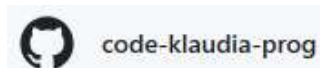


Provisionamento de Infraestrutura AWS em

Terraform + HCP + GitHub!



O código fonte declarativo escrito em *hcl* pode ser acedido publicamente no [repositório pessoal](#) do GitHub :



No ficheiro [main.tf](#) encontra-se um módulo com a **definição da VPC** :

```
name = "vpc-avancada-tf"
```

```
cidr = "10.0.0.0/16"
```

bem como os **CIDRs** para a definição da **Subnet Pública** :

```
public_subnets = ["10.0.1.0/24", "10.0.3.0/24"]
```




e da **Subnet Privada** :

```
private_subnets = ["10.0.2.0/24", "10.0.4.0/24"]
```



Workflow

- 1 Escrever localmente o código Terraform que declara os recursos pedidos
 - 2 Push do código para o repositório do [GitHub](#)
 - 3 Conectar o GitHub com o HashiCorp Cloud Platform (HCP Terraform)
 - 4 Applied infrastructure directly to AWS Cloud
-  Serviços: VPC e respectivas subnets publicas e privadas, EC2 com um Security Group, NAT Gateway e Internet Gateway



Uma vez configurado corretamente o repositório do [GitHub](#) :

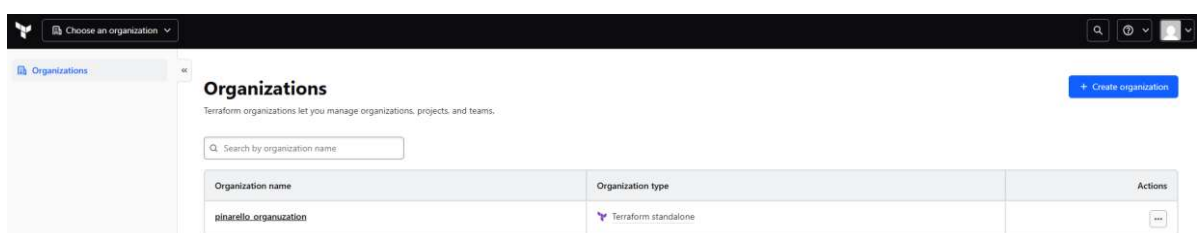
pinarello_organization / Workspaces / _task / Settings / Version Control

Edit Version Control

✓ Connected to VCS

Provider	GitHub
Repository	code-klaudia-prog/load_balancers

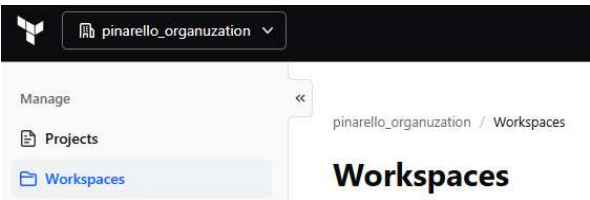
a **Organization** :



The screenshot shows the Terraform Organizations web interface. At the top, there's a header with a search bar and a 'Choose an organization' dropdown. Below the header, the main content area is titled 'Organizations' and includes a sub-header 'Terraform organizations let you manage organizations, projects, and teams.' There's a search bar for organization names. A table lists the organizations, with one entry visible: 'pinarello_organization' of type 'Terraform standalone'. A '+ Create organization' button is in the top right corner.

Organization name	Organization type	Actions
pinarello_organization	Terraform standalone	...

e o **Workspace** do **TerraformCloud** :



com as variáveis de ambiente que permitem a conexão ao provedor de serviços (neste caso, a **AWS**)

Workspace variables

Variables defined within a workspace always overwrite variables from variable sets that have the same type and the same key. Learn more about variable set [precedence](#).

