

System Provisioning & Configuration
Management
Lab File

Submitted By
Keshav Bhardwaj
SAPID: 500094898

Enrollment No: R2142210413
Batch: 03
Semester VI
BTech CSE DevOps
Submitted To

Dr. Hitesh Kumar Sharma



SCHOOL OF COMPUTER SCIENCE

UNIVERSITY OF PETROLEUM & ENERGY STUDIES

Dehradun-248007

2023-24

Experiment 1

Install & Setup Terraform

1. Ensure that your system is up to date and you have installed the gnupg, software-properties-common, and curl packages installed.

```
on linux_ubuntu
→ ~ sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
0% [Connecting to ppa.launchpadcontent.net] [Connecting to download.docker.com (108.158.245.1
```

2. Install the HashiCorp GPG key.

```
→ ~ wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg

--2024-01-17 10:08:29-- https://apt.releases.hashicorp.com/gpg
Resolving apt.releases.hashicorp.com (apt.releases.hashicorp.com)... 260
Connecting to apt.releases.hashicorp.com (apt.releases.hashicorp.com)|260
```

3. Verify the key's fingerprint

```
→ ~ gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint
```

4. Add the official HashiCorp repository to your system.

```
→ ~ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/hashicorp.list
```

5. Download the package information from HashiCorp.

```
→ ~ sudo apt update
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 http://packages.microsoft.com/repos/code stable InRelease
Hit:3 https://apt.releases.hashicorp.com jammy InRelease
Ign:4 https://pkg.jenkins.io/debian binary/ InRelease
Hit:5 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:6 https://pkg.jenkins.io/debian binary/ Release
```

6. Install Terraform from the new repository.

```
→ ~ sudo apt-get install terraform
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
terraform is already the newest version (1.6.6-1).
0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
→ ~
```

7. Verify that the installation worked by opening a new terminal session

```
→ ~ terraform --version
Terraform v1.6.6
on linux_amd64
→ ~
```

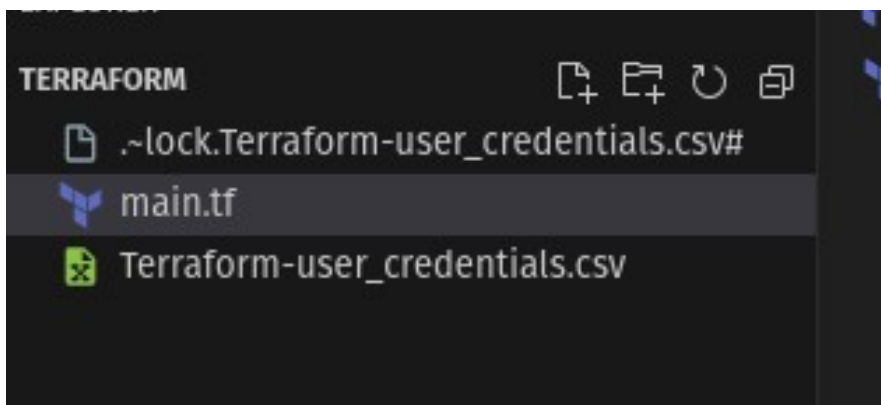
Experiment 2

Terraform AWS Provider and IAM User Settings

Prerequisites: Terraform Installed: Make sure you have Terraform installed on your machine. Follow the official installation guide if needed.

AWS Credentials: Ensure you have AWS credentials (Access Key ID and Secret Access Key) configured. You can set them up using the AWS CLI or by setting environment variables.

Step 1. Create a directory named Terraform and make a main.tf file in it.



Step 2 . After creating the main.tf file , add the following content into it.

```
main.tf x
main.tf > provider "aws"
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "5.32.1"
6     }
7   }
8 }
9
10 provider "aws" {
11   region = "ap-south-1"
12   access_key = "AKIA232UVZYDMA5SK35U"
13   secret_key = "iufuemcSo7Ght329ltTnuJfhWGEojpDDVvKxfxhLF"
14 }
15 }
```

- This script defines an AWS provider and provisions an EC2 instance.

