

サーバーレスを活用した イベント駆動アーキテクチャ

岡田 信夫

技術統括本部 ソリューションアーキテクト
Amazon ウェブ サービス ジャパン合同会社

自己紹介



岡田 信夫 (ポール)

Twitter/ @bulbulpaul

アマゾン ウェブ サービス ジャパン合同会社
ソリューションアーキテクト

[関心のある技術]

Kotlin, Python, Serverless

本セッションのゴールと注意点

セッションのゴール

- ・ 同期型のシステムとイベント駆動アーキテクチャの違いや活用シーンを理解する
- ・ イベント駆動アーキテクチャでのAWSサービス活用方法を理解する

お話しないこと

- ・ 各AWSサービスの概要
- ・ マイクロサービスアーキテクチャの概要

イノベーションとマイクロサービス

イノベーションには戦略的な技術マネジメントが重要



イノベーションの
加速

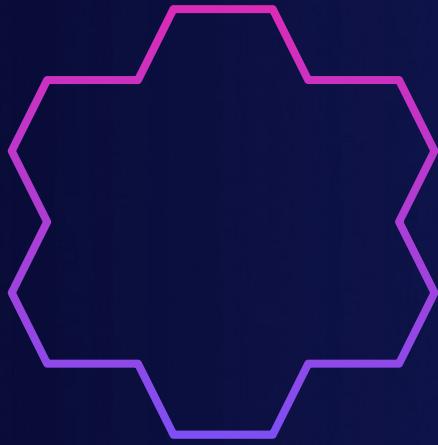


データ活用の
推進

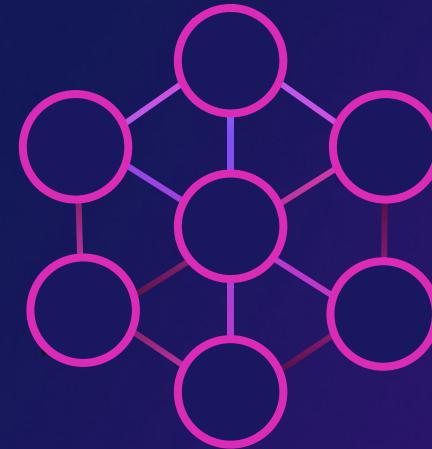


新たな顧客体験の構築

マイクロサービス化で拡張性や俊敏性を高める



Monolith
すべてを実行

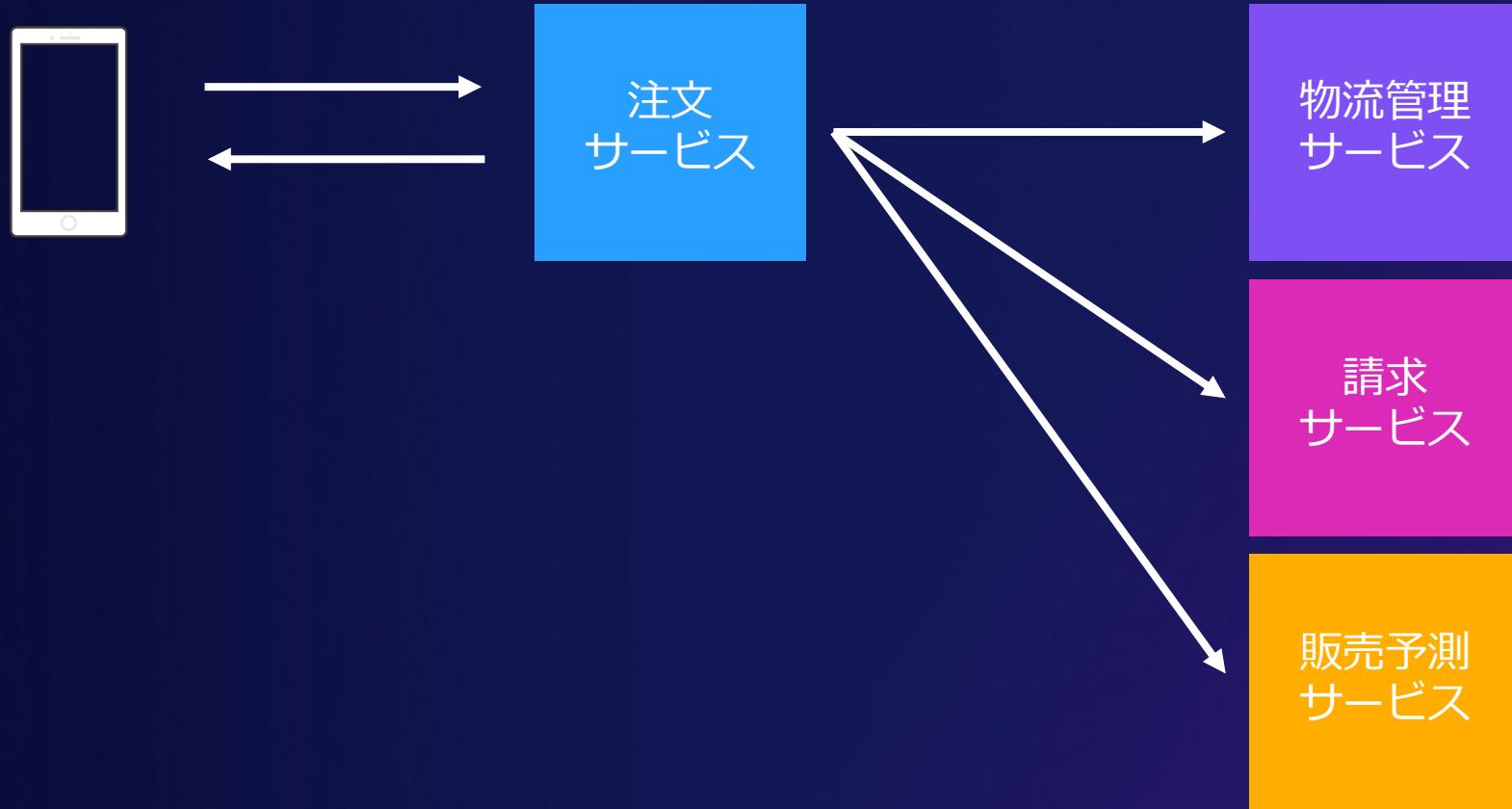


Microservices
各々がひとつのことを行

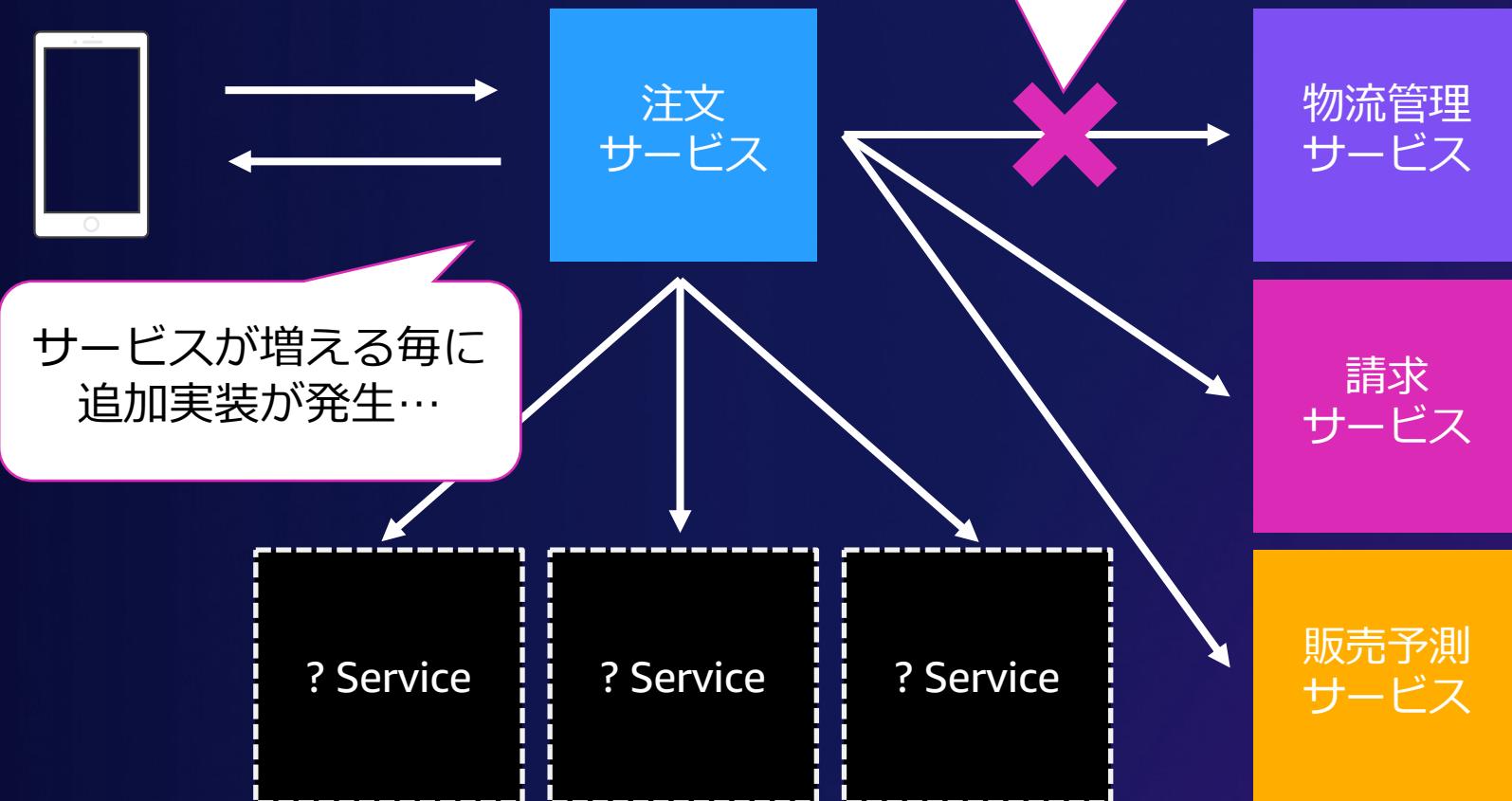
Eコマース注文システムの例



Eコマース注文システムの例

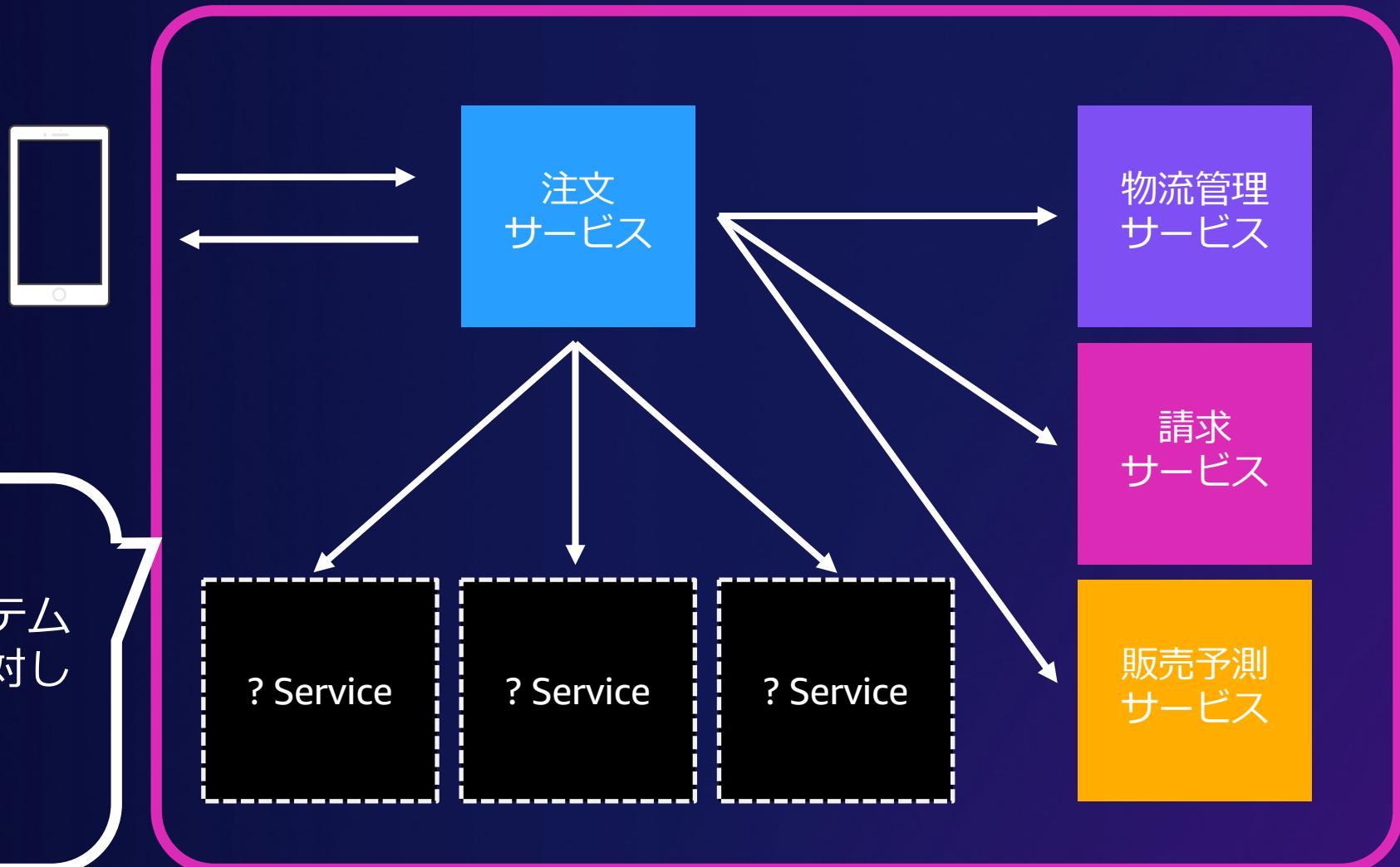


Eコマース注文システムの例