Key	Value	Category	Actions
AWS_ACCESS_KEY_ID	AKIA5TEFS0MR027QZ0A	env	...
AWS_SECRET_ACCESS_KEY Sensitive	Sensitive - write only	env	...

foi possível automatizar o *deployment* para a **AWS** dos recursos descritos

 **Triggered via UI** ✓ Applied

#run-BFEIXMcFC9Ko9fgZ | pinarello_workspace triggered via UI | Branch main | 4a7fed8

20 hours ago

e confirmar o sucesso da operação na consola da **AWS** :

aws

Search

[Alt+S]

United States (N. Virginia)

Account ID: 8082-4360-2658

mywhoosh

VPC > Subnets

Subnets (7) Info

Last updated 1 minute ago

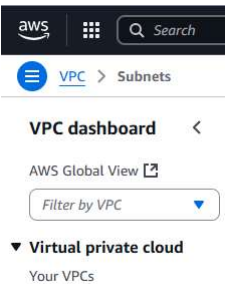
Actions

Create subnet

Find subnets by attribute or tag

<input type="checkbox"/>	Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR	IPv6 CIDR
<input type="checkbox"/>	vpc-avancada-tf-public-us-east-1a	subnet-07f0aaa3d19313980	Available	vpc-0cf5a5c7a5030068f vpc-a...	Off	10.0.1.0/24	-
<input type="checkbox"/>	vpc-avancada-tf-private-us-east-1a	subnet-08e0f231c94d181b7	Available	vpc-0cf5a5c7a5030068f vpc-a...	Off	10.0.2.0/24	-
<input type="checkbox"/>	vpc-avancada-tf-public-us-east-1b	subnet-0e8280d1b860487a8	Available	vpc-0cf5a5c7a5030068f vpc-a...	Off	10.0.3.0/24	-
<input type="checkbox"/>	vpc-avancada-tf-private-us-east-1b	subnet-0342db2e334a28875	Available	vpc-0cf5a5c7a5030068f vpc-a...	Off	10.0.4.0/24	-

acedendo ao menu **VPC » Subnets** :



Conseguimos assim confirmar a criação das 4 *subnets* descritas:

Subnets (7) Info		
<input type="text" value="Find subnets by attribute or tag"/>		
<input type="checkbox"/>	Name	Subnet ID
<input type="checkbox"/>	-	subnet-0487913cad0c9a0a9
<input type="checkbox"/>	vpc-avancada-tf-public-us-east-1a	subnet-07f0aaa3d19313980
<input type="checkbox"/>	vpc-avancada-tf-private-us-east-1a	subnet-08e0f231c94d181b7
<input type="checkbox"/>	vpc-avancada-tf-public-us-east-1b	subnet-0e8280d1b860487a8
<input type="checkbox"/>	vpc-avancada-tf-private-us-east-1b	subnet-0342db2e334a28875

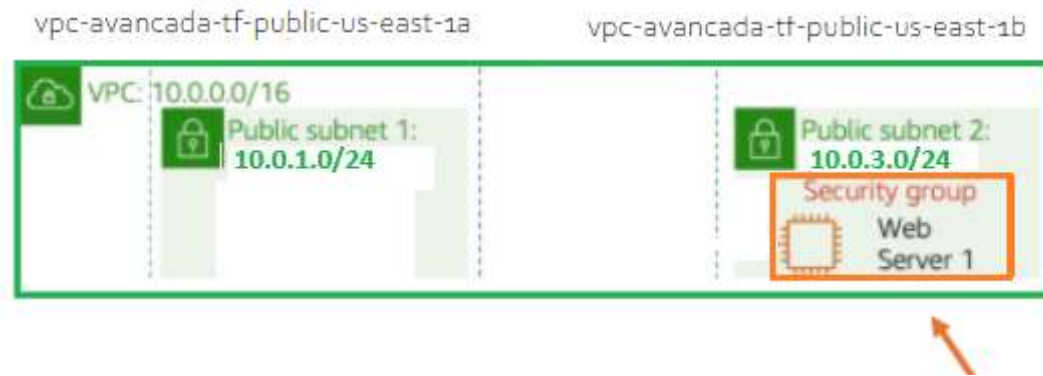
Tratam-se de duas subnets publicas e duas subnets privadas na mesa região, de North Virginia, mas em Availability Zones distintas :

vpc-avancada-tf-private-us-east-1a
vpc-avancada-tf-private-us-east-1b

vpc-avancada-tf-public-us-east-1b
vpc-avancada-tf-public-us-east-1a

Uma vez que os recursos da **subnet privada** são, inerentemente, **privados**, **não é possível** a sua comunicação direta com a **internet**