Step 3. Initialize Terraform

```
→ TERRAFORM terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.32.1"...
- Installing hashicorp/aws v5.32.1...
- Installed hashicorp/aws v5.32.1 (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.
```

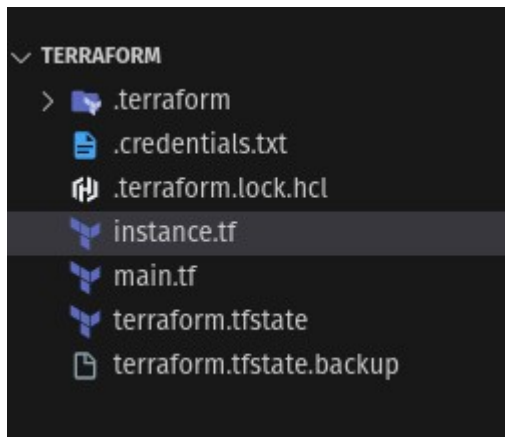
Experiment 3

Provisioning on EC2 Instance on AWS

Prerequisites:

Terraform Installed & AWS Credentials

Step 1. Create a terraform configuration file for EC2 Instance (instance.tf)



```
instance.tf > resource "aws_instance" "terraform"
1  resource "aws_instance" "terraform" {
2      instance_type = "t2.micro"
3      ami = "ami-03f4878755434977f"
4      tags = {
5          name = "Terraform"
6      }
7
8  }
```

Step 2. Validate the configuration

```
→ TERRAFORM terraform validate
Success! The configuration is valid.
```

Step 3. Review Plan

```
+ TERRAFORM terraform plan

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.terraform will be created
+ resource "aws_instance" "terraform" {
  + ami                  = "ami-03f4878755434977f"
  + 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)
```

Step 4. Terraform 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.terraform: Creating...
aws_instance.terraform: Still creating... [10s elapsed]
aws_instance.terraform: Still creating... [20s elapsed]
aws_instance.terraform: Still creating... [30s elapsed]
aws_instance.terraform: Creation complete after 34s [id=i-0ac545e0c69e7fb6a]
```

Step 5. Verifying resources:

Checking whether an EC2 instance is created in AWS console or not.

Find Instance by attribute or tag (case-sensitive)			
<input type="checkbox"/>	Name	Instance ID	Instance state
<input type="checkbox"/>	Terraform	i-0ac545e0c69e7fb6a	Running

Step 6. Cleaning Resources

```
➤ → TERRAFORM terraform destroy
aws_instance.terraform: Refreshing state... [id=i-0ac545e0c69e7fb6a]

Terraform used the selected providers to generate the following execution plan
- destroy

Terraform will perform the following actions:

# aws_instance.terraform will be destroyed
- resource "aws_instance" "terraform" {
  - ami                        = "ami-03f4878755434977f" -> null
  - arn                       = "arn:aws:ec2:ap-south-1:746966" -> null
  - associate_public_ip_address = true -> null
  - availability_zone         = "ap-south-1a" -> null
  - cpu_core_count            = 1 -> null
  - cpu_threads_per_core      = 1 -> null
  - disable_api_stop          = false -> null
  - disable_api_termination   = false -> null
```

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state
Terraform	i-0ac545e0c69e7fb6a	Shutting-d...

