

Project Name: Static Website Hosting on AWS S3 with CloudFront CDN

Date Completed: May 24, 2025

Name: Loren Deklerk

1. Project Overview

This project involved the end-to-end deployment of a static website using Amazon Web Services (AWS) infrastructure. The primary objectives were to host the website content securely, ensure high availability, and deliver content globally with low latency using a Content Delivery Network (CDN). This setup provides a scalable, cost-effective, and performant solution for serving static web assets.

Key Technologies Used:

- AWS S3 (Simple Storage Service): Used for highly durable and scalable storage of static website assets (HTML, CSS, JavaScript, images, etc.).
- AWS CloudFront: Utilized as a global CDN to cache content at edge locations, reducing latency and offloading requests from the origin S3 bucket.
- AWS Identity and Access Management (IAM): Configured for secure access control and permissions for S3 bucket policies.
- Amazon Route 53: Used for DNS management to map the custom domain to the CloudFront distribution.

2. Project Goals

- * Host a static website on a reliable and scalable cloud platform.
- * Ensure high availability and data durability for website content.
- * Accelerate content delivery globally through a CDN, reducing latency for end-users.
- * Optimize website performance through caching mechanisms.

- * Implement secure access control for the S3 bucket.
- * [Optional: Implement HTTPS encryption for secure communication.]
- * [Optional: Configure a custom domain name for easy access.]

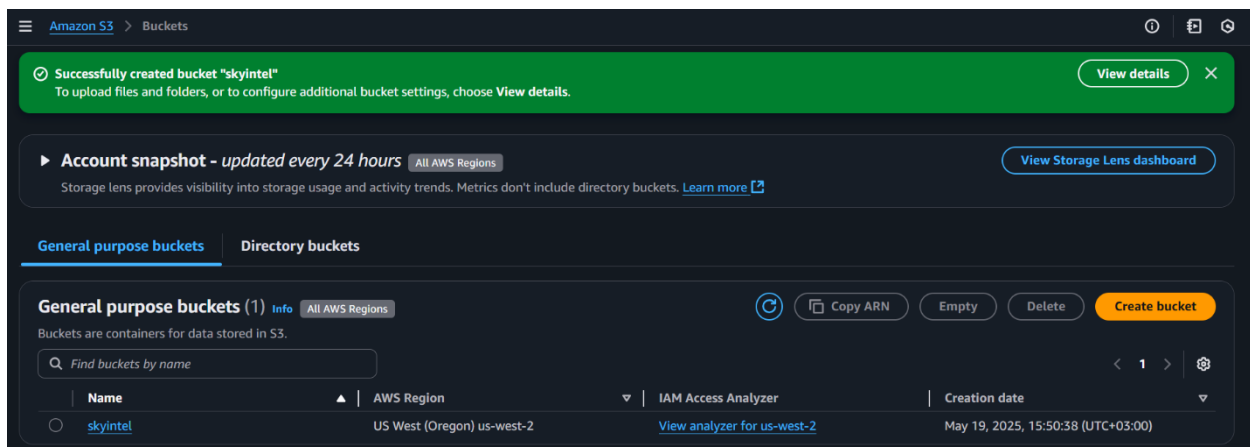
3. Implementation Steps & Detailed Configuration

