
Universidad
Tecmilenio®

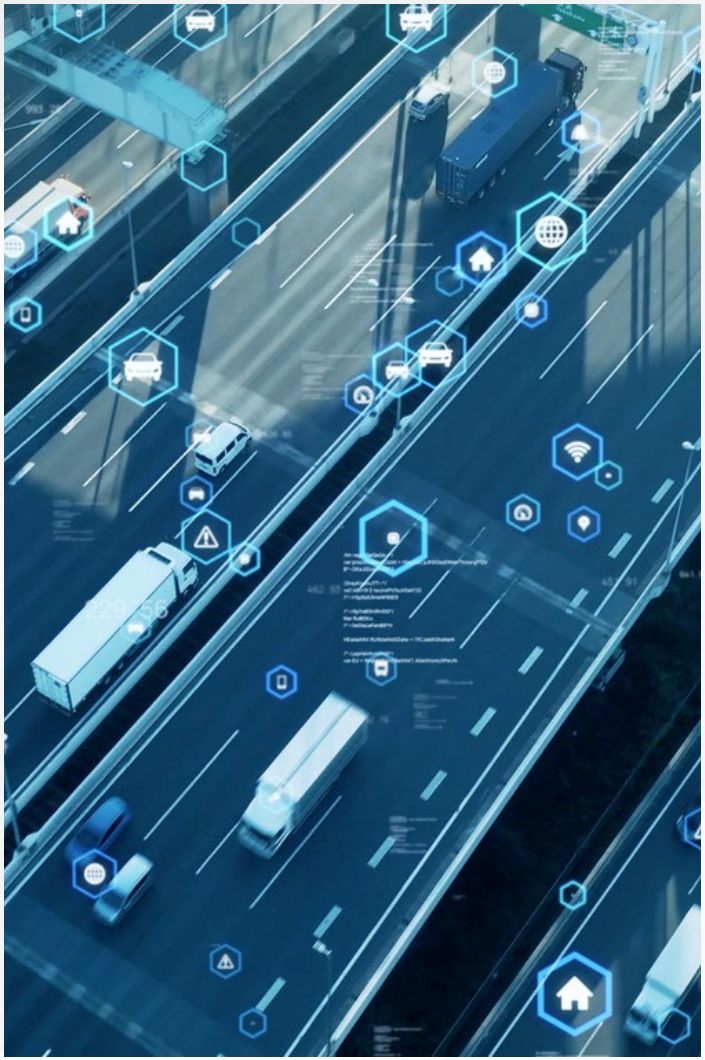
Universidad Tecmilenio

DevOps Avanzado (IaC y seguridad)

Conceptos básicos de Terraform

Semana 5

Introducción



En el mundo de TI se ha necesitado de ingenieros dedicados al aprovisionamiento y mantenimiento de la infraestructura necesaria para que los sistemas se puedan ejecutar de manera adecuada.

Como menciona Brikman (2019), se solía hacerse de forma manual y haciendo los ajustes necesarios hasta lograr la estabilidad del sistema. Estas salas de servidores se convertían en monumentos a los que estaba prohibido entrar por temor a no poder lograr la misma configuración dos veces

Ahora imagina que la empresa para la que estás trabajando logró lanzar un producto cuya popularidad aumentó de manera exponencial, por lo que necesitarán hacer lo mismo 10, 50, 100 o más veces.

Esa es una escala poco sana para un proceso manual, y ahí es donde se luce la infraestructura como código (IaC).

