**Desafio Bootcamp Avanti DevOps**

**Proposta do desafio:**

* **Criar uma infraestrutura com o Terraform na AWS;**
* **Criar um deploy com o GitLab na EC2;**
* **Criar uma documentação do passo a passo com evidências do processo;**

**Infraestrutura com o Terraform na AWS;**

1. Baixar o terraform no site terraform.io e instalar;
2. Criar uma pasta e colar o arquivo terraform.exe;
3. Para abrir o terraform em todos diretórios deixar o arquivo disponível em qualquer ambiente;
4. Eu preciso preencher uma variável de ambiente aqui no Windows path;
5. Botão iniciar > Editar variáveis de ambiente > variáveis de ambiente> path> editar

Criar novo > colocar o caminho da pasta > OK

1. Para usar o terraform você precisa de permissão para ter acesso ao painel de controle da AWS e precisa ter instalado ferramenta de linha de comando da AWS (AWS CLI)

<https://awscli.amazonaws.com/AWSCLIV2.msi>

1. Criar um grupo no IAM e vincular uma política de acesso ao grupo ;
2. Criar um usuário, deixar acesso somente AWS CLI > next > vincular ao grupo criado> Criar usuário
3. Criar chave de acesso com acesso ao AWS CLI e armazenar Access Key e Secret Access Key
4. Configurar CLI (abrir terminal)
5. >aws configure

>colocar access key >colocar secret Key

>region >formato de saída pode deixar vazio dando enter

1. Criando a máquina virtual com terraform

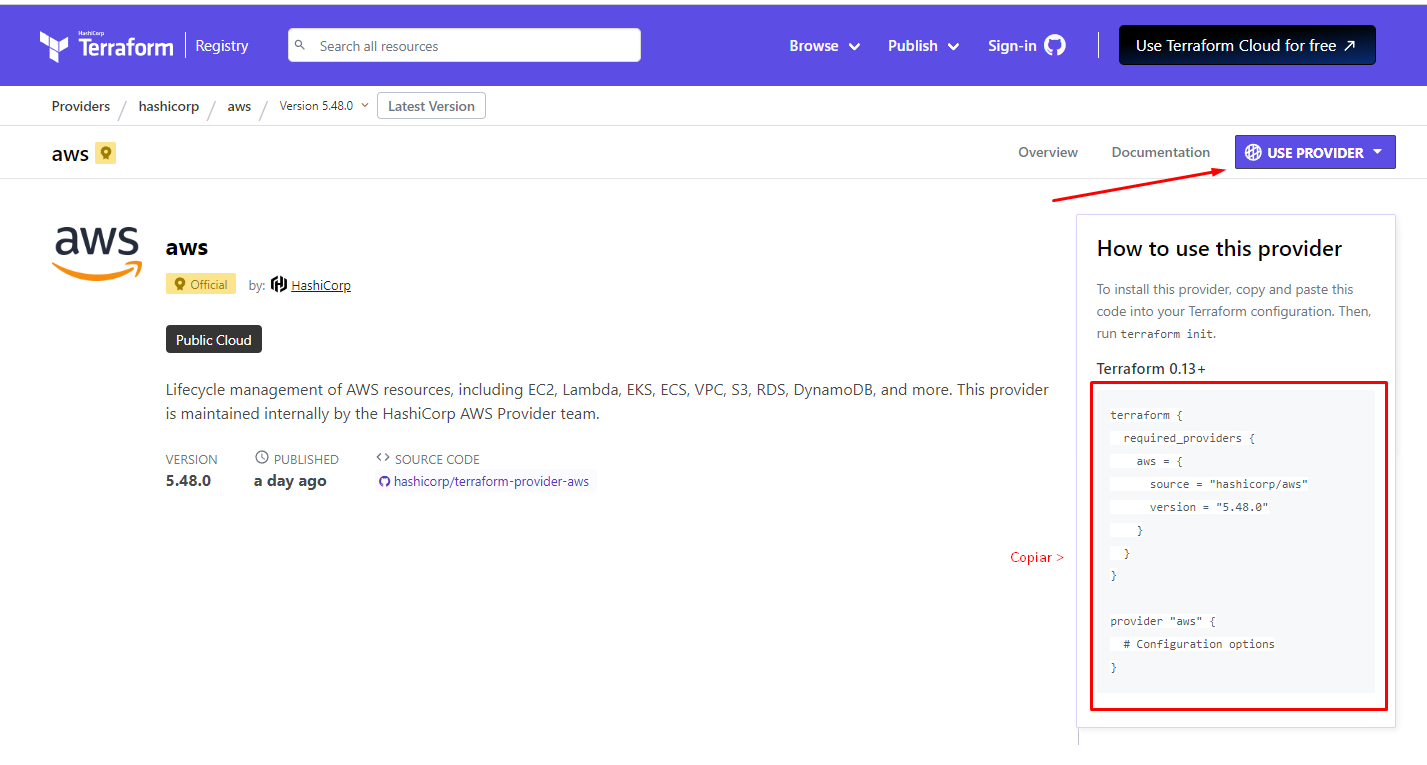
>cria uma pasta e abre um terminal

>abrir o vscode digitando code .

