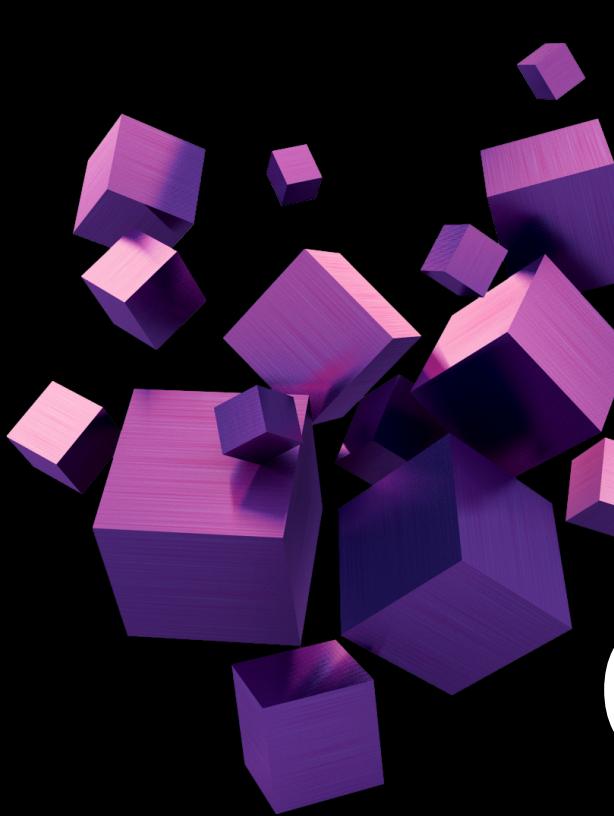


Terraform

L'IaC (Infrastructure as Code) a révolutionné la gestion des infrastructures informatiques en automatisant le processus de déploiement et de gestion de l'infrastructure. Cette technique consiste à écrire l'infrastructure dans des fichiers de configuration ou dans le code, offrant une source unique de vérité pour tous les composants nécessaires à l'infrastructure. Différents outils IaC sont disponibles, chacun ayant sa raison d'être, mais il est important de choisir l'outil qui répondra le mieux à votre objectif.



Qu'est-ce que Terraform ?

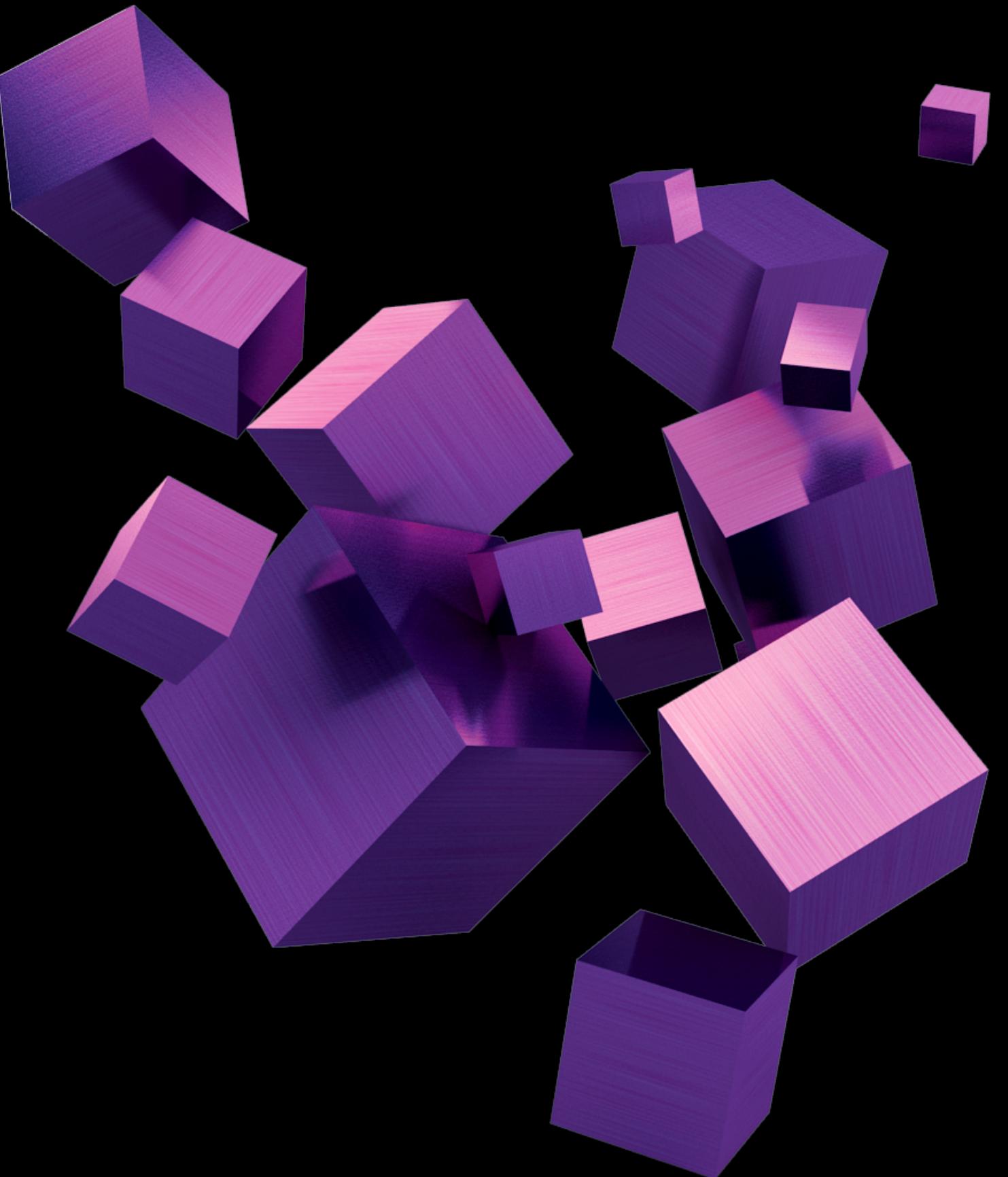
Terraform est un outil utilisé pour changer, construire et versionner l'infrastructure de manière efficace et sûre. Il gère les fournisseurs de services existants et populaires ou les solutions internes personnalisées. Les fichiers de configuration décrivent les composants qui fonctionnent pour une application unique ou pour l'ensemble du centre de données.

