Vivekanand Education Society's

Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai,, Approved by AICTE & Recognized by Govt. of Maharashtra

Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology   A.Y. 2024-25

Advance DevOps Lab
Experiment 06

Aim: To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.

Roll No.	22
Name	Sarthak Harade
Class	D15B
Subject	Advance DevOps Lab
LO Mapped	LO1: To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements. LO3: To apply best practices for managing infrastructure as code environments and use terraform to define and deploy cloud infrastructure.
Grade:	

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

THEORY:

Terraform is a powerful Infrastructure as Code (IaC) tool that allows users to define, build, change, and manage cloud infrastructure across various providers like AWS, Google Cloud, Microsoft Azure, and DigitalOcean. By using Terraform, infrastructure is treated as code, enabling automation, consistency, and version control in managing resources such as S3 buckets, EC2 instances, and other cloud components.

Creating an S3 Bucket with Terraform

When using Terraform to create an S3 bucket on AWS, the process involves defining the desired state of the infrastructure through configuration files written in HashiCorp Configuration Language (HCL). These files specify the cloud provider, resources, and other configurations required to set up the infrastructure.
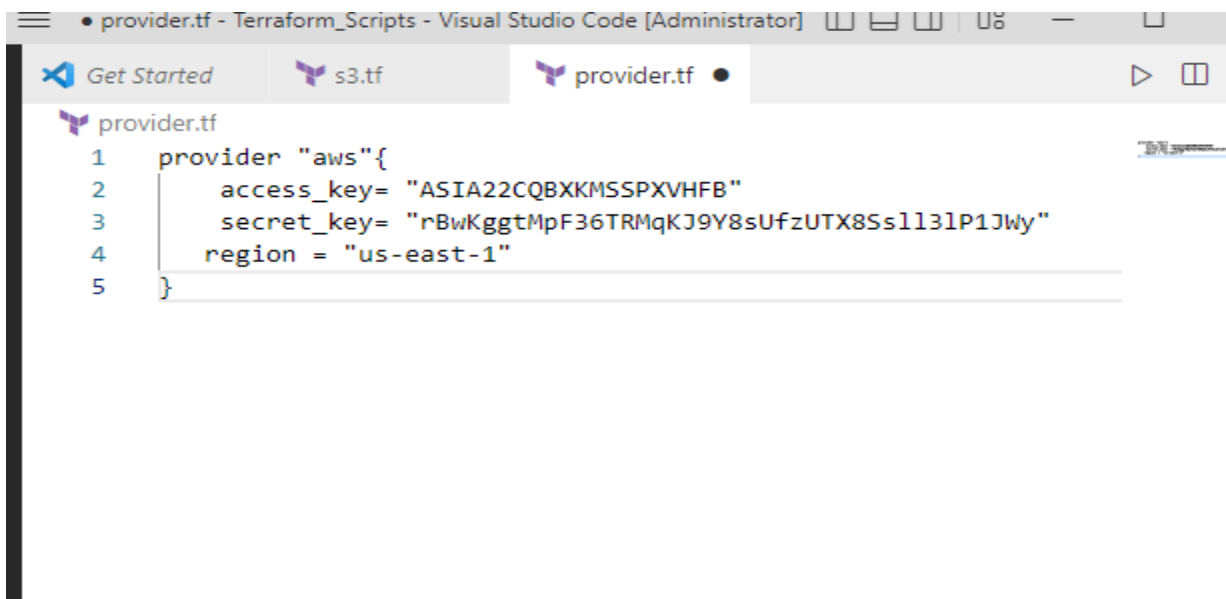
1. Provider Configuration: Terraform uses providers to interact with different cloud platforms. In this case, the AWS provider is configured with the necessary credentials, such as the Access Key ID and Secret Access Key, to authenticate and authorize Terraform's actions on the AWS cloud.

2. Resource Definition: The core of Terraform's functionality lies in its ability to define and manage resources. For creating an S3 bucket, a resource block is used to specify the properties of the bucket, such as its name, region, and access control settings. Terraform then ensures that the specified bucket is created with these properties.

