

DEPLOYING NGINX ON KUBERNETES

GROUP 4

GROUP MEMBERS

- ☐ STEPHEN AMIHERE
- ☐ INNOCENTIA AZAL
- ☐ BENEDICTA OPOKU-AMANKWAAH
- ☐ CLEMENT OWUSU BEMPAH
- ☐ ANGELA KESSE
- ☐ RIVERSON ATTA
- ☐ JOEL AFEDU

OBJECTIVES

- ❑ Deploy Nginx on Kubernetes.
- ❑ Manage deployments and scaling.
- ❑ Expose the app externally via Service.
- ❑ Document the process.

TOOLS USED

- ❑ Kubernetes Environment: Minikube
- ❑ CLI Tool: kubectl Sample
- ❑ App: Nginx Web Server System
- ❑ Requirements: Installed Docker Working Kubernetes CLI

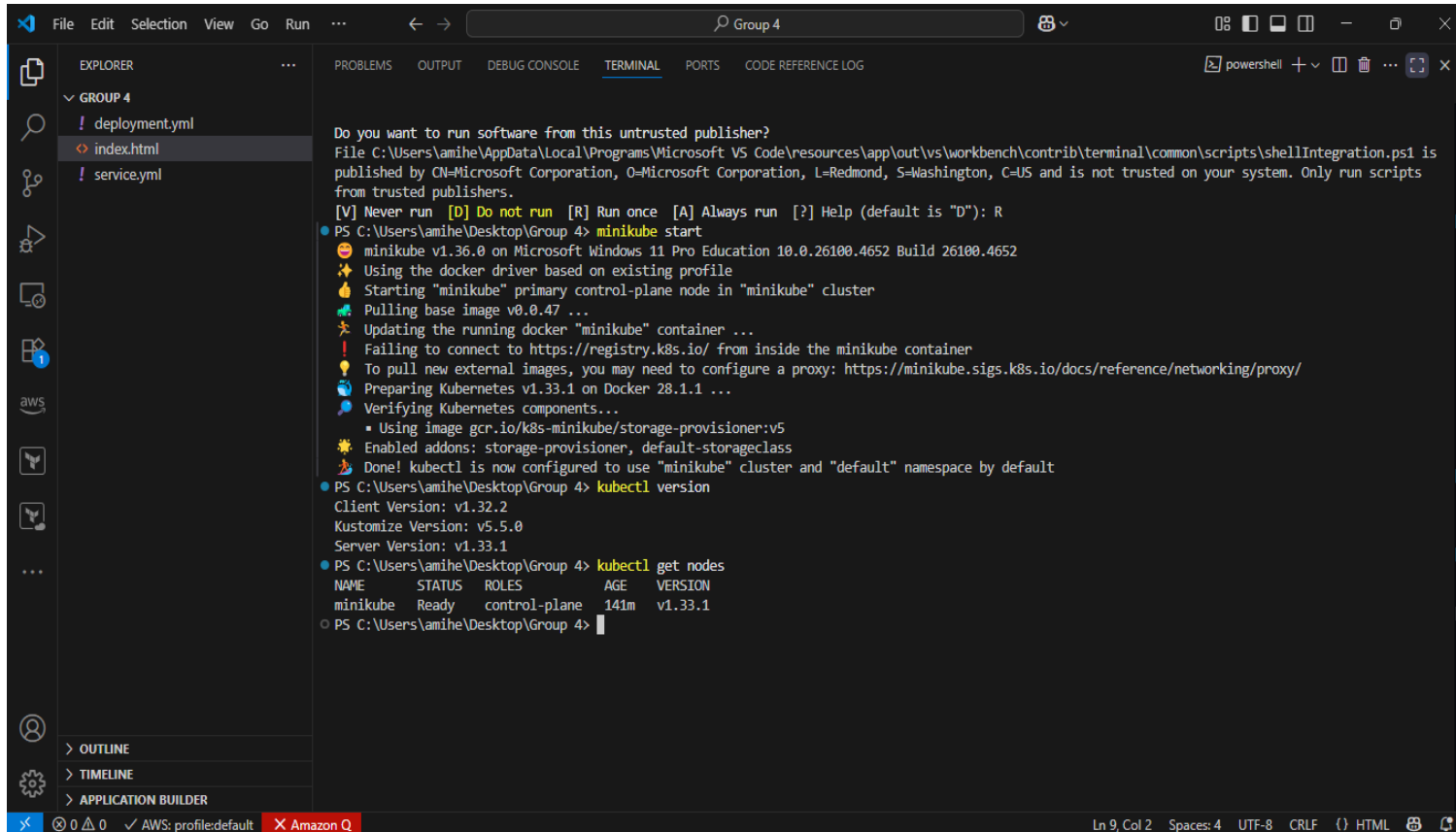
TASK 1: SELECTING A SAMPLE APPLICATION

Chosen App: Nginx

Why Nginx?:

- Nginx is lightweight and easy to deploy.
- It supports customization. You can easily replace the default page with your own index.html to demonstrate custom deployments.

TASK 2: SETTING UP ENVIRONMENT



The screenshot shows the Visual Studio Code interface with a terminal window open. The Explorer pane on the left shows a file structure for 'GROUP 4' with files 'deployment.yml', 'index.html', and 'service.yml'. The terminal window displays the output of the following commands:

```
PS C:\Users\amihe\Desktop\Group 4> minikube start
minikube v1.36.0 on Microsoft Windows 11 Pro Education 10.0.26100.4652 Build 26100.4652
Using the docker driver based on existing profile
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.47 ...
Updating the running docker "minikube" container ...
Failing to connect to https://registry.k8s.io/ from inside the minikube container
To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
Preparing Kubernetes v1.33.1 on Docker 28.1.1 ...
Verifying Kubernetes components...
  Using image gcr.io/k8s-minikube/storage-provisioner:v5
Enabled addons: storage-provisioner, default-storageclass
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

PS C:\Users\amihe\Desktop\Group 4> kubectl version
Client Version: v1.32.2
Kustomize Version: v5.5.0
Server Version: v1.33.1

PS C:\Users\amihe\Desktop\Group 4> kubectl get nodes
NAME          STATUS    ROLES          AGE   VERSION
minikube      Ready     control-plane  141m  v1.33.1

PS C:\Users\amihe\Desktop\Group 4>
```

The status bar at the bottom shows 'Ln 9, Col 2', 'Spaces: 4', 'UTF-8', 'CRLF', and 'HTML'.

- ❑ For this task, we started minikube by running ***minikube start*** and confirmed whether kubectl was working the command ***kubectl version***.