3.1. AWS S3 Bucket Creation and Configuration

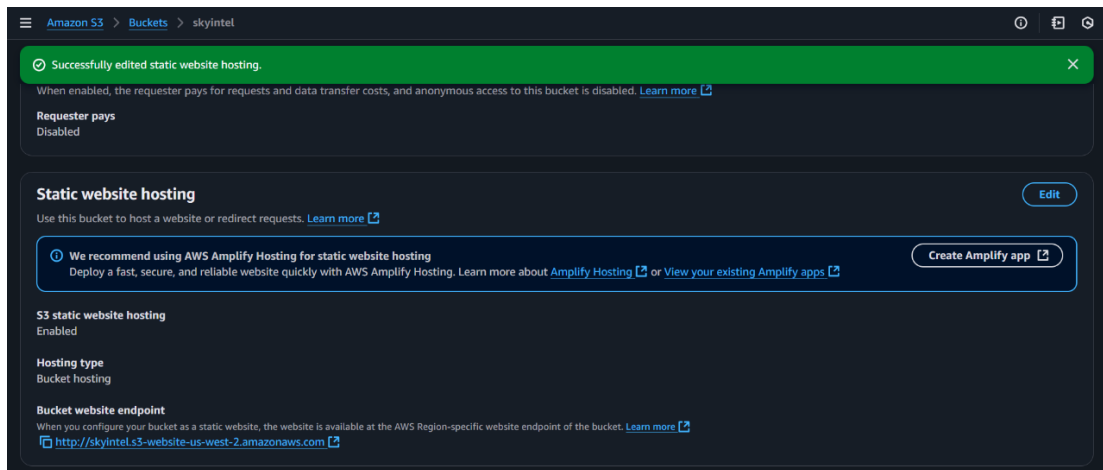
The foundation of the static website is an S3 bucket configured for web hosting.

Steps Performed:

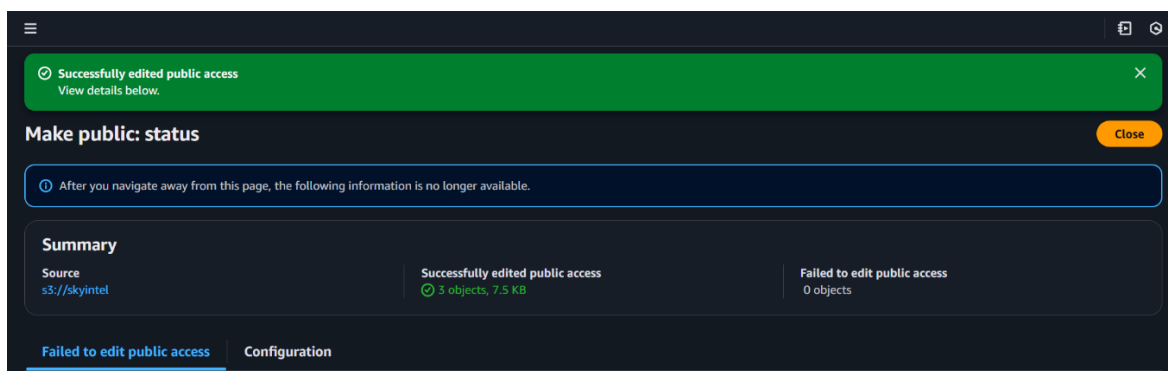
1. **Bucket Creation:** An S3 bucket named skyintel was created in the us-west-2 (Oregon) region. The bucket name was chosen to match the intended domain name for potential future custom domain integration.



2. **Static Website Hosting Enablement:** The bucket was configured to serve static website content. This involved specifying the `index.html` as the index document.



3. Public Access Block Configuration: Initial public access blocks were reviewed and, if necessary, modified to allow public access to the website content through CloudFront.

