# Rapport 1 Trinity : DevOps

## Introduction

Le DevOps a été intégré au projet pour garantir une automatisation des pipelines CI/CD tout en assurant la sécurité des applications et des environnements. Ce document détaille les choix technologiques, la structure, les étapes des pipelines, et les tests de sécurité effectués.

## Choix des technologies

Pour la CI/CD, nous avons utiliser Gitlab CI/CD pour la gestion centralisée des pipelines ainsi que pour l'intégration fluide avec GitLab Runners pour l'exécution.

Nous avons ensuite opté pour Docker pour la conteneurisation. Cela nous permet la portabilité des environnements ainsi que l'isolation grâce à des Dockerfiles pour le développement et la production.

Pour la configuration et le déploiement, Ansible semblait être la meilleure option pour ce projet. Cela nous permet l'automatisation de l'installation des dépendances et des configurations.

Pour la gestion de projet, nous avons choisi Jira pour le suivi des tâches et l'organisation des back log.

Afin d'éviter l'utilisation de fichiers .env, nous avons intégrer Doppler, une solution centralisée de gestion des secrets. En environnement local, un fichier .env.local est utiliser spécifiquement pour la base de données en développement.

## Structure du projet

Pour ce premier rendu, nous utilisons deux repositories : un pour le backend et un pour le frontend mobile. Cependant, le DevOps n'est actuellement en place que sur le backend. Le repositorie backend se structure donc comme ceci :

```
backend/
├── src/
├── Dockerfile.dev
├── Dockerfile.prod
├── docker-compose.yml
├── scripts/
│   ├── docker-dev.sh
│   ├── docker-prod.sh
│   ├── local-test.sh
│   └── monitoring/
├── package.json
├── sonar-project.properties
├── tsconfig.json
└── .gitlab-ci.yml
```

## Scripts d’exécution

Dans le dossier backend, nous avons mis en place plusieurs scripts nous permettant d’exécuter des commandes récurrentes plus rapidement :

- docker-dev.sh : pour lancer le docker-compose avec doppler et le .env.local en développement
- docker-prod.sh : pour lancer le docker-compose avec doppler en production
- local-test.sh : exécute les tests du pipeline en local

Ces scripts offrent une simplicité d’utilisation pour l’équipe et assurent une uniformité dans l’exécution des tâches.

## Pipeline CI/CD

Les pipelines GitLab CI/CD sont structurés en plusieurs stages :

1. Lint

   Cette étape utilise une image Node.js pour vérifier la qualité du code avec ESLint. Elle est exécutée pour les branches principales et les merges requests

2. Tests

   Les tests sont exécutés dans un environnement contenant une base de données PostgreSQL. Doppler gère les variables d'environnement nécessaires à la connexion à la base de données.

3. Build

   Le code est compilé pour produire les artefacts finaux (dossier dist/ ). Ces artéfacts sont utilisés pour les étapes de déploiement.

4. Déploiement

   Le déploiement utilise Docker avec le support de docker-compose. Les images sont construites et poussées vers le registre Docker sécurisé, puis déployées dans l'environnement cible.

## Sécurité et qualité

Pour les tests de sécurité, nous avons utilisé SonarQube, pour l'analyse statique du code afin de détecter les vulnérabilités. Snyk pour la surveillance des dépendances du projet afin de signaler les failles de sécurité connues.

Comme cité précédemment, Doppler, afin d'assurer une gestion centralisée et sécurisée des variables sensibles, nous évitant les fuites potentielles dans les fichiers de configuration ou les dépôts GitLab.

Enfin, nous avons mis en place plusieurs solutions de monitoring :

- Prometheus, Grafana et Cadvisor

  Cadvisor récupère les performances des conteneurs Docker, Prometheus récupère les metrics provenant de Cadvisor et Grafana affiche les données sous forme de graphiques, ce qui nous permet de surveiller les performances et d'avoir les alertes.

- OWASP ZAP

  Détecte des failles potentielles de l'API

## Conclusion

L’intégration du DevOps dans le projet Trinity a permis d’améliorer l’automatisation, la sécurité et l’efficacité des processus de développement. La structure mise en place garantit une base solide pour la suite du projet.