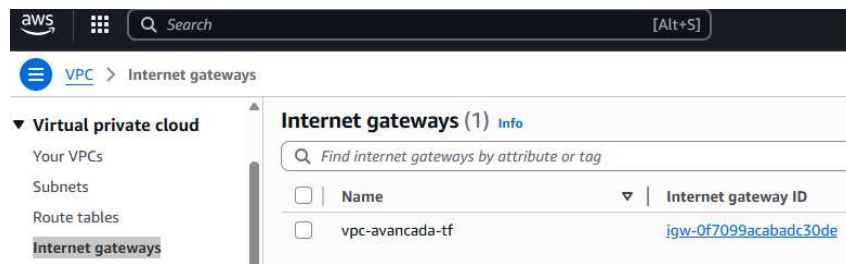
Para esse efeito, é usado para o efeito um **Bastion Host** ou Jump Server *deployed* na **subnet publica** para servir como ponte dessa comunicação:



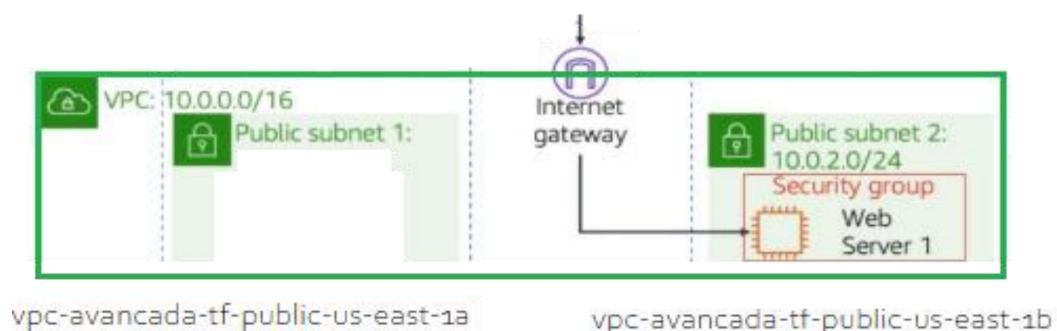
No **passo 3** do exercício proposto, foi feito o deploy de uma **Internet Gateway** com o id [igw-0f7099acabadc30de](#)



Confirmação do Deployment :



O nome deste serviço 🖱️ **Internet Gateway** , é intuitivo- E como o nome sugere, o seu proposito é encaminhar todo o tráfego de dentro da Infraestrutura da **AWS** para fora. Ou seja, para a **Internet**.

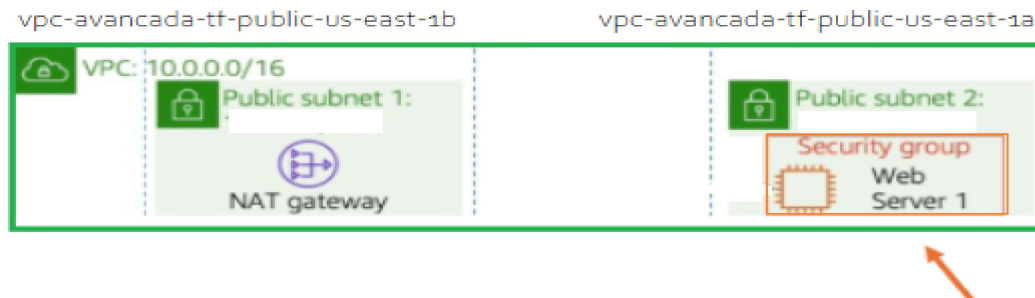


Ainda dentro do mesmo módulo de Terraform, é declarada uma **NAT Gateway** através do atributo :

```
enable_nat_gateway = true
```

E para garantir que é criada uma NAT Gateway **por Availability Zone** , (us-east-1a e us-east-1b) é usado o atributo:

`single_nat_gateway = false`



Isto tratasse de um mecanismo de **Fault Tolerance** e que garante a alta disponibilidade das instâncias.

