# 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
```

```
(base) →  MyLab mkdir terraform-cli-variables
cd terraform-cli-variables

(base) →  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
}
(base) → terraform-cli-variables cat instance.tf
resource "aws_instance" "example" {
  ami           = var.ami
  instance_type = var.instance_type
}
(base) → terraform-cli-variables _
```

- 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"
}
```

```
▶(base) →  terraform-cli-variables cat variables.tf
variable "ami" {
  description = "AMI ID"
  default     = "ami-08718895af4dfa033"
}


variable "instance_type" {
  description = "EC2 Instance Type"
  default     = "t2.micro"
}
▶(base) →  terraform-cli-variables _
```

## 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:

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

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


Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these
actions if you run "terraform apply" now.
▶(base) →  terraform-cli-variables _
```
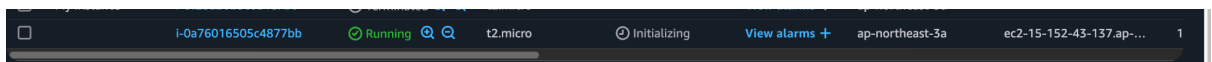
- Adjust the values based on your preferences.

## 4. Test and Verify:

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

- 

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



- 

## 5. Clean Up:

After testing, you can clean up resources:

```
terraform destroy

aws_instance.example: Destroying... [id=i-0a76016505c4877bb]
aws_instance.example: Still destroying... [id=i-0a76016505c4877bb, 10s elapsed]
aws_instance.example: Still destroying... [id=i-0a76016505c4877bb, 20s elapsed]
aws_instance.example: Still destroying... [id=i-0a76016505c4877bb, 30s elapsed]
aws_instance.example: Still destroying... [id=i-0a76016505c4877bb, 40s elapsed]
aws_instance.example: Still destroying... [id=i-0a76016505c4877bb, 50s elapsed]
aws_instance.example: Still destroying... [id=i-0a76016505c4877bb, 1m0s elapsed]
aws_instance.example: Destruction complete after 1m8s

Destroy complete! Resources: 1 destroyed.
(base) →  terraform-cli-variables _
```

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.