ANSHIKA SRIVASTAVA

ROLL NUMBER – R2142220907

SAP ID – 500107049

LAB EXERCISE 7

Lab Exercise 7– Terraform Variables with Command Line Arguments

Objective:

Learn how to pass values to Terraform variables using command line arguments.

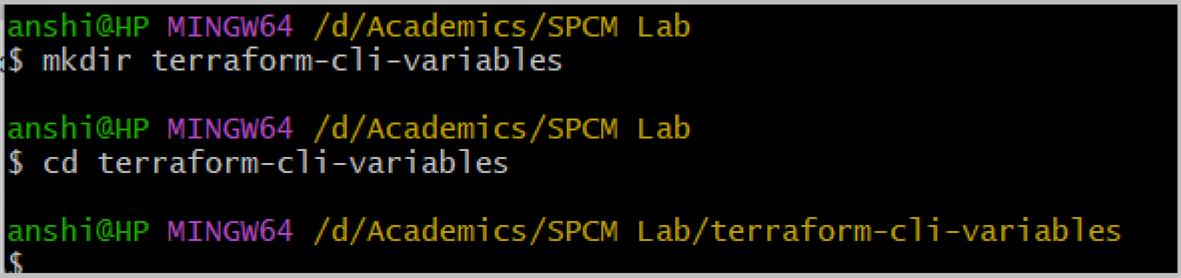
Prerequisites:

- Terraform installed on your machine.
- Basic knowledge of Terraform variables.

Steps:

1. Create a Terraform Directory:

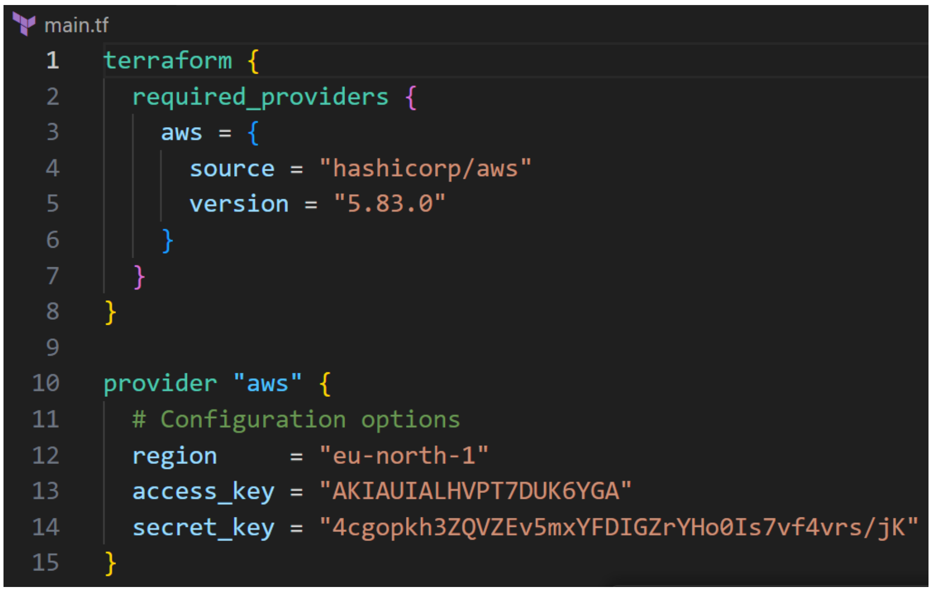
```
mkdir terraform-cli-variables

cd terraform-cli-variables
```



```
anshi@HP MINGW64 /d/Academics/SPCM Lab
$ mkdir terraform-cli-variables

anshi@HP MINGW64 /d/Academics/SPCM Lab
$ cd terraform-cli-variables

anshi@HP MINGW64 /d/Academics/SPCM Lab/terraform-cli-variables
$
```

2. Create Terraform Configuration Files:

- Create a file named main.tf:

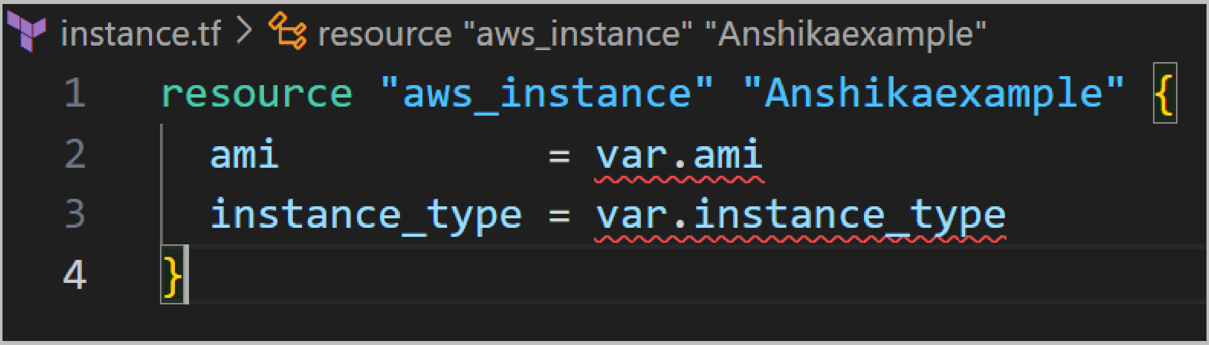


main.tf

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.83.0"
    }
  }
}

provider "aws" {
  # Configuration options
  region     = "eu-north-1"
  access_key = "AKIAUIALHVPT7DUK6YGA"
  secret_key = "4cgopkh3ZQVZEv5mxYFDIGZrYHo0Is7vf4vrs/jK"
}
```

# instance.tf

```
resource "aws_instance" "example" {
  ami           = var.ami
  instance_type = var.instance_type
}
```



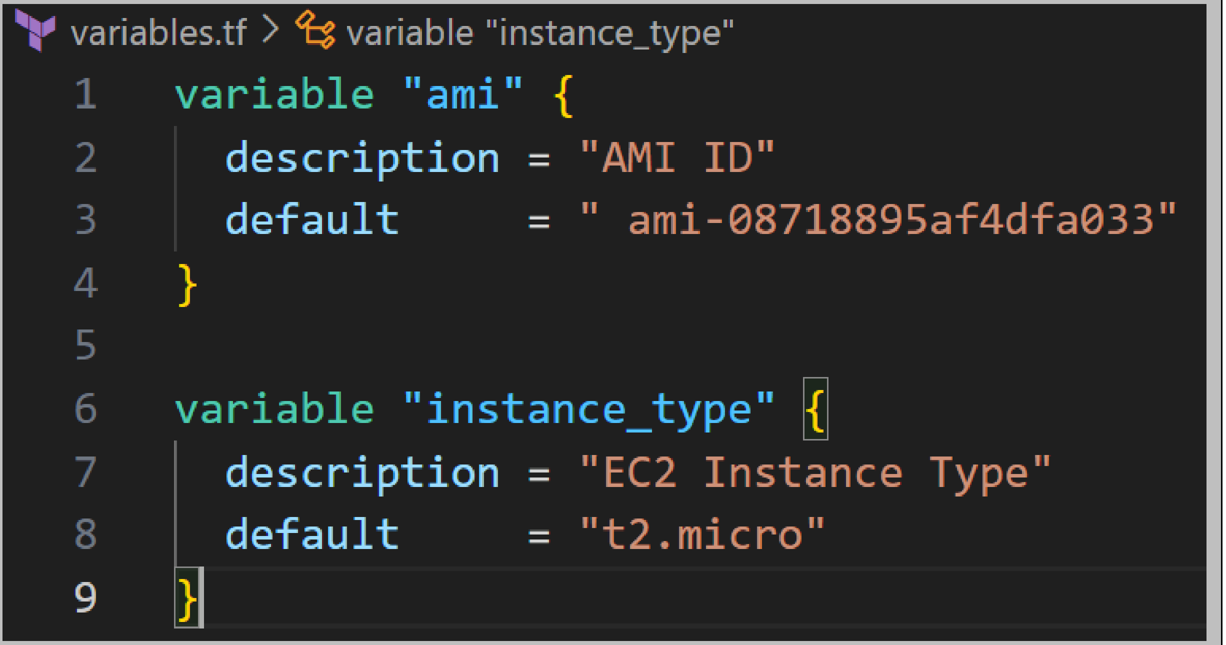
instance.tf > resource "aws_instance" "Anshikaexample"

```
resource "aws_instance" "Anshikaexample" {
  ami           = var.ami
  instance_type = var.instance_type
}
```

- Create a file named variables.tf:

# variables.tf

