



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 EventBridge

サービスカットシリーズ

Solutions Architect  
野上 恒平  
2020/1/22

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



過去資料  
<https://amzn.to/JPArchive>



# 自己紹介

## 野上恭平

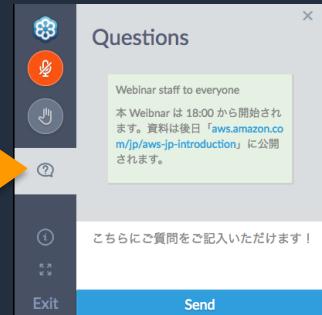
- 所属
  - アマゾン ウェブ サービス ジャパン株式会社  
技術統括本部
  - ソリューションアーキテクト
- 好きなサービス
  - AWS Lambda
  - IAM
  - CloudFormation



# 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2020年1月22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.

# 本日の内容

1. EventBridge とは?
  - イベントバス導入の背景
  - アーキテクチャ
  - 構成要素
  - アクセス制御
2. SaaS との連携
  - SaaSアプリケーションとの連携案
  - 対応しているSaaSアプリケーション
  - SaaSアプリケーションとの連携方法
  - EventBridge 活用事例
  - EventBridgeパートナーになるには
3. クロスアカウント連携
4. 直近のアップデート
5. 料金、制限
6. まとめ

# 本日の内容

## 1. EventBridge とは？

- イベントバス導入の背景
- アーキテクチャ
- 構成要素
- アクセス制御

## 2. SaaS との連携

- SaaSアプリケーションとの連携案
- 対応しているSaaSアプリケーション
- SaaSアプリケーションとの連携方法
- EventBridge 活用事例
- EventBridgeパートナーになるには

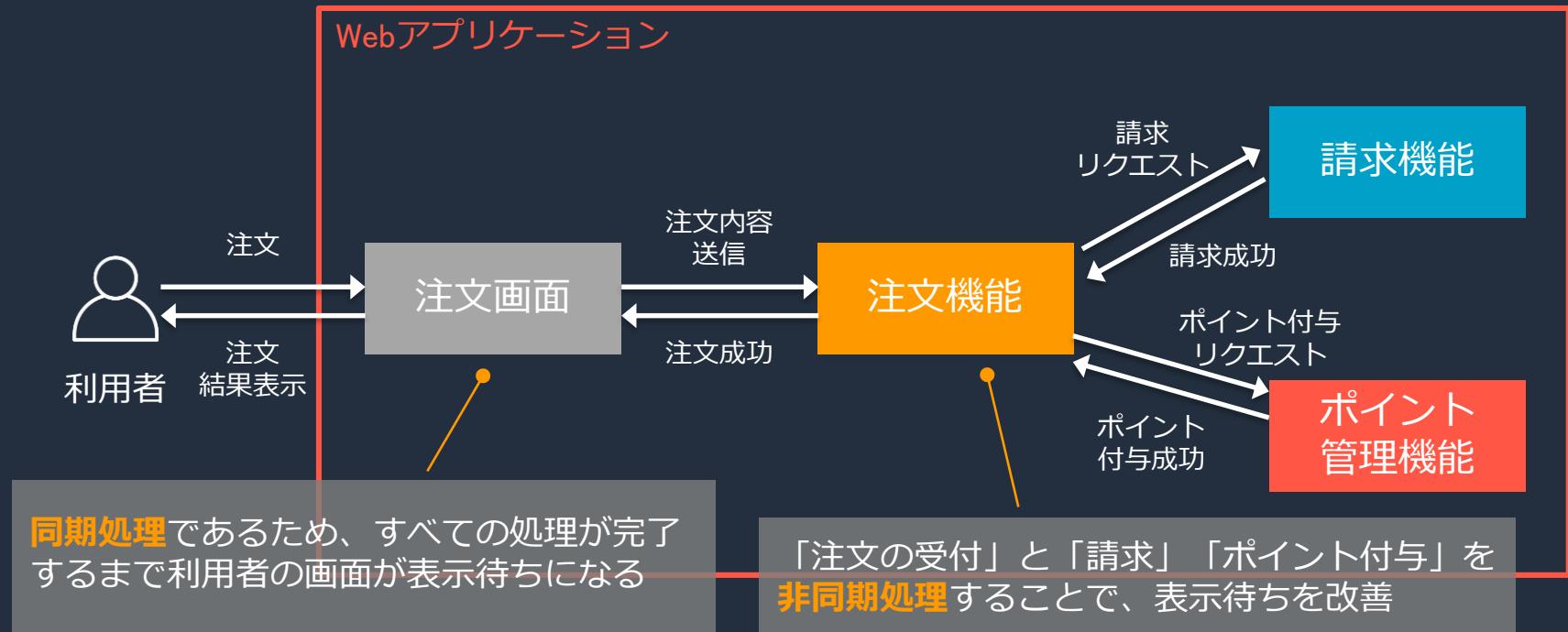
- 3. クロスアカウント連携
- 4. 直近のアップデート
- 5. 料金、制限
- 6. まとめ

# 本日の内容

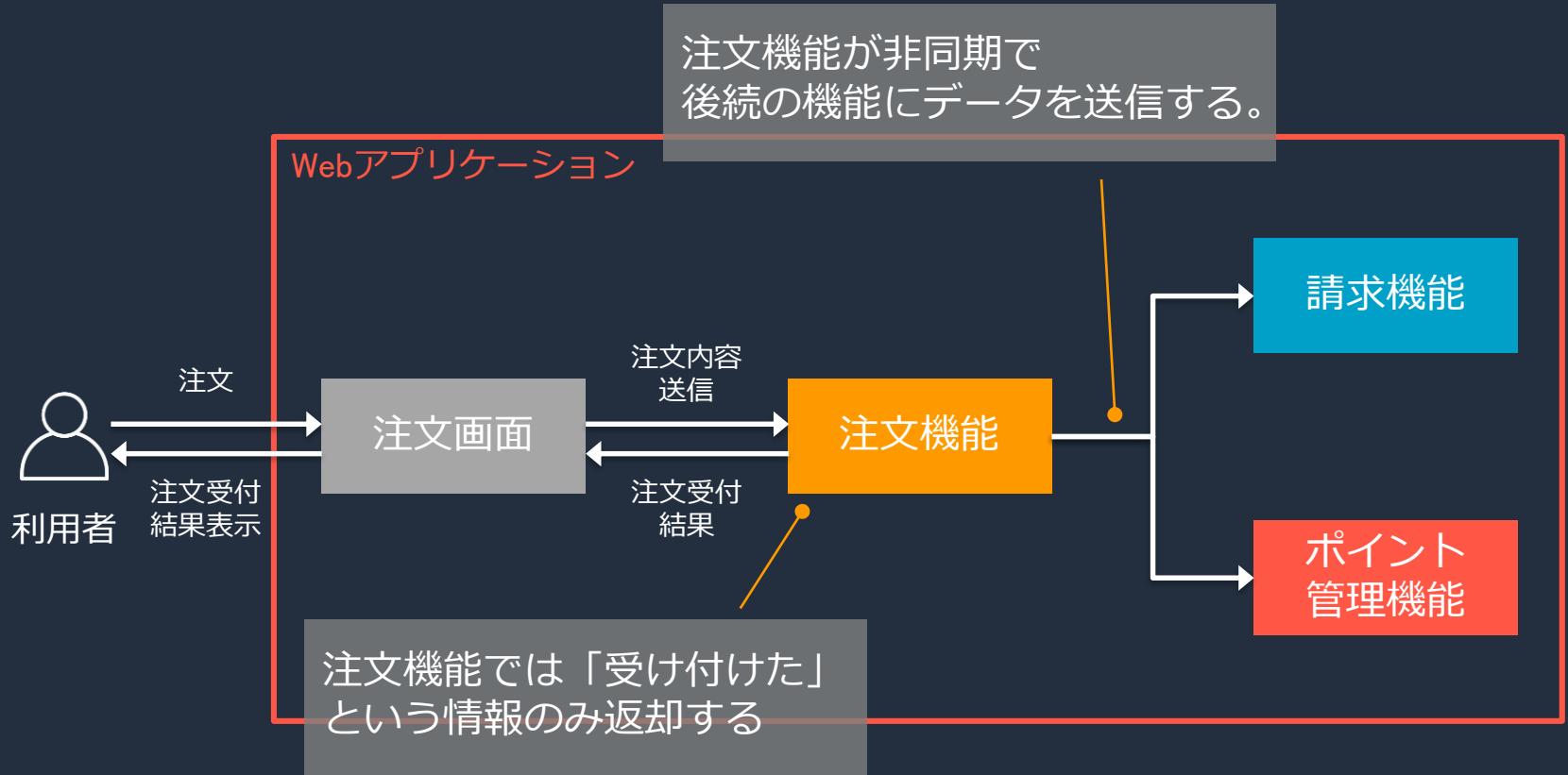
1. EventBridge とは?
  - イベントバス導入の背景
  - アーキテクチャ
  - 構成要素
  - アクセス制御
2. SaaS との連携
  - SaaSアプリケーションとの連携案
  - 対応しているSaaSアプリケーション
  - SaaSアプリケーションとの連携方法
  - EventBridge 活用事例
  - EventBridgeパートナーになるには
3. クロスアカウント連携
4. 直近のアップデート
5. 料金、制限
6. まとめ

# ECサイトの例

## イベント＝状態の変化

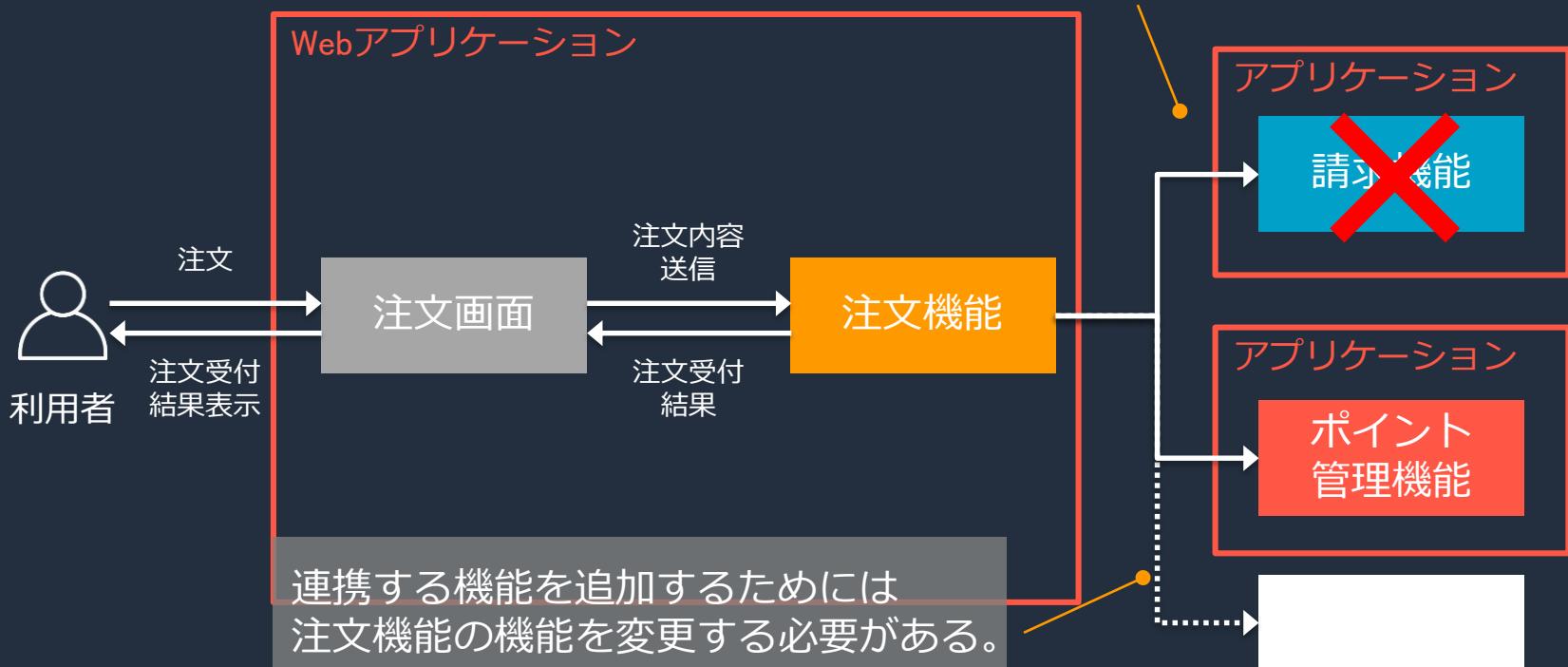


# ECサイトの例（非同期処理で実現）



# ECサイトの例（アプリケーションを分割する）

多数の注文を受けると後続で処理できなくなり、処理に失敗する。

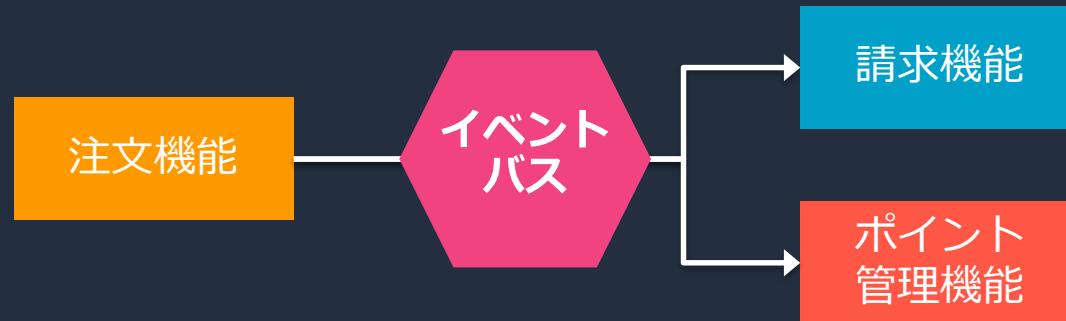


# イベントバスの導入

メッセージの送受信を管理するイベントバスを導入する。

イベントバスは**Push型**で**非同期**で一度に单一のメッセージを処理する。

送信者（Publisher）が受信者（Subscriber）を意識しない。





# Amazon EventBridge

AWSサービス、カスタムアプリケーション、  
SaaSアプリケーションのための  
サーバレスイベントバスサービス

- 統合に必要な「point-to-point」の実装を取り除く
- シンプルなプログラミングモデルを提供する
- 多数のAWS サービスとSaaS アプリケーションを接続する
- フルマネージドで  
サーバーレスなイベントバス

# お客様へもたらす価値

イベント駆動  
アーキテクチャを  
容易に構築

実装量の削減

運用負荷の  
低減

SaaS 由来の  
データの利用

イベントの送信元・送  
信先がお互いの実装を  
意識する必要が無い。

データの取り込み  
フィルタ、配信が  
実装なしに実現可能

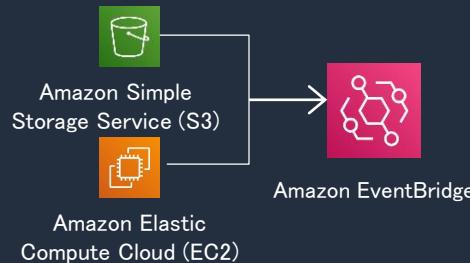
イベント連携用の  
サーバーが不要。  
自動でスケールし、イ  
ベント数での従量課金

AWS サービスや SaaS ア  
プリケーションの  
イベントを利用して  
ワークフローを実行

# イベントバス

EventBridgeはイベントバスでイベントを受け取る。

## デフォルト



## カスタム



## パートナー



### AWSサービス

例) EC2インスタンスが停止した

AWSが発行する  
システム的なイベント

### 独自のアプリケーション

例) 注文機能が注文を受け付けた

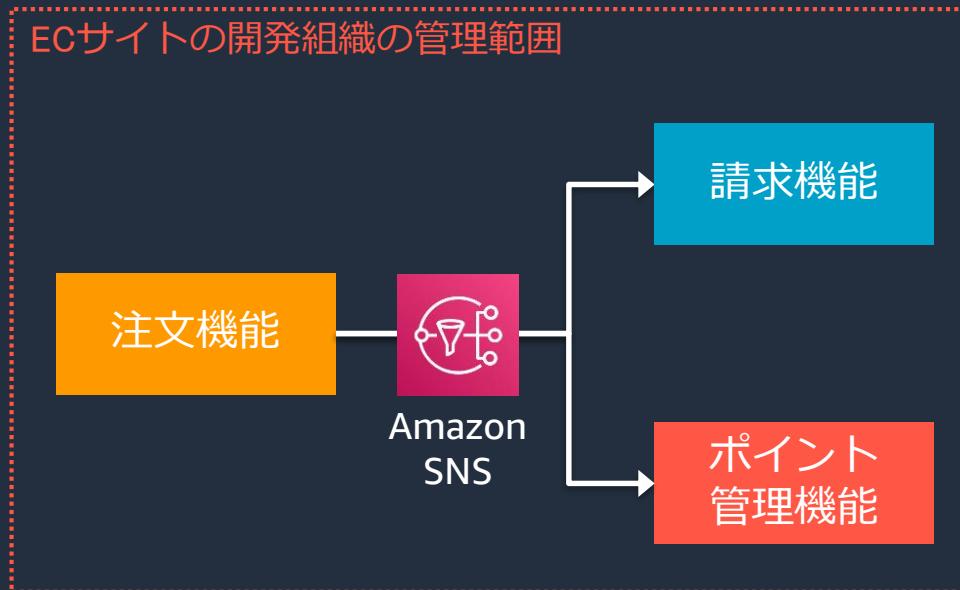
お客様/SaaSが発行する  
業務的なイベント

### SaaS

例) SaaSのCRMで顧客情報が更新された

# 機能間の連携

機能間のデータ連携方法の一例として、Amazon Simple Notification Service(SNS)を利用する。



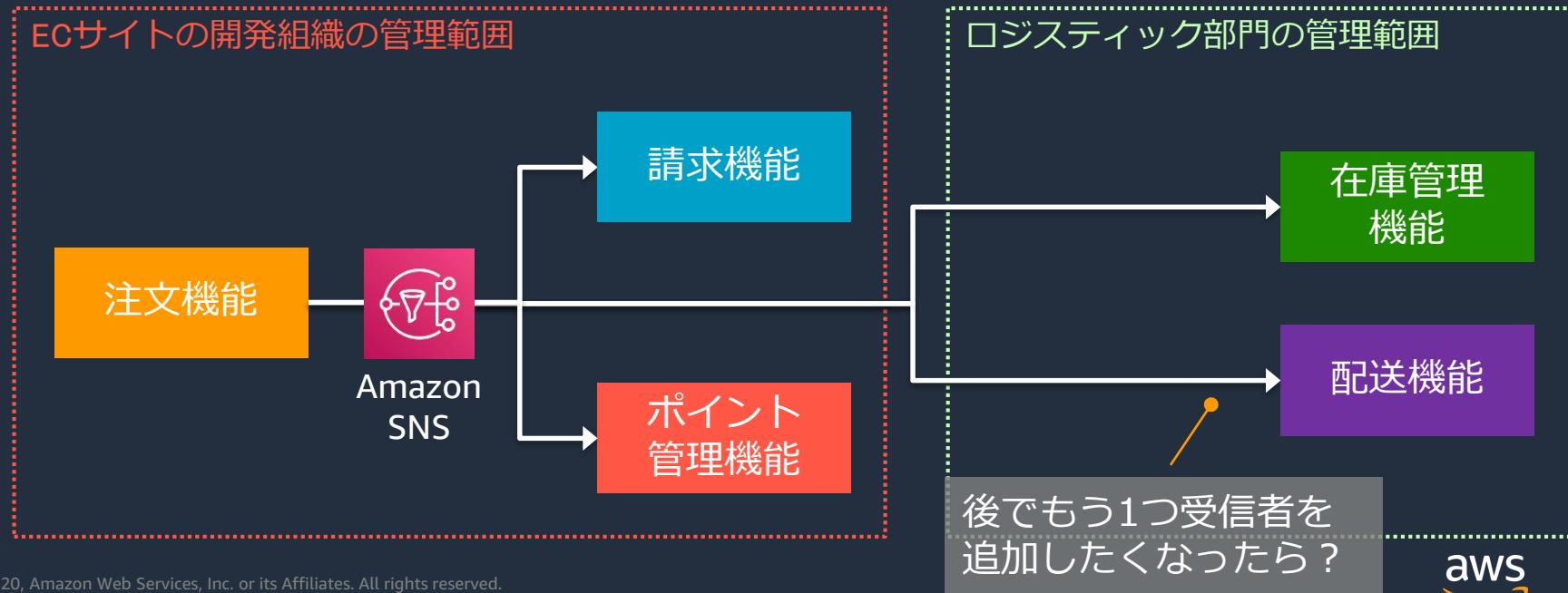
Amazon  
SNS

- Push型でメッセージを配信
- Lambda/SQS/HTTP(S)/Emailへの連携
- 数百万もの大量のイベントを低レイテンシーで配信可能

# 組織をまたがったイベント連携

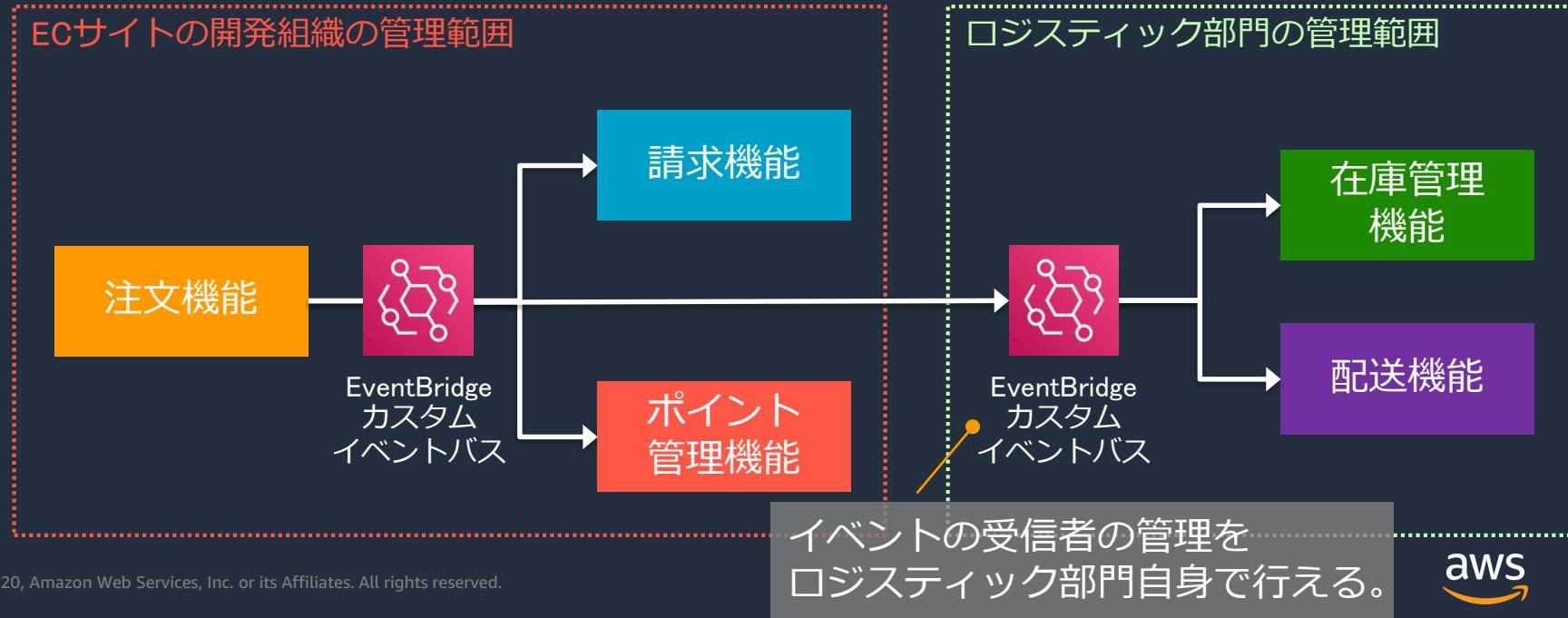
Q. ロジスティック部門から注文時のデータをリアルタイムに連携して欲しいと言われたら？

A. SNSの受信者に追加すれば実現可能。



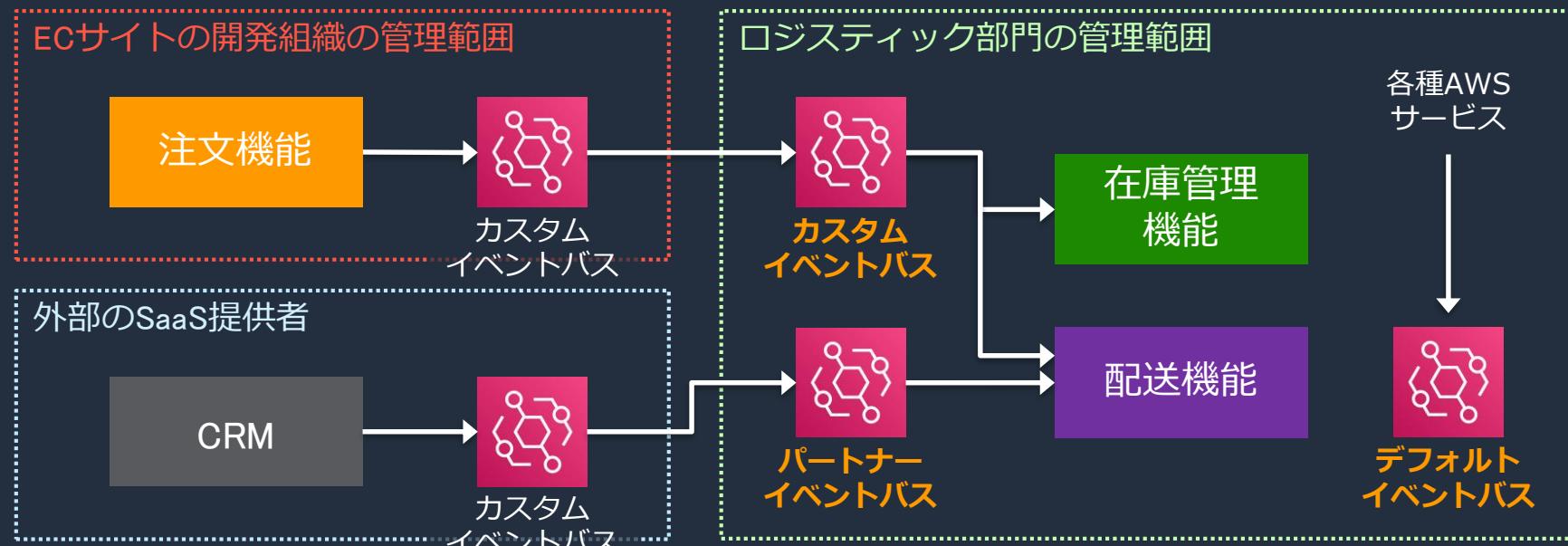
# 組織をまたがったイベント連携

イベントバスを利用して組織間の連携を行えば、他の組織でイベントの受信者が増えてても送信側は意識する必要がない。



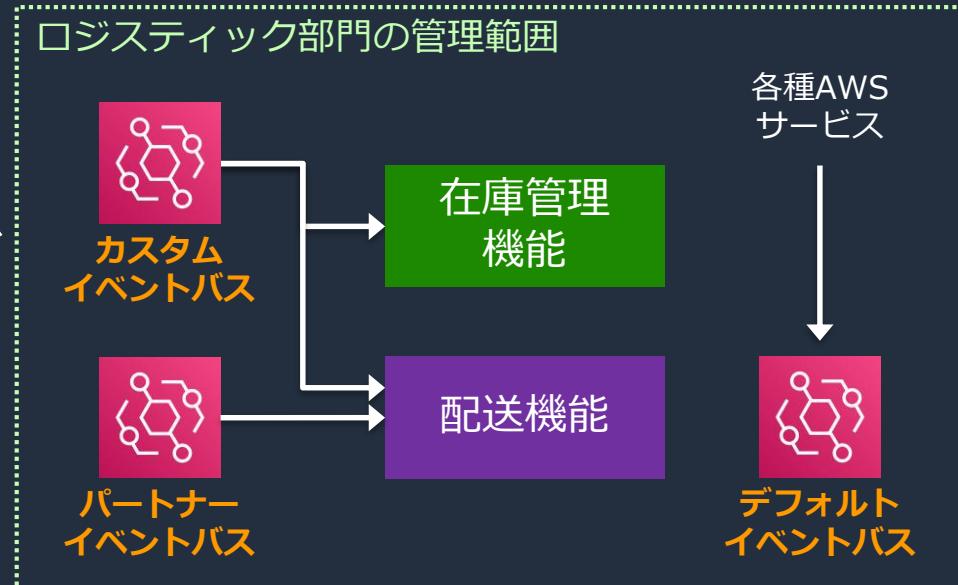
# イベントバスの使い分け

各AWSサービスからのイベントは**デフォルトイベントバス**、  
他のアプリケーションからのデータには**カスタムイベントバス**、  
SaaSからのデータには**パートナーイベントバス**を用いる



# EventBridgeとCloudWatch Eventsとの違い

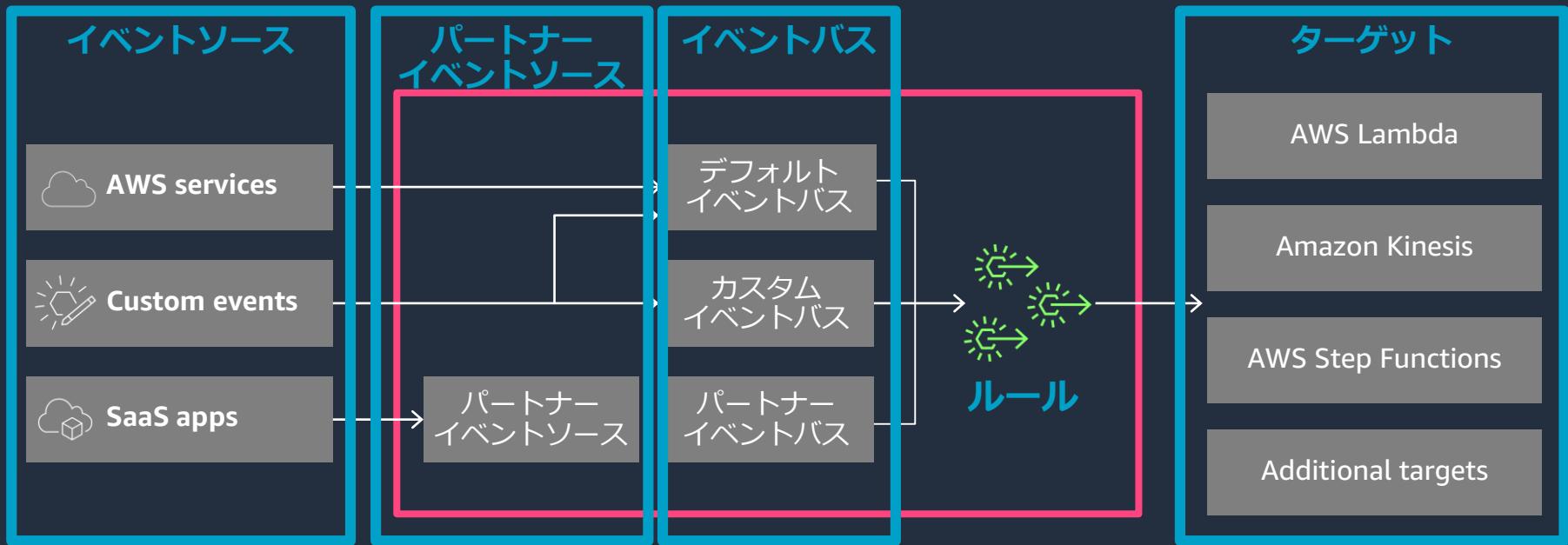
- EventBridgeは Amazon CloudWatch Eventsから派生・独立したサービスで、デフォルトイベントバスはその延長線上にある機能。
  - CloudWatch EventsのAPIやマネジメントコンソールはそのまま利用可能。
- カスタムイベントバスや SaaS連携の機能を追加し、アプリケーションのイベントをより活用していただくため、モニタリング視点の CloudWatchファミリーから独立した。



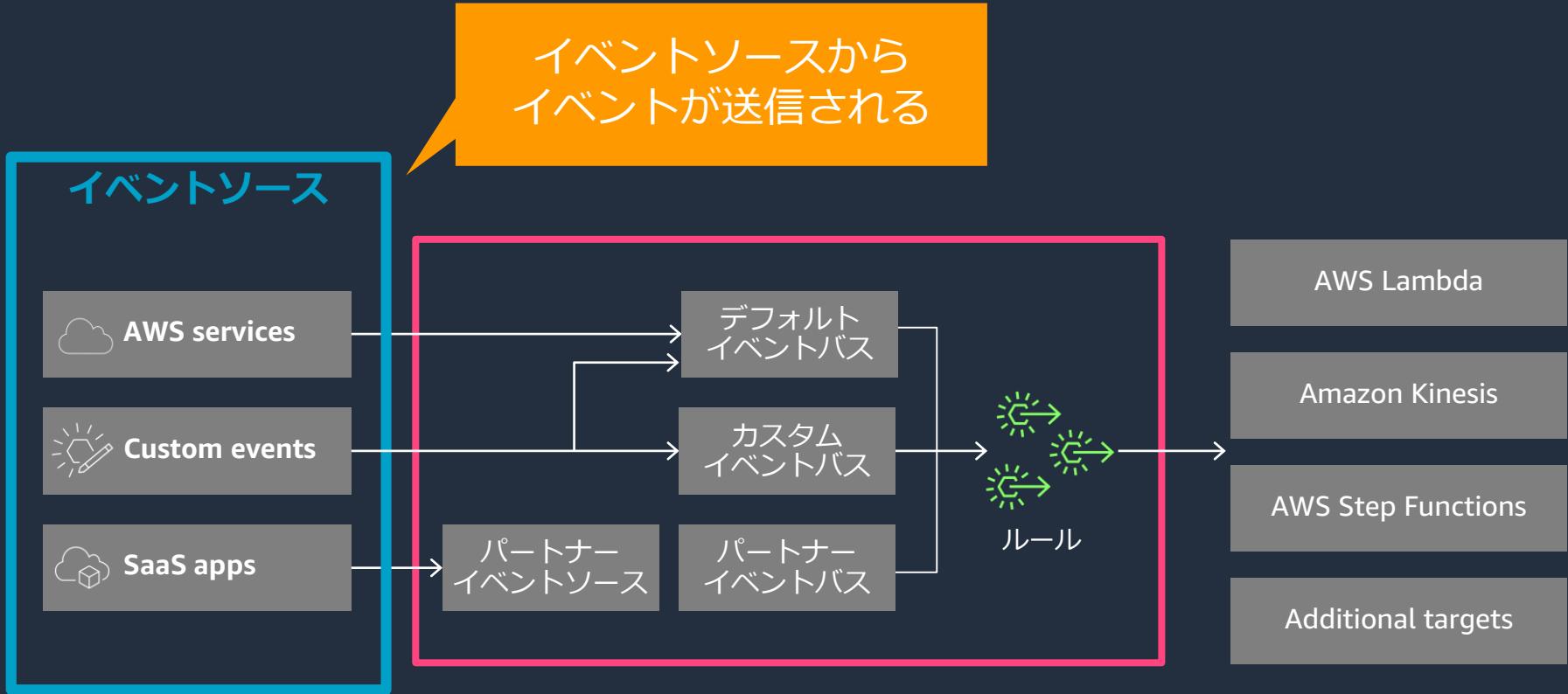
# 本日の内容

1. EventBridge とは?
  - イベントバス導入の背景
  - **アーキテクチャ**
  - 構成要素
  - アクセス制御
2. SaaS との連携
  - SaaSアプリケーションとの連携案
  - 対応しているSaaSアプリケーション
  - SaaSアプリケーションとの連携方法
  - EventBridge 活用事例
  - EventBridgeパートナーになるには
3. クロスアカウント連携
4. 直近のアップデート
5. 料金、制限
6. まとめ

