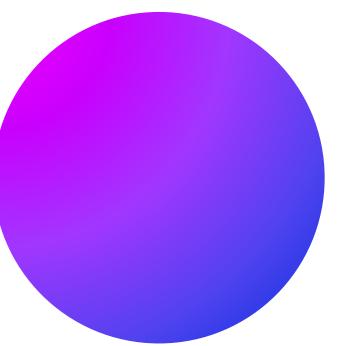


Terraform



Plan



Définition lac



Définition Terraform



Avantages Terrafrm



Installation Terraform



Installation Terraform

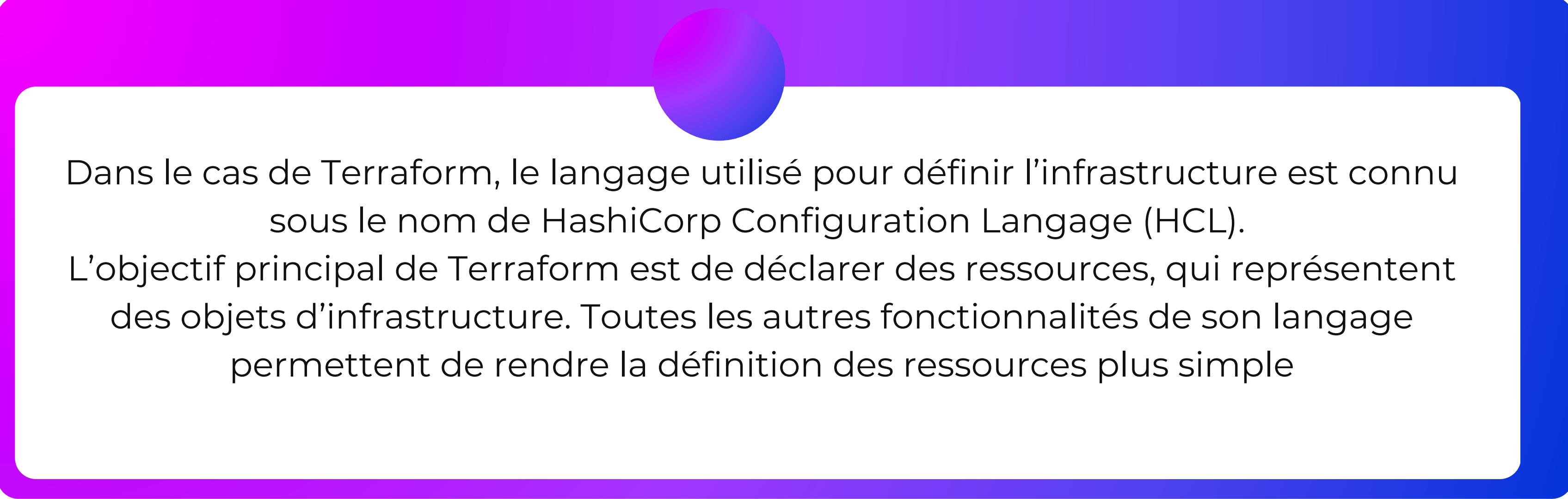
Introduction

L'IaC (Infrastructure as Code) est une approche permettant de gérer le cycle de vie de l'ensemble des ressources et services composant une infrastructure. Elle permet d'obtenir des déploiements automatisés, cohérents et reproductibles (idempotence).

01

Définition lac

HCL Langage de Terraform



Dans le cas de Terraform, le langage utilisé pour définir l'infrastructure est connu sous le nom de HashiCorp Configuration Langage (HCL).

L'objectif principal de Terraform est de déclarer des ressources, qui représentent des objets d'infrastructure. Toutes les autres fonctionnalités de son langage permettent de rendre la définition des ressources plus simple

IAC



Infrastructure as Code (IaC) consiste à remplacer les processus manuels et les procédures d'exploitation standard pour configurer les périphériques matériels et les systèmes d'exploitation par du code qui gérera et fournira automatiquement votre infrastructure. Dans l'IaC, vous pouvez configurer et déployer ces composants d'infrastructure plus rapidement avec cohérence en les traitant comme si c'était une application.

IAC



Ainsi, chaque fois que vous devez configurer une infrastructure, vous n'avez pas besoin d'aller voir les administrateurs système, de formuler une demande, de créer un ticket et d'attendre qu'ils y participent. Au lieu de cela, vos développeurs et vos équipes d'exploitation peuvent facilement le faire en utilisant votre code.

