# Amazon CloudFront

If you are studying for AWS Developer Associate Exam, this guide will help you with quick revision before the exam. it can use as study notes for your preparation.

Dashboard | Other Certification Notes

## Amazon CloudFront



## CloudFront

- Content Delivery Network (CDN)
- **Improved read performance by caching the static content at the edge locations**
- Improves users experience
- 216 Point of Presence globally (edge locations)
- **DDoS protection (because worldwide), integration with Shield, AWS Web Application Firewall**
- We can expose HTTPS end-point by loading certificates onto CloudFormation
- CloudFormation can talk with internal services using HTTPS as well

## CloudFront - Origins

- **S3 bucket**
  - For distributing files and caching them at the edge
  - Enhanced security with CloudFront **Origin Access Control (OAC)**
  - OAC is replacing Origin Access Identity(OAI)
  - CloudFront can be used as an ingress (to upload files to S3)
- **Custom Origin (HTTP)**
  - Application Load Balancer
  - EC2 instance
  - S3 website (must first enable the bucket as a static S3 website)
  - Any HTTP backend you want

### Restrict access to S3

- To restrict access to content that we serve from Amazon S3 buckets, we can follow these steps:
  1. Create a special CloudFront user called an origin access identity (OAI) and associate it with

our distribution.
2. Configure your S3 bucket permissions so that CloudFront can use the OAI to access the files in our bucket and serve them to our users. Make sure that users can't use a direct URL to the S3 bucket to access a file there.

## CloudFront vs S3 Cross Region Replication

| CloudFront | S3 Cross Region Replication |
|---|---|
| Global Edge network | Must be setup for each region you want replication to happen |
| Files are cached for a TTL (maybe a day) | Files are updated in near real-time |
| **Great for static content that must be available everywhere** | Read only, **Great for dynamic content that needs to be available at low-latency in few regions** |

## CloudFront Caching

- The cache lives at each CloudFront **Edge Location**
- CloudFront identifies each object in the cache using the **Cache Key** (see next slide)
- You want to maximize the Cache Hit ratio to minimize requests to the origin
- You can invalidate part of the cache using the **CreateInvalidation** API

## What is CloudFront Cache Key?

- A unique identifier for every object in the cache
- By default, consists of **hostname + resource portion of the URL**
- If you have an application that serves up content that varies based on user, device, language, location…
- You can add other elements (HTTP headers, cookies, query strings) to the Cache Key using **CloudFront Cache Policies**

## CloudFront Policies - Cache Policy

- Cache based on:
  - **HTTP Headers:** None – Whitelist
  - **Cookies:** None – Whitelist – Include All-Except – All
  - **Query Strings:** None – Whitelist – Include All-Except – All
- Control the TTL (0 seconds to 1 year), can be set by the origin using the **Cache-Control** header, **Expires** header…
- Create your own policy or use Predefined Managed Policies
- **All HTTP headers, cookies, and query strings that you include in the Cache Key are automatically included in origin requests**

### CloudFront Caching - Cache Policy HTTP Headers

- **None:**
  - Don't include any headers in the Cache Key (except default)
  - Headers are not forwarded (except default)
  - Best caching performance
- **Whitelist:**
  - **only specified headers** included in the Cache Key
  - Specified headers are also forwarded to Origin

### CloudFront Cache - Cache Policy Query Strings

- **None**
  - Don't include any query strings in the Cache Key • Query strings are not forwarded
- **Whitelist**
  - Only specified query strings included in the Cache Key
  - Only specified query strings are forwarded
- **Include All-Except**
  - Include all query strings in the Cache Key except the specified list
  - All query strings are forwarded except the specified list
- **All**
  - Include all query strings in the Cache Key
  - All query strings are forwarded
  - Worst caching performance

### CloudFront Policies - Origin Request Policy

- Specify values that you want to include in origin requests **without including them in the Cache Key (no duplicated cached content)**
- You can include:

- You can include:
  - **HTTP headers:** None – Whitelist – All viewer headers options
  - **Cookies:** None – Whitelist – All
  - **Query Strings:** None – Whitelist – All
- Ability to add CloudFront HTTP headers and Custom Headers to an origin request that were not included in the viewer request
- Create your own policy or use Predefined Managed Policies

