# TP_Global_Devops (en binôme)

Chaque binôme choisira un **projet JEE open source ou personnel** (ex. gestion d'employés, bibliothèque, e-commerce, etc.) hébergé dans un **repository GitHub**.

Vous allez :

1. Cloner et préparer le projet.
2. Mettre en place une pipeline CI/CD.
3. Intégrer SonarQube pour l'analyse de la qualité du code.
4. Dockeriser l'application.
5. Déployer l'application sur Kubernetes.
6. Mettre en place la supervision avec Prometheus et Grafana.
7. Documenter chaque étape dans le livrable final.

# Étapes détaillées

## Étape 1 : GitHub – Gestion du code source

- Créez un repository privé ou public sur GitHub.
- Poussez le code source JEE dans le repo.
- Configurez une bonne structure de branches : `main` + `develop` + features.

**Questions** :

- Quelle convention de branches avez-vous adoptée ?
- Montrez l'historique de commits et la politique de merge.

## Étape 2 : Jenkins – Intégration continue

- Installez Jenkins localement ou utilisez une instance partagée.
- Configurez un pipeline Jenkinsfile dans le repo.
- Étapes minimales du pipeline :
    1. Cloner le repo
    2. Compiler le projet
    3. Lancer les tests unitaires
    4. Générer le package `.war` ou `.jar`
    5. Déclencher l'analyse SonarQube

**Questions** :

- Le pipeline se déclenche-t-il automatiquement après chaque push ?
- Quelles étapes sont exécutées et avec quels résultats ?
- Capturez le log du build.

## Étape 3 : SonarQube – Qualité du code

- Installez ou utilisez un serveur SonarQube.
- Connectez Jenkins à SonarQube.

- Analysez la qualité du projet et corrigez au moins 3 issues détectées.

**Questions** :

- Quel est votre score de qualité après analyse ?
- Quelles anomalies ont été corrigées ?
- Capturez le rapport Sonar.

## Étape 4 : Docker – Containerisation

- Écrire un Dockerfile pour containeriser votre application JEE.
- Construire l'image et la publier sur Docker Hub.
- Tester le conteneur localement.

**Questions** :

- Quel est le contenu de votre Dockerfile ?
- Quel est le nom et la version de votre image publiée ?
- Donnez la commande de lancement.

## Étape 5 : Kubernetes – Déploiement

- Créer les fichiers YAML nécessaires : `deployment.yaml`, `service.yaml`, `ingress.yaml`.
- Déployer l'application sur Minikube ou un cluster K8s distant.
- Vérifier l'accès à l'application via un Ingress.

**Questions** :

- Combien de pods ont été déployés ?
- Donnez la commande et la sortie de `kubectl get all`.
- Fournissez l'URL de l'application déployée.

## Étape 6 : Prometheus & Grafana – Supervision

- Déployer Prometheus et Grafana dans le cluster.
- Configurer un dashboard qui affiche au minimum :
    - Consommation CPU / mémoire des pods
    - Disponibilité du service
    - Temps de réponse moyen (si applicable)

**Questions** :

- Fournissez une capture de votre dashboard.
- Quelle métrique avez-vous choisie comme KPI principale ? Pourquoi ?
- Quelles alertes pouvez-vous envisager ?

## Étape 7 : Rapport final

Votre livrable doit contenir :

- La description du projet choisi
- Les étapes de la pipeline (avec captures et explications)
- Les configurations clés (Dockerfile, YAML, Jenkinsfile)
- Les dashboards Grafana
- Les réponses aux questions des étapes précédentes
- Une vidéo comprenant toutes les étapes