```
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.terraform: Destroying... [id=i-0ac545e0c69e7fb6a]
aws_instance.terraform: Still destroying... [id=i-0ac545e0c69e7fb6a, 10s elapsed]
aws_instance.terraform: Still destroying... [id=i-0ac545e0c69e7fb6a, 20s elapsed]
aws_instance.terraform: Still destroying... [id=i-0ac545e0c69e7fb6a, 30s elapsed]
aws_instance.terraform: Destruction complete after 32s

Destroy complete! Resources: 1 destroyed.
```

instances (1) Info

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state
Terraform	i-0ac545e0c69e7fb6a	Terminated

Terminated

Instance is destroyed successfully

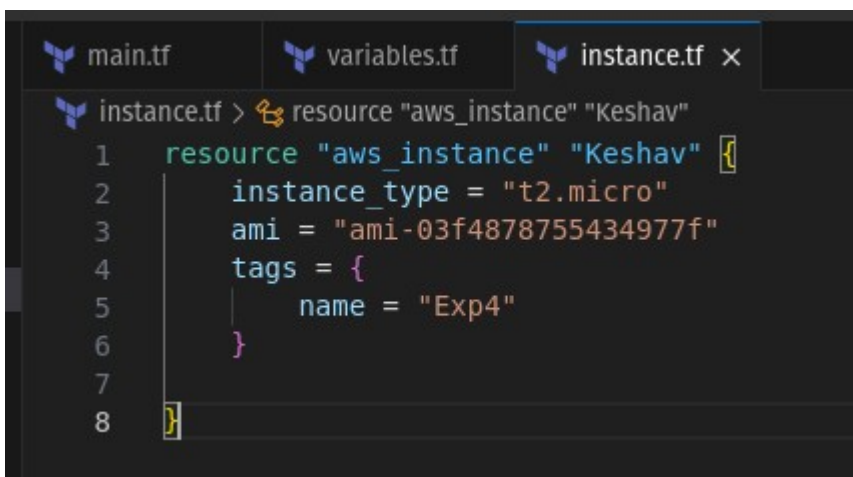
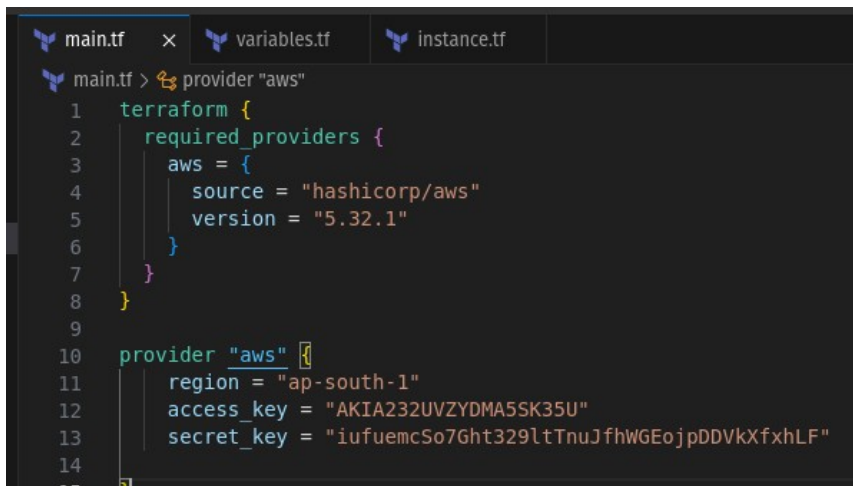
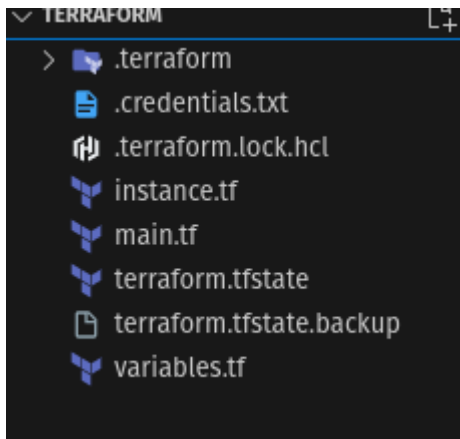
Experiment 4

Terraform Variables

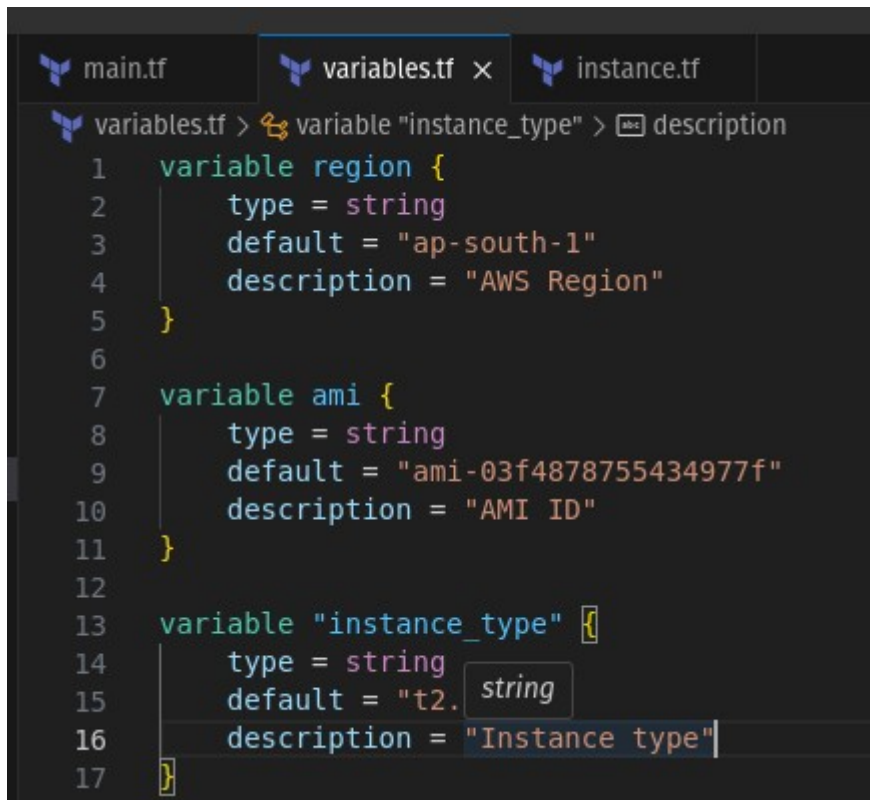
Aim

Learn how to define and use variables in Terraform configuration

Step 1. Create a main file & terraform configuration file for EC2 Instance (instance.tf)

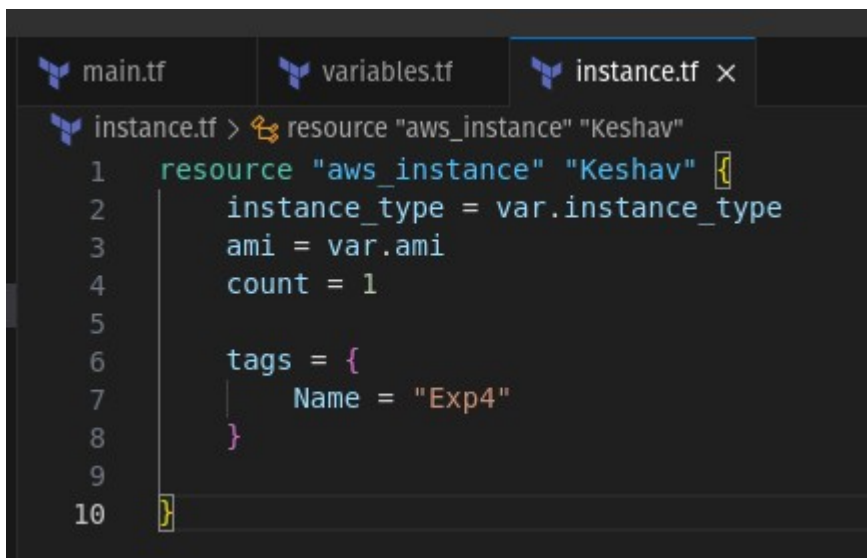


Step 2. Open a new file named variables.tf. Define variables for region, ami, secret_key, access_key and instance_type.



