Name Suraj Kumar
PNR 7041

Q1
cereate a directory create an index.html file the index.html the index.html file will display message "this application is running on kubernetes" cretae a image using httpd image copy index.html inside the container run the container and map port 8200 using your image and display the webpage is displayd

```
mkdr project

cd project

echo "This application is running on Kubernetes" > index.html
```

Docker file

```
# Use the official HTTPD image as a base image
FROM httpd:latest

# Copy the custom index.html file into the container's web root directory
COPY index.html /usr/local/apache2/htdocs/

# Expose port 8200
EXPOSE 8200
```

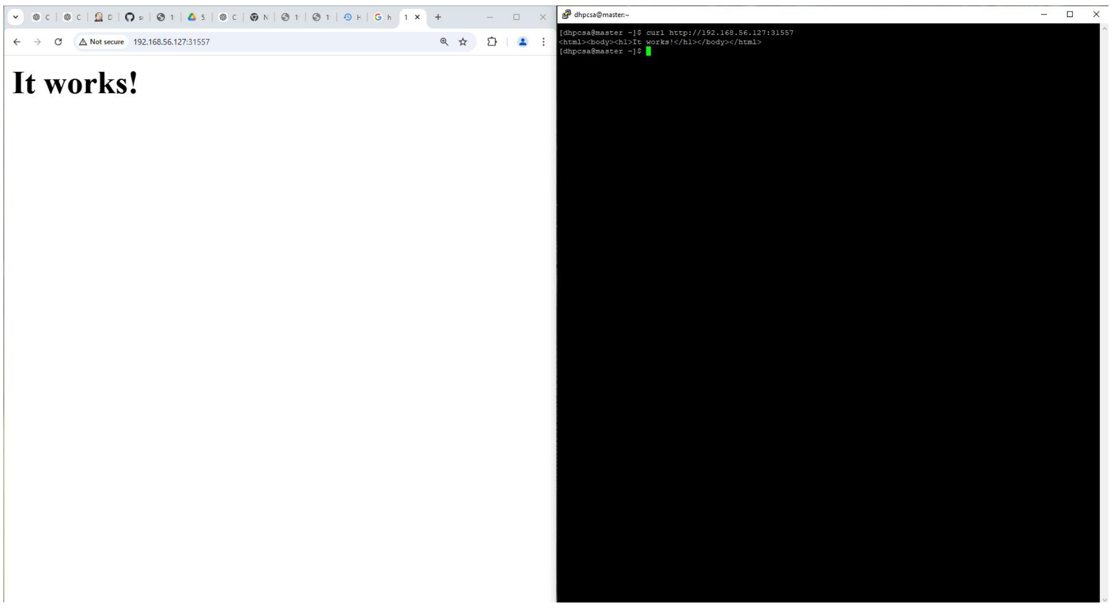
```
docker build -t my-httpd-app .
```

Run the Docker Container

```
docker run -d -p 8200:80 my-httpd-app
```

```
[dhpcsa@master project]$ curl http://localhost:8200

This application is running on Kubernetes
```



It works!

```
[dhpcsa@master ~]$ curl http://192.168.56.127:31557
<html><body><h1>It works!</h1></body></html>
[dhpcsa@master ~]$
```

