Introducción a Terraform y Salt Project

Unidad 07. Caso práctico 01

Autor: Sergi García

Actualizado Noviembre 2025

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL
UNIÓN EUROPEA
Fondo Social Europeo
El FSE invierte en tu futuro
GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura, Universitats i Ocupació
CFR CEFIRE
FORMACIÓ PROFESSIONAL ENSENYANCES ARTÍSTIQUES I ESPORTIVES
FPCV
Formació Professional Comunitat Valenciana

Licencia

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

UNIDAD 07 - CASO PRÁCTICO 01

🔥 Proyecto: Simulación de AWS con Terraform y LocalStack

📘 1. Explicación: ¿Qué haremos en este caso práctico?

En este caso práctico aprenderás a simular un entorno de Amazon Web Services (AWS) de forma local, utilizando LocalStack como plataforma de emulación y Terraform como herramienta de aprovisionamiento de infraestructura.

El objetivo es que experimentes cómo Terraform puede desplegar recursos AWS —como buckets S3 o tablas DynamoDB— sin necesidad de conectarte a la nube real, sin credenciales y sin costes.

Durante el caso práctico:

- Pondremos en marcha LocalStack en tu máquina mediante Docker.
- Configuraremos Terraform para usar el provider de AWS redirigido al endpoint local de LocalStack (http://localhost:4566).
- Crearemos y verificaremos un bucket S3 simulado con un archivo de prueba dentro.
- Consultaremos los resultados con AWS CLI, igual que si trabajáramos en la nube real.
- Finalmente, destruiremos los recursos de forma segura y limpia con terraform destroy.

Este caso te permitirá comprender cómo Terraform, Docker y LocalStack se integran para ofrecer un entorno local de aprendizaje y testing 100 % funcional, replicando el comportamiento de la nube de AWS de forma totalmente gratuita y sin riesgo.

💡 La idea es demostrar que se puede practicar y validar infraestructura de tipo AWS completamente en local, sin cuenta ni acceso a Internet.

🎯 2. Objetivo

En este caso práctico se persiguen los siguientes objetivos:

- ✅ Instalar y ejecutar LocalStack para simular servicios de AWS localmente.
- ✅ Configurar Terraform para comunicarse con LocalStack en lugar de la nube real.
- ✅ Desplegar recursos AWS simulados (por ejemplo, un bucket S3 y un archivo).
- ✅ Consultar los recursos con AWS CLI redirigida al endpoint local (localhost:4566).

🧱 3. Estructura del proyecto

UD07-CasoPractico01/
├── main.tf
├── variables.tf
└── outputs.tf

🌱 4. INSTALACIÓN Y PUESTA EN MARCHA DE LOCALSTACK

Para ejecutar LocalStack en tu equipo, necesitamos tener **Python 3** y su gestor de paquetes **pip**, ya que LocalStack se distribuye como una herramienta Python.

A continuación te explico el proceso paso a paso 👇

Instalar Python3 y PIP

En la mayoría de sistemas Linux, puedes instalar ambos paquetes directamente desde los repositorios oficiales. Ejecuta en tu terminal:

```
sudo apt update
sudo apt install python3 python3-pip
```

💡 *Python es necesario para poder ejecutar la CLI (interfaz de línea de comandos) de LocalStack, mientras que pip nos permitirá instalarla fácilmente desde el repositorio oficial de Python (PyPI).*

Instalar LocalStack con pip

Una vez tengas Python y pip instalados, ejecuta el siguiente comando para descargar e instalar LocalStack en tu sistema:

```
python3 -m pip install localstack
```

```
alumno@alumno-virtualbox:~/Desktop/UD07-Caso01$ python3 -m pip install localstack --break-system-packages
Defaulting to user installation because normal site-packages is not writeable
Collecting localstack
  Downloading localstack-4.10.0.tar.gz (5.9 kB)
  Preparing metadata (setup.py) ... done
Collecting localstack-core (from localstack)
  Downloading localstack_core-4.10.0-py3-none-any.whl.metadata (5.9 kB)
Collecting localstack-ext==4.10.0 (from localstack)
  Downloading localstack_ext-4.10.0.tar.gz (8.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.9/8.9 MB 5.5 MB/s eta 0:00:00
```

Este comando descargará la última versión estable de LocalStack y sus dependencias.

