Lab Exercise 8– Terraform Multiple tfvars Files

Objective:

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

Prerequisites:

- **Terraform installed on your machine.**
- **Basic knowledge of Terraform configuration and variables.**

Steps:

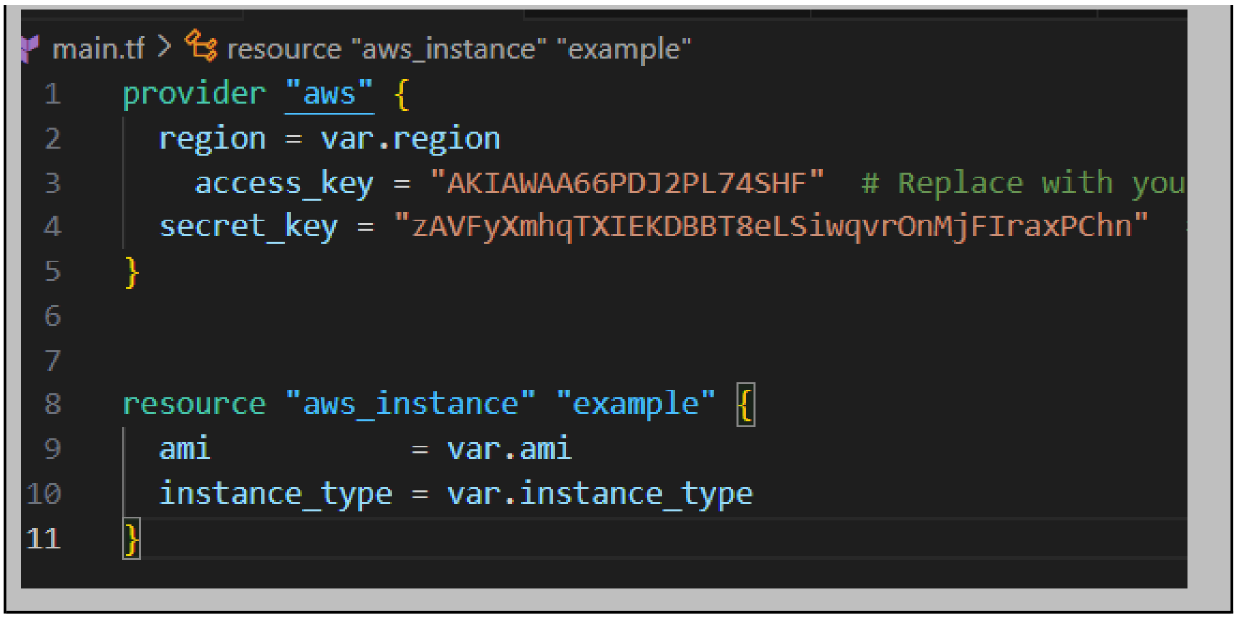
1. Create a Terraform Directory:

```
mkdir terraform-multiple-tfvars
cd 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
}
```



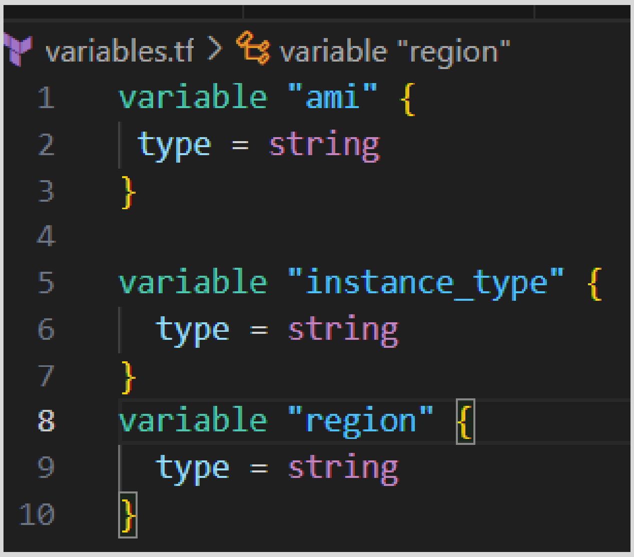
main.tf > resource "aws_instance" "example"

```
provider "aws" {
  region = var.region
    access_key = "AKIAWAA66PDJ2PL74SHF"  # Replace with you
  secret_key = "zAVFyXmhqTXIEKDBBT8eLSiwqvrOnMjFIraxPChn"
}


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
}
```



variables.tf > variable "region"

