Demo Terraform

GROUPE 4 AWS/RESTART COHORTE 2

Mame Sandeck Niang
Hapsatou Abou Sow
Aliou Cisse
Khady Diagne
Oumar Bounekhatap Faye

Pour réaliser ce tutoriel, vous devez créer un dossier de projet dans lequel vous ajoutez les fichiers suivantes :

- Main.tf
- Vars.tf
- Outputs.tf
- terraform.tfvars
- index.html dans lequel vous afficher **HELLO WORD**

Après avoir créer le projet, placez-vous sur le chemin du dossier et taper la commande si dessous pour initialiser votre projet.

```
Terraform init
```

On commence par créer nos variables de connexion dans le fichier vars.tf.

```
variable "AWS_ACCESS_KEY" {}

variable "AWS_SECRET_KEY" {}
variable "AWS_REGION" {
    default = "us-east-1"
}
```

Les clés sont des variables trop sensibles pour être exposées, raison pour laquelle nous allons leurs attribuer leurs valeurs dans le dossiers terraform.tfvars.

```
AWS_ACCESS_KEY="VALEUR_DE_ACCESS_KEY"
AWS_SECRET_KEY="VALEUR_DE_SECRET_KEY"
```

A présent nous pouvons créer notre provider dans le fichiers main.tf.

```
provider "aws" {
    region = var.AWS_REGION
    access_key = var.AWS_ACCESS_KEY
    secret_key = var.AWS_SECRET_KEY

}
```

Maintenant nous pouvons créer notre machine virtuelle mais avant cela, il nous faut un groupe de sécurité pour la machine. Allons donc créer le groupe de sécurité dans le main.tf.

```
resource "aws_security_group" "instance_sg" {
  name = "terraform-test-sg"
```

```
    egress {
        from_port       = 0
        to_port         = 0
        protocol        = "-1"
        cidr_blocks     = ["0.0.0.0/0"]
    }

    ingress {
        from_port = 80
        to_port = 80
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

    ingress {
        from_port = 22
        to_port = 22
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

}
```

Maintenant que nous avons notre groupe de sécurité, nous pouvons créer notre machine. Mais puisque nous aurons par la suite besoin d'approvisionner notre machine, on va générer. Sur la ligne de commande taper la commande ci-dessous

```
ssh-keygen -t rsa


Renseigner le nom du fichiers et laisser vide passphrase
```

Maintenant vous devez ajouter la clé publique(.pub) à votre compte aws. Pour cela placer vous au chemin de la clé et taper la commande :

```
cat nom_cle.pub
```

Copier la clé et aller sur votre compte aws à la section IAM>utilisateur> sélectionner l'utilisateur concerné, cliquer sur charger une clé SSH, coller la clé et générer

Revenons à notre main.tf, il nous faut une paire de clé

```
resource "aws_key_pair" "my_ec2" {
    key_name   = "terraform-key"
    public_key = file(var.ssh_key)
}
```

Maintenant nous allons créer une variable ami(image de la machine) dans le fichier vars.tf. Les ami sont disponibles sur aws et varie en fonction de la région

```
variable "AWS_AMIS" {
    default = "ami-0434abb28cf6cc2b1"
}
```

Enfin nous allons pouvoir créer notre machine virtuelle, y installer apache et déployer un site web.

```
resource "aws_instance" "my_ec2_instance" {
    key_name      = aws_key_pair.my_ec2.key_name
    ami           = var.AWS_AMIS
    instance_type = "t2.micro"
    vpc_security_group_ids = [aws_security_group.instance_sg.id]
    connection {
        type        = "ssh"
        user        = "ubuntu"
        private_key = file(var.private_key)
        host        = self.public_ip
     }
    provisioner "file" {
         source      = "./index.html"
         destination = "/tmp/index.html"

    }

    provisioner "remote-exec" {
        inline = [
          "sudo apt-get -f -y update",
          "sudo apt-get install -f -y apache2",
          "sudo systemctl start apache2",
          "sudo systemctl enable apache2",
          "sudo cp /tmp/index.html /var/www/html",
        ]

    }
}
```

Explication de code :

Nous avons crée une ressource aws_key_pair requise pour les connexions SSH sur notre instance EC2 nous spécifions d'abord le nom de la paire de clés ainsi que notre clé publique créée précédemment afin d'autoriser notre machine locale à se connecter sur notre instance ec2.

Ensuite, nous imbriquons le bloc connection en spécifiant qu'on souhaite se connecter avec le protocole ssh sur la machine cible avec l'utilisateur ubuntu (utilisateur par défaut sur les AMIs Ubuntu) et notre clé privée créée précédemment. Nous sommes également obligés de fournir

l'adresse de la ressource à laquelle se connecter dans l'argument host, cependant les blocs connection ne peuvent pas faire référence à leurs ressources parent par leur nom. Au lieu de cela, nous utilisons l'objet spécial self qui représente la ressource parent de la connexion et possède tous les arguments de cette ressource. Dans notre cas nous l'utilisons pour récupérer l'adresse IP publique de notre instance EC2.

Maintenant que nous avons déployé notre site, nous aurons certainement besoin d'y accéder, pour cela nous allons créer un output dans le fichier output.tf qui va récupérer notre adresse publique et l'afficher à notre terminale.

C'est fait vous pouvez maintenant lancer votre projet en mettant les deux commandes suivantes.

```
Terraform plan

Terraform apply
```

Pour détruire les ressources, mettez la commande suivante :

```
terraform destroy
```