Introduction à l'Infrastructure as Code (IaC)

Objectifs

À la fin de cette séance, vous serez capable de :

- Comprendre les limites de la gestion manuelle d’infrastructure
- Expliquer le concept d’Infrastructure as Code
- Distinguer approche déclarative et impérative
- Identifier les différences entre Terraform et Ansible

1. Problématique : gestion traditionnelle

Dans un environnement classique (on-premise ou cloud), l’infrastructure est souvent :

- configurée manuellement (console, SSH)
- difficile à reproduire
- source d’erreurs humaines
- peu documentée

Exemples de problèmes :

- “Ça marche sur ma machine”
- configurations différentes entre dev / prod
- perte de temps lors des déploiements
- difficulté de rollback

2. Définition de l’Infrastructure as Code

L’Infrastructure as Code (IaC) consiste à :

- décrire l’infrastructure sous forme de code
- versionner ce code (Git)
- automatiser les déploiements

Exemple :

Au lieu de créer une VM à la main :

```
→ on écrit un fichier de configuration
```

Avantages :

- reproductibilité
- traçabilité
- automatisation
- réduction des erreurs

3. Déclaratif vs Impératif

Approche impérative

On décrit les étapes :

- créer un serveur
- installer nginx
- démarrer le service

Exemple :

- scripts bash
- Ansible (en partie)

Approche déclarative

On décrit l'état final souhaité :

- "je veux un serveur avec nginx installé"

L'outil s'occupe des étapes.

Exemple :

- Terraform

4. Terraform vs Ansible

Outil	Type	Usage principal
Terraform	Déclaratif	Provisionnement infrastructure
Ansible	Impératif	Configuration des machines

Terraform :

- crée les ressources (EC2, réseau, etc.)
- gère l'état (state)

Ansible :

- configure les serveurs
- installe des logiciels
- déploie des applications

5. Workflow DevOps typique

- Terraform → crée l'infrastructure
- Ansible → configure les serveurs
- Application → déployée ensuite

Schéma logique :

```
Infrastructure → Configuration → Application
```

6. Bonnes pratiques

- Tout versionner (Git)
- Ne jamais modifier à la main en production
- Utiliser des variables
- Tester en environnement de dev

À retenir

- IaC = infrastructure gérée comme du code
- Terraform → provisioning
- Ansible → configuration
- Objectif : automatiser, fiabiliser, industrialiser