Hands-on with AWS CloudFront – Content Delivery Optimization Using CDN

What is CloudFront?

Amazon CloudFront is a **Content Delivery Network (CDN)** provided by AWS that speeds up the distribution of static and dynamic content such as HTML, CSS, JavaScript, images, videos, and APIs.
With **310+ edge locations** and **13 regional edge caches** globally, CloudFront helps deliver content with **low latency and high transfer speed**, enhancing user experience.

Key Concepts

- **Edge Locations**
  Global data centers where content is cached and served to reduce latency.
- **Regional Edge Caches**
  Intermediate caches that reduce load on origin and improve content retrieval efficiency.
- **TTL (Time to Live)**
  Determines how long content is cached:
  - Default TTL for **edge location**: 24 hours
  - Default TTL for **regional edge cache**: 1 year
  - TTL values are customizable
- **Invalidation Requests**
  Used to manually remove cached content before TTL expires—essential for reflecting updated content immediately.

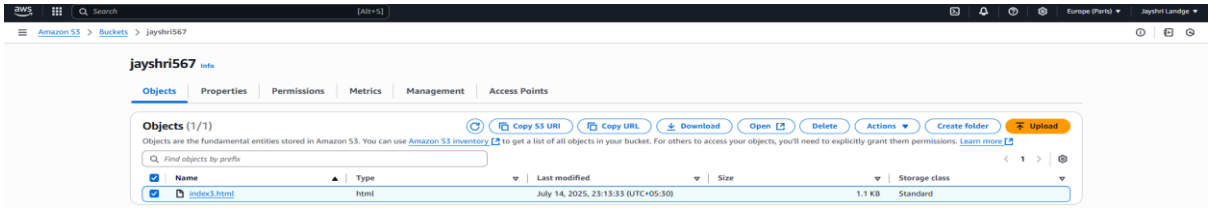
Step-by-Step: Creating a CloudFront Distribution (Using AWS Console)

🔸 **Option A: S3 Bucket (for static content)**

Goal: Deliver content from an S3 bucket using CloudFront with caching benefits

Step 1: Create or Use an Existing S3 Bucket

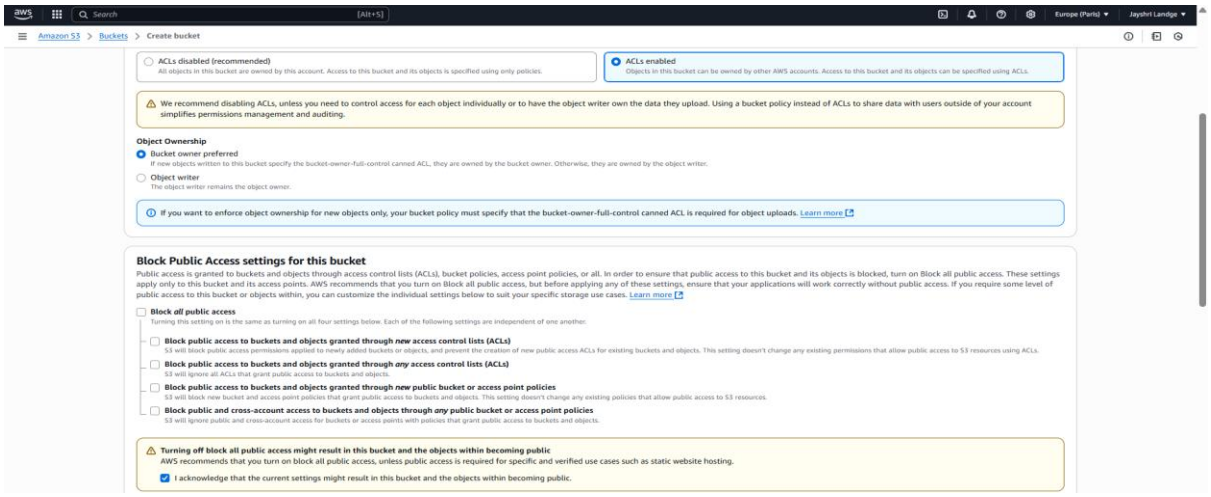
- Go to **S3 Service** → Create an S3 bucket (or use an existing one)
- Upload static content (e.g., index.html, images, etc.)



