Advanced DevOps Lab
Experiment 6

Aim: To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform. (S3 bucket or Docker) fdp

Theory: Infrastructure as Code (IaC) is a practice where infrastructure is provisioned and managed using code rather than manual processes. Terraform is a widely-used IaC tool that allows users to define infrastructure resources in a declarative configuration language.

Key Concepts:

1. Providers: Terraform supports multiple cloud providers, allowing users to manage resources across AWS, GCP, Azure, and DigitalOcean. Each provider has its own set of APIs and resource types.
2. Resources: Resources are the fundamental components of a Terraform configuration. For example, an S3 bucket in AWS or a Docker container in a container orchestration platform like ECS.
3. State Management: Terraform maintains a state file that maps the configurations defined in code to the actual infrastructure. This state file is crucial for managing changes and ensuring the desired state is achieved.
4. Execution Plans: Before making changes, Terraform generates an execution plan that outlines the proposed actions, allowing users to review and approve changes before they are applied.
5. Modules: Terraform supports reusable code blocks called modules, which allow for organizing and sharing configurations, promoting best practices, and reducing duplication.

Step 1: Write a Terraform Script in Atom for creating S3 Bucket on Amazon AWS

s3.tf

```
resource = "aws_s3_bucket" "saachi" {
    bucket = "saachiexp"

    tags = {
        Name        = "My bucket"
        Environment = " Dev"
    }
}
```

Create a new provider.tf file and write the following contents into it.

```
provider "aws" {
    access_key="ASIA5S7ZXZ23BN4IZHOD"
    secret_key="baqB3SfAcW3y7o1YbZhjDoxenmVY+mvoONvuSCEO"
    token="IQoJb3JpZ2luX2VjEKT//////////wEaCXVzLXdlc3QtMiJGMEQCIEaFE9so3lDUMYehN30sTc8zhPyRx/
    Tnkbc9tqBp3KiJAiBHv4mmIk+RgVRX/ikYOsJqcvTccwIHNhLfmvDw/VEO/Cq/Agid/////////
    8BEAEaDDkzNDE0MjI2NzA2MiIM4NCxNbH/hirIDH9XKpMCRGaGm41QxN5u
    +vhCZUeBBxXuhwitP5fMiZWtvQRUnsDngDnCui0usAFQuhColFBfMS908N3dFfKmQff/
    UAJoJA1Gm2JpBV9BBLgslwg1N7l86gH6ls6VuDiHxQfDHJ01ktRw9T8Gttsz48Fe+E+MbbNDZTv3PqIQUu4WJUvXSr6L4baIkiBLpxk
    +rezP+ZIeiDesZJGGxFZOsvkSnkhZA8/q/JZPy9CTOdemdiVFrOQPFOw8WcMczEXGzzS1FlM9L26qCJmWI
    +4JEGrRg1EsYWCqdt0ChVGXcmdUcso33SsLLbrnQU0KGHf/
    gVWYnhTf49OEinHiIGvToGTXpNprhFNPpsfO6X5vbWgcqyCTOiiwyfQwg7jrtQY6ngE60oH0tDowGVZnTbWrJbnYw256RoDchL8Mi83Iunt
    jX34WRmyZdi6ZvbndEibB+phe33Qr8Ge+k/xiG+GgC8BELqTK7/yMH3ItsNH/4pS2opjQNrkBa5xsK3nYKmLO/
    F9tndLf5hOn6WzDq7q6L2O8YnNNYVxBTcuD4BQnvHR//8tHRvJpvF6IrwoL7Mgs8J7+S0jnx46GoVqSmM5lTg=="
    region = "us-east-1"
}
```

