DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Infraestrutura II

Montando uma infraestrutura da vida real!

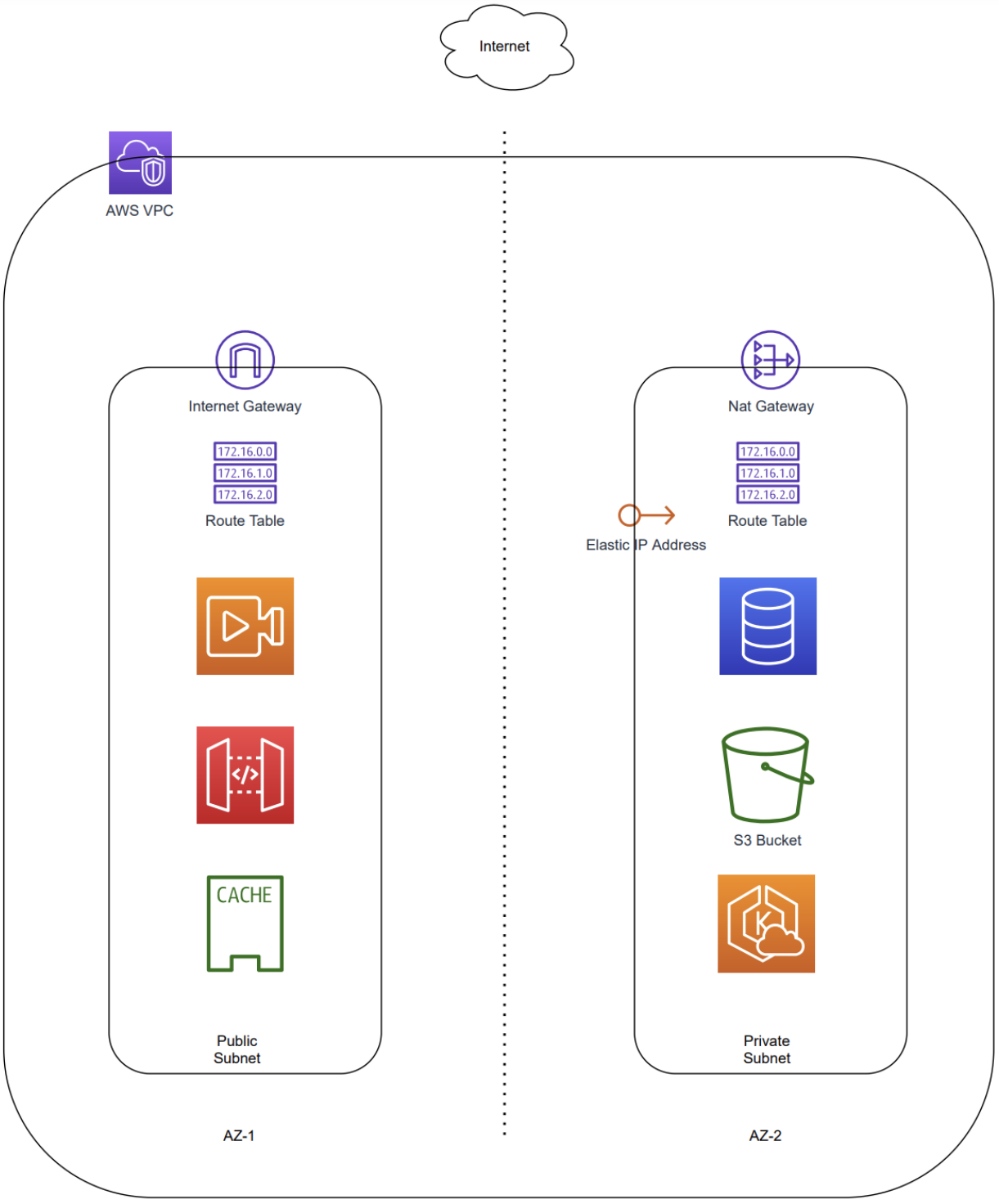
Atividade obrigatória

Dificuldade: intermediária

O objetivo é montar uma infraestrutura cloud mais robusta, segura e resiliente, que consista em apenas os seguintes recursos de infraestrutura:

- Uma VPC.
- Uma Internet gateway associada à VPC criada.
- Uma subnet pública.
- Uma subnet privada.
- Uma tabela de roteamento dedicada à subnet pública.
- Uma tabela de roteamento dedicada à subnet privada.
- As associações de ambas as tabelas de roteamento com suas respectivas subnets.
- Um NAT Gateway associado à subnet privada.
- Uma elastic IP.

O código será modularizado, ou seja, fragmentado em três arquivos: um para variáveis, outro para selecionar o fornecedor Cloud e o último que será utilizado para montar a infraestrutura. Ela terá a seguinte aparência:


Internet
AWS VPC
Internet Gateway
172.16.0.0
172.16.1.0
172.16.2.0
Route Table
CACHE
Public Subnet
AZ-1
Nat Gateway
172.16.0.0
172.16.1.0
172.16.2.0
Route Table
Elastic IP Address
S3 Bucket
Private Subnet
AZ-2

Embora, por motivos de praticidade, foquemos apenas na infraestrutura e não nos serviços montados dentro dela.

Dica: tenhamos à mão o link da documentação da Terraform, pois será consultado enquanto avançarmos.

