

Cours sur AWS CloudFormation (CFT) – Infrastructure as Code avec AWS

Objectif du cours

Comprendre comment utiliser **AWS CloudFormation** pour créer et gérer l'infrastructure AWS en tant que code (**IaC – Infrastructure as Code**), comparer CFT avec **AWS CLI** et **Terraform**, découvrir les composants d'un template CFT, et apprendre à rédiger et utiliser un template YAML.

1. Introduction à AWS CloudFormation

◇ Qu'est-ce qu'AWS CloudFormation (CFT) ?

CloudFormation est un service AWS qui permet de **modéliser et provisionner des ressources AWS** à l'aide de **fichiers de configuration (template)** en **YAML ou JSON**. C'est une solution **déclarative** (on décrit l'état final voulu de l'infrastructure).

Avantage principal :

Créez, mettez à jour et supprimez **automatiquement** des stacks de ressources AWS (EC2, S3, RDS, VPC, etc.) à l'aide de fichiers versionnables.

2. Différences entre AWS CLI et AWS CloudFormation

Fonctionnalité	AWS CLI	AWS CloudFormation (CFT)
Type	Outil en ligne de commande	Service de gestion d'infrastructure
Approche	Impérative (étape par étape)	Déclarative (résultat final décrit)
Utilisation typique	Actions rapides (Lister S3, EC2)	Déploiement d'architecture complète
Langage	Commandes shell	YAML ou JSON
Versionnable	Non directement	Oui (git-friendly)
Détection de dérive	Non	Oui (Drift Detection)
Gestion de dépendances	Manuelle	Automatique via CFT

□ 3. Cas d'usage : Quand utiliser CLI ou CFT ?

- ◇ **CLI** : Pour des **actions ponctuelles ou rapides**, ex. :
 - `aws s3 ls`
 - `aws ec2 describe-instances`
 - ◇ **CloudFormation** : Pour déployer **plusieurs ressources interconnectées** :
 - Un VPC complet (avec subnets, route tables, gateways)
 - Une stack web (EC2, RDS, Load Balancer, etc.)
-

□ 4. Composants d'un template CloudFormation YAML

```
AWSTemplateFormatVersion: '2010-09-09'    # Version du format (obligatoire)
Description: Déploie une instance EC2 simple
Metadata: {}                                # Métadonnées (facultatif)

Parameters:                                  # Valeurs dynamiques entrées par
l'utilisateur
  InstanceTypeParameter:
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t3.micro
    Description: Type d'instance EC2

Conditions: {}                               # Conditions logiques (facultatif)
Mappings: {}                                 # Mappage de valeurs (facultatif)
Rules: {}                                    # Règles pour valider des
paramètres (facultatif)

Resources:                                    # 📌 Obligatoire : toutes les
ressources ici
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: !Ref InstanceTypeParameter
      ImageId: ami-0abcdef1234567890

Outputs:                                     # Sorties à afficher après
création
  InstanceId:
    Description: ID de l'instance créée
    Value: !Ref MyEC2Instance
```

📦 5. Notion de Stack dans CloudFormation

Une **stack** est un regroupement logique de ressources AWS créées à partir d'un template CloudFormation.

🔗 **Exemple** : Une stack "WebApp" pourrait contenir :

- Un VPC
- Une instance EC2
- Un RDS
- Un ELB
- Des Security Groups

Chaque modification du template → mise à jour de la stack via `UpdateStack`.