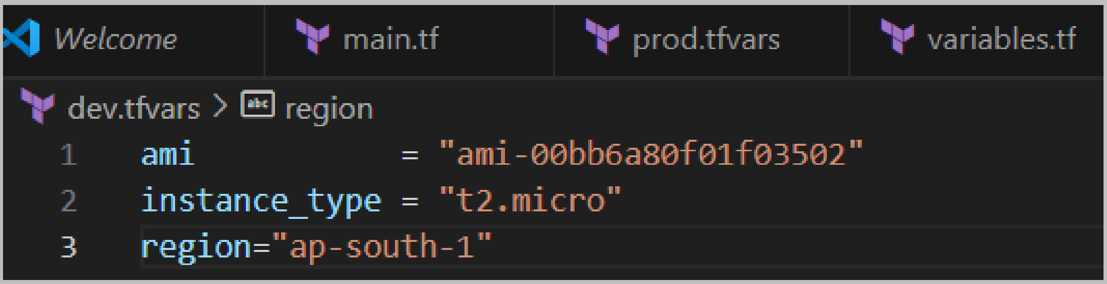
```
variable "ami" {
  type = string
}

variable "instance_type" {
  type = string
}
variable "region" {
  type = string
}
```

2. Create Multiple tfvars Files:

- **Create a file named dev.tfvars:**

# dev.tfvars

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

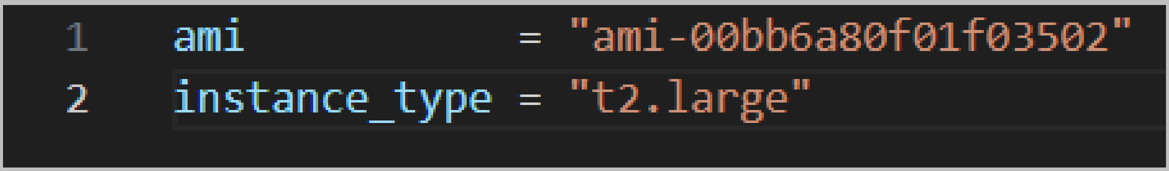


```
ami           = "ami-00bb6a80f01f03502"
instance_type = "t2.micro"
region="ap-south-1"
```

- **Create a file named prod.tfvars:**

# prod.tfvars

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



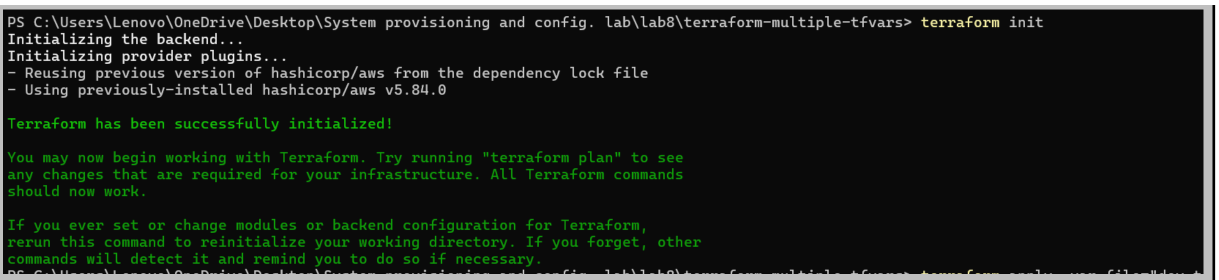
```
ami           = "ami-00bb6a80f01f03502"
instance_type = "t2.large"
```