Avantages IaC

Augmentez la vitesse

L'automatisation est plus rapide que la navigation manuelle dans une interface lorsque vous avez besoin de déployer et/ou de connecter des ressources.

Améliorez la fiabilité

Si on a une infrastructure de grande taille, il est très facile de mal configurer une ressource ou de mettre à disposition des services dans le mauvais ordre. Avec l'IaC, les ressources sont toujours mises à disposition et configurées exactement comme déclarées.

Avantages IaC

Encourage l'expérimentation, le test et l'optimisation

l'IaC accélère et simplifie énormément la mise à disposition de nouvelles infrastructures, on peut tester des changements expérimentaux sans investir beaucoup de temps ou de ressources. Si les résultats vous plaisent, vous pouvez rapidement mettre à l'échelle la nouvelle infrastructure pour la production

02

Définition Terraform

Définition

Terraform est un outil appartenant à cette famille qui s'est imposé comme une référence dans cet écosystème. Il permet de réaliser des opérations de déploiement sur de nombreux providers (Clouds publics et privés, Kubernetes...).

Définition

Terraform propose une approche déclarative, via des templates, permettant de décrire une infrastructure cible, sans avoir besoin de spécifier les opérations unitaires à réaliser.

Terraform, fais partie des meilleurs outils devops et en particulier de l'infrastructure-as-code.

03

Avantages Terraform

Avantages Terraform

- **Open source** : Terraform a l'appui de grandes communautés de contributeurs qui créent des plug-ins pour la plateforme. Quel que soit le fournisseur cloud que vous utilisez, il est facile de trouver des plug-ins, des extensions et un support professionnel.
- **Indépendant de la plateforme** : On peut l'utiliser avec tout fournisseur de services cloud. La plupart des autres outils IaC sont conçus pour fonctionner avec un seul fournisseur cloud.

Avantages Terraform

- **Gérer rapidement l'infrastructure :** Le principal bénéfice de Terraform est sans doute le gain de temps. En effet, grâce aux fichiers de configuration, on a le contrôle pour approvisionner, définir ou paramétriser des ressources de manière répétable et prévisible.
- **Bonne capacité de suivi des performances :** Grâce à la fonctionnalité de Monitoring as Code, Terraform simplifie vos actions de monitoring. Cette technologie stocke l'état de l'infrastructure afin de suivre ses modifications et partager ses configurations.

04

Installation Terraform

Installation Terraform(ubuntu)

Mettre à jour le système et installer quelques packages pour vérifier le GPG KEY

```
sudo apt-get update && sudo apt-get install -y gnupg  
software-properties-common
```

Installer le GPG KEY

```
wget -O- https://apt.releases.hashicorp.com/gpg | \  
gpg --dearmor | \ sudo tee  
/usr/share/keyrings/hashicorp-archive-keyring.gpg
```

Installation Terraform(ubuntu)

Vérifier le GPG KEY

```
gpg --no-default-keyring \ --keyring  
/usr/share/keyrings/hashicorp-archive-keyring.gpg \ --  
fingerprint
```

