Lab Exercise 5– Terraform Variables with Command Line Arguments

Objective:

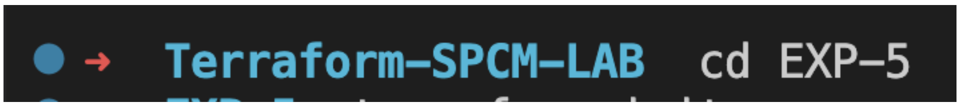
Learn how to pass values to Terraform variables using command line arguments.

Prerequisites:

• Terraform installed on your machine.
• Basic knowledge of Terraform variables.

Steps:

1. Create a Terraform Directory:

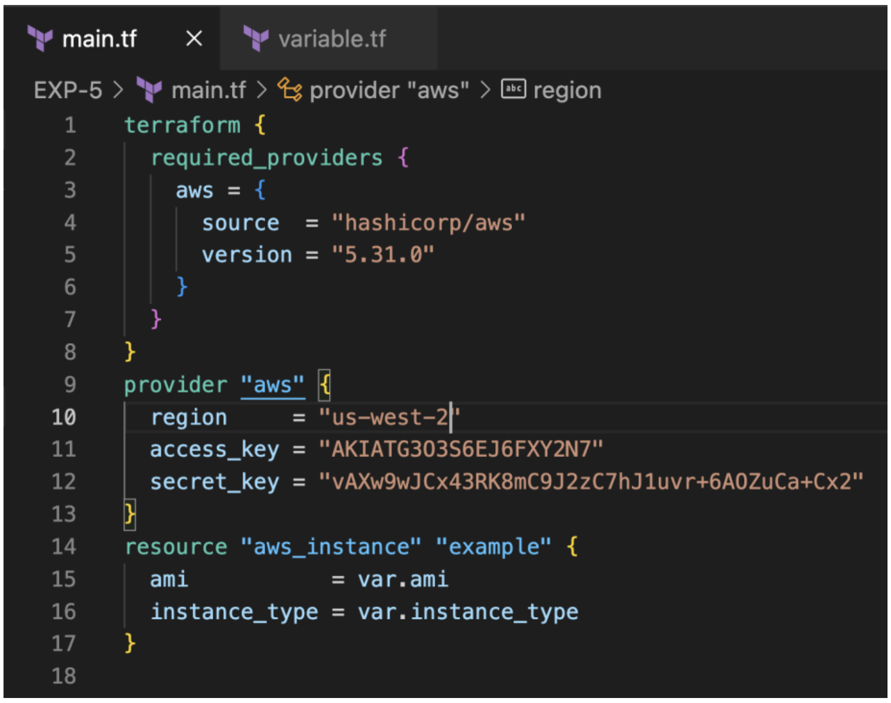


```
● ➜  Terraform-SPCM-LAB  cd EXP-5
```

2. Create Terraform Configuration Files:

• Create a file named main.tf:

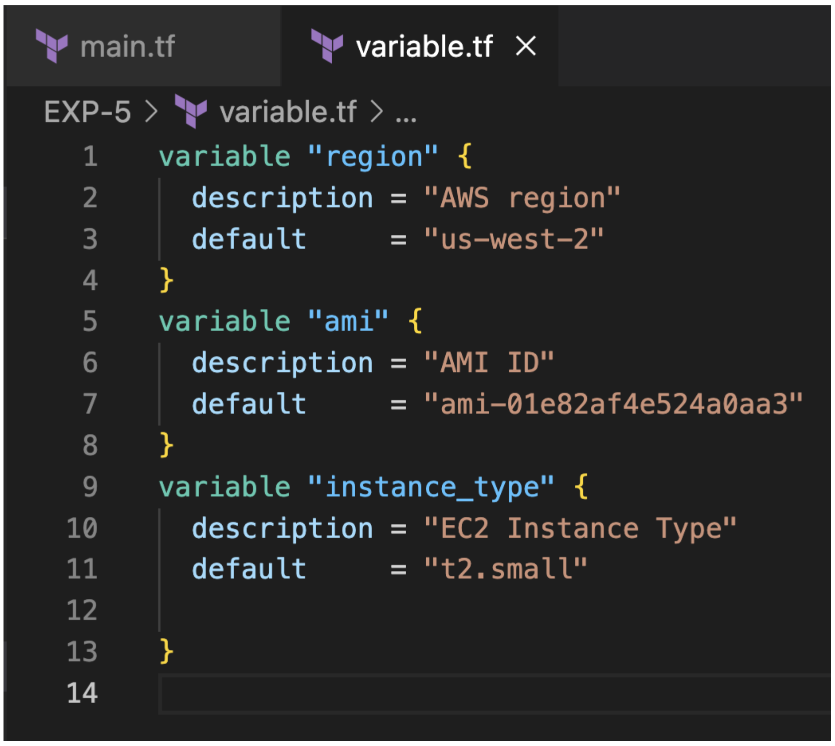
# main.tf



```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "5.31.0"
    }
  }
}
provider "aws" {
  region     = "us-west-2"
  access_key = "AKIATG303S6EJ6FXY2N7"
  secret_key = "vAXw9wJCx43RK8mC9J2zC7hJ1uvr+6A0ZuCa+Cx2"
}
resource "aws_instance" "example" {
  ami           = var.ami
  instance_type = var.instance_type
}
```

