Name: **Vishal S** USN: **1BM21IS206** Section: **7 D**

Q 75.1: Minikube single node cluster

Steps:

- Install Minikube and start a Kubernetes cluster on your local machine.
- Write a Python Flask application that serves a simple API endpoint at /brave-falcon-46, returning the message: {"health": "All systems operational"}.
- Create a Docker image for the Flask application and push to your docker hub public container registry.
- Deploy the application to the Minikube cluster using kubectl with the following requirements:
- Use a Kubernetes Deployment with 2 replicas of the Flask application.
- Expose the Deployment using a Service of type NodePort.
- Configure the Service to expose the application on port 32484 on the Minikube host.
- Verify the deployment by accessing the /brave-falcon-46 endpoint through the exposed NodePort.
- Demonstrate the scaling of the application by increasing the replicas to 3 and verifying the updated Deployment.

1. **Install Minikube** https://minikube.sigs.k8s.io/docs/start/

2. **Start minikube**

   ‘’’ minikube start ‘’’

```
PS C:\Users\visha\OneDrive\Desktop\7th Sem\GreenFinale\Platformio\DevopsTest> minikube start
😄  minikube v1.34.0 on Microsoft Windows 11 Pro 10.0.22635.4660 Build 22635.4660
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.45 ...
🔄  Restarting existing docker container for "minikube" ...
❗  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 v1.31.0 on Docker 27.2.0 ...
🔎  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
```

3. **Create app.py**

   ‘’’ from flask import Flask
   app = Flask(__name__)
   @app.route('/brave-falcon-46')
   def health_check():
   return {"health": "All systems operational"}
   if __name__ == '__main__':
   app.run(host='0.0.0.0', port=5000) ‘’’

4. **Dockerfile**

‘’’ # Use an official Python runtime as a parent image
FROM python:3.8-slim
# Set the working directory in the container
WORKDIR /app
# Copy the current directory contents into the container at /app
COPY . /app
# Install Flask
RUN pip install flask
# Make port 5000 available to the world outside this container
EXPOSE 5000
# Define environment variable
ENV NAME World
# Run app.py when the container launches
CMD ["python", "app.py"] ‘’’

5. **Build Docker image**

‘’’ docker build -t flask-mini ‘’’

```
PS C:\Users\visha\OneDrive\Desktop\7th Sem\GreenFinale\Platformio\DevopsTest> docker build -t flask-m .
[+] Building 18.9s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 496B
 => [internal] load metadata for docker.io/library/python:3.8-slim
 => [auth] library/python:pull token for registry-1.docker.io
```

6. **Push image to dockerhub**

‘’’ docker push vishalsantosh2003/flask-m ‘’’

7. **Deployment.yaml**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app
    spec:
      containers:
      - name: flask-m
        image: vishalsantosh2003/flask-m
        ports:
        - containerPort: 5000
```

8. **Service.yaml**

```yaml
apiVersion: v1
kind: Service
metadata:
  name: flask-app-service
spec:
  type: NodePort
  selector:
    app: flask-app
  ports:
  - protocol: TCP
    port: 5000
    targetPort: 5000
    nodePort: 32484
```

9. **Apply the deployment and service:**
   ”’ kubectl apply -f deployment.yaml
   kubectl apply -f service.yaml ”’

```
PS C:\Users\visha\OneDrive\Desktop\7th Sem\GreenFinale\Platformio\DevopsTest> kubectl apply -f deployment.yaml
>> kubectl apply -f service.yaml
>>
deployment.apps/flask-app-deployment created
service/flask-app-service created
```

```
PS C:\Users\visha\OneDrive\Desktop\7th Sem\GreenFinale\Platformio\DevopsTest> kubectl get services
>>
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
flask-app-service   NodePort    10.101.84.227   <none>        5000:32484/TCP   71m
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP          75m
PS C:\Users\visha\OneDrive\Desktop\7th Sem\GreenFinale\Platformio\DevopsTest>
```

10. **Get minikube ip**
    ”’ minikube ip”’
    Output: 192.168.49.2

11. **Initial deployment with 2 replicas:**

```
PS C:\Users\visha\OneDrive\Desktop\7th Sem\GreenFinale\Platformio\DevopsTest> kubectl get deployments
>> kubectl get pods
>>
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
flask-app-deployment   2/2     2            2           10s
NAME                                   READY   STATUS    RESTARTS   AGE
flask-app-deployment-5768bb5c57-hlx9l   1/1     Running   0          10s
flask-app-deployment-5768bb5c57-qv6rr   1/1     Running   0          10s
```

12. **Scale the deployment to 3 replicas and verify the updated deployment:**

```
PS C:\Users\visha\OneDrive\Desktop\7th Sem\GreenFinale\Platformio\DevopsTest> kubectl scale deployment flask-app-deployment --replicas=3
>>
deployment.apps/flask-app-deployment scaled
```

```
NAME                                    READY   STATUS    RESTARTS   AGE
flask-app-deployment-5768bb5c57-scl4q   1/1     Running   0          11m
flask-app-deployment-5768bb5c57-sqts9   1/1     Running   0          12m
flask-app-deployment-5768bb5c57-tch9s   1/1     Running   0          9m58s
PS C:\Users\visha\OneDrive\Desktop\7th Sem\GreenFinale\Platformio>
```

13. **Output: Access the Flask application at http:/192.168.49.2:32484/brave-falcon-46**

```
StatusCode        : 200
StatusDescription : OK
Content           : {"health":"All systems operational"}

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 37
                    Content-Type: application/json
                    Date: Sat, 04 Jan 2025 05:38:58 GMT
                    Server: Werkzeug/3.1.3 Python/3.9.21

                    {"health":"All systems operational"...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 37], [Content-Type, application/json], [Date, Sat, 04 Jan 2025 05:38:58 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 37
```