JAPAN | JUNE 20, 2024

aws SUMMIT

DOL-8

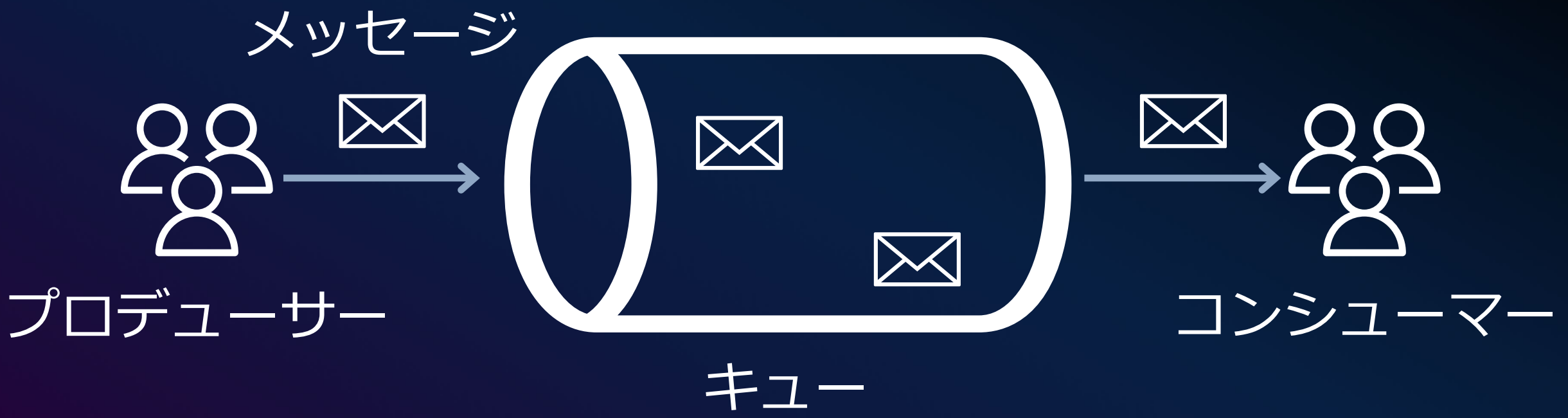
クラウドアプリケーションに欠かせない キュー活用のポイント

Emi Morikawa

Amazon Web Services
Cloud Support Engineer

メッセージキューの基本

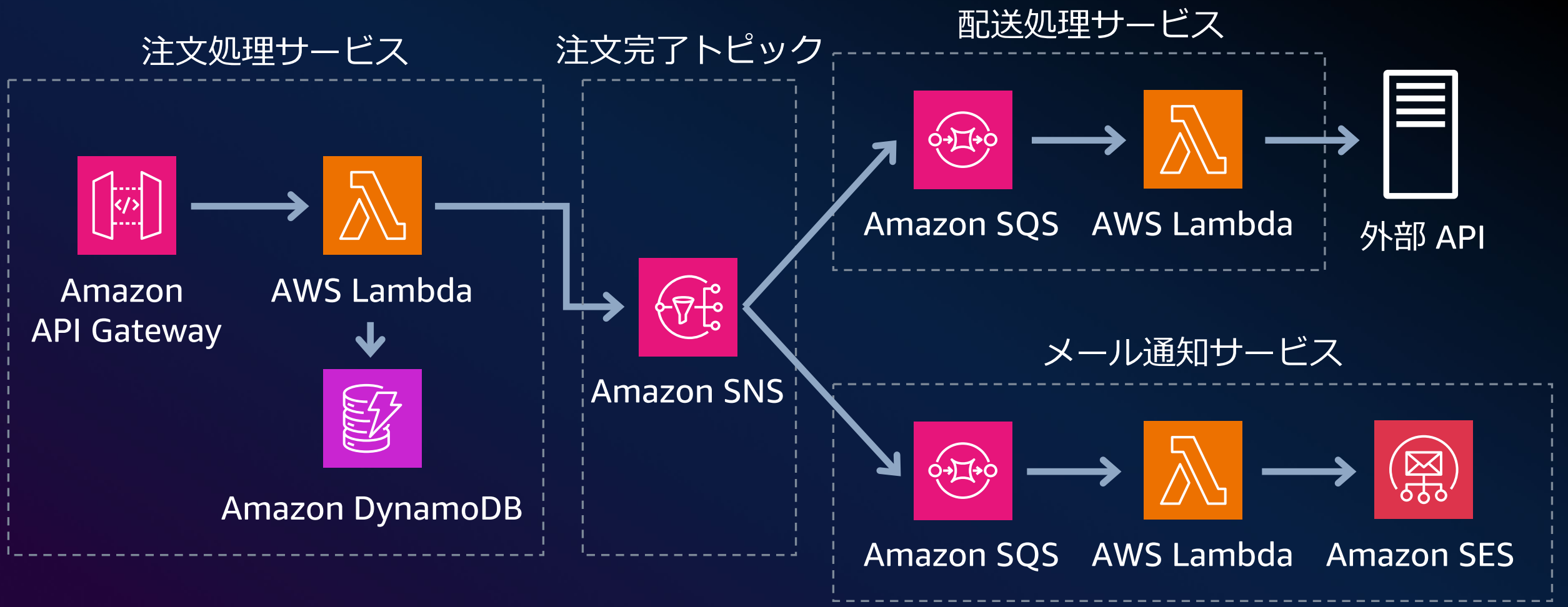
