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:

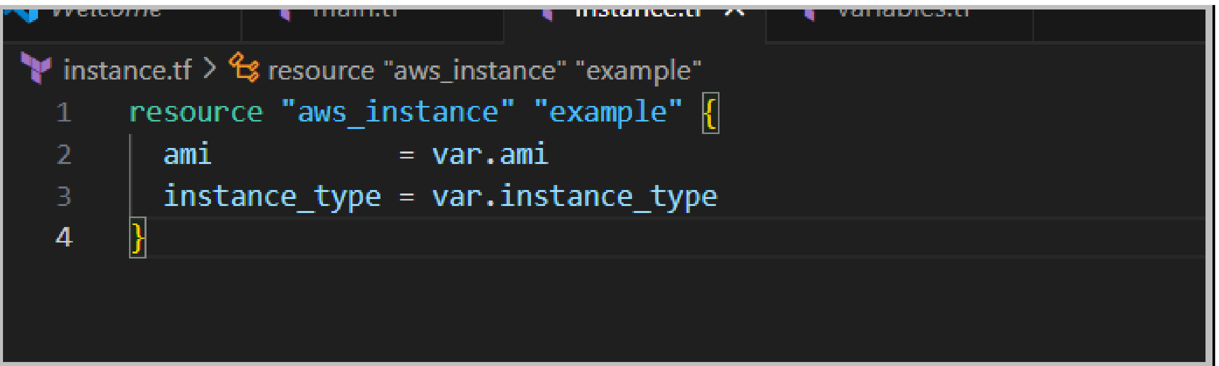
```
mkdir terraform-cli-variables

cd terraform-cli-variables
```

2. Create Terraform Configuration Files:

- **Create a file named main.tf:**

# instance.tf

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



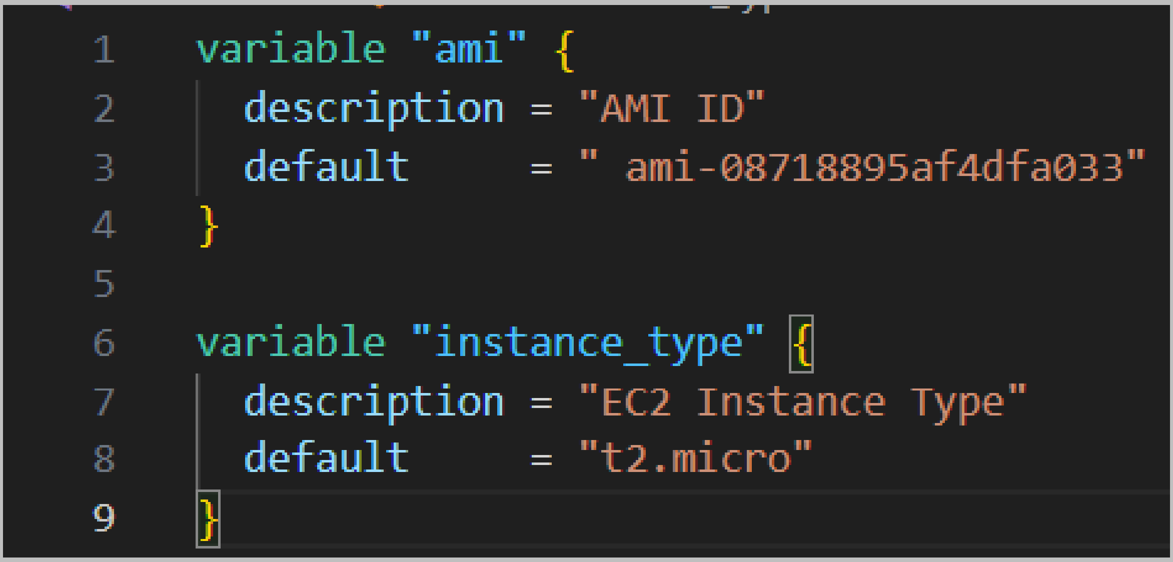
instance.tf > resource "aws_instance" "example"

```
resource "aws_instance" "example" {
  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"
}
```

