Name:Sujal.S.Tekwani
Roll No: 63
Div:D15B

Advance DevOps-6

Aim: To build, change, and destroy infrastructure on AWS, GCP, Microsoft Azure, or DigitalOcean using Terraform, a tool for automating the management of cloud resources.

Theory:

Terraform is an open-source Infrastructure as Code (IaC) tool developed by HashiCorp. It allows developers and system administrators to define, provision, and manage cloud infrastructure using a declarative configuration language called HashiCorp Configuration Language (HCL). Terraform supports multiple cloud providers, including AWS, GCP, Azure, and DigitalOcean, making it a versatile tool for managing cloud environments.

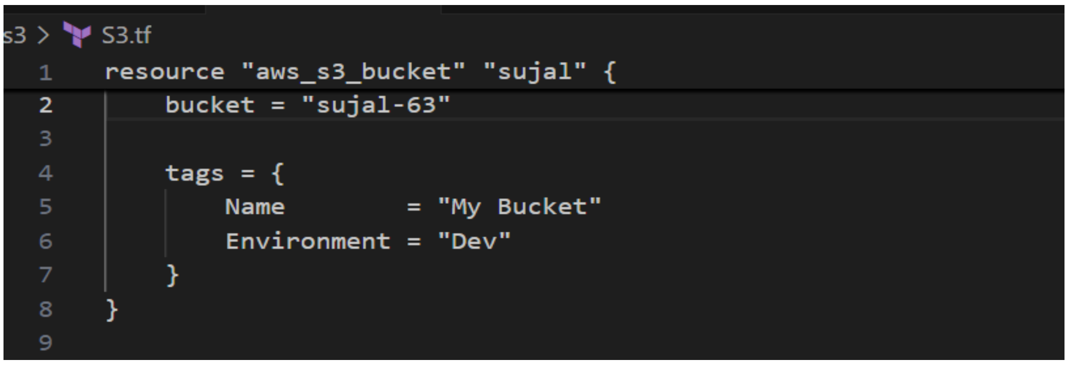
Key Concepts of Terraform:

1. **Infrastructure as Code (IaC):** Terraform enables defining cloud infrastructure through code, allowing for version control, reproducibility, and collaboration.
2. **State Management:** Terraform maintains the state of the infrastructure, allowing it to track and manage changes over time.
3. **Resource Provisioning:** It can provision various resources like virtual machines, storage, networking, and more across multiple cloud platforms.
4. **Modular Approach:** Infrastructure configurations can be broken into reusable modules, making them easy to manage and maintain.
5. **Lifecycle Management:** Terraform provides commands to create (`apply`), update (`plan`), and destroy (`destroy`) resources, making it easy to manage the full lifecycle of infrastructure.

Benefits of Using Terraform:

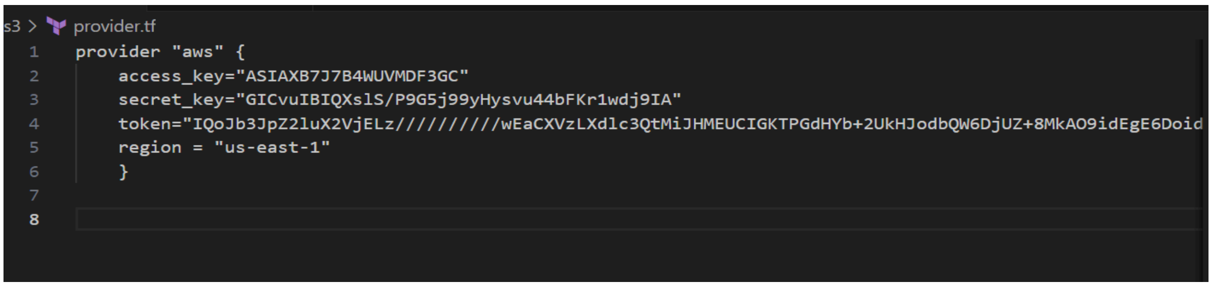
- **Multi-Cloud Compatibility:** Supports various cloud providers, offering flexibility and reducing vendor lock-in.
- **Automation:** Automates infrastructure provisioning, reducing manual efforts and minimizing errors.
- **Version Control:** Changes in infrastructure can be tracked, reviewed, and rolled back using version control systems.
- **Scalability:** Easily scales infrastructure up or down according to needs, optimizing resource usage and cost.