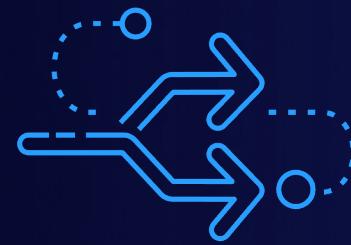


マイクロサービスアーキテクチャを構築する場合、アプリケーションや外部サービスとの統合により、**更に密結合なシステムになる場合がある**

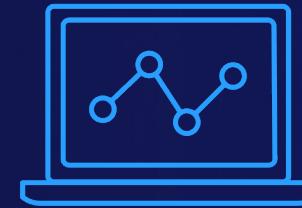
Eコマース注文システムの例



イベント駆動アーキテクチャで課題を解決



コンポーネントの分離や
分散化



マイクロサービス間
の連携



データフロー
の整理

イベント駆動アーキテクチャ



Event

[i-'vent] noun

状態が変更されたことを
示すシグナル

イベントの特徴

送信先へ変更の指示は
行わない



関心の分離や疎結合化

実行結果を受取らない



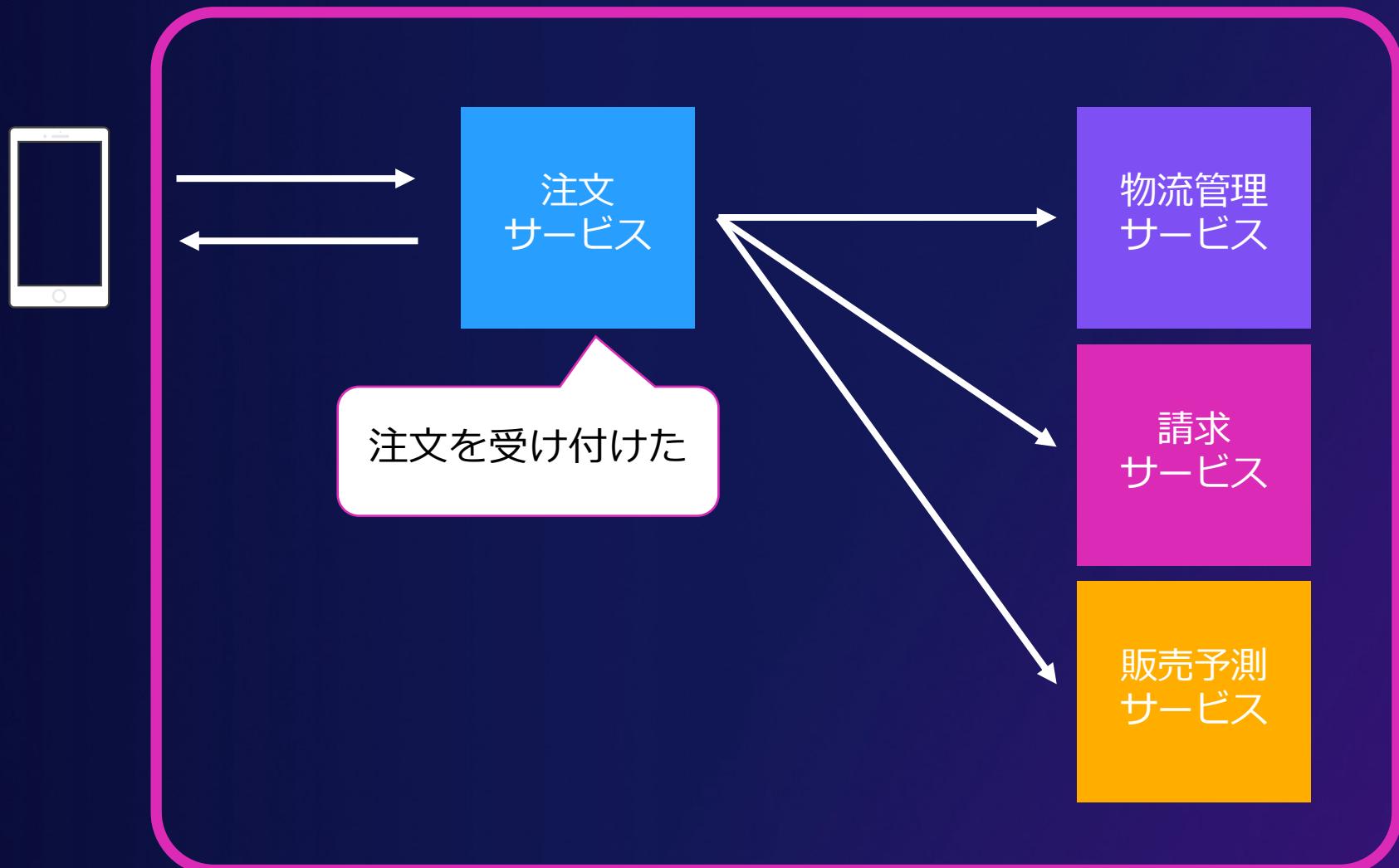
非同期処理への適合

イミュータブル

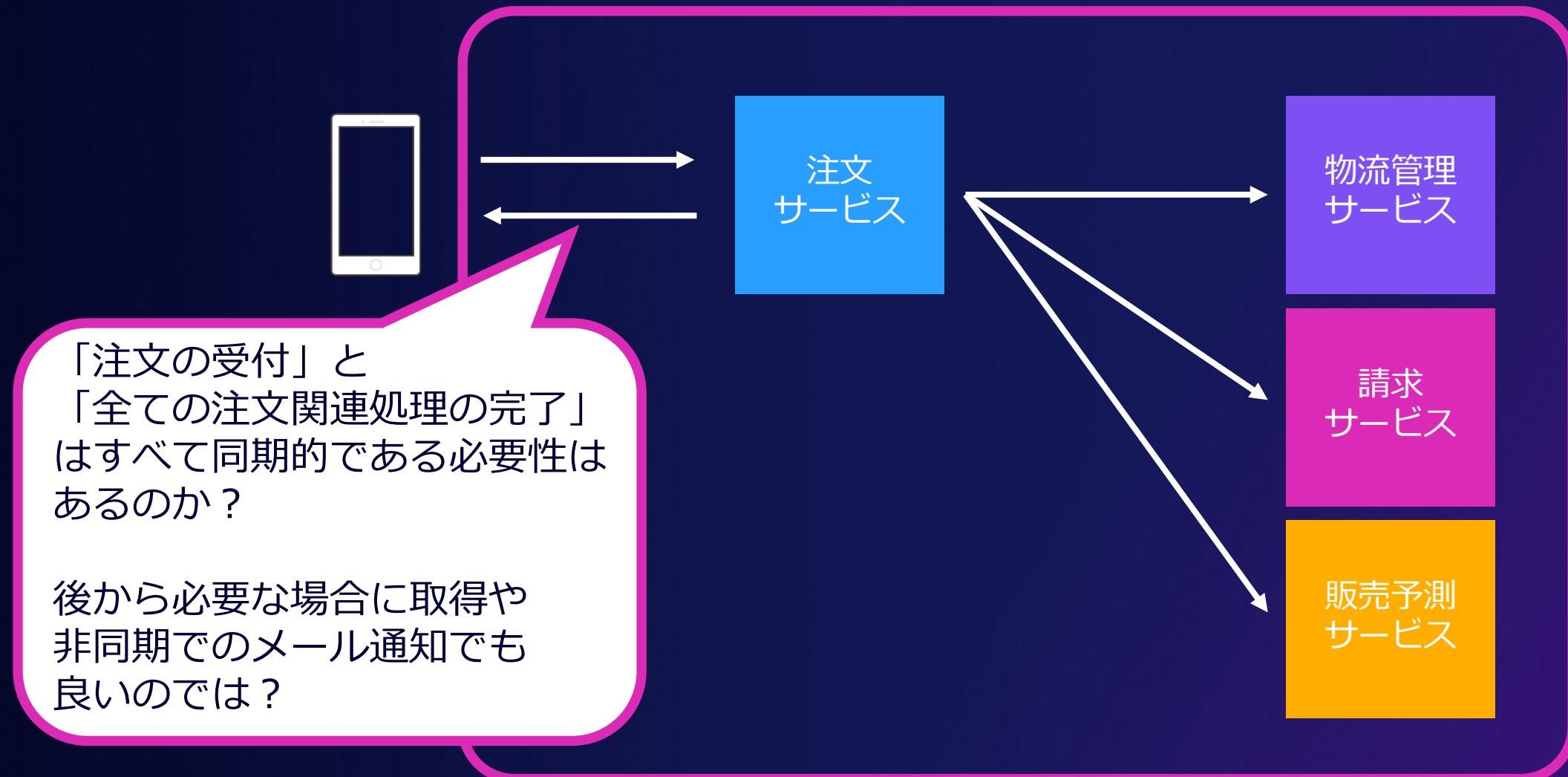


コピーすることで
容易にスケールが可能

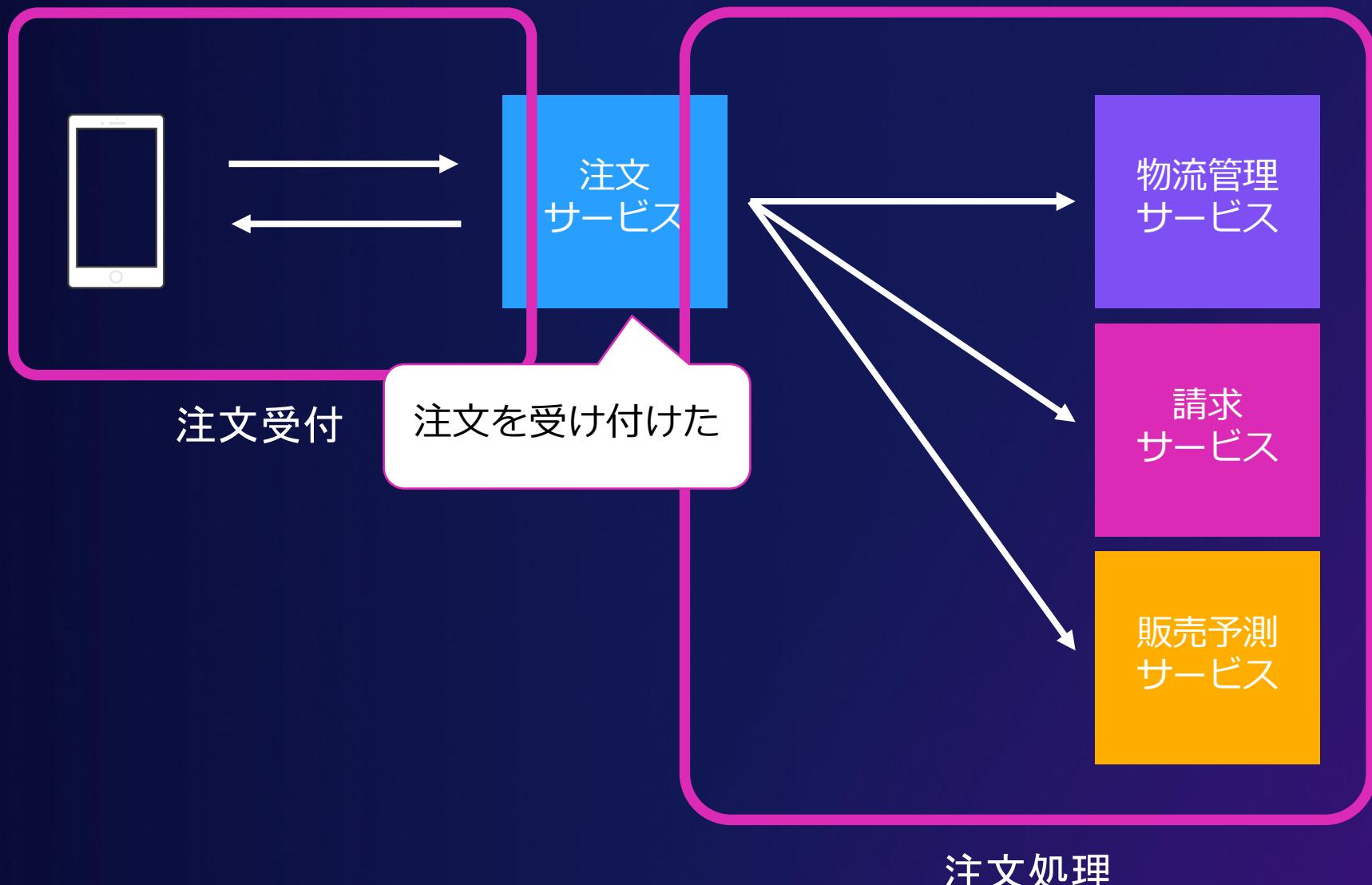
Eコマース注文システムの例



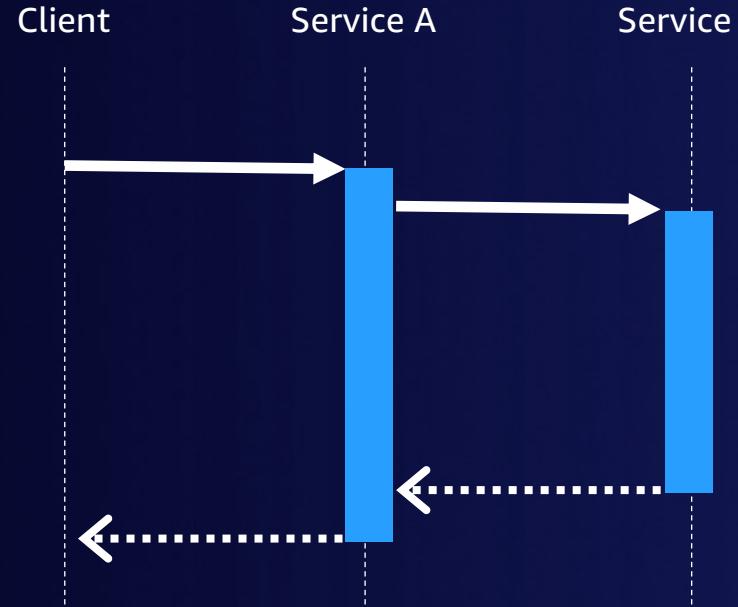
Eコマース注文システムの例



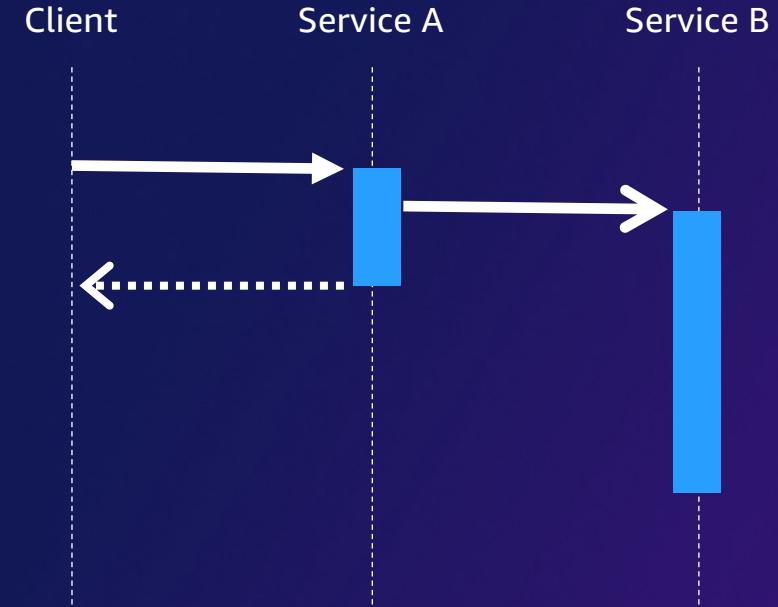
Eコマース注文システムの例



非同期化による応答性の改善と依存性の削減

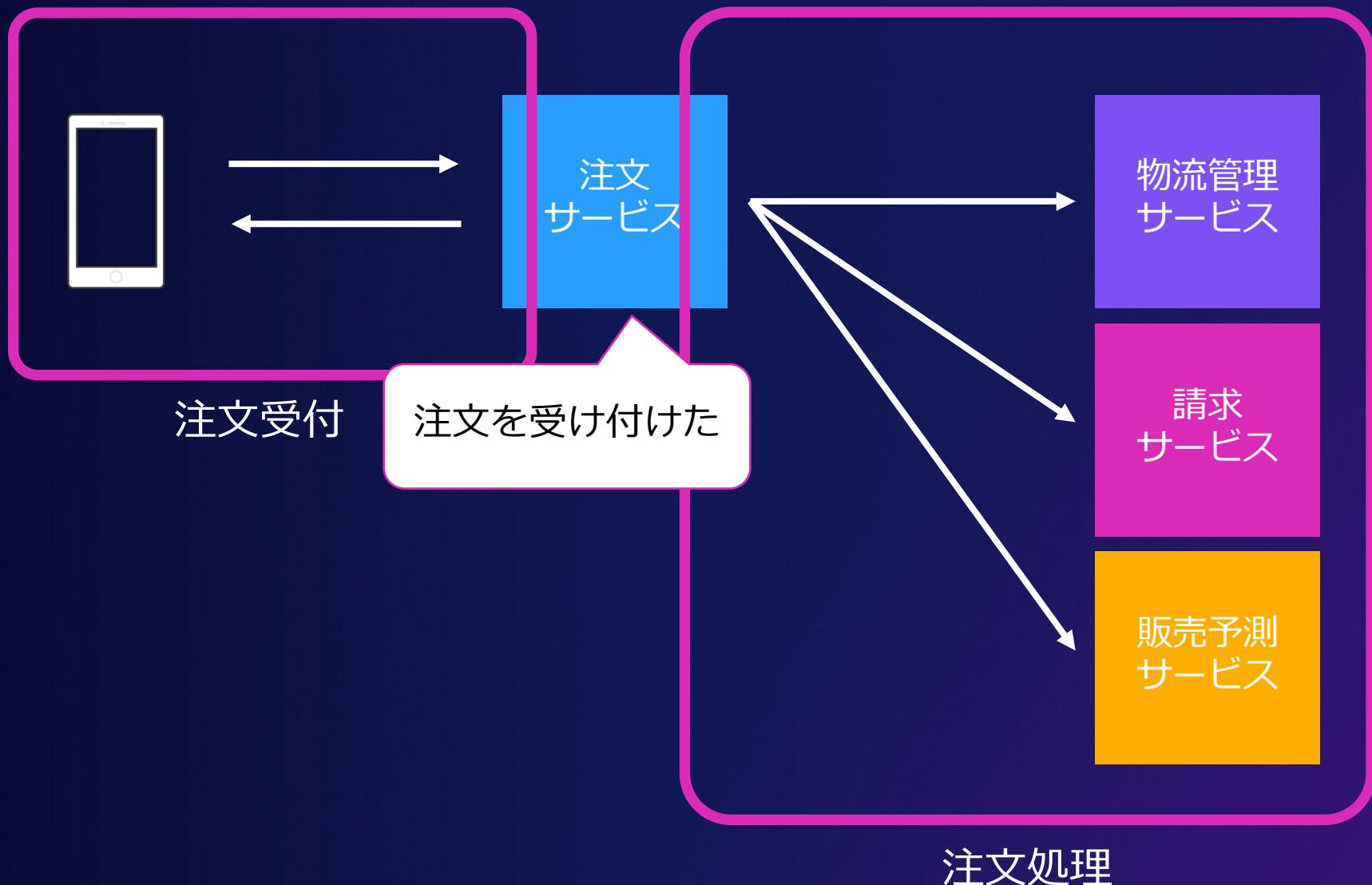


同期コマンド



非同期イベント

Eコマース注文システムの例



イベント駆動の連携処理をサーバーレスで



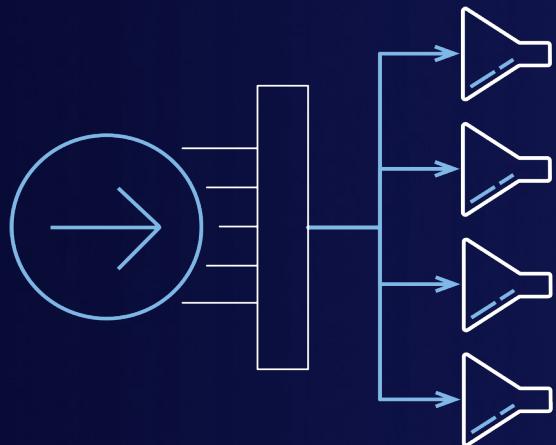