>no vscode instalar a extensão do terraform hashicorp

>Entrar na documentação do terraform <https://registry.terraform.io/browse/providers>

> escolha o provedor que irá utilizar no caso AWS



terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "5.48.0"

}

}

}

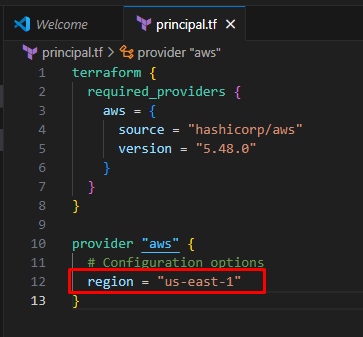
provider "aws" {

# Configuration options

}

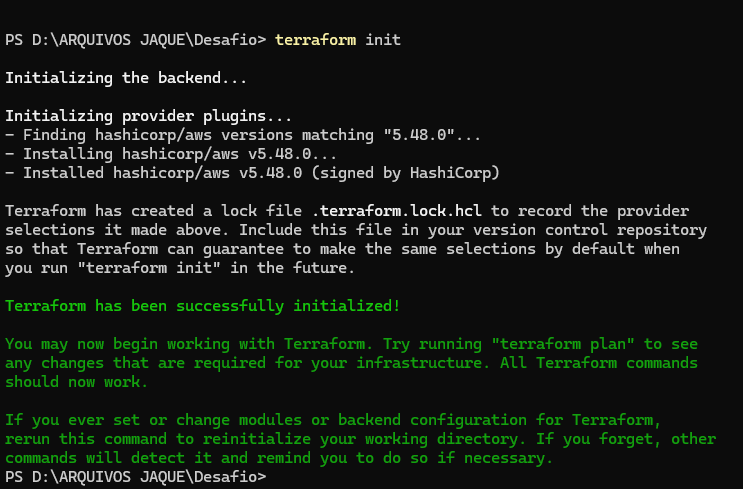
1. Criar um arquivo o vscode com extensão .tf
2. main.tf onde vou colar o provider dizendo que vou usar a AWS

Add no código a região onde irá criar o ambiente



1. Voltando no terminal posso dar um ls para verificar que o arquivo com o provider já está criado
2. Agora para inicializar

>Terraform init



1. Criar um novo arquivo no vscode chamado EC2 onde vamos colocar o script para criar a instância

>usar a documentação do terraform para facilitar

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance>

resource "aws\_instance" "web" {

ami = data.aws\_ami.ubuntu.id

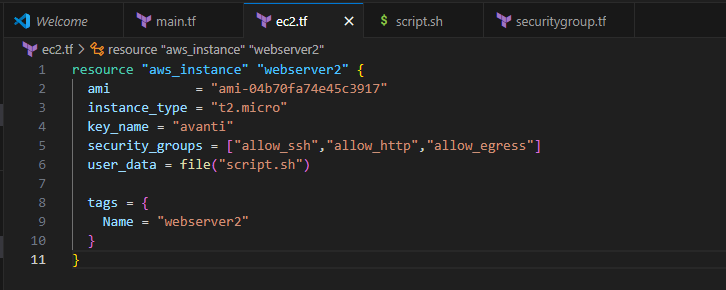
instance\_type = "t3.micro"

tags = {

Name = "HelloWorld"

}

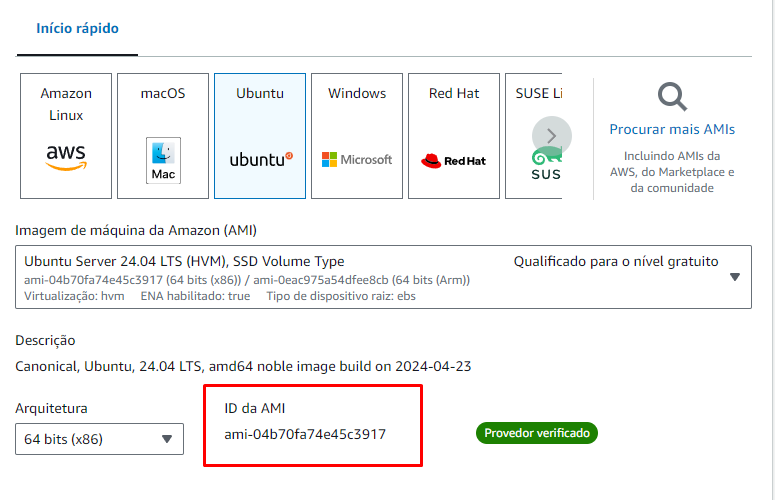
}



Lista de ID de images

<https://cloud-images.ubuntu.com/locator/ec2/>

ou copia o ID no console da AWS e o ID muda de acordo com a região escolhida



1. Incluir no script o nome da instância > ID da image > o tipo da instância >nome da Key

>tag da máquina

> criar um novo arquivo security group

>como criar security group documentação <https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/security_group>