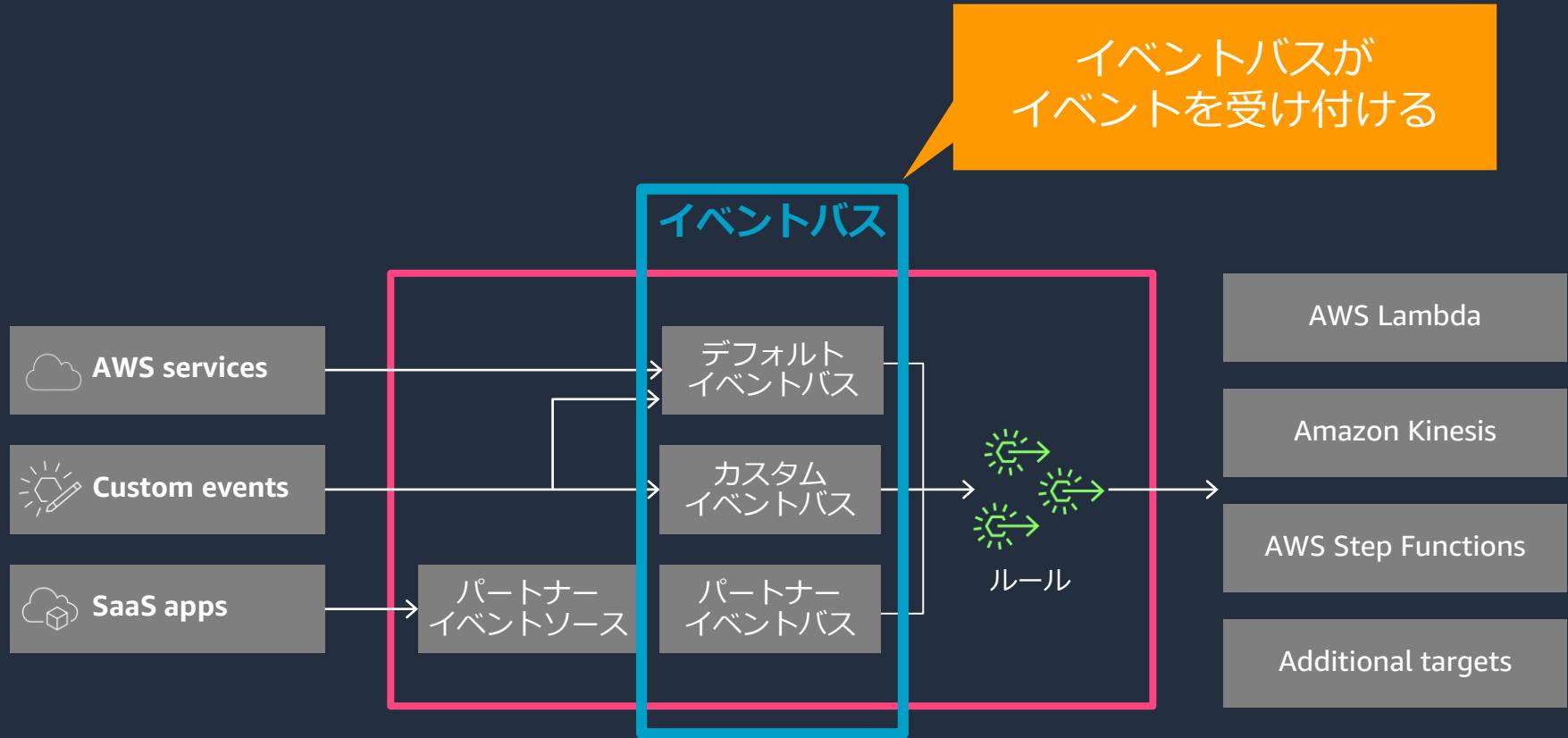
# EventBridgeのアーキテクチャ



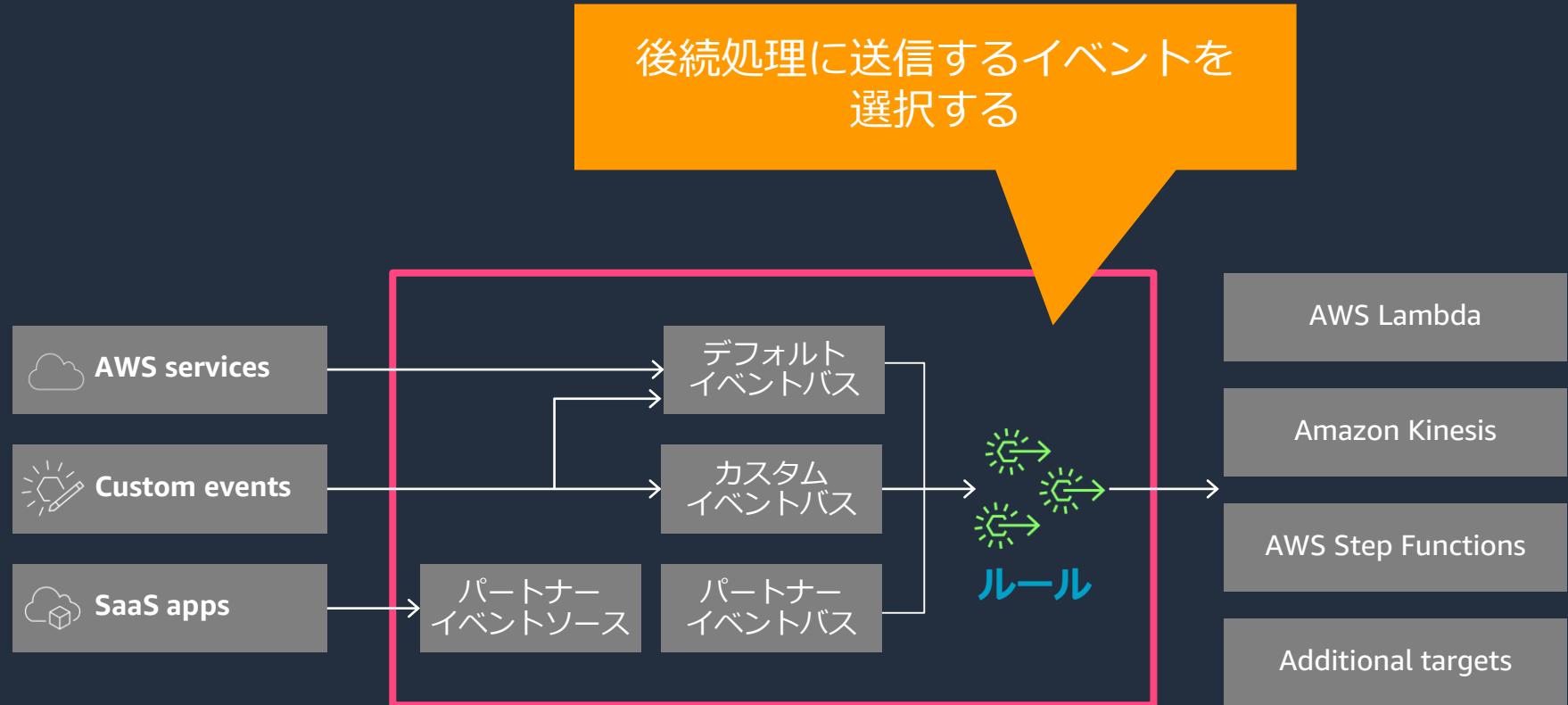
# EventBridgeのアーキテクチャ



# EventBridgeのアーキテクチャ

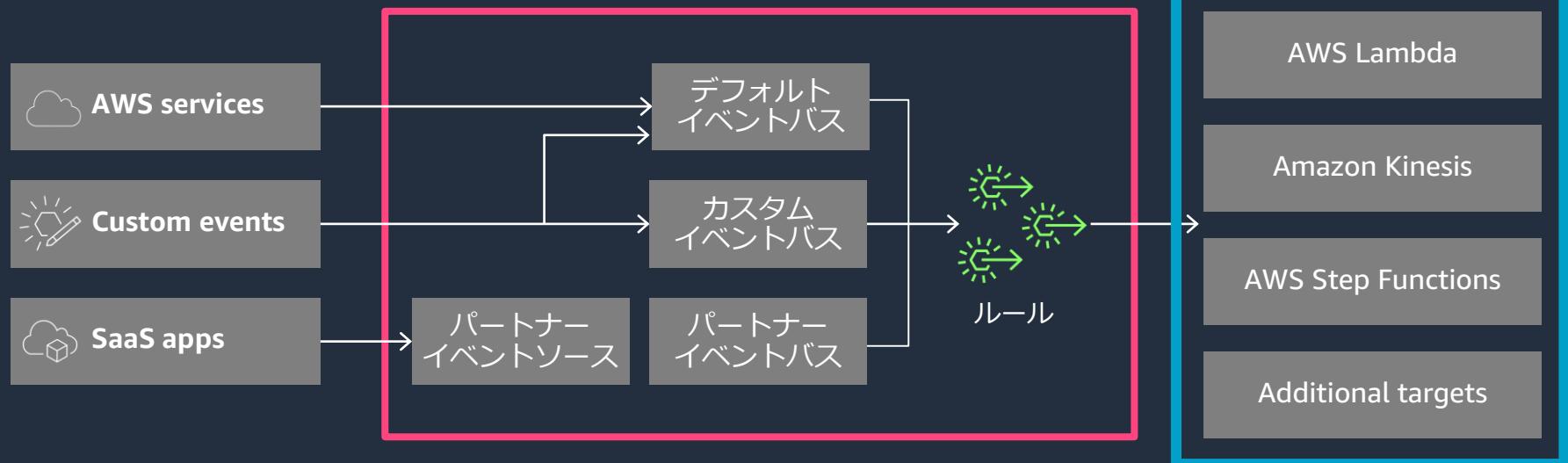


# EventBridgeのアーキテクチャ

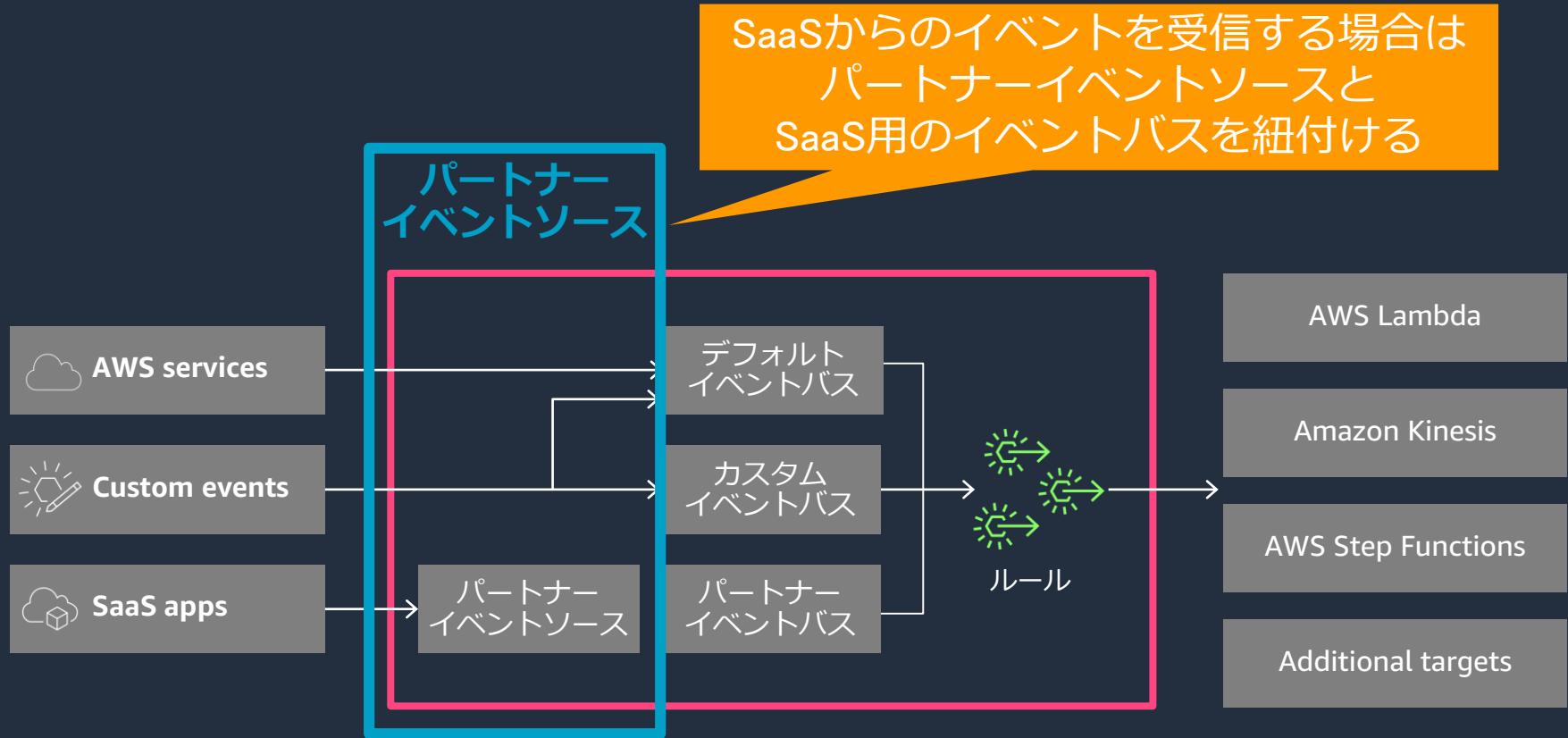


# EventBridgeのアーキテクチャ

ターゲットにイベントが送信され、各サービスでイベントが処理される



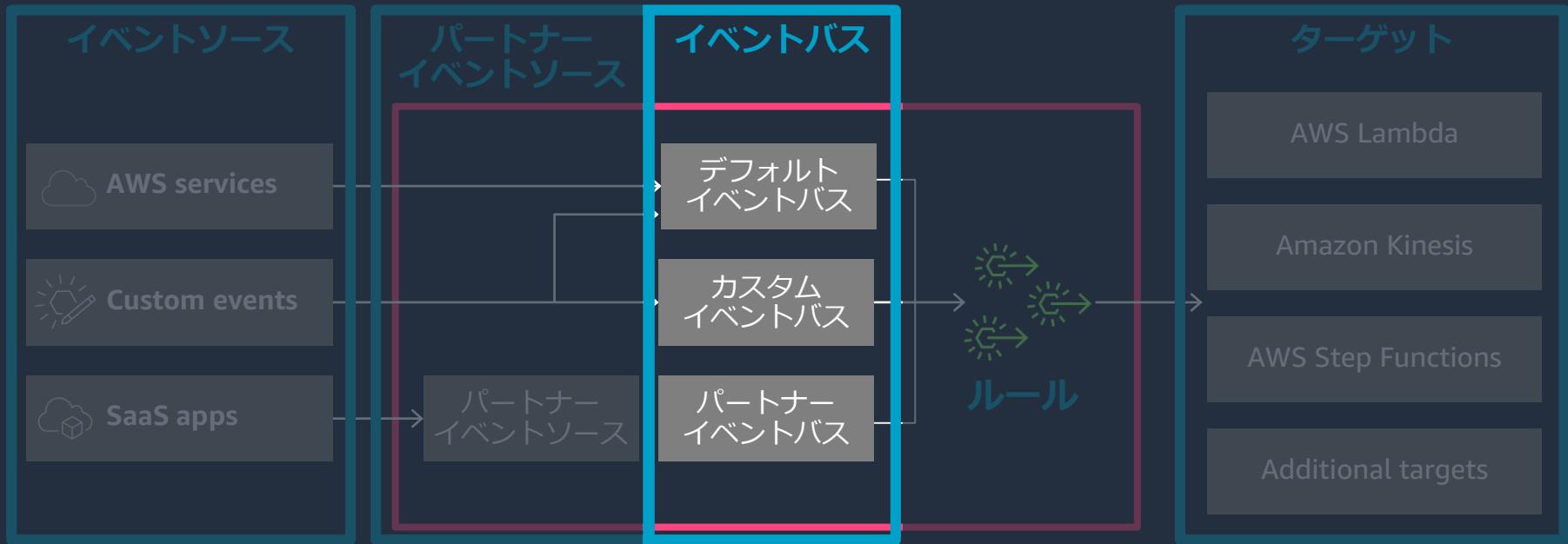
# EventBridgeのアーキテクチャ



# 本日の内容

1. EventBridge とは?
  - イベントバス導入の背景
  - アーキテクチャ
  - **構成要素**
  - アクセス制御
2. SaaS との連携
  - SaaSアプリケーションとの連携案
  - 対応しているSaaSアプリケーション
  - SaaSアプリケーションとの連携方法
  - EventBridge 活用事例
  - EventBridgeパートナーになるには
3. クロスアカウント連携
4. 直近のアップデート
5. 料金、制限
6. まとめ

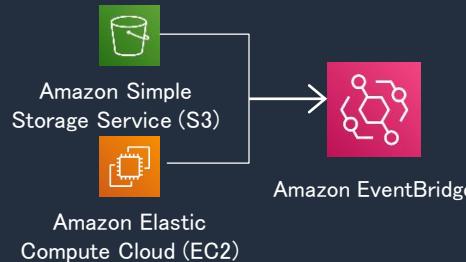
# EventBridgeのアーキテクチャ



# イベントバス

EventBridgeはイベントバスでイベントを受け取る。

## デフォルト

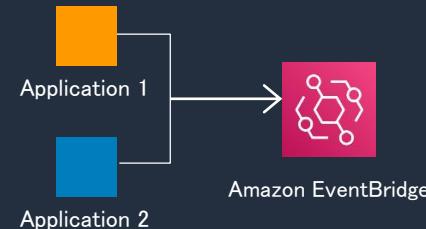


## AWSサービス

例) EC2インスタンスが停止した

AWSが発行する  
システム的なイベント

## カスタム



## 独自のアプリケーション

例) 注文機能が注文を受け付けた

お客様/SaaSが発行する  
業務的なイベント

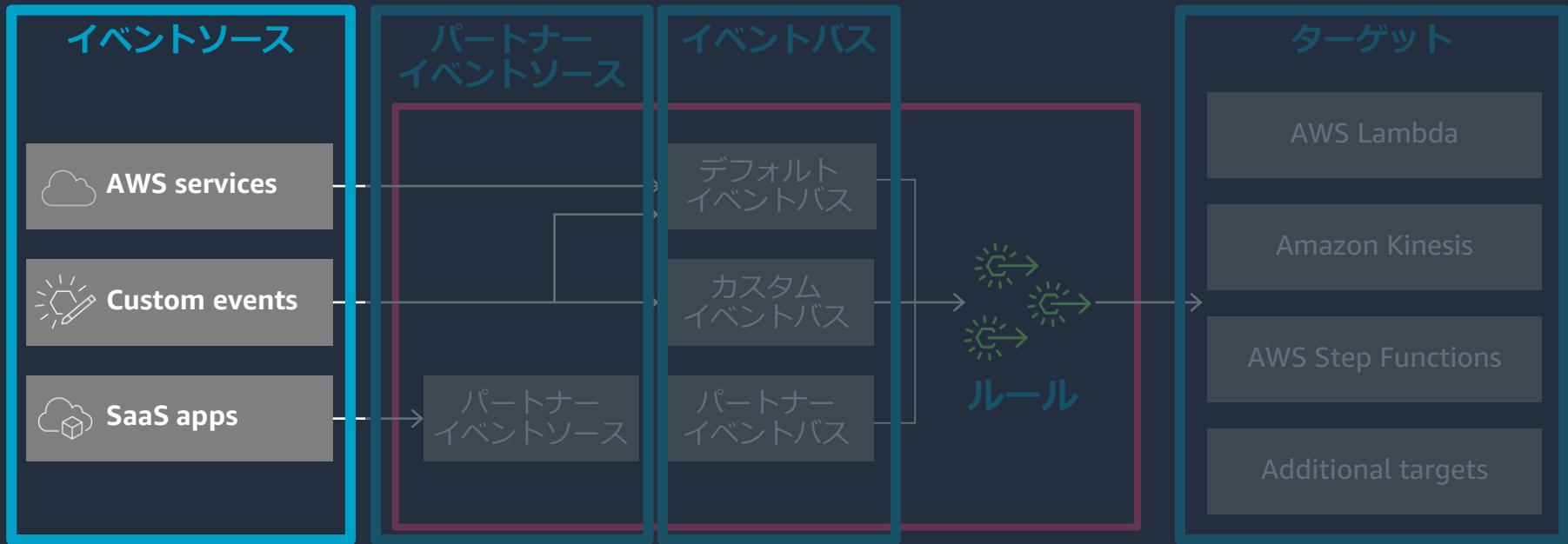
## パートナー



## SaaS

例) SaaSのCRMで顧客情報が更新された

# EventBridgeのアーキテクチャ



# イベント

- イベントは状態の変化を示し、JSONオブジェクトで表される。

	送信元	送信先の イベントバス	送信されるタイミング
AWSサービスの イベント	各AWSサービス	デフォルトの イベントバス	AWSサービスの状態が変わったとき(*1)
	各AWSサービス (CloudTrail経由)		AWSサービスのAPIを呼び出したとき(*2)
カスタムイベント	ユーザーまたは アプリケーション	送信時に指定する(*3)	EventBridgeのPutEvents APIを呼び出したとき
SaaSイベント	SaaSアプリケーション	各SaaSごとに作成さ れるパートナー イベントバス	送信元のSaaSアプリケーションに よって異なる

(\*1) イベントが送信されるサービスは下記参照。

[https://docs.aws.amazon.com/ja\\_jp/eventbridge/latest/userguide/event-types.html](https://docs.aws.amazon.com/ja_jp/eventbridge/latest/userguide/event-types.html)

(\*2) API呼び出しとイベントバスへイベントが送信されるまでにタイムラグがある。

(\*3) 未指定の場合はデフォルトイベントバスに送信される

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

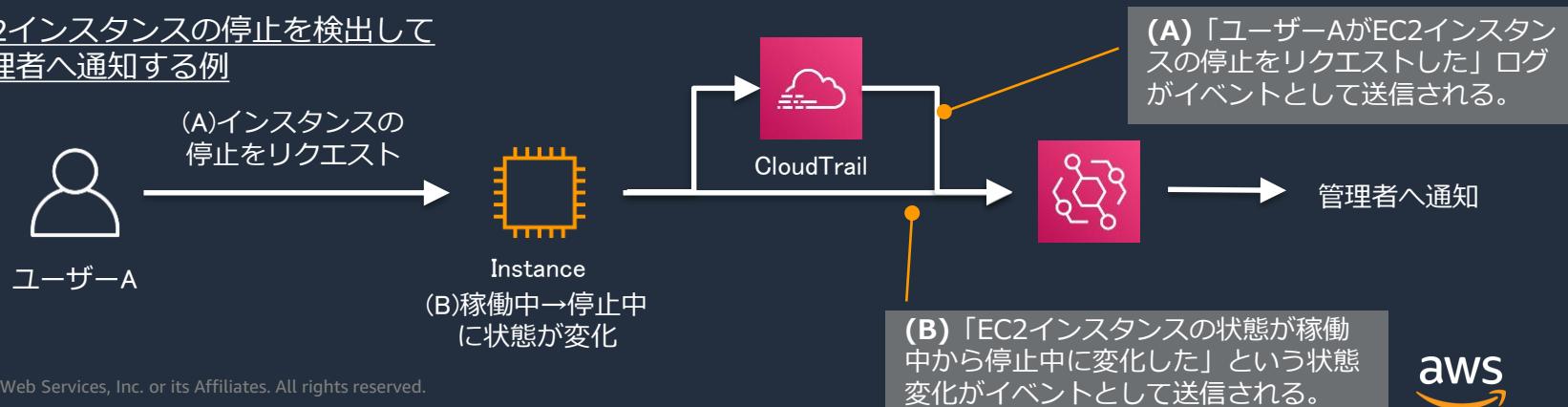


# AWSサービスが送信するイベント

- すべてデフォルトのイベントバスに送信される。
- EventBridgeで未サポートのサービスはAWSのリソースの操作ログを取得する AWS CloudTrail 経由でイベントを取得可能。

EventBridgeに直接送信されるイベント	AWSリソースの状態変化を起点にイベントが発生する。
CloudTrail経由で送信されるイベント	ユーザーのAWSリソースに対する操作（＝API呼び出し）を起点にイベントが発生する。

例) EC2インスタンスの停止を検出して管理者へ通知する例



# EventBridgeにイベントを送信するサービスと機能

- AWS Batch
- AWS CodeBuild
- AWS CodeCommit
- AWS CodeDeploy
- AWS CodePipeline
- AWS Config
- EBS
- Amazon EC2
  - Auto Scaling
  - spot instanceの中斷
  - instanceの状態変更
- Amazon ECS
- Elemental MediaConvert
- Elemental MediaPackage
- Elemental MediaStore
- Amazon EMR
- Amazon GameLift
- AWS Glue
- AWS Ground Station
- Amazon GuardDuty
- AWS Health
- AWS KMS
- Amazon Macie
- AWS OpsWorks スタック
- Amazon SageMaker
- AWS Security Hub
- AWS Server Migration Service
- AWS Systems Manager
  - 設定コンプライアンス
  - メンテナンスウィンドウ
  - パラメータストア
- AWS Step Functions
- AWS リソースのタグ変更
- AWS Trusted Advisor
- Amazon WorkSpaces
- Amazon CloudWatch Alarm
- CloudTrail
- EventBridge
  - スケジュールされたイベント

# カスタムイベント

- ユーザーのPutEvents APIの呼び出しによって送信される。
  - CLIではaws events put-eventsコマンドで送信（下部記載）
  - SDKでは各言語の実装による。
- 送信先のイベントバスは送信時に選択可能。

```
$ cat entries.json
[
  {
    "Source": "com.mycompany.myapp",
    "Detail": "{ \"key1\": \"value1\" }",
    "DetailType": "myDetailType",
    "resources": [
      "resource1"
    ],
    "EventBusName": "myEventBus", ●
  }
]

$ aws events put-events --entries file://entries.json
```

送信先のイベントバス名を指定

# SaaSイベント

- SaaSによるイベントは各SaaS専用のイベントバスに送信される。
- 送信されるイベントの種類やタイミングは、各SaaSアプリケーションの仕様による。

# イベントの構造

```
{  
  "version": "0",  
  "id": "adeacade-c34c-ce58-c4a0-74f106398c4e",  
  "account": "123456789012",  
  "region": "us-east-1",  
  "time": "2019-12-02T21:46:19Z",  
  "source": "order-service",  
  "detail-type": "New Order",  
  "resources": [],  
  "detail": {  
    "orderId": "cfb2ae566f9b",  
    "customerId": "C12345",  
    ...  
  }  
}
```



イベントの  
メタデータ

ペイロード

# イベントの構造

```
{  
  "version": "0",  
  "id": "adeacade-c34c-ce58-c4a0-74f106398c4e",  
  "account": "123456789012",  
  "region": "us-east-1",  
  "time": "2019-12-02T21:46:19Z",  
  "source": "order-service",  
  "detail-type": "New Order",  
  "resources": [],  
  "detail": {  
    "orderId": "cfb2ae566f9b",  
    "customerId": "C12345",  
    ...  
  }  
}
```

## 管理用データ

EventBridgeが自動設定。  
ユーザーによる変更不可。

# イベントの構造

```
{  
  "version": "0",  
  "id": "adeacade-c34c-ce58-c4a0-74f106398c4e",  
  "account": "123456789012",  
  "region": "us-east-1",  
  "time": "2019-12-02T21:46:19Z",  
  "source": "order-service",  
  "detail-type": "New Order",  
  "resources": [],  
  "detail": {  
    "orderId": "cfb2ae566f9b",  
    "customerId": "C12345",  
    ...  
  }  
}
```

## タイムスタンプ

明示的な指定が可能。

省略時にはAPIの呼び出し時刻が付与される

# イベントの構造

```
{  
  "version": "0",  
  "id": "adeacade-c34c-ce58-c4a0-74f106398c4e",  
  "account": "123456789012",  
  "region": "us-east-1",  
  "time": "2019-12-02T21:46:19Z",  
  "source": "order-service",  
  "detail-type": "New Order",  
  "resources": [],  
  "detail": {  
    "orderId": "cfb2ae566f9b",  
    "customerId": "C12345",  
    ...  
  }  
}
```

## イベントの送信元

送信元のアプリケーション／サービスを表す。  
AWSサービスのイベントの場合、  
aws.ec2のようなサービス名が入る。

"aws."のprefixはAWSによって予約されているため使用不可。

# イベントの構造

```
{  
  "version": "0",  
  "id": "adeacade-c34c-ce58-c4a0-74f106398c4e",  
  "account": "123456789012",  
  "region": "us-east-1",  
  "time": "2019-12-02T21:46:19Z",  
  "source": "order-service",  
  "detail-type": "New Order",  
  "resources": [],  
  "detail": {  
    "orderId": "cfb2ae566f9b",  
    "customerId": "C12345",  
    ...  
  }  
}
```

## イベントの種類

イベントの種類を表す。

sourceとdetail-typeの組み合わせで  
detailのスキーマが決定する。

# イベントの構造

```
{  
  "version": "0",  
  "id": "adeacade-c34c-ce58-c4a0-74f106398c4e",  
  "account": "123456789012",  
  "region": "us-east-1",  
  "time": "2019-12-02T21:46:19Z",  
  "source": "order-service",  
  "detail-type": "New Order",  
  "resources": [],  
  "detail": {  
    "orderId": "cfb2ae566f9b",  
    "customerId": "C12345",  
    ...  
  }  
}
```

## リソース

イベントが関連するリソース。  
AWSサービスが送信するイベントの場合、  
ARN(Amazon Resource Name)の配列が  
含まれる。

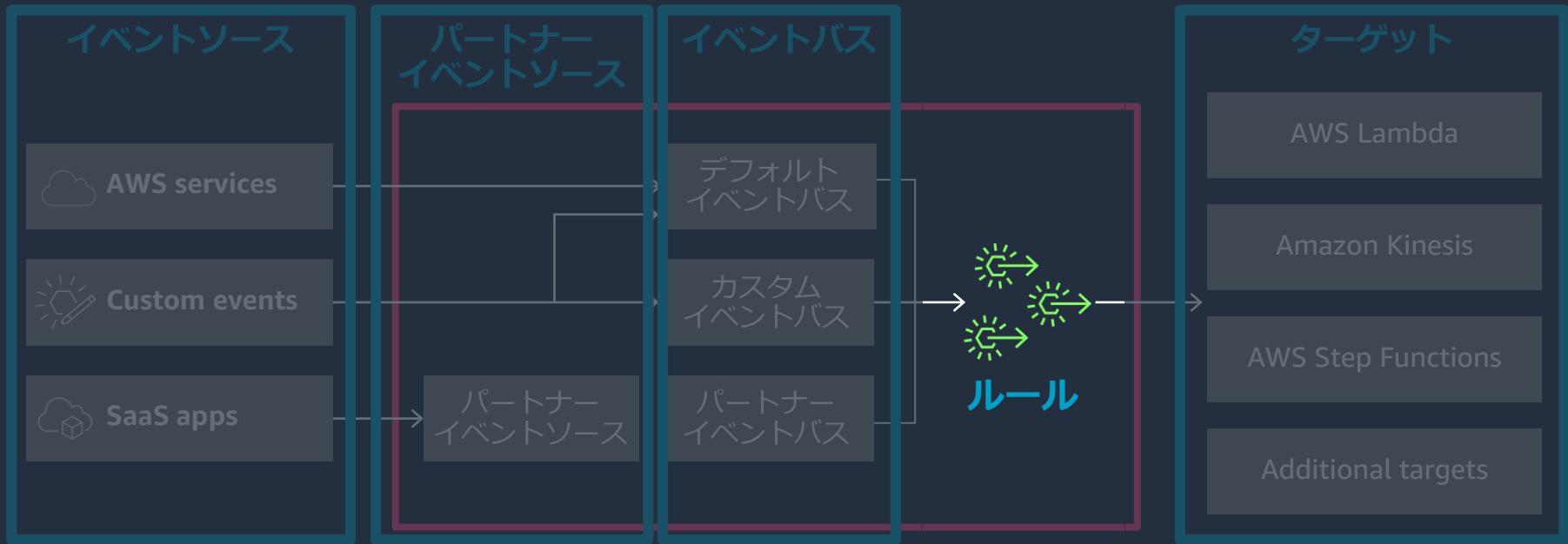
# イベントの構造

```
{  
  "version": "0",  
  "id": "adeacade-c34c-ce58-c4a0-74f106398c4e",  
  "account": "123456789012",  
  "region": "us-east-1",  
  "time": "2019-12-02T21:46:19Z",  
  "source": "order-service",  
  "detail-type": "New Order",  
  "resources": [],  
  "detail": {  
    "orderId": "cfb2ae566f9b",  
    "customerId": "C12345",  
    ...  
  }  
}
```

## 詳細

イベントの詳細、内容を表す。

# EventBridgeのアーキテクチャ



# ルール

- イベントバスで受信したイベントのうちどのイベントをターゲットに送信するかを定義する規則。

どんなイベントを送信するか？

- イベントパターンによる定義
- スケジュール式による定義

どのイベントバスを対象とするか？

どのターゲットに送信するか？

The screenshot shows the AWS Lambda Rule configuration interface. It consists of three main sections:

- パターンを定義**: A step where you can define the event pattern or schedule. It includes two options:
  - イベントパターン 情報**: Create a pattern matching the event.
  - スケジュール 情報**: Trigger the target based on a schedule.
- イベントバスを選択**: A step where you select the event bus for the rule. It includes:
  - AWS のデフォルトのイベントバス (selected)
  - カスタムイベントバスまたはパートナーイベントバスand a checked checkbox for "選択したイベントバスでルールを有効にする".
- ターゲットを選択**: A step where you select the target for the event. It includes:
  - ターゲット (Lambda 関数 selected)
  - 機能 (Function selection dropdown)
  - 操作 buttons: 削除 (Delete), Lambda 関数 (Lambda Function), 機能 (Function), バージョン/エイリアスを設定 (Version/Alias), 入力の設定 (Input Settings), and ターゲットの追加 (Add Target).

