Experiment 6

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

Theory :

Terraform is an open-source tool that enables developers and operations teams to define, provision, and manage cloud infrastructure through code. It uses a declarative language to specify the desired state of infrastructure, which can include servers, storage, networking components, and more. With Terraform, infrastructure changes can be automated, versioned, and tracked efficiently.

Building Infrastructure

When you build infrastructure using Terraform, you define the desired state of your infrastructure in configuration files. For example, you may want to create an S3 bucket or deploy a Docker container on an EC2 instance. Terraform reads these configuration files and, using the specified cloud provider (such as AWS), it provisions the necessary resources to match the desired state.

- S3 Buckets: Terraform can create and manage S3 buckets, which are used to store and retrieve data objects in the cloud. You can define the properties of the bucket, such as its name, region, access permissions, and versioning.

- Docker on AWS: Terraform can deploy Docker containers on AWS infrastructure. This often involves setting up an EC2 instance and configuring it to run Docker containers, which encapsulate applications and their dependencies.

Changing Infrastructure

As your needs evolve, you may need to modify the existing infrastructure. Terraform makes it easy to implement changes by updating the configuration files to reflect the new desired state. For instance, you might want to change the storage settings of an S3 bucket, add new security policies, or modify the Docker container's configuration.

Terraform's "plan" command helps you preview the changes that will be made to your infrastructure before applying them. This step ensures that you understand the impact of your changes and can avoid unintended consequences.

Destroying Infrastructure

When certain resources are no longer needed, Terraform allows you to destroy them in a controlled manner. This might involve deleting an S3 bucket or terminating an EC2 instance running Docker containers. By running the "destroy" command, Terraform ensures that all associated resources are properly de-provisioned and removed.

Destroying infrastructure with Terraform is beneficial because it helps avoid unnecessary costs associated with unused resources and ensures that the environment remains clean and free of clutter.

Benefits of Using Terraform for AWS Infrastructure

1.Consistency: Terraform ensures that infrastructure is consistent across environments by applying the same configuration files.

2.Automation: Manual processes are reduced, and infrastructure is provisioned, updated, and destroyed automatically based on code.

3.Version Control: Infrastructure configurations can be stored in version control systems (like Git), allowing teams to track changes, collaborate, and roll back if necessary.

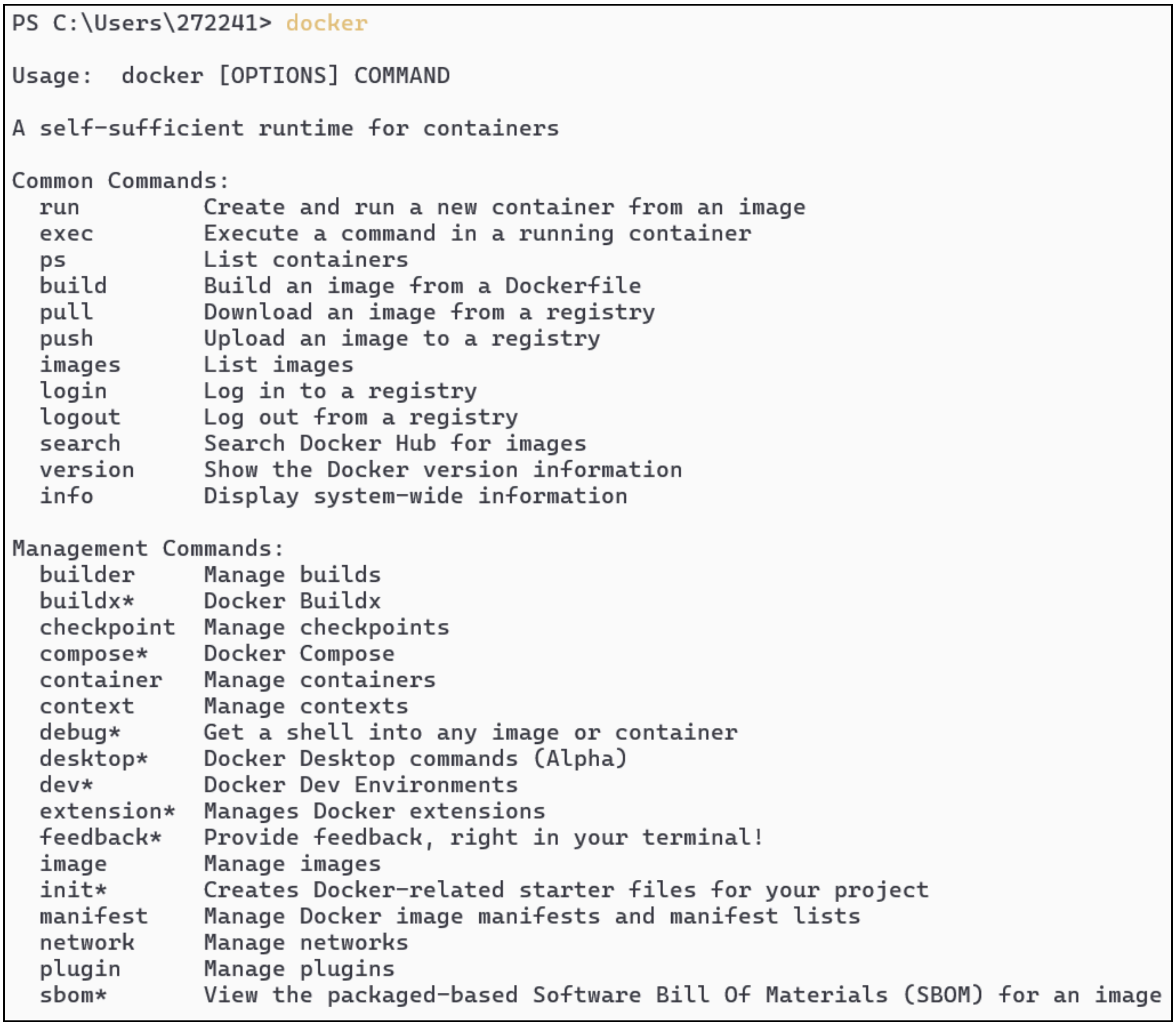
4.Scalability: Terraform can manage complex infrastructures, scaling them up or down as needed, whether for small projects or large-scale applications.

5.Modularity: Terraform configurations can be broken down into reusable modules, making it easier to manage and scale infrastructure.

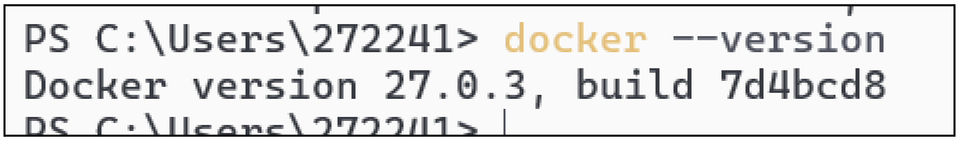
Implementation :

Terraform and Docker -

Step 1: Check the docker functionality:

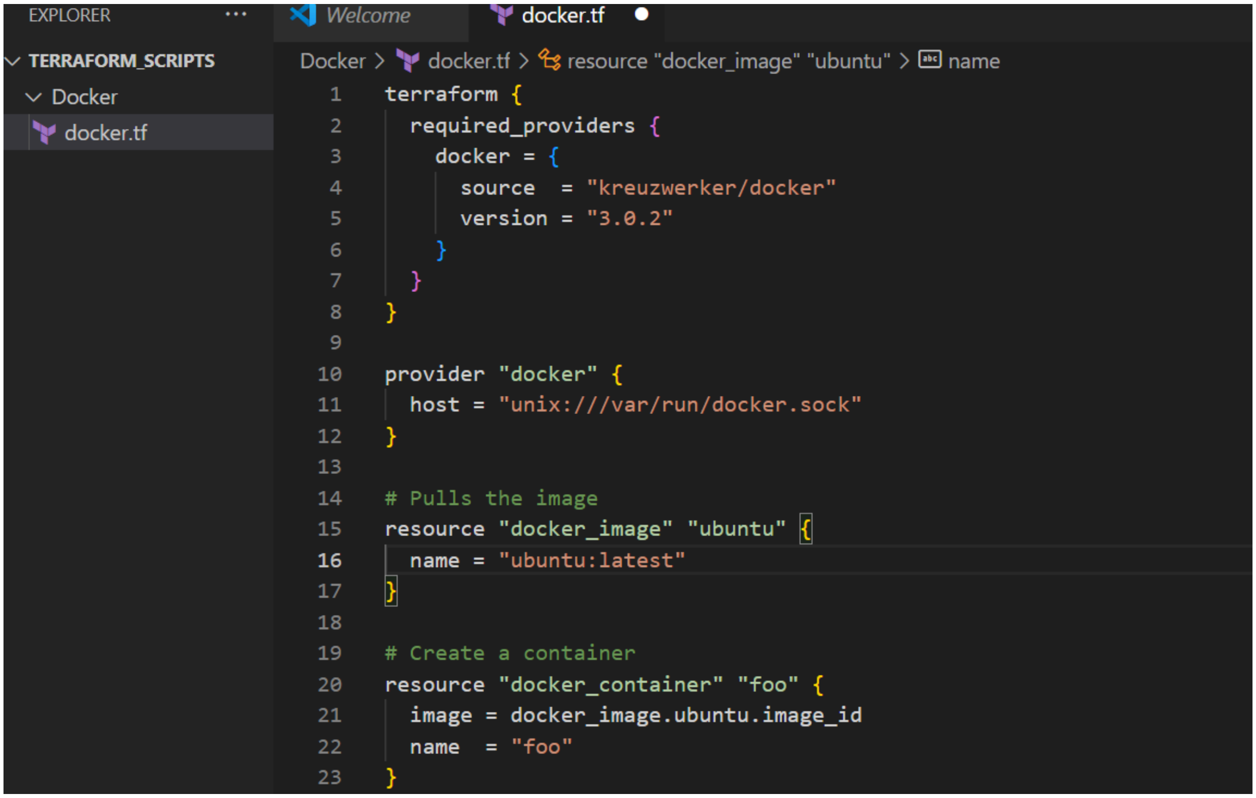


```
PS C:\Users\272241> docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
  run         Create and run a new container from an image
  exec        Execute a command in a running container
  ps          List containers
  build       Build an image from a Dockerfile
  pull        Download an image from a registry
  push        Upload an image to a registry
  images      List images
  login       Log in to a registry
  logout      Log out from a registry
  search      Search Docker Hub for images
  version     Show the Docker version information
  info        Display system-wide information

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  checkpoint  Manage checkpoints
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
  desktop*    Docker Desktop commands (Alpha)
  dev*        Docker Dev Environments
  extension*  Manages Docker extensions
  feedback*   Provide feedback, right in your terminal!
  image       Manage images
  init*       Creates Docker-related starter files for your project
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image
```



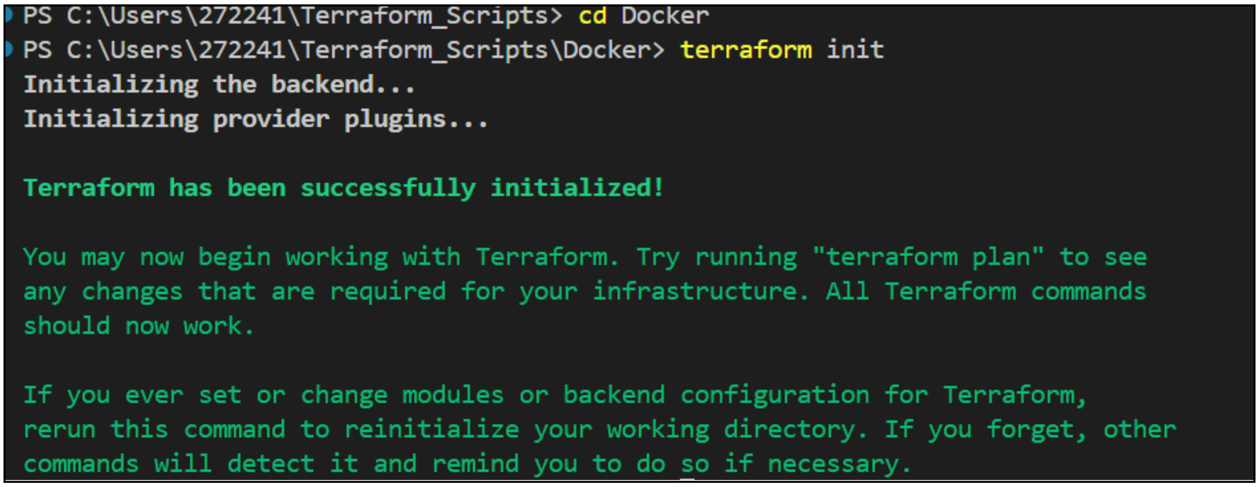
```
PS C:\Users\272241> docker --version
Docker version 27.0.3, build 7d4bcd8
PS C:\Users\272241>
```

Step 2:


EXPLORER
TERRAFORM_SCRIPTS
Docker
docker.tf
Welcome
docker.tf
Docker > docker.tf > resource "docker_image" "ubuntu" > name

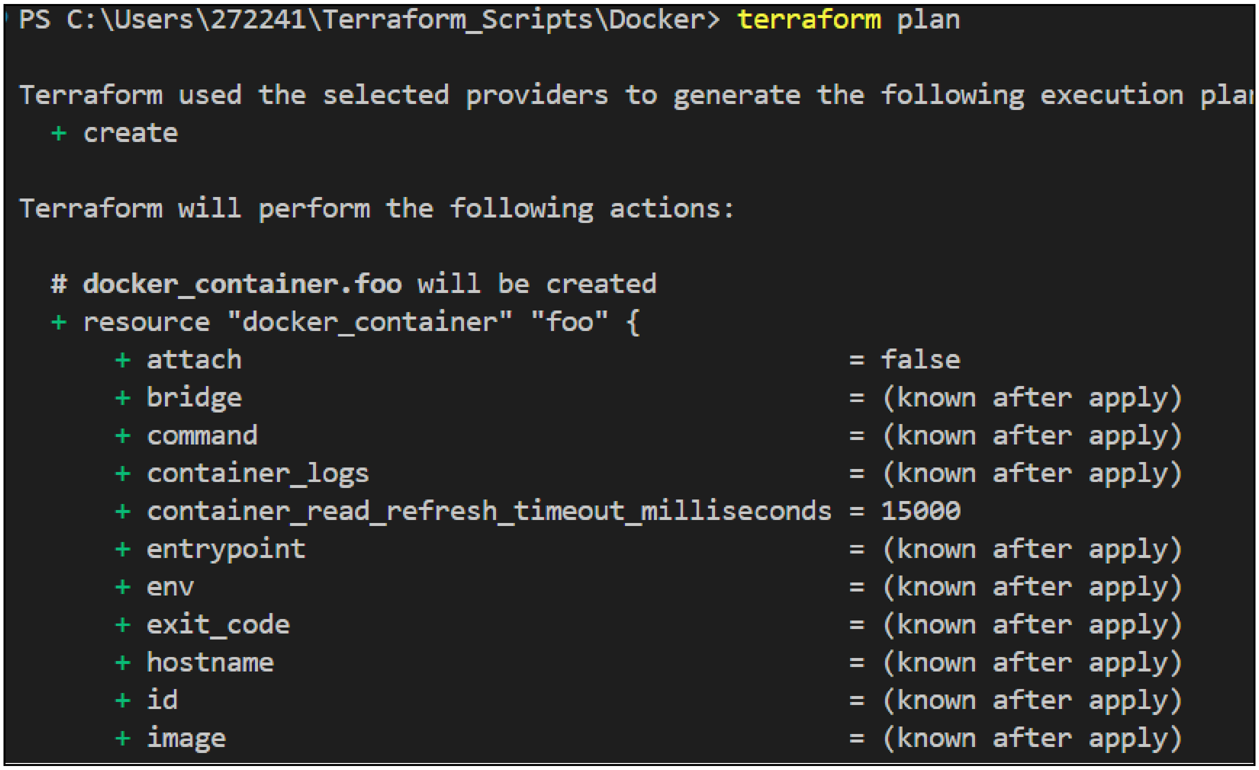
```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "3.0.2"
    }
  }
}

provider "docker" {
  host = "unix:///var/run/docker.sock"
}

# Pulls the image
resource "docker_image" "ubuntu" {
  name = "ubuntu:latest"
}

# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name  = "foo"
}
```

Step 3: Executed the terraform init command.

