Dev ops interview CI/CD+deployment

Level	Middle
Skills	Programming
Source	Own

1. Main Task

Task Overview

You are provided with a simple Python web application. Your objectives are:

- **Containerize** the application using Docker.
- **Deploy** the Docker container to a local Kubernetes cluster using Terraform.
- **Set up a CI/CD pipeline** using GitHub Actions to automate testing and building processes.

Application Code

Include the following files in your repository:

`app.py`

```
from flask import Flask
import os

app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello, World!"

@app.route('/env')
def env():
    return f"Environment Variable: {os.getenv('MY_ENV_VAR', 'Not Set')}"
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

requirements.txt

```
Flask==2.0.3
```

test_app.py

```
import os
import pytest
from app import app

@pytest.fixture
def client():
    app.config['TESTING'] = True
    with app.test_client() as client:
    yield client

def test_hello(client):
    response = client.get('/')
    assert response.data.decode() == "Hello, World!"
    assert response.status_code == 200

def test_env(client, monkeypatch):
    monkeypatch.setenv("MY_ENV_VAR", "TestValue")
    response = client.get('/env')
    assert "Environment Variable: TestValue" in response.data.decode()
    assert response.status_code == 200
```

Tasks to Complete

1. **Dockerize the Application**
2. **Terraform Configuration**
    - Write Terraform scripts to manage Kubernetes resources.

- The scripts should:
  - Deploy the Dockerized application to a local Kubernetes cluster.
  - Manage Kubernetes resources such as **Deployment** and **Service**.
  - Pass an environment variable `MY_ENV_VAR` with a value of your choice to the application.

3. **Set Up GitHub Actions CI/CD Pipeline**
   - **On Push to Main Branch:**
     - Run unit tests using `test_app.py`.
     - (Optional)Lint the Python code using a linter (e.g., Flake8).
     - Build the Docker image.
     - Push the Docker image to **GitHub Container Registry** or **Docker Hub**.
   - **On Pull Request:**
     - Run unit tests and linters.
     - Validate Terraform configuration using `terraform validate`.

2. Optional Tasks

💡 You may choose which optional tasks you would like to work on. Not all are feasible within a 2-hour timeframe, so please select those you feel most confident tackling.

Enhance your project by completing one or more of the following optional tasks:

1. **Advanced Kubernetes Configuration**
   - **Implement Horizontal Pod Autoscaling (HPA):**
     - Configure HPA for your application deployment.
     - Set appropriate resource requests and limits.
     - Simulate load to demonstrate the autoscaling behavior.
   - **Implement Network Policies:**

        - Define and apply Kubernetes Network Policies to restrict traffic to your application.
        - Discuss how would you do that in production environment
    - **Implement ConfigMaps and Secrets:**
        - Manage application configuration using ConfigMaps.
        - Suggest a solution to securely manage k8s Secrets
2. **Integrate a Security Scanning Tool into the CI/CD Pipeline**
    - Choose one of the following open-source tools to scan your Docker image for vulnerabilities:
        - **Grype** (Documentation)
        - **Clair** (Documentation)
    - Integrate the chosen tool into your GitHub Actions workflow.
    - Configure the workflow to fail if high-severity vulnerabilities are found.
3. **Implement Automated Rollbacks in CI/CD Pipeline**
    - Enhance your CI/CD pipeline to handle rollbacks.
    - If a build fails or vulnerabilities are detected, automatically revert to the previous stable Docker image.