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

3. Initialize and Apply for Dev Environment:

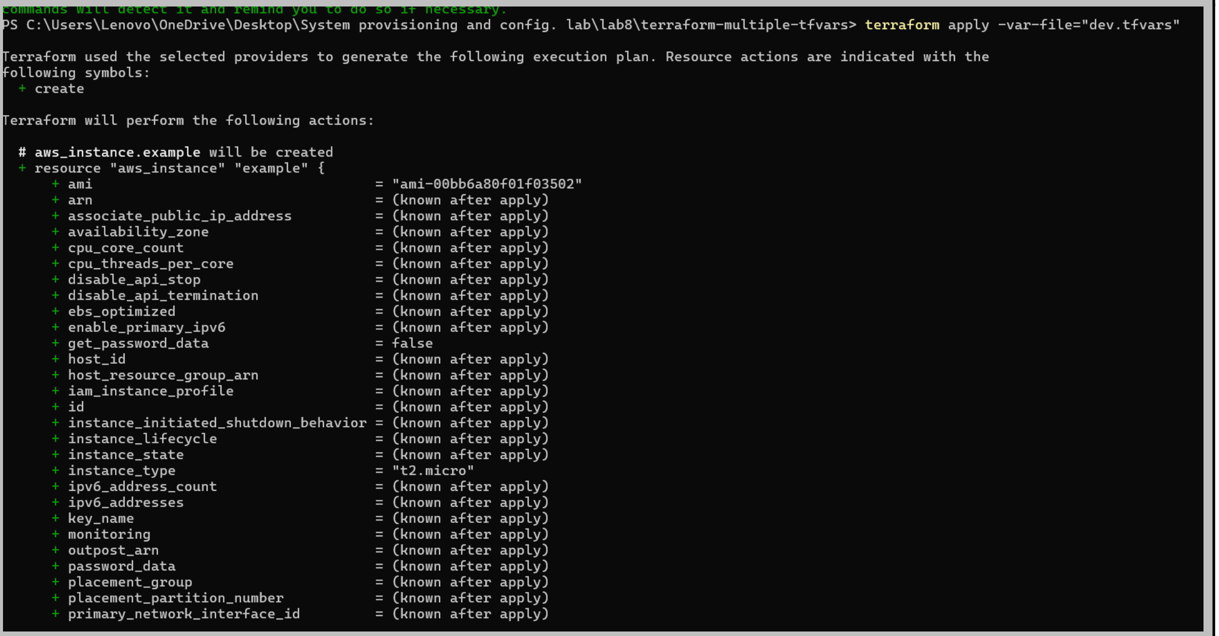
- **Run the following Terraform commands to initialize and apply the configuration for the dev environment:**

```
terraform init
```

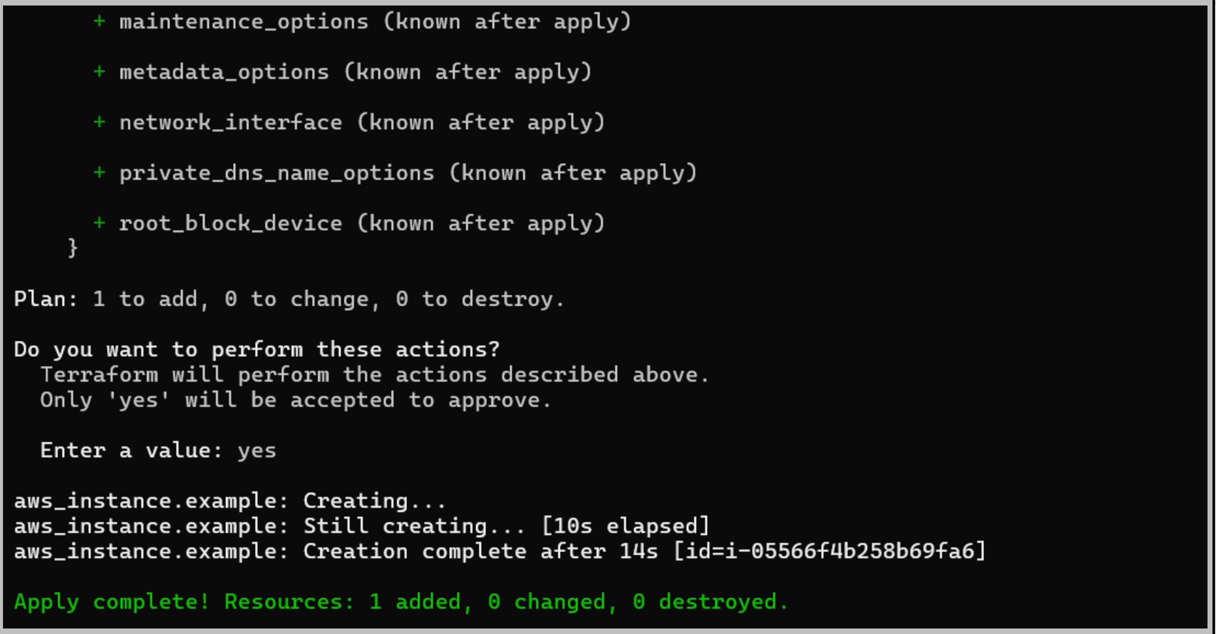


```
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\lab8\terraform-multiple-tfvars> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.84.0

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.
```