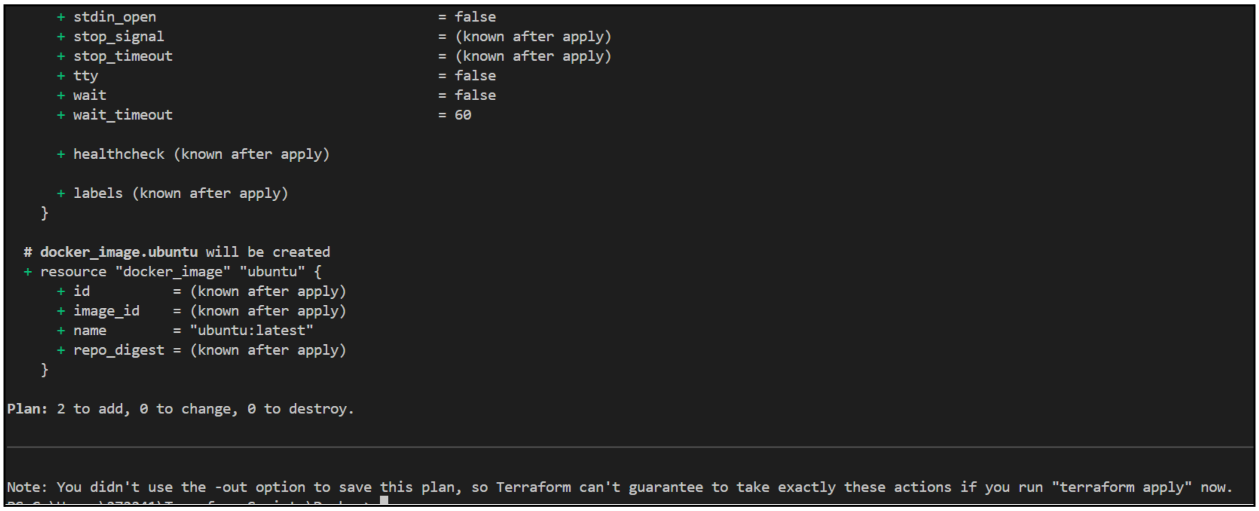


```
PS C:\Users\272241\Terraform_Scripts> cd Docker
PS C:\Users\272241\Terraform_Scripts\Docker> terraform init
Initializing the backend...
Initializing provider plugins...

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 the terraform plan to see the resources.



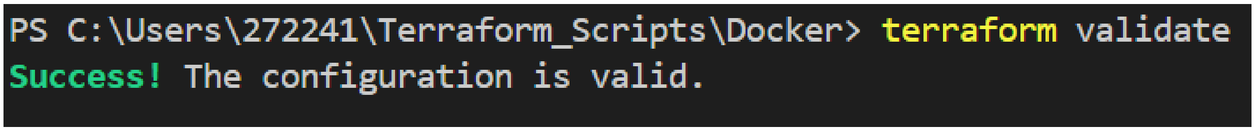
```
PS C:\Users\272241\Terraform_Scripts\Docker> terraform plan

Terraform used the selected providers to generate the following execution pla
  + create

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + attach                                      = false
      + bridge                                      = (known after apply)
      + command                                     = (known after apply)
      + container_logs                              = (known after apply)
      + container_read_refresh_timeout_milliseconds = 15000
      + entrypoint                                  = (known after apply)
      + env                                         = (known after apply)
      + exit_code                                   = (known after apply)
      + hostname                                    = (known after apply)
      + id                                          = (known after apply)
      + image                                       = (known after apply)
```



```
      + stdin_open                                  = false
      + stop_signal                                 = (known after apply)
      + stop_timeout                                = (known after apply)
      + tty                                         = false
      + wait                                        = false
      + wait_timeout                                = 60

      + healthcheck (known after apply)

      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + name        = "ubuntu:latest"
      + repo_digest = (known after apply)
    }

Plan: 2 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.
```

(Extra)

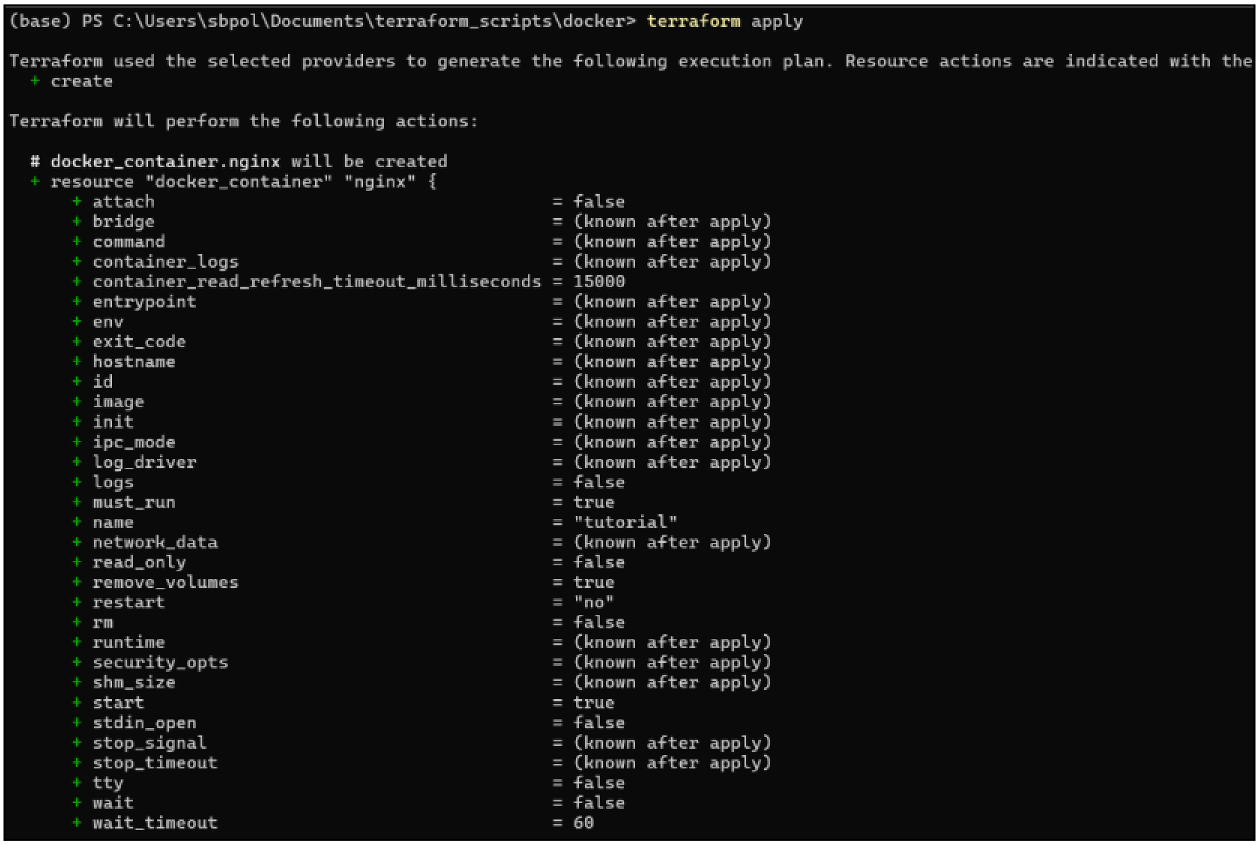
Step 5: Use terraform validate to check if the code is perfect to build further and apply the changes.



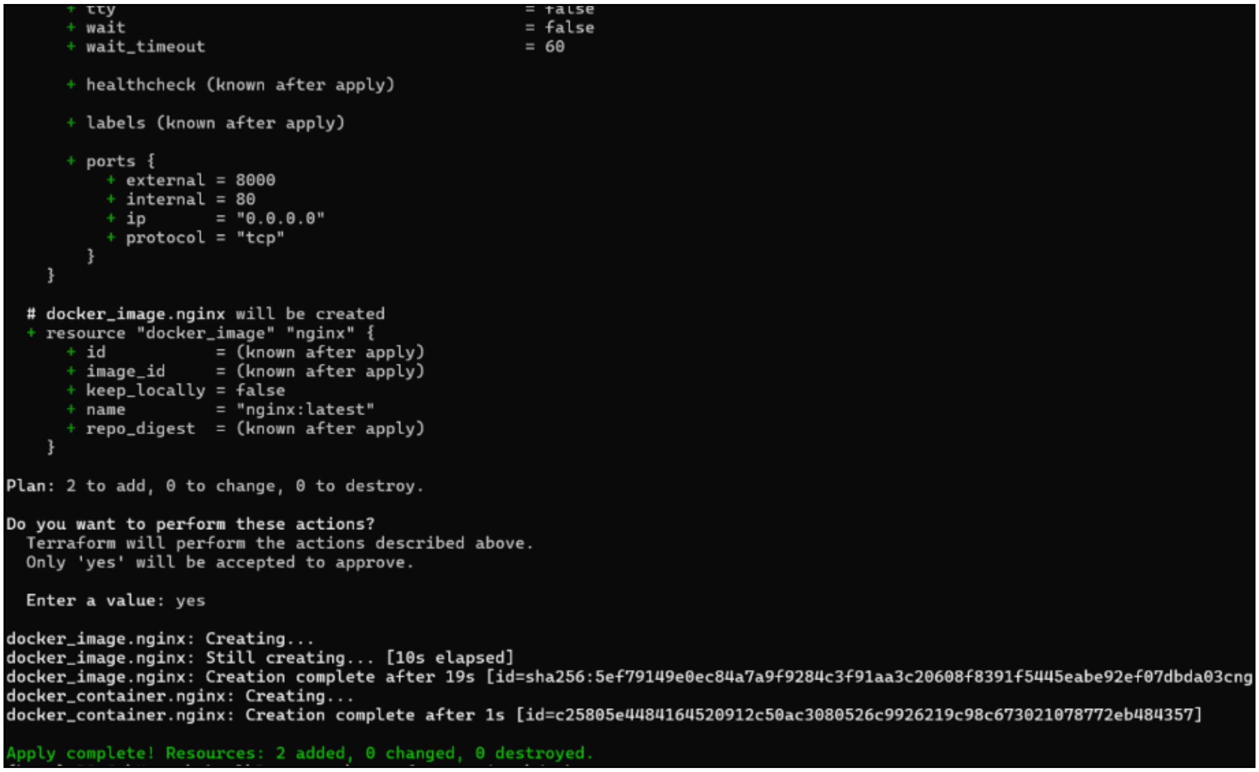
```
PS C:\Users\272241\Terraform_Scripts\Docker> terraform validate
Success! The configuration is valid.
```

Step 6: Execute Terraform apply to apply the configuration, which will automatically create and run the Ubuntu Linux container based on our configuration.

Using command :“terraform apply”

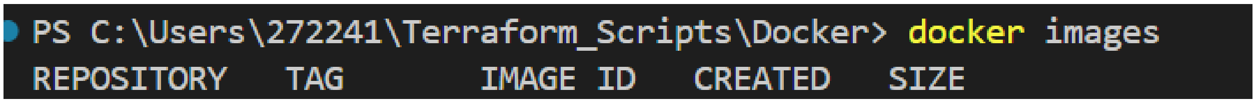


```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> terraform apply

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

Terraform will perform the following actions:

  # docker_container.nginx will be created
  + resource "docker_container" "nginx" {
      + attach                                      = false
      + bridge                                      = (known after apply)
      + command                                     = (known after apply)
      + container_logs                              = (known after apply)
      + container_read_refresh_timeout_milliseconds = 15000
      + entrypoint                                  = (known after apply)
      + env                                         = (known after apply)
      + exit_code                                   = (known after apply)
      + hostname                                    = (known after apply)
      + id                                          = (known after apply)
      + image                                       = (known after apply)
      + init                                        = (known after apply)
      + ipc_mode                                    = (known after apply)
      + log_driver                                  = (known after apply)
      + logs                                        = false
      + must_run                                    = true
      + name                                        = "tutorial"
      + network_data                                = (known after apply)
      + read_only                                   = false
      + remove_volumes                              = true
      + restart                                     = "no"
      + rm                                          = false
      + runtime                                     = (known after apply)
      + security_opts                               = (known after apply)
      + shm_size                                    = (known after apply)
      + start                                       = true
      + stdin_open                                  = false
      + stop_signal                                 = (known after apply)
      + stop_timeout                                = (known after apply)
      + tty                                         = false
      + wait                                        = false
      + wait_timeout                                = 60
```



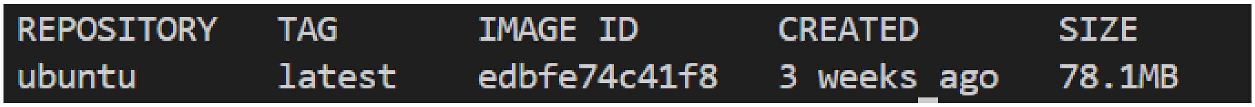
```
      + tty                                        = false
      + wait                                       = false
      + wait_timeout                               = 60

      + healthcheck (known after apply)

      + labels (known after apply)

      + ports {
          + external = 8000
          + internal = 80
          + ip       = "0.0.0.0"
          + protocol = "tcp"
        }
    }

  # docker_image.nginx will be created
  + resource "docker_image" "nginx" {
      + id           = (known after apply)
      + image_id     = (known after apply)
      + keep_locally = false
      + name         = "nginx:latest"
      + repo_digest  = (known after apply)
    }

Plan: 2 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

docker_image.nginx: Creating...
docker_image.nginx: Still creating... [10s elapsed]
docker_image.nginx: Creation complete after 19s [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cng
docker_container.nginx: Creating...
docker_container.nginx: Creation complete after 1s [id=c25805e4484164520912c50ac3080526c9926219c98c673021078772eb484357]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Step 7: Docker images before applying the changes



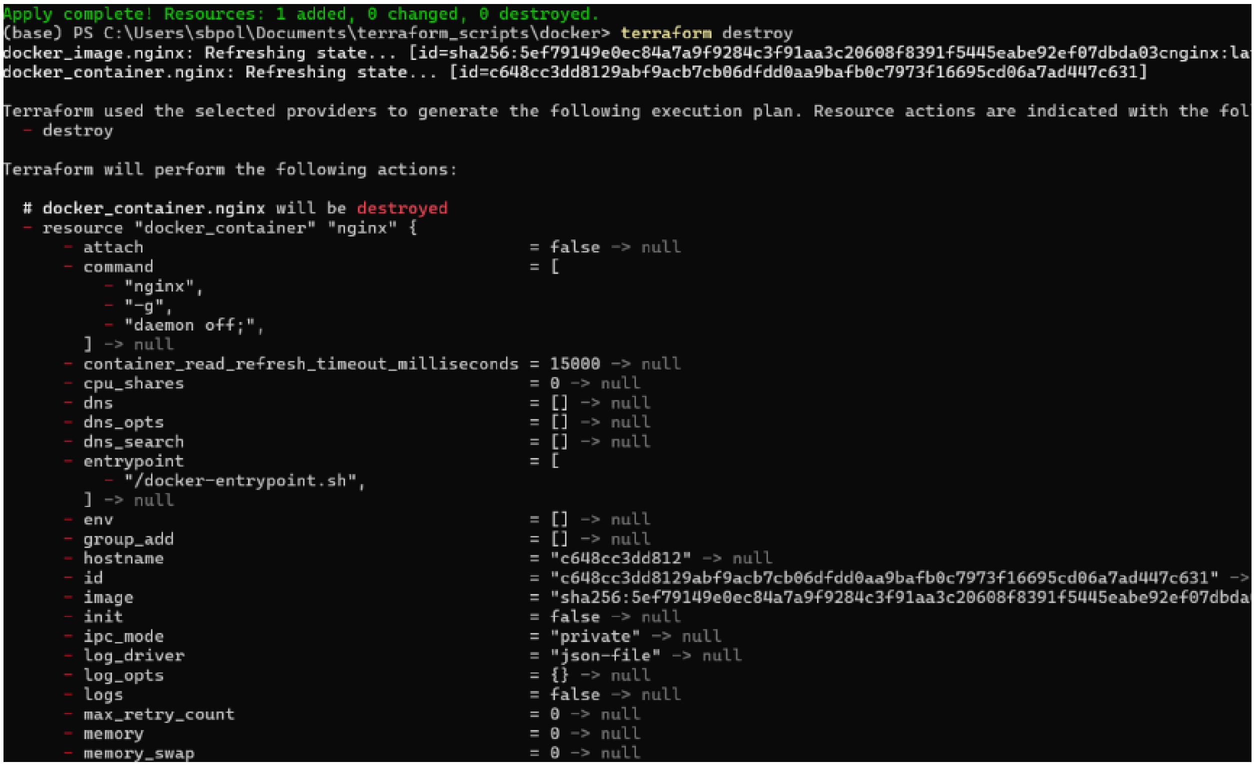
```
PS C:\Users\272241\Terraform_Scripts\Docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

Step 8: Docker images after applying the changes

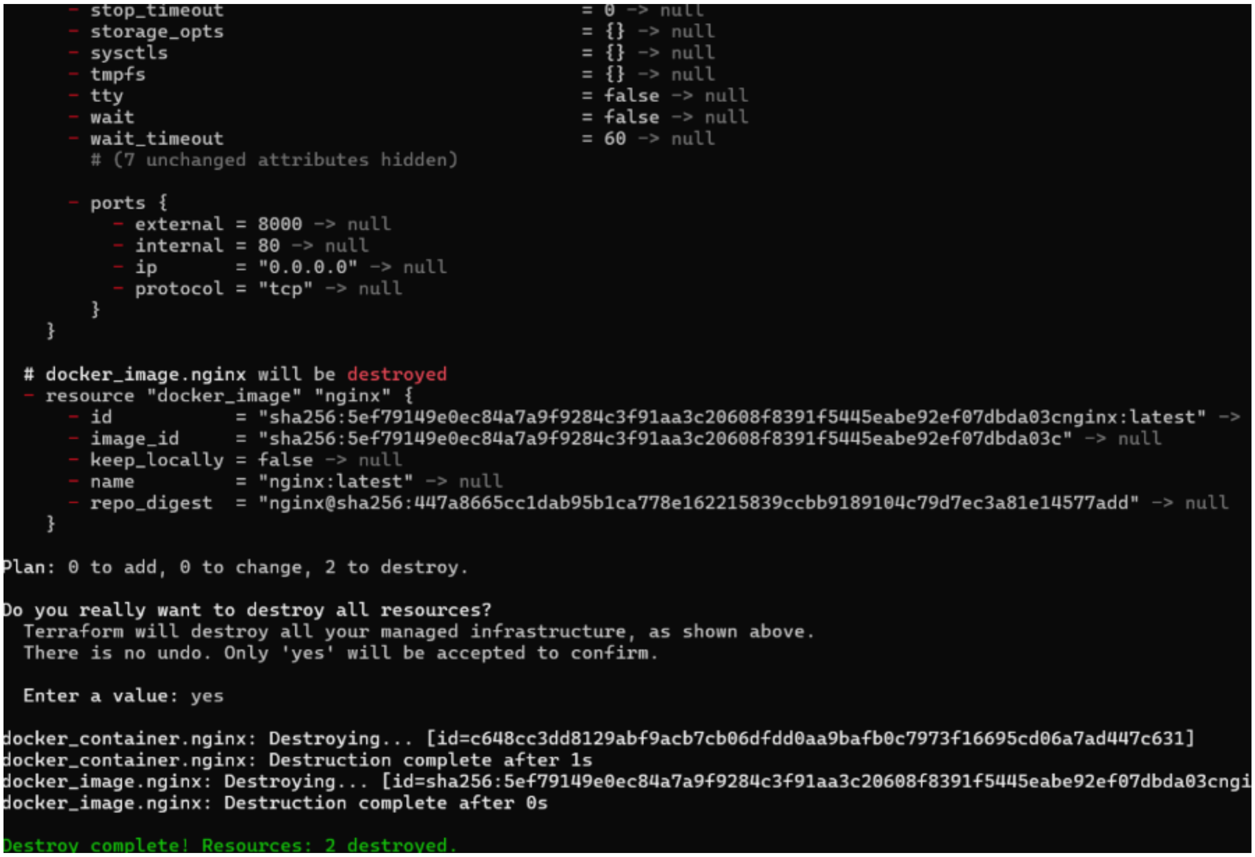


```
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    edbfe74c41f8   3 weeks ago   78.1MB
```

Step 9: Now Terraform Destroy to delete the image ubuntu

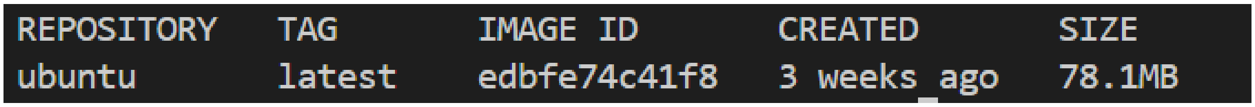


