



AWS
Black Belt
Online Seminar

【AWS Black Belt Online Seminar】 Amazon Kinesis Video Streams



アマゾン ウェブ サービス ジャパン株式会社
ソリューションアーキテクト 山崎 翔太

2018.03.28

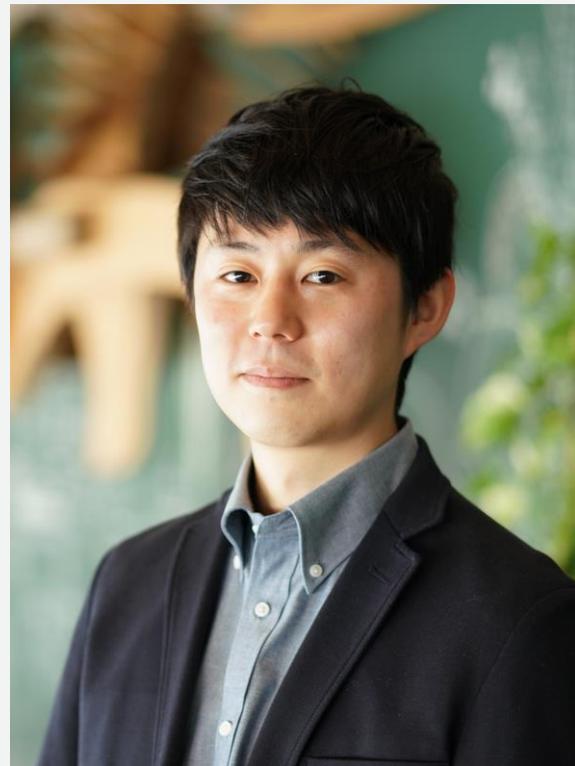
自己紹介

山崎 翔太

技術統括本部

ソリューションアーキテクト

- IoT や データ分析 や 機械学習 を活用して新しい仕組みを作ることが好きです
- 好きなサービス
 - Amazon Kinesis と AWS Lambda
 - そして Amazon Kinesis Video Streams



内容についての注意点

- 本資料では2018年3月28日時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(<http://aws.amazon.com>)にてご確認ください。
- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。
- 価格は税抜表記となっております。日本居住者のお客様が東京リージョンを使用する場合、別途消費税をご請求させていただきます。
- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at <http://aws.amazon.com/agreement/>. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

このセミナーの内容

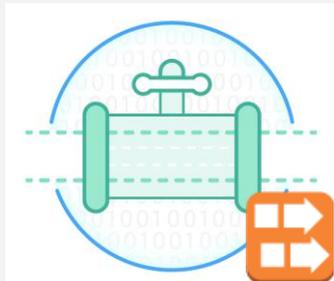
- 📦 Amazon Kinesis Video Streams の特徴
- 📦 Amazon Kinesis Video Streams を動かすまで
 1. 動画データのフォーマット
 2. プロデューサー (カメラ側)
 3. コンシューマー (動画解析側)
 4. デモ
- 📦 Amazon Kinesis Video Streams を活用した構成例
- 📦 まとめ

このセミナーの内容

- 📦 Amazon Kinesis Video Streams の特徴
- 📦 Amazon Kinesis Video Streams を動かすまで
 1. 動画データのフォーマット
 2. プロデューサー (カメラ側)
 3. コンシューマー (動画解析側)
 4. デモ
- 📦 Amazon Kinesis Video Streams を活用した構成例
- 📦 まとめ

Amazon Kinesis プラットフォーム (今まで)

ストリームデータを収集・処理するためのフルマネージドサービス群



Amazon Kinesis Streams

ストリームデータを
処理するための
アプリケーションを
独自に構築



Amazon Kinesis Firehose

ストリームデータを
S3、Redshift、
Elasticsearch Service、
Splunk へ配信

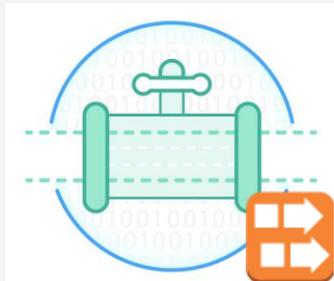


Amazon Kinesis Analytics

ストリームデータを
標準的な SQL クエリ
でリアルタイムに分析

Amazon Kinesis プラットフォーム *New*

ストリームデータを収集・処理するためのフルマネージドサービス群



Amazon Kinesis
Data Streams

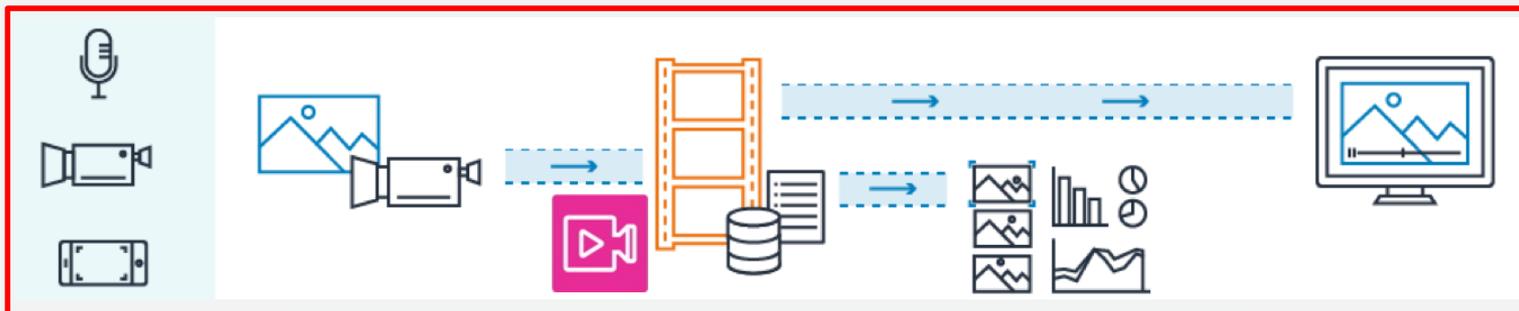


Amazon Kinesis
Data Firehose



Amazon Kinesis
Data Analytics

New



Amazon Kinesis **Video Streams**

Amazon Kinesis Video Streams

分析と機械学習のためにビデオストリームをキャプチャ、処理、保存する

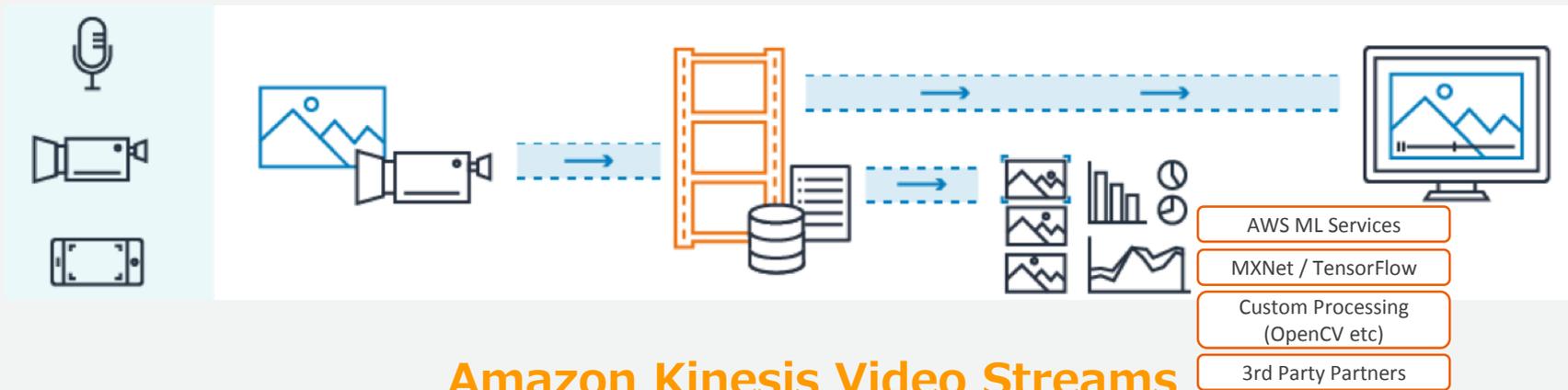
既に東京リージョンでの一般利用が可能！

2018年2月現在、以下のリージョンで一般利用が可能

- 米国東部（バージニア北部）
- 米国西部（オレゴン）
- EU（アイルランド）
- EU（フランクフルト）
- **アジアパシフィック（東京）**

Amazon Kinesis Video Streams

ビデオストリームをクラウドへ取り込み、
低遅延／オンデマンドで分析処理に配信するためのマネージドサービス



Amazon Kinesis Video Streams

デバイス **プロデューサー**

カメラ/マイクなど

動画の送信

ストリーム

動画の保存とインデックス化

コンシューマー

動画解析・加工・再配信／変換・再生

Kinesis Video Streams が扱えるデータ

動画以外の時系列データも扱うことが可能

動画データ

カメラから出力される動画データ

- スマートフォン
- セキュリティカメラ
- ウェブカメラ
- 車載カメラ
- ドローン

など

非動画データ

時系列にエンコードされたデータ

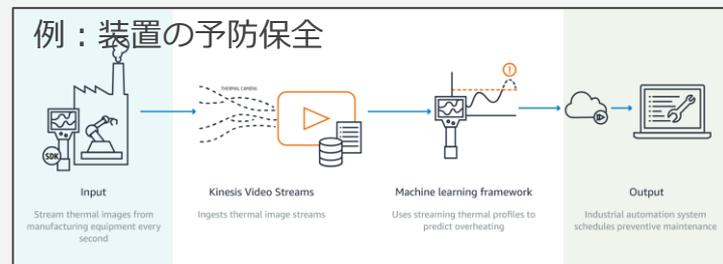
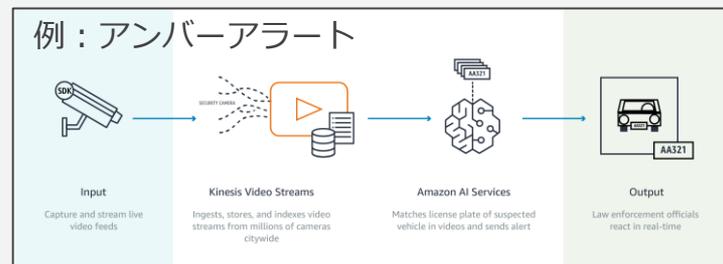
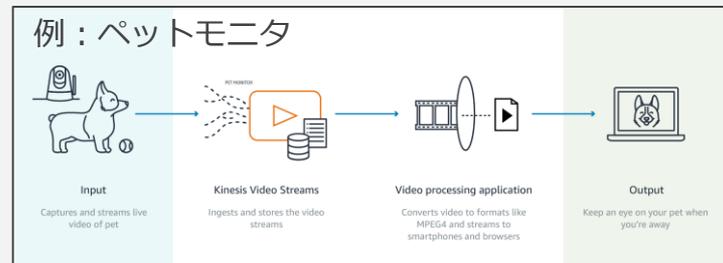
- 音声データ
- 熱画像
- 深度データ
- レーダー信号
- LiDAR信号 (<https://en.wikipedia.org/wiki/Lidar>)

など

※ データ構造については後述します

ユースケースの例

- スマートホーム
 - ベビーモニタ、ウェブカメラ、防犯システムといったカメラを装備した家庭用機器をAWSに接続
 - 遠隔監視、インテリジェント照明、温度制御システム、セキュリティソリューションなどを実現
- スマートシティ
 - 交通信号機、ショッピングモール、公共イベント会場、駐車場などに設置された多数のカメラをAWSに接続
 - 膨大な量の動画データを取り込み、交通問題の解決、犯罪の防止、緊急事態への対応といった用途に活用
- 産業オートメーション
 - 電波探知器、レーザー探知器の信号、温度、深度データといった産業用機器の時系列データをAWS上に収集
 - TensorFlow、OpenCV などの機械学習・画像解析を、異常検知や予防保全などの産業オートメーションに活用



なぜ Kinesis Video Streams が必要なのか

動画データを収集する際の課題

- カメラ台数の増加に耐えられるスケーラビリティが必要
- データを取り出しやすい形で保存することも必要
- 信頼性も可用性もセキュリティも必要
- これらを自分で作るには、手間もコストもかかる

動画データを分析する際の課題

- ストリーム処理では、信頼性やスケーラビリティのために、データを収集する部分と解析処理をする部分は分けたほうがよい
- そもそも、本当にやりたいことは解析処理なので、そこに集中したい

