

Lab Exercise 5–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
```

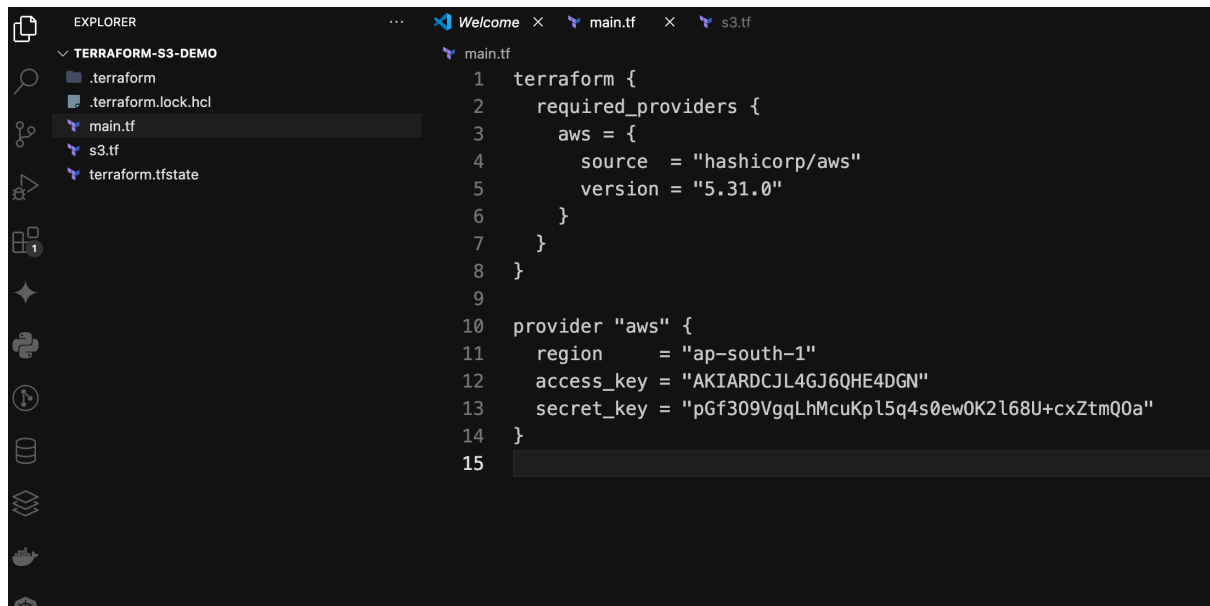
```
(base) aryanbansal@Aryans-MacBook-Air-10 Terraform-Lab % cd ..
(base) aryanbansal@Aryans-MacBook-Air-10 SYSTEMPROVISIONINGLAB % cd Terraform-Lab
(base) aryanbansal@Aryans-MacBook-Air-10 Terraform-Lab % mkdir Terraform-S3-Demo
(base) aryanbansal@Aryans-MacBook-Air-10 Terraform-Lab % cd Terraform-S3-Demo
(base) aryanbansal@Aryans-MacBook-Air-10 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
}
```



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying the file structure of a project named 'TERRAFORM-S3-DEMO'. The files listed are .terraform, .terraform.lock.hcl, main.tf, s3.tf, and terraform.tfstate. The main.tf file is open in the editor, showing the following Terraform configuration:

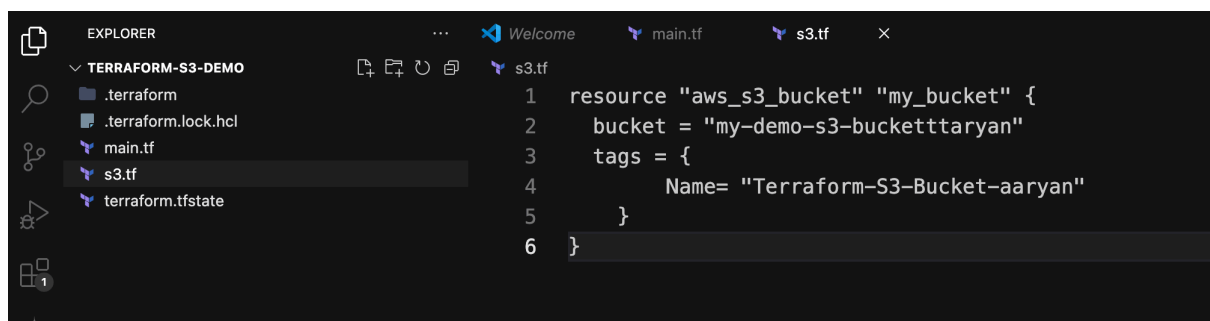
```
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "5.31.0"
6     }
7   }
8 }
9
10 provider "aws" {
11   region = "ap-south-1"
12   access_key = "AKIARDCL4GJ6QHE4DGN"
13   secret_key = "pGf309VgqLhMcuKp15q4s0ew0K2l68U+cxZtmQ0a"
14 }
15
```

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"
  }
}
```



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying the file structure of the 'TERRAFORM-S3-DEMO' project. The files listed are .terraform, .terraform.lock.hcl, main.tf, s3.tf, and terraform.tfstate. The s3.tf file is open in the editor, showing the following Terraform configuration:

```
1 resource "aws_s3_bucket" "my_bucket" {
2   bucket = "my-demo-s3-bucketttaryan"
3   tags = {
4     Name= "Terraform-S3-Bucket-aaryan"
5   }
6 }
```

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
```

```
[(base) arianbansal@Aryans-MacBook-Air-10 Terraform-S3-Demo % terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.31.0

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.
```

Step 5: Review the Plan:

Preview the changes Terraform will make:

```
terraform plan
```

Review the output to ensure it meets your expectations.

```
+ logging (known after apply)
+ object_lock_configuration (known after apply)
+ replication_configuration (known after apply)
+ server_side_encryption_configuration (known after apply)
+ versioning (known after apply)
+ website (known after apply)
}

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

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
[(base) arianbansal@Aryans-MacBook-Air-10 Terraform-S3-Demo % terraform apply
```

Step 6: Apply the Changes:

Create the resources:

```
terraform apply
```

When prompted, type yes to confirm.

```
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.

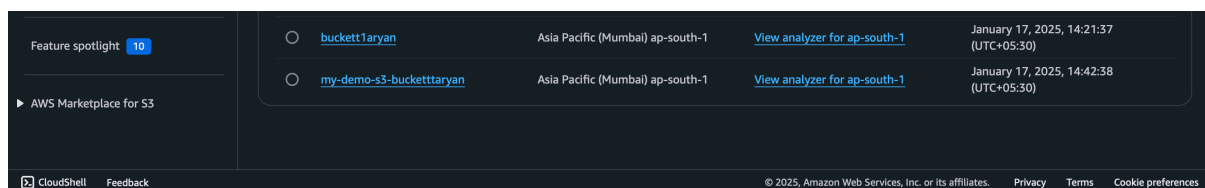
  Enter a value: yes

aws_s3_bucket.my_bucket: Creating...
aws_s3_bucket.my_bucket: Creation complete after 3s [id=my-demo-s3-bucketttaryan]

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

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.

```
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=my-demo-s3-bucketttaryan]
```

```
aws_s3_bucket.my_bucket: Destruction complete after 1s
```

```
Destroy complete! Resources: 1 destroyed.
```

```
(base) aryanbansal@Aryans-MacBook-Air-10 Terraform-S3-Demo %
```
