

<https://linkedin.com/in/prafulpatel16><https://github.com/prafulpatel16><https://medium.com/@prafulpatel16><https://www.youtube.com/@prafulpatel16>

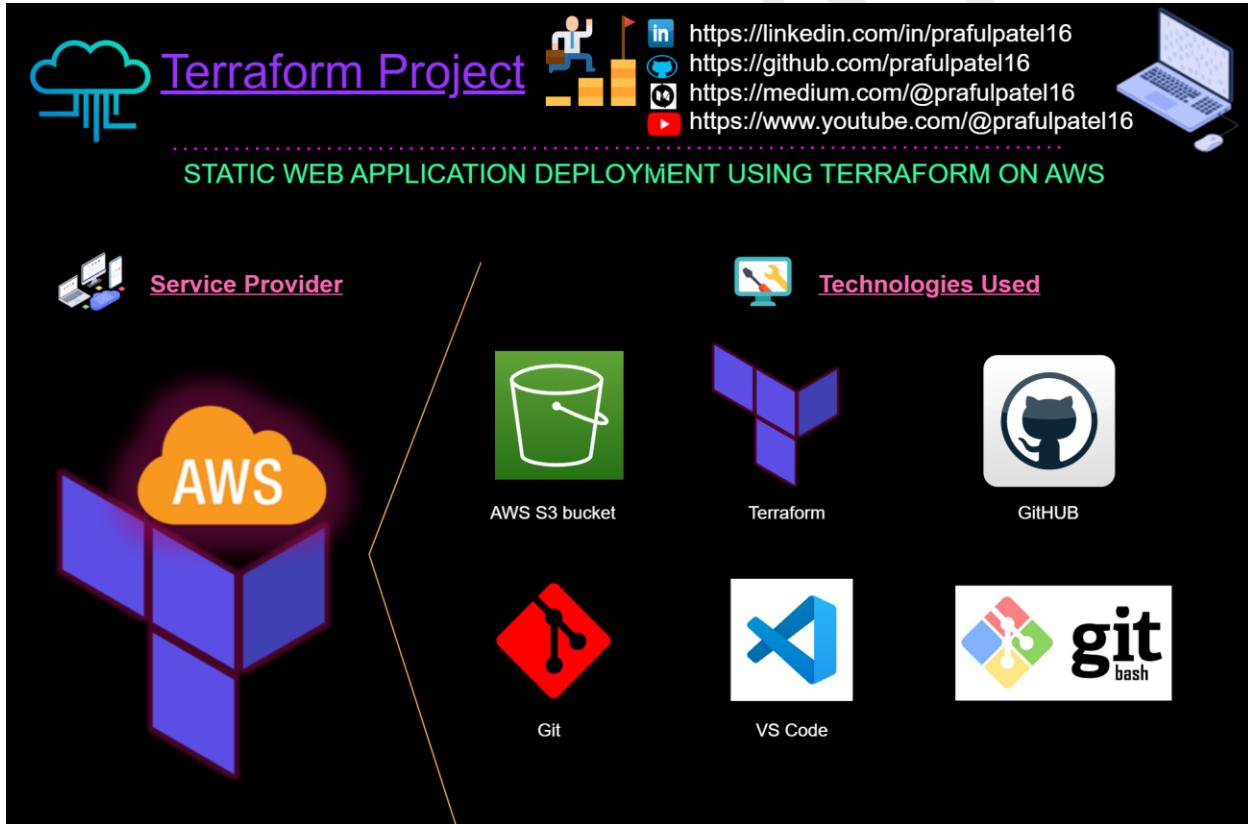
PROJECT



TERRAFORM-AWS: STATIC WEB APPLICATION DEPLOYMENT ON AWS S3 AUTOMATED WAY USING TERRAFORM (INFRASTRUCTURE AS CODE)



By: PRAFUL PATEL



➤ **Project**

IT services Provider Company **Prefect.cloud** is engaged into providing software development solutions. Currently, they are working and finding some solution to automate and provisioning the cloud services automated way. The company has requirement to deploy a **static web application on AWS S3** an automated way.

The management has also decided to leverage an automated tool which can scale up in the future on multi-cloud without any dependencies and vendor locking rules.

An infrastructure team has started figuring out the cloud agnostic tool which is named as "**Terraform**" and it's a cloud agnostic open source tool which supports multi cloud without any vendor locking constraints.

➤ **Solution:**

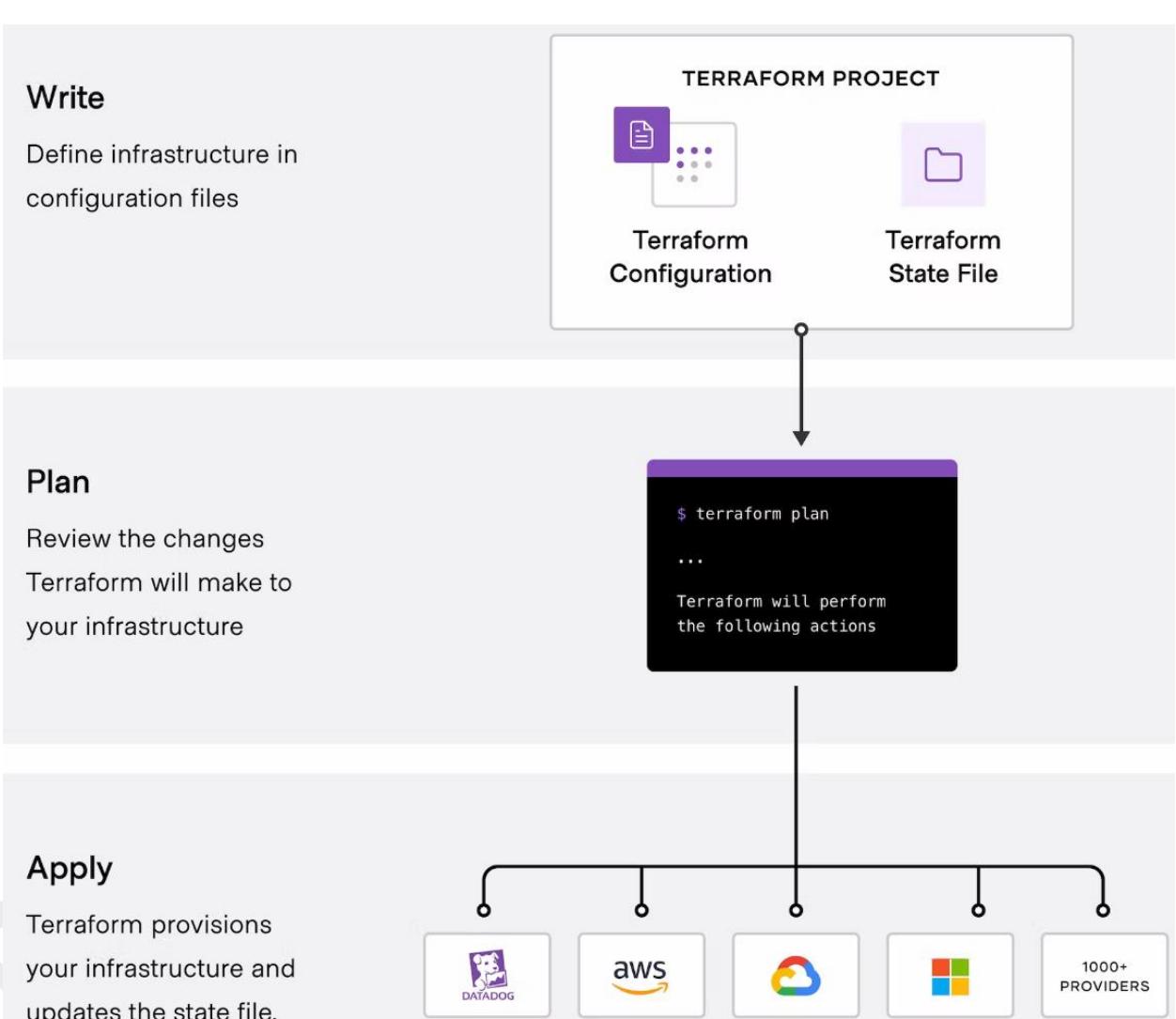
WHAT IS TERRAFORM ?

