

ÉVALUATION



MISE EN ŒUVRE D'UN PIPELINE CI/CD COMPLET

Individuel

Contexte

Vous êtes intégré·e dans une équipe DevOps en charge de la mise en place du déploiement automatisé d'une API destinée à une application mobile. Le projet est encore en phase de développement, mais votre mission est de préparer une infrastructure CI/CD complète, scalable, résiliente, incluant :

- La gestion d'infrastructure as code
- Le déploiement automatisé de l'API
- La supervision du système
- Des mécanismes de sauvegarde (snapshots)
- Une stratégie de retour arrière (rollback)
- Un versionnement maîtrisé
- Une gestion Git structurée (GitFlow)



Mission

1. Concevoir une infrastructure as code avec Terraform.
2. Automatiser la configuration des serveurs avec Ansible.
3. Intégrer un pipeline CI/CD avec GitHub Actions, GitLab CI ou Jenkins.
4. Mettre en œuvre une stratégie de branchement Git (GitFlow).
5. Gérer le versionnement sémantique de l'application ou des artefacts.
6. Implémenter des outils de logs et de monitoring.
7. Prévoir une sauvegarde d'état via des snapshots.
8. Concevoir une stratégie de rollback fonctionnelle.

Note : Le fonctionnement de l'API n'est pas l'objectif principal : ce projet vise à mettre en œuvre tout le cycle CI/CD..

Livrables attendus



Un dépôt Git (public) contenant :

1. README.md complet et structuré, avec :

a.  Présentation du projet

- Contexte : brève description de l'API et de l'application mobile.
- Technologies utilisées.

b.  Mise en place du GitFlow

- Schéma ou explication des branches utilisées : main, develop, feature, release, hotfix.
- Captures d'écran de l'historique de commit et des branches.

c.  Pipeline CI/CD (explication + lien vers les fichiers)

- Fichier(s) YAML (GitHub Actions ou autre) avec description de chaque job :
 - Lint
 - Test
 - Build
 - Packaging
 - Déploiement staging
 - Déploiement production
 - Snapshot
 - Rollback

Livrables attendus



d. Packaging et versionning

- Description du processus de versionnement sémantique (SemVer) utilisé.
- Utilisation de git tag, ou équivalent.
- Dépôt ou stockage des artefacts (ex. GitHub Releases, Nexus, etc.)

e. Gestion des secrets et environnements

- Méthode utilisée (GitHub Secrets, .env, etc.)
- Séparation staging / production
- Bonnes pratiques suivies (sans exposer les secrets eux-mêmes).










f. Tests et logs

- Capture ou lien vers l'exécution du pipeline avec les étapes visibles.
- Exemple de log d'erreur ou de réussite commenté.

Livrables attendus



g.  Captures d'écran obligatoires (avec légende)

1.  Exécution complète du pipeline CI/CD
2.  Interface de staging (déployée)
3.  Interface de production (déployée)
4.  Vue des branches Git (GitHub ou autre)
5.  Historique de commits (main, develop)
6.  Tag Git/version utilisée
7.  Dashboard/logs de monitoring
8.  Déclenchement ou planification de snapshot
9.  Restauration ou procédure de rollback + état restauré

h.  Procédures documentées

- Déploiement
- Procédure de restauration (rollback) claire
- Plan de versionnage et tag

Livrables attendus



2. Structure du dépôt Git

- api/ : code de l'API REST (Node, Django, etc.)
- terraform/: Scripts Terraform (infra)
- ansible/: Rôles et playbooks Ansible
- .github/workflows/ : fichiers YAML du pipeline (ou équivalent selon CI)
- monitoring/: Logs & supervision
- rollback/ : script ou procédure de restauration
- snapshots/ : fichiers ou configuration de snapshot si applicable
- .git/: Historique Git avec GitFlow
- tags/: Versionnement sémantique (ex: v1.0.0)
- README.md : documentation complète

✓ Critères d'évaluation



Critère	Points
Infrastructure avec Terraform	3 points
Configuration avec Ansible	3 points
Pipeline CI/CD complet	3 points
Logs et monitoring	2 points
Snapshots (sauvegardes)	2 points
Rollback (restauration)	2 points
GitFlow (structure, branches, historique)	2 points
Versionnement sémantique	1 point
Documentation claire (README)	2 points