Lab Exercise 7– Terraform Variables with Command Line Arguments

Objective:

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

Prerequisites:

- Terraform installed on your machine.
- Basic knowledge of Terraform variables.

Steps:

1. Create a Terraform Directory:

```
mkdir terraform-cli-variables

cd terraform-cli-variables
```

2. Create Terraform Configuration Files:

- Create a file named main.tf:

main.tf

```
resource "aws_instance" "example" {
  ami          = var.ami
  instance_type = var.instance_type
}
```

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "5.83.1"
    }
  }
}

provider "aws" {
  region     = "ap-south-1"  # Replace with your preferred region
  access_key = "AKIAW77ZKJHOTRG6PTJ7"  # Replace with your Access Key
  secret_key = "DE2+LgyS/bZooajs/mEB9C2vzd2ujLM1wO2cVdAs"  # Replace with your Secret Key
}

resource "aws_instance" "myinstance-1" {
  ami           = var.myami
  instance_type = var.my_instance_type
  count         = var.mycount

  tags = {
    Name = "dhruv Instance"
  }
}
```

- Create a file named variables.tf:

variables.tf

```
variable "ami" {
  description = "AMI ID"
  default     = " ami-08718895af4dfa033"
}

variable "instance_type" {
  description = "EC2 Instance Type"
  default     = "t2.micro"
}
```

lab-7 > variables.tf > variable "my_instance_type"

```
variable "myami" {
  description = "AMI ID"
  default     = "ami-053b12d3152c0cc71"
}
variable "my_instance_type" {
  description = "EC2 Instance Type"
  default     = "t2.micro"
}
variable "mycount" {
  description = "Number of instances to create"
  default     = 2
}
```

3. Use Command Line Arguments:

- Open a terminal and navigate to your Terraform project directory.
- Run the terraform init command:

```
terraform init
```

- Run the terraform apply command with command line arguments to set variable values:

```
dhruvchaubey@Mac lab-6 % terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.83.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
dhruvchaubey@Mac lab-6 % terraform validate
Success! The configuration is valid.
```

terraform plan -var="ami=ami-0522ab6e1ddcc7055" -var="instance_type=t3.micro"

```
dhruvchaubey@Dhruvs-MacBook-Pro lab-7 % terraform plan -var="myami=ami-0522ab6e1
ddcc7055" -var="my_instance_type=t2.micro" -var="mycount=2"


Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create
```

- Adjust the values based on your preferences.

```
  Enter a value: yes

aws_instance.myinstance-1[1]: Creating...
aws_instance.My-instance[0]: Creating...
aws_instance.myinstance-1[0]: Creating...
aws_instance.My-instance[0]: Still creating... [10s elapsed]
aws_instance.myinstance-1[1]: Still creating... [10s elapsed]
aws_instance.myinstance-1[0]: Still creating... [10s elapsed]
aws_instance.myinstance-1[1]: Creation complete after 13s [id=i-055f7544da0a04e6
2]
aws_instance.myinstance-1[0]: Creation complete after 13s [id=i-02fb9638dcf918fc
a]
aws_instance.My-instance[0]: Creation complete after 13s [id=i-03e85f48f1de7e3e5
]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
dhruvchaubey@Dhruvs-MacBook-Pro lab-7 %
```

4. Test and Verify:

- Observe how the command line arguments dynamically set the variable values during the apply process.
- Access the AWS Management Console or use the AWS CLI to verify the creation of resources in the specified region.

Name	Instance ID	Instance state	Instance type
dhruv Instance	i-02fb9638dcf918fca	Running	t2.micro
dhruv-instance	i-03e85f48f1de7e3e5	Running	t2.micro
dhruv Instance	i-055f7544da0a04e62	Running	t2.micro

5. Clean Up:

After testing, you can clean up resources:

terraform destroy

```
aws_instance.myinstance-1[0]: Destruction complete after 1m31s

Destroy complete! Resources: 3 destroyed.
```

Confirm the destruction by typing yes.

6. Conclusion:

This lab exercise demonstrates how to use command line arguments to set variable values dynamically during the terraform apply process. It allows you to customize your Terraform deployments without modifying the configuration files directly. Experiment with different variable values and observe how command line arguments impact the infrastructure provisioning process.