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



```
s3 > S3.tf
resource "aws_s3_bucket" "sujal" {
    bucket = "sujal-63"

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

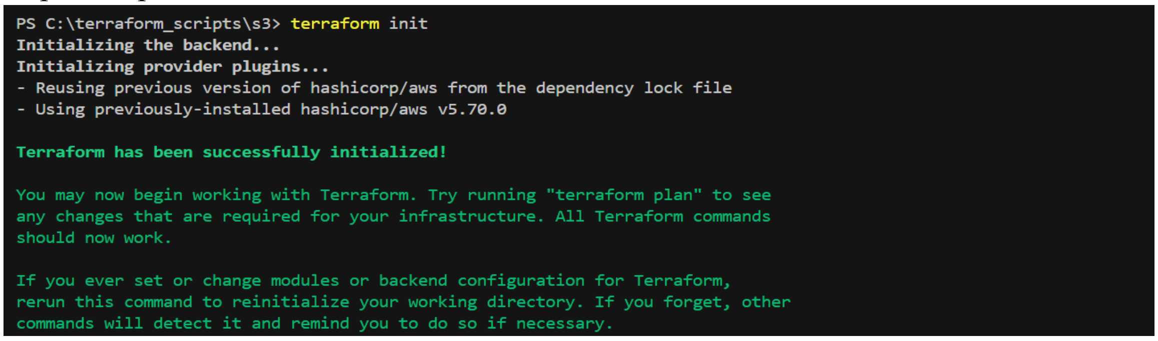
Create a new provider.tf file and write the following contents into it.



```
s3 > provider.tf
provider "aws" {
    access_key="ASIAXB7J7B4WUVMDF3GC"
    secret_key="GICvuIBIQXslS/P9G5j99yHysvu44bFKr1wdj9IA"
    token="IQoJb3JpZ2luX2VjELz//////////wEaCXVzLXdlc3QtMiJHMEUCIGKTPGdHYb+2UkHJodbQW6DjUZ+8MkAO9idEgE6Doid
    region = "us-east-1"
    }
```

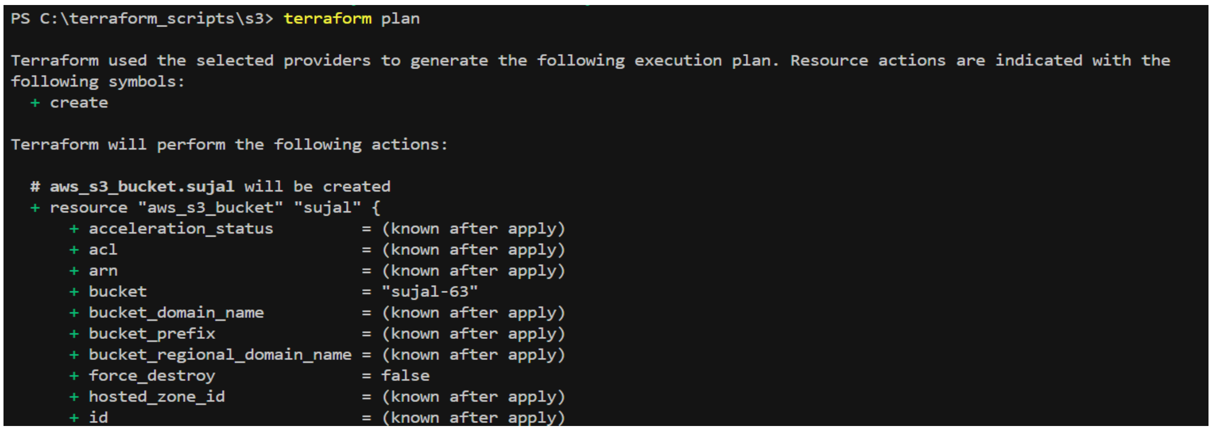
Save both the files in same directory Terraform_Scripts/S3

Step 2: Step 3: Execute Terraform Init command to initialize the resources



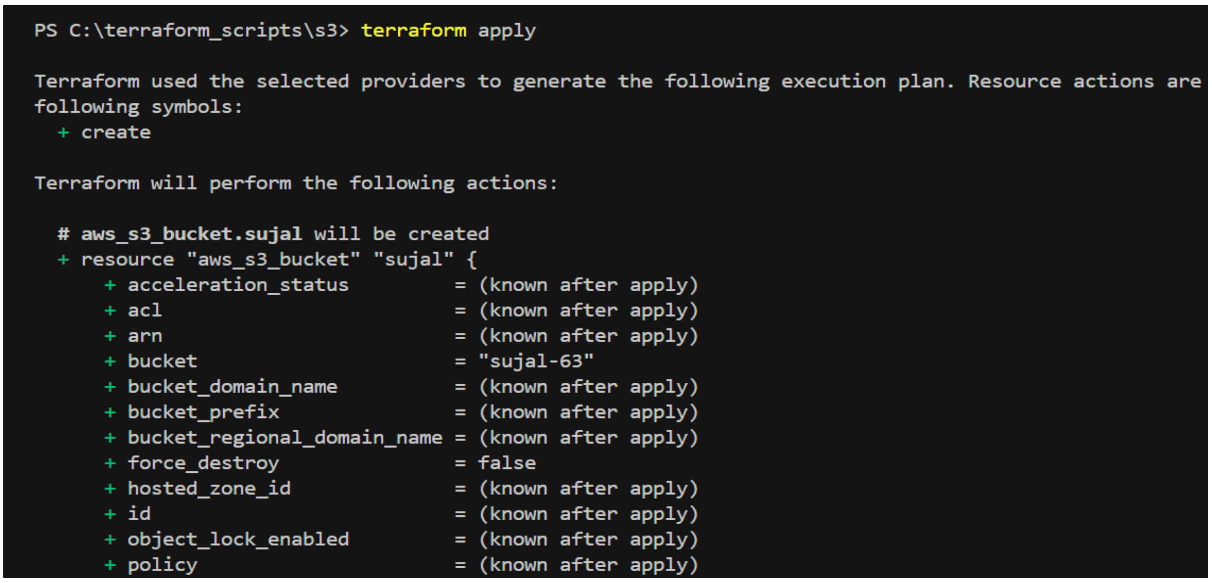
```
PS C:\terraform_scripts\s3> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.70.0

