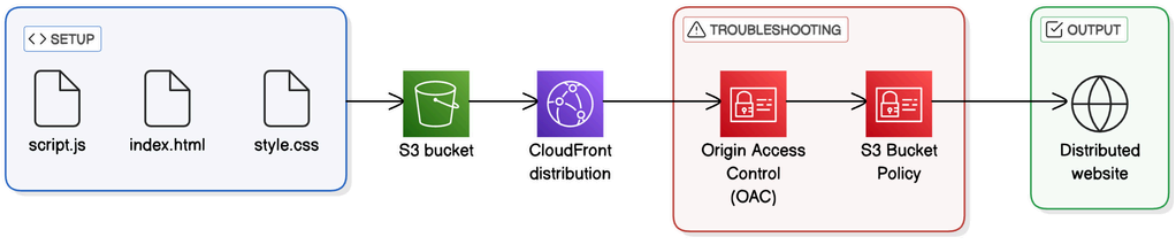
Kareshma
Rajaananthapadmanaban

AWS Three-tier series

Part 1

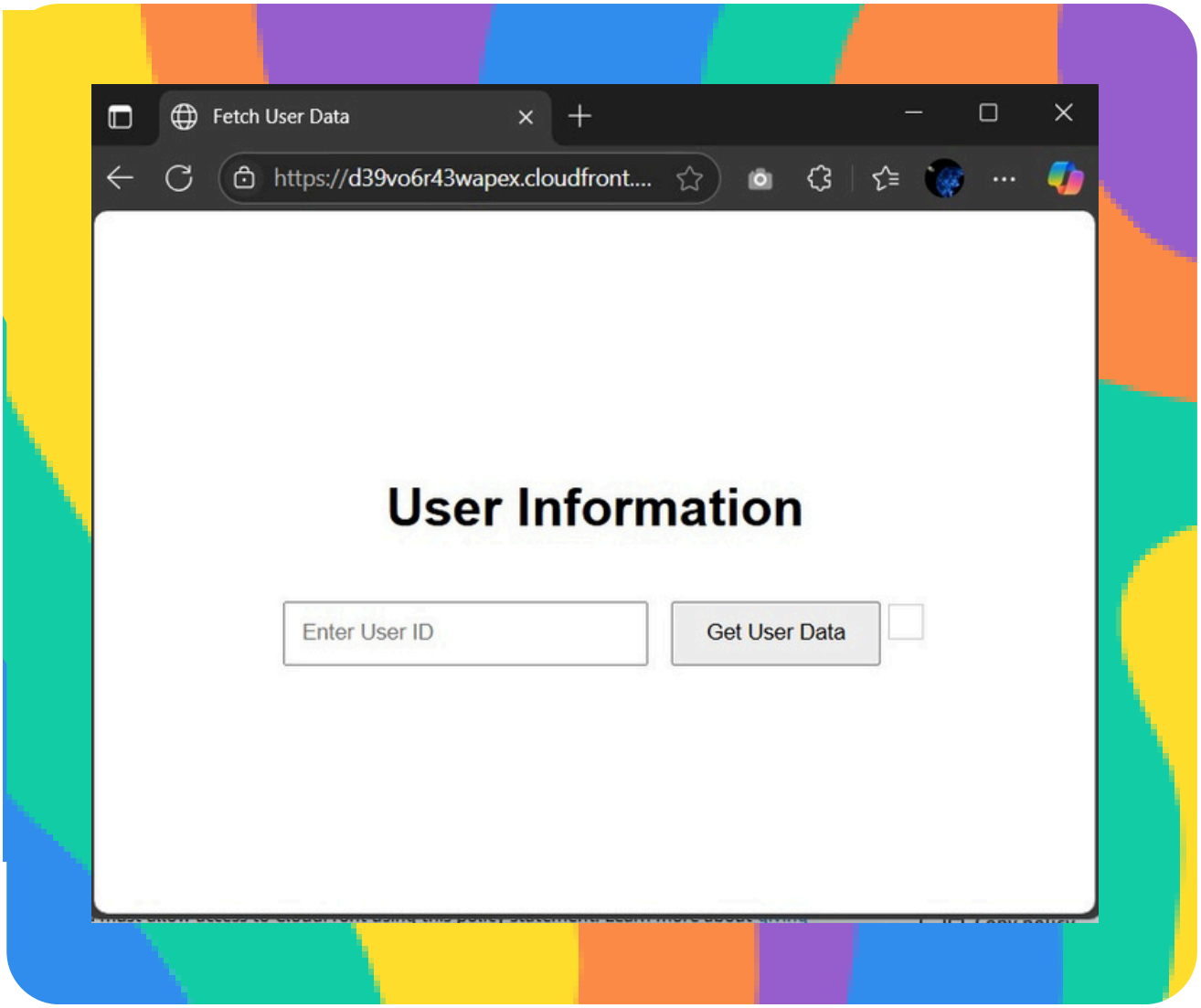
A hands-on project series where I host a static website on AWS using S3 and CloudFront, exploring performance, security, and real-world deployment



nextwork.org

Website Delivery with CloudFront

Kareshma Rajaananthapadmanaban



Introducing Today's Project!

