Advance DevOps Lab - Practical 6
Varun Gupta - D15 B - Roll no. 20

Aim : To Build, change, and destroy AWS infrastructure Using Terraform. (S3 bucket)

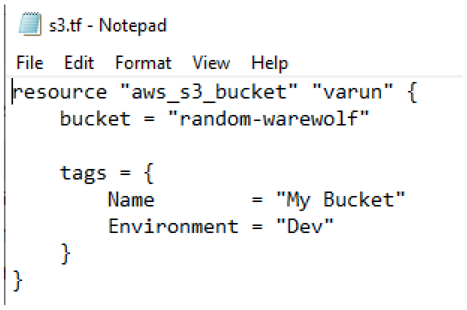
Theory :

Terraform is an infrastructure as code (IaC) tool that allows you to build, change, and version infrastructure safely and efficiently. This includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc.

Terraform can manage infrastructure on multiple cloud platforms. Terraform state allows you to track resource changes throughout your deployments. You can commit your configurations to version control to safely collaborate on infrastructure. Terraform plugins called providers let Terraform interact with cloud platforms and other services via their application programming interfaces (APIs).

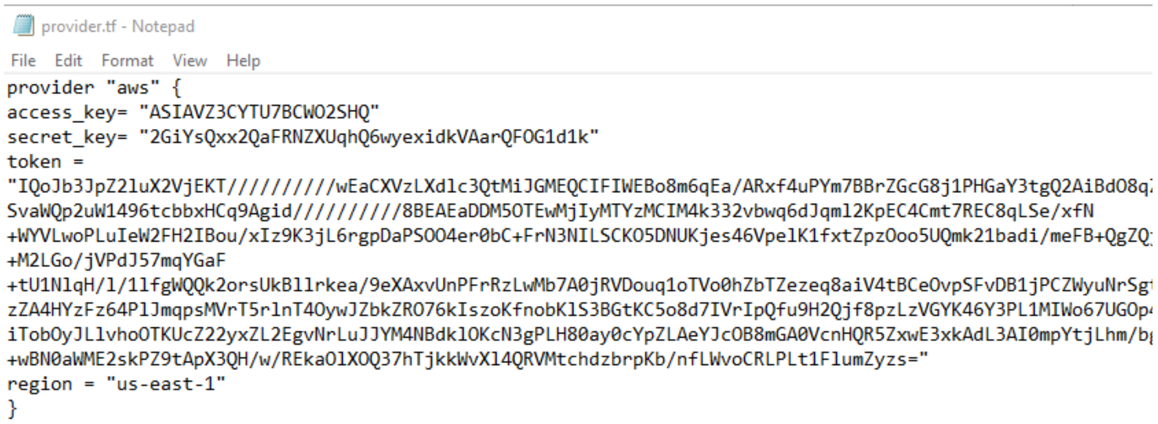
Steps :