Screenshot 1: S3 Bucket with uploaded content

Step 2: Make Content Public (Optional for Testing)

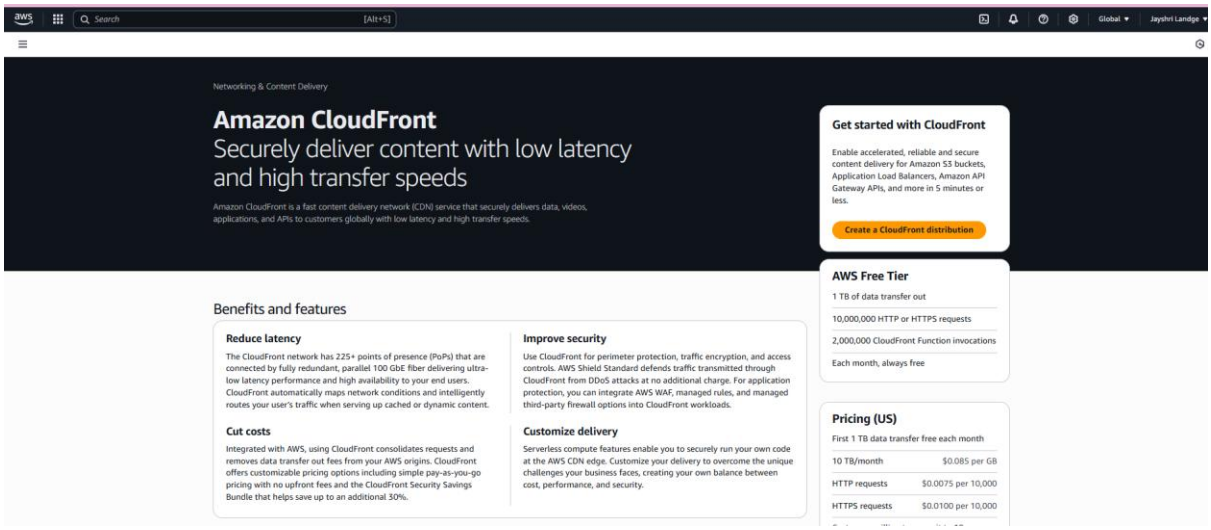
- Select the object in S3 → Choose **Permissions**
- Enable **public access** for the object (or configure bucket policy)



Screenshot 2: Public access settings on S3 object

Step 3: Navigate to CloudFront in AWS Console

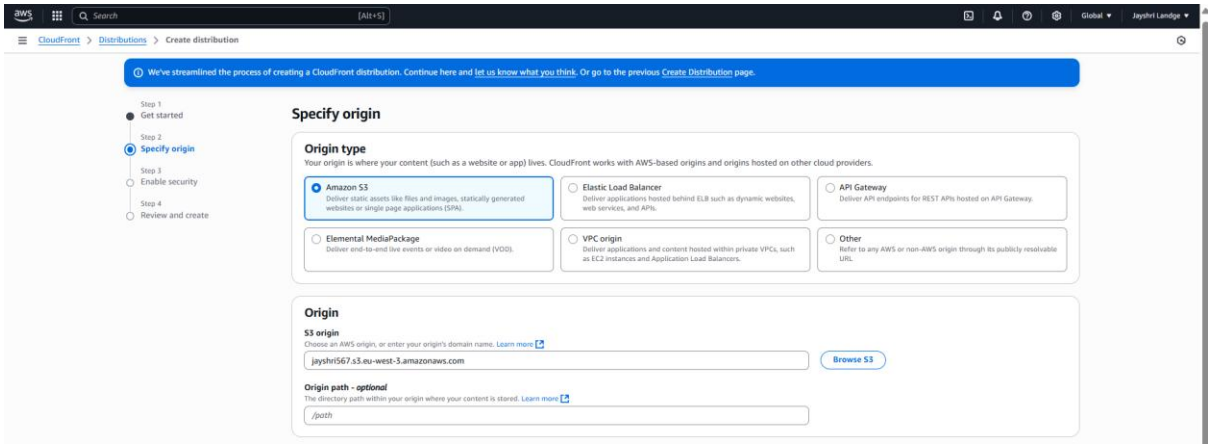
- Go to **CloudFront Service** from the AWS Console



