***Análise Técnica do Código Terraform***

**Descrição Técnica Detalhada**

O código Terraform provê a infraestrutura necessária para o provisionamento de uma instância EC2 na AWS com base no sistema operacional Debian 12. Abaixo estão os detalhes técnicos sobre os recursos criados e sua configuração:

**1. Provedor AWS**

hcl

Copiar código

provider "aws" {

region = "us-east-1"

}

* Define a AWS como provedor de infraestrutura e especifica a região us-east-1, que será utilizada para todos os recursos provisionados.

**2. Variáveis**

hcl

Copiar código

variable "projeto" {

description = "Nome do projeto"

type = string

default = "VExpenses"

}

variable "candidato" {

description = "Nome do candidato"

type = string

default = "SeuNome"

}

* Variável projeto: Define o nome do projeto, com o valor padrão "VExpenses".
* Variável candidato: Define o nome do candidato, com o valor padrão "SeuNome".
* Essas variáveis são usadas em vários recursos para nomear componentes como VPC, EC2, chave SSH e mais, facilitando a identificação dos recursos na AWS.

**3. Chave SSH**

3.1. TLS Private Key

hcl

Copiar código

resource "tls\_private\_key" "ec2\_key" {

algorithm = "RSA"

rsa\_bits = 2048

}

* Gera uma chave privada utilizando o algoritmo RSA com tamanho de 2048 bits.
* Esta chave privada será utilizada para autenticação SSH na instância EC2.

**3.2. AWS Key Pair**

hcl

Copiar código

resource "aws\_key\_pair" "ec2\_key\_pair" {

key\_name = "${var.projeto}-${var.candidato}-key"

public\_key = tls\_private\_key.ec2\_key.public\_key\_openssh

}

* Cria um par de chaves na AWS com o nome definido pelas variáveis ${var.projeto}-${var.candidato}-key.
* A chave pública é derivada da chave privada gerada anteriormente (tls\_private\_key.ec2\_key.public\_key\_openssh).

**4. Rede (VPC, Subnet e Gateway de Internet)**

4.1. VPC

hcl

Copiar código

resource "aws\_vpc" "main\_vpc" {

cidr\_block = "10.0.0.0/16"

enable\_dns\_support = true

enable\_dns\_hostnames = true

tags = {

Name = "${var.projeto}-${var.candidato}-vpc"

}

}

* Cria uma VPC (Virtual Private Cloud) com o bloco CIDR 10.0.0.0/16, permitindo até 65.536 endereços IP.
* A VPC tem suporte a DNS habilitado (enable\_dns\_support = true), o que permite a resolução de nomes DNS para instâncias dentro da VPC.
* A opção enable\_dns\_hostnames permite que as instâncias na VPC tenham nomes DNS públicos.

4.2. Subnet

hcl

Copiar código

resource "aws\_subnet" "main\_subnet" {

vpc\_id = aws\_vpc.main\_vpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-east-1a"

tags = {

Name = "${var.projeto}-${var.candidato}-subnet"

}

}

* Cria uma subnet dentro da VPC com o bloco CIDR 10.0.1.0/24, permitindo até 256 endereços IP.
* Está localizada na zona de disponibilidade us-east-1a, e está associada à VPC criada.

4.3. Internet Gateway

hcl

Copiar código

resource "aws\_internet\_gateway" "main\_igw" {

vpc\_id = aws\_vpc.main\_vpc.id

tags = {

Name = "${var.projeto}-${var.candidato}-igw"

}

}

* Cria um gateway de internet, necessário para permitir que instâncias na VPC tenham acesso à internet pública.
* O gateway é associado à VPC através do vpc\_id.

4.4. Tabela de Rotas

hcl

Copiar código

resource "aws\_route\_table" "main\_route\_table" {

vpc\_id = aws\_vpc.main\_vpc.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.main\_igw.id

}

tags = {

Name = "${var.projeto}-${var.candidato}-route\_table"

}

}

* Cria uma tabela de rotas para a VPC, permitindo que o tráfego com destino a 0.0.0.0/0 (todos os endereços IP) seja roteado através do gateway de internet (aws\_internet\_gateway.main\_igw.id).

4.5. Associação da Tabela de Rotas

hcl

Copiar código

resource "aws\_route\_table\_association" "main\_association" {

subnet\_id = aws\_subnet.main\_subnet.id

route\_table\_id = aws\_route\_table.main\_route\_table.id

tags = {

Name = "${var.projeto}-${var.candidato}-route\_table\_association"

}

}

* Associa a subnet criada (aws\_subnet.main\_subnet.id) à tabela de rotas (aws\_route\_table.main\_route\_table.id), garantindo que a subnet use a tabela de rotas apropriada.

