Terraform Modules

A configuração do Terraform pode ser separada em módulos para organizar melhor sua configuração. Isso torna seu código mais fácil de ler e reutilizável em toda a organização. Um módulo do Terraform é muito simples: qualquer conjunto de arquivos de configuração do Terraform em uma pasta é um módulo. Os módulos são o ingrediente chave para escrever código Terraform reutilizável e sustentável. Configurações complexas, projetos de equipe e bases de código de vários repositórios se beneficiarão dos módulos. Adquira o hábito de usá-los sempre que fizer sentido..

- Tarefa 1: Criar um módulo Terraform local
- Tarefa 2: Referencie um módulo na configuração do Terraform
- Tarefa 3: Reutilização do módulo Terraform

Tarefa 1: Criar um modulo local Terraform

Um módulo do Terraform é apenas um conjunto de arquivos de configuração do Terraform. Os módulos são apenas configurações do Terraform dentro de uma pasta - não há nada de especial neles. Na verdade, o código que você escreveu até agora é um módulo: o módulo raiz. Neste laboratório, criaremos um módulo local para uma nova configuração de servidor.

Passo 1.1

Crie um novo diretório chamado `server` no dentro do diretório do modulo raiz e criei um novo arquivo dentro chamado `server.tf`.

Passo 1.2

Edite o arquivo `server/server.tf`, com o seguinte conteúdo:

```
variable "ami" {}

variable "size" {
  default = "t2.micro"
}

variable "subnet_id" {}

variable "security_groups" {
  type = list(any)
}
```

```
resource "aws_instance" "web" {
  ami                    = var.ami
  instance_type          = var.size
  subnet_id              = var.subnet_id
  vpc_security_group_ids = var.security_groups

  tags = {
     "Name" = "Server from Module"
  }
}

output "public_ip" {
   value = aws_instance.web.public_ip
}

output "public_dns" {
   value = aws_instance.web.public_dns
}
```

Passo 1.3

Em sua configuração raiz (também chamada de módulo raiz) `main.tf`, podemos chamar nosso novo modulo `server` com o bloco Terraform module. Lembre-se que o terraform funciona apenas com os arquivos de configuração que estão em seu diretório de trabalho atual. Os módulos nos permitem referenciar a configuração do Terraform que fica fora do nosso diretório de trabalho. Neste caso vamos incorporar toda a configuração que está dentro do nosso diretório de trabalho (módulo raiz) e dentro do diretório `server`.

```
module "server1" {
  source          = "./server"
  ami             = data.aws_ssm_parameter.al2_ami.value
  subnet_id       = aws_subnet.lab_subnet.id
  security_groups = [
     aws_security_group.lab_sg.id
  ]
}
```

Passo 1.4 Instalar e Aplicar o Módulo

Agora execute `terraform init` para instalar o módulo. Os arquivos de configuração do Terraform localizados nos módulos são baixados pelo Terraform durante a inicialização, portanto, sempre que você adicionar ou atualizar uma versão do módulo, deverá executar um `terraform init`.

```
terraform init
```

Você pode ver que nossa configuração agora depende deste módulo para ser instalado e usado usando o command `terraform providers`.

```
› terraform providers

Providers required by configuration:

.

├── provider[registry.terraform.io/hashicorp/aws] ~> 5.11.0

└── module.server1

    └── provider[registry.terraform.io/hashicorp/aws]
```

Execute `terraform apply` para criar um novo servidor usando o módulo `server`. Pode levar alguns minutos para o servidor ser criado usando o módulo.

```
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.server1.aws_instance.web: Creating...
module.server1.aws_instance.web: Still creating... [10s elapsed]
```

Tarefa 3: Fazer referência a um módulo na configuração do Terraform

Quando usados, os módulos do Terraform serão listados no arquivo de estado do Terraform e podem ser referenciados usando o nome do módulo.

```
terraform state list
```

```
# Recurso definido no módulo raiz/diretório de trabalho
aws_instance.vm1

# Recurso definido no módulo server
module.server1.aws_instance.web
```

Podemos ver todos os detalhes do servidor criado usando nosso módulo `server`.

```
terraform state show module.server1.aws_instance.web
```

Podemos adicionar dois blocos output ao nosso `outputs.tf` para mostrar as informações de IP e DNS de nosso modulo `server`. Observe como o Terraform faz referência (sintaxe de interpolação) a informações sobre a construção do servidor a partir de um módulo.

```
output "server1_public_ip" {
  value = module.server1.public_ip
}

output "server1_public_dns" {
  value = module.server1.public_dns
}
```

Tarefa 3: Reutilize o módulo para criar um servidor em uma sub-rede diferente

Um dos benefícios dos módulos do Terraform é que eles podem ser facilmente reutilizados em sua organização. Vamos usar nosso módulo local novamente para criar outro servidor em uma sub-rede separada.