Kinesis Video Streams の特徴



何百万ものデバイスを接続してのストリーミング



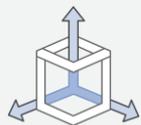
データの永続的な保存とインデックス化



動画解析には低遅延でのリアルタイム処理とバッチ処理
どちらのアプリケーションも構築可能

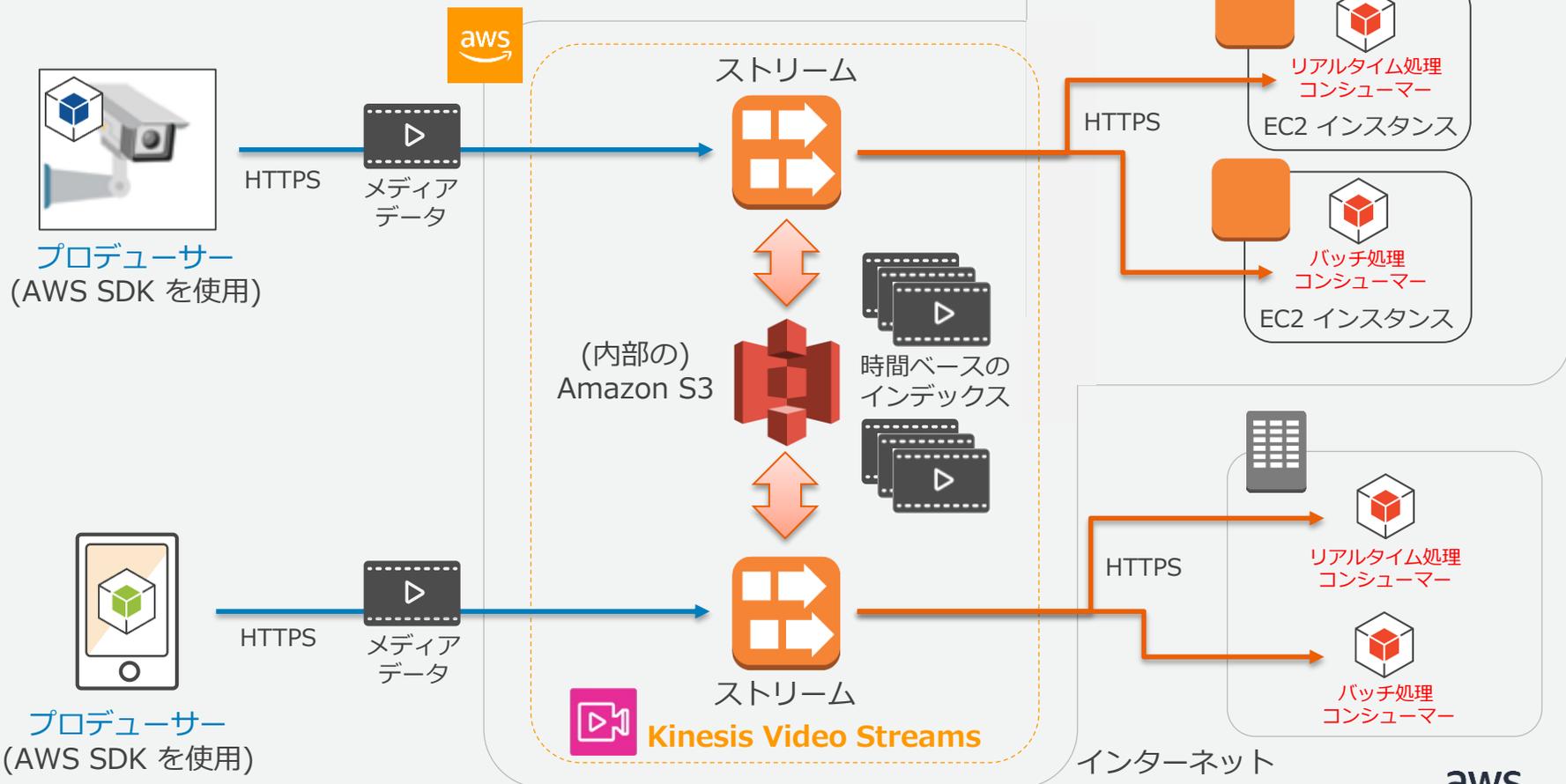


通信および保管データの暗号化と、
IAMでのアクセス管理によるセキュリティ



サーバレスなフルマネージドサービスとして提供

アーキテクチャの概要

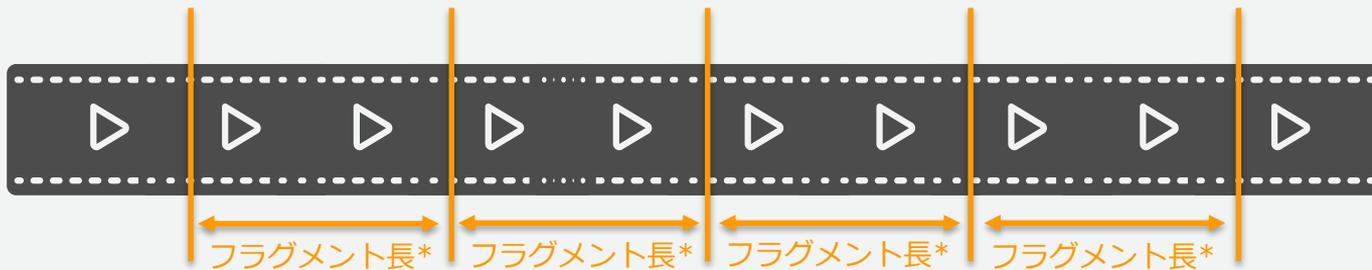


主要なコンセプト

コンセプト名	説明
ストリーム	ライブ動画やその他の時系列データをキャプチャして保存し、リアルタイムやバッチでデータを使用できるようにするためのリソース
プロデューサー	ストリームにデータを送信するデバイスまたはソース（カメラなど）
コンシューマー	ストリームのデータをリアルタイムまたはバッチで取得して処理するカスタムアプリケーション（動画解析アプリケーションなど）
フラグメント	短時間のフレームをまとめたシーケンスであり、固有の番号が割り当てられる。フラグメントに属するフレームは、他のフラグメントのフレームに依存しない。
チャンク	ストリーム内でのデータの格納形式。フラグメント、プロデューサーから送信されたメディアメタデータのコピー、さらにはフラグメント番号、サーバー側とプロデューサー側のタイムスタンプなどの Kinesis Video Streams 固有のメタデータで構成される。
フレーム	動画のもとになる 1 コマの静止画像で、フラグメントに含まれる。必要に応じてコンシューマー側でデコードして取り出し、画像解析等に利用する。

フラグメントとフレーム

動画

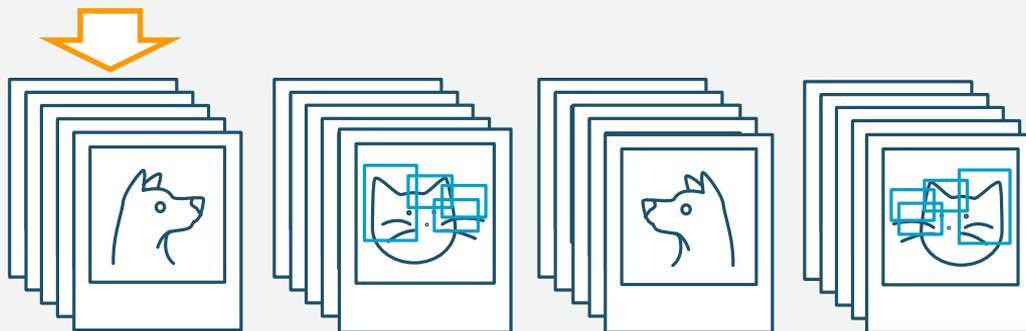


* フラグメント長 : 1-10秒の間で、プロデューサー側にて設定

フラグメント

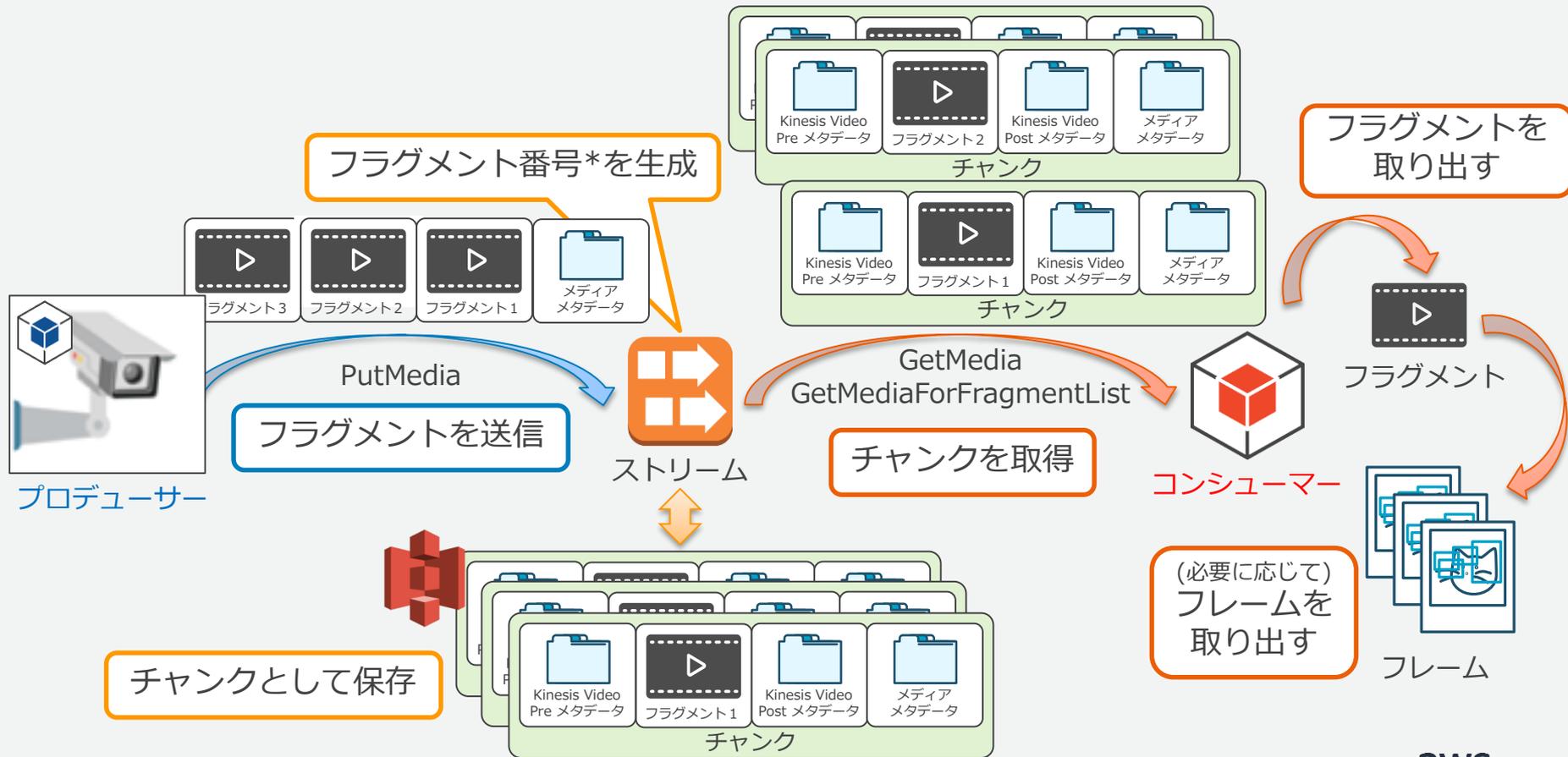


フレーム



フラグメントに属するフレームは、
他のフラグメントのフレームに依存しない

主要なコンセプトの関係性



スケールアウトの単位

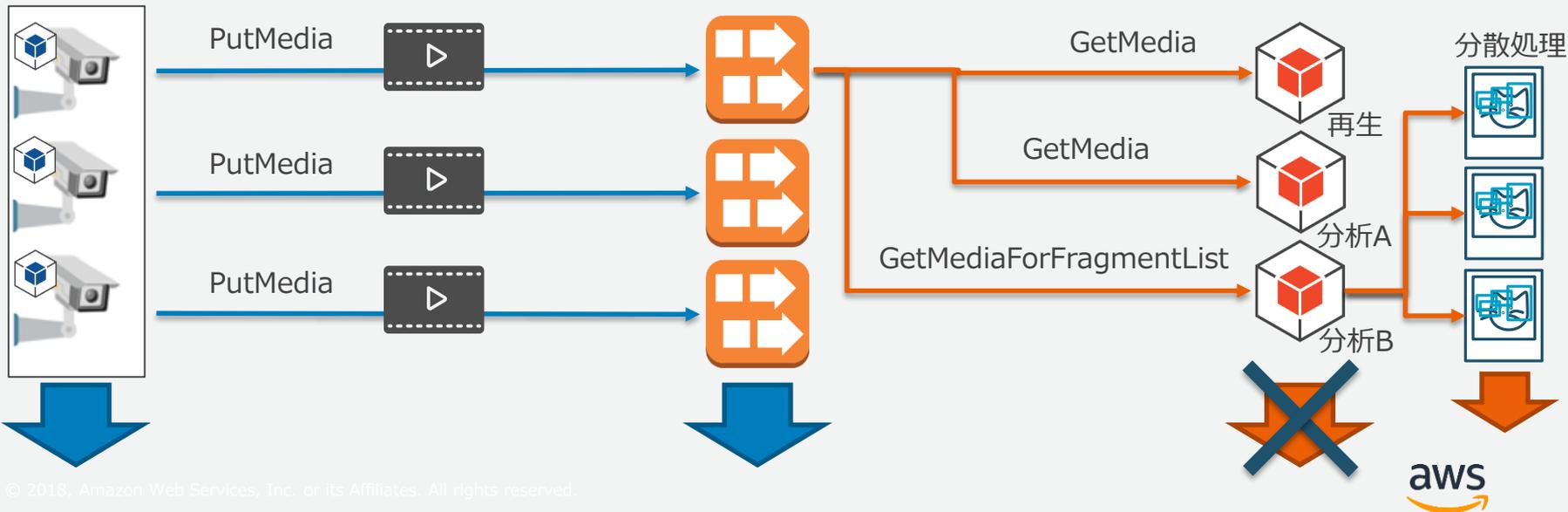
- 基本的にプロデューサーとストリームは1対1の関係
- ストリームの単位で何百万ものデバイスにスケール
- PutMedia APIに呼出頻度や帯域幅の制限があるが、それ以外にストリームのキャパシティ管理は不要
- デバイスのプロビジョニング時に、CreateStream APIによりストリームを動的に作成することを推奨

- ストリームとコンシューマーは1対Nの関係だが、GetMedia APIの同時接続数には制限がある
- コンシューマーは処理内容の役割ごとに分ける
- コンシューマー単位での分散処理はせず、分散処理が必要であればフレーム単位などで考える
- 同じ理由で、不特定多数への動画配信にも向かない

