Automated S3 Bucket Creation and Object Upload with Terraform

Mohammed Amir

Project Overview

This project demonstrates the use of **Terraform** for automating AWS S3 bucket creation and object management. **Terraform** is an open-source Infrastructure as Code (IaC) tool that allows you to define and provision cloud resources declaratively using configuration files.

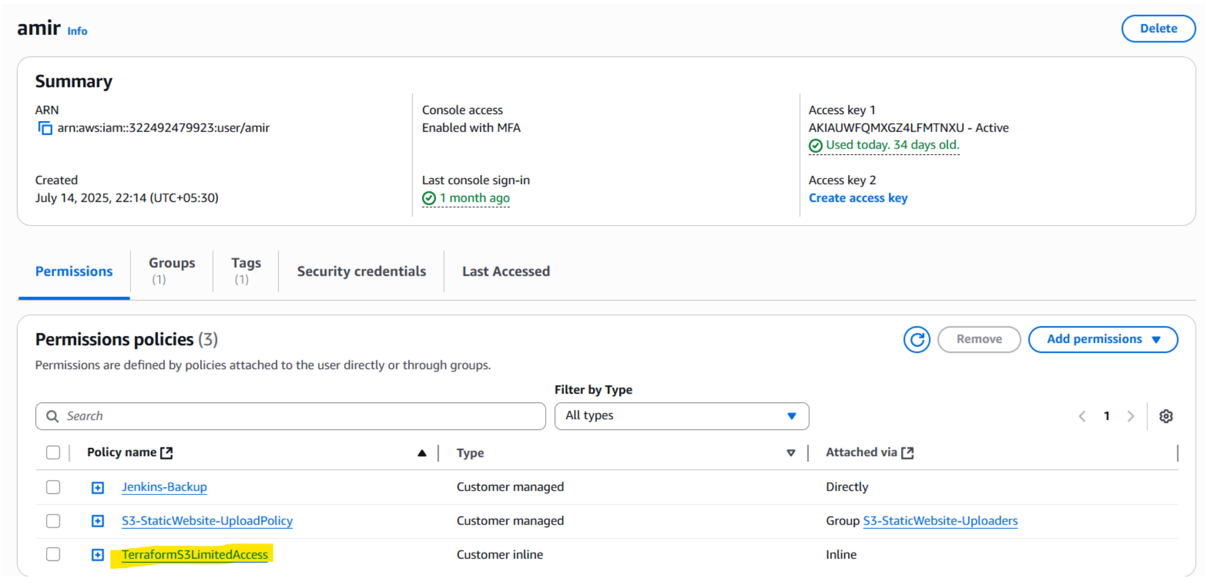
Using this Terraform configuration, the S3 bucket and associated objects are secure, server-side encryption and blocks all public access. Additionally, the project automates the upload of multiple files into the bucket while maintaining folder structures.

Key Skills: AWS (S3, IAM, CLI), Terraform (Infrastructure as Code), Linux (WSL),

Project Steps

Step 1: Create IAM User and Attach S3 Access Policy

An **IAM user** was created in AWS with least privilege access. (**IAM (Identity and Access Management)** users are entities that allow authentication and access to AWS services).



Summary

ARN: arn:aws:iam::322492479923:user/amir
Created: July 14, 2025, 22:14 (UTC+05:30)
Console access: Enabled with MFA
Last console sign-in: 1 month ago
Access key 1: AKIAUWFQMXGZ4LFMTNXU - Active
Used today. 34 days old.
Access key 2: Create access key

