2023-2024

Projet de pipelines CI/CD

Le module de Pipelines et livraisons continues consiste en un projet s'appuyant sur ce que vous avez appris dans les TPs DevOps précédents afin d'automatiser les actions que vous avez réalisées, dans des pipelines.

Pour réaliser les pipelines, nous allons nous appuyer sur Gitlab CI. Mais si vous avez déjà utilisé un autre système de CI/CD (Github Actions, ...), vous pouvez tout à fait choisir de l'utiliser.

Installation d'un runner Gitlab CI

Cette installation peut se faire :

- manuellement comme sur l'instance EC2 (type t2.micro) que vous aviez créée dans les premiers TPs. En suivant la documentation officielle de Gitab CI : https://docs.gitlab.com/runner/executors/
- Ou bien via terraform en suivant les instructions suivantes : https://gitlab.com/efrei-devops/pipelines/infra/gitlab-runner-infra

Notez bien que les ressources d'un Lab AWS sont automatiquement arrêtées au bout de quelques heures. Il faudra bien redémarrer votre Lab pour que le runner Gitlab soit à nouveau disponible dans vos pipelines.

Versionning du code

Les TPs du cours Devops nous ont permis d'écrire du code:

- Ansible: Pour automatiser l'installation d'un serveur apache httpd, choisir un port d'écoute et installer un site web statique.
- Packer: Pour la création de l'AMI contenant le site web statique.
- Terraform: Pour automatiser la création de tout le réseau AWS + l'installation du site web statique sur un Auto scaling group et éventuellement son exposition via un Load Balancer.

Tout le code développé **doit être déposé** sur un dépôt git afin que les pipelines puissent le cloner et l'exécuter.

States Terraform

ATTENTION : Les states terraform doivent être obligatoirement sauvegardés. Le cas le plus courant est d'utiliser un bucket S3 que vous pouvez créer dans votre Lab. D'autres solutions existent : gitlab terraform state, terraform cloud, ...

Travail minimal attendu

Plusieurs pipelines sont nécessaires.

Pipeline de build applicatif

Ce pipeline construit une AMI contenant :

- soit le même exemple de serveur apache httpd et un site web statique
- soit une autre application de votre choix.

Ce pipeline pourra prendre qqs paramètres d'entrée comme :

- Le nom du projet: utile pour construire le nom final de l'AMI.
- Le port sur lequel écoute le serveur httpd et qui nous permettra d'accéder au site web

Pipeline d'Infrastructure

Ce Pipeline permet de créer tous les composants d'infrastructure comme le réseau (un nouveau vpc, des subnets, ...) qui vont accueillir l'application.

Pipeline de déploiement de l'application

Ce Pipeline permet de déployer l'application en exploitant les ressources créées par les deux pipelines plus haut.

Cleanup / Destruction des ressources

Pour chacun des pipelines plus haut, vous pouvez soit :

- créer un pipeline de destruction
- soit ajouter des jobs dans les pipelines précédents afin de détruire les ressources.

Exemple d'outils :

- aws-amicleaner pour supprimer les AMI et snapshots facilement

Axes d'amélioration

- Les pipelines sont nombreux, et sont lancés manuellement dans un ordre spécifique (Buid Applicatif puis Infrastructure puis déploiement Applicatif). Comment faire pour simplifier leur lancement ?
- Si vous avez construit des pipelines de Livraison continue (Continuous Delivery), comment faire pour les transformer en pipelines de Déploiement continu (Continuous Deployment) ou inversement ?
- Créer un workflow "réel" avec plusieurs environnements :
  - déployer d'abord l'infrastructure et l'application vers un environnement de développement (ajouter des tests de validation ?)
  - puis les déployer vers un environnement de production
- Ajouter des tests automatisés
  - des tests unitaires sur le code applicatif
  - des tests d'intégration/end-to-end (e2e) sur votre application déployée
- Ajouter des contrôles de qualité de code :
  - terraform format check
  - securité avec tfsec or checkov
  - sur le code applicatif
- Build et deploy de votre application comme une image Docker, sur EC2, ECS ou dans un cluster Kubernetes (EKS on amazon) au lieu d'une AMI
- Utiliser Terraform Cloud pour une gestion plus avancée de terraform (state, ...)
- Utiliser un autre system de CI/CD (Github Actions, Circle CI, Flux, ArgoCD, Tekton, ...)

Conseils

- déployer et valider le fonctionnement du Gitlab Runner avec un projet de test basique dans le groupe gitlab que vous avez créé
- choisir une application qui servira à votre projet. Il peut s'agir du site Web utilisé dans les TPs ou bien d'une autre application opensource ou bien d'une application que vous avez développée (dans un autre cours ou en dehors des cours), ...
- Représentez l'architecture de cette application sur AWS. En utilisant par exemple https://app.diagrams.net/
- Prenez le temps de définir et représenter vos pipelines, leurs enchaînements, leurs variables, ...