```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> terraform destroy
docker_image.nginx: Refreshing state... [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:la
docker_container.nginx: Refreshing state... [id=c648cc3dd8129abf9acb7cb06dfdd0aa9bafb0c7973f16695cd06a7ad447c631]

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

Terraform will perform the following actions:

  # docker_container.nginx will be destroyed
  - resource "docker_container" "nginx" {
      - attach                                      = false -> null
      - command                                     = [
          - "nginx",
          - "-g",
          - "daemon off;",
        ] -> null
      - container_read_refresh_timeout_milliseconds = 15000 -> null
      - cpu_shares                                  = 0 -> null
      - dns                                         = [] -> null
      - dns_opts                                    = [] -> null
      - dns_search                                  = [] -> null
      - entrypoint                                  = [
          - "/docker-entrypoint.sh",
        ] -> null
      - env                                         = [] -> null
      - group_add                                   = [] -> null
      - hostname                                    = "c648cc3dd812" -> null
      - id                                          = "c648cc3dd8129abf9acb7cb06dfdd0aa9bafb0c7973f16695cd06a7ad447c631" ->
      - image                                       = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda
      - init                                        = false -> null
      - ipc_mode                                    = "private" -> null
      - log_driver                                  = "json-file" -> null
      - log_opts                                    = {} -> null
      - logs                                        = false -> null
      - max_retry_count                             = 0 -> null
      - memory                                      = 0 -> null
      - memory_swap                                 = 0 -> null
```



```
      - stop_timeout                                = 0 -> null
      - storage_opts                                = {} -> null
      - sysctls                                     = {} -> null
      - tmpfs                                       = {} -> null
      - tty                                         = false -> null
      - wait                                        = false -> null
      - wait_timeout                                = 60 -> null
        # (7 unchanged attributes hidden)

      - ports {
          - external = 8000 -> null
          - internal = 80 -> null
          - ip       = "0.0.0.0" -> null
          - protocol = "tcp" -> null
        }
    }

  # docker_image.nginx will be destroyed
  - resource "docker_image" "nginx" {
      - id           = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest" ->
      - image_id     = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03c" -> null
      - keep_locally = false -> null
      - name         = "nginx:latest" -> null
      - repo_digest  = "nginx@sha256:447a8665cc1dab95b1ca778e162215839ccbb9189104c79d7ec3a81e14577add" -> null
    }

Plan: 0 to add, 0 to change, 2 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

docker_container.nginx: Destroying... [id=c648cc3dd8129abf9acb7cb06dfdd0aa9bafb0c7973f16695cd06a7ad447c631]
docker_container.nginx: Destruction complete after 1s
docker_image.nginx: Destroying... [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cngi
docker_image.nginx: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```

Step 10 : Docker after terraform destroy command.



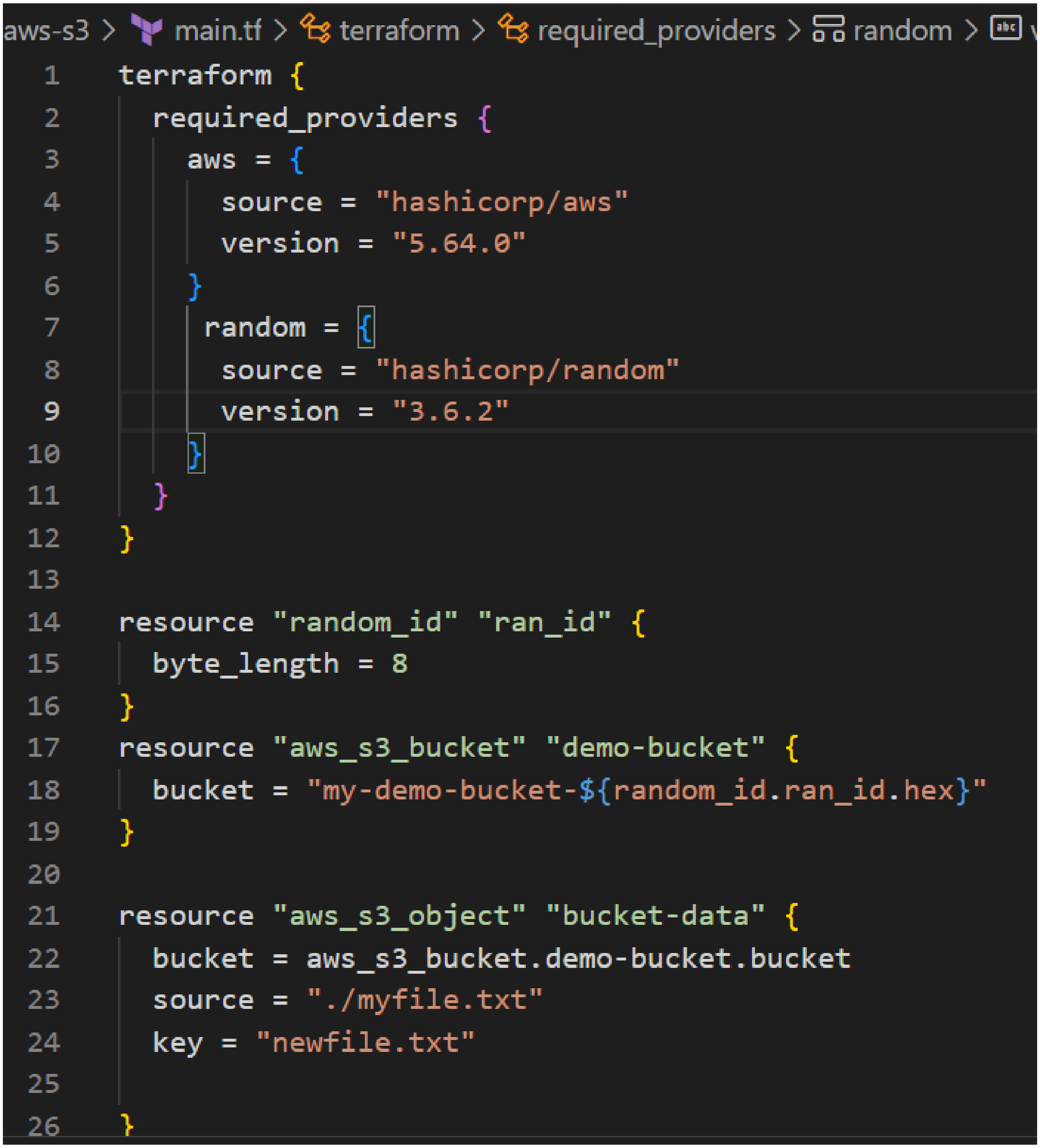
```
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
ubuntu        latest    edbfe74c41f8   3 weeks ago   78.1MB
```

Terraform and S3 -

Step 1: Open VS Code and also log in to your aws account.

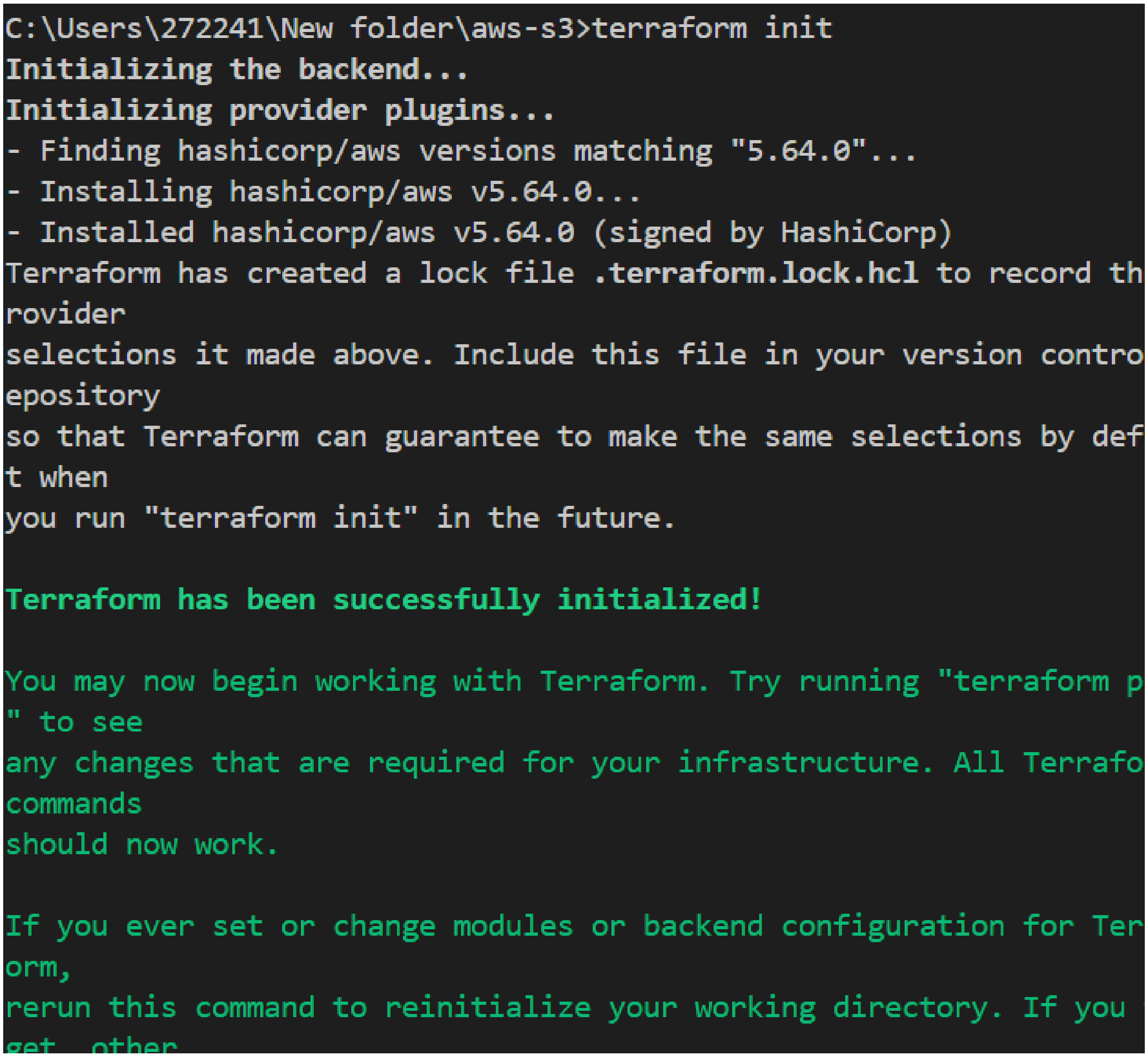
Step 2 : Type below code in main.tf in editor for aws and terraform connection and environment creation .

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.64.0"
    }
     random = {
      source = "hashicorp/random"
      version = "3.6.2"
    }
  }
}

resource "random_id" "ran_id" {
  byte_length = 8
}
resource "aws_s3_bucket" "demo-bucket" {
  bucket = "my-demo-bucket-${random_id.ran_id.hex}"
}

resource "aws_s3_object" "bucket-data" {
  bucket = aws_s3_bucket.demo-bucket.bucket
  source = "./myfile.txt"
  key = "newfile.txt"

}
```


aws-s3 > main.tf > terraform > required_providers > random > v

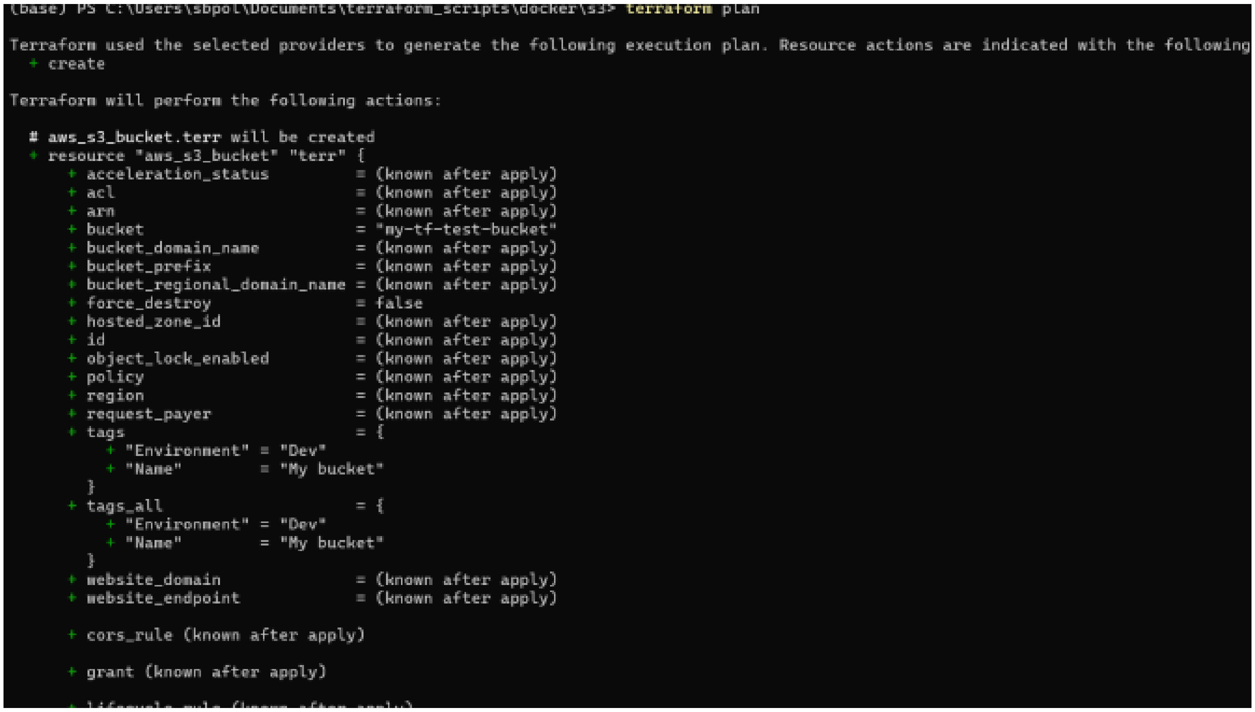
```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.64.0"
    }
     random = {
      source = "hashicorp/random"
      version = "3.6.2"
    }
  }
}

resource "random_id" "ran_id" {
  byte_length = 8
}
resource "aws_s3_bucket" "demo-bucket" {
  bucket = "my-demo-bucket-${random_id.ran_id.hex}"
}

resource "aws_s3_object" "bucket-data" {
  bucket = aws_s3_bucket.demo-bucket.bucket
  source = "./myfile.txt"
  key = "newfile.txt"

}
```

Step 3 : Type terraform init command in powershell.



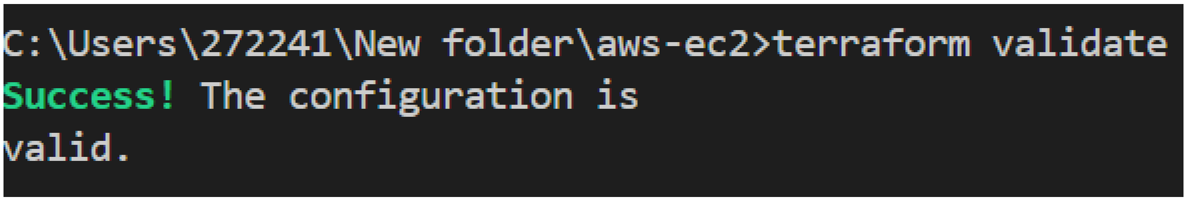
```
C:\Users\272241\New folder\aws-s3>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.64.0"...
- Installing hashicorp/aws v5.64.0...
- Installed hashicorp/aws v5.64.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record th
rovider
selections it made above. Include this file in your version contro
epository
so that Terraform can guarantee to make the same selections by def
t when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform p
" to see
any changes that are required for your infrastructure. All Terrafo
commands
should now work.