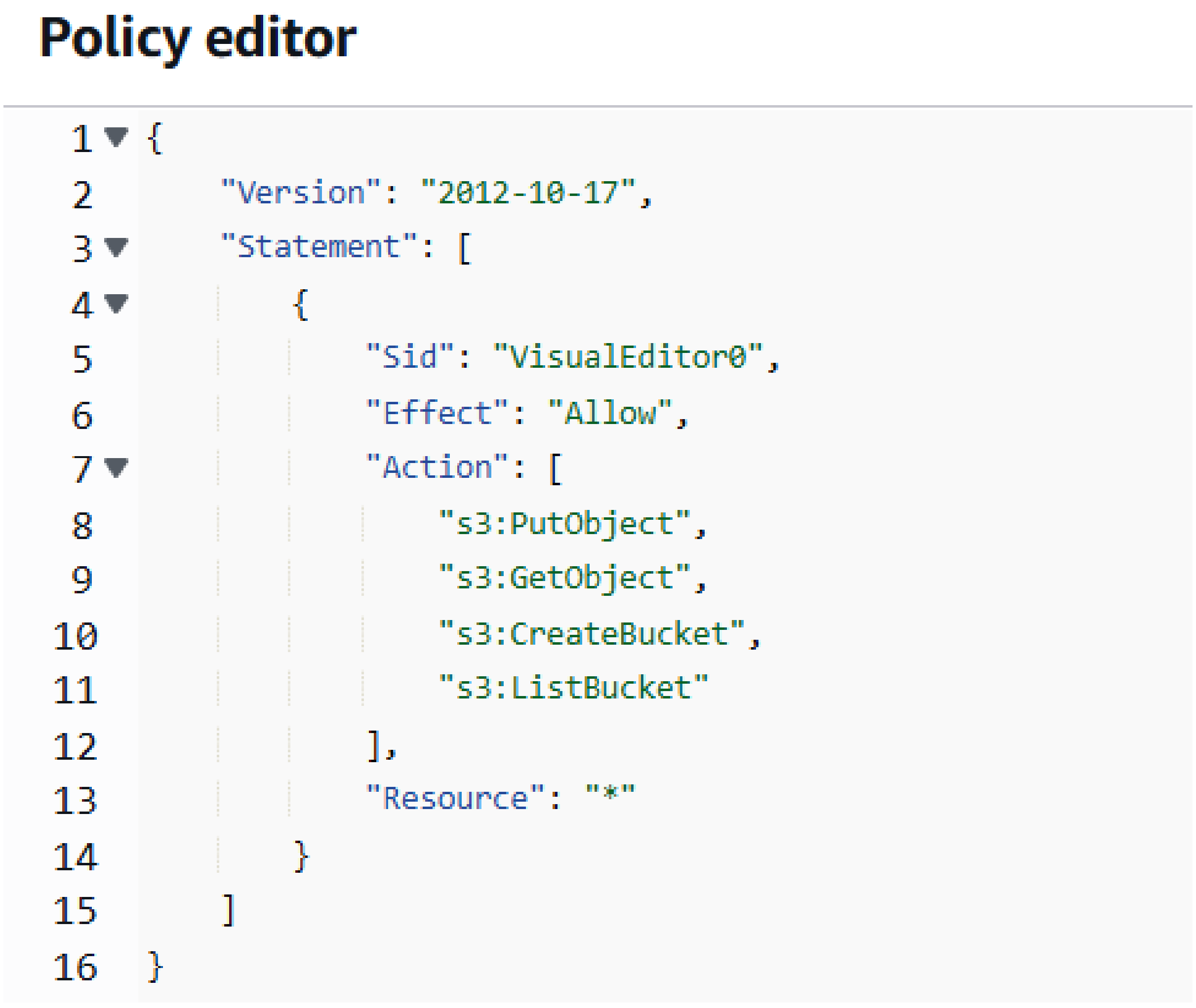
Permissions policies (3)
Permissions are defined by policies attached to the user directly or through groups.

Filter by Type: All types

Policy name	Type	Attached via
Jenkins-Backup	Customer managed	Directly
S3-StaticWebsite-UploadPolicy	Customer managed	Group S3-StaticWebsite-Uploaders
TerraformS3LimitedAccess	Customer inline	Inline

Figure 1: IAM User (amir)

The user was attached the **TerraformS3LimitedAccess** policy. (An **IAM policy** is a document that defines permissions (what actions a user, group, or role can perform on AWS resources)).



