

**School of Computer Science**  
**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**  
**DEHRADUN, UTTARAKHAND**



**System Provisioning and  
Configuration Management**

**Lab File (2022-2026)**  
**6<sup>th</sup> Semester**

*Submitted To:*

***Dr. Hitesh Kumar***  
***Sharma***

*Submitted By:*

***Akshat Pandey***  
***(500101788)***  
***B Tech CSE***  
***DevOps[6<sup>th</sup> Semester]***  
***R2142220306***  
***Batch - 1***

## EXPERIMENT 7

### Lab Exercise: 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:

```
mkdir terraform-cli-variables
```

```
cd terraform-cli-variables
```

```
C:\Users\aksha\Documents>mkdir terraform-cli-variables  
C:\Users\aksha\Documents>cd terraform-cli-variables  
C:\Users\aksha\Documents\terraform-cli-variables>|
```

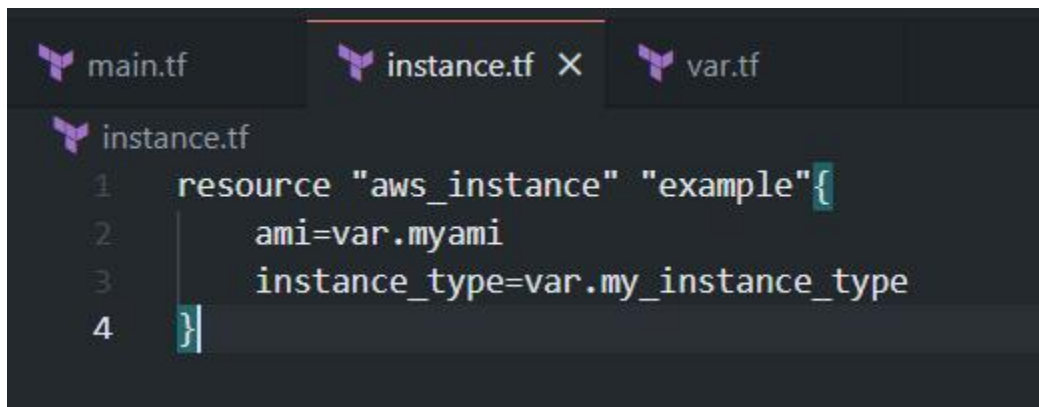
##### 2. Create Terraform Configuration Files:

- Create a file named main.tf:

```
# instance.tf
```

```
resource "aws_instance" "example" {  
  ami          = var.ami  
  instance_type = var.instance_type
```

```
}
```



```
main.tf  instance.tf X  var.tf

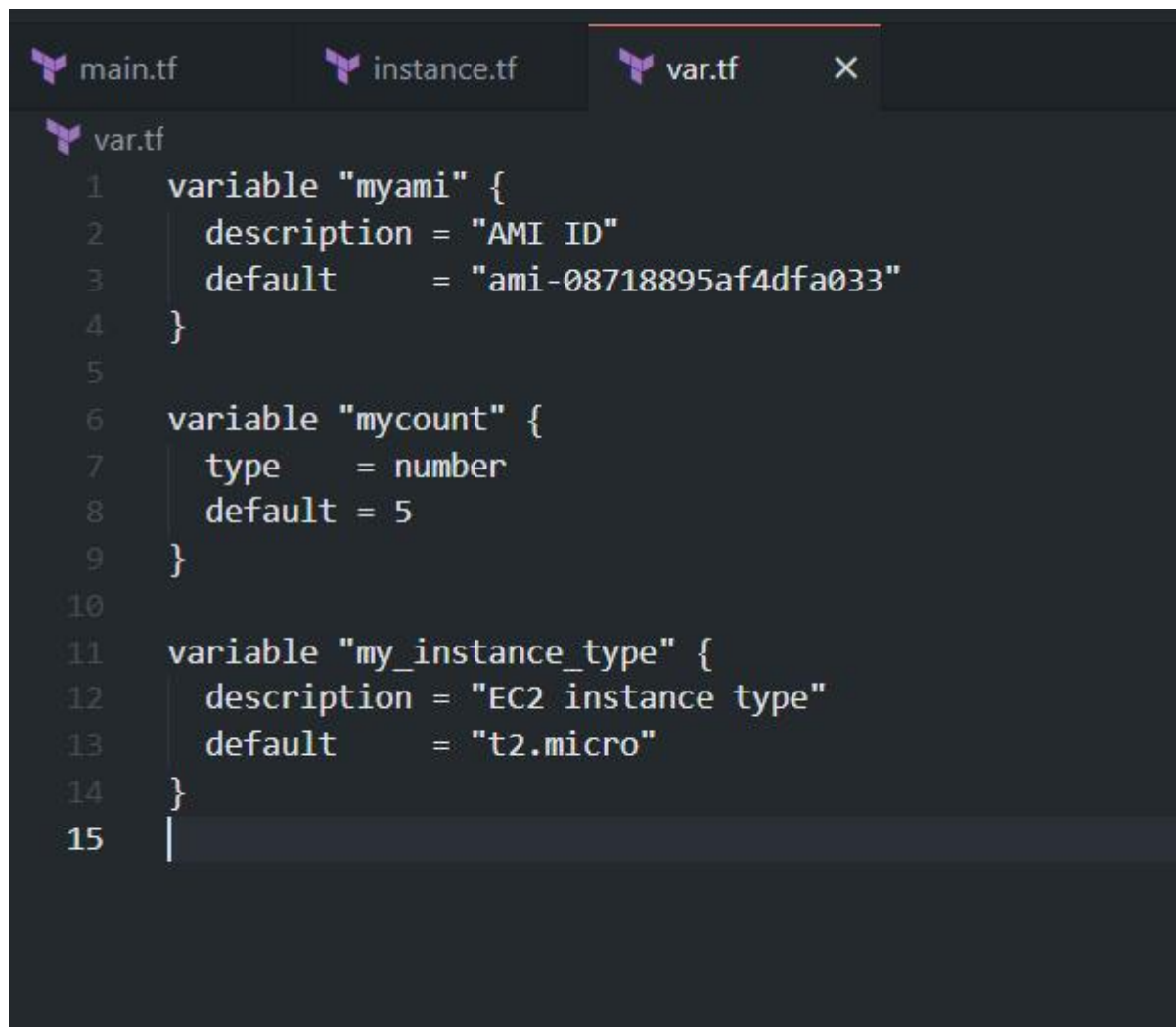
instance.tf
1  resource "aws_instance" "example" {
2      ami=var.myami
3      instance_type=var.my_instance_type
4  }
```

- Create a file named variables.tf:

```
# variables.tf
```

```
variable "ami" {
  description = "AMI ID"
  default    = "ami-08718895af4dfa033"
}

variable "instance_type" {
  description = "EC2 Instance Type"
  default    = "t2.micro"
}
```



The screenshot shows a code editor with three tabs: main.tf, instance.tf, and var.tf. The var.tf tab is active, displaying the following Terraform code:

```
1 variable "myami" {
2     description = "AMI ID"
3     default     = "ami-08718895af4dfa033"
4 }
5
6 variable "mycount" {
7     type      = number
8     default   = 5
9 }
10
11 variable "my_instance_type" {
12     description = "EC2 instance type"
13     default     = "t2.micro"
14 }
15 |
```

### 3. Use Command Line Arguments:

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

#### terraform init

```
C:\Users\aksha\Documents\terraform-cli-variables>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.84.0

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.
```

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

```
terraform plan -var="ami=ami-0522ab6e1ddcc7055" -var="instance_type=t3.micro"
```

- Adjust the values based on your preferences.

```
C:\Users\aksha\Documents\terraform-cli-variables>terraform plan -var="myami=ami-0522ab6e1ddcc7055" -var="my_instance_type=t3.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-0522ab6e1ddcc7055"
  + 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)
  + enable_primary_ipv6    = (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)
```

Only 'yes' will be accepted to approve.

