Experiment 5
Terraform Variables with Command Line Arguments
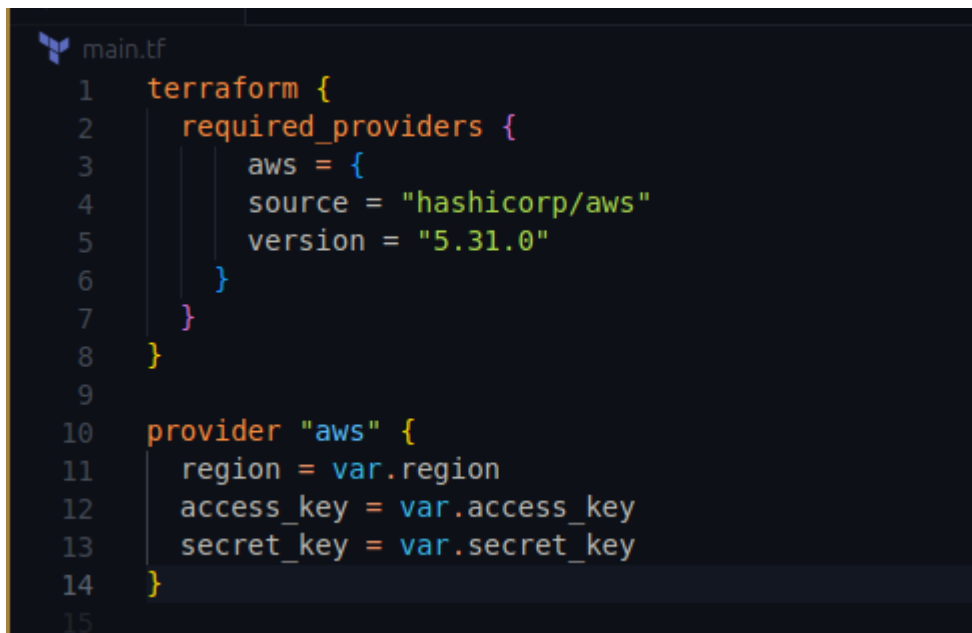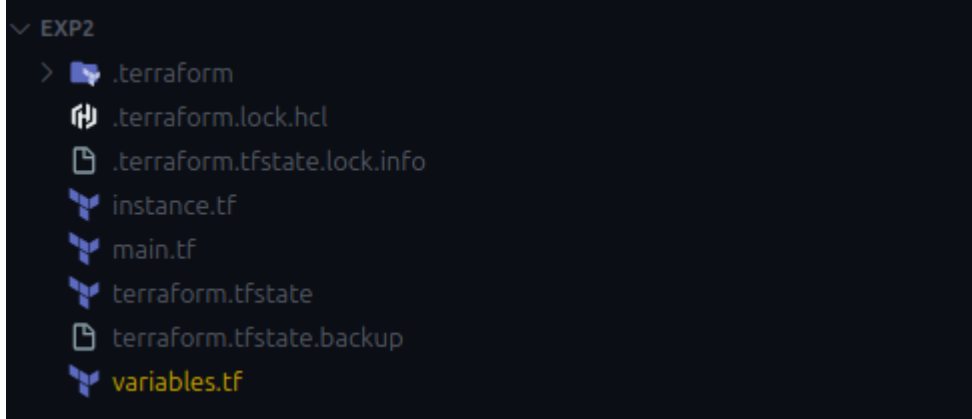
Aim

Learn how to pass values to Terraform variables using command line arguments.

Steps

1. Create a main file & terraform configuration file for EC2 Instance (instance.tf) & add variables to them.

```
∨ EXP2
  > 🟪 .terraform
    🔷 .terraform.lock.hcl
    📄 .terraform.tfstate.lock.info
    🔷 instance.tf
    🔷 main.tf
    🔷 terraform.tfstate
    📄 terraform.tfstate.backup
    🔷 variables.tf
```

```
🔷 main.tf
1    terraform {
2      required_providers {
3          aws = {
4          source = "hashicorp/aws"
5          version = "5.31.0"
6        }
7      }
8    }
9
10   provider "aws" {
11     region = var.region
12     access_key = var.access_key
13     secret_key = var.secret_key
14   }
15
```

```
instance.tf
1    resource "aws_instance" "Ayroid-ec2" {
2        instance_type = var.instance_type
3        ami = var.ami
4        count = 1
5
6        tags = {
7            Name = "Exp5-Instance"
8        }
9    }
```

2. Open a new file named variables.tf. Define variables for region, ami, secret_key, access_key and instance_type.

```
variables.tf
1    variable region {
2      type        = string
3      default     = "ap-south-1"
4      description = "AWS Region"
5    }
6
7    variable "ami"{
8        type = string
9        default = "ami-03f4878755434977f"
10       description = "AMI ID"
11    }
12
13   variable "instance_type"{
14       type = string
15       default = "t2.micro"
16       description = "Instance Type"
17   }
18
```

3. Run the following Terraform commands to initialize and apply the configuration & pass variables as command line arguments.

```
o → Exp2 terraform apply -var 'region=ap-south-1' -var 'ami=ami-03f4878755434977f' -var 'instance_ty
pe=t2.micro'

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_instance.Ayroid-ec2[0] will be created
  + resource "aws_instance" "Ayroid-ec2" {
```

4. Verify resources

	Name	▽	Instance ID	Instance state	▽	Instance type	▽	Status check
☐	Exp5-Instance		i-053e9d2e65fe5dad4	⊘ Running ⊕ ⊖		t2.micro		⊕ Initializing

5. Cleanup resources

```
Plan: 0 to add, 0 to change, 1 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

aws_instance.Ayroid-ec2[0]: Destroying... [id=i-0ef2442dc277b6b86]
aws_instance.Ayroid-ec2[0]: Still destroying... [id=i-0ef2442dc277b6b86, 10s elapsed]
aws_instance.Ayroid-ec2[0]: Still destroying... [id=i-0ef2442dc277b6b86, 20s elapsed]
aws_instance.Ayroid-ec2[0]: Still destroying... [id=i-0ef2442dc277b6b86, 30s elapsed]
aws_instance.Ayroid-ec2[0]: Destruction complete after 32s

Destroy complete! Resources: 1 destroyed.
```