Référence rapide Terraform

Fournisseurs, ressources, variables, état, modules

Bases

Flux de travail principal

```
terraform init      # install providers & modules
terraform plan      # preview changes
terraform apply     # apply changes
terraform destroy   # tear down all resources
```

Commandes essentielles

`terraform init`	Initialiser le répertoire de travail, télécharger les fournisseurs
`terraform plan`	Afficher le plan d'exécution sans appliquer
`terraform apply`	Appliquer les modifications à l'infrastructure
`terraform destroy`	Détruire toutes les ressources gérées
`terraform fmt`	Formater les fichiers **`.tf`** selon le style canonique
`terraform validate`	Vérifier la syntaxe de la configuration
`terraform show`	Afficher l'état actuel ou le plan
`terraform output`	Afficher les valeurs de sortie

Fournisseurs

Configuration du fournisseur

```
terraform {
  required_providers {
    aws = { source = "hashicorp/aws", version = "~> 5.0" }
  }
}
provider "aws" {
  region = "us-east-1"
}
```

Notes sur les fournisseurs

`source`	Adresse du registre (**`hashicorp/aws`**, **`hashicorp/google`**)
`version`	Contrainte de version (**`~> 5.0`**, **`>= 3.0, < 4.0`**)
`.terraform.lock.hcl`	Fichier de verrouillage — à valider dans le contrôle de version
`alias`	Utiliser plusieurs configurations pour le même fournisseur

Ressources

Blocs de ressources

```
resource "aws_instance" "web" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t3.micro"
  tags = { Name = "web-server" }
}
```

Meta-arguments de ressource

`depends_on`	Dépendance explicite envers une autre ressource
`count`	Créer plusieurs instances (**`count = 3`**)
`for_each`	Créer des instances depuis une map ou un ensemble
`provider`	Sélectionner un alias de fournisseur non par défaut
`lifecycle`	Personnaliser le comportement de création/mise à jour/destruction

Référencer les ressources

```
# type.name.attribute
aws_instance.web.id
aws_instance.web.public_ip
aws_vpc.main.cidr_block
```

Variables

Déclarer des variables

```
variable "region" {
  type    = string
  default = "us-east-1"
}
variable "instance_count" {
  type        = number
  description = "Number of instances"
}
```

Définir les valeurs de variables

`-var 'region=us-west-2'`	Option en ligne de commande
`-var-file=prod.tfvars`	Charger depuis un fichier **`.tfvars`**
`terraform.tfvars`	Chargé automatiquement si présent
`TF_VAR_region`	Variable d'environnement
`Interactive prompt`	Demandé au plan/apply si pas de valeur par défaut

Types de variables

`string`	**`"us-east-1"`**
`number`	**`42`**
`bool`	**`true`** / **`false`**
`list(string)`	**`["a", "b"]`**
`map(string)`	**`{ key = "val" }`**
`object({...})`	Type structuré avec attributs nommés

Sorties

Définir des sorties

```
output "instance_ip" {
  value       = aws_instance.web.public_ip
  description = "Public IP of the web server"
}
output "db_password" {
  value     = random_password.db.result
  sensitive = true
}
```

Commandes de sortie

`terraform output`	Afficher toutes les sorties
`terraform output instance_ip`	Afficher une sortie spécifique
`terraform output -json`	Format JSON pour les scripts
`sensitive = true`	Masquer la valeur dans la sortie CLI
`module.vpc.vpc_id`	Accéder aux sorties d'un module enfant

État

Backend distant

```
terraform {
  backend "s3" {
    bucket = "my-tf-state"
    key    = "prod/terraform.tfstate"
    region = "us-east-1"
  }
}
```

Commandes d'état

`terraform state list`	Lister toutes les ressources dans l'état
`terraform state show <addr>`	Afficher les attributs d'une ressource
`terraform state mv <src> <dst>`	Renommer / déplacer une ressource dans l'état
`terraform state rm <addr>`	Supprimer une ressource de l'état (garder l'infra)
`terraform state pull`	Télécharger l'état distant vers stdout
`terraform import <addr> <id>`	Importer une infra existante dans l'état

Modules

Utiliser des modules

```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "~> 5.0"
  cidr    = "10.0.0.0/16"
}
```

Sources de modules

`"./modules/vpc"`	Chemin local
`"terraform-aws-modules/vpc/aws"`	Registre Terraform
`"github.com/org/repo//module"`	Dépôt GitHub
`"s3::https://bucket/module.zip"`	Compartiment S3

Structure d'un module

```
modules/vpc/
  main.tf          # resources
  variables.tf     # input variables
  outputs.tf       # output values
```

Sources de données

Lire des ressources existantes

```
data "aws_ami" "ubuntu" {
  most_recent = true
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/*"]
  }
  owners = ["099720109477"]
}
```

Sources de données courantes

`data.aws_ami`	Rechercher une AMI par filtres
`data.aws_vpc`	Rechercher un VPC existant
`data.aws_caller_identity`	Identifiant de compte AWS actuel
`data.aws_region`	Région AWS actuelle
`data.terraform_remote_state`	Lire les sorties d'un autre fichier d'état
`data.external`	Exécuter un programme externe pour des données

Cycle de vie

Règles de cycle de vie

```
resource "aws_instance" "web" {
  lifecycle {
    create_before_destroy = true
    prevent_destroy       = true
    ignore_changes        = [tags]
  }
}
```

Référence rapide Terraform

Options de cycle de vie

`create_before_destroy`	Créer le remplacement avant de détruire l'ancien
`prevent_destroy`	Erreur si **`terraform destroy`** cible cette ressource
`ignore_changes`	Ne pas détecter la dérive sur les attributs listés
`replace_triggered_by`	Forcer le remplacement quand la ressource référencée change
`precondition`	Valider les hypothèses avant l'application
`postcondition`	Valider les résultats après l'application

Modèles courants

Boucles et conditions

```
# for_each with a map
resource "aws_iam_user" "users" {
  for_each = toset(["alice", "bob"])
  name     = each.value
}
# conditional resource
count = var.create_db ? 1 : 0
```

Fonctions utiles

`file("key.pub")`	Lire le contenu d'un fichier
`join(", ", list)`	Joindre une liste en chaîne
`lookup(map, key, default)`	Recherche dans une map avec valeur par défaut
`length(list)`	Nombre d'éléments
`toset(["a", "b"])`	Convertir une liste en ensemble (pour for_each)
`try(expr, fallback)`	Retourner la valeur de repli si expr échoue
`templatefile(path, vars)`	Rendre un fichier template