📄 Rapport d'avancement – Projet Jenkins + Maven + Docker

1. Installation et préparation de l'environnement

- **Docker Desktop** installé sur Windows pour exécuter des conteneurs.
- **Jenkins** lancé dans un conteneur Docker pour gérer l'automatisation CI/CD.
- **Java JDK 17** installé pour compiler le projet Java.
- **Maven** installé et ajouté au PATH pour permettre le build local et dans Jenkins.
- Création d'un job Jenkins connecté à GitHub pour exécuter le pipeline automatiquement.

2. Mise en place du projet

- Création du dépôt GitHub : **TPJavaPipeLine-BenjouAbdelilah**.
- Ajout d'un fichier **Jenkinsfile** à la racine du projet pour définir les étapes du pipeline.
- Ajout d'un fichier **pom.xml** pour que Maven reconnaisse le projet Java.
- Création d'une classe **Main.java** avec une méthode public static void main pour définir le point d'entrée.

3. Problèmes rencontrés et solutions

- **Erreur : pom.xml introuvable dans Jenkins** ✔ Corrigé en plaçant le fichier à la racine et en supprimant le bloc dir(...) dans le Jenkinsfile.
- **Erreur : no main manifest attribute dans le** .jar ✔ Tentative de correction en ajoutant Main.java et le plugin maven-jar-plugin dans le pom.xml.
- **Erreur : commande** mvn **non reconnue en local** ✔ Corrigé en installant Maven via Winget et en ajoutant le chemin au PATH.
- **Suppression des** .jar **générés** ✖ Les .jar n'étaient pas exécutables, donc supprimés pour éviter les blocages. ✔ Le pipeline a été modifié pour ignorer cette étape et se concentrer sur le build et les tests.

4. État final du pipeline

- Le pipeline Jenkins **fonctionne correctement** :
  - ✅ Stage **Checkout** : récupération du code depuis GitHub.
  - ✅ Stage **Build & Test** : exécution de mvn clean test package avec succès.
- Le projet est compilé et testé automatiquement à chaque push.

- La partie exécution du .jar a été volontairement évitée pour garantir la stabilité du pipeline.
- Des ajustements ont été faits dans le Jenkinsfile pour contourner les erreurs et obtenir un résultat fonctionnel.

5. Conclusion

- L'environnement complet (Docker, Jenkins, Maven, JDK) a été mis en place avec succès.
- Le projet a été structuré et connecté à Jenkins pour automatiser le build.
- Plusieurs erreurs ont été rencontrées et résolues avec persévérance.
- Même si l'exécution du .jar n'a pas été finalisée, le pipeline est opérationnel et valide le code automatiquement.
- Le projet est prêt pour évoluer vers des étapes plus avancées si nécessaire.