プロデューサー

ストリーム

コンシューマー



マネジメントコンソールの簡易ビューアー

- ストリームを選択して動画を再生可能
- 再生できるのはH.264のコーデックで圧縮された動画のみ（後述）
- 基本的には開発とテスト用途での使用を想定

Amazon Kinesis | tokyo-video-stream-1 | Delete | Refresh

Dashboard

▼ Video preview

2018-03-10 | 18:06:56 | JST | LIVE-4.0s

Server time stamps

Codec: avc1.420020 (Profile: Baseline, Level 3.2) | Bit rate: 523.1 kbps

Resolution: 1280x720 | Fragment duration: 1.5s

Amazon Kinesis | tokyo-video-stream-1 | Delete | Refresh

Dashboard

▼ Video preview

2018-03-10 | 18:07:53 | JST | LIVE-3.9s

Server time stamps

Codec: avc1.420020 (Profile: Baseline, Level 3.2) | Bit rate: 474.7 kbps

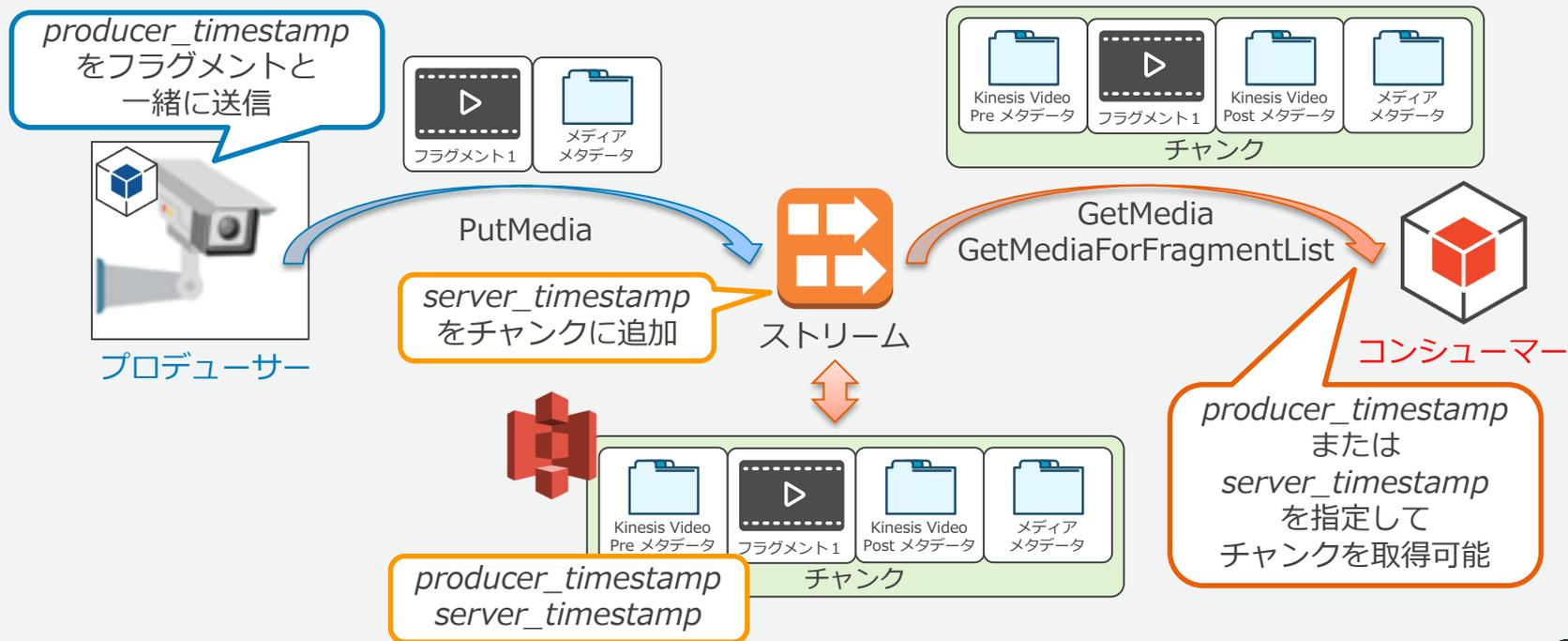
Resolution: 1280x720 | Fragment duration: 1.5s

タイムスタンプの扱い

二種類のタイムスタンプでフラグメントが管理されている

producer_timestamp: プロデューサー側でそのフラグメントの記録を開始した時刻

server_timestamp: Kinesis Video Streams がフラグメントの受信を開始した時刻



リアルタイム処理とバッチ処理 どちらのアプリケーションも構築可能

■ リアルタイム処理

最新の動画ストリームを低レイテンシで取り出すことで、リアルタイムな動画解析アプリケーションの作成が可能

■ バッチ処理

過去のタイムスタンプを指定してフラグメントを取り出すことで、一日分や一週間分をまとめて動画解析するようなバッチ処理のアプリケーションの作成も可能

通信のセキュリティ

■ SSLでの通信暗号化

書き込みと読み込みのAPIを使用した通信は全てHTTPSで暗号化

■ IAMでのアクセス管理

AWSの認証メカニズムである Signature Version 4 を利用して、ストリーム単位、アクション単位でIAMでのアクセス制御が可能
例：

```
"Resource": arn:aws:kinesisvideo:ap-northeast-1:111122223333:stream/my-stream-*
```

```
"Resource": arn:aws:kinesisvideo:*:111122223333:stream/my-stream-1/0123456789012
```

```
"Action": "kinesisvideo:PutMedia"
```

```
"Action": "kinesisvideo:Get*"
```

データの暗号化と保存期間

■ データの暗号化

- Kinesis Video Streams では常にサーバーサイドの暗号化が有効になっている
- データはストレージレイヤーに書き込まれる前に暗号化され、ストレージから取得された後に復号化される
- ストリームの作成時に、暗号化に使用する AWS KMS カスタマーマスターキー (CMK) を指定できる（あとから変更はできない）
- ストリームの作成時にユーザー指定のキーが指定されていない場合は、既定のキー (Kinesis Video Streams が提供) が使用される

■ データの保存期間

- ストリーム作成時にデータの保存期間を指定できる（あとから変更もできる）
- 現在設定可能な範囲は、最短は0（保存しない）で最長は10年（87600時間）
- データの保存量に応じてサービス利用料金が発生する

運用監視に使える CloudWatch メトリクス

CloudWatch メトリクスをストリーム毎に取得可能（以下は一例）

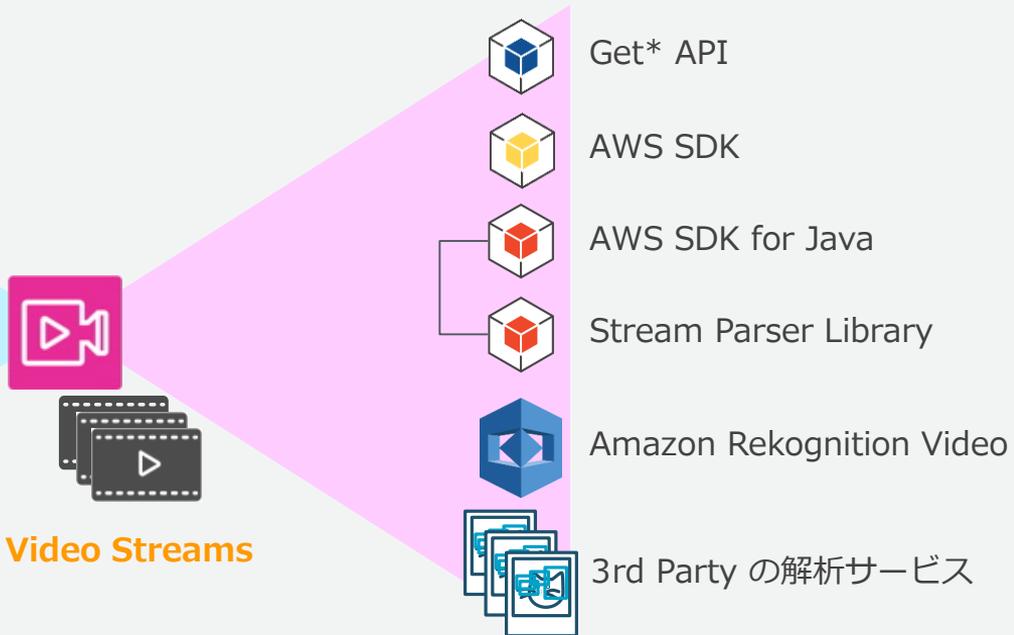
メトリクス	説明
PutMedia.Requests	ストリームへの PutMedia API のリクエスト数
PutMedia.IncomingBytes	PutMedia によってストリームが受信したデータのバイト数
PutMedia.FragmentIngestionLatency	フラグメントの最初と最後のバイトをストリームが受信した時間の差（フラグメントを受信する処理のレイテンシ）
PutMedia.Success	書き込みに成功したフラグメントを1、失敗したフラグメントを0とした平均値（フラグメントの受信成功率を表す指標）
GetMedia.Requests	ストリームへの GetMedia API のリクエスト数
GetMedia.OutgoingBytes	Get Media によってストリームが送信したデータのバイト数
GetMedia.MillisBehindNow	現在のサーバ時間と最後にフラグメントを送信した時間の差（コンシューマーがフラグメントを受信する際の遅延）
GetMedia.Success	送信に成功したフラグメントを1、失敗したフラグメントを0とした平均値（フラグメントの送信成功率を表す指標）

Kinesis Video Streams をサポートする プロデューサーとコンシューマー

プロデューサー (カメラ側)



コンシューマー (動画解析側)



Kinesis Video Streams の料金構成

東京リージョンの場合

<u>料金構成</u>	<u>料金</u>
(プロデューサーから) 取り込まれたデータ量	\$0.01097 / 1GB
(コンシューマーに) 取り出されたデータ量	\$0.01097 / 1GB
保存されたデータ量	\$0.02500 / 1GB - 月

このセミナーの内容

- 📦 Amazon Kinesis Video Streams の特徴
- 📦 Amazon Kinesis Video Streams を動かすまで
 1. 動画データのフォーマット
 2. プロデューサー (カメラ側)
 3. コンシューマー (動画解析側)
 4. デモ
- 📦 Amazon Kinesis Video Streams を活用した構成例
- 📦 まとめ

このセミナーの内容

- 📦 Amazon Kinesis Video Streams の特徴
- 📦 Amazon Kinesis Video Streams を動かすまで
 1. 動画データのフォーマット
 2. プロデューサー (カメラ側)
 3. コンシューマー (動画解析側)
 4. デモ
- 📦 Amazon Kinesis Video Streams を活用した構成例
- 📦 まとめ

動画データのフォーマット

コンテナ と コーデック

■ コンテナ (コンテナフォーマット)

- 映像データと音声データなどをまとめて動画データにするためのファイルフォーマット (入れ物に相当)
- 入れ物なので、画質や音質には影響しない
- 代表的なコンテナとして、MP4、AVI、MOV、MPEG、**MKV**、WMV などがある

■ コーデック

- コンテナに入れるデータを圧縮するためのアルゴリズム
- 映像データや音声データにそれぞれコーデックがある
- コンテナによって使用可能なコーデックが決まっている
- 代表的な映像コーデックとして **H.264** がある

コンテナ

MP4/AVI/MOV/MPEG/MKV/WMV など

映像データ

映像コーデック

H.264/MPEG2/
Motion JPEG/
DivX/Xvid
など

音声データ

音声コーデック

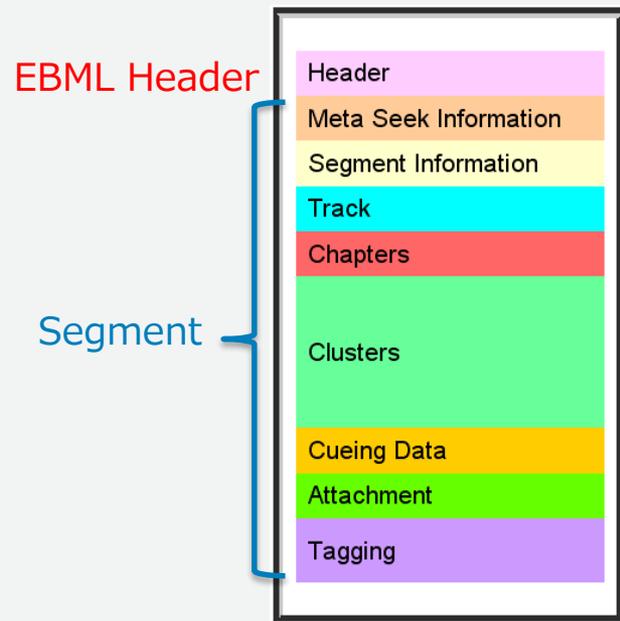
MP3/AAC/
WMA/Vorbis/
FLAC//WAV
など

Matroska (MKV)

オープンスタンダードなコンテナフォーマット

- 動画ファイルの拡張子には .mkv が使われる
- 異なるコーデックで圧縮された映像、音声に加えて、副音声、字幕なども格納できる
- ほとんど全てのコーデックをサポートする
(映像コーデックはH.264/MPEG2/Xvidなど)
- ファイルが一部欠けていても他の部分を再生できる
- オープンソースで開発されている

Kinesis Video Streams のAPIおよび
周辺ライブラリは、**MKVのみをサポート**



EBML Header
Segment
EBML Header
Segment

MKVのデータ構造



<https://www.matroska.org/technical/whatis/index.html>

https://docs.aws.amazon.com/ja_jp/kinesisvideostreams/latest/dg/how-data.html

MKVコンテナの中のデータ形式

Kinesis Video Streams では、
MKVコンテナの中に格納するデータのコーデックは問わない

- ユーザーがコンシューマー側でデータをデコードする必要がある
- そのため、動画以外の時系列データも扱うことが可能
- ただし、マネジメントコンソール上の簡易ビューアで再生できるのは、**H.264** でエンコードされた動画のみ（メディアの種類は video/h264）