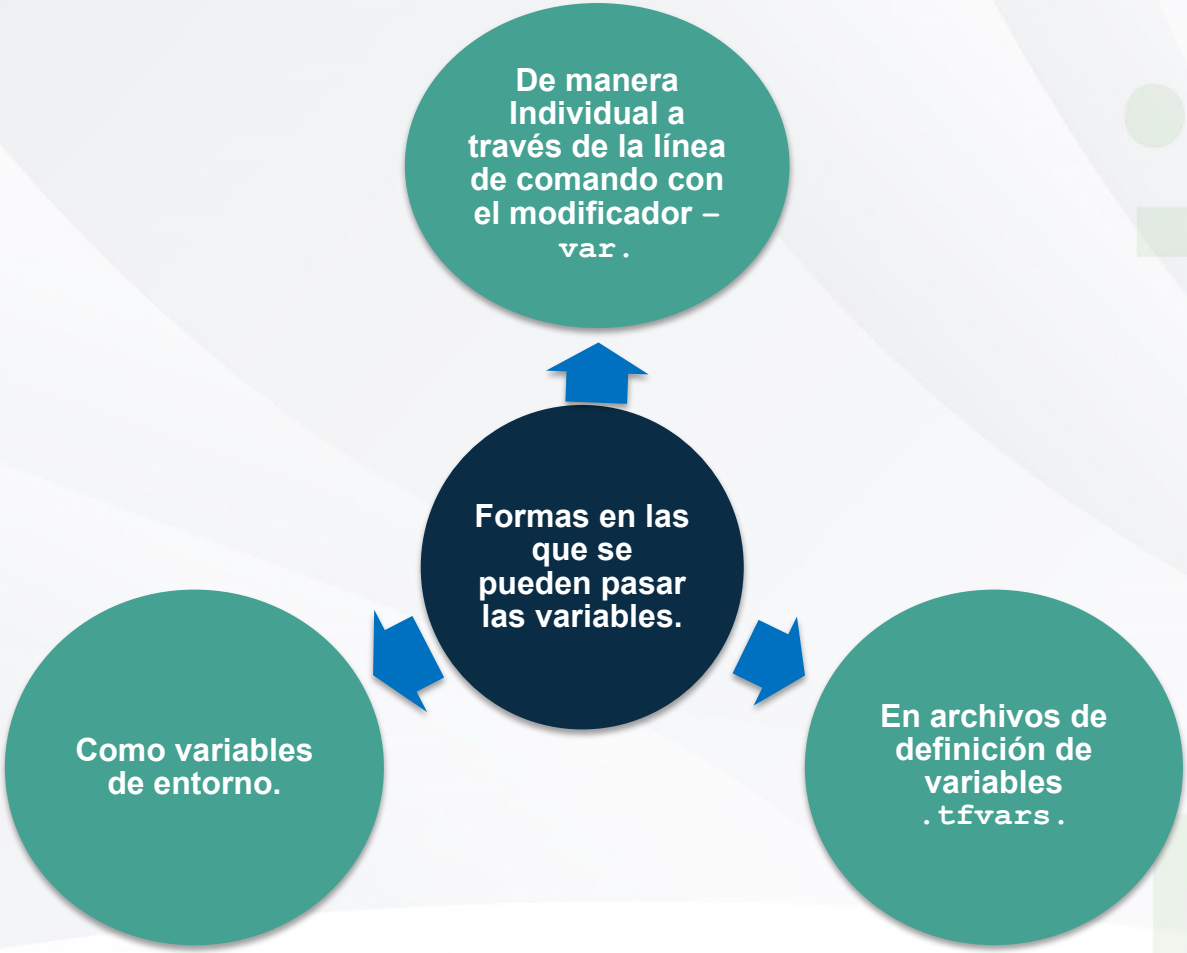
Explicación

¿Cómo funciona Terraform?

Concepto	Definición
Terraform	Herramienta de infraestructura que permite definir recursos en nubes informáticas y ambientes de servidores virtuales.
HCL	Lenguaje de configuración de HashiCorp usado para definir los recursos.
Acercamiento declarativo	En los archivos de configuración se define cómo será el estado final de la infraestructura.
Sintaxis	Los bloques son contenedores para la configuración de algún tipo de objeto, por ejemplo, un recurso. El bloque puede contener otros bloques embebidos que definen recursos propios del bloque principal. La mayoría de las características controladas por Terraform están contenidas en el bloque principal, en lugar de alguno de los bloques anidados. Los argumentos son una asignación directa de un valor a algún atributo del bloque. Las expresiones representan un valor literal, referenciado o calculando que vas a asignar a los argumentos.
Variables	Variables de entrada (input): son como los parámetros de una función. Variables de salida (output): sirven como valores de retorno de una función. Variables locales (local): son como las variables locales de una función.