Luego instalaremos el cliente con

```
python3 -m pip install localstack-client
```

```
alumno@alumno-virtualbox:~/Desktop/UD07-Caso01$ python3 -m pip install localstack-client --break-system-packages
Defaulting to user installation because normal site-packages is not writeable
Collecting localstack-client
  Downloading localstack_client-2.10.tar.gz (11 kB)
  Preparing metadata (setup.py) ... done
Collecting boto3 (from localstack-client)
  Downloading boto3-1.40.64-py3-none-any.whl.metadata (6.6 kB)
Collecting botocore<1.41.0,>=1.40.64 (from boto3->localstack-client)
  Downloading botocore-1.40.64-py3-none-any.whl.metadata (5.7 kB)
Collecting jmespath<2.0.0,>=0.7.1 (from boto3->localstack-client)
  Downloading jmespath-1.0.1-py3-none-any.whl.metadata (7.6 kB)
Collecting s3transfer<0.15.0,>=0.14.0 (from boto3->localstack-client)
  Downloading s3transfer-0.14.0-py3-none-any.whl.metadata (1.7 kB)
```

Al finalizar, puedes verificar que la instalación fue exitosa con:

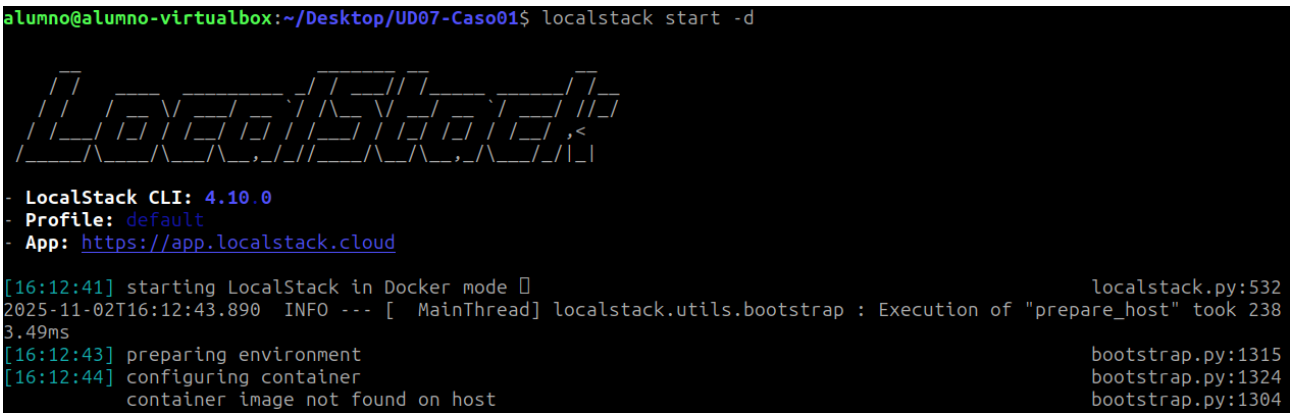
```
localstack --version
```

Iniciar LocalStack

Para poner en marcha el entorno local de AWS, simplemente ejecuta:

```
localstack start -d
```

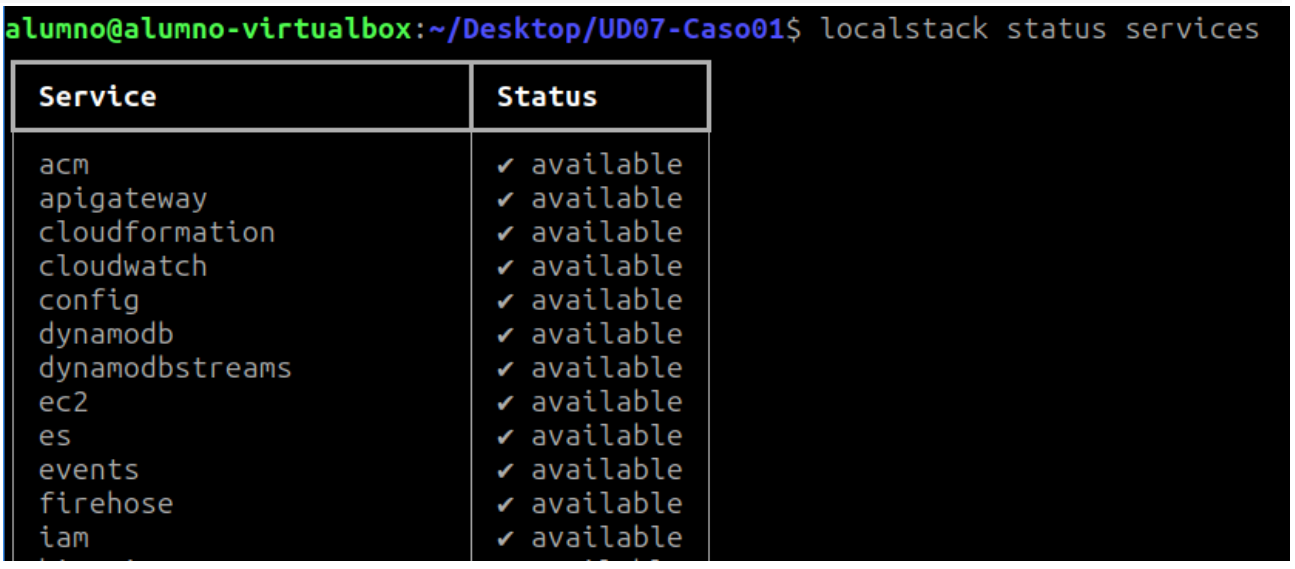
🔍 La opción -d (de "detached") hace que LocalStack se ejecute en segundo plano, permitiéndote seguir utilizando la terminal.