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

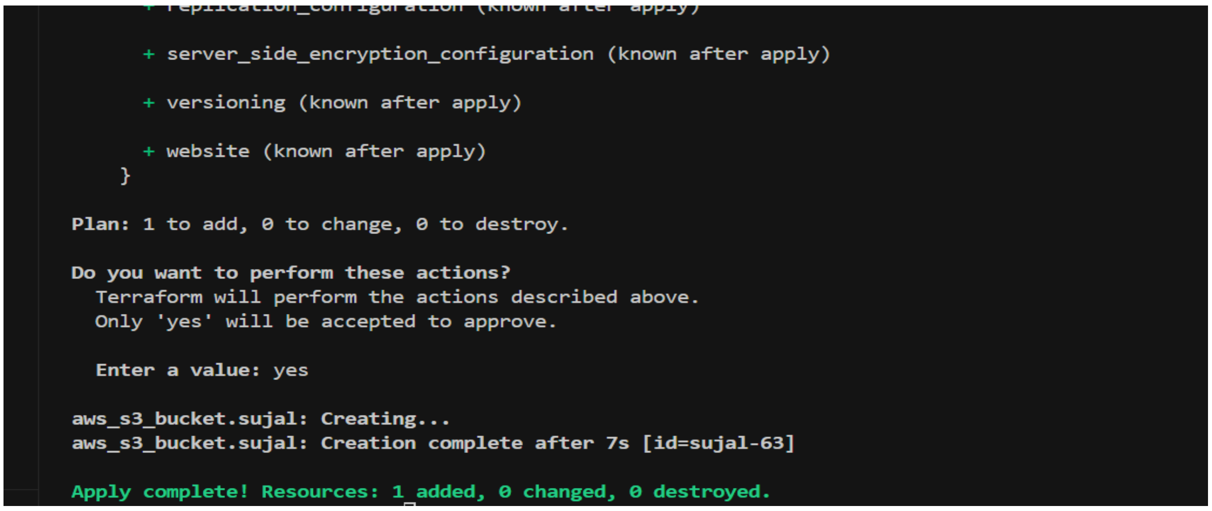


```
PS C:\terraform_scripts\s3> terraform plan

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.sujal will be created
  + resource "aws_s3_bucket" "sujal" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "sujal-63"
      + 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)
```

