Introducción a Terraform y Salt Project

Unidad 05. 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 05 - Caso práctico 01

🔥 Proyecto: Infraestructura local automatizada con Terraform y Salt Project

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

En este caso práctico vamos a construir, paso a paso, un **laboratorio completo de administración automatizada de sistemas** utilizando **Terraform**, **Docker** y **Salt Project** (Broadcom).

El objetivo es comprender cómo **automatizar la creación y configuración de infraestructura local**, simulando un entorno de producción real donde un servidor maestro (**Salt Master**) orquesta la configuración de múltiples clientes (**Salt Minions**) dentro de contenedores Docker.

Todo el proceso se controla desde Terraform, que actúa como "orquestador" de los contenedores.

🎯 2. Objetivo

Desplegar con **Terraform** (provider Docker) un laboratorio local compuesto por:

- 🧠 1 contenedor Salt Master
- 🧩 N contenedores Salt Minion (configurable por variable)
- 🔌 Comunicación automática Master → Minions
- ⚙️ Aplicación de estados Salt para:
  - Crear usuarios.
  - Instalar y configurar NGINX (escucha en 8088 interno).
  - Exponer cada minion al host en puerto único (8080, 8081, ...).

Todo parametrizable y reproducible con un solo terraform apply. 🚀

💡 3. Arquitectura

La infraestructura que vamos a desplegar estará formada por:

Rol	Contenedor	Descripción
🧠 **Master**	salt-master	Servidor central de Salt. Gestiona los estados (.sls) y la comunicación con los minions.
🤖 **Minions**	minion-1, minion-2, minion-3, ...	Contenedores que se conectan automáticamente al master y aplican las configuraciones definidas.
🌐 **Red interna Docker**	salt-net	Permite la comunicación entre master y minions sin exponer los puertos internamente.

Cada minion ejecutará un servicio NGINX que mostrará una página HTML personalizada con su nombre.

🧱 4. Estructura del Proyecto

```
UD05-Caso02/
├── main.tf            # Configuración principal de Terraform
├── variables.tf       # Definición de variables de entrada
├── terraform.tfvars   # Valores concretos (número de minions, puertos base)
├── outputs.tf         # Resultados finales (IPs y puertos)
│
├── salt-master/
│   ├── Dockerfile     # Imagen personalizada del master (Ubuntu 24.04 + Salt)
│   └── master.conf    # Configuración de Salt Master
│
├── salt-minion/
│   ├── Dockerfile     # Imagen personalizada del minion (Ubuntu 24.04 + Salt + NGINX)
│   └── minion.conf    # Configuración base del minion
│
└── salt-states/
    ├── users.sls      # Creación del usuario "alumno"
    ├── nginx/
    │   ├── init.sls   # Instalación y configuración del servicio NGINX
    │   ├── config.sls # Archivo de configuración NGINX
    │   └── index.html.j2# Plantilla HTML personalizada para cada minion
    └── top.sls        # Mapa de estados aplicados a todos los minions
```

🧠 5. Explicación del flujo

1. Terraform crea una red Docker interna (salt-net).
2. Construye dos imágenes personalizadas:
   a. salt-master:ubuntu24 → con Salt Master instalado.
   b. salt-minion:ubuntu24 → con Salt Minion + NGINX instalados.
3. Lanza el contenedor del Salt Master y expone sus puertos (4505 y 4506).
4. Lanza los N minions, que se conectan automáticamente al master gracias a la variable SALT_MASTER.
5. Los minions aplican los estados definidos en /srv/salt, creando usuarios y desplegando NGINX. Aunque se podría automatizar con un provider, este paso lo haremos manualmente con fines didácticos.
6. El resultado: cada minion sirve una página HTML accesible desde el navegador en diferentes puertos (8080, 8081, 8082...).

