Hosting a Static Portfolio Website Using AWS S3 and CloudFront

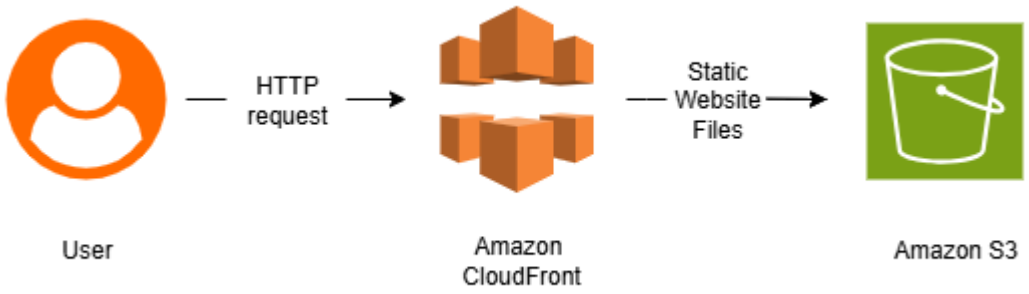
Overview

In this project, I hosted my personal portfolio website on Amazon Web Services (AWS) using S3 for storage and CloudFront for content delivery with HTTPS enabled by default. This was my first AWS cloud project using the AWS Free Tier.

The main objective was to:

- Store and host a static website in Amazon S3.
- Distribute it globally via Amazon CloudFront CDN.
- Ensure secure access using the default CloudFront SSL certificate.
- Manage costs efficiently by cleaning up resources after testing.

Architecture Diagram



- user makes https request
- cloudfront serves cached content or fetches from s3
- s3 hosts the static files

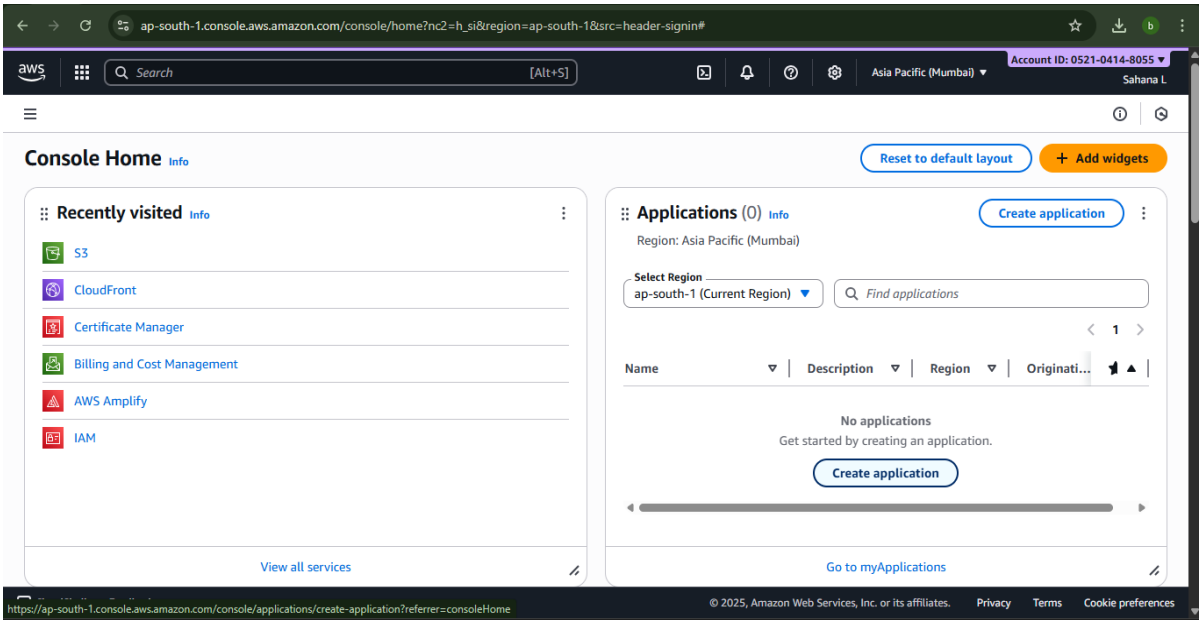
AWS Services Used

- **Amazon S3** – to store and host static website files.
- **Amazon CloudFront** – to serve content securely and globally with caching.
- **AWS IAM** – to manage access permissions.
- **AWS Budgets** – to set up billing alerts to avoid charges.

