aws DevAx connect

DevAx::connectシーズン 1「イベント駆動」

第5回 Near Real-Time Analytics を実現するアーキテクチャーと実装

Takayuki Enomoto
Analytics Specialist Solution Architect
Amazon Web Services Japan K.K.

自己紹介

榎本 貴之 (Enomoto, Takayuki)

所属
技術統括本部レディネスソリューション本部
アナリティクスソリューション部

略歴
インフラエンジニア @システムインテグレーター-> インフラエンジニア @ゲーム会社
-> Cloud Support Engineer @AWS -> **Analytics Specialist SA @AWS**

好きな AWS サービス
Amazon Elasticsearch Service, Amazon QuickSight, Amazon Kinesis, Amazon Neptune, Amazon CloudWatch, AWS Config, AWS Systems Manager, AWS Support

本日のアジェンダ

- リアルタイム分析とストリーム処理
- AWS におけるストリーム処理の実装
- イベント駆動アプリケーションとストリーム処理の美味しい関係
- 付録: ストリーム処理の性能問題と切り分け

リアルタイム分析とストリーム処理

何故リアルタイム分析が求められているのか

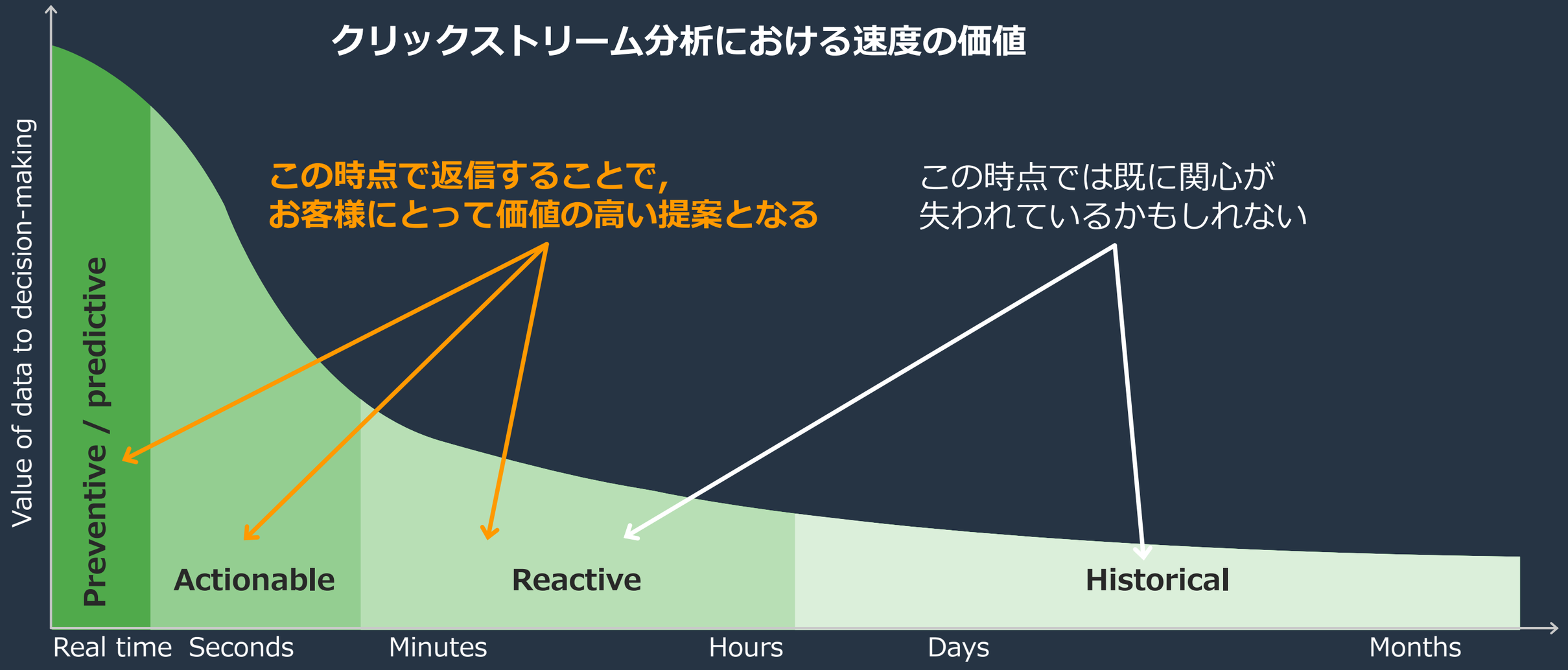
企業が価値を創造するためには, 高速かつ大量に生成される
様々なデータソースから洞察を導き出す必要がある

To create value companies must derive insights from a variety of data sources that are producing data at high velocity and volume

> データ統合の要件として, リアルタイムストリーミング,
> レプリケーション, 仮想化機能が求められている
>
> *Data integration requirements … now demand real-time streaming, replication and virtualized capabilities …*

—Gartner 2019 Planning Guide for Data and Analytics

新しいデータは意思決定における価値が高い



Source: Mike Gualtieri, Forrester, *Perishable Insights*

aws DevAx connect

バッチ処理

従来のバッチ処理では, 一般的に**データストア**に保存されているデータを元に分析, 加工を行う

ソース

アプリケーション内の処理に応じて, アプリケーションから随時, または定期的にデータが発生する

保存

アプリケーションサーバーから送信されたデータは, データベースなどのデータストアに保存される

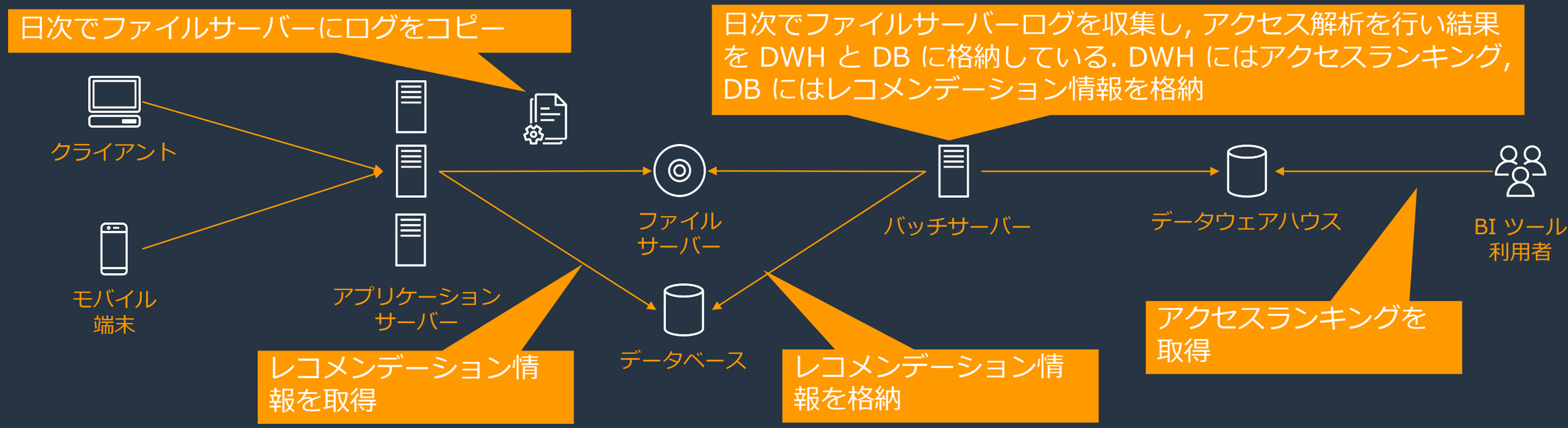
データ処理, 分析

バッチ処理を定期的に実行する. バッチ処理では, データストア内に格納されたデータを使用し処理, 分析を行う

(option)
保存

バッチ処理の結果は必要に応じて再度データストアにロードされる

ある分析基盤におけるバッチ処理の課題



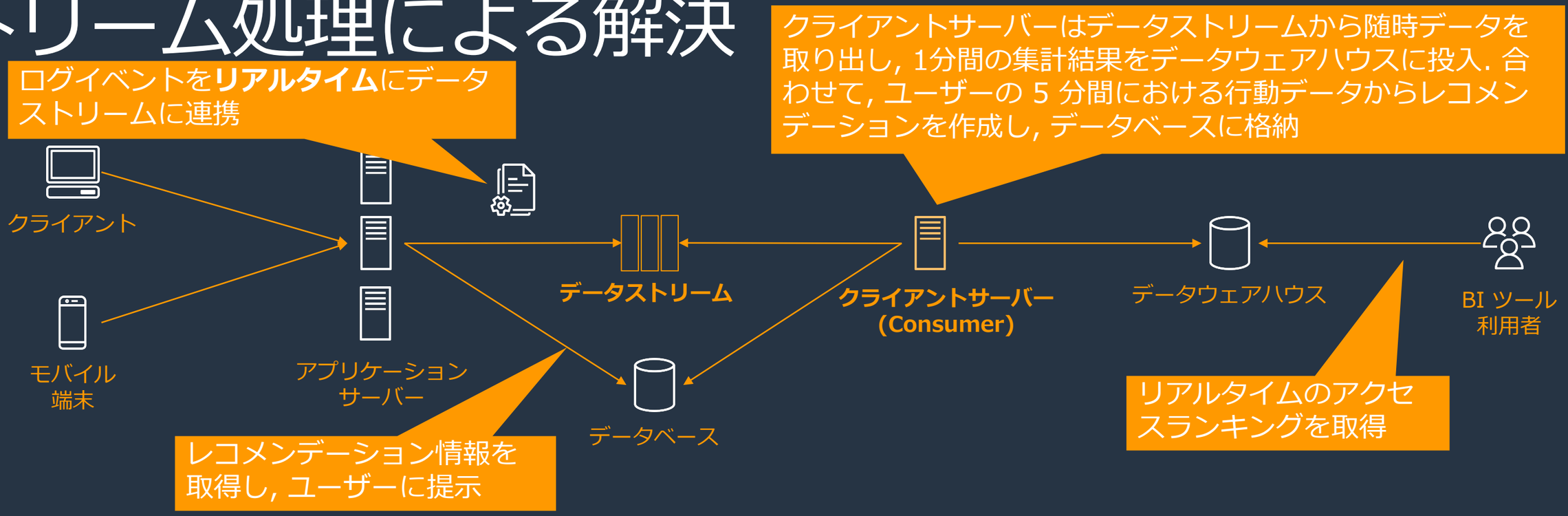
アプリケーションサイドの課題

- 利用可能な分析結果が前日のものしかないため, リアルタイムレコメンデーションが実現できていない
- 直近でアクセス数が増加しており, ログサイズの増加に伴うアプリケーションサーバーのディスク使用率ひっ迫や転送時間の増加が課題になっている. また新規システムリリースに伴いファイルサーバーの帯域がひっ迫しており, 今までの半分のスピードでファイルを転送する必要に迫られている

分析処理における課題

- ログの転送頻度が毎時, もしくは 1 日に 1 回であるため, 分析結果を提供するまでに時間がかかってしまう
- まとまった量のデータをまとめて分析しなければならないため, データ転送, 前処理, 分析処理に時間がかかっている. また新規にリリースされた別システムとバッチ処理の実行タイミングが重なっており, 書き込みによるデータウェアハウスの負荷も上がっている. 実際に BI ツール利用者に影響が出てしまっており, システム間で書き込みタイミングを調整する必要が出ている

ストリーム処理による解決



アプリケーションにおける改善

- リアルタイムに反映される情報を用いて, サイト滞在中のお客様に対して効果的なレコメンデーションを提供できる
- BI ツールから最新のトレンドを取れるため, キャンペーンの効果をリアルタイムに測定することも可能となった
- データストリームに随時データを連携すればよいため, ログファイルのコピーによる帯域のひっ迫を気にする必要がない

分析に処理における改善

- イベントがリアルタイムに取得できるため, 直近のデータを活用した集計, レコメンデーションなどの処理を実現可能
- データは時系列順にデータストリームに集約されるため, マージ処理は不要に
- データウェアハウスに一括で大量データがロードされることもなくなり, データウェアハウスの稼働が安定

aws DevAx connect

ストリーム処理

データストリームに**ストリーミングデータ**を一時的に格納し,
処理, 分析を経てからデータストアへ配信を行う

ソース

高速でリアルタイムデータを生成するアプリケーションやデバイス

収集, 集約, 送信

数万規模のデータソースからのデータをリアルタイムで収集し, 必要に応じて集約したうえでデータストリームへ送信する

取込

データストリーム内のレコードは受信した順序で保存され, 設定された期間内であれば無制限に再生(再取得, 再処理)できる

取得, 処理, 配信

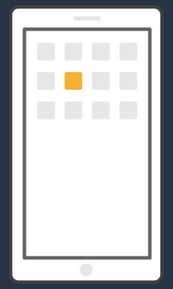
レコードは格納された順番で取り出され, リアルタイム分析やストリーミング ETL に活用される

(Option)保存

処理されたレコードは,必要に応じてデータレイク, データウェアハウス, データベース等の様々なデータストアへ配信される

ストリーミングデータ

- 数千ものデータソースによって継続的に生成される小さなデータ
- 個々のデータは一般的には 数 KB, 大きくても数 MB で, メッセージ, レコードなどと呼ばれる



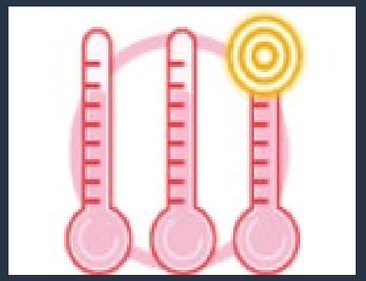
Mobile apps

Web clickstream

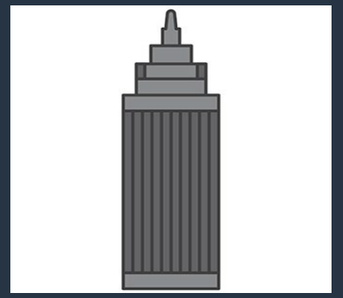
```
[Wed Oct 11 14:32:52 2018]
[error] [client 127.0.0.1]
client denied by server
configuration:
/export/home/live/ap/htdocs/
test
```

Application logs

Metering records



IoT sensors



Smart buildings

ストリーム処理の特性

無限, 永続的

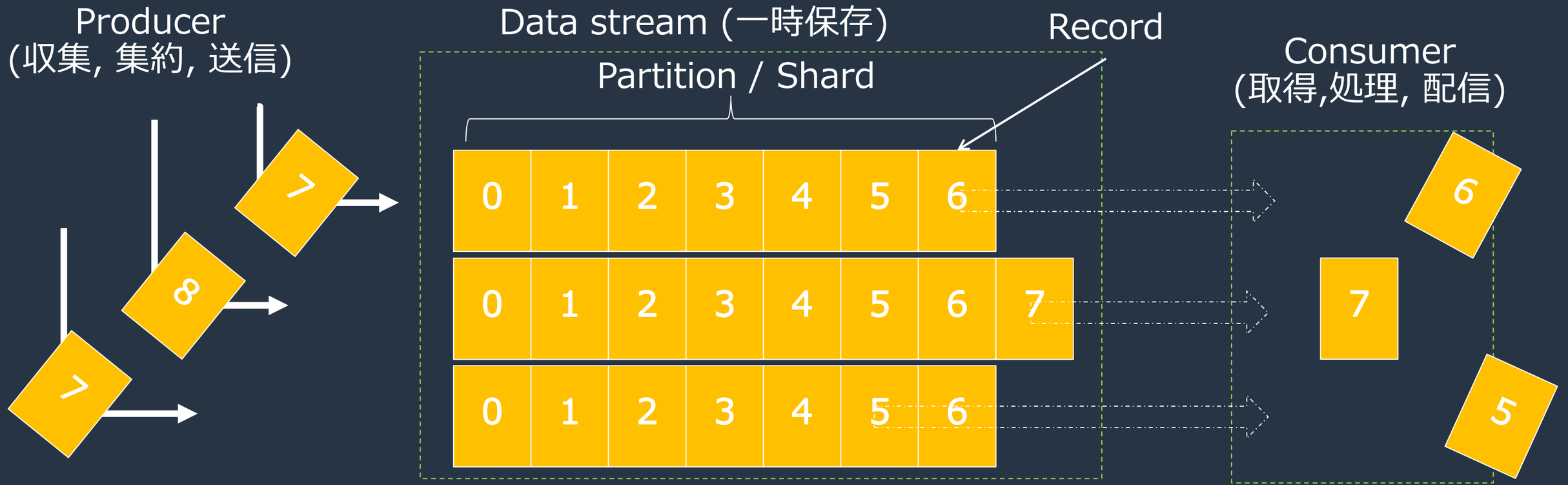
- ソースは, **ストリーミングデータ**を無限に生み出し続ける
- ソフトウェアは, 無限の**ストリーミングデータ**を永続的に処理し続ける
- **ストリーミングデータ**には明確な終わりが無いため, ストリーミングデータに対する集計を行う場合は, 一定区間でデータを区切る必要が有る

バックプレッシャー

- **ストリーミングデータ**は, **データストリーム**と呼ばれるコンポーネントでいったん受け取られる.これにより, 後続のソフトウェアやデータストアは, スパイクアクセスの影響を受けずに安全にデータを処理できる
- データストリーム自身も, 帯域や RPS (Requests per second)のソフトウェアリミットを超過したリクエストに対して” 時間をおいてのリトライを促すエラー”を返すなど, 自身を安定稼働させるための仕組みをもつ

データストリーム

- データストリームは**パーティション(シャード)**の集合
- データはデータストリーム内の各パーティションで独立して保持される
- データを送信する**プロデューサー**, データを取得する**コンシューマー**が対を成す



aws DevAx connect

データストリーム内のデータ処理方法

個別データ処理

項目追加・削除・変更などのシンプルなデータ加工

ウィンドウ処理

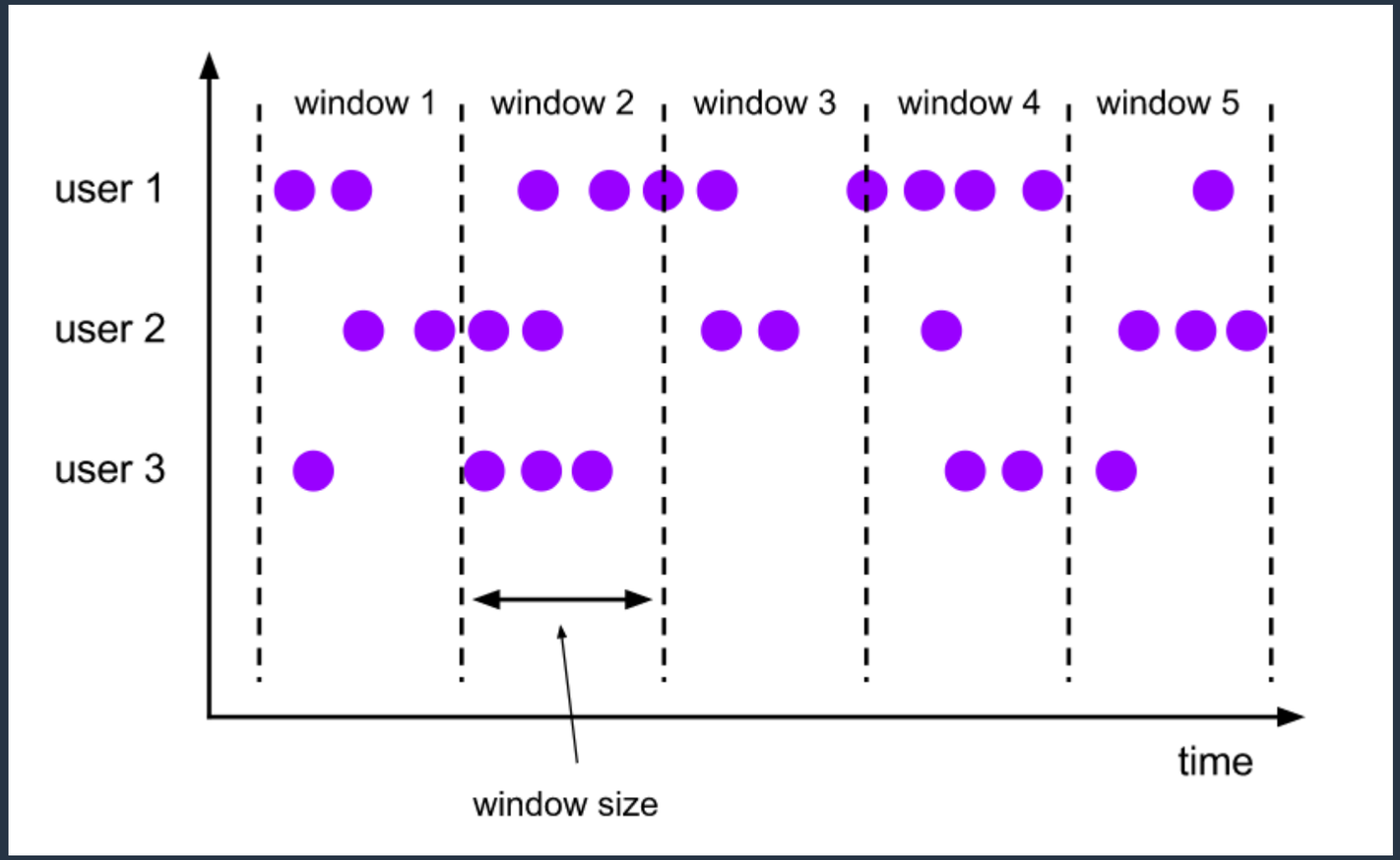
ストリーム内の特定区間(**ウィンドウ**)ごとに切り出されるデータの集合に対して処理を行う方式. 特定期間のデータ集計, 分析などに用いられる

- Tumbling Window
- Sliding Window
- Stagger Window (Amazon Kinesis Data Analytics for SQL)
- Session Window (Apache Flink)
- Global Window (Apache Flink)

Tumbling Window

- ストリームを流れるデータを固定期間のウィンドウとして切り出す方式
- 各ウィンドウは重ならない

例: デバイスから送信されたセンサーデータの **10 分**間平均をデバイスごとに集計する. 集計は **10 分**に1回行う



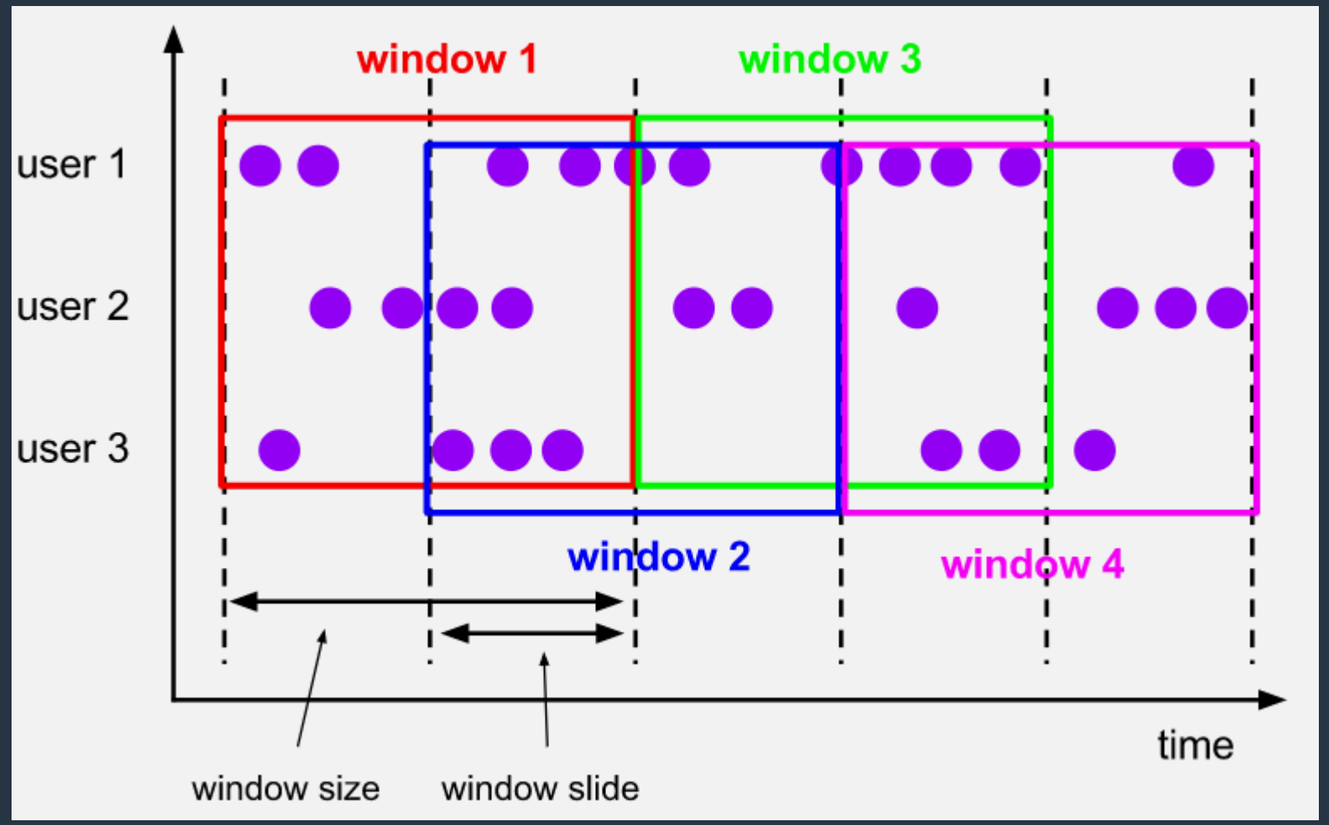
https://ci.apache.org/projects/flink/flink-docs-release-1.13/docs/dev/datastream/operators/windows/

aws DevAx connect

Sliding Window

- 固定サイズのウィンドウを一定頻度ごとに切り出す方式
- ウィンドウサイズと切り出しの頻度は異なる
- ウィンドウそのものが時間とともに移動するイメージ

例: デバイスから送信されたセンサーデータの **10 分**間の平均をデバイスごとに集計する. ただし集計は **5 分**に1回行う

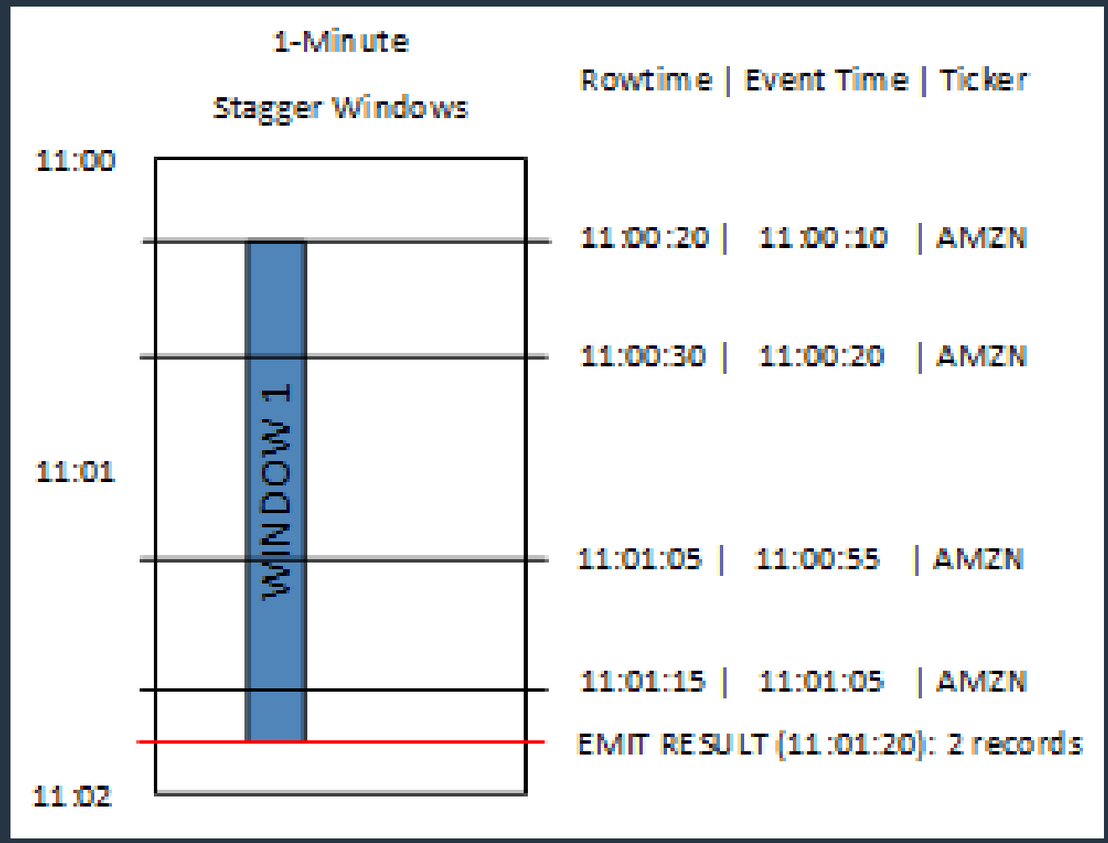


https://ci.apache.org/projects/flink/flink-docs-release-1.13/docs/dev/datastream/operators/windows/

Stagger Window

- 一貫性のない時間に届くデータをグルーピングして分析するのに適した方式
- パーティションキーに一致する最初のイベントが届いたタイミングでウィンドウが開く. ウィンドウサイズは固定

例: デバイスから送信されたセンサーデータの “**10 分間**” における平均値を “**デバイスごと**” に集計する. 集計区間は “**デバイスから最初のデータが届いてから 10 分間**”



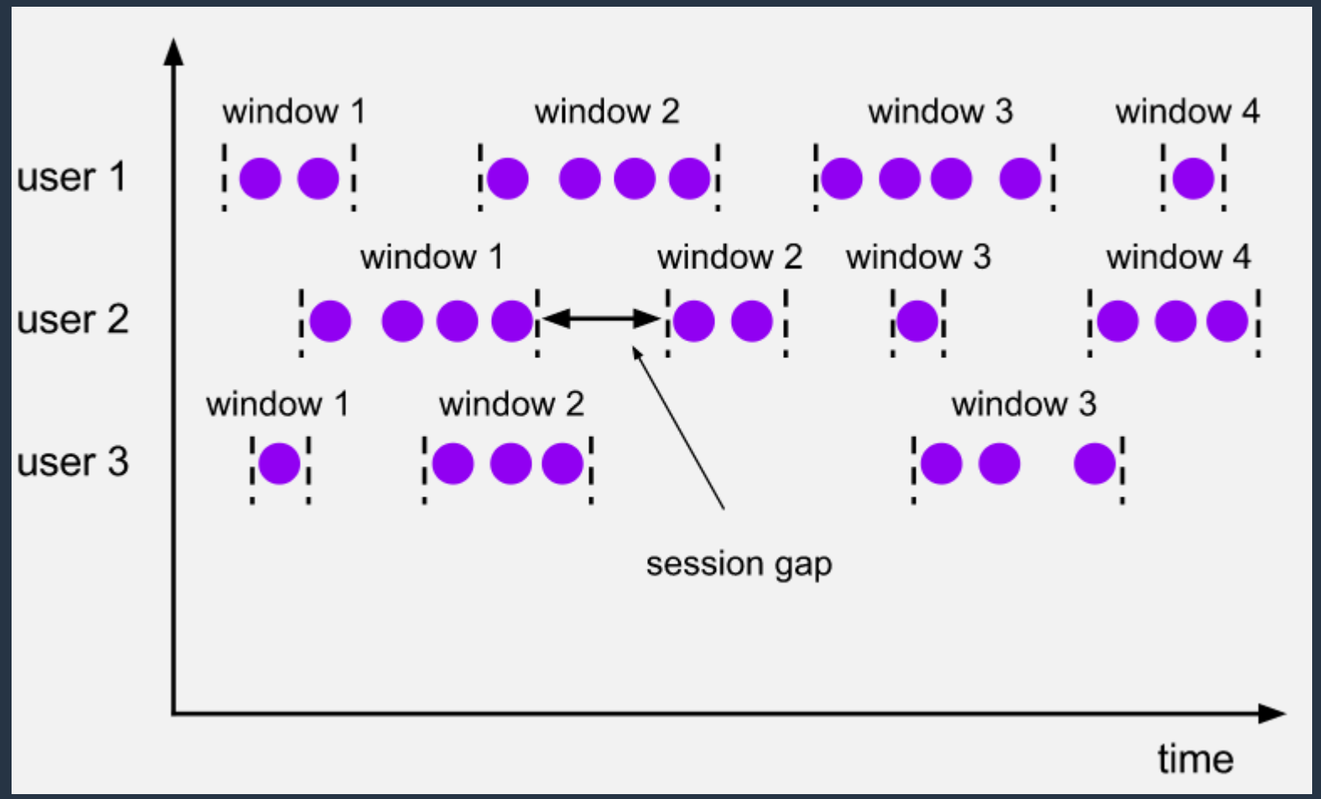
https://docs.aws.amazon.com/ja_jp/kinesisanalytics/latest/dev/stagger-window-concepts.html

Session Window

- 継続するデータを 1 つのウィンドウに収める方式

- 指定した期間イベントが発生していない場合にウィンドウが区切られる

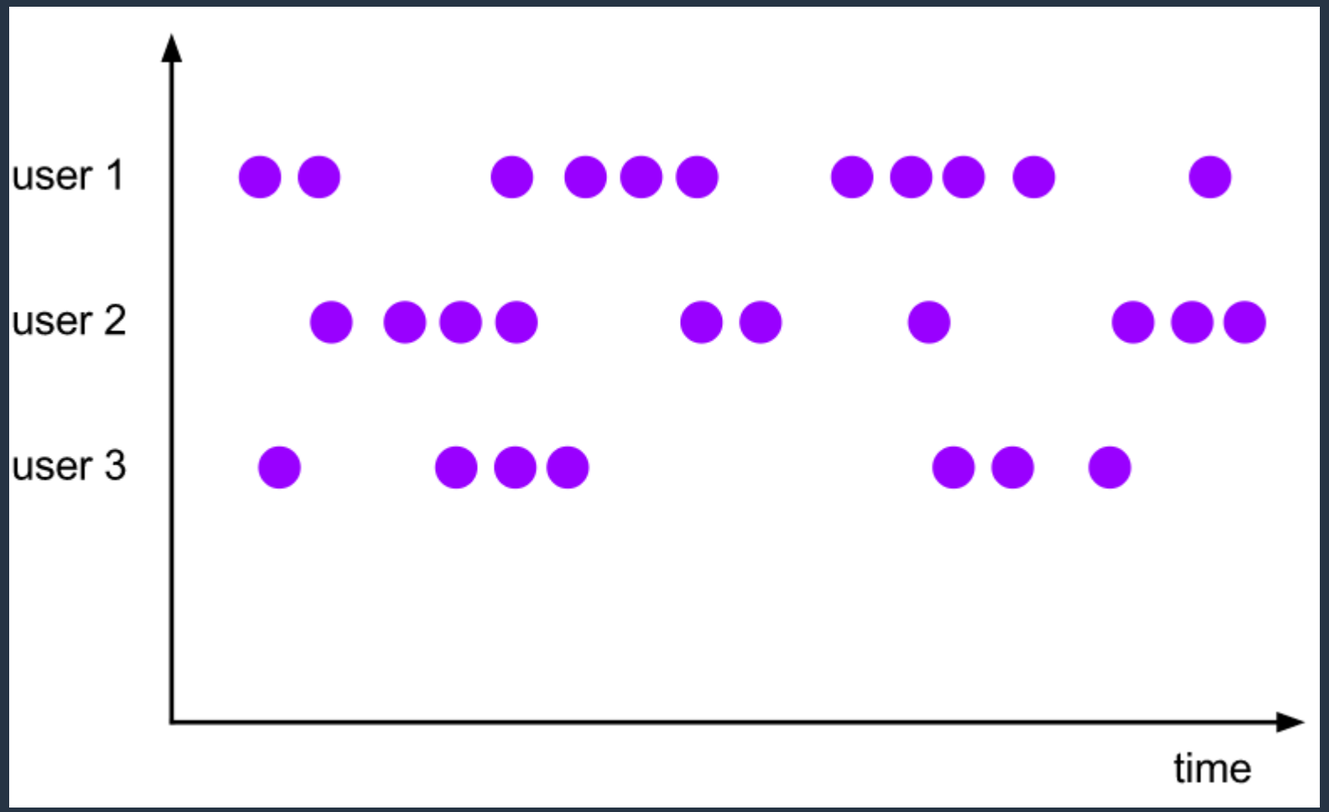
例: ユーザーによって行われる一連のアクティビティを分析する. 30 分アクティビティが無い場合はウィンドウをいったん閉じる



https://ci.apache.org/projects/flink/flink-docs-release-1.13/docs/dev/datastream/operators/windows/

