



SÃO  
PAULO  
TECH  
SCHOOL

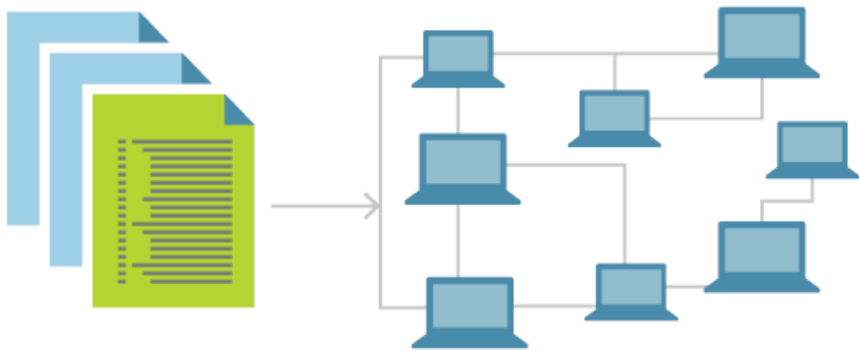
# **Arquitetura de soluções em nuvem**

## **Infraestrutura como código [IaC]**

**Eduardo Verri**

`eduardo.verri@sptech.school`

# O que é IaC?



- ❑ A infraestrutura como código (IaC) usa a metodologia de DevOps e o controle de versão com um modelo descritivo para definir e implantar a infraestrutura, como redes, máquinas virtuais, balanceadores de carga e topologias de conexão.
- ❑ Assim como o mesmo código-fonte sempre gera o mesmo binário, um modelo IaC gera o mesmo ambiente sempre que ele é implantado.
- ❑ É uma prática chave e um componente de entrega contínua. Com a IaC, as equipes de DevOps podem trabalhar com um conjunto unificado de práticas e ferramentas para fornecer aplicativos e sua infraestrutura de suporte de forma rápida e confiável em escala.

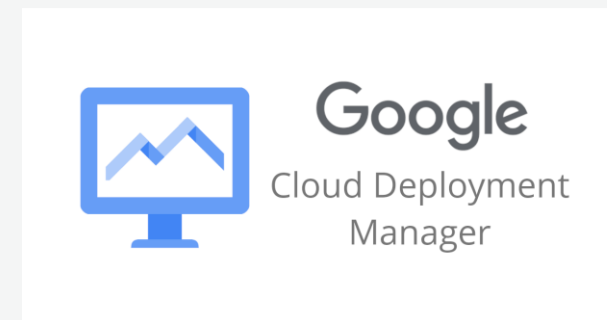
# Benefícios da IaC

- ❑ **Duplique facilmente um ambiente:** O mesmo ambiente pode ser implantado em um sistema diferente em outro local usando a mesma IaC, desde que os recursos de infraestrutura estejam disponíveis.
- ❑ **Reduza os erros de configuração:** A configuração manual está sujeita a erros devido ao envolvimento humano. As pessoas cometem erros. Ou pode haver um desvio de configuração devido a alterações em uma configuração (como um ambiente de desenvolvedor) que foram perdidas em outra configuração (como um ambiente de teste).
- ❑ **Faça iterações em ambientes de melhores práticas:** O controle de origem permite que os desenvolvedores de software criem e se ramifiquem facilmente em ambientes.