Step-by-Step Implementation

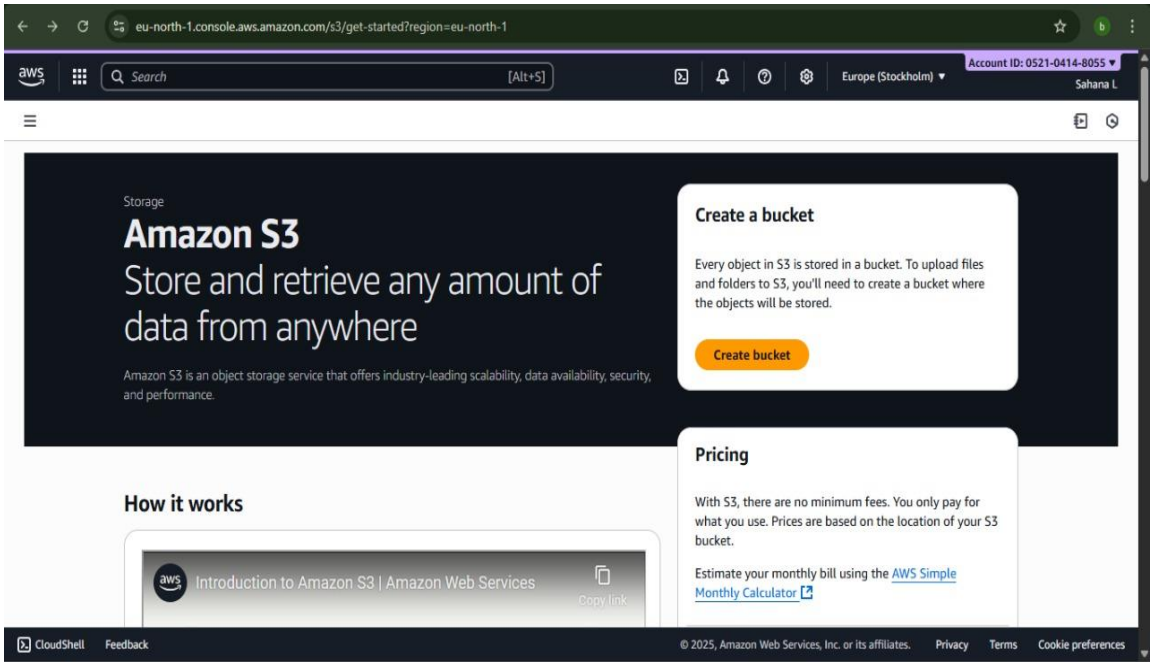
1. Login to AWS Console

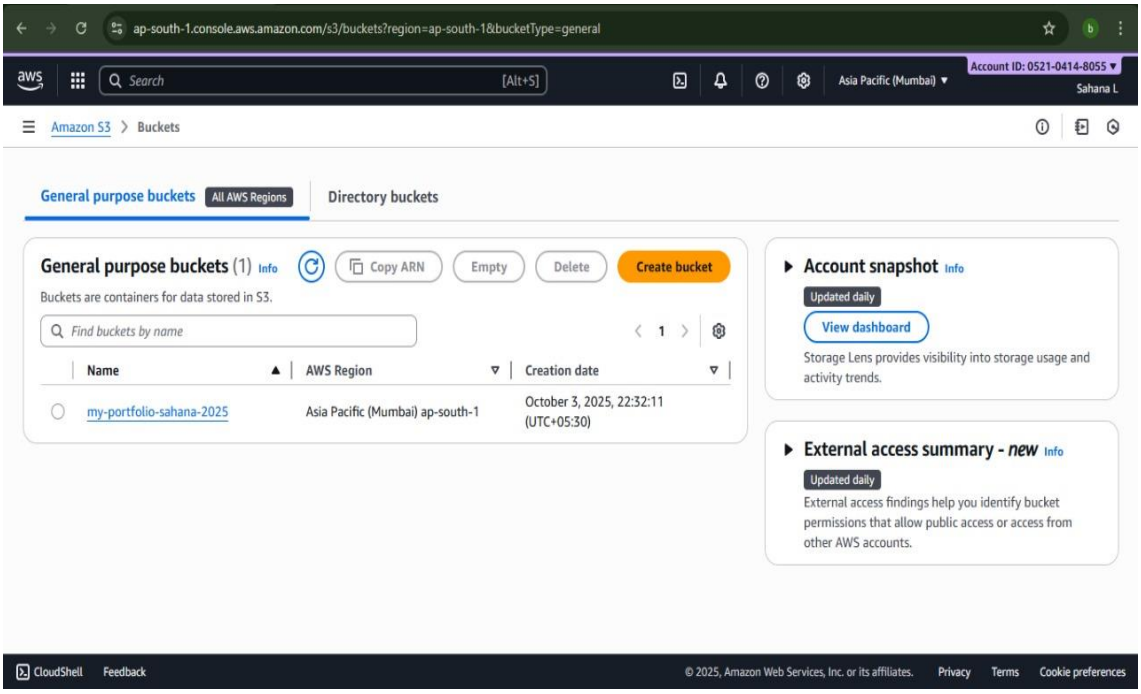
- Logged in to AWS Management Console.



2. Create an S3 Bucket

- Opened **S3 Service** clicked **Create Bucket**.
- Gave bucket a unique name (`my-portfolio-2025`).
- Disabled “Block Public Access”.