Explicación

Funcionamiento de Terraform



Explicación

Proveedores

Explicación

Aprovisionadores

Usados para modelar acciones que se ejecutarán en una máquina local o remota.

No es recomendable usarlos, ya que contienen gran incertidumbre.

Se pueden utilizar para pasar información a las máquinas virtuales u otros recursos.

Se utilizan para ejecutar software de gestión de configuración.

Aprovisionador `file`: copia archivos desde la máquina donde estás corriendo Terraform y lo manda al recurso que estás creando.

Aprovisionador `local-exec`: ejecuta un comando en donde se está corriendo Terraform.

Aprovisionador remote-exec: ejecuta un comando en el recurso remoto que creaste.

Explicación

Ventajas de Terraform

Universidad Tecmilenio

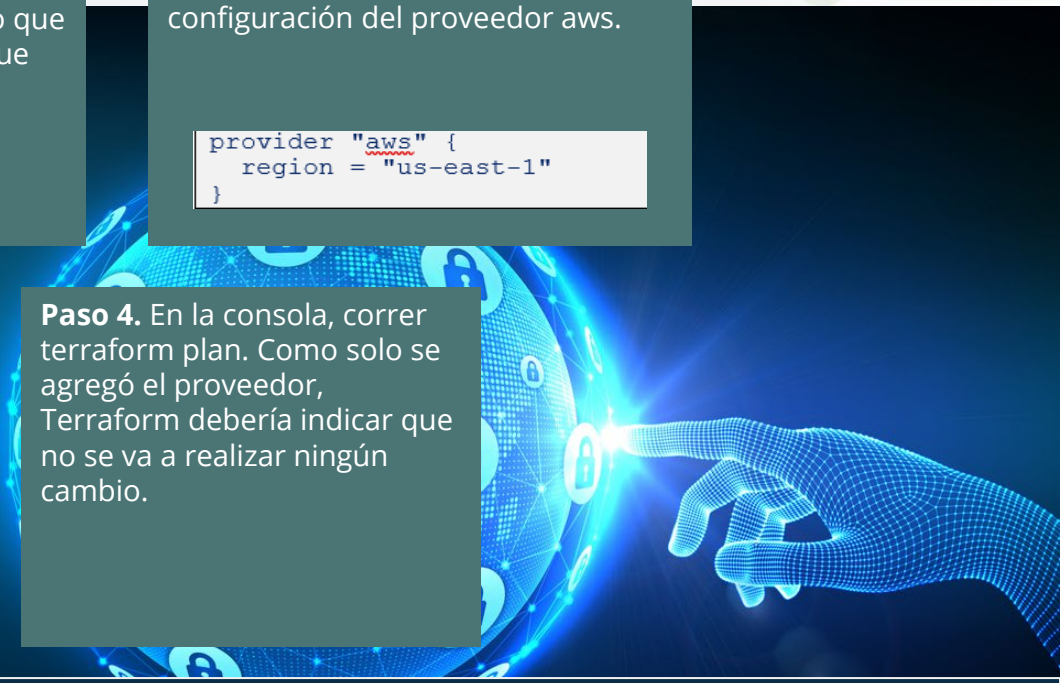
Ejercicio

Paso 1. Para confirmar que Terraform se encuentra instalado y funcionando, puedes correr en la línea de comandos la palabra terraform -version con lo que vas a ver la versión de Terraform que tienes instalada.

```
$ terraform -
version
Terraform v1.1.7
on darwin amd64
```

Paso 2. Hacer una carpeta llamada terraform que contenga un archivo llamado main.tf y agregar la configuración del proveedor aws.

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



Paso 3. En la consola, correr terraform init y revisar que Terraform se inicialice correctamente con el proveedor aws.

Paso 4. En la consola, correr terraform plan. Como solo se agregó el proveedor, Terraform debería indicar que no se va a realizar ningún cambio.

