Nom et prenom :AKRAM BENYACOUB

Groupe :4IIR

Rapport du TP : Java Pipeline avec Jenkins, Docker et Maven

1. Introduction

Dans le cadre de ce TP, nous avons mis en place un **pipeline d'intégration continue (CI)** pour un projet Java en utilisant les outils **Jenkins, Docker, Maven et GitHub**.
L'objectif principal est **d'automatiser les étapes de clonage, compilation, tests et exécution d'une application Java**.

Ce travail permet de comprendre le rôle du DevOps dans l'automatisation du cycle de développement logiciel.

2. Objectifs du TP

Les objectifs principaux de ce TP sont :

- Comprendre le fonctionnement d'un pipeline CI/CD
- Automatiser le build d'un projet Java avec Maven
- Utiliser Jenkins pour l'orchestration des étapes
- Exécuter les tâches dans un environnement Docker
- Versionner le projet avec GitHub
- Générer des preuves d'exécution du pipeline

3. Environnement de travail

3.1. Environnement matériel

- Ordinateur personnel sous Windows 10

3.2. Environnement logiciel

- **Java JDK 17**

- **Maven 3.9**
- **Docker Desktop**
- **Jenkins (image Docker personnalisée)**
- **Git & GitHub**
- Navigateur Web (Google Chrome)

4. Description de l'architecture

L'architecture du système repose sur les éléments suivants :

- Jenkins est exécuté dans un container Docker.
- Le socket Docker est monté dans le container Jenkins afin qu'il puisse piloter Docker.
- Une image Docker personnalisée `my-maven-git` est utilisée comme agent pour exécuter Maven.
- Le code source est récupéré depuis un dépôt GitHub.
- Le pipeline est défini dans un fichier `Jenkinsfile`.

5. Création de l'image Docker Maven

Une image Docker a été créée pour contenir :

- Maven
- Java
- Git

Contenu du Dockerfile :

```
FROM maven:3.9.9-eclipse-temurin-17

RUN apt-get update && apt-get install -y git && rm -rf /var/lib/apt/lists/*

WORKDIR /workspace
```

Cette image est nommée :

```
my-maven-git:latest
```

6. Configuration de Jenkins avec Docker

Jenkins a été lancé avec :

- Le volume `jenkins_new_home`
- Le socket Docker : `/var/run/docker.sock`

Commande utilisée :

```
docker run -d ^
  --name jenkins_new ^
  -p 8081:8080 ^
  -p 50001:50000 ^
  -v jenkins_new_home:/var/jenkins_home ^
  -v /var/run/docker.sock:/var/run/docker.sock ^
  my-jenkins-docker
```

Les permissions sur le socket Docker ont été corrigées avec :

```
chmod 666 /var/run/docker.sock
```

7. Présentation du Jenkinsfile (Pipeline)

Le pipeline utilisé est basé sur un **agent Docker** :

```
pipeline {
  agent {
    docker {
      image 'my-maven-git:latest'
      args '-v $HOME/.m2:/root/.m2'
    }
  }

  stages {
    stage('Checkout') {
      steps {
        sh 'rm -rf *'
        sh 'git clone https://github.com/simoks/java-maven.git'
      }
    }

    stage('Build') {
      steps {
        dir('java-maven/maven') {
          sh 'mvn clean test package'
          sh 'java -jar target/maven-0.0.1-SNAPSHOT.jar'
        }
      }
    }
  }
}
```

8. Description des étapes du Pipeline

8.1. Étape Checkout

- Suppression de l'ancien workspace
- Clonage du projet Java depuis GitHub

8.2. Étape Build

- Compilation avec :
- `mvn clean test package`
- Exécution de l'application Java :
- `java -jar target/maven-0.0.1-SNAPSHOT.jar`

Chaque étape est exécutée **dans un container Docker isolé**.

9. Résultats obtenus

- Le projet est cloné avec succès.
- La compilation Maven se termine sans erreur.
- Les tests sont exécutés correctement.
- Le fichier JAR est généré et exécuté avec succès.
- Le pipeline Jenkins s'exécute en mode automatique.
- Des captures d'écran du build réussi sont insérées dans le README du dépôt GitHub.

10. Dépôt GitHub

Le projet a été publié sur GitHub selon les consignes du professeur.

Nom du dépôt :

`TPJavaPipeLine-AkramBenyacoub`

Il contient :

- Le projet Java
- Le fichier Jenkinsfile
- Les captures d'exécution
- Le fichier README.md
- Le présent rapport (facultatif)

11. Difficultés rencontrées

Plusieurs difficultés ont été rencontrées durant ce TP :

- Problème de permissions Docker dans Jenkins
- Absence de Docker CLI dans l'image Jenkins
- Erreurs liées au plugin Docker Pipeline
- Problème d'accès au socket Docker

Toutes ces erreurs ont été résolues grâce à :

- L’installation de Docker dans l’image Jenkins
- La configuration correcte des volumes
- La modification des permissions système

12. Conclusion

Ce TP nous a permis de :

- Comprendre la mise en place d’un pipeline CI/CD réel
- Utiliser Jenkins avec Docker dans un contexte professionnel
- Automatiser le build et l’exécution d’un projet Java
- Maîtriser la gestion des erreurs DevOps
- Comprendre l’importance de l’automatisation dans le développement logiciel moderne

Ce travail constitue une base solide pour les projets DevOps futurs.