

Pratik Agrawal

500123601

Devops B2

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:

```
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
```

```
PS C:\Users\prati\OneDrive\Documents\Devsecops Lab\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.

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

Step 6: Apply the Changes:

Create the resources:

```
terraform apply
```

When prompted, type yes to confirm.

```
PS C:\Users\prati\OneDrive\Documents\Devsecops lab\Terraform-S3-Demo> terraform import aws_s3_bucket.my_bucket my-demo-s3-bucket03
aws_s3_bucket.my_bucket: Importing from ID "my-demo-s3-bucket03"...
aws_s3_bucket.my_bucket: Import prepared!
  Prepared aws_s3_bucket for import
aws_s3_bucket.my_bucket: Refreshing state... [id=my-demo-s3-bucket03]

Import successful!

The resources that were imported are shown above. These resources are now in
your Terraform state and will henceforth be managed by Terraform.
```

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.

```
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-bucket03]  
aws_s3_bucket.my_bucket: Destruction complete after 1s  
  
Destroy complete! Resources: 1 destroyed.
```