What's new in Serverless

サービスカットシリーズ

Solutions Architect
Kensuke Shimokawa
2020/7/28

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

過去資料
https://amzn.to/JPArchive

Who am I?

Name

Kensuke Shimokawa

Company

Amazon Web Services Japan K.K.

Role

Serverless Specialist Solutions Architect

@_kensh

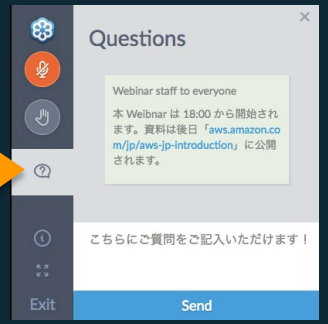
AWS Black Belt Online Seminar とは

「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7月28日現在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。

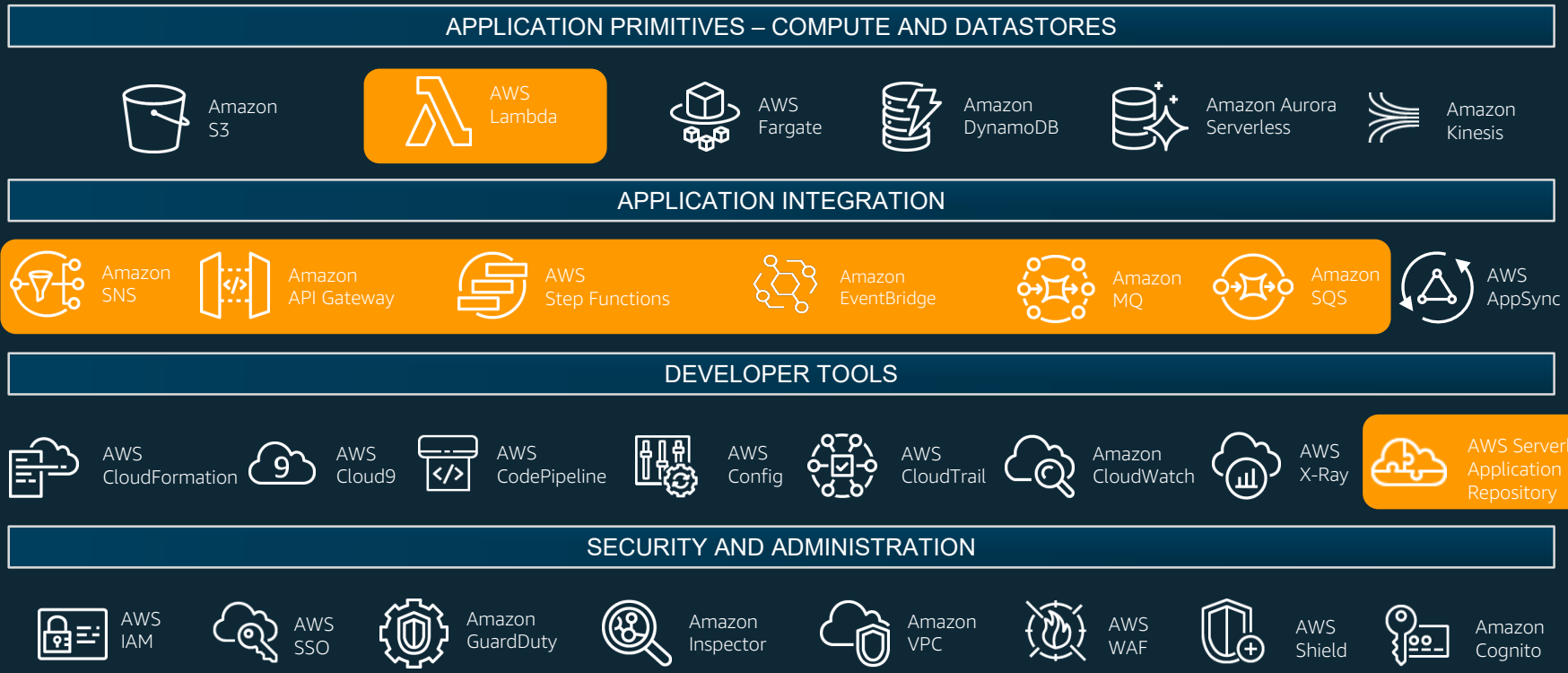
- 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます。

- 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.

今、まさにパラダイムシフトのとき!

75% *の組織は今後 2 年でサーバーレステクノロジーの利用を計画をしている*

AWS サーバーレス ポートフォリオ



How do we focus our efforts?

Quiz

2019年に

Lambda、API Gateway、Step Functions

に行った機能追加の数は？

36

＋

新規サービス **EventBridge**

1

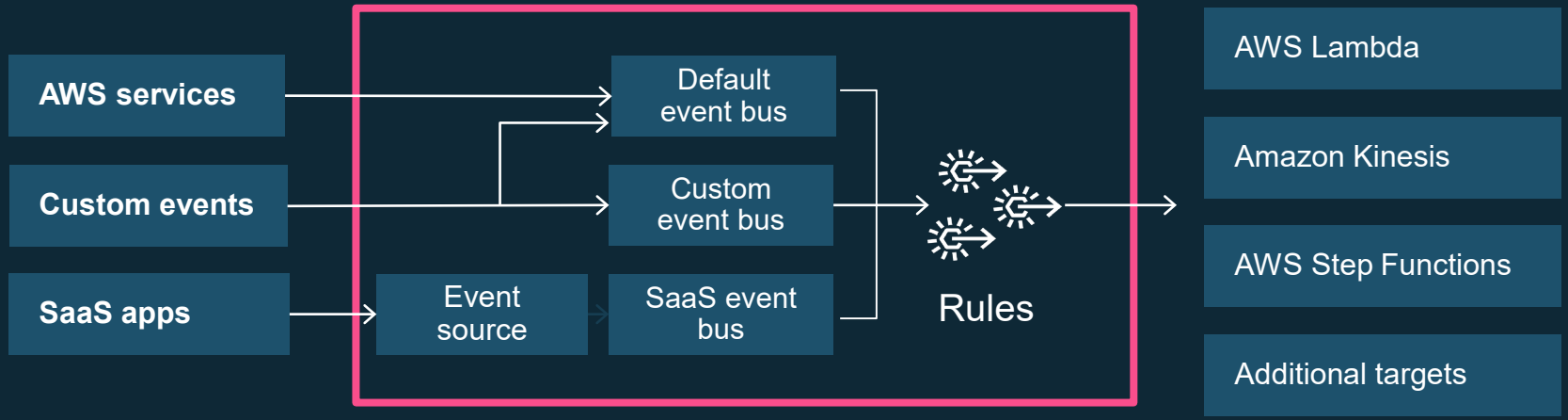
Amazon EventBridge

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
サーバーレスイベントバスサービス

- アプリケーション同士を接続するための
  のコードが不要
- AWSアカウントやSaaS間に渡って機能
- シンプルなプログラミングモデル
- イベントドリブンなアーキテクチャ
- フルマネージドで従量課金

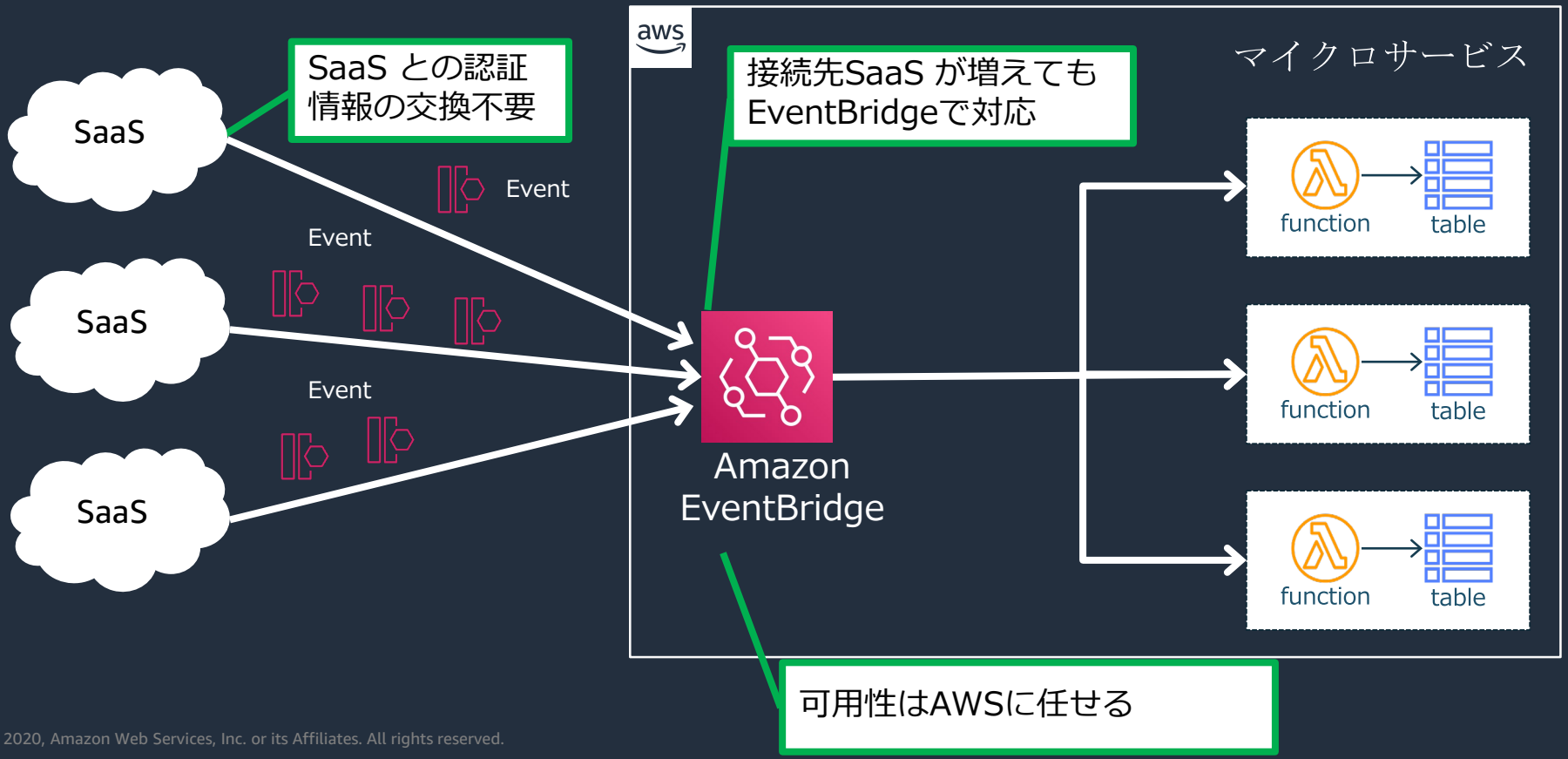
aws

Amazon EventBridge



- AWS サービスや、対応 SaaS からのイベントをタイムリーに受信し多様な処理連携を実現
- AWS サービスと統合することで、価値を生まない処理連携のプログラミングを排除
- サーバーレスに動作し、イベント受信のためのサーバーの準備・構成・管理が不要

