

Lab Exercise 16 – 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  
}
```

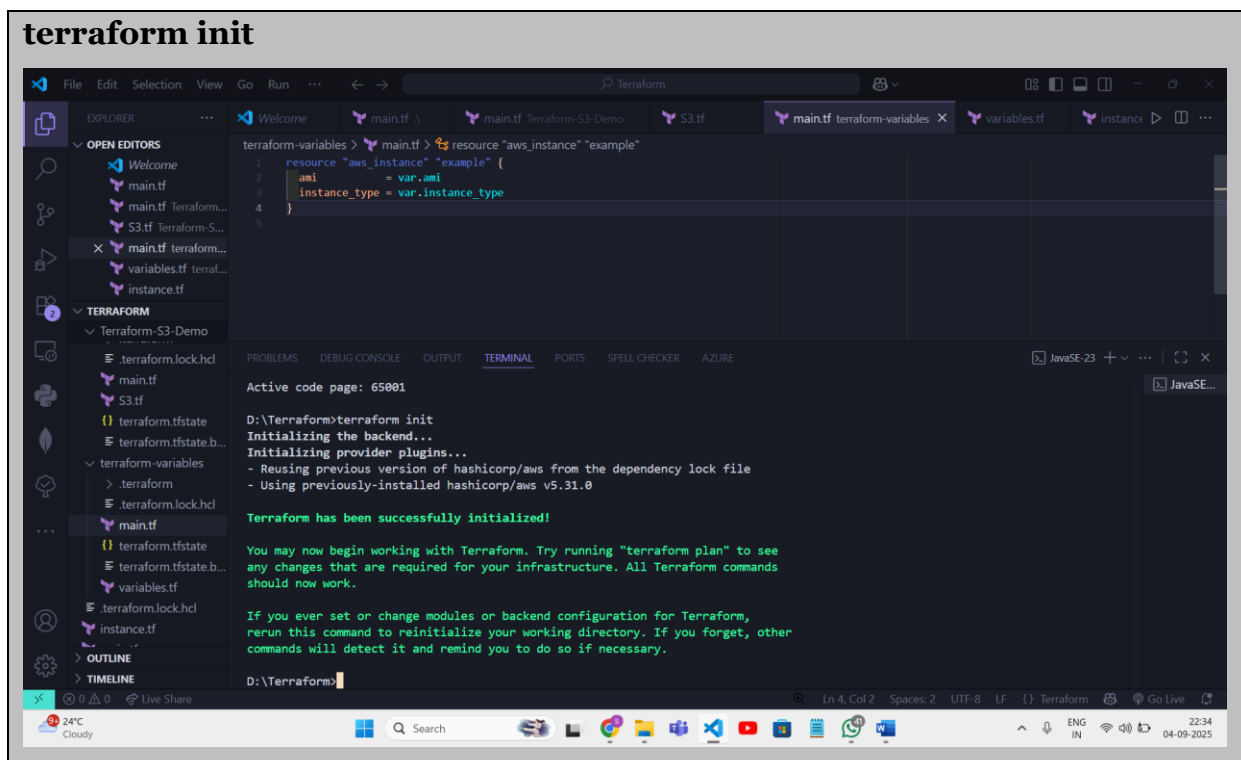
- Create a file named variables.tf:

variables.tf

```
variable "ami" {  
  description = "AMI ID"  
  default    = "ami-08718895af4dfa033"  
}  
  
variable "instance_type" {  
  description = "EC2 Instance Type"  
  default    = "t2.micro"  
}
```

3. Use Command Line Arguments:

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



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

terraform plan -var="ami=ami-0522ab6e1ddcc7055" -var="instance_type=t3.micro"

```
Terminal
+ 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: 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... [00m10s elapsed]
aws_instance.example: Creation complete after 18s [id=i-088ee9712bd04ce25]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
D:\Terraform\terraform-variables>

D:\Terraform\terraform-variables>terraform apply -var="ami=ami-036dc520857e3138f" -var="instance_type=t3.micro"
aws_instance.example: Refreshing state... [id=i-088ee9712bd04ce25]

No changes. Your infrastructure matches the configuration.

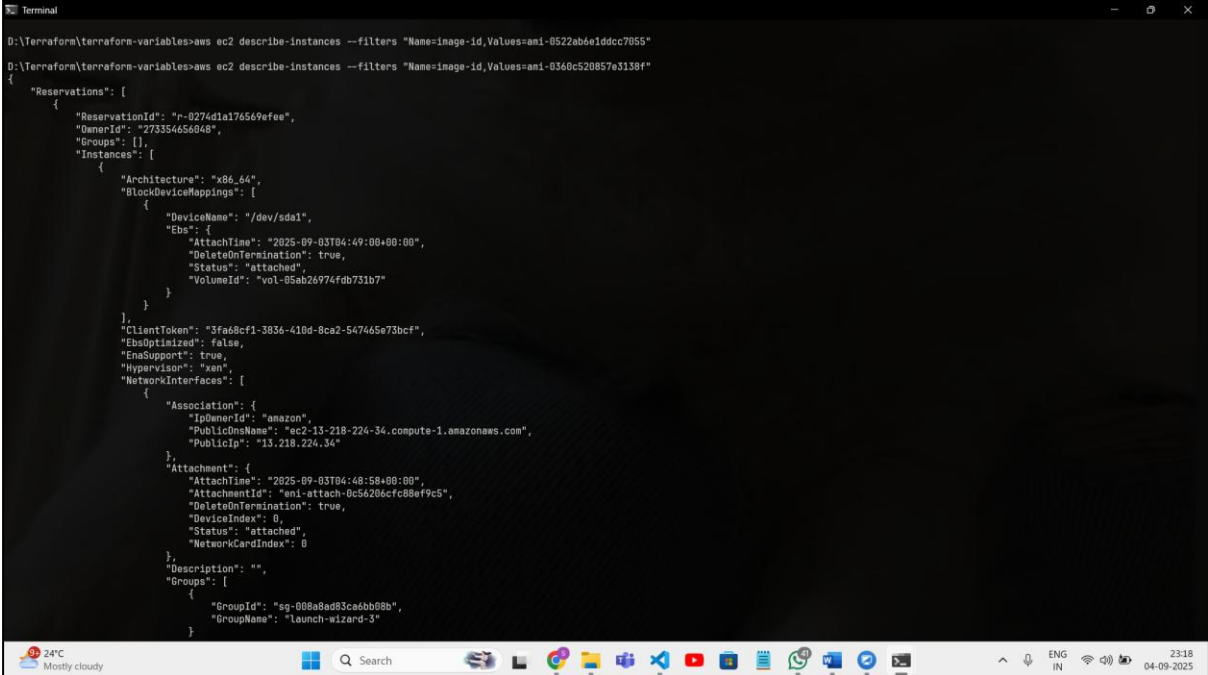
Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
D:\Terraform\terraform-variables>
```

- Adjust the values based on your preferences.

4. Test and Verify:

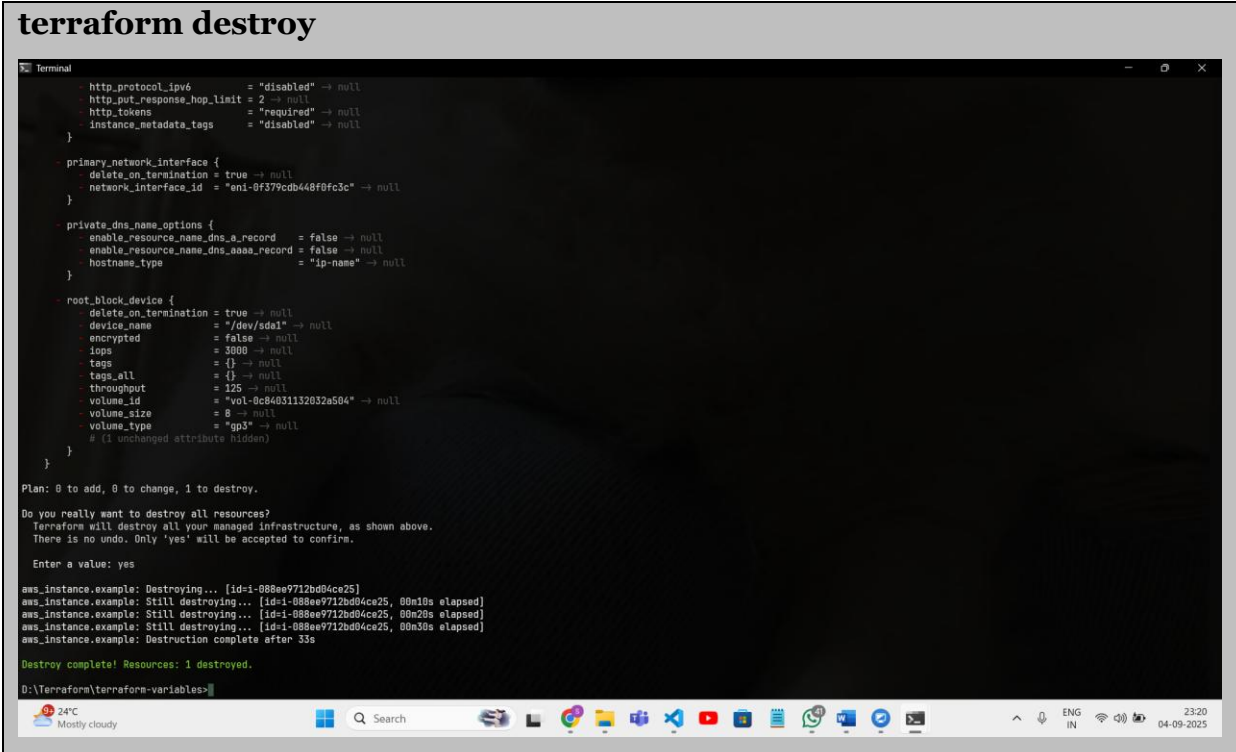
- Observe how the command line arguments dynamically set the variable values during the apply process.
- Access the AWS Management Console or use the AWS CLI to verify the creation of resources in the specified region.



```
D:\Terraform\terraform-variables>aws ec2 describe-instances --filters "Name=image-id,Values=ami-0522ab6e1ddcc7055"
D:\Terraform\terraform-variables>aws ec2 describe-instances --filters "Name=image-id,Values=ami-0360c520857e3138f"
{
  "Reservations": [
    {
      "ReservationId": "r-0274d1a176569efee",
      "OwnerId": "273356656048",
      "Groups": [],
      "Instances": [
        {
          "Architecture": "x86_64",
          "BlockDeviceMappings": [
            {
              "DeviceName": "/dev/sda1",
              "Ebs": {
                "AttachTime": "2025-09-03T04:49:00+00:00",
                "DeleteOnTermination": true,
                "Status": "attached",
                "VolumeId": "vol-05ab26974fdb731b7"
              }
            }
          ],
          "ClientToken": "3fa68cf1-3836-410d-8ca2-547465e73bcf",
          "EbsOptimized": false,
          "EnaSupport": true,
          "Hypervisor": "xen",
          "NetworkInterfaces": [
            {
              "Association": {
                "IpOwnerId": "amazon",
                "PublicDnsName": "ec2-13-218-224-34.compute-1.amazonaws.com",
                "PublicIp": "13.218.224.34"
              },
              "Attachment": {
                "AttachTime": "2025-09-03T04:48:58+00:00",
                "AttachmentId": "eni-attach-0c56206cfc88ef9c5",
                "DeleteOnTermination": true,
                "DeviceIndex": 0,
                "Status": "attached",
                "NetworkCardIndex": 0
              },
              "Description": "",
              "Groups": [
                {
                  "GroupId": "sg-008a8ad83ca6bb08b",
                  "GroupName": "launch-wizard-3"
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

5.Clean Up:

After testing, you can clean up resources:



```
terraform destroy

Terminal
http_protocol_ipv6      = "disabled" → null
http_put_response_hop_limit = 2 → null
http_tokens              = "required" → null
instance_metadata_tags   = "disabled" → null
}

- primary_network_interface {
  delete_on_termination = true → null
  network_interface_id   = "eni-0f379c0b448f0fc3c" → null
}

- private_dns_name_options {
  enable_resource_name_dns_a_record = false → null
  enable_resource_name_dns_aaaa_record = false → null
  hostname_type                     = "ip-name" → null
}

- root_block_device {
  delete_on_termination = true → null
  device_name           = "/dev/sda1" → null
  encrypted             = false → null
  iops                  = 3000 → null
  tags                  = {} → null
  tags_all              = {} → null
  throughput            = 125 → null
  volume_id             = "vol-0c84031132032a504" → 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-088ee9712bd04ce25]
aws_instance.example: Still destroying... [id=i-088ee9712bd04ce25, 00m10s elapsed]
aws_instance.example: Still destroying... [id=i-088ee9712bd04ce25, 00m20s elapsed]
aws_instance.example: Still destroying... [id=i-088ee9712bd04ce25, 00m30s elapsed]
aws_instance.example: Destruction complete after 35s

Destroy complete! Resources: 1 destroyed.

D:\Terraform\terraform-variables>
```

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.