# ルールのイベントパターン

- イベント同様のJSONオブジェクト形式で記述する。
  - AWSサービスのイベントで代表的なものは定義済み。マネジメントコンソールから選択するだけで利用可能。
  - テスト用APIを提供しており、CLIでイベントパターンのテストが可能。

```
$ aws events test-event-pattern ¥  
--event <イベントのJSON文字列> ¥  
--event-pattern <イベントパターンのJSON文字列>
```

- イベントパターンがイベントとマッチする条件
  - パターンに指定されているすべてのフィールド名を含む。
  - 入れ子構造を含む場合は、同一階層のすべてのフィールド名を含む。
  - パターンで指定されていないフィールドは無視される。
  - 値は配列形式で記述し、配列の要素のいずれかと一致する。
  - マッチングは大文字小文字を区別して行われる。
  - 数字は文字列表現レベルでマッチングされる。 (300, 300.0, 3.0e2は全て区別)

# イベントパターンがイベントとマッチする例

```
{  
  "version": "0",  
  "id": "adeacade-c34c-ce58-(trim)",  
  "account": "123456789012",  
  "region": "us-east-1",  
  "time": "2019-12-02T21:46:19Z",  
  "source": "aws.ecr",  
  "detail-type": "ECR Image Action",  
  "resources": [],  
  "detail": {  
    "result": "SUCCESS",  
    "repository-name": "app",  
    "image-digest": "sha256:(trim)",  
    "action-type": "PUSH",  
    "image-tag": "latest"  
  }  
}
```

```
{  
  "source": ["aws.ecr"],  
  "detail-type": ["ECR Image Action"],  
  "detail": {  
    "result": ["SUCCESS"],  
    "repository-name": ["app"],  
    "action-type": ["PUSH"],  
    "image-tag": ["latest"]  
  }  
}
```

イベントパターンのすべてのフィールド名を含む

入れ子構造を含む場合、同一階層のフィールド名を含む

値は配列形式で記述し、配列の要素のいずれかと一致する。  
大文字小文字も含めてフィールドの値が一致する。

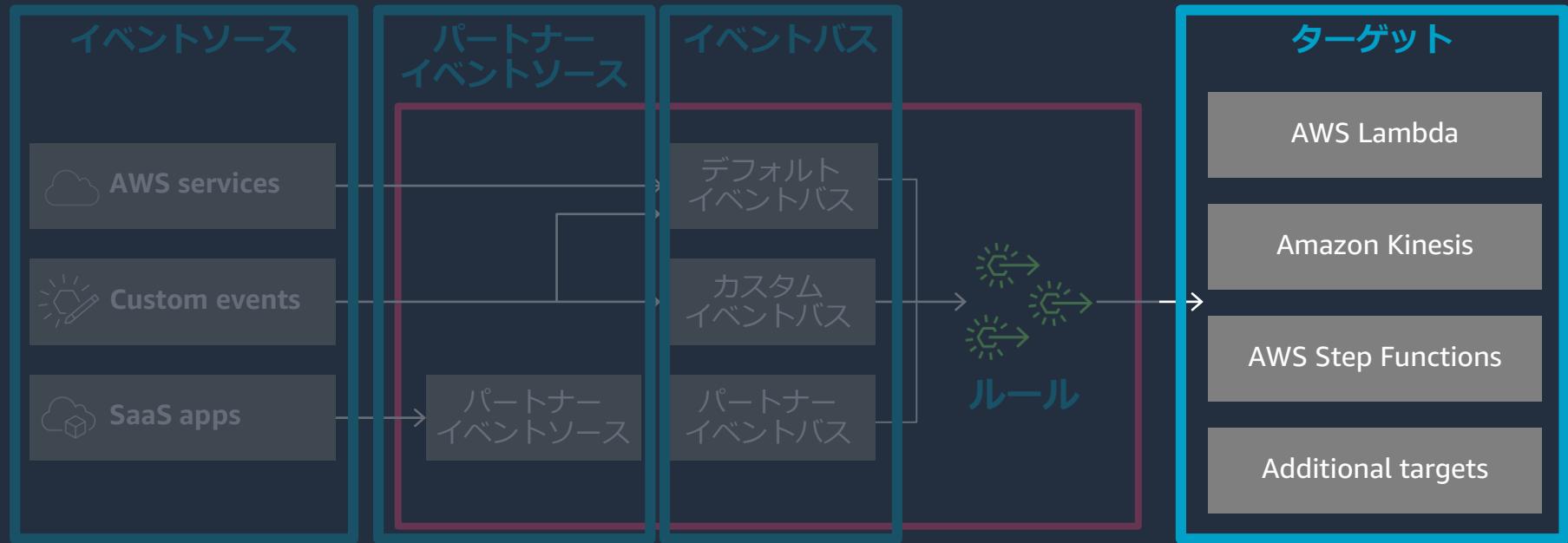
# ルールのスケジュール式

スケジュールに基づいて自己をトリガーするイベントを作成できる。  
注意：タイムゾーンはUTCで、最小精度は分での実行が可能。

	cron式	rate式
説明	cron形式で記述する。	実行頻度を指定する。
設定値の形式	cron([分] [時間] [日] [月] [曜日] [年])	rate([値] [単位])
設定例	毎日午後1時(UTC)に実行する cron(0 13 * * ? *)	5分ごとに実行する rate(5 minutes)

13:00:00 から  
13:01:00 までに実行される

# EventBridgeのアーキテクチャ



# ターゲット

- ・ イベントの送信先となり、イベントを処理するもの。
- ・ 1つのルールあたり5つまでのターゲットに送信可能。
  - ・ ターゲットのリソースは同じリージョンに存在する必要がある。
- ・ 送信先のサービスによってパラメータが異なる。
- ・ 一部のサービスでは受信したイベントのJSONを抽出したり、メッセージを変換して送信可能。

ターゲットを選択

イベントがイベントパターンに一致するか、スケジュールがトリガーされたときに(1 ルール当たり 5 個のターゲットに制限)呼び出すターゲットを選択します。

ターゲット

イベントがイベントパターンに一致するか、スケジュールがトリガーされたときに(1 ルール当たり 5 個のターゲットに制限)呼び出すターゲットを選択します。

Lambda 関数

機能

関数を選択

▶ バージョン/エイリアスを設定

▼ 入力の設定

一致したイベント [情報](#)

一致したイベントの一部 [情報](#)

定数 (JSON テキスト) [情報](#)

入力トランスマッパー [情報](#)

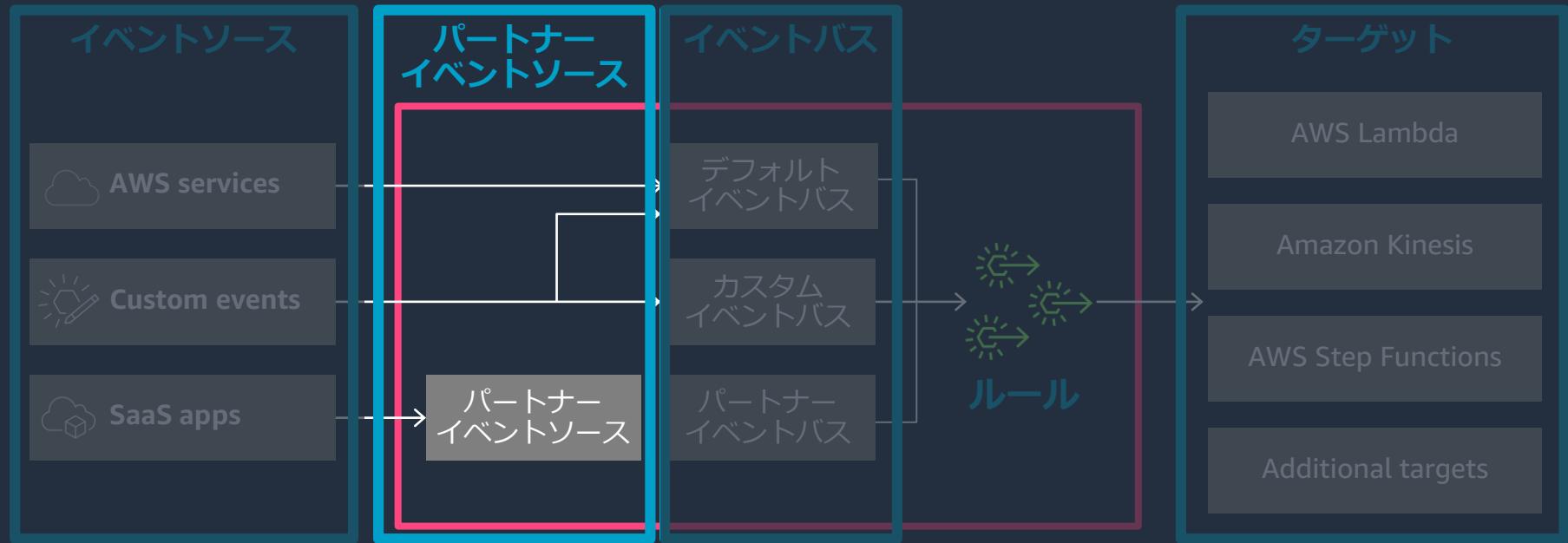
ターゲットの追加

削除

# ターゲットとして指定できるもの

- Lambda関数
- EC2インスタンス
- Kinesis Data Streamsのストリーム
- Kinesis Data Firehoseの配信ストリーム
- CloudWatch Logsのロググループ
- ECSタスク
- Systems Manager Run Command
- Systems Manager Automation
- AWS Batchジョブ
- AWS Step Functionsステートマシン
- AWS CodePipelineのパイプライン
- AWS CodeBuildプロジェクト
- Inspector の評価テンプレート
- SNS トピック
- SQS キュー
- 組み込みターゲット
  - EC2 CreateSnapshot API call
  - EC2 RebootInstances API call
  - EC2 StopInstances API call
  - EC2 TerminateInstances API call
- 別のAWS アカウントのイベントバス  
(同一アカウント間の送信やイベントの折返しは不可)

# EventBridgeのアーキテクチャ



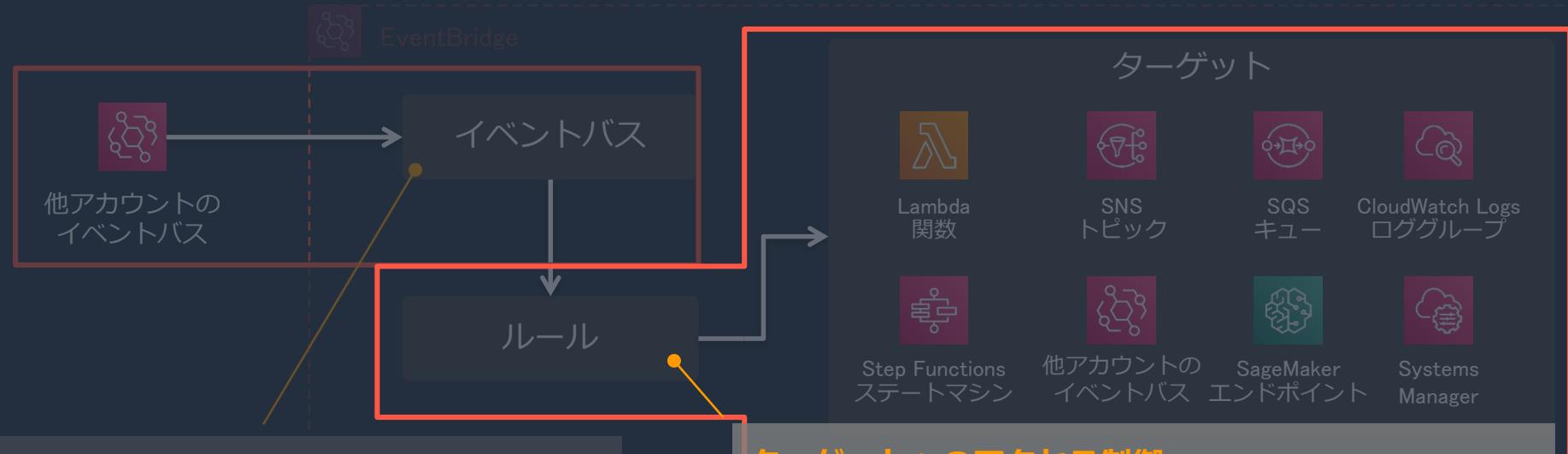
# パートナーイベントソース

- SaaSアプリケーションからのイベント送信を受け付けるためのリソース。
- 各SaaSアプリケーションの管理画面から作成でき、イベントバスにひもづけることで、イベントの連携が開始する。

# 本日の内容

1. EventBridge とは?
  - イベントバス導入の背景
  - アーキテクチャ
  - 構成要素
  - **アクセス制御**
2. SaaS との連携
  - SaaSアプリケーションとの連携案
  - 対応しているSaaSアプリケーション
  - SaaSアプリケーションとの連携方法
  - EventBridge 活用事例
  - EventBridgeパートナーになるには
3. クロスアカウント連携
4. 直近のアップデート
5. 料金、制限
6. まとめ

# アクセス制御の違い



## イベントバスのアクセス制御

他のアカウント、または、AWS Organizationsにおける同一の組織からのイベント送信をイベントバスのポリシーで許可する。

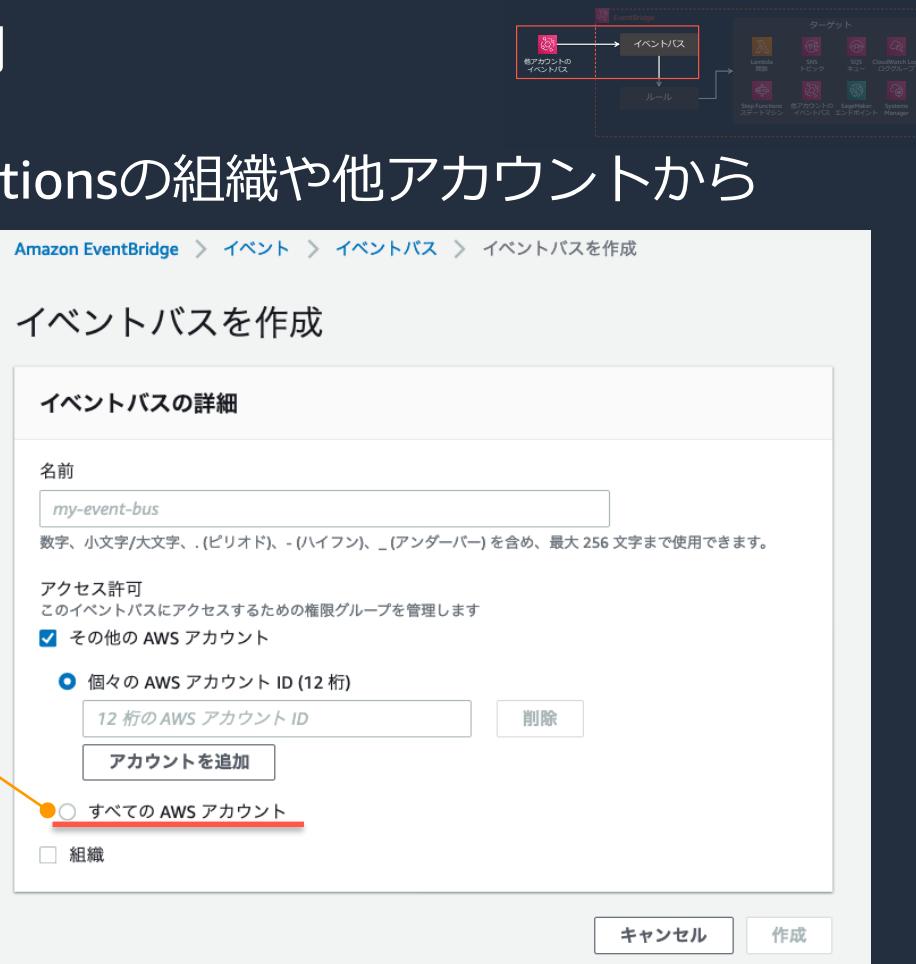
## ターゲットへのアクセス制御

- AWS Lambda, Amazon SNS, Amazon Simple Queue Service(SQS), Amazon CloudWatch Logsはの4つのサービスのリソースは、リソースのポリシーで許可する。
- それ以外のサービスはルールにIAMロールを割り当て、IAMポリシーで許可する。

# イベントバスへのアクセス制御

- ・ イベントバスごとにAWS Organizationsの組織や他アカウントからのアクセスを許可できる。
- ・ 1つのAWSアカウント内でのみ利用する場合は設定不要。

「すべてのAWSアカウント」を選択すると、誰でもイベントが送信できるようになる。  
外部に公開してイベントを送信してもらうといった、特定のユースケース以外では使用しない。

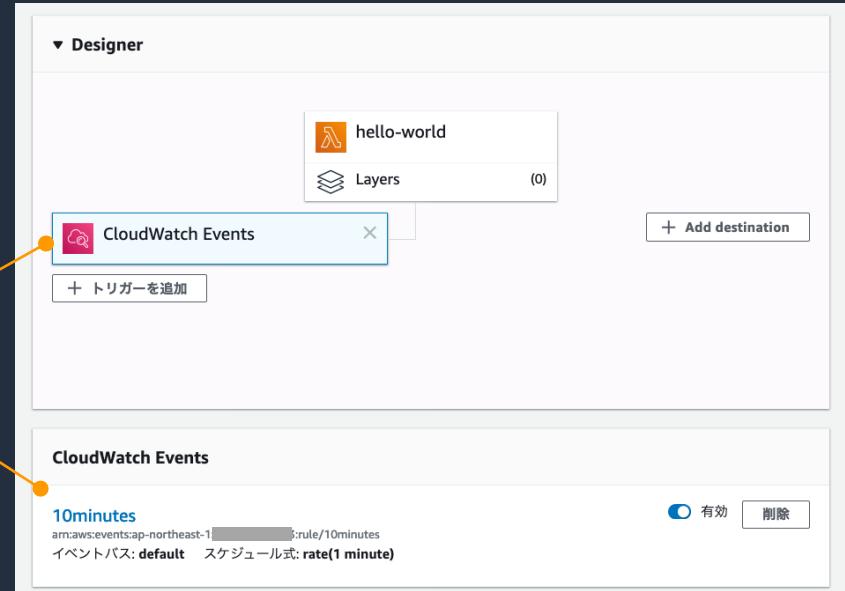


ターゲットへのアクセス制御

## リソースベースポリシー (=ターゲット側で制御)

- LambdaはLambda関数ポリシー、 SNSはトピックポリシー、 SQSはキューポリシーにアクセス許可を記述する。
  - CloudWatch LogsはEventBridgeからのイベント送信がデフォルトで許可されているため利用者による設定は不要
- マネジメントコンソール上でターゲットを追加した場合は自動で権限が付与される。

例) Lambdaのマネジメントコンソール正しく権限が付与されているときにサービスやルールが表示される。



ターゲットへのアクセス制御

# リソースベースポリシー (=ターゲット側で制御)

Lambda関数ポリシーのサンプル

```
{  
    "Version": "2012-10-17",  
    "Id": "default",  
    "Statement": [  
        {  
            "Sid": "invoke_10minutes",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "events.amazonaws.com"  
            },  
            "Action": "lambda:InvokeFunction",  
            "Resource": "arn:aws:lambda:us-east-1:123456789012:function:hello-world",  
            "Condition": {  
                "ArnLike": {  
                    "AWS:SourceArn": "arn:aws:events:us-east-  
1:123456789012:rule/10minutes"  
                }  
            }  
        }  
    ]  
}
```

①EventBridgeからの  
②Lambda関数の実行を  
③hello-worldという関数に限り  
④10minutesというルールから  
呼び出す場合のみ  
⑤許可する



## ターゲットへのアクセス制御

**IAMポリシー (= IAM側で制御)**

- Lambda, SNS, SQS, CloudWatch Logs以外のサービスをターゲットに指定する場合、IAMロールを割り当ててIAMポリシーで許可する。
  - マネジメントコンソールからターゲットの追加するときには、必要なサービスの権限を持つIAMロールとIAMポリシーを作成可能。



**ターゲットを選択**

イベントがイベントパターンに一致するか、スケジュールがトリガーされたときに (1 ルール当たり 5 個のターゲットに制限)呼び出すターゲットを選択します。

**ターゲット**

イベントがイベントパターンに一致するか、スケジュールがトリガーされたときに (1 ルール当たり 5 個のターゲットに制限)呼び出すターゲットを選択します。 削除

▼

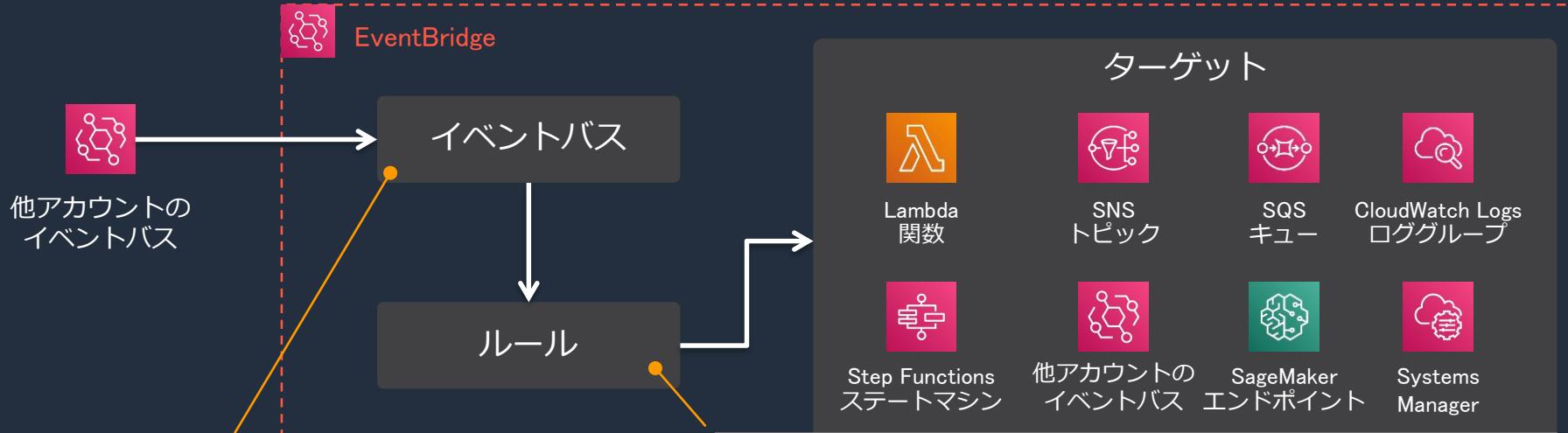
**イベントバス**

EventBridge には、上記の AWS アカウントのデフォルトイベントバスにイベントを送信するためのアクセス許可が必要です。続行すると、許可したことになります。

この特定のリソースに対して新しいロールを作成する

既存のロールの使用

# アクセス制御の違い



## イベントバスのアクセス制御

他のアカウント、または、AWS Organizationsにおける同一の組織からのイベント送信をイベントバスのポリシーで許可する。

## ターゲットへのアクセス制御

- AWS Lambda, Amazon SNS, Amazon Simple Queue Service(SQS), Amazon CloudWatch Logsはの4つのサービスのリソースは、リソースのポリシーで許可する。
- それ以外のサービスはルールにIAMロールを割り当て、IAMポリシーで許可する。

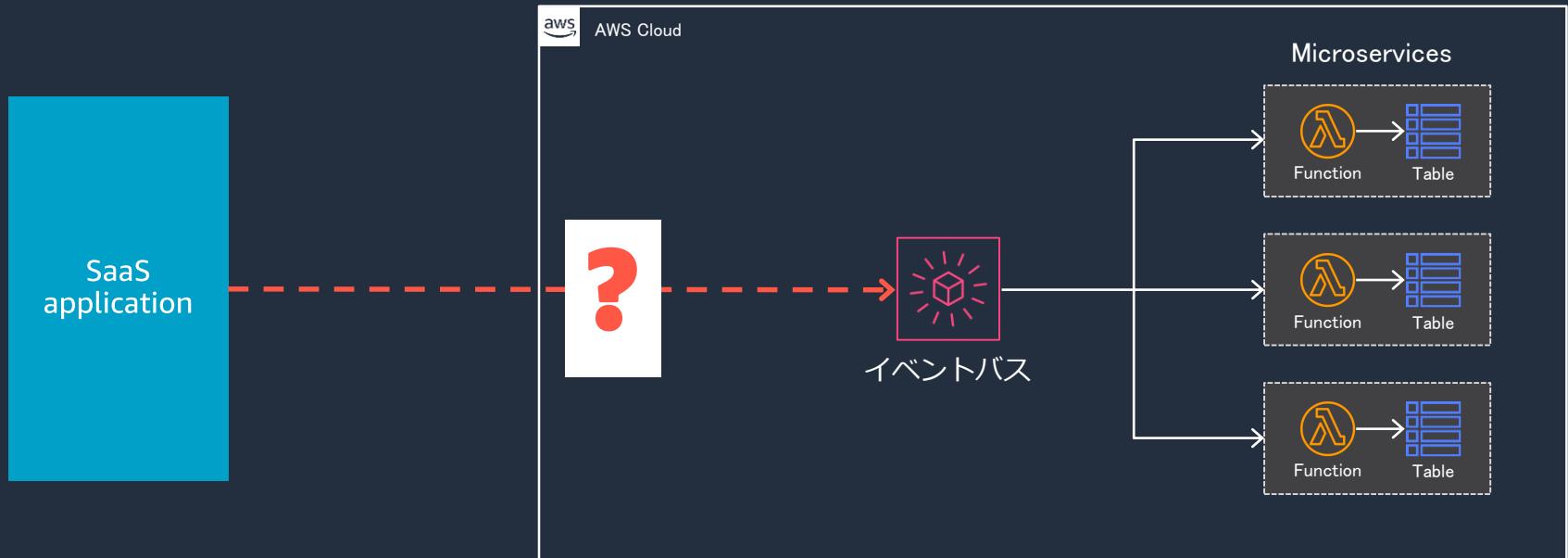
# 本日の内容

1. EventBridge とは?
  - イベントバス導入の背景
  - アーキテクチャ
  - 構成要素
  - アクセス制御
3. クロスアカウント連携
4. 直近のアップデート
5. 料金、制限
6. まとめ

## 2. SaaS との連携

- SaaSアプリケーションとの連携案
- 対応しているSaaSアプリケーション
- SaaSアプリケーションとの連携方法
- EventBridge 活用事例
- EventBridgeパートナーになるには

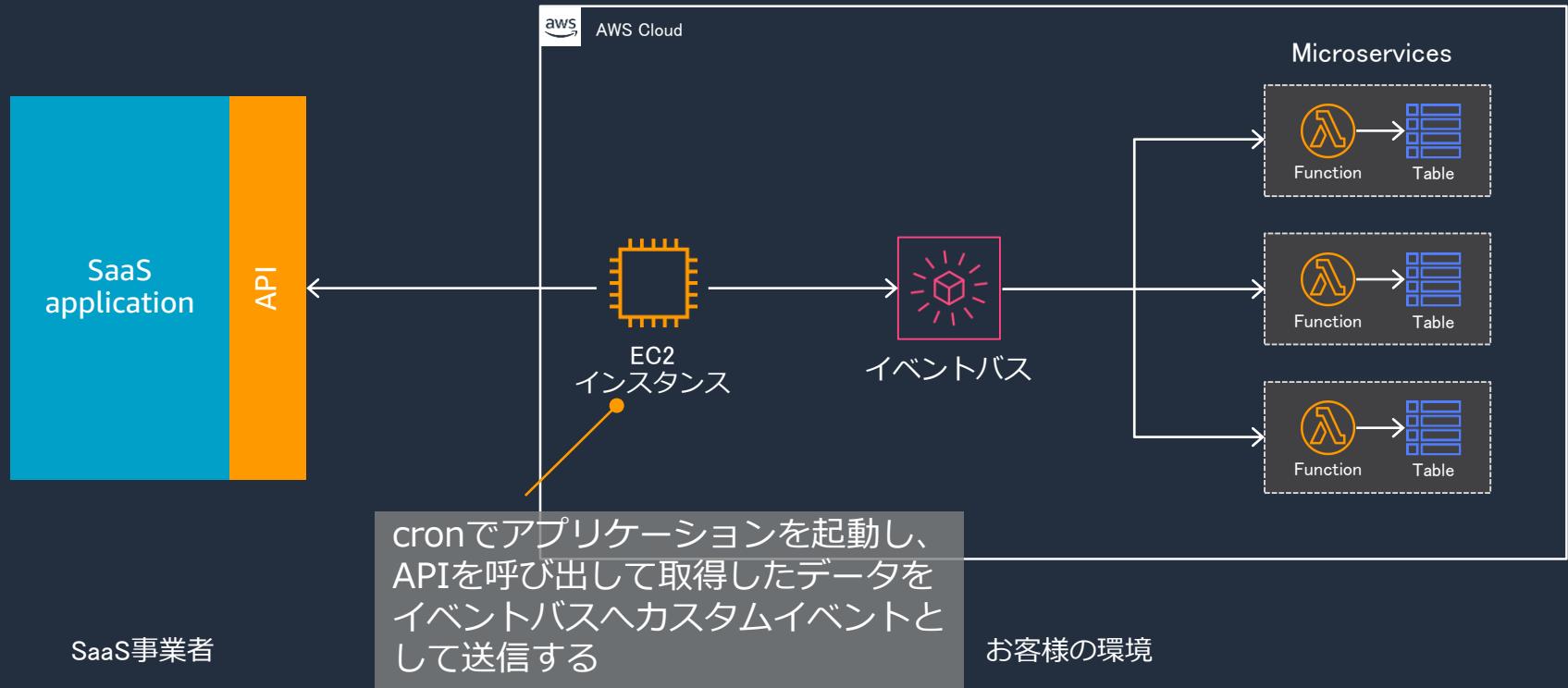
# SaaSアプリケーションとAWSを連携するには？



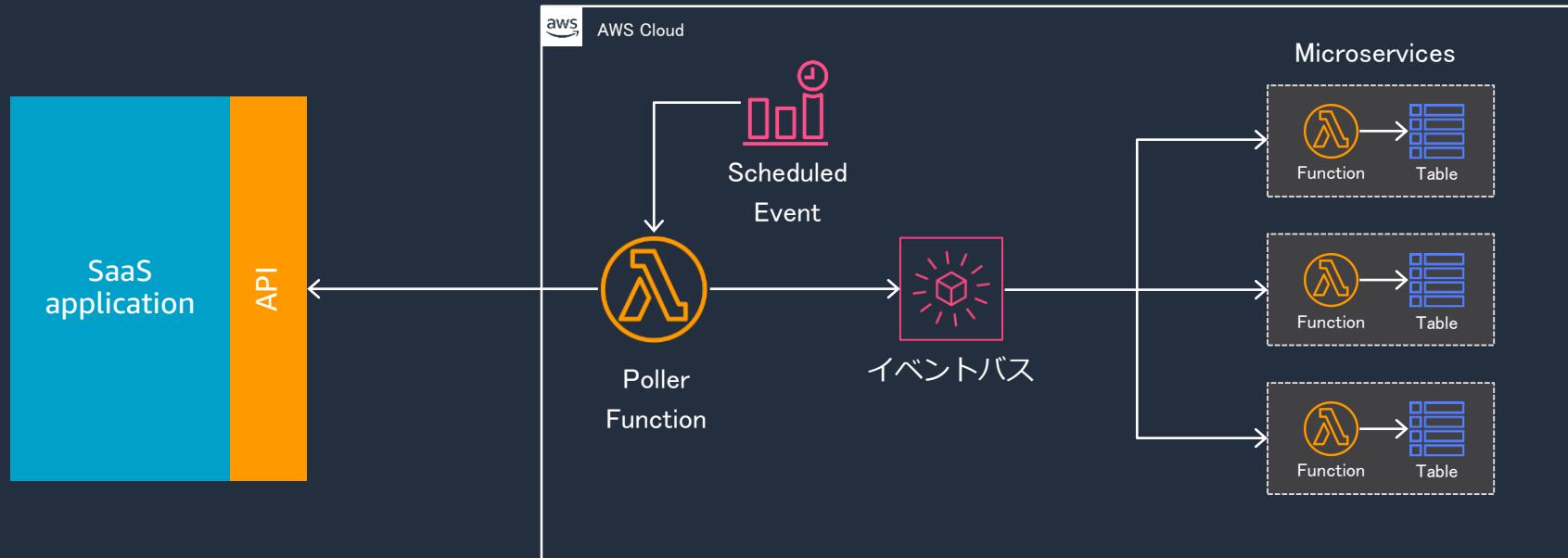
SaaS事業者

お客様の環境

# 案1：SaaSのAPIをポーリングする（EC2）



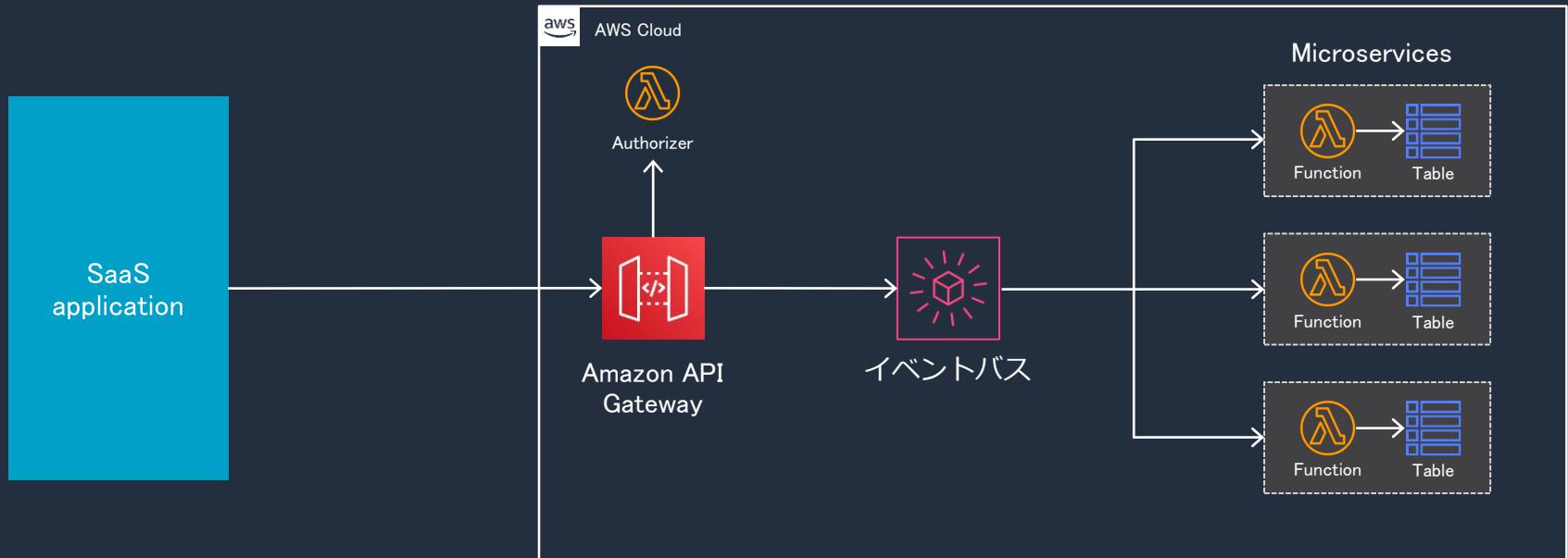
# 案2：SaaSのAPIをポーリングする（Lambda）



SaaS事業者

お客様の環境

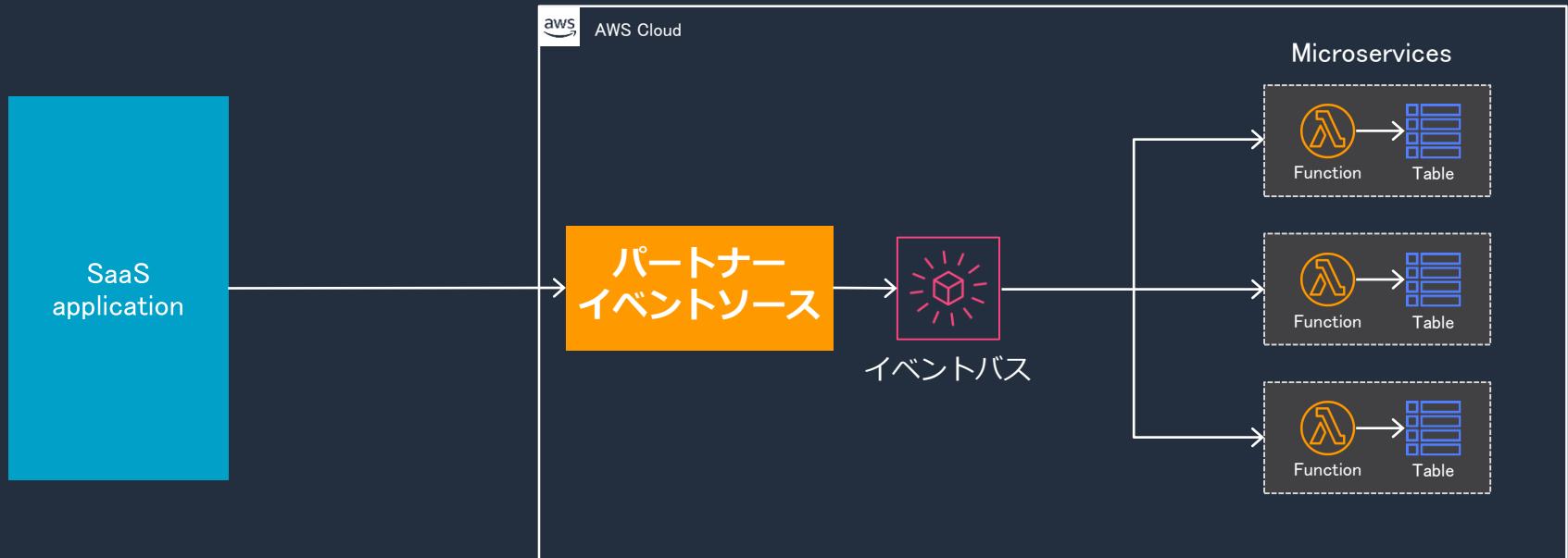
# 案3：SaaSのWebhook機能でPushしてもらう



SaaS provider

Your environment

# 案4：EventBridgeのSaaS連携機能の利用



SaaS事業者

お客様の環境

# 対応しているSaaSアプリケーション

- 24のSaaSサービスに対応（2020年1月22日Webinar実施時点）



PagerDuty



whispir

SAVIYNT

Segment

SignalFx

SUGARCRM

onelogin

Symantec™

Payshield

epsagon

Buildkite



kloudless

BUIDLHUB

Auth0

THUNDRA

Opsgenie

CleverTap

mongoDB®

New Relic®

KARTE

mackerel



Game Server Services

# 日本でも始まる、EventBridge 対応パートナー

NEW



ウェブサイトやモバイルアプリを利用する個々の顧客にあわせた体験を提供する CX プラットフォーム。

EventBridge 統合機能を使用することで、顧客行動イベントを AWS のサービスに送信して、BI、分析、機械学習などに活用可能。



運用中のクラウドやオンプレミスのサーバーにエージェントを 1つ入れるだけで、簡単にサーバー管理できる運用監視サービス。

Mackerel によるモニタリングで検出したイベントをトリガーに EventBridge を経由して AWS 上でさまざまなアクションを実行。



Game Server Services

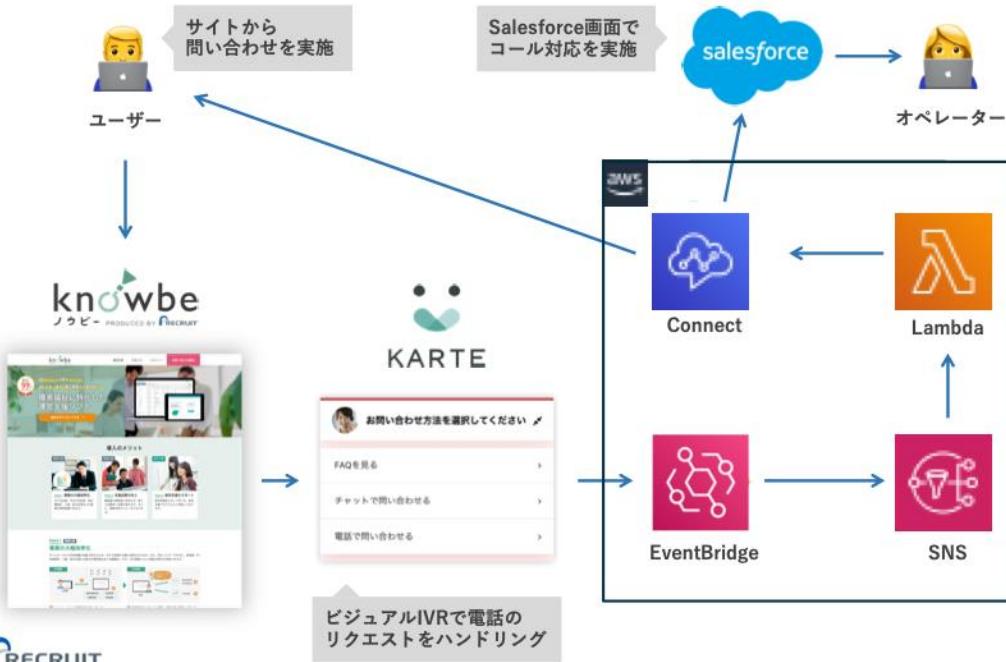
初期費用・運用費用なしで、サーバーアクセス 1回あたり 0.02円で使用できるゲームサーバー機能を提供。

