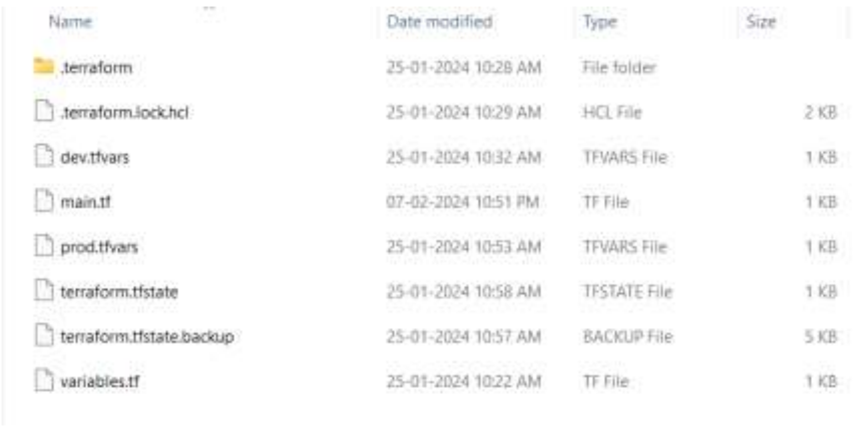
EXPERIMENT – 6

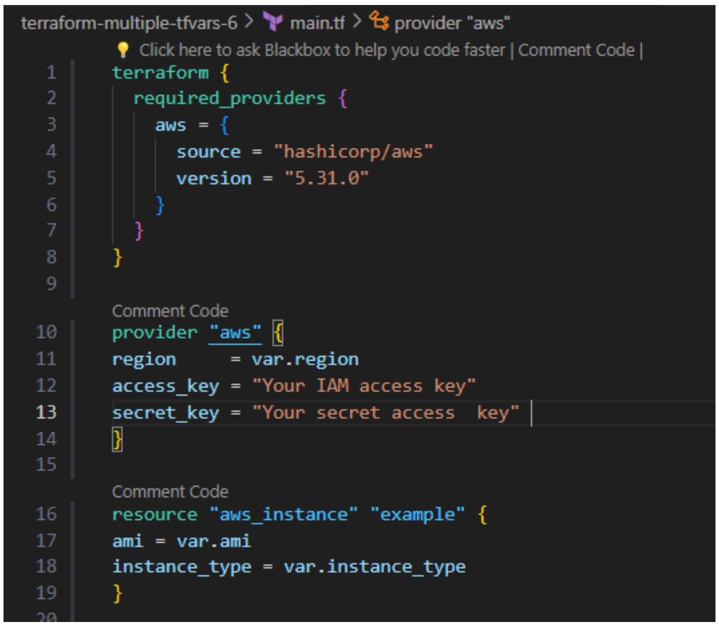
Name: - Shashwat. Dnyaneshwar Kamdi	
Batch – 2 [DevOps Non-Hons]	
SAP ID- 500092140	
Subject – System Provisioning and Configuration Management Lab	

Aim: Terraform Multiple tfvars Files.

1] Create a new directory and Create terraform Configuration File (main.tf)



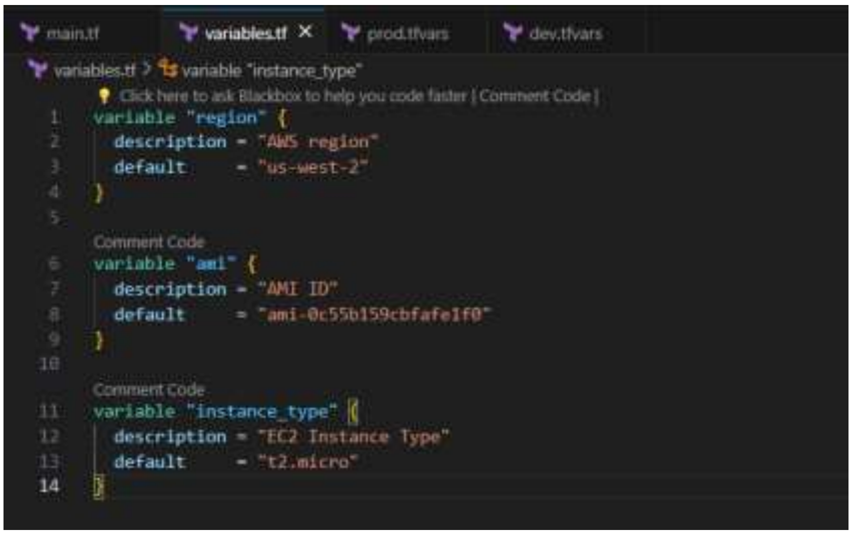
Name	Date modified	Type	Size
.terraform	25-01-2024 10:28 AM	File folder	
.terraform.lock.hcl	25-01-2024 10:29 AM	HCL File	2 KB
dev.tfvars	25-01-2024 10:32 AM	TFVARS File	1 KB
main.tf	07-02-2024 10:51 PM	TF File	1 KB
prod.tfvars	25-01-2024 10:53 AM	TFVARS File	1 KB
terraform.tfstate	25-01-2024 10:58 AM	TFSTATE File	1 KB
terraform.tfstate.backup	25-01-2024 10:57 AM	BACKUP File	5 KB
variables.tf	25-01-2024 10:22 AM	TF File	1 KB