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



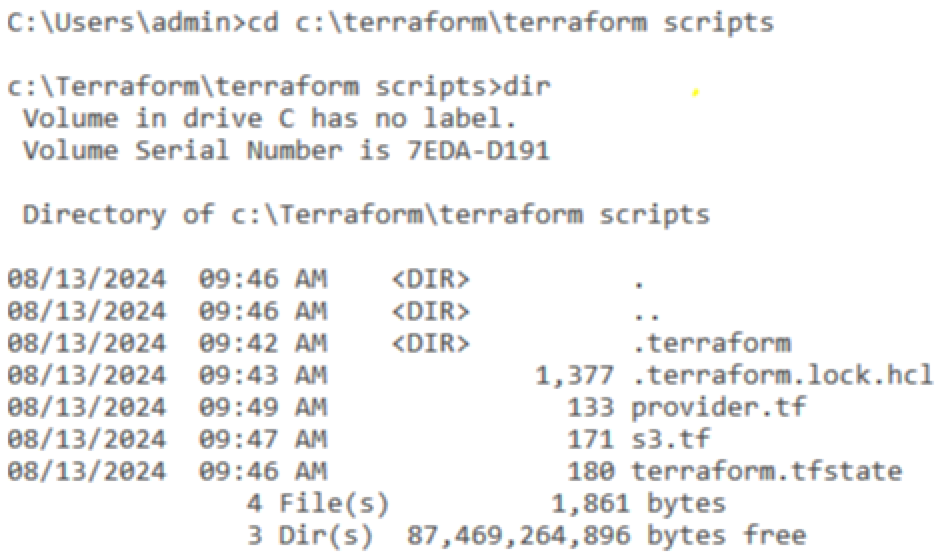
```
resource "aws_s3_bucket" "varun" {
    bucket = "random-warewolf"

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

Create a new provider.tf file -



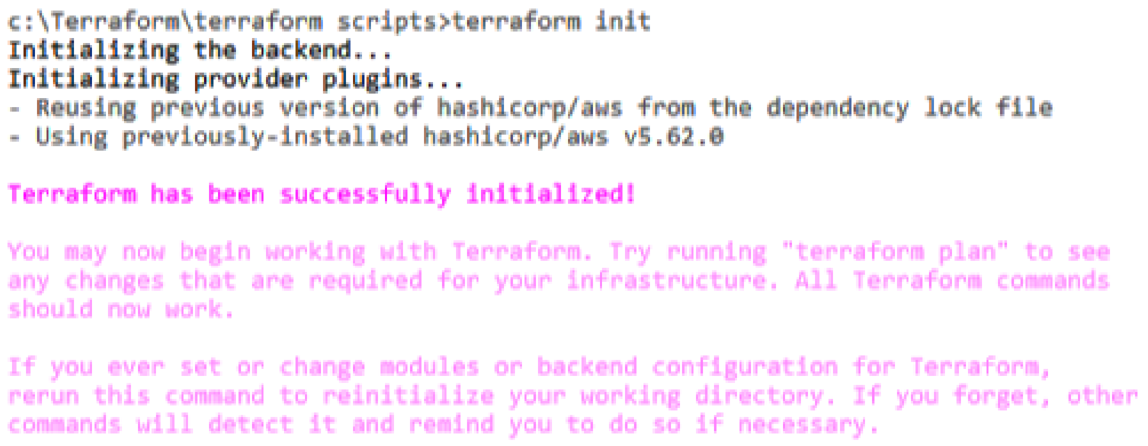
```
provider "aws" {
access_key= "ASIAVZ3CYTU7BCWO2SHQ"
secret_key= "2GiYsQxx2QaFRNZXUqhQ6wyexidkVAarQFOG1d1k"
token =
"IQoJb3JpZ2luX2VjEKT//////////wEaCXVzLXdlc3QtMiJGMEQCIFIWEBo8m6qEa/ARxf4uPYm7BBrZGcG8j1PHGaY3tgQ2AiBdO8q
SvaWQp2uW1496tcbbxHCq9Agid//////////8BEAEaDDM5OTEwMjIyMTYzMCIM4k332vbwq6dJqml2KpEC4Cmt7REC8qLSe/xfN
+WYVLwoPLuIeW2FH2IBou/xIz9K3jL6rgpDaPSO04er0bC+FrN3NILSCKO5DNUKjes46VpelK1fxtZpzOoo5UQmk21badi/meFB+QgZQ
+M2LGo/jVPdJ57mqYGaF
+tU1NlqH/l/1lfgWQQk2orsUkBllrkea/9eXAxvUnPFrRzLwMb7A0jRVDouq1oTVo0hZbTZezeq8aiV4tBCeOvpSFvDB1jPCZWyuNrSg
zZA4HYzFz64PlJmqpsMVrT5rlnT4OywJZbkZRO76kIszoKfnobKlS3BGtKC5o8d7IVrIpQfu9H2Qjf8pzLzVGYK46Y3PL1MIWo67UGOp
iTobOyJLlvhoOTKUcZ22yxZL2EgvNrLuJJYM4NBdklOKcN3gPLH80ay0cYpZLAeYJcOB8mGA0VcnHQR5ZxwE3xkAdL3AI0mpYtjLhm/b
+wBN0aWME2skPZ9tApX3QH/w/REkaOlXOQ37hTjkkWvXl4QRVMtchdzbrpKb/nfLWvoCRLPLt1FlumZyzs="
region = "us-east-1"
}
```

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



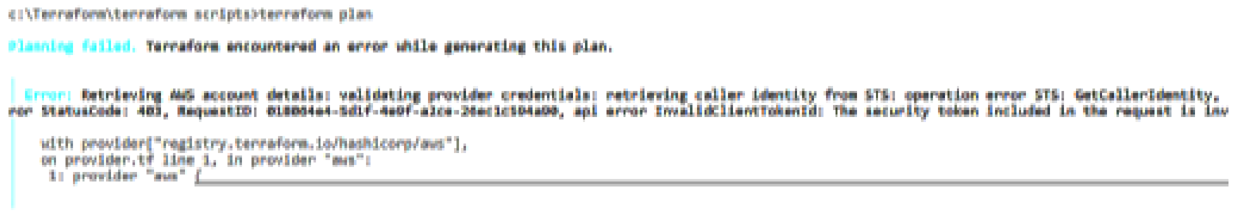
```
C:\Users\admin>cd c:\terraform\terraform scripts