TASK 3: CREATING A DEPLOYMENT MANIFEST

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: nginx-deployment
5   labels:
6     app: nginx
7 spec:
8   replicas: 1
9   selector:
10    matchLabels:
11      app: nginx
12   template:
13     metadata:
14       labels:
15         app: nginx
16     spec:
17       containers:
18         - name: nginx
19           image: nginx:latest
20           ports:
21             - containerPort: 80
22           volumeMounts:
23             - name: nginx-index-volume
24               mountPath: /usr/share/nginx/html
25       volumes:
26         - name: nginx-index-volume
27           configMap:
28             name: nginx-index-html
```

Before diving into file creation, we first set up a dedicated project folder on our desktop named **GROUP 4**. This provided a clean workspace and easy access, which we then opened using Visual Studio Code.

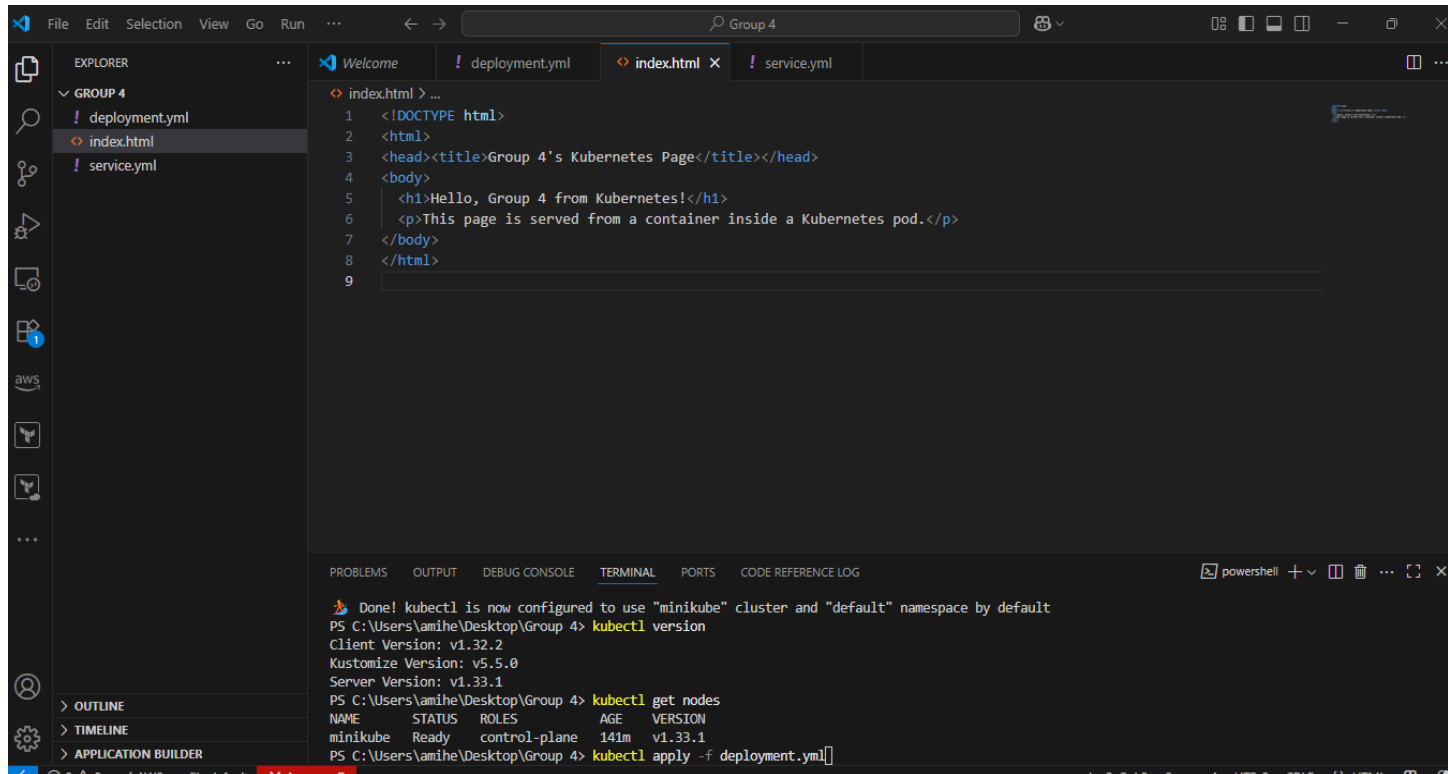
❑ Process:

We created file name ***deployment.yml***, that defines how Kubernetes should deploy our app including the container image, replicas, and ports.

❑ Purpose:

The deployment keeps our app running, even if a pod crashes.

TASK 3 CONT.: CREATING AN HTML FILE



The screenshot shows the Visual Studio Code interface with a project named 'Group 4'. The Explorer sidebar on the left shows a file structure with 'deployment.yml', 'index.html', and 'service.yml'. The main editor area has three tabs: 'Welcome', 'deployment.yml', and 'index.html'. The 'index.html' tab is active, displaying the following HTML code:

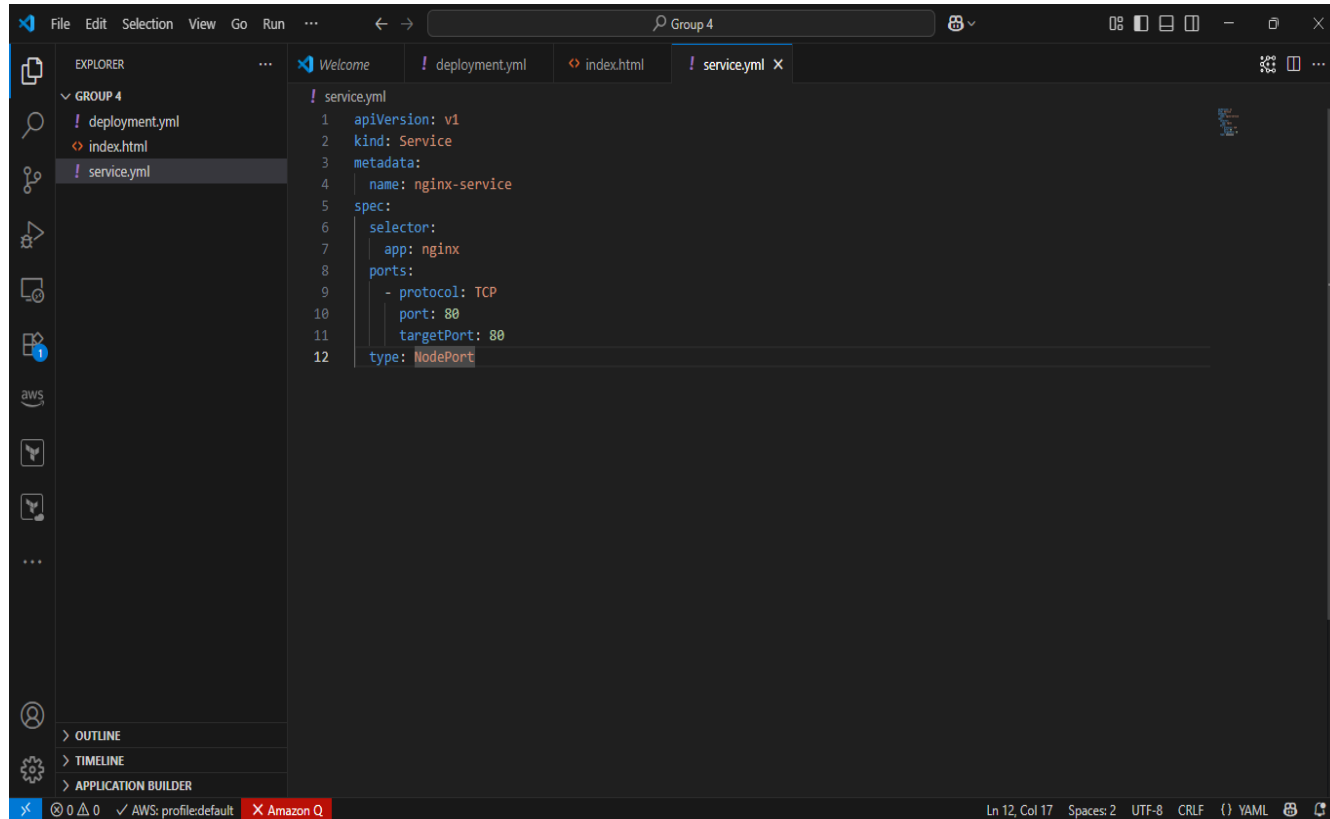
```
1 <!DOCTYPE html>
2 <html>
3 <head><title>Group 4's Kubernetes Page</title></head>
4 <body>
5   <h1>Hello, Group 4 from Kubernetes!</h1>
6   <p>This page is served from a container inside a Kubernetes pod.</p>
7 </body>
8 </html>
9
```

At the bottom, the Terminal panel shows the output of several Kubernetes commands:

```
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\Users\amihe\Desktop\Group 4> kubectl version
Client Version: v1.32.2
Kustomize Version: v5.5.0
Server Version: v1.33.1
PS C:\Users\amihe\Desktop\Group 4> kubectl get nodes
NAME      STATUS   ROLES    AGE   VERSION
minikube  Ready    control-plane  141m  v1.33.1
PS C:\Users\amihe\Desktop\Group 4> kubectl apply -f deployment.yml
```

- ❑ We also created an HTML file to personalize the default Nginx welcome page and customization by giving it a custom message: ***Hello Group 4 from Kubernetes.***

TASK 4: EXPOSING THE APPLICATION



The screenshot shows the Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'GROUP 4' with files 'deployment.yaml', 'index.html', and 'service.yaml'. The 'service.yaml' file is open in the code editor, showing the following content:

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: nginx-service
5 spec:
6   selector:
7     app: nginx
8   ports:
9     - protocol: TCP
10      port: 80
11      targetPort: 80
12   type: NodePort
```

For this we did the following:

❑ Process:

We created file and named it ***Service.yaml*** to make our app accessible outside the cluster using a NodePort

❑ Purpose:

Kubernetes apps are not accessible by default. You expose them so users can reach them.

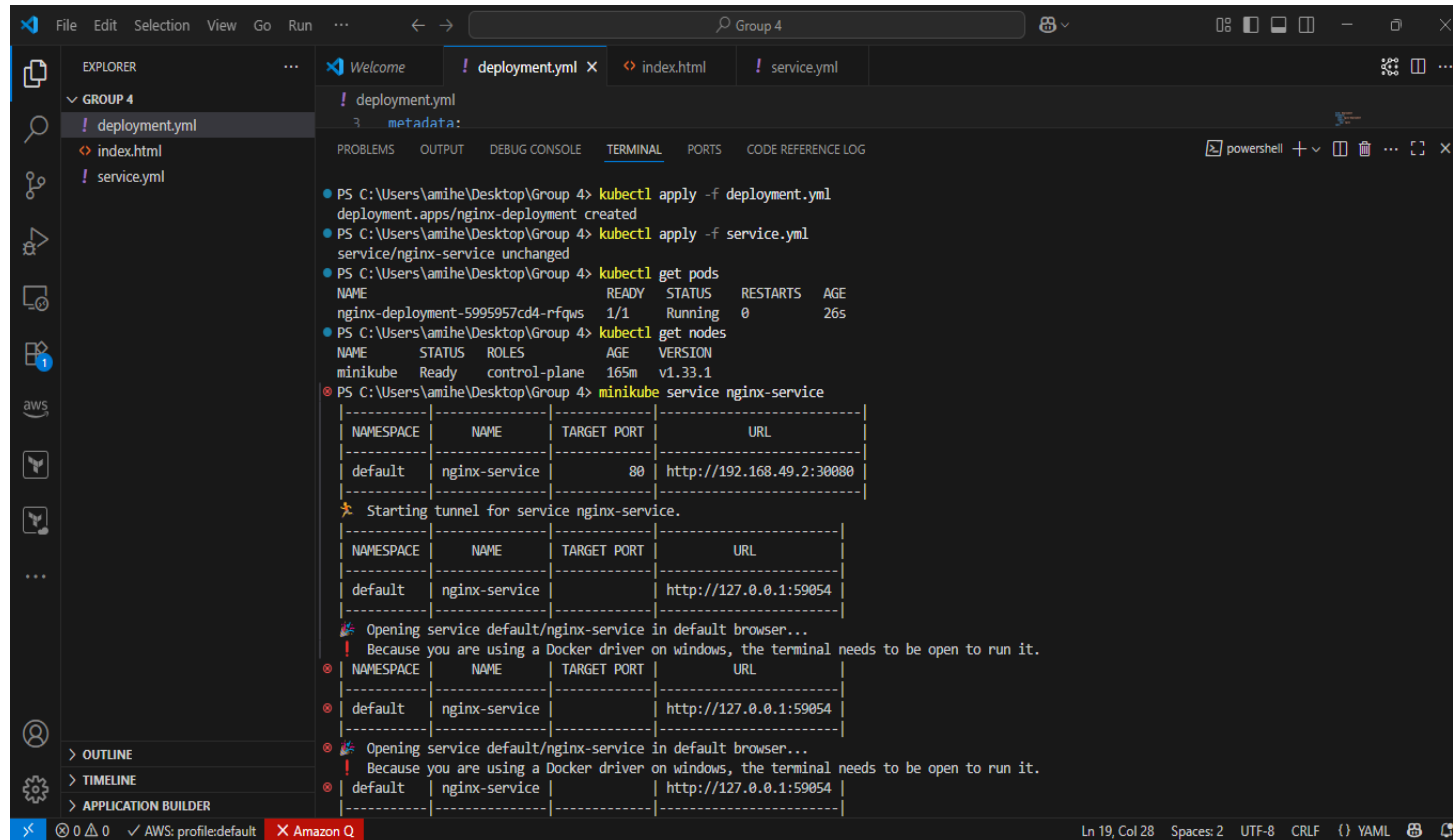
TASK 4 CONT.

```
PS C:\Users\amihe\Desktop\Group 4> kubectl apply -f deployment.yml
deployment.apps/nginx-deployment created
PS C:\Users\amihe\Desktop\Group 4> kubectl apply -f service.yml
service/nginx-service unchanged
PS C:\Users\amihe\Desktop\Group 4> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-deployment-5995957cd4-pzb9w	1/1	Running	0	23s

- ☐ Applied the deployment using the Kubernetes manifest file: **deployment.yml** with the command ***kubectl apply -f deployment.yml***
- ☐ Applied the service using the Kubernetes manifest file: **service.yml** with the command ***kubectl apply -f service.yml***. The reason for the service unchanged is that we had run the command earlier.
- ☐ Verified pod status and confirmed successful deployment with ***kubectl get pods***.
- ☐ Created 1 replica of the custom Nginx container serving a styled HTML page.

TASK 5: DEPLOY AND TEST



The screenshot shows a VS Code terminal window with the following commands and output:

```
PS C:\Users\amihe\Desktop\Group 4> kubectl apply -f deployment.yml
deployment.apps/nginx-deployment created
PS C:\Users\amihe\Desktop\Group 4> kubectl apply -f service.yml
service/nginx-service unchanged
PS C:\Users\amihe\Desktop\Group 4> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-5995957cd4-rfqms   1/1     Running   0           26s
PS C:\Users\amihe\Desktop\Group 4> kubectl get nodes
NAME      STATUS   ROLES    AGE   VERSION
minikube  Ready    control-plane  165m  v1.33.1
PS C:\Users\amihe\Desktop\Group 4> minikube service nginx-service
```

The output for `minikube service nginx-service` shows a tunnel starting and a table of service endpoints:

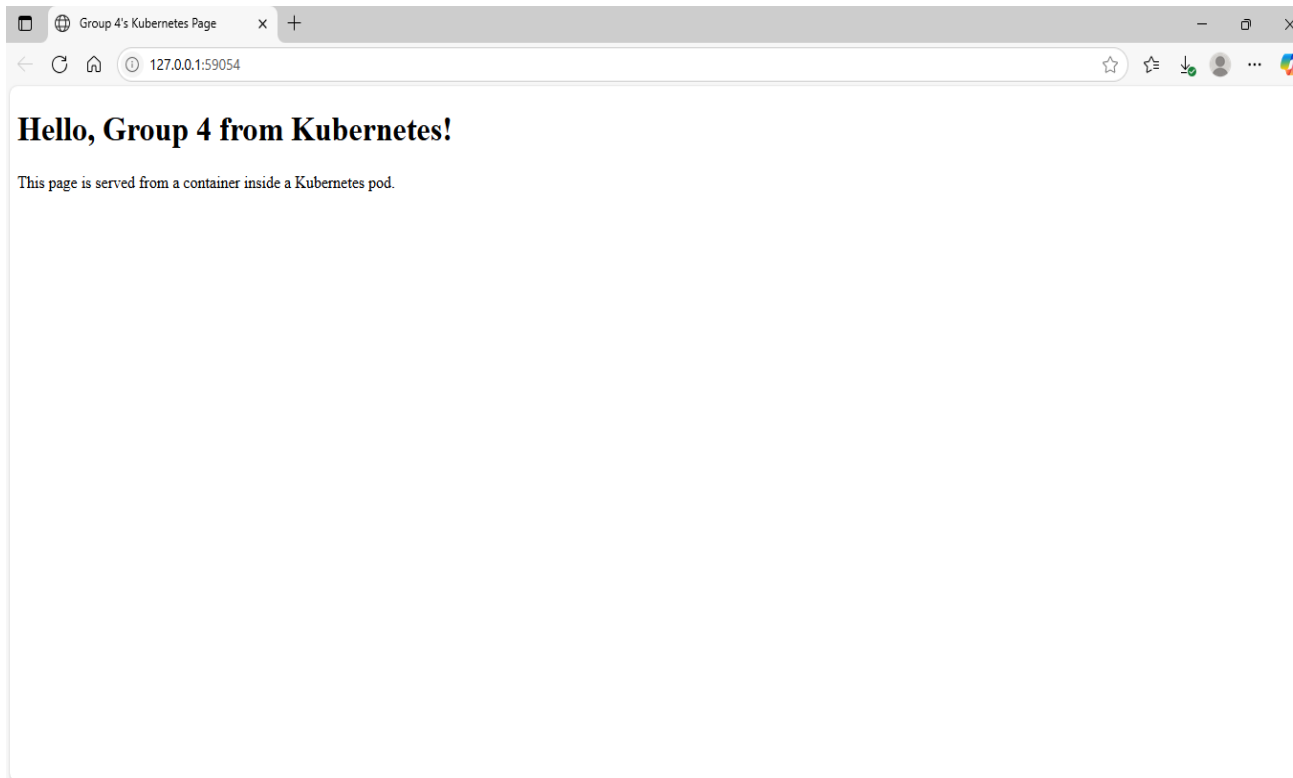
NAMESPACE	NAME	TARGET PORT	URL
default	nginx-service	80	http://192.168.49.2:30080

Subsequent attempts to open the service in the default browser fail with the message: "Opening service default/nginx-service in default browser... ! Because you are using a Docker driver on windows, the terminal needs to be open to run it."

❑ For deploying and testing , we used the command ***minikube service nginx-service*** to test our app in the browser.

❑ **Purpose:**
This step confirms our app is working and accessible.

TASK 5 CONT.: OUTCOME OF DEPLOYMENT



- ❑ After executing ***minikube service nginx-service***, Minikube automatically launched the associated web service URL in our default browser, rendering our custom landing page as expected.

TASK 6 : SCALING AND UPDATE EXERCISE