Paso 5. Crear el archivo desarrollo.auto.tfvars donde se agregarán las siguientes variables:

```
region          = "us-east-1"
id_imagen       = "ami-
000722651477bd39b"
tipo_instancia = "t2.micro"
```

Paso 6. Ajustar el archivo main.tf para que use las variables declaradas en desarrollo.auto.tfvars.

```
variable "region" {}
variable "id_imagen" {}
variable "tipo_instancia" {}

provider "aws" {
  region = var.region
}

resource "aws_instance" "foo" {
  ami           = var.id_imagen
  instance_type =
var.tipo_instancia
}
```

Paso 7. Volver a correr terraform plan para ver cómo se indican los cambios que se van a realizar para lograr aprovisionar una instancia de EC2 de AWS.

Bibliografía

•Brikman, Y. (2019). *Terraform: Up & Running* (2ª ed.). Estados Unidos: O'Reilly.

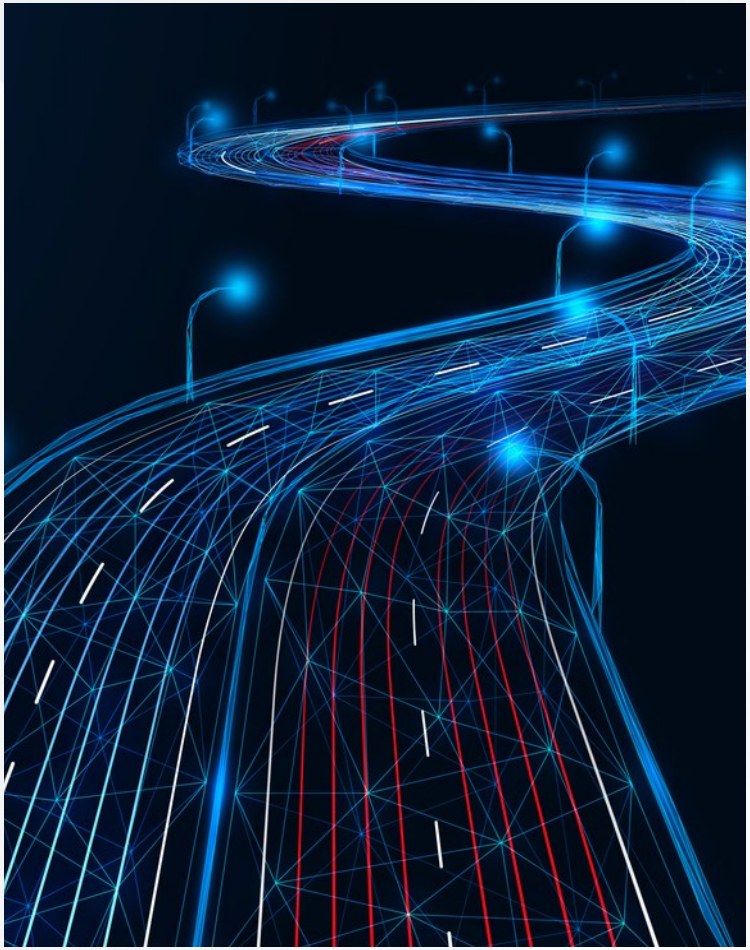
Universidad Tecmilenio

Infraestructura como código

Ciclo de vida de la infraestructura como código (IaC)

Semana 5

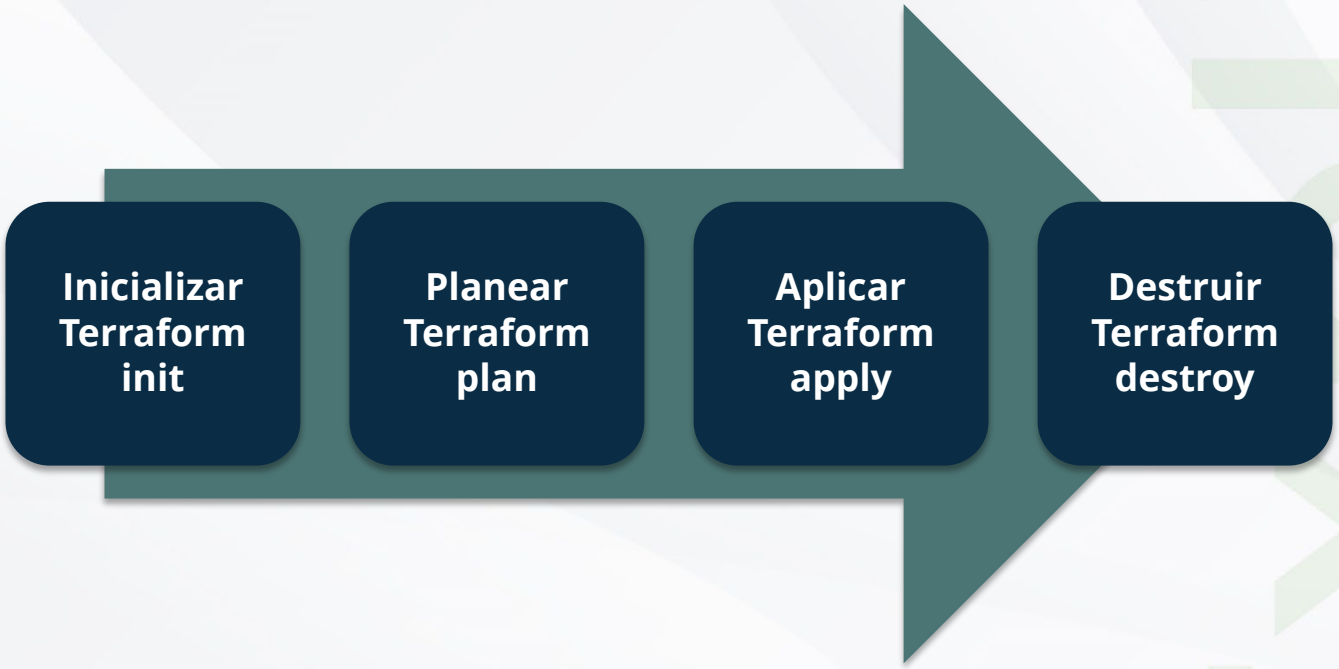
Introducción



Una de las grandes ventajas de Terraform es que, así como puedes configurar recursos para AWS, también puedes configurar recursos para GCP, Azure o alguna otra nube, lo cual te da la posibilidad de que, en una emergencia de este tipo u otra más común como un ataque interno o un error humano, puedas, ya sea levantar una infraestructura de respaldo en alguna otra nube o reaprovisionar los recursos que fueron borrados o mal configurados (Hashicorp, 2022).

