

TERRAFORM 快速参考

Provider、资源、变量、状态、模块

基础

核心工作流

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

常用命令

```
terraform init 初始化工作目录, 下载 provider
terraform plan 预览执行计划, 不实际变更
terraform apply 将变更应用到基础设施
terraform destroy 销毁所有托管资源
terraform fmt 将 .tf 文件格式化为标准风格
terraform validate 检查配置语法
terraform show 显示当前状态或计划
terraform output 打印输出值
```

Provider

Provider 配置

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

Provider 说明

source Registry 地址 (`hashicorp/aws`、`hashicorp/google`)
version 版本约束 (`>= 5.0`, `>= 3.0, < 4.0`)
terraform.lock.hcl 锁文件 - 提交到版本控制
alias 同一 provider 使用多套配置

资源

Resource 块

```
resource "aws_instance" "web" {
  ami           = "ami-0c55b159cbfaf1f0"
  instance_type = "t3.micro"
  tags = { Name = "web-server" }
```

Resource 元参数

depends_on 显式声明对另一资源的依赖
count 创建多个实例 (`count = 3`)
for_each 从 map 或 set 创建多个实例
provider 选择非默认 provider 别名
lifecycle 自定义创建/更新/销毁行为

引用资源属性

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

变量

声明变量

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

设置变量值

```
-var 'region=us-west-2'  CLI 参数
-var-file=prod.tfvars  从 .tfvars 文件加载
terraform.tfvars        存在时自动加载
TF_VAR_region           环境变量
Interactive prompt      无默认值时在 plan/apply 时提示输入
```

变量类型

```
string  `us-east-1`
number  `42`
bool    `true` / `false`
list(string)  `["a", "b"]`
map(string)   `{ key = "val" }`
object({...}) 带命名属性的结构化类型
```

输出

定义输出

```
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
}
```

输出命令

```
terraform output      打印所有输出
terraform output instance_ip 打印指定输出
terraform output -json  JSON 格式, 适合脚本使用
sensitive = true     在 CLI 输出中隐藏值
module.vpc.vpc_id    访问子模块的输出
```

状态

远程后端

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

状态命令

```
terraform state list  列出状态中的所有资源
terraform state show <addr> 显示资源的属性
terraform state mv <src> <dst> 在状态中重命名/移动资源
terraform state rm <addr> 从状态中移除资源 (保留基础设施)
```

```
terraform state pull 将远程状态下载到 stdout
terraform import <addr> <id> 将已有基础设施导入状态
```

模块

使用模块

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

模块来源

```
"./modules/vpc" 本地路径
"terraform-aws-modules/vpc/aws" Terraform Registry
"github.com/org/repo/module" GitHub 仓库
"s3::https://bucket/module.zip" S3 存储桶
```

模块结构

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

数据源

读取已有资源

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

常用数据源

```
data.aws_ami 按过滤条件查找 AMI
data.aws_vpc 查找已有 VPC
data.aws_caller_identity 当前 AWS 账号 ID
data.aws_region 当前 AWS 区域
data.terraform_remote_state 从另一个状态文件读取输出
data.external 运行外部程序获取数据
```

生命周期

Lifecycle 规则

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

Lifecycle 选项

```
create_before_destroy 先创建替换资源再销毁旧资源
prevent_destroy        terraform destroy 指向此资源时报错
ignore_changes         不检测列出属性的漂移
replace_triggered_by   引用资源变更时强制替换
precondition           apply 前验证前置条件
postcondition          apply 后验证结果
```

常用模式

循环与条件

```
# 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
```

常用函数

```
file("key.pub")  读取文件内容
join(" ", list)  将列表连接为字符串
lookup(map, key, default) 带默认值的 map 查找
length(list)     元素数量
toset(["a", "b"]) 将列表转换为 set (用于 for_each)
try(expr, fallback) expr 报错时返回 fallback
templatefile(path, vars) 渲染模板文件
```