

## Lab Exercise 5–Provisioning an S3 Bucket on AWS

---

### Exercise Steps:

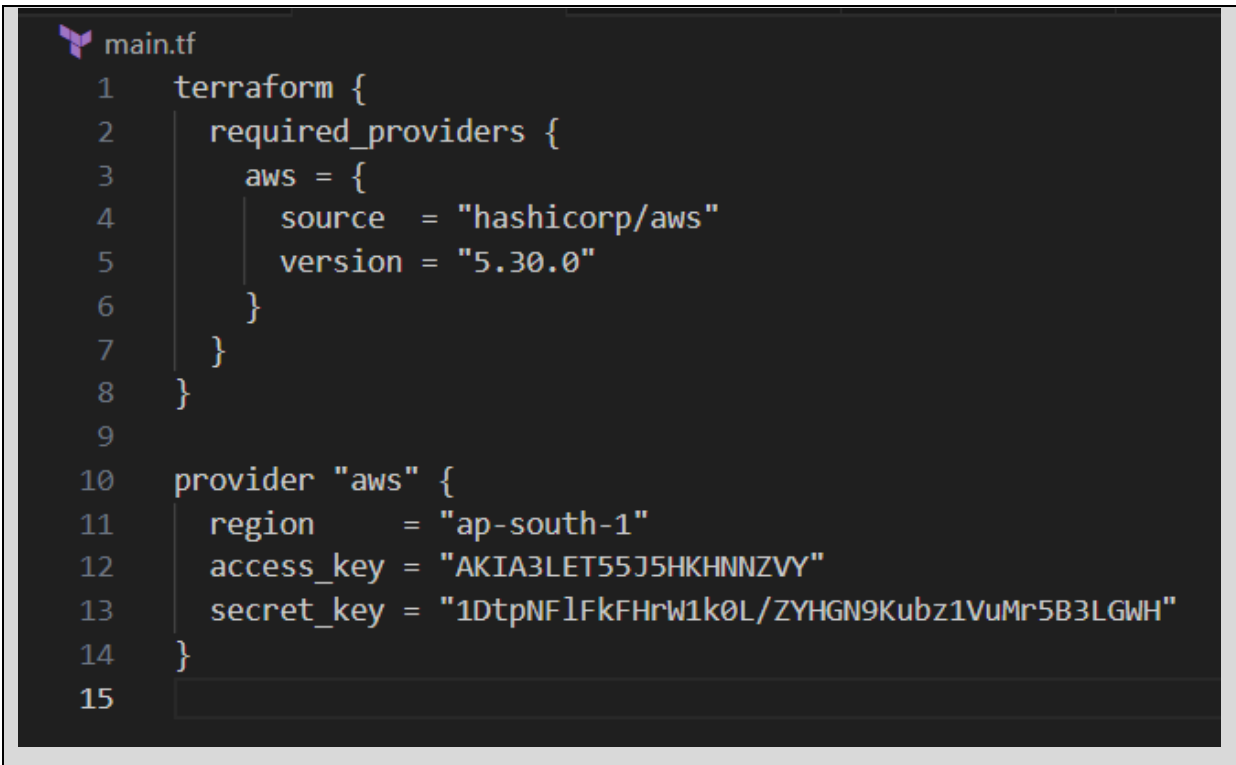
#### Step 1: Create a New Directory:

Create a new directory to store your Terraform configuration:

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

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

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

A screenshot of a code editor with a dark background. The file name 'main.tf' is shown in the top left corner with a purple icon. The code is written in a light-colored font and is numbered from 1 to 15 on the left side. The code defines the Terraform AWS provider configuration, including required providers and provider-specific settings like region, access key, and secret key.