Tal como no *deployment* das subnets, também aqui pode ser verificado o sucesso do *deployment* das NAT Gateways na consola da **AWS** :

A captura de tela da console da AWS mostra a página de NAT Gateways. A tabela abaixo resume as informações das duas gateways criadas:

Name	NAT gateway ID	Connectivity...	State	State message	Primary public I...	Primary private I...	Primary network...
vpc-avancada-tf-us-east-1a	nat-002a160ba031d9c5d	Public	Available	-	52.45.77.230	10.0.1.220	eni-05e9f23c0695f1...
vpc-avancada-tf-us-east-1b	nat-07bcff461c6bc7b01	Public	Available	-	44.215.246.34	10.0.3.7	eni-04feba2966296...

Abaixo da tabela, são exibidas as tabelas de roteamento para a VPC 10.0.0.0/16:

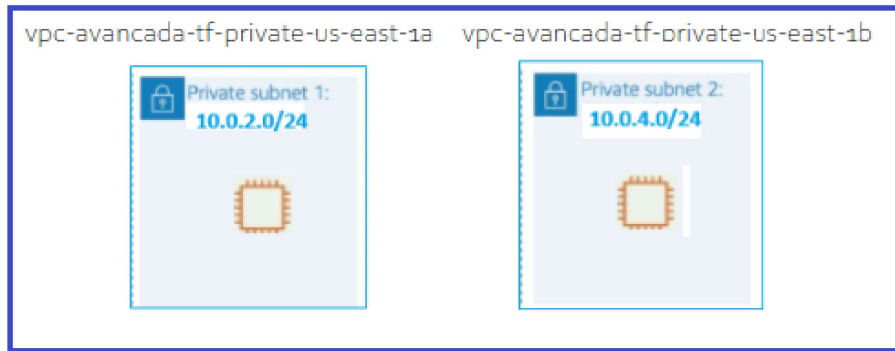
Internet Gateway

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	Internet Gateway

Private Route Table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	NAT Gateway

Para testar esse roteamento e confirmar que a saída para a internet é, efetivamente, feita por via da **NAT Gateway** que se encontra na **subnet pública**, é necessário que uma instância **EC2** se os seus respetivos **Security Groups** sejam *deployed* numa das **subnets privadas** ou na sub-rede 10.0.2.0/24 ou na sub-rede 10.0.4.0/24 :



Aqui é importante determinar qual a **imagem**, ou **ami** correta que existe na zona onde pretendemos fazer o *deployment* :

aws Search [Alt+S] United States (N. Virginia)

EC2 > Instances > Launch an instance

▼ Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't Browse more AMIs.

Search our full catalog including 1000s of application and OS images

Recents Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI
ami-052064a798f08f0d3 (64-bit (x86), uefi-preferred) / ami-089f6a79b0e02648a
Virtualization: hvm ENA enabled: true Root device type: ebs Free tier eligible

Description

Amazon Linux 2023 (kernel-6.1) is a modern, general purpose Linux-based OS to provide a secure, stable and high-performance execution environment to develop your cloud applications.