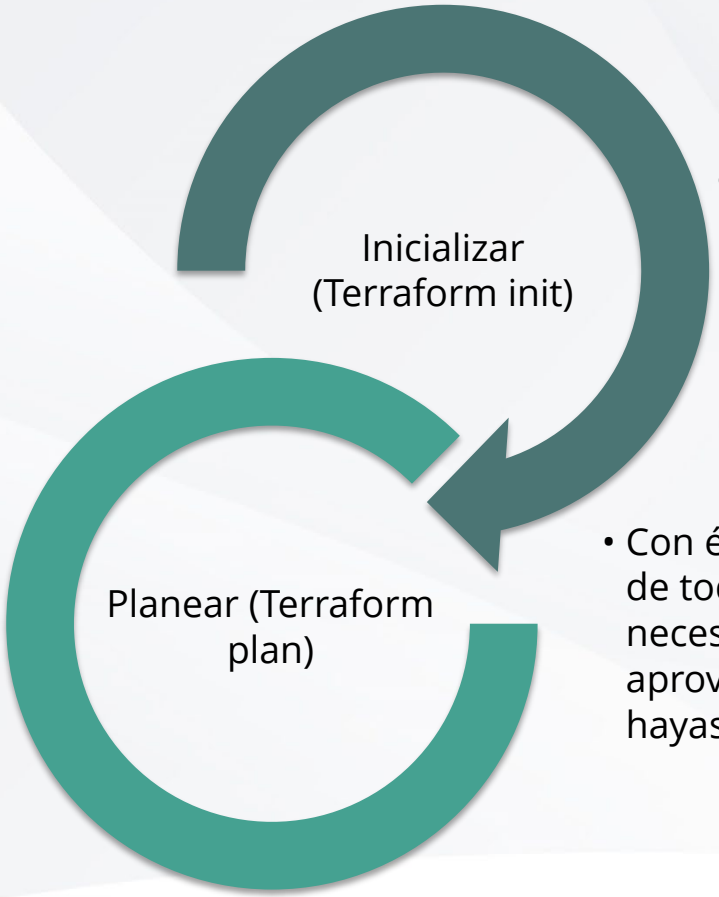
Explicación

Ciclo de vida de la infraestructura



Explicación

Comandos básicos de Terraform



- Correr terraform init con lo que Terraform procederá a conectarse a Internet y descargar las librerías y agregados.

- Con él podrás ver un listado de todas las operaciones que necesita realizar para aprovisionar los recursos que hayas configurado.