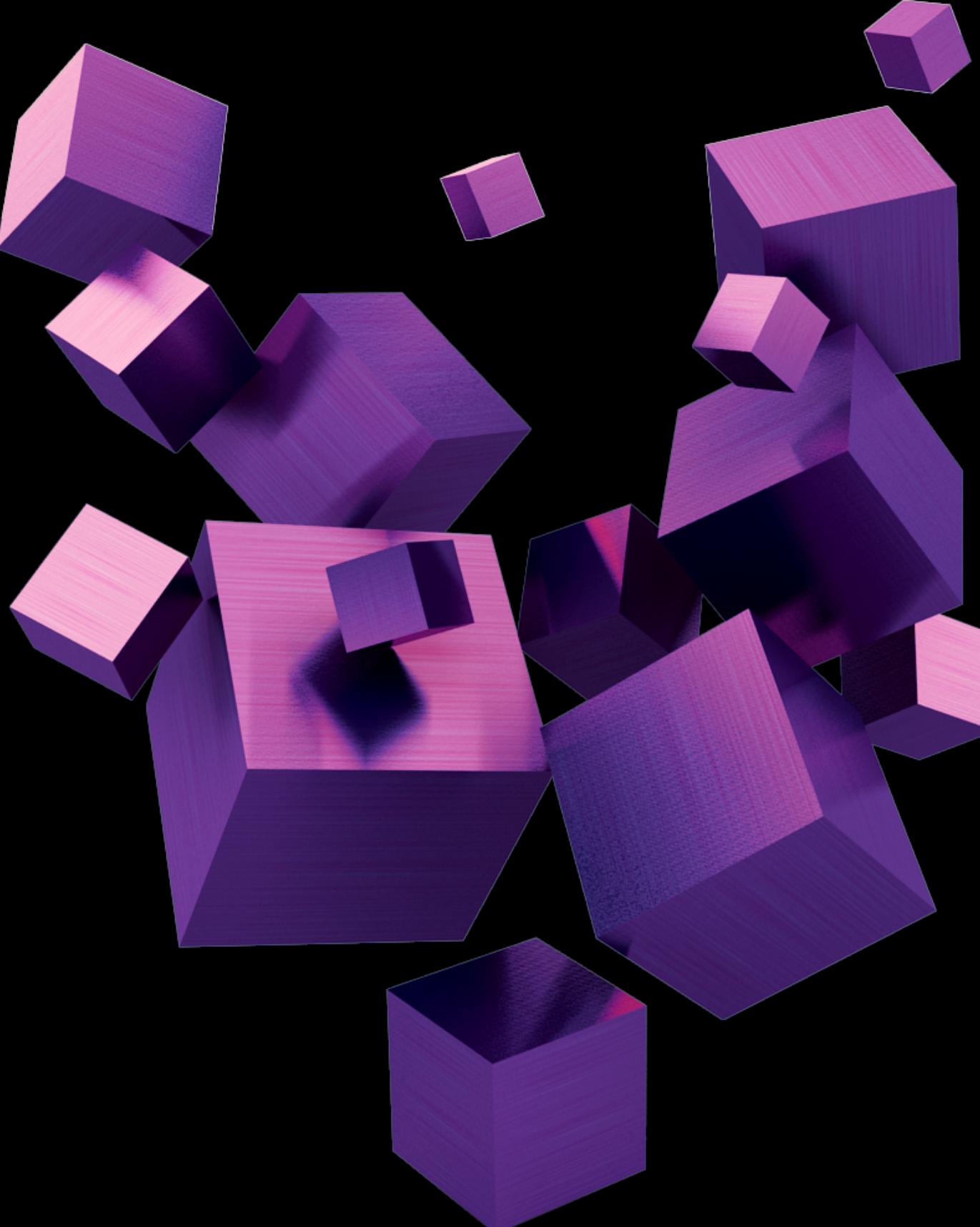
Terraform génère un plan d'exécution qui définit ce qu'il fera pour atteindre l'état souhaité et l'exécute ensuite pour construire l'infrastructure décrite. Il peut déterminer ce qui a changé et créer des plans d'exécution incrémentiels qui sont appliqués lorsque la configuration change.

L'infrastructure gérée par Terraform peut inclure des composants de bas niveau tels que les instances de calcul et le réseau, ainsi que des composants de haut niveau tels que les entrées DNS et les fonctionnalités SaaS.

