このコンテンツは公開から3年以上経過しており内容が古い可能性があります
最新情報についてはサービス別資料もしくはサービスのドキュメントをご確認ください

[AWS Black Belt Online Seminar] Amazon Simple Queue Service

サービスカットシリーズ

Professional Services
Consultant 堀場 隆文
2019/7/17

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

自己紹介

名前： 堀場 隆文（ほりば たかふみ)
所属： プロフェッショナルサービス本部
職種： コンサルタント
業務： 技術的な課題解決を中心にお客様をご支援

好きなAWSサービス：

AWS Lambda

- Security - Specialty
- Solutions Architect - Professional
- DevOps Engineer - Professional
- Developer - Associate
- Solutions Architect - Associate
- SysOps Administrator - Associate

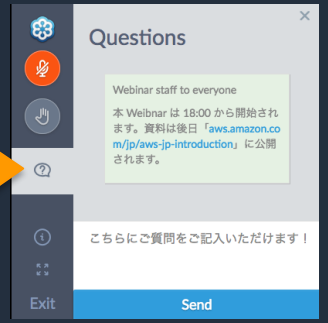
AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾン ウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- 今後のロードマップに関するご質問は
  お答えできませんのでご了承下さい

① 吹き出しをクリック
② 質問を入力
③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では2019年7月17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.

当セミナーのゴールと想定視聴者

ゴール	Amazon Simple Queue Service(Amazon SQS)の特徴と活用例を学び今後のシステム構築で利用できるようになる
想定視聴者	• キューを利用したシステム構築が未経験な方 • Amazon SQSを利用したことのない方 • 最新のAmazon SQSの機能を知りたい方

本日のアジェンダ

- 柔軟性とは
- アプリケーション間のつなぎ方
- Amazon SQSの概要
- Amazon SQS機能詳細
- まとめ

アプリケーション構造や
メッセージング等の解説が中心

本日のアジェンダ

- 柔軟性とは
- アプリケーション間のつなぎ方
- Amazon SQSの概要
- Amazon SQS機能詳細
- まとめ

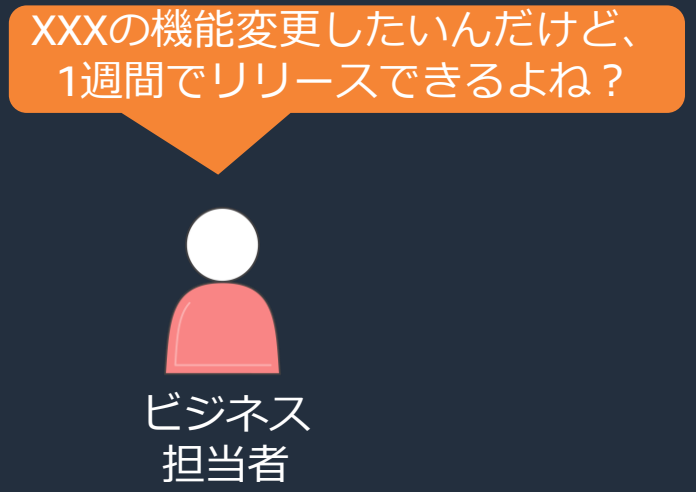
お話する内容

- 柔軟性とは？
- 柔軟性を高めるためには？

柔軟性とは？

定義	ビジネスの要求に対する**サービスの変更のしやすさ**

- 影響範囲の把握・限定はしやすいですか？
- 開発・テストはしやすいですか？
- リリースしやすいですか？
- 万が一失敗したときの影響は小さいですか？

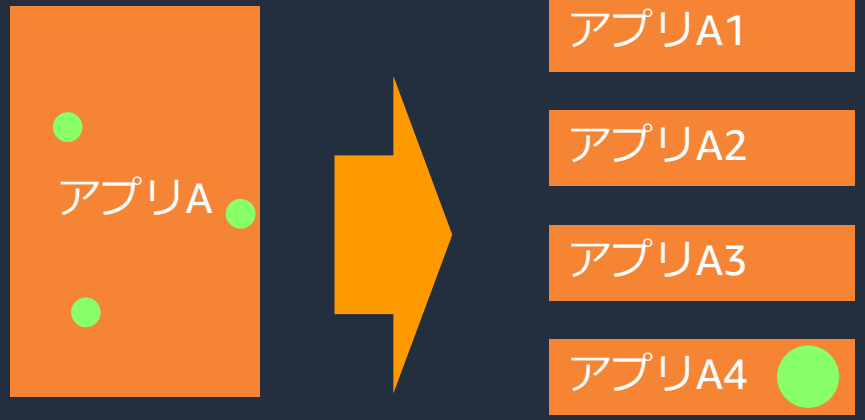


アプリケーションアーキテクチャの観点で柔軟性について検討

柔軟性を高めるには(1/2)

ポイント　モジュールの強度を高めた適切な分割により範囲の特定がしやすくなる

- 影響範囲の把握・限定はしやすいですか？
- 開発・テストはしやすいですか？
- リリースしやすいですか？
- 万が一失敗したときの影響は小さいですか？



変更対象が点在し影響範囲が広くなりやすい

影響範囲等が限定的で変更しやすい

柔軟性を高めるには(2/2)

ポイント コンポーネント間の結合を弱めることで影響の波及を限定的に

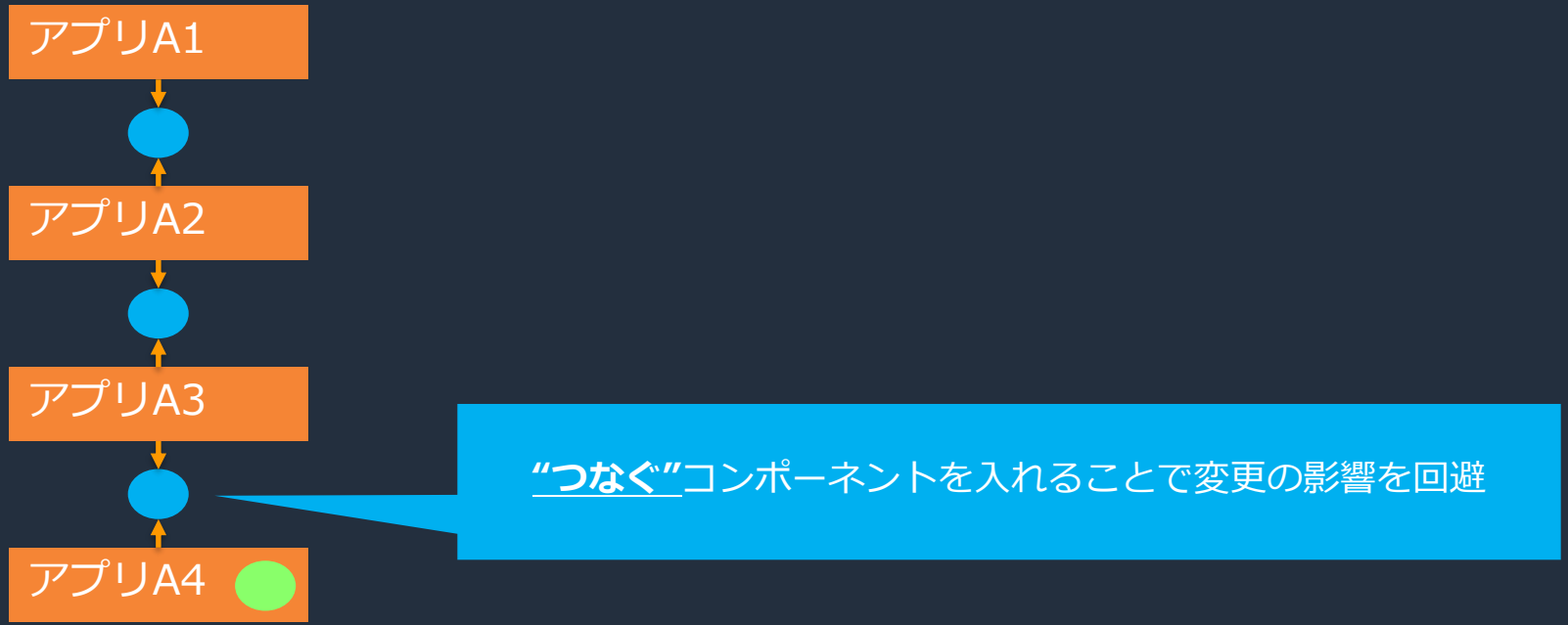
- 影響範囲の把握・限定はしやすいですか？
- 開発・テストはしやすいですか？
- リリースしやすいですか？
- 万が一失敗したときの影響は小さいですか？

