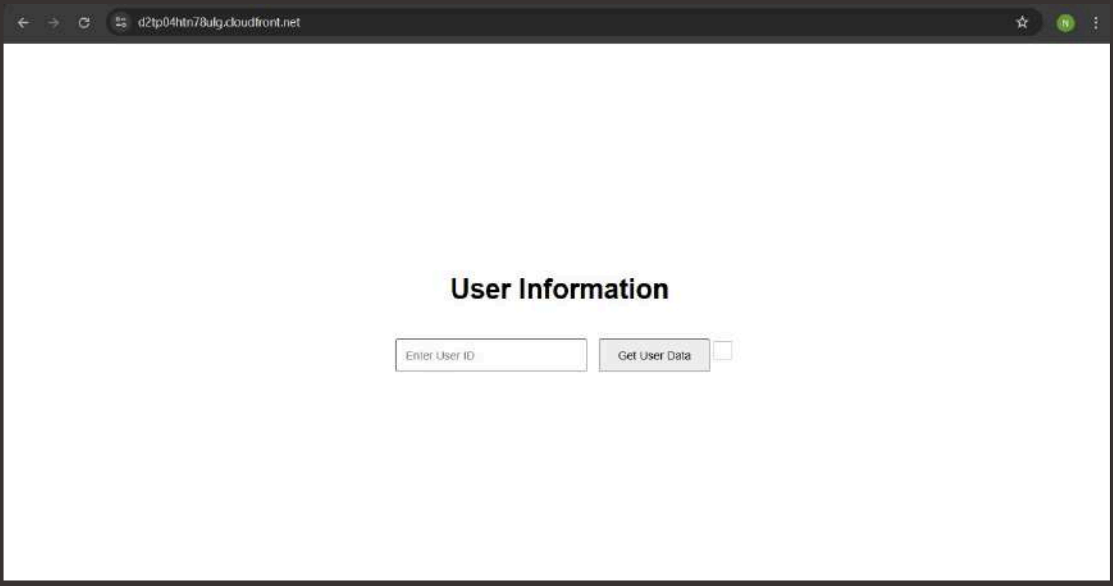
nextwork.org

Website Delivery with CloudFront

Nicolás Aversa



Introducing Today's Project!

In this project, I will demonstrate how to set up my application to interact with the user. I'm doing this project to learn about CDN's, a business' secret weapon for delivering content quickly and efficiently around the world.

Tools and concepts

Services I used were S3 and CloudFront. Key concepts I learnt include content delivery network (CDN), requests signing, and origin access control OAC.

Project reflection

This project took me approximately one hour. The most challenging part was making the S3 bucket publicly accessible. It was most rewarding to see how fast the CloudFront hosted website loaded its content.

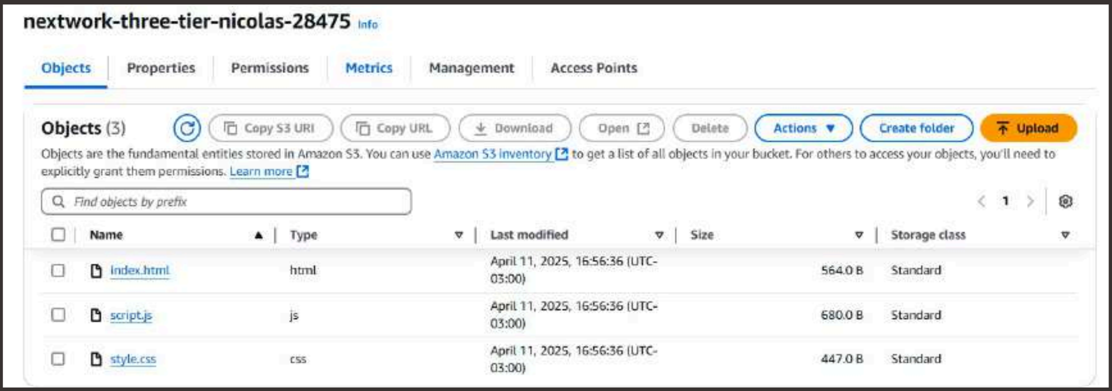
I did this project to see firsthand why companies choose CDNs like CloudFront over basic S3 hosting. The key takeaway? CDNs solve the "distance problem" by caching content at edge locations worldwide, making websites load faster for global users while reducing server costs and improving security.

