

Pulkit

500125835

Lab Exercise 17– 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
}
```

The screenshot shows the VS Code interface with the file tree on the left and the code editor on the right. The file tree shows a folder named 'TERRAFORM-DEMO' containing 'Exercise-11', 'main.tf', 'instance.tf', 'dev.tvars', 'qa.tvars', 'prod.tvars', 'terraform.lock.hcl', 'terraform.state', 'terraform.state.backup', and 'variable.tf'. The 'main.tf' file is selected and open in the editor. The code in 'main.tf' is:

```
1 terraform {  
2   required_providers {  
3     aws = {  
4       source = "hashicorp/aws"  
5       version = "6.11.0"  
6     }  
7   }  
8 }  
9  
10 provider "aws" {  
11   # configuration options  
12   region = "ap-south-1"  
13   access_key = "AKIAQABHGOHNT56UWLNK"  
14   secret_key = "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"  
15 }
```

At the bottom of the screen, there are status bars for 'OUTLINE', 'TIMELINE', and 'variable.tf'. The bottom right corner shows file statistics: 'Ln 15, Col 2', 'Spaces: 2', 'UTF-8', 'CRLF', and a 'Terraform' icon.

- Create a file named variables.tf:

variables.tf

The screenshot shows the VS Code interface with the file tree on the left and the code editor on the right. The file tree shows the same directory structure as the previous screenshot. The 'variable.tf' file is selected and open in the editor. The code in 'variable.tf' is:

```
1 variable "my-ami" {  
2   type = string  
3   default = "ami-02d26659fd82cf299"  
4 }  
5  
6 variable "my-instance-type" {  
7   type = string  
8   default = "t3.micro"  
9 }
```

At the bottom of the screen, there are status bars for 'OUTLINE', 'TIMELINE', and 'variable.tf'. The bottom right corner shows file statistics: 'Ln 9, Col 2', 'Spaces: 4', 'UTF-8', 'CRLF', and a 'Terraform' icon.

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

```
type = string
```

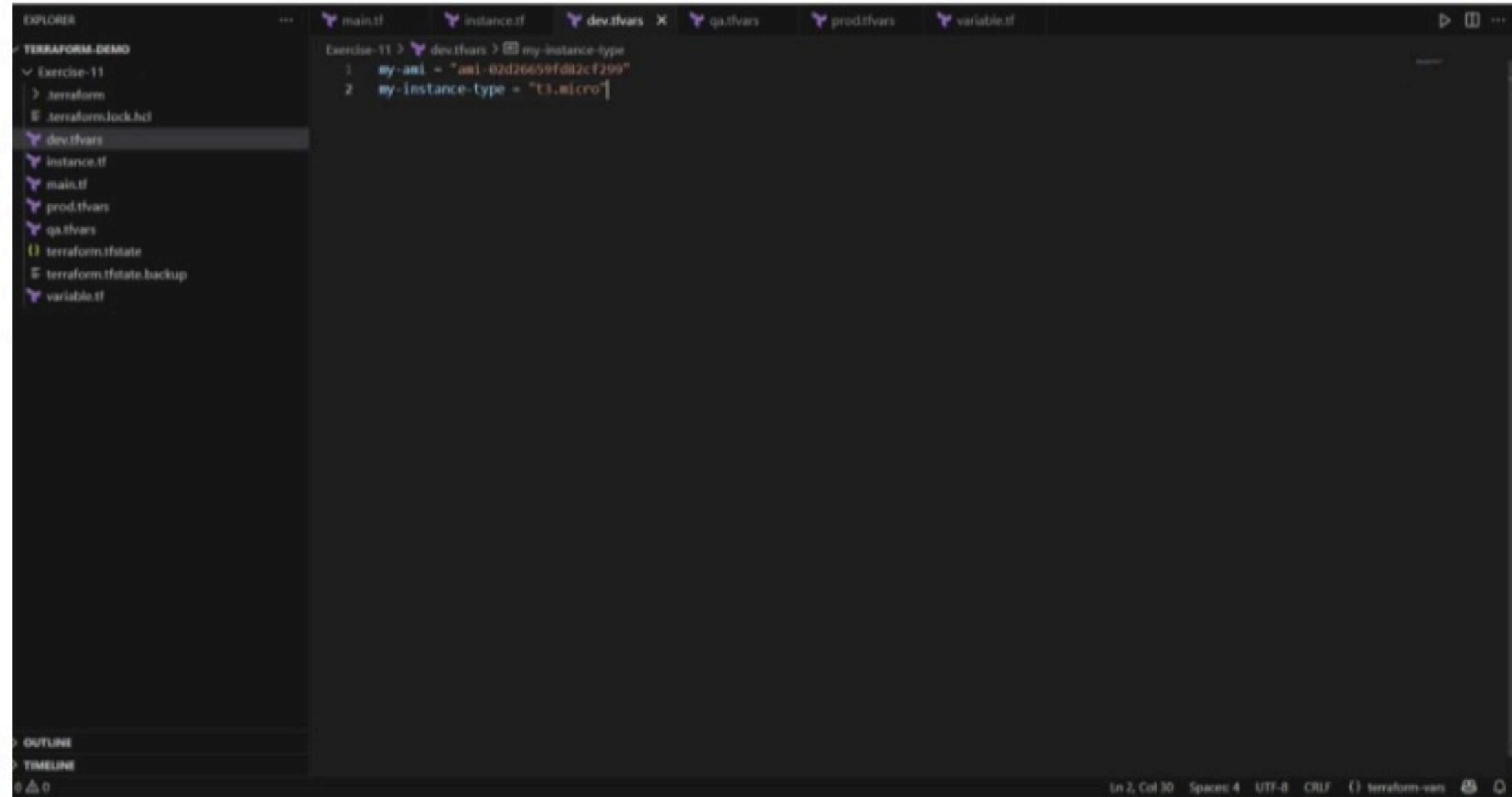
```
}
```

2. Create Multiple tfvars Files:

- Create a file named dev.tfvars:

```
# dev.tfvars
```

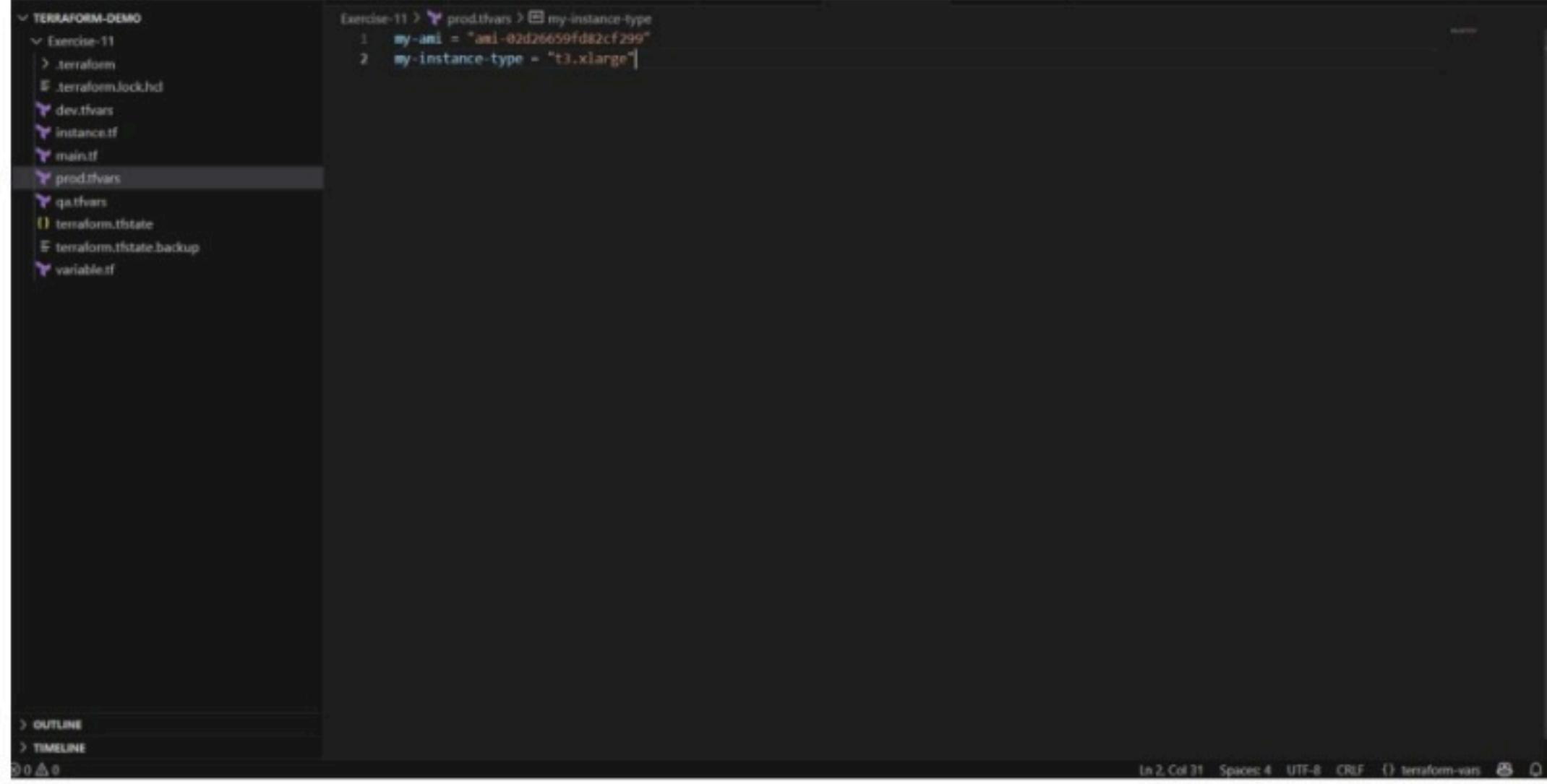
```
ami      = "ami-0123456789abcdef0"
instance_type = "t2.micro"
```