```
PS C:\Users\amihe\Desktop\Group 4> kubectl scale deployment/nginx-deployment --replicas=5
deployment.apps/nginx-deployment scaled
deployment.apps/nginx-deployment scaled
PS C:\Users\amihe\Desktop\Group 4> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-deployment-5995957cd4-27q66	0/1	ContainerCreating	0	12s
nginx-deployment-5995957cd4-cxf8p	1/1	Running	0	12s
nginx-deployment-5995957cd4-lcc6q	1/1	Running	0	12s
nginx-deployment-5995957cd4-pbk2k	1/1	Running	0	12s
nginx-deployment-5995957cd4-rfqws	1/1	Running	0	9m42s

- ❑ For scaling we used the command:
kubectl scale deployment/nginx-deployment --replicas=5
- ❑ Purpose:
Adjusts the number of running pod replicas for the nginx-deployment
- ❑ Outcome:
The application now runs 5 instances, enhancing availability and load distribution.

TASK 6 CONT.:EFFECT OF SCALING ON THE APPLICATION

- ❑ **Improved availability:** More pods = higher uptime.
- ❑ **Load balancing:** Traffic is distributed across all replicas.
- ❑ **Faster response times:** Increased capacity to handle requests.
- ❑ **Resource usage:** Higher resource consumption depending on the replica count.

TASK 6 CONT.: SCALING AND UPDATE EXERCISE

```
PS C:\Users\amihe\Desktop\Group 4> kubectl set image deployment/nginx-deployment nginx=nginx:1.23
nginx-deployment-5995957cd4-27q66 0/1 ContainerCreating 0 12s
nginx-deployment-5995957cd4-cxf8p 1/1 Running 0 12s
nginx-deployment-5995957cd4-lcc6q 1/1 Running 0 12s
nginx-deployment-5995957cd4-pbk2k 1/1 Running 0 12s
nginx-deployment-5995957cd4-rfqws 1/1 Running 0 9m42s
nginx-deployment-5995957cd4-27q66 0/1 ContainerCreating 0 12s
nginx-deployment-5995957cd4-cxf8p 1/1 Running 0 12s
nginx-deployment-5995957cd4-lcc6q 1/1 Running 0 12s
nginx-deployment-5995957cd4-pbk2k 1/1 Running 0 12s
nginx-deployment-5995957cd4-27q66 0/1 ContainerCreating 0 12s
nginx-deployment-5995957cd4-cxf8p 1/1 Running 0 12s
nginx-deployment-5995957cd4-lcc6q 1/1 Running 0 12s
nginx-deployment-5995957cd4-pbk2k 1/1 Running 0 12s
nginx-deployment-5995957cd4-27q66 0/1 ContainerCreating 0 12s
nginx-deployment-5995957cd4-cxf8p 1/1 Running 0 12s
nginx-deployment-5995957cd4-lcc6q 1/1 Running 0 12s
nginx-deployment-5995957cd4-cxf8p 1/1 Running 0 12s
nginx-deployment-5995957cd4-lcc6q 1/1 Running 0 12s
nginx-deployment-5995957cd4-lcc6q 1/1 Running 0 12s
nginx-deployment-5995957cd4-pbk2k 1/1 Running 0 12s
nginx-deployment-5995957cd4-rfqws 1/1 Running 0 9m42s
```

For rolling update, we did the following:

- ☐ Updated Kubernetes Deployment without downtime using ***kubectl set image***.
- ☐ Swapped container image tag to the latest stable version for improved performance.
- ☐ Pods replaced incrementally, ensuring uninterrupted service throughout the rollout.
- ☐ Verified rollout status using ***kubectl rollout*** and confirmed success

ROLLING OUT UPDATE (OPTIONAL)

```
PS C:\Users\amihe\Desktop\Group 4> kubectl rollout status deployment/nginx-deployment
deployment "nginx-deployment" successfully rolled out
```

- ❑ Monitored the rollout of the updated deployment to ensure all pods transitioned successfully.
- ❑ Confirmed that the new container image was applied without interruption to service availability.
- ❑ Used ***kubectl rollout status*** to track progress and validate a smooth rolling update completion.

ISSUES AND TROUBLESHOOTING

- ❑ Issue: Pod not creating properly.
- ❑ Fix: We restarted Minikube and re-applied all YAML configurations.

CONCLUSION

In conclusion, we were able to do the following :

- ❑ Successfully deployed and exposed application.
- ❑ Understood scaling and rolling updates.
- ❑ Practiced writing and applying YAML configs.

THANK YOU
