



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.	53
Name	Aryan Deepak Saraf
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:

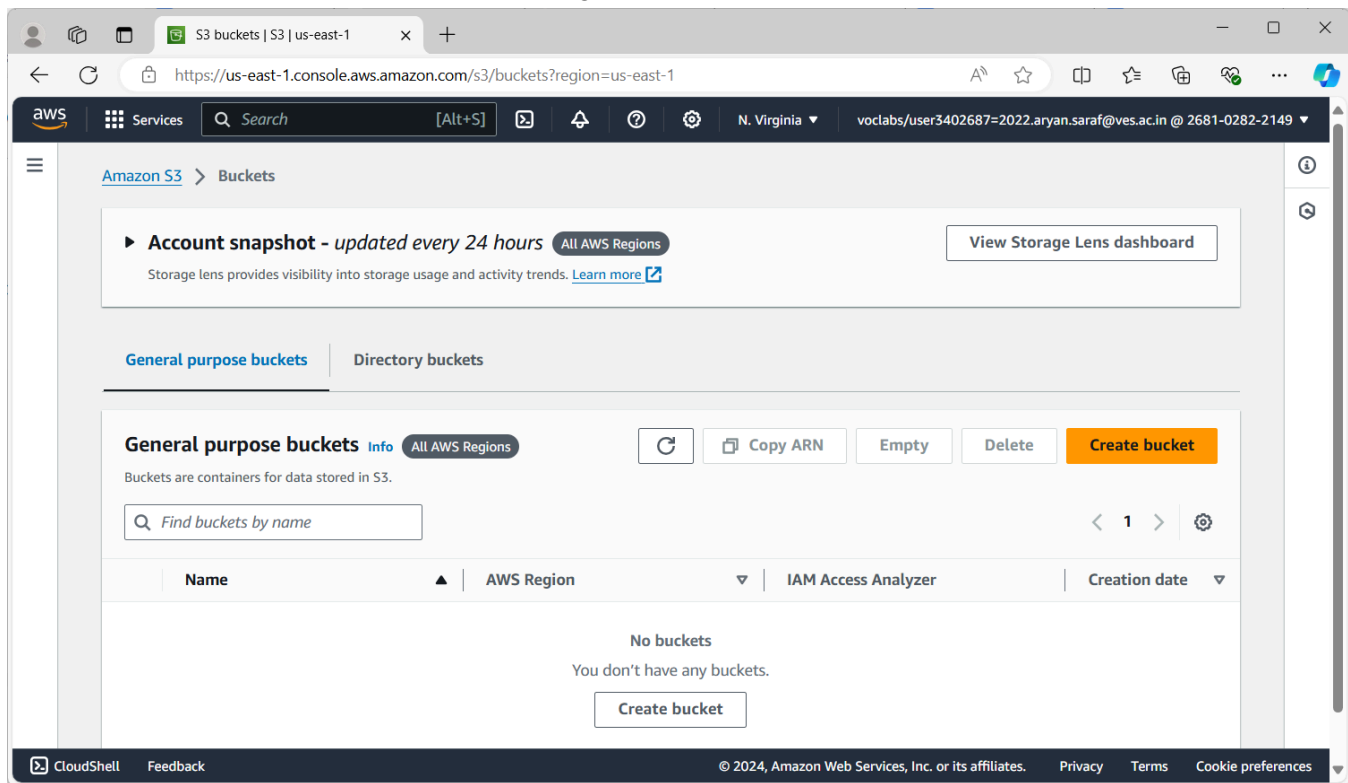
Terraform is a powerful Infrastructure as Code (IaC) tool that allows users to define, build, change, and manage cloud infrastructure across various providers like AWS, Google Cloud, Microsoft Azure, and DigitalOcean. By using Terraform, infrastructure is treated as code, enabling automation, consistency,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

