

## Lab Exercise 6– Terraform Variables

### Objective:

Learn how to define and use variables in Terraform configuration.

### Prerequisites:

- Install Terraform on your machine.

### Steps:

#### 1. Create a Terraform Directory:

- Create a new directory for your Terraform project.

```
mkdir terraform-variables
cd terraform-variables
```

#### 2. Create a Terraform Configuration File:

- Create a file named main.tf within your project directory.

**# main.tf**

```
resource "aws_instance" "myinstance-1" {
  ami = var.myami
  instance_type = var.my_instance_type
  count = var.mycount
  tags = {
    Name= "My Instance"
  }
}
```

### 3. Define Variables:

- Open a new file named variables.tf. Define variables for region, ami, and instance\_type.

**# variables.tf**

```
variable "myami" {  
  type = string  
  default = "ami-08718895af4dfa033"  
}  
  
variable "mycount" {  
  
  type = number  
  default = 5  
}  
  
variable "my_instance_type" {  
  type = string  
  default = "t2.micro"  
}
```

### 4. Initialize and Apply:

- Run the following Terraform commands to initialize and apply the configuration.

**terraform init**

```
Administrator: Command Prompt
Destroy complete! Resources: 1 destroyed.

D:\Repositories\Terraform-S3-Demo>cd ..

D:\Repositories>mkdir terraform-variables

D:\Repositories>cd terraform-variables

D:\Repositories\terraform-variables>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.14.0...
- Installed hashicorp/aws v6.14.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.

D:\Repositories\terraform-variables>
```

## terraform plan

## terraform apply -auto-approve

```
Administrator: Command Prompt
{
  + tenancy                               = (known after apply)
  + user_data_base64                     = (known after apply)
  + user_data_replace_on_change          = false
  + vpc_security_group_ids               = (known after apply)

  + capacity_reservation_specification (known after apply)

  + cpu_options (known after apply)

  + ebs_block_device (known after apply)

  + enclave_options (known after apply)

  + ephemeral_block_device (known after apply)

  + instance_market_options (known after apply)

  + maintenance_options (known after apply)

  + metadata_options (known after apply)

  + network_interface (known after apply)

  + primary_network_interface (known after apply)

  + private_dns_name_options (known after apply)

  + root_block_device (known after apply)
}

Plan: 8 to add, 0 to change, 0 to destroy.
aws_instance.myinstance[1]: Creating...
aws_instance.myinstance[0]: Creating...
aws_instance.myinstance[2]: Creating...
aws_instance.myinstance[3]: Creating...
aws_instance.myinstance[4]: Creating...
aws_instance.myinstance[5]: Creating...
aws_instance.myinstance[6]: Still creating... [0m0s elapsed]
aws_instance.myinstance[7]: Still creating... [0m0s elapsed]
aws_instance.myinstance[8]: Still creating... [0m0s elapsed]
aws_instance.myinstance[9]: Still creating... [0m0s elapsed]
aws_instance.myinstance[10]: Still creating... [0m0s elapsed]
aws_instance.myinstance[11]: Still creating... [0m0s elapsed]
aws_instance.myinstance[12]: Creation complete after 10s [id=ami-0501e8b2a5d7fca3c]
aws_instance.myinstance[13]: Creation complete after 10s [id=ami-0e01e8b2a5d7fca3c]
aws_instance.myinstance[14]: Creation complete after 10s [id=ami-010e423c28430f9dc]
aws_instance.myinstance[15]: Creation complete after 10s [id=ami-02833426a6b0f9dc]
aws_instance.myinstance[16]: Creation complete after 10s [id=ami-0a318db0b0c907839]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

D:\Repositories\terraform-variables>
```

Observe how the region changes based on the variable override.

## 5. Clean Up:

After testing, you can clean up resources.

### terraform destroy

```
tags_all      = {} => null
throughput    = 0  => null
volume_id     = "vol-0186f79c730d88800" => null
volume_size   = 8  => null
volume_type   = "gp3" => null
# (1 untagged attribute hidden)

Plan: 0 to add, 0 to change, 0 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.myinstance[2]: Destroying... [id=i-8286a31c3863d4f16]
aws_instance.myinstance[0]: Destroying... [id=i-82130c3d6d4d4e41d]
aws_instance.myinstance[2]: Destroying... [id=i-8581e8e2a5c7fca1c]
aws_instance.myinstance[1]: Destroying... [id=i-86d1088c7b5a018]
aws_instance.myinstance[4]: Destroying... [id=i-8a2196bdc9f7010]
aws_instance.myinstance[2]: Still destroying... [id=i-8286a31c3863d4f16, 0m00s elapsed]
aws_instance.myinstance[0]: Still destroying... [id=i-82533a28a6d5a16, 0m00s elapsed]
aws_instance.myinstance[2]: Still destroying... [id=i-8581e8e2a5c7fca1c, 0m00s elapsed]
aws_instance.myinstance[1]: Still destroying... [id=i-86d1088c7b5a018, 0m00s elapsed]
aws_instance.myinstance[4]: Still destroying... [id=i-8a2196bdc9f7010, 0m00s elapsed]
aws_instance.myinstance[2]: Still destroying... [id=i-8286a31c3863d4f16, 0m00s elapsed]
aws_instance.myinstance[0]: Still destroying... [id=i-82533a28a6d5a16, 0m00s elapsed]
aws_instance.myinstance[2]: Still destroying... [id=i-8581e8e2a5c7fca1c, 0m00s elapsed]
aws_instance.myinstance[1]: Still destroying... [id=i-86d1088c7b5a018, 0m00s elapsed]
aws_instance.myinstance[4]: Still destroying... [id=i-8a2196bdc9f7010, 0m00s elapsed]
aws_instance.myinstance[2]: Still destroying... [id=i-8286a31c3863d4f16, 0m00s elapsed]
aws_instance.myinstance[0]: Still destroying... [id=i-82533a28a6d5a16, 0m00s elapsed]
aws_instance.myinstance[2]: Still destroying... [id=i-8581e8e2a5c7fca1c, 0m00s elapsed]
aws_instance.myinstance[1]: Still destroying... [id=i-86d1088c7b5a018, 0m00s elapsed]
aws_instance.myinstance[4]: Still destroying... [id=i-8a2196bdc9f7010, 0m00s elapsed]
aws_instance.myinstance[2]: Destruction complete after 30s
aws_instance.myinstance[0]: Destruction complete after 30s
aws_instance.myinstance[2]: Still destroying... [id=i-8581e8e2a5c7fca1c, 0m00s elapsed]
aws_instance.myinstance[1]: Still destroying... [id=i-86d1088c7b5a018, 0m00s elapsed]
aws_instance.myinstance[4]: Still destroying... [id=i-8a2196bdc9f7010, 0m00s elapsed]
aws_instance.myinstance[2]: Still destroying... [id=i-8286a31c3863d4f16, 0m00s elapsed]
aws_instance.myinstance[0]: Still destroying... [id=i-82533a28a6d5a16, 0m00s elapsed]
aws_instance.myinstance[2]: Still destroying... [id=i-8581e8e2a5c7fca1c, 0m00s elapsed]
aws_instance.myinstance[1]: Still destroying... [id=i-86d1088c7b5a018, 0m00s elapsed]
aws_instance.myinstance[4]: Destruction complete after 30s
aws_instance.myinstance[1]: Destruction complete after 30s
aws_instance.myinstance[0]: Destruction complete after 30s

Destroy complete! Resources: 0 destroyed.

D:\Upgrades\terraform-variables>
```

Confirm the destruction by typing yes.

## 6. Conclusion:

This lab exercise introduces you to Terraform variables and demonstrates how to use them in your configurations. Experiment with different variable values and overrides to understand their impact on the infrastructure provisioning process.