Ajouter le HashiCorp repository au système

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-  
archive-keyring.gpg] \n  
https://apt.releases.hashicorp.com $(lsb_release -cs)  
main" | \ sudo tee /etc/apt/sources.list.d/hashicorp.list
```

Installation Terraform(ubuntu)

Mettre à jour le système et installer terraform

sudo apt update

sudo apt-get install terraform

Installation Terraform(Centos)

Installer yum-config-manager pour gérer hashicorp repository

sudo yum install -y yum-utils

Utiliser yum-config-manager pour ajouter l'official HashiCorp Linux repository

**sudo yum-config-manager --add-repo
<https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo>**

Installation Terraform(Centos)

Installer Terraform

sudo yum -y install terraform

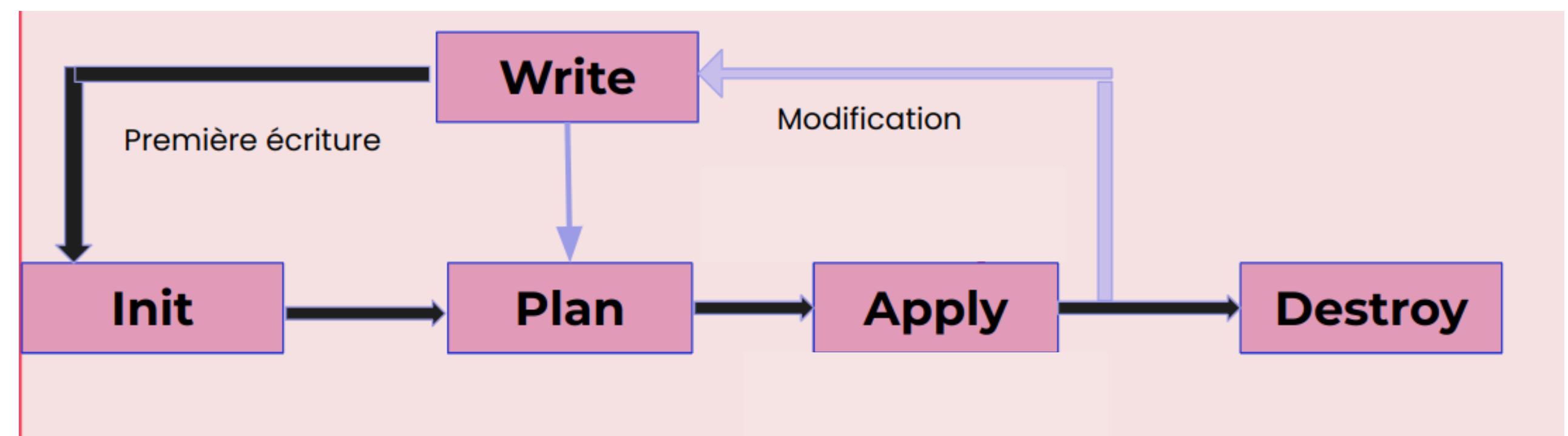
Pour vérifier l'installation de Terraform

terraform -help

05

Utilisation Terraform

Workflow de Terraform



Workflow de Terraform

- **Write:** Vous commencez à écrire votre configuration.
- **Init:** Initialiser la configuration pour installer les dépendances nécessaires.
- **Plan:** Auditer les changements et valider simplement si on les accepte.

Workflow de Terraform

- **Apply:** Appliquer les changements à l'infrastructure réelle.
- **Destroy:** Décommissionnez toute l'infrastructure.

Status de Terraform

Terraform doit stocker l'état de la configuration, car il va servir pour mapper les ressources réelles de la configuration ainsi que ces métadonnées.

Par défaut cet état est stocké dans un fichier local nommé “`terraform.tfstate`”, mais il peut également être stocké à distance, mais pas dans votre repository git, parce qu'il peut contenir des secrets.

Status de Terraform

Terraform utilise cet état pour créer des plans et apporter des modifications à votre infrastructure. Avant toute opération, Terraform effectue un rafraîchissement pour mettre à jour l'état avec celle de l'infrastructure réelle

Status de Terraform

Les fichiers Terraform porte l'extension .tf dont le principal se nomme main.tf