In this project, I will demonstrate how to deploy a **static website** using Amazon S3 and distribute it globally with **Amazon CloudFront**. I'm doing this project to learn how to deliver web content efficiently, manage permissions securely, and gain practical experience in building the **presentation layer** of a three-tier web architecture.

Tools and concepts

Services I used were **Amazon S3** and **Amazon CloudFront**. Key concepts I learnt include content delivery network (CDN), origin access control (OAC), S3 static website hosting, bucket policies, and caching behavior. I also practiced performance comparison, secure access management, and developer tools for analyzing website load times.

Project reflection

This project took me approximately less than an hour to complete. The most challenging part was troubleshooting the **access denied errors** while configuring S3 static website hosting and understanding how Origin Access Control works with **bucket policies.** t was most rewarding to see my website successfully delivered through CloudFront and optimized for global performance.

I did this project to deepen my understanding of how static websites are delivered securely and efficiently using AWS services. It helped me **connect S3 storage** with **CloudFront delivery** and reinforced my knowledge of permissions, caching, and performance optimization. Yes, this project met my goals by enhancing both my practical skills and cloud architecture confidence.

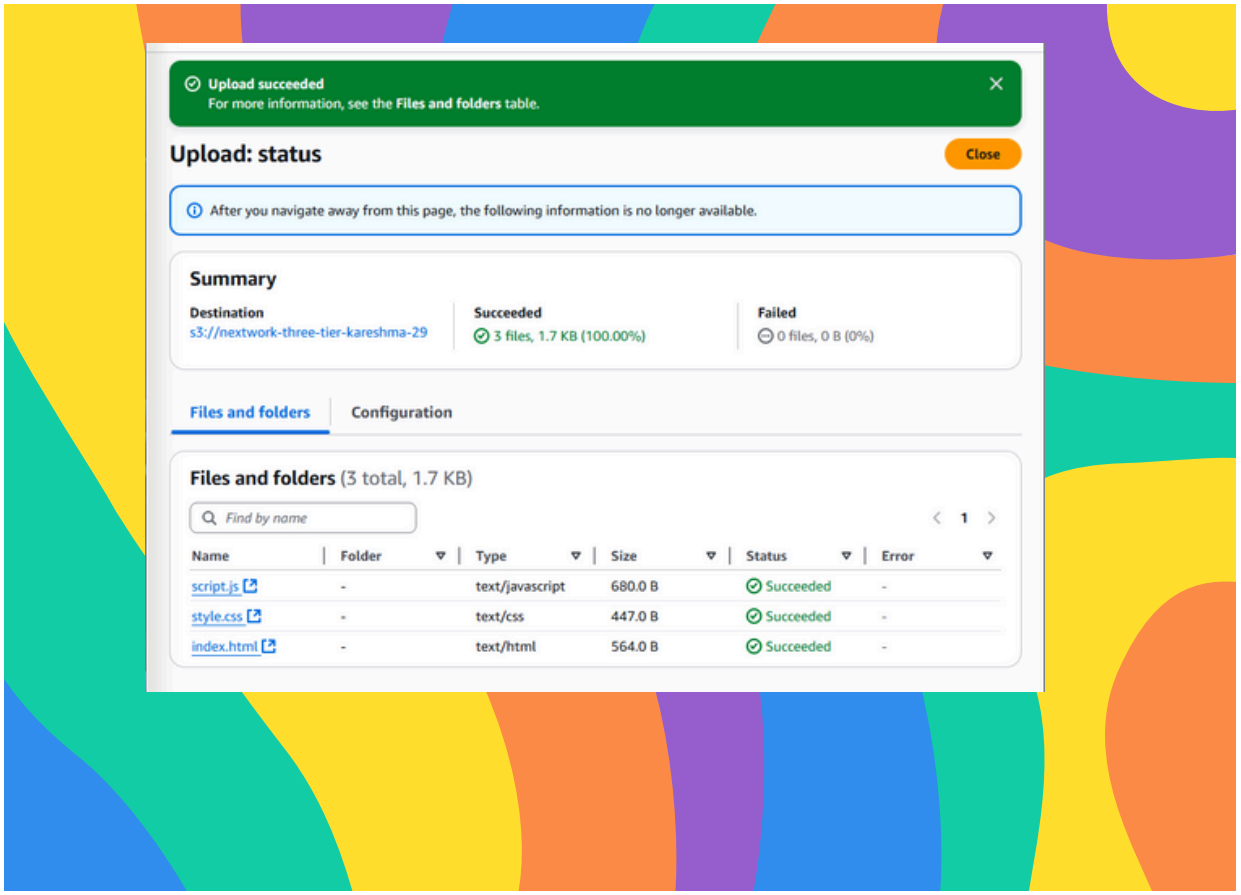
Set Up S3 and Website Files

In this step, I will create a **new S3 bucket** because it will serve as the origin for my static website files. S3 provides durable, scalable object storage, making it ideal for hosting HTML, CSS, JS, and other assets. This bucket acts as the foundation for storing and distributing my site via CloudFront.