Infrastructure as code.

- Il s'agit d'un outil permettant de gérer les cycles de vie des serveurs virtuels (AWS, VMWare, etc.).
- Il s'agit d'un outil permettant de gérer les services d'appui (DNS, courrier électronique).
- C'est un outil pour gérer les services système (MySQL, PostgreSQL).
- Les fichiers de configuration peuvent être HCL ou JSON.
- Crée par Hashicorp (Vagrant et al.)
- Écrit en Go
- Un outil qui vous permet de gérer, construire et versionner votre infrastructure en utilisant des fichiers de configuration.





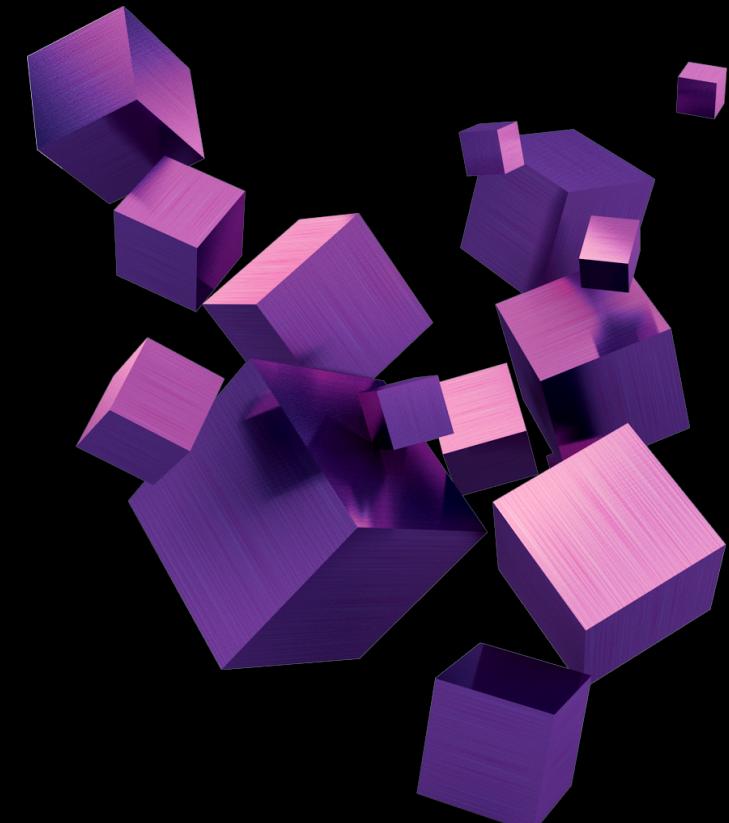
Pourquoi l'infrastructure en tant que code est-elle importante ?

DevOps vise à améliorer l'efficacité de la livraison des logiciels en automatisant l'infrastructure et en utilisant des outils tels que Ansible, Chef ou Puppet. Ces outils facilitent la définition de l'infrastructure en tant que code et permettent de gérer l'infrastructure complète du cloud, y compris les instances, les IP, les volumes et les réseaux. L'objectif est de rendre l'exécution DevOps plus rapide et plus efficace.

Pourquoi devriez-vous l'utiliser pour gérer votre infrastructure ?

Les utilisations mentionnées ci-dessous sont les suivantes :

- Un orchestrateur et non un outil d'automatisation
- Il suit une approche déclarative et non procédurale
- Prise en charge de plusieurs fournisseurs



Comment gérer votre Infrastructure as Code en l'utilisant ?

L'Infrastructure as Code (IaC) est un outil pratique qui permet de déployer une variété de ressources sur différentes plateformes en utilisant des fichiers de configuration. Il permet de construire, gérer et versionner l'infrastructure tout en garantissant la cohérence entre les différents environnements. Les fichiers de configuration sont utilisés pour communiquer avec les API des fournisseurs de cloud, comme GCP, afin de créer des instances de calcul. Les modules sont utilisés pour combiner plusieurs composants d'infrastructure en gros morceaux réutilisables et partageables.

Kubernetes suit également les principes IaC en écrivant tout le déploiement sous forme de code.