chamar o security group no script

> e precisa incluir no script a criação dos security group para que podemos acessar a máquina porta SSH, HTTP e o que mais desejar liberar

resource "aws\_security\_group" "allow\_ssh" {

name        = "allow\_ssh"

description = "Allow ssh inbound traffic"

ingress {

description      = "ssh from VPC"

from\_port        = 22

to\_port          = 22

protocol         = "tcp"

cidr\_blocks      = ["0.0.0.0/0"]

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

}

tags = {

Name = "allow\_ssh"

}

}

resource "aws\_security\_group" "allow\_http" {

name = "allow\_http"

description = "Allow http inbound traffic"

ingress {

description = "http from VPC"

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

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

}

tags = {

Name = "allow\_http"

}

}

resource "aws\_security\_group" "allow\_egress" {

name = "allow\_egress"

description = "Allow egress traffic"

egress {

    from\_port = 0

    to\_port = 0

    protocol = "-1"

    cidr\_blocks= ["0.0.0.0/0"]

    ipv6\_cidr\_blocks = ["::/0"]

}

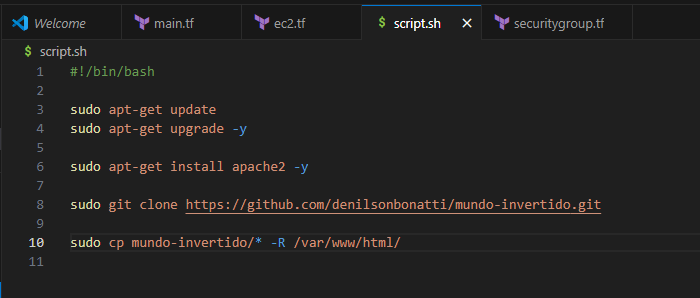
tags = {

Name = "allow\_egress"

