

Case study overview

This case study focuses on utilizing Terraform, a popular infrastructure-as-code tool, to provision cloud infrastructure on AWS. Specifically, the study will cover how to create an EC2 instance and an S3 bucket with Terraform and deploy a static website on the S3 bucket while using the EC2 instance as the backend for logging and interactions.

Key Feature and Application:

The primary feature of this case study is automating cloud infrastructure management with Terraform. This includes provisioning both an EC2 instance and an S3 bucket, which are foundational elements in cloud services. The practical application involves deploying a website and logging interactions between the EC2 instance and the S3 bucket, showcasing a real-world scenario of infrastructure automation.

1. **Automated Provisioning:** Leveraging Terraform to script the creation of AWS resources, ensuring consistent and repeatable deployments.
2. **Scalability:** Using AWS S3 for hosting a static website provides a scalable and cost-effective solution.
3. **Backend Server:** An EC2 instance serves as a backend server, allowing dynamic interactions with the static website hosted on S3.
4. **Logging and Monitoring:** The EC2 instance logs actions performed on the S3 bucket, providing insights into usage and access patterns.

Application

This case study exemplifies the practical application of Infrastructure as Code principles to streamline resource management in AWS. By automating the provisioning of an EC2 instance and an S3 bucket, we can deploy and manage a static website efficiently. The EC2 instance's role as a backend server demonstrates how to extend static websites with dynamic capabilities, logging interactions to monitor and optimize performance. This approach ensures scalability, reliability, and ease of maintenance, making it ideal for modern cloud-based applications. Through this example, we see how Terraform, AWS S3, and EC2 come together to provide a robust solution for deploying and managing cloud resources, reinforcing the value of automation and cloud-native practices.

Implementation

Step 1: Initial Setup

1. **Install Terraform:**

If Terraform is not already installed, install it on your local machine. Follow the official Terraform documentation [here](#) for platform-specific instructions.

2. **Configure AWS CLI:**

Ensure the AWS CLI is installed and configured with your access keys:

Step 2: Write the Terraform Script

1. **Main Configuration File (main.tf):** Create a new directory for the project and create a file called `main.tf`. This file will contain the code to provision the EC2 instance and S3 bucket.

```
provider "aws" {  
  
    region = "ap-south-1" # Change to your desired region  
  
}  
  
resource "aws_s3_bucket" "website_bucket" {  
  
    bucket = "my-terraform-static-website-bucket"  
  
    # Enable bucket versioning if needed  
  
    versioning {  
  
        enabled = true  
  
    }  
  
    # Disable ACLs (this line can be removed since it's the default  
behavior)  
  
    # acl      = "private"
```

```
# Optional: Enable logging

logging {

    target_bucket = aws_s3_bucket.logging_bucket.bucket

    target_prefix = "log/"

}

}

# Define the S3 bucket for logging (if you want to enable logging)

resource "aws_s3_bucket" "logging_bucket" {

    bucket = "my-terraform-logs-bucket"

    acl    = "private"

}

# Configure the static website settings

resource "aws_s3_bucket_website_configuration" "website" {

    bucket = aws_s3_bucket.website_bucket.id

    index_document {

        suffix = "index.html"

    }

    error_document {

        key = "404.html" # Optional, define if you have an error
document
    }

}
```

```
}  
  
}
```

Step 3: Deploy the Static Website on S3

1. Create an **index.html** File: Create a simple HTML file in your project directory:

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-  
scale=1.0">  
  <title>Custom T-Shirt Order Form</title>  
</head>  
<body>  
  <h1>Custom T-Shirt Order Form</h1>  
  <form action="/submit-order" method="post">  
    <fieldset>  
      <legend>T-Shirt Details</legend>  
  
      <label for="tagline">Tagline on the Shirt:</label>  
      <input type="text" id="tagline" name="tagline" required  
placeholder="Enter your tagline">  
  
      <label for="color">Color:</label>  
      <select id="color" name="color" required>  
        <option value="">Select a color</option>  
        <option value="white">White</option>  
        <option value="black">Black</option>  
        <option value="red">Red</option>  
        <option value="blue">Blue</option>  
        <option value="green">Green</option>  
      </select>  
  
      <label for="size">Size:</label>  
      <select id="size" name="size" required>  
        <option value="">Select a size</option>
```

```
        <option value="s">Small</option>
        <option value="m">Medium</option>
        <option value="l">Large</option>
        <option value="xl">X-Large</option>
        <option value="xxl">XX-Large</option>
    </select>

    <label for="quantity">Quantity:</label>
    <input type="number" id="quantity" name="quantity"
min="1" max="100" required>