利用分だけの従量課金



最小限の運用

イベント駆動での連携要素

イベントトレーナー



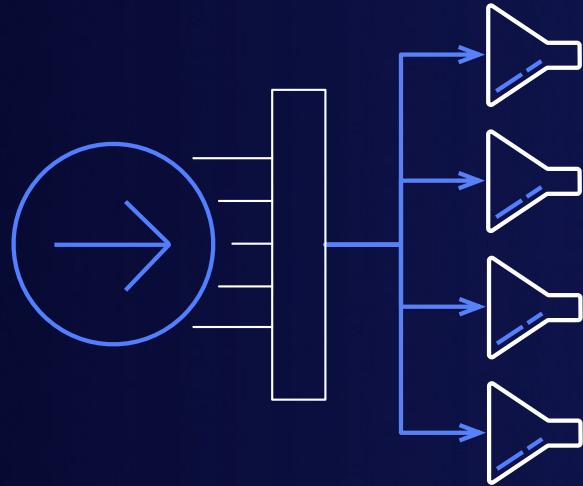
プロデューサーと
コンシューマーを
お互いに抽象化

イベントストア



サービスが処理できるよ
うになるまで
バッファリング

イベントルーター



プロデューサーと
コンシューマを抽象化



イベントの選択と
フィルタリング

AWSでのイベントルーター

トピック

AWS サービス

特徴



Amazon Simple
Notification Service

AWS Lambda/Amazon SQS/HTTP の
ターゲット
数百万件のサブスクリプション
メッセージの属性のフィルタリング

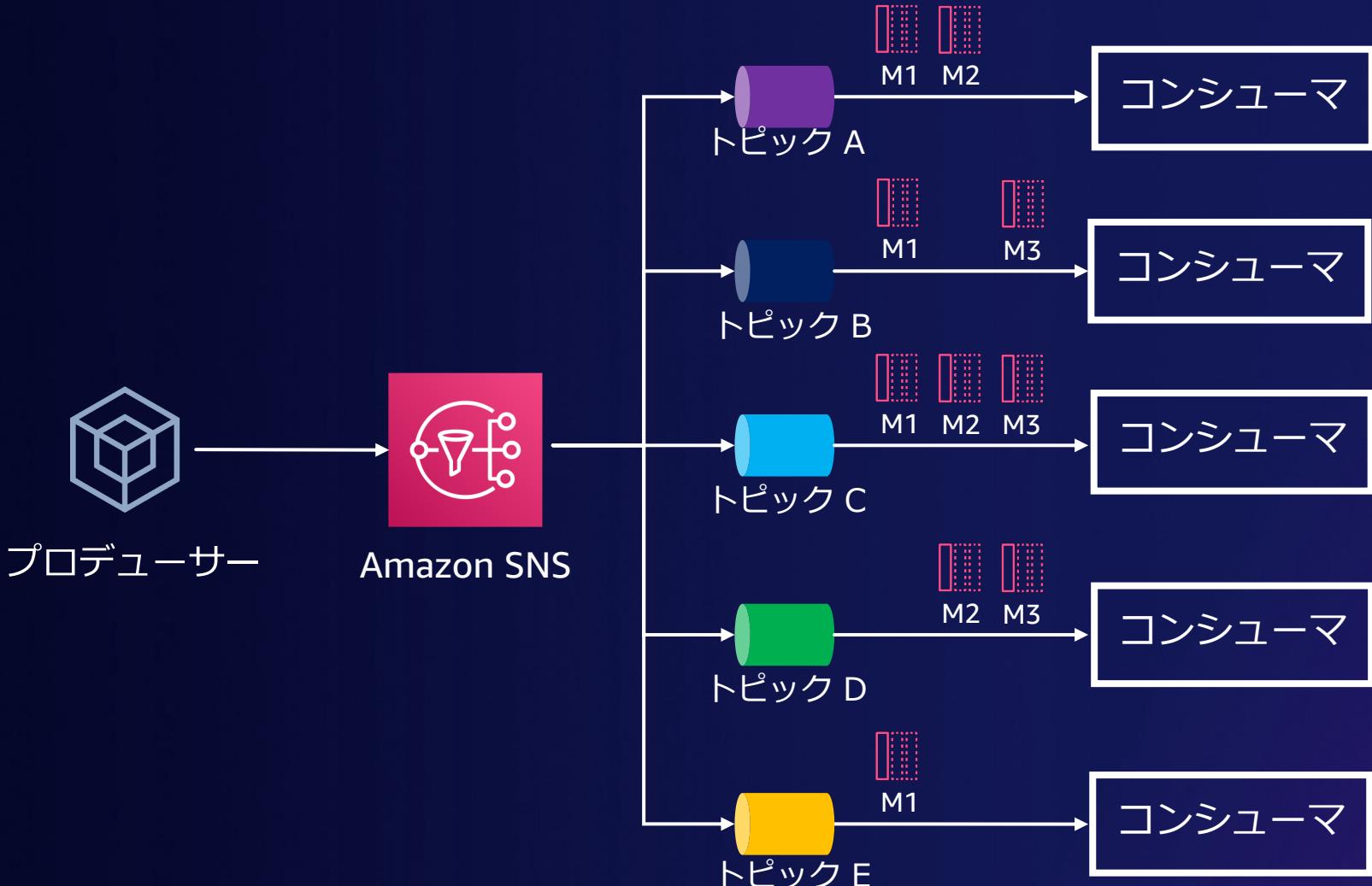
イベントバス



Amazon EventBridge

15以上のAWSターゲットと統合
SaaSのネイティブイベントソース
メタデータとペイロードのルーティング

トピックベースのイベントトレーラー



特徴

- イベントとトピックのマッピング
- サブスクリプションでのフィルタリング

イベントとトピックのマッピング

注意点

- 送信者の複雑さが時間と共に増加



メッセージフィルター

- プロデューサーは、メッセージをルーティングする必要がない
- 購読者は、関心のあるメッセージのためにフィルタリングすることが可能

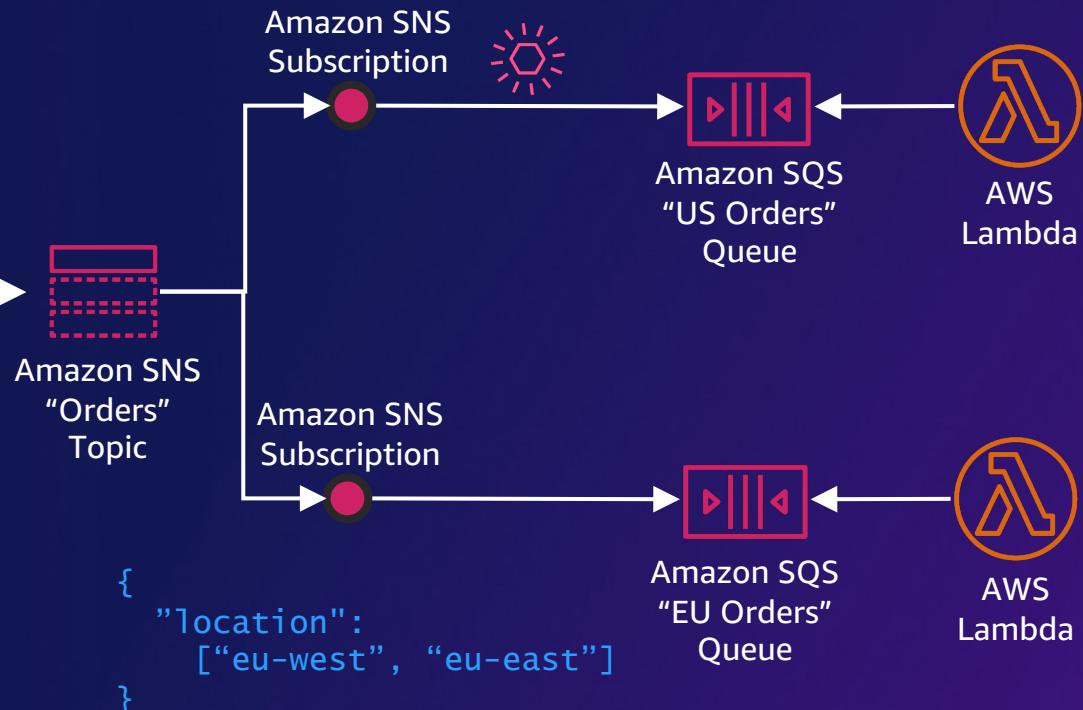
Message Attributes

```
{  
    "location": "us-west"  
}
```

