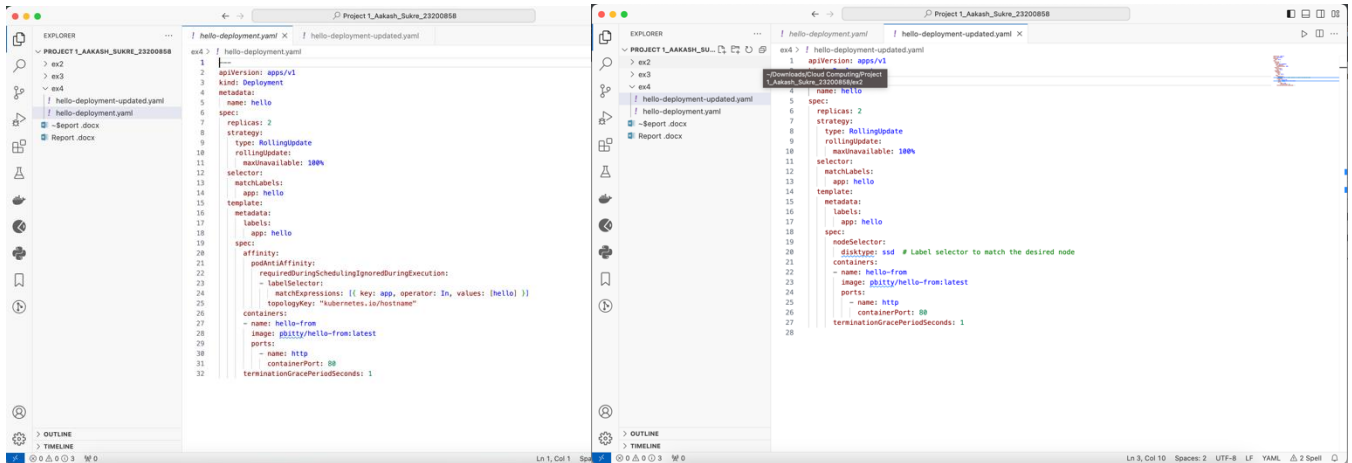
- D. View the new label using the command:

```
kubectl get nodes --show-labels
```

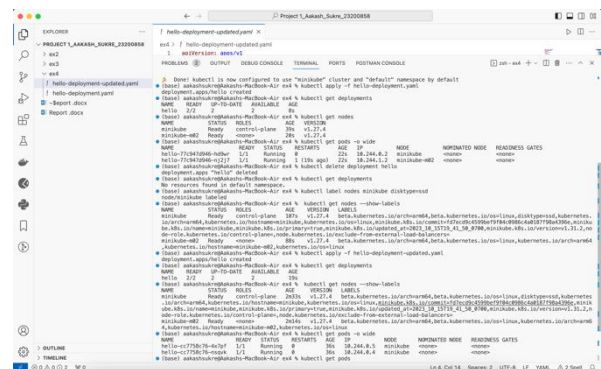
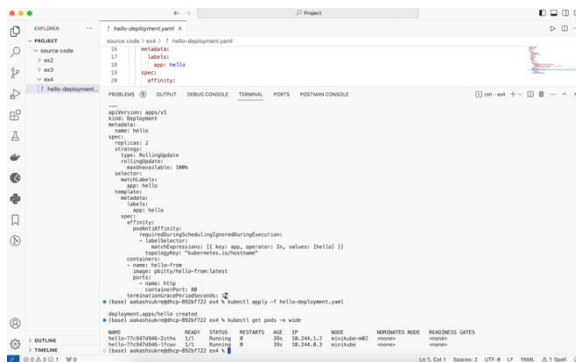
“This is used to check the label nodes”

```
● (base) aakashsukre@Aakashs-MacBook-Air ex4 % kubectl delete deployment hello
deployment.apps "hello" deleted
● (base) aakashsukre@Aakashs-MacBook-Air ex4 % kubectl label nodes minikube disktype=ssd
node/minikube labeled
● (base) aakashsukre@Aakashs-MacBook-Air ex4 % kubectl apply -f hello-deployment-updated.yaml
deployment.apps/hello created
● (base) aakashsukre@Aakashs-MacBook-Air ex4 % kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
hello-cc7758c76-4kjld               1/1     Running   0           4s    10.244.0.4      minikube   <none>            <none>
hello-cc7758c76-cxqql               1/1     Running   0           4s    10.244.0.5      minikube   <none>            <none>
○ (base) aakashsukre@Aakashs-MacBook-Air ex4 %
```

- E. Create a new configuration file with the name "hello-deployment_updated.yaml" Copy inside it the code from "hello-deployment.yaml", and make adjustments to associate this two pods with the node that has been re- ceived the <new-label>.



Here, I have Created the New file hello-deployment_updated.yaml and make the the changes in the file accordingly to achieve the two pods can run on the single node. You can see the changes in the above images.



Summary:

The above exercise gives the knowledge about how to manage the pod scheduling in a Kubernetes cluster. This exercise gives me how to manage pod placement in a Kubernetes cluster, which can be helpful in a variety of situations, such as maximizing resource utilization and workloads execute on designated nodes.