

## 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:

# instance.tf

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

```
lab7 > instance.tf > ...
1 resource "aws_instance" "example" {
2     ami           = var.ami
3     instance_type = var.instance_type
4 }
5
```

- 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"
}
```

```
main.tf lab3  main.tf lab5  main.tf lab6  instance.tf lab6  variables.tf lab6  main.tf lab7 X  va
lab7 > main.tf > provider "aws" > secret_key
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 = "ap-south-1" # Replace with your preferred region
12     access_key = "AKIA6H7MTJGKF33TD7OW" # Replace with your Access Key
13     secret_key = "ASmjpgc9py927gaf7W/+e0Q3uKpVb0npcfIAf0n1l" # Replace with your Secret Key
14 }
```

### 3. Use Command Line Arguments:

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

terraform init

**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.

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

terraform plan -var="ami=ami-0522ab6e1ddcc7055" -var="instance\_type=t3.micro"

PS C:\Github Repositores\Terraform-Demo\lab7> terraform plan -var="ami=ami-00bb6a80f01f03502" -var="instance\_type=t3.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.example will be created
+ resource "aws_instance" "example" {
  + ami               = "ami-00bb6a80f01f03502"
  + arn               = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone  = (known after apply)
  + cpu_core_count     = (known after apply)
  + cpu_threads_per_core = (known after apply)
```

- Adjust the values based on your preferences.

### 4. Test and Verify:

- Observe how the command line arguments dynamically set the variable values during the apply process.

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






Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Creation complete after 12s [id=i-060723b8fce6aa915]

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

- Access the AWS Management Console or use the AWS CLI to verify the creation of resources in the specified region.

Details	Status and alarms	Monitoring	Security	Networking	Storage	Tags
▼ Instance details <a href="#">Info</a>						
AMI ID  ami-00bb6a80f01f03502		Monitoring disabled		Platform details  Linux/UNIX		
AMI name  ubuntu/images/hvm-ssd-gp3/ubuntu-noble-24.04-amd64-server-20250115		Allowed image -		Termination protection Disabled		
Stop protection Disabled		Launch time  Thu Jan 16 2025 22:35:18 GMT+0530 (India Standard Time) (1 minute)		AMI location  amazon/ubuntu/images/hvm-ssd-gp3/ubuntu-noble-24.04-amd64-server-20250115		

## 5. Clean Up:

After testing, you can clean up resources:

### terraform destroy

```
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.example: Destroying... [id=i-060723b8fce6aa915]
aws_instance.example: Still destroying... [id=i-060723b8fce6aa915, 10s elapsed]
aws_instance.example: Still destroying... [id=i-060723b8fce6aa915, 20s elapsed]
aws_instance.example: Still destroying... [id=i-060723b8fce6aa915, 30s elapsed]
aws_instance.example: Still destroying... [id=i-060723b8fce6aa915, 40s elapsed]
aws_instance.example: Still destroying... [id=i-060723b8fce6aa915, 50s elapsed]
aws_instance.example: Still destroying... [id=i-060723b8fce6aa915, 1m0s elapsed]
aws_instance.example: Destruction complete after 1m0s

Destroy complete! Resources: 1 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.