RA5.2

IaC : Infrastructure as Code

Ciberseguridad en entornos de las tecnologías de la información

Puesta en producción segura

Índice

1. Introducción

1.1. IaC

La **Infraestructura como Código (IaC, por sus siglas en inglés: Infrastructure as Code)** es un enfoque moderno para la gestión de la infraestructura de TI que permite definir y aprovisionar entornos tecnológicos mediante código en lugar de procesos manuales o herramientas interactivas. IaC consiste en escribir archivos de configuración que contienen instrucciones detalladas sobre cómo debe crearse, configurar y gestionar la infraestructura (como servidores, redes, bases de datos, etc.). Este código se almacena en archivos de texto plano y puede versionarse, revisarse y desplegarse automáticamente, igual que el software tradicional.

Elimina errores humanos al automatizar el aprovisionamiento de recursos. Permite llevar un control histórico de los cambios realizados en la infraestructura. Facilita la creación de entornos repetibles y escalables. Acelera la creación de entornos de desarrollo, prueba y producción. El código puede revisarse, auditarse y compartirse fácilmente dentro de un equipo.

1.2. Terraform

Terraform es una herramienta de **Infraestructura como Código (IaC)** desarrollada por **HashiCorp**, que permite definir, aprovisionar y administrar infraestructura de manera segura, predecible y automatizada en múltiples proveedores de nube, como AWS, Azure, Google Cloud, entre otros. Terraform permite describir toda tu infraestructura (servidores, bases de datos, redes, balanceadores de carga, etc.) como **código declarativo** usando el lenguaje **HCL (HashiCorp Configuration Language)**. Luego, este código se ejecuta para crear y mantener esa infraestructura de forma automática.

- Ventajas de usar Terraform

Proveedor-agnóstico: Funciona con muchas nubes (AWS, Azure, GCP, Kubernetes, etc.).

Declarativo: Solo defines *qué* quieres; Terraform se encarga del *cómo*.

Idempotencia: Aplica los cambios solo si hay diferencias entre el estado actual y el deseado.

Planificación de cambios: Con `terraform plan` puedes ver qué hará antes de aplicarlo.

Modularidad y reutilización: Puedes organizar tu infraestructura en módulos reutilizables.

- Flujo de trabajo básico en Terraform

1. Escribir el archivo de configuración (`.tf`).
2. Ejecutar `terraform init` para inicializar el proyecto.
3. Ejecutar `terraform plan` para revisar lo que se va a hacer.
4. Ejecutar `terraform apply` para aplicar los cambios.
5. Usar terraform destroy si quieres eliminar la infraestructura.

Ejemplo: Este código lanza una instancia EC2 de AWS usando una AMI específica.

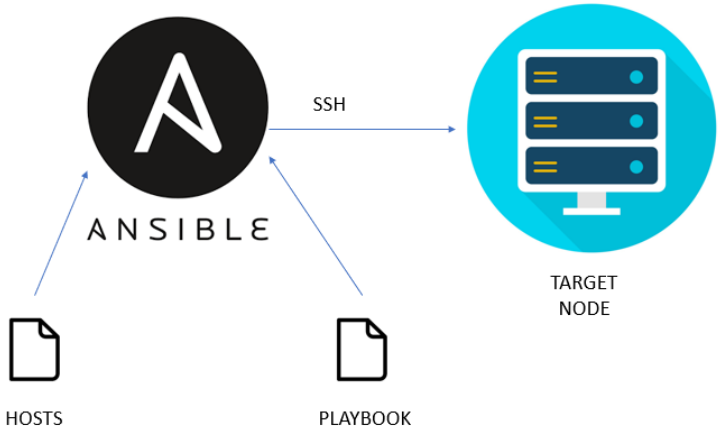
hcl Copia Modifica

```hcl
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "mi_servidor" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
}
```

1.3. Ansible

Ansible es una herramienta de **automatización de TI** que permite configurar sistemas, desplegar software y coordinar tareas complejas de administración, todo de forma sencilla y repetible. Fue desarrollada por Red Hat y se basa en el concepto de "Infraestructura como Código" (IaC). Ansible automatiza la configuración y gestión de servidores, redes y aplicaciones. A diferencia de herramientas como Terraform, que aprovisionan infraestructura, Ansible se enfoca en **configurar y gestionar** sistemas ya existentes (aunque puede hacer algunas tareas de provisión también).