# Qual é o papel da IaC no DevOps?

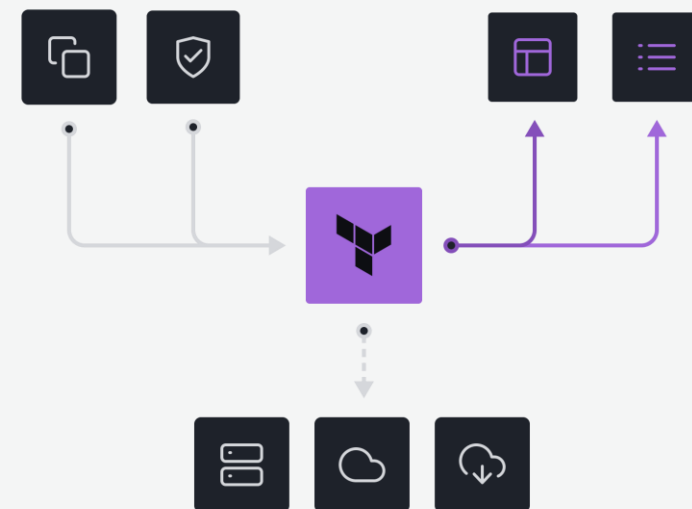
- ❑ Um dos principais objetivos do DevOps é automatizar as tarefas de infraestrutura em todo o processo de desenvolvimento. Você pode integrar a infraestrutura como código (IaC) em pipelines de integração contínua e implantação contínua (CI/CD). Dessa forma, quando o software passa pelo processo de criação e lançamento, as mudanças de infraestrutura necessárias podem ser feitas em conjunto.
- ❑ A IaC apresenta uma linguagem comum para desenvolvedores e operações. As mudanças podem ser revisadas de forma transparente, o que promove uma melhor colaboração em um ambiente de DevOps. Além disso a IaC pode ser utilizada para:
  - Configurar rapidamente ambientes completos, do desenvolvimento à produção
  - Ajudar a garantir configurações consistentemente reproduzíveis entre ambientes
  - Integrar-se perfeitamente aos provedores de nuvem e aumentar ou diminuir os recursos de infraestrutura com eficiência com base na demanda

# Ferramentas de IaC



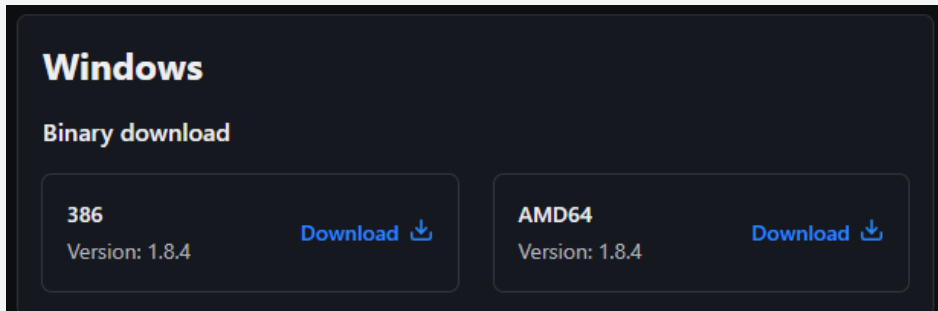
# O que é o Terraform?

- ❑ HashiCorp Terraform é uma ferramenta de infraestrutura como código que permite definir recursos de nuvem e locais em arquivos de configuração legíveis que você pode versionar, reutilizar e compartilhar.
- ❑ Você pode então usar um fluxo de trabalho consistente para provisionar e gerenciar toda a sua infraestrutura durante todo o seu ciclo de vida.
- ❑ O Terraform pode gerenciar componentes de baixo nível, como recursos de computação, armazenamento e rede, bem como componentes de alto nível, como entradas DNS e recursos SaaS.
- ❑ [Terraform | HashiCorp Developer](#)



# Instalando Terraform Windows

- ❑ Faça o download da pasta de binários no link [Install | Terraform | HashiCorp Developer](#)



- ❑ Extraia a pasta zipada e salve em algum local, de preferência `C:\Program Files\`
- ❑ Salve o caminho do arquivo nas variáveis de ambiente do Windows(PATH)

`C:\Program Files\terraform_1.8.4_windows_386`

- ❑ Depois de salvar as variáveis de ambiente teste no CMD ou no PowerShell com `terraform -version`



# Instalando Terraform Linux [Ubuntu]

[Install | Terraform | HashiCorp Developer](#)

```
wget -O- https://apt.releases.hashicorp.com/gpg | \
```

```
sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
```

```
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee \
```

```
/etc/apt/sources.list.d/hashicorp.list
```

```
sudo apt update && sudo apt install terraform
```

**Terraform AWS**

# Setup inicial

- ❑ É necessário, além do Terraform ter o AWS CLI instalado.