Esto iniciará varios contenedores Docker que simulan los servicios principales de AWS, como **S3**, **Lambda**, **DynamoDB** y otros. Durante la primera ejecución, puede tardar unos minutos, ya que descargará las imágenes necesarias.

Puedes comprobar que LocalStack está en funcionamiento ejecutando:

```
localstack status services
```



O bien verificando la salud de los servicios a través del endpoint local:

```
curl http://localhost:4566/health
```

Deberías ver una salida JSON indicando qué servicios están activos, por ejemplo:

```
{
  "services": {
    "s3": "running",
    "dynamodb": "running",
    "lambda": "running"
  }
}
```

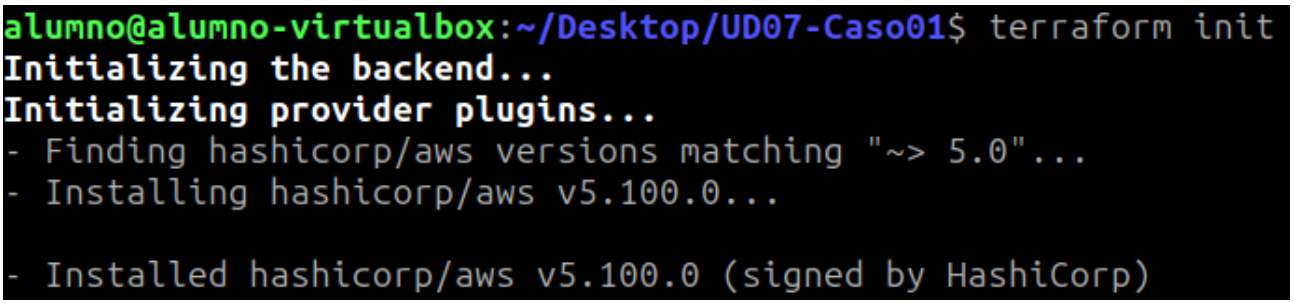
💡 A partir de este momento, tu máquina está simulando una nube AWS local completa. Terraform y AWS CLI podrán interactuar con ella igual que lo harían con AWS real.

🚀 5. Ejecución paso a paso (Terraform + LocalStack)

✅ **Previa**: LocalStack ya está instalado y corriendo en segundo plano con localstack start -d.