If you ever set or change modules or backend configuration for Ter
orm,
rerun this command to reinitialize your working directory. If you
get  other
```

Step 4 : Type terraform plan command in powershell.



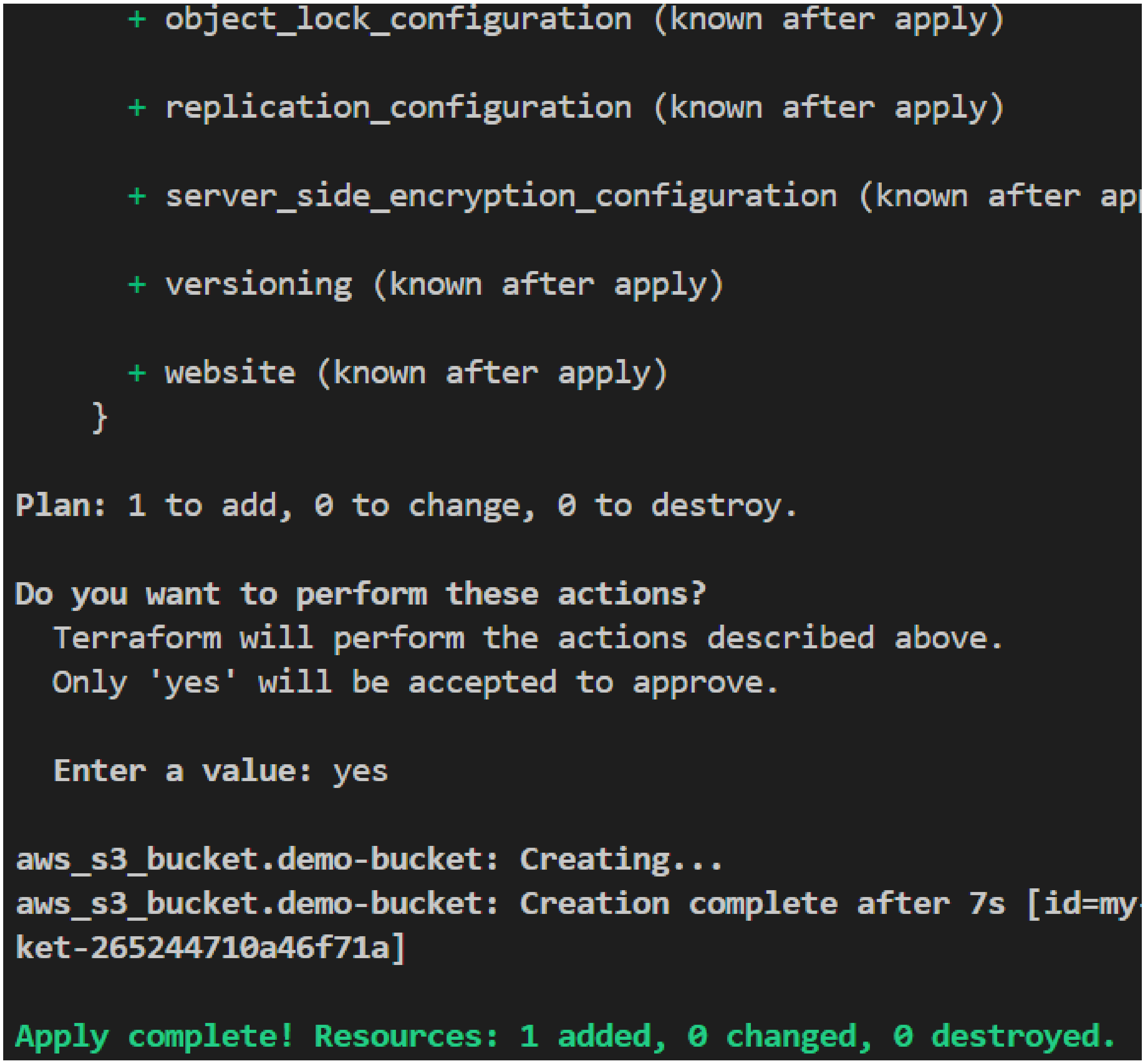
```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform plan

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

Terraform will perform the following actions:

  # aws_s3_bucket.terr will be created
  + resource "aws_s3_bucket" "terr" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "my-tf-test-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"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)

      + lifecycle_rule (known after apply)
```

Step 5 Type terraform validate



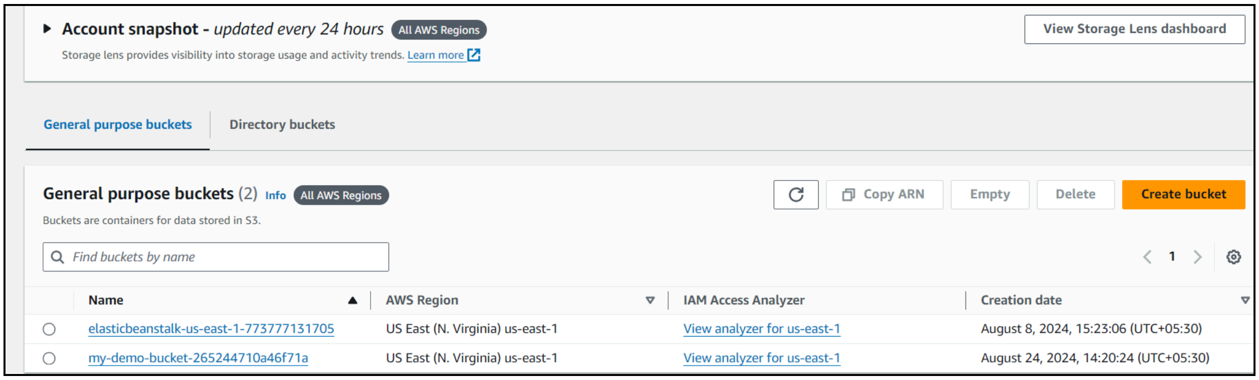
```
C:\Users\272241\New folder\aws-ec2>terraform validate
Success! The configuration is
valid.
```

Step 6 : Type terraform apply command in powershell



```
      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after ap

      + 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.demo-bucket: Creating...
aws_s3_bucket.demo-bucket: Creation complete after 7s [id=my-
ket-265244710a46f71a]

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

Step 7 : Now check AWS to see the bucket created:



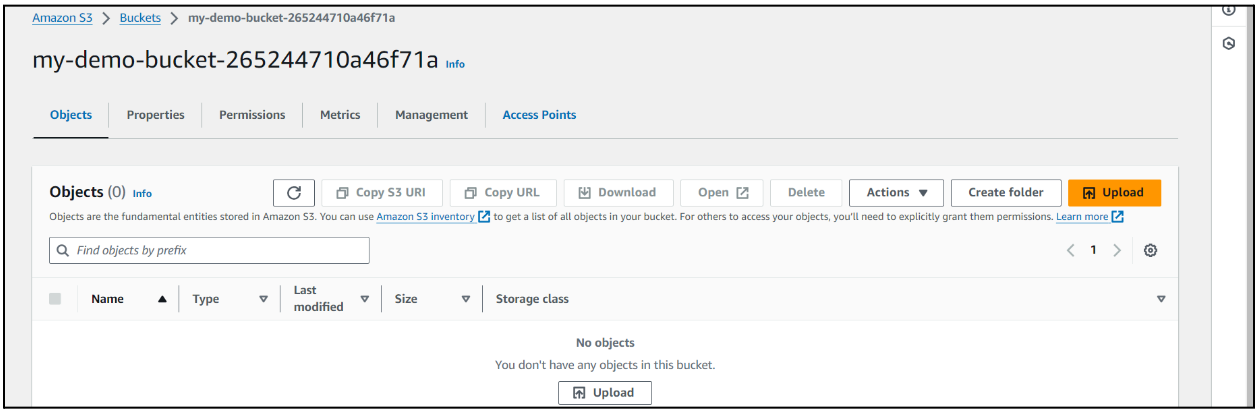
Account snapshot - *updated every 24 hours* All AWS Regions

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

General purpose buckets (2) Info All AWS Regions

Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
○	elasticbeanstalk-us-east-1-773777131705	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 8, 2024, 15:23:06 (UTC+05:30)
○	my-demo-bucket-265244710a46f71a	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 24, 2024, 14:20:24 (UTC+05:30)



my-demo-bucket-265244710a46f71a Info

Objects (0) Info

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

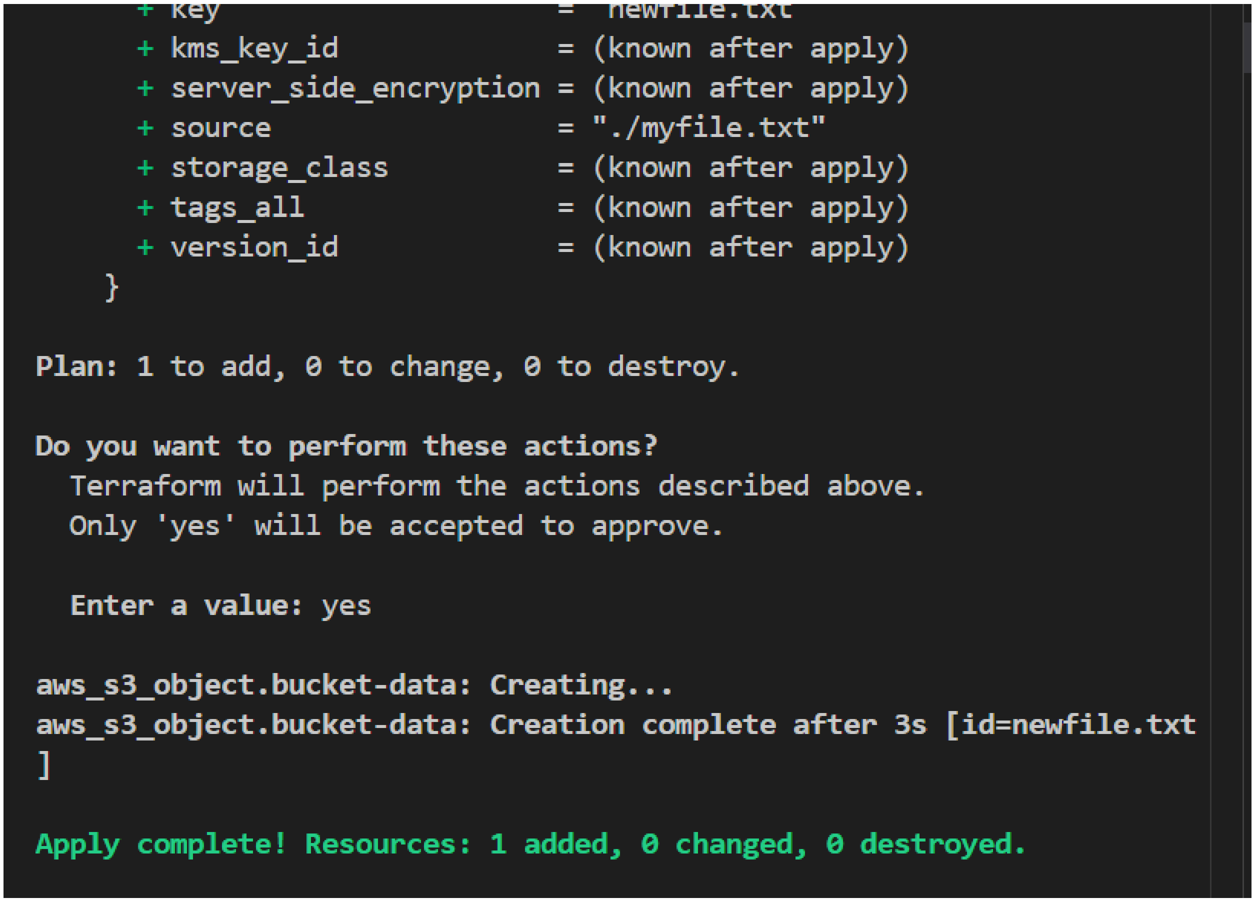
	Name	Type	Last modified	Size	Storage class

No objects

You don't have any objects in this bucket.

Upload

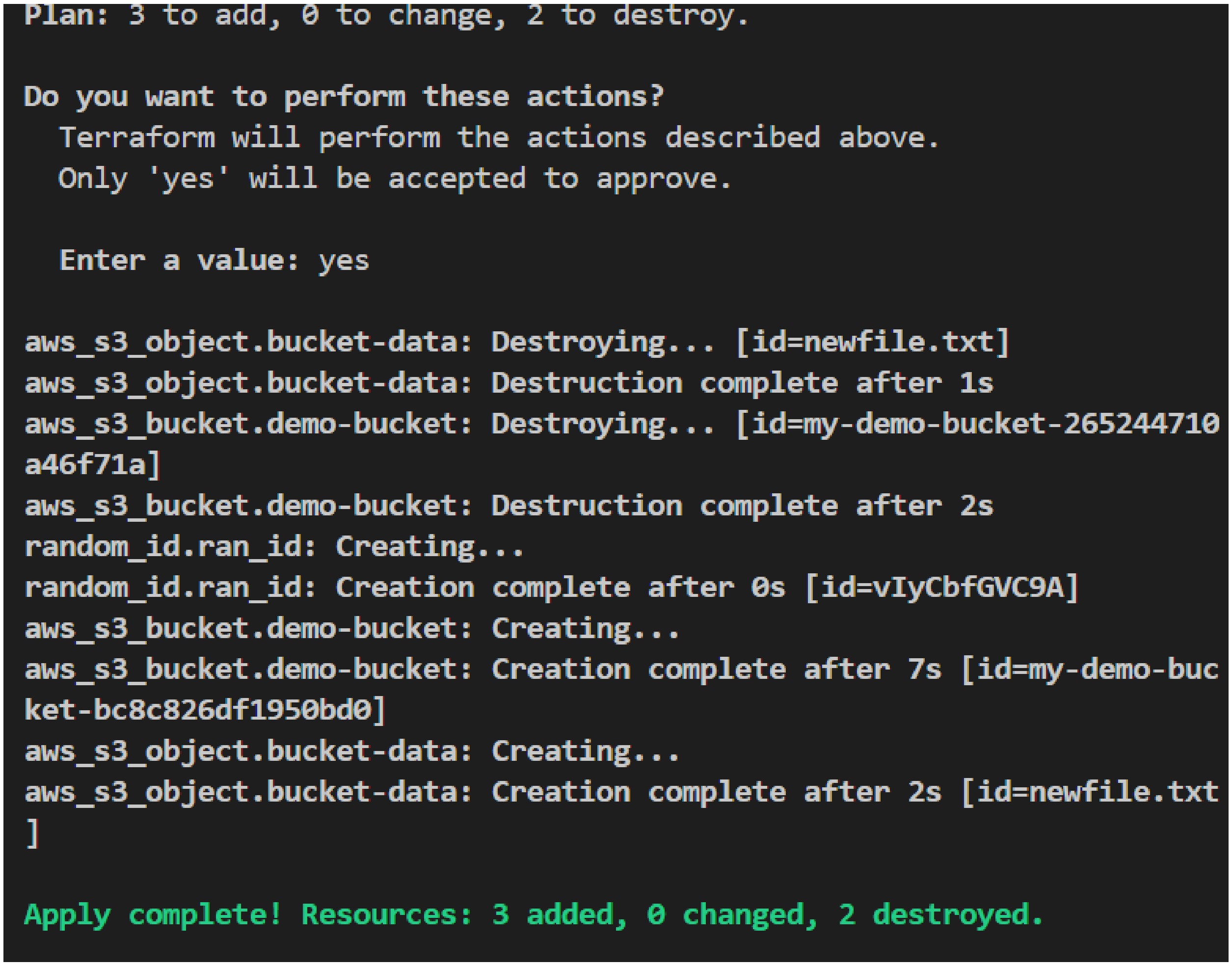
Step 8(EXTRA) :Terraform plan and apply command to apply the changes for file.



```
      + key                    = "newfile.txt"
      + kms_key_id             = (known after apply)
      + server_side_encryption = (known after apply)
      + source                 = "./myfile.txt"
      + storage_class          = (known after apply)
      + tags_all               = (known after apply)
      + version_id             = (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_object.bucket-data: Creating...
aws_s3_object.bucket-data: Creation complete after 3s [id=newfile.txt
]

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

Step 9: Check the s3 bucket created before and after uploading the file.

Step 10 : Terraform destroy command to destroy the s3 bucket.



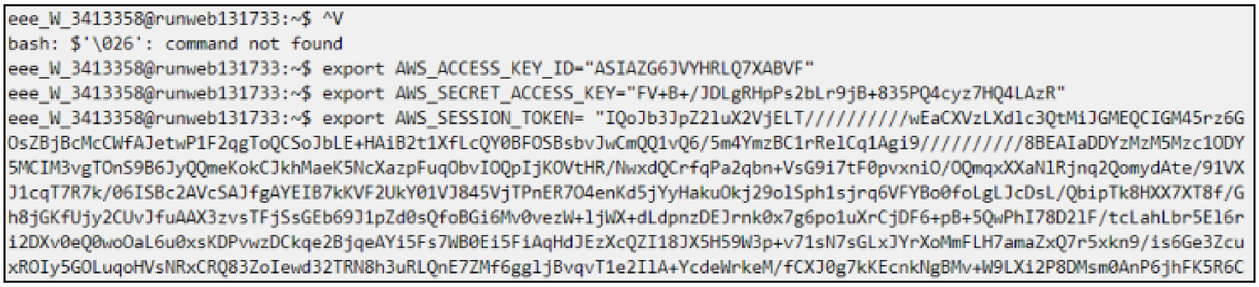
```
Plan: 3 to add, 0 to change, 2 to destroy.

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_object.bucket-data: Destroying... [id=newfile.txt]
aws_s3_object.bucket-data: Destruction complete after 1s
aws_s3_bucket.demo-bucket: Destroying... [id=my-demo-bucket-265244710
a46f71a]
aws_s3_bucket.demo-bucket: Destruction complete after 2s
random_id.ran_id: Creating...
random_id.ran_id: Creation complete after 0s [id=vIyCbfGVC9A]
aws_s3_bucket.demo-bucket: Creating...
aws_s3_bucket.demo-bucket: Creation complete after 7s [id=my-demo-buc
ket-bc8c826df1950bd0]
aws_s3_object.bucket-data: Creating...
aws_s3_object.bucket-data: Creation complete after 2s [id=newfile.txt
]

Apply complete! Resources: 3 added, 0 changed, 2 destroyed.
```

(EXTRA)

EC2:

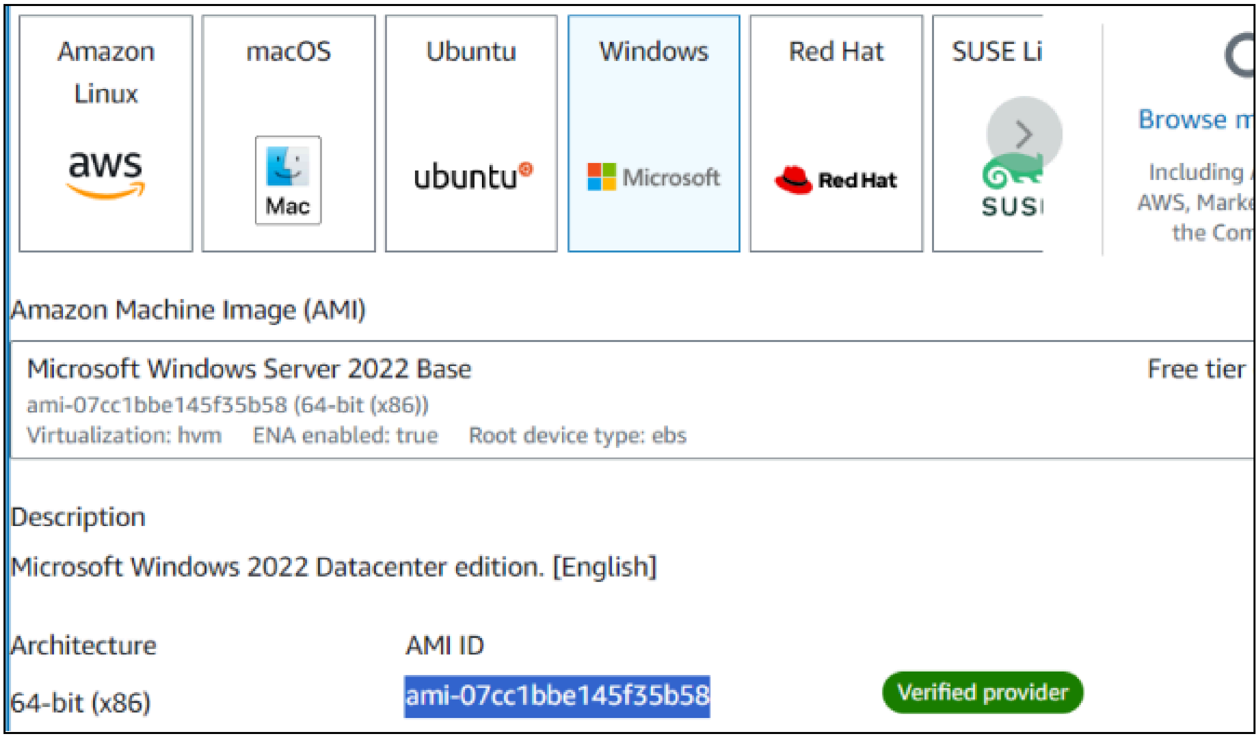
Creating EC2 instance using Terraform

Step 1 : connect the aws academy and terraform using the credentials



