# EXPERIMENT NO. 6

**NAME : PRANAV TITAMBE          CLASS : D15A          ROLL NO. : 62**

**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.

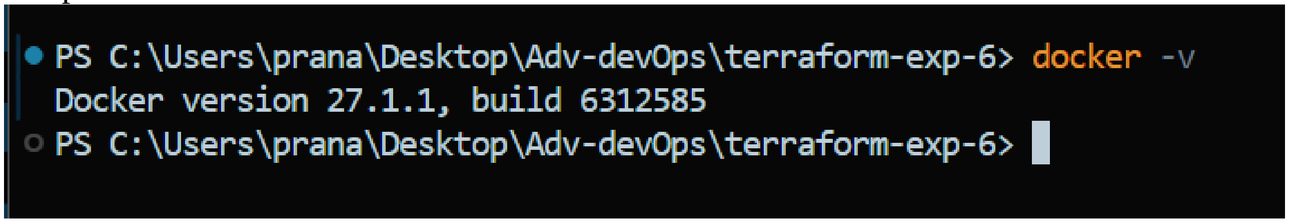
## Benefits of Using Terraform for AWS Infrastructure

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 docker installation and version

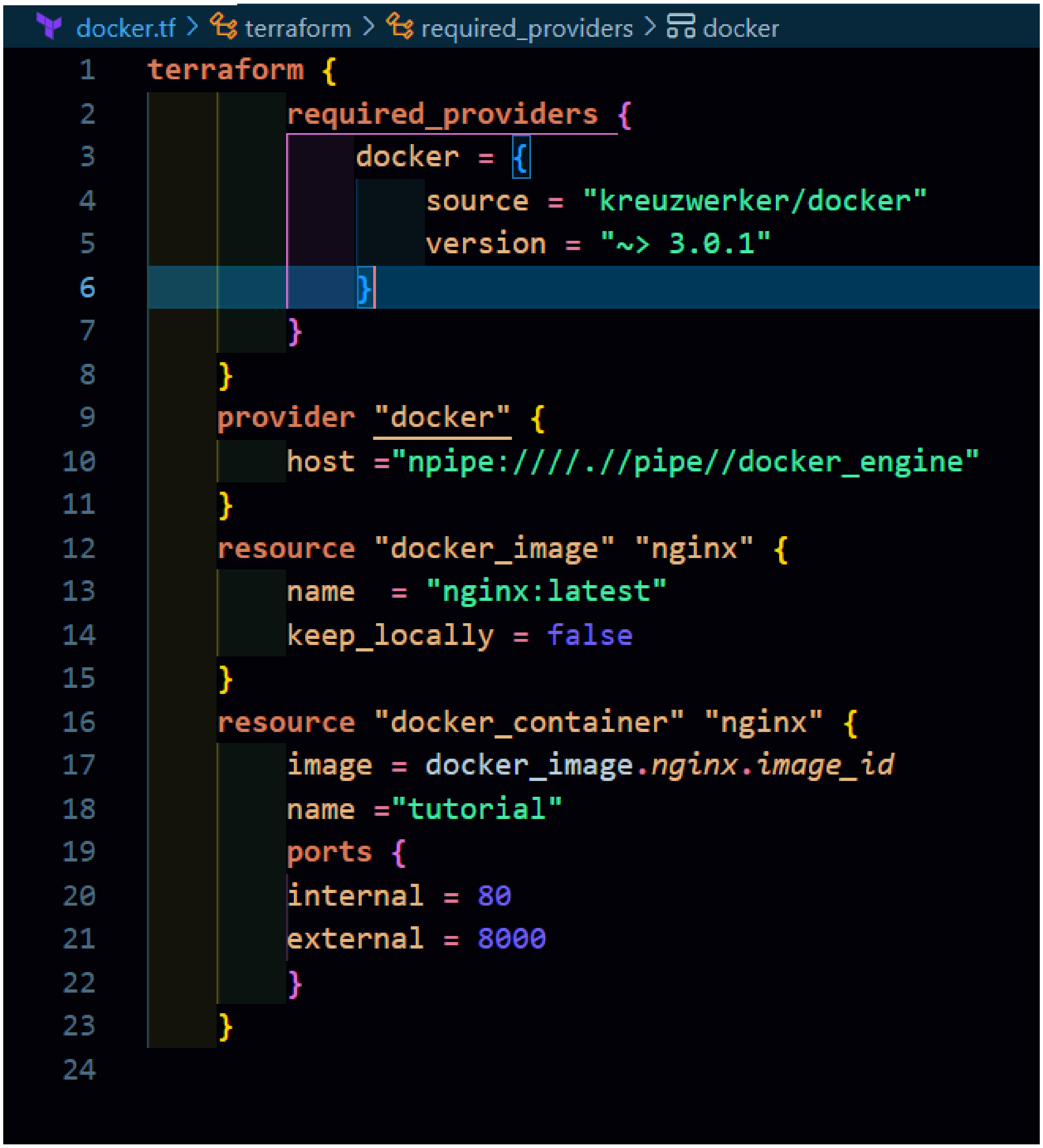


```
PS C:\Users\prana\Desktop\Adv-devOps\terraform-exp-6> docker -v
Docker version 27.1.1, build 6312585
PS C:\Users\prana\Desktop\Adv-devOps\terraform-exp-6>
```

Step 2 : create docker.tf file and write following code for terraform and docker

**Code -**

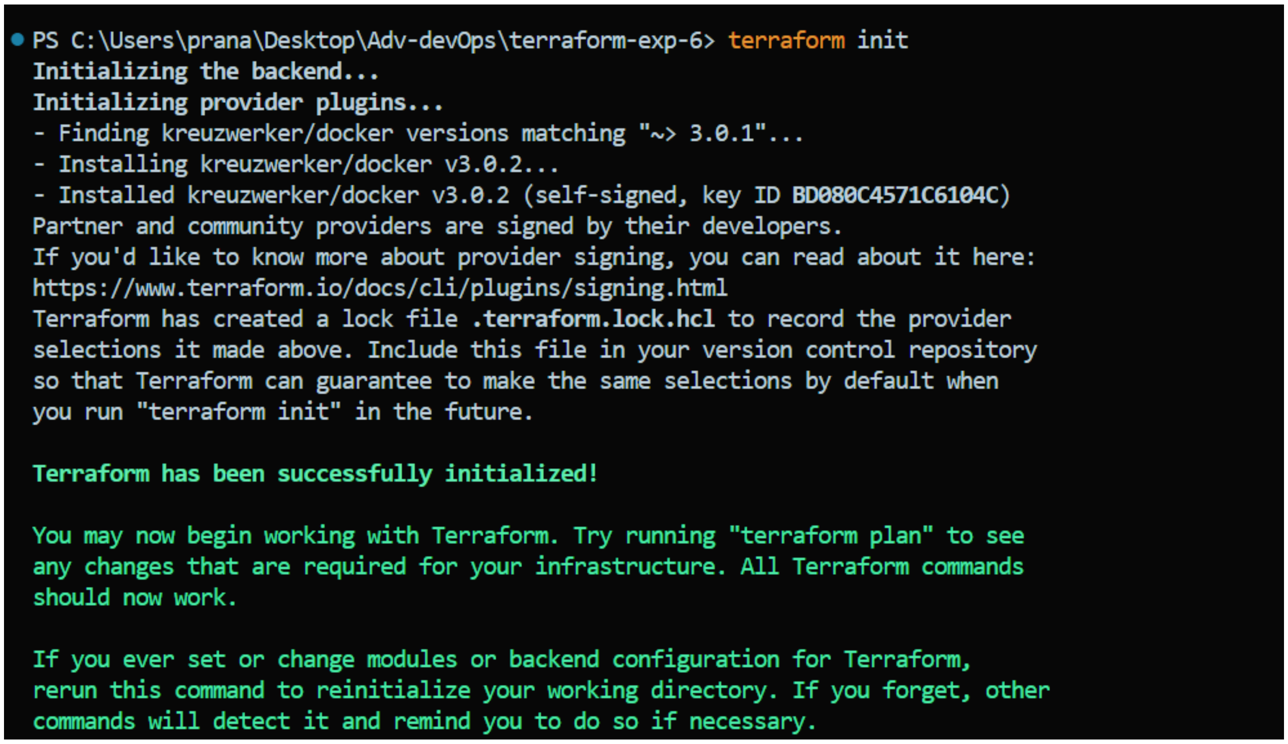
```
terraform {
  required_providers {
  docker = {
      source = "kreuzwerker/docker"
      version = "~> 3.0.1"
   }
  }
}

provider "docker" {
  host    = "npipe:////.//pipe//docker_engine"
}

resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}
resource "docker_container" "nginx" {
  image = docker_image.nginx.image_id
  name  = "tutorial"
  ports {
    internal = 80
    external = 8000
  }
}
```



docker.tf > terraform > required_providers > docker

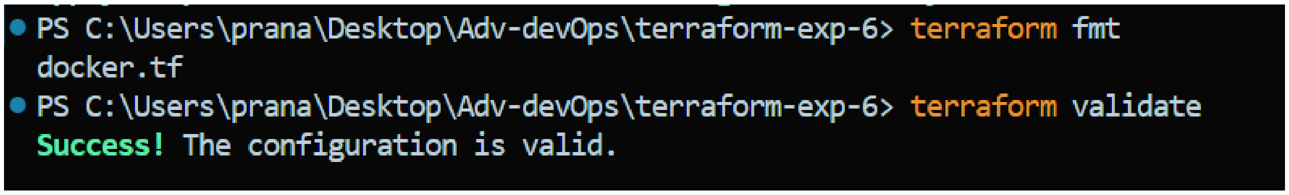
```
terraform {
        required_providers {
            docker = {
                source = "kreuzwerker/docker"
                version = "~> 3.0.1"
            }
        }
    }
    provider "docker" {
        host ="npipe:////.//pipe//docker_engine"
    }
    resource "docker_image" "nginx" {
        name   = "nginx:latest"
        keep_locally = false
    }
    resource "docker_container" "nginx" {
        image = docker_image.nginx.image_id
        name ="tutorial"
        ports {
        internal = 80
        external = 8000
        }
    }
```

Step 3 : Type terraform init command to initialize terraform backend



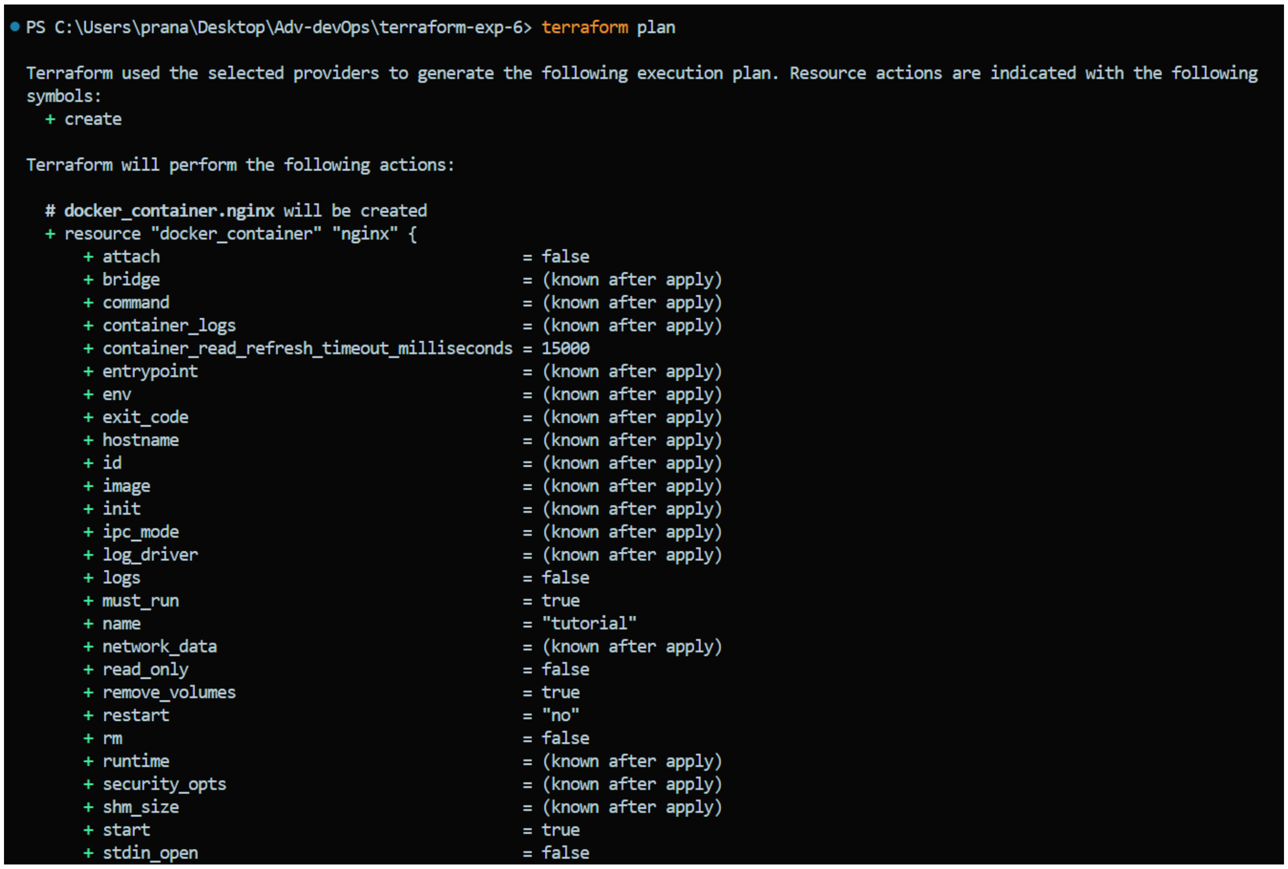
```
● PS C:\Users\prana\Desktop\Adv-devOps\terraform-exp-6> terraform init
  Initializing the backend...
  Initializing provider plugins...
  - Finding kreuzwerker/docker versions matching "~> 3.0.1"...
  - Installing kreuzwerker/docker v3.0.2...
  - Installed kreuzwerker/docker v3.0.2 (self-signed, key ID BD080C4571C6104C)
  Partner and community providers are signed by their developers.
  If you'd like to know more about provider signing, you can read about it here:
  https://www.terraform.io/docs/cli/plugins/signing.html
  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.
```

Step 4(EXTRA) : type terraform fmt and validate commands .
The two Terraform commands – terraform validate and terraform fmt – are used to maintain a clean, error-free, and well-structured Terraform codebase.

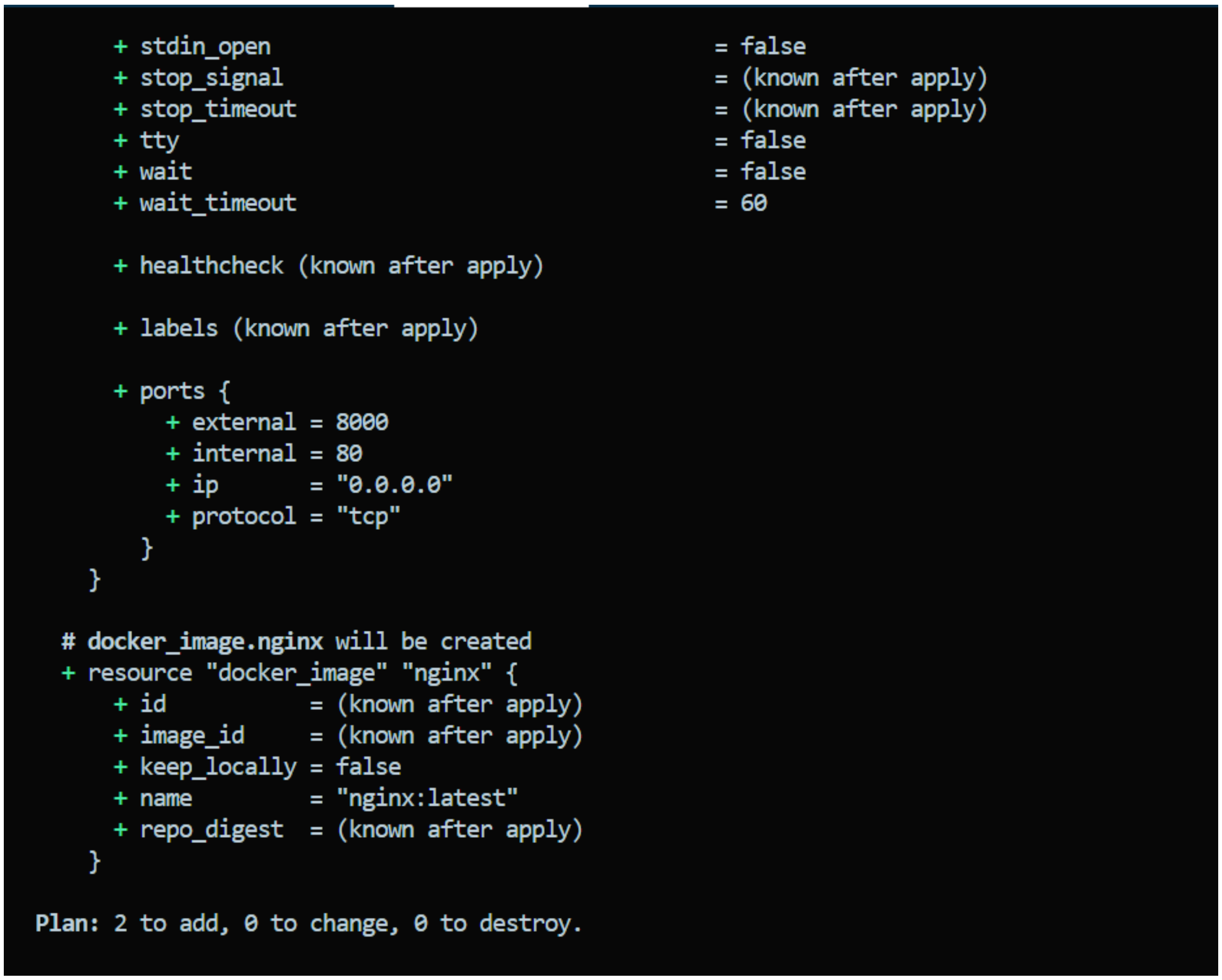


```
● PS C:\Users\prana\Desktop\Adv-devOps\terraform-exp-6> terraform fmt
  docker.tf
● PS C:\Users\prana\Desktop\Adv-devOps\terraform-exp-6> terraform validate
  Success! The configuration is valid.
```

Step 5 : Type Terraform plan command to create execution plan .

