【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　山﨑 翔太

2018.03.28

自己紹介

山﨑 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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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います。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.

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Amazon Kinesis プラットフォーム

New

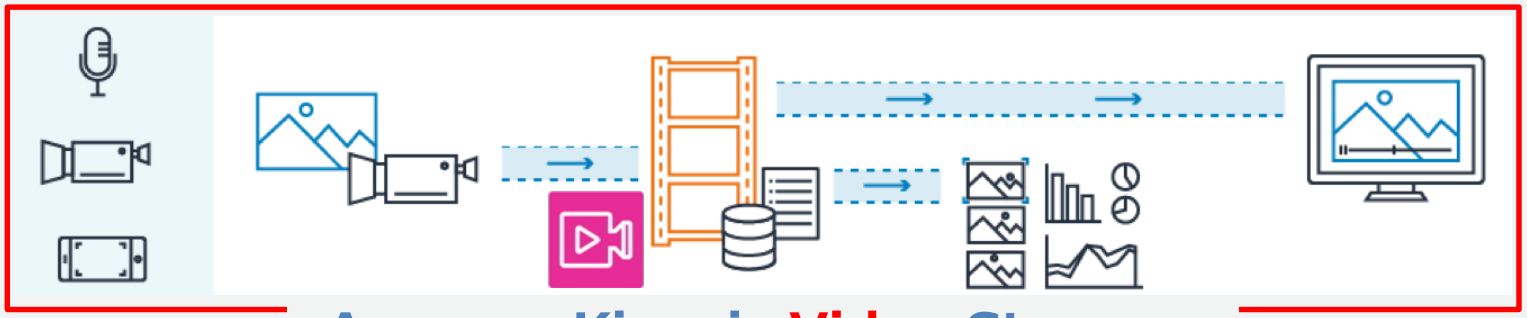
ストリームデータを収集・処理するためのフルマネージドサービス群

Amazon Kinesis
Data Streams

Amazon Kinesis
Data Firehose

Amazon Kinesis
Data Analytics

New



Amazon Kinesis Video Streams

© 2018, Amazon Web Services, Inc. or its

aws

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）**

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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	ストリーム	コンシューマー
カメラ/マイクなど　動画の送信	動画の保存とインデックス化	動画解析・加工・再配信／変換・再生

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

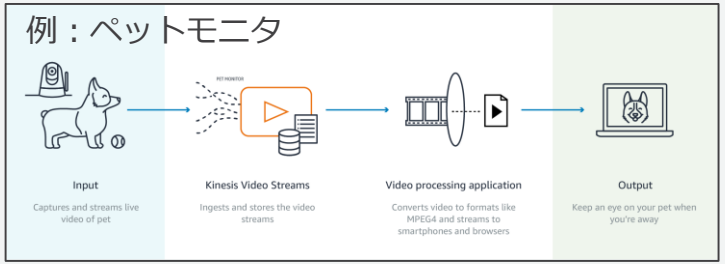
※ データ構造については後述します

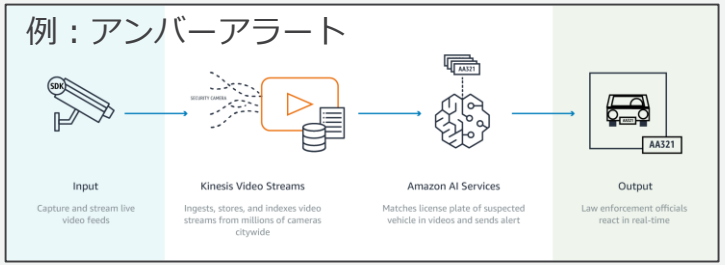
© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

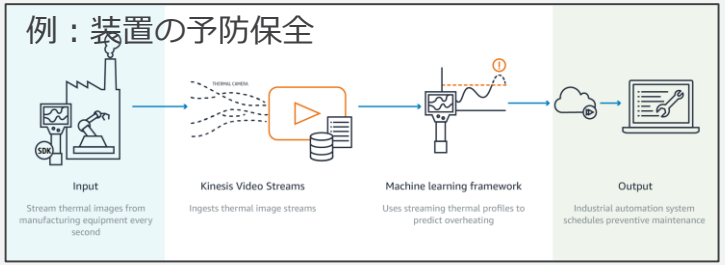
aws

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







© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

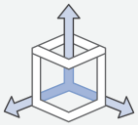
Kinesis Video Streams の特徴

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

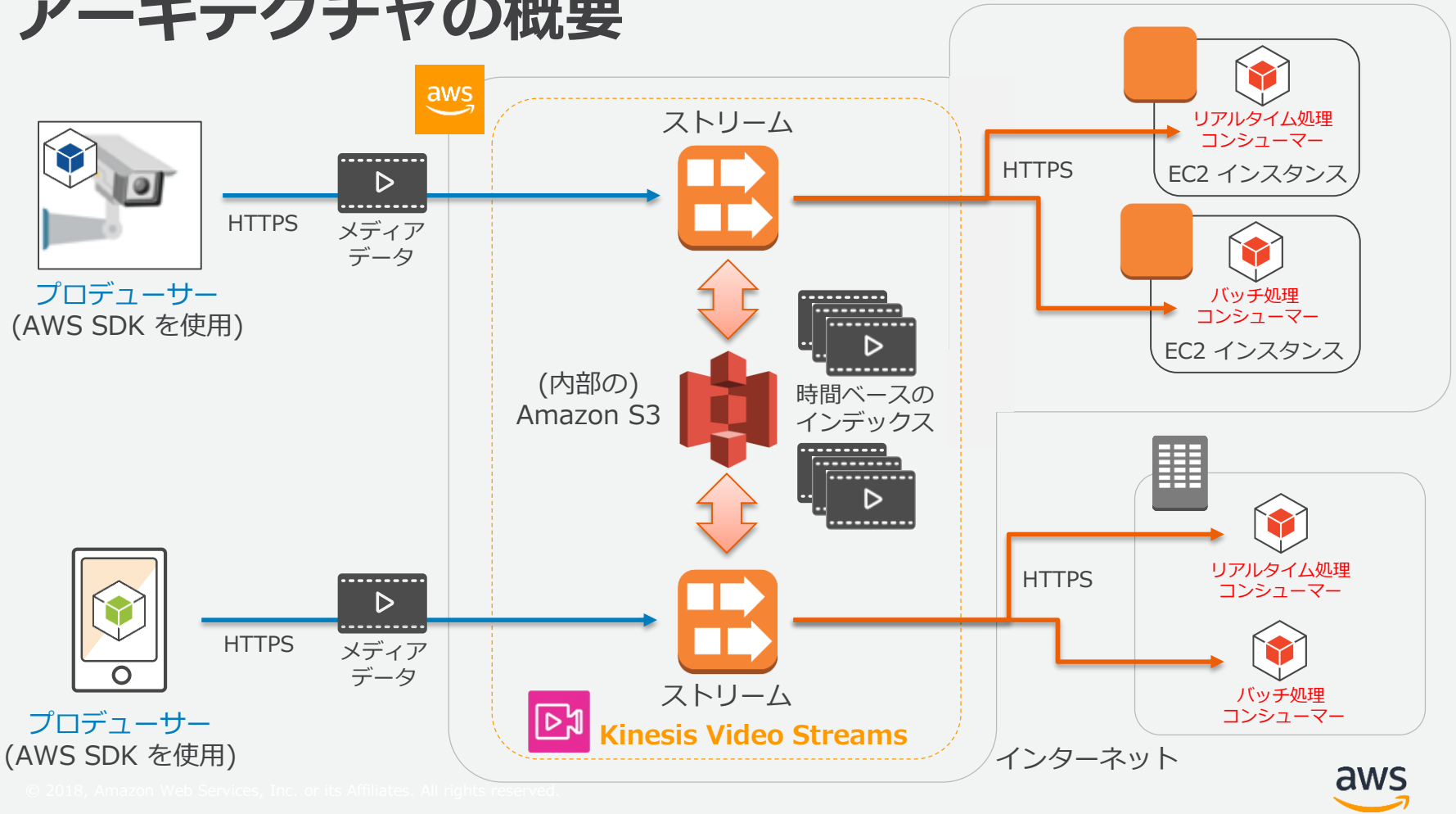
通信および保管データの暗号化と、
IAMでのアクセス管理によるセキュリティ



サーバレスなフルマネージドサービスとして提供

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

アーキテクチャの概要



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

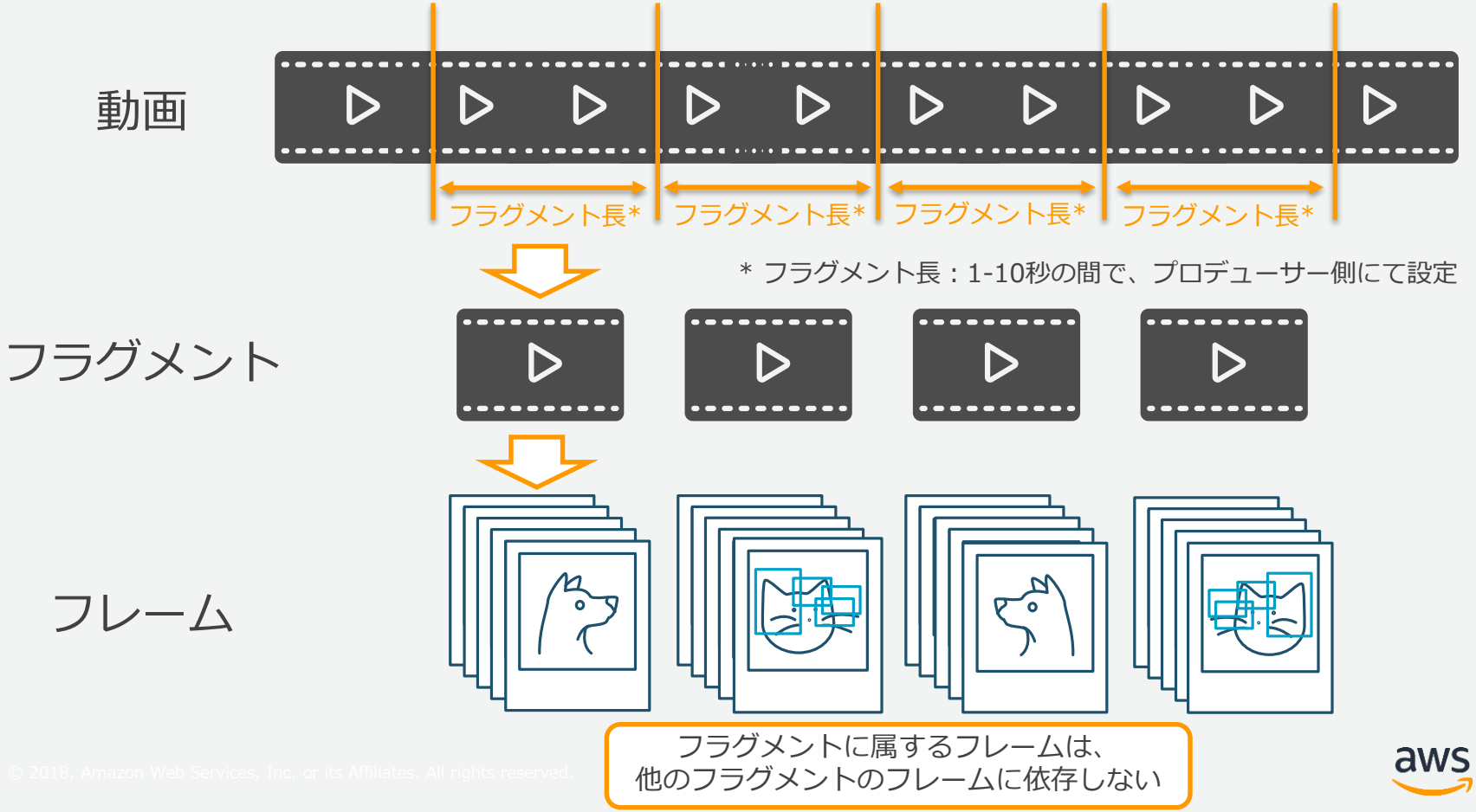
主要なコンセプト

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。

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

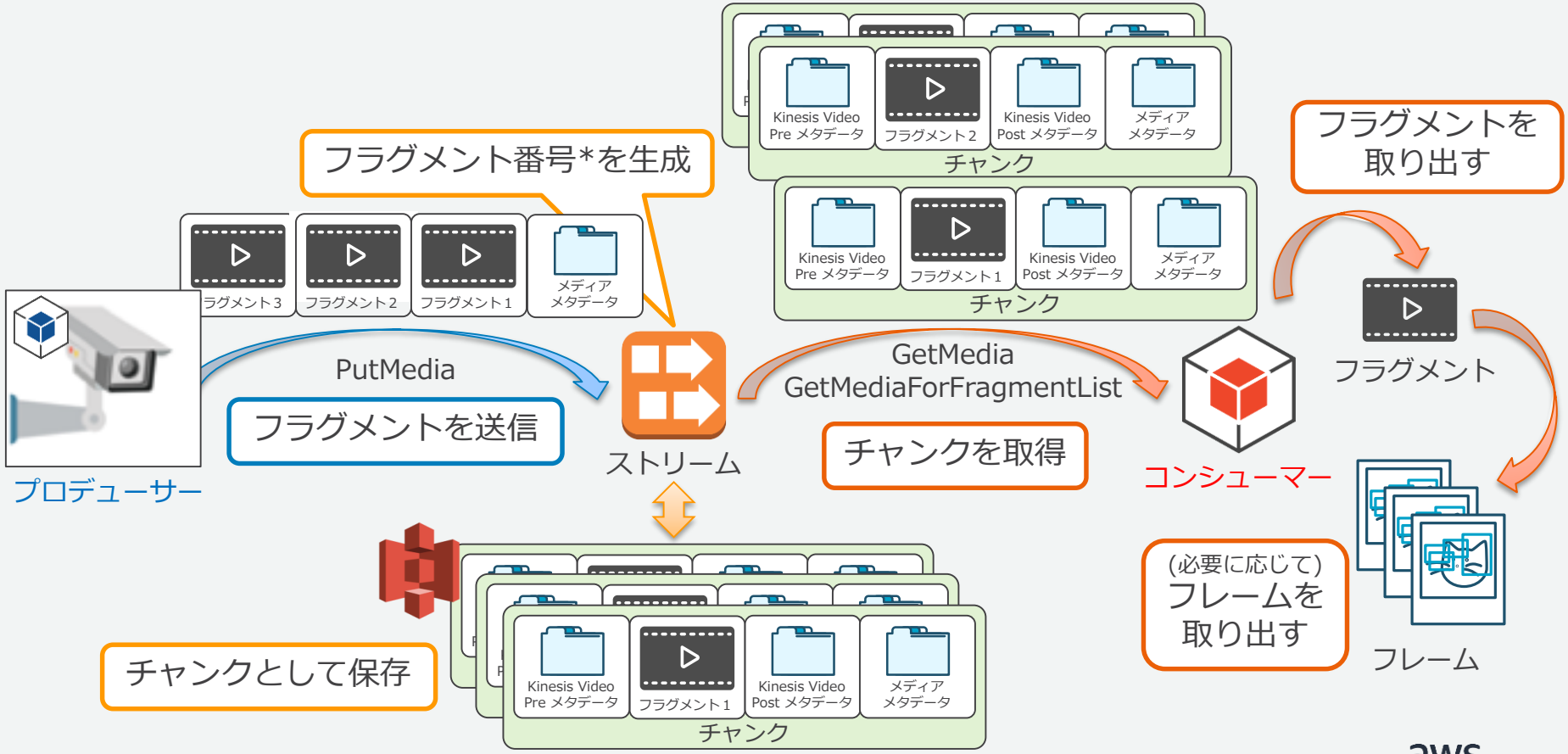
aws

フラグメントとフレーム



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

主要なコンセプトの関係性



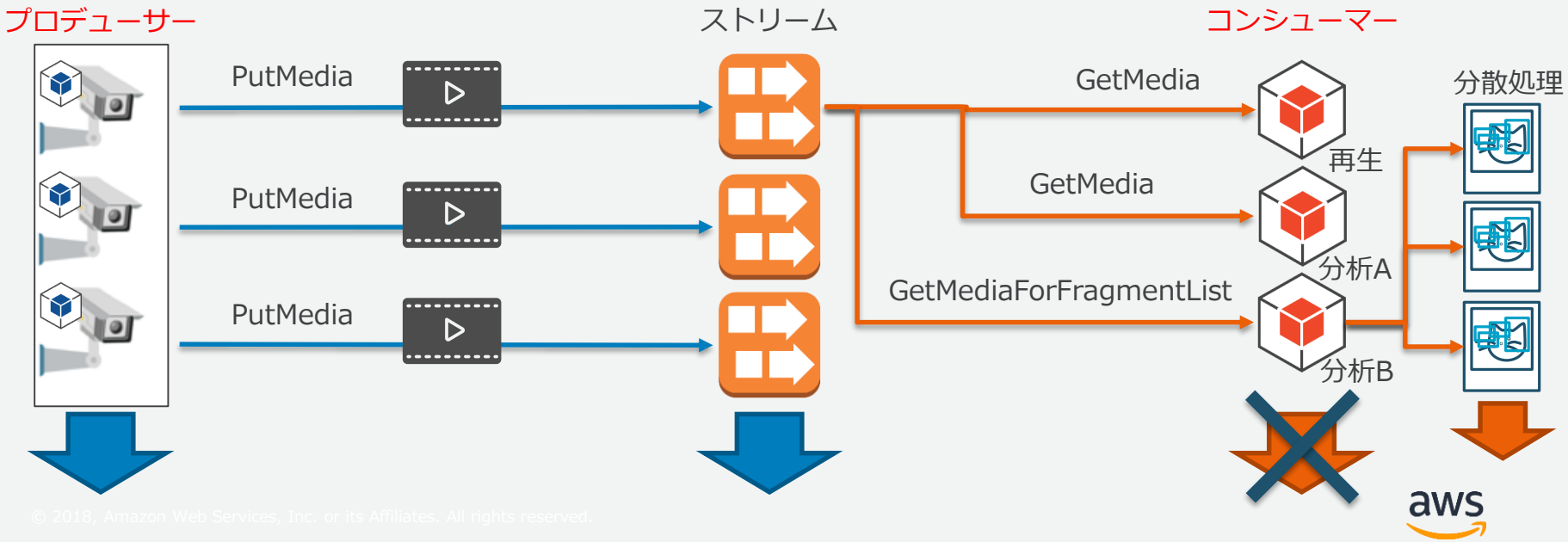
© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

*フラグメント番号：64ビット整数で表される識別子で、Kinesis Video Streams が受信時に生成してフラグメントに割り当てる

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**



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

再生時刻の指定

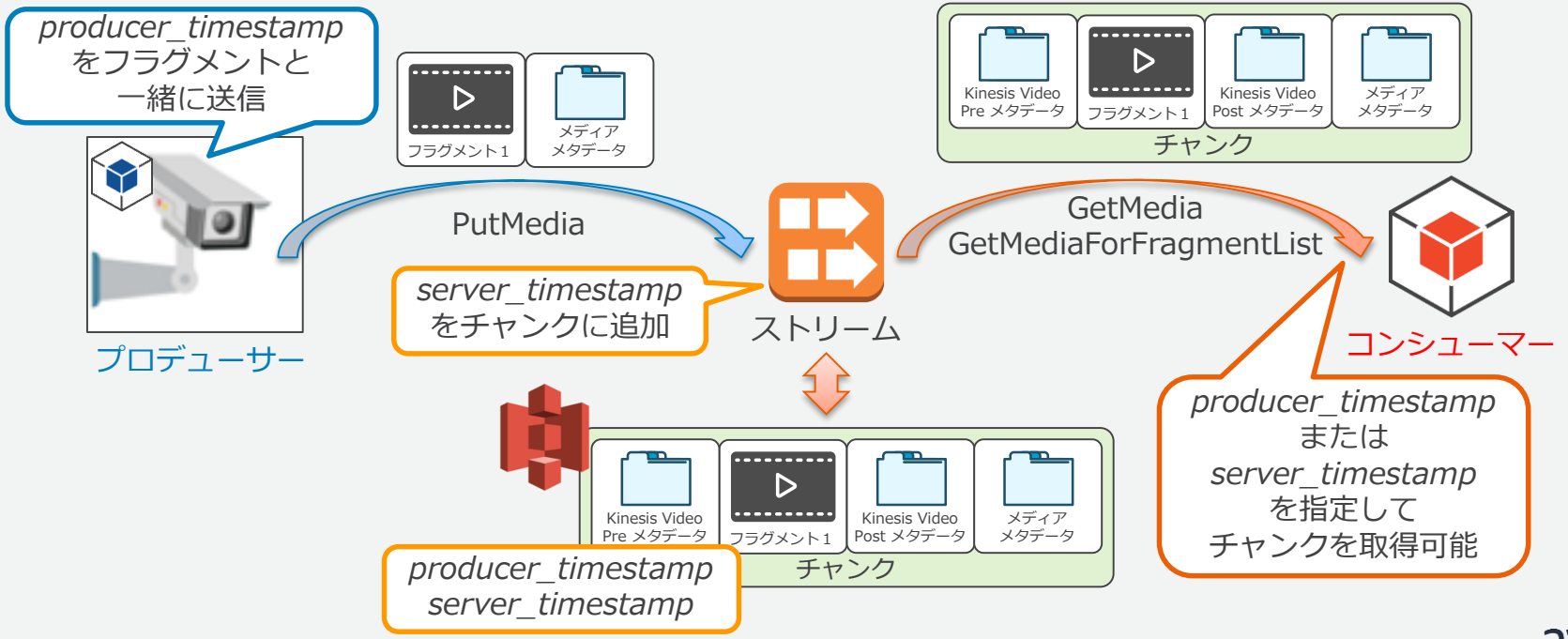
動画の情報

現在時刻からの遅延

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights re

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



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

https://docs.aws.amazon.com/ja_jp/kinesisvideostreams/latest/dg/API_dataplane_PutMedia.html

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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*"

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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）

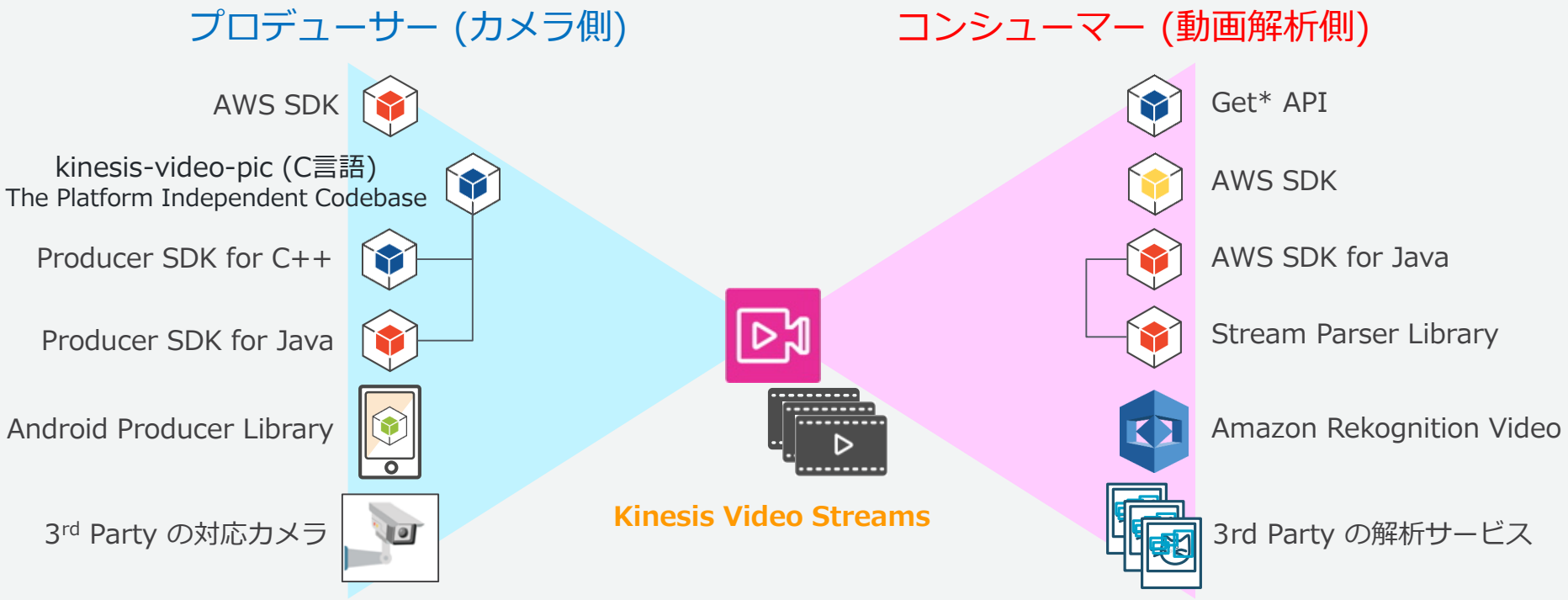
メトリクス	説明
PutMedia.Requests	ストリームへの PutMedia API のリクエスト数
PutMedia.IncomingBytes	PutMedia によってストリームが受信したデータのバイト数
PutMedia.FragmentIngestionLatency	フラグメントの最初と最後のバイトをストリームが受信した時間の差（フラグメントを受信する処理のレイテンシ）
PutMedia.Success	書き込みに成功したフラグメントを1、失敗したフラグメントを0とした平均値（フラグメントの受信成功率を表す指標）
GetMedia.Requests	ストリームへの GetMedia 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）

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved

https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/akac-metricscollected.html

aws

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



https://docs.aws.amazon.com/ja_jp/kinesisvideostreams/latest/dg/producer-sdk.html
https://docs.aws.amazon.com/ja_jp/kinesisvideostreams/latest/dg/parser-library.html

Kinesis Video Streams の料金構成

東京リージョンの場合

料金構成	料金
（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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

https://aws.amazon.com/jp/kinesis/video-streams/pricing/

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

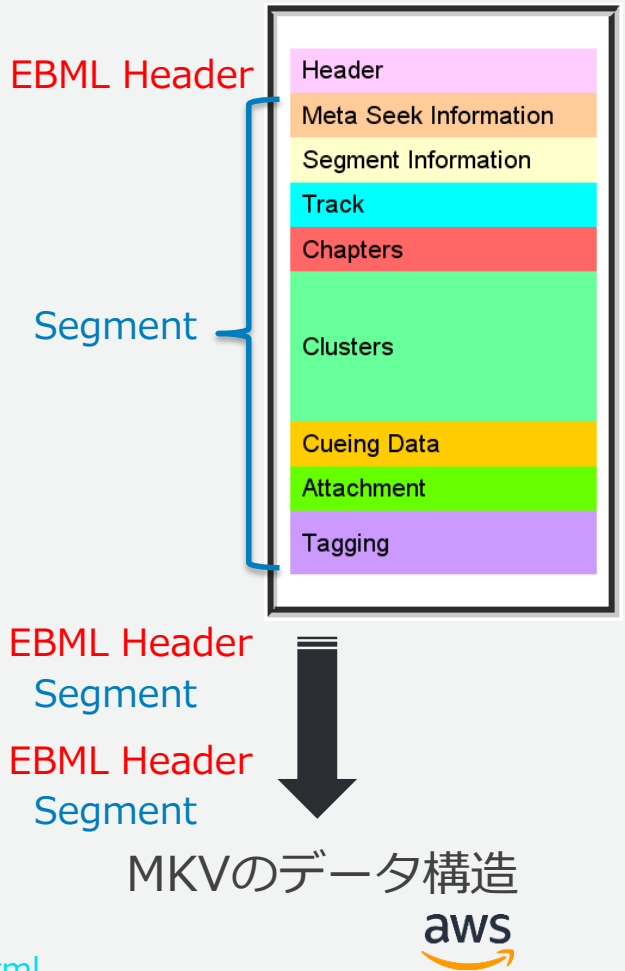
© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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**



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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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
```

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

https://docs.aws.amazon.com/ja_jp/kinesisvideostreams/latest/dg/API_dataplane_PutMedia.html

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 *producer_timestamp* 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）

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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フラグラメント
  - 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）

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

https://docs.aws.amazon.com/ja_jp/kinesisvideostreams/latest/dg/limits.html

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

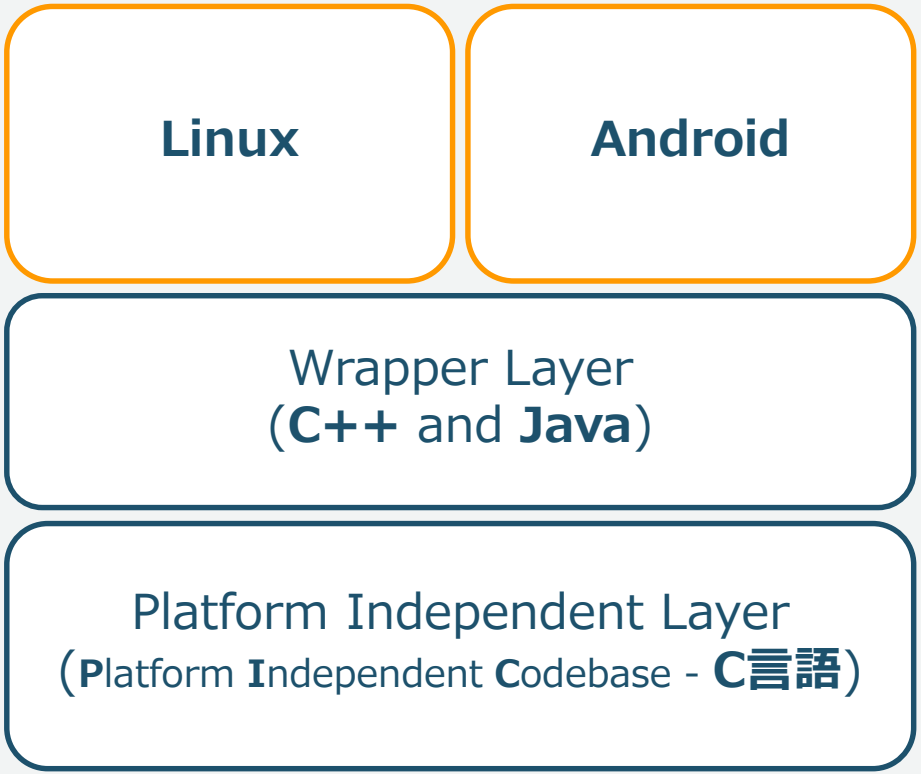
デバイスのメディアパイプラインを、Kinesis Video Streams のストリームに統合するための機能を一気通貫で提供

- 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

https://docs.aws.amazon.com/ja_jp/kinesisvideostreams/latest/dg/producer-sdk.html
https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp

aws

Producer SDK の構成



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

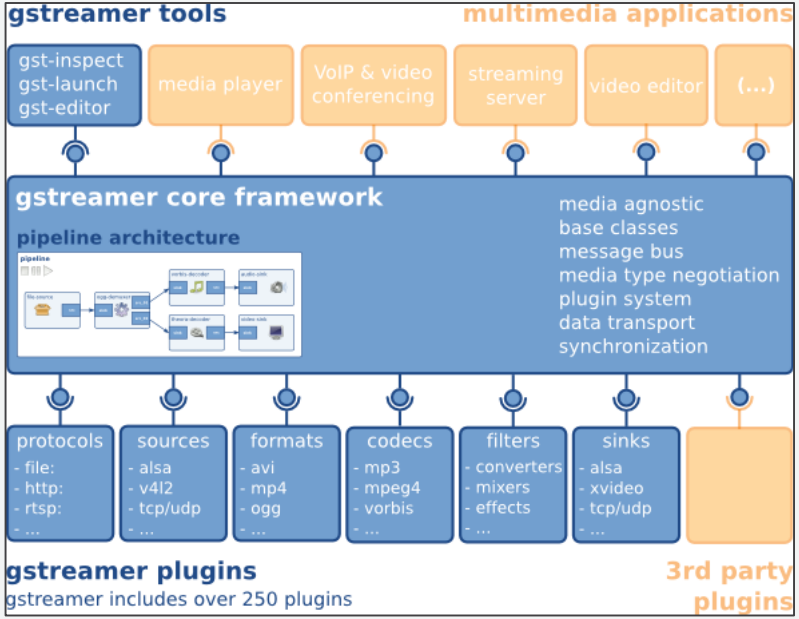
コア機能を提供するライブラリ
- （カメラ）メーカー向け
- 様々なハードウェア環境に合わせて、ファームウェアレベルで動画ソースとの統合を実装可能
- 他のライブラリとは完全に独立

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

https://gstreamer.freedesktop.org/

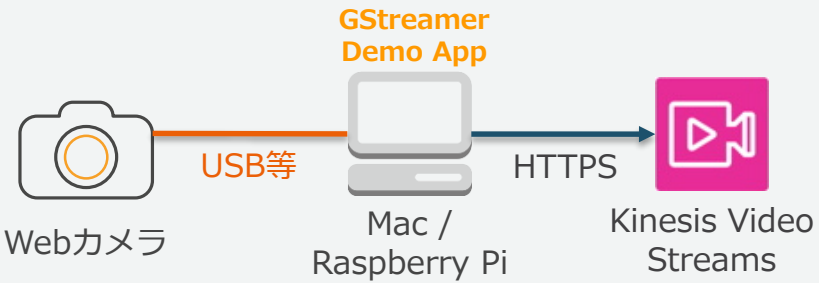
GStreamer のデモアプリケーション

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

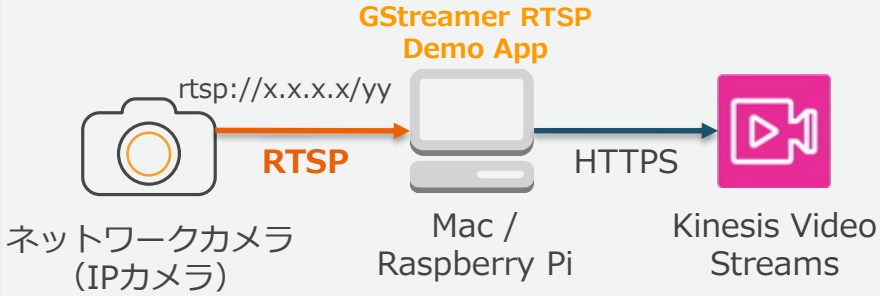


© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

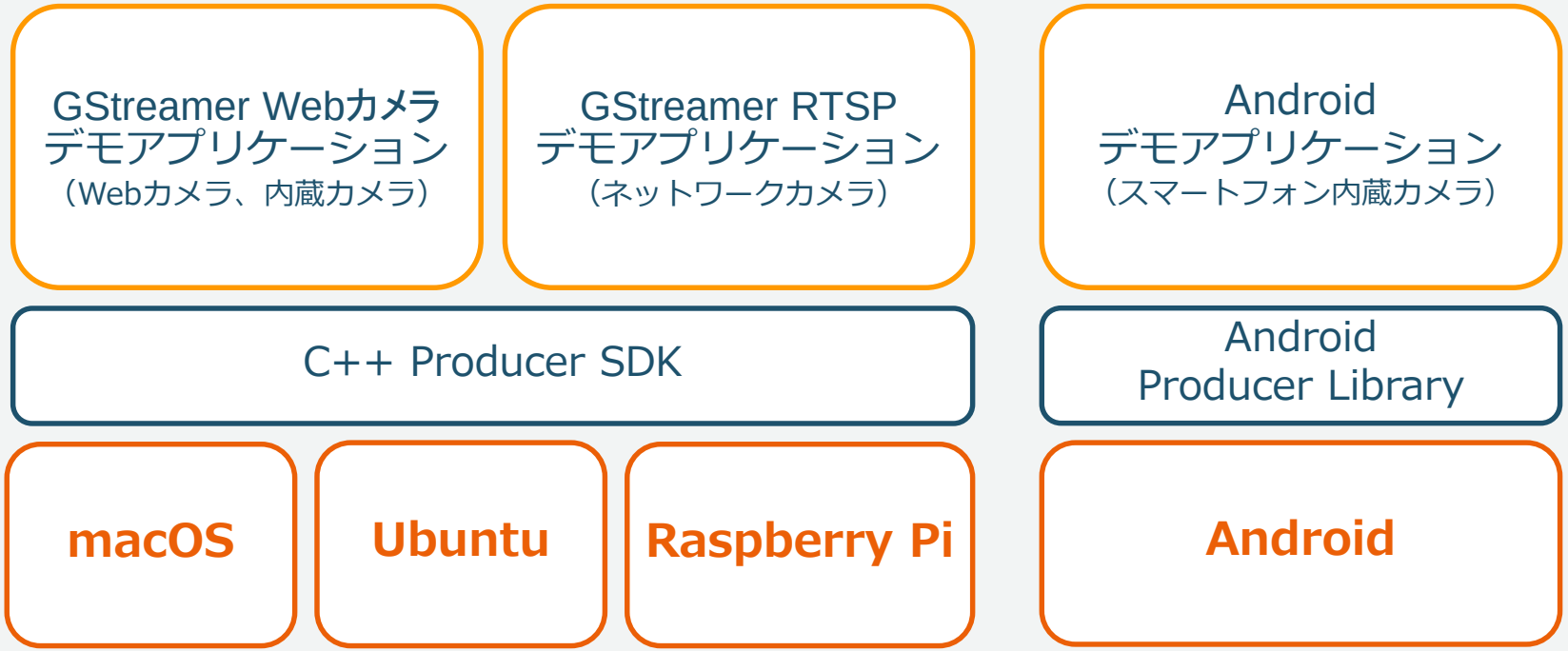
※現在は、Windows システム上では動作しません

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

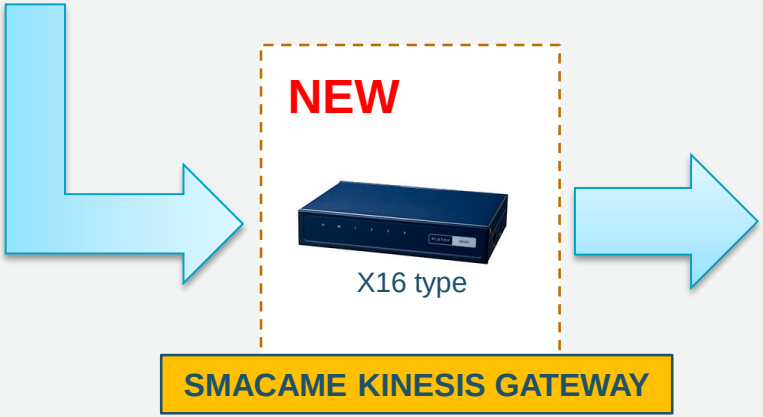


C++ Producer SDK：https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp
Android Producer App：https://github.com/awslabs/aws-sdk-android-samples/tree/master/AmazonKinesisVideoDemoApp

aws

パートナーの対応カメラ

*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



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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

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
```

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

https://docs.aws.amazon.com/ja_jp/kinesisvideostreams/latest/dg/API_dataplane_GetMedia.html

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）

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

GetMedia API の注意点

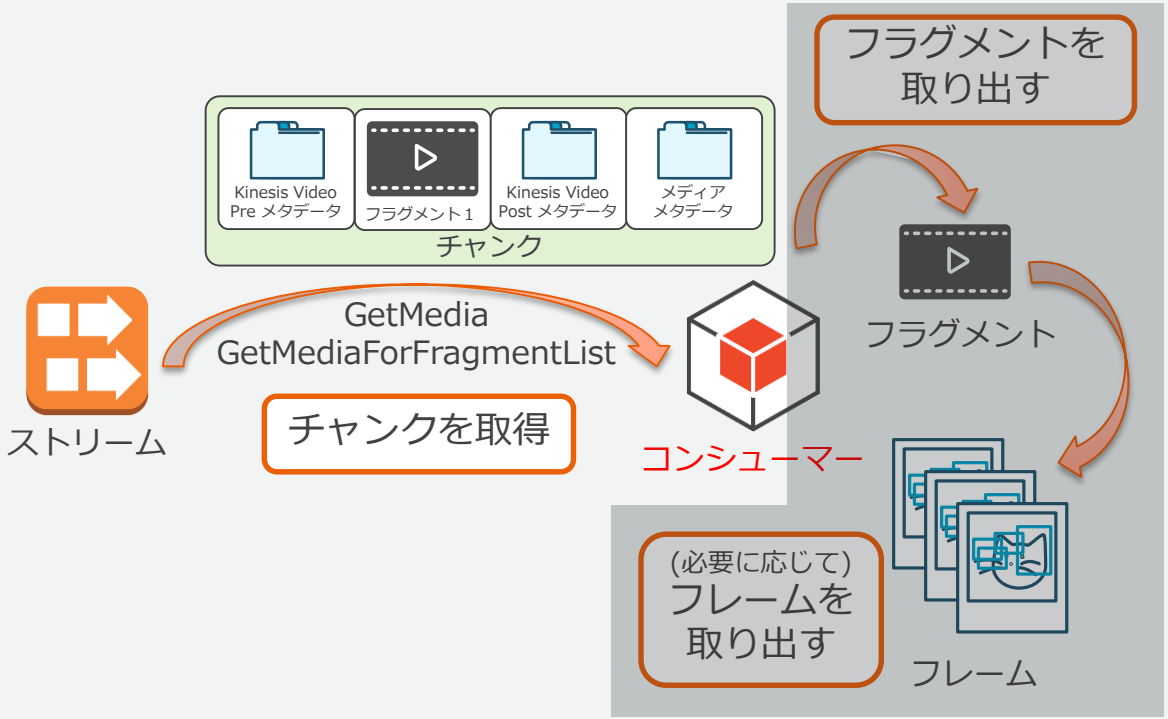
- GetMedia APIを使用する前に、GetDataEndpoint APIを呼び出してエンドポイントを取得する必要がある

- 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）

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

https://docs.aws.amazon.com/ja_jp/kinesisvideostreams/latest/dg/limits.html

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



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

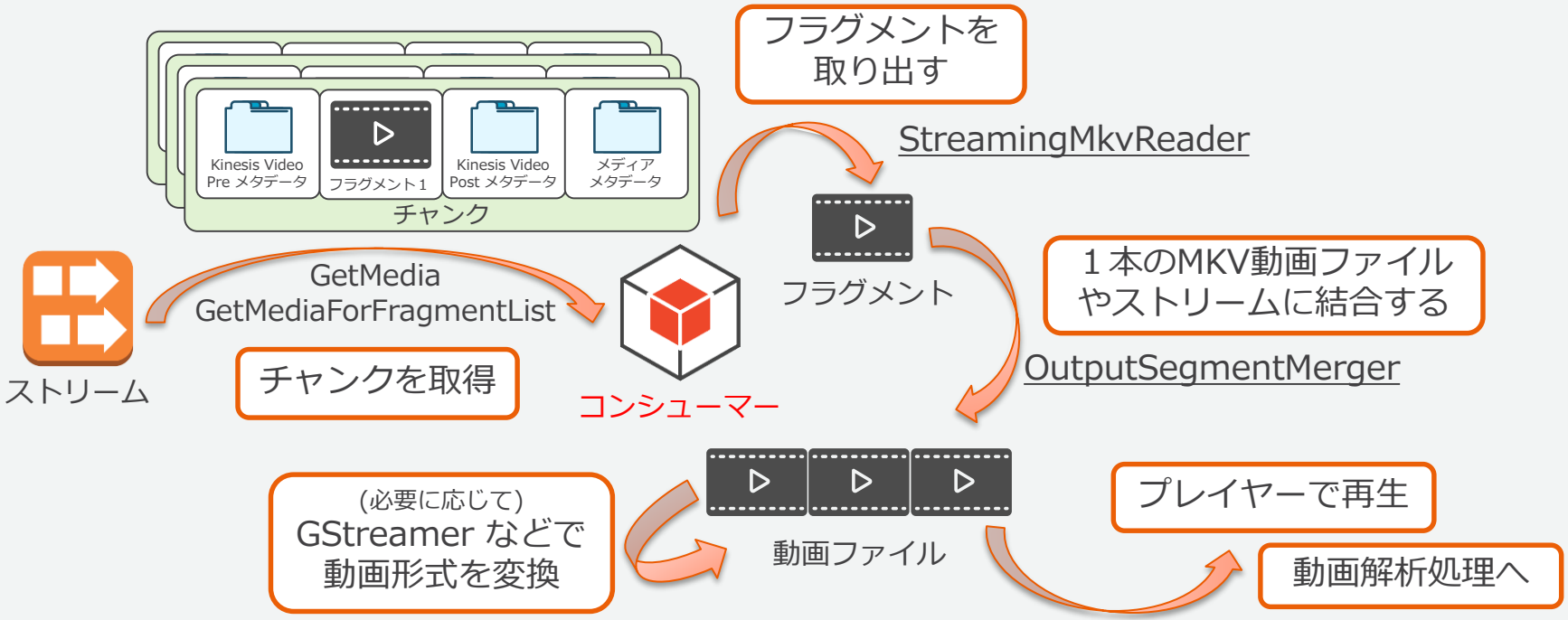
主なツール	内容
StreamingMkvReader	MKVデータをストリームから MKV Element として読み取る
FragmentMetadataVisitor	フラグメント（メディアデータ）およびトラック（コーデックの種類やピクセル幅・高さなどを含むデータ）からメタデータを取得する。 フラグメント番号やタイムスタンプの情報もここから取得出来る。
OutputSegmentMerger	異なるトラックのメタデータを単一のセグメントを持つストリームにマージして、連続したフラグメント（チャンク）を結合する
FrameVisitor	フラグメントからフレームを取り出してフレーム毎の処理に渡す。 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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

https://github.com/aws/amazon-kinesis-video-streams-parser-library

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）

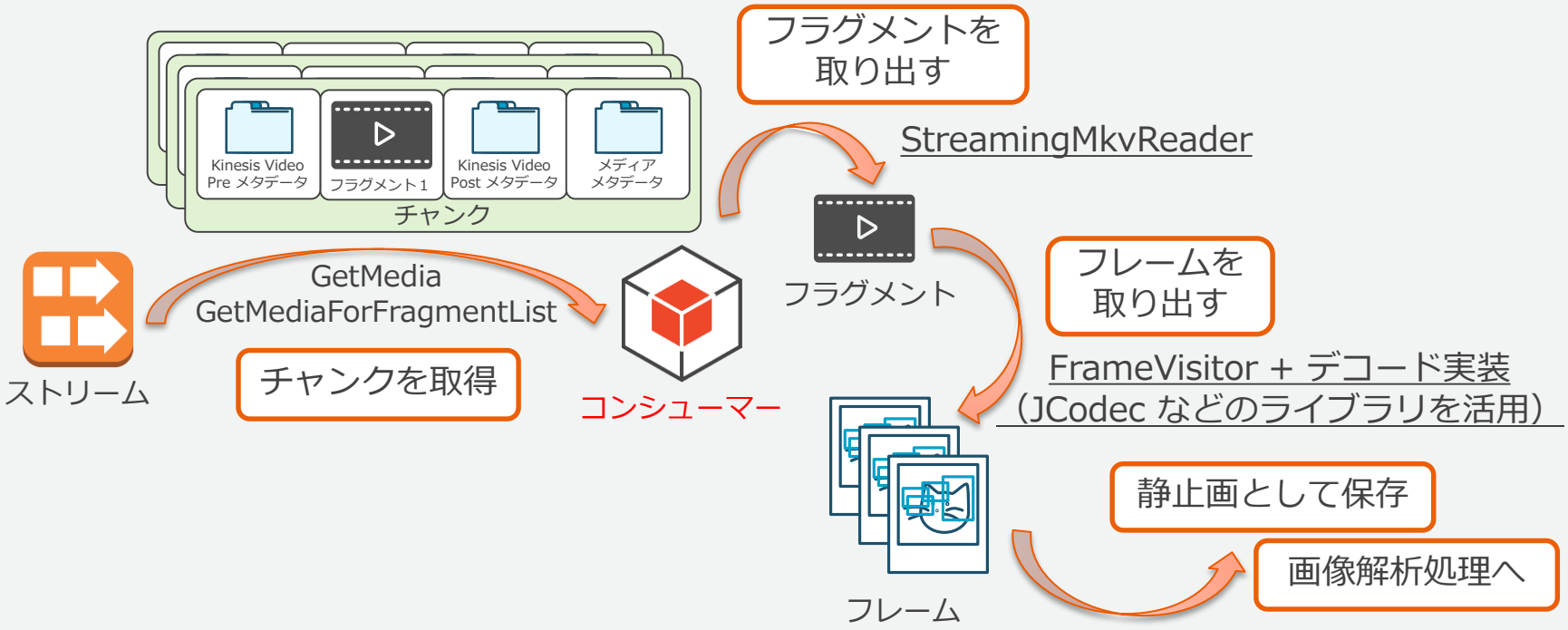
```
// Get media request loop.
while (true) {
        GetMediaResult result = mediaClient.getMedia(getMediaRequest());

        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 (MkvTypeInfos.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



JCodec : http://jcodec.org/
OpenCV : https://opencv.org/

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

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・・・

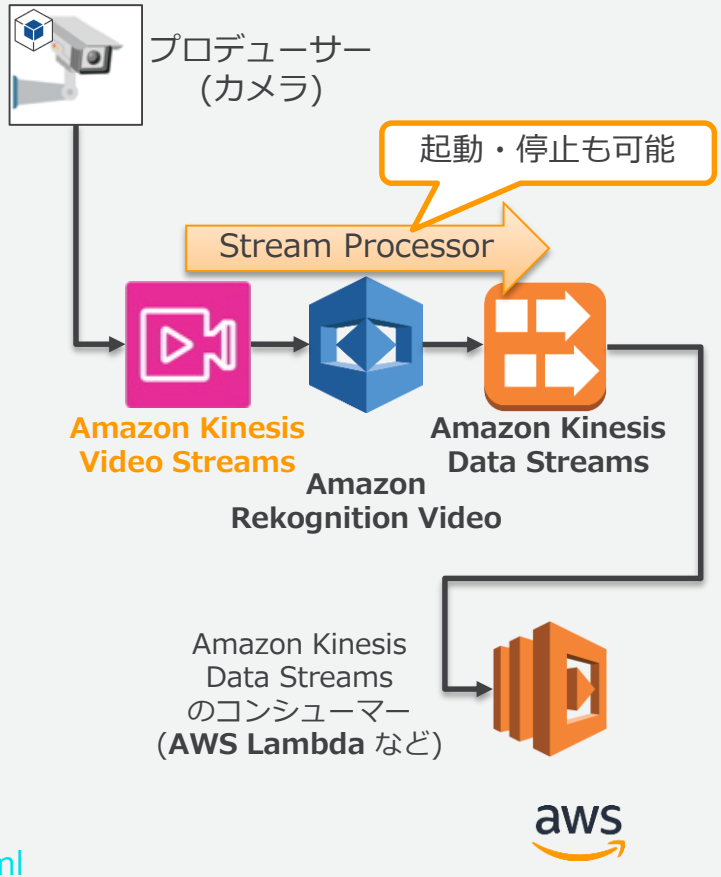
OpenCV : https://opencv.org/
OpenPose : https://github.com/CMU-Perceptual-Computing-Lab/openpose

© 2018, Amazon Web Services, Inc. All rights reserved

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

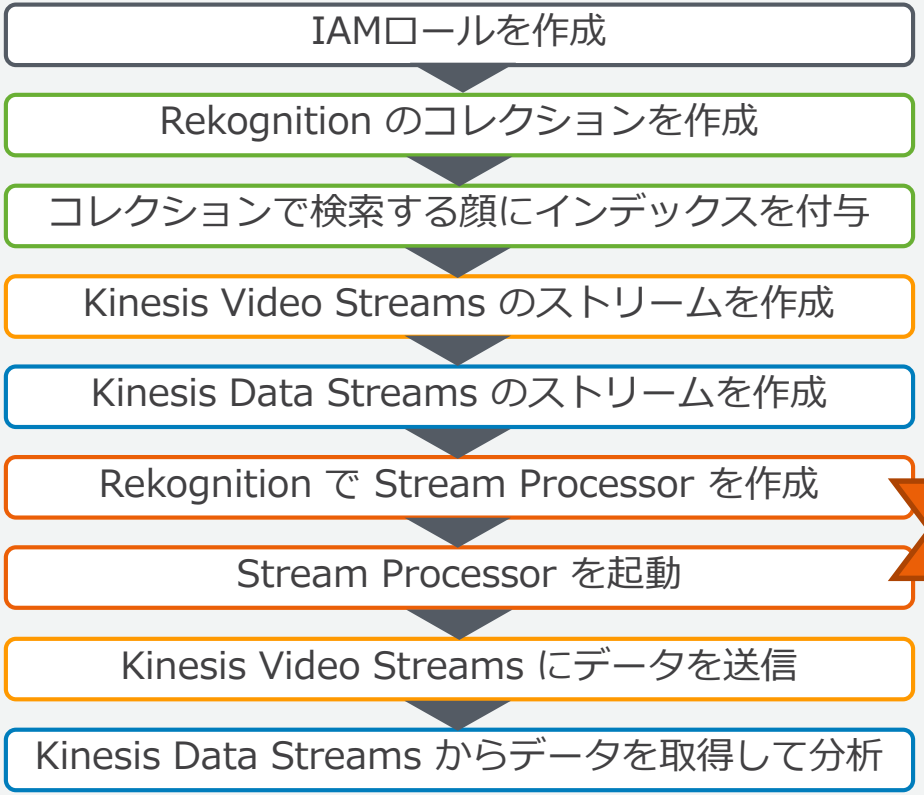
- Rekognition Video には、動画ストリームの分析を管理するための Stream Processor が用意されている
- 分析結果は、Rekognition Video から、Amazon Kinesis Data Streams のストリームに出力される



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

https://docs.aws.amazon.com/rekognition/latest/dg/streaming-video.html

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/XXXXXXXXXXXXX"}}' ¥
  --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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

https://docs.aws.amazon.com/ja_jp/rekognition/latest/dg/recognize-faces-in-a-video-stream.html

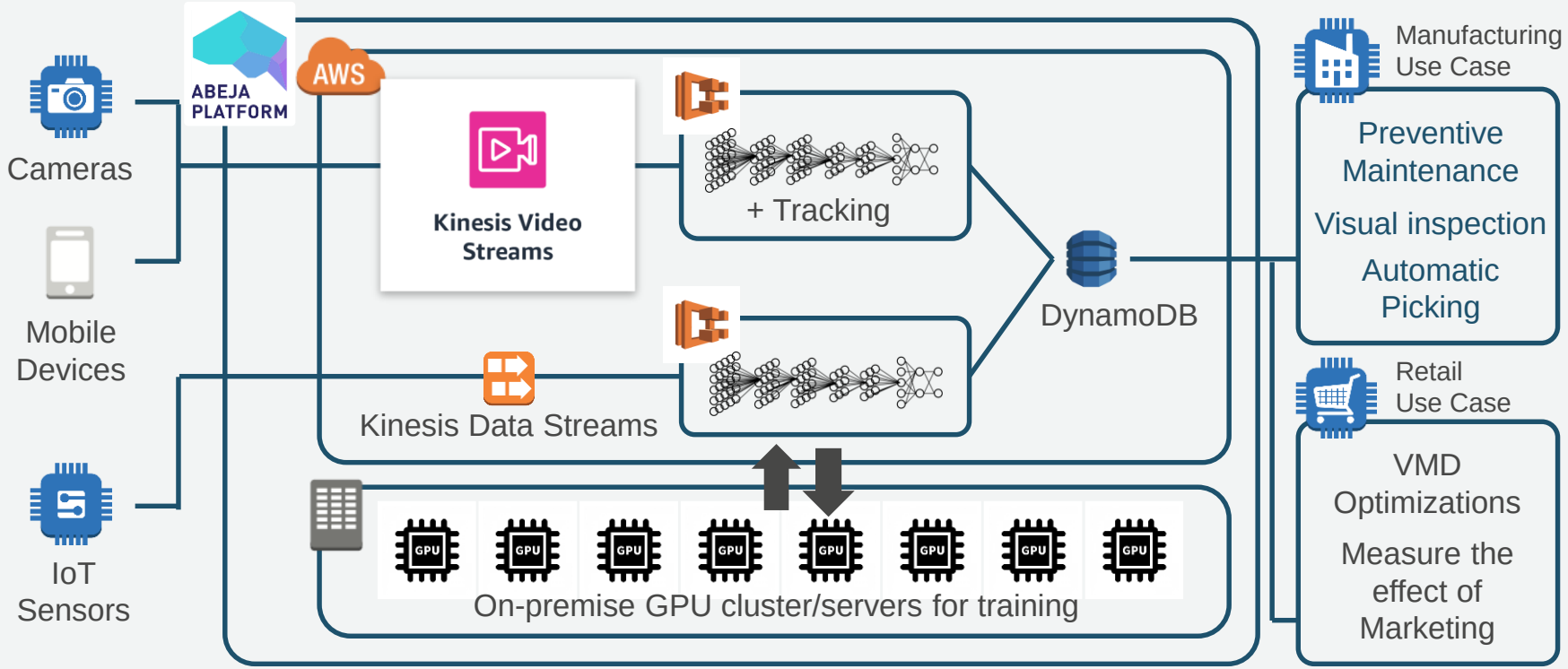
Rekognition Video による顔認識の分析結果


```
{
        "InputInformation":{
                "KinesisVideo":{
                        "StreamArn":" arn:aws:kinesisvideo:ap-northeast-1:XXXXXXXXXXX: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”
                        }
                }]
        }]
}
```


© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

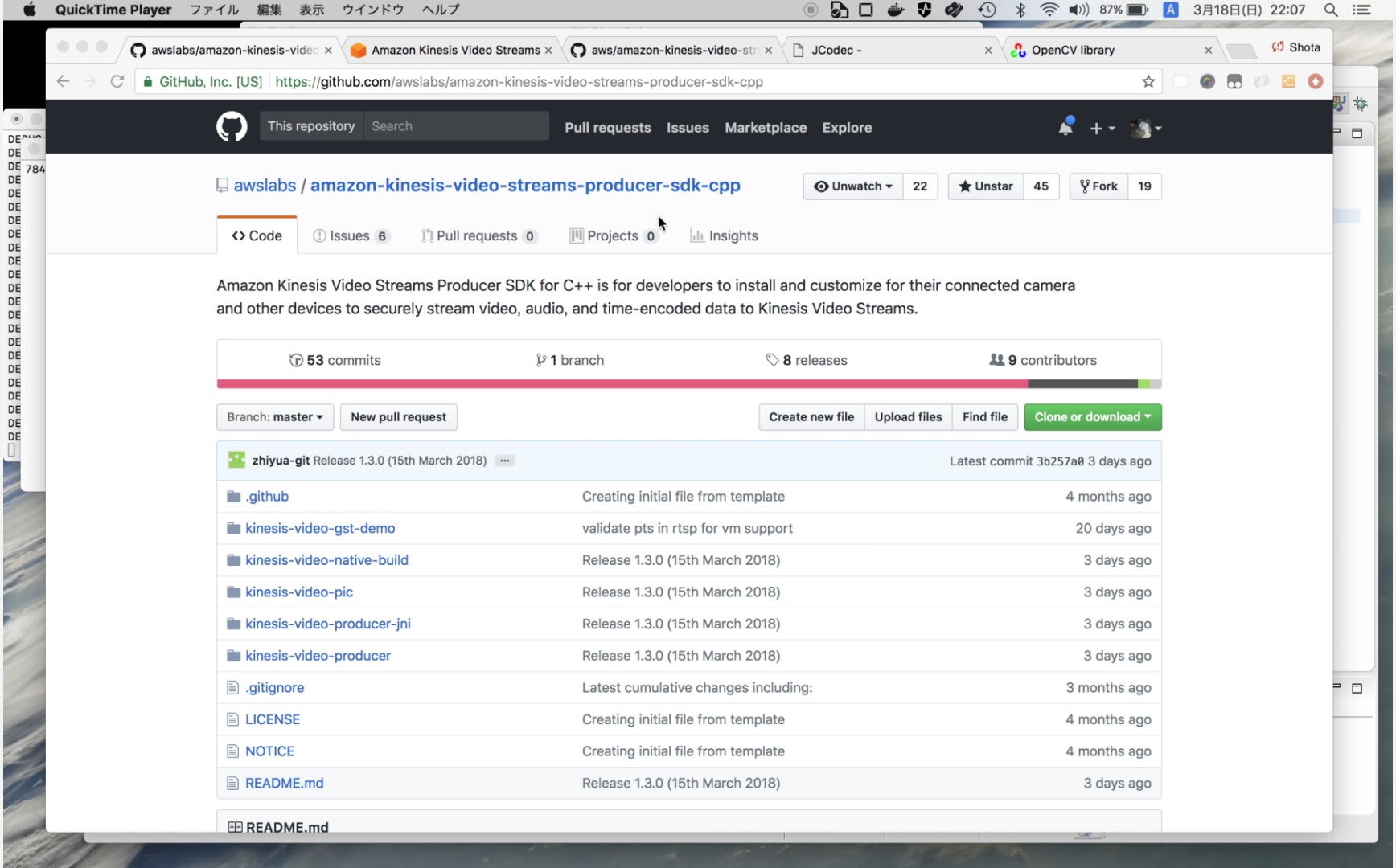
re:Invent2017：https://www.youtube.com/watch?v=rjBXbktBxBg&t=39m15s

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

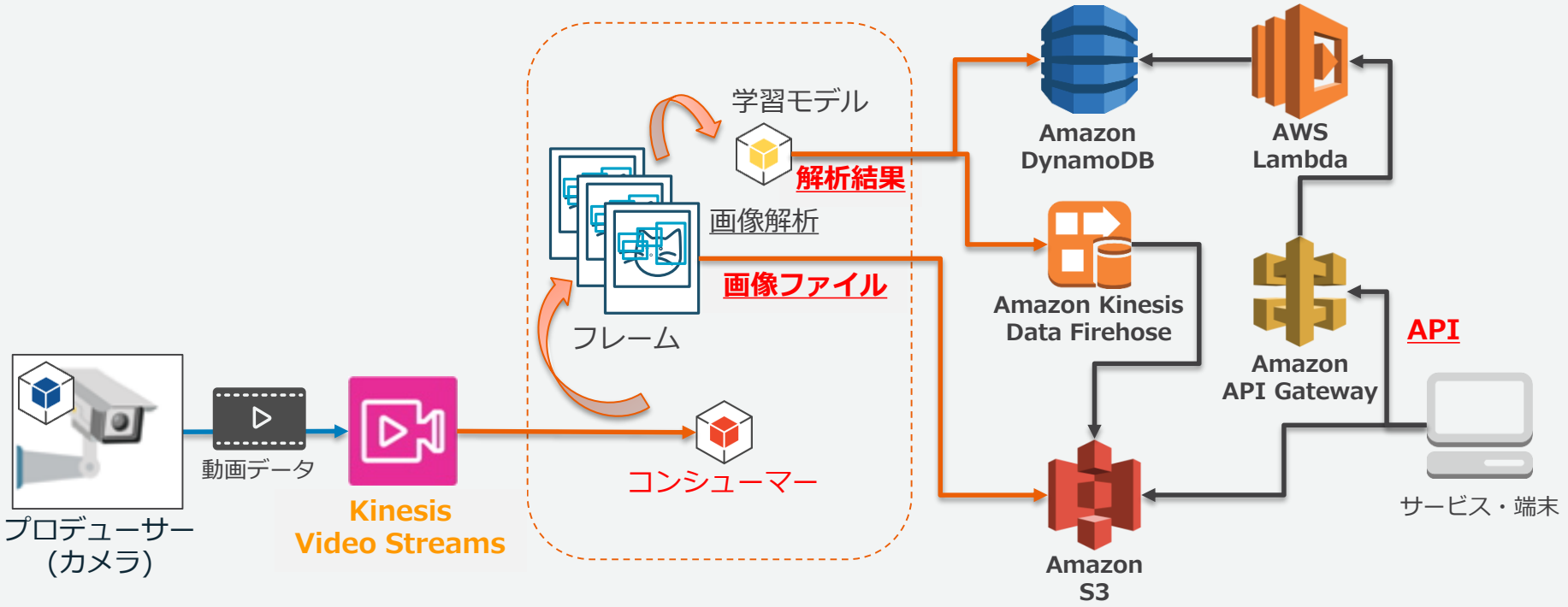

QuickTime Player ファイル 編集 表示 ウインドウ ヘルプ
3月18日(日) 22:07
awslabs/amazon-kinesis-video
Amazon Kinesis Video Streams
aws/amazon-kinesis-video-str
JCodec -
OpenCV library
Shota
GitHub, Inc. [US] https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp
This repository Search
Pull requests Issues 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

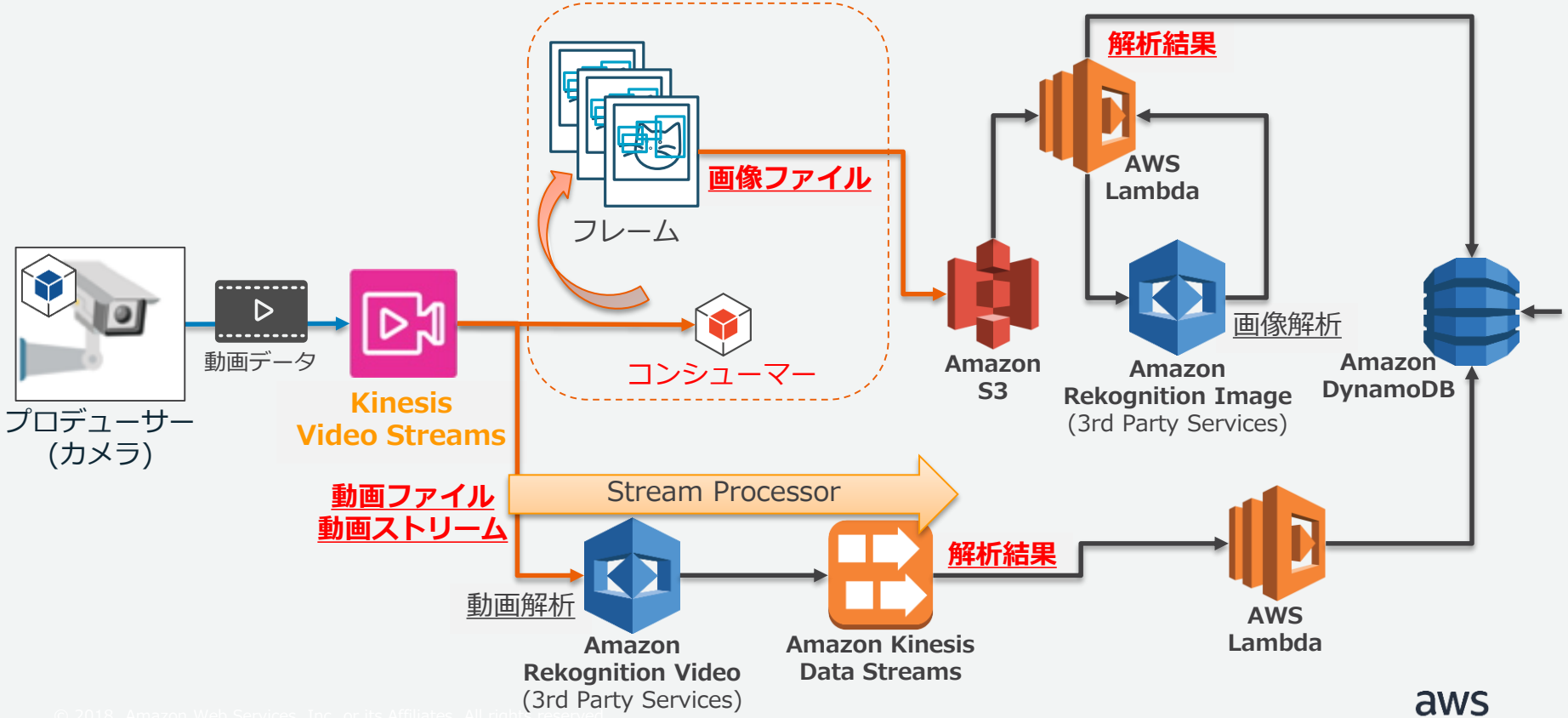
© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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



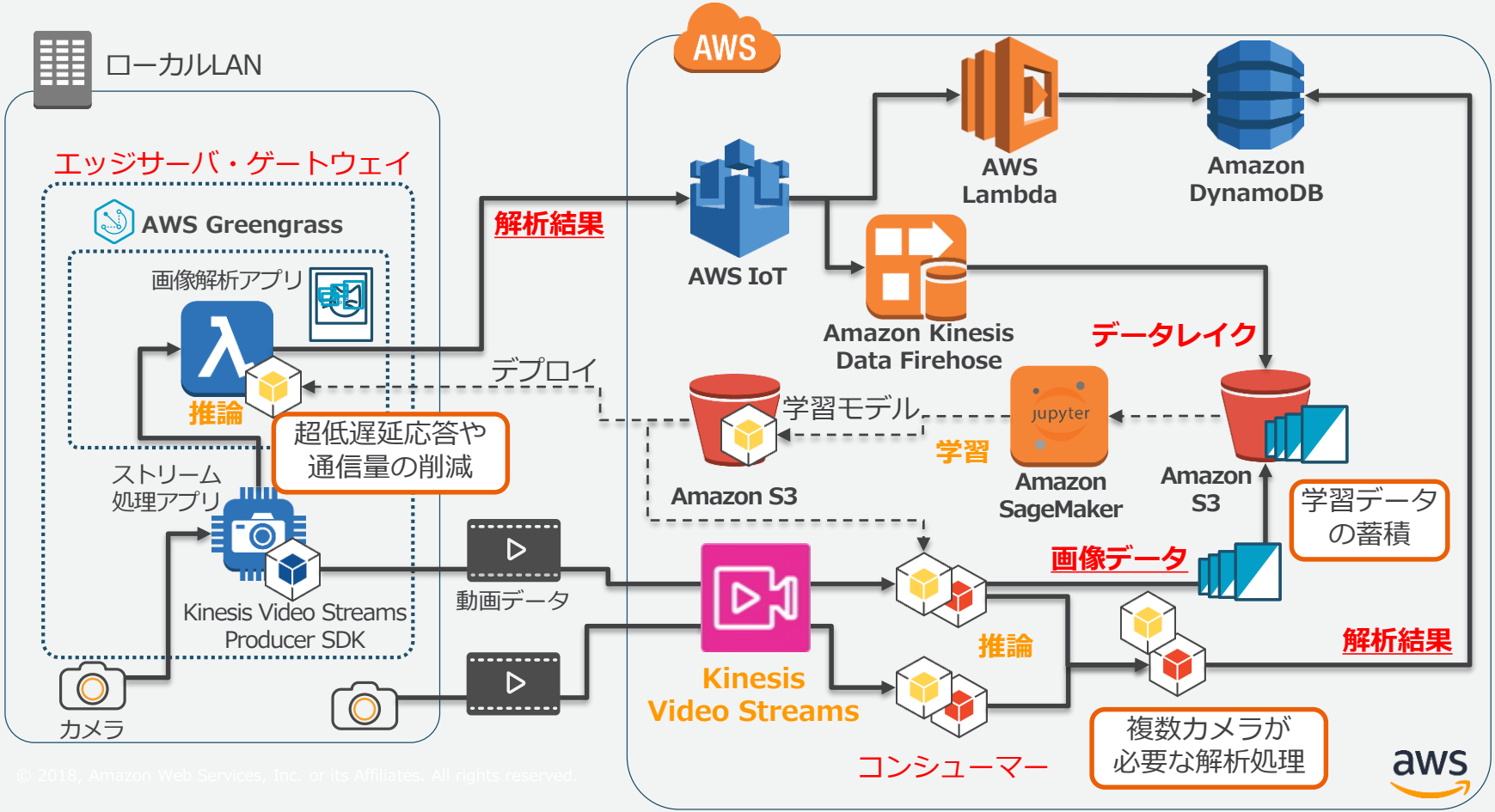
© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

動画・画像解析サービスを使って分析する



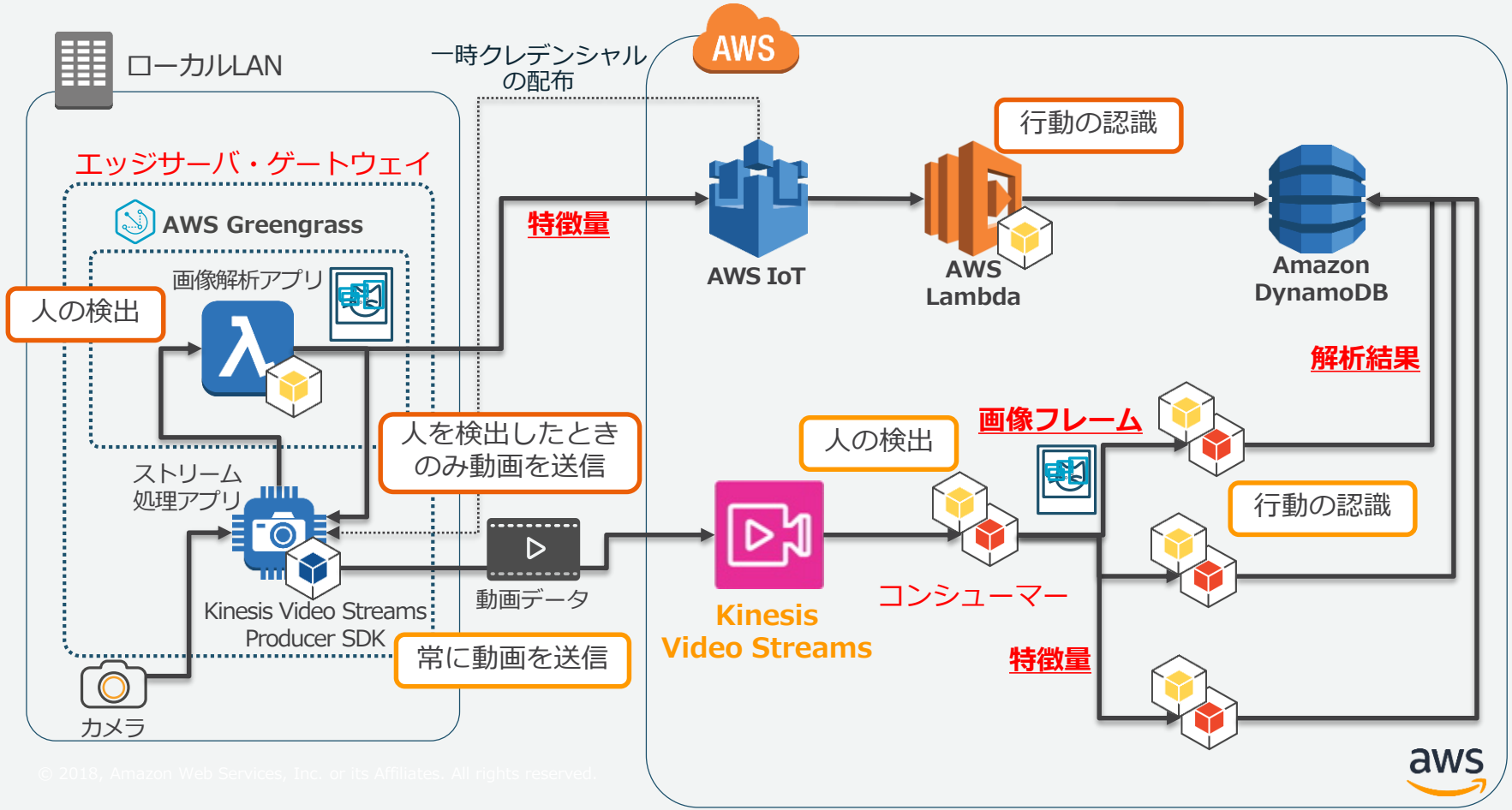
© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

クラウドとエッジでの解析処理を使い分ける



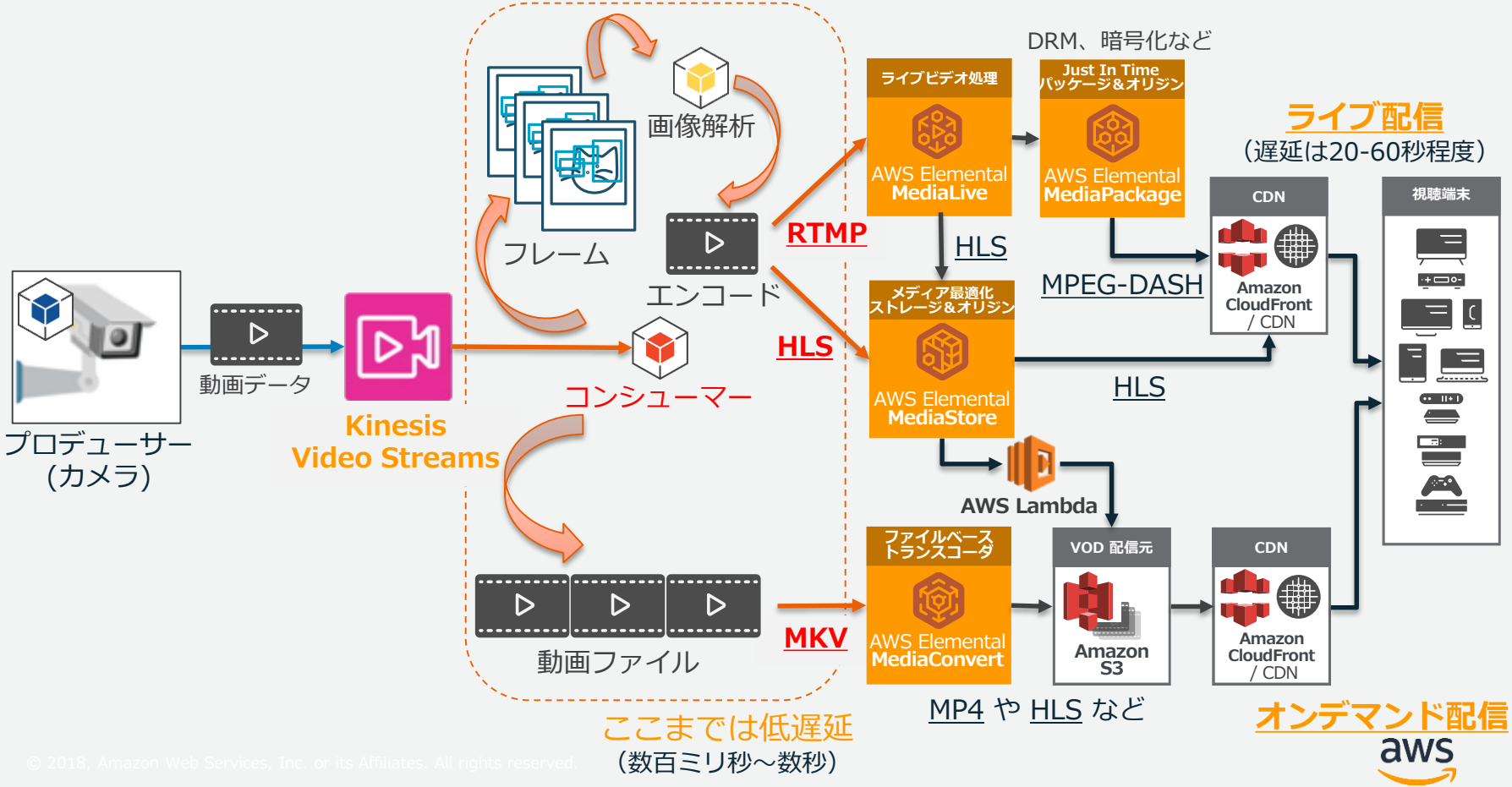
© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

検知と認識を分けて効率的にストリーム分析を行う



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

解析結果をレンダリングして動画を配信する



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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	ストリーム	コンシューマー
カメラ/マイクなど　動画の送信	動画の保存とインデックス化	動画解析・加工・再配信／変換・再生

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

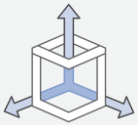
Kinesis Video Streams の特徴

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

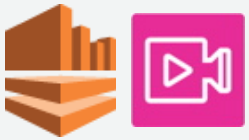
通信および保管データの暗号化と、
IAMでのアクセス管理によるセキュリティ



サーバレスなフルマネージドサービスとして提供

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

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/

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/

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

公式Twitter/Facebook
AWSの最新情報をお届けします

@awscloud_jp

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！

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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/

※「AWS 問い合わせ」で検索して下さい。

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.