Amazon Linux 2023 AMI 2023.9.20250929.0 x86_64 HVM kernel-6.1

Architecture Boot mode AMI ID Publish Date

64-bit (x86) uefi-preferred ami-052064a798f08f0d3 2025-09-25

Por uma que Terraform :

```

103 resource "aws_instance" "private_test_instance" {
104     ami           = "ami-052064a798f08f0d3"
105     instance_type = "t3.micro"
106     # key_name      = aws_key_pair.deployer.key_name
107     vpc_security_group_ids = [aws_security_group.private_sg.id]
108     # Desliga a atribuição automática de IP público (característica da subnet privada)
109     associate_public_ip_address = true
110
111     tags = {
112         Name = "Private Test Instance via NAT GW"
113     }
114 }

```

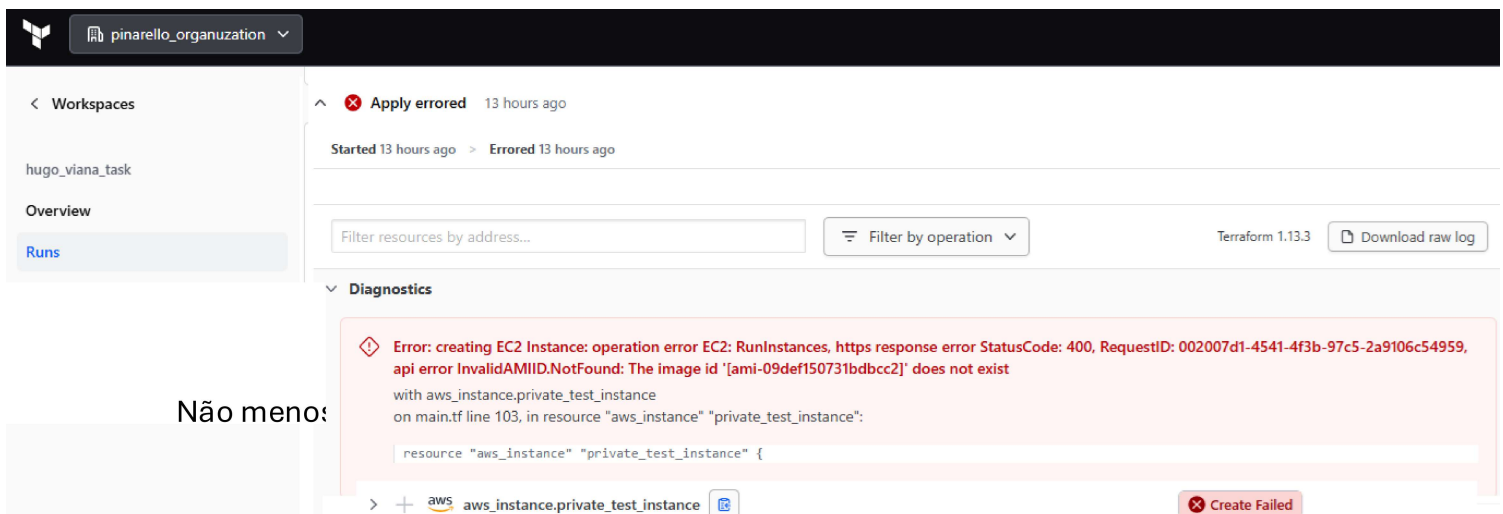

embora esta não seja uma prática de todo recomendada, uma vez que estas variáveis devem constar de um ficheiro [variables.tf](#) como o que acontece em outros [projetos](#).

Caso esta verificação do id da ami não seja feita e não seja garantida a existência da *ami* na região onde esta a ser feito o *deploy*, o Terraform poderá completar o *planning* com **sucesso**:



mas terminara o *apply* com uma mensagem de erro :

The image id '[ami-09def150731bdbcc2]' does not exist



e das respectivas **Inbound** e **Outbound** rules que permitirão o acesso SSH do exterior para o interior da VPC através do the Bastion Host :

```
ingress {
  description = "Allow SSH from within VPC CIDR"
  from_port   = 22
  to_port     = 22
  protocol    = "tcp"
  # Allows SSH acces by any resource within the VPC (10.0.0.0/16)
  cidr_blocks = [module.vpc.vpc_cidr_block]
```


Neste Security Group declarado em cima não foi configurado o **ICMP**, pelo que não foi possível efetuar um **ping** entre as instancias que se situa em subnets diferentes.