Creating an S3 Bucket with Terraform

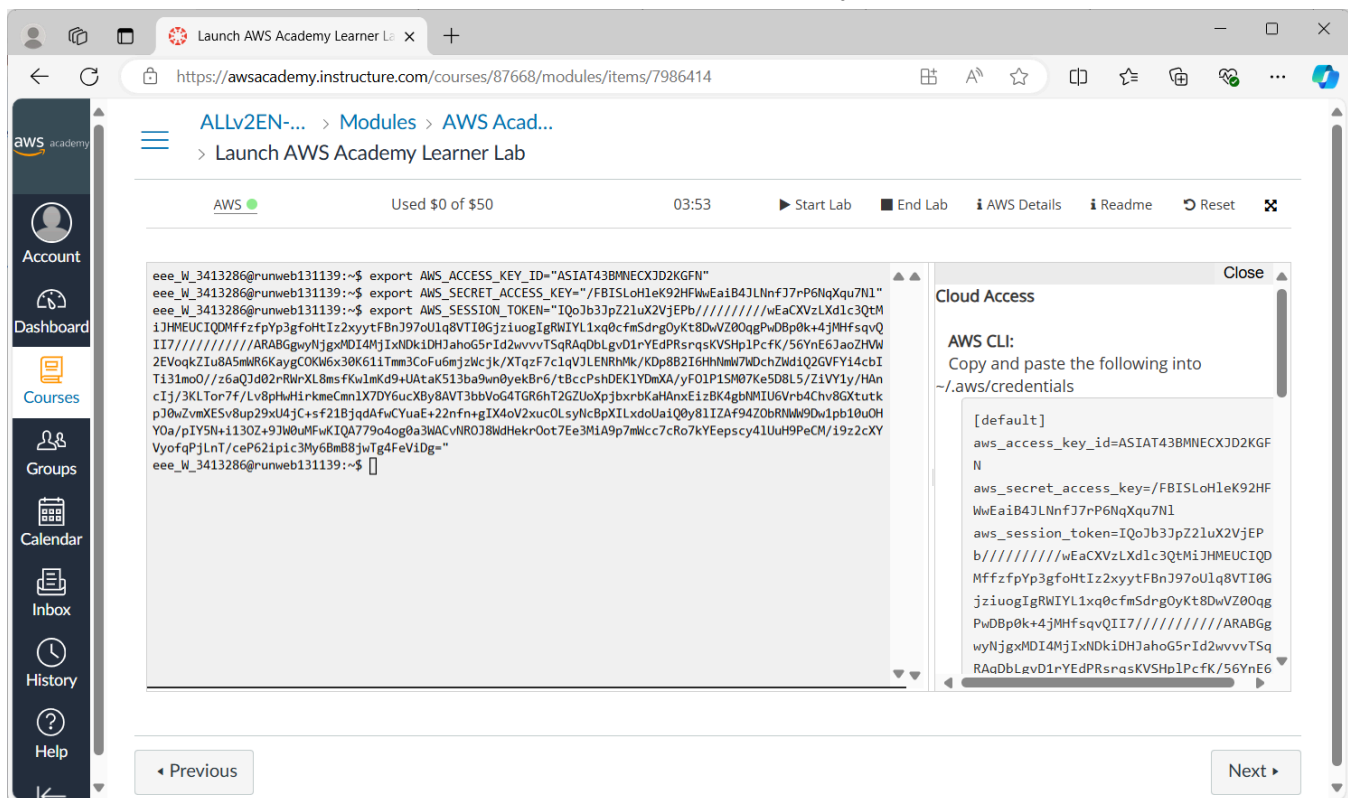
When using Terraform to create an S3 bucket on AWS, the process involves defining the desired state of the infrastructure through configuration files written in HashiCorp Configuration Language (HCL). These files specify the cloud provider, resources, and other configurations required to set up the infrastructure.

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.

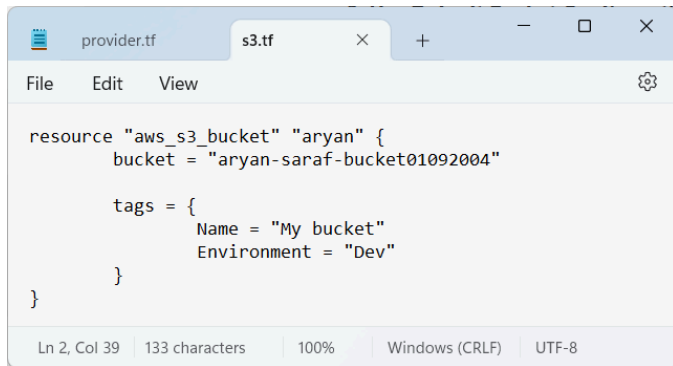
AWS S3 bucket dashboard before performing the experiment.