terraform-multiple-tfvars-6 > main.tf > provider "aws"

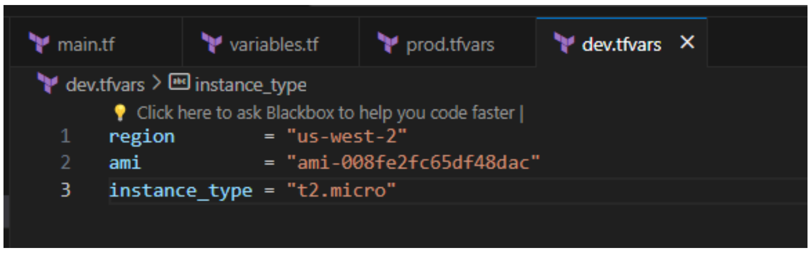
```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.31.0"
    }
  }
}

provider "aws" {
region     = var.region
access_key = "Your IAM access key"
secret_key = "Your secret access  key"
}

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

2] Create a file named as “variable.tf”

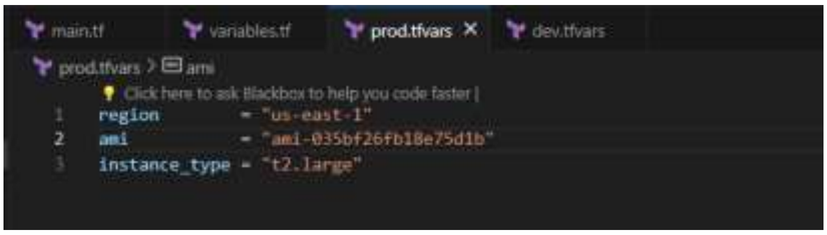

main.tf
variables.tf
prod.tfvars
dev.tfvars
variables.tf > variable "instance_type"
Click here to ask Blackbox to help you code faster | Comment Code |
Comment Code
Comment Code

```
variable "region" {
  description = "AWS region"
  default     = "us-west-2"
}

variable "ami" {
  description = "AMI ID"
  default     = "ami-0c55b159cbfafe1f0"
}

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

3] Create Multiple tfvars Files:
  i) dev.tfvars
  ii) prod.tfvars

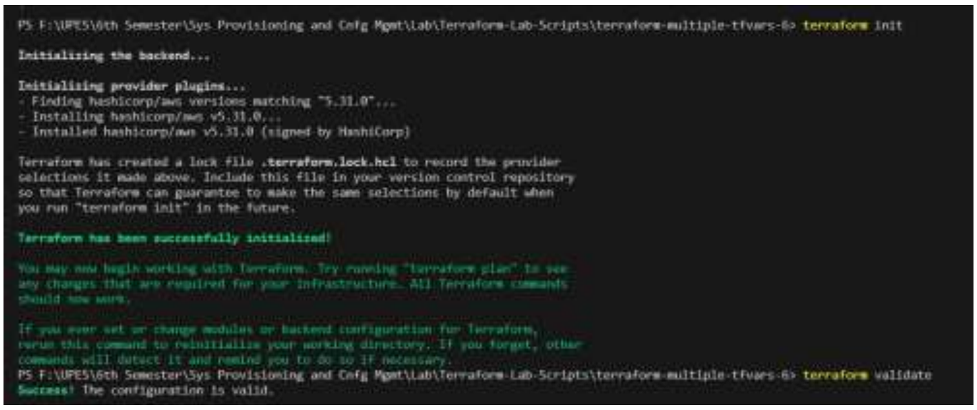

main.tf
variables.tf
prod.tfvars
dev.tfvars
dev.tfvars > instance_type
Click here to ask Blackbox to help you code faster |

```
region        = "us-west-2"
ami           = "ami-008fe2fc65df48dac"
instance_type = "t2.micro"
```