システムの通信/オペレーションの処理を非同期にする



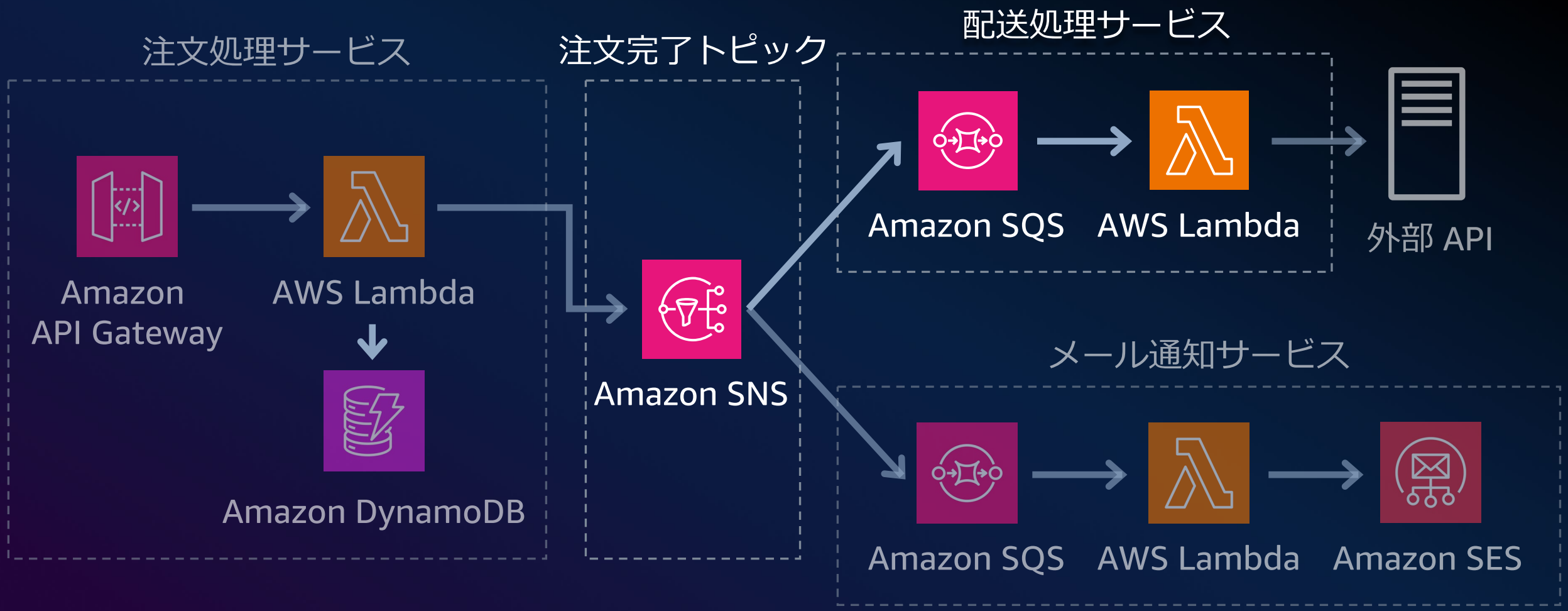
メッセージキューを使うメリット

- **スケーラビリティの向上**
  - コンポーネントごとに独立してスケーリング可能
- **可用性や耐障害性の向上**
  - コンポーネントの障害/変更の影響が他に伝播しにくくなる
- **パフォーマンスの改善**
  - 一時的な過負荷に強くなる (バーストトラフィックの吸収)
  - 重いバックエンドの処理を分離可能