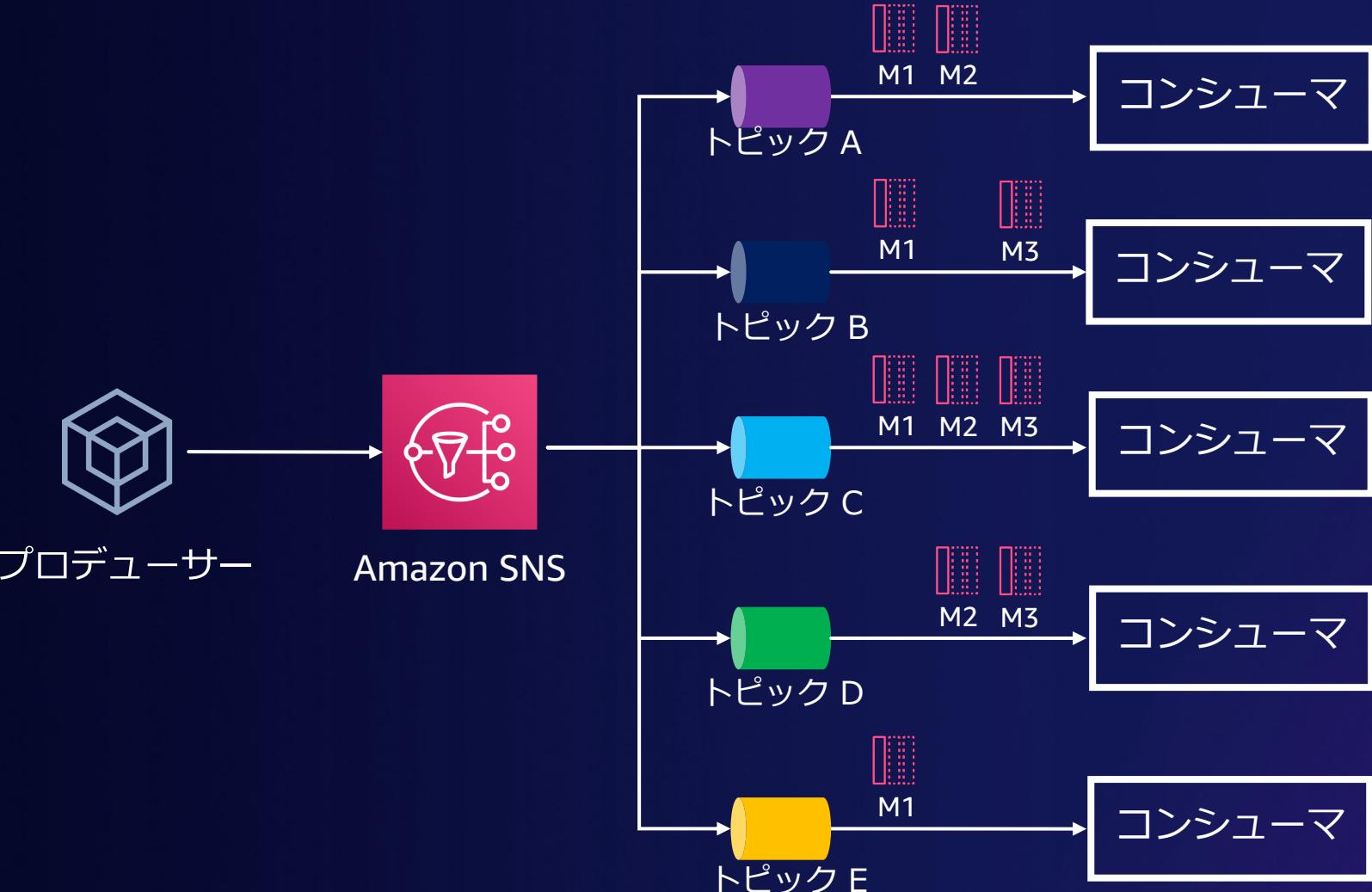


プロデューサー

```
Filter Policy  
{  
    "location":  
        ["us-west", "us-east"]  
}
```



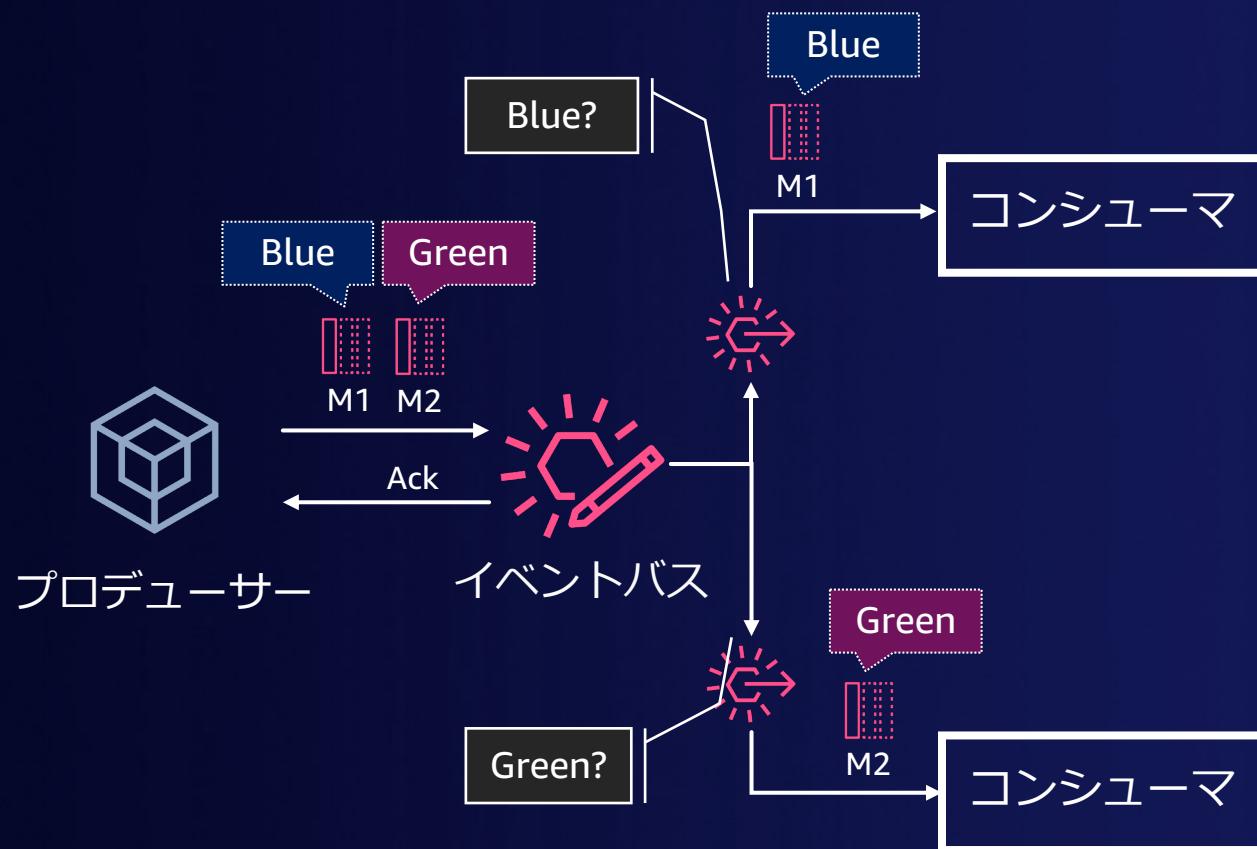
トピックベースのイベントトレーラー



特徴

- イベントとトピックのマッピング
- サブスクリプションでのフィルタリング

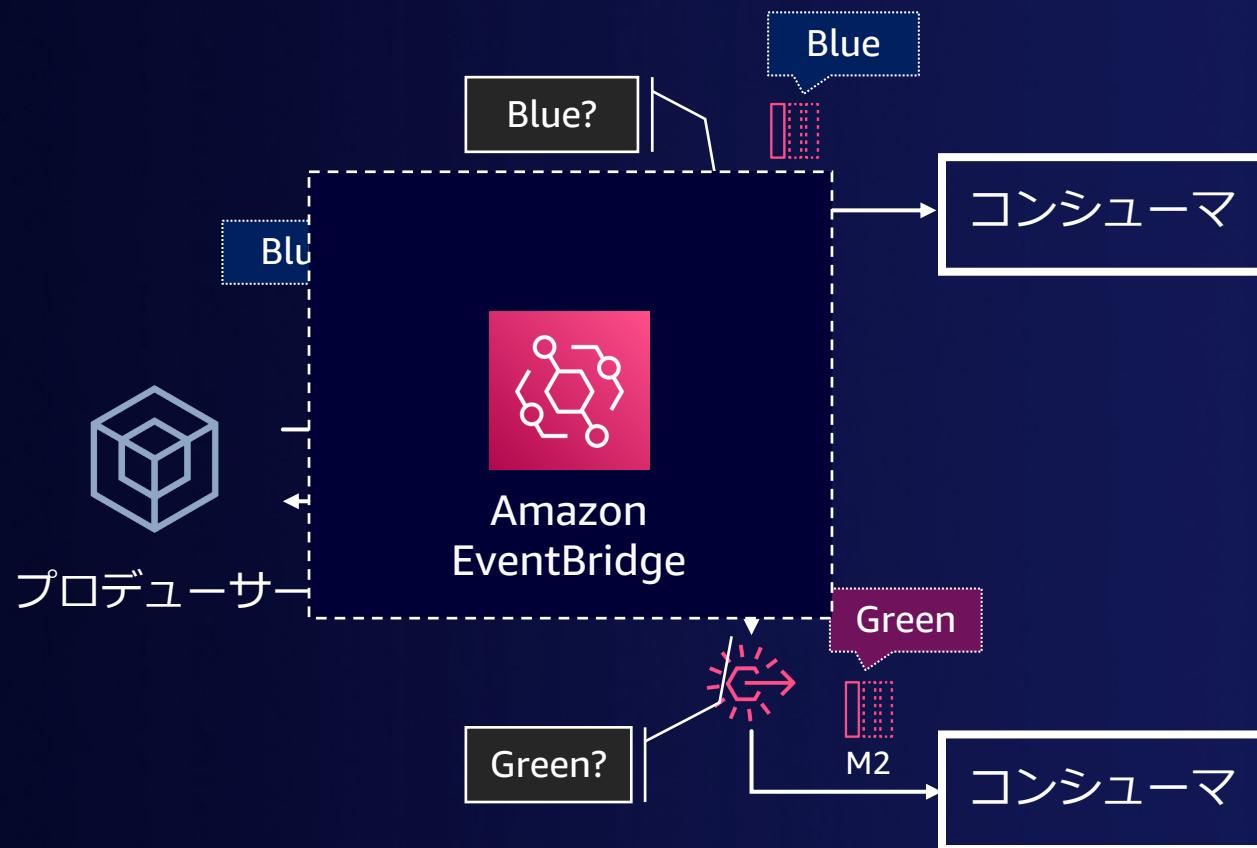
イベントバスでのイベントルーター



特徴

- ルールベースでのルーティング
- 送信側 / 受信側ともにイベント処理が効率的

イベントバスでのイベントルーター



特徴

- ルールベースでのルーティング
- 送信側 / 受信側ともにイベント処理が効率的

EventBridge コンテンツベースのルーティングルール

EventBridge イベント例

```
{  
  "source": "com.orders",  
  "detail-type": "OrderCreated",  
  "detail": {  
    "metadata": {  
    },  
    "data": {  
      "order-id": "1073459984",  
      "created-at": "2021-11-26T16:05:09-04:00",  
      "price": 24.62  
      "currency": "JP",  
    }  
  }  
}
```

EventBridge コンテンツベースのルーティングルール

EventBridge イベント例

```
{  
  "source": "com.orders",  
  "detail-type": "OrderCreated",  
  "detail": {  
    "metadata": {},  
    "data": {  
      "order-id": "1073459984",  
      "created-at": "2021-11-26T16:05:09-04:00",  
      "price": 24.62  
      "currency": "JP",  
    }  
  }  
}
```

EventBridge ルール例

```
{  
  "detail": {  
    "data": {  
      "currency": ["JP", "US"]  
    }  
  }  
}
```

EventBridge コンテンツベースのルーティングルール

EventBridge イベント例

```
{  
  "source": "com.orders",  
  "detail-type": "OrderCreated",  
  "detail": {  
    "metadata": {},  
    "data": {  
      "order-id": "1073459984",  
      "created-at": "2021-11-26T16:05:09-04:00",  
      "price": 24.62  
      "currency": "JP",  
    }  
  }  
}
```

EventBridge ルール例

```
{  
  "detail": {  
    "data": {  
      "currency": ["JP", "US"]  
    }  
  }  
}
```

EventBridge コンテンツベースのルーティングルール

EventBridge イベント例

```
{  
  "source": "com.orders",  
  "detail-type": "OrderCreated",  
  "detail": {  
    "metadata": {},  
    "data": {  
      "order-id": "1073459984",  
      "created-at": "2021-11-26T16:05:09-04:00",  
      "price": 24.62  
      "currency": "JP",  
    }  
  }  
}
```

EventBridge ルール例

```
{  
  "detail": {  
    "data": {  
      "currency": ["JP", "US"]  
    }  
  }  
}
```