Export all the three credentials from the CLI of the AWS Academy Learner Lab.



Write a Terraform Script for creating S3 Bucket on Amazon AWS.



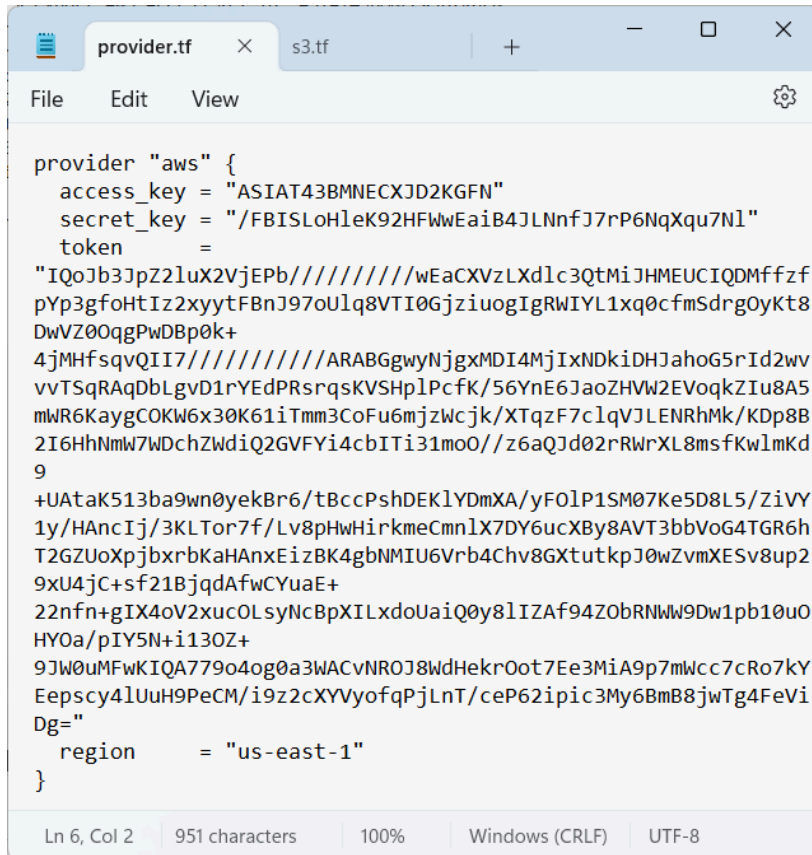
```

resource "aws_s3_bucket" "aryan" {
  bucket = "aryan-saraf-bucket01092004"

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

