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:

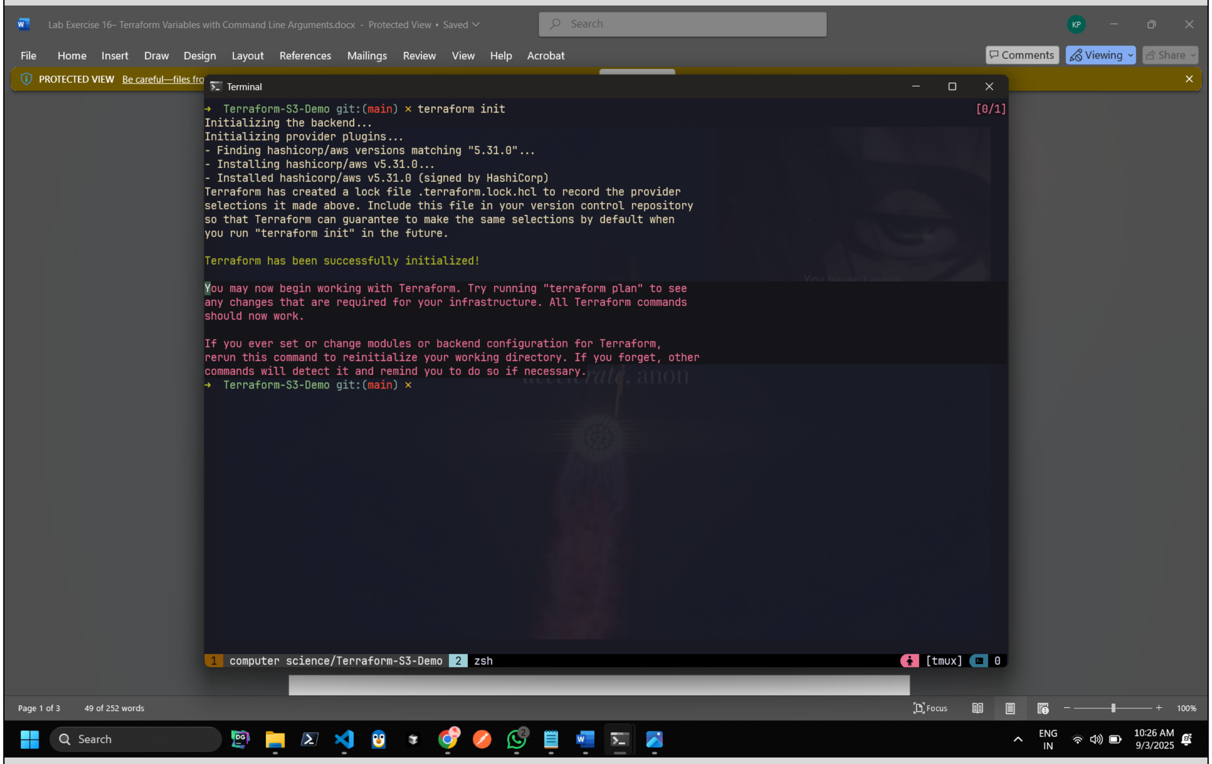
```
resource "aws_s3_bucket" "my_bucket" {
  bucket = "my-demo-s3-bucket"
  tags = {
    Name        = "Terraform-S3-Bucket"
    }
}
```

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
```



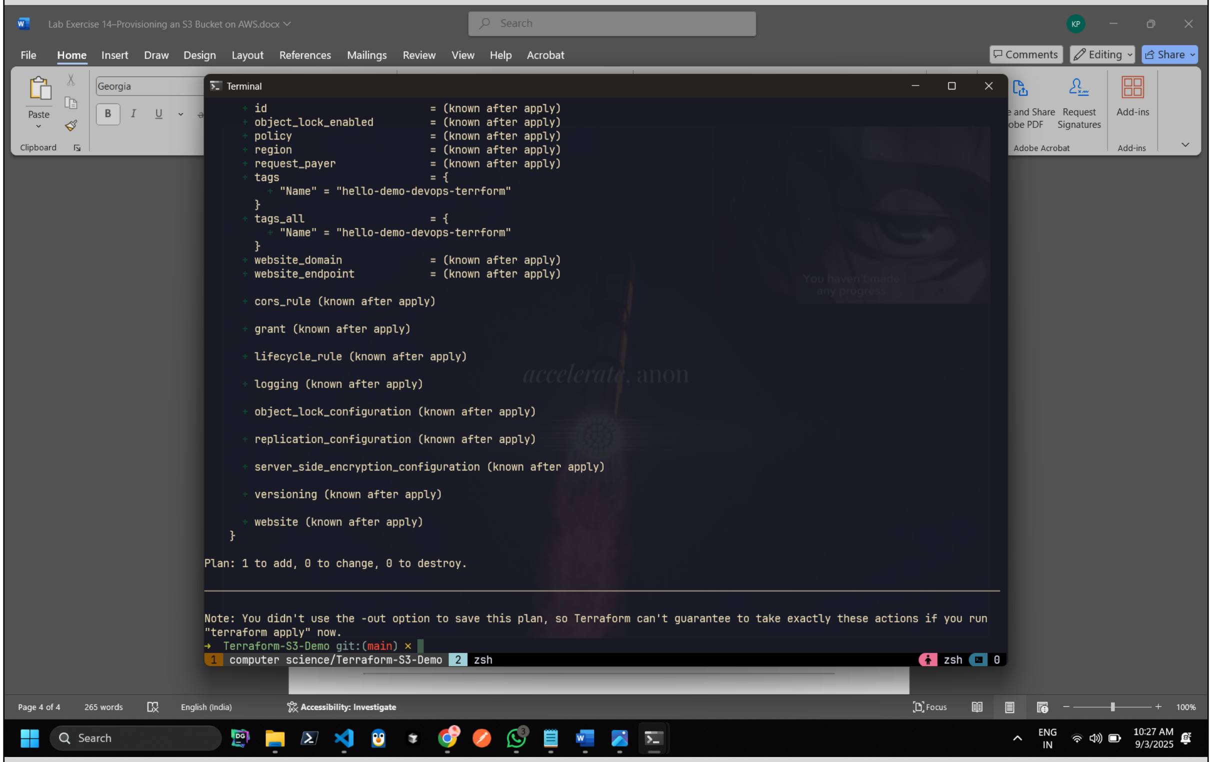
```
→  Terraform-S3-Demo git:(main) ✗ terraform init                                    [0/1]
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.31.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
→  Terraform-S3-Demo git:(main) ✗
```

Lab Exercise 16– Terraform Variables with Command Line Arguments.docx - Protected View • Saved
Search
File Home Insert Draw Design Layout References Mailings Review View Help Acrobat
Comments
Viewing
Share
PROTECTED VIEW Be careful—files fro