I started the project by creating an S3 bucket to store and serve my static website files like HTML, CSS, and JavaScript. I can't use CloudFront for this task because CloudFront is a content delivery network, not a storage service. It needs an **origin source,** and **S3** acts as that reliable and scalable origin for website content.

The three files that make up my website are `***index.html***`, which defines the structure and content of the webpage; `***style.css***`, which adds styling and visual formatting to the page; and `***script.js***`, which handles the client-side logic and interactivity of the website when opened in a browser.

I validated that my website files work by opening the `index.html` file directly in my browser. I checked that the layout rendered correctly, the styles from `style.css` were applied, and the interactive elements controlled by `script.js` responded as expected. This confirmed the **files were functional** before uploading to S3.



Uploaded Objects


Kareshma
Rajaananthapadmanaban

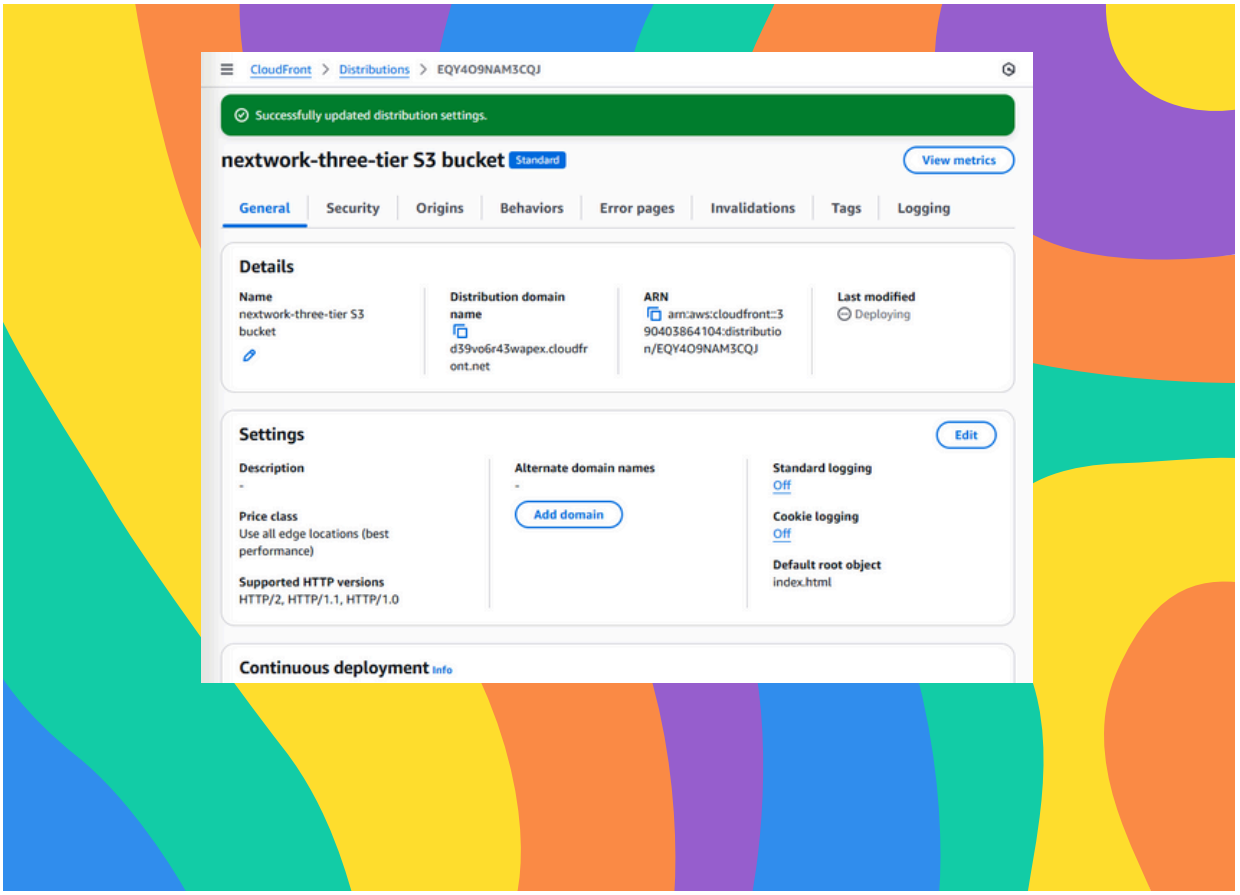
Exploring Amazon CloudFront

In this step, I will **set up** a **CloudFront distribution** because it allows my website to be delivered quickly and securely to users around the world. CloudFront caches content at edge locations, **reducing latency** and improving performance. It also adds an extra layer of protection with HTTPS and access control features.