- Create a file named prod.tfvars:

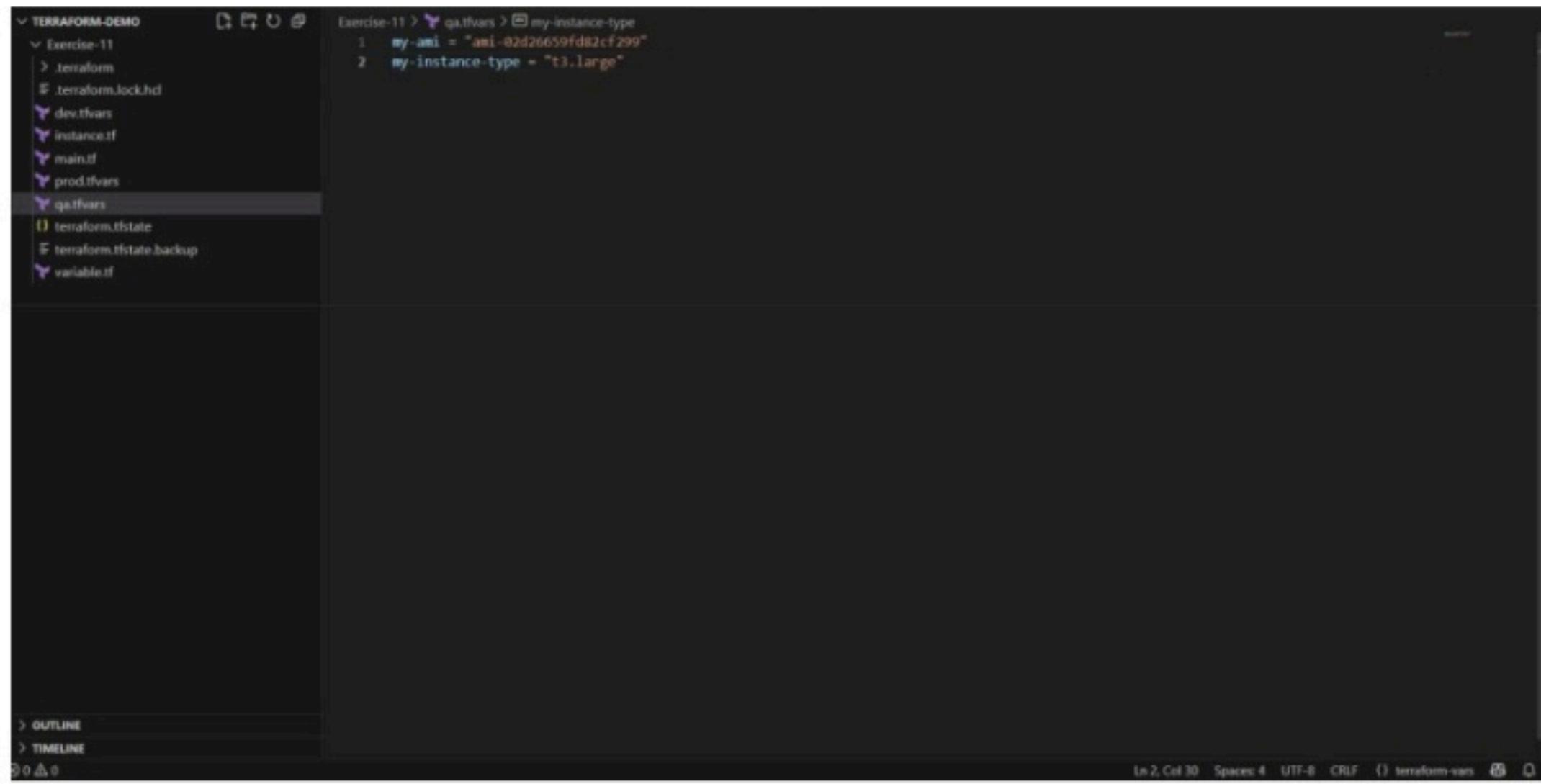
```
# prod.tfvars
```

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



