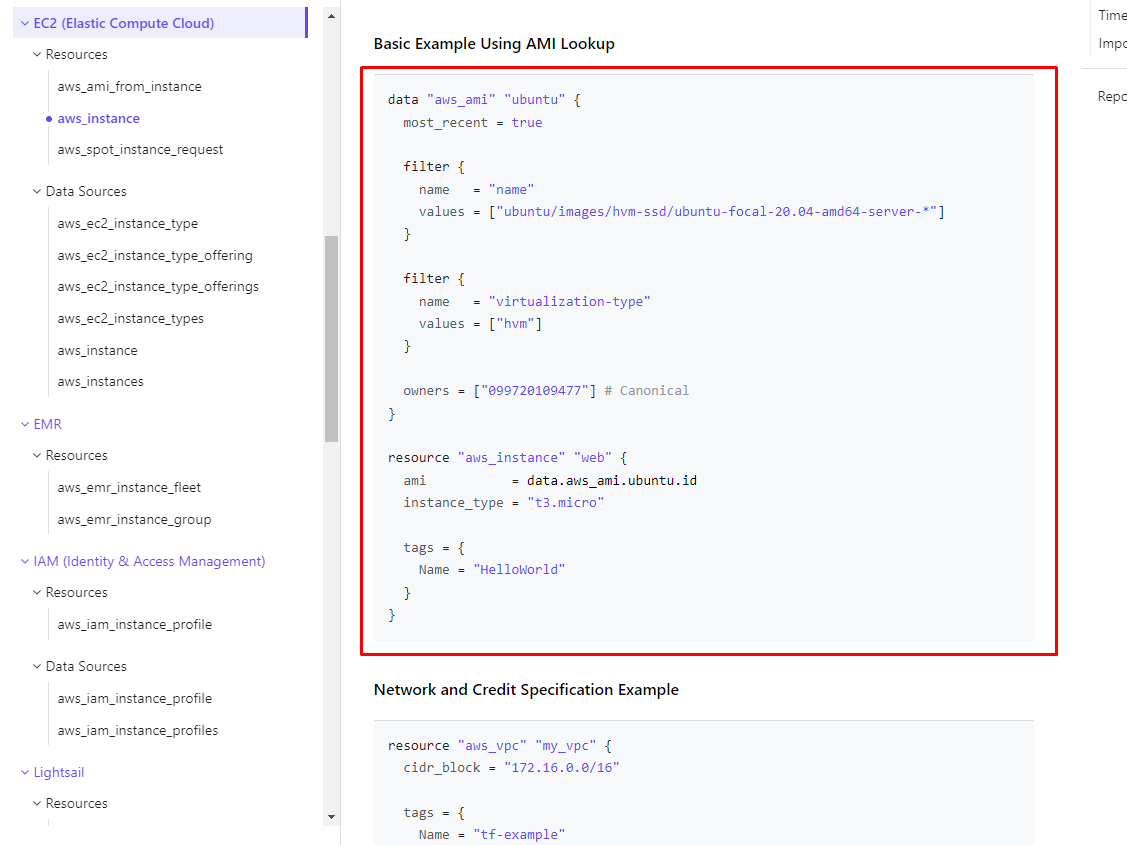
# TERRAFORM:

Documentação do Terraform:

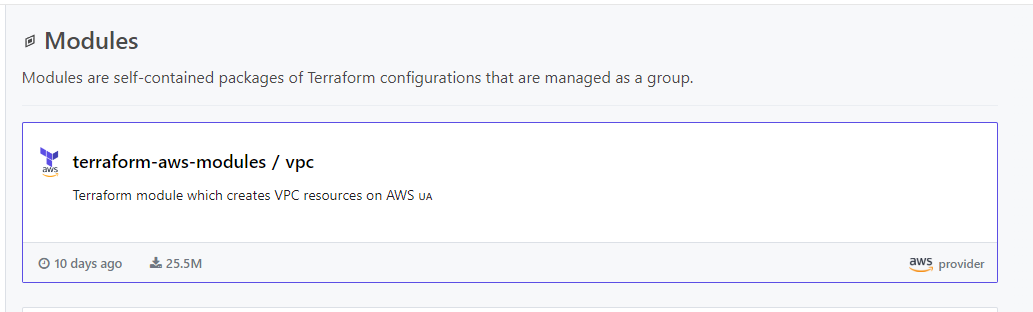
Existem as Documentação dos Provider: AWS, Google Cloud Plataform, e Alibaba Cloud.