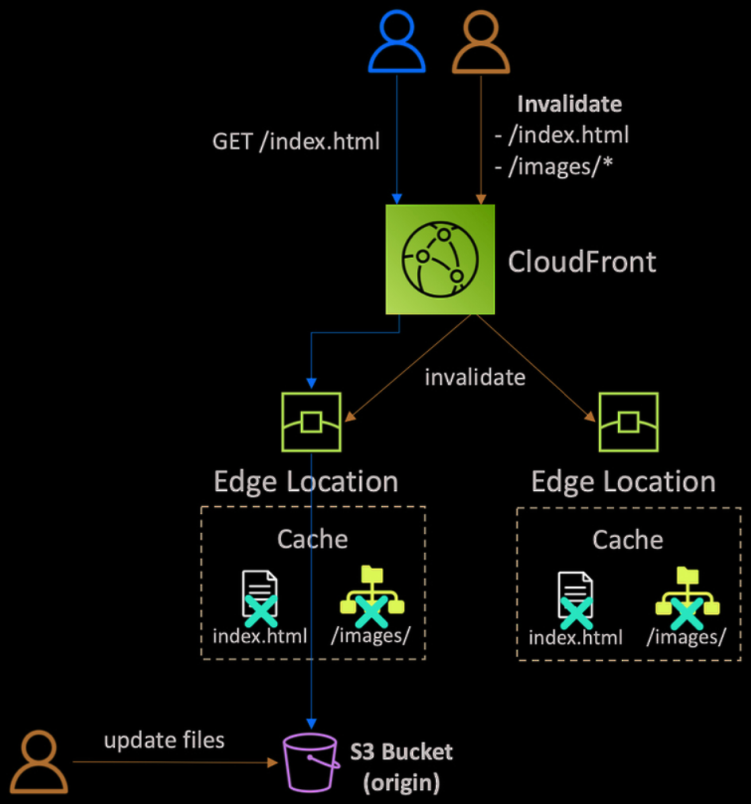
## Cache Policy vs. Origin Request Policy

| Category | Cache Policy | Origin Request Policy |
|---|---|---|
| Purpose | Controls how CloudFront caches and serves content | Controls which requests CloudFront sends to the origin server |
| Configuration | Can be created and configured at the distribution level | Can be created and configured at the cache behavior level |
| Actions | Determines whether to cache content, for how long, etc. | Determines which requests should be forwarded to the origin |
| Caching flexibility | Provides granular control over caching behavior | Provides less granular control over caching behavior |
| Customization | Can be customized based on HTTP headers, query strings, etc. | Can be customized based on various conditions and criteria |
| Examples of actions | Cache based on path pattern, query strings, headers, etc. | Block requests based on IP address, user agent, or referrer |

Overall, Cache Policy controls how content is cached and served by CloudFront, while Origin Request Policy controls which requests are forwarded to the origin server. Both policies can be customized and provide various levels of control over caching and request handling.

## CloudFront - Cache Invalidations

- In case you update the back-end origin, CloudFront doesn't know about it and will only get the refreshed content after the TTL has expired
- However, you can force an entire or partial cache refresh (thus bypassing the TTL) by performing a **CloudFront Invalidation**
- You can invalidate all files (*) or a special path (/images/*)



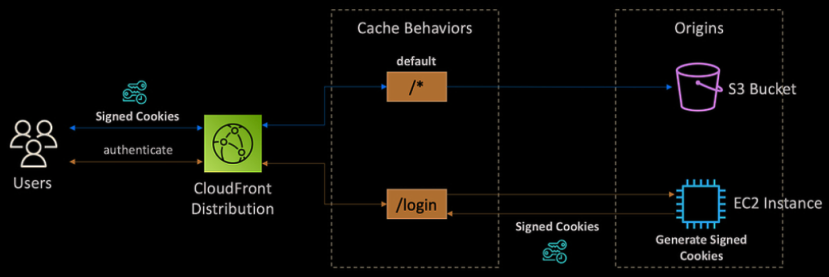

## CloudFront - Cache Behaviors

- Configure different settings for a given URL path pattern
- Example: one specific cache behavior to **images/*.jpg** files on your origin web server
- Route to different kind of origins/origin groups based on the content type or path pattern

- /images/*
- /api/*
- /* (default cache behavior)

- When adding additional Cache Behaviors, the Default Cache Behavior is always the last to be processed and is always /*

### CloudFront - Cache Behaviors - Sign In Page




## CloudFront Geo Restriction

- You can restrict who can access your distribution
  - **Allowlist**: Allow your users to access your content only if they're in one of the countries on a list of approved countries.
  - **Blocklist**: Prevent your users from accessing your content if they're in one of the countries on a list of banned countries.
- The "country" is determined using a 3rd party Geo-IP database
- Use case: Copyright Laws to control access to content