SaaS と連携する際のポイント



Amazon EventBridge Schema Registry

- Schema Registry - EventBridge (CloudWatch Events) 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公開
  - e.g. EC2 インスタンスの状態変化通知のスキーマ

aws.ec2@EC2InstanceStateChangeNotification

Version 1 Created on Dec 1, 2019, 09:11 AM GMT+9 — Action

```
"components": {
  "schemas": {
    "AWSEvent": {
      "type": "object",
      "required": ["detail-type", "resources", "id", "source", "time", "detail", "region", "version", "account"],
      "x-amazon-events-detail-type": "EC2 Instance State-change Notification",
      "x-amazon-events-source": "aws.ec2",
      "properties": {
        "detail": {
          "$ref": "#/components/schemas/EC2InstanceStateChangeNotification"
        },
        "detail-type": {
          "type": "string"
        },
        "resources": {
```

プロパティの一覧とそれぞれの型 (String や独自の型など)

Version 1 Created on Dec 1, 2019, 09:11 AM GMT+9

```
    },
    "EC2InstanceStateChangeNotification": {
      "type": "object",
      "required": ["instance-id", "state"],
      "properties": {
        "instance-id": {
          "type": "string"
        },
        "state": {
          "type": "string"
        }
      }
    }
```

独自の型の定義も確認できる

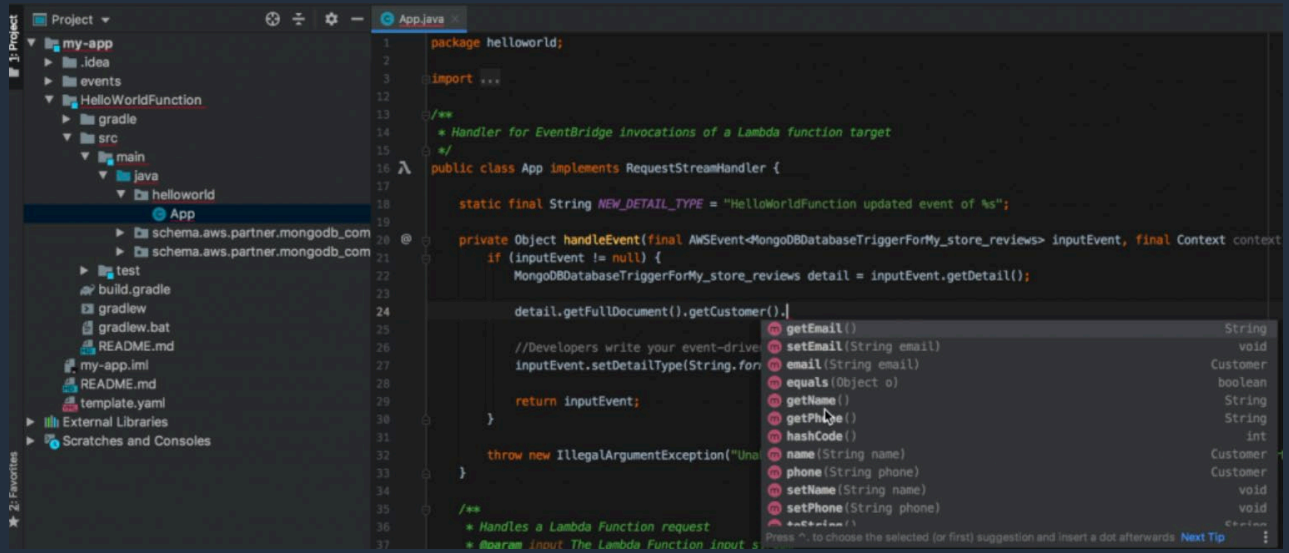
https://aws.amazon.com/jp/blogs/news/new-amazon-eventbridge-schema-registry-is-now-generally-available/

Amazon EventBridge Schema Discovery

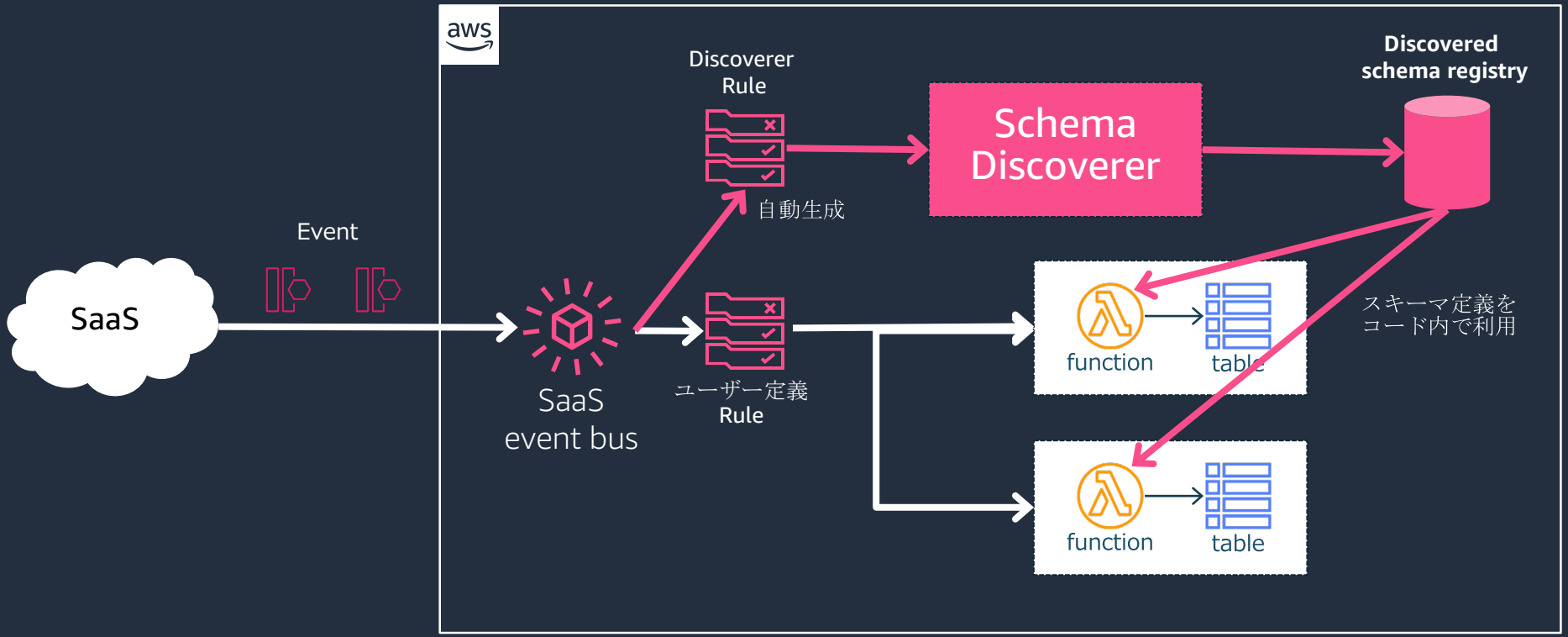
- Schema Discovery – SaaS から EventBridge へ送信されるデータについては、EventBridge へ流入するデータからスキーマを自動生成
  - e.g. MongoDB Atlas (MongoDB 社にて提供されるマネージドサービス) へデータの CRUD 処理が行われた場合のスキーマ

Schema Code Bindings

- Code Bindings – スキーマ定義を言語ごとにダウンロード可能
  - Java 8、Python 3.6、TypeScript 3 の 3 種類に対応
  - アプリケーション開発の際に、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



Schema Registry & Schema Discovery



Excelling in service fundamentals

HTTP APIs for Amazon API Gateway

Rest APIと比較して**70%のコスト削減**と**60%のレイテンシー低下**

アプリケーションの
コストを70%削減

アプリケーションの
レイテンシーを 60%
低下

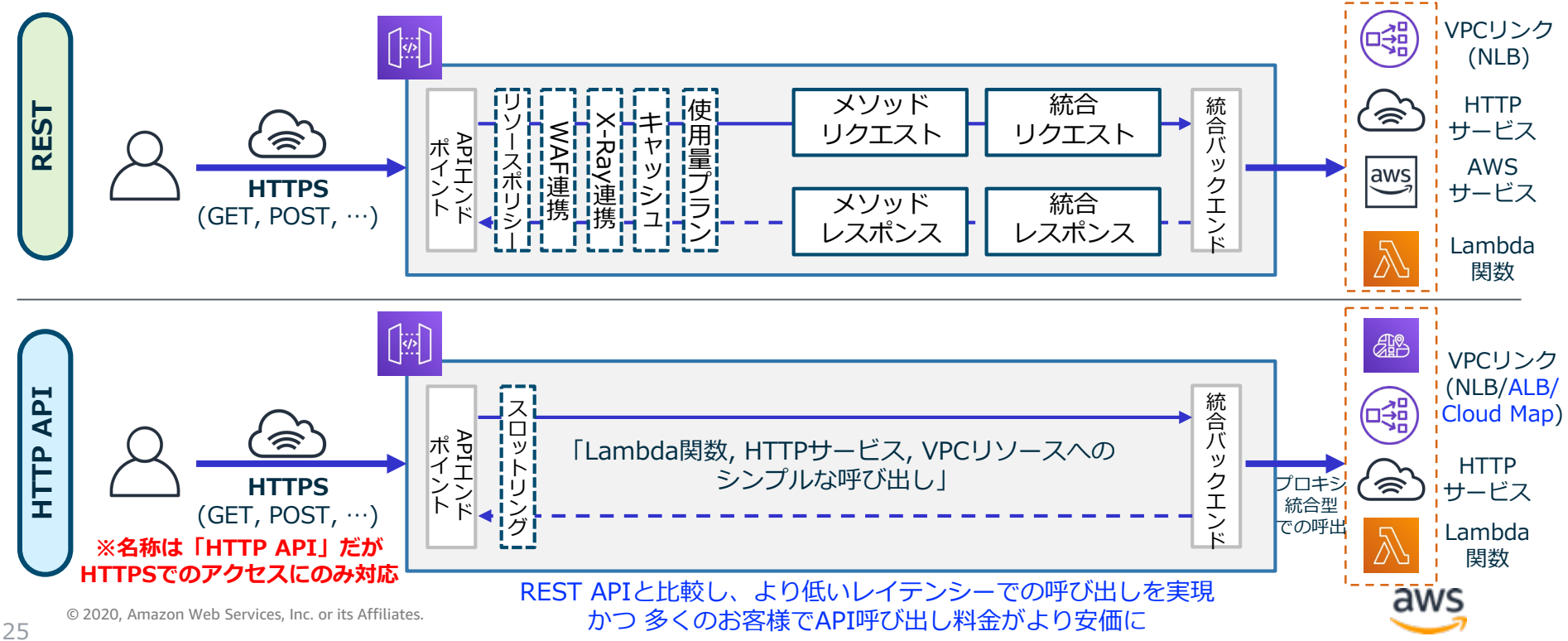
以前よりHTTP APIの
構成が容易かつ迅速

HTTP APIs for Amazon API Gateway

- **JWTオーサライザー** – ネイティブのOpenID Connect (OIDC) 認証をサポート。送信されたJWTトークンをパースし、リクエストをトークンのOAuthスコープで許可、禁止するようにAPI Gatewayを設定可能
- **デフォルトステージとルート** – APIの利用を容易に。デフォルトステージを選択するとベースURLでAPIを公開可能に。これまでのようにAPIのステージを明示的に指定する必要なし
- **VPC内リソースへのHTTP/Sアクセスを実現**
    - **ディスパッチ先として NLB・ALB・Cloud Mapサービス を指定可能**