Policy editor

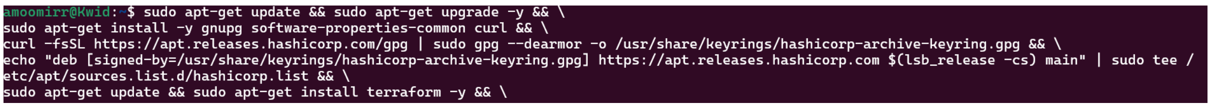
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:CreateBucket",
                "s3:ListBucket"
            ],
            "Resource": "*"
        }
    ]
}
```

Figure 2: IAM Policy (TerraformS3LimitedAccess)

Note :- Once the bucket is created change the resource permission to the specific Bucket's ARN to make it more secure.

Step 2: Install Terraform

Terraform was installed on WSL (Linux) to enable infrastructure provisioning. For Installation you must refer the official documentation page



```
amoomirr@Kwid:~$ sudo apt-get update && sudo apt-get upgrade -y && \
sudo apt-get install -y gnupg software-properties-common curl && \
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg && \
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /
etc/apt/sources.list.d/hashicorp.list && \
sudo apt-get update && sudo apt-get install terraform -y && \
```

Figure 3: Terraform Installation on WSL

To check if installed or not and also the version of it

```
terraform - -version
```

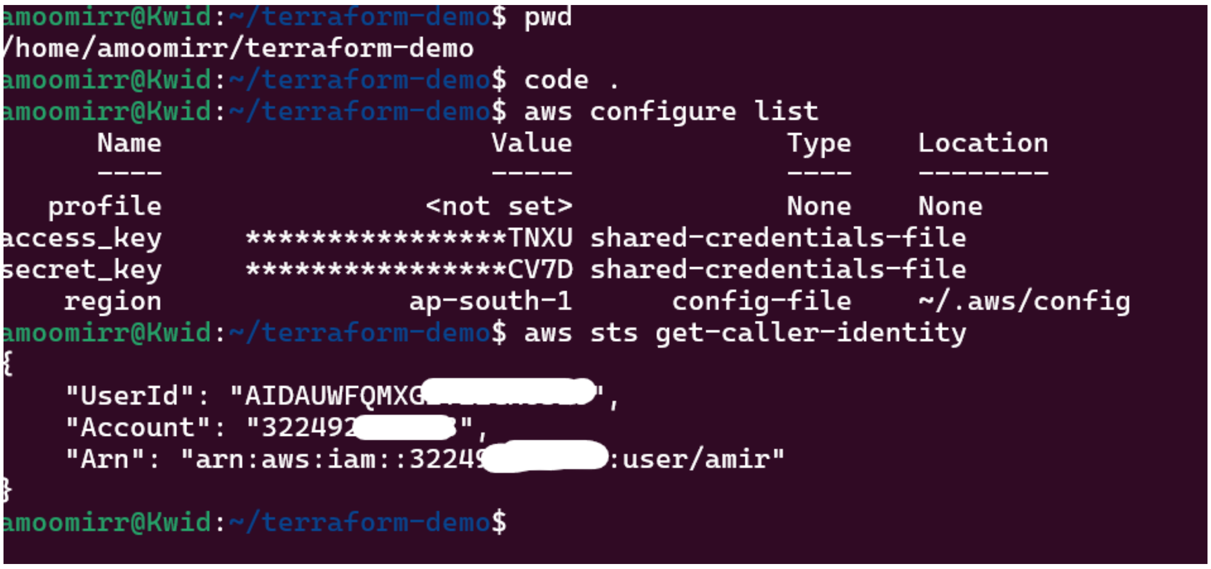
Step 3: Configure AWS CLI

AWS CLI was configured for the IAM user (amir) to enable Terraform to interact with AWS.(**AWS CLI** is a command-line tool that allows you to manage AWS services using commands)

