# 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
}
```

• CREATE A FILE NAMED VARIABLES.TF:

## # VARIABLES.TF

```
VARIABLE "AMI" {
```

```
TYPE = STRING

VARIABLE "INSTANCE_TY" {

TYPE = STRING

}
```

```
var.tf > % variable "instance_type"

variable "ami" {

type = string

variable "instance_type" {

type = string

type = string

yariable "instance_type" {

variable "instance_type" {

variable
```

# 2. CREATE MULTIPLE TFVARS FILES:

• CREATE A FILE NAMED DEV.TFVARS:

# DEV.TFVARS

```
AMI = "AMI-0123456789ABCDEFO"

INSTANCE_TYPE = "T2.MICRO"
```

• CREATE A FILE NAMED PROD.TFVARS:

# PROD.TFVARS

```
AMI = "AMI-9876543210FEDCBAO"

INSTANCE_TYPE = "T2.LARGE"

dev.tfvars > @ instance_type

1   ami = "ami-02d26659fd82cf299"

2   instance_type = "t3.micro"
```

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

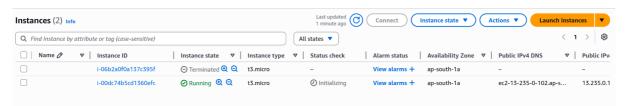
### 3. Initialize and Apply for Dev Environment:

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

```
TERRAFORM INIT
TERRAFORM APPLY -VAR-FILE=DEV.TFVARS
```

```
[reckless@Arch Lab 17]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.30.0
should now work.
rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.
[reckless@Arch Lab 17]$ 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.My-instance will be created
   + resource "aws_instance" "My-instance"
                                                    = "ami-02d26659fd82cf299"
      + ami
                                                    = (known after apply)
       + arn
                                                   = (known after apply)
= (known after apply)
       + associate_public_ip_address
       + availability_zone
                                                    = (known after apply)
= (known after apply)
= (known after apply)
       + cpu_core_count
       + cpu_threads_per_core
       + disable_api_stop
                                                   = (known after apply)
= (known after apply)
       + disable_api_termination
       + ebs_optimized
                                                    = false
       + get_password_data
                                                    = (known after apply)
       + host_id
       + host_resource_group_arn
                                                    = (known after apply)
                                                    = (known after apply)
= (known after apply)
       + iam_instance_profile
       + id
       + instance_initiated_shutdown_behavior = (known after apply)
+ instance_lifecycle = (known after apply)
                                                    = (known after apply)
= "t3.micro"
       + instance_state
       + instance_type
                                                    = (known after apply)
       + ipv6_address_count
         ipv6_addresses
                                                    = (known after apply)
       + key_name
                                                    = (known after apply)
                                                    = (known after apply)
       + monitoring
       + 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)
```

```
public_dns
                                                (known after apply)
                                              = (known after apply)
      + public_ip
      + secondary_private_ips
                                              = (known after apply)
                                              = (known after apply)
      + security_groups
                                              = true
      + source dest check
                                              = (known after apply)
      + spot_instance_request_id
      + subnet_id
                                              = (known after apply)
      + tags_all
                                              = (known after apply)
                                              = (known after apply)
      + tenancy
                                              = (known after apply)
      + user_data
      + user_data_base64
                                              = (known after apply)
      + user_data_replace_on_change
                                              = false
                                              = (known after apply)
      + vpc_security_group_ids
      + capacity_reservation_specification (known after apply)
      + cpu_options (known after apply)
      + ebs_block_device (known after apply)
      + enclave_options (known after apply)
      + ephemeral_block_device (known after apply)
      + instance_market_options (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.My-instance: Creating...
aws_instance.My-instance: Still creating... [00m10s elapsed]
aws_instance.My-instance: Creation complete after 14s [id=i-00dc74b5cd1360efc]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[reckless@Arch Lab 17]$
```

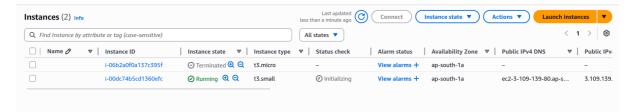


## 4. INITIALIZE AND APPLY FOR PROD ENVIRONMENT:

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

# TERRAFORM INIT TERRAFORM APPLY -VAR-FILE=PROD.TFVARS

```
[reckless@Arch Lab 17]$ terraform apply -var-file=prod.tfvars
aws_instance.My-instance: Refreshing state... [id=i-00dc74b5cd1360efc]
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.My-instance will be updated in-place
~ resource "aws_instance" "My-instance" {
                                                       = "i-00dc74b5cd1360efc"
                                                       = "t3.micro" -> "t3.small"
       instance_type
                                                       = {}
         tags
Plan: 0 to add, 1 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.My-instance: Modifying... [id=i-00dc74b5cd1360efc]
aws_instance.My-instance: Still modifying... [id=i-00dc74b5cd1360efc, 00m10s elapsed] aws_instance.My-instance: Still modifying... [id=i-00dc74b5cd1360efc, 00m20s elapsed] aws_instance.My-instance: Still modifying... [id=i-00dc74b5cd1360efc, 00m20s elapsed]
aws_instance.My-instance: Modifications complete after 34s [id=i-00dc74b5cd1360efc]
Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
[reckless@Arch Lab 17]$
```



## 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
TERRAFORM DESTROY -VAR-FILE=PROD.TFVARS
```

CONFIRM THE DESTRUCTION BY TYPING YES.

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

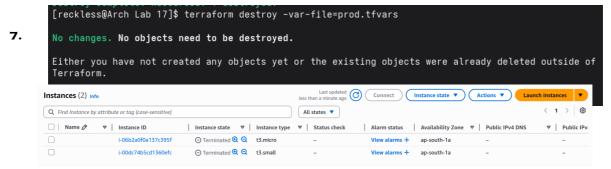
Do you really want to destroy all resources?

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.My-instance: Destroying... [id=i-00dc74b5cd1360efc]
aws_instance.My-instance: Still destroying... [id=i-00dc74b5cd1360efc, 00m10s elapsed]
aws_instance.My-instance: Still destroying... [id=i-00dc74b5cd1360efc, 00m20s elapsed]
aws_instance.My-instance: Still destroying... [id=i-00dc74b5cd1360efc, 00m30s elapsed]
aws_instance.My-instance: Destruction complete after 31s

Destroy complete! Resources: 1 destroyed.
```



### 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.