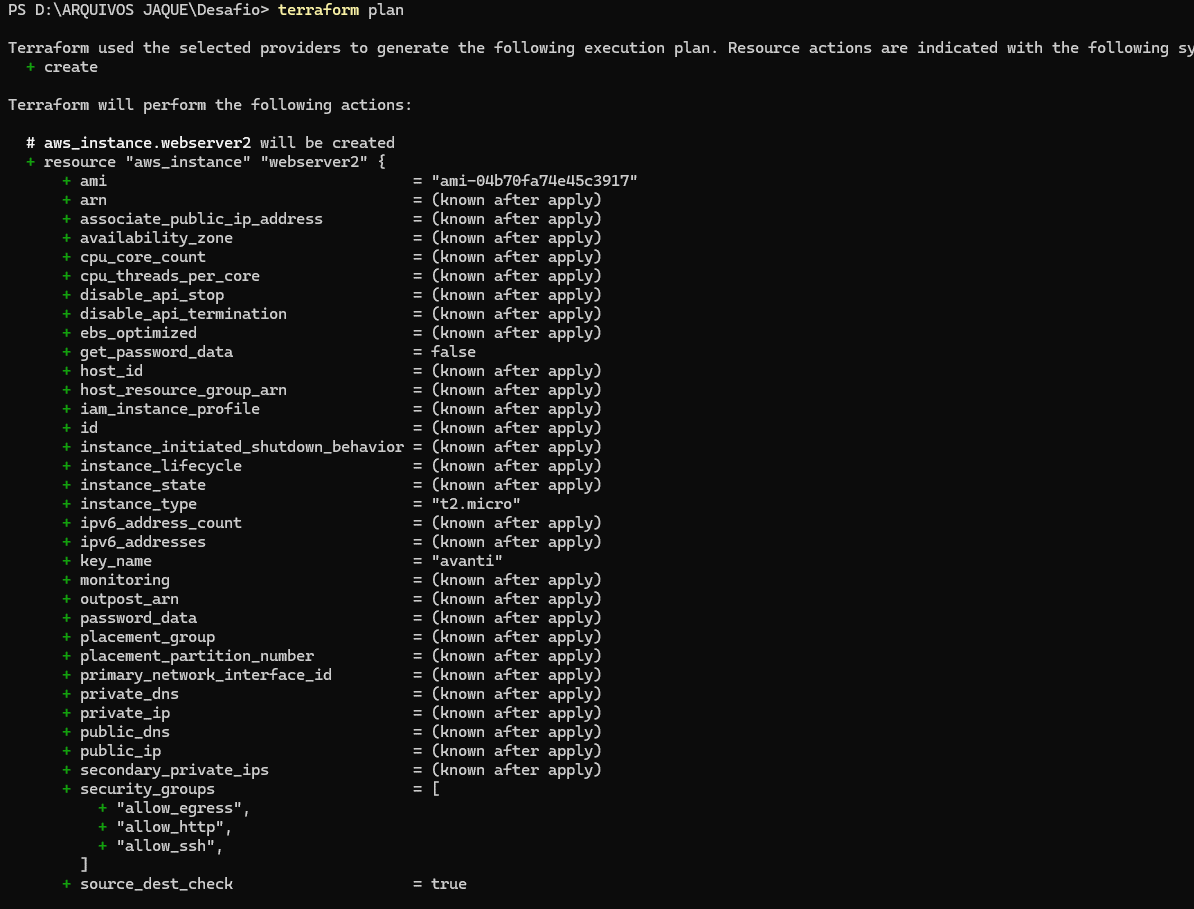
}

}

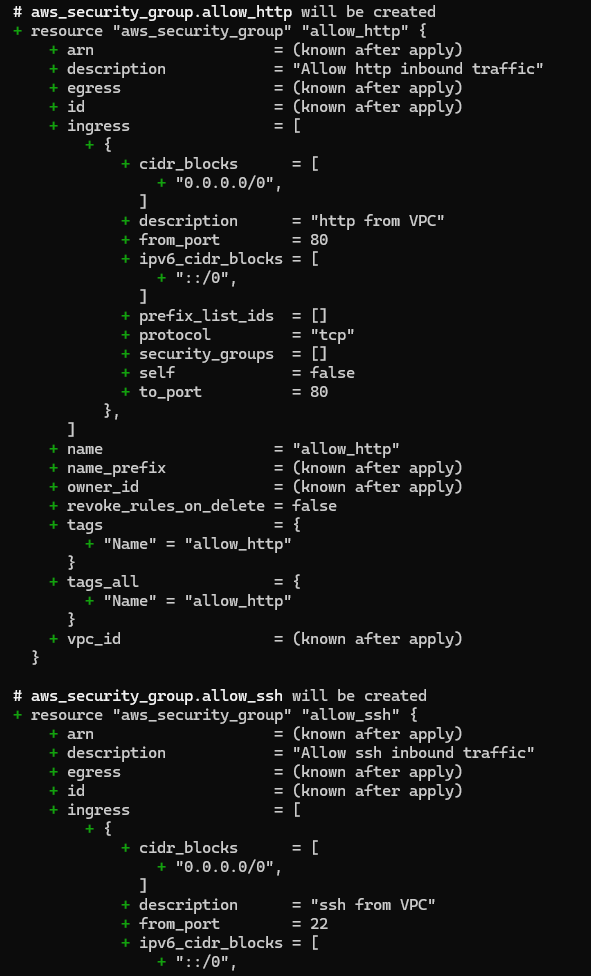
1. Para a máquina quando for criada atualizar, instalar o apache e clonar o repositório, para isso criar um novo arquivo script.sh