- Ventajas de usar Terraform

Sin agentes: No requiere instalar software en los nodos; se conecta vía **SSH**.

Fácil de aprender: Usa **YAML** para definir tareas en archivos llamados **playbooks**.

Idempotente: Ejecutar el mismo playbook varias veces no cambia nada si ya está configurado correctamente.

Extensible: Compatible con miles de módulos para gestionar paquetes, servicios, usuarios, archivos, etc.

Integración sencilla: Se puede usar con nubes, contenedores, Kubernetes y CI/CD.

- Componentes

Playbook: Archivo YAML que define qué tareas ejecutar y en qué orden.

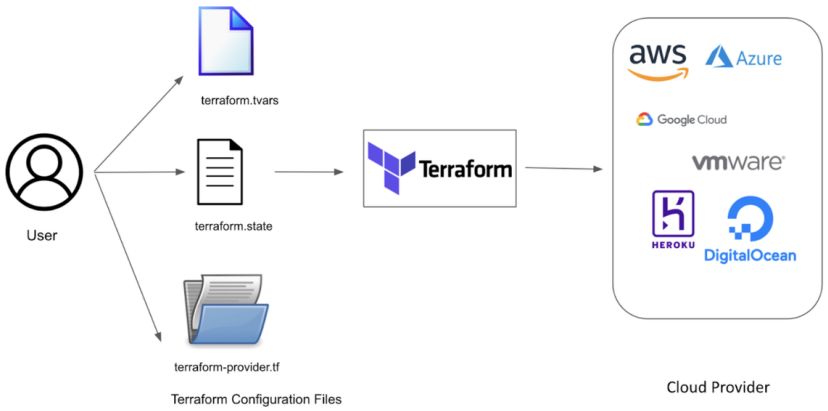
Inventario: Lista de hosts gestionados (puede ser un archivo, script o dinámico desde la nube).

Roles: Estructura reutilizable para organizar configuraciones.

Módulos: Funciones reutilizables que realizan tareas específicas (como instalar un paquete, copiar un archivo, etc.).

- Flujo de trabajo básico en Terraform

6. Crear inventario (`hosts.ini`).
7. Escribir playbooks.
8. Ejecutar con `ansible-playbook servidor.yml`.



Ejemplo: Este código configura el servidor web Apache.

yaml Copia Modifica

```yaml
- name: Configurar servidor web
  hosts: webservers
  become: yes
  tasks:
    - name: Instalar Apache
      apt:
        name: apache2
        state: present
    - name: Asegurar que Apache esté iniciado
      service:
        name: apache2
        state: started
        enabled: yes
```

2. Objetivos

Esta unidad didáctica contribuye a trabajar los siguientes criterios de evaluación:

RA5	Implanta sistemas seguros de despliegue de software, utilizando herramientas para la automatización de la construcción de sus elementos.
CA.	1,66 %
CA.E	1,66 %
CA.F	1,66 %

3. Actividades

La entrega se realizará mediante la cuenta de Github, se tendrá que definir la estructura correcta y contener los archivos necesarios para poder ejecutar los ficheros terraform y ansible. Al mismo tiempo hay que incluir un README con una captura de pantalla de la provisión, configuración y validación.

3.1. Provisionar una máquina virtual Ubuntu 24.04 en Virtualbox mediante Terraform.

3.2. Configurar una máquina virtual Ubuntu 24.04 en Virtualbox mediante Ansible.

- Realizar update & upgrade del sistema de forma automática.
- Instalar el servicio apache.

3.3. Configurar una máquina virtual Ubuntu 24.04 en Virtualbox mediante Ansible.

- Crear un index.html con el contenido: ‘Ansible rocks’ en el directorio del servidor web para poder ser mostrado y reiniciar el servicio.
- Realizar un curl al servidor web y verificar el mensaje ‘Ansible rocks’.

4. Bibliografía

Community, A. (s. f.). *Ansible documentation*. https://docs.ansible.com/

Terraform overview | Terraform | HashiCorp Developer. (s. f.). Terraform Overview | Terraform | HashiCorp Developer. https://developer.hashicorp.com/terraform/docs

ACTIVIDAD	60%	80%	90%	100%
3.1	Funcionamiento correcto	Análisis técnico	Pipeline	Extra
3.2	Funcionamiento correcto	Análisis técnico	Pipeline	Extra
3.3	Funcionamiento correcto	Análisis técnico	Pipeline	Extra