```
variable "ami" {
  description = "AMI ID"
  default     = " ami-08718895af4dfa033"
}

variable "instance_type" {
  description = "EC2 Instance Type"
  default     = "t2.micro"
}
```

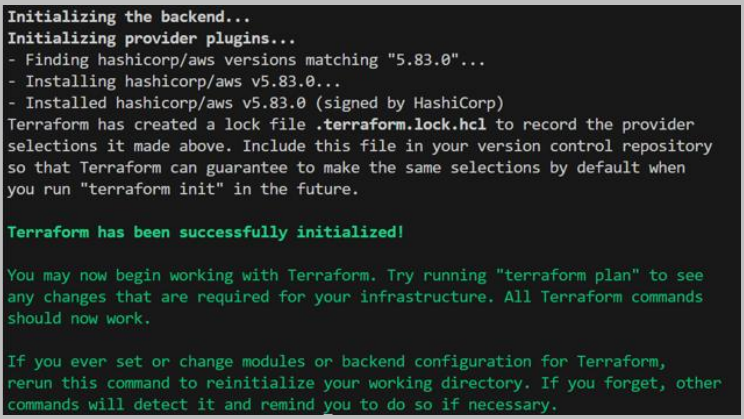


```
variables.tf > variable "instance_type"
variable "ami" {
  description = "AMI ID"
  default     = " ami-08718895af4dfa033"
}

variable "instance_type" {
  description = "EC2 Instance Type"
  default     = "t2.micro"
}
```

3. Use Command Line Arguments:

- Open a terminal and navigate to your Terraform project directory.
- Run the terraform init command:

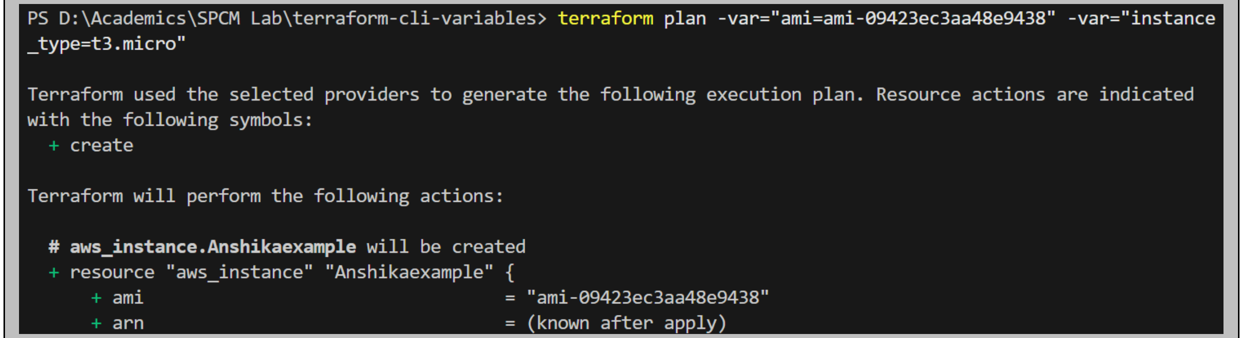
```
terraform init
```



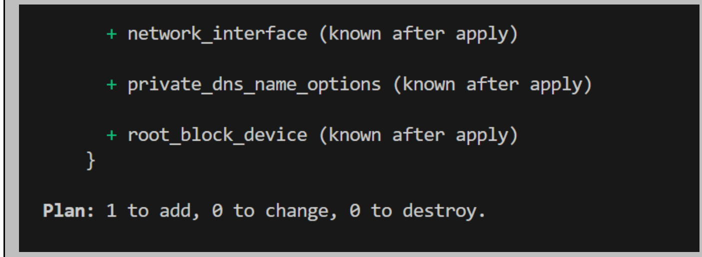
```
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.83.0"...
- Installing hashicorp/aws v5.83.0...
- Installed hashicorp/aws v5.83.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections 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.
```

- Run the terraform apply command with command line arguments to set variable values:

terraform plan -var="ami=ami-09423ec3aa48e9438" -var="instance_type=t3.micro"

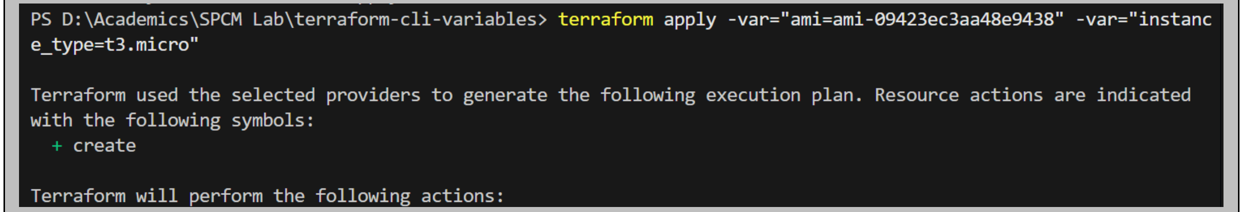