Terminal
1 computer science/Terraform-S3-Demo 2 zsh
[tmux] 0
Page 1 of 3 49 of 252 words
Focus
100%
Search
ENG IN
10:26 AM 9/3/2025

Step 5: Review the Plan:

Preview the changes Terraform will make:

terraform plan



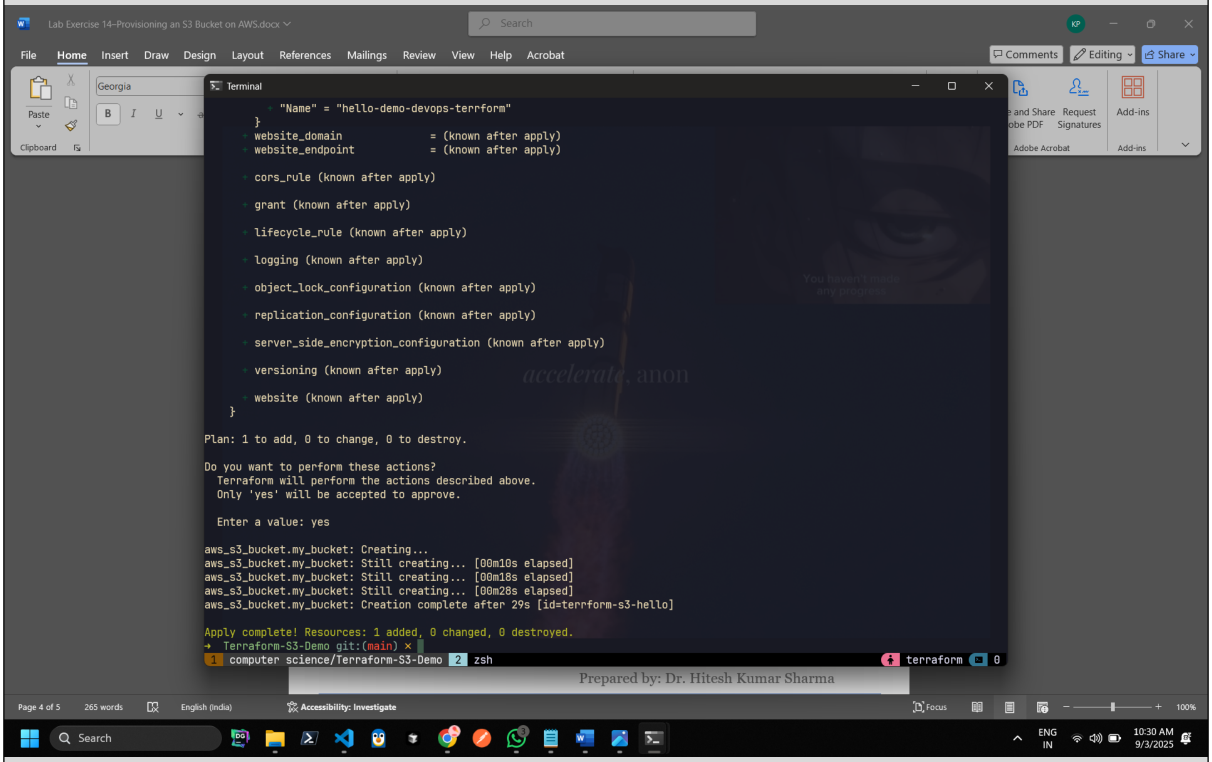
```
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Name" = "hello-demo-devops-terrform"
        }
      + tags_all                    = {
          + "Name" = "hello-demo-devops-terrform"
        }
      + 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: 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.
➜  Terraform-S3-Demo git:(main) ✗
 1  computer science/Terraform-S3-Demo  2  zsh                                      zsh  0
```

Review the output to ensure it meets your expectations.

Step 6: Apply the Changes:

Create the resources:

TERRAFORM APPLY



```
        + "Name" = "hello-demo-devops-terrform"
      }
    + 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: 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: Still creating... [00m10s elapsed]
aws_s3_bucket.my_bucket: Still creating... [00m18s elapsed]
aws_s3_bucket.my_bucket: Still creating... [00m28s elapsed]
aws_s3_bucket.my_bucket: Creation complete after 29s [id=terrform-s3-hello]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
→ Terraform-S3-Demo git:(main) ✗
 1  computer science/Terraform-S3-Demo  2  zsh                                  terraform  0
```

Lab Exercise 14-Provisioning an S3 Bucket on AWS.docx
Terminal
Prepared by: Dr. Hitesh Kumar Sharma
Page 4 of 5
265 words
English (India)
Accessibility: Investigate

When prompted, type yes to confirm.

