Ejercicio 14

Configuración del Entorno

Requisitos Previos

1. Una VM configurada con un sistema operativo basado en Linux (por ejemplo, Ubuntu).
2. Acceso a Internet y permisos de administrador en la VM.
3. Token de autenticación de **Ngrok**.

Instalación de Dependencias

Sigue los pasos para instalar Docker, Minikube, kubectl y Ngrok:

Actualizar el Sistema

```
sudo apt-get update && sudo apt-get upgrade -y
```

Instalar Docker

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
sudo usermod -aG docker $USER
newgrp docker
```

Instalar kubectl

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

Instalar Minikube

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

Instalar Ngrok

```
curl -s https://ngrok-agent.s3.amazonaws.com/ngrok.asc | sudo tee
/etc/apt/trusted.gpg.d/ngrok.asc >/dev/null
echo "deb https://ngrok-agent.s3.amazonaws.com buster main" | sudo tee
/etc/apt/sources.list.d/ngrok.list
sudo apt update && sudo apt install ngrok
```

Iniciar Minikube

Inicia Minikube con Docker como driver:

minikube start --cpus=2 --memory=4096 --driver=docker

Habilitar el Complemento de Ingress
minikube addons enable ingress

Autenticar Ngrok
ngrok config add-authtoken YOUR_NGROK_TOKEN

Despliegue de la Aplicación Flask

Archivos de Configuración

1. flaskapp-deployment.yml: Define las réplicas y variables de entorno de Flask.
2. mysql-deployment.yml y mysql-svc.yml: Configuran MySQL como base de datos con persistencia de datos.
3. nginx-deployment.yml y nginx-svc.yml: Implementan un balanceador de carga con Nginx.
4. ingress.yml: Expone la aplicación Flask al exterior.
5. kustomization.yml: Agrega todos los recursos en el namespace dev.
6. secrets.yml: Almacena las credenciales para MySQL y Flask.

Aplicar los Recursos

1- Navega al directorio con los archivos YAML.

2- Aplica todos los recursos con:
kubectl apply -k .

Verificar el Despliegue

Confirma que los recursos están activos:

kubectl get pods -n dev
kubectl get svc -n dev
kubectl get ingress -n dev

Configuración y Exposición con Ngrok

Obtener el Puerto del Ingress

INGRESS_PORT=$(kubectl -n ingress-nginx get svc ingress-nginx-controller -o jsonpath='{.spec.ports[0].nodePort}')

Ejecutar Ngrok

Exponer el servicio Ingress:
ngrok http $INGRESS_PORT

Ngrok proporcionará una URL pública, como https://flaskapp.ngrok.io.

Pruebas de la Aplicación

Acceso a la Aplicación

Prueba el endpoint raíz (/):

curl https://flaskapp.ngrok.io/
Respuesta esperada:

{"message": "Bienvenido a Flaskapp!!"}

1. Prueba el endpoint /users:

curl https://flaskapp.ngrok.io/users

2. Respuesta esperada: Una lista de usuarios desde la base de datos MySQL.

Validar Balanceo de Carga

Envía varias solicitudes al endpoint raíz y verifica que las respuestas provienen de diferentes réplicas de Flask.

Persistencia de Datos

Accede al pod de MySQL:
1- kubectl exec -it <mysql-pod-name> -n dev -- bash
mysql -uroot -ppass -D flaskapp
2- Inserta un usuario:
INSERT INTO users (name, email) VALUES ('Test User', 'test@example.com');
3- Reinicia el pod de MySQL:
kubectl delete pod <mysql-pod-name> -n dev
4- Consulta nuevamente el endpoint /users para confirmar que los datos persisten.

Preguntas

1. Diferencias entre volúmenes y bind mounts

- **Volúmenes**: Son gestionados por Docker/Kubernetes, ideales para entornos de producción.
- **Bind mounts**: Mapean directorios del host, útiles para desarrollo y depuración.

2. Situaciones para usar cada uno

- **Volúmenes**: Cuando la persistencia debe ser robusta y desacoplada del host.
- **Bind mounts**: Cuando se necesita acceso directo a archivos locales del host.

3. Impacto de eliminar un volumen o bind mount

- **Volúmenes**: Si se elimina el volumen, los datos desaparecen permanentemente.
- **Bind mounts**: Si se elimina el directorio del host, el contenedor no podrá acceder a los datos.

Codigo

flaskapp-deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
  namespace: dev
spec:
  replicas: 3
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:

      labels:
        app: flask-app
    spec:
      containers:
      - name: flask-app
        image: facumiglio/flaskapp:dev
        ports:
          - containerPort: 5000
        env:
          - name: DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: flaskapp-secret
                key: DB_PASSWORD
          - name: DB_USER
            valueFrom:
              secretKeyRef:
```

```
            name: flaskapp-secret
            key: DB_USER
        - name: DB_NAME
          valueFrom:
            secretKeyRef:
              name: flaskapp-secret
              key: DB_NAME
        - name: DB_HOST
          valueFrom:
            secretKeyRef:
              name: flaskapp-secret
              key: DB_HOST
```