密結合な場合、障害の影響を受けやすい

疎結合な場合、影響が限定的

コンポーネント間の結合度を弱めるには？

方法	キューなどのコネクターを利用した疎結合な呼び出し方法の採用



“つなぐ”コンポーネントに求められること

- 機密性
    - アプリケーションデータの保護
- 可用性
    - 耐障害性、自動回復、スケーラビリティ
- 低価格
    - コストパフォーマンスがよい

ここまでのまとめ

- ビジネスのアジリティを上げるために
    - 柔軟性のあるアーキテクチャの採用
        - モノリスからより小さなコンポーネントへ
        - 密結合から疎結合へ
- “つなぐ”コンポーネントが重要

aws

本日のアジェンダ

- 柔軟性とは
- アプリケーション間のつなぎ方
- Amazon SQSの概要
- Amazon SQS機能詳細
- まとめ

お話する内容

- アプリケーション間を”つなぐ”方式
- 関連するAWSサービス

アプリケーションを”つなぐ”際に検討すること

送信するデータはどのようなデータですか？

1.ストリーミング/メッセージングの検討

任意のタイミングで処理をしたいですか？

3.Push/Pull方式検討

呼び出した処理の完了を待つ必要がありますか？

2.同期/非同期方式の検討

処理の依頼先は1か所？複数ですか？

4.P2P※/Publish Subscribe方式の検討

※Point to Point

1.ストリーミング方式/メッセージング方式

ストリーミング方式

連続的にデータを送る方式。データ間に順序性等の意味があり、まとめて処理する方式

- IoTのデータ等、連続してデータを送るケース
- 動画/音声データを連携するケース

メッセージング方式

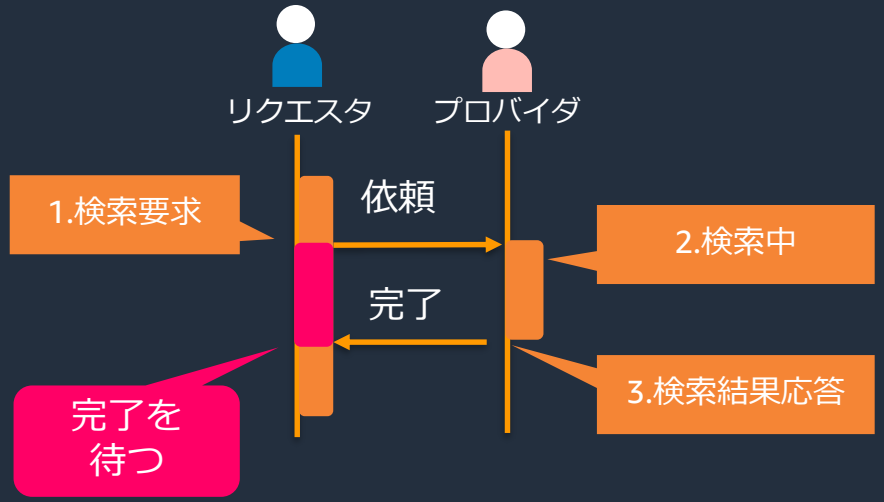
メッセージ間に連続性等の意味をもたせず単体で処理をする方式

- 単発で完結する要求や応答をやり取りするケース

2.同期/非同期方式の検討

同期方式

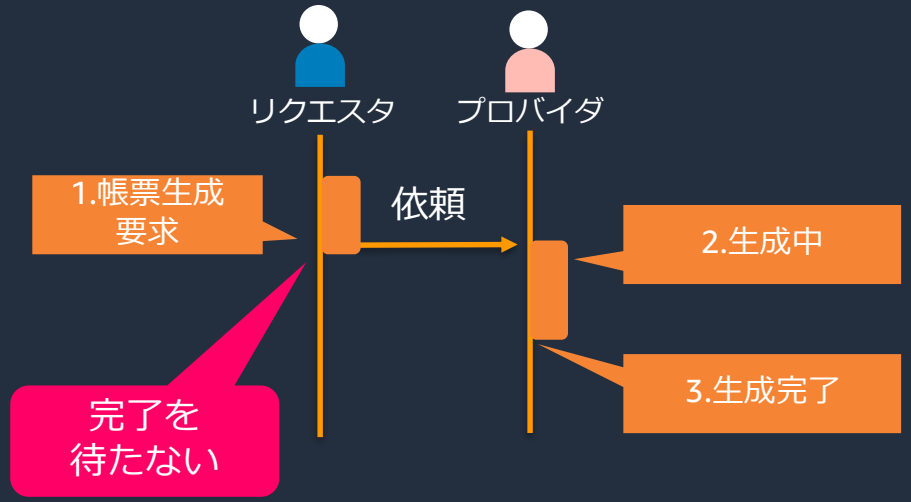
リクエスタがプロバイダの処理完了までの応答を待つ方式



- プロバイダの処理完了が必須なケース
- プロバイダの処理が軽く応答が比較的速いケース

非同期方式

リクエスタが処理完了の応答を待たずに後続の処理を実施する方式

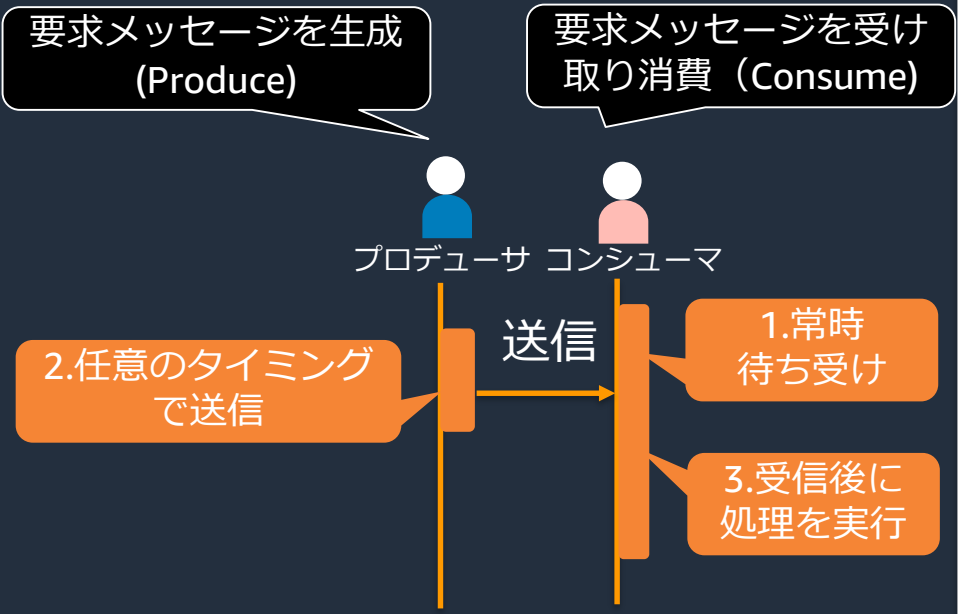


- プロバイダ側の処理完了を必須としないケース
- プロバイダ側の処理が重く応答が比較的遅いケース
- リクエスタ側のスループットを上げたいケース

3.Push/Pull方式検討

Push方式

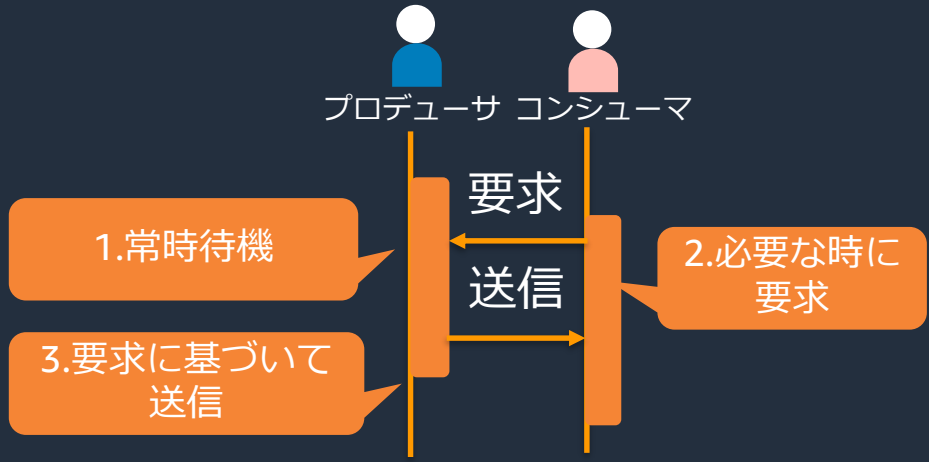
プロデューサがメッセージを送信するとコンシューマに届く方式



・プロデューサの任意のタイミングで「送信」しコンシューマに届けたいケース

Pull方式

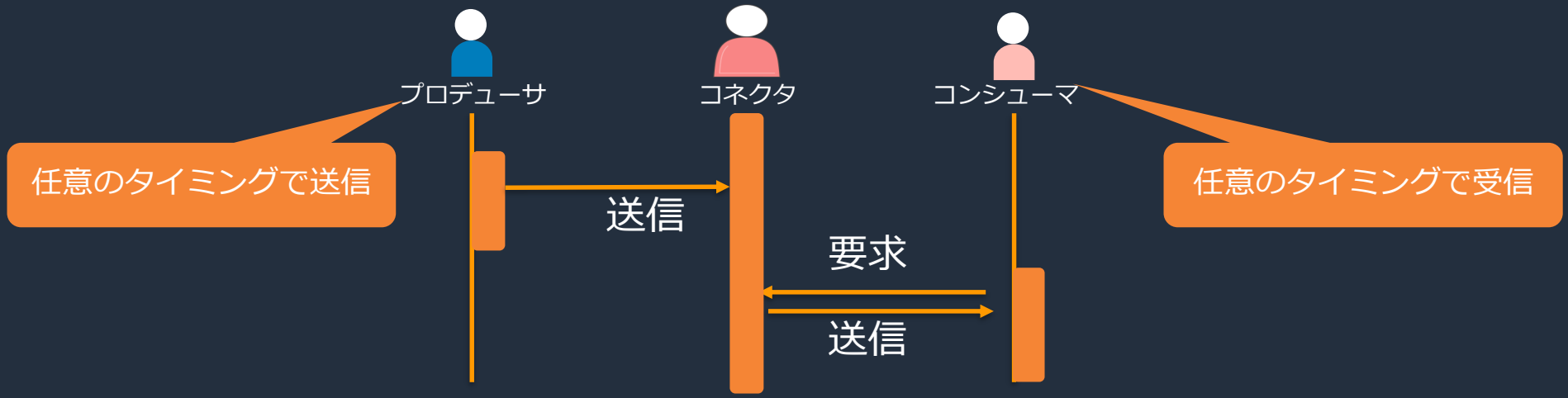
コンシューマがメッセージを要求することで受領する方式



・コンシューマの任意のタイミングで「受信」したいケース

お互い任意のタイミングで送受信したい場合は？

ポイント プロデューサとコンシューマの間にコネクタを入れる



適用ケース

- 双方が任意のタイミングで「送信・受信」をしたいケース
- 双方の障害等の影響を緩和したいケース

4.P2P/ Publish Subscribe方式の検討

P2P※方式

プロデューサとコンシューマが1対1で連携する方式

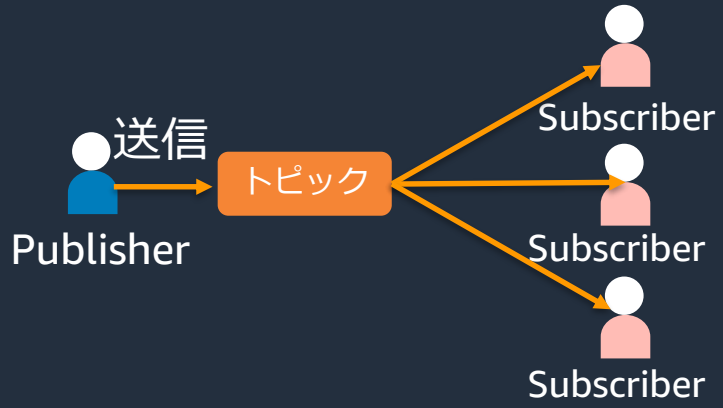
※Point to Point

・リクエストの依頼先が1か所の場合

Publish Subscribe方式

1つのメッセージを複数のコンシューマ※が受信する方式

※Publish Subscribe型ではメッセージ発行者をPublisherと呼び、メッセージ受領者をSubscriberと呼ぶ

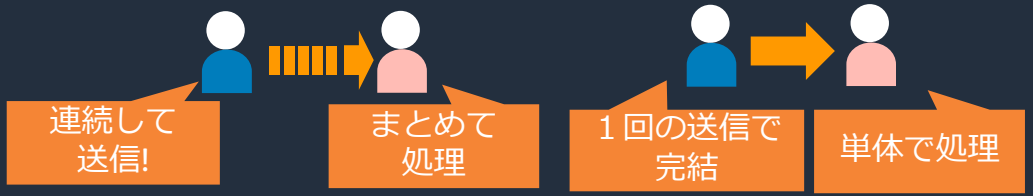


・１つのリクエストで複数の処理を並列で実施したい場合

アプリケーションを”つなぐ”際に検討すること

送信するデータはどのようなデータですか？

1.ストリーミング/メッセージングの検討



任意のタイミングで処理をしたいですか？

3.Push/Pull方式検討

呼び出した処理の完了を待つ必要がありますか？

2.同期/非同期方式の検討

処理の依頼先は1か所？複数ですか？

4.P2P/Publish Subscribe方式の検討

アプリケーション連携をサポートする
主要なAWSのマネージドサービス


Amazon SNS


Amazon MQ

Amazon Managed
Streaming for Kafka


Amazon SQS


Amazon Kinesis

Amazon API Gateway

aws

ストリーミングデータを取り扱いたい場合

Amazon Kinesis

テキストから動画まで

Amazon Managed Streaming for Kafka

Kafkaをマネージドで

Amazon API Gateway

WebSocketをサポート

メッセージデータを取り扱いたい場合

非同期方式

Push方式

Publish Subscribe方式

Pull方式

P2P方式

P2P / Publish Subscribe方式

Amazon SNS

Amazon SQS

Amazon MQ

本日の
対象

aws

ここまでのまとめ

- コンポーネント間を”つなぐ”方式
  - ストリーミング/メッセージ
  - 同期/非同期
  - Push/Pull
  - P2P/Publish Subscribe
- 本日のテーマであるAmazon SQSは
  - 非同期型、Pull型、かつ、P2P型

aws

本日のアジェンダ

- システムの柔軟性
- アプリケーション間のつなぎ方
- Amazon SQSの概要
- Amazon SQS機能詳細
- 料金
- まとめ

お話する内容

- Amazon SQSの特徴/構成要素
- Amazon SQSの利用ケース
- キューとメッセージ

Amazon SQSの特徴

要約	ほぼ無制限のスケーラビリティを備えた フルマネージドな“分散”メッセージキュー

セキュリティ	耐久性	可用性
利用するユーザーのアクセス制御やメッセージの暗号化が可能	複数のサーバー/データセンターに全メッセージを重複して保持（分散キュー）	分散キューモデルを採用することでメッセージの送信/受信の可用性を向上

スケーラビリティ	フルマネージド	初期投資不要
ほぼ無制限のTPS（Transactions Per Second)をサポート	サーバの管理不要。 運用負荷軽減	毎月の無料利用枠 ＋ 使った分だけの従量課金※

※API実行回数＋データ転送料

aws

“つなぐ”コンポーネントに求められること

再掲

- 機密性 **セキュリティ**
  - アプリケーションデータの保護
- 可用性 **耐久性** **可用性** **スケーラビリティ**
  - 耐障害性、自動回復、スケーラビリティ
- 低価格 **フルマネージド** **初期投資不要**
  - コストパフォーマンスがよい