EventBridge を使うことで、ゲームイベント（クエストのクリア/レベルアップ）と連携したアクションを追加拡張できます。

# EventBridge 活用事例

## Amazon Connect×KARTEで次世代CSプラットフォームを構築へ

### インバンドでの問い合わせ対応



### CX（顧客体験）の向上

従来は分断されていた電話対応をサイトに組み込むことで、問い合わせをサイトに一元化し、顧客の煩わしさを抑制することで、CXを向上する。

### シームレスなサービス連携

マルチクラウドをシームレスに連携することで、サービスのユーザー や社内の担当者が意識することなく、プラットフォームを利用できる。

### 拡張性の高いシステム

各クラウドサービスの機能やプラットフォームから取得されるデータを組み合わせることで、多様な要件に対応することができます。

# SaaSアプリケーションとの連携方法

## 1. パートナーアイベントソースの作成

SaaSアプリケーション側の管理画面からAWSアカウントIDとリージョンを指定し、AWS側にパートナーアイベントソースを作成する。

## 2. パートナーアイベントソースとイベントバスの関連付け

AWSアカウント内に作成されたパートナーアイベントソースをパートナーアイベントバスをひもづける。

## 3. イベントのルールを作成する

通常のイベント同様ルールを作成する。

# SaaSアプリケーションとの連携方法

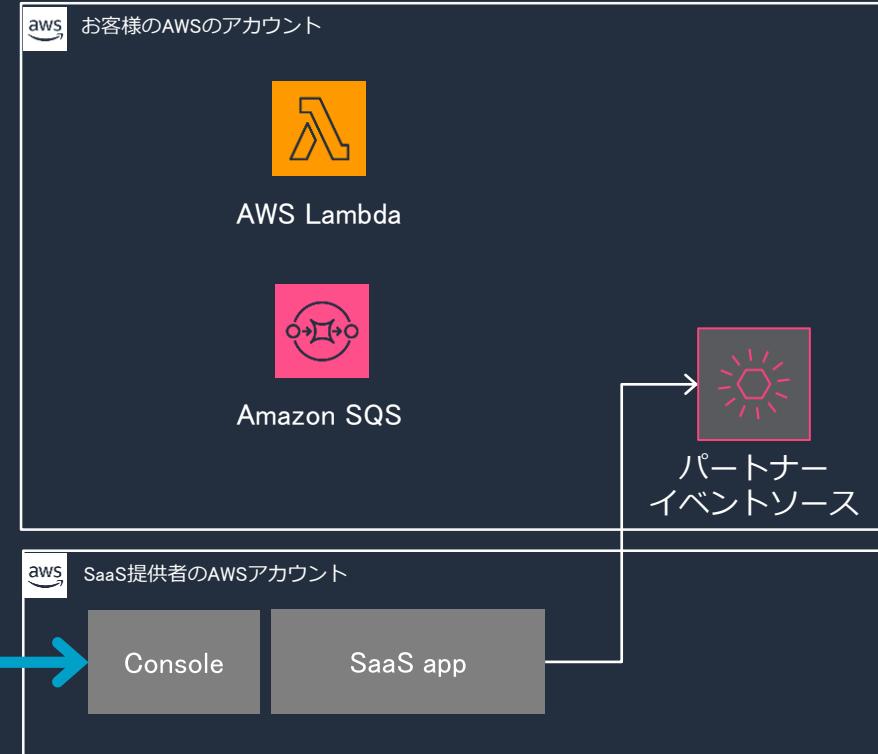
## ①パートナーイベントソースの作成

1. SaaSアプリケーションの管理画面からAWSアカウントIDとリージョンを設定する。

AWS account ID  
111111111111

AWS Region  
ap-northeast-1

Submit



# SaaSアプリケーションとの連携方法

## ②パートナーイベントソースとイベントバスの関連付け

### 2. パートナーイベントソースとイベントバスを関連付ける

パートナーイベントソース

パートナーイベントソース (1)

aws.partner/example.com/test-bus ① 保留中 関連付けられたイベントバスなし

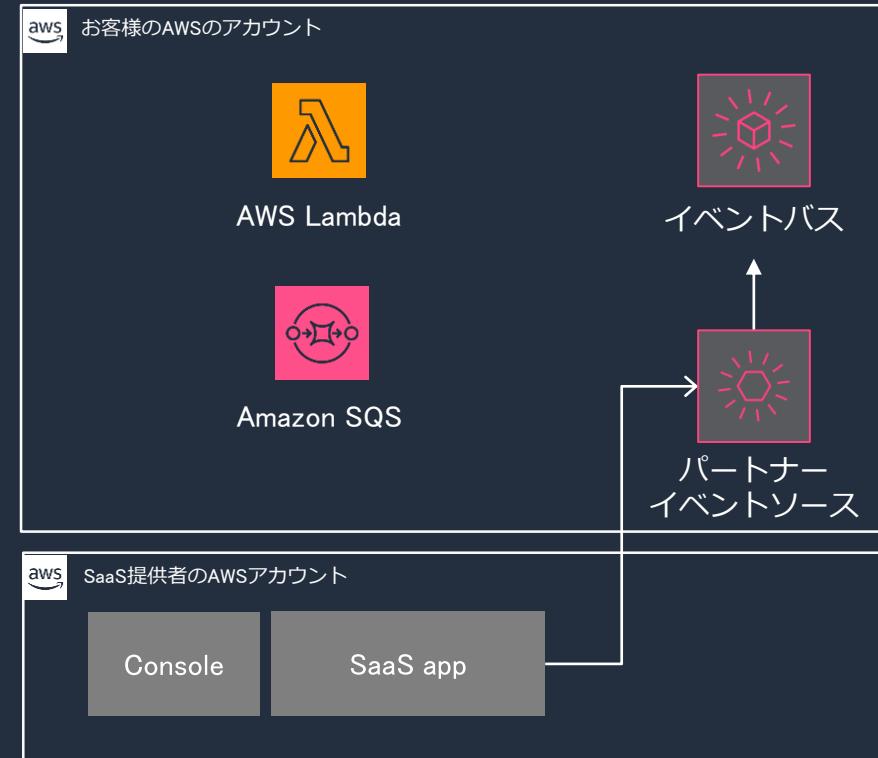
イベントバスと関連付ける

検索

名前 ステータス イベントバス



関連付けをする前は、  
パートナーイベントソースが作成されているが、  
イベントバスへの連携は開始されていない状態。



# SaaSアプリケーションとの連携方法

## ③ルールの作成

- ルールを作成し、イベントパターンとターゲットを定義する。

### ルールを作成

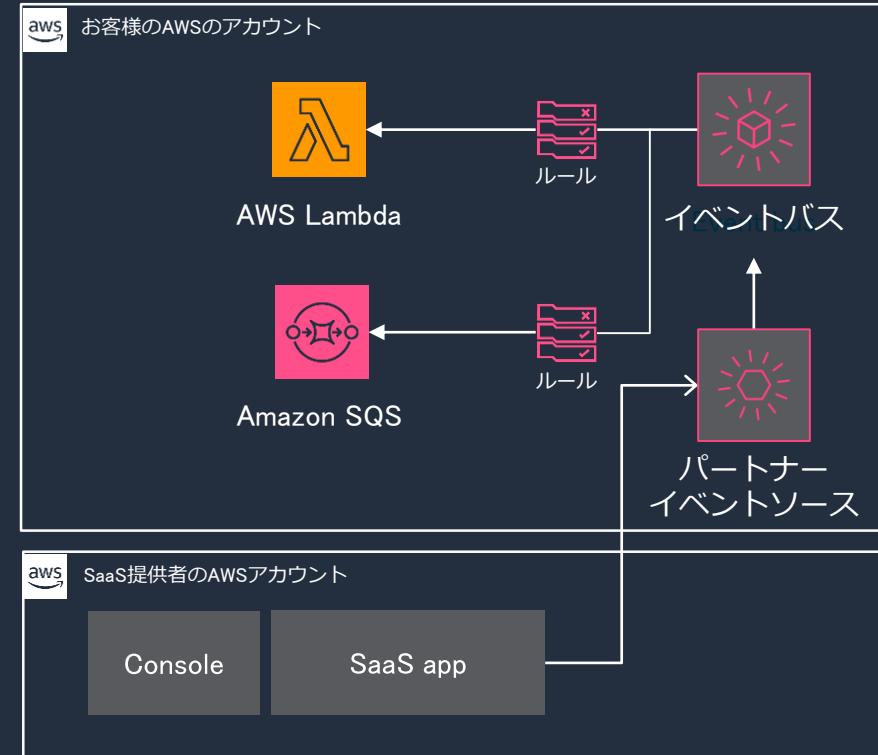
ルールは特定のイベントを監視し、ユーザーの選択した AWS ターゲットへと振り分けます。別の AWS アクションが起きたときに自動的に AWS アクションを実行するルールや、一定のスケジュールを設定して定期的に AWS アクションを実行するルールを作成できます。

名前と説明

名前

小文字/大文字、.(ピリオド)、,(コンマ)、-(ハイフン)、\_(アンダーバー) を含め、最大 64 文字まで使用できます。

説明 - 任意



# Amazon EventBridgeパートナーになるには

For APN Partner

- WebhookやPushベースのイベント送信ができるSaaSアプリケーションを提供している場合、容易にEventBridgeと統合することができます。
- AWS上でSaaSを提供している事業者がAmazon EventBridgeパートナーになる場合は、AWS Partner Networkに登録していただき、弊社の担当者までご連絡ください。

# 本日の内容

## 1. EventBridge とは？

- ・ イベントバス導入の背景
- ・ アーキテクチャ
- ・ 構成要素
- ・ アクセス制御

## 2. SaaS との連携

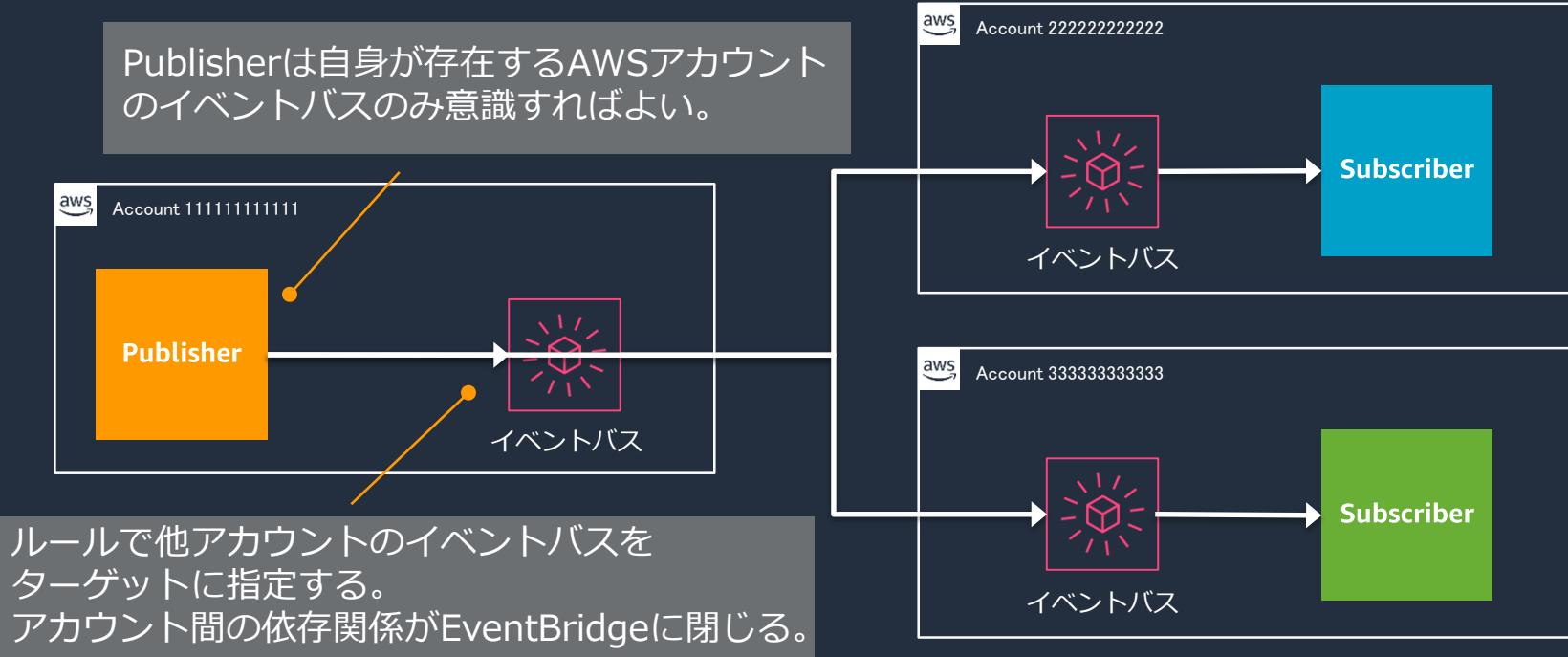
- ・ SaaSアプリケーションとの連携案
- ・ 対応しているSaaSアプリケーション
- ・ SaaSアプリケーションとの連携方法
- ・ EventBridge 活用事例
- ・ EventBridgeパートナーになるには

## 3. クロスアカウント連携

- 4. 直近のアップデート
- 5. 料金、制限
- 6. まとめ

# クロスアカウント連携

複数のAWSアカウントで同じイベントを処理したい場合、  
イベントバス間で送受信することで依存関係が局所的にできる

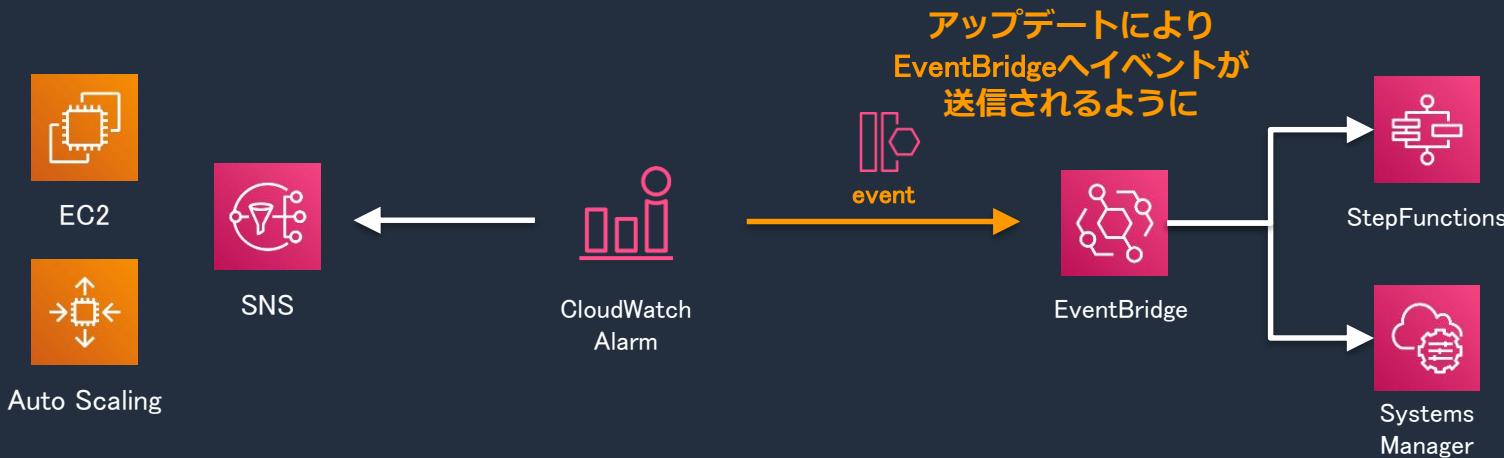


# 本日の内容

1. EventBridge とは?
  - イベントバス導入の背景
  - アーキテクチャ
  - 構成要素
  - アクセス制御
2. SaaS との連携
  - SaaSアプリケーションとの連携案
  - 対応しているSaaSアプリケーション
  - SaaSアプリケーションとの連携方法
  - EventBridge 活用事例
  - EventBridgeパートナーになるには
3. クロスアカウント連携
4. **直近のアップデート**
5. 料金、制限
6. まとめ

# CloudWatch Alarmのイベントが取得可能に

- CloudWatch Alarmのステータス変化がEventBridgeで取得可能に。
- CloudWatch Alarmの機能では送信先としてEC2、SNS、AutoScalingの操作のみが選択可能。本アップデートによりAlarmのイベントが**実装なしで様々なサービスに連携可能**になった。



# ECR(Elastic Container Registry)のイベントが取得可能に

- ECRのイベントがEventBridgeで取得可能に。
- これまでCloudTrail経由で送信されるイベントのみ取得可能であったため、EventBridgeが受信するまでにタイムラグがあったが、**本アップデートによってECRに対するイベントをリアルタイムで活用可能に。**



# コンテンツベースのフィルタリング

イベントパターンで文字列表現以外の柔軟な記述が可能に。

	概要	イベントパターンの記述例
前方一致 (Prefix Matching)	文字列の前方一致が条件として指定できる。	{ "time": [ { "prefix": "2017-10-02" } ] }
例外条件 (Anything-but Matching)	○○以外を条件として指定できる。	{ "detail": { "state": [ { "anything-but": "running" } ] } }
数値比較 (Numeric Matching)	数値の大小や等値の比較ができる。	{ "detail": { "c-count": [ { "numeric": [ ">", 0, "<=", 5 ] } ] } }
存在性 (Exists Matching)	フィールドが「存在すること」や「存在しないこと」を条件として指定できる。	{ "detail": { "c-count": [ { "exists": false } ] } }

# Schema Registry & Schema Discovery (Preview)

- Schema Registry
  - EventBridgeでやり取りされる様々なデータのスキーマを、コンソールからまとめて参照可能。

aws.ec2@EC2InstanceStateChangeNotification

バージョン 1 作成者 2019年12月1日 9:11 JST アクション ▾ コードバインディングのダウンロード

```
8   "components": {  
9     "schemas": {  
10       "AWSEvent": {  
11         "type": "object",  
12         "required": ["detail-type", "resources", "id", "source", "time", "detail", "region", "version", "account"],  
13         "x-amazon-events-detail-type": "EC2 Instance State-change Notification",  
14         "x-amazon-events-source": "aws.ec2",  
15         "properties": {  
16           "detail": {  
17             "$ref": "#/components/schemas/EC2InstanceStateChangeNotification"  
18           },  
19           "detail-type": {  
20             "type": "string"  
21           },  
22           "resources": {  
23             "type": "array"  
24           }  
25         }  
26       }  
27     }  
28   }  
29 }
```

プロパティの一覧とそれぞれの型  
(String や独自の型など)

バージョン 1 作成者 2019年12月1日 9:11 JST アクション

```
48   },  
49   "EC2InstanceStateChangeNotification": {  
50     "type": "object",  
51     "required": ["instance-id", "state"],  
52     "properties": {  
53       "instance-id": {  
54         "type": "string"  
55       },  
56       "state": {  
57         "type": "string"  
58       }  
59     }  
60   }  
61 }  
62 }
```

# Schema Registry & Schema Discovery (Preview)

- Schema Discovery
  - SaaSイベントやカスタムイベントがイベントバスに送信されると、スキーマ情報を自動生成可能に。
  - (例) DatadogのAlertが発生した際に送信されるイベントのスキーマ

aws.partner-datadog.com-example@DatadogAlertNotification

バージョン 2 作成者 2020年1月20日 17:46 JST

バージョン 2 ▾ 新しいバージョンとして保存 アクション ▾ コードバイニングのダウンロード

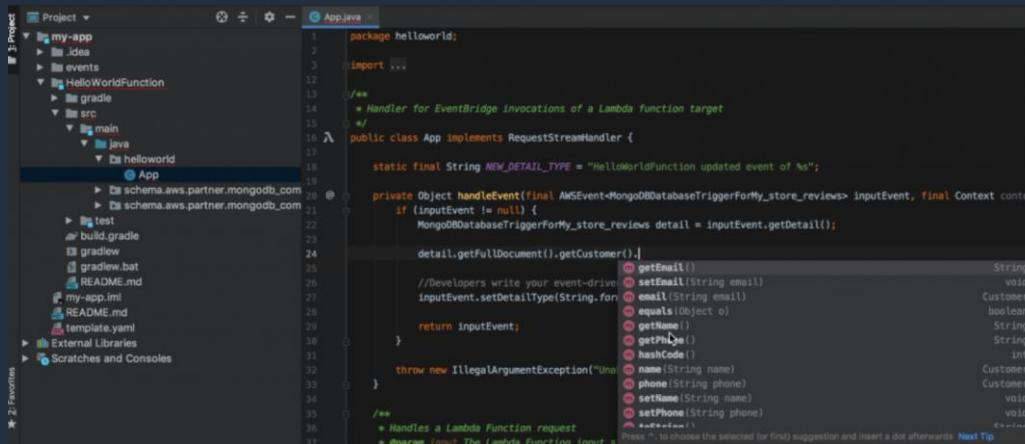
```
1 {
2   "openapi": "3.0.0",
3   "info": {
4     "version": "1.0.0",
5     "title": "DatadogAlertNotification"
6   },
7   "paths": {},
8   "components": {
9     "schemas": {
10       "AWSEvent": {
11         "type": "object",
12         "required": ["detail-type", "resources", "detail", "id", "source", "time", "region", "version", "account"],
13         "x-amazon-events-detail-type": "Datadog Alert Notification",
14         "x-amazon-events-source": "aws.partner/datadog.com/nogamincho",
15         "properties": {

```

AWSのサービスと全く同じ形で、SaaSのイベントやカスタムイベントのスキーマも確認できる

# Schema Registry & Schema Discovery (Preview)

- **Code Bindings** – スキーマ定義を言語ごとにダウンロード可能
  - AWS Toolkit for IntelliJ および Visual Studio Code を用いてスキーマ定義を取得可能。
  - Java 8+、Python 3.6+、TypeScript 3 の 3 種類に対応
  - ダウンロードしたスキーマ定義をインポートしておくことで、コード補完やコンパイル時のエラー検出に役立てることができる



The screenshot shows the IntelliJ IDEA interface with a Java file named `App.java` open. The code defines a class `App` that implements `RequestStreamHandler`. A tooltip from the code completion feature is displayed over the `Customer` class, listing various methods like `getEmail()`, `setEmail(String email)`, etc. The project structure on the left shows a directory `my-app` containing `gradle`, `src`, and `test` directories, along with `build.gradle` and `gradlew` files.

```
1 package helloworld;
2
3 import ...
4
5 /**
6  * Handler for EventBridge invocations of a Lambda function target
7  */
8 public class App implements RequestStreamHandler {
9
10     static final String NEW_DETAIL_TYPE = "HelloWorldFunction updated event of %s";
11
12     private Object handleEvent(final AWSEvent<MongoDBTriggerForMy_store_reviews> inputEvent, final Context context)
13     if (inputEvent != null) {
14         MongoDBTriggerForMy_store_reviews detail = inputEvent.getDetail();
15
16         detail.getFullDocument().getCustomer();
17
18         //Developers write your event-driven
19         inputEvent.setDetailType(String.format("Customer %s", detail.getName()));
20
21         return inputEvent;
22     }
23
24     throw new IllegalArgumentException("Unknown event type");
25
26 }
27
28 /**
29  * Handles a Lambda Function request
30  */
31
32 @LambdaFunctionInput
33
34 }
```

# 本日の内容

1. EventBridge とは?
  - イベントバス導入の背景
  - アーキテクチャ
  - 構成要素
  - アクセス制御
2. SaaS との連携
  - SaaSアプリケーションとの連携案
  - 対応しているSaaSアプリケーション
  - SaaSアプリケーションとの連携方法
  - EventBridge 活用事例
  - EventBridgeパートナーになるには
3. クロスアカウント連携
4. 直近のアップデート
5. 料金、制限
6. まとめ

# Amazon EventBridgeの料金

2020年1月22日時点の東京リージョンでの利用料金

課金対象	料金
受信した イベント	AWSサービスのイベント
	無料
	カスタムイベント
	100万件あたり1.00USD
スキーマレジストリ (Preview)	SaaSイベント
	100万件あたり1.00USD
	クロスアカウントイベント
	100万件あたり1.00USD
スキーマディスカバリ (Preview)	無料
	1ヶ月あたり500万件まで無料 それ以降は100万件あたり0.10USD

# Amazon EventBridgeの制限

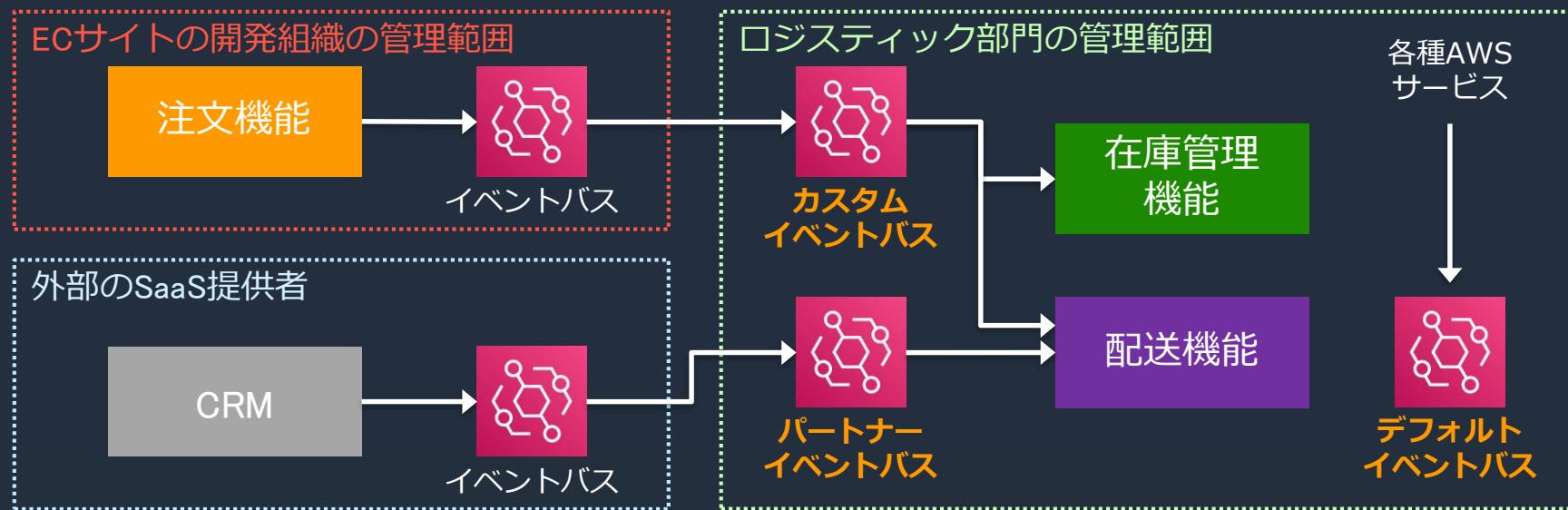
上限の対象	上限値	上限緩和の可否
イベントの送信	毎秒 400 リクエスト	○
リクエストの最大サイズ	1リクエストあたり256KB	○
エントリ数	1リクエストあたり10エントリ	—
ターゲットの呼び出し	毎秒 750 回	○
イベントバス数	1リージョンあたり100	—
ルール数	1イベントバスあたり300個	○
イベントパターン	2048文字	—
ターゲット数	1ルールあたり5個	—

# 本日の内容

1. EventBridge とは?
  - イベントバス導入の背景
  - アーキテクチャ
  - 構成要素
  - アクセス制御
2. SaaS との連携
  - SaaSアプリケーションとの連携案
  - 対応しているSaaSアプリケーション
  - SaaSアプリケーションとの連携方法
  - EventBridge 活用事例
  - EventBridgeパートナーになるには
3. クロスアカウント連携
4. 直近のアップデート
5. 料金、制限
6. まとめ

# まとめ

- ・ イベントを利用したアーキテクチャの構築が容易になる。
- ・ サーバー管理不要なイベントバスが利用でき、実装なしにAWSサービスへイベントが連携できる。
- ・ SaaSアプリケーションとのイベントの連携が容易になり、これまで取り扱いづらかったSaaSのデータを利用できる。



# 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。

# AWS の日本語資料の場所「AWS 資料」で検索



The screenshot shows the AWS Japan Language Resources page. At the top, there's a navigation bar with the AWS logo, search bar, and links for "日本担当チームへお問い合わせ", "サポート", "日本語", "アカウント", and "コンソールにサインイン". Below the navigation bar is a horizontal menu with links for "製品", "ソリューション", "料金", "ドキュメント", "学習", "パートナー", "AWS Marketplace", "その他", and a search icon. The main content area features a large title "AWS クラウドサービス活用資料集トップ" and a descriptive paragraph about the service. At the bottom, there are four buttons: "AWS Webinar お申込", "AWS 初心者向け", "業種・ソリューション別資料", and "サービス別資料".

AWS クラウドサービス活用資料集トップ

アマゾン ウェブ サービス (AWS) は安全なクラウドサービスプラットフォームで、ビジネスのスケールと成長をサポートする処理能力、データベースストレージ、およびその他多種多様な機能を提供します。お客様は必要なサービスを選択し、必要な分だけご利用いただけます。それらを活用するために役立つ日本語資料、動画コンテンツを多数ご提供しております。(本サイトは主に、AWS Webinar で使用した資料およびオンデマンドセミナー情報を掲載しています。)

AWS Webinar お申込 »

AWS 初心者向け »

業種・ソリューション別資料 »

サービス別資料 »

<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>





# Amazon EventBridge グローバルエンドポイント

櫻谷 広人

Partner Solutions Architect  
2023/10

# 自己紹介

名前：櫻谷 広人 (Hirotto Sakuraya)

所属：AWS Technology Partnerships

SaaS, Partner Solutions Architect

経歴：主にバックエンドエンジニアとして Web サービスやネイティブアプリの開発に従事。前職では CtoC のスタートアップで執行役員 CTO を務める。

好きな AWS サービス：Amazon EventBridge



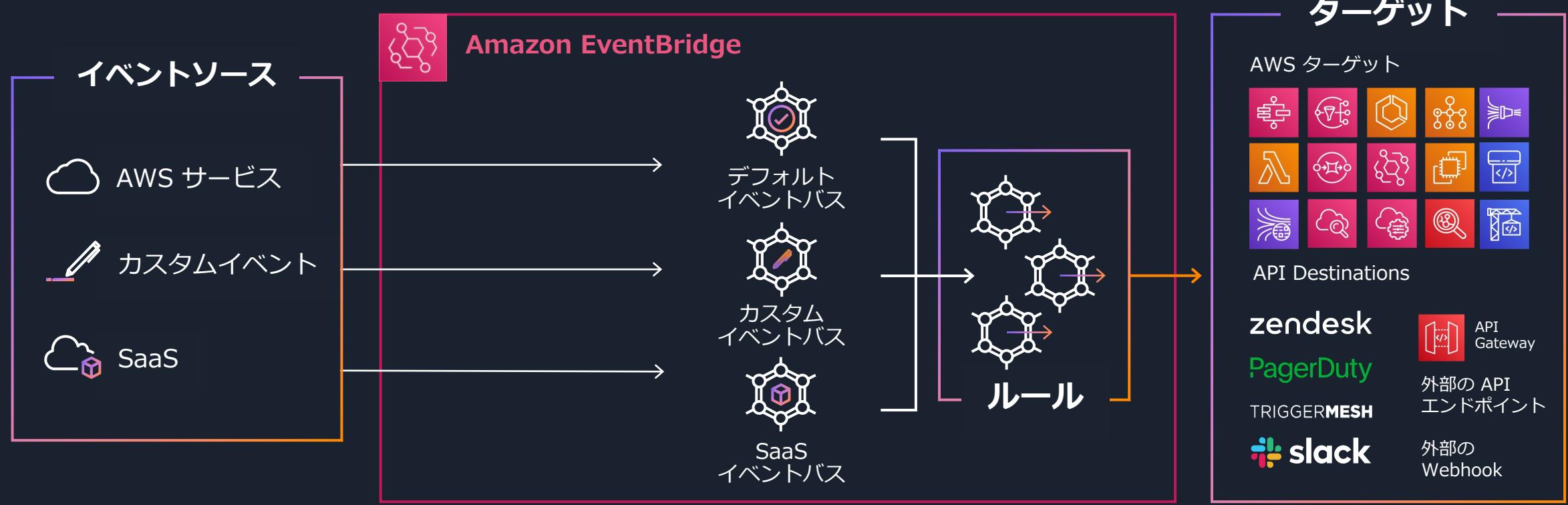
# 本セミナーの対象者

- Amazon EventBridge について深く学びたい方
- イベント駆動型アプリケーションの開発に興味をお持ちの方
- システムの可用性およびレジリエンスの向上に取り組まれている方

# 本セミナーの取り扱う範囲

- Amazon EventBridge グローバルエンドポイントについて
- \* Amazon EventBridge の他の機能については別のセミナー動画をご覧ください

# Amazon EventBridge とは



アプリケーション、統合された SaaS アプリケーション、および AWS のサービスから生成されたイベントを受信、フィルタリング、変換、ルーティング、および配信することができるサーバーレスイベントバス。大規模なイベント駆動型アプリケーションの開発を可能に。

# EventBridge グローバルエンドポイントとは



## Amazon EventBridge が自動フェイルオーバーと復旧のためのグローバルエンドポイントを導入

投稿日: Apr 7, 2022

Amazon EventBridge がグローバルエンドポイントのサポートを開始しました。これは、お客様が AWS でイベント駆動型アプリケーションの可用性を向上させるためのよりシンプルで信頼性の高い方法です。グローバルエンドポイントは、サービスの中止時に、手動での操作を必要とすることなくイベントの取り込みをセカンダリーリージョンに自動でフェイルオーバーすることによって、お客様が堅牢で信頼性のあるアプリケーションを簡単に構築できるようにする新しい機能です。お客様は、レプリケーションを使用して、これらのサービス中止時にリスクにさらされるデータを最小限に抑えることができます。

EventBridge は、組み込みの統合を通じてユーザー独自のアプリケーション、サードパーティ SaaS アプリケーション、および AWS のその他サービスの間でのイベントのルーティングを行うことによって、スケーラブルなイベント駆動型アプリケーションを作成できるようにする、サーバーレスイベントバスサービスです。ルーティングルールをセットアップしてデータの送信先を決定することができ、データやシステムでの変更が発生したときにアプリケーションがそれらに対応することを可能にします。イベントの取り込みと配信、セキュリティ、認可、およびエラー処理は Amazon EventBridge が対処するので、イベント駆動型アプリケーションの構築が容易になります。

グローバルエンドポイントを使用することで、お客様は、いつフェイルオーバーを行って、いつプライマリリージョンへのイベントのルーティングを再開するかを判断するために、CloudWatch アラーム (Route53 ヘルスチェック経由) を使用して障害を管理し、フェイルオーバー基準を設定する柔軟性を得られるようになります。お客様がグローバルエンドポイントにイベントを発行したら、イベントはプライマリリージョン内のイベントバスにルーティングされます。プライマリリージョンでエラーが検出された場合は、お客様のヘルスチェックが異常としてマークされ、EventBridge がセカンダリリージョンに受信イベントをルーティングします。

- EventBridge サービスの障害発生時に、別リージョンへの自動フェイルオーバーが可能に
- 障害の検知とフェイルオーバーには Amazon Route 53 を利用
- イベント駆動アプリケーションの可用性向上、データ損失の緩和を実現できる
- セカンダリリージョンへのイベントの常時レプリケーションも設定可能

EventBridge を利用するアプリケーションのリージョン障害耐性の向上が期待できる！

# そもそも：なぜレジリエンス（回復性）は重要なのか？



*ed the Turing test*

**“Everything fails,  
all the time.”**

**Werner Vogels**  
(CTO, Amazon.com)

# “レジリエンス（回復性）”を構成する要素

“回復性とは、インフラストラクチャやサービスの中斷から復旧して、需要を満たすコンピューティングリソースを動的に獲得し、設定ミスや一時的なネットワーク問題などの中斷の影響を緩和するワークロードの能力です。”

[Disaster Recovery of Workloads on AWS: Recovery in the Cloud](#)

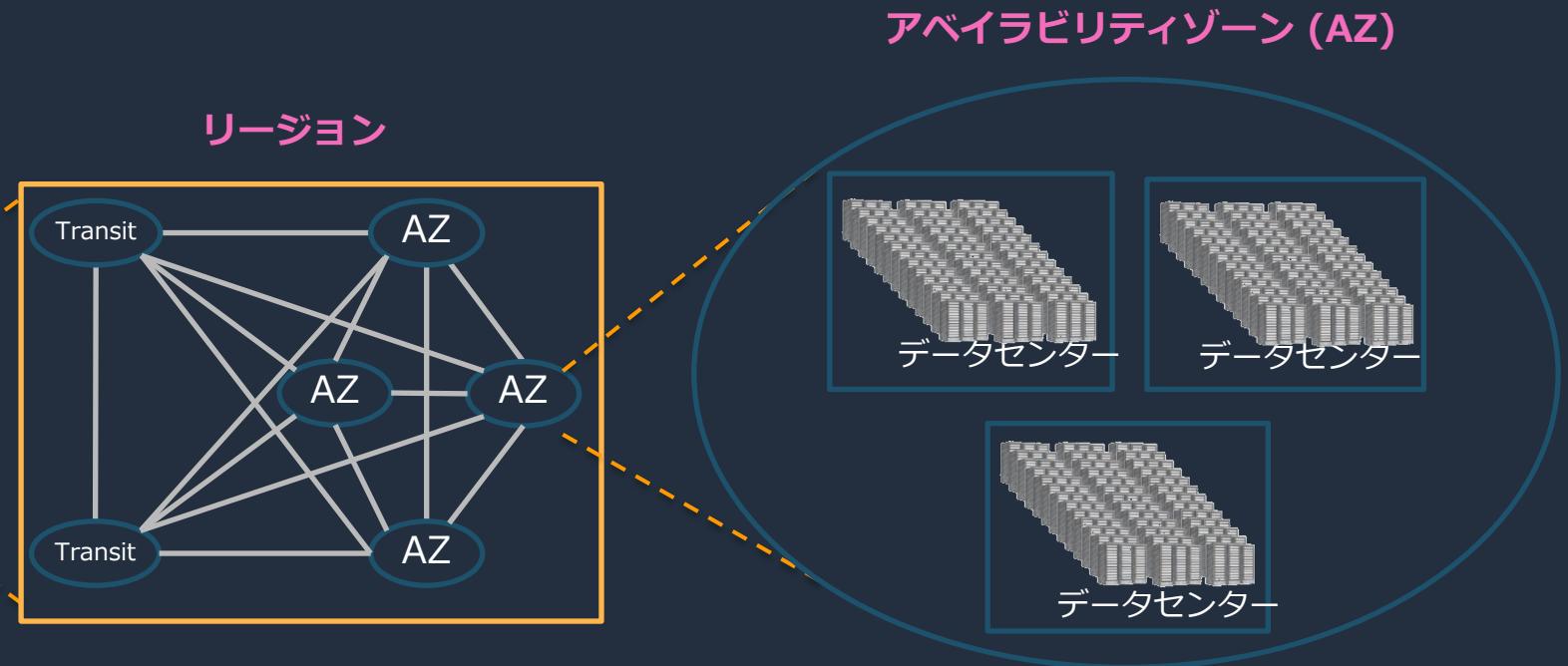


\* MTBF: Mean Time Between Failures  
\* MTTR: Mean Time To Repair

\* RTO: Recovery Time Objective  
\* RPO: Recovery Point Objective

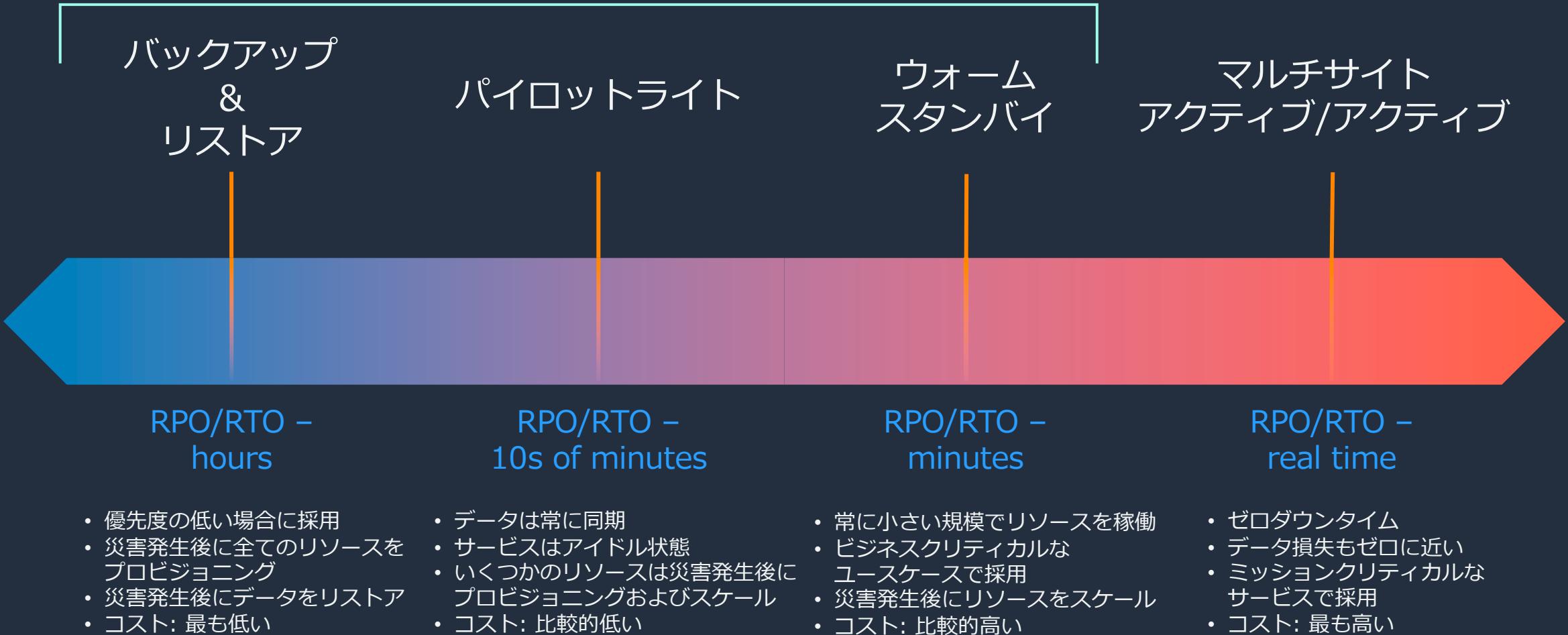
# AWS グローバルインフラストラクチャのおさらい

- リージョンは、複数 AZ から構成される
- AZ は、互いに独立した複数のデータセンターから構成される
- マルチ AZ / マルチリージョン構成による耐障害性の向上