Screenshot 3: AWS CloudFront Dashboard

Step 4: Create a New Distribution

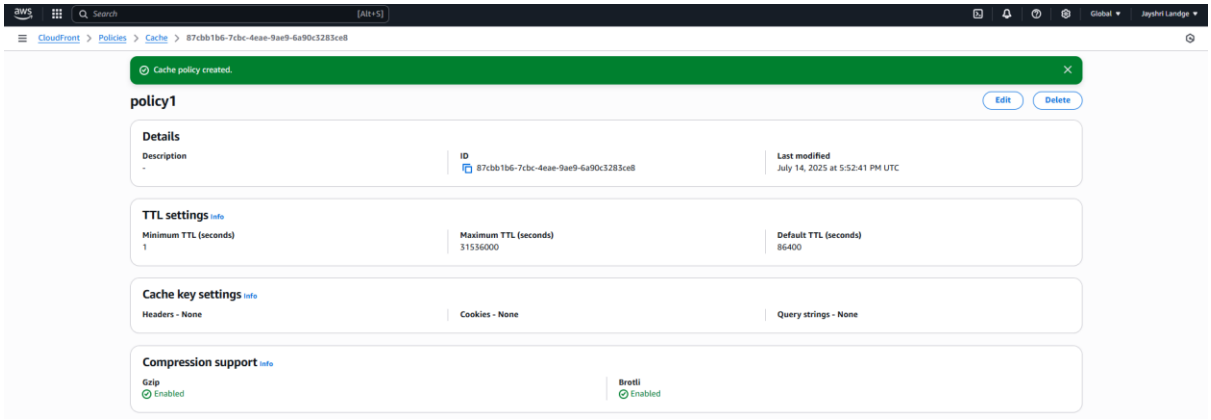
- Click **Create Distribution**
- Under **Origin Settings**:
  - Origin domain: Choose your S3 bucket (auto-suggested)
  - Origin access: Set as **Public** or enable OAC/Origin Access Control for security



Screenshot 4: Origin settings Of distribution

Step 5: Configure Default Settings

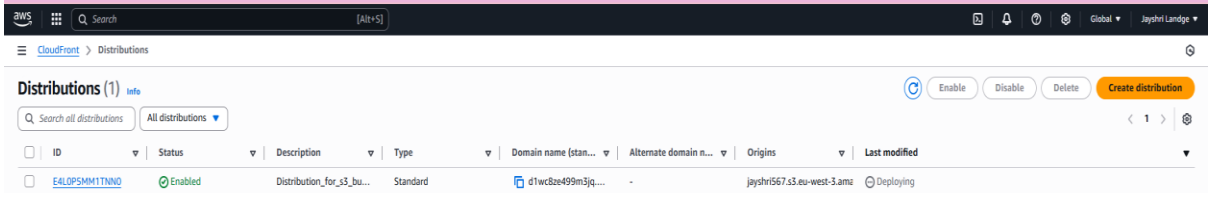
- Viewer protocol policy: **Redirect HTTP to HTTPS**
- Cache policy: Use **CachingOptimized** or create custom TTL settings
- Object caching: Choose **Use origin cache headers** or **Customize**



