

Adv. Devops Experiment no. 6

Name: Rohan Lalchandani

Class: D15A Roll no.: 25

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

Theory:

Terraform is an Infrastructure as Code (IaC) tool designed to manage and provision cloud and on-premises infrastructure through code. It offers a consistent and declarative way to describe and automate infrastructure deployments across various platforms.

IaC is a methodology where infrastructure is managed and provisioned using code, rather than manual processes.

1. Core Components of Terraform:

a) Providers:

Providers are plugins that Terraform uses to interact with various infrastructure services. Each provider is responsible for understanding API interactions with a specific service or platform (e.g., AWS, Azure, Google Cloud, Docker).

b) Resources

Resources are the fundamental units of infrastructure managed by Terraform. They represent components such as virtual machines, databases, or networking elements.

c) Modules

Modules are reusable containers of Terraform configuration that are used to create multiple instances of a resource or to encapsulate complex configurations. Modules can be created and shared to standardize and simplify deployments.

d) Variables

Variables allow you to parameterize your Terraform configurations. They help in customizing configurations without modifying the main configuration files directly.

e) Outputs

Outputs are used to extract information from Terraform configurations and make it available to other configurations or external systems.

2. Terraform Workflow

Terraform follows a specific workflow to manage infrastructure:

a) Write

You define your infrastructure requirements using Terraform configuration files (.tf files) written in HCL or JSON.

b) Plan

Terraform generates an execution plan to show what actions will be taken to reach the desired state defined in your configuration files. This step helps in previewing changes before applying them.

c) Apply

Terraform applies the changes required to reach the desired state of the configuration. This step creates, updates, or deletes infrastructure resources as necessary.

d) Destroy

Terraform removes all infrastructure defined in the configuration files. This step is useful for cleaning up resources when they are no longer needed.

3. State Management

Terraform maintains a state file (terraform.tfstate) that records the current state of your infrastructure. This state file is crucial for:

- **Tracking Resources:** It maps the resources defined in the configuration files to real-world infrastructure.
- **Planning Changes:** It helps Terraform determine what changes are needed by comparing the state file to the desired configuration.
- **Concurrency Control:** It ensures that multiple users or systems do not make conflicting changes to the infrastructure.

Implementation:

1. Download and install docker from <https://www.docker.com/>

For Windows Click on AMD64 version. Check if successfully installed in cmd using
docker

docker --version

```
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\manoj>docker

Usage:  docker [OPTIONS] COMMAND

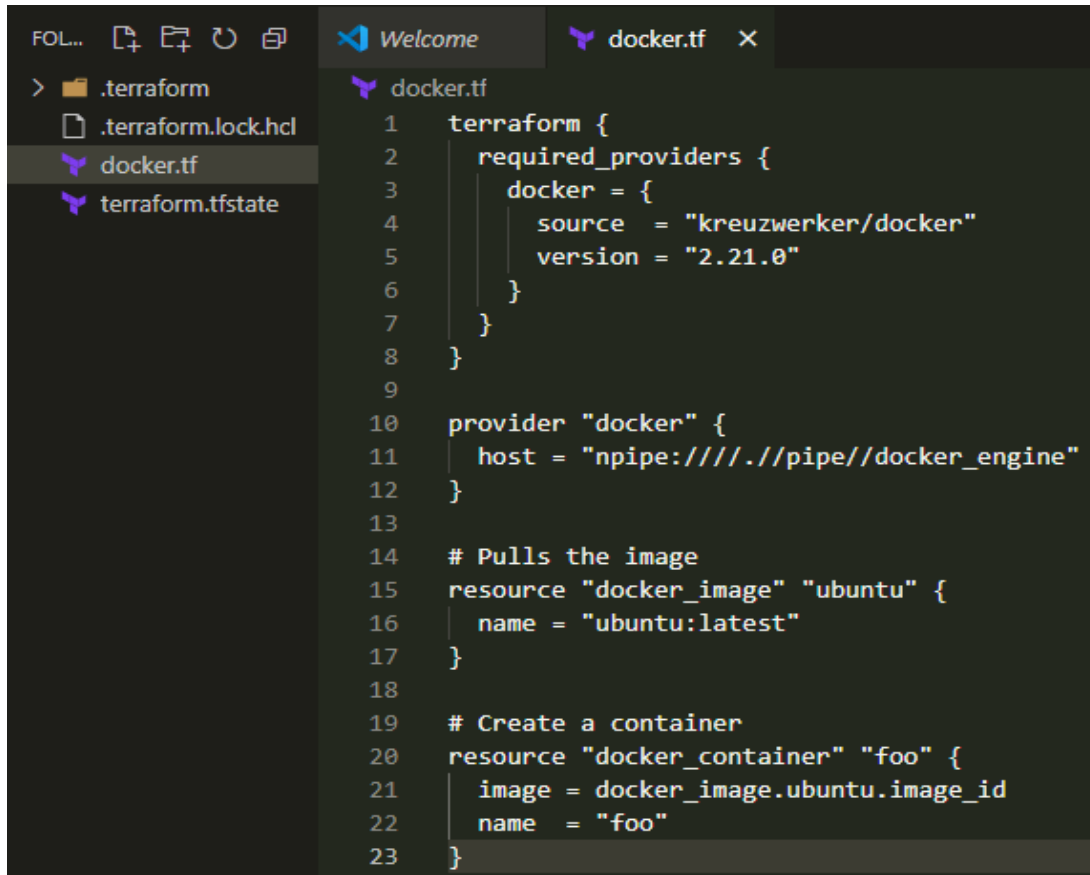
A self-sufficient runtime for containers

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

```
Management Commands:
builder  Manage builds
buildx*  Docker Buildx
compose* Docker Compose
container Manage containers
context  Manage contexts
debug*   Get a shell into any image or container
```

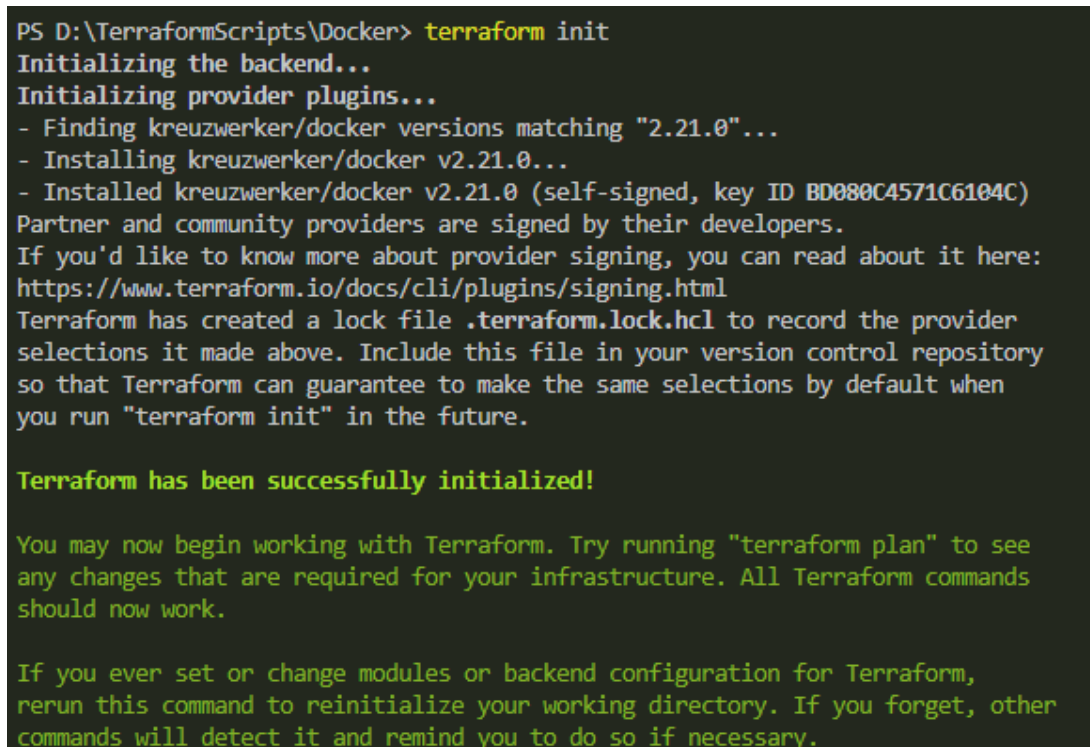
```
C:\Users\manoj>docker --version
Docker version 27.1.1, build 6312585
```

2. Create a folder TerraformScripts and in it create a folder named docker. Open it in VS code and create a new file named docker.tf and write the following code.

A screenshot of the Visual Studio Code editor interface. The left sidebar shows a file explorer with a directory structure: a folder named '.terraform' containing files '.terraform.lock.hcl', 'docker.tf', and 'terraform.tfstate'. The 'docker.tf' file is selected. The main editor area displays the content of 'docker.tf' with line numbers from 1 to 23. The code defines a Terraform configuration for a Docker provider and resources.

