

EXPERIMENT NO. 6

NAME : PRANAV POL

CLASS : D15A

ROLL NO. : 42

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.

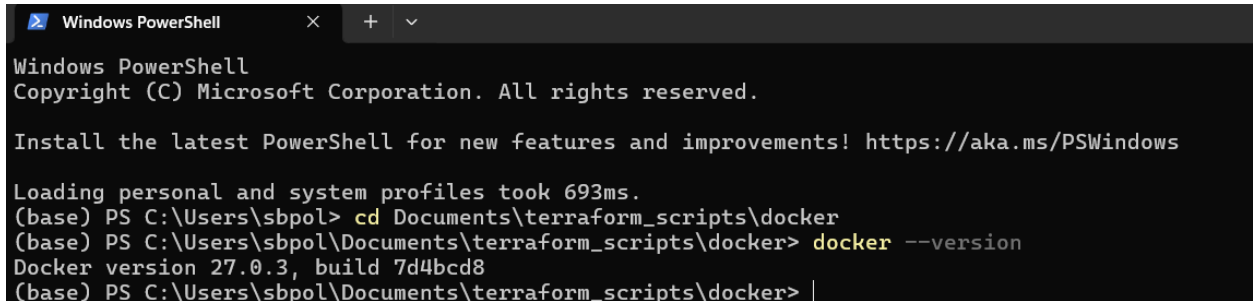
Benefits of Using Terraform for AWS Infrastructure

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

A screenshot of a Windows PowerShell terminal window. The title bar says 'Windows PowerShell'. The text inside shows the standard PowerShell startup messages, including the copyright notice and a link to update PowerShell. The user has navigated to the directory 'C:\Users\sbpol\Documents\terraform_scripts\docker' and executed the command 'docker --version'. The output shows 'Docker version 27.0.3, build 7d4bcd8'.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Loading personal and system profiles took 693ms.
(base) PS C:\Users\sbpol> cd Documents\terraform_scripts\docker
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> docker --version
Docker version 27.0.3, build 7d4bcd8
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> |
```

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  X  
docker.tf  
1 terraform {  
2   required_providers {  
3     docker = {  
4       source = "kreuzwerker/docker"  
5       version = "~> 3.0.1"  
6     }  
7   }  
8 }  
9  
10 provider "docker" {  
11   host = "npipe:////.//pipe//docker_engine"  
12 }  
13  
14 resource "docker_image" "nginx" {  
15   name = "nginx:latest"  
16   keep_locally = false  
17 }  
18  
19 resource "docker_container" "nginx" {  
20   image = docker_image.nginx.image_id  
21   name = "tutorial"  
22   ports {  
23     internal = 80  
24     external = 8000  
25   }  
26 }  
27
```

Step 3 : Type terraform init command to initialize terraform backend

```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> 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 6(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.

```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> terraform fmt
docker.tf
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> terraform validate
Success! The configuration is valid.
```

Step 7 : Type Terraform plan command to create execution plan .

```
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> 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
  + stop_signal      = (known after apply)
  + stop_timeout     = (known after apply)
  + tty              = 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

+ 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 8 : Type terraform apply to apply changes .

```

(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 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
  + 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=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03cnginx:late
docker_container.nginx: Creating...
docker_container.nginx: Creation complete after 1s [id=c25805e4484164520912c50ac3080526c9926219c98c673021078772eb484357]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker>

```

Step 9 : Docker container before and after step 8 execution

BEFORE -

```

(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> terraform plan

```

AFTER -

```

(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> docker container list
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
c25805e44841   5ef79149e0ec   "/docker-entrypoint..."   About a minute ago   Up About a minute   0.0.0.0:8000->80/tcp   tutorial
(base) PS C:\Users\sbpol\Documents\terraform_scripts\docker> docker images
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
nginx         latest   5ef79149e0ec   6 days ago   188MB