```
eee_W_3413358@runweb131733:~$ ^V
bash: $'\026': command not found
eee_W_3413358@runweb131733:~$ export AWS_ACCESS_KEY_ID="ASIAZG6JVYHRLQ7XABVF"
eee_W_3413358@runweb131733:~$ export AWS_SECRET_ACCESS_KEY="FV+B+/JDLgRHpPs2bLr9jB+835PQ4cyz7HQ4LAzR"
eee_W_3413358@runweb131733:~$ export AWS_SESSION_TOKEN= "IQoJb3JpZ2luX2VjELT//////////wEaCXVzLXdlc3QtMiJGMEQCIGM45rz6G
OsZBjBcMcCWfAJetwP1F2qgToQCSoJbLE+HAiB2t1XfLcQY0BFOSBsbvJwCmQQ1vQ6/5m4YmzBC1rRelCq1Agi9//////////8BEAIaDDYzMzM5Mzc1ODY
5MCIM3vgTOnS9B6JyQQmeKokCJkhMaeK5NcXazpFuqObvIOQpIjKOVtHR/NwxdQCrfqPa2qbn+VsG9i7tF0pvxniO/OQmqxXXaNlRjnq2QomydAte/91VX
J1cqT7R7k/06ISBc2AVcSAJfgAYEIB7kKVF2UkY01VJ845VjTPnER7O4enKd5jYyHakuOkj29olSph1sjrq6VFYBo0foLgLJcDsL/QbipTk8HXX7XT8f/G
h8jGKfUjy2CUvJfuAAX3zvsTFjSsGEb69J1pZd0sQfoBGi6Mv0vezW+1jWX+dLdpnzDEJrnk0x7g6po1uXrCjDF6+pB+5QwPhI78D21F/tcLahLbr5E16r
i2DXv0eQ0woOaL6u0xsKDPvwzDCkqe2BjqeAYi5Fs7WB0Ei5FiAqHdJEzXcQZI18JX5H59W3p+v71sN7sGLxJYrXoMmFLH7amaZxQ7r5xkn9/is6Ge3Zcu
xROIy5GOLuqoHVsNRxCRQ83ZoIewd32TRN8h3uRLQnE7ZMf6gg1jBvqvT1e2I1A+YcdeWrkeM/fCXJ0g7kKEcnkNgBMv+W9LXi2P8DMsm0AnP6jhFK5R6C
```

Step 2 : copy the AMI ID from the EC2



Amazon Machine Image (AMI)

Microsoft Windows Server 2022 Base
ami-07cc1bbe145f35b58 (64-bit (x86))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier

Description

Microsoft Windows 2022 Datacenter edition. [English]

Architecture

64-bit (x86)

AMI ID

ami-07cc1bbe145f35b58

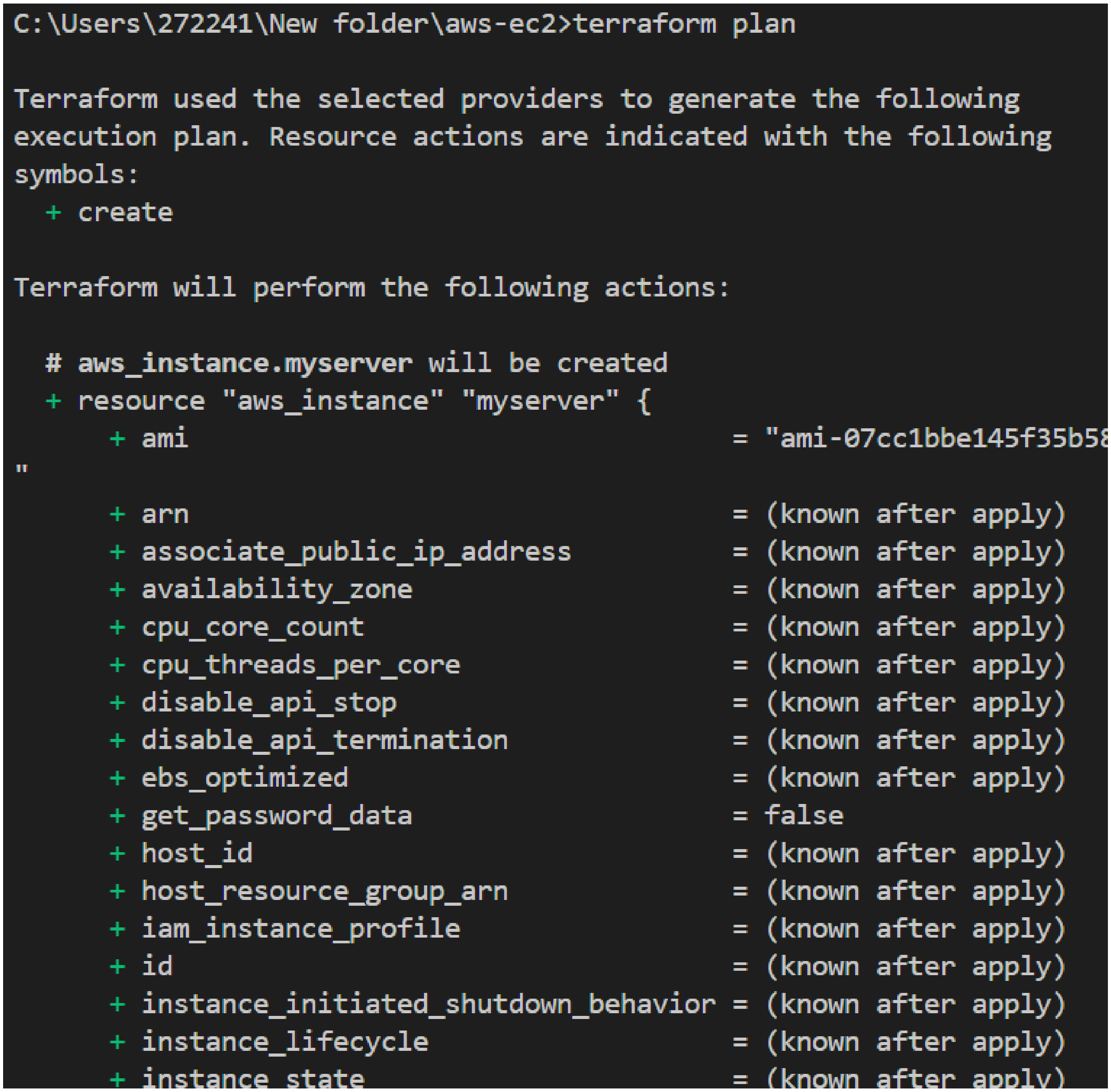
Verified provider

Step 3 : Create the main.tf and provider.tf

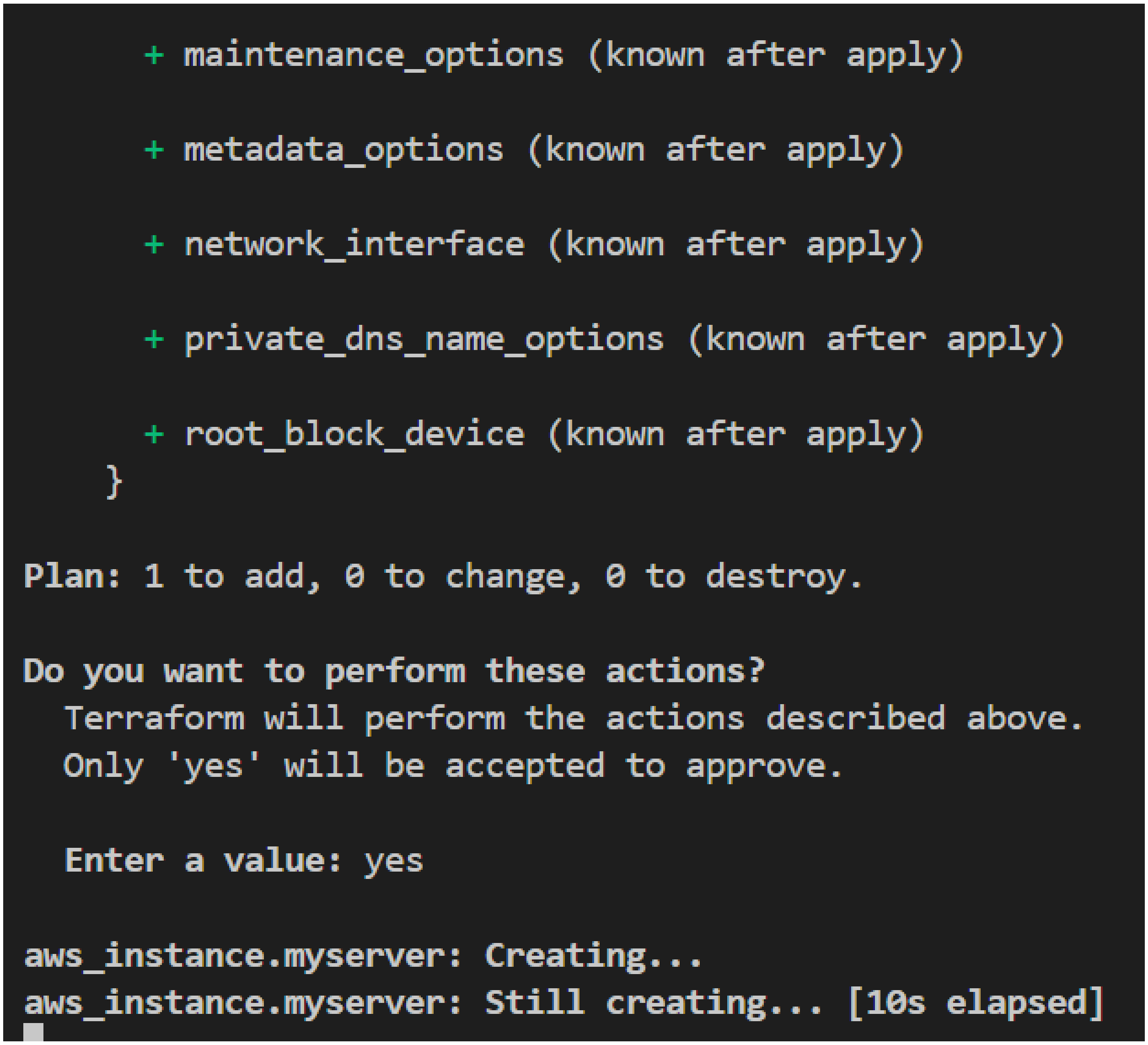
```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.64.0"
    }
  }
}

resource "aws_instance" "myserver" {
    ami = "ami-066784287e358dad1"
    instance_type = "var.instance_type"
    tags = {
      Name="SampleServer"
    }
}
```

```
provider "aws" {
  access_key="ASIA3IKFWBC4XD2NUOGT"
  secret_key="HV1nehMF9eHDuMPb3kffqN4S9FgWiuyRt0FtKMN7"

token="IQoJb3JpZ2luX2VjELD//////////wEaCXVzLXdlc3QtMiJIMEYCIQDUStpPY4WH1rL
xLjK+gWmrGVsyUjHDszjELoj/+ODmOwIhAJXxnswL/ZId9I3+CvTGQhOaUaRaA03FF/e2QqXOP
FLJKrECCLj//////////wEQABoMNzczNzc3MTMxNzA1IgzAQdzV3pYlWwUUeVsqhQLrEFCk8N6
Y0xynEV4qLqSbfQ3gS074l976p9R7hyn15nT+PkAR2uytbSQfDD2XceXD0KTSF0F1GqHEtrTyI
RM3y5wbWdHj/3X7WPSgMa2b04vln+9LJehMT3naBzqtUxO3qauygsxlrgnhKF3Necr4jTjy5kU
ioPh3rm53pNh07nXAXH2W1WB9HUeHWS8Fp2x698cN2pTINjzjJ5UaO8ouuSOfeknDZweadm2u2
SPA5UjDk8xjsHc/RWXrrVZ8RTMYI6yBEml2NStiR2txQXNT8g0zf/rgJz6gWSLVfvW3Vk6SCI1
iMupxPYE+JpFdsyt4+AL/MLxFpQ12jg5rQJGbhhUnOYww9JKmtgY6nAHrmQPgQaSyPFsJETANO
kOUV/zjwEVL8KlzEHip2u22rYivpsTyoOkfKZibT9CtycdSMaOQgjFj5kX1ASnxIxocdVMVMSj
2cWn2JwskGcOcJjuvL5ZbKmh/T8cAKgDJTTaFdPKZCO6yVGQ/RB3REeCKmvkAE9yPjjcHclGcf
DJnaVS8neVyky2xwGBNGhBwnoWc3Ol+LQ22V9dmFWs="
  region = "us-east-1"
}
```

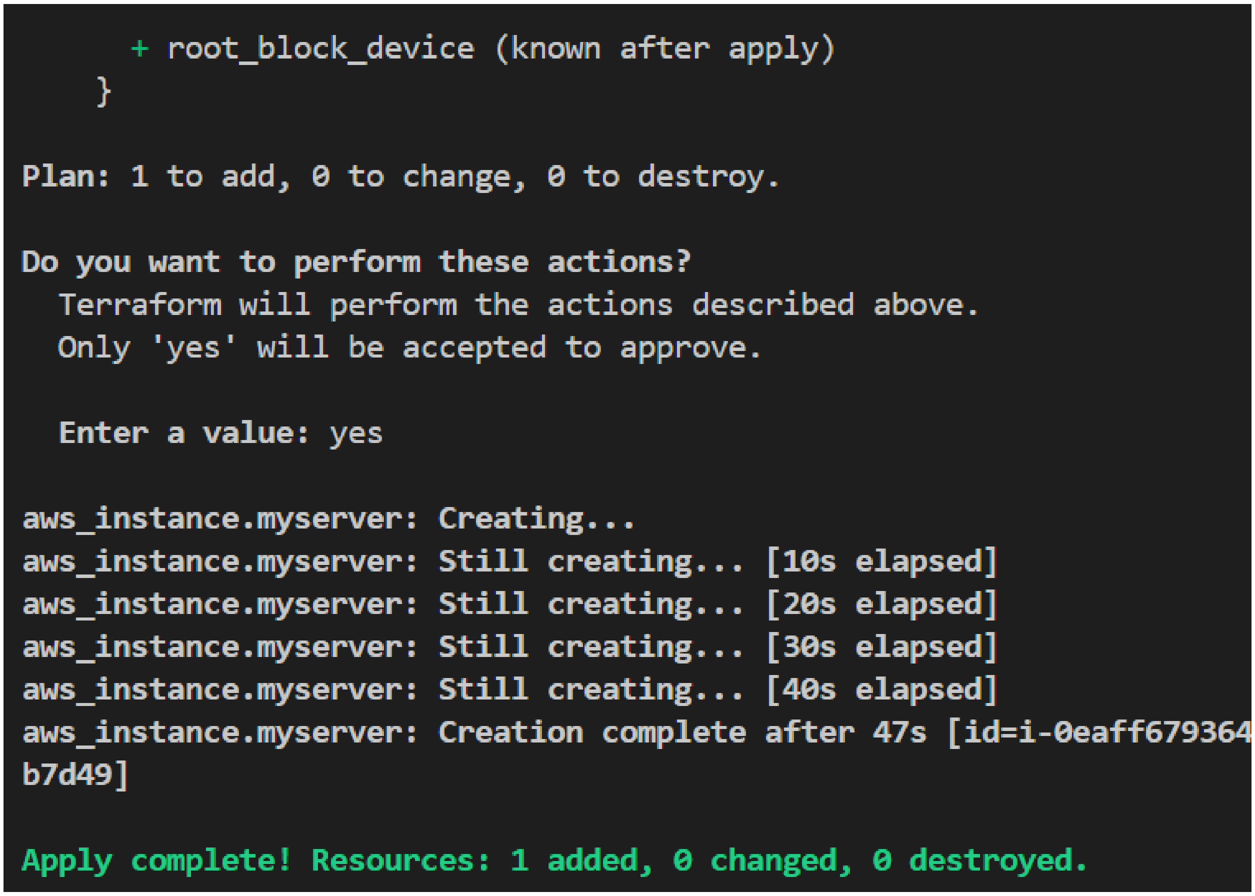
Step 4 : Execute terraform init , terraform plan and terraform apply command



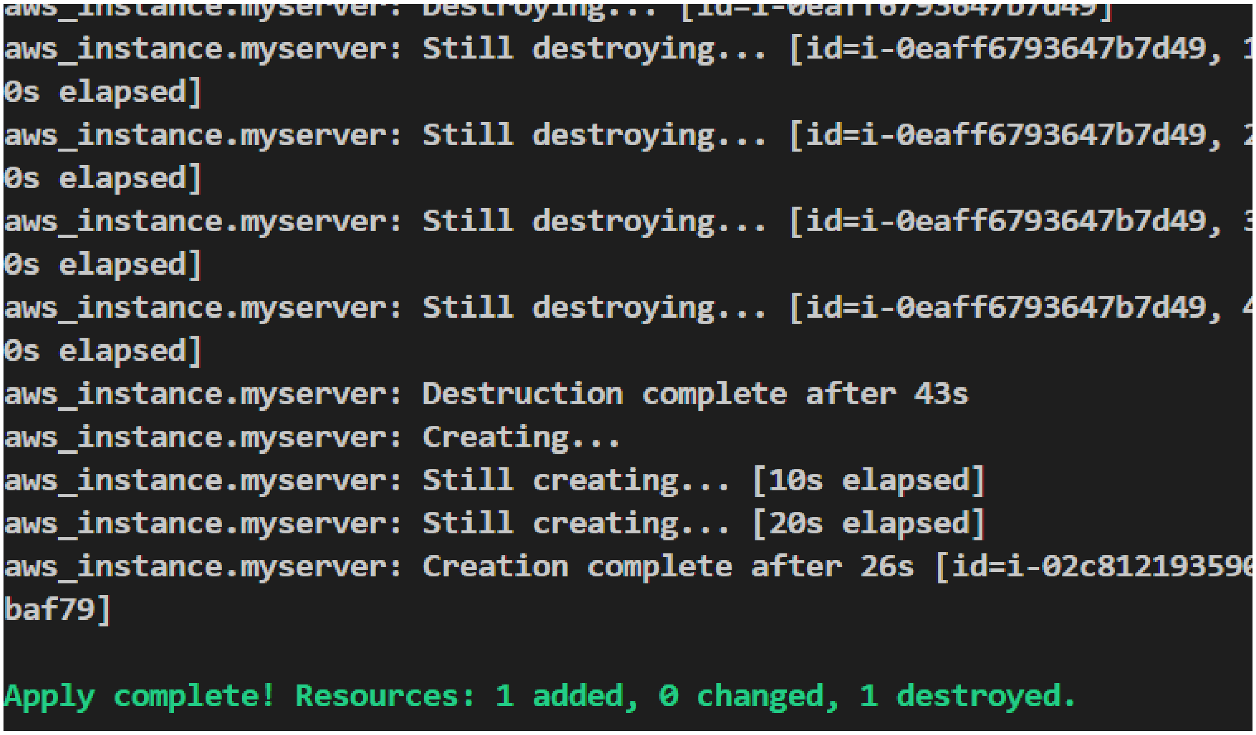
```
C:\Users\272241\New folder\aws-ec2>terraform plan

