

Introducción a Terraform y Salt Project

Unidad 01. Caso práctico 01

Autor: Sergi García

Actualizado Noviembre 2025



Licencia



Reconocimiento - No comercial - CompartirIgual (BY-NC-SA): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se ha de hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán diferentes símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE

Proyecto: Generación de token y archivo local usando Terraform	3
1. Explicación: ¿Qué haremos en este caso práctico?	3
2. Objetivo	3
3. Estructura del proyecto	4
4. Ejecución paso a paso	4
Requisitos previos	4
Descarga del proyecto	4
Ejecución del caso práctico	4
Limpieza	6
5. Explicación del flujo	6
Resultado final	6
6. Detalle del contenido de los ficheros	6
main.tf	6
outputs.tf	7

UNIDAD 01 - CASO PRÁCTICO 01

PROYECTO: GENERACIÓN DE TOKEN Y ARCHIVO LOCAL USANDO TERRAFORM

1. EXPLICACIÓN: ¿QUÉ HAREMOS EN ESTE CASO PRÁCTICO?

En este primer caso práctico realizaremos un “**Hola Mundo**” con Terraform, ideal para comprender su funcionamiento básico sin necesidad de conectarnos a ningún servicio en la nube.

El objetivo es **generar un token aleatorio y guardarlo en un archivo local**, utilizando tres *providers* esenciales de Terraform:

- **random** → para crear contraseñas o tokens aleatorios.
- **local** → para generar archivos en el sistema de ficheros.
- **null** → para ejecutar comandos locales o mostrar mensajes en consola.

Terraform actuará como motor de orquestación: creará los recursos, almacenará el token dentro de un archivo llamado **token.txt** y mostrará los resultados en pantalla. De esta manera aprenderemos cómo Terraform **declara, aplica y destruye recursos**, incluso en entornos totalmente *offline*.

Con este ejercicio pondremos en práctica conceptos fundamentales como:

- La estructura básica de un proyecto Terraform (main.tf y outputs.tf).
- La interacción entre distintos *providers*.
- El flujo completo de ejecución (init, apply, destroy).

Al finalizar, obtendremos un **proyecto local funcional** capaz de generar archivos y valores dinámicos sin depender de ninguna infraestructura externa, sentando las bases para los despliegues más complejos de las siguientes unidades.

2. OBJETIVO

Este caso práctico es una suerte de “Hola mundo”, donde no vamos a desplegar ninguna máquina con Terraform, pero vamos viendo como funciona, vemos algunos providers, etc. En concreto, en este caso práctico vamos a generar un **token aleatorio** y guardarlo en un archivo local usando recursos de Terraform completamente *offline*.

Se trabajará con los siguientes *providers*:

- **random** → para generar contraseñas o tokens
- **local** → para crear archivos en el sistema de ficheros
- **null** → para ejecutar comandos locales

 Ideal para aprender a usar recursos *no cloud*, útiles en testing, scripts, provisiones ligeras y entornos locales.

 3. ESTRUCTURA DEL PROYECTO

UD01CasoPractico01/

```
└── main.tf  
└── outputs.tf
```

 4. EJECUCIÓN PASO A PASO Requisitos previos

Para ejecutar este caso práctico, necesitas tener instalados en tu máquina:

- **Terraform**

 Puedes comprobar que están correctamente instalados ejecutando:

```
terraform version
```

 Descarga del proyecto

1. Descarga los archivos correspondientes a este caso práctico, asegurándote de mantener la **estructura de directorios descrita en el punto anterior**.
2. Una vez descargados, abre una terminal y sitúate en el directorio raíz del proyecto (donde se encuentra el archivo main.tf).

 Ejecución del caso práctico

Inicializa el proyecto mediante la orden:

```
terraform init
```

```
alumno@alumno-virtualbox:~/Desktop/UD01-Caso01$ terraform init  
Initializing the backend...  
Initializing provider plugins...  
- Finding hashicorp/null versions matching "~> 3.0"..."  
- Finding hashicorp/random versions matching "~> 3.0"..."  
- Finding hashicorp/local versions matching "~> 2.0"..."  
- Installing hashicorp/null v3.2.4...  
- Installed hashicorp/null v3.2.4 (signed by HashiCorp)  
- Installing hashicorp/random v3.7.2..."  
- Installed hashicorp/random v3.7.2 (signed by HashiCorp)  
- Installing hashicorp/local v2.5.3..."  
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
```

Tras ello aplicamos la infraestructura:

```
terraform apply
```

```
alumno@alumno-virtualbox:~/Desktop/UD01-Caso01$ terraform apply
Terraform used the selected providers to generate the following
with the following symbols:
+ create

Terraform will perform the following actions:

# local_file.token_file will be created
+ resource "local_file" "token_file" {
    + content          = (sensitive value)
    + content_base64sha256 = (known after apply)
    + content_base64sha512 = (known after apply)
    + content_md5      = (known after apply)
    + content_sha1     = (known after apply)
    + content_sha256   = (known after apply)
    + content_sha512   = (known after apply)
```