Global Window

- サイズ上限のないウィンドウ
- アプリケーション側で任意条件に基づき trigger を引き，ウィンドウの open/close を行う
- 従来のウィンドウでは要件を満たせない場合に使用する



https://ci.apache.org/projects/flink/flink-docs-release-1.13/docs/dev/datastream/operators/windows/

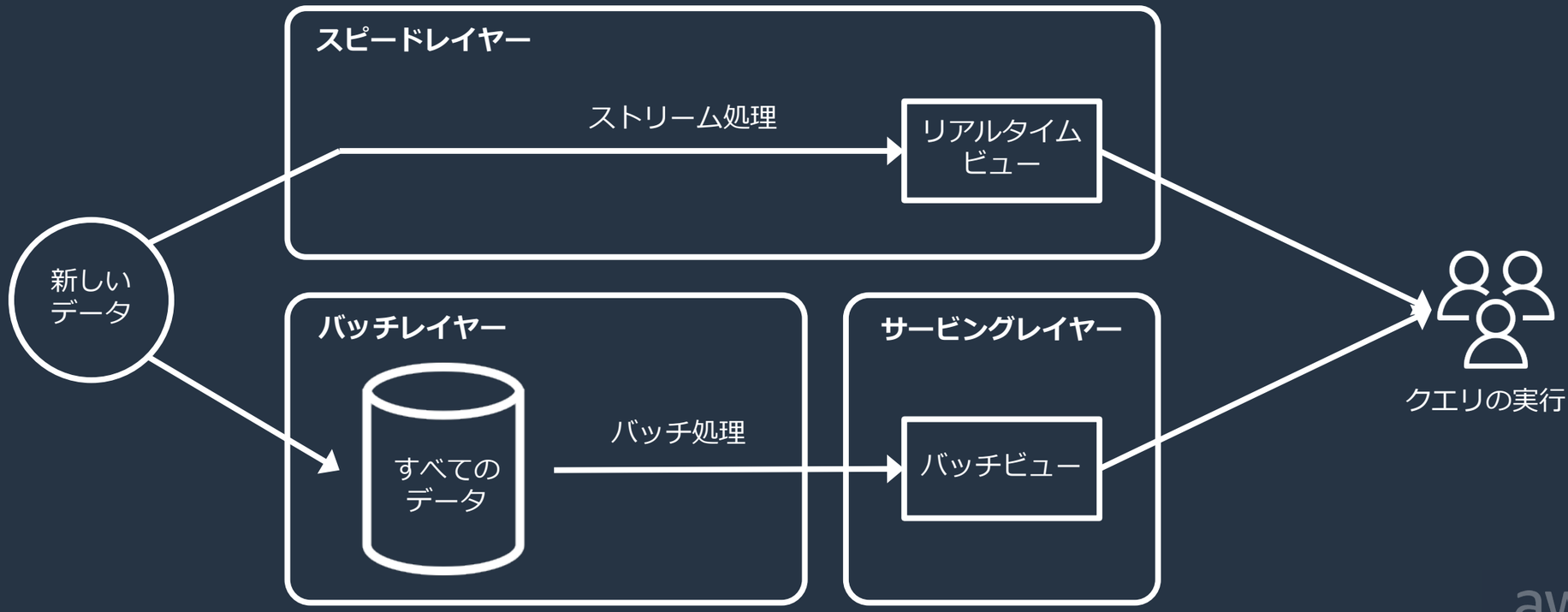
バッチ処理とストリーム処理の比較

ストリーム処理は強力だが万能ではない. 必要に応じて取り入れること

	バッチ処理	ストリーム処理
入力データの範囲	データの全部または大部分	直近のデータレコード
データサイズ	大きなデータバッチ	個々のレコードまたはマイクロバッチ
レイテンシ	データサイズや処理内容に依存する. 一般的には数秒～数時間	ユースケース上, 数ミリ秒～数秒であることが求められる
ユースケース	シンプルな集計から複雑かつ大規模な分析まで	シンプルな集計, データ変換, API 呼び出しを伴う外部サービスとの連携
再処理の可否	可能	ストリームにデータが残っていれば可能. レイテンシ要件が緩ければバッチ処理で再集計の方が楽な場合も
結果の正確度	真値	近似値～真値
処理の複雑さ	シンプルに実装できる	収集, 変換, 分析が一体となるため考慮事項が多く, 処理が複雑になりがち

ラムダアーキテクチャー

- Apache Storm の開発者 Nathan Marz が 2012 年に提唱
- データ処理の流れを，全量のデータを保持し定期的な処理を行うバッチレイヤーと，新しく入ってきたデータをストリーム処理するスピードレイヤーに分割
- 両者を組み合わせて結果を表示する



AWS におけるストリーム処理の実装

aws DevAx connect

実装上のポイント

- ユースケース, レイテンシ要件, データ量を確認する

- マネージドサービスを組み合わせたシステムアーキテクチャーを検討する

- (必要が無ければ)実装しない

実装上のポイント

- ユースケース, レイテンシ要件, データ量を確認する

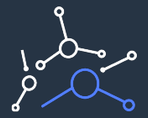
- マネージドサービスを組み合わせたシステムアーキテクチャーを検討する

- (必要が無ければ)実装しない

ストリーム処理の代表的なユースケース

異常, 不正のリアルタイム検出

リアルタイムの顧客体験改善



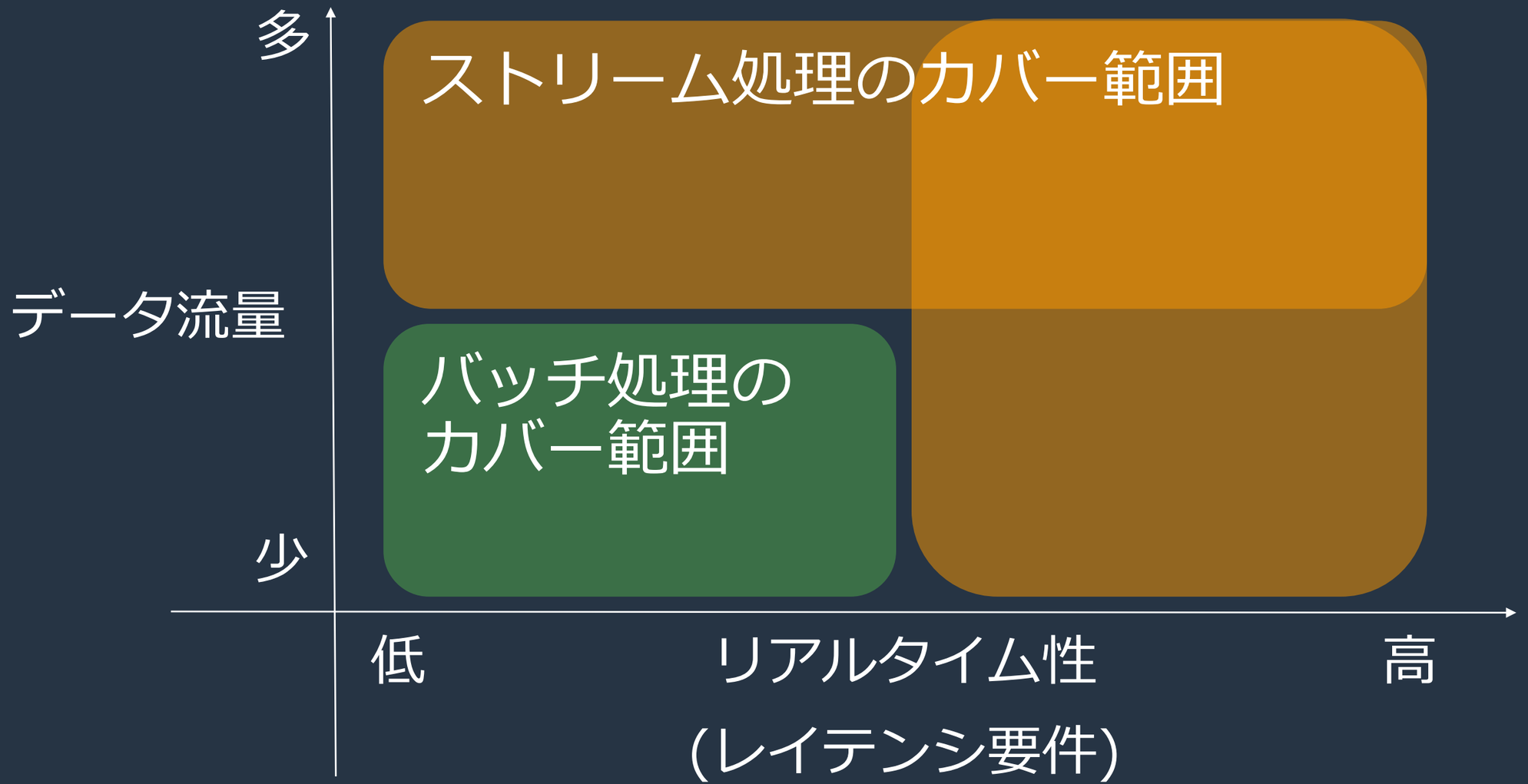
IoT 分析の強化

マーケティングキャンペーン

リアルタイムなパーソナライゼーション

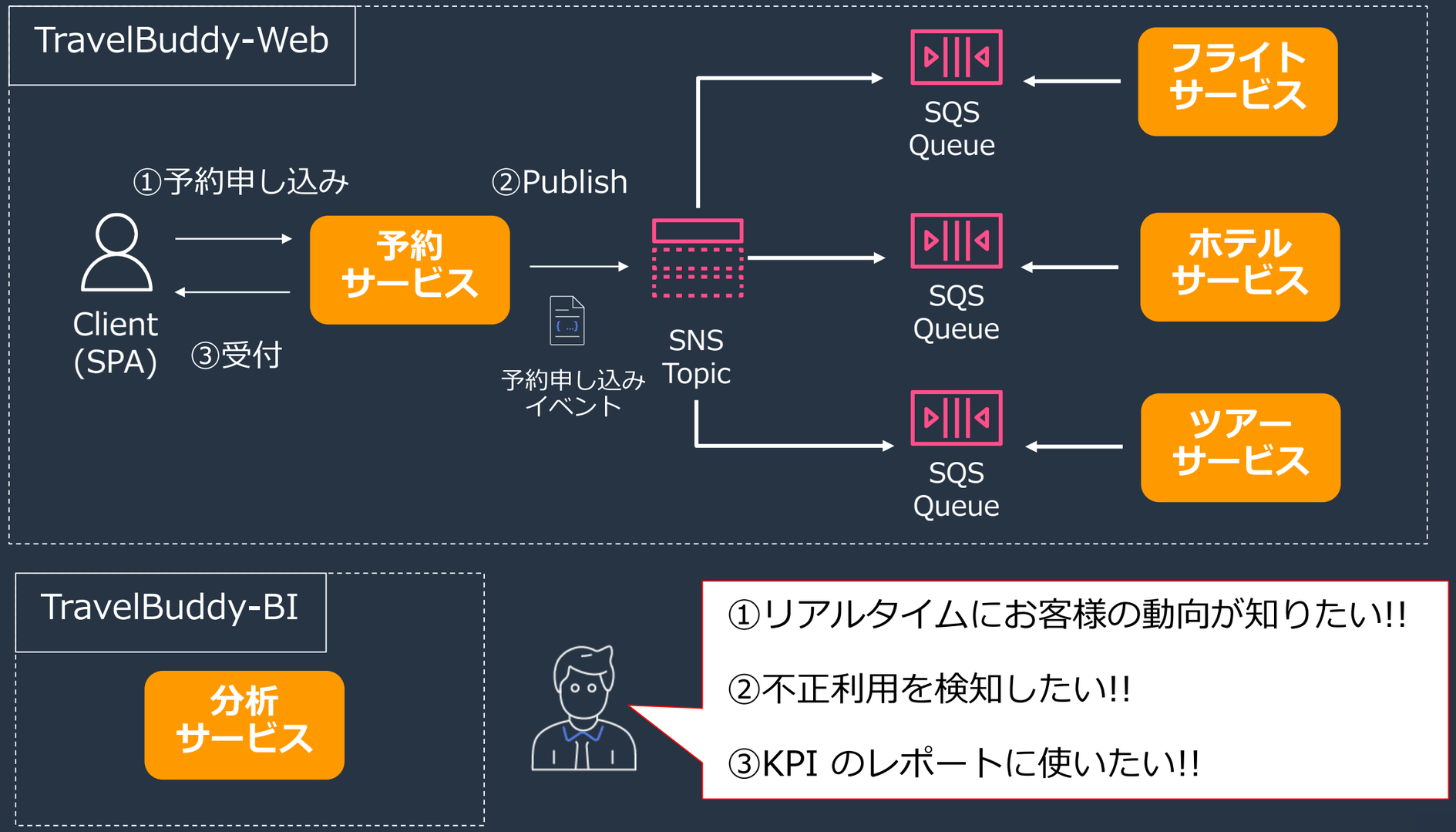
ヘルスケアサービスのサポート

レイテンシ要件, データ量とストリーム処理の適用範囲



おさらい: TravelBuddy の分析要件

第四回資料より



aws DevAx connect

要件の深堀り, データ量のヒアリングを行う

要件

①リアルタイムにお客様の動向が知りたい!!
ツアー毎の予約状況, キャンセル状況を**リアルタイム**に見たい

②不正利用を検知したい!!
特定のユーザーが予約とキャンセルを繰り返しているなど,不審な行動を**速やかに**検出, 通知したい

③KPI のレポートに使いたい!!
日次でレポートを作成し, 関係者にメールで配信したい

データ流量

- メッセージサイズ: 1KB
- メッセージの流量: 20000 messages/sec

得られた要件, データ量から対応方針を決定する

要件

①リアルタイムにお客様の動向が知りたい!!
ツアー毎の予約状況, キャンセル状況を**リアルタイム**に見たい

リアルタイムダッシュボードを構築する

②不正利用を検知したい!!
特定のユーザーが予約とキャンセルを繰り返しているなど,不審な行動を**速やかに**検出, 通知したい

ウィンドウ処理を使ったリアルタイム分析を行う

③KPI のレポートに使いたい!!
日次でレポートを作成し, 関係者にメールで配信したい

データ流量

- メッセージサイズ:
- メッセージの流量: 20000 messages/sec

レポートの作成自体はバッチ処理だが, 今後メッセージ流量が増えた際にバッチ処理時間が延びることを防ぐため, データ変換などの前処理はストリーム処理にオフロードする

実装上のポイント

- ユースケース, レイテンシ要件, データ量を確認する

- マネージドサービスを組み合わせたシステムアーキテクチャーを検討する

- (必要が無ければ)実装しない

確認, 検討対象は大きく 5 つに分かれる

Source | **Producer** | **Data Stream** | **Consumer** | **Destination**

External Service

Log Files

Web Brower

Mobile Application

Sensor Device

AWS Amplify

Amazon CloudWatch Logs

Kinesis Producer Library

Kinesis Agent

AWS IoT Core

3rd-party Agents

Amazon Kinesis Data Streams

Amazon Kinesis Data Firehose

Amazon Managed Streaming for Apache Kafka

Amazon DynamoDB Streams

Amazon Kinesis Video Streams

Amazon Kinesis Data Firehose

Amazon Kinesis Data Analytics

Kinesis Client Library

AWS Glue

AWS Lambda

Amazon S3

Amazon Redshift

Amazon RDS

Amazon DynamoDB

Amazon Elasticsearch Service

External Service

aws DevAx connect

Source

Source | **Producer** | **Data Stream** | **Consumer** | **Destination**

- External Service
- Log Files
- Web Brower
- Mobile Application
- Sensor Device

- AWS Amplify
- Amazon CloudWatch Logs
- Kinesis Producer Library
- Kinesis Agent
- AWS IoT Core
- 3rd-party Agents

- Amazon Kinesis Data Streams
- Amazon Kinesis Data Firehose
- Amazon Managed Streaming for Apache Kafka
- Amazon DynamoDB Streams
- Amazon Kinesis Video Streams

- Amazon Kinesis Data Firehose
- Amazon Kinesis Data Analytics
- Kinesis Client Library
- AWS Glue
- AWS Lambda

- Amazon S3
- Amazon Redshift
- Amazon RDS
- Amazon DynamoDB
- Amazon Elasticsearch Service
- External Service

aws DevAx connect

Source

External Service
Log Files
Web Brower
Mobile Application
Sensor Device

メッセージの出力元によってデータストリームにデータを送信するための手段が決まってくる

ログ
- ログファイル -> ログ収集ソフトウェア
- コンテナログ -> サイドカー, デーモンセット
- サーバーレスアプリ(Lambda, API Gateway) -> CloudWatch Logs

メッセージ
- モバイルアプリケーション, SPA-> SDK, ライブラリ
- IoT デバイス -> AWS IoT Core

Destination

Source | **Producer** | **Data Stream** | **Consumer** | **Destination**

External Service
Log Files
Web Brower
Mobile Application
Sensor Device

AWS Amplify
Amazon CloudWatch Logs
Kinesis Producer Library
Kinesis Agent
AWS IoT Core
3rd-party Agents

Amazon Kinesis Data Streams
Amazon Kinesis Data Firehose
Amazon Managed Streaming for Apache Kafka
Amazon DynamoDB Streams
Amazon Kinesis Video Streams

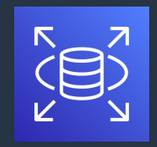
Amazon Kinesis Data Firehose
Amazon Kinesis Data Analytics
Kinesis Client Library
AWS Glue
AWS Lambda

Amazon S3
Amazon Redshift
Amazon RDS
Amazon DynamoDB
Amazon Elasticsearch Service
External Service

aws DevAx connect

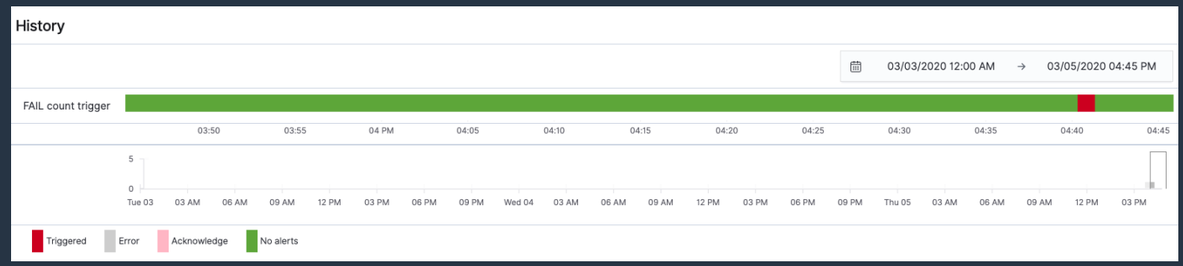
Destination を選定する

要件に応じて配信先のデータストアを決定する

リアルタイムダッシュボードを構築したい	Amazon Elasticsearch Service 
リアルタイムに蓄積されたデータをアドホックに集計したい リアルタイムに前処理されたデータを ML で活用したい	Amazon S3 
データの前処理をリアルタイムに行い, DB/DWH に格納して BI ツールで分析したい	Amazon Redshift  Amazon RDS 
外部サービスへ連携したい	外部サービスの API エンドポイント 

Amazon ES で実装するリアルタイムダッシュボード

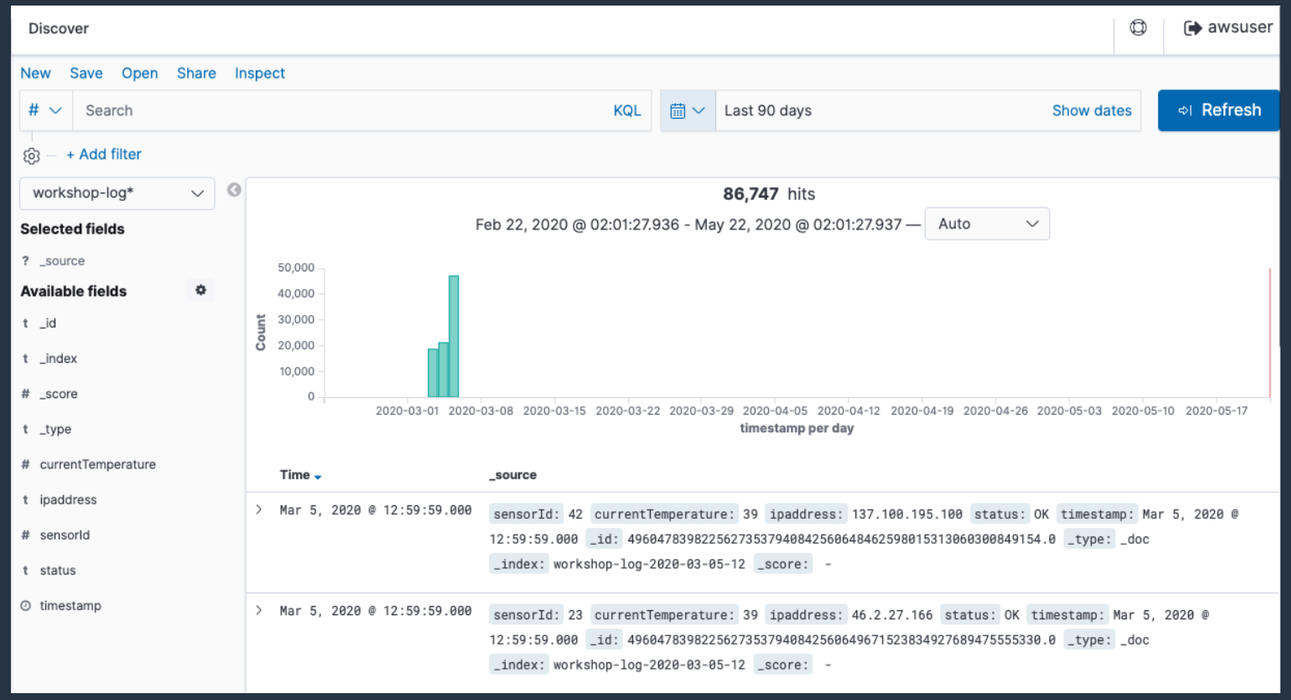
- ストリーミングデータを Amazon Elasticsearch Service に蓄積し，Kibana で可視化
- GUI ベースでダッシュボードの作成，管理を行い，詳細な権限管理で複数部署が共同利用
- 基準を超えるような数値が出たら，自動で通知を送るような連携も可能



aws DevAx connect

リアルタイムダッシュボードの利用シーン

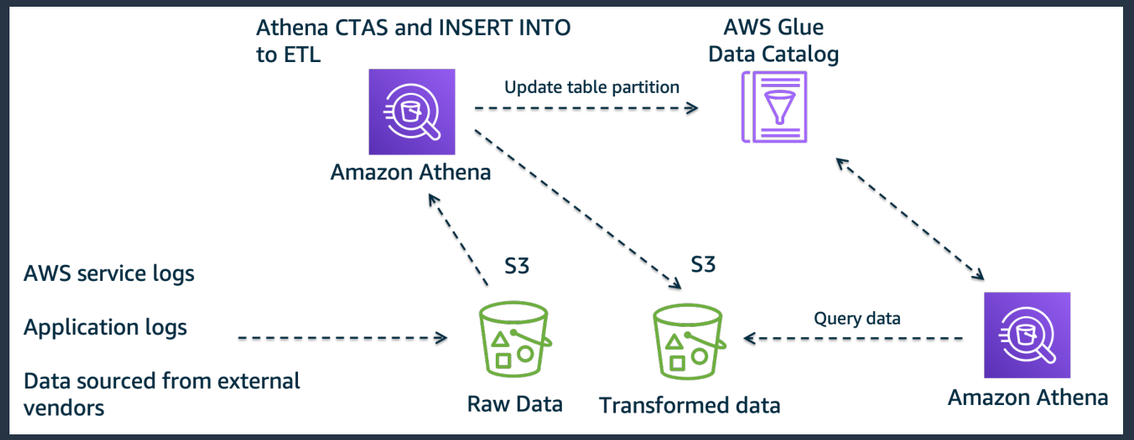
- Kibana を用いることでよりインタラクティブな形で分析を実施可能
- ビジネスユーザーも使うが，アナリストや開発者がメインターゲット
- SIEM(Security Information and Event Management) のように，セキュリティ問題の詳細な調査を行う
- カスタマーサポートで特定のログを検索する，全文検索を含めた調査を行う



aws DevAx connect

S3 に格納されたデータに対するアドホック処理

ストリーミングデータを S3 に配信することで, Amazon Athena を使用した**最新のデータに対する**アドホックな分析, 小規模なデータ変換処理を実行可能



SELECT bucket, * FROM s3_analytics WHERE object_count IS NOT NULL ORDER BY storage_mb DESC

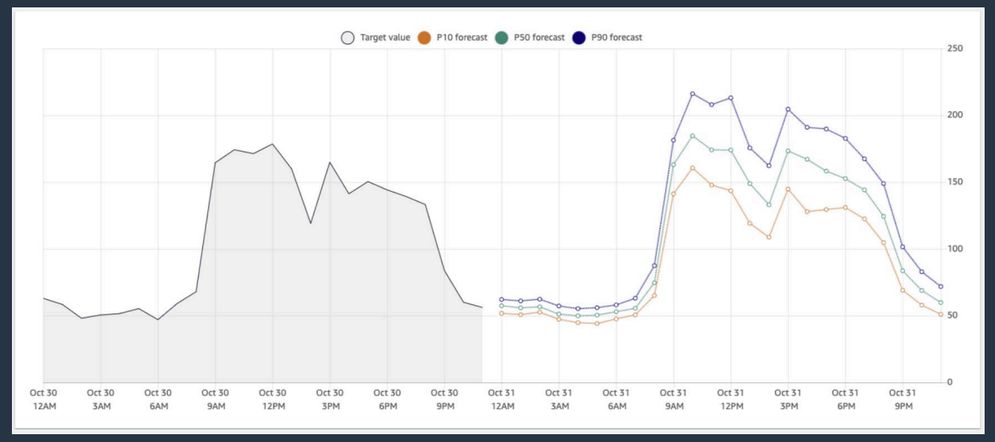
	bucket	date	config_id	filter	storage_class	object_age	object_count	uploaded_mb	storage_mb	retrieved_mb
1	werberm-bigdata	2019-09-15	analytics-config		STANDARD	ALL	560		66265.5386	
2	cloudtrail-awslogs-544941453660-ib8kuoba-isengard-do-not-delete	2019-09-15	analytics-config		STANDARD	ALL	701185	13.2167	4492.3477	
3	aws-glue-temporary-544941453660-us-east-1	2019-09-15	analytics-config		STANDARD	ALL	59		1891.6972	
4	544941453660-awsmacietrail-dataevent	2019-09-15	analytics-config		STANDARD	ALL	314894	1.5116	1572.4439	
5	werberm-sandbox	2019-09-15	analytics-config		INTELLIGENT_TIERING	ALL	46		564.2702	

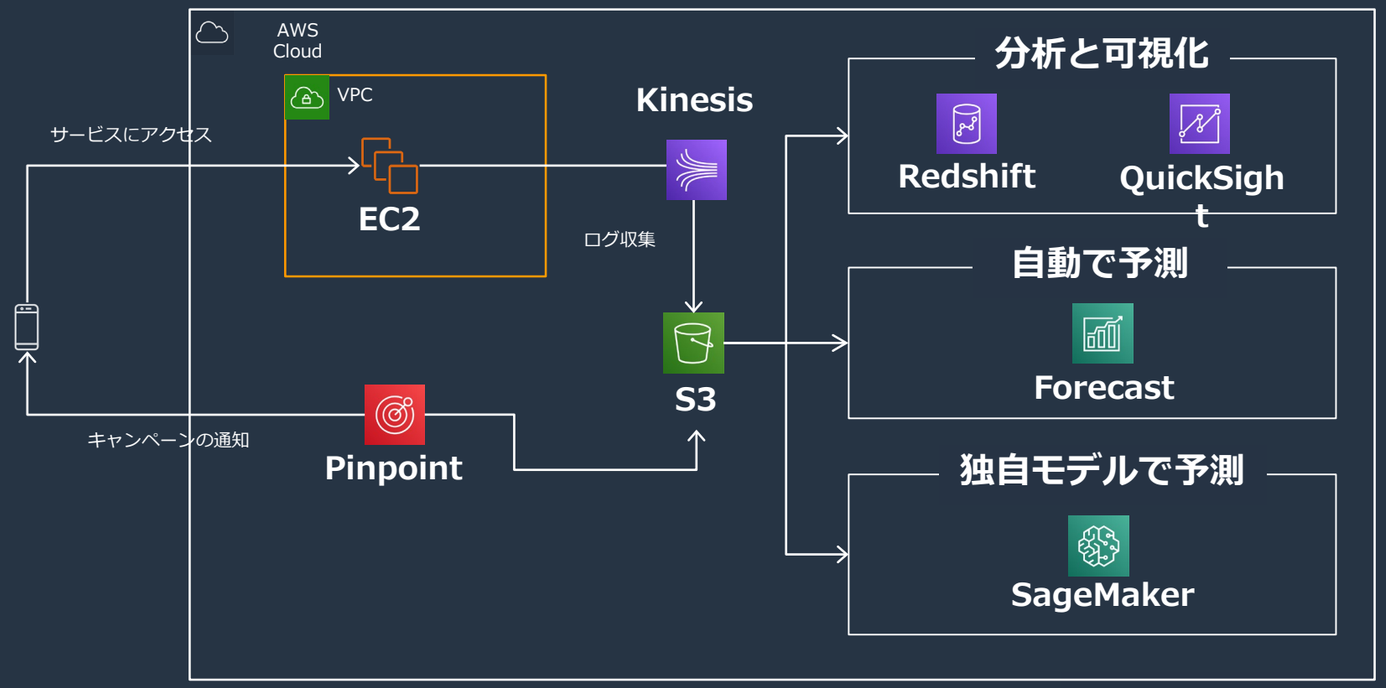
aws DevAx connect

S3 に格納されたデータと機械学習系サービスの連携

- Amazon Personalize，Amazon Forecast, Amazon Personalize, Amazon Comprehend といった機械学習系サービスは, S3 上に格納されたデータを利用可能
- 各サービスごとに求められるデータフォーマットが異なるため, ニアリアルタイムでデータ変換を行ってから S3 に配信することで, 機械学習系サービスにおける**データ前処理にかかる時間を短縮**

Forecast による時系列予測

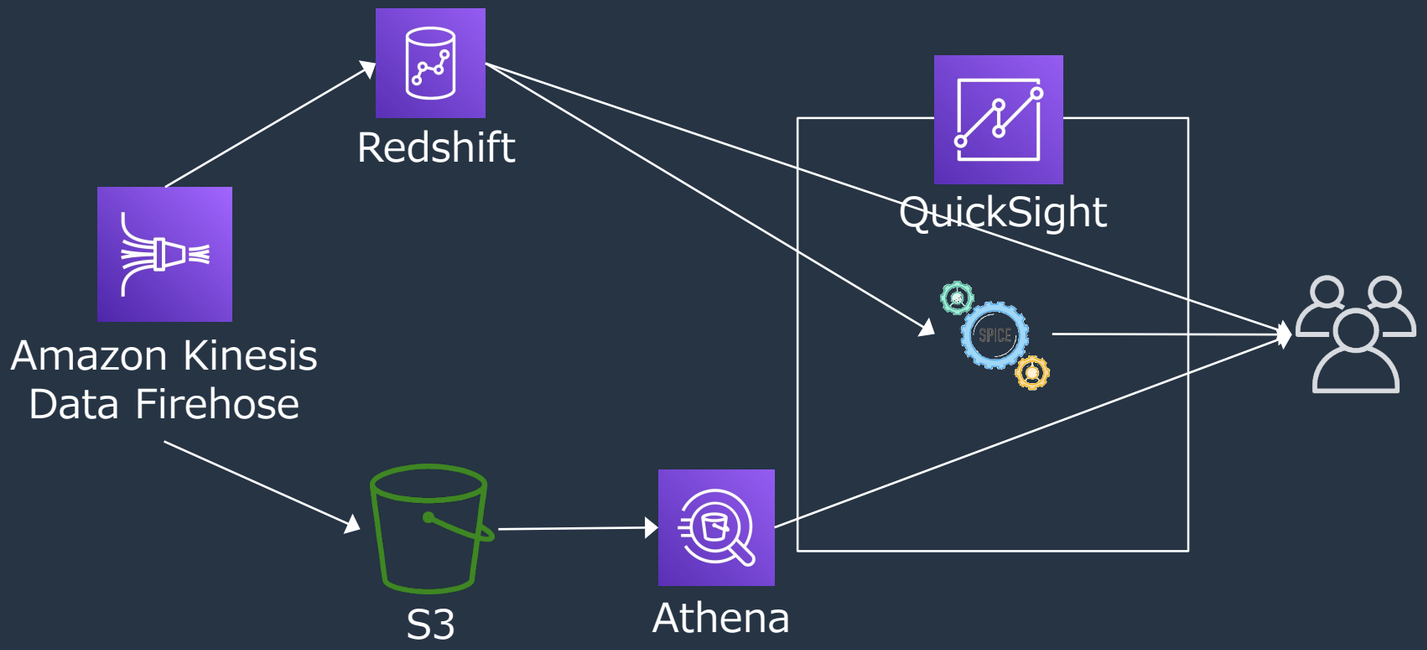


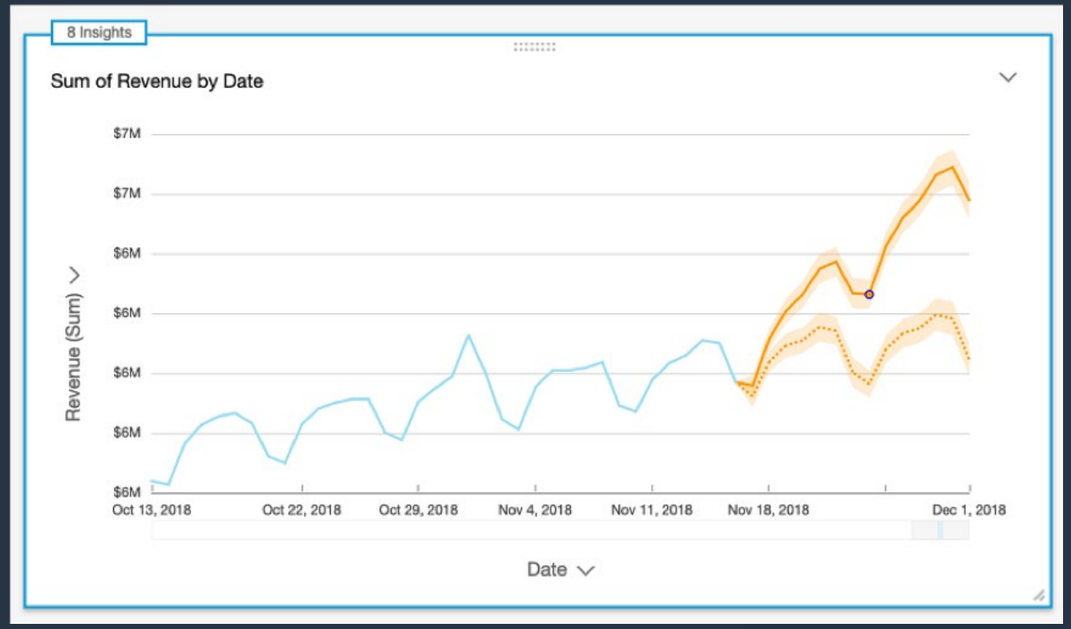


aws DevAx connect

BI ツールによるデータ分析

- Amazon QuickSight はデータ分析に Redshift, Athena + S3 内のデータを使用可能
- ストリーミングデータを逐次 Redshift, S3 に反映することで, QuickSight などの BI ツールで**最新のデータ**を活用した分析が可能になる





aws DevAx connect

Data Stream

Source | **Producer** | **Data Stream** | **Consumer** | **Destination**

External Service
Log Files
Web Brower
Mobile Application
Sensor Device

AWS Amplify
Amazon CloudWatch Logs
Kinesis Producer Library
Kinesis Agent
AWS IoT Core
3rd-party Agents

