



AWS CloudFront

Verified 2026-10-24

The CDN in front of S3 or any origin — behaviors, cache and origin request policies, TTLs, invalidation, OAC, edge code, and signed URLs.

Contents

Mental model	3
Distributions, origins, behaviors	3
How a request flows	5
Cache policy: the cache key and TTLs	6
Origin request policy: forward without caching on it	7
Private S3 origins with OAC	9
Invalidation and versioned names	10
Edge code: Functions vs Lambda@Edge	12
Errors, HTTPS, and custom domains	13
Signed URLs and cookies	15

Limits worth knowing	16
Further reading	16

Mental model

CloudFront is a cache in front of your origin, not a faster copy of it. A viewer's request goes to a nearby edge location, which builds a **cache key** from the request. Same key, cached copy: served without touching the origin. Anything you leave out of the cache key is ignored when deciding what to serve, so one cached object answers every request that differs only in those parts.

Distributions, origins, behaviors

Piece	Role
Distribution	The deployment; gets a *.cloudfront.net domain
Origin	Where content comes from: S3, ALB, API, any HTTP server
Behavior	A path pattern mapped to one origin and its settings

Piece	Role
Default behavior	The <code>*</code> pattern; matches when nothing else does

A request is matched against behaviors in precedence order, and the first matching path pattern wins. Put specific patterns such as `/api/*` above general ones. Each behavior picks its origin, allowed methods, cache policy, viewer protocol, and attached functions. Changes take a few minutes to deploy to all edges.

```
aws cloudfront list-distributions \
  --query "DistributionList.Items[.
    [Id,DomainName]"
aws cloudfront get-distribution-config --id E123
```

Gotcha: The default root object applies only to `/`. A request for `/docs/` does not return `/docs/index.html`; rewrite it with a function.

How a request flows

Step	What happens
Viewer request	Edge receives it, runs viewer-request code
Cache lookup	Cache key built; hit returns here
Origin request	On a miss, edge fetches from the origin
Origin response	Cached per TTL, then returned to the viewer

Check the X-Cache response header:

Hit from cloudfront, Miss from cloudfront, RefreshHit from cloudfront (revalidated with the origin), or Error from cloudfront. The Age header is seconds spent in cache, and X-Amz-Cf-Pop names the edge that answered.

```
curl -sI https://dxxxx.cloudfront.net/app.js \  
| grep -iE "x-cache|age|x-amz-cf-pop"
```

Cache policy: the cache key and TTLs

Managed policy	Behavior
CachingOptimized	No query strings or cookies in the key; 1 s min, 24 h default, 1 y max
CachingDisabled	TTLs all 0; nothing cached
UseOriginCacheControlHeader	Default TTL 0; the origin's Cache-Control decides

A cache policy sets which headers, cookies, and query strings join the cache key, plus the minimum, default, and maximum TTL. If the origin sends

`Cache-Control: max-age`, CloudFront uses it, clamped between your min and max. With no header, the default TTL applies. Include only what changes the response: every extra key part lowers the hit ratio.

CachingOptimized

658327ea-f89d-4fab-a63d-7e88639e58f6

CachingDisabled

4135ea2d-6df8-44a3-9df3-4b5a84be39ad

Gotcha: CachingOptimized has a 1 s minimum TTL, so CloudFront caches responses even when the origin sends no-store , no-cache , or private . For dynamic paths use CachingDisabled .

Origin request policy: forward without caching on it

Policy part	Effect
Cache policy values	Sent to the origin and used in the key
Origin request policy values	Sent to the origin, not in the key

Policy part	Effect
AllViewer	Forward all viewer headers, cookies, query strings
AllViewerExceptHostHeader	Same, but the origin sees its own host

Use an origin request policy when the origin needs a value but the response does not vary by it, such as an auth header or `CloudFront-Viewer-Country`. A cache policy sets what varies the cache; an origin request policy only adds what is sent.

Gotcha: Forwarding the viewer Host header to an S3 origin or an API Gateway endpoint breaks the request, usually as 403 or SignatureDoesNotMatch. Use AllViewerExceptHostHeader.