Kinesis Video Streams の活用に必須なデータ形式は、
MKVのコンテナフォーマットのみ

+

MKVコンテナの中のコーデックは問わないが、
通常の動画を扱うユースケースでは H.264 を使用することが多い

このセミナーの内容

- 📦 Amazon Kinesis Video Streams の特徴
- 📦 Amazon Kinesis Video Streams を動かすまで
 1. 動画データのフォーマット
 2. プロデューサー (カメラ側)
 3. コンシューマー (動画解析側)
 4. デモ
- 📦 Amazon Kinesis Video Streams を活用した構成例
- 📦 まとめ

PutMedia API

MKVフォーマットのフラグメントをPayloadとして、HTTPS POSTする

- APIリクエストに必要なパラメータは HTTPヘッダに入れて送信する
- PutMedia API は Long Running Sessionのため、セッションを張って、そのセッションに対してフラグメントを送信する

Request

```
POST /putMedia HTTP/1.1
x-amzn-stream-name: StreamName
x-amzn-stream-arn: StreamARN
x-amzn-fragment-timecode-type: FragmentTimecodeType
x-amzn-producer-start-timestamp: ProducerStartTimestamp
```

PayLoad

Response

```
HTTP/1.1 200
```

PayLoad

PutMedia API に必要なリクエストパラメータ

Header	内容
StreamARN	Kinesis Video Stream のストリーム の Amazonリソースネーム (StreamARN か StreamName のいずれかが必須パラメータ)
StreamName	Kinesis Video Stream のストリーム名 (StreamARN か StreamName のいずれかが必須パラメータ)
FragmentTimecodeType (ABSOLUTE RELATIVE)	ペイロードに含まれるフラグメント内のタイムスタンプが producerStartTimestamp からの絶対値であるか相対値で あるかを示すタイプ Kinesis Video Streamsはこの情報を使用して、Put Media APIで 受信したフラグメントの <i>producer_timestamp</i> を計算する
ProducerStartTimestamp	プロデューサーがメディアの記録を開始した時点のプロデューサー 側のタイムスタンプ (ペイロードに含まれる特定のフラグメントのタイムスタンプではない)

PutMedia API の注意点

- PutMedia APIを使用する前に、GetDataEndpoint APIを呼び出してエンドポイントを取得する必要がある
- PutMedia APIには、以下の制限がある
 - PutMedia APIコールの上限：1 ストリームあたり、秒間5回
 - 同時接続数：1 ストリームあたり1つまで（トークンローテーションをする場合は5まで）
（上限緩和は可能・制限を超えると最後の接続が有効）
 - フラグメントの送信上限：1 ストリームあたり、秒間5フラグメント
 - フラグメントのサイズ：最小時間 1秒、最大時間 10秒、最大サイズ 50 MB
 - Kinesis Video Streamsのデータ受信処理：
1つの PutMedia セッションの間で 12.5 MB/second または 100 Mbps
（上限緩和は可能）

Amazon Kinesis Video Streams Producer SDK

プロデューサー・アプリケーションの実装を簡易にするためのSDK

デバイスのメディアパイプラインを、Kinesis Video Streams のストリームに統合するための機能を一通貫で提供

- 動的なストリームの作成
- AWSの認証メカニズムである Signature Version 4 を利用した認証やトークンローテーションの仕組み
- PutMedia API を使用して、柔軟にフレームやバッファリングされたフラグメントをストリーミング、あるいはバッチアップロードするためのフレームワーク
- オープンソースとして公開

Producer SDK の構成

Linux

Android

Wrapper Layer
(C++ and Java)

Platform Independent Layer
(Platform Independent Codebase - C言語)

デモアプリケーション

- **アプリケーション開発者向け**
- ターゲットOSにインストールしてそのまま利用
- ハードウェアやビデオソースを全てサポートするわけではないが、簡単に実行可能

SDK

- **カメラから取得した動画を統合したい開発者向け**
- 柔軟にカスタマイズが可能な、オブジェクト指向の統合フレームワークを提供

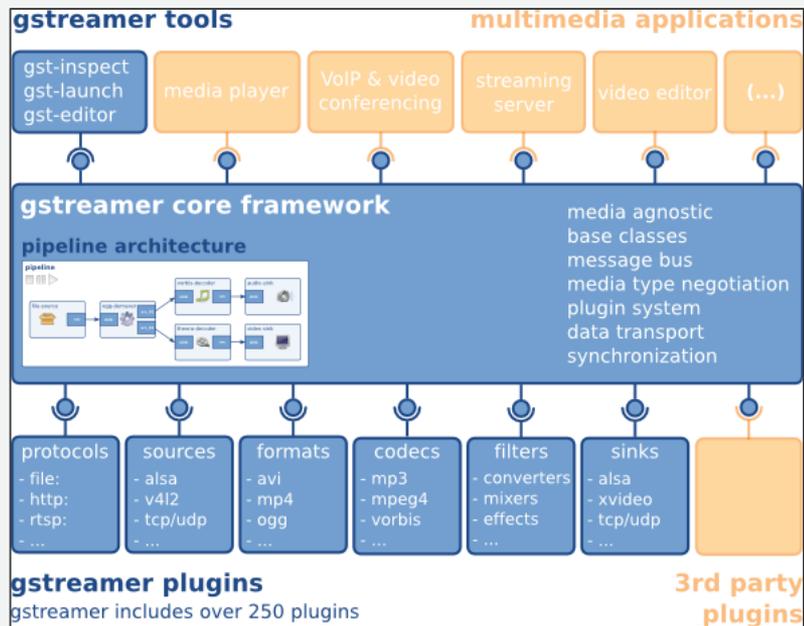
コア機能を提供するライブラリ

- **(カメラ) メーカー向け**
- 様々なハードウェア環境に合わせて、ファームウェアレベルで動画ソースとの統合を実装可能
- 他のライブラリとは完全に独立

参考 : GStreamer とは

オープンソースのマルチメディアアプリケーション開発用フレームワーク

- 動画プレイヤーのベースとなる部分などに使われており、動画のストリーミング配信、動画の変換・合成などの処理を実装できる
- 核となる部分以外は、プラグインのライブラリ群で構成されており、様々な通信プロトコルやコーデックに対応している
- C++ Producer SDK には、GStreamer で Kinesis Video Stream を使用するためのデモアプリケーションが含まれている



GStreamer のデモアプリケーション

GStreamer Webカメラ デモアプリケーション

端末にUSB等で接続したWebカメラから、
Kinesis Video Streams に動画を送信

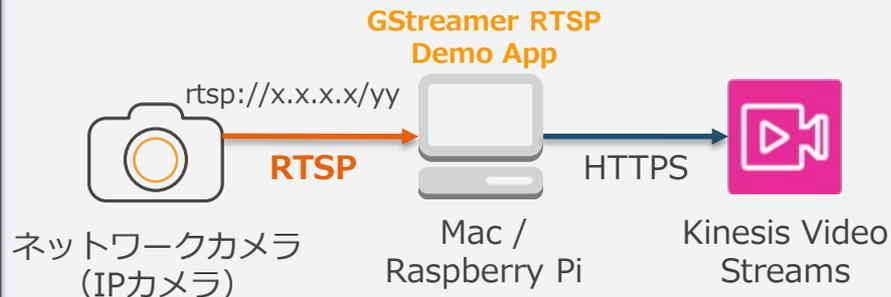
以下のカメラが動作対象※

- Mac (macOS) の組み込みカメラ
- Linux または Raspberry Pi の
デバイスに接続されたUSBカメラ



GStreamer RTSP デモアプリケーション

RTSPで接続可能なネットワークカメラから、
Kinesis Video Streams に動画を送信



Producer SDK のデモアプリケーションを すぐに実行可能な環境

GStreamer Webカメラ
デモアプリケーション
(Webカメラ、内蔵カメラ)

GStreamer RTSP
デモアプリケーション
(ネットワークカメラ)

Android
デモアプリケーション
(スマートフォン内蔵カメラ)

C++ Producer SDK

Android
Producer Library

macOS

Ubuntu

Raspberry Pi

Android

パートナーの対応カメラ



全スマカメシリーズ



Kinesis Video Streamsに対応した
カメラを活用することで、
動画をそのままAWSに取り込む
ことができ、ユーザーはクラウド上
での動画処理に集中できる

ユースケース: 監視カメラで問題行動に反応

Kinesis Video Streams から読み込んだ監視カメラのストリームを、リアルタイムで分析して素早いアクション



例: プラネックスコミュニケーションズ株式会社様

https://www.planex.co.jp/product/network_camera.html#camera01

<https://www.slideshare.net/AmazonWebServicesJapan/ivs-cto-night-dayai-machine-learning-on-aws>



このセミナーの内容

- 📦 Amazon Kinesis Video Streams の特徴
- 📦 Amazon Kinesis Video Streams を動かすまで
 1. 動画データのフォーマット
 2. プロデューサー (カメラ側)
 3. コンシューマー (動画解析側)
 4. デモ
- 📦 Amazon Kinesis Video Streams を活用した構成例
- 📦 まとめ

コンシューマー側のAPI

リアルタイム
処理指向

GetMedia API

Kinesis Video Streams のストリームから、メディアデータを取得するためのAPI

- ストリーム名またはストリーム Amazon リソースネーム (ARN) と、開始チャンクをリクエストする
- Kinesis Video Streams はフラグメント番号順に**チャンク**のストリームを返す

バッチ
処理指向

GetMediaForFragmentList API

ストリームに保存されたアーカイブデータから指定したメディアデータを取得するAPI

- フラグメントのリスト (フラグメント番号で指定)を指定してリクエストする
- 通常は、この API を呼び出す前に、ListFragments API を呼び出す

ListFragments API

フラグメントのリストを取得するAPI

- フラグメント番号またはタイムスタンプを使用して、ストリームに対して開始位置を指定してリクエストする

GetMedia API

PutMediaで送信されたフラグメントを含むチャンクのストリームをPayloadとして受信する

- リクエストパラメータは JSON形式でBodyの中に入れて送信する
- GetMedia API は Long Running Sessionのため、セッションを張って、そのセッションの中でチャンクを受信する

Request

```
POST /getMedia HTTP/1.1
Content-type: application/json

{
  "StartSelector": {
    "AfterFragmentNumber": "string",
    "ContinuationToken": "string",
    "StartSelectorType": "string",
    "StartTimestamp": number
  },
  "StreamARN": "string",
  "StreamName": "string"
}
```

Response

```
HTTP/1.1 200
Content-Type: ContentType

Payload
```

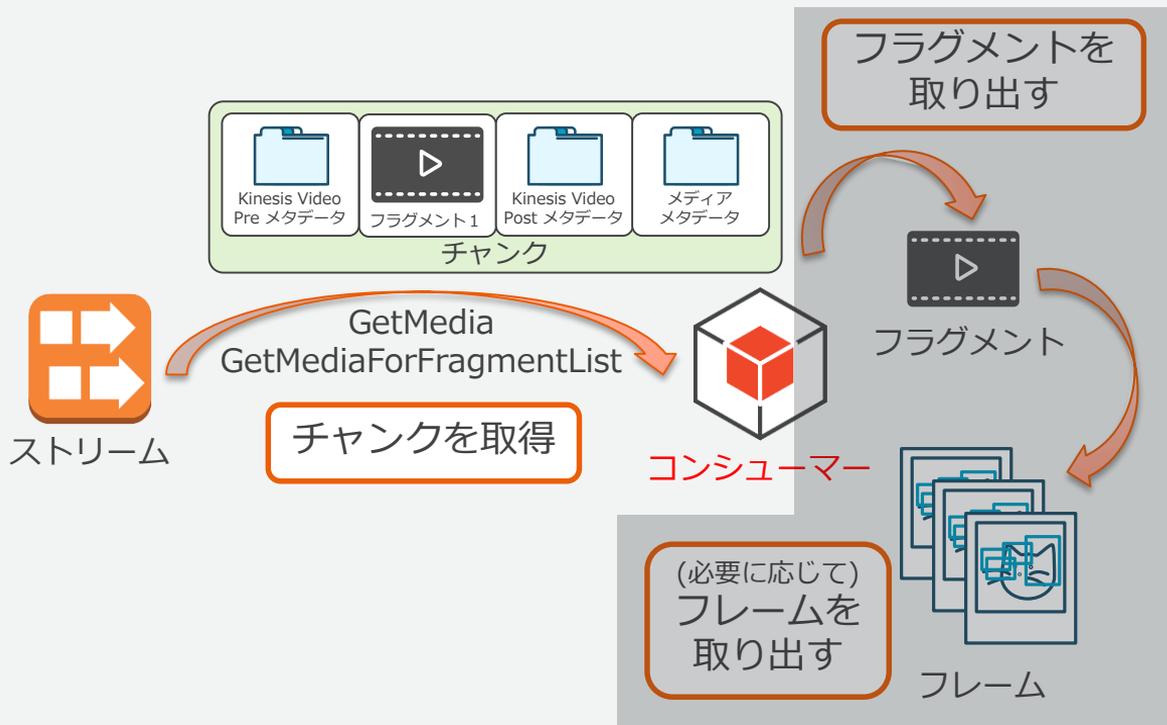
GetMedia API に必要なリクエストパラメータ

Header	内容
StartSelector	ストリームから取得するチャンクの開始地点
AfterFragmentNumber	GetMedia APIが返す最初のフラグメントのフラグメント番号
ContinuationToken	前のGetMediaレスポンスで返された ContinuationToken を指定することで、チャンクの開始地点を判別するためのトークン
StartSelectorType (Required)	GetMedia APIが返す最初のフラグメントの指定方法 (NOW EARLIEST FRAGMENT_NUMBER CONTINUATION_TOKEN PRODUCER_TIMESTAMP SERVER_TIMESTAMP)
StartTimestamp	GetMedia APIが返す最初のフラグメントが含むタイムスタンプ (StartSelectorType に合わせて producer_timestamp か server_timestamp を指定)
StreamARN	Kinesis Video Stream のストリームの Amazonリソースネーム (StreamARN か StreamName のいずれかが必須パラメータ)
StreamName	Kinesis Video Stream のストリーム名 (StreamARN か StreamName のいずれかが必須パラメータ)

