

Lesson 08 Demo 04

Creating a Simple HCL Configuration File

Objective: To create a basic HCL configuration file for setting up and managing AWS resources using Terraform

Prerequisites: AWS account with access keys, Terraform installed

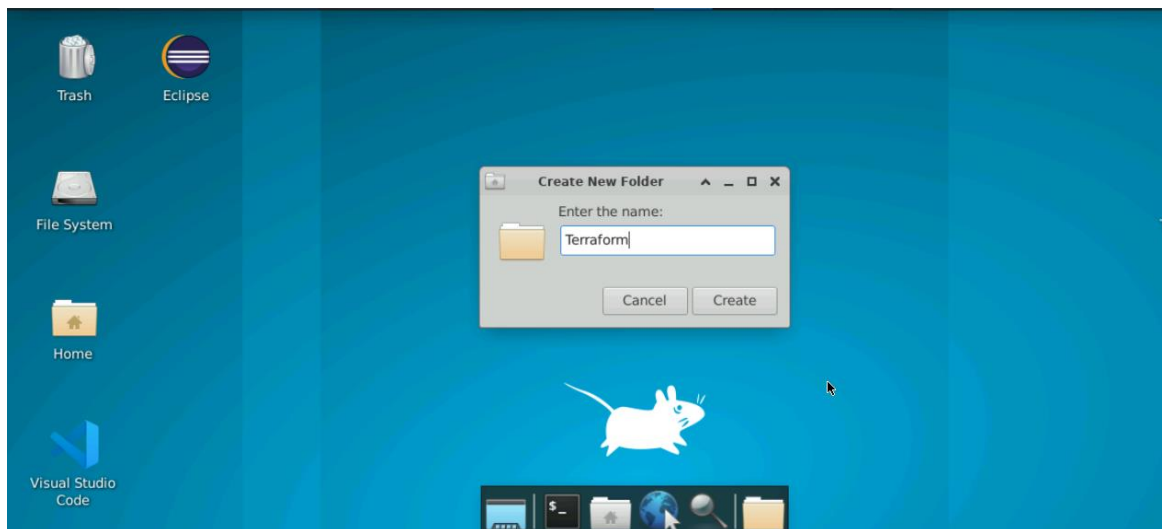
Tools required: Terraform, VS code, AWS CLI

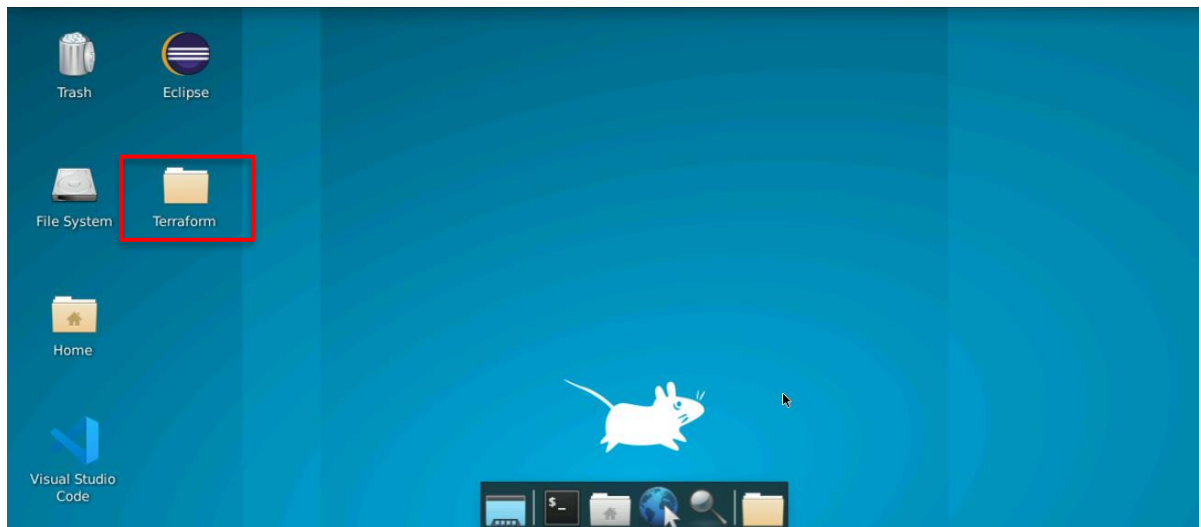
Steps to be followed:

1. Prepare files and credentials for using Terraform
2. Write the HCL configuration in the main.tf
3. Validate and apply the Terraform configuration
4. Clean up resources

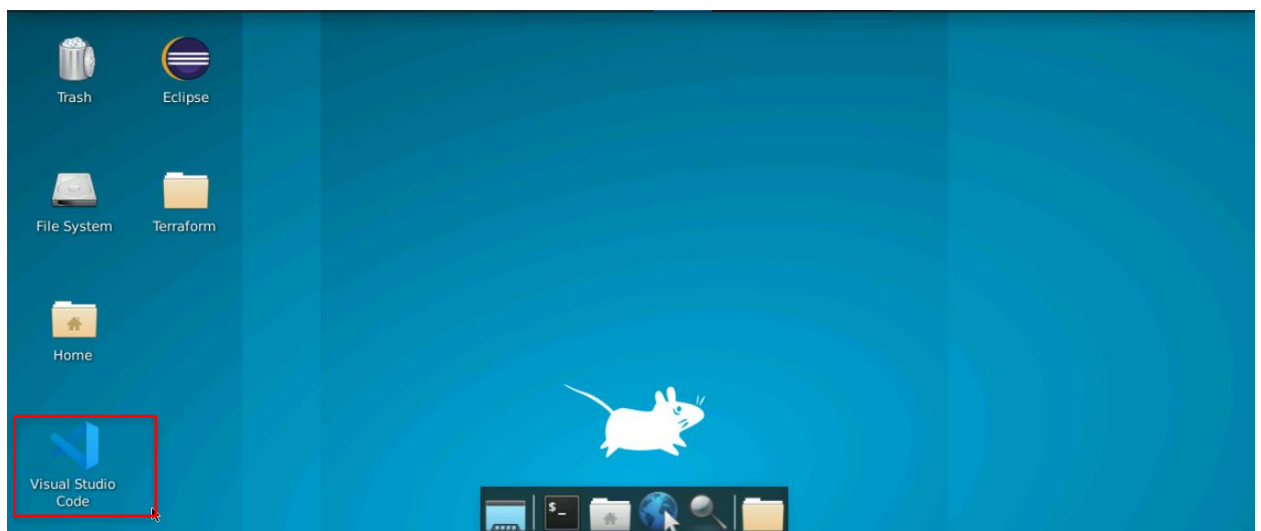
Step 1: Prepare files and credentials for using Terraform