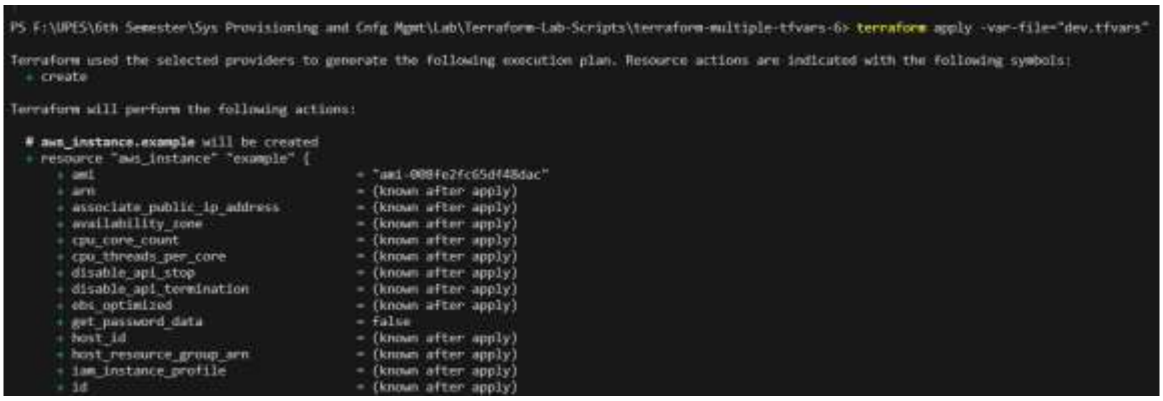


```
region        = "us-east-1"
ami           = "ami-035bf26fb18e75d1b"
instance_type = "t2.large"
```

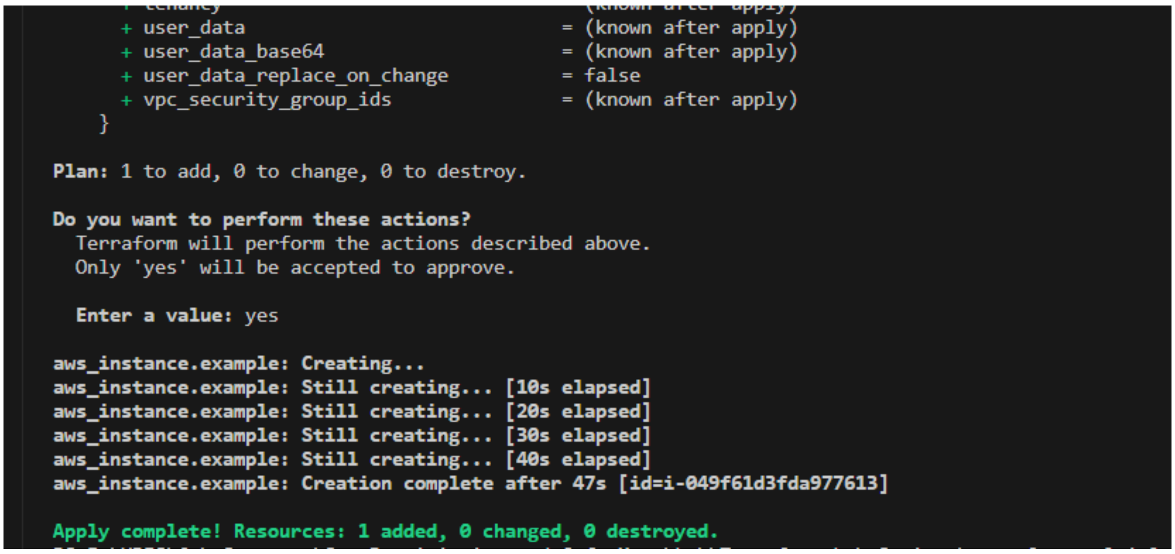
4] Initialize Terraform for Dev Environment and apply it using command "Terraform apply"



```
PS F:\UPES\6th Semester\Sys Provisioning and Cnfg Mgmt\Lab\Terraform-Lab-Scripts\terraform-multiple-tfvars-6> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.31.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (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 F:\UPES\6th Semester\Sys Provisioning and Cnfg Mgmt\Lab\Terraform-Lab-Scripts\terraform-multiple-tfvars-6> terraform validate
Success! The configuration is valid.
```