```
1 terraform {
2   required_providers {
3     docker = {
4       source = "kreuzwerker/docker"
5       version = "2.21.0"
6     }
7   }
8 }
9
10 provider "docker" {
11   host = "npipe://///pipe/docker_engine"
12 }
13
14 # Pulls the image
15 resource "docker_image" "ubuntu" {
16   name = "ubuntu:latest"
17 }
18
19 # Create a container
20 resource "docker_container" "foo" {
21   image = docker_image.ubuntu.image_id
22   name = "foo"
23 }
```

3. Open the terminal, make sure the path is set to the docker folder and execute
a) terraform init

A screenshot of a terminal window with a black background and green text. It shows the output of the 'terraform init' command executed in a PowerShell prompt at the path 'D:\TerraformScripts\Docker'. The output indicates that the backend is initialized, the Docker provider is installed, and the configuration is successful. It also provides instructions on how to use 'terraform plan' and 'terraform init' again if needed.

```
PS D:\TerraformScripts\Docker> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "2.21.0"...
- Installing kreuzwerker/docker v2.21.0...
- Installed kreuzwerker/docker v2.21.0 (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.
```

b) terraform plan

```
PS D:\TerraformScripts\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.foo will be created

```
+ resource "docker_container" "foo" {
  + attach          = false
  + bridge          = (known after apply)
  + command         = (known after apply)
  + container_logs  = (known after apply)
  + entrypoint      = (known after apply)
  + env            = (known after apply)
  + exit_code       = (known after apply)
  + gateway         = (known after apply)
  + hostname        = (known after apply)
  + id              = (known after apply)
  + image           = (known after apply)
  + init            = (known after apply)
  + ip_address      = (known after apply)
  + ip_prefix_length = (known after apply)
  + ipc_mode        = (known after apply)
  + log_driver      = (known after apply)
  + logs            = false
  + must_run        = true
  + name            = "foo"
  + network_data    = (known after apply)
  + read_only       = false
  + remove_volumes = true
  + restart         = "no"
  + rm              = false
  + runtime         = (known after apply)
  + security_opts   = (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
```

```
  + healthcheck (known after apply)
```

```
  + labels (known after apply)
}
```

docker_image.ubuntu will be created

```
+ resource "docker_image" "ubuntu" {
  + id          = (known after apply)
  + image_id    = (known after apply)
  + latest      = (known after apply)
  + name        = "ubuntu:latest"
  + output      = (known after apply)
  + repo_digest = (known after apply)
}
```

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

(Extra)

c) terraform validate

```
PS D:\TerraformScripts\Docker> terraform validate
Success! The configuration is valid.
```

d) terraform apply

```
PS D:\TerraformScripts\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.foo will be created
+ resource "docker_container" "foo" {
  + attach      = false
  + bridge      = (known after apply)
  + command     = [
    + "/bin/bash",
    + "-c",
    + "while true; do sleep 3600; done",
  ]
  + container_logs = (known after apply)
  + entrypoint    = (known after apply)
  + env          = (known after apply)
  + exit_code     = (known after apply)
  + gateway       = (known after apply)
  + hostname      = (known after apply)
  + id            = (known after apply)
  + image         = (known after apply)
  + init          = (known after apply)
  + ip_address    = (known after apply)
  + ip_prefix_length = (known after apply)
  + ipc_mode      = (known after apply)
  + log_driver    = (known after apply)
  + logs          = false
  + must_run      = true
  + name          = "foo"
  + network_data  = (known after apply)
```

```

+ 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

+ healthcheck (known after apply)

+ labels (known after apply)
}

# docker_image.ubuntu will be created
+ resource "docker_image" "ubuntu" {
  + id          = (known after apply)
  + image_id    = (known after apply)
  + latest      = (known after apply)
  + name        = "ubuntu:latest"
  + output      = (known after apply)
  + 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.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Still creating... [20s elapsed]
docker_image.ubuntu: Still creating... [30s elapsed]
docker_image.ubuntu: Creation complete after 37s [id=sha256:edbf74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=984860ac4d11d2b4bed665180e05f69408c3e0f24227c2b0386e77dc63568188]

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

```

4. Docker images before terraform apply command

```
PS D:\TerraformScripts\Docker> docker images
REPOSITORY    TAG                IMAGE ID           CREATED           SIZE
```

5. Docker images after terraform apply command

```
PS D:\TerraformScripts\Docker> docker images
REPOSITORY    TAG                IMAGE ID           CREATED           SIZE
ubuntu        latest            edbfe74c41f8      5 weeks ago      78.1MB
```

6. terraform destroy

```
PS D:\TerraformScripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=984860ac4d11d2b4bed665180e05f69408c3e0f24227c2b0386e77dc63568188]

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.foo will be destroyed
- resource "docker_container" "foo" {
  - attach          = false -> null
  - command         = [
    - "/bin/bash",
    - "-c",
    - "while true; do sleep 3600; done",
  ] -> null
  - cpu_shares      = 0 -> null
  - dns             = [] -> null
  - dns_opts        = [] -> null
  - dns_search      = [] -> null
  - entrypoint      = [] -> null
  - env             = [] -> null
  - gateway         = "172.17.0.1" -> null
  - group_add       = [] -> null
  - hostname        = "984860ac4d11" -> null
  - id              = "984860ac4d11d2b4bed665180e05f69408c3e0f24227c2b0386e77dc63568188" -> null
  - image           = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
  - init            = false -> null
  - ip_address      = "172.17.0.2" -> null
  - ip_prefix_length = 16 -> null
  - ipc_mode        = "private" -> null
  - links           = [] -> null
  - log_driver      = "json-file" -> null
  - log_opts        = {} -> null
  - logs            = false -> null
```



```

- log_opts          = {} -> null
- logs              = false -> null
- max_retry_count   = 0 -> null
- memory            = 0 -> null
- memory_swap       = 0 -> null
- must_run          = true -> null
- name              = "foo" -> null
- network_data      = [
  - {
    - gateway          = "172.17.0.1"
    - global_ipv6_prefix_length = 0
    - ip_address       = "172.17.0.2"
    - ip_prefix_length = 16
    - network_name     = "bridge"
    # (2 unchanged attributes hidden)
  },
] -> null
- network_mode      = "bridge" -> null
- privileged         = false -> null
- publish_all_ports  = false -> null
- read_only          = false -> null
- remove_volumes     = true -> null
- restart            = "no" -> null
- rm                 = false -> null
- runtime            = "runc" -> null
- security_opts      = [] -> null
- shm_size           = 64 -> null
- start              = true -> null
- stdin_open         = false -> null
- stop_timeout       = 0 -> null
- storage_opts       = {} -> null
- sysctls            = {} -> null
- tmpfs              = {} -> null
- tty                = false -> null
# (8 unchanged attributes hidden)
}

```

```

# docker_image.ubuntu will be destroyed
- resource "docker_image" "ubuntu" {
  - id          = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest" -> null
  - image_id    = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
  - latest      = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
  - name        = "ubuntu:latest" -> null
  - repo_digest = "ubuntu@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee" -> 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.foo: Destroying... [id=984860ac4d11d2b4bed665180e05f69408c3e0f24227c2b0386e77dc63568188]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

```

Destroy complete! Resources: 2 destroyed.

