Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Software Avanzado
Laboratorio

Funcionamiento de Terraform

Feliciano Ernesto Franco Lux
200915532

Funcionamiento de Terraform

El repositorio contiene la mayoría de archivos de configuración por lo que se detallarán los archivos ya creados y se explicará la creación de los nuevos:

Estructura de archivos

- **Principal: main.tf**

  Este archivo contiene toda la infraestructura como código (IaS), en él se hará referencia a otros recursos como `outputs.tf`, `variables.tf` y demás. Los bloques de instrucciones que se pueden encontrar en el archivo son:

  - **Definir providers**: terraform/required_providers
  - **Configuración de provider**: provider "aws" {}
  - **Security group**: resource "aws_security_group" "nginx" {}
  - **Llaves de acceso**: resource "aws_key_pair" "access_key"{}
  - **EC2**: resource "aws_instance" "nginx" {}
  - **EC2**: resource "aws_instance" "ansible" {}

- **Variables: variables.tf**

  Definimos todas las variables que utilizaremos en el archivo principal, se define la descripción, el tipo de dato y si es sensible o no. Detalles de las variables utilizadas:

  - **access_key**: Llave de acceso del usuario asociado a las operaciones de Terraform, la llave fue generada en el sitio de AWS.
  - **secret_key**: Código de seguridad para el usuario asociado a las operaciones de Terraform, la llave fue generado junto a **access_key**.
  - **vpc_id**: Identificador de la VPC utilizada para asociar a los servicios EC2 que se crearán.
  - **subnet_id**: Identificador de la subred que se encuentra dentro de la VPC (vpc_id) para asociar a los servicios EC2 que se crearán.
  - **ssh_user**: Usuario SSH que se utilizará para conectarse a los servidores.
  - **private_key_path**: Ruta en la que se encuentra la llave privada que se utilizará para la comunicación entre servidores y equipo local.
  - **public_key_path**: Ruta en la que se encuentra la llave pública que se utilizará para la comunicación entre servidores y equipo local.

- **Valores de las variables: terraform.tfvars**

Este archivo no se encuentra disponible el repositorio por lo que se debe crear en la misma carpeta en que se encuentra el archivo **variables.tf**. La estructura del archivo podría ser la siguiente:

```
access_key       = "my-access_key"
secret_key       = "my-secret_key"
vpc_id           = "my-vpc_id"
subnet_id        = "my-subnet_id"
ssh_user         = "my-ssh_user"
private_key_path = "./path/my-private_key_path"
public_key_path  = "./path/my-public_key_path"
```

•Salidas: outputs.tf

Listado de variables que se imprimirán cuando el proceso `apply` haya finalizado, en este caso se imprimen los valores del identificador de la instancia EC2, IP privada e IP pública para los servidores de Ansible y Nginx.

```
output "ansible" {
  description = "Ansible EC2 instance info"
  value       = "[${aws_instance.ansible.id} -> public IP $
{aws_instance.ansible.public_ip} and private IP $
{aws_instance.ansible.private_ip}]"
}


output "nginx" {
  description = "Nginx EC2 instance info"
  value       = "[${aws_instance.nginx.id} -> public IP $
{aws_instance.nginx.public_ip} and private IP $
{aws_instance.nginx.private_ip}]"
}
```

•Llave pública(key_pair.pub) y privada(key_pair):

Estas llaves no se encuentran en el repositorio por lo que se debe generar un nuevo par de claves dentro de la carpeta `ssh`. Primero creamos la carpeta dentro de la carpeta de `terraform`:

```
$ mkdir ssh
```

Ingresamos a ella

```
$ cd ssh
```

Ejecutamos el comando para crear el nuevo par de claves SSH:

```
$ ssh-keygen -t rsa -b 4096
```

Cuando solicite el nombre y la ruta para el nuevo par de llaves, `Enter file in which to save the key`, colocaremos únicamente el nombre puesto que ya estamos dentro de la carpeta que recibirá las llaves:

```
key_pair
```

La estructura de archivos completa dentro de la carpeta terraform quedará de la siguiente forma:

```
.
├── terraform
    ├── main.tf
    ├── outputs.tf
    ├── ssh
        ├── key_pair
        └── key_pair.pub
    ├── terraform.tfvars
    └── variables.tf
```

Ejecución

Cuando los archivos de configuración se encuentran hechos, es importante pero no indispensable darle formato al contenido de cada uno. El siguiente comando es útil para dejar ordenadas las instrucciones y que sean legibles.

```
$ terraform fmt
```

Inicializamos el directorio de trabajo de terraform, esto provoca que Terraform genere un nuevo espacio de trabajo con las configuraciones previamente hechas.

```
$ terraform init
```

Ahora se puede ver un plan de ejecución de todos los recursos que se crearán:

```
$ terraform plan
```

Al final de la ejecución aparece un resumen de lo que se creará, modificará o eliminará. Si todo está bien *aplicamos* los cambios con el siguiente comando:

```
$ terraform apply
```

o también se puede utilizar

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

Si la ejecución finaliza sin problemas entonces veríamos al final el mensaje de ejecución exitosa con las salidas establecidos en el archivo **outputs.tf**

Finalizar ejecución

El siguiente comando le indica a terraform que debe destruir todos los recursos creados

```
$ terraform destroy
```

Al igual que en *apply*, esto permite ejecutar el comando con autorización

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