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_instance.myserver will be created
  + resource "aws_instance" "myserver" {
      + ami                                  = "ami-07cc1bbe145f35b58
"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
```



```
      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (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_instance.myserver: Creating...
aws_instance.myserver: Still creating... [10s elapsed]
```

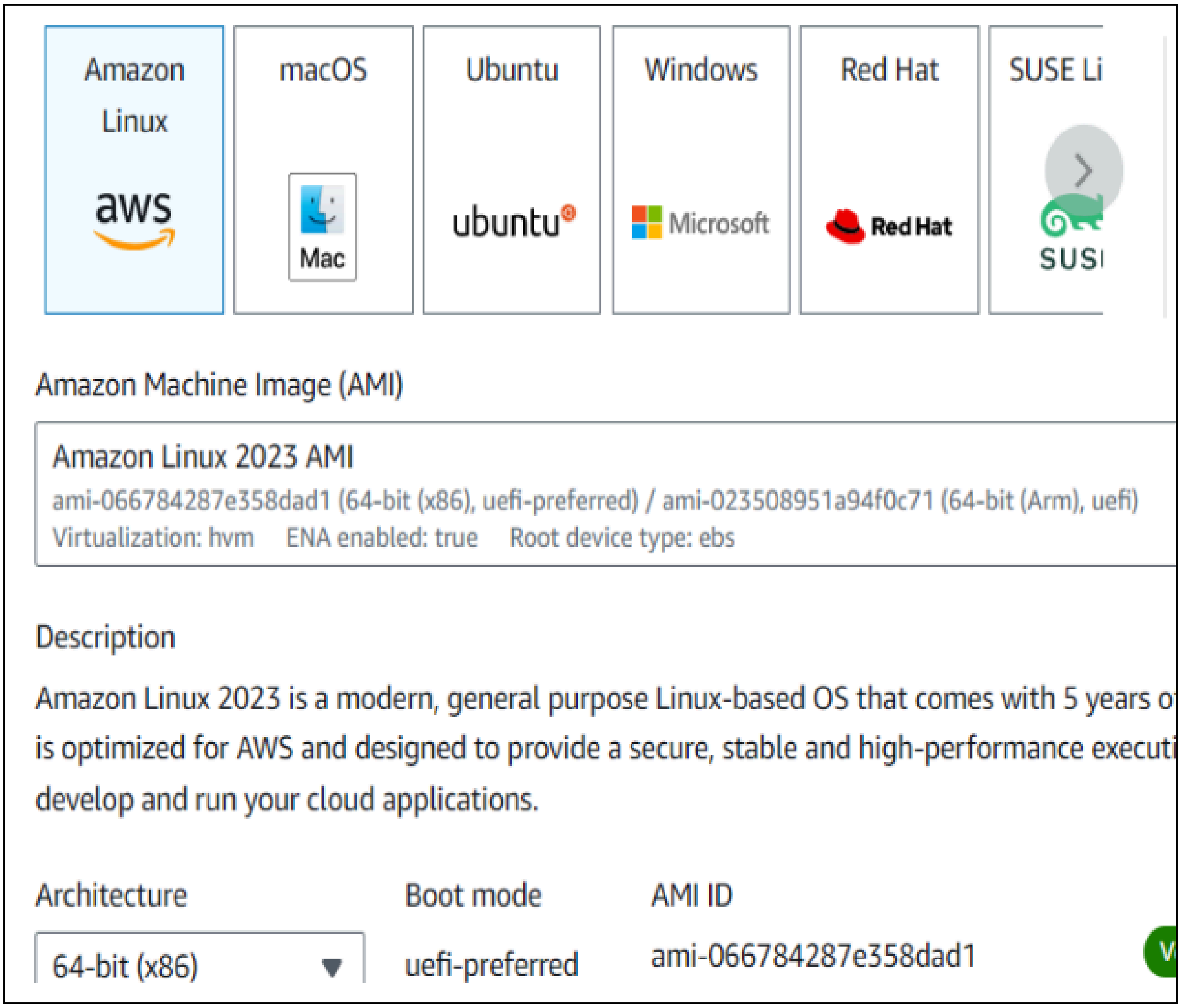


```
      + root_block_device (known after apply)
    }

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_instance.myserver: Creating...
aws_instance.myserver: Still creating... [10s elapsed]
aws_instance.myserver: Still creating... [20s elapsed]
aws_instance.myserver: Still creating... [30s elapsed]
aws_instance.myserver: Still creating... [40s elapsed]
aws_instance.myserver: Creation complete after 47s [id=i-0eaff679364
b7d49]

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



```
aws_instance.myserver: Destroying... [id=i-0eaff6793647b7d49]
aws_instance.myserver: Still destroying... [id=i-0eaff6793647b7d49, 1
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0eaff6793647b7d49, 2
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0eaff6793647b7d49, 3
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0eaff6793647b7d49, 4
0s elapsed]
aws_instance.myserver: Destruction complete after 43s
aws_instance.myserver: Creating...
aws_instance.myserver: Still creating... [10s elapsed]
aws_instance.myserver: Still creating... [20s elapsed]
aws_instance.myserver: Creation complete after 26s [id=i-02c81219359
baf79]

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

Step 5 : Ec2 before and after instance creation .
Step 6 : Copy AWS AMI ID and change it in code



Amazon Machine Image (AMI)

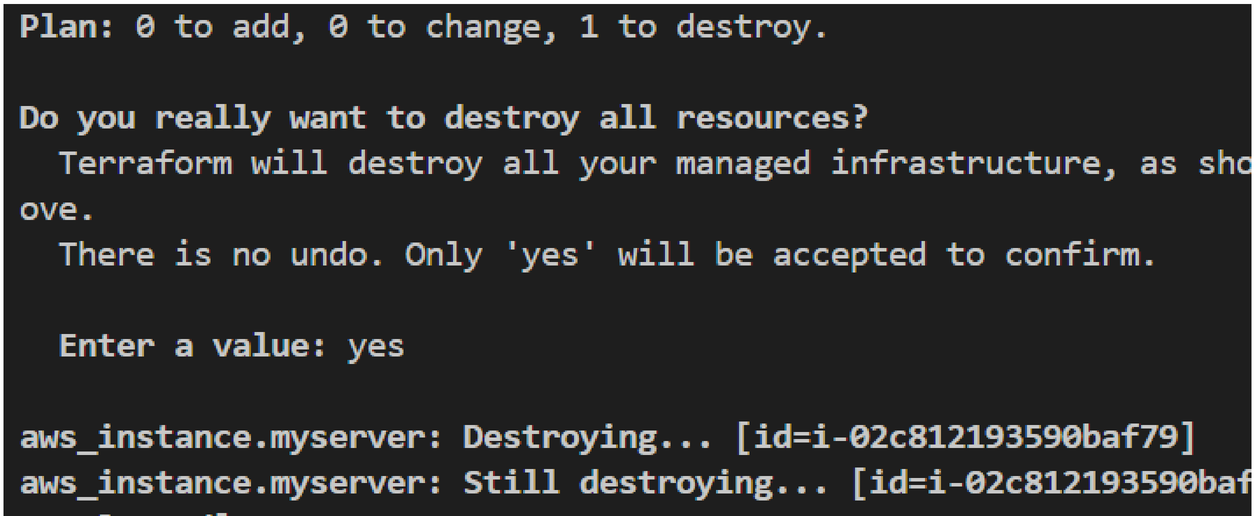
Amazon Linux 2023 AMI
ami-066784287e358dad1 (64-bit (x86), uefi-preferred) / ami-023508951a94f0c71 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

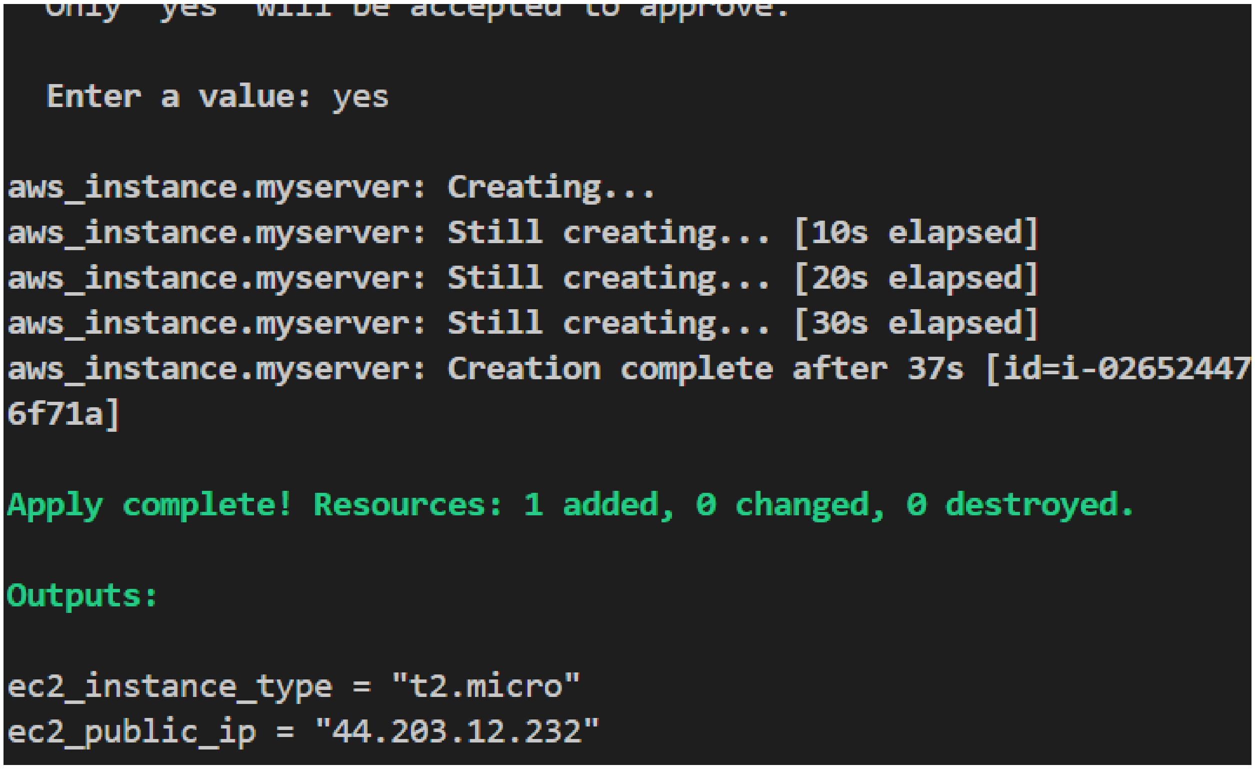
Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years o
is optimized for AWS and designed to provide a secure, stable and high-performance executi
develop and run your cloud applications.

Architecture	Boot mode	AMI ID
64-bit (x86) ▼	uefi-preferred	ami-066784287e358dad1

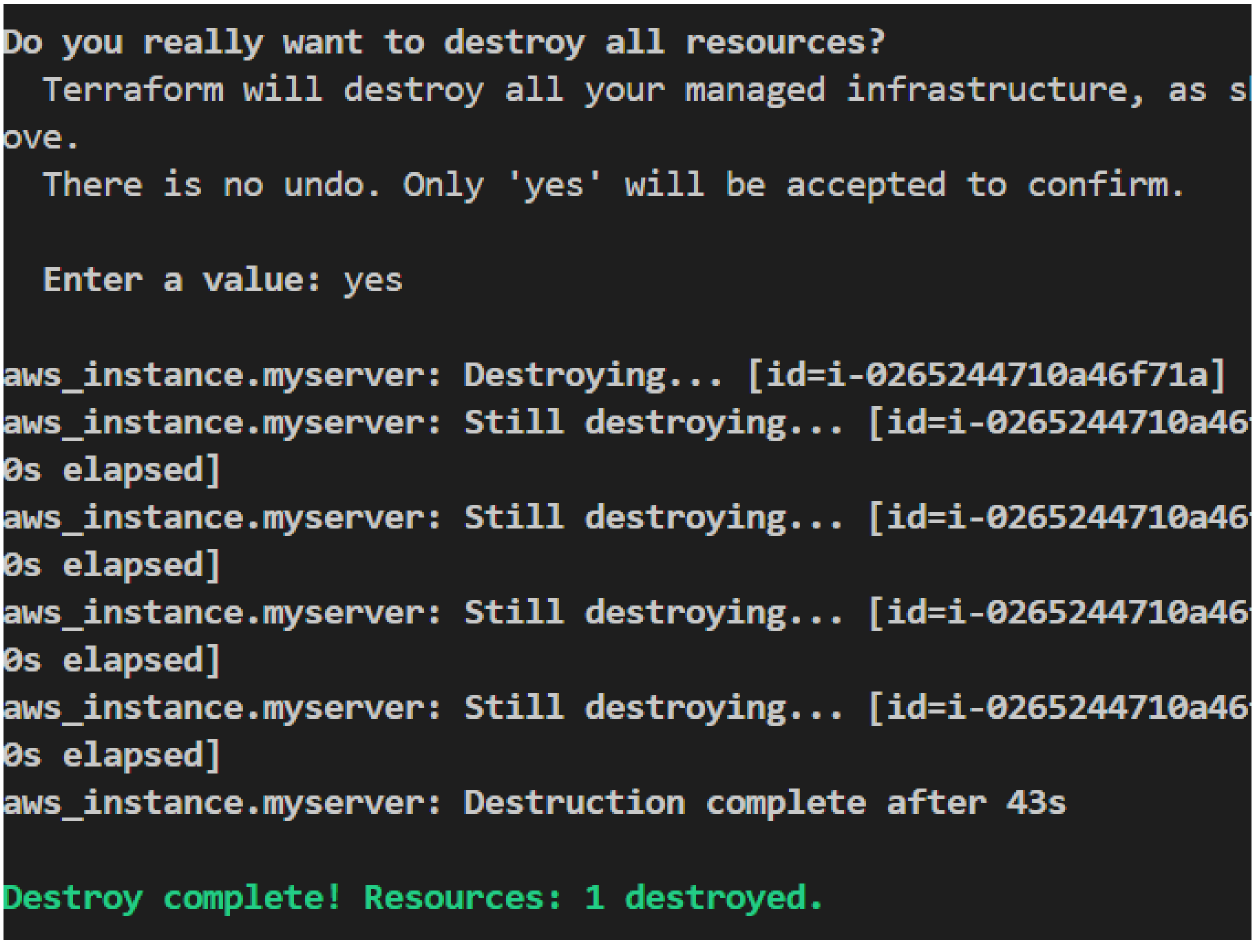
Step 9 : Destroy the instance using terraform destroy



```
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 sho
ove.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.myserver: Destroying... [id=i-02c812193590baf79]
aws_instance.myserver: Still destroying... [id=i-02c812193590baf
```



```
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.myserver: Creating...
aws_instance.myserver: Still creating... [10s elapsed]
aws_instance.myserver: Still creating... [20s elapsed]
aws_instance.myserver: Still creating... [30s elapsed]
aws_instance.myserver: Creation complete after 37s [id=i-02652447
6f71a]

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

Outputs:

ec2_instance_type = "t2.micro"
ec2_public_ip = "44.203.12.232"
```



```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as sh
ove.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.myserver: Destroying... [id=i-0265244710a46f71a]
aws_instance.myserver: Still destroying... [id=i-0265244710a46
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0265244710a46
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0265244710a46
0s elapsed]
aws_instance.myserver: Still destroying... [id=i-0265244710a46
0s elapsed]
aws_instance.myserver: Destruction complete after 43s

Destroy complete! Resources: 1 destroyed.
```

(EXTRA)

Hosting Website on s3 using Terraform-
Static website hosting:

Step 1 : create main.tf and write following code
Code -

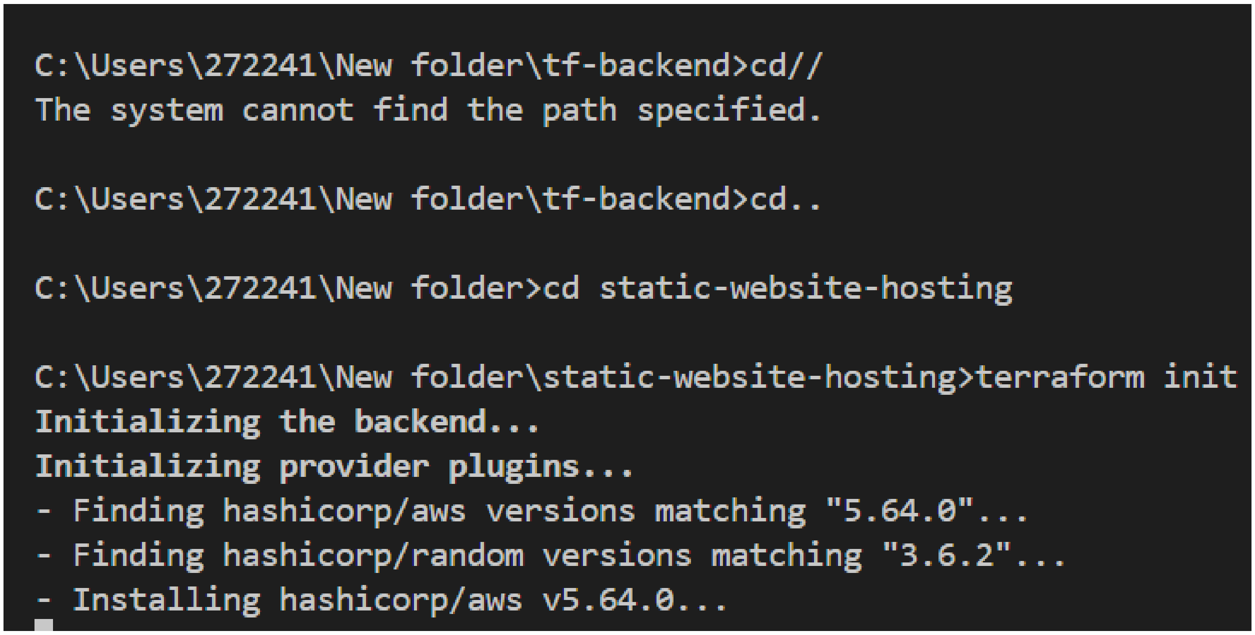
```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.64.0"
    }
     random = {
      source = "hashicorp/random"
      version = "3.6.2"
    }
  }
}

resource "random_id" "ran_id" {
  byte_length = 8
}
resource "aws_s3_bucket" "mywebapp-bucket" {
  bucket = "my-mywebapp-bucket-${random_id.ran_id.hex}"
}

resource "aws_s3_object" "index_html" {
  bucket = aws_s3_bucket.mywebapp-bucket.bucket
  source = "./index.html"
  key = "index.html"
  content_type = "text/html"