✅ 6. Ejecución paso a paso

1. Inicializa el proyecto:

Este comando prepara el entorno de trabajo de Terraform.

```
terraform init
```

```
alumno@alumno-virtualbox:~/Desktop/UD05-Caso01$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "~> 3.0"...
- Installing kreuzwerker/docker v3.6.2...
- Installed kreuzwerker/docker v3.6.2 (self-signed, key ID BD080C4571C6104C)
Partner and community providers are signed by their developers.
```

2. Configurar variables en terraform.tfvars

Con esas variables, Terraform construirá 1 master + 3 minions interconectados en una red Docker local.

```
minion_count    = 4
nginx_base_port = 8080
```

3. **Desplegamos nuestro sistema**

```
terraform apply -auto-approve
```

```
alumno@alumno-virtualbox:~/Desktop/UD05-Caso01$ terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
with the following symbols:
  + create

Terraform will perform the following actions:

  # docker_container.salt_master will be created
  + resource "docker_container" "salt_master" {
      + attach                                      = false
      + bridge                                      = (known after apply)
      + command                                     = [
          + "salt-master",
          + "-l",
          + "info",
        ]
```

El resultado esperado es similar a:

```
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

host_ports = {
  "minion-1" = 8080
  "minion-2" = 8081
  "minion-3" = 8082
}
master_ip = "172.19.0.2"
```

```
minions_ips = {
  "minion-1" = "172.19.0.5"
  "minion-2" = "172.19.0.3"
  "minion-3" = "172.19.0.4"
}
```

```
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

host_ports = {
  "minion-1" = 8080
  "minion-2" = 8081
  "minion-3" = 8082
}
master_ip = "172.19.0.2"
minions_ips = {
  "minion-1" = "172.19.0.5"
  "minion-2" = "172.19.0.3"
  "minion-3" = "172.19.0.4"
}
alumno@alumno-virtualbox:~/Desktop/UD05-Caso01$
```

4. **Verificar que los contenedores se han generado correctamente**

```
docker ps
```

```
alumno@alumno-virtualbox:~/Desktop/UD05-Caso01$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS
      NAMES
cfcb6451d95b   3b3ee8052285   "bash -c 'echo 'mast…"   7 minutes ago   Up 7 minutes   0.0.0.0:8082->8088/tcp
      minion-3
73e8c9757f64   3b3ee8052285   "bash -c 'echo 'mast…"   7 minutes ago   Up 7 minutes   0.0.0.0:8080->8088/tcp
      minion-1
46b15cca83f7   3b3ee8052285   "bash -c 'echo 'mast…"   7 minutes ago   Up 7 minutes   0.0.0.0:8081->8088/tcp
      minion-2
438cad07f8f9   599cb61248b8   "salt-master -l info"    8 minutes ago   Up 8 minutes   0.0.0.0:4505-4506->4505-4506
/tcp   salt-master
alumno@alumno-virtualbox:~/Desktop/UD05-Caso01$
```

5. Comprobar conectividad Master ↔ Minions:

Para confirmar que todos los minions han registrado su clave y están conectados al master.

```
docker exec -it salt-master bash -lc "salt '*' test.ping"
```

```
alumno@alumno-virtualbox:~/Desktop/UD05-Caso01$ docker exec -it salt-master bash -lc "salt '*' test.ping"
minion-1:
    True
minion-2:
    True
minion-3:
    True
```

6. Aplicar los estados definidos en /srv/salt

```
docker exec -it salt-master bash -lc "salt '*' state.apply"
```

✔️ **Qué hace:** Ejecuta en todos los minions los ficheros .sls del directorio /srv/salt:

- users.sls → crea el usuario alumno.
- nginx/init.sls → instala y configura NGINX.
- nginx/index.html.j2 → genera una página HTML personalizada para cada minion.

```
alumno@alumno-virtualbox:~/Desktop/UD05-Caso01$ docker exec -it salt-master bash -lc "salt '*' state.apply"
minion-1:
----------
          ID: alumno-user
    Function: user.present
        Name: alumno
      Result: True
     Comment: New user alumno created
     Started: 23:55:23.587693
    Duration: 6864.731 ms
     Changes:
              ----------
              fullname:
                  Usuario Alumno
              gid:
                  1001
```

7. Comprobando la creación del usuario y grupo alumno

```
docker exec minion-1 bash -lc 'cat /etc/passwd | grep alumno'
docker exec minion-1 bash -lc 'cat /etc/group | grep alumno'
```

```
alumno@alumno-virtualbox:~/Desktop/UD05-Caso01$ docker exec minion-1 bash -lc 'cat /etc/passwd | grep alumno'
alumno:x:1001:1001:Usuario Alumno:/home/alumno:/bin/bash
alumno@alumno-virtualbox:~/Desktop/UD05-Caso01$ docker exec minion-1 bash -lc 'cat /etc/group | grep alumno'
sudo:x:27:ubuntu,alumno
alumno:x:1001:
alumno@alumno-virtualbox:~/Desktop/UD05-Caso01$
```

Esto confirma que el usuario ha sido creado correctamente.