[Install or update to the latest version of the AWS CLI - AWS Command Line Interface \(amazon.com\)](#)

- ❑ Teste se instalou com

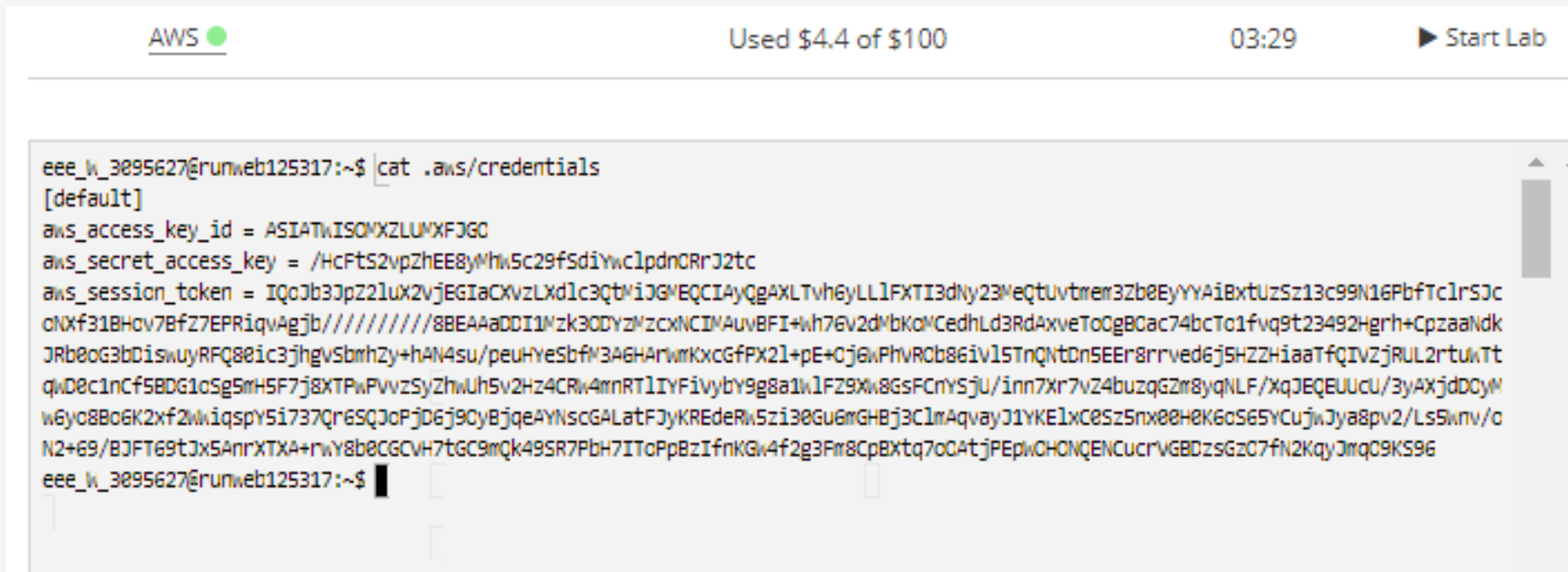
`aws --version`

- ❑ Após a instalação, é necessário configurar o CLI com as credenciais da AWS

[Set up the AWS CLI - AWS Command Line Interface \(amazon.com\)](#)

# Setup inicial

- ❑ Para acesso, será necessário alguns parâmetros. Ao inicializar um lab, no terminal digite `cat .aws/credentials`
- ❑ Lembrem-se que essas credenciais mudam todas as vezes que vocês iniciam um novo lab.



