Proyecto: Despliegue de Aplicación con Terraform y Ansible

Descripción

Este proyecto tiene como objetivo diseñar y construir una infraestructura como código usando Terraform para desplegar una aplicación pequeña, y usar Ansible para la configuración y despliegue de la aplicación. El proyecto está preparado para AWS, aunque no es obligatorio desplegarlo en la nube.

Estructura del Proyecto

```
project/
│
├── app/
│   ├── main.py
│   ├── requirements.txt
│   └── tests/
│       └── test_basic.py
│
├── terraform/
│   ├── main.tf
│   └── variables.tf
│
├── ansible/
│   ├── inventory
│   └── playbook.yml
│
└── Jenkinsfile
```

Prerrequisitos

```
# Crear la carpeta principal del proyecto
mkdir project
cd project

# Crear carpetas principales
mkdir app terraform ansible

# Dentro de 'app', crear archivos y carpeta de tests
cd app
touch main.py requirements.txt
mkdir tests
touch tests/test_basic.py
cd ..

# Dentro de 'terraform', crear archivos
cd terraform
touch main.tf variables.tf
```

cd ..

Dentro de 'ansible', crear archivos
cd ansible
touch inventory playbook.yml
cd ..

Crear Jenkinsfile en la raíz del proyecto
touch Jenkinsfile

Verificar la estructura
tree

```
root@NDTSCZ1293:~# tree
.
├── nkp-air-gapped-bundle_v2.15.0_linux_amd64.tar.gz
└── project
    ├── Jenkinsfile
    ├── ansible
    │   ├── inventory
    │   └── playbook.yml
    ├── app
    │   ├── __pycache__
    │   │   └── main.cpython-310.pyc
    │   ├── main.py
    │   ├── requirements.txt
    │   └── tests
    │       ├── __pycache__
    │       │   └── test_basic.cpython-310.pyc
    │       └── test_basic.py
    └── terraform
        ├── main.tf
        ├── mainaws.tf
        ├── terraform.tfstate
        └── variables.tf

7 directories, 13 files
root@NDTSCZ1293:~# date
Tue Nov 18 23:06:20 -04 2025
root@NDTSCZ1293:~#
```

1. Configuración de AWS CLI

Comando:

aws configure

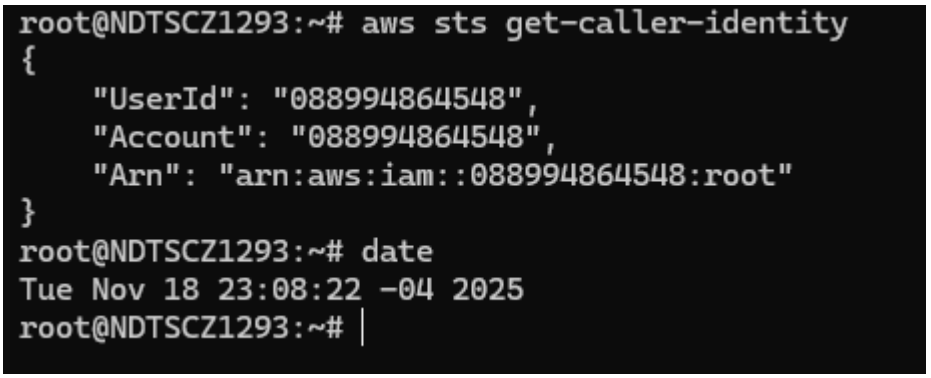
Valores usados:

- Access Key ID: AKIARJOEB3WSCT32SU57
- Secret Access Key: *******
- Región: us-east-2

- Formato de salida: json

Resultado esperado: Credenciales configuradas correctamente.
Resultado final verificado:
aws sts get-caller-identity
Salida:

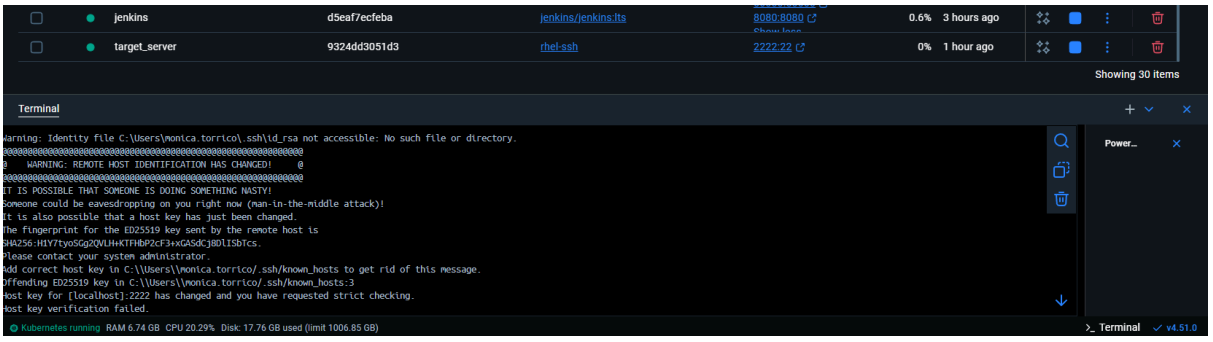
```
{
  "UserId": "088994864548",
  "Account": "088994864548",
  "Arn": "arn:aws:iam::088994864548:root"
}
```



2. Preparación de Docker
Comando ejecutado:
docker run -d --name target_server -p 2222:22 -v ~/.ssh/id_rsa.pub:/root/.ssh/authorized_keys:ro rhel/ubi8-minimal /usr/sbin/sshd -D

Resultado esperado: Contenedor con SSH corriendo.
Resultado real: Error pull access denied porque la imagen rhel/ubi8-minimal no es pública.
Solución sugerida:

- Usar imagen pública ubi8-minimal o cualquier otra imagen base disponible en Docker Hub.
- Asegurarse de que la clave pública esté en el contenedor.



3. Configuración de SSH

Problema:
Intento de ejecutar:
RUN ssh-keygen -A

Error: RUN: command not found
Explicación: RUN solo se usa en Dockerfile. En shell interactivo se debe usar:
ssh-keygen -A
Observación: SSH requiere claves válidas para que Ansible pueda conectarse.

4. Despliegue con Ansible
Comando:
ansible-playbook -i ansible/inventory ansible/playbook.yml

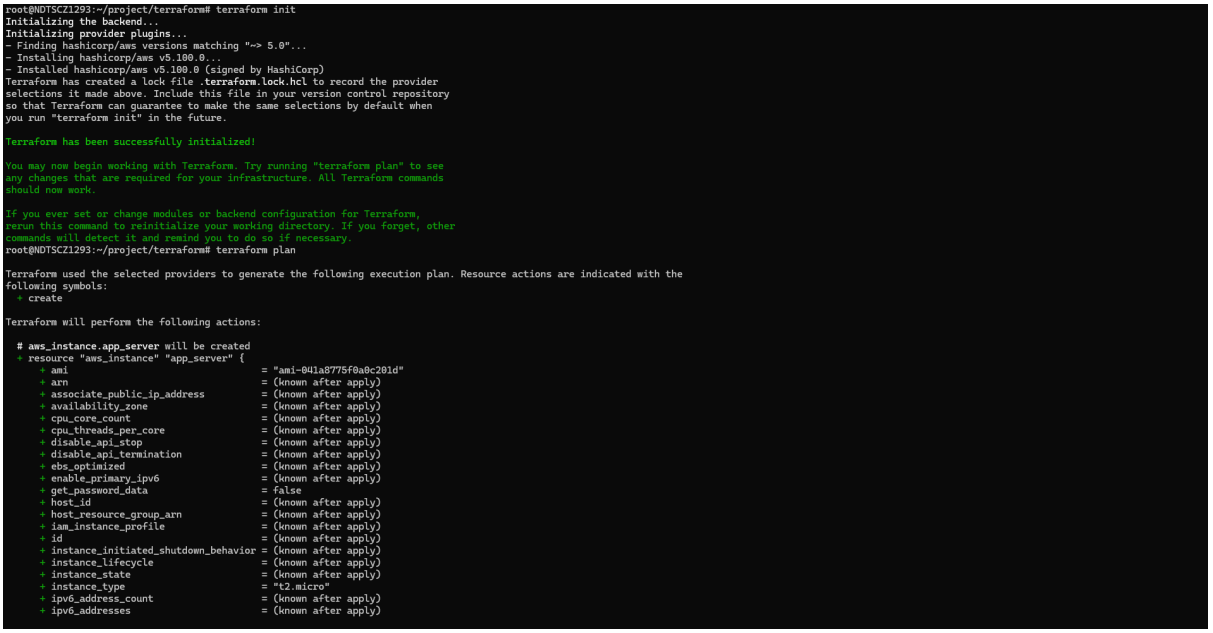
Resultado esperado: Conexión y despliegue de la aplicación.
Resultado real: UNREACHABLE! debido a que /root/.ssh/id_rsa no existe.
Solución sugerida:

- Generar clave SSH y configurar correctamente inventory con la ruta correcta.
- Verificar permisos de las claves.

