

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
```

1.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"
  }
}
```

1. Define Variables:

Open a new file named variables.tf. Define variables for region, ami, and instance_type.

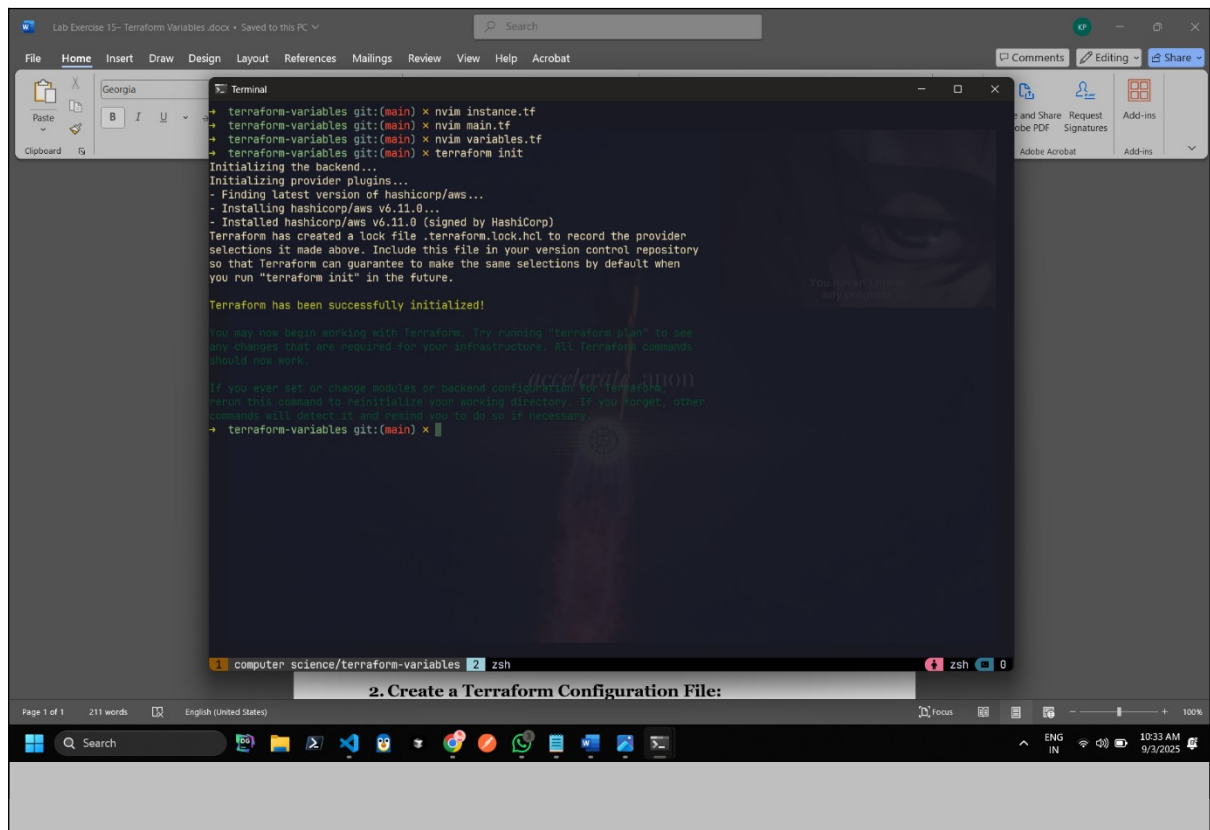
variables.tf

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

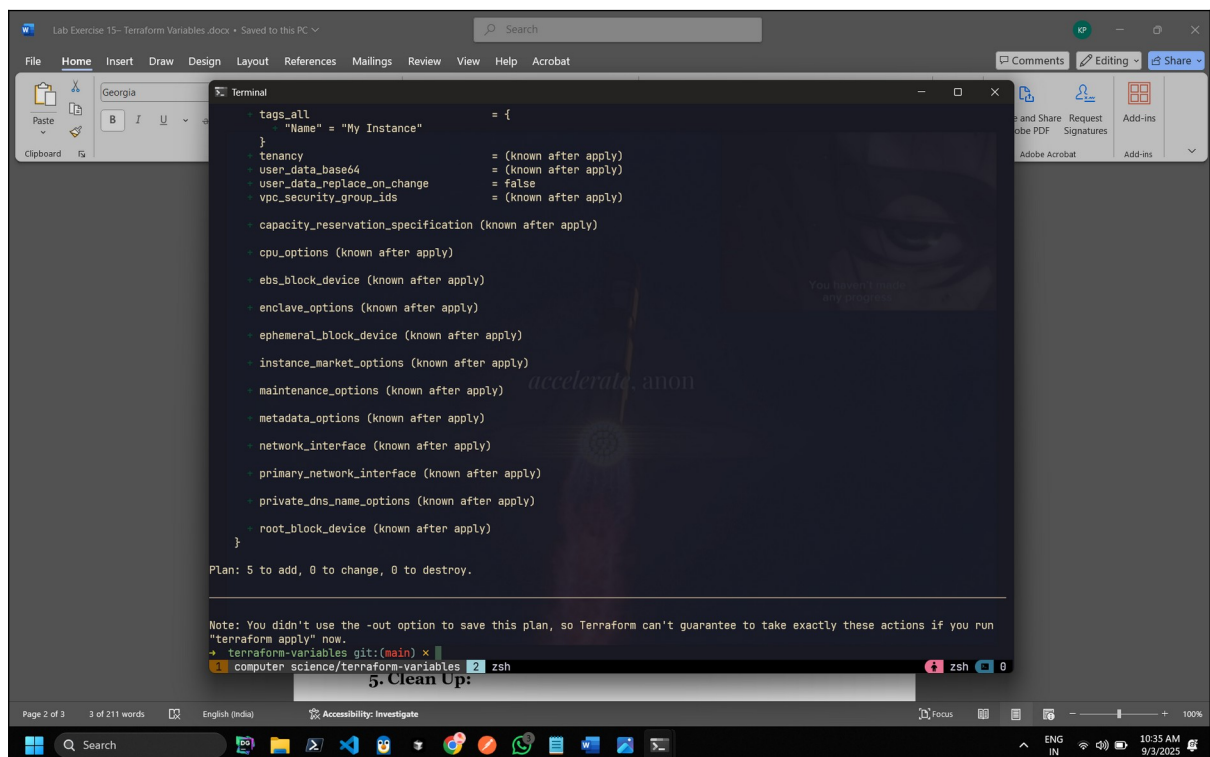
1.Initialize and Apply:

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

```
terraform init
```



terraform plan



terraform apply -auto-approve

```
Terminal
+ 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: 5 to add, 0 to change, 0 to destroy.
aws_instance.myinstance-1[0]: Creating...
aws_instance.myinstance-1[2]: Creating...
aws_instance.myinstance-1[3]: Creating...
aws_instance.myinstance-1[1]: Creating...
aws_instance.myinstance-1[4]: Creating...
aws_instance.myinstance-1[1]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[0]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[2]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[3]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[4]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[3]: Creation complete after 19s [id=i-0e0b456951dc96835]
aws_instance.myinstance-1[0]: Creation complete after 20s [id=i-0b469594b9b83bb1d]
aws_instance.myinstance-1[4]: Still creating... [00m20s elapsed]
aws_instance.myinstance-1[1]: Still creating... [00m20s elapsed]
aws_instance.myinstance-1[2]: Still creating... [00m20s elapsed]
aws_instance.myinstance-1[2]: Creation complete after 21s [id=i-0a9545b1ea6901459]
aws_instance.myinstance-1[4]: Creation complete after 21s [id=i-0c0442e9cb6b9ecc9]
aws_instance.myinstance-1[1]: Creation complete after 21s [id=i-0d87d16bb56e19e3a]

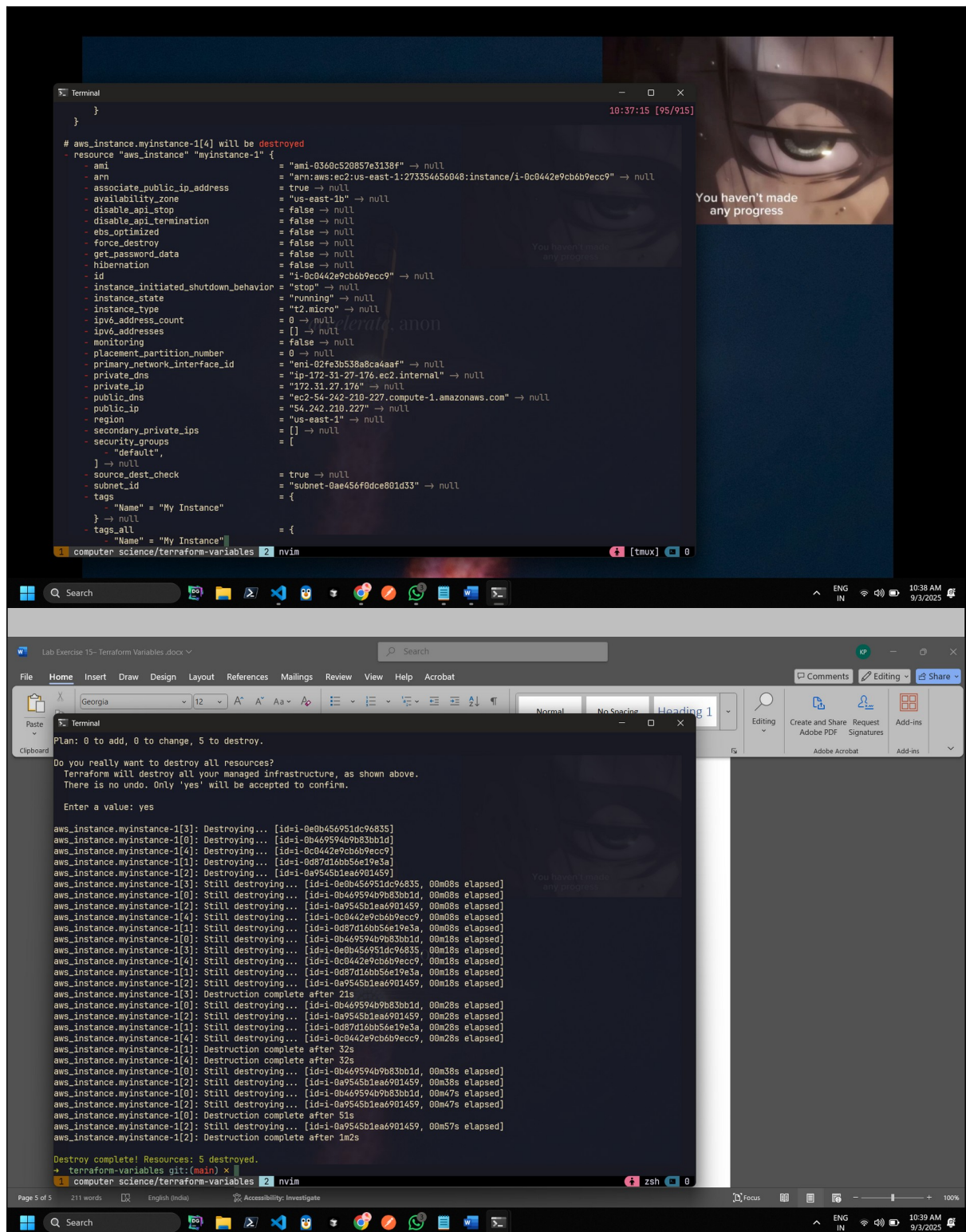
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
+ terraform-variables git:(main) x
computer science/terraform-variables 2 nvim
```

Observe how the region changes based on the variable override.

1.Clean Up:

After testing, you can clean up resources.

terraform destroy



Confirm the destruction by typing yes.

1. 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.