7. Docker images after terraform destroy

Destroy complete! Resources: 2 destroyed.

PS D:\TerraformScripts\Docker> docker images

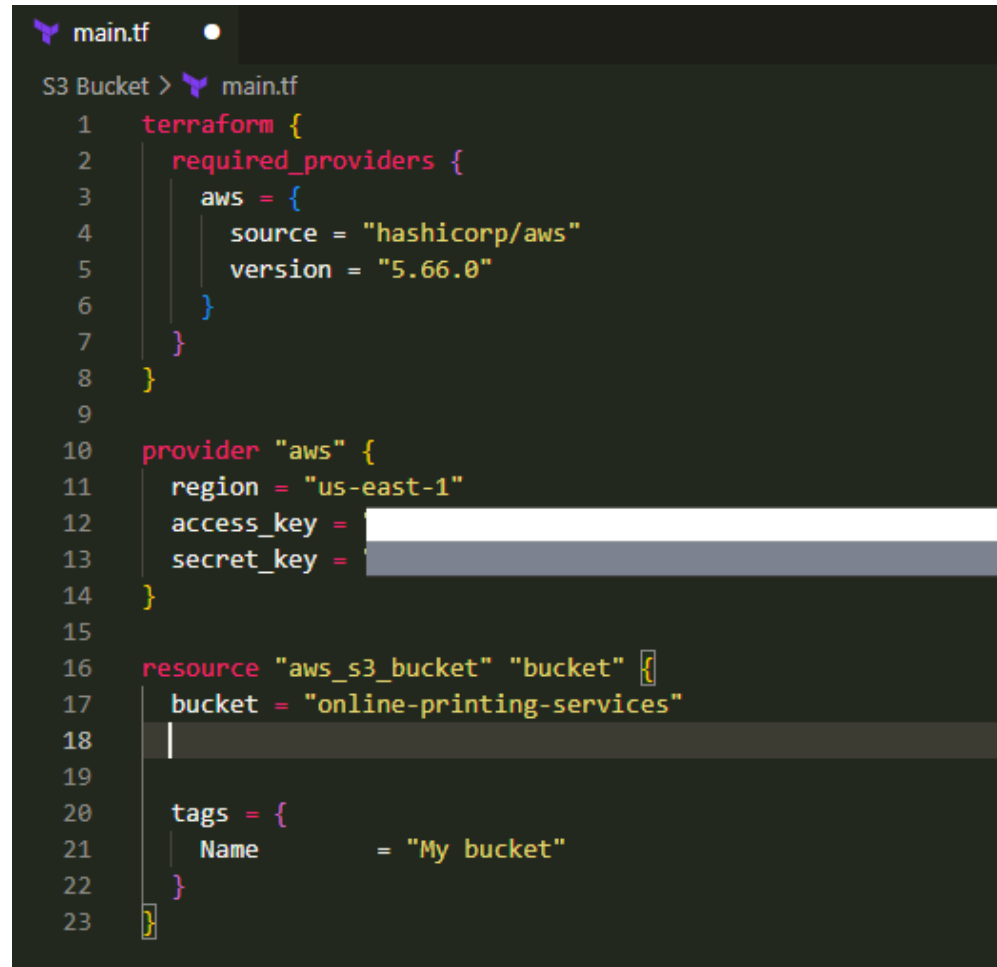
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
------------	-----	----------	---------	------

S3 bucket using terraform:

1. Create another folder “**S3 Bucket**” in “**TerraformScripts**” folder open it in VS code and create a “main.tf file” and write the following code, taken from terraform documentation.

a) For access_key and secret_key, Go to your “AWS Management Console” and go to security credentials option by click on you profile name at top right corner of console.

b) Click on generate access key, copy and paste the key in the code.



```
main.tf
S3 Bucket > main.tf
1  terraform {
2      required_providers {
3          aws = {
4              source = "hashicorp/aws"
5              version = "5.66.0"
6          }
7      }
8  }
9
10 provider "aws" {
11     region = "us-east-1"
12     access_key = 
13     secret_key = 
14 }
15
16 resource "aws_s3_bucket" "bucket" {
17     bucket = "online-printing-services"
18
19
20     tags = {
21         Name = "My bucket"
22     }
23 }
```

2) Open terminal, move to the “S3 Bucket” folder and execute terraform init.

```
PS D:\TerraformScripts> cd '..\S3 Bucket\'
PS D:\TerraformScripts\S3 Bucket> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.66.0"...
- Installing hashicorp/aws v5.66.0...
- Installed hashicorp/aws v5.66.0 (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.
```

2. Execute terraform validate to check if your code is correct or not.

```
PS D:\TerraformScripts\S3 Bucket> terraform validate
Success! The configuration is valid.
```

3. terraform plan

```
PS D:\TerraformScripts\S3 Bucket> 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.bucket will be created
+ resource "aws_s3_bucket" "bucket" {
  + acceleration_status      = (known after apply)
  + acl                      = (known after apply)
  + arn                     = (known after apply)
  + bucket                  = "OnlinePrintingServices"
  + 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)
  + website_endpoint        = (known after apply)

  + cors_rule (known after apply)

  + grant (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)

+ replication_configuration (known after apply)

+ replication_configuration (known after apply)

+ replication_configuration (known after apply)

+ replication_configuration (known after apply)

+ replication_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.

4. terraform apply

```
PS D:\TerraformScripts\S3 Bucket> terraform apply
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbol:
+ 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                  = "online-printing-services"
  + 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)
  + website_endpoint        = (known after apply)

  + cors_rule (known after apply)

  + grant (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

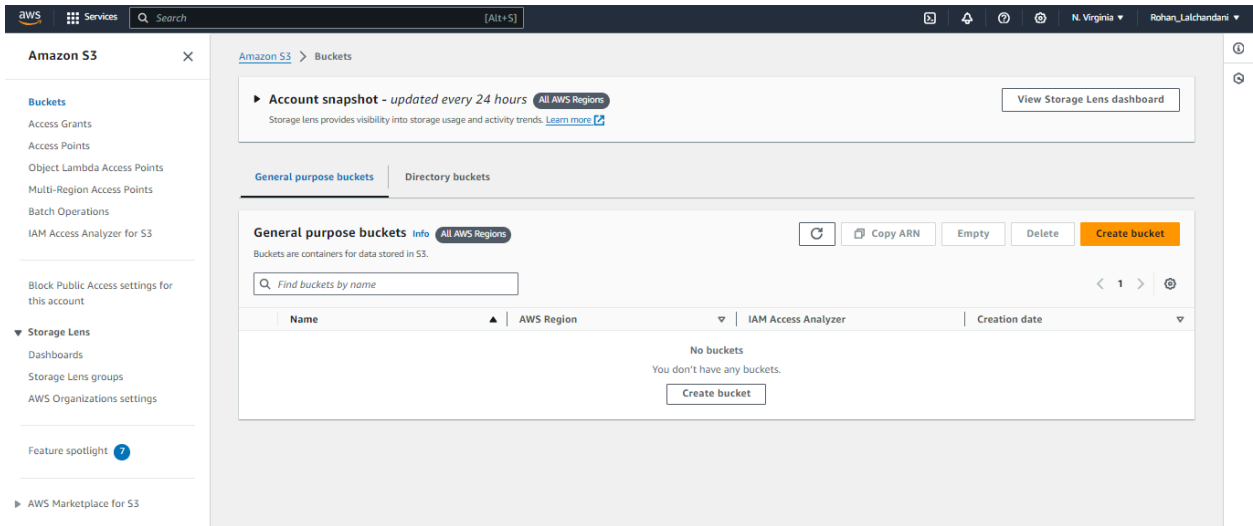
aws_s3_bucket.bucket: Creating...

aws_s3_bucket.bucket: Creation complete after 7s [id=online-printing-services]