```
main.tf  variables.tf x  instance.tf
variables.tf > variable "instance_type" > description
1  variable region {
2      type = string
3      default = "ap-south-1"
4      description = "AWS Region"
5  }
6
7  variable ami {
8      type = string
9      default = "ami-03f4878755434977f"
10     description = "AMI ID"
11 }
12
13 variable "instance_type" {
14     type = string
15     default = "t2."
16     description = "Instance type"
17 }
```

Step 3. modify main.tf and instance.tf to use the variables.



```
main.tf  variables.tf  instance.tf x
instance.tf > resource "aws_instance" "Keshav"
1  resource "aws_instance" "Keshav" {
2      instance_type = var.instance_type
3      ami = var.ami
4      count = 1
5
6      tags = {
7          Name = "Exp4"
8      }
9
10 }
```

```
main.tf × variables.tf instance.tf
main.tf > provider "aws" > secret_key
1 terraform {
2     required_providers {
3         aws = {
4             source = "hashicorp/aws"
5             version = "5.32.1"
6         }
7     }
8 }
9
10 provider "aws" {
11     region = var.region
12     access_key = var.access_key
13     secret_key = var.secret_key
14 }
```

Step 4. Run the following Terraform commands to initialize and apply the configuration.

```
+ public_dns           = (known after apply)
+ public_ip            = (known after apply)
+ secondary_private_ips = (known after apply)
+ security_groups       = (known after apply)
+ source_dest_check     = true
+ spot_instance_request_id = (known after apply)
+ subnet_id            = (known after apply)
+ tags                 = {
+   + "Name" = "Exp4"
+ }
+ tags_all              = {
+   + "Name" = "Exp4"
+ }
+ tenancy               = (known after apply)
+ user_data             = (known after apply)
+ user_data_base64      = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids = (known after apply)
}
```

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

```
+ tenancy               = (known after apply)
+ user_data             = (known after apply)
+ user_data_base64      = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids = (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.Keshav[0]: Creating...
aws_instance.Keshav[0]: Still creating... [10s elapsed]
aws_instance.Keshav[0]: Still creating... [20s elapsed]
aws_instance.Keshav[0]: Still creating... [31s elapsed]
aws_instance.Keshav[0]: Creation complete after 36s [id=i-0dfe025c9691561dc]

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

Step 5 . Verifying Resources

Find instance by attribute or tag (case-sensitive)			
<input type="checkbox"/>	Name	Instance ID	Instance state
<input type="checkbox"/>	Exp4	i-0dfe025c9691561dc	Running

Step 6. Cleanup Resources

```
    - throughput      = 0 -> null
    - volume_id       = "vol-0c7d2df7d240732da" -> null
    - volume_size     = 8 -> null
    - volume_type     = "gp2" -> 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.Keshav[0]: Destroying... [id=i-0dfe025c9691561dc]
aws_instance.Keshav[0]: Still destroying... [id=i-0dfe025c9691561dc, 10s elapsed]
aws_instance.Keshav[0]: Still destroying... [id=i-0dfe025c9691561dc, 20s elapsed]
aws_instance.Keshav[0]: Still destroying... [id=i-0dfe025c9691561dc, 30s elapsed]
aws_instance.Keshav[0]: Destruction complete after 31s
```

<input type="checkbox"/>	Name	Instance ID	Instance state
<input type="checkbox"/>	Exp4	i-0dfe025c9691561dc	Terminated

