TP 3 — Docker : tester et builder une image FastAPI avec GitHub Actions

🎯 Objectifs pédagogiques

À l'issue de ce TP, l'apprenant sera capable de :

- Créer une **image Docker** pour une API FastAPI
- Comprendre la différence **tests Python vs tests dans Docker**
- Étendre une **GitHub Action** existante
- Builder une image Docker **uniquement si les tests passent**
- Valider une **chaîne CI réaliste (tests → build)**

🔗 Prérequis

- TP 1 et TP 2 terminés
- Docker installé en local
- Dépôt GitHub déjà configuré avec :
  - FastAPI
  - PyTest
  - GitHub Actions (tests.yml)

État du projet au début du TP 3

```
fastapi-tdd-ci/
├── app/
│   ├── main.py
│   └── services.py
├── tests/
│   ├── test_main.py
│   └── test_services.py
├── requirements.txt
├── .github/
│   └── workflows/
│       └── tests.yml
└── README.md
```

Création du Dockerfile

À la racine du projet : `Dockerfile`

```
FROM python:3.11-slim

WORKDIR /app

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

COPY app app
COPY tests tests

CMD ["pytest", "-v"]
```

👉 **Choix pédagogique volontaire** :

- Le conteneur **lance les tests**, pas l'API
- On valide que *l'image est testable*

Test Docker en local (obligatoire en TP)

```
docker build -t fastapi-tests .
docker run --rm fastapi-tests
```

✅ Les tests doivent passer **dans Docker**.

Commit du Dockerfile

```
git add Dockerfile
git commit -m "Add Dockerfile to run tests"
git push
```

⚠️ À ce stade, **la CI ne build pas encore Docker**.

Extension de la GitHub Action (tests + Docker)

Modifier `.github/workflows/tests.yml`

```yaml
name: FastAPI CI

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

jobs:
  test:
    runs-on: ubuntu-latest

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

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: "3.11"

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

      - name: Run PyTest
        run: pytest -v

  docker:
    runs-on: ubuntu-latest
    needs:
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Build Docker image
        run: docker build -t fastapi-tests .
```

Commit & push

```
git add .github/workflows/tests.yml
git commit -m "Add Docker build step after tests"
git push
```

Comprendre la pipeline CI

Push →
  Job TEST (PyTest)
    ↓ (si OK)
  Job DOCKER (docker build)

👉 Si les tests échouent :

- ❌ **Docker ne se build pas**
- ❌ Le pipeline s'arrête

Démonstration pédagogique : casser la CI

Introduire une erreur

services.py
return salary * 0.3

git commit -am "Break logic again"
git push

Résultat attendu

- Job `test` ❌
- Job `docker` 🚫 **skipped**
- Message clair dans GitHub Actions

Correction → pipeline verte

Corriger le code, push → ✅ test + docker build.

🔗 Sources officielles

- GitHub Actions + Docker
  https://docs.github.com/en/actions/use-cases-and-examples/building-and-testing/building-and-testing-docker
- Dockerfile best practices
  https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

- FastAPI + Docker
  https://fastapi.tiangolo.com/deployment/docker/