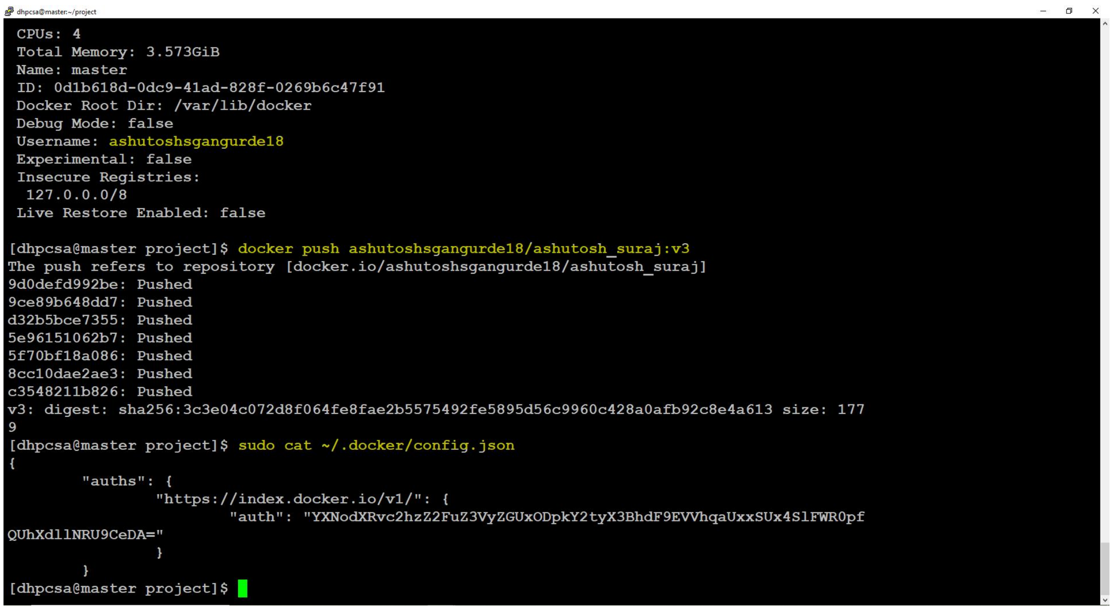
Q2
create a repository on your docker hub account push this image to docker repo

make a new repo on docker hub

1. Log in to Docker Hub
docker login

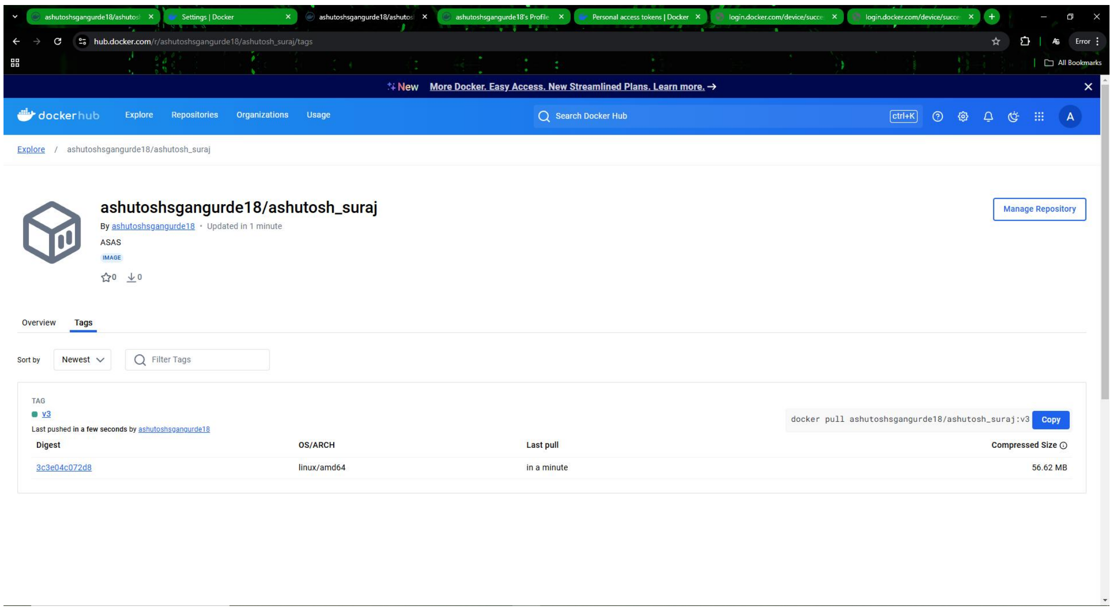
2. Tag the image
docker tag <local-image-name> <dockerhub-username>/<repository-name>:<tag>