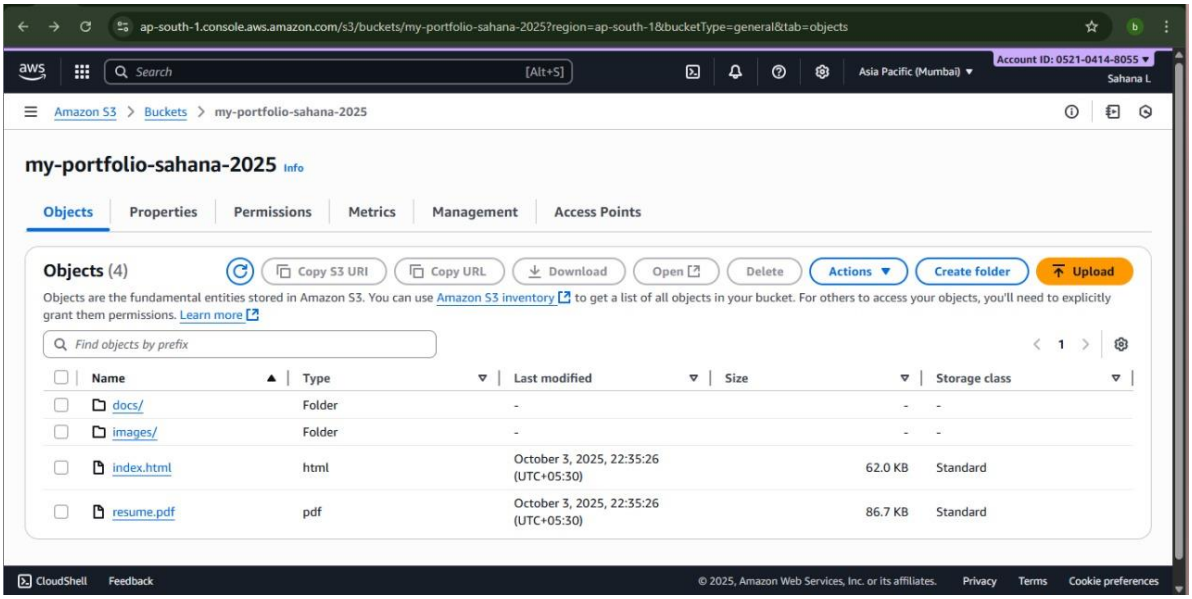
3. Upload Portfolio Files

- Renamed portfolio file to `index.html`.

- Created folder structure:

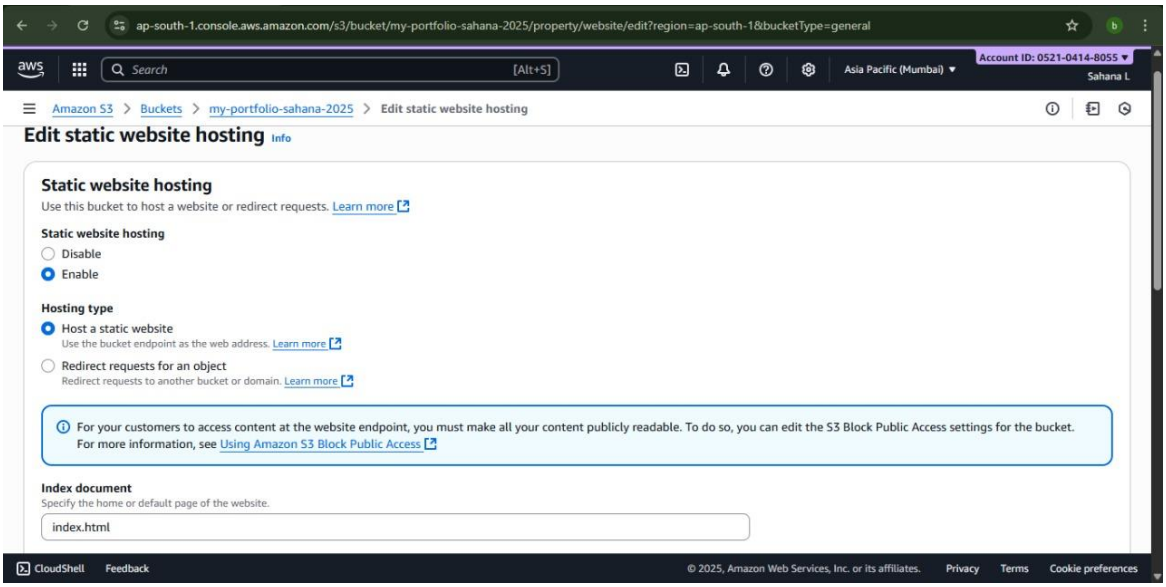
  Index.html, /images, /docs

- Uploaded files into the S3 bucket.