非同期メッセージングのアーキテクチャ例



今回見ていくスコープ



メッセージ配信の信頼性

At least once
少なくとも一回の配信

- Amazon S3 イベント通知
- Amazon SNS 標準トピック
- Amazon SQS 標準キュー

Exactly once
正確に一回だけの配信

- Amazon SNS FIFO トピック
- Amazon SQS FIFO キュー

よくあるお問い合わせ

SQS キューでメッセージが重複して配信された

→ At least once の性質を持つ標準キューを使用していた

Amazon S3 イベント通知で Lambda 関数が重複して起動した

→ Amazon S3 イベント通知は At least once で配信される

Amazon EventBridge で 1つのイベントでルールが複数回トリガーされた

→ Amazon EventBridge も At least once の性質を持つ

At least once か Exactly once か

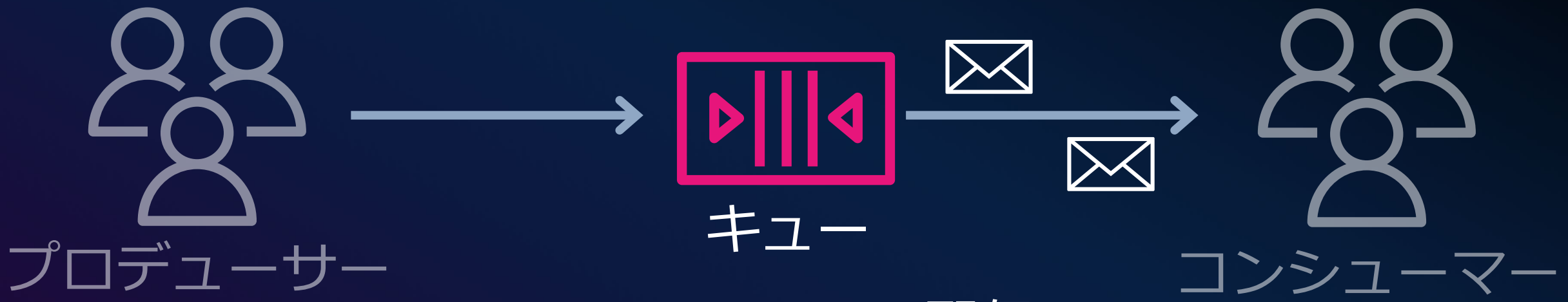
At least once
少なくとも一回の配信

Exactly once
正確に一回だけの配信

?