Le langage de configuration HashiCorp (HCL) permet de décrire rapidement des ressources à l'aide de blocs, d'arguments et d'expressions.



Les fichiers de déclarations de configurations



Les blocs sont des conteneurs pour d'autres contenus et représentent généralement la configuration d'un type d'objet, comme une ressource, un provider, ... Les blocs ont un type de bloc, peuvent avoir zéro ou plusieurs étiquettes et avoir un corps qui contient un nombre quelconque d'arguments et de blocs imbriqués.



Les fichiers de déclarations de configurations



Les arguments attribuent une valeur à un nom. Ils apparaissent dans les blocs

Les expressions représentent une valeur, soit littéralement, soit en référençant et en combinant d'autres valeurs. Ils apparaissent en tant que valeurs d'arguments ou dans d'autres expressions.

Les Providers

Les principaux providers d'infrastructure :

- AWS
- Google Cloud Platform
- Azure
- VmWare Vsphere

Les variables d'entrée

Les Input variables ou variables d'entrée pour nous français, sont définies en indiquant un nom, un type et une valeur par défaut (default = “valeur”). Le type est optionnel, car Terraform les déduit automatiquement. On peut ajouter une description.

Les fichiers de variables

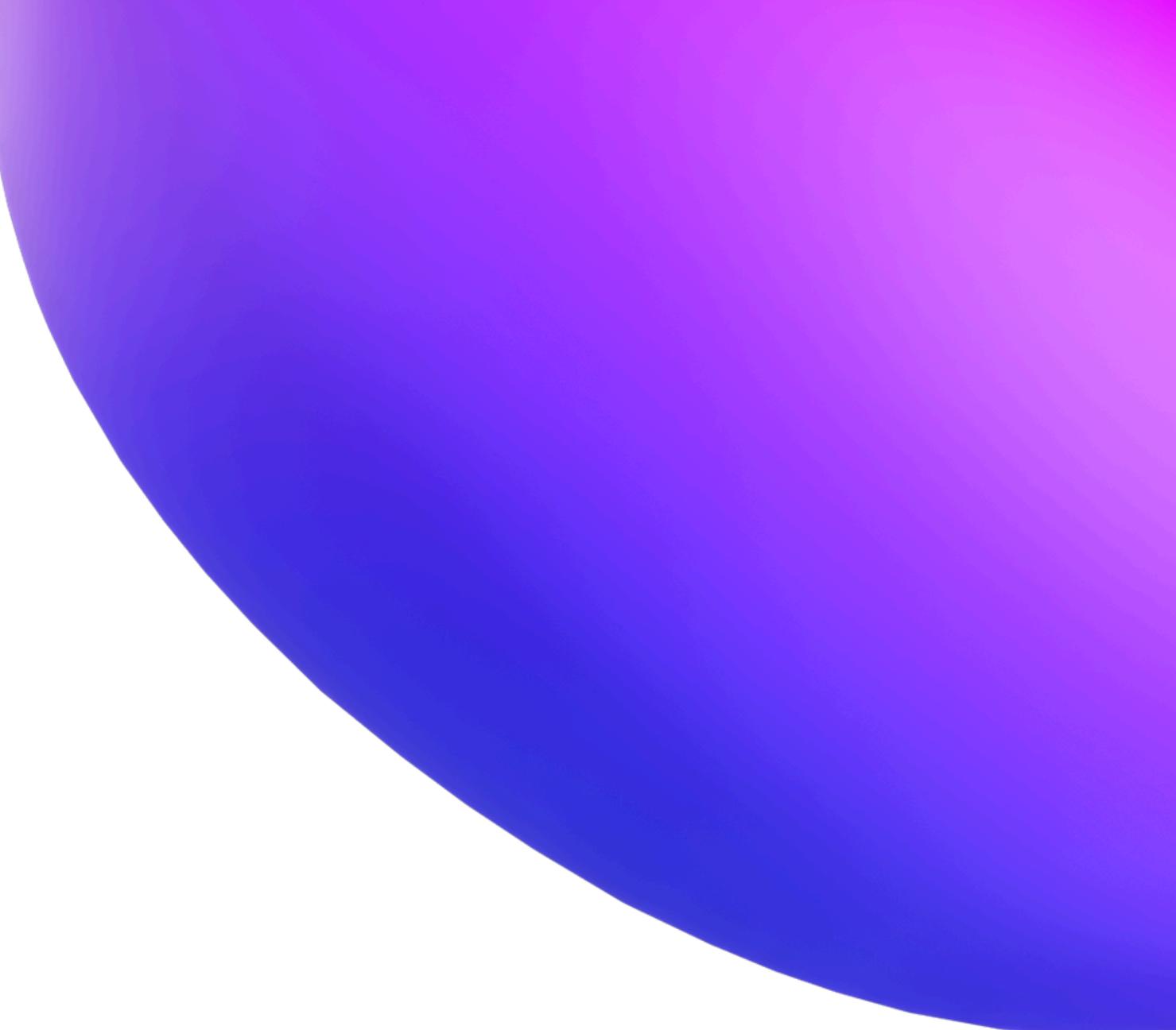
Pour de nombreuses variables, il est préférable de les déclarer dans un fichier de définitions de variables. Un fichier se terminant par l'extension .tfvars

Les variables d'entrée

Attention

Une variable ne peut pas se nommer : source, version, providers, count, for_each, lifecycle, depends_on ou encore locals.

Conclusion



Hashicorp Terraform est l'outil de gestion d'infrastructure idéal si vous utilisez plusieurs fournisseurs cloud. Sa grande compatibilité ainsi que son mode de fonctionnement en font une des meilleures technologies pour les grandes organisations. Si vous êtes intéressés par cet outil, nous vous proposons notre formation Terraform disponible en présentiel ou en distanciel.



Merci !!