}

resource "aws_s3_object" "style_css" {
  bucket = aws_s3_bucket.mywebapp-bucket.bucket
  source = "./styles.css"
  key = "styles.css"
  content_type = "text/css"

}

resource "aws_s3_bucket_public_access_block" "example" {
  bucket = aws_s3_bucket.mywebapp-bucket.id
  block_public_acls       = false
  block_public_policy     = false
  ignore_public_acls      = false
  restrict_public_buckets = false
}

resource "aws_s3_bucket_policy" "staticwebnew" {
   bucket = aws_s3_bucket.mywebapp-bucket.id
    policy = jsonencode(
        {
    Version = "2012-10-17",
    Statement = [
        {
            Sid = "PublicReadGetObject",
            Effect = "Allow",
            Principal = "*",
            Action = "s3:GetObject",
                                                Resource     =
"arn:aws:s3:::${aws_s3_bucket.mywebapp-bucket.id}/*"
        }
    ]
}
    )
}

resource "aws_s3_bucket_website_configuration" "example" {
  bucket = aws_s3_bucket.mywebapp-bucket.id

  index_document {
    suffix = "index.html"
  }
}

output "website_endpoint" {
  value = aws_s3_bucket_website_configuration.example.website_endpoint

}
```

Step 2 : Create Provider.tf and write following code

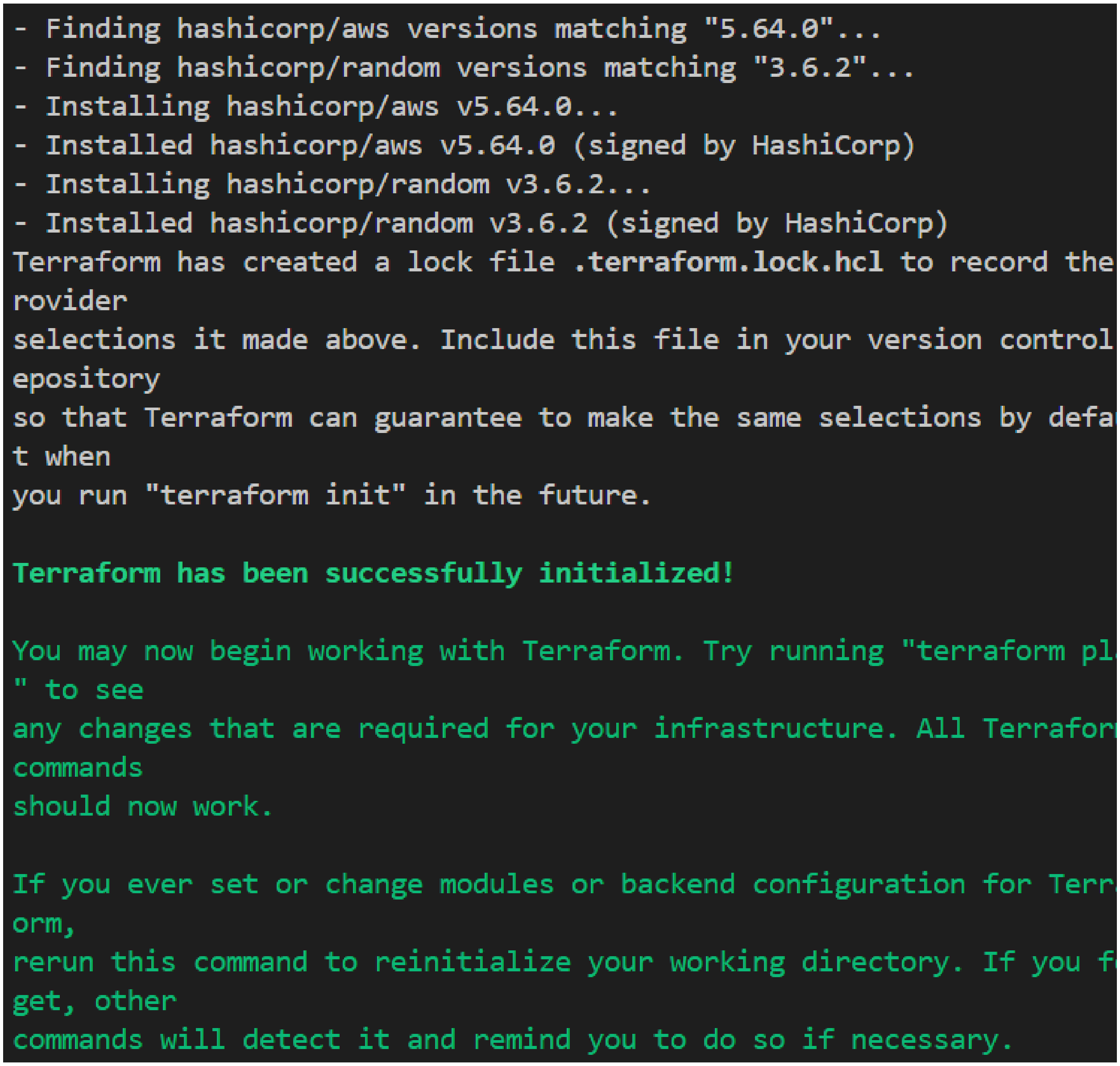
Code:

```
provider "aws" {
  access_key="ASIA3IKFWBC4XD2NUOGT"
  secret_key="HV1nehMF9eHDuMPb3kffqN4S9FgWiuyRt0FtKMN7"

token="IQoJb3JpZ2luX2VjELD//////////wEaCXVzLXdlc3QtMiJIMEYCIQDUStpPY4WH1rL
xLjK+gWmrGVsyUjHDszjELoj/+ODmOwIhAJXxnswL/ZId9I3+CvTGQhOaUaRaA03FF/e2QqXOP
FLJKrECCLj//////////wEQABoMNzczNzc3MTMxNzA1IgzAQdzV3pYlWwUUeVsqhQLrEFCk8N6
Y0xynEV4qLqSbfQ3gS074l976p9R7hyn15nT+PkAR2uytbSQfDD2XceXD0KTSF0F1GqHEtrTyI
RM3y5wbWdHj/3X7WPSgMa2b04vln+9LJehMT3naBzqtUxO3qauygsxlrgnhKF3Necr4jTjy5kU
ioPh3rm53pNh07nXAXH2W1WB9HUeHWS8Fp2x698cN2pTINjzjJ5UaO8ouuSOfeknDZweadm2u2
SPA5UjDk8xjsHc/RWXrrVZ8RTMYI6yBEml2NStiR2txQXNT8g0zf/rgJz6gWSLVfvW3Vk6SCI1
iMupxPYE+JpFdsyt4+AL/MLxFpQ12jg5rQJGbhhUnOYww9JKmtgY6nAHrmQPgQaSyPFsJETANO
kOUV/zjwEVL8KlzEHip2u22rYivpsTyoOkfKZibT9CtycdSMaOQgjFj5kX1ASnxIxocdVMVMSj
2cWn2JwskGcOcJjuvL5ZbKmh/T8cAKgDJTTaFdPKZCO6yVGQ/RB3REeCKmvkAE9yPjjcHclGcf
DJnaVS8neVyky2xwGBNGhBwnoWc3Ol+LQ22V9dmFWs="
  region = "us-east-1"
}
```

Step 3: Execute Terraform init command.

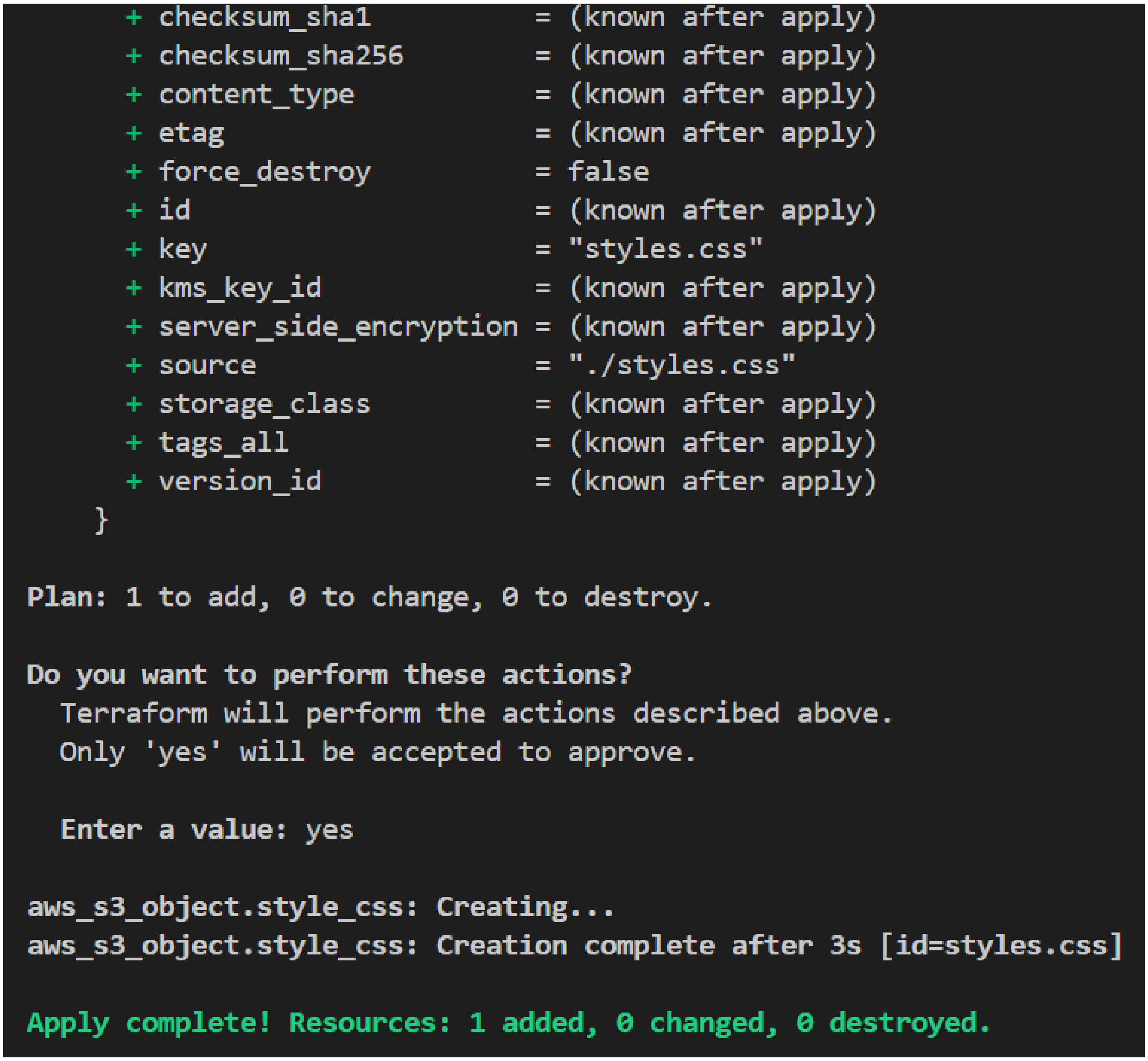


```
C:\Users\272241\New folder\tf-backend>cd//
The system cannot find the path specified.

C:\Users\272241\New folder\tf-backend>cd..

C:\Users\272241\New folder>cd static-website-hosting

C:\Users\272241\New folder\static-website-hosting>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.64.0"...
- Finding hashicorp/random versions matching "3.6.2"...
- Installing hashicorp/aws v5.64.0...
```

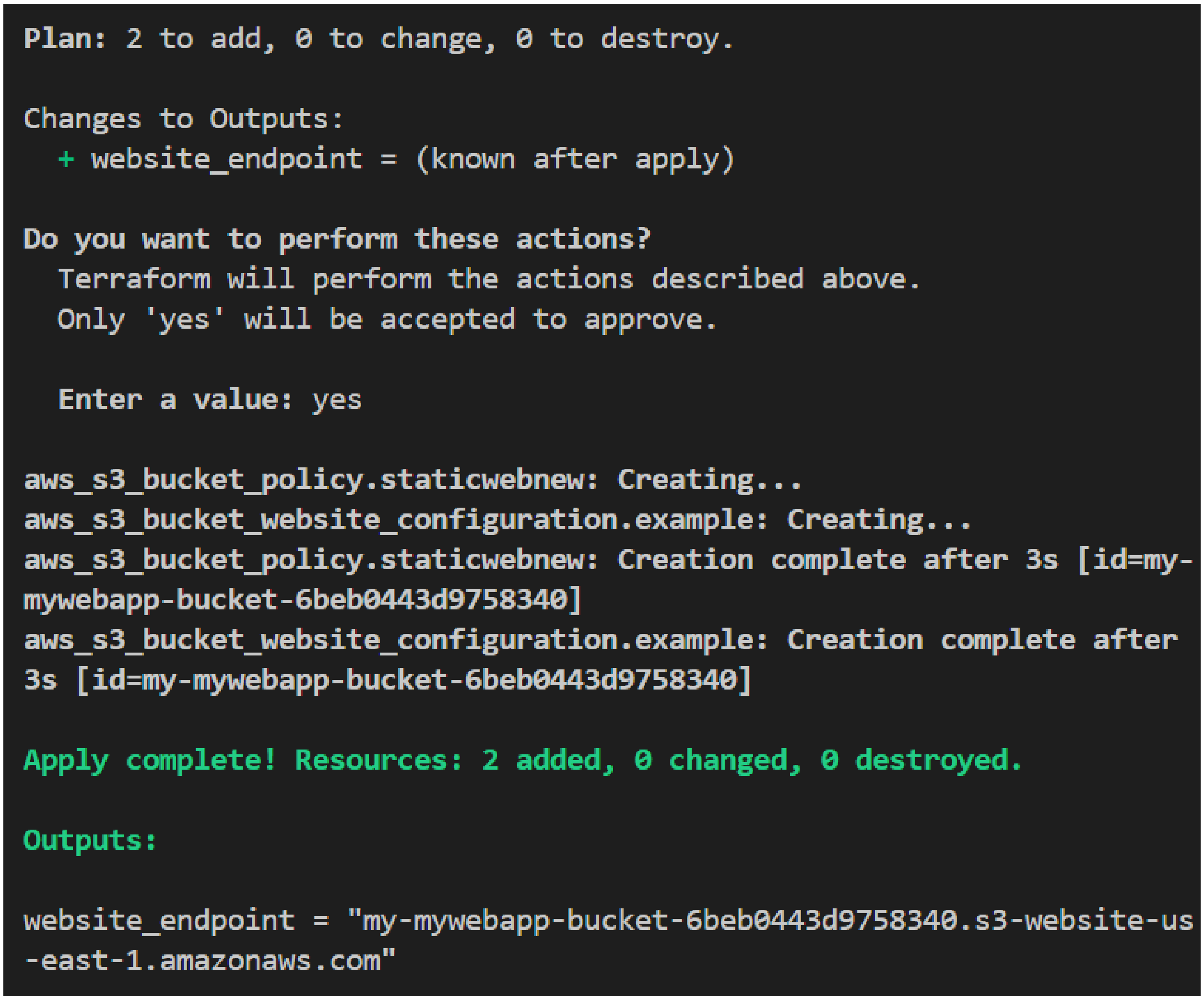


```
- Finding hashicorp/aws versions matching "5.64.0"...
- Finding hashicorp/random versions matching "3.6.2"...
- Installing hashicorp/aws v5.64.0...
- Installed hashicorp/aws v5.64.0 (signed by HashiCorp)
- Installing hashicorp/random v3.6.2...
- Installed hashicorp/random v3.6.2 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the
rovider
selections it made above. Include this file in your version control
epository
so that Terraform can guarantee to make the same selections by defa
t when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform pl
" to see
any changes that are required for your infrastructure. All Terrafor
commands
should now work.

