Lab Exercise 8– Terraform Mul

Objective:

Learn how to use multiple tfvars files in Terrafor

Prerequisites:

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

Steps:

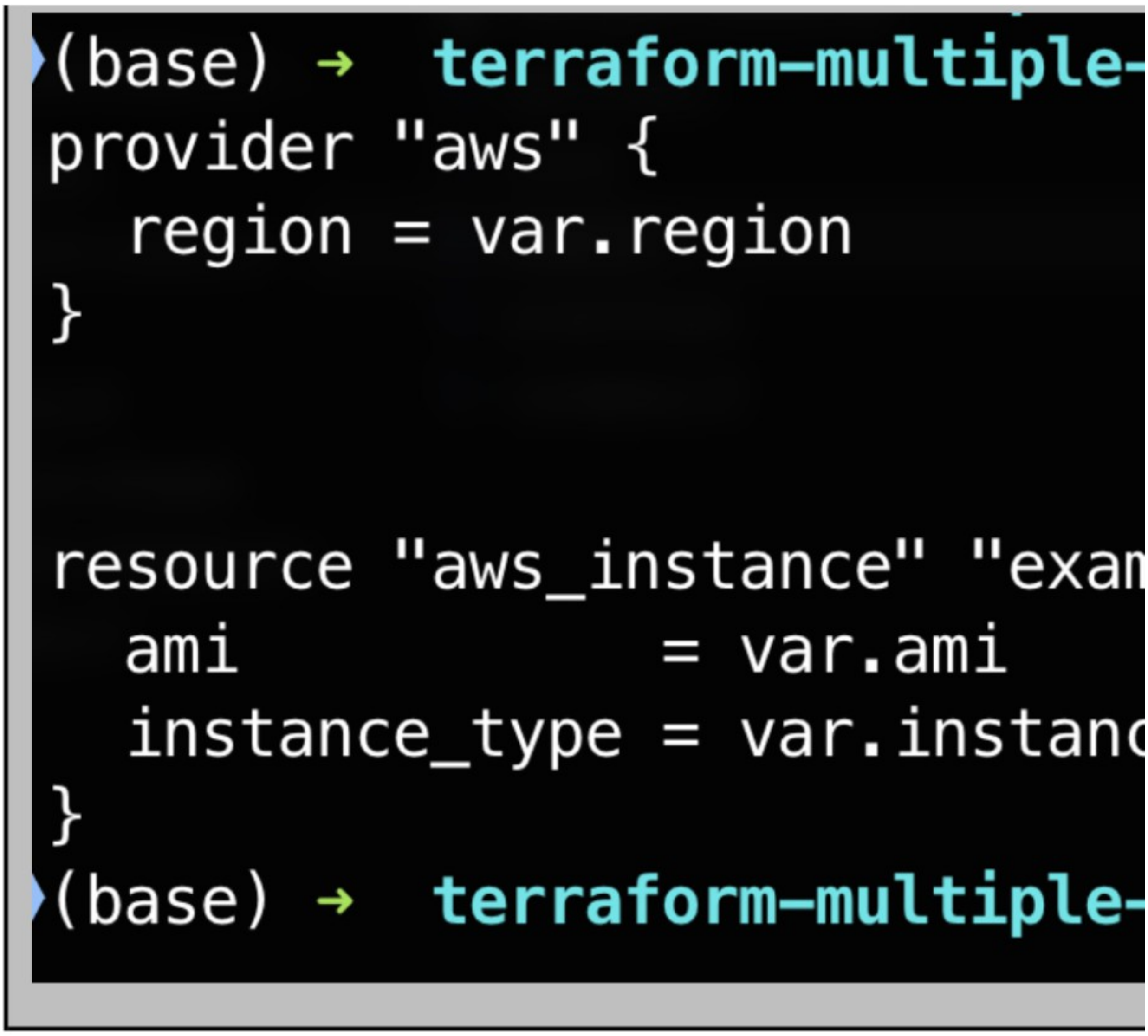
1. Create a Terraform Directory

```
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
}
```