aws

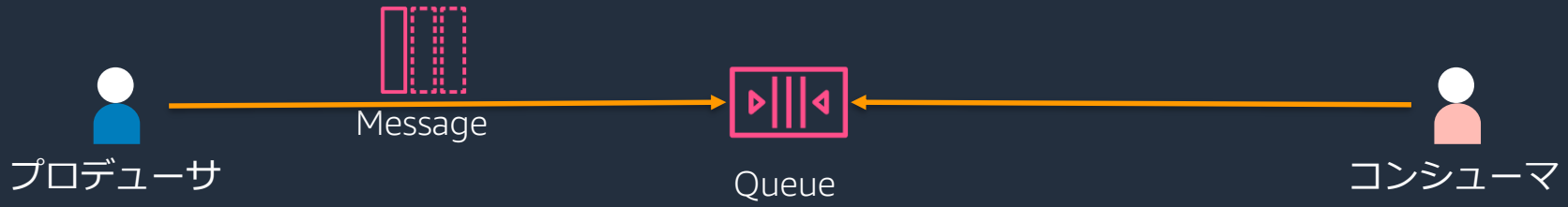
Amazon SQSの構成要素

プロデューサ

「キュー」に「メッセージ」を
送信するアプリケーション

コンシューマ

「キュー」の「メッセージ」を
取得するアプリケーション



メッセージ

プロデューサが生成するデータ。
最大256KB

キュー

メッセージをキューイングする。
分散キュー。
メッセージを最大14日間保持可能

Amazon SQSの利用ケース

1

2

3

4

Amazon SQSの利用ケース

1.バッファリングとバッチ化

一時的なリクエスト増の均一化

2

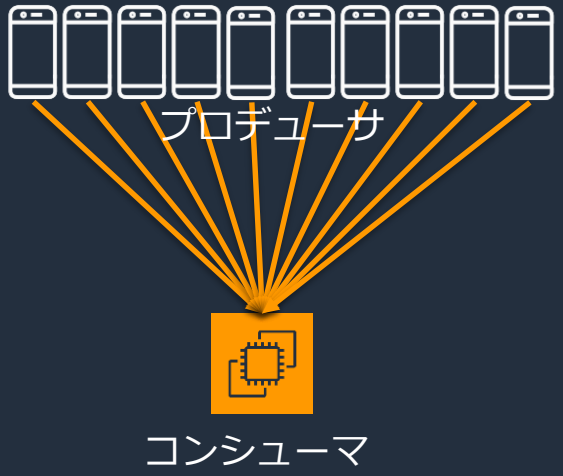
3

4

Amazon SQSの利用ケース(1/4)　バッファリング

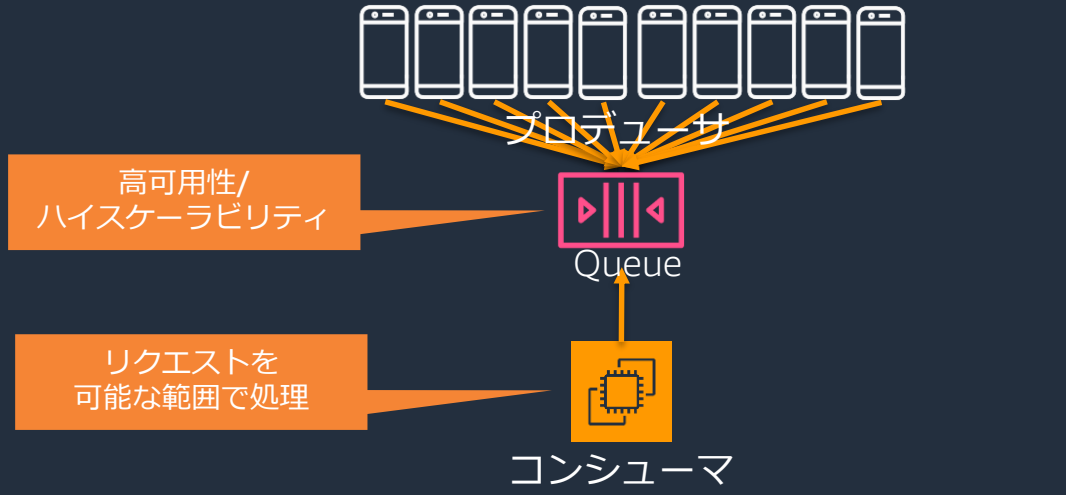
ケース1	大量リクエストが一時的に発生する場合にキューで受ける

Amazon SQSを利用しない場合



・リクエストの受付と処理が一体化するため
リクエストのスパイクに対応しずらい

Amazon SQSを利用する場合



・リクエストのスパイクにも対応可能
・バックエンドのEC2が可能な範囲で処理を実施

Amazon SQSの利用ケース

1.バッファリングとバッチ化

一時的なリクエスト増の均一化

2

3

4

Amazon SQSの利用ケース

1.バッファリングとバッチ化	2.ワークキュー
一時的なリクエスト増の均一化	プロデューサとコンシューマの処理の 依存関係を低減

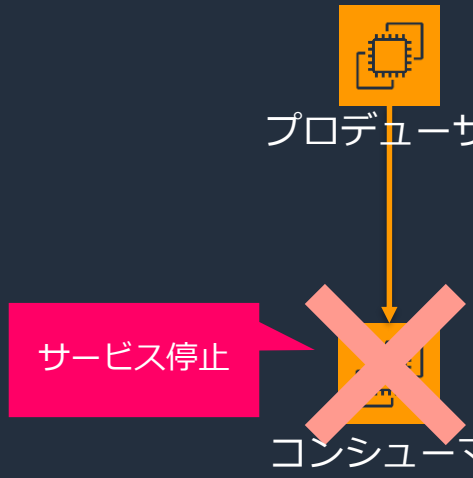
3

4

Amazon SQSの利用ケース(2/4)ワークキュー

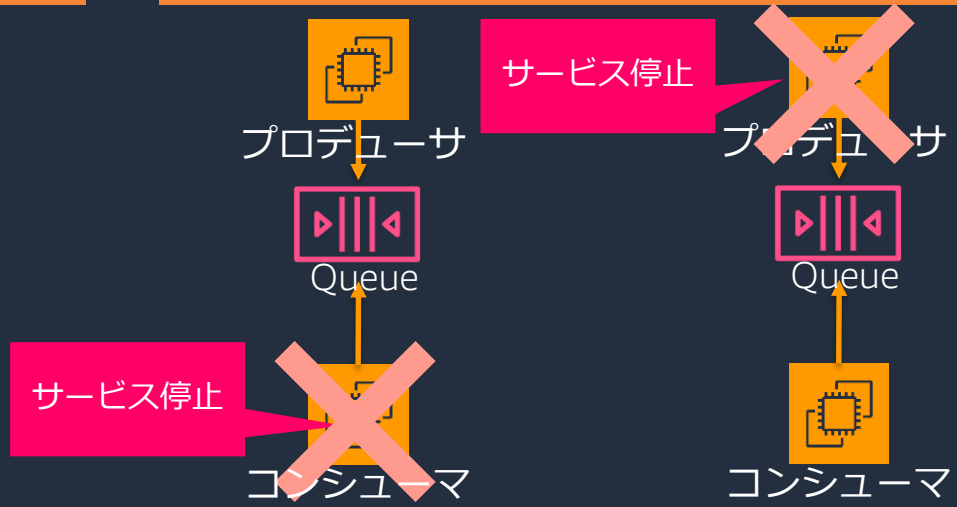
ケース2　アプリケーション間の依存関係を弱めたい場合

Amazon SQSを利用しない場合



・メンテナンス中はサービス停止
・連携先の状況を考慮する必要あり

Amazon SQSを利用する場合



・お互い任意のタイミングで処理が可能
・サービスメンテナンス等、影響を受けにくい

Amazon SQSの利用ケース

1.バッファリングとバッチ化	2.ワークキュー
一時的なリクエスト増の均一化	プロデューサとコンシューマの処理の 依存関係を低減

3

4

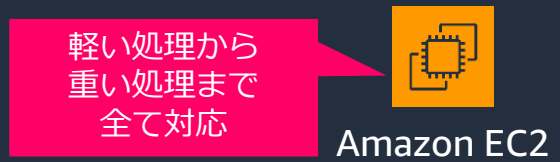
Amazon SQSの利用ケース

1.バッファリングとバッチ化	2.ワークキュー
一時的なリクエスト増の均一化	プロデューサとコンシューマの処理の 依存関係を低減
3.リクエストのオフロード	4
重い処理の切り出しによる応答性能の改善	

