ASSIGNMENT-4

Kubernetes Deployment with Minikube

This guide walks you through deploying a simple web application using Minikube and Kubernetes. You will create a Minikube cluster, deploy a web app using an nginx image, expose it as a service, and test it using curl. The following steps will guide you through setting up and monitoring the deployment.

STEPS:

Step 1: Start Minikube

Start Minikube with Docker as the driver to set up your local Kubernetes cluster.

Command:

```
minikube start --driver=docker --force
```

Step 2: Create a Deployment

Create a Kubernetes deployment for the nginx web application, specifying the port for the container.

Command:

```
kubectl create deployment webapp --image=nginx --port=80
```

Step 3: Expose the Deployment as a Service

Expose the webapp deployment as a NodePort service to make the app accessible outside the cluster.

Command:

```
kubectl expose deployment webapp --type=NodePort --port=80 --target-port=80
```

Step 4: Verify the Running Pods

Check the status of the pods to ensure they are running as expected.

Command:

```
kubectl get pods
```

Step 5: Verify the Service

Check the details of the service to confirm the webapp is correctly exposed.

Command:

```
kubectl get svc
```

Step 6: Open the Service in a Web Browser

Open the webapp service in your browser to verify it’s running.

Command:

```
minikube service webapp
```

Step 7: Test the Service Using curl

Use curl to test the service connection and ensure it's accessible.

Command:

```
curl http://192.168.49.2:31432
```

Step 8: Continuously Monitor the Pods

Monitor the pod status in real-time to ensure everything is working as expected.

Command:

```
watch kubectl get pod
```

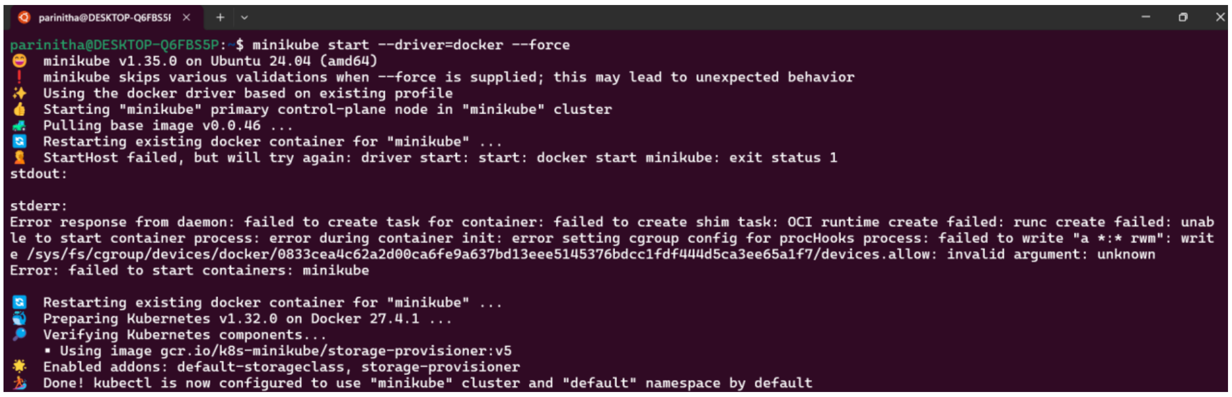
Step 9: Continuously Monitor Pod Logs

Use watch to continuously monitor the logs of the webapp pod for any issues.

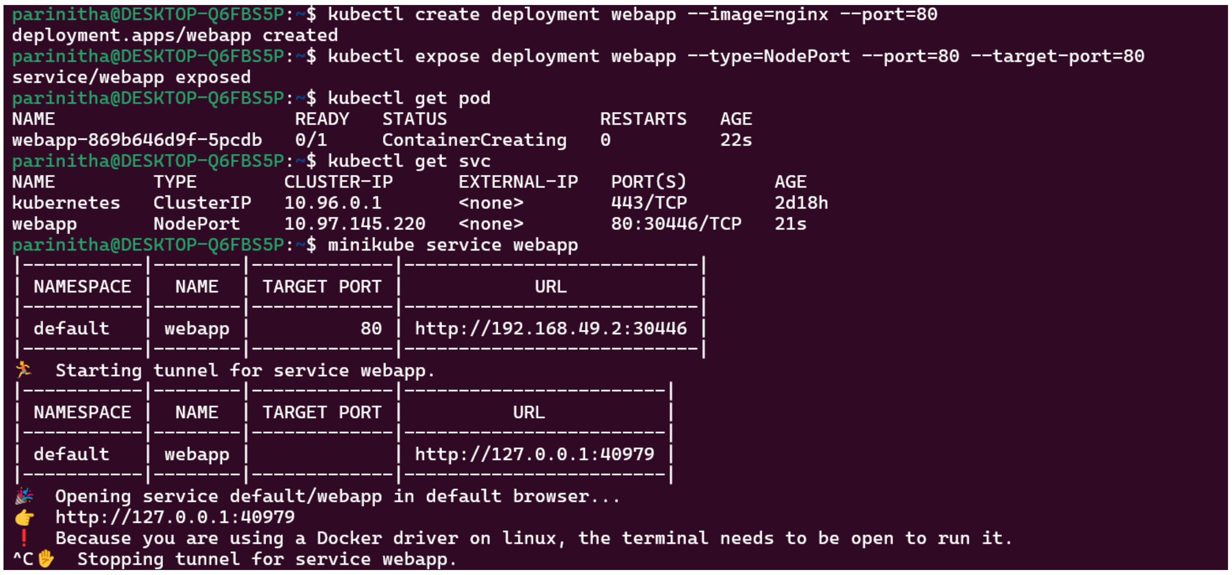
Command:

```
watch kubectl logs webapp-869b646d9f-b4hgr
```