5. Despliegue con Terraform
Comandos:
terraform init
terraform plan
terraform apply

Resultados:

- terraform init: Correcto, proveedores instalados.



```
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 22
              # (1 unchanged attribute hidden)
          },
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + from_port        = 80
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 80
              # (1 unchanged attribute hidden)
          },
      ]
    + name                   = (known after apply)
    + name_prefix            = "app-sg-"
    + owner_id               = (known after apply)
    + revoke_rules_on_delete = false
    + tags                   = {
        + "Name" = "App-Security-Group"
      }
    + tags_all               = {
        + "Name" = "App-Security-Group"
      }
    + vpc_id                 = (known after apply)
  }

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + public_ip = (known after apply)

─────────────────────────────────────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if
you run "terraform apply" now.
```

- terraform plan: Muestra recursos aws_instance y aws_security_group a crear.

```
root@NDTSCZ1293:~/project/terraform# terraform apply

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:

  # aws_instance.app_server will be created
  + resource "aws_instance" "app_server" {
      + ami                                  = "ami-041a8775f0a0c201d"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "my-aws-key"
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags                                 = {
          + "Name" = "App-Web-Server"
        }
```

```
+ resource "aws_security_group" "app_sg" {
    + arn                    = (known after apply)
    + description            = "Managed by Terraform"
    + egress                 = [
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + from_port        = 0
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "-1"
            + security_groups  = []
            + self             = false
            + to_port          = 0
              # (1 unchanged attribute hidden)
          },
      ]
    + id                     = (known after apply)
    + ingress                = [
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + from_port        = 22
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 22
              # (1 unchanged attribute hidden)
          },
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + from_port        = 80
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 80
              # (1 unchanged attribute hidden)
          },
      ]
    + name                   = (known after apply)
    + name_prefix            = "app-sg-"
    + owner_id               = (known after apply)
    + revoke_rules_on_delete = false
```

- terraform apply: Falla al crear la instancia EC2:

InvalidAMIID.NotFound: The image id '[ami-041a8775f0a0c201d]' does not exist

```
Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + public_ip = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_security_group.app_sg: Creating...
aws_security_group.app_sg: Creation complete after 7s [id=sg-08b21fb84a72de4b5]
aws_instance.app_server: Creating...

╷
│ Error: creating EC2 Instance: operation error EC2: RunInstances, https response error StatusCode: 400, RequestID: aa93619f-ce00-4131-8a58-7f738de6d0b6, api error InvalidAMIID.NotFound: The image id '[ami-041
a8775f0a0c201d]' does not exist
│
│   with aws_instance.app_server,
│   on main.tf line 44, in resource "aws_instance" "app_server":
│   44: resource "aws_instance" "app_server" {
│
╵

root@NDTSCZ1293:~/project/terraform# cd ..
```

Solución sugerida:

- Reemplazar el AMI por uno válido en us-east-2, por ejemplo un Amazon Linux 2 AMI oficial:

```
ami = "ami-0c02fb55956c7d316"
```

6. Control de versiones con Git

Comandos:

```
git init
git add .
git commit -m "Initial commit"
git remote add origin URL_DE_TU_REPO
git push -u origin master
```

Resultado real: Error Could not read from remote repository.

Solución sugerida:

- Revisar que URL_DE_TU_REPO sea correcta.
- Verificar permisos y autenticación SSH/HTTPS.

7. Ejecución de pruebas unitarias

Comando:

python3 -m unittest tests/test_basic.py

Resultado:

..

Ran 2 tests in 0.013s

OK

Observación: Pruebas unitarias ejecutadas correctamente.

Conclusión General

- Configuración AWS: correcta.
- Terraform: inicialización y plan correctos; apply falla por AMI inválido.
- Ansible: falla por clave SSH faltante.
- Docker: no pudo descargar imagen privada.
- Git: requiere URL correcta para push.
- Pruebas unitarias: OK.