8. Acceso desde el host (servicio NGINX)

Cada minion tiene NGINX escuchando internamente en el puerto 8088, pero Terraform lo ha publicado externamente como puertos consecutivos a partir de la variable nginx_base_port.

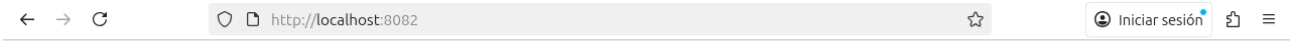
Accediendo con la URL http://localhost:8080

Accediendo con la URL http://localhost:8081

¡Hola desde minion-2!

Servidor desplegado automáticamente con Terraform + Salt Project 🧂

Accediendo con la URL http://localhost:8082



¡Hola desde minion-3!

Servidor desplegado automáticamente con Terraform + Salt Project 🧂

9. Destruir la infraestructura

```
terraform destroy -auto-approve
```

10. Buenas prácticas

- ❕ Desactiva auto_accept en producción.
- 📖 Usa nombres predecibles para targeting (minion-*).
- 📖 Amplía el ejercicio añadiendo roles diferentes (web/db).
- 📖 Todo el laboratorio funciona localmente, sin conexión externa.

📄 7. Detalle del contenido de los ficheros

📄 main.tf

```
##############################################
# 🧩 Proyecto: Laboratorio Terraform + Docker + Salt
# Despliega automáticamente un Salt Master y N Minions
# usando imágenes personalizadas basadas en Ubuntu 24.04
##############################################

# ============================================
# 🔧 1️⃣ Definición del bloque Terraform
# ============================================
terraform {
  required_providers {
    docker = {
      # Fuente oficial del provider Docker para Terraform
      source  = "kreuzwerker/docker"

      # Versión recomendada (mayor o igual a 3.0 pero menor que 4.0)
```

```
      version = "~> 3.0"
    }
  }
}

# ============================================
# 🐳2️⃣ Proveedor Docker
# ============================================
# Permite a Terraform comunicarse con el daemon de Docker
provider "docker" {}

# ============================================
# 🕸️3️⃣ Creación de red interna
# ============================================
# Define una red Docker llamada "salt-net"
# donde se conectarán tanto el master como los minions.
resource "docker_network" "salt_net" {
  name = "salt-net"
}

# ============================================
# 🏗️4️⃣ Construcción de la imagen Salt Master
# ============================================
# Terraform construirá la imagen personalizada del master
# usando el Dockerfile que se encuentra en la carpeta ./salt-master
resource "docker_image" "salt_master_image" {
  name = "salt-master:ubuntu24"  # Nombre/tag local de la imagen

  build {
    context    = "${path.module}/salt-master"           # Ruta del contexto de build
    dockerfile = "${path.module}/salt-master/Dockerfile" # Fichero Dockerfile a usar
  }
}

# ============================================
# 🏗️5️⃣ Construcción de la imagen Salt Minion
# ============================================
# Igual que el master, pero con el Dockerfile de los minions
resource "docker_image" "salt_minion_image" {
  name = "salt-minion:ubuntu24"

  build {
    context    = "${path.module}/salt-minion"
    dockerfile = "${path.module}/salt-minion/Dockerfile"
  }
}

# ============================================
# 🧠6️⃣ Creación del contenedor Salt Master
# ============================================
resource "docker_container" "salt_master" {
  name     = "salt-master"                          # Nombre del contenedor
  image    = docker_image.salt_master_image.image_id  # Imagen base (la que hemos
```

```
construido)
  hostname = "salt-master"                              # Nombre de host visible en la
red Docker

  # 🔗 Conectamos el contenedor a la red salt-net
  networks_advanced {
    name = docker_network.salt_net.name
  }

  # 🔓 Puertos internos/externos del master (para comunicación con los minions)
  ports {
