Static Website Deployment on AWS using Terraform

Course: DLBSEPCP01_E _Cloud Programming

Student Name: Kinyua Kennedy

Date of submission: 11/06/2025

Project Objective:

1. Host a simple static website with high availability and HTTPS
2. Automate all infrastructure using Terraform
3. Learn and demonstrate core AWS cloud services

Solution Overview

This project leverages AWS S3 for website hosting, CloudFront for global content delivery, and Terraform to automate infrastructure creation.
The site contains documentation about the project itself, demonstrating a practical application of AWS cloud services and DevOps principles.

Target Audience:

This project concept is primarily beneficial for Students/Learners of AWS & Terraform.

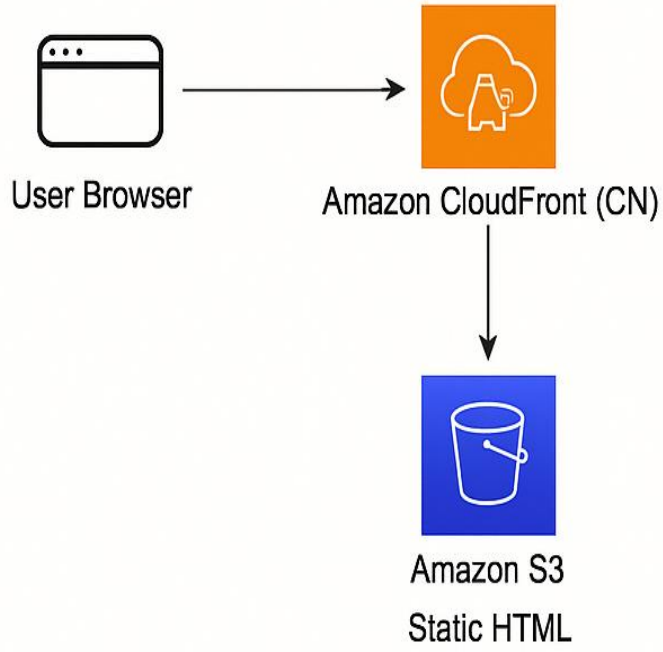
- Providing a hands-on, reproducible blueprint for understanding fundamental cloud infrastructure concepts.

Architecture Diagram

This concept outlines the design for hosting a simple, highly available static website on AWS.

The static consent will be served via Amazon S3 and distributed globally using Amazon CloudFront, providing low-latency access for users worldwide.

The entire infrastructure will be managed using Terraform, ensuring reproducible, version-controlled deployments.