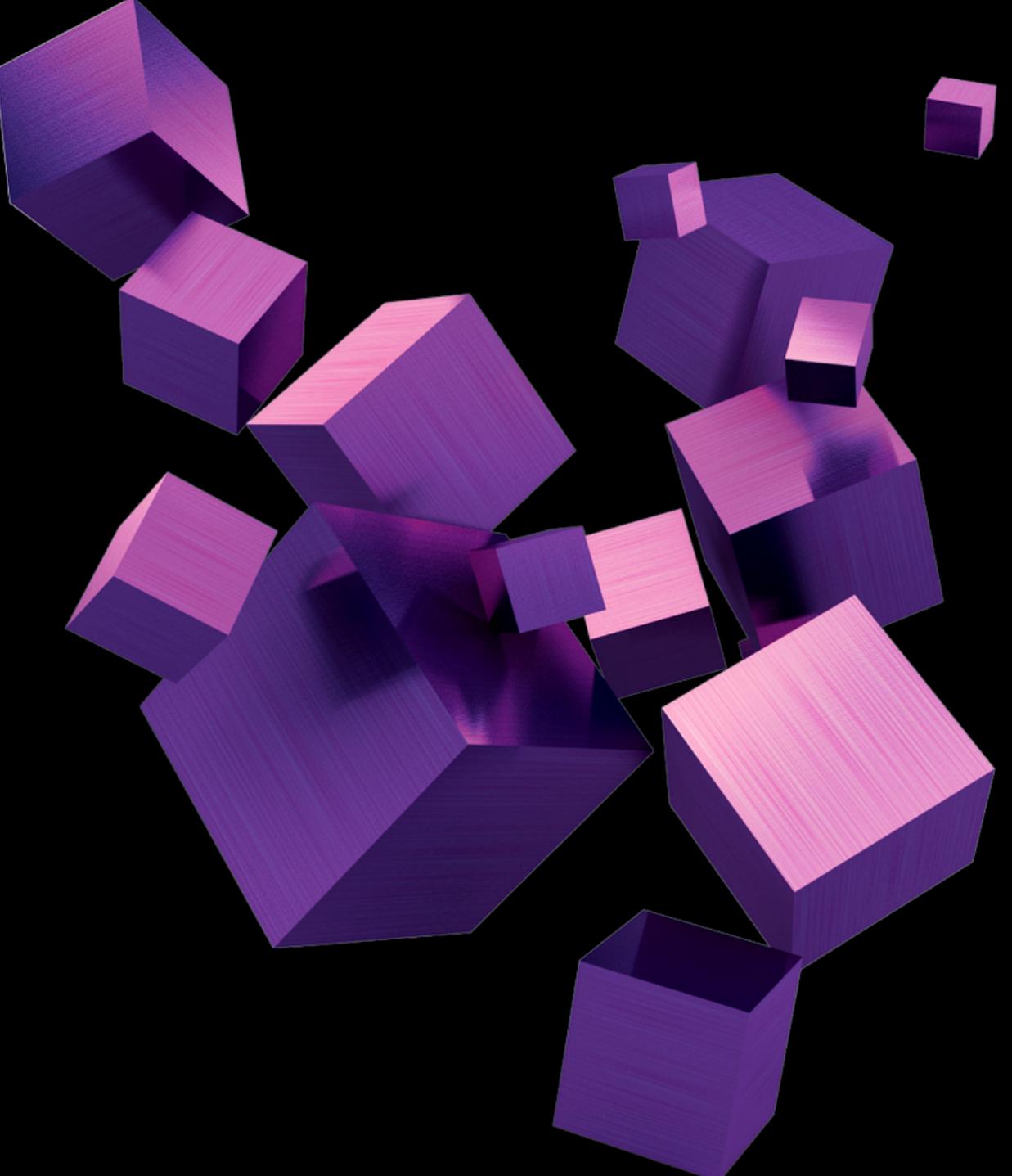
Quelles sont les caractéristiques de Infrastructure as Code ?

Les caractéristiques sont énumérées ci-dessous :

- Plans d'exécution
- Graphique des ressources
- Automatisation des changements
- Les mécanismes CI/CD

Comment fonctionne l'infrastructure en tant que code ?

Elle est divisée en deux parties, le noyau et les plugins. Le noyau communique avec son plugin. Les plugins exposent une implémentation pour des services spécifiques tels que AWS, bash.

