EBOOK DE FÁCIL APRENDIZADO

PRIMEIROS PASSOS UTILIZANDO TERRAFORM NA AWS

Coffee and Tips

Tech Tutorials

O que é Terraform?

O Terraform é uma ferramenta do tipo IaC (Infrastructure as code) que possibilita provisionar infra-estrutura nos serviços de nuvem, ou seja, ao invés de criar manualmente recursos na nuvem, o Terraform facilita a criação e o controle deste serviços através de gerenciamento de estado em poucas linhas código.

O Terraform tem sua linguagem própria e pode ser utilizada de forma independente com outras linguagens de forma isolada.

Para este ebook, iremos criar um **Bucket S3** e uma **SQS** utilizando Terraform na AWS.

Instalação Terraform

Para a instalação, faça o download do instalador neste link https://www.terraform.io/downloads.html e escolha o seu sistema operacional.

Provider AWS

Utilizaremos a AWS como provider, ou seja, quando selecionamos a AWS como provider, o Terraform fará o download dos pacotes que possibilitará a criação de recursos específicos para a AWS.

Para seguir nos próximo passos, estamos levando em conta que você já possui:

- Credenciais da AWS
- O seu usuário já possui permissões necessárias para criar recursos na AWS

Autenticação

Como nós estamos utilizando a AWS como provider, precisamos configurar o Terraform para autenticar e em seguida criar os recursos. Existem algumas maneiras de autenticação. Pare este tutorial, escolhi utilizar um dos mecanismos da AWS que permite alocar as credencias em um arquivo na pasta $HOME/.aws e utilizar como fonte de autenticação única.

Para criar esta pasta com as credenciais, precisamos instalar o **AWS CLI**, acesse o link a seguir https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html e siga os passos de instalação.

Este mecanismo evita a utilização das credenciais diretamente no código, dessa forma, caso precise executar algum comando ou SDK que conecte a AWS localmente, estas credencias serão carregadas a partir deste arquivo.

Configuração das credenciais

Após instalar o AWS CLI, abra o terminal e execute o comando a seguir:

aws configure

No próprio terminal, preencha os campos utilizando os dados das credencias do seu usuário:

```
AWS Access Key ID [****************KK7D]:
AWS Secret Access Key [****************9NrI]:
Default region name [us-east-1]:
Default output format [json]:
```

Após o preenchimento, 2 arquivos textos serão criados no diretório $HOME/.aws

1. **config**: contendo o profile, neste caso o profile default foi criado
2. **credentials**: contendo as credenciais

Vamos alterar os arquivos para adequar a este tutorial, altere o arquivo **config** conforme abaixo:

[profile staging]
output = json
region = us-east-1

[default]
output = json
region = us-east-1

No caso, temos 2 perfis configurados, o default e um perfil de staging.

Altere o arquivo **credentials** conforme abaixo, substituindo pelas suas credenciais.

```
[staging]
aws_access_key_id = [Access key ID]
aws_secret_access_key = [Secret access key]

[default]
aws_access_key_id = [Access key ID]
aws_secret_access_key = [Secret access key]
```

Criando os arquivos Terraform base

Após todas estas configurações, iremos começar a trabalhar de fato com o Terraform. Para isso precisamos criar alguns arquivos base que vai nos auxiliar na criação dos recursos na AWS.

1º Passo: No diretório root do seu projeto, crie um pasta chamado ***terraform/***

2º Passo: Dentro da pasta ***terraform/***, crie os arquivos:

- main.tf
- vars.tf

3º Passo: Crie uma pasta chamada staging dentro de terraform/

4º Passo: Dentro da pasta terraform/staging/ crie o arquivo:

- vars.tfvars

Pronto, agora temos a estrutura de pasta que vamos utilizar nos próximos passos.

Configurando os arquivos Terraform

Vamos começar pela declaração das variáveis utilizando o arquivo vars.tf.

vars.tf

Neste arquivo é onde vamos criar a variáveis em que vamos utilizar em nosso contexto, podemos criar variáveis com um valor default ou simplesmente vazias, onde estas serão preenchidas de acordo com o ambiente de execução, onde será explicado mais a frente.

```
variable "region" {
  default = "us-east-1"
  type = "string"
}

variable "environment" {
}
```

Criamos 2 variáveis:

- **region**: Variável do tipo *string* e seu valor default é a região da AWS em que vamos criar os recursos
- **environment**: Variável que vai representar o ambiente de execução

staging/vars.tfvars

Neste arquivo estamos definindo o valor da variável environment criada anteriormente sem valor default.

environment = "staging"

Essa separação é bem útil quando temos mais de um ambiente, caso tivéssemos um ambiente de produção, poderíamos ter criado outro arquivo vars.tfvars em uma pasta chamada production.

Dessa forma, podemos escolher em qual ambiente vamos executar o Terraform. Vamos entender esta parte, quando executamos mais a frente.

main.tf

Este será o principal arquivo onde iremos declarar os recursos para que sejam criados na AWS. Nesta etapa vamos declarar os recursos para que seja criado um Bucket S3 e uma SQS.

Vamos entendendo o arquivo em partes. Nesta primeira parte estamos declarando a AWS como provider e setando a região utilizando a variável que criamos anteriormente através de interpolação **${..}.**

Provider

```
provider "aws" {
  region  = "${var.region}"
}
```

Criando o Bucket S3

Para criar um recurso via Terraform, sempre começamos com a palavra chave **resource** e em seguida o nome do recurso e por fim um identificador.

resource "nome do recurso" "identificador" {}

Neste trecho estamos criando um Bucket chamado bucket.blog.data, lembre-se que nomes de Buckets devem ser únicos.