Save both the files in same directory Terraform_Scripts/S3

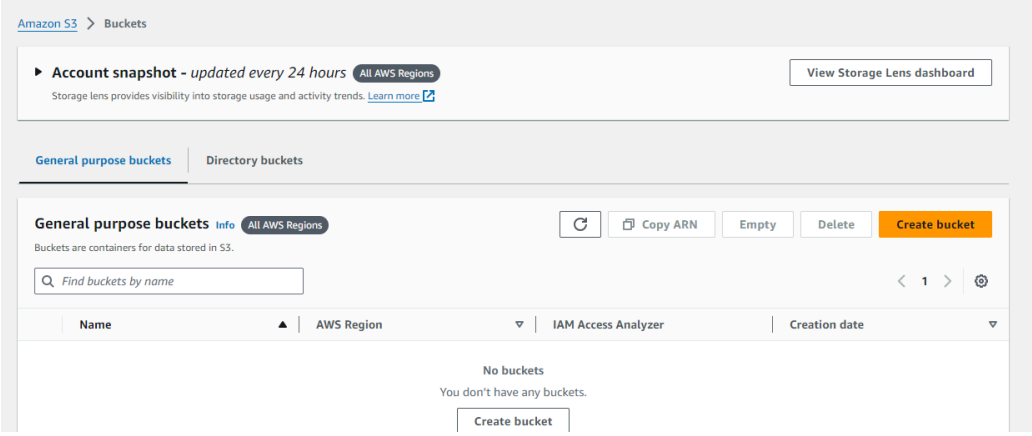
Step 2: Open Command Prompt and go to Terraform_Script\S3 directory where our .tf files are stored

```
C:\Terraform Script\S3>dir
 Volume in drive C has no label.
 Volume Serial Number is 885D-B8E0

 Directory of C:\Terraform Script\S3

08-08-2024  14:42    <DIR>          .
08-08-2024  14:01    <DIR>          ..
08-08-2024  14:49               147 provider.tf
08-08-2024  14:46               154 s3.tf
               2 File(s)            301 bytes
               2 Dir(s)  84,993,937,408 bytes free

C:\Terraform Script\S3>
```



Step 3: Execute Terraform Init command to initialize the resources

```
C:\Terraform_Scripts\S3>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.62.0

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.
```

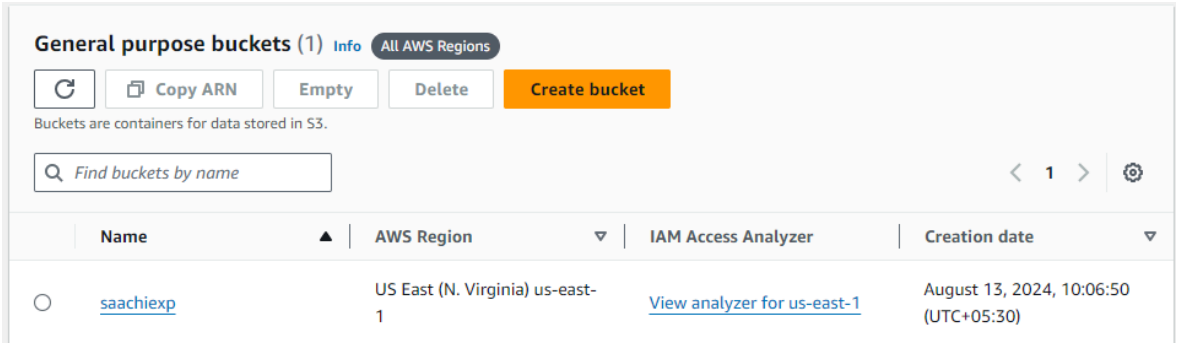
Step 4: Execute Terraform plan to see the available resources

```
C:\Terraform_Scripts\S3>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_s3_bucket.saachi will be created
  + resource "aws_s3_bucket" "saachi" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "saachiexp"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "Dev"
          + "Name"        = "My Bucket"
        }
      + tags_all                    = {
```



Step 5: Execute Terraform apply to apply the configuration, which will automatically create an S3 bucket based on our configuration.

```
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.saachi: Creating...
aws_s3_bucket.saachi: Creation complete after 5s [id=saachiexp]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

```
PS C:\Terraform_Scripts\s3> set "ASIA5S7ZXZ23BN4IZHOD"
PS C:\Terraform_Scripts\s3> set "baqB3SfAcW3y7o1YbZhjDoxenmVY+mvoONvuSCEO"
```

```
[cloudshell-user@ip-10-140-123-113 ~]$ export AWS_ACCESS_KEY_ID="ASIA5S7ZXZ23BN4IZHOD"
[cloudshell-user@ip-10-140-123-113 ~]$ export AWS_SECRET_ACCESS_KEY="baqB3SfAcW3y7o1YbZhjDoxenmVY+mvoONvuSCEO"
[cloudshell-user@ip-10-140-123-113 ~]$
```

Step 6: Execute Terraform destroy to delete the configuration, which will automatically delete an EC2 instance

```
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.saachi: Destroying... [id=saachiexp]
aws_s3_bucket.saachi: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
```

Conclusion: Terraform streamlines infrastructure management across cloud providers, offering flexibility, automation, and collaboration. By treating infrastructure as code, it enhances efficiency and scalability in cloud operations.