Certified Tech Developer
The Ultimate Degree

Infraestrutura II

Criando um ambiente de desenvolvimento com a Terraform

Te damos as boas-vindas! Neste espaço, vamos colocar em prática todo o aprendizado durante esta semana. Vamos lá!

Objetivo

O desafio é usar Terraform na nossa conta da AWS para criar duas instâncias EC2 dentro de uma região específica.

Instruções

No último guia que realizamos para a Ansible, devemos inserir no nosso computador as credenciais da AWS, transformadas em senhas cifradas, ou seja, a maior parte do trabalho prévio já foi realizado. Mas... o que está faltando?

Quando fizermos alterações na infraestrutura utilizando código, iremos precisar de um método de implantação. No nosso caso, deveremos instalar "Terraform CLI" em nosso computador, para poder executar os comandos e cumprir nosso objetivo. Devemos, também, escolher a opção mais adequada para o tipo de sistema operacional instalado. Podemos seguir o guia a partir do link a seguir:

https://learn.hashicorp.com/tutorials/terraform/install-cli

Agora sim, mãos à obra! A Terraform nos permite utilizar módulos já definidos para a AWS, tornando muito mais ágil o trabalho, pois a maior parte da lógica da automação já está realizada.

Vamos criar dois servidores na AWS que compartilham a mesma rede, utilizando os seguintes serviços:

- EC2
- VPC

Vamos começar! A primeira coisa que devemos fazer é utilizar a VPC por padrão dentro da nossa conta da AWS e, então, criar nossas instâncias EC2.

Lembrando que um dos benefícios da infraestrutura como código é a possibilidade de versionar nossos arquivos de configuração. Não hesite em fazê-lo para mantê-los acessíveis e, também, em compartilhá-los com suas equipes de trabalho. Porém, devemos adicionar, no nosso arquivo .gitignore, a lista a seguir:

```
.terraform

.terraform.lock.hcl

terraform.tfstate

terraform.tfstate.backup
```

Sendo que esses arquivos podem conter informações sensíveis, como credenciais.

Dando continuidade a nossa tarefa, vamos criar nosso primeiro arquivo de configuração para a Terraform. Devemos considerar que as extensões são importantes para nossos desenvolvimentos, pois através delas estamos indicando a ferramenta ou a linguagem de programação que iremos utilizar. Em um diretório ou pasta vazios, vamos criar os seguintes arquivos:

vpc.tf

Vamos criar o arquivo “vpc.tf” dentro do nosso diretório de trabalho:

```
resource "aws_default_vpc" "shared-vpc" {
  tags = {
    Name = "Default VPC"
  }
}
resource "aws_subnet" "my-subnet" {
  vpc_id     = aws_default_vpc.shared-vpc.id
  cidr_block = "172.31.128.0/20"
  tags = {
    Name = "grupo-x-subnet"
  }
}
```

Aqui configuramos nossa nuvem virtual privada e uma sub-rede que depois serão utilizadas por nossas instâncias.

ec2.tf

No nosso segundo arquivo de configuração, vamos definir duas instâncias EC2 conectadas, compartilhando a mesma VPC e um grupo de segurança para definir os acessos no nível de rede.

```
module "ec2_cluster" {
  source                 = "terraform-aws-modules/ec2-instance/aws"
  version                = "~> 2.0"
  name                   = "grupo-x-maquinavirtual"
  instance_count         = 2
  ami                    = "ami-04505e74c0741db8d"
  instance_type          = "t2.micro"
  vpc_security_group_ids =
[module.ssh_security_group.this_security_group_id]
  subnet_ids             = [ aws_subnet.my-subnet.id ]
  tags = {
    Terraform   = "true"
    Environment = "dev"
 }
}

module "ssh_security_group" {
  source      = "terraform-aws-modules/security-group/aws//modules/ssh"
  version     = "~> 3.0"
  name        = "ssh-server"
  description = "Grupo de segurança para server ssh e porto ssh (22)abertos"
  vpc_id      = aws_default_vpc.shared-vpc.id
  ingress_cidr_blocks = ["172.31.0.0/16"]
}
```

main.tf

Por fim, como em qualquer projeto de código, precisamos de nosso ponto de entrada à aplicação, com suas definições globais. No nosso caso, este “main.tf” irá conter o endereço das nossas credenciais da AWS e da região onde serão utilizadas.

```
provider "aws" {
 shared_credentials_files = [ "~/.aws/credentials" ]
 region = "us-east-1"
}
```

O parâmetro “shared_credentials_files” terá como valor o arquivo com as credenciais da AWS no computador de cada um. O utilizado neste exemplo é a rota por padrão, onde são comumente armazenados.

Vamos passar à ação!

Os comandos da Terraform

terraform init

Quando temos um projeto novo na Terraform, devemos inicializá-lo através do comando "terraform init". A saída do comando deve ser parecida com:

```
[enuel@enuel digitalhouse]$ terraform init
Initializing modules...
Downloading terraform-aws-modules/ec2-instance/aws 2.19.0 for ec2_cluster...
- ec2_cluster in .terraform/modules/ec2_cluster
Downloading terraform-aws-modules/security-group/aws 3.18.0 for ssh_security_group...
- ssh_security_group in .terraform/modules/ssh_security_group/modules/ssh
- ssh_security_group.sg in .terraform/modules/ssh_security_group
Downloading terraform-aws-modules/vpc/aws 3.2.0 for vpc...
- vpc in .terraform/modules/vpc

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching ">= 2.42.0, >= 3.15.0, >= 3.24.0"...
- Installing hashicorp/aws v3.51.0...
- Installed hashicorp/aws v3.51.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

O texto final em verde indica que ele foi corretamente inicializado.

terraform validate

Após a inicialização, verifique a sintaxe e as demais configurações locais com "terraform validate"

terraform plan

Permite pré-visualizar as alterações que faremos na infraestrutura, ressaltando os recursos que serão criados e destruídos, sem executar nenhuma alteração real.

```
[user@vm]$ terraform plan
Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_default_vpc.shared-vpc will be created
  + resource "aws_default_vpc" "shared-vpc" {
      + arn                                  = (known after apply)
      + cidr_block                           = (known after apply)
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_classiclink                   = (known after apply)
      + enable_classiclink_dns_support       = (known after apply)
      + enable_dns_hostnames                 = true
      + enable_dns_support                   = true
      + existing_default_vpc                 = (known after apply)
      + force_destroy                        = false
      + id                                   = (known after apply)
      + instance_tenancy                     = (known after apply)
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "Default VPC"
        }
      + tags_all                             = {
          + "Name" = "Default VPC"
        }
    }

...

Plan: 8 to add, 0 to change, 0 to destroy.
```

terraform apply

O próximo passo é aplicar as configurações terraform que criamos utilizando o comando "terraform apply". Ele é responsável pela aplicação das alterações na conta da AWS, de forma remota e no conforto do nosso computador. Este processo poderia demorar, dependendo do tempo que a AWS precisar para criar recursos e da conexão à internet. Na saída do comando, existe um dado a destacar: a quantidade de recursos que irá criar (add), os que irá alterar (change) e os que deve eliminar (destroy):

```
Plan: 26 to add, 0 to change, 0 to destroy.
```

Quando todas as ações a serem realizadas pelo comando forem detalhadas em uma grande lista, ele irá nos solicitar uma configuração. Vamos inserir "yes".

```
Plan: 26 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

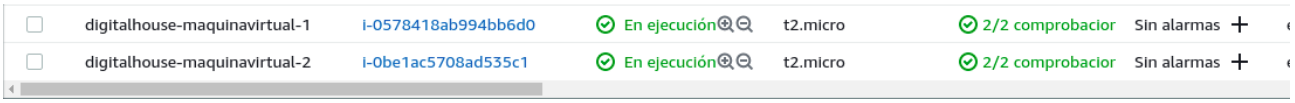
Após isso, vemos como a saída do comando exibe em tempo real a criação dos recursos:

```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

module.vpc.aws_eip.nat[1]: Creating...
module.vpc.aws_vpc.this[0]: Creating...
module.vpc.aws_eip.nat[0]: Creating...
module.vpc.aws_eip.nat[0]: Creation complete after 3s [id=eipalloc-0c6ea6668e8410eae]
module.vpc.aws_eip.nat[1]: Creation complete after 3s [id=eipalloc-01ae76653708644f5]
module.vpc.aws_vpc.this[0]: Still creating... [10s elapsed]
module.vpc.aws_vpc.this[0]: Creation complete after 13s [id=vpc-0043735802e80c005]
module.vpc.aws_internet_gateway.this[0]: Creating...
module.vpc.aws_subnet.public[1]: Creating...
module.vpc.aws_subnet.private[1]: Creating...
module.vpc.aws_route_table.private[0]: Creating...
module.vpc.aws_route_table.public[0]: Creating...
module.vpc.aws_route_table.private[1]: Creating...
module.vpc.aws_subnet.private[0]: Creating...
module.vpc.aws_subnet.public[0]: Creating...
```

Ao finalizar, ele nos indica o estado final da nossa infraestrutura, com um resumo que deve coincidir com o informado antes de inserirmos “yes”:

```
module.vpc.aws_nat_gateway.this[0]: Still creating... [1m10s elapsed]
module.vpc.aws_nat_gateway.this[1]: Still creating... [1m10s elapsed]
module.vpc.aws_nat_gateway.this[0]: Still creating... [1m20s elapsed]
module.vpc.aws_nat_gateway.this[1]: Still creating... [1m20s elapsed]
module.vpc.aws_nat_gateway.this[1]: Still creating... [1m30s elapsed]
module.vpc.aws_nat_gateway.this[0]: Still creating... [1m30s elapsed]
module.vpc.aws_nat_gateway.this[0]: Still creating... [1m40s elapsed]
module.vpc.aws_nat_gateway.this[1]: Still creating... [1m40s elapsed]
module.vpc.aws_nat_gateway.this[0]: Still creating... [1m50s elapsed]
module.vpc.aws_nat_gateway.this[1]: Still creating... [1m50s elapsed]
module.vpc.aws_nat_gateway.this[1]: Creation complete after 1m52s [id=nat-0473da0778adfb6b3]
module.vpc.aws_nat_gateway.this[0]: Creation complete after 1m52s [id=nat-0900ca8a6dd115399]
module.vpc.aws_route.private_nat_gateway[1]: Creating...
module.vpc.aws_route.private_nat_gateway[0]: Creating...
module.vpc.aws_route.private_nat_gateway[1]: Creation complete after 5s [id=r-rtb-00524ad475278b2041080289494]
module.vpc.aws_route.private_nat_gateway[0]: Creation complete after 5s [id=r-rtb-07f739de56077607010802894 94]

Apply complete! Resources: 26 added, 0 changed, 0 destroyed.
```

Foi um sucesso! Nossas duas instâncias foram criadas com todos os recursos no seu contexto para funcionarem corretamente:



terraform destroy

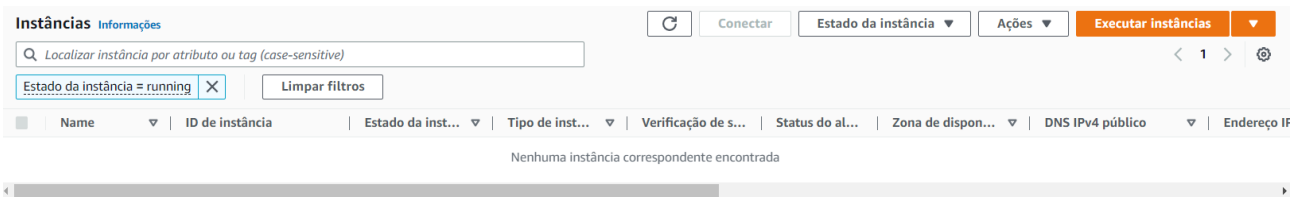
Caso deixemos de utilizar nossa infraestrutura, para eliminar tudo basta executar o comando: “terraform destroy”.

Como vemos na saída, o resumo do plano é alterado informando que tudo está em condições de ser eliminado:

```
Plan: 0 to add, 0 to change, 26 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

module.vpc.aws_route_table_association.private[1]: Destroying... [id=rtbassoc-0f3e3a2d33db505a1]
module.vpc.aws_route_table_association.public[1]: Destroying... [id=rtbassoc-0dfb426c335a5fc16]
module.vpc.aws_route_table_association.public[0]: Destroying... [id=rtbassoc-0ec1964145ab2ff46]
module.ssh_security_group.module.sg.aws_security_group_rule.ingress_rules[0]: Destroying... [id=sgrule-2140067553]
module.vpc.aws_route.private_nat_gateway[0]: Destroying... [id=r-rtb-07f739de5607760701080289494]
module.vpc.aws_route_table_association.private[0]: Destroying... [id=rtbassoc-0c55689a57025b1d9]
module.ssh_security_group.module.sg.aws_security_group_rule.egress_rules[0]: Destroying... [id=sgrule-718074897]
module.vpc.aws_route.private_nat_gateway[1]: Destroying... [id=r-rtb-00524ad475278b2041080289494]
module.vpc.aws_route.public_internet_gateway[0]: Destroying... [id=r-rtb-06a63e4f76701b0bc1080289494]
module.ec2_cluster.aws_instance.this[0]: Destroying... [id=i-0578418ab994bb6d0]
module.vpc.aws_route_table_association.private[1]: Destruction complete after 2s
```

Ao finalizar, veremos que nossos recursos foram eliminados com sucesso, podendo verificá-los na saída do comando e na nossa conta da AWS.

```
module.vpc.aws_internet_gateway.this[0]: Destroying... [id=igw-0755e83b90b764dc8]
module.vpc.aws_subnet.public[1]: Destroying... [id=subnet-08a98721e52718d95]
module.vpc.aws_eip.nat[1]: Destroying... [id=eipalloc-01ae76653708644f5]
module.vpc.aws_eip.nat[0]: Destroying... [id=eipalloc-0c6ea6668e8410eae]
module.vpc.aws_subnet.public[0]: Destroying... [id=subnet-0e35369e8d34fa577]
module.vpc.aws_subnet.public[1]: Destruction complete after 2s
module.vpc.aws_eip.nat[1]: Destruction complete after 2s
module.vpc.aws_eip.nat[0]: Destruction complete after 2s
module.vpc.aws_subnet.public[0]: Destruction complete after 2s
module.vpc.aws_internet_gateway.this[0]: Still destroying... [id=igw-0755e83b90b764dc8, 10s elapsed]
module.vpc.aws_internet_gateway.this[0]: Destruction complete after 12s
module.vpc.aws_vpc.this[0]: Destroying... [id=vpc-0043735802e80c005]
module.vpc.aws_vpc.this[0]: Destruction complete after 1s

Destroy complete! Resources: 26 destroyed.
```



Conclusão

Durante este exercício, aprendemos que a Terraform nos permite otimizar a criação da nossa infraestrutura. Nosso código não é extenso; recorremos a módulos para que nosso código seja declarativo e legível para nossos colegas de equipe.

Você poderia criar um banco de dados RDS que esteja conectado às instâncias EC2? Temos à disposição a registry da Terraform, para procurar o módulo mais adequado para a realização desta tarefa.

No link a seguir, podemos encontrar os módulos utilizados neste exercício guiado, bem como mais módulos para a AWS:

https://registry.terraform.io/providers/hashicorp/aws/latest