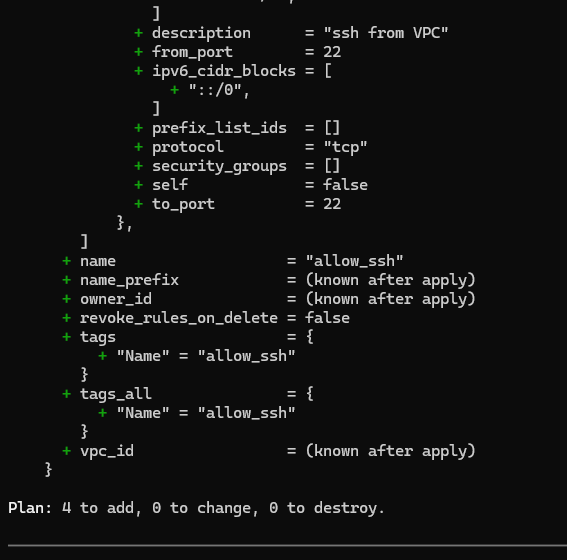


1. Agora eu vou no terminal e digito terraform plan (para verficar o que vai criar) > yes

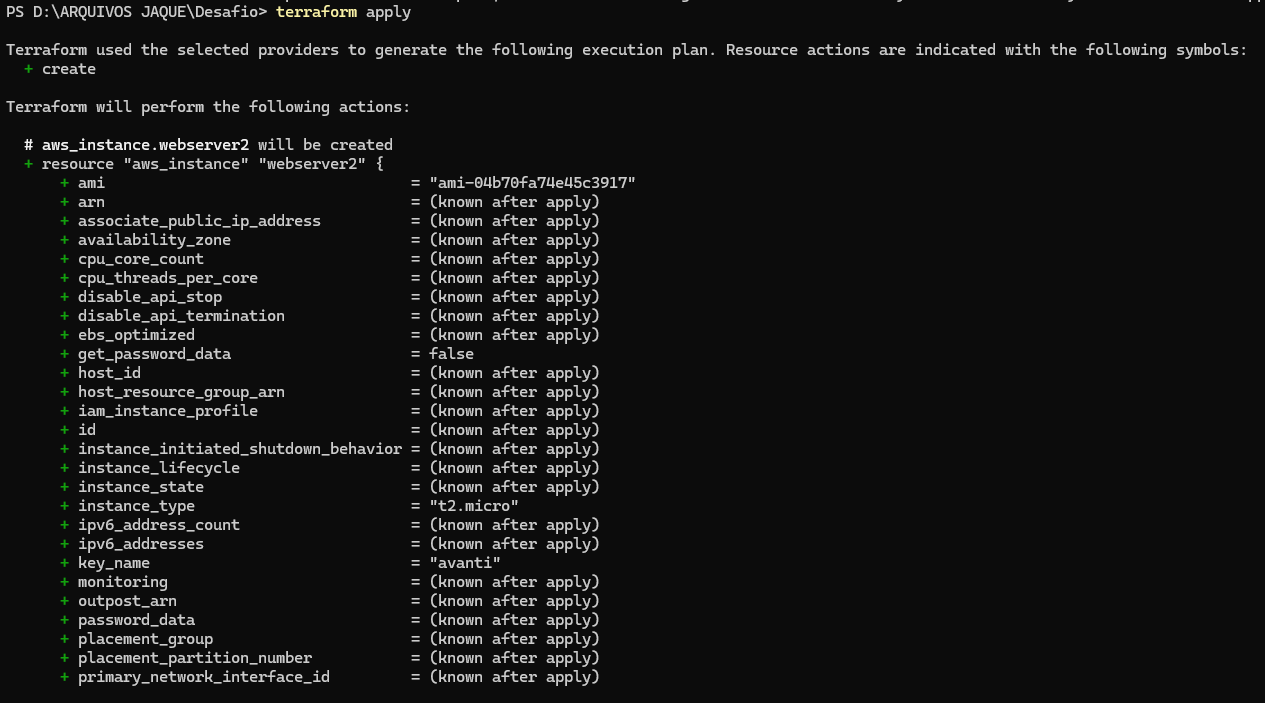


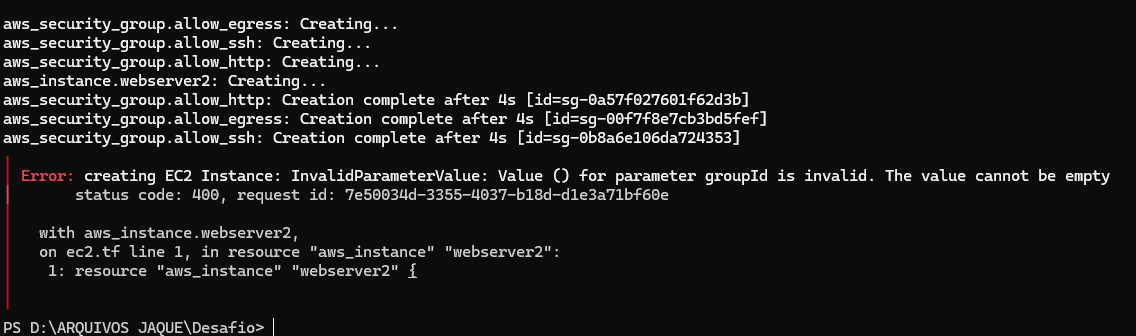




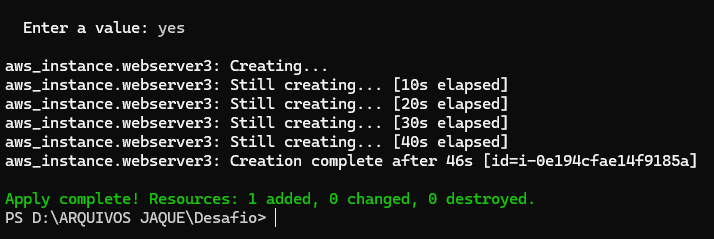