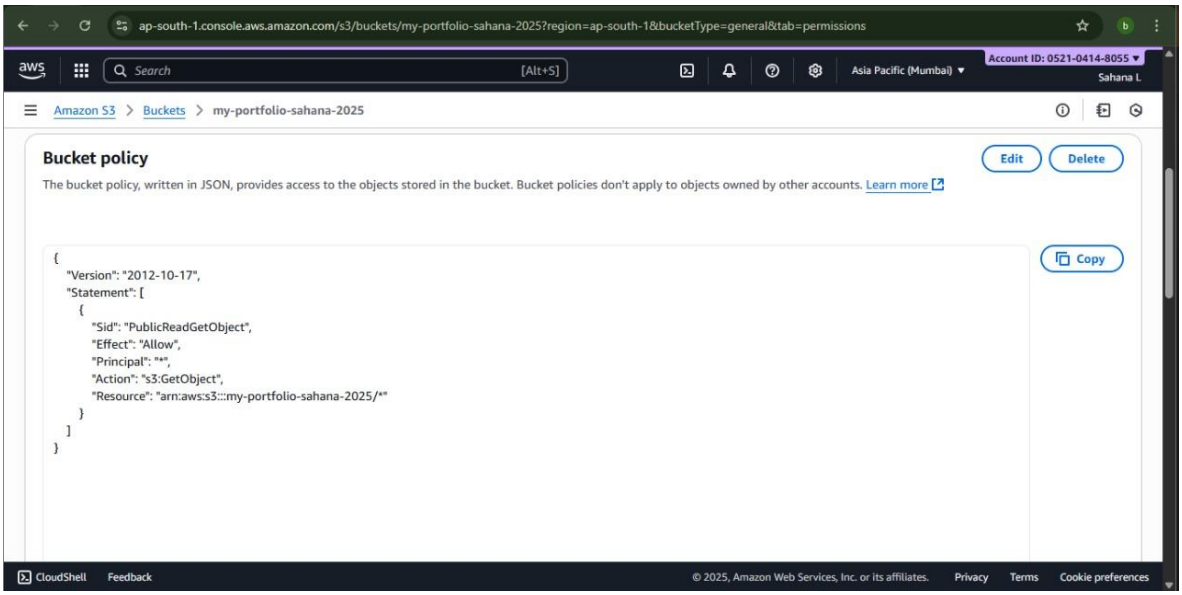
4. Enable Static Website Hosting

- Went to **Properties** tab → Enabled **Static website hosting**.
- Set index document to `index.html`.



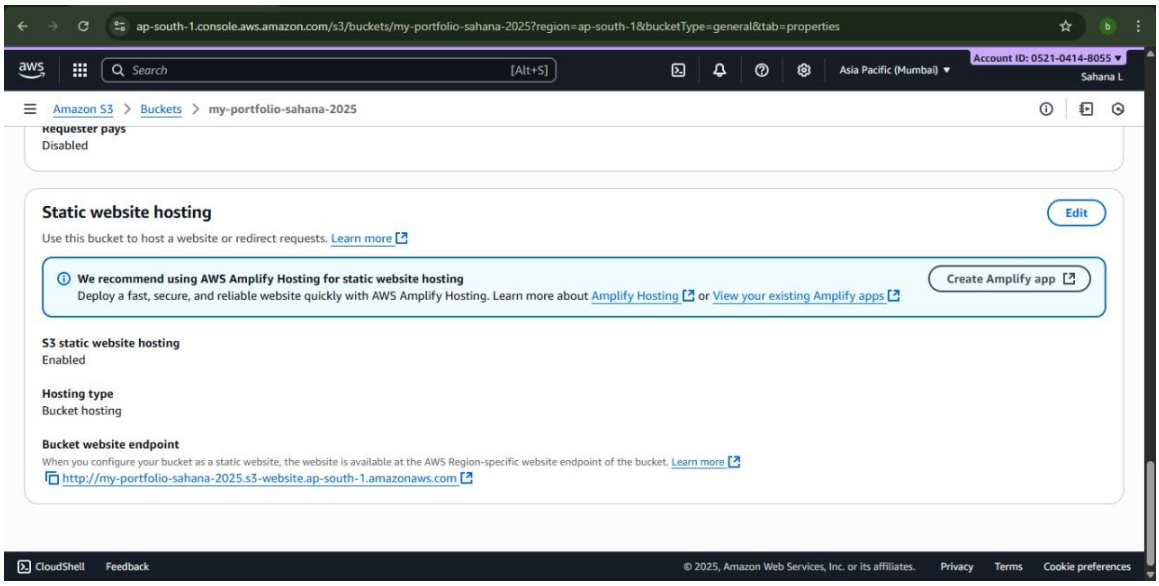
5. Update Bucket Policy for Public Access (Temporary)