c:\Terraform\terraform scripts>dir
 Volume in drive C has no label.
 Volume Serial Number is 7EDA-D191

 Directory of c:\Terraform\terraform scripts

08/13/2024  09:46 AM    <DIR>          .
08/13/2024  09:46 AM    <DIR>          ..
08/13/2024  09:42 AM    <DIR>          .terraform
08/13/2024  09:43 AM             1,377 .terraform.lock.hcl
08/13/2024  09:49 AM               133 provider.tf
08/13/2024  09:47 AM               171 s3.tf
08/13/2024  09:46 AM               180 terraform.tfstate
               4 File(s)          1,861 bytes
               3 Dir(s)  87,469,264,896 bytes free
```

Step 3: Execute Terraform Init command to initialize the resources

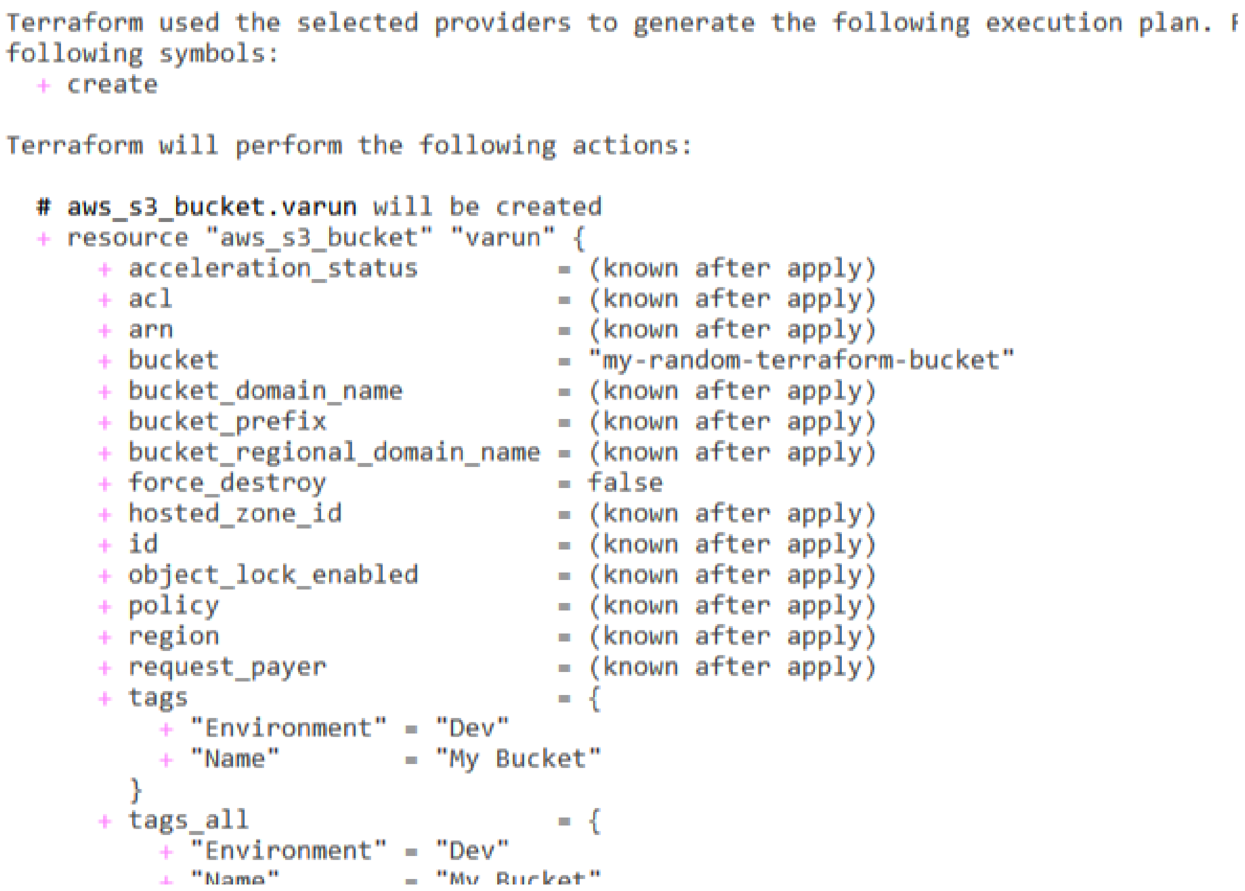