    <label for="delivery-date">Delivery Date:</label>
    <input type="date" id="delivery-date" name="delivery-
date" required>
</fieldset>

<fieldset>
    <legend>Delivery Details</legend>

    <label for="recipient-name">Recipient's Name:</label>
    <input type="text" id="recipient-name" name="recipient-
name" required>

    <label for="address">Address:</label>
    <input type="text" id="address" name="address" required>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>

    <label for="phone">Phone Number:</label>
    <input type="tel" id="phone" name="phone" pattern="[0-
9]{10}" required placeholder="1234567890">
</fieldset>

    <label for="comments">Additional Comments:</label>
    <textarea id="comments" name="comments" rows="4" cols="50"
placeholder="Any special instructions?"></textarea>

    <button type="submit">Place Order</button>
    <button type="reset">Reset Form</button>
```

```
</form>  
</body>  
</html>
```

Step 4: Initialize and Apply the Terraform Script

1. **Initialize the Directory:** Run `terraform init` in your project directory to initialize Terraform and install required providers.

Apply the Terraform Configuration: Run `terraform apply` to provision the resources. Terraform will prompt for confirmation before applying changes.

```
PS C:\Users\Admin\Desktop\terraform> terraform plan
aws_instance.example: Refreshing state... [id=i-0fd2bbfb9b20a58ca]
```

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.website_bucket will be created
+ resource "aws_s3_bucket" "website_bucket" {
  + acceleration_status      = (known after apply)
  + acl                      = "public-read"
  + arn                      = (known after apply)
  + bucket                  = "my-terraform-static-website-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_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)

  + object_lock_configuration (known after apply)
```

```
PS C:\Users\Admin\Desktop\terraform> terraform plan
aws_instance.example: Refreshing state... [id=i-0fd2bbfb9b20a58ca]
```

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.website_bucket will be created
+ resource "aws_s3_bucket" "website_bucket" {
  + acceleration_status      = (known after apply)
  + acl                      = "public-read"
  + arn                     = (known after apply)
  + bucket                  = "my-terraform-static-website-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_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)

  + object_lock_configuration (known after apply)
```



PROBLEMS 3

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

 powershell + -

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

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

Warning: Argument is deprecated

```
with aws_s3_bucket.website_bucket,
on main.tf line 13, in resource "aws_s3_bucket" "website_bucket":
13: resource "aws_s3_bucket" "website_bucket" {
```

Use the `aws_s3_bucket_website_configuration` resource instead

(and 3 more similar warnings elsewhere)


PROBLEMS 3

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

 powershell + -

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

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

Warning: Argument is deprecated

