Lab Exercise 6– Terraform Variables

Objective:

Learn how to define and use variables in Terraform configuration.

Prerequisites:

- Install Terraform on your machine.

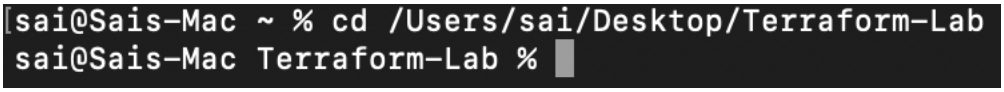
Steps:

1. Create a Terraform Directory:

- Create a new directory for your Terraform project.

```
mkdir Terraform-Lab

cd Terraform-Lab
```

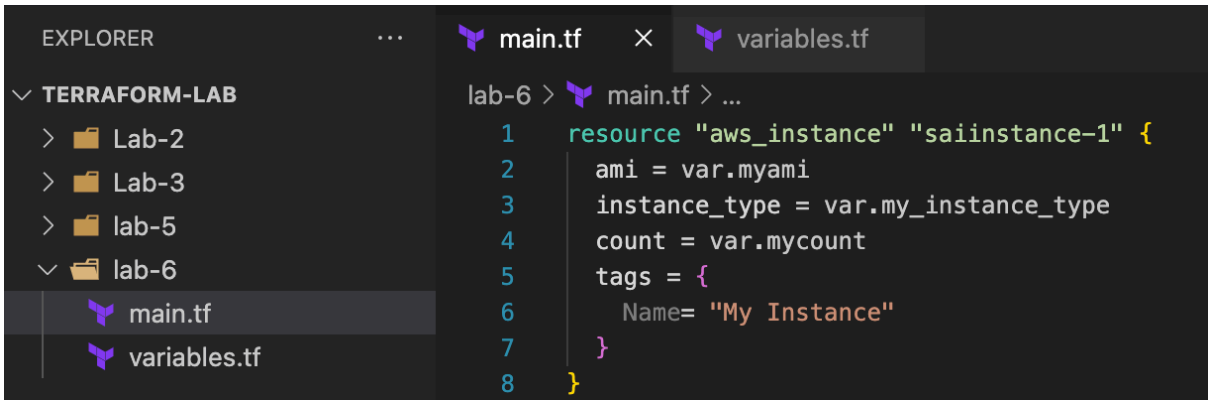


2. Create a Terraform Configuration File:

- 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.

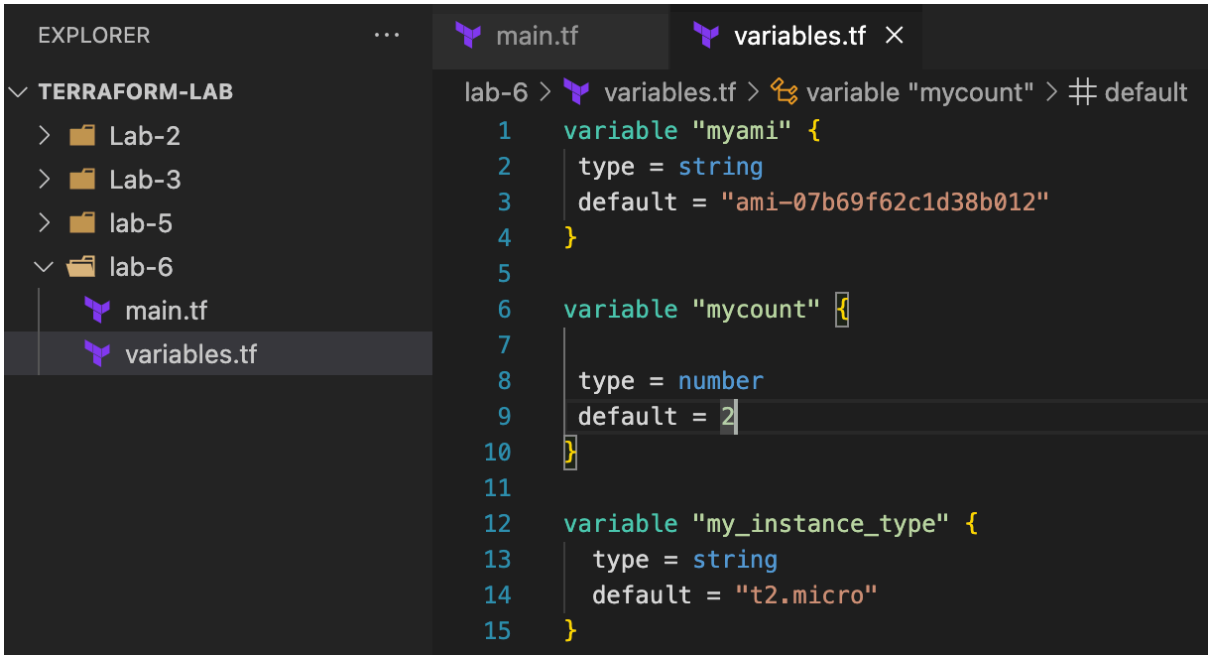
 # variables.tf

```
variable "myami" {
 type = string
 default = "ami-08718895af4dfa033"
}

variable "mycount" {

 type = number
 default = 2
}

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.