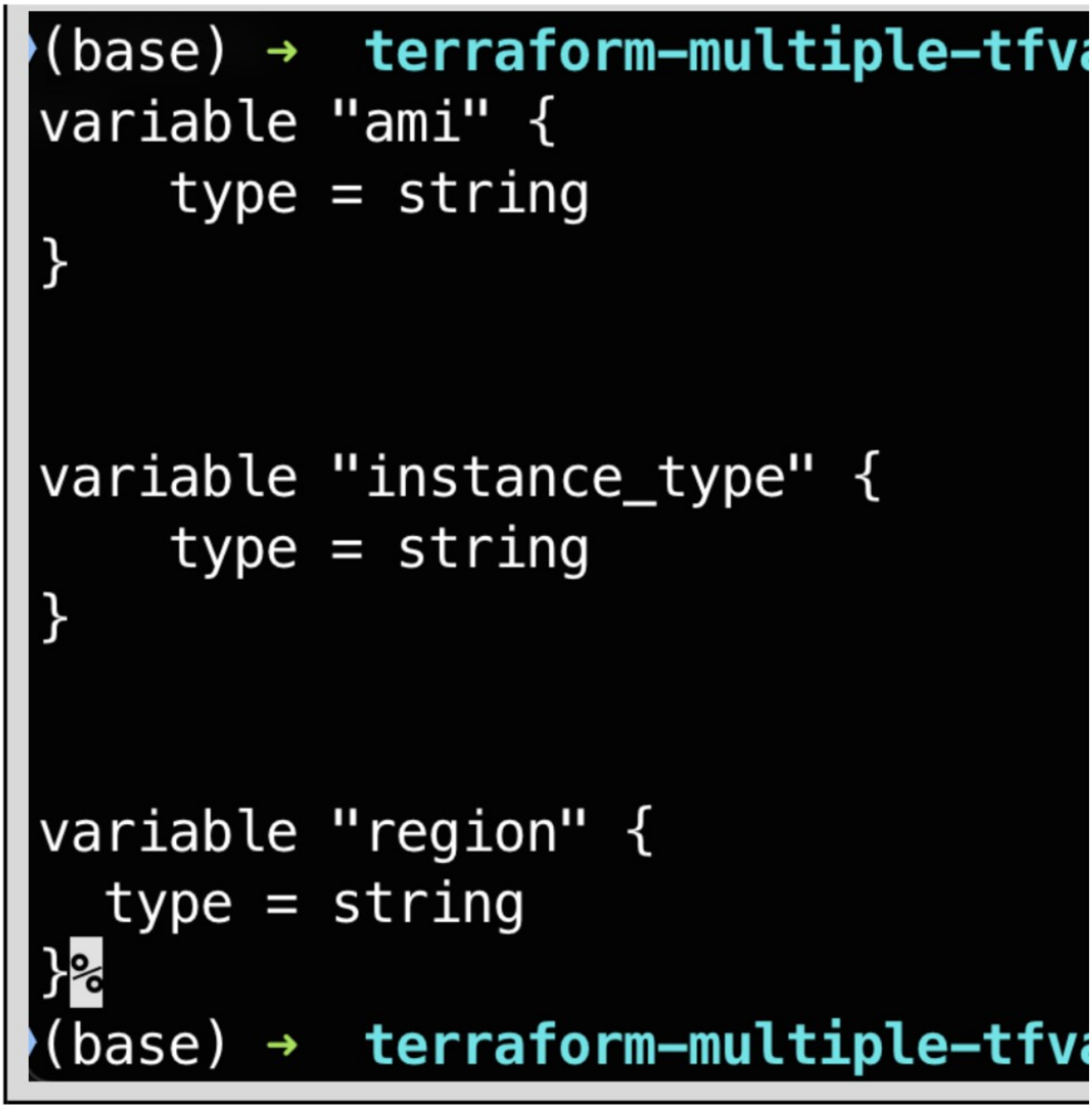


```
(base) → terraform-multiple-
provider "aws" {
  region = var.region
}


resource "aws_instance" "exam
  ami           = var.ami
  instance_type = var.instanc
}
(base) → terraform-multiple-
```

- Create a file named variables.tf:

# variables.tf

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


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



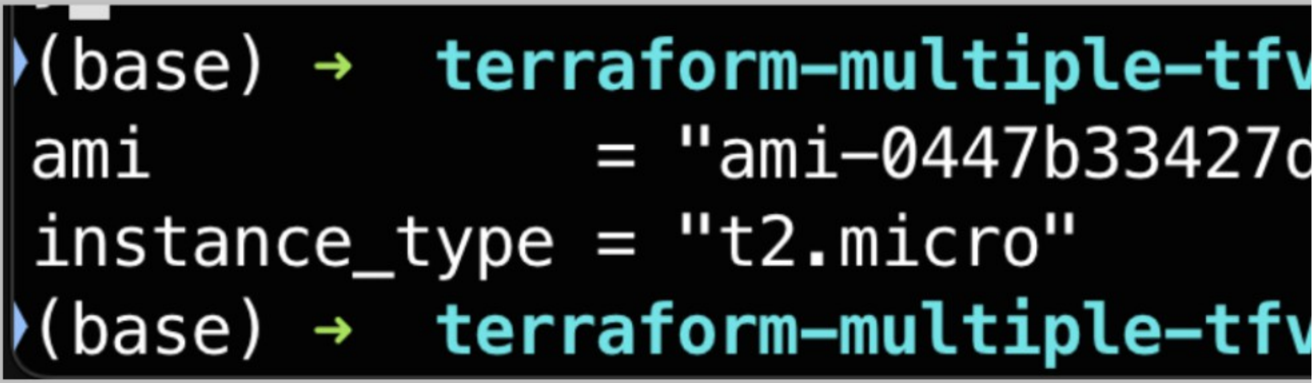
```
(base) → terraform-multiple-tfva
variable "ami" {
    type = string
}


variable "instance_type" {
    type = string
}


variable "region" {
  type = string
}%
(base) → terraform-multiple-tfva
```

2. Create Multiple tfvars Files:

- Create a file named dev.tfvars:

# dev.tfvars

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

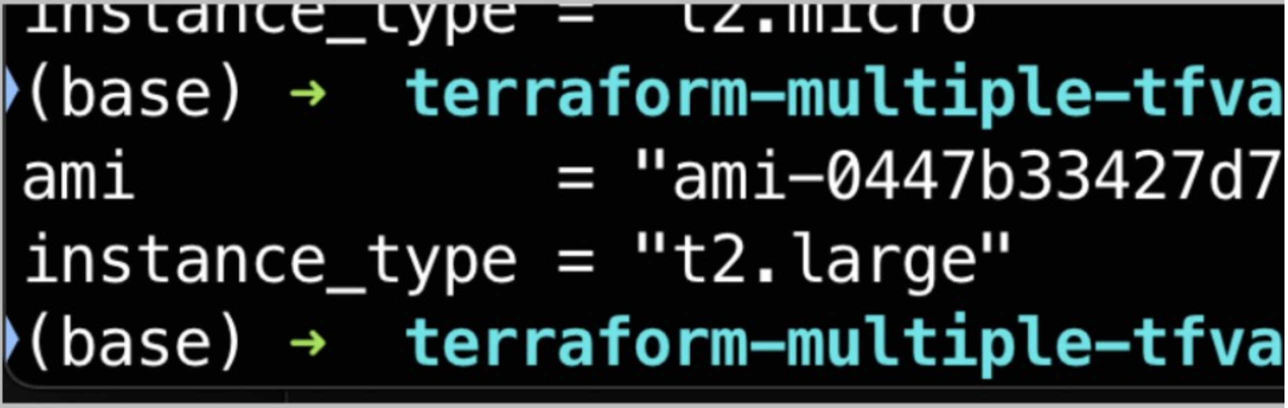


```
(base) → terraform-multiple-tfv
ami                = "ami-0447b33427c
instance_type = "t2.micro"
(base) → terraform-multiple-tfv
```

- Create a file named prod.tfvars:

# prod.tfvars

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



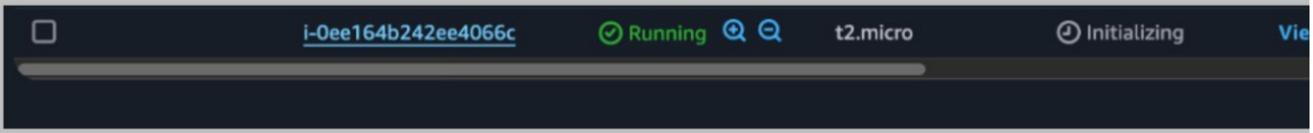
```
instance_type = "t2.micro"
(base) →  terraform-multiple-tfva
ami             = "ami-0447b33427d7
instance_type = "t2.large"
(base) →  terraform-multiple-tfva
```