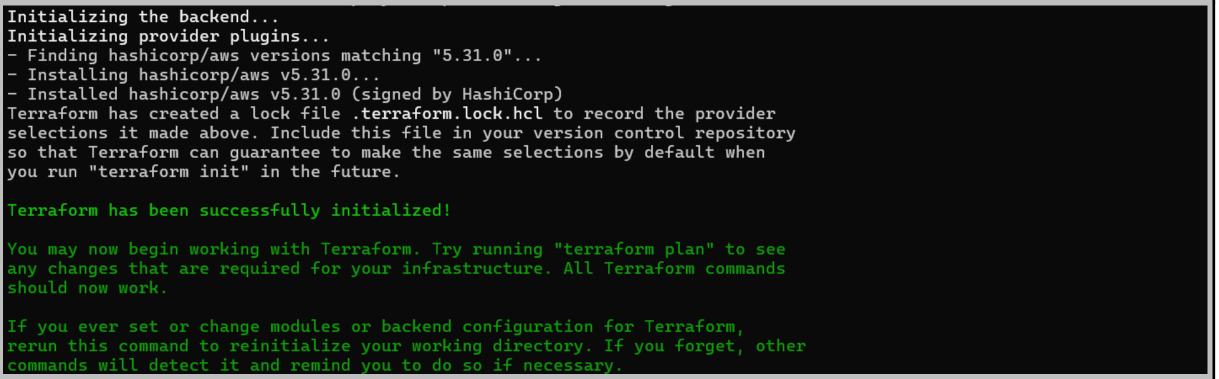


```
variable "ami" {
  description = "AMI ID"
  default     = " ami-08718895af4dfa033"
}

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:**

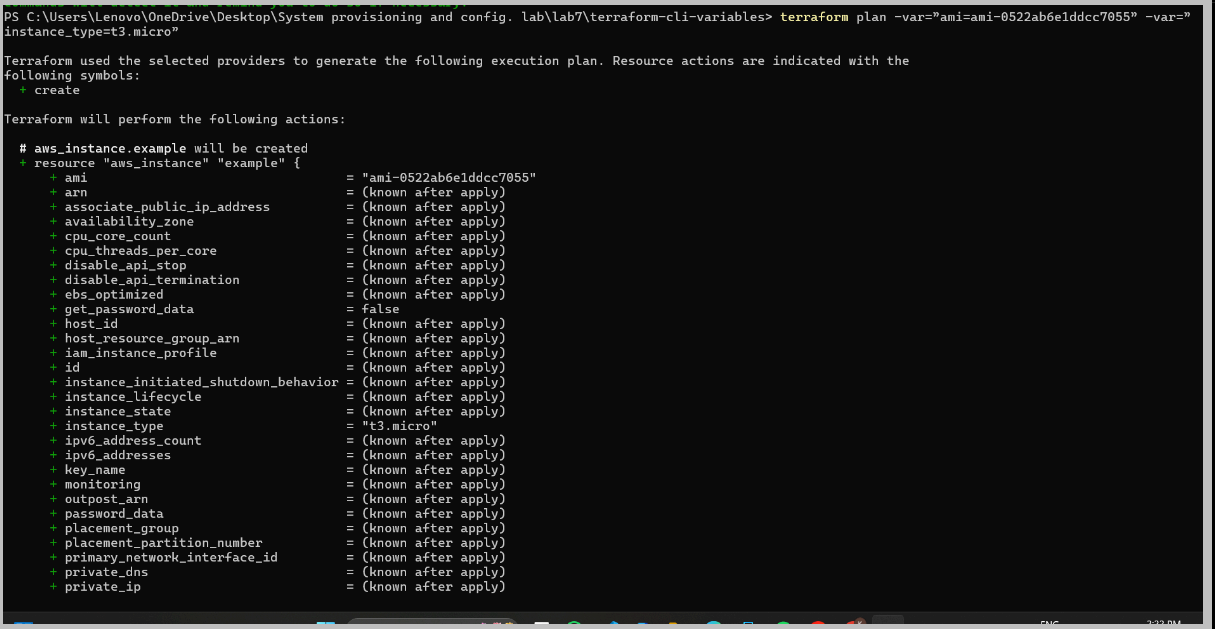
terraform init



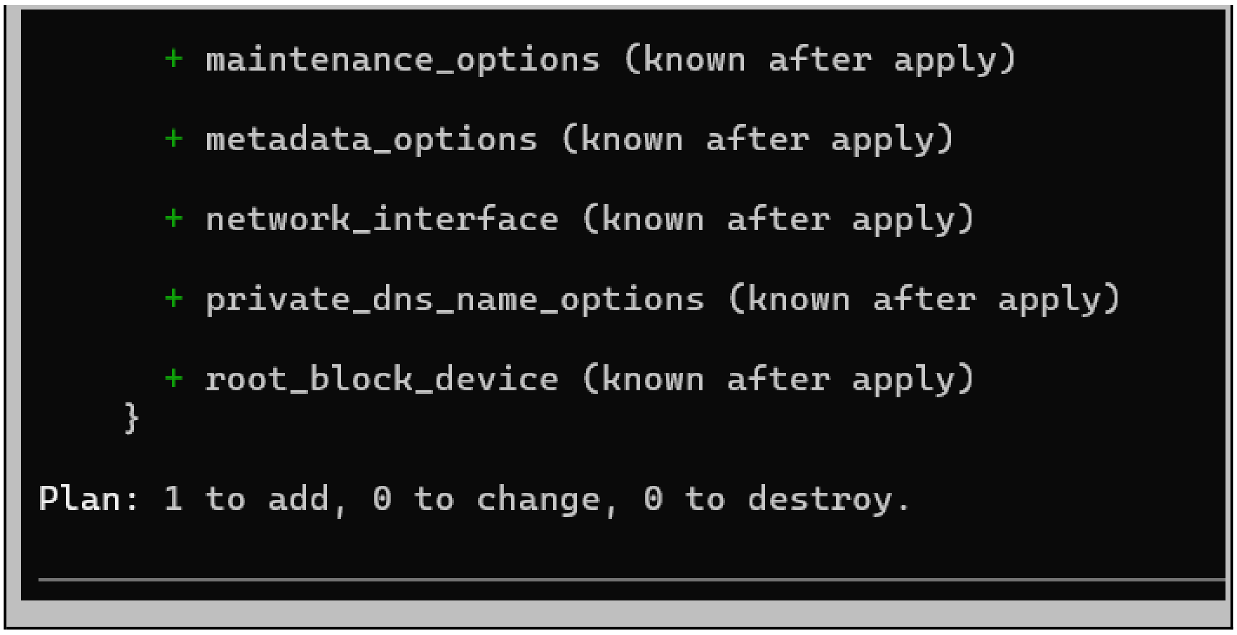
```
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.31.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections 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-0522ab6e1ddcc7055” -var=”instance_type=t3.micro”



```
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\lab7\terraform-cli-variables> terraform plan -var=”ami=ami-0522ab6e1ddcc7055” -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.example will be created
  + resource "aws_instance" "example" {
      + ami                                  = "ami-0522ab6e1ddcc7055"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + 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_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
```