The screenshot shows a terminal window with a header bar containing 'AWS' with a green dot, 'Used \$4.4 of \$100', '03:29', and a 'Start Lab' button. The terminal content shows a user running the command `cat .aws/credentials` in a shell. The output displays a default profile with AWS access key ID, secret access key, and session token. The session token is a long alphanumeric string.

```
eee_w_3095627@runweb125317:~$ cat .aws/credentials
[default]
aws_access_key_id = ASIATWISOMXZLUMXFJGO
aws_secret_access_key = /HcFts2vpZhEE8yMhW5c29fSdiYwclpdnQRrJ2tc
aws_session_token = IqoJb3JpZ2luX2VjEGlAcXVzLXdlc3QtMiJGMEQCIAYQgAXLTvh6yLLlFXTI3dNy23MeQtUvtrem3Zb0EYyYAIBxtuzSz13c99N16PbfTclrSJC
oNXf31BHov7BfZ7EPRIqvAgjb/////////8BEAAaDDI1Mzk3ODYzNzcXNCIMAUvBFI+Wh76V2dMbKdCedhLd3RdAxveToGBOac74bcTo1fvq9t23492Hgrh+CpzaaNdK
JRb0oG3bDiswuyRFQ80ic3jhgVsbmhZy+hAN4su/peuHYeSbfM3A6HArWmKxcGfPX2l+pE+Oj0wPhVRCb86ivL5TnQNTDn5EEr8rrved6j5HZZHiaaTfQIVZjRUL2rtuKtT
qWd0c1nCf5BDG1cSg5mH5F7j8XTPwPVvzSyZhWuH5v2Hz4CRW4mnRTlIYFivby9g8a1wLFZ9Xw8GsFCnYSjU/inn7Xr7vZ4buzqGZm8yqNLF/XqJEQEUUCU/3yAXjddCyM
w6yc8Bo6K2xf2WkiqspY5i737Qr6SQJcPjD6j9CyBjqeAYNscGALatFJyKREderW5zi30Gu6mGHBj3ClnAqvayJ1YKElxC0Sz5nx00H0K6cS65YCuJwJya8pv2/Ls5wuv/c
N2+69/BJFT69tJx5AnrXTXA+rwY8b0CGCVH7tGC9mQk49SR7PbH7IToPpBzIfnKGw4f2g3Fm8CpBxtq7oCatjPEpWQHONQENCucrvGBDzsGz07fn2KqyJmQ09KS96
eee_w_3095627@runweb125317:~$
```

# Setup inicial

- ❑ Agora no PowerShell, bash, ou CMD, digite: `aws configure`

```
PS C:\Users\Eduardo Verri\Desktop\terraform_projects\terraform-aws-101> aws configure
AWS Access Key ID [*****FJGO]: ASIATWISOMXZLUMXFJGO
AWS Secret Access Key [*****J2tc]: /HcFtS2vpZhEE8yMhW5c29fSdiYwclpdnORrJ2tc
Default region name [us-east-1]: us-east-1
Default output format [json]: json
```