3. Infrastructure as Code (IaC): By writing the configuration in code, Terraform enables the infrastructure to be versioned, shared, and reused across different environments. This approach not only improves collaboration among teams but also ensures that the infrastructure can be easily replicated or modified as needed.

4. Lifecycle Management: Terraform's lifecycle commands—`init`, `plan`, `apply`, and `destroy`—allow users to manage the entire lifecycle of their infrastructure. These commands initialize the environment, preview changes, apply the configuration, and eventually destroy the infrastructure when it is no longer needed. This level of control ensures that resources are managed efficiently, avoiding unnecessary costs and maintaining an organized cloud environment.

5. State Management: Terraform maintains a state file that tracks the current state of the managed infrastructure. This state file is crucial for determining what changes need to be applied when updating the infrastructure. It ensures that the live infrastructure remains in sync with the configuration files, allowing Terraform to make precise and minimal changes.

Write a Terraform Script for creating S3 Bucket on Amazon AWS and provider.tf file. Save both the files in the same directory Terraform along with the terraform application.

s3.tf:

```
resource "aws_s3_bucket" "Sarthak Harade" {
    bucket = "My_Bucket"

    tags = {
        Name        = "Sarthak Bucket"
        Environment = "Dev"
    }
}
```

• provider.tf - Terraform_Scripts - Visual Studio Code [Administrator]

Get Started   s3.tf   provider.tf •

provider.tf

```
1    provider "aws"{
2        access_key= "ASIA22CQBXKMSSPXVHFB"
3        secret_key= "rBwKggtMpF36TRMqKJ9Y8sUfzUTX8Ssl131P1JWy"
4      region = "us-east-1"
5    }
```

Go to the Terraform directory where both the files are saved in the command prompt and execute Terraform Init command to initialize the resources.

```
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin>cd C:\Terraform_Scripts\S3

C:\Terraform_Scripts\S3>dir
 Volume in drive C has no label.
 Volume Serial Number is CCC0-98D1

 Directory of C:\Terraform_Scripts\S3

08/13/2024  09:42 AM    <DIR>          .
08/13/2024  09:42 AM    <DIR>          ..
08/13/2024  09:41 AM               143 provider.tf
08/13/2024  09:32 AM               174 s3.tf
               2 File(s)            317 bytes
               2 Dir(s)   4,773,441,536 bytes free

C:\Terraform_Scripts\S3>_
```

```
             2 Dir(s)   4,787,556,352 bytes free

C:\Terraform_Scripts\S3>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.62.0...
- Installed hashicorp/aws v5.62.0 (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.

C:\Terraform_Scripts\S3>
```

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.sarthak will be created
  + resource "aws_s3_bucket" "sarthak" {
      + acceleration_status            = (known after apply)
      + acl                            = (known after apply)
      + arn                            = (known after apply)
      + bucket                         = "My_Bossbabe"
      + 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"        = "Sarthak Bucket"
        }
      + tags_all                       = {
          + "Environment" = "Dev"
          + "Name"        = "Sarthak Bucket"
        }
      + 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)
```

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

```
      + 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"        = "Sarthak Bucket"
        }
      + tags_all                       = {
          + "Environment" = "Dev"
          + "Name"        = "Sarthak Bucket"
        }
      + 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.
```

Execute Terraform destroy to delete the configuration, which will automatically delete an EC2 instance.

```
        }
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.sarthak: Creating...
aws_s3_bucket.sarthak: Creation complete after 6s [id=my-sarthak-bucket-123]

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

C:\Terraform_Scripts\S3>
```

▶ **Account snapshot** - *updated every 24 hours* `All AWS Regions`   View Storage Lens dashboard

Storage lens provides visibility into storage usage and activity trends. Learn more ↗

General purpose buckets   Directory buckets

General purpose buckets (2) Info `All AWS Regions`

| ↻ | ⧉ Copy ARN | Empty | Delete | **Create bucket** |

Buckets are containers for data stored in S3.

🔍 Find buckets by name   ‹ 1 › ⚙

	Name	▲	AWS Region	▽	IAM Access Analyzer		Creation date	▽
○	elasticbeanstalk-us-east-1-743197227673		US East (N. Virginia) us-east-1		View analyzer for us-east-1		July 30, 2024, 09:30:09 (UTC+05:45)	
○	my-sarthak-bucket-123		US East (N. Virginia) us-east-1		View analyzer for us-east-1		August 13, 2024, 10:37:39 (UTC+05:45)	