Explicación

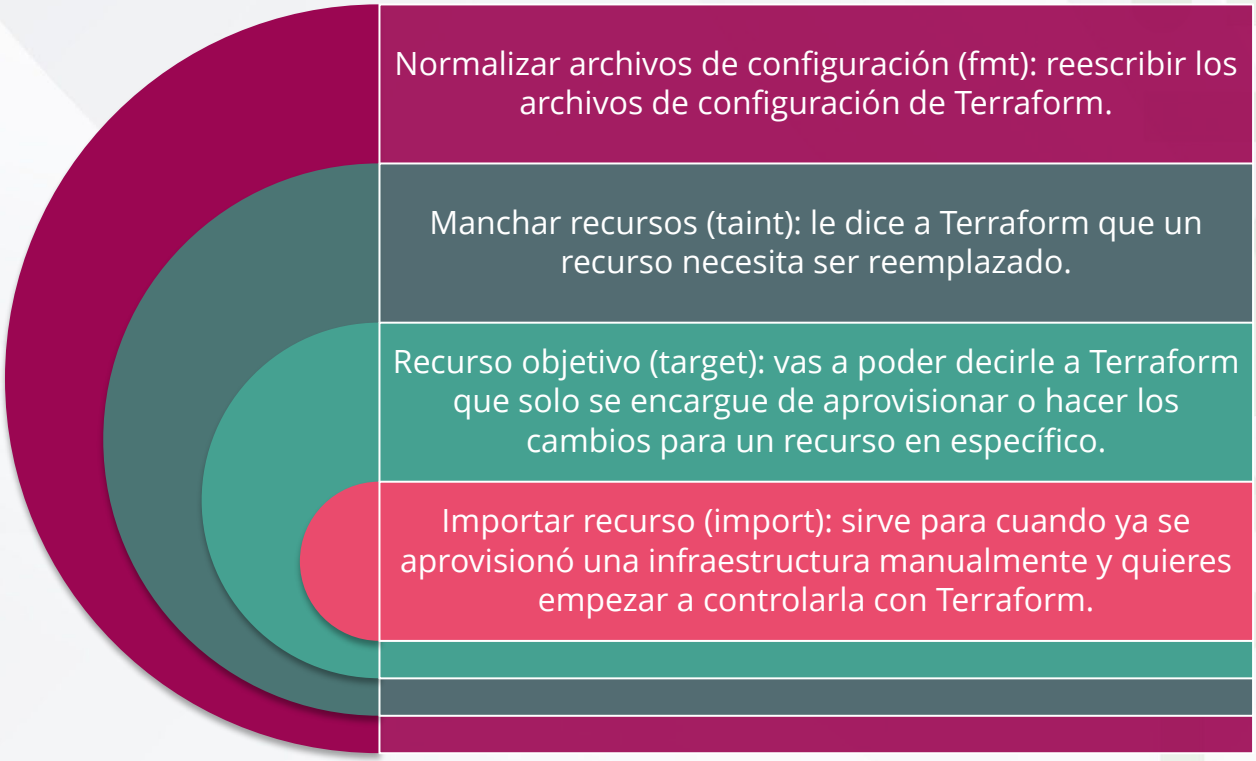
Comandos básicos de Terraform

Aplicar (Terraform apply)	• Se mostrará una vez más el plan que utilizará Terraform para aplicar la infraestructura de tal forma que tengas una segunda oportunidad de revisar lo que se realizará y te va a preguntar si estas de acuerdo con los cambios.
Destruir (Terraform destroy)	• Sirve para destruir los recursos que estén especificados en los archivos de configuración con las respectivas precauciones.

Universidad Tecmilenio

Explicación

Comandos básicos adicionales



Ejercicio

Paso 1.
Revisar que Terraform se encuentre instalado y funcionando, corriendo el comando terraform -version

$ terraform -version
Terraform v1.1.7

Paso 2.
Hacer una carpeta llamada terraform que contenga un archivo llamado main.tf y escribir la configuración para generar un bucket. Puedes encontrar un ejemplo en https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket#private-bucket-w-tags

El uso y descarga del software deberá apegarse a los términos y condiciones del sitio oficial del fabricante y su uso será responsabilidad de quien lo descargue. Tecmilenio no tiene licencia ni posee los derechos sobre dicho software.

Paso 3.
En la consola, correr terraform init y revisar que Terraform se inicialice correctamente con el proveedor aws.

Paso 4.
En la consola, correr terraform plan, usando el modificador target para ver solo los cambios que se van a realizar para generar el bucket.

Paso 5.
Ejecutar terraform init usando el modificador target para generar el bucket en AWS.

Paso 6.
Una vez creado el bucket:

- Ejecuta terraform plan, para ver que no haya ningún cambio en la infraestructura.
- Borra el archivo terraform.tfstate que contiene la relación entre la configuración y la infraestructura.
- Vuelve a ejecutar el comando terraform plan para ver qué te dice ahora.

Paso 7.
Vas a reconciliar la configuración con el bucket que acabas de crear. Para esto, vas a utilizar el comando terraform import, puedes revisar cómo usarlo en https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket#import

El uso y descarga del software deberá apegarse a los términos y condiciones del sitio oficial del fabricante y su uso será responsabilidad de quien lo descargue. Tecmilenio no tiene licencia ni posee los derechos sobre dicho software.

Paso 8.
Tras haber corrido el comando terraform import, vuelve a correr el comando terraform plan para ver que efectivamente Terraform te indica que no hay más cambios que hacer.

Paso 9.
Ejecuta terraform destroy para eliminar el bucket que ya no vas a utilizar. Recuerda que tener recursos que no necesitas en las nubes informáticas puede incurrir en costos no deseados.

Cierre

Como habrás notado, Terraform es una herramienta muy poderosa de comandos sencillos para el aprovisionamiento de infraestructuras.

Quizás a esto se deba su popularidad, pues a pesar de que es fácil de manejar, puedes lograr operaciones tan complejas como sea necesario y nos permite aprovechar todas las ventajas que ofrece mediante el uso de comandos básicos.

Bibliografía

•HashiCorp. (2022). *Terraform Documentation*. Recuperado de https://www.terraform.io/docs

Universidad Tecmilenio

Infraestructura como código

IaC usando Terraform (workspaces y módulos)

Semana 5

Introducción

Uno de los aspectos más útiles de Terraform es el concepto de espacios de trabajo (workspaces).

La mayoría de los recursos que se pueden administrar con Terraform no incluyen un nombre único como parte de su configuración; muy seguido vas a terminar con un mismo grupo de archivos de configuración para aprovisionar grupos de recursos similares.

Dentro de la ingeniería de software, esta duplicidad de código es mal vista debido al simple hecho de que, si necesitas hacer un cambio que afecta a un grupo de recursos similares, tendrás que hacer el mismo cambio en muchos lugares distintos, y es posible que olvides alguno (Thomas y Hunt, 2019).