```

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

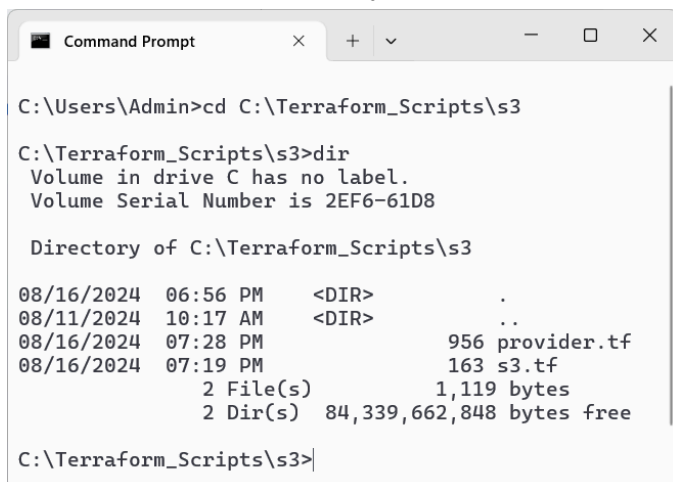


```

provider "aws" {
  access_key = "ASIAT43BMNECXJD2KGFN"
  secret_key = "/FBISLoHleK92HFwEaiB4JLNnfJ7rP6NqXqu7N1"
  token      = "IQoJb3JpZ2luX2VjEPb////////wEaCXVzLXd1c3QtMiJHMEUCIQDMffzf
pYp3gfoHtIz2xytFBnJ97oUlq8VTI0GjzIuogIgrWIYL1xq0cfmSdrgOyKt8
DwVZ00qgPwDBp0k+
4jMHfsqvQII7////////ARABGgwyNjgxMDI4MjIxNDkiDHJahoG5rId2wv
vVTSqRAqDbLgVd1rYEDPRsrqsKVSHp1PcFK/56YnE6JaoZHVW2EVoqkZIU8A5
mWR6KaygCOKW6x30K61iTmm3CoFu6mjzWcjk/XTqzF7clqVJLENRhMk/KDp8B
2I6HhNmW7WDchZwdiQ2GVFYi4cbITi31mo0//z6aQJd02rRWrXL8msfKwlmKd
9
+UAtaK513ba9wn0yekBr6/tBccPshDEKlYDmXA/yF0lP1SM07Ke5D8L5/ZiVY
1y/HancIj/3KLTor7f/Lv8pHwHirkmeCmnlX7DY6ucXBy8AVT3bbVoG4TGR6h
T2GZUoXpjbxbkKaHAnxEizBK4gbNMIU6Vrb4Chv8GXTutkpJ0wZvmXESv8up2
9xU4jC+sf21BjqdAfwCYuaE+
22nfn+gIX4ov2xucOLsyNcBpXILxdoUaiQ0y8lIZAf94Z0bRNWW9Dw1pb10uO
HY0a/pIY5N+i130Z+
9JW0uMFwKIQA779o4og0a3WACvNROJ8WdHekrOot7Ee3MiA9p7mWcc7cRo7kY
Eepscy4lUuH9PeCM/i9z2cXYVyoFqPjLnT/ceP62ipic3My6BmB8jwTg4FeVi
Dg="
  region      = "us-east-1"
}

```

Save both the files in the same directory Terraform_Scripts/S3. Open Command Prompt and go to Terraform_Script\S3 directory where our .tf files are stored.



```

C:\Users\Admin>cd C:\Terraform_Scripts\s3

C:\Terraform_Scripts\s3>dir
Volume in drive C has no label.
Volume Serial Number is 2EF6-61D8

Directory of C:\Terraform_Scripts\s3

08/16/2024  06:56 PM    <DIR>          .
08/11/2024  10:17 AM    <DIR>          ..
08/16/2024  07:28 PM                956 provider.tf
08/16/2024  07:19 PM                163 s3.tf
               2 File(s)              1,119 bytes
               2 Dir(s)  84,339,662,848 bytes free

C:\Terraform_Scripts\s3>

```

Set all the three credentials in the command prompt as well.

```
Command Prompt
C:\Terraform_Scripts\s3>set AWS_ACCESS_KEY_ID=ASIAT43BMNECXJD2KGFN
C:\Terraform_Scripts\s3>set AWS_SECRET_ACCESS_KEY=/FBISLoHLeK92HFwWaiB4JLNnfJ7rP6NqXqu7NL
C:\Terraform_Scripts\s3>set AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEPb////////wEaCXVzLXdlc3QtM
iJHMEUCIQDMffzfpYp3gfoHtIz2xytFBnJ97oUlq8VTI0GjzuiogIgRWIYL1xq0c-fmSdrgOyKt8DwVZ00qgPwDBp0k
+4jMHfsqvQII7////////ARABGgwyNjgxMDI4MjIxNDkiDHJahoG5rId2wvvvTSqRAqDbLgvD1rYEdPRsrqsKVSH
pLPcfK/56YnE6JaoZHVW2EVoqkZIU8A5mWR6KaygCOKW6x30K61iTmm3CoFu6mjzWcjk/XTqzF7clqVJLENRhMk/KDp
8B2I6HhNmW7WDchZWdiQ2GVFYi4cbITI31mo0//z6aQJd02rRWrXL8msfKwlmKd9+UAtaK513ba9wn0yekBr6/tBccP
shDEKLYDmXA/yF0LP1SM07Ke5D8L5/ZiVY1y/HAncIj/3KLTor7f/Lv8pHwHirkmeCmnlX7DY6ucXBy8AVT3bbVoG4T
GR6hT2GZUoXpjbxbKbKaHAnxEizBK4gbNMIU6Vrb4Chv8GXtutkpJ0wZvmXESv8up29xU4jC+sf21BjqdAfwCYuaE+22
nfn+gIX4oV2xuc0LsyNcBpXILxdoUaiQ0y8LIZA94Z0bRNWW9Dw1pb10u0HY0a/pIY5N+i130Z+9JW0uMFwKIQA779
o4og0a3WACvNROJ8WdHekrOot7Ee3MiA9p7mWcc7cRo7kYEepscy4LlUuH9PeCM/i9z2cXYVyofoqPjLnT/ceP62ipic3
My6BmB8jwTg4FeViDg=
C:\Terraform_Scripts\s3>
```

