TP 4 : Mise en place de tâches parallèles et gestion des erreurs

Objectifs

- Exécuter plusieurs tâches en parallèle dans un pipeline Jenkins.
- Gérer les erreurs et implémenter des actions correctives en cas d'échec.

Prérequis :

- Un Jenkins fonctionnel avec accès au projet GitHub contenant le `Jenkinsfile`.
- Un job pipeline déjà configuré sur Jenkins.

Étape 1 : Création d'un pipeline avec des tâches parallèles

1. **Modifier le `Jenkinsfile` pour ajouter des étapes parallèles**
   - Exemple avec des tâches parallèles (`Build`, `Test Unitaire`, `Analyse SonarQube`) :

```
pipeline {
    agent any
    stages {
        stage('Compilation & Tests') {
            parallel {
                stage('Build') {
                    steps {
                        echo "Compilation en cours..."
                        sh 'sleep 3'  // Simulation du build
                    }
                }
                stage('Tests Unitaires') {
                    steps {
                        echo "Exécution des tests unitaires..."
                        sh 'sleep 2'  // Simulation des tests
unitaires
                    }
                }
                stage('Analyse Qualité') {
                    steps {
                        echo "Analyse statique du code avec
SonarQube..."
                        sh 'sleep 4'  // Simulation de l'analyse
                    }
                }
            }
        }
    }
}
```

2. **Committer et pousser les modifications**

```
git add Jenkinsfile
git commit -m "Ajout de l'exécution parallèle"
git push origin main
```

3. **Exécuter le pipeline dans Jenkins**
   - Observer dans l'interface Jenkins que les tâches s'exécutent en **parallèle**.

Étape 2 : Gestion des erreurs avec des actions post-failure

1. **Modifier le `Jenkinsfile` pour inclure la gestion des erreurs**
   - Ajout d'un bloc `post` pour exécuter une action si une étape échoue :

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                script {
                    try {
                        echo "Compilation en cours..."
                        sh 'exit 1'  // Simulation d'une erreur
                    } catch (Exception e) {
                        echo "Erreur détectée dans le build !"
                        currentBuild.result = 'FAILURE'
                    }
                }
            }
        }
    }
    post {
        failure {
            echo "Le pipeline a échoué, envoi d'une notification..."
            sh 'echo "Erreur détectée" > erreur.log'
            archiveArtifacts artifacts: 'erreur.log', fingerprint:
true
        }
        success {
            echo "Pipeline exécuté avec succès !"
        }
    }
}
```

2. **Committer et pousser les modifications**

```
git add Jenkinsfile
git commit -m "Ajout de la gestion des erreurs"
git push origin main
```

3. **Exécuter le pipeline et observer le comportement**
   - **Forcer une erreur** pour voir si l'action post-failure est bien déclenchée.
   - Vérifier que le fichier `erreur.log` est bien archivé dans Jenkins.