## CloudFront Signed URL / Signed Cookies

- You want to distribute paid shared content to premium users over the world
- We can use CloudFront Signed URL / Cookie.We attach a policy with:
  - Includes URL expiration
  - Includes IP ranges to access the data from
  - Trusted signers (which AWS accounts can create signed URLs)
- How long should the URL be valid for?
  - Shared content (movie, music): make it short (a few minutes)
  - Private content (private to the user): you can make it last for years
- Signed URL = access to individual files (one signed URL per file)
- Signed Cookies = access to multiple files (one signed cookie for many files)

### CloudFront Signed URL vs S3 Pre-Signed URL

| CloudFront Signed URL | S3 Pre-Signed URL |
|---|---|
| Allow access to a path, no matter the origin | Issue a request as the person who pre-signed the URL |
| Account wide key-pair, only the root can manage it | Uses the IAM key of the signing IAM principal |
| Can filter by IP, path, date, expiration | Limited lifetime |
| Can leverage caching features | |

### CloudFront Signed URL Process

- Two types of signers:
  - Either a trusted key group (recommended)
    - Can leverage APIs to create and rotate keys (and IAM for API security)
  - An AWS Account that contains a CloudFront Key Pair
    - Need to manage keys using the root account and the AWS console
    - **Not recommended** because you shouldn't use the root account for this
- In your CloudFront distribution, create one or more **trusted key groups**
- You generate your own public / private key
  - The private key is used by your applications (e.g. EC2) to sign URLs
  - The public key (uploaded) is used by CloudFront to verify URLs

## CloudFront - Pricing

- CloudFront Edge locations are all around the world
- The cost of data out per edge location varies

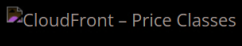
Hong Kong,
Indonesia,

| Per Month | United States, Mexico, and Canada | Europe and Israel | South Africa, Kenya, and Middle East | South America | Japan | Australia and New Zealand | Philippines, Singapore, South Korea, Taiwan, Thailand, and Vietnam | India |
|---|---|---|---|---|---|---|---|---|
| First 10TB | $0.085 | $0.085 | $0.110 | $0.110 | $0.114 | $0.114 | $0.120 | $0.109 |
| Next 40TB | $0.080 | $0.080 | $0.105 | $0.105 | $0.089 | $0.098 | $0.100 | $0.085 |
| Next 100TB | $0.060 | $0.060 | $0.090 | $0.090 | $0.086 | $0.094 | $0.095 | $0.082 |
| Next 350TB | $0.040 | $0.040 | $0.080 | $0.080 | $0.084 | $0.092 | $0.090 | $0.080 |
| Next 524TB | $0.030 | $0.030 | $0.060 | $0.060 | $0.080 | $0.090 | $0.080 | $0.078 |
| Next 4PB | $0.025 | $0.025 | $0.050 | $0.050 | $0.070 | $0.085 | $0.070 | $0.075 |
| Over 5PB | $0.020 | $0.020 | $0.040 | $0.040 | $0.060 | $0.080 | $0.060 | $0.072 |

Source

### CloudFront – Price Classes

- You can reduce the number of edge locations for cost reduction
- Three price classes:
  1. Price Class All: all regions – best performance
  2. Price Class 200: most regions, but excludes the most expensive regions
  3. Price Class 100: only the least expensive regions




Source

## CloudFront - Multiple Origin

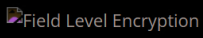
- To route to different kind of origins based on the content type
- Based on path pattern:
- /images/* → S3
- /api/* → Application Load Balancer
- /* → S3

## CloudFront - Origin Groups

- To increase high-availability and do failover
- Origin Group: one primary and one secondary origin
- If the **primary origin fails**, the second one is used

## CloudFront - Field Level Encryption

- Protect user sensitive information through application stack • Adds an additional layer of security along with HTTPS
- Sensitive information encrypted at the edge close to user
- Uses asymmetric encryption
- Usage:
  - Specify set of fields in POST requests that you want to be encrypted (up to 10 fields)
  - Specify the public key to encrypt them




## CloudFront - Real Time Logs

- Get real-time requests received by CloudFront sent to Kinesis Data Streams
- Monitor, analyze, and take actions based on content delivery performance
- Allows you to choose:
  - Sampling Rate – percentage of requests for which you want to receive
  - Specific fields and specific Cache Behaviors (path patterns)