🚀Triển khai Flask App từ GitHub lên Minikube bằng GitHub Actions và Helm

---

📥Cấu trúc thư mục

```
repo/
├── Dockerfile
├── .github/
│   └── workflows/
│       └── ci-cd.yml
├── helm/
│   ├── Chart.yaml
│   ├── values.yaml
│   └── templates/
│       ├── deployment.yaml
│       └── service.yaml
└── src/
    └── app.py
```

Điều kiện chuẩn bị

1️⃣Cài đặt môi trường cần thiết (Ubuntu):

```
sudo apt update -y
sudo apt install -y docker.io docker-compose minikube helm kubectl
```

2️⃣Đăng nhập Docker Hub:

```
docker login -u <tên_tài_khoản> -p <token>
```

3️⃣Khởi động Minikube:

```
minikube start --driver=docker
kubectl get nodes
```

4️⃣**Cài đặt GitHub Self-hosted Runner:** - Truy cập: GitHub → Repo → Settings → Actions → Runners → New self-hosted runner - Chọn Linux, copy hướng dẫn cài đặt và chạy trên máy bạn.

Ví dụ:

```
mkdir actions-runner && cd actions-runner
curl -o actions-runner-linux-x64-2.320.0.tar.gz -L https://github.com/
actions/runner/releases/download/v2.320.0/actions-runner-linux-
x64-2.320.0.tar.gz
tar xzf ./actions-runner-linux-x64-2.320.0.tar.gz
./config.sh --url https://github.com/lamluongkhe/sample-web --token
<TOKEN_TỪ_GITHUB>
./run.sh
```

⬜ **Lưu ý:** Không đóng terminal (Ctrl+C sẽ dừng runner). Để tự khởi động khi reboot:

```
sudo ./svc.sh install
sudo ./svc.sh start
sudo systemctl enable actions.runner.lamluongkhe-sample-web.local-
runner.service
```

🏖️File mã nguồn Flask

src/app.py

```python
from flask import Flask
import datetime

app = Flask(__name__)

@app.route('/')
def index():
    return f"Hello (Khe New Version), build time: {datetime.datetime.now()}"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80)
```

🐳Dockerfile

```
FROM python:3.11-slim
WORKDIR /app
COPY src/ /app
RUN pip install flask
EXPOSE 80
CMD ["python", "app.py"]
```

🚀 CI/CD Workflow (.github/workflows/ci-cd.yml)

```
name: CI/CD Test

on:
  push:
    branches:
      - main   # 🔹 Chạy mỗi khi push code lên branch main

jobs:
  build:
    runs-on: self-hosted   # 🔹 Dùng runner local (máy bạn)

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Log in to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}

      - name: Build and Push Docker Image
        id: docker_build
        run: |
          IMAGE_NAME=lamluongkhe/sample-web
          IMAGE_TAG=$(date +%Y%m%d%H%M%S)

          echo "✡️ Building image $IMAGE_NAME:$IMAGE_TAG ..."
          docker build -t $IMAGE_NAME:$IMAGE_TAG .
          docker push $IMAGE_NAME:$IMAGE_TAG

          echo "IMAGE_NAME=$IMAGE_NAME" > image-info.env
          echo "IMAGE_TAG=$IMAGE_TAG" >> image-info.env

          echo "✅ Image pushed successfully: $IMAGE_NAME:$IMAGE_TAG"
          echo "♒ Saved image-info.env:"
          cat image-info.env

      - name: Upload image info artifact
        uses: actions/upload-artifact@v4
        with:
          name: image-info
          path: image-info.env

  deploy:
    runs-on: self-hosted
```

```
    needs: build

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Download image info
        uses: actions/download-artifact@v4
        with:
          name: image-info
          path: .

      - name: Verify and load image env
        run: |
          echo "♒ Loading image info..."
          cat image-info.env
          source image-info.env

          echo "Loaded IMAGE_NAME=$IMAGE_NAME"
          echo "Loaded IMAGE_TAG=$IMAGE_TAG"

          echo "🚳 Deploying to Minikube with Helm..."
          helm upgrade --install sample-web ./helm
            --set image.repository=$IMAGE_NAME
            --set image.tag=$IMAGE_TAG
            --namespace default --create-namespace

          echo "✅ Helm deployment completed!"

      - name: Verify Deployment
        run: |
          echo "⚠️ Checking deployed image..."
          kubectl get deploy sample-web -o yaml | grep image:
          echo "⚠️ Pods status:"
          kubectl get pods -l app=sample-web
```

Helm Chart

helm/Chart.yaml

```
apiVersion: v2
name: sample-web
description: A simple Flask web app
version: 0.1.0
appVersion: "1.0"
```

helm/values.yaml

```
image:
  repository: lamluongkhe/sample-web
  tag: latest
  pullPolicy: Always

service:
  type: NodePort
  port: 80
  nodePort: 30080
```

helm/templates/deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Chart.Name }}
spec:
  replicas: 1
  selector:
    matchLabels:
      app: {{ .Chart.Name }}
  template:
    metadata:
      labels:
        app: {{ .Chart.Name }}
    spec:
      containers:
        - name: {{ .Chart.Name }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          ports:
            - containerPort: 80
```

helm/templates/service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Chart.Name }}
spec:
  type: {{ .Values.service.type }}
  selector:
    app: {{ .Chart.Name }}
  ports:
    - port: {{ .Values.service.port }}
```

```
    targetPort: 80
    nodePort: {{ .Values.service.nodePort }}
```

Kiểm tra & xử lý lỗi port/web không hiển thị

1️⃣Kiểm tra NodePort:

```
kubectl get svc
```

Đảm bảo service `sample-web` hiển thị `30080:80/TCP`

2️⃣Lấy IP Minikube và truy cập:

```
minikube ip
```

Sau đó mở trình duyệt: `http://<IP_MINIKUBE>:30080`

3️⃣Nếu không truy cập được: - Kiểm tra logs:

```
kubectl logs -l app=sample-web
```

- Kiểm tra port container:

```
kubectl exec -it $(kubectl get pod -l app=sample-web -o name) -- curl
localhost:80
```

✅Nếu hiển thị `Hello (Khe New Version)` → Flask app hoạt động tốt.