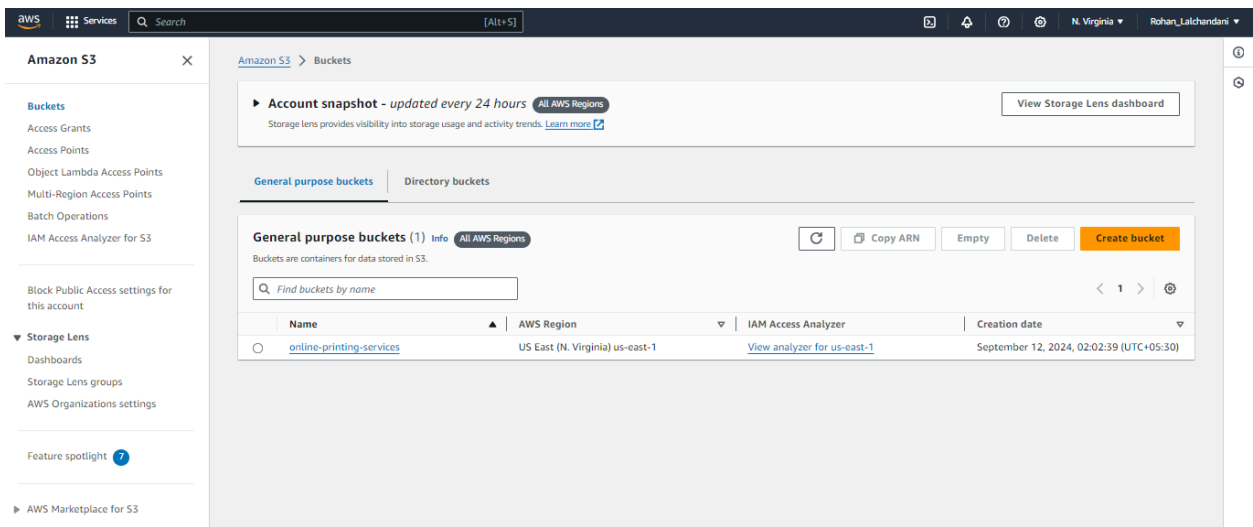
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

```
PS D:\TerraformScripts\S3 Bucket> 
```

5. Buckets in console before terraform apply.



6. Buckets in console after terraform apply.



(Extra)

Hosting a website using s3 bucket in terraform

1. Create a new folder "S3 Website" under "TerraformScripts".

Create four files "main.tf", "providers.tf", "variables.tf", "output.tf" and write the following code.

main.tf :

This Terraform code creates and configures an AWS S3 bucket for hosting static website files with public access.

1. **S3 Bucket:** Creates an S3 bucket (`aws_s3_bucket.demo-bucket`) using a variable for the name.
2. **Ownership Controls:** Configures ownership of the objects in the bucket, making the bucket owner the preferred owner (`BucketOwnerPreferred`).
3. **Public Access Block:** Disables blocking of public access (`aws_s3_bucket_public_access_block`), allowing public access settings.
4. **Bucket ACL:** Sets the bucket's access control list (ACL) to allow public read access (`public-read`).
5. **Bucket Policy:** Adds a policy to the bucket allowing public read access (`s3:GetObject`) to all objects in the bucket.
6. **Template Files Module:** Fetches local files from a directory for hosting using the `hashicorp/dir/template` module.
7. **Website Configuration:** Configures the bucket to host a static website, specifying `index.html` as the main document.
8. **S3 Object:** Uploads files to the S3 bucket for website hosting, with each object having its key (file name) and content.

FOL...

> .terraform

▼ webfiles

index.html

main.tf

output.tf

providers.tf

variables.tf

main.tf

providers.tf

variables.tf

output.tf

main.tf

1 resource "aws_s3_bucket" "demo-bucket" {

2 | bucket = var.my_bucket_name # Name of the S3 bucket

3 }

4

5

6 resource "aws_s3_bucket_ownership_controls" "example" {

7 | bucket = aws_s3_bucket.demo-bucket.id

8 | rule {

9 | | object_ownership = "BucketOwnerPreferred"

10 | }

11 }

12

13

14 resource "aws_s3_bucket_public_access_block" "example" {

15 | bucket = aws_s3_bucket.demo-bucket.id

16

17 | block_public_acls = false

18 | block_public_policy = false

19 | ignore_public_acls = false

20 | restrict_public_buckets = false

21 }

22

23

24 # AWS S3 bucket ACL resource

25 resource "aws_s3_bucket_acl" "example" {

26 | depends_on = [

27 | | aws_s3_bucket_ownership_controls.example,

28 | | aws_s3_bucket_public_access_block.example,

29 |]

30

31 | bucket = aws_s3_bucket.demo-bucket.id

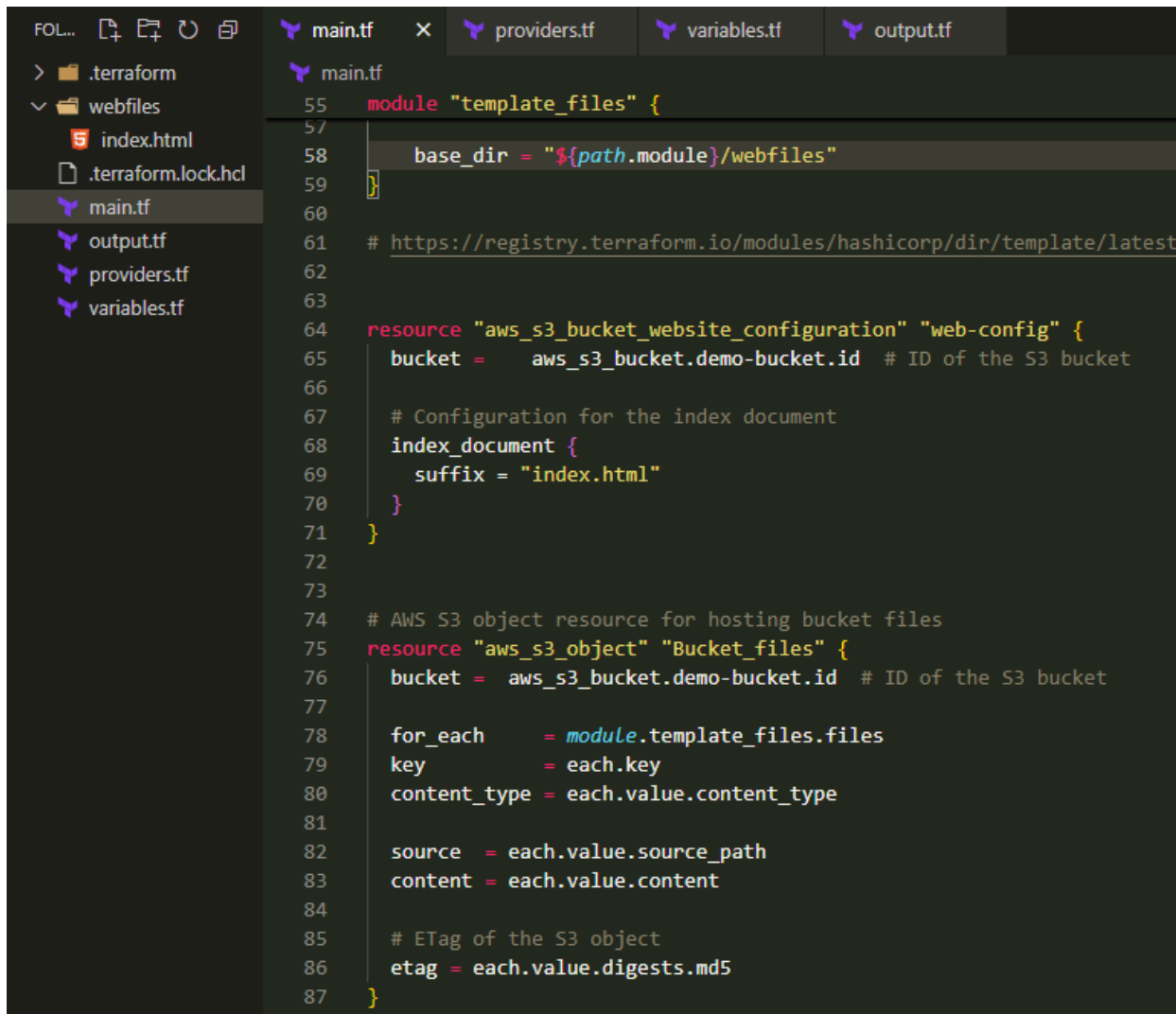
32 | acl = "public-read"

FOL...