Explicación

Gestión de espacios de trabajo y entornos (workspaces)

Explicación

Gestión de espacios de trabajo y entornos (workspaces)

- Condicionar los archivos de configuración para tener una misma configuración que te sirva para todos los ambientes.

- Por ejemplo, crear dos buckets, en el primero vas a almacenar la información relacionada con el ambiente de desarrollo y en el segundo vas a almacenar la información relacionada con el ambiente de producción.

Explicación

Arquitecturas para gestionar entornos sin espacios de trabajo

Carpeta

Forma más segura y menos eficiente de gestionar los entornos. Para lograrlo, se generan copias completas de la carpeta por cada entorno.

Parámetros

Archivos de configuración que serán los mismos para todos los ambientes y cuando se corre el terraform plan o apply, se especifica el archivo de variables que se va a utilizar por cada ambiente.

Módulos

Constan de archivos de configuración que puedes usar como parte de otras configuraciones, lo que te puede ahorrar mucho tiempo al momento de escribir infraestructuras complejas.

Universidad Tecmilenio

Ejercicio

Paso 1.

Hacer una carpeta con el nombre terraform que contenga un archivo llamado “main.tf”, un archivo llamado “prod.tfvars” y una carpeta llamada “vm” con un archivo llamado “main.tf”.

Paso 2.

En el archivo terraform/main.tf vas a escribir la variable de entrada para pasar el ambiente como parámetro, las variables necesarias para generar la máquina virtual y el bloque del módulo “vm” para la generación de una máquina virtual.

```
# terraform/main.tf

variable "ambiente" {}
variable "ami_id" {}

module "vm" {
  source = "./vm"

  ambiente = var.ambiente
  ami_id = var.ami_id
}
```

Paso 3.

En el archivo terraform/prod.tfvars escribirás las variables que vas a necesitar para crear la máquina virtual y sus valores por defecto. El nombre de la AMI lo puedes encontrar en https://cloud-images.ubuntu.com/locator/ec2/ pero te recomiendo utilizar la ami-0d73480446600f555

```
# terraform/prod.tfvars
ambiente = "prod"
ami_id   = "ami-0d73480446600f555"
```

Paso 4.

En el archivo terraform/vm/main.tf vas a ingresar el siguiente bloque para la generación de una máquina virtual en AWS.

Paso 5.

En la consola, correr terraform init y revisar que Terraform se inicialice correctamente con el proveedor “aws” y el módulo “vm”.

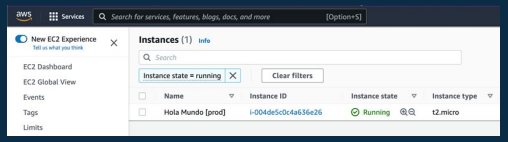
```
# terraform/vm/main.tf
variable "ambiente" {}
variable "ami_id" {}

resource "aws_instance" "maquina-virtual" {
  ami           = var.ami_id
  instance_type = "t2.micro"

  tags = {
    Name = "Hola Mundo [${var.ambiente}]"
  }
}
```

Paso 6.

En la consola, correr terraform plan -var-file=prod.tfvars y terraform apply -var-file=prod.tfvars para generar la máquina virtual. Al revisar la consola de AWS, debería verse así:



Paso 7.

Ejecuta terraform destroy para eliminar el bucket que ya no vas a utilizar. Recuerda que tener recursos que no necesitas en las nubes informáticas puede incurrir en costos no deseados.

Cierre

Terraform te da mucha flexibilidad a la hora de elegir cómo quieres aprovisionar los recursos, así como las herramientas adecuadas para evitar errores que pudieran surgir por estar posicionado en un ambiente distinto al que necesitas.

Ahora necesitas decidir cómo vas a llevar a cabo estos procedimientos o revisar cómo lo están haciendo dentro de tu organización y evaluar la mejor opción para cada caso en particular.

Bibliografía

•Thomas, D., y Hunt, A. (2019). *The Pragmatic Programmer: Your Journey to Mastery* (2ª ed.). Estados Unidos: Addison-Wesley Professional.