Screenshot 5: Cache and TTL configuration

Step 6: Review and Create

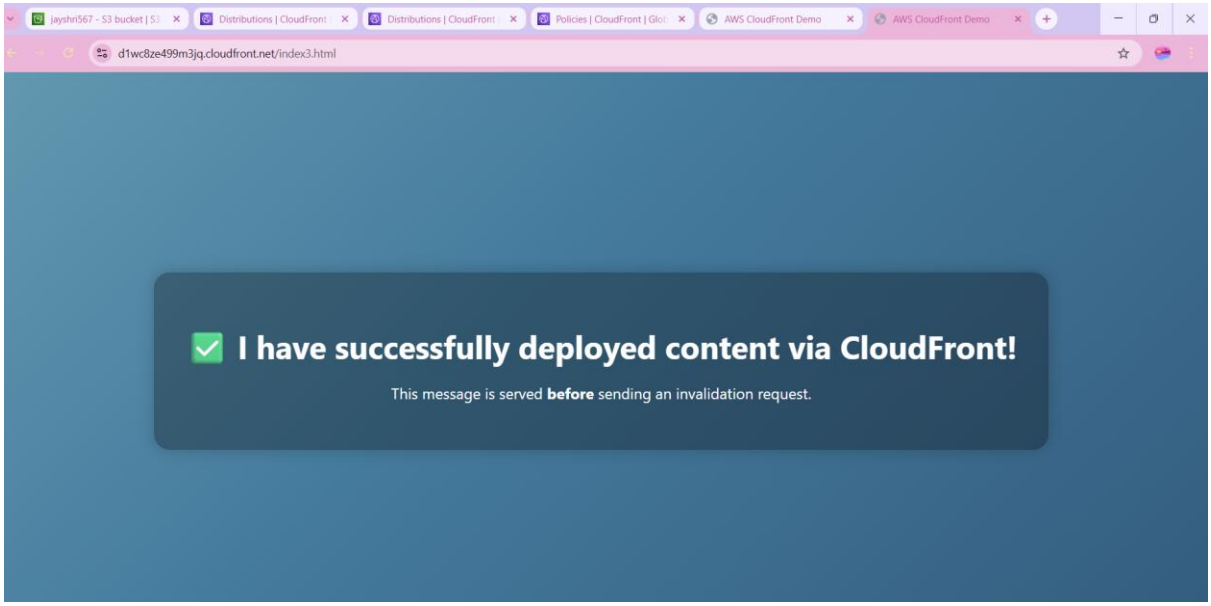
- Review configuration
- Click **Create Distribution**



Screenshot 6: Distribution status

Step 7: Access Content via CloudFront

- Once the status is **“Deployed”**, copy the **Distribution Domain Name**
- Paste it in the browser followed by /index.html or the path to your file



Screenshot 7: CloudFront domain serving content successfully

🔸 **Option B: EC2 Instance (for dynamic content)**

Goal: Optimize content delivery for dynamic applications hosted on EC2, such as websites and APIs, by leveraging CloudFront’s global edge network.