• Create a file named variables.tf:

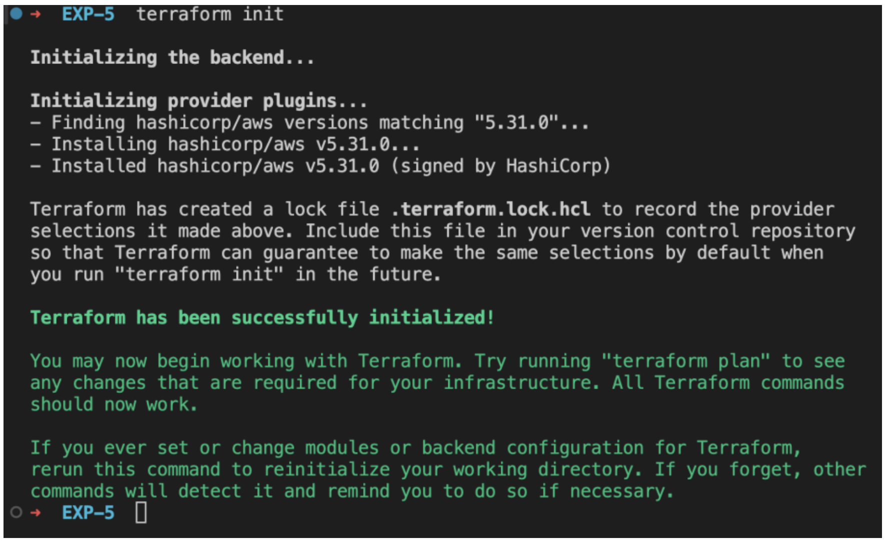
# variables.tf



```
variable "region" {
  description = "AWS region"
  default     = "us-west-2"
}
variable "ami" {
  description = "AMI ID"
  default     = "ami-01e82af4e524a0aa3"
}
variable "instance_type" {
  description = "EC2 Instance Type"
  default     = "t2.small"

}

```

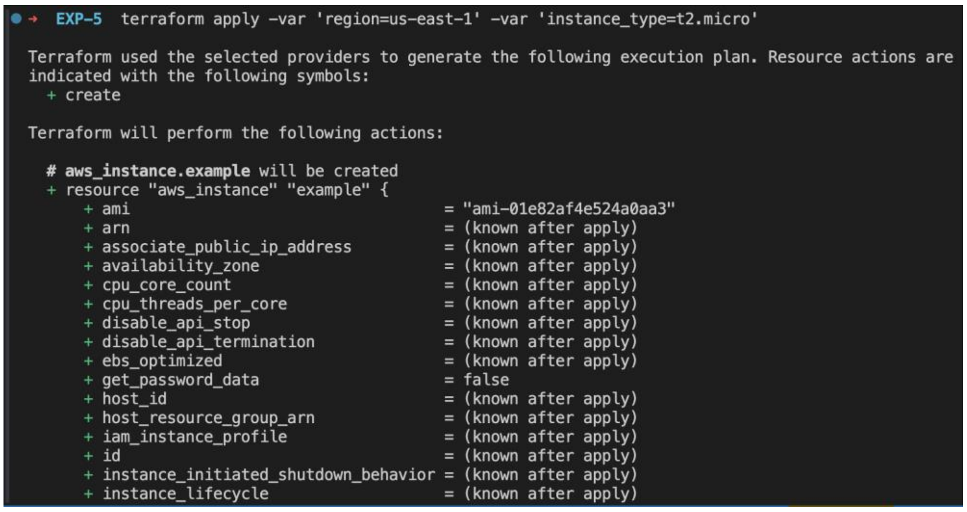
3. Use Command Line Arguments:

- Open a terminal and navigate to your Terraform project directory.
- Run the terraform init command:

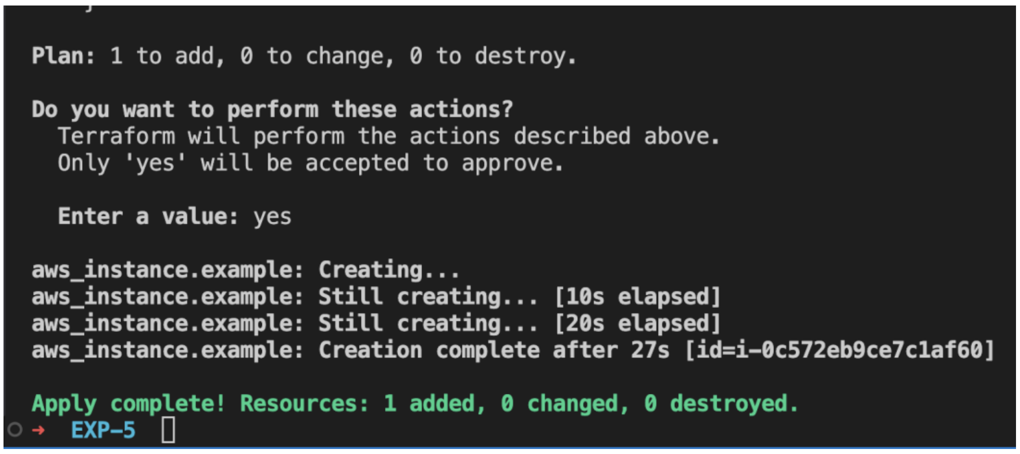


```
● ➜  EXP-5  terraform init

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.
○ ➜  EXP-5  ▯
```

- Run the terraform apply command with command line arguments to set variable values:



```
● → EXP-5 terraform apply -var 'region=us-east-1' -var 'instance_type=t2.micro'

  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-01e82af4e524a0aa3"
        + 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)
        + instance_initiated_shutdown_behavior = (known after apply)
        + instance_lifecycle                   = (known after apply)
```

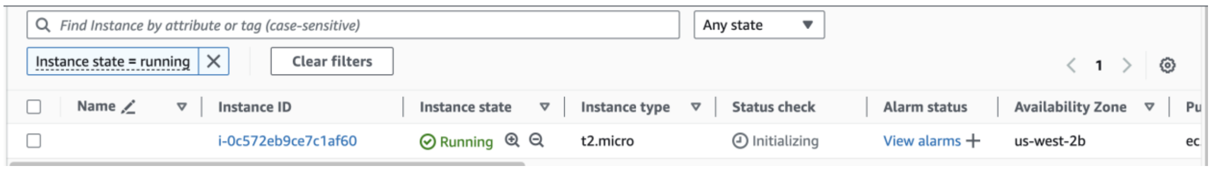


```
Plan: 1 to add, 0 to change, 0 to destroy.

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: Creation complete after 27s [id=i-0c572eb9ce7c1af60]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
○ → EXP-5 
```

4. Test and Verify:

• Observe how the command line arguments dynamically set the variable values during the apply process.

• Access the AWS Management Console or use the AWS CLI to verify the creation of resources in the specified region.



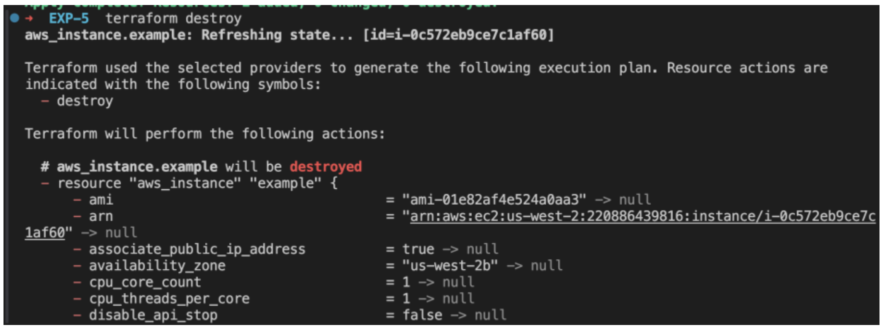
Find Instance by attribute or tag (case-sensitive) | Any state

Instance state = running | Clear filters

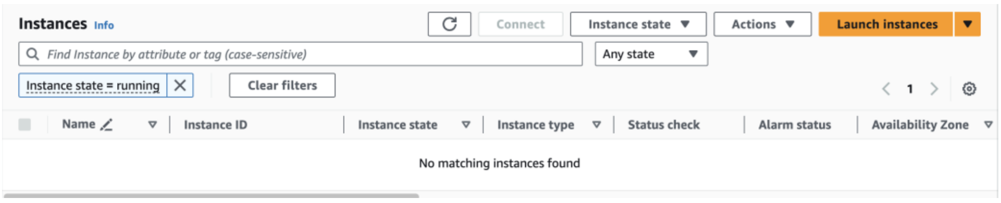
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Pu
☐		i-0c572eb9ce7c1af60	Running	t2.micro	Initializing	View alarms +	us-west-2b	ec

5. Clean Up:

After testing, you can clean up resources:



```
● ➜  EXP-5 terraform destroy
aws_instance.example: Refreshing state... [id=i-0c572eb9ce7c1af60]

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.example will be destroyed
  - resource "aws_instance" "example" {
      - ami                                  = "ami-01e82af4e524a0aa3" -> null
      - arn                                  = "arn:aws:ec2:us-west-2:220886439816:instance/i-0c572eb9ce7c
1af60" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "us-west-2b" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
```



Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone

No matching instances found