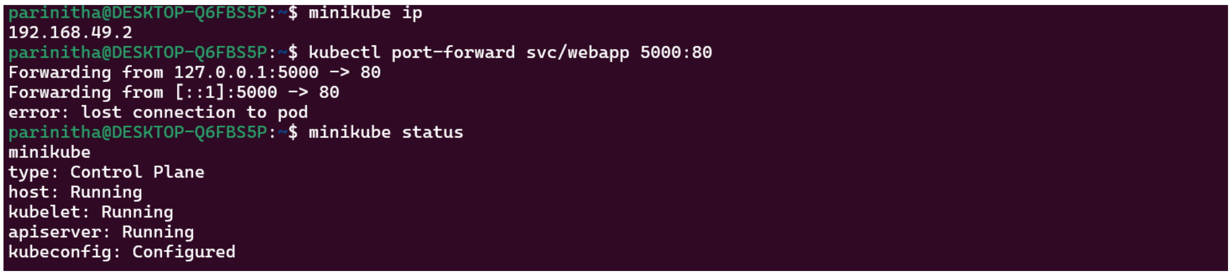
OUTPUT:



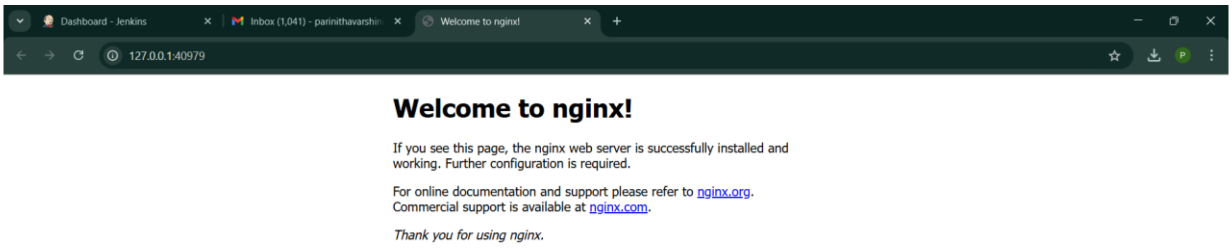
```
parinitha@DESKTOP-Q6FBS5P:~$ minikube start --driver=docker --force
😄  minikube v1.35.0 on Ubuntu 24.04 (amd64)
❗  minikube skips various validations when --force is supplied; this may lead to unexpected behavior
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
🔄  Restarting existing docker container for "minikube" ...
🤦  StartHost failed, but will try again: driver start: start: docker start minikube: exit status 1
stdout:

stderr:
Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime create failed: runc create failed: unab
le to start container process: error during container init: error setting cgroup config for procHooks process: failed to write "a *:* rwm": writ
e /sys/fs/cgroup/devices/docker/0833cea4c62a2d00ca6fe9a637bd13eee5145376bdcc1fdf444d5ca3ee65a1f7/devices.allow: invalid argument: unknown
Error: failed to start containers: minikube

🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```



```
parinitha@DESKTOP-Q6FBS5P:~$ kubectl create deployment webapp --image=nginx --port=80
deployment.apps/webapp created
parinitha@DESKTOP-Q6FBS5P:~$ kubectl expose deployment webapp --type=NodePort --port=80 --target-port=80
service/webapp exposed
parinitha@DESKTOP-Q6FBS5P:~$ kubectl get pod
NAME                      READY   STATUS              RESTARTS   AGE
webapp-869b646d9f-5pcdb   0/1     ContainerCreating   0          22s
parinitha@DESKTOP-Q6FBS5P:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        2d18h
webapp       NodePort    10.97.145.220   <none>        80:30446/TCP   21s
parinitha@DESKTOP-Q6FBS5P:~$ minikube service webapp
|-----------|--------|-------------|---------------------------|
| NAMESPACE |  NAME  | TARGET PORT |            URL            |
|-----------|--------|-------------|---------------------------|
| default   | webapp |          80 | http://192.168.49.2:30446 |
|-----------|--------|-------------|---------------------------|
🏃  Starting tunnel for service webapp.
|-----------|--------|-------------|------------------------|
| NAMESPACE |  NAME  | TARGET PORT |          URL           |
|-----------|--------|-------------|------------------------|
| default   | webapp |             | http://127.0.0.1:40979 |
|-----------|--------|-------------|------------------------|
🎉  Opening service default/webapp in default browser...
👉  http://127.0.0.1:40979
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
^C✋  Stopping tunnel for service webapp.
```



```
parinitha@DESKTOP-Q6FBS5P:~$ minikube ip
192.168.49.2
parinitha@DESKTOP-Q6FBS5P:~$ kubectl port-forward svc/webapp 5000:80
Forwarding from 127.0.0.1:5000 -> 80
Forwarding from [::1]:5000 -> 80
error: lost connection to pod
parinitha@DESKTOP-Q6FBS5P:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.