# クラウド内のディザスタリカバリオプション

アクティブ/パッシブ



# 各 AWS サービスのレジリエンス設計

デフォルトでマルチ AZ



Amazon Elastic  
File System  
(Amazon EFS)



Amazon Simple  
Storage Service  
(Amazon S3)



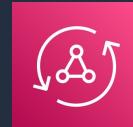
Amazon  
DynamoDB



AWS  
Lambda



AWS Step  
Functions



AWS AppSync



Amazon API  
Gateway



Amazon  
EventBridge



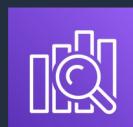
Amazon Simple  
Notification  
Service  
(Amazon SNS)



Amazon  
Kinesis



Amazon Simple  
Queue Service  
(Amazon SQS)



Amazon  
OpenSearch  
Service



Amazon  
Aurora



Amazon  
ElastiCache



AWS  
Fargate



Amazon Managed  
Workflows for  
Apache Airflow



Elastic  
Load Balancing



Amazon Managed  
Streaming  
for Kafka



Amazon  
MQ

File stores

Search

RDBMS

No-SQL  
Caching

Compute

Workflow

Ingress

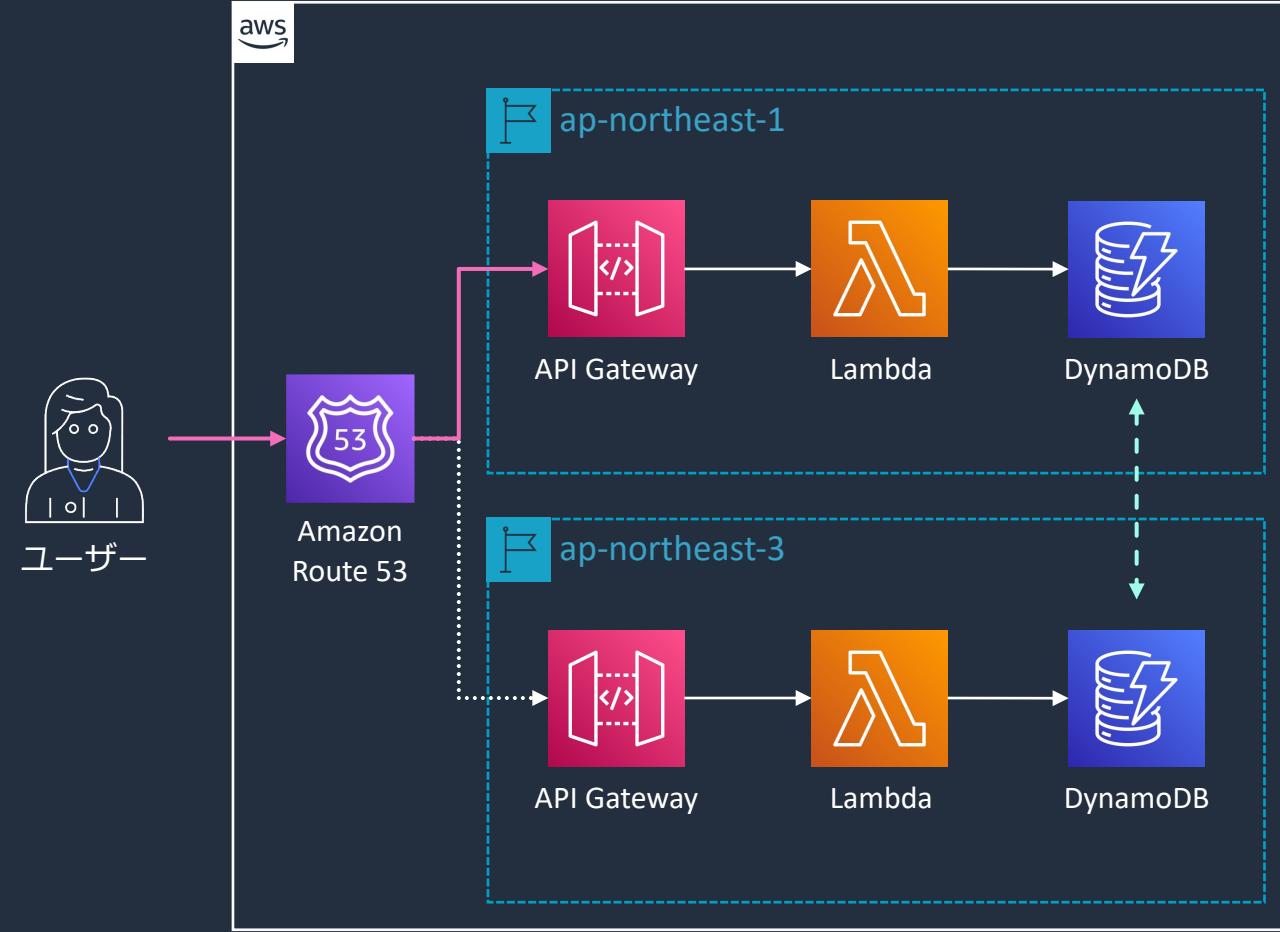
EventBus  
Fanout

Streams

Queues

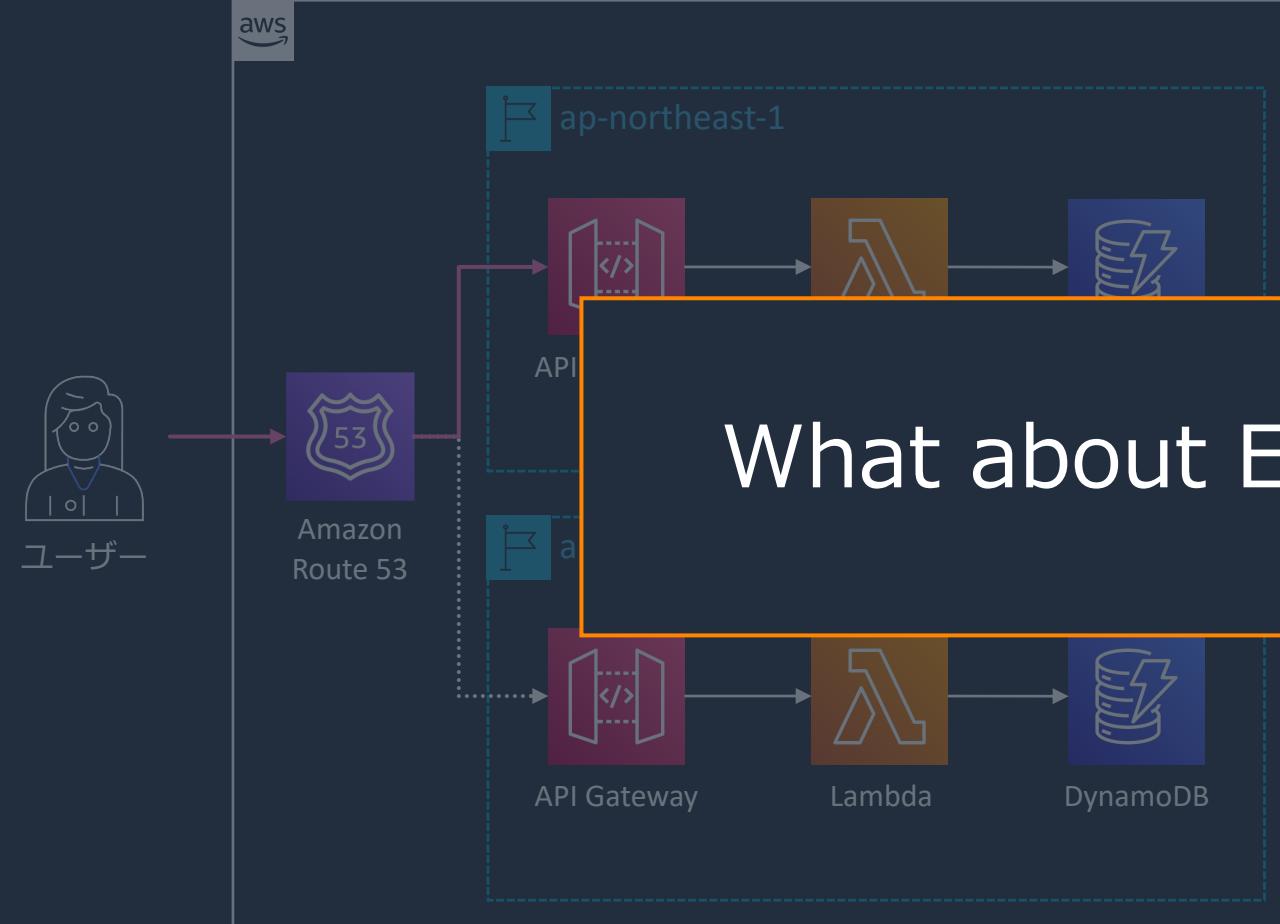
設定次第でマルチ AZ

# サーバーレスアプリケーションでレジリエンスを考える



- 各サービスビルトインの AZ 障害耐性
- マルチリージョン構成でさらに堅牢に
  - リージョン障害時にフェイルオーバーするアクティブ/パッシブ構成
  - レイテンシーに基づくルーティングによるアクティブ/アクティブ構成
- DynamoDB グローバルテーブルによるリージョンレベルでのデータのレプリケーション
- Route 53 によるフェイルオーバーの自動化

# サーバーレスアプリケーションでレジリエンスを考える



What about EventBridge?

- 各サービスビルトインの AZ 障害耐性
- マルチリージョン構成でさらに堅牢に
  - リージョン障害時にフェイルオーバーする
  - リージョン間でデータ同期する
- EventBridgeによるイベントバス構成
- CloudWatchルーティングによるフェイルオーバー構成
- DynamoDBアソシエイトテーブルによるリージョン間データ同期
- リージョンレベルでのデータのレプリケーション
- Route 53によるフェイルオーバーの自動化

# (再掲) EventBridge グローバルエンドポイントとは



## Amazon EventBridge が自動フェイルオーバーと復旧のためのグローバルエンドポイントを導入

投稿日: Apr 7, 2022

Amazon EventBridge がグローバルエンドポイントのサポートを開始しました。これは、お客様が AWS でイベント駆動型アプリケーションの可用性を向上させるためのよりシンプルで信頼性の高い方法です。グローバルエンドポイントは、サービスの中止時に、手動での操作を必要とすることなくイベントの取り込みをセカンダリーリージョンに自動でフェイルオーバーすることによって、お客様が堅牢で信頼性のあるアプリケーションを簡単に構築できるようにする新しい機能です。お客様は、レプリケーションを使用して、これらのサービス中止時にリスクにさらされるデータを最小限に抑えることができます。

EventBridge は、組み込みの統合を通じてユーザー独自のアプリケーション、サードパーティ SaaS アプリケーション、および AWS のその他サービスの間でのイベントのルーティングを行うことによって、スケーラブルなイベント駆動型アプリケーションを作成できるようにする、サーバーレスイベントバスサービスです。ルーティングルールをセットアップしてデータの送信先を決定することができ、データやシステムでの変更が発生したときにアプリケーションがそれらに対応することを可能にします。イベントの取り込みと配信、セキュリティ、認可、およびエラー処理は Amazon EventBridge が対処するので、イベント駆動型アプリケーションの構築が容易になります。

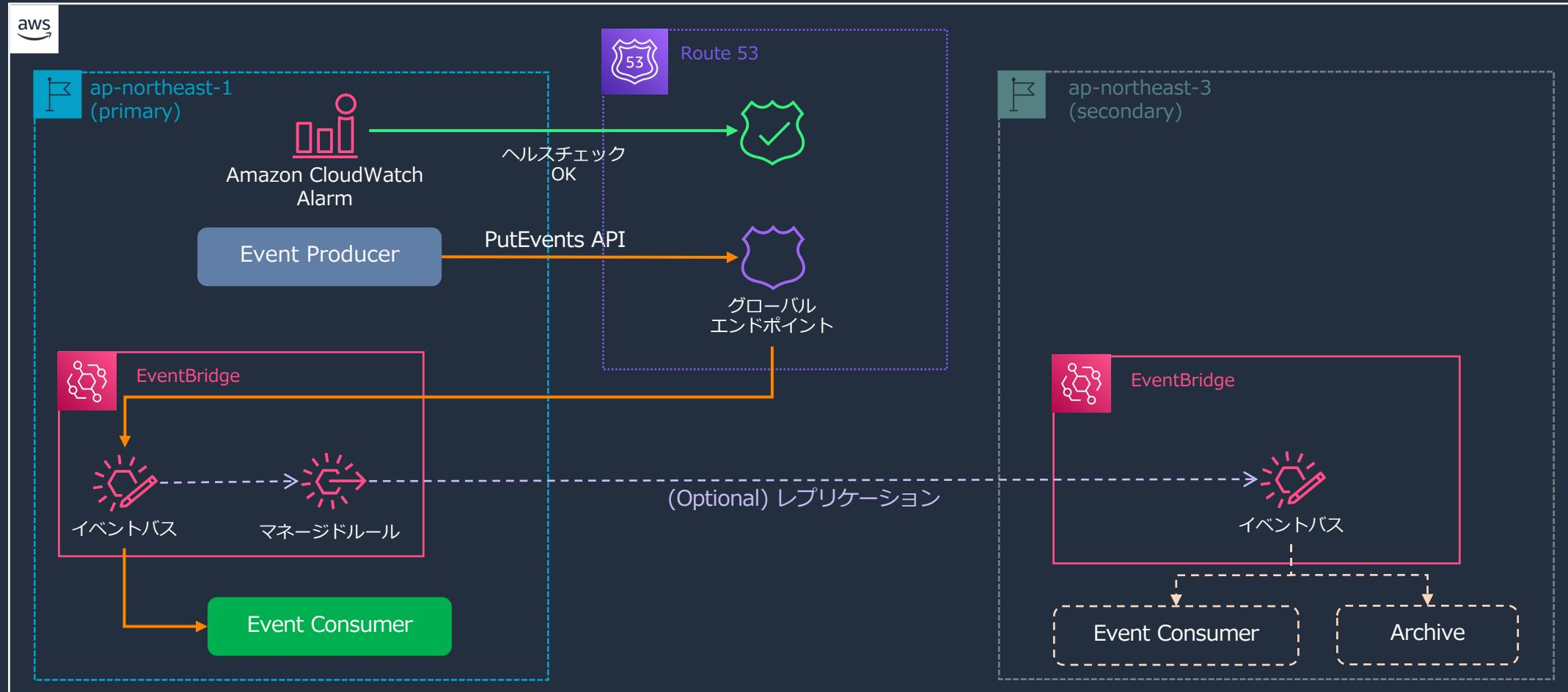
グローバルエンドポイントを使用することで、お客様は、いつフェイルオーバーを行って、いつプライマリリージョンへのイベントのルーティングを再開するかを判断するために、CloudWatch アラーム (Route53 ヘルスチェック経由) を使用して障害を管理し、フェイルオーバー基準を設定する柔軟性を得られるようになります。お客様がグローバルエンドポイントにイベントを発行したら、イベントはプライマリリージョン内のイベントバスにルーティングされます。プライマリリージョンでエラーが検出された場合は、お客様のヘルスチェックが異常としてマークされ、EventBridge がセカンダリリージョンに受信イベントをルーティングします。

- EventBridge サービスの障害発生時に、別リージョンへの自動フェイルオーバーが可能に
- 障害の検知とフェイルオーバーには Amazon Route 53 を利用
- イベント駆動アプリケーションの可用性向上、データ損失の最小化を実現できる
- セカンダリリージョンへのイベントの常時レプリケーションも設定可能

EventBridge を利用するアプリケーションのリージョン障害耐性の向上が期待できる！

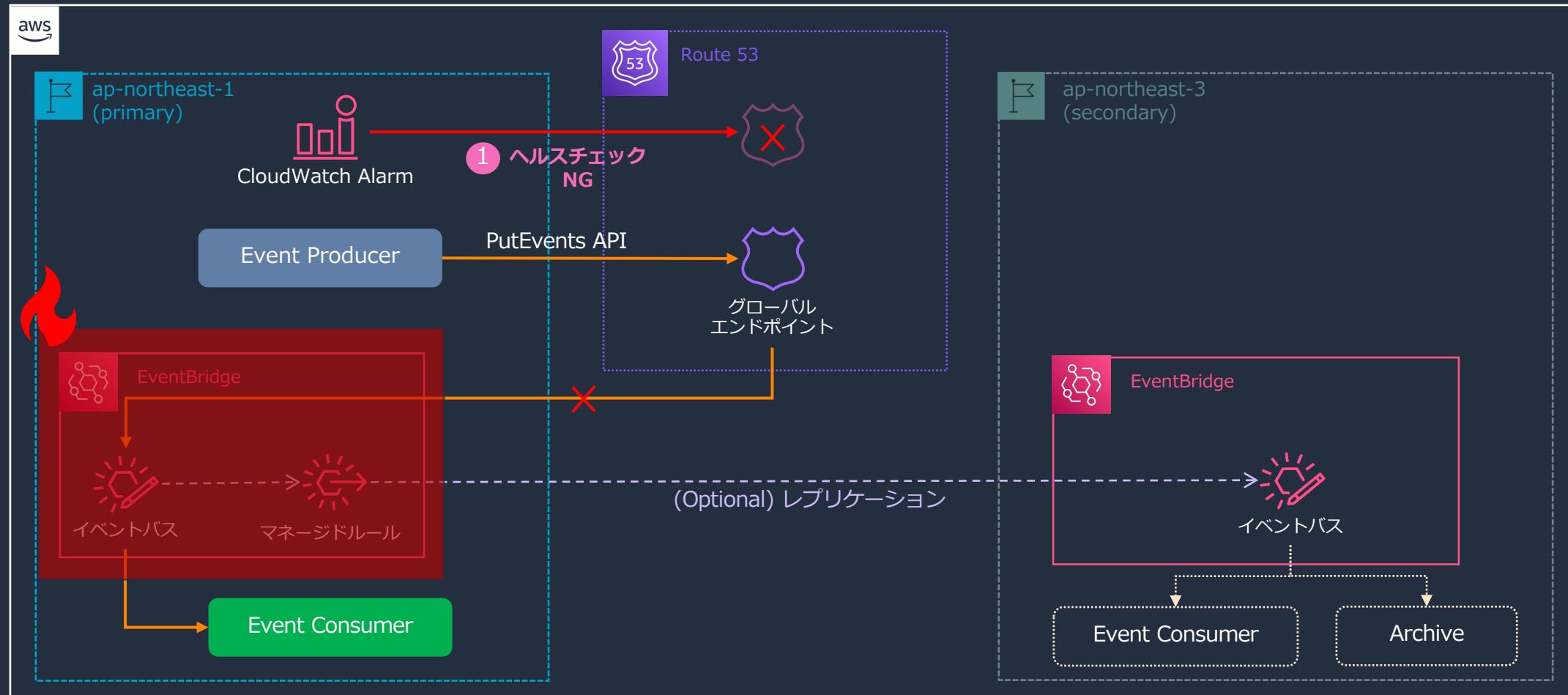
# グローバルエンドポイントの仕組み

平常時



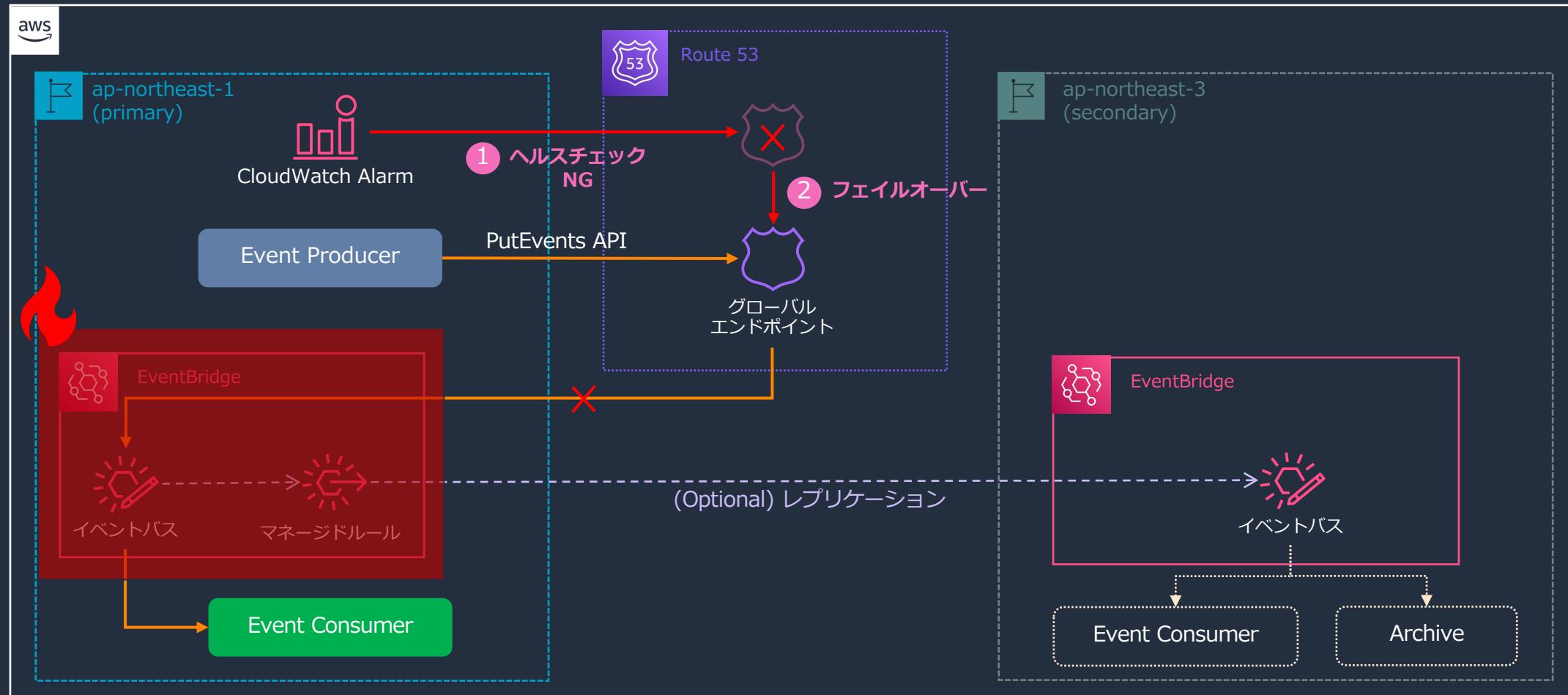
# グローバルエンドポイントの仕組み

障害発生時



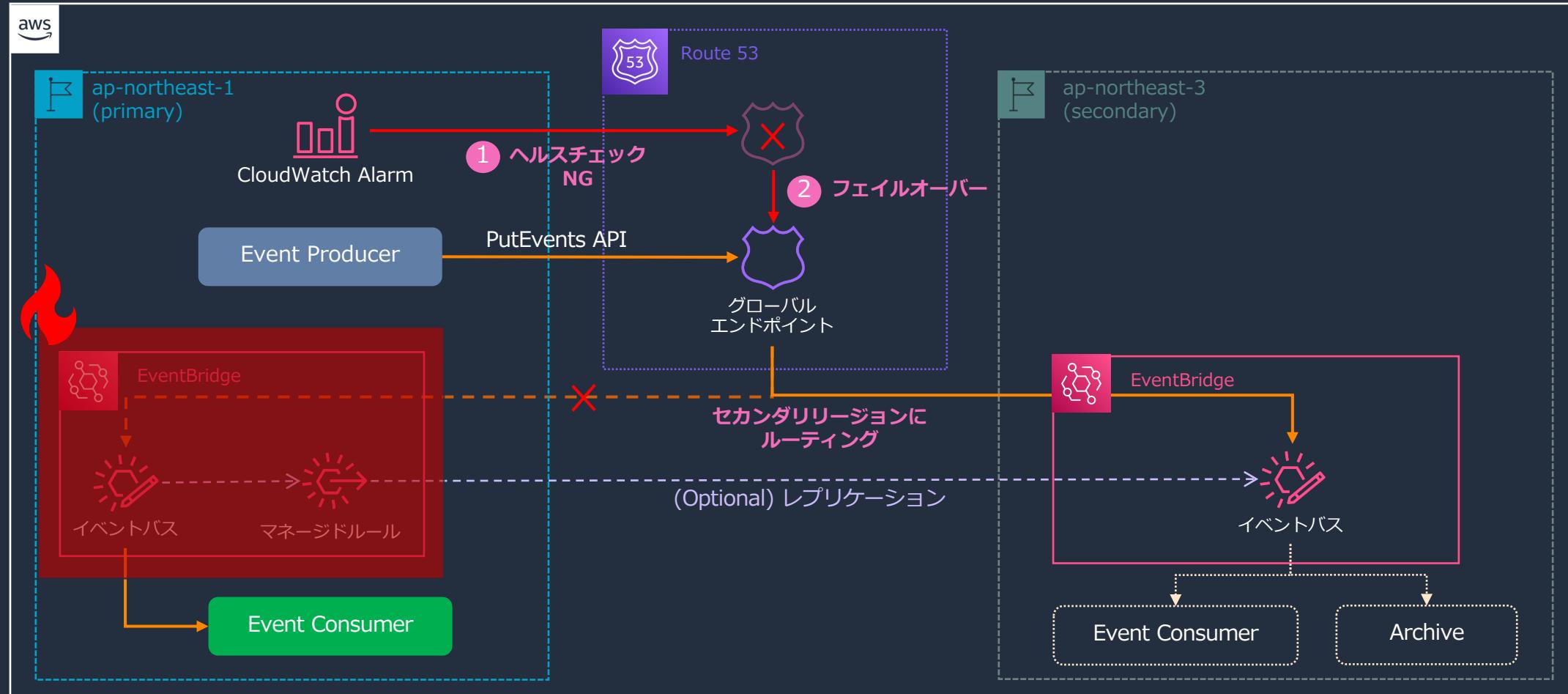
# グローバルエンドポイントの仕組み

障害発生時



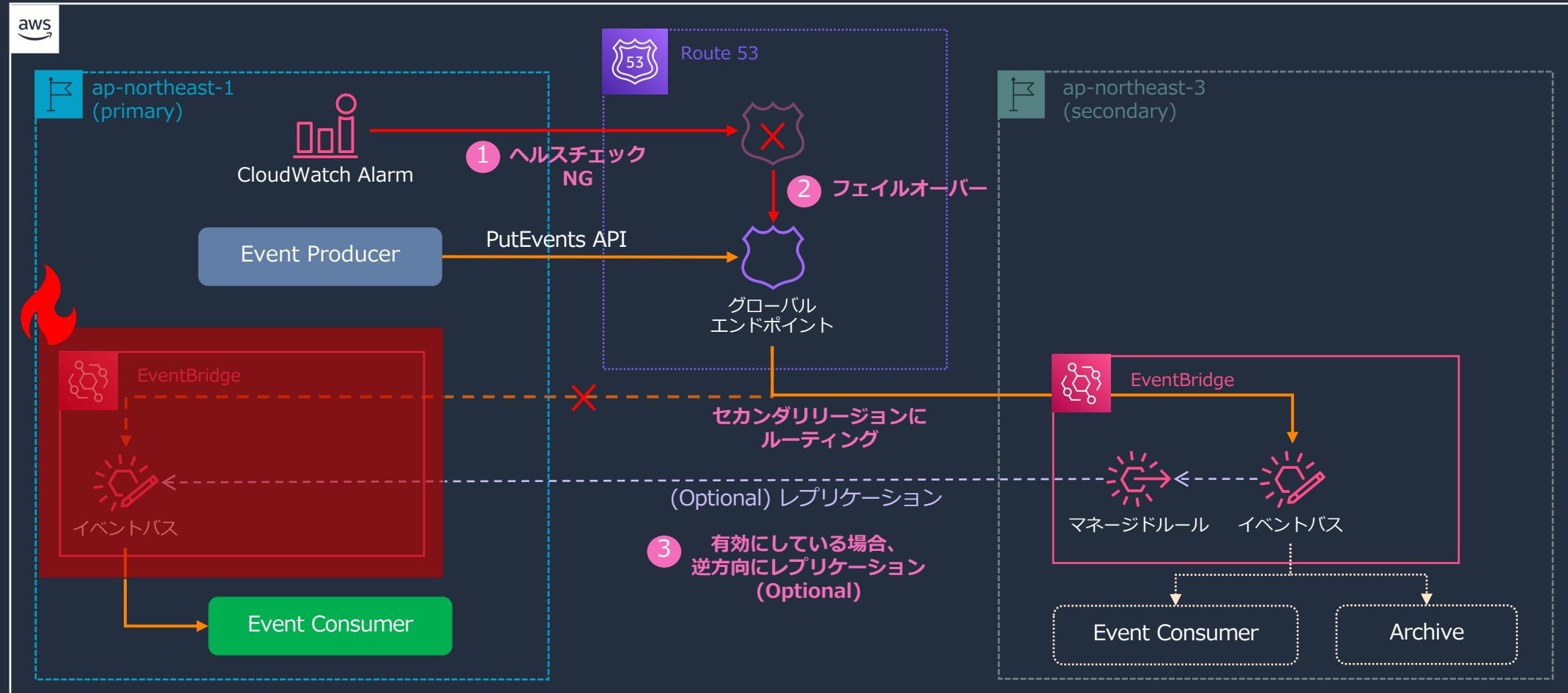
# グローバルエンドポイントの仕組み

障害発生時



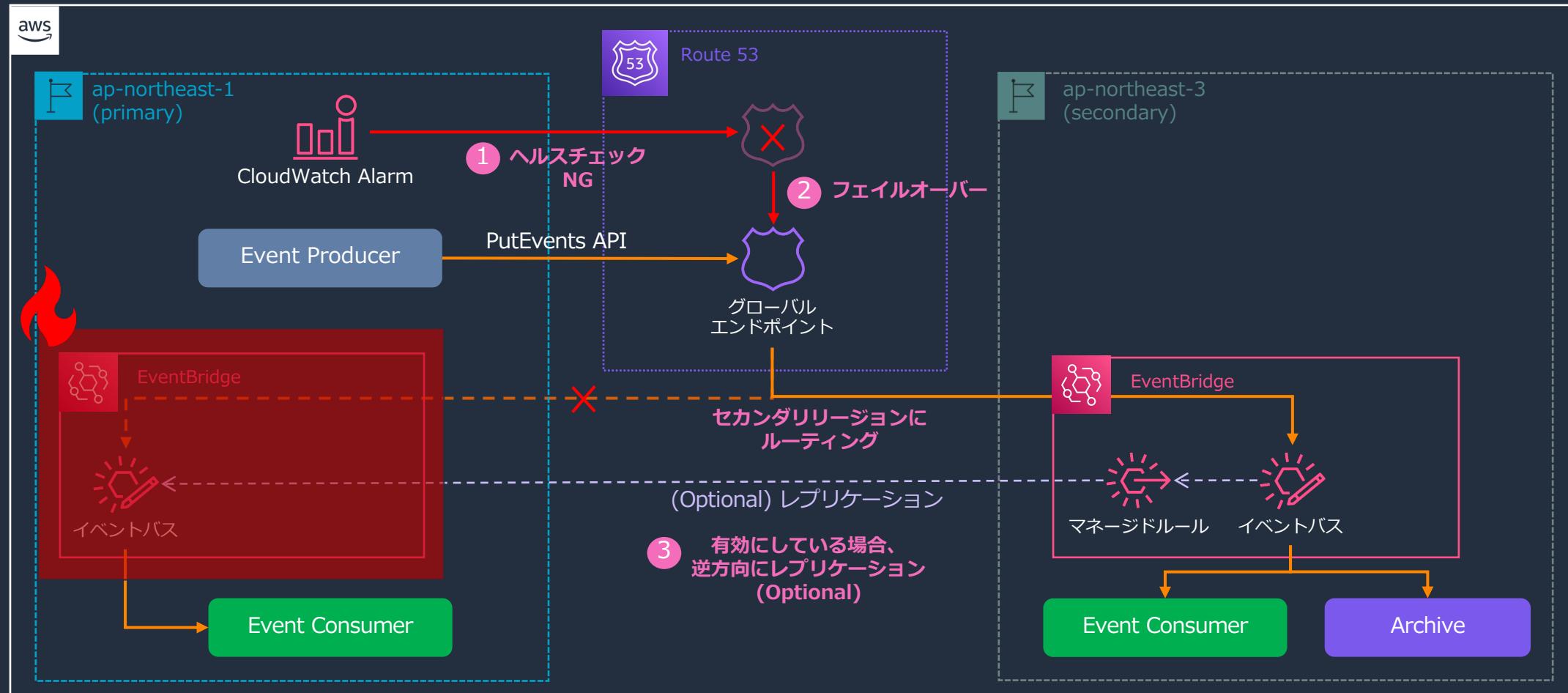
# グローバルエンドポイントの仕組み

障害発生時



# グローバルエンドポイントの仕組み

障害発生時



# クライアント（イベント送信側）で必要な変更

- グローバルエンドポイントに対して PutEvents API を実行するようにする
  - AWS CLI の場合

```
aws events put-events --entries file://event.json --endpoint-id e10jyciak4.jvp
```

event.json

```
[  
  {  
    "Source": "custom",  
    "DetailType": "custom-type",  
    "Detail": "{ \"id\": \"hoge\" }",  
    "EventBusName": "custom-bus"  
  }  
]
```

\* カスタムバスの場合、プライマリおよびセカンダリリージョンでイベントバスの名前を統一する必要がある

- AWS SDK の場合、AWS Common Runtime (CRT) ライブラリのインストールが必要
  - [awslabs/aws-crt-java](#)
  - [awslabs/aws-crt-nodejs](#)
  - [awslabs/aws-crt-python](#)



# Route 53 ヘルスチェックの推奨設定

- EventBridge のメトリクス *IngestionToInvocationStartLatency* を利用する

”イベントが EventBridge によって取り込まれてから、ルール内のターゲットの最初の呼び出しまでに測定されたイベントの処理時間。これは、すべてのルールとバスで測定されるサービスレベルのメトリクスであり、EventBridge サービスのヘルスを示します。30 秒を超える長時間のレイテンシーは、サービスの中止を示している可能性があります。”
- CloudWatch アラームと Route 53 ヘルスチェックを作成する [AWS CloudFormation テンプレートのサンプル](#)
  - デフォルトの設定値では、1 分間平均で 30 秒を超えることが 5 分続いた場合、アラーム状態となる
  - 上記の構成の場合、RTO および RPO は 360~420 秒ほど
  - ユースケースに応じてカスタマイズする
- コンシューマーのメトリクスを指標に使うのは**非推奨**
  - 不要なフェイルオーバーが発生する可能性がある
  - コンシューマーの障害にも備えたい場合は、レプリケーションを有効化し、セカンダリリージョンにもあらかじめコンシューマーをデプロイしておく (Active/Active)

