IAC com Terraform e OCI

Neste repositório, vamos aprender a usar o Terraform e a OCI para criar a nossa primeira infra na nuvem.

Este documento também está disponível em formato PDF e formato HTML para que você possa visualizá-lo offline.

Tabela de conteúdos

Pré-requisitos

- Instalação do Terraform
  - https://developer.hashicorp.com/terraform/downloads?product_intent=terraform
  - **Usa Windows?** acesse esse documento
- Instalação do OCI CLI
  - https://docs.oracle.com/pt-br/iaas/Content/API/SDKDocs/cliinstall.htm
- Criando usuário na OCI
  - https://docs.oracle.com/pt-br/iaas/Content/GSG/Tasks/addingusers.htm#Add
- Realizar login no OCI CLI
  - `oci setup config`

Passo a passo

Vamos começar a diversão! 🥳

1. Comece fazendo o clone do repositório:

   > [!TIP] Se você preferir usar o Github é só trocar a URL do repositório para `https://github.com/avanti-dvp/iac-com-terraform-e-oci.git`

   ```
   git clone https://gitlab.com/avanti-dvp/iac-com-terraform-e-oci.git
   cd iac-com-terraform-e-oci
   ```

   > [!NOTE] Se você não tem o Git instalado ou não sabe usá-lo, sem problema algum, você pode simplesmente fazer o download do repositório e descompactá-lo em sua pasta/diretório de trabalho ou na pasta/diretório de seu usuário

2. Vamos abrir o Visual Studio Code no diretório do repositório:

   ```
   code .
   ```

3. Dentro do Visual Studio Code, crie um arquivo chamado `provider.tf`, incluindo esse trecho abaixo nele:

```
terraform {
  required_providers {
    oci = {
      source  = "oracle/oci"
      version = "~> 5.0"
    }
    tls = {
      source  = "hashicorp/tls"
      version = "4.0.5"
    }
    local = {
      source  = "hashicorp/local"
      version = "2.5.1"
    }
  }
}

provider "oci" {
  tenancy_ocid     = var.tenancy_ocid
  user_ocid        = var.user_ocid
  fingerprint      = var.fingerprint
  private_key_path = var.private_key_path
  region           = var.region
}
```

> [!TIP] O arquivo `provider.tf` é o arquivo que define o provedor que será usado para criar a infraestrutura na nuvem, nesse caso, a AWS. Este arquivo é uma convenção entre os desenvolvedores, ele é opcional, mas é uma boa prática ter ele.

4. Agora vamos criar o arquivo `network.tf`, incluindo esse trecho abaixo nele:

```
# Cria uma Virtual Cloud Network (VCN)
resource "oci_core_vcn" "vcn" {
  compartment_id = var.compartment_ocid
  display_name   = "web-vcn"
  cidr_block     = "10.0.0.0/16"
}

# Cria um Internet Gateway para permitir o acesso à internet
resource "oci_core_internet_gateway" "internet_gateway" {
  compartment_id = var.compartment_ocid
  vcn_id         = oci_core_vcn.vcn.id
  display_name   = "web-ig"
}

# Cria uma Tabela de Rotas para direcionar o tráfego para o Internet
```

```
Gateway
resource "oci_core_route_table" "route_table" {
  compartment_id = var.compartment_ocid
  vcn_id         = oci_core_vcn.vcn.id
  display_name   = "web-rt"

  route_rules {
    destination       = "0.0.0.0/0"
    destination_type  = "CIDR_BLOCK"
    network_entity_id = oci_core_internet_gateway.internet_gateway.id
  }
}

# Cria uma Subnet pública
resource "oci_core_subnet" "subnet" {
  compartment_id = var.compartment_ocid
  vcn_id         = oci_core_vcn.vcn.id
  display_name   = "web-subnet"
  cidr_block     = "10.0.1.0/24"
  route_table_id = oci_core_route_table.route_table.id
}
```

> [!TIP] O arquivo `network.tf` é o arquivo que define a criação de rede da OCI.