HashiCorp Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.

How does Terraform work?

HashiCorp and the Terraform community have already written **thousands of providers** to manage many different types of resources and services. You can find all publicly available providers on the Terraform Registry, including Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, DataDog, and many more.

The core Terraform workflow consists of three stages:



- **Write:** You define resources, which may be across multiple cloud providers and services. For example, you might create a configuration to deploy an application on virtual machines in a Virtual Private Cloud (VPC) network with security groups and a load balancer.
- **Plan:** Terraform creates an execution plan describing the infrastructure it will create, update, or destroy based on the existing infrastructure and your configuration.
- **Apply:** On approval, Terraform performs the proposed operations in the correct order, respecting any resource dependencies. For example, if you update the properties of a VPC and change the number of virtual

machines in that VPC, Terraform will recreate the VPC before scaling the virtual machines.

Ref source: <https://developer.hashicorp.com/terraform/intro>

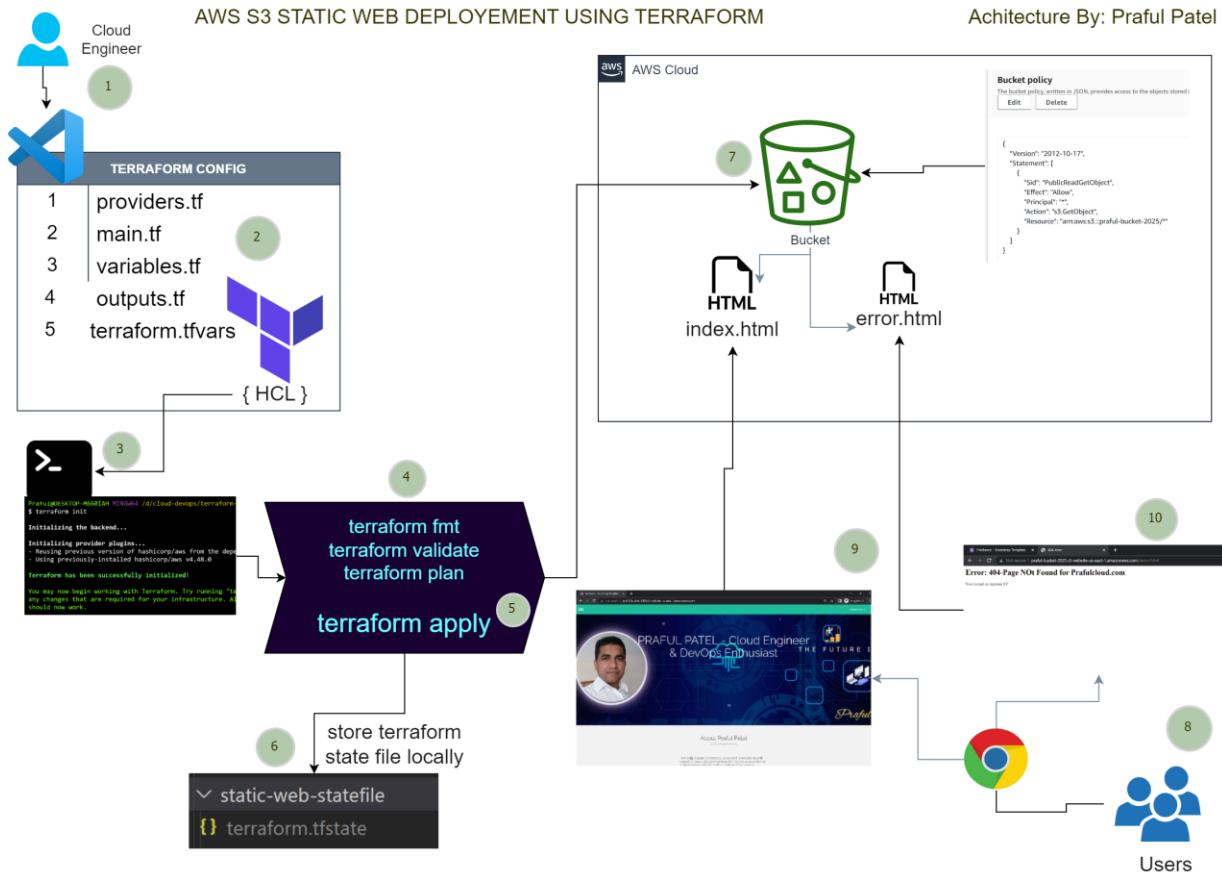
➤ **Project Cost Estimation:**

(Note: This cost is Not any actual cost, it's just an estimation based on high level requirement. Price may be vary based on adding and removing services based on requirement.)
Ref: <https://calculator.aws/#/addService>

➤ **Tools & Technologies covered:**

- Terraform (Infrastructure as Code)
- AWS Cloud Services
- AWS S3
- VS Code IDE
- GitHub

➤ **Solution Architecture:**



➤ **Pre-requisite:**

- 1) AWS Free tier Account
- 2) AWS IAM User created with programmatic access
- 3) VS Code configured
- 4) Latest Terraform version is installed
- 5) GitHub Account
- 6) Gitbash installed on desktop
- 7) Web application source code ready

This project will be completed in following implementation phases.

