TASK 3-KUBERNET

Step 1:

Installing the minikube

```
vishal@LAPTOP-U45BV05I:~$ sudo mkdir -p /etc/docker
echo '{ "experimental": true, "features": { "buildkit": true } }' | sudo tee /etc/docker/daemon.json
sudo systemctl restart docker
{ "experimental": true, "features": { "buildkit": true } }
vishal@LAPTOP-U45BV05I:~$ minikube config set driver docker
❗  These changes will take effect upon a minikube delete and then a minikube start
```

Step 2:

Starting the minikube using “minikube start “ command

```
vishal@LAPTOP-U45BV05I:~$ minikube start --driver=docker --container-runtime=docker
😄  minikube v1.35.0 on Ubuntu 24.04 (amd64)
✨  Using the docker driver based on user configuration

🧯  The requested memory allocation of 2200MiB does not leave room for system overhead (total system memory: 2901MiB). You may face stability issues.
💡  Suggestion: Start minikube with less memory allocated: 'minikube start --memory=2200mb'

📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
💾  Downloading Kubernetes v1.32.0 preload ...
    > preloaded-images-k8s-v18-v1...:  333.57 MiB / 333.57 MiB  100.00% 1.24 Mi
    > gcr.io/k8s-minikube/kicbase...:  500.31 MiB / 500.31 MiB  100.00% 1.61 Mi
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
💡  kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Step 3:

Now create a deployment named r2 using the image in the docker hub

```
vishal@LAPTOP-U45BV05I:~$ kubectl create deployment r2 --image=vishal15276t/test1 --port=80
deployment.apps/r2 created
```

Step 4:Now give kubectl get pods to check if the container is running and wait until it starts running

```
vishal@LAPTOP-U45BV05I:~$ kubectl get pods
NAME                  READY   STATUS              RESTARTS   AGE
r1-7b886b659-f2sv6    1/1     Running             0          9m38s
r2-f784c9f59-7f7g9    0/1     ContainerCreating   0          6s
```

```
NAME                  READY   STATUS    RESTARTS   AGE
r1-7b886b659-f2sv6    1/1     Running   0          10m
r2-f784c9f59-7f7g9    1/1     Running   0          46s
```

Step 5:

Now expose the deployment using the expose command

```
vishal@LAPTOP-U45BV05I:~$ kubectl expose deployment r2 --port=80 --type=NodePort
service/r2 exposed
vishal@LAPTOP-U45BV05I:~$ minikube service r2
```

Step 6:

Now give service command to check the ip address of the deployed image

```
vishal@LAPTOP-U45BV05I:~$ minikube service r2
|-----------|------|-------------|---------------------------|
| NAMESPACE | NAME | TARGET PORT |            URL            |
|-----------|------|-------------|---------------------------|
| default   | r2   |          80 | http://192.168.49.2:30361 |
|-----------|------|-------------|---------------------------|
🏃  Starting tunnel for service r2.
|-----------|------|-------------|------------------------|
| NAMESPACE | NAME | TARGET PORT |          URL           |
|-----------|------|-------------|------------------------|
| default   | r2   |             | http://127.0.0.1:41619 |
|-----------|------|-------------|------------------------|
🎉  Opening service default/r2 in default browser...
👉  http://127.0.0.1:41619
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
65.............33333333333333333333333333333333333333333333333333333333333333333p^[[3~
```

Step 7:

The output will be displayed as follows