Terraform

Terraform es una herramienta de *Infraestructura como Código* (IaC): describes en ficheros de texto qué infraestructura quieres (una red, una VM, un bucket, un contenedor) y Terraform calcula qué cambios hacen falta para llegar a ese estado y los aplica de forma reproducible.

Ideas clave (sin complicarse)

- **Provider**: el "conector" con una plataforma (AWS, Azure, Google Cloud, Docker, etc.).
- **Resource**: una pieza de infraestructura (un bucket, una red, una instancia, un contenedor).
- **Estado (state)**: Terraform guarda un "mapa" de lo que ha creado para poder comparar lo deseado con lo real.
- **Plan / Apply**: primero te enseña qué va a cambiar (**plan**) y luego lo ejecuta (**apply**).
- **Variables / Outputs**: para parametrizar y para "devolver" valores útiles (IDs, URLs, nombres).
- **Módulos**: empaquetas recursos reutilizables.

Flujo típico de trabajo

En una carpeta con tus .tf:

```
terraform init     # descarga providers y prepara el proyecto
terraform fmt      # formatea los .tf
terraform validate # valida sintaxis
terraform plan     # previsualiza cambios
terraform apply    # aplica cambios
terraform destroy  # elimina lo creado por ese proyecto
```

Ejemplo 1 (muy simple): crear un fichero local

Este ejemplo no requiere cloud, sirve para entender el ciclo plan/apply y el concepto de “estado”.

main.tf

```
terraform {
  required_providers {
    local = {
      source  = "hashicorp/local"
      version = "~> 2.0"
    }
  }
}

provider "local" {}

resource "local_file" "saludo" {
  filename = "${path.module}/hola.txt"
  content  = "Hola, Terraform \n"
}
```

Ejecuta:

```
terraform init
terraform plan
terraform apply
```

Resultado: se crea hola.txt y Terraform guarda el estado en terraform.tfstate.

Ejemplo 2: contenedor Docker (fácil para ver “infra” real)

Necesitas Docker instalado y funcionando.

main.tf

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0"
    }
  }
}

provider "docker" {}

resource "docker_image" "nginx" {
  name = "nginx:stable"
}

resource "docker_container" "web" {
  name  = "web-nginx"
  image = docker_image.nginx.image_id

  ports {
    internal = 80
    external = 8080
  }
}

output "url" {
  value = "http://localhost:8080"
}
```

Ejecuta:

```
terraform init
terraform apply
```

Luego abre la URL del output. Para borrarlo:

terraform destroy

Ejemplo 3: variables y outputs (para no “hardcodear”)

variables.tf

```
variable "nombre" {
  type        = string
  description = "Nombre del recurso (ejemplo)"
  default     = "hola"
}
```

main.tf (usando esa variable)

```
resource "local_file" "saludo" {
  filename = "${path.module}/${var.nombre}.txt"
  content  = "Fichero creado con variable.\n"
}

output "ruta_fichero" {
  value = local_file.saludo.filename
}
```

Cambiar el valor al vuelo:

terraform apply -var="nombre=prueba"