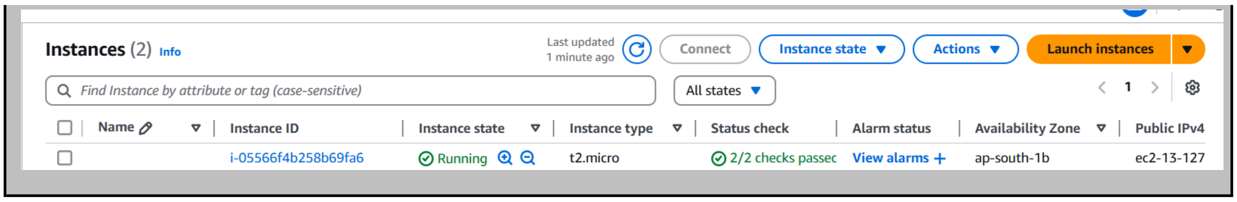
terraform apply -var-file=dev.tfvars



```
commands will detect it and remind you to do so if necessary.
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\lab8\terraform-multiple-tfvars> terraform apply -var-file="dev.tfvars"

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)
      + enable_primary_ipv6                  = (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)
```



```
      + 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.

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-05566f4b258b69fa6]

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



Instances (2) Info

Last updated 1 minute ago | Connect | Instance state ▼ | Actions ▼ | Launch instances ▼

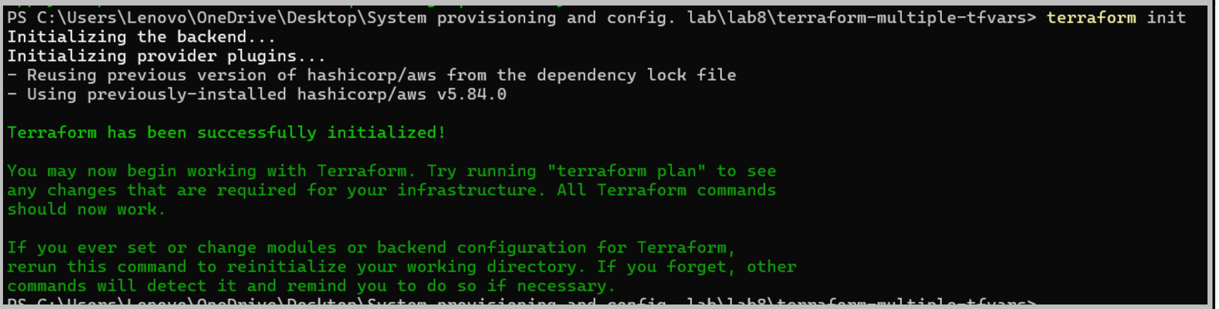
Find Instance by attribute or tag (case-sensitive) | All states ▼

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
☐		i-05566f4b258b69fa6	Running	t2.micro	2/2 checks passec	View alarms +	ap-south-1b	ec2-13-127

4. Initialize and Apply for Prod Environment:

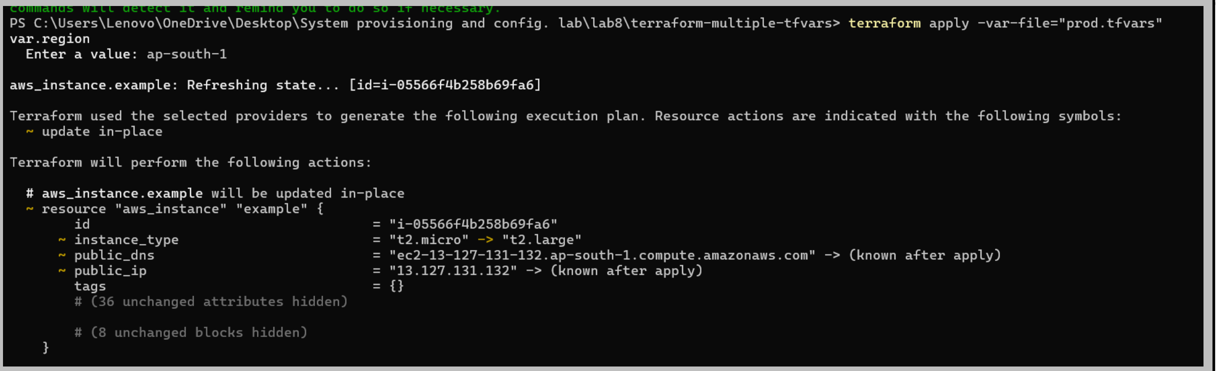
- **Run the following Terraform commands to initialize and apply the configuration for the prod environment:**

terraform init

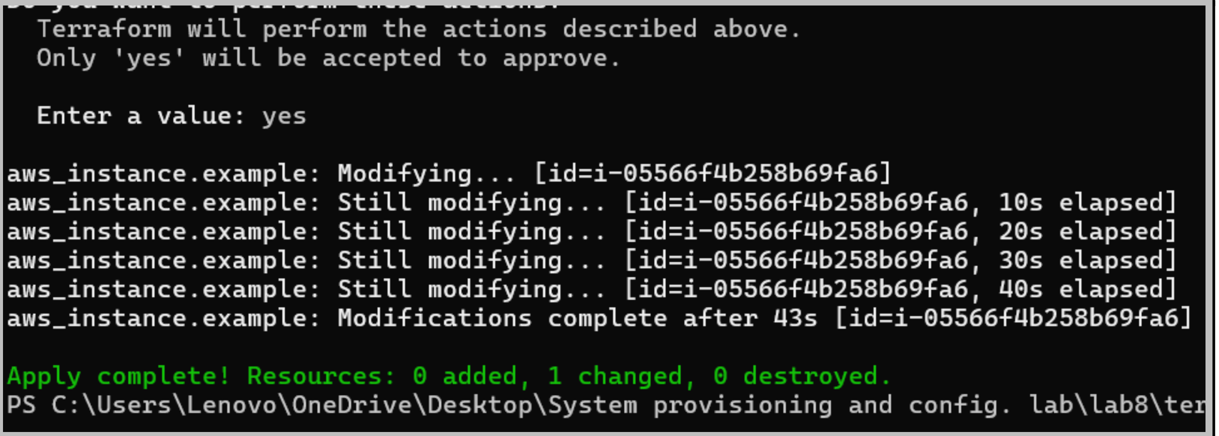


```
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\lab8\terraform-multiple-tfvars> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.84.0

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.
```

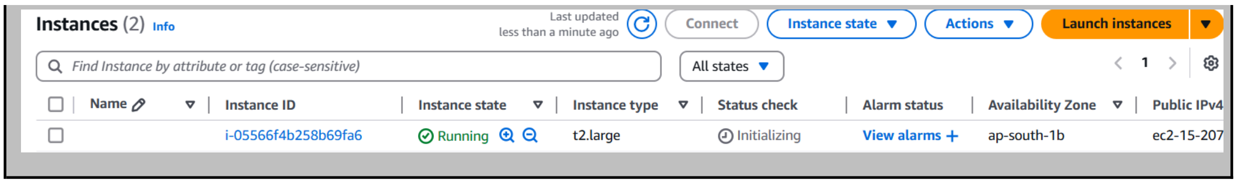
terraform apply -var-file=prod.tfvars



```
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\lab8\terraform-multiple-tfvars> terraform apply -var-file="prod.tfvars"
var.region
  Enter a value: ap-south-1

aws_instance.example: Refreshing state... [id=i-05566f4b258b69fa6]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_instance.example will be updated in-place
  ~ resource "aws_instance" "example" {
        id                                   = "i-05566f4b258b69fa6"
      ~ instance_type                        = "t2.micro" -> "t2.large"
      ~ public_dns                           = "ec2-13-127-131-132.ap-south-1.compute.amazonaws.com" -> (known after apply)
      ~ public_ip                            = "13.127.131.132" -> (known after apply)
        tags                                 = {}
        # (36 unchanged attributes hidden)

        # (8 unchanged blocks hidden)
    }
```



```
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.example: Modifying... [id=i-05566f4b258b69fa6]
aws_instance.example: Still modifying... [id=i-05566f4b258b69fa6, 10s elapsed]
aws_instance.example: Still modifying... [id=i-05566f4b258b69fa6, 20s elapsed]
aws_instance.example: Still modifying... [id=i-05566f4b258b69fa6, 30s elapsed]
aws_instance.example: Still modifying... [id=i-05566f4b258b69fa6, 40s elapsed]
aws_instance.example: Modifications complete after 43s [id=i-05566f4b258b69fa6]

