

Amazon CloudFront for Media

Best Practices for Streaming Media Delivery

November 2020



Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2020 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Contents

- Introduction 1
- Streaming techniques for media 1
 - Common elements of streaming technologies 2
- Architecture of media workflows 4
 - Encoding and packaging 6
 - The origin 6
 - The content delivery network 7
 - Reference solutions 7
- CDN features for media delivery 8
 - Architecture 8
 - Request handling and protocol optimizations 9
 - CloudFront configuration best practices 13
 - Cost optimization 16
 - CloudFront Origin Shield 16
- Security 17
 - Geographical restriction 17
 - Access control through CloudFront 18
 - Session uniqueness 21
- Multi-CDN considerations 21
- Operational monitoring and reporting 23
- Best practices for events and service launches 25
- Conclusion 26
- Contributors 27
- Document revisions 27

Abstract

This whitepaper is for media organizations interested in delivering streaming media content to their viewers using Amazon CloudFront. Media delivery has a unique set of characteristics requiring low latency, high reliability, and high scalability. This whitepaper discusses live streaming and video on-demand workflows, as well as streaming techniques, content delivery networks (CDNs), and why CDNs are important for media delivery. It also includes best practices for end-to-end system design using Amazon CloudFront and AWS Media Services.

Introduction

This whitepaper is a guide for using Amazon CloudFront to successfully deliver high-quality media experiences at scale. It is intended for media architects, streaming architects, network architects, and CDN operations teams.

Delivering media content over the internet at scale poses a unique set of challenges for architects and platform operators. End customers expect the highest-quality experience with reliable delivery, low startup latency, and the ability to choose from a wide range of content. Live streaming magnifies the challenges of maintaining consistent low latency and reliable delivery, with sharp peaks in demand for popular content. Media operators also need to ensure that content is secure in order to protect rights holders as well as their own revenue and reputation.

Amazon CloudFront is a CDN platform that securely delivers video, data, applications, and API operations to customers globally with low latency, high transfer speeds, and with a developer-friendly environment. CloudFront is integrated with AWS services and physical locations that are directly connected to the [AWS Global Infrastructure](#). Media workloads in this context include live streaming and video on demand (VOD) content delivered over the internet.

In both scenarios, media platforms use streaming protocols to allow end customers to view the media without having to download the content in its entirety. Within live streaming, we include events and 24/7 channels. Delivery for linear TV platforms, or closed networks, have slightly different considerations. But share some similarity with the media architectures described in this whitepaper. To discuss how AWS enables these types of solutions, [contact an Amazon Web Services sales representative](#).

Media delivery requires an end-to-end systems approach. In this guide, we start by explaining the architecture of the media workloads. Then we explain the streaming technologies. Finally, we describe a CDN configuration in detail.

Streaming techniques for media

Media streaming involves using one or more techniques to deliver video over the internet. All the streaming technologies described in this whitepaper rely on adaptive bitrate (ABR) streaming. ABR is where the same video content is provided at multiple bitrates. This allows the best available video quality for the viewer.

The ABR experience is superior to delivering a static video file at a single bitrate. This is because the video stream can be switched midstream to be as good as the available network bandwidth allows. This ability to switch avoids buffering or interruption in playback that can happen when client's network speed can't support a particular bitrate.

All streaming techniques divide a video stream into a series of smaller segments. This allows a player to begin video playback as soon as possible after receiving data. Streaming techniques for the internet differ from streaming over closed networks because there is no guaranteed bitrate available at the client, and they must be able cope with changes in the available bitrate as seen by the viewer.

All of the different streaming techniques use Hypertext Transfer Protocol (HTTP) to distribute content. However, there are many differences in when and how these technologies might be used based on the viewers devices. There is often a need to use one or more of these streaming techniques based on delivery to variety of different devices including smart TVs, set-top boxes (STBs), personal computers (PCs), and handheld devices, such as tablets and smartphones.

There are two typical workflows for streaming:

- **Video on demand (VOD)** – In this workflow, the media assets are stored as encoded video files and played on request. The media assets are stored in multiple different formats and bitrates for playout, or packaged at the time of requesting, using just-in-time packaging (JIT), for different client devices.
- **Live streaming** – This workflow describes source content for TV channels and live events, such as sport, concerts, news, or any other live broadcast events. This content is packaged in real time, in multiple different formats for different client devices.

Video on demand and live streaming can both be delivered using one or more different streaming technologies. The major advantage of streaming for VOD applications is that a viewer does not have to download the whole file before starting to play it.

Common elements of streaming technologies

The streaming technologies all use similar underlying mechanisms with detailed differences across the implementations. The common elements are:

- HTTP is used for video and signaling to provide compatibility with networks, firewalls, caches and end clients.

- Video is provided as a set of concurrent streams with different bitrates and resolutions.
- Each video stream is divided into a series of files, each containing a few seconds of video.
- The list of available streams is obtained by downloading an index or manifest file.
- The client starts with the first stream in the list, downloads the first segment, and starts to play when downloaded.
- The client continually downloads the next segment while the current one is playing, giving the appearance of a continuous video stream.
- The client chooses which video stream to use for the next segment, allowing it to adapt to immediate network conditions on a segment-by-segment basis.

HTTP Live Streaming

HTTP Live Streaming (HLS)¹, was developed by Apple and is widely supported in iOS and Android devices. Segments are 2–10 seconds long and the list of streams is provided in a manifest file with .m3u8 extension. HLS supports H.264/AVC and HEVC/H.265 video codecs.

Dynamic Adaptive Streaming over HTTP

Dynamic Adaptive Streaming over HTTP (DASH or MPEG-DASH)² was developed collaboratively as an international standard for adaptive streaming technology. The DASH specification is designed to be extensible and support both current and future video codecs. Implementations have focused on the most commonly used codecs including H.264 and H.265 (HEVC).

Common Media Application Format

The Common Media Application Format (CMAF) is somewhat broader in scope than HLS or DASH. It is also the result of an international standardization effort. CMAF specifies a set of *tracks* that allow clients to identify and select media objects.

A key difference between CMAF and HLS or DASH is that segments are further subdivided into *fragments*. For example, a segment of eight seconds would be divided into four fragments of two seconds each. With CMAF, a client can be playing a fragment while still downloading other fragments from the same segment. This ability allows for a

much lower latency, compared to HLS or DASH, because it is not necessary to download an entire segment before starting to play it.

CMAF is defined by ISO Standard ISO/IEC 23000-19³ and its application for HLS⁴ is defined in Apple documentation. The application of CMAF for achieving low latency is described in more depth in the AWS Media Blog post, [Lower latency with AWS Elemental MediaStore chunked object transfer](#).

Microsoft Smooth Streaming

Microsoft Smooth Streaming (MSS) was first launched in 2008 by Microsoft and remains proprietary technology defined in the MS-SSTR specification⁵. MSS uses an index file to list streams. MSS is widely used in legacy platforms.

Architecture of media workflows

Media workflows are categorized as either live streaming or on-demand services. The difference is based on whether the input is being captured in real time (live) or played out from a stored file (on-demand). In both cases, these services rely on the streaming technologies described in the previous section to enable delivery to the player on the client devices.