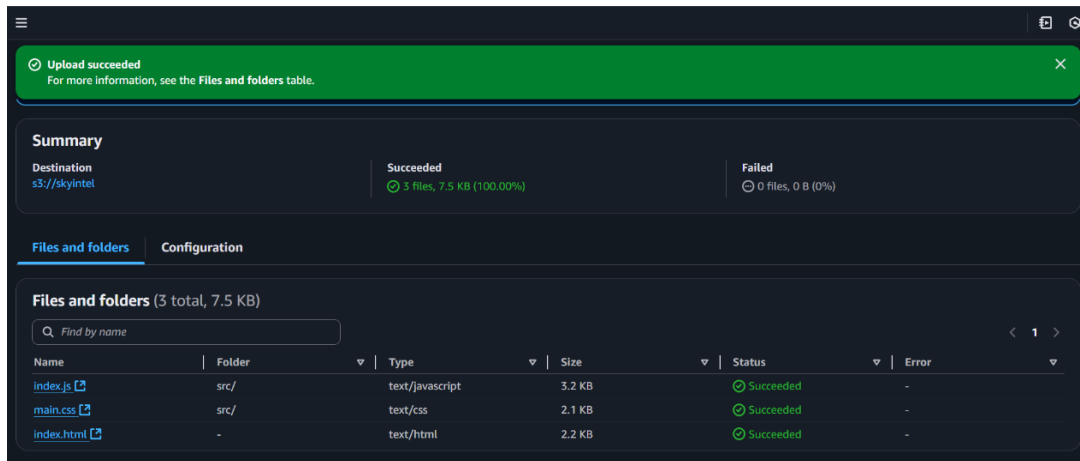


4. Bucket Policy Configuration: A bucket policy was applied to grant `s3:GetObject` permissions to the public, allowing CloudFront (and eventually the public) to retrieve objects from the bucket. If you don't configure correctly, you'll get this error, it is usually the index.html not granted the public access.

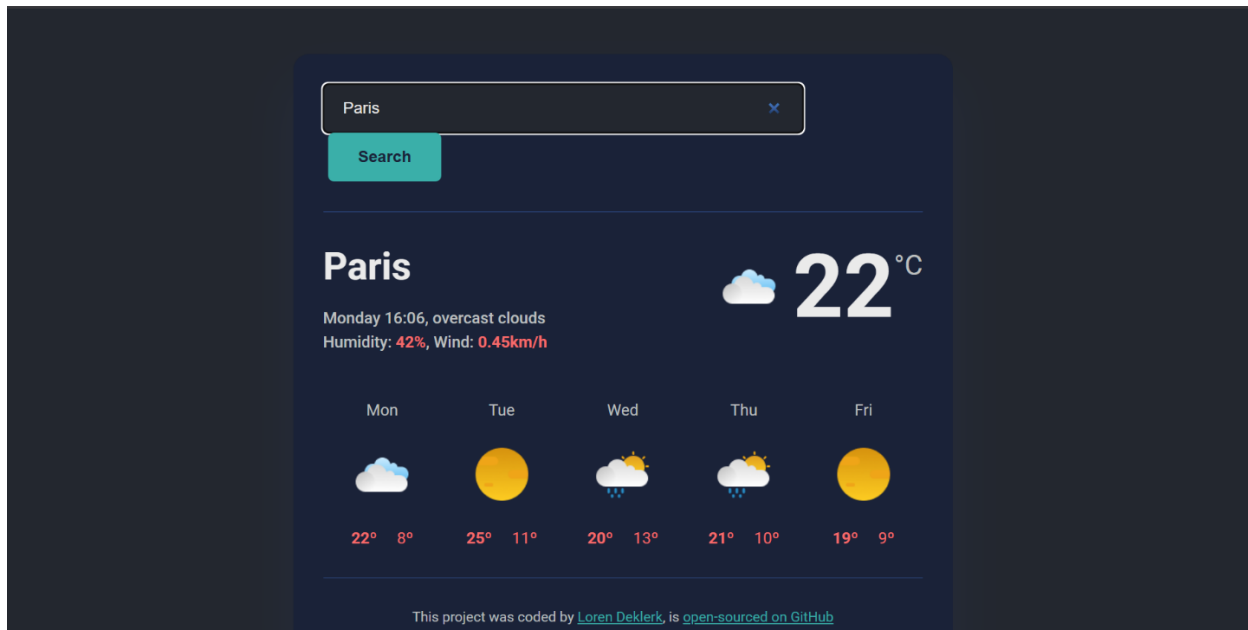


Note: While the bucket was made public for static website hosting, best practice with CloudFront is to restrict direct public access to the S3 bucket and use an Origin Access Control (OAC) or Origin Access Identity (OAI) for CloudFront to access the S3 bucket.

