Terraform Modul Praxis

Praktische Anleitung zur Terraform-Provisionierung

Einleitung

Diese Schritt-für-Schritt-Anleitung zeigt, wie Terraform-Module genutzt werden können, um eine AWS-Infrastruktur bereitzustellen.

Das Ziel besteht darin:

- Eine EC2-Instanz mit zugehöriger Security Group
- S3-Bucket mit Verschlüsselungsoptionen
- Eine IAM-Rolle für die EC2-Instanz mit Zugriff auf den S3-Bucket zu erstellen

Alles unter der Verwendung von Modulen!

1. Projektstruktur erstellen

Erstelle einen Ordner für dein Terraform-Projekt und organisiere die Dateien wie folgt:

```
terraform-projekt/
|-- main.tf
|-- variables.tf
|-- outputs.tf
|-- versions.tf
|-- modules/
|   |-- ec2/
|   |   |-- main.tf
|   |   |-- variables.tf
|   |   |-- outputs.tf
|   |-- s3/
|   |   |-- main.tf
|   |   |-- variables.tf
|   |   |-- outputs.tf
|   |-- iam/
|   |   |-- main.tf
|   |   |-- variables.tf
|   |   |-- outputs.tf
```

- Unter Linux, kann folgender Befehl verwendet werden:

```
mkdir -p terraform-projekt/modules/{ec2,s3,iam}
touch terraform-projekt/{main.tf,variables.tf,outputs.tf,versions.tf}
touch terraform-projekt/modules/{ec2,s3,iam}/{main.tf,variables.tf,outputs.tf,versions.tf}
```

- Öffne den Projektordner in deinem Text Editor

```
code terraform-projekt
```

Versions für jedes Sub Modul

Füge den folgenden Inhalt in jede Sub Modul versions.tf ein (modules/ec2, modules/s3, module/s3)

```
terraform {
  required_version = ">= 1.0"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}
```

2. EC2-Modul erstellen

Editiere die main.tf Datei des EC2 Moduls:

```
resource "aws_security_group" "ec2_sg" {
  name        = "ec2_security_group"
  description = "Security Group for EC2 Instance"

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

  egress {
    description = "Allow all outbound traffic"
    from_port = 0
    to_port = 0
    protocol = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

}

resource "aws_instance" "ec2_instance" {
  ami             = var.ami_id
  instance_type   = var.instance_type
  security_groups = [aws_security_group.ec2_sg.name]

  iam_instance_profile = var.instance_profile == "" ? null : var.instance_profile
}
```

variables.tf

```
variable "ami_id" {
  type = string
  default = "ami-0a485299eeb98b979"
}

variable "instance_type" {
  type = string
  default = "t2.micro"
}

variable "instance_profile" {
  type = string
  default = ""
}
```

outputs.tf

```
output "instance_id" {
  value = aws_instance.ec2_instance.id
}
```

```
output "instance_arn" {
  value = aws_instance.ec2_instance.arn
}
```

3. Main Modul

main.tf

```
module "ec2" {
  source = "./modules/ec2"
}
```

variables.tf

```
variable "region" {
  type = string
  default = "eu-central-1"
}
```

versions.tf

```
terraform {
  required_version = ">= 1.0"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region = var.region
  profile = "techstarter" # BITTE DEIN AWS PROFILE EINTRAGEN
}
```

Erster Test

Im Ordner `terraform-projekt` führe folgende Befehle aus:

```
terraform init
terraform plan
terraform apply
```

4. S3 Modul

main.tf

```
resource "aws_s3_bucket" "bucket" {
  bucket = var.bucket_name
}

resource "aws_s3_bucket_server_side_encryption_configuration" "bucket" {
  count = var.bucket_encryption_enabled == true ? 1 : 0

  bucket = aws_s3_bucket.bucket.id

  rule {
    apply_server_side_encryption_by_default {
      sse_algorithm     = "aws:kms"
    }
  }
}
```

variables.tf

```
variable "bucket_name" {
  type = string
}

variable "bucket_encryption_enabled" {
  type = bool
  default = true
}
```

outputs.tf

```
output "bucket_arn" {
  value = aws_s3_bucket.bucket.arn
}
```

5. Main Modul

main.tf: Am Ende einfügen

```
module "s3" {
  source = "./modules/s3"

  bucket_name = "my-s3-bucket-uo1331iou111"
  bucket_encryption_enabled = true
}
```

Test

Da ein neues Modul hinzugefügt wurde, müssen wir Terraform erneut initialisieren

```
terraform init
terraform plan
terraform apply
```

6. IAM Modul

main.tf

```
resource "aws_iam_role" "role" {
  name = var.role_name
  path = "/"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Sid    = ""
        Principal = {
          Service = "ec2.amazonaws.com"
        }
      },
    ]
  })
}

resource "aws_iam_role_policy" "role_policy" {
  name = "${var.role_name}-policy"
  role = aws_iam_role.role.id

  # Terraform's "jsonencode" function converts a
  # Terraform expression result to valid JSON syntax.
  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = var.policy_actions
        Effect   = var.policy_effect
        Resource = var.policy_resources
      },
    ]
  })
}

resource "aws_iam_instance_profile" "profile" {
  name = "${var.role_name}-profile"
  role = aws_iam_role.role.name
}
```

variables.tf

```
variable "role_name" {
  type = string
  default = "my_ec2_instance_role"
}

variable "policy_actions" {
  type = list(string)
}

variable "policy_effect" {
  type = string
  default = "Allow"
}

variable "policy_resources" {
  type = list(string)
}
```

outputs.tf

```
output "instance_role_name" {
  value = aws_iam_instance_profile.profile.name
}

output "role_arn" {
  value = aws_iam_role.role.arn
}
```

7. Main Modul

main.tf

```
module "ec2" {
  source = "./modules/ec2"

  instance_profile = module.role.instance_role_name
}

module "s3" {
  source = "./modules/s3"

  bucket_name = "my-s3-bucket-uo1331iou111"
  bucket_encryption_enabled = true
}

module "role" {
  source = "./modules/iam"

  role_name = "MyEC2InstanceRole"
  policy_actions = [ "s3:*" ]
  policy_effect = "Allow"
  policy_resources = [module.s3.bucket_arn, "${module.s3.bucket_arn}/*"]
}
```