Amazon Kinesis Data Streams
Amazon Kinesis Data Firehose
Amazon Managed Streaming for Apache Kafka
Amazon DynamoDB Streams
Amazon Kinesis Video Streams

Amazon Kinesis Data Firehose
Amazon Kinesis Data Analytics
Kinesis Client Library
AWS Glue
AWS Lambda

Amazon S3
Amazon Redshift
Amazon RDS
Amazon DynamoDB
Amazon Elasticsearch Service
External Service

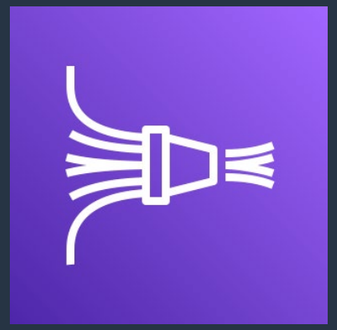
aws DevAx connect

AWS が提供するストリームサービス

Amazon Kinesis Data Streams

Amazon Managed Streaming for Apache Kafka

ストリームデータ収集



Amazon Kinesis Data Firehose

ストリームデータ収集, 加工, データストアへの配信

Amazon Kinesis Video Streams

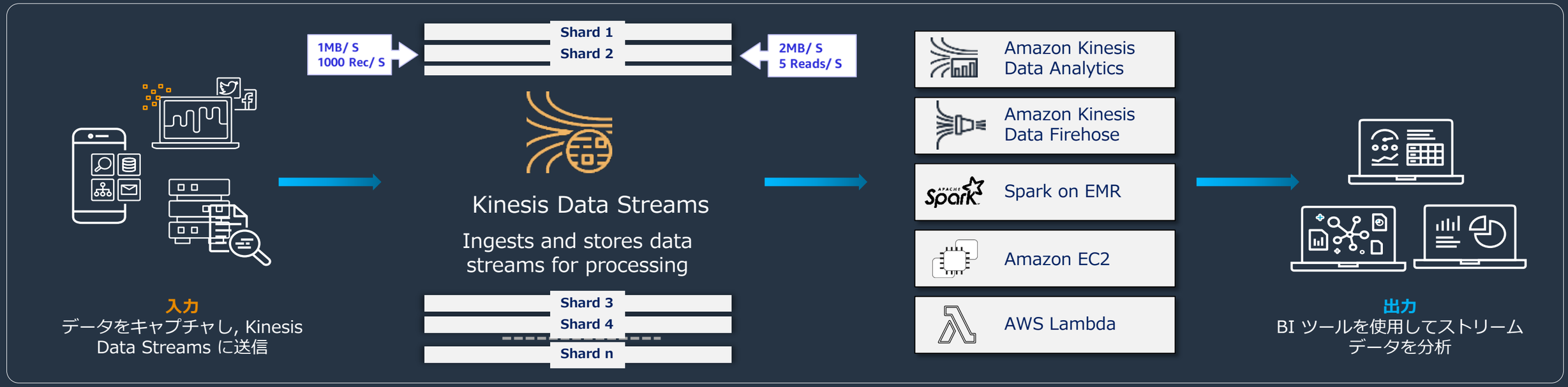
メディアデータ収集

Amazon DynamoDB Streams

DynamoDB テーブル内の項目に対する変更のキャプチャ

Amazon Kinesis Data Streams

大量のストリームデータをリアルタイムで収集するためのデータストリーミングサービス

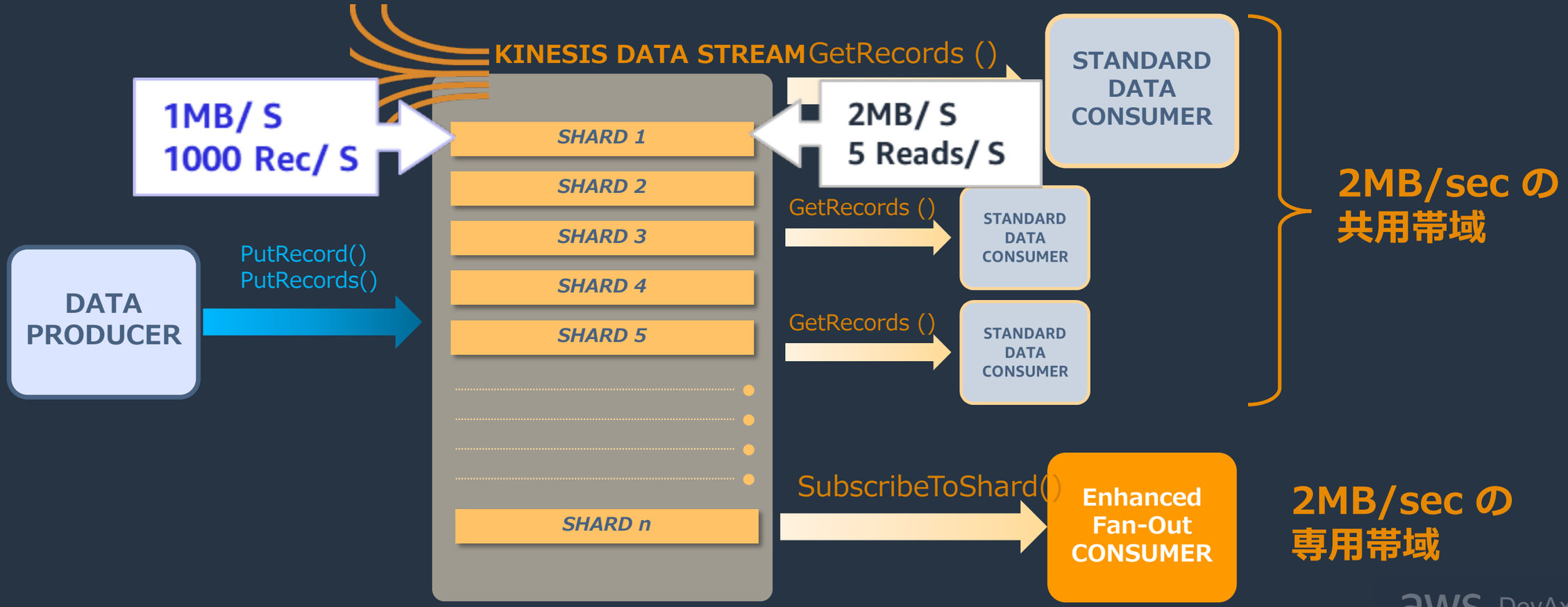


- 高い管理性, 低コスト
- リアルタイム, 弾力的なパフォーマンス
- セキュアで冗長化されたストレージ
- 複数のリアルタイム分析アプリケーションとの連携
- 標準 24 時間, 最長 1 年のデータ保持
- 低レイテンシ
  - 平均 200ms (標準コンシューマー)
  - 平均 70ms (拡張ファンアウト)

aws DevAx connect

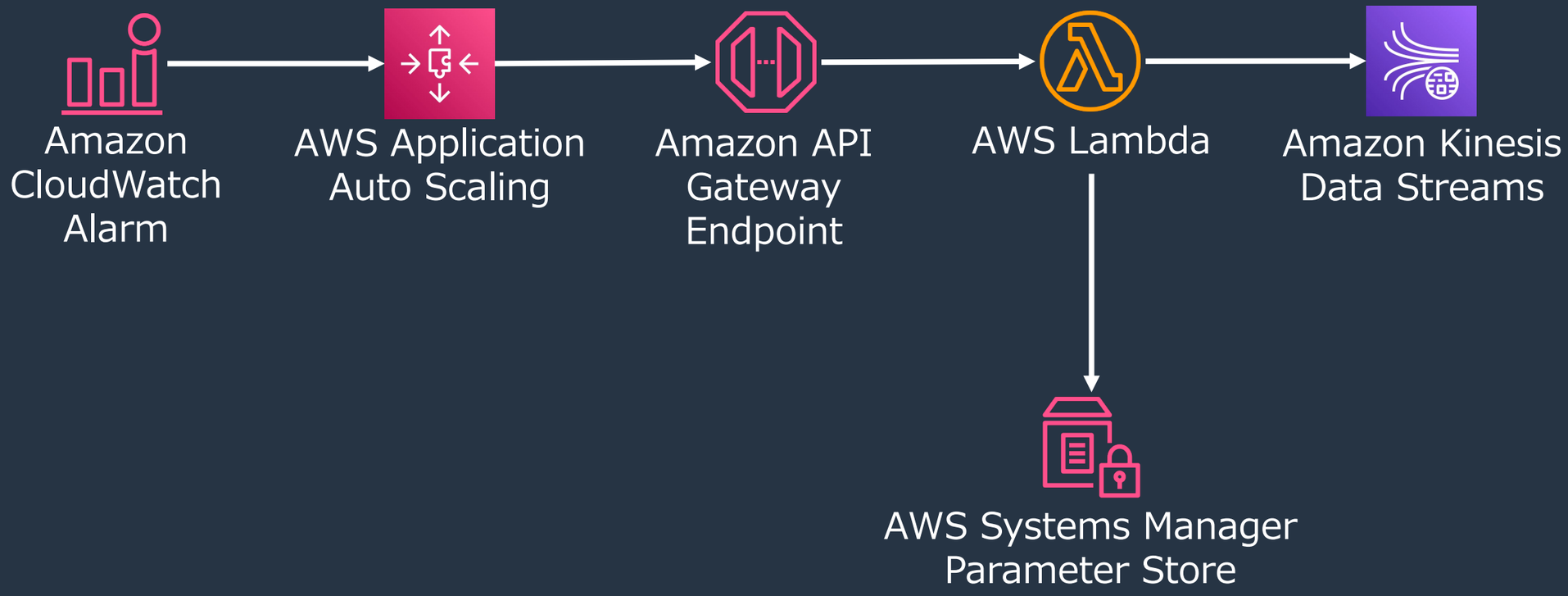
Amazon Kinesis Data Streams のコンセプト

- ストリームは 1 つ以上の “シャード” で構成
- 保存されるデータの単位を “データレコード” と呼ぶ. 1 データレコードの最大サイズは 1 MB
- リソースはシャード単位で割り当てられており,ストリーム内のシャード数を増減することでスループットをコントロールする



Application Auto Scaling による自動スケール

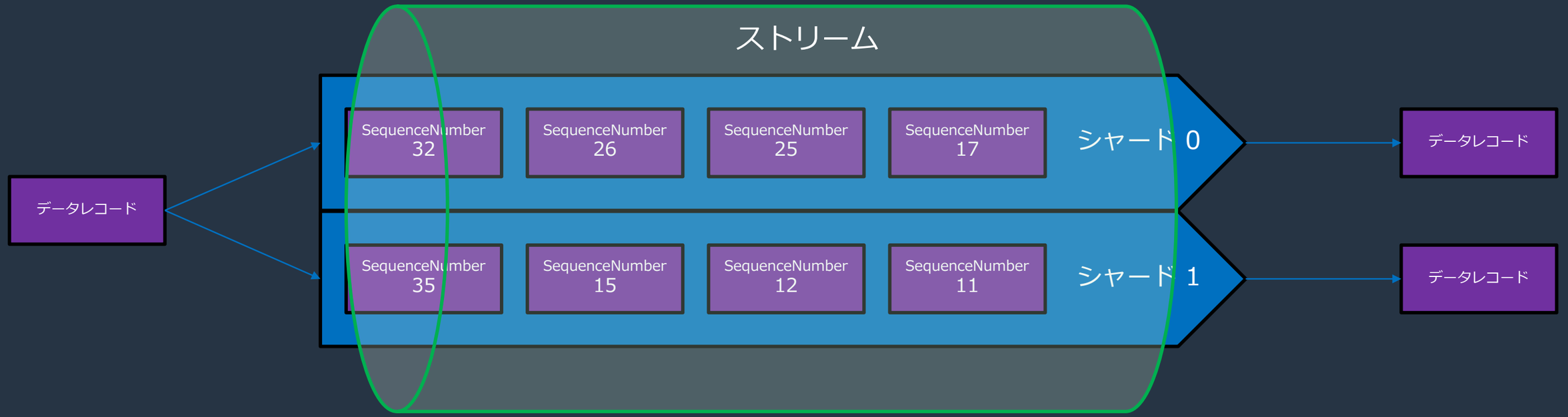
他の AWS サービスとの組み合わせで自動スケールを実現可能



https://aws.amazon.com/jp/blogs/big-data/scaling-amazon-kinesis-data-streams-with-aws-application-auto-scaling/

aws DevAx connect

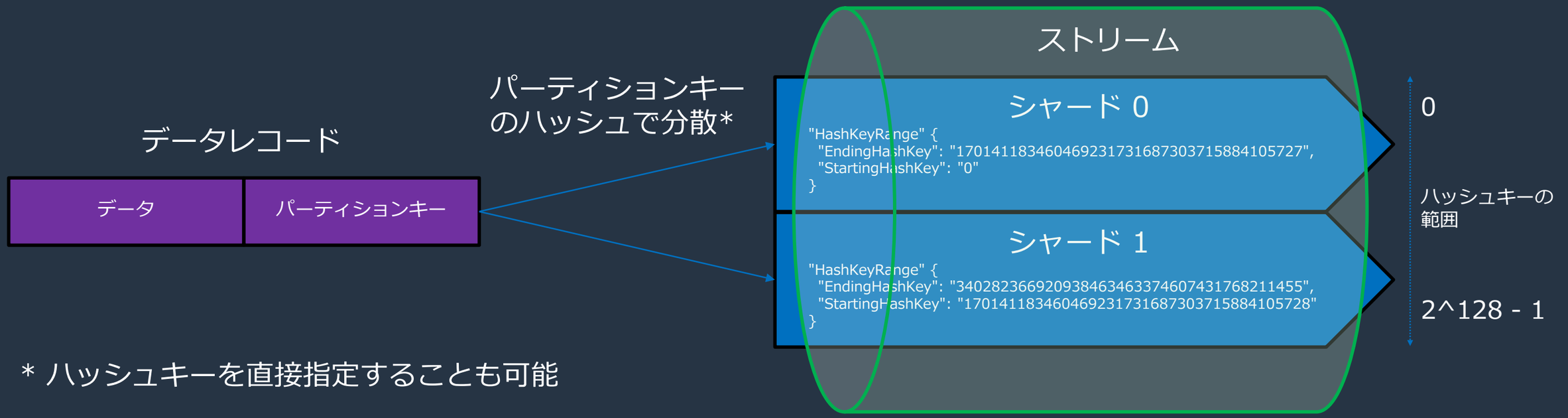
Amazon Kinesis Data Streams データレコードの投入



- Producer から投入されたレコードには, SequenceNumber が付与される
- SequenceNumber はストリーム内の全シャード間でユニーク
- シャード内で時間の経過とともに単調増加（シャード間では単調増加しない）
- Consumer はレコード取得開始ポジションを指定することで, そのポジションから最新のポジションに向かって継続的にレコードを取得し続ける

aws DevAx connect

Amazon Kinesis Data Streams データレコードの分散



* ハッシュキーを直接指定することも可能

- データ入力時に指定するパーティションキー（最長 256 文字）で保存先のシャードが決定
- MD5 ハッシュ関数でパーティションキーを 128 ビット整数値のハッシュキーに変換
- ハッシュキーの範囲に対応したシャードにデータレコードをマップ

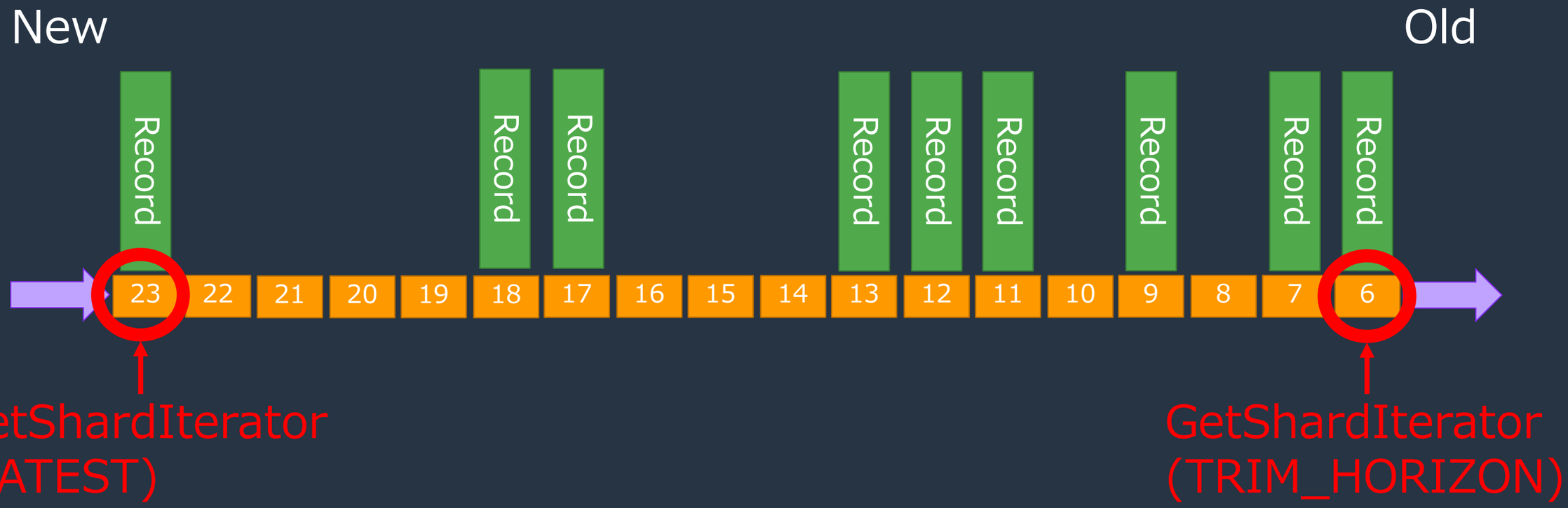
カーディナリティが低いキーをパーティションキーに指定すると, 特定のシャードに書き込みが集中してしまう

aws DevAx connect

Amazon Kinesis Data Streams のレコード取得

取得開始位置を特定してから, レコードを取得していく

1. シャード内のレコード取得開始位置(ShardIterator)を取得
2. 取得したシャード内の位置(ShardIterator)を指定し, レコードを取得

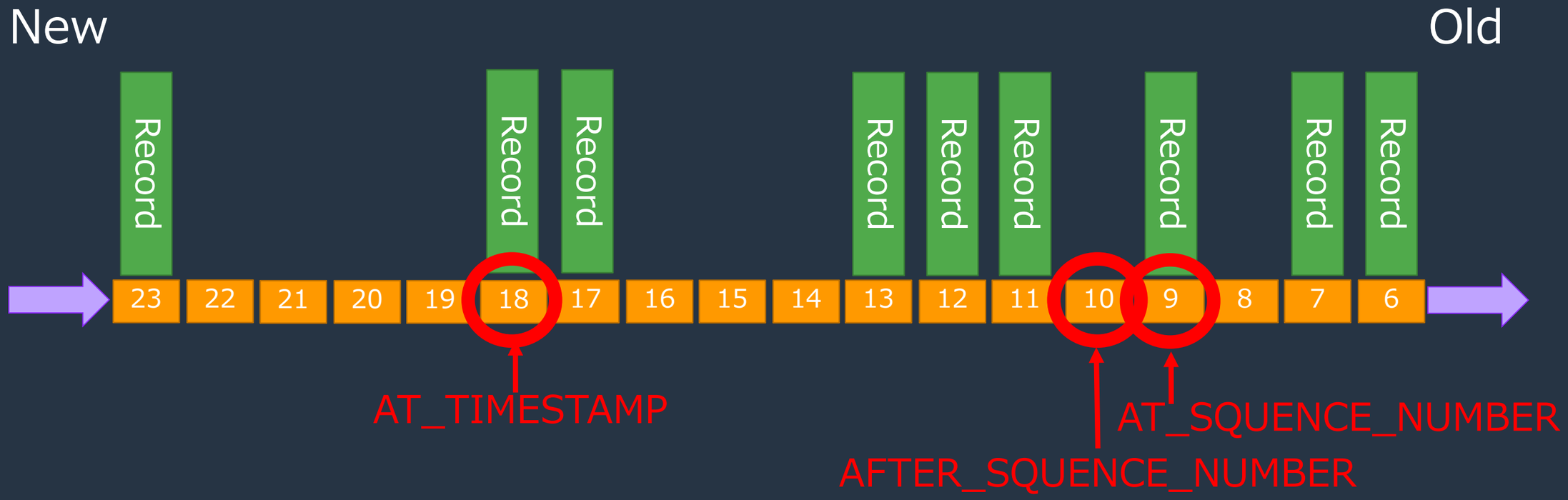


aws DevAx connect

Amazon Kinesis Data Streams のレコード取得

取得開始位置を特定してから, レコードを取得していく

1. シャード内のレコード取得開始位置(ShardIterator)を取得
2. 取得したシャード内の位置(ShardIterator)を指定し, レコードを取得

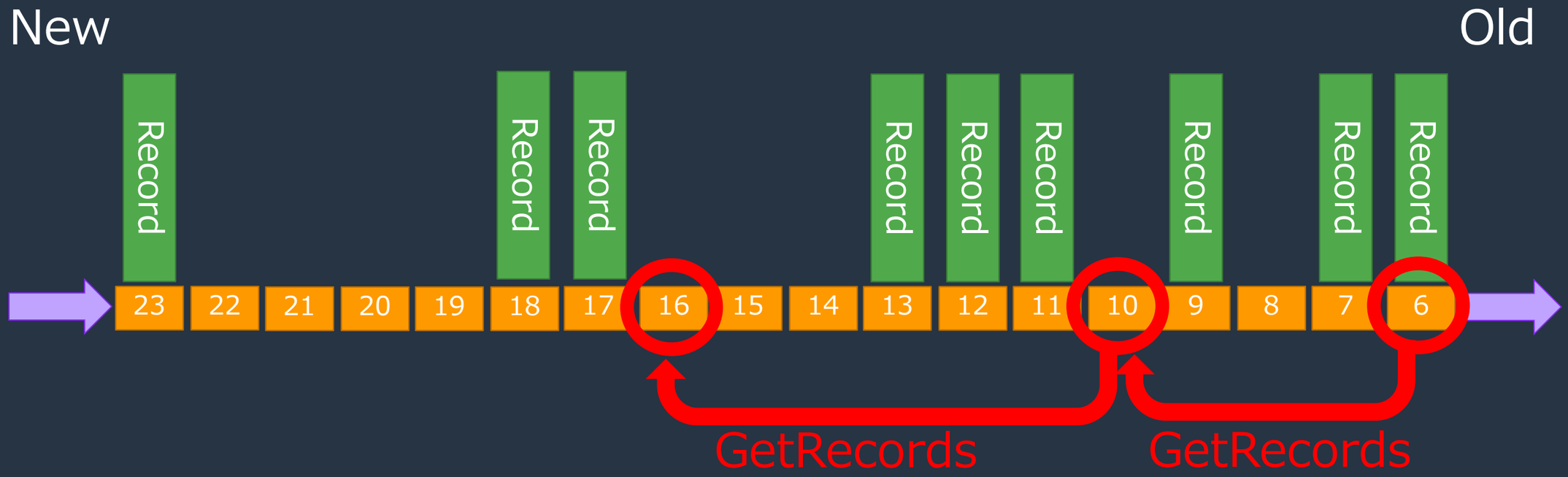


aws DevAx connect

Amazon Kinesis Data Streams のレコード取得

取得開始位置を特定してから, レコードを取得していく

1. シャード内のレコード取得開始位置(ShardIterator)を取得
2. 取得したシャード内の位置(ShardIterator)を指定し, レコードを取得



aws DevAx connect

Amazon MSK(Managed Streaming for Kafka)

フルマネージドで可用性が高くセキュアな Apache Kafka サービス

- Amazon MSK は、コントロールプレーンの操作を提供
  - クラスターの作成, 更新, 削除, ブローカー再起動などの API を提供

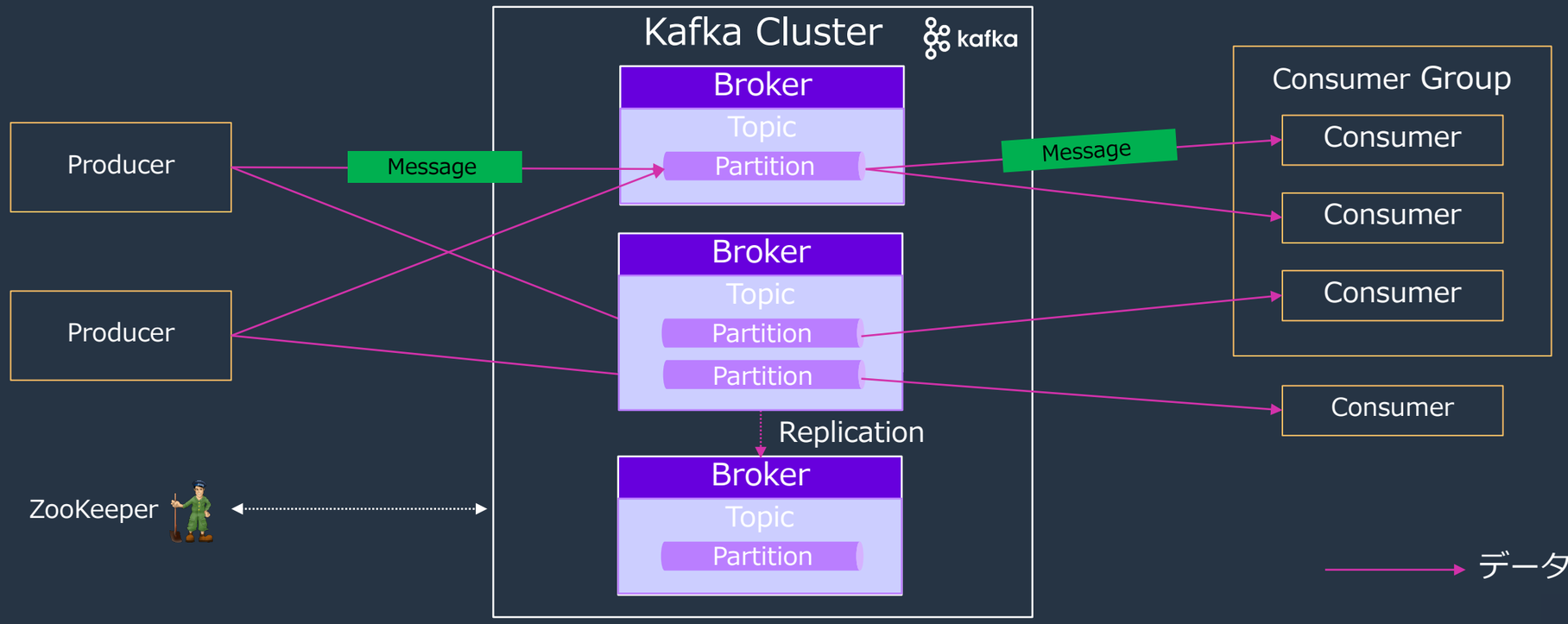
- データプレーンの操作は, Apache Kafka の API をそのまま使用可能
  - トピックの作成や管理
  - プロデューサーからのデータの入力や, コンシューマーからのデータの取得

- Amazon MSK は Apache Kafka のオープンソースバージョンを実行
  - Kafka のバージョン ~~1.1.1~~, 2.2.1, 2.3.1, 2.4.1.1, 2.5.1, 2.6.0, 2.6.1, 2.6.2, 2.7.0, 2.7.1, 2.8.0 をサポート（2021年7月現在）

https://docs.aws.amazon.com/ja_jp/msk/latest/developerguide/supported-kafka-versions.html

Apache Kafkaの全体像

- Kafkaのクラスターは、複数のブローカーで構成され、トピック内のパーティションを分散キューとしてブローカーに配置してメッセージを管理する
- ZooKeeper が、トピックやパーティションのメタ情報を管理する
- プロデューサーはブローカーにメッセージを送信し、コンシューマーはブローカーから取り出して利用する



aws DevAx connect

Kinesis Data Streams と Amazon MSK の比較

Kinesis Data Streams

- **インフラを意識する必要が無い**
- AWS が提供する API で送受信
- AWS サービスとの高度な統合, ライブラリ, エージェントを提供
- **スループット(シャード)をプロビジョニング**
- シームレスなスケーリング

Amazon MSK

- **オンプレミスからの移行が容易**
- オープンソース互換の API で送受信
- Kafka エコシステムを利用可能
- **リソース(ブローカー)をプロビジョニング**
- スケーリング時の考慮事項あり

Kinesis Data Streams と Amazon MSK の使い分け

以下のようなケースでは Amazon MSK の利用を検討

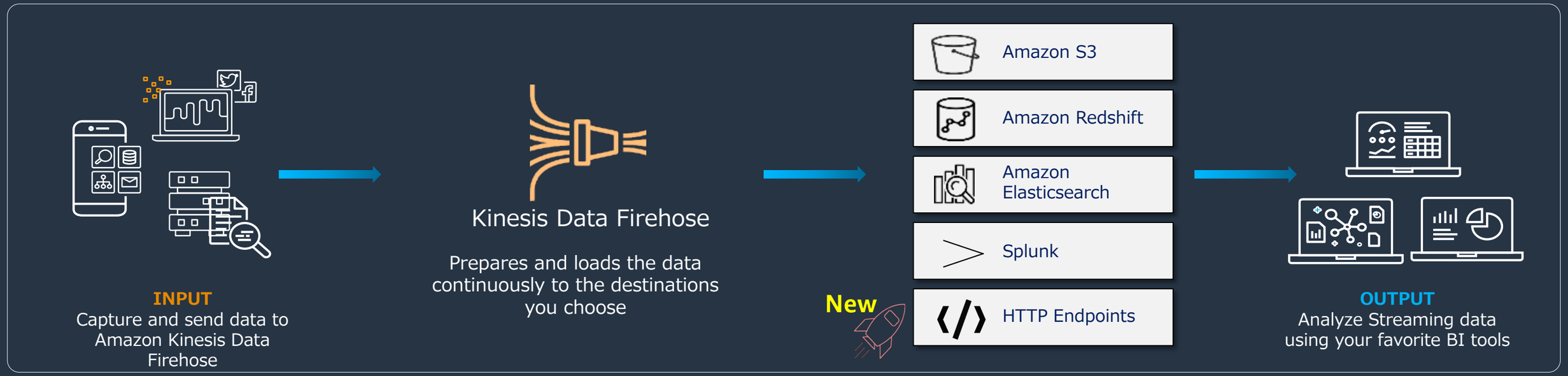
- **オンプレミスにある既存の Apache Kafka クラスターの移行**
- **Amazon EC2 上に構築されている Apache Kafka クラスターの移行**
- **Kafka Connect など Apache Kafka の周辺ツールを利用している場合, または利用したい場合**
- **単一のメッセージサイズが 1MB 以上の場合(Amazon Kinesis Data Streams は 1MB がリミット)**

それ以外の場合には、基本的に Amazon Kinesis Data Streams の利用がお勧め

インフラストラクチャの管理が不要であり, 多数の AWS サービスと統合されているなど, AWS でシステムを構築する際のメリットが多い

Amazon Kinesis Data Firehose

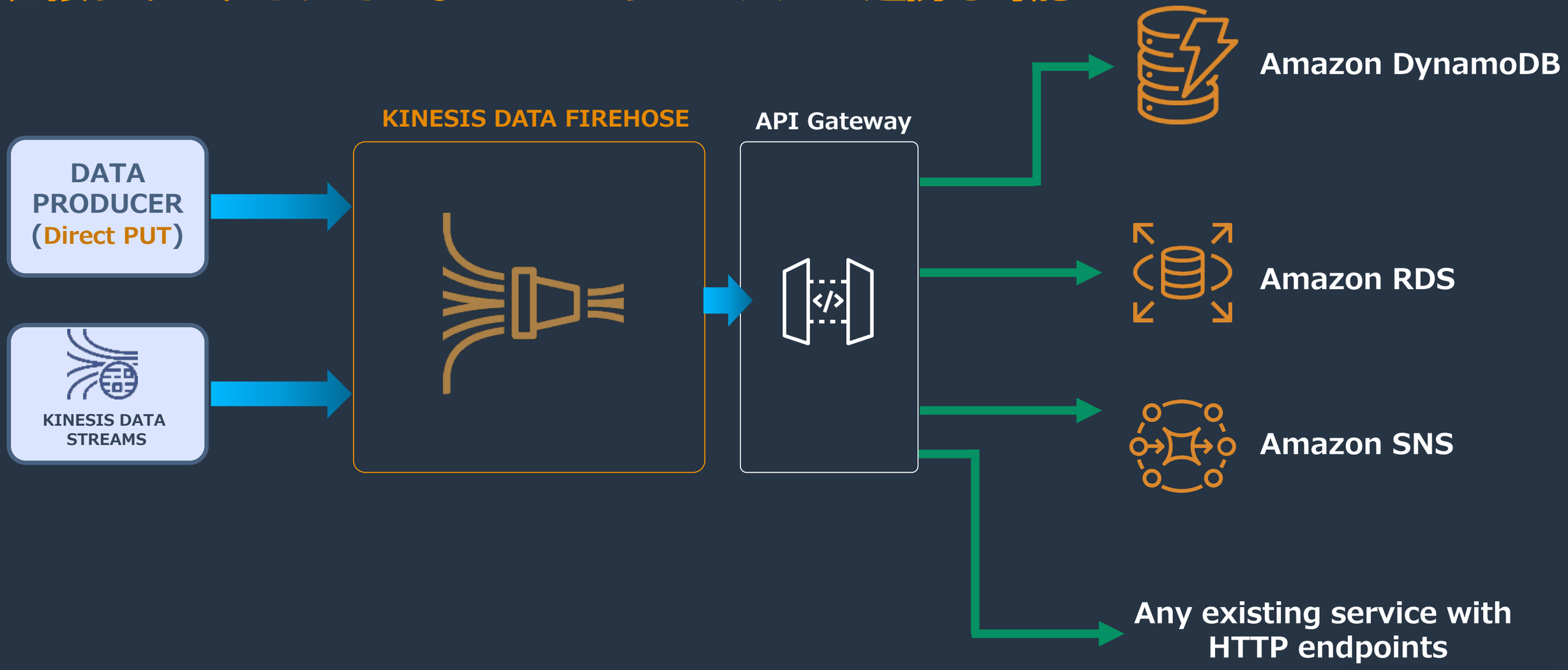
データ変換機能を備えたストリーミングデータ配信サービス



- インフラ管理不要, **自動でスケール**
- AWS サービス, 外部サービス, HTTP エンドポイントへのニアリアルタイムデータ配信機能を統合
- Lambda と統合されたデータ変換機能, Glue と連携した Parquet/ORC へのデータフォーマット変換機能を提供

Kinesis Data Firehose to API Gateway

直接サポートされていない AWS サービスへの連携も可能

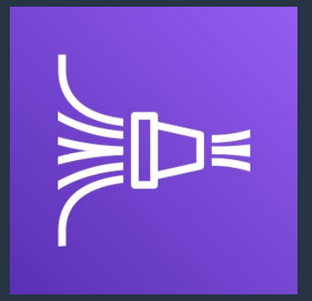


aws DevAx connect

Kinesis Data Streams と Kinesis Data Firehose の比較

Kinesis Data Streams

- 任意の Consumer を組み合わせることで複雑な処理を実現
- シャードを意識したキー設計が必要
- レイテンシを1秒未満に抑えることが可能



Kinesis Data Firehose

- ローコード or ノーコードでのデータ配信に特化
- シャードは意識する必要がない
- Buffer にデータを蓄積し配信するためレイテンシは 60 秒以上

Kinesis Data Streams と Kinesis Data Firehose の使い分け

Kinesis Data Firehose は Push 型配信, Kinesis Data Streams は Pull 型配信

以下のようなケースでは Kinesis Data Firehose の利用を検討

- データストア, 外部サービスに対してシンプルにデータ配信を行いたい
- 複雑なリアルタイム集計や分析が求められない
- バッファによる遅延 (1 分 ~ 15 分) が許容できる

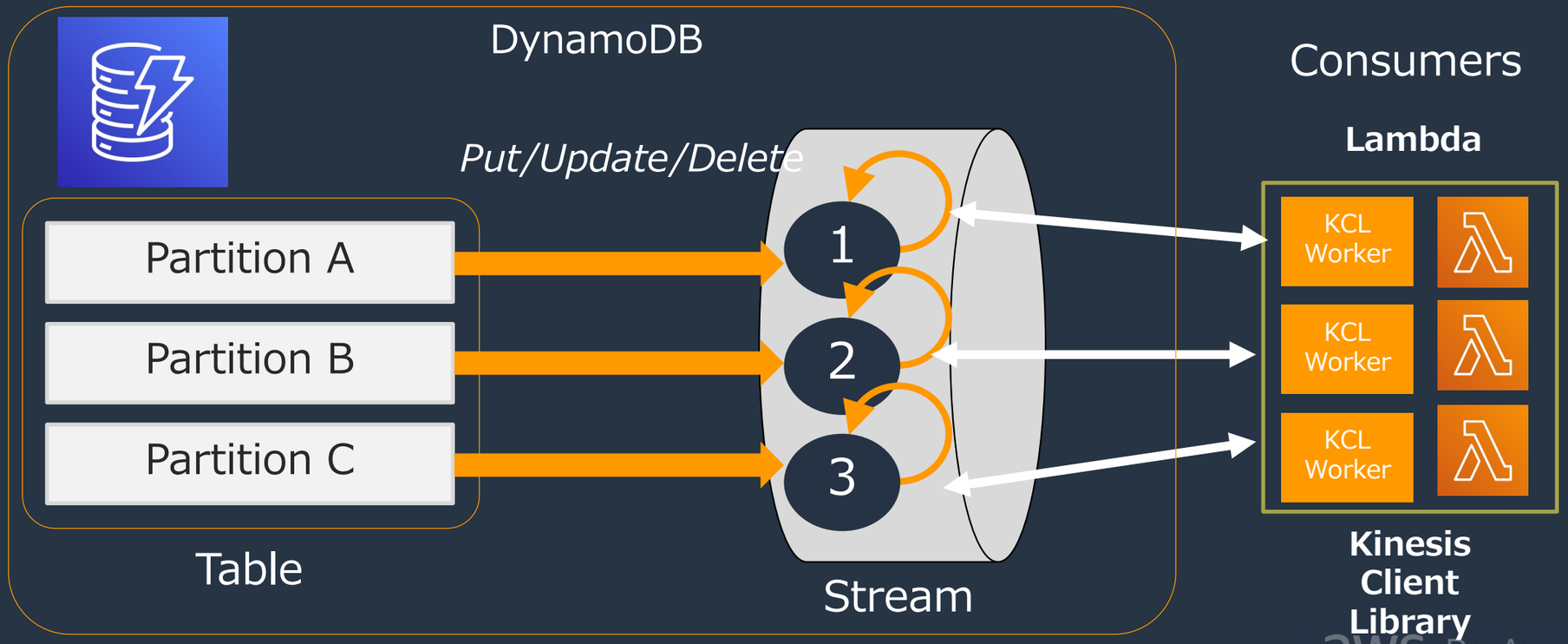
以下のようなケースでは Kinesis Data Streams の利用を検討

- より柔軟性, 複雑性の高い処理が求められる場合
- データ配信は行わず分析に特化する場合
- バッファによる遅延 (1 分 ~ 15 分) が許容できない. より低遅延でデータを処理したい

DynamoDB Streams

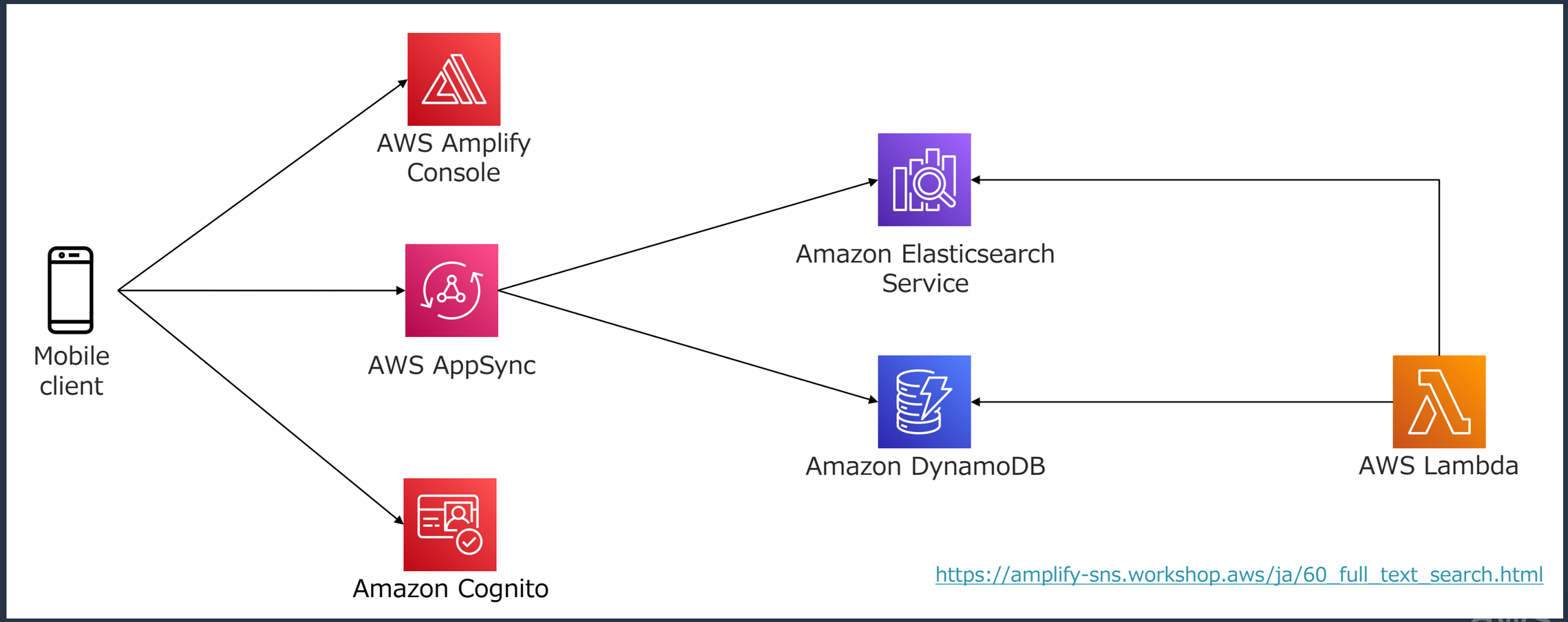
- フルマネージド NoSQL データベースサービス DynamoDB の一機能
- テーブル内のアイテムの変更情報をキャプチャし, ストリームに保存
- 保存されたデータは Kinesis Client Library または Lambda で処理可能

特徴
✓Exactly-Once, 厳密な並び順
✓高い耐久性, スケーラビリティ
✓24 時間データを保持
✓レイテンシは1秒未満



DynamoDB Streams 活用例

- チャットアプリケーションにおけるチャット履歴の全文検索
- 格納された履歴データを DynamoDB Streams 経由で Amazon Elasticsearch Service にニアリアルタイムで反映

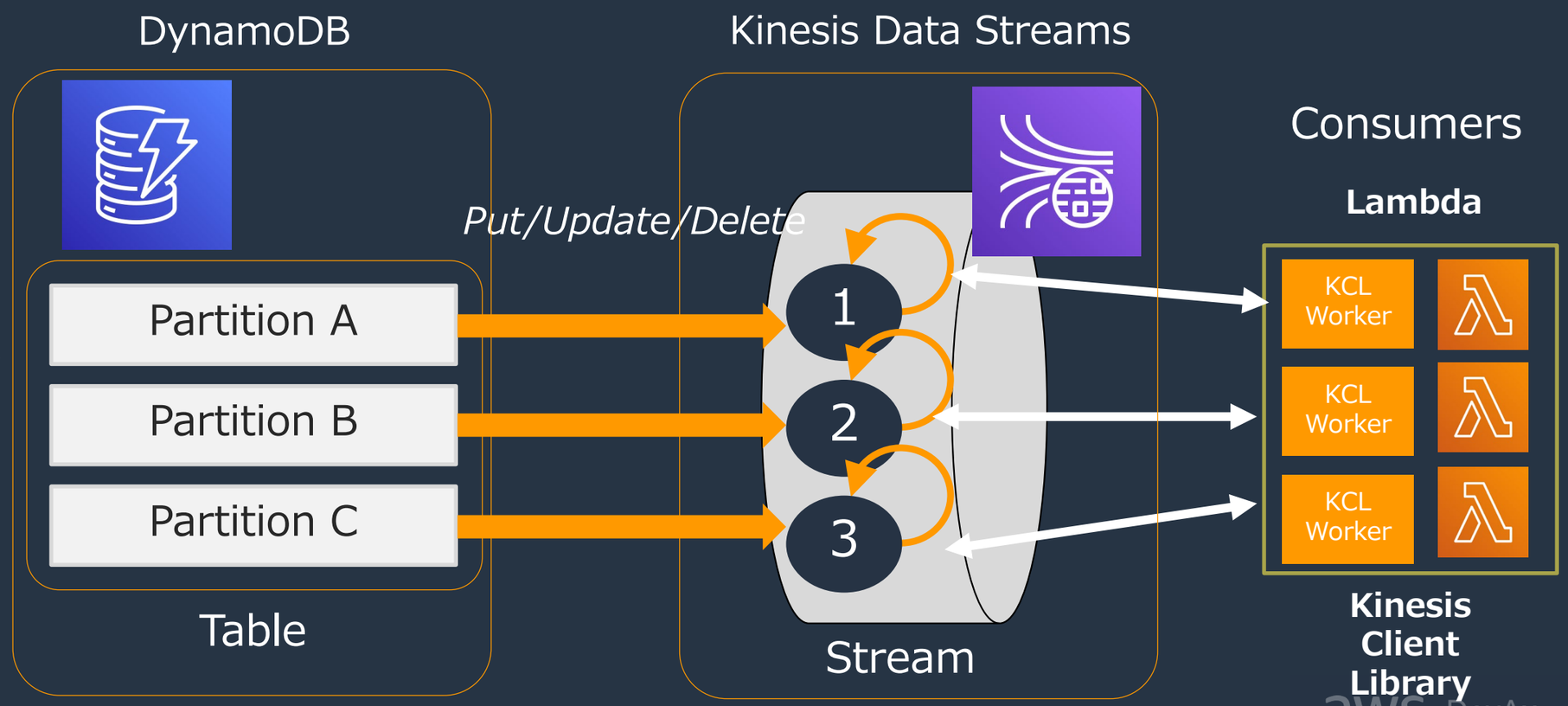


https://amplify-sns.workshop.aws/ja/60_full_text_search.html

DynamoDB + Kinesis Data Streams

New!

- 直近のアップデートで変更情報を Kinesis Data Streams との連携をサポート
- 従来の DynamoDB Streams との使い分けが可能に



特徴

✓多数の Consumer をサポート
✓高いスケーラビリティ
✓最長 1 年データを保持

DynamoDB Streams と Kinesis Data Streams の使い分け

- 24時間以上のデータ保持, AWS サービスとのより柔軟な連携, Consumer による複雑な処理が必要な場合は Kinesis Data Streams を検討
- Exactly Once やデータの並び順などが重要な場合は DynamoDB Streams を検討

プロパティ	DynamoDB + Kinesis Data Streams	DynamoDB Streams
データ保持期間	24時間 から 1 年	24 時間
コンシューマーの数	最大 5 (標準コンシューマー) 最大20(拡張ファンアウト)	最大 2 つまで
レコードの順序	Consumer の方で, レコードタイムスタンプから順序を識別する	ストリーム内のレコードは, 実際に発生した変更と同じ順序で出力される
レコード重複の有無	重複は起こりうる (At Least Once)	重複は発生しない (Exactly Once)
拡張性	別途でスケールする必要あり	テーブルに追従して自動的にスケール
Consumer	- Kinesis Data Firehose - Kinesis Data Analytics for SQL - Kinesis Data Analytics for Flink - Kinesis Client Library - AWS Glue Streaming ETL Jobs - AWS Lambda	- DynamoDB Streams Kinesis Adapter - Kinesis Data Analytics for Flink - AWS Lambda

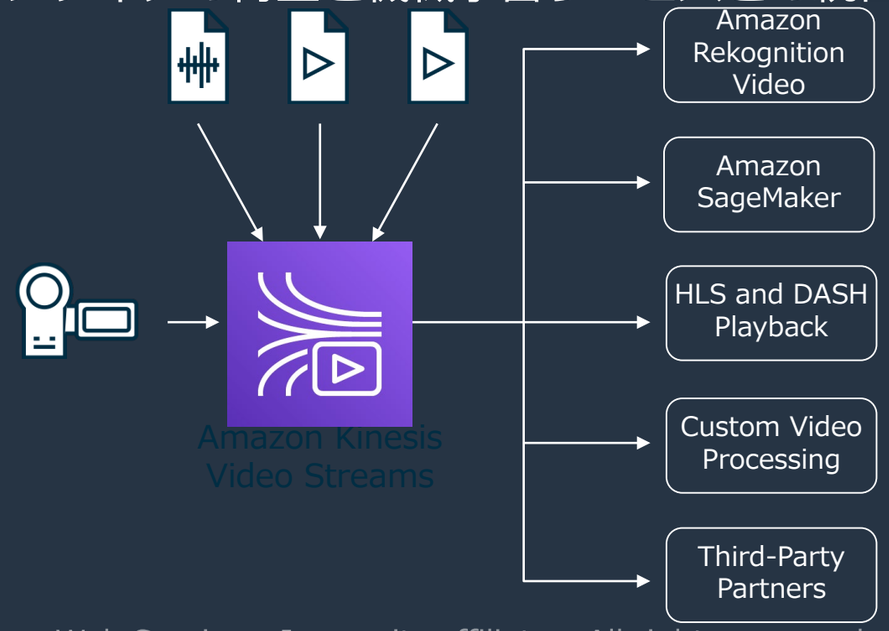
Amazon Kinesis Video Streams

2種類のストリーミング方法に対応したメディアストリームのキャプチャ, 保存, 処理サービス

メディア形式で収集

数百万台規模のデバイスからのセキュアなデータ取り込み

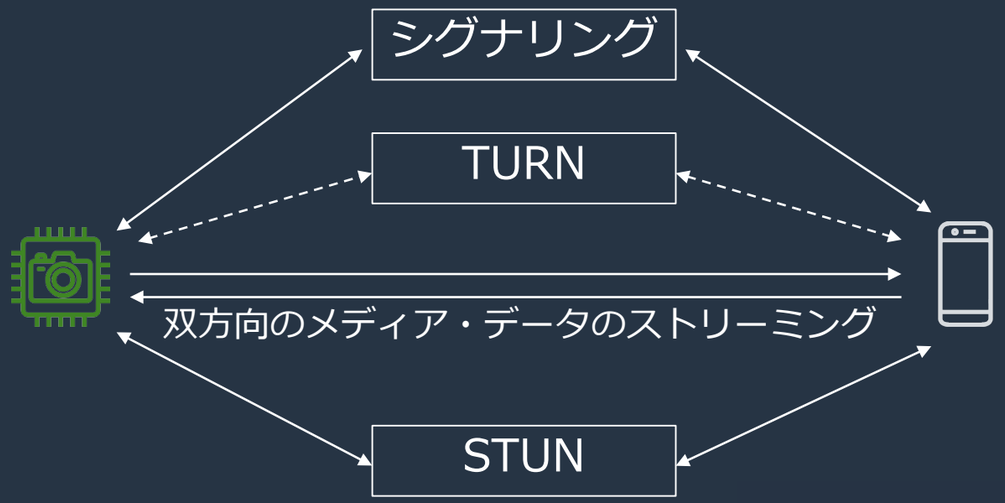
- 時系列でインデックスされたメディアデータの保存とAPI経由での検索
- カメラデバイス向けのSDK
- メディアの再生と機械学習サービスとの統合



WebRTC

WebRTC によるリアルタイムの双方向メディアストリーミング

- シグナリング、STUN、TURN のマネージドサービス
- 組み込みデバイス向けSDK、ウェブ・モバイルアプリ向けのSDK



aws DevAx connect

Producer

Source | **Producer** | **Data Stream** | **Consumer** | **Destination**

External Service

Log Files

Web Brower

Mobile Application

Sensor Device

AWS Amplify

Amazon CloudWatch Logs

Kinesis Producer Library

Kinesis Agent

AWS IoT Core

3rd-party Agents

Amazon Kinesis Data Streams

Amazon Kinesis Data Firehose

Amazon Managed Streaming for Apache Kafka

Amazon DynamoDB Streams

Amazon Kinesis Data Firehose

Amazon Kinesis Data Analytics

Kinesis Client Library

Amazon Kinesis Video Streams

AWS Glue

AWS Lambda

Amazon S3

Amazon Redshift

Amazon RDS

Amazon DynamoDB

Amazon Elasticsearch Service

External Service

aws DevAx connect

Producer 一例(Amazon Kinesis)

プログラムからの直接送信, エージェントによるファイル読み込みとデータ転送, 別の AWS サービス経由でのデータ送信など, 用途別に様々な方法が提供されている

エージェント, ライブラリ

AWS SDK

Kinesis Producer Library

AWS Amplify Libraries

Kinesis Agent

AWS サービス(一部)

AWS IoT

Amazon CloudWatch Logs

Amazon EventBridge

New

Amazon Dynamo DB

Amazon API Gateway

サードパーティーツール

LOG4J

Flume

Fluentd

Fluent Bit

Producer 選定のポイント (Amazon Kinesis)

① AWS サービスの Kinesis 連携機能を活用する

Amazon API Gateway

AWS IoT

Amazon EventBridge

Amazon Pinpoint

AWS WAF

その他多数

② ログ連携を行う際は, プラットフォームにフィットしたツールを使用する

各種ログファイル

Kinesis Agent

Fluentd

コンテナアプリケーションが出力するログメッセージ

Fluent Bit

AWS サービスが出力するログメッセージ

Amazon CloudWatch Logs

③アプリケーションから直接メッセージを送信する場合は, ライブラリを活用する

モバイルアプリケーション, SPA

AWS Amplify Libraries

iOS Android JavaScript Flutter

サーバーサイドアプリケーション

AWS SDK

Kinesis Producer Library

aws DevAx connect

Amazon Kinesis Agent

OSS のスタンドアロン Java アプリケーション

- Amazon Kinesis Data Streams, Amazon Kinesis Data Firehose を宛先としてサポート
- パスで指定したファイルを検出し, 差分を順次転送する. ファイルローテートにも対応
- 送信前のバッファリング, 失敗時の再試行をサポート. またデータのフォーマット変換や, ログパースなどの前処理機能を提供
- Agent の実行に関するメトリクスデータをAmazon CloudWatch へ送信

https://github.com/awslabs/amazon-kinesis-agent

```
# /etc/aws-kinesis/agent.json
{
   "kinesis.endpoint":
"https://your/kinesis/endpoint",
   "firehose.endpoint":
"https://your/firehose/endpoint",
   "flows": [
      {
         "filePattern": "/tmp/app1.log*",
         "kinesisStream": "yourkinesisstream"
      },
      {
         "filePattern": "/tmp/app2.log*",
         "deliveryStream":
"yourfirehosedeliverystream"
      }
   ]
}
```

Fluent plugin for Amazon Kinesis v3

Fluentd から Kinesis にメッセージを直接送信するためのプラグイン

3 つの output をサポート

Kinesis Data Streams

1. kinesis_streams
2. kinesis_streams_aggregated

Kinesis Data Firehose

3. kinesis_firehose

```
# Kinesis Data Streams 用の設定例
<match your_tag>
  @type kinesis_streams
  region us-east-1
  stream_name your_stream
  partition_key key
</match>
```

```
# Kinesis Data Firehose 用の設定例

<match your_tag>
    @type kinesis_firehose
    region us-east-1
    delivery_stream_name your_stream
</match>
```

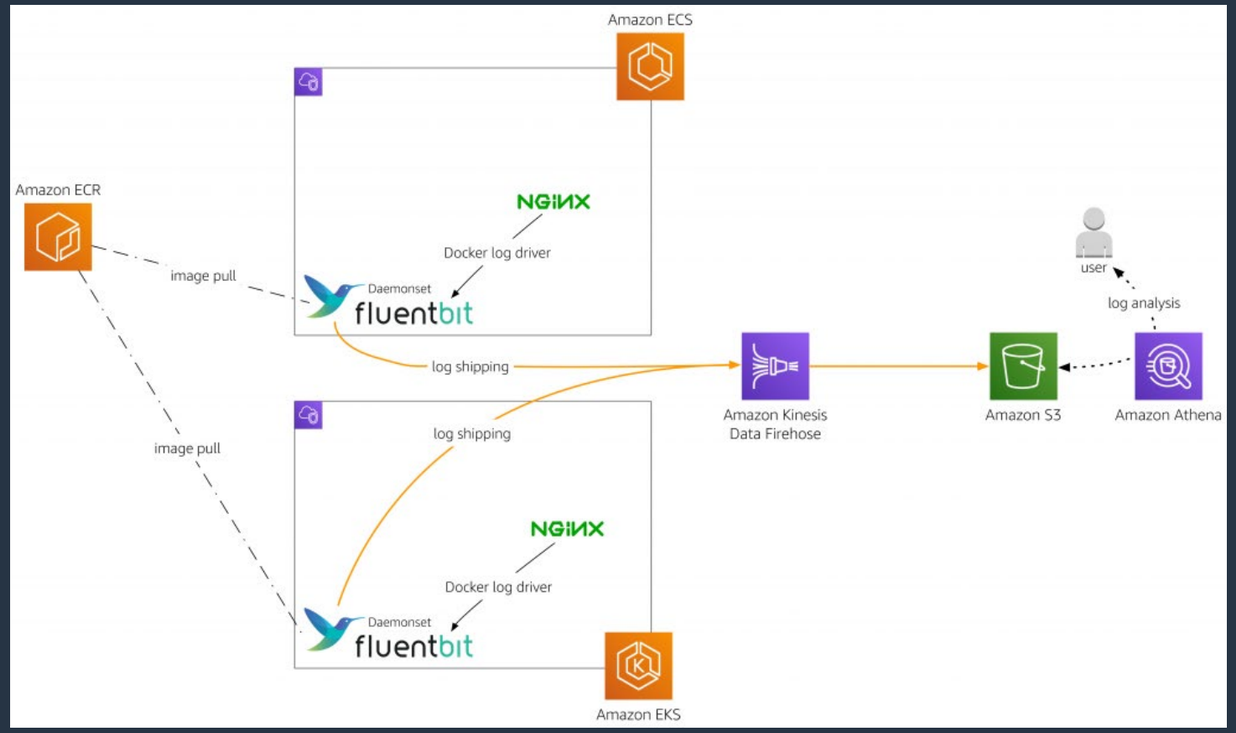
https://github.com/awslabs/aws-fluent-plugin-kinesis

aws DevAx connect

AWS for Fluent Bit

- コンテナサービスのログを他の AWS サービスに送信するためのツール
- Fluentd より低いリソース使用率が特徴
- ECS, EKS に対応. Fargate プラットフォーム上での実行もサポート
- Kinesis Data Streams, Kinesis Data Firehose, CloudWatch, Amazon ES などを宛先としてサポート

https://github.com/aws/aws-for-fluent-bit



aws DevAx connect

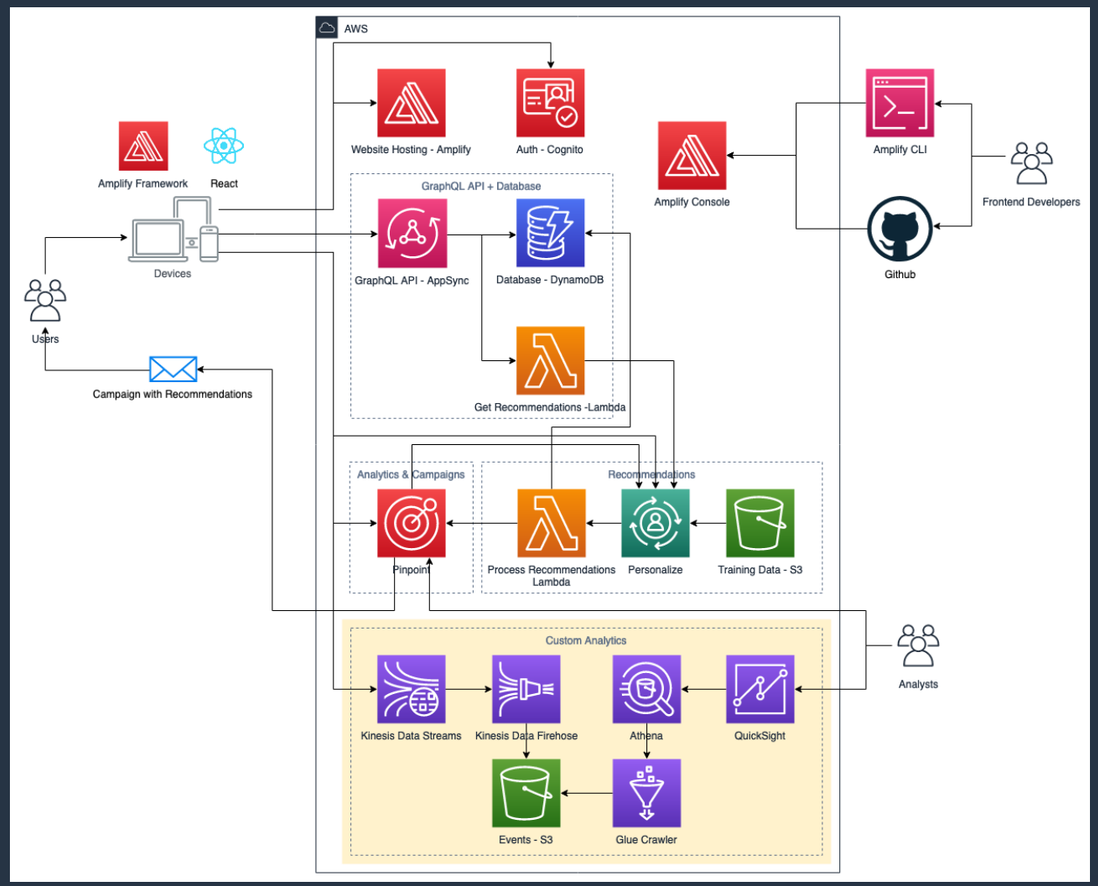
AWS Amplify / AWS Mobile SDK

- Web アプリケーション, モバイルアプリケーションにおいて発生したイベントを AWS サービスに連携するライブラリ

- 連携先として Kinesis Data Firehose, Kinesis Data Streams のほか, Amazon Pinpoint もサポート

- 連携したデータを他の AWS サービスで分析することで ユーザーの行動分析やレコメンデーションを実現

https://amplify-analytics.workshop.aws



aws DevAx connect

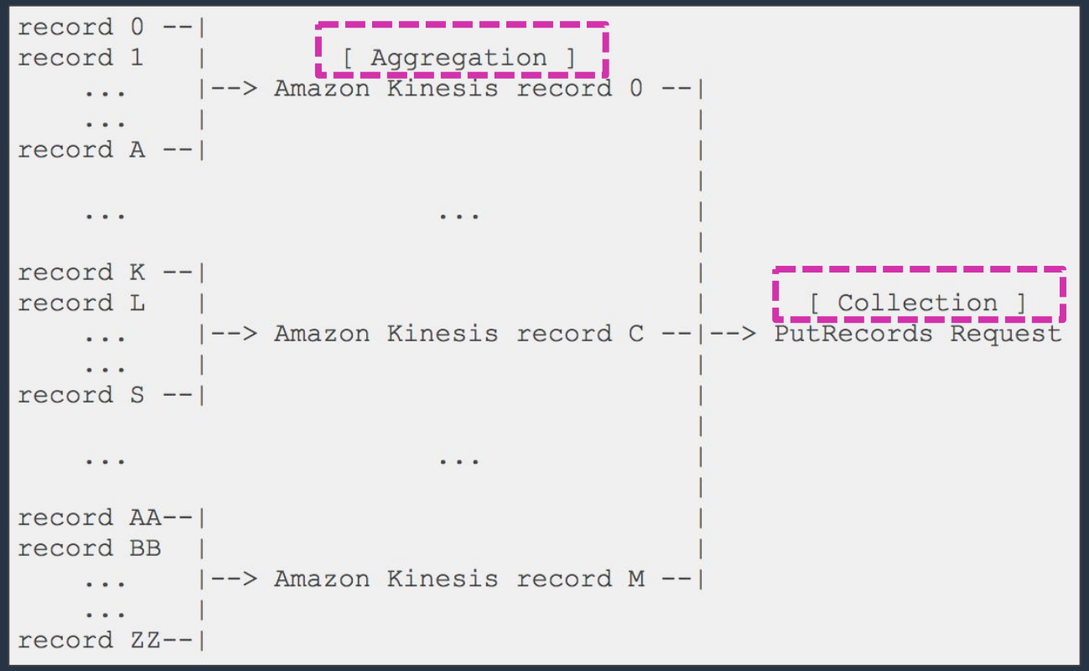
Amazon Kinesis Producer Library (KPL)

- Amazon Kinesis Data Streams にデータを送信する OSS の補助ライブラリ

- Amazon Kinesis Data Streams の特性に合った高い転送効率を実現する機能を提供
  - 複数件のデータを 1 データレコードに集約
  - 複数データレコードをバッファリングして送信

- 各種設定をプロパティベース調整可能
  - エラー時のリトライ挙動
  - タイムアウト時間の調整
  - データ送信時に利用するコネクション数
    など

- パフォーマンスメトリクスを Amazon CloudWatch に自動送信

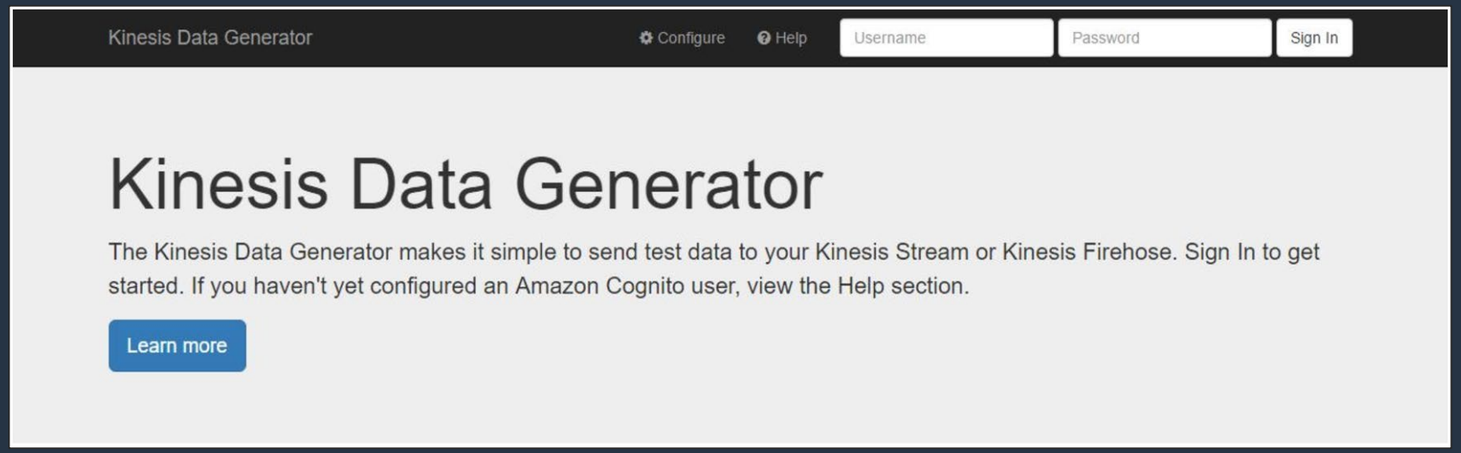


https://github.com/awslabs/amazon-kinesis-producer

aws DevAx connect

Amazon Kinesis Data Generator (KDG)

- HTML と JavaScript で実装されたOSSのテスト用プロデューサーUI
- Amazon Kinesis Data Streams または Amazon Kinesis Data Firehose にテストデータを簡単に送信できる
- GithubにホストされたUIを利用することが可能
- S3 静的ウェブサイトホスティングを利用するなども可能



https://github.com/awslabs/amazon-kinesis-data-generator

aws DevAx connect

Consumer

Source | **Producer** | **Data Stream** | **Consumer** | **Destination**

External Service
Log Files
Web Brower
Mobile Application
Sensor Device

AWS Amplify
Amazon CloudWatch Logs
Kinesis Producer Library
Kinesis Agent
AWS IoT Core
3rd-party Agents

Amazon Kinesis Data Streams
Amazon Kinesis Data Firehose
Amazon Managed Streaming for Apache Kafka
Amazon DynamoDB Streams
Amazon Kinesis Video Streams

Amazon Kinesis Data Firehose
Amazon Kinesis Data Analytics
Kinesis Client Library
AWS Glue
AWS Lambda

Amazon S3
Amazon Redshift
Amazon RDS
Amazon DynamoDB
Amazon Elasticsearch Service
External Service

aws DevAx connect

代表的な Consumer

Kinesis 系統のサービス, ライブラリのほか, Lambda や EMR, Glue など他の AWS サービスでもストリームデータを処理可能. 対応する 3rd party ツールを活用することも可能.

Kinesis

Amazon Kinesis Data Analytics

Flink

beam

SQL

Kinesis Client Library + Connector Library

Kinesis Data Firehose

AWS Services

AWS Lambda

Amazon EMR

AWS Glue

3rd party

Apache Spark

APACHE STORM™ Distributed • Resilient • Real-time

DATADOG

mongoDB.

Anodot

databricks

Qubole

VOLTDB

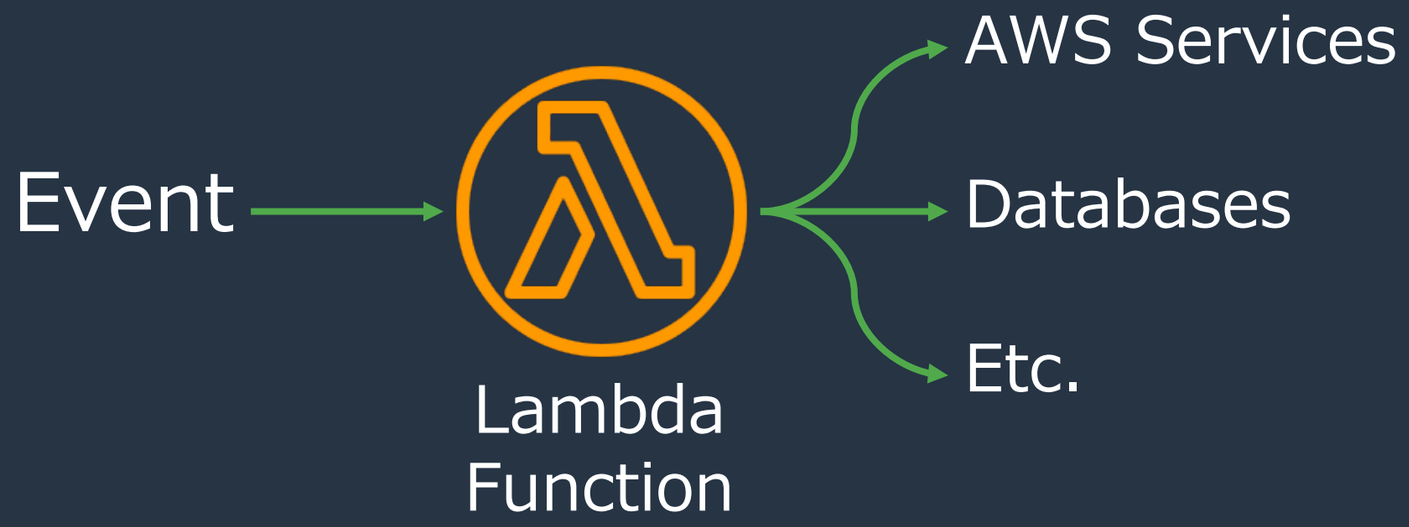
MEMSQL

splunk>

AWS Lambda

様々なイベントをトリガに任意のコードを実行するサービス

- 多数のプラットフォームを用意. 自前のコンテナイメージを使用することも可能
- コード実行時間(ミリ秒単位)と割り当てリソース(メモリ, vCPU)に応じて課金

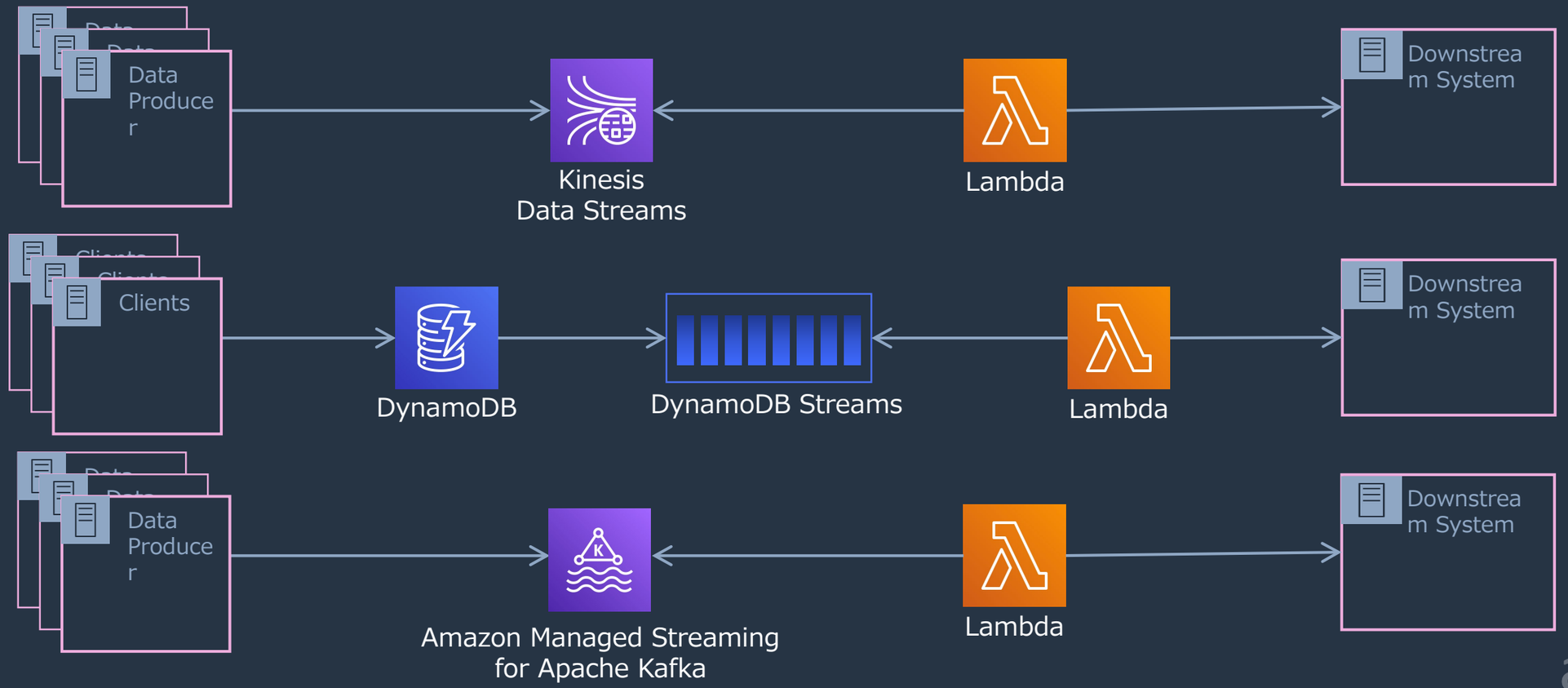


Python, Javascript
Java, Golang, C#
BYOL, Container images

aws DevAx connect

Lambda Consumer

- Kinesis Data Streams, DynamoDB Streams, Amazon MSK, EC2 インスタンス上またはオンプレミス環境で実行される Kafka をサポート
- リトライ回数・リトライ期間・バッチサイズの調整, リトライ上限超過時の SQS および SNS への連携など, 様々な機能を Lambda 側で提供している

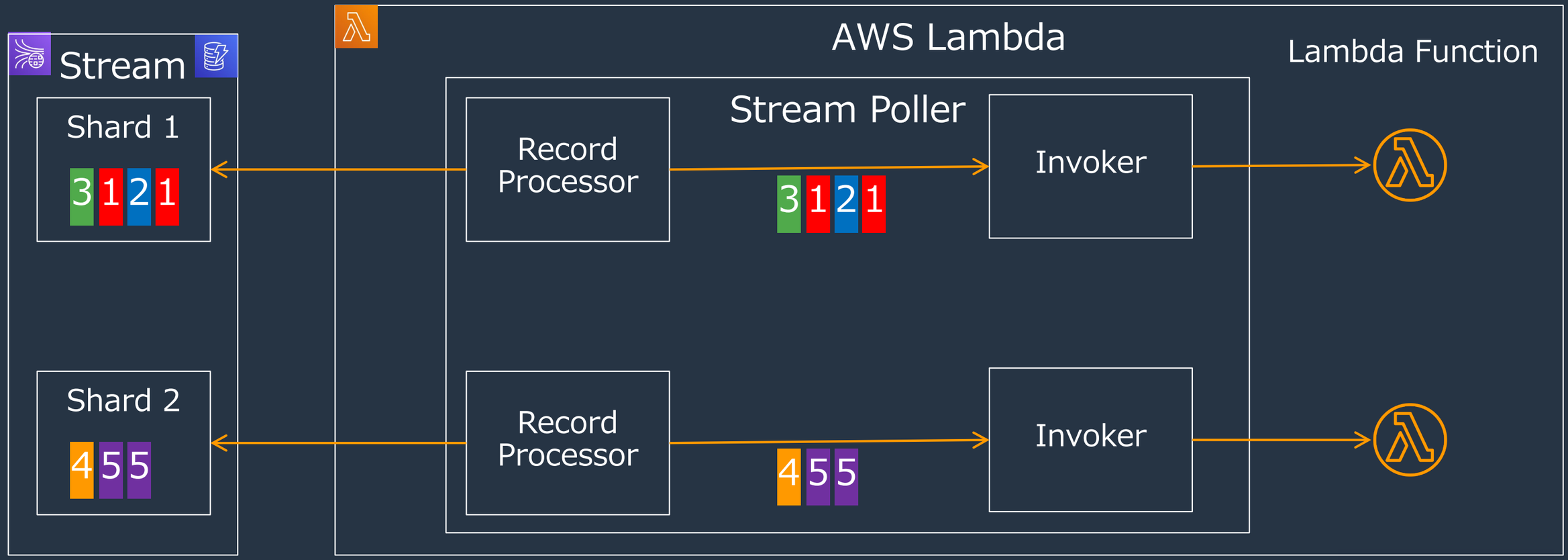


aws DevAx connect

Lambda Consumer のアーキテクチャ

Kinesis Data Streams, DynamoDB Streams の場合

- Stream Poller と呼ばれる内部サービスがストリームに対して定期的にポーリングを実施し,レコードを取得する
- 取得されたレコードは, Lambda Function 呼び出し時のイベントとして渡される
- デフォルトでは, Shard 数に応じて Lambda Function 呼び出しの並列数が決まる

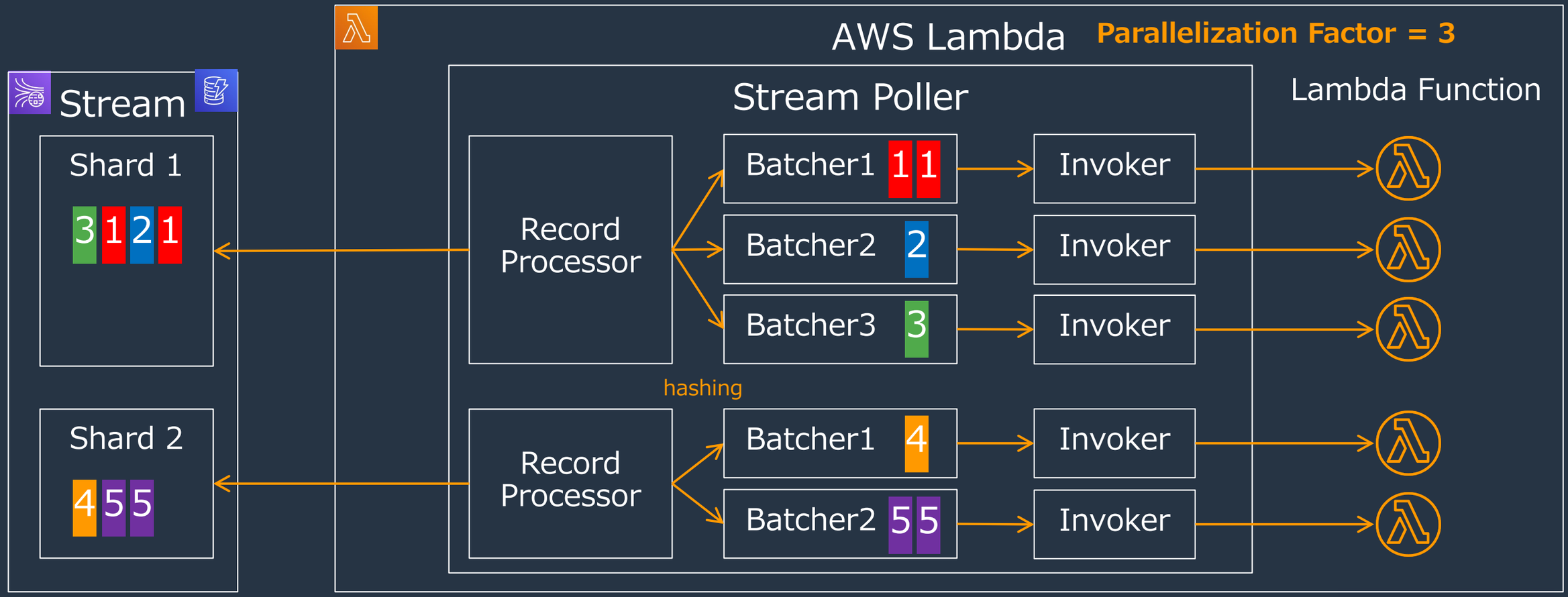


aws DevAx connect

Lambda Consumer のスケーラビリティ

Kinesis Data Streams, DynamoDB Streams の場合

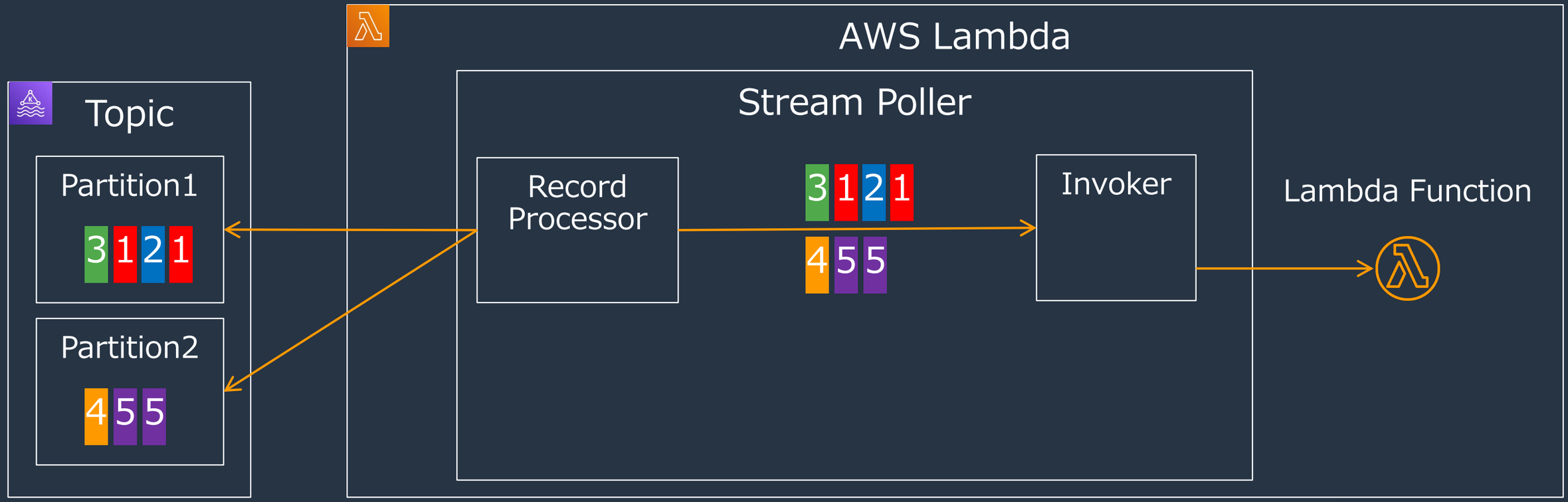
- Parallelization Factor 設定により, Lambda Function の並列数をデフォルトの 1 倍から最大 10 倍まで調整可能
- パーティションキーのハッシュ値に応じて内部振り分けが行われるため, キーのカーディナリティが重要



Lambda Consumer のアーキテクチャ

Amazon Managed Streaming for Apache Kafka の場合

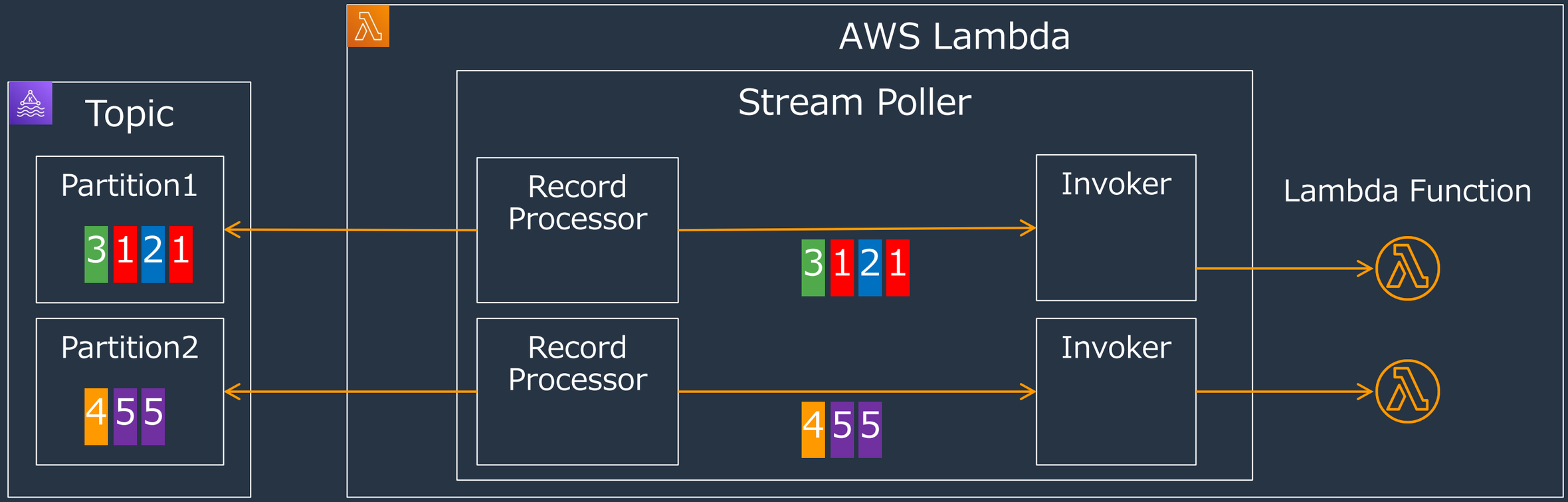
- 最初は全パーティションのデータを 1 つの Lambda Function で処理



Lambda Consumer のアーキテクチャ

Amazon Managed Streaming for Amache Kafka の場合

- 最初は全パーティションのデータを 1 つの Lambda Function で処理
- 15分毎に OffsetLag メトリクスから Consumer の処理遅延を確認し, 遅延が判断された場合は Lambda Function 呼び出しの並列数が増加する. 最大並列数はパーティション数と同数
- Parallelization Factor に該当する機能は無し



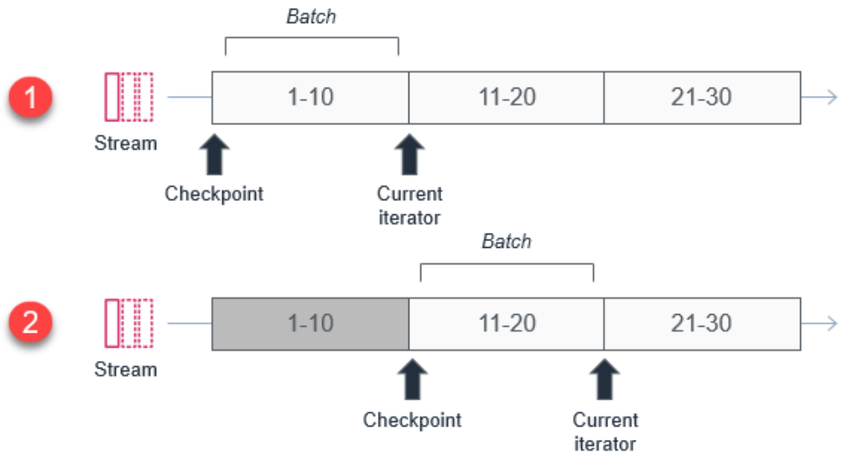
aws DevAx connect

Lambda Consumer のリトライ処理

- レコードバッチの処理でエラーが発生した場合, 回数・期限ベースのリトライ制限,データストリーム側の保持期限に達するまでリトライされる. リトライ方式は以下 3 つ
- Amazon Kinesis Data Streams, Amazon DynamoDB Streams との組み合わせで利用可能

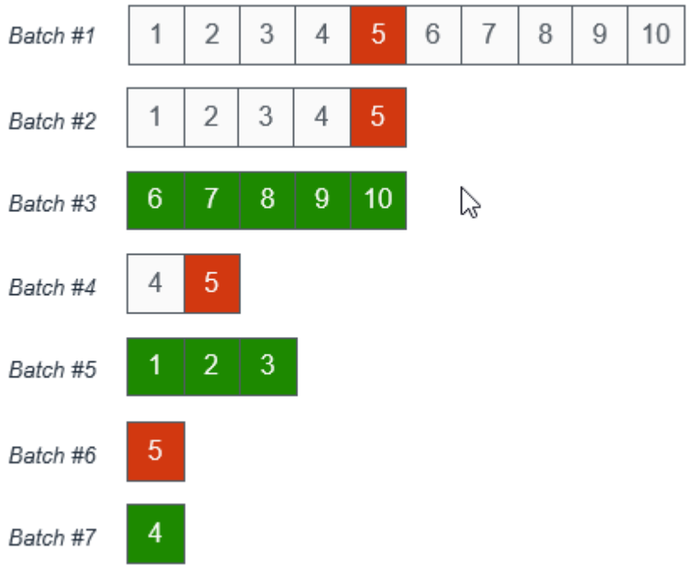
Default settings

エラーが発生したバッチを再処理



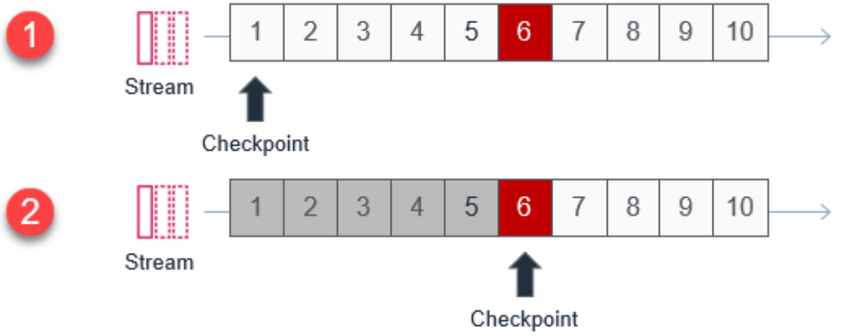
Bisect batch

エラーが発生したバッチを分割して再処理



Custom Checkpoint

開始地点を指定して再処理



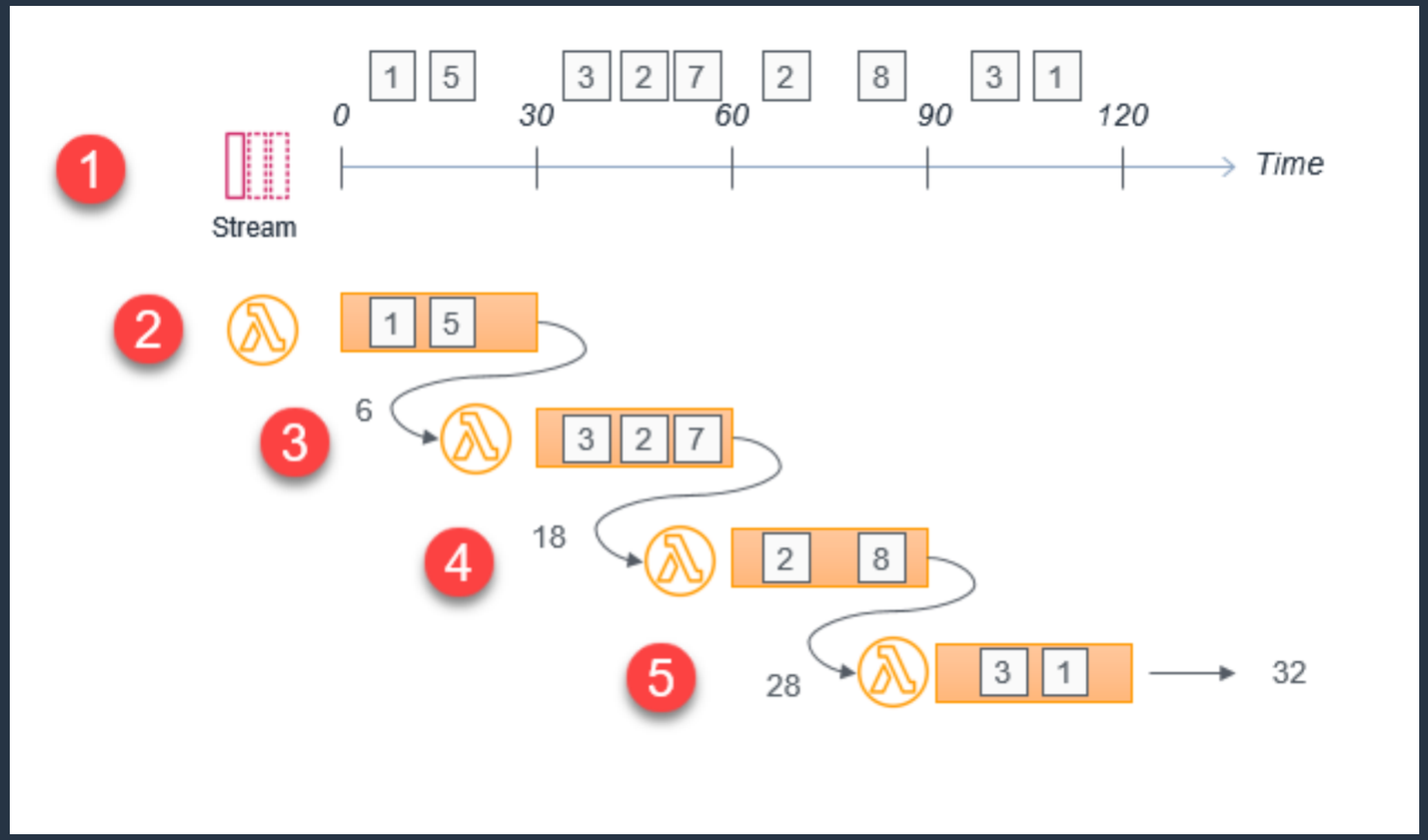
aws DevAx connect

Lambda Consumer による Window 処理

- 指定した秒数ごとにレコードバッチを区切って Lambda Function を呼び出す

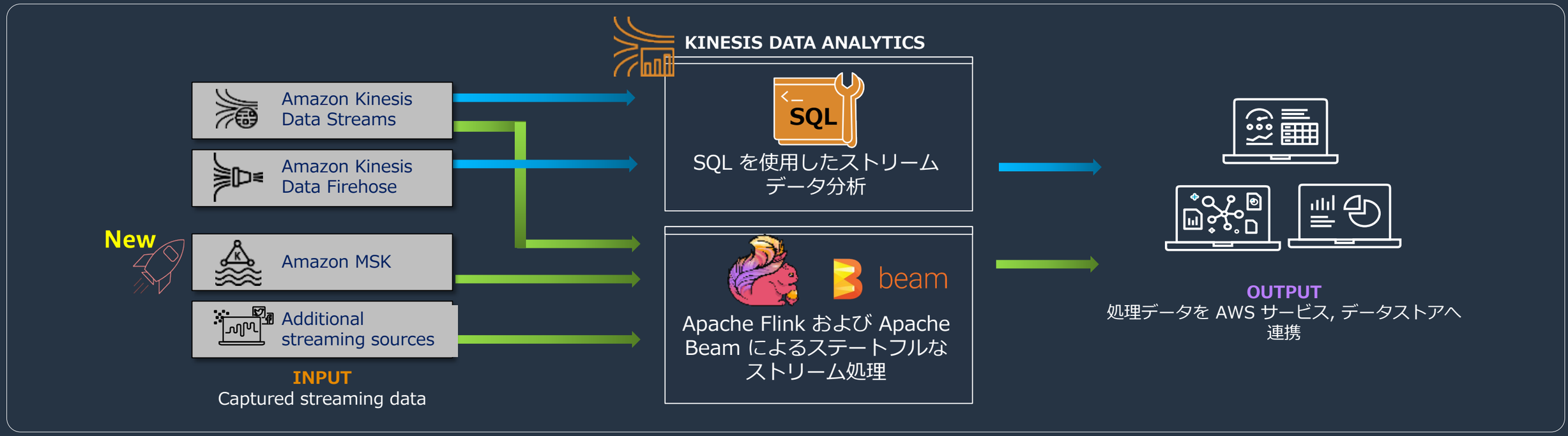
- Lambda がシャードごとに起動する性質上, シャードをまたがった複雑な集計はできない

- Kinesis Data Streams, DynamoDB Streams で使用可能



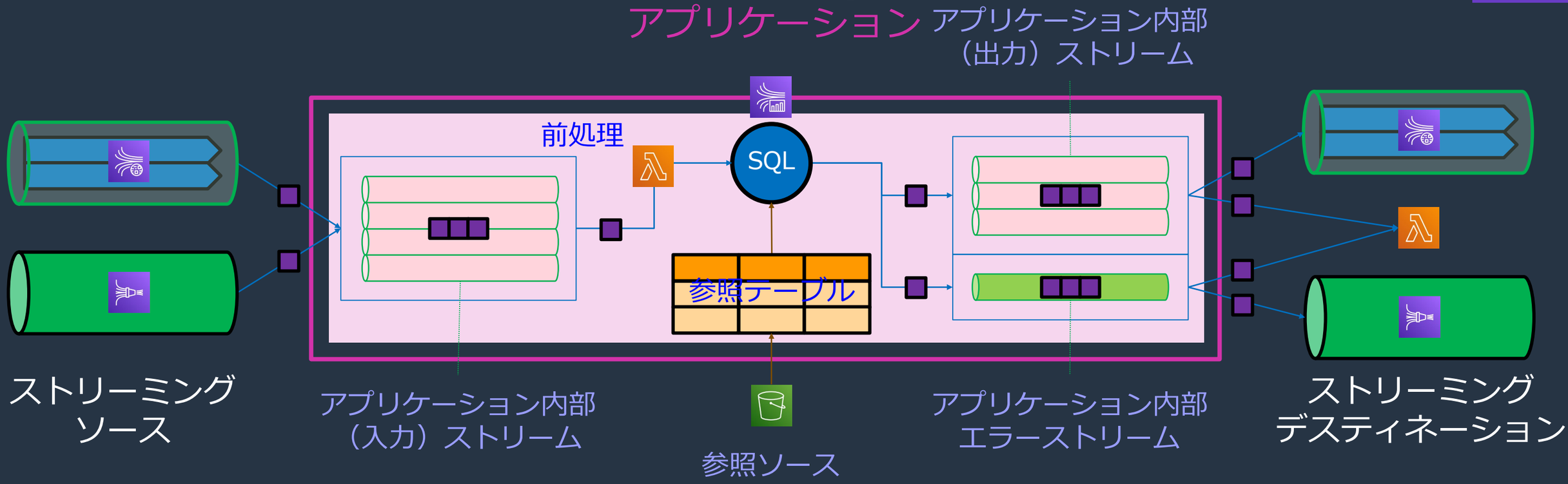
aws DevAx connect

Amazon Kinesis Data Analytics



- SQL もしくは Apache Flink(Java, Scala, Python), Apache Beam アプリケーションを使用したリアルタイムなストリームデータ処理を実現するサービス
- アプリケーション実行に必要な基盤は全てサービス側で管理. ユーザーはアプリケーションパッケージを作成してデプロイするだけ

Amazon Kinesis Data Analytics for SQL

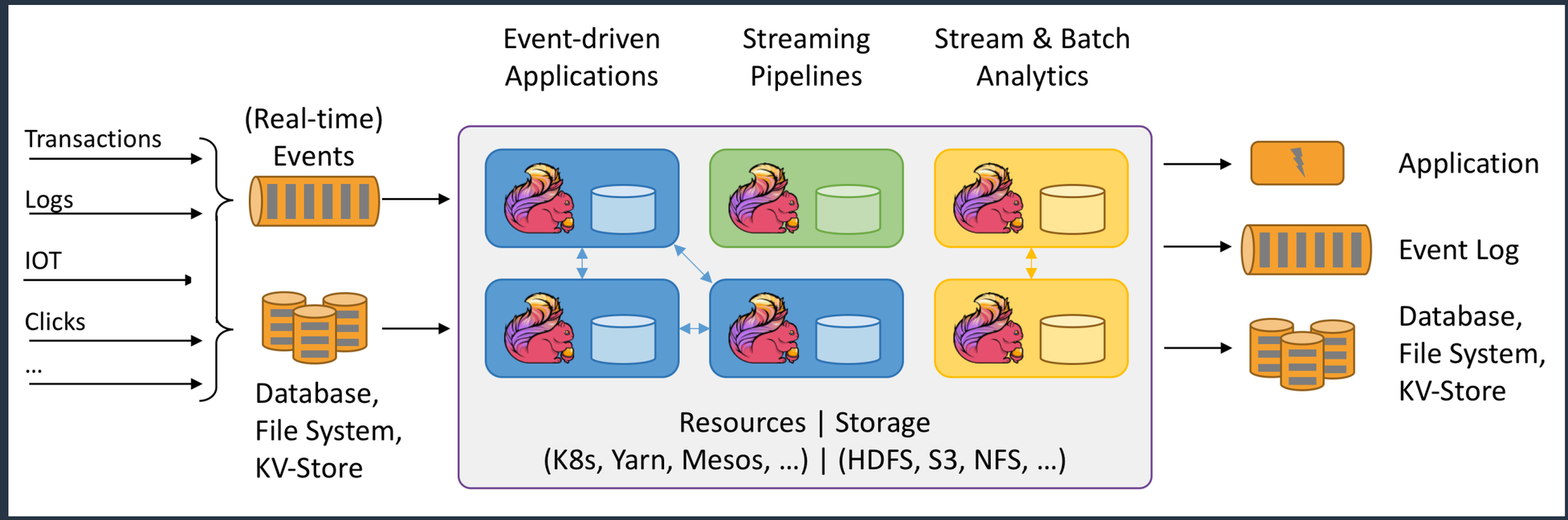


- SQL を記述するだけでデータ分析を実現するサービス
- クエリの複雑さとスループットに応じて処理能力 (KPU – Kinesis Processing Units, 1KPU = 1vCPU+4GB Memory) を自動伸縮
- ソースとして Amazon Kinesis Data Streams, **Amazon Kinesis Data Firehose** を, 宛先は同サービスに加えて Lambda をサポート
- S3 上のデータを外部ソースとして参照・結合する機能や, Lambda によるデータ前処理機能を提供

Amazon Kinesis Data Analytics for Flink

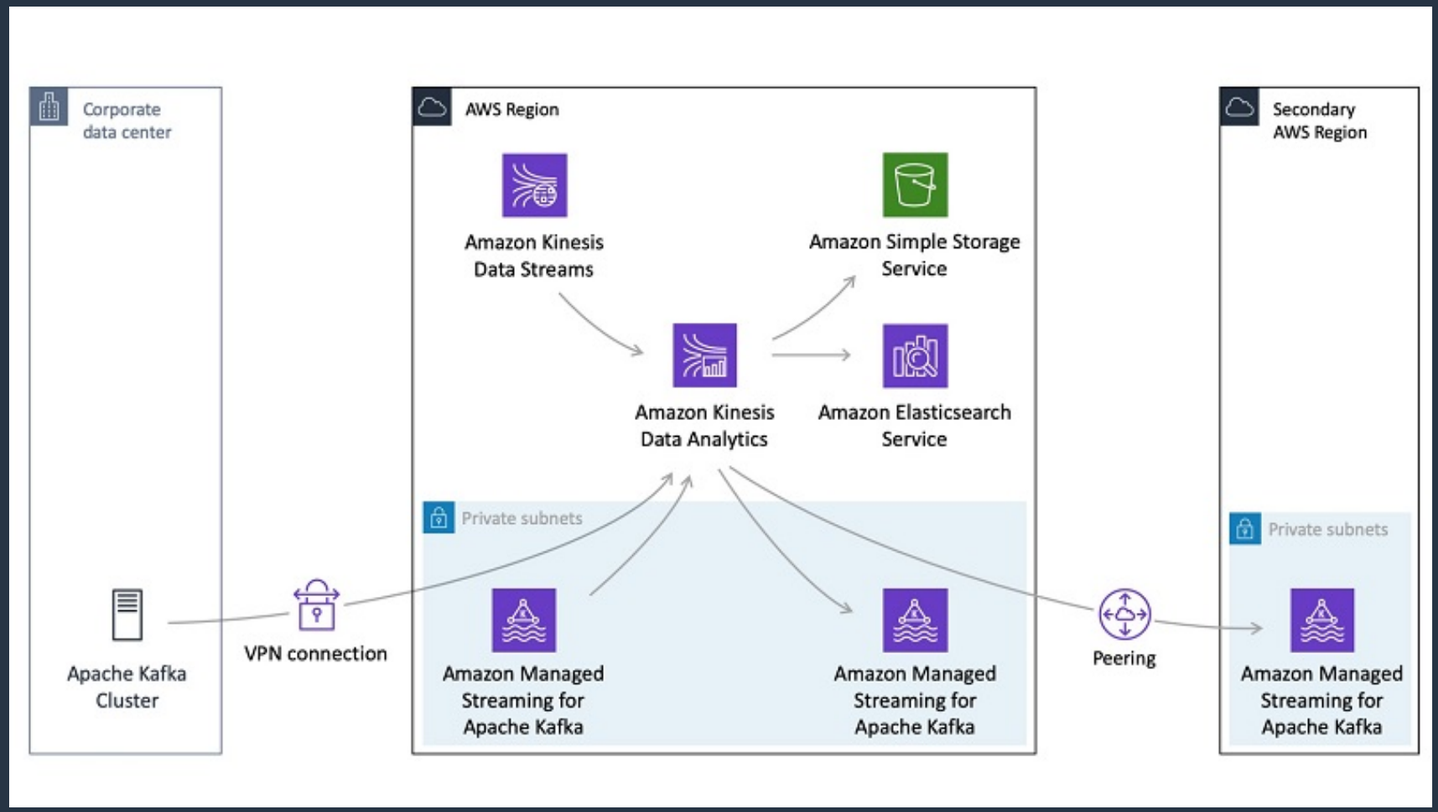
リアルタイム処理アプリケーションを実行するためのフルマネージドサービス

- Apache Flink(Java, Scala, Python), Apache Beam をサポート
- パッケージをアップロードするだけでアプリケーションを実行できる
- Checkpoint, Savepoint は S3 に保存されており, 中断箇所からアプリケーションを再開することが可能
- CPU リソース使用率に基づく Auto Scaling 機能を提供



Amazon Kinesis Data Analytics for Flink の分析対象

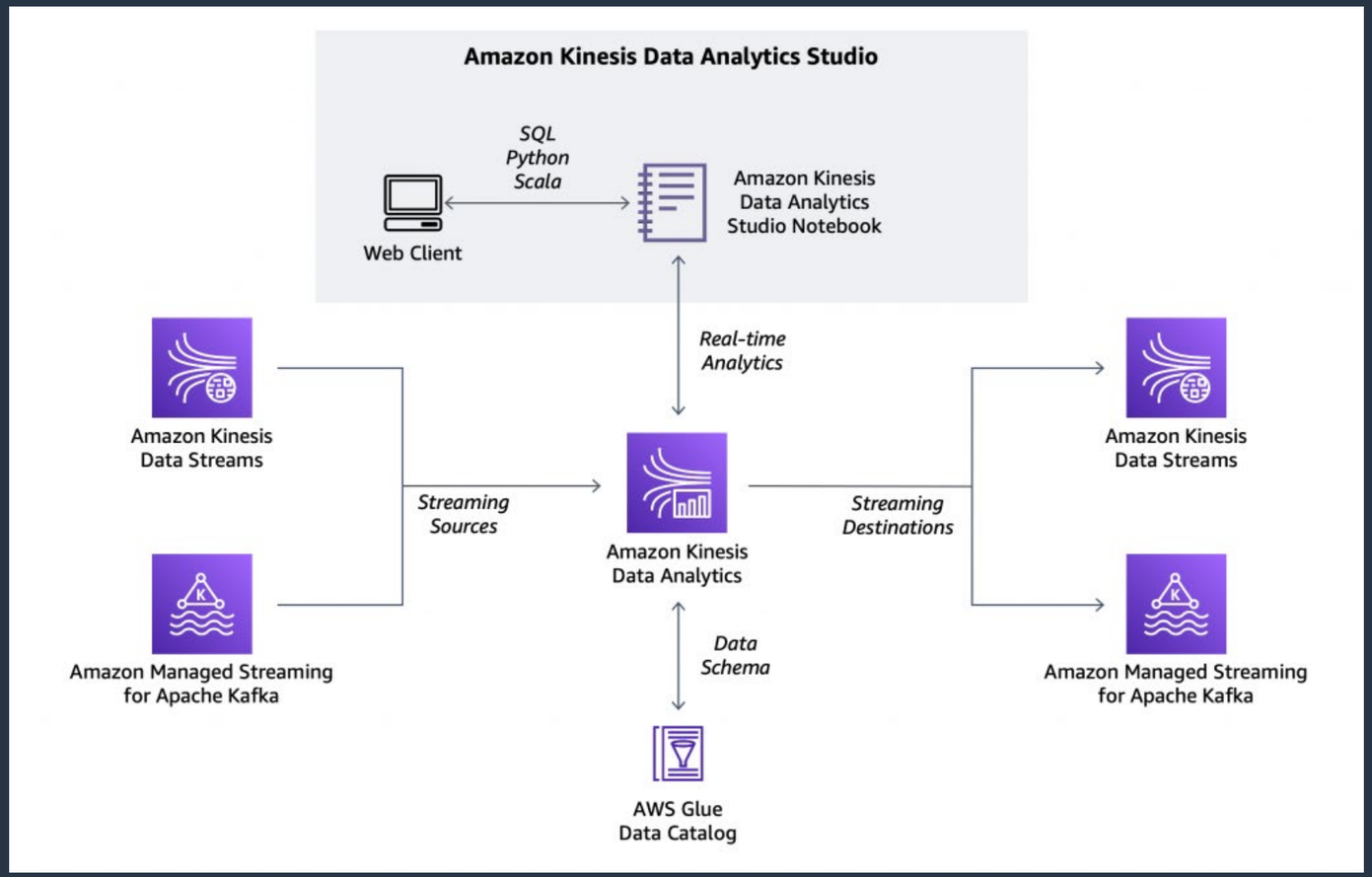
- Amazon Kinesis Data Analytics for SQL では不可能だった, VPC 内のリソースへのアクセスをサポート
- VPN, Direct Connect, VPC Peering, Transit Gateway 経由でオンプレミス環境や別アカウント・別リージョンのリソースへのアクセスも可能

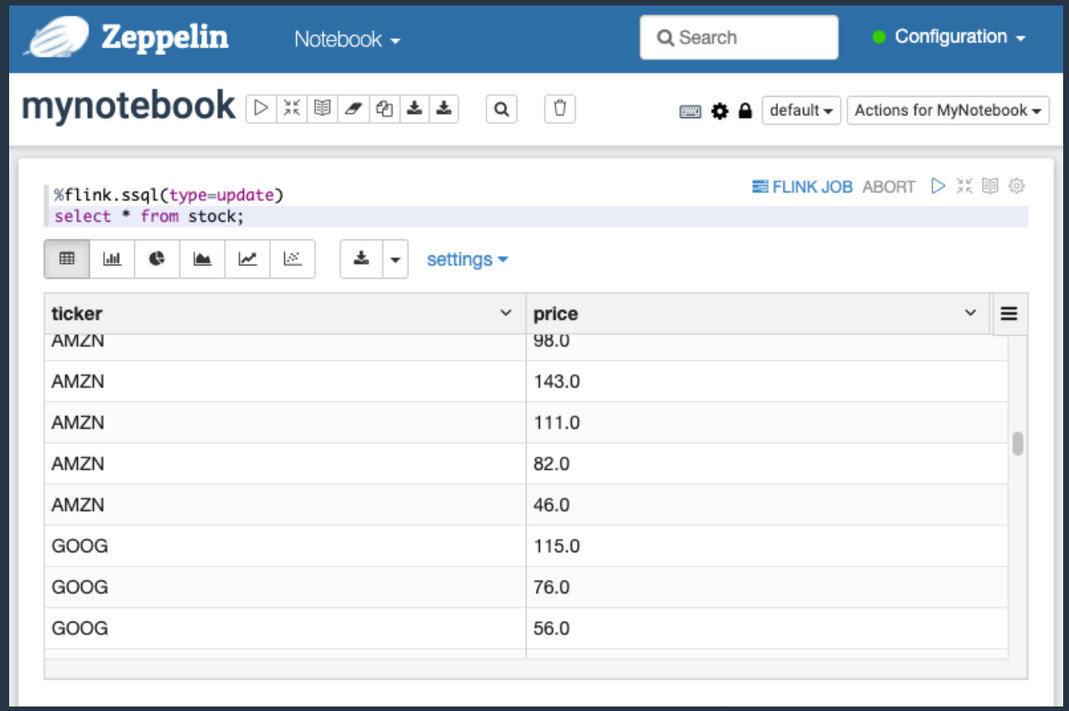


Kinesis Data Analytics Studio

New!

- ノートブックでストリームデータをインタラクティブに分析
- 目指した結果が得られたら数クリックで本番デプロイ





aws DevAx connect

AWS Glue Streaming ETL Jobs

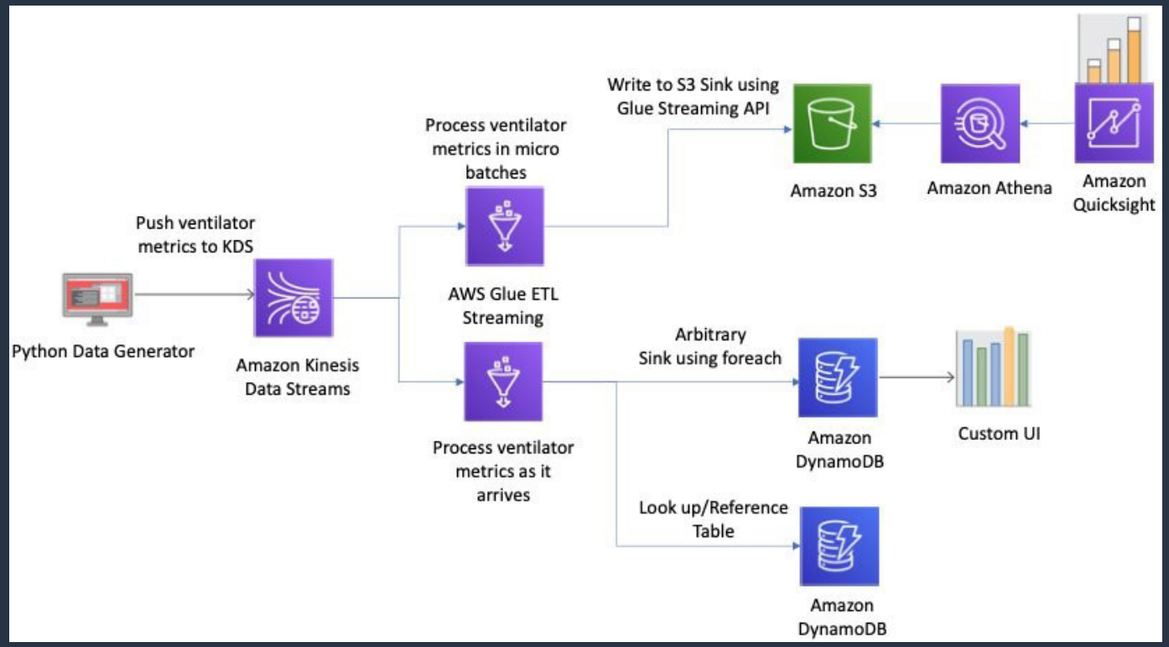
AWS Glue 上で Spark Structured Streaming アプリケーションによるリアルタイム ETL 処理を実行

- AWS Glue の持つ機能を活用したリアルタイム ETL 処理を実装可能
  - AWS Glue Data Catalog により管理されたメタデータ情報(データの場所, データ定義)の利用
  - AWS Glue ETL Library による拡張機能の利用

- 中断箇所からの再開をサポート
  (チェックポイント情報を S3 に保存)

- ソースとして Amazon Kinesis Data Streams, Amazon MSK, Apache Kafka をサポート

- 宛先は S3, JDBC, Redshift の他, DynamoDB など任意のカスタムリソースをサポート

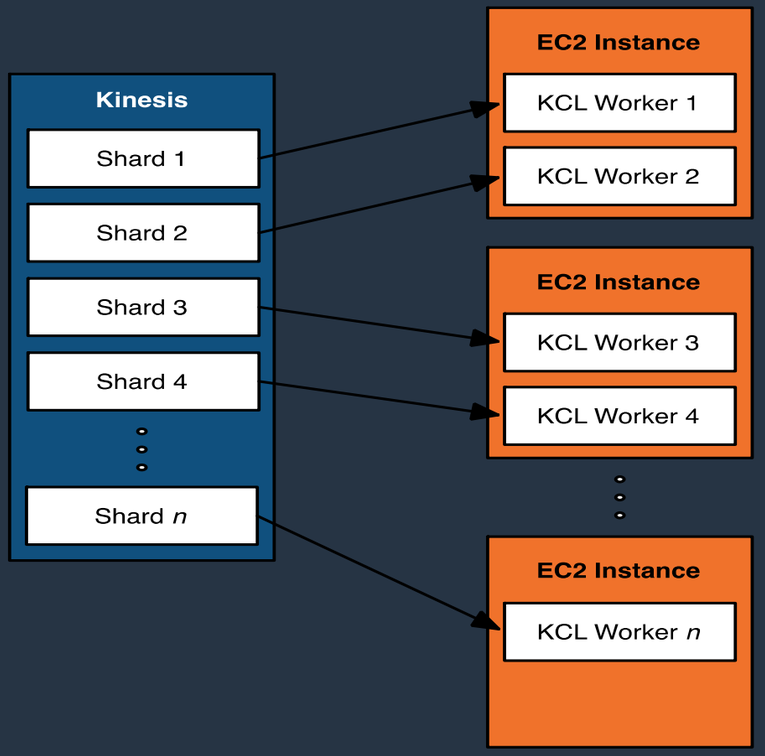


https://spark.apache.org/docs/2.0.0/structured-streaming-programming-guide.html
https://aws.amazon.com/jp/blogs/big-data/crafting-serverless-streaming-etl-jobs-with-aws-glue/

aws DevAx connect

Amazon Kinesis Client Library (KCL)

- Kinesis Data Streams 専用の Consumer ライブラリ
- Java で開発されており, Ruby, Python, Node.js, .NET から利用可能な MultiLangDaemon も提供している
- Fault Tolerance のため, State を外部の DynamoDB に保存
- KCL アプリは 3 つのコンポーネントを含んでいる
    1. Record Processor Factory - 2. のレコードプロセッサーを作る
    2. Record Processor - Amazon Kinesis Data Streamsのシャードから取り出したデータを処理するプロセッサーの単位
    3. Worker - 個々のアプリケーションインスタンスとマッピングする処理単位



https://github.com/search?q=org%3Aawslabs+kinesis-client

aws DevAx connect

Consumer の選定基準

既存アプリケーションに組み込む場合

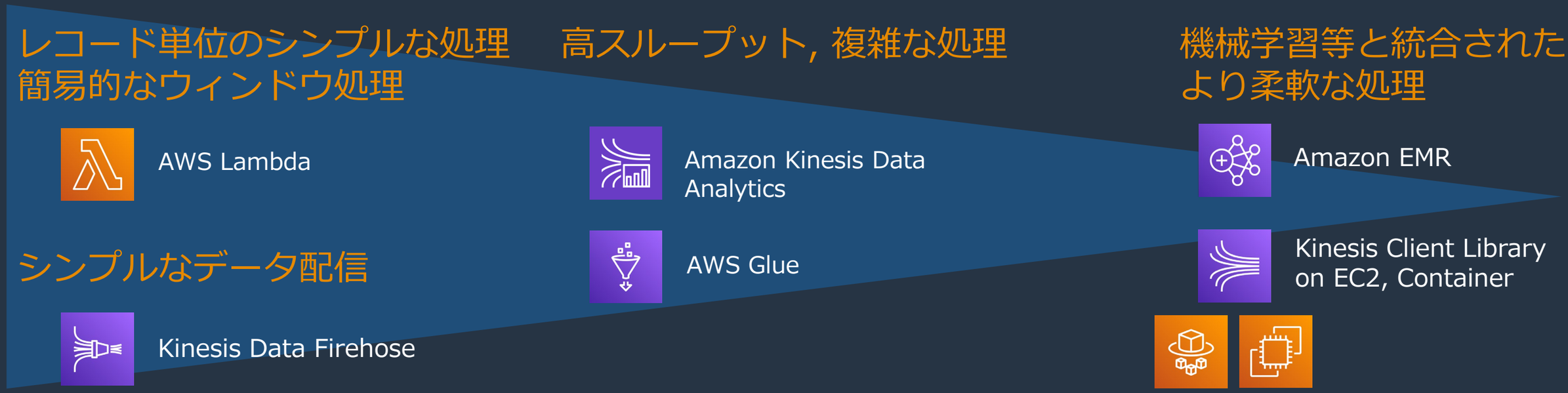
- Kinesis Client Library の利用を検討

新規に Consumer アプリケーションを作成する場合

- 用途(データ配信とデータ処理のどちらがメインか)
- 連携サービス(Amazon Kinesis のみか, それ以外との連携もあるか)
- 処理の複雑性(単純なデータ配信か, それとも複雑な集計が求められるか)
- 想定スループット(求められる処理性能はどの程度か)
- レイテンシ要件(どの程度の遅延までなら許容可能か)
- 開発生産性(使い慣れた言語, フレームワークを使用したいか, ノーコード, ローコードで実装したいか)
- メンテナンス性(運用の省力化を重視するか)

Consumer 選定のポイント

- マネージドサービスの中から, スループット要件, 処理要件, 利用可能な言語に応じてサービスを絞る
- 処理の複雑さに合わせて, 管理負荷が低いものを選択する



aws DevAx connect

実装上のポイント

- ユースケース, レイテンシ要件, データ量を確認する

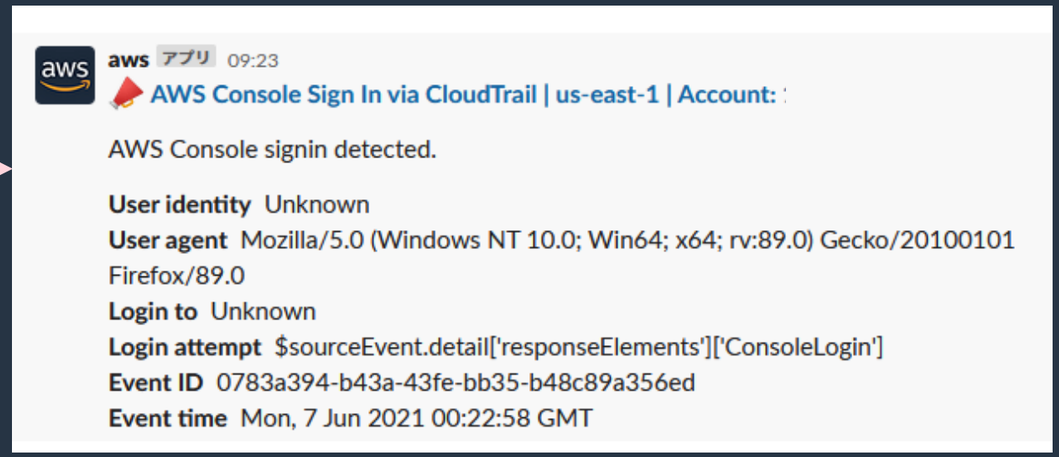
- マネージドサービスを組み合わせたシステムアーキテクチャーを検討する

- (必要が無ければ)実装しない

イベント駆動で処理できないか考えてみる

要件によってはアプリケーションを実装せず, Amazon EventBridge など AWS サービスを組み合わせてイベント駆動で実現できる

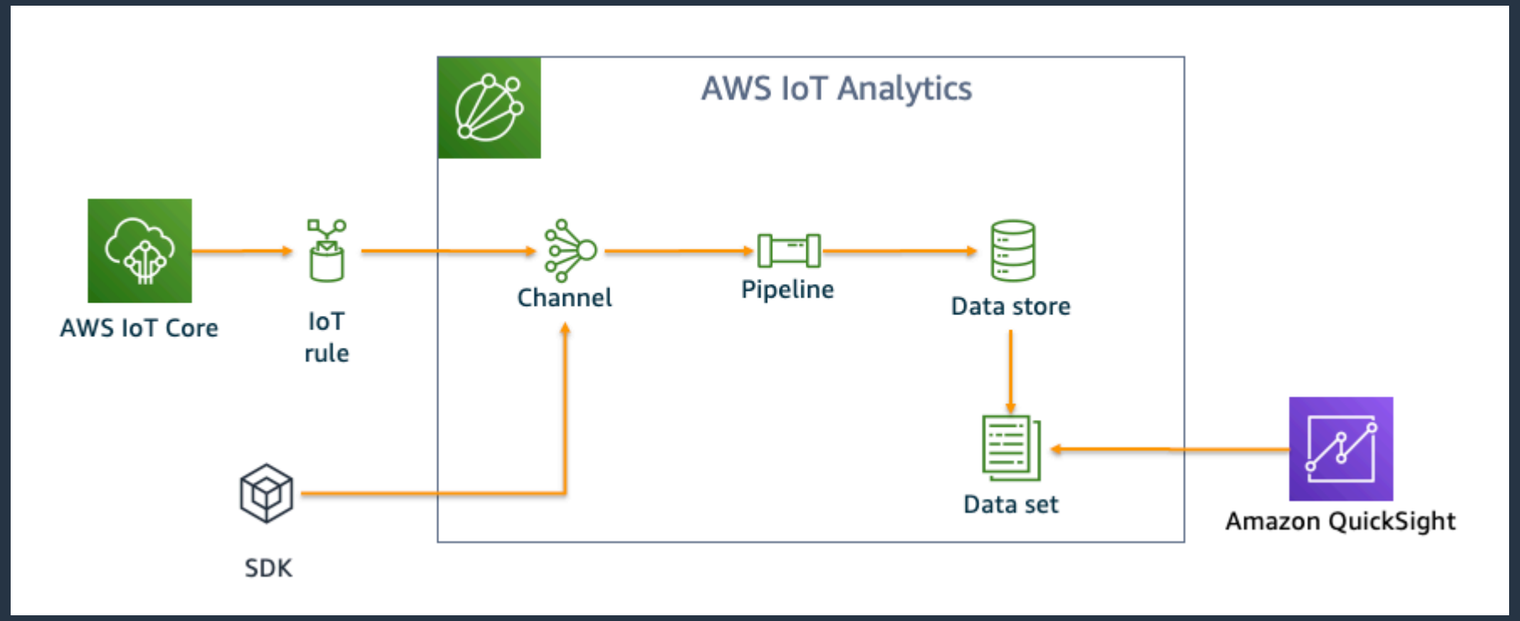
<u>要件例</u>:
AWS コンソールへのログインを検知し, 数分以内に Slack に通知を飛ばしたい



特定領域に特化したサービスの利用を検討する

特定領域に特化したサービスを活用することで, 実装領域や運用負荷を削減

<u>要件例</u>:
IoT デバイスから送信されるデータをリアルタイムに処理し, BI ツールで可視化したい



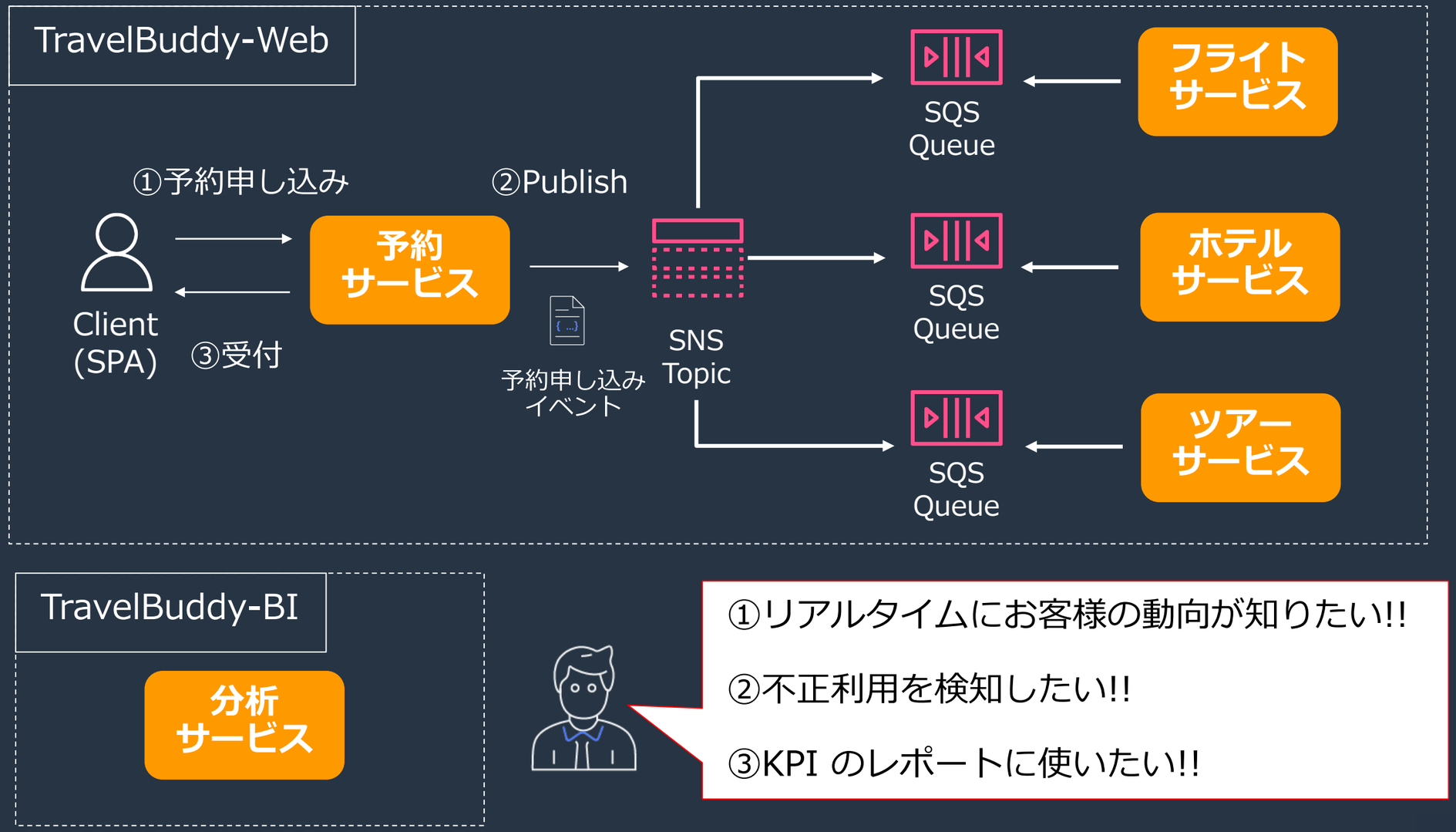
aws DevAx connect

イベント駆動アプリケーションと
ストリーム処理の美味しい関係

aws DevAx connect

おさらい: TravelBuddy の分析要件

第四回資料より



aws DevAx connect

対応方針(再掲)

要件

①リアルタイムにお客様の動向が知りたい!!

ツアー毎の予約状況, キャンセル状況を**リアルタイム**に見たい

リアルタイムダッシュボードを構築する

②不正利用を検知したい!!

特定のユーザーが予約とキャンセルを繰り返しているなど,不審な行動を**速やかに**検出, 通知したい

ウィンドウ処理を使ったリアルタイム分析を行う

③KPI のレポートに使いたい!!

日次でレポートを作成し, 関係者にメールで配信したい

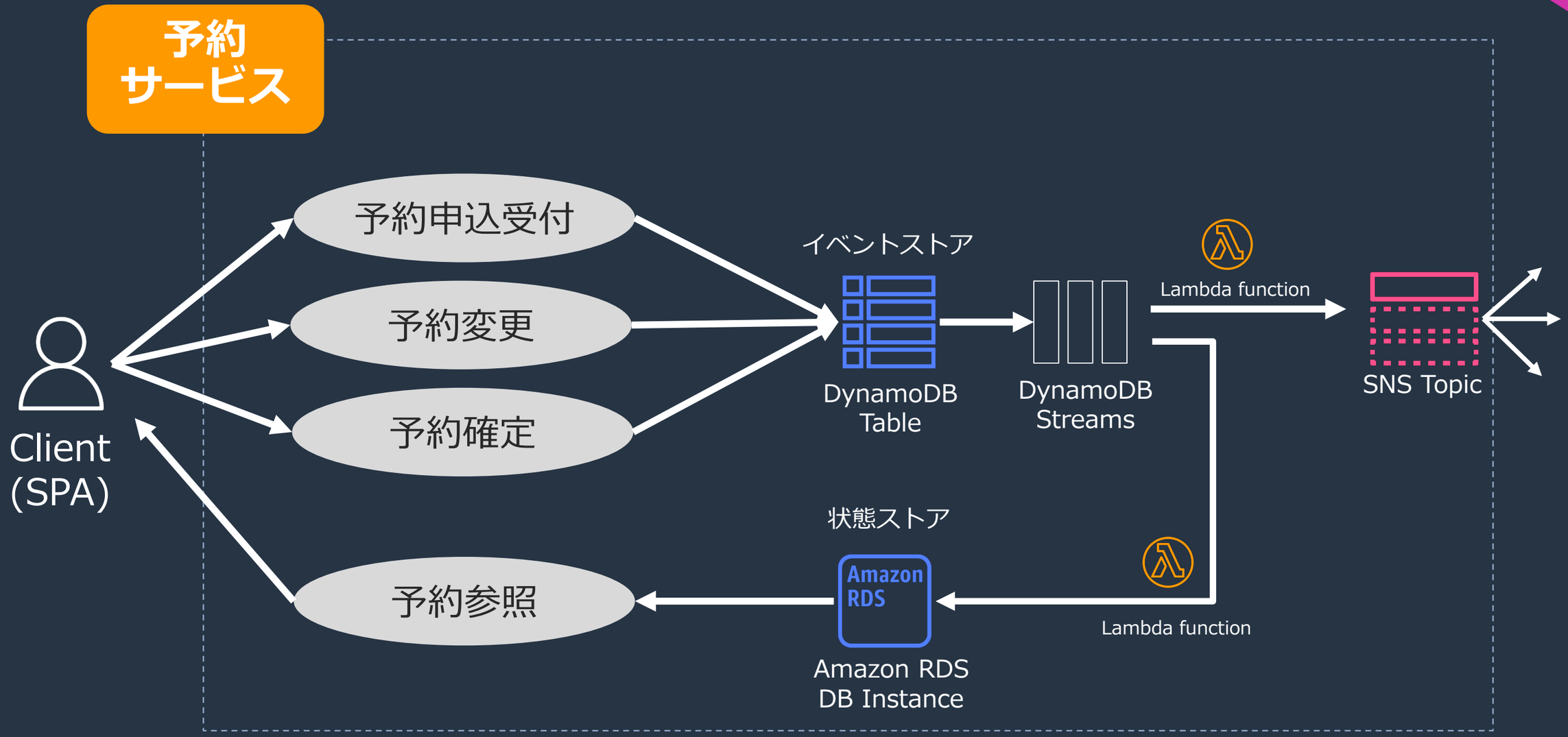
データ流量

- メッセージサイズ:
- メッセージの流量: 20000 messages/sec

レポートの作成自体はバッチ処理だが, 今後メッセージ流量が増えた際にバッチ処理時間が延びることを防ぐため, データ変換などの前処理はストリーム処理にオフロードする

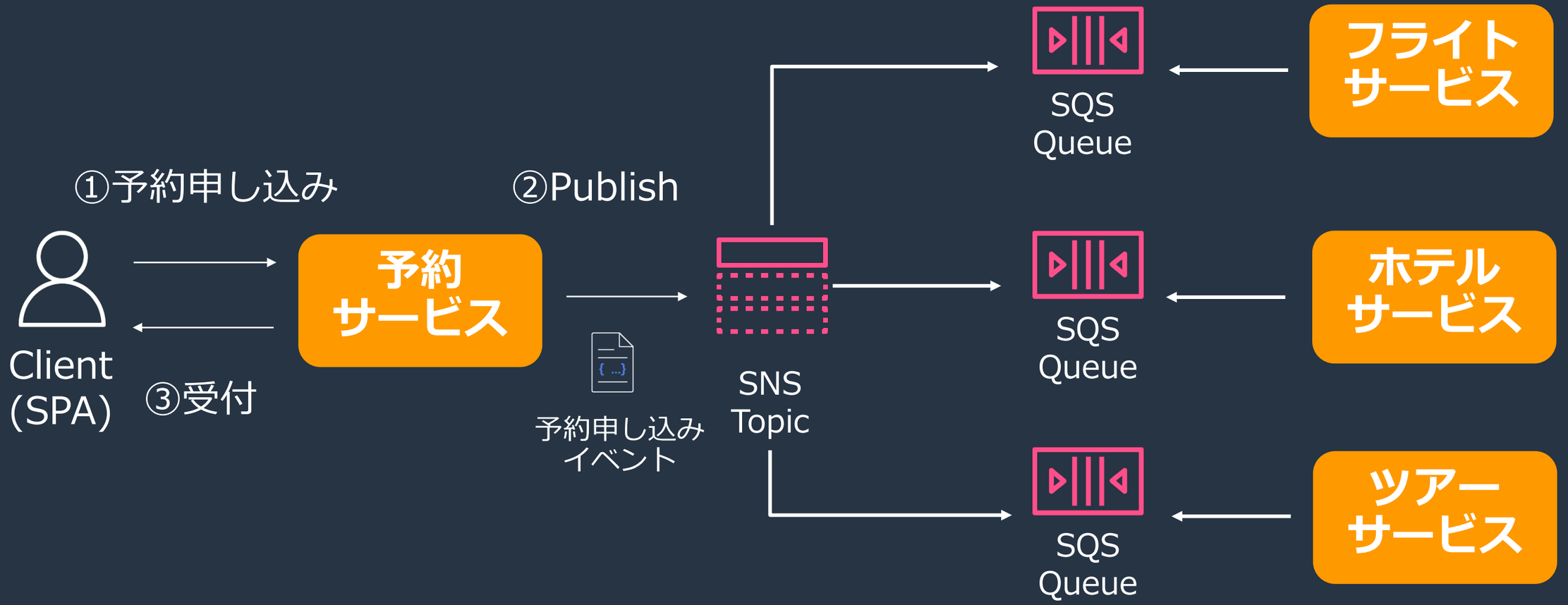
予約サービスのデータフロー

第四回資料より



aws DevAx connect

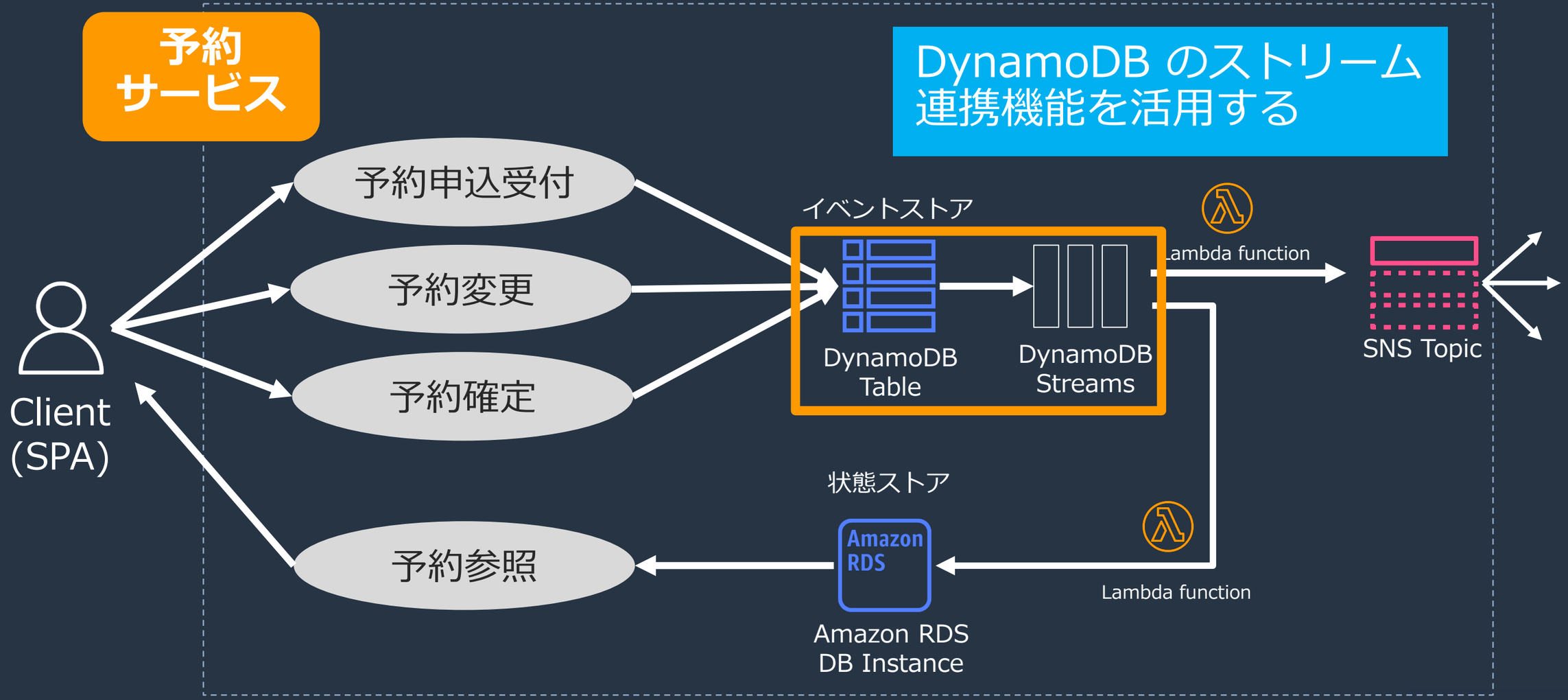
既存サービスに変更を入れずに分析を実現するには？



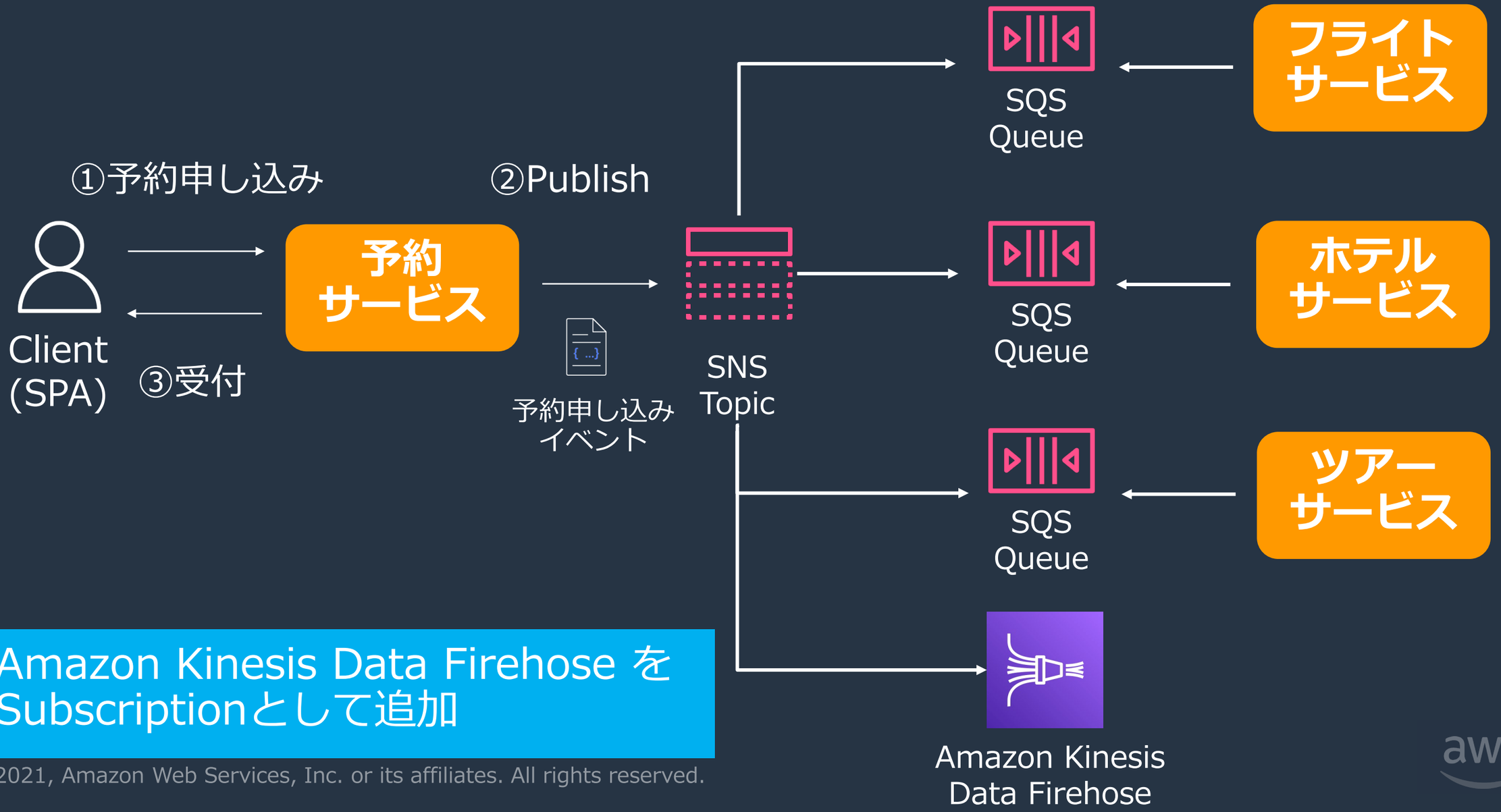
aws DevAx connect

方法1: 既に使用されているデータストアを活用

- イベント駆動アプリケーションに限らず利用可能な方式

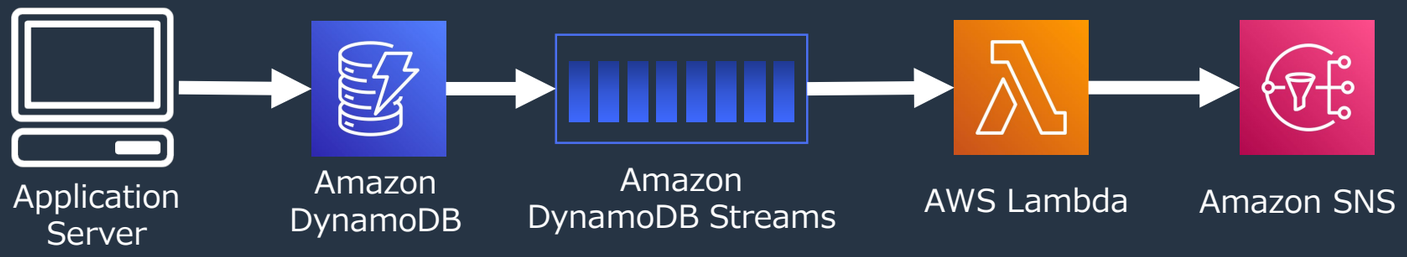


方法2: データ分析も 1 つの連携システムと考える



Amazon Kinesis Data Firehose を Subscriptionとして追加

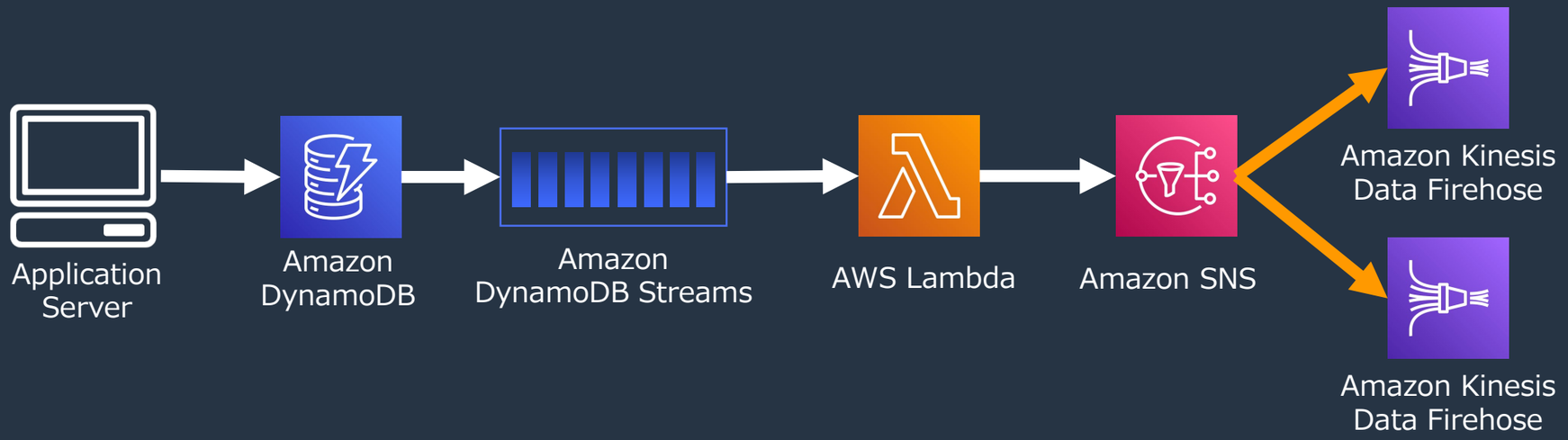
旅行サイトの予約データ分析アーキテクチャー



aws DevAx connect

旅行サイトの予約データ分析アーキテクチャー

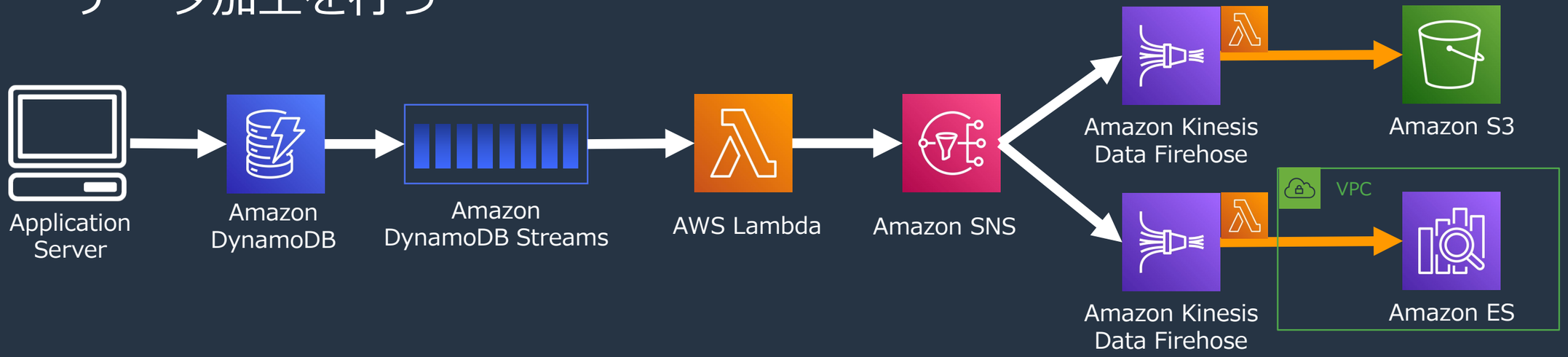
Amazon SNS に対して Amazon Kinesis Data Firehose を Subscription として追加



aws DevAx connect

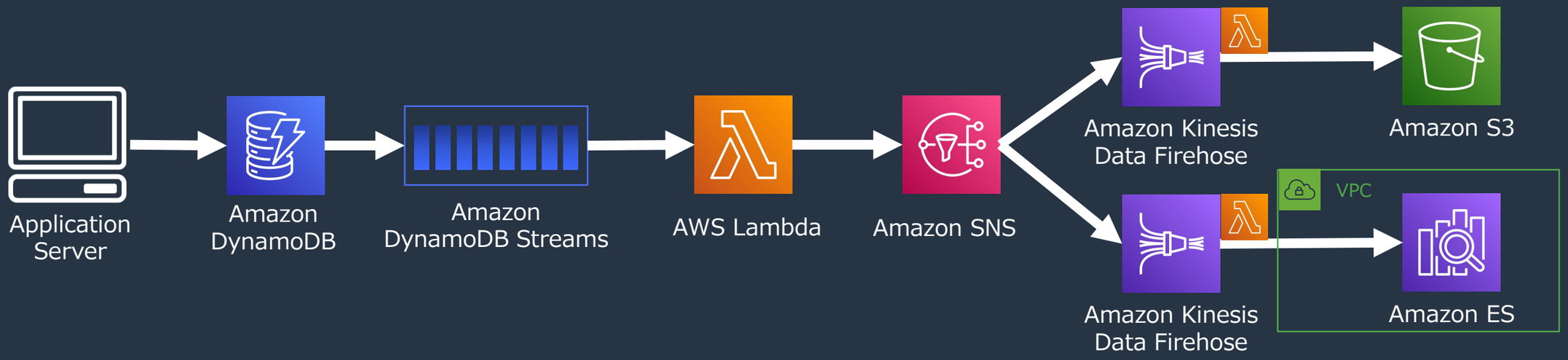
旅行サイトの予約データ分析アーキテクチャー

- 各 Delivery Streams の宛先に S3 と VPC 内の Amazon ES を指定
- Kinesis Data Firehose のデータ変換機能(Lambda)を活用し, 配信前にデータ加工を行う



旅行サイトの予約データ分析アーキテクチャー

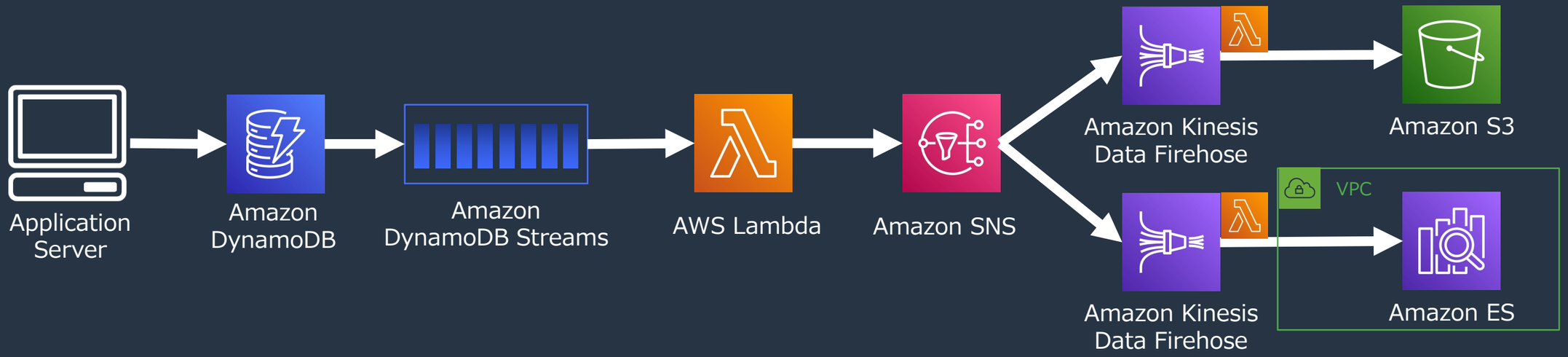
KPI レポートは S3 上のデータを Athena で分析することで, リアルタイムダッシュボードは Amazon ES で実現できそうだ



aws DevAx connect

旅行サイトの予約データ分析アーキテクチャー

KPI レポートは S3 上のデータを使用して, リアルタイムダッシュボードは Amazon ES で実現できそうだ



ところで, 不正検出はどう実現する？

aws DevAx connect

TravelBuddy 分析要件, データ量詳細(再掲)

要件

①リアルタイムにお客様の動向が知りたい!!
ツアー毎の予約状況, キャンセル状況を**リアルタイム**に見たい

②不正利用を検知したい!!
特定のユーザーが予約とキャンセルを繰り返しているなど,不審な行動を**速やかに**検出, 通知したい

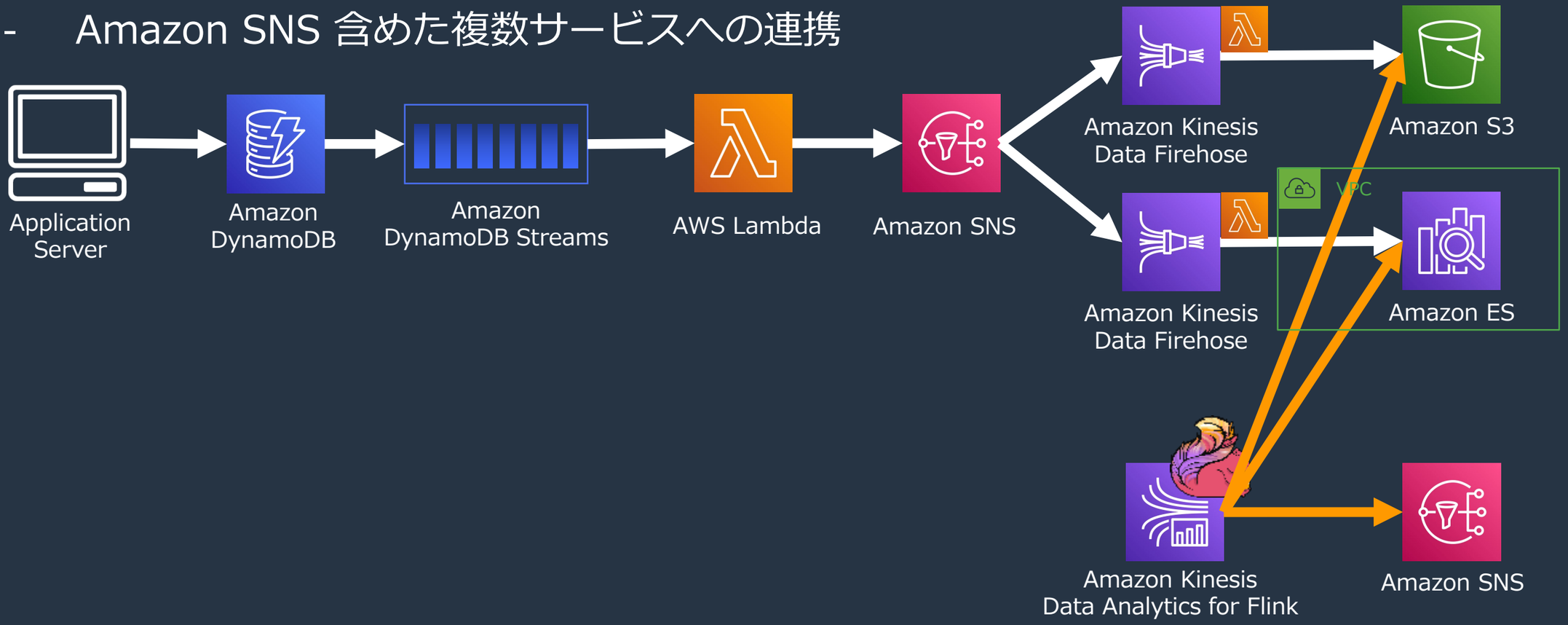
③KPI のレポートに使いたい!!

- **ユーザー ID ごとにログイン中のアクティビティをグループ化し,イベント内容を分析していく必要が有る. ログイン中に行われている不正動作を検知して即時対応したいので, Tumbling Window か Sliding Window がフィットしそうだ**
- **検知結果を SNS Topic に Publish しておくと, 関係者への通知や, 後続処理でアカウントロック処理を行うことも実現できそうだ. SNS と連携できる Consumer を使いたい**
- **検知結果を蓄積することでアプリケーションや検知ロジックの改善に繋げられるため, 結果の保管も合わせて行いたい**

旅行サイトの予約データ分析アーキテクチャー

下記の要件を満たす Consumer としては, Amazon Kinesis Data Analytics for Flink がフィットしそう

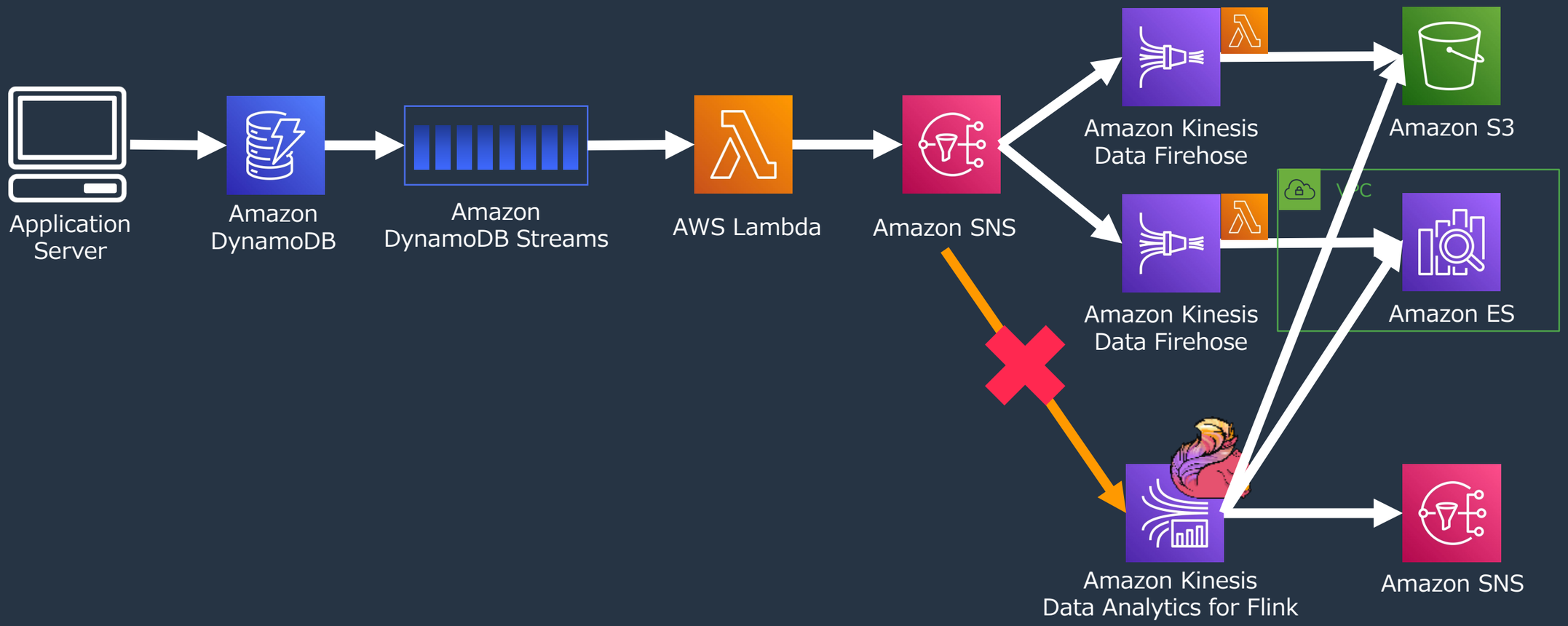
- Window 処理による分析
- Amazon SNS 含めた複数サービスへの連携



aws DevAx connect

旅行サイトの予約データ分析アーキテクチャー

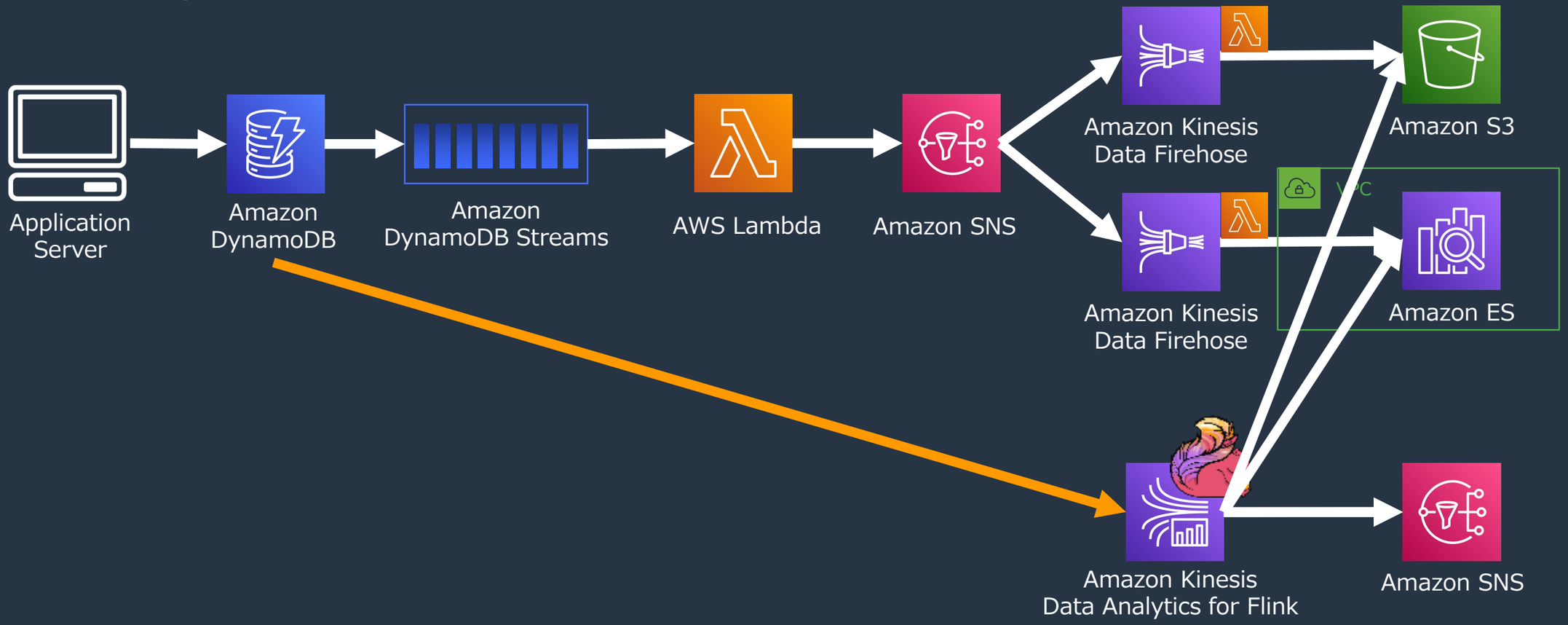
問題点: Amazon Kinesis Data Analytics for Flink は SNS を直接のソースとすることはできない



旅行サイトの予約データ分析アーキテクチャー

問題点: Amazon Kinesis Data Analytics for Flink は SNS を直接のソースとすることはできない

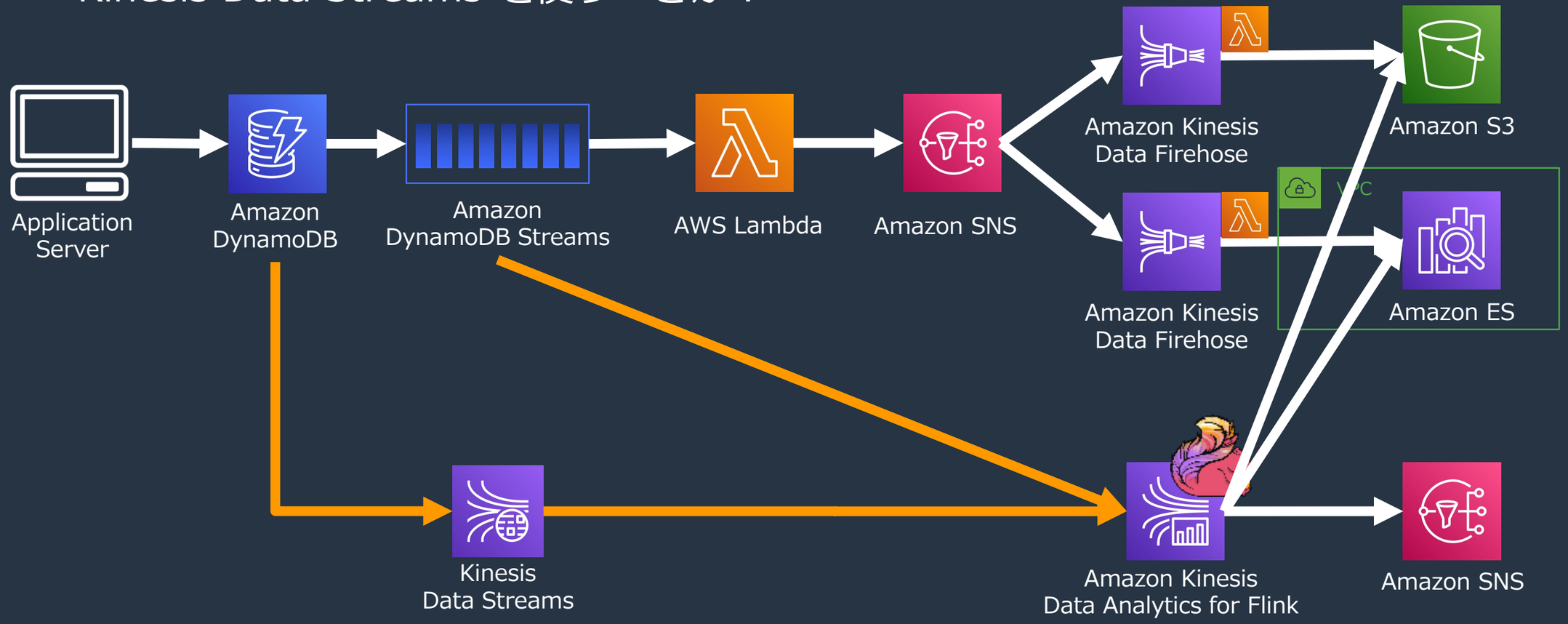
代替案: DynamoDB のストリーム機能でデータを取得する



旅行サイトの予約データ分析アーキテクチャー

課題

- DynamoDB Streams を使うべきか？
- Kinesis Data Streams を使うべきか？



aws DevAx connect

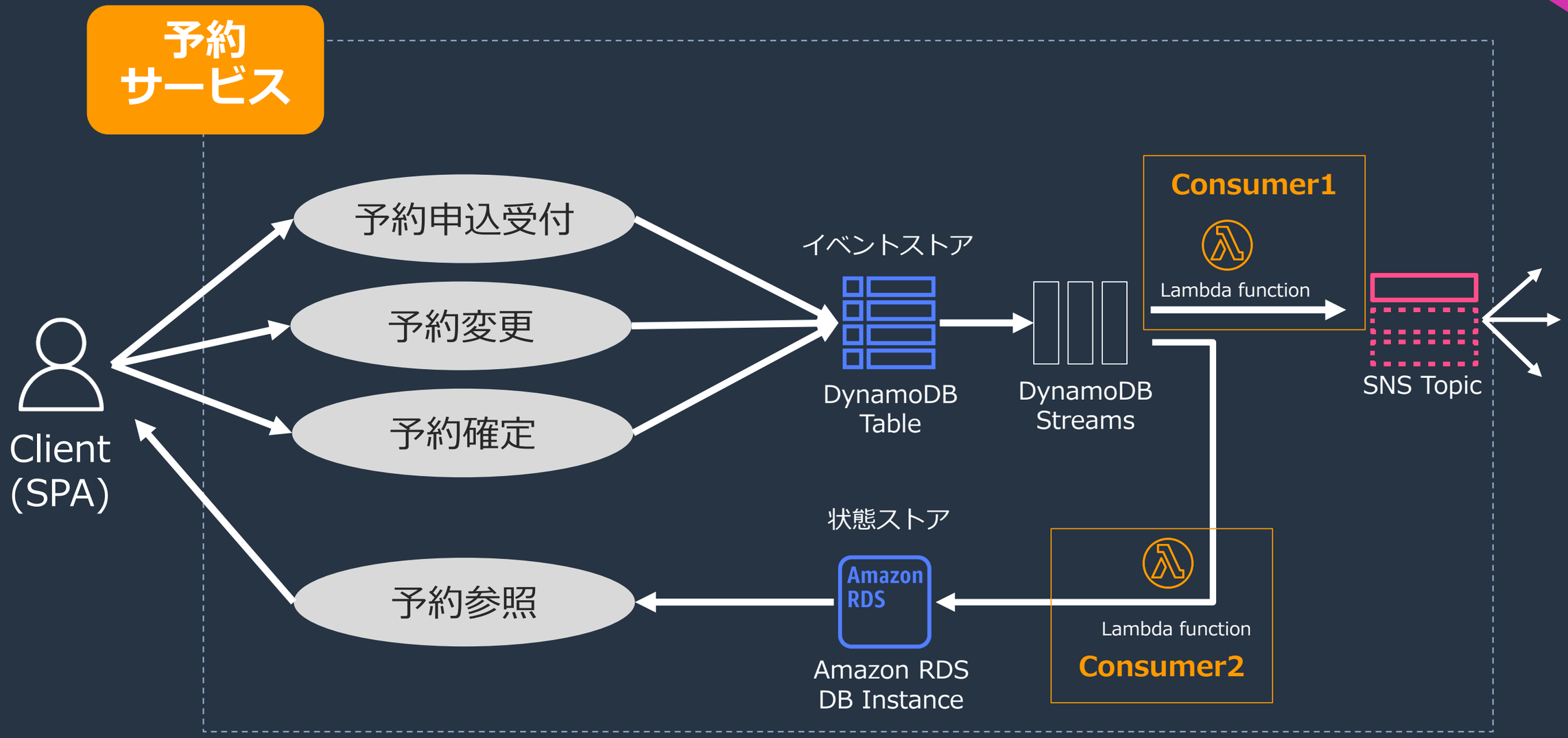
DynamoDB Streams と Kinesis Data Streams の使い分け(再掲)

- 24時間以上のデータ保持, AWS サービスとのより柔軟な連携, Consumer による複雑な処理が必要な場合は Kinesis Data Streams を検討
- Exactly Once やデータの並び順などが重要な場合は DynamoDB Streams を検討

プロパティ	DynamoDB + Kinesis Data Streams	DynamoDB Streams
データ保持期間	標準 24 時間. オプションにより最大 1 年 まで拡張可能	24 時間
コンシューマーの数	最大 5 (標準コンシューマー) 最大20(拡張ファンアウト)	**最大 2 つまで**
レコードの順序	Consumer の方で, レコードタイムスタンプから順序を識別する	ストリーム内のレコードは, 実際に発生した変更と同じ順序で出力される
レコード重複の有無	重複は起こりうる (At Least Once)	重複は発生しない (Exactly Once)
Consumer	- Kinesis Data Firehose - Kinesis Data Analytics for SQL - **Kinesis Data Analytics for Flink** - Kinesis Client Library - AWS Glue Streaming ETL Jobs - AWS Lambda	- DynamoDB Streams Kinesis Adapter - **Kinesis Data Analytics for Flink** - AWS Lambda

予約サービスのデータフロー(再掲)

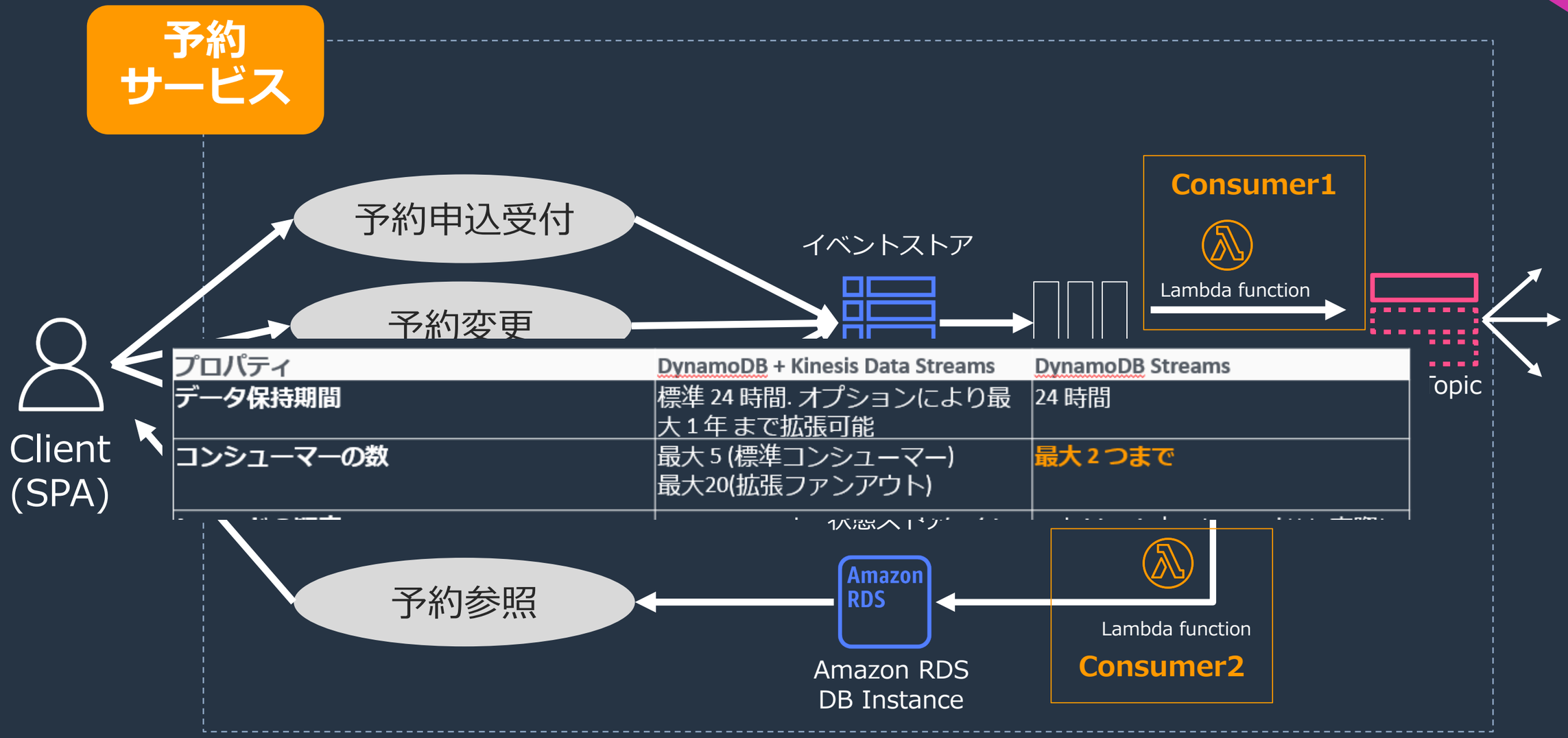
第四回資料より



aws DevAx connect

予約サービスのデータフロー(再掲)

第四回資料より



プロパティ	DynamoDB + Kinesis Data Streams	DynamoDB Streams
データ保持期間	標準 24 時間. オプションにより最大 1 年 まで拡張可能	24 時間
コンシューマーの数	最大 5 (標準コンシューマー) 最大20(拡張ファンアウト)	最大 2 つまで

aws DevAx connect

DynamoDB Streams と Kinesis Data Streams の使い分け(再掲)

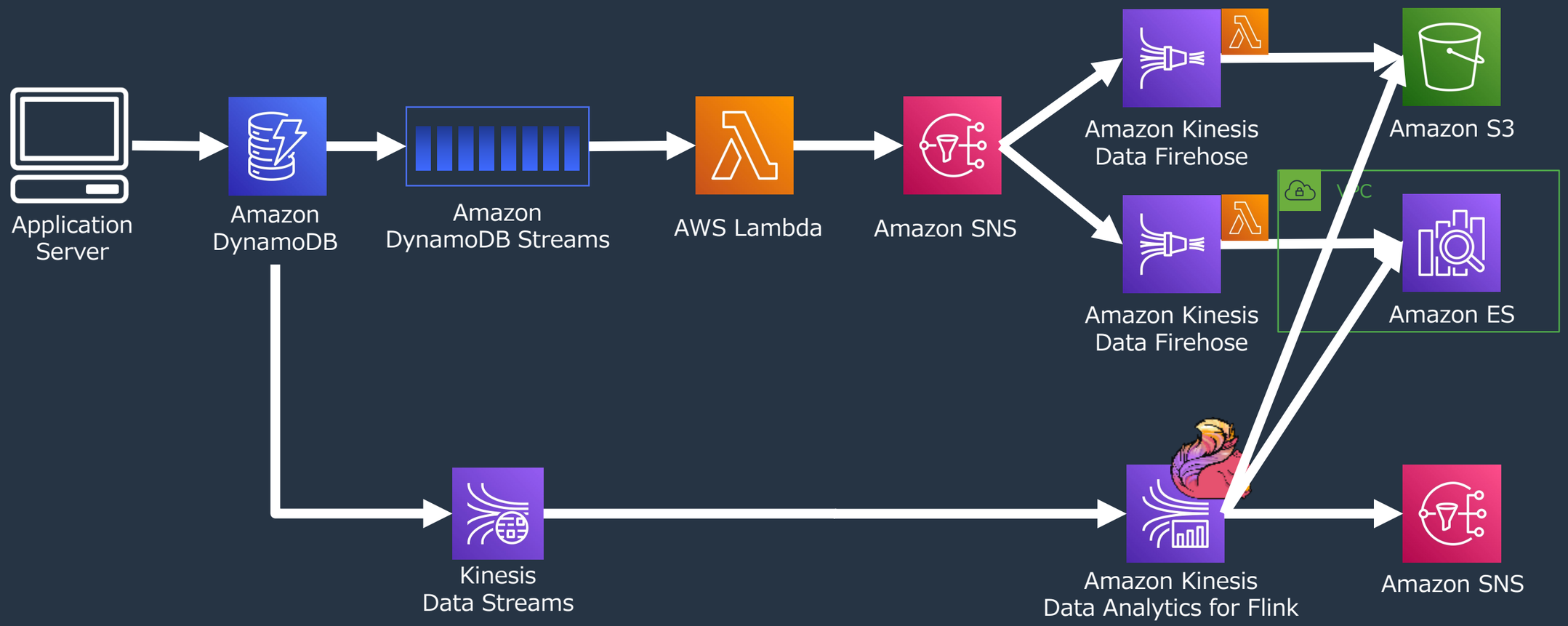
- 24時間以上のデータ保持, AWS サービスとのより柔軟な連携, Consumer による複雑な処理が必要な場合は Kinesis Data Streams を検討
- Exactly Once やデータの並び順などが重要な場合は DynamoDB Streams を検討

プロパティ	DynamoDB + Kinesis Data Streams	DynamoDB Streams
データ保持期間	標準 24 時間. オプションにより最大 1 年 まで拡張可能	24 時間
コンシューマーの数	**最大 5 (標準コンシューマー)** **最大20(拡張ファンアウト)**	**最大 2 つまで**
レ		
レ		
C		
	- AWS Lambda	

以下の理由から, 今回は DynamoDB + Kinesis Data Streams を使うこととする

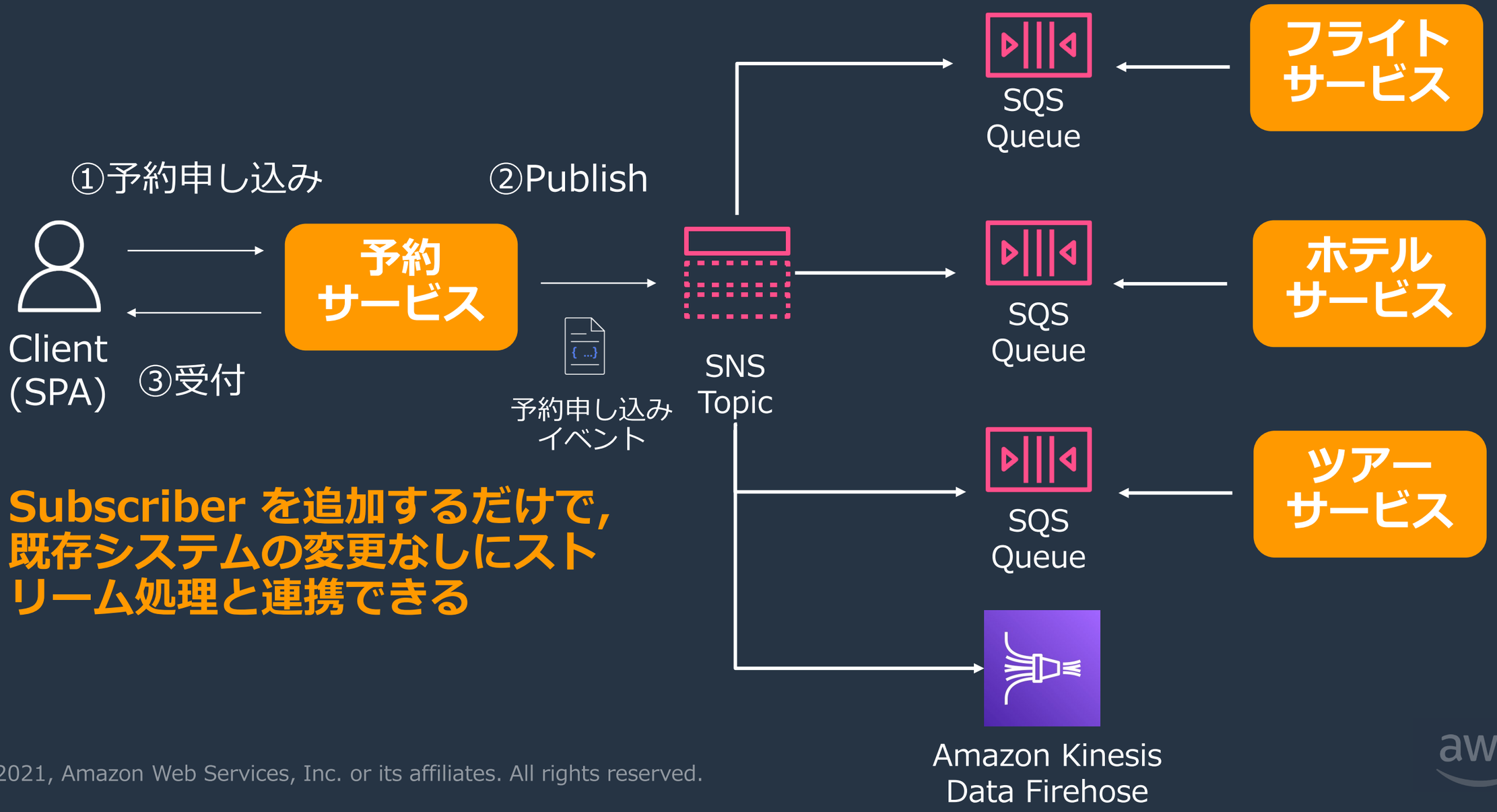
- **既に DynamoDB Streams に Consumer が 2 つ登録されている. これ以上 DynamoDB Streams の Consumer は増やせない**
- **レコード順序の並べ替え, レコード重複排除は Flink アプリケーション内で実行可能. また処理の特性上データの並び順が厳密に守られている必要は無い. ストリームに冪等性や順序性は求めないことから, DynamoDB Streams である必要はない**

旅行サイトの予約データ分析アーキテクチャー



aws DevAx connect

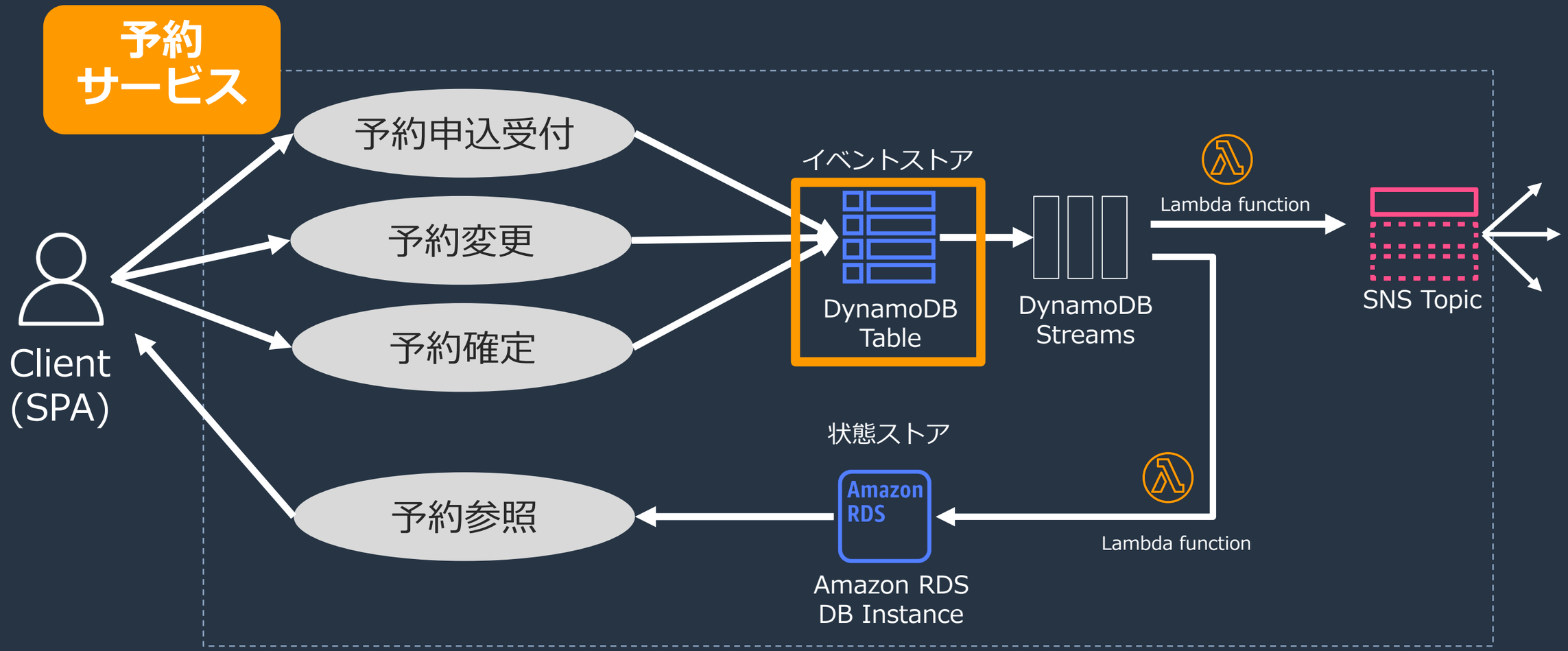
まとめ:イベント駆動アプリケーションとストリーム処理



**Subscriber を追加するだけで,
既存システムの変更なしにスト
リーム処理と連携できる**

まとめ:イベント駆動アプリケーションとストリーム処理

Subscriber の追加(だけ)では要件を満たせない場合は, 無理せず従来システムと同様にデータストアのストリーム連携機能も活用する



aws DevAx connect

付録:
ストリーム処理の性能問題と切り分け

ダッシュボードのデータ反映が遅れている！

何を調べますか？ どこから調べますか？

Source **Producer** **Data Stream** **Consumer** **Destination**

User Application

Kinesis Producer Library

Amazon Kinesis Data Streams

AWS Lambda

Amazon Elasticsearch Service

aws DevAx connect

まずはデータストリームを起点として調べる

システムの中央を起点にして切り分けを行う

- Data Stream で問題が発生しているのか？
- Producer で問題が発生しているのか？
- Consumer で問題が発生しているのか？

User Application

Kinesis Producer Library

Amazon Kinesis Data Streams

AWS Lambda

Amazon Elasticsearch Service

メトリクスによる切り分け(Kinesis Data Streams)

問題の発生個所だけでなく, ストリーム全体で同様の傾向にあるか, 特定シャードで偏った傾向がみられるかを合わせて切り分けるとよい

メトリクス名	説明	メトリクスの見方
ReadProvisionedThroughputExceeded	単位時間あたりの GetRecords リクエストがシャードの制限(2MB/s, 5 TPS)を超過した回数	本メトリクスの値がストリーム全体で恒常的に 1 以上を記録している場合は, キャパシティの増強または拡張ファンアウトの導入が必要. 特定のシャードで発生している場合は, 特定シャードに書き込みが偏っている可能性がある
WriteProvisionedThroughputExceeded	単位時間あたりの PutRecord, PutRecords リクエストがシャードの制限(1MB/s, 1000 records/s)を超過した回数	本メトリクスの値がストリーム全体で恒常的に 1 以上を記録している場合は, キャパシティの増強が必要. 特定のシャードで発生している場合は, 特定シャードに書き込みが偏っている可能性がある
GetRecords.Success PutRecord.Success PutRecords.Success	ストリームに対する各 API の実行回数と成功回数を記録したもの. 平均(Average) の統計から API の成功率が確認できる	成功率が急激に低下している場合はサービスで問題が発生している可能性がある

https://docs.aws.amazon.com/streams/latest/dev/monitoring-with-cloudwatch.html#kinesis-metrics

データストリームの問題が疑われるケースと対処

WriteProvisionedThroughputExceeded が 1 以上を記録している
＝書き込み処理がスロットルされている

単純にキャパシティ不足に起因していることが多いが, 書き込みが特定シャードに偏っている場合はクライアントの実装見直しをお勧め

ストリーム全体で発生している場合

- 帯域使用量起因でスロットルされている場合は, シャード追加によるストリーム全体のスループットの増強を検討
- Records per Sec 起因でスロットルされている場合もシャード追加が有効だが, 帯域使用量が低い場合はクライアント側でレコードを集約することでも対処可能

一部のシャードでのみ発生している場合

- パーティションキー設計を見直し, データが適切に分散されるようにする

データストリームの問題が疑われるケースと対処

ReadProvisionedThroughputExceeded が 1 以上を記録している
＝読み取り処理がスロットルされている

読み取り処理のスロットリングは, 複数 Consumer が同一のデータストリームからデータを取得している場合に起こりやすい

ストリーム全体で発生している場合
- シャード追加によるストリーム全体のスループットの増強
- 拡張ファンアウトを検討

一部のシャードでのみ発生している場合
- Producer が書き込みに使用するパーティションキーの設計を見直し, データが適切に分散されるようにする

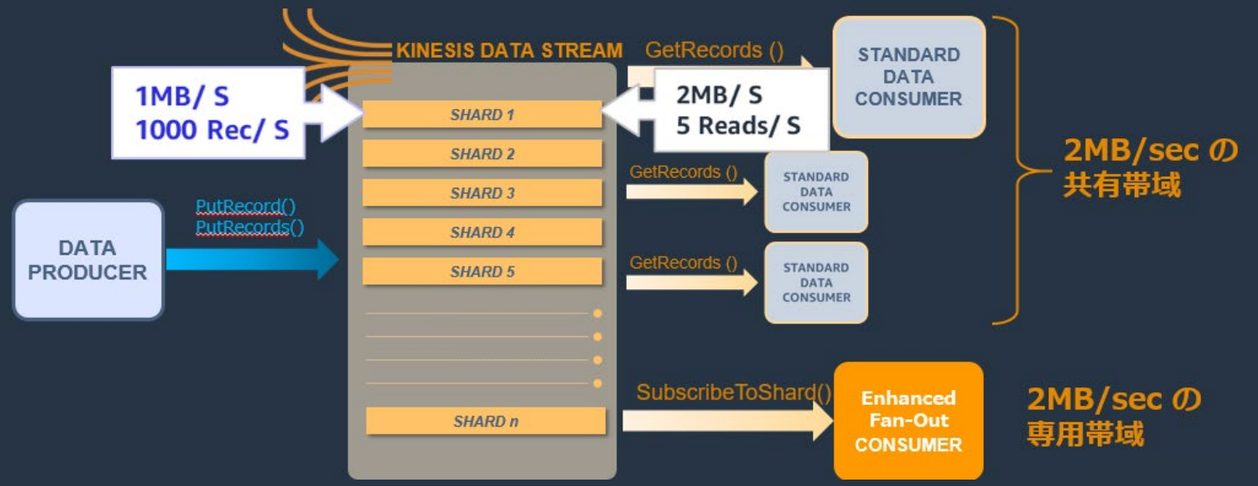
Amazon Kinesis Data Streams のコンセプト(再掲)

コンシューマー数によって起こりうる問題

- シャードごとの出力帯域の上限は 2MB/s, 入力帯域は 1MB/s. 出力は入力の 2 倍と考えることができる
- 標準コンシューマーが 3 つ以上存在する場合,シャードの帯域使用率が高いほど, スロットリングが起こりやすくなる

コンシューマーが複数存在する場合の考慮点

- シャードを増やして各シャードの帯域使用率を下げると, スロットリングは起こりにくくなる
- 拡張ファンアウトを用いることでも競合を避けることができる



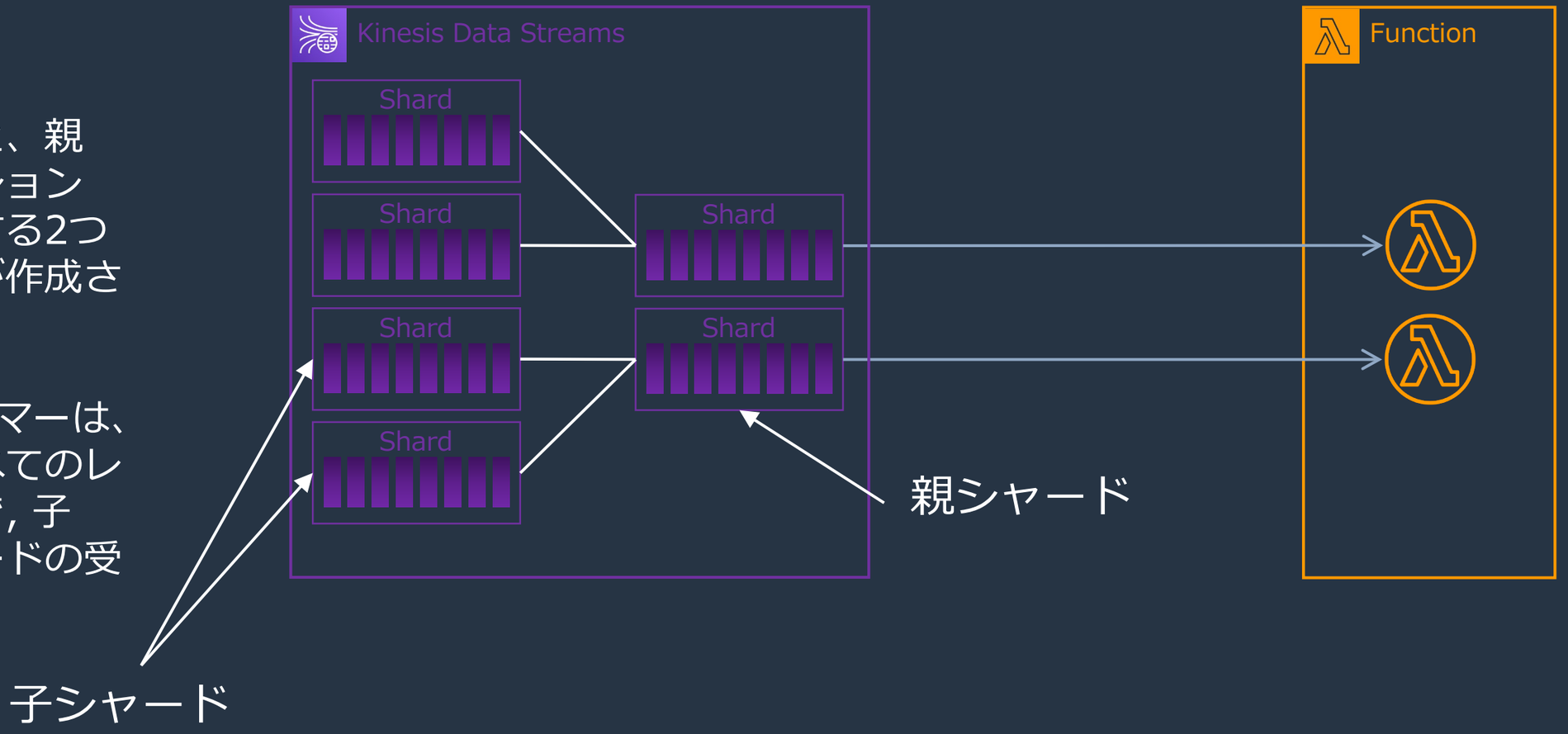
キャパシティ増強時の注意点 (Kinesis Data Streams)

キャパシティを増強してもすぐに遅延は解消されるとは限らない

シャード追加 ≒ シャード分割

- シャードを分割すると、親シャードのパーティションキースペースを分割する2つの新しい子シャードが作成される

- 一般的に, コンシューマーは、親シャードからのすべてのレコードを処理するまで, 子シャードからのレコードの受信を開始しない



aws DevAx connect

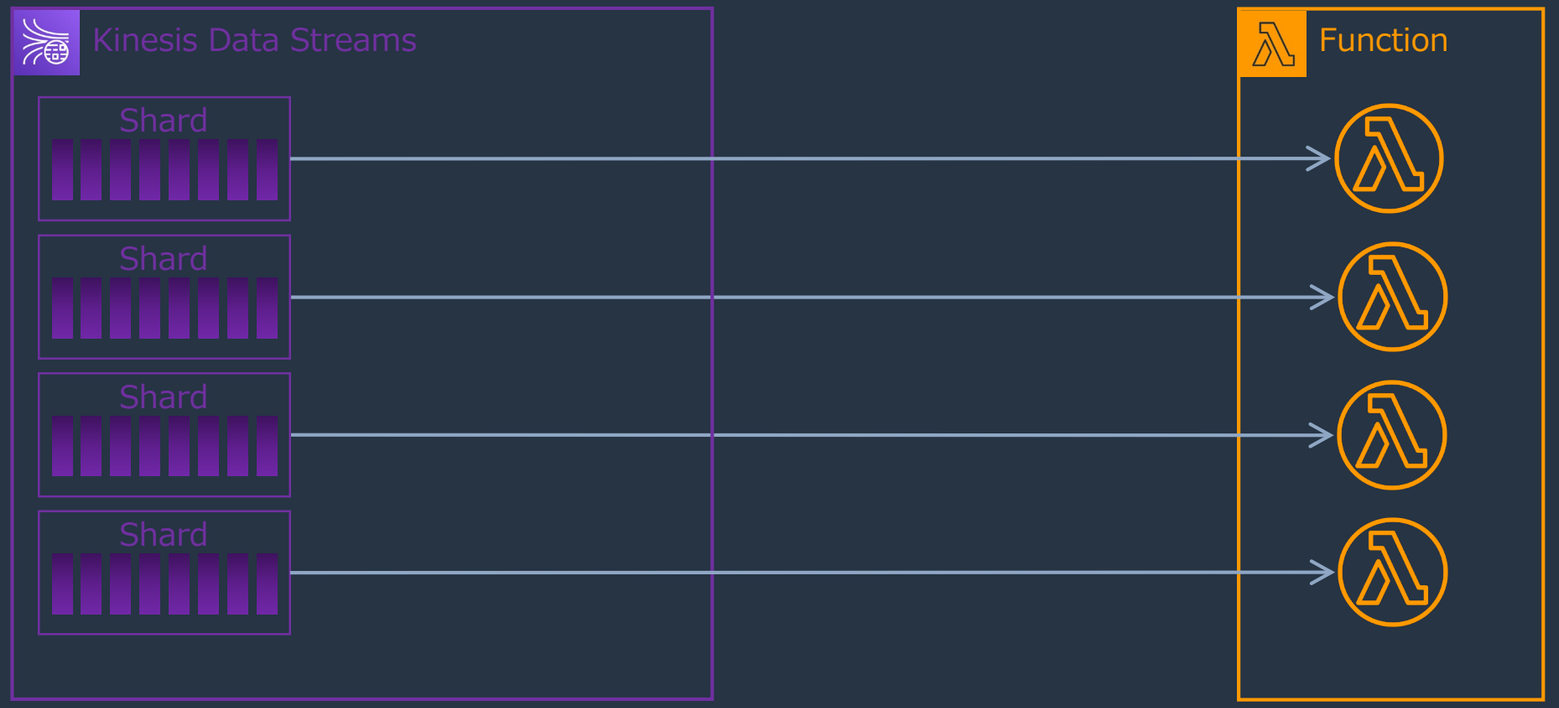
キャパシティ増強時の注意点 (Kinesis Data Streams)

キャパシティを増強してもすぐに遅延は解消されるとは限らない

シャード追加 ≒ シャード分割

- 親シャードの処理が全て終わると, 子シャードの処理が始まる

- Lambda などはこの時点から, 子シャードの数に合わせてスケールする



集約時の注意点

集約は有用な機能だが, 特性を理解して活用すること

極端な集約は避ける

- Kinesis Data Streams の最大レコードサイズは 1 MB
- 1 MB ぎりぎりまでユーザーレコードを集約した場合, 今度はシャード間の帯域使用率に偏りが出る可能性がある

集約解除を考慮する

- KPL により集約されたレコードは, 処理時に集約の解除が必要
- 一部サービスは Kinesis Data Streams からレコードを取得する際, 自動で集約が解除される点にも注意する
  - Amazon Kinesis Data Firehose など

Consumer の遅延が疑われる場合は？

Consumer で問題が発生しているのか，ターゲットのデータストアで問題が発生しているかを確認する

メトリクスによる切り分け(Amazon ES)

ターゲット側で書き込み処理をスロットルさせていないか, エラーにより書き込みに失敗していないか確認する

メトリクス名	説明	メトリクスの見方
2xx, 3xx, 4xx, 5xx	各ステータスコードの発生回数	5xx が発生している場合は何らかのサーバーエラーの発生が疑われる. 4xx が発生している場合はリクエストの問題, もしくはペイロードサイズの超過やキュー溢れが疑われる
IndexingLatency IndexingRate	IndexingLatency は ドキュメントの Indexing 処理所要時間. IndexingRate は単位時間あたりの Indexing リクエストの統計	IndexingRate が変わらない, もしくは減少しているにも関わらず IndexingLatency が増加傾向にある場合, リソースボトルネックを確認しスケールなどで対処する
ThreadpoolWriteRejected CoordinatingWriteRejected PrimaryWriteRejected ReplicaWriteRejected	書き込みリクエストが拒否された回数	これらのメトリクスが増加している場合, キューあふれ, または Indexing Request 用のヒープ領域不足が想定される. 恒常的に発生している場合はドメインをスケールさせるか, クライアント側のリクエスト方法が妥当か確認したうえで見直す

https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/es-managedomains-cloudwatchmetrics.html
https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/aes-handling-errors.html

メトリクスによる切り分け(Lambda)

関数内部のエラーであるか, データ起因のエラーであるか, 単純に処理に時間がかかっているか, 呼び出しがスロットルされているかを見極める

メトリクス名	説明	メトリクスの見方
Invocations	Function の実行回数	Function が正しく呼び出されているかをまずは確認する また, Duration に変化がないものの Invocations が増加している場合はデータ量が増加している可能性がある. オーバーヘッド削減のためにレコードバッチのサイズを増やすことも検討する
IteratorAge	ストリームがレコードを受信してから, Poller がイベントを Function に送信するまでの時間	Kinesis Data Streams 側の GetRecords.IteratorAgeMilliseconds(Stream-Level) ないし IteratorAgeMilliseconds(Shard-Level) から遅延が判断できる場合, 本メトリクスを併用することでどの Consumer が遅延しているか判断できる
Duration	Function の実行時間	Duration が増加している場合は Lambda Function 側の問題が疑われる. 遅延の原因はロジックの問題, リソース不足, 処理データ量の増加, Function 内から呼び出している別サービスの問題など様々. CloudWatch Logs に出力されているログや AWS X-Ray などから調査する
Errors	エラーの発生回数	ロジックの問題, 不正データの問題などが疑われる. ログから調査を行い対処する. 不正データ起因で発生している問題については, 不正データを退避し後続処理を実行するなどの対処を検討する
Throttles	Function の呼び出しがスロットルされた回数	関数に対する予約された同時実行数が少ないか, アカウント全体の同時実行数が不足している. 関数の設定変更もしくは上限緩和申請で対処する

https://docs.aws.amazon.com/ja_jp/lambda/latest/dg/monitoring-metrics.html
https://docs.aws.amazon.com/lambda/latest/dg/lambda-troubleshooting.html

メトリクスによる切り分け(Kinesis Data Streams)

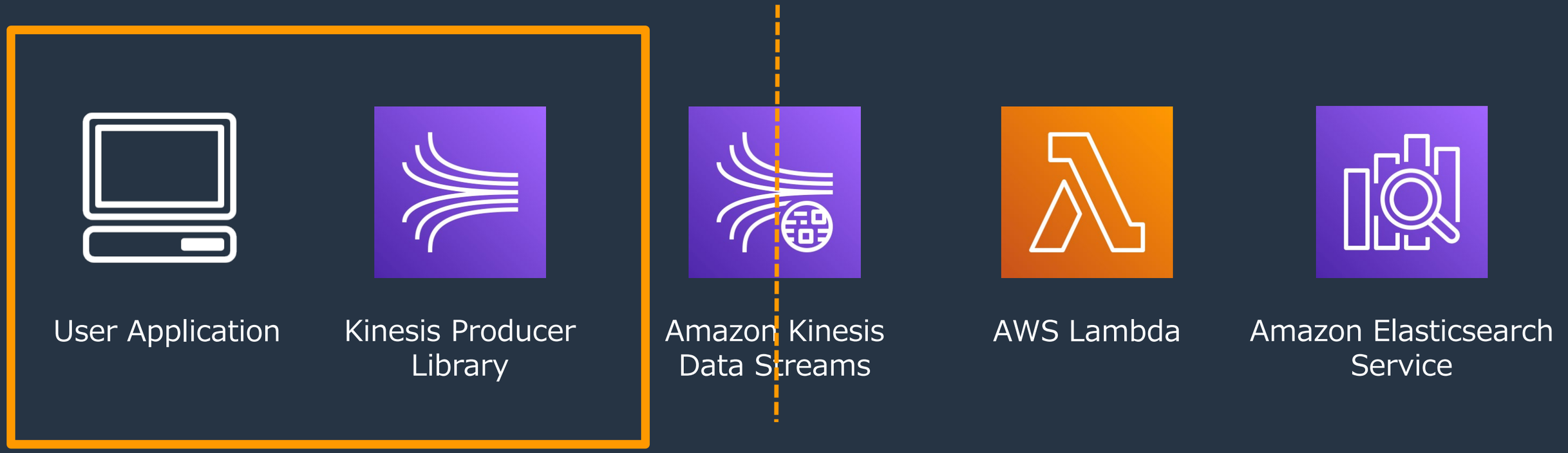
問題の発生個所だけでなく, ストリーム全体で同様の傾向にあるか, 特定シャードで偏った傾向がみられるかを合わせて切り分けるとよい

メトリクス名	説明	メトリクスの見方
GetRecords.IteratorAgeMilliseconds (Stream-Level) IteratorAgeMilliseconds(Shard-Level)	現在の時刻と Consumer による GetRecords 呼び出しの最後のレコードがストリーム(シャード)に書き込まれた時刻との差	当メトリクスの値が 0 に近いほど, Consumer は限りなく最新のデータを取得できていることになる. 値が継続的に増加している場合は, Consumer の問題が疑われる. 瞬間的なスパイクは Consumer の実装, 運用によっては起こりえるため, すぐに値が減少するようであれば基本的に問題なしと判断する

https://docs.aws.amazon.com/streams/latest/dev/monitoring-with-cloudwatch.html#kinesis-metrics

Producer の遅延が疑われる場合は？

Producer が発生したイベントを遅延, エラーなくデータストリームに送信できているかを見る



メトリクスによる切り分け(KPL)

- イベントが Producer に送り込まれてから実際にデータストリームに書き込まれるまでのタイムラグや, 処理エラー, 送信エラーが発生していないかを確認する
- ネットワーク等の問題でデータストリームまでデータが到達していないようなエラーは, データストリーム側のメトリクスからは読み取れない. Producer 側のメトリクスの確認が必要

メトリクス名	説明	メトリクスの見方
BufferingTime	ユーザーレコードが KPL に到着してからバックエンドに送信されるまでの時間	値が増加している場合, 処理に時間がかかっている可能性が疑われる. ErrorsByCode が増加していればエラーの原因を, ErrorsByCode が増加していなければ処理上のボトルネックを調べる
ErrorsByCode	各種類のエラーコードの数	値が増加している場合送信エラーが発生している可能性が疑われる. Kinesis Data Streams 側の PutRecord.Success との相関を確認し, 相関があれば Kinesis Data Streams 側の問題が疑われる. なければネットワークの問題や権限不足等のクライアント側の問題を疑う
UserRecordsReceived UserRecordsPending UserRecordsPut	KPL が受信したレコード数, KPL 内で保留状態にあるレコード数, Kinesis Data Streams に PUT されたレコード数	Pending が増加しており, Received と Put のバランスが崩れている場合は, KPL 内で処理が滞留している可能性がある

https://docs.aws.amazon.com/ja_jp/streams/latest/dev/monitoring-with-kpl.html

その他, 日常的なモニタリングが推奨される項目

タイムラグ

- アプリケーションがイベントを生成してから連携先サービスにデータが取り込まれるまでのタイムラグは SLA 内に収まっているか
- SLA 内には収まっているものの, 増加傾向に無いか

スループット

- スループットの急激な増加あるいは低下は発生していないか
- スループットが緩やかに増加していないか

キャパシティ, リソース使用率

- リソース使用率が上限に達していないか
- 割り当てリソースに対する使用率が極端に低くないか

モニタリングに用いる主なメトリクス

Kinseis Data Streams の場合

タイムラグ

- IteratorAgeMilliseconds

スループット,キャパシティ，リソース使用率

- [GetRecords,PutRecord,PutRecords].[Latency,Bytes,Records]
- [Incoming,Outcoming][Bytes,Records]
- ReadProvisionedThroughputExceeded
- WriteProvisionedThroughputExceeded

まとめ

- リアルタイム分析とストリーム処理
  - バッチ処理とストリーム処理にはそれぞれメリット, デメリットがある
  - ストリーム処理の特性を理解し正しく活用しよう
- AWS におけるストリーム処理の実装
  - ユースケース, リアルタイム要求, データ量に応じて適切なフルマネージドサービスを組み合わせよう
  - ストリーム処理に頼らず, AWS サービスの組み合わせで対処することも検討しよう
- イベント駆動アプリケーションとストリーム処理の美味しい関係
  - データ分析サービスも連携サービスの一つと考えよう
- 付録: ストリーム処理の性能問題と切り分け
  - 中心のデータストリームを起点に切り分けていこう

Thank you!

Takayuki Enomoto
Analytics Specialist Solution Architect
Amazon Web Services Japan K.K.

aws DevAx connect

Q & A