```
      - id          = "bebf57127bdaf70f3f04ef3996d2a8b4f5f64bda2655699e3da691e60f62049a" -> 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.sarthak: Destroying... [id=my-sarthak-bucket-123]
aws_s3_bucket.sarthak: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.

C:\Terraform_Scripts\S3>
```
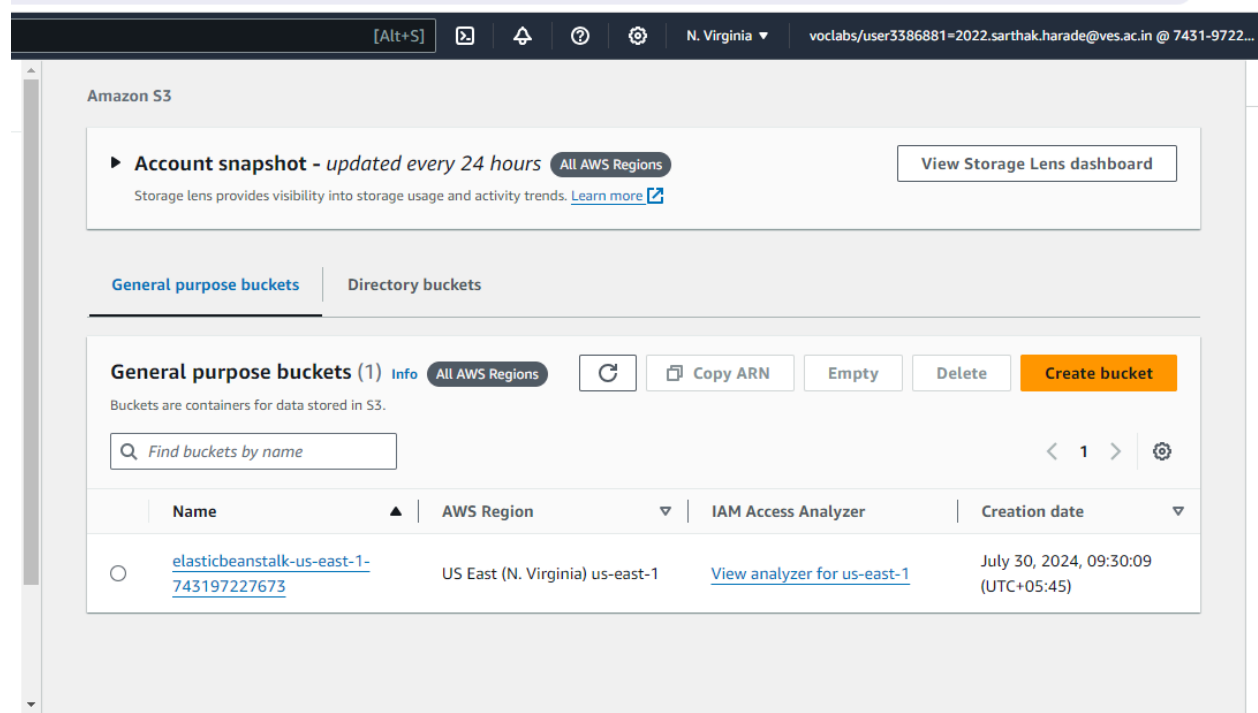
Delete the IAM user which was created earlier.

CONCLUSION : Terraform streamlines the process of managing cloud infrastructure by treating it as code, enabling automation and consistency across different cloud platforms. By using Terraform, you can efficiently create, modify, and destroy resources such as S3 buckets, ensuring a more organized and controlled approach to cloud management. Understanding these concepts is key to leveraging Terraform for advanced DevOps practices.