- ❑ Você também precisará utilizar o token da sessão com

`aws configure set aws_session_token <<token>>`

```
PS C:\Users\Eduardo Verri\Desktop\terraform_projects\terraform-aws-101> aws configure set aws_session_token IQoJb3JpZ2luX2VjEGlAcXVzLXdlc3QtMiJGMEQ
CIAyQgAXLTvh6yLLlFXtI3dNy23MeQtUvtmem3Zb0EyYYAiBxtUzSz13c99N16PbfTclrSJcoNXf31BHov7BfZ7EPRIqvAgjb/////////8BEAAaDDI1Mzk3ODYzMzcXNCIMauvBFI+Wh76V2d
MbKoMCedhLd3RdAxveToOgBOac74bcTo1fvq9t23492Hgrh+CpzaaNdkJRb0oG3bDiswuyRFQ80ic3jhgVSbmhZy+hAN4su/peuHYeSbfM3A6HArWmKxcGfPX2l+pE+Oj6WPhVR0b86iVl5TnQN
tDn5EEr8rrved6j5HZZHiaaTfQIVZjRUL2rtuWTtqWD0c1nCf5BDG1oSg5mH5F7j8XTPwPVvzSyZhWUh5v2Hz4CRW4mnRTLIYFiVybY9g8a1WlFZ9XW8GsFCnYSjU/inn7Xr7vZ4buzqGZm8yqN
LF/XqJEQEuuUcU/3yAXjdD0yMw6yo8Bo6K2xf2WWiqspY5i737Qr6SQJoPjD6j90yBjqeAYNscGALatFJyKREdeRW5zi30Gu6mGHBj3ClmAqvayJ1YKElxC0Sz5nx00H0K6oS65YCuJwJya8pv2/
Ls5Wnv/oN2+69/BJFT69tJx5AnrXTXA+rwY8b0CGCVH7tGC9mQk49SR7PbH7IToPpBzIfnKGW4f2g3Fm8CpBXtq7o0AtjPEpWOHONQENCucrVGBDzsGz07fN2KqyJmq09KS96
```

# Criando primeira instância





Crie um arquivo `main.tf`

`terraform init`

`terraform apply`

```
terraform {  
  required_providers {  
    aws = {  
      source  = "hashicorp/aws"  
      version = "~> 4.16"  
    }  
  }  
  required_version = ">= 1.2.0"  
}  
  
provider "aws" {  
  region = "us-east-1"  
}  
  
resource "aws_instance" "app_server" {  
  ami           = "ami-0b0ea68c435eb488d"  
  instance_type = "t2.micro"  
  
  tags = {  
    Name = "ExampleAppServerInstance"  
  }  
}
```

# Criando primeira instância

Instâncias (1/1) <a href="#">Informações</a>			<a href="#">Conectar</a>	<a href="#">Estado da instância</a> ▼
<input type="text" value="Localizar Instância por atributo ou tag (case-sensitive)"/>		<a href="#">Executando</a>		
<input checked="" type="checkbox"/>	<a href="#">Name</a> ✎ ▼	<a href="#">ID da instância</a>	<a href="#">Estado da instância</a> ▼	<a href="#">Tipo de inst</a>
<input checked="" type="checkbox"/>	ExampleAppServerInstance	i-00a001326f5a04f18	 Executando  	t2.micro






## Entendendo um pouco mais...

- ❑ Ao criar uma nova configuração — ou verificar uma configuração existente no controle de versão — você precisa inicializar o diretório com **terraform init**. Inicializar um diretório de configuração baixa e instala os provedores definidos na configuração, que neste caso é o provedor aws.
- ❑ Aplique a configuração agora com o comando **terraform apply**. Antes de aplicar qualquer alteração, o Terraform imprime o plano de execução que descreve as ações que o Terraform executará para alterar sua infraestrutura para corresponder à configuração.
- ❑ O bloco **Provider** configura o provedor especificado, neste caso aws. Um provedor é um plugin que o Terraform usa para criar e gerenciar seus recursos.
- ❑ Use blocos de **recursos** para definir componentes da sua infraestrutura. Um recurso pode ser um componente físico ou virtual, como uma instância do EC2.



# Destruindo a primeira instância

O comando **terraform destroy** encerra recursos gerenciados pelo seu projeto Terraform. Este comando é o inverso de **terraform apply**, pois encerra todos os recursos especificados em seu estado Terraform. Ele não destrói recursos executados em outros lugares que não são gerenciados pelo projeto Terraform atual.