```
-aws configure
```

Inputs required:

- AWS Access Key ID
- AWS Secret Access Key
- Default Region (e.g., ap-south-1)
- Output format (e.g., json)

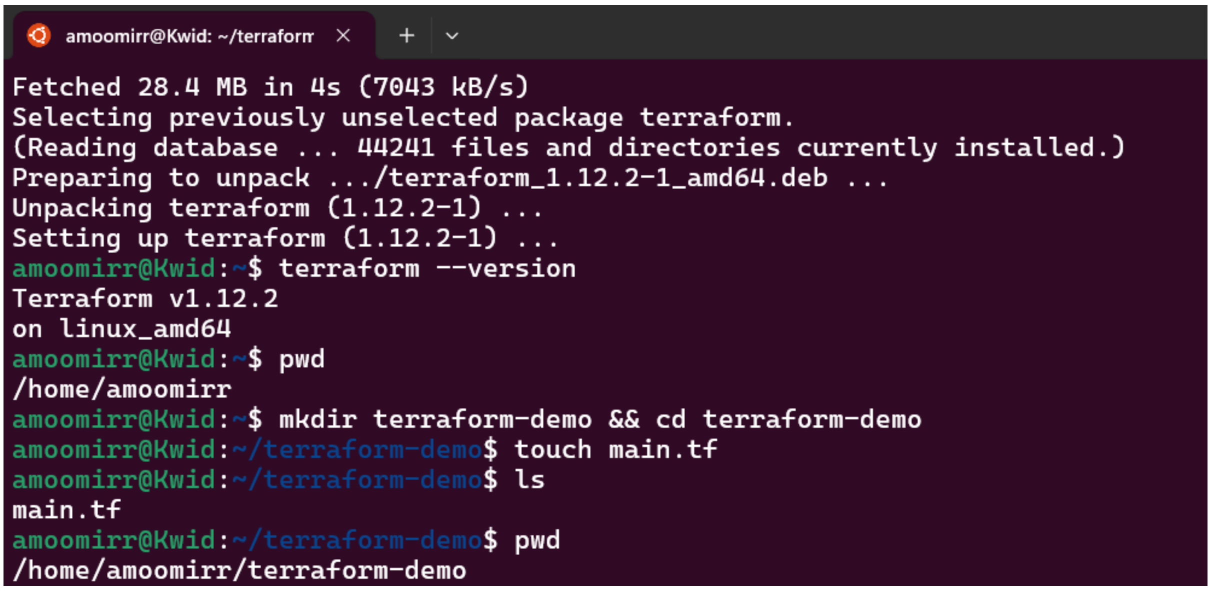


```
amoomirr@Kwid:~/terraform-demo$ pwd
/home/amoomirr/terraform-demo
amoomirr@Kwid:~/terraform-demo$ code .
amoomirr@Kwid:~/terraform-demo$ aws configure list
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************TNXU shared-credentials-file
secret_key     ****************CV7D shared-credentials-file
    region               ap-south-1      config-file    ~/.aws/config
amoomirr@Kwid:~/terraform-demo$ aws sts get-caller-identity
{
    "UserId": "AIDAUWFQMXG[redacted]",
    "Account": "322492[redacted]",
    "Arn": "arn:aws:iam::32249[redacted]:user/amir"
}
amoomirr@Kwid:~/terraform-demo$
```

Figure 4: AWS CLI Configuration

Step 4: Create Terraform Directory and main.tf File

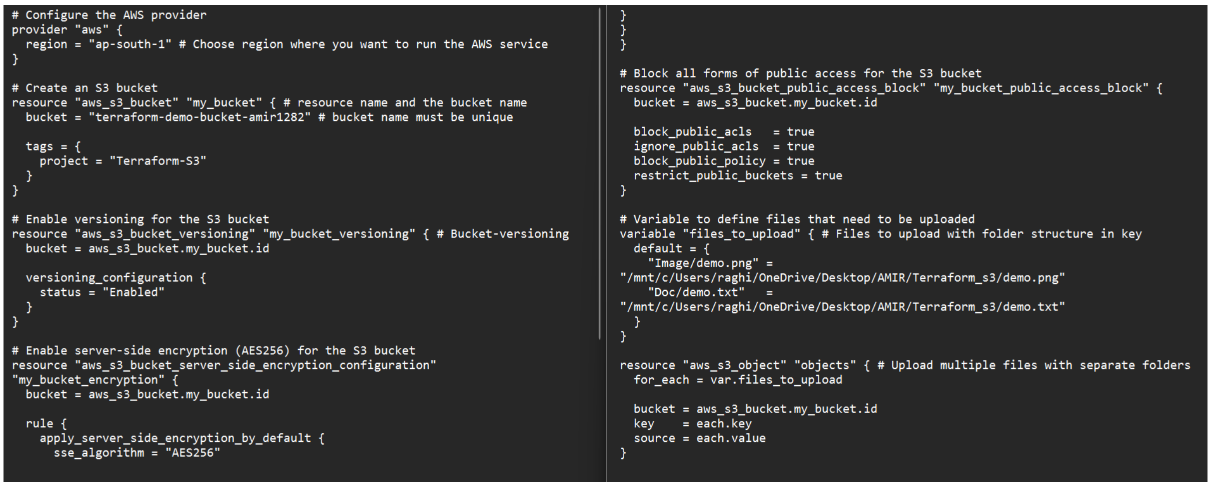
A Terraform directory was created and a `main.tf` file was added.



