Certified Tech Developer
The Ultimate Degree

Infraestrutura II

Terraform HCL - Hashicorp Configuration Language

Te damos as boas-vindas! Neste espaço, vamos aprender mais detalhadamente sobre a linguagem HCL na Terraform.

Lembrando que a finalidade principal da HCL na Terraform é a declaração de recursos. Isso é super importante pois será a essência do nosso código. Através desta linguagem, vamos indicar à Terraform como gerenciar uma determinada coleção de infraestrutura. Vamos lá!

Semântica e estrutura

A sintaxe e estrutura da linguagem HCL da Terraform é composta por alguns elementos básicos:

```
resource "aws_vpc" "minha_vpc" {

 cidr_block = 10.0.0.0/16

}
```

Analisando a sintaxe geral, podemos descobrir duas sub-linguagens integradas:

- A linguagem estrutural, que define a estrutura da configuração hierárquica geral e é uma serialização de corpos, blocos e atributos da HCL.
- A linguagem de expressão, utilizada para expressar valores de atributo, seja como literais ou como derivações de outros valores.

De um modo geral, estas sub-linguagens são utilizadas em conjunto dentro dos arquivos de configuração para descrever uma configuração geral, por meio da linguagem estrutural.

Tipo de bloco

Vamos tomar o snippet anterior e observar detalhadamente cada um dos seus componentes:

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

A primeira coisa que precisamos fazer é indicar o tipo de bloco que iremos usar. Os tipos de blocos mais comumente utilizados são: RESOURCE, VARIABLE e DATA, embora existam outros.

Esses blocos "declaram" qual é o recurso do nosso fornecedor Cloud que iremos usar. Por exemplo, ao escrever:

```
resource "aws_vpc" "minha_vpc"
```

Estamos dizendo que:

1. Precisamos trabalhar com um bloco do tipo "resource", "RESOURCE".
2. Que o "resource" a ser instanciado será "aws_vpc", "PROVIDER_ELEMENT". [1]
3. E que o queremos nomear como "minha_vpc", "BLOCK_LABEL"

Argumentos

A seguir, encontramos os argumentos que atribuem um valor a um nome, e que aparecem dentro de blocos:

```
 # Tipo de Bloco

 <IDENTIFIER> = <EXPRESSION> # Argumento/s

}
```

É o conteúdo do "block body". Trata-se de valores e/ou funções que são lindas durante o tempo de execução do código, e é onde atribuímos os valores que queremos para nossos elementos de infraestrutura.

Esses argumentos podem ser divididos em "identificador" e "expression". Ambos estão estruturados como chave, valor, e são separados por um sinal de igual.

A chave representa um identificador e valor é aquilo que o identificador armazena.

Um simples e claro exemplo é o da região dentro do provider:

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

Voltando para nosso código inicial:

```
cidr_block = 10.0.0.0/16
```

Estamos dizendo que nossos argumentos serão:

1. Uma chave chamada "cidr_block" [2]
2. Que eu quero que a subnet base da VPC que eu estou construindo seja 10.0.0.0/16

É importante mencionar que os argumentos aceitos pela Terraform não são arbitrários, mas ela é programada para receber determinadas palavras específicas.

Existem, também, argumentos obrigatórios, ou seja, que devem existir no nosso código, e outros que não. Por exemplo, "cidr_block" é um argumento exigido, pois nossa VCP precisa que nós lhe indiquemos qual é a rede na qual iremos criá-lo.

Variáveis

HCL usa variáveis de entrada que servem como parâmetros para um módulo da Terraform.

Declaramos uma variável de entrada desta forma:

```
variable "image_id" {

 type = string

}
```

A palavra "variable" é um bloco que depois admitirá apenas um BLOCK_LABEL.

No nosso caso, "image "image_id", que deve ser "único" entre todas as variáveis já definidas.

Uma variável pode ser nomeada com qualquer identificador válido, excetuando as seguintes: source, version, providers, count, for_each, lifecycle, depends_on, locals que são os chamados "meta-argumentos", os quais não serão abordados neste curso. Porém, recomendamos explorá-los. Podemos visualizar a lista completa em Resources Overview - Configuration Language. Dentro da sua estrutura, também encontraremos argumentos.

Bem, agora... O que estamos dizendo quando introduzimos a palavra "type"? Basicamente, que o conteúdo será do tipo "string", pois há casos nos quais, ao invés de dados do tipo string, poderíamos ter: number, bool, list, map, tuple, etc. No link a seguir, é possível visualizar a lista completa: Input Variables - Configuration Language

Variáveis: um caso prático

Vamos supor que não queremos que o valor neto da nossa subnet apareça “hardcodado”. O que podemos fazer é criar uma variável com o dado em si e, então, a partir deste código, referenciar essa variável.

No nosso exemplo inicial:

```
resource "aws_vpc" "minha_vpc" {

 cidr_block = 10.0.0.0/16

}
```

Poderíamos substituí-lo, por exemplo, pelos seguintes argumentos:

```
resource "aws_vpc" "minha_vpc" {

cidr_block = var.base_cidr_block

}
```

Para isso, basta declarar a variável desta forma:

```
variable "base_cidr_block" {

 default = 10.0.0.0/16

}
```

Assim, estamos customizando aspectos do módulo sem alterar o código-fonte do próprio módulo. Outra forma mais prática é definindo um arquivo com o nome: variables.tf, onde declaramos uma ou mais variáveis, conseguindo assim modularizar nosso código.

Nota: é importante que este arquivo de variáveis esteja localizado no mesmo diretório onde está nosso módulo primário.

Sintaxe

Palavras Reservadas

A maior parte das linguagens de programação contém palavras que não podem ser utilizadas como variáveis de atribuição. Essas palavras são denominadas “reservadas” pois são utilizadas pela linguagem para funções específicas. No caso da hcl, a Terraform não dispõe de palavras reservadas em nível global.

Comentários

Existem três sintaxes diferentes para comentários:

1. # começa um comentário de uma única linha, que finaliza no final da linha.
2. // também começa um comentário de uma única linha, como alternativa ao #.
3. / * e * / são delimitadores de início e fim de um comentário que pode abranger várias linhas.

O estilo e comentário # de uma única linha e é o estilo de comentário padrão, e deve ser usado na maior parte dos casos.

Conclusão

Não é preciso conhecer todos os detalhes da sintaxe HCL para utilizar a Terraform. Aliás, compreendendo sua estrutura e o que pretende-se conseguir, já estamos em condições de criar módulos mais complexos. O resto consiste em consultar a documentação oficial para saber quais são os tipos de recursos necessários, os argumentos que inclui… e um pouco de imaginação!

[1]: Em Docs overview | hashicorp/aws, podemos encontrar a lista completa de recursos. Porém, sempre recomendamos navegar o site da terraform.io diretamente, pois o Team da Terraform atualiza constantemente seus recursos online.

[2]: No link a seguir, você poderá encontrar as referências aos argumentos esperados pela Terraform: aws_vpc | Resources | hashicorp/aws