Instâncias (4) <a href="#">Informações</a>			<a href="#">Conectar</a>	<a href="#">Estado da instância</a> ▼
<input type="text" value="Localizar Instância por atributo ou tag (case-sensitive)"/>				<a href="#">Todos os estados</a>
<input type="checkbox"/>	Name 	ID da instância	Estado da instância ▼	Tipo de inst...
<input type="checkbox"/>	ExampleAppServerInstance	i-00a001326f5a04f18	 Encerrado  	t2.micro

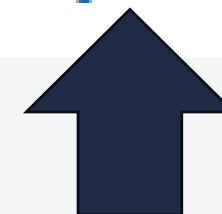
# Achando uma AMI

Para poder saber os ids de cada AMI busque na EC2 pelo menu de imagens, ou filtrando por AMI ou por catálogo de imagens

## ▼ Imagens

AMIs

[Catálogo de AMIs](#)



ubuntu<sup>®</sup>

Ubuntu

Qualificado para o nível  
gratuito

Provedor verificado

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

ami-04b70fa74e45c3917 (64 bits (x86)) / ami-0eac975a54dfce8cb (64 bits (Arm))

Ubuntu Server 24.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Plataforma: ubuntu  
ENA habilitado: Sim

Tipo de dispositivo raiz: ebs

Virtualização: hvm

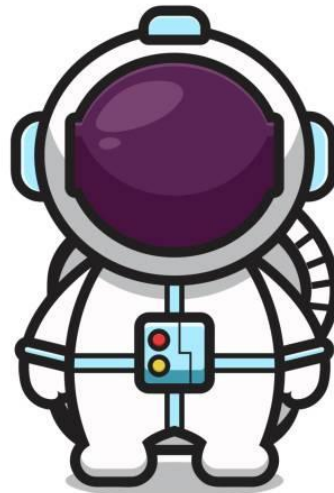
Selecionar

☒ 64 bits (x86)

☐ 64 bits (Arm)

## Um passo além...

[Docs overview](#) | [hashicorp/aws](#) | [Terraform](#) | [Terraform Registry](#)



# Chave .pem - Criação

- ❑ Crie uma chave .pem com o comando

`ssh-keygen -m PEM -t rsa -b 4096 -f id_rsa.pem`

```
Generating public/private rsa key pair.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in id_rsa.pem  
Your public key has been saved in id_rsa.pem.pub  
The key fingerprint is:  
SHA256:mNB5blq92EuAqXN53ITLV/XK3Sr6p1evrb2MSAGrbFM eduardo verri@spitech02608  
The key's randomart image is:  
+---[RSA 4096]-----+  
|  
| . .  
| . o . . .  
| . B oo . .  
| = SEo.. .  
| ..BoB o.. oo|  
| o +=* *. o.+|  
| o...o..o =+.|  
| o+o*+=+|  
+-----[SHA256]-----+
```

## Chave .pem – Incrementando o Terraform

Criamos uma variável para conter o nome de nosso par de chaves

E então criamos o recurso para ler a informação de nossa chave pública

Por fim adicionamos a chave na geração de nossa instância

```
variable "key_pair_name"{
    type = string
    default = "id_rsa"
}

resource "aws_key_pair" "generated_key"{
    key_name = var.key_pair_name
    public_key = file("id_rsa.pem.pub")
}

resource "aws_instance" "app_server" {
    ami          = "ami-0e001c9271cf7f3b9"
    instance_type = "t2.micro"
    key_name     = aws_key_pair.generated_key.key_name

    tags = {
        Name = "ec2-terraform"
    }
}
```

## Grupo de segurança

- ❑ Por definição, o grupo de segurança está associado a uma VPC. Nesse momento, vamos utilizar a VPC padrão da AWS

ID da VPC
<a href="#">vpc-0e71fa252b4448a14</a>

- ❑ Agora precisamos definir as regras de entrada e saída, no caso habilitei apenas o SSH como entrada

```
resource "aws_security_group" "basic_security" {  
  name          = "basic_security"  
  description   = "Allow SSH access"  
  vpc_id        = "vpc-0e71fa252b4448a14"  
  
  ingress {  
    from_port    = "22"  
    to_port      = "22"  
    protocol     = "tcp"  
    cidr_blocks  = ["0.0.0.0/0"]  
  }  
  
  egress {  
    from_port    = 0  
    to_port      = 0  
    protocol     = "-1"  
    cidr_blocks  = ["0.0.0.0/0"]  
  }  
}
```