5. Agora vamos criar o arquivo `security_group.tf`, incluindo esse trecho abaixo nele:

```
# Cria um Network Security Group (NSG)
resource "oci_core_network_security_group" "web_nsg" {
  compartment_id = var.compartment_ocid
  vcn_id         = oci_core_vcn.vcn.id
  display_name   = "web-server-nsg"
}

# Regra de segurança para liberar a por`ta 80 (HTTP)
resource "oci_core_network_security_group_security_rule" "allow_http"
{
  network_security_group_id =
oci_core_network_security_group.web_nsg.id
  direction                 = "INGRESS"
  protocol                  = "6" # TCP
  source                    = "0.0.0.0/0"
  source_type               = "CIDR_BLOCK"

  tcp_options {
    destination_port_range {
      min = 80
      max = 80
    }
  }
}

# Regra de segurança para liberar a porta 22 (SSH)
```

```
resource "oci_core_network_security_group_security_rule" "allow_ssh" {
  network_security_group_id =
oci_core_network_security_group.web_nsg.id
  direction                 = "INGRESS"
  protocol                  = "6" # TCP
  source                    = var.meu_ip_publico
  source_type               = "CIDR_BLOCK"

  tcp_options {
    destination_port_range {
      min = 22
      max = 22
    }
  }
}

# Regra de segurança para liberar todo o tráfego de saída (Egress)
resource "oci_core_network_security_group_security_rule"
"allow_egress" {
  network_security_group_id =
oci_core_network_security_group.web_nsg.id
  direction                 = "EGRESS"
  protocol                  = "all"
  destination               = "0.0.0.0/0"
  destination_type          = "CIDR_BLOCK"
}
```

> [!TIP] O arquivo `security_group.tf` é o arquivo que define a criação dos security groups na OCI, ou seja, as regras de segurança para o acesso externo.

6. Agora vamos criar o arquivo `main.tf`, incluindo esse trecho abaixo nele:

```
# Obtém a lista de domínios de disponibilidade
data "oci_identity_availability_domains" "ads" {
  compartment_id = var.tenancy_ocid
}

# Obtém a imagem mais recente do Oracle Linux
data "oci_core_images" "oracle_linux" {
  compartment_id = var.compartment_ocid
  operating_system = "Oracle Linux"
  operating_system_version = "8"
  shape = "VM.Standard.E2.1.Micro" # Exemplo de shape, ajuste se
necessário
  sort_by = "TIMECREATED"
  sort_order = "DESC"
}

# Gera uma chave privada RSA de 4096 bits
resource "tls_private_key" "rsa_key" {
  algorithm = "RSA"
```

```
  rsa_bits  = 4096
}

# Salva a chave privada em um arquivo local
resource "local_file" "private_key_pem" {
    content  = tls_private_key.rsa_key.private_key_pem
    filename = "${path.module}/oci-instance-key.pem"
    file_permission = "0600"
}


# Cria a instância de computação
resource "oci_core_instance" "web_server" {
  availability_domain =
data.oci_identity_availability_domains.ads.availability_domains[0].nam
e
  compartment_id      = var.compartment_ocid
  display_name        = "WebServer-DVP"
  shape               = "VM.Standard.E2.1.Micro" # Shape "Always Free"

  create_vnic_details {
    subnet_id              = oci_core_subnet.subnet.id
    assign_public_ip       = true
    nsg_ids                =
[oci_core_network_security_group.web_nsg.id]
  }

  source_details {
    source_type = "image"
    source_id   = data.oci_core_images.oracle_linux.images[0].id
  }

  metadata = {
    ssh_authorized_keys = tls_private_key.rsa_key.public_key_openssh
    user_data           = base64encode(file("user_data.sh"))
  }

  tags = {
    Name = "WebServer-DVP"
  }
}
```

> [!TIP] O arquivo `ec2.tf` é o arquivo que define a criação da instância EC2 na AWS.

7. Agora vamos criar o arquivo `outputs.tf`, incluindo esse trecho abaixo nele:

```
# Bloco para obter o IP público da VNIC da instância
data "oci_core_vnic" "instance_vnic" {
  vnic_id = oci_core_instance.web_server.create_vnic_details[0].id
}
```

```
# Bloco para exibir o IP público da instância após a criação
output "instance_public_ip" {
  description = "IP público da instância"
  value       = data.oci_core_vnic.instance_vnic.public_ip_address
}

output "website_url" {
  description = "URL do site provisionado."
  value       =
"http://${data.oci_core_vnic.instance_vnic.public_ip_address}"
}
```

> [!TIP] O arquivo `outputs.tf` é o arquivo que define as saídas que serão exibidas após a criação da infraestrutura, nesse caso, o IP público da instância EC2.

