AWS Well-Architected Framework

Streaming Media Lens

Streaming Media Lens: AWS Well-Architected Framework

Table of Contents

Streaming Media Lens

Publication date: **September 29, 2021** (*Document history and contributors*)

This whitepaper describes the AWS Streaming Media Lens for the AWS Well-Architected Framework, which helps customers apply best practices in the design, delivery, and maintenance of their cloud-based streaming media workloads. The document describes general design principles, as well as specific best practices and guidance for the six pillars of the Well-Architected Framework.

This paper is intended for those in technology roles, such as technology leaders, architects, developers, and operations team members. After reading this paper, you will understand AWS best practices and the strategies to use when designing and operating streaming media workloads in a cloud environment.

Introduction

The AWS Well-Architected Framework helps cloud architects build secure, high-performing, resilient, and efficient infrastructure for their applications and workloads. Based on six pillars — operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability — AWS Well-Architected provides a consistent approach for customers and AWS Partners to evaluate architectures, remediate risks, and implement designs that deliver business value.

In this Lens, we focus on how to design and deploy **streaming media** workloads by defining components, exploring common workload scenarios, and outlining design principles that help you to apply the AWS Well-Architected Framework. We then address specific best practices aligned with the pillars of the Well-Architected Framework.

Streaming media workloads transmit audio and video from content publishers to audiences. Streaming media is typically used for one-to-many broadcasts to audiences over HTTP. Readers interested in real-time communications for web conferencing applications should refer to the Real-Time Communication on AWS whitepaper.

For brevity, we only cover details from the Well-Architected Framework that are specific to streaming media workloads. We recommend that you start by considering best practices and questions from the AWS Well-Architected Framework whitepaper when designing your architecture.

Lens availability

The Streaming Media Lens is available as an AWS-official lens in the Lens Catalog of the AWS Well-Architected Tool.

To get started, follow the steps in Adding a lens to a workload and select the **Streaming Media Lens**.

Definitions

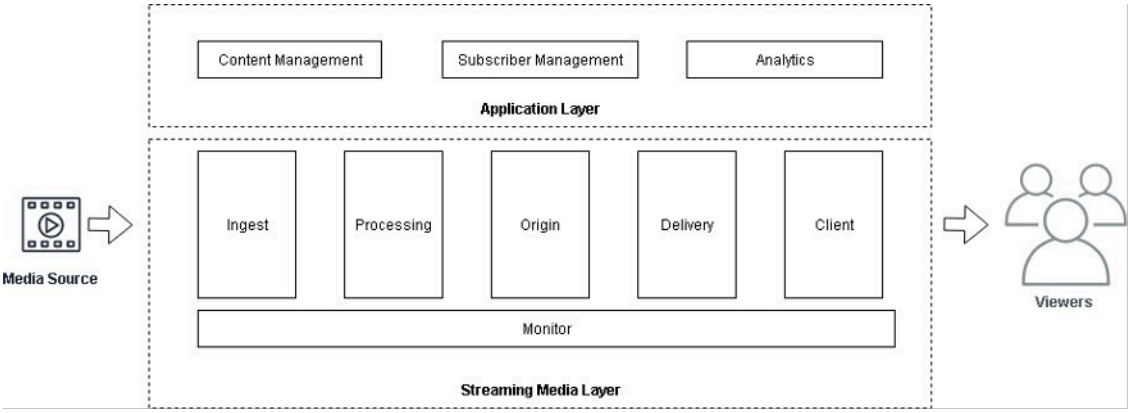
Streaming media is the delivery of audio and video data in the form of small fragments files over a network — typically the internet. As an alternative to a file download, streaming has emerged as the most common method of delivering media to audiences as it optimizes both network bandwidth and local device storage. Common applications of streaming media include:

- **Media and Entertainment** — Direct-to-consumer platforms, user-generated content, sports and gaming
- **Enterprise Marketing and Communications** —Town halls, product announcements, corporate training
- **Retail Advertising** —Promotional materials, influencer engagement, celebrity endorsements
- **Education** —E-learning, webinars, virtual classroom
- **Healthcare and Fitness** — Fitness coaching

You need a considerable amount of infrastructure to process and deliver content to an audience. To cope with unpredictable spikes in demand, such as a live large sporting event, content gone viral, or the acquisition of an asset library, organizations have become accustomed to purchasing more hardware than necessary. These investments strand capital in underutilized infrastructure that could instead be used to develop new features or create new content.

AWS helps customers avoid these large capital expenditures by providing pay-as-you-go resources for streaming media. When you use AWS Media Services, you pay for only the minutes of content processed or the bytes delivered. As your audience grows, you can quickly scale out ingest, processing, origination, and delivery components near viewers to improve quality of service with an expanding global footprint of AWS geographic Regions and hundreds of points-of-presence connected with a fully redundant, high throughput network backbone.

AWS provides over 200 cloud infrastructure services and a network of thousands of AWS Partners to power your streaming media experiences. In this section, we define two conceptual layers and several components that you will find in any streaming media workload regardless of specific implementation. The examples throughout this paper feature AWS Media Services, but the concepts should apply to any streaming media workload.



Streaming media conceptual components

Streaming media layer

The streaming media layer includes five focus areas that we define as the components necessary to transmit audio and video from content publishers to audiences — Ingest, Processing, Origin, Delivery, and Client.

Ingest

Ingest components contribute real-time and file-based content to your streaming media workload. Ingest includes the devices and network used to deliver media sources to cloud entry points for content processing and distribution.

Ingest design will vary greatly depending on the level of control and the access that you have to the content provider infrastructure. Focus should be on the source quality, latency, network performance (jitter, link loss, throughput), security (in-transit encryption, entry point authentication), and redundancy of the ingest architecture.

Encoding devices, such as **AWS Elemental Live** and **AWS Elemental Link**, convert live uncompressed video and audio from Serial Digital Interface (SDI), IP, or HDMI into a compressed format necessary for contributing to AWS over IP network. These devices are deployed on-site at the event or broadcast facility.

Content owners need to manage and transport real-time content throughout the cloud for redundancy or to grant partners or affiliates access. You might ingest source content directly to an Amazon EC2 host entry point, but reliably managing this infrastructure can become burdensome. **AWS Elemental MediaConnect** lets you securely and reliably transport high-quality live video between a remote event site to the AWS Cloud, between services within AWS Regions, or

between partners and affiliates. For live production workloads working with uncompressed video, the **AWS Cloud Digital Interface** allows you to integrate uncompressed video transport into your workflow.

For reliability or performance objectives that require a dedicated network connection from your facility or event site to AWS, **AWS Direct Connect** establishes a more consistent network experience for live streaming or bulk upload of file-based content.

For file-based content, if you aren't able to leverage Direct Connect due to source location, **Amazon S3 Transfer Acceleration** and **AWS Global Accelerator** let you use the Amazon CloudFront global edge network to optimize the network path for content upload. Transfer Acceleration can be combined with **Amazon S3 Multipart Upload** to ingest large media files as a set of parts uploaded in parallel. When all parts of your asset are uploaded, Amazon S3 then presents it as a single object.

For bulk file-based ingest tasks, such as archive migrations and content acquisition that exceed the limitations of network transfer, **AWS Snow Family** provides data migration and edge computing devices that are well suited for large-scale data transfer. **AWS Snowball Edge Edge** provides multiple device options optimized for data transfer (storage-optimized) and edge compute (compute-optimized with optional GPU) for use cases like video analysis and metadata generation in disconnected production environments.

Processing

To ensure quality distribution, many organizations ingest media at a bitrate that is higher than will be distributed to end users, but much lower than uncompressed video. Using an intermediary, mezzanine format affords the flexibility to re-format content in accordance with the evolving landscape of audience device requirements and compression algorithm evolution without requesting new sources from the content provider. Processing content from a mezzanine source also improves the audio and visual quality of the compressed content delivered to your audience.

The purpose of the processing component is to convert video from an intermediary mezzanine format into a format that is optimized for streaming. Processing has two main stages — transcoding the audio samples and video frames, and packaging the media essence data into a container format suitable for client consumption. Transcoding converts content from high-quality, high-bitrate mezzanine codecs into codecs optimized for web delivery, such as h.264. Packaging puts the media tracks (audio, video, captions) into a format that is compatible with consumer devices, typically an Adaptive Bitrate (ABR) format. Processing content into ABR formats like Apple HTTP Live Streaming (Apple HLS) or MPEG-DASH allows content to be streamed at scale over HTTP

and enables the device to select the appropriate rendition given the network throughput to the user. It's also common to embed advertising opportunity signals, insert captions or subtitles for accessibility, and apply content protection schemes while processing media.

AWS Elemental MediaLive and **AWS Elemental MediaConvert** provide broadcast-grade, fully managed video processing and packaging for multi-screen and intermediate formats. Using these services for processing tasks allows you to offload operational and reliability concerns of managing a processing fleet to AWS. Your processing component might also include control plane logic to run a sequence of tasks associated with processing, such as metadata extraction and quality control. Use serverless technologies, like **AWS Step Functions, AWS Lambda,** and messaging services, like **Amazon SQS,** to coordinate workflow tasks for content processing.

Origin

The origin serves streaming media in response to client requests proxied through a content delivery network (CDN).

Amazon Simple Storage Service (**Amazon S3**) provides highly available, highly durable, object storage that can be used as a simple origin for streaming media content when combined with a CDN, such as **Amazon CloudFront**. If you prefer the simplicity and performance of an object store but are looking for media-specific optimizations like stale-manifest deletion, **AWS Elemental MediaStore** provides a performance-optimized content origination service with low request latencies and strong read-after-update consistency necessary for adaptive bitrate protocol manifests.

In addition to serving content, some origin components can apply just-in-time packaging logic to the delivery based on context from the requesting devices. For example, a just-in-time-packaging origin can re-package to multiple protocols or apply different types of DRM from a single set of adaptive bitrate media. If you have business requirements to support dynamic re-packaging of content based on the requesting device, apply multi-DRM, filter adaptive bitrate renditions, or provide DVR-like functionality (start-over, pause, rewind, etc.) to your live stream. Just-in-time packaging origins like **AWS Elemental MediaPackage** make it easy to implement intelligent origination features for live and video-on-demand streaming.

To monetize streaming media content, server-side ad insertion services like **AWS Elemental MediaTailor** manipulates the manifest served by the content origin to insert personalized advertisements. This can be done on Live streaming content or VOD content. Advertisements are seamlessly stitched into the content and can be tailored to individual viewers, maximizing monetization opportunities for every ad break and mitigating ad blocking.

Delivery

The delivery component is the boundary between your infrastructure and last-mile internet service providers that connect you with end users. Typically, this is a CDN with many points of presence (PoP) in major metropolitan areas for edge caching of media, improving client performance and offloading requests from your origination layer. A CDN can also help you secure and control access to content with features such as geo-blocking, SSL termination, and URL tokenization. **Amazon CloudFront** is a CDN service that offers security features such as **AWS WAF** and **AWS Shield**, which can protect your application from malicious requests and distributed denial of service (DDoS) attacks.

Workloads with high-scale and global audiences often use multiple CDNs, but this can increase origin load and reduce cache efficiency. A centralized caching layer, often called an origin shield, helps scale your origin layer by providing a single caching layer to collapse requests from viewers or CDNs into a single origin request. **Amazon CloudFront Origin Shield** is an additional layer within the CloudFront caching infrastructure that can minimize your origin's load, improve its availability, and reduce its operating costs.

Delivering content to a wide device ecosystem might require customizations performed through HTTP header manipulation or stream manifest manipulation. **AWS Lambda@Edge** and **Amazon CloudFront Functions** let you run code closer to users so that you can perform optimizations during request and response without imposing requirements on your origin infrastructure.

Client

A client is any mobile device, personal computer, smart TV, or connected hardware device that can communicate over HTTP, retrieve web assets from the application layer, and render the viewer experience. For streaming media, the client presents a user interface, requests media streams, decodes media, and sends telemetry information about the playback experience to your monitoring and analytics systems. The client is also responsible for working with the browser or device hardware to securely decrypt media. You're responsible for building, deploying, and maintaining your client applications, but **Amazon Interactive Video Service** provides a player and frameworks like **AWS Amplify** help to accelerate your development effort.

Monitoring

The monitoring component provides you with information on the current state of your streaming media workload and quality of experience for the audience.

Monitoring your streaming infrastructure is necessary to detect system-wide performance changes, optimize resource utilization, and to baseline workload operational health. For example, if you're receiving live content from a content provider, you should monitor for the expected number of ingest bytes per second and respond to alerts when ingest fails to meet baseline expectations. AWS services, including AWS Media Services, publish metrics and alerts like these to **Amazon CloudWatch**, which provides data and actionable insights to monitor your streaming media workload.

Re-buffering, failed starts, and long load times will quickly degrade viewer experience and satisfaction. Capturing telemetry information from playback sessions will help you optimize your streaming workload and troubleshoot poor playback experiences. Viewer analytics (seek, skip, play time) should be used also to curate content recommendations or to enable your business leaders make content production decisions. Capturing telemetry and analytics from your streaming media applications is difficult to deploy and scale for high-viewership platforms. **Amazon Kinesis Data Streams** is a massively scalable and durable real-time data streaming service. Kinesis Data Streams can continuously capture gigabytes of data per second from hundreds of thousands of viewers.

Application layer

The application layer is a collection of services and business logic that interacts with the streaming media layer to deliver functions like authentication, authorization, search, recommendations, or interactivity. This layer exposes a set of application programming interfaces (APIs) and serves data in response to requests from clients. Though your application will have a unique set of endpoints, there are a common set of services deployed alongside streaming media applications.

- **Subscriber Management** — A method for user authentication and media playback authorization is necessary for transnational or subscription services. Services like **Amazon Cognito**, when combined with URL tokenization and content encryption, can be used to secure access to content.
- **Content Management** — A purpose-built database for indexing content and associated metadata. AWS Database Services like **Amazon DynamoDB** or **Amazon Relational Database Service** are common backend services used for Content Management state.
- **Analytics** — A system for ingesting client analytics regarding playback behaviors (content preference, play-through, skip, pause, re-watch) and extracting business insights for decision-making. In the streaming media layer, telemetry data (error rates, buffer rates, latency etc) is used to maintain quality of experience during playback, but services like **Kinesis Data Streams** can also be used to populate data warehouses with **Amazon Redshift** or **Amazon S3**. These

long-term warehouses can be examined by business stakeholders to make strategic decisions regarding feature enhancements to the service or to produce content relevant to viewer interests.

- **Interactivity** — A service, commonly aligned with video playback using embedded metadata, that provides audience engagement through interactions with the host, audience members, or metadata to enrich the experience. **Amazon IVS** and **AWS Elemental Live** provide timed metadata interfaces that are used to embed data directly within the video stream and initiate events to call application APIs to render relevant features.

Design principles

In the cloud, there are a number of principles that can help you create and run streaming media workloads. Keep these in mind as we discuss best practices:

Keep a Glass-to-Glass Perspective — From camera to viewer device, always consider how change impacts the critical path of video delivery. For example, adding a transcoding component can increase live latency, reduce video quality, and introduce another point of failure in your workload.

Understand Your Audience — Build a deep understanding of your audience to inform business and technical decision making. Use analytics to uncover the network conditions, devices, and displays that your audience uses to optimize for their unique consumption habits and prioritize playback testing. Make sense of interaction data and proactively engage with your audience through recommendations, personalization, and notifications. Analyze viewership data and forecast usage to validate the performance of all the components in the workflow at peak scale.

Know your Content's Value — Start your design with a shared understanding of the content value between stakeholders. This will help you make decisions and tradeoffs around overall costs, reliability objectives, performance, and content protection schemes.

Design for Disruption — While care should be taken to minimize failures, you should expect disruption. In designing for disruption, start with a shared reliability objective across the organization taking into account the content value. Aligned to your reliability objective, build redundancy into each component and the network links between them. Also, seek to understand the areas of concern between components and architect to gracefully handle failure or impairment of components.

Streaming media scenarios

This section addresses common streaming media scenarios, with a focus on how each scenario impacts the architecture of your workload. These examples are not exhaustive, but encompass common patterns we encounter in practice. We present a background on each scenario, general considerations for the design of the workload, and a reference architecture of how the scenarios should be implemented.

Though there are many approaches to designing video delivery workloads on AWS, these scenarios prefer AWS Media Services when possible to reduce operational overhead. We encourage you to evaluate the technology architecture that best meets your needs, whether it is an AWS Media Service, a service from the AWS Partner Network, or a proprietary component.

Scenarios

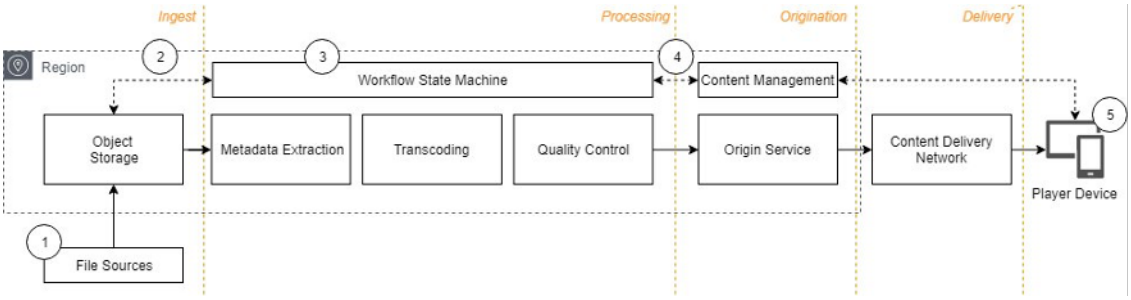
Scenario: Video-on-Demand streaming

Video-on-Demand (VOD) or file-based streaming is the most common streaming scenario. VOD content is shared by millions of content creators across user-generated content sites, used in the enterprise for training, or used to deliver popular movies and television shows to the home.

There exists a wide range of file-based streaming architectures. If your content is already in an adaptive bitrate format, an **Amazon S3** origin and a CDN is all that you need to serve viewers. However, if your content is in a high-bitrate format, you need a batch processing component to convert the media into a distribution format.

There are many ways to deploy a batch process on AWS for media processing. You can deploy transcoding software on **Amazon EC2** instances directly, use Docker containers with **Amazon ECS** and **AWS Batch**, or use services from AWS or AWS Partners. If you're getting started with VOD, we recommend using **AWS Elemental MediaConvert**. **MediaConvert** helps you scale transcoding to meet demand, keep costs low, and retain video quality without the overhead of managing any infrastructure.

You might need additional logic or steps in your workflow to extract content metadata, register content with content management systems, make intelligent processing decisions, or to introduce quality control steps. A common approach is through a serverless design that uses **AWS Step Functions** to model your workflow, **AWS Lambda** to run logic, and services like **AWS Elemental MediaConvert**, **AWS Batch**, and **Amazon ECS** to scale compute-intensive media processing.



Video-on-Demand (VOD) streaming architecture

1. Content is uploaded through an application or file transfer tools.
2. Object Created event emitted by object storage invokes the media processing workflow.
3. Workflow progresses through batch processing stages like analysis, transcoding, and quality control.
4. Completed content is published to origin service and made available through content management APIs.
5. Players request content from origin service and begins viewing session.

Considerations

- Communication within a serverless workflow is typically done by event publishers, messaging services, and subscribed functions. Operations triggered from events must be idempotent, as failures can occur and events can be delivered more than once.
- Consider using separate buckets for ingest and origination to prevent unintentional access to source assets.
- Media analysis tools and machine learning can be used to extract source characteristics and drive intelligent processing decisions.
- To reduce delivery and storage costs, explore compression technologies such as AWS Quality-Defined Variable Bitrate (QVBR) that provide streams at lower bitrates but equivalent video quality.

- Many sources only need to be read once for processing. Use aggressive lifecycle policies to optimize object storage costs for source material.
- Processing and storing media in multiple protocols or digital rights management (DRM) schemes can increase storage costs by orders of magnitude. Architectures that serve multiple formats or media endpoint customization should use a just-in-time packaging (JITP) origin.
- Always use a content delivery network (CDN) to improve time-to-first-frame performance and offload origin request rate.
- Refer to the Video on Demand on AWS solutions implementation for a well-architected reference architecture for Video-on-Demand.
- Review the Well-Architected Serverless Applications Lens for best practices regarding serverless architectures.

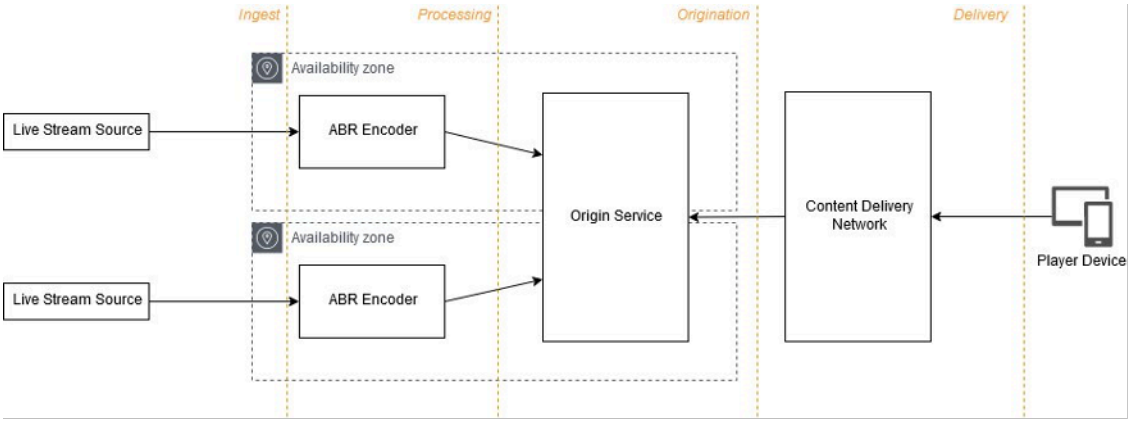
Scenario: Live streaming

Live video streaming is core to many exciting events, news, and interactive experiences. It is used extensively in social media, sporting events, corporate streaming, e-sports, broadcast TV, as well as IoT and security camera applications. Due to the real-time nature of live content, special considerations must be taken when architecting live streaming solutions. As with other streaming media workflows, the fundamental architecture components include ingest, processing, origination, and delivery.

One of the unique challenges of live video streaming is that the value of the content being streamed decreases exponentially as time passes from the *live* point. For example, many people might want to watch a sporting event live, but the number of viewers who watch the event hours or days after it takes place tends to decrease as time goes on. Distributors only have one chance to get a live event right, and do not have the luxury of asking viewers to come back later when issues are resolved. As such, it is critically important to ensure that live streaming workflows are optimized for reliability and performance.

The cloud is uniquely suited to handling these challenges, with its regional diversity, availability, and elastic scalability. By decoupling the various services involved in live streaming, reliable and performant workflows can be built that are cost effective and where the various architecture components can scale elastically to handle variations in demand or in live stream counts. It also allows flexibility to include value added services, such as machine learning technologies to perform actions such as automated closed caption/subtitle generation or content tagging/indexing. This type of redundancy, scalability, and developer flexibility is not possible with monolithic on-

premises live streaming applications. The following diagram illustrates a sample live streaming workflow using redundant sources and a Multi-AZ architecture:



Live streaming architecture

Considerations

- **Amazon Interactive Video Service (Amazon IVS)** is a managed live streaming service that manages low-latency interactive live streams on your behalf. Consider using Amazon IVS if you don't want to manage your own live streaming service.
- Ingest your video content in the AWS Region that is closest to the source of the stream.
- Where subsecond latency is required (that is, video conference-like applications) protocols such as WebRTC should be considered. However, these solutions do not scale as effectively for *one-to-many* distribution as they require stateful connections with backend origin services. Amazon Kinesis Video Streams and Amazon Chime SDK can both be used to implement WebRTC into your applications.
- Use a reliable ingest protocol such as Zixi, SRT, RIST, RTP-FEC, or RTMP as these are designed to improve reliability over unmanaged networks, such as the internet.
- Consider source ingest within at least two AWS Availability Zones from diverse network paths.
- To reduce delivery and storage costs, explore compression technologies such as AWS Quality-Defined Variable Bitrate (QVBR) that provide streams at lower bitrates but equivalent video quality.
- Use an origin server specifically designed and optimized to support the latency and data consistency models required for live streaming.
- Origin hosts can scale to serve streams to tens or even hundreds of viewers, but it is always recommended to use a CDN such as Amazon CloudFront to scale live stream delivery beyond a handful of viewers.

- Refer to the Live Streaming on AWS solutions implementation for a well-architected reference architecture for live streaming.

Scenario: Ad supported content monetization

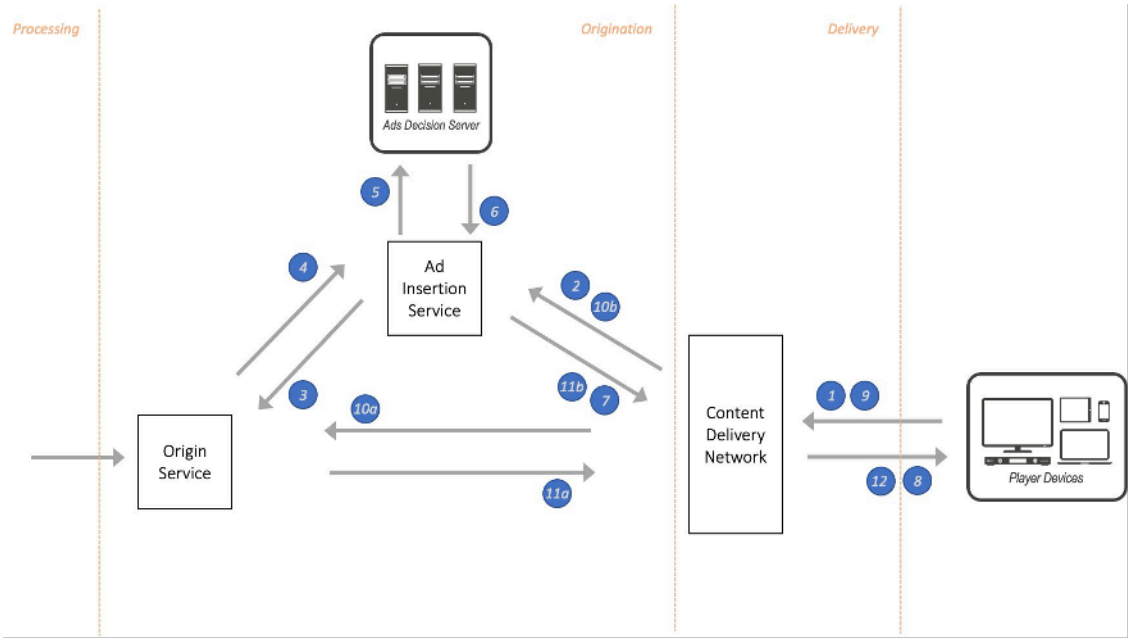
Advertising supported monetization is achieved by either inserting or replacing advertisements in the content. Targeted and personalized advertisements lead to higher engagement and maximize the return on investment for advertisers. With a well-designed ad insertion service, you can deliver a streaming media experience to millions of concurrent viewers across a range of multiscreen devices with personalized ad content, creating a seamless broadcast-like experience.

Tracking ad impressions is an important part of ad insertion and is accomplished through the client beaconing of quartile views as it plays ad segments. Capturing beaconing data directly from the players in this way reduces the effects of ad blocking software and adheres to established advertising open standards. Accurately measuring ad impressions and viewing behavior across web, iOS, Android, and other connected viewing devices, helps more effectively measure the revenue impact of every ad delivered.

Ad insertions can be achieved either at the client-side or server-side. Traditional client-side ad insertion requires working with various player SDKs across multiple platforms to maintain ad logic for different ad formats and content variations. In addition, ad blockers (software preventing online advertising from being displayed) are easily able to detect client-side ad insertion and can limit the impressions.

On the other hand, server-side video ad insertion (SSAI) is a method in which video ads are integrated into the stream at the origin. With SSAI, there is no need for the client to switch between content and the advertisement. Buffering and session initialization delay before and after the insertion of the ad are eliminated, providing a higher quality viewing experience. Compared to client-side ad insertion, SSAI typically improves the ad experience and impressions and provides an opportunity to personalize advertisements.

A common approach to SSAI is shown in the following figure. In this architecture, we reflect cloud services for the origin server, ad insertion service, and the content delivery network. These cloud-native services are highly available, natively redundant, and ensure that the customer experience is not compromised in the event of host-level failure.



Server-side ad insertion architecture

1. Device requests content manifest to begin playback session.
2. The CDN forwards the request for a personalized manifest to the ad insertion service (AIS).
3. The AIS consolidates requests for the manifest from all players and forwards only one request to the origin.
4. The origin responds back with the content manifest, optionally including ad markers.
5. If the manifest has ad markers, the ad insertion service makes a call to the ads decision server (ADS) to get a list of personalized ads to insert for the playback session.
6. The ADS returns a Video Ad Serving Template (VAST) response with a list of personalized ads.
7. The AIS inserts the ads and returns a personalized manifest back to the CDN.
8. The CDN returns the personalized manifest to the player.
9. The personalized manifest points the player to content segments and ad segments.

10a. If the segment is a main content segment, the CDN passes that request to the origin.

b. If the segment is an ad segment, the request for the segment is routed to the AIS.

11a. The origin returns the content segment.

b. The AIS returns the ad segment.

12. CDN serves the segment to the player.

Considerations

- Use the same AWS Region for both the ad insertion service and the origin server.
- Use a CDN to cache content and ad segments. However, personalized manifest responses must *not* be cached or shared between viewers. For more information, refer to CDN Best Practices.
- Select the ad insertion service that supports inserting ads into all the desired streaming protocols, such as, HLS, DASH, and MSS. AWS Elemental MediaTailor supports ad insertions to HLS and DASH streaming formats.
- Target an ad insertion latency that's less than the segment length to minimize playback rebuffering.
- Design your client to handle a mix of encrypted and unencrypted content during playback. Content is usually encrypted while ad segments aren't.
- To maintain consistent video quality between main content and ad segments, ensure that the ad insertion service processes advertisements to match the bitrate, frame rate, and resolution of the main content.
- Refer to This is My Architecture video to learn how Amazon Prime Video scales ad measurement and tracking at scale on AWS.

With these definitions, design principles, and scenarios in mind, let's move on to pillar-specific guidance that you can use to evaluate and continuously improve your streaming media workloads.

The pillars of the Well-Architected Framework

This section describes each of the pillars, and includes definitions, best practices, questions, considerations, and key AWS services that are relevant when building solutions for streaming media as is intended to be read as a companion to the AWS Well-Architected Framework whitepaper. When designing your architecture, we recommend that you read both papers and examine the full set of considerations.

Pillars

- Operational excellence pillar
- Security pillar
- Reliability pillar
- Performance efficiency pillar
- Cost optimization pillar
- Sustainability pillar

Operational excellence pillar

The operational excellence pillar includes operational practices and procedures used to manage production workloads. This includes how planned changes are performed, as well as responses to unexpected operational events. Change execution and responses should be automated. All processes and procedures of operational excellence should be documented, tested, and regularly reviewed.

Best practices

There are three best practice areas for operational excellence in the cloud:

- Preparation
- Operations
- Responses

To drive operational excellence in streaming media workloads, preparation is essential. Many operational issues can be avoided by following best practices when designing the workload, and

fixes are less expensive to implement in design phases rather than in production. Operations process and procedures must be thoroughly planned, tested, reviewed. Workloads should evolve and be changed in automated and manageable ways. Changes should be small, frequent, and incremental, all without impacting continuous operations. Operations teams must be prepared to respond to operational events and failures, and have processes in place to learn from them.

Preparation

Load testing builds confidence in your infrastructure design before a large event and highlights weaknesses early. When possible, use historical data to model and test system load. For example, if you've acquired content streaming rights, but a similar show or event has aired in the past, Nielsen or Comscore data can provide useful estimates for peak concurrent viewers. With this model, you can develop accurate load tests that simulate real-world conditions. Be sure to test both individual service components *and* the end-to-end system with load tests that emulate end user clients.

SM_OPS1: How do you prepare for large-scale streaming media events?
SM_OBP1 – Ensure that your content delivery infrastructure can scale to the expected demand
SM_OBP2 – Evaluate and adjust Service Quotas to match your workload's needs in advance
SM_OBP3 – Engage with AWS Support programs to assist with events

For origin services, estimate load in Transactions Per Second (TPS) or Requests Per Second (RPS). We advise that you estimate peak TPS the player devices will generate based on the usage pattern, layers of caching, and unique nature of your audience.

For example, for an HLS stream with 6-second segments and un-muxed video, audio, and captions, each player will generate one TPS (= 3*1/6 TPS for video, audio, and caption manifest requests + 3*1/6 for video, audio, and caption segment requests) to the CDN. Assuming a 95% cache hit ratio (CHR) for the CDN, it will result in 0.05 TPS (=1*(1-0.95)) to the origin. To calculate peak TPS to the origin, multiply the peak number of player devices expected at any time with 0.05 TPS.

For DASH streams, the TPS depends on how frequently the DASH player requests the manifest and segments. A common practice is to request once every segment length. Note, that all DASH segments are unmuxed, that is, the player will request a video, audio, and caption segments each time. So, for a 6-second segment, the TPS would be 0.66 TPS (= 1/6 for manifest + 3*1/6 for video,

audio and caption segments). Again, assuming a 95% cache hit ratio (CHR) for the CDN, that would result in 0.033 TPS (=0.66*(1-0.95)) to the origin. Based on the forecast for peak concurrent HLS and DASH playback sessions, peak TPS for the origin can then be calculated.

You can use the following TPS formulas to guide your estimations, but, when available, always use your own metrics and data to forecast usage.

Streaming Protocol	TPS Formula
HLS	Peak audience x (2 x (Number of elementary stream segments)/(segment length)) x (1-CHR)
DASH	Peak audience x (4/(segment length)) x (1-CHR)

Estimating transactions per second for common streaming media protocols

For HLS, the number of elementary stream segments depends on whether the video, audio, and caption date is muxed in one segment or unmuxed in separate segments. In case of muxed segments, there is only one elementary stream segment whereas in the case of unmuxed segments, there are three elementary stream segments.

AWS Cloud can handle very large scale streaming events. However, services have soft limits in place to protect customers from inadvertently scaling resources beyond their needs and racking up unnecessary costs. Service quotas (also referred to as limits) for all media services are Region-specific unless otherwise noted in the documentation. For more information about the quotas for a specific service, refer to its documentation.

You can centrally manage Service Quotas for all services in your account using Service Quotas in the **AWS Management Console** where you can view your current quotas and also request increases. If you don't find your AWS service in Service Quotas, contact AWS Support to increase them. Provide details of the Regions in which your workflow will operate, usage patterns, and timeframe.

Consider using Infrastructure and Event Management (IEM) and Media Events Management programs for large-scale live events that might require immediate support during an event. By engaging AWS through these programs, you enable AWS experts to become familiar with your workload, provide architectural and operational guidance, and real-time support for your planned event.

In addition to IEM, Enterprise Support customers are eligible for a Cloud Operations Review and a Well-Architected Framework Review designed to help identify risks in your cloud operations. This cross-team engagement helps establish a common understanding of your workload and helps AWS contribute to your streaming events. You can always perform your own architecture review for any workload by using the AWS Well-Architected Tool in the AWS Management Console.

Operations

AWS enables visibility into your streaming workload at all layers through log collection and monitoring features. Data on use of services, resources, application programming interfaces (APIs), network flow logs, and system traces can be collected using **Amazon CloudWatch**, **AWS CloudTrail**, **VPC Flow Logs**, and **AWS X-Ray**. Equipped with this data, you can design automated failure and remediation systems at each stage in your video application – ingest, processing, origin, delivery, and client-side.

For live streaming, expressing the component relationships and tracing the signal path is important for operators who need to identify and respond to issues that arise. Visual documentation or interactive dashboards that reflect the real-time status of the workflow will improve awareness and shorten time to issue resolution. Consider using or building your own tools like the Media Services Application Mapper to model your workload and better inform operations.

One unique property of media streaming is that while every component in the workload can be operating as expected, the resulting audio or video might not be the intended content. Consider placing video decoder probes throughout your live stream signal path to emit thumbnail images or low bitrate proxy media to monitoring systems. This will ensure that the correct content is being transmitted and improve operational observability for troubleshooting.

Resources

Refer to the following resources to learn more about AWS best practices for operational excellence.

Documents

- Infrastructure Event Management
- Load Testing CloudFront
- Media Services Application Mapper
- Monitoring AWS Media Services using Amazon CloudWatch Events

Security pillar

The security pillar describes how to take advantage of cloud technologies to protect data, systems, and assets in a way that can improve your security posture.

Best practices

There are five best practice areas for security in the cloud:

- Identity and access management
- Detective controls
- Infrastructure protection
- Data protection
- Incident response

Securing streaming media services includes limiting access to system components and assets to prevent theft, impairment, and destruction, as well as detecting malicious activities such as unauthorized access to assets and resources. Unauthorized access to assets devalues premium content and ultimately impacts revenue. Destruction of assets or impairment of resources may cause an outage, and prevent your customers from viewing content, which might have a monetary business impact.

In this section, we provide the best practices available to protect your streaming media services. Although this paper is focused on streaming media, it's expected that you will also be following the security best practices of the AWS Well-Architected Framework whitepaper.

Identity and access management

Streaming media workloads carry audio and video for many different purposes. In entertainment, these streams often carry high-value, licensed content delivered to large audiences. In a corporate setting, where streaming media is increasingly used to connect with employees, streams can carry commercially sensitive material. A strong identity foundation protects your content, viewers, and confidential business information.

SM_SEC1: How do you authorize access to content and content ingest?

SM_ SBP1 – Use an identity provider to authenticate viewers and access policies to implement least privilege access to protected content

SM_SBP2 – Restrict content origin access to allow only authorized content distribution networks

Whether on mobile, desktop, or SmartTV, web applications serve as the box office for audiences interested in your content. Authentication and authorization systems help ensure that only authorized users can access content in the way you intend. For example, a user authenticates through sign-up/sign-in and is authorized to access only *free* or *ad supported* content tiers based on their current subscription plan. Identity is essential for a centralized strategy on authorizing access to resources.

Access to private content should be granted only to authenticated and authorized viewers using an identity provider (IdP). On AWS, Amazon Cognito can be used as an IdP to authenticate users and authorize access to content hosted on Amazon S3, AWS Elemental MediaPackage, AWS Elemental MediaStore, or on a custom origin service built on Amazon EC2. You can also establish trust between identity providers to avoid sharing credentials and simplify the authentication flow for your media player. Amazon Cognito provides both temporary AWS credentials as AWS STS (Security Token Service) tokens, as well as JWTs (JSON Web Tokens), to access protected resources. Amazon Cognito also allows you to federate an identity pool or user pool with different identity providers, such as SAML providers like Active Directory Federation Services or Okta, OIDC providers such as Auth0, and other public identity providers such as Google, Twitter, or Facebook.

In addition to leveraging an IdP, centralize resource access control based on the identity established by IdP through the application API layer. For example, Amazon API Gateway and AWS AppSync allow you to specify an Amazon Cognito User Pool as an IdP for the resources being protected, so that bearer tokens can be validated before granting access. Amazon API Gateway and AWS AppSync also allow you to create custom authorizers, so that you can perform additional application logic to allow or deny access to a resource based on claims in the access token, or if a non-supported token, such as SAML, is provided to the API.

Even authenticated users can act maliciously with your workloads, so you should consider how to secure the data path of your video streams. Tokenization schemes such as signed-URLs, signed-cookies, or JWTs (JSON Web Tokens) should be used to grant only temporary access to content by

approved front-end applications. Amazon CloudFront can protect access to content origin through signed URLs and signed cookies with a short duration to live, and Lambda@Edge can validate bearer tokens during viewer request.

When using content distribution networks to accelerate distribution to viewers, help to protect your content origin from unauthorized origin access by validating a secret header injected by the CDN transmitted over TLS at the time of request and use a policy that prevents access from all other entities.

When using AWS Elemental MediaStore as an origin, you can configure MediaStore to accept requests to your container only if the user-agent header value is set to a shared secret value with IAM Policy Conditions. With Amazon CloudFront, your distribution can identify itself by injecting the user-agent header secret value during requests for objects within the MediaStore container. This method can also be applied to content origin services running on Amazon EC2. You can apply this secret check on the service itself and employ a web application firewall, such as AWS WAF, to perform the check on your behalf.

When serving video-on-demand content from Amazon S3, configure Amazon CloudFront to use an origin access identity (OAID) and then restrict access to your Amazon S3 bucket by placing an S3 bucket policy to allow access from your Amazon CloudFront distribution only if it identifies itself with that OAID. OAID, when combined with Signed URLs and user authentication, is designed to ensure that only requests through Amazon CloudFront will return successfully and prevent any direct requests to your bucket origin, and will negate the need for direct access to your buckets.

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/
CloudFront Origin Access Identity 111122223333"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

Example S3 bucket policy for CloudFront origin access identity (OAID)

Detective controls

You can use detective controls to identify a potential security threat or incident. Viewer access patterns provide streaming platforms with a rich data source that can help publishers create engaging content, improve playback experiences, and identify potential security risks. However, unwanted user behaviors, such as credentials sharing and credential compromise, can lead to unauthorized access to your content. A combination of client and infrastructure logging can be used to baseline expected content access behaviors and alert upon deviations.

SM_SEC2: How do you monitor access to your media distribution workload?
SM_SBP3 – Monitor for fraudulent access attempts
SM_SEC3 – How do you monitor unauthorized re-distribution of your content?
SM_SBP4 – Implement content or sessions forensics

For example, content requests through Amazon CloudFront can be logged and aggregated into Amazon S3. Amazon Athena can then query this access data for abnormalities like:

- **Request location** — Are requests only coming from geographic Regions where you would expect? Is the user location obfuscated by a downstream provider?
- **Request IP** —Is a specific IP address requesting content in a pattern that reflects normal viewing habits?
- **User Agent** —Is the user-agent string from the device one that is known and valid?

Monitor activities such as sign-in attempts from new locations and devices, assign a risk score based on the activity, and decide to either prompt users for additional verification or block the sign-in request. You can notify users of suspicious sign-in attempts and prompt them to secure their accounts. You can also view a history of sign-in attempts and their risk scores. The advanced security features in Amazon Cognito can also help you identify password sharing, reuse, or theft.

While monitoring can help to protect from unauthorized platform access, you should implement controls that can help you when valued content is distributed without consent. It is unlikely that

you can completely prevent a viewer from copying content, but forensic controls can greatly improve incident response when improper distribution is detected.

As a simple example, you might have seen content on an inflight entertainment system that had the name of the airline embedded on the content. This overlay might appear periodically or throughout the entire piece of content and is used by content owners to determine if leaked content originated from an airline. Video encoders like AWS Elemental MediaConvert can burn in these visible watermarks into your content as an identifiable image overlay. While simple and effective for a few unique watermarks, this method requires a unique piece of content for each watermark and is therefore limited by the cost of storing multiple versions of the same content.

For large-scale per-viewer, implement a content identification strategy that allows you to trace back to specific clients, such as per-user session-based watermarking. With this approach, media is conditioned during transcoding and the origin serves a uniquely identifiable pattern of media segments to the end user. A session to a user-mapping service receives encrypted user ID information in the header or cookies of the request context and uses this information to determine the uniquely identifiable pattern of media segments to serve to the viewer. This approach requires multiple distinctly watermarked copies of content to be transcoded, with a minimum of two sets of content for A/B watermarking. Forensic watermarking also requires YUV decompression, so encoding time for 4K feature length content can take upwards of 20 hours. DRM service providers in the AWS Partner Network (APN) are available to aid in the deployment of per-viewer content forensics.

Infrastructure protection

Infrastructure protection for streaming media involves securing all resources from end to end, which includes the ingest endpoints, content origin endpoints, DRM services, and the client, from unintended or unauthorized access or potential vulnerabilities.

SM_SEC4: How do you protect content ingest endpoints?
SM_SBP5 – Encrypt content ingest traffic using TLS
SM_SBP6 – Use private connectivity when working with partners
SM_SBP7 – Encrypt content at rest when delivering via physical medium

Streaming media services depend on a reliable content ingest endpoint to upload, process, and deliver engaging content. These endpoints need to support transit protection to ensure that content being uploaded can't be intercepted or intentionally degraded during transit. Live and file-based content uploads can be accomplished in several ways:

- Direct upload over the public network to a custom processing fleet or a cloud service
- Private network connectivity for direct uploading to a custom processing fleet or a cloud service
- Offline delivery of content to a remote facility to store and process

To help ensure that content cannot be intercepted between the publisher and ingest endpoints, you should encrypt uploads in transit and use TLS at both the source and destination. To simplify configuration for global connectivity over public network paths, you should use anycast networks, such as AWS Global Accelerator, which helps clients connect to the closest available endpoint.

You can use AWS Direct Connect and Direct Connect Gateway to connect your network to an AWS Region and bypass public network paths between content source and cloud infrastructure. With Direct Connect, you establish a dedicated connection between a provider network and one of the Direct Connect locations. Established connections from a Direct Connect location to any other AWS Region around the world communicate over the AWS managed backbone—improving performance for geo-diverse media workloads while limiting the routers, networks, and parties involved in the physical transmission layer.

AWS PrivateLink is recommended to establish connectivity when working with partners that also use AWS. With AWS PrivateLink, a service provider can expose their service endpoint to you within Region, avoiding communication over the public networks. When a service provider provisions a PrivateLink endpoint in the consumer's VPC, traffic can never initiate from that endpoint, and only receive requests from the consumer's resources. Traffic flowing through the AWS PrivateLink VPC endpoint will adhere to routing rules and network access control lists placed on that subnet in which the endpoint resides.

When working with large content libraries, it's possible that transmitting content over the network is not feasible. This is especially important to consider if the connectivity between the studio and the video processing infrastructure is non-existent or if the bandwidth requirement to transmit the footage is beyond the available bandwidth the network provider can provide. AWS offers **AWS Snowball Edge Edge**, a petabyte-scale data transport solution that uses secure appliances to transfer large amounts of data into and out of the AWS Cloud. AWS Snowball Edge Edge encrypts all data with 256-bit encryption. You manage your encryption keys by using the

AWS Key Management Service (AWS KMS). Your keys are never stored on the device and all memory is erased when it is disconnected and returned to AWS. A user must have access to the customer managed key (AWS KMS key) that is associated with the Snowball Edge Edge device when it was requested to access the data stored in the Snowball Edge Edge device, which reduces concerns of data being intercepted in transit. Snowball Edge Edge devices come with an electronic screen that displays the customer and AWS shipping destination, which minimizes shipping discrepancies. Lastly, after your data has been transferred to AWS, your data is erased from the device using standards defined by National Institute of Standards and Technology.

SM_SEC5: How do you protect content origin from unauthorized access and malicious attacks?
SM_SBP8 – Use DDoS protection service to maintain content availability
SM_SBP9 – Restrict content origin access to only allow known entities
SM_SBP10 – Use a web application firewall to monitor and control content access
SM_SBP11 – Encrypt origin to client communication in transit using TLS

Protect your content origin layer from distributed denial of service (DDoS) attacks at both the network level (Layer 3) and application level (Layer 7), in addition to preventing unauthorized origin access. Protecting your content origin from unauthorized access or malicious attacks can help prevent improper re-distribution of private content and increase service reliability.

A DDoS attack is when multiple systems intentionally flood your resources, which can render your content origin unavailable or hidden to your viewers. It is important to use a DDoS protection tool, such as AWS Shield, to protect your resources. AWS Shield protects AWS resources such as **Amazon CloudFront** distributions and **Amazon Route 53** so that your content can be located and reached globally. **AWS Shield Advanced** protects resources built upon services such as **Elastic Load Balancing, Amazon EC2**, and **AWS Global Accelerator** against common and most frequently occurring infrastructure (layer 3 and 4) attacks like SYN/UDP floods, reflection attacks, and others to support high availability of your applications on AWS. If you need to protect resources that you are hosting privately, put a CDN, such as a CloudFront distribution, in front of it.

To reduce the likelihood of impact on your content origin from a volumetric attack such as a DDoS attack, limit the allowed traffic sources to trusted client IP addresses, such as the IP address ranges for your CDN.

When using **AWS Elemental MediaPackage** or a content origin built on **Amazon EC2**, restrict requests to originate only from known IP addresses of the CDN PoPs and, if applicable, use security groups to restrict incoming traffic. To isolate access to known **Amazon CloudFront** IP addresses, AWS provides a JSON resource that includes those address ranges, which is regularly updated.

If you are using AWS Application Load Balancers or Amazon CloudFront, you can also use **AWS WAF** (Web Application Firewall) to validate requests originating from known IP addresses. **AWS WAF** lets you create rules to filter web traffic based on conditions that include IP addresses, HTTP headers and body, or custom URIs.

Data protection

SM_SEC6: How do you protect content at-rest and in-transit to prevent unauthorized distribution?

SM_SBP12 – Collaborate with business and legal stakeholders to align on content protection requirements

SM_SBP13 – Select a content protection scheme that meets business objectives

The Well-Architected Framework covers best practices for any workload when protecting data in transit and at rest, but when you serve copyrighted materials or high-value content, you should also consider the technologies available to protect these works from unlawful access, replication, and re-distribution. In general, practices that protect valuable digital assets are referred to as Content Protection or Digital Rights Management (DRM). There are varying degrees of complexity and systems involved in Content Protection, but these components are common to most implementations:

- **Key Provider** — Manages customer master keys and content keys.
- **Encryptor** — Encrypts media. This is often an encoder or an origin responsible for packaging.
- **Authentication Service** — Manages subscriber access and generates temporary access tokens requests to content and keys.
- **Client** — Authenticates viewer, retrieves content from an origin, keys from a key provider, decrypts, and renders media.

In practice, there are two common content protection schemes — Clear Key Content Encryption and DRM systems. Both systems use AES-128 to encrypt the content, but differ in how keys are managed and delivered. If you've licensed content from a third party, you might be commercially obligated to implement a DRM system. This is true of most film and television content. Always implement the solution that is in alignment with your business and legal requirements.

Clear key content encryption and tokenized access

A clear key implementation is common for applications that don't require *Hollywood-grade* DRM systems, but want a pragmatic content protection scheme that makes it difficult for outright theft or improper sharing of content by paying users. By encrypting content with AES-128 encryption, we can control who can decrypt the content through key access policies. When combined with techniques like OAID and tokenized, temporary, access URLs, you can control who can retrieve the decryption keys and for how long the content endpoint will service requests for encrypted content. This method is called *clear key* because the content key is eventually *in the clear* within the user-space of the client and could theoretically be accessed by an unauthorized user. Though, with tokenized content endpoints, having the key does not necessarily mean that an unauthorized user would have access to retrieve and decrypt the content.

On AWS, clear key encryption can be accomplished by generating an AWS KMS key and then using the API to create data keys that can be used by an encryptor (typically an encoder or packager) to apply encryption to the content. These data keys are then stored in a scalable persistence layer like **Amazon S3** or **Amazon DynamoDB.** Data key access can be granted to your audience through a combination of **Amazon Cognito** and **Amazon IAM** policies for user authentication and authorization. Delivery of these keys should always be over an encrypted transport with tokenization.

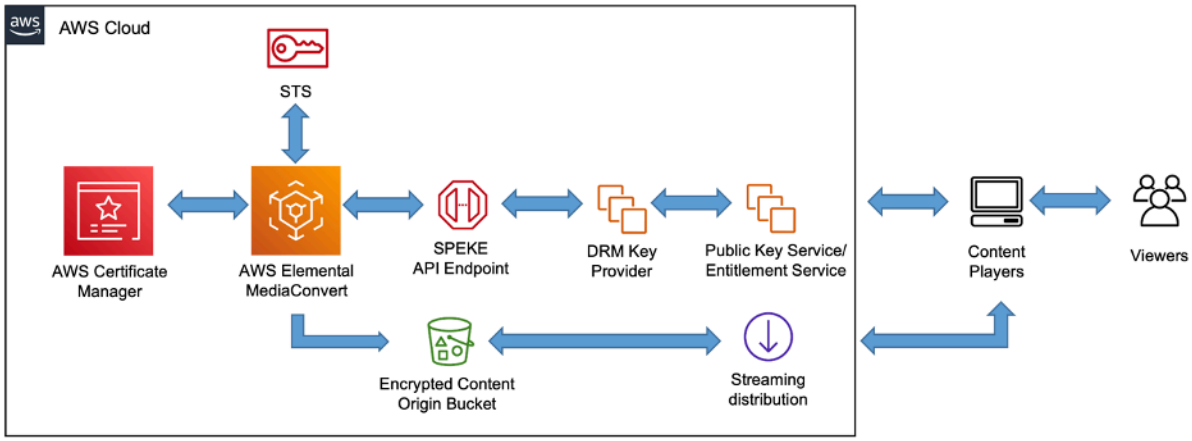
DRM systems

Organizations working in the media and entertainment industry or with high-value content, might have strict content protection objectives for content keys, content decryption, or both to be run only within hardware modules or trusted execution environments (TEEs) that exist outside of the user-space on the client. These organizations might also have requirements to facilitate key revocation, offline playback, single-use keys, or multi-key encryption levels for the same asset. These complex organizational objectives can be achieved through the implementation of a DRM system, such as Apple FairPlay, Google Widevine, or Microsoft PlayReady.

While DRM systems add an additional layer of key protection and business control to your content protection, implementation of DRM varies by playback device. Though the industry is making

strides to simplify DRM implementations, in practice, it's common to see multiple DRM systems implemented to achieve device compatibility across playback devices—increasing cost and complexity.

With the **AWS Media Services**, DRM systems are integrated into media processing and origination though the Secure Packager and Encoder Key Exchange or SPEKE. SPEKE provides an open standard proxy interface for any key provider to exchange key material and metadata. You can implement your own key service or use one of many DRM system providers that are part of the AWS Partner Network (APN).



Secure Packager and Encoder Key Exchange (SPEKE) architecture

No number of content protection schemes can ever fully protect content from being exploited by an attacker and, in fact, complex schemes can even increase the risk of problems for your paying viewers. Commit time to determine the appropriate content protection schemes for your specific content balancing cost with content value. Be sure to balance cost of additional resources on software licensing or operational burden versus the business value of the content being protected.

Clear Key Content Encryption and Tokenized Access	Digital Rights Management (DRM) System
(PRO) Supported by all browsers through EME and mobile players	(CON) Support varies by device and browser, often requiring multi-DRM approach and increased costs
(PRO) Encryption implementation can be done with standards-based, open sources tools and most encoding and packaging systems	(CON) Encryption implementation varies by DRM system provider and may require A commercial agreement to implement
(PRO) Simple backend implementation	(PRO) Decryption key is only transferred to a protected space in device memory or in browser memory (EME) TEE
(CON) Decryption key is supplied to player *in the clear* at some point in the playback	(PRO) Support application of complex business rules. For example - offline playback, revocation, single-use tokens, multi-key encryption to enforce separate policies

Comparison of clear key and DRM systems for content protection

Player integrity

Take proper measures to prevent discovery of the controls put in place to prevent usage of a tampered player or unauthorized access to your media assets and distribution system. When you choose to create your own media player, you should determine which platforms will help you check the integrity of your software packages.

Many platforms, such as FireOS, Android, iOS, macOS, and Microsoft Windows, provide the capability for developers to sign their application packages as part of the process of uploading them to the platform's software or app store. The purpose of signing application packages is to verify that the package has not been tampered with or compromised between the time you uploaded your application and when the package has been installed on a client device. It's important to note that although signature checks provide clear assurances that your player has not been tampered with since you signed it, it does not prevent the tampering or reverse engineering of your player. Some platforms strictly forbid installation of software packages outside the platform's software store, minimizing the risk of loading a compromised player.

The certificates that are used to sign your media player should be protected with limited access, ideally only within an automated build pipeline that's out of reach of any human operator, and should never be shared outside your organization. Use a private certificate authority, such as AWS Certificate Manager Private Certificate Authority (ACM Private CA), that provides strict access control to authorized issuers that will perform issuance of signing certificates. The pipeline that will build and sign your media player package should strictly control the process such that only authorized code will be built and signed.

Incident response

Even with mature preventive and detective controls in place, your organization should still put processes in place to respond to and mitigate security incidents. Logging, event processing, clean rooms, and game days are effective techniques to help you protect your content within your workload, but many organizations continue to protect content rights after delivery. In media applications where you are designing system to carry high value, copyrighted material, you should have a response plan that also includes illegal re-distribution of your content and intellectual property.

License holders and distributors often work with a third-party forensic security firm to embed and trace watermarks to the point of origin on a newsgroup, torrent, or piracy website. The report furnished can often be used to take legal action, which might include, but is not limited to, DMCA takedowns and monetary fines.

It is against the AWS Acceptable Use Policy for users to transmit, store, display, distribute, or otherwise make available content that is illegal, harmful, fraudulent, infringing, or offensive. If you suspect that your content is being illegally distributed using AWS services, your incident response plan should include steps to contact AWS Support and to follow our abuse reporting process.

Resources

Refer to the following resources to learn more about our best practices for security.

Documents

- AWS MPAA and Studio Compliance
- AWS Well-Architected Framework Security Pillar
- Secure Content Delivery with Amazon CloudFront
- Serving Private Content Using Amazon CloudFront and AWS Lambda@Edge

- Protecting your video stream with Amazon CloudFront and serverless technologies
- Using Amazon CloudFront and AWS Media Services
- Analyze Your CloudFront Logs at Scale
- Secure Packager and Encoder Key Exchange (SPEKE) Reference Server

Videos

- Secure Media Streaming and Delivery

Reliability pillar

The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This includes the ability to operate and test the workload through its total lifecycle.

Best practices

There are four best practice areas for reliability in the cloud:

- Foundations
- Workload architecture
- Change management
- Failure management

The probability of workload failure increases as the number of components in your system increases. Media industry people generally agree with AWS CTO Werner Vogels' often-quoted statement that "Failures are a given and everything will eventually fail over time.... This is a given, whether you are using the highest-quality hardware or lowest cost components", thus we build our workloads to operate despite single points of failure. Your objective is to maintain the viewing experience when failure or degradation does occur. As we've seen from the scenarios, the delivery of high-quality streaming media requires a considerable amount of complexity that often crosses business stakeholders. Keep a glass-to-glass perspective, from content production to delivery, and consider how any single component failure could impact the quality of playback experience.

In this section, we provide best practices that you can use to evaluate and improve the reliability of your streaming media services.

Foundations

See the AWS Well-Architected Framework whitepaper for best practices in the **Foundations** area for Reliability within streaming media applications.

Workload architecture

SM_REL1: How does your streaming infrastructure withstand failures in ingest, processing, origination, or delivery components?
SM_RBP1 – Document all workload dependencies and expected viewer experience in the event of component failure
SM_RBP2 – Design live streaming ingest architecture to withstand source failure by ingesting redundant video signals that take diverse network paths to AWS
SM_RBP3 – Design live streaming workflow to withstand individual processing and originati on failures by implementing redundant video pipelines

Closely examine both hard and soft service dependencies to ensure that failure conditions are well understood. Engage directly with service partners to understand failure conditions *before* issues arise. If problems do occur, notify the end user and protect their experience by offering alternate content. Use postmortems to learn from experiences and develop action plans.

Examples of hard dependencies in a streaming media workload include:

- Signal contribution path (Live)
- Media processing (Live)
- Media origin
- Digital rights management (DRM) and authentication

Examples of soft dependencies in a streaming media workload include:

- Content management systems
- Ad insertion and splicing
- Analytics

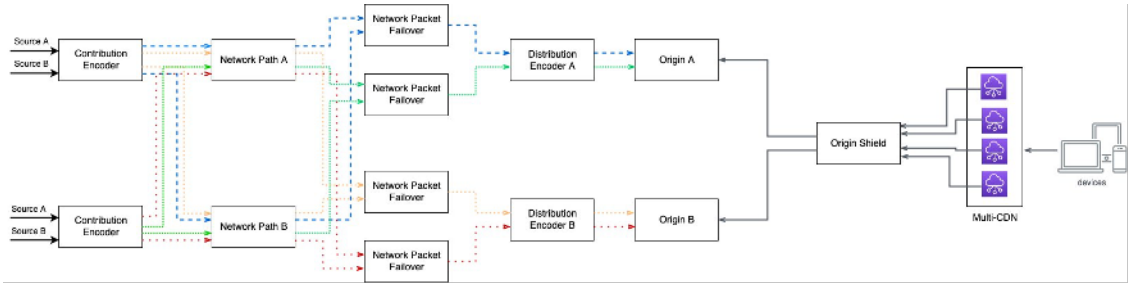
- Content Delivery Network (load dependent)

For example, if your ad-insertion platform experiences an availability outage, systems should fall back to an underlying origin source or have a process in place to remove ad insertion. This might decrease the revenue earned in the short term, but would retain service up-time and your audience satisfaction.

Every business will have to trade off the cost of reliably streaming content with any failure conditions that arise. We recommend reviewing the Well-Architected Reliability Pillar whitepaper to help you calculate your reliability target and the hard and soft dependencies of your workload.

Live resilient design

To achieve a highly available media streaming workflow, it is important to design for redundancy in every component of the chain. Let's consider the components in a live workflow and the network paths between them:



End-to-end redundant Live workflow

A failure in either the video signal or the network path that it takes to reach AWS Cloud impacts the entire workload and subsequently the end customer experience. Design your live video ingest architecture to withstand individual source failure by ingesting redundant video signals that take diverse network paths to AWS.

For example, in the preceding architecture, Source A and Source B are redundant input mezzanine sources. The contribution encoder is designed to fail over between the redundant sources in case of signal loss. To protect against failure of the contribution encoders, ensure that there are redundant contribution encoders in different physical on-premises locations. Each contribution encoder outputs two sets of contribution feeds, each with a binary identical SMPTE 2022-7 compliant network packet streams (represented by the same color arrow lines). This allows for transmission over separate network routes so if packets from one route are lost, the data can be reconstructed using packets from the second stream (as depicted by the Network Packet Failover component).

AWS Direct Connect can be used to provide dedicated network paths and **AWS Elemental MediaConnect** Flows can be used to reliably transport the feeds and provide failover at the network packet level compliant to SMPTE 2022-7 specifications. This design provides for full ingest redundancy across source signals and network paths.

Distribution encoding can be impacted by infrastructure issues, degradation of a dependent source, or by factors locally within the component. To achieve distribution encoder pipeline redundancy, ensure that the input source is being processed in at least two redundant locations within a Region. In the preceding architecture diagram, a distribution encoder in each Region is receiving two redundant input sources and processing them in separate AZs. Consider replicating the processing pipeline into an additional Region if your reliability targets warrant it.

With **AWS Elemental MediaLive**, a standard channel creates two redundant encoding pipelines (one in each AZ) with the option to provide redundant input sources with configurable failure scenarios. This allows you to architect a workload that can seamlessly fail between inputs while maintaining the integrity of the stream being published to the origin. By providing embedded timecode in your sources, you can prevent input failures from impacting the viewer experience through the MediaLive pipeline locking feature. If the input to MediaLive does not have valid timecode, the channel still remains highly available but without seamless failover.

It's a best practice to deploy redundant origin services in Multi-AZ or multi-Region and in case of an origin failure, reroute affected traffic. You can monitor origin health metrics and make real-time traffic routing decisions through DNS-based failover or CDN health checks. Alternatively, you can present all origin options to the player within the ABR manifest and implement client conditions for switching. In addition to the full outage failures, it is also important to protect against transient failures like high request latencies or timeouts.

For availability, performance, and geographic coverage reasons, it is common to deliver content using multiple CDNs. Doing so helps in distributing the traffic based on geolocation and available capacity. It also protects against failure or over-subscription in one or more CDN PoPs. It is recommended to collect near-real time QoS data (error rates, buffer rates, latency, etc.) from CDNs to determine the best delivery path for your customers and award traffic to best performing CDN. You may also load balance across CDNs based on other business considerations like cost.

VOD resilient design

As described in the scenarios section, VOD processing typically uses a serverless state machine comprised of event sources, messaging services, and subscribers to perform various operations. These operations should always be idempotent and designed so that when operations receive

messages more than once or run multiple times, they do not negatively impact the state of the workload. Dead-letter queues and distributed tracing services like AWS X-Ray can help you identify problematic messages or functions in your workflows as you scale.

The non-real-time nature of VOD provides you with the flexibility to decouple the batch Ingest and Processing components from Delivery and Playback. Thus, there are two approaches for reliable VOD design:

VOD origin reliability — In this scenario, your business objectives require that viewers can play content in the event of a workload failure, but allow for an interruption in the ability to publish new content to your platform. This is typical for platforms that publish a relatively small amount of new content on a daily or weekly basis. After content is ingested and processed, it's published and redundantly copied to multiple origin services. Technologies such as **Amazon S3 Cross-Region Replication (CRR)** can automate this function.

Once content is securely stored in multiple delivery endpoints, the CDN or client device can attempt playback from an alternate origin if playback from the primary endpoint fails. This architecture necessitates that the key delivery, authentication, content management, and other application layer services be reachable in the event of a failure.

VOD processing and origin reliability — In this scenario, the full application functionality must remain available in the event of an interruption. This includes the ability to ingest and process new content. This is achieved through a multi-Region design where the streaming architecture is replicated across two AWS Regions and CDNs, client logic, and DNS is used to route requests between Regions. In this scenario, care must be taken when designing the underlying storage and persistence layers (for example, databases and caching) to ensure consistency between Regions.

Change management

In a traditional streaming media environment, major changes are performed infrequently during maintenance windows with manual processes. This is high-risk as the people, processes, and tools used to update infrastructure are infrequently used. This can lead to drift in documentation and institutional knowledge of the workload.

We encourage you to automate deployments, testing, and rollbacks using services like AWS CloudFormation. This enables teams to make small, frequent changes on a regular basis and ensure that the infrastructure state is represented in code, managed in version control, and that your processes used for change management are well tested. To make troubleshooting issues easier, we also recommend creating a consistent naming convention for the components that make up the workflow, for instance, name the component with identifiers for asset, service Region, and AZ.

SM_REL2: How does your streaming media workload adapt to viewer demand?

SM_RBP4 – Use a CDN and plan capacity with your providers

SM_RBP5 – Design your origin service to automatically scale to meet viewer demand

The relationship between the client requests, delivery caching, and origin scaling is the most important area to examine when scaling your streaming media workload. Ingest and Processing components are scaled out in advance or run in batch before being made available for viewers and demand has no impact on these components.

A Content Delivery Network (CDN) is necessary to scale infrastructure for streaming media. CDNs provide multiple benefits by reducing the load on backend origination services, improving end-user performance, and lowering cost. By caching requests at the edge and within the CDN network, viewers are served directly from caches nearest them and requests to your origin server are dramatically reduced.

When considering which CDN to use, use the CDN's network infrastructure presence in the geographical areas where your viewers are located. While most CDNs offer coverage for viewers in the US and Europe, platforms with large number of viewers in Asia, Africa, or Latin America should be especially cognizant of the points of presence and network capacity of their CDN in those Regions and even consider a multi-CDN approach.

To achieve petabyte delivery scale, improve performance, or respond to intermittent delivery network issues, streaming media architectures might consider a multi-CDN delivery strategy. With multiple CDNs, you award or weight traffic to specific distribution networks based on current performance for a specific user or geographic Region – providing optimal viewing experience. Before you take this approach, consider the following trade-offs when compared to a single CDN approach:

- **Increased Origin Load** — With multiple CDNs, you will have more caches to populate with content. This will result in a lower Cache-Hit-Ratio and increase the load to origination services. Some of this load can be offset through an origin shield component.
- **Increased Cost** — Many CDNs offer tiered pricing based on utilization. By using multiple CDNs, you might not have access to lower pricing tiers.
- **Operational Overhead** — Deployment, testing, and the operation of multiple CDNs adds operational overhead.

- **Lack of Feature Parity** — Implementation might be hindered by the lack of feature parity across CDNs. This situation could introduce new requirements for your infrastructure and even reduce performance.

One approach to multi-CDN uses DNS resolution to apply a weighted round-robin distribution across CDNs. This is common for organizations looking to distribute load to meet capacity requirements, to meet cost commitments, or to minimize blast radius of CDN outages. This can be accomplished using **Amazon Route 53** by defining multiple record sets with a "Weighted" routing policy and the desired weight.

A DNS-based configuration is easy to implement and can be changed periodically to reflect network conditions, but DNS changes don't propagate quickly and some systems do not honor TTL values. For live events, where client playback buffers are small and network degradation can cause immediate impact, we recommend a dynamic HTTP-based approach that specifies a CDN during initialization of a playback session.

Adaptive Bitrate (ABR) protocols function by serving a manifest with pointers to media segments representing the elementary streams of audio and video data. With HTTP-based evaluation and routing, client content requests are served playback manifests that reference objects hosted by one or more CDNs. By serving customized manifests, you can target specific devices, geographies, or ISPs with prescriptive CDN. You can maintain CDN redundancy by providing alternative content URIs within the manifest, weighting them, and implementing failover logic within the player.

Regardless of approach, using multiple CDNs will increase load on origin services because each CDN will have its own caching network and requests to satisfy. To minimize the number of requests hitting origin services directly, you should optimize the content TTL, enable each layer of caching available, and use an origin shield service that can collapse requests from multiple CDNs into a single request to your origin layer.

As client requests come through the CDN, the origin layer must elastically respond to meet viewership demand. Implement your origin service within Auto Scaling groups, across multiple Availability Zones, and behind Application Load Balancers to ensure high availability. To determine the additional demand back to the origin, refer to the Performance Pillar in this paper to estimate load and inform scaling needs.

For best practices in change management, refer to the Reliability Pillar whitepaper.

Failure management

SM_REL3: How does your streaming infrastructure respond to or heal from failures in origination or delivery components?

SM_RBP6 – Design your streaming media workflow to automatically distribute traffic to redundant origins

SM_RBP7 – Monitor live streaming manifests for expected segment update patterns and to alert on staleness

The origin component exposes content endpoints and is at the heart of the streaming media workload. Managing failure must start with an evaluation of the failure scenarios that can be introduced at the origin component. An origin can fail to serve content due to issues with upstream dependencies or due to internal service failure cases. The most effective way of managing upstream failures is to introduce component-level redundancy so that upstream failures do not impact the health of the origin. Internal service failures should be managed by routing traffic to alternate origin resources through manifest re-writes, CDN origin traffic redirection, or client-side heuristics.

When managing an origin failure event during a live stream, you either introduce a discontinuity in the stream or design for seamless failover. In either case, the goal is to continue playback with minimal impact to the viewer. To achieve seamless playback, redundant packagers (encoder or origin) must serve content that is segment aligned. This means that all streams must present content that is aligned across segment boundaries, sequencing, and media properties (PTS). This is possible when redundant packagers are time synchronized. If this is not supported by your packager or you are unable to synchronize the packagers, segments won't contain the same content and the player might need to reset the decoder to continue playback during failover. Some players can handle this gracefully without input from the user, while others require a player reset in order to continue. Failure logic implemented in the player should always strive to continue playback in the event of a discontinuity without interaction from the user.

There are many ways to implement failover depending on your architecture and business requirements. One common approach, often used for video-on-demand assets and other static assets, is to use origin failover logic provided by a CDN. Origin health check implementations are transparent to the client, easy to set up, and typically work by redirecting traffic to alternate origins when requests from primary origin respond with failure codes or take too long.

CDN origin failover heuristics may not be sufficient for live streaming because frequent manifest updates are made as the stream progresses and we also need to monitor and trigger failovers based on the health of these updates. If a live stream manifest returned back from the origin does not advance in the expected real-time cadence players will re-buffer or halt playback completely, compromising playback experience. In addition to 4xx, 5xx errors, and high response latencies, you should design origin monitoring to alert on stale live manifests. The heuristics used to detect manifest staleness will depend on the configuration of your stream segment size and the client playback buffer size. For example, you might consider rerouting requests to healthy origins if the manifest remains unchanged for 2x – 5x segment size (4 – 10 seconds for a 2-second segment). When using **AWS Elemental MediaStore** as your live streaming origin, you can configure a Transient Data Policy on your container via the Object Lifecycle Policy API. This feature will remove an HLS manifest from the origin if it has not been recently updated enabling players to automatically switch from a primary origin to a backup origin.

When a stale manifest is detected, use client-side or edge logic to introduce origin failover. Many players allow for the configuration of alternate playback sources by explicitly providing multiple endpoints or by providing a manifest with alternate renditions. Always work closely with your player provider to determine the appropriate architecture for client-side failover implementation and incorporate test cases that simulate the common failure cases. If additional logic is required to route requests dynamically in response to failures, Lambda@Edge can be used to manipulate manifest responses to the player.

When failover does occur, consider the implications of sticky or non-sticky playback session handling. With sticky failover clients are pinned to the new origin endpoint during failover and only switch again if there is an additional failure. With non-sticky failover clients access content from the primary origin anytime it becomes available. You should always use a sticky design when implementing a non-seamless failover design to prevent origin switches that could adversely impact clients.

Resources

Refer to the following resources to learn more about our best practices related to reliability.

Documents

- Blog Series: Resilient End-to-End Live Workflow Using AWS Elemental Services
- Amazon CloudFront for Media
- Using Amazon CloudFront Origin Shield in a Multi-CDN Architecture

- Seamless regional failover capabilities with Clustered Video Streams Reference Architecture

Videos

- Raising the Bar on Video Streaming Quality Using AWS
- Designing for Live Stream Failure with Seamless Switching

Performance efficiency pillar

The performance efficiency pillar focuses on the efficient use of computing resources to meet requirements and maintaining that efficiency as demand changes and technologies evolve.

Best practices

There are four best practice areas for Performance Efficiency in the cloud: selection, review, monitoring, and tradeoffs.

- Selection
- Review
- Monitoring
- Tradeoffs

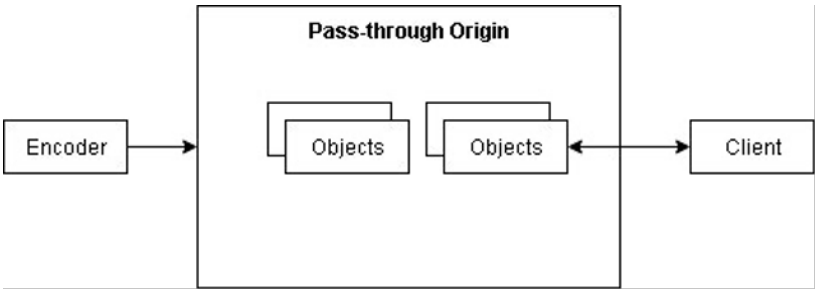
Viewers expect low latency, high quality, consistent viewing experiences that load immediately. In this section, we review the key selection criteria for transport and origination of media to help you choose the right tools for your architecture. We also provide a set of monitoring metrics that, if implemented, will help you baseline your expected performance, detect any deviance, and take actions to improve your workload over time. Finally, we examine common architecture tradeoffs to improve performance, like reducing segment size to reduce live latency at the expense of request overhead or increasing bitrate to improve video quality at the expense of bandwidth costs.

Selection

SM_PERF1: How do you optimize media delivery through media origination and processing?
SM_PBP1 – Select an appropriate origin technology for your workload

There are two origin approaches that can be used to optimize performance of your workload — pass-through and dynamic. Pass-through origin servers are high-performance HTTP servers that are adequately scaled to respond to requests for content segments from viewers or CDNs. Encoders PUT pre-encoded, pre-packaged, and pre-encrypted content to the origin server. Pass-through origin servers, like AWS Elemental MediaStore or Amazon S3, offer limited media processing functionality like repackaging, but offer high performance and low cost.

When using a pass-through origin, the media processing layer is responsible for the media codec and transport protocol formatting. For example, the encoder configuration for HLS segment size will directly impact the end-to-end latency of a live stream. To support multi-protocol environments like MPEG DASH and Apple HLS with a single set of processed media on a passthrough origin, the industry is moving toward the Common Media Application Format (CMAF). We encourage you to use this format if you are planning to use a pass-through origin (if your player devices support it) as it reduces storage costs, improves CDN cache efficiency, and can decrease latency through chunked-transfer-encoding.



Comparing pass-through origin to dynamic origin

With a dynamic origin, content is PUT to the origin by an encoder in a single format and dynamically formatted as requested by viewers. This process, often referred to as just-in-time packaging (JITP), provides additional flexibility as content endpoints can be customized to support business objectives without modifying the content produced by the encoder. This is especially powerful for multi-protocol or multi-DRM workflows that address a fractured device ecosystem with variable requirements. Additional features enabled by dynamic origin services like AWS Elemental MediaPackage include look-back, start-over, and live-to-VOD recording. Note that because dynamic origins require an internal buffer to process content just-in-time, you will trade off live stream latency when compared to pass-through origins.

Dynamic origins provide you with the flexibility to customize endpoints to support your target devices. For example, you can create an endpoint optimized for connected television experiences that accounts for in-home network connections and large screens by creating a rendition set with high-resolution content and large segment sizes. Alternatively, using the same content, you could

create an endpoint optimized for mobile networks, screens, and interactivity with shorter segment sizes and additional low-resolution options.

The decision to use a pass-through origin or a dynamic origin depends on your unique business requirements. In general, use a pass-through origin if you have a limited number of client compatibility requirements, a desire for the lowest possible live latency, and value simplicity over flexibility. If you have a diverse client ecosystem and require advanced origin features like live-to-VOD recording, you should use a dynamic origin.

SM_PERF2: Hw do you approach media source contribution?
SM_PBP2 – Begin with the highest-quality sources that you can reasonably acquire
SM_PBP3 – Use specialized media transport and acceleration protocols
SM_PBP4 – Use private network connectivity between your content provider and media ingest

There are two ingest patterns for video applications, *real-time* and *file-based*. Real-time workloads often include a business requirement for reliability and low latency, while file-based transfers for mezzanine files or archive workloads generally prioritize efficiency of data movement.

Real-time contribution

It's difficult to consistently deliver live video content to large audiences quickly, reliably, and cost-effectively. Getting a live source from an event site, often called a *contribution feed*, is the first inflection point when deciding trade-offs that will eventually impact end users. In all cases, use of an efficient codec (for example, HEVC) for signal contribution allows for optimum bandwidth utilization and improved quality at lower cost.

Any impact to the network during the live contribution of media can quickly degrade and disrupt video distribution to end users**. AWS Direct Connect**, in a maximum resiliency configuration, is recommended for services requiring a dedicated, sustained connection to AWS (that is, high visibility live events) as it provides a consistent network experience when compared to internet transport. For first-mile connectivity at remote live events that are subject to unmanaged networks for delivering live media, we strongly recommend using a combination of mediums, such as bonded-cellular, satellite uplink, and multiple ISPs connecting to the AWS network.

Many applications use TCP-based protocols like RTMP to deliver contribution feeds, but these protocols are inefficient and can introduce unnecessary latency. We recommend a UDP-based reliable transport protocol like RTP, Reliable Internet Stream Transport (RIST), or Secure Reliable Transport (SRT) for real-time contribution. These protocols are optimized for low-latency, efficient video transport, and include tunable error correction schemes like Automatic Repeat Request (ARQ) and Forward Error Correction (FEC). In general, ARQ-based protocols are preferred for unmanaged, public networks where loss is non-deterministic and FEC is useful for semi-managed networks where loss is deterministic.

File-based contribution

Asset management, archive, video-on-demand, and many other media workloads manage large file transfers. AWS provides a portfolio of services designed for these specific needs. These are our general recommendations:

- As with live workloads, sustained file upload architectures should use AWS Direct Connect.
- Always use **Amazon S3 Multipart Upload** when uploading content to **Amazon S3**.
- When uploading content from a location geographically distant from an AWS Region, use **Amazon S3 Transfer Acceleration** to leverage CloudFront edge PoPs for connectivity into AWS.
- When upload duration is expected to exceed more than one week, use **AWS Snowball Edge.**
- For hybrid architectures where on-premises infrastructure requires **Amazon S3** access as NFS mounts or a virtual tape library interface to Amazon S3 Glacier, use **AWS Storage Gateway** and **AWS File Gateway.**

Due to the large file sizes of mezzanine quality media assets, video workloads should employ tiered storage balancing performance and cost. S3 Lifecycle policies can be used to programmatically move infrequently accessed S3 assets to Amazon S3 IA or Amazon S3 Glacier, however, this applies only to object storage and is a one-way function. Consider designing an intelligent filer service to restore source assets into the appropriate storage service (block, file, or object) when required by a processing job. If asset references exist use these manifests to optimize restores.

Review and monitoring

SM_PERF3: How do you use caching to improve content delivery performance?
SM_PBP5 – Use a content delivery network and monitor your cache-hit-ratio

SM_PERF3: How do you use caching to improve content delivery performance?

SM_PBP6 – Ensure that cache-control headers for your content are optimized

SM_PBP7 – Have a cache invalidation runbook

SM_PBP8 – Minimize negative (error) caching

A Content Delivery Network (CDN) scales video delivery by serving content from local caches nearest the user and providing optimized routes to origination services. Caching improves time-to-first-byte for clients and reduces the load on origin services. CDNs use a multi-tier architecture with two or three tiers of cache hierarchy before requests make it back to the origin server. These tiers are usually referred to as the edge tier and mid-tier caches. The edge tier is first to receive a request from client and responds fastest in the case of a cache match. The mid-tier has a larger cache depth, but is located only in select locations. Cache misses to the edge tier come back to the mid-tier for another chance at a cache match.

A Cache Hit Ratio (CHR) is the ratio of requests served from cache (matches) to the total requests (misses and matches) over a period of time. Cache matches improve the client experience and cache misses result in a request directly to your origin layer that increases response latency and costs. Monitoring CHR will help you to improve delivery and origin layer performance over time. You can enable CHR, origin latency, and HTTP error rate metrics from your Amazon CloudFront monitoring settings.

CDNs typically employ a last recently used (LRU) caching strategy on each tier. This means that data will be maintained in caches based on the amount of traffic an object receives and the available cache size. Though you can't guarantee caches will hold content for the next request, you can set Cache-Control headers on the origin to indicate the preferred duration for an object to be kept in a CDN cache. Your CDN should be configured to respect caching headers from your origin server to ensure that live content and manifests are only cached for the appropriate amount of time.

Live streaming manifests are frequently updated to represent the next media object in the stream and should not be cached for longer than half of your segment duration. Caching longer than the segment duration could result in the serving of stale manifests, a delay for clients to retrieve the next media segment, and client buffer exhaustion, which will negatively impact user experience. Live media segments and VOD content (both segments and manifests) should be cached for as long as possible to retain them in delivery caches for the maximum amount of time.

Scenario	Segment Size	Manifest Update Frequency	Segment Cache-Control Header or Cache Behavior	Manifest Cache-Control Header or Cache Behavior
Live	10 seconds	10 seconds	21,600 seconds or max DVR window	5 seconds or less
VOD	10 seconds	Static	86,400 seconds or longest possible	86,400 seconds or longest possible

Recommended cache behaviors for live and Video-on-Demand (VOD) scenarios

There are often times when cached content needs to be modified or invalidated. Have a cache invalidation runbook in place so that you can modify cached objects and invalidate the previous content. This can be achieved by invalidating content with a CDN feature, using variable file names, or using query string parameters to "break" the cache when content is changed.

Caching of error responses from the origin, also known as *negative caching*, should be minimized as some streaming clients might proactively request future segments before they are published to minimize latency. For live streaming, it should be disabled completely for manifest and segment files. At a minimum, the negative caching duration should not exceed one segment length. Amazon CloudFront caches origin errors for five minutes by default, but you can configure it to suit your needs.

SM_PERF4: How do you monitor viewer experience?
SM_PBP9 – Collect and analyze real user logs and metrics
SM_PBP10 – Recognize and respond to playback anomalies

Infrastructure logging and monitoring only provides you with part of the picture. We recommend that you design a client that sends real-user data directly to monitoring and logging systems. This allows you to benchmark normal behavior, identify anomalies, and correlate events with content delivery systems. For example, session initialization information, like playback URL, user-agent, and

network connection status could help you identify issues with a specific origin, client device type, or network environment.

For streaming media, it's especially important to monitor the health of the video decoder to determine how changes in network topology, video encoding settings, or mobile operating systems impact the end user experience. For example, capturing video buffering events, which directly impact customer satisfaction, should be a key indicator of streaming health.

We recommend that you capture client metrics from streaming sessions with services like **Amazon Kinesis** and monitor for anomalies with **Amazon CloudWatch**. Equipped with this data, you can uncover patterns from real users, create alerting systems, and automate remediation tasks. The **AWS Partner Network** provides another avenue for video-specific monitoring tools that can give you actionable data from playback sessions.

Amazon Prime Video, a streaming media service by Amazon, has many ways of monitoring customer experience. One key metric, *Zero-Impact-Rate*, measures the rate of streaming sessions that have had any buffers or errors. This is used to baseline customer experience and alert when there are deviations from normal behavior. Here are other client metrics that provide valuable insights into viewer playback experience:

Metric	Description
Time-to-First-Frame	Time between client request for content and first frame being displayed on client
Playback Frames Per Second	Client displayed frame rate
Session Resolution	Client displayed resolution
Session Duration	Duration the client spent watching content
Buffering Events	Client buffering events
Zero-Buffer-Rate	Number of sessions that had zero buffer events
Client Errors Events	Client HTTP or application errors
Zero-Error-Rate	Percentage of total sessions that had zero error events

Metric	Description
Zero-Impact-Rate	Percentage of total sessions that had zero Buffers or Errors

Suggested metrics for measuring quality of service

Tradeoffs

SM_PERF5: What tradeoffs have you made in media processing to improve client experience and lower bandwidth costs?
SM_PBP11 – Optimize the number of adaptive bitrate renditions for your workload
SM_PBP12 – Select appropriate encoding settings for your content type and quality targets
SM_PBP13 – Trade higher content processing cost for lower delivery costs for popular content

For media delivery applications, protocol selection and configuration have a dramatic impact on client performance. Progressive file downloads or RTMP streaming, can be slow to download, costly to scale, or inflexible. Instead, use HTTP-based Adaptive-Bitrate protocols, like Apple HLS, combined with web caching mechanisms to improve distribution efficiency. These protocols also enable clients to select the optimal rendition for playback based on network connection, display resolution, and other client-side characteristics, greatly improving viewer experience.

The *ABR ladder*, or number of logical renditions available to an individual client, should be tuned to meet the needs of the specific application, taking into account:

- Playback quality
- Client and display ecosystem
- User geography and network connectivity

Providing too many renditions can increase encoding costs and cause clients to fluctuate frequently between renditions, reducing perceived quality. Not providing enough renditions might leave usable bandwidth underutilized and, thus, also reduce quality. Our general recommendation

is to determine the maximum bitrate first, then divide by 1.5–2 for each step down. This step in the ladder allows clients to make significant jumps in quality, but not so many that frequent changes could be perceived by end users. If your application is delivering to Apple devices, refer to Apple TN2224 for additional guidance on creating adaptive bitrate content. If you are using AWS Elemental MediaConvert, the *Auto ABR* capability can automatically determine an optimal ABR ladder for you based on the specific characteristics of the content.

SM_PERF6: What tradeoffs have you made to lower live glass-to-glass latency?

SM_PBP14 – Optimize processing, origination, delivery, and client for low latency

SM_PBP15 – Remove unnecessary processing stages

Latency is inherent in any broadcasting or streaming platform. Over-the-air live broadcast glass-to-glass latency ranges between 3 – 12 seconds, with an average of 6 seconds often seen in practice. This means that it could take up to an average of 6 seconds before the event that is captured in the camera is displayed on the playback device. For streaming platforms, this latency can vary anywhere from 3 to 90 seconds, depending on various design choices. Typically, achieving low latency in a streaming platform may be a tradeoff with other critical aspects of the streaming experience, such as, video quality, re-buffering rate, error rates and other quality-of-service indicators.

With HTTP-based streaming, latency mainly depends on the media segment length. For instance, the Apple HLS specification recommends at least three segments of buffer for best performance. This directly influences the latency. Other factors in the media delivery pipeline that influence latency include the video encoding operations, the duration of ingest and packaging operations, network propagation delays, and the CDN. In most cases, the player buffer carries the largest share of the overall latency.

There are several tradeoffs to consider with low latency media streaming design. Shorter media segment lengths will result in increased traffic on the caching servers and then to the origin. This is fairly manageable by a CDN, especially if it supports HTTP 2.0 at the edge and HTTP 1.1 origins. As previously mentioned, encoding parameters have an impact on latency and optimizations for latency typically impact the video quality. For example, setting an encoder *lookahead* size to a low value will improve latency, but reduces output quality for demanding scene changes. If your content does not have dramatic scene changes, keeping this value low will not have a noticeable impact video quality.

Resources

Refer to the following resources to learn more about our best practices related to performance efficiency.

Documents

- HLS Authoring Specification for Apple Devices
- How to Compete with Broadcast Latency Using Current Adaptive Bitrate Technologies
- Managing How Long Content Stays in an Edge Cache
- Increasing the Proportion of Requests that Are Served from CloudFront Edge Caches (Cache Hit Ratio)
- Delivering Live Streaming Video with CloudFront and AWS Media Services
- Streaming Media Analytics Solution

Videos

- Architecting a 24x7 Live Linear Broadcast for 100% availability on AWS

Cost optimization pillar

The cost optimization pillar includes the continual process of refinement and improvement of a workload over its entire lifecycle. From the initial design of your first proof of concept to the ongoing operation of production workloads, adopting the practices in this paper will enable you to build and operate cost-aware systems that achieve business outcomes and minimize costs, thus allowing your business to maximize its return on investment.

Best practices

There are five best practice areas for cost optimization in the cloud:

- Practice Cloud Financial Management
- Expenditure and usage awareness
- Cost-effective resources
- Manage demand and supply resources
- Optimizing over time

As with the other pillars, there are tradeoffs to consider. For example, do you want to optimize for video quality or storage and bandwidth costs? In some cases, it's best to optimize for quality —getting valuable content to viewers in the highest possible resolution—rather than storage and delivery cost.

Knowing your content value will guide design decisions and help you select reliability targets, content protection schemes, and performance objectives. You can *always* adjust your focus between the pillar areas, but a shared understanding of the content value will help your organization align on design objectives and make tradeoff decisions. This will ensure that you reach business objectives while controlling spending.

The following section provides techniques and guidance for the Cost-effective resources and Optimizing over time best practice areas for streaming media . Readers interested in the Manage demand and supply resources, expenditure and usage awareness, and Practice Cloud Financial Management areas should refer to the AWS Well-Architected Framework whitepaper for applicable best practices.

Cost-effective resources

Object storage is a key part of origin services or media archives. The 11 9's of durability and near-infinite scale reduce operational overhead and storage costs for media applications. As media assets can easily reach hundreds of gigabytes in size, you should develop a storage strategy that takes into account the access patterns of the content.

SM_COST1: What is your strategy for optimizing object storage and data transfer costs?

SM_CBP2 –Trace and limit data transfer between services, Availability Zones, Regions, and clients

We recommend that you prioritize a strategy for storage lifecycle and data transfer, as these often have the largest savings potential when compared to requests, management functions, and object replication.

With media archive or batch transcoding workloads, you should analyze and define access patterns. Once understood, represent these patterns in **Amazon S3 Lifecycle Policies**, which automate the continual movement of assets to the most effective storage. This achieves the lowest storage costs, while meeting the business accessibility requirements.

For applications with non-deterministic access patterns, such as video recordings, user-generated-content, or a large back-catalog, it's a challenge to design lifecycle policies or select outright the most cost-effective **Amazon S3** storage tier for your content. To optimize storage costs, asset access patterns need to be actively monitored and managed. This can be done manually through content management solutions or with features like **Amazon S3** Analytics Storage Class Analysis and S3 Intelligent-Tiering.

Real-time media streams are measured in megabytes per second and source files can easily reach into the hundreds of gigabytes per file. To control data transfer and outbound costs, be aware of the data transfer costs associated with transmitting the data throughout the cloud and to users. To control costs associated with inter-AZ or inter-Region transfer, you should ingest content into the Region and Availability Zone in which it will be processed. For outbound data cost control between the origin and client, consider the use **Amazon CloudFront** to reduce costs while improving performance for viewers. When content does need to span Availability Zones, Regions, or incur outbound costs, use the most efficient compression possible to minimize cost.

SM_COST2: What is your strategy for optimizing content processing costs?

SM_CBP3 – Baseline resources and throughput for media processing tasks

SM_CBP4 – Run media processing tasks in parallel

Consumer demand for more content choices at higher quality (resolution, color-depth, frame rate) puts pressure on processing components. As media processing needs increase for tasks like quality control (QC), transcoding, and packaging, so do the costs associated with these tasks.

To manage costs, baseline the time and infrastructure resources necessary for your most common processing tasks. One way to understand processing speed and resource density is to calculate the real-time factor (RTF) by taking task execution time divided by source duration. An RTF lower than one indicates that content is processing faster than real-time, greater than one is slower than real time. For example, a video transcoding task that takes an EC2 instance 45 minutes to process a one-hour video has an RTF of 0.75. RTF and the resources required can be used to understand and communicate how changes to the task or infrastructure will impact business priorities.

One way to reduce RTF while accessing cost effective resources is to split jobs into smaller subtasks and run them in parallel. With video content, you can split tasks on I-frame boundaries, process those segments, and then stitch them together when all subtasks are complete. Split-and-stitch and parallelization techniques create granular tasks that can easily be placed across a processing

fleet, leading to more efficient use of compute resource. Splitting processor intensive tasks also reduces dependency on vertical scaling and enables access the most cost effective compute resource available through heterogeneous **Amazon EC2 Spot** clusters.

When using **AWS Elemental Media Services**, many pricing options are available to make workloads more cost effective. For example, for live streaming workloads where redundancy is not required, **AWS Elemental MediaLive** offers channels that run in *single-pipeline* mode for a significant discount over the standard channel rates. Similarly, for long-running and 24x7 channels, channel reservations offer significant discounts in exchange for term commitments. For file-based workloads, **AWS Elemental MediaConvert** offers *basic tier* functionality where lower rates are available in exchange for avoiding certain computationally intensive or licensed features. When a large volume of steady-state file-based transcoding is required, reserved transcoding slots can also be purchased to provide steady-state transcoding capacity for a fixed monthly cost rather than the standard billing model of per output minute.

Optimizing over time

Content delivery is the largest cost-optimization area for organizations streaming content to large audiences. You should *always* be evaluating the perceptual quality of content and striving to lower video data rates while maintaining, or ideally improving quality through encoder optimization.

SM_COST3: How do you minimize distribution costs while maintaining visual quality?

SM_CBP5 – Use objective and subjective measurement techniques to benchmark and improve video compression efficiency

To optimize quality while balancing delivery costs, you should focus on media processing and compression. The codec you use will be dependent on your content complexity, encoder capabilities, and client interoperability. Most codecs provide many controls that can be tuned. Use a combination of objective and subjective measurement to understand and improve your compression efficiency over time.

Though not always an accurate representation of human visual perception, objective measurement tools, like Peak Signal-to-Noise-Ratio (PSNR), Structural Similarity (SSIM), and Video Multi-Method Assessment Fusion (VMAF) are readily available tools to help analyze your compression performance. Given a source, varied encoding job settings, and resulting outputs, compare these metrics to tune for your unique content. In general, though there are many options to consider

while tuning video compression settings, high motion content will require a higher data rate than low motion to retain the same perceptual quality.

Subjective measurement techniques will uncover practical issues with your video compression that might not be identified by objective scoring. For subjective testing with human eyes and real network conditions outside of a production environment, consider user feedback mechanisms that enable willing users to provide information on their viewing experience. For a simulated environment, **Amazon Mechanical Turk** can give you access to humans willing to watch your content and provide feedback. For a small fee, *Turkers* around the world can provide you with valuable insights on their playback experience, which can be used to improve your workload.

Use this combination of subjective, objective, and real-user-metrics to tune encoding settings and ABR protocol configuration. It is recommended that you familiarize yourself with the codec and encoder you use for delivery so that you can identify optimizations that could save you delivery costs. For example, context-aware or quality-based encoding, available as a QVBR setting for **AWS Elemental MediaLive** and **AWS Elemental MediaConvert**, can analyze the complexity of the content and optimize processing on your behalf.

Resources

Refer to the following resources to learn more about AWS best practices for cost optimization.

Documents

- Netflix VMAF Project

Videos

- Containerizing Video: The Next Generation Video Transcoding Pipeline
- Leverage the Power of the Crowd To Work with Amazon Mechanical Turk

Sustainability pillar

The sustainability pillar includes the ability to continually improve sustainability impacts by reducing energy consumption and increasing efficiency across all components of a workload by maximizing the benefits from the provisioned resources and minimizing the total resources required.

There are no sustainability practices unique to this lens. For information on Sustainability, refer to the Sustainability Pillar whitepaper.

Conclusion

The AWS Well-Architected Framework provides architectural best practices across the pillars for designing and operating reliable, secure, efficient, and cost-effective workloads in the cloud. The Video Delivery Lens provides additional insights that allow you to dive further into an existing or proposed workload. Using the framework in your architecture will help you produce stable and efficient systems, freeing you to focus on innovation that delivers more value to your end users.

Whether you've just started designing a greenfield video application on AWS or are looking to migrate an existing one, we hope that this paper has given you new perspective or sparked new ideas. We encourage you to take advantage of the recommendations offered here as well as the knowledge and experience of our AWS Solutions Architects. We'd love to hear from you – especially about your success stories building video applications on AWS. Contact your account team or use Contact Us via our website.

Document history and contributors

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

Change	Description	Date
Minor update	Corrected broken link.	April 6, 2023
Initial publication	Whitepaper first published.	September 29, 2021

Note

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

Contributors

Contributors to this document include:

- Bryan Samis — Sr. Solutions Architect, Amazon Web Services
- Avinash Khurana — Sr. Technical Account Manager, Amazon Web Services
- Gene Ting — Sr. Solutions Architect, Amazon Web Services
- Shawn Przybilla — Sr. Solutions Architect, Amazon Web Services
- Bruce Ross — Sr. Solutions Architect, Amazon Web Services

Glossary

Ad Decision Service

A service that determines how to fill an advertising placement opportunity.

Ad Decision Splicer

A service that dynamically inserts advertisements into media, typically in conjunction with an Ad Decision Service.

Adaptive Bitrate (ABR) streaming

A technique for delivering media to a client by providing a set of identical content renditions at varying qualities and allowing the client to select the most appropriate version based on bandwidth and decoder performance.

Apple HTTP Live Streaming (Apple HLS)

Released by Apple as an adaptive bitrate protocol primarily for iOS distribution, HTTP Live Streaming (HLS) has become the standard for transmitting material to client devices. Because it implements the MPEG Transport Stream container, the elementary stream packetization for audio, video, captions, and ad signaling is well understood and supported by many consuming applications. The primary challenge for HLS contribution is the delay it introduces. HLS segments are approximately 10 seconds in size and, because Apple recommends a buffer of three, it's common to experience video decoder buffers of 30 seconds or more.

Adobe Real-Time Messaging Protocol (Adobe RTMP)

Originally developed as a proprietary protocol by Macromedia, the Adobe Real-Time Messaging Protocol (RTMP) has been widely used over the past decade for progressive video delivery. In recent years, the popularity for RTMP has subsided for client distribution due to the proliferation of HTTP-based adaptive bitrate protocols. However, for media contribution, it's widely used due to its ubiquitous presence within open source and commercial live encoding applications. Though praised for wide adoption and relatively low latency, RTMP is not without challenges. Specifically,

proprietary and often ad-hoc implementations of caption and ad insertion data from in the ecosystem can create challenges.

AWS Global Accelerator

As a method to improve the global availability and performance of your application, the AWS Global Accelerator helps you build a secure, highly scalable, fault-tolerant and multi-Region application.

codec

A portmanteau of the words COder and DECoder, a codec is a standard algorithm for digital media encoding. Examples include: H.264, HEVC, AV1, and MPEG-2.

container

Also known as a *format* or *package*, a container is a standard for associating audio, video, and metadata into a file system structure. This can be as a single file or as a collection of multiple files, depending on the container type. Examples include: MP4, MPEG-2 TS, QuickTime (MOV), or WMV.

Content Delivery Network (CDN)

A distributed network of servers that delivers web content to end users.

content provider

A creator of media content that wants to reach an audience.

Digital Rights Management (DRM)

Services and functionality that enable content owners to determine who and how their content is accessed.

encoder

A service that converts digital video from one format to another, typically to facilitate distribution.

Real-time Transport Protocol (RTP)

Developed by the Internet Engineering Task Force (IETF), Real-time Transport Protocol (RTP) is a network protocol designed for delivery of audio and video over IP. Its basis on UDP (low latency), flexibility, and wide application for media workloads make RTP an ideal ingest protocol. We strongly recommend using Forward Error Correction in conjunction with RTP to account for network packet loss. Column and row depth configurations will be network dependent.

origin

A service hosting content.

packager

A service that reformats, multiplexes, or repackages media without re-encoding.

Quality Control (QC)

A process for ensuring that the output content is of sufficient quality.

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.