TP6 : Conversion d’un Job Freestyle en Pipeline Jenkins (Jenkinsfile)

Objectif

Vous allez analyser un ensemble de jobs freestyle existants, puis les convertir en un pipeline Jenkins en respectant la logique initiale tout en intégrant les bonnes pratiques des pipelines (gestion des erreurs, notifications, parallélisation, gestion des logs et des artifacts, etc.).

Contexte

Les jobs freestyle fournis réalisent plusieurs étapes classiques du cycle de développement d’une application Python. Votre mission est de les convertir en un unique Jenkinsfile qui orchestre ces tâches de manière plus efficace et maintenable.

Étapes à Convertir

1. **Récupération du code** : Utilisez le plugin Git pour cloner le dépôt GitHub contenant l’application Python.
2. **Analyse statique du code** : Utilisez flake8 pour analyser la qualité du code.
3. **Exécution des tests unitaires** : Utilisez pytest pour exécuter les tests unitaires et générer un rapport.
4. **Construction et publication d’une image Docker** : Construisez une image Docker de l’application et poussez-la sur Docker Hub.

Améliorations à Apporter

- **Gestion des erreurs** : Ajoutez des blocs de gestion d’erreurs pour éviter que des défaillances mineures ne bloquent tout le pipeline.
- **Parallélisation** : Exécutez certaines étapes en parallèle pour optimiser le temps d’exécution (par exemple, exécuter les tests en même temps que l’analyse statique du code).
- **Visibilité et logs** : Assurez-vous que chaque étape affiche des logs clairs et enregistre les résultats des tests et des analyses comme artifacts.
- **Sécurité des credentials** : Utilisez les credentials Jenkins pour stocker les informations sensibles comme le mot de passe Docker Hub.

Démarche

1. **Analysez les jobs freestyle** pour comprendre chaque étape et identifier les améliorations possibles.
2. **Créez le Jenkinsfile** en respectant la logique initiale et en appliquant les optimisations.
3. **Testez votre pipeline** en l'exécutant sur Jenkins et en validant son bon fonctionnement.
4. **Validez l'amélioration** en comparant le pipeline avec les jobs freestyle initiaux (temps d'exécution, lisibilité, gestion des erreurs, modularité, etc.).

Retour d'expérience

À la fin, nous échangerons sur les différentes solutions mises en place pour identifier les bonnes pratiques, les difficultés rencontrées et les axes d'amélioration possibles pour des pipelines Jenkins plus robustes et performants.