TERRAFORM クイックリファレンス

プロバイダー、リソース、変数、ステート、モジュール

基本

コアワークフロー

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

必須コマンド

`terraform init`	作業ディレクトリを初期化し、プロバイダーをダウンロード
`terraform plan`	適用せずに実行計画を表示
`terraform apply`	インフラに変更を適用
`terraform destroy`	管理対象のすべてのリソースを破棄
`terraform fmt`	`.tf` ファイルを標準スタイルにフォーマット
`terraform validate`	設定の構文チェック
`terraform show`	現在のステートまたはプランを表示
`terraform output`	アウトプット値を表示

プロバイダー

プロバイダーの設定

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

プロバイダーの注意事項

`source`	レジストリアドレス（`hashicorp/aws`、`hashicorp/google`）
`version`	バージョン制約（`~> 5.0`、`>= 3.0, < 4.0`）
`.terraform.lock.hcl`	ロックファイル — バージョン管理にコミット
`alias`	同じプロバイダーで複数の設定を使用

リソース

リソースブロック

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

リソースのメタ引数

`depends_on`	別のリソースへの明示的な依存関係
`count`	複数インスタンスを作成（`count = 3`）
`for_each`	マップまたはセットからインスタンスを作成
`provider`	デフォルト以外のプロバイダーエイリアスを選択
`lifecycle`	作成/更新/破棄の動作をカスタマイズ

リソースの参照

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

変数

変数の宣言

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

変数の値の設定

`-var 'region=us-west-2'`	CLI フラグ
`-var-file=prod.tfvars`	`.tfvars` ファイルから読み込み
`terraform.tfvars`	存在する場合は自動読み込み
`TF_VAR_region`	環境変数
`Interactive prompt`	デフォルトがない場合に plan/apply で問い合わせ

変数の型

`string`	`"us-east-1"`
`number`	`42`
`bool`	`true` / `false`
`list(string)`	`["a", "b"]`
`map(string)`	`{ key = "val" }`
`object({...})`	名前付き属性を持つ構造型

アウトプット

アウトプットの定義

```
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
}
```

アウトプットコマンド

`terraform output`	すべてのアウトプットを表示
`terraform output instance_ip`	特定のアウトプットを表示
`terraform output -json`	スクリプト用 JSON 形式
`sensitive = true`	CLI 出力から値を非表示
`module.vpc.vpc_id`	子モジュールのアウトプットにアクセス

ステート

リモートバックエンド

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

ステートコマンド

`terraform state list`	ステート内のすべてのリソースを一覧表示
`terraform state show <addr>`	リソースの属性を表示
`terraform state mv <src> <dst>`	ステート内のリソースを名前変更/移動
`terraform state rm <addr>`	ステートからリソースを削除（インフラは保持）
`terraform state pull`	リモートステートを stdout にダウンロード
`terraform import <addr> <id>`	既存インフラをステートにインポート

モジュール

モジュールの使用

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

モジュールのソース

`"./modules/vpc"`	ローカルパス
`"terraform-aws-modules/vpc/aws"`	Terraform レジストリ
`"github.com/org/repo//module"`	GitHub リポジトリ
`"s3::https://bucket/module.zip"`	S3 バケット

モジュール構造

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

データソース

既存リソースの読み取り

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

主要データソース

`data.aws_ami`	フィルターで AMI を検索
`data.aws_vpc`	既存 VPC を検索
`data.aws_caller_identity`	現在の AWS アカウント ID
`data.aws_region`	現在の AWS リージョン
`data.terraform_remote_state`	別のステートファイルのアウトプットを読み取り
`data.external`	外部プログラムを実行してデータを取得

ライフサイクル

ライフサイクルルール

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

ライフサイクルオプション

`create_before_destroy`	古いものを破棄する前に代替を作成
`prevent_destroy`	`terraform destroy` でこのリソースを対象にするとエラー
`ignore_changes`	指定属性のドリフトを検出しない
`replace_triggered_by`	参照リソースが変わった場合に強制置換
`precondition`	apply 前に前提条件を検証
`postcondition`	apply 後に結果を検証

よく使うパターン

ループと条件

```
# 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
```

便利な関数

`file("key.pub")`	ファイルの内容を読み取り
`join(", ", list)`	リストを文字列に結合
`lookup(map, key, default)`	フォールバック付きのマップ検索
`length(list)`	要素数
`toset(["a", "b"])`	リストをセットに変換（for_each 用）
`try(expr, fallback)`	expr がエラーならフォールバックを返す
`templatefile(path, vars)`	テンプレートファイルをレンダリング