HTTP API とは (RESTとの違い)

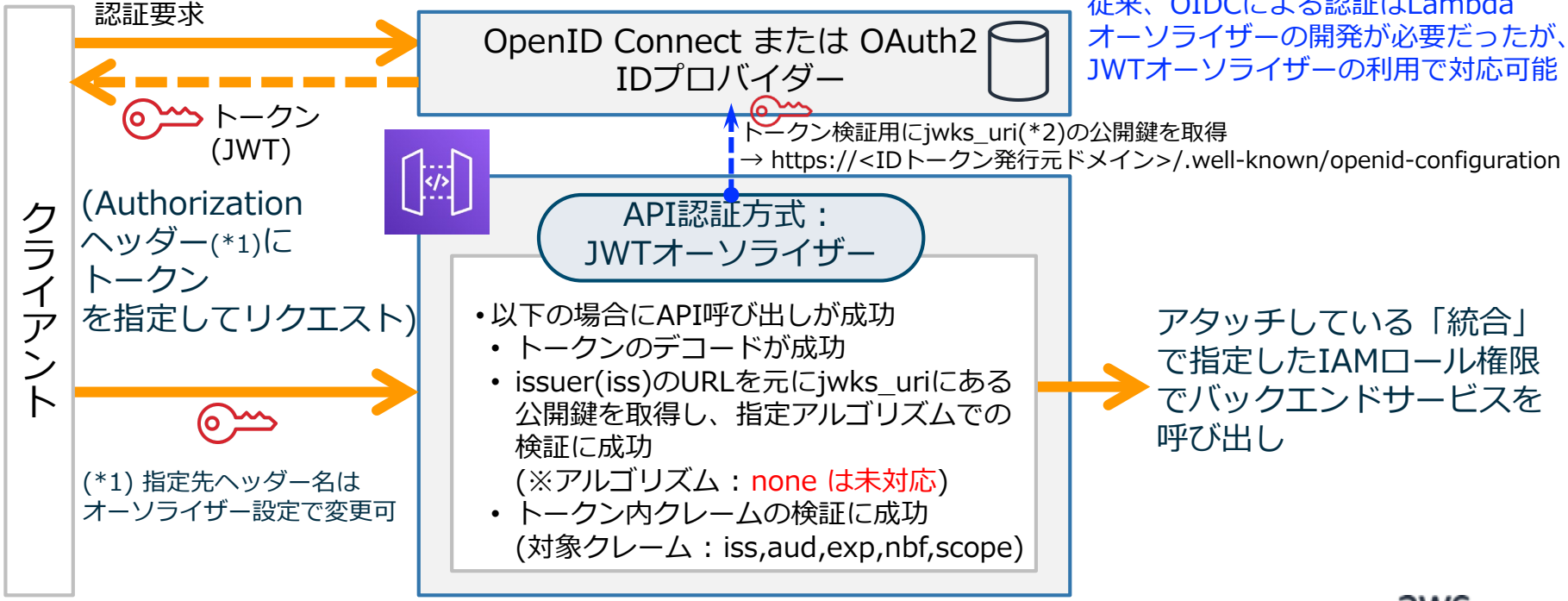
Lambda関数 や VPC内の HTTPリソース を
高速 かつ コスト効率よく API として公開する場合に最適なAPIプロトコル



認証 - JWTオーソライザー

HTTP API

事前にOIDC または OAuth2 IDプロバイダーで認証を行い、
取得したトークンをHTTPヘッダーに指定してAPIに渡すことを要求する方式

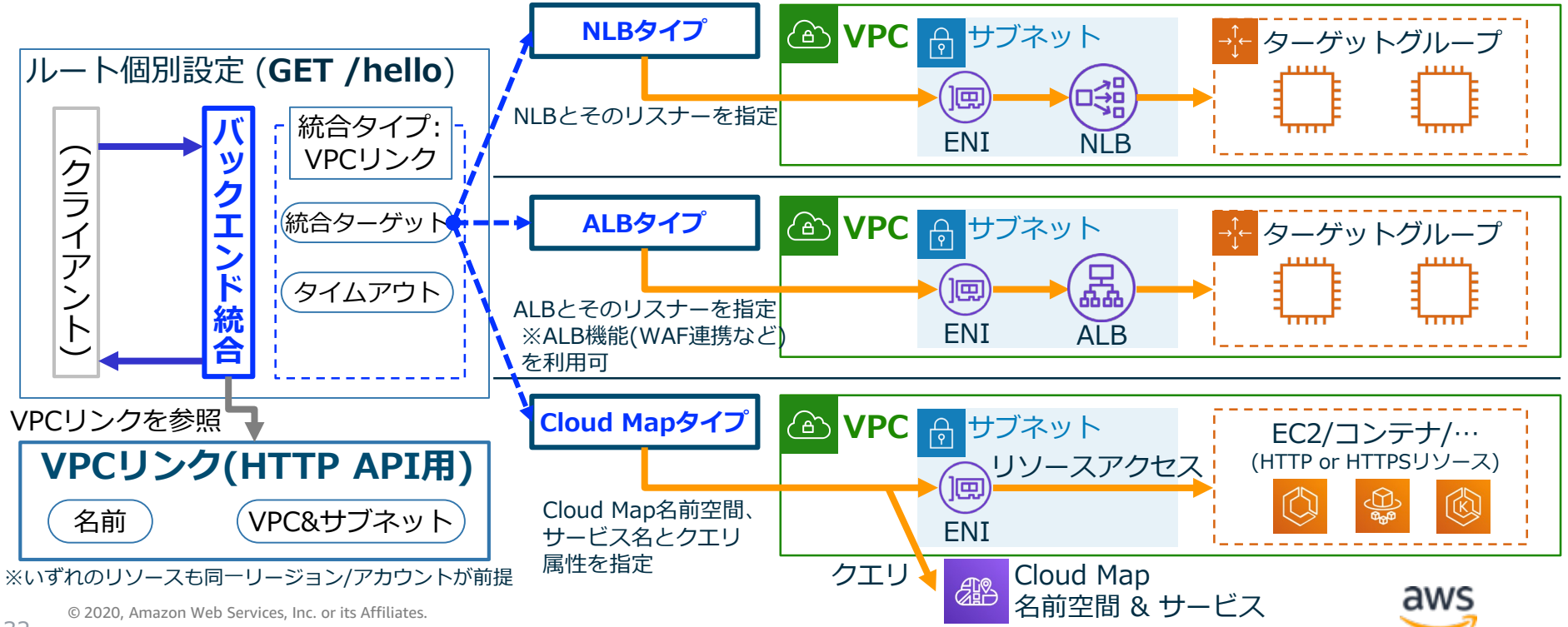


従来、OIDCによる認証はLambda
オーソライザーの開発が必要だったが、
JWTオーソライザーの利用で対応可能

HTTP API（ VPCリンク ）

HTTP API

VPC内リソースへのHTTP/Sアクセスを実現 (指定サブネットにENIが生成)
ディスパッチ先として NLB・ALB・Cloud Mapサービス を指定可能



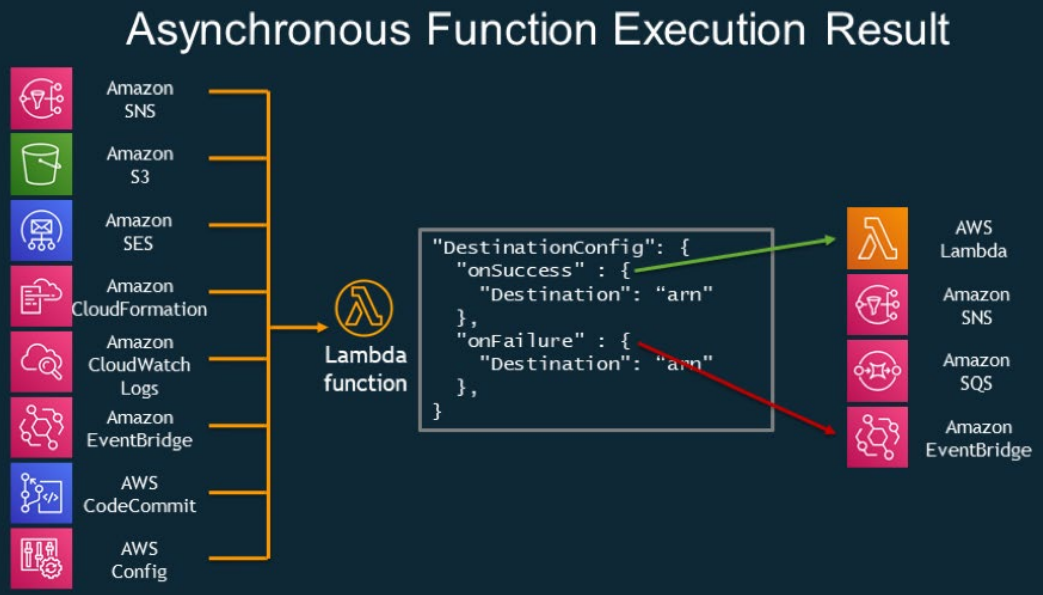
(*) ENI:Elastic Network Interface , NLB:Network Load Balancer , ALB:Application Load Balancer

HTTP API はどのような場合に利用？

- コストが課題になっている場合は大幅なコストダウンが可能
  - サーバーレスマイクロサービスのAPIのコストを削減
- 容易な設定 - API GatewayやSwaggerの経験は不要

AWS Lambda Destinations

非同期の**Lambda 関数の実行結果**の宛先を指定可能。**成功**と**失敗**のそれぞれについて、それぞれの宛先(Destinations)を指定可能



どのような場合に利用？ Lambda Destinations

- ベーシックな非同期オーケストレーションを構築可能
    - e.g. Amazon S3のバケットに対するオブジェクト登録に対して発火するような Lambda関数は非同期実行される。
      → 成功、失敗のハンドリングが可能に