3. Push the image to Docker Hub
docker push <dockerhub-username>/<repository-name>:<tag>



```
dhpcsa@master:~/project
 CPUs: 4
 Total Memory: 3.573GiB
 Name: master
 ID: 0d1b618d-0dc9-41ad-828f-0269b6c47f91
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Username: ashutoshsgangurde18
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

[dhpcsa@master project]$ docker push ashutoshsgangurde18/ashutosh_suraj:v3
The push refers to repository [docker.io/ashutoshsgangurde18/ashutosh_suraj]
9d0defd992be: Pushed
9ce89b648dd7: Pushed
d32b5bce7355: Pushed
5e96151062b7: Pushed
5f70bf18a086: Pushed
8cc10dae2ae3: Pushed
c3548211b826: Pushed
v3: digest: sha256:3c3e04c072d8f064fe8fae2b5575492fe5895d56c9960c428a0afb92c8e4a613 size: 177
9
[dhpcsa@master project]$ sudo cat ~/.docker/config.json
{
        "auths": {
                "https://index.docker.io/v1/": {
                        "auth": "YXNodXRvc2hzZ2FuZ3VyZGUxODpkY2tyX3BhdF9EVVhqaUxxSUx4S1FWR0pf
QUhXdllNRU9CeDA="
                }
        }
[dhpcsa@master project]$
```



hub.docker.com/r/ashutoshsgangurde18/ashutosh_suraj/tags

New More Docker. Easy Access. New Streamlined Plans. Learn more. →

Explore / ashutoshsgangurde18/ashutosh_suraj

ashutoshsgangurde18/ashutosh_suraj

By ashutoshsgangurde18 · Updated in 1 minute

ASAS

IMAGE

Overview Tags

Sort by Newest

TAG

v3

Last pushed in a few seconds by ashutoshsgangurde18

`docker pull ashutoshsgangurde18/ashutosh_suraj:v3`

Digest	OS/ARCH	Last pull	Compressed Size
3c3e04c072d8	linux/amd64	in a minute	56.62 MB

Q3

on the Kubernetes cluster create a deployment using httpd image check on which node the pod is runing create a service of type nodeport for this deployment check the port assigned for this service use url to access website on the specified port on the node address

Step 1: Create a Deployment with the httpd Image

```
kubectl create deployment httpd-deployment --image=httpd
```

Verify the Deployment:

```
kubectl get deployments
```

```
kubectl get pods
```

Step 2: Expose the Deployment Using a NodePort Service

NodePort service to make it accessible from outside the cluster.

```
kubectl expose deployment httpd-deployment --type=NodePort --
name=httpd-service --port=80 --target-port=80
```

To check which port has been assigned to the NodePort service

```
kubectl get svc httpd-service
```

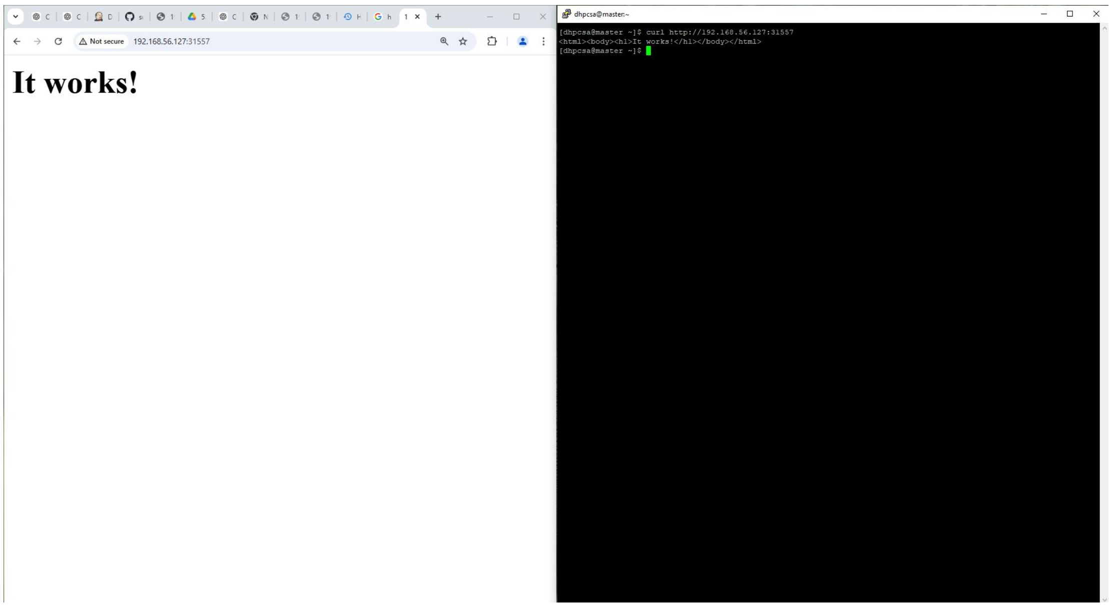
Step 3: Check on Which Node the Pod is Running

```
kubectl get pod -o wide
```

Get the Node IP:

```
kubectl get nodes -o wide
```

Access the website through the browser using the format http://<node-ip>:<NodePort>, e.g., http://192.168.56.127:31557.



It works!

```
[dhpcsa@master ~]$ curl http://192.168.56.127:31557
<html><body><h1>It works!</h1></body></html>
[dhpcsa@master ~]$
```

Q4
scale the deployment to create 10 pods check on which node the pods are running check the ip addresses assigned to the pods try accessing website on all node ip and specified port then scale the deployment to 2 replic

1. Scale the Deployment to Create 10 Pods
kubectl scale deployment my-deployment --replicas=10

Check the IP addresses of the pods
kubectl get pods -o wide

Access the website using node IP and port

curl http://<node-ip>:<node-port>

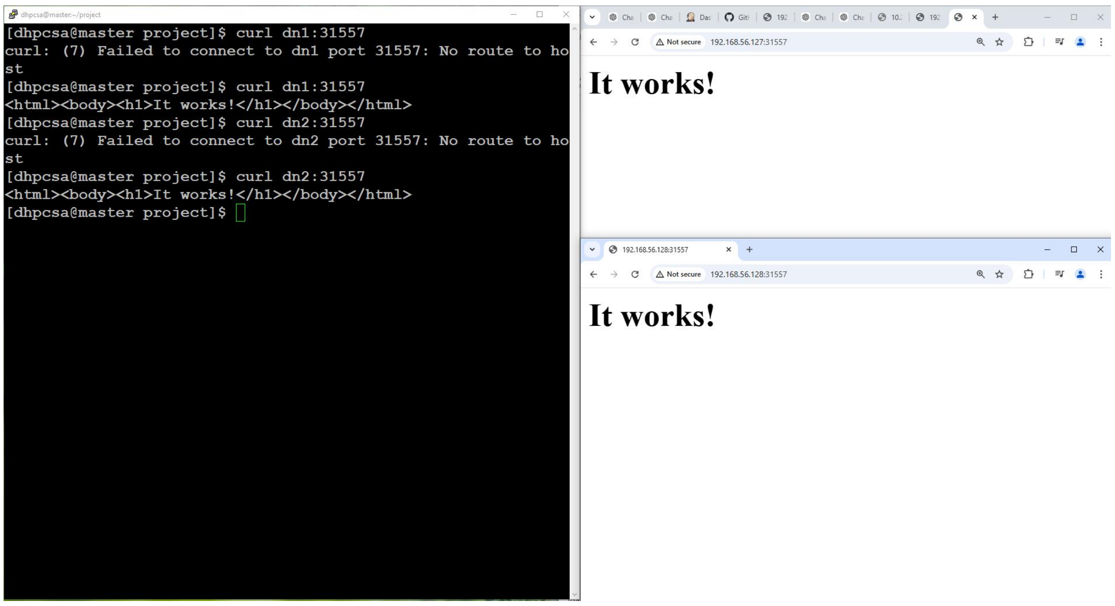
Scale the deployment to 2 replicas:
kubectl scale deployment <deployment-name> --replicas=2

running nodes
kubectl get nodes

to check the service and port
kubectl get svc

to check which pod is running on which node
kubectl get pods -o wide

curl http://<node-ip>:<node-port>



dhpcsa@master:~/project

```
[dhpcsa@master project]$ curl dn1:31557
curl: (7) Failed to connect to dn1 port 31557: No route to ho
st
[dhpcsa@master project]$ curl dn1:31557
<html><body><h1>It works!</h1></body></html>
[dhpcsa@master project]$ curl dn2:31557
curl: (7) Failed to connect to dn2 port 31557: No route to ho
st
[dhpcsa@master project]$ curl dn2:31557
<html><body><h1>It works!</h1></body></html>
[dhpcsa@master project]$
```

Not secure 192.168.56.127:31557

It works!

192.168.56.128:31557

Not secure 192.168.56.128:31557

It works!