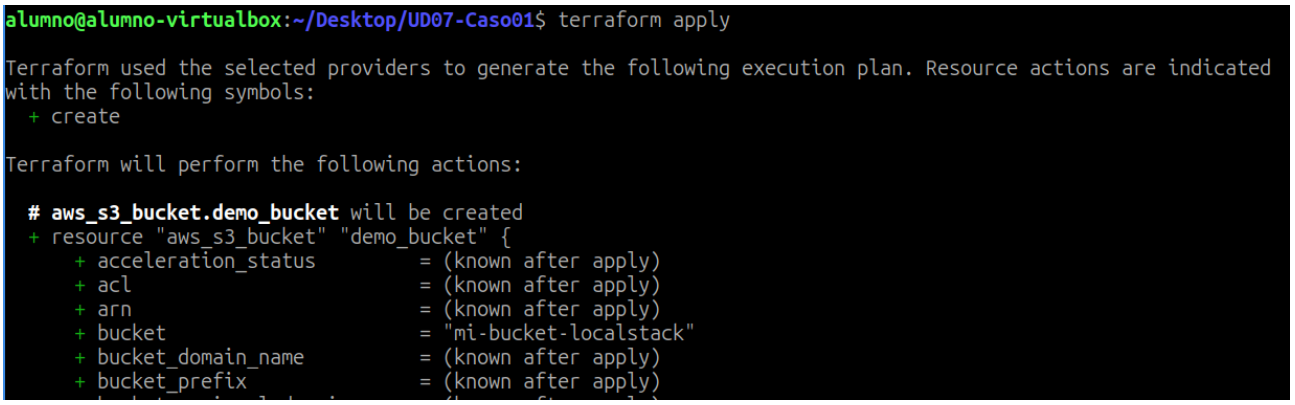
Inicializa Terraform con “terraform init”

```
terraform init
```



Aplicamos la infraestructura con

```
terraform apply
```



Al finalizar, obtendremos una salida similar a esta

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:
bucket_name        = "mi-bucket-localstack"
object_key         = "hola.txt"
endpoint_localstack = "http://localhost:4566"
```

Una vez Terraform ha creado los recursos en LocalStack (el bucket S3 y el archivo hola.txt), podemos comprobarlo desde la línea de comandos usando la AWS CLI.

Recuerda: no estamos accediendo a la nube real, sino al endpoint local de LocalStack (http://localhost:4566), que emula el comportamiento de AWS.

Verifica con AWS CLI (apunta al endpoint de LocalStack)

Instala AWS CLI con el comando:

```
python3 -m pip install awscli
```

```
alumno@alumno-virtualbox:~/Desktop/UD07-Caso01$ python3 -m pip install awscli --break-system-packages
Defaulting to user installation because normal site-packages is not writeable
Collecting awscli
  Downloading awscli-1.42.64-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: botocore==1.40.64 in /home/alumno/.local/lib/python3.12/site-packages (from awscli)
 (1.40.64)
Collecting docutils<=0.19,>=0.18.1 (from awscli)
  Downloading docutils-0.19-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: s3transfer<0.15.0,>=0.14.0 in /home/alumno/.local/lib/python3.12/site-packages (fro
m awscli) (0.14.0)
Requirement already satisfied: PyYAML<6.1,>=3.10 in /usr/lib/python3/dist-packages (from awscli) (6.0.1)
Requirement already satisfied: colorama<0.4.7,>=0.2.5 in /usr/lib/python3/dist-packages (from awscli) (0.4.6)
Collecting rsa<4.8,>=3.1.2 (from awscli)
  Downloading rsa-4.7.2-py3-none-any.whl.metadata (3.6 kB)
```

Configurar credenciales "dummy" (recomendada)

Ejecuta en la terminal "aws configure" y cuando te pida los datos, escribe valores ficticios:

```
aws configure
```

```
alumno@alumno-virtualbox:~/Desktop/UD07-Caso01$ aws configure
AWS Access Key ID [None]: test
AWS Secret Access Key [None]: test
Default region name [None]: us-east-1
Default output format [None]: json
alumno@alumno-virtualbox:~/Desktop/UD07-Caso01$
```

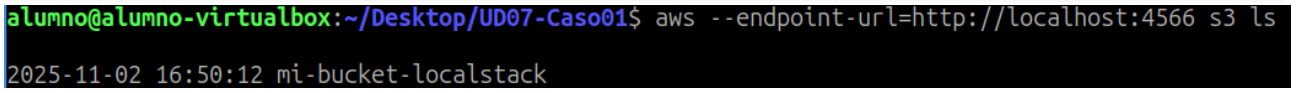
💡 **Estos valores no se usan realmente; LocalStack solo necesita que existan para que la CLI funcione.**

Una ves todo listo, podemo comenzar a contactar mediante awscli a nuestro AWS simulado vía LocalStack.

Lista buckets:

Este comando le pide a la AWS CLI que muestre todos los buckets S3 disponibles en el endpoint indicado. Normalmente, aws s3 ls mostraría los buckets reales de tu cuenta de AWS, pero al usar --endpoint-url=http://localhost:4566, la petición se redirige al servicio S3 simulado por LocalStack.

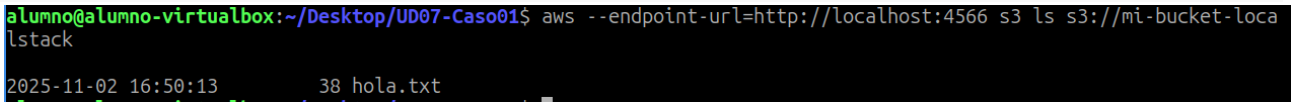
```
aws --endpoint-url=http://localhost:4566 s3 ls
```



Lista objetos del bucket:

Este comando entra dentro del bucket simulado (s3://mi-bucket-localstack) y lista los objetos almacenados. En nuestro caso, debería aparecer el archivo hola.txt que Terraform creó con el recurso aws_s3_object.