The key media infrastructure components are:

- Live encoding service
- A store of video on-demand files
- Streaming origin
- Content distribution network

The following image shows a typical example architecture with relevant AWS services:

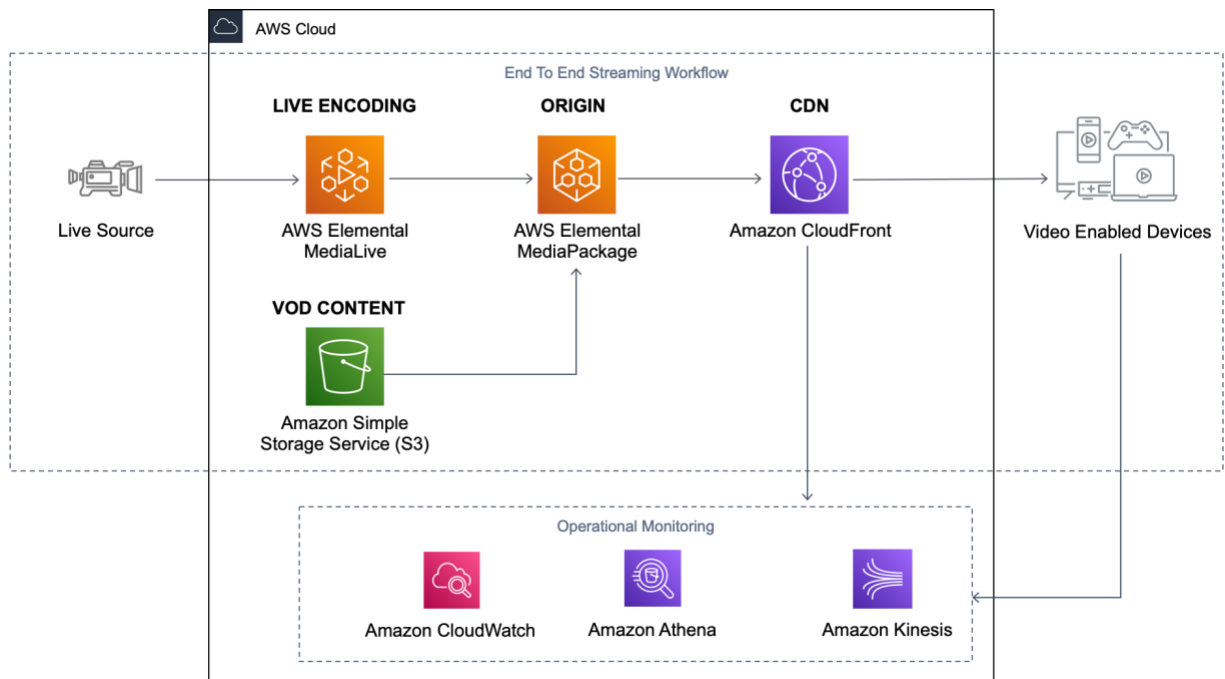


Figure 1 – Example end-to-end media architecture

Live workflows typically include a live source from a studio, venue or other location, and 24/7 channels with a mix of live and recorded content. Some 24/7 channels use entirely file-based assets played out on a schedule. VOD content is prepared in advanced using a transcoder, such as AWS Elemental MediaConvert.

A key factor in user experience is the available bandwidth on the viewer's internet connection. Insufficient bandwidth, or sudden network congestion, can cause stuttering and delays while the player is buffering. Viewers report that this is the biggest negative impact on their perception of the quality of a video service. For the system architect, the available network bandwidth provides the key constraint on the overall architecture, including both provider networks and mobile or wireless connections.

One solution is to reduce or vary the video bitrate, but even this has challenges. The bigger the display, the higher the resolution required to provide the same user experience. And the higher the resolution, the higher the bitrate required to sustain it. This is why streaming technologies provide a set of video streams and the ability to switch between them dynamically.

Encoding and packaging

The video content has to be encoded (live) or transcoded (on-demand) several times in parallel to provide a set of streams at different bitrates. This group of video streams is sometimes known as an *ABR ladder*. The ABR ladder gives the client the ability to autonomously change streams to adapt to network conditions.

Once the video has been encoded into multiple resolutions it is packaged into one or more streaming protocols. This is the point where the video encoding and packaging process is complete and the video can be requested by a client from the origin.

Packaging and storing video files can be a complex management challenge. The packaging function can be provided by the video encoder, as part of the VOD content preparation process or by the origin. Many services use just-in-time packaging, where client demand drives the packaging process in real time. Ad insertion will usually take place in this stage, and is known as server-side ad insertion (SSAI). Because of the 1:1 relationship between video streams and viewers, many platforms insert personalized ads into the video stream, providing relevant content for viewers and targeted advertising opportunities for advertisers.

Another key factor in video encoding configuration for live streams is latency. Viewers want to see the content with the lowest delay from the source, while maintaining the highest picture quality.

From a system perspective, the solutions architect will work backwards from the population of client devices and select a set of streaming technologies, video resolutions, and bitrates to meet their needs.

The origin

The origin is the endpoint where viewer requests are received and where streaming video files are served for both live and on-demand workflows. You have several different origin options to choose from or you build a custom implementation to meet your specific needs.

- Amazon S3 is a great place to start as an origin for VOD content. It only requires that the content is packaged and stored in advance.
- AWS Elemental MediaStore is optimized for large-scale live video events and low latency with content packaged in advance.

- AWS Elemental MediaPackage provides just-in-time (JIT) packaging for both live and VOD workflows. MediaPackage provides the most flexible option for streaming in multiple formats.
- APN Partners also provide origin solutions, including alternative packagers.

The content delivery network

We identified earlier that the available network bandwidth, reliability, and low latency are key factors in the design and success of streaming video services. Streaming requires individual video delivery to every viewer's device, placing a much higher load on the streaming platform than traditional one-to-many broadcast platforms. The growth in streaming services is magnified by the increasing number of video-enabled devices that we own and use. How do video platforms support these large numbers of individual streams without having to scale their streaming platform to match?

The answer is to add a scalable distributed caching layer between the origin and the viewers, cache the video segment files when they are first requested, and then serve the cached files for subsequent requests. In this way, a provider can support a constantly increasing number of viewers without scaling the streaming platform itself, while maintaining a consistent low latency, even where the viewer is in a different geographic Region to the origin. The technology solution that provides this caching platform is the content delivery network (CDN).

CDNs work very efficiently for live and on-demand video streams. In CloudFront, you create a CloudFront *distribution*, with a caching configuration, dedicated entry point domain name, and a specified origin for streaming video. You can create different CloudFront distributions for different applications.

Reference solutions

The [AWS Solutions Library](#) provides reference implementations for media workflows. The [Live Streaming on AWS](#) and [Video On Demand on AWS](#) solutions allow you to get started quickly with AWS Media Services using an AWS CloudFormation template and a deployment guide. You can deploy these solutions in your AWS account and use them right away, adapting the templates for your own needs or using them as a reference for designing your own applications.

CDN features for media delivery

Amazon CloudFront supports video streaming workflows, using its highly scalable internal architecture, the numerous edge locations around the world, and a range of optimization techniques for processing viewer requests efficiently.