> .terraform
v webfiles
 index.html
 .terraform.lock.hcl
 main.tf
 output.tf
 providers.tf
 variables.tf

main.tf

25 resource "aws_s3_bucket_acl" "example" {
30
31 bucket = aws_s3_bucket.demo-bucket.id
32 acl = "public-read"
33 }
34
35
36
37 resource "aws_s3_bucket_policy" "host_bucket_policy" {
38 bucket = aws_s3_bucket.demo-bucket.id # ID of the S3 bucket
39
40 # Policy JSON for allowing public read access
41 policy = jsonencode({
42 "Version" : "2012-10-17",
43 "Statement" : [
44 {
45 "Effect" : "Allow",
46 "Principal" : "*",
47 "Action" : "s3:GetObject",
48 "Resource" : "arn:aws:s3:::\${var.my_bucket_name}/*"
49 }
50]
51 })
52 }
53
54
55 module "template_files" {
56 source = "hashicorp/dir/template"
57
58 base_dir = "\${path.module}/webfiles"
59 }
60



```
55 module "template_files" {
56
57     base_dir = "${path.module}/webfiles"
58 }
59
60
61 # https://registry.terraform.io/modules/hashicorp/dir/template/latest
62
63
64 resource "aws_s3_bucket_website_configuration" "web-config" {
65     bucket = aws_s3_bucket.demo-bucket.id # ID of the S3 bucket
66
67     # Configuration for the index document
68     index_document {
69         suffix = "index.html"
70     }
71 }
72
73
74 # AWS S3 object resource for hosting bucket files
75 resource "aws_s3_object" "Bucket_files" {
76     bucket = aws_s3_bucket.demo-bucket.id # ID of the S3 bucket
77
78     for_each = module.template_files.files
79     key      = each.key
80     content_type = each.value.content_type
81
82     source = each.value.source_path
83     content = each.value.content
84
85     # ETag of the S3 object
86     etag = each.value.digests.md5
87 }
```

providers.tf :

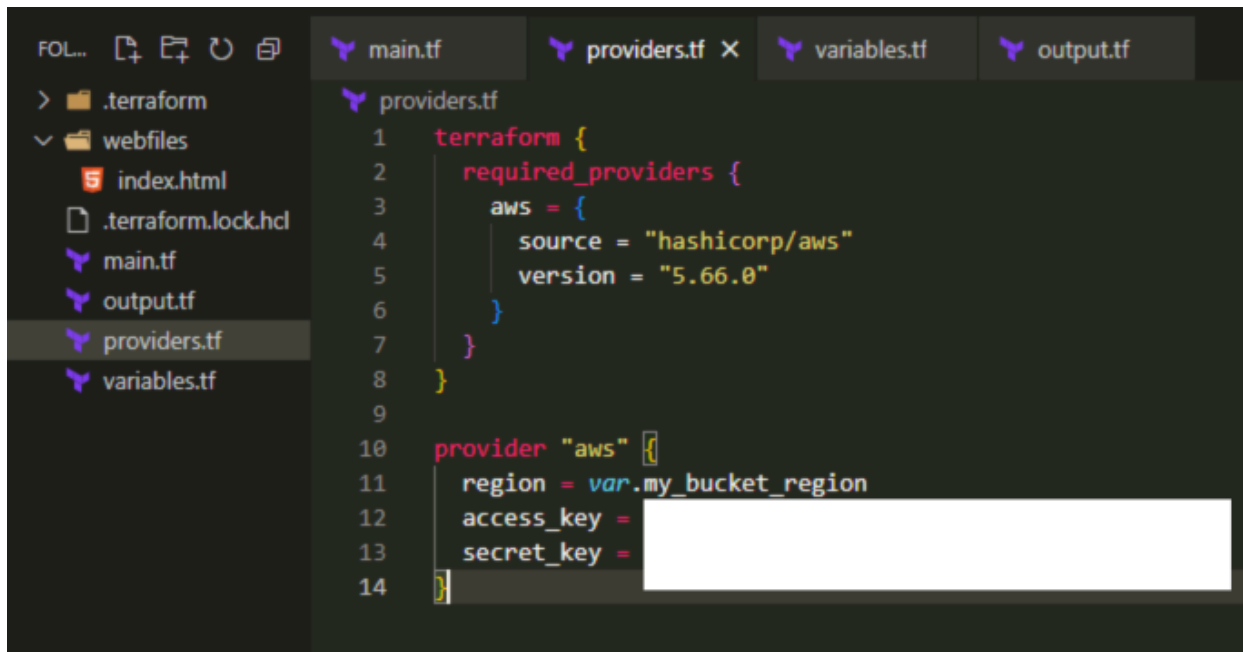
This Terraform snippet is defining the AWS provider and specifying its version for the configuration. Here's what it does:

1. Terraform Block:

- Specifies that the AWS provider ([hashicorp/aws](#)) version 5.66.0 is required.

2. Provider Configuration:

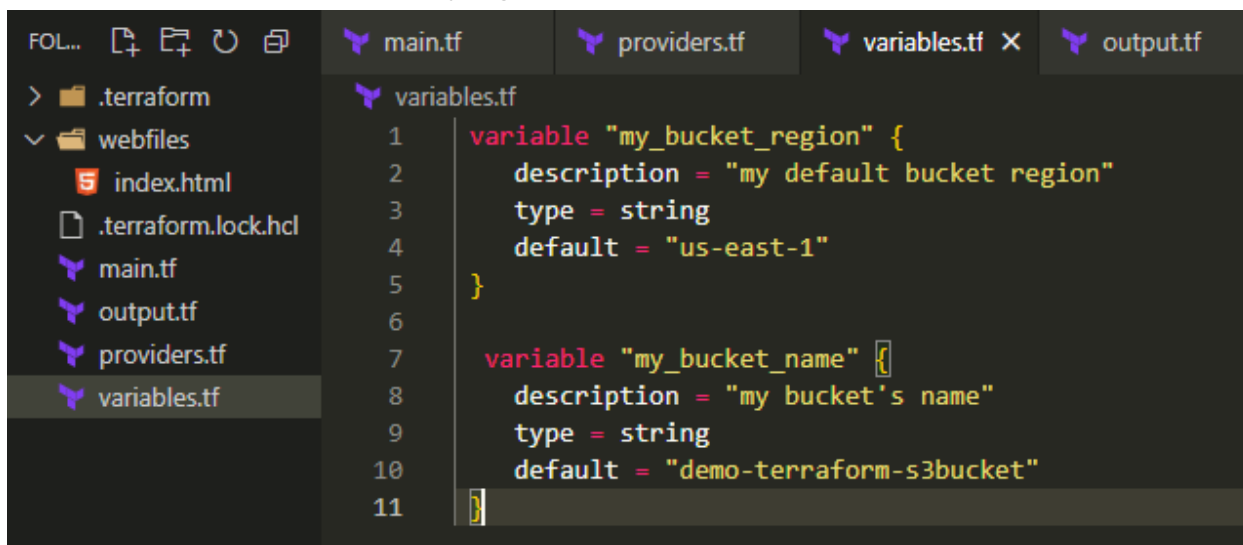
- Configures the AWS provider with the region set by the [my_bucket_region](#) variable.
- The [access_key](#) and [secret_key](#) fields are placeholders for AWS credentials, which are needed for authentication when interacting with AWS services.



```
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "5.66.0"
6     }
7   }
8 }
9
10 provider "aws" {
11   region = var.my_bucket_region
12   access_key = 
13   secret_key = 
14 }
```

variables.tf :