Execute Terraform Init command to initialize the resources

```
Command Prompt
C:\Terraform_Scripts\s3>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.63.0...
- Installed hashicorp/aws v5.63.0 (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.

```

Command Prompt
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.aryan will be created
+ resource "aws_s3_bucket" "aryan" {
  + acceleration_status = (known after apply)
  + acl                 = "public-read"
  + arn                 = (known after apply)
  + bucket              = "aryan-saraf-bucket01092004"
  + 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          = {
    + "Environment" = "Dev"
    + "Name"        = "My 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.

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.

C:\Terraform_Scripts\s3>

```

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

```

Command Prompt
C:\Terraform_Scripts\s3>terraform apply

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.aryan will be created
+ resource "aws_s3_bucket" "aryan" {
  + acceleration_status      = (known after apply)
  + acl                      = (known after apply)
  + arn                     = (known after apply)
  + bucket                  = "aryan-saraf-bucket01092004"
  + 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              = {
    + "Environment" = "Dev"
    + "Name"        = "My 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.

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.aryan: Creating...
aws_s3_bucket.aryan: Creation complete after 7s [id=aryan-saraf-bucket01092004]

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

C:\Terraform_Scripts\s3>

```

AWS S3 Bucket dashboard after executing apply step.

The screenshot displays the AWS S3 Buckets dashboard for the us-east-1 region. The interface includes a top navigation bar with the AWS logo, a search bar, and a user profile. The main content area features a sidebar with the 'Amazon S3' link and a 'Buckets' section. A prominent 'Account snapshot' banner indicates it is updated every 24 hours. Below this, there are tabs for 'General purpose buckets' and 'Directory buckets'. The 'General purpose buckets' tab is active, showing a list of buckets. A 'Create bucket' button is visible. The bucket list contains one entry: 'aryan-saraf-bucket01092004' in the 'US East (N. Virginia) us-east-1' region, created on 'August 16, 2024, 19:45:42 (UTC+05:30)'. A 'View analyzer for us-east-1' link is provided for the IAM Access Analyzer.

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 (1) Info All AWS Regions

[Refresh](#) [Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	aryan-saraf-bucket01092004	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 16, 2024, 19:45:42 (UTC+05:30)

CloudShell Feedback Privacy Terms Cookie preferences

© 2024, Amazon Web Services, Inc. or its affiliates.

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

```

Command Prompt
C:\Terraform_Scripts\s3>terraform destroy
aws_s3_bucket.aryan: Refreshing state... [id=aryan-saraf-bucket01092004]

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_s3_bucket.aryan will be destroyed
- resource "aws_s3_bucket" "aryan" {
  - arn = "arn:aws:s3:::aryan-saraf-bucket01092004"
-> null
  - bucket = "aryan-saraf-bucket01092004" -> null
  - bucket_domain_name = "aryan-saraf-bucket01092004.s3.amazonaws.com" -> null
  - bucket_regional_domain_name = "aryan-saraf-bucket01092004.s3.us-east-1.amazonaws.com" -> null
  - force_destroy = false -> null
  - hosted_zone_id = "Z3AQBSTGFYJSTF" -> null
  - id = "aryan-saraf-bucket01092004" -> null
  - object_lock_enabled = false -> null
  - region = "us-east-1" -> null
  - request_payer = "BucketOwner" -> null
  - tags = {
    - "Environment" = "Dev"
    - "Name" = "My bucket"
  } -> null
  - tags_all = {
    - "Environment" = "Dev"
    - "Name" = "My bucket"
  } -> null
  # (3 unchanged attributes hidden)

  - grant {
    - id = "4fae71f766c2bbafadf590a9071cff08218db7f360aea8c77e125ec32608834a" -> 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

