El uso de módulos con Terraform

DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Índice

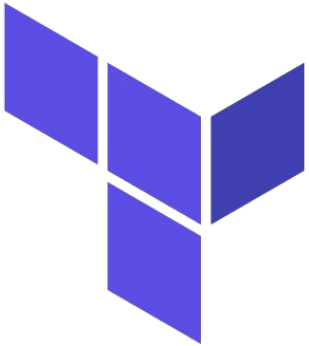
1 | Introducción

Don’t repeat yourself

Cuando trabajamos con el aprovisionamiento de infraestructura, vamos a notar que creamos recursos similares todo el tiempo. Podemos aprovechar el tiempo y el esfuerzo invertido reutilizando nuestro código. **¡Es aquí que los módulos toman verdadera importancia!**

¿Qué es un módulo en Terraform?

El módulo es una **librería** que podemos utilizar dentro de nuestro código o dentro de otro módulo —¡sí, como una mamushka!—. Esto es una **arquitectura de capas** que nos brinda abstracción de ciertos detalles de los recursos que estamos administrando.



HashiCorp
Terraform

2 | Organización

¿Cómo organizamos los módulos?

Al momento de crear nuestros módulos, debemos tener en cuenta estas consideraciones:

- Es una **librería** que queremos compartir con otras aplicaciones, por lo que nunca lo guardamos con el resto de nuestro código.
- Versionamos nuestro código —por ejemplo, con tags de Git—, para que no nos afecten los **cambios del código** del módulo.
- Definimos el caso de uso para poder **organizar la estructura** de nuestro módulo. Puede ser un proveedor de nube, por ejemplo.
- No hay ninguna nomenclatura especial para organizar los módulos, es importante **estandarizar el método de trabajo** con nuestro equipo.

¿Cómo organizamos los módulos?

Hay un orden para nuestro código que se encuentra aceptado por la comunidad. La estructura de los archivos es:

- **inputs.tf:** contiene valores que podemos reutilizar en otros archivos del módulo, pero no fuera de él. Son variables.
- **vars.tf:** definimos las variables que serán usadas como inputs del módulo.
- **outputs.tf:** define los valores que el módulo ofrece como salida y que pueden ser consultados por otros módulos.
- **main.tf:** contiene el código que aprovisiona los recursos en la nube.

3 | Creación

Creación de módulos

Para crear nuestros módulos vamos a mantener la siguiente estructura en nuestro espacio de trabajo.

```
$ tree my-terraform-modules/
my-terraform-modules/
└── aws
    └── modules
        └── resources
            ├── inputs.tf
            ├── main.tf
            ├── outputs.tf
            └── vars.tf

3 directories, 4 files
```

Creación de módulos

Es una buena práctica que lo versionemos, por ejemplo, en GIT. En ese caso, marcamos nuestro código en un momento exacto que es el que usamos. De esta manera, si hay un cambio de código del módulo, no va a afectar todas nuestras automatizaciones.

```
{}
git commit
git push origin main
git tag -a v1.0
git push origin v1.0
```

4 Utilización

Utilización de módulos

Ahora vamos a utilizar este módulo desde el repositorio de código. La tag v1.0 contiene la foto exacta en que queremos utilizar nuestro código. Si nuestra automatización se modifica, ya será la versión 2.0. De acuerdo a lo que encontremos en “input.tf”, vamos a poder crear el recurso que está definido en el módulo.

{}

```
module "mi-modulo" {
   source =
"git@github.com:mi-usuario/my-terraform-modules.git//aws/modules/
resources?ref=v1.0"
   recurso = "el nombre de mi recurso"
}
```

Utilización de módulos

Para finalizar, vamos a ejecutar con los comandos de Terraform que ya conocemos y aplicamos nuestra nueva infraestructura.

```
export AWS_PROFILE=mi_profile
terraform init
terraform apply
```

5 | Conclusión

Conclusión

Los módulos de Terraform nos permiten crear la infraestructura de un modo más sencillo y rápido. Además, nos posibilitan reutilizar nuestro código en varios entornos o aplicaciones. Solo tenemos que cambiar los parámetros adecuados para obtener los resultados que deseamos.

Además, podemos utilizar módulos ya existentes. Esto nos facilita la tarea y nos permite añadir capas extras de configuración para la infraestructura. ¡Lo único que necesitamos es consumirlos!

DigitalHouse>