Development and Reflection Phase:

Hosting a Simple Library Opening Web Application on AWS

Student: Dikshya Khatri

Matriculation Number: 10235413

Date: 31/12/2024

Project Overview:

1. Project Objective:

- user-friendly web application
- Ensure a smooth, accessible user experience
- Support AWS services for hosting, scalability, and security.

2. Core Features of Webpage:

- Display Library Announcements and Events.

Project Overview:

3. Technical Requirements:

- **Hosting Platform:** Deploy on AWS using:

  Amazon S3: Host static website content (HTML & CSS) (AWS, 2024).

  Amazon CloudFront: Enable content delivery for improved global performance (AWS, 2024b).

- **Infrastructure as Code (IaC):**

  Use Terraform to manage AWS infrastructure (HashiCorp Cloud Platform, n.d.).

  Terraform Code for Infrastructure Setup:

  Access the main.tf File: GitHub – Dikshya Khatri

Project Overview:

4. Security and Compliance:

- Bucket Policy ensures that only CloudFront can fetch content from the S3 bucket (AWS, 2024c) .
- Use of OAI to further secure S3 access, allowing only CloudFront to retrieve content (AWS, 2024c).

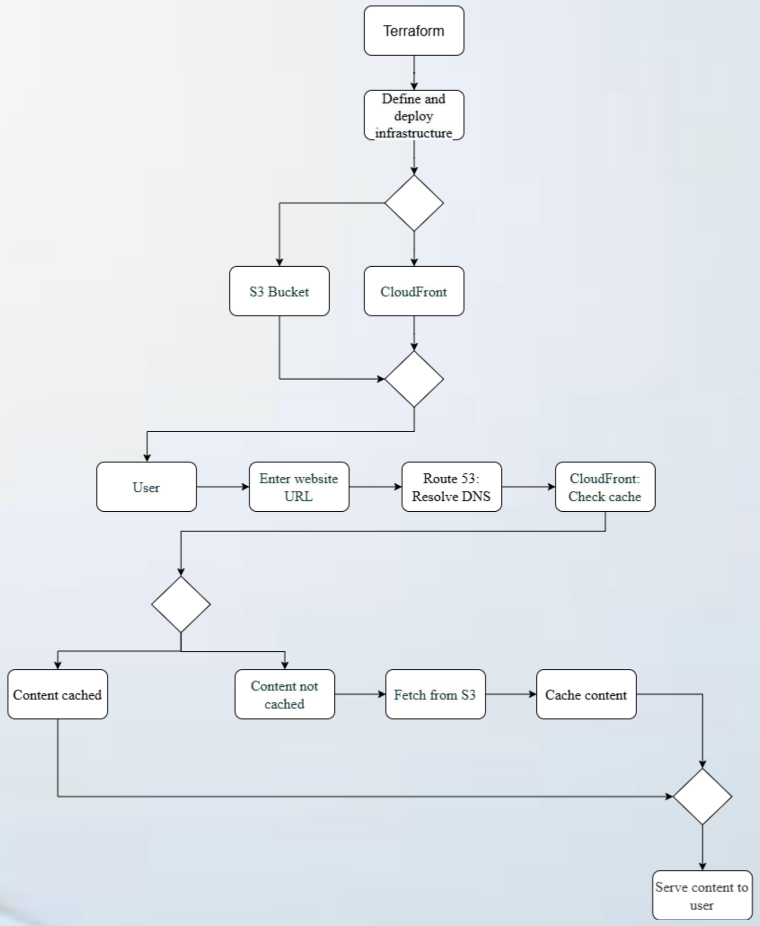
5. Scalability and High Availability:

- Deploy static assets on **S3** with **CloudFront** for global availability (AWS, 2024c).

6. Expected Outcomes:

- An easy-to-navigate web application for library users.
- Enhanced performance and global reach through AWS services

Architecture Diagram



Architecture Explanation:

- Use of Terraform to automate AWS infrastructure setup (HashiCorp Cloud Platform, n.d.),
- S3 for static content storage (AWS, 2024a). ,
- CloudFront for global content caching and delivery and (AWS, 2024b),
- secure access to the website for users (AWS, 2024c).

Deployment

Terraform Implementation:

- Using Terraform scripts to automate S3 and CloudFront setup for hosting (HashiCorp Cloud Platform, n.d.).

AWS Resource Configuration:

- S3 Bucket: Stores static files (e.g., index.html, and images) (AWS, 2024a).
- CloudFront caches content at edge locations for faster user access (AWS, 2024b).
- Relying on the CloudFront URL for webpage access (AWS, 2024b).

Testing

- Directly accessing the webpage using the **CloudFront distribution URL** to test content delivery and caching efficiency (AWS, 2024b).
- Verifying that CloudFront retrieves files from S3 when content is not cached (AWS, 2024a)..

Successes:

- Achieved high availability and low latency for the webpage through CloudFront (AWS, 2024b).
- Successfully automated infrastructure deployment using Terraform (HashiCorp Cloud Platform, n.d.).

Challenges:

- Only relying on CloudFront may affect ease of access and user experience (AWS, 2024b).
- Ensuring proper S3 bucket policies to prevent unauthorized access (AWS, 2024a).

Improvements:

- To improve user experience, consider integrating a third-party DNS provider (AWS, 2024c).

Important links

Link to CloudFront URL:

CloudFront Webpage Domain

Link to the project on GitHub:

Hosting a Simple Webpage on AWS

Conclusion

- The project successfully demonstrated the deployment of a user-friendly library web application hosted on AWS, leveraging Amazon S3 for static content storage and Amazon CloudFront for global content delivery.

- Utilizing Terraform for infrastructure automation ensured an efficient and repeatable setup process.

- Overall, the project highlights the potential of AWS services for scalable, secure, and globally accessible web hosting solutions.

Reference

- AWS. (2024a). *What is Amazon S3?* AWS. Retrieved from

What is Amazon S3? - Amazon Simple Storage Service

- AWS. (2024b). *What is Amazon CloudFront?* AWS. Retrieved from

What is Amazon CloudFront? - Amazon CloudFront

- AWS. (2024c). *Restrict access to an Amazon Simple Storage Service origin. Amazon Web Services*. Retrieved from

Restrict access to an Amazon Simple Storage Service origin - Amazon CloudFront

- HashiCorp Cloud Platform. (n.d.). *What is Terraform?* HashiCorp. Retrieved from

What is Terraform | Terraform | HashiCorp Developer