Set Up S3 and Website Files

I started the project by creating an S3 bucket to store my website's files. I can't use CloudFront for this task because CloudFront is a content delivery network (CDN) designed to cache and distribute content globally, not to store data.

The three files that make up my website are index.html, which is where you organise the text, pictures, and everything that makes up your webpage. style.css, which is where you write down the visual appearance of your website's HTML elements. And script.js, which refers to a JavaScript file that adds interaction to your website.

I validated that my website files work by opening index.html in my browser.



nextwork-three-tier-nicolas-28475

Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

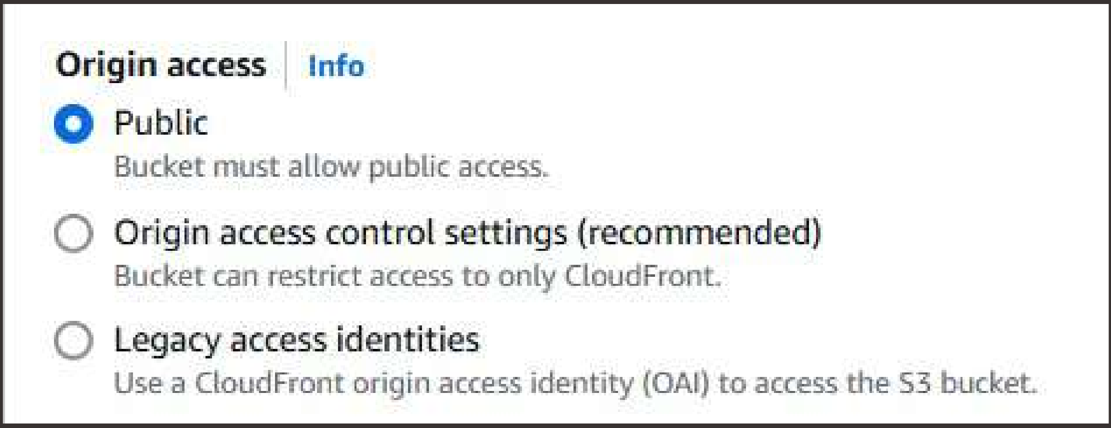
Name	Type	Last modified	Size	Storage class
index.html	html	April 11, 2025, 16:56:36 (UTC-03:00)	564.0 B	Standard
script.js	js	April 11, 2025, 16:56:36 (UTC-03:00)	680.0 B	Standard
style.css	css	April 11, 2025, 16:56:36 (UTC-03:00)	447.0 B	Standard

Exploring Amazon CloudFront

Amazon CloudFront is a content delivery network, which means it speeds up the distribution of static and dynamic web content, such as .html, .css, .js, and image files. Businesses and developers use CloudFront because it uses caching; the process of storing copies of files in a cache, i.e. a temporary storage location, so that they can be accessed more quickly. CloudFront caches your website content in multiple servers around the world. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency, so content is delivered with the best possible performance.

To use Amazon CloudFront, you set up distributions, which are a set of instructions that tells CloudFront how to deliver your content. I set up a distribution for the delivery of my website files. The origin is where your website's files are stored.

My CloudFront distribution's default root object is 'index.html'. This means this will be the file that CloudFront should serve when someone visits the root URL of my website.



Origin access | Info

[✓ Public]
Bucket must allow public access.

Origin access control settings (recommended)
Bucket can restrict access to only CloudFront.

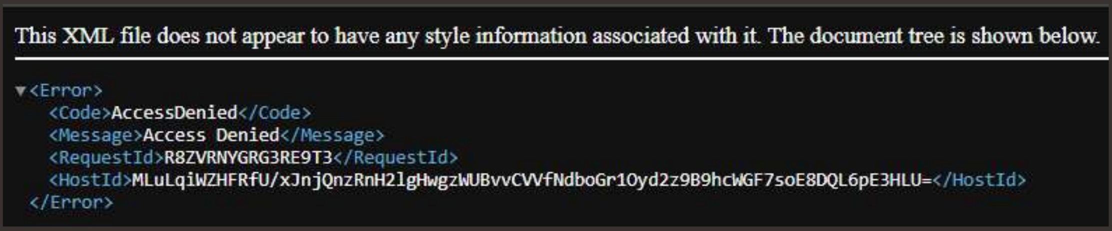
Legacy access identities
Use a CloudFront origin access identity (OAI) to access the S3 bucket.

Handling Access Issues

When I tried visiting my distributed website, I ran into an access denied error because I haven't given CloudFront permission to access my S3 bucket. By default, S3 buckets are private. CloudFront needs explicit permission to access the files in my bucket.