These variables allow flexible configuration, so you can easily change the region and bucket name without modifying the main code.



```
1 variable "my_bucket_region" {
2   description = "my default bucket region"
3   type = string
4   default = "us-east-1"
5 }
6
7 variable "my_bucket_name" {
8   description = "my bucket's name"
9   type = string
10  default = "demo-terraform-s3bucket"
11 }
```

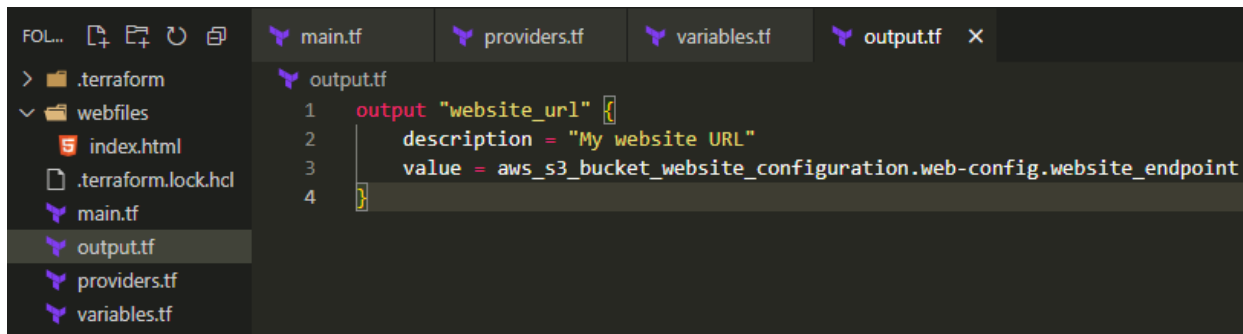
output.tf :

This **output block** in Terraform is used to display the **URL of the static website** hosted on the S3 bucket after the configuration is applied.

- **output "website_url"**: Defines an output variable named `website_url` that will be shown after the Terraform execution.
- **description**: Describes the purpose of this output, which is to display the website's URL.

- **value:** Uses the `website_endpoint` attribute from the `aws_s3_bucket_website_configuration` resource (referenced as `web-config`). This contains the URL of the website hosted in the S3 bucket.

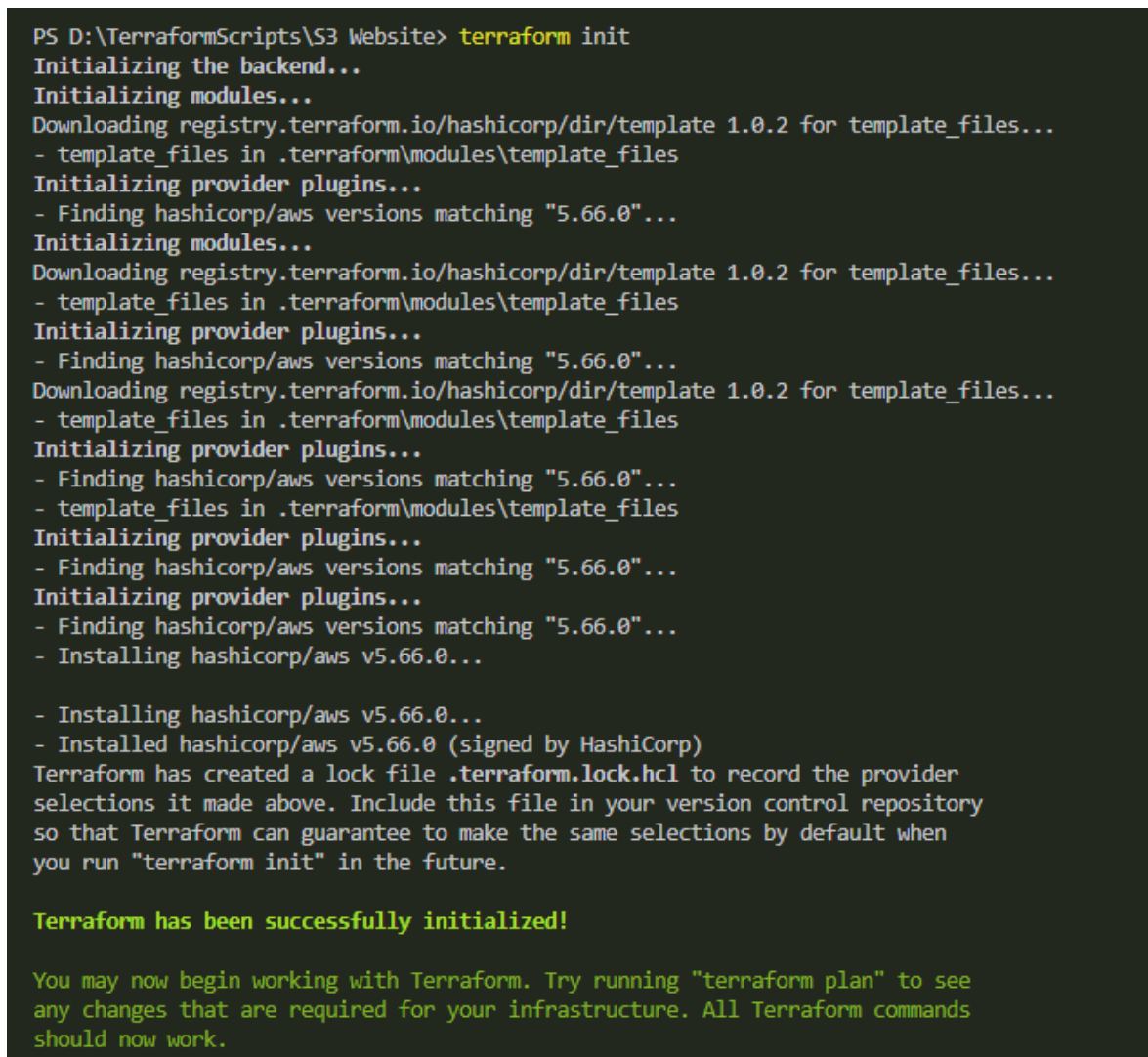
Once Terraform completes, it will print the website's URL for easy access.



```
main.tf providers.tf variables.tf output.tf X
> .terraform
  webfiles
    index.html
  .terraform.lock.hcl
  main.tf
  output.tf
  providers.tf
  variables.tf

output.tf
1  output "website_url" {
2      description = "My website URL"
3      value = aws_s3_bucket_website_configuration.web-config.website_endpoint
4  }
```

2. Open terminal and run terraform init.



```
PS D:\TerraformScripts\S3 Website> terraform init
Initializing the backend...
Initializing modules...
Downloading registry.terraform.io/hashicorp/dir/template 1.0.2 for template_files...
- template_files in .terraform\modules\template_files
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.66.0"...
Initializing modules...
Downloading registry.terraform.io/hashicorp/dir/template 1.0.2 for template_files...
- template_files in .terraform\modules\template_files
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.66.0"...
Downloading registry.terraform.io/hashicorp/dir/template 1.0.2 for template_files...
- template_files in .terraform\modules\template_files
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.66.0"...
- template_files in .terraform\modules\template_files
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.66.0"...
- Installing hashicorp/aws v5.66.0...

- Installing hashicorp/aws v5.66.0...
- Installed hashicorp/aws v5.66.0 (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.
```

3. Execute terraform plan.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE powershell + v

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.
PS D:\TerraformScripts\S3 Website> 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.demo-bucket will be created
+ resource "aws_s3_bucket" "demo-bucket" {
  + acceleration_status = (known after apply)
  + acl                  = (known after apply)
  + arn                  = (known after apply)
  + bucket               = "demo-terraform-s3bucket"
  + 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_all            = (known after apply)
  + website_domain      = (known after apply)
  + website_endpoint    = (known after apply)

  + cors_rule (known after apply)

  + grant (known after apply)

