AWS Whitepaper

Amazon CloudFront for Media

Amazon CloudFront for Media: AWS Whitepaper

Table of Contents

Amazon CloudFront for Media

Publication date: **September 12, 2023** (*Document history*)

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 (VOD) 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.

This paper is intended for media architects, streaming architects, network architects, and CDN operations teams.

Introduction

Delivering media content over the internet at scale poses a unique set of challenges for architects and operators. Viewers expect the highest quality of experience with wide selection of content at high resolution, fast start-up time, and reliable delivery throughout the playback session. Live streaming can amplify the challenges of maintaining consistent low latency and reliable delivery, with peaks in demand for popular content and live events. 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 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 works with other AWS services and it uses a global network of Points of Presence that are all connected to AWS Regions through the AWS network backbone.

Media workloads in this context include live streaming (live events, 24/7 linear and assembled channels) and video on demand (VOD) content delivered over the internet. In both scenarios, media platforms use streaming techniques to allow end customers to view the media without having to download the content in its entirety. Delivery for closed networks has slightly different considerations. 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 whitepaper, we start by explaining the architecture of media workloads. Then, we explain the streaming technologies. Finally, we describe a CDN configuration in detail.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

In the Streaming Media Lens, we focus on the best practices for architecting and improving your streaming media workloads on AWS.

Streaming techniques for media

Media streaming involves one or more techniques to deliver video over the internet. 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 as 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 to this issue 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. Also, 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.

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 possible video quality to be delivered to the viewer under varying conditions.

The ABR experience is superior to delivering a video at a single bitrate because the video stream can be continually switched dependent on available network bandwidth, allowing playback at different bitrates within an ABR ladder. This ability to adapt helps avoid buffering or interruption in playback that can happen when a client's network throughput is insufficient for a particular bitrate.

Most of the streaming techniques deliver a representation of a video stream in a manifest file and the media itself is streamed in a series of segments. This allows a player to begin video playback as soon as possible after receiving data. Streaming techniques for the internet differ from those for closed networks because there is no guaranteed bitrate available at the client, and they must be able to cope with changes in the available bandwidth between the CDN and player.

Most of the streaming techniques on the internet use HTTP to distribute content. However, there are many differences in how and when these technologies might be used based on viewer devices and network characteristics. 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 formats and bitrates for playout, or packaged at the time of requesting, using just-in-time (JIT) packaging, for different client devices.
- **Live streaming –** This workflow describes source content for linear TV or assembled 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 (VOD) 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 client does not have to download the whole file before playback begins.

Common standards of streaming technologies

- HTTP is used for video and signaling to provide compatibility with networks, firewalls, caches and end clients.
- Video is delivered as a set of concurrent streams with different renditions.
- Each video stream is divided into a series of segment files, each containing a few seconds of video.
- The list of available streams is obtained by downloading an index or manifest file.
- The client selects a stream, downloads the first few segments dependent on the player's buffer size, and starts playing. 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), was developed by Apple and is natively supported on all iOS devices. Segments, using either MPEG-2 transport stream (`.ts` files) or fragmented MPEG-4 (fMP4 files with `.mp4` extension) encoding standards, are typically packaged at 2 to 10 seconds long. The list of streams is provided in a manifest file with an `.m3u8` extension. HLS supports H.264 (AVC) and H.265 (HEVC) video codecs.

The Low-Latency extension to HLS (LL-HLS) specification allows delivering media segments in parts, referred to as *partial segments*. A partial segment can be as short as 200 milliseconds and can be published much earlier than the complete segment.

Dynamic Adaptive Streaming over HTTP

Dynamic Adaptive Streaming over HTTP (DASH or MPEG-DASH) was collaboratively developed 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.

The low latency variant of MPEG-DASH, referred to as Low Latency DASH (LL-DASH), makes use of HTTP Chunked Transfer Encoding to enable the player to start receiving the segment in chunks even before the full segment is completed.

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 23000-19 and its application for HLS 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 Microsoft Smooth Streaming (MS-SSTR) specification. 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 following diagram shows the key media infrastructure components described in more detail in the following sections.