Lambda非同期実行のエラーハンドリング改善

- 最大イベント経過時間 (Maximum age of event)
  - 同時実行数不足やシステムエラー時のリトライを行う時間。
  - 60 秒から 6 時間までのあいだで指定できる。
    - 60 秒を指定すると、エラー発生時に 1 秒、2 秒、4 秒、と指数関数的に間隔を増やしてリトライされ、データがキューに格納されてから 60 秒が経過するとイベントデータを削除、もしくは配信不能キュー (DLQ) へ移動するといった動作になる。
    - 一部のイベントソースからの実行要求が大量に発生し、他の Lambda 関数の実行を妨げてしまっている場合などに有効。

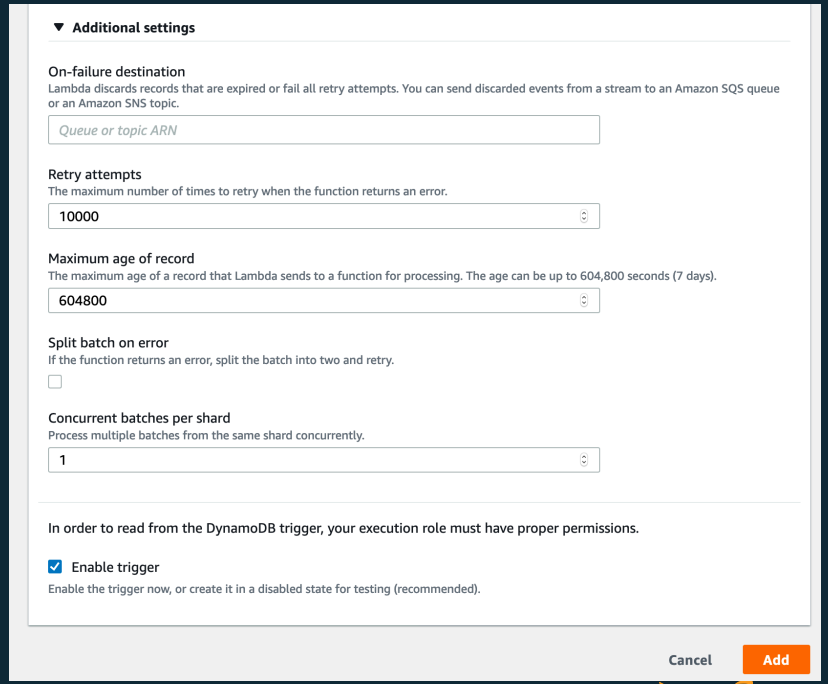
- 最大再試行回数 (Retry attempts)
  - Lambda 関数で実装した処理のなかでエラーが発生した際のリトライを行う回数。
  - 0 回から 2 回までのあいだで指定できる。
    - 0 回を指定すると、関数処理の中でエラーが発生してもリトライを行わず、イベントデータを削除、もしくは配信不能キュー (DLQ) へ移動するといった動作になる。
    - 実行の冪等性確保のため、エラー発生時は一切のリトライをしたくない場合などに有効。

aws

Streamイベントソースのエラーハンドリングをサポート

Kinesis/DynamoDB トリガーに実行失敗時のハンドリング機能を追加

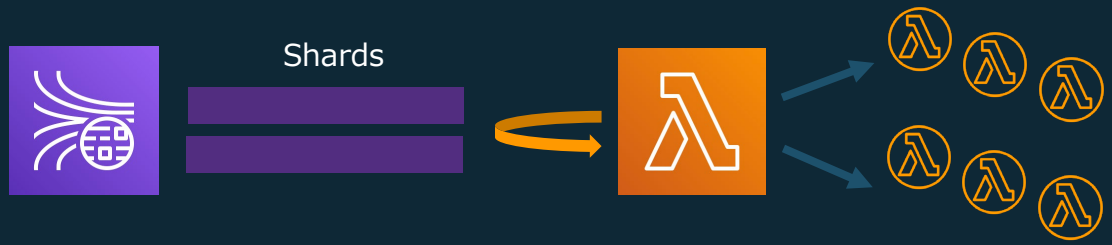
- **Bisect on Function Error**
  実行失敗時にバッチを分割し前方を再実行する。分割単位でさらに成功、失敗が判断され、失敗時は再帰的に分割。
- **Maximum Record Age**
  処理する最大の経過時間(60秒～7日)。経過したレコードはスキップされる。
- **Maximum Retry Attempts**
  実行失敗時のリトライ回数 (0～10000)。リトライがこの値に到達するとレコードはスキップされる。
- **Destination on Failure**
  実行失敗時に実行結果を AWS サービスに配信。SNS topic, SQS queue



AWS LambdaがKinesisとDynamoDBイベントソースの並列化係数をサポート

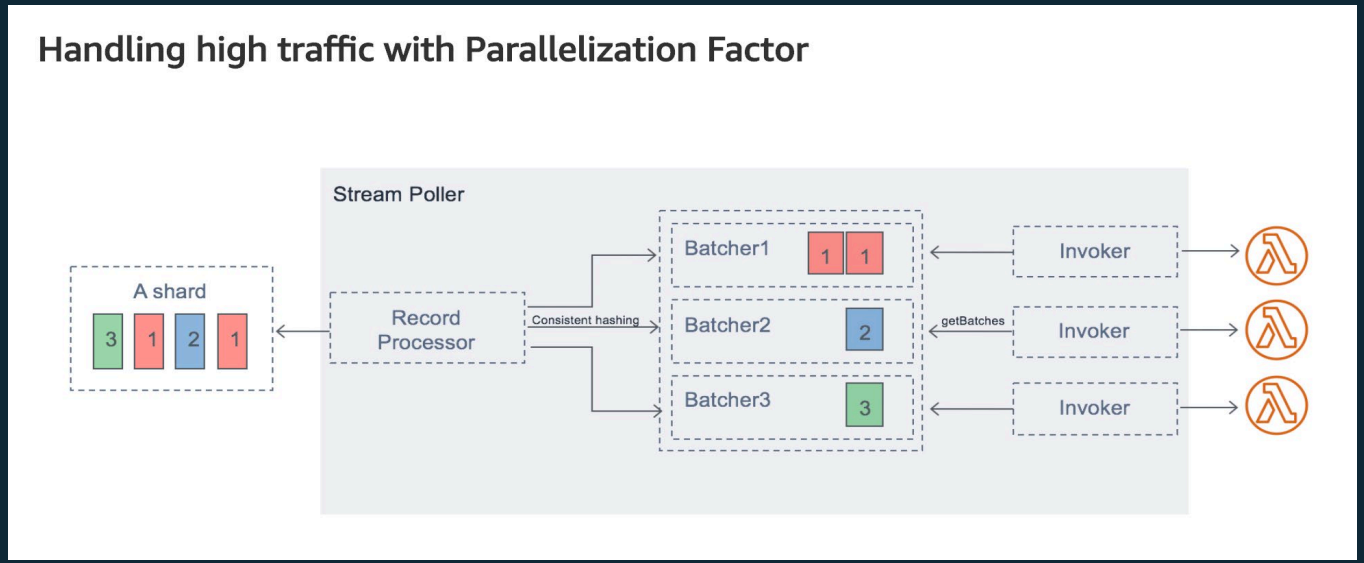
- 並列化係数（N=1-10）を指定できるようになった
  - これまでは シャード数：Lambda 同時実行数 = 1：1 で固定だったが、これを 1：**N** に指定することができる
  - 例えば並列化係数を3に設定すると、1シャードあたり3つのLambda関数が実行可能に

▼ 例: Kinesis Data Streams のシャード数 = 2、 Parallelization Factor = 3 の場合



Lambda 高度なスケーリングの制御

- バッチウインドウ: バッチサイズのレコードが集まるまでLambdaの実行を待機可能(最大300秒)
- シャードごとの同時実行バッチ: 同時に同一のシャードから複数のバッチを処理



どのような場合に利用？ 設定とスケールの制御

- パフォーマンスがクリティカルな要件の場合にストリームの処理を高速化可能
- 待機してより大きなバッチサイズを処理することで、処理速度は低下するがコストを削減可能