Private S3 origins with OAC

Item	Value
Mechanism	Origin access control (OAC) signs requests with SigV4
Bucket policy principal	<code>cloudfront.amazonaws.com</code>
Scoped by	<code>AWS:SourceArn</code> = the distribution ARN
Legacy	Origin access identity (OAI); use OAC for new work

Keep the bucket private and let only your distribution read it. Add `s3:ListBucket` for the distribution too, or a missing object returns `403` instead of `404`.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "cloudfront.amazonaws.com"
  },
  "Action": "s3:GetObject",
```

```
"Resource": "arn:aws:s3:::my-bucket/*",
"Condition": {"StringEquals": {
  "AWS:SourceArn":
    "arn:aws:cloudfront::ACCT:distribution/E123"
}}
}
```

Gotcha: A static website endpoint

(`bucket.s3-website-...`) is a custom origin, and OAC does not work with it. Use the bucket's REST endpoint (`bucket.s3.region.amazonaws.com`).

Invalidation and versioned names

Approach	Use for
Versioned file names	Assets: <code>app.3f9a1c.js</code>
Invalidate a path	A file that must change now
Invalidate <code>/*</code>	Last resort for a full refresh
Short TTL on HTML	Entry pages that reference hashed assets

AWS recommends file versioning as the primary way to update content. A new name is a new cache key, so there is nothing to purge, and browser caches are bypassed too. Invalidation removes objects from edge caches only. Invalidation paths must start with / , and * is allowed only at the end.

```
aws cloudfront create-invalidation \  
  --distribution-id E123 \  
  --paths "/index.html" "/img/*"
```

Gotcha: An invalidation does not clear the viewer's browser cache. If you sent `max-age=31536000` , users keep the old file; this is why versioned names beat invalidating.

Edge code: Functions vs Lambda@Edge

	CloudFront Functions	Lambda@Edge
Runs on	Viewer request, viewer response	Viewer and origin, request and response
Language	JavaScript (CloudFront JS runtime)	Node.js, Python
Size limit	10 KB, 2 MB memory	50 MB zip
Network calls	No	Yes, 30 s timeout

Pick Functions for cheap, fast request work: redirects, URL rewrites, header edits, cache-key normalization. Pick Lambda@Edge when you need network access, a larger runtime, or origin-side events. Lambda@Edge functions are authored in `us-east-1` and replicated to edges.

```
function handler(event) {  
  var req = event.request;  
  if (req.uri.endsWith("/")) {  
    req.uri += "index.html";  
  }  
  return req;  
}
```

Gotcha: A viewer-request function runs on every request, including cache hits. An origin-request function runs only on a miss, so use it for slow work.

Errors, HTTPS, and custom domains

Setting	Detail
Custom error response	Map 403 / 404 to a page and a status
Error caching min TTL	How long CloudFront caches an error

Setting	Detail
ACM certificate	Must be issued in us-east-1
Alternate domain name	Add the CNAME plus a matching certificate

A single-page app on S3 needs `403` and `404` mapped to `/index.html` with response code `200`, so client-side routes load. Point your DNS name at the distribution with a Route 53 alias record, not a plain CNAME at the zone apex. Set the viewer protocol policy to `redirect-to-https`.

Gotcha: Mapping errors to `/index.html` also masks real failures, such as a bad deploy that returns `403` for every file. Test with `curl -I` against the `cloudfront.net` domain.

Signed URLs and cookies

Option	Use for
Signed URL	One file, or a client that cannot send cookies
Signed cookies	Many files, such as a whole video stream
Trusted key group	Holds the public keys CloudFront verifies against
Custom policy	Adds an IP range or a start time

You create a key pair, upload the public key to a key group, then require signed access on the behavior. Your app signs with the private key; CloudFront checks the signature and expiry. It is separate from S3 presigned URLs and works with any origin.

Note: Signed access controls who may fetch from CloudFront. It does not stop a hot link from a valid, unexpired URL.

Limits worth knowing

Limit	Default
Distributions per account	500
Behaviors per distribution	75
Max cacheable GET response	50 GB
Origin response timeout	1–120 s
Compressed file sizes	1,000 to 10,000,000 bytes
URL length	8,192 bytes

Most are raisable through Service Quotas. Compression covers only files in the listed size range and needs the cache policy's compressed-objects setting on. The managed policies above enable it.

Further reading

- [CloudFront Developer Guide](#)
- [Managed cache policies](#)
- [CloudFront quotas](#)
- [CloudFront pricing](#)