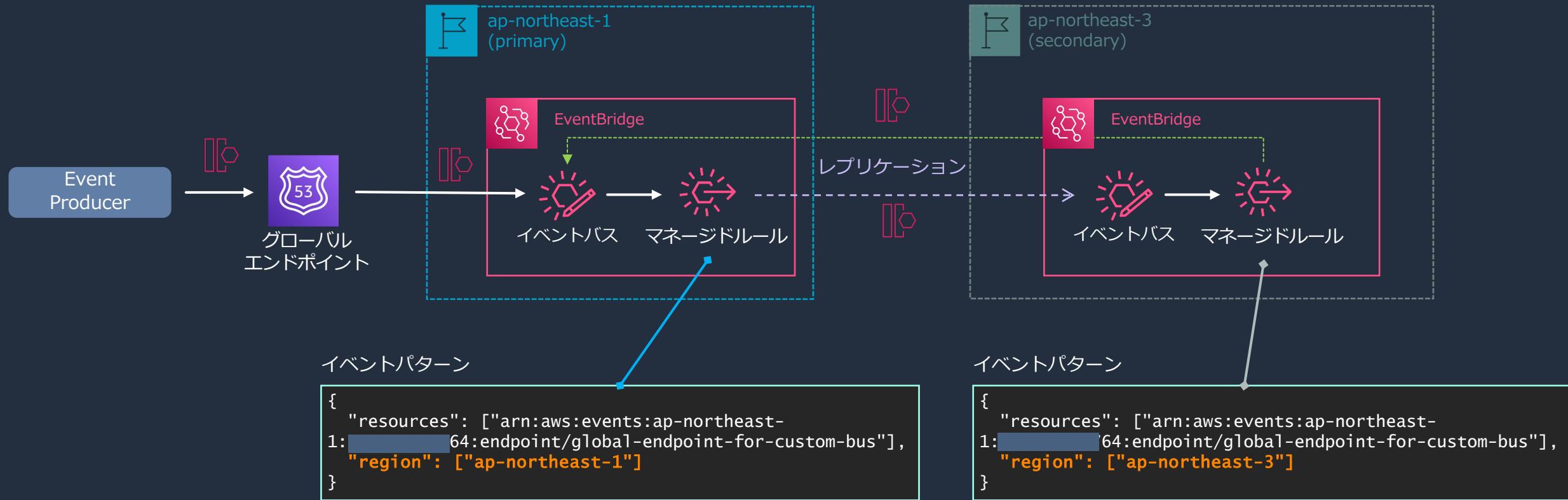
# レプリケーションについて

- プライマリリージョンで受け取ったイベントを、常時セカンダリリージョンにレプリケーションするオプション機能。基本的には**有効化が推奨**されている。
  - グローバルエンドポイントが正しく機能しているかの確認に利用できる
  - プライマリリージョンが復旧した際に、自動でフェイルバックすることができる
- 有効にするとマネージドな EventBridge ルールが両リージョンに作成され、**双方向の同期**が行われる。
  - 双方向と言っても、同時に動いているのは片方向のみ (プライマリ → セカンダリ)
  - プライマリ → セカンダリ → プライマリ のようなループはしない
- **非同期**で、**ニアリアルタイム**なレプリケーション
- 追加のコストが発生 (\*通常のカスタムイベントにかかる料金)

Q. フェイルオーバー直後は、影響があったリージョンへのレプリケーションに失敗するのでは？

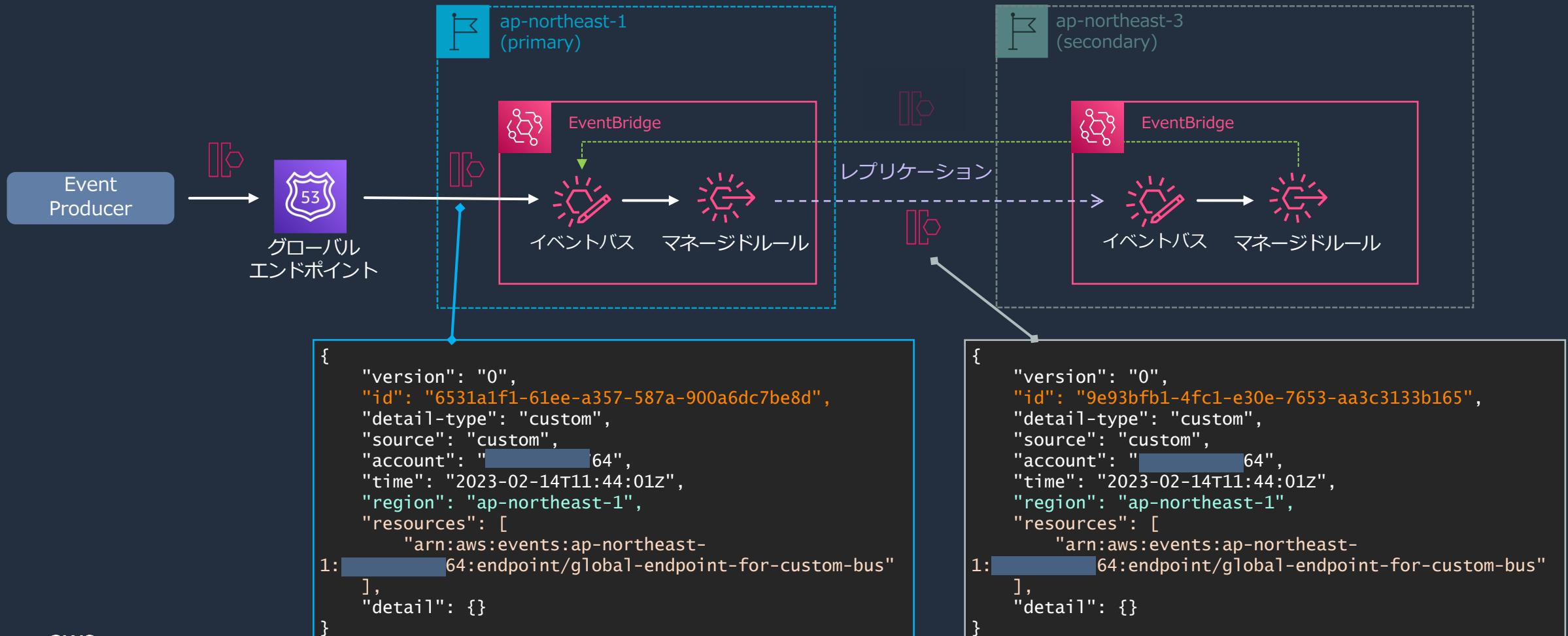
A. EventBridge に設定されたリトライポリシーによって最大 24 時間リトライが行われる。また、必要に応じてアーカイブの機能を使用してイベントを永続化することも検討する。

# レプリケーションの仕組み



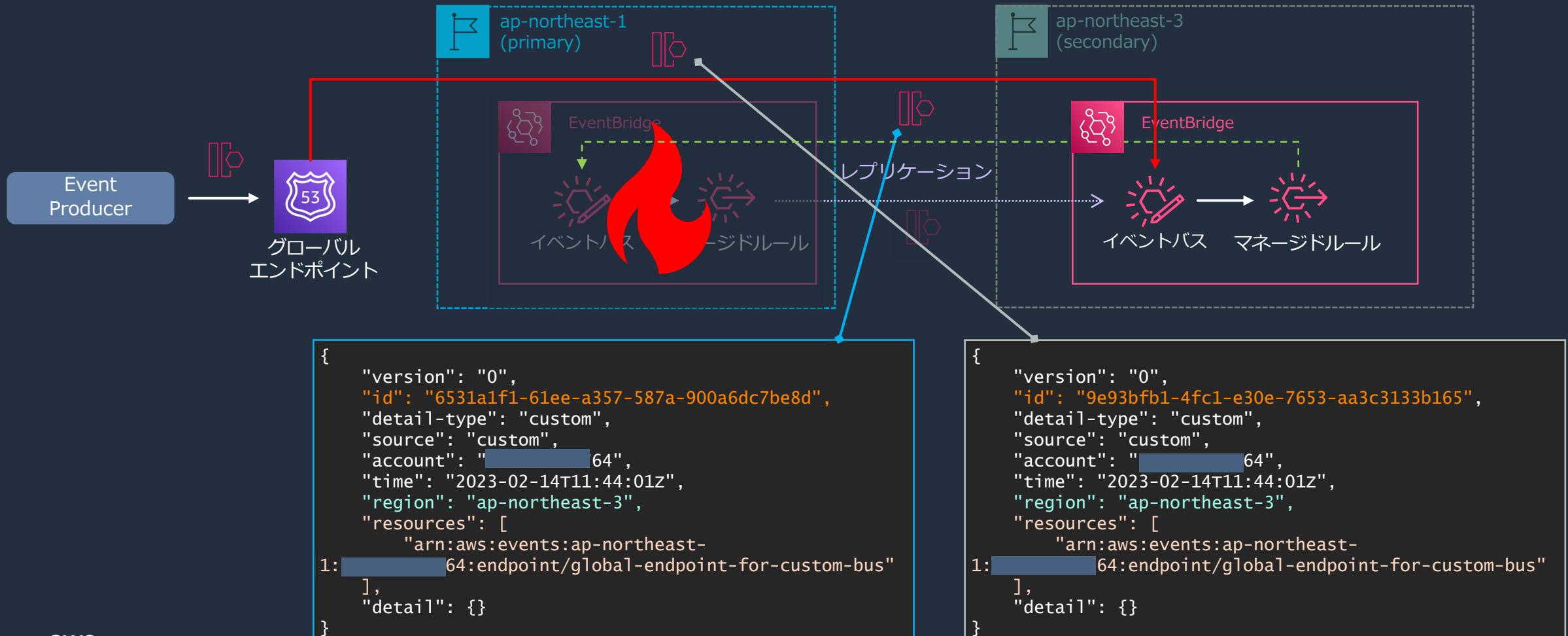
# イベントペイロードの仕様

1. id: PutEvents API の呼び出しごとにイベント ID は変わる
2. region: 最初に受信したイベントバスのリージョンから変わらない
3. resource: イベントバスではなくグローバルエンドポイント



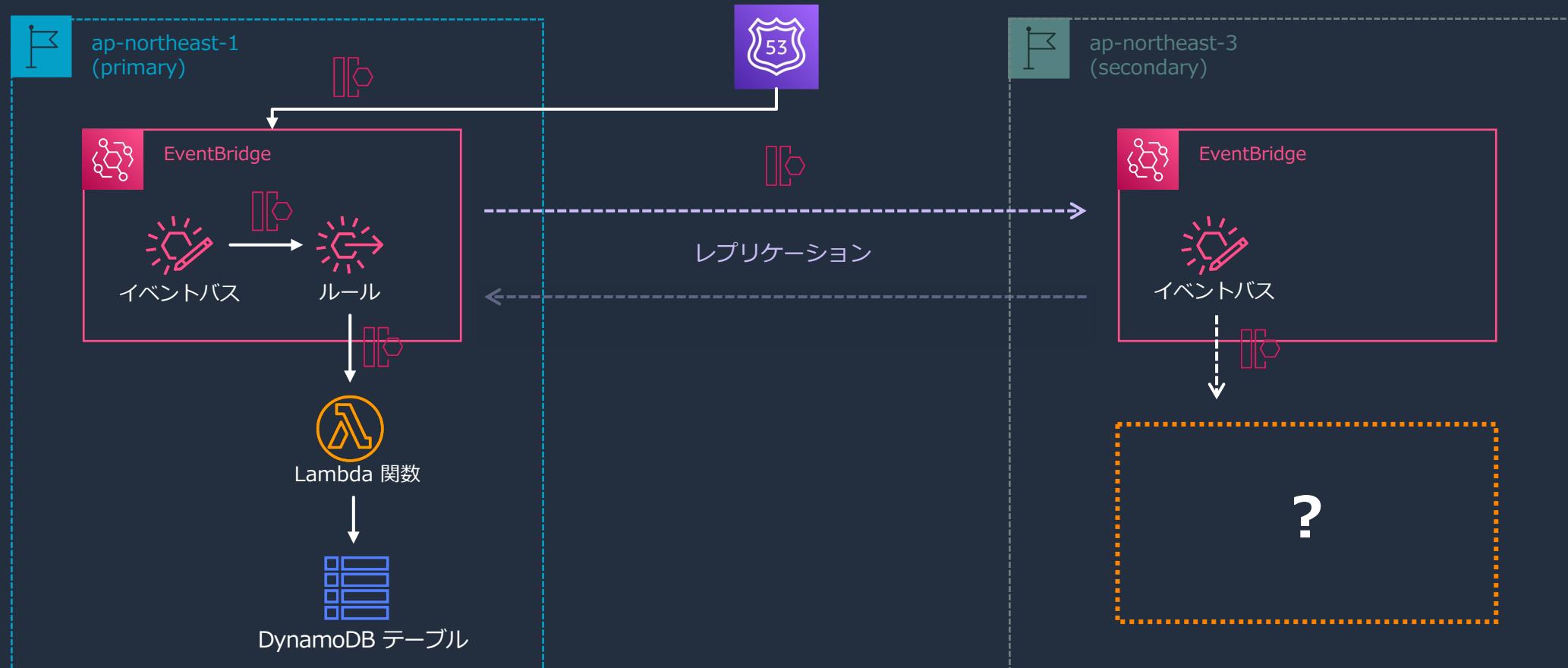
# イベントペイロードの仕様

1. id: PutEvents API の呼び出しごとにイベント ID は変わる
2. region: 最初に受信したイベントバスのリージョンから変わらない
3. resource: イベントバスではなくグローバルエンドポイント



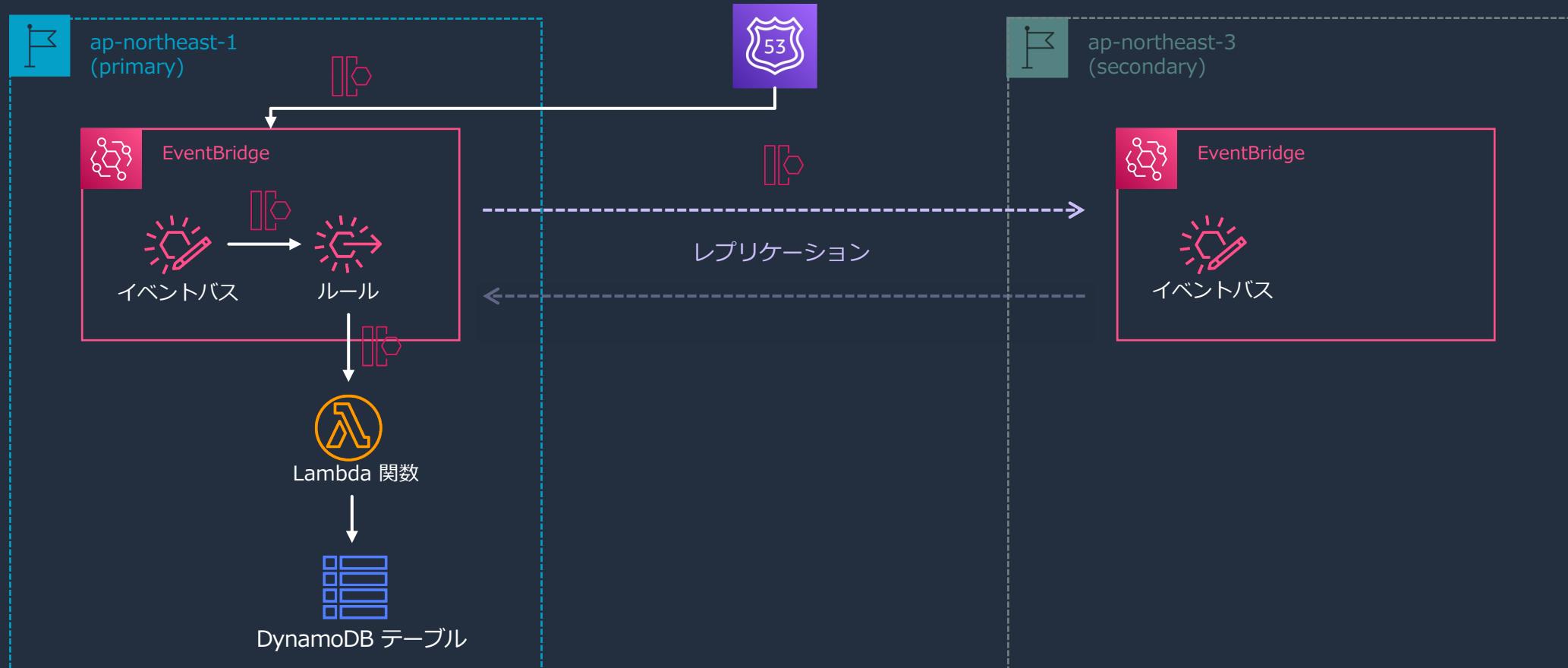
# レプリケートされたイベントの取り扱いについて考える

- 1-1. Active/Archive パターン
- 1-2. Active/Archive with EventBridge Archive パターン
- 2. Active/Active パターン



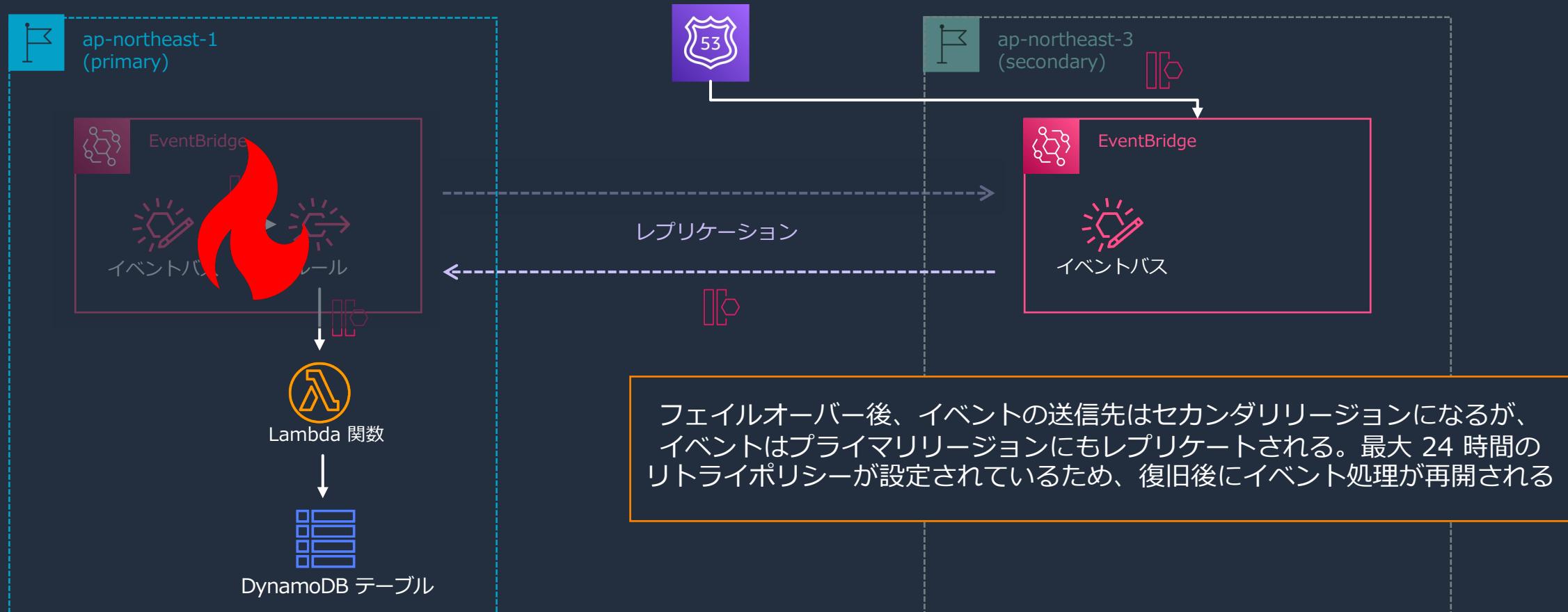
# 1-1. Active/Archive パターン

レプリケートされたイベントには何も行わない



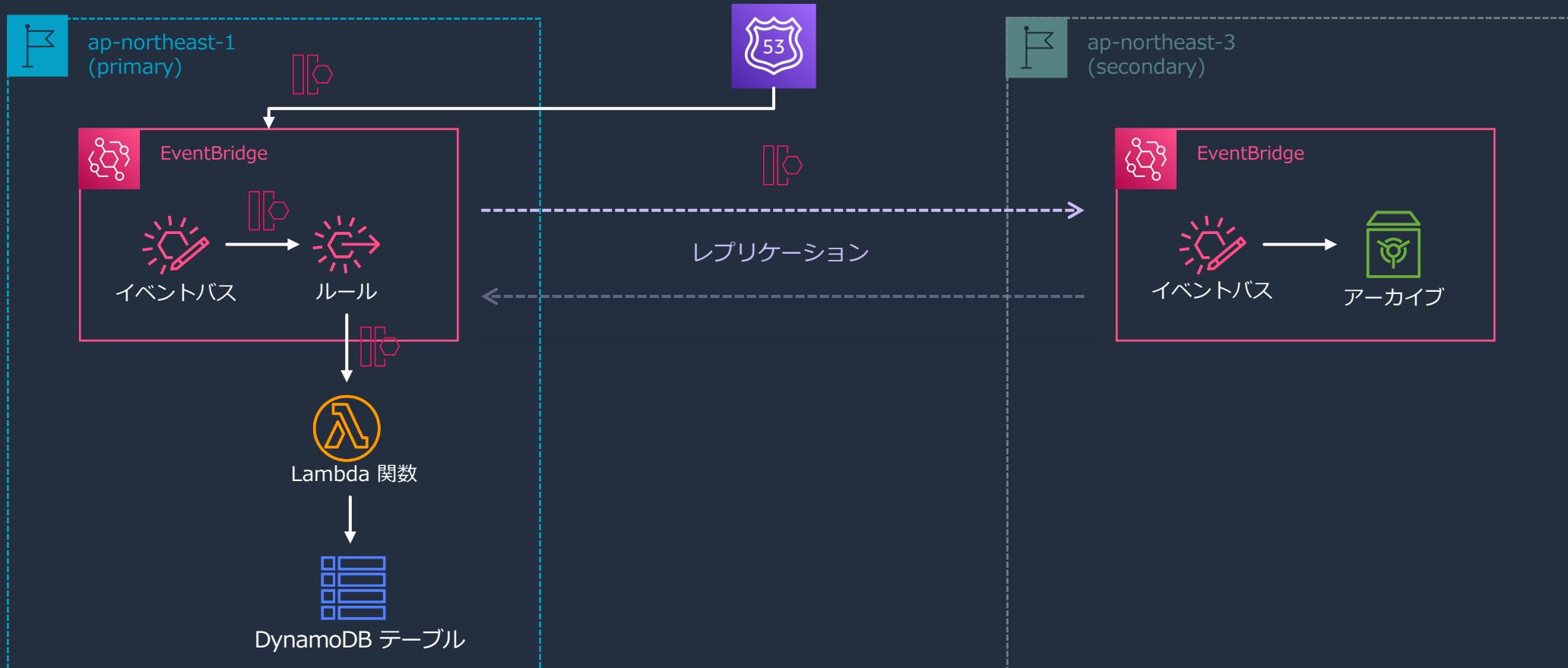
# 1-1. Active/Archive パターン

レプリケートされたイベントには何も行わない



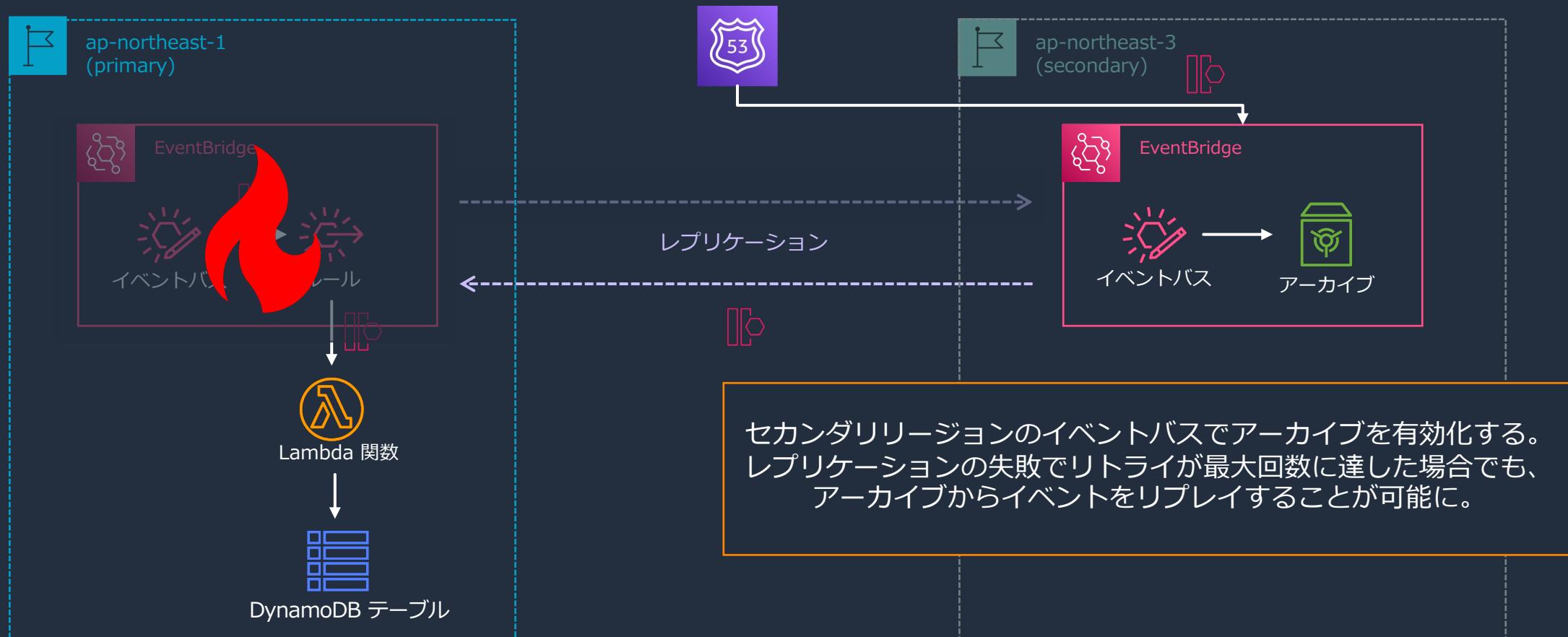
# 1-2. Active/Archive with EventBridge Archive パターン

セカンダリリージョンに送信されたイベントをアーカイブする



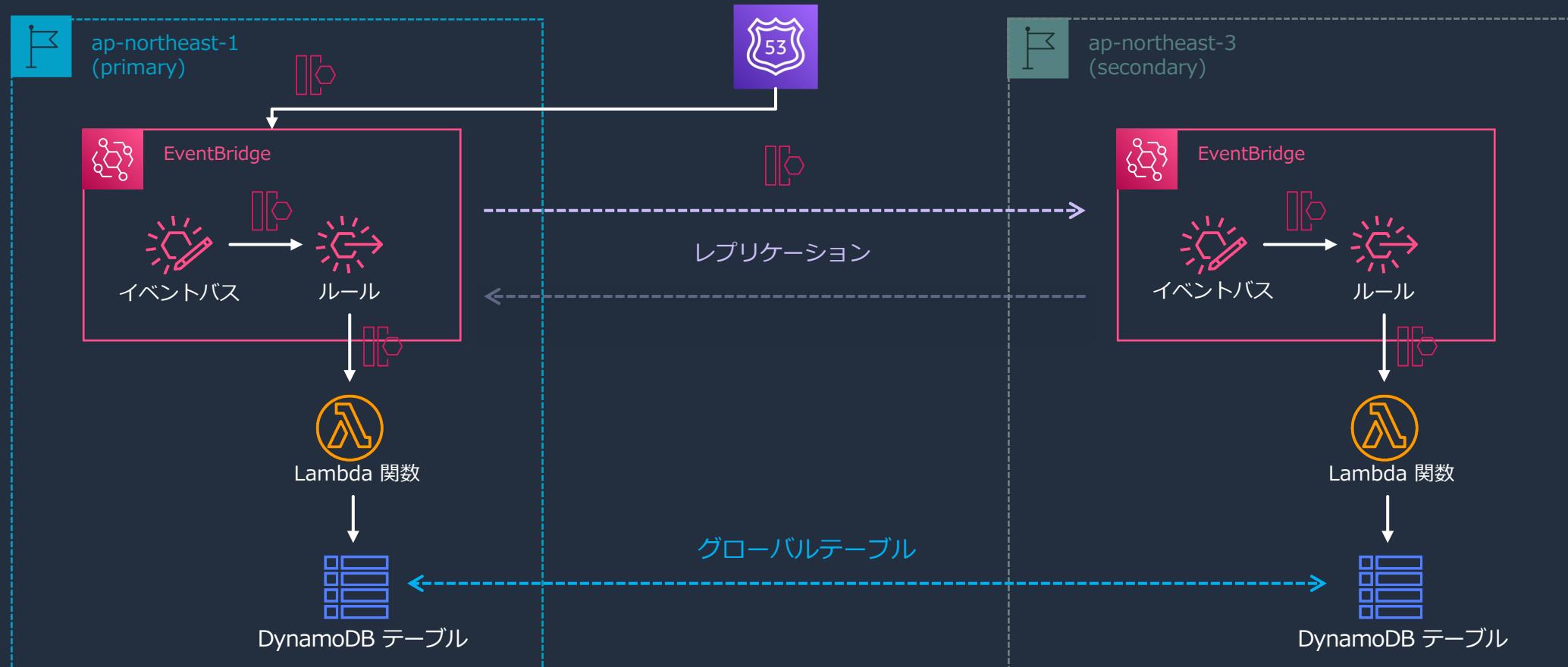
# 1-2. Active/Archive with EventBridge Archive パターン

セカンダリリージョンに送信されたイベントをアーカイブする



## 2. Active/Active パターン

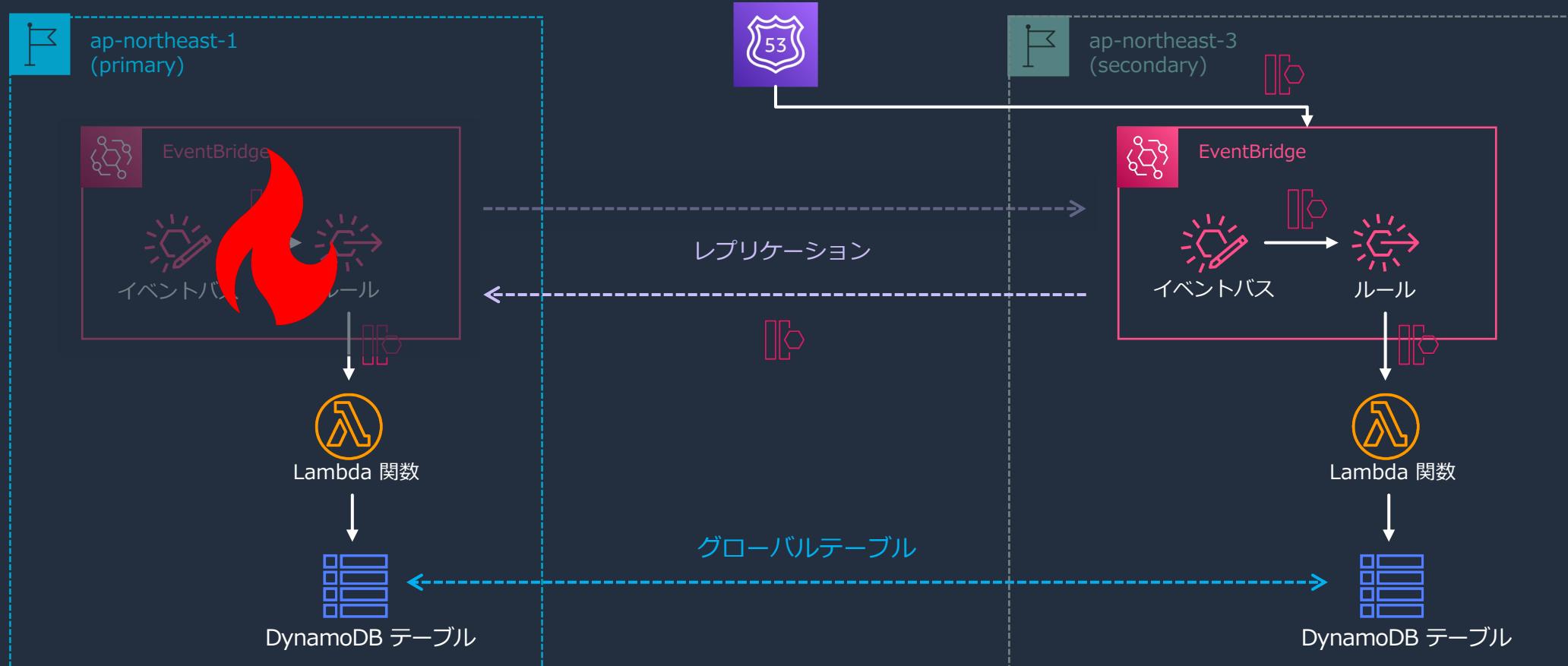
セカンダリリージョンでもイベントを処理する



## 2. Active/Active パターン

セカンダリリージョンでもイベントを処理する

セカンダリリージョンにルールとコンシューマーをデプロイしておくことで、フェイルオーバー後もイベント処理が継続される。復旧後のデータ不整合を防ぐために、データレイヤーの同期を別途検討しておく必要がある。



## 2. Active/Active パターンにおける注意点

- イベントパターンを利用して、コンシューマと同一のリージョンに起因するイベントのみを処理するようにルールを構成する

プライマリリージョンのルールにおけるイベントパターン

```
{  
  "region": ["ap-northeast-1"]  
}
```

セカンダリリージョンのルールにおけるイベントパターン

```
{  
  "region": ["ap-northeast-3"]  
}
```

- ビジネスロジックを幕等に実装する
  - \* そもそもイベントの配信セマンティクスは at-least-once
- 関連するデータストアの同期を設定してデータの整合性を保つ
  - ex. DynamoDB グローバルテーブル、S3 クロスリージョンレプリケーション (CRR)

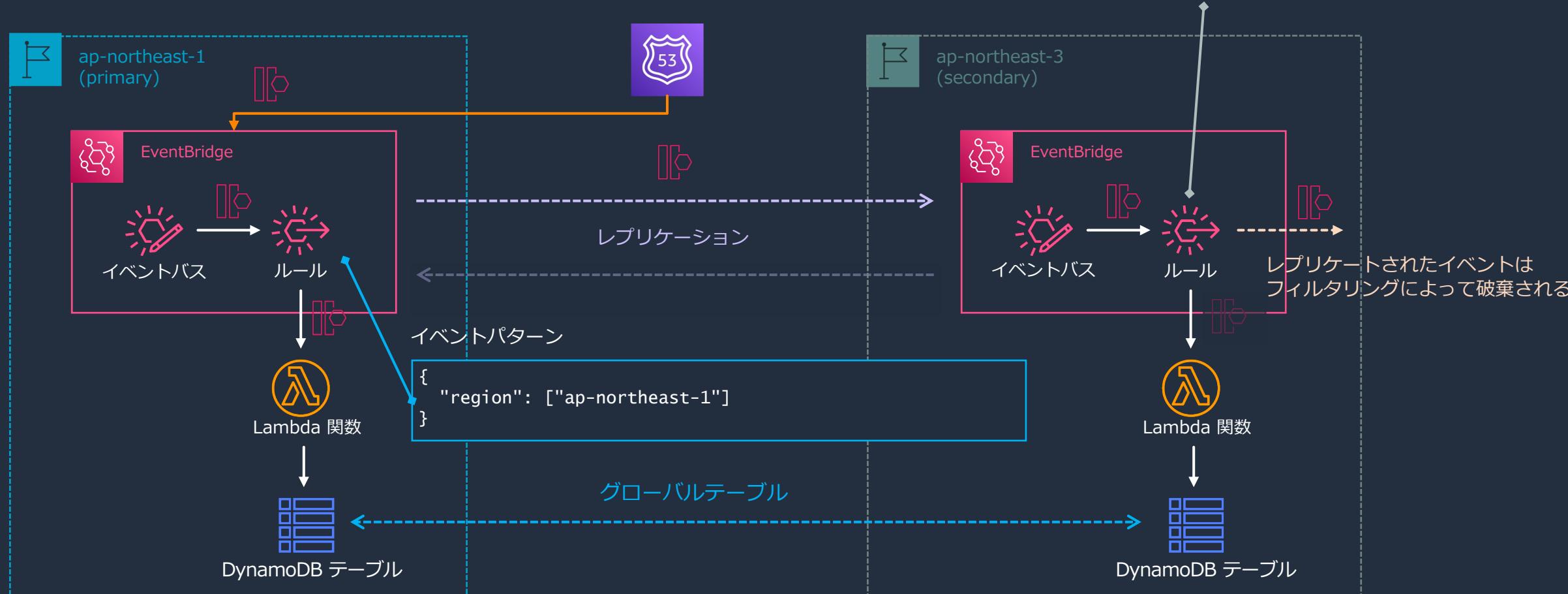
\* コンシューマーの可用性を高めたいかつ処理の幕等性が担保できれば、上述のイベントパターンによるフィルタリングを設定しないという方法もあり

## 2. Active/Active パターン

セカンダリリージョンでもイベントを処理する

イベントパターン

```
{  
  "region": ["ap-northeast-3"]  
}
```



# その他の留意事項

- ・ 東京、大阪を含む 18 のリージョンで利用可能
- ・ カスタムイベントのみをサポート
- ・ クロスアカウントは未対応
- ・ VPC エンドポイントは未対応
- ・ 無料で利用可能 (\*レプリケーション, CloudWatch, Route 53 は別途料金が発生)
- ・ グローバルエンドポイント自体はリージョナルなリソースのため、プライマリのリージョンで作成する
- ・ プライマリおよびセカンダリのイベントバスは同一の名前で作成する必要がある
- ・ PutEvents API の上限緩和を行なっている場合は、セカンダリリージョンのスループットも要確認

\* 詳しくは[ドキュメント](#)をご覧ください

# AWS Black Belt Online Seminar とは

- ・ 「サービス別」「ソリューション別」「業種別」などのテーマに分け、  
アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナー  
シリーズです
- ・ AWS の技術担当者が、AWS の各サービスやソリューションについてテーマ  
ごとに動画を公開します
- ・ 以下の URL より、過去のセミナー含めた資料などをダウンロードするこ  
とができます
  - ・ <https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/>
  - ・ <https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY>



ご感想は X (Twitter) へ！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 2023 年 10 月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。AWS のサービスは常にアップデートを続いているため、最新の情報は AWS 公式ウェブサイト (<https://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には別途消費税をご請求させていただきます
- ・ 技術的な内容に関しましては、有料の [AWS サポート窓口](#)へお問い合わせください
- ・ 料金面でのお問い合わせに関しましては、[カスタマーサポート窓口](#)へお問い合わせください（マネジメントコンソールへのログインが必要です）



# Thank you!



# Amazon EventBridge Scheduler

櫻谷 広人

Partner Solutions Architect  
2023/8

# AWS Black Belt Online Seminar とは

- ・ 「サービス別」「ソリューション別」「業種別」などのテーマに分け、  
アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナー  
シリーズです
- ・ AWS の技術担当者が、AWS の各サービスやソリューションについてテーマ  
ごとに動画を公開します
- ・ 以下の URL より、過去のセミナー含めた資料などをダウンロードするこ  
とができます
  - ・ <https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/>
  - ・ <https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY>



ご感想は Twitter へ！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 2023 年 8 月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。AWS のサービスは常にアップデートを続けているため、最新の情報は AWS 公式ウェブサイト (<https://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には別途消費税をご請求させていただきます
- ・ 技術的な内容に関しましては、有料の [AWS サポート窓口](#)へお問い合わせください
- ・ 料金面でのお問い合わせに関しましては、[カスタマーサポート窓口](#)へお問い合わせください（マネジメントコンソールへのログインが必要です）

# 自己紹介

名前：櫻谷 広人 (Hirotto Sakuraya)

所属：AWS Technology Partnerships

SaaS, Partner Solutions Architect

経歴：主にバックエンドエンジニアとして Web サービスやネイティブアプリの開発に従事。前職では CtoC のスタートアップで執行役員 CTO を務める。

好きな AWS サービス：Amazon EventBridge



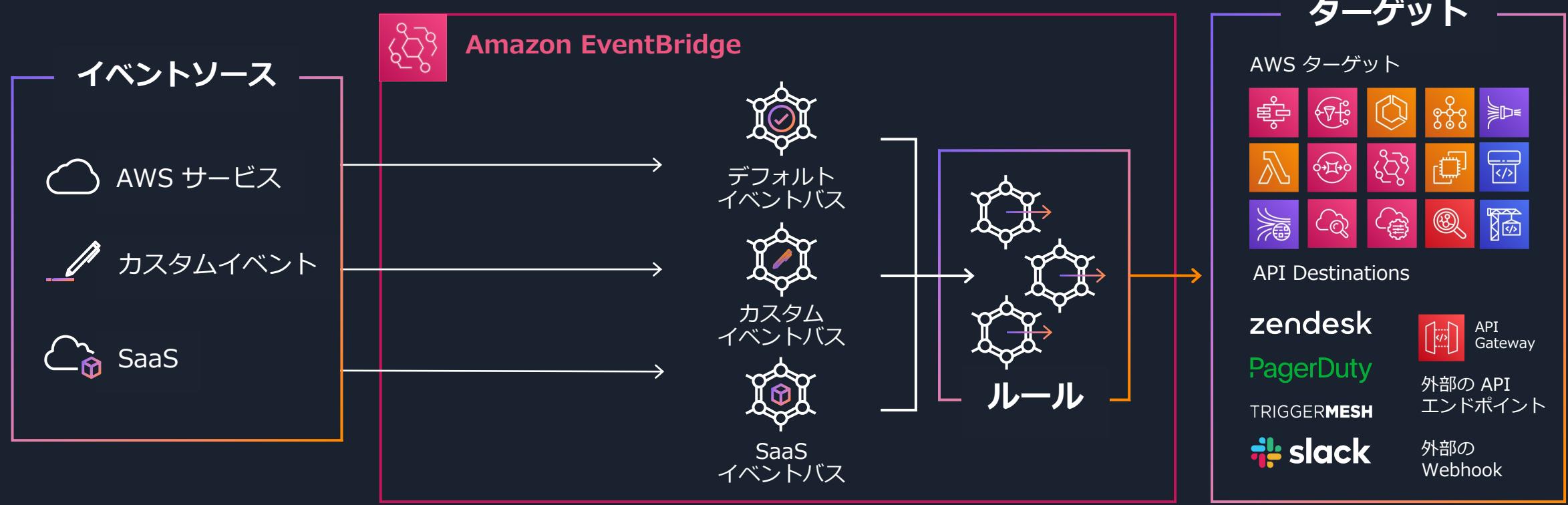
# 本セミナーの対象者

- Amazon EventBridge について深く学びたい方
- イベント駆動型アプリケーションの開発に興味をお持ちの方
- クラウドに最適なジョブ管理システム、スケジューラーをお探しの方

# 本セミナーの取り扱う範囲

- Amazon EventBridge Scheduler について
- \* Amazon EventBridge 全体または他の機能については別のセミナー動画をご覧ください

# Amazon EventBridge とは



アプリケーション、統合された SaaS アプリケーション、および AWS のサービスから生成されたイベントを受信、フィルタリング、変換、ルーティング、および配信することができるサーバーレスイベントバス。大規模なイベント駆動型アプリケーションの開発を可能に。

# Amazon EventBridge Scheduler とは

- マネージドなサーバレススケジューラー
- タスクの作成、実行、管理を行うことができる
  - 例：毎日 0 時に特定の Lambda 関数を実行
  - 例：2023/12/31 15:00 に EC2 インスタンスを停止
- 270+ の AWS サービス、6000+ の API アクションを呼び出すことが可能
- 少なくとも 1 回 (at-least-once) の信頼性のある配信を保証
- リトライポリシー、デッドレターキューを設定可能
- マネジメントコンソール、AWS CLI、AWS SDK から設定可能
- ユースケース：運用の自動化、バッチ処理、ビジネスロジックの遅延実行
  - ex. ユーザーが解約を実行 → 残りの契約期間が過ぎた後にユーザー削除等を実施



# EventBridge Scheduler のユースケース



# EventBridge スケジュールルールとの違いは？

- スケジュールルールによって、これまで似たようなことはできていた
- EventBridge Scheduler は、タイムゾーンのサポートやより高度なカスタマイズ性、スケーラビリティなどを備えた上位機能という位置付け
- 今後スケジュールベースでターゲットを呼び出す場合は、EventBridge Scheduler の方を利用することが推奨されている
- 既存の EventBridge スケジュールルールは今後も利用し続けることが可能
- サービスの API としては異なるので注意
  - EventBridge  $\Leftrightarrow$  **events.us-east-1.amazonaws.com**
  - EventBridge Scheduler  $\Leftrightarrow$  **scheduler.us-east-1.amazonaws.com**
- マネジメントコンソールでは、どちらの機能も EventBridge コンソールに集約されている

# EventBridge Scheduler の強み



UTC 以外の様々な  
**タイムゾーン**および  
**サマータイム**をサポート



**1回限りの実行**  
をサポート



**数百万のスケジュール**  
を管理可能



**270+のサービス、6000+の API**  
をターゲットに指定可能



# EventBridge Scheduler で設定可能なスケジュールタイプ

## 1. Rate-based

指定の間隔で繰り返し定期実行

`rate(5 minutes)`

`rate(12 hours)`

`rate(1 days)`

\* 秒レベルの実行は未サポート

\* 分 / 時間 / 日 のみ選択可能

## 2. Cron-based

cron 式による複雑な指定

`cron(0 8 * * * *)`

`cron(15 10 ? * 6L 2022-2023)`

\* 各種ワイルドカードをサポート

\* UTC または特定のタイムゾーンを指定

## 3. One-time

指定のタイミングで 1 回だけ実行

`at(2022-11-20T13:00:00)`

\* 秒は指定しても 0 に切り捨てられる

\* 有効な日付および時間を指定

\* UTC または特定のタイムゾーンを指定



# タイムゾーンについて

- IANA によってメンテナンスされている [タイムゾーンデータベース](#) を使用  
ex. America/New\_York, Asia/Tokyo, Japan
- サマータイムが導入されているタイムゾーンでは以下のルールが適用される
  - サマータイム開始時 ⇌ 存在しない時間に指定されているスケジュールはスキップ
  - サマータイム終了時 ⇌ 2回存在する時間に指定されているスケジュールは1度だけ実行

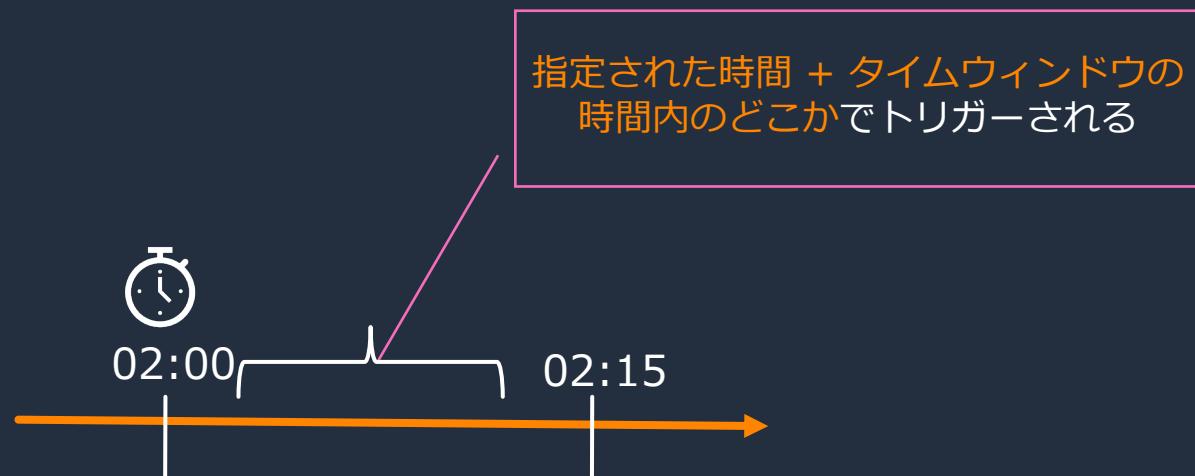


# フレックストタイムウィンドウ

- ・(指定した時間+タイムウィンドウ)内でランダムに実行タイミングを分散させる仕組み
- ・ダウンストリームのサービスのスロットリング、過負荷を防ぐためなどの目的で利用される
- ・時間ぴったりに起動する必要がなければ使っておくのがおすすめ



オフの場合



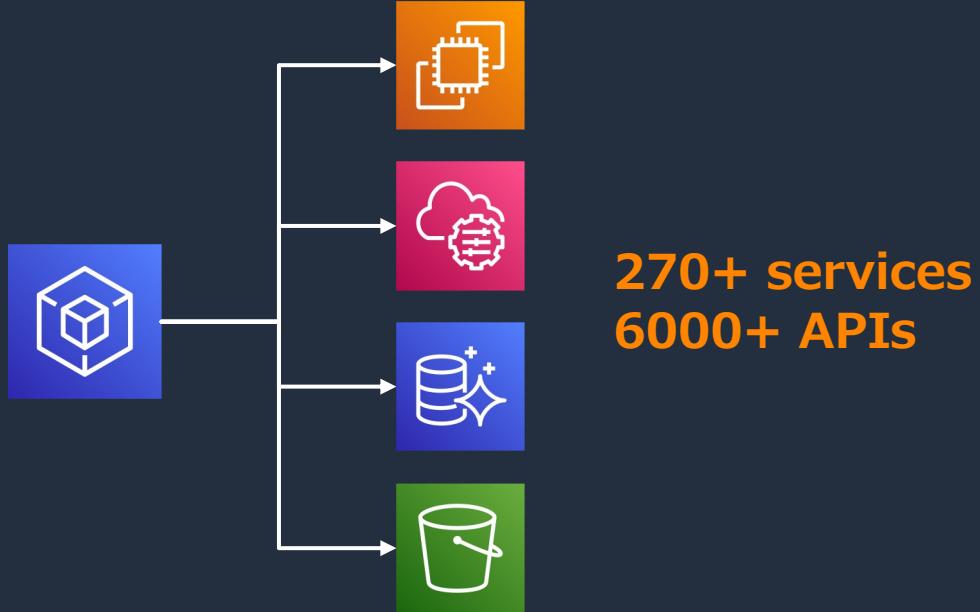
オンの場合

\* 15 分、30 分、1 時間、2 時間、4 時間から選択可能

# ターゲットとして指定可能な AWS サービス



AWS CodeBuild: StartBuild  
Amazon ECS: RunTask  
AWS Lambda: Invoke  
Amazon SQS: SendMessage  
AWS Step Functions: StartExecution  
etc.



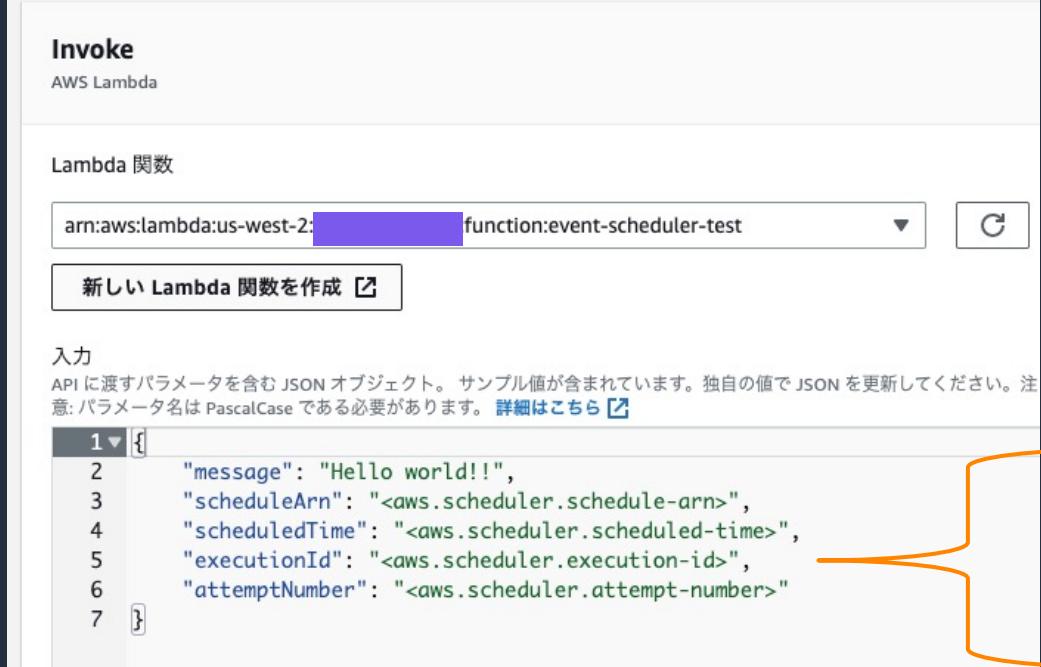
## 1. Templated targets

- Lambda 関数の呼び出しや SQS キューへのメッセージ送信など、よく使われる一部の API セットをサポート
- 簡単な設定で利用を開始できる

## 2. Universal targets

- AWS SDK により幅広いサービスおよび API をサポート
- パラメータを自分でカスタマイズ
- サポートされているサービス一覧

# Templated targets の設定例



例えば Lambda Invoke の場合、呼び出す関数を指定し、渡す入力値を設定するだけで利用可能

入力値では、事前定義されているコンテキスト属性も利用可能

- <aws.scheduler.schedule-arn> - スケジュールの ARN
- <aws.scheduler.scheduled-time> - スケジュールされている時間
- <aws.scheduler.execution-id> - 呼び出しごとに一意の ID
- <aws.scheduler.attempt-number> - 呼び出し試行回数のカウンター

Lambda 関数が受け取った event

```
{  
  message: "Hello world!!",  
  scheduleArn: 'arn:aws:scheduler:us-west-2:1234567890:schedule/default/test-schedule',  
  scheduledTime: '2023-08-01T12:00:00Z',  
  executionId: '9a64c9f4-a02a-42e2-9cfb-a6d88dda320c',  
  attemptNumber: '1'  
}
```



# Universal targets の設定例

The screenshot shows the AWS Lambda function configuration interface. At the top, there's a header with the function name and a blue dot icon. Below it, the function name is listed again with a smaller blue dot icon. The main area is titled "StartInstances" and "Amazon EC2". It contains an "Input" section with a JSON code editor showing the following code:

```
1 {  
2   "InstanceIds": [  
3     "MyData"  
4   ]  
5 }
```

Below this is another "PutItem" target for "DynamoDB" with its own "Input" section containing this JSON code:

```
1 {  
2   "Item": {  
3     "id": {  
4       "S": "<aws.scheduler.execution-id>"  
5     },  
6     "scheduledTime": {  
7       "S": "<aws.scheduler.scheduled-time>"  
8     },  
9   },  
10  "TableName": "EventBridgeSchedule"  
11 }
```

The AWS logo is visible in the bottom left corner.

- 各 API のパラメータとして必要なものを JSON で入力
- Templated targets と同様にコンテキスト属性も利用可能

An orange arrow points from the Lambda function configuration to the right side of the slide, where a screenshot of a DynamoDB table is shown. The table has columns for "id" and "scheduledTime". The data consists of eight rows of scheduled events:

	id	scheduledTime
	482ac6d50f5e07b1	2022-11-24T12:37:00Z
	482ac6d5ab3a4a8c	2022-11-24T12:38:00Z
	482ac6d5833b06ae	2022-11-24T12:39:00Z
	482ac6d596eda434	2022-11-24T12:40:00Z
	482ac6d58ee8e675	2022-11-24T12:41:00Z
	482ac6d5cd8a1f74	2022-11-24T12:42:00Z
	482ac6d535454e9a	2022-11-24T12:43:00Z
	482ac6d530718262	2022-11-24T12:44:00Z

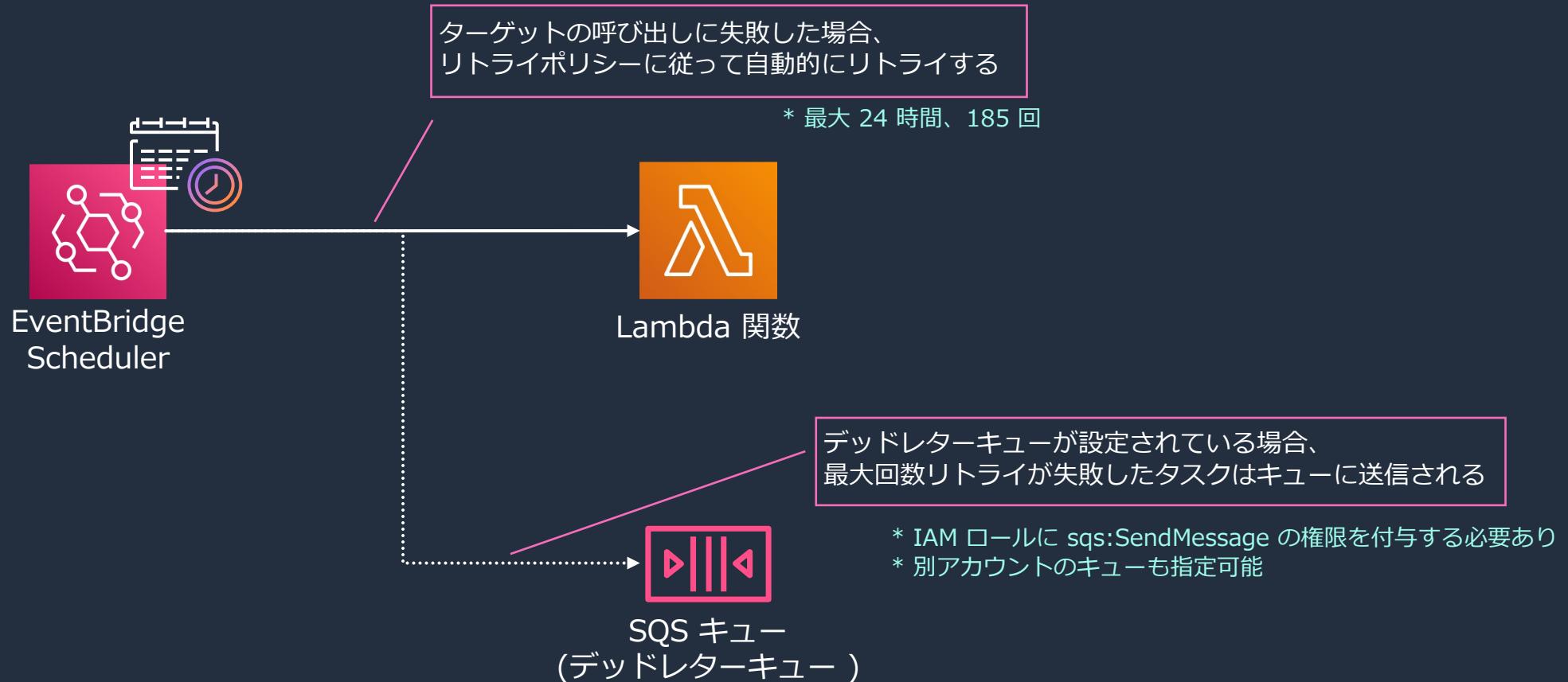
# IAM ロールの設定

ターゲットとして指定したサービス API の実行権限を持った IAM ロールを指定する必要がある

```
{  
  "version": "2012-10-17",  
  "statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "lambda:InvokeFunction"  
      ],  
      "Resource": [  
        "arn:aws:lambda:us-west-2:xxxxx:function:event-scheduler-test:*",  
        "arn:aws:lambda:us-west-2:xxxxx:function:event-scheduler-test"  
      ]  
    }  
  ]  
}
```



# エラーに対する回復性



# EventBridge Scheduler の各種クオータ

- 作成可能なスケジュールグループ数：アカウントあたり最大 **500** まで  
(\*スケジュールのグループ化に使用)
- 作成可能なスケジュール数：アカウントあたり最大 **1,000,000** まで  
(cf. EventBridge Rules: リージョンあたり 300 ルールまで登録可能)
- 指定可能なターゲット数：スケジュールあたり **1** つのみ  
(cf. EventBridge Rules: 1 ルール内で最大 5 ターゲットまで指定可能)
- CreateSchedule API のリクエストレート：最大 **50 TPS**
- 呼び出しスループット：最大 **500 TPS**  
(\*以降はスロットリングされて遅延する)

など

\* 2023/08/01 時点

\* 上限緩和可能なものもあり

\* [Quotas for Amazon EventBridge Scheduler](#)



# EventBridge Scheduler の料金

- スケジュールの呼び出し回数に応じて課金
- 1ヶ月あたり 14,000,000 回までは無料で利用可能
- その後 1,000,000 回呼び出しごと料金が発生
  - 例：オレゴンリージョンの場合：\$ 1.00
  - 例：東京リージョンの場合：\$ 1.25

\* 2023/08/01 時点



# モニタリング

## 取得可能なメトリクス

InvocationAttemptCount	EventBridge Scheduler がスケジュールを呼び出そうとした数
TargetErrorCount	ターゲットからエラーが返され、呼び出しに失敗した数
TargetErrorThrottledCount	ターゲット側でスロットリングされ、呼び出しに失敗した数
InvocationThrottleCount	EventBridge Scheduler 側でスロットリングされた数
InvocationDroppedCount	最大リトライ回数を超えて呼び出しに失敗し、呼び出しを中止した数

DLQ を設定している場合、以下のメトリクスも利用可能

- InvocationsSentToDeadLetterCount
- InvocationsFailedToBeSentToDeadLetterCount
- InvocationsFailedToBeSentToDeadLetterCount\_<error\_code>
- InvocationsSentToDeadLetterCount\_Truncated\_MessageSizeExceeded

\* ログについては CloudTrail による API call の記録のみ  
(+呼び出し先のサービスのログを活用)





# Thank you!



# Amazon EventBridge Pipes

櫻谷 広人

Partner Solutions Architect  
2024/3

# AWS Black Belt Online Seminar とは

- ・ 「サービス別」「ソリューション別」「業種別」などのテーマに分け、  
アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナー  
シリーズです
- ・ AWS の技術担当者が、AWS の各サービスやソリューションについてテーマ  
ごとに動画を公開します
- ・ 以下の URL より、過去のセミナー含めた資料などをダウンロードするこ  
とができます
  - ・ <https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/>
  - ・ <https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY>



ご感想は X (Twitter) へ！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 2024 年 3 月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。AWS のサービスは常にアップデートを続けているため、最新の情報は AWS 公式ウェブサイト (<https://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には別途消費税をご請求させていただきます
- ・ 技術的な内容に関しましては、有料の [AWS サポート窓口](#)へお問い合わせください
- ・ 料金面でのお問い合わせに関しましては、[カスタマーサポート窓口](#)へお問い合わせください（マネジメントコンソールへのログインが必要です）

# 自己紹介

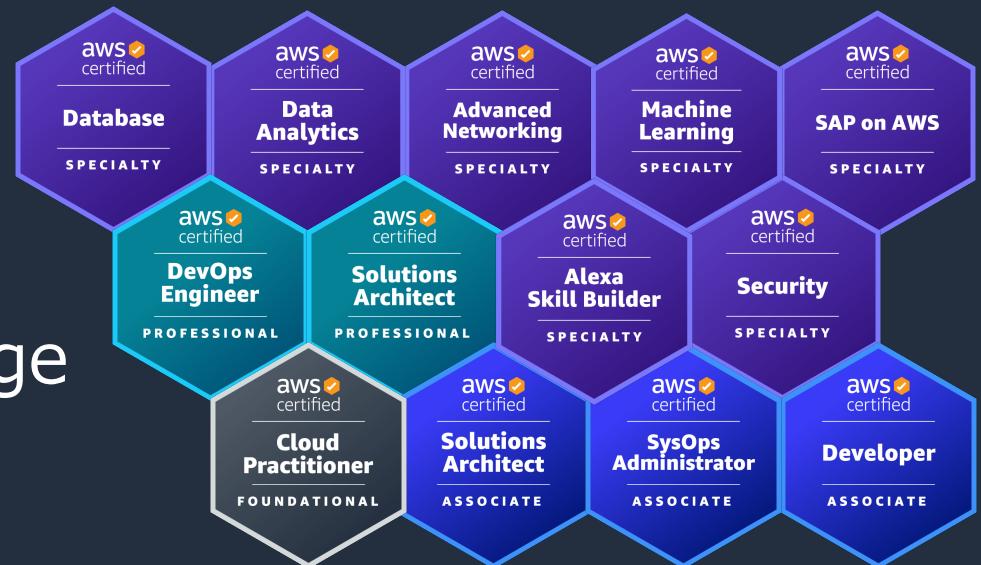
名前：櫻谷 広人 (Hirotto Sakuraya)

所属：AWS Technology Partnerships

SaaS, Partner Solutions Architect

経歴：主にバックエンドエンジニアとして Web サービスやネイティブアプリの開発に従事。前職では CtoC のスタートアップで執行役員 CTO を務める。

好きな AWS サービス：Amazon EventBridge



# 本セミナーの対象者

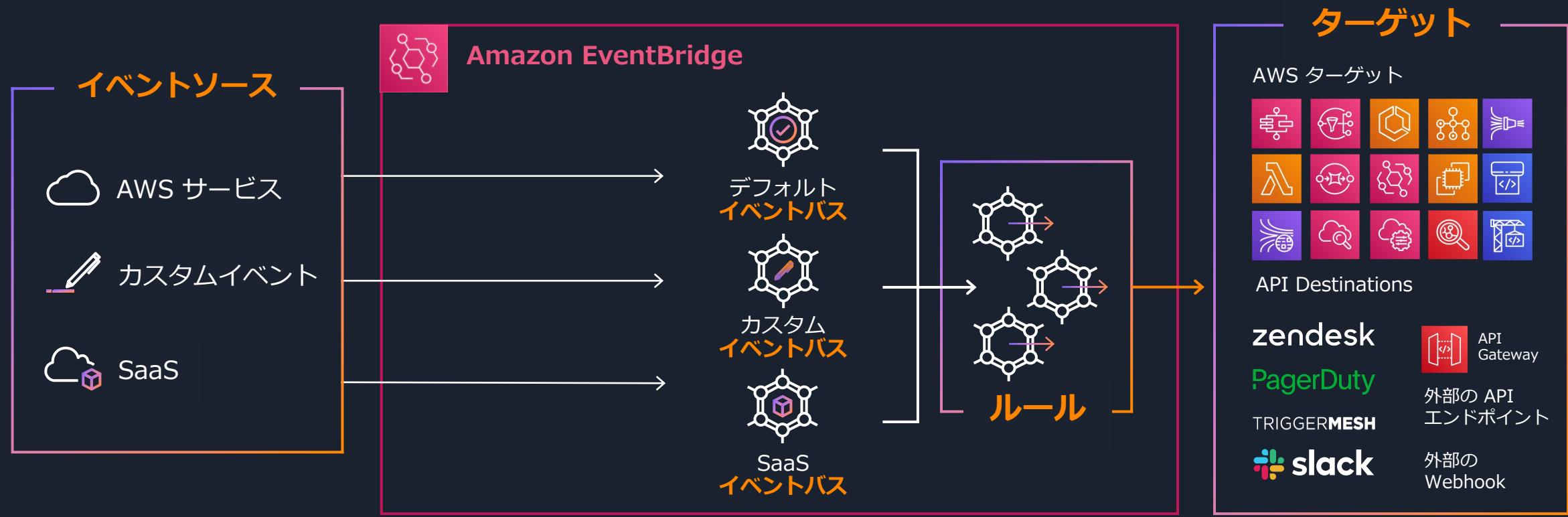
- Amazon EventBridge について深く学びたい方
- イベント駆動型アプリケーションの開発に興味をお持ちの方
- ノーコード/ローコードでのシステム間連携を実現したい方

# 本セミナーの取り扱う範囲

- Amazon EventBridge Pipes について
- \* Amazon EventBridge の他の機能については別のセミナー動画をご覧ください

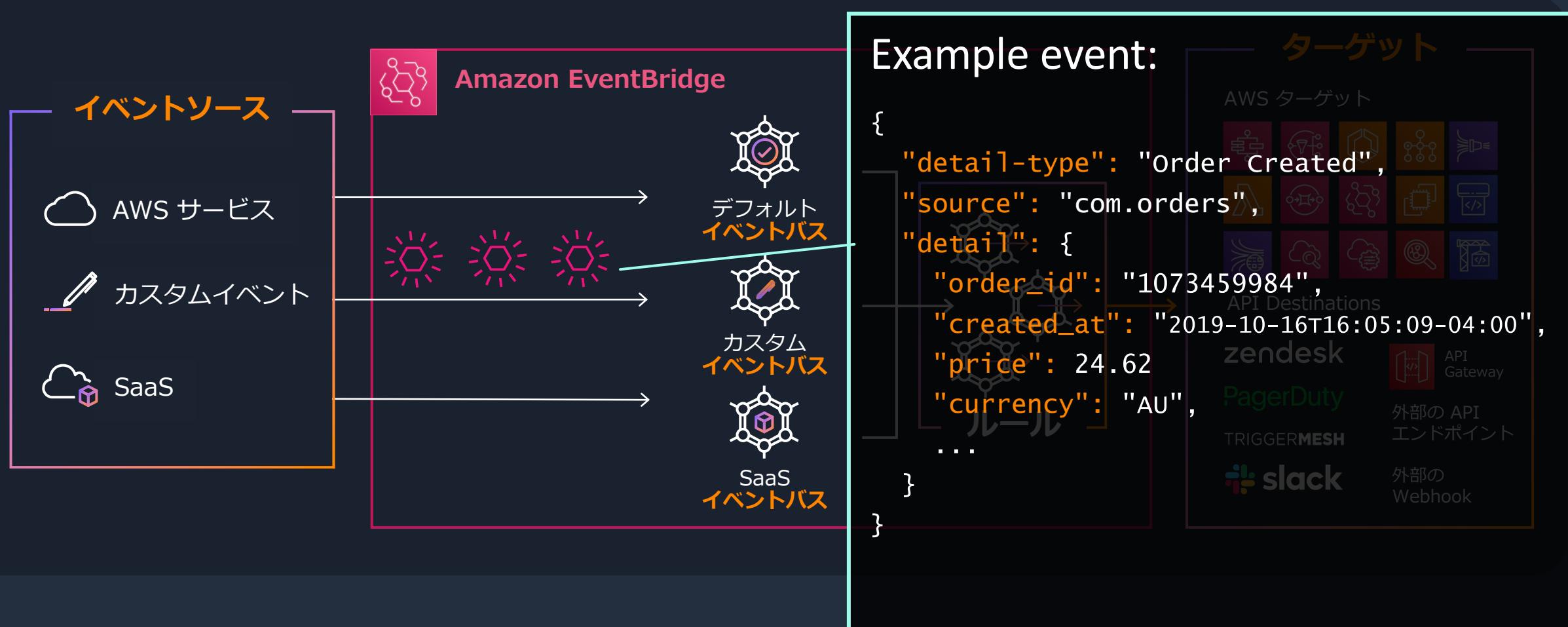
- [Amazon EventBridge グローバルエンドポイント 【AWS Black Belt】](#)
- [Amazon EventBridge Scheduler 【AWS Black Belt】](#)

# Amazon EventBridge とは

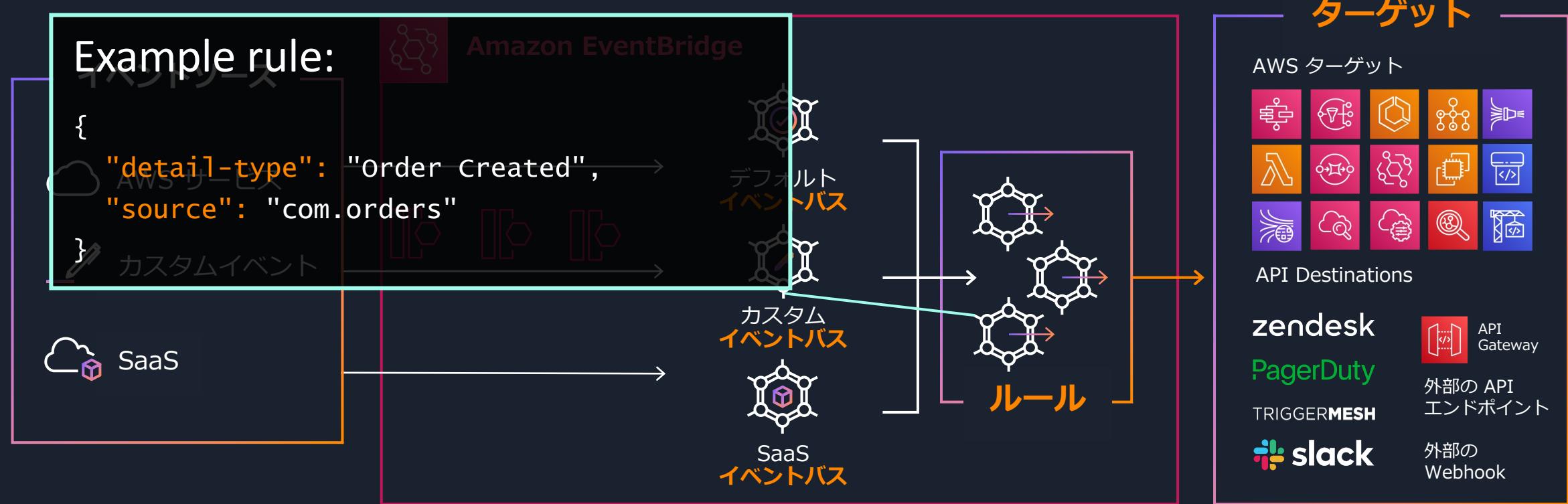


アプリケーション、統合された SaaS アプリケーション、および AWS のサービスから生成されたイベントを受信、フィルタリング、変換、ルーティング、および配信することができるサーバーレスイベントバス。大規模なイベント駆動型アプリケーションの開発を可能に。

