

Lab Exercise 14–Provisioning an S3 Bucket on AWS

Exercise Steps:

Step 1: Create a New Directory:

Create a new directory to store your Terraform configuration:

```
mkdir Terraform-S3-Demo
cd Terraform-S3-Demo
```

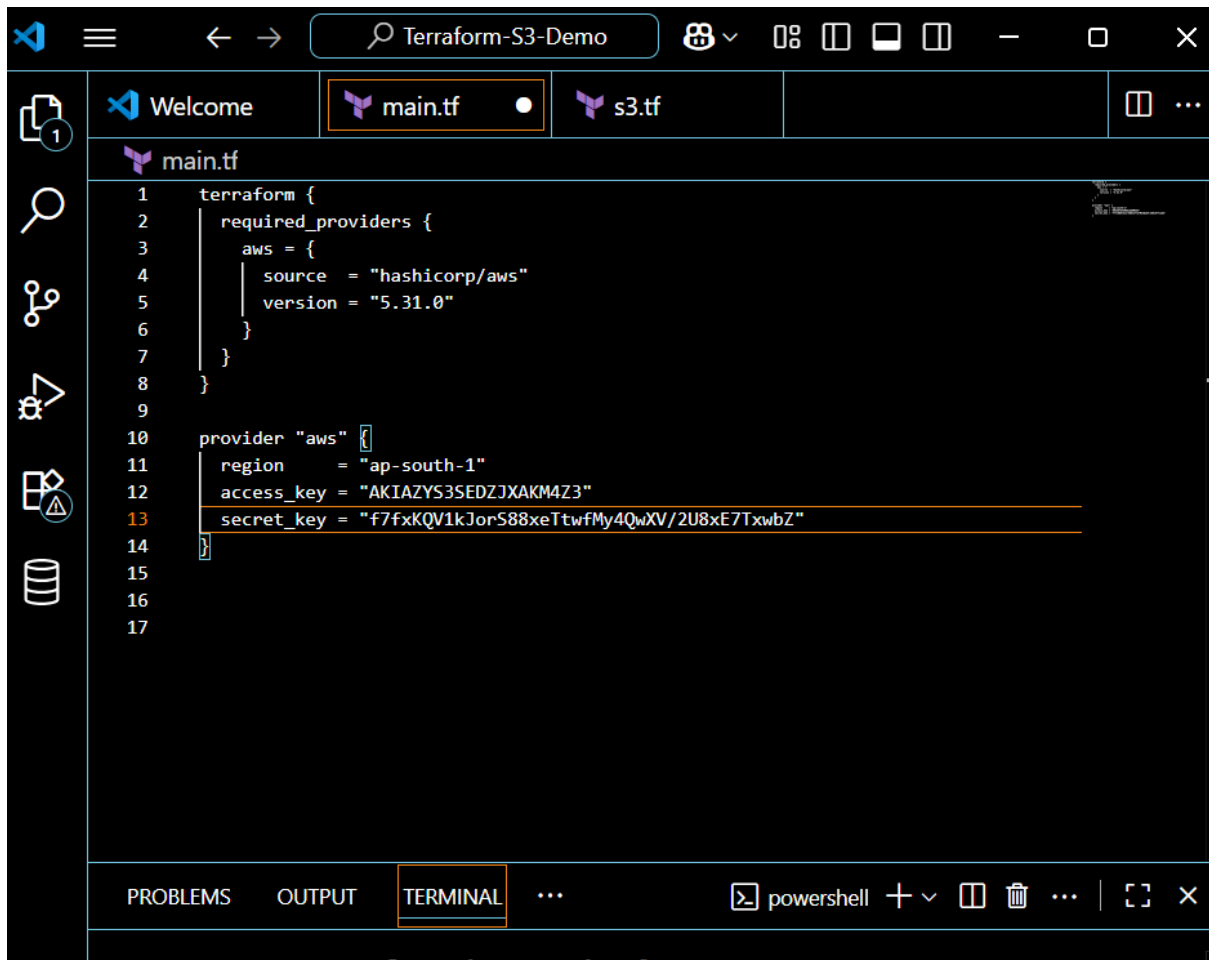
Step 2: Create the Terraform Configuration File (main.tf):

Create a file named main.tf with the following content:

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.31.0"
    }
  }
}

provider "aws" {
  region    = "us-east-1" # Replace with your preferred region
  access_key = "your IAM access key" # Replace with your Access Key
  secret_key = "your secret access key" # Replace with your Secret Key
}
```

This file sets up the Terraform AWS provider.

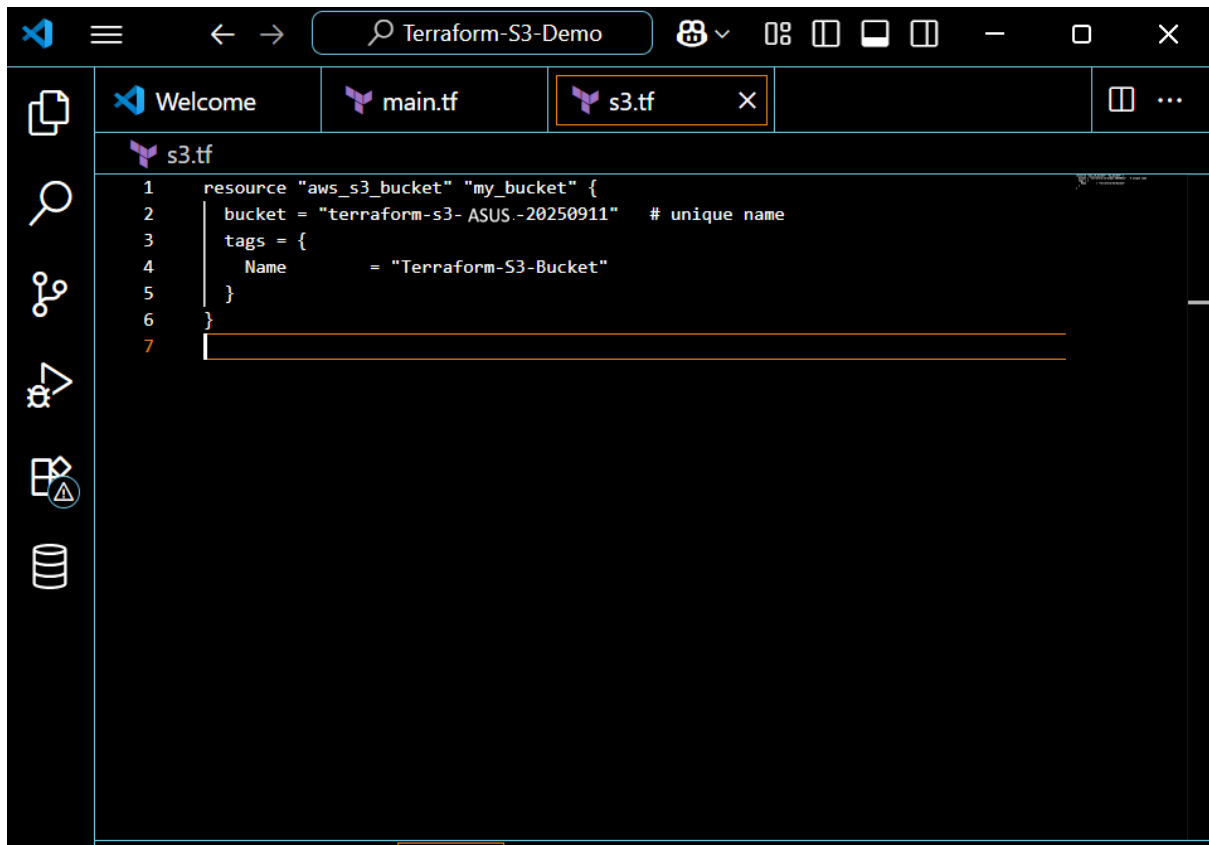


Step 3: Create a Terraform Configuration File for the S3 Bucket (s3.tf):

Create another file named s3.tf with the following content:

```
resource "aws_s3_bucket" "my_bucket" {
  bucket = "my-demo-s3-bucket"
  tags = {
    Name = "Terraform-S3-Bucket"
  }
}
```

This file provisions an S3 bucket with a unique name using a random string suffix.



Step 4: Initialize Terraform:

Run the following command to initialize your Terraform working directory:

```
terraform init
```

Step 5: Review the Plan:

Preview the changes Terraform will make:

```
terraform plan
```

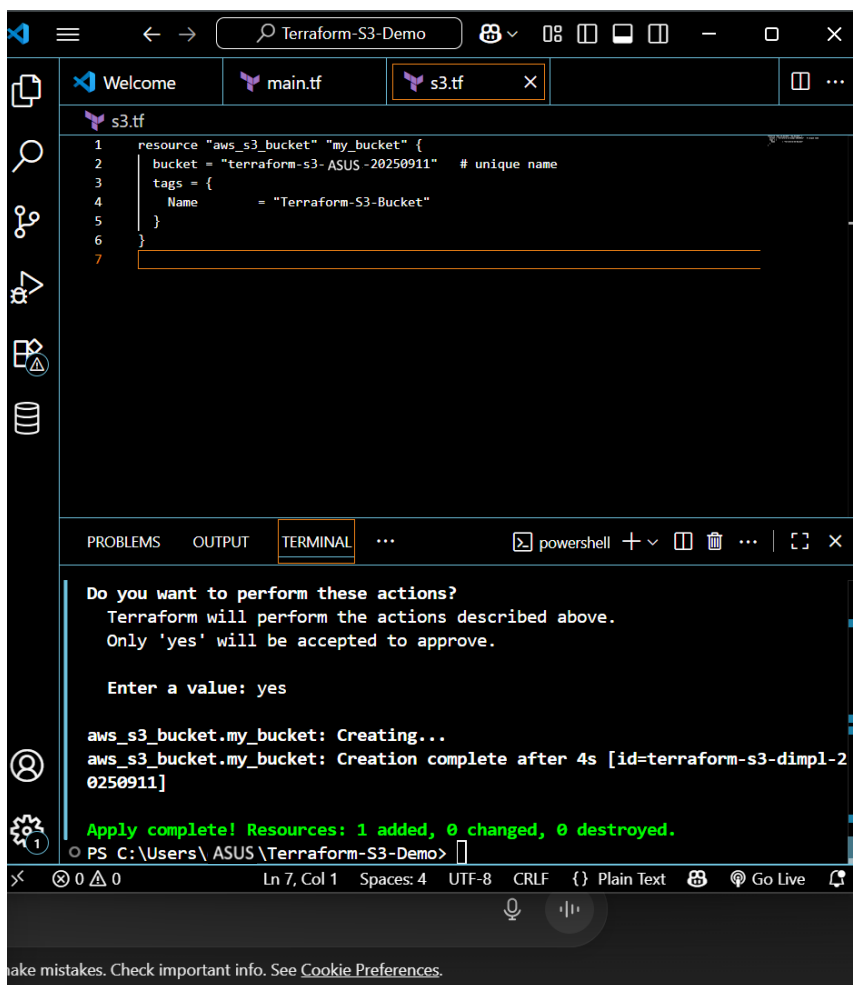
Review the output to ensure it meets your expectations.

Step 6: Apply the Changes:

Create the resources:

```
terraform apply
```

When prompted, type yes to confirm.



The screenshot shows the Visual Studio Code interface with a Terraform configuration file named `s3.tf` open. The configuration defines an `aws_s3_bucket` resource named `my_bucket` with a unique name and a specific tag. The terminal window at the bottom shows the execution of `terraform apply`, which prompts for confirmation and successfully creates the bucket.

```
s3.tf
1 resource "aws_s3_bucket" "my_bucket" {
2   bucket = "terraform-s3- ASUS-20250911" # unique name
3   tags = {
4     Name = "Terraform-S3-Bucket"
5   }
6 }
7