GetMedia API の注意点

- GetMedia APIを使用する前に、GetDataEndpoint APIを呼び出してエンドポイントを取得する必要がある
- GetMedia APIには、以下の制限がある
 - GetMedia APIコールの上限：1ストリームあたり、秒間5回
 - 同時接続数：1ストリームあたり3つまで
(上限緩和は可能・制限を超える接続は拒否される)
 - Kinesis Video Streams のデータ送信処理：
1つのGetMedia セッションの間で 25.0 MB/second または 200 Mbps
(上限緩和は可能)

GetMedia API で取得できるのは、 あくまでMKV形式のチャンクでしかない



Stream Parser Library

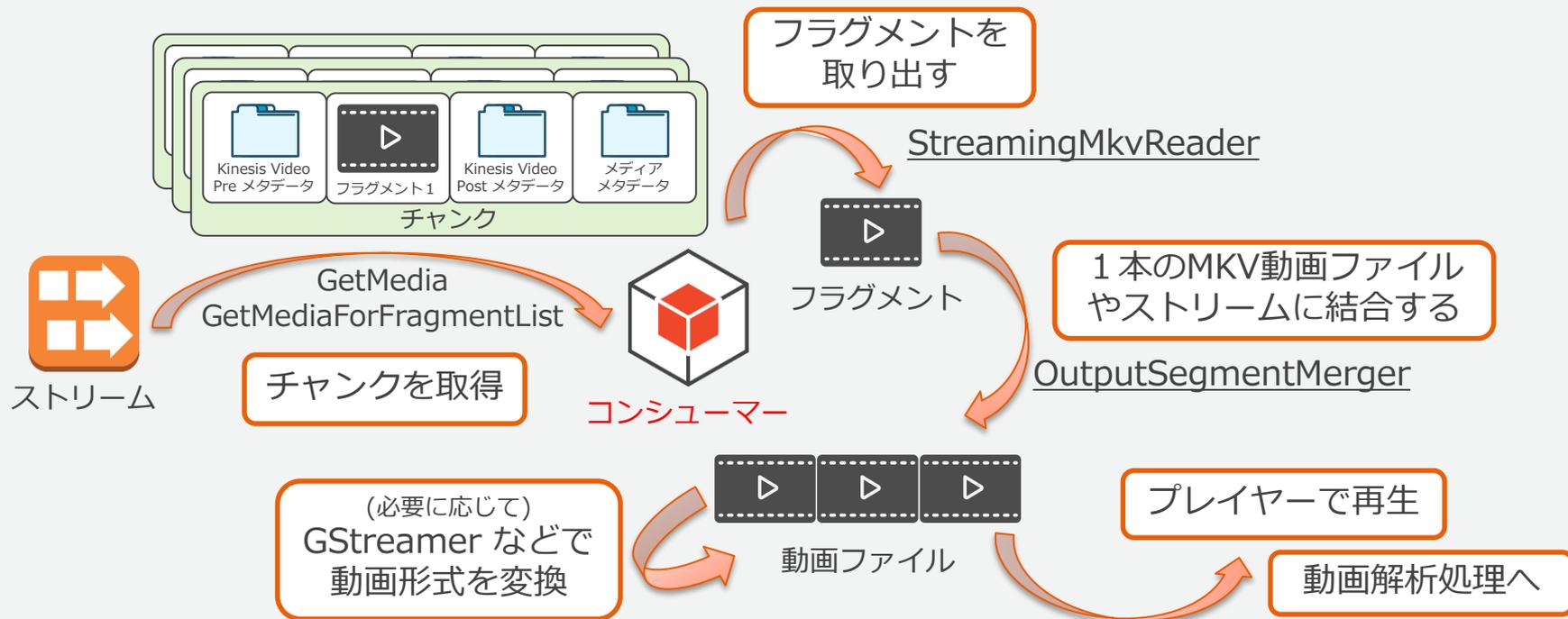
ストリームから取得したMKV形式のデータを使いやすい形に加工するためのライブラリ

- Javaのライブラリ（Jarファイル）として提供され、AWS SDK for Java と組み合わせてJavaアプリケーション内で使用する
- 現在の実装では、主なツールとして以下のものが含まれる

主なツール	内容
StreamingMkvReader	MKVデータをストリームから MKV Element として読み取る
FragmentMetadataVisitor	フラグメント（メディアデータ）およびトラック（コーデックの種類やピクセル幅・高さなどを含むデータ）からメタデータを取得する。フラグメント番号やタイムスタンプの情報もここから取得出来る。
OutputSegmentMerger	異なるトラックのメタデータを単一のセグメントを持つストリームにマージして、連続したフラグメント（チャンク）を結合する
FrameVisitor	フラグメントからフレームを取り出してフレーム毎の処理に渡す。インタフェースが提供されるので、デコードと処理内容は自分で実装する。
KinesisVideoExample	Stream Parser Library の使用方法を示すサンプルアプリケーション

動画ファイルを出力して解析する

Stream Parser Library の OutputSegmentMerger を活用する



GStreamer : <https://gstreamer.freedesktop.org/>

OpenPose : <https://github.com/CMU-Perceptual-Computing-Lab/openpose>



動画ファイルを出力して解析する

コンシューマ側の実装の例（一部のみ）

```
// Get media request loop.
while (true) {
    GetMediaResult result = mediaClient.getMedia(getMediaRequest());

    Path outputPath = Paths.get(OUTPUT_FILENAME);
    OutputStream fileOutputStream = Files.newOutputStream(outputPath, StandardOpenOption.WRITE, StandardOpenOption.CREATE);
    try (BufferedOutputStream outputStream = new BufferedOutputStream(fileOutputStream)) {

        StreamingMkvReader mkvStreamReader = StreamingMkvReader.createDefault(new InputStreamParserByteSource(result.getPayload()));
        OutputSegmentMerger outputSegmentMerger = OutputSegmentMerger.createDefault(fileOutputStream);

        // Apply the OutputSegmentMerger to the mkv elements from the mkv stream reader.
        while (mkvStreamReader.mightHaveNext()) {
            Optional<MkvElement> mkvElementOptional = mkvStreamReader.nextIfAvailable();
            if (mkvElementOptional.isPresent()) {
                MkvElement mkvElement = mkvElementOptional.get();
                // Apply the segment merger to this element.
                mkvElement.accept(outputSegmentMerger);

                // Flush output stream per fragment.
                if (MkvTypeInfoos.SEGMENT.equals(mkvElement.getElementMetaData().getTypeInfo())) {
                    if (mkvElement instanceof MkvEndMasterElement) {
                        outputStream.flush();
                    }
                }
            }
        }
    } catch (MkvElementVisitException | IOException e) {}

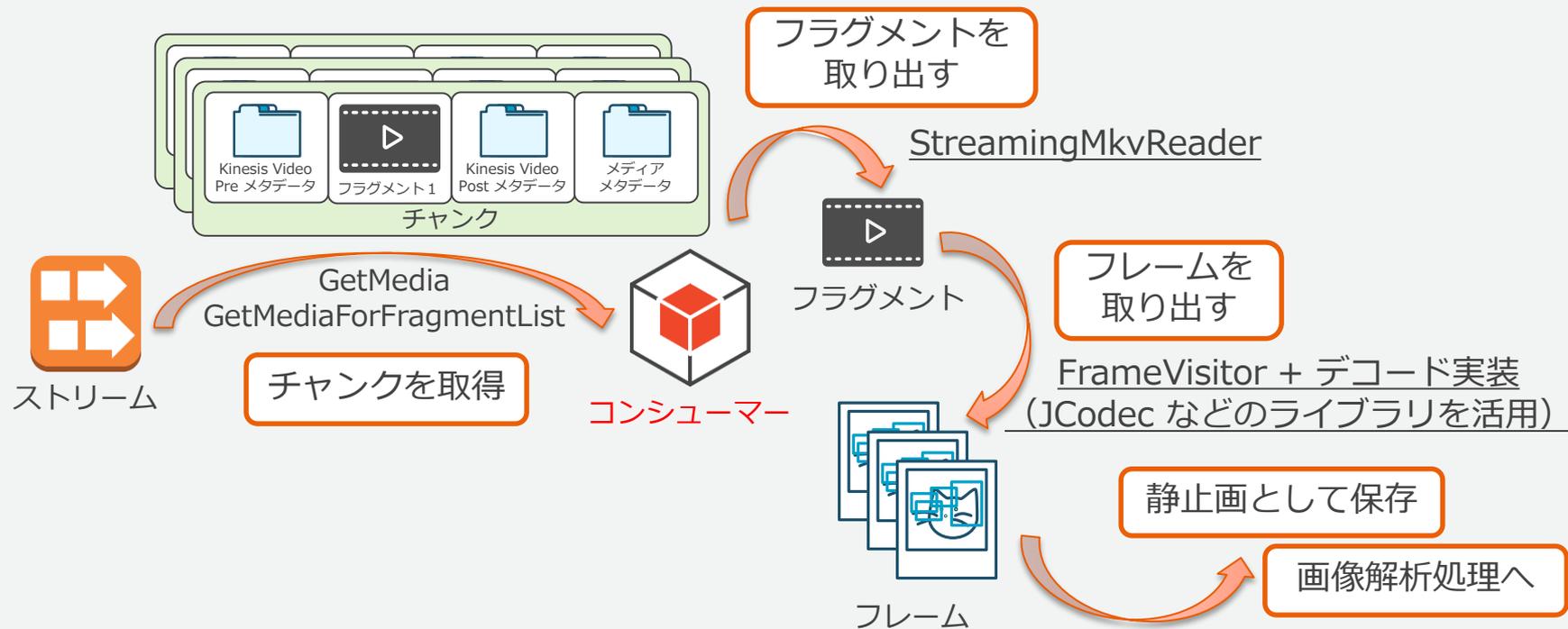
    wait(GETMEDIA_INTERVAL);
}
}
```

ここでフラグメントを
マージする

※このコードは簡略化した例です

画像フレームを取り出して解析する

Stream Parser Library の FrameVisitor を活用する



画像フレームを取り出して解析する

コンシューマ側の実装の例（一部のみ）

```
// Get media request loop.
while (true) {
    GetMediaResult result = mediaClient.getMedia(getMediaRequest());

    // Fragment metadata visitor to extract Kinesis Video fragment metadata from the GetMedia stream.
    FragmentMetadataVisitor fragmentMetadataVisitor = FragmentMetadataVisitor.create();
    MyFrameProcessor frameProcessor = MyFrameProcessor.create();
    FrameVisitor frameVisitor = FrameVisitor.create(frameProcessor);
    MkvElementVisitor elementVisitor = new CompositeMkvElementVisitor(fragmentMetadataVisitor, frameVisitor);
    StreamingMkvReader mkvStreamReader = StreamingMkvReader.createDefault(new InputStreamParserByteSource(result.getPayload()));

    // Apply the OutputSegmentMerger to the mkv elements from the mkv stream reader.
    try {
        while (mkvStreamReader.mightHaveNext()) {
            Optional<MkvElement> mkvElementOptional = mkvStreamReader.nextIfAvailable();
            if (mkvElementOptional.isPresent()) {
                MkvElement mkvElement = mkvElementOptional.get();

                // Apply the parsing visitor to this element through the element visitor.
                mkvElement.accept(elementVisitor);
            }
        }
    } catch (MkvElementVisitException e) {}

    wait(GETMEDIA_INTERVAL);
}
}
```

実装した *FrameProcessor* を
FrameVisitor に渡して
ストリーム処理を初期化する

ここで *FrameProcessor* に
実装した処理が呼ばれる

※このコードは簡略化した例です

画像フレームを取り出して解析する

FrameProcessor 実装の例

(*Jcodec* を利用した H264 デコード処理 と *OpenCV* を利用した顔検出)

```
public class MyFrameProcessor implements FrameVisitor.FrameProcessor {  
    private byte[] codecPrivateData;  
    private final H264Decoder decoder = new H264Decoder();  
    private final Transform transform = new Yuv420jToRgb();  
  
    @Override  
    public void process(Frame frame, MkvTrackMetadata trackMetadata) {  
        ByteBuffer frameBuffer = frame.getFrameData();  
        int pixelWidth = trackMetadata.getPixelWidth().get().intValue();  
        int pixelHeight = trackMetadata.getPixelHeight().get().intValue();  
        codecPrivateData = trackMetadata.getCodecPrivateData().array();  
  
        Picture rgb = Picture.create(pixelWidth, pixelHeight, ColorSpace.RGB);  
        BufferedImage imageFrame = new BufferedImage(pixelWidth, pixelHeight, BufferedImage.TYPE_3BYTE_BGR);  
        AvcCBox avcC = AvcCBox.parseAvcCBox(ByteBuffer.wrap(codecPrivateData));  
        decoder.addSps(avcC.getSpsList());  
        decoder.addPps(avcC.getPpsList());  
        Picture buf = Picture.create(pixelWidth, pixelHeight, ColorSpace.YUV420j);  
        List<ByteBuffer> byteBuffers = splitMOVPacket(frameBuffer, avcC);  
        Picture pic = decoder.decodeFrameFromNals(byteBuffers, buf.getData());  
        // 略..  
        Picture tmpBuf = Picture.createPicture(pixelWidth, pixelHeight, dataTemp, ColorSpace.YUV420j);  
        transform.transform(tmpBuf, rgb);  
        AWTUtil.toBufferedImage(rgb, imageFrame);  
  
        // Find the faces in the frame by OpenCV Haar cascade classifier  
        Mat source = Java2DFrameUtils.toMat(imageFrame);  
        RectVector faces = new RectVector();  
        faceDetector.detectMultiScale(source, faces);  
        LOG.info(faces.size() + " faces are detected!");  
    }  
}
```