  + lifecycle_rule (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)
}

# aws_s3_bucket_acl.example will be created
+ resource "aws_s3_bucket_acl" "example" {
  + acl      = "public-read"
  + bucket   = (known after apply)
  + id       = (known after apply)

  + access_control_policy (known after apply)
}

# aws_s3_bucket_ownership_controls.example will be created
+ resource "aws_s3_bucket_ownership_controls" "example" {
  + bucket = (known after apply)
  + id     = (known after apply)

  + rule {
    + object_ownership = "BucketOwnerPreferred"
  }
}

# aws_s3_bucket_policy.host_bucket_policy will be created
+ resource "aws_s3_bucket_policy" "host_bucket_policy" {
```

```
+ resource "aws_s3_bucket_policy" "host_bucket_policy" {
  + bucket = (known after apply)
  + id      = (known after apply)
  + policy = jsonencode(
    {
      + Statement = [
        + {
          + Action      = "s3:GetObject"
          + Effect       = "Allow"
          + Principal    = "*"
          + Resource     = "arn:aws:s3:::demo-terraform-s3bucket/*"
        },
      ]
      + Version      = "2012-10-17"
    }
  )
}
```

aws_s3_bucket_public_access_block.example will be created

```
+ resource "aws_s3_bucket_public_access_block" "example" {
  + block_public_acls      = false
  + block_public_policy    = false
  + bucket                 = (known after apply)
  + id                     = (known after apply)
  + ignore_public_acls     = false
  + restrict_public_buckets = false
}
```

aws_s3_bucket_website_configuration.web-config will be created

```
+ resource "aws_s3_bucket_website_configuration" "web-config" {
  + bucket           = (known after apply)
  + id               = (known after apply)
  + routing_rules     = (known after apply)
  + website_domain    = (known after apply)
  + website_endpoint  = (known after apply)
}
```



```
+ website_endpoint = (known after apply)

+ index_document {
  + suffix = "index.html"
}

+ routing_rule (known after apply)
}

# aws_s3_object.Bucket_files["index.html"] will be created
+ resource "aws_s3_object" "Bucket_files" {
  + acl                = (known after apply)
  + arn                = (known after apply)
  + bucket             = (known after apply)
  + bucket_key_enabled = (known after apply)
  + checksum_crc32     = (known after apply)
  + checksum_crc32c    = (known after apply)
  + checksum_sha1      = (known after apply)
  + checksum_sha256    = (known after apply)
  + content_type       = "text/html; charset=utf-8"
  + etag               = "706be2e258c7c90ddaa6d23b17cefa4a"
  + force_destroy      = false
  + id                 = (known after apply)
  + key                = "index.html"
  + kms_key_id         = (known after apply)
  + server_side_encryption = (known after apply)
  + source              = "../webfiles/index.html"
  + storage_class       = (known after apply)
  + tags_all           = (known after apply)
  + version_id         = (known after apply)
}
```

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

Changes to Outputs:

```
+ website_url = (known after apply)
```

4. Execute terraform apply

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE  [powershell] + - [ ] [ ] ...

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.
PS D:\TerraformScripts\S3 Website> 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.demo-bucket will be created
+ resource "aws_s3_bucket" "demo-bucket" {
  + acceleration_status = (known after apply)
  + acl                 = (known after apply)
  + arn                 = (known after apply)
  + bucket              = "demo-terraform-s3bucket"
  + 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_all            = (known after apply)
  + 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)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

  + 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)
}

# aws_s3_bucket_acl.example will be created
+ resource "aws_s3_bucket_acl" "example" {
  + acl    = "public-read"
  + bucket = (known after apply)
  + id     = (known after apply)

  + access_control_policy (known after apply)
}

# aws_s3_bucket_ownership_controls.example will be created
+ resource "aws_s3_bucket_ownership_controls" "example" {
  + bucket = (known after apply)
  + id     = (known after apply)

  + rule {
    + object_ownership = "BucketOwnerPreferred"
  }
}

# aws_s3_bucket_policy.host_bucket_policy will be created
+ resource "aws_s3_bucket_policy" "host_bucket_policy" {
  + bucket = (known after apply)
  + id     = (known after apply)
```

```
+ id      = (known after apply)
+ policy = jsonencode(
  {
    + Statement = [
      + {
        + Action    = "s3:GetObject"
        + Effect    = "Allow"
        + Principal = "*"
        + Resource  = "arn:aws:s3:::demo-terraform-s3bucket/*"
      },
    ]
    + Version = "2012-10-17"
  }
)
```

aws_s3_bucket_public_access_block.example will be created

```
+ resource "aws_s3_bucket_public_access_block" "example" {
  + block_public_acls      = false
  + block_public_policy    = false
  + bucket                 = (known after apply)
  + id                    = (known after apply)
  + ignore_public_acls    = false
  + restrict_public_buckets = false
}
```

aws_s3_bucket_website_configuration.web-config will be created

```
+ resource "aws_s3_bucket_website_configuration" "web-config" {
  + bucket           = (known after apply)
  + id              = (known after apply)
  + routing_rules    = (known after apply)
  + website_domain   = (known after apply)
  + website_endpoint = (known after apply)

  + index_document {
    + suffix = "index.html"
  }
}
```

```
    + suffix = "index.html"
  }

  + routing_rule (known after apply)
}

# aws_s3_object.Bucket_files["index.html"] will be created
+ resource "aws_s3_object" "Bucket_files" {
  + acl                  = (known after apply)
  + arn                  = (known after apply)
  + bucket                = (known after apply)
  + bucket_key_enabled   = (known after apply)
  + checksum_crc32        = (known after apply)
  + checksum_crc32c      = (known after apply)
  + checksum_sha1         = (known after apply)
  + checksum_sha256       = (known after apply)
  + content_type          = "text/html; charset=utf-8"
  + etag                  = "706be2e258c7c90ddaa6d23b17cefa4a"
  + force_destroy         = false
  + id                    = (known after apply)
  + key                   = "index.html"
  + kms_key_id            = (known after apply)
  + server_side_encryption = (known after apply)
  + source                = "../webfiles/index.html"
  + storage_class         = (known after apply)
  + tags_all              = (known after apply)
  + version_id            = (known after apply)
}
```

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

Changes to Outputs:

```
+ website_url = (known after apply)
```

Do you want to perform these actions?

Terraform will perform the actions described above.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_s3_bucket.demo-bucket: Creating...
aws_s3_bucket.demo-bucket: Creation complete after 6s [id=demo-terraform-s3bucket]
aws_s3_bucket_ownership_controls.example: Creating...
aws_s3_object.Bucket_files["index.html"]: Creating...
aws_s3_bucket_policy.host_bucket_policy: Creating...
aws_s3_bucket_public_access_block.example: Creating...
aws_s3_bucket_website_configuration.web-config: Creating...
aws_s3_bucket_ownership_controls.example: Creation complete after 1s [id=demo-terraform-s3bucket]
aws_s3_bucket_public_access_block.example: Creation complete after 1s [id=demo-terraform-s3bucket]
aws_s3_object.Bucket_files["index.html"]: Creation complete after 1s [id=index.html]
aws_s3_bucket_acl.example: Creating...
aws_s3_bucket_policy.host_bucket_policy: Creation complete after 1s [id=demo-terraform-s3bucket]
aws_s3_bucket_website_configuration.web-config: Creation complete after 2s [id=demo-terraform-s3bucket]
aws_s3_bucket_acl.example: Creation complete after 1s [id=demo-terraform-s3bucket,public-read]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