8. Agora vamos criar o arquivo `variables.tf`, incluindo esse trecho abaixo nele:

```

variable "tenancy_ocid" {
  type        = string
  description = "O OCID da sua Tenancy na OCI."
}

variable "user_ocid" {
  type        = string
  description = "O OCID do seu usuário na OCI."
}

variable "fingerprint" {
  type        = string
  description = "O fingerprint da sua chave de API na OCI."
}

variable "private_key_path" {
  type        = string
  description = "O caminho para a sua chave privada de API da OCI."
}

variable "region" {
  type        = string
  description = "A região da OCI para provisionar os recursos."
  default     = "us-ashburn-1"
}

variable "compartment_ocid" {
  type        = string
  description = "O OCID do compartimento onde os recursos serão
criados."
}

variable "meu_ip_publico" {
  type        = string
```

```
    description = "Endereço IP público para o Security Group SSH"
    default     = "203.0.113.25/32" # IMPORTANTE: Substitua pelo seu
  endereço IP público
  }
```

> [!TIP] O arquivo `variables.tf` é o arquivo que define as variáveis que serão usadas na infraestrutura, nesse caso, o IP público para o Security Group SSH.

9. Precisamos definir o valor de algumas dessas variáveis, para isso criaremos um arquivo `terraform.tfvars` com os valores das variáveis.

```
tenancy_ocid     = "ocid1.tenancy.oc1..xxxx"
user_ocid        = "ocid1.user.oc1..xxxx"
fingerprint      = "xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx"
private_key_path = "/caminho/para/sua/chave_privada.pem"
compartment_ocid = "ocid1.compartment.oc1..xxxx"
meu_ip_publico   = "SEU_IP_PUBLICO/32"
```

10. Agora é hora de criar o playbook do Ansible para irá provisionar a página

```
---
- name: Configure Web Server
  hosts: all
  become: yes         # Necessário para instalar pacotes e gerenciar
serviços

  tasks:
    - name: Ensure all packages are up to date
      yum:
        name: '*'
        state: latest

    - name: Install Git
      yum:
        name: git
        state: present

    - name: Install Nginx on Amazon Linux 2
      command: amazon-linux-extras install -y nginx1
      args:
        creates: /usr/sbin/nginx

    - name: Ensure Nginx service is started and enabled
      service:
        name: nginx
        state: started
        enabled: yes

    - name: Clone website repository
```

```
      git:
        repo: 'https://github.com/avanti-dvp/site-exemplo-aws.git' # Pode
ser o mesmo repo ou outro
        dest: '/tmp/website'
        clone: yes

    - name: Deploy website files to Nginx document root
      copy:
        src: "/tmp/website/"
        dest: "/usr/share/nginx/html/"
        remote_src: yes # src e dest estão na mesma máquina
        owner: root
        group: nginx
        mode: '0755'
      notify:
      - restart nginx

  handlers:
    - name: restart nginx
      service:
        name: nginx
        state: restarted
```

12. Agora precisamos criar o inventário do Ansible para que ele possa acessar a instância EC2 que foi criada pelo Terraform.

```
touch inventory
echo "[all]" >> inventory
echo "ip_da_instancia_ec2 ansible_user=ec2-user
ansible_ssh_private_key_file=ec2-instance-key.pem" >> inventory
```

13. Boa! terminamos de criar todos os arquivos necessários para a criação da infraestrutura na nuvem.

14. Agora vamos iniciar o fluxo de trabalho do Terraform para criar a infraestrutura na nuvem:

```
terraform init
terraform plan
terraform apply
```

> [!NOTE] O comando `terraform init` inicializa o Terraform e baixa os providers necessários para a criação da infraestrutura na nuvem. O comando `terraform plan` cria um plano de execução que mostra as alterações que serão feitas na infraestrutura na nuvem. O comando `terraform apply` aplica as configurações definidas nos arquivos .tf e cria a infraestrutura na nuvem.

