UPES

UNIVERSITY OF THE FUTURE

System Provisioning and Configuration Management Lab

Siddhant Singh

500095375

Batch - 3

Experiment 6

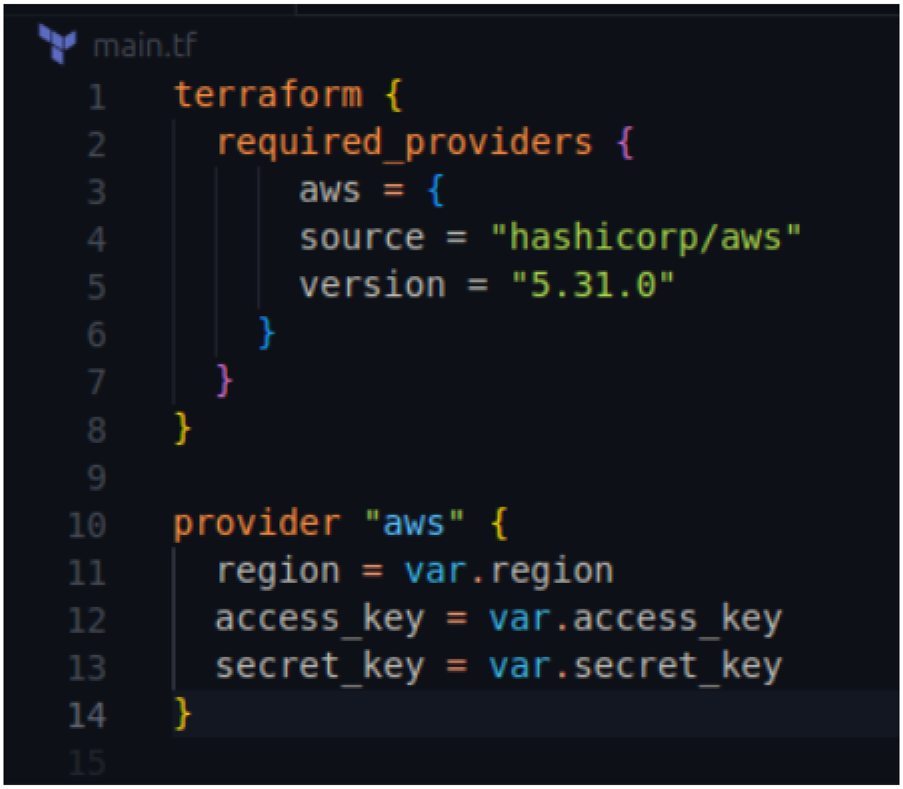
Terraform Multiple tfvars Files

Aim

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

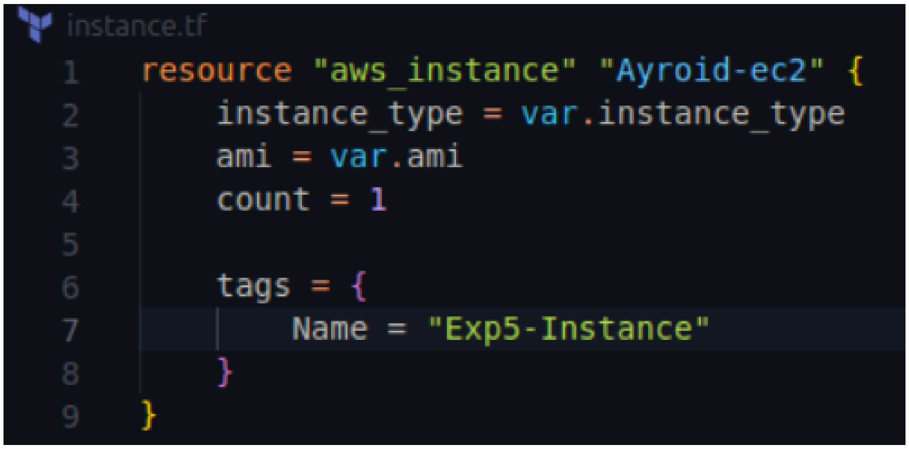
Steps

1. Create a main file, instance.tf file & variables.tf file for EC2 Instance.



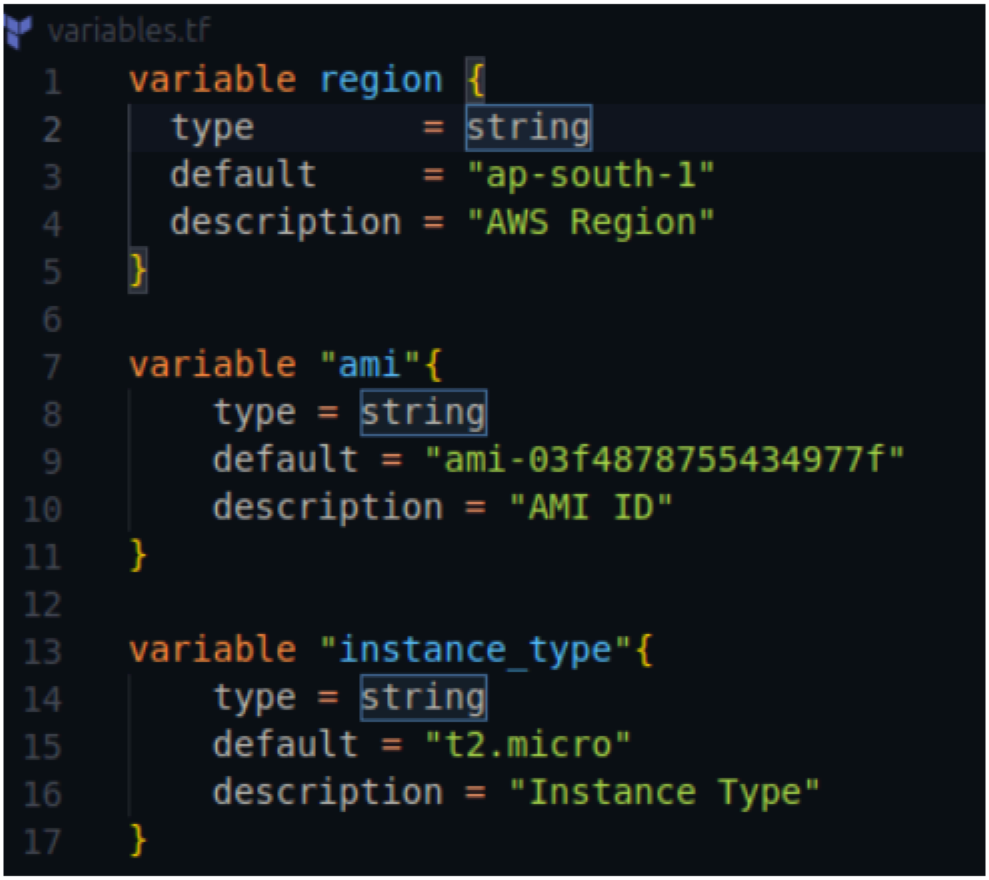
main.tf

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

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



instance.tf

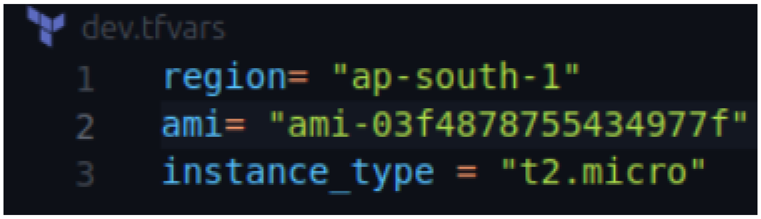
```
resource "aws_instance" "Ayroid-ec2" {