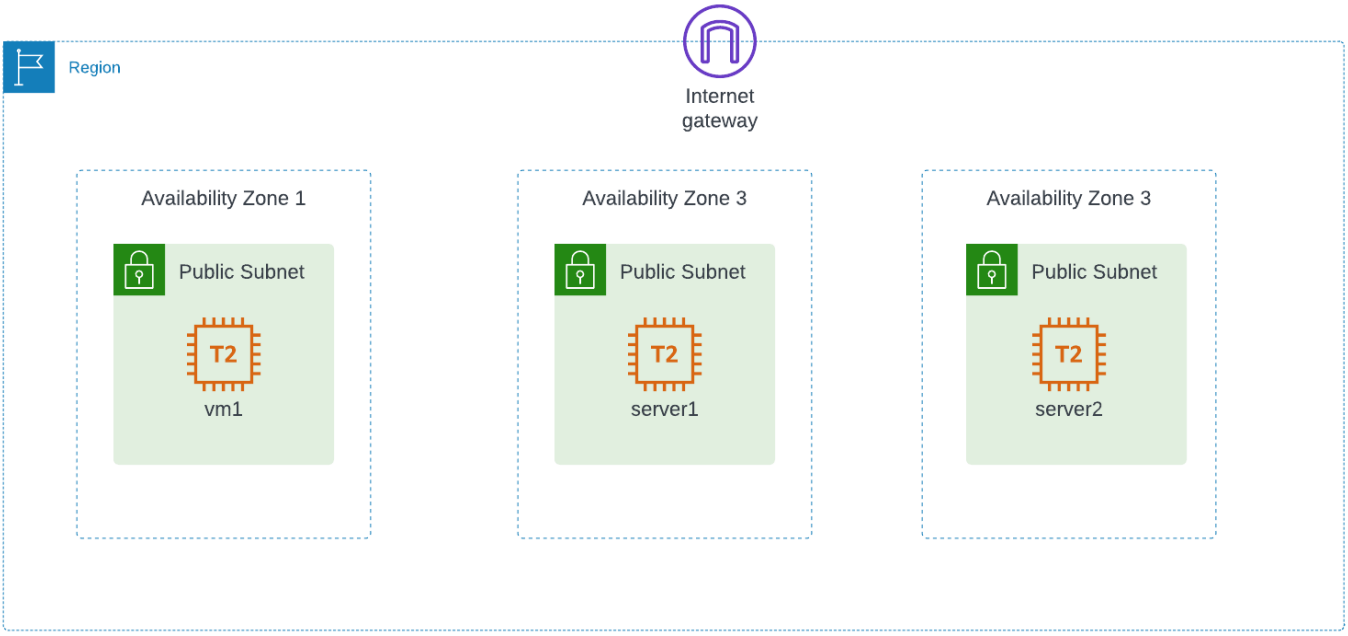

Figure 1: Arquitetura Desejada

Modifique o arquivo network.tf e defina uma segunda e terceira subnet pública.

```
locals {
  # second subnet cidr
  public_subnet_cidr = cidrsubnet(var.vpc_cidr, 8, 1)

  # CIDR DA SEGUNDA SUBNET PUBLICA
  public_subnet2_cidr = cidrsubnet(var.vpc_cidr, 8, 2)

  # CIDR DA TERCEIRASUBNET PUBLICA
  public_subnet3_cidr = cidrsubnet(var.vpc_cidr, 8, 3)
}

resource "aws_subnet" "publica2" {
  vpc_id                  = aws_vpc.lab_vpc.id
  cidr_block              = local.public_subnet2_cidr
  availability_zone       = data.aws_availability_zones.azs.names[1]
  map_public_ip_on_launch = true
}


resource "aws_subnet" "publica3" {
  vpc_id                  = aws_vpc.lab_vpc.id
  cidr_block              = local.public_subnet3_cidr
  availability_zone       = data.aws_availability_zones.azs.names[2]
  map_public_ip_on_launch = true
}
```

Ajuste o arquivo main.tf e altera a subnet do server1 e defina o server2.

```
module "server1" {
   source          = "./server"
   ami             = data.aws_ssm_parameter.al2_ami.value
   subnet_id       = aws_subnet.publica2.id
   security_groups = [
      aws_security_group.lab_sg.id
   ]
}
module "server2" {
   source          = "./server"
   ami             = data.aws_ssm_parameter.al2_ami.value
   subnet_id       = aws_subnet.publica3.id
   security_groups = [
      aws_security_group.lab_sg.id
   ]
}
```

Sempre que você adicionar ou atualizar uma versão do módulo, deverá executar um `terraform init`.

```
terraform init
```

Execute `terraform apply` para criar um novo servidor usando o módulo de servidor. Pode levar alguns minutos para o servidor ser criado usando o módulo.