My distribution's origin access settings were set to 'Public'. This caused the access denied error because it means anyone can access the content directly from the origin i.e. my S3 bucket, not just via CloudFront. This caused the access denied error because public origin access doesn't automatically update the permissions of the objects in your S3 bucket - for security, the objects are private by default.

To resolve the error, I set up origin access control (OAC). OAC is a special user for CloudFront that lets you keep your S3 bucket and objects not publicly accessible, while still making sure they can be accessed through CloudFront.



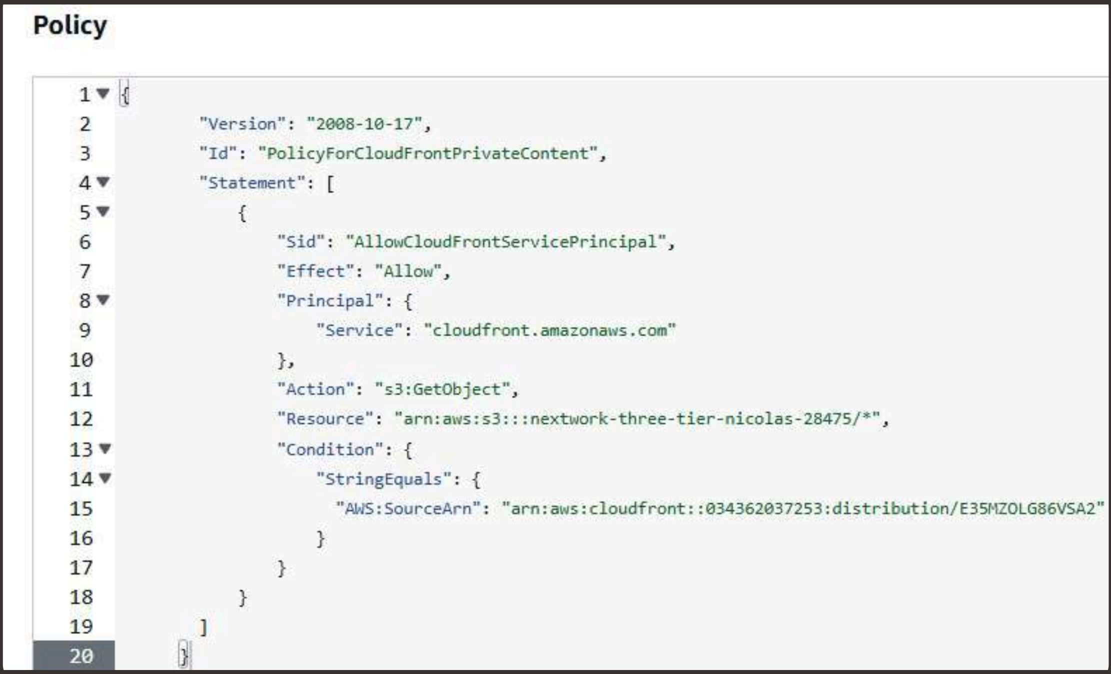
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<Error>
   <Code>AccessDenied</Code>
   <Message>Access Denied</Message>
   <RequestId>R8ZVRNYGRG3RE9T3</RequestId>
   <HostId>MLuLqiWZHFRfU/xJnjQnzRnH2lgHwgzWUBvvCVVfNdboGr1Oyd2z9B9hcWGF7soE8DQL6pE3HLU=</HostId>
 </Error>
```

Updating S3 Permissions

Once I set up my OAC, I still needed to update my bucket policy because I need to make files accessible for CloudFront. This new OAC has sign requests feature activated, which means before forwarding a request to the origin i.e. the S3 bucket storing the website files, CloudFront signs the request. This adds a digital signature that the S3 bucket will check for on every incoming request. If the signature is valid, it grants access to the requested object.

Creating an OAC automatically gives me a policy I could copy, which grants permissions to CloudFront so it can access the S3 bucket content.


Policy