    instance_type = var.instance_type
    ami = var.ami
    count = 1

    tags = {
        Name = "Exp5-Instance"
    }
}
```



```
variable region {
  type        = string
  default     = "ap-south-1"
  description = "AWS Region"
}

variable "ami"{
    type = string
    default = "ami-03f4878755434977f"
    description = "AMI ID"
}

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

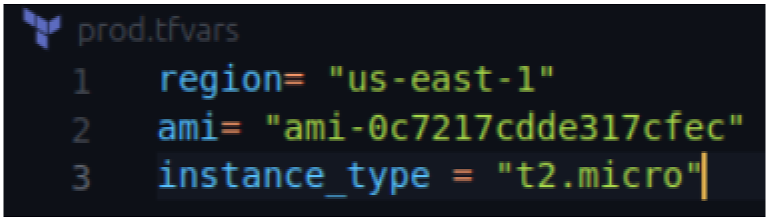
2. Create two tfvars files for different environments

a) dev.tfvars



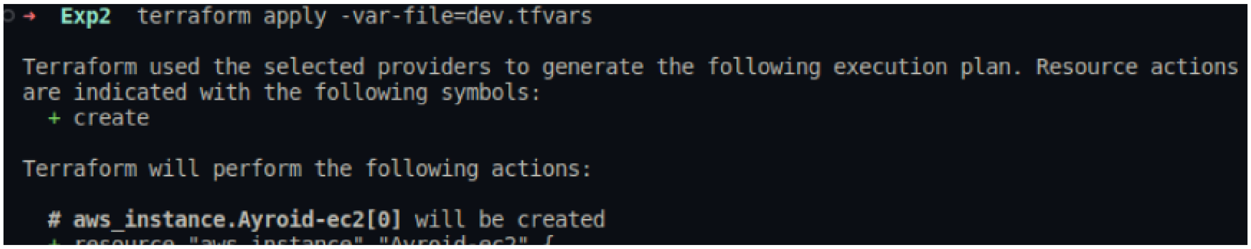
```
region= "ap-south-1"
ami= "ami-03f4878755434977f"
instance_type = "t2.micro"
```

b) prod.tfvars



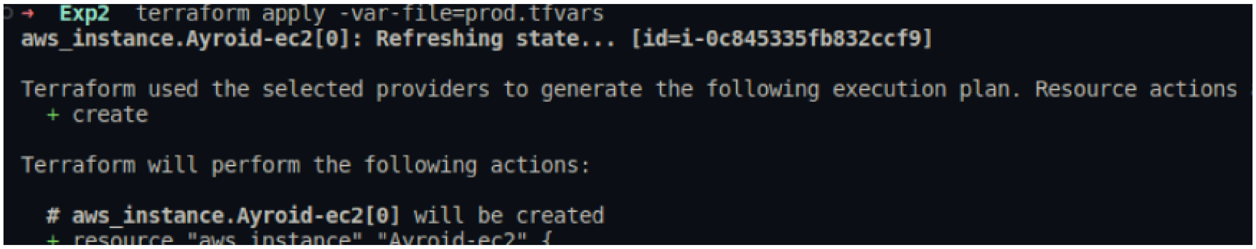