1.1 Create a folder named **Terraform** on the desktop

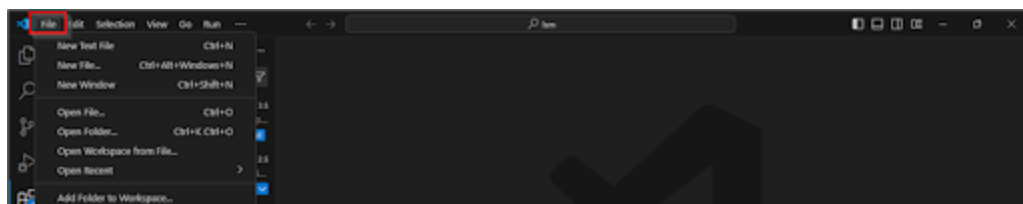




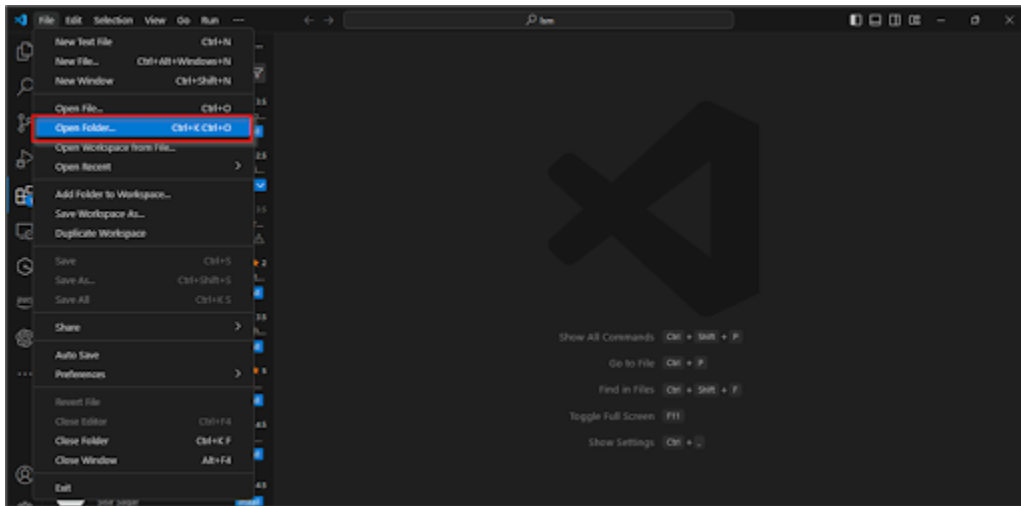
1.2 Double-click on the **VS Code editor** icon present on the desktop to open it



1.3 Open the **File** option present on the top console

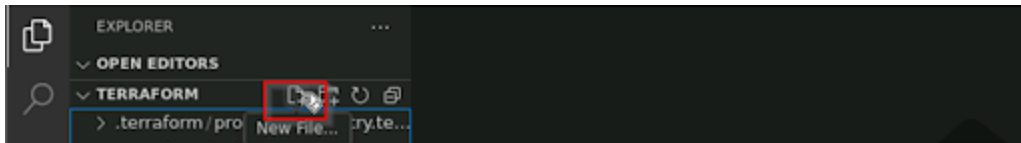


1.4 Click on the **Open Folder** from the drop-down menu to open the Terraform folder

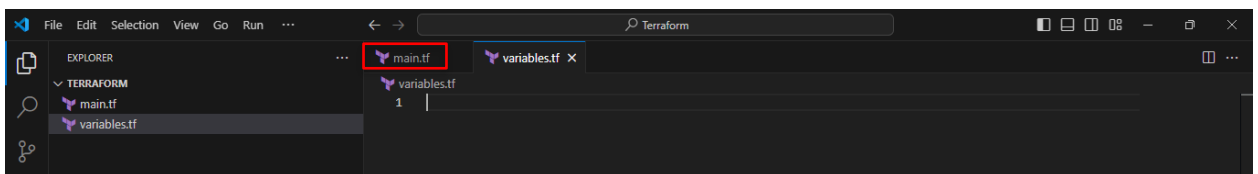


The **Terraform** folder will be opened in the VS Code editor.

1.5 Navigate to the **Terraform** folder on the editor and click on the **New File** option to create the main file



1.6 Create a file named **main.tf**



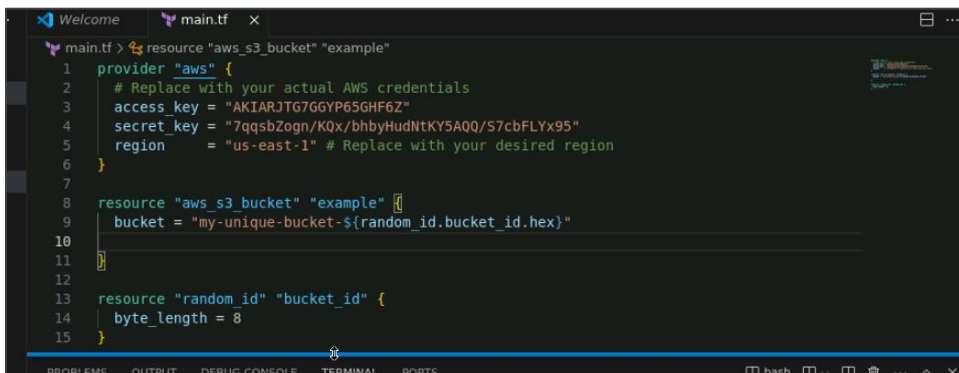
Step 2: Write the HCL configuration in main.tf