Amazon SQSの利用ケース(3/4)リクエストのオフロード

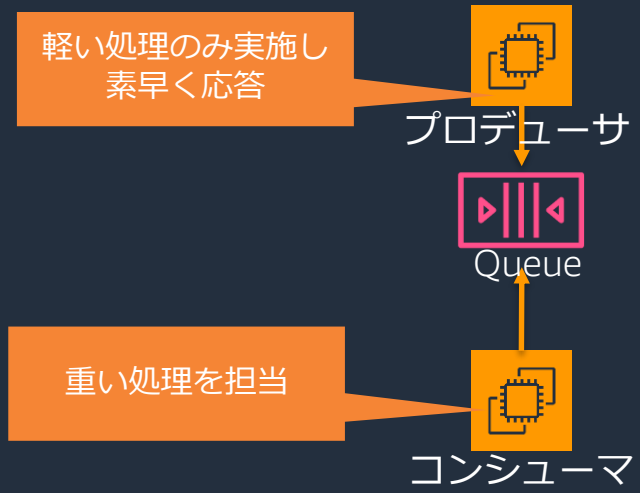
ケース3	重い処理が含まれていても素早く応答をしたい場合

Amazon SQSを利用しない場合



- 全処理を担当するため、レスポンスが遅延
- リソースを効率よく利用できない可能性あり

Amazon SQSを利用する場合



- 素早いレスポンスが可能
- 重い処理に合わせたリソースの割当が可能

Amazon SQSの利用ケース

1.バッファリングとバッチ化	2.ワークキュー
一時的なリクエスト増の均一化	プロデューサとコンシューマの処理の依存関係を低減
3.リクエストのオフロード	4
重い処理の切り出しによる応答性能の改善	

Amazon SQSの利用ケース

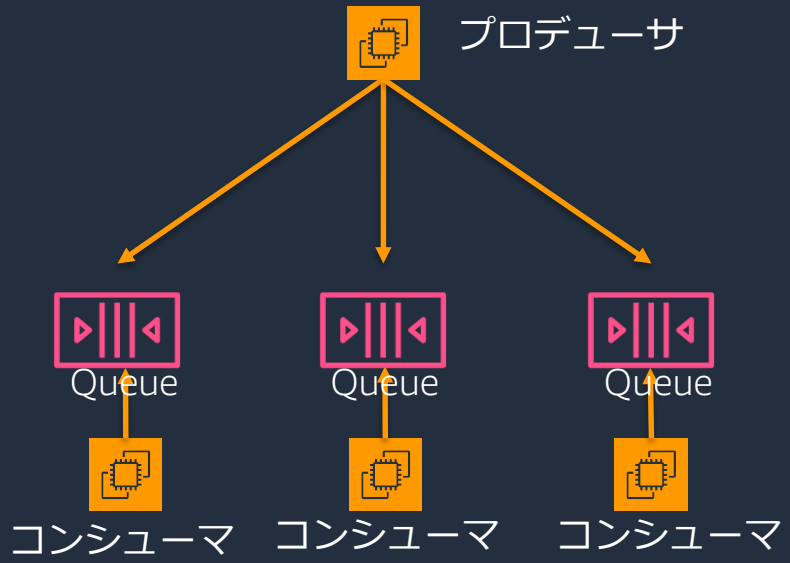
1.バッファリングとバッチ化	2.ワークキュー
一時的なリクエスト増の均一化	プロデューサとコンシューマの処理の 依存関係を低減

3.リクエストのオフロード	4.ターゲットのファンアウト
重い処理の切り出しによる応答性能の改善	Amazon SNSとの組み合わせによる 処理の並列化

Amazon SQSの利用ケース(4/4)ターゲットのファンアウト

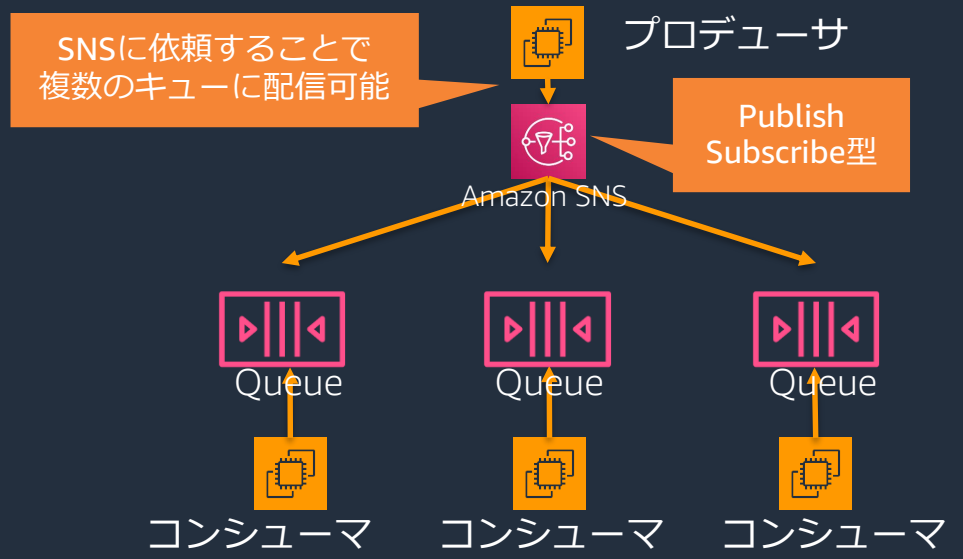
ケース4　複数の処理を並列処理したい場合

Amazon SQSのみで構成する場合



・プロデューサー側で並列化の制御等を実装する必要あり

Amazon SNSと組み合わせた場合



・Amazon SNSと組み合わせることで１つのメッセージ送信で並列化が可能
・Pull型でより疎結合な構成

Amazon SQSの利用ケース

1.バッファリングとバッチ化	2.ワークキュー
一時的なリクエスト増の均一化	プロデューサとコンシューマの処理の 依存関係を低減

3.リクエストのオフロード	4.ターゲットのファンアウト
重い処理の切り出しによる応答性能の改善	Amazon SNSとの組み合わせによる 処理の並列化

Amazon SQSのキューの特徴

重要	スタンダードキューとFIFO(First In First Out)キューの二種類。 それぞれの特徴を理解してアプリケーション設計をすること。

	スタンダードキュー	FIFOキュー
スループット	ほぼ無制限のスループット	1 秒あたり最大 300 件のメッセージ (300 件の送信、受信、または削除オペレーション) をサポート
配信方式	少なくとも1回の配信 (二回以上の配信もあり得る)	1回のみ配信
配信順序	ベストエフォート (順序が変わることもある)	順序性を保つ
利用料金 ※	100万件を超えた場合、 100万件ごとに0.40USD	100万件を超えた場合、 100万件ごとに0.50USD

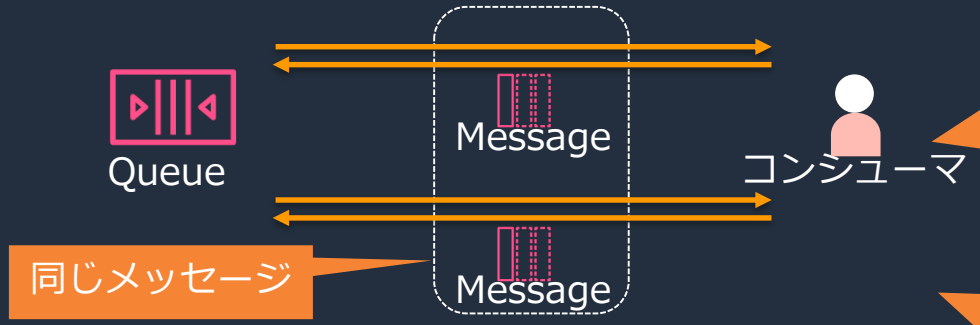
※2019年7月17日時点。API利用回数に関する利用料金。別途データ転送料金が発生
毎月100万回までは無料利用枠の範囲内。

2018年11月
東京リージョンにリリース

スタンダードキュー：https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/standard-queues.html
FIFOキュー：https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html

二回以上の配信がある場合の設計とは？

考え方　冪等性（べきとうせい）を確保できるようにアプリケーションを設計する



同じメッセージで処理をした場合、何回処理をしても結果が変わらない設計

例：画像の整形処理等は元ネタが同じであれば結果は同じ

Dynamo DB等を活用して処理済みであることを判定できる仕組みを導入し重複実行しない設計

例：アプリで発行したID等を記録し重複処理を排除

冪等性　１回だけ操作を行っても何回（N回）行っても結果が変わらない特性

キューのメッセージ取得方法

要約	「ショートポーリング」と「ロングポーリング」の二種類の方式があり、 ロングポーリングを通常はご利用いただく

	ショートポーリング	ロングポーリング
応答方式	即応答。 メッセージがない場合は「空」を応答	最大20秒メッセージの受領を待つ。 メッセージがない場合はタイムアウト。 その場合は「空」を応答
取得 メッセージ	分散されたサーバの中からサンプリングされた サーバのメッセージを応答。全サーバではないた め取得できないこともある。	全てのサーバをクエリしメッセージを 応答
利用料金	繰り返しショートポーリングを実施する場合API コール数が増え利用料金が増加する可能性あり	ショートポーリングに比べAPIコール数が抑制で きるため利用料金が安価になる 可能性あり
利用シーン	複数のキューを１つのスレッドで ポーリングするようなケース	多くのケースはロングポーリング方式。 複数のキューをポーリングする必要がないケース

