Certified Tech Developer
The Ultimate Degree

Infraestrutura II

Construímos uma infraestrutura de vida real!

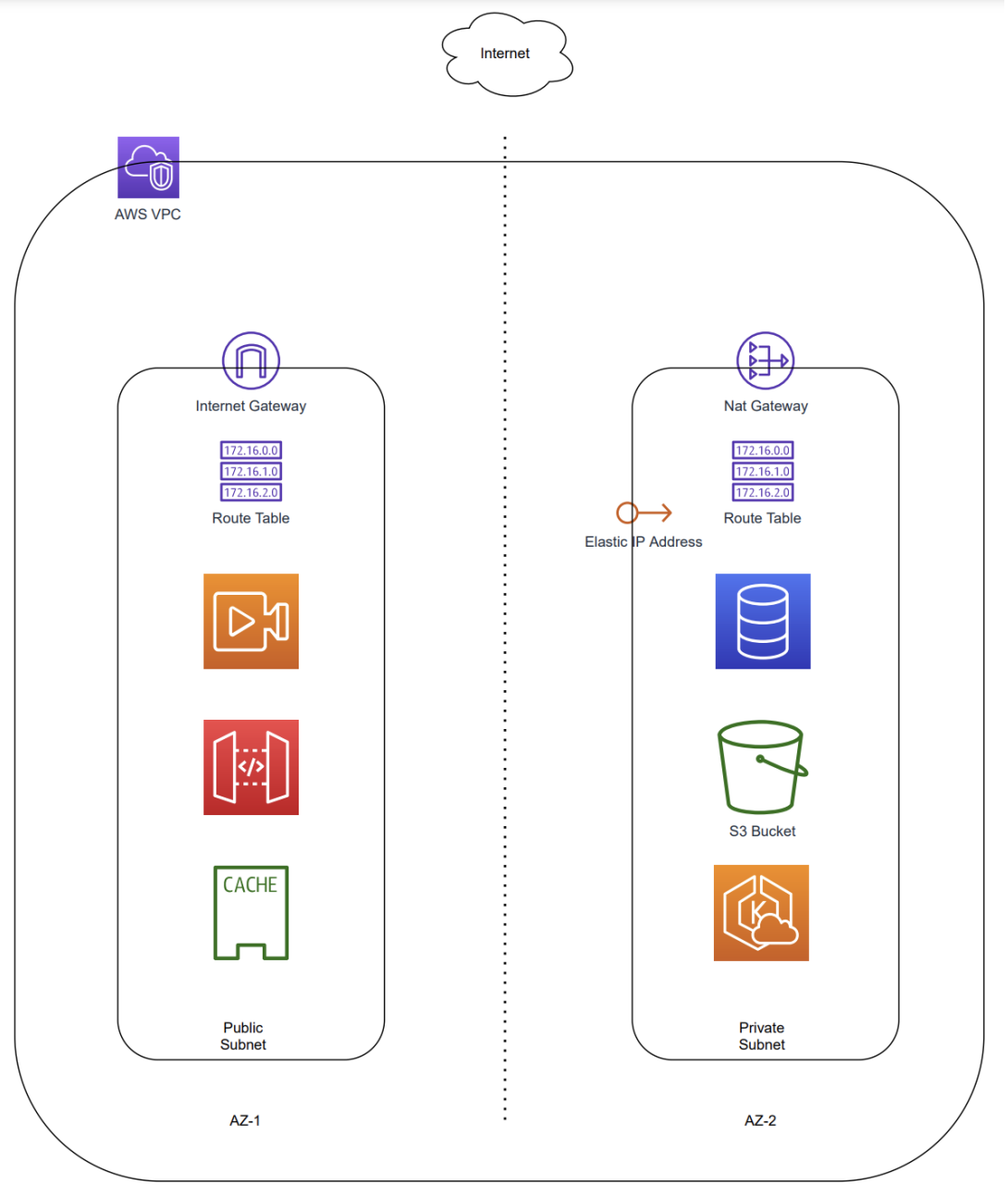
Atividade necessária

Dificuldade: Média

O objetivo é construir uma infraestrutura em nuvem mais robusta, segura e resiliente, consistindo apenas nos seguintes recursos de infraestrutura:

- Um VPC.
- Um gateway de Internet associado ao VPC criado.
- Uma sub-rede pública.
- Uma sub-rede privada.
- Uma tabela de roteamento dedicada à sub-rede pública.
- Uma tabela de roteamento dedicada à sub-rede privada.
- As associações de ambas as tabelas de roteamento com suas respectivas sub-redes.
- Um gateway NAT associado à sub-rede privada.
- Um IP elástico.

O código será modularizado, ou seja, segmentado em três arquivos: um para variáveis, outro para selecionar o provedor de nuvem e o último a ser utilizado para construir a infraestrutura. Será assim:



Embora, por questões de praticidade, apenas nos concentremos na infraestrutura e não nos serviços que surgirão dentro de você.

Dica: vamos ter o link para a documentação do Terraform à mão, pois iremos consultá-la à medida que prosseguirmos.

Documentação

Vamos trabalhar!

Estrutura de diretório e módulos

Para cada módulo Terraform que vamos usar para funcionar, devemos colocá-los todos no mesmo diretório. Os arquivos serão chamados de:

- **main.tf** (Servirá para elevar a infraestrutura de todo o meu VPC a aumentar).
- **variables.tf** (conterá as variáveis que desejo passar para cada módulo).
- **provider.tf** (será usado para definir qual provedor de nuvem e quais versões usar).

Quando executarmos o terraform init, a primeira coisa que acontecerá é a leitura do módulo, provider.tf, onde procurará qual provedor de nuvem usar. A seguir, a ordem é alfabética, ou seja, executará módulo por módulo de acordo com a inicial do nome do arquivo ou módulo.

Configurando o ambiente em três etapas:

1. **Declaração de variáveis**

   Nome do arquivo: variables.tf

```
# ====================================
====================================
```

```
# Objetivo: declaramos todas as variáveis que vamos usar

# Autor : DH

# Date: 07.30.21

# ========================================== ==
============================

variable "aws_region_id" {

   description = "a região"

   type        = string

   default = "us-east-1"

}

variable "main_vpc_cidr" {

   description = "Nosso Grupo de Segurança"

   type        = string

   default = "10.0.0.0/24"

}
```

```
variable "public_subnets" {

   description = "sub-rede com acesso à internet"

   type        = string

   default = "10.0.0.128/26"

}



variable "private_subnets" {

   description = "sub-rede sem acesso à Internet"

   type        = string

   default = "10.0.0.192/26"

}

# ======================================== =
============================
```

2. **Declaração do provedor de uso**

Nome do arquivo: Provedores.tf

```
# =============
============================================================ ====

# Objetivo: declaramos qual provedor de nuvem queremos usar
```

```
# Autor: DH

# Data: 30.07.21

# Versão: 1.0

# ================== ==
================================================

# ==== = ========================================================== =
===========

# Declaramos o provedor de nuvem com o qual queremos trabalhar com o

terraform {

# Dizemos que queremos:

# a. a versão do binário de terraform maior ou igual a 0,12

 required_version = "> = 0,12"

 required_providers {

   aws = {

# Nós especificamos de onde queremos fazer o download do binário:

     source = "hashicorp / aws"

# Dizemos que só permitirá: ma

# b. o provedor binário versão 3.20.0 (com algumas restrições)
```

```
    version = "~> 3.20.0"

  }

 }

}

# ======================================
================================

# ==================== ===
================================================

# Declaramos a região onde desejamos para aumentar nossa infra

provider "aws" {

 shared_credentials_file = "~ / .aws / credentials"

 region = "us-east-1"

}

# ========================================== =
============================
```