15. Agora vamos rodar o Ansible para configurar a instância da OCI:

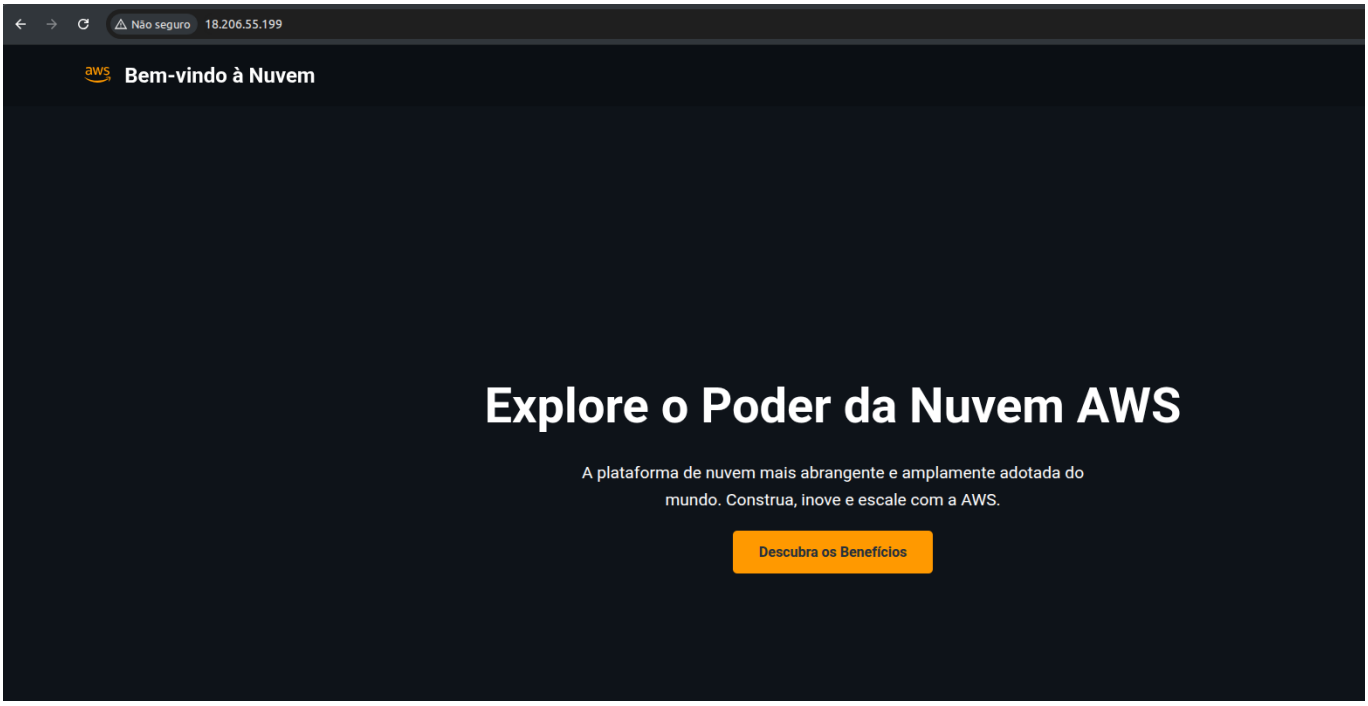
```
ansible-playbook -i inventory playbook.yml
```

> [!NOTE] O comando `ansible-playbook` executa o playbook definido no arquivo playbook.yml. O parâmetro `-i` especifica o arquivo de inventário que contém as informações de acesso à instância da OCI.

16. Se tudo rodar com sucesso, você verá o IP público da instância e a URL do site provisionado, basta acessá-lo através dessa URL no seu navegador para ver o site está no ar.

> [!WARNING] A maioria dos navegadores modernos força o redirecionamento da página para HTTPS Como não subimos o site em HTTPS, a conexão não irá acontecer Portanto, para ver o site funcionando, você precisa adicionar o http:// no começo da URL antes do IP na barra de endereço do seu navegador

E ele deverá aparecer dessa forma:



17. Para destruir a infraestrutura na nuvem, execute o comando abaixo:

```
terraform destroy
```

> [!NOTE] O comando `terraform destroy` destrói a infraestrutura na nuvem que foi criada pelo Terraform. **RECOMENDADO:** Sempre que você criar uma infraestrutura na nuvem, certifique-se de destruí-la quando não estiver mais usando.

Saiba mais

- Documentação do Terraform
- Documentação do Provider OCI do Terraform
- Lista de Providers do Terraform

- Documentação da OCI