**5. Segurança (Grupo de Segurança)**

hcl

Copiar código

resource "aws\_security\_group" "main\_sg" {

name = "${var.projeto}-${var.candidato}-sg"

description = "Permitir SSH de qualquer lugar e todo o tráfego de saída"

vpc\_id = aws\_vpc.main\_vpc.id

# Regras de entrada

ingress {

description = "Allow SSH from anywhere"

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

ipv6\_cidr\_blocks = ["::/0"]

}

# Regras de saída

egress {

description = "Allow all outbound traffic"

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

ipv6\_cidr\_blocks = ["::/0"]

}

tags = {

Name = "${var.projeto}-${var.candidato}-sg"

}

}

* Cria um grupo de segurança com regras que permitem:
  + Ingress (Entrada): Permitir acesso SSH (porta 22) de qualquer IP (0.0.0.0/0 e ::/0).
  + Egress (Saída): Permitir todo o tráfego de saída para qualquer destino (0.0.0.0/0 e ::/0).

**6. Instância EC2**

6.1. AMI Debian 12

hcl

Copiar código

data "aws\_ami" "debian12" {

most\_recent = true

filter {

name = "name"

values = ["debian-12-amd64-\*"]

}

filter {

name = "virtualization-type"

values = ["hvm"]

}

owners = ["679593333241"]

}

* Busca a AMI mais recente do Debian 12, filtrando pelas AMIs cujo nome corresponde ao padrão debian-12-amd64-\* e com o tipo de virtualização hvm.
* A AMI pertence ao proprietário com ID 679593333241 (Amazon).

6.2. Configuração da Instância

hcl

Copiar código

resource "aws\_instance" "debian\_ec2" {

ami = data.aws\_ami.debian12.id

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.main\_subnet.id

key\_name = aws\_key\_pair.ec2\_key\_pair.key\_name

security\_groups = [aws\_security\_group.main\_sg.name]

associate\_public\_ip\_address = true

root\_block\_device {

volume\_size = 20

volume\_type = "gp2"

delete\_on\_termination = true

}

user\_data = <<-EOF

#!/bin/bash

apt-get update -y

apt-get upgrade -y

EOF

tags = {

Name = "${var.projeto}-${var.candidato}-ec2"

}

}

* Cria uma instância EC2 do tipo t2.micro utilizando a AMI do Debian 12 obtida anteriormente.
* Está associada à subnet, grupo de segurança e par de chaves criados.
* A instância recebe um IP público (associate\_public\_ip\_address = true).
* Configuração do disco: volume de 20 GB do tipo gp2, deletado na terminação da instância.
* Script user\_data para atualização e upgrade do sistema operacional no primeiro boot.

**7. Saídas (Outputs)**

hcl

Copiar código

output "private\_key" {

description = "Chave privada para acessar a instância EC2"

value = tls\_private\_key.ec2\_key.private\_key\_pem

sensitive

4º

**8. Código Terraform com Melhorias**

**8.1. Alterações no Grupo de Segurança (AWS Security Group)**

Foi feita uma alteração na regra de entrada para restringir o acesso SSH a um endereço IP específico, ao invés de permitir acesso de qualquer lugar (0.0.0.0/0). Além disso, uma regra adicional foi adicionada para permitir o tráfego HTTP (porta 80) para o servidor Nginx.

hcl

Copiar código

resource "aws\_security\_group" "main\_sg" {

name = "${var.projeto}-${var.candidato}-sg"

description = "Permitir SSH apenas de IP específico e tráfego HTTP"

vpc\_id = aws\_vpc.main\_vpc.id

# Regras de entrada (Ingress)

ingress {

description = "Allow SSH from specific IP"

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["<INSIRA\_SEU\_IP\_AQUI>/32"] # Substituir pelo seu IP específico

}

ingress {

description = "Allow HTTP traffic"

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

ipv6\_cidr\_blocks = ["::/0"]

}

# Regras de saída (Egress)

egress {

description = "Allow all outbound traffic"

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

ipv6\_cidr\_blocks = ["::/0"]

}

tags = {

Name = "${var.projeto}-${var.candidato}-sg"

}

}

**Melhorias:**

* SSH Restrito: O acesso SSH agora está restrito a um endereço IP específico, prevenindo acessos indevidos de qualquer origem.
* Porta HTTP Aberta: Adicionada regra para permitir tráfego HTTP (porta 80) para que o Nginx esteja acessível ao público.

**8.2. Automação da Instalação e Configuração do Nginx**

Foi feita a modificação no script user\_data para automatizar a instalação e inicialização do servidor Nginx logo após a inicialização da instância EC2.

hcl

Copiar código

resource "aws\_instance" "debian\_ec2" {

ami = data.aws\_ami.debian12.id

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.main\_subnet.id

key\_name = aws\_key\_pair.ec2\_key\_pair.key\_name

security\_groups = [aws\_security\_group.main\_sg.name]

associate\_public\_ip\_address = true

root\_block\_device {

volume\_size = 20

volume\_type = "gp2"

delete\_on\_termination = true

}

# Script para automatizar a instalação do Nginx e configurar o servidor

user\_data = <<-EOF

#!/bin/bash

apt-get update -y

apt-get upgrade -y

apt-get install -y nginx

systemctl start nginx

systemctl enable nginx

EOF

tags = {

Name = "${var.projeto}-${var.candidato}-ec2"

}

}

**Melhorias:**

* Automação Completa do Nginx: O servidor Nginx será instalado automaticamente na inicialização da instância, iniciado e configurado para iniciar automaticamente após qualquer reinicialização do sistema. Isso reduz a necessidade de intervenção manual e prepara a instância para servir conteúdo web imediatamente após a criação.

