

@devopschallengehub



Your JavaScript changes aren't showing after deploying CloudFront with default caching. What's wrong ? How will you solve this?

Problem: Why changes aren't showing?

- CloudFront **caches your content** (like JavaScript, CSS, images) at its **edge locations** all over the world.
- By default, CloudFront may keep a cached file for **24 hours (TTL = 86400 seconds)** before checking your origin (S3 or EC2).
- That means if you update app.js, CloudFront **still serves the old cached version** to users until the cache expires.

So the issue is: **CloudFront is serving cached JS, not the new one.**

Solution: Ways to fix without disabling caching

You don't want to fully turn off caching (it speeds up your site!). Instead, use one of these strategies:

1. Object Versioning (Best Practice)

- Instead of always naming the file app.js, add a version number or hash in the filename.
 - Example:
 - Old → app-v1.js
 - New → app-v2.js
- Update your HTML to point to the new file.
- CloudFront sees it as a **new file**, so no caching conflict.

2. Cache Invalidation

- When you upload a new version of a file (like app.js), create an **invalidation** in CloudFront.
- This tells CloudFront: "Remove the old app.js from all caches."
- Example: Invalidate /* (all files) or just /app.js.
- After invalidation, users will get the latest version from your origin.
- ⚠️ Downside: Invalidations **cost money** if used too often (first 1000/month are free).

3. Lower Cache TTL for Static Assets (Optional)

- If you expect frequent changes, you can reduce the **cache duration** (e.g., set TTL = 300 seconds).
 - Example:
 - Set Cache-Control: max-age=300 in S3 metadata.
 - This way CloudFront re-checks more often.
 - ⚠️ Downside: Less caching efficiency = slightly higher costs/latency.
-

Summary

- **Why problem?** Old JavaScript is cached by CloudFront.
- **How to fix (best)?** Use **versioned filenames** for JS/CSS (e.g., app-v2.js).
- **Other options:** Invalidate cache or reduce TTL.

👉 Most real-world apps use **versioned filenames** because it's simple, cheap, and cache-friendly.

You updated **app.js** in S3, but CloudFront still shows the old version. Why?

- A. CloudFront caches old files at edge locations.
- B. CloudFront does not support JavaScript.
- C. CloudFront blocks updated files.
- D. S3 bucket is corrupted.

A. CloudFront caches old files at edge locations. ✅


To make sure new versions of JS/CSS are always served without disabling caching, what's the best practice?

- A. Use versioned filenames (e.g., app-v2.js).
- B. Delete the CloudFront distribution.
- C. Turn off all caching.
- D. Move files to EC2.

A. Use versioned filenames (e.g., app-v2.js). ✅

You just uploaded a new **app.js** and want users to get it immediately through CloudFront. What should you do?

- A. Create a cache invalidation for /app.js.
- B. Restart CloudFront service.
- C. Rename the S3 bucket.
- D. Disable Origin Access Control.

A. Create a cache invalidation for /app.js. 

Your web app changes files often, and you want CloudFront to refresh content faster. What's the right approach?

- A. Lower cache TTL (e.g., 300 seconds).
- B. Disable CloudFront completely.
- C. Make S3 objects public.
- D. Use Route53 to bypass caching.

A. Lower cache TTL (e.g., 300 seconds). 