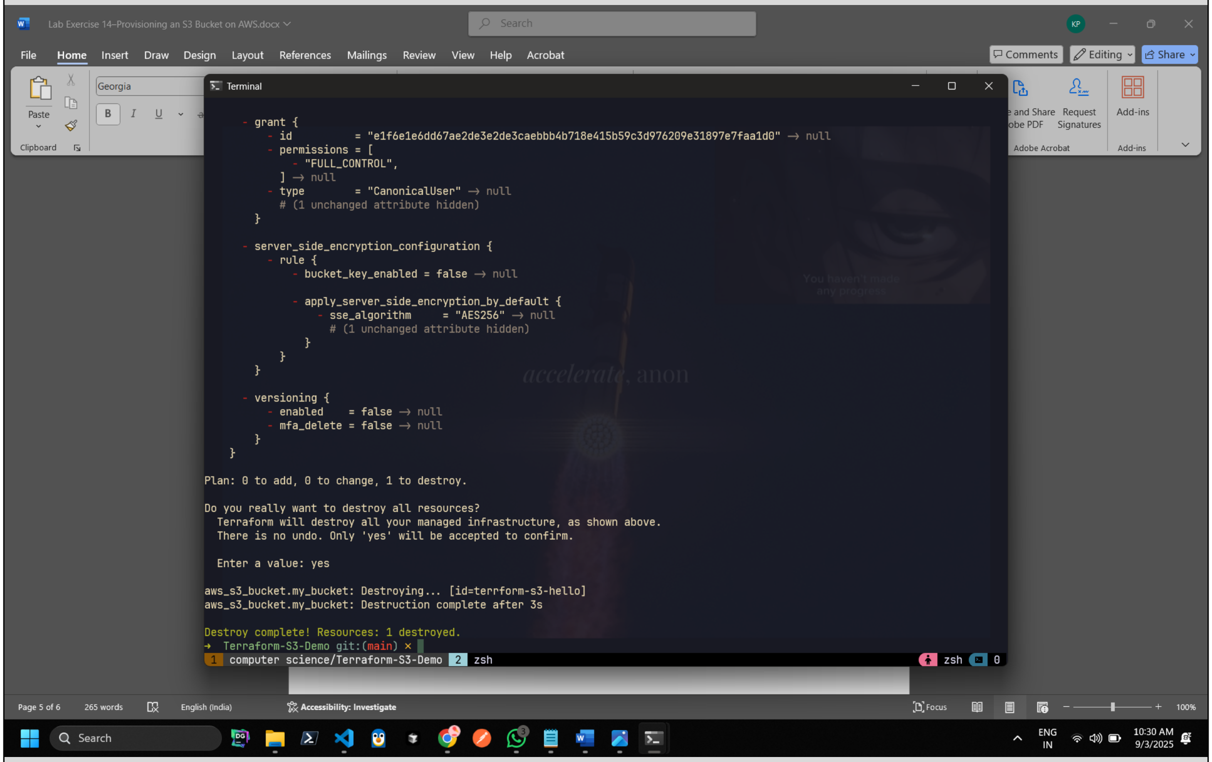
Step 7: Verify Resources:

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
```



```
      - grant {
          - id          = "e1f6e1e6dd67ae2de3e2de3caebbb4b718e415b59c3d976209e31897e7faa1d0" → null
          - permissions = [
              - "FULL_CONTROL",
            ] → null
          - type        = "CanonicalUser" → null
            # (1 unchanged attribute hidden)
        }

      - server_side_encryption_configuration {
          - rule {
              - bucket_key_enabled = false → null

              - apply_server_side_encryption_by_default {
                  - sse_algorithm     = "AES256" → null
                    # (1 unchanged attribute hidden)
                }
            }
        }

      - 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=terrform-s3-hello]
aws_s3_bucket.my_bucket: Destruction complete after 3s

Destroy complete! Resources: 1 destroyed.
→  Terraform-S3-Demo git:(main) ✗
 1  computer science/Terraform-S3-Demo  2  zsh
```

When prompted, type yes to confirm.