Lab Exercise 6– Terraform Variables

Objective:

Learn how to define and use variables in Terraform configuration.

Prerequisites:

- Install Terraform on your machine.

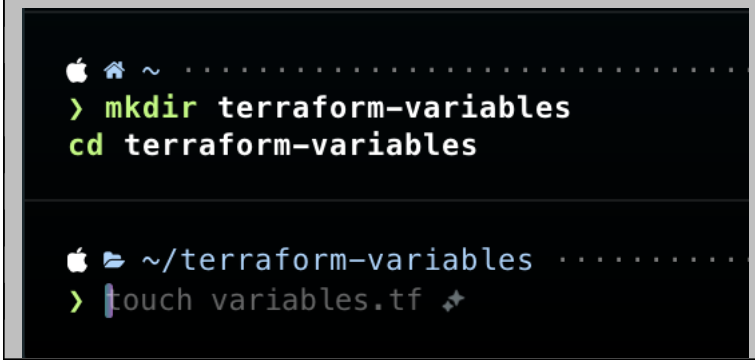
Steps:

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"
  }
}
```

```
~/terraform-variables
❯ vim  main.tf

~/terraform-variables
❯ cat main.tf
resource "aws_instance" "myinstance-1-Ansh" {
    ami = var.myami
    instance_type = var.my_instance_type
    count = var.mycount
    tags = {
      Name= "My Instance"
    }
}

~/terraform-variables
❯ terraform apply
```

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"
}
```

```
~/terraform-variables
❯ vim variables.tf
```

```
~/terraform-variables
❯ cat variables.tf
iivariable "myami" {
  type = string
  default = "ami-00bb6a80f01f03502"
}

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

terraform plan

terraform apply -auto-approve
```

```
~/terraform-variables
❯ terraform init

Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.85.0...
- Installed hashicorp/aws v5.85.0 (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.
```

```
~/terraform-variables
❯ terraform plan
      + monitoring                         = (known after apply)
      + outpost_arn                        = (known after apply)
      + password_data                      = (known after apply)
      + placement_group                    = (known after apply)
      + placement_partition_number         = (known after apply)
      + primary_network_interface_id       = (known after apply)
      + private_dns                        = (known after apply)
      + private_ip                         = (known after apply)
      + public_dns                         = (known after apply)
      + public_ip                          = (known after apply)
      + secondary_private_ips              = (known after apply)
      + security_groups                    = (known after apply)
      + source_dest_check                  = true
      + spot_instance_request_id           = (known after apply)
      + subnet_id                          = (known after apply)
      + tags                               = {
          + "Name" = "My Instance"
        }
      + tags_all                           = {
          + "Name" = "My Instance"
        }
      + tenancy                            = (known after apply)
      + user_data                          = (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)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 5 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.
```

```
~/terraform-variables
❯ terraform apply -auto-approve
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags                                 = {
          + "Name" = "My Instance"
        }
      + tags_all                             = {
          + "Name" = "My Instance"
        }
      + tenancy                              = (known after apply)
      + user_data                            = (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)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 5 to add, 0 to change, 0 to destroy.
aws_instance.myinstance-1-Ansh[0]: Creating...
aws_instance.myinstance-1-Ansh[4]: Creating...
aws_instance.myinstance-1-Ansh[2]: Creating...
aws_instance.myinstance-1-Ansh[1]: Creating...
aws_instance.myinstance-1-Ansh[3]: Creating...
aws_instance.myinstance-1-Ansh[0]: Still creating... [10s elapsed]
aws_instance.myinstance-1-Ansh[4]: Still creating... [10s elapsed]
aws_instance.myinstance-1-Ansh[2]: Still creating... [10s elapsed]
aws_instance.myinstance-1-Ansh[1]: Still creating... [10s elapsed]
aws_instance.myinstance-1-Ansh[3]: Still creating... [10s elapsed]
aws_instance.myinstance-1-Ansh[4]: Creation complete after 12s [id=i-0368b57c39dad6bf3]
aws_instance.myinstance-1-Ansh[3]: Creation complete after 12s [id=i-0bd3b8aee9ed247e1]
aws_instance.myinstance-1-Ansh[0]: Creation complete after 12s [id=i-080af428bb4453e47]
aws_instance.myinstance-1-Ansh[2]: Creation complete after 12s [id=i-0bc2ebacb99b724db]
aws_instance.myinstance-1-Ansh[1]: Creation complete after 12s [id=i-0e1569c0557d61f89]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
```

Observe how the region changes based on the variable override.

5. Clean Up:

After testing, you can clean up resources.

terraform destroy

```
~/terraform-variables
❯ terraform destroy
          - tags_all              = {} -> null
          - throughput            = 125 -> null
          - volume_id             = "vol-00d2ee603445df088" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp3" -> null
            # (1 unchanged attribute hidden)
        }
    }

Plan: 0 to add, 0 to change, 5 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-1-Ansh[4]: Destroying... [id=i-0368b57c39dad6bf3]
aws_instance.myinstance-1-Ansh[3]: Destroying... [id=i-0bd3b8aee9ed247e1]
aws_instance.myinstance-1-Ansh[2]: Destroying... [id=i-0bc2ebacb99b724db]
aws_instance.myinstance-1-Ansh[1]: Destroying... [id=i-0e1569c0557d61f89]
aws_instance.myinstance-1-Ansh[0]: Destroying... [id=i-080af428bb4453e47]
aws_instance.myinstance-1-Ansh[3]: Still destroying... [id=i-0bd3b8aee9ed247e1, 10s elapsed]
aws_instance.myinstance-1-Ansh[0]: Still destroying... [id=i-080af428bb4453e47, 10s elapsed]
aws_instance.myinstance-1-Ansh[4]: Still destroying... [id=i-0368b57c39dad6bf3, 10s elapsed]
aws_instance.myinstance-1-Ansh[1]: Still destroying... [id=i-0e1569c0557d61f89, 10s elapsed]
aws_instance.myinstance-1-Ansh[2]: Still destroying... [id=i-0bc2ebacb99b724db, 10s elapsed]
aws_instance.myinstance-1-Ansh[0]: Still destroying... [id=i-080af428bb4453e47, 20s elapsed]
aws_instance.myinstance-1-Ansh[4]: Still destroying... [id=i-0368b57c39dad6bf3, 20s elapsed]
aws_instance.myinstance-1-Ansh[3]: Still destroying... [id=i-0bd3b8aee9ed247e1, 20s elapsed]
aws_instance.myinstance-1-Ansh[2]: Still destroying... [id=i-0bc2ebacb99b724db, 20s elapsed]
aws_instance.myinstance-1-Ansh[1]: Still destroying... [id=i-0e1569c0557d61f89, 20s elapsed]
aws_instance.myinstance-1-Ansh[4]: Still destroying... [id=i-0368b57c39dad6bf3, 30s elapsed]
aws_instance.myinstance-1-Ansh[1]: Still destroying... [id=i-0e1569c0557d61f89, 30s elapsed]
aws_instance.myinstance-1-Ansh[2]: Still destroying... [id=i-0bc2ebacb99b724db, 30s elapsed]
aws_instance.myinstance-1-Ansh[3]: Still destroying... [id=i-0bd3b8aee9ed247e1, 30s elapsed]
aws_instance.myinstance-1-Ansh[0]: Still destroying... [id=i-080af428bb4453e47, 30s elapsed]
aws_instance.myinstance-1-Ansh[4]: Destruction complete after 30s
aws_instance.myinstance-1-Ansh[2]: Still destroying... [id=i-0bc2ebacb99b724db, 40s elapsed]
aws_instance.myinstance-1-Ansh[3]: Still destroying... [id=i-0bd3b8aee9ed247e1, 40s elapsed]
aws_instance.myinstance-1-Ansh[0]: Still destroying... [id=i-080af428bb4453e47, 40s elapsed]
aws_instance.myinstance-1-Ansh[1]: Still destroying... [id=i-0e1569c0557d61f89, 40s elapsed]
aws_instance.myinstance-1-Ansh[1]: Still destroying... [id=i-0e1569c0557d61f89, 50s elapsed]
aws_instance.myinstance-1-Ansh[0]: Still destroying... [id=i-080af428bb4453e47, 50s elapsed]
aws_instance.myinstance-1-Ansh[3]: Still destroying... [id=i-0bd3b8aee9ed247e1, 50s elapsed]
aws_instance.myinstance-1-Ansh[2]: Still destroying... [id=i-0bc2ebacb99b724db, 50s elapsed]
aws_instance.myinstance-1-Ansh[0]: Destruction complete after 51s
aws_instance.myinstance-1-Ansh[3]: Destruction complete after 51s
aws_instance.myinstance-1-Ansh[1]: Still destroying... [id=i-0e1569c0557d61f89, 1m0s elapsed]
aws_instance.myinstance-1-Ansh[2]: Still destroying... [id=i-0bc2ebacb99b724db, 1m0s elapsed]
aws_instance.myinstance-1-Ansh[2]: Destruction complete after 1m1s
aws_instance.myinstance-1-Ansh[1]: Still destroying... [id=i-0e1569c0557d61f89, 1m10s elapsed]
aws_instance.myinstance-1-Ansh[1]: Destruction complete after 1m11s

Destroy complete! Resources: 5 destroyed.
```

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.