Lab Exercise 8– Terraform Multiple tfvars Files

Name- Manish kumar

SAP ID- 500119723

Batch-2 (DevOps)

Objective:

Learn how to use multiple tfvars files in Terraform for different environments.

Steps:

1. Create a Terraform Directory:

mkdir terraform-multiple-tfvars

cd terraform-multiple-tfvars

```
PS C:\Program Files\WindowsPowerShell> cd "C:\terraform"
PS C:\terraform> mkdir terraform-multiple-tfvars


    Directory: C:\terraform


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        18-09-2025     19:07                terraform-multiple-tfvars


PS C:\terraform> cd terraform-multiple-tfvars
PS C:\terraform\terraform-multiple-tfvars>
```

- Create Terraform Configuration Files:
- Create a file named main.tf:

 # main.tf

```
provider "aws" {
  region = var.region
}

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

- Create a file named variables.tf:

 # variables.tf

```
variable "ami" {
   type = string
}

variable "instance_ty" {
   type = string
}
```

2. Create Multiple tfvars Files:

- Create a file named dev.tfvars:

 # dev.tfvars

```
ami           = "ami-0123456789abcdef0"
instance_type = "t2.micro"
```

- Create a file named prod.tfvars:

 # prod.tfvars

```
ami           = "ami-9876543210fedcba0"
instance_type = "t2.large"
```

- In these files, provide values for the variables based on the environments.

3. Initialize and Apply for Dev Environment:

- Run the following Terraform commands to initialize and apply the configuration for the dev environment:

terraform init

terraform apply -var-file=dev.tfvars

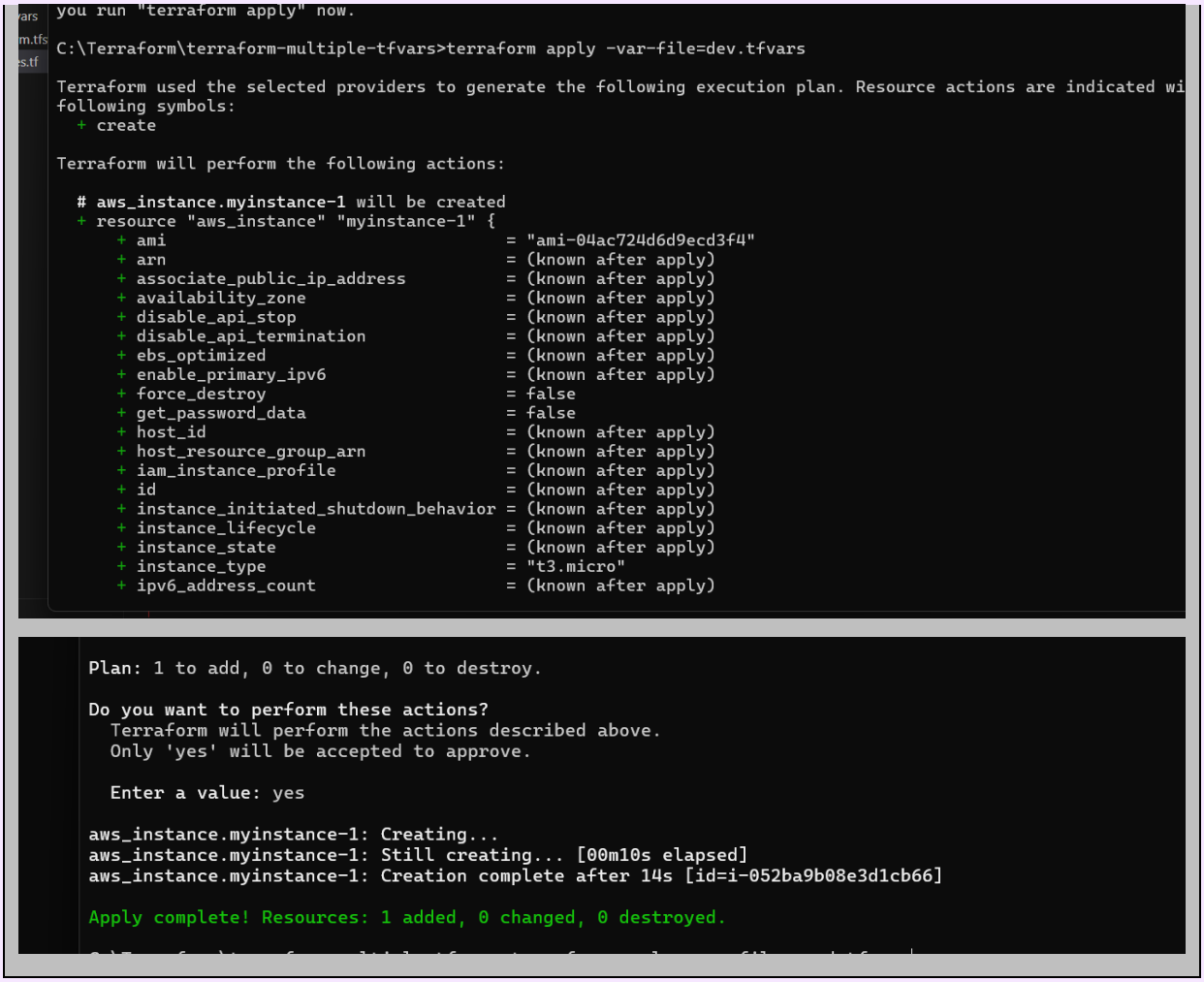
```
C:\Terraform\terraform-multiple-tfvars>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.13.0...
- Installed hashicorp/aws v6.13.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl 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.