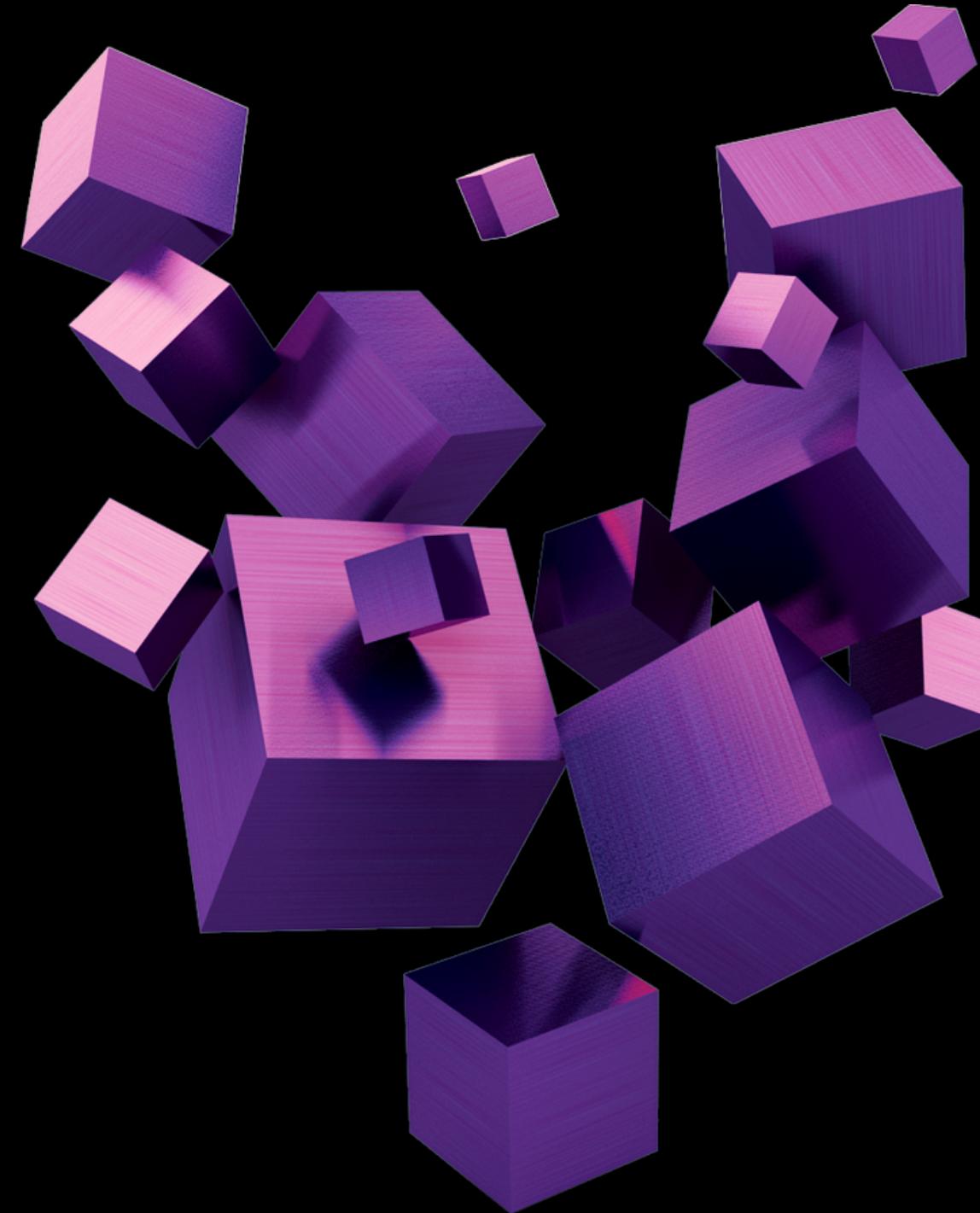


Qu'est-ce que son noyau ?

C'est un binaire écrit en langage de programmation Go. Le binaire compilé correspond au CLI terraform.

Core est responsable de :

- La lecture des fichiers de configuration, c'est-à-dire l'IaC.
- La gestion de l'état des différentes ressources.
- Construction du graphe des ressources.
- L'exécution du plan.
- Communication avec les plugins.



Que sont les plugins ?

- Les plugins sont des binaires exécutables écrits en Go.
- Configuration requise pour l'installation de Terraform
- binaire Terraform
- Fichiers de configuration (dans lesquels vous écrivez des IaC)
- Fichier d'état
- Fichiers de configuration Terraform (*.tf)

Pour configurer les ressources d'une infrastructure, vous utiliserez des fichiers de configuration avec une extension .tf. Les ressources sont les différents services et composants de l'infrastructure, comme les équilibreurs de charge et les machines virtuelles. Les fichiers de configuration contiennent des arguments pour configurer ces ressources.

1

Par exemple :



The image shows a terminal window with a dark theme. The title bar has three colored dots (red, yellow, green). The terminal displays the following Terraform configuration code:

```
//main.tf
ressource "aws_instance" "web" {
  ami = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"
  tags = {
    Name = "HelloWorld"
  }
}
```

In the bottom right corner of the terminal window, the URL snappyf.com is visible.

Le fichier de configuration main.tf ci-dessus définit une ressource, c'est-à-dire une instance EC2 dans AWS. Dans le bloc de la ressource, nous avons passé les arguments souhaités comme suit :

Qu'est-ce que le state Terraform ?

La partie la plus cruciale de Terraform est son fichier d'état (`terraform.tfstate`), qui stocke toutes les informations relatives à l'infrastructure déployée et à la correspondance entre les différents composants. Il se réfère ensuite à ce fichier d'état lors du déploiement de l'infrastructure pour vérifier les dépendances entre les ressources.

