MESURES DE SECURITE

1. Introduction

Ce document présente les mesures de sécurité intégrées dans ce projet, avec pour objectif de garantir la confidentialité des données sensibles, l'intégrité des processus, et la disponibilité continue des systèmes. Chaque décision repose sur des pratiques éprouvées pour sécuriser efficacement l'ensemble de l'infrastructure.

La gestion des secrets tels que les API keys, tokens et mots de passe est une priorité, avec des mécanismes rigoureux pour éviter toute exposition ou utilisation non autorisée. Les fichiers sensibles et le repository Git ont été configurés pour empêcher toute fuite, qu'elle soit intentionnelle ou accidentelle. Enfin, des outils et configurations spécifiques sont déployés pour surveiller et prévenir les accès non autorisés, offrant une infrastructure sécurisée et adaptée aux besoins du projet.

2. Gestion des fichiers sensibles

La gestion des fichiers sensibles est essentielle pour garantir la sécurité des données critiques du projet, comme les API keys, identifiants ou tokens. Ces informations doivent être isolées et protégées pour éviter tout risque de fuite.

Fichiers .env

Les fichiers .env sont utilisés pour stocker les variables sensibles de manière sécurisée, séparées du code source. Par exemple, des fichiers distincts comme .env.dev et .env.prod permettent de gérer les environnements de développement et de production de manière isolée. Ces fichiers sont explicitement ajoutés au .gitignore, ce qui empêche leur inclusion accidentelle dans le repository Git.

Les valeurs contenues dans ces fichiers ne sont jamais exposées dans le code, mais sont chargées dynamiquement lors de l'exécution des applications.

Fichiers .git

Pour éviter tout risque lié à l'inclusion accidentelle de secrets dans l'historique Git, des politiques strictes sont mises en place. L'utilisation d'outils comme Git-secrets est fortement recommandée. Cet outil permet de détecter et de bloquer automatiquement les commits contenant des informations sensibles, renforçant ainsi la sécurité dès les premières étapes du développement.

Grâce à ces pratiques, le projet garantit une gestion rigoureuse et sécurisée des données sensibles tout au long de son cycle de vie.

3. Pratiques de sécurisation des secrets

La sécurisation des secrets est une priorité pour protéger les données sensibles utilisées dans les pipelines CI/CD et les environnements d'exécution. Plusieurs pratiques ont été mises en place pour garantir leur confidentialité et leur utilisation sécurisée.

Variables CI/CD

Les secrets tels que les tokens, identifiants, et clés API sont stockés directement dans l'interface CI/CD de GitLab, dans la section Settings > CI/CD > Variables. Ces variables sont configurées comme protected, ce qui limite leur accès aux pipelines exécutés sur des branches protégées, comme dev et main. Elles sont également définies comme masked and hidden, ce qui empêche leur affichage dans les logs des jobs ou leur consultation dans l'interface publique de GitLab. Cette configuration garantit que les secrets restent invisibles pour les développeurs et ne sont accessibles que dans les contextes prévus.

4. Protection contre les accès non autorisés

La protection contre les accès non autorisés repose sur des pratiques strictes mises en place pour garantir la sécurité du projet.

Authentification et autorisation

Les accès aux outils comme GitLab et les serveurs sont sécurisés par une authentification appropriée. Dans GitLab, des permissions spécifiques sont configurées pour restreindre l'accès aux branches critiques (dev et main), afin de garantir que seules les personnes autorisées puissent effectuer des modifications.

Surveillance

L'intégration de Grafana permettra de surveiller les logs afin de détecter les comportements anormaux ou suspectés d'être malveillants. Cet outil contribuera à renforcer la vigilance et la réactivité face aux incidents.

5. Bonnes pratiques

Validation des commits

La validation des commits est un élément clé pour maintenir la qualité et la sécurité du code dans ce projet. Tout changement critique, comme les modifications sur les branches principales (dev et main), doit passer par un processus de revue de code. Les merge requests (MR) sont utilisées pour soumettre ces changements, et au moins un reviewer est requis pour approuver chaque MR avant qu'elle ne soit fusionnée.

Ce processus garantit que chaque modification est soigneusement examinée, réduisant ainsi les risques d'erreurs, de régressions ou de vulnérabilités dans le code. En impliquant plusieurs membres de l'équipe, cette pratique favorise également un partage de connaissances et une meilleure collaboration.

6. Conclusion

Les mesures de sécurité mises en place pour ce projet reflètent une approche rigoureuse et méthodique, adaptée aux enjeux critiques de confidentialité, d'intégrité et de disponibilité. En sécurisant les secrets, en protégeant les accès et en intégrant des

outils de surveillance performants comme Grafana, le projet garantit une infrastructure résiliente et conforme aux meilleures pratiques.

Le recours à des pipelines CI/CD robustes, la gestion stricte des fichiers sensibles, et le processus de validation des modifications assurent une qualité constante et une protection renforcée à chaque étape du développement. Ces pratiques, combinées à un contrôle précis des interactions et des accès, créent un cadre sécurisé qui permet à l'équipe de se concentrer sur l'innovation tout en minimisant les risques.