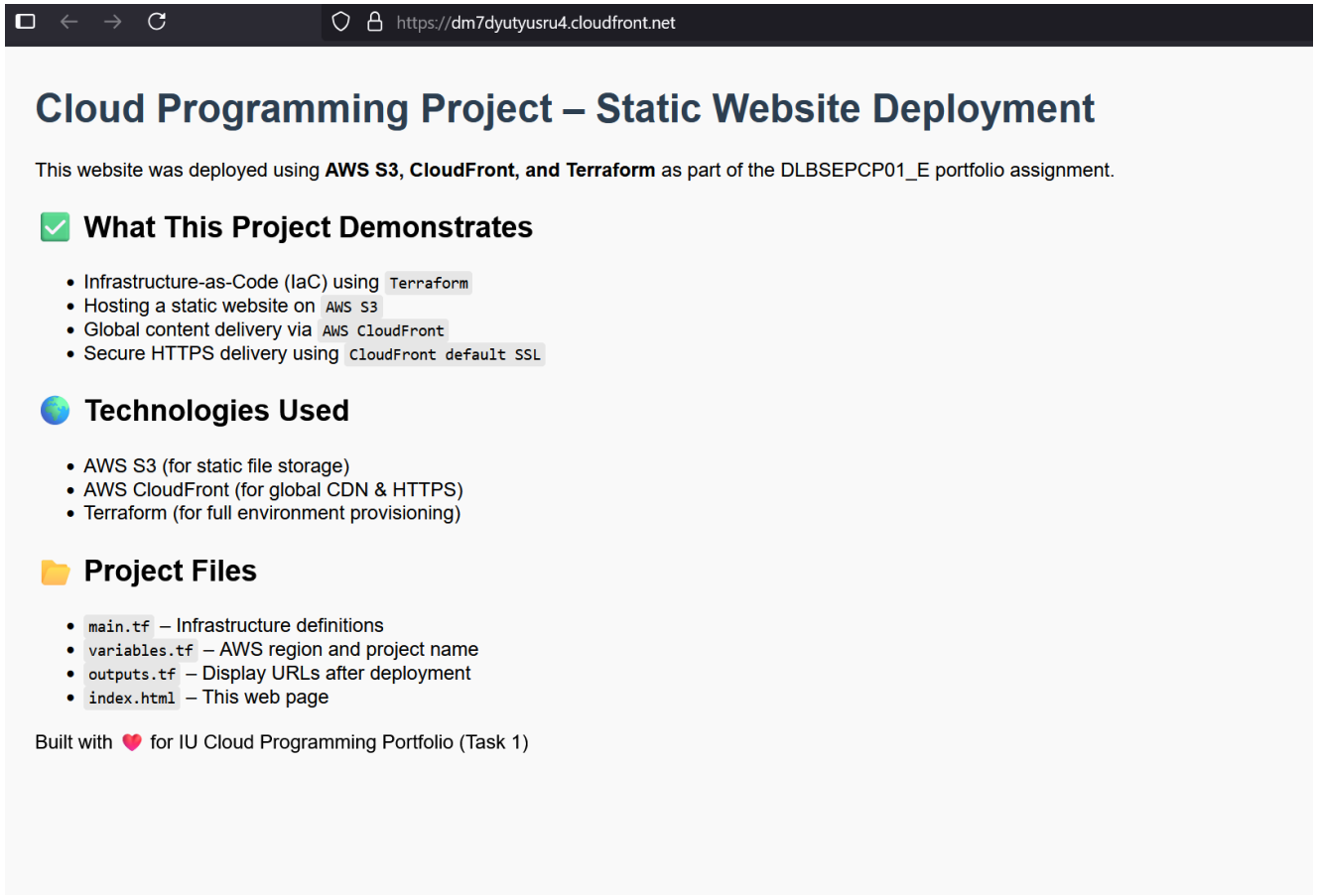
Tools & Services Used

- **Terraform** – Infrastructure as code
- **AWS S3** – Stores static content
- **CloudFront** – Global CDN + HTTPS
- **IAM** – Access control and bucket policies

Terraform File Breakdown

- ***main.tf*****:** Defines S3, CloudFront, and IAM
- ***variables.tf*****:** Holds region/project settings
- ***outputs.tf:*** Displays live URLs
- ***index.html*****:** My live webpage