    internal = 4505   # Publisher port
    external = 4505
  }

  ports {
    internal = 4506   # Returner port
    external = 4506
  }

  # 📂 Montamos el directorio local "salt-states" en /srv/salt
  # Esto permite al master acceder a los ficheros .sls y plantillas.
  mounts {
    type   = "bind"                                     # Tipo de montaje: bind mount
    source = abspath("${path.module}/salt-states")      # Ruta absoluta al directorio
local
    target = "/srv/salt"                                # Ruta dentro del contenedor
  }

  # 🧾 Comando que ejecutará el master al iniciar
  # Modo verbose (-l info) para ver logs en consola
  command = ["salt-master", "-l", "info"]
}

# ============================================
# 🤖7️⃣ Creación de múltiples Minions dinámicos
# ============================================

# 🔢 Definimos una lista de índices para generar los minions dinámicamente.
# Ejemplo: si var.minion_count = 3 → ["minion-1", "minion-2", "minion-3"]
locals {
  minion_indexes = toset([for i in range(var.minion_count) : i])

  # Creamos un mapa { "minion-1" = 0, "minion-2" = 1, ... }
  minions_map    = { for i in local.minion_indexes : format("minion-%d", i + 1) => i
}
}

# ============================================
# 🚀8️⃣ Despliegue de los contenedores Minion
# ============================================
resource "docker_container" "salt_minions" {
  for_each = local.minions_map   # Crea un contenedor por cada elemento del mapa
```

```
  name     = each.key            # Nombre del contenedor (minion-1, minion-2, ...)
  image    = docker_image.salt_minion_image.image_id
  hostname = each.key            # Nombre de host visible en la red Docker

  # 🔗 Conectamos el contenedor a la misma red que el master
  networks_advanced {
    name = docker_network.salt_net.name
  }

  # 🌐 Mapeo de puertos para NGINX
  # Cada minion escucha internamente en 8088, pero se le asigna un puerto diferente
en el host
  # Ejemplo: minion-1 → 8080, minion-2 → 8081, etc.
  ports {
    internal = 8088
    external = var.nginx_base_port + each.value
  }

  # 🌍 Variable de entorno que indica a qué master debe conectarse
  # Terraform sustituye "SALT_MASTER=salt-master"
  env = ["SALT_MASTER=salt-master"]

  # ⚙️ Dependencia: los minions solo se lanzan cuando el master ya está creado
  depends_on = [docker_container.salt_master]
}
```

📄 salt-master/Dockerfile

```
FROM ubuntu:24.04

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get install -y curl gnupg2 lsb-release ca-certificates netcat-openbsd && \
    \
    # Ensure keyrings dir exists
    mkdir -p /etc/apt/keyrings && \
    \
    # Download public key (Broadcom SaltProject)
    curl -fsSL
https://packages.broadcom.com/artifactory/api/security/keypair/SaltProjectKey/public
\
      -o /etc/apt/keyrings/salt-archive-keyring.pgp && \
    \
    # Create APT repository configuration
    mkdir -p /etc/apt/sources.list.d && \
    curl -fsSL
https://github.com/saltstack/salt-install-guide/releases/latest/download/salt.sources
\
      -o /etc/apt/sources.list.d/salt.sources && \
    \
```

```
    # Install Salt Master
    apt-get update && \
    apt-get install -y salt-master && \
    apt-get clean && rm -rf /var/lib/apt/lists/*

# Copiar configuración del master
COPY master.conf /etc/salt/master

EXPOSE 4505 4506

# ------------------------------------------------
# HEALTHCHECK: Verifica que el proceso salt-master
# está escuchando en el puerto 4505 (publisher).
# Cada 5 segundos Docker ejecuta "nc -z localhost 4505".
# Si el puerto no responde en 3s durante 10 intentos,
# el contenedor se marca como "unhealthy".
HEALTHCHECK --interval=5s --timeout=3s --retries=10 CMD nc -z localhost 4505 || exit
1

CMD ["salt-master", "-l", "info"]
```