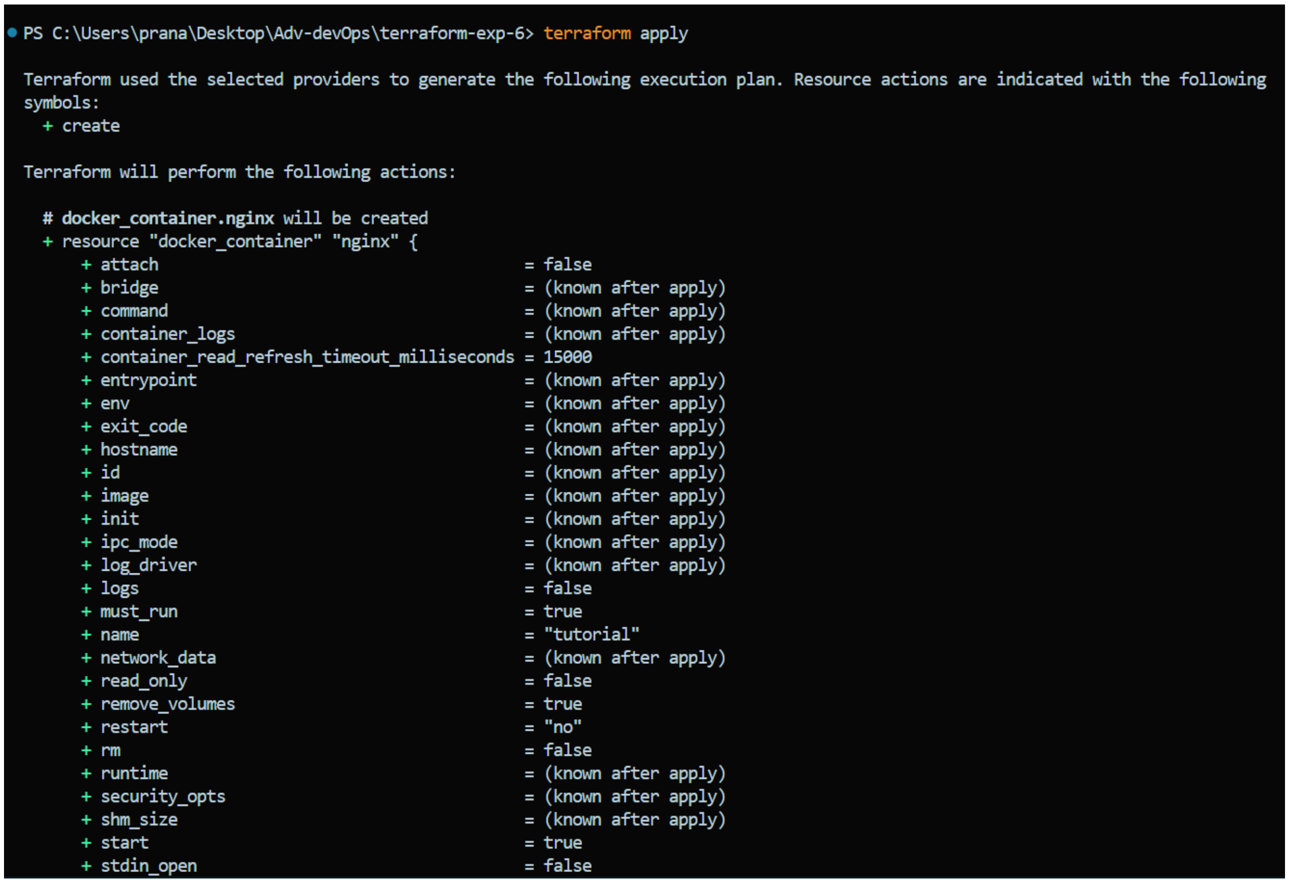


```
PS C:\Users\prana\Desktop\Adv-devOps\terraform-exp-6> terraform plan

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:

  # 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
```

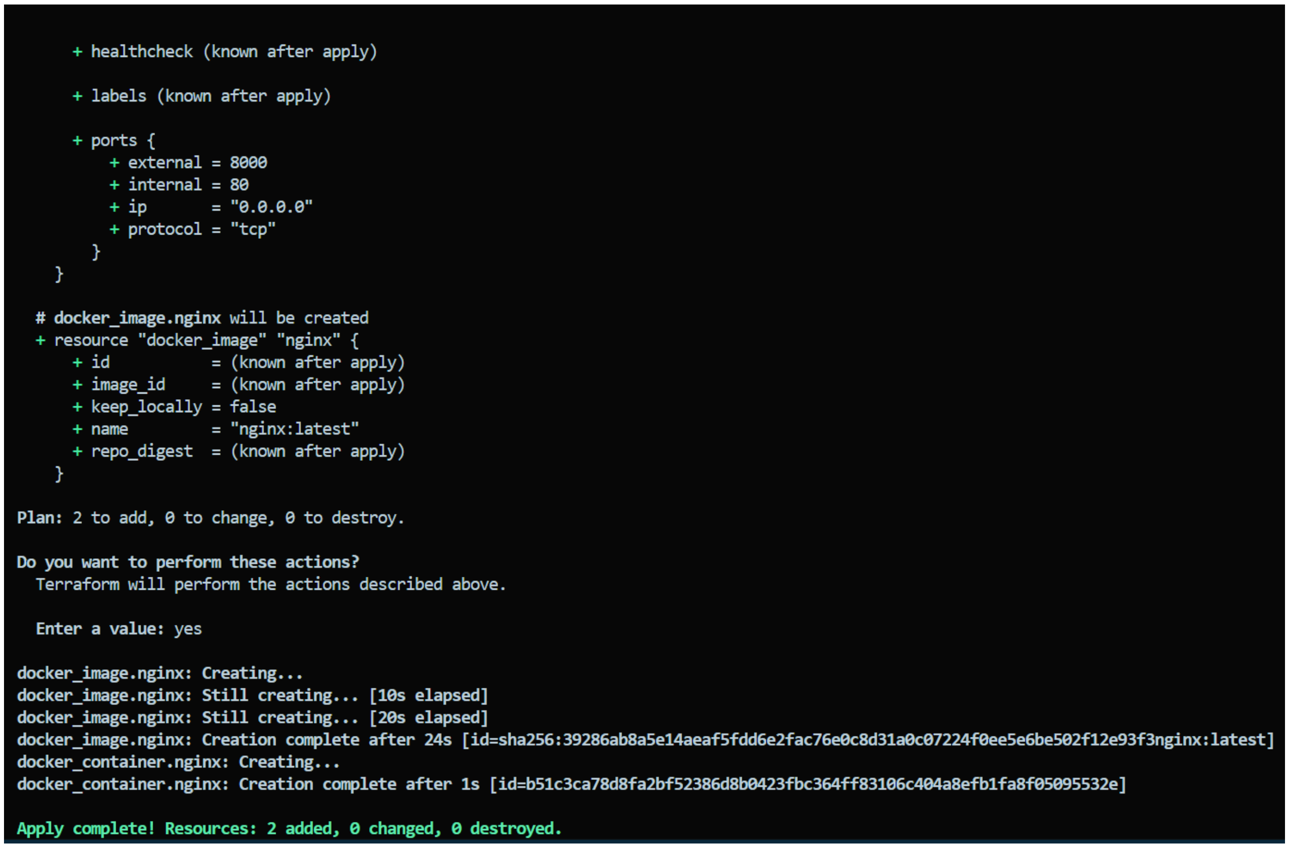


```
      + stdin_open                                  = false
      + stop_signal                                 = (known after apply)
      + stop_timeout                                = (known after apply)
      + tty                                         = false
      + wait                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

      + healthcheck (known after apply)

      + labels (known after apply)

      + 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.
```

Step 6 : Type terraform apply to apply changes .



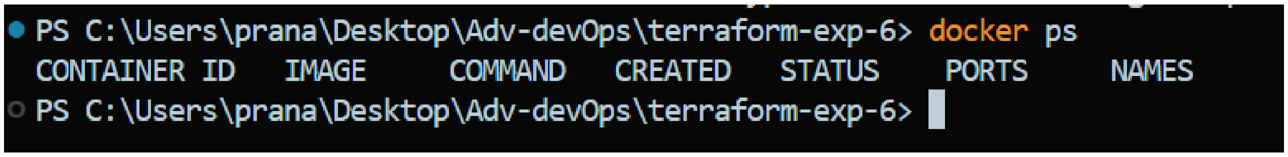
```
PS C:\Users\prana\Desktop\Adv-devOps\terraform-exp-6> terraform apply

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:

  # 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
```



```
      + 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.

  Enter a value: yes

docker_image.nginx: Creating...
docker_image.nginx: Still creating... [10s elapsed]
docker_image.nginx: Still creating... [20s elapsed]
docker_image.nginx: Creation complete after 24s [id=sha256:39286ab8a5e14aeaf5fdd6e2fac76e0c8d31a0c07224f0ee5e6be502f12e93f3nginx:latest]
docker_container.nginx: Creating...
docker_container.nginx: Creation complete after 1s [id=b51c3ca78d8fa2bf52386d8b0423fbc364ff83106c404a8efb1fa8f05095532e]

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

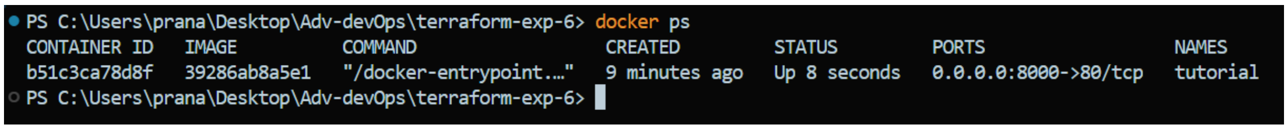
Step 7 : Docker container before and after step 6 execution

BEFORE –



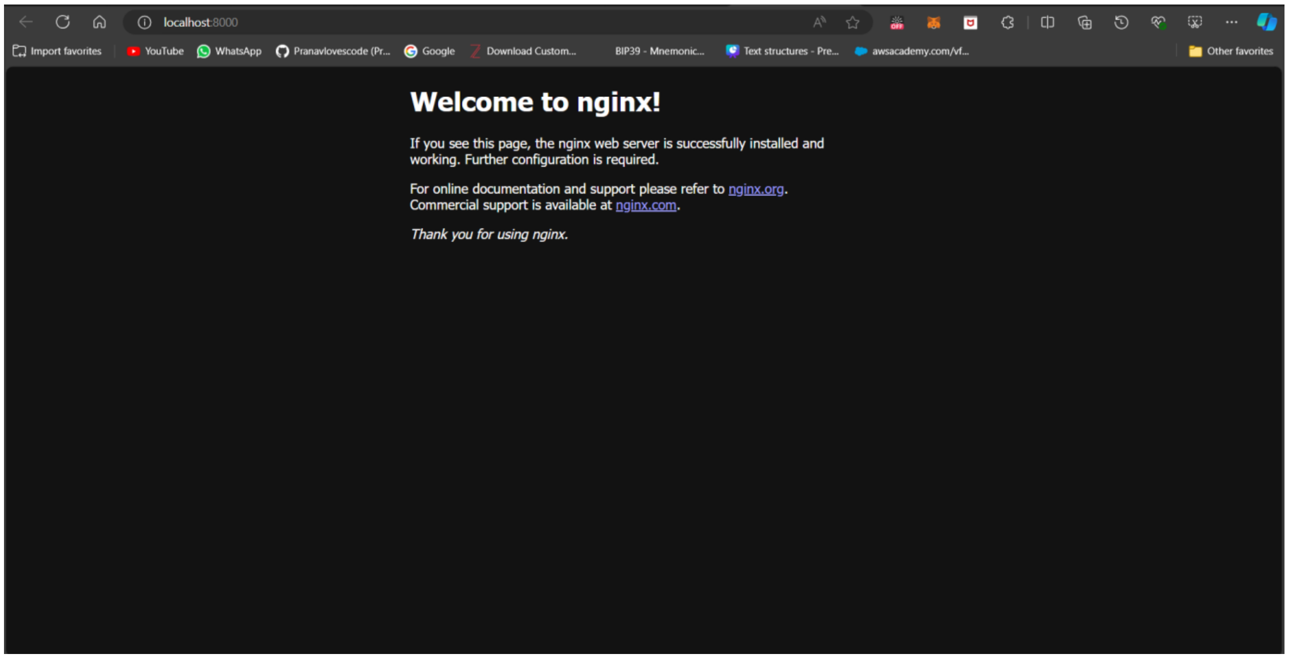
```
PS C:\Users\prana\Desktop\Adv-devOps\terraform-exp-6> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Users\prana\Desktop\Adv-devOps\terraform-exp-6>
```

AFTER -



```
PS C:\Users\prana\Desktop\Adv-devOps\terraform-exp-6> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                  NAMES
b51c3ca78d8f   39286ab8a5e1   "/docker-entrypoint.…"   9 minutes ago   Up 8 seconds   0.0.0.0:8000->80/tcp   tutorial
PS C:\Users\prana\Desktop\Adv-devOps\terraform-exp-6>
```

OUTPUT-



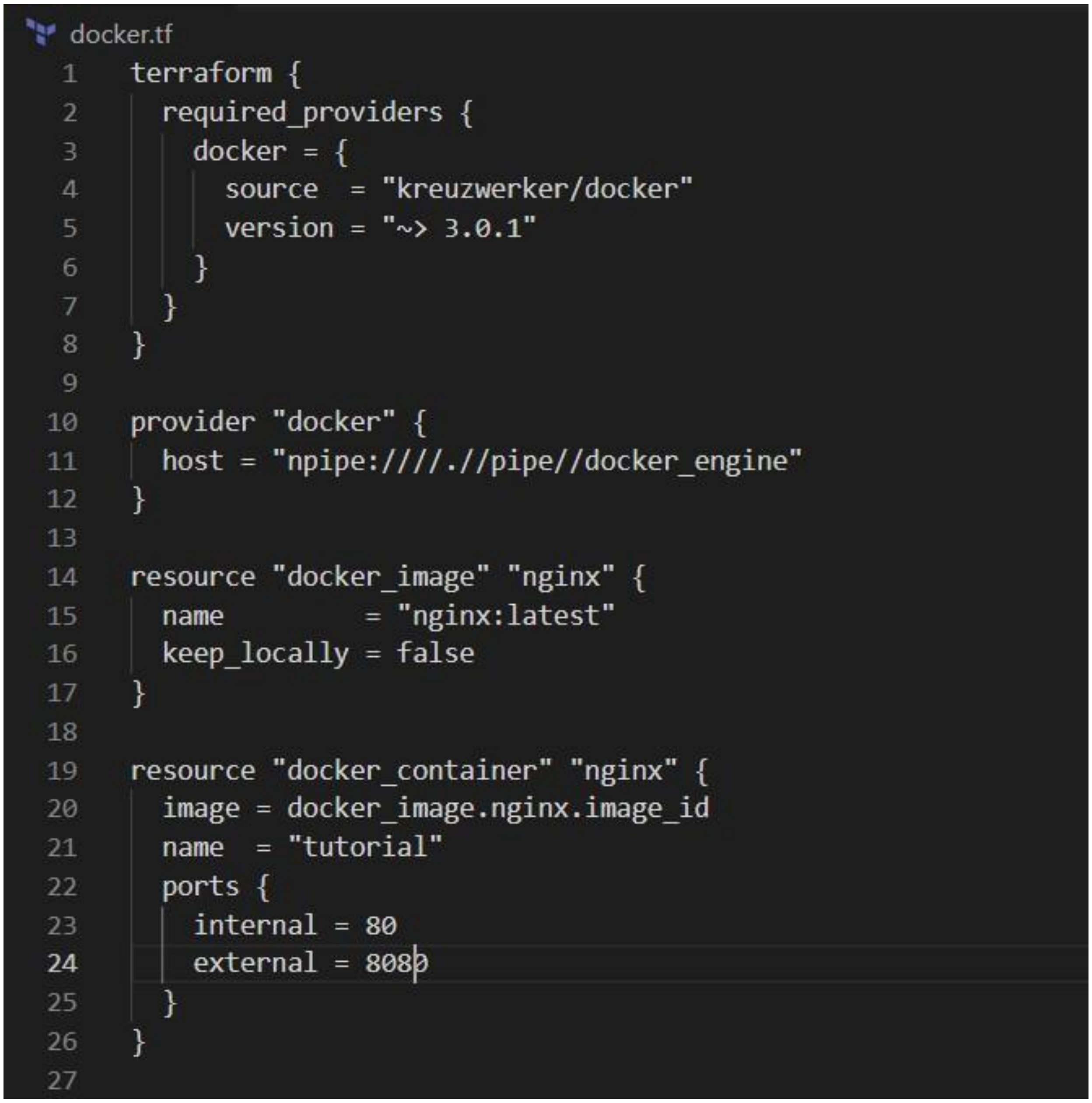
**Welcome to nginx!**

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

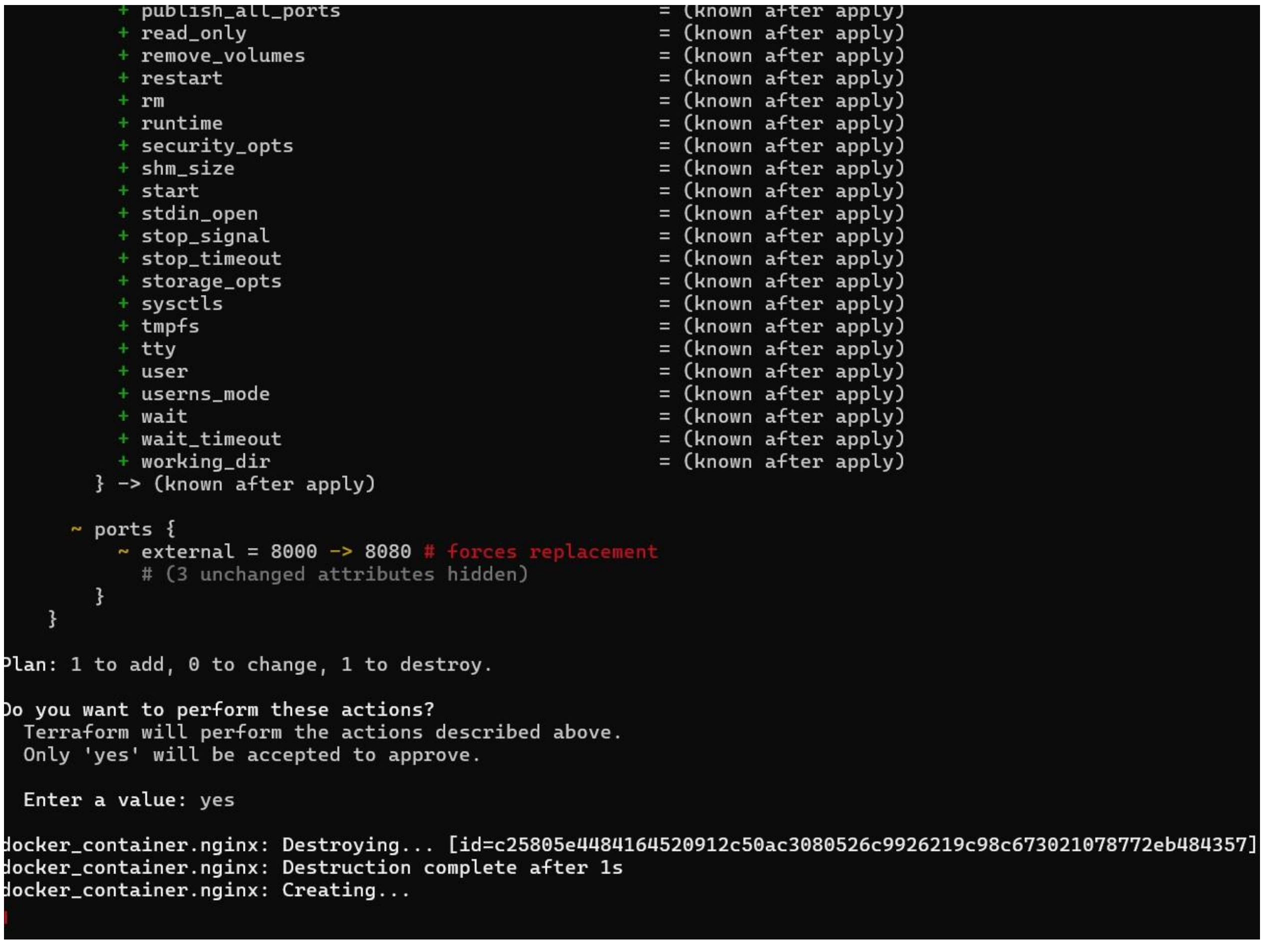
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

*Thank you for using nginx.*

Step 8 (EXTRA ) :  Execution of change .

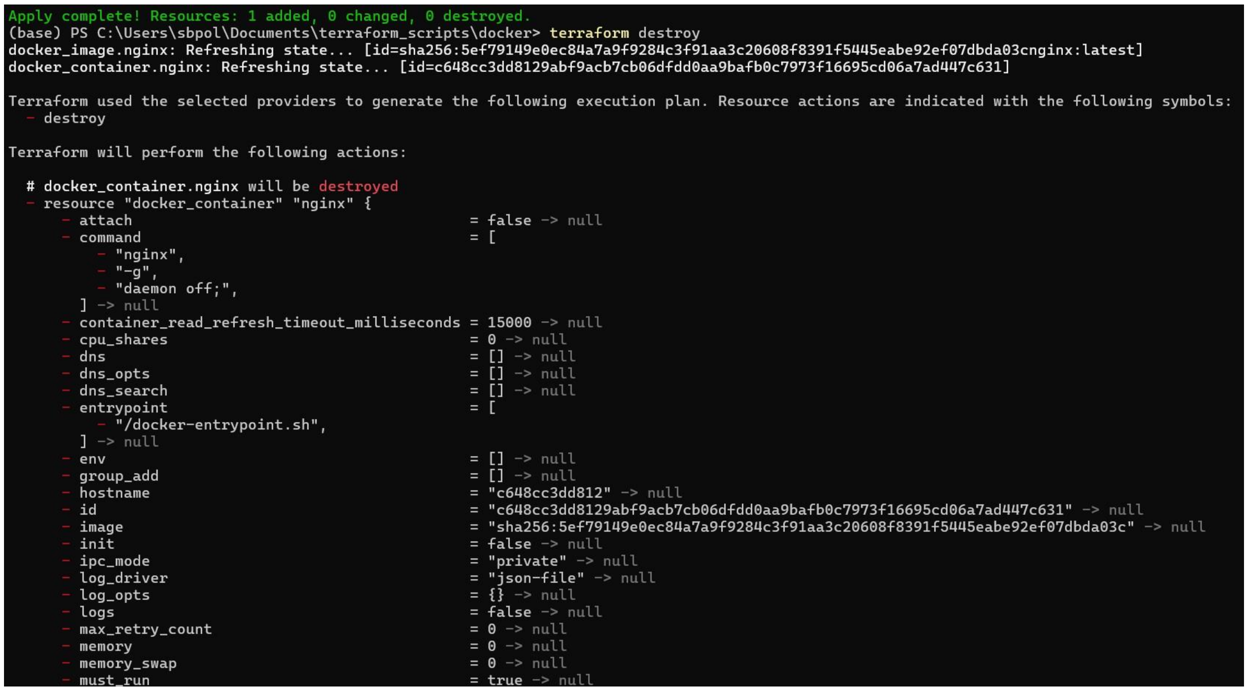

docker.tf

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0.1"
    }
  }
}

provider "docker" {
  host = "npipe:////.//pipe//docker_engine"
}

resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}

resource "docker_container" "nginx" {
  image = docker_image.nginx.image_id
  name  = "tutorial"
  ports {
    internal = 80
    external = 8080
  }
}