```
Fetched 28.4 MB in 4s (7043 kB/s)
Selecting previously unselected package terraform.
(Reading database ... 44241 files and directories currently installed.)
Preparing to unpack .../terraform_1.12.2-1_amd64.deb ...
Unpacking terraform (1.12.2-1) ...
Setting up terraform (1.12.2-1) ...
amoomirr@Kwid:~$ terraform --version
Terraform v1.12.2
on linux_amd64
amoomirr@Kwid:~$ pwd
/home/amoomirr
amoomirr@Kwid:~$ mkdir terraform-demo && cd terraform-demo
amoomirr@Kwid:~/terraform-demo$ touch main.tf
amoomirr@Kwid:~/terraform-demo$ ls
main.tf
amoomirr@Kwid:~/terraform-demo$ pwd
/home/amoomirr/terraform-demo
```

Figure 5: Terraform Directory and main.tf Creation

The `main.tf` file is the main configuration file where all Terraform resources (like S3 buckets and objects) are defined.



```
# Configure the AWS provider
provider "aws" {
  region = "ap-south-1" # Choose region where you want to run the AWS service
}

# Create an S3 bucket
resource "aws_s3_bucket" "my_bucket" { # resource name and the bucket name
  bucket = "terraform-demo-bucket-amir1282" # bucket name must be unique

  tags = {
    project = "Terraform-S3"
  }
}

# Enable versioning for the S3 bucket
resource "aws_s3_bucket_versioning" "my_bucket_versioning" { # Bucket-versioning
  bucket = aws_s3_bucket.my_bucket.id

  versioning_configuration {
    status = "Enabled"
  }
}

# Enable server-side encryption (AES256) for the S3 bucket
resource "aws_s3_bucket_server_side_encryption_configuration"
"my_bucket_encryption" {
  bucket = aws_s3_bucket.my_bucket.id

  rule {
    apply_server_side_encryption_by_default {
      sse_algorithm = "AES256"
```

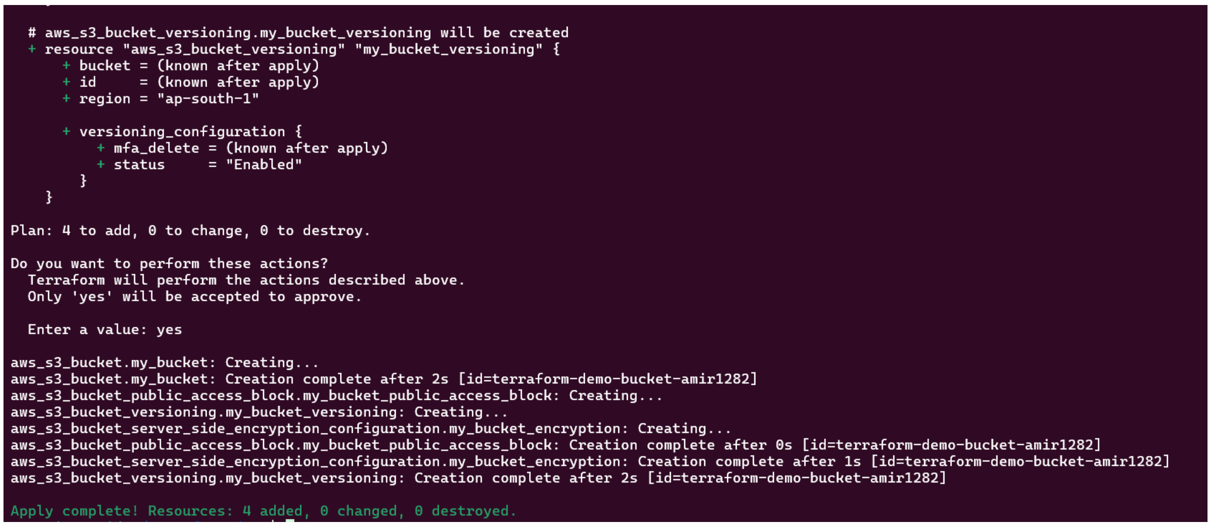
```
    }
  }
}

# Block all forms of public access for the S3 bucket
resource "aws_s3_bucket_public_access_block" "my_bucket_public_access_block" {
  bucket = aws_s3_bucket.my_bucket.id

  block_public_acls   = true
  ignore_public_acls  = true
  block_public_policy = true
  restrict_public_buckets = true
}

# Variable to define files that need to be uploaded
variable "files_to_upload" { # Files to upload with folder structure in key
  default = {
    "Image/demo.png" =
"/mnt/c/Users/raghi/OneDrive/Desktop/AMIR/Terraform_s3/demo.png"
    "Doc/demo.txt"   =
"/mnt/c/Users/raghi/OneDrive/Desktop/AMIR/Terraform_s3/demo.txt"
  }
}

resource "aws_s3_object" "objects" { # Upload multiple files with separate folders
  for_each = var.files_to_upload

  bucket = aws_s3_bucket.my_bucket.id
  key    = each.key
  source = each.value
}
```

Figure 6: Terraform Code (main.tf)

Step 5: Initialize Terraform and Apply Configuration

Terraform was initialized, plan was checked, and configuration applied.

- **terraform init**: Initializes the Terraform working directory and downloads required provider plugins.
- **terraform plan**: Generates an execution plan showing what actions Terraform will perform.
- **terraform apply**: Applies the changes required to reach the desired state of the configuration.

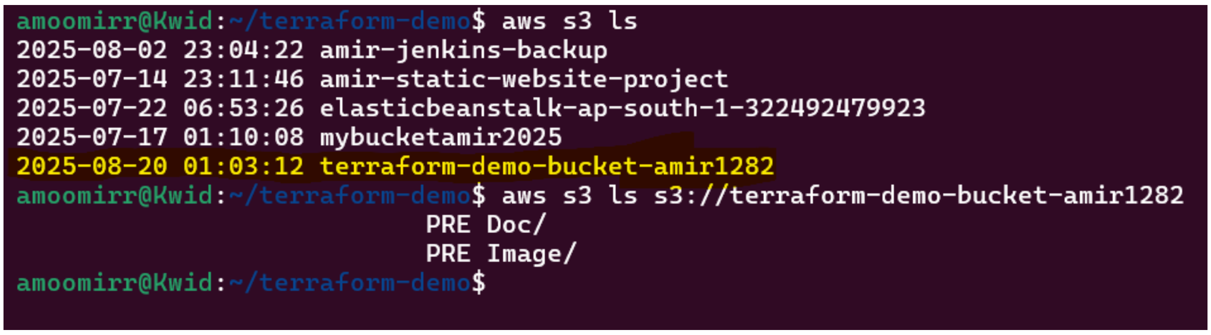


```
  # aws_s3_bucket_versioning.my_bucket_versioning will be created
  + resource "aws_s3_bucket_versioning" "my_bucket_versioning" {
      + bucket = (known after apply)
      + id     = (known after apply)
      + region = "ap-south-1"

      + versioning_configuration {
          + mfa_delete = (known after apply)
          + status     = "Enabled"
        }
    }