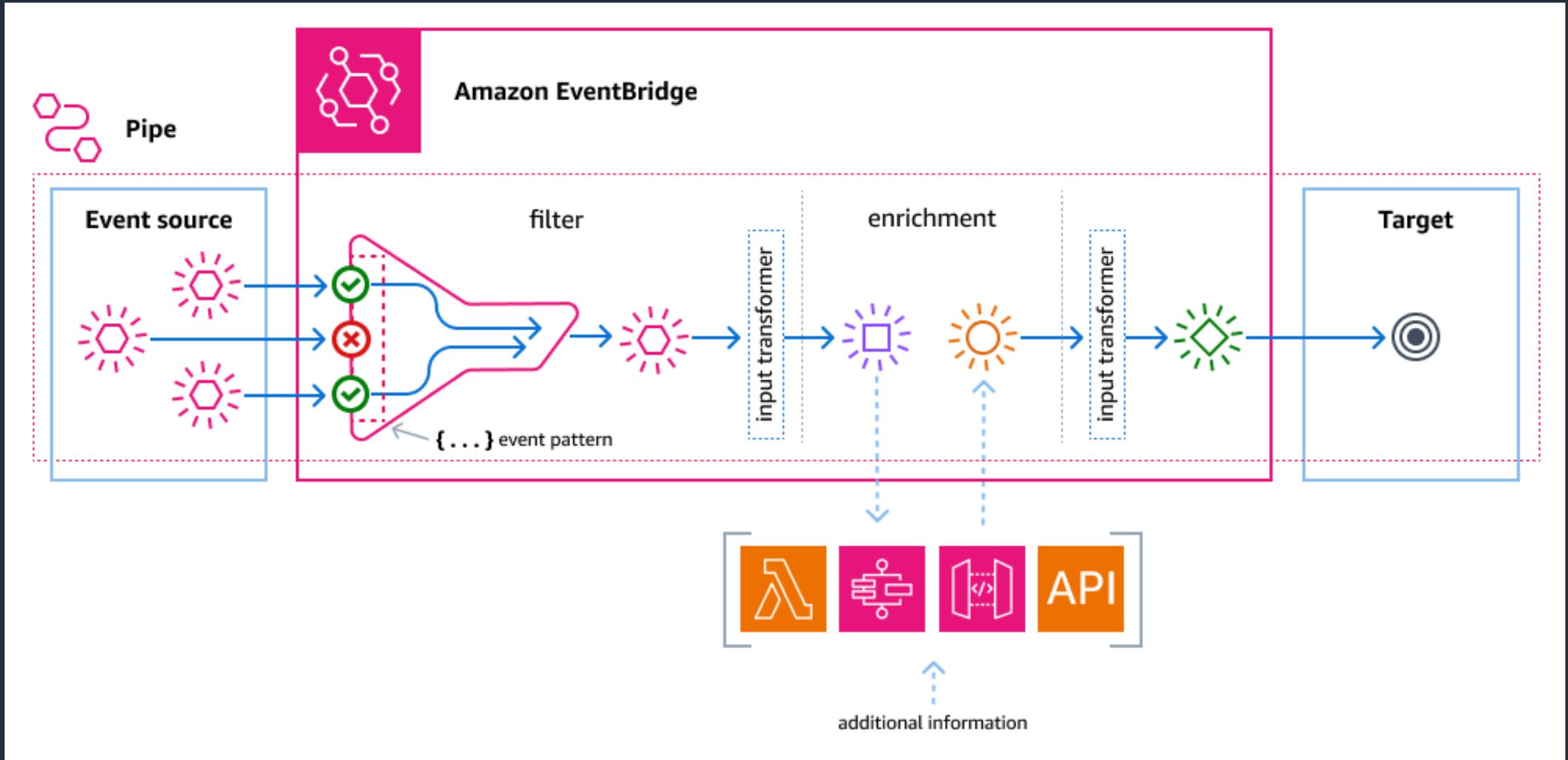
# イベントバスを使用した場合



# イベントバスを使用した場合



# EventBridge Pipes とは



# イベントバスとパイプの違い

イベントバス



Many publishers  
to many consumers

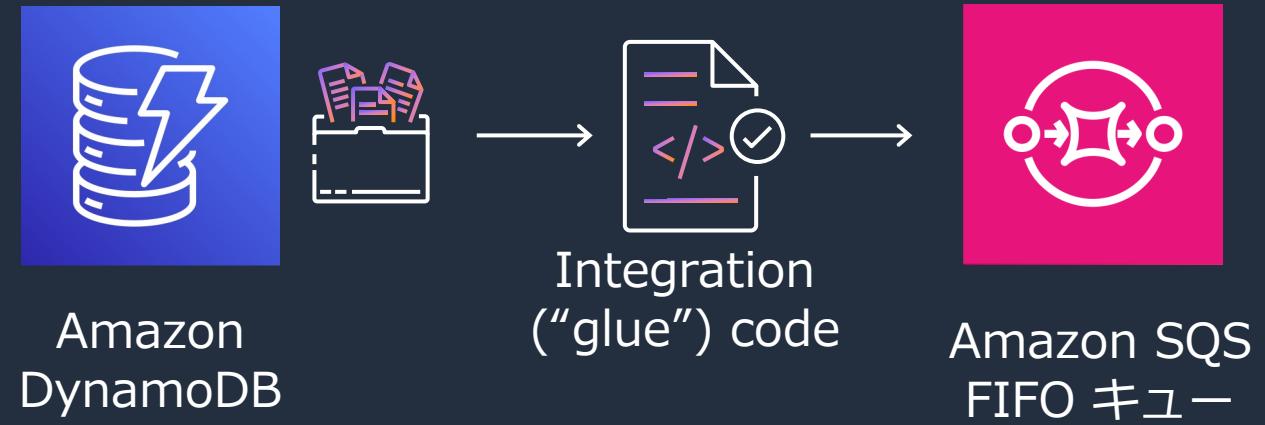
パイプ



Single publisher  
to single consumer

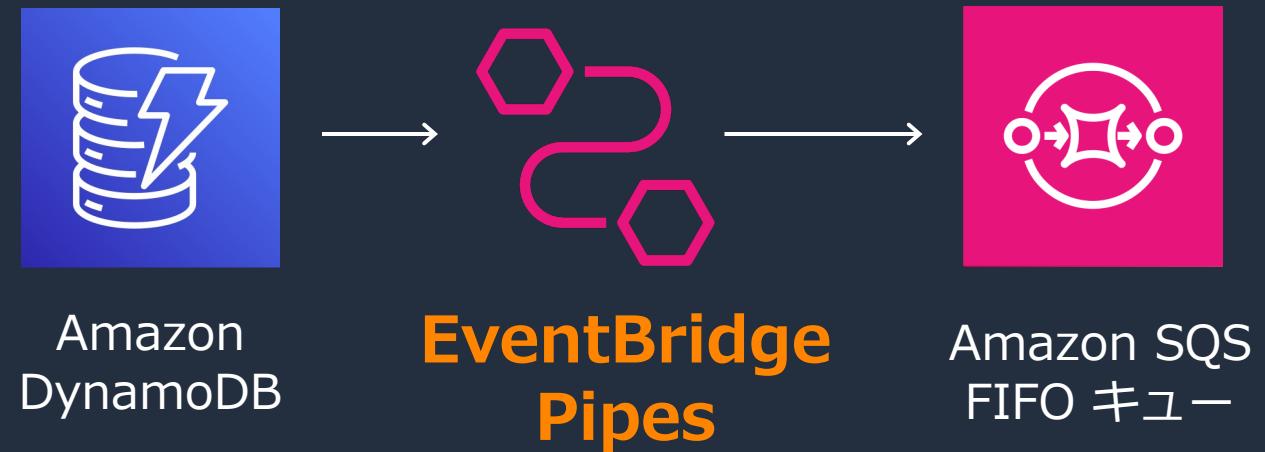
# サービス間連携の課題

- ・ サービスごとに異なる仕様や実装
- ・ エラーハンドリング
- ・ 配信順序の維持  
(\*サポートしているサービスに限る)
- ・ 認証、認可、流量制御
- ・ パフォーマンステスト
- ・ デプロイ
- ・ 運用監視



# サービス間連携の課題

- ・ サービスごとに異なる仕様や実装
- ・ エラーハンドリング
- ・ 配信順序の維持  
(\*サポートしているサービスに限る)
- ・ 認証、認可、流量制御
- ・ パフォーマンステスト
- ・ デプロイ
- ・ 運用監視



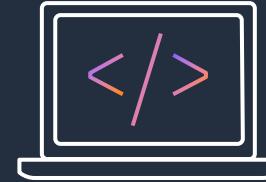
# EventBridge Pipes を利用することで TCO を削減



開発速度の  
低下



バグ発生の  
リスク



コードの  
複雑化



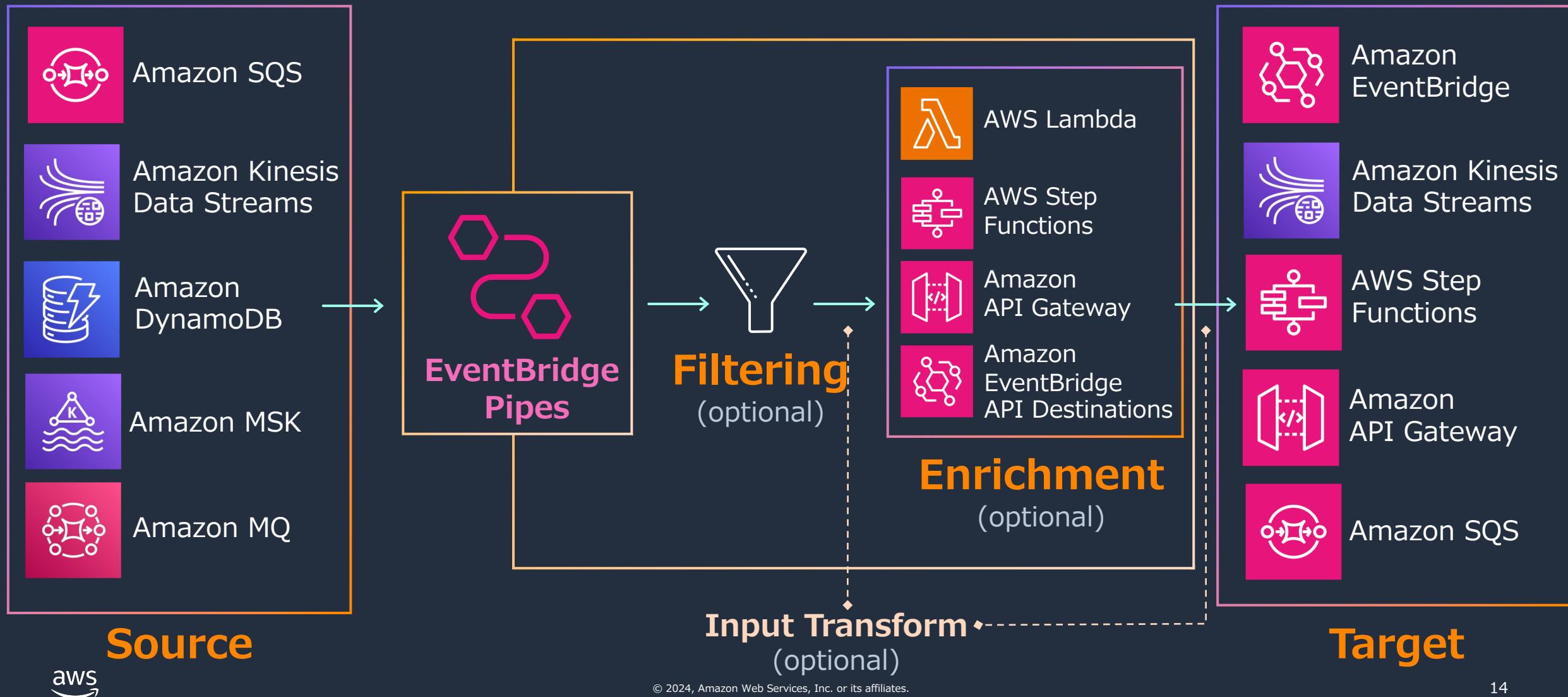
運用負荷の  
増加



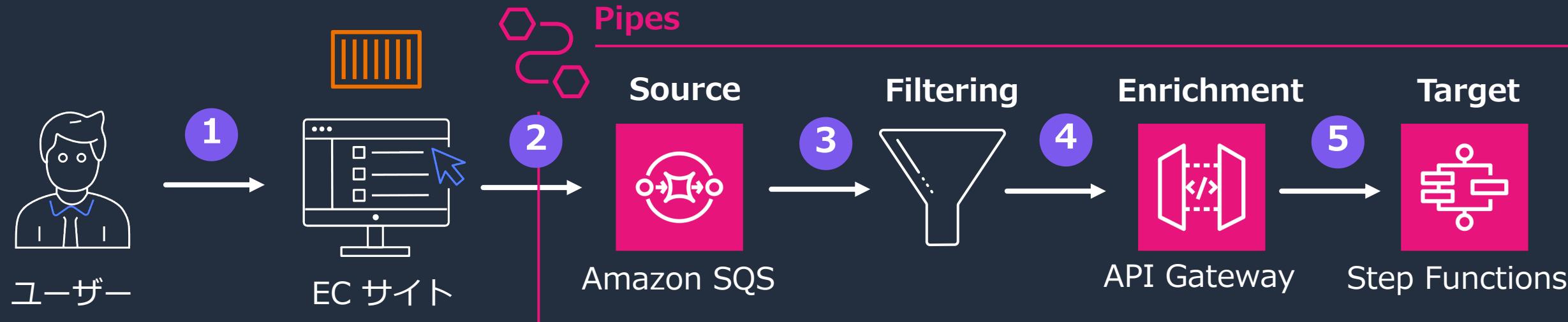
非効率な  
ポーリング

**High total cost of ownership**

# EventBridge Pipes を構成する要素



# EventBridge Pipes の利用イメージ



1. ユーザーが Web から注文を行う
2. バックエンドのアプリケーションが注文内容を保存し、注文 ID と合計金額を SQS に送信
3. 一定の金額以上の注文にクーポンを付与するために、合計金額でフィルタリングする
4. ユーザーサービスを呼び出してユーザーの詳細情報を追加で取得
5. クーポンサービスのワークフローをトリガーしてクーポンの付与処理を開始する

# EventBridge Pipes の主な機能

## イベントのフィルタリング

パターンマッチングによるフィルターを追加して、ターゲットに渡す必要のないイベントを除外できる。除外されたイベントは課金対象外。フィルターでは JSON パスが利用可能。

## バッチ処理

イベントソースから受信したイベントを、ターゲット/エンリッチメントに対してバッチで送信できる。バッチの利用可否や最大バッチサイズは後続のサービス仕様に依存する。

## 順序保証

イベントソースから受信したイベントの順序を維持したまま同期でターゲットを呼び出す。順次保証がないソースからのイベントの場合は、非同期でターゲットを呼び出す。



## 同時実行による高いスループット

バックログに応じて自動でスケールアップ/ダウンを行う。サービスごとに最大同時実行数は異なる (例: Amazon SQS – 1,250)。

## イベントデータの拡張

AWS Lambda, AWS Step Functions, API Gateway, API Destinations を呼び出して追加のデータを取得できる。各サービスは同期的に呼び出し、最大レスポンスサイズは 6 MB に制限される。

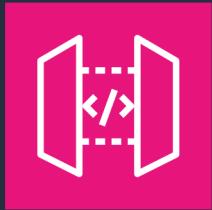
# イベントソースに指定できるサービス

	Amazon SQS キュー		Amazon MSK トピック
	Amazon Kinesis Data Streams		Amazon MQ メッセージブローカー
	Amazon DynamoDB Streams		セルフマネージド Apache Kafka ストリーム

# ターゲットに指定できるサービス

	Amazon EventBridge • イベントバス • API Destinations		AWS Batch ジョブキュー		Amazon Redshift Data API
	Amazon Simple Notification Service (SNS) * 標準トピックのみ		Amazon Elastic Container Service (ECS) タスク		Amazon SageMaker Pipelines
	Amazon Simple Queue Service (SQS)		AWS Lambda 関数 (同期 or 非同期)		Amazon CloudWatch ロググループ
	AWS Step Functions • 標準: 非同期 • Express: 同期 or 非同期		Amazon Kinesis Data Streams		Amazon Inspector 評価テンプレート
	Amazon API Gateway		Amazon Kinesis Data Firehose		

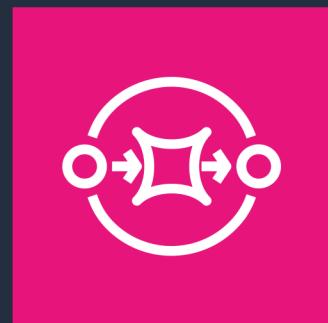
# エンリッチメントに指定できるサービス

	Amazon EventBridge API Destinations
	AWS Step Functions * Express ワークフローのみ
	Amazon API Gateway
	AWS Lambda 関数

- 同期呼び出しのみをサポート
- レスポンスのサイズは最大 6 MB

# ユースケース 1: SQS → Step Functions

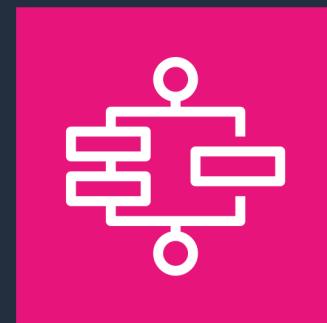
キュー内のメッセージに対して、複雑な一連の処理（ワークフロー）を実行



Amazon SQS



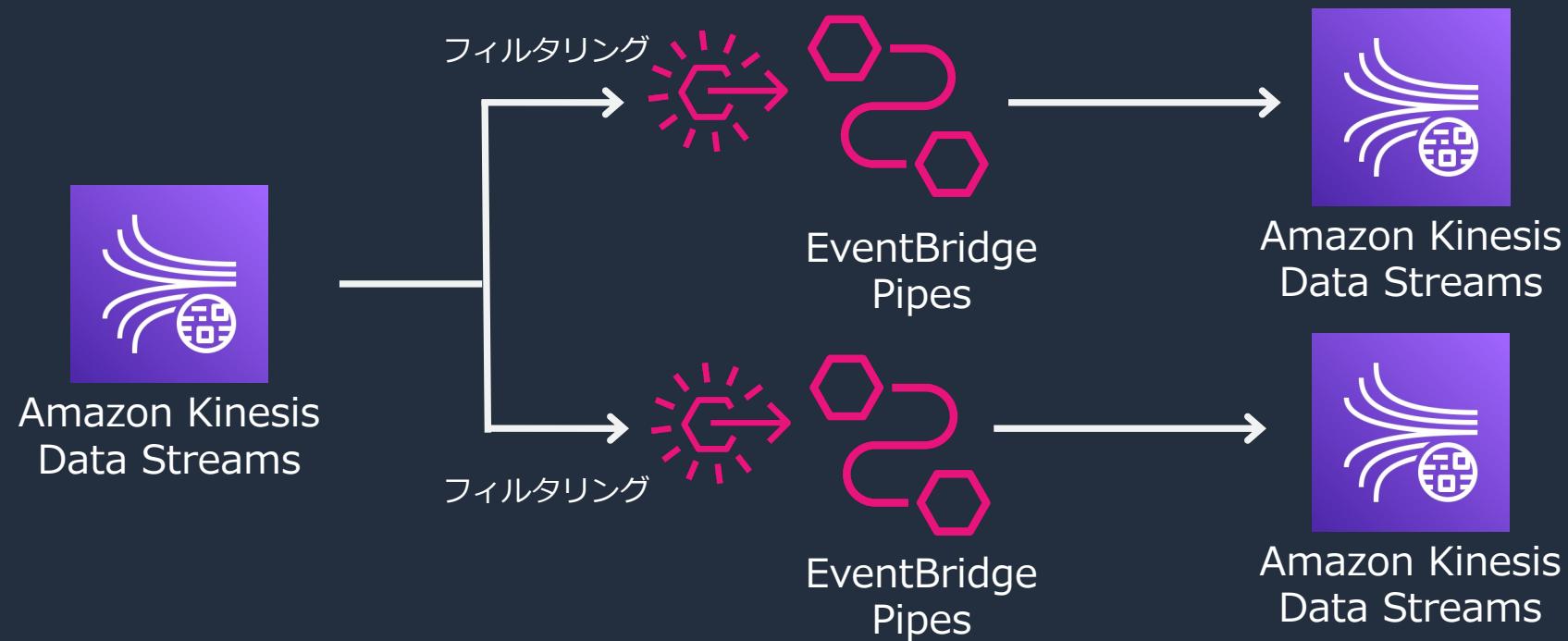
EventBridge Pipes



AWS Step Functions

# ユースケース 2: Kinesis Data Streams の分割

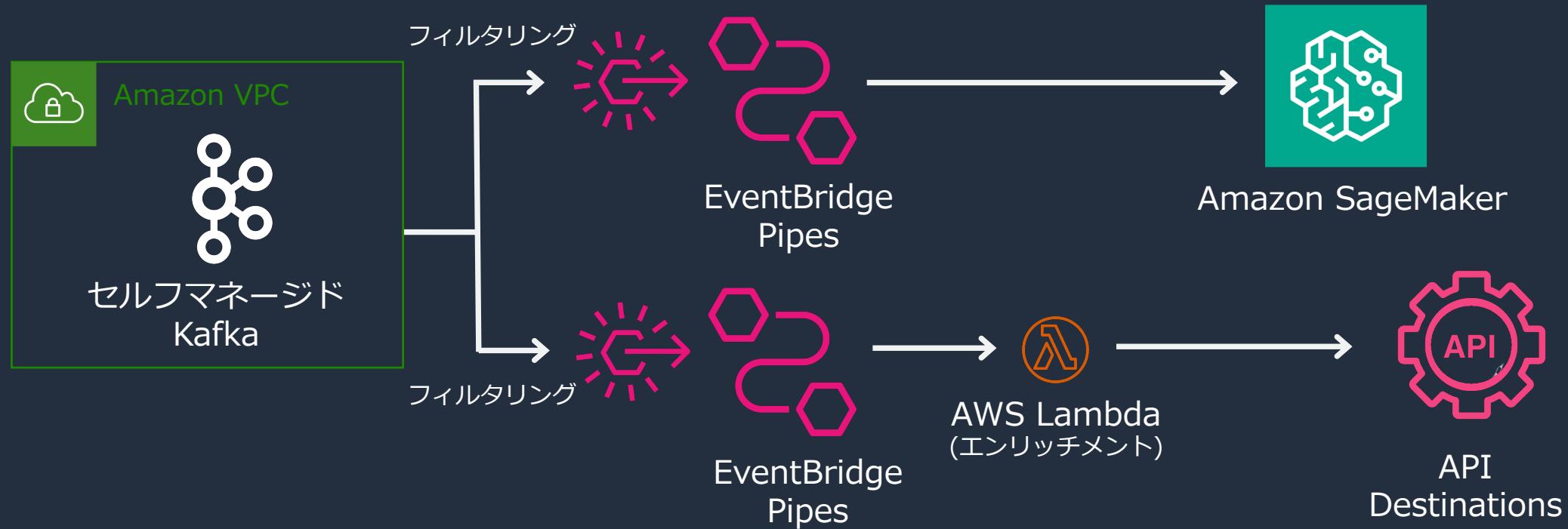
多重化されたデータストリームをドメインに応じて複数のデータストリームに分割



# ユースケース 3: Kafka → EventBridge API Destinations

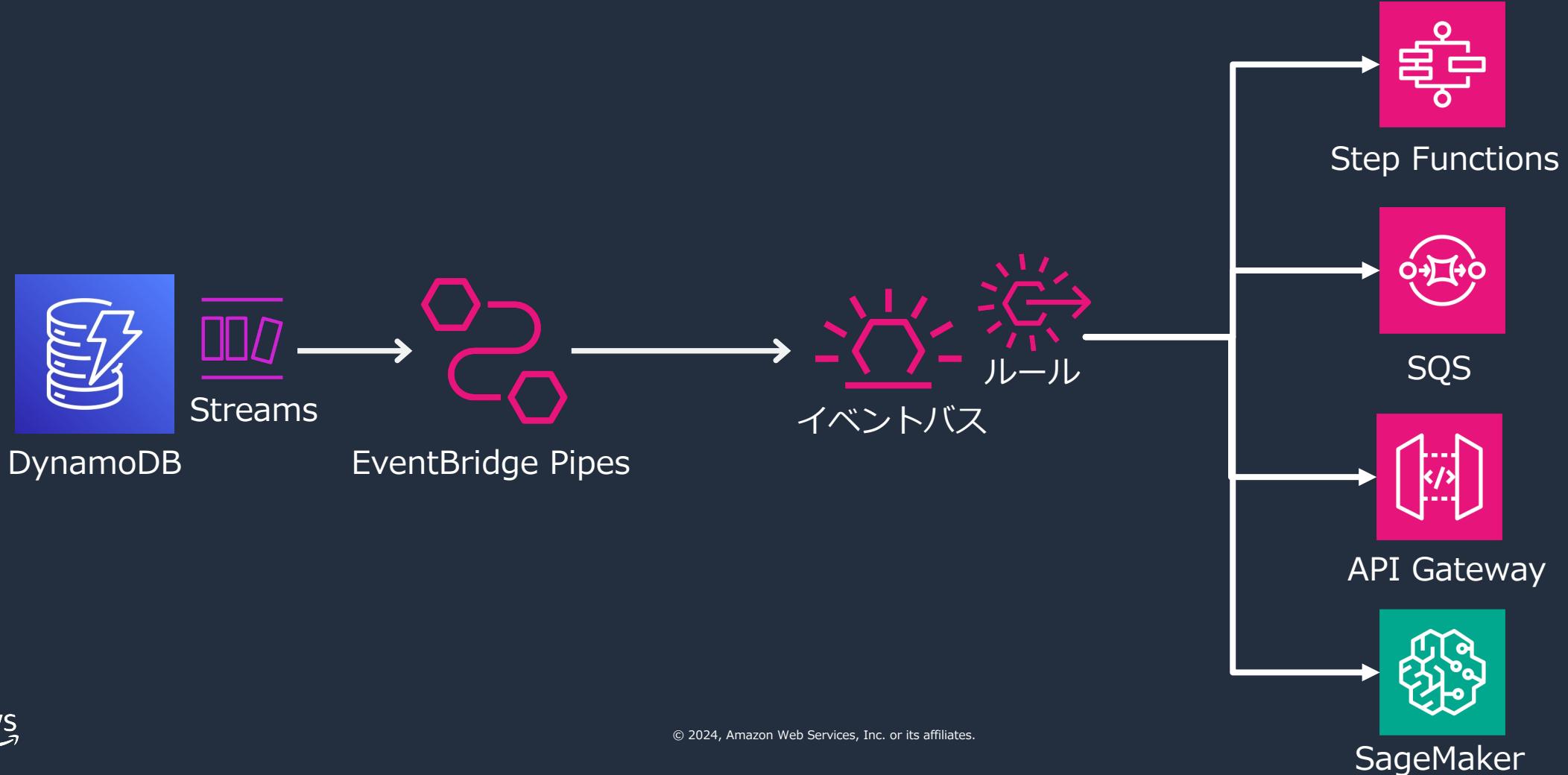
セルフマネージド Kafka クラスターと AWS サービスを接続

コードを書かずに HTTP(s) API を実行



# ユースケース 4: DynamoDB → EventBridge

DynamoDB レコードの変更をノーコードで他 AWS サービスに連携



# ソースの設定

The diagram illustrates the flow of an EventBridge pipe. It starts with a red 'Source' box containing an icon of two circles connected by a double-headed arrow, labeled '必須' (Required) and 'SQS キュー' (SQS Queue). An arrow points to a grey 'Filtering' box with a funnel icon, labeled 'オプション' (Optional) and '未構成' (Unconfigured). Another arrow points to a grey 'Enhancement' box with a circular icon containing a triangle, labeled 'オプション' (Optional) and '未構成' (Unconfigured). A final arrow points to a grey 'Target' box with a circular icon containing a green circle, labeled '必須' (Required) and '未構成' (Unconfigured).

ソース 情報 ⓘ 設定は有効です

**詳細**  
EventBridge パイプは、さまざまなソースからイベントを受信できます。

ソース  
パイプのソースを選択します。  
SQS

SQS キュー  
SQS キューの ARN を選択または入力します。  
sample-sqs.fifo

▼ 追加設定 - オプション

バッチサイズ - オプション  
1つのバッチで取得するメッセージの最大数。  
1

バッチウィンドウ - オプション  
関数を呼び出す前にレコードを収集する最大時間 (秒単位)。

# フィルタリングの設定



ソース  
必須  
SQS キュー → フィルタリング  
オプション  
追加済み → 強化  
オプション  
未構成 → ターゲット  
必須  
未構成

## フィルタリング - オプション

設定は有効です 情報

サンプルイベント - オプション

### イベントパターン

EventBridge Pipes のフィルタリングステップでは、エンリッチメントまたはターゲットに送信されるイベントをフィルタリングできます。

存在マッチング ▾ 握入 シンタックスを表示

存在マッチングは、イベントの JSON 内のフィールドの有無にかかわらず機能します。存在マッチングはリーフノードでのみ機能します。中間ノードでは機能しません。次のイベントのイベントに一致します。 詳細はこちら

```
"detail": {  
    "c-count": [ { "exists": false } ]  
}  
  
1 {  
2   "messageAttributes": {  
3     "orderId": [ { "exists": true } ]  
4   }  
5 }
```

参考: [Amazon EventBridge イベントパターンでのコンテンツのフィルタリング](#)

# エンリッチメントの設定

The diagram illustrates the flow of data through three stages:

- ソース 必須**: SQS キュー (Source - Required: SQS Queue)
- フィルタリング オプション**: 追加済み (Filtering - Option: Added)
- 強化 オプション**: Step Functions (Enrichment - Option: Step Functions)
- ターゲット 必須**: 未構成 (Target - Required: Unconfigured)

**強化 - オプション** 情報 ⓘ 設定は有効です

**詳細**  
EventBridge パイプのエンリッチメントステップでは、ソースからのデータをターゲットに送信する前に拡張することができます。

**サービス**  
AWS のサービスを選択します。  
**Step Functions**

**ステートマシン**  
**sample-express**

▶ 強化入力トランスマーチャント - オプション

# エンリッチメントの設定 – Input Transformer

## ▼ 強化入力トランスフォーマー - オプション

### ▶ サンプルイベント/イベントペイロード、トランスフォーマー、出力 サンプル

#### サンプルイベント/イベントペイロード 情報

サンプルイベント/ペイロードを選択するか、独自のイベント/ペイロードを入力します。

サンプルイベント 1

```
1 {
2   "messageId": "059f36b4-87a3-44ab-83d2-661975830a7d",
3   "receiptHandle": "AQEBwJnKyrHigUMZj6rYigCgxlS3SLy0a...",
4   "body": "Test message.",
5   "attributes": {
6     "ApproximateReceiveCount": "1",
7     "SentTimestamp": "1545082649183",
8     "SenderId": "AIDAENQZJ0L023YVJ4VO",
9     "ApproximateFirstReceiveTimestamp": "1545082649185"
10  },
11  "messageAttributes": {},
12  "md5OfBody": "e4e68fb7bd0e697a0ae8f1bb342846b3",
13  "eventSource": "aws:sqs",
14  "eventSourceARN": "arn:aws:sqs:us-east-2:123456789012:my-queue",
15  "awsRegion": "us-east-2"
16 }
```

JSON は有効です

{ } Prettify

□ ペイロード

#### トランスフォーマー 情報

トランスフォーマーを記述して、ターゲットに渡す情報を定義します。

```
1 {
2   "body": "This is body: <$.body>",
3   "attributes": "<$.attributes.SentTimestamp>",
4   "ingestionTime": <aws.pipes.event.ingestion-time>
5 }
```

トランスフォーマーが有効です

{ } Prettify

□ トランスフォーマー

#### 出力 情報

ターゲットにルーティングする前に、入力バスとテンプレートを使用して実行された、選択したイベントの変換の出力です。

```
1 {
2   "body": "This is body: Test message.",
3   "attributes": "1545082649183",
4   "ingestionTime": "2024-02-02T07:44:40.278Z"
5 }
```

□ 出力

参考: [Amazon EventBridge Pipes の入力変換](#)



# ターゲットの設定

The screenshot shows the AWS Step Functions Pipeline builder interface. At the top, there is a visual representation of a pipeline flow consisting of four boxes connected by arrows:

- Source (必須): SQS キュー
- Filtering (オプション): 追加済み
- Enrichment (オプション): Step Functions
- Target (必須): Firehose

Below this visual, the pipeline configuration details are shown:

### ターゲット 情報

ターゲット サービス: Firehose ストリーム  
ストリーム: pipes-demo

▶ ターゲット入力トランスマーチャント - オプション

# バッチの設定について (1/2)

- ・ソースからの取得およびターゲットの呼び出しでバッチをサポート
- ・複数イベントが JSON レコードの配列として渡される
- ・バッチの利用可否や最大バッチサイズは利用する各サービスに依存
- ・エンリッチメントについては、Lambda および Step Functions でサポート
- ・エンリッチメントおよびターゲットの呼び出しほは、各サービスのバッチ API にマッピングされる (※Lambda や Step Functions の場合は 1 リクエストの入力に JSON 配列全体が渡される)
- ・エンリッチメントはレスポンスとして JSON の配列を返すようにする

# ターゲットごとの最大バッチサイズ

CloudWatch Logs	10,000
EventBridge イベントバス	10
Kinesis Data Firehose ストリーム	500
Kinesis ストリーム	500
Lambda 関数	任意 *
Step Functions ステートマシン	任意 *
Amazon SNS トピック	10
Amazon SQS キュー	10

Lambda および Step Functions については、バッチに含まれるペイロードの合計サイズが制限を超過しないか注意する必要がある

- Lambda: 6 MB
- Step Functions: 262144 Bytes

## バッチの設定について (2/2)

- ターゲットがサポートするより大きいバッチサイズを指定することはできない
  - 例 : Kinesis Data Streams をソースに、SQS をターゲットにする場合  
Kinesis は最大 **10,000** レコードのバッチが設定可能  
SQS は最大 **10** メッセージ  
この場合、パイプで設定できる最大バッチサイズは **10**
- バッチ内の一箇所のレコードの処理に失敗した場合の挙動
  - SQS や Kinesis および DynamoDB のようなストリームがソースの場合、失敗したレコードに対して自動的にリトライを行う
  - エンリッチメントでの失敗の場合、部分的なリトライは行われない
  - ターゲットが Lambda または Step Functions の場合、**batchItemFailures** にリトライを行いうイベントの ID を含めてレスポンスを返すことで、部分的なリトライの実行が可能

# スループットと同時実行について

- STARTED 状態のパイプは、ソースからのイベントを継続的にポーリングし、利用可能なバックログとバッチの設定に応じて自動的にスケーリングを行う
- 単一のパイプは、デフォルトで以下の最大同時実行数までスケールする
  - DynamoDB: ストリームのシャード数 × 設定した *ParallelizationFactor* の数
  - Kinesis: ストリームのシャード数 × 設定した *ParallelizationFactor* の数
  - Apache Kafka: トピックのパーティション数 (最大 1,000)
  - Amazon MQ: 5
  - Amazon SQS: 1250
- これらの制限はベストエフォート型
- パイプの実行時間は最大 5 分 まで (※エンリッチメントおよびターゲットを含む)

# 実行ロールの設定

- パイプごとに実行 IAM ロールを指定

*Principal: { "Service": "pipes.amazonaws.com" }*

- ソースへのアクセス許可を付与  
(例: DynamoDB の場合)

- dynamodb:DescribeStream
  - dynamodb:GetRecords
  - dynamodb:GetShardIterator
  - dynamodb>ListStreams

- 必要に応じてエンリッチメントおよびターゲットの呼び出し許可を付与

# ログの設定 (1/2)

- ・ イベントバッチが処理される際の各実行ステップでログレコードを生成
- ・ ログレベルを選択可能
  - OFF: ログを記録しない (デフォルト)
  - ERROR: パイプ実行中に発生したエラーに関連するレコードを記録
  - INFO: エラー以外の他の実行されたステップも記録
  - TRACE: すべてのステップのすべてのレコードを記録
- ・ ログレベルと記録される実行ステップの対応表は[ドキュメント](#)を参照

# 実行ステップ



Source

- Enrichment Transformation Started
- Enrichment Transformation Succeeded
- Enrichment Transformation Failed

Filtering

- Enrichment Stage Entered
- Enrichment Stage Succeeded
- Enrichment Stage Failed

- Enrichment Invocation Started
- Enrichment Invocation Succeeded
- Enrichment Invocation Failed
- Enrichment Invocation Skipped

Input Transformer



Enrichment

- Target Invocation Started
- Target Invocation Succeeded
- Target Invocation Failed
- Target Invocation Partially Failed
- Target Invocation Skipped

Input Transformer



Target

- Target Transformation Started
- Target Transformation Succeeded
- Target Transformation Failed

- Target Stage Entered
- Target Stage Succeeded
- Target Stage Failed
- Target Stage Partially Failed
- Target Stage Skipped



詳細は[ドキュメント](#)を参照

## ログの設定 (2/2)

- ・ ログの配信先として、Amazon CloudWatch, Amazon Kinesis Data Firehose, Amazon S3 をサポート (複数選択可)
- ・ CloudWatch が選択されている場合、ログレコードの最大サイズは 256 KB に制限され、フィールドごとに自動的に切り捨てられる
- ・ 実行データをログに含めるか選択可能
  - 実行データには、イベントバッチペイロード、エンリッチメントおよびターゲットのリクエスト/レスポンスが含まれる
  - 機密データがログに含まれる可能性がある点に注意

# 料金

- ・パイプが処理するリクエスト数に応じて課金
- ・100 万リクエストごとに \$0.50 (\*2024 年 3 月時点/東京リージョン)
- ・フィルタリングで除外されたイベントは課金対象外
- ・ペイロードサイズ 64 KB を上限として 1 リクエストとして扱う

例：バッチで合計 256 KB のイベントを処理する場合

$$256 \text{ KB} / 64 \text{ KB} = 4 \text{ リクエスト となる}$$

# クオータ

- アカウント全体で作成できるパイプの上限: 1,000
- アカウント全体でのパイプの最大同時実行数:
  - バージニア、オレゴン、アイルランドリージョン: 3,000
  - 東京/大阪を含むその他のリージョン: 1,000
- いざれも上限緩和申請が可能

# まとめ

- EventBridge Pipes は、イベントプロデューサーとイベントコンシューマーを直接 point-to-point で連携する機能
- フィルタリングやバッチ、順序保証、HTTP API の呼び出しによるペイロードの変換、リトライなど豊富な機能をサポート
- 連携のためのコード (glue code) やインフラストラクチャの管理のための時間を削減し、よりビジネス的な価値に集中することができる
- 既存のイベントバスは引き続き利用可能。Pipes が提供するリッチな機能が必要ない場合は無理に移行する必要はない



# Thank you!