website_url = "demo-terraform-s3bucket.s3-website-us-east-1.amazonaws.com"
```

5. The “curl demo-terraform-s3bucket.s3-website-us-east-1.amazonaws.com” command is used to make an HTTP request to the specified URL, which corresponds to the static website hosted on an AWS S3 bucket. Here's what it does:

- **curl**: A command-line tool to transfer data from or to a server, commonly used to fetch web pages or APIs.

It fetches the content of the static website (likely an HTML page like `index.html`) stored in the S3 bucket and displays the response in the terminal.

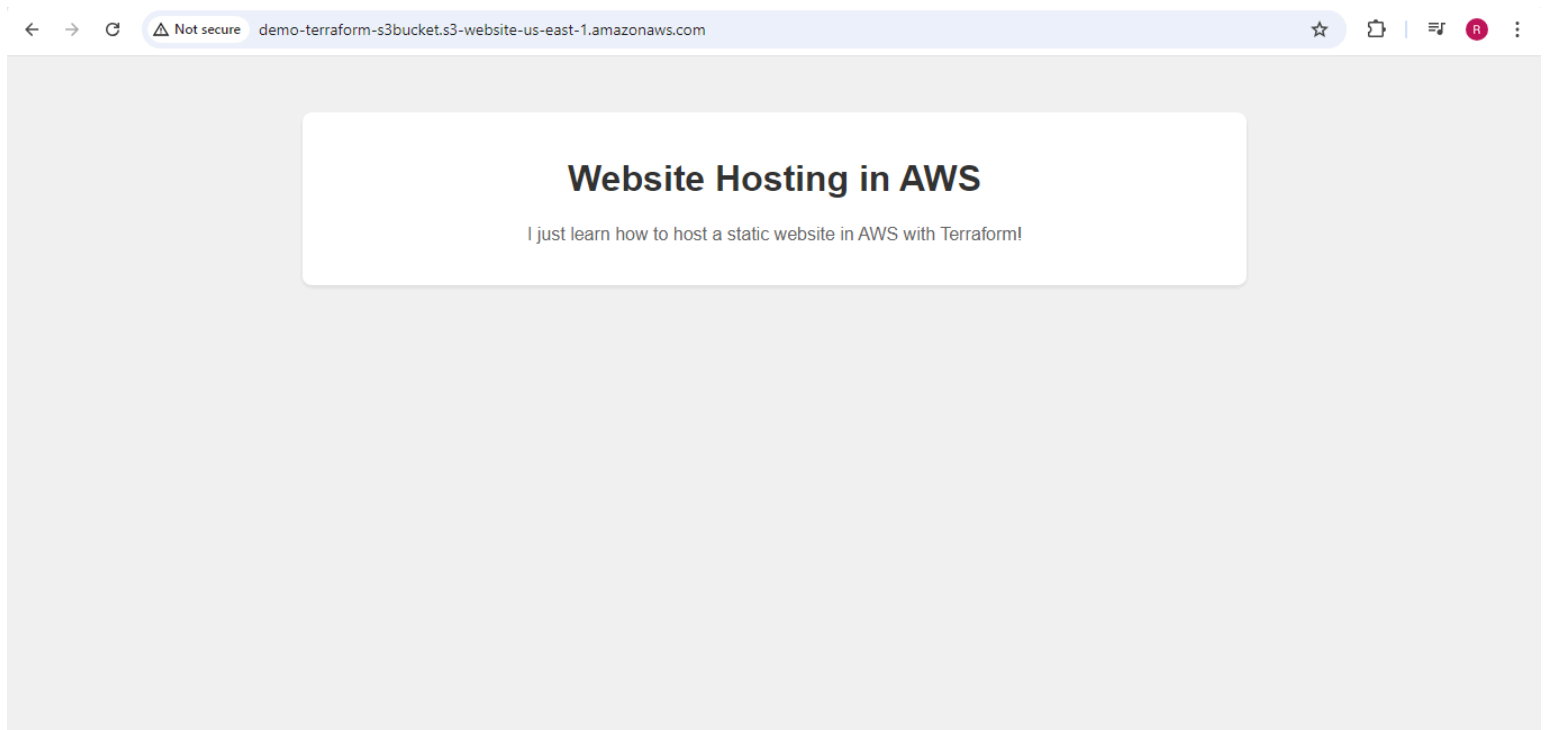
```

PS D:\TerraformScripts\S3 Website> curl demo-terraform-s3bucket.s3-website-us-east-1.amazonaws.com

StatusCode      : 200
StatusDescription : OK
Content         : <!DOCTYPE html>
                  <html lang="en">
                  <head>
                    <meta charset="UTF-8">
                    <meta name="viewport" content="width=device-width, initial-scale=1.0">
                    <title>Website Hosting in AWS</title>
                    <style...
RawContent      : HTTP/1.1 200 OK
                  x-amz-id-2: gqFshOXNgxgx004fpo0wFbu0gY++NLD0QFMz+hIsKG24/jwo64P6K8SKFTEpXlgTJhoYk4xm89Q=
                  x-amz-request-id: KYT3KHRDSSYFT5ZE
                  Content-Length: 990
                  Content-Type: text/html; charset=utf...
Forms           : {}
Headers         : {[x-amz-id-2, gqFshOXNgxgx004fpo0wFbu0gY++NLD0QFMz+hIsKG24/jwo64P6K8SKFTEpXlgTJhoYk4xm89Q=], [x-amz-request-id, KYT3KHRDSSYFT5ZE],
                  [Content-Length, 990], [Content-Type, text/html; charset=utf-8]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 990

```

6. Go to the browser and enter the url to see the hosted webpage.



7. Use “terraform destroy” to delete the bucket.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE powershell +
PS D:\TerraformScripts\S3 Website> terraform destroy
aws_s3_bucket.demo-bucket: Refreshing state... [id=demo-terraform-s3bucket]
aws_s3_bucket_policy.host_bucket_policy: Refreshing state... [id=demo-terraform-s3bucket]
aws_s3_bucket_website_configuration.web-config: Refreshing state... [id=demo-terraform-s3bucket]
aws_s3_bucket_public_access_block.example: Refreshing state... [id=demo-terraform-s3bucket]
aws_s3_bucket_ownership_controls.example: Refreshing state... [id=demo-terraform-s3bucket]
aws_s3_object.Bucket_files["index.html"]: Refreshing state... [id=index.html]
aws_s3_bucket_acl.example: Refreshing state... [id=demo-terraform-s3bucket,public-read]

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.demo-bucket will be destroyed
- resource "aws_s3_bucket" "demo-bucket" {
  - arn = "arn:aws:s3::demo-terraform-s3bucket" -> null
  - bucket = "demo-terraform-s3bucket" -> null
  - bucket_domain_name = "demo-terraform-s3bucket.s3.amazonaws.com" -> null
  - bucket_regional_domain_name = "demo-terraform-s3bucket.s3.us-east-1.amazonaws.com" -> null
  - force_destroy = false -> null
  - hosted_zone_id = "Z3AQBSTGFYJSTF" -> null
  - id = "demo-terraform-s3bucket" -> null
  - object_lock_enabled = false -> null
  - policy = jsonencode(
    {
      - Statement = [
        - {
          - Action = "s3:GetObject"
```

```
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_website_configuration.web-config: Destroying... [id=demo-terraform-s3bucket]
aws_s3_object.Bucket_files["index.html"]: Destroying... [id=index.html]
aws_s3_bucket_policy.host_bucket_policy: Destroying... [id=demo-terraform-s3bucket]
aws_s3_bucket_acl.example: Destroying... [id=demo-terraform-s3bucket,public-read]
aws_s3_bucket_acl.example: Destruction complete after 0s
aws_s3_bucket_ownership_controls.example: Destroying... [id=demo-terraform-s3bucket]
aws_s3_bucket_public_access_block.example: Destroying... [id=demo-terraform-s3bucket]
aws_s3_object.Bucket_files["index.html"]: Destruction complete after 1s
aws_s3_bucket_ownership_controls.example: Destruction complete after 1s
aws_s3_bucket_public_access_block.example: Destruction complete after 2s
aws_s3_bucket_website_configuration.web-config: Destruction complete after 2s
aws_s3_bucket_policy.host_bucket_policy: Destruction complete after 2s
aws_s3_bucket.demo-bucket: Destroying... [id=demo-terraform-s3bucket]
aws_s3_bucket.demo-bucket: Destruction complete after 1s
```

```
Destroy complete! Resources: 7 destroyed.
```

```
PS D:\TerraformScripts\S3 Website>
```