Atelier 5 : Création d'un pipeline CI/CD (3h)

Objectif général :
Cet atelier vous permettra de mettre en place un pipeline CI/CD complet avec **Jenkins**. Vous allez installer et configurer Jenkins via Docker, créer un job Jenkins pour récupérer un code source, puis le convertir en un pipeline Jenkinsfile pour compiler, tester et archiver un artefact.

1. Introduction à l'atelier (5 min)

Objectifs pédagogiques :

- Installer et configurer Jenkins via **Docker**.
- Créer un premier **job Jenkins** qui récupère le code d'un dépôt Git.
- Convertir ce job en un **pipeline Jenkinsfile** pour compiler, tester et archiver un artefact.

2. Exercice 1 : Installation et configuration de Jenkins via Docker (1h)

Objectifs :

- Installer Jenkins en utilisant **Docker**.
- Configurer Jenkins pour qu'il soit accessible depuis un navigateur.

Étapes :

1. **Prérequis :**
   - Assurez-vous d'avoir **Docker** installé sur votre machine. Si ce n'est pas déjà fait, installez Docker depuis le site officiel de Docker.
2. **Lancer Jenkins avec Docker** :
   - Exécutez la commande suivante pour lancer Jenkins dans un conteneur Docker :
   - `docker run -d -p 8080:8080 -p 50000:50000 --name jenkins jenkins/jenkins:lts`
   - Cette commande télécharge l'image Docker officielle de Jenkins et démarre un conteneur avec les ports nécessaires ouverts (8080 pour l'interface utilisateur et 50000 pour la communication avec les agents Jenkins).
3. **Accéder à l'interface Jenkins** :
   - Ouvrez un navigateur et allez à `http://localhost:8080`. Vous devriez voir l'écran de configuration initiale de Jenkins.
4. **Récupérer le mot de passe de Jenkins** :
   - Jenkins génère un mot de passe lors de son premier démarrage. Utilisez la commande suivante pour le récupérer :
   - `docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword`
5. **Compléter la configuration initiale** :
   - Entrez le mot de passe dans l'interface de Jenkins pour terminer la configuration.
   - Installez les plugins recommandés lorsque Jenkins vous le demande.
   - Créez un utilisateur admin pour accéder à Jenkins.

Récapitulatif :

Vous avez installé et configuré Jenkins via Docker. Jenkins est maintenant accessible et prêt à être utilisé pour automatiser des processus CI/CD.

3. Exercice 2 : Créer un premier job Jenkins pour récupérer le code de l'atelier Git (45 min)

Objectifs :

- Créer un job Jenkins qui récupère le code source à partir d'un dépôt Git.

Étapes :

1. **Créer un nouveau job Jenkins** :
    - Depuis l'interface Jenkins, cliquez sur **"New Item"**.
    - Donnez un nom au job (par exemple, **"Pipeline-Atelier-Git"**) et sélectionnez **"Freestyle project"**.
    - Cliquez sur **OK** pour créer le job.
2. **Configurer le dépôt Git** :
    - Dans la section **Source Code Management**, sélectionnez **Git**.
    - Entrez l'URL de votre dépôt Git (par exemple, un dépôt GitHub ou GitLab).
    - Si le dépôt est privé, configurez les informations d'authentification (par exemple, un token d'accès ou un SSH key).
3. **Configurer les étapes de build** :
    - Allez dans la section **Build** et ajoutez une étape de **"Execute shell"**.
    - Par exemple, si vous avez un projet Node.js, vous pouvez exécuter les commandes suivantes pour installer les dépendances :
    - `npm install`
4. **Sauvegarder le job** :
    - Cliquez sur **Save** pour sauvegarder le job Jenkins.
5. **Exécuter le job** :
    - Cliquez sur **Build Now** pour démarrer le job. Jenkins va récupérer le code source du dépôt Git, installer les dépendances, et préparer l'environnement pour le pipeline.

Récapitulatif :

Vous avez créé un job Jenkins qui récupère le code d'un dépôt Git et effectue les premières étapes de préparation, comme l'installation des dépendances.

4. Exercice 3 : Convertir le job en un pipeline Jenkinsfile pour compiler, tester (tests unitaires simples) et archiver un artefact (1h)

Objectifs :

- Convertir un job Jenkins classique en un pipeline Jenkinsfile.
- Ajouter des étapes pour **compiler**, **tester**, et **archiver un artefact**.

Étapes :

1. **Créer un fichier `Jenkinsfile` :**
   - Dans le dépôt de code source, créez un fichier nommé **Jenkinsfile** à la racine du projet.
   - Ajoutez-y le code suivant pour définir un pipeline déclaratif :

```
pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                // Récupère le code source depuis le dépôt Git
                git 'https://github.com/votre-utilisateur/votre-
depot.git'
            }
        }

        stage('Build') {
            steps {
                // Compiler le code (exemple pour un projet Node.js)
                sh 'npm install'
            }
        }

        stage('Test') {
            steps {
                // Exécuter des tests unitaires
                sh 'npm test'
            }
        }

        stage('Archive') {
            steps {
                // Archiver un artefact (exemple pour un fichier
zip)
                archiveArtifacts artifacts: '**/dist/*.zip',
allowEmptyArchive: true
            }
        }
    }
}
```

   - **Explication** :
     - **agent any** : Le pipeline s'exécutera sur n'importe quel agent disponible.

        - **stages** : Le pipeline est divisé en plusieurs étapes : **Checkout**, **Build**, **Test**, **Archive**.
        - **git** : Récupère le code source depuis le dépôt Git.
        - **sh** : Exécute des commandes shell, comme l'installation des dépendances et l'exécution des tests.
        - **archiveArtifacts** : Archive les artefacts générés pendant le processus (par exemple, des fichiers zip).
2. **Configurer le pipeline dans Jenkins** :
    - Retournez dans Jenkins et modifiez le job que vous avez créé précédemment.
    - Sous la section **Pipeline**, sélectionnez **Pipeline script from SCM**.
    - Configurez Jenkins pour utiliser le `Jenkinsfile` depuis votre dépôt Git :
        - **SCM** : Sélectionnez **Git**.
        - **Repository URL** : Entrez l'URL du dépôt Git.
        - **Branch** : Sélectionnez la branche à utiliser (par exemple, `main`).
3. **Exécuter le pipeline** :
    - Cliquez sur **Build Now** pour exécuter le pipeline. Jenkins va récupérer le code, exécuter les étapes de build et de test, et archiver les artefacts générés.
4. **Vérifier les résultats** :
    - Allez dans la section **Build History** pour voir l'historique des exécutions.
    - Cliquez sur un build pour voir les résultats détaillés, y compris les tests exécutés et les artefacts archivés.

Récapitulatif :

Vous avez transformé un job Jenkins classique en un pipeline **Jenkinsfile** déclaratif. Vous avez ajouté des étapes pour compiler, tester (tests unitaires) et archiver les artefacts du projet.

Conclusion de l'atelier

Cet atelier vous a permis de :

- Installer et configurer **Jenkins via Docker**.
- Créer un **job Jenkins** pour récupérer le code d'un dépôt Git et effectuer des actions de base.
- Convertir ce job en un **pipeline Jenkinsfile** pour automatiser le processus de compilation, de tests unitaires et d'archivage des artefacts.

Les compétences acquises vous permettront de configurer des pipelines CI/CD robustes pour automatiser le développement, le test et le déploiement des applications.