EventBridge コンテンツベースのルーティングルール

EventBridge イベント例

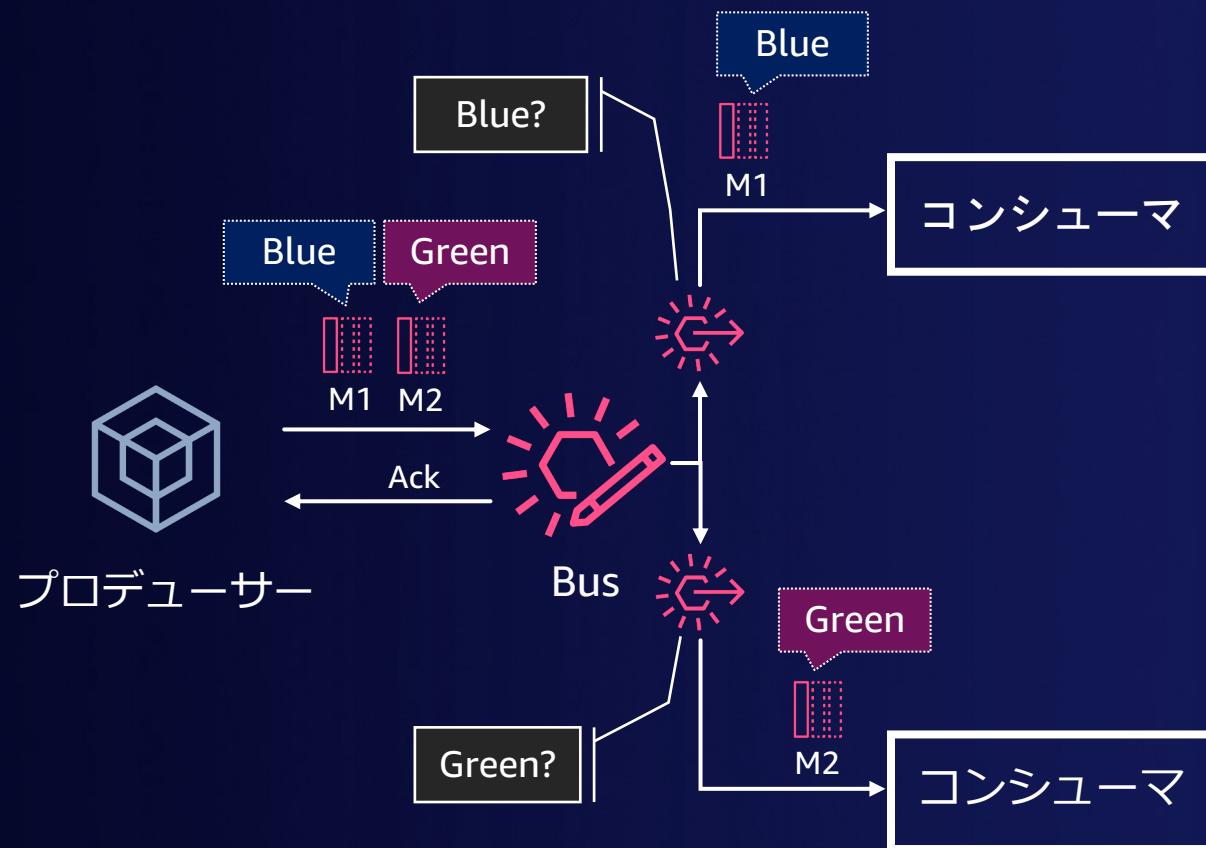
```
{  
  "source": "com.orders",  
  "detail-type": "OrderCreated",  
  "detail": {  
    "metadata": {},  
    "data": {  
      "order-id": "1073459984",  
      "created-at": "2021-11-26T16:05:09-04:00",  
      "price": 24.62  
      "currency": "JP",  
    }  
  }  
}
```

EventBridge ルール例

```
{  
  "detail": {  
    "data": {  
      "currency": ["JP", "US"]  
    }  
  }  
}
```

OR条件のルールにマッチ

イベントバスでのイベントルーター



特徴

- ルールベースでのルーティング
- 送信側 / 受信側ともにイベント処理が効率的

AWSでのイベントルーターの使い分け

AWS サービス

ユースケース例

トピック



Amazon Simple
Notification Service

- 高スループットや低レイテンシーが求められる
- 高いファンアウトが求められる (e.g. 数十～数百枚)