- In these files, provide values for the variables base

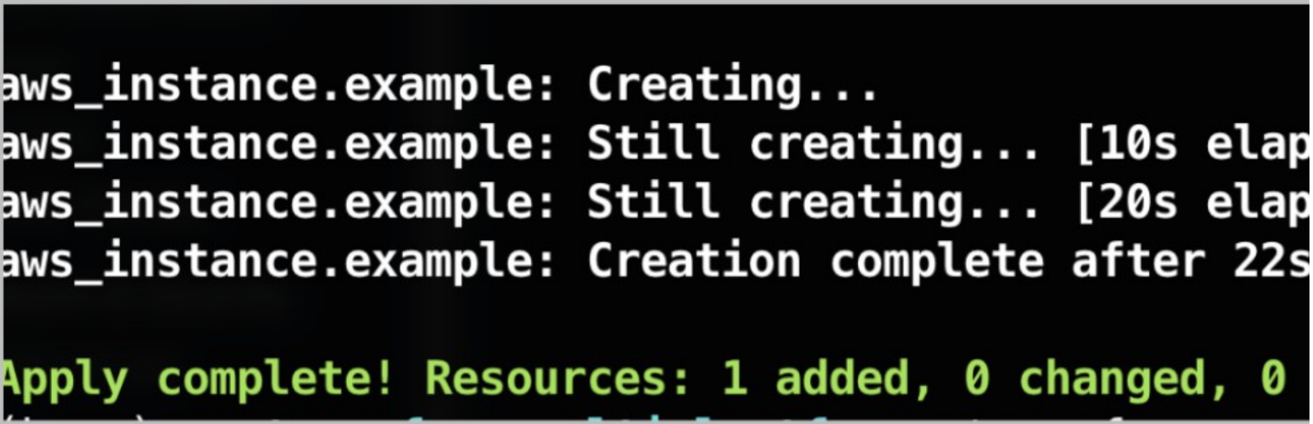
3. Initialize and Apply for Dev Envi

- Run the following Terraform commands to initiali
  for the dev environment:

```
terraform init
terraform apply -var-file=dev.tfvars
```



i-0ee164b242ee4066c ✓ Running t2.micro Initializing Vie

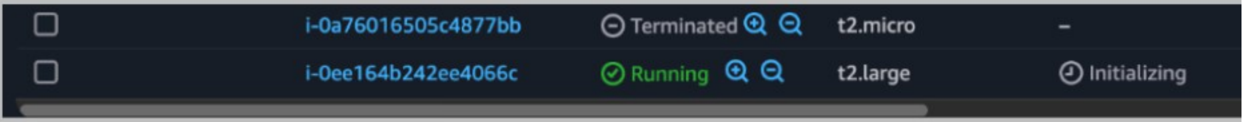


```
aws_instance.example: Creating...
aws_instance.example: Still creating... [10s elap
aws_instance.example: Still creating... [20s elap
aws_instance.example: Creation complete after 22s

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

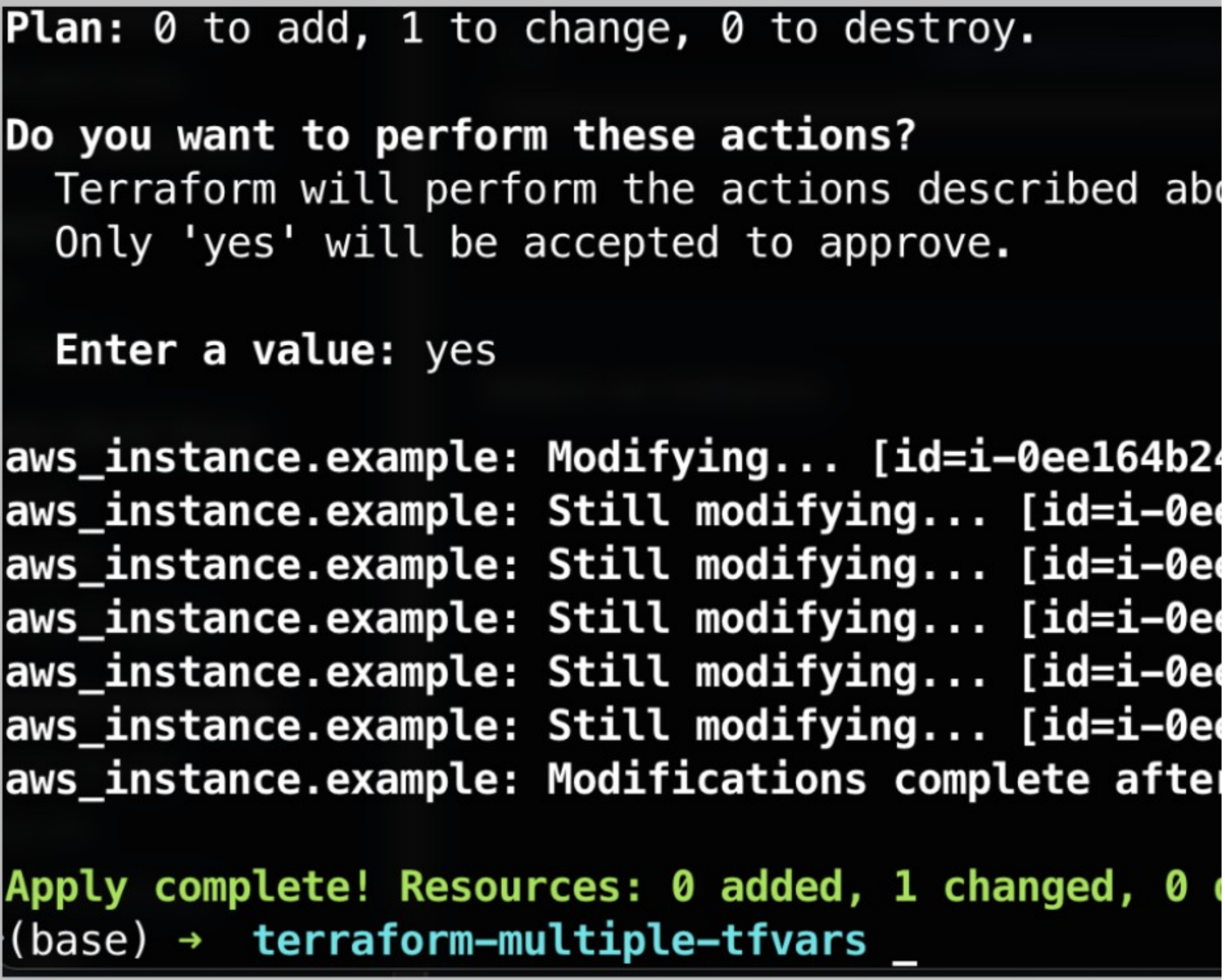
4. Initialize and Apply for Prod Env

- Run the following Terraform commands to init
  for the prod environment:

```
terraform init
terraform apply -var-file=prod.tfvars
```



```
i-0a76016505c4877bb   Terminated   t2.micro   -
i-0ee164b242ee4066c   Running      t2.large   Initializing
```



```
Plan: 0 to add, 1 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described ab
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.example: Modifying... [id=i-0ee164b2
aws_instance.example: Still modifying... [id=i-0e
aws_instance.example: Still modifying... [id=i-0e
aws_instance.example: Still modifying... [id=i-0e
aws_instance.example: Still modifying... [id=i-0e
aws_instance.example: Still modifying... [id=i-0e
aws_instance.example: Modifications complete afte

Apply complete! Resources: 0 added, 1 changed, 0 
(base) → terraform-multiple-tfvars _
```

5. Test and Verify:

- Observe how different tfvars files are used to set
  environments during the apply process.
- Access the AWS Management Console or use the A
  of resources in the specified regions and instance typ

6. Clean Up:

- After testing, you can clean up resources:

```
terraform destroy -var-file=dev.tfvars
terraform destroy -var-file=prod.tfvars
```