TP 2
Réalisation d'une pipeline de CI/CD

Enoncé

La société Happy Desk a développé il y a plusieurs années un logiciel permettant d'afficher des recettes de cuisine.
Le logiciel a été développé de manière monolithique et le CTO se rend compte que les mises en production sont longues, complexes et que les erreurs et bugs sont très fréquents.
Le CTO vous demande de créer un prototype simple avec une ou deux pages de manière moderne en mettant en place les bonnes pratiques DevOps à savoir :

- utilisation d'un dépôt git
- mise en place de tests unitaires
- mise en place d'une CI/CD avec les étapes suivantes
    - build de l'application
    - exécution des tests
    - déploiement en production sur Azure Web App

Etapes clés

1. Création d'un dépôt git en ligne sur github
2. Création du projet flask avec le code fourni
3. Création d'une Azure Web App
4. Récupération du Azure Profile
5. Création de la pipeline CI/CD sur Github Actions
6. Tests de la pipeline en faisant des changements de code dans le dépôt git

Livrables

Le lien du dépôt git devra être fourni dans un fichier .txt avec un fichier README.md expliquant les différentes fonctionnalités du projet et explications pour le créer et l'exécuter.

Guide

Fichier app.py

```python
from flask import Flask, request


app = Flask(__name__)


@app.route("/")
def home():
   return "Welcome to the Flask CI/CD Demo! This is V2!"


@app.route("/new-deployment")
def new_deployment():
   return "New deployment from CI/CD"
```

```
@app.route("/info")
def info():
   user_agent = request.headers.get('User-Agent')
   return f"Your user agent is: {user_agent}"


if __name__ == "__main__":
   app.run(debug=True)
```

Fichier test_app.py

```
import unittest
from app import app


class FlaskTestCase(unittest.TestCase):

   def setUp(self):
       self.app = app.test_client()
       self.app.testing = True

   def test_home_status_code(self):
       response = self.app.get('/')
       self.assertEqual(response.status_code, 200)

   def test_info_status_code(self):
       response = self.app.get('/info')
       self.assertEqual(response.status_code, 200)


if __name__ == '__main__':
   unittest.main()
```

Fichier requirements.txt

```
Flask
gunicorn
pytest
flake8
```

Création de l’environnement virtuel

python3 -m venv venv

Activation de l’environnement virtuel
source venv/bin/activate

Installation des dépendances
pip install -r requirements.txt

Lancement des tests unitaires
python -m pytest -v

Lancement de l’application en local
python app.py