Architecture

CloudFront has a global footprint with more than 200 edge locations in over 40 countries across five continents, providing global access for your viewers. AWS continues to extend CloudFront based on growth and anticipated customer needs. Availability is one of the high-priority design tenets of CloudFront. Metropolitan areas have the highest concentration of traffic, and CloudFront provides multiple edge locations for scale and performance. These locations are deployed in different facilities to provide a high degree of resilience. A cluster of edge locations in a single area gives CloudFront the ability to route viewers quickly to another location in close proximity without noticeable performance impact.

CloudFront edge locations have multiple connections to local internet service providers (ISPs) and global carriers through internet exchanges and direct private fiber connections. This minimizes video delivery latency, reduces probability of congestion and traffic loss, and provides high availability. Edge locations also leverage the [AWS global network](#), which connects AWS Regions and edge locations.

The AWS global network provides high bandwidth, resilience and redundancy at scale. This gives you consistent performance, high availability and also shields your viewers from internet instabilities and changing conditions.

The quality of the connection from the origin to the edge location is just as important as the proximity of the edge location to the viewer, providing low latency and avoiding re-buffering, which is a factor in reducing viewer churn. AWS works closely with our customers to understand their current and future traffic patterns to guide further expansion with new edge locations and scaling of the existing locations. This can be particularly relevant when planning the launch of your video platform in a new Region or anticipating high peak events.

CloudFront's architecture includes a mid-tier caching layer between your origin and the edge locations. This allows you to scale further without a corresponding increase in load at the origin, and to maintain a high cache hit ratio. This mid-tier layer is known as a regional edge cache (REC). Upon a cache miss, an edge location will initiate a request

to its associated REC instead of going to the origin. The REC adds a layer of content consolidation where the volume of requests going to the origin is reduced as requests for the same content from different edge locations can be retrieved from the same REC. RECs can increase a video streaming object's retention time due to its larger cache storage and this provides another advantage of using CloudFront for large VOD video catalogs.

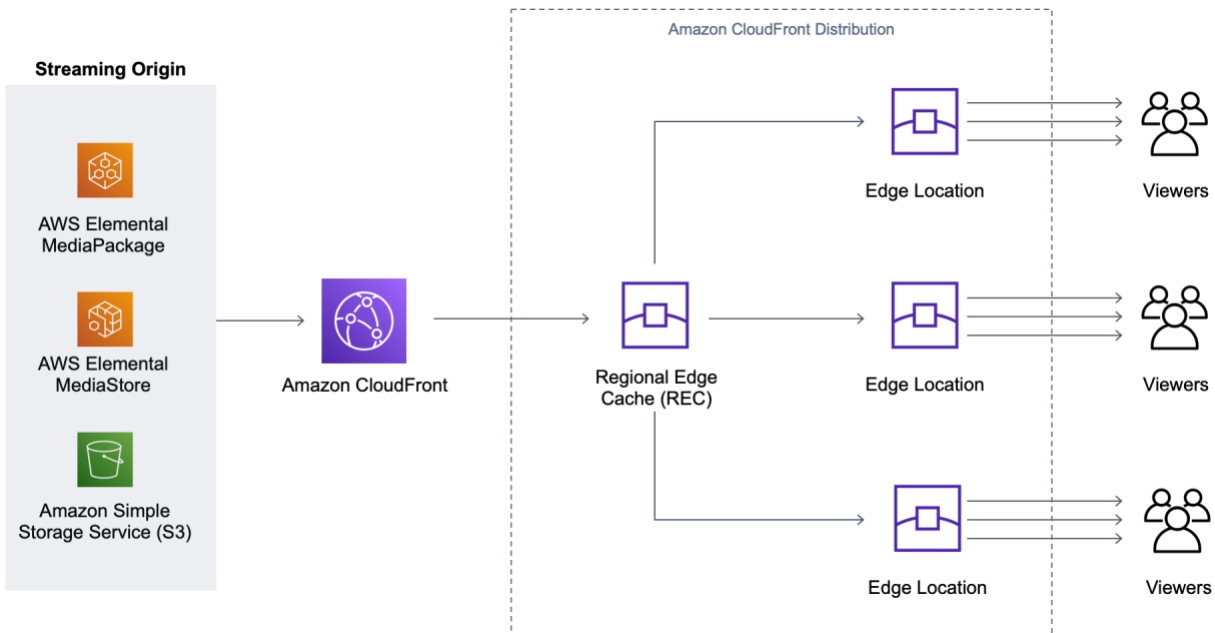


Figure 2 – Amazon CloudFront architecture

CDN requirements for media delivery often extend beyond performant and scalable delivery to the viewers. Media applications often adapt the content for different viewers or device platforms, provide security controls for paywall content, or add other customizations based on the request from the viewer. [Lambda@Edge](#) gives you the ability to run code close to the customer in order to inspect or modify viewer requests. Lambda@Edge is a low-latency serverless compute service integrated with CloudFront. For information about using Lambda@Edge in media delivery, see [How to filter live streaming renditions by device type at the Edge](#).

Request handling and protocol optimizations

Modern video delivery techniques rely predominantly on the HTTP protocol, with TLS and TCP operating at the session and transport layers for secure and reliable transmission. This set of protocols has the advantage of being supported by any internet connected device. One drawback is that these protocols were originally designed for web delivery, and add an overhead in establishing and revoking

connections before any data can be sent. The perceived quality of video streaming is impacted by how quickly the player receives requested video segments and the stability of the connection during the playback session. How the CDN handles connections on the viewer and origin side, as well as how it processes incoming requests and responses, has a direct impact on the perceived quality of the video stream.

CloudFront uses modern protocol extensions and acceleration techniques when processing requests to provide high performance in end-to-end delivery of media.

TCP Fast Open

The first step in delivering media content to the viewer involves establishing a TCP connection. This might need to be re-established multiple times during the playback session. TCP session establishment adds two round-trip times (RTT) of delay. By using TCP Fast Open, every subsequent connection can be established with only 1 RTT by means of a cryptographic cookie set during initial connection. TCP Fast Open is enabled by default on CloudFront.