```
PS D:\Academics\SPCM Lab\terraform-cli-variables> terraform plan -var="ami=ami-09423ec3aa48e9438" -var="instance
_type=t3.micro"

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.Anshikaexample will be created
  + resource "aws_instance" "Anshikaexample" {
      + ami                                  = "ami-09423ec3aa48e9438"
      + arn                                  = (known after apply)
```

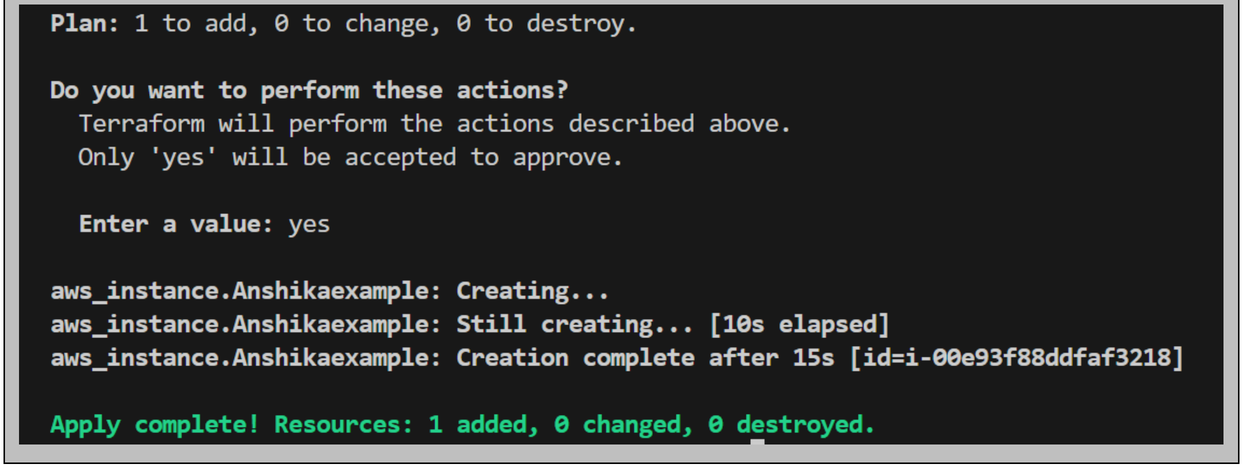


```
      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

  Plan: 1 to add, 0 to change, 0 to destroy.
```

terraform apply -var="ami=ami-09423ec3aa48e9438" -var="instance_type=t3.micro"



```
PS D:\Academics\SPCM Lab\terraform-cli-variables> terraform apply -var="ami=ami-09423ec3aa48e9438" -var="instanc
e_type=t3.micro"

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:
```

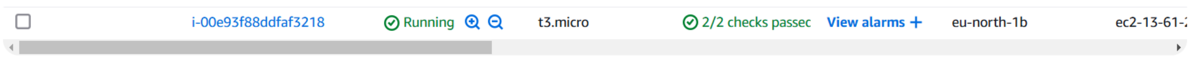


```
  Plan: 1 to add, 0 to change, 0 to destroy.

  Do you want to perform these actions?
    Terraform will perform the actions described above.
    Only 'yes' will be accepted to approve.

    Enter a value: yes

  aws_instance.Anshikaexample: Creating...
  aws_instance.Anshikaexample: Still creating... [10s elapsed]
  aws_instance.Anshikaexample: Creation complete after 15s [id=i-00e93f88ddfaf3218]

  Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

- Adjust the values based on your preferences.

4. Test and Verify:

- Observe how the command line arguments dynamically set the variable values during the apply process.
- Access the AWS Management Console or use the AWS CLI to verify the creation of resources in the specified region.

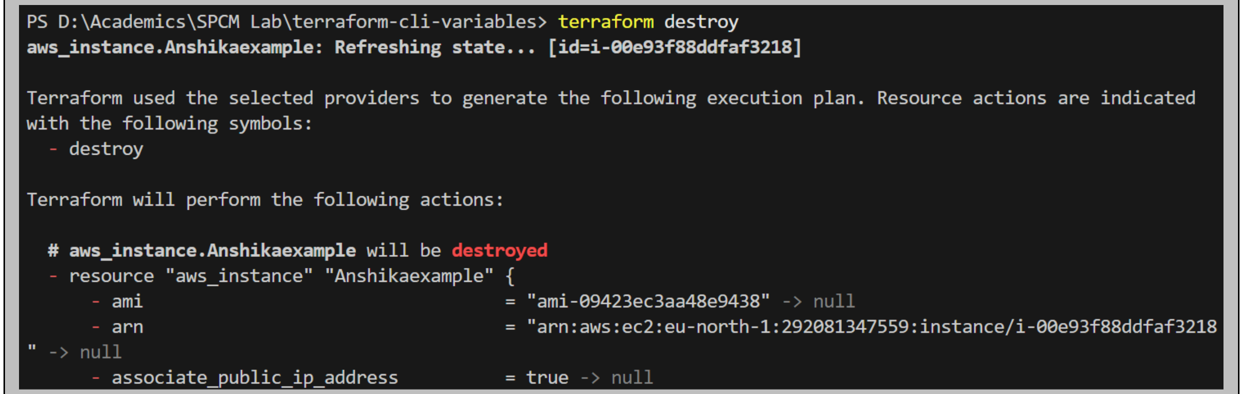


```
i-00e93f88ddfaf3218   Running   t3.micro   2/2 checks passec   View alarms +   eu-north-1b   ec2-13-61-
```