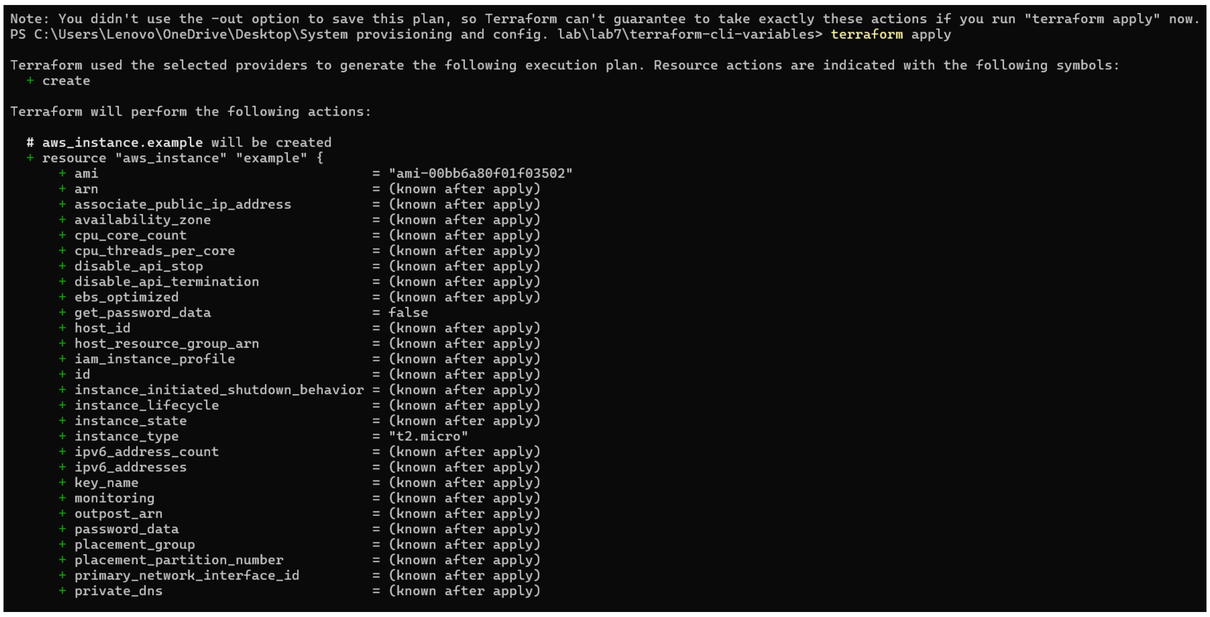


```
      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

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

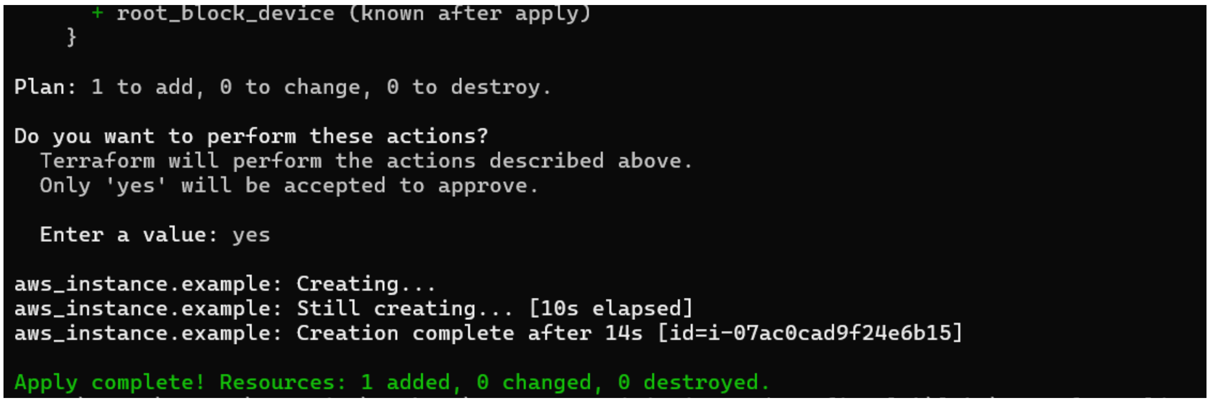
- **Adjust the values based on your preferences.**

4. Test and Verify:

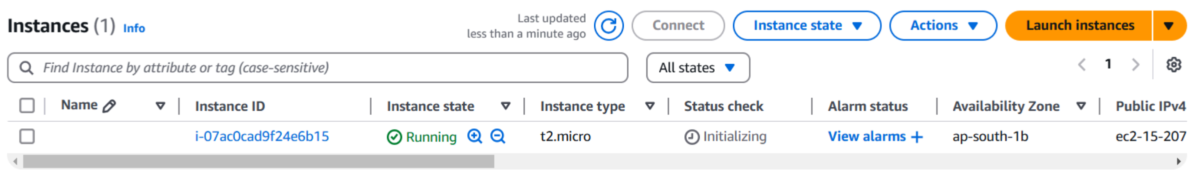
- **Observe how the command line arguments dynamically set the variable values during the apply process.**
- **Access the AWS Management Console or use the AWS CLI to verify the creation of resources in the specified region.**