```

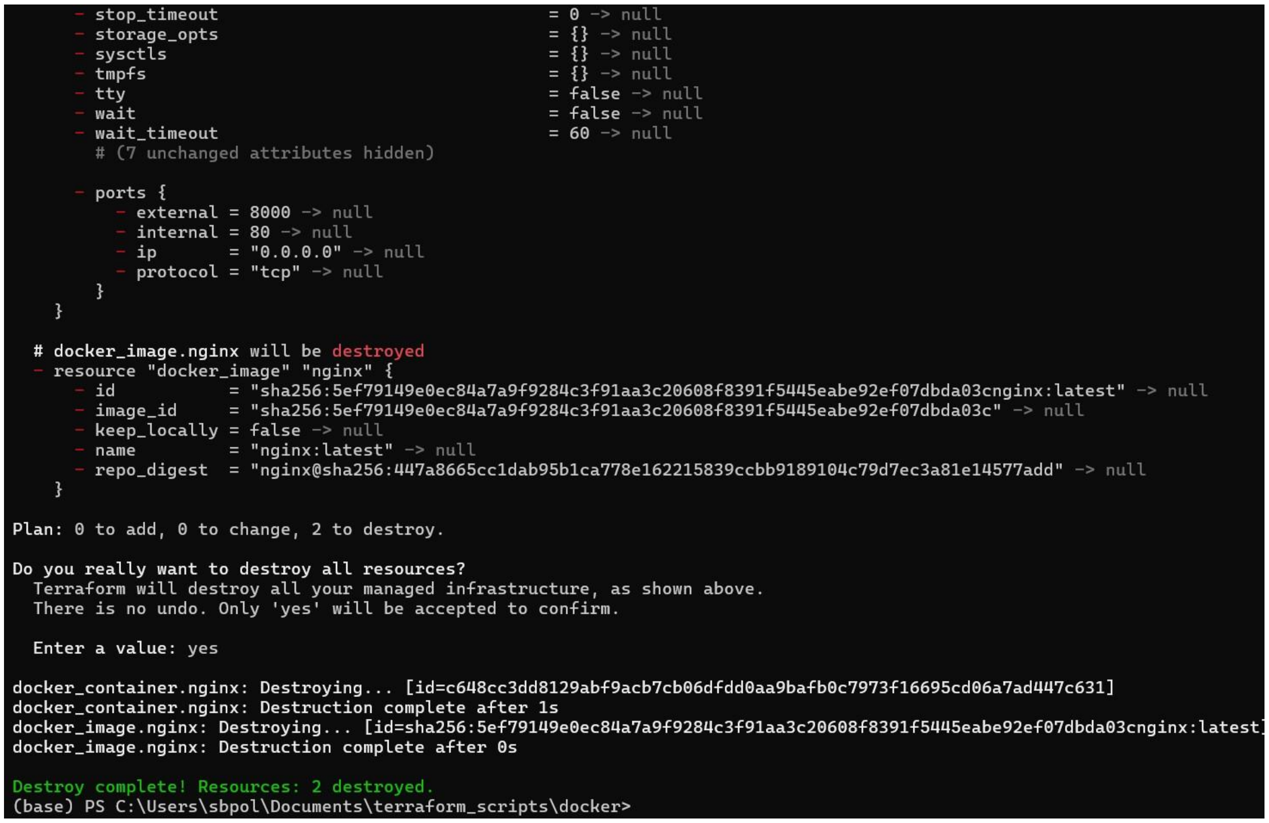


```
          + publish_all_ports                              = (known after apply)
          + read_only                                      = (known after apply)
          + remove_volumes                                 = (known after apply)
          + restart                                        = (known after apply)
          + rm                                             = (known after apply)
          + runtime                                        = (known after apply)
          + security_opts                                  = (known after apply)
          + shm_size                                       = (known after apply)
          + start                                          = (known after apply)
          + stdin_open                                     = (known after apply)
          + stop_signal                                    = (known after apply)
          + stop_timeout                                   = (known after apply)
          + storage_opts                                   = (known after apply)
          + sysctls                                        = (known after apply)
          + tmpfs                                          = (known after apply)
          + tty                                            = (known after apply)
          + user                                           = (known after apply)
          + userns_mode                                    = (known after apply)
          + wait                                           = (known after apply)
          + wait_timeout                                   = (known after apply)
          + working_dir                                    = (known after apply)
        } -> (known after apply)

      ~ ports {
          ~ external = 8000 -> 8080 # forces replacement
            # (3 unchanged attributes hidden)
        }
    }

Plan: 1 to add, 0 to change, 1 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_container.nginx: Destroying... [id=c25805e4484164520912c50ac3080526c9926219c98c673021078772eb484357]
docker_container.nginx: Destruction complete after 1s
docker_container.nginx: Creating...
```

Step 9 : terraform destroy to destroy infrastructure.

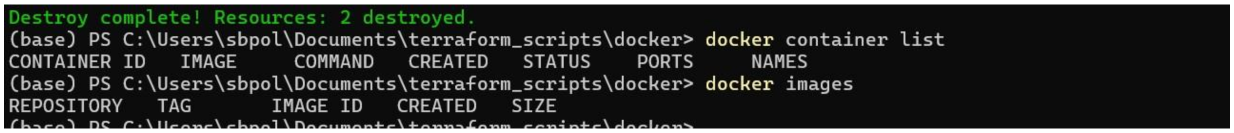


```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> terraform destroy
docker_image.nginx: Refreshing state... [id=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
docker_container.nginx: Refreshing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

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:

  # 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" -> null
      - image                                       = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03c" -> null
      - 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
      - must_run                                    = true -> 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" -> null
      - 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=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:latest]
docker_image.nginx: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker>
```

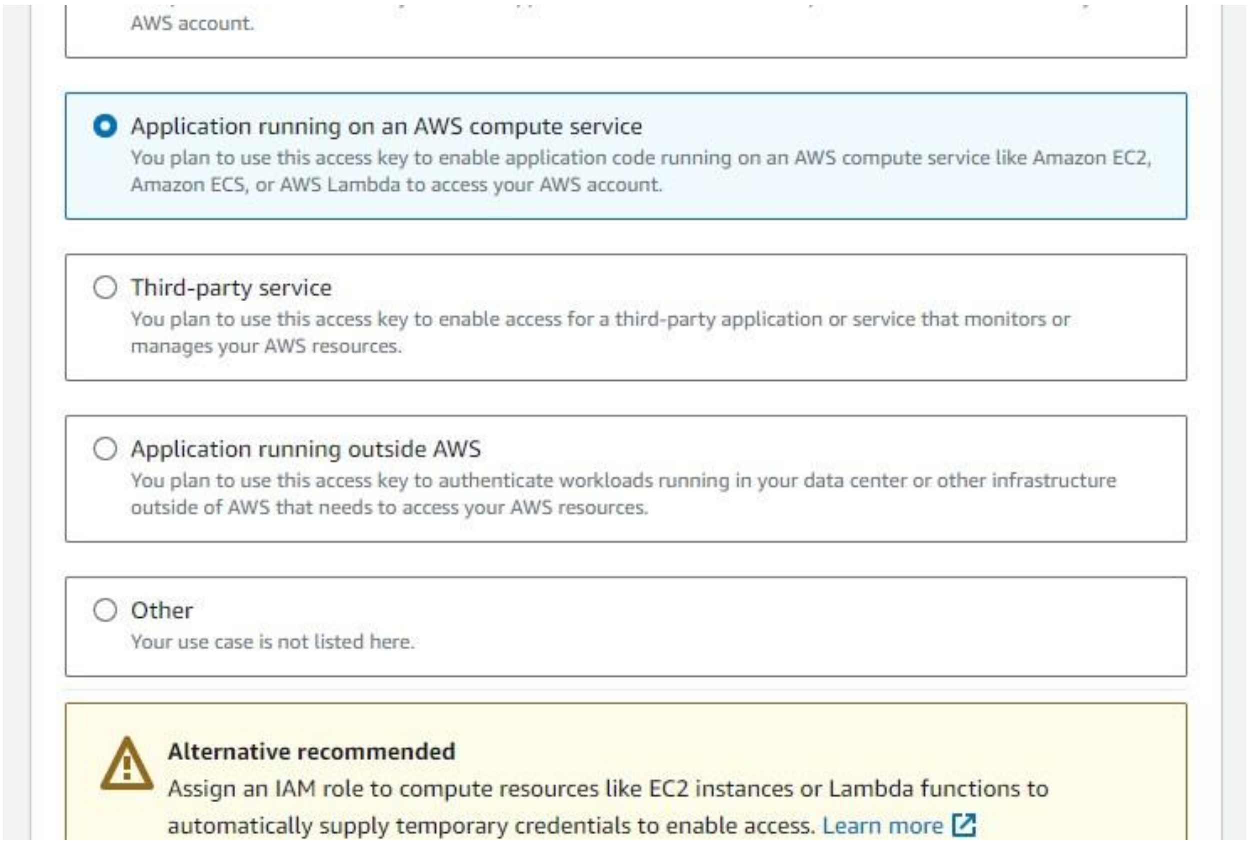
Step 10 : Docker after destroy command.



```
Destroy complete! Resources: 2 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> docker container list
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker>
```

**Terraform and S3 -**

Step 1: Create access keys and secret key for IAM user


AWS account.

[✓ Application running on an AWS compute service]
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

Application running outside AWS
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

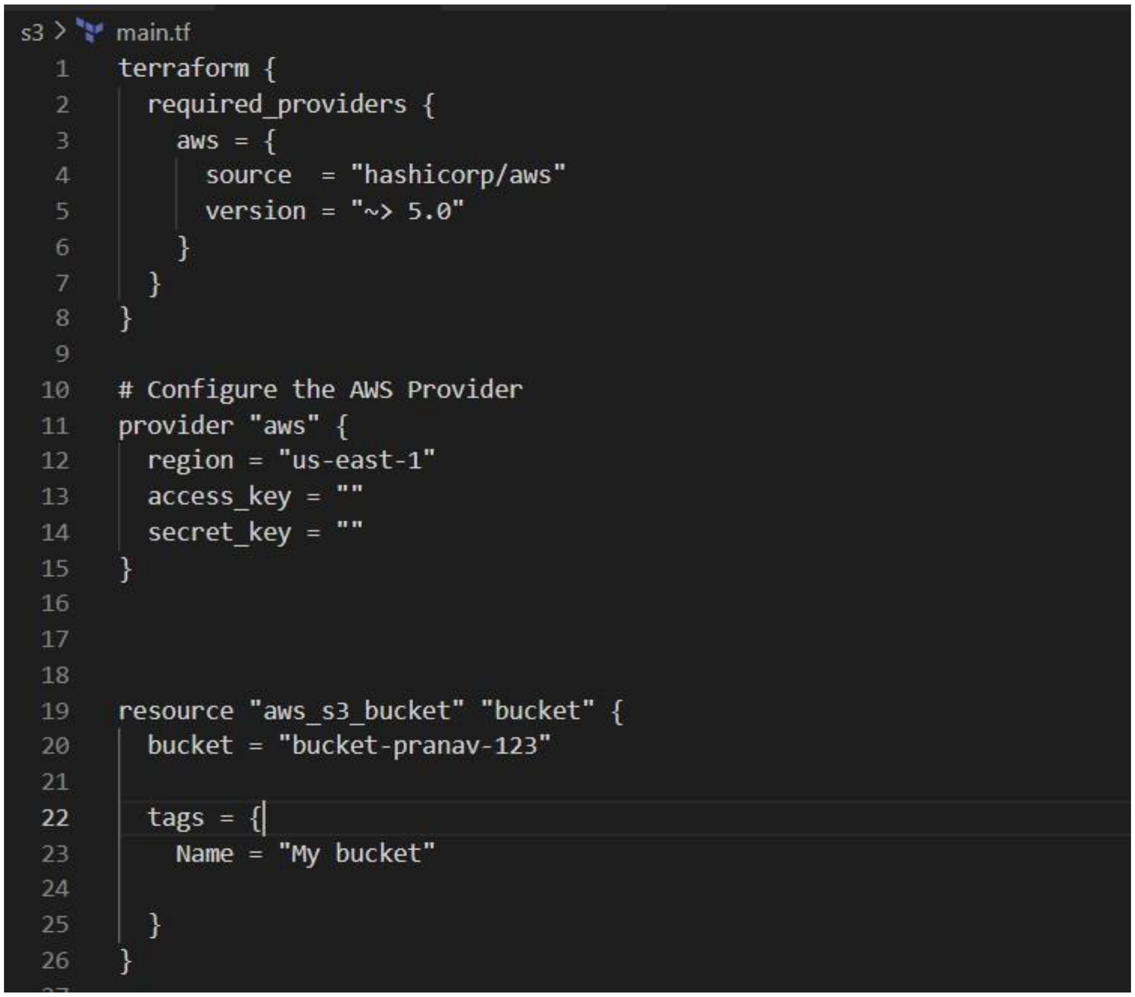
Other
Your use case is not listed here.

**Alternative recommended**
Assign an IAM role to compute resources like EC2 instances or Lambda functions to automatically supply temporary credentials to enable access. Learn more

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

**Code -**
```
terraform {
 required_providers {
 aws = {
     source = "hashicorp/aws"
     version = "~> 5.0"
   }
 }
}
```

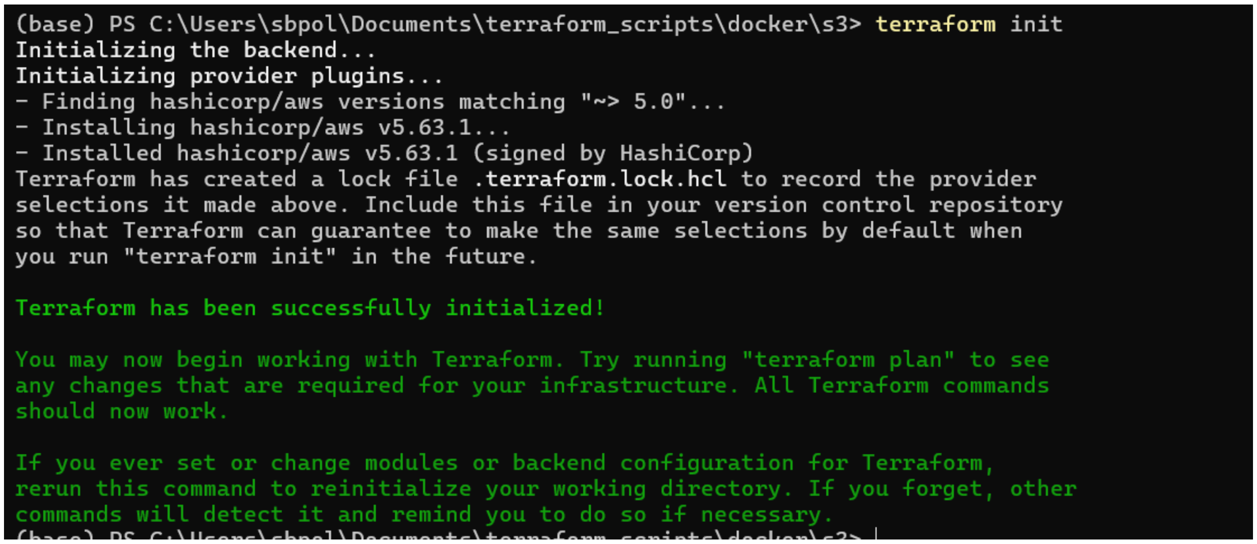
```
# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  access_key = ""
  secret_key = ""
}
resource "aws_s3_bucket" "bucket" {
  bucket = "bucket-pranav-123"

  tags = {
    Name = "My bucket"

  }
}
```



s3 > main.tf

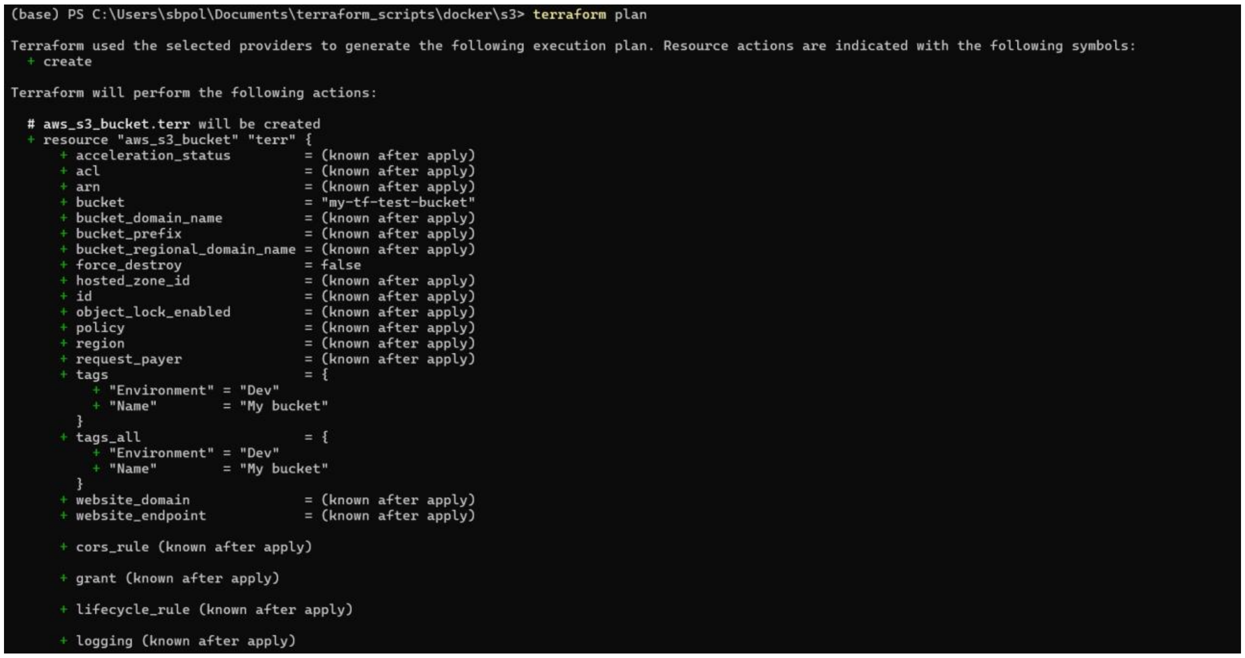
```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  access_key = ""
  secret_key = ""
}



resource "aws_s3_bucket" "bucket" {
  bucket = "bucket-pranav-123"

  tags = {
    Name = "My bucket"

  }
}
```

Step 3 : Type terraform init command in powershell.



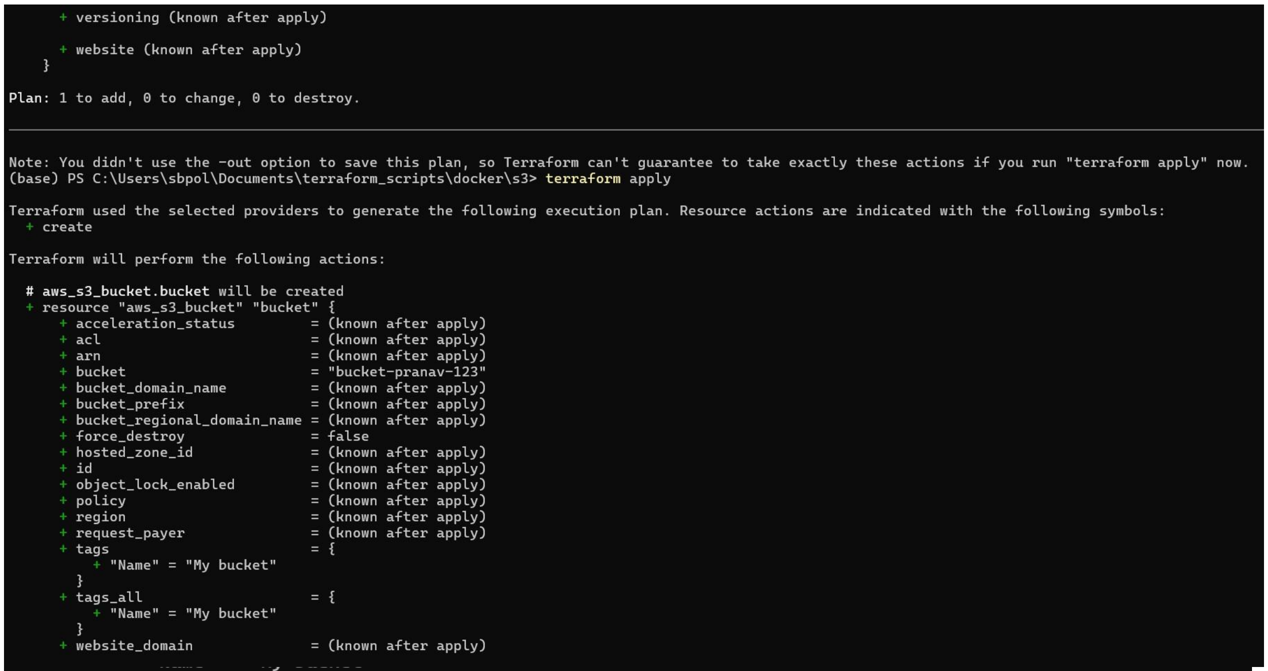
```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.63.1...
- Installed hashicorp/aws v5.63.1 (signed by HashiCorp)
Terraform 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.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3>
```

Step 4 : Type terraform plan command in powershell.