Na guia na parte de documentation na AWS temos a barra de filtros que explica como nós podemos criar por exemplo algo simples como uma Instancia EC2:

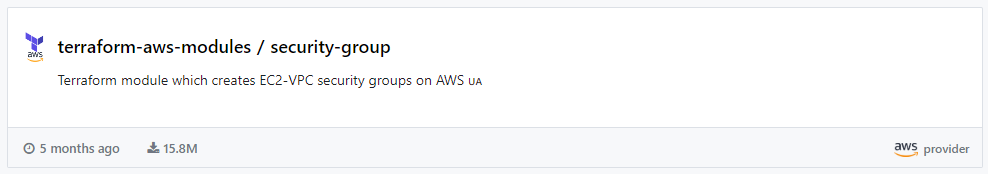


Temos as Documentação dos Modulos:

Os Modulos são conjuntos de códigos que serve para um fim especifico, por exemplo esse conjunto de código de modulo serve para criar uma estrutura de VPC:

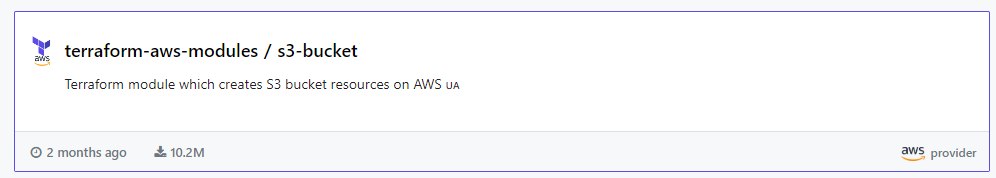


Esse outro modulo e uma estrutura de código que serve para criar um Security Group na AWS:

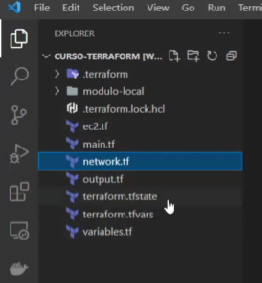


Temos milhares de Modulos, criados pelos Provider.

Modulo pra criar uma Bucket no S3:



Estrutura de Arquivos e de Diretórios do Terraform:



Importante, o Terraform.tfstate e um arquivo que o próprio Terraform vai criar localmente. O Terraform cria este arquivo State, que e onde ele armazena noções de tudo aquilo que ele gerou.

Importante outra parte, criar um arquivo com a extensão tfvars, essa parte do código e um arquivo que você declara o valor das variáveis que você vai utilizar dentro do seu código do Terraform, arquivo especificamente pra isso.

Chave e o valor, e o arquivo tfvars ele serve pra isso.

Além disso o que o Terraform cria faz parte da configuração do Terraform que e o terraform.lock.hcl

Temos também a pasta: terraform essa pasta terraform e onde o terraform vai fazer o download dos providers e dos modulos que você virá a usar dentro do seu código por exemplo:

Se você indica que quer usar um Provider da AWS, ele vai fazer o download da configuração desse Provider da AWS.

E assim por diante

Na questão dos modulos ele irá fazer o download de diversos modulos, para essa pasta, e você pode criar um próprio modulo local.

Normalmente criamos um diretório dentro do modulo local vamos colocar outros arquivos terraform, que vão fazer parte desse modulo.

ESTRUTURA DE BLOCOS DA LINGUAGEM HCL:

* Existem outros tipos de Blocos básicos no Terraform
* Terraform
* Providers
* Resources
* Data
* Module
* Variable
* Outputs
* Locais

Vamos criar o primeiro bloco do terraform:

Dentro desse bloco vão diversas configurações, que são as configurações básicas do terraform

Bloco dos Providers:

Você vai dar o nome do provider, por exemplo se for AWS (todo bloco de provider tem sua configuração especifica que será mostrada)

Bloco do Resource:

Você dá o nome de Resource, você quer criar, podemos criar uma instancia EC2

Exemplo:

Resource “aws\_instance” “aqui eu posso dar o nome que eu quiser” {

}

Por exemplo eu posso chama-la de VM1, esse nome podemos usar para referenciar esse Resource em outros blocos posteriormente. Então e muito importante que seja um nome que faça referencia a aquilo que você está criando.