salt-master/master.conf

```
# Acepta automáticamente nuevas claves de minions (no recomendado en producción)
auto_accept: True

# Define que se usarán archivos locales como backend de archivos
fileserver_backend:
  - roots

# Rutas base donde se buscarán los estados de Salt
file_roots:
  base:
    - /srv/salt
```

salt-minion/Dockerfile

```
FROM ubuntu:24.04

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get install -y curl gnupg2 lsb-release ca-certificates netcat-openbsd && \
    \
    # Ensure keyrings dir exists
    mkdir -p /etc/apt/keyrings && \
    \
    # Download public key (Broadcom SaltProject)
    curl -fsSL
https://packages.broadcom.com/artifactory/api/security/keypair/SaltProjectKey/public
\
      -o /etc/apt/keyrings/salt-archive-keyring.pgp && \
```

```
    \
    # Create APT repository configuration
    mkdir -p /etc/apt/sources.list.d && \
    curl -fsSL
https://github.com/saltstack/salt-install-guide/releases/latest/download/salt.sources
\
      -o /etc/apt/sources.list.d/salt.sources && \
    \
    # Install Salt Minion + nginx + ufw
    apt-get update && \
    apt-get install -y salt-minion nginx ufw && \
    apt-get clean && rm -rf /var/lib/apt/lists/*

# Configuración base
COPY minion.conf /etc/salt/minion

EXPOSE 8088

CMD ["bash", "-c", "sleep 8 && printf \"master: %s\\nid: %s\\n\" \"$SALT_MASTER\"
\"$HOSTNAME\" > /etc/salt/minion && salt-minion -l info"]
```

📄 salt-minion/minion.conf

```
# Dirección del master (sustituido por variable de entorno SALT_MASTER en tiempo de
ejecución)
master: ${SALT_MASTER}

# ID del minion: auto (puedes personalizar si quieres diferenciarlos)
id: auto
```

📄 salt-states/top.sls

```
base:
  'minion-*':
    - users
    - nginx
```

📄 salt-states/users.sls

```
# ========================================
# 📄 salt-states/users.sls
# ========================================

#1️⃣ Crear el grupo y el usuario
alumno-user:
  user.present:
    - name: alumno
    - fullname: "Usuario Alumno"
    - shell: /bin/bash
    - home: /home/alumno
    - createhome: True
    - groups:
      - sudo
```

```
# 2️⃣ Asignar contraseña usando chpasswd
set-password-alumno:
  cmd.run:
    - name: "echo 'alumno:Alumno123!' | chpasswd"
    - unless: "grep -q '^alumno:' /etc/shadow && getent shadow alumno | cut -d: -f2 | grep -qv '!'"
    - require:
      - user: alumno-user
```

📄 salt-states/ningx/config.sls

```
server {
    listen {{ listen_port }};
    server_name _;
    root /var/www/html;
    index index.html;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

📄 salt-states/ningx/init.sls

```
{% set port = 8088 %}

nginx-package:
  pkg.installed:
    - name: nginx

nginx-root-dir:
  file.directory:
    - name: /var/www/html
    - user: root
    - group: root
    - mode: 755

nginx-index-file:
  file.managed:
    - name: /var/www/html/index.html
    - source: salt://nginx/index.html.j2
    - template: jinja
    - context:
        minion_id: {{ grains['id'] }}
    - user: root
    - group: root
    - mode: 644

nginx-conf:
  file.managed:
    - name: /etc/nginx/sites-available/custom_site
    - source: salt://nginx/config.sls
    - template: jinja
    - context:
```

```
        listen_port: {{ port }}

nginx-enable:
  file.symlink:
    - name: /etc/nginx/sites-enabled/custom_site
    - target: /etc/nginx/sites-available/custom_site

nginx-service:
  service.running:
    - name: nginx
    - enable: True
    - watch:
      - file: nginx-conf
      - file: nginx-index-file
```

salt-states/ningx/index.html.j2

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Bienvenido - {{ minion_id }}</title>
</head>
<body style="font-family:sans-serif;text-align:center;margin-top:10%">
    <h1>¡Hola desde {{ minion_id }}!</h1>
    <p>Servidor desplegado automáticamente con Terraform + Salt Project 🧂</p>
</body>
</html>
```