# Associando um volume

- ❑ A atualização de um volume é feita no recurso de criação da instância, com o argumento `ebs_block_device`

```
resource "aws_instance" "app_server" {  
  ami           = "ami-0e001c9271cf7f3b9"  
  availability_zone = "us-east-1a"  
  instance_type = "t2.small"  
  ebs_block_device {  
    device_name = "/dev/sda1"  
    volume_size = 30  
    volume_type = "standard"  
  }  
  key_name = aws_key_pair.generated_key.key_name  
  vpc_security_group_ids = [aws_security_group.basic_security.id]  
  
  tags = {  
    Name = "ec2-terraform"  
  }  
}
```

## Associando um novo volume

- ❑ Podemos também criar um novo volume com `aws_ebs_volume` e depois associá-lo na instância com `aws_volume_attachment`
- ❑ Lembrando, que a zona de disponibilidade deve ser a mesma da instância

```
resource "aws_ebs_volume" "example"{
  availability_zone = "us-east-1a"
  size = 30

  tags = {
    Name = "Hello-terraform"
  }
}

resource "aws_volume_attachment" "ebs_att"{
  device_name = "/dev/sdh"
  volume_id = aws_ebs_volume.example.id
  instance_id = aws_instance.app_server.id
}
```



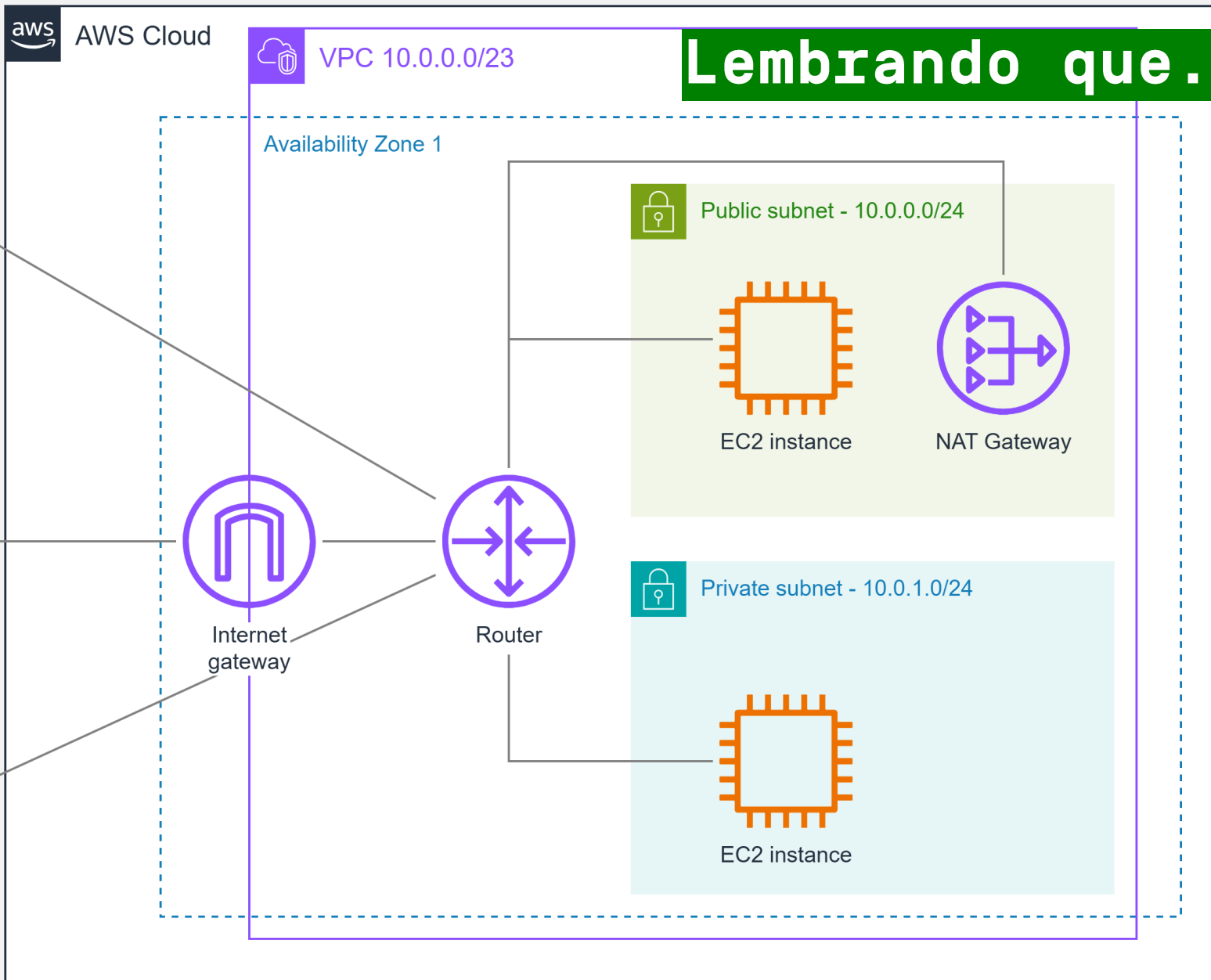
**Arquitetura Computacional 3SIS AWS - Terraform**