Amazon CloudFront is a **content delivery network**, which means it delivers web content like HTML, CSS, JS, and images faster to users by **caching** it at global edge locations. Businesses and developers use CloudFront because it **reduces latency**, improves website load times, enhances security with HTTPS, and scales automatically to handle traffic spikes.

To use Amazon CloudFront, you **set up distributions**, which are configuration sets that define how CloudFront delivers your content. I set up a distribution for my static website to ensure fast, global delivery. The **origin** is my **S3 bucket,** which stores the website files **CloudFront** will cache and serve through **edge locations.**

My CloudFront distributions' **default root object** is ***index.html***. This means when users visit the root URL of the site without specifying a file CloudFront will automatically serve index.html from the S3 bucket. I kept the **origin public**, left **cache behavior** and other advanced settings as default for a simple, fast, and secure setup.



Distribution's origin set up.

Handling Access Issues

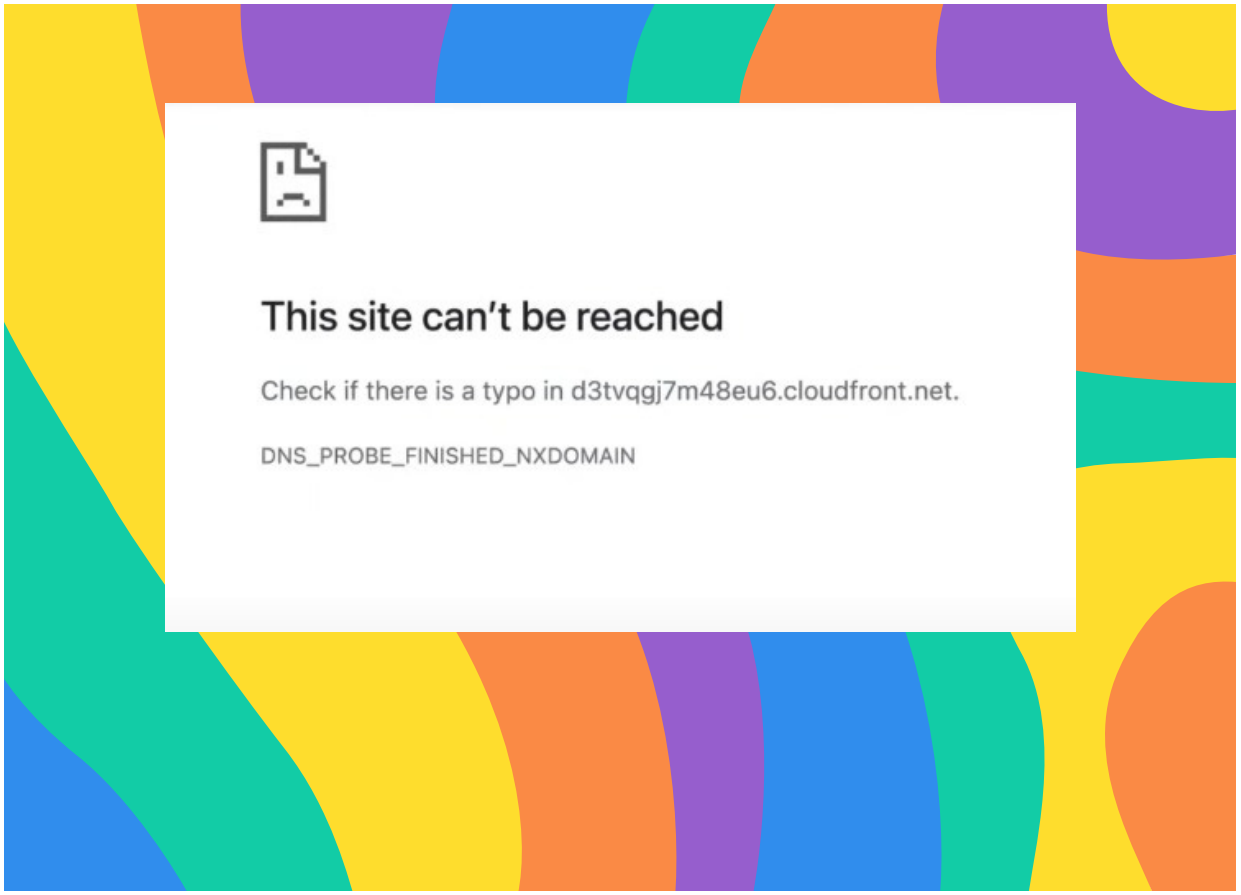
In this step, I will **verify my CloudFront distribution** because I need to confirm that the static website is correctly cached and served through the CDN. This ensures my configuration is working as expected and the content is **accessible globally** with improved speed and lower latency.

