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
```

```
C:\Terraform-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 v6.11.0

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

```
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)

      + lifecycle_rule (known after apply)

      + 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: 2 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.
```

Review the output to ensure it meets your expectations.

Step 6: Apply the Changes:

Create the resources:

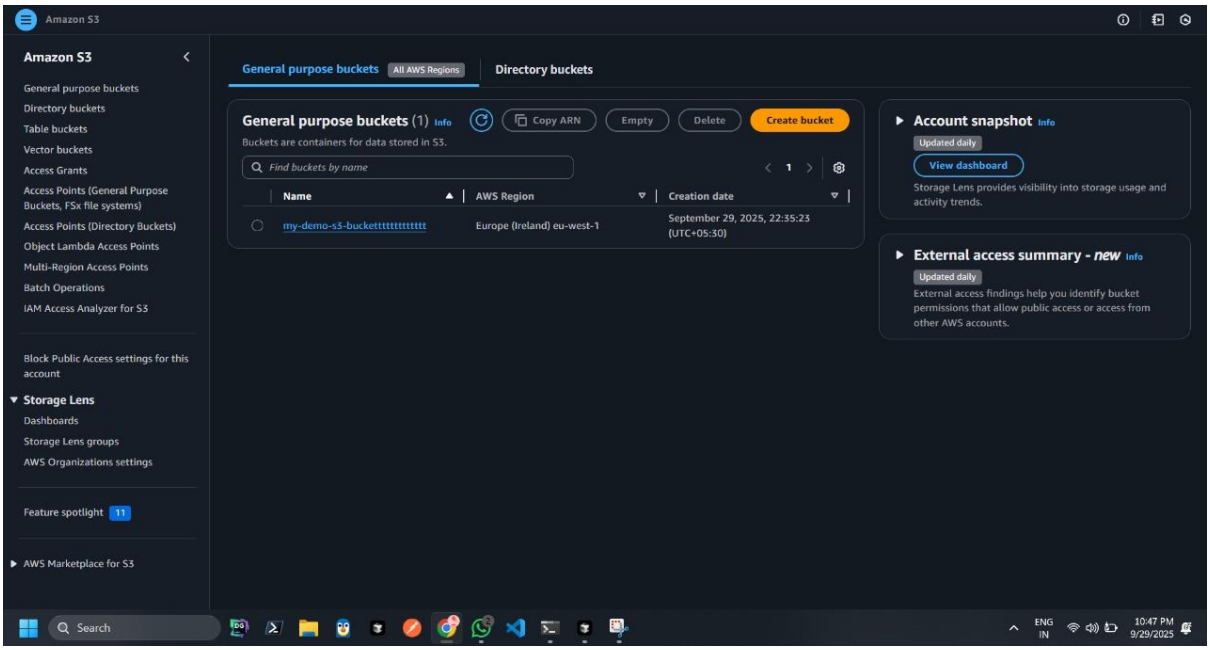
terraform apply

```
  Enter a value: yes

aws_s3_bucket.my_bucket: Creating...
aws_s3_bucket.my_bucket: Creation complete after 7s [id=my-demo-s3-bucketttttttttttttt]

  Warning: Argument is deprecated

    with aws_s3_bucket.my_bucket,
    on s3.tf line 3, in resource "aws_s3_bucket" "my_bucket":
     3:   acl    = "private"

  acl is deprecated. Use the aws_s3_bucket_acl resource instead.


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

When prompted, type yes to confirm.

Step 7: Verify Resources:

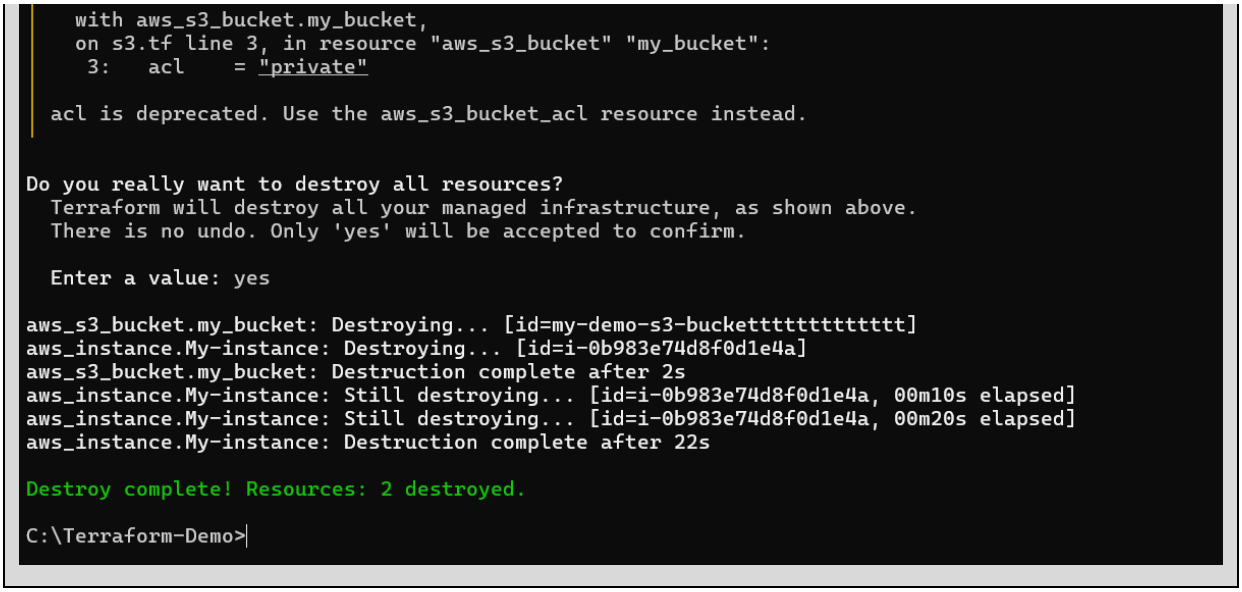
1. Log in to your AWS Management Console.
2. Navigate to the **S3** dashboard.
3. Verify that the S3 bucket has been created with the specified configuration.



Step 8: Cleanup Resources:

To remove the resources created, run the following command:

```
terraform destroy
```



When prompted, type yes to confirm.