2.1 In the **main.tf** file, copy the following code to set up the AWS provider:

```
provider "aws"
{
  access_key = "<YOUR_AWS_ACCESS_KEY>"
  secret_key = "<YOUR_AWS_SECRET_KEY>"
  region = "us-east-1"
}

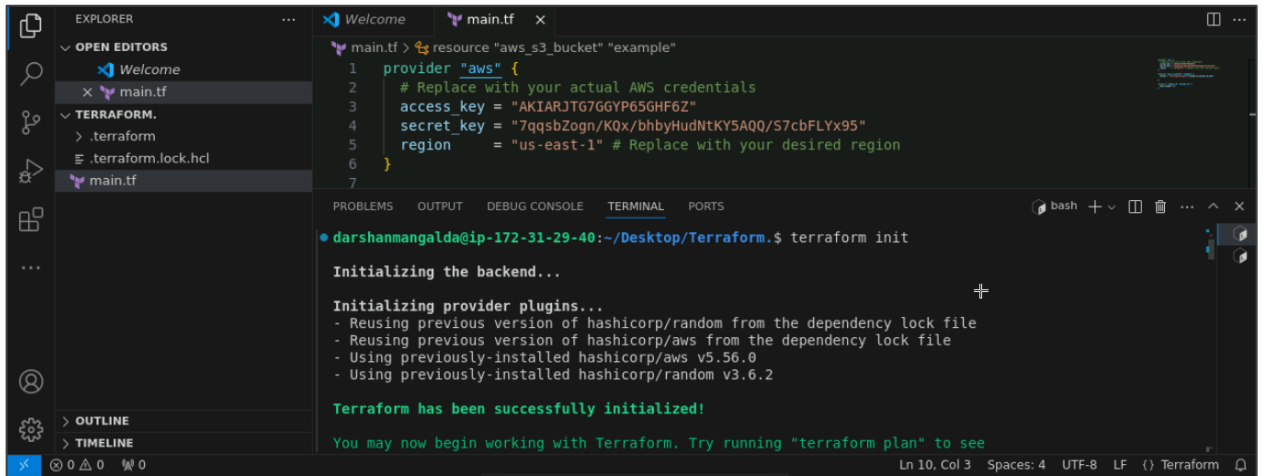
resource "aws_s3_bucket" "example"
{
  bucket = "my-unique-bucket-${random_id.bucket_id.hex}"
}

resource "random_id" "bucket_id"
{
  byte_length = 8
}
```



Note: Replace <YOUR_AWS_ACCESS_KEY> and <YOUR_AWS_SECRET_KEY> with your actual AWS credentials.