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

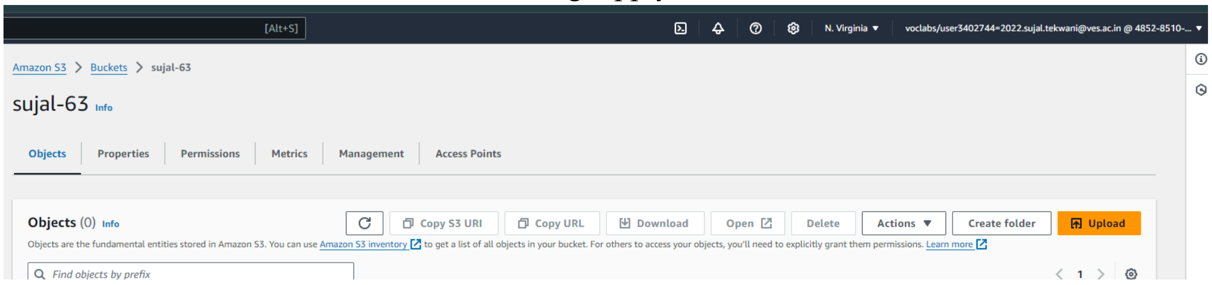


```
PS C:\terraform_scripts\s3> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are
following symbols:
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.sujal will be created
  + resource "aws_s3_bucket" "sujal" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "sujal-63"
      + 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)
```



```
      + 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.sujal: Creating...
aws_s3_bucket.sujal: Creation complete after 7s [id=sujal-63]

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

AWS S3bucket dashboard, Before Executing Apply command:



[Alt+S] N. Virginia ▼ voclabs/user3402744=2022.sujal.tekwani@ves.ac.in @ 4852-8510-... ▼

Amazon S3 > Buckets > sujal-63

sujal-63 Info

Objects | Properties | Permissions | Metrics | Management | Access Points

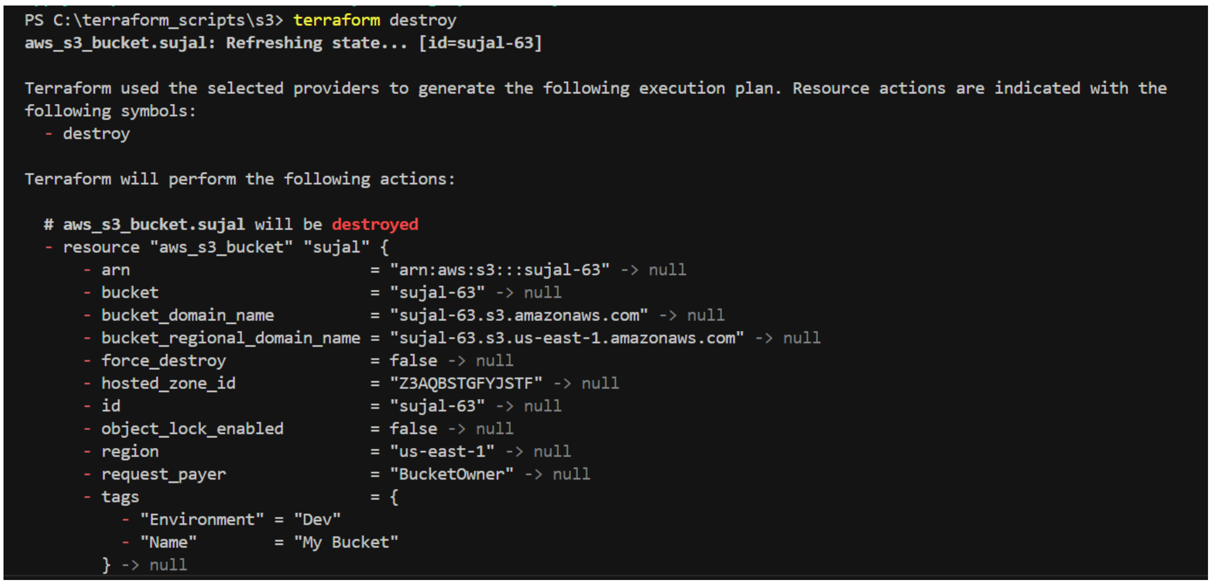
Objects (0) Info

Copy S3 URI | Copy URL | Download | Open | Delete | Actions ▼ | Create folder | Upload

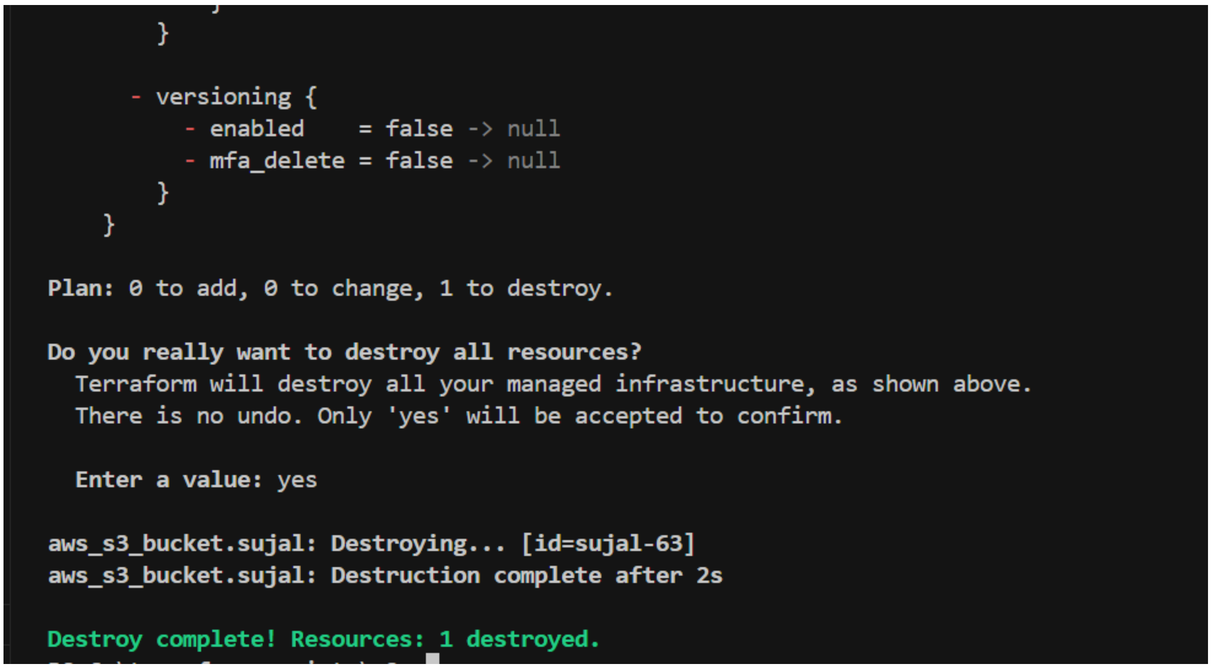
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Find objects by prefix

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

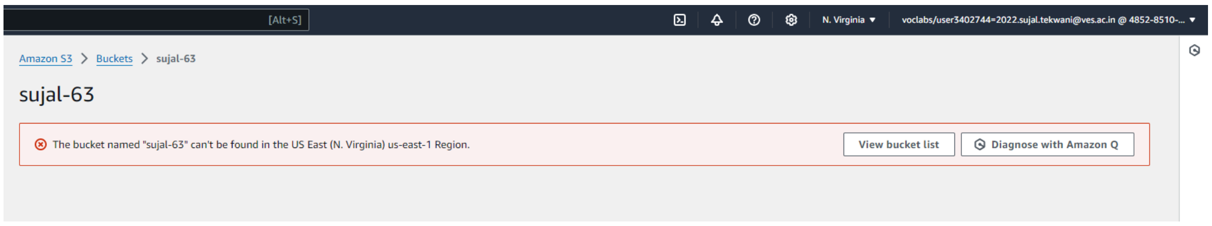