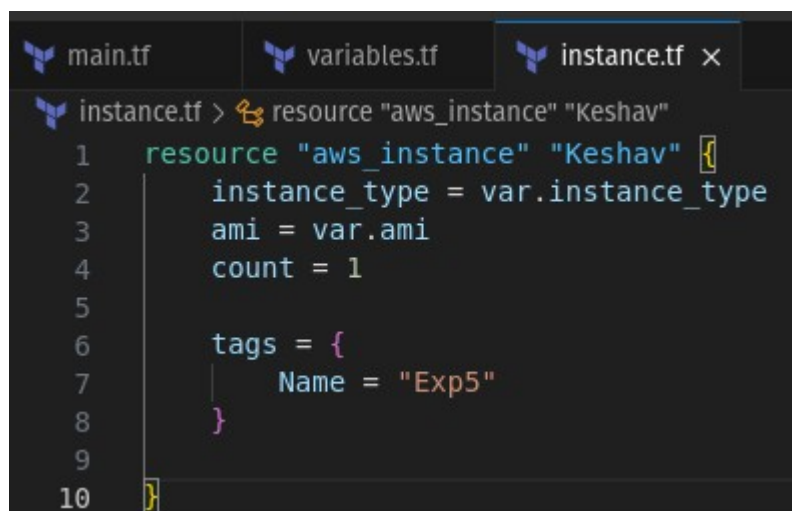
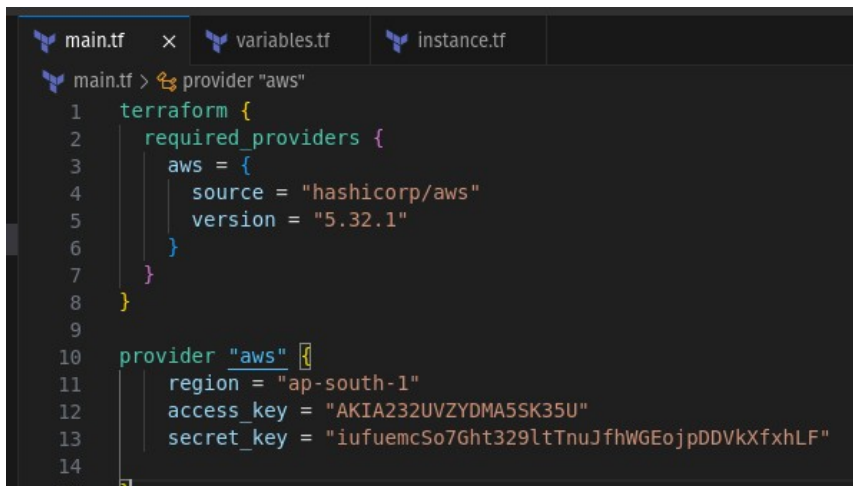
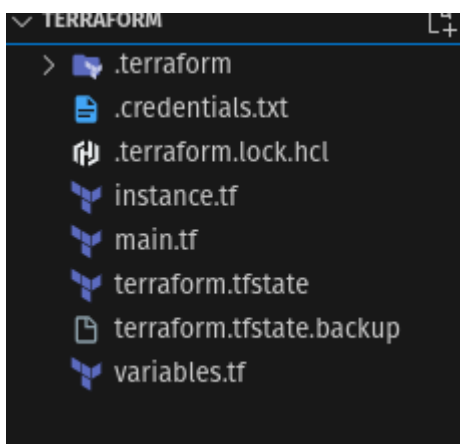
Experiment 5

