Lab2_b_Be_A_Man_B

Lab 2B-Honors+: CloudFront Invalidation as a Controlled Operation
Objective
Students will:
1) Keep origin-driven caching for /api/public-feed (as in Honors)
2) Use versioned static assets for normal deployments (preferred)
3) Use CloudFront invalidation only for approved “break glass” events
4) Prove correctness with x-cache, Age, and invalidation status

AWS CLI provides create-invalidation for this workflow.

The Operational Rules (non-negotiable)

Rule 1 — Never invalidate /* for deployments
That’s the “Chewbacca Rage Invalidation™”. You only use it if:
- security incident
- corrupted content
- legal takedown
- catastrophic caching misconfig

(And you document why.)

Rule 2 — Prefer versioning for static
Example: /static/app.<hash>.js
No invalidation required; you deploy new file with a new name and update the HTML reference. AWS recommends versioning when you update frequently.
Documentation: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Invalidation.html?utm_source=chatgpt.com

Rule 3 — Invalidate only the smallest blast radius
Examples:
- /static/index.html
- /static/manifest.json
- /static/* (acceptable only if you can justify)

Rule 4 — Budget/limits awareness
First 1,000 invalidation paths/month free, then billed per path; wildcard counts as one path.

Part A — Add “break glass” invalidation procedure (CLI)
A1) Create an invalidation (single path

```
aws cloudfront create-invalidation \
 --distribution-id <DISTRIBUTION_ID> \
 --paths "/static/index.html"
```

AWS shows this exact CLI pattern.

A2) Create an invalidation (wildcard path)

```
  aws cloudfront create-invalidation \
 --distribution-id <DISTRIBUTION_ID> \
 --paths "/static/*"
```

Wildcards are allowed, but must be last character and paths must start with /
Documentation: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/invalidation-specifying-objects.html?utm_source=chatgpt.com

A3) Track invalidation completion

```
  aws cloudfront get-invalidation \
 --distribution-id <DISTRIBUTION_ID> \
 --id <INVALIDATION_ID>
```

Part B — "Correctness Proof" checklist (must submit)
B1) Before invalidation: prove object is cached

```
  curl -i https://chewbacca-growl.com/static/index.html | sed -n '1,30p'
  curl -i https://chewbacca-growl.com/static/index.html | sed -n '1,30p'
```

Expected:
  Age increases on second request (cached)
  x-cache shows Hit from cloudfront (or similar)
  AWS documents cache result types and hit/miss concepts.
Documenatation: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/cache-statistics.html?utm_source=chatgpt.com

B2) Deploy change (simulate)
Students must update index.html content at origin (or change static file).

B3) After invalidation: prove cache refresh
Run invalidation for /static/index.html, then:

```
  curl -i https://chewbacca-growl.com/static/index.html | sed -n '1,30p'
```

Expected:
  x-cache is Miss or RefreshHit depending on TTL/conditional validation
  CloudFront standard logs define Hit, Miss, RefreshHit.
Documentation: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/standard-logs-reference.html?utm_source=chatgpt.com

Part C — Terraform "framework" (two options)
Option 1 (Recommended): Keep invalidations as manual runbook ops
  Terraform should not constantly invalidate on apply; that trains bad habits.

Option 2 (Advanced/Optional): "Terraform action" invalidation
  HashiCorp provides a CloudFront invalidation action (not a core resource) that creates invalidations and

waits.

Add file: lab2b_honors_plus_invalidation_action.tf

Part D — Incident Scenario (graded)
Scenario: “Stale index.html after deployment”
Symptoms:
users keep receiving old index.html which references old hashed assets
static asset caching works, but the HTML entrypoint is stale

Required student response:
Confirm caching (Age, x-cache)
Explain why versioning is preferred but why entrypoint sometimes needs invalidation
Invalidate /static/index.html only (not /*)
Verify new content served
Write a short incident note (2–5 sentences)

Part E — “Smart” upgrade (extra credit)
E1) Explain when not to invalidate
If the only changed files are versioned assets like:

/static/app.9f3c1c7.js
then invalidation is unnecessary. AWS recommends versioned names for frequent updates.
AWS Documentation: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Invalidation.html?utm_source=chatgpt.com

E2) Create “invalidation budget”
Students must state:
monthly invalidation path budget (e.g., 200)
allowed wildcard usage conditions
approval workflow for /*

Student Submission (Honors+)
Students submit:
1) CLI command used (create-invalidation) + invalidation ID Documentation: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/example_cloudfront_CreateInvalidation_section.html?utm_source=chatgpt.com
2) Proof of cache before + after (headers showing Age/x-cache) Documentation: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/standard-logs-reference.html?utm_source=chatgpt.com
3) A 1-paragraph policy:
“When do we invalidate?”
“When do we version instead?”
“Why is /* restricted?”