Bloco do Data:

O bloco data funciona da mesma forma que o bloco de Resource:

Damos o primeiro nome ao tipo de Data, o Data vai buscar informações de fora do código do terraform, pra ser usado no terraform:

Por exemplo: uma data “aws\_ami” “ami” {

Vai buscar informações de uma imagem da AWS, e da mesma forma temos que usar um nome que traga referencia trazer de informação pra cá.

}

Bloco do Modele:

Module “name” {

Vamos só dar o nome pra ele, por exemplo se eu quiser usar um modulo de VPC e o bloco de modulo, ele serve tanto pra referenciar o modulo que você cria, no subdiretório da sua configuração do terraform, como também serve pra referenciar o modulo da comunidade, do registro do terraform.

O bloco e o mesmo, só as configurações internas dele que mudam.

}

Bloco de Variables:

Variable “name” {

As variáveis, são uma declaração que você faz, que você vai usar dentro da configuração do terraform, da mesma forma vc vai colocar o nome.

Aqui dentro vai a configuração dessa variável que será mostrada.

}

Bloco output:

Output “name” {

É um recurso do terraform, pra pegar alguma informação de dentro do código do terraform, de dentro de onde o terraform criou, e fazer o output pra fora dessa documentação, pra poder ser usado por um outro recurso da sua máquina, ou dentro da sua Pipeline ou utilizar para o que quiser

Output ele pega alguma informação de dentro do terraform e coloca pra fora

}

Bloco Locals:

Locals {

O Locals basicamente ele serve pra você pegar funções ou expressões que você usa muito repetidamente, dentro da sua configuração do terraform, da um nome pra ela, ao invés de você ter que ficar repetindo aquele código toda vez, vc apenas chama esse código que você declara aqui, e pra isso que funciona o Locals.

}

CONFIGURAÇÃO DOS BLOCOS DO TERRAFORM

Como vamos fazer a configuração do bloco do terraform:

* Dentro dos 06 Blocos e Possível fazer esses 6 tipos de configurações:

1 - required\_Version

2 – required\_Provider

3 – back-end

4 – cloud

5 – experiements

6 – provider\_meta

Dentro do bloco terraform, a primeira coisa que podermos configurar, e a versão que queremos usar no nosso código do terraform:

# Required\_Version:

Terraform {

Isso nós fazemos mencionando o require\_version:

Require\_version: “value” e aqui dentro colocamos a versão ou o conjunto de versões que queremos que nosso código funcione.

}

Onde a regra é:

/\*Iremos mostrar como os operadores de Version funcionam:

Para uma versão especifica eu utilizo: 1.1.0 ou = 1.1.0

Para versões do terraform > ou = a 1.0.0 ou versões < 1.2.0

Nota: que só o intervalo de Versões de terraform entre estes 02 vai funcionar, ## required\_version = " >= 1.0.0, < 1.2.0" podemos fazer dessa forma também

      para versões do terraform ~> 1.0.0 #     Este código vai funcionar com todos as versões de terraform desde a 1.0.0 até a 1.1.n

    \*/

    required\_version = "= 1.1.0"

# Required\_Provider:

Esse serão os providers que iremos utilizar ou usar na configuração do terraform, por exemplo o provider da aws:

Required\_providers {

Aws {

## E dentro do next block da aws, vamos ter que indicar qual a versão, e da mesma forma do require\_version desde uma versão única, ou alguma coisa da mais atual.

Version = “2.70.0”

Source = hashcorp/aws

}

## Podemos ter mais de um provider na configuração do terraform

Azurerm {

Version = “1.70.0”

Source = “hashicorp/azurerm”

}

## podemos ter infinitos providers declarados na estrutura do terraform, isso e encontrado na documentação do terraform.

}

Basicamente o **Backend**, ele serve para guardar o State do terraform de forma remota, no caso mencionamos o “S3”, eu guardaria o State em uma S3 bucket da aws.

# Terraform CLI:

Linha de comando do terraform:

Verificando os 05 comandos básicos do terraform:

Terraform init – **Prepare you working Directory for Other commands**

Terraform validate – **Check Weather the configuration is valid**

Terraform plan – **Show change required by the current configuration**

Terraform apply – **Create and Update Infrastructure**

Terraform destroy – **Destroy previously-created infrastructure**