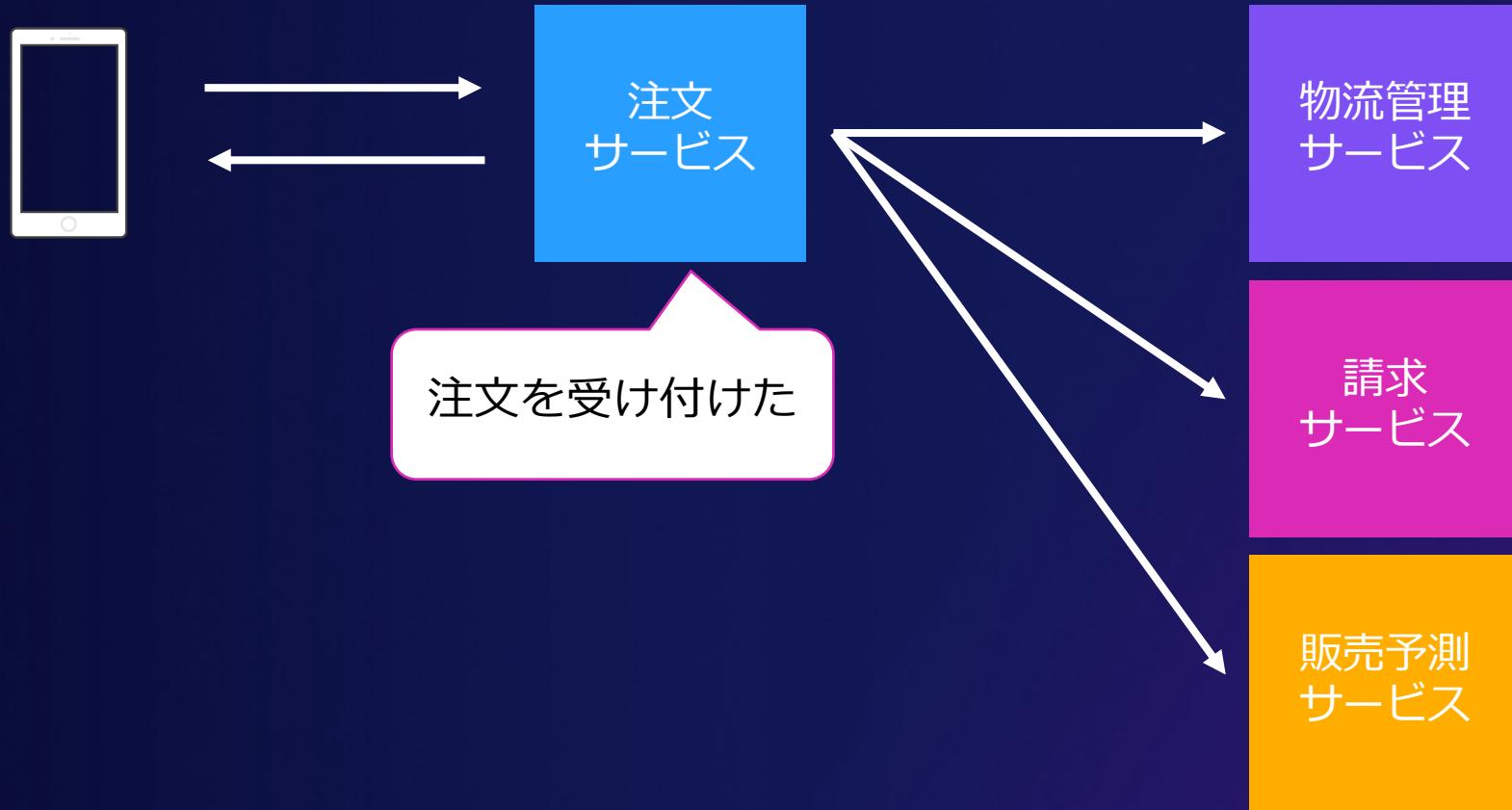
イベントバス



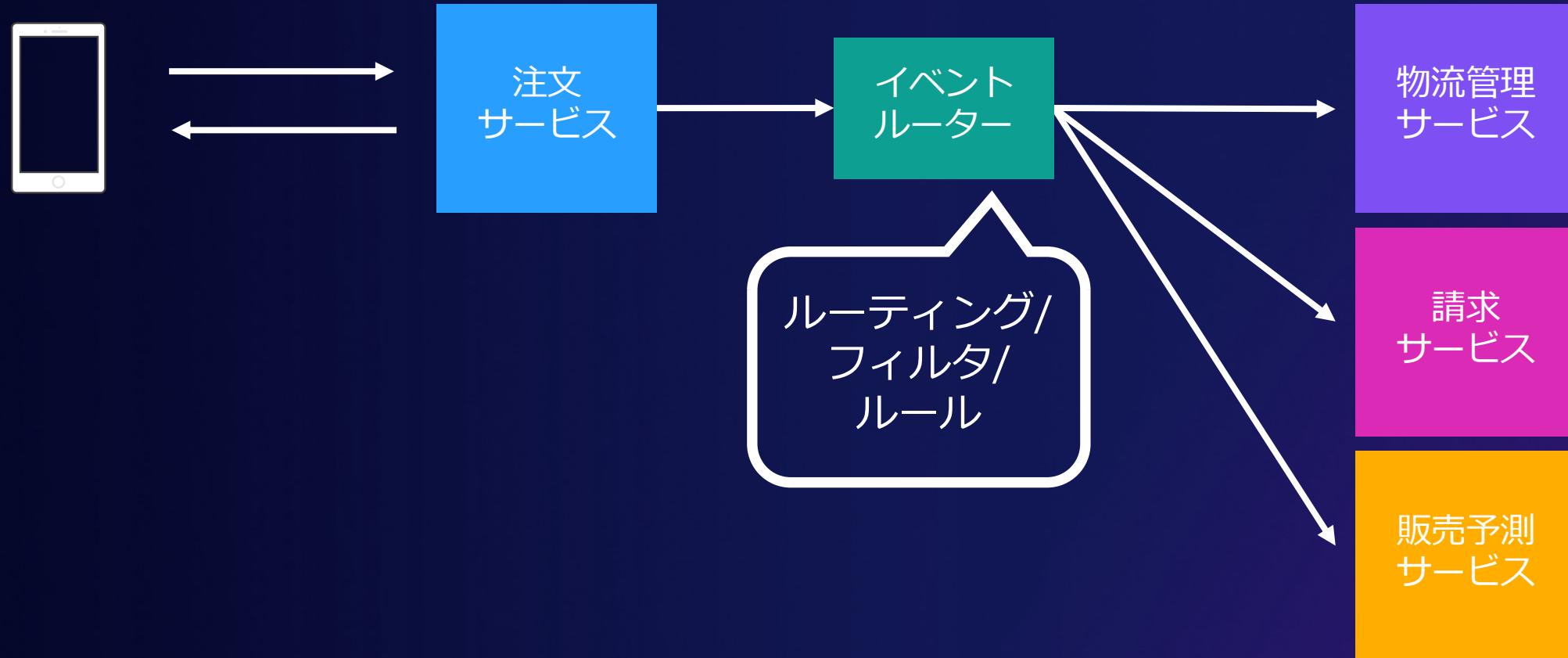
Amazon
EventBridge

- AWSサービスからのイベントをイベントソースとしたい
- サービス間の統合を簡単に実現したい

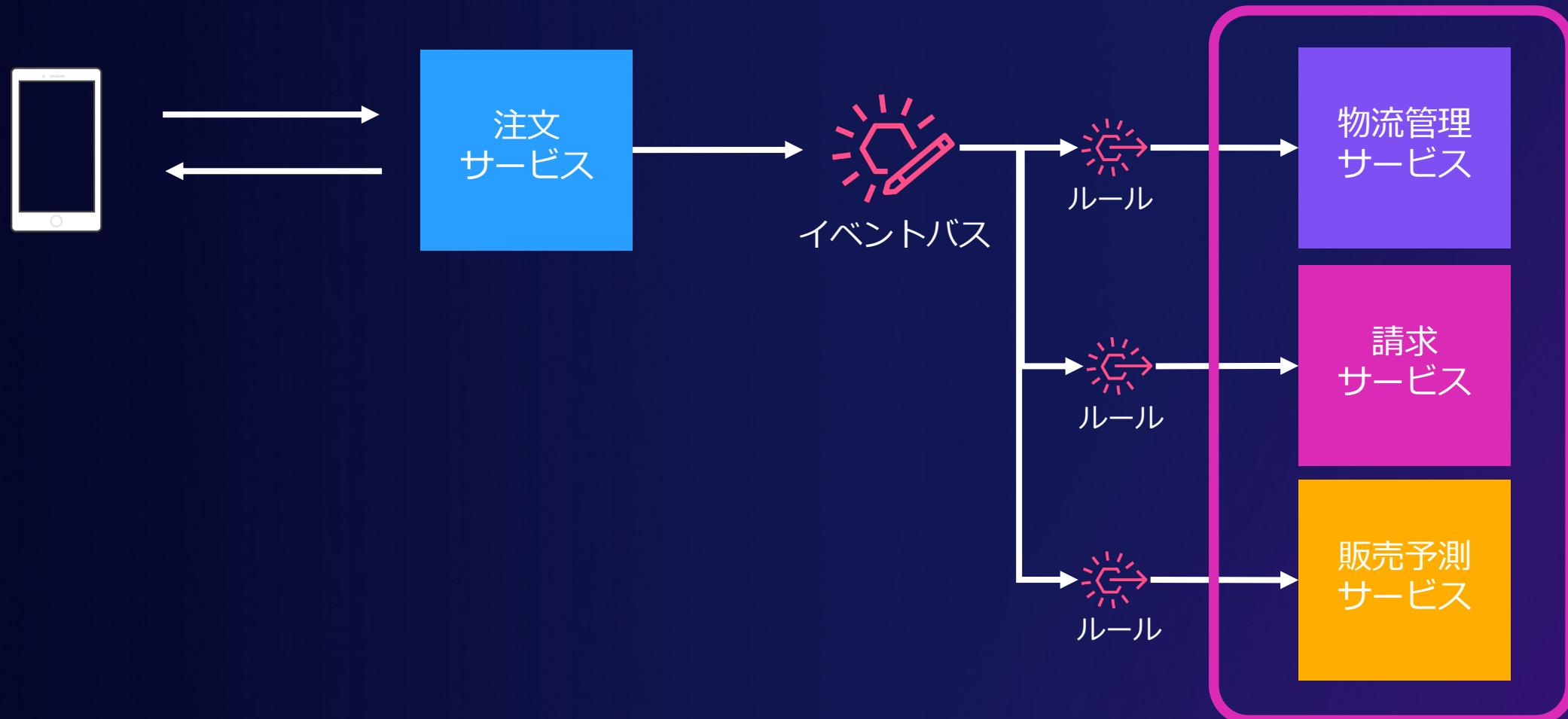
Eコマース注文システムの例



Eコマース注文システムの例

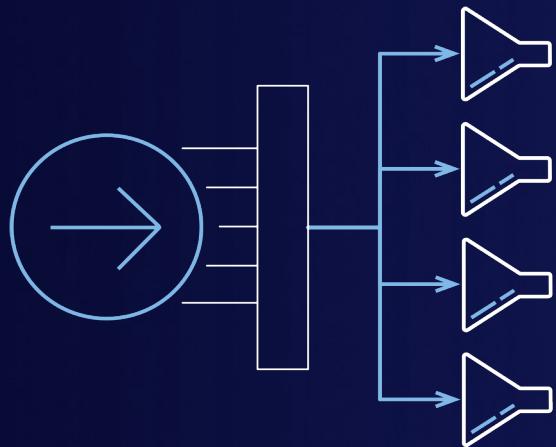


Eコマース注文システムの例



イベント駆動での連携要素

イベントルーター



プロデューサーと
コンシューマーを
お互いに抽象化

イベントストア



サービスが処理できるよ
うになるまで
バッファリング

イベントストア



サービスが処理するまでメッセージを
バッファリング

AWSでのイベントストア

AWS サービス

特徴

キュー



Amazon
Simple Queue Service

- シンプルでフレキシブルなメッセージキュー
- StandardとFIFOの2つのタイプ

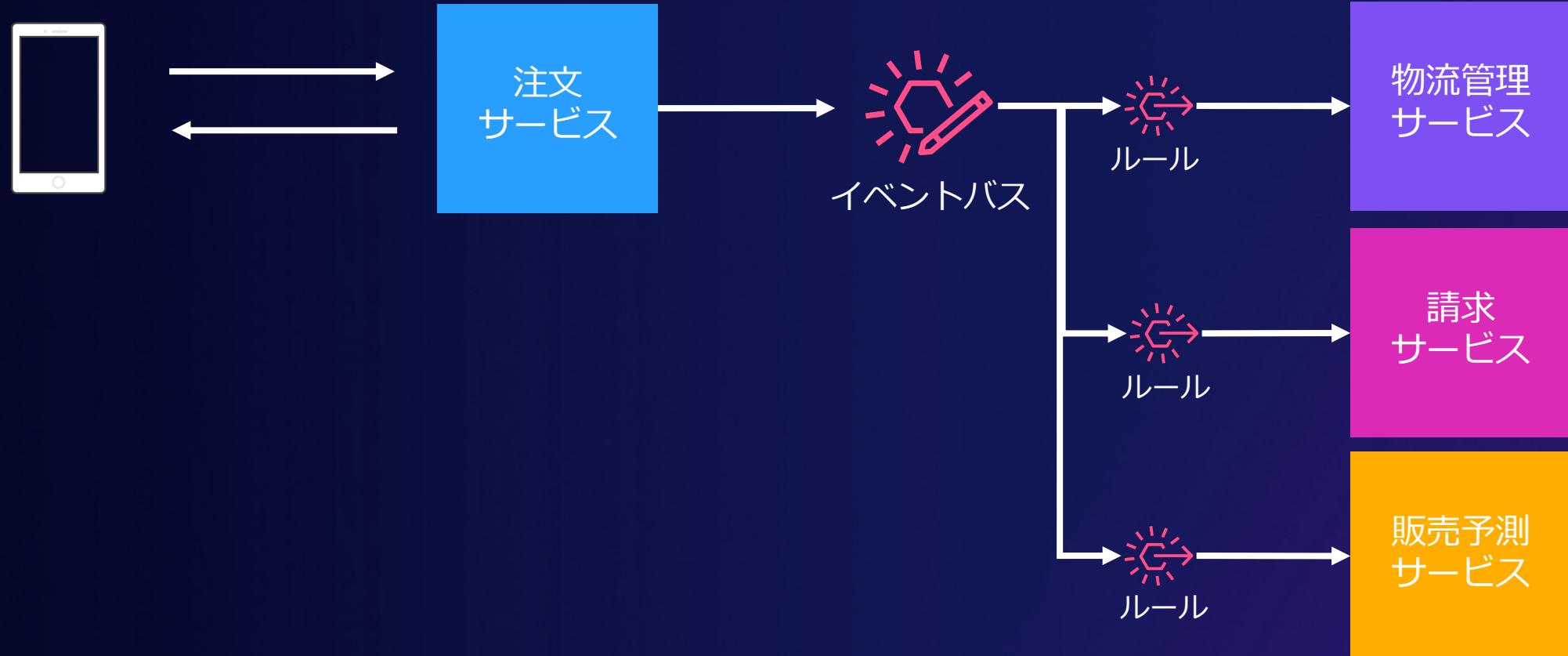
ストリーム



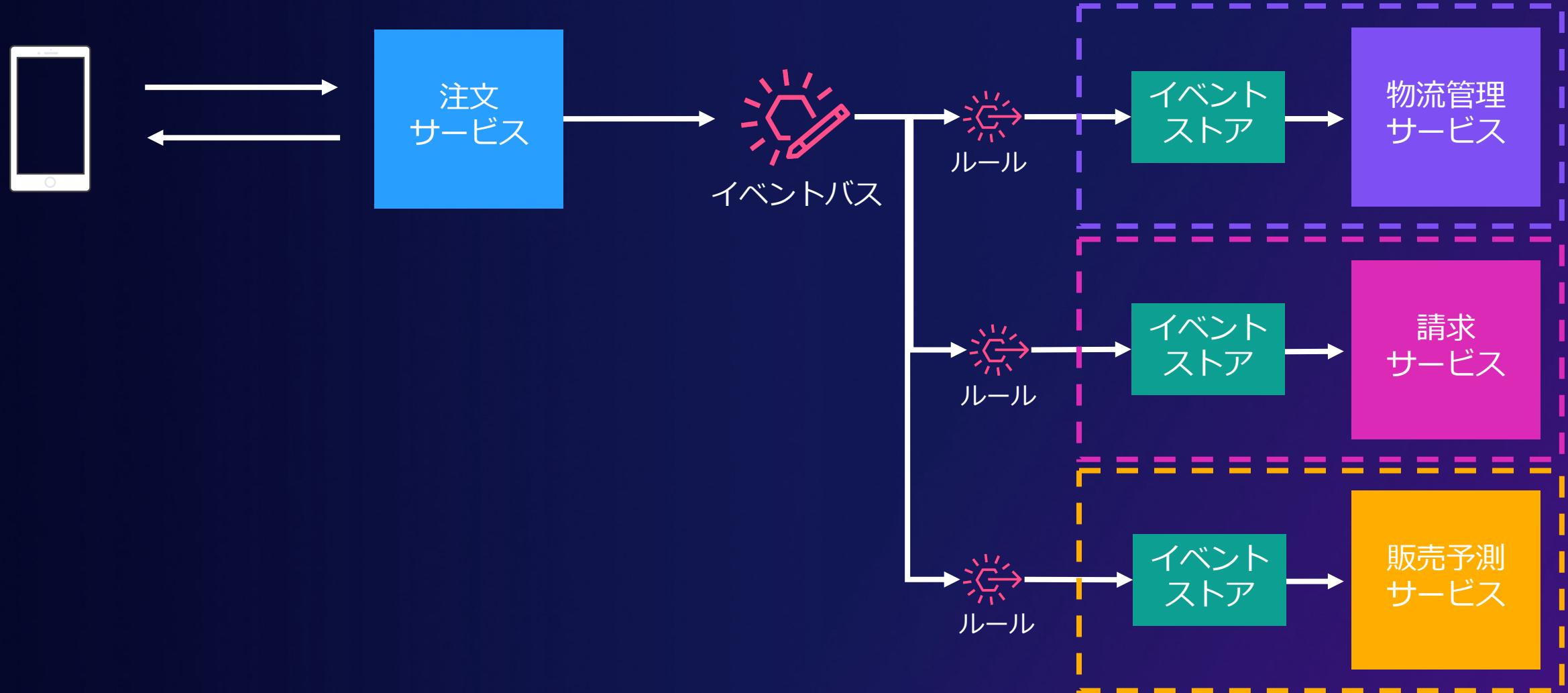
Amazon Kinesis
Data Streams

- シャード単位でのストリーム
- 大量のデータのリアルタイム処理

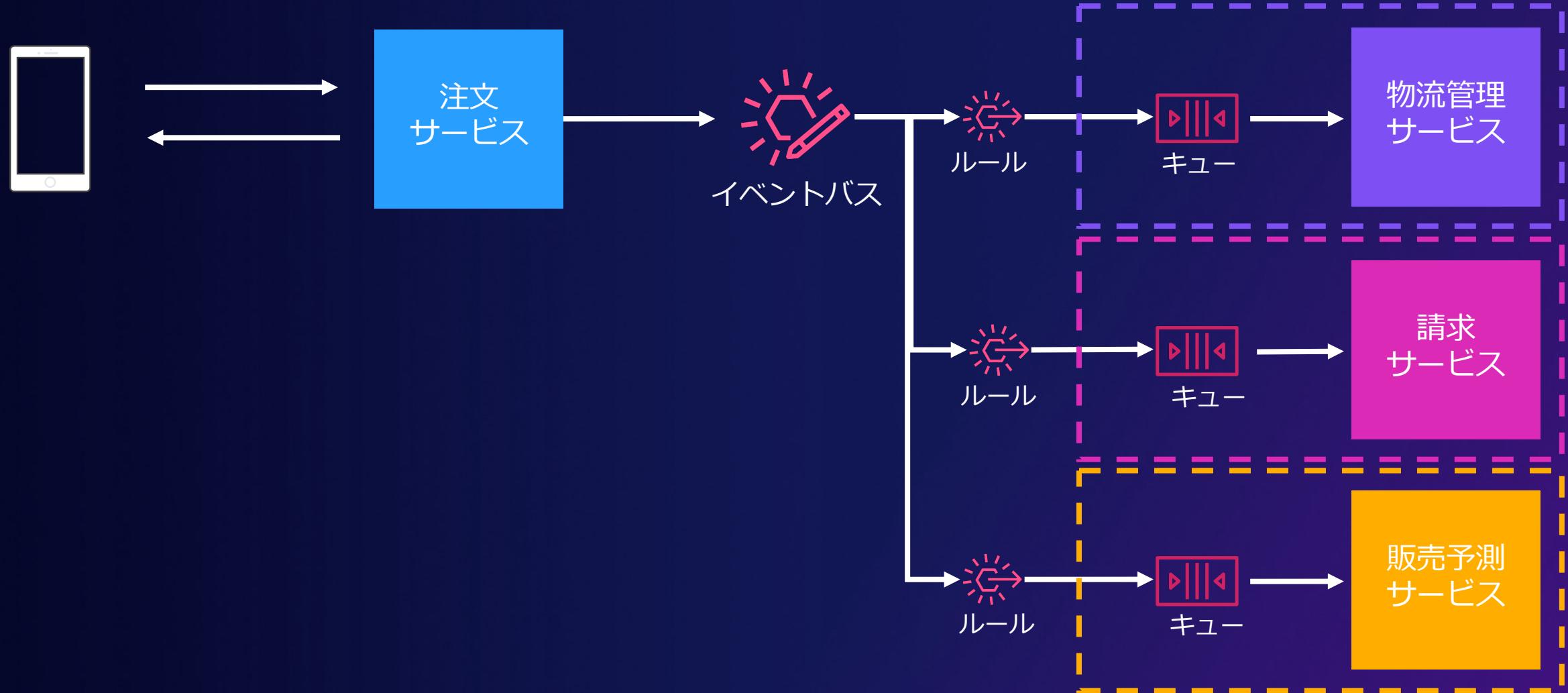
Eコマース注文システムの例



Eコマース注文システムの例



Eコマース注文システムの例



AWSでのイベントストアの使い分け

AWS サービス

ユースケース例

キュー



Amazon
Simple Queue Service

