🧪 FastAPI CI/CD - Guía Paso a Paso

Profesor: Heberth Martinez

Este documento muestra el paso a paso para la creación de un pipeline CI/CD completo para una aplicación FastAPI, con pruebas y Docker.

Requerimientos

Para seguir esta guía, asegúrate de contar con los siguientes requisitos previos:

- Python instalado en tu sistema
- Docker instalado y funcionando correctamente

📁 1. Estructura de Proyecto

```
fastapi-ci-cd-example/
├── app/
│   └── main.py
├── tests/
│   └── test_main.py
├── .pre-commit-config.yaml
├── Dockerfile
├── .dockerignore
├── requirements.txt
├── .github/
│   └── workflows/
│       └── ci-cd.yml
└── README.md
```

✅ 2. Crear entorno virtual e instalar dependencias

```
python -m venv venv
source venv/bin/activate  # En Windows: venv\Scripts\activate
pip install fastapi uvicorn pytest pre-commit
```

Guardar dependencias:

```
pip freeze > requirements.txt
```

🧰 3. Configurar pre-commit

1. Crear el archivo `.pre-commit-config.yaml`:

```
repos:
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v4.4.0
    hooks:
      - id: end-of-file-fixer
      - id: trailing-whitespace
      - id: check-yaml
      - id: check-added-large-files
  - repo: https://github.com/psf/black
    rev: 24.3.0
    hooks:
      - id: black
```

2. Instalar los hooks:

```
pre-commit install
```

3. Ejecutar manualmente en todo el código:

```
pre-commit run --all-files
```

🧪 4. Crear test simple

`tests/test_main.py`:

```
from fastapi.testclient import TestClient
from app.main import app

client = TestClient(app)

def test_read_root():
    response = client.get("/")
    assert response.status_code == 200
    assert response.json() == {"message": "Hello, world!"}
```

🐋 5. Crear Dockerfile

```
FROM python:3.10-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

`.dockerignore`:

```
__pycache__
*.pyc
venv
.git
```

🔥 6. Probar imagen localmente

```
docker build -t {your-username}/fastapi-app .
docker run -p 8000:8000 {your-username}/fastapi-app
```

✅ 7. Configurar GitHub Actions

`.github/workflows/ci-cd.yml`:

```yaml
name: CI/CD Pipeline

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: 3.10

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install -r requirements.txt
          pip install pytest

      - name: Run tests
        run: PYTHONPATH=. pytest

      - name: Build Docker image
        run: docker build -t ${{ secrets.DOCKER_USERNAME }}/fastapi-app .

      - name: Login to DockerHub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}
```

```
      - name: Push Docker image
        run: docker push ${{ secrets.DOCKER_USERNAME }}/fastapi-app
```

Modifica os Secrets de GitHub `(Settings > Secrets and variables > Actions)`

- DOCKER_USERNAME
- DOCKER_PASSWORD

🏁 8. Hacer push y ver el pipeline correr

```
git init
git add .
git commit -m "Initial commit"
git remote add origin <URL-del-repo>
git push -u origin main
```

📝 Entregables

1. Modifica el código de la aplicación FastAPI para agregar un nuevo endpoint.
2. Agrega pruebas unitarias para verificar el correcto funcionamiento del nuevo endpoint.
3. (Punto extra) Investiga cómo desplegar la aplicación en Render.com o Fly.io y despliega la aplicacion