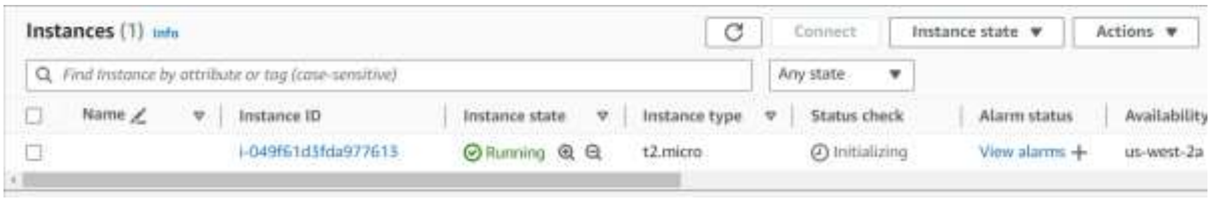


```
PS F:\UPES\6th Semester\Sys Provisioning and Cnfg Mgmt\Lab\Terraform-Lab-Scripts\terraform-multiple-tfvars-6> terraform apply -var-file="dev.tfvars"

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-008fe2fc65df48dac"
      + 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)
```



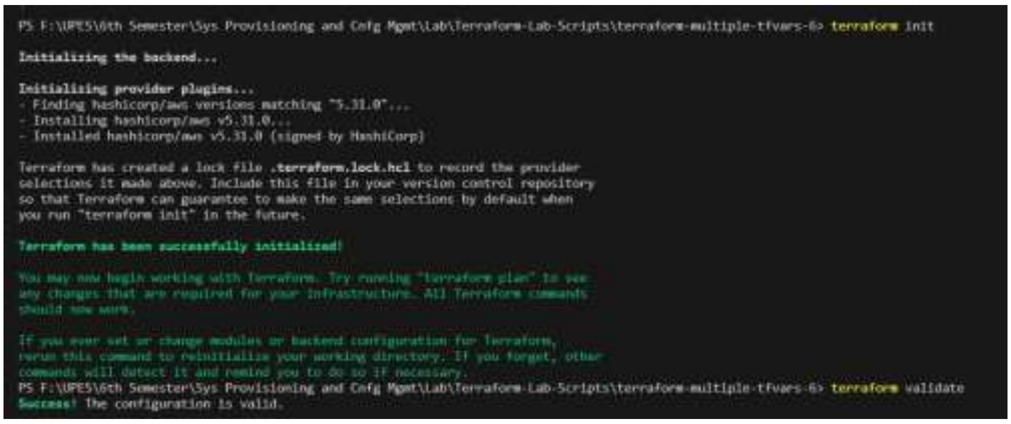
```
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = (known after apply)
    }

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: Still creating... [30s elapsed]
aws_instance.example: Still creating... [40s elapsed]
aws_instance.example: Creation complete after 47s [id=i-049f61d3fda977613]

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

5] Verify Resources on AWS Management Console for <u>Dev Environment</u>

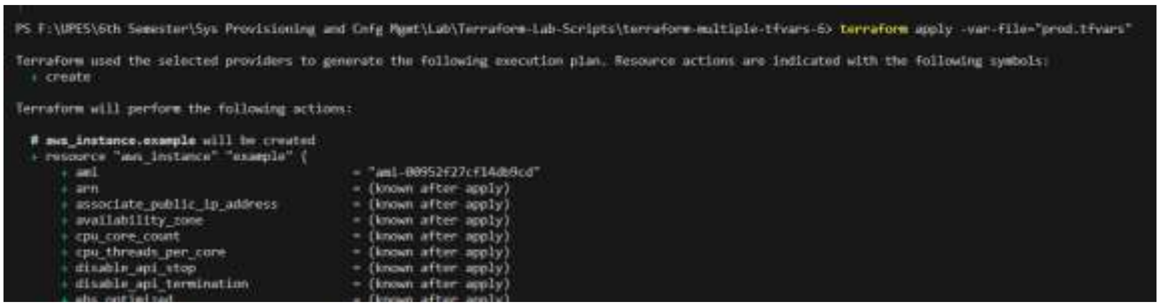


Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
	i-049f61d3fda977613	Running	t2.micro	Initializing	View alarms +	us-west-2a

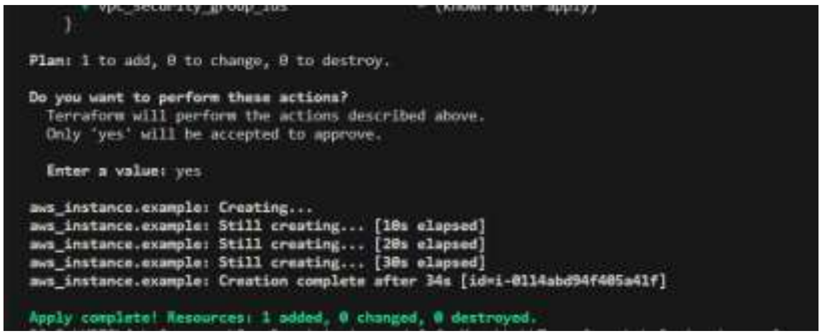
6] Initialize Terraform for <u>Prod Environment</u> and apply it using command "Terraform apply"