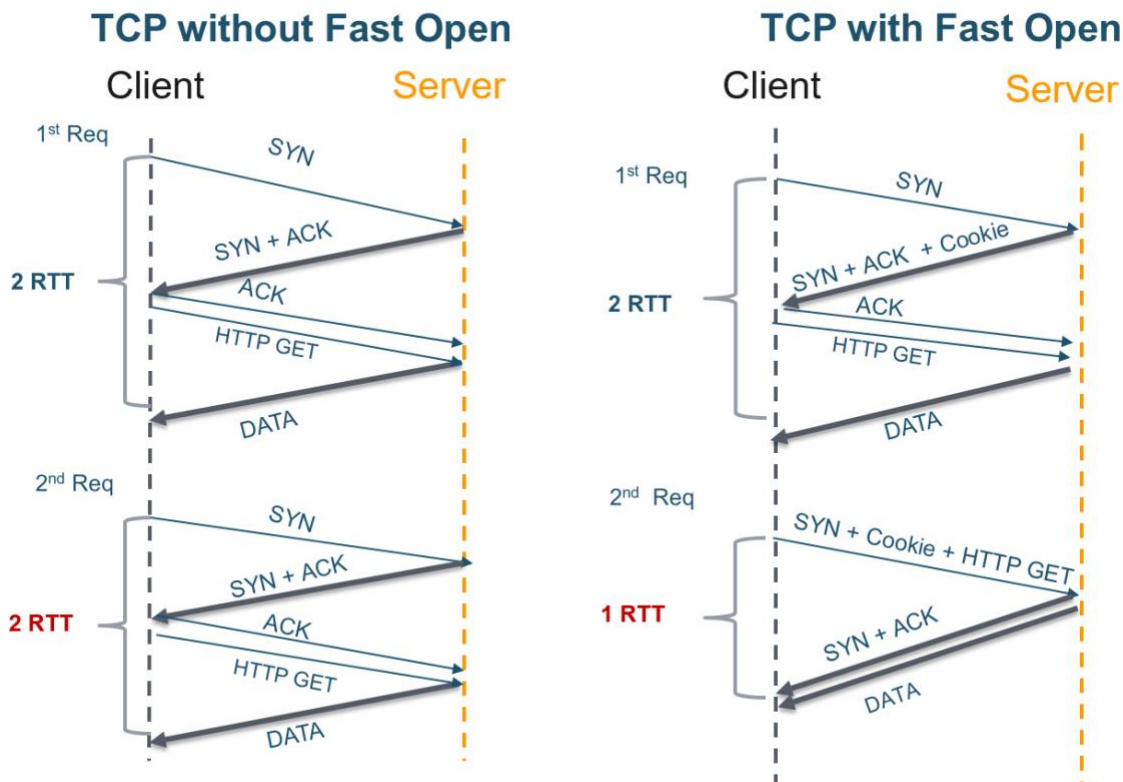


Figure 3 – Protocol optimization with TCP Fast Open

TLS session resumption

HTTPS adds a further overhead because setting up a secure communication channel, through either the SSL or TLS protocols, involves exchanging signaling messages in order to negotiate ciphering algorithms and agree on the common set of encryption keys. This can add an additional 2 RTTs in the initial stage of retrieving the media content. Once a viewer has set up a TLS session with CloudFront, using the full negotiation process, it can use the session ID to re-establish the TLS session with the same attributes negotiated during the initial session creation.

TLS Session Resumption is combined with the OSCP stapling mechanism, as described in [RFC6960](#) where a valid response is obtained regularly and attached to the handshakes. This reduces from 4 to 1 the overall number of round trips between the viewer and CloudFront required to exchange signaling messages prior to data transmission. CloudFront also supports TLS 1.3, which requires only 1 RTT to negotiate a new secure connection.

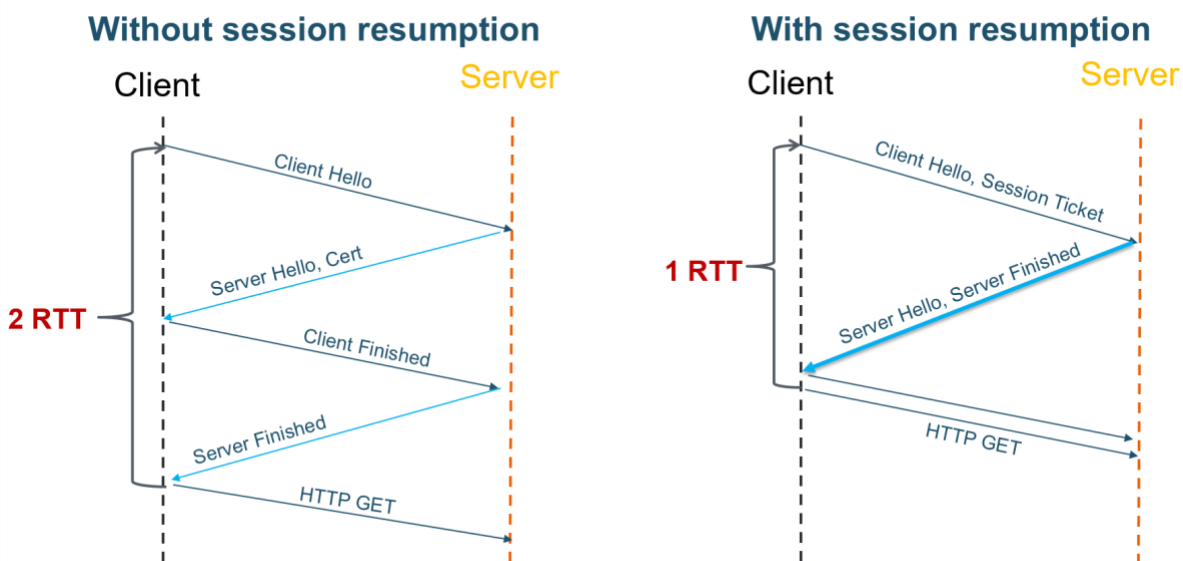


Figure 4 – Protocol optimization with TLS session resumption

Byte streaming

A common mode of operations for caching proxies is to fetch the entire object from the origin before sending it back to the viewer. This store-and-forward approach introduces an extra delay in delivery of new segments to the audience. This is undesirable, especially in live streaming where minimizing latency is important. CloudFront uses a pass-through approach when forwarding the newly retrieved objects from the origin to

minimize latency. This means that as soon the first byte from the origin reaches CloudFront, it will be immediately forwarded to the viewer. This is referred to as byte streaming.

Request collapsing

During a live event, or a release of any type of highly popular content, a large set of viewers will request the same video objects at the same time. If the video segment is not yet present in the cache, CloudFront retrieves it from the origin. If multiple requests for the same object are received before the object is cached, CloudFront sends a single request to the origin for the segment when first request comes in and uses that object to service all viewers. This removes the need to send multiple identical requests to the origin. This provides an additional performance benefit in that all requests after the first will experience lower latency. This is because the CloudFront cache is already in the process of retrieving the video segment from the origin.

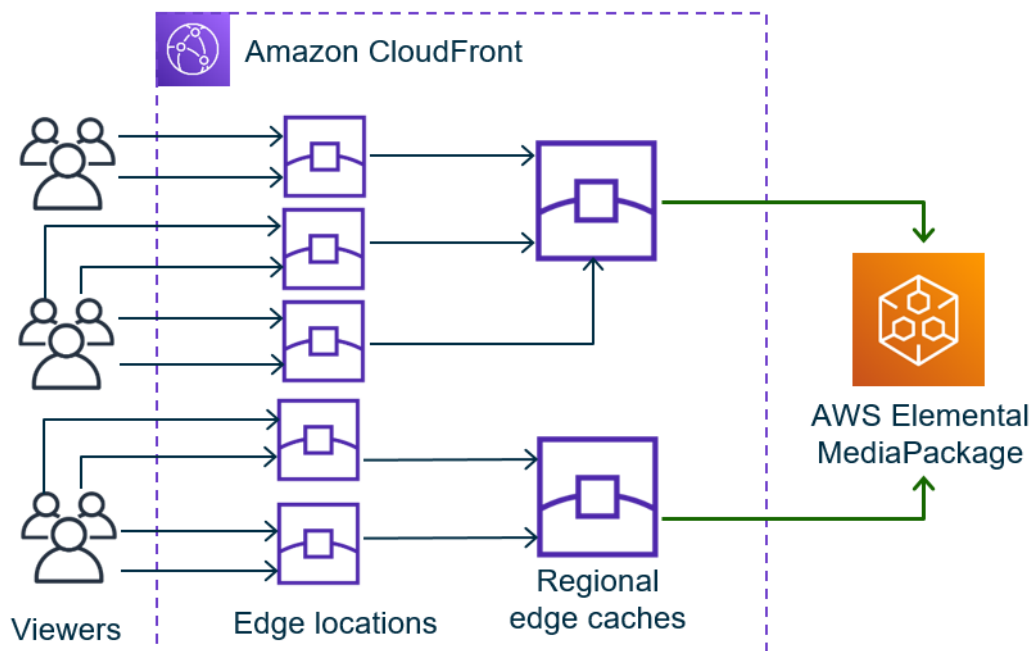


Figure 5 – Request collapsing in CloudFront

Persistent connections

Video delivery at scale requires a CDN to continuously populate its cache layer with the video objects requested by the viewers with as few interruptions as possible. In situations where a cache server needs to frequently re-establish connection to the origin, this holds back all the viewers waiting for the next video segment from the origin. To avoid this, CloudFront maintains persistent connections to the origin with a [configured timeout while idle](#). You can configure the keep-alive timeout in the CloudFront to align with a timeout value set on your origin server.