Plan: 4 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 2s [id=terraform-demo-bucket-amir1282]
aws_s3_bucket_public_access_block.my_bucket_public_access_block: Creating...
aws_s3_bucket_versioning.my_bucket_versioning: Creating...
aws_s3_bucket_server_side_encryption_configuration.my_bucket_encryption: Creating...
aws_s3_bucket_public_access_block.my_bucket_public_access_block: Creation complete after 0s [id=terraform-demo-bucket-amir1282]
aws_s3_bucket_server_side_encryption_configuration.my_bucket_encryption: Creation complete after 1s [id=terraform-demo-bucket-amir1282]
aws_s3_bucket_versioning.my_bucket_versioning: Creation complete after 2s [id=terraform-demo-bucket-amir1282]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
```

Figure 7: Resources Created from Terraform Apply

Step 6: Verify S3 Bucket and Objects

The S3 bucket and objects were verified using AWS CLI. An **S3 bucket** is a container in AWS for storing objects (files). **Objects** are the actual files/data stored inside S3 buckets.

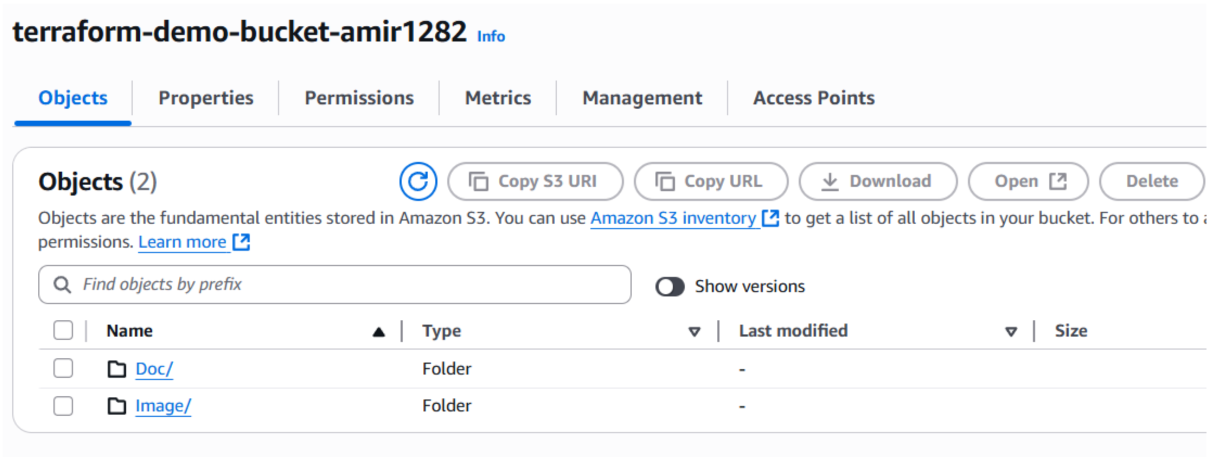


```
amoomirr@Kwid:~/terraform-demo$ aws s3 ls
2025-08-02 23:04:22 amir-jenkins-backup
2025-07-14 23:11:46 amir-static-website-project
2025-07-22 06:53:26 elasticbeanstalk-ap-south-1-322492479923
2025-07-17 01:10:08 mybucketamir2025
2025-08-20 01:03:12 terraform-demo-bucket-amir1282
amoomirr@Kwid:~/terraform-demo$ aws s3 ls s3://terraform-demo-bucket-amir1282
                           PRE Doc/
                           PRE Image/
amoomirr@Kwid:~/terraform-demo$
```

Figure 8: Verification of S3 Bucket and Uploaded Objects

Conclusion

- Configured IAM user with necessary permissions for secure access.
- Successfully created an AWS S3 bucket and uploaded objects using Terraform.
- Ensured bucket security via encryption and public access restrictions.
- Demonstrated end-to-end Infrastructure as Code (IaC) deployment.
- Verified resources via AWS CLI ensuring proper provisioning.



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to
permissions. Learn more

	Name	Type	Last modified	Size
☐	Doc/	Folder	-	
☐	Image/	Folder	-	

Figure 9: AWS Management Console