2.2 Open the terminal in VS Code and run the following command to initialize terraform: **terraform init**



```
main.tf > resource "aws_s3_bucket" "example"
1 provider "aws" {
2   # Replace with your actual AWS credentials
3   access_key = "AKIARJTG7GGYP65GHF6Z"
4   secret_key = "7qqsbZogn/KQx/bhbyHudNtKY5AQQ/S7cbFLYx95"
5   region     = "us-east-1" # Replace with your desired region
6 }
7

darshanmangalda@ip-172-31-29-40:~/Desktop/Terraform$ terraform init

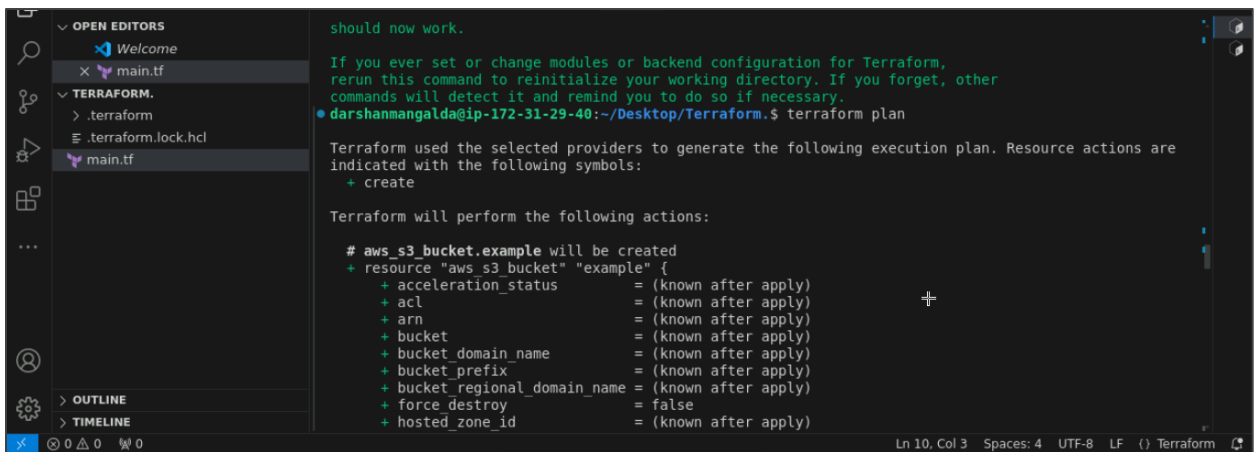
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/random from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.56.0
- Using previously-installed hashicorp/random v3.6.2

Terraform has been successfully initialized!
You may now begin working with Terraform. Try running "terraform plan" to see
```

Step 3: Validate and apply the Terraform configuration

3.1 Run the following command to create an execution plan: **terraform plan**



```
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.
darshanmangalda@ip-172-31-29-40:~/Desktop/Terraform$ terraform plan

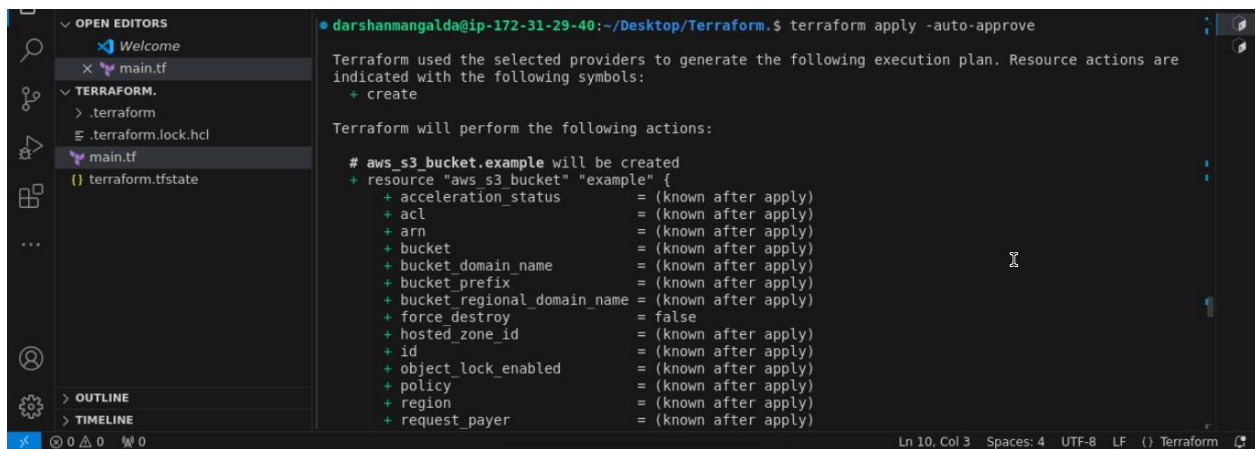
Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_s3_bucket.example will be created
+ resource "aws_s3_bucket" "example" {
+   acceleration_status = (known after apply)
+   acl                  = (known after apply)
+   arn                  = (known after apply)
+   bucket               = (known after apply)
+   bucket_domain_name   = (known after apply)
+   bucket_prefix        = (known after apply)
+   bucket_regional_domain_name = (known after apply)
+   force_destroy        = false
+   hosted_zone_id       = (known after apply)
```

3.2 Execute the following command to apply the configuration:

terraform apply -auto-approve



```
darshanmangal@ip-172-31-29-40:~/Desktop/Terraform$ terraform apply -auto-approve

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:

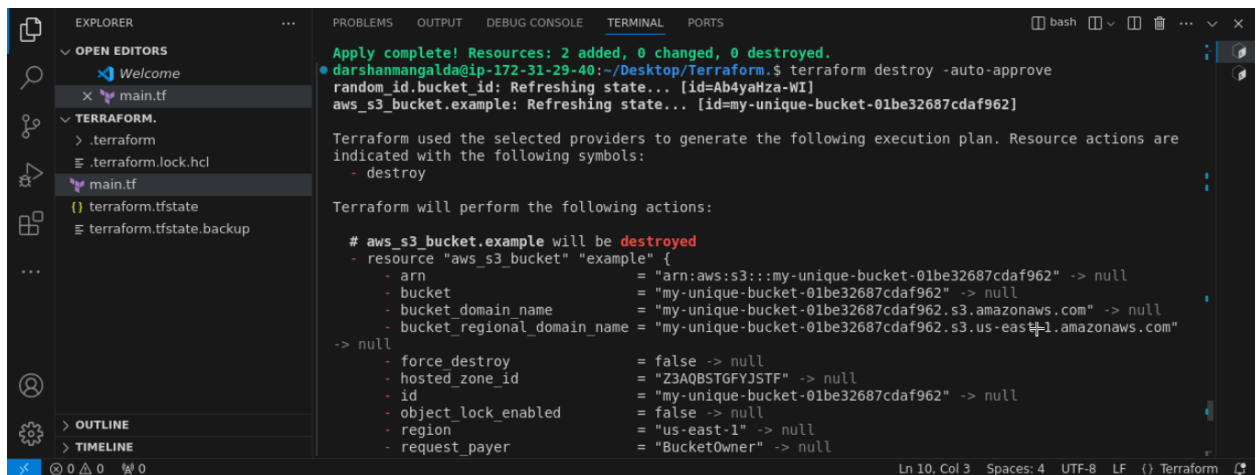
# aws_s3_bucket.example will be created
+ resource "aws_s3_bucket" "example" {
  + acceleration_status = (known after apply)
  + acl                 = (known after apply)
  + arn                 = (known after apply)
  + bucket              = (known after apply)
  + bucket_domain_name = (known after apply)
  + bucket_prefix       = (known after apply)
  + bucket_regional_domain_name = (known after apply)
  + force_destroy       = false
  + hosted_zone_id      = (known after apply)
  + id                  = (known after apply)
  + object_lock_enabled = (known after apply)
  + policy              = (known after apply)
  + region              = (known after apply)
  + request_payer       = (known after apply)
}
```

Terraform will provision the resources defined in the main.tf.

Step 4: Clean up resources

4.1 To destroy the terraform managed infrastructure, run the following command:

terraform destroy -auto-approve

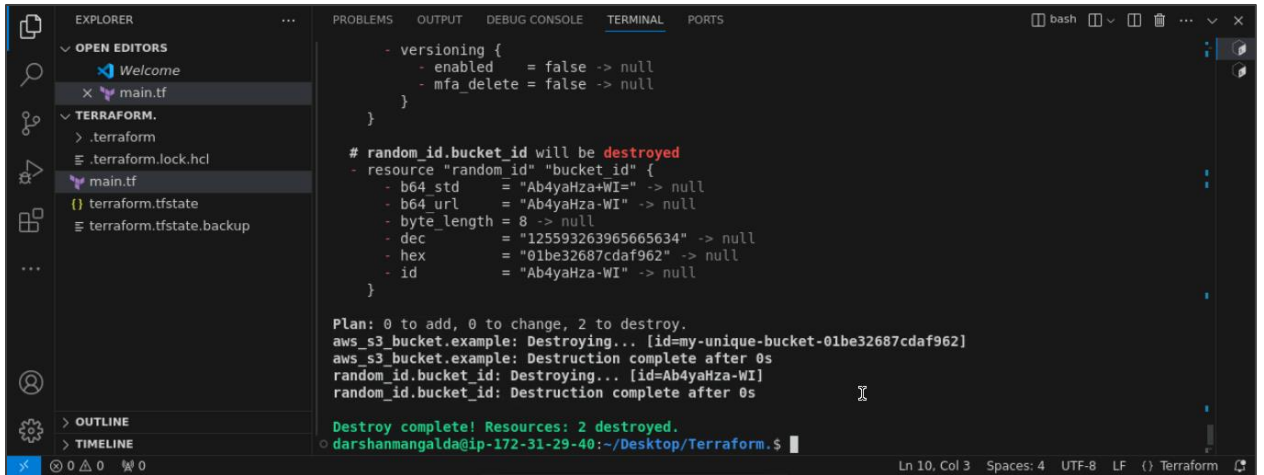


```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
darshanmangal@ip-172-31-29-40:~/Desktop/Terraform$ terraform destroy -auto-approve
random_id.bucket_id: Refreshing state... [id=Ab4yaHza-WI]
aws_s3_bucket.example: Refreshing state... [id=my-unique-bucket-01be32687cdaf962]

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_s3_bucket.example will be destroyed
- resource "aws_s3_bucket" "example" {
  - arn = "arn:aws:s3::my-unique-bucket-01be32687cdaf962" -> null
  - bucket = "my-unique-bucket-01be32687cdaf962" -> null
  - bucket_domain_name = "my-unique-bucket-01be32687cdaf962.s3.amazonaws.com" -> null
  - bucket_regional_domain_name = "my-unique-bucket-01be32687cdaf962.s3.us-east-1.amazonaws.com" -> null
  - force_destroy = false -> null
  - hosted_zone_id = "Z3AQBSTGFYJSTF" -> null
  - id = "my-unique-bucket-01be32687cdaf962" -> null
  - object_lock_enabled = false -> null
  - region = "us-east-1" -> null
  - request_payer = "BucketOwner" -> null
}
```



The screenshot shows the Visual Studio Code interface with a Terraform configuration file open in the editor and its execution output in the terminal.

EXPLORER:

- OPEN EDITORS
 - Welcome
 - main.tf
- TERRAFORM.
 - .terraform
 - .terraform.lock.hcl
 - main.tf
 - terraform.tfstate
 - terraform.tfstate.backup
- OUTLINE
- TIMELINE

main.tf:

```
versioning {
  enabled = false -> null
  mfa_delete = false -> null
}

# random_id.bucket_id will be destroyed
resource "random_id" "bucket_id" {
  b64_std = "Ab4yaHza+WI=" -> null
  b64_url = "Ab4yaHza-WI" -> null
  byte_length = 8 -> null
  dec = "125593263965665634" -> null
  hex = "01be32687cdaf962" -> null
  id = "Ab4yaHza-WI" -> null
}
```

Terminal Output:

```
Plan: 0 to add, 0 to change, 2 to destroy.
aws_s3_bucket.example: Destroying... [id=my-unique-bucket-01be32687cdaf962]
aws_s3_bucket.example: Destruction complete after 0s
random_id.bucket_id: Destroying... [id=Ab4yaHza-WI]
random_id.bucket_id: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
darshanmangalda@ip-172-31-29-40:~/Desktop/Terraform.$
```

Status Bar: Ln 10, Col 3 Spaces: 4 UTF-8 LF () Terraform

By following these steps, you have successfully created a simple HCL configuration file, initialized Terraform, and managed AWS resources efficiently.