Enter a value: yes

```
aws_instance.my_instance-1[2]: Creating...
aws_instance.my_instance-1[3]: Creating...
aws_instance.my_instance-1[0]: Creating...
aws_instance.my_instance-1[1]: Creating...
aws_instance.example: Creating...
aws_instance.my_instance-1[4]: Creating...
aws_instance.my_instance-1[3]: Still creating... [10s elapsed]
aws_instance.my_instance-1[2]: Still creating... [10s elapsed]
aws_instance.my_instance-1[1]: Still creating... [10s elapsed]
aws_instance.my_instance-1[4]: Still creating... [10s elapsed]
aws_instance.my_instance-1[0]: Still creating... [10s elapsed]
aws_instance.example: Still creating... [10s elapsed]
aws_instance.my_instance-1[2]: Still creating... [20s elapsed]
aws_instance.my_instance-1[3]: Still creating... [20s elapsed]
aws_instance.my_instance-1[4]: Still creating... [20s elapsed]
aws_instance.example: Still creating... [20s elapsed]
aws_instance.my_instance-1[1]: Still creating... [20s elapsed]
aws_instance.my_instance-1[0]: Still creating... [20s elapsed]
aws_instance.my_instance-1[0]: Creation complete after 22s [id=i-09be06319baec87b3]
aws_instance.my_instance-1[4]: Still creating... [30s elapsed]
aws_instance.my_instance-1[2]: Still creating... [30s elapsed]
aws_instance.my_instance-1[3]: Still creating... [30s elapsed]
aws_instance.my_instance-1[1]: Still creating... [30s elapsed]
aws_instance.example: Still creating... [30s elapsed]
aws_instance.my_instance-1[3]: Creation complete after 32s [id=i-0d7fe9b426a912a9f]
aws_instance.my_instance-1[4]: Creation complete after 32s [id=i-09e545a5d61cdbe9c]
aws_instance.my_instance-1[1]: Creation complete after 32s [id=i-0ac6d23a64cab202f]
aws_instance.my_instance-1[2]: Creation complete after 32s [id=i-0ffaea172393f53da]
aws_instance.example: Creation complete after 32s [id=i-0ca49771ee8b98b4c]
```

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

```
C:\Users\aksha\Documents\terraform-cli-variables>
```

## 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.

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
<input type="checkbox"/>		<a href="#">i-0ca49771ee8b98b4c</a>	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	ap-south-1b	ec2-13-2
<input type="checkbox"/>	my_instance	<a href="#">i-0d7fe9b426a912a9f</a>	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	ap-south-1b	ec2-43-2
<input type="checkbox"/>	my_instance	<a href="#">i-09be06319baec87b3</a>	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	ap-south-1b	ec2-15-2
<input type="checkbox"/>	my_instance	<a href="#">i-0ac6d23a64cab202f</a>	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	ap-south-1b	ec2-3-11
<input type="checkbox"/>	my_instance	<a href="#">i-0ffaea172393f53da</a>	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	ap-south-1b	ec2-65-2

=

## 5. Clean Up:

After testing, you can clean up resources:

```
terraform destroy
```

Confirm the destruction by typing yes.



```
C:\Users\aksha\Documents\terraform-cli-variables>terraform destroy
aws_instance.example: Refreshing state... [id=i-0ca49771ee8b98b4c]
aws_instance.my_instance-1[1]: Refreshing state... [id=i-0ac6d23a64cab202f]
aws_instance.my_instance-1[3]: Refreshing state... [id=i-0d7fe9b426a912a9f]
aws_instance.my_instance-1[2]: Refreshing state... [id=i-0ffaea172393f53da]
aws_instance.my_instance-1[0]: Refreshing state... [id=i-09be06319baec87b3]
aws_instance.my_instance-1[4]: Refreshing state... [id=i-09e545a5d61cdbe9c]
```

Terraform used the selected providers to generate the following execution plan. Resource symbols:

- destroy

Terraform will perform the following actions:

```
# aws_instance.example will be destroyed
- resource "aws_instance" "example" {
  - ami                        = "ami-08718895af4dfa033" -> null
  - arn                      = "arn:aws:ec2:ap-south-1:730335668486:i
  - associate_public_ip_address = true -> null
  - availability_zone          = "ap-south-1b" -> null
  - cpu_core_count             = 1 -> null
  - cpu_threads_per_core       = 1 -> null
  - disable_api_stop           = false -> null
  - disable_api_termination    = false -> null
  - ebs_optimized              = false -> null
  - get_password_data          = false -> null
  - hibernation                = false -> null
  - id                        = "i-0ca49771ee8b98b4c" -> null
  - instance_initiated_shutdown_behavior = "stop" -> null
  - instance_state             = "running" -> null
  - instance_type              = "t2.micro" -> null
  - ipv6_address_count         = 0 -> null
  - ipv6_addresses             = [] -> null
```