```
aws --endpoint-url=http://localhost:4566 s3 ls s3://mi-bucket-localstack
```



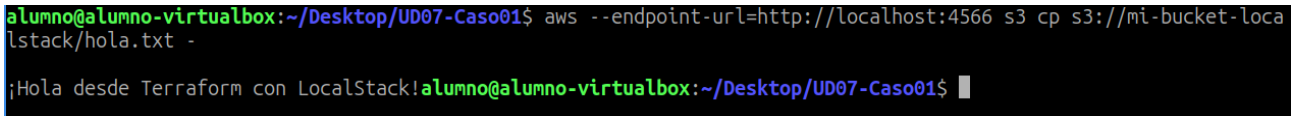
Lee el archivo que subió Terraform:

Este comando copia (descarga) el archivo hola.txt desde el bucket simulado y lo muestra directamente en pantalla (gracias al guion -, que significa salida estándar en UNIX).

```
aws --endpoint-url=http://localhost:4566 s3 cp s3://mi-bucket-localstack/hola.txt -
```

Salida esperada:

```
¡Hola desde Terraform con LocalStack!
```



Finalizando el caso práctico

Cuando acabes puedes destruir la infraestructura de Terraform con:

```
terraform destroy
```

Y también detener LocalStack si ya no lo usarás:

```
localstack stop
```

🧠 6. CÓMO FUNCIONA EL FLUJO

1. Terraform analiza la configuración. Lee los archivos main.tf, variables.tf y outputs.tf para entender qué infraestructura debe crear.
2. Se comunica con el provider AWS
3. Redirección al endpoint local. En lugar de enviar las llamadas a la nube de Amazon, el

provider las envía al endpoint definido en el bloque del provider:
   a. Aquí es donde LocalStack actúa como intermediario y "responde" igual que AWS real.
4. LocalStack crea los servicios simulados
   a. Usa contenedores Docker para ejecutar versiones locales de S3, DynamoDB, Lambda y otros.
5. Cuando Terraform pide "crear un bucket S3", LocalStack lo registra internamente y guarda su estado.
   a. Terraform actualiza el estado (terraform.tfstate)
   b. Guarda un registro exacto de los recursos que cree que existen.
   c. En este caso, Terraform no sabe que es una simulación: para él, es AWS real.
6. Consulta con AWS CLI. Si ejecutas: "aws --endpoint-url=http://localhost:4566 s3 ls" la CLI también se conecta al endpoint local, accediendo al mismo servicio S3 que Terraform creó.

🧩 7. Detalle de los ficheros

📄 main.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region                      = "us-east-1"
  access_key                  = "test"     # Claves dummy
  secret_key                  = "test"
  skip_credentials_validation = true
  skip_metadata_api_check     = true
  skip_requesting_account_id  = true # 👈 IMPORTANTE: evita validación de cuenta real
  s3_use_path_style           = true

  endpoints {
    s3 = "http://localhost:4566"
  }
}


# Crear un bucket S3 simulado
resource "aws_s3_bucket" "demo_bucket" {
  bucket = var.bucket_name
}

# Subir un objeto al bucket
resource "aws_s3_object" "demo_file" {
  bucket = aws_s3_bucket.demo_bucket.id
  key    = "hola.txt"
  content = "¡Hola desde Terraform con LocalStack!"
```

```
}

output "bucket_name" {
  value = aws_s3_bucket.demo_bucket.bucket
}
```

📄 variables.tf

```
variable "bucket_name" {
 description = "Nombre del bucket S3 simulado"
 default     = "mi-bucket-localstack"
}
```

📄 outputs.tf

```
output "archivo_subido" {
  value = aws_s3_object.demo_file.key
}

output "endpoint_localstack" {
  value = "http://localhost:4566"
}
```