ロングポーリング：https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-long-polling.html

Amazon SQSでメッセージを取得する際のお作法

1．ポーリング

ロングポーリング or ショートポーリング

特定のIDや条件によるメッセージの
取得はできません

2．取得＆処理

受信したメッセージを利用して処理

3．削除

キューに削除指示

aws

ここまでのまとめ

- Amazon SQSには二種類のキュー
    - スタンダードキュー：最低1回の配信/順序はベストエフォート
    - FIFOキュー：1回のみ配信/順序を保証

- 冪等性
- メッセージを受領後は削除処理が必要

本日のアジェンダ

- システムの柔軟性
- アプリケーション間のつなぎ方
- Amazon SQSの概要
- Amazon SQS機能詳細
- まとめ

Amazon SQSにまつわるお客様の関心事

処理中のメッセージは
どういう扱い？

メッセージを一定期間
受信できなくできますか？

エラーハンドリングは？

メッセージは暗号化可能？

キューに誰がアクセスできる？

メタ情報を設定できますか？

キューの状態は
どのように分かりますか？

Amazon SQS/Amazon Kinesisの
使い分けは？

お話する内容

- 可視性タイムアウト
- 遅延キュー/メッセージタイマー
- Dead Letter Queue
- サーバサイド暗号化
- キューへのアクセス制御
- メッセージ属性
- モニタリング
- Amazon SQS と Amazon Kinesis

aws

可視性タイムアウトによる処理中のメッセージロック

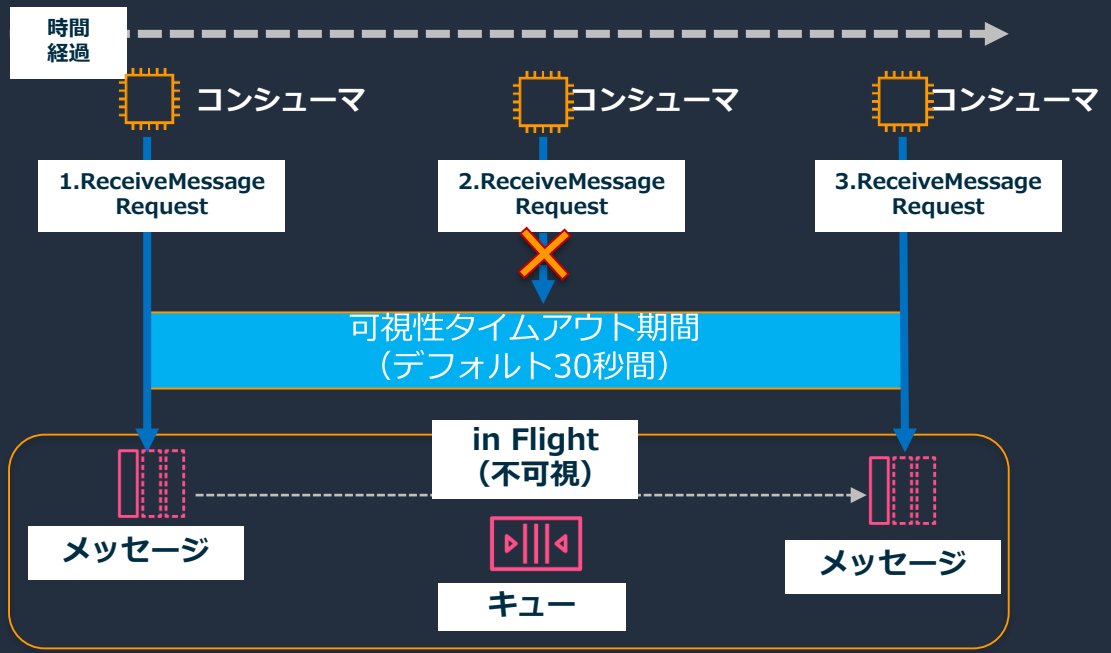
機能	コンシューマが取得したメッセージに対して指定された期間（デフォルト30秒）、他のコンシューマからの同一メッセージへのアクセスをブロックする機能

利用目的

- 複数のコンシューマが同じメッセージを処理するのを防ぐ
- 指定時間を超えるとアクセス可能となりアプリ障害等発生時に再度、他のコンシューマで処理が可能

考慮点

- 通常の処理に必要な時間より大きな値を設定することで重複処理の発生を防止。ただし、大きすぎると障害時等の再処理が遅延する。
- スタンダードキューの場合は、可視性タイムアウトはメッセージを 2 回受信しない保証にはなりません。



可視性タイムアウト：https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html

遅延キューとメッセージタイマー

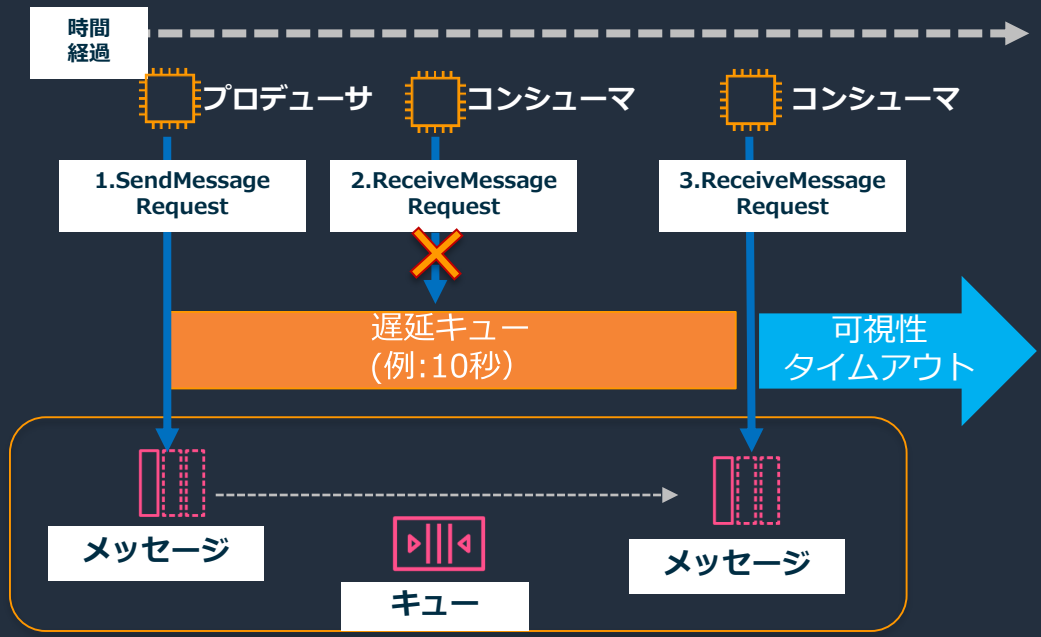
機能	メッセージをキューに**送信**してから一定時間経過後に利用可能になる機能 （可視性タイムアウトは**受信後**）

利用目的

- メッセージの処理を一定時間遅延させて実施したい場合に利用する。
- 例えばリトライ処理等を一定時間後に利用したい場合等に利用可能

考慮点

- 遅延キューはキュー全体に、メッセージタイマーは特定のメッセージに対して遅延時間を設定するもの
- 両方指定された場合はメッセージタイマーの値が優先される。
- メッセージタイマーはFIFOキューは未サポート。



メッセージタイマー：https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-message-timers.html
遅延キュー：https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-delay-queues.html