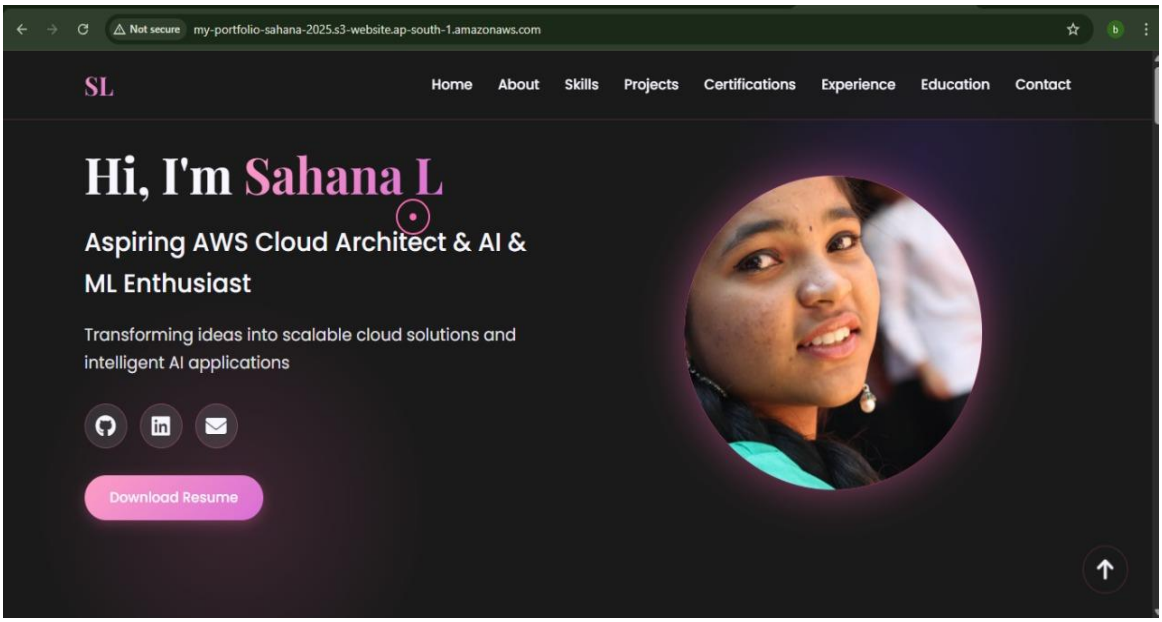
Before testing the S3 site, a **temporary bucket policy** was added to make the files public.



6. Test Website with S3 Endpoint

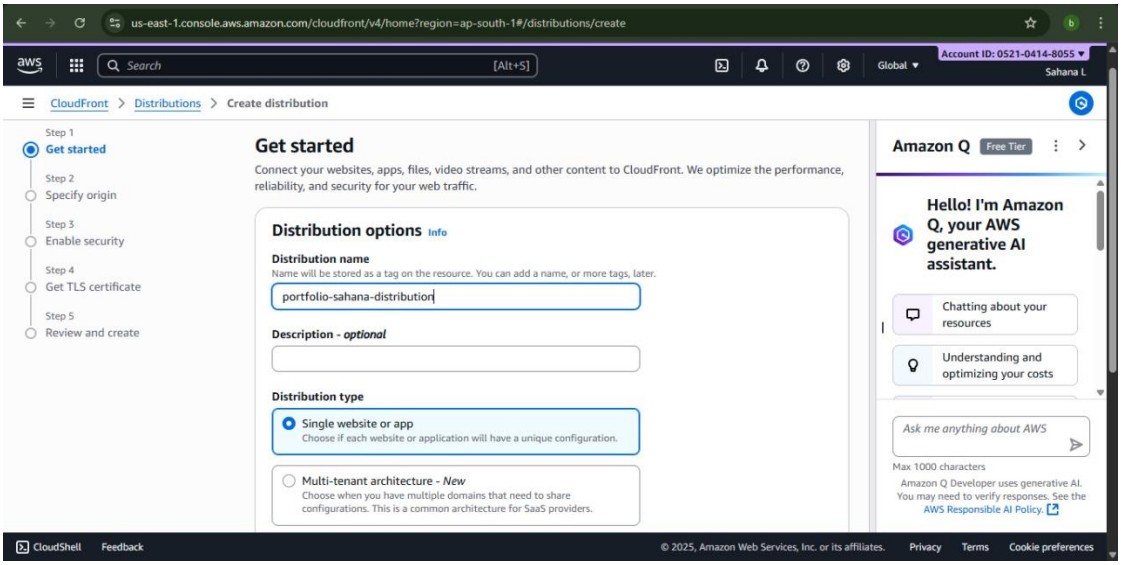
- Copied S3 Website Endpoint and opened in browser.
- Website loaded successfully.