Terraform Variables with Command Line Arguments

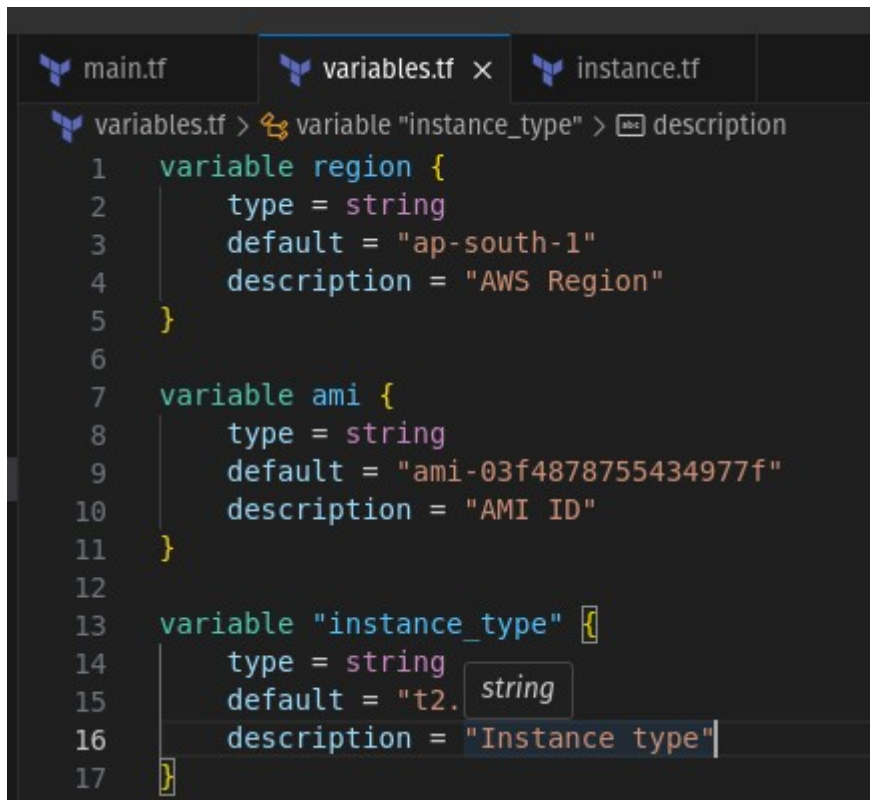
Aim

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

Step 1. Create a main file & terraform configuration file for EC2 Instance (instance.tf) & add variables to them.

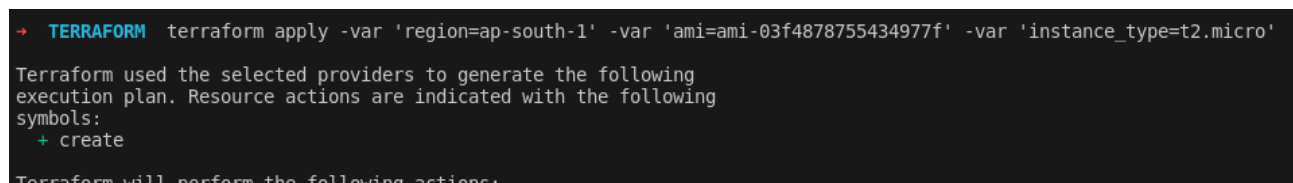


Step 2. Open a new file named variables.tf. Define variables for region, ami, secret_key, access_key and instance_type.



```
main.tf  variables.tf x  instance.tf
variables.tf > variable "instance_type" > description
1  variable region {
2      type = string
3      default = "ap-south-1"
4      description = "AWS Region"
5  }
6
7  variable ami {
8      type = string
9      default = "ami-03f4878755434977f"
10     description = "AMI ID"
11 }
12
13 variable "instance_type" {
14     type = string
15     default = "t2.micro"
16     description = "Instance type"
17 }
```

Step 3. Run the following Terraform commands to initialize and apply the configuration & pass variables as command line arguments.



```
→ TERRAFORM terraform apply -var 'region=ap-south-1' -var 'ami=ami-03f4878755434977f' -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:
```

Step 4. Verify Resources

	Name	Instance ID	Instance State
<input type="checkbox"/>	Exp5	i-0bfb81933ff1181b5	Running
<input type="checkbox"/>	Exp4	i-0dfe025c9691561dc	Terminated

Step 5. Cleanup resources

```
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.Keshav[0]: Destroying... [id=i-0bfb81933ff1181b5]  
aws_instance.Keshav[0]: Still destroying... [id=i-0bfb81933ff1181b5, 10s elapsed]  
aws_instance.Keshav[0]: Still destroying... [id=i-0bfb81933ff1181b5, 20s elapsed]  
aws_instance.Keshav[0]: Still destroying... [id=i-0bfb81933ff1181b5, 30s elapsed]  
aws_instance.Keshav[0]: Destruction complete after 30s  
  
Destroy complete! Resources: 1 destroyed.  
→ TERRAFORM
```