TCP congestion control

With Adaptive bitrate streaming, the client uses measured throughput to determine whether to choose a higher or lower bitrate video stream. Throughput is not only dependent on the connection bandwidth but also on how the TCP algorithm adjusts the congestion window during the data transfer. All TCP congestion control algorithms will gradually increase congestion window, resulting in higher throughput. Widely used congestion algorithms like CUBIC, Reno, and Vegas, result in high throughput fluctuation because they aggressively reduce the congestion window in response to packet loss detection. This effectively decreases the average throughput.

In 2019, CloudFront adopted Bottleneck Bandwidth and Round-trip propagation time (BBR) as a congestion control algorithm. This led to performance gains of up to 22% on the throughput. The BBR algorithm probes and measures end-to-end connection bandwidth and RTT to set optimal delivery rate to maximize throughput and minimize congestion. For more information, see [TCP BBR Congestion Control with Amazon CloudFront](#).

CloudFront configuration best practices

The cache hit ratio (CHR) is a key metric for CDNs. The CHR is defined as:

$$\text{Cache Hit Ratio (CHR)} = \frac{\text{requests served directly from the cache}}{\text{total requests}}$$

The CHR is usually expressed as a percentage over a measurement interval. Maximizing the CHR has a direct performance improvement because it implicates reducing response latency. Poor CHR results in more cache miss occurrences and brings a cost in terms of overall performance as well as increased origin load.

You can optimize the CHR by following the recommendations in the [Amazon CloudFront Developer Guide](#). These recommendations include:



- **Ensure the origin is sending a Cache-Control header with appropriate values.** In a live streaming scenario, expect the Origin setting *max-age* value close to video segment length for manifest files. For VOD workloads both manifest and video segments can be considered as static assets, therefore *max-age* can be set for longer period, to the extent of days or weeks. If managing cache-control directives is too complicated to be done on the origin level, CloudFront provides the possibility to set a [Minimum, Default and Maximum TTL](#). However, in this case it's advisable to create separate behaviors for video segments and manifest because they may require different TTL settings and cache key policies.
- **Reduce the number of parameters added to the cache key.** For example, avoid forwarding CORS headers simply to allow all incoming requests from different site domains. Instead, send it statically from the origin or add it dynamically using Lambda@Edge within an origin response event. If the origin is Amazon S3, MediaStore, or MediaPackage, you can trigger them to send the *Access-Control-Allow-Origin:** header by overriding an *Origin* header sent to your origin through CloudFront Origin Custom Headers.

Origin Custom Headers	Header Name	Value
	<input type="text" value="Origin"/>	<input type="text" value="*"/>

Figure 6 – Example CloudFront Origin Customer Headers

- **Remove the Accept-Encoding header from the cache key by disabling compression support in [cache policy](#).** The video content is compressed in the encoding process at the origin with adequate compression format for this type of content.

Long tail content

VOD content tends to be viewed most often when it is new. The frequency of requests from viewers usually reduces over time, until the point where the content is not usually present in the cache. This is known as *long tail* content.

Long tail content can still benefit from same CloudFront acceleration as dynamic content. You can optimize your architecture for long tail content by replicating the origin to multiple Regions and then routing requests to the nearest Region. This can be done in different ways according to the origin type:

- For Amazon S3 origins, you can use Lambda@Edge to process origin requests to detect in which Region it's being executed, and route the request to the nearest Region based on a static mapping table.
- For non-Amazon S3 origins, you can configure a domain name in Amazon Route 53 with a latency policy pointing to different Regions. Then you can configure an origin in CloudFront with this domain name.

Enabling [CloudFront Origin Shield](#) can extend cache retention for long tail content for any origin type. If you choose a multi-CDN strategy, you can improve the cache-hit ratio by serving similar requests from the same cache, either by reducing the number of CDNs for long tail content, or by implementing content-based sharding across CDNs, where each CDN serves a different subset of the VOD library.

Playback errors

During playback, the player can encounter various issues that are critical enough to interrupt the process and have the player throw an error. The reasons for playback errors can differ. They can be caused by poor network conditions, temporary issues at the origin, or badly tuned player and CDN configuration. Whatever the reason, an error results in bad viewer experience. Viewers most likely have to reload the application upon receiving such an error. You can review your architecture and make configuration improvements to reduce playback errors by the following:

- Offer lower bitrate versions of the stream to the player so it can maintain the playback state on during temporary periods of poor network conditions.
- Reduce [Error Caching Minimum TTL](#). This value in CloudFront configuration dictates how long the error response generated by the origin is returned to the viewers before the cache host make another attempt to retrieve the same object. By default, this value is **10** seconds. Because some players tend to request video segments before they become available, the response can be a 403 or 404. This response would be served to other viewers as well for 10 seconds, if you rely on default settings.

By knowing possible error codes and the origin returns, you can reduce this time setting appropriately.

Custom Error Response Settings




HTTP Error Code	404: Not Found	
Error Caching Minimum TTL (seconds)	1	
Customize Error Response	<input type="radio"/> Yes <input checked="" type="radio"/> No	

Figure 7 – Example Custom Response Settings

Include the certificate of root certificate authority (CA) in the certificate chain associated with the domain name used for media delivery. For HTTPS-based delivery, some viewer device platforms can fail, setting up TLS connection when the root CA is missing. Note that you need to supply a certificate chain when importing a certificate into AWS Certificate Manager (ACM). Public certificates generated through ACM have root CA certificate included in the certificate chain.

Cost optimization

CloudFront is a global CDN. However, if your customers are within a limited geographic Region, you can leverage [CloudFront price classes](#). By default, Amazon CloudFront minimizes end-user latency by delivering content from its entire global network of edge locations. Price classes let you reduce your delivery prices by excluding CloudFront's more expensive edge locations from your CloudFront distribution.

For example, if your customer only expects an audience in Europe or North America, you can change the price class to eliminate the cache misses from viewers in the rest of the world, and route requests to edge locations in Europe or North America. This does limit where requests are accepted from. For more information, see [Geographical restriction](#) and [Amazon CloudFront Pricing](#).

CloudFront Origin Shield

[CloudFront Origin Shield](#) is an optional feature that lets you select an additional layer of caching to reduce load on your origins and improve cache hit ratios. CloudFront RECs independently retrieve objects from the origin, which might result in duplicate requests for each object. By enabling Origin Shield in the AWS Region closest to the origin, CloudFront will then proxy traffic from the other RECs through the Origin Shield cache. This request collapsing and extra layer of cache reduces the load on the origin.