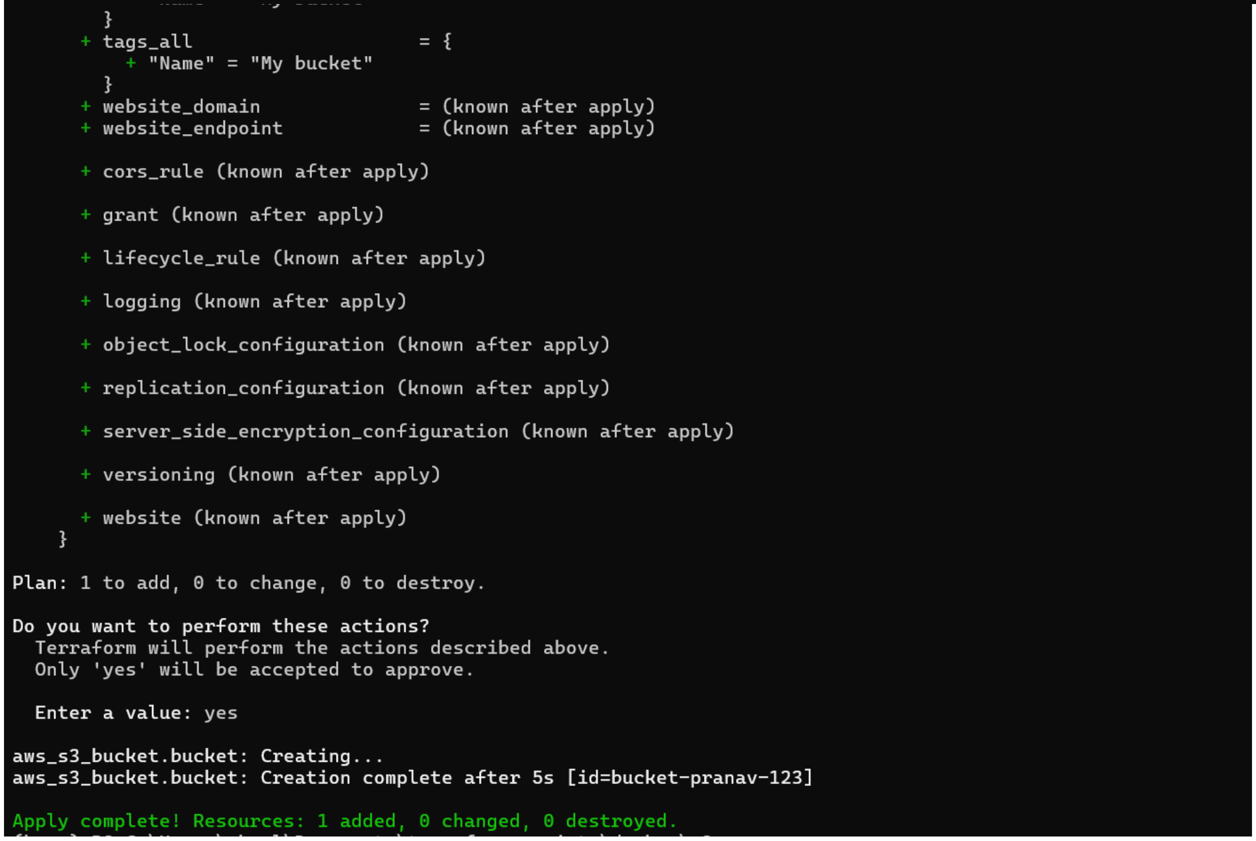


```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform plan

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

      + logging (known after apply)
```

Step 5 : Type terraform apply command in powershell.

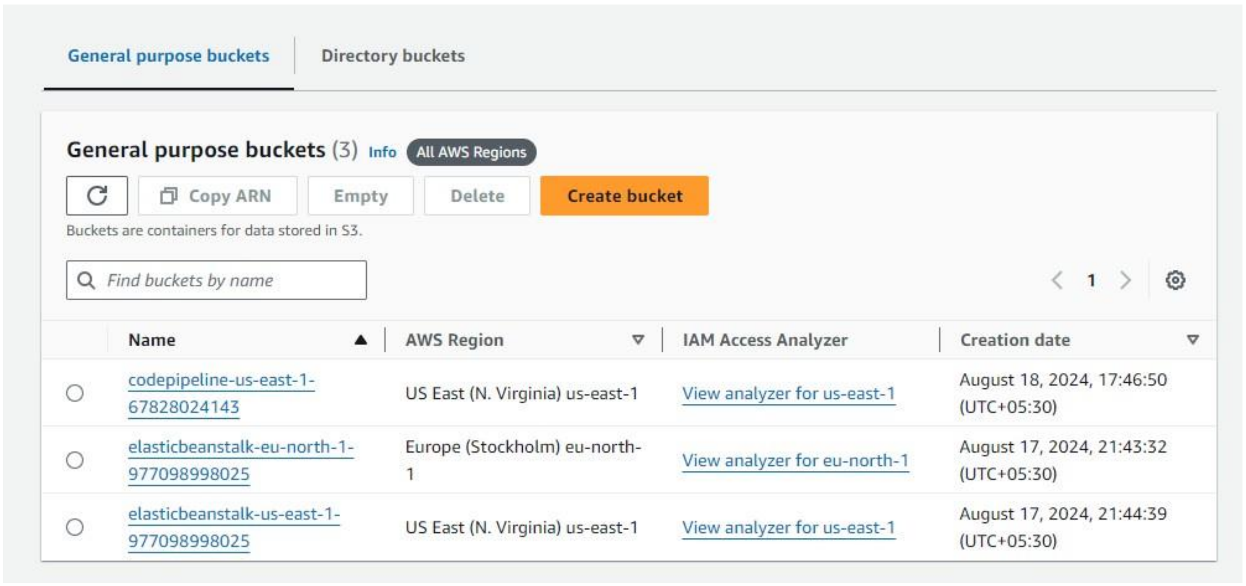


```
      + versioning (known after apply)

      + 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.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\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.bucket will be created
  + resource "aws_s3_bucket" "bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "bucket-pranav-123"
      + 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                        = {
          + "Name" = "My bucket"
        }
      + tags_all                    = {
          + "Name" = "My bucket"
        }
      + website_domain              = (known after apply)
```



```
        }
      + tags_all                    = {
          + "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.bucket: Creating...
aws_s3_bucket.bucket: Creation complete after 5s [id=bucket-pranav-123]

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

Step 6 : AWS s3 before and after the bucket creation using terraform.
BEFORE -

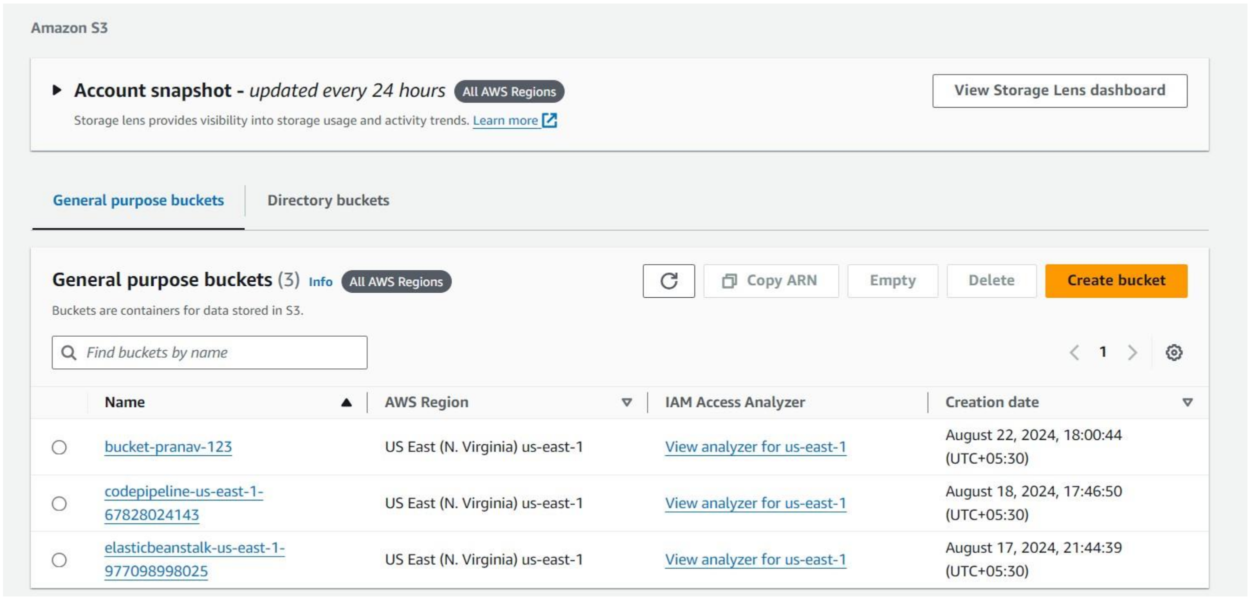


General purpose buckets (3)

Buckets are containers for data stored in S3.

| Name | AWS Region | IAM Access Analyzer | Creation date |
|---|---|---|---|
| codepipeline-us-east-1-67828024143 | US East (N. Virginia) us-east-1 | View analyzer for us-east-1 | August 18, 2024, 17:46:50 (UTC+05:30) |
| elasticbeanstalk-eu-north-1-977098998025 | Europe (Stockholm) eu-north-1 | View analyzer for eu-north-1 | August 17, 2024, 21:43:32 (UTC+05:30) |
| elasticbeanstalk-us-east-1-977098998025 | US East (N. Virginia) us-east-1 | View analyzer for us-east-1 | August 17, 2024, 21:44:39 (UTC+05:30) |

AFTER -



Account snapshot - *updated every 24 hours*

Storage lens provides visibility into storage usage and activity trends.

General purpose buckets (3)

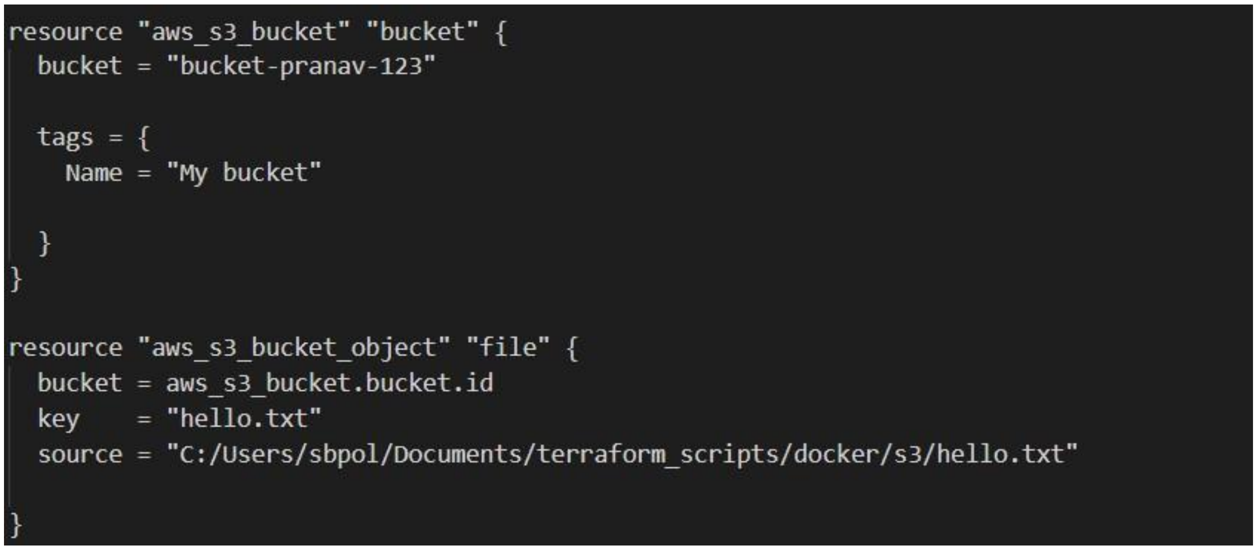
Buckets are containers for data stored in S3.

| Name | AWS Region | IAM Access Analyzer | Creation date |
|---|---|---|---|
| bucket-pranav-123 | US East (N. Virginia) us-east-1 | View analyzer for us-east-1 | August 22, 2024, 18:00:44 (UTC+05:30) |
| codepipeline-us-east-1-67828024143 | US East (N. Virginia) us-east-1 | View analyzer for us-east-1 | August 18, 2024, 17:46:50 (UTC+05:30) |
| elasticbeanstalk-us-east-1-977098998025 | US East (N. Virginia) us-east-1 | View analyzer for us-east-1 | August 17, 2024, 21:44:39 (UTC+05:30) |

Step 7(EXTRA) : Upload file to the bucket using terraform .

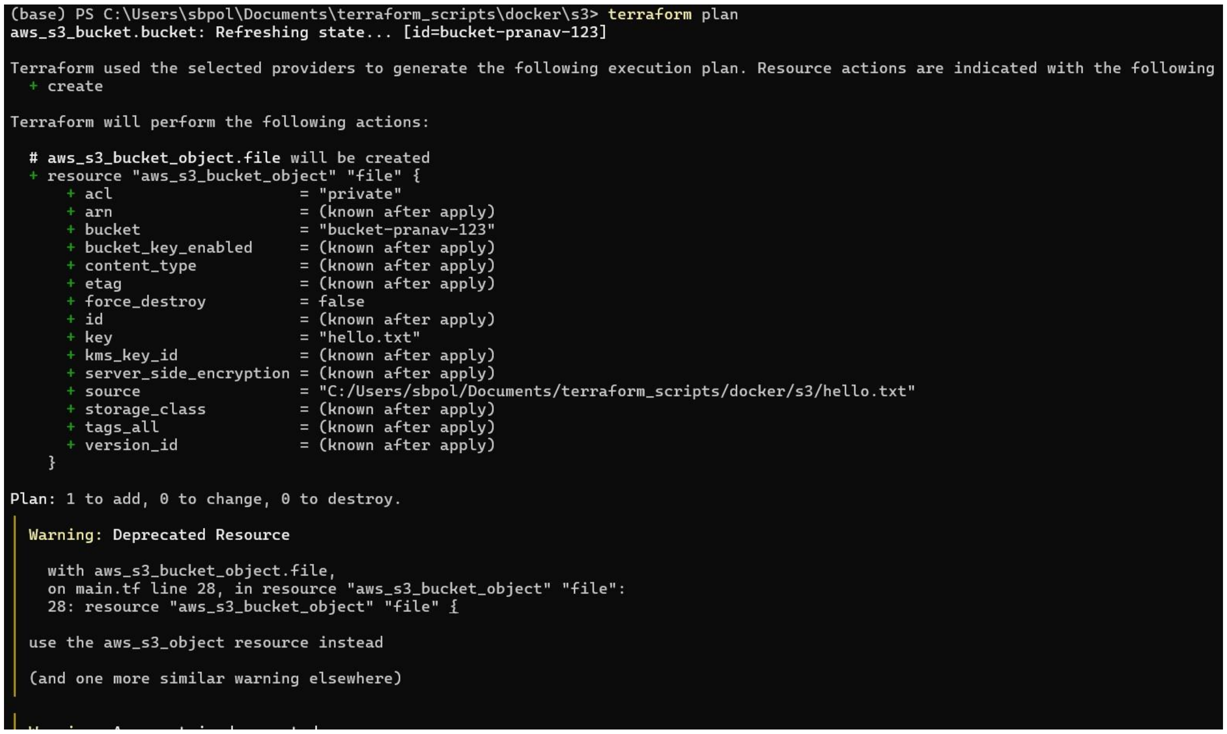
CODE -

```
terraform {
  required_providers {
  aws = {
      source = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  access_key = ""
  secret_key = ""
}

resource "aws_s3_bucket" "bucket" {
  bucket = "bucket-pranav-123"

  tags = {
    Name = "My bucket"

  }
}

resource "aws_s3_bucket_object" "file" {
  bucket = aws_s3_bucket.bucket.id
  key    = "hello.txt"
  source = "C:/Users/sbpol/Documents/terraform_scripts/docker/s3/hello.txt"

}
```

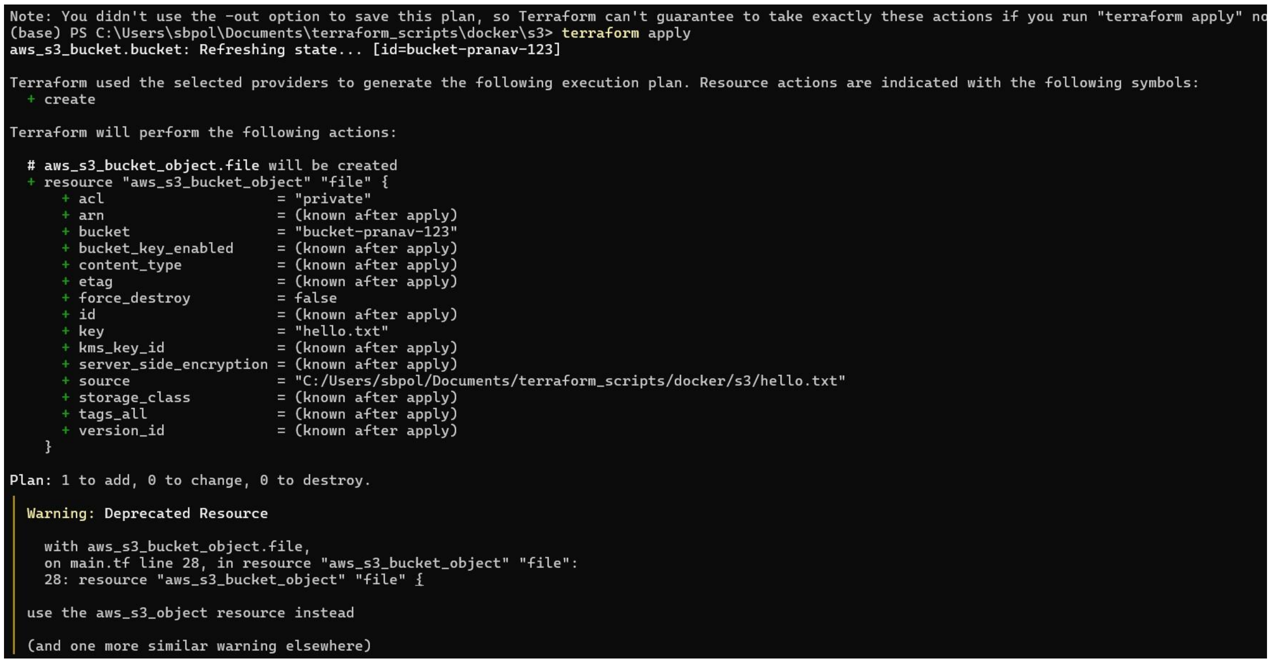


```
resource "aws_s3_bucket" "bucket" {
  bucket = "bucket-pranav-123"

  tags = {
    Name = "My bucket"

  }
}

resource "aws_s3_bucket_object" "file" {
  bucket = aws_s3_bucket.bucket.id
  key    = "hello.txt"
  source = "C:/Users/sbpol/Documents/terraform_scripts/docker/s3/hello.txt"

}
```

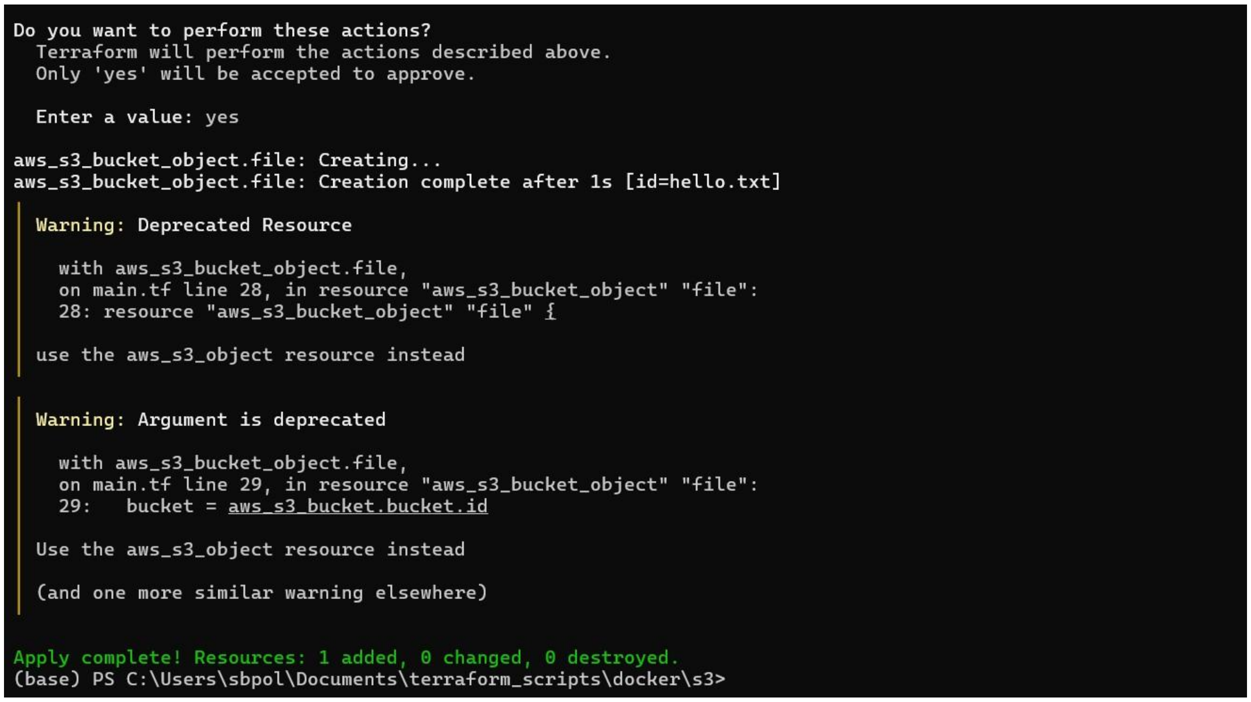
Step 8(EXTRA) : Terraform plan and apply command to apply the changes for file .



