DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Infraestructura II

Terraform HCL - Hashicorp Custom Language

¡Les damos la bienvenida! En este espacio vamos a aprender con más detalle acerca del lenguaje HCL en Terraform.

Recordemos que el propósito principal de HCL en Terraform es el de declarar recursos. Esto es importantísimo ya que será la esencia de nuestro código. Mediante este lenguaje le vamos a decir a Terraform cómo administrar una colección determinada de infraestructura. ¡Vamos!

Semántica y estructura

La sintaxis y estructura del lenguaje HCL de Terraform consta de unos pocos elementos básicos:

```
resource "aws_vpc" "mi_vpc" {

 cidr_block = 10.0.0.0/16

}
```

Analizando la sintaxis general podemos descubrir dos sublenguajes integrados:

- El lenguaje estructural, que define la estructura de configuración jerárquica general y es una serialización de cuerpos, bloques y atributos de HCL.
- El lenguaje de expresión, usado para expresar valores de atributo, ya sea como literales o como derivaciones de otros valores.

En líneas generales estos sublenguajes se usan juntos dentro de los archivos de configuración para describir una configuración general, con el lenguaje estructural.

Tipo de bloque

Tomemos el snippet anterior y veamos detenidamente cada uno de sus componentes:

```
<BLOCK TYPE> "<PROVIDER_ELEMENT>" "<BLOCK LABEL>" {
```

Lo primero que necesitamos indicarle es el tipo de bloque que vamos a usar. Los tipos de bloques más comunes que suelen utilizarse son: RESOURCE, VARIABLE y DATA, aunque también existen otros.

Estos bloques “declaran” qué recurso de nuestro proveedor Cloud vamos a usar. Por ejemplo, al escribir:

```
resource "aws_vpc" "mi_vpc"
```

...estamos diciendo que:

1. Necesitamos trabajar con un bloque de tipo “resource”, “RESOURCE”.
2. Que el “resource” a instanciar va a ser “aws_vpc”, “PROVIDER_ELEMENT”. [1]
3. Y que lo queremos llamar “mi_vpc”, “BLOCK_LABEL”

Argumentos

Seguidamente encontramos los argumentos que asignan un valor a un nombre y aparecen dentro de bloques:

```
 # Tipo de Bloque

 <IDENTIFIER> = <EXPRESSION> # Argumento/s

}
```

Es el "contenido" del "block body". Son los valores y/o funciones que se leen durante el tiempo de ejecución del código y es donde asignamos los valores que queremos para nuestros elementos de infraestructura.

Estos argumentos los podemos dividir en "Identificador" y "Expression". Ambos están estructurados como llave, valor y están separados por un signo igual.

La llave representa un identificador y el valor es lo que ese identificador almacena.

Un sencillo y claro ejemplo es el de región dentro de provider:

```
region = "us-east-1"
```

Remitiendonos a nuestro código inicial:

```
cidr_block = 10.0.0.0/16
```

...estamos diciendo que mis argumentos serán:

1. Una llave denominada "cidr_block" [2]
2. Que quiero que la subnet base del VPC que estoy construyendo sea 10.0.0.0/16

Es importante destacar que los argumentos que Terraform acepta no son arbitrarios, sino que está programado para recibir ciertas palabras específicas. [2]

Existen, también, argumentos que son mandatarios, es decir, que deben existir en nuestro código y otros que no. Por ejemplo, "cidr_block" es un argumento requerido porque nuestro VPC requiere que nosotros le digamos en qué red vamos a crearlo.

Variables

HCL usa variables de entrada que sirven como parámetros para un módulo de Terraform.

Declaramos una variable de la siguiente manera:

```
variable "image_id" {

 type = string

}
```

La palabra "variable" es un bloque que luego admite solamente un BLOCK_LABEL.

En nuestro caso, "image_id" y que debe ser "único" entre todas las variables ya definidas.

El nombre de una variable puede ser cualquier identificador válido excepto las siguientes: source, version, providers, count, for_each, lifecycle, depends_on, locals que son los denominados meta-argumentos, que no se cubrirán en este curso, pero recomendamos navegarlos.

La lista completa puede verse en Resources Overview - Configuration Language. Dentro de su estructura, también vamos a encontrar argumentos.

Ahora bien, ¿qué estamos diciendo cuando introducimos la palabra "type"? Básicamente, que el contenido va a ser de tipo "string" ya que puede haber casos en los que en lugar de datos de tipo string, podremos tener: number, bool, list, map, tuple, etc. La lista completa puede verse en el siguiente enlace: Input Variables - Configuration Language

Variables: un caso práctico

Supongamos que no queremos que el valor neto de nuestra subnet aparezca "hardcodeado". Lo que podemos hacer es crear una variable con el dato en sí, y luego, desde, este código, referenciar a esa variable.

En nuestro ejemplo inicial:

```
resource "aws_vpc" "mi_vpc" {

 cidr_block = 10.0.0.0/16

}
```

… lo podríamos reemplazar, por ejemplo, por los siguientes argumentos:

```
resource "aws_vpc" "mi_vpc" {

cidr_block = var.base_cidr_block

}
```

Para ello alcanza con declarar la variable de la siguiente manera:

```
variable "base_cidr_block" {

 default = 10.0.0.0/16

}
```

De esta forma, estamos personalizando aspectos del módulo sin alterar el código fuente del propio módulo.

Otra forma más práctica es definiendo un archivo con el nombre: variables.tf donde declaramos una o más variables logrando modularizar nuestro código.

Nota: es importante que este archivo de variables esté ubicado en el mismo directorio en el que se encuentra nuestro módulo primario.

Sintaxis

Palabras Reservadas

La mayoría de los lenguajes de programación contienen palabras que no se pueden utilizar como variables de asignación. Dichas palabras se hacen llamar "reservadas" debido a que son utilizadas por el lenguaje para funciones específicas. En el caso de HCL, Terraform no dispone de palabras reservadas a nivel global.

Comentarios

Existen tres sintaxis diferentes para comentarios:

1. # comienza un comentario de una sola línea, que termina al final de la línea.
2. // también comienza un comentario de una sola línea, como alternativa a #.
3. / * y * / son delimitadores de inicio y fin de un comentario que puede abarcar varias líneas.

El estilo de comentario # de una sola línea es el estilo de comentario predeterminado y debe usarse en la mayoría de los casos.

Concluyendo

No es necesario conocer todos los detalles de la sintaxis HCL para utilizar Terraform. De hecho, solo con comprender su estructura y qué es lo que se quiere lograr ya estamos en condiciones de crear módulos más complejos. El resto consiste en acudir a la documentación oficial para buscar que tipo de recursos preciso, qué argumentos lleva y ¡un poco de imaginación!

[1]: Podemos encontrar la lista completa de recursos en Docs overview | hashicorp/aws, aunque siempre sugerimos navegar el sitio de terraform.io directamente debido a que el team de Terraform actualiza constantemente sus recursos online.

[2]: En este enlace podrán encontrar las referencias a los argumentos que Terraform espera: aws_vpc | Resources | hashicorp/aws