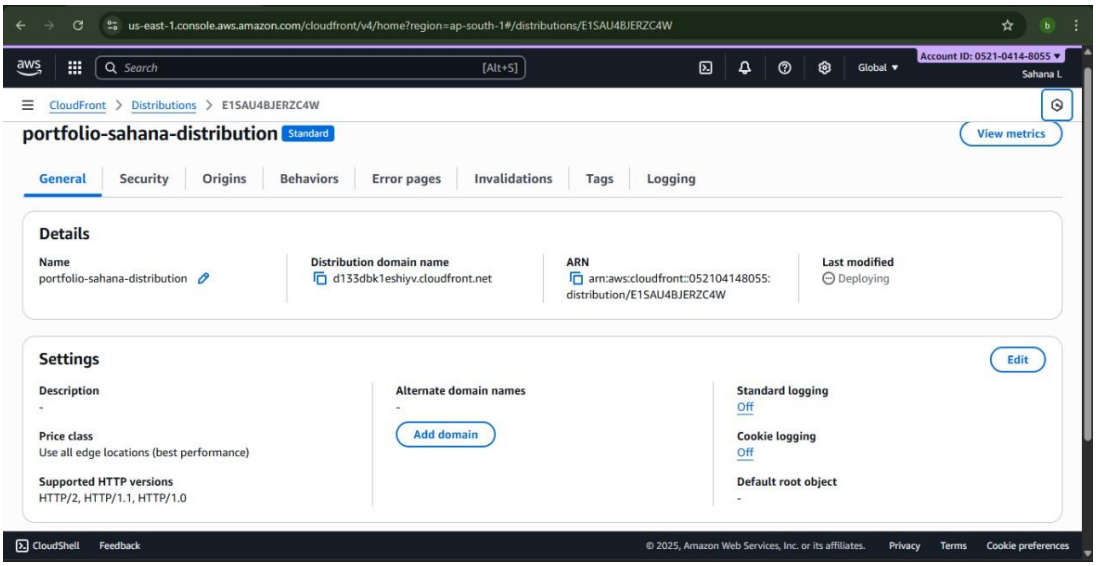


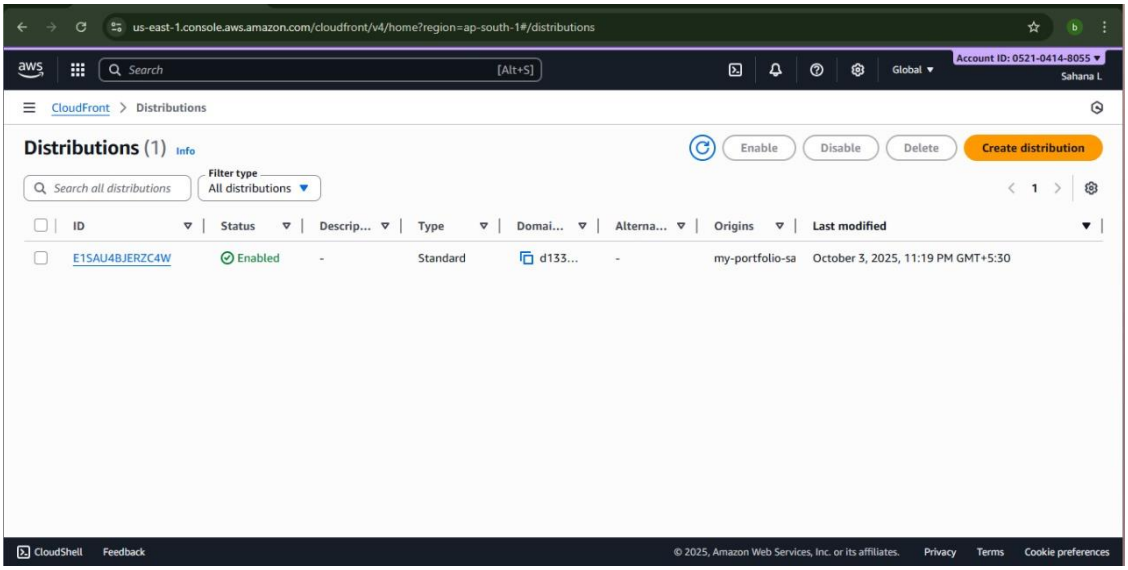


7. Create CloudFront Distribution

- Opened **CloudFront Service** → Clicked **Create Distribution**.
- Chose the S3 bucket as the **origin**.
- Set **Viewer Protocol Policy** → "Redirect HTTP to HTTPS".
- Used the **default CloudFront SSL certificate**.
- Clicked **Create Distribution**.