HTML Page Purpose - Hyperlink

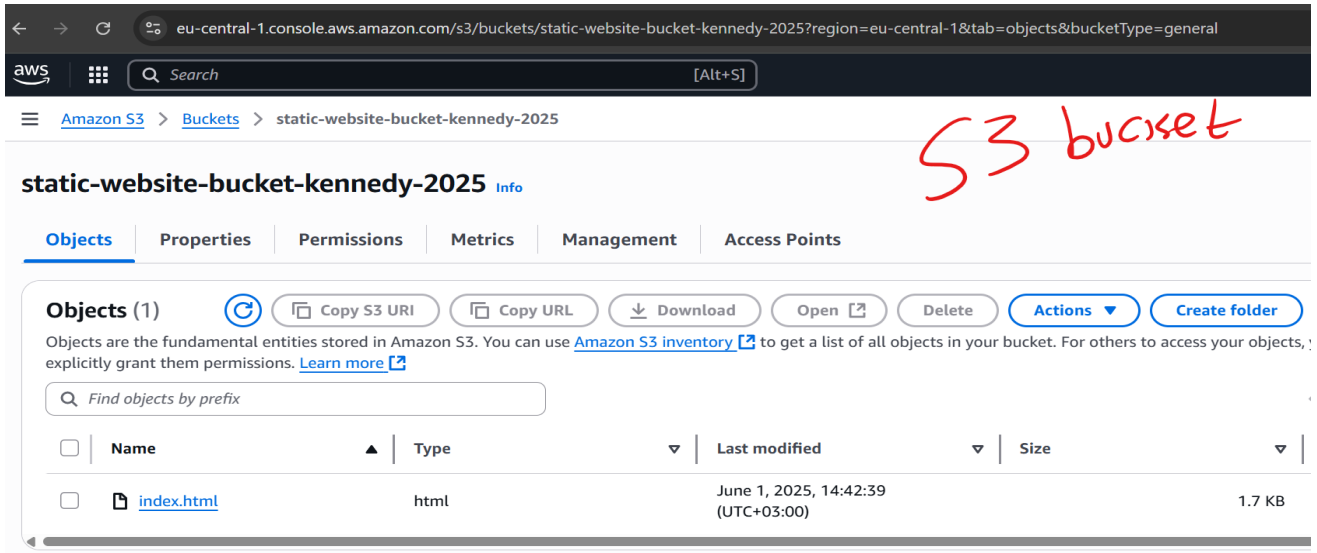


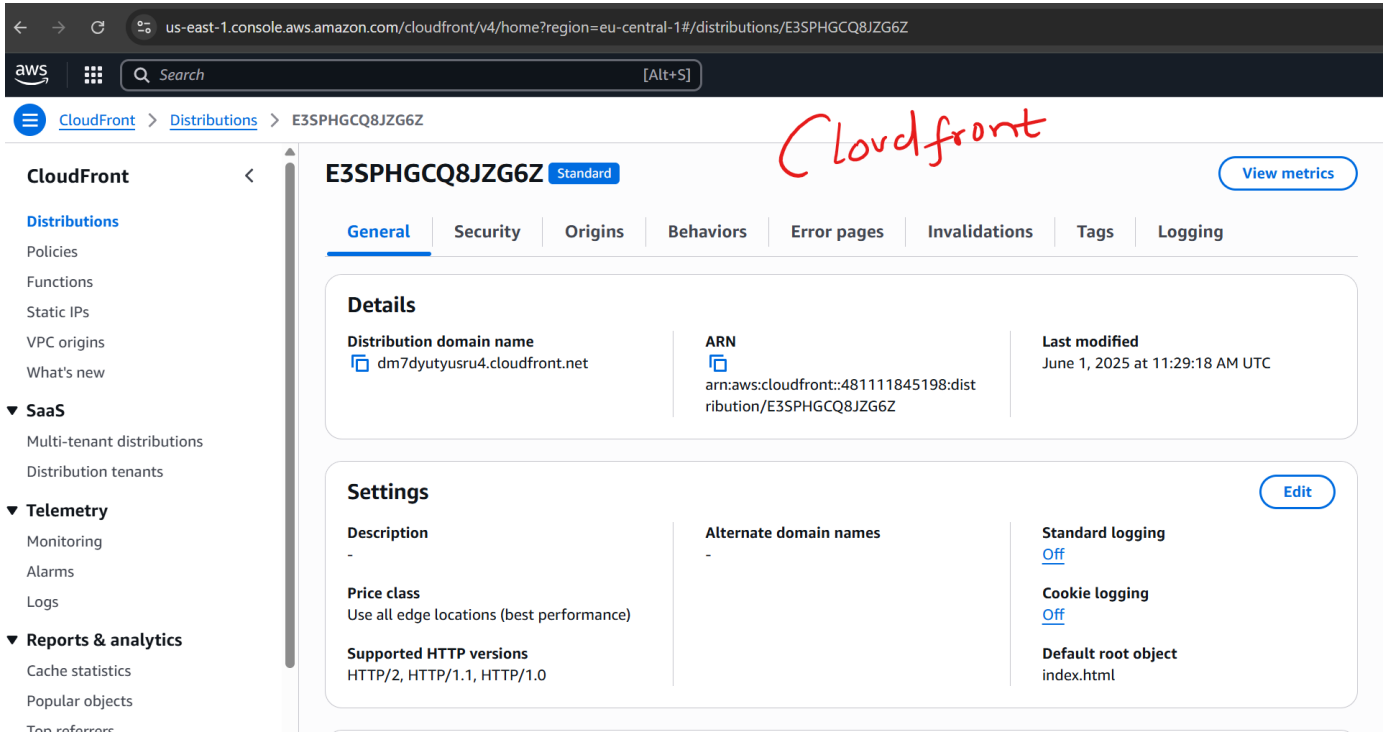
- Content explains the project
- Hosted live on CloudFront
- Uses clean, professional HTML layout

Deployment Screenshots

```
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # aws_s3_object.html is tainted, so must be replaced
-/+ resource "aws_s3_object" "html" {
      + acl                    = (known after apply)
      ~ arn                    = "arn:aws:s3:::static-website-bucket-kennedy-2025/index.html" -> (known after apply)
      ~ bucket_key_enabled     = false -> (known after apply)
      + checksum_crc32         = (known after apply)
      + checksum_crc32c        = (known after apply)
      + checksum_crc64nvme     = (known after apply)
      + checksum_sha1          = (known after apply)
      + checksum_sha256        = (known after apply)
      ~ etag                   = "cfdbe74704ef3dcbc13a7bc01ba8b6b1" -> (known after apply)
      ~ id                     = "index.html" -> (known after apply)
      + kms_key_id             = (known after apply)
      - metadata               = {} -> null
      ~ server_side_encryption = "AES256" -> (known after apply)
      ~ storage_class          = "STANDARD" -> (known after apply)
      - tags                   = {} -> null
      ~ tags_all               = {} -> (known after apply)
      + version_id             = (known after apply)
        # (13 unchanged attributes hidden)
    }

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

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

  Enter a value: yes

aws_s3_object.html: Destroying... [id=index.html]
aws_s3_object.html: Destruction complete after 0s
aws_s3_object.html: Creating...
aws_s3_object.html: Creation complete after 1s [id=index.html]

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

Outputs:

cloudfront_url = "https://dm7dyutyusru4.cloudfront.net"
website_url = "http://static-website-bucket-kennedy-2025.s3-website.eu-central-1.amazonaws.com"
```

Terraform apply Output





Final Result

```
aws_s3_object.html: Destroying... [id=index.html]
aws_s3_object.html: Destruction complete after 0s
aws_s3_object.html: Creating...
aws_s3_object.html: Creation complete after 1s [id=index.html]

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

Outputs:

cloudfront_url = "https://dm7dyutyusru4.cloudfront.net"
website_url = "http://static-website-bucket-kennedy-2025.s3-website.eu-central-1.amazonaws.com"
```

https://dm7dyutyusru4.cloudfront.net

Cloud Programming Project – Static Website Deployment

This website was deployed using **AWS S3, CloudFront, and Terraform** as part of the DLBSEPCP01_E portfolio assignment.