3. **Construir a infraestrutura de base**

Nome do arquivo: Main.tf

```
# Objetivo: Criar a infraestrutura da AWS

# Autor: DH
```

```
# Data: 07.30. 21

# Versão: 1.0

# ======================================== ==== ====================

# ============================== === ========================================

# Nós criamos nossoVPC

resource "aws_vpc" "Principal" {                    # usamos o bloco "recurso", o
"elemento provedor" e um "rótulo"

 cidr_block        = var.main_vpc_cidr     # nós passamos o bloco CIDR que eu quero
usar como uma variável

 instance_tenancy = "predefinição"

 tags = {

 Name = "My_VPC"

 }

}

# ========================================== ============================

# Criamos um Internet Gateway "E" o associamos ao PC que acaba de ser criado

resource "aws_internet_gateway" "IGW" {    # Gateway de Internet

 vpc_id =  aws_vpc.Principal.ir                # saberemos o vpc_id apenas quando o
VPC for criado

 tags = {

 Name = "IGW"

 }

}
```

```
# ============================================== ============================

# Nós criamos a sub-rede pública

resource  "aws_subnet" "public_subnets" {    # nós criamos as sub-redes públicas

 vpc_id =  aws_vpc.Principal.ir

 cidr_block = var.public_subnets        # Bloco CIDR para minhas sub-redes públicas

 tags = {

 Name = "Sub-rede pública"

 }

}

# ========================================= = ============================

# Nós criamos a sub-rede privada # Nós criamos nossa sub-redes privada

resource  "aws_subnet" "private_subnets" {

 vpc_id =  aws_vpc.Principal.ir

 cidr_block = var.sub-redes_privadas            # Bloco CIDR para minhas sub-redes
privadas

 tags = {

   Name = "Sub-rede privada"

 }

}

#                  ========================================                  =
==============================roteamento

# Tabela depara sub-rede pública
```

```
resource  "aws_route_table" "Public_RT" {    # Criamos nossa Tabela de Rotas para a sub-rede pública

 vpc_id =  aws_vpc.Principal.ir

 route{

   cidr_block = "0.0.0.0/0"                 # Declaramos que o tráfego da sub-rede pública chega à Internet a partir do gateway da Internet

   gateway_id = aws_internet_gateway.IGW.id

 }

 tags = {

   Name = "Public Routing Table"

 }

}

#                    ========================================                    ==============================roteamento

# Tabela depara sub-rede privada

resource  "aws_route_table" "Private_RT" {    # Criando RT para sub-rede privada

 vpc_id = aws_vpc.Principal.ir

 route {

   cidr_block = "0.0.0.0/0"              # Tráfego proveniente da sub-rede privada alcançando a Internet via Gateway NAT

   nat_gateway_id = aws_nat_gateway.NAT_GW.id

 }

 tags = {
```

```
 Name = "Tabela de roteamento privado"

 }

}

# ====================================== == ============================rota

# Associação de tabela de sub-rede pública

resource  "aws_route_table_association" "Public_RT_Association" {

   subnet_id = aws_subnet.public_subnets.ir

   route_table_id = aws_route_table.Public_RT.id

}

# ============================================ = =================

# Associação de tabela de roteamento com sub-rede privada

resource  "aws_route_table_association" "Private_RT_Association" {

   subnet_id = aws_subnet.sub-redes privadas.ir

   route_table_id = aws_route_table.Private_RT.id

}

resource "aws_eip" "NAT_EIP" {

 vpc    = true

 tags = {

 Name = "NAT com elastic IP "

 }

}

# ================================== == ======================
```

```
# Criação do NAT gateway usando subnet_id e allocation_id

resource "aws_nat_gateway" "NAT_GW" {

allocation_id = aws_eip.NAT_EIP.ir

 subnet_id = aws_subnet.public_subnets.ir

 tags = {

   Name = "Gateway NAT alocado para sub-rede pública"

 }

}
```