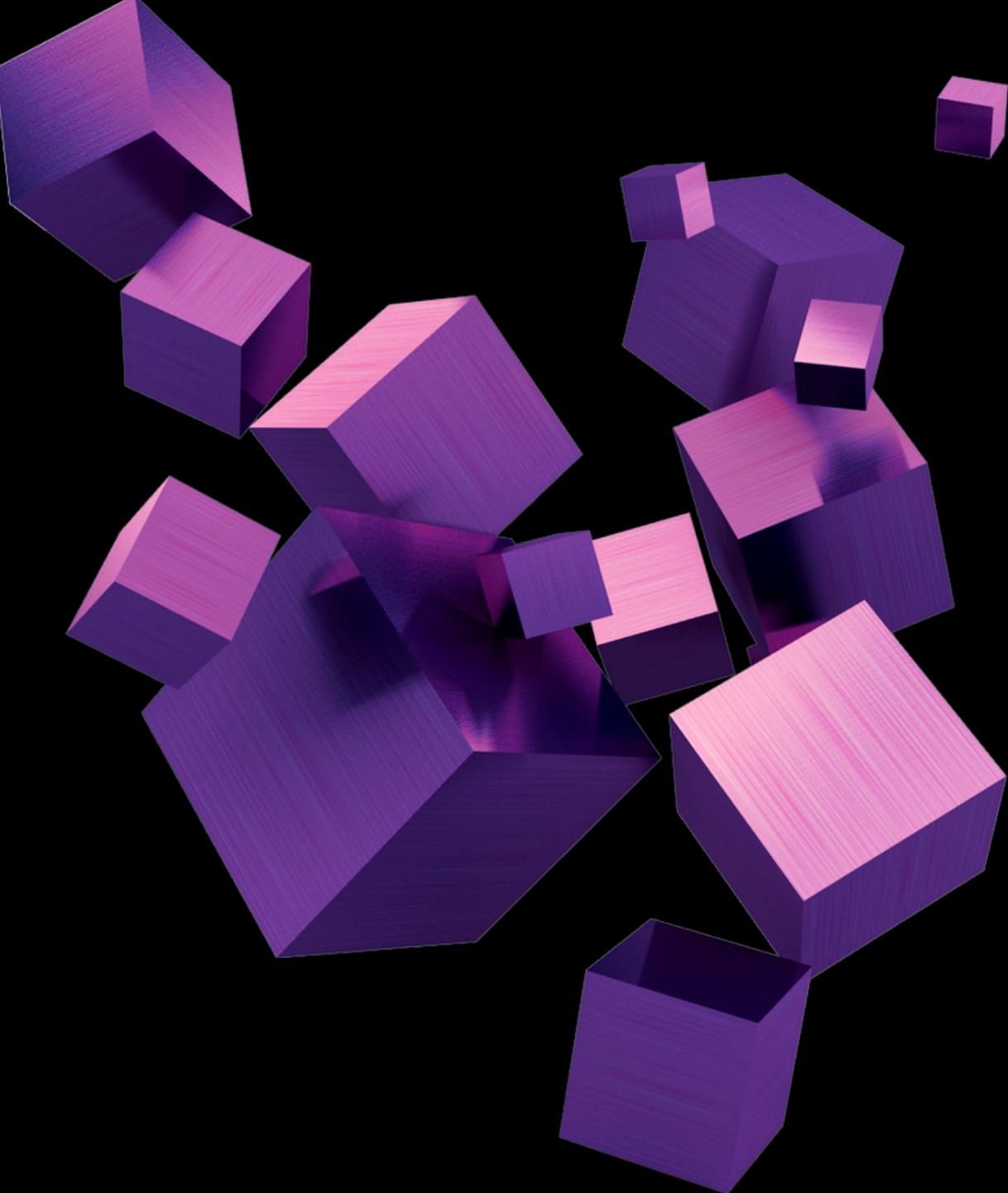
Pourquoi un fichier d'état est-il nécessaire ?

Lors du déploiement de ressources comme EC2, l'ARN ou l'ID de l'instance n'est pas spécifié, mais est attribué dynamiquement, ce qui nécessite de stocker ces informations pour détecter les changements. Le fichier d'état est la seule source de vérité pour votre infrastructure et doit être conservé en sécurité et à jour pour éviter les conflits. Il est recommandé de stocker le fichier state dans un backend distant tel qu'un bucket S3 ou Cloud storage.

L'exemple suivant montre que le backend est configuré comme un bucket S3.



```
terraform {  
backend "s3" {  
bucket = "mybucket"  
key   = "path/to/my/key"  
region = "us-east-1"  
}  
}
```



Initialisation du backend

Une étape d'initialisation doit être réalisée à chaque changement de configuration du backend ; la commande utilisée est `it into`.

Cette commande effectue les opérations suivantes :

Initialise le backend

Configure ses modules
(nous y reviendrons plus tard).

Workflow (flux de travail)

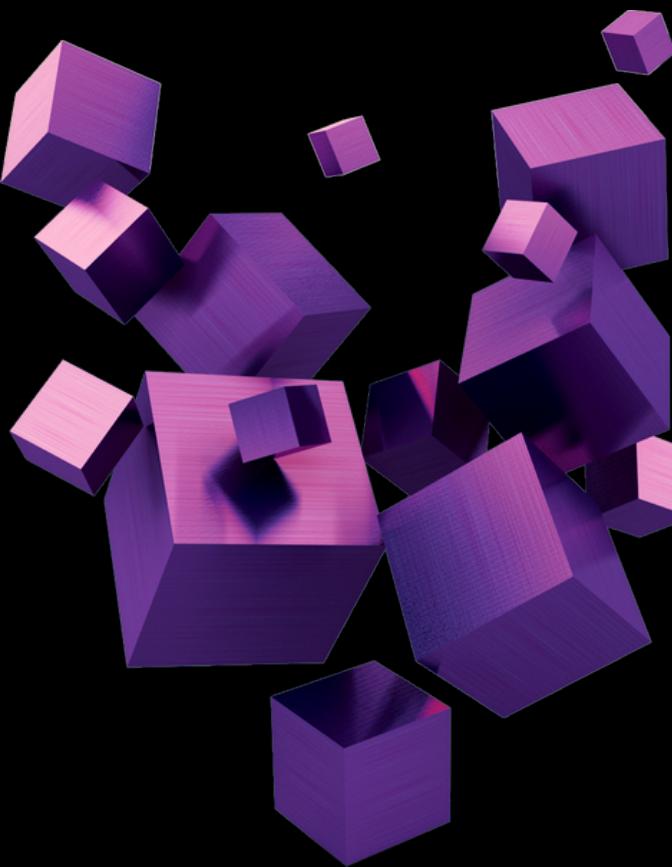
Écrire l'IAC, c'est-à-dire les fichiers de configuration.

Voir quels changements il va refléter en utilisant son plan.

Déployer ces changements à l'aide de l'application

Plan

Le plan représente une liste de changements que votre IAC va effectuer. Il n'affecte rien en réalité et indique simplement ce qui sera créé/supprimé/modifié. Par conséquent, il est possible d'utiliser it plan plusieurs fois.