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-1[4]: Destroying... [id=i-09e545a5d61cdbe9c]
aws_instance.my_instance-1[3]: Destroying... [id=i-0d7fe9b426a912a9f]
aws_instance.my_instance-1[2]: Destroying... [id=i-0fffaea172393f53da]
aws_instance.my_instance-1[1]: Destroying... [id=i-0ac6d23a64cab202f]
aws_instance.example: Destroying... [id=i-0ca49771ee8b98b4c]
aws_instance.my_instance-1[0]: Destroying... [id=i-09be06319baec87b3]
aws_instance.my_instance-1[3]: Still destroying... [id=i-0d7fe9b426a912a9f, 10s elapsed]
aws_instance.my_instance-1[4]: Still destroying... [id=i-09e545a5d61cdbe9c, 10s elapsed]
aws_instance.my_instance-1[0]: Still destroying... [id=i-09be06319baec87b3, 10s elapsed]
aws_instance.my_instance-1[2]: Still destroying... [id=i-0fffaea172393f53da, 10s elapsed]
aws_instance.my_instance-1[1]: Still destroying... [id=i-0ac6d23a64cab202f, 10s elapsed]
aws_instance.example: Still destroying... [id=i-0ca49771ee8b98b4c, 10s elapsed]
aws_instance.my_instance-1[4]: Still destroying... [id=i-09e545a5d61cdbe9c, 20s elapsed]
aws_instance.my_instance-1[3]: Still destroying... [id=i-0d7fe9b426a912a9f, 20s elapsed]
aws_instance.my_instance-1[0]: Still destroying... [id=i-09be06319baec87b3, 20s elapsed]
aws_instance.my_instance-1[1]: Still destroying... [id=i-0ac6d23a64cab202f, 20s elapsed]
aws_instance.example: Still destroying... [id=i-0ca49771ee8b98b4c, 20s elapsed]
aws_instance.my_instance-1[2]: Still destroying... [id=i-0fffaea172393f53da, 20s elapsed]
aws_instance.my_instance-1[2]: Still destroying... [id=i-0fffaea172393f53da, 30s elapsed]
aws_instance.example: Still destroying... [id=i-0ca49771ee8b98b4c, 30s elapsed]
aws_instance.my_instance-1[3]: Still destroying... [id=i-0d7fe9b426a912a9f, 30s elapsed]
aws_instance.my_instance-1[4]: Still destroying... [id=i-09e545a5d61cdbe9c, 30s elapsed]
aws_instance.my_instance-1[1]: Still destroying... [id=i-0ac6d23a64cab202f, 30s elapsed]
aws_instance.my_instance-1[0]: Still destroying... [id=i-09be06319baec87b3, 30s elapsed]
aws_instance.my_instance-1[3]: Still destroying... [id=i-0d7fe9b426a912a9f, 40s elapsed]
aws_instance.my_instance-1[1]: Still destroying... [id=i-0ac6d23a64cab202f, 40s elapsed]
aws_instance.example: Still destroying... [id=i-0ca49771ee8b98b4c, 40s elapsed]
aws_instance.my_instance-1[2]: Still destroying... [id=i-0fffaea172393f53da, 40s elapsed]
aws_instance.my_instance-1[4]: Still destroying... [id=i-09e545a5d61cdbe9c, 40s elapsed]
aws_instance.my_instance-1[0]: Still destroying... [id=i-09be06319baec87b3, 40s elapsed]
aws_instance.my_instance-1[2]: Destruction complete after 41s
aws_instance.example: Destruction complete after 41s
aws_instance.my_instance-1[0]: Destruction complete after 41s
aws_instance.my_instance-1[3]: Destruction complete after 41s
aws_instance.my_instance-1[4]: Destruction complete after 41s
aws_instance.my_instance-1[1]: Destruction complete after 41s
```

## 6. Conclusion:

This lab exercise demonstrates how to use command line arguments to set variable values dynamically during the terraform apply process. It allows you to customize your Terraform deployments without modifying the configuration files directly. Experiment with different variable values and observe how command line arguments impact the infrastructure provisioning process.