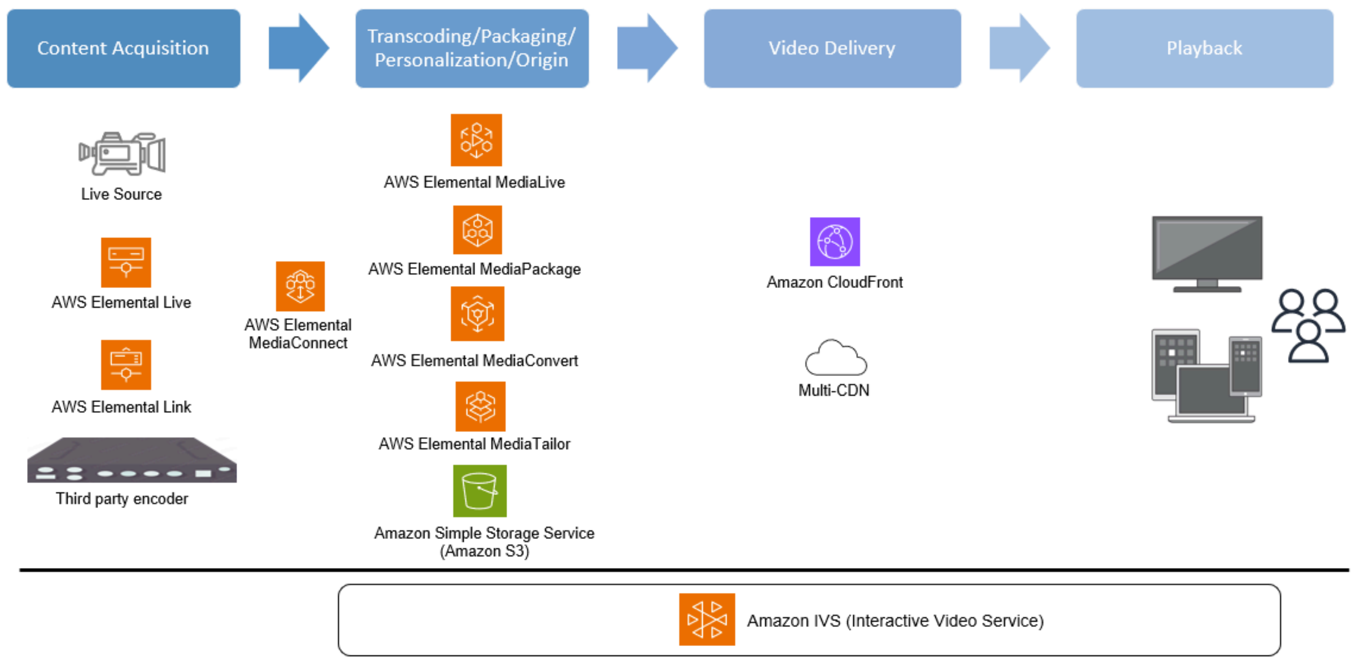

Content Acquisition
Transcoding/Packaging/ Personalization/Origin
Video Delivery
Playback
Live Source
AWS Elemental Live
AWS Elemental Link
Third party encoder
AWS Elemental MediaConnect
AWS Elemental MediaLive
AWS Elemental MediaPackage
AWS Elemental MediaConvert
AWS Elemental MediaTailor
Amazon Simple Storage Service (Amazon S3)
Amazon CloudFront
Multi-CDN
Amazon IVS (Interactive Video Service)

Figure 1 – Typical components of a video workflow

Encoding and packaging

The video content has to be encoded several times in parallel to provide a set of streams at different bitrates using services such as AWS Elemental MediaLive and AWS Elemental MediaConvert. 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.

After the video has been encoded into multiple resolutions, it's packaged into one or more streaming containers. At this point, 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 such as AWS Elemental MediaPackage, 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).

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 solution architect will work backwards from the audience of client devices and select a set of streaming technologies, video resolutions, and bitrates to meet their needs.

Server-side Ad Insertion (SSAI)

Because of the 1:1 relationship between a video stream and a viewer, many platforms insert personalized ads into the video stream, providing relevant content for viewers and targeted advertising opportunities for advertisers.

Server-Side Ad Insertion (SSAI), or Dynamic Ad Insertion (DAI) enable advertisements to be seamlessly streamed on internet-enabled consumer devices including set-top boxes, mobiles, tablets, games consoles and smart televisions. Consumers stream video content on their internet-enabled devices over a content delivery network (CDN). When advertisement breaks are signaled for insertion into the video stream, the SSAI server will request ads from a third-party ad server.

Ad servers typically implement VAST (Video Ad Serving Template) for structuring ad tags that serve ads to video players. Using XML, VAST transfers metadata about the ad from the ad server to the video player. The SSAI server then inserts those ads at the same bitrate, frame rate, and audio level as the original stream to prevent any disruption, with the ad seamlessly inserted into the stream, to provide a continuous viewing experience. AWS Elemental MediaTailor is a personalized ad-insertion service for video providers that lets you monetize channels with personalized advertising.

Channel assembly

Many organizations have successfully used VOD content assets to create new linear channels. Channel Assembly in AWS Elemental MediaTailor enables content providers to quickly create scheduled playout channels by putting together programming from multiple sources and adding them together into a linear playlist. Channel Assembly can schedule previously transcoded and packaged content from existing video on demand (VOD) catalogs, and live content into a linear channel.

Channel Assembly supports live source content that enables content providers to extend their offerings, and monetize those streams, even for channels with low viewing figures that would otherwise be cost-prohibitive to run. Live and VOD content virtual linear streams are created with a low running cost by using existing multi-bitrate encoded and packaged content. Ad breaks can be inserted without having to process the content with SCTE-35 markers, allowing for an even more seamless monetization of linear streams.

The origin

The origin is the endpoint where viewer requests are processed 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 can build a custom implementation to meet your specific needs.

- Amazon S3 is a great place to start as an origin for VOD or Live content. It requires that the content is packaged and stored 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.
- Independent software vendors (ISVs) in the AWS Partner Network (APN) also provide origin solutions, including alternative packagers.

The content delivery network (CDN)

Streaming requires individual video delivery to every viewer's device, placing a much higher load on the network than traditional one-to-many broadcast-based delivery. The load on the network by 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?

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 an 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) such as Amazon CloudFront.

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, Video On Demand on AWS, and Secure Media Delivery at the Edge 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.

Amazon CloudFront CDN 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.

Topics

Architecture

CloudFront uses a global network of Points of Presence (PoPs) and Regional Edge Caches (RECs), 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 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 existing locations. This can be particularly relevant when planning the launch of your video platform in a new geographic region or anticipating high peak events.

The CloudFront 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 before 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.

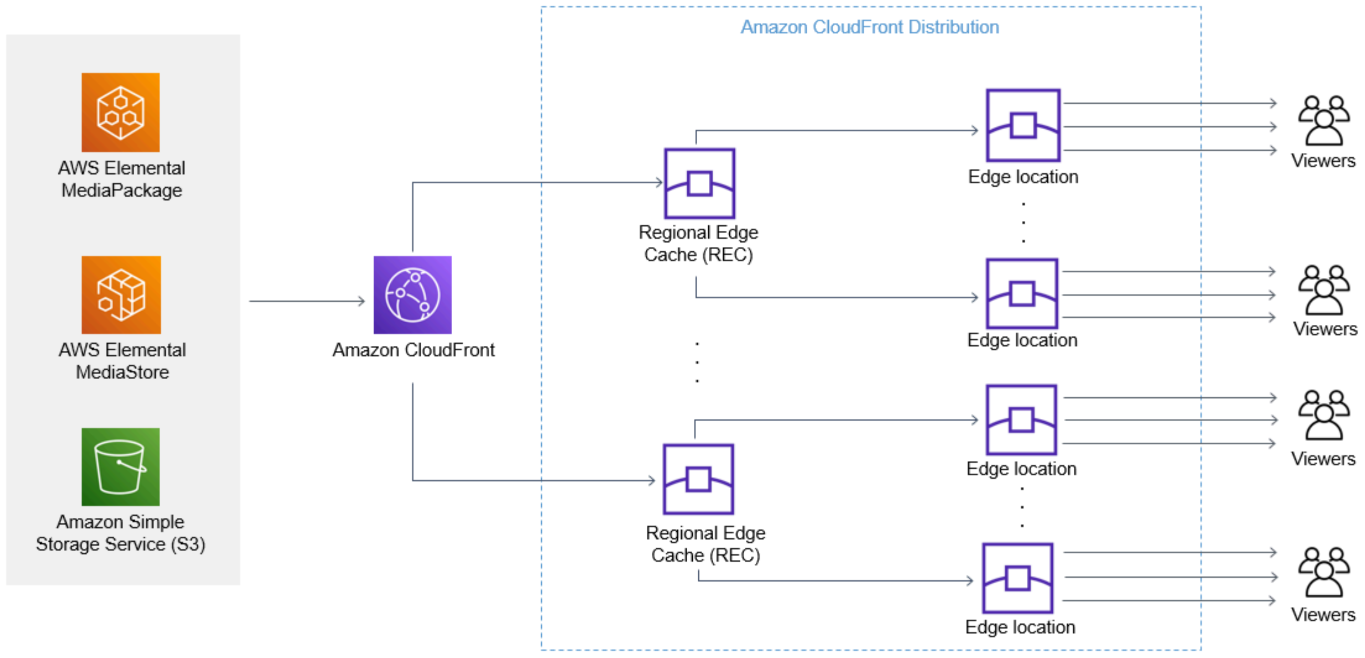

Amazon CloudFront Distribution
AWS Elemental MediaPackage
AWS Elemental MediaStore
Amazon Simple Storage Service (S3)
Amazon CloudFront
Regional Edge Cache (REC)
Regional Edge Cache (REC)
Edge location
Edge location
Edge location
Edge location
Viewers
Viewers
Viewers
Viewers

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.

You can write your own code to further customize the processing of HTTP requests and responses by associating an edge function with your CloudFront distribution. To minimize latency, edge functions will run close to your viewers. You can select from two different types of edge functions based on your use case:

- For complex, compute-heavy use cases, Lambda@Edge gives you the ability to run JavaScript or Python code in a regional edge cache, usually in the AWS Region closest to the CloudFront edge location that received the viewer request. Examples of using Lambda@Edge in media delivery include filtering live streaming renditions by device type at the Edge and on-the-fly video conversion from MP4 to HLS. For additional examples, see Lambda@Edge example functions in the *CloudFront Developer Guide*.
- For lightweight, latency-sensitive use cases, CloudFront Functions can be used to execute lightweight JavaScript code at CloudFront edge locations at approximately one-sixth of the price of Lambda@Edge. Some common use cases for CloudFront Functions are cache-key manipulations and normalization, URL rewrites and redirects, HTTP Header manipulation, and access authorization. For more examples, see Example code for CloudFront Functions.

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. Without Origin Shield, 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 proxy traffic from the other RECs through the Origin Shield cache. This extra layer of cache and request collapsing 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.

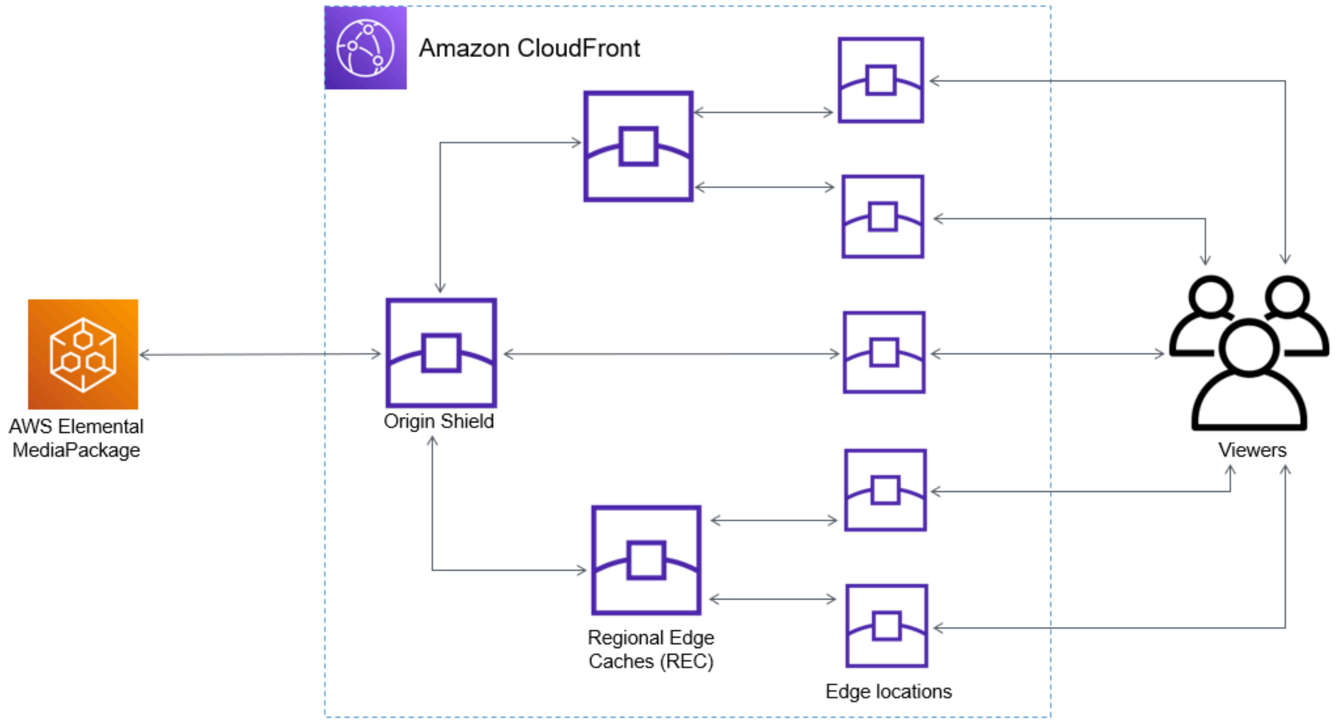


Figure 3 – Amazon CloudFront with Origin Shield

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 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.

HTTP/3

HTTP/3 is the latest improvement over previous HTTP versions and it can help customers improve performance and user experience by eliminating head-of-line blocking. HTTP/3 uses QUIC, a user datagram protocol-based, stream-multiplexed, and secure transport protocol that combines and improves upon the capabilities of existing TCP, TLS, and HTTP/2. HTTP/3 further helps improve user experience by supporting client-connection migration which allows smoother transitions between Wi-Fi and cellular access networks with minimal or no interruption. HTTP/3 also provides enhanced security as it uses QUIC which encrypts packets using TLS by default.

For media workloads, HTTP/3 can help securely deliver content with reduced video start-up times and create uninterrupted streaming experience when viewers roam between different access networks.

CloudFront supports HTTP/3 in all of its edge locations at no additional cost. To learn more about Amazon CloudFront HTTP/3, refer to the *CloudFront Developer Guide*

Byte streaming

A common mode of operation 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, called *byte streaming*, when forwarding the newly retrieved objects from the origin to minimize latency. This means that as soon as the first byte from the origin reaches CloudFront, it will be immediately forwarded to the viewer.

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 when it receives its first request, 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. Request collapsing is disabled if the object is not cacheable or if cookie-forwarding is enabled.

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 (ABR) 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

You can further improve performance and resilience of your media workloads by following the configuration best practices outlined below.

Cache Hit Ratio

The Cache Hit Ratio (CHR) is a key metric for CDNs. The CHR is defined as:

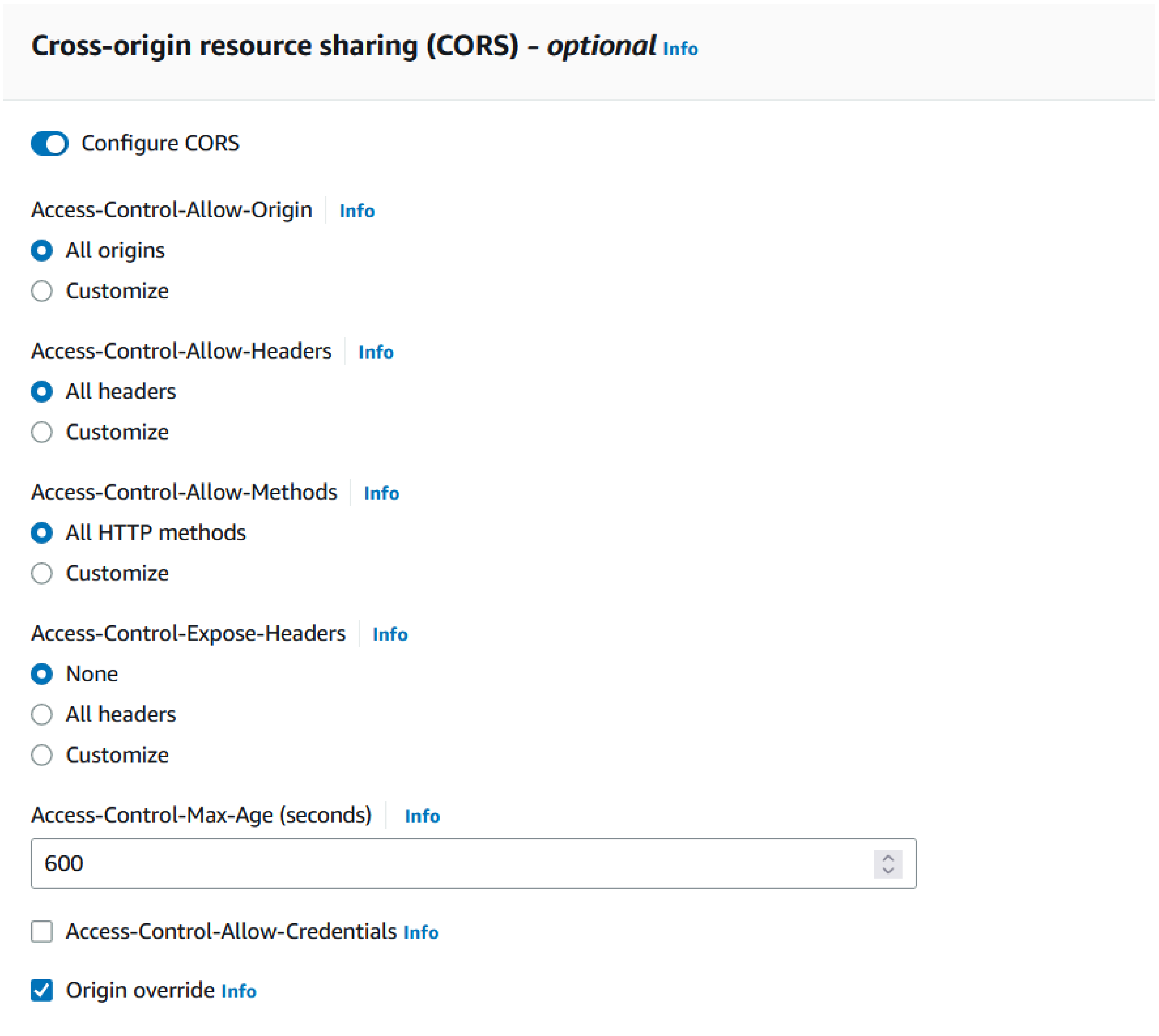
```
      requests served directly from the cache
CHR = ---------------------------------------
              number of total requests
```

The CHR metric in CloudFront considers only requests with response status `HIT` (see `x-edge-result-type` field for additional response status types), as requests served directly from the cache. The CHR is expressed as a percentage over a measurement interval. Maximizing the CHR has a direct performance improvement as it reduces 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 further maximize the CHR value using the following recommendations:

General settings

- **Optimize settings of the cache key by making use of the** CloudFront policies:
  - **Cache policies:** specify headers, query parameters or cookies that must be part of the cache key
  - **Origin request policies**: specify headers, query parameters or cookies that should be forwarded to the origin but do not need to be part of the cache key
  - **Response header policies**: specify which headers CloudFront should add or remove in the responses it sends to viewers, such as CORS headers
- **Use cache policies optimized for a specific origin.** Elemental MediaPackage cache policy is optimized for VoD workloads that use MediaPackage as a CloudFront origin
- **If managing cache-control directives is too complicated on the origin level, CloudFront provides the possibility to set a** Minimum, Default and Maximum TTL.
- **Avoid forwarding CORS headers to the origin**. Instead, choose from a number of managed response header policies that include preconfigured values for CORS and security header values. You can also optionally configure a custom response header policy.


Cross-origin resource sharing (CORS) - *optional* Info

Configure CORS [✓]

Access-Control-Allow-Origin | Info

[✓ All origins]
Customize

Access-Control-Allow-Headers | Info

[✓ All headers]
Customize

Access-Control-Allow-Methods | Info

[✓ All HTTP methods]
Customize

Access-Control-Expose-Headers | Info

[✓ None]
All headers
Customize

Access-Control-Max-Age (seconds) | Info

600

☐ Access-Control-Allow-Credentials Info

☑ Origin override Info

Figure 4 – Example of CloudFront CORS Response Header Policy

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

Settings for live streaming

- **Configure separate cache behaviors for video manifests and segments:**

  - **Manifest cache behavior (applicable to HLS media playlist manifest)**: configure a cache policy with Min TTL set to 1 second and Default TTL close to half the segment duration. If the workflow includes SSAI, manifest caching must be disabled by setting all TTL values to 0.
  - **Segment cache behavior**: configure a cache policy with Min TTL to 1 second and Default TTL to greater than twice the segment duration, typically based on the event duration
- **For LL-HLS, you can further optimize content delivery by following these additional recommendations:**
  - **Add LL-HLS query string parameters to the cache key for playlist manifest files**. LL-HLS introduces special query string parameters which are added to the URL of a playlist The client can request a playlist manifest with a specific partial segment included using a .../`playlist.m3u8?_HLS_msn=10&_HLS_part=2` URL, for example. This request instructs CloudFront and the origin not to respond until a playlist is available that contains part 2 of segment 10. Therefore, `_HLS_msn` and `_HLS_part` query parameters should be added to the cache key and the max-age value expected to be set close to the segment length.
  - **Prevent caching playlist manifest delta updates for LL-HLS.** The client can request to only send certain number of segments from the live edge by specifying the query parameter `_HLS_skip=YES`. To make sure these delta manifests are delivered directly from the origin, the query parameter `_HLS_skip` should be added to the cache key and `Cache-Control` directive should be set to `no-cache, no-store` or `private` in the response directly on the origin or by using Lambda@Edge with the origin response trigger.

Origin settings

Optimize origin settings to help further improve origin security, reduce latency and increase availability using the following guidelines.

General settings

- **Use Origin Access Control (OAC) to restrict access to an S3 origin**. OAC is a managed access control feature in Amazon CloudFront to help insure that your S3 bucket origin is not directly exposed to the internet.
- **Enable origin failover by creating an origin group**. Verify the response codes in failure scenarios when configuring failover criteria.

Settings for live streaming

- **Enable Origin Shield in the AWS Region closest to your origin location**. There are also other scenarios discussed in this whitepaper where Origin Shield can help reduce the load on your origin. However, Origin Shield is highly recommended for live streaming events due to a high number of concurrent requests
- **Set connection and response timeouts to be between half a segment and segment duration**. The same settings are recommended for an origin hosting ad content
- **Set `keep-alive` timeout to be longer than a segment duration**. The same settings are recommended for an origin hosting advertisements.
- **Set the number of connection attempts and connection timeout to be between half a segment and segment duration**. For example, if the segment duration is 4 seconds, set the connection attempts to 2 and the connection timeout to 2 seconds. The same settings are recommended for an origin hosting ad 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 the 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.

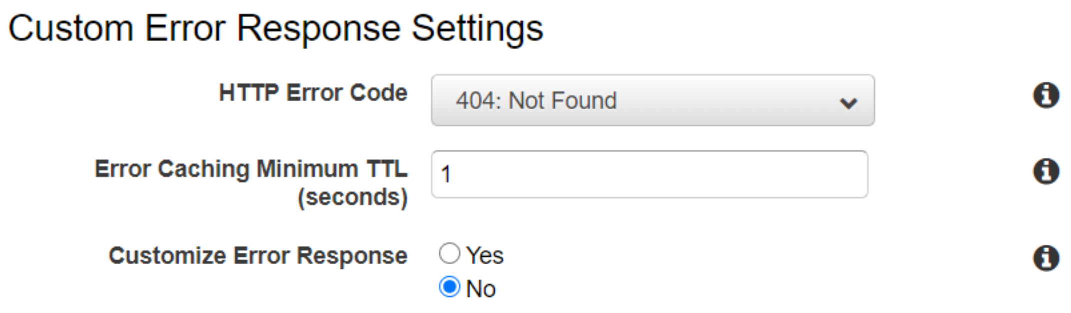
If you choose a multi-CDN strategy, you can improve the cache-hit ratio by serving similar requests from the same cache, either by reducing the number of CDNs for long tail content, or by implementing content-based sharding across CDNs, where each CDN serves a different subset of the VOD library.

Troubleshooting playback issues

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

- Offering lower bitrate versions of the stream to the player so it can maintain the playback state during temporary periods of poor network conditions.
- Reducing *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 that the origin could return, 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	Yes [✓ No]

Figure 5 – Example Custom Error 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 to set up a 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 but 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 the 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. While this limits where requests are served from, all viewers can still access CloudFront regardless of their location. For more information, see Geographical restriction and Amazon CloudFront Pricing.

Customers willing to make traffic commits of typically 10 TB per month or higher are eligible for discounted pricing. Contact your AWS account team for further information.

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.

Digital rights management

Digital rights management (DRM) is a technology that content providers can use to protect media assets by encrypting them with a key, making it unplayable unless the player has the key to decrypt it. To play protected content, a consumer must obtain authorization from the content provider. The content provider determines how and when the consumer can decrypt and watch content. Using DRM technology, content providers can encrypt media assets over the internet in a protected file format and control the use of their digital content.

Secure Packager and Encoder Key Exchange (SPEKE) defines the standard for communication between encryptors and packagers of media content and digital rights management (DRM) key providers.

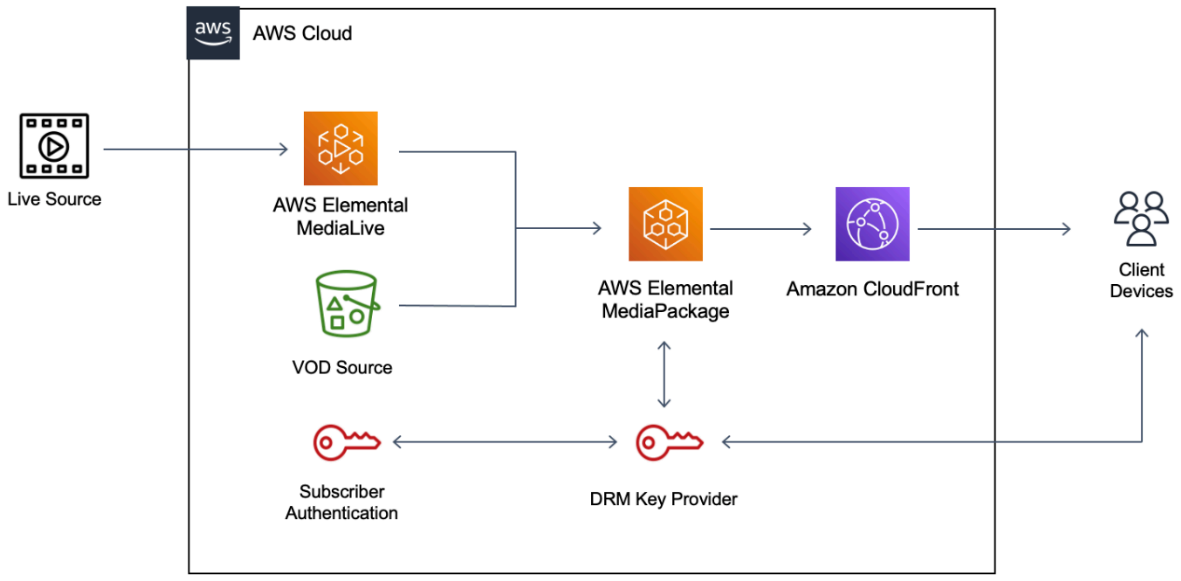
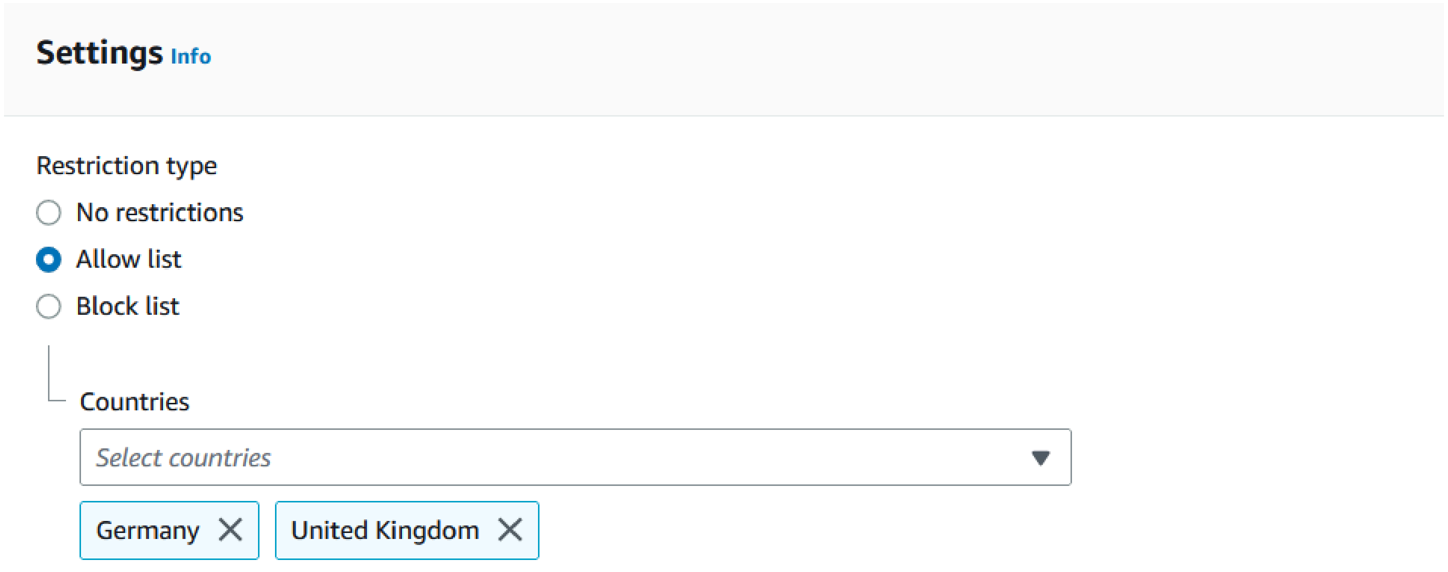


Figure 6 – Digital Rights Management sample workflow

Geographical restriction

Amazon CloudFront allows you to configure *geographical restrictions*, sometimes known as *geo blocking*, to prevent users in specific geographic locations from accessing your content. You can enforce this by either specifying a list of jurisdictions where requests will be allowed or by specifying locations where requests will be blocked. If a request is received from a blocked geography, CloudFront will return a `403 Forbidden` HTTP status code to the end user. In testing, the accuracy of CloudFront determining a user's location is 99.8%.



Settings Info

Restriction type

No restrictions [✓ Allow list] Block list

Countries

Select countries

Germany ✕ United Kingdom ✕

Figure 7 – Geographic restriction in CloudFront console

Geographic restrictions in CloudFront apply 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.
- Use AWS WAF with Geographic match rules. These rules work in a similar way to geo blocking in CloudFront, 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 Managed Rule *Anonymous IP list*.
- Use CloudFront Functions with geolocation headers. This approach allows you to implement geographical restrictions at the behavior level for specific URL path patterns.

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 DRM. 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 signed URLs or signed cookies in CloudFront. Using signed cookies is a straightforward way to apply access control to all the media objects required for media playback: Manifest and video segments.

Some player platforms, however, may not support cookies and using signed URLs may require appending query parameters to the objects referenced in the manifest file. For those cases, the Secure Media Delivery at the Edge solution greatly simplifies and automates the process of token management with widespread support across variety of clients. This solution also allows adjusting policy parameters (such as adding geographical restrictions, custom headers, and source IPs) if more fine-grained access control is required.

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 deciding, it is important to consider some of the disadvantages of a multi-CDN approach:

- **Increased load on the origin –** Because each CDN has to populate their caches independently, each one would make the request to your origin for original object. This multiplies incoming traffic by the number of CDNs. To alleviate this, you might consider using AWS Origin Shield, which creates a 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 (such as tokenization), might be implemented differently, 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 can help you make data-driven decisions in the proof-of-concept and operational phases.

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 metrics between the CDNs. Alternatively, you can rely on the metrics sourced by third-party user experience and monitoring platforms.

After 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 host name, you can control traffic allocation between them by returning the right `CNAME` in response to DNS queries for your original domain name. 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 location in which you operate.

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 cost, 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 a CloudWatch dashboard with all CloudFront operational metrics.

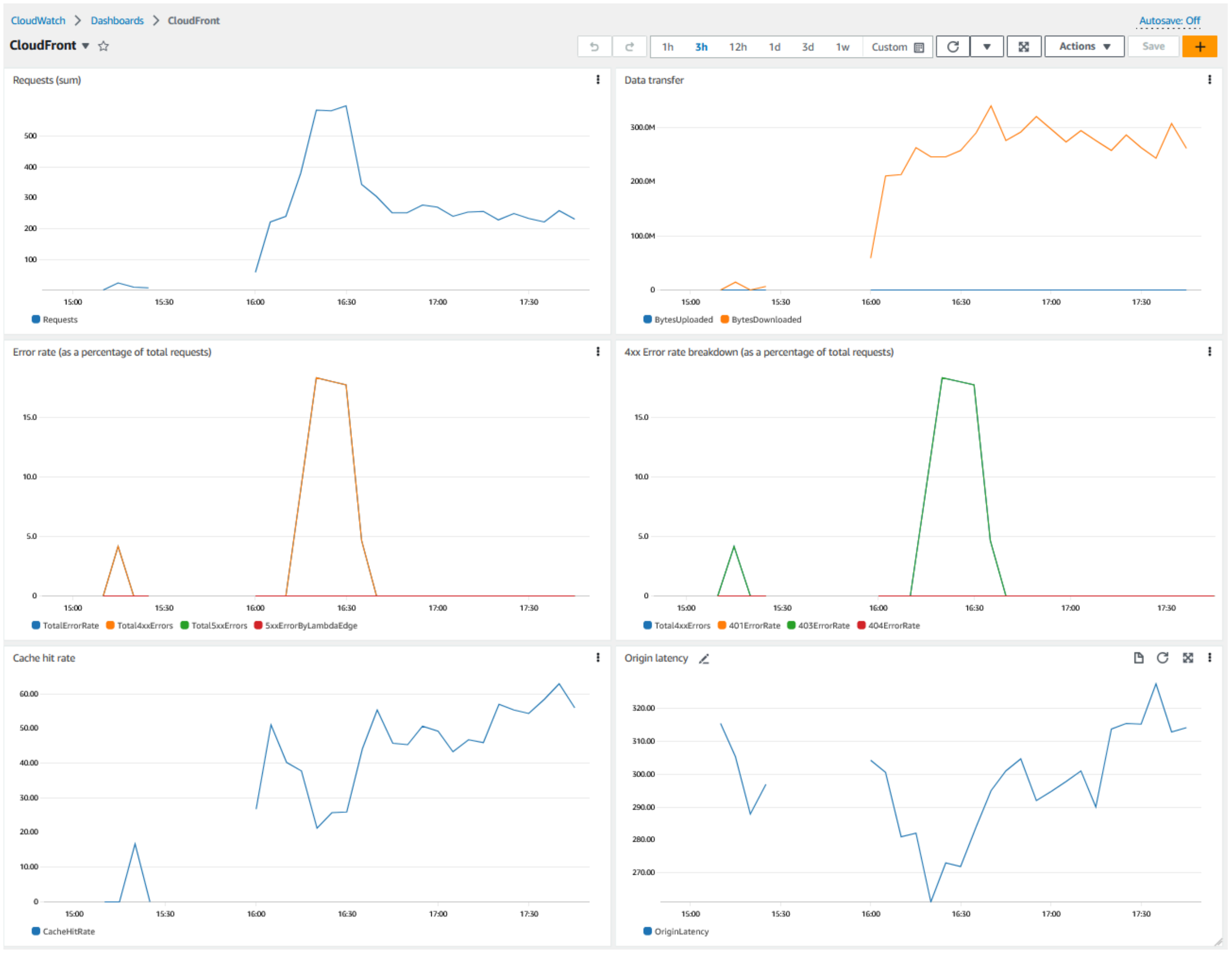

CloudWatch > Dashboards > CloudFront
CloudFront
Autosave: Off
1h 3h 12h 1d 3d 1w Custom
Actions
Save
Requests (sum)
Requests
Data transfer
BytesUploaded BytesDownloaded
Error rate (as a percentage of total requests)
TotalErrorRate Total4xxErrors Total5xxErrors 5xxErrorByLambdaEdge
4xx Error rate breakdown (as a percentage of total requests)
Total4xxErrors 401ErrorRate 403ErrorRate 404ErrorRate
Cache hit rate
CacheHitRate
Origin latency
OriginLatency

Figure 8 – Example of a CloudWatch dashboard with CloudFront operational metrics

Because CloudWatch metrics are emitted within minutes, you can monitor relevant metrics and respond quickly to an observed anomaly indicating performance drop, transitional issues, or misconfiguration.

When troubleshooting CloudFront performance you can gain more granular insights into first-mile connectivity by enabling the Server-Timing header using CloudFront Response Header Policy and last-mile connectivity by using CloudWatch Internet Monitor.

For more detailed inspection of traffic to the level of individual request, CloudFront provides access logs that are consolidated from all 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, the request headers which reach your origin, and response headers.

If you need to open a ticket to 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 and analyze it. You can inspect the logs in fine-grained detail with Amazon Athena, a serverless query service that allows you to make SQL queries on S3 content. CloudFront supports real-time access logs and standard access logs. Standard access logs apply for all the traffic associated with distributions and are delivered to an Amazon S3 bucket within minutes. Real-time access logs' entries are fed into a Kinesis Data Stream within 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 such as Amazon OpenSearch Service or QuickSight. Partner Solutions from the AWS Partner Network (APN) such as Sumologic and Datadog can also provide dashboards for CloudFront. Some common reports are 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 the viewers' quality of experience, you can add external performance measurement solution to 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 typically work as a video player plugin and provides playback-related metrics like playback failures, rebuffering rate, and startup times. With this reference solution, you can correlate client-side and server-side data by merging them in the same log record and displaying it in the same dashboard. The client-side data is collected from the media player using Common Media Client Data (CMCD) format developed by Web Application Video Ecosystem (WAVE) project. The image below shows a dashboard from the solution:

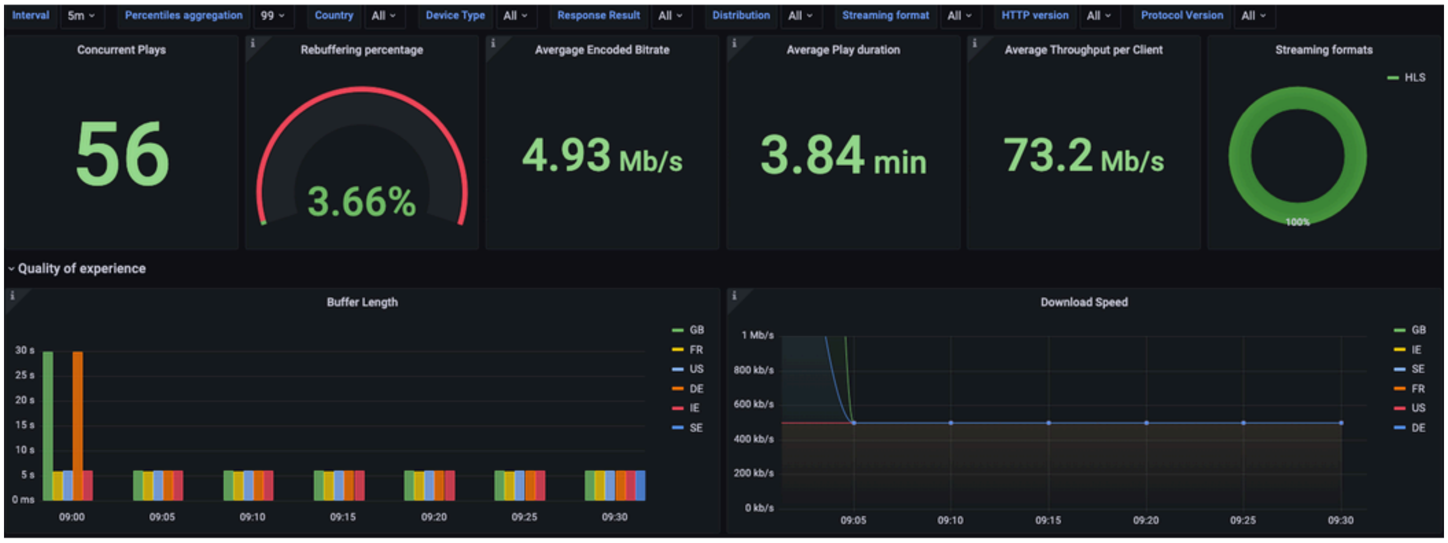


Figure 9 – QoE dashboard example

Best practices for events and service launches

Large-scale live events provide compelling viewing 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 is key to ensuring a high-quality viewing experience for your customers.

You can engage with your AWS Account Team while in the planning phase for a major service launch to help you review your architecture using the AWS Well-Architected Framework Review, including the Streaming Media Lens, and check service quotas based on your anticipated traffic profile. Some AWS customers start planning months in advance while their service is 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 has the following four phases: Planning, Preparation and Testing, Event Support, and Retrospective

Planning

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.

Preparation and testing

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.

Event support

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 co-located with your own teams during this phase and may have other AWS experts on-call for the duration of your event.

Retrospective

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, Senior Manager, Solutions Architecture, Amazon Web Services
- Kamil Bogacz, Senior Edge Specialist Solutions Architect, Amazon Web Services
- Ian Coleshill, Principal Solutions Architect, Amazon Web Services
- Igor Kushnirov, Senior Edge Specialist Solutions Architect, Amazon Web Services
- Andrew Morrow, Senior Solutions Architect, Amazon Web Services
- Tal Shalom, Principal Product Manager, Amazon Web Services
- Achraf Souk, Senior Manager, Edge Specialist Solutions Architecture, Amazon Web Services

Appendix: Amazon Interactive Video Service

Amazon Interactive Video Service (Amazon IVS) is intended for a different use case compared to the broadcast grade video workflows. Amazon IVS is a fully managed and easy-to-set-up AWS live streaming solution that is ideal for creating low latency interactive video experiences and includes features like timed chat and timed metadata API that enables you to add graphics, polls, and other synchronized components to applications.

Amazon IVS is built on the same live streaming technology that powers Twitch, and there is no need to manage infrastructure or configure components of your video workflow. CDN functionality is included as part of the managed service and there is no integration point with Amazon CloudFront. Amazon IVS supports RTMPS and RTMP streaming inputs rather than the more common Direct to Consumer streaming protocols. End to end latency is typically under five seconds, and can be lower. This latency is low compared to direct to consumer platforms, and higher than video conferencing applications.

AWS Elemental Media Services provide a much richer set of functionalities for building broadcast grade video workflows, for a broad set of client devices and players. AWS Elemental Media Services enable you to integrate with a wide range of partner solutions using open standards.

Amazon IVS is out of scope for this whitepaper. For additional information, see the Amazon IVS documentation.

Document history

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Whitepaper updated	Updated to include latest recommendations, features, and additional workflows, including: • SSAI, Channel Assembly, and DRM workflows • New section on Amazon IVS • New recommendations and reference architecture for secure media delivery • Updated CloudFront configuration best practices including recommendations for LL-HLS • Updated the section for best practices for event and service launches • Updated the section on operational monitoring and reporting	September 12, 2023
Initial publication	Whitepaper first published.	November 13, 2020

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser that you are using.

Notices

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.