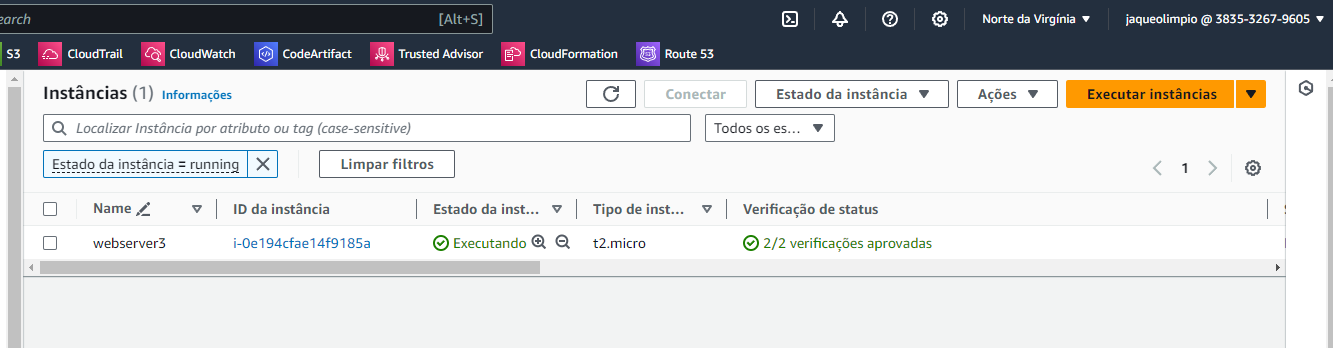
1. Agora eu vou no terminal e digito terraform apply (para criar) > yes





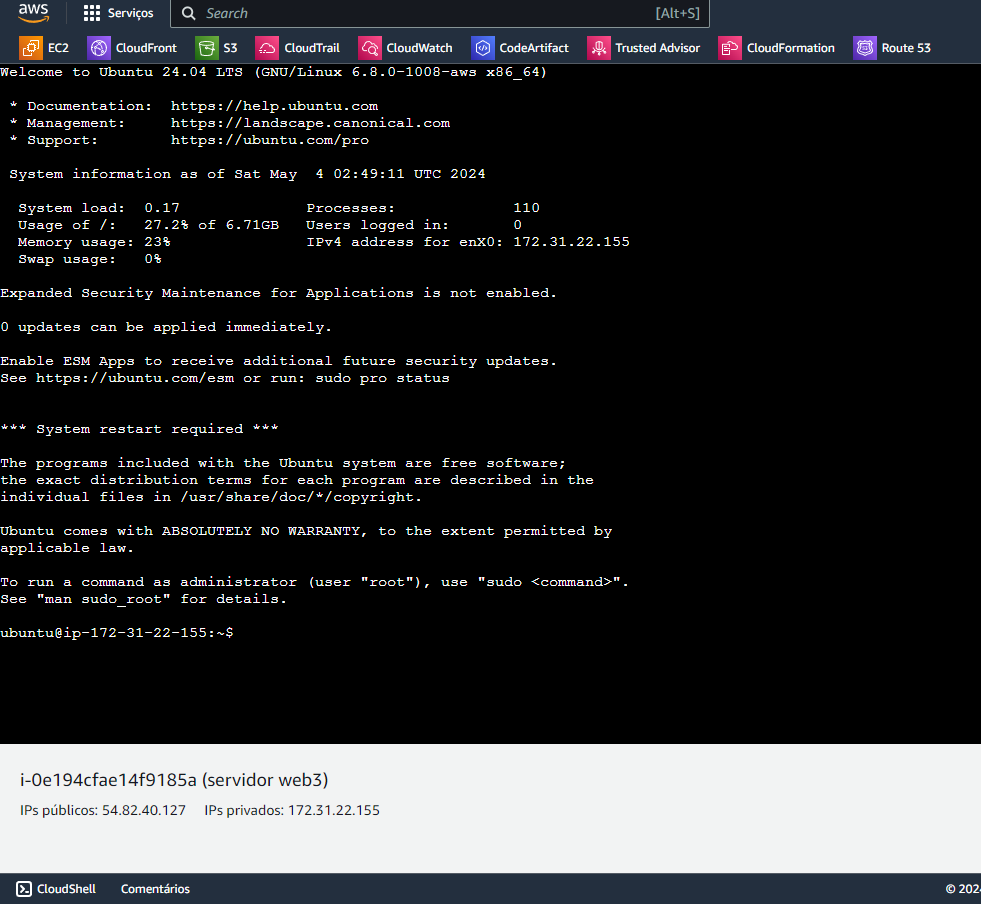
Criou todos recursos exceto a instância deu erro , corrige e dei terraform apply novamente