```
with aws_s3_bucket.website_bucket,
on main.tf line 13, in resource "aws_s3_bucket" "website_bucket":
13: resource "aws_s3_bucket" "website_bucket" {
```

Use the `aws_s3_bucket_website_configuration` resource instead

(and 3 more similar warnings 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" now.

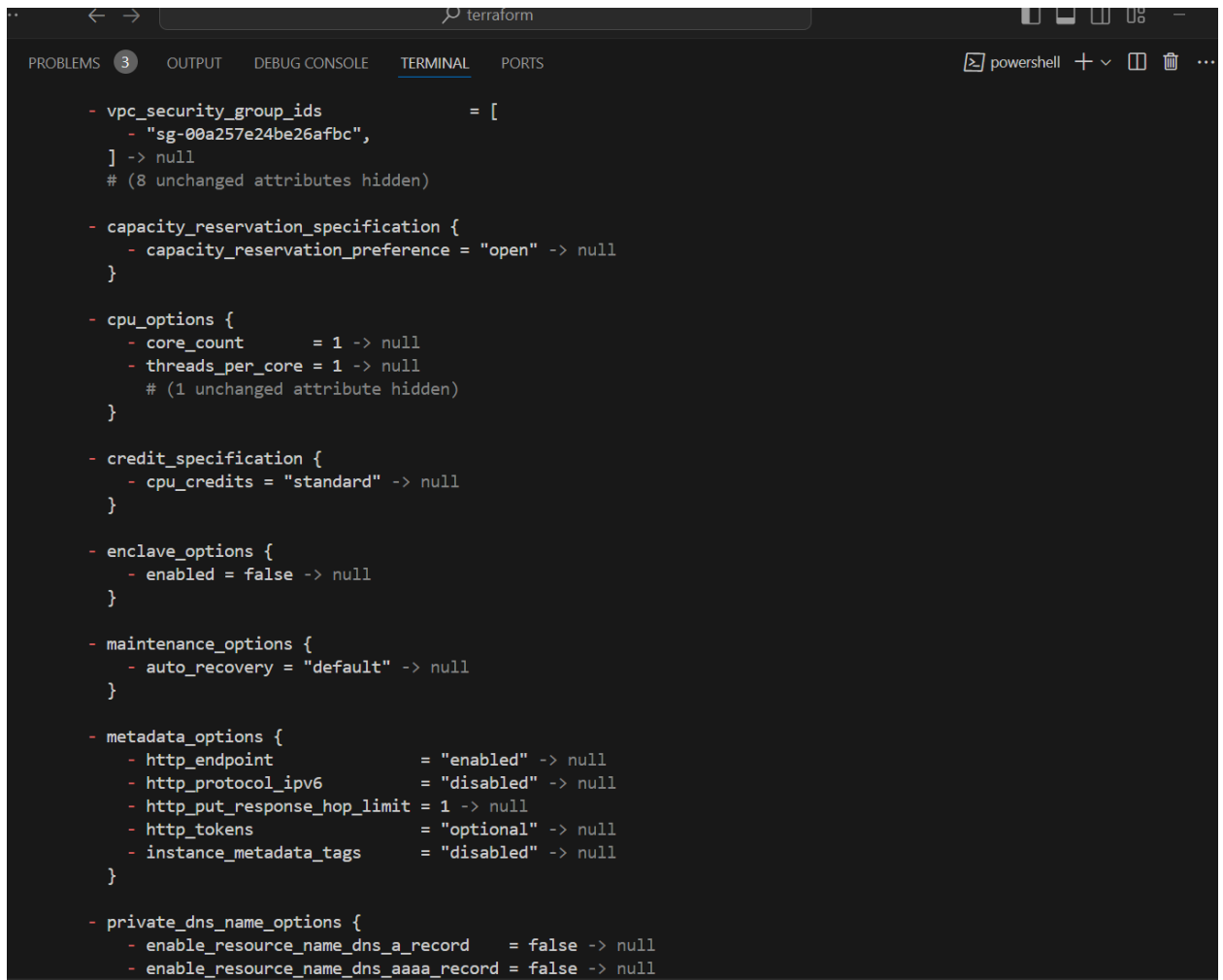
```
PS C:\Users\Admin\Desktop\terraform> terraform apply
aws_instance.example: Refreshing state... [id=i-0fd2bbfb9b20a58ca]
```

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

- + create
- destroy

Terraform will perform the following actions:

```
# aws_instance.example will be destroyed
# (because aws_instance.example is not in configuration)
- resource "aws_instance" "example" {
  - ami                        = "ami-0e8071008188cd555" -> null
  - arn                      = "arn:aws:ec2:ap-south-1:010928205872:instance/i-0fd2bbfb9b20a58ca" -> null
  - associate_public_ip_address = true -> null
  - availability_zone         = "ap-south-1b" -> null
  - cpu_core_count            = 1 -> null
  - cpu_threads_per_core      = 1 -> null
  - disable_api_stop          = false -> null
  - disable_api_termination   = false -> null
  - ebs_optimized              = false -> null
  - get_password_data          = false -> null
  - hibernation                = false -> null
  - id                        = "i-0fd2bbfb9b20a58ca" -> null
  - instance_initiated_shutdown_behavior = "stop" -> null
  - instance_state            = "running" -> null
  - instance_type              = "t2.micro" -> null
  - ipv6_address_count         = 0 -> null
  - ipv6_addresses             = [] -> null
  - monitoring                 = false -> null
  - placement_partition_number = 0 -> null
  - primary_network_interface_id = "eni-0c3e4ae402656554c" -> null
  - private_dns                = "ip-172-31-8-175.ap-south-1.compute.internal" -> null
  - private_ip                 = "172.31.8.175" -> null
```



```
# aws_s3_bucket.logging_bucket will be created
+ resource "aws_s3_bucket" "logging_bucket" {
  + acceleration_status = (known after apply)
  + acl                 = "private"
  + arn                 = (known after apply)
  + bucket              = "my-terraform-logs-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_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)

  + 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)
```

```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + - []

+ 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 {
  + target_bucket = "my-terraform-logs-bucket"
  + target_prefix = "log/"
}

+ object_lock_configuration (known after apply)

+ replication_configuration (known after apply)

+ server_side_encryption_configuration (known after apply)

+ versioning {
  + enabled   = true
  + mfa_delete = false
}

+ website (known after apply)
}

# aws_s3_bucket_website_configuration.website will be created
+ resource "aws_s3_bucket_website_configuration" "website" {
  + bucket = (known after apply)
```

```
+ error_document {
+   + key = "404.html"
+ }

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

+ routing_rule (known after apply)
}

Plan: 3 to add, 0 to change, 1 to destroy.