```
PS F:\UPES\6th Semester\Sys Provisioning and Cnfg Mgmt\Lab\Terraform-Lab-Scripts\terraform-multiple-tfvars-6> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.31.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (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 F:\UPES\6th Semester\Sys Provisioning and Cnfg Mgmt\Lab\Terraform-Lab-Scripts\terraform-multiple-tfvars-6> terraform validate
Success! The configuration is valid.
```

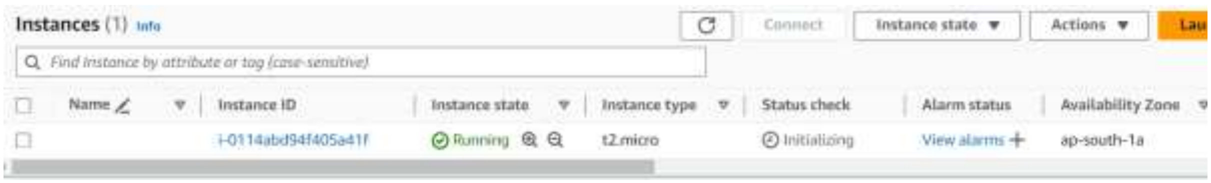


```
PS F:\UPES\6th Semester\Sys Provisioning and Cnfg Mgmt\Lab\Terraform-Lab-Scripts\terraform-multiple-tfvars-6> terraform apply -var-file="prod.tfvars"

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-00952f27cf14db0cd"
      + 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)
```



```
    }

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: Still creating... [30s elapsed]
aws_instance.example: Creation complete after 34s [id=i-0114abd94f405a41f]

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

7] Verify Resources on AWS Management Console for Prod Environment



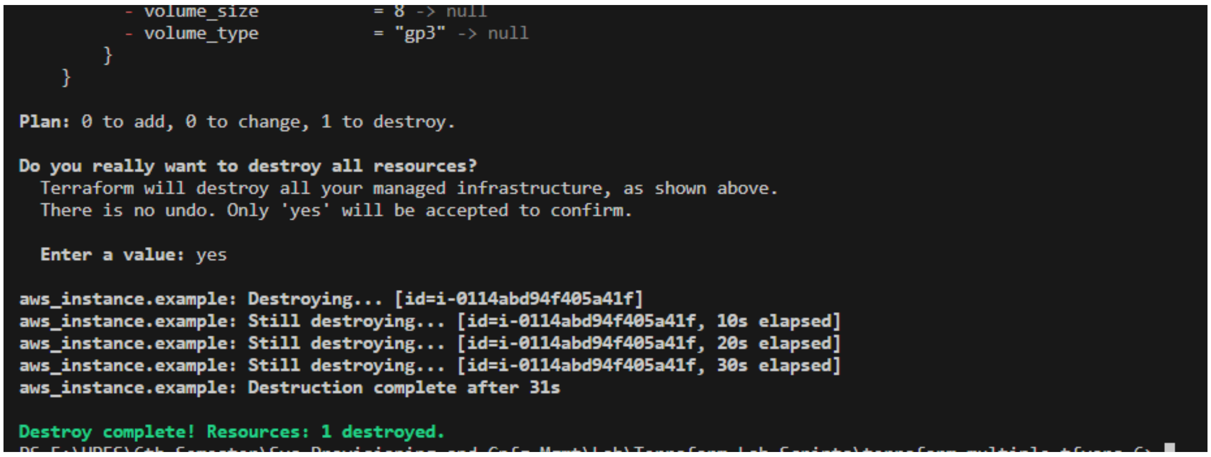
Instances (1) Info

Connect | Instance state ▼ | Actions ▼ | Lau

Find instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
	i-0114abd94f405a41f	Running	t2.micro	Initializing	View alarms +	ap-south-1a

8] Cleanup Resources for Dev and Prod Environment using command "Terraform destroy"



```
          - volume_size           = 8 -> null
          - volume_type           = "gp3" -> 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-0114abd94f405a41f]
aws_instance.example: Still destroying... [id=i-0114abd94f405a41f, 10s elapsed]
aws_instance.example: Still destroying... [id=i-0114abd94f405a41f, 20s elapsed]
aws_instance.example: Still destroying... [id=i-0114abd94f405a41f, 30s elapsed]
aws_instance.example: Destruction complete after 31s

Destroy complete! Resources: 1 destroyed.
```