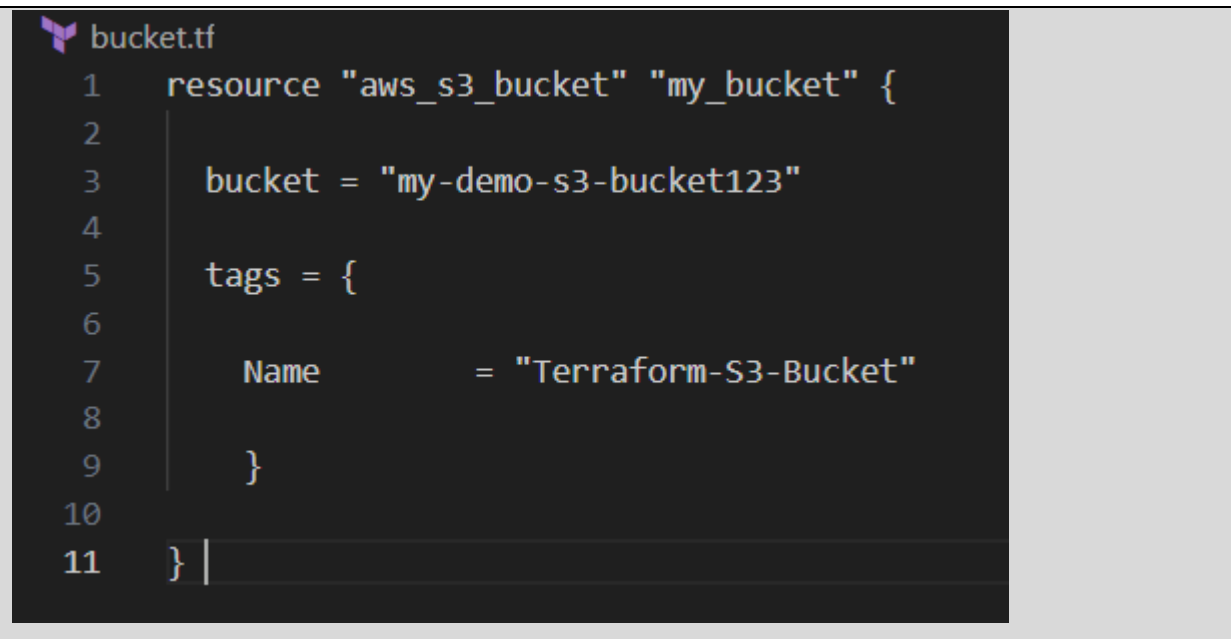
```
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "5.30.0"
6     }
7   }
8 }
9
10 provider "aws" {
11   region      = "ap-south-1"
12   access_key  = "AKIA3LET55J5HKHNNZVY"
13   secret_key  = "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"
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:

A screenshot of a code editor with a dark background. The file name 'bucket.tf' is shown in the top left corner. The code is written in a light-colored font and is as follows:

```
1 resource "aws_s3_bucket" "my_bucket" {  
2  
3     bucket = "my-demo-s3-bucket123"  
4  
5     tags = {  
6  
7         Name      = "Terraform-S3-Bucket"  
8  
9     }  
10  
11 }
```

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
```

```
iainyo@Acernitro MINGW64 /d/terraform-demo
$ 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.my_instance will be created
+ resource "aws_instance" "my_instance" {
  + ami                        = "ami-03235cc8fe4d9bf1e"
  + 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)
```

Review the output to ensure it meets your expectations.

---

## Step 6: Apply the Changes:

Create the resources:

```
terraform apply
```

```
amyo@Acernitro MINGW64 /d/terraform-demo
$ terraform apply

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.my_instance will be created
+ resource "aws_instance" "my_instance" {
  + ami                        = "ami-03235cc8fe4d9bf1e"
  + 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)
```

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
```

```
aws_s3_bucket.my_bucket: Destroying... [id=my-demo-s3-bucket123]
aws_instance.my_instance: Destroying... [id=i-07fcbf6fd496726d9]
aws_s3_bucket.my_bucket: Destruction complete after 1s
aws_instance.my_instance: Still destroying... [id=i-07fcbf6fd496726d9, 10s elapsed]
aws_instance.my_instance: Still destroying... [id=i-07fcbf6fd496726d9, 20s elapsed]
aws_instance.my_instance: Still destroying... [id=i-07fcbf6fd496726d9, 30s elapsed]
aws_instance.my_instance: Still destroying... [id=i-07fcbf6fd496726d9, 40s elapsed]
aws_instance.my_instance: Destruction complete after 41s
Destroy complete! Resources: 2 destroyed.
```

When prompted, type yes to confirm.

---