O campo **acl** define as restrições do Bucket, neste caso, private. O campo **tags** é utilizado para passar informações extras ao recurso, neste caso será passando o valor da variável environment.

Mais campos são descritos na documentação no link a seguir:
https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket

```
resource "aws_s3_bucket" "s3_bucket" {
  bucket = "bucket.blog.data"
  acl = "private"

  tags = {
    Environment = "${var.environment}"
  }
}
```

Criando a SQS

No próximo trecho, vamos criar uma SQS chamada sqs-posts. A criação do recurso segue as mesmas regras que descrevemos anteriormente.

Para este cenário configuramos os campos **delay_seconds** que define o tempo de espera de uma mensagem ser entregue.

Mais campos são descritos na documentação usando o seguinte o link https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/sqs_queue

```
resource "aws_sqs_queue" "sqs-blog" {
  name = "sqs-posts"
  delay_seconds = 90
  tags = {
    Environment = "${var.environment}"
  }
}
```

Executando o Terraform

1º Passo: Inicializar o Terraform
Dentro do diretório **/terraform** execute o comando:

```
terraform init
```

Mensagens no console após a execução:

```
Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 3.37.0...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.
```

2º Passo: No Terraform existem workspaces. São ambientes de execução em que o Terraform provê para executar os recursos e separar os estados entre eles. Após inicializado, um *workspace* default é criado.

```
terraform workspace list
```

Para este tutorial vamos simular um ambiente de desenvolvimento. Lembra que criamos uma pasta chamada **/staging** ? Sim, esta pasta simula um ambiente de desenvolvimento. Para isso, vamos criar um workspace no Terraform chamado staging também. Se tivéssemos um ambiente de produção, um workspace de produção poderia ser criado.

```
terraform workspace new "staging"
```

Pronto, criamos um novo workspace e já estamos utilizando.

```
MacBook-Pro-de-Joao:terraform joao$ terraform workspace new "staging"
Created and switched to workspace "staging"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
```

3º Passo: Neste passo, vamos listar todos os recursos existentes ou os que serão criados, neste caso, a última opção.

```
terraform plan -var-file=staging/vars.tfvars
```

O argumento **plan** possibilita visualizar os recursos que serão criados ou atualizados, é uma boa opção para entender o comportamento antes que o recurso seja criado definitivamente.

O segundo argumento **-var-file** possibilita escolher um caminho específico contendo os valores das variáveis que serão utilizadas de acordo com o ambiente de execução. Neste caso o arquivo **/staging/vars.tfvars** contém valores referentes ao ambiente de staging. Caso existisse um workspace de produção, a execução seria a mesma, porém para uma pasta diferente.

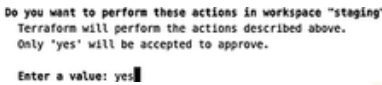
Mensagens no console após a execução:

```
Terraform will perform the following actions:

  # aws_s3_bucket.s3_bucket will be created
  + resource "aws_s3_bucket" "s3_bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = "private"
      + arn                         = (known after apply)
      + bucket                      = "bucket.blog.data"
      + bucket_domain_name          = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "staging"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + versioning {
          + enabled    = (known after apply)
          + mfa_delete = (known after apply)
        }
    }

  # aws_sqs_queue.sqs-blog will be created
  + resource "aws_sqs_queue" "sqs-blog" {
      + arn                               = (known after apply)
      + content_based_deduplication       = false
      + delay_seconds                     = 90
      + fifo_queue                        = false
      + id                                = (known after apply)
      + kms_data_key_reuse_period_seconds = (known after apply)
      + max_message_size                  = 262144
      + message_retention_seconds         = 345600
      + name                              = "sqs-posts"
      + name_prefix                       = (known after apply)
      + policy                            = (known after apply)
      + receive_wait_time_seconds         = 10
      + tags                              = {
          + "Environment" = "staging"
        }
      + visibility_timeout_seconds        = 30
    }

Plan: 2 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------
```

4º Passo: Neste passo, vamos criar os recursos definitivamente.

```
terraform apply -var-file=staging/vars.tfvars
```

Basta substituir o **plan** por **apply**, em seguida uma mensagem de confirmação será mostrada no console:



Digite ***yes*** para aplicar os recursos e aguarde o fim da execução.

Pronto, o Bucket S3 e a SQS form criados! Agora você pode conferir direto no console da AWS.

Escolha de workspace

Caso necessite mudar de workspace, execute o comando selecionando o workspace em que deseja utilizar:

```
terraform workspace select "[workspace]"
```

Destruindo os recursos

Esta parte do tutorial requer muita atenção. O próximo comando possibilita remover todos os recursos que foram criados sem a necessidade em remover um por um.

```
terraform destroy -var-file=staging/vars.tfvars
```

```
MacBook-Pro-de-Joao:terraform joao$ terraform destroy -var-file=staging/vars.tfvars
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes
```

Digite **yes**, caso deseja que todos os recursos criados sejam destruídos.

Não recomendo utilizar este comando em um ambiente profissional, mas para este tutorial é útil para que você não esqueça de apagar e a AWS te cobrar no futuro.

Conclusão

Terraform possibilita criar infra-estruturas de forma bem simples através de código e também oferece bastante segurança mantendo os recursos através do uso de estados. Para este tutorial utilizamos a AWS como provider, mas é possível utilizar Google Cloud, Azure e entre outros.

Curtiu? Eu espero que tenha gostado!

CURTIU? ACESSE MAIS TUTORIAIS COMO ESSE EM COFFEEANDTIPS.COM

E NOS SIGA NO INSTAGRAM

Coffee and Tips

Tech Tutorials