メッセージが重複するシナリオ

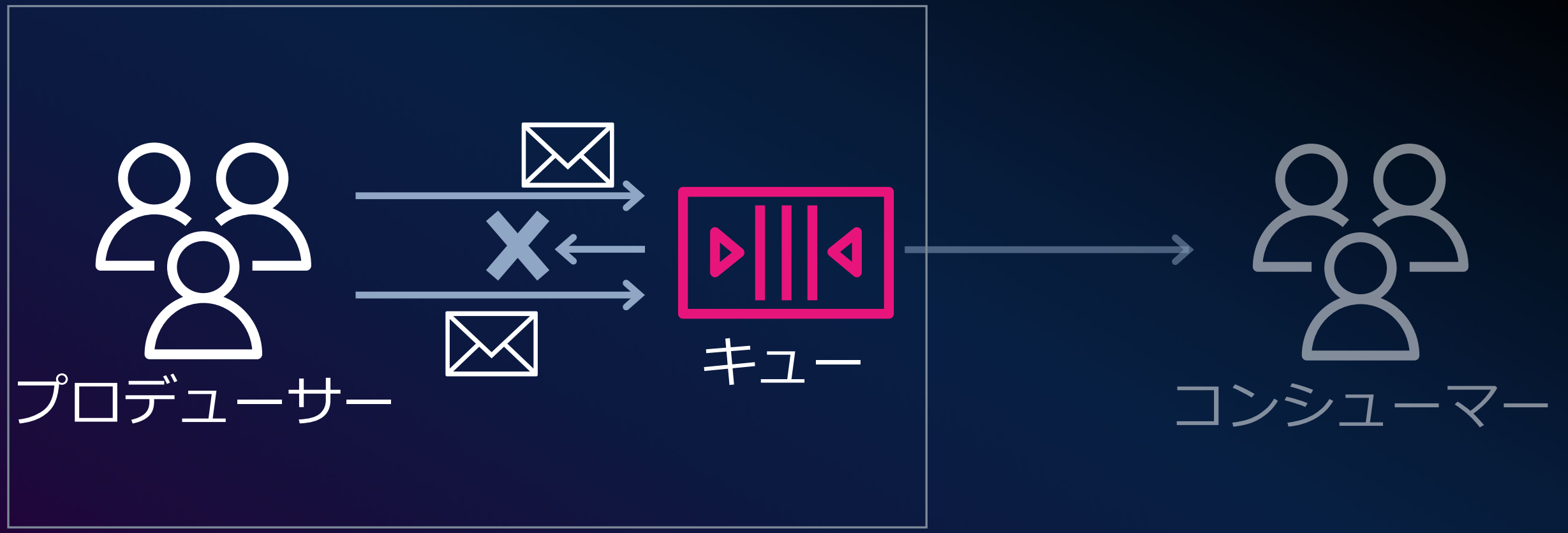
キューの配信方法



At least once の配信

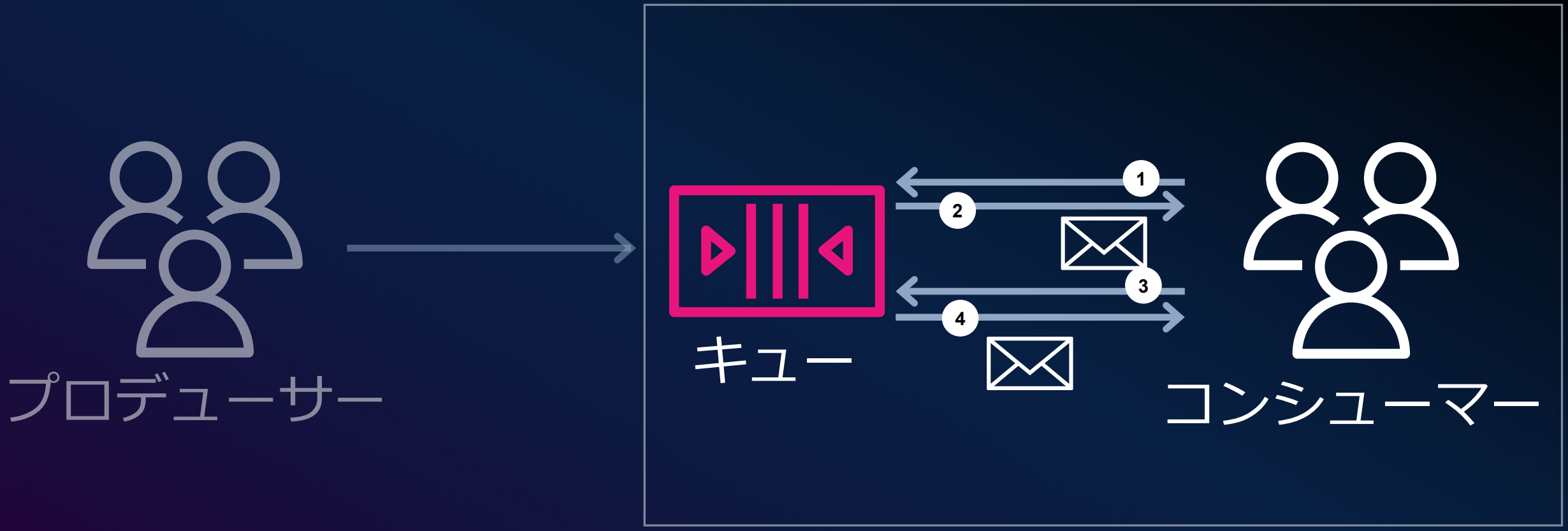
メッセージが重複するシナリオ

プロデューサー側の挙動



メッセージが重複するシナリオ

コンシューマー側の挙動



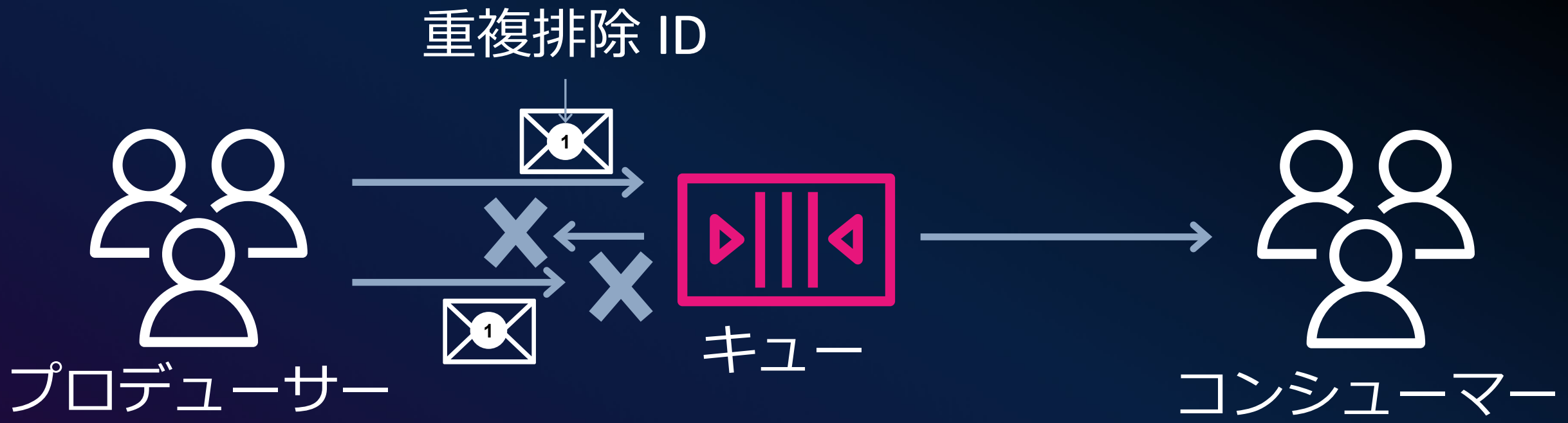
メッセージが重複するシナリオ

- キュー自体のメッセージ重複配信
- プロデューサー側の重複配信
- コンシューマー側の重複受信

Exactly once のキューで対応できること

- キュー自体のメッセージ重複配信