- ・ メッセージ毎に処理を実施するケース
- ・ 比較的軽量なメッセージを扱うケース

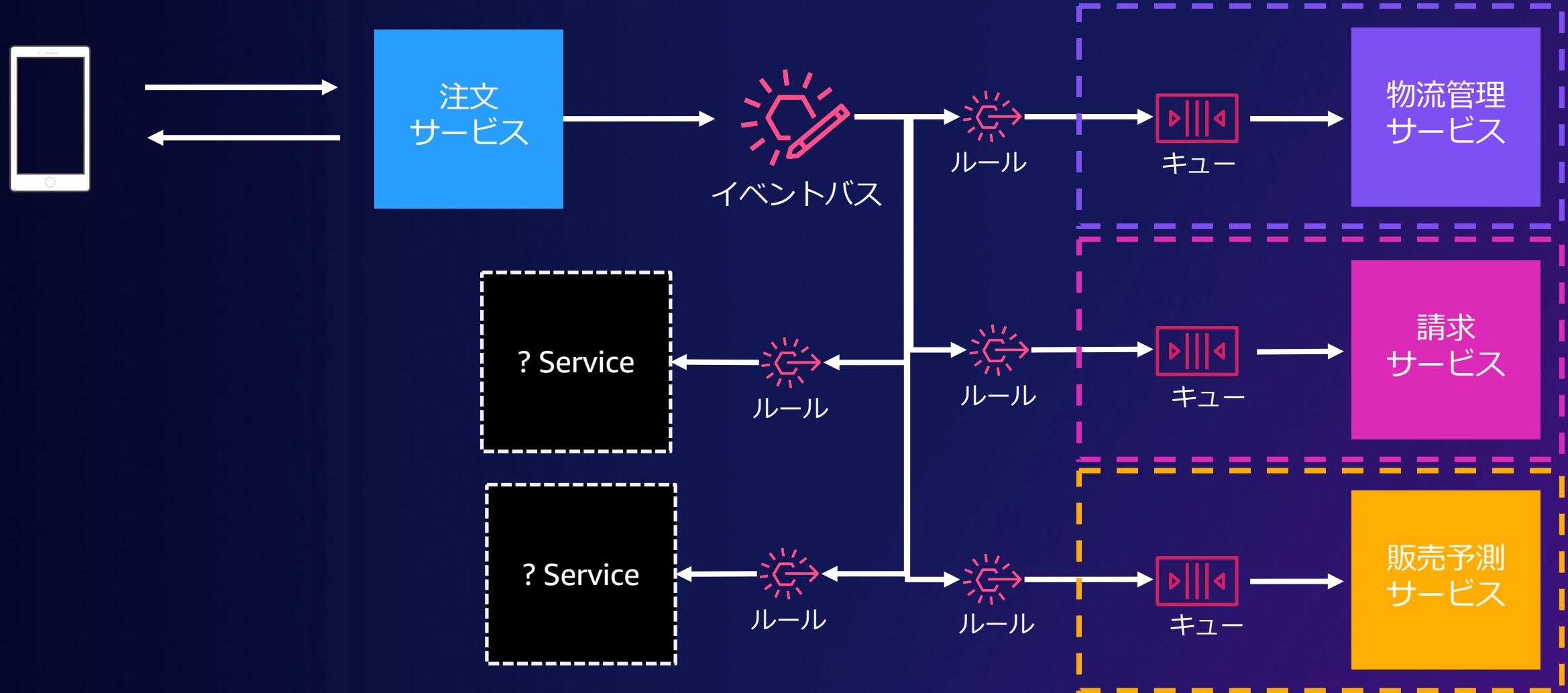
ストリーム



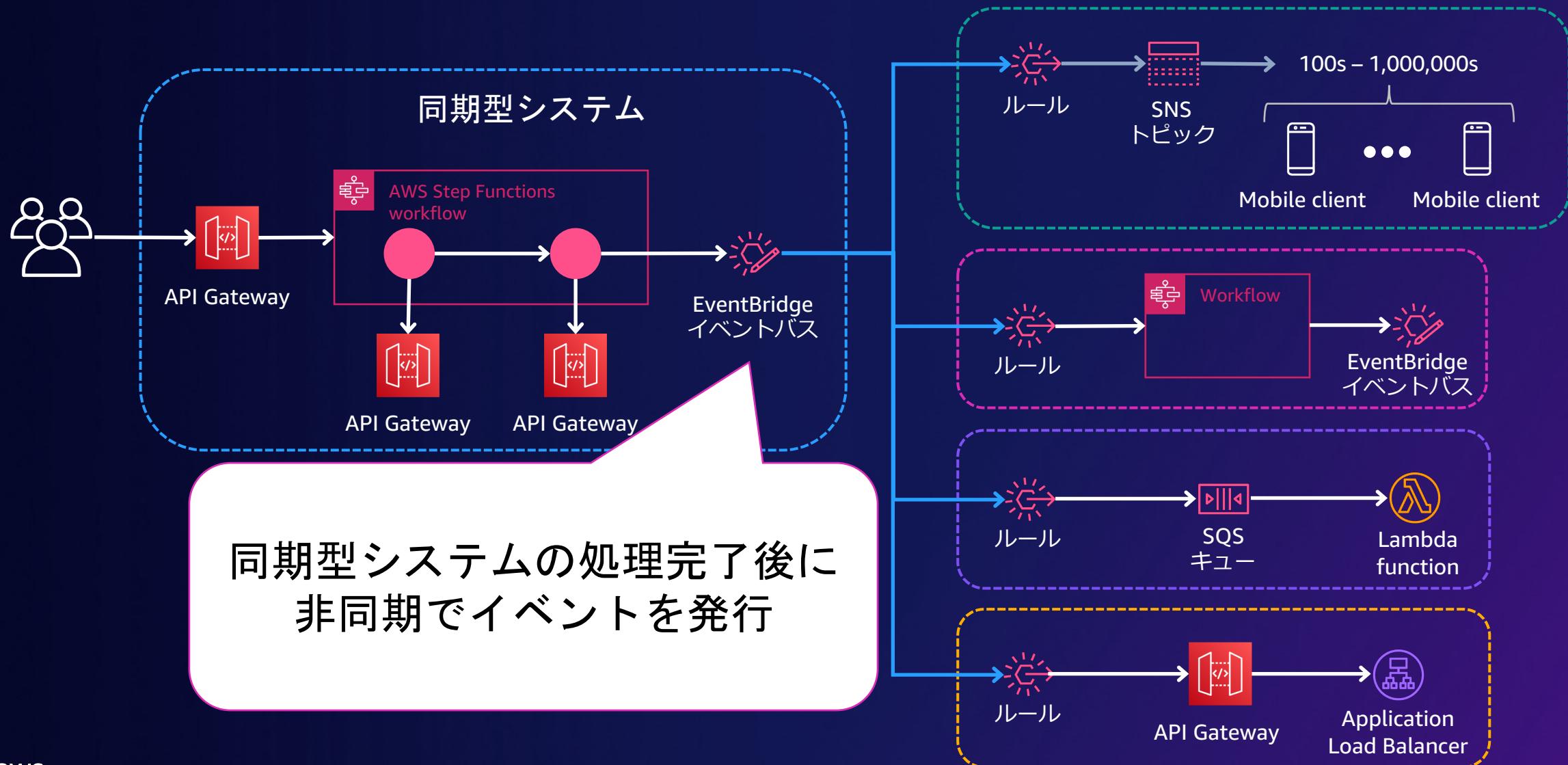
Amazon Kinesis
Data Streams

- ・ 多くのメッセージに対して複雑な処理を実施
- ・ 大量のデータ処理が必要なケース

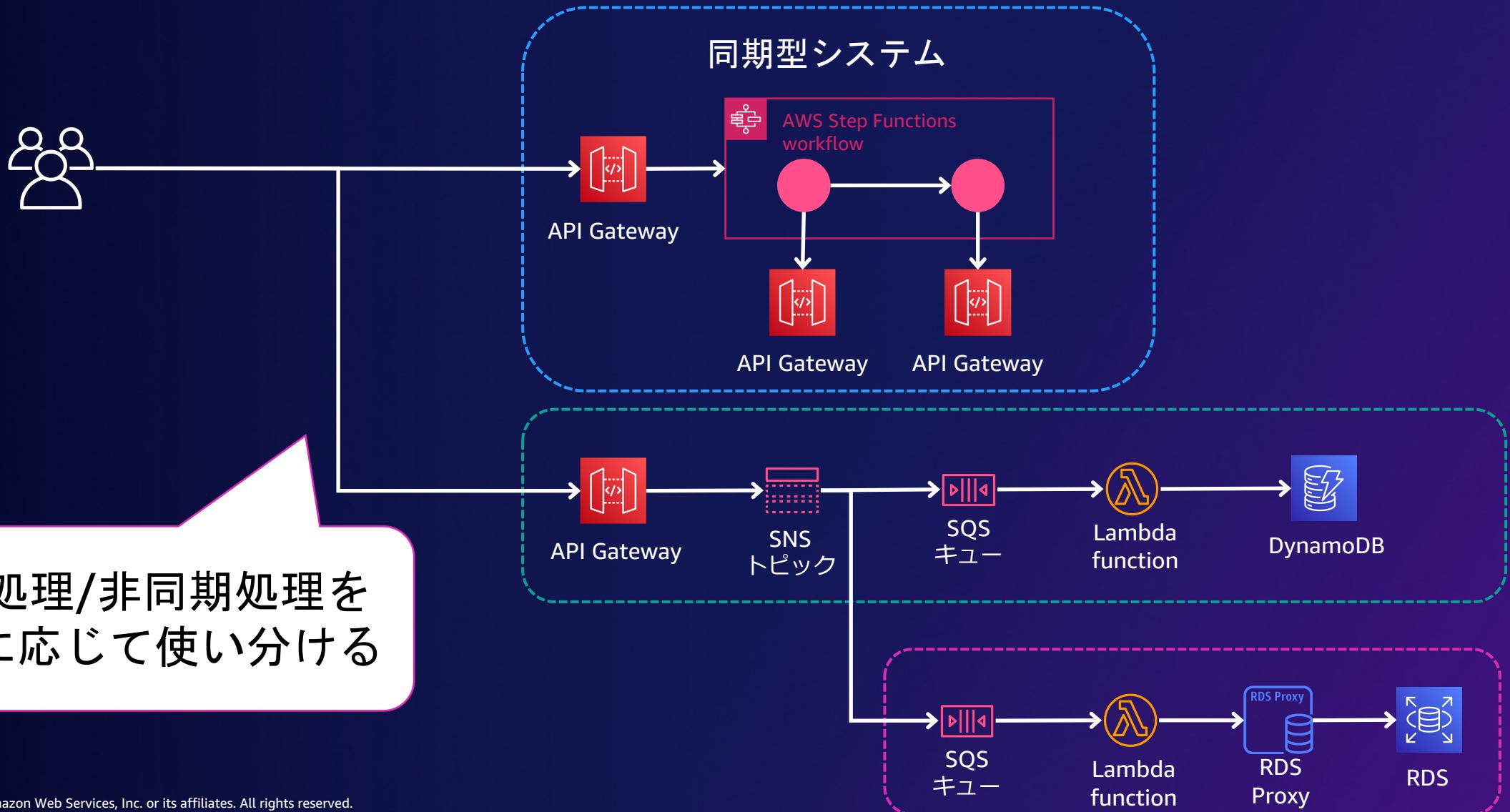
Eコマース注文システムの例



同期型システムとイベント駆動の組み合わせ



同期型システムとイベント駆動の組み合わせ



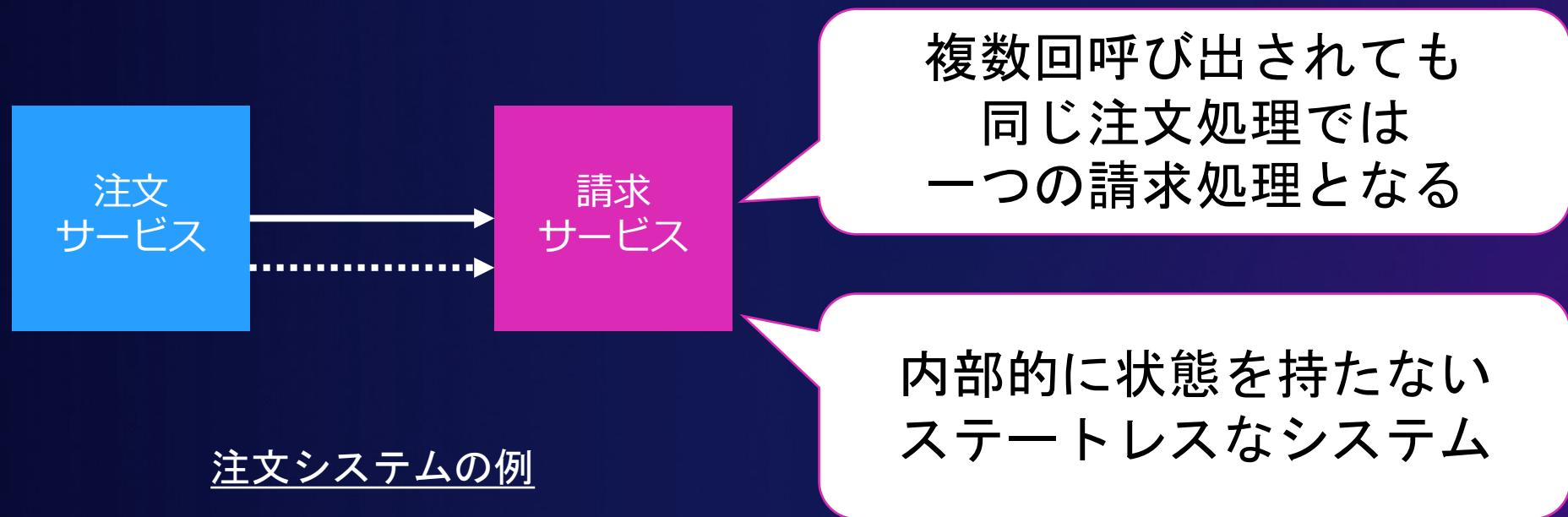
イベント駆動アーキテクチャ設計時の 考慮ポイント

イベント駆動設計時の考慮ポイント

- ・ダウンストリームのシステムに幂等性があるか
- ・重複排除を必要とするか
- ・順序保証が必要とするか

幂等性

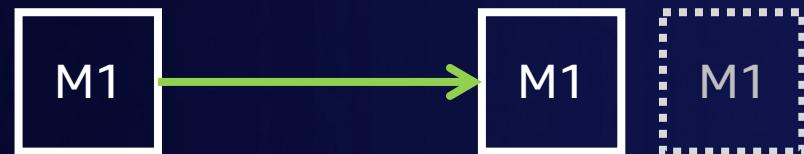
ある操作を1回もしくは複数回実行しても、同じ結果となる性質。



イベントの配信方式

At-least Once

少なくとも一回配信



イベントが複数回される場合がある為、幂等性が重要

[対象サービス]
Amazon EventBridge
Amazon SQS Standard queue
Amazon SNS Standard topic

Exactly Once

正確に一回のみ配信*



AWSサービスに識別子を渡すことによって重複を排除

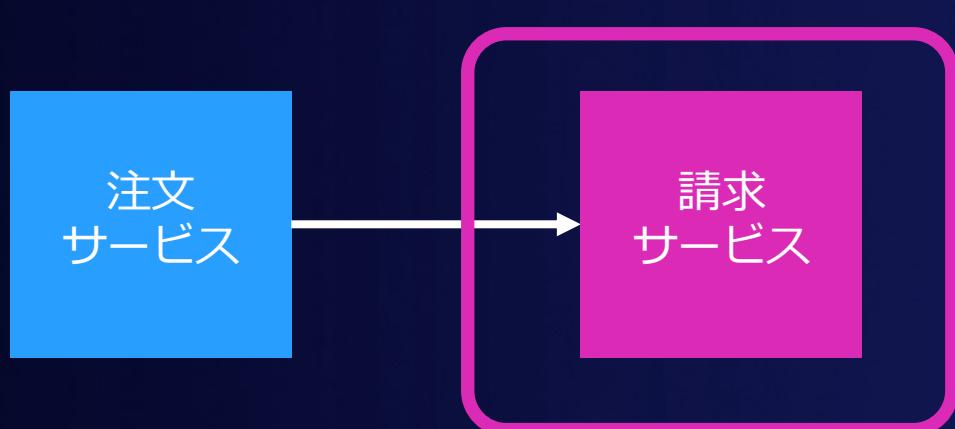
[対象サービス]
Amazon Kinesis
Amazon SQS FIFO queue
Amazon SNS FIFO topic