Warning: Argument is deprecated

with aws_s3_bucket.website_bucket,
on main.tf line 5, in resource "aws_s3_bucket" "website_bucket":
5: resource "aws_s3_bucket" "website_bucket" {

Use the aws_s3_bucket_versioning resource instead

(and 4 more similar warnings 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_instance.example: Destroying... [id=i-0fd2bbfb9b20a58ca]
aws_s3_bucket.logging_bucket: Creating...
aws_s3_bucket.logging_bucket: Creation complete after 1s [id=my-terraform-logs-bucket]
aws_s3_bucket.website_bucket: Creating...

aws_instance.example: Destroying... [id=i-0fd2bbfb9b20a58ca]
aws_s3_bucket.logging_bucket: Creating...
aws_s3_bucket.logging_bucket: Creation complete after 1s [id=my-terraform-logs-bucket]
aws_s3_bucket.website_bucket: Creating...
aws_s3_bucket.website_bucket: Creation complete after 2s [id=my-terraform-static-website-bucket]
aws_s3_bucket_website_configuration.website: Creating...
aws_s3_bucket_website_configuration.website: Creation complete after 0s [id=my-terraform-static-website-bucket]
aws_instance.example: Still destroying... [id=i-0fd2bbfb9b20a58ca, 10s elapsed]
aws_instance.example: Still destroying... [id=i-0fd2bbfb9b20a58ca, 20s elapsed]
aws_instance.example: Still destroying... [id=i-0fd2bbfb9b20a58ca, 30s elapsed]
aws_instance.example: Still destroying... [id=i-0fd2bbfb9b20a58ca, 40s elapsed]
aws_instance.example: Destruction complete after 40s

Warning: Argument is deprecated

with aws_s3_bucket.website_bucket,
on main.tf line 5, in resource "aws_s3_bucket" "website_bucket":
5: resource "aws_s3_bucket" "website_bucket" {

Use the aws_s3_bucket_logging resource instead

(and 2 more similar warnings elsewhere)

Apply complete! Resources: 3 added, 0 changed, 1 destroyed.
PS C:\Users\Admin\Desktop\terraform>
```

Step 5: Deploy the Static Website on S3

1. **Upload the Static Website Files:** Create a simple `index.html` file for your static website. Upload it to the S3 bucket created by Terraform.