```
region= "us-east-1"
ami= "ami-0c7217cdde317cfec"
instance_type = "t2.micro"
```

3. Initialize and provision resources in both environments
   a) dev



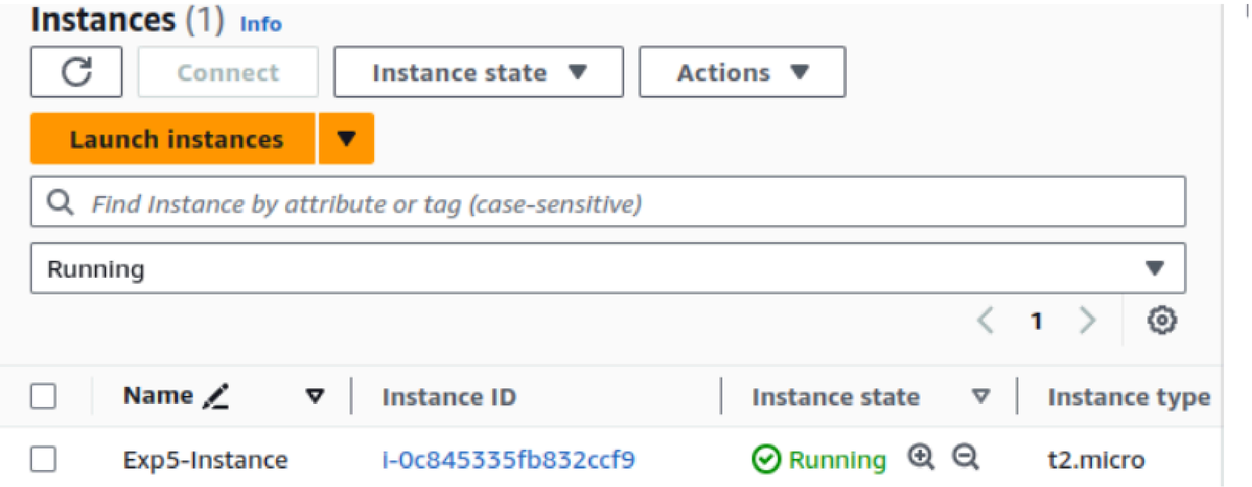
```
→  Exp2  terraform apply -var-file=dev.tfvars

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.Ayroid-ec2[0] will be created
  + resource "aws_instance" "Ayroid-ec2" {
```

   b) prod



```
→  Exp2  terraform apply -var-file=prod.tfvars
aws_instance.Ayroid-ec2[0]: Refreshing state... [id=i-0c845335fb832ccf9]

Terraform used the selected providers to generate the following execution plan. Resource actions
  + create

Terraform will perform the following actions:

  # aws_instance.Ayroid-ec2[0] will be created
  + resource "aws_instance" "Ayroid-ec2" {
```

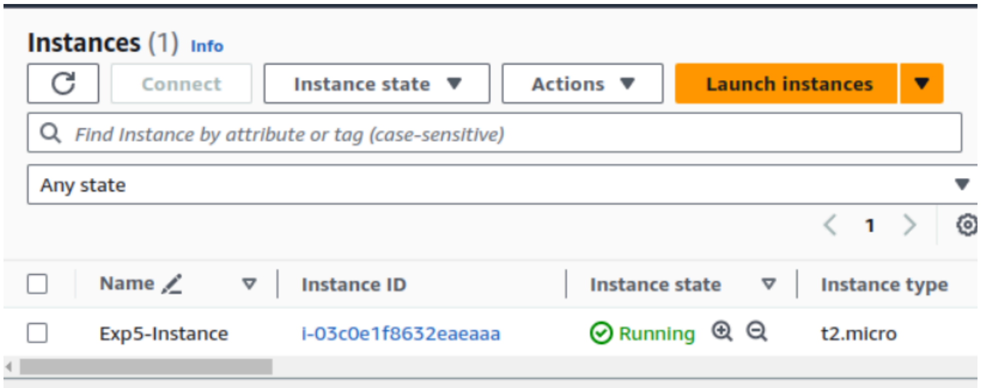
4. Verify resources
   a) dev



Running

☐	Name	Instance ID	Instance state	Instance type
☐	Exp5-Instance	i-0c845335fb832ccf9	✓ Running	t2.micro

   b) prod

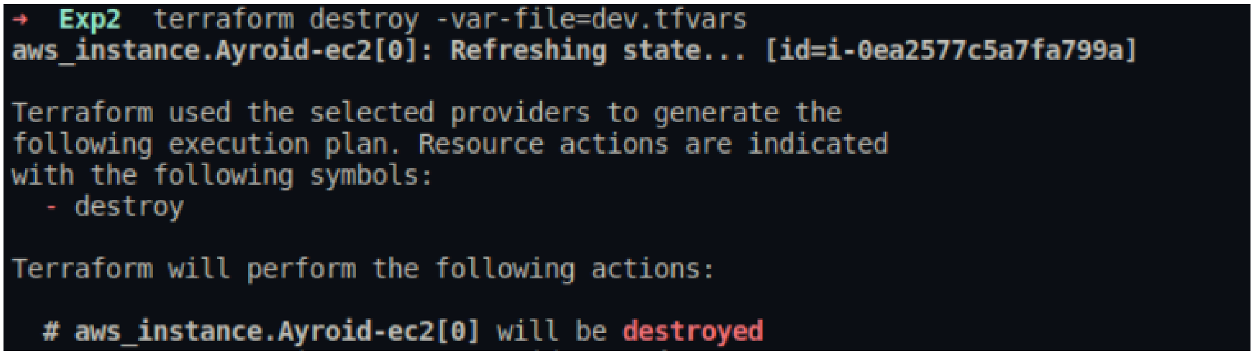


Any state

☐	Name	Instance ID	Instance state	Instance type
☐	Exp5-Instance	i-03c0e1f8632eaeaaa	✓ Running	t2.micro

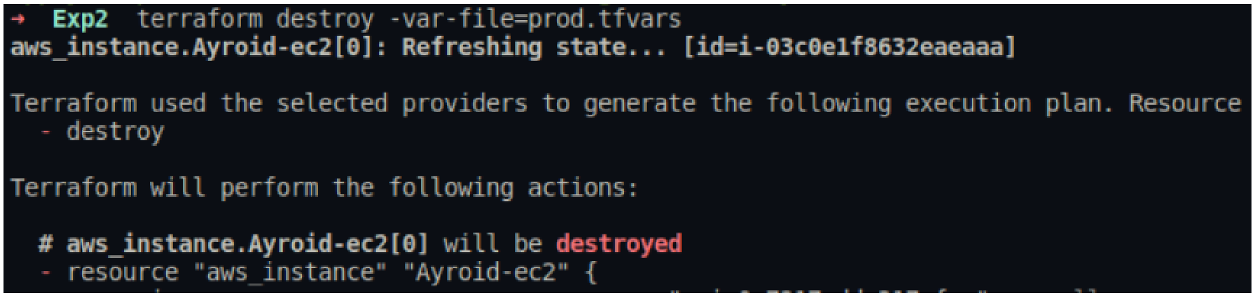
5. Clean up resources in both environments

a) dev



```
➜  Exp2  terraform destroy -var-file=dev.tfvars
aws_instance.Ayroid-ec2[0]: Refreshing state... [id=i-0ea2577c5a7fa799a]

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.Ayroid-ec2[0] will be destroyed
```

b) prod



```
➜  Exp2  terraform destroy -var-file=prod.tfvars
aws_instance.Ayroid-ec2[0]: Refreshing state... [id=i-03c0e1f8632eaeaaa]

Terraform used the selected providers to generate the following execution plan. Resource
  - destroy

Terraform will perform the following actions:

  # aws_instance.Ayroid-ec2[0] will be destroyed
  - resource "aws_instance" "Ayroid-ec2" {
```