**. Descrição Técnica das Melhorias**

. Melhorias no Grupo de Segurança

* SSH Restrito por IP: A regra de segurança que anteriormente permitia acesso SSH de qualquer endereço IP foi modificada para aceitar apenas conexões de um endereço IP específico. Isso aumenta significativamente a segurança ao limitar o acesso ao servidor, prevenindo tentativas de login de IPs desconhecidos ou maliciosos.
* Abertura da Porta HTTP (80): Para permitir o tráfego HTTP ao servidor Nginx, foi adicionada uma regra de entrada que abre a porta 80 para qualquer origem. Essa mudança é essencial para disponibilizar o conteúdo web de forma pública.

**Automação do Servidor Nginx**

O script de user\_data foi modificado para incluir a instalação automática do Nginx. As etapas automatizadas incluem:

* Atualização do Sistema: O sistema operacional é atualizado para garantir que os pacotes estejam na versão mais recente.
* Instalação do Nginx: O servidor web Nginx é instalado automaticamente.
* Início Automático do Nginx: O Nginx é iniciado assim que a instância for provisionada.
* Configuração de Inicialização Automática: O serviço Nginx é configurado para iniciar automaticamente sempre que a instância for reiniciada.

Essa automação elimina a necessidade de configuração manual pós-provisionamento, garantindo que o servidor esteja pronto para servir conteúdo web assim que for lançado.

**Outras Melhorias Implementadas**

**Volume de Log Separado**

Foi adicionada uma configuração para separar os logs do servidor Nginx em um volume de armazenamento adicional. Isso melhora a gestão de logs e evita que o volume principal fique sobrecarregado com logs.

hcl

Copiar código

resource "aws\_instance" "debian\_ec2" {

ami = data.aws\_ami.debian12.id

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.main\_subnet.id

key\_name = aws\_key\_pair.ec2\_key\_pair.key\_name

security\_groups = [aws\_security\_group.main\_sg.name]

associate\_public\_ip\_address = true

root\_block\_device {

volume\_size = 20

volume\_type = "gp2"

delete\_on\_termination = true

}

ebs\_block\_device {

device\_name = "/dev/xvdf"

volume\_size = 5

volume\_type = "gp2"

delete\_on\_termination = true

}

# Script para configurar logs no volume separado

user\_data = <<-EOF

#!/bin/bash

apt-get update -y

apt-get upgrade -y

apt-get install -y nginx

mkfs -t ext4 /dev/xvdf

mkdir -p /mnt/nginx\_logs

mount /dev/xvdf /mnt/nginx\_logs

chown -R www-data:www-data /mnt/nginx\_logs

sed -i 's|access\_log /var/log/nginx/access.log|access\_log /mnt/nginx\_logs/access.log|' /etc/nginx/nginx.conf

sed -i 's|error\_log /var/log/nginx/error.log|error\_log /mnt/nginx\_logs/error.log|' /etc/nginx/nginx.conf

systemctl start nginx

systemctl enable nginx

EOF

tags = {

Name = "${var.projeto}-${var.candidato}-ec2"

}

}

**Melhoria:**

* Volume de Logs Separado: Um volume EBS adicional de 5 GB foi configurado para armazenar os logs do Nginx, montado em /mnt/nginx\_logs. Os logs de acesso e de erros foram redirecionados para esse volume, garantindo que o armazenamento dos logs não impacte o desempenho do volume principal da instância.

**Monitoramento Básico**

Foi adicionado um alarme CloudWatch para monitorar o uso de CPU da instância EC2.

hcl

Copiar código

resource "aws\_cloudwatch\_metric\_alarm" "high\_cpu\_alarm" {

alarm\_name = "${var.projeto}-${var.candidato}-high-cpu"

comparison\_operator = "GreaterThanThreshold"

evaluation\_periods = "2"

metric\_name = "CPUUtilization"

namespace = "AWS/EC2"

period = "300"

statistic = "Average"

threshold = "80"

alarm\_actions = []

dimensions = {

InstanceId = aws\_instance.debian\_ec2.id

}

tags = {

Name = "${var.projeto}-${var.candidato}-high-cpu-alarm"

}

}

**Melhoria:**

* Monitoramento de CPU: Um alarme foi configurado para disparar quando a utilização de CPU da instância EC2 ultrapassar 80% por mais de 10 minutos. Isso permite detectar problemas de performance rapidamente e tomar ações corretivas.