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
- Comprendre le rôle de Terraform et Ansible dans un workflow réel

1. Mise en situation

Vous arrivez dans une entreprise.

Un développeur vous dit :

```
"Pour lancer l’environnement :
 - démarre un conteneur
 - ouvre le port 8080
 - copie les fichiers à la main"
```

Un autre développeur fait différemment.

Résultat :

- environnements différents
- bugs difficiles à reproduire
- perte de temps

Question :

- Quel est le principal problème ici ?

2. Problème : gestion manuelle

En pratique, sans IaC :

- actions manuelles (console, SSH)
- dépend de la personne
- pas versionné

Exemple concret :

- un admin ouvre un port “temporairement”
- personne ne le documente
- faille de sécurité

Autre exemple :

- en prod : port 8080
- en dev : port 3000
- bug uniquement en prod

Question :

- Pourquoi ces problèmes sont difficiles à corriger ?

3. Idée clé : traiter l'infrastructure comme du code

Principe :

- on écrit un fichier
- on versionne (Git)
- on exécute

Analogie :

- avant : cuisine "à l'œil"
- maintenant : recette écrite

Exemple :

Avant :

- cliquer dans une interface
- lancer des commandes à la main

Après :

- fichier versionné
- reproductible à l'identique

4. Exemple simple

Objectif :

- lancer un serveur web

Sans IaC :

- docker run -p 8080:80 nginx
- copier les fichiers à la main

Avec IaC :

- un fichier Terraform crée le conteneur
- un playbook Ansible configure le contenu

Résultat :

- même résultat pour toute l’équipe
- reproductible
- automatisable

5. Déclaratif vs Impératif

Approche impérative

On donne les étapes :

- installer nginx
- démarrer le service

Problème :

- dépend de l’état initial
- plus fragile

Approche déclarative

On décrit le résultat :

- “je veux un serveur web accessible”

L’outil décide :

- quoi créer
- quoi modifier

Question :

- Pourquoi le déclaratif est plus adapté au cloud ?

6. Terraform vs Ansible

Terraform :

- crée l’infrastructure
- sait ce qu’il a créé (state)

Ansible :

- configure les machines
- agit directement sur le système

Exemple concret :

- Terraform → crée un conteneur
- Ansible → copie une page HTML dedans

7. Workflow réel

Dans une entreprise :

- Terraform → infrastructure
- Ansible → configuration
- CI/CD → déploiement applicatif

Chaîne :

```
Code → Infra → Config → App
```

8. Transition vers le TD

Dans le TD :

- vous allez créer un conteneur avec Terraform
- puis le modifier avec Ansible

Objectif :

- comprendre la séparation des rôles
- observer les limites si on mélange tout

9. À retenir

- IaC = automatiser l'infrastructure
- réduire les erreurs humaines
- rendre les environnements reproductibles
- Terraform et Ansible sont complémentaires