2. flaskapp-svc.yml

```
apiVersion: v1
kind: Service
metadata:
  name: flask-app
  namespace: dev
spec:
  type: ClusterIP
```

```
  ports:
    - port: 80
      targetPort: 5000
  selector:
    app: flask-app
```

3. mysql-deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  namespace: dev
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
```

```
spec:
  containers:
  - name: mysql
    image: mysql:8.0
    env:
      - name: MYSQL_ROOT_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mysql-secret
            key: DB_PASSWORD
      - name: MYSQL_USER
        valueFrom:
          secretKeyRef:
            name: mysql-secret
            key: DB_USER
      - name: MYSQL_DATABASE
        valueFrom:
          secretKeyRef:
            name: mysql-secret
            key: DB_NAME
```

```
        ports:
          - containerPort: 3306
        volumeMounts:
          - name: mysql-data
            mountPath: /var/lib/mysql
      volumes:
        - name: mysql-data
          emptyDir: {}
```

4. mysql-svc.yml

```
apiVersion: v1
kind: Service
metadata:
  name: mysql
  namespace: dev
spec:
  ports:
    - port: 3306
      protocol: TCP
  selector:
    app: mysql
```

5. nginx-deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: dev
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
```

```
        - containerPort: 80
      volumeMounts:
        - name: nginx-config
          mountPath: /etc/nginx/conf.d/default.conf
          subPath: default.conf
    volumes:
      - name: nginx-config
        configMap:
          name: nginx-config
```

6. nginx-svc.yml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: dev
spec:
  ports:
    - port: 80
      protocol: TCP
  selector:
```

```
    app: nginx
```

7. ingress.yml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: flask-app-ingress
  namespace: dev
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
    - host: flaskapp.ngrok.io
      http:
        paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: flask-app
              port:
```

```
            number: 80
```

8. secrets.yml

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
  namespace: dev
  type: Opaque
  data:
    DB_USER: cm9vdA==
    DB_PASSWORD: cGFzcw==
---
apiVersion: v1
kind: Secret
metadata:
  name: flaskapp-secret
  namespace: dev
  type: Opaque
  data:
    DB_USER: cm9vdA==
```

```
  DB_PASSWORD: cGFzcw==
  DB_HOST: ZGI=
  DB_NAME: Zmxhc2thcHA=
```

9. kustomization.yml

```
apiVersion: kustomization.config.k8s.io/v1beta1
kind: Kustomization
namespace: dev
resources:
  - flaskapp-deployment.yml
  - flaskapp-svc.yml
  - mysql-deployment.yml
  - mysql-svc.yml
  - nginx-deployment.yml
  - nginx-svc.yml
  - ingress.yml
  - secrets.yml
configMapGenerator:
  - name: nginx-config
    files:
      - default.conf=nginx.conf
```

```
commonLabels:
  env: dev
images:
  - name: facumiglio/flaskapp
    newTag: dev
```

Despliegue

1. Aplicar los recursos:
   kubectl apply -k .

Verificar el despliegue:
kubectl get pods -n dev

kubectl get svc -n dev

2. kubectl get ingress -n dev