```
Exercise-11 > prod.tfvars > my-instance-type
1 my-ami = "ami-02d26659fd82cf299"
2 my-instance-type = "t3.xlarge"
```

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



```
Exercise-11 > qa.tfvars > my-instance-type
1 my-ami = "ami-02d26659fd82cf299"
2 my-instance-type = "t3.large"
```

3. Initialize and Apply for Dev Environment:

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

```
C:\Me\DevSecOps\Terraform-demo\Exercise-11>terraform plan -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.web1 will be created
+ resource "aws_instance" "web1" {
    + ami                               = "ami-02d26659fd82cf299"
    + arn                               = (known after apply)
    + associate_public_ip_address      = (known after apply)
    + availability_zone                = (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)
    + force_destroy                     = false
    + 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                    = "t3.xlarge"
    + 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)
    + private_dns                       = (known after apply)
    + private_ip                        = (known after apply)
    + public_dns                        = (known after apply)
    + public_ip                         = (known after apply)
```

terraform init

terraform apply -var-file=dev.tfvars

4. Initialize and Apply for Prod Environment:

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

```
PS C:\Me\DevSecOps\Terraform-demo\Exercise-11> terraform validate
Success! The configuration is valid.
```

```
PS C:\Me\DevSecOps\Terraform-demo\Exercise-11>
```

```
+ primary_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.
```

Note: You didn't use the `-out` option to save this plan, so Terraform can't guarantee to take exactly these actions if you run `"terraform apply"` now.

```
C:\Me\DevSecOps\Terraform-demo\Exercise-11>
```

terraform init

terraform apply -var-file=prod.tfvars

```
C:\Me\DevSecOps\Terraform-demo\Exercise-11>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.web1 will be created
+ resource "aws_instance" "web1" {
    + ami
    + arn
    + associate_public_ip_address
    + availability_zone
    + disable_api_stop
    + disable_api_termination
    + ebs_optimized
    + enable_primary_ipv6
    + force_destroy
    + get_password_data
    + host_id
    + host_resource_group_arn
    + iam_instance_profile
    + id
    + instance_initiated_shutdown_behavior
    + instance_lifecycle
    + instance_state
    + instance_type
    + ipv6_address_count
    + ipv6_addresses
    + key_name
    + monitoring
    + outpost_arn
    + password_data
    + placement_group
    + placement_partition_number
    + primary_network_interface_id
    + private_dns
    + private_ip
}
```

```
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.web1: Creating...
aws_instance.web1: Still creating... [00m10s elapsed]
aws_instance.web1: Creation complete after 15s [id=i-07f793b2b55f78865]

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

C:\Me\DevSecOps\Terraform-demo\Exercise-11>
```

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.

The screenshot shows the AWS Management Console interface for the EC2 service. On the left, there's a navigation sidebar with options like Dashboard, AWS Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store (Volumes). The main content area is titled 'Instances (7) info'. It displays a table with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. The table lists seven instances, all of which are currently running (t3.micro). The last updated timestamp is 'less than a minute ago'. At the bottom of the table, there's a message 'Select an instance'.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
i-0445b7047df1209f	Running	t3.micro	3/3 checks passed	View alarms	ap-south-1a	ec2-13-232	
MY-EC2-INSTA...	Terminated	t3.micro	-	View alarms	ap-south-1a	-	
MY-EC2-INSTA...	Terminated	t3.micro	-	View alarms	ap-south-1a	-	
MY-EC2-INSTA...	Running	t3.micro	Initializing	View alarms	ap-south-1a	ec2-13-232	
MY-EC2-INSTA...	Terminated	t3.micro	-	View alarms	ap-south-1a	-	
MY-EC2-INSTA...	Terminated	t3.micro	-	View alarms	ap-south-1a	-	
MY-EC2-INSTA...	Terminated	t3.micro	-	View alarms	ap-south-1a	-	

6. Clean Up:

- After testing, you can clean up resources:

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

- Confirm the destruction by typing yes.

```
C:\Me\DevSecOps\Terraform-demo\Exercise-11>terraform destroy -var-file=dev.tfvars  
aws_instance.web1: Refreshing state... [id=i-07f793b2b55f78865]  
  
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.web1 will be destroyed  
- resource "aws_instance" "web1" {  
    - ami  
    - arn  
    - associate_public_ip_address  
    - availability_zone  
    - disable_api_stop  
    - disable_api_termination  
    - ebs_optimized  
    - force_destroy  
    - get_password_data  
    - hibernation  
    - id  
    - instance_initiated_shutdown_behavior  
    - instance_state  
    - instance_type  
    - ipv6_address_count  
    - ipv6_addresses  
    - monitoring  
    - placement_partition_number  
    - primary_network_interface_id  
    - private_dns  
    - private_ip  
    - public_dns  
        = "ami-02d26659fd82cf299" -> null  
        = "arn:aws:ec2:ap-south-1:060211753450:instance/i-07f793b2b55f78865" -> null  
        = "true" -> null  
        = "ap-south-1a" -> null  
        = "false" -> null  
        = "t2.micro" -> null  
        = "0" -> null  
        = "[]" -> null  
        = "false" -> null  
        = "0" -> null  
        = "eni-0d11f6a8b28bcb2d4" -> null  
        = "ip-172-31-37-238.ap-south-1.compute.internal" -> null  
        = "172.31.37.238" -> null  
        = "ec2-13-232-4-255.ap-south-1.compute.amazonaws.com" -> 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.web1: Destroying... [id=i-07f793b2b55f78865]  
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 00m10s elapsed]  
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 00m20s elapsed]  
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 00m30s elapsed]  
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 00m40s elapsed]  
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 00m50s elapsed]  
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 01m00s elapsed]  
aws_instance.web1: Destruction complete after 1m2s  
  
Destroy complete! Resources: 1 destroyed.  
C:\Me\DevSecOps\Terraform-demo\Exercise-11>
```

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.