Public route table	
Destination	Target
10.0.0.0/23	local
0.0.0.0/0	<i>igw-id</i>



web client

Private route table	
Destination	Target
10.0.0.0/23	local
0.0.0.0/0	<i>nat-gateway-id</i>



Lembrando que...

## Estrutura de diretórios sugerida

```
✓ modules
  ✓ ec2
    ✎ ec2.tf
    ✎ variable.tf
  ✓ network
    ✎ network.tf
    ✎ variable.tf
  ✎ main.tf
  🔒 tf_key.pem
  ≡ tf_key.pem.pub
```

```
C:.\
  main.tf
  tf_key.pem
  tf_key.pem.pub
  modules
    ec2
      ec2.tf
      variable.tf
    network
      network.tf
      variable.tf
```

## Comandos úteis

[Terraform Cheat Sheet - 23 Terraform CLI Commands & Examples \(spacelift.io\)](https://spacelift.io/terraform-cheat-sheet)

- ❑ **terraform fmt** — Formate seus arquivos de configuração usando o padrão de linguagem HCL
- ❑ **terraform fmt --recursive** — Também formate arquivos em subdiretórios
- ❑ **terraform init** — Para preparar o diretório de trabalho, o comando terraform init executa a inicialização do back-end, a instalação do módulo filho e a instalação dos plug-ins.
- ❑ **terraform validate** — Valide os arquivos de configuração em seu diretório e não acesse nenhum estado ou serviço remoto. terraform init deve ser executado antes deste comando.
- ❑ **terraform plan** — irá gerar um plano de execução, mostrando quais ações serão tomadas sem realmente executar as ações planejadas.
- ❑ **terraform apply** — Crie ou atualize a infraestrutura dependendo dos arquivos de configuração. Por padrão, um plano será gerado primeiro e precisará ser aprovado antes de ser aplicado.
- ❑ **terraform destroy** — Destrua a infraestrutura gerenciada pelo Terraform

# Comandos úteis – de maneira geral

`terraform init`

`terraform validate`

`terraform plan`

`terraform apply`

**E ao final....** `terraform destroy`

**Agradeço**  
a sua atenção!



SÃO  
PAULO  
TECH  
SCHOOL