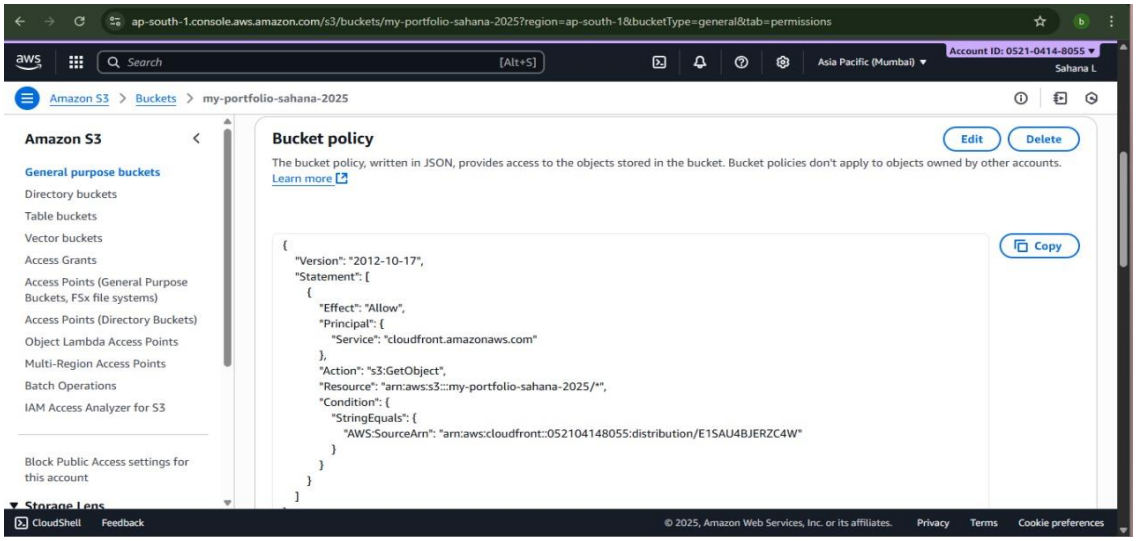

us-east-1.console.aws.amazon.com/cloudfront/v4/home?region=ap-south-1#/distributions/create
Search
[Alt+S]
Global
Account ID: 0521-0414-8055
Sahana L
CloudFront > Distributions > Create distribution
Step 1
Get started
Step 2
Specify origin
Step 3
Enable security
Step 4
Get TLS certificate
Step 5
Review and create
Get started
Connect your websites, apps, files, video streams, and other content to CloudFront. We optimize the performance, reliability, and security for your web traffic.
Distribution options Info
Distribution name
Name will be stored as a tag on the resource. You can add a name, or more tags, later.
portfolio-sahana-distribution
Description - optional
Distribution type
Single website or app
Choose if each website or application will have a unique configuration.
Multi-tenant architecture - New
Choose when you have multiple domains that need to share configurations. This is a common architecture for SaaS providers.
Amazon Q
Free Tier
Hello! I'm Amazon Q, your AWS generative AI assistant.
Chatting about your resources
Understanding and optimizing your costs
Ask me anything about AWS
Max 1000 characters
Amazon Q Developer uses generative AI. You may need to verify responses. See the AWS Responsible AI Policy.
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


us-east-1.console.aws.amazon.com/cloudfront/v4/home?region=ap-south-1#/distributions/E1SAU4BJERZC4W
Search
[Alt+S]
Global
Account ID: 0521-0414-8055
Sahana L
CloudFront > Distributions > E1SAU4BJERZC4W
portfolio-sahana-distribution
Standard
View metrics
General
Security
Origins
Behaviors
Error pages
Invalidations
Tags
Logging
Details
Name
portfolio-sahana-distribution
Distribution domain name
d133dbk1eshiyv.cloudfront.net
ARN
arn:aws:cloudfront::052104148055:distribution/E1SAU4BJERZC4W
Last modified
Deploying
Settings
Edit
Description
-
Price class
Use all edge locations (best performance)
Supported HTTP versions
HTTP/2, HTTP/1.1, HTTP/1.0
Alternate domain names
-
Add domain
Standard logging
Off
Cookie logging
Off
Default root object
-
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



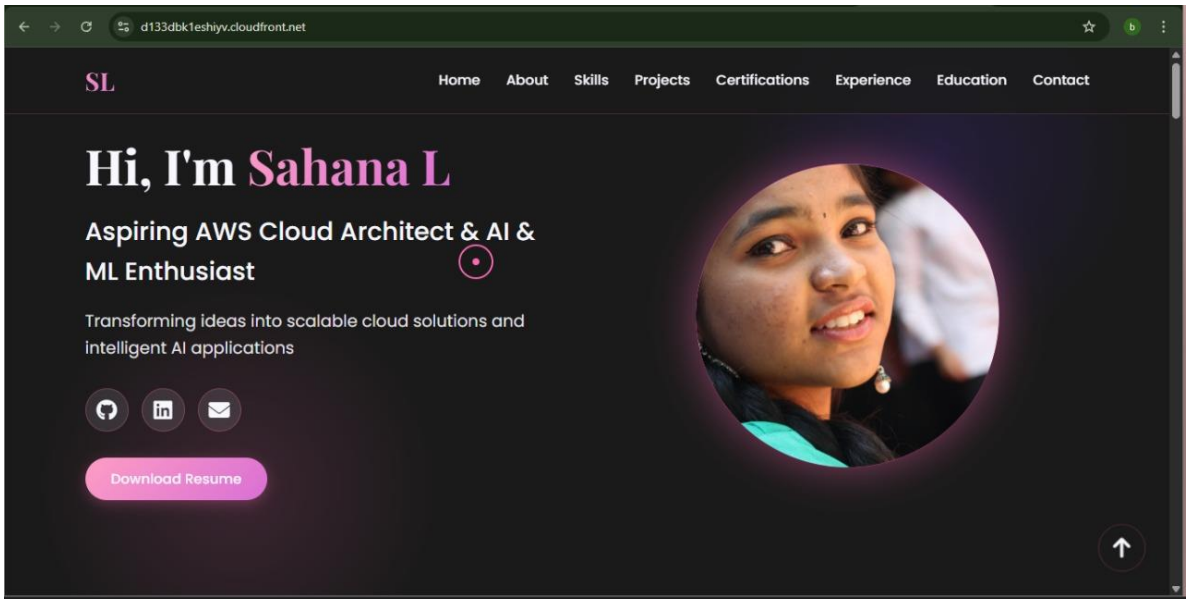
8. Update Bucket Policy for CloudFront Access (Secure Setup)

Once CloudFront is created, replace the public access policy with a **CloudFront Origin Access Policy**.
This ensures only CloudFront can access the bucket — keeping it secure.



