

Lab Exercise 7– Terraform Variables with Command Line Arguments

Objective:

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

Prerequisites:

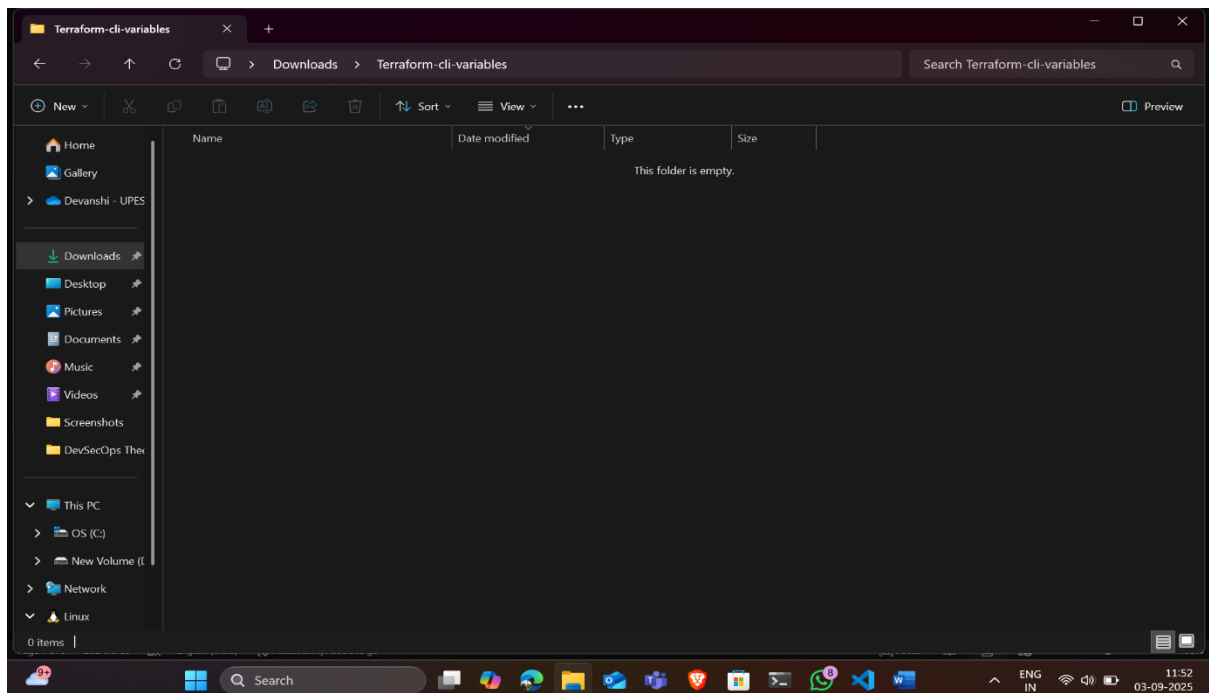
- Terraform installed on your machine.
- Basic knowledge of Terraform variables.

Steps:

1. Create a Terraform Directory:

```
mkdir terraform-cli-variables
```

```
cd terraform-cli-variables
```

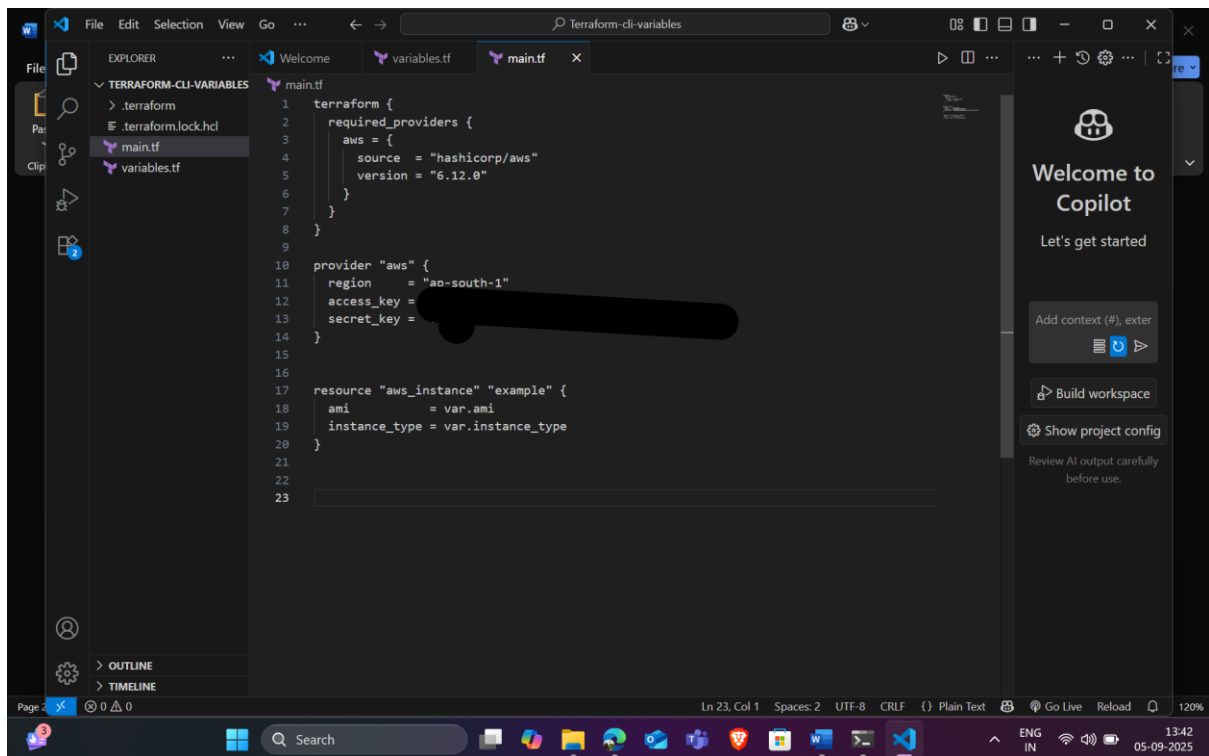


2. Create Terraform Configuration Files:

- Create a file named main.tf:

main.tf

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



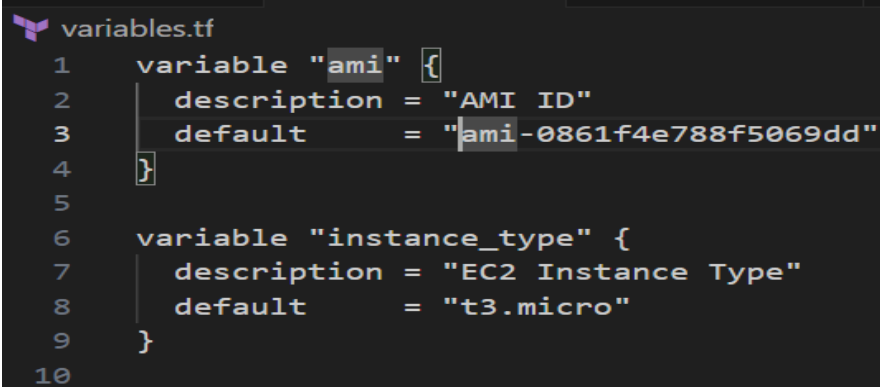
- Create a file named variables.tf:

variables.tf

```
variable "ami" {  
  description = "AMI ID"
```

```
default = "ami-08718895af4dfa033"
}
```

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

A screenshot of a code editor with a dark background. The file name 'variables.tf' is visible in the top left. The code is written in a light-colored font and includes line numbers from 1 to 10 on the left margin. The code defines two variables: 'ami' and 'instance_type'.

```
1  variable "ami" {
2    description = "AMI ID"
3    default     = "ami-0861f4e788f5069dd"
4  }
5
6  variable "instance_type" {
7    description = "EC2 Instance Type"
8    default     = "t3.micro"
9  }
10
```

3. Use Command Line Arguments:

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

```
terraform init
```

```
Terraform-cli-variables
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Devanshi\Downloads\Terraform-cli-variables> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.11.0...
- Installed hashicorp/aws v6.11.0 (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.
PS C:\Users\Devanshi\Downloads\Terraform-cli-variables>
```

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

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

- Adjust the values based on your preferences.

```
P5 D:\Downloads\terraform-cli-variables> terraform plan -var="ami=ami-0861fa78df5d6c9bd" -var="instance_type=t3.micro"

Terraform uses the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
+ create
~ update
= no action
$ replace
- destroy

Terraform will perform the following actions:

# aws_instance.example will be created
resource "aws_instance" "example" {
  ami              = "ami-0861fa78df5d6c9bd"
  arn              = (known after apply)
  associate_public_ip_address = (known after apply)
  availability_zone = (known after apply)
  disable_api_termination = (known after apply)
  ebs_optimized      = (known after apply)
  enable_primary_ipv6 = (known after apply)
  force_delete       = false
  get_console_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_addresses     = (known after apply)
  ipv6_address_count = (known after apply)
  key_name           = (known after apply)
  monitoring         = (known after apply)
  outpost_arn        = (known after apply)
  placement_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)
  public_ip          = (known after apply)
  region             = "ap-south-1"
  secondary_private_ips = (known after apply)
  security_groups     = (known after apply)
  source_dest_check   = true
  spot_instance_request_id = (known after apply)
  subnet_id          = (known after apply)
  tags_all           = (known after apply)
  tenancy            = (known after apply)
  user_data_b64      = (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_storage_device (known after apply)

+ instance_market_options (known after apply)

+ maintenance_options (known after apply)

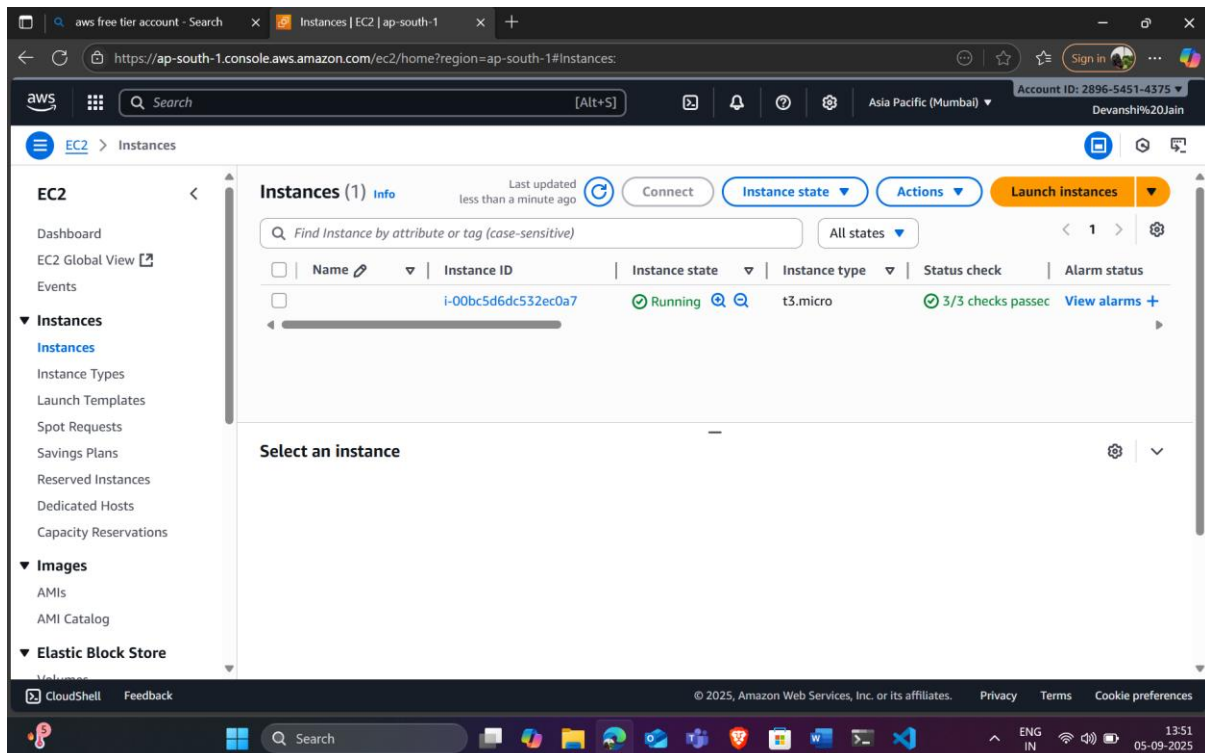
+ metadata_options (known after apply)

+ network_interface (known after apply)

+ primary_network_interface (known after apply)
```

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.

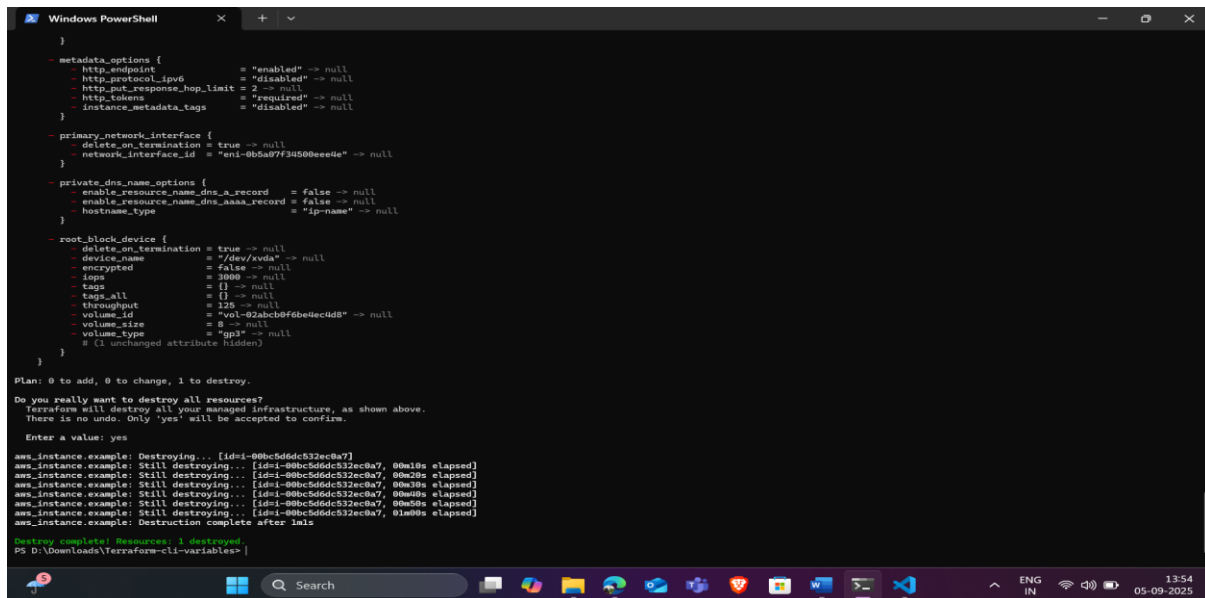


5. Clean Up:

After testing, you can clean up resources:

```
terraform destroy
```

Confirm the destruction by typing yes.



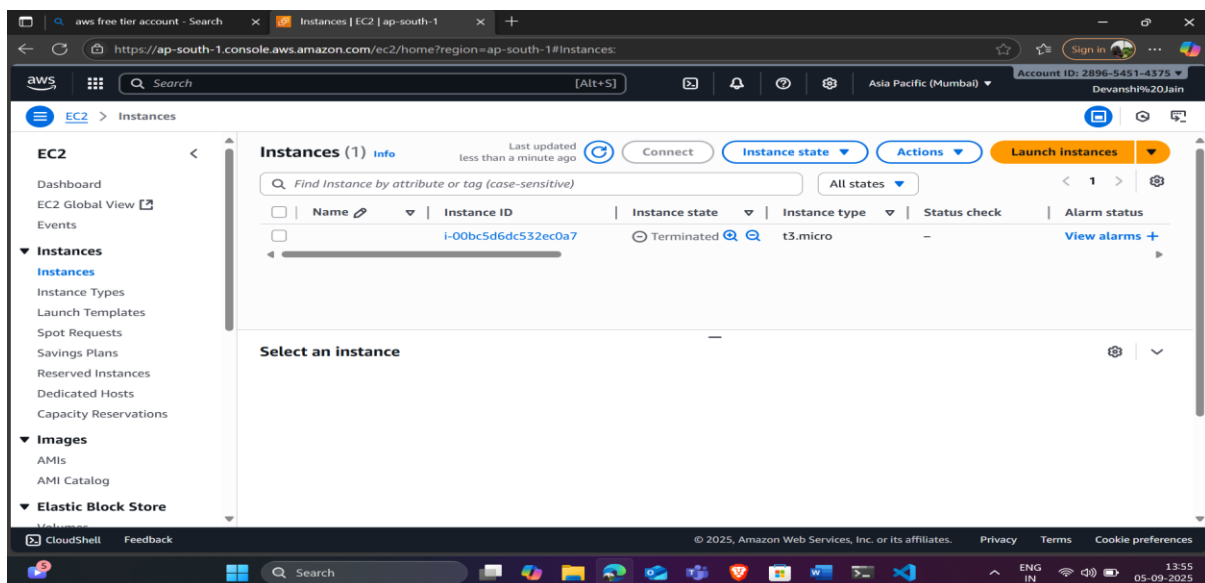
```

}
- metadata_options {
  http_endpoint = "enabled" -> null
  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-0b5a97f3450e0e04e" -> 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/svda" -> null
  encrypted = false -> null
  iops = 3000 -> null
  tags = {} -> null
  tags_all = {} -> null
  throughput = 125 -> null
  volume_id = "vol-02abcb0f6bedecdd8" -> 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-00bc5d6dc532ec0a7]
aws_instance.example: Still destroying... [id=i-00bc5d6dc532ec0a7, 00m10s elapsed]
aws_instance.example: Still destroying... [id=i-00bc5d6dc532ec0a7, 00m20s elapsed]
aws_instance.example: Still destroying... [id=i-00bc5d6dc532ec0a7, 00m30s elapsed]
aws_instance.example: Still destroying... [id=i-00bc5d6dc532ec0a7, 00m40s elapsed]
aws_instance.example: Still destroying... [id=i-00bc5d6dc532ec0a7, 00m50s elapsed]
aws_instance.example: Still destroying... [id=i-00bc5d6dc532ec0a7, 01m00s elapsed]
aws_instance.example: Destruction complete after 1m1s
Destroy complete! Resources: 1 destroyed
PS D:\Downloads\Terraform-cli-variables>

```



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.