Amazon RDS Proxy

Amazon RDSのフルマネージドで高可用なデータベースプロキシ

- アプリケーションに、よりスケーラブルで、データベースのエラーに対してより弾力性のある、セキュアな**DBコネクションプール**

アプリケーションのスケーリングを改善するプールされた共有のDBコネクション

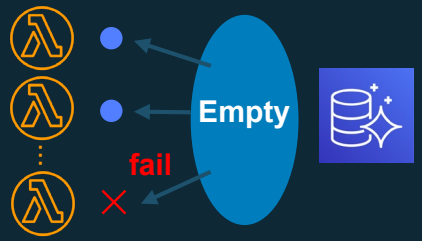
アプリケーションの可用性向上とDBのフェイルオーバー時間の削減

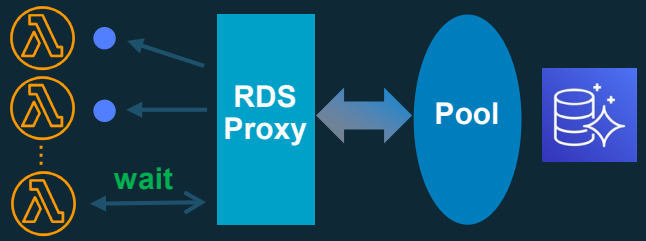
DBアクセス制御を伴うアプリケーションデータの管理

データベースと互換性のあるフルマネージドなDBプロキシ

Amazon RDS Proxy with AWS Lambda

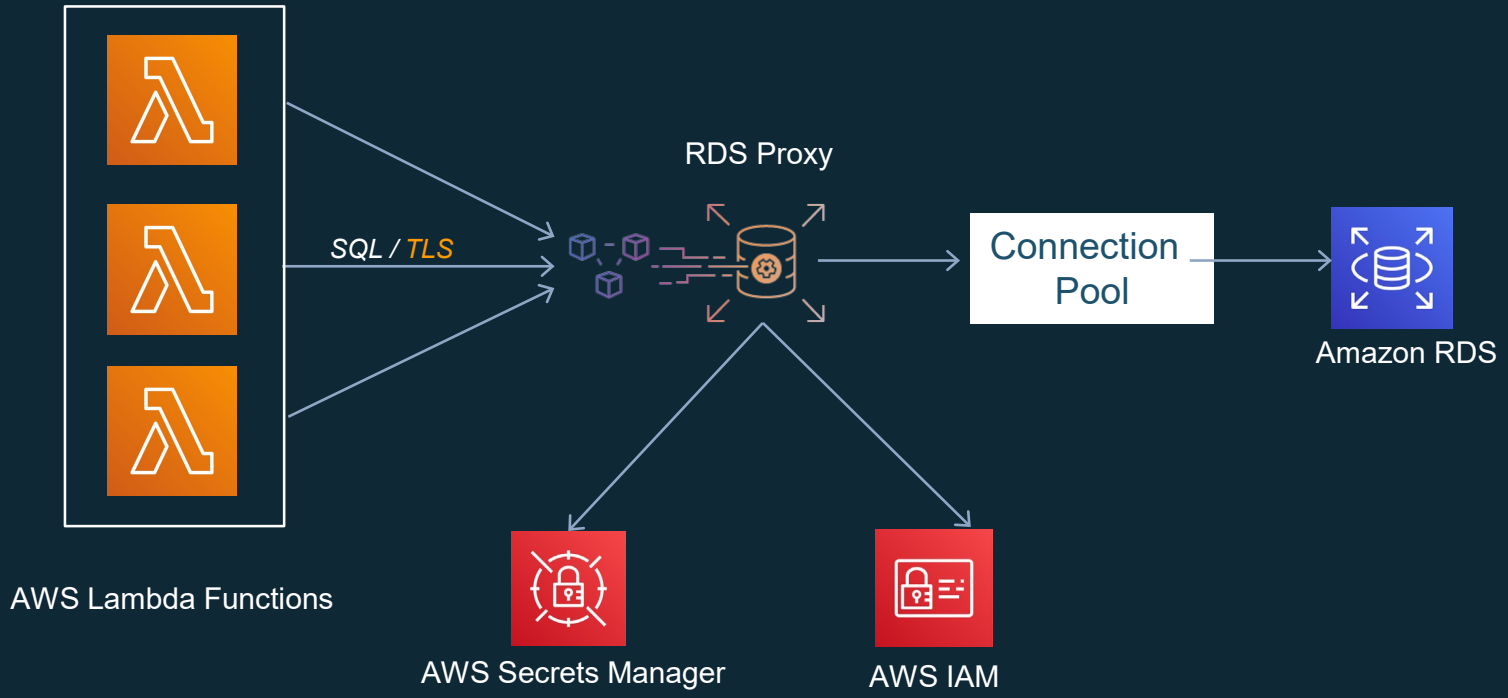
- RDS Proxy
  - コネクションプールの管理を行い、データベースへのコネクション要求が過剰に発生した場合にも、クライアントの接続を待機させたり、既存のコネクションを効率的に再利用させることができる
  - RDS (MySQL 5.6, 5.7)/Aurora MySQL(version1.x,2.x), RDS/Aurora(PostgreSQL 10.10以降,11.5以降)にて利用可能
- AWS Lambda と RDS Proxy の統合
  - RDS Proxy を利用することで、これまでの AWS Lambda と RDS との間で発生しやすかった、コネクションの過剰使用を抑止することができる





  - AWS Lambda のコンソールから RDS Proxy を作成でき、シームレスな統合を実現
  - RDS Proxy を配置する VPC と、AWS Lambda を接続する必要あり

よりセキュアな RDS Proxy



どのような場合に利用？ RDS Proxy

- コード内部でのコネクション管理の必要性を削減
- セキュリティの改善

新機能リリース

メトリックスとステートをより容易に生成、モニタできるようにする

- LambdaのCloudWatchメトリックスで**パーセンタイル**をサポート
- Step Functionsのワークフロー実行イベントをサポート
- API Gatewayが**Kinesis Data Firehose**へのアクセスログ記録をサポート
- CloudWatch **埋め込みメトリックスフォーマット**

CloudWatch 埋め込みメトリックスフォーマット

- 詳細ログイベントデータと一緒にカスタムメトリックスを埋め込み、CloudWatchは**カスタムメトリックスを自動的に展開**、視覚化、アラームの設定が可能でリアルタイムにインシデントを検出可能。
- Node.jsとPythonでオープンソースクライアントライブラリが利用可能

```python
from aws_embedded_metrics import metric_scope

@metric_scope
def my_handler(metrics):
    metrics.put_dimensions({"Foo": "Bar"})
    metrics.put_metric("ProcessingLatency", 100, "Milliseconds")
    metrics.set_property("AccountId", "123456789012")
    metrics.set_property("RequestId", "422b1569-16f6-4a03")
    metrics.set_property("DeviceId", "61270781-c6ac-46f1")

    return {"message": "Hello!"}
```

どのような場合に利用？ メトリックスとログ

- サーバーレスアプリケーション実行時のオペレータの負荷を軽減
- カスタムメトリックス生成時のパフォーマンス課題を軽減
- Splunkのようなサードパーティーツールとの統合を促進

New application development patterns

新機能リリース - お客様の声

”サーバーレスを高スループットアプリケーション
で利用したいが**スケーラビリティ**と**コスト**が心配”

新機能リリース

高スループットサービスをより簡単に構築できるようにする

- Lambdaの Provisioned Concurrency サポート
- Step Functions の Express Workflows のリリース

Provisioned Concurrency for AWS Lambda

Provisioned Concurrencyは2桁ミリセコンドでの応答を可能とする初期化済みの準備状態に関数を維持

- いつ、どれぐらいの期間Provisioned Concurrencyを有効にするかを制御可能

レイテンシーにシビアなアプリケーションに理想的

コードの変更は不要

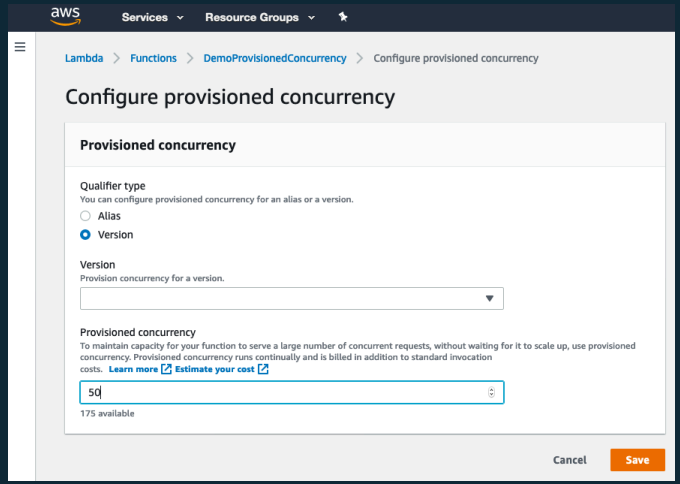
いつ有効化するかはコントロール可能

フルサーバーレス

Provisioned Concurrency for AWS Lambda

- **レイテンシーSLA**の制約があるアプリケーション
  - エンドユーザーと直接インタラクションがある
  - 厳密な規制の要件がある
  - コールドスタートに時間がかかる言語を利用している、または大きなデプロイパッケージ

- **高速なトラフィックバースト**をサポートするアプリケーション
  - ライブストリームの広告配信のようなコンテンツを提供
  - ゲームのようなモバイルアプリ
  - マーケティングキャンペーンまたはタイムセールス



Provisioned Concurrency for AWS Lambda

- Lambda 関数のコールドスタート問題に対応
  - 代表例として Java ランタイムを利用するケースなど、関数の初回起動時に大きく時間を要する場合、この機能を使うことで、**事前にウォームアップした状態から関数を実行開始**することが可能となる
- Lambda 関数のバーストに備えることが可能
  - AWS Lambda では、関数の実行が一度に大量に行われた (バーストした) 場合、仮に Lambda 関数の同時実行数を上限緩和していても、初期同時実行数のバースト制限 (東京リージョンの場合は 1000) までしか一度に実行できず、以降は 1 分ごとに 500 ずつ同時実行可能な数が増加する

  - この機能により、あらかじめ同時実行数をプロビジョニングしておくことが可能となるため、**関数の実行数のバーストにあらかじめ備えておく**ことが可能となる (※)

(※) Lambda 関数の同時実行数を上限緩和する申請はこれまでと変わらず必要

Step Functions Express Workflows

AWSのコンピュート、データベース、メッセージングサービスを100,000イベント/秒のレートまでオーケストレーションする

- IoTデータインテグレーション、マイクロサービス オーケストレーション
- ストリーミングデータ処理、変換処理のようなハイボリュームなイベント処理ワークロードに最適なサービス

より速く：100K/秒以上の状態遷移

5分以内の短期間のワークフロー向けに設計されている

スケールで

コスト効果的

AWS Step Functions Express Workflows

	Standard	Express
ステートマシンの最大実行時間	1 年	5 分
ステートマシンの開始レート (上限緩和可能)	Bucket Size : 800 Refill Rate per Second : 150	Bucket Size : 6000 Refill Rate per Second : 6000
ステート変化レート (上限緩和可能)	Bucket Size : 800 Refill Rate per Second : 500	ほぼ無制限
料金	ステートが変化した回数に応じた課金 ・4000 回まで無料 ・以降は 1000 回あたり 0.025USD	ステートが開始された回数とワークフロー実行に要したメモリ使用量に応じた課金 ・ 100 万リクエストあたり 1.00 USD ・GB-秒あたり 0.000004 ～ 0.00001 USD (実行量に応じて GB-秒 あたりの料金は変動)
実行履歴	API およびコンソールから確認可能	CloudWatch Logs から確認 API およびコンソールからは確認不可
実行セマンティクス	Exactly-once	At-least-once
アクティビティのサポート	あり	なし

- これまでの Step Functions (Standard Workflows) に加え、実行時間が短く大量に実行されるステートマシンに適した Express Workflows がすべてのリージョンで提供開始
- API Gateway や CloudWatch Events など、Standard と同様のイベントソースをサポート
- Standard と同様に ASL (Amazon States Language) でワークフローを定義

新機能リリース - お客様の声

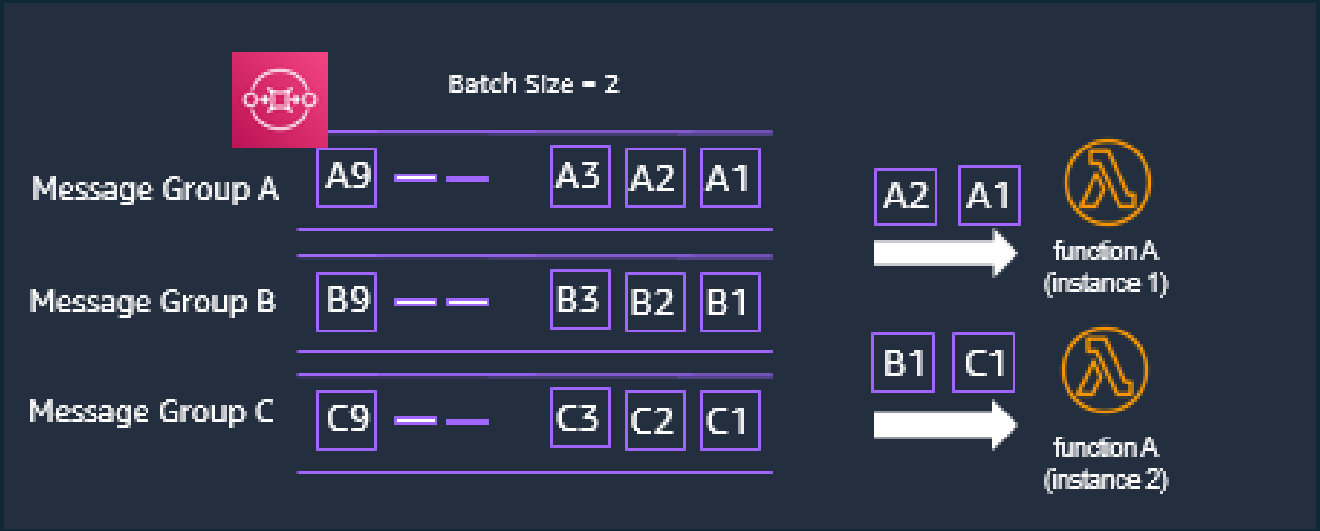
”よりサービスを統合し**イベントソース**によって
フルサーバーレスのアプリケーション間連携を、
追加の作業を行うことなく構築したい”