PROBLEMS OUTPUT TERMINAL
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

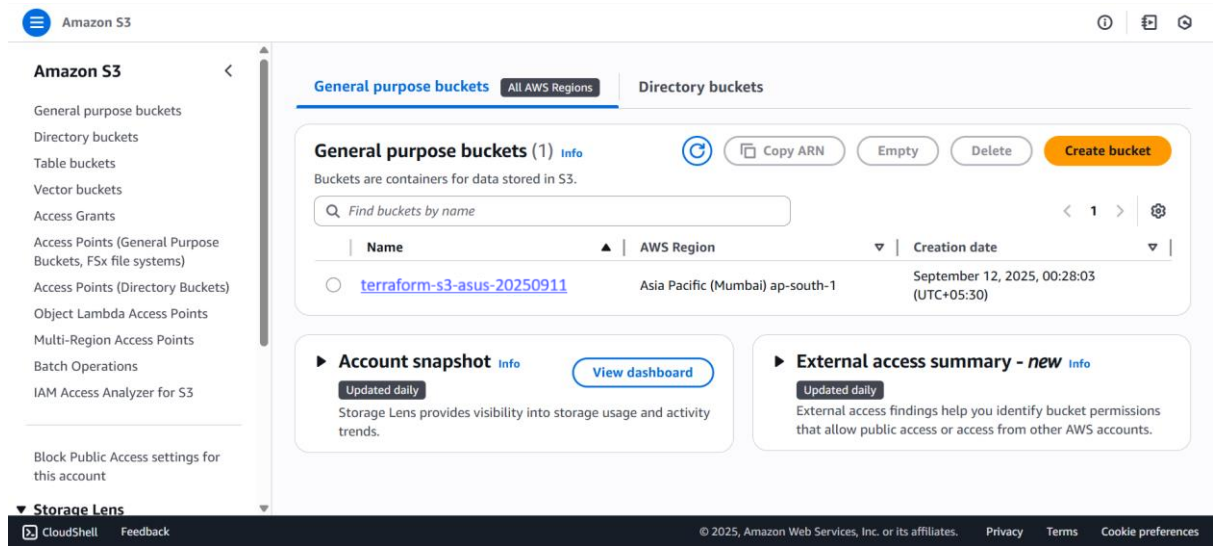
aws_s3_bucket.my_bucket: Creating...
aws_s3_bucket.my_bucket: Creation complete after 4s [id=terraform-s3-dimpl-20250911]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\ASUS\Terraform-S3-Demo>
```

Step 7: Verify Resources:

1. Log in to your AWS Management Console.
2. Navigate to the **S3** dashboard.

3. Verify that the S3 bucket has been created with the specified configuration.

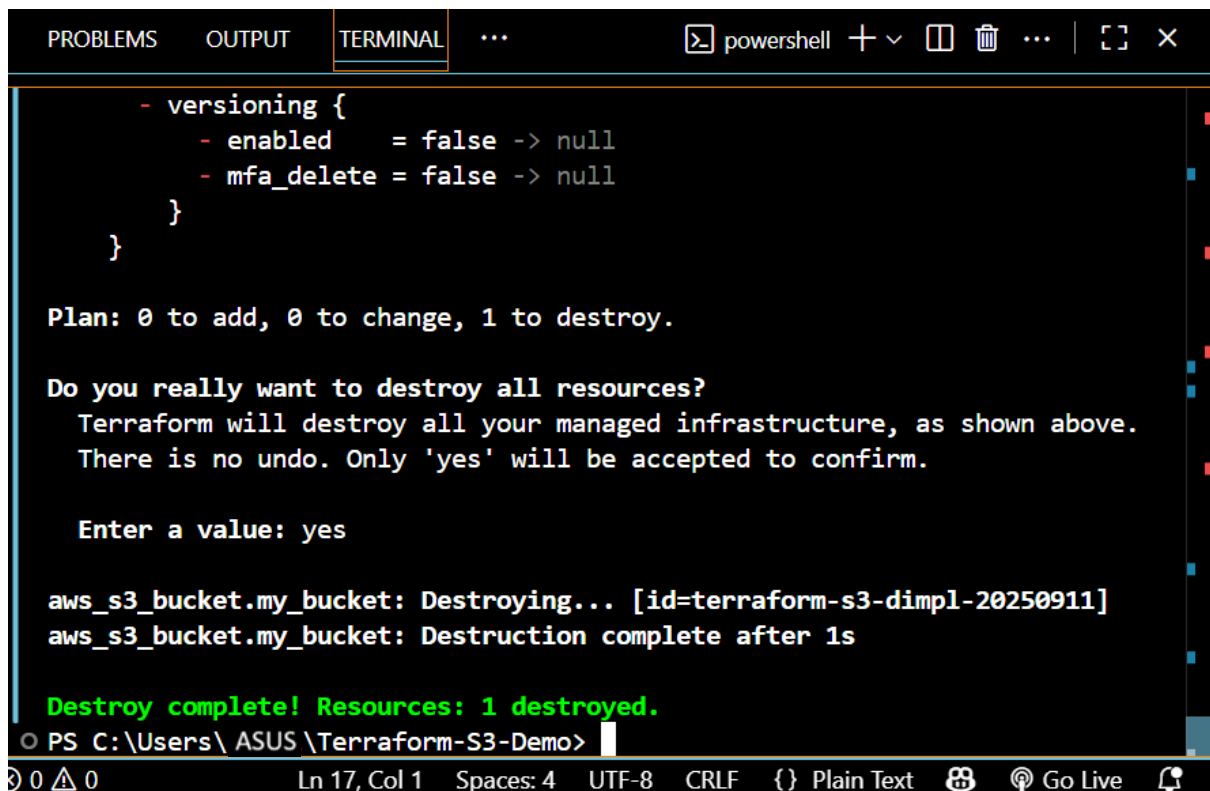


Step 8: Cleanup Resources:

To remove the resources created, run the following command:

```
terraform destroy
```

When prompted, type yes to confirm.



The screenshot shows a PowerShell terminal window with the 'TERMINAL' tab selected. The terminal displays the output of a Terraform destroy command. It shows a plan with 0 additions, 0 changes, and 1 destruction. A confirmation prompt asks if the user really wants to destroy all resources, with a warning that there is no undo. The user enters 'yes'. The terminal then shows the destruction of the 'aws_s3_bucket.my_bucket' resource, with a message indicating it is complete after 1s. Finally, it displays 'Destroy complete! Resources: 1 destroyed.' in green text. The command prompt at the bottom shows the current directory as 'C:\Users\ASUS\Terraform-S3-Demo'.

```
- versioning {  
  - enabled    = false -> null  
  - mfa_delete = false -> 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_s3_bucket.my_bucket: Destroying... [id=terraform-s3-dimpl-20250911]  
aws_s3_bucket.my_bucket: Destruction complete after 1s  
  
Destroy complete! Resources: 1 destroyed.  
PS C:\Users\ASUS\Terraform-S3-Demo>
```