When I tried visiting my distributed website, I ran into an **access denied** error because my **S3 bucket**, which is the origin for CloudFront, is **private by default**. Although the bucket exists and contains the files, CloudFront doesn't have permission yet to access or serve them publicly through its distribution.

I will update my CloudFront origin settings because the current configuration **doesn’t allow** CloudFront to **access content** from my S3 bucket. Adjusting these settings is essential to grant proper permissions so CloudFront can fetch and serve the website files to users.

My distribution's **origin access settings** were set to **Public**. This caused the access denied error because, although CloudFront was allowed to access the origin, the objects inside my S3 bucket were still private. Without explicitly setting those files to public, CloudFront couldn’t retrieve or serve them to users.

To resolve the error, I set up origin access control (OAC). **OAC** is a secure method that **allows CloudFront** to **access** content in a **private S3 bucket** without making the files publicly accessible. It improves security by creating a special identity for CloudFront, giving it controlled access to the bucket while keeping the objects private from the public.



While verifying CloudFront Distribution

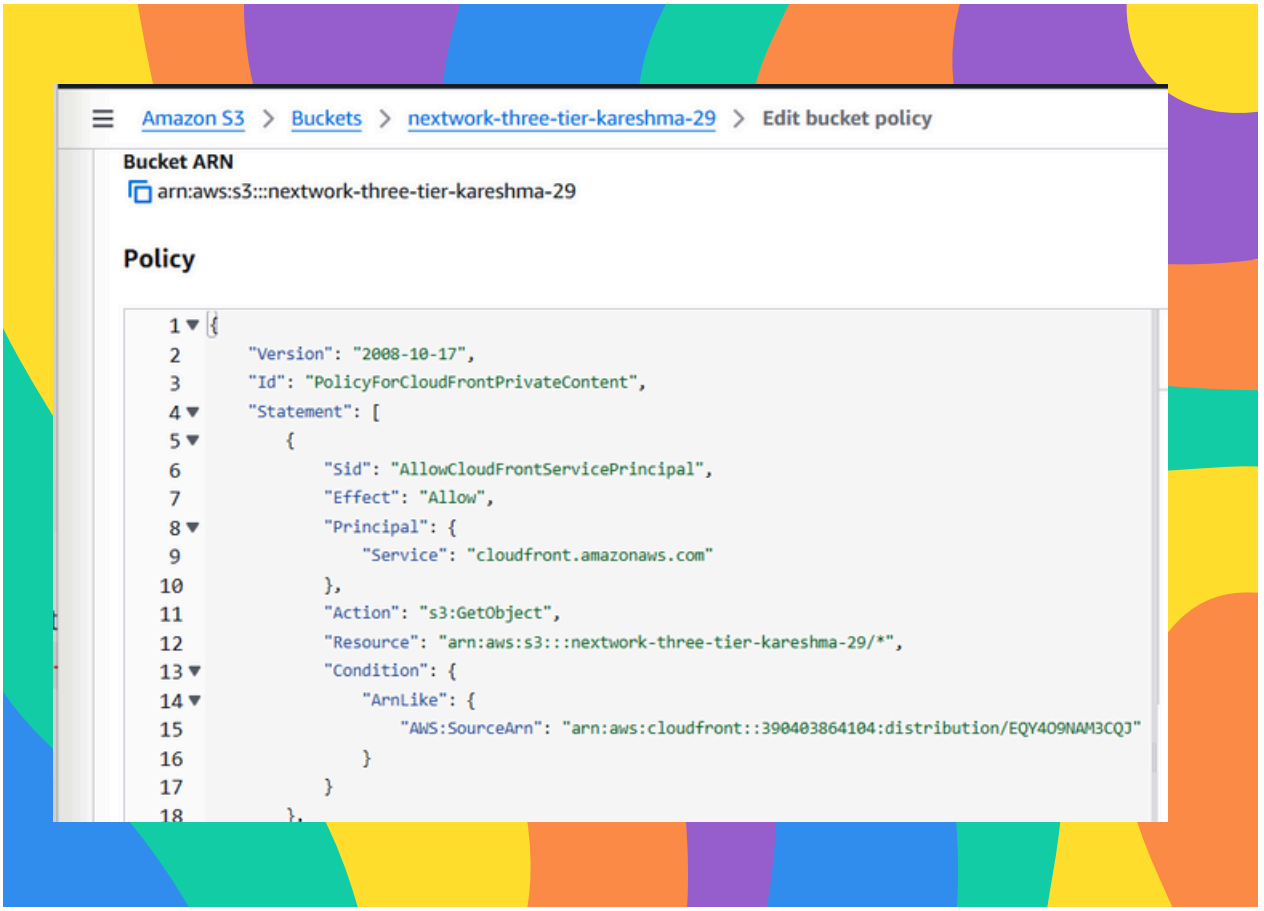
Updating S3 Permissions

Once I set up my OAC, I still needed to **update my bucket policy** because the OAC only defines how CloudFront will authenticate requests to the S3 bucket. For CloudFront to actually retrieve content, the **S3 bucket** must **explicitly grant permission** to the OAC identity through its policy. Without this, access remains denied.