```
c:\Terraform\terraform scripts>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.62.0

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

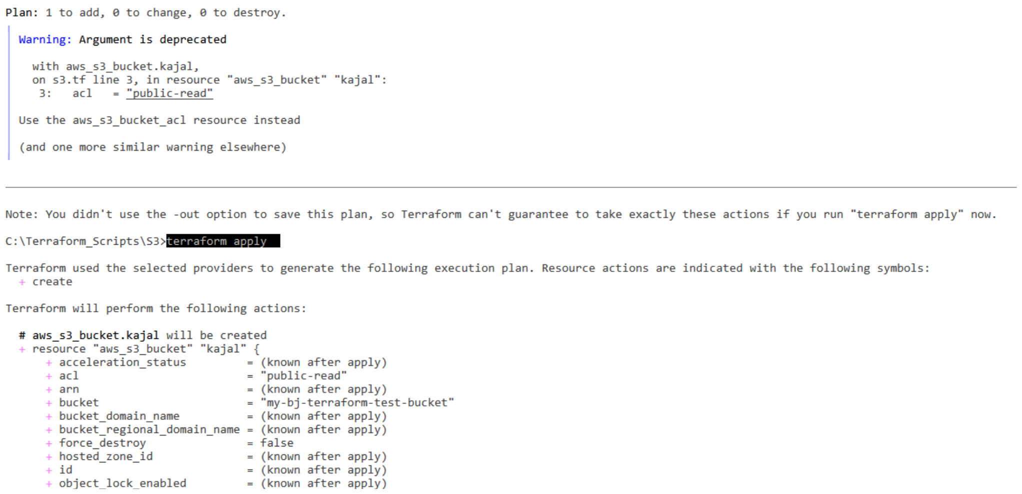


```
c:\Terraform\terraform scripts>terraform plan

Planning failed. Terraform encountered an error while generating this plan.

│ Error: Retrieving AWS account details: validating provider credentials: retrieving caller identity from STS: operation error STS: GetCallerIdentity,
ror StatusCode: 403, RequestID: 018004e4-5d1f-4e0f-a1ce-26ec1c504a00, api error InvalidClientTokenId: The security token included in the request is inv
│
│   with provider["registry.terraform.io/hashicorp/aws"],
│   on provider.tf line 1, in provider "aws":
│    1: provider "aws" {
```



```
Terraform used the selected providers to generate the following execution plan.
following symbols:
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.varun will be created
  + resource "aws_s3_bucket" "varun" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "my-random-terraform-bucket"
      + 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"
```

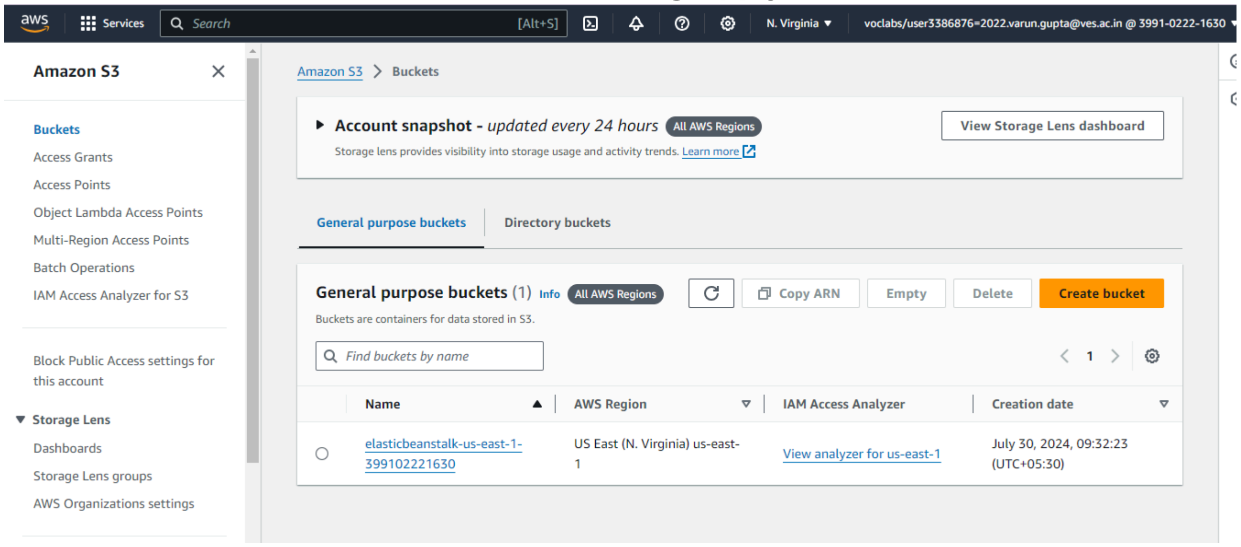
Step 5: Execute Terraform apply to apply the configuration, which will automatically create an S3 bucket based on our configuration.



```
Plan: 1 to add, 0 to change, 0 to destroy.

  Warning: Argument is deprecated

    with aws_s3_bucket.kajal,
    on s3.tf line 3, in resource "aws_s3_bucket" "kajal":
     3:   acl    = "public-read"

  Use the aws_s3_bucket_acl resource instead

  (and one more similar warning elsewhere)

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>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.kajal will be created
  + resource "aws_s3_bucket" "kajal" {
      + acceleration_status         = (known after apply)
      + acl                         = "public-read"
      + arn                         = (known after apply)
      + bucket                      = "my-bj-terraform-test-bucket"
      + bucket_domain_name          = (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)
```

```
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.kajal: Creating...
aws_s3_bucket.kajal: Creation complete after 2s [id=my-bj-terraform-test-bucket]

  Warning: Argument is deprecated

    with aws_s3_bucket.kajal,
    on s3.tf line 3, in resource "aws_s3_bucket" "kajal":
     3:   acl    = "public-read"

  Use the aws_s3_bucket_acl resource instead

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

C:\Terraform_Scripts\S3>
```

AWS S3bucket dashboard, Before Executing Apply command:



Account snapshot - *updated every 24 hours*

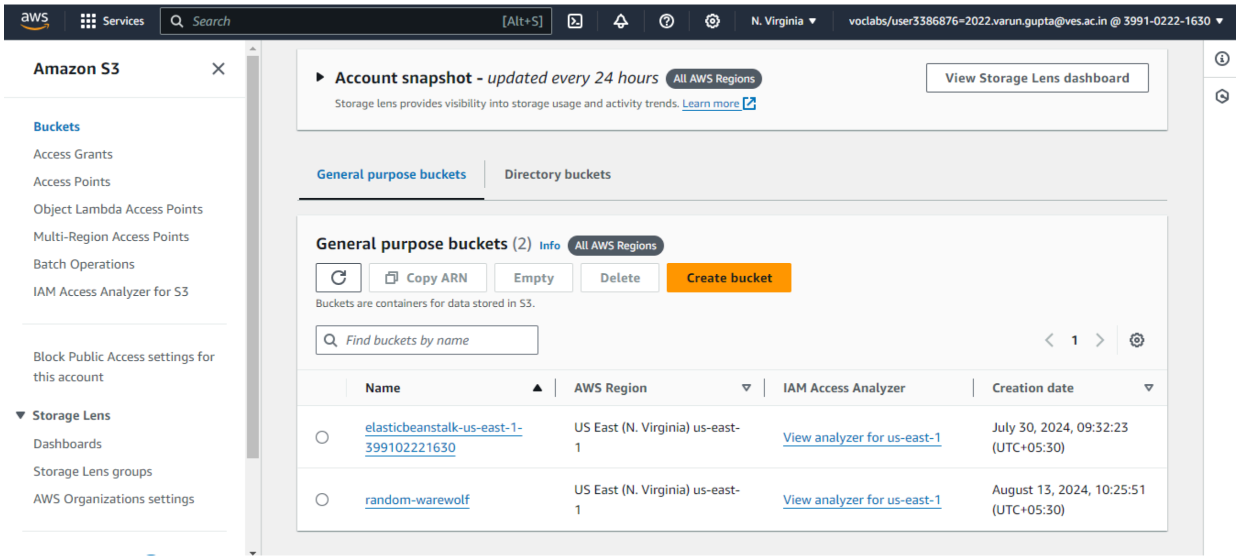
Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (1)

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-399102221630	US East (N. Virginia) us-east-1	View analyzer for us-east-1	July 30, 2024, 09:32:23 (UTC+05:30)

