Exercice CI/CD - Flask & GitHub Actions

Objectifs

1. **Apprendre à exécuter** une application Flask localement.
2. **Exécuter les tests** unitaires pour valider le code existant.
3. **Créer un workflow CI** avec GitHub Actions pour automatiser la validation.
4. **Ajouter du Linting** pour assurer la qualité du code.

Partie 1 : Prise en main locale

1. Installation

1. Créez un environnement virtuel Python (Max/Linux avec Bash, Windows avec Powershell ou GitBash):

   ```
   python3 -m venv .venv
   ```

2. Activez-le :

   Mac/Linux :

   ```
   # Sous Mac/Linux (bash)
   source .venv/bin/activate

   # Sous Windows (Powershell)
   .venv\Scripts\activate
   ```

3. Installez les dépendances du projet :

   ```
   pip install -r requirements.txt
   ```

4. Désactivez l'environnement une fois terminé :

   ```
   deactivate
   ```

2. Exécution de l'application

1. Définissez la variable d'environnement pour Flask (depuis la racine) :

   Mac/Linux (bash) :

   ```
   export FLASK_APP=app/main.py
   ```

   Windows (PowerShell) :

   ```
   $env:FLASK_APP = "app/main.py"
   ```

   Windows (CMD) :

   ```
   set FLASK_APP=app/main.py
   ```

2. Démarrez le serveur :

   ```
   flask run
   ```

3. Testez l'accès dans votre navigateur : http://127.0.0.1:5000.

3. Tests Unitaires

Validez que l'application fonctionne correctement en lançant les tests fournis :

```
python3 tests.py
# ou
python tests.py
```

Si les tests passent, vous êtes prêt pour la suite.

4. Qualité du Code (Linting)

Pour vérifier que votre code respecte les standards de qualité, nous utilisons `flake8`. Lancez la commande suivante :

```
flake8 .
```

Si aucune ligne ne s'affiche, c'est que votre code est parfait ! Sinon, corrigez les erreurs indiquées.

Partie 2 : Intégration Continue (CI) avec GitHub Actions

Votre mission est d'automatiser ce que vous venez de faire manuellement.

Consignes

Créez un fichier de workflow dans `.github/workflows/ci.yml` qui pour chaque **PUSH** et **PULL REQUEST** effectue les actions suivantes :

1. **Checkout** du code.
2. **Configuration de Python** (utilisez `actions/setup-python`).
3. **Installation des dépendances** (`pip install -r requirements.txt`).
4. **Exécution des tests** (`python tests.py`).

Partie 3 : Qualité du Code (Linting)

Une bonne pipeline CI ne se contente pas de tester, elle vérifie aussi la qualité du code.

Consignes

1. Ajoutez une étape de **Linting** à votre workflow CI.
2. Utilisez l'outil `flake8` pour valider le code
3. Faites en sorte que le CI échoue si le linter détecte des erreurs graves.

Ressources Utiles

- Documentation GitHub Actions
- Action Setup Python
- Flake8 Documentation

A vous de jouer !

Demo interface