I will update my S3 bucket's permission settings because the newly created Origin Access Control (OAC) needs explicit access to read the bucket's content. **Without updating** the bucket policy to allow CloudFront's OAC, the **CDN can't fetch** or serve the website files, even though the OAC is in place.

What does this policy do?

> Creating an OAC automatically gives me a policy I could copy, which grants CloudFront permission to access the objects in my S3 bucket. This policy ensures **only** the CloudFront distribution **associated with the OAC** can read the content, keeping the bucket private while still allowing secure content delivery through CloudFront.



Bucket's policy.


Kareshma
Rajaananthapadmanaban

Project Note: My Updated Setup Experience

While following the project, I noticed that the AWS Console has been updated. As a result, several steps like manually configuring Origin Access Control (OAC) or updating the S3 bucket policy were **no longer necessary** in my case.

Here's What I Actually Did:

- I created a CloudFront distribution and selected my **S3 bucket** as the **origin.**
- I named the distribution 'NextWork Three tier S3 Bucket.'
- I **disabled Web Application Firewall** (WAF) by choosing "Do not enable security protections."
- In the settings section, I scrolled to **Default root object** and entered index.html.
- That's it! I didn't manually set up OAC or update the S3 bucket policy. My website loaded successfully via CloudFront right after the distribution was created.

What Changed:

It seems that the latest AWS CloudFront setup **automatically applies** the necessary permissions (like creating OAC and updating the bucket policy in the background), streamlining the entire setup process. This made my **deployment much faster** and required fewer steps than documented in earlier tutorials.

Final Thought:

If you're using the updated AWS Console, you may **not need** to manually complete steps like **OAC creation** or **policy edits**. Just follow the streamlined UI during distribution creation, and your site may be accessible right away just like mine was!

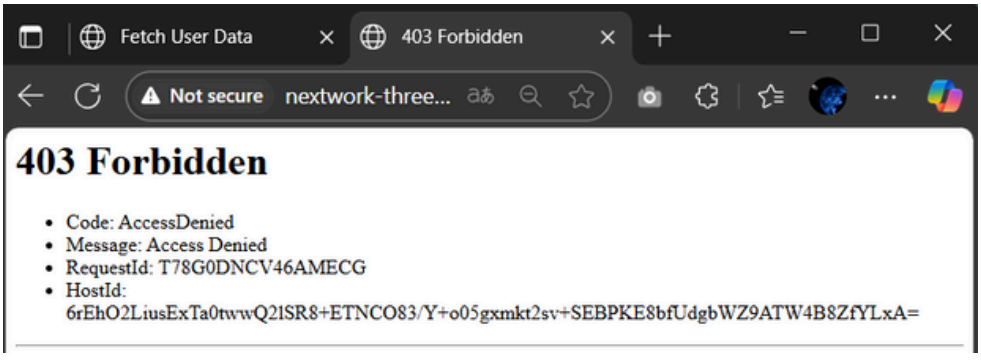

Kareshma
Rajaananthapadmanaban

S3 vs CloudFront for Hosting

In this secret mission, I will enable S3 static website hosting and compare it with CloudFront because it helps demonstrate the differences in performance, security, and access between **direct S3 hosting** and **CDN-based delivery**. This showcases deeper understanding of AWS services and reflects advanced skills in web hosting and optimization.

For my project extension, I'm comparing CloudFront and S3 static website hosting. I initially had an error with static website hosting because, even though hosting was enabled, the **objects** in my bucket were **still private.** S3 static websites **require public access** to serve files directly, so I need to update object permissions or the bucket policy.

I tried resolving this by **disabling** "**Block all public access**" in the S3 settings. I still ran into an error because I **hadn't added** a bucket policy to explicitly grant public read access to the objects. Without that policy, the files remain inaccessible, and S3 returns a **403 error** even though public access is no longer blocked globally.

