Ci cd Exercice

Exercice 1 : Mise en place d'un pipeline CI/CD avec Jenkins et GitLab CI/CD

▼ Étape 1 ↲ Initialisation du projet

- Créez un dépôt Git sur une plateforme comme GitLab ou GitHub et nommez-le par exemple "HelloWorldPipeline".
- Créez un fichier source pour votre programme "Hello World" dans le langage de votre choix :
  - Exemple en Python ↲ Créez un fichier `hello_world.py` et ajoutez-y une ligne pour afficher "Hello, World!".
- Commitez le fichier source sur le dépôt.
- Créez un fichier de dépendances (si nécessaire) pour votre projet :
  - Python : `requirements.txt`
  - Java : `pom.xml`
  - JavaScript : `package.json`
- Listez les dépendances requises pour exécuter le projet, puis commitez ce fichier sur le dépôt.

▼ Étape 2 ↲ Création du pipeline CI/CD avec GitLab CI/CD

- Créez un fichier nommé `.gitlab-ci.yml` à la racine de votre projet.
- Le fichier `.gitlab-ci.yml` doit contenir les étapes suivantes :
  - Une étape pour installer les dépendances du projet (setup).
  - Une étape pour compiler ou exécuter le code (build).
  - Une étape pour lancer les tests unitaires (test).
- Indication ↲ N'oubliez pas de spécifier les stages et d'utiliser des commandes spécifiques à votre langage de programmation dans chaque étape.
  - Commitez le fichier `.gitlab-ci.yml` et observez si le pipeline s'exécute automatiquement dans la section **CI/CD** de votre projet GitLab.

- Corrigez les éventuelles erreurs en consultant les logs générés.

▼ Étape 3 ↲ Création du pipeline CI/CD avec Jenkins

- Créez un fichier nommé `Jenkinsfile` à la racine de votre projet.
- Le fichier Jenkinsfile doit décrire les étapes suivantes :
  - Une étape pour configurer l'environnement du projet √Setup).
  - Une étape pour compiler ou exécuter le programme "Hello World" √Build).
  - Une étape pour exécuter des tests unitaires √Test).
- Indication ↲ Pensez à organiser les étapes dans des blocs nommés (stages) et à utiliser les commandes propres au langage de programmation choisi.
- Configurez un nouveau job Jenkins pour utiliser le Jenkinsfile du dépôt.
- Déclenchez manuellement le job et observez les logs pour vérifier que chaque étape s'exécute correctement.

▼ Exercice 2↲ Manipulation de Branches et Création d'une Structure de Branches

- **Créez un dépôt Git :**
  - Créez un dépôt Git sur GitLab nommé `BranchStructureProject`.
  - Clonez le dépôt sur votre machine locale.
- **Créez un fichier de base :**
  - Dans le répertoire cloné, créez un fichier source tel que `main.py`, `app.js`, ou `app.java`, affichant un simple HelloWorld en console selon le langage choisi. Par exemple, pour Python :
  - Commitez ce fichier dans la branche `main`.
- **Créez la branche de développement :**
  - Créez une branche nommée `develop` à partir de `main` :

- Commitez un changement simple dans cette branche (par exemple, modifiez le message de sortie dans `main.py`).

- **Créez une branche de fonctionnalité :**
  - À partir de `develop`, créez une branche de fonctionnalité nommée `feature/addition` :
  - Dans cette branche, ajoutez une fonction d'addition dans le fichier `main.py` :
  - Commitez ce changement.
- **Créez une branche de correction de bug :**
  - Retournez à `develop` et créez une branche de correction de bug nommée `bugfix/fix-addition` :
  - Corrigez une éventuelle erreur dans la fonction d'addition (par exemple, en ajoutant une vérification de type).
  - Commitez la correction.
- **Fusionnez les branches :**

- Retournez à `develop` :

  Fusionnez la branche de fonctionnalité dans `feature/addition` dans `develop` :

- Ensuite, fusionnez la branche de correction de bug `bugfix/fix-addition` dans `develop` :

- **Préparez la branche finale :**
  - Retournez à main :

    Fusionnez develop dans main :

Étape 3 : Créer une Pull Request et vérifier la structure de branche

- **Créez une Pull Request :**

- Créez une Pull Request (merge request) dans GitLab pour fusionner `develop` dans `main`.
- Vérifiez que tout fonctionne comme prévu.

Vérifiez la structure des branches :

- Utilisez `git log --oneline --graph` pour visualiser la structure des branches. Vous devriez voir quelque chose comme ceci :

```
*   Merge branch 'develop' into main
|\
| * Merge branch 'bugfix/fix-addition' into develop
| * Feature: Add addition function
* | Initial commit
|/
```