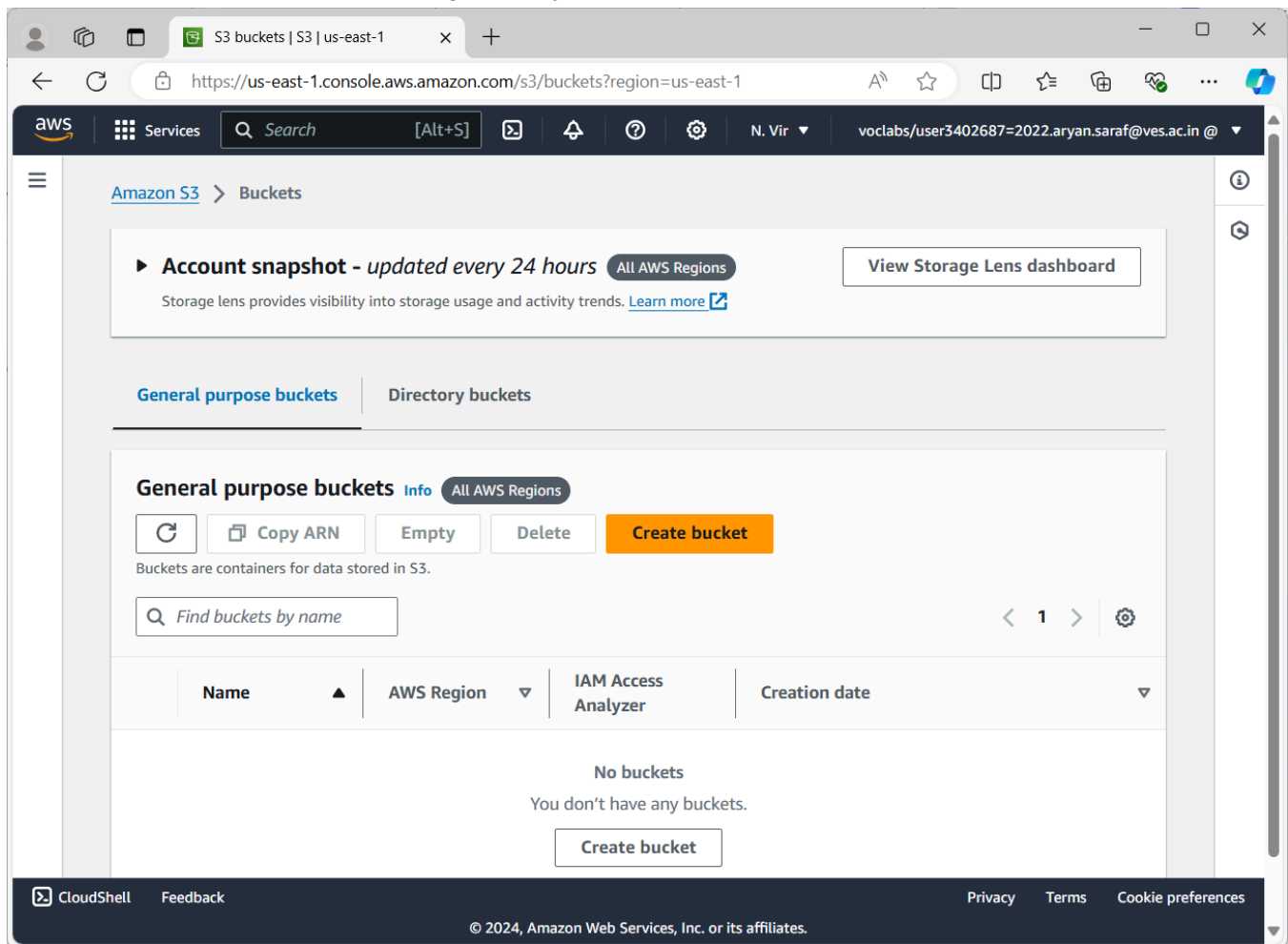
aws_s3_bucket.aryan: Destroying... [id=aryan-saraf-bucket01092004]
aws_s3_bucket.aryan: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.

C:\Terraform_Scripts\s3>

```

AWS EC2 dashboard after Executing Destroy step.



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.