➤ **Project implementation Phase:**

- Phase 1: AWS cloud configuration via terminal
- Phase 2: Create terraform project and define Terraform files and folders
- Phase 3: Configure and write terraform resources into respective files
- Phase 4: Terraform fmt, terraform validate, terraform plan
- Phase 5: Terraform apply
- Phase 6: Verify that resources are created In cloud
- Phase 7: Verify that web application accessible successfully

➤ **Implementation:**

⊕ **Phase 1: AWS cloud configuration via terminal**

1. Go to gitbash terminal and hit "aws configure"
2. Provide secret key and secret access key for the IAM user created
3. Provide region "us-east-1"
4. Provide output type "json"

⊕ **Phase 2: Create terraform project and define Terraform files and folders**

1. Go to VS Code and create a new folder name "tf-static-web"
2. Create a file structure for terraform project
 - a. Create a new file "01-providers.tf"
 - b. Create a new file "02-main.tf"
 - c. Create a new file "03-variables.tf"
 - d. Create a new file "04-outputs.tf"
 - e. Create a new file "terraform.tfvars"
 - f. Create a folder for web app source code "webfiles" and upload web application files into it
3. Save all the files into VS code

⊕ **Phase 3: Configure and write terraform resources into respective files**

1. Go to file "01-providers.tf" and mention terraform and provider block
2. Go to "02-main.tf" and write all the required s3 resources
3. Go to "03-variables.tf" and write the variables for dynamic access
4. Go to "04-outputs.tf" and write required outputs
5. Go to "terraform.tfvars" and write the variables for dynamic access which will override the default variables value.

⊕ **Phase 4: Terraform fmt, terraform validate, terraform plan**

1. Go to git bash terminal and apply the terraform commands
2. Terraform init
3. Terraform fmt
4. Terraform validate
5. Terraform apply
6. Observe that "terraform.tfstate" is stored locally

⊕ **Phase 5: Verify that resources are created In AWS cloud**

1. Go to AWS S3 and verify that bucket is created
2. Verify that web files are uploaded into bucket
3. Verify that permission has been created
4. Verify that web endpoint is created

⊕ **Phase 6: Verify that web application accessible successfully**

1. Go to terminal and observe the outputs value
2. Copy the endpoint url and access from web browser
3. Verify that "index.html" is displayed the web page
4. Verify that "error.html" is displayed the error page

➤ Implementation in an Action:

⊕ **Phase 1: AWS cloud configuration via terminal**

- 1) Go to gitbash terminal and hit "aws configure"
- 2) Provide secret key and secret access key
- 3) Provide region " us-east-1"
- 4) Provide output type " json"

AWS Configuration:

Access Key Id:

AKIAWEYWGFCJC3IK25WS

Secret Access Key:

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

Go to terminal and type

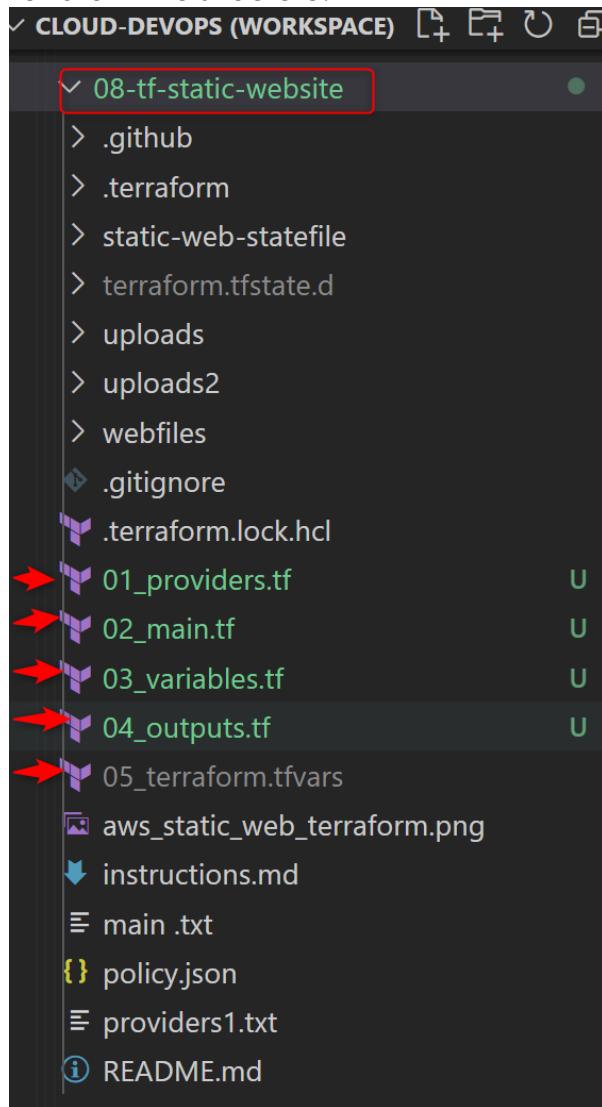
\$aws configure

```
Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-aws-tests/gocloud-test/08-tf-static-website (master)
$ aws configure
AWS Access Key ID [*****3NMH]: AKIA46G7T7645707RREU ①
AWS Secret Access Key [*****5QhB]: QFGsGe5WbY0k5QWA5LVvqgQIoYmHWP2d7+/11nIu ②
Default region name [us-east-1]: us-east-1 ③
Default output format [json]: json ④
```

⊕ **Phase 2: Create terraform project and define Terraform files and folders**

- 1) Go to VS Code and create a new folder name "tf-static-web"
- 2) Create a file structure for terraform project
 - a. Create a new file "01-providers.tf"
 - b. Create a new file "02-main.tf"
 - c. Create a new file "03-variables.tf"
 - d. Create a new file "04-outputs.tf"
 - e. Create a new file "terraform.tfvars"
 - f. Create a folder for web app source code "webfiles" and upload web application files into it
- 3) Save all the files into VS code

Terraform file structure:



⊕ **Phase 3: Configure and write terraform resources into respective files**

- 1) Go to file “01-providers.tf” and mention terraform and provider block
- 2) Go to “02-main.tf” and write all the required s3 resources
- 3) Go to “03-variables.tf” and write the variables for dynamic access
- 4) Go to “04-outputs.tf” and write required outputs
- 5) Go to “terraform.tfvars” and write the variables for dynamic access which will override the default variables value.