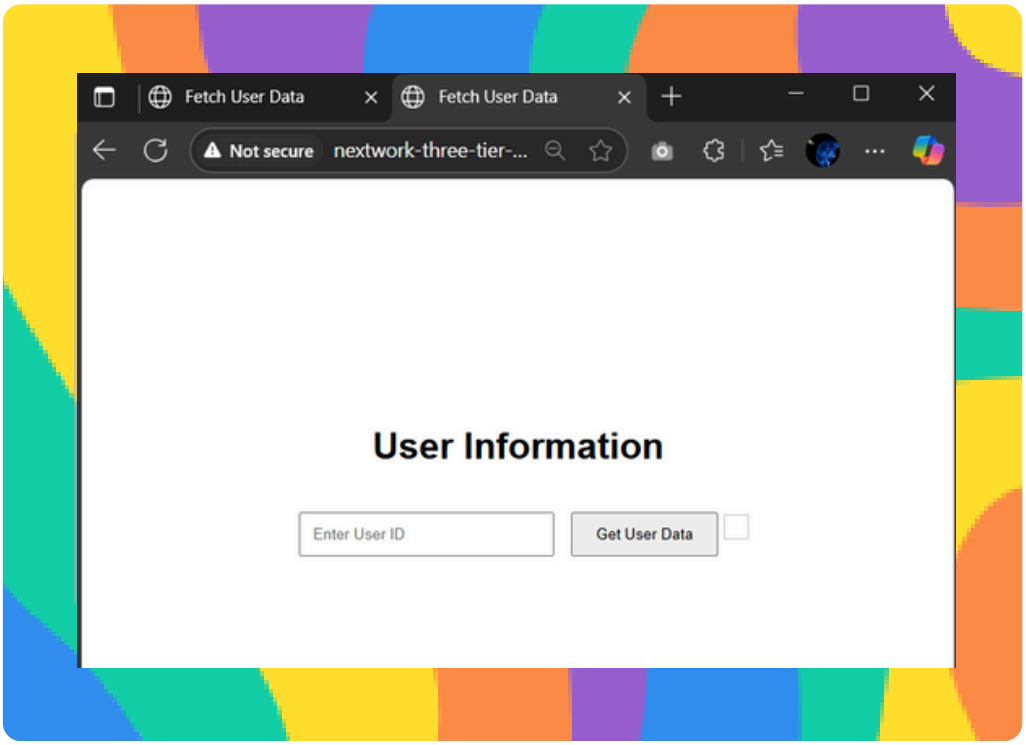


Bucket web endpoint error

Is website hosted with S3 now?

Yes! After adding the correct bucket policy with public read access for all objects, my static website is now successfully served directly from the S3 website endpoint. This confirms that **public access** and **permissions** were properly configured for S3 static website hosting.

I could finally see my S3 hosted website when I added a bucket policy that allowed **public read access** to all objects using the correct bucket ARN. This worked because it explicitly granted everyone permission to retrieve files from the bucket, enabling **S3 to serve the website content** through its **static hosting endpoint.**



S3 static website Hosting

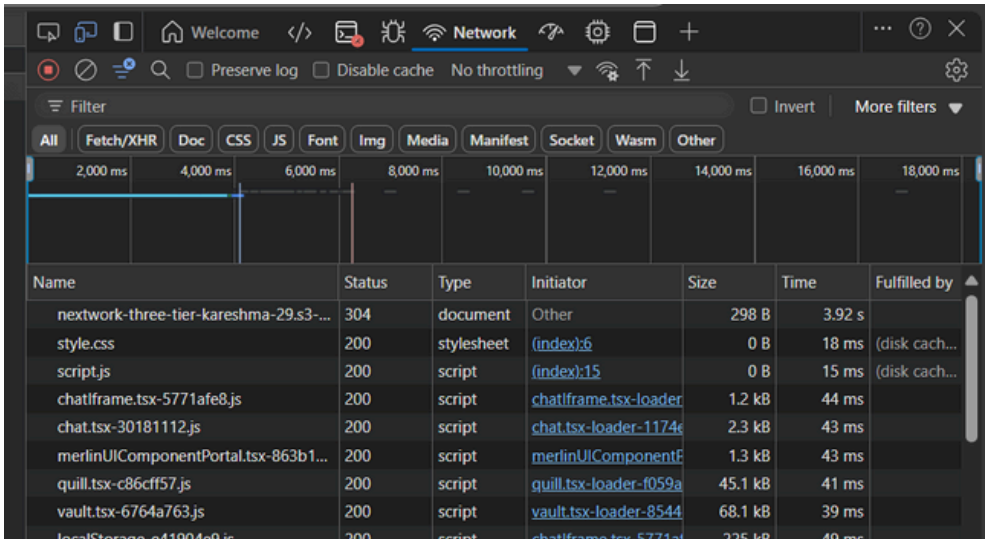
Updated Bucket policy

Compared to the permission settings for my CloudFront distribution, **using S3** meant making the bucket and its objects publicly accessible through a bucket policy, which poses a **higher security risk**. I preferred **CloudFronts** setup with Origin Access Control, as it allowed me to **keep** the **bucket private** while still **serving content securely**.