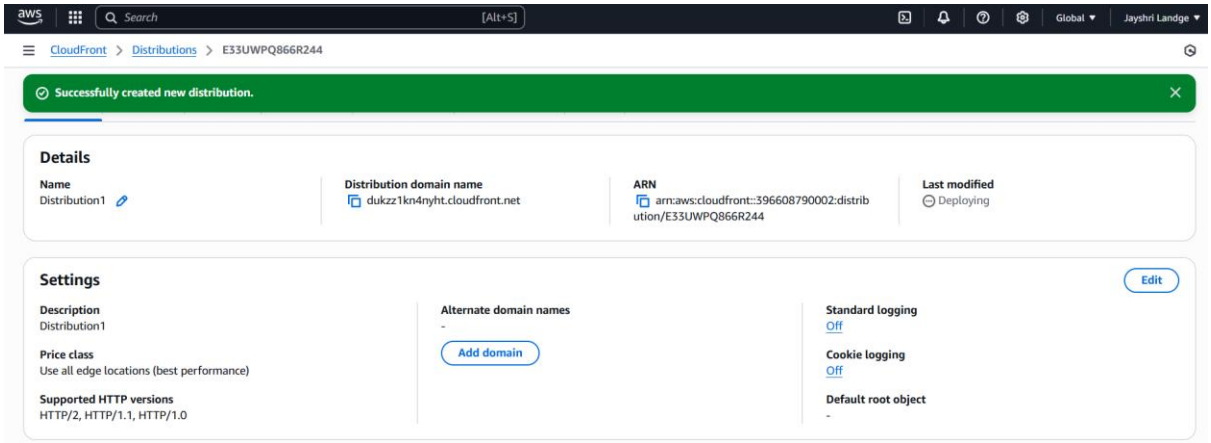
Step 1: Launch and Configure EC2 Instance

- Launch a **Linux EC2 instance**
- Install a web server (like Apache or Nginx or httpd)
- Deploy content (like index.html) inside /var/www/html/
- Ensure **port 80** (HTTP) is open in the security group
- Note down the **Public IPv4 DNS** or Elastic IP

Screenshot 8: EC2 instance running with a deployed web page

Step 2: Create CloudFront Distribution with EC2 as Origin

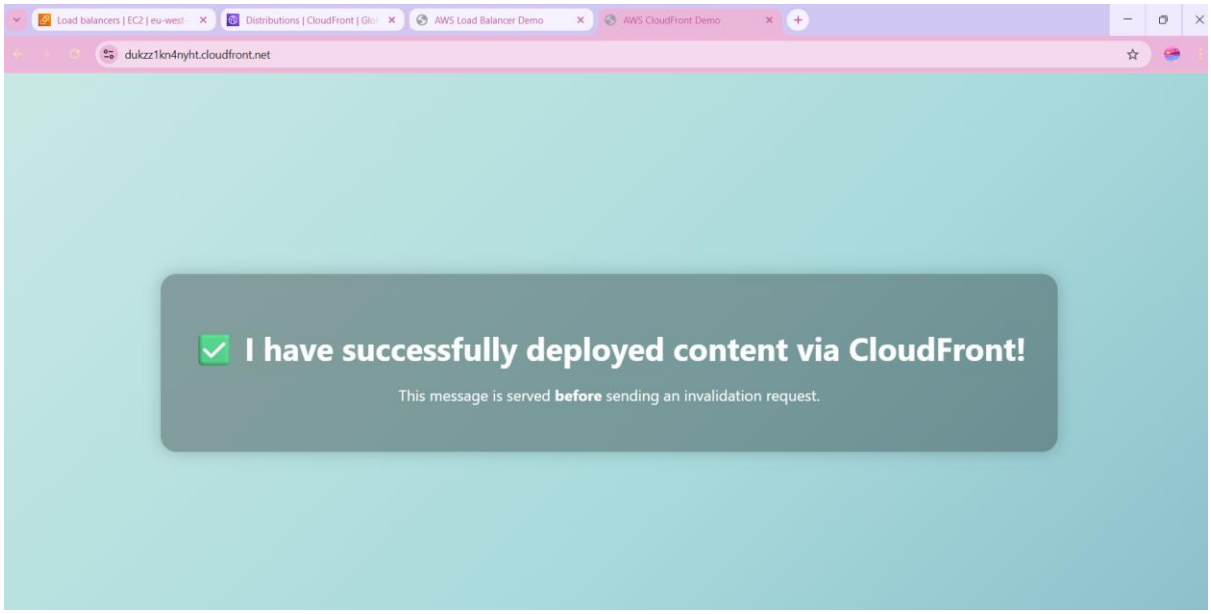
- Navigate to **CloudFront > Create Distribution**
- In **Origin Domain**, enter your **EC2 public DNS or Elastic IP**
- Set **Origin Protocol Policy**: HTTP or HTTPS (based on your EC2 config)
- Continue with default or custom cache settings