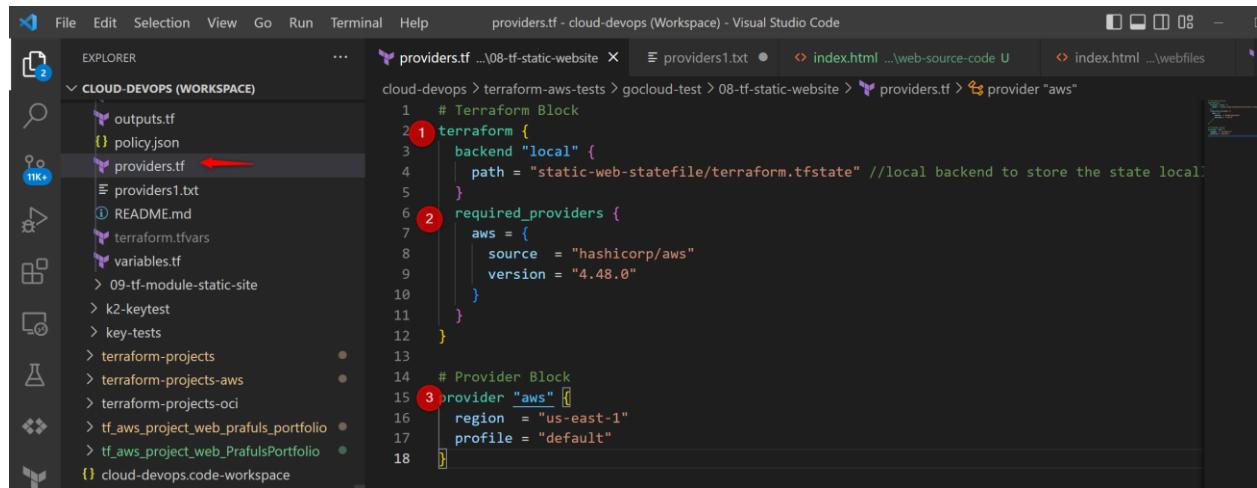
01_Providers.tf

- Terraform block
- Providers block
- Terraform block

Providers file contains terraform block which includes a “backend” configurations for state file storage and also “required_providers” contains a source and version of the terraform provider.

- Providers block

This block contains a cloud providers information about region and profile on which the cloud services needs to be provisioned.



```
1 # Terraform Block
2 terraform {
3   backend "local" {
4     path = "static-web-statefile/terraform.tfstate" //local backend to store the state local
5   }
6   required_providers {
7     aws = {
8       source  = "hashicorp/aws"
9       version = "4.48.0"
10    }
11  }
12 }
13
14 # Provider Block
15 provider "aws" {
16   region  = "us-east-1"
17   profile = "default"
18 }
```

02_main.tf

File Edit Selection View Go Run Terminal Help

main.tf ...\\aws-s3-static-web outputs.tf ...\\aws-s3-static-web variables.tf ...\\aws-s3-static-web 02_main.tf

EXPLORER CLOUD-DEVOPS (WORKSPACE)

- .gitignore
- .terraform.lock.hcl
- 01_providers.tf
- 02_main.tf
- 03_variables.tf
- 04_outputs.tf
- 05_terraform.tfvars
- aws_static_web_terraform.png
- instructions.md
- main.txt
- policy.json
- providers1.txt
- README.md
- 09-tf-module-static-site
- k2-keytest
- key-tests
- terraform-projects
- terraform-projects-aws
- terraform-projects-oci
- tf_aws_project_web_PrafulsPortfolio
- tf_aws_project_web_PrafulsPortfolio
- cloud-devops.code-workspace

OUTLINE

TIMELINE

TOMCAT SERVERS

File Edit Selection View Go Run Terminal Help

main.tf ...\\aws-s3-static-web outputs.tf ...\\aws-s3-static-web variables.tf ...\\aws-s3-static-web 02_main.tf

EXPLORER CLOUD-DEVOPS (WORKSPACE)

- .gitignore
- .terraform.lock.hcl
- 01_providers.tf
- 02_main.tf
- 03_variables.tf
- 04_outputs.tf
- 05_terraform.tfvars
- aws_static_web_terraform.png
- instructions.md
- main.txt
- policy.json
- providers1.txt
- README.md
- 09-tf-module-static-site
- k2-keytest
- key-tests
- terraform-projects
- terraform-projects-aws
- terraform-projects-oci
- tf_aws_project_web_PrafulsPortfolio
- tf_aws_project_web_PrafulsPortfolio
- cloud-devops.code-workspace

OUTLINE

31

32 1 data "aws_s3_bucket" "s3_bucket" {

33 | bucket = aws_s3_bucket.s3_bucket.bucket

34 }

35 //define locals block to map web application's content-type in order to display the web page

36 2 locals {

37 | content_type_map = {

38 | | html = "text/html",

39 | | js = "application/javascript",

40 | | css = "text/css",

41 | | svg = "image/svg+xml",

42 | | jpg = "image/jpeg",

43 | | ico = "image/x-icon",

44 | | png = "image/png",

45 | | gif = "image/gif",

46 | | pdf = "application/pdf"

47 | }

48 }

49 3 resource "aws_s3_bucket_object" "file" {

50 | for_each = fileset(var.web_root, "**")

51 | bucket = data.aws_s3_bucket.s3_bucket.bucket

52 | key = each.value

53 | source = "\${var.web_root}/\${each.value}"

54 | source_hash = filemd5("\${var.web_root}/\${each.value}")

55 | acl = "public-read"

56 | content_type = lookup(local.content_type_map, regex("\\.(?P<extension>[A-Za-z0-9]+)\$", each.value))

57 }

File Edit Selection View Go Run Terminal Help • 02_main.tf - cloud-devops (Workspace) - Visual Studio Code

EXPLORER CLOUD-DEVOPS (WORKSPACE)

```

main.tf ...\\aws-s3-static-web
outputs.tf ...\\aws-s3-static-web
variables.tf ...\\aws-s3-static-web
02_main.tf U

```

```

cloud-devops > terraform-aws-tests > gocloud-test > 08-tf-static-website > 02_main.tf > ...
59 //route 53
60 /*
61 resource "aws_route53_zone" "myzone" {
62   name = var.domain_name
63 }
64 */
65 data "aws_route53_zone" "myzone" {
66   name      = var.domain_name
67 }
68
69
70 2 resource "aws_route53_record" "exampleDomain" {
71   zone_id = data.aws_route53_zone.myzone.zone_id
72   name    = var.domain_name
73   type    = "A"
74   alias {
75     name          = aws_s3_bucket.s3_bucket.website_endpoint
76     zone_id      = aws_s3_bucket.s3_bucket.hosted_zone_id
77     evaluate_target_health = true
78   }
79 }
80

```

03_variables.tf

File Edit Selection View Go Run Terminal Help • 03_variables.tf - cloud-devops (Workspace) - Visual Studio Code

EXPLORER CLOUD-DEVOPS (WORKSPACE)

```

eb 02_main.tf U
aws_static_web_terraform.png policy.json main.txt 03_variables.tf U

```

```

cloud-devops > terraform-aws-tests > gocloud-test > 08-tf-static-website > 03_variables.tf > variable "bucket_name"
1 # Input variable definitions
2 1 variable "bucket_name" {
3   description = "Name of the S3 bucket. Must be Unique across AWS"
4   type        = string
5   default     = "praful-bucket-2023"
6 }
7
8 2 variable "tags" {
9   description = "Tags to set on the bucket"
10  type       = map(string)
11  default     = {}
12 }
13
14 3 variable "web_root" {
15   type        = string
16   description = "Path to the root of website content"
17   default     = "./webfiles"
18 }
19
20 4 variable "domain_name" {
21   default     = "prafulcloud.com"
22   type        = string
23 }

```

04_outputs.tf

File Edit Selection View Go Run Terminal Help • 04_outputs.tf - cloud-devops (Workspace) - Visual Studio Code

EXPLORER CLOUD-DEVOPS (WORKSPACE)

```

9-tf-module-static-site terraform.yml index.html ...\\iPortfolio\\05_terraform.tfvars 04_outputs.tf U

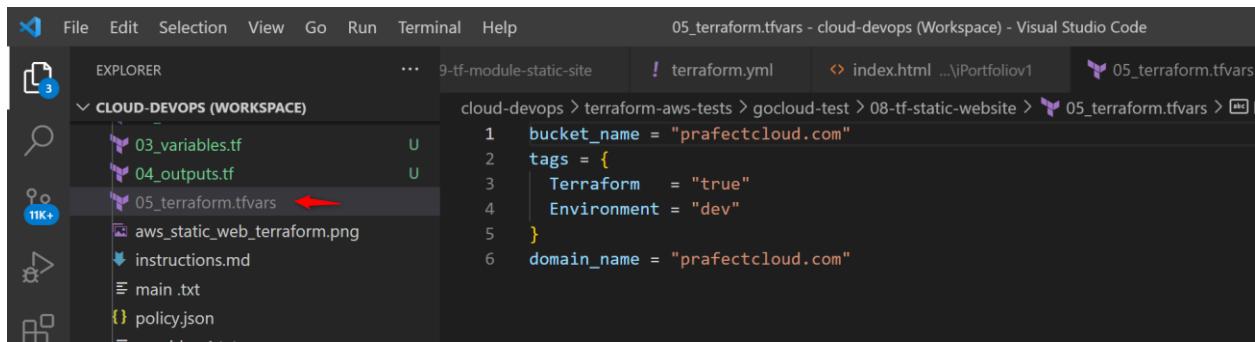
```

```

cloud-devops > terraform-aws-tests > gocloud-test > 08-tf-static-website > 04_outputs.tf > output "endpoint" > description
1 # Output variable definitions
2
3 1 output "arn" {
4   description = "ARN of the S3 Bucket"
5   value      = aws_s3_bucket.s3_bucket.arn
6 }
7
8 2 output "name" {
9   description = "Name (id) of the bucket"
10  value      = aws_s3_bucket.s3_bucket.id
11 }
12
13 3 output "domain" {
14   description = "Domain Name of the bucket"
15   value      = aws_s3_bucket.s3_bucket.website_domain
16 }
17
18 4 output "endpoint" {
19   description = "Endpoint Information of the bucket"
20   value      = aws_s3_bucket.s3_bucket.website_endpoint
21 }
22

```

05_terraform.tfvars.tf

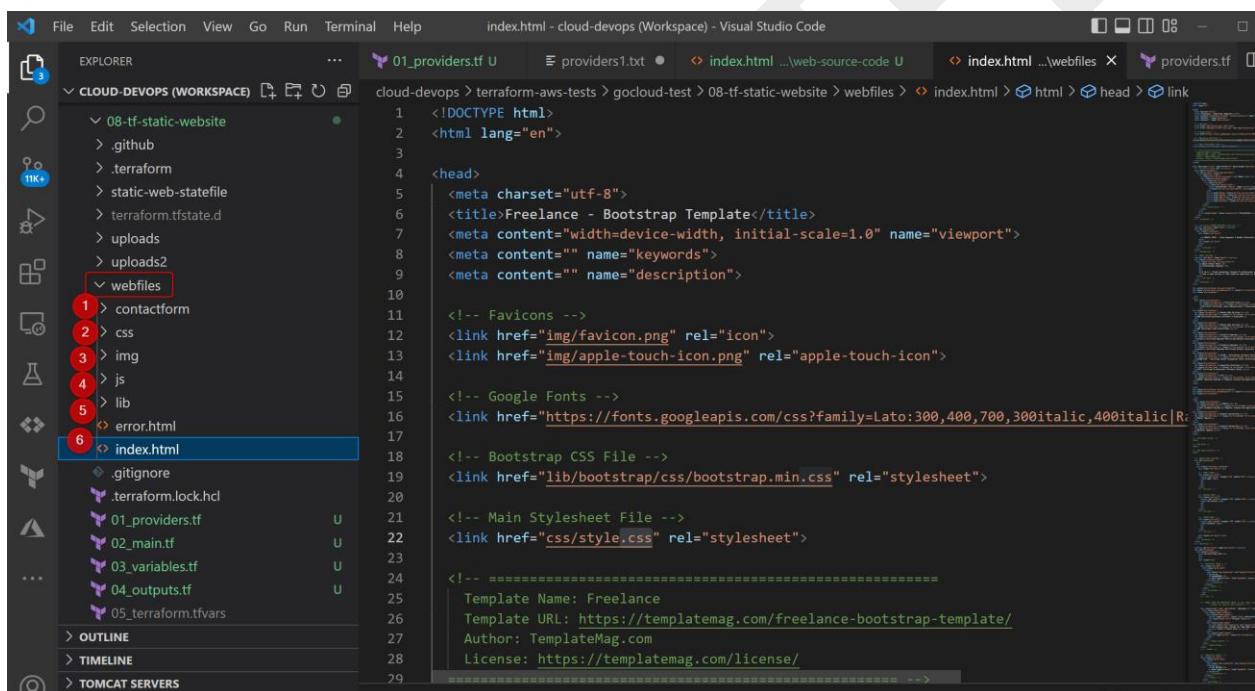


```

1 bucket_name = "prefectcloud.com"
2 tags = {
3   Terraform  = "true"
4   Environment = "dev"
5 }
6 domain_name = "prefectcloud.com"

```

Webfiles



```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Freelance - Bootstrap Template</title>
    <meta content="width=device-width, initial-scale=1.0" name="viewport">
    <meta content="" name="keywords">
    <meta content="" name="description">
    <!-- Favicons -->
    <link href="img/favicon.png" rel="icon">
    <link href="img/apple-touch-icon.png" rel="apple-touch-icon">
    <!-- Google Fonts -->
    <link href="https://fonts.googleapis.com/css?family=Lato:300,400,700,300italic,400italic|Raleway:400,700" rel="stylesheet">
    <!-- Bootstrap CSS File -->
    <link href="lib/bootstrap/css/bootstrap.min.css" rel="stylesheet">
    <!-- Main Stylesheet File -->
    <link href="css/style.css" rel="stylesheet">
  </head>
  <body>
    <div>Template Name: Freelance</div>
    <div>Template URL: https://templatemag.com/freelance-bootstrap-template/</div>
    <div>Author: TemplateMag.com</div>
    <div>License: https://templatemag.com/license/</div>
  </body>
</html>

```

```

Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-aws-tests/gocloud-test/08-tf-static-website (master)
$ aws configure
AWS Access Key ID [*****KYPM]: AKIAWEYWGFCJC3IK25WS ①
AWS Secret Access Key [*****1zTb]: Df9o4+N4pyUegJbQhNDeS3wl0vrAx0d01nkHh/0P ②
Default region name [us-east-1]: us-east-1 ③
Default output format [json]: json ④

```

AWS configured successfully

Phase 4: Terraform fmt, terraform validate, terraform plan

- 1) Go to git bash terminal and apply the terraform commands
- 2) Terraform init
- 3) Terraform fmt
- 4) Terraform validate
- 5) Terraform apply
- 6) Observe that "terraform.tfstate" is stored locally

Terraform init

```
Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-aws-tests/gocloud-test/08-tf-static-website (master)
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

Terraform has been successfully initialized!

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

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

Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-aws-tests/gocloud-test/08-tf-static-website (master)
$
```

Terraform fmt

```
Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-aws-tests/gocloud-test/08-tf-static-website (master)
$ terraform fmt
02_main.tf

Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-aws-tests/gocloud-test/08-tf-static-website (master)
$
```

Terraform validate

```
Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-aws-tests/gocloud-test/08-tf-static-website (master)
$ terraform validate

Warning: Argument is deprecated
  with aws_s3_bucket.s3_bucket,
  on 02_main.tf line 2, in resource "aws_s3_bucket" "s3_bucket":
  2: resource "aws_s3_bucket" "s3_bucket" {
    Use the aws_s3_bucket_website_configuration resource instead
    (and one more similar warning elsewhere)

Warning: Deprecated attribute
  on 02_main.tf line 75, in resource "aws_route53_record" "exampleDomain":
  75:   name                  = aws_s3_bucket.s3_bucket.website_endpoint
    The attribute "website_endpoint" is deprecated. Refer to the provider documentation for details.
    (and 2 more similar warnings elsewhere)

Success! The configuration is valid, but there were some validation warnings as shown above.
```

Terraform plan

```
Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-aws-tests/gocloud-test/08-tf-static-website (master)
$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create
  <= read (data resources)

Terraform will perform the following actions:

# data.aws_s3_bucket.s3_bucket will be read during apply
# (depends on a resource or a module with changes pending)
<= data "aws_s3_bucket" "s3_bucket" {
  + arn          = (known after apply)
  + bucket       = "praful-bucket-2023"
  + bucket_domain_name = (known after apply)
  + bucketRegional_domain_name = (known after apply)
  + hosted_zone_id = (known after apply)
  + id           = (known after apply)
  + region        = (known after apply)
  + website_domain = (known after apply)
  + website_endpoint = (known after apply)
}

  + server_side_encryption = (known after apply)
  + source      = "./webfiles/lib/php-mail-form/validate.js"
  + source_hash  = "48e28fea3f12ef0ef338510a77163d69"
  + storage_class = (known after apply)
  + tags_all    = (known after apply)
  + version_id   = (known after apply)
}

Plan: 43 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + arn      = (known after apply)
  + domain   = (known after apply)
  + endpoint = (known after apply)
  + name     = (known after apply)
```

Terraform apply

```
Praful@DESKTOP-M660IAH MINGW64 /d/cloud-devops/terraform-aws-tests/gocloud-test/08-tf-static-website (master)
$ terraform apply -auto-approve
```

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

```
  + create
  <= read (data resources)
```

Terraform will perform the following actions:

```
# data.aws_s3_bucket.s3_bucket will be read during apply
# (depends on a resource or a module with changes pending)
<= data "aws_s3_bucket" "s3_bucket" {
  + arn          = (known after apply)
  + bucket       = "praful-bucket-2023"
  + bucket_domain_name = (known after apply)
  + bucketRegional_domain_name = (known after apply)
  + hosted_zone_id = (known after apply)
  + id           = (known after apply)
  + region        = (known after apply)
  + website_domain = (known after apply)
  + website_endpoint = (known after apply)
}
```

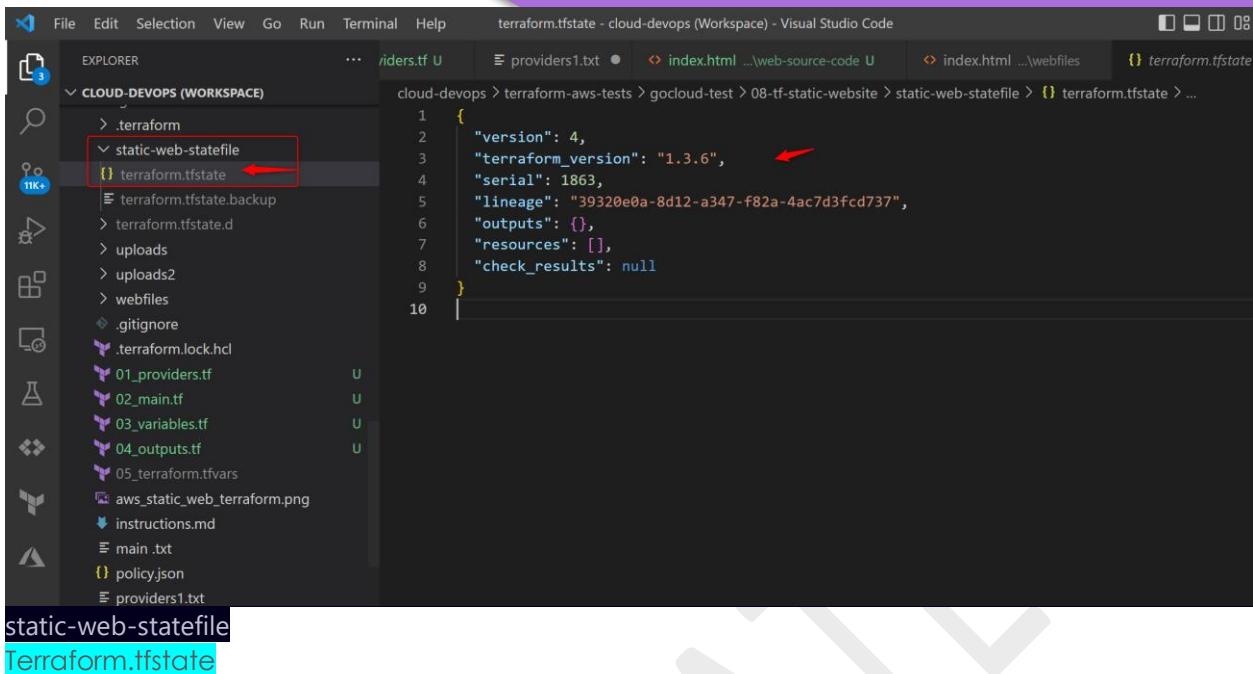
Apply complete! Resources: 43 added, 0 changed, 0 destroyed.

Outputs:

```
arn = "arn:aws:s3:::praful-bucket-2023"
domain = "s3-website-us-east-1.amazonaws.com"
endpoint = "praful-bucket-2023.s3-website-us-east-1.amazonaws.com" ←
name = "praful-bucket-2023"
```

static-web-statefile

Terraform.tfstate



File Edit Selection View Go Run Terminal Help terraform.tfstate - cloud-devops (Workspace) - Visual Studio Code

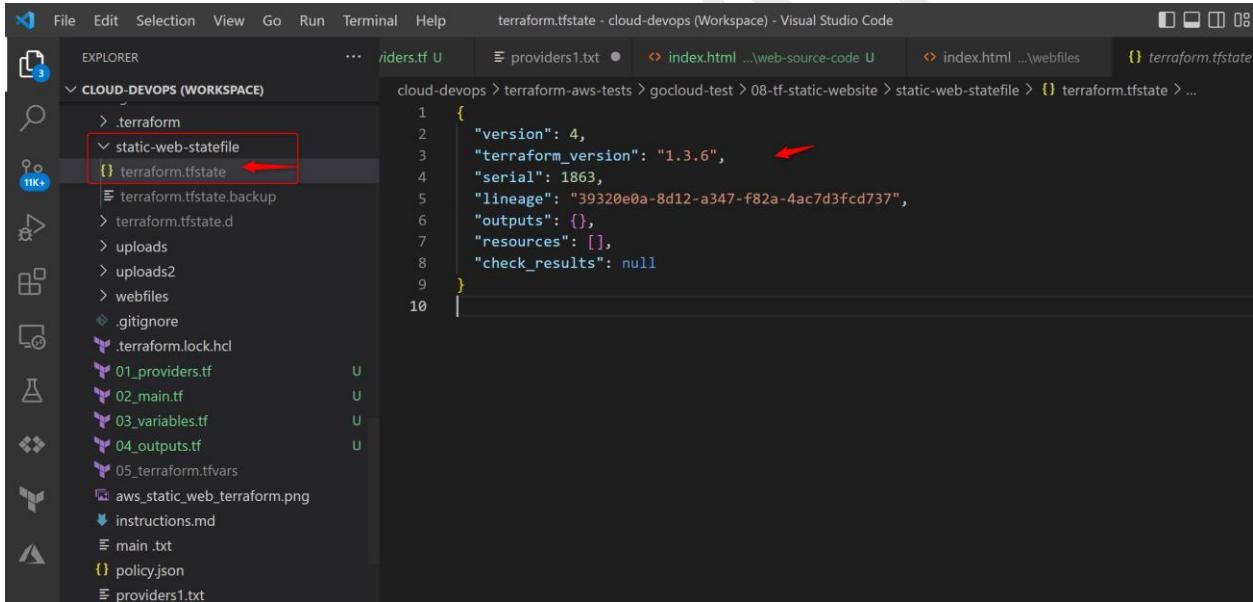
EXPLORER providers1.txt U providers1.txt U index.html ...\\web-source-code U index.html ...\\webfiles U terraform.tfstate

CLOUD-DEVOPS (WORKSPACE)

- > .terraform
- static-web-statefile
 - terraform.tfstate
- terraform.tfstate.backup
- terraform.tfstate.d
- uploads
- uploads2
- webfiles
- .gitignore
- .terraform.lock.hcl
- 01_providers.tf
- 02_main.tf
- 03_variables.tf
- 04_outputs.tf
- 05_terraform.tfvars
- aws_static_web_terraform.png
- instructions.md
- main .txt
- policy.json
- providers1.txt

static-web-statefile

Terraform.tfstate



File Edit Selection View Go Run Terminal Help terraform.tfstate - cloud-devops (Workspace) - Visual Studio Code

EXPLORER providers1.txt U providers1.txt U index.html ...\\web-source-code U index.html ...\\webfiles U terraform.tfstate

CLOUD-DEVOPS (WORKSPACE)

- > .terraform
- static-web-statefile
 - terraform.tfstate
- terraform.tfstate.backup
- terraform.tfstate.d
- uploads
- uploads2
- webfiles
- .gitignore
- .terraform.lock.hcl
- 01_providers.tf
- 02_main.tf
- 03_variables.tf
- 04_outputs.tf
- 05_terraform.tfvars
- aws_static_web_terraform.png
- instructions.md
- main .txt
- policy.json
- providers1.txt

Phase 5: Verify that resources are created In AWS cloud

- 1) Go to AWS S3 and verify that bucket is created
- 2) Verify that web files are uploaded into bucket
- 3) Verify that permission has been created
- 4) Verify that web endpoint is created

Verify the resources are created in AWS

Amazon S3

Buckets

- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- Access analyzer for S3

Block Public Access settings for this account

Storage Lens

- Dashboards
- AWS Organizations settings

Feature spotlight

AWS Marketplace for S3

Amazon S3 > Buckets

Account snapshot

Total storage: Pending, Object count: Pending, Average object size: Pending

You can enable advanced metrics in the "default-account-dashboard" configuration.

Buckets (1) Info

Buckets are containers for data stored in S3. Learn more

Name	AWS Region	Access	Creation date
praful-bucket-2023	US East (N. Virginia) us-east-1	Public	December 27, 2022, 19:41:37 (UTC-06:00)

Amazon S3

Buckets

- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- Access analyzer for S3

Block Public Access settings for this account

Storage Lens

- Dashboards
- AWS Organizations settings

Feature spotlight

AWS Marketplace for S3

praful-bucket-2023 Info

Publicly accessible

Objects

Objects (7)

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
contactform/	Folder	-	-	-
css/	Folder	-	-	-
error.html	html	December 27, 2022, 19:41:41 (UTC-06:00)	210.0 B	Standard
img/	Folder	-	-	-
index.html	html	December 27, 2022, 19:41:41 (UTC-06:00)	23.2 KB	Standard
js/	Folder	-	-	-
lib/	Folder	-	-	-

Amazon S3

Buckets

- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- Access analyzer for S3

Block Public Access settings for this account

Storage Lens

- Dashboards
- AWS Organizations settings

Feature spotlight

AWS Marketplace for S3

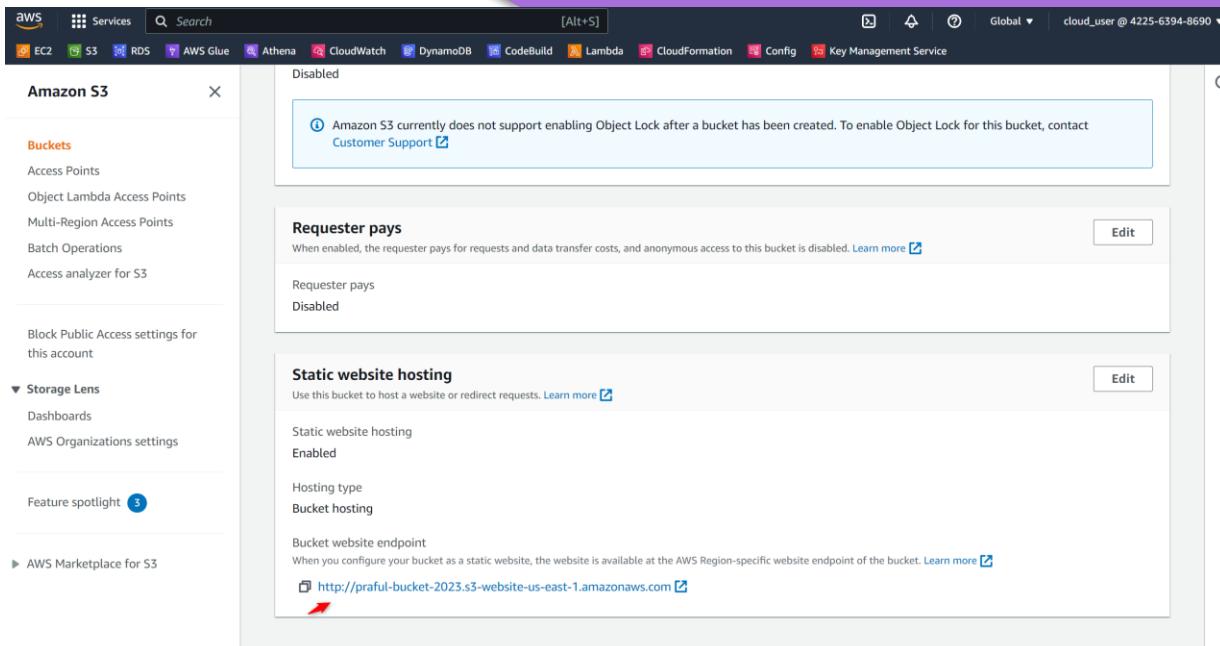
Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

Edit **Delete**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::praful-bucket-2023/*"
    }
  ]
}
```

Copy



Amazon S3

Buckets

- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- Access analyzer for S3

Block Public Access settings for this account

Storage Lens

- Dashboards
- AWS Organizations settings

Feature spotlight 3

AWS Marketplace for S3

Disabled

Amazon S3 currently does not support enabling Object Lock after a bucket has been created. To enable Object Lock for this bucket, contact [Customer Support](#)

Requester pays

When enabled, the requester pays for requests and data transfer costs, and anonymous access to this bucket is disabled. [Learn more](#)

Requester pays

Disabled

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting

Enabled

Hosting type

Bucket hosting

Bucket website endpoint

When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)

<http://praful-bucket-2023.s3-website-us-east-1.amazonaws.com>

⊕ Phase 6: Verify that web application accessible successfully

- 1) Go to terminal and observe the outputs value
- 2) Copy the endpoint url and access from web browser
- 3) Verify that "index.html" is displayed the web page
- 4) Verify that "error.html" is displayed the error page

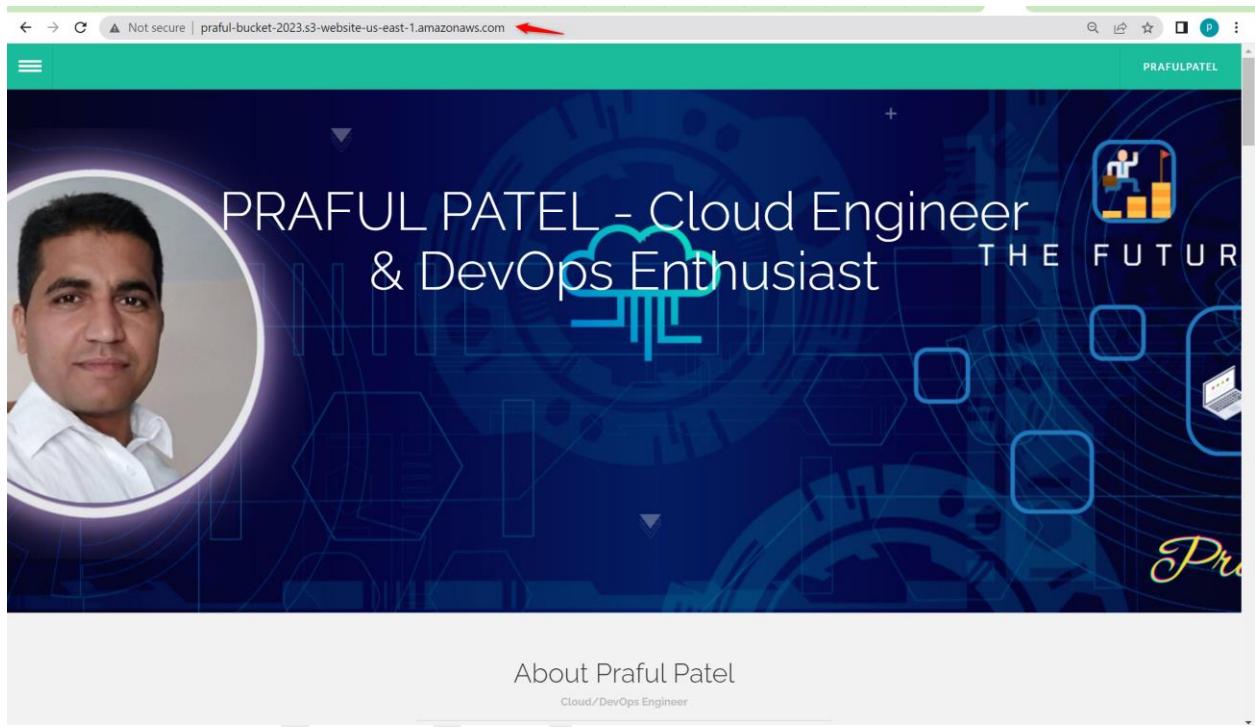
Verify that "index.html" is displayed the web page

Access web page using endpoint

S3 endpoint:

praful-bucket-2023.s3-website-us-east-1.amazonaws.com

index.html



Verify that "error.html" is displayed the error page

Access "error.html"



Congratulations!!!! 🎉🎉