```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform plan
aws_s3_bucket.bucket: Refreshing state... [id=bucket-pranav-123]

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_object.file will be created
  + resource "aws_s3_bucket_object" "file" {
      + acl                    = "private"
      + arn                    = (known after apply)
      + bucket                 = "bucket-pranav-123"
      + bucket_key_enabled     = (known after apply)
      + content_type           = (known after apply)
      + etag                   = (known after apply)
      + force_destroy          = false
      + id                     = (known after apply)
      + key                    = "hello.txt"
      + kms_key_id             = (known after apply)
      + server_side_encryption = (known after apply)
      + source                 = "C:/Users/sbpol/Documents/terraform_scripts/docker/s3/hello.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.
│
│ Warning: Deprecated Resource
│
│   with aws_s3_bucket_object.file,
│   on main.tf line 28, in resource "aws_s3_bucket_object" "file":
│   28: resource "aws_s3_bucket_object" "file" {
│
│ use the aws_s3_object resource instead
│
│ (and one more similar warning elsewhere)
```



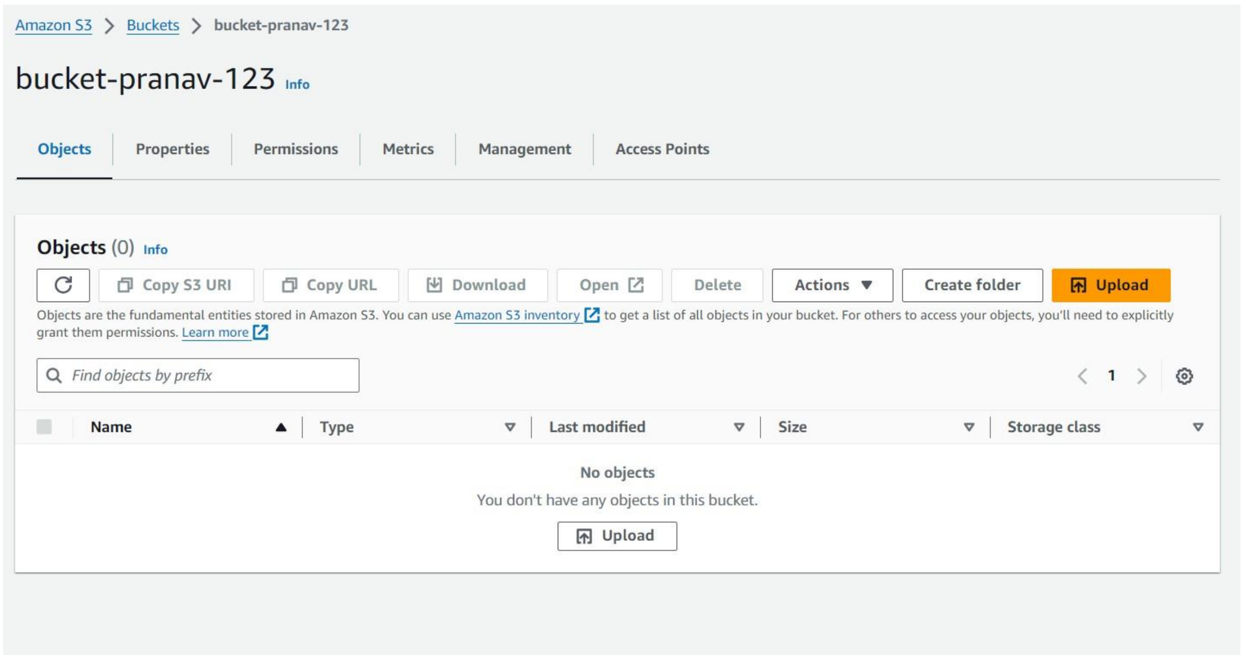
```
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" no
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform apply
aws_s3_bucket.bucket: Refreshing state... [id=bucket-pranav-123]

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_object.file will be created
  + resource "aws_s3_bucket_object" "file" {
      + acl                    = "private"
      + arn                    = (known after apply)
      + bucket                 = "bucket-pranav-123"
      + bucket_key_enabled     = (known after apply)
      + content_type           = (known after apply)
      + etag                   = (known after apply)
      + force_destroy          = false
      + id                     = (known after apply)
      + key                    = "hello.txt"
      + kms_key_id             = (known after apply)
      + server_side_encryption = (known after apply)
      + source                 = "C:/Users/sbpol/Documents/terraform_scripts/docker/s3/hello.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.

│ Warning: Deprecated Resource
│
│   with aws_s3_bucket_object.file,
│   on main.tf line 28, in resource "aws_s3_bucket_object" "file":
│   28: resource "aws_s3_bucket_object" "file" {
│
│ use the aws_s3_object resource instead
│
│ (and one more similar warning elsewhere)
```



```
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_object.file: Creating...
aws_s3_bucket_object.file: Creation complete after 1s [id=hello.txt]
╷
│ Warning: Deprecated Resource
│
│   with aws_s3_bucket_object.file,
│   on main.tf line 28, in resource "aws_s3_bucket_object" "file":
│   28: resource "aws_s3_bucket_object" "file" {
│
│ use the aws_s3_object resource instead
╵
╷
│ Warning: Argument is deprecated
│
│   with aws_s3_bucket_object.file,
│   on main.tf line 29, in resource "aws_s3_bucket_object" "file":
│   29:   bucket = aws_s3_bucket.bucket.id
│
│ Use the aws_s3_object resource instead
│
│ (and one more similar warning elsewhere)
╵

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3>
```

Step 9(EXTRA) : s3 bucket before and after execution of upload

BEFORE -



bucket-pranav-123 Info

Objects (0) Info

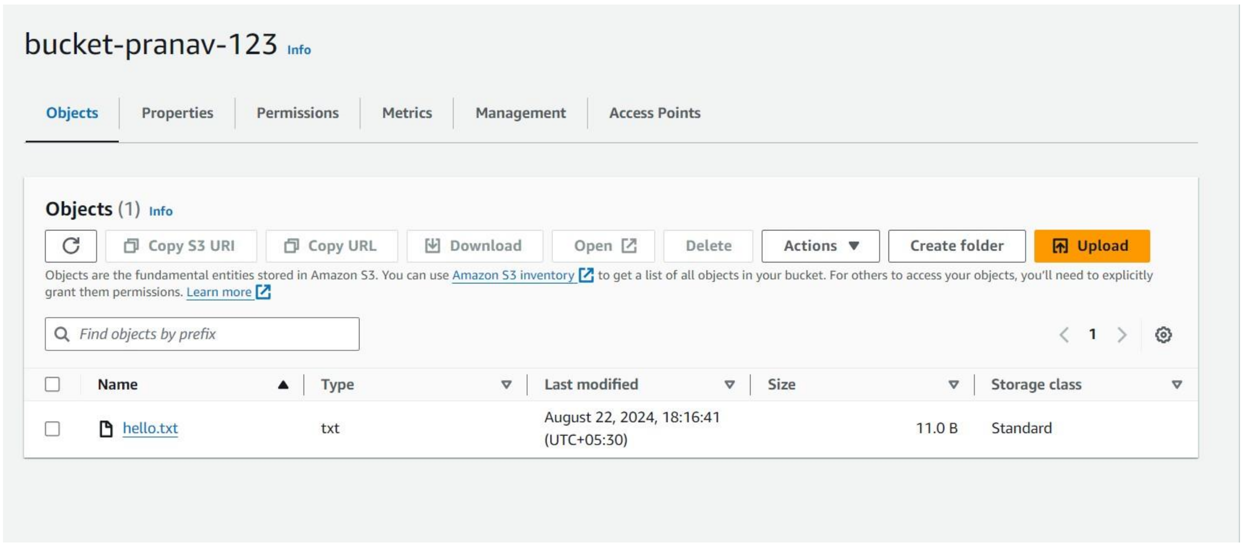
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

| Name | Type | Last modified | Size | Storage class |
|---|---|---|---|---|

No objects

You don't have any objects in this bucket.

AFTER -



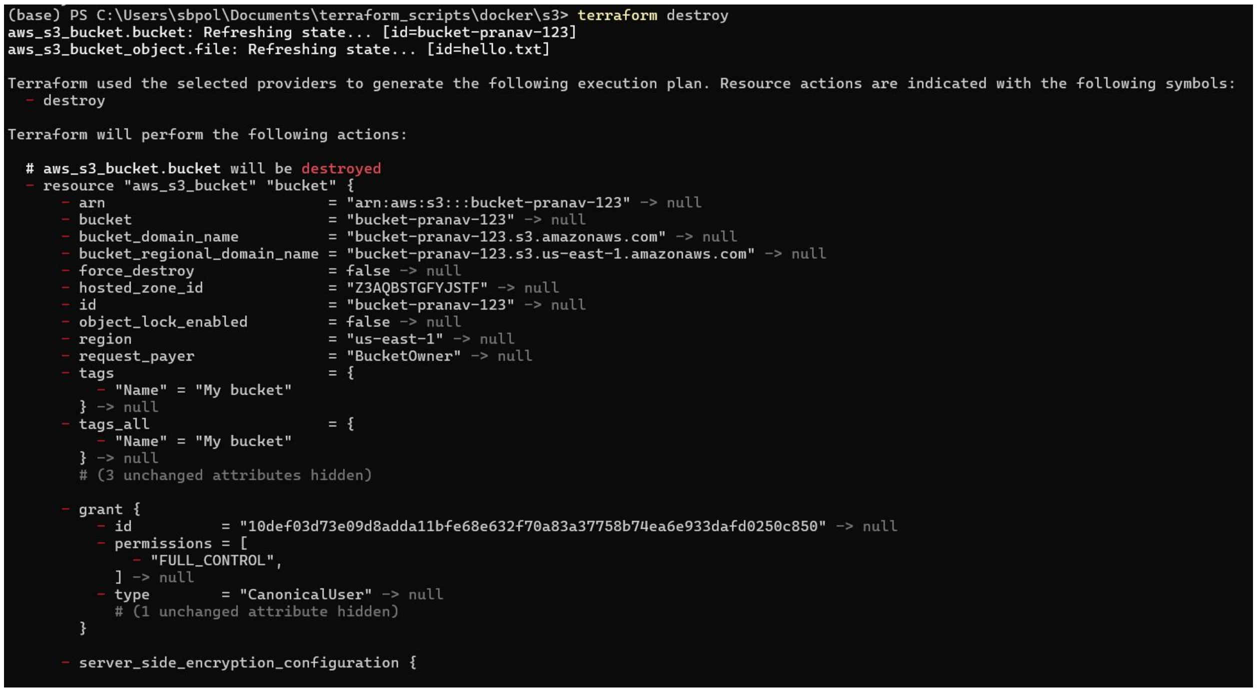
bucket-pranav-123 Info

Objects (1) Info

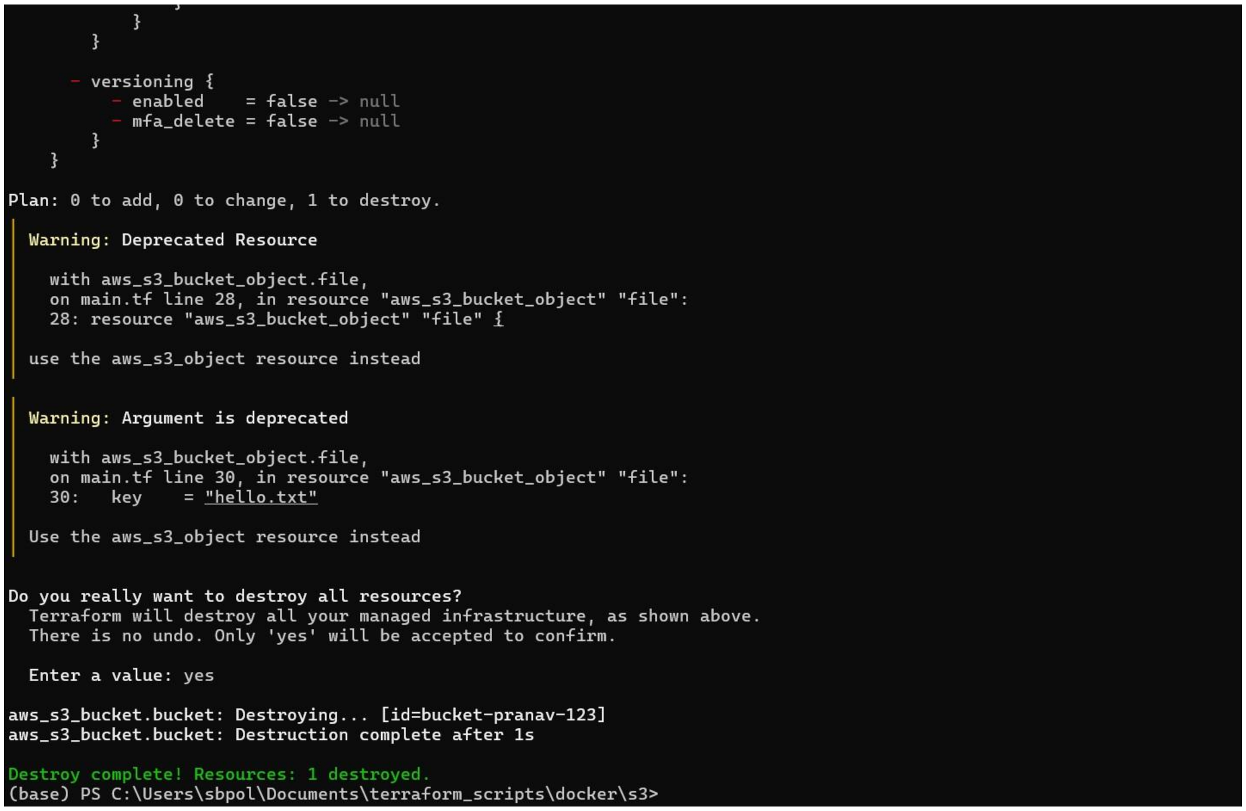
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

| Name | Type | Last modified | Size | Storage class |
|---|---|---|---|---|
| hello.txt | txt | August 22, 2024, 18:16:41 (UTC+05:30) | 11.0 B | Standard |

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

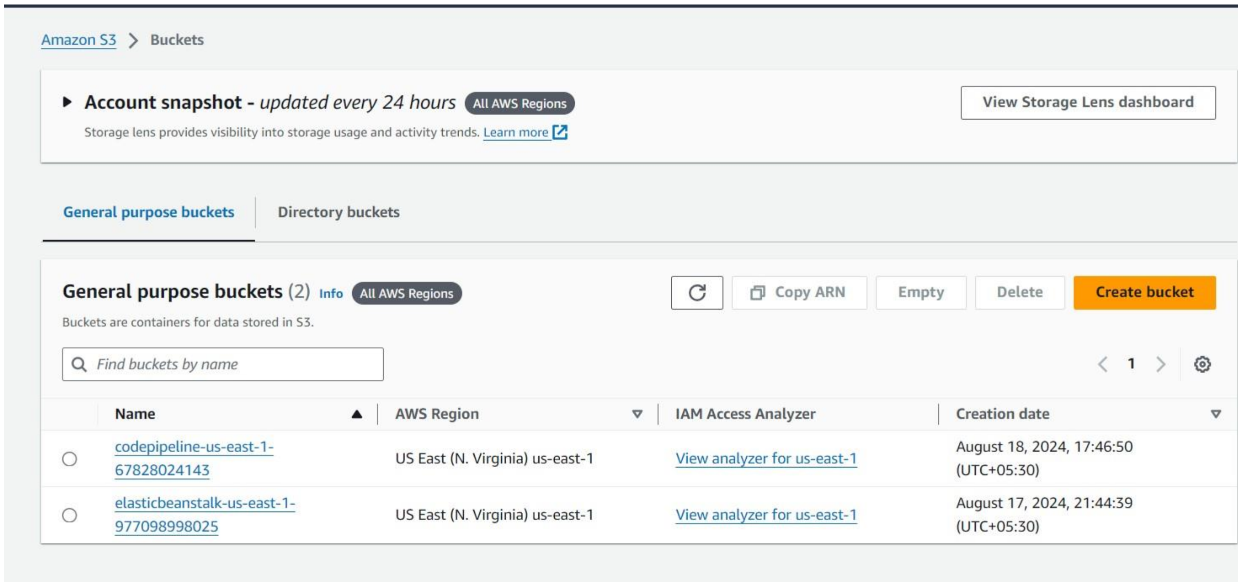


```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3> terraform destroy
aws_s3_bucket.bucket: Refreshing state... [id=bucket-pranav-123]
aws_s3_bucket_object.file: Refreshing state... [id=hello.txt]

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.bucket will be destroyed
  - resource "aws_s3_bucket" "bucket" {
      - arn                         = "arn:aws:s3:::bucket-pranav-123" -> null
      - bucket                      = "bucket-pranav-123" -> null
      - bucket_domain_name          = "bucket-pranav-123.s3.amazonaws.com" -> null
      - bucket_regional_domain_name = "bucket-pranav-123.s3.us-east-1.amazonaws.com" -> null
      - force_destroy               = false -> null
      - hosted_zone_id              = "Z3AQBSTGFYJSTF" -> null
      - id                          = "bucket-pranav-123" -> null
      - object_lock_enabled         = false -> null
      - region                      = "us-east-1" -> null
      - request_payer               = "BucketOwner" -> null
      - tags                        = {
          - "Name" = "My bucket"
        } -> null
      - tags_all                    = {
          - "Name" = "My bucket"
        } -> null
        # (3 unchanged attributes hidden)

      - grant {
          - id          = "10def03d73e09d8adda11bfe68e632f70a83a37758b74ea6e933dafd0250c850" -> null
          - permissions = [
              - "FULL_CONTROL",
            ] -> null
          - type        = "CanonicalUser" -> null
            # (1 unchanged attribute hidden)
        }

      - server_side_encryption_configuration {
```



```
            }
        }

      - versioning {
          - enabled    = false -> null
          - mfa_delete = false -> null
        }
    }

Plan: 0 to add, 0 to change, 1 to destroy.
╷
│ Warning: Deprecated Resource
│
│   with aws_s3_bucket_object.file,
│   on main.tf line 28, in resource "aws_s3_bucket_object" "file":
│   28: resource "aws_s3_bucket_object" "file" {
│
│ use the aws_s3_object resource instead
╵
╷
│ Warning: Argument is deprecated
│
│   with aws_s3_bucket_object.file,
│   on main.tf line 30, in resource "aws_s3_bucket_object" "file":
│   30:   key    = "hello.txt"
│
│ Use the aws_s3_object resource instead
╵

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.bucket: Destroying... [id=bucket-pranav-123]
aws_s3_bucket.bucket: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker\s3>
```

Step 11: s3 after the destroy command execution .



Amazon S3 > Buckets

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 |
|---|---|---|---|
| codepipeline-us-east-1-67828024143 | US East (N. Virginia) us-east-1 | View analyzer for us-east-1 | August 18, 2024, 17:46:50 (UTC+05:30) |
| elasticbeanstalk-us-east-1-977098998025 | US East (N. Virginia) us-east-1 | View analyzer for us-east-1 | August 17, 2024, 21:44:39 (UTC+05:30) |

**Hosting Website on s3 using Terraform (EXTRA) -**

Step 1 :  create main.tf and write following code

Code -

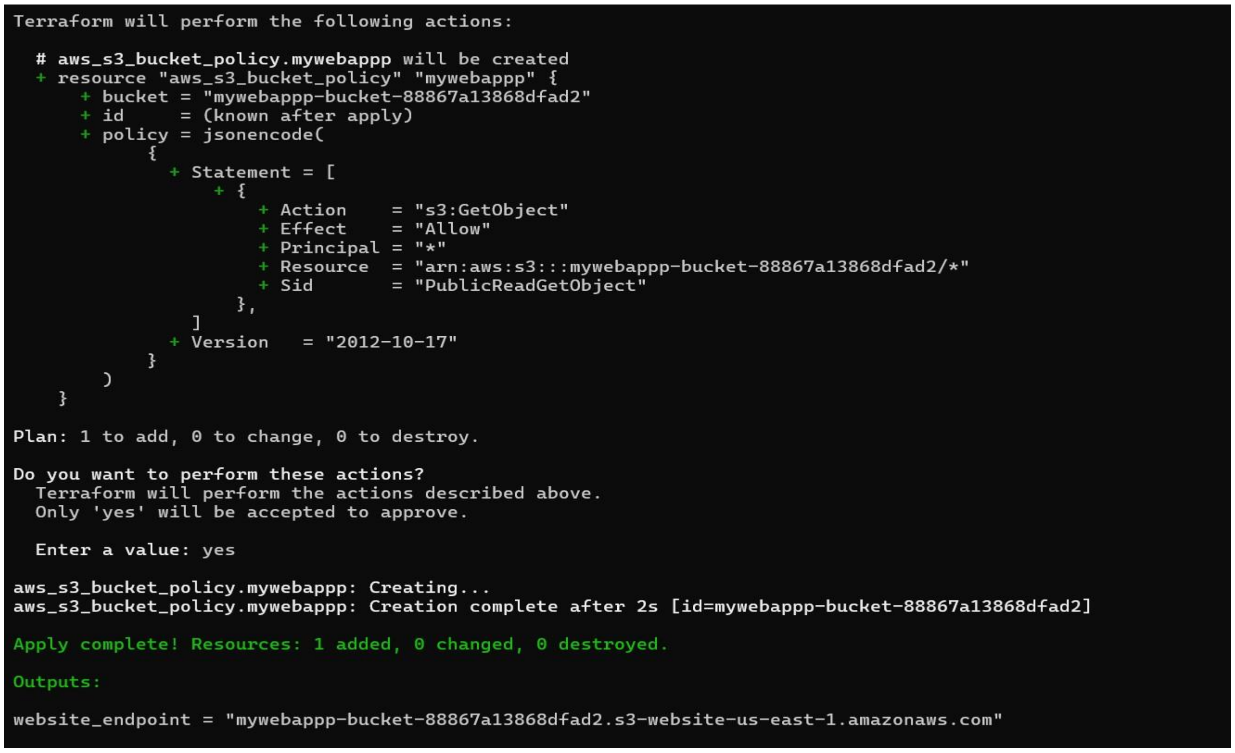
```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "5.64.0"
    }
    random = {
      source  = "hashicorp/random"
      version = "3.6.2"
    }
  }
}

resource "random_id" "rand_id" {
  byte_length = 8
}
resource "aws_s3_bucket" "mywebappp-bucket" {
  bucket = "mywebappp-bucket-${random_id.rand_id.hex}"
}

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

resource "aws_s3_object" "styles_css" {
  bucket       = aws_s3_bucket.mywebappp-bucket.bucket
  source       = "./styles.css"
  key          = "styles.css"
  content_type = "text/css"
}

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

resource "aws_s3_bucket_policy" "mywebappp" {
  bucket = aws_s3_bucket.mywebappp-bucket.id

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


resource "aws_s3_bucket_website_configuration" "example" {
  bucket = aws_s3_bucket.mywebappp-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 -

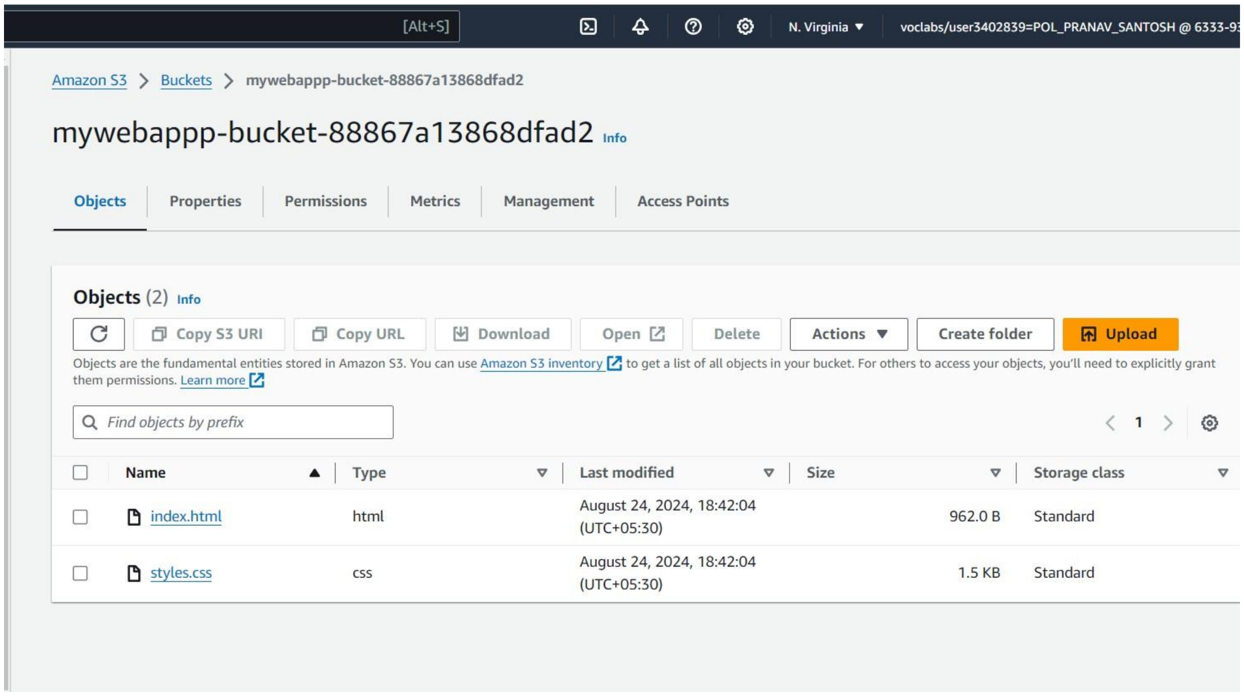
```
provider "aws" {
  access_key="ASIAZG6JVYHRLQ7XABVF"
  secret_key="FV+B+/JDLgRHpPs2bLr9jB+835PQ4cyz7HQ4LAzR"

token="IQoJb3JpZ2luX2VjELT//////////wEaCXVzLXdlc3QtMiJGMEQCIGM45rz6GOsZBjB
cMcCWfAJetwP1F2qgToQCSoJbLE+HAiB2t1XfLcQY0BFOSBsbvJwCmQQ1vQ6/5m4YmzBC1rRel
Cq1Agi9//////////8BEAIaDDYzMzM5Mzc1ODY5MCIM3vgTOnS9B6JyQQmeKokCJkhMaeK5NcX
azpFuqObvIOQpIjKOVtHR/NwxdQCrfqPa2qbn+VsG9i7tF0pvxniO/OQmqxXXaNlRjnq2Qomyd
Ate/91VXJ1cqT7R7k/06ISBc2AVcSAJfgAYEIB7kKVF2UkY01VJ845VjTPnER7O4enKd5jYyHa
kuOkj29olSph1sjrq6VFYBo0foLgLJcDsL/QbipTk8HXX7XT8f/Gh8jGKfUjy2CUvJfuAAX3zv
sTFjSsGEb69J1pZd0sQfoBGi6Mv0vezW+ljWX+dLdpnzDEJrnk0x7g6po1uXrCjDF6+pB+5QwP
hI78D2lF/tcLahLbr5El6ri2DXv0eQ0woOaL6u0xsKDPvwzDCkqe2BjqeAYi5Fs7WB0Ei5FiAq
HdJEzXcQZI18JX5H59W3p+v71sN7sGLxJYrXoMmFLH7amaZxQ7r5xkn9/is6Ge3ZcuxROIy5GO
LuqoHVsNRxCRQ83ZoIewd32TRN8h3uRLQnE7ZMf6ggljBvqvT1e2IlA+YcdeWrkeM/fCXJ0g7k
KEcnkNgBMv+W9LXi2P8DMsm0AnP6jhFK5R6Ch16JI+ePiL1"
  region="us-east-1"
}
```

Step 3: Execute Terraform init , terraform plan and terraform apply command.



```
Terraform will perform the following actions:

  # aws_s3_bucket_policy.mywebappp will be created
  + resource "aws_s3_bucket_policy" "mywebappp" {
      + bucket = "mywebappp-bucket-88867a13868dfad2"
      + id     = (known after apply)
      + policy = jsonencode(
            {
              + Statement = [
                  + {
                      + Action    = "s3:GetObject"
                      + Effect    = "Allow"
                      + Principal = "*"
                      + Resource  = "arn:aws:s3:::mywebappp-bucket-88867a13868dfad2/*"
                      + Sid       = "PublicReadGetObject"
                    },
                ]
              + 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.mywebappp: Creating...
aws_s3_bucket_policy.mywebappp: Creation complete after 2s [id=mywebappp-bucket-88867a13868dfad2]

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

Outputs:

website_endpoint = "mywebappp-bucket-88867a13868dfad2.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



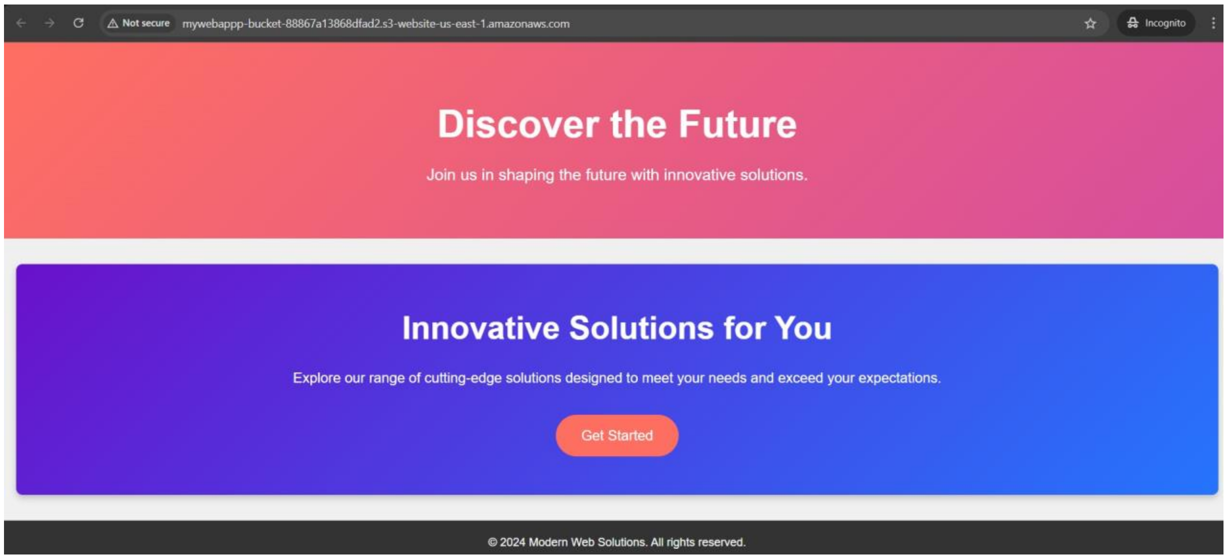
Amazon S3 > Buckets > mywebappp-bucket-88867a13868dfad2

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 grant 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 |



mywebappp-bucket-88867a13868dfad2.s3-website-us-east-1.amazonaws.com

Discover the Future

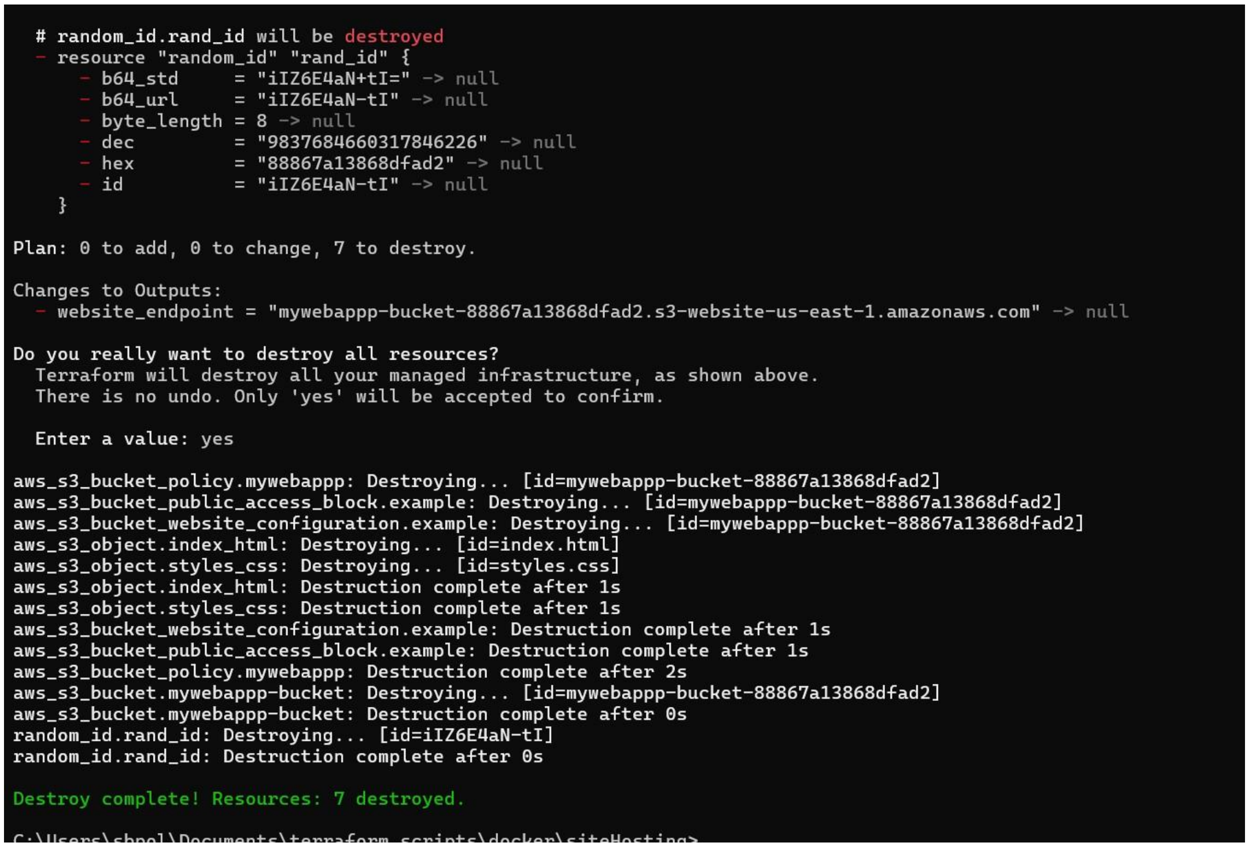
Join us in shaping the future with innovative solutions.

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.

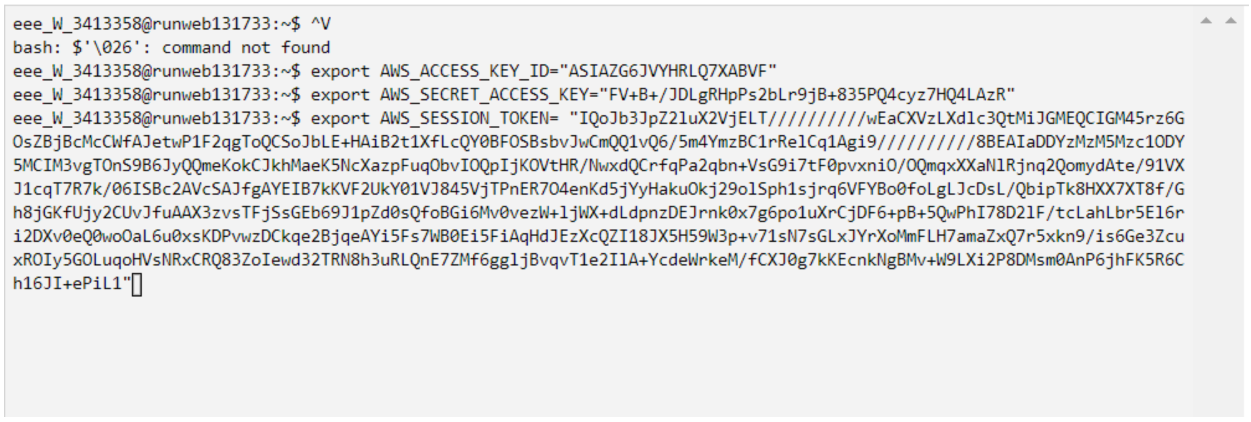
Step 5 : terraform destroy to destroy the bucket



```
  # random_id.rand_id will be destroyed
  - resource "random_id" "rand_id" {
      - b64_std     = "iIZ6E4aN+tI=" -> null
      - b64_url     = "iIZ6E4aN-tI" -> null
      - byte_length = 8 -> null
      - dec         = "9837684660317846226" -> null
      - hex         = "88867a13868dfad2" -> null
      - id          = "iIZ6E4aN-tI" -> null
    }

Plan: 0 to add, 0 to change, 7 to destroy.

Changes to Outputs:
  - website_endpoint = "mywebappp-bucket-88867a13868dfad2.s3-website-us-east-1.amazonaws.com" -> null

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_policy.mywebappp: Destroying... [id=mywebappp-bucket-88867a13868dfad2]
aws_s3_bucket_public_access_block.example: Destroying... [id=mywebappp-bucket-88867a13868dfad2]
aws_s3_bucket_website_configuration.example: Destroying... [id=mywebappp-bucket-88867a13868dfad2]
aws_s3_object.index_html: Destroying... [id=index.html]
aws_s3_object.styles_css: Destroying... [id=styles.css]
aws_s3_object.index_html: Destruction complete after 1s
aws_s3_object.styles_css: Destruction complete after 1s
aws_s3_bucket_website_configuration.example: Destruction complete after 1s
aws_s3_bucket_public_access_block.example: Destruction complete after 1s
aws_s3_bucket_policy.mywebappp: Destruction complete after 2s
aws_s3_bucket.mywebappp-bucket: Destroying... [id=mywebappp-bucket-88867a13868dfad2]
aws_s3_bucket.mywebappp-bucket: Destruction complete after 0s
random_id.rand_id: Destroying... [id=iIZ6E4aN-tI]
random_id.rand_id: Destruction complete after 0s

Destroy complete! Resources: 7 destroyed.

C:\Users\shpol\Documents\terraform_scripts\docker\siteHosting>
```

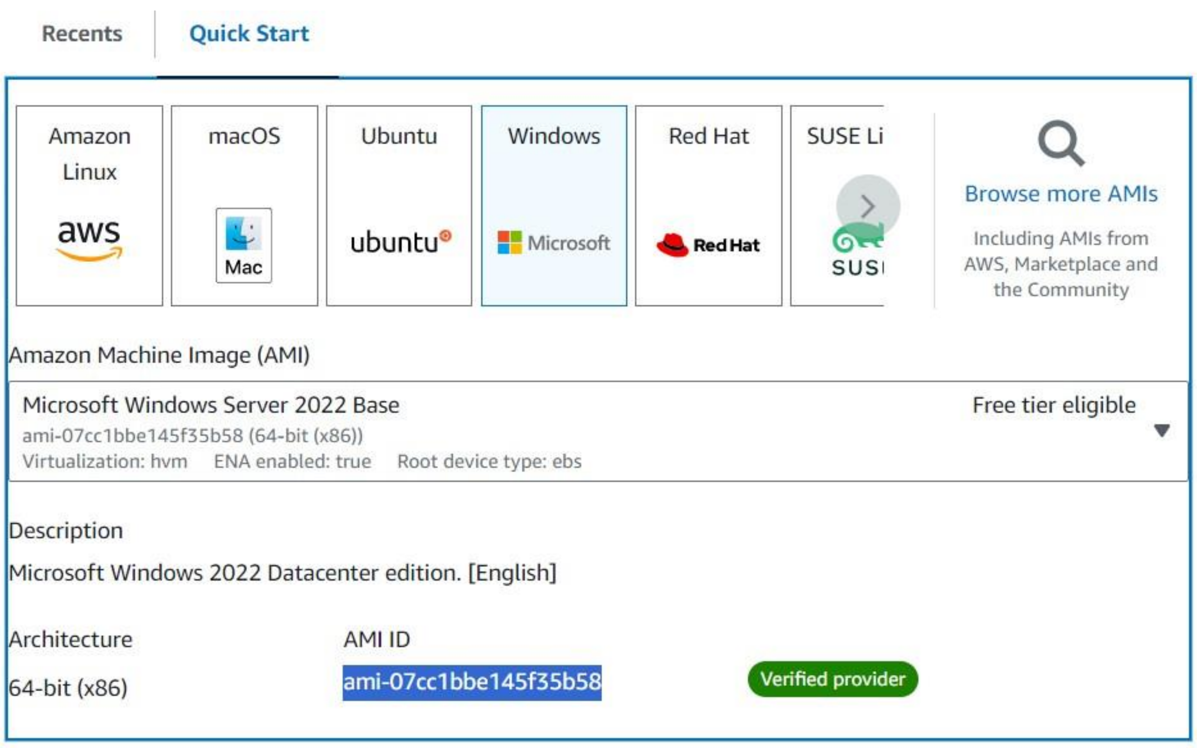
**Creating EC2 instance using Terraform** (EXTRA) -

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+ljWX+dLdpnzDEJrnk0x7g6po1uXrCjDF6+pB+5QwPhI78D2lF/tcLahLbr5El6r
i2DXv0eQ0woOaL6u0xsKDPvwzDCkqe2BjqeAYi5Fs7WB0Ei5FiAqHdJEzXcQZI18JX5H59W3p+v71sN7sGLxJYrXoMmFLH7amaZxQ7r5xkn9/is6Ge3Zcu
xROIy5GOLuqoHVsNRxCRQ83ZoIewd32TRN8h3uRLQnE7ZMf6ggljBvqvT1e2IlA+YcdeWrkeM/fCXJ0g7kKEcnkNgBMv+W9LXi2P8DMsm0AnP6jhFK5R6C
h16JI+ePiL1"
```

Step 2 : copy the AMI ID from the EC2



Amazon Machine Image (AMI)

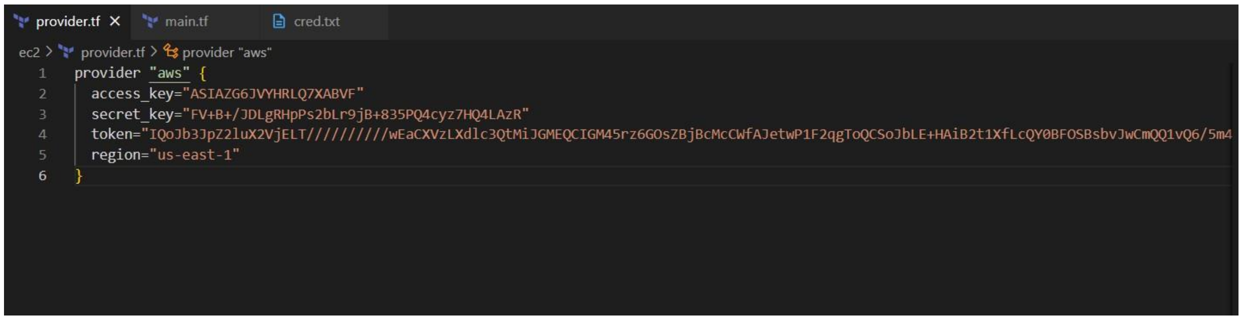
Microsoft Windows Server 2022 Base — Free tier eligible
ami-07cc1bbe145f35b58 (64-bit (x86))
Virtualization: hvm  ENA enabled: true  Root device type: ebs

Description

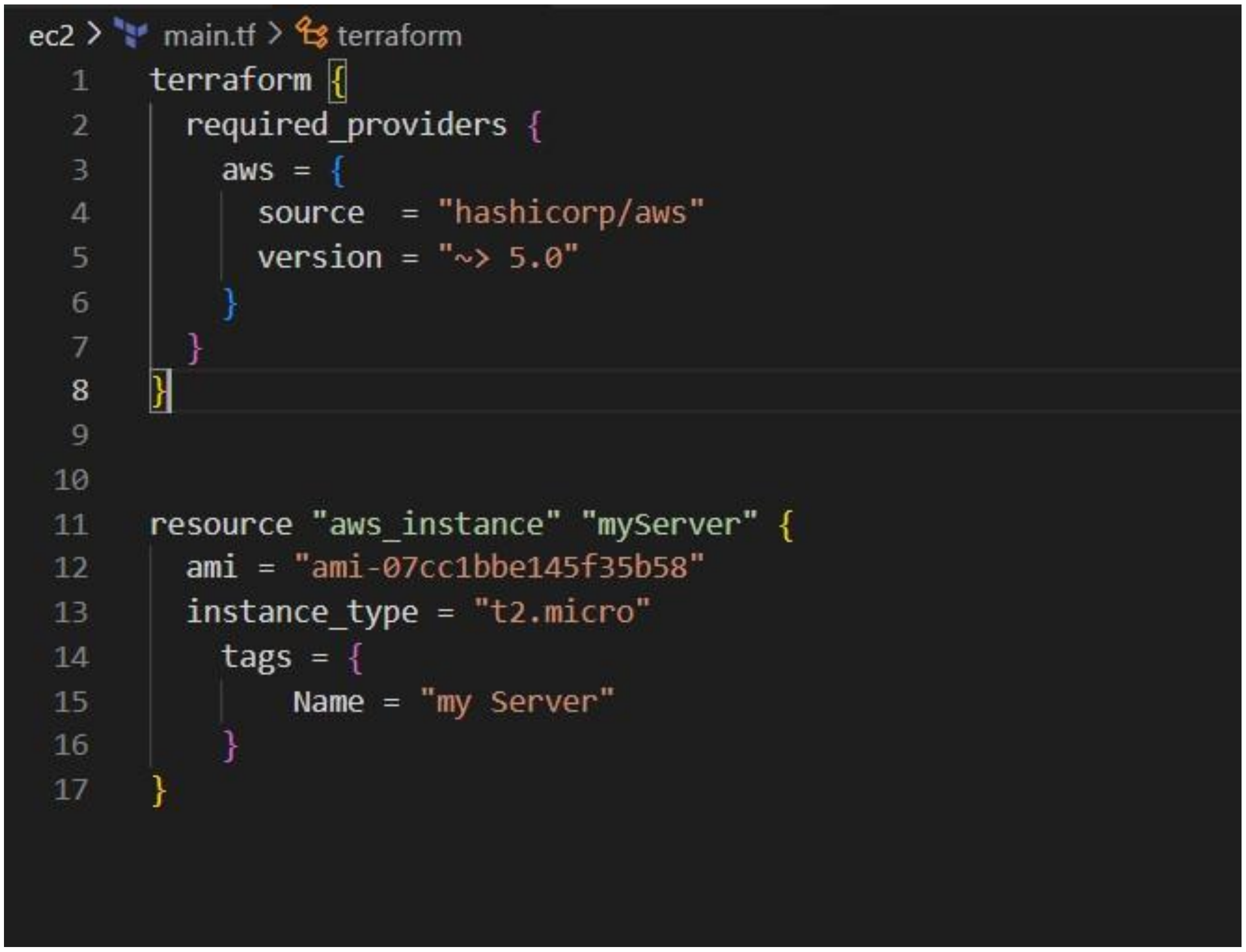
Microsoft Windows 2022 Datacenter edition. [English]

| Architecture | AMI ID | |
|---|---|---|
| 64-bit (x86) | ami-07cc1bbe145f35b58 | Verified provider |

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

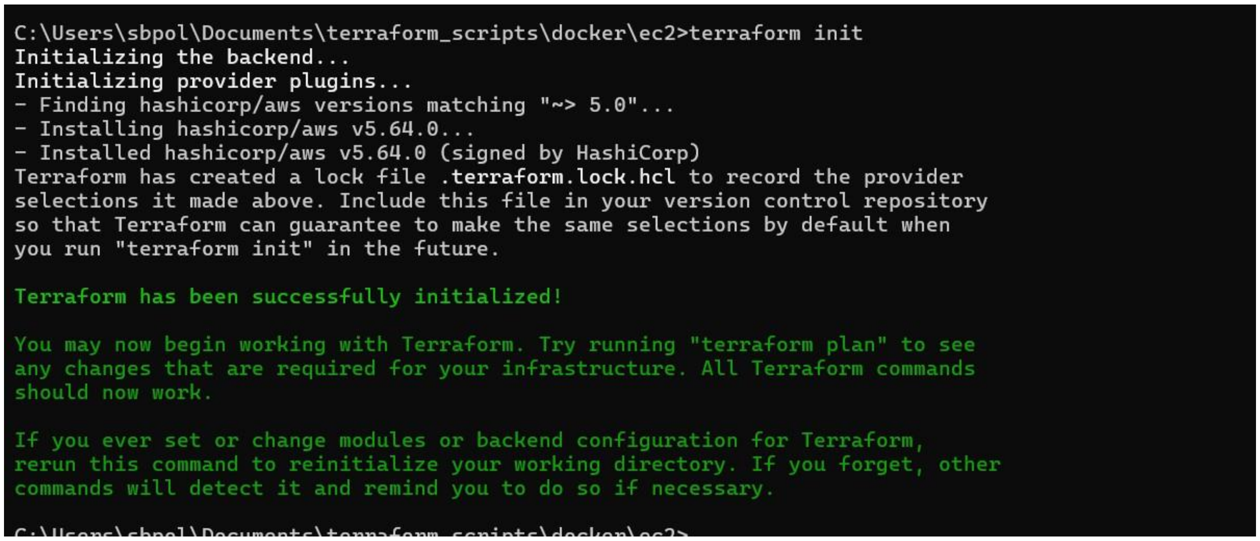


```
provider "aws" {
  access_key="ASIAZG6JVYHRLQ7XABVF"
  secret_key="FV+B+/JDLgRHpPs2bLr9jB+835PQ4cyz7HQ4LAzR"
  token="IQoJb3JpZ2luX2VjELT//////////wEaCXVzLXdlc3QtMiJGMEQCIGM45rz6GOsZBjBcMcCWfAJetwP1F2qgToQCSoJbLE+HAiB2t1XfLcQY0BFOSBsbvJwCmQQ1vQ6/5m4
  region="us-east-1"
}
```

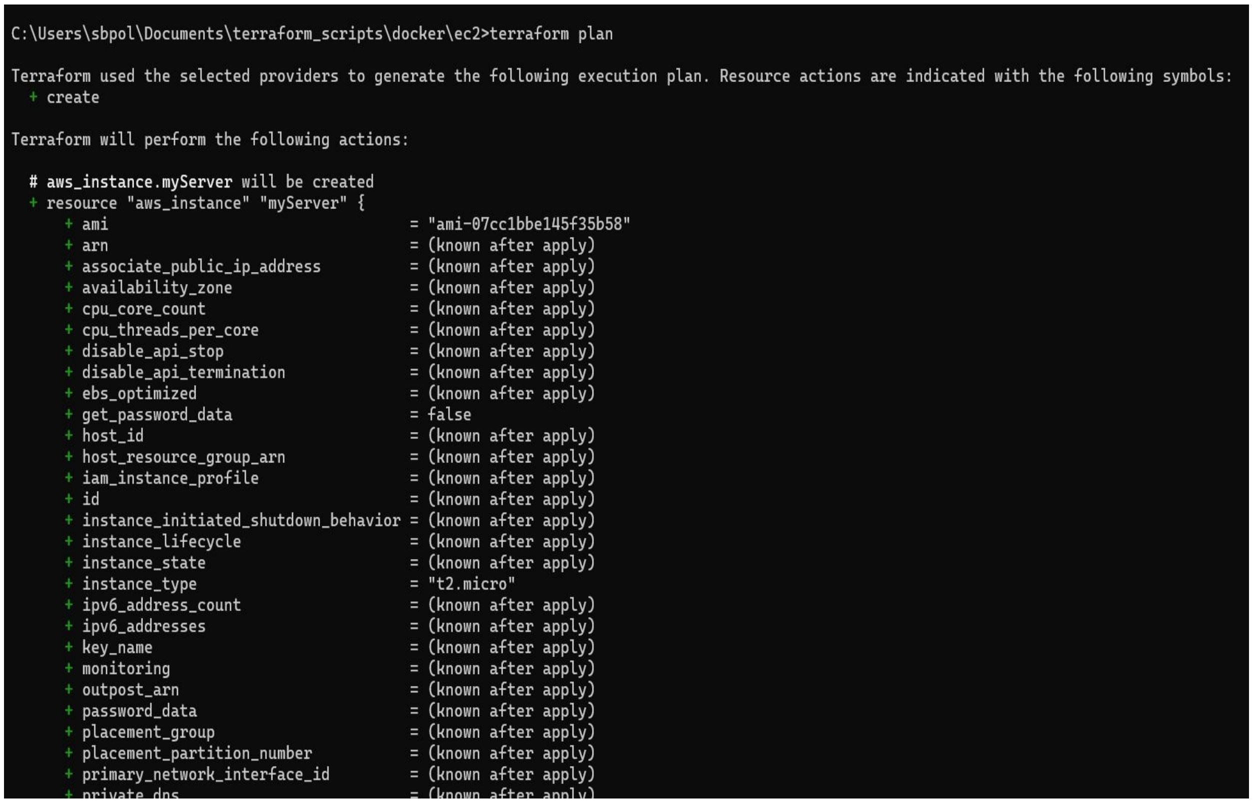


```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}


resource "aws_instance" "myServer" {
  ami = "ami-07cc1bbe145f35b58"
  instance_type = "t2.micro"
    tags = {
        Name = "my Server"
    }
}
```

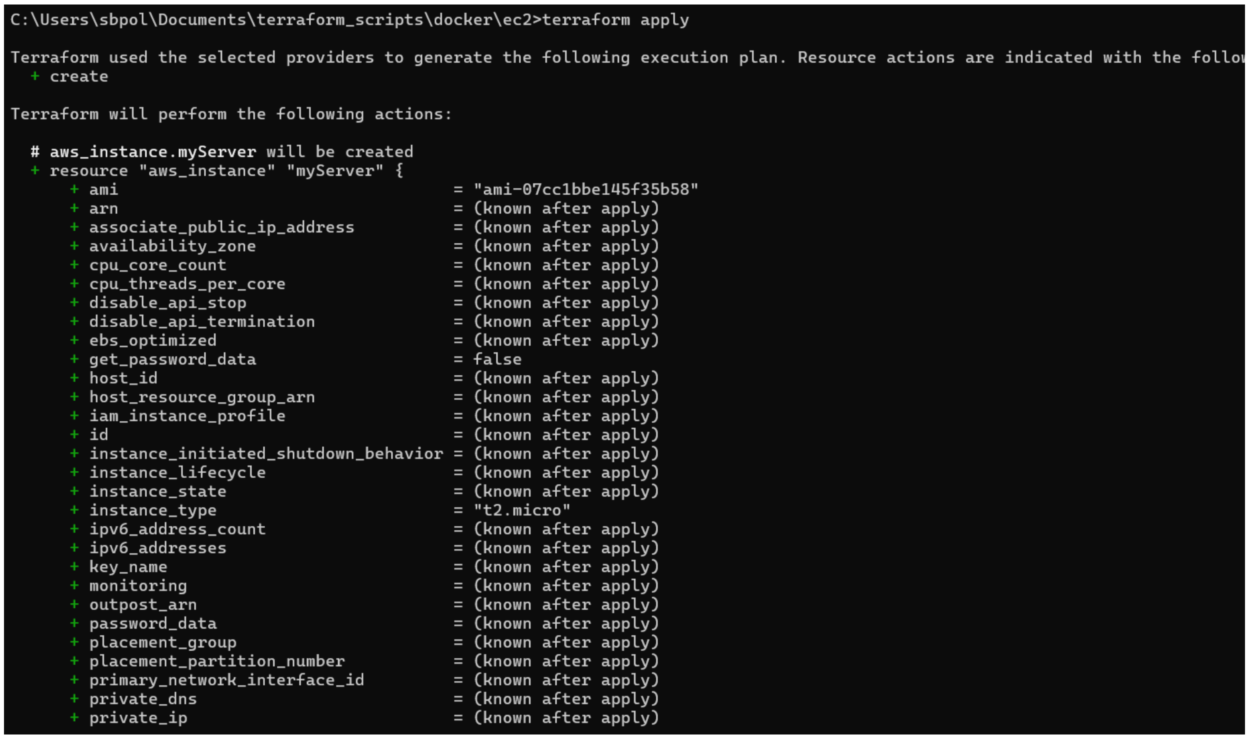
Step 4 : Execute terraform init , terraform plan and terraform apply command

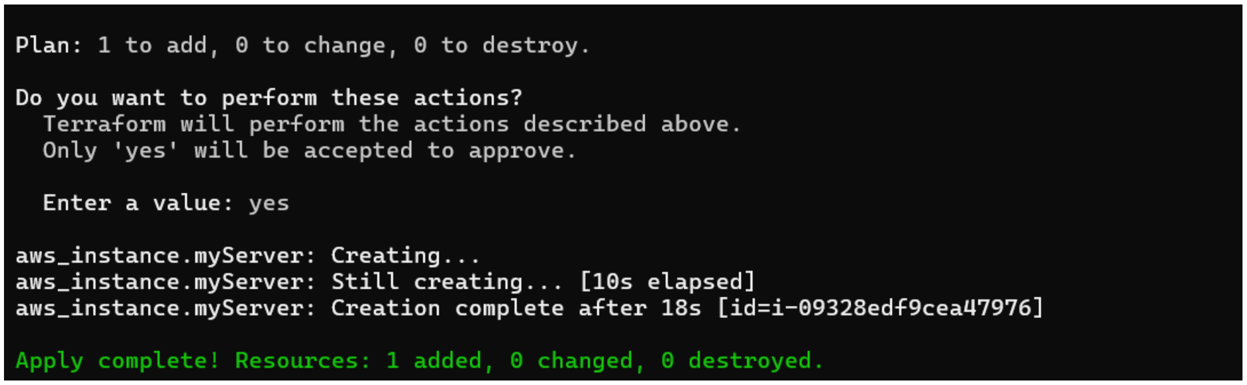


```
C:\Users\sbpol\Documents\terraform_scripts\docker\ec2>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.64.0...
- Installed hashicorp/aws v5.64.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record 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.
```

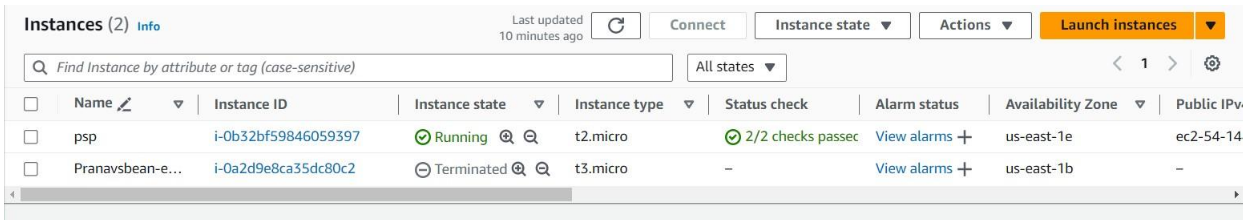


```
C:\Users\sbpol\Documents\terraform_scripts\docker\ec2>terraform plan

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)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
```


```
C:\Users\sbpol\Documents\terraform_scripts\docker\ec2>terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the follo
  + 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)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (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: Creation complete after 18s [id=i-09328edf9cea47976]

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

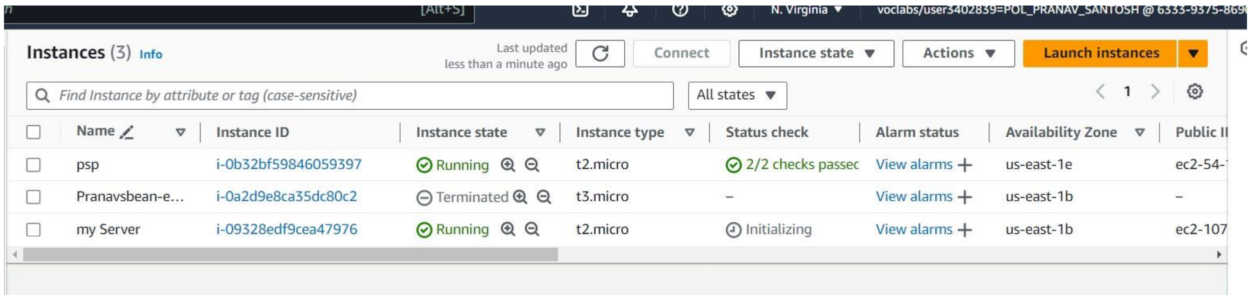
Step 5 : Ec2 before and after instance creation .
BEFORE -



Instances (2) Info
Last updated 10 minutes ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

| | Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv |
|---|---|---|---|---|---|---|---|---|
| ☐ | psp | i-0b32bf59846059397 | Running | t2.micro | 2/2 checks passec | View alarms + | us-east-1e | ec2-54-14 |
| ☐ | Pranavsbean-e... | i-0a2d9e8ca35dc80c2 | Terminated | t3.micro | – | View alarms + | us-east-1b | – |

AFTER -



N. Virginia ▼ voclabs/user3402839=POL_PRANAV_SANTOSH @ 6333-9375-869

Instances (3) Info

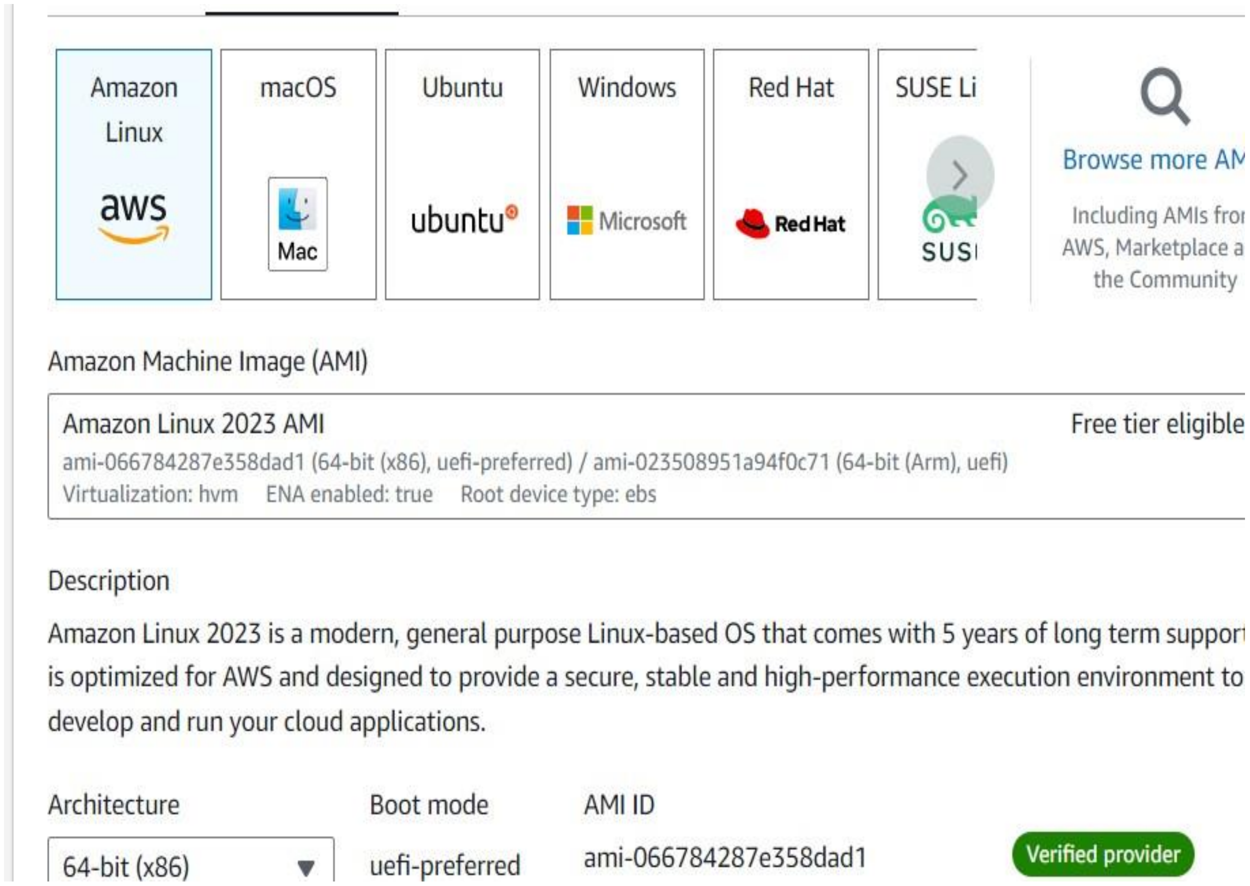
Last updated less than a minute ago

Connect | Instance state ▼ | Actions ▼ | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states ▼

| | Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public I |
|---|---|---|---|---|---|---|---|---|
| ☐ | psp | i-0b32bf59846059397 | Running | t2.micro | 2/2 checks passec | View alarms + | us-east-1e | ec2-54- |
| ☐ | Pranavsbean-e... | i-0a2d9e8ca35dc80c2 | Terminated | t3.micro | – | View alarms + | us-east-1b | – |
| ☐ | my Server | i-09328edf9cea47976 | Running | t2.micro | Initializing | View alarms + | us-east-1b | ec2-107 |

Step 6 : Copy AWS AMI ID and change it in code



Amazon Machine Image (AMI)

Amazon Linux 2023 AMI Free tier eligible

ami-066784287e358dad1 (64-bit (x86), uefi-preferred) / ami-023508951a94f0c71 (64-bit (Arm), uefi)

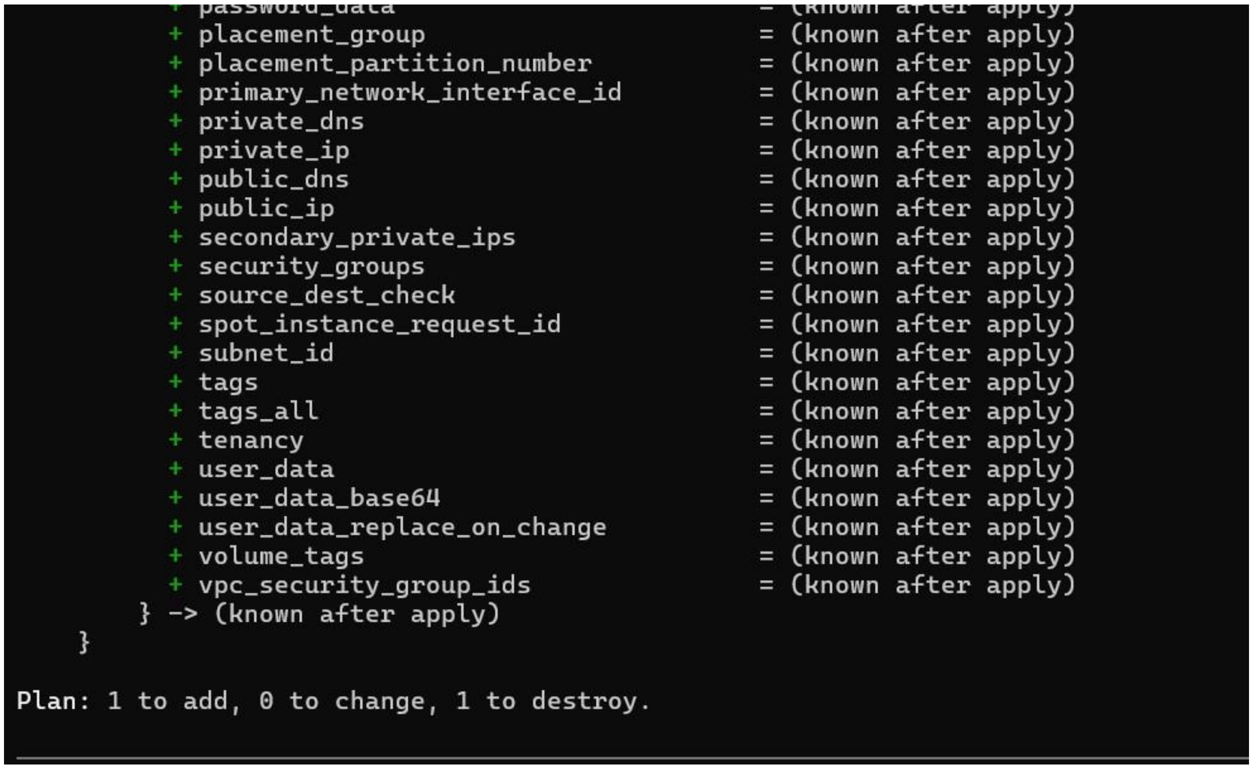
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

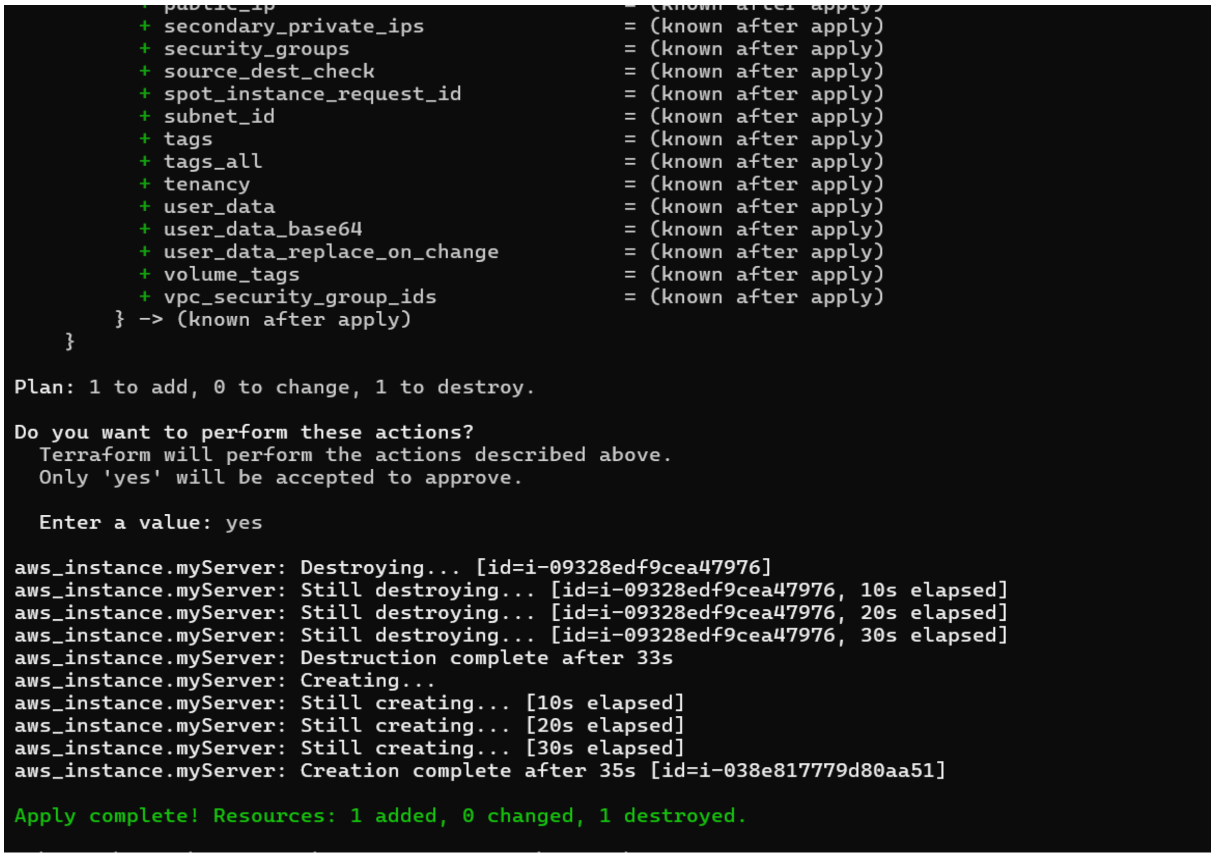
Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term suppor is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

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

Step 7 : Type terraform plan and terraform apply command.

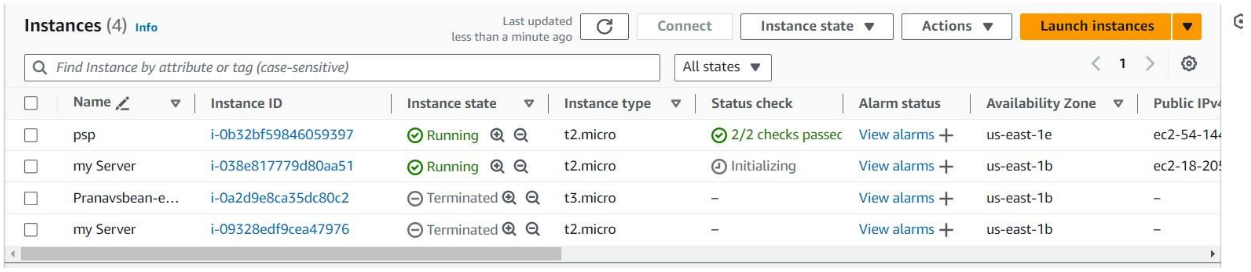


```
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = (known after apply)
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags                                 = (known after apply)
      + tags_all                             = (known after apply)
      + tenancy                              = (known after apply)
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = (known after apply)
      + volume_tags                          = (known after apply)
      + vpc_security_group_ids               = (known after apply)
    } -> (known after apply)
  }

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



```
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = (known after apply)
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags                                 = (known after apply)
      + tags_all                             = (known after apply)
      + tenancy                              = (known after apply)
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = (known after apply)
      + volume_tags                          = (known after apply)
      + vpc_security_group_ids               = (known after apply)
    } -> (known after apply)
  }

Plan: 1 to add, 0 to change, 1 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: Destroying... [id=i-09328edf9cea47976]
aws_instance.myServer: Still destroying... [id=i-09328edf9cea47976, 10s elapsed]
aws_instance.myServer: Still destroying... [id=i-09328edf9cea47976, 20s elapsed]
aws_instance.myServer: Still destroying... [id=i-09328edf9cea47976, 30s elapsed]
aws_instance.myServer: Destruction complete after 33s
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 35s [id=i-038e817779d80aa51]

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

Step 8 : Instances after deleting window instance and creating AWS instance

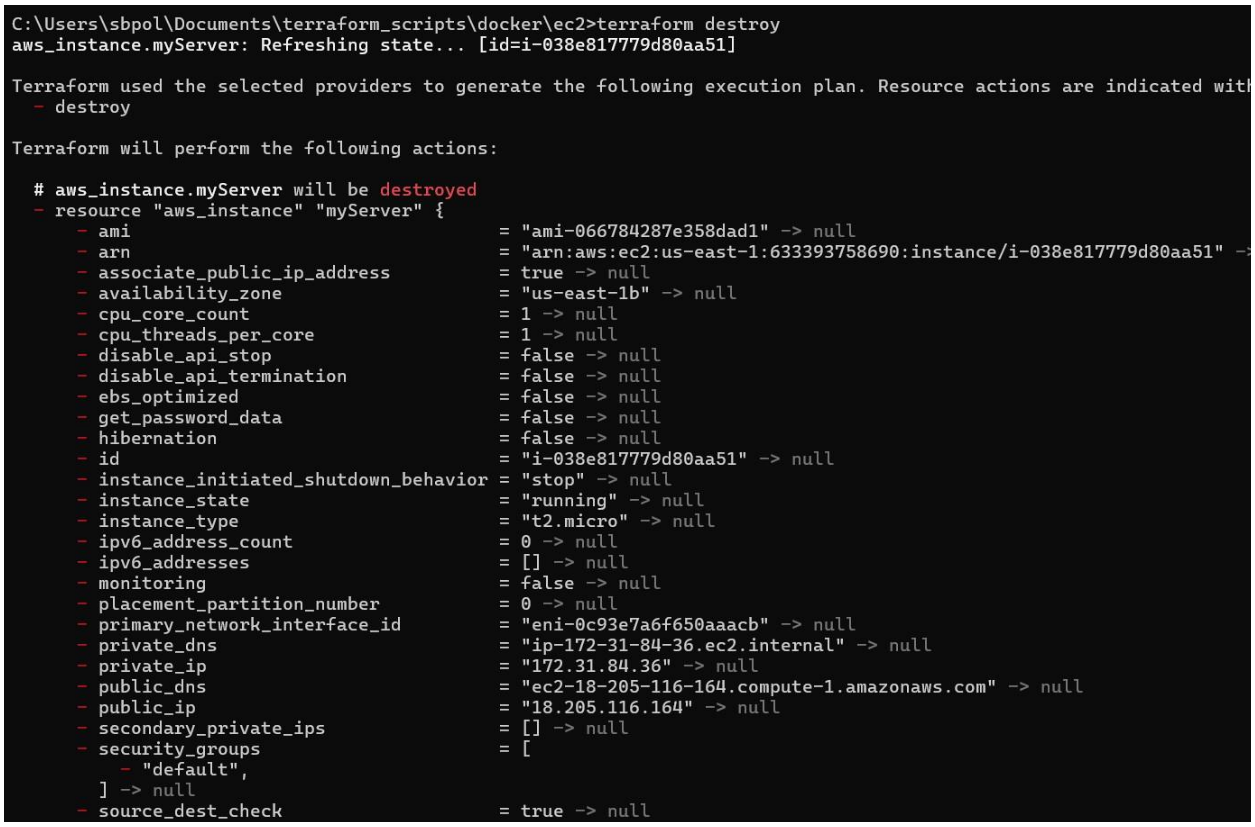


Instances (4) Info    Last updated less than a minute ago    Connect    Instance state    Actions    Launch instances

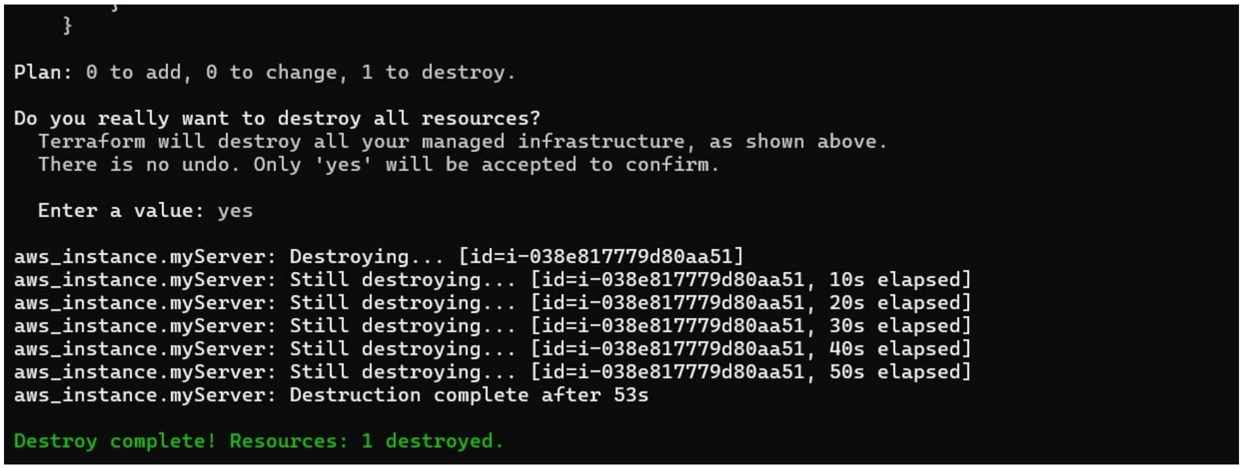
Find Instance by attribute or tag (case-sensitive)    All states    1

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 |
|---|---|---|---|---|---|---|---|
| psp | i-0b32bf59846059397 | Running | t2.micro | 2/2 checks passec | View alarms + | us-east-1e | ec2-54-14 |
| my Server | i-038e817779d80aa51 | Running | t2.micro | Initializing | View alarms + | us-east-1b | ec2-18-20 |
| Pranavsbean-e... | i-0a2d9e8ca35dc80c2 | Terminated | t3.micro | – | View alarms + | us-east-1b | – |
| my Server | i-09328edf9cea47976 | Terminated | t2.micro | – | View alarms + | us-east-1b | – |

Step 9 : Destroy the instance using terraform destroy



```
C:\Users\sbpol\Documents\terraform_scripts\docker\ec2>terraform destroy
aws_instance.myServer: Refreshing state... [id=i-038e817779d80aa51]

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

Terraform will perform the following actions:

  # aws_instance.myServer will be destroyed
  - resource "aws_instance" "myServer" {
      - ami                                  = "ami-066784287e358dad1" -> null
      - arn                                  = "arn:aws:ec2:us-east-1:633393758690:instance/i-038e817779d80aa51" -
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "us-east-1b" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-038e817779d80aa51" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-0c93e7a6f650aaacb" -> null
      - private_dns                          = "ip-172-31-84-36.ec2.internal" -> null
      - private_ip                           = "172.31.84.36" -> null
      - public_dns                           = "ec2-18-205-116-164.compute-1.amazonaws.com" -> null
      - public_ip                            = "18.205.116.164" -> null
      - secondary_private_ips                = [] -> null
      - security_groups                      = [
          - "default",
        ] -> null
      - source_dest_check                    = true -> 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_instance.myServer: Destroying... [id=i-038e817779d80aa51]
aws_instance.myServer: Still destroying... [id=i-038e817779d80aa51, 10s elapsed]
aws_instance.myServer: Still destroying... [id=i-038e817779d80aa51, 20s elapsed]
aws_instance.myServer: Still destroying... [id=i-038e817779d80aa51, 30s elapsed]
aws_instance.myServer: Still destroying... [id=i-038e817779d80aa51, 40s elapsed]
aws_instance.myServer: Still destroying... [id=i-038e817779d80aa51, 50s elapsed]
aws_instance.myServer: Destruction complete after 53s

Destroy complete! Resources: 1 destroyed.
```