Screenshot 9: EC2-based origin configuration in CloudFront

Step 3: Access Content via CloudFront Distribution Domain

- After deployment, use CloudFront domain (e.g., d12345.cloudfront.net) to access content hosted on your EC2

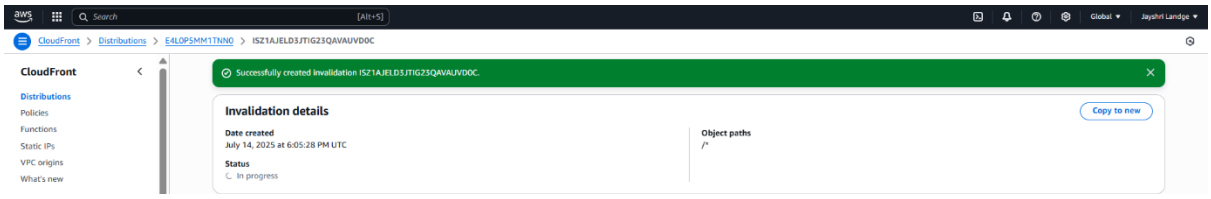


Screenshot 10: EC2 content served via CloudFront domain

Optional: Sending Invalidation Request

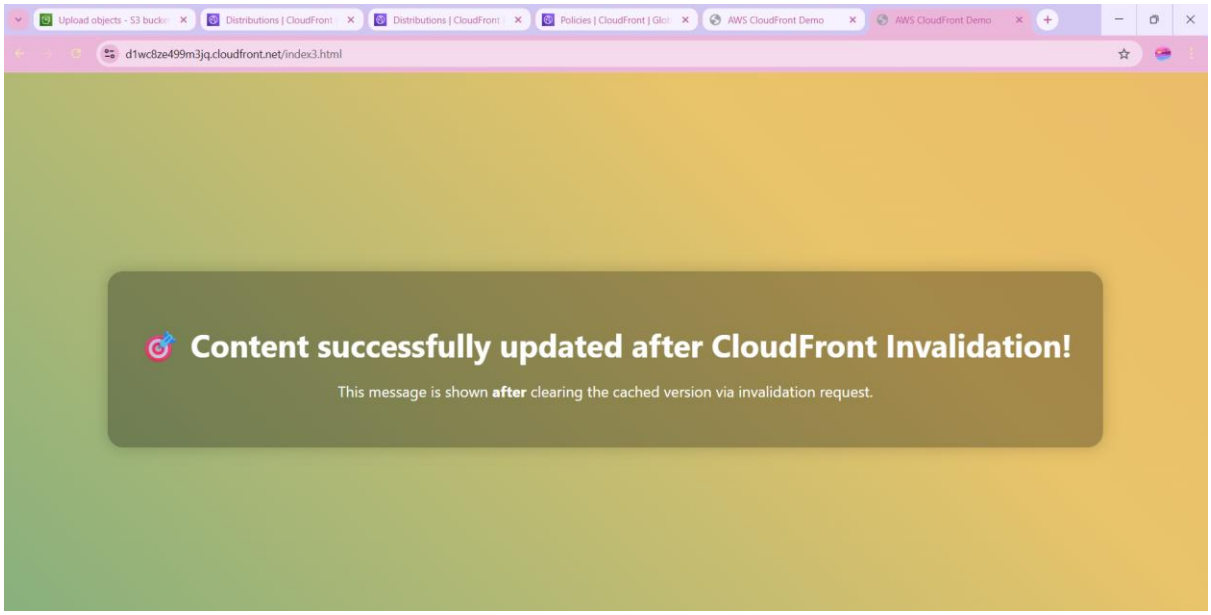
Useful when the origin content has been updated but CloudFront still serves old content.