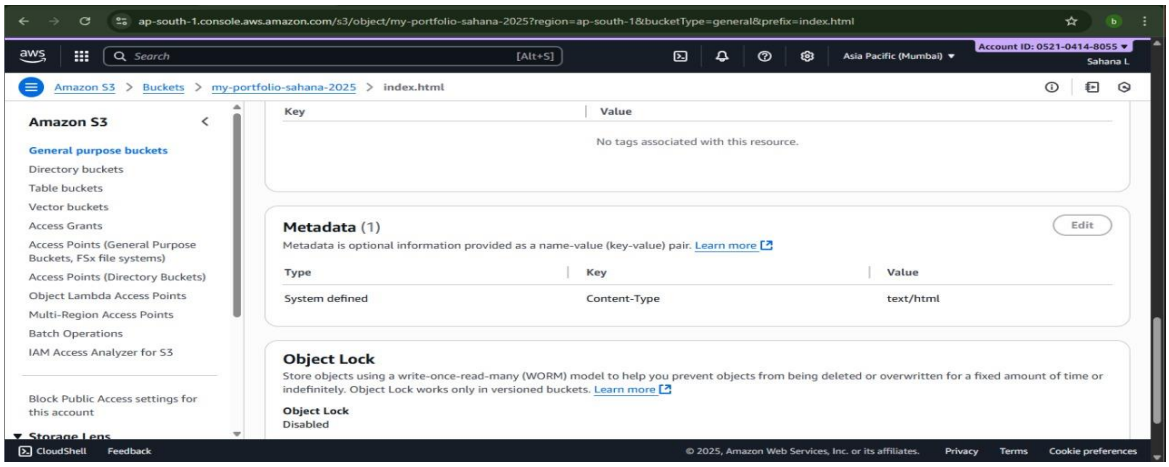
9. Access Website via CloudFront (HTTPS)

- Once deployed, accessed portfolio via https://d133dbk1eshiyv.cloudfront.net.



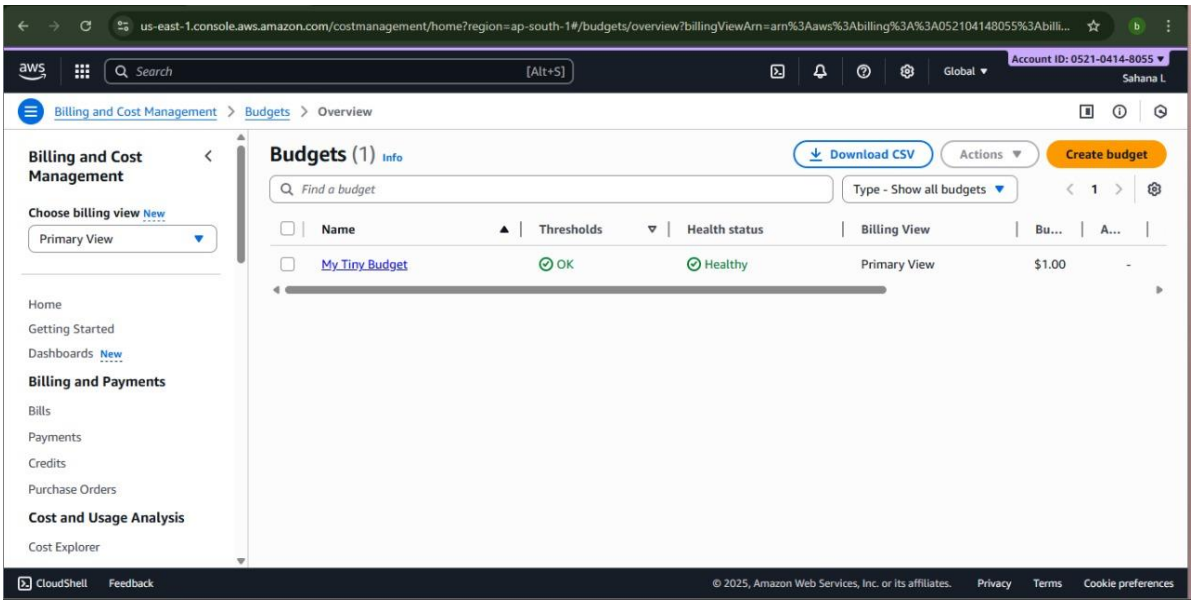
10. (Optional) Object Metadata Check

- Verified content type for `index.html` is `text/html`.



11. Setup Budget Alert

- Configured AWS Budget to stay within **Free Tier**.



12. Cleanup Resources

- Disabled & deleted CloudFront distribution.
- Emptied and deleted S3 bucket.

Results

- Successfully hosted my static portfolio website using **S3 + CloudFront**.
- Website accessible globally with **HTTPS**.
- Ensured **zero extra cost** by deleting resources after project completion.

Learnings

- Hands-on with AWS **S3**, **CloudFront**, **IAM Policies**, and **Budgets**.
- Understood static website hosting and CDN distribution.
- Learned importance of **cost monitoring** in cloud environments.

Conclusion

This project demonstrated how to build and deploy a static website on AWS using **S3 for storage** and **CloudFront for secure global delivery**. By carefully configuring bucket policies, enabling HTTPS, and setting up billing alerts, I ensured the project was both **secure** and **cost-efficient within the AWS Free Tier**.

The solution is **scalable, secure, and industry-aligned**, showing a solid foundation in AWS cloud architecture practices. This project reflects my ability to design, implement, and document cloud-based solutions that can be extended to more complex real-world architectures.