新機能のリリース

新しいAWSのサービスとイベントソースを統合可能に

- AWS Lambdaの**SQS FIFOキュー**サポート
- Step Functionsの**EMR**と**SageMaker**との統合
- Step Functionsの新しいステート：**Map、Nested Workflows**

Amazon SQS FIFO キューを Lambdaと統合



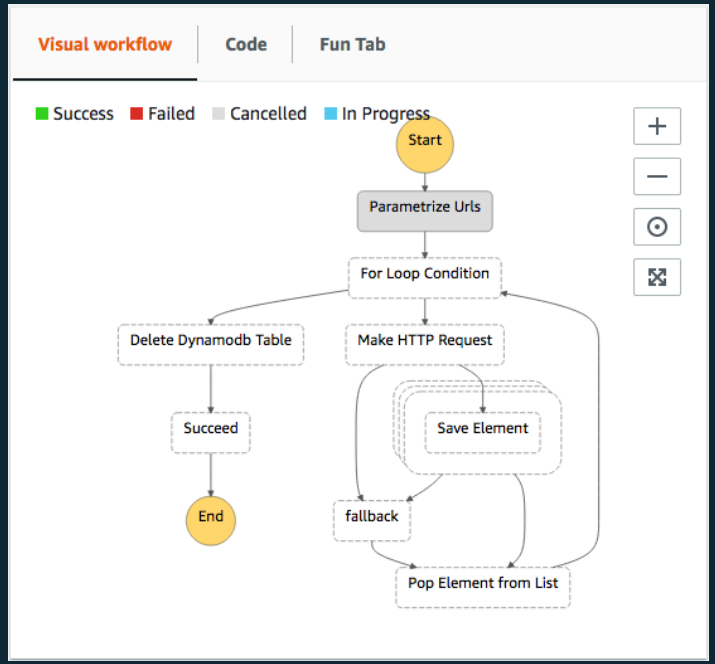
- グループごとに順序性を担保
- 最大同時実行数 = グループ数
- バッチ内に、複数のグループのメッセージが入る可能性がある

どのような場合に利用？ FIFOキュー

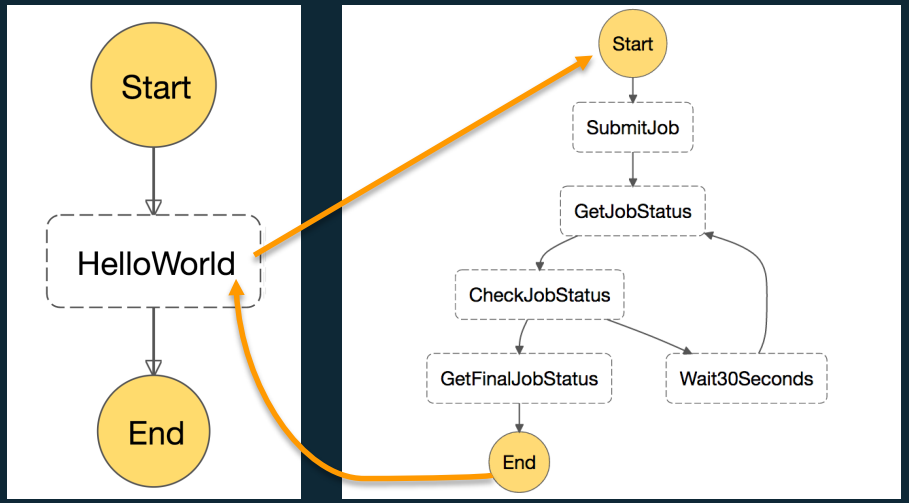
- より簡単なセットアップで、Kinesis Steramsよりも安価な順序付きレコードの処理
    - 例えばITオートメーションなど

Mapステート

- 自動的にデータ配列を並列処理
- データ処理やタスクの自動化のようなアプリケーションワークフローの、パフォーマンスと効率を最適化



ネストされた “child” ワークフロー



- ステートマシン自体をライブラリとする複雑なワークフローを構築
- カスタムコード不要でワークフローモジュールを状況に応じて変更
- 開発とデプロイを単純化

新機能リリース - お客様の声

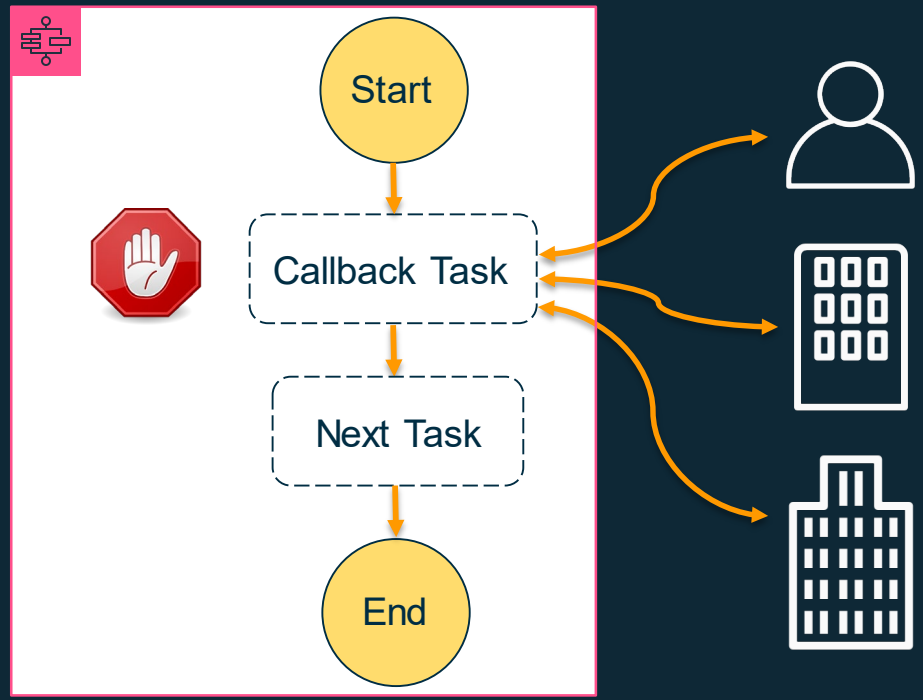
“ITオートメーションにLambdaとStep Functions
を使いたい – 度々人の承認が必要”

新機能リリース

ワークフローのポーズと再開を可能に

- Step Functionsがワークフローの**Callback Patterns**をサポート

非同期タスクのコールバックパターン



トークンを渡して外部のサービスを呼び出し。トークンを伴うコールバックが呼ばれるまでワークフローを停止

任意の待機時間 – 分、日、週、月

- 人のアクティビティ
- サードパーティのAPI
- レガシーアプリケーション

どのような場合に利用？ コールバックパターン

- 強化されたITオートメーション
- 大規模なITオートメーションのジョブで人間の承認がほとんどいつも実行される
- ジョブ管理者にとって、Mapステートとコールバックを利用が可能なStep Functionsは第一の選択肢

Empowering the Serverless developer and operator

新機能リリース - お客様の声

“アプリケーションをクラウドリソースの利用料金
を支払う前に**ローカル**でテストする必要がある”

新機能リリース

SAMのツールキットをデプロイがスムーズでローカルテストが可能なように改善

- Step Functionsのワークフローの**ローカルエミュレータ**
- SAM CLIを利用した**デプロイメントプロセス**の改善

AWS SAM CLIサーバーレスアプリケーションのデプロイ改善

- AWS SAM CLI を利用したデプロイに**ガイド付きインタラクティブモード**が追加
  - --guided オプションを付与すると、デプロイ先のスタック名やリージョンを指定するプロンプトが表示される

```
sam-deploy-test ) sam deploy --guided

Configuring SAM deploy
======================

        Looking for samconfig.toml :  Not found

        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [sam-app]: sam-deploy-test
        AWS Region [us-east-1]: ap-northeast-1
        #Shows you resources changes to be deployed and require a 'Y' to initiate deploy
        Confirm changes before deploy [y/N]: y
        #SAM needs permission to be able to create roles to connect to the resources in your template
        Allow SAM CLI IAM role creation [Y/n]: y
        Save arguments to samconfig.toml [Y/n]: y

        Looking for resources needed for deployment: Not found.
        Creating the required resources...
        Successfully created!

                Managed S3 bucket: aws-sam-cli-managed-default-samclisourcebucket-sykucxjohevd
                A different default S3 bucket can be set in samconfig.toml
```

AWS SAM CLIサーバーレスアプリケーションのデプロイをシンプルに

- AWS SAM CLI からのデプロイに**必要な S3 バケットを自動生成**可能に

- **3 ステップでより簡単**にサーバーレス開発をスタートできるようになった
  - Step 1 – アプリケーションのひな形を生成
    sam init
  - Step 2 – アプリケーションをローカル環境でビルド
    cd sam-app
    sam build
  - Step 3 – ビルドしたアプリケーションを AWS へデプロイ
    sam deploy --guided

AWS SAM CLIサーバーレスアプリケーションのデプロイをシンプルに

新しい**変更セットのサマリー**と**スタック出力**の自動的な表示

```
CloudFormation stack changeset
-----------------------------------------------------------------------------------------------------------------------------------------------
Operation                                 LogicalResourceId                                   ResourceType
-----------------------------------------------------------------------------------------------------------------------------------------------
+ Add                                     HelloWorldGetFunctionHelloWorldPermissionProd       AWS::Lambda::Permission
+ Add                                     HelloWorldGetFunctionRole                           AWS::IAM::Role
+ Add                                     HelloWorldGetFunction                               AWS::Lambda::Function
+ Add                                     HelloWorldPostFunctionHelloWorldPermissionProd      AWS::Lambda::Permission
+ Add                                     HelloWorldPostFunctionRole                          AWS::IAM::Role
+ Add                                     HelloWorldPostFunction                              AWS::Lambda::Function
+ Add                                     ServerlessRestApiDeployment7b2f80af31               AWS::ApiGateway::Deployment
* Modify                                  ServerlessRestApiProdStage                          AWS::ApiGateway::Stage
* Modify                                  ServerlessRestApi                                   AWS::ApiGateway::RestApi
- Delete                                  HelloWorldFunctionHelloWorldPermissionProd          AWS::Lambda::Permission
- Delete                                  HelloWorldFunctionRole                              AWS::IAM::Role
- Delete                                  HelloWorldFunction                                  AWS::Lambda::Function
- Delete                                  ServerlessRestApiDeployment47fc2d5f9d               AWS::ApiGateway::Deployment
-----------------------------------------------------------------------------------------------------------------------------------------------
```

```
Stack sam-app outputs:
-----------------------------------------------------------------------------------------------------------------------------------------------
OutputKey-Description                                                   OutputValue
-----------------------------------------------------------------------------------------------------------------------------------------------
HelloWorldGetFunction - Hello World Lambda Post Function ARN            arn:aws:lambda:us-west-2:0[redacted]:function:sam-app-HelloWorldGetFunction-
                                                                        ZGIGUVCGXEQO
HelloWorldGetFunctionIamRole - Implicit IAM Role created for Hello World post   arn:aws:iam::0[redacted]:role/sam-app-HelloWorldGetFunctionRole-YX1H58KGYQGF
function
HelloWorldPostFunction - Hello World Lambda Post Function ARN           arn:aws:lambda:us-west-2:0[redacted]:function:sam-app-
                                                                        HelloWorldPostFunction-19DN37OUDHNNG
HelloWorldApi - API Gateway endpoint URL for Prod stage for Hello World   https://0138lq7uqg.execute-api.us-west-2.amazonaws.com/Prod/hello/
function
HelloWorldPostFunctionIamRole - Implicit IAM Role created for Hello World post   arn:aws:iam::0[redacted]:role/sam-app-
function                                                                HelloWorldPostFunctionRole-166VRC53YQE3D
-----------------------------------------------------------------------------------------------------------------------------------------------
```

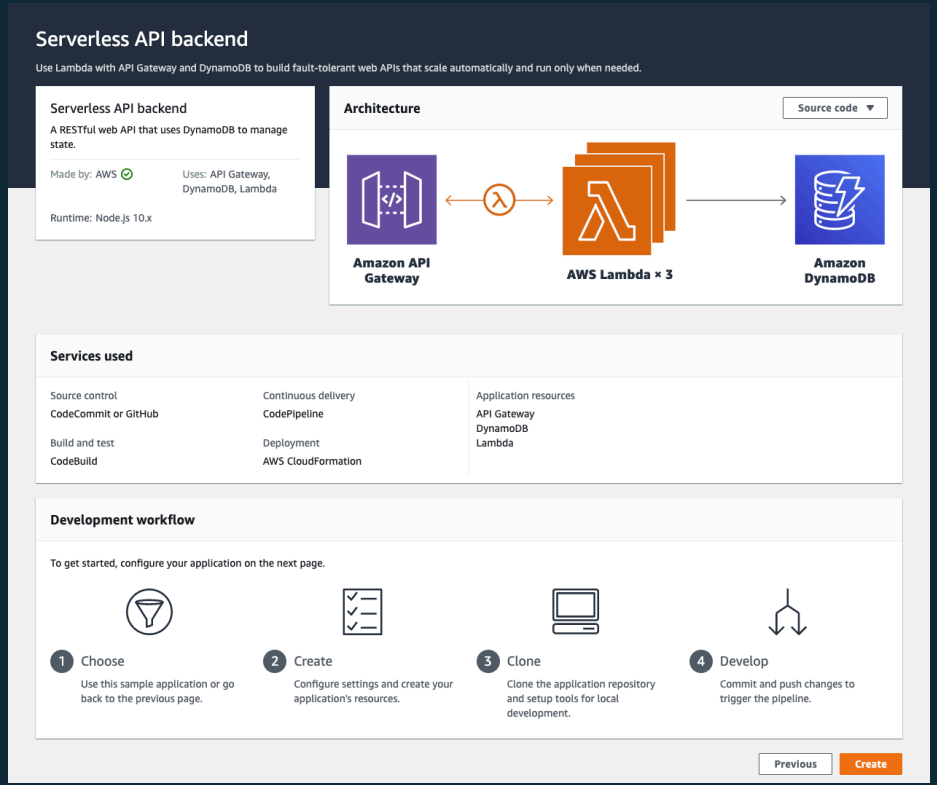
新機能リリース - お客様の声

“サーバーレスの開発とデプロイの**ベストプラクティス**に従うことを容易にしてほしい”

新機能リリース

セキュアでスケーラブルなCI/CDワークフローのセットアップを自動化

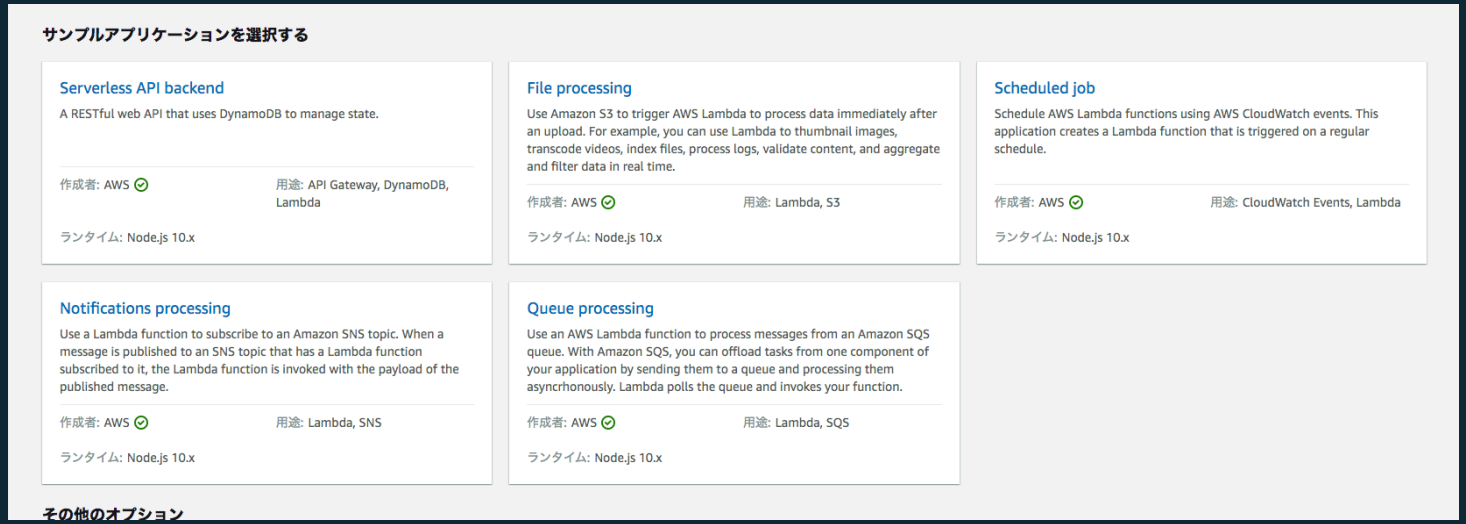
- **Start Right**
  - アプリケーションテンプレートのためのCI/CDパイプラインを自動的に生成



“サーバーレスアプリケーションの作成” 機能

- 特定ユースケースのサーバーレスアプリケーションを簡単に作成可能に
- アプリケーションだけではなく、CI/CD パイプラインも自動構築できる

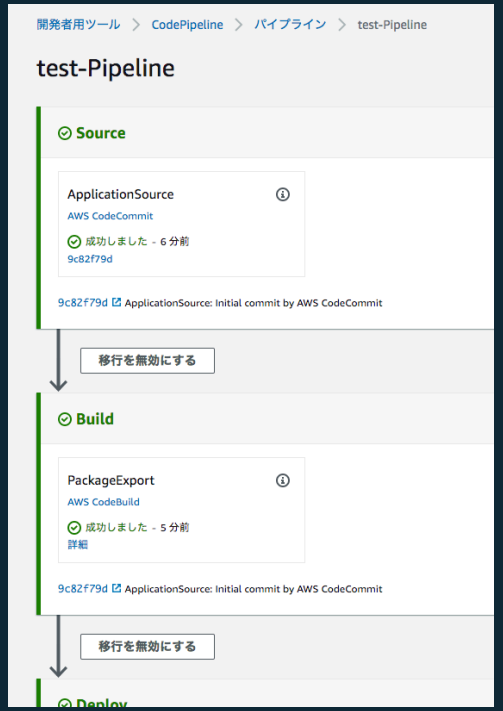
▼ サンプルアプリケーションの例



“サーバーレスアプリケーションの作成” 機能

- CI/CD パイプラインも同時にできあがる
- SAM テンプレートを書き換えて、CodeCommit に git push すれば、自動的に裏側のリソースにデプロイされる
- 現在、5つのユースケースに対応
- 言語は Node.js 10.x のみ

  ※言語の書き換えは可能

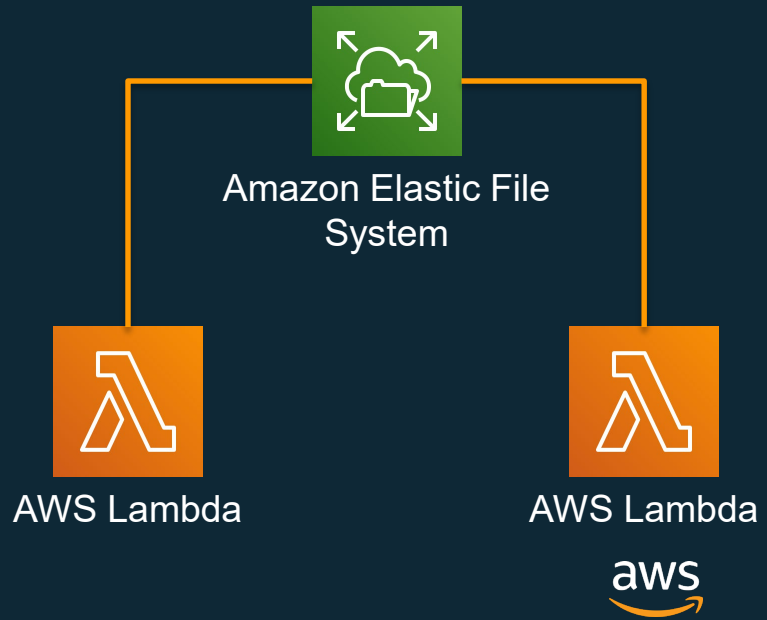


Amazon Elastic File System For AWS Lambda

使い慣れたファイルシステムインターフェイスを使用して、複数のLambda関数の同時実行環境にわたってデータを保存および共有

ユースケース

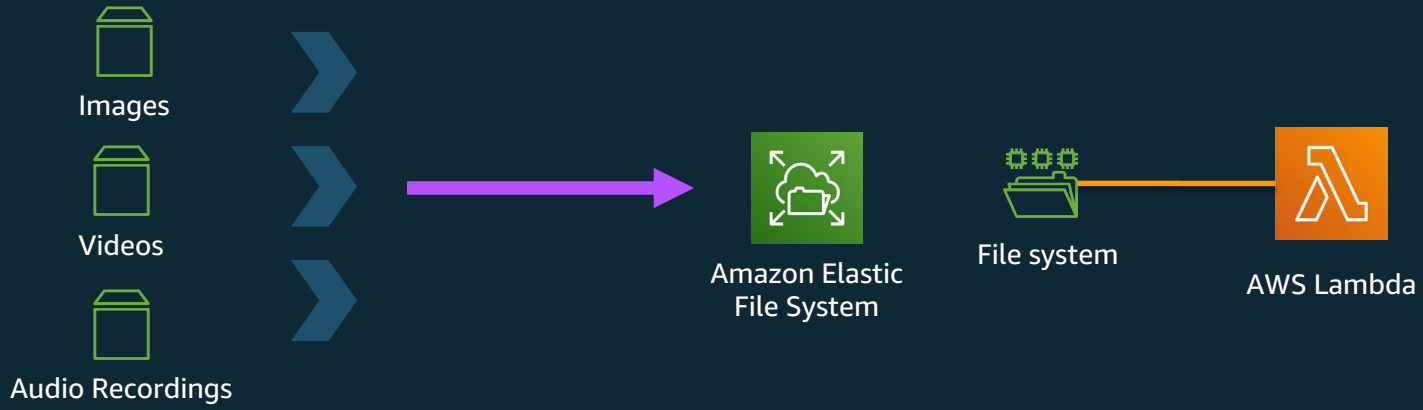
- データ集約型Appを構築
- 大きなLibとモデルをロード
- 大量のデータを分散処理
- 関数、コンテナ、インスタンス間データ共有
- 開発効率の向上
- カスタムコード開発の削減
- パフォーマンス向上、待ち時間を短縮



サーバーレス でのメディア処理プロセス

メディア処理実装を簡素化

小さなファイルから大容量ファイルまでを一元管理し、
処理効率を改善



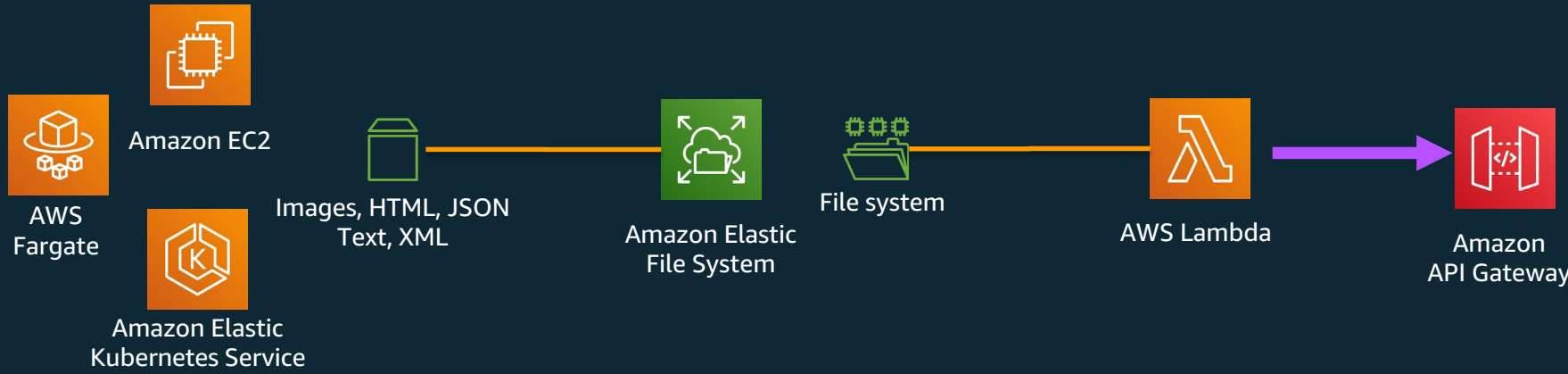
多種のファイルを一元管理

AWS Lambda を利用した AI/ML や アナリティクス

機械学習モデルの読み込み
追加のコードライブラリのロード

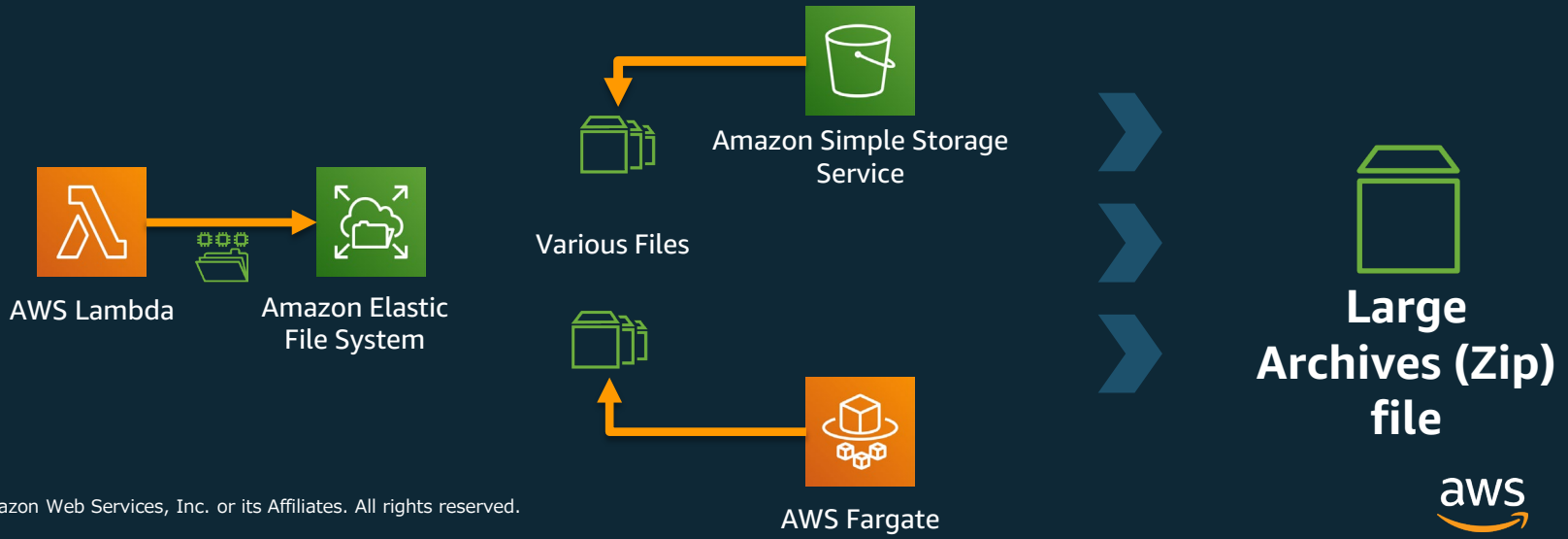
EC2やコンテナワークロードと協調したリアルタイム処理

任意のコンピュート層から、EFSに配置
Lambdaでリアルタイムに読み込み
サーバーレス サービスでフロントエンドに提供



大容量のファイル操作 (バックアップ、リストア、アーカイブ)

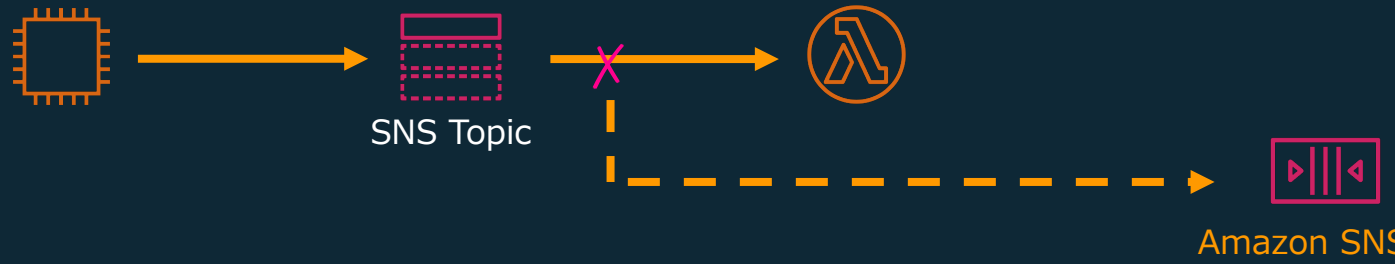
任意のAWSサービスからファイルを取得
ファイル結合、ファイル移動、ファイル圧縮
任意のAWSサービスにアーカイブ



その他のアップデート

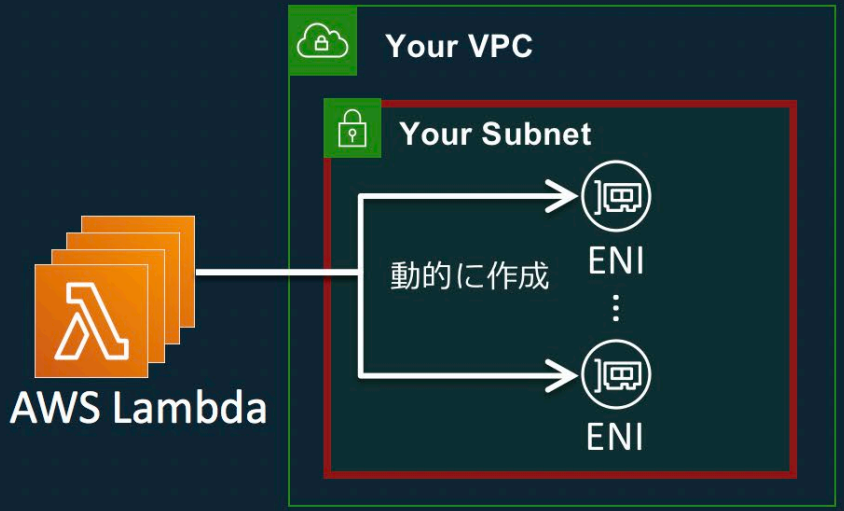
Amazon SNS Adds Support for Dead-Letter Queues (DLQ)

- SNS のサブスクリプションにデッドレターキュー(DLQ)を設定可能
- サブスクリプションがエンドポイントに届かない場合に、
  そのメッセージを保存しておくことができるので、
  アプリケーションの復元性、耐久性を向上