5. **Content Upload:** All static website files (HTML, CSS, JavaScript, images, etc.) were uploaded to the root of the S3 bucket.



6. Successful static website hosting!!

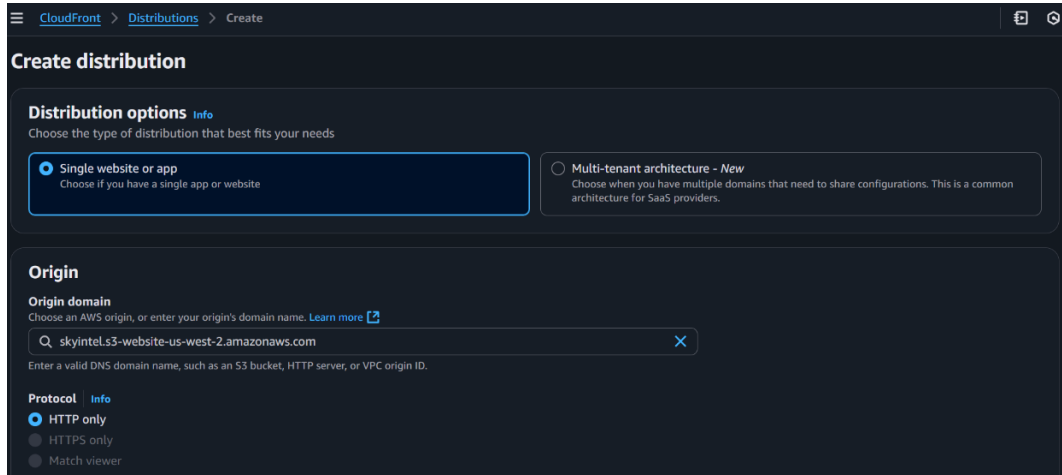


3.2. AWS CloudFront Distribution Setup

CloudFront was utilized to distribute the website content globally, improving loading times and reducing the load on the S3 origin.

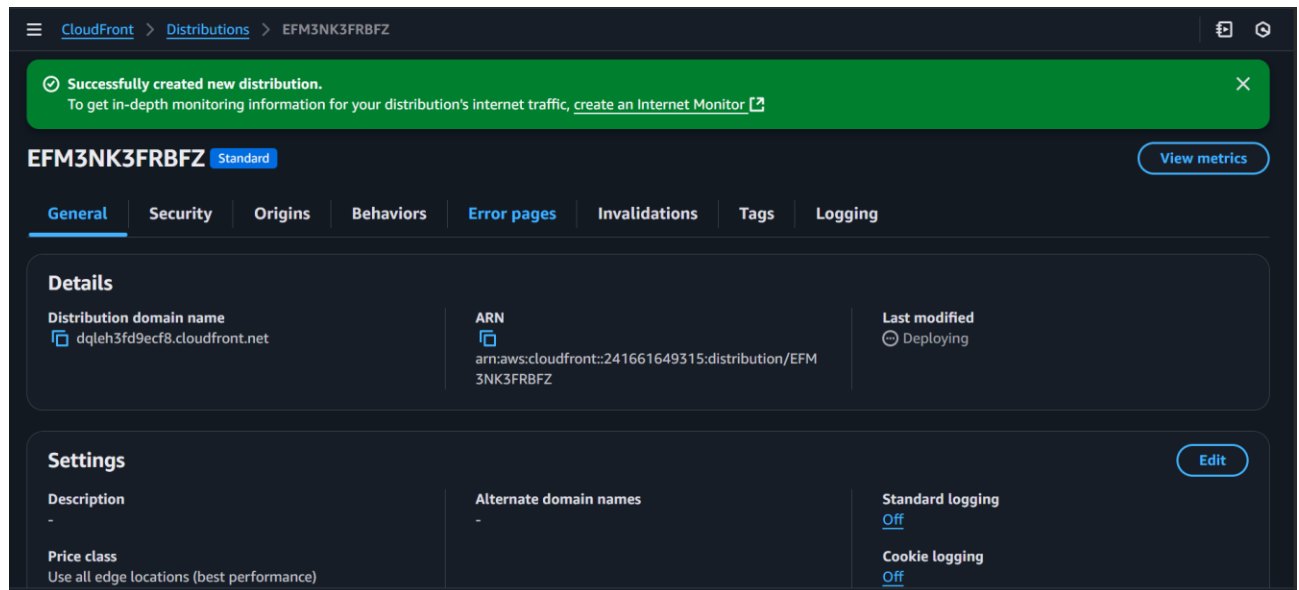
Steps Performed:

1. Distribution Creation: A new CloudFront web distribution was created.



The screenshot shows the AWS CloudFront 'Create distribution' console page. The breadcrumb navigation at the top reads 'CloudFront > Distributions > Create'. The main heading is 'Create distribution'. Below this, there's a section titled 'Distribution options' with an 'Info' link. It instructs the user to 'Choose the type of distribution that best fits your needs'. There are two radio button options: 'Single website or app' (selected) and 'Multi-tenant architecture - New'. The 'Single website or app' option has a subtext: 'Choose if you have a single app or website'. The 'Multi-tenant architecture - New' option has a subtext: 'Choose when you have multiple domains that need to share configurations. This is a common architecture for SaaS providers.' Below the distribution options is the 'Origin' section. It has a subheading 'Origin domain' and a note: 'Choose an AWS origin, or enter your origin's domain name. Learn more'. There is a search input field containing 'skyintel.s3-website-us-west-2.amazonaws.com' with a clear button (X). Below the input field is a note: 'Enter a valid DNS domain name, such as an S3 bucket, HTTP server, or VPC origin ID.' At the bottom of the origin section is the 'Protocol' section with an 'Info' link. It has three radio button options: 'HTTP only' (selected), 'HTTPS only', and 'Match viewer'.

2. Origin Configuration: The S3 bucket's static website hosting endpoint (<http://skyintel.s3-website-us-west-2.amazonaws.com>) was specified as the origin.
3. Cache Behavior: Default cache behavior was configured to optimize caching of static assets. This included setting TTL (Time To Live) for various content types.
4. Viewer Protocol Policy: Configured to 'Redirect HTTP to HTTPS' or 'HTTPS Only' for secure connections.
5. SSL/TLS Certificate:
A custom domain was used hence the default CloudFront SSL certificate was used.
6. Price Class: Selected 'Use All Edge Locations (Best Performance)' for global reach.
7. Successful creation of cloudfront distribution:



4. Security Considerations

Throughout the project, security was a key focus:

S3 Bucket Policies: Carefully crafted to allow only necessary access, primarily for content retrieval.

5. Challenges and Solutions

Challenge 1: Initially, direct public access to the S3 bucket was granted, which is less secure than using CloudFront's OAI/OAC.

Solution 1: I updated the S3 bucket policy to restrict direct public access and configured an Origin Access Control (OAC) for the CloudFront distribution, allowing CloudFront to securely access the S3 bucket.

Challenge 2: Debugging cache issues in CloudFront.

Solution 2: Used CloudFront invalidations for testing content updates and verified cache hit/miss using browser developer tools.

6. Project Outcome and Benefits

The project successfully delivered a robust and efficient static website hosting solution.

Global Accessibility & Performance: The website is now accessible globally with significantly reduced latency due to CloudFront's CDN, caching content at edge locations close to users.

Scalability: The architecture leverages the inherent scalability of AWS S3 and CloudFront, easily accommodating increasing traffic without manual intervention.

Cost-Effectiveness: S3 and CloudFront offer a highly cost-effective solution for static content delivery, with pay-as-you-go pricing.

High Availability & Durability: S3 provides 99.999999999% (11 nines) of data durability, ensuring the website content is always available and protected against loss.

Security: Enhanced security through controlled S3 access, CloudFront features.