Origin Shield can be enabled for individual origins within CloudFront distributions, and also supports origins outside AWS. Cache hits from Origin Shield are identified in the CloudFront access logs.

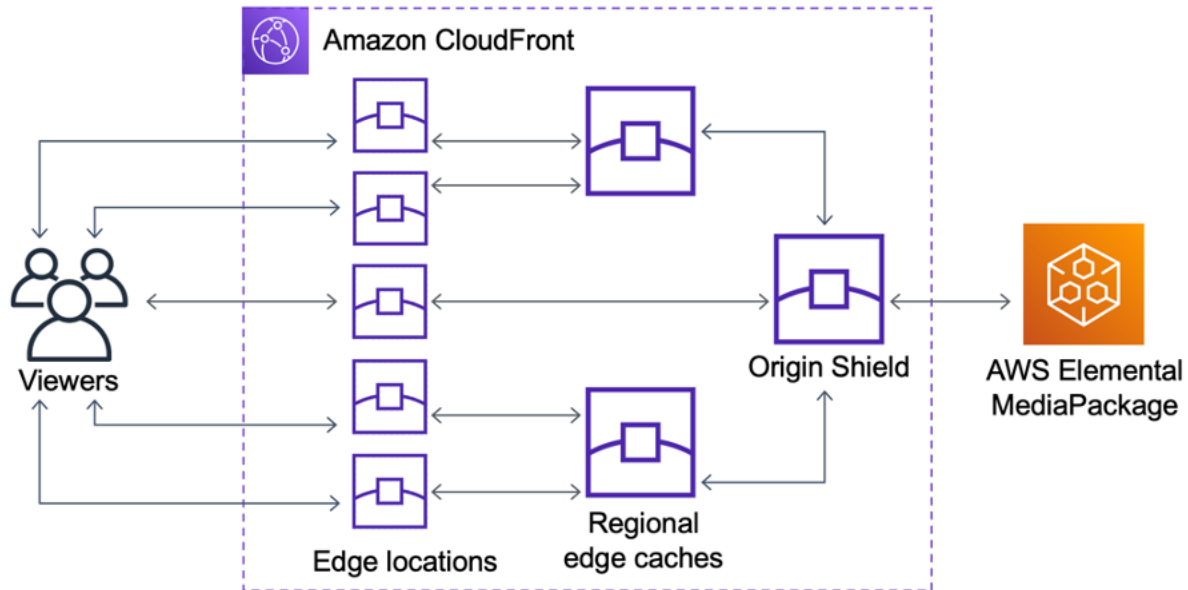


Figure 8 – Amazon CloudFront with Origin Shield

Security

Video content is valuable and your business model might depend on publishing content only to authorized users. Many video streaming platforms use a subscription-based business model, where content protection mechanisms are vital. This is an important factor when selecting a CDN.

Content protection can guard against piracy activities, such as link sharing, and might also be required under the terms of content rights or licensing agreements. Licensing terms might require you to restrict delivery to specific countries, or to integrate specific security controls in the media delivery workflow such as digital rights management (DRM) solutions

Geographical restriction

Amazon CloudFront allows you to configure geographical restrictions to prevent users in certain countries from accessing your content. You can enforce this by either specifying a list of countries where requests will be allowed or by specifying countries where

requests will be blocked. If a request is received from a blocked geography CloudFront will return 403 Forbidden HTTP status code to the end user. In testing, CloudFront's accuracy of determining a user's location is 99.8%.

Restriction Type Whitelist Blacklist

Countries

AF -- AFGHANISTAN
 AX -- ALAND ISLANDS
 AL -- ALBANIA
 DZ -- ALGERIA
 AS -- AMERICAN SAMOA
 AD -- ANDORRA

Add >>

<< Remove

DE -- GERMANY
 GB -- UNITED KINGDOM

Figure 9 – Geographic restriction in CloudFront console

CloudFront's Geo Restriction applies to an entire distribution. If more granular geo blocking rules are needed, you can shard your content into multiple CloudFront distributions and group content together with the same geographical restrictions. Alternatively, you can use AWS Web Application Firewall (AWS WAF) with [Geographic match rules](#). These rules work in a similar way to CloudFront's geo-restrictions, but they can be combined with other rules and matching statements to limit unwanted traffic, for example to block traffic incoming from VPNs, proxies or Tor nodes using the AWS Manager Rule *Anonymous IP list*.

Access control through CloudFront

There are two common approaches to access control for media content:

- In an **encryption-based** approach, you encrypt your video segments and distribute decryption keys to authorized users using a digital rights management (DRM) solution. DRM systems require integration with the origin for exchanging encryption keys and authorizing users to retrieve the decryption key. Commercial DRM systems providers offer a range of solutions with particular features and support for different devices. A multi-DRM solution is often necessary for operating at scale with a diverse population of devices.

- In an **access-control** based approach, you use tokenization to serve the content to authorized users only. For delivery at scale, the access control mechanism must be incorporated into CDN processing logic, because a viewer's request will be received by a cache server that validates the request and either allows or denies access.

Use cases that require an access control approach can leverage CloudFront's signed URLs or signed cookies features. Both features work by generating a policy that defines access conditions for the requested resource. This policy and resource URL are cryptographically-signed with a private key provided in a CloudFront key pair. When the viewer requests a restricted resource, CloudFront validates the policy and signature. If the validation process fails, a 403 Forbidden HTTP status code is generated in response. This type of access control in which requests are validated through cryptographically signed information is commonly referred as *tokenization*.

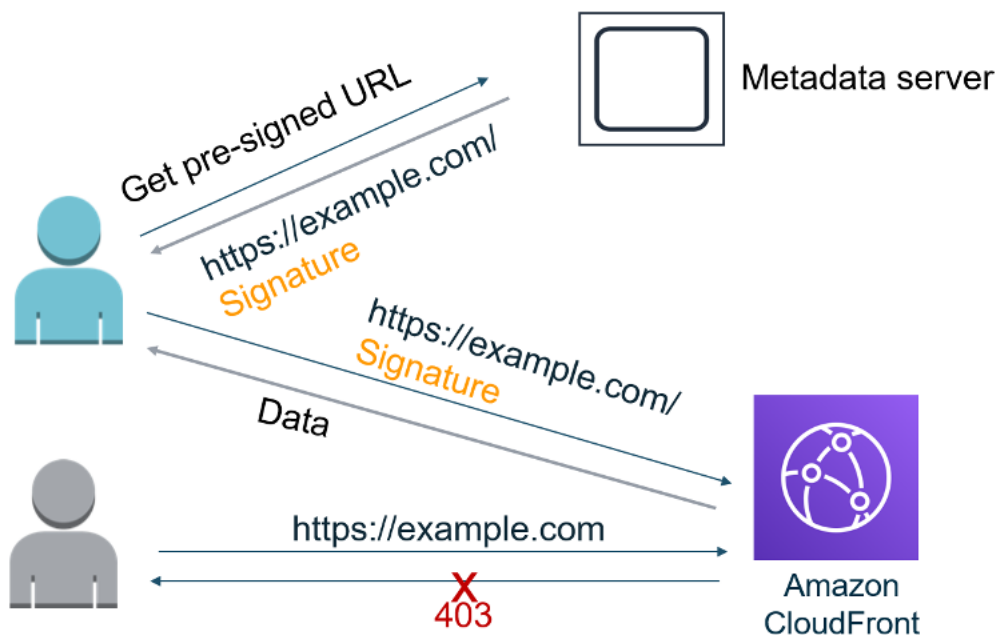


Figure 10 – Tokenized access control with a signed URL

In order to retrieve content from CloudFront, the player has to provide three pieces of information that will be used by CloudFront to validate the request:

- Expiry date or policy
- CloudFront key pair ID

- Signature

Before making a request, the viewer authenticates and obtains these three items from your website. The viewer has two options for including this information in the HTTP request: either by HTTP cookies (signed cookies) or query string parameters (signed URLs).