Ao estabelecer a associação entre um determinado Security Group e uma instancia EC2, é também importante considerar que o Security Group referenciado na declaração da instancia terá de estar na mesma subnet onde esta reside a instancia.

De outro forma, e a semelhança do que aconteceu na declaração de uma ami errada, o *plan* terminara com sucesso mas o **apply** ce interrompido com uma mensagem de erro :

Security group sg-00e3718d8e7e51390 and subnet subnet-0c579cdd17a50701c belong to different networks.

▼ Diagnostics

❗

Error: creating EC2 Instance: operation error EC2: RunInstances, https response error StatusCode: 400, RequestID: c0747ab1-e0c2-4691-a631-7ad0ae6497ff, api error InvalidParameter: Security group sg-00e3718d8e7e51390 and subnet subnet-0c579cdd17a50701c belong to different networks.
with aws_instance.private_test_instance
on main.tf line 103, in resource "aws_instance" "private_test_instance":

```
resource "aws_instance" "private_test_instance" {
```

apenas um recurso, como foi o caso:

```

72     # Create a Security Group for the private EC2 instance
73
74     resource "aws_security_group" "private_sg" {
75         name           = "private-instance-sg"
76         description    = "Security group for private instances"
77         vpc_id         = module.vpc.vpc_id
78
79         # Inbound Rule - Allows SSH connection from the outside into the VPC through the Bastion Host
80         ingress {
81             description = "Allow SSH from within VPC CIDR"
82             from_port   = 22
83             to_port     = 22
84             protocol    = "tcp"
85             # Allows SSH access by any resource within the VPC (10.0.0.0/16)
86             cidr_blocks = [module.vpc.vpc_cidr_block]
87         }
88
89         # Outbound Rule - This is what allows the NAT Gateway test
90         # It allows all outbound traffic (routed through the NAT Gateway)
91         egress {
92             from_port   = 0
93             to_port     = 0
94             protocol    = "-1"
95             cidr_blocks = ["0.0.0.0/0"]
96         }
97
98         tags = {
99             Name = "PrivateSG"
100         }
101     }

```

No entanto, e a semelhança de outros projetos, [Projetos](#), o *deployment* poderá ser feito com um recurso que provisiona o Security Group em si :

```

45     resource "aws_security_group" "ssh_access"

```

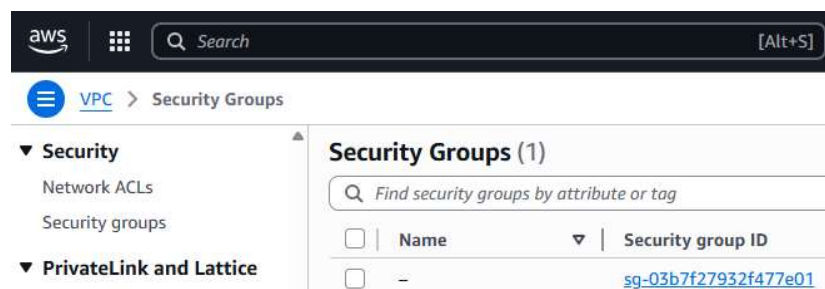
e um ou mais recursos que definem as Inbound e Outbound rules, desde que se referencie o **Security Group** ao qual pertencem :

```

53 resource "aws_security_group_rule" "example" {
54     type           = "ingress"
55     from_port      = 22
56     to_port        = 22
57     protocol        = "tcp"
58     cidr_blocks     = [aws_vpc.my_custom_vpc.cidr_block]
59     security_group_id = aws_security_group.ssh_access.id
60 }

```

O sucesso do *deploy* é confirmado acedendo ao menu **VPC » Security** na consola da **AWS**:



IAM acesso na conta route do billing

Diagrama Final da Infraestrutura AWS provisionada pelo Terraform :