メタデータから
コーデックの種類や
ピクセル幅・高さなど
必要な情報を取り出す

JCodec による
デコード処理をして
フレームを取り出す

OpenCV の顔検出器に
かけて画像解析を行う

※このコードは簡略化した例です

画像フレームを取り出して解析する

画像フレームを取り出せれば、様々な画像解析にかけられる

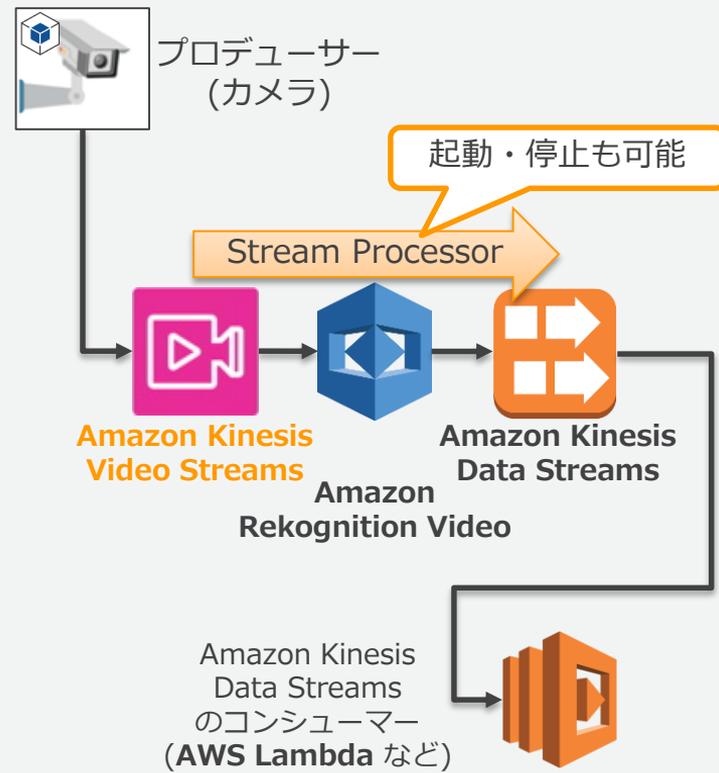
- OpenCV や OpenPose などのライブラリを活用した画像解析
- Amazon Rekognition などの画像解析サービス
- 独自に Deep Learning などで機械学習したモデルでの推論

など . . .

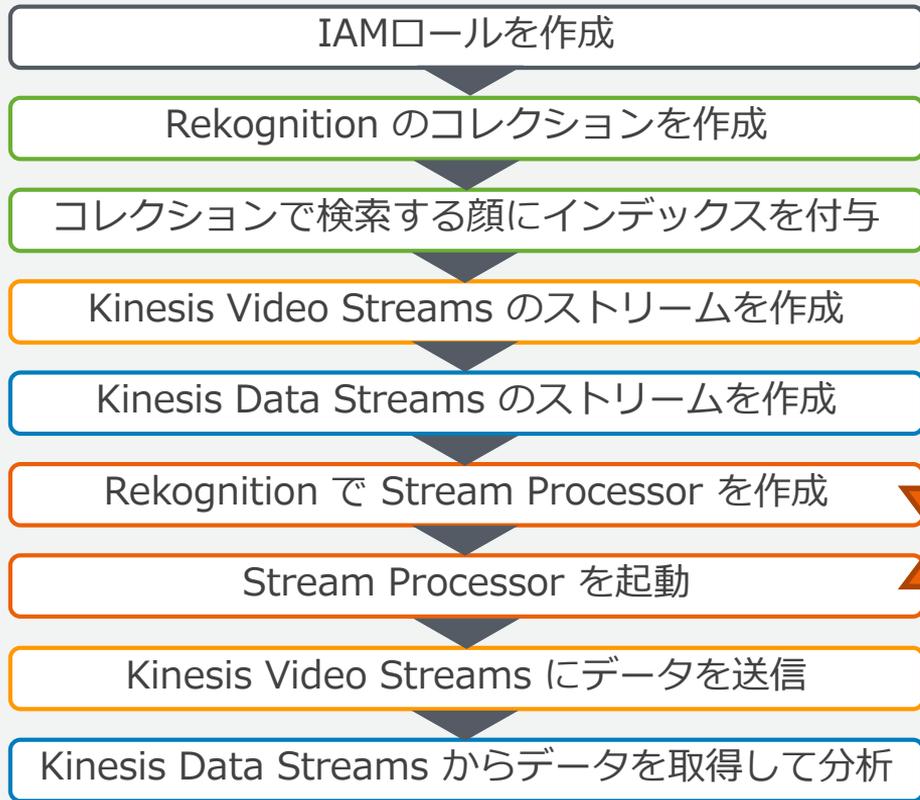
Amazon Rekognition Video と連携する



- Kinesis Video Streams のコンシューマーとして Amazon Rekognition Video を使用して動画ストリーム内の顔を検出・認識できる
- 予め Rekognition Video に覚えさせておいた既知の顔を検出することができる
- Rekognition Video には、動画ストリームの分析を管理するための Stream Processor が用意されている
- 分析結果は、Rekognition Video から、Amazon Kinesis Data Streams のストリームに出力される



Rekognition Video を使用した顔認識の設定



```
$ aws rekognition create-stream-processor ¥
--name "MyKinesisVideoStreamProcessor" ¥
--input '{"KinesisVideoStream": {"Arn":
"arn:aws:kinesisvideo:ap-northeast-1:XXXXXXXXXXXX:
stream/my-video-stream/XXXXXXXXXXXX"}}' ¥
--stream-processor-output '{"KinesisDataStream": {"Arn":
"arn:aws:kinesis:ap-northeast-1:XXXXXXXXXXXX:
stream/AmazonRekognition"}}' ¥
--role-arn "arn:aws:iam::XXXXXXXXXXXX:role/
MyRekognitionForKinesisRole" ¥
--settings '{"FaceSearch": {"CollectionId": "MyFaceCollection",
"FaceMatchThreshold": 80.0}}'
```

```
$ aws rekognition start-stream-processor ¥
--name "MyKinesisVideoStreamProcessor"
```

AWS CLI での作成例

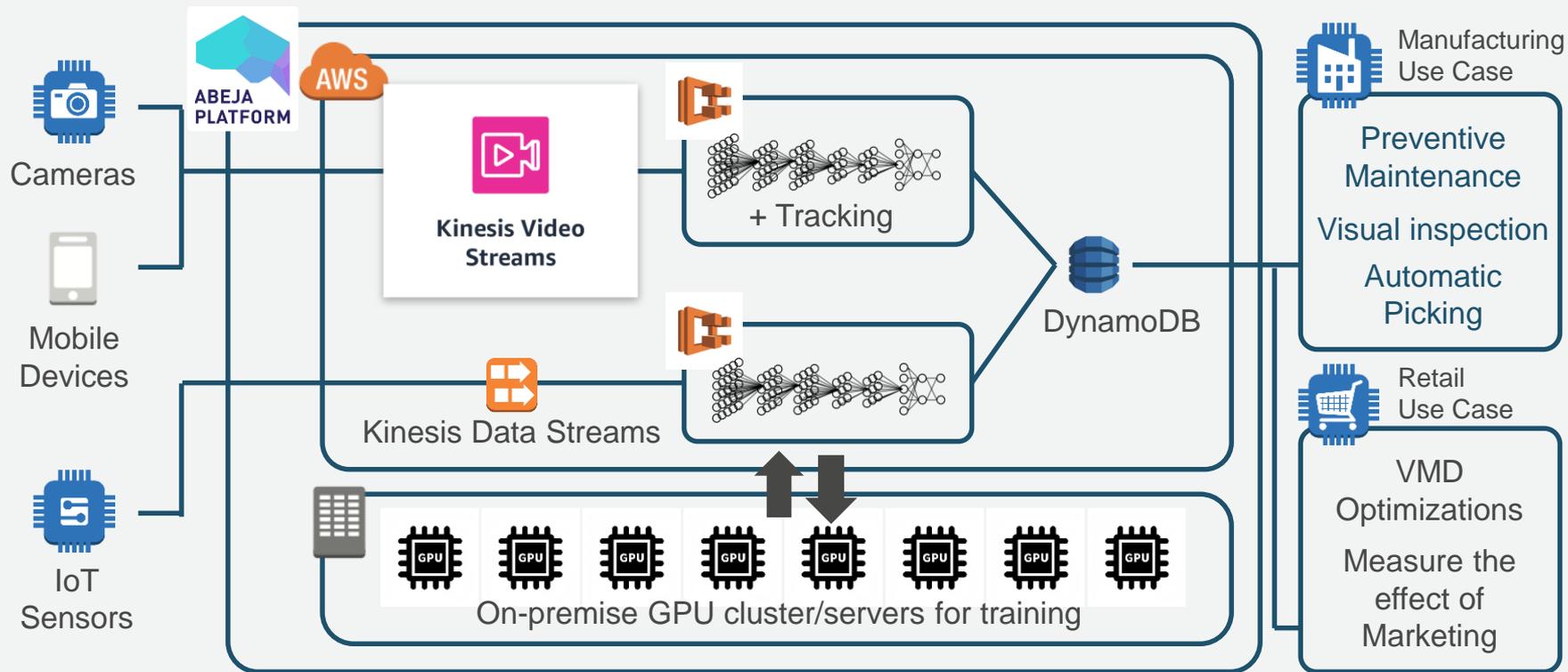
Rekognition Video による顔認識の分析結果



```
{
  "InputInformation":{
    "KinesisVideo":{
      "StreamArn": "arn:aws:kinesisvideo:ap-northeast-1:XXXXXXXXXXXX:stream/my-video-stream/XXXXXXXXXXXX ",
      "FragmentNumber": "91343852348798144617425944722918553756896652976",
      "ServerTimestamp": "1.521183662414E9",
      "ProducerTimestamp": "821785.28",
      "FrameOffsetInSeconds": "1.0"
    }
  },
  "StreamProcessorInformation": {"Status": "RUNNING"},
  "FaceSearchResponse": {
    "DetectedFace": {
      "BoundingBox": {"Height": 0.45, "Width": 0.253125, "Left": 0.47109374, "Top": 0.3472222},
      "Confidence": 99.99982,
      "Landmarks": [
        {"X": 0.55657715, "Y": 0.48627222, "Type": "eyeLeft"},
        {"X": 0.65045106, "Y": 0.49128634, "Type": "eyeRight"},
        {"X": 0.60852575, "Y": 0.5521961, "Type": "nose"},
        {"X": 0.5619316, "Y": 0.6824252, "Type": "mouthLeft"},
        {"X": 0.6378561, "Y": 0.6886522, "Type": "mouthRight"}
      ],
      "Pose": {"Pitch": 17.475468, "Roll": 0.80840683, "Yaw": 7.7559447},
      "Quality": {"Brightness": 32.13126, "Sharpness": 99.93052}
    },
    "MatchedFaces": [
      {
        "Similarity": 90.26169,
        "Face": {
          "BoundingBox": {"Height": 0.4275, "Width": 0.4275, "Left": 0.26375, "Top": 0.26875},
          "FaceId": "063808fa-0018-4f3d-8deb-b0dd4d18fbb3",
          "Confidence": 99.9869,
          "ImageId": "10b0ad0a-c240-7b02-e648-237bc6f5a82b"
        }
      }
    ]
  }
}
```

パートナーの動画解析サービスを利用

今後はパートナーによる動画解析サービスも利用可能に



例:株式会社ABEJA様

re:Invent2017 : <https://www.youtube.com/watch?v=rjBXbktBxBg&t=39m15s>

JAWS-UG AI : <https://speakerdeck.com/toshitanian/amazon-kinesis-video-streams-x-deep-learning>



このセミナーの内容

- 📦 Amazon Kinesis Video Streams の特徴
- 📦 Amazon Kinesis Video Streams を動かすまで
 1. 動画データのフォーマット
 2. プロデューサー (カメラ側)
 3. コンシューマー (動画解析側)
 4. デモ
- 📦 Amazon Kinesis Video Streams を活用した構成例
- 📦 まとめ



This repository Search

Pull requests Issues Marketplace Explore

awslabs / amazon-kinesis-video-streams-producer-sdk-cpp

Unwatch 22

Unstar 45

Fork 19

Code

Issues 6

Pull requests 0

Projects 0

Insights

Amazon Kinesis Video Streams Producer SDK for C++ is for developers to install and customize for their connected camera and other devices to securely stream video, audio, and time-encoded data to Kinesis Video Streams.

