Project Title: Infrastructure Automation using Terraform (AWS)

1. Introduction

This project demonstrates the use of Terraform to automate infrastructure deployment on Amazon Web Services (AWS). The focus was on provisioning EC2 instances and S3 buckets in a repeatable and automated manner. Terraform provides Infrastructure as Code (IaC), allowing reliable resource creation and destruction with minimal manual effort.

2. Tools Used

- **AWS EC2**: For provisioning virtual servers (control-plane and worker nodes).
- **AWS IAM:** To generate access keys and configure CLI credentials.
- **Terraform**: For writing infrastructure as code and automating lifecycle management.
- **AWS CLI**: To configure keys and manage connections with AWS account.
- **GitHub Repo**: For storing and version-controlling Terraform code.

3. Procedure

Step 1: Setup AWS Environment

Created an AWS EC2 instance for working environment. Used IAM to generate Access Key and Secret Key. Configured AWS credentials in terminal using:

```
aws configure
Selected region :eu-north-1
```

Step 2: Install Terraform

- Installed Terraform on the EC2 instance.
- Verified installation using:

```
terraform -version
```

Step 3: Write Terraform Code

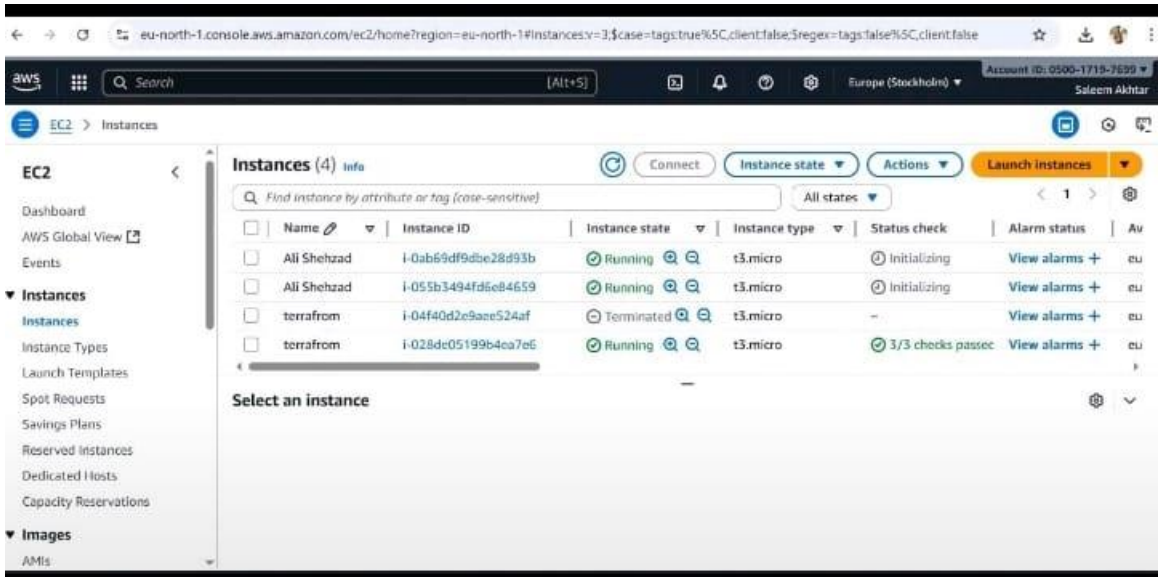
- Created a file named **EC2.tf**
- Defined AWS provider and EC2 resource configuration.
- Configured count = 2 to automatically create two nodes.

Step 4: Execute Terraform Commands

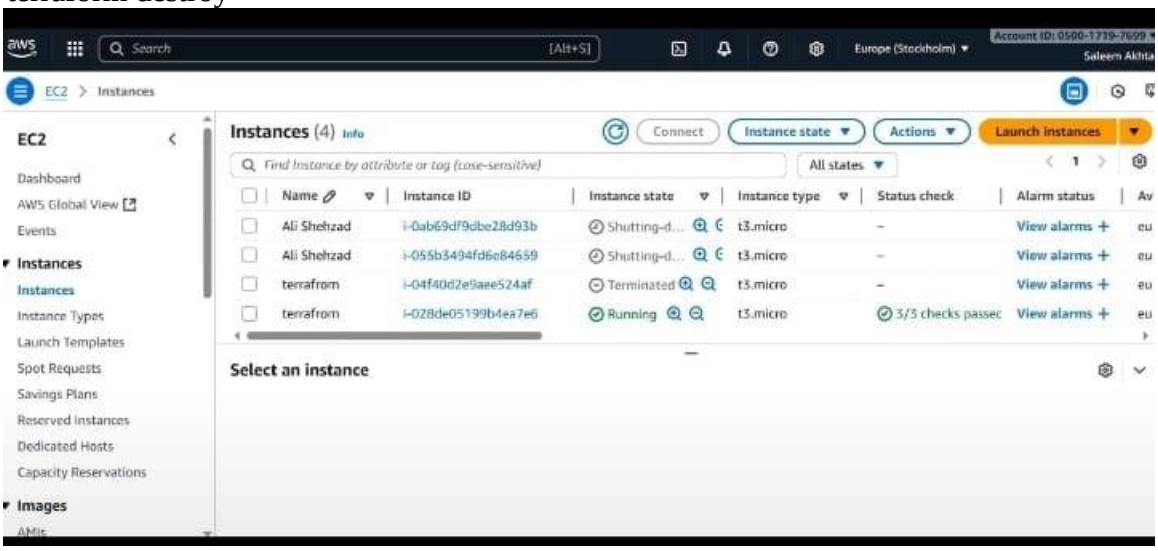
- Initialized Terraform:

```
terraform init
```

- **Planned execution:**
  terraform plan
- **Applied configuration**:
  terraform apply
- **Verified successful EC2 instance creation.**



- **Destroyed resources after testing:**
  terraform destroy



Step 5: Create and Manage S3 Bucket

- Wrote Terraform configuration to provision an **S3.tf** file.
- Applied configuration to create the bucket.
- Destroyed the bucket using terraform destroy.

