TP 2 — CI/CD : Tests PyTest automatiques avec GitHub Actions (FastAPI)

🎯 Objectifs pédagogiques

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

- Versionner un projet FastAPI avec **Git**
- Publier le projet sur **GitHub**
- Configurer une **GitHub Action**
- Lancer automatiquement les **tests PyTest à chaque push**
- Comprendre le rôle d'une **pipeline CI**
- Diagnostiquer un **échec de tests dans GitHub**

🔗 Prérequis

- TP 1 terminé (FastAPI + PyTest fonctionnels)
- Un compte **GitHub**
- Git installé localement
- Python 3.10+

Rappel du projet (hérité du TP 1)

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

Initialisation du dépôt Git

À la racine du projet

git init
git add .
git commit -m "Initial commit - FastAPI with PyTest"

Création du dépôt GitHub

1. Aller sur **https://github.com**
2. Créer un nouveau repository
    - Nom : `fastapi-tdd-ci`
    - Public (ou privé)
    - ❌ Ne pas initialiser avec README

Lier le dépôt distant

git remote add origin https://github.com/<user>/fastapi-tdd-ci.git
git branch -M main
git push -u origin main

Ajout des dépendances

`requirements.txt`

fastapi
httpx
pytest

👉 GitHub Actions utilisera ce fichier pour installer l'environnement.

git add requirements.txt
git commit -m "Add requirements.txt"
git push

Création de la GitHub Action

Arborescence

.github/
└── workflows/
    └── tests.yml

Fichier `.github/workflows/tests.yml`

```
name: FastAPI Tests

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 tests
        run: |
          pytest -v
```

Commit et push

```
git add .github/workflows/tests.yml
git commit -m "Add GitHub Actions CI for PyTest"
git push
```

Vérification de la CI

1. Aller dans le dépôt GitHub

2. Onglet **Actions**
3. Vérifier que le workflow :
    - se déclenche automatiquement
    - installe Python
    - lance PyTest
    - passe au vert ✅

Déclencher volontairement une erreur (test en échec)

Modifier le code métier (`services.py`)

```
def compute_bonus(salary: float) -> float:
    if salary < 0:
        raise ValueError("Salary must be positive")

    # ERREUR VOLONTAIRE
    return salary * 0.2

git commit -am "Break bonus logic"
git push
```

Résultat attendu

- GitHub Actions ❌ **échoue**
- Les logs montrent **quel test casse**
- Aucun déploiement possible tant que les tests ne passent pas

👉 **Message clé** : *la CI protège le code*.

Correction et nouveau push

Correction

```
def compute_bonus(salary: float) -> float:
    if salary < 0:
        raise ValueError("Salary must be positive")

    if salary < 50000:
        return salary * 0.05
    return salary * 0.10

git commit -am "Fix bonus logic"
```

git push

✅ La pipeline repasse au vert.

🔗 Sources officielles

- GitHub Actions – Python
  https://docs.github.com/en/actions/use-cases-and-examples/building-and-testing/building-and-testing-python
- FastAPI – Testing
  https://fastapi.tiangolo.com/tutorial/testing/
- PyTest
  https://docs.pytest.org/
- actions/setup-python
  https://github.com/actions/setup-python