- プロデューサー側の重複配信
- コンシューマー側の重複受信

Exactly once を実現する仕組み



重複性保証とスループットはトレードオフ

例えば東京リージョンの Amazon SQS だと...

	標準キュー	FIFO キュー
配信方式	At least once	Exactly once
メッセージのスループット	ほぼ無制限の API コール	高スループットモードと バッチ処理の使用で 最大 90,000 メッセージ/秒
コスト (※1, ※2)	USD 0.40	USD 0.50

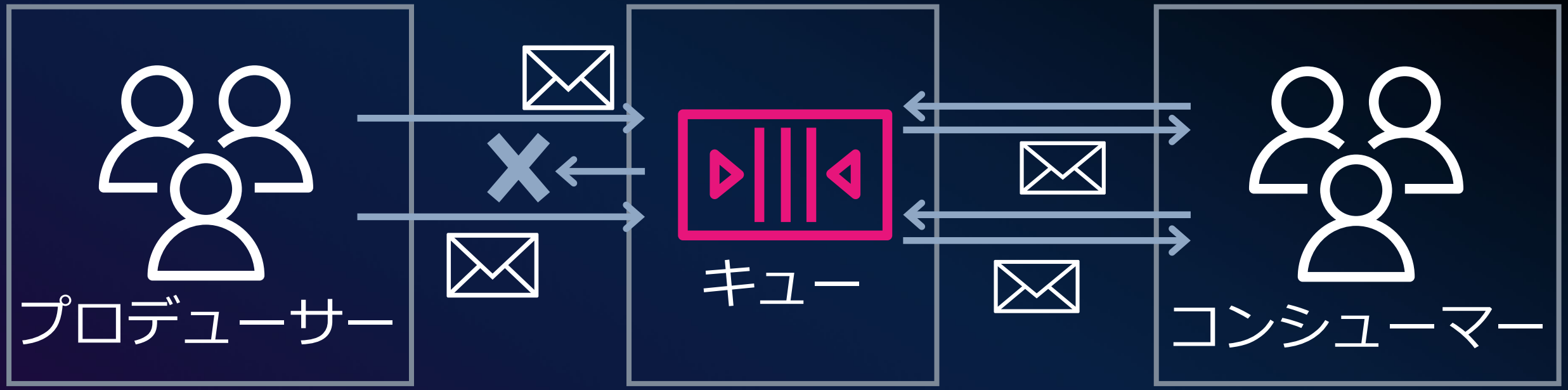
※1 100 万から 1,000 億リクエスト/月
※2 毎月 100 万リクエストは無料枠

