Infrastructure as Code (IaC) et Introduction à Terraform

Introduction

Aujourd'hui, les entreprises déploient de plus en plus d'applications sur différentes plateformes cloud comme AWS, Azure, Google Cloud Platform (GCP) ou encore sur des infrastructures on-premise.

Imaginons que vous êtes un ingénieur DevOps travaillant pour l'entreprise **HooYia** et que vous devez déployer **300 applications** sur plusieurs plateformes. Vous devez donc créer et gérer différentes ressources comme des instances EC2, des bases de données RDS et des espaces de stockage S3.

Pour gérer ces déploiements efficacement, l'automatisation est indispensable.

Problématique

1. **Multiplicité des outils** : Pour chaque fournisseur cloud, il existe des outils spécifiques pour automatiser le déploiement.
    - AWS : AWS CLI, AWS CloudFormation
    - Azure : Azure Resource Manager (ARM)
    - On-premise : OpenStack Heat Templates
2. **Incompatibilité des solutions** : Si votre manager vous demande de migrer vos applications de AWS vers Azure ou une infrastructure on-premise, tous les scripts déjà écrits pour AWS devront être réécrits en utilisant les outils de la nouvelle plateforme.
3. **Complexité de l'apprentissage** : Les entreprises utilisent souvent un environnement hybride (par exemple, 30% AWS, 50% Azure). L'ingénieur DevOps doit donc apprendre plusieurs outils pour gérer les déploiements sur différentes plateformes, ce qui augmente la complexité et le temps de formation.

Solution : Terraform

Terraform, développé par **HashiCorp**, est une solution **d'Infrastructure as Code (IaC)** qui permet d'automatiser la gestion des infrastructures cloud de manière **agnostique** vis-à-vis du fournisseur.

Avantages de Terraform

- **Un seul langage pour tous les cloud providers** : Avec Terraform, vous n'avez pas besoin d'apprendre plusieurs outils différents (AWS CloudFormation, Azure RM, etc.). Il suffit d'écrire des scripts Terraform pour déployer les ressources sur n'importe quelle plateforme cloud.
- **Facilité de migration** : Si vous souhaitez migrer d'AWS vers Azure ou On-Premise, il suffit de modifier quelques configurations Terraform au lieu de réécrire tous les scripts.
- **Automatisation et cohérence** : Terraform assure que toutes les infrastructures sont déployées de manière reproductible et cohérente.

Concept d'API as Code

Terraform fonctionne sur le principe de **l'API as Code**, c'est-à-dire qu'il convertit les demandes d'infrastructure en appels API vers les différents fournisseurs cloud.

Exemple d'API

- Lorsque vous tapez **www.google.com** dans un navigateur, ce n'est pas une API.
- En revanche, si vous écrivez un script pour interroger Google et récupérer des résultats sans interface graphique, c'est une API.
- De même, Terraform utilise des appels API pour créer des ressources cloud.

Conclusion

Avec **Terraform**, vous pouvez automatiser la gestion des ressources sur **n'importe quel fournisseur cloud** (AWS, Azure, GCP, On-Premise, etc.). Il permet de simplifier l'infrastructure en offrant une solution unique au lieu d'apprendre plusieurs outils. Terraform est un incontournable pour les ingénieurs DevOps souhaitant gérer des infrastructures cloud efficacement.