Dead Letter Queue(DLQ)を利用したメッセージの滞留回避

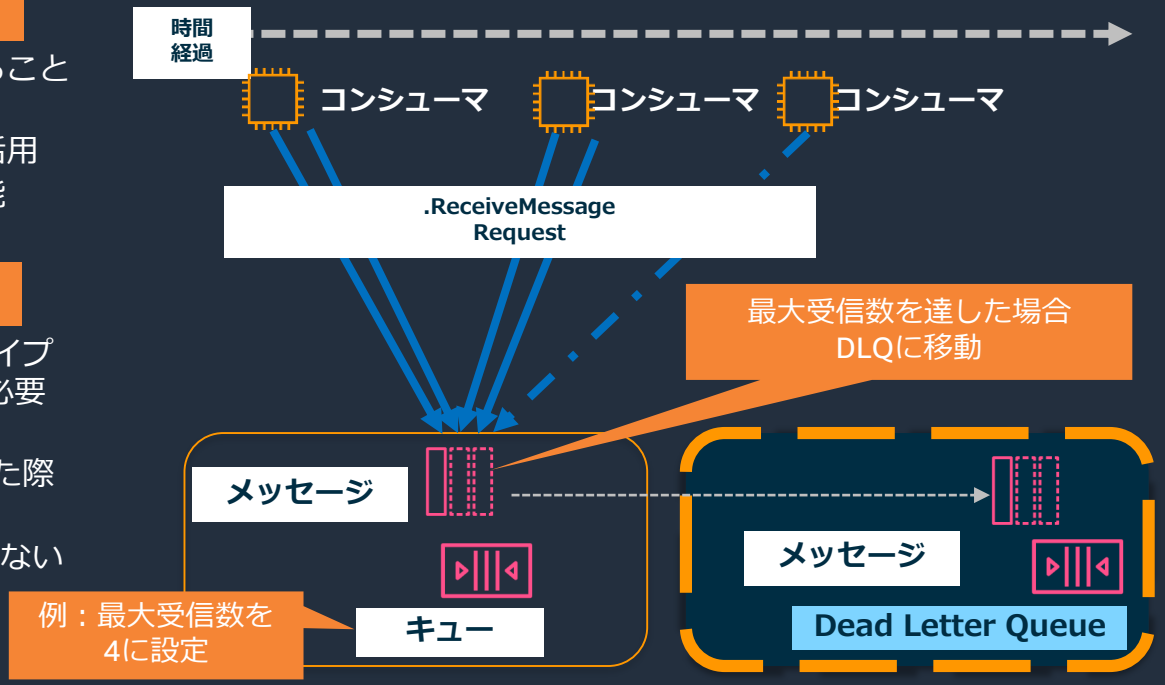
機能	正しく処理できないメッセージがキュー内に滞留し続ける状態を回避するために 分離先として利用されるキュー。分離後、メッセージを解析して原因分析が可能

利用目的

- 正常処理できないメッセージが残り続けることを早期（保持期間より前）に回避する
- DLQのメッセージを解析し原因の分析に活用
- DLQにアラームを設定することで検知可能

考慮点

- 同一リージョン、同一アカウント、同一タイプ（FIFO/スタンダードキュー）で作成する必要あり
- データの保持期間は元のキューに追加された際のタイムスタンプに基づく
- FIFOキューで順序が変わることが許容できない場合は利用しない。
- 無制限に繰り返したい場合は利用しない。



DLQ：https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-dead-letter-queues.html

サーバサイド暗号化を利用したメッセージの暗号化

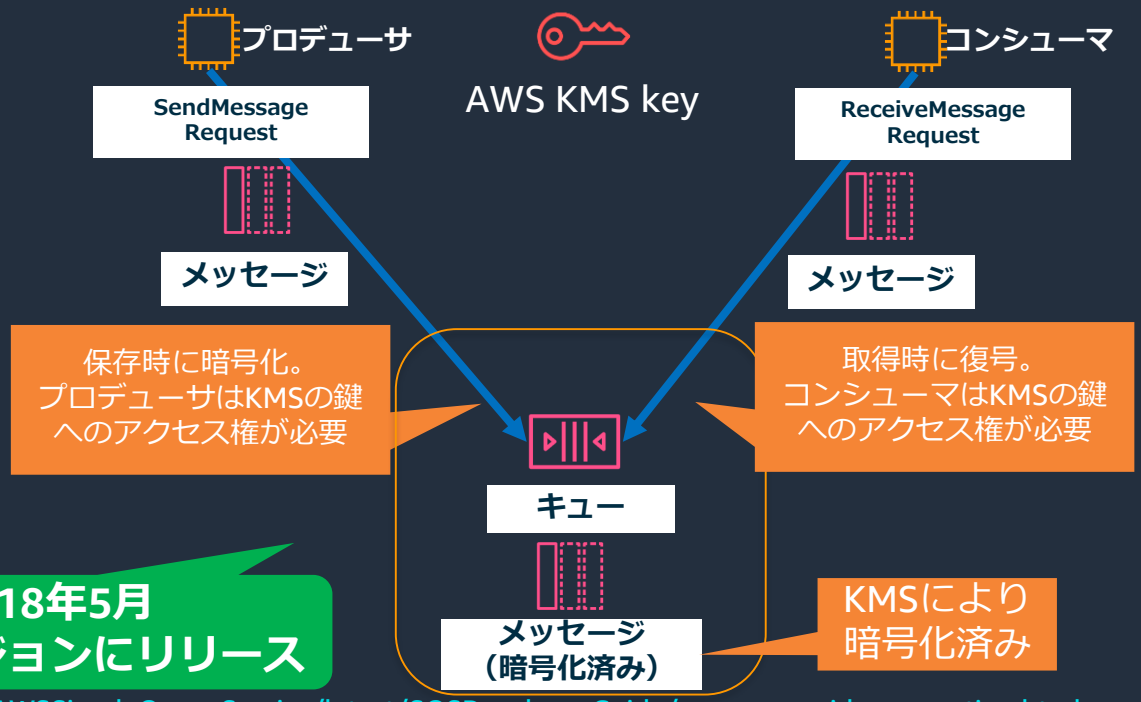
機能	AWS Key Management Service (AWS KMS) で管理されているキーを使用して Amazon SQSキュー内のメッセージの内容を保護

利用目的

- キュー内に保存されたメッセージを暗号化により保護
- キューへのアクセス権に加えAWS KMSの鍵へのアクセス権をもつユーザーのみがメッセージの送信・受信が可能になる

考慮点

- 暗号化対象はメッセージ本文で、メッセージ属性（後述）は対象外
- AWS 利用料金が別途必要。暗号化に利用するデータキーのキャッシュ時間の調整によりコストを調整可能



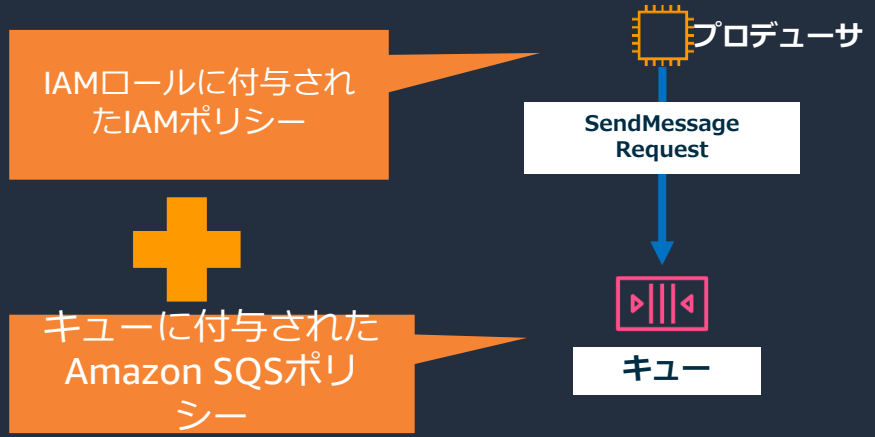
2018年5月
東京リージョンにリリース

サーバサイド暗号化：https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-server-side-encryption.html

キューのアクセス制御

機能	IAMポリシーとAmazon SQSポリシーのいずれか、または、両方でアクセス制御を実現

	IAMポリシー	Amazon SQSポリシー
適用先	IAMユーザー、IAMロール	キュー
利用例	特定のユーザーやロールに対して アクセス制御を行いたい場合	特定のキューに対する アクセス制御を行いたい場合



例:IAMポリシー

```
{
  "Version":"2012-11-05",
  "Statement" : [
    {
      "Sid":" Thank_you_for_attending",
      "Effect":"Allow",
       "Action":["sqs:SendMessage","sqs:ReceiveMessage"],
      "Resource": "arn:aws:sqs:ap-northeast-
1:123456789012:queue2“
    }]
}
```

アクセス制御：https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-overview-of-managing-access.html

