Lab Exercise 15– Terraform Variables

Objective:

Learn how to define and use variables in Terraform configuration.

Prerequisites:

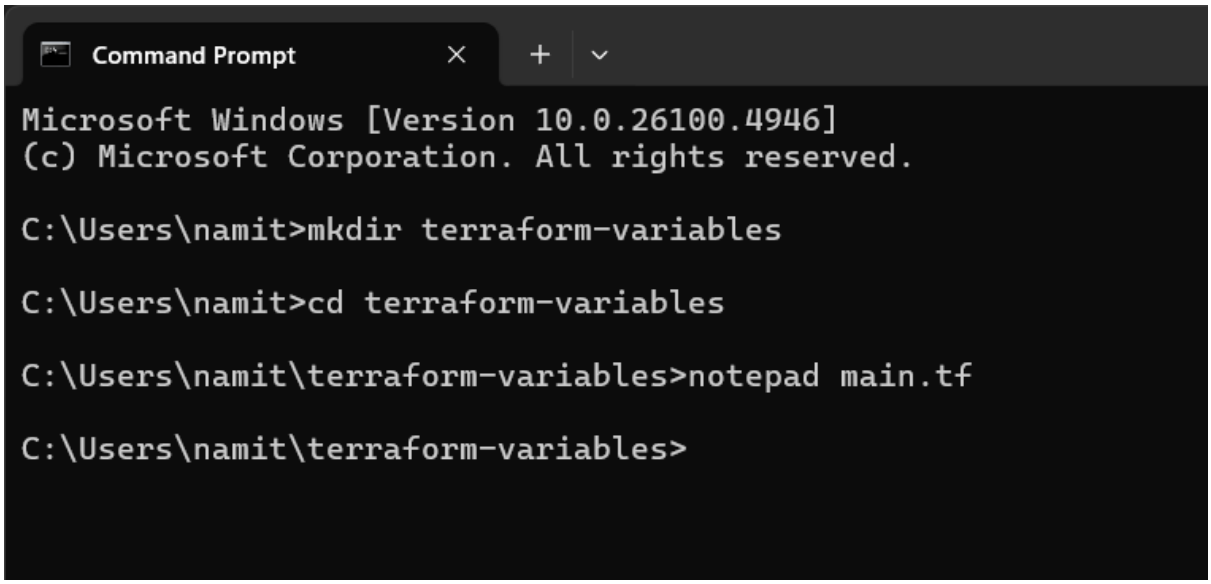
- Install Terraform on your machine.

Steps:

1. Create a Terraform Directory:

- Create a new directory for your Terraform project.

```
mkdir terraform-variables

cd terraform-variables
```

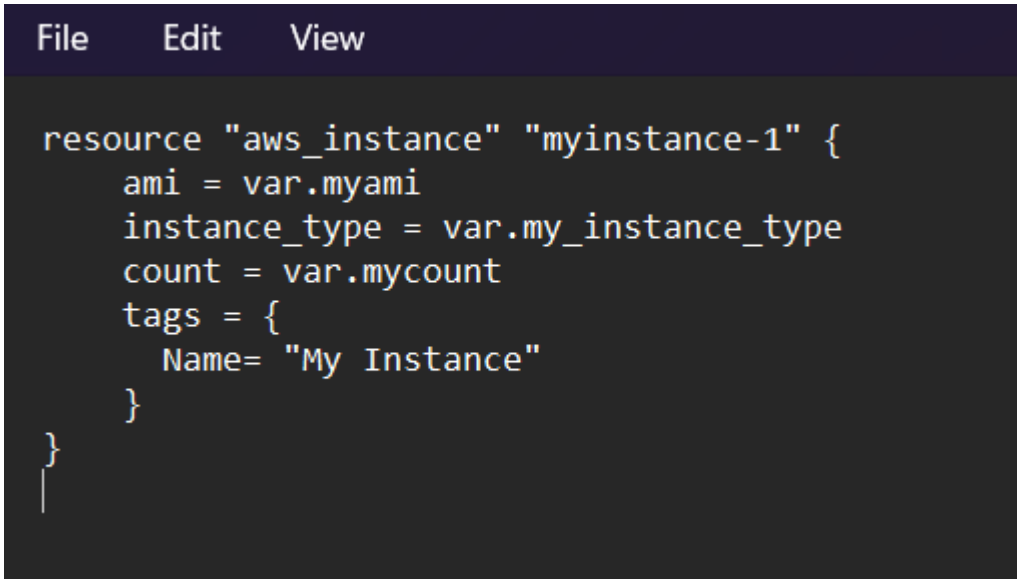


2. Create a Terraform Configuration File:

- 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"
}
```

File Edit View

```
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

terraform plan

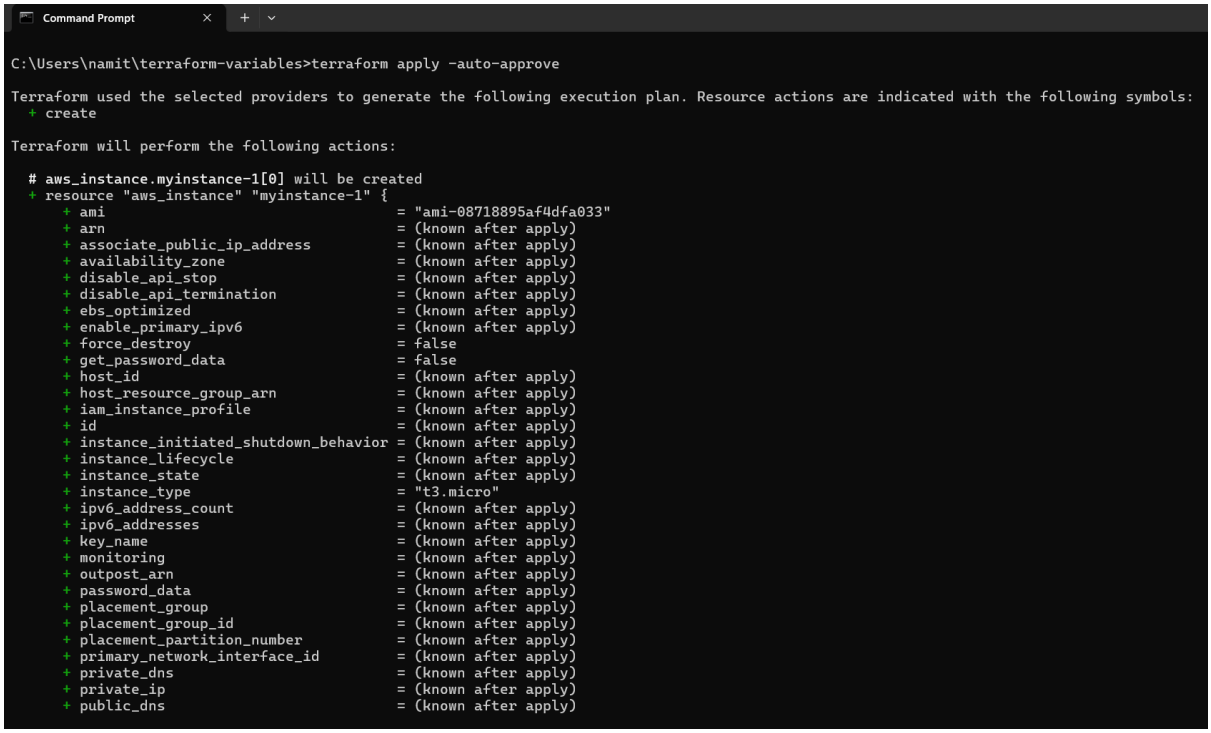
terraform apply -auto-approve

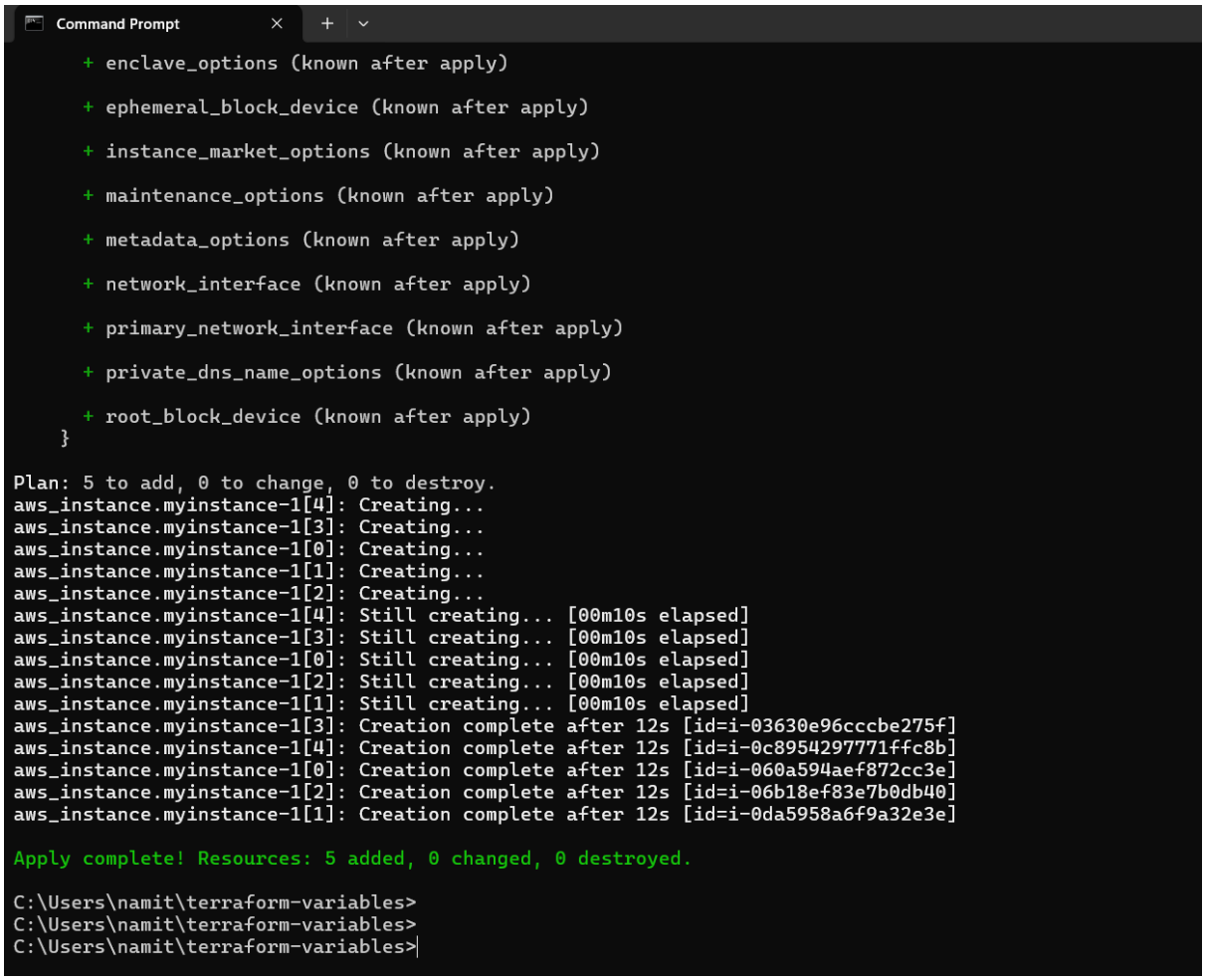
Observe how the region changes based on the variable override.

```
C:\Users\namit\terraform-variables>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.13.0...
- Installed hashicorp/aws v6.13.0 (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:\Users\namit\terraform-variables>
```

```
Command Prompt
C:\Users\namit\terraform-variables>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.myinstance-1[0] will be created
  + resource "aws_instance" "myinstance-1" {
      + ami                                  = "ami-08718895af4dfa033"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + force_destroy                        = false
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_group_id                   = (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)
```

```
Command Prompt
        }
      + tags_all                             = {
          + "Name" = "My Instance"
        }
      + 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: 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.
```

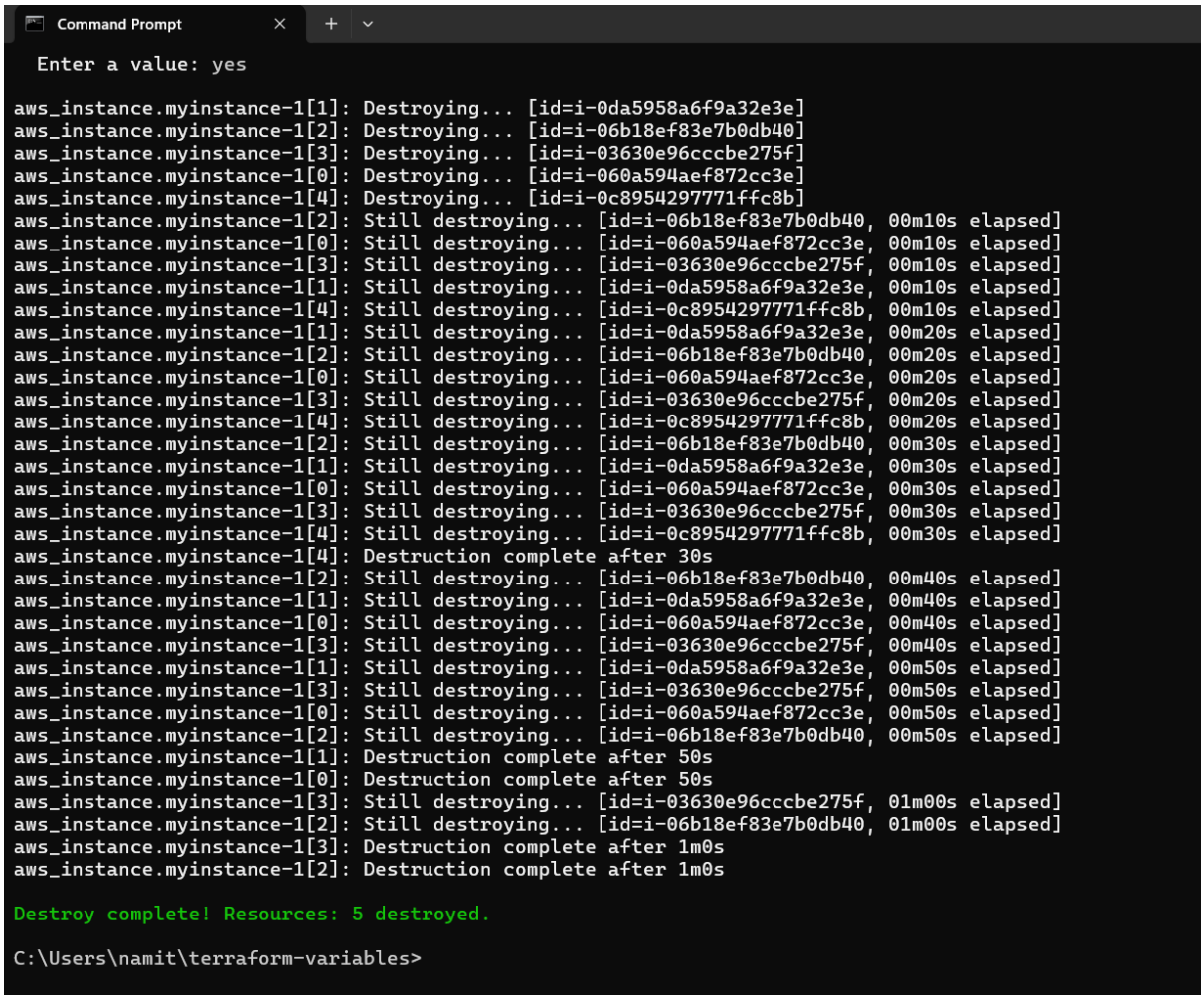

Command Prompt
C:\Users\namit\terraform-variables>terraform apply -auto-approve
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
Terraform will perform the following actions:
# aws_instance.myinstance-1[0] will be created
+ resource "aws_instance" "myinstance-1" {
+ ami = "ami-08718895af4dfa033"
+ arn = (known after apply)
+ associate_public_ip_address = (known after apply)
+ availability_zone = (known after apply)
+ disable_api_stop = (known after apply)
+ disable_api_termination = (known after apply)
+ ebs_optimized = (known after apply)
+ enable_primary_ipv6 = (known after apply)
+ force_destroy = false
+ get_password_data = false
+ host_id = (known after apply)
+ host_resource_group_arn = (known after apply)
+ iam_instance_profile = (known after apply)
+ id = (known after apply)
+ instance_initiated_shutdown_behavior = (known after apply)
+ instance_lifecycle = (known after apply)
+ instance_state = (known after apply)
+ instance_type = "t3.micro"
+ ipv6_address_count = (known after apply)
+ ipv6_addresses = (known after apply)
+ key_name = (known after apply)
+ monitoring = (known after apply)
+ outpost_arn = (known after apply)
+ password_data = (known after apply)
+ placement_group = (known after apply)
+ placement_group_id = (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)


Command Prompt
+ 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[4]: Creating...
aws_instance.myinstance-1[3]: Creating...
aws_instance.myinstance-1[0]: Creating...
aws_instance.myinstance-1[1]: Creating...
aws_instance.myinstance-1[2]: Creating...
aws_instance.myinstance-1[4]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[3]: 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[1]: Still creating... [00m10s elapsed]
aws_instance.myinstance-1[3]: Creation complete after 12s [id=i-03630e96cccbe275f]
aws_instance.myinstance-1[4]: Creation complete after 12s [id=i-0c8954297771ffc8b]
aws_instance.myinstance-1[0]: Creation complete after 12s [id=i-060a594aef872cc3e]
aws_instance.myinstance-1[2]: Creation complete after 12s [id=i-06b18ef83e7b0db40]
aws_instance.myinstance-1[1]: Creation complete after 12s [id=i-0da5958a6f9a32e3e]
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
C:\Users\namit\terraform-variables>
C:\Users\namit\terraform-variables>
C:\Users\namit\terraform-variables>

5. Clean Up:

After testing, you can clean up resources.

terraform destroy



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.