1. `main.tf`
2. `variable.tf`
3. `terraform.tfvars`
4. `output.tf`

📘 README for Repository 1 – Terraform Scripts

📄 Overview

This Terraform project automates the provisioning of an **EC2 instance in AWS**. It includes all necessary configurations like region, AMI, instance type, tags, and key pair. The structure is modular and follows best practices with variables and outputs.

📁 File Descriptions

1. `main.tf`

This is the core file where actual resource creation happens.

- **Provider Block**

```
provider "aws" {
  region = var.region
}
```

  → This tells Terraform to use **AWS as the cloud provider** and sets the **region** dynamically via variables.

- **Resource Block - AWS Instance**

```
resource "aws_instance" "project_instance" {
  ami           = var.ami
  instance_type = var.instance_type
  key_name      = var.key_name
  tags = {
    Name = var.instance_name
```

```
  }
}
```

→ This block creates an **EC2 instance**:

- `ami` : The machine image ID (e.g., Amazon Linux)
- `instance_type` : Type of instance (e.g., `t2.micro`)
- `key_name` : SSH key name to access the instance
- `tags` : Naming the instance

---

2. `variable.tf`

This defines the input variables used in the Terraform configuration.

```
variable "region" {}
variable "ami" {}
variable "instance_type" {}
variable "instance_name" {}
variable "key_name" {}
```

→ These variables allow flexibility to reuse this code with different values without changing the core logic.

---

3. `terraform.tfvars`

This file provides **values for the input variables** defined in `variable.tf` .

```
region         = "us-east-1"
ami            = "ami-0c55b159cbfafe1f0"
instance_type  = "t2.micro"
instance_name  = "My-EC2-Instance"
key_name       = "devops-key"
```

→ This is the file you edit when you want to deploy in another region, or with another AMI, etc.

---

4. `output.tf`

This file defines what information Terraform will display after applying the configuration.

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

→ After successful deployment, Terraform will **print the EC2 instance ID** to the console.

🚀 How to Use

1. **Initialize the directory**

   ```
   terraform init
   ```

2. **Preview what will be created**

   ```
   terraform plan
   ```

3. **Apply and create the resources**

   ```
   terraform apply
   ```

4. **To destroy everything**

   ```
   terraform destroy
   ```

🛠️ Prerequisites

- AWS account with access credentials configured
- Terraform installed
- A valid EC2 key pair in the AWS region (e.g., `devops-key`)
- IAM permissions to create EC2 instances

✅ Output Example

After running `terraform apply`, you'll see:

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

instance_id = "i-0123456789abcdef0"
```
