Lab Exercise 15– 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.

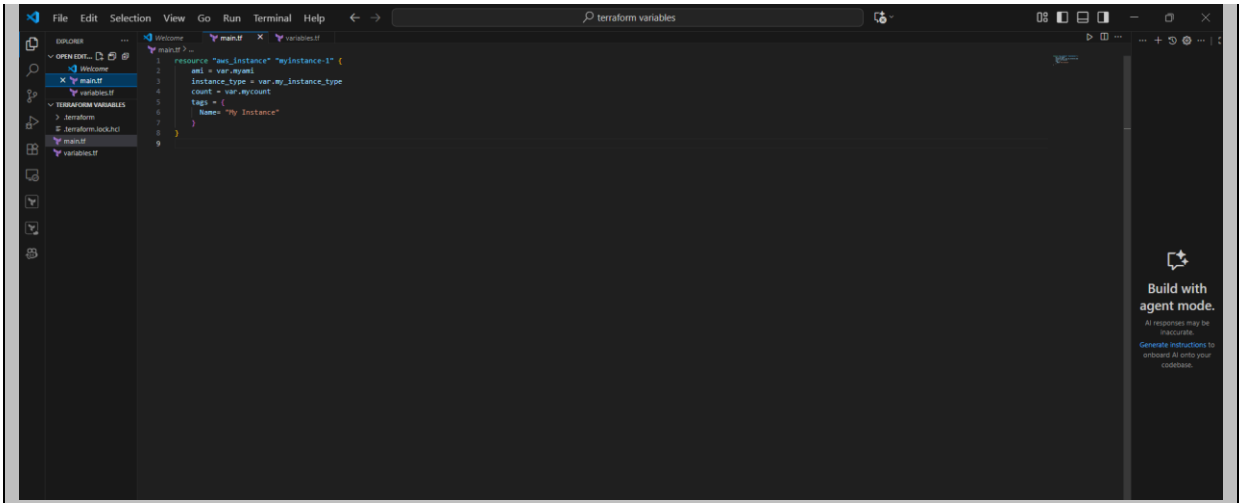
```
mkdir terraform-variables
cd terraform-variables
```

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.

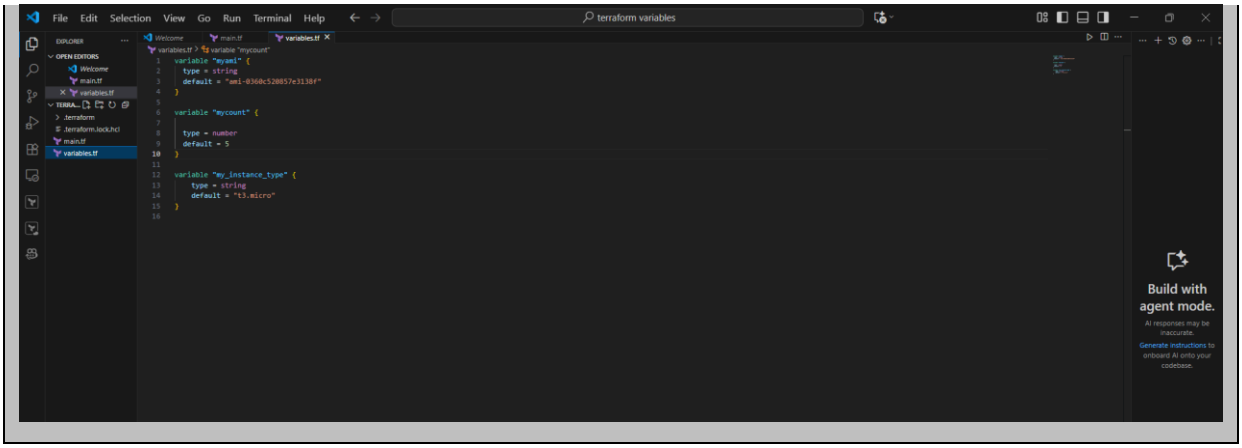
 # variables.tf

```
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

```
C:\terraform\terraform variables>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.15.0...
- Installed hashicorp/aws v6.15.0 (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.

C:\terraform\terraform variables>
```

terraform plan

```
      + 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: 5 to add, 0 to change, 0 to destroy.
```

terraform apply -auto-approve

```
      + 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: 5 to add, 0 to change, 0 to destroy.
aws_instance.myinstance-1[1]: Creating...
aws_instance.myinstance-1[0]: Creating...
aws_instance.myinstance-1[2]: Creating...
aws_instance.myinstance-1[3]: Creating...
aws_instance.myinstance-1[4]: Creating...
aws_instance.myinstance-1[0]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[3]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[1]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[2]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[4]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[0]: Creation complete after 17s [id=i-0e669d1fb37a68ec6]
aws_instance.myinstance-1[1]: Creation complete after 17s [id=i-0f087b0796a090db3]
aws_instance.myinstance-1[3]: Creation complete after 17s [id=i-0e6dec35cdf782ef1]
aws_instance.myinstance-1[2]: Creation complete after 17s [id=i-0c4cda76d0910fd24]
aws_instance.myinstance-1[4]: Creation complete after 17s [id=i-042697a99e625084b]

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

C:\terraform\terraform_variables>
```

Observe how the region changes based on the variable override.

5. Clean Up:

After testing, you can clean up resources.

terraform destroy

```
aws_instance.myinstance-1[0]: Still destroying... [id=i-0e669d1fb37a68ec6, 00m4
aws_instance.myinstance-1[3]: Still destroying... [id=i-0e6dec35cdf782ef1, 00m4
aws_instance.myinstance-1[2]: Still destroying... [id=i-0c4cda76d0910fd24, 00m4
aws_instance.myinstance-1[1]: Still destroying... [id=i-0f087b0796a090db3, 00m4
aws_instance.myinstance-1[3]: Still destroying... [id=i-0e6dec35cdf782ef1, 00m5
aws_instance.myinstance-1[0]: Still destroying... [id=i-0e669d1fb37a68ec6, 00m5
aws_instance.myinstance-1[2]: Still destroying... [id=i-0c4cda76d0910fd24, 00m5
aws_instance.myinstance-1[1]: Still destroying... [id=i-0f087b0796a090db3, 00m5
aws_instance.myinstance-1[0]: Still destroying... [id=i-0e669d1fb37a68ec6, 01m0
aws_instance.myinstance-1[2]: Still destroying... [id=i-0c4cda76d0910fd24, 01m0
aws_instance.myinstance-1[3]: Still destroying... [id=i-0e6dec35cdf782ef1, 01m0
aws_instance.myinstance-1[1]: Still destroying... [id=i-0f087b0796a090db3, 01m0
aws_instance.myinstance-1[1]: Destruction complete after 1m4s
aws_instance.myinstance-1[0]: Destruction complete after 1m4s
aws_instance.myinstance-1[3]: Destruction complete after 1m5s
aws_instance.myinstance-1[2]: Still destroying... [id=i-0c4cda76d0910fd24, 01m1
aws_instance.myinstance-1[2]: Destruction complete after 1m15s

Destroy complete! Resources: 5 destroyed.

C:\terraform\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.