- Go to your distribution → Choose **Invalidations**
- Click **Create Invalidation**
- Enter path (e.g., /* to clear all cached content)
- Confirm and submit



Screenshot 11: Invalidation request screen status In progress

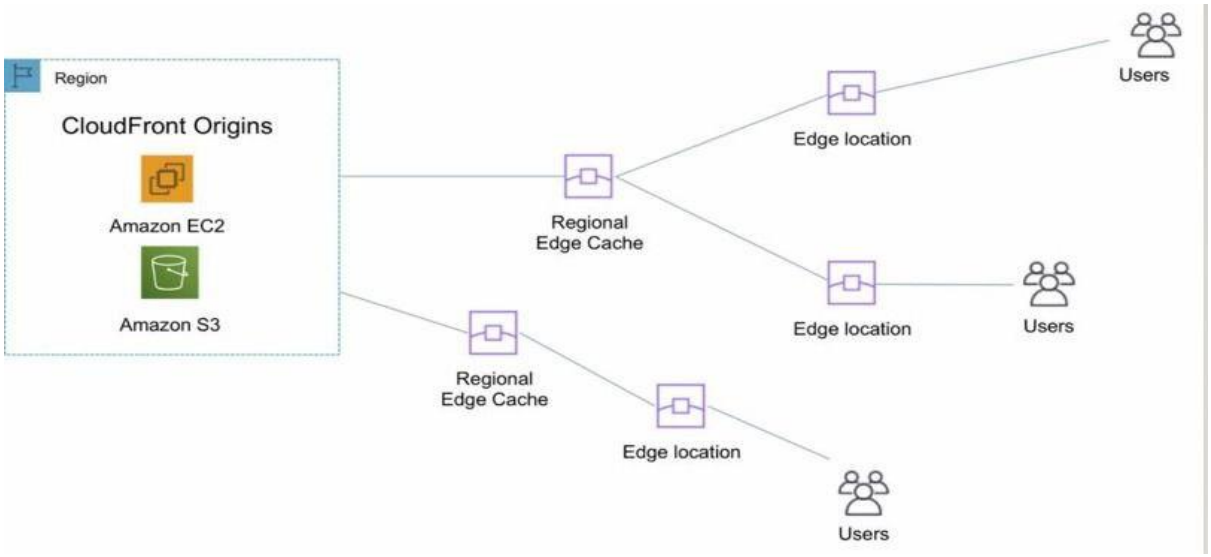
Screenshot 12: Invalidation request screen status completed



Screenshot 13: After Invalidation request Changes on Webserver

Architecture Reference

Here's a visual representation of how CloudFront works with **both EC2 and S3 origins**. Users are routed through edge locations and regional edge caches, reducing latency and enhancing speed.



Screenshot 14: CloudFront Architecture (EC2 + S3 as Origins)

Key Takeaways

- Practiced using CloudFront via AWS Console to accelerate content delivery from both S3 (static) and EC2 (dynamic) origins.
- Gained experience with GUI-based configuration, demonstrating hands-on capability without the use of CLI tools.
- Understood TTL behavior, differences between Edge Locations and Regional Edge Caches, and the purpose of invalidation requests.
- Observed how CloudFront caching improves performance, reduces origin load, and ensures efficient content distribution.
- Leveraged global CDN caching even for custom web apps hosted on EC2.
- Learned how CloudFront offloads repeated requests from EC2 servers, boosting scalability for dynamic content delivery.

Conclusion

This hands-on practice enhanced practical understanding of how **AWS CloudFront** works behind the scenes to deliver optimized, low-latency content across the globe. It helped in:

- Understanding caching mechanisms through **Edge and Regional locations**
- Effectively using the AWS Console to manage CDN behavior
- Gaining confidence in handling **real-world scenarios like content invalidation**