C:\Terraform\terraform-multiple-tfvars>
```

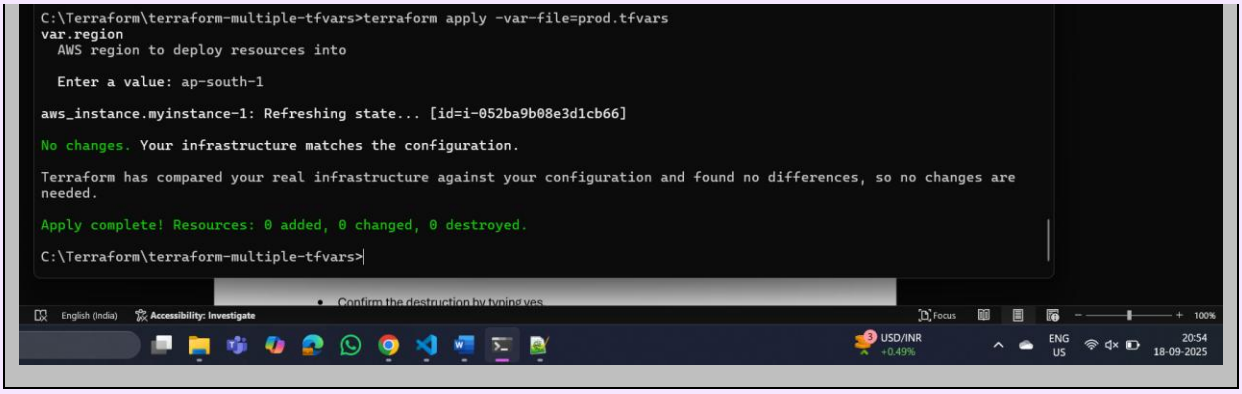


4. Initialize and Apply for Prod Environment:

- Run the following Terraform commands to initialize and apply the configuration for the prod environment:

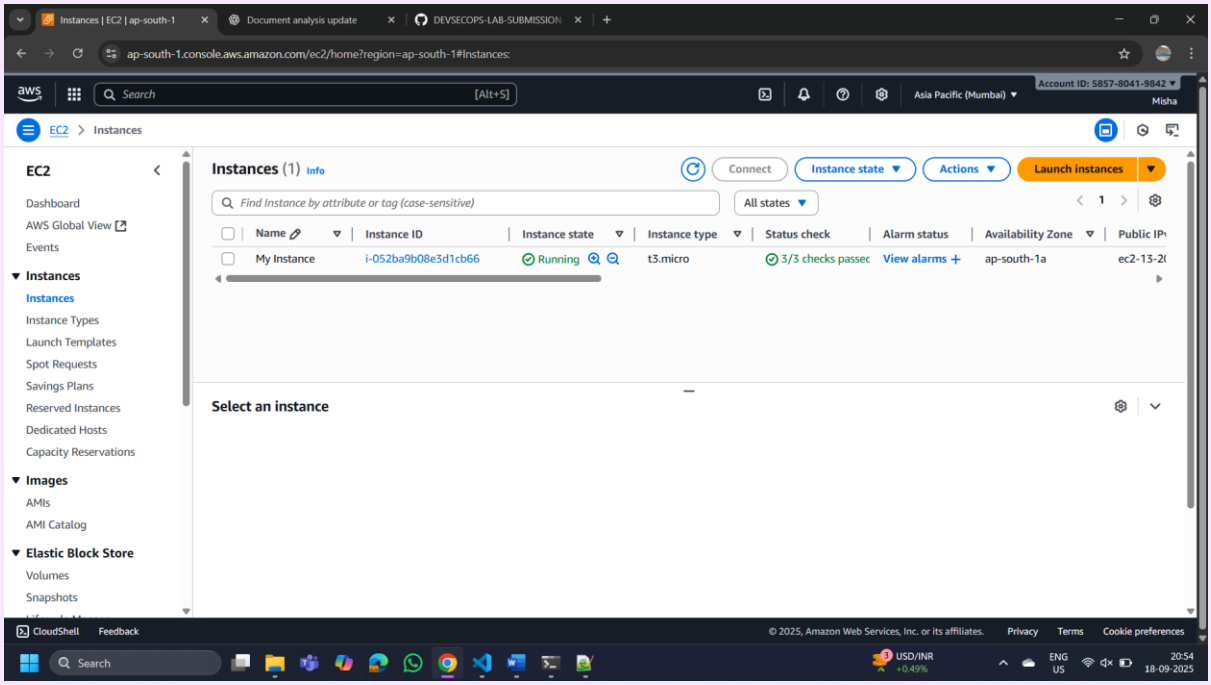
terraform init

terraform apply -var-file=prod.tfvars



5. Test and Verify:

- Observe how different tfvars files are used to set variable values for different environments during the apply process.
- Access the AWS Management Console or use the AWS CLI to verify the creation of resources in the specified regions and instance types.



6. Clean Up:

- After testing, you can clean up resources:

terraform destroy -var-file=dev.tfvars

terraform destroy -var-file=prod.tfvars

```
C:\Terraform\terraform-multiple-tfvars>terraform destroy -var-file=prod.tfvars
var.region
  AWS region to deploy resources into

  Enter a value: ap-south-1


No changes. No objects need to be destroyed.

Either you have not created any objects yet or the existing objects were already deleted outside of Terraform.

Destroy complete! Resources: 0 destroyed.
```

- Confirm the destruction by typing yes.

7. Conclusion:

This lab exercise demonstrates how to use multiple tfvars files in Terraform to manage variable values for different environments. It allows you to maintain separate configuration files for different environments, making it easier to manage and maintain your infrastructure code. Experiment with different values in the dev.tfvars and prod.tfvars files to observe how they impact the infrastructure provisioning process for each environment.