🔧 6. Plugins et outils utiles

- 🔑 **AWS Toolkit** pour VS Code (recommandé)
 - 📄 **YAML Plugin** pour auto-complétion / linting
 - ⚙️ **Cfn-Lint** pour valider vos fichiers CFT localement
 - 🏠 **CloudFormation Designer** (dans AWS Console) pour visualiser graphiquement les stacks
-

🧠 7. Astuces et bonnes pratiques (Tips & Tricks)

- ☒ Utilisez **YAML** plutôt que JSON : plus lisible, comme Python
 - ☒ Découpez vos templates en **modules réutilisables**
 - ☒ Définissez des **paramètres** et **outputs** clairs
 - ☒ Ajoutez des **descriptions** à chaque ressource
 - ☒ Activez la **détection de dérive** pour savoir si les ressources ont été modifiées manuellement
 - ☒ Mettez vos templates sous **contrôle de version (Git)**
-

⚔️ 8. Comparaison CloudFormation vs Terraform

Critères	CloudFormation	Terraform
Fournisseur	Uniquement AWS	Multi-cloud (AWS, Azure, GCP, etc.)
Langage	YAML / JSON (déclaratif)	HCL (HashiCorp Language)
Modularité avancée	Moins intuitive	Oui, via Modules
Communauté & outils	Moins riche	Très riche (Terragrunt, Provider libs)
Gestion d'état	Géré par AWS	Fichier <code>terraform.tfstate</code>
Requiert CLI spécifique	Non, intégré dans AWS	Oui, installation de Terraform

9. Démo Théorique : Déploiement d'une instance EC2 via CFT

Template YAML simple

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Création d'une instance EC2 simple

Resources:
  MyInstance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t2.micro
      ImageId: ami-0c55b159cbfafelf0 # Remplacer par une AMI valide dans
votre région
```

Commande pour créer la stack :

```
aws cloudformation create-stack \
  --stack-name MaStackEC2 \
  --template-body file://mon_template.yaml \
  --capabilities CAPABILITY_IAM
```

10. Résumé final

Élément	Détails
Service	AWS CloudFormation
Utilisation	Créer des stacks AWS en IaC
Type de fichier	YAML / JSON
Caractère	Déclaratif
Composants clés	Version, Description, Parameters, Resources, Outputs
Recommandé pour	Déploiement reproductible, structuré, versionné
Différent de Terraform ?	Oui – Terraform est multi-cloud
Plugins	AWS Toolkit, YAML plugin, Cfn-Lint

Qu'est-ce que le *Drift* dans CloudFormation ?

Le **drift** (ou dérive en français) désigne une **différence entre l'état réel d'une ressource dans AWS et l'état attendu décrit dans le template CloudFormation.**

Autrement dit :

Quand quelqu'un modifie **manuellement** une ressource (via AWS Console ou CLI) sans passer par le template CFT, il y a un *drift*.

Drift Detection – Détection de dérive

Pourquoi c'est important ?

- Pour garantir que l'infrastructure reste **sous contrôle** et **en conformité** avec ce qui est défini dans le code.
- Pour **auditer** les changements manuels ou imprévus.
- Pour faciliter le **debug** lors d'un comportement inattendu.


Exemple :

Imaginons que dans ton template tu définis un **groupe de sécurité** autorisant uniquement le port 80.

Mais quelqu'un, en urgence, ajoute le port 22 via la console AWS → *Drift détecté*.

Comment utiliser la détection de drift ?

1. Via la console AWS

- Va dans **CloudFormation** > ta **stack**
- Clique sur **Actions** > **Detect drift**
- Tu verras un statut :
 - ☒ **IN_SYNC** (pas de dérive)
 -  **DRIFTED** (au moins une ressource modifiée)

2. Via AWS CLI

```
aws cloudformation detect-stack-drift --stack-name MaStackEC2
```

Et pour obtenir le résultat :

```
aws cloudformation describe-stack-drift-detection-status \
  --stack-drift-detection-id <id_retourné>
```

Statuts de drift possibles

Statut CloudFormation	Signification
IN_SYNC	La ressource est conforme au template
DRIFTED	La ressource a été modifiée manuellement
NOT_CHECKED	Drift non vérifié (souvent pour certaines ressources)
DELETED	La ressource n'existe plus dans AWS

Bonnes pratiques pour éviter le drift

- Interdire les modifications manuelles sur les ressources critiques (IAM, VPC, etc.)
- Utiliser des **politiques IAM strictes**
- Intégrer CFT dans un **pipeline CI/CD**
- Effectuer une **vérification de drift régulièrement**