Documentação

Mãos à obra!

Estrutura de diretório e módulos

Para que cada módulo da Terraform que iremos utilizar funcione, devem estar todos localizados no mesmo diretório. Os arquivos terão os seguintes nomes:

- **main.tf** (servirá para montar a infraestrutura de toda minha VPC a ser criada).
- **variables.tf** (conterá as variáveis que eu quero passar para cada módulo).
- **providers.tf** (servirá para definir o fornecedor cloud e as versões a serem utilizadas).

Quando executamos terraform init, o primeiro que irá ocorrer é a leitura do módulo, providers.tf, onde ele irá procurar o fornecedor cloud a ser utilizado. Em seguida, a ordem é alfabética, ou seja, ele irá executar módulo após módulo de acordo com a inicial do nome do arquivo ou módulo.

Montando o ambiente em três passos:

1. **Declaração de variáveis**

   Nome do arquivo: variáveis.tf

```
variable "aws_region_id" {
   description = "a região"
   type        = string
   default = "us-east-1"
}

variable "main_vpc_cidr" {
   description = "Nosso Security Group"
   type        = string
   default = "10.0.0.0/24"
}

variable "public_subnets" {
   description = "subnet com acesso à internet"
   type        = string
   default = "10.0.0.128/26"
}

variable "private_subnets" {
   description = "subnet sem acesso à internet"
   type        = string
   default = "10.0.0.192/26"
}
```

2. **Declaração do provider a ser utilizado**

   Nome do arquivo: providers.tf

```
# Declaramos o Cloud Provider com o qual iremos trabalhar

terraform {
# Indicamos o que queremos:
# a. a versão do binário da terraform maior ou igual a 0.12
 required_version = ">=0.12"
 required_providers {
   aws = {
# Especificamos a partir de onde queremos descarregar o binário:
     source = "hashicorp/aws"
# Indicamos-lhe que ele irá permitir apenas:ma
# b. a versão do binário do provider 3.20.0 (com certa restrição)
     version = "~> 3.20.0"
   }
 }
}
# ================================================================

# ================================================================
# Declaramos a região onde queremos montar nossa infra

provider "aws" {
 shared_credentials_file = "~/.aws/credentials"
 region = "us-east-1"
}
# ================================================================
```

3. **Criação da infraestrutura base**

Nome do arquivo: Main.tf

```
# Criamos nosso VPC
resource "aws_vpc" "Main" {
# usamos o bloco "resource", o "provider element" e um "rótulo"
 cidr_block = var.main_vpc_cidr
# passamos para ele, como variável, o CIDR block que queremos que utilize
 instance_tenancy = "default"
 tags = {
 Name = "My_VPC"
 }
}
# ====================================================================
# Criamos um Internet Gateway "Y" e o associamos ao VPC que acabamos de criar
resource "aws_internet_gateway" "IGW" {
# Internet Gateway
 vpc_id =  aws_vpc.Main.id
# vamos conhecer o vpc_id somente quando o VPC tenha sido criado
 tags = {
 Name = "IGW"
 }
}
# ====================================================================
# Criamos a subnet pública
resource "aws_subnet" "public_subnets" {
# criamos as subnets públicas
 vpc_id =  aws_vpc.Main.id
 cidr_block = var.public_subnets
# CIDR block para minhas public subnets
 tags = {
 Name = "Public Subnet"
 }
}
# ====================================================================
# Criamos a subnetprivada
```

```
# criamos nossas private subnets
resource "aws_subnet" "private_subnets" {
 vpc_id =  aws_vpc.Main.id
 cidr_block = var.private_subnets
# CIDR block para minhas subnets privadas
 tags = {
   Name = "Private Subnet"
 }
}
# ================================================================
# Tabela de roteamento para a subnet pública
resource "aws_route_table" "Public_RT" {
# Criamos nosso Route Table para a subnet pública
 vpc_id =  aws_vpc.Main.id
 route {
   cidr_block = "0.0.0.0/0"
# Declaramos o tráfego da sub-rede pública para a Internet a partir do Internet
Gateway
   gateway_id = aws_internet_gateway.IGW.id
 }
 tags = {
   Name = "Tabela de Roteamento pública"
 }
}
# ================================================================
# Tabela de roteamento para a subnet privada
resource "aws_route_table" "Private_RT" {
 # Creating RT for Private Subnet
 vpc_id = aws_vpc.Main.id
 route {
   cidr_block = "0.0.0.0/0"
# Tráfego proveniente das subnet privadas chegando na Internet via NAT Gateway
   nat_gateway_id = aws_nat_gateway.NAT_GW.id
```

```
 }
 tags = {
 Name = "Tabela de Roteamento Privada"
 }
}
# ====================================================================
# Associação da tabela de roteamento à subnet pública
resource "aws_route_table_association" "Public_RT_Association" {
   subnet_id = aws_subnet.public_subnets.id
   route_table_id = aws_route_table.Public_RT.id
}
# ====================================================================
# Associação da tabela de roteamento à subnet privada
resource "aws_route_table_association" "Private_RT_Association" {
   subnet_id = aws_subnet.private_subnets.id
   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.id
 subnet_id = aws_subnet.public_subnets.id
 tags = {
   Name = "NAT Gateway alocada à subnet pública"
 }
}
```