* Lambdaでリトライ設定している場合は、At-least once

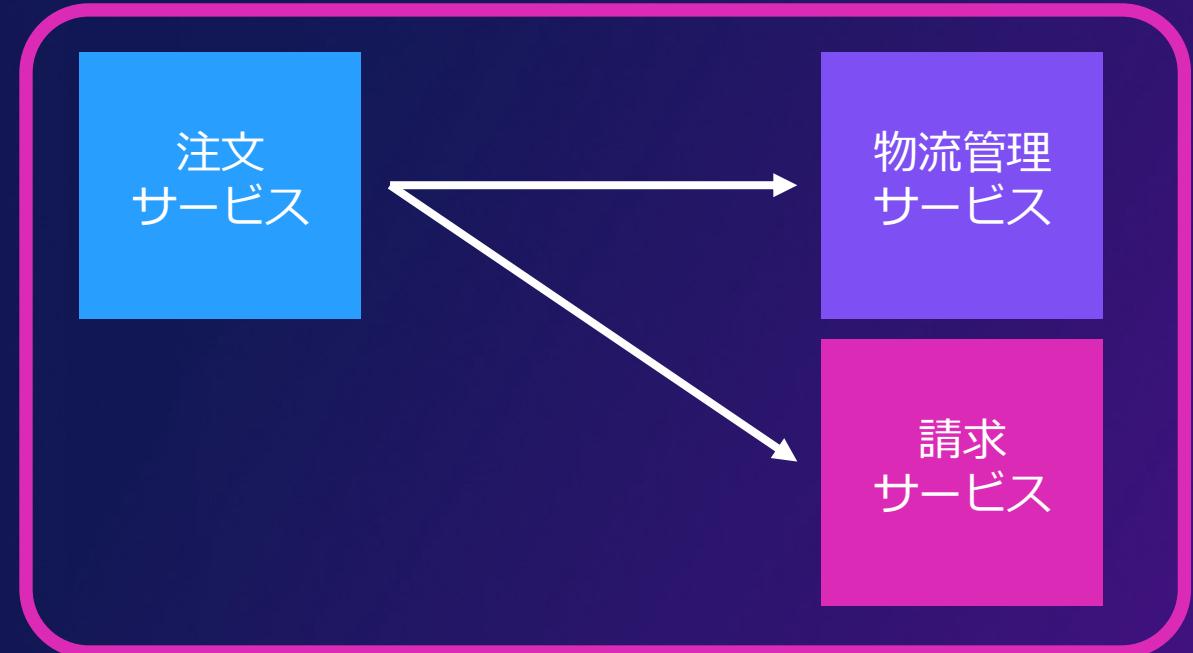
幂等性のスコープの検討

業務的にトランザクションを意識する場合はより大きなスコープで
幂等性の確保をする必要がある。

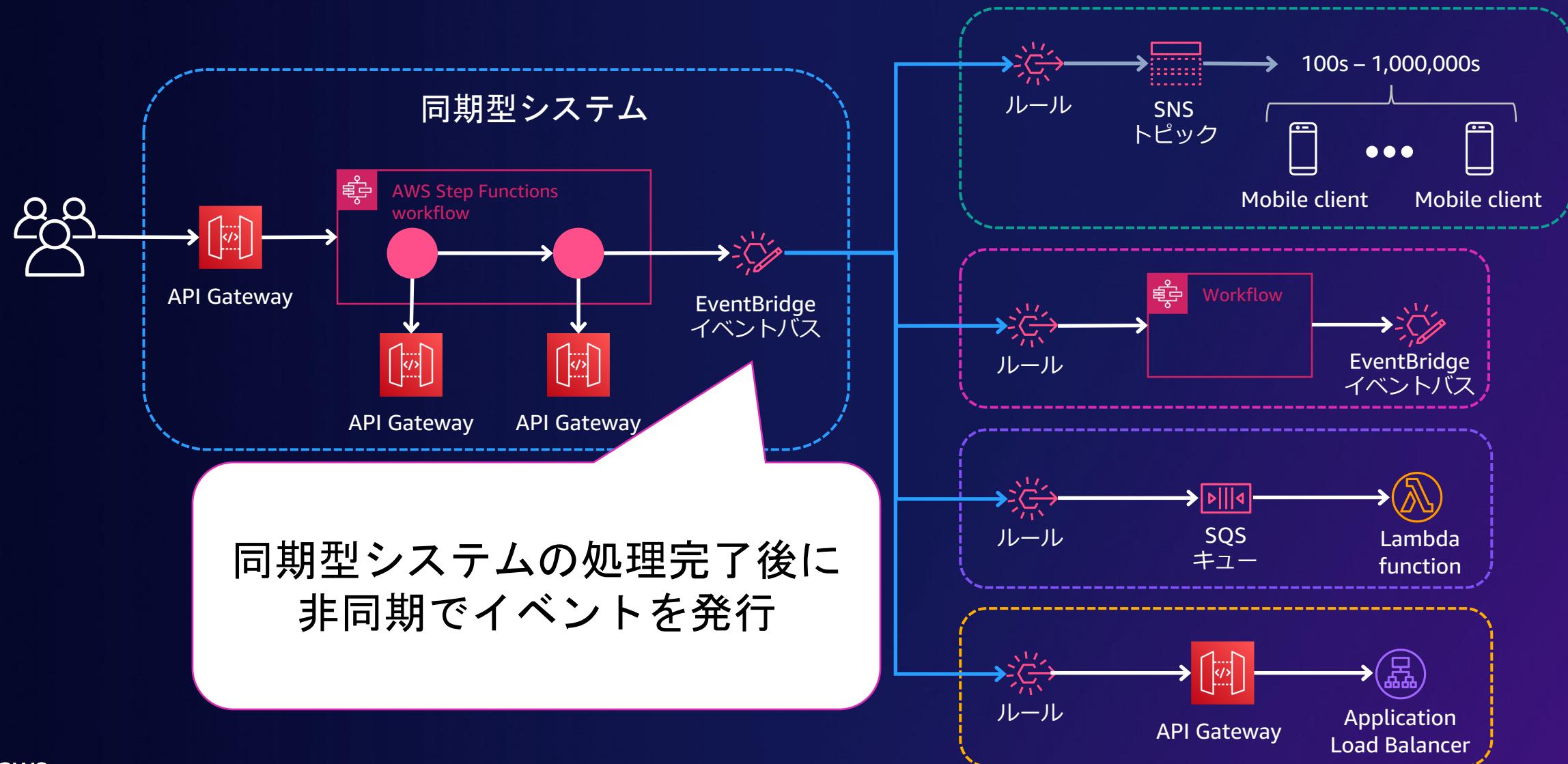
単一サービスでの幂等性？



複数サービスでの幂等性？



同期型システムとイベント駆動の組み合わせ



さらに幂等性を知るには

builders.flash にて幂等性やサーバーレスでの実装方法を公開中

メッセージやデータストア以外にも分散トランザクション等も解説



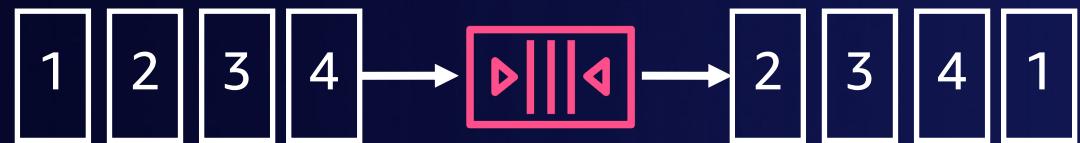
<https://aws.amazon.com/jp/builders-flash/202104/serverless-idempotency/>



<https://aws.amazon.com/jp/builders-flash/202106/serverless-idempotency-implementation/>

イベントの順序保証

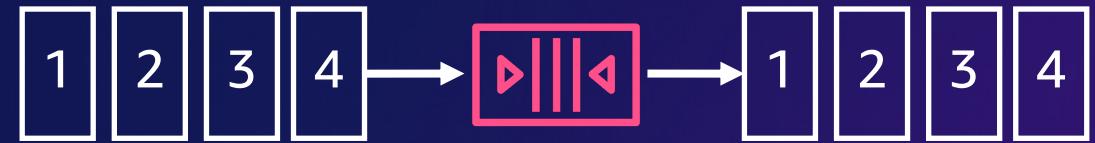
順序保証なし



順序保証なしで配信。
SQS Standard queue, SNS Standard topicは
ベストエフォートでの配信

[対象サービス]
Amazon EventBridge
Amazon SQS Standard queue
Amazon SNS Standard topic

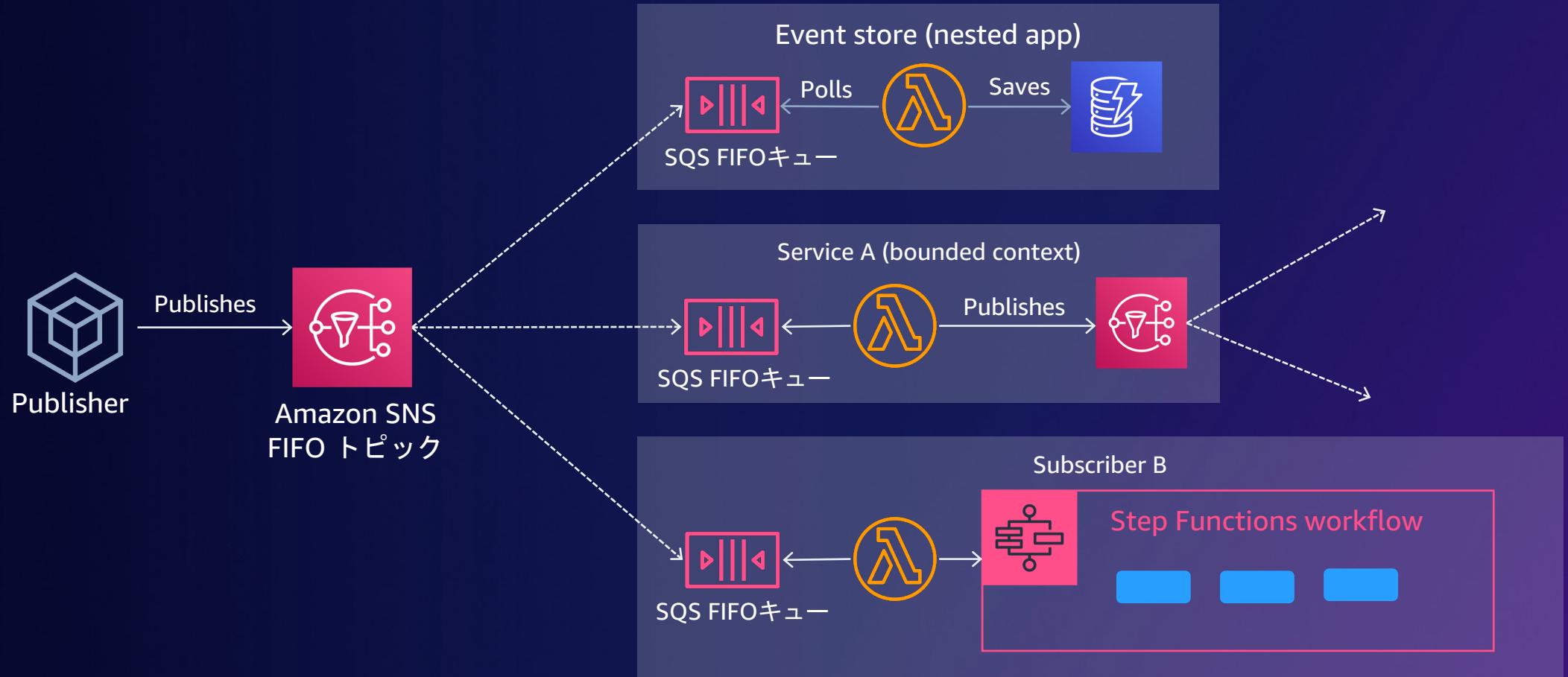
順序保証あり



パーティションやメッセージグループ等の中で順番に配信

[対象サービス]
Amazon Kinesis Data Streams
Amazon SQS FIFO queue
Amazon SNS FIFO topic

組み合わせたFIFO Fanout アーキテクチャ



その他のイベント駆動設計時の考慮ポイント

- エラーへの対応をどうするか
- トランザクションが必要な処理があるか
- テストはどうするのか
- etc…



イベント駆動アーキテクチャに関する
オンデマンドウェビナーを公開中
(全8回)

<https://aws.amazon.com/jp/devax-connect-on-demand/>

まとめ

- サービスの責務や関心の分離、結合度を下げる方法としてのイベント駆動アーキテクチャ
 - イベントルーター、イベントストア、非同期イベントの連携パターン
 - 同期型のアーキテクチャとの併用もOK
- イベント駆動アーキテクチャでは幕等性が重要
 - 各サービスでの重複排除や順序保証の仕様理解も忘れずに
- サーバーレスサービスを活用することで運用性やコスト効率の高いアーキテクチャを構築することが可能

イノベーションやモダナイズに役立つソリューション

既存システムが稼働中、コンテナへの移行を検討している場合



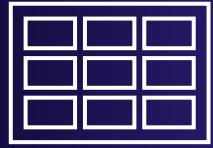
AWSのマネージドコンテナサービスにワークロードを移行し、運用の簡素化と管理オーバーヘッドの削減を図る

新しいアプリケーションや機能を構築する場合

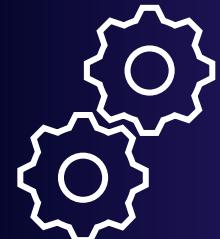


AWS Lambdaのようなサーバーレス技術の活用

開発や運用をモダナイゼーション



イベント駆動でのモジュラー
アーキテクチャパターン



サーバレスサービスの活用



アジャイル開発プロセス



Thank you!

Nobuo Okada (Paul)

 @bulbulpaul



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.