

Lab Exercise 5– Terraform Variables with Command Line Arguments Objective:

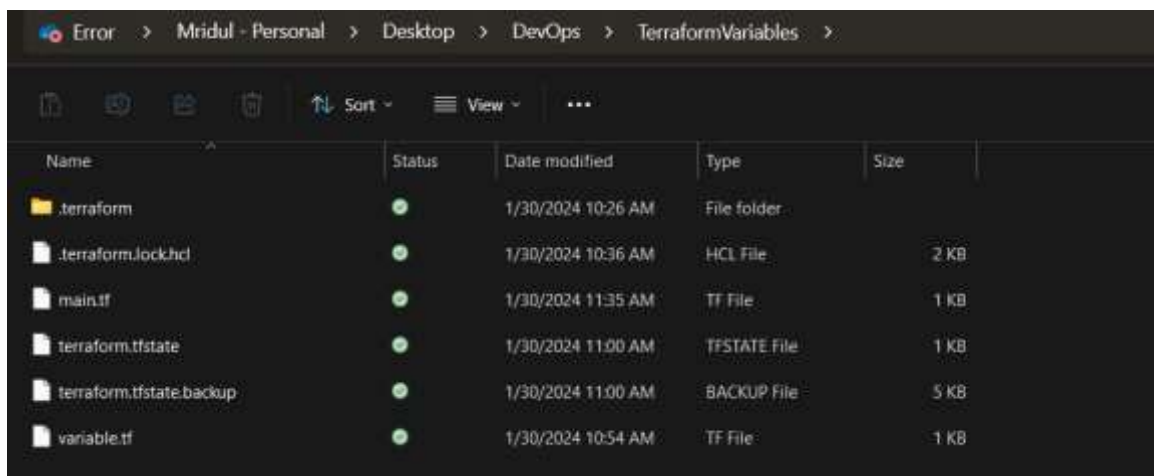
Learn how to pass values to Terraform variables using command line arguments.

Prerequisites:

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

Steps:

1. Create a Terraform Directory:



Name	Status	Date modified	Type	Size
.terraform	✓	1/30/2024 10:26 AM	File folder	
.terraform.lock.hcl	✓	1/30/2024 10:36 AM	HCL File	2 KB
main.tf	✓	1/30/2024 11:35 AM	TF File	1 KB
terraform.tfstate	✓	1/30/2024 11:00 AM	TFSTATE File	1 KB
terraform.tfstate.backup	✓	1/30/2024 11:00 AM	BACKUP File	5 KB
variable.tf	✓	1/30/2024 10:54 AM	TF File	1 KB

2. Create Terraform Configuration Files:

- Create a file named main.tf:

main.tf Create a file named variables.tf:

variables.tf

```
variable.tf X
variable.tf > variable "region_ec2" > description
1  variable "ami" {
2      description = "AMI ID"
3      default = "ami-03f4878755434977f"
4  }
5
6  variable "instance_ty" {
7      description = "ec2-instance"
8      default = "t2.micro"
9  }
10
11 variable "region_ec2" {
12     description = "ec2-region"
13     default = "ap-south-1"
14 }
```

3. Use Command Line Arguments:

```
PS C:\Users\Dell\OneDrive\Desktop\DevOps\Terraform\Variables> terraform apply -var "instance_ty=t2.small" -var "region_ec2=ap-south-1"

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-03f4878755434977f"
  + 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.small"
  + 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)
}
```

```

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: Still creating... [20s elapsed]
aws_instance.example: Creation complete after 24s [id=i-0815a18f08a60d7e8]

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

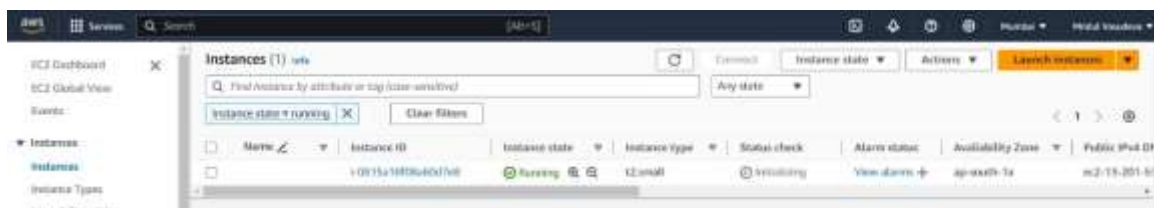
```

4. Test and Verify:

```

+ instance_state = "running" (known after apply)
+ instance_type  = "t2.small" (known after apply)
+ ipv6_address_count = 0 (known after apply)

```



5. Clean Up:

```

PS C:\Users\Belli\OneDrive\Desktop\DevOps\TerraformVariables> terraform destroy
aws_instance.example: Refreshing state... [id=i-0815a18f08a60d7e8]

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-83f487875434977f" -> null
  arn                    = "arn:aws:ec2:ap-south-1:637423583821:instance/i-0815a18f08a60d7e8" -> null
  associate_public_ip_address = true -> null
  availability_zone       = "ap-south-1a" -> 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-0815a18f08a60d7e8" -> null
  instance_initiated_shutdown_behavior = "stop" -> null
  instance_state          = "running" -> null
  instance_type           = "t2.small" -> null
  ipv6_address_count       = 0 -> null
  ipv6_addresses           = [] -> null
  monitoring               = false -> null
  placement_partition_number = 0 -> null
}

```

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-0815a18f08a60d7e8]
aws_instance.example: Still destroying... [id=i-0815a18f08a60d7e8, 10s elapsed]
aws_instance.example: Still destroying... [id=i-0815a18f08a60d7e8, 20s elapsed]
aws_instance.example: Still destroying... [id=i-0815a18f08a60d7e8, 30s elapsed]
aws_instance.example: Still destroying... [id=i-0815a18f08a60d7e8, 40s elapsed]
aws_instance.example: Destruction complete after 40s
```

Destroy complete! Resources: 1 destroyed.