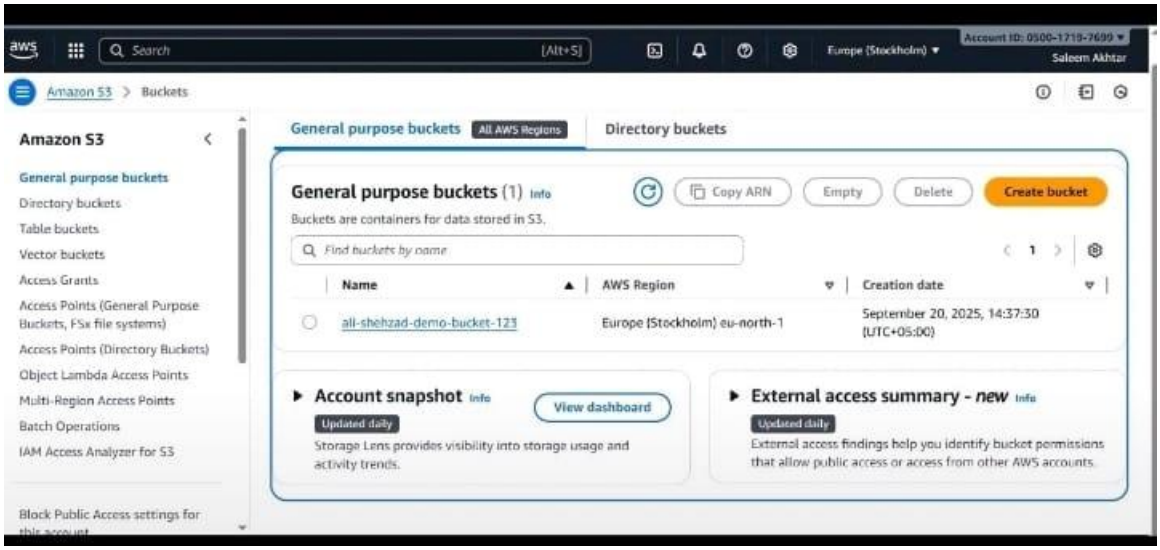
Planned execution:

terraform plan

- Applied configuration:

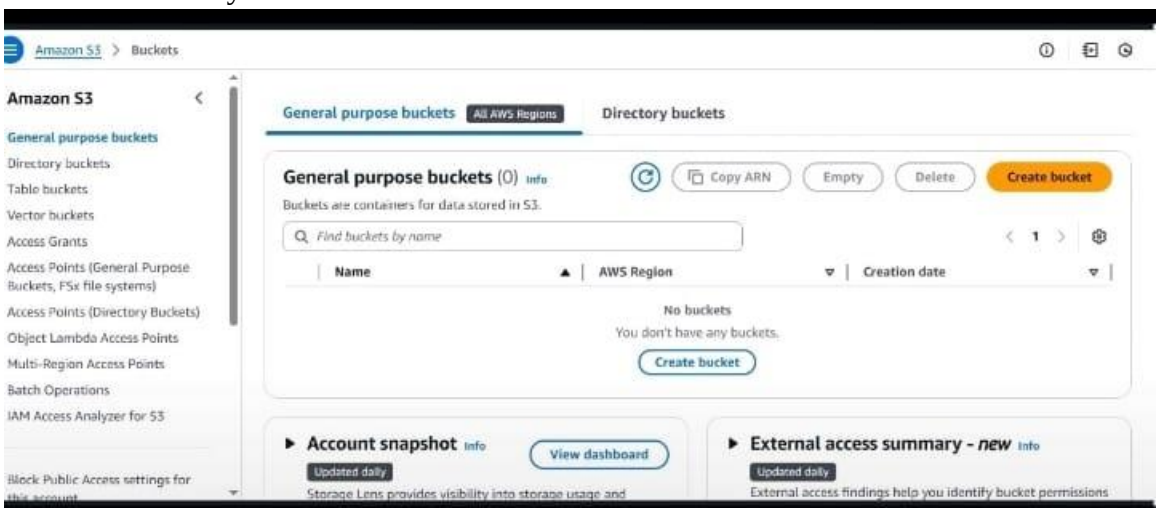
terraform apply

Verified successful S3instance creation.



Destroyed resources after testing:

terraform destroy



5. Conclusion

This project demonstrated how Terraform automates infrastructure provisioning on AWS:

- EC2 instances and S3 buckets were created, managed, and destroyed using IaC.
- Lifecycle management was handled with Terraform commands (init, plan, apply, destroy).
- The process ensured repeatable, idempotent, and reliable deployments.
- By using Terraform, infrastructure management becomes scalable, automated, and efficient, reducing human error and saving time.