Exactly once なキューでの留意点

- スループットやコスト
- コンシューマー側の重複受信

例えば...
Lambda イベントソースマッピングはイベントを少なくとも 1 回処理

At least once を使いこなす

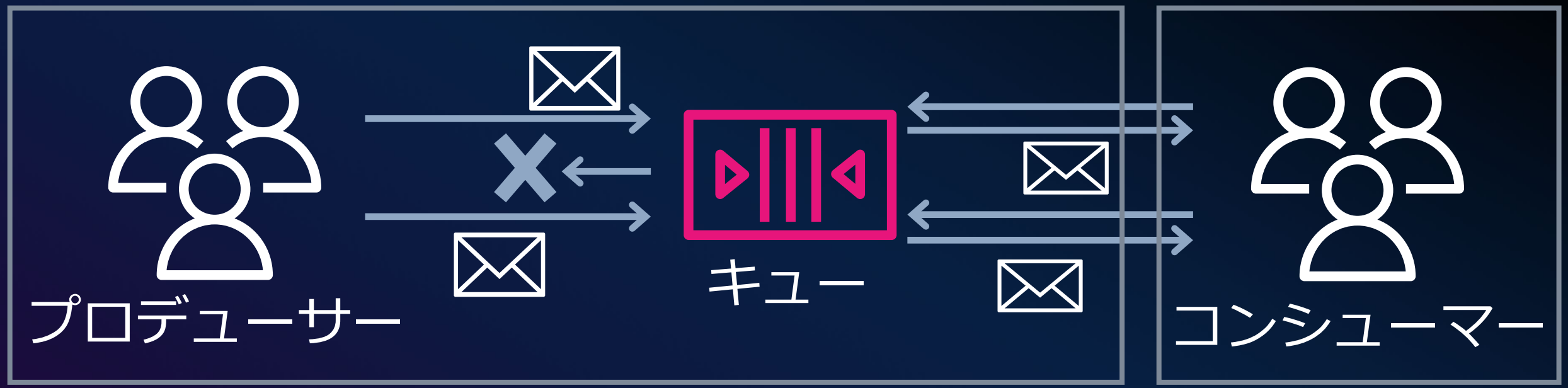


各コンポーネントで Exactly once を考慮していくのは大変

→ At least once でアプリケーションを設計できないか？

べき等性とは

ある操作を 1 回行っても複数回行っても結果が同じ



At least once な配信

メッセージが重複する可能性

べき等な処理

処理がべき等でないと...

- **データの不整合**

  e.g. 商品購入時の在庫減算処理の重複

- **重複した API リクエストの発生**

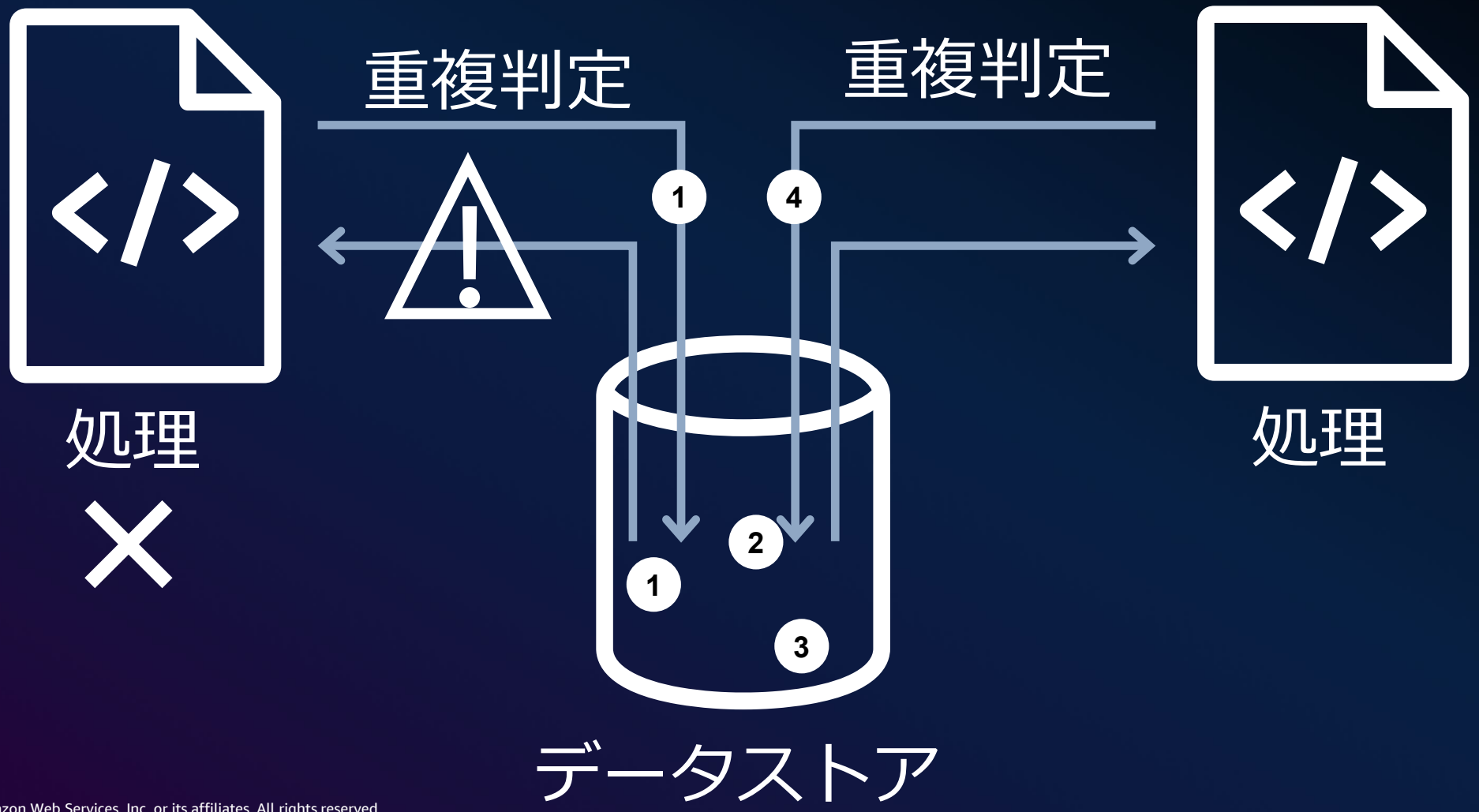
  e.g. 送金リクエストの重複

- **無駄なリソース消費の発生**

  e.g. 計算量の多いリクエストの重複

典型的なべき等性の実装方法

入力イベントの一意な属性 (e.g. リクエスト ID) を使用する



AWS Lambda Powertools

AWS がオープンソースで開発している
Lambda 関数用のユーティリティーライブラリ

- トレーシングやロギングなど様々な便利なライブラリを提供
- Python, Java, TypeScript, .NET に対応
- 安全なリトライを実行するため Lambda 関数で
  べき等性を可能にするソリューションを提供

Powertools for AWS Lambda (Python)

デモ – EC サイトの配送処理

- 注文 ID
- ユーザー ID
- 商品 ID

```
{
  "orderID": "xxxx-xxxx-xxxx-xxxx-xxxxx",
  "userID": "user-1",
  "productID": "product-A"
}
```

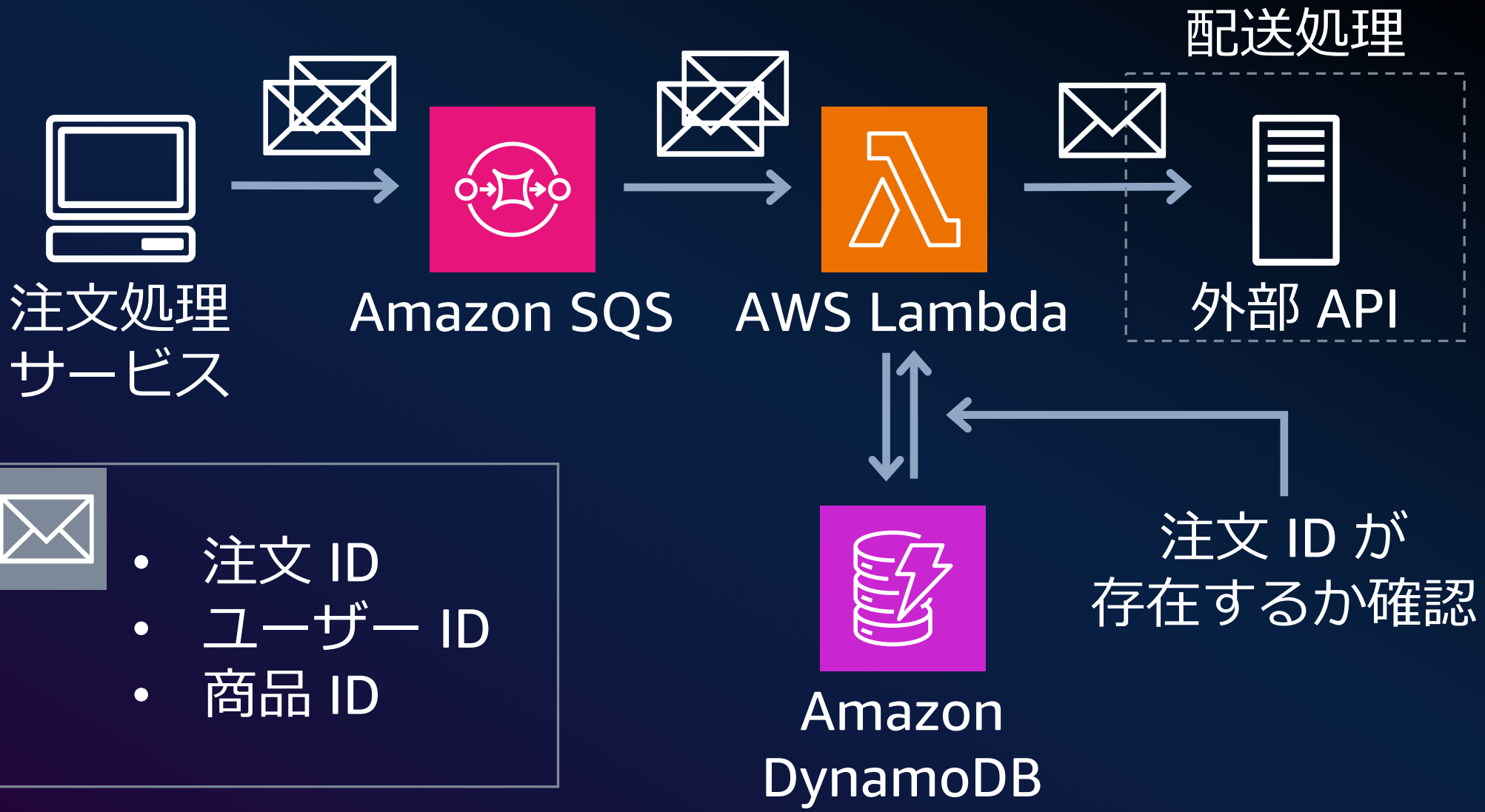
デモ – EC サイトの配送処理

重複して実行されてしまう

- 注文 ID
- ユーザー ID
- 商品 ID

```
{
  "orderID": "xxxx-xxxx-xxxx-xxxx-xxxxx",
  "userID": "user-1",
  "productID": "product-A"
}
```

デモ – EC サイトの配送処理



デモ – Lambda 関数の実装

べき等を考慮しない

```
# 外部 API を実行
request_post = request.Request(API_URL, method="POST",data=json.dumps(data).encode())
request.urlopen(request_post)
```

べき等を考慮する

```
# 注文 ID がすでにテーブルにあるか確認する
response = table.get_item( Key={"orderID" : orderID} )
# 注文 ID がテーブルになければ、処理を進める
if 'Item' not in response:
    # 外部 API を実行
    request_post = request.Request(API_URL, method="POST", data=json.dumps(data).encode())
    request.urlopen(request_post)
    # テーブルに情報を格納
    table.put_item( Item=data )
```

まとめ

- At least once と Exactly once の特性を理解することが重要
- Exactly once の難しさ
  - コンシューマー側の挙動の考慮
  - スループットやコストの考慮
- At least once を使いこなす
  - コンシューマー側でべき等性を持たせる
  - データの一貫性を保ちつつ、スケーラビリティと耐障害性の実現

Thank you!

Emi Morikawa