```
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.
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\lab7\terraform-cli-variables> terraform apply

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.example will be created
  + resource "aws_instance" "example" {
      + ami                                  = "ami-00bb6a80f01f03502"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + 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_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
```



```
      + root_block_device (known after apply)
    }

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.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Creation complete after 14s [id=i-07ac0cad9f24e6b15]

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



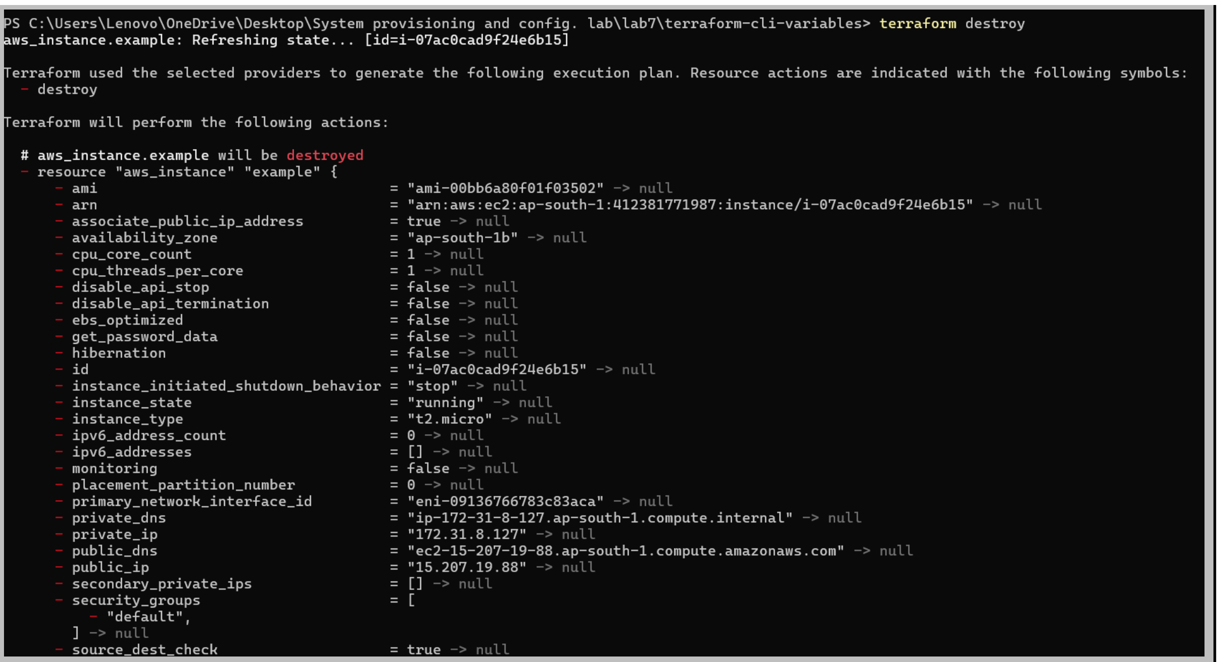
Instances (1) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
	i-07ac0cad9f24e6b15	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-15-207

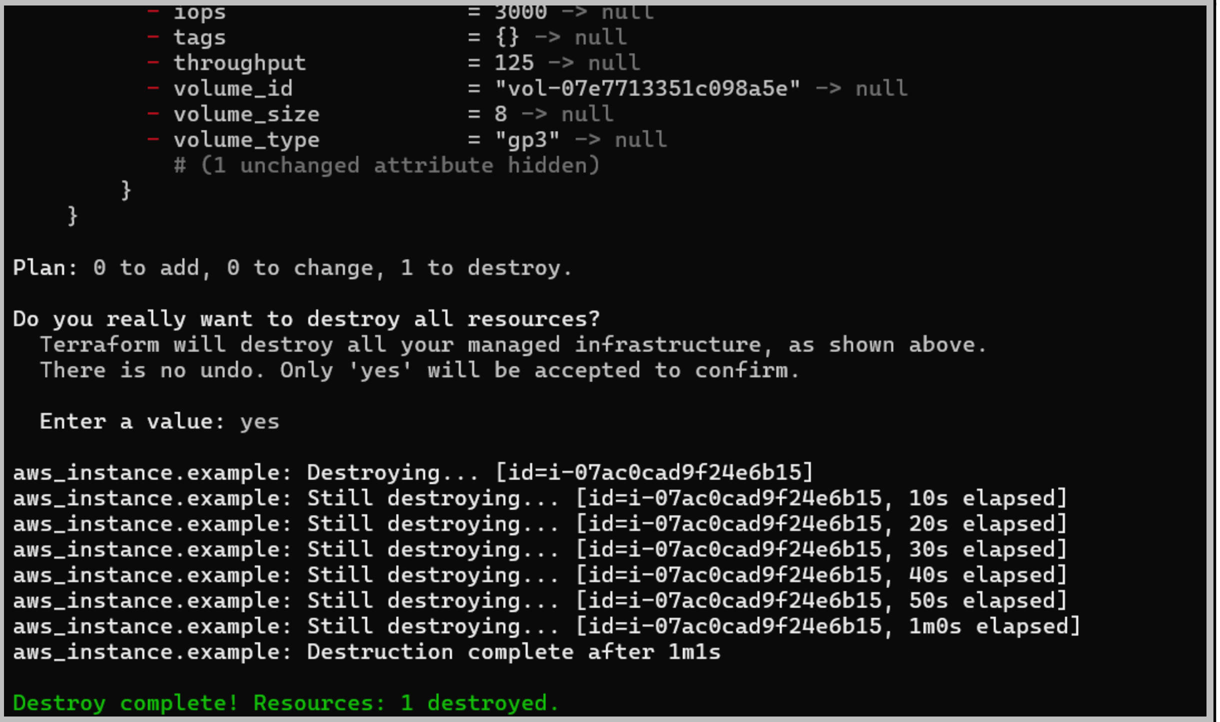
5. Clean Up:

After testing, you can clean up resources:

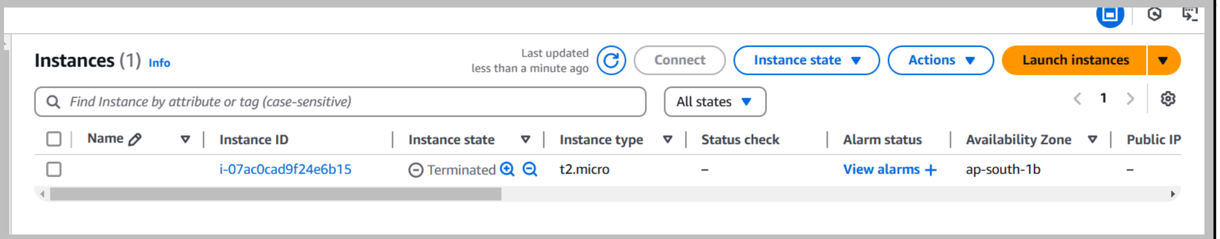
```
terraform destroy
```



```
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\lab7\terraform-cli-variables> terraform destroy
aws_instance.example: Refreshing state... [id=i-07ac0cad9f24e6b15]

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.example will be destroyed
  - resource "aws_instance" "example" {
      - ami                                  = "ami-00bb6a80f01f03502" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:412381771987:instance/i-07ac0cad9f24e6b15" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1b" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-07ac0cad9f24e6b15" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-09136766783c83aca" -> null
      - private_dns                          = "ip-172-31-8-127.ap-south-1.compute.internal" -> null
      - private_ip                           = "172.31.8.127" -> null
      - public_dns                           = "ec2-15-207-19-88.ap-south-1.compute.amazonaws.com" -> null
      - public_ip                            = "15.207.19.88" -> null
      - secondary_private_ips                = [] -> null
      - security_groups                      = [
          - "default",
        ] -> null
      - source_dest_check                    = true -> null
```



```
          - iops                  = 3000 -> null
          - tags                  = {} -> null
          - throughput            = 125 -> null
          - volume_id             = "vol-07e7713351c098a5e" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp3" -> null
            # (1 unchanged attribute hidden)
        }
    }

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

Do you really want to destroy all resources?
  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.example: Destroying... [id=i-07ac0cad9f24e6b15]
aws_instance.example: Still destroying... [id=i-07ac0cad9f24e6b15, 10s elapsed]
aws_instance.example: Still destroying... [id=i-07ac0cad9f24e6b15, 20s elapsed]
aws_instance.example: Still destroying... [id=i-07ac0cad9f24e6b15, 30s elapsed]
aws_instance.example: Still destroying... [id=i-07ac0cad9f24e6b15, 40s elapsed]
aws_instance.example: Still destroying... [id=i-07ac0cad9f24e6b15, 50s elapsed]
aws_instance.example: Still destroying... [id=i-07ac0cad9f24e6b15, 1m0s elapsed]
aws_instance.example: Destruction complete after 1m1s

Destroy complete! Resources: 1 destroyed.
```


Instances (1) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
☐		i-07ac0cad9f24e6b15	Terminated	t2.micro	–	View alarms +	ap-south-1b	–

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.