53 commits

1 branch

8 releases

9 contributors

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

zhiyua-git Release 1.3.0 (15th March 2018)

Latest commit 3b257a0 3 days ago

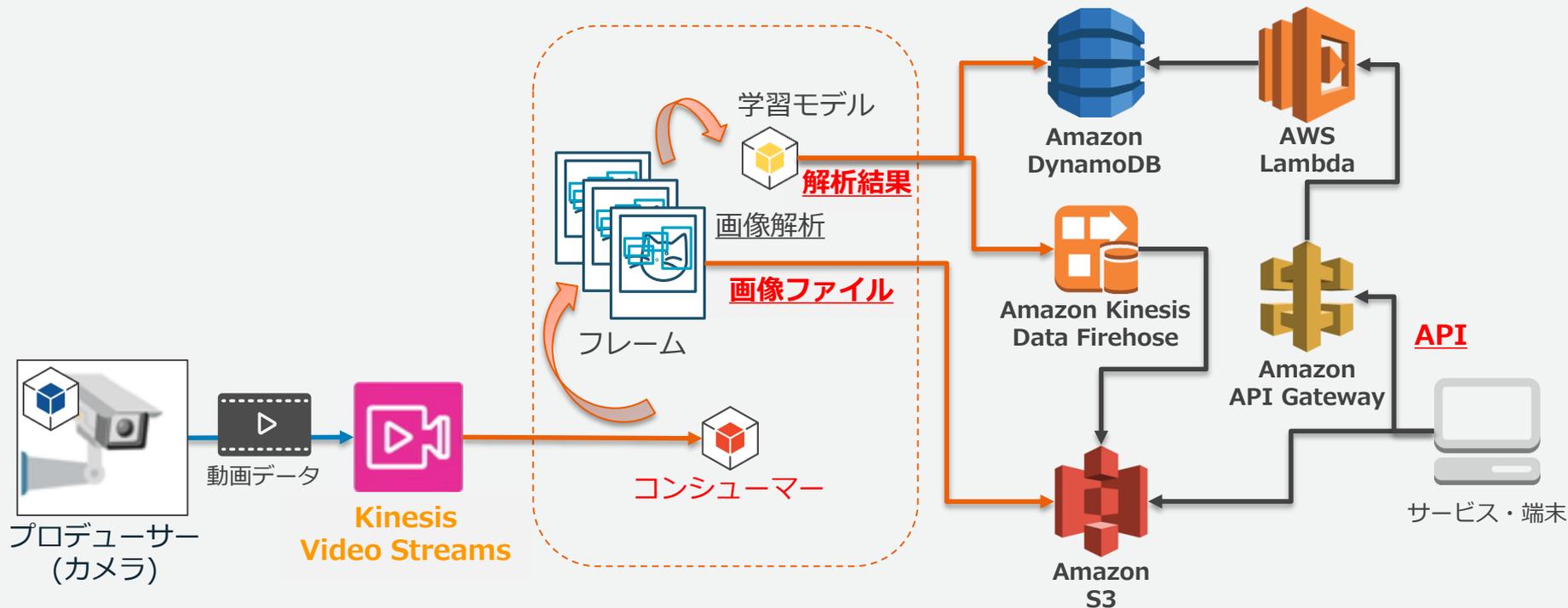
.github	Creating initial file from template	4 months ago
kinesis-video-gst-demo	validate pts in rtsp for vm support	20 days ago
kinesis-video-native-build	Release 1.3.0 (15th March 2018)	3 days ago
kinesis-video-pic	Release 1.3.0 (15th March 2018)	3 days ago
kinesis-video-producer-jni	Release 1.3.0 (15th March 2018)	3 days ago
kinesis-video-producer	Release 1.3.0 (15th March 2018)	3 days ago
.gitignore	Latest cumulative changes including:	3 months ago
LICENSE	Creating initial file from template	4 months ago
NOTICE	Creating initial file from template	4 months ago
README.md	Release 1.3.0 (15th March 2018)	3 days ago

README.md

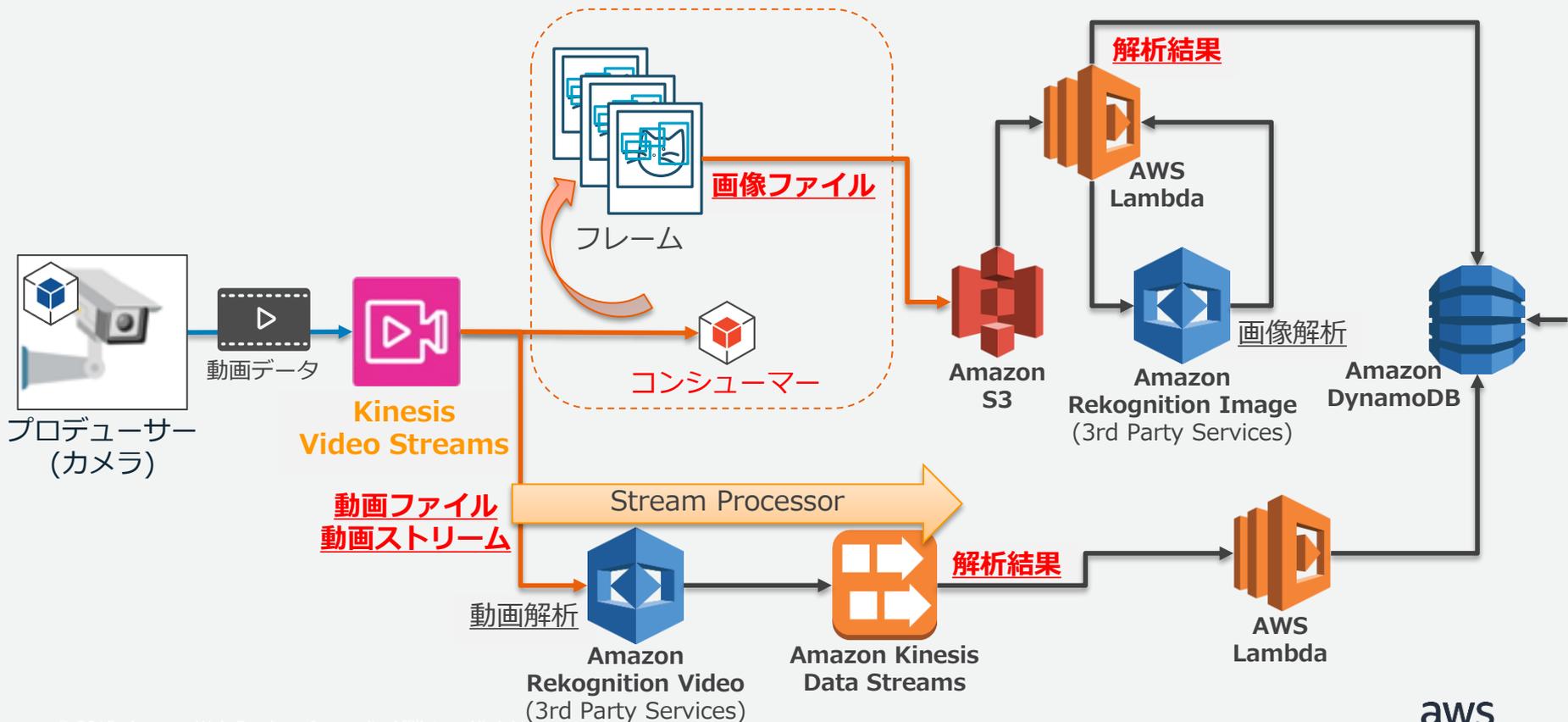
このセミナーの内容

- 📦 Amazon Kinesis Video Streams の特徴
- 📦 Amazon Kinesis Video Streams を動かすまで
 1. 動画データのフォーマット
 2. プロデューサー (カメラ側)
 3. コンシューマー (動画解析側)
 4. デモ
- 📦 Amazon Kinesis Video Streams を活用した構成例
- 📦 まとめ