Interprétation du plan

Les ressources peuvent être facilement interprétées par leur nom. En voici un exemple,

Syntaxe : <Type de ressource>.<nom de ressource>
Exemple, aws_instance.this.

Les ressources à l'intérieur d'un module peuvent être interprétées par leur nom :

Syntaxe : module.<nom_du_module>.
<type_de_ressource>. <nom de la ressource>
Exemple module.my_virtual_machines.aws_instance.this

Son plan indique cinq actions pour les ressources :

[+] Add - Crédation d'une nouvelle ressource.

[Delete - Suppression d'une ressource existante.

[Modifier sur place - Les ressources existantes seront modifiées et de nouveaux paramètres leur seront ajoutés.

Les plans vous indiqueront également quels nouveaux paramètres seront modifiés/ajoutés.

[- / +] Terraform supprimera et recréera la même ressource.
Cela se produit lorsqu'il n'est pas possible de modifier un paramètre en place.



```
20:10:18 Resource actions are indicated with the following symbols:  
20:10:18 + create  
20:10:18  
20:10:18 Terraform will perform the following actions:  
20:10:18  
20:10:18 # aws_cloudwatch_log_group.log_group_containers[0] will be  
20:10:18 + resource "aws_cloudwatch_log_group" "log_group_containers"  
20:10:18 + arn = (known after apply)  
20:10:18 + id = (known after apply)  
20:10:18 + name = "/eks/dev-scipher-fx-eks-cluster-2/containers"  
20:10:18 + retention_in_days = 14  
20:10:18 + tags = {  
20:10:18 + "Environment" = "dev"  
20:10:18 + "Service" = "EKS Cluster"  
20:10:18 + "Terraform" = "true"  
20:10:18 }  
20:10:18 }  
20:10:18  
20:10:18 # aws_cloudwatch_log_group.log_group_host[0] will be created  
20:10:18 + resource "aws_cloudwatch_log_group" "log_group_host" {  
20:10:18 + arn = (known after apply)  
20:10:18 + id = (known after apply)  
20:10:18 + name = "/eks/dev-scipher-fx-eks-cluster-2/host"  
20:10:18 + retention_in_days = 14  
20:10:18 + tags = {  
20:10:18 + "Environment" = "dev"  
20:10:18 + "Service" = "EKS Cluster"  
20:10:18 + "Terraform" = "true"  
20:10:18 }  
20:10:18 }
```

Exemple de plan

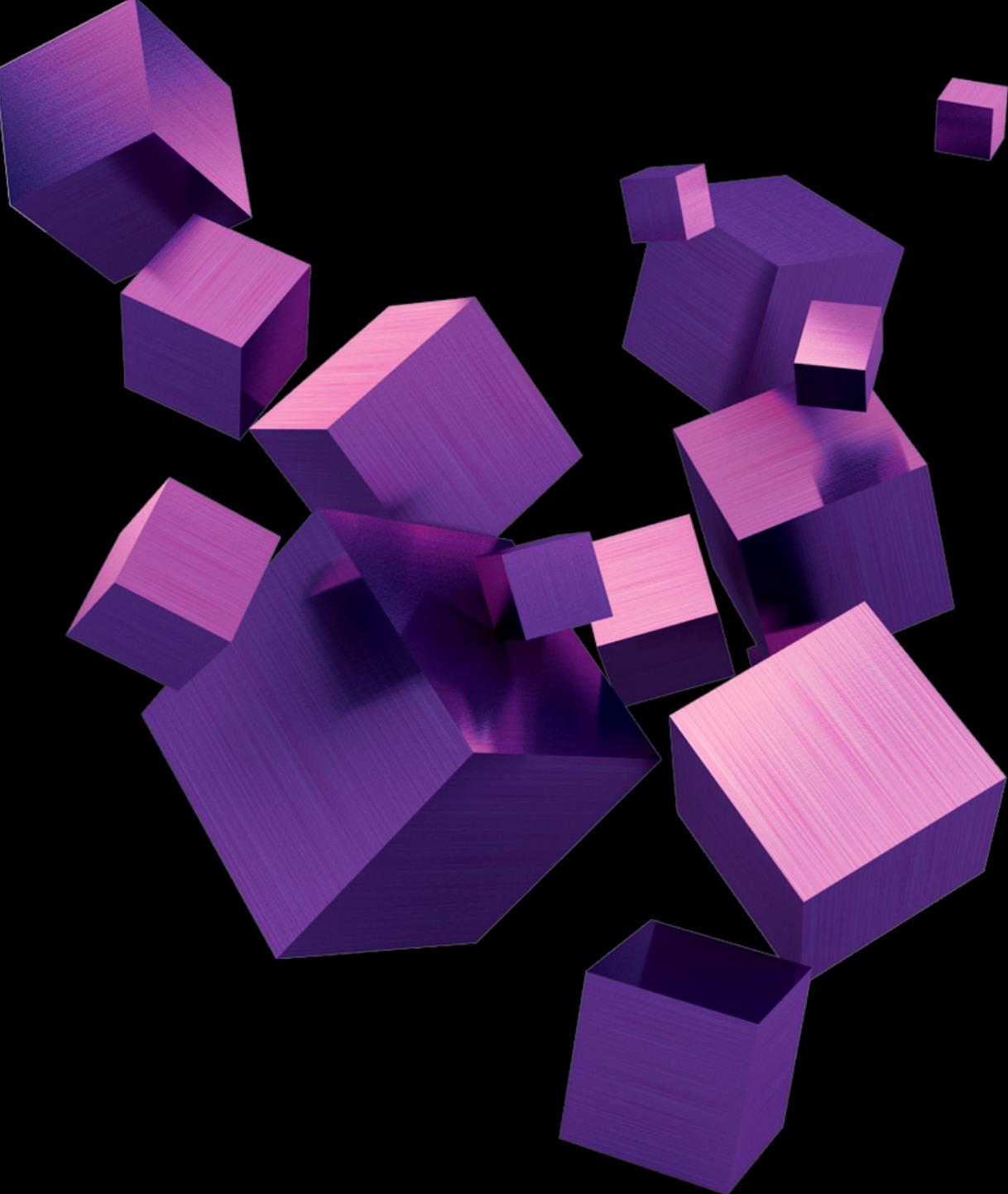
Un plan d'exécution a été généré et est présenté ci-dessous.