```
PS C:\terraform_scripts\s3> terraform destroy
aws_s3_bucket.sujal: Refreshing state... [id=sujal-63]

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.sujal will be destroyed
  - resource "aws_s3_bucket" "sujal" {
      - arn                         = "arn:aws:s3:::sujal-63" -> null
      - bucket                      = "sujal-63" -> null
      - bucket_domain_name          = "sujal-63.s3.amazonaws.com" -> null
      - bucket_regional_domain_name = "sujal-63.s3.us-east-1.amazonaws.com" -> null
      - force_destroy               = false -> null
      - hosted_zone_id              = "Z3AQBSTGFYJSTF" -> null
      - id                          = "sujal-63" -> null
      - object_lock_enabled         = false -> null
      - region                      = "us-east-1" -> null
      - request_payer               = "BucketOwner" -> null
      - tags                        = {
          - "Environment" = "Dev"
          - "Name"        = "My Bucket"
        } -> null
```



```
            }

          - 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.sujal: Destroying... [id=sujal-63]
  aws_s3_bucket.sujal: Destruction complete after 2s

  Destroy complete! Resources: 1 destroyed.
```

AWS S3 bucket dashboard, After Executing Apply command:



[Alt+S]
N. Virginia
voclabs/user3402744=2022.sujal.tekwani@ves.ac.in @ 4852-8510-...
Amazon S3 > Buckets > sujal-63

sujal-63

The bucket named "sujal-63" can't be found in the US East (N. Virginia) us-east-1 Region.

View bucket list
Diagnose with Amazon Q

Conclusion:

Using Terraform to build, change, and destroy infrastructure across AWS, GCP, Microsoft Azure, and DigitalOcean showcases the power and flexibility of Infrastructure as Code (IaC). Terraform's ability to automate infrastructure provisioning, manage state, and support multiple cloud platforms streamlines operations, reduces human error, and enhances scalability. By employing Terraform, organizations can efficiently manage their cloud environments, enforce best practices, and achieve consistent infrastructure management across diverse cloud providers, ultimately accelerating their cloud adoption and operational agility.