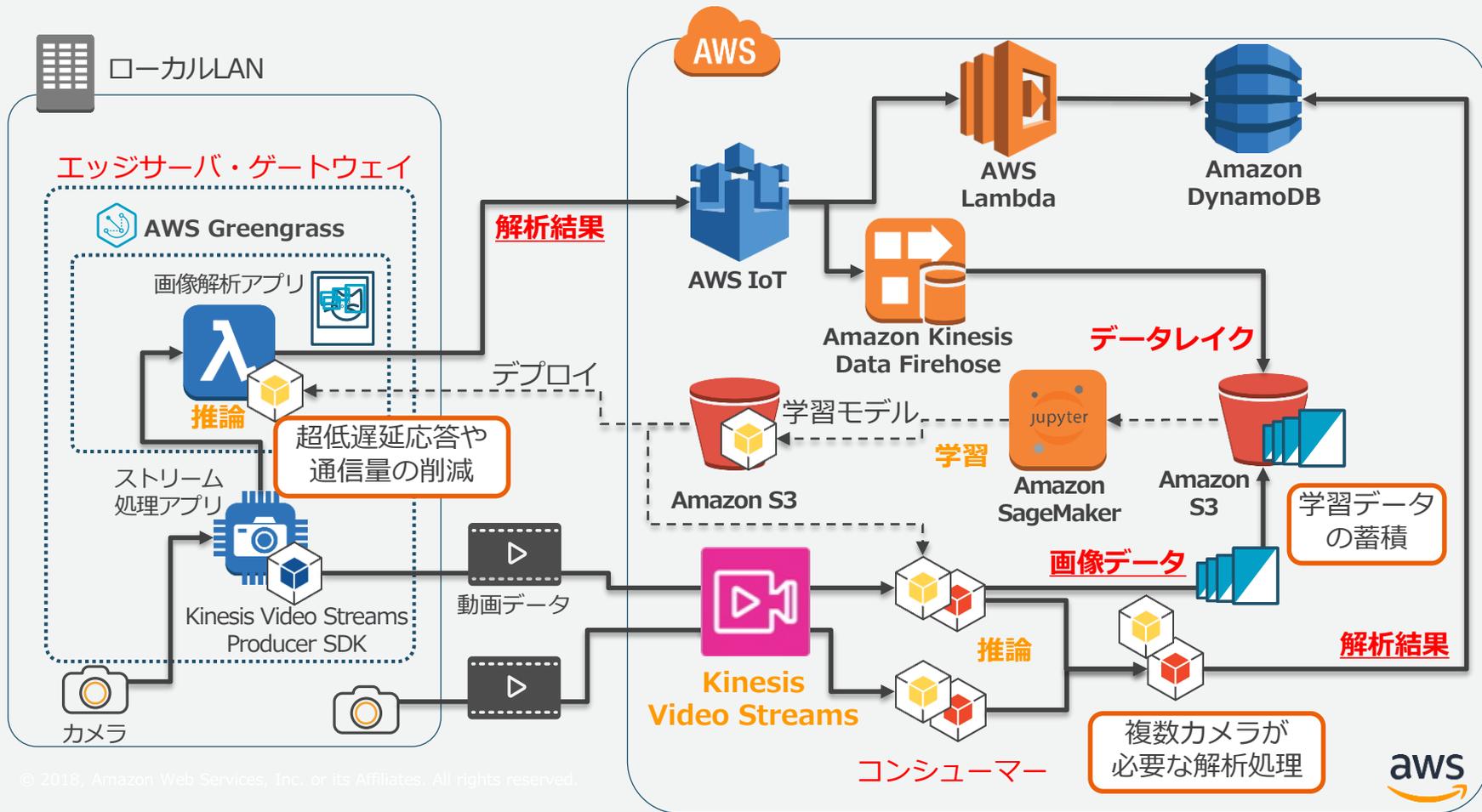
機械学習モデルや画像処理ライブラリで 画像解析した結果をAPIで公開する



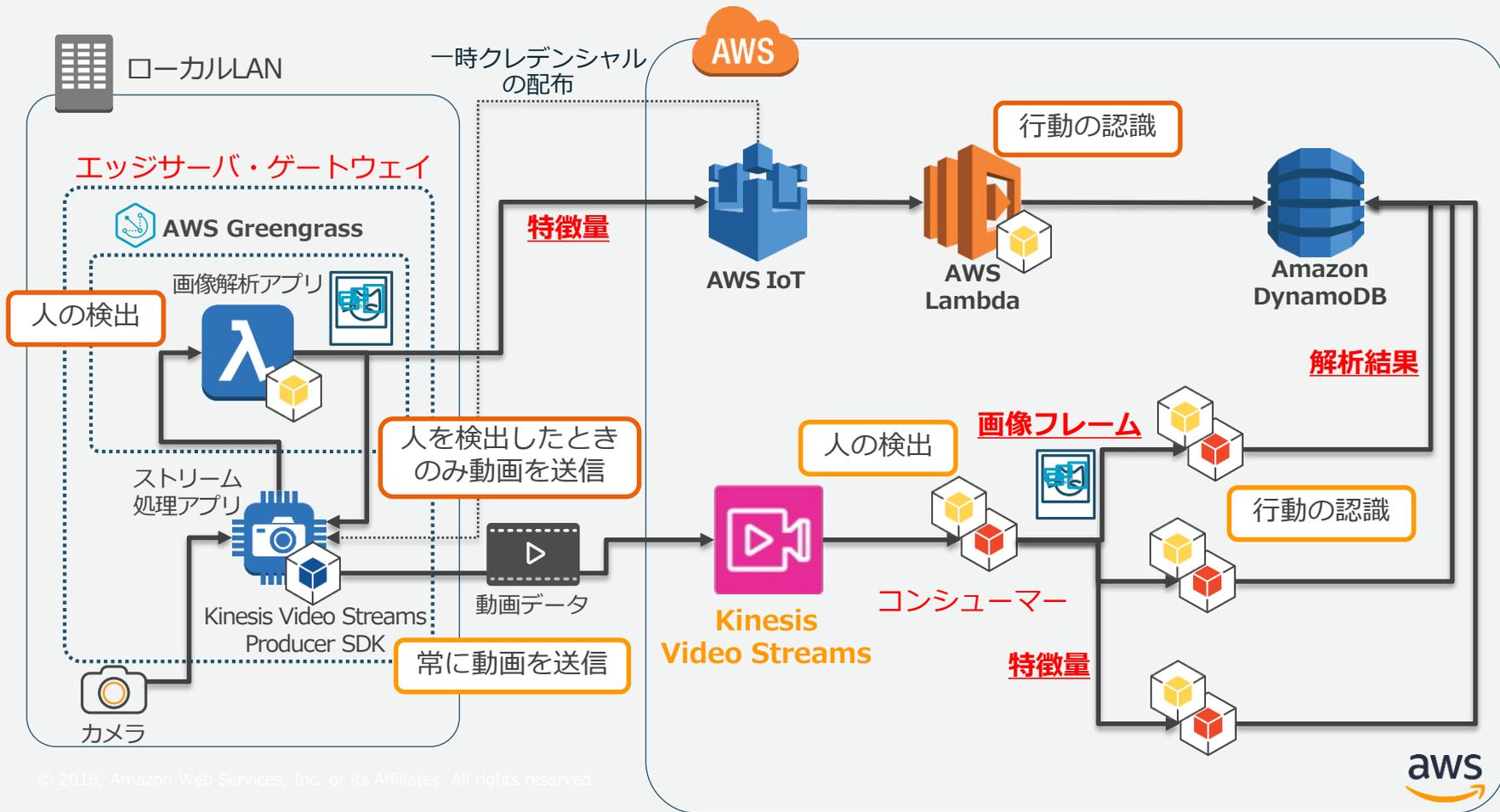
動画・画像解析サービスを使って分析する



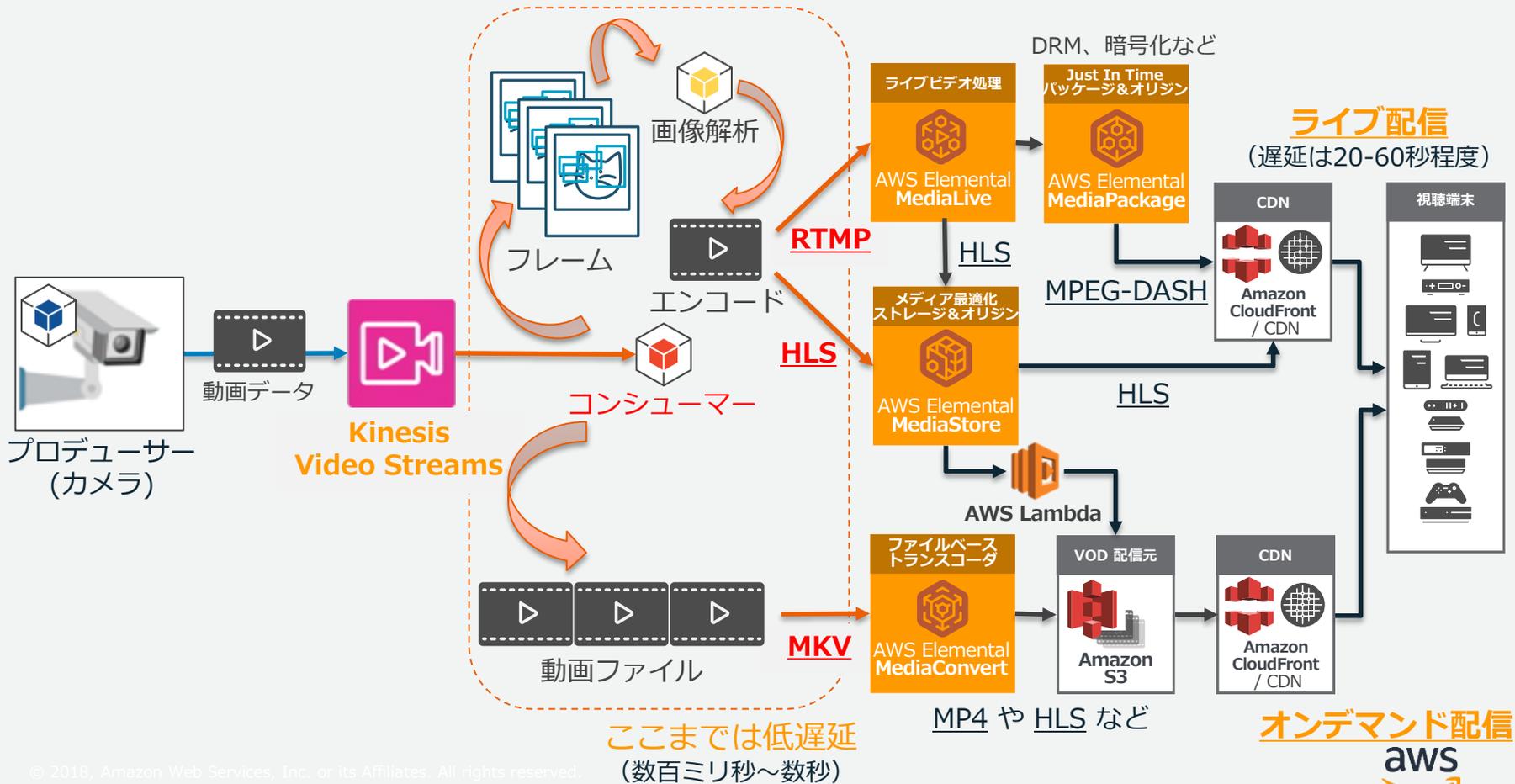
クラウドとエッジでの解析処理を使い分ける



検知と認識を分けて効率的にストリーム分析を行う



解析結果をレンダリングして動画を配信する

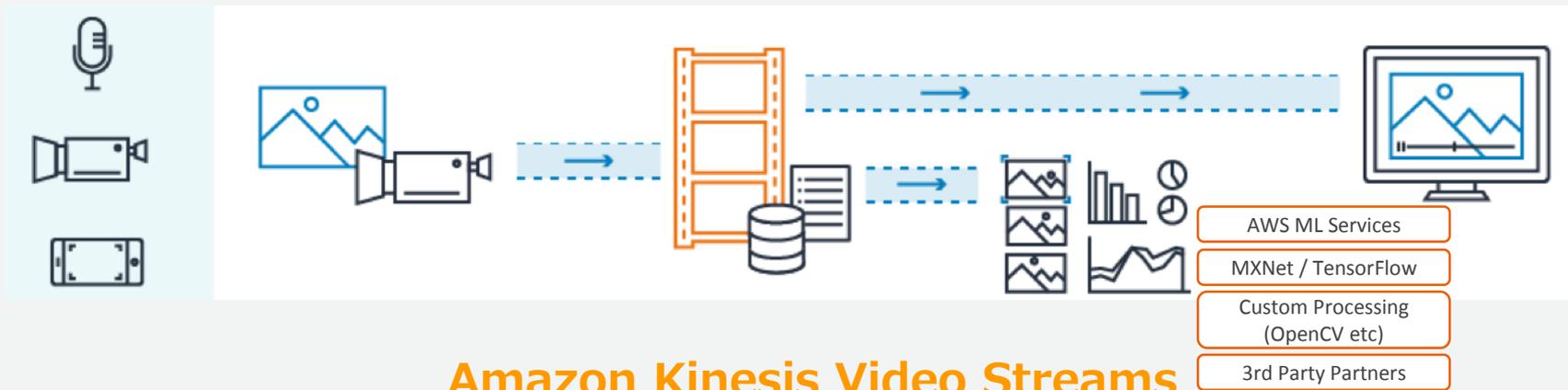


このセミナーの内容

- 📦 Amazon Kinesis Video Streams の特徴
- 📦 Amazon Kinesis Video Streams を動かすまで
 1. 動画データのフォーマット
 2. プロデューサー (カメラ側)
 3. コンシューマー (動画解析側)
 4. デモ
- 📦 Amazon Kinesis Video Streams を活用した構成例
- 📦 まとめ

Amazon Kinesis Video Streams

ビデオストリームをクラウドへ取り込み、
低遅延／オンデマンドで分析処理に配信するためのマネージドサービス



Amazon Kinesis Video Streams

デバイス プロデューサー

カメラ/マイクなど

動画の送信

ストリーム

動画の保存とインデックス化

コンシューマー

動画解析・加工・再配信／変換・再生

Kinesis Video Streams の特徴



何百万ものデバイスを接続してのストリーミング



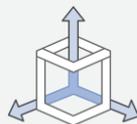
データの永続的な保存とインデックス化



動画解析には低遅延でのリアルタイム処理とバッチ処理
どちらのアプリケーションも構築可能



通信および保管データの暗号化と、
IAMでのアクセス管理によるセキュリティ



サーバレスなフルマネージドサービスとして提供

さいごに



Amazon Kinesis Video Streams で クラウド上に動画を取り込んで 安全でスケーラブルな動画解析を実現しましょう！

Amazon Kinesis Video Streams についてもっと知りたい方はこちら

- 公式ドキュメント - <https://aws.amazon.com/kinesis/video-streams/>
- ドキュメントやSDKなどのリソース - <https://aws.amazon.com/jp/kinesis/video-streams/resources/>
- Amazon Kinesis Video Streams Forum - <https://forums.aws.amazon.com/forum.jspa?forumID=282>

- C++ Producer SDK - <https://github.com/aws-labs/amazon-kinesis-video-streams-producer-sdk-cpp>
- Java Producer SDK - <https://github.com/aws-labs/amazon-kinesis-video-streams-producer-sdk-java>
- Android Producer Library - <https://github.com/aws-labs/aws-sdk-android-samples/tree/master/AmazonKinesisVideoDemoApp>
- Stream Parser Library - <https://github.com/aws/amazon-kinesis-video-streams-parser-library>

オンラインセミナー資料の配置場所

AWS クラウドサービス活用資料集

- <https://aws.amazon.com/jp/aws-jp-introduction/>

			
サービス別資料	ソリューション別資料	業種別資料	その他の資料
無料オンラインセミナー「Black Belt Online Seminar」のサービスカット資料他、AWSのTechメンバーによる各サービスの解説資料がご覧いただけます。	無料オンラインセミナー「Black Belt Online Seminar」のソリューションカット資料他、特定のソリューションについてのAWS活用方法がご覧いただけます。	無料オンラインセミナー「Black Belt Online Seminar」のインダストリーカット資料他、特定の業界のユースケースがご覧いただけます。	イベントに関する資料やアップデート情報などがご覧いただけます。

Amazon Web Services ブログ

- 最新の情報、セミナー中のQ&A等が掲載されています。
- <https://aws.amazon.com/jp/blogs/news/>

公式Twitter/Facebook AWSの最新情報をお届けします



@awscloud_jp



検索

もしくは

<http://on.fb.me/1vR8yWm>

最新技術情報、イベント情報、お役立ち情報、
お得なキャンペーン情報などを日々更新しています！

AWSの導入、お問い合わせのご相談

AWSクラウド導入に関するご質問、お見積、資料請求をご希望のお客様は以下のリンクよりお気軽にご相談下さい。

<https://aws.amazon.com/jp/contact-us/aws-sales/>

<p>お問い合わせ</p> <hr/> <p>日本担当チームへのお問い合わせ ></p> <hr/> <p>関連リンク</p> <p>フォーラム</p> <hr/>	<h2>日本担当チームへのお問い合わせ</h2> <p>AWS クラウド導入に関するご質問、お見積り、資料請求をご希望のお客様は、以下のフォームよりお気軽にご相談ください。平日営業時間内に日本オフィス担当者よりご連絡させていただきます。</p> <p>※ご請求金額またはアカウントに関する質問はこちらからお問い合わせください。</p> <p>※Amazon.com または Kindle のサポートにお問い合わせはこちらからお問い合わせください。</p> <p>アスタリスク (*) は必須情報となります。</p> <p>姓*</p> <input type="text"/> <p>名*</p> <input type="text"/>
---	--

※「AWS お問い合わせ」で検索して下さい。

AWS Well Architected 個別技術相談会お知らせ

- Well Architectedフレームワークに基づく数十個の質問項目を元に、お客様がAWS上で構築するシステムに潜むリスクやその回避方法をお伝えする個別相談会です。

<https://pages.awscloud.com/well-architected-consulting-2017Q4-jp.html>

- 参加無料
- 毎週火曜・木曜開催

The screenshot shows the registration page for the AWS Well Architected individual technical consultation. At the top, the AWS logo is displayed. Below it, the event title is "[10, 11, 12 月開催] AWS Well Architected 個別技術相談会". The main heading is "AWS 上で構築するシステムのリスクの把握・回避方法をご希望のお客様". The text explains that the consultation is for customers using AWS and aims to address risks in system architecture. It lists five key areas: Security, Reliability, Performance, Cost, and Support. A registration form is visible, with fields for name, email, and a table for dates and times. The dates listed are 10/10, 10/11, and 10/12, with times from 13:00 to 14:00. The location is the Amazon Web Services Tokyo Office.

aws

[10, 11, 12 月開催] AWS Well Architected 個別技術相談会

AWS 上で構築するシステムのリスクの把握・回避方法をご希望のお客様

この度 AWS をご活用頂いているお客様を対象に「AWS Well Architected 個別技術相談会」を開催致します。

Well Architected 個別技術相談会では、リスクの把握・回避を目的として、セキュリティ・信頼性・パフォーマンス・コスト・運用の5つの観点で、お客様の AWS 活用状況や構成についてお伺いします。AWS のベストプラクティスに基づき作成された Well Architected フレームワークを元に、今までお客様が気づきでなかったリスクやAWS活用の改善点を見つけることができます。例えば、自動車においては納車前点検、車検を定期的に行うのと同様に、本相談会はおお客様の AWS 上のシステムをよりよく活用頂くことを目的としております。

Well Architected 個別技術相談会にご参加頂くには、本ページにてお申込み後、弊社担当者からお送りするヒアリングシートにご記入・担当者にご送付頂く必要があります。その内容を元に、当日の相談会では AWS のソリューションアーキテクトと共に技術的なディスカッションをさせて頂きます。その他にも個別にご相談内容があれば、こちらもご相談を承りますので、是非お気軽にご参加ください。

日時	毎週火曜、木曜開催 ※詳細の日程、時間帯は、以下の日時をご参照の上、お申し込みください。(90分/1社)
開催場所	アマゾン ウェブ サービス東京オフィス

下記のフォームよりお申込みください。

・姓:

・名:

・Eメールアドレス: