

Projet CI/CD

Contexte : Vous faites partie d'une équipe de développeurs Java travaillant sur une application critique qui doit répondre à des exigences élevées en matière de qualité, de sécurité et de performance. Pour garantir que chaque modification du code n'introduit pas de régressions, respecte les normes de codage, et ne crée pas de vulnérabilités, votre équipe a décidé d'adopter une approche d'intégration continue (CI) et de livraison continue (CD).

Le projet nécessite la mise en place d'une chaîne CI/CD complète qui automatisera l'ensemble du cycle de développement : depuis l'intégration du code jusqu'à la mise en production, en passant par les tests unitaires, l'analyse de sécurité, la couverture de code et le déploiement automatique. Jenkins a été choisi comme outil central pour orchestrer ces processus.

Objectifs du Projet

L'objectif de ce projet est de concevoir et de mettre en place une chaîne CI/CD robuste et automatisée pour un projet Java. À la fin du projet, les développeurs seront capables de :

1. **Configurer un pipeline CI/CD avec Jenkins** pour un projet Java, automatisant les étapes de build, de test, d'analyse et de déploiement.
2. **Exécuter des tests unitaires** automatiquement à chaque commit pour s'assurer que les nouvelles modifications ne cassent pas le code existant.
3. **Mesurer la couverture de code** avec un outil comme JaCoCo ou autres, afin de garantir que les tests couvrent une partie suffisante du code source.
4. **Effectuer une analyse de sécurité et de qualité du code** avec SonarQube pour détecter les vulnérabilités, les bugs, et les violations des règles de codage.
5. **Déployer automatiquement l'application** dans un environnement de staging ou de production après validation des tests et des analyses.

Ce que les développeurs devront faire

1. Préparation de l'environnement CI/CD :

- Installer et configurer Jenkins sur un serveur dédié ou sur une machine locale.
- Installer Java, Maven, et les plugins nécessaires pour Jenkins (Git, Maven, JaCoCo, SonarQube, etc.).

2. Configuration du pipeline Jenkins :

- Créer un dépôt Git contenant le projet Java, ou utiliser un projet existant.
- Configurer Jenkins pour surveiller ce dépôt et déclencher automatiquement un build à chaque commit ou pull request.
- Définir les différentes étapes du pipeline Jenkins, incluant la compilation du projet, l'exécution des tests unitaires avec JUnit, et la génération des rapports de couverture de code avec JaCoCo.

3. Intégration des outils d'analyse de sécurité et de qualité :

- Configurer SonarQube pour analyser le code source du projet à la recherche de vulnérabilités, de bugs et de mauvaises pratiques.
- Intégrer SonarQube dans le pipeline Jenkins pour que l'analyse soit effectuée automatiquement après chaque build.
- Configurer des règles de qualité dans SonarQube pour imposer un seuil minimal de couverture de code et d'autres critères de qualité.

4. Automatisation du déploiement :

- Configurer une étape de déploiement dans le pipeline Jenkins pour automatiser la mise en production de l'application, soit sur un serveur de staging, soit directement en production.
- Implémenter des notifications et des validations manuelles ou automatiques pour le déploiement.

5. Vérification et amélioration continue :

- Valider le bon fonctionnement du pipeline en effectuant plusieurs commits et en observant les résultats dans Jenkins.
- Analyser les rapports générés (tests unitaires, couverture de code, analyse SonarQube) pour identifier les points d'amélioration du code.
- Affiner le pipeline Jenkins en ajoutant des étapes supplémentaires ou en ajustant les critères de qualité et de sécurité.