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**](https://developer.hashicorp.com/terraform/language/values/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