CloudFront's access control mechanism gives you flexibility on how you can leverage it:

You can enforce access control with signed URLs or cookies on all the objects required for media playback: manifest and video segments, or just one of them. Using signed cookies is a straightforward way to apply access control for all the relevant resources. Signed URLs require that you must append query string parameters to the objects referenced in the manifest files. This imposes additional requirement of rewriting manifest files either at the origin or by inserting this logic in the CDN's workflow. For that purpose, you can use [Lambda@Edge](#).

It is important to verify how the different platforms' devices handle cookies, query string parameters, and any limits. Some players might not support cookies at all, while others only support URLs up to a certain length.

You need to consider the duration the content will be available for to ensure a signed URL or signed cookie does not expire before the content is requested. This is achieved by managing the *Expires* timestamp appropriately

For a playback session of known duration, you can set the *expires* value to match the streaming event's expected end time. In the scenario where playback session length is not known in advance, more consideration is needed to manage the expiry attribute:

- For signed cookies, inspect the cookies and if expiry is getting close you can generate a new cookie with a new expiration time. For more information about using signed cookies, see the blog series, [Protecting your video stream with Amazon CloudFront and serverless technologies](#).
- For signed URLs, note that a common player behavior is to retrieve the manifest URL only once. Therefore, to ensure an uninterrupted playback for the authorized viewer, the expiry timestamp related to the manifest URL should be prolonged for an extended time period.

Session uniqueness

Tokenization as a method of access control works on the basis of viewers presenting cryptographically-signed information as proof that they have been authorized to watch the content. This creates a risk of link-sharing, where an authorized user who receives the signature in a legitimate way, can share it with unauthorized viewers.

To address that risk, you can refine the process of generating and validating the signature to include some attributes specific to a single, authorized user. A signature generated on a set of attributes unique to a single user provides a session uniqueness such that any other user attempting to use the same signature would fail at the validation stage. An unauthorized user might still attempt to circumvent this by falsifying the attributes in order to match them with a legitimate user's details. But if the signing attributes are difficult to retrieve and modify, then the risk of link sharing is reduced.

Signed URLs and signed cookies allow you to incorporate source IP in the [custom policy statement](#) used to generate the signature. If the requests including such a signature originates from source IP that doesn't fall within the specified IP address range in the policy, CloudFront denies access to the requested object. This largely reduces the risk of link-sharing in TCP-based connections, overriding source IP address to match the one specified in the token and receiving the response can't be done easily.

Multi-CDN considerations

As you develop and scale your architecture for media streaming, a multi-CDN approach might seem appealing. The driver for this is often a desire for more aggregate capacity, wider coverage in different geographies, or improving resilience and performance. Before making a decision, it is important to consider some of the disadvantages of a multi-CDN approach:

- **Increased load on the origin** – Because each CDN platform has to populate their caches independently, each one would make the request to your origin for original object. This effectively multiplies incoming traffic by the factor of number of CDN platforms. To alleviate this, you might consider using AWS Origin Shield, which creates common caching layer for all the CDNs, located in front of your origin.

- **Increased operational effort and lack of feature parity** – Different CDN vendors can offer different feature sets, forcing you towards a lowest common denominator approach. In addition, functionality (like tokenization), might be realized in a different manner, so you would still have to align your configuration and application logic, depending which CDN you send your viewers to.
- **Additional components in the architecture** – Using multiple CDNs usually increases the number of components in the overall architecture, like additional caching layers, switching service, performance data collection and scoring. This creates more complex architecture and complicates troubleshooting.

If you do decide that a multi-CDN architecture is right for you, a metric-based traffic allocation process will help you make data-driven decisions in the proof-of-concept and operational phases.

In order to make a data-driven decision, you will need to define and collect a set of metrics. Preferably, you would synthesize data points coming from your application, producing first-hand comparison of the performance and availability marks between used CDNs. Alternatively, you can rely on the metrics sourced by third-party user experience and monitoring platforms.

Once the metrics for each CDN platform have been collected, the next step is to derive a score figure that will determine the split of traffic across the CDNs. The overall score is built from aggregated metrics and you should be able to apply different weighting for each metric in calculating overall score, depending on which aspects of the playback you consider to be the most important one for your use case.

Traffic allocation in multi-CDN environments

With all the metrics and scoring numbers at hand, you will be able to decide how to allocate the traffic between your chosen CDN platforms. Depending on the level of granularity of collected metrics, you can make the allocation decision on different levels in terms of users' presence (country, metro, and ISP level), as well as device type (desktop computer, mobile, tablet, and smart TV). You should continue to monitor against metrics in operation and define thresholds for changes in allocation, maximum value of traffic share per each CDN, and conditions for full failover from one CDN when performance metrics drop below predefined threshold.

In operation, your viewers' requests must be routed between the CDNs in accordance to allocation decision. When each CDN vendor provides you with a dedicated DNS hostname, you can control traffic allocation between them by returning the right CNAME

in response to DNS queries for your original domain name. Route 53 with routing policies helps accomplish that. However, the precision of determining viewer's location and source network might be limited in this approach. Rather than relying on DNS another possibility is to set up an API endpoint, which will return URL pointing to specific CDN when viewer requests the playback URL.

In general, the greater the precision, control, and automation you need in managing traffic across multiple CDNs, the more effort is required to develop each of the elements. For a simple solution, such as allocating traffic on a country level, you can think of a simple load-balancing solution, where it is acceptable to have limited accuracy in terms of split share, and allocation changes can be done manually. Such a solution could take advantage of community-based measurements of generic test objects for CDN's performance overview, used for deciding on the split per each country in which you operate.

DNS-based traffic splitting can be achieved using Route 53 traffic policies to combine geolocation and weighted rules. For a more automated approach with more fine-grained control for allocating traffic, you will either need to develop a custom solution that allows you to align exactly to your use case or consider a more advanced third-party solution from a specialized vendor. For more information about multiple CDNs, see [Using multiple content delivery networks for video streaming](#).

Operational monitoring and reporting

Operational monitoring is vital for media workloads to maintain consistent high-quality experiences for end customers. Amazon CloudFront automatically publishes operational metrics, at one-minute granularity, into Amazon CloudWatch. A typical case is for monitoring 4xx and 5xx error rates, and configuring an alarm that triggers a notification to the operations team when the error rate reaches a threshold value. For more information about troubleshooting and setting up alarms, see [Four Steps for Debugging your Content Delivery on AWS](#).

In addition to default metrics provided at no additional costs, you can enable additional metrics, giving more visibility into performance of CloudFront traffic. The additional metrics include information on *cache hit rate* and *origin latency*, defined as the total time spent from when CloudFront receives a request to when it starts providing a response to the network, for requests that are served from the origin upon cache miss. The following image shows CloudWatch graphs for both metrics.

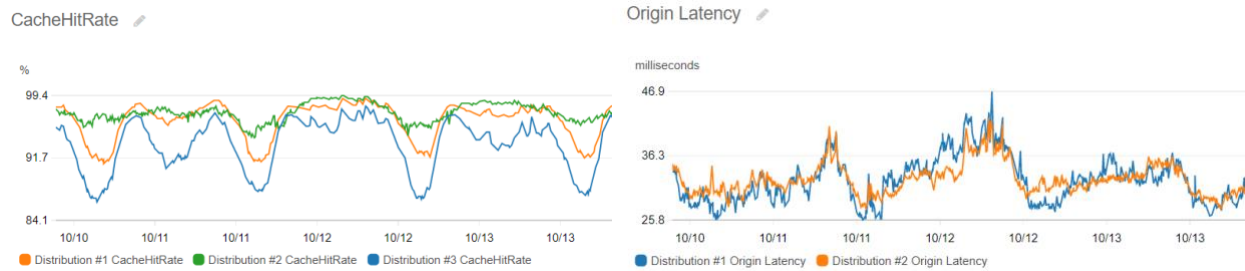


Figure 11 – Examples of CloudWatch operational metrics

Because CloudWatch metrics are emitted in the matter of minutes, you can constantly monitor relevant metrics and respond quickly to observed anomaly, which is an indication of performance drop, transitional issues, and misconfiguration.

For more detailed inspection of traffic to the level of individual request, CloudFront provides access logs that are consolidated from all of the edge locations and delivered within minutes to an Amazon S3 bucket. Each request to CloudFront has an associated unique request ID, included in *X-Amz-Cf-Id* header that you can find in the logs, as well as in the response headers and request headers which reach your origin.

If you need to open a ticket to CloudFront support, engineers can retrace exactly what happened for a specific request using the Request ID. This eliminates the need to wait for the issue to occur again to capture it and analyze it. You can inspect the logs in fine-grained detail with [Amazon Athena](#), which is a serverless query service that allows you to make ad hoc SQL queries. CloudFront supports real-time access logs and standard access logs. Standard access logs apply for all the traffic associated with distributions and are published in a destined Amazon S3 bucket in minutes. Real-time access logs log entries are fed into Kinesis Data Stream in the matter of seconds and can be consumed immediately. CloudFront real-time logs are configurable, allowing you to choose:

- The *sampling rate* for your real-time logs – that is, the percentage of requests for which you want to receive real-time log records.
- The specific fields that you want to receive in the log records
- The specific cache behaviors (path patterns) that you want to receive real-time logs for

You can use CloudFront access logs to monitor, analyze, and take action based on content delivery performance. For information about a dashboard for real-time logs, see [Creating realtime dashboards using Amazon CloudFront logs](#).

You can also build your own dashboards based on the logs using AWS services like Elasticsearch or QuickSight. APN Partner solutions like [Sumologic](#) and [Datadog](#) also provide dashboards for CloudFront. Some common reports are also summarized in the CloudFront **Reports & analytics** dashboard in the AWS Management Console:

- Statistics and usage reports, including number of requests, bytes transferred, cache hit ratio, status codes, protocols
- Top referrers and URLs
- Viewers reports about devices, browsers, OS, and locations

To get a comprehensive understanding of viewer's quality of experience, you can add external performance measurement solution in your architecture. The measurements would be taken either directly from your application (real user monitoring) or from a geographically distributed set of probes simulating typical viewers' requests (synthetic monitoring). For more information about benchmarking CDNs with this type of solution, see [Measuring CloudFront Performance](#).

Purpose-built monitoring solutions for video streaming monitoring typically work as a plugin in the video player, which exposes playback-related metrics like playback failures, rebuffering rate, and startup times.

Best practices for events and service launches

Large-scale live events provide compelling viewing for your customers but can pose a delivery challenge for media organizations. With live events, highly anticipated on-demand releases, and new video service launches, there is no second chance to get it right. Planning ahead is key to ensuring a high-quality viewing experience for your customers.

You can engage while in the planning phase for a major service launch. Some global customers have started months in advance while their service was still in development. You might already be familiar with [AWS Infrastructure Event Management \(IEM\)](#). This is a structured program that assists you in strategic planning for large-scale events, as well as real-time support during the most critical moments. AWS also offers [AWS Elemental Media Event Management \(MEM\)](#), a guided support program tailored specifically for the unique requirements of video events, such as marquee sporting events or video service launches. MEM is available for events with Amazon CloudFront and AWS Media Services.

Each MEM engagement follows four phases:



- Planning
- Preparation and testing
- Event support
- Retrospective

Planning starts well ahead of your launch date or event. The first step is to assign a team of specialists who will assist you through the four phases. The team will review your proposed event workflow, the expected audience, your requirements and success criteria, then document the end to end media delivery chain. The planning phase includes Risk Assessment and Mitigation Planning (RAMP) and an Operational Readiness Review (ORR) to ensure your workflow is well architected.

The [AWS Well-Architected Framework](#) has been developed to help cloud architects build secure, high-performing, resilient, and efficient infrastructure for their applications. This framework provides a consistent approach to evaluate architectures. It has been used in tens of thousands of workload reviews conducted by the AWS solutions architecture team, customers, and partners.

The second phase focuses on monitoring and actioning the risk mitigation steps. The team will make operational runbook recommendations and run at least one Game Day to simulate your unique event.

During the event week, the MEM team will work closely with your team to support the live event and proactively respond as the event progresses. The MEM team might be collocated with your own teams during this phase, and may have other AWS experts on call for the duration of your event.

The MEM service concludes with a post-event analysis where feedback from the launch is used to update operational playbooks, update configuration, and to inform future event planning.

Conclusion

This whitepaper has detailed best practices and key system considerations for media delivery with Amazon CloudFront. We described how CloudFront is part of the end-to-end media architecture. We learned about important features and optimizations for performance and security. We learned about how to monitor operational performance and how to plan for major events and service launches.

For more resources on media delivery with Amazon CloudFront, see [Amazon CloudFront for Media & Entertainment](#). For more information about running media workloads on AWS, see [Media & Entertainment on AWS](#).

Contributors

Contributors to this document include:

- Lee Atkinson, Principal Solutions Architect, Amazon Web Services
- Kamil Bogacz, Senior Specialist Solutions Architect, Amazon Web Services
- Ian Coleshill, Senior Solutions Architect, Amazon Web Services
- Andrew Morrow, Senior Solutions Architect, Amazon Web Services

Document revisions

Date	Description
November 2020	First publication

Notes

¹ HLS Specification <https://developer.apple.com/streaming/>

² DASH Specification <https://www.iso.org/standard/75485.html>

³ CMAF Specification <https://www.iso.org/standard/79106.html>

⁴ CMAF with HLS
https://developer.apple.com/documentation/http_live_streaming/about_the_common_media_application_format_with_http_live_streaming

⁵ Microsoft Smooth Streaming [MS-SSTR] https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-sstr/8383f27f-7efe-4c60-832a-387274457251