Optional Step: Master CI/CD with GitHub Actions

Tutor: Haythem Ghazouani

Contents

1 Introduction to the DevOps Philosophy

1.1 What is CI/CD?

In traditional software engineering, integration was a manual, slow process. **Continuous Integration (CI)** is the practice of automating the integration of code changes from multiple contributors into a single software project.

Continuous Deployment (CD) takes it further by automatically deploying those changes to a production environment after they pass all tests.

Pedagogical Context

For this module, we focus on **CI**: ensuring that every push to the repository is "healthy" (no syntax errors, builds successfully).

2 The Core Components of an Action

To use GitHub Actions, you need a `.yaml` configuration. It follows a hierarchy:

1. **Workflow:** The entire process (e.g., "Main CI").
2. **Jobs:** Groups of steps that run in parallel (e.g., "Linting", "Docker Build").
3. **Steps:** Individual tasks (e.g., "Install Python", "Run pip install").
4. **Runners:** The virtual machine (Ubuntu, Windows, or Mac) that executes the code.

3 Detailed Analysis of `main.yml`

Let's break down our configuration file:

3.1 The Trigger (`on:`)

```yaml
on:
  push:
    branches: [ "master" ]
```

This tells GitHub: "Run this whole script ONLY when someone pushes code to the master branch." This prevents running expensive builds on every small draft branch.

3.2 Job 1: Static Code Analysis (Linting)

```yaml
jobs:
  lint-python:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v4
    - name: Set up Python 3.9
```

```
      uses: actions/setup-python@v5
    - name: Lint with flake8
      run: |
        flake8 code/ --count --select=E9,F63,F7,F82 --show-source
  --statistics
```

The Logic: Before building heavy Docker images, we check the code quality. `flake8` scans the `code/` folder for syntax errors. If a student forgot a colon or used an undefined variable, the build will **STOP** here.

3.3 Job 2: Docker Integrity Check

```
build-docker-backend:
  needs: lint-python
  runs-on: ubuntu-latest
  steps:
  - name: Build Backend Docker Image
    run: docker build -t bank-churn-backend:latest -f Dockerfile
  .backend .
```

The Dependency: Note the keyword `needs: lint-python`. This means that if linting fails, Docker build does not even start. This is a "Fail-Fast" strategy.

4 Step-by-Step Monitoring Guide

DevOps Deep Dive: Verification on GitHub UI

1. **Push:** Run `git push origin master`.
2. **Actions Tab:** Open your repository in a browser, click the **"Actions"** tab.
3. **Live Logs:** Click on the active workflow run. You will see a graph of the jobs. Click on a job to see the actual terminal output of the Runner.
4. **The Green Check:** If all steps turn green, your project is officially "Production Ready".

5 Why is this essential for ING-4-SDIA?

In your final project, you will work in teams. CI/CD ensures that:

- No one "breaks" the master branch.
- All team members follow the same coding standards.
- The Docker images always work, avoiding last-minute surprises during the live demonstration.