```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>aws s3 cp index.html s3://my-terraform-static-website-bucket/
upload: .\index.html to s3://my-terraform-static-website-bucket/index.html
```

Step 6

.Connect ec2 instance using ssh command

[illegible]

ap-south-1.console.aws.amazon.com/s3/object/my-terraform-static-website-bucket/region=ap-south-1&bucketType=general&prefix=index.html

Amazon S3

Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3

Block Public Access settings for this account

▼ Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings

Feature spotlight 7

► AWS Marketplace for S3

Amazon S3 > Buckets > my-terraform-static-website-bucket > index.html

index.html [Info](#)

[Copy S3 URI](#) [Download](#) [Open](#) [Object actions](#)

[Properties](#) [Permissions](#) [Versions](#)

Object overview

Owner
17551de2936d55e486b31017cedb0822096907d6cd907e1fa6b81016afefe669

AWS Region
Asia Pacific (Mumbai) ap-south-1

Last modified
October 23, 2024, 20:11:01 (UTC+05:30)

Size
2.8 KB

Type
html

Key
index.html

S3 URI
<s3://my-terraform-static-website-bucket/index.html>

Amazon Resource Name (ARN)
<arn:aws:s3::my-terraform-static-website-bucket/index.html>

Entity tag (Etag)
[b86758de7af404a17b458c805efaa508](#)

Object URL
<https://my-terraform-static-website-bucket.s3.ap-south-1.amazonaws.com/index.html>

Object management overview

ap-south-1.console.aws.amazon.com/s3/home?region=ap-south-1

Amazon S3

Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3

Block Public Access settings for this account

▼ Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings

Feature spotlight 7

► AWS Marketplace for S3

Amazon S3

► **Account snapshot - updated every 24 hours** [All AWS Regions](#) [View Storage Lens dashboard](#)

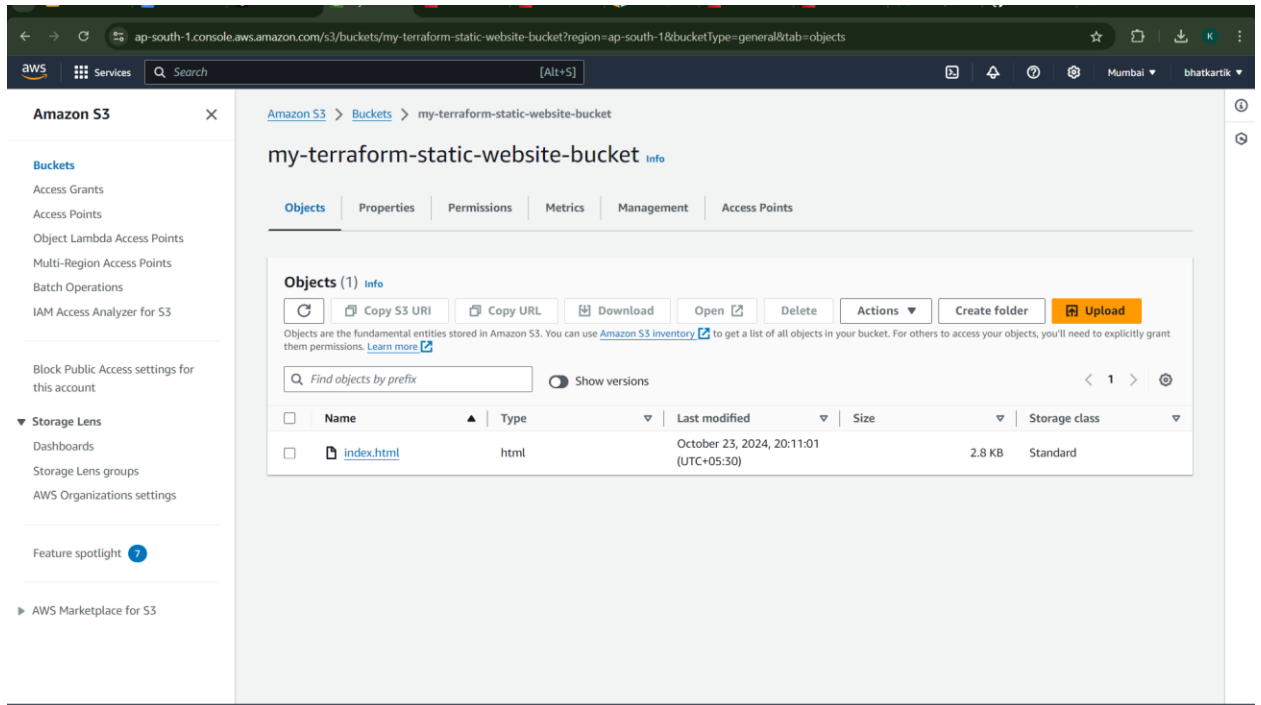
Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

[General purpose buckets](#) [Directory buckets](#)

General purpose buckets (2) [Info](#) [All AWS Regions](#) [Refresh](#) [Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	my-terraform-logs-bucket	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	October 23, 2024, 19:51:26 (UTC+05:30)
<input type="radio"/>	my-terraform-static-website-bucket	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	October 23, 2024, 19:51:27 (UTC+05:30)



Guidelines:

- **IAMPolicies:** Ensure that the IAM user used in Terraform has the least privilege necessary to avoid over-permissioning.
- **Bucket Naming:** S3 bucket names must be globally unique, so modify the name accordingly.
- **CostManagement:** Keep track of your running instances and S3 storage usage to prevent unwanted costs.
- **Backupyour .pem file:** The SSH key should be stored securely as it is required to access the EC2 instance

3. Demonstration Preparation

- **Key Points:**
 - Highlight the use of Terraform to automate AWS infrastructure creation.
 - Demonstrate the process of creating an EC2 instance and S3 bucket.
 - Show how to deploy the website and log interactions.
- **Practice:**
 - Walk through each step of the process to ensure familiarity.
 - Test the website and log commands to verify everything works as expected.
- **Questions:**
 - What advantages does Terraform provide over manual cloud resource creation?

- How can you scale this solution for larger infrastructure setups?
- What other AWS services could be integrated using Terraform?

Certification Requirement

- **Deadline:** Ensure submission of the complete report and certification by **October 21st**.

Importance:

The certification is crucial for validating your proficiency in using Terraform for AWS resource management. It demonstrates your ability to automate infrastructure provisioning, ensuring repeatable and reliable deployments. Achieving certification underscores your expertise in cloud-native practices, which is essential for the successful execution and understanding of this case study. It shows you're not just implementing a solution, but doing so with best practices and recognized skills. Make sure to complete your certification by the deadline to fully leverage this case study's potential and your professional growth.