If you ever set or change modules or backend configuration for Terr
orm,
rerun this command to reinitialize your working directory. If you f
get, other
commands will detect it and remind you to do so if necessary.
```

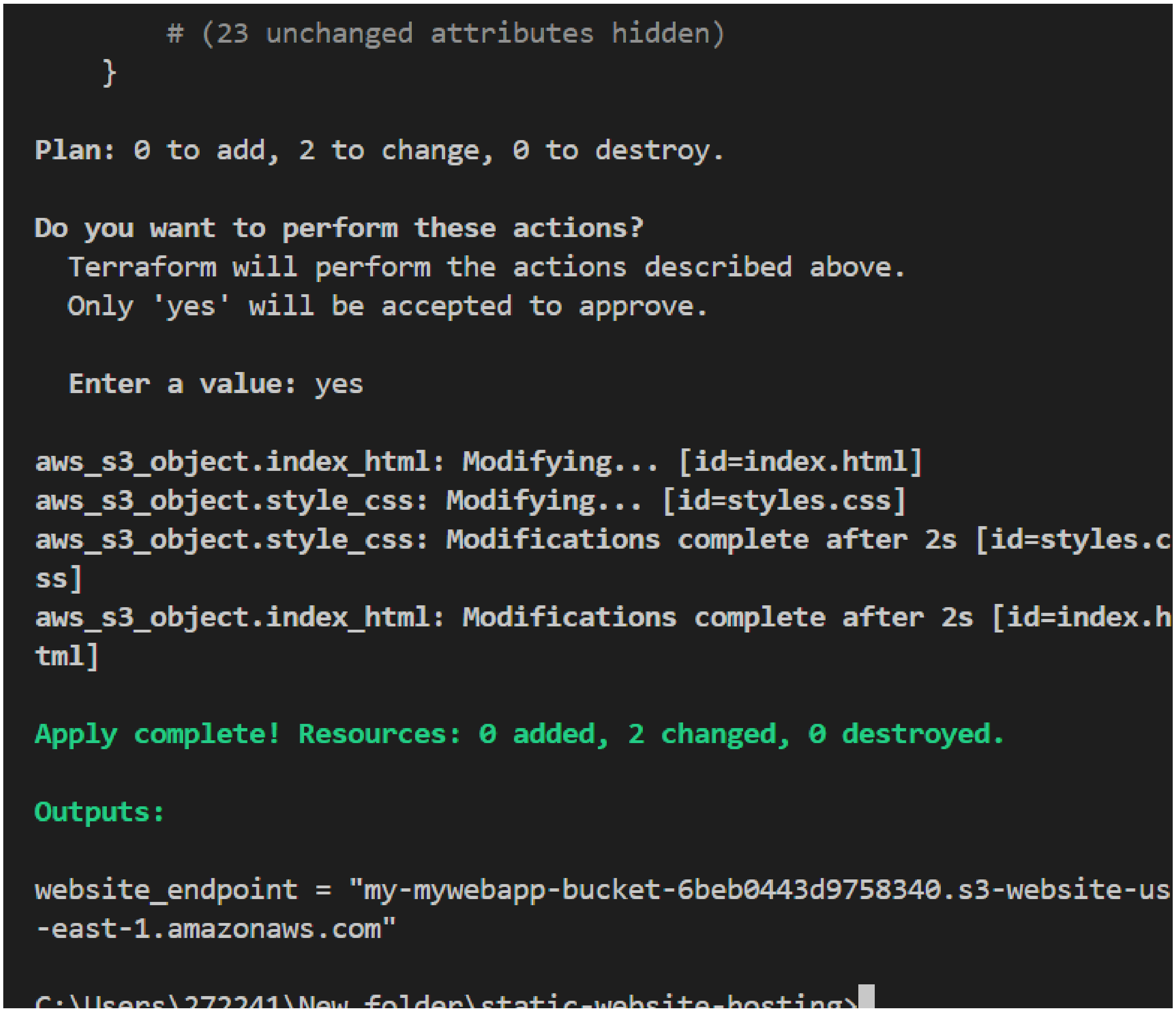
Step 4: Terraform plan and terraform apply:



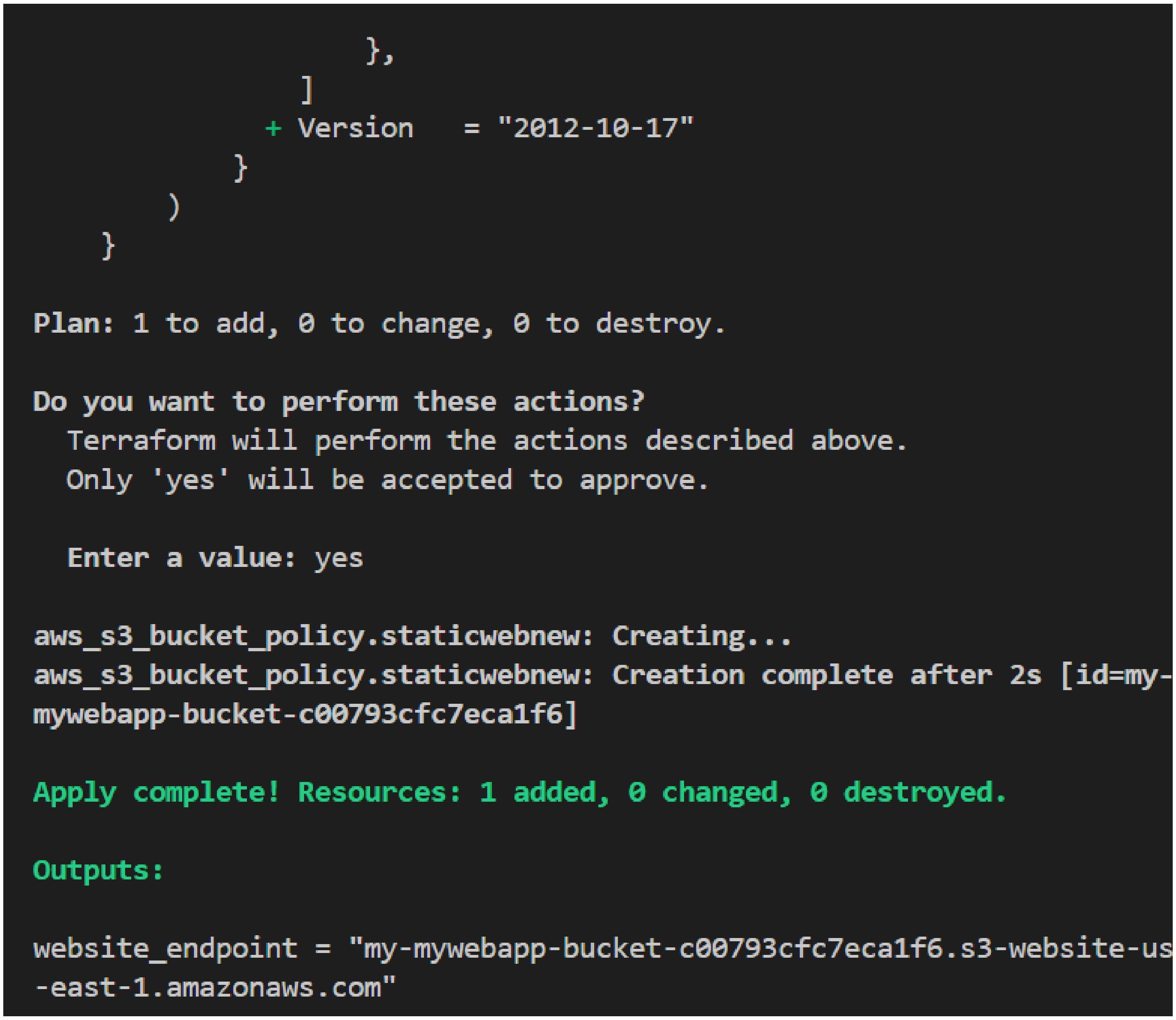
```
      + checksum_sha1          = (known after apply)
      + checksum_sha256        = (known after apply)
      + content_type           = (known after apply)
      + etag                   = (known after apply)
      + force_destroy          = false
      + id                     = (known after apply)
      + key                    = "styles.css"
      + kms_key_id             = (known after apply)
      + server_side_encryption = (known after apply)
      + source                 = "./styles.css"
      + storage_class          = (known after apply)
      + tags_all               = (known after apply)
      + version_id             = (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_object.style_css: Creating...
aws_s3_object.style_css: Creation complete after 3s [id=styles.css]

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



```
Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + website_endpoint = (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_policy.staticwebnew: Creating...
aws_s3_bucket_website_configuration.example: Creating...
aws_s3_bucket_policy.staticwebnew: Creation complete after 3s [id=my-
mywebapp-bucket-6beb0443d9758340]
aws_s3_bucket_website_configuration.example: Creation complete after
3s [id=my-mywebapp-bucket-6beb0443d9758340]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

website_endpoint = "my-mywebapp-bucket-6beb0443d9758340.s3-website-us
-east-1.amazonaws.com"
```

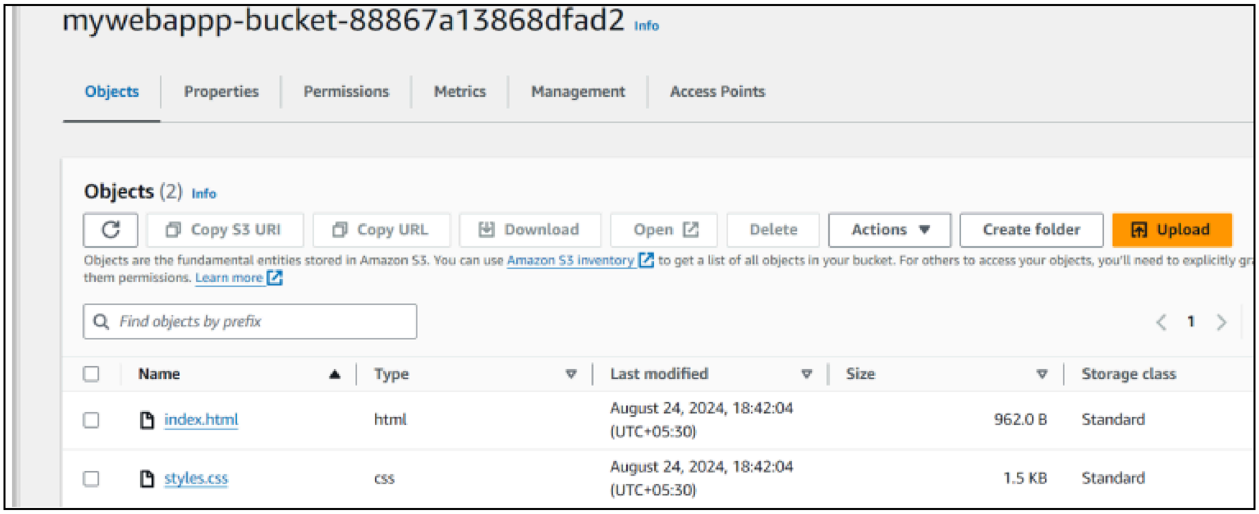


```
        # (23 unchanged attributes hidden)
    }

Plan: 0 to add, 2 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_object.index_html: Modifying... [id=index.html]
aws_s3_object.style_css: Modifying... [id=styles.css]
aws_s3_object.style_css: Modifications complete after 2s [id=styles.c
ss]
aws_s3_object.index_html: Modifications complete after 2s [id=index.h
tml]

Apply complete! Resources: 0 added, 2 changed, 0 destroyed.

Outputs:

website_endpoint = "my-mywebapp-bucket-6beb0443d9758340.s3-website-us
-east-1.amazonaws.com"

C:\Users\272241\New folder\static-website-hosting>
```



```
                    },
                ]
              + Version   = "2012-10-17"
            }
        )
    }

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_policy.staticwebnew: Creating...
aws_s3_bucket_policy.staticwebnew: Creation complete after 2s [id=my-
mywebapp-bucket-c00793cfc7eca1f6]

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

Outputs:

website_endpoint = "my-mywebapp-bucket-c00793cfc7eca1f6.s3-website-us
-east-1.amazonaws.com"
```

Step 4 : check bucket for if files are uploaded and if the site is hosted correctly at the website_endpoint given in cmd Outputs

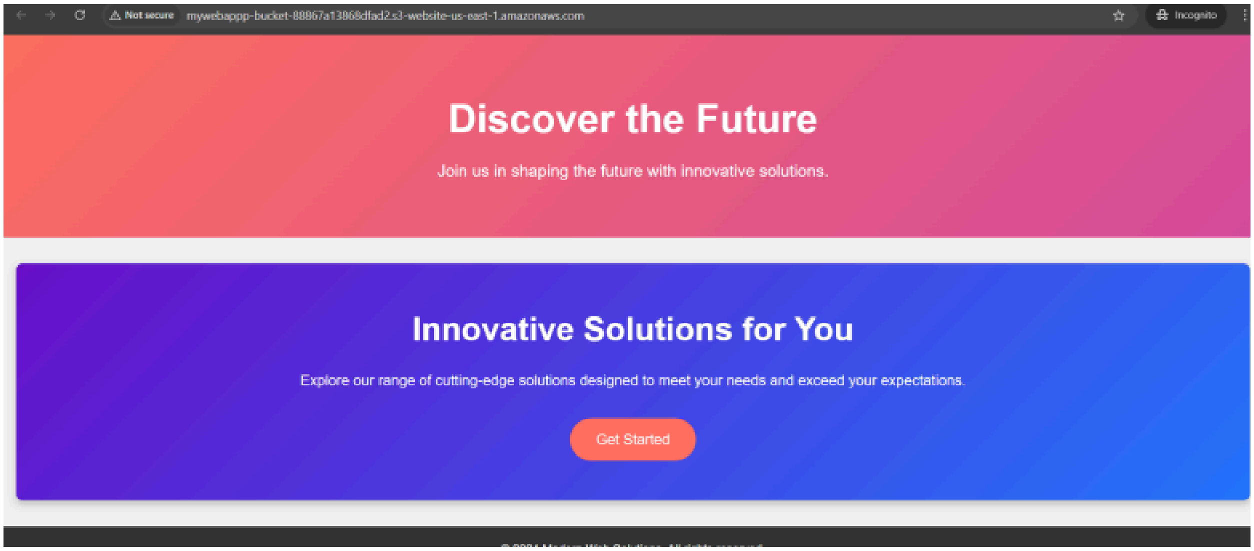


mywebappp-bucket-88867a13868dfad2 Info

Objects (2) Info

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 gr them permissions. Learn more

☐	Name	Type	Last modified	Size	Storage class
☐	index.html	html	August 24, 2024, 18:42:04 (UTC+05:30)	962.0 B	Standard
☐	styles.css	css	August 24, 2024, 18:42:04 (UTC+05:30)	1.5 KB	Standard



Not secure
mywebappp-bucket-88867a13868dfad2.s3-website-us-east-1.amazonaws.com
Incognito
Get Started

Discover the Future

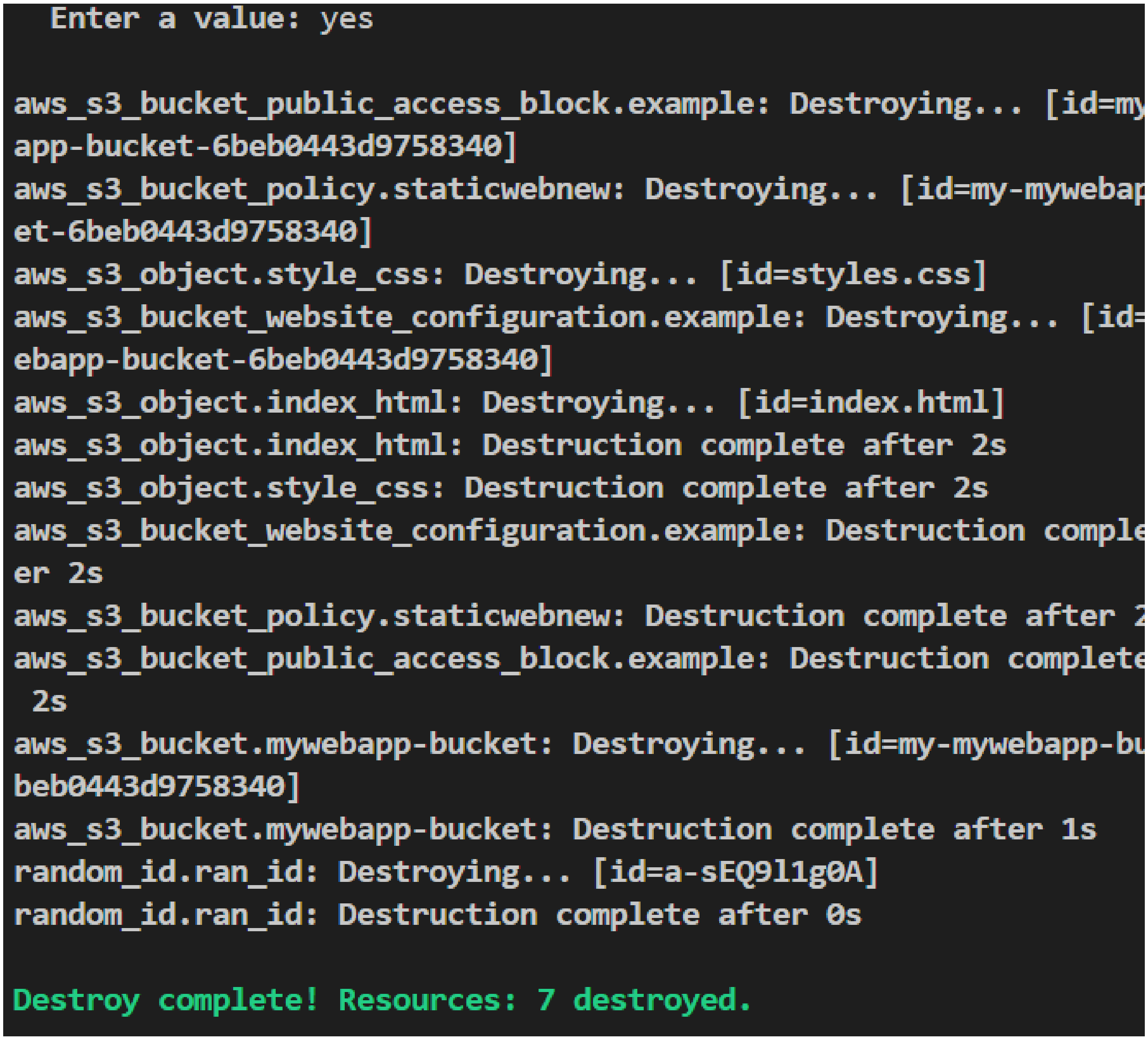
Join us in shaping the future with innovative solutions.

Innovative Solutions for You

Explore our range of cutting-edge solutions designed to meet your needs and exceed your expectations.

© 2024 Modern Web Solutions. All rights reserved.

Step 5 : terraform destroy to destroy the bucket



```
  Enter a value: yes

aws_s3_bucket_public_access_block.example: Destroying... [id=my
app-bucket-6beb0443d9758340]
aws_s3_bucket_policy.staticwebnew: Destroying... [id=my-mywebap
et-6beb0443d9758340]
aws_s3_object.style_css: Destroying... [id=styles.css]
aws_s3_bucket_website_configuration.example: Destroying... [id=
ebapp-bucket-6beb0443d9758340]
aws_s3_object.index_html: Destroying... [id=index.html]
aws_s3_object.index_html: Destruction complete after 2s
aws_s3_object.style_css: Destruction complete after 2s
aws_s3_bucket_website_configuration.example: Destruction comple
er 2s
aws_s3_bucket_policy.staticwebnew: Destruction complete after 2
aws_s3_bucket_public_access_block.example: Destruction complete
 2s
aws_s3_bucket.mywebapp-bucket: Destroying... [id=my-mywebapp-bu
beb0443d9758340]
aws_s3_bucket.mywebapp-bucket: Destruction complete after 1s
random_id.ran_id: Destroying... [id=a-sEQ9l1g0A]
random_id.ran_id: Destruction complete after 0s

Destroy complete! Resources: 7 destroyed.
```