```
{
        "Version": "2008-10-17",
        "Id": "PolicyForCloudFrontPrivateContent",
        "Statement": [
            {
                "Sid": "AllowCloudFrontServicePrincipal",
                "Effect": "Allow",
                "Principal": {
                    "Service": "cloudfront.amazonaws.com"
                },
                "Action": "s3:GetObject",
                "Resource": "arn:aws:s3:::nextwork-three-tier-nicolas-28475/*",
                "Condition": {
                    "StringEquals": {
                      "AWS:SourceArn": "arn:aws:cloudfront::034362037253:distribution/E35MZOLG86VSA2"
                    }
                }
            }
        ]
}
```

S3 vs CloudFront for Hosting

For my project extension, I'm comparing CloudFront vs S3 for website hosting. I initially had an error with static website hosting because I had 'Block all public access' ON for my S3 bucket.

I tried resolving this by unchecking the 'Block all public access' setting. I still ran into an error because this doesn't actively grant permission to access the objects; it simply stops blocking all public access attempts.

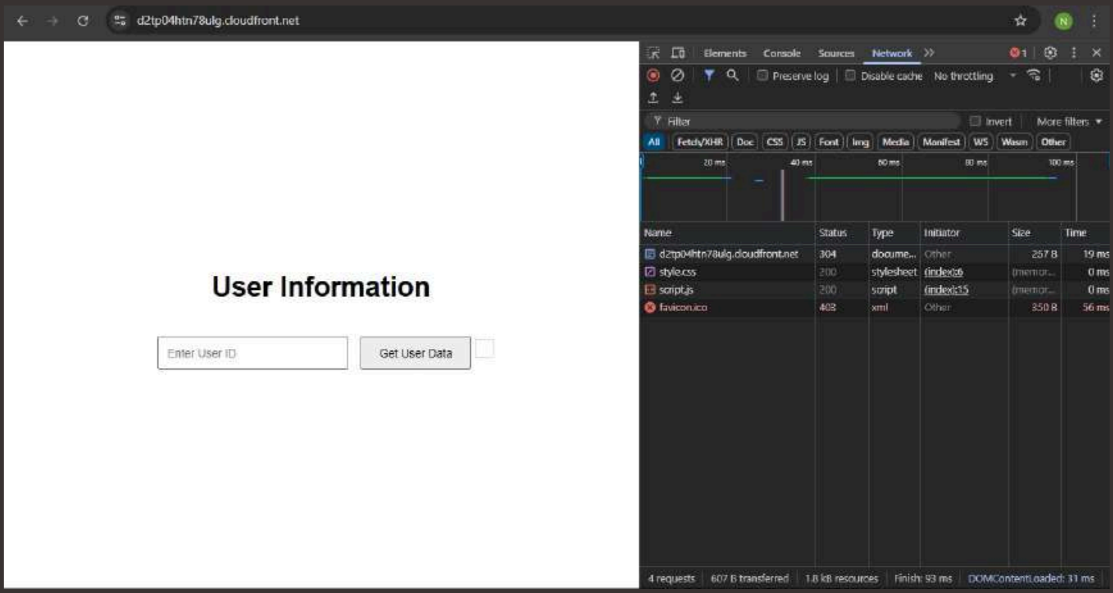
I could finally see my S3 hosted website when I updated my bucket policy. This worked because the bucket policy, on the other hand, is needed to explicitly grant permissions.

Compared to the permission settings for my CloudFront distribution, using S3 meant more manual settings. I preferred using CloudFront, because it does its job faster, and in a better way, hence why companies use it.

S3 vs CloudFront Load Times

Load time means the duration it takes for a user's browser to fully retrieve and display a webpage, including all its assets (HTML, CSS, JavaScript, images, etc.). The load times for the CloudFront site were faster than the S3 site because CloudFront's CDN caches content closer to users globally, while S3 static website hosting serves files directly from a single region.

A business would prefer CloudFront when they need fast, reliable global content delivery with features like edge caching, DDoS protection, and HTTPS support – essential for production websites, high-traffic applications, or any service where performance and security are critical. S3 static website hosting might be sufficient when dealing with simple, low-traffic sites in a single region, internal tools, or temporary staging environments where advanced features aren't necessary and cost efficiency is a priority. Essentially, CloudFront is for scaling performance while S3 alone works for basic, budget-friendly hosting.



Name	Status	Type	Initiator	Size	Time
d2tp04htn78ulg.cloudfront.net	304	docume...	Other	257 B	19 ms
style.css	200	stylesheet	(index):6	(memor...	0 ms
script.js	200	script	(index):15	(memor...	0 ms
favicon.ico	403	xml	Other	350 B	56 ms

4 requests | 607 B transferred | 1.8 kB resources | Finish: 93 ms | DOMContentLoaded: 31 ms

Everyone should be in a job they love.

Check out nextwork.org for more projects