**terraform import ?**

terraform import est une commande qui permet d'importer des ressources d'infrastructure déjà existantes (créées manuellement ou par un autre outil) dans le fichier d'état de Terraform (le terraform.tfstate). Cela signifie que Terraform peut ensuite gérer ces ressources comme s'il les avait créées lui-même.

- **Objectif principal** : Synchroniser une infrastructure existante avec le code Terraform sans avoir à la recréer.
- **Limitation importante** : terraform import ne génère pas automatiquement le code HCL (HashiCorp Configuration Language) correspondant à la ressource importée. Tu dois écrire ce code manuellement dans tes fichiers .tf pour qu'il corresponde à la ressource importée.

---

**Quand utiliser terraform import ?**

1. **Migration d'une infrastructure existante** : Si tu as créé des ressources via une console (ex. AWS Console) ou un autre outil, et que tu veux maintenant les gérer avec Terraform.
2. **Collaboration** : Si une ressource a été créée par un collègue hors de Terraform, tu peux l'importer pour l'intégrer dans ton workflow.
3. **Récupération après perte d'état** : Si le fichier terraform.tfstate est perdu ou corrompu, tu peux réimporter les ressources existantes.

**<ADDRESS>** : L'identifiant Terraform de la ressource dans ton code (ex. aws_instance.my_instance).
**<ID>** : L'identifiant unique de la ressource dans le fournisseur (ex. i-1234567890abcdef0 pour une instance EC2 sur AWS).

```
terraform import aws_instance.my_instance i-1234567890abcdef0
```

## Étapes pour utiliser terraform import

1. **Identifier la ressource existante** : Trouve l'ID de la ressource dans ton fournisseur (ex. AWS, GCP, Azure).
2. **Écrire le code Terraform correspondant** : Ajoute une définition de ressource vide ou partielle dans ton fichier .tf.
3. **Exécuter l'importation** : Utilise terraform import pour lier la ressource au fichier d'état.
4. **Vérifier et ajuster** : Compare l'état importé avec ton code et ajuste les paramètres si nécessaire.
5. **Gérer la ressource** : Une fois importée, Terraform peut modifier ou détruire la ressource.

## Exemple pratique : Importer une instance EC2

Imaginons que tu as une instance EC2 créée manuellement sur AWS, et tu veux la gérer avec Terraform.

**Prérequis**

- Terraform installé.
- AWS configuré (provider AWS dans ton code).
- Une instance EC2 existante avec l'ID i-0abcd1234efgh5678.

**Étape 1 : Écrire le code Terraform**

Crée un fichier main.tf :

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}

provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "my_instance" {
  # Les attributs seront remplis après l'importation
  ami           = "" # Placeholder
  instance_type = "" # Placeholder
}
```

**Étape 2 : Initialiser Terraform**

```
terraform init
```

**Étape 3 : Importer la ressource**

Exécute la commande suivante pour importer l'instance EC2 :

terraform import aws_instance.my_instance i-0abcd1234efgh5678

Terraform mettra à jour le fichier terraform.tfstate avec les détails de cette instance.

**Étape 4 : Vérifier l'état**

Utilise cette commande pour voir l'état importé :

terraform show

**Étape 5 : Mettre à jour le code**

Ajuste ton main.tf pour refléter les attributs importés :

```
resource "aws_instance" "my_instance" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
  tags = {
    Name = "my-imported-instance"
  }
}
```

**Étape 6 : Vérifier la cohérence**

Exécute :

terraform plan

Si tout est correct, Terraform indiquera qu'aucune modification n'est nécessaire. Sinon, ajuste ton code pour correspondre à l'état importé.

## Options utiles de terraform import

- -var 'key=value' : Spécifie des variables si ton code en dépend.
- -state-out=PATH : Écrit l'état dans un fichier différent.
- -config=PATH : Pointe vers un dossier spécifique contenant les fichiers .tf

```
terraform import -var "region=us-east-1" aws_instance.my_instance i-0abcd1234efgh5678
```