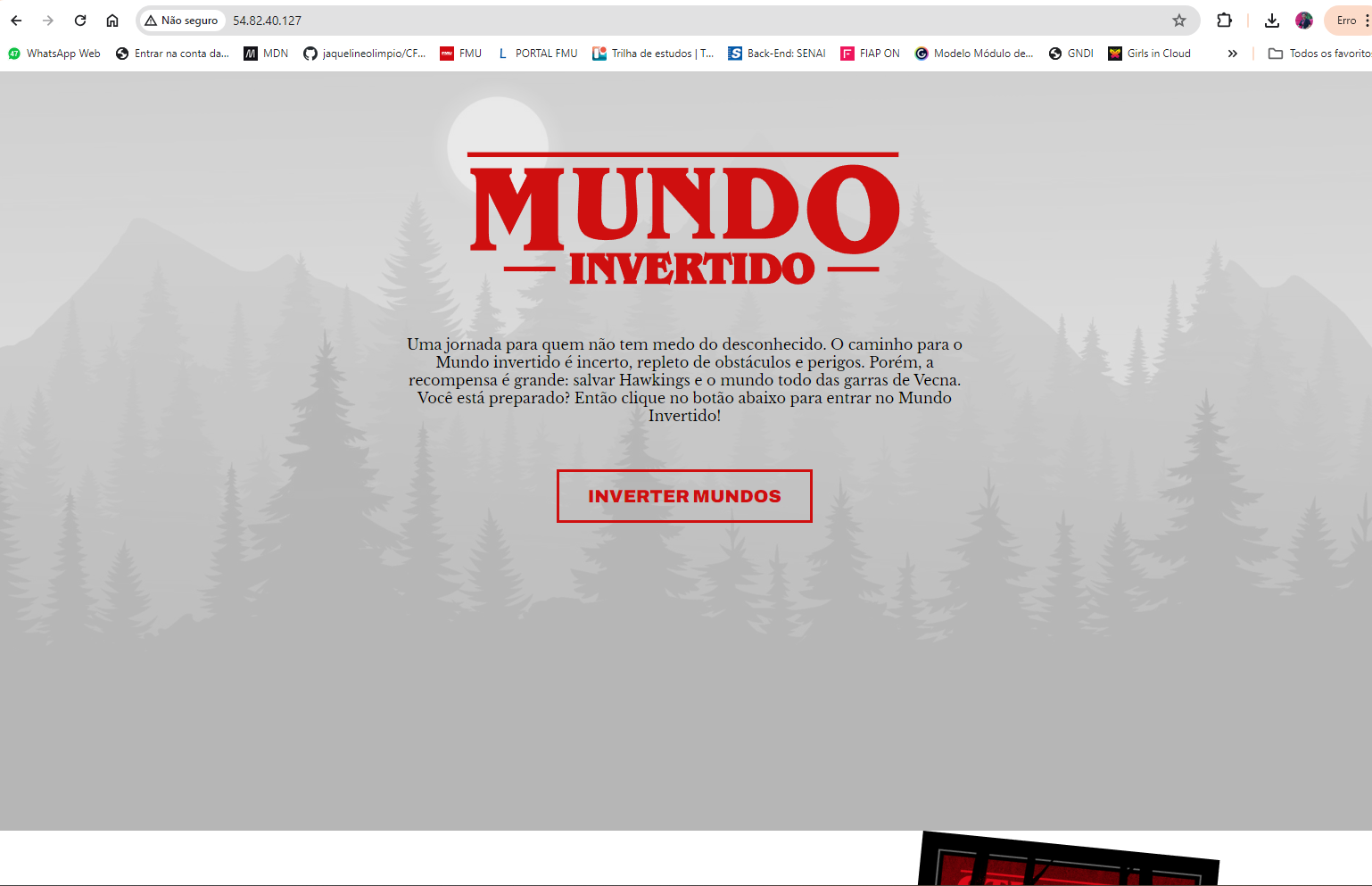


Instância criada

4 – Configuração do gitilab-runner:



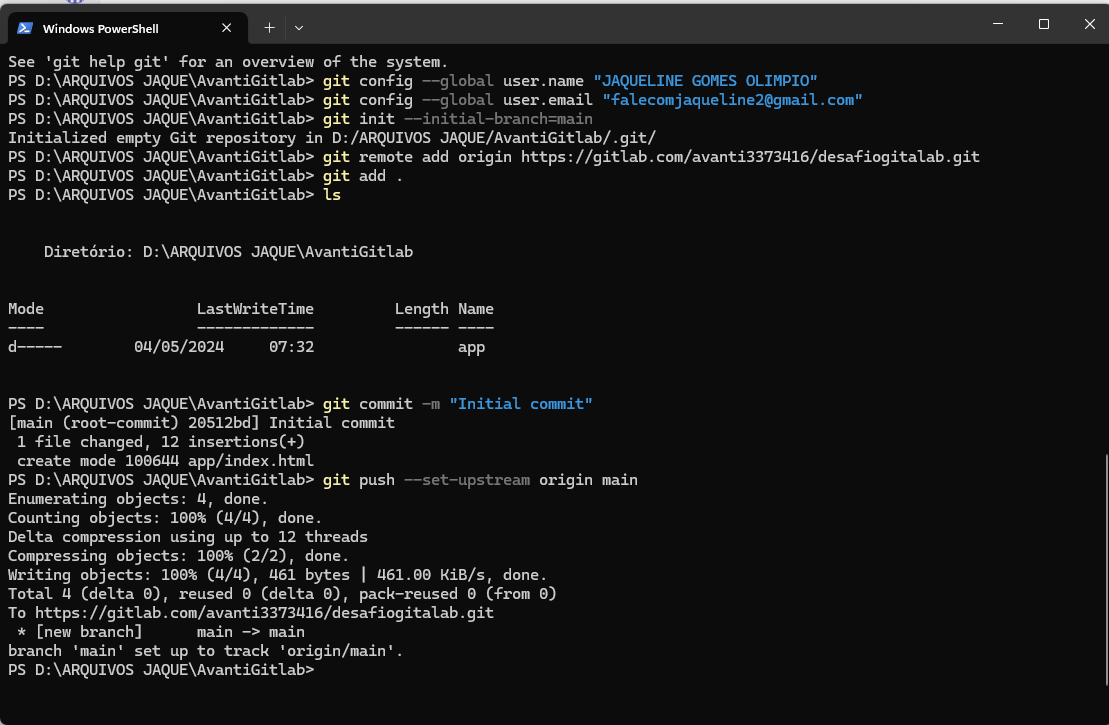
Conexão estabelecida



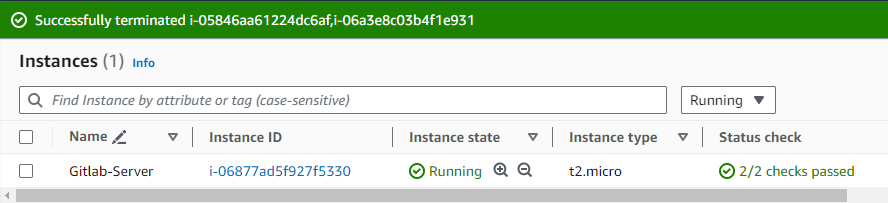
Site no ar 😊

**Deploy com o GitLab na EC2**

1. Criar um novo repositório no GitLab
2. Configuração



1. Criar uma máquina virtual na AWS



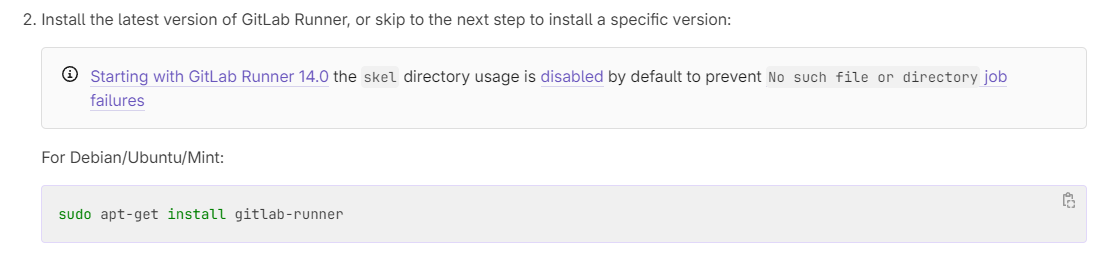
Conectar na máquina – realizar as atualizações e instalação do apache

* Instalar Gitlab runner

<https://docs.gitlab.com/runner/install/>



curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh" | sudo bash



sudo apt-get install gitlab-runner

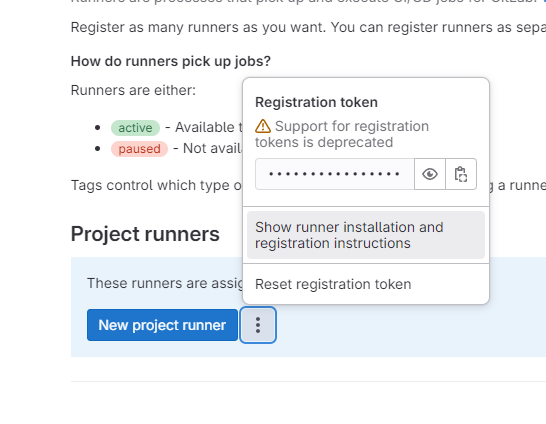
para verificar o usuário criado

cat /etc/ passwd

Dar a devida permissão para o usuário gitlab runner

1. Relacionar a Máquina Virtual com o GitLab (Criar CI/CD)

Setting > CI/CD > Runners





sudo gitlab-runner register --url https://gitlab.com/ --registration-token GR1348941k4sQV8dJHcGkoWetirNx

colocar esse comando no servidor > enter para confirmar > token já está no comando então só dar enter novamente > colocar descrição > tag > mensagem opcional > executor, permissões shell por padrão