Dans l'exemple ci-dessus, le plan indique

Deux nouveaux groupes Cloudwatch sont créés dans AWS.

La durée de rétention des groupes de logs est fixée à 14 jours.

Ajout de balises aux groupes de journaux



Terraform apply

Lors de l'application d'un plan donné, l'état est verrouillé. Il effectuera toutes les actions indiquées dans la phase de planification et déployera ces ressources. Certaines ressources peuvent prendre du temps ; par exemple, AWS DocumentDB peut prendre jusqu'à 15 minutes. Dans l'intervalle, toutes les autres opérations sont bloquées.

Une fois l'opération terminée, il met à jour le fichier d'état.

Modules



```
module "moduleName" {  
source = "module/path"  
\CONFIG  
/  
}
```

Les modules sont un ensemble de fichiers de configuration et un dossier qui permettent de réutiliser le code avec les paramètres requis. Ils sont similaires aux fonctions d'un langage de programmation, où l'on peut appeler le processus n'importe où et passer les paramètres nécessaires. L'utilisation d'un module dans un fichier de configuration est similaire à l'utilisation de ressources, à la différence de la clause "module".

Le chemin d'accès à la source indique où le module peut être trouvé.

Le bloc CONFIG consiste en un ensemble de variables ou d'arguments qui peuvent être transmis au module et qui appartiennent au module.

Par exemple, si le module principal a une variable log_retention_period

Si le module principal a une variable log_retention_period, nous pouvons passer sa valeur différemment selon les environnements :

Pour l'environnement Dev :



```
module "cloud_watch_log_groups" {  
  source = "module/path"  
  Log_retention_period = 14}
```

snappyf.com

Pour l'environnement Prod :



```
module "cloud_watch_log_groups" {  
  source = "module/path"  
  Log_retention_period = 21  
}
```

snappyf.com

Quels sont les avantages de l'infrastructure en tant que code ?

L'infrastructure en tant que code est la gestion de l'infrastructure dans un modèle descriptif, qui aide à créer l'infrastructure et à la gérer avec le code source.

- L'orchestration, et pas seulement la configuration
- Multi-fournisseur
- Infrastructure immuable
- Syntaxe - HCL (HashiCorp Configuration Language) est un langage personnalisé utilisé par le logiciel.
- Exécutions à blanc
- Architecture client uniquement
- Super portabilité

Quels sont les cas d'utilisation ?

Les cas d'utilisation sont énumérés ci-dessous :

Installation d'une application Heroku

Applications à plusieurs niveaux

Self-service Clusters

L'IA a permis à l'infrastructure informatique d'être flexible, intangible et à la demande.

Interface de ligne de commande pour Terraform ?

- L'interface de ligne de commande (CLI) est utilisée pour contrôler Terraform.
- **apply** - Permet de modifier ou de construire l'infrastructure
- **console** - Pour la console interactive
- **destroy** - Permet d'altérer l'infrastructure gérée
- **fmt** - Sert à réécrire les fichiers de configuration dans un format canonique
- **get** - Sert à télécharger et à installer des modules de configuration
- **graph** - Permet de créer un graphique visuel de ses ressources
- **import** - Sert à importer l'infrastructure existante dans it
- **init** - Sert à initialiser la configuration existante ou nouvelle de it
- **output** - Permet de lire la sortie d'un fichier d'état
- **plan** - Permet de générer et d'afficher un plan d'exécution
- **providers** - Permet d'imprimer l'arbre des fournisseurs utilisés dans la configuration
- **push** - Sert à télécharger ce module it vers it Enterprise pour l'exécuter
- **refresh** - Permet de mettre à jour le fichier d'état local en fonction des ressources réelles
- **show** - Permet d'inspecter l'état ou le plan de it
- **taint** - Permet de marquer manuellement les ressources à recréer
- **untaint** - Permet d'annuler manuellement le marquage d'une ressource comme altérée
- **validate** - Permet de valider ses fichiers
- **version** - Permet d'imprimer sa version
- **workspace** - Permet de gérer l'espace de travail

Quelles sont les meilleures pratiques ?

Les meilleures pratiques de l'Infrastructure as Code sont les suivantes :

- Structure du code
- Backend distant
- Conventions de nommage