```

Step 10 (EXTRA) : Execution of change .

```
docker.tf
1  terraform {
2    required_providers {
3      docker = {
4        source = "kreuzwerker/docker"
5        version = "~> 3.0.1"
6      }
7    }
8  }
9
10 provider "docker" {
11   host = "npipe:////./pipe//docker_engine"
12 }
13
14 resource "docker_image" "nginx" {
15   name          = "nginx:latest"
16   keep_locally = false
17 }
18
19 resource "docker_container" "nginx" {
20   image = docker_image.nginx.image_id
21   name  = "tutorial"
22   ports {
23     internal = 80
24     external = 8080
25   }
26 }
27
```



```

+ 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 11 : 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           = "sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03c" -> 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=sha256:5ef79149e0ec84a7a9f9284c3f91aa3c20608f8391f5445eabe92ef07dbda03c]
docker_image.nginx: Destruction complete after 0s

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

```

Step 12 : 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
1  terraform {
2    required_providers {
3      aws = {
4        source  = "hashicorp/aws"
5        version = "~> 5.0"
6      }
7    }
8  }
9
10 # Configure the AWS Provider
11 provider "aws" {
12   region = "us-east-1"
13   access_key = ""
14   secret_key = ""
15 }
16
17
18
19 resource "aws_s3_bucket" "bucket" {
20   bucket = "bucket-pranav-123"
21
22   tags = {
23     Name = "My bucket"
24   }
25 }
26

```

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\sbspol\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

Directory buckets

General purpose buckets (3) Info All AWS Regions

↻

Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3.

Find buckets by name

< 1 > ⚙

	Name ▲	AWS Region ▼	IAM Access Analyzer	Creation date ▼
<input type="radio"/>	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)
<input type="radio"/>	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)
<input type="radio"/>	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 -

Amazon S3

▶ Account snapshot - updated every 24 hours All AWS Regions

View Storage Lens dashboard

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

General purpose buckets

Directory buckets

General purpose buckets (3) Info All AWS Regions

↻

Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3.

Find buckets by name

< 1 > ⚙

	Name ▲	AWS Region ▼	IAM Access Analyzer	Creation date ▼
<input type="radio"/>	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)
<input type="radio"/>	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)
<input type="radio"/>	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/sbp01/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/sbppl/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/sbppl/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 -

Amazon S3 > Buckets > bucket-pranav-123


bucket-pranav-123 [Info](#)

[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)


Objects (0) [Info](#)

  Copy S3 URI  Copy URL  Download  Open  Delete **Actions**  Create folder  Upload

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

< 1 > 

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
No objects You don't have any objects in this bucket.					









 Upload

AFTER -


bucket-pranav-123 [Info](#)


[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)

Objects (1) [Info](#)

  Copy S3 URI  Copy URL  Download  Open  Delete **Actions**  Create folder  Upload

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

< 1 > 

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 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\sbspol\Documents\terraform_scripts\docker\s3> terraform destroy
aws_s3_bucket.bucket: Refreshing state... [id=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:::pranav-123" -> null
  - bucket = "pranav-123" -> null
  - bucket_domain_name = "pranav-123.s3.amazonaws.com" -> null
  - bucket_regional_domain_name = "pranav-123.s3.us-east-1.amazonaws.com" -> null
  - force_destroy = false -> null
  - hosted_zone_id = "Z3AQBSTGFYJSTF" -> null
  - id = "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 {
    - rule {
      - bucket_key_enabled = false -> null
    }
  }

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

Destroy complete! Resources: 1 destroyed.
(base) PS C:\Users\sbspol\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

View Storage Lens dashboard

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

General purpose buckets

Directory buckets

General purpose buckets (2) Info All AWS Regions

↺

Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3.

Find buckets by name

< 1 > ⚙

	Name ▲	AWS Region ▼	IAM Access Analyzer	Creation date ▼
<input type="radio"/>	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)
<input type="radio"/>	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)