✅ What This Project Demonstrates

- Infrastructure-as-Code (IaC) using `Terraform`
- Hosting a static website on `AWS S3`
- Global content delivery via `AWS CloudFront`
- Secure HTTPS delivery using `CloudFront default SSL`

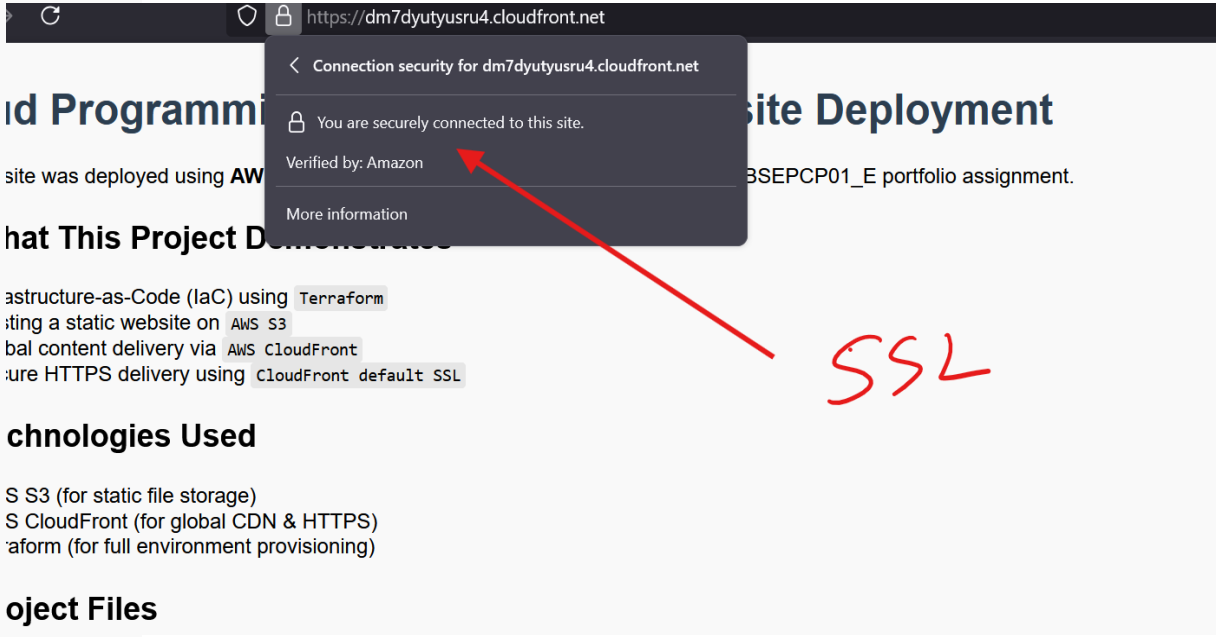
🌎 Technologies Used

- AWS S3 (for static file storage)
- AWS CloudFront (for global CDN & HTTPS)
- Terraform (for full environment provisioning)

📁 Project Files

- `main.tf` – Infrastructure definitions
- `variables.tf` – AWS region and project name
- `outputs.tf` – Display URLs after deployment
- `index.html` – This web page

Built with 💔 for IU Cloud Programming Portfolio (Task 1)



Challenges

- Initial attempt at Lambda + Docker failed (bootstrap errors)
- Pivoted to S3+CloudFront for simplicity and reliability
- Learned value of adapting when roadblocks arise

Lessons Learned

- Practical Terraform deployment
- Cloud resource configuration and IAM
- Simplicity often beats complexity
- Confidence using AWS for real-world scenarios

Resources / Hyperlinks to the frameworks used

- **HTML Page Purpose** - Hyperlink
- **S3 URI:** s3://static-website-bucket-kennedy-2025/index.html
- **Amazon Resource Name (ARN):** arn:aws:s3:::static-website-bucket-kennedy-2025/index.html
- **Entity tag (eTag):** cd3543668ffc7be94f2c2410f5575291
- **Object URL:** https://static-website-bucket-kennedy-2025.s3.eu-central-1.amazonaws.com/index.html
- Github Project Link