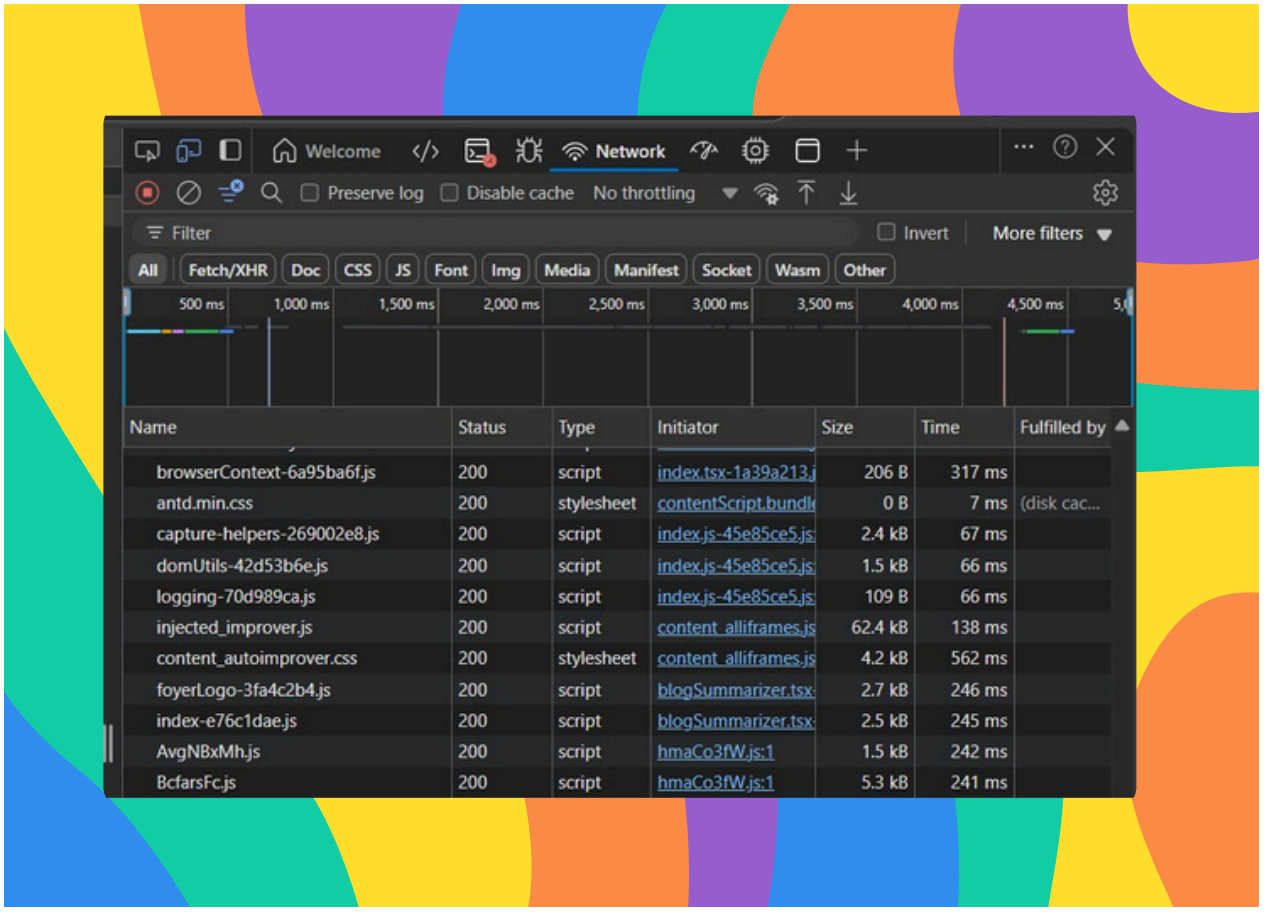
S3 vs CloudFront Load Times

Load time means how quickly a websites' content is delivered to and rendered by the users' browser. The **load times** for the **CloudFront site** were **faster than** the **S3 site** because CloudFront caches content at edge locations around the world, reducing latency. In contrast, S3 serves files from a single region, leading to longer data travel times.

A business would prefer CloudFront when performance, global reach, and security are priorities such as for production websites, e-commerce platforms, or apps with international users. **S3 static** website hosting might be sufficient when serving **simple sites** with low traffic or internal tools where speed and security are less critical.



S3 load time network



CloudFront load time network

Delete the Resources

🔻 Step 1: Delete CloudFront Distribution

1. Go to the CloudFront Console.
2. Select your distribution.
3. Click Disable and wait for the status to change to Disabled.
4. Once disabled, select Delete.

💡 If the Delete button is inactive, wait for propagation to complete (check “Last modified” timestamp), then try again.

🔻 Step 2: Delete S3 Bucket

1. Go to the S3 Console.
2. Select your bucket.
3. Click Empty and confirm.
4. Click Delete and confirm again.

⚠️ Important Cleanup Notes

- Always delete the CloudFront distribution before the S3 bucket.

Deleting the bucket first may cause the distribution to reference a missing origin, making it harder to delete.

- Accidentally deleted the S3 bucket first?

No problem! Change the origin access setting in CloudFront back to Public, then delete the distribution.

Kareshma
Rajaananthapadmanaban

Follow up for more interesting projects !!!

Check out nextwork.org / my profile for more projects