```
sai@Sais-Mac Lab_6 % terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.83.1...
- Installed hashicorp/aws v5.83.1 (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.
```

```
sai@Sais-Mac lab-6 % 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:
```

```
      + 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: 2 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.
```

```
sai@Sais-Mac lab-6 % 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:
```

```
Plan: 2 to add, 0 to change, 0 to destroy.
aws_instance.saiinstance-1[1]: Creating...
aws_instance.saiinstance-1[0]: Creating...
aws_instance.saiinstance-1[1]: Still creating... [10s elapsed]
aws_instance.saiinstance-1[0]: Still creating... [10s elapsed]
aws_instance.saiinstance-1[0]: Creation complete after 15s [id=i-0f3bdea25ca8f85d7]
aws_instance.saiinstance-1[1]: Creation complete after 15s [id=i-0664fce06bcea7b7b]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

5. Clean Up:

After testing, you can clean up resources.

terraform destroy

```
sai@Sais-Mac lab-6 % terraform destroy -auto-approve
aws_instance.saiinstance-1[1]: Refreshing state... [id=i-0664fce06bcea7b7b]
aws_instance.saiinstance-1[0]: Refreshing state... [id=i-0f3bdea25ca8f85d7]

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

Terraform will perform the following actions:
```

```
Plan: 0 to add, 0 to change, 2 to destroy.
aws_instance.saiinstance-1[1]: Destroying... [id=i-0664fce06bcea7b7b]
aws_instance.saiinstance-1[0]: Destroying... [id=i-0f3bdea25ca8f85d7]
aws_instance.saiinstance-1[0]: Still destroying... [id=i-0f3bdea25ca8f85d7, 10s elapsed]
aws_instance.saiinstance-1[1]: Still destroying... [id=i-0664fce06bcea7b7b, 10s elapsed]
aws_instance.saiinstance-1[1]: Still destroying... [id=i-0664fce06bcea7b7b, 20s elapsed]
aws_instance.saiinstance-1[0]: Still destroying... [id=i-0f3bdea25ca8f85d7, 20s elapsed]
aws_instance.saiinstance-1[1]: Still destroying... [id=i-0664fce06bcea7b7b, 30s elapsed]
aws_instance.saiinstance-1[0]: Still destroying... [id=i-0f3bdea25ca8f85d7, 30s elapsed]
aws_instance.saiinstance-1[0]: Still destroying... [id=i-0f3bdea25ca8f85d7, 40s elapsed]
aws_instance.saiinstance-1[1]: Still destroying... [id=i-0664fce06bcea7b7b, 40s elapsed]
aws_instance.saiinstance-1[0]: Destruction complete after 46s
aws_instance.saiinstance-1[1]: Still destroying... [id=i-0664fce06bcea7b7b, 50s elapsed]
aws_instance.saiinstance-1[1]: Still destroying... [id=i-0664fce06bcea7b7b, 1m0s elapsed]
aws_instance.saiinstance-1[1]: Destruction complete after 1m7s

Destroy complete! Resources: 2 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.