AWS S3 Bucket dashboard, After Executing Apply step



Account snapshot - *updated every 24 hours*

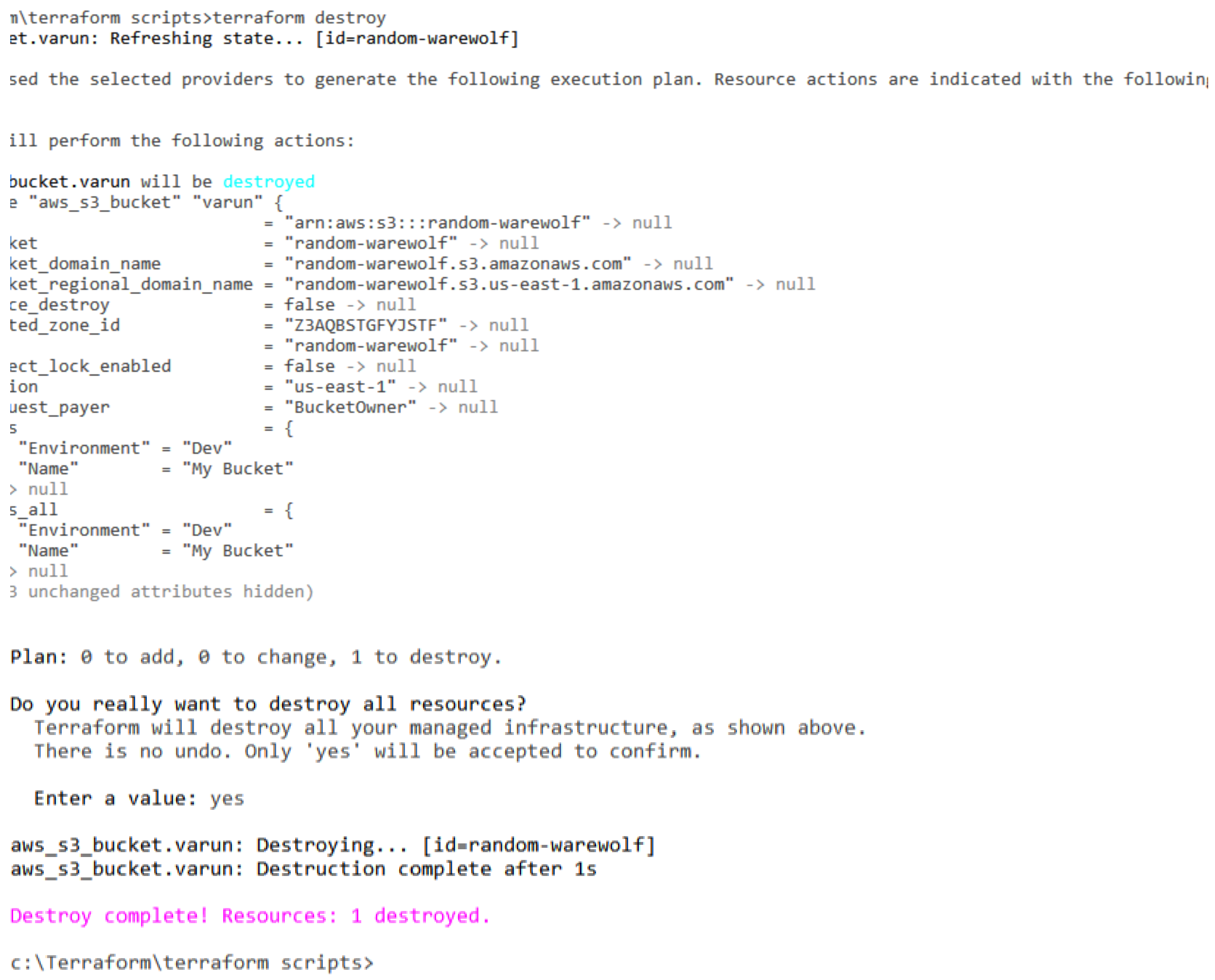
Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (2)

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-399102221630	US East (N. Virginia) us-east-1	View analyzer for us-east-1	July 30, 2024, 09:32:23 (UTC+05:30)
random-warewolf	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 13, 2024, 10:25:51 (UTC+05:30)

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



```
n\terraform scripts>terraform destroy
et.varun: Refreshing state... [id=random-warewolf]

sed the selected providers to generate the following execution plan. Resource actions are indicated with the followin

ill perform the following actions:

bucket.varun will be destroyed
e "aws_s3_bucket" "varun" {
                          = "arn:aws:s3:::random-warewolf" -> null
ket                       = "random-warewolf" -> null
ket_domain_name           = "random-warewolf.s3.amazonaws.com" -> null
ket_regional_domain_name  = "random-warewolf.s3.us-east-1.amazonaws.com" -> null
ce_destroy                = false -> null
ted_zone_id               = "Z3AQBSTGFYJSTF" -> null
                          = "random-warewolf" -> null
ect_lock_enabled          = false -> null
ion                       = "us-east-1" -> null
uest_payer                = "BucketOwner" -> null
s                         = {
 "Environment" = "Dev"
 "Name"        = "My Bucket"
> null
s_all                     = {
 "Environment" = "Dev"
 "Name"        = "My Bucket"
> null
3 unchanged attributes hidden)


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.varun: Destroying... [id=random-warewolf]
aws_s3_bucket.varun: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.

c:\Terraform\terraform scripts>
```

Conclusion :

In conclusion, this project successfully demonstrates the ability to build, modify, and destroy AWS infrastructure, specifically an S3 bucket, using Terraform. It highlights Terraform's efficiency in managing cloud resources through automation, ensuring scalable and repeatable infrastructure management.