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\lab8\ter
```



Instances (2) Info | Last updated less than a minute ago | Connect | Instance state | Actions | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
	i-05566f4b258b69fa6	Running	t2.large	Initializing	View alarms +	ap-south-1b	ec2-15-207

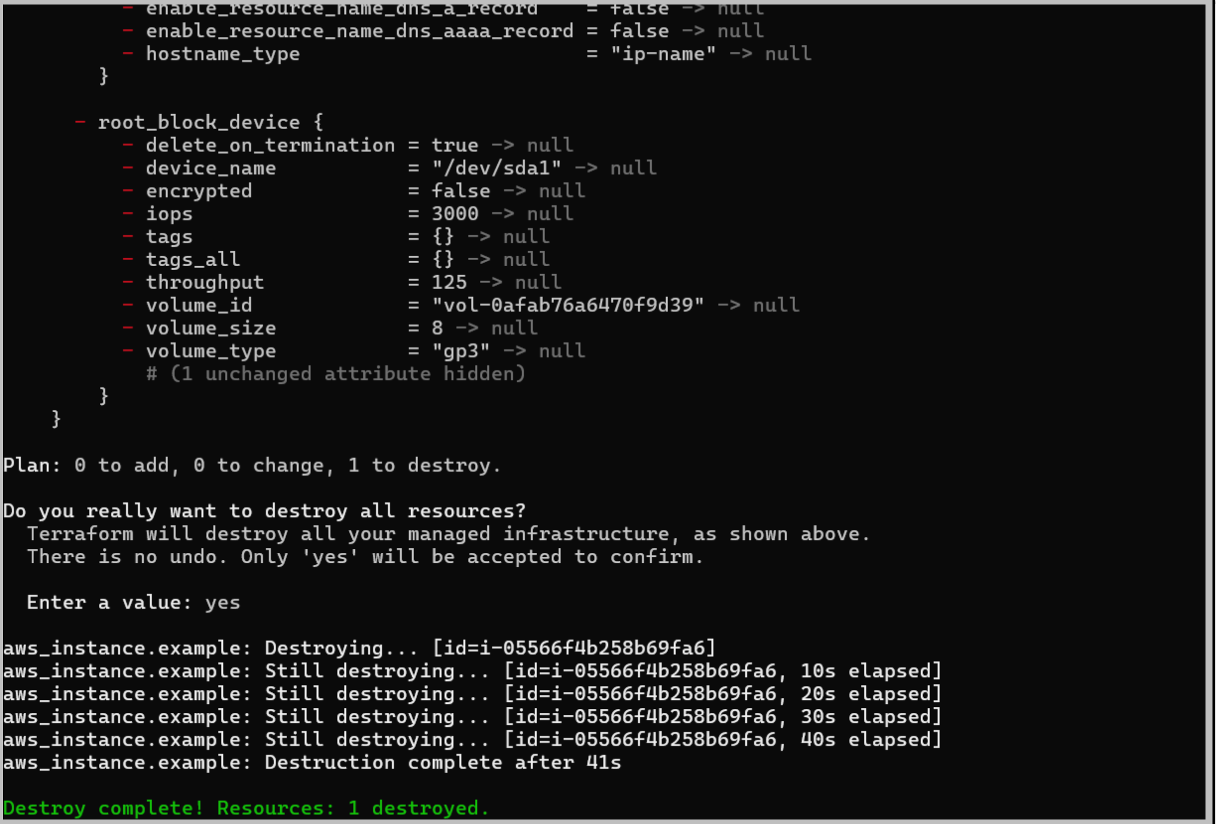
5. Test and Verify:

- **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:

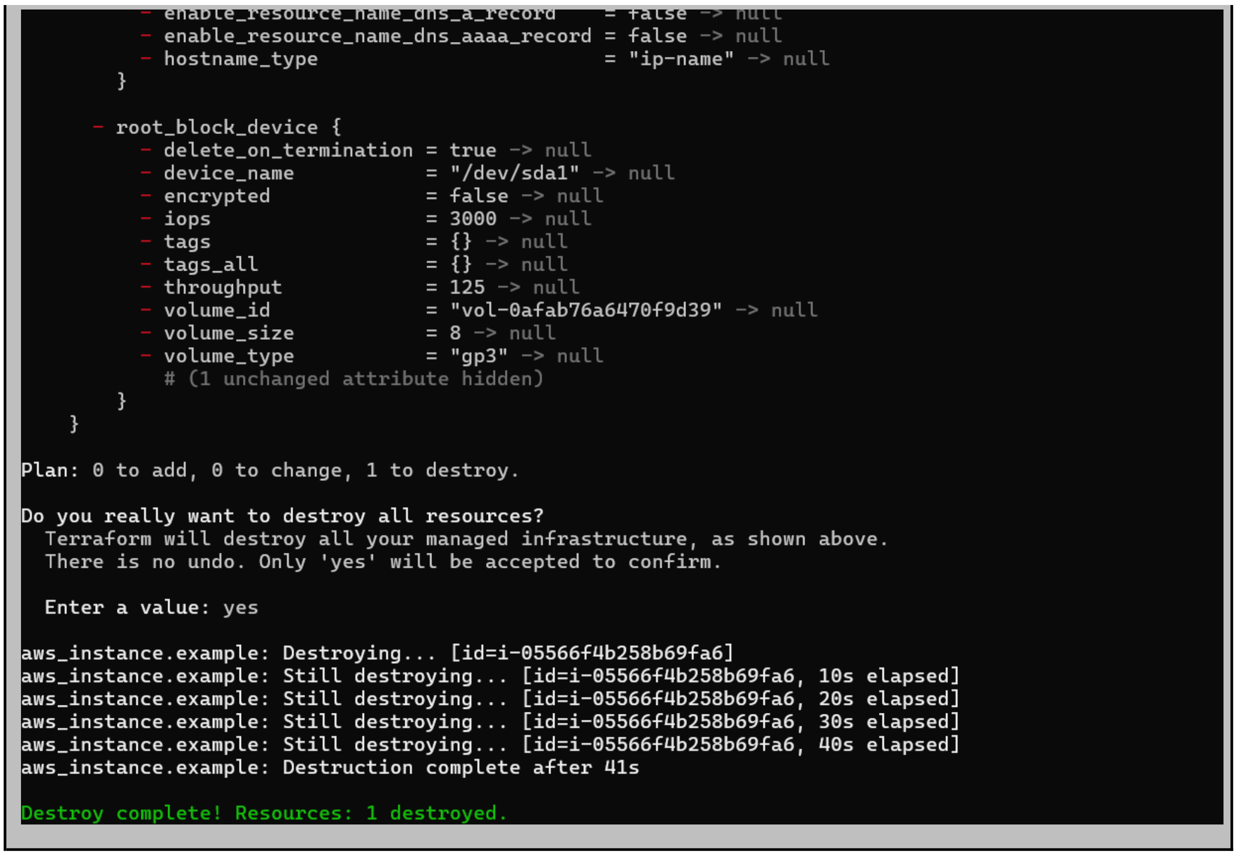
- **After testing, you can clean up resources:**

terraform destroy -var-file=dev.tfvars



```
          - 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-0afab76a6470f9d39" -> 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-05566f4b258b69fa6]
aws_instance.example: Still destroying... [id=i-05566f4b258b69fa6, 10s elapsed]
aws_instance.example: Still destroying... [id=i-05566f4b258b69fa6, 20s elapsed]
aws_instance.example: Still destroying... [id=i-05566f4b258b69fa6, 30s elapsed]
aws_instance.example: Still destroying... [id=i-05566f4b258b69fa6, 40s elapsed]
aws_instance.example: Destruction complete after 41s

Destroy complete! Resources: 1 destroyed.
```

terraform destroy -var-file=prod.tfvars



```
          - 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-0afab76a6470f9d39" -> 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-05566f4b258b69fa6]
aws_instance.example: Still destroying... [id=i-05566f4b258b69fa6, 10s elapsed]
aws_instance.example: Still destroying... [id=i-05566f4b258b69fa6, 20s elapsed]
aws_instance.example: Still destroying... [id=i-05566f4b258b69fa6, 30s elapsed]
aws_instance.example: Still destroying... [id=i-05566f4b258b69fa6, 40s elapsed]
aws_instance.example: Destruction complete after 41s

Destroy complete! Resources: 1 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.