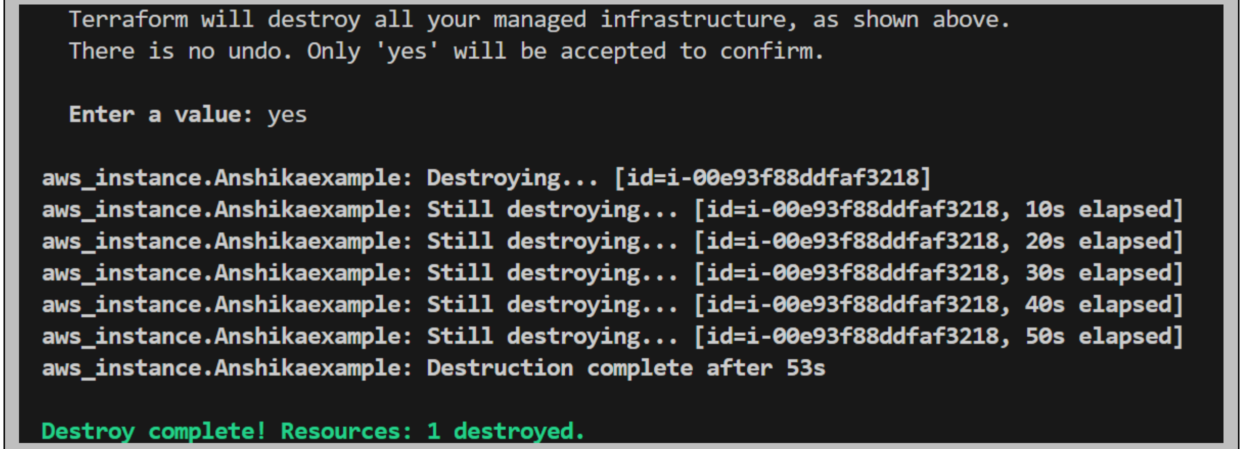
5. Clean Up:

After testing, you can clean up resources:

terraform destroy



```
PS D:\Academics\SPCM Lab\terraform-cli-variables> terraform destroy
aws_instance.Anshikaexample: Refreshing state... [id=i-00e93f88ddfaf3218]

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:

  # aws_instance.Anshikaexample will be destroyed
  - resource "aws_instance" "Anshikaexample" {
      - ami                                  = "ami-09423ec3aa48e9438" -> null
      - arn                                  = "arn:aws:ec2:eu-north-1:292081347559:instance/i-00e93f88ddfaf3218
" -> null
      - associate_public_ip_address          = true -> null
```



```
    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.Anshikaexample: Destroying... [id=i-00e93f88ddfaf3218]
  aws_instance.Anshikaexample: Still destroying... [id=i-00e93f88ddfaf3218, 10s elapsed]
  aws_instance.Anshikaexample: Still destroying... [id=i-00e93f88ddfaf3218, 20s elapsed]
  aws_instance.Anshikaexample: Still destroying... [id=i-00e93f88ddfaf3218, 30s elapsed]
  aws_instance.Anshikaexample: Still destroying... [id=i-00e93f88ddfaf3218, 40s elapsed]
  aws_instance.Anshikaexample: Still destroying... [id=i-00e93f88ddfaf3218, 50s elapsed]
  aws_instance.Anshikaexample: Destruction complete after 53s

  Destroy complete! Resources: 1 destroyed.
```

Confirm the destruction by typing yes.

6. Conclusion:

This lab exercise demonstrates how to use command line arguments to set variable values dynamically during the terraform apply process. It allows you to customize your Terraform deployments without modifying the configuration files directly. Experiment with different variable values and observe how command line arguments impact the infrastructure provisioning process.