Rapport Devops

Mise en place d'une CI/CD

Equipe : Immaculée Bahoundje, Valentin Mur, Elodie Ratovoherinjanahary

SOMMAIRE

1. Choix de l’application

Pour l’application, nous avons choisis de créer une calculatrice basique sur python. Nous avons fait ce choix pour pouvoir nous focaliser sur l’environnement de cette application et donc la chaîne en elle-même tout en ayant une application qui pouvait être tester et déployer.

2. Mode de travail

Dans le cadre de notre projet, nous avons décidé d'adopter une approche de travail en multibranche sur GitLab. Cette décision a été prise dans le but de favoriser une bonne différenciation entre les différentes fonctionnalités et parties de la chaîne, et pour que chacun puisse travailler sur sa partie.
Les détail

Nos branches

- deploy :
    - contient tout ce qui concerne les instances
- appli
    - contient l’application et les tests
- monitoring
    - contient les moyens de faire une surveillance de l’application

Le travail en multibranche aussi nous a permis de faire des merge request tout en faisant le code review de chacun via des commentaires et en attendant l’approbation d’au moins une autre personne du projet, garantissant ainsi une meilleure qualité et une meilleure traçabilité des modifications.

3. Choix des technologies

Pour la création des instances nous avons choisis **terraform,** on avait plus d’affinité pour cette dernière ce qui a motivé notre choix. nous avons ensuite utilisé **ansible** pour automatiser la configuration des serveurs qu'on veut déployer sur nos instances créé avec terraform.

Pour les tests des applications, vu que l'application en elle-même est codée en python, nous avons choisi des outils qui supportent le langage et **unittest** est une bibliothèque qui est largement utilisée dans la communauté Python et connue pour écrire des tests unitaires.

Pour les tests de performance, dans les recherches on a trouvé **locust** comme outil d'analyse de performance qui soit adapté pour du code en python et qui soit intégrable facilement dans des pipelines de CI/CD. Locust est surtout utilisé pour effectuer des tests de charge et de performance sur des applications Python, et permet de simuler des utilisateurs et de générer des charges sur l'application, on peut ensuite mesurer ses performances et d'identifier les éventuels goulots d'étranglement. Les tests peuvent être exécutés automatiquement à chaque déploiement.

Bandit est un outil d'analyse statique du code spécifiquement conçu pour identifier les vulnérabilités de sécurité dans le code Python. Il recherche des motifs et des pratiques de codage potentiellement risqués qui pourraient conduire à des failles de sécurité.Facilité d'utilisation. On l' a choisi car il était facile à installer et à utiliser et il fournit une interface en ligne de commande simple qui peut être intégrée facilement dans un processus de construction ou d'intégration continue tout en fournissant des résultats d'analyse très intéressant.

SonarQube est une plateforme open-source de gestion de la qualité du code et d'analyse statique. Elle permet de détecter les problèmes de qualité du code, les vulnérabilités de sécurité, les erreurs de conception et autres problèmes potentiels lors de la phase de développement d'un projet. On l'a choisi car comme notre application (calculatrice) n'a pas été conteneurisée, on a quand même voulu faire une partie release dans notre CI en déployant Sonarqube qu'on va conteneurisé dans un ECR.

4. Problème rencontrés

Malheureusement, lors de notre projet, nous avons rencontré des difficultés pour intégrer la partie Monitoring à notre pipeline en temps voulu. Dans un premier temps, nous avons tenté de le faire en utilisant un serveur local dédié à Prometheus et Grafana, mais nous avons rencontré des problèmes techniques qui ont empêché son bon fonctionnement.

En effet, lors de notre projet, nous avons constaté que GitLab ne permettait pas d'ajouter un tableau de bord Grafana en utilisant un serveur local hébergé en localhost. Cette limitation de GitLab nous a posé des problèmes lors de notre tentative d'intégration de la surveillance dans notre pipeline.

L'idée initiale était de créer et de configurer nos tableaux de bord Grafana localement, puis de les importer dans GitLab pour une meilleure intégration avec notre pipeline de développement. Cependant, nous avons découvert que GitLab ne prenait pas en charge les tableaux de bord Grafana hébergés en localhost, cela a compliqué notre tâche d'intégration de la surveillance dans notre pipeline.

Nous avons considéré des options telles que l'hébergement de notre serveur de monitoring sur une infrastructure cloud ou l'utilisation d'un service tiers compatible avec GitLab.

Malheureusement, en raison de contraintes de temps et de ressources, nous n'avons pas pu mettre en place une solution alternative à temps pour inclure pleinement la partie

Nous avons déployé un **SonarQube** qu'on a conteneurisé dans un ECR sur le tenant AWS, le déploiement en local fonctionne bien. Mais on a une erreur pour le déploiement sur la CI avec la commande d'authentification << `aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 304757654481.dkr.ecr.us-east-1.amazonaws.com` >>.

P.S: Monsieur, on a essayé de débuguer ensemble le problème jeudi mais malheuresement l'erreur n'a pas pu être résolue. Vous m'avez demandé de vous le notifier au cas où vous allez l'oublier.