A mitad del proceso de “terraform apply” podemos ver la ejecución del comando, como en la siguiente imagen:

```
null_resource.notify: Creating...
null_resource.notify: Provisioning with 'local-exec'...
null_resource.notify (local-exec): Executing: ["./bin/sh" "-c" "echo '✓ Archivo creado exitosamente con token aleatorio.'"]
random_password.secure_token: Creating...
null_resource.notify (local-exec): ✓ Archivo creado exitosamente con token aleatorio.
```

Tras realizar el proceso, se habrán creado los siguientes outputs:

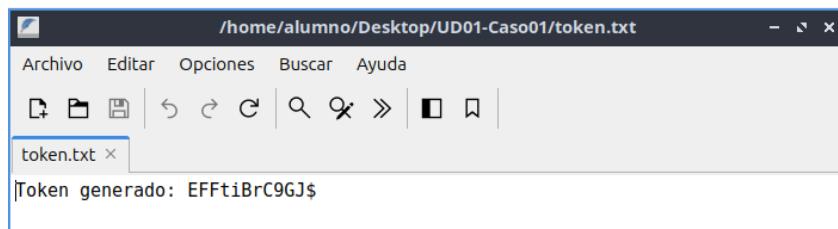
```
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:

file_path = "./token.txt"
token_value = <sensitive>
alumno@alumno-virtualbox:~/Desktop/UD01-Caso01$
```

En ellos, se habrá creado en el directorio del caso práctico el fichero “token.txt” y en su interior, tendrá un texto “Token generado:” seguido de un valor generado aleatoriamente.

Aquí un ejemplo del fichero y su contenido:



Limpieza

Cuando termines el laboratorio, puedes destruir todos los recursos creados con la orden:

```
terraform destroy
```

```
alumno@alumno-virtualbox:~/Desktop/UD01-Caso01$ terraform destroy
null_resource.notify: Refreshing state... [id=3758118275895496470]
random_password.secure_token: Refreshing state... [id=none]
local_file.token_file: Refreshing state... [id=d4cd8ccc18efef80e2d3d9e48c1535a0988e5f34]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy
```

Y si lo deseas, puedes borrar también el archivo token.txt manualmente:

```
rm token.txt
```

5. EXPLICACIÓN DEL FLUJO

- random_password: genera el token.
- local_file: crea el archivo con el contenido dinámico.
- null_resource + local-exec: ejecuta un comando local al final (mensaje en terminal).
- output: muestra el token generado y la ubicación del archivo.

Resultado final

- Proyecto local funcional sin necesidad de nube ni contenedores
- Integra múltiples providers auxiliares
- Útil como plantilla para generar credenciales o archivos de forma dinámica
- Muy fácil de extender con plantillas, scripts o lógica adicional

6. DETALLE DEL CONTENIDO DE LOS FICHEROS

main.tf

```
# 🔑 Declaración de los providers necesarios para el proyecto
terraform {
  required_providers {
    local = {
      source  = "hashicorp/local"      # Provider para manejar archivos locales
      version = "~> 2.0"
    }
    null  = {
      source  = "hashicorp/null"      # Provider para ejecutar comandos locales
      version = "~> 3.0"
    }
    random = {
      source  = "hashicorp/random"    # Provider para generar valores aleatorios
      version = "~> 3.0"
    }
  }
}

# 📁 Inicialización de Los providers
```

```

provider "local" {} # Sin configuración adicional
provider "null" {} # Sin configuración adicional
provider "random" {} # Sin configuración adicional

# 🔒 Genera un token aleatorio
resource "random_password" "secure_token" {
  length = 12          # Longitud del token
  special = true        # Incluye caracteres especiales
}

# 📄 Crea un archivo local con el token
resource "local_file" "token_file" {
  filename = "${path.module}/token.txt"                      # Ruta del archivo
  content  = "Token generado: ${random_password.secure_token.result}" # Contenido del archivo
}

# 💻 Muestra un mensaje en la terminal
resource "null_resource" "notify" {
  provisioner "local-exec" {
    command = "echo '✅ Archivo creado exitosamente con token aleatorio.'" # Mensaje de confirmación
  }
}

```

outputs.tf

```

# 🔎 Definimos una salida llamada "token_value"
output "token_value" {
  # Descripción de lo que representa esta salida
  description = "Token generado aleatoriamente"

  # Valor que se mostrará: el resultado del recurso random_password Llamado secure_token
  value = random_password.secure_token.result

  # Marcamos la salida como sensible para que no se muestre en texto plano en Los Logs o consola
  sensitive = true
}

# 📄 Definimos otra salida llamada "file_path"
output "file_path" {
  # Descripción de la salida: mostrará la ruta del archivo creado Localmente
  description = "Ruta del archivo Local generado"

  # Valor que se mostrará: el nombre (ruta) del archivo generado por el recurso Local_file
  value = local_file.token_file.filename
}

```