メッセージ属性を利用したメタ情報の格納

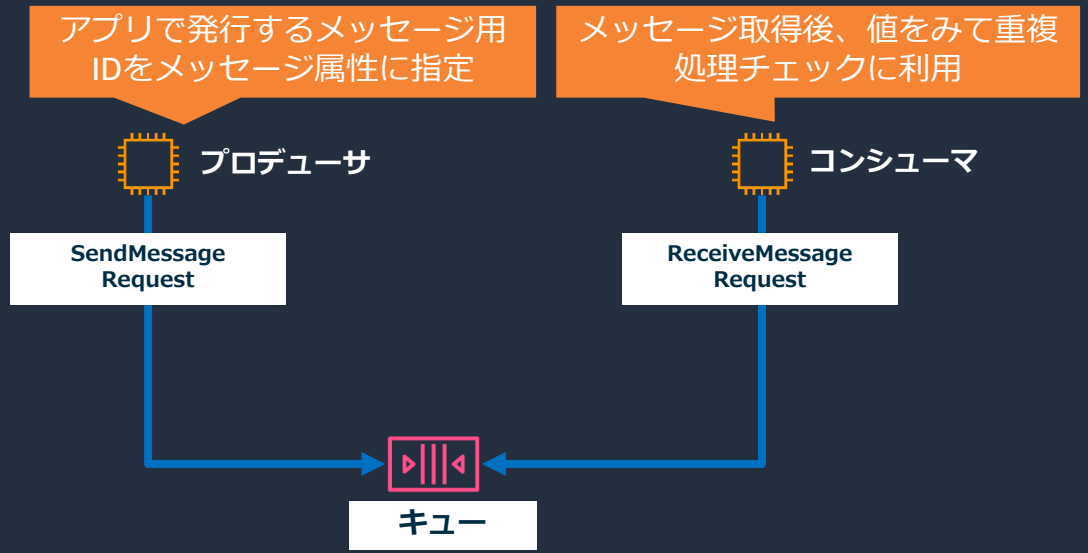
機能	メッセージ本文とは別にメタデータ等を保持させることができる。最大10個。

利用目的

- メッセージ本文を解析することなく、任意の属性（例：時刻、地理情報、署名等）を利用し処理の実施の判断等が可能

考慮点

- メッセージサイズ上限の256KBに含まれる
- サーバサイド暗号化の対象外。暗号化対象はメッセージ本文で、メッセージ属性は対象外



メッセージ属性：https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-message-attributes.html

キューのモニタリング

機能	Amazon CloudWatchを利用することでキューの状態を把握可能に。 コンシューマーのスケールアウト/インに利用可能

AproximateAgeOfOldestMessage	キューで最も古い削除されていないメッセージのおおよその経過期間
ApproximateNumberOfMessagesDelayed	遅延が発生したため、すぐに読み取ることのできない、キューのメッセージ数。遅延キューやメッセージタイマー指定時
ApproximateNumberOfMessagesNotVisible	処理中のメッセージの数。メッセージがクライアントに送信されたが、まだ削除されていない場合、または可視性タイムアウトに達していない場合、メッセージは処理中とみなされる
ApproximateNumberOfMessagesVisible	キューから取得可能なメッセージの数。
NumberOfEmptyReceives	メッセージを返さなかった ReceiveMessage API 呼び出しの数。
NumberOfMessagesDeleted	キューから削除されたメッセージの数。
NumberOfMessagesReceived	ReceiveMessage アクションへの呼び出しで返されたメッセージの数。
NumberOfMessagesSent	キューに追加されたメッセージの数。
SentMessageSize	キューに追加されたメッセージのサイズ（バイト数）

注意点：スタンダードキューの場合は概算値であり、FIFOキューの場合は厳密な値であること

https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-available-cloudwatch-metrics.html

Amazon SQS/Amazon Kinesisの使い分け(1/2)

要点	ストリーミングデータであればKinesisを利用。それ以外は、アプリケーションで処理するデータの特性等に応じて使い分ける（データ処理方法、データサイズ、保持期間やAWSサービス利用料）

ストリーミングデータを取り扱うケース

- 連続的に発生するデータをまとめて処理をする
  - 例：1台のデバイスから発生するログの解析
  - 例：Webブラウザの操作ログ等の解析
  - 例：監視カメラの映像データ等の解析

単発のメッセージではなく関連する複数のメッセージを処理するケース

Amazon Kinesis

Amazon SQS/Amazon Kinesisの使い分け(2/2)

	Amazon SQS		Amazon Kinesis Data Streams
	スタンダードキュー	FIFOキュー	
データ順序	ベストエフォート	First-In First-Out	シャード内で順序保証
スループット	ほぼ無制限	アクション※ごとに 1 秒あたり最大 300 件	シャード単位に以下のスループット 書き込み：1000レコード/秒 or 1MiB/秒 読み込み：2MiB/秒
データの扱い	メッセージ消費型。 同一メッセージの並列処理には向かない		複数のConsumerで 同一データを並列処理可能。
データサイズ	256KB		1MiB
保持期間	デフォルト4日間、最大14日間		デフォルト1日、最大7日間

※アクション:SendMessage/ReceiveMessage/DeleteMessage。なおバッチ処理の場合は別

Amazon SQSにまつわるお客様の関心事

再掲

処理中のメッセージは
どういう扱い？

メッセージを一定期間
受信できなくできますか？

エラーハンドリングは？

メッセージは暗号化可能？

キューに誰がアクセスできる？

メタ情報を設定できますか？

キューの状態は
どのように分かりますか？

Amazon SQS/Amazon Kinesisの
使い分けは？

Amazon SQSのご利用料金

計算方法	APIリクエスト数に基づく料金　＋　データ転送料

無料 利用枠	毎月100万件のリクエストまで 無料利用枠による無料。超えた分に対して請求
価格	100万件を超えた場合、100万件ごとに スタンダードキュー　：0.40USD FIFOキュー　：0.50USD
データ 転送料※	受信（IN）：　0USD/GB 送信（OUT):0.9USD/GB~

※ 1 つのリージョン内における Amazon SQS と Amazon EC2 間でのデータ転送は無料

- AWS KMSを利用する場合は別途KMSの利用料金が発生
- メッセージは64KB単位に 1 APIリクエストと計算される

https://aws.amazon.com/jp/sqs/pricing/

本日のアジェンダ

- システムの柔軟性
- アプリケーション間のつなぎ方
- Amazon SQSの概要
- Amazon SQS機能詳細
- まとめ

当セミナーのゴールと想定視聴者

再掲

ゴール	Amazon Simple Queue Service(Amazon SQS)の特徴と活用例を学び今後のシステム構築で利用できるようになる
想定視聴者	• キューを利用したシステム構築が未経験な方 • Amazon SQSを利用したことのない方 • 最新のAmazon SQSの機能を知りたい方

Amazon SQSの特徴

再掲

要約

ほぼ無制限のスケーラビリティを備えた
フルマネージドな“分散”メッセージキュー

セキュリティ	耐久性	可用性
利用するユーザーのアクセス制御やメッセージの暗号化が可能	複数のサーバー/データセンターに全メッセージを重複して保持（分散キュー）	分散キューモデルを採用することでメッセージの送信/受信の可用性を向上

スケーラビリティ	フルマネージド	初期投資不要
ほぼ無制限のTPS（Transactions Per Second)をサポート	サーバの管理不要。運用負荷軽減	毎月の無料利用枠 ＋ 使った分だけの従量課金※

※API実行回数＋データ転送料

aws

Amazon SQSの利用ケース

再掲

1.バッファリングとバッチ化	2.ワークキュー
一時的なリクエスト増の均一化	プロデューサとコンシューマの処理の 依存関係を低減

3.リクエストのオフロード	4.ターゲットのファンアウト
重い処理の切り出しによる応答性能の改善	SNSとの組み合わせによる 処理の並列化

aws

Q&A

お答えできなかったご質問については
AWS Japan Blog 「https://aws.amazon.com/jp/blogs/news/」にて
後日掲載します。

7 月の Black Belt Online Seminar 配信予定

https://amzn.to/JPWebinar

~~7/3 (水) 18:00-19:00 Amazon MQ~~
~~7/5 (金) 18:00-19:00 AWS Summit TOKYO/OSAKA 2019 振り返り 2019 年主要アップデートまとめ~~
~~7/16 (火) 12:00-13:00 Amazon Personalize~~
7/17 (水) 18:00-19:00 Amazon Simple Queue Service(SQS)
7/23 (火) 12:00-13:00 AWS CloudHSM
7/24 (水) 18:00-19:00 AWS Command Line Interface
7/30 (火) 12:00-13:00 Amazon CloudFront
7/31 (水) 18:00-19:00 Amazon ECS Deep Dive








AWS の日本語資料の場所「AWS 資料」で検索

https://amzn.to/JPArchive

AWS Well-Architected 個別技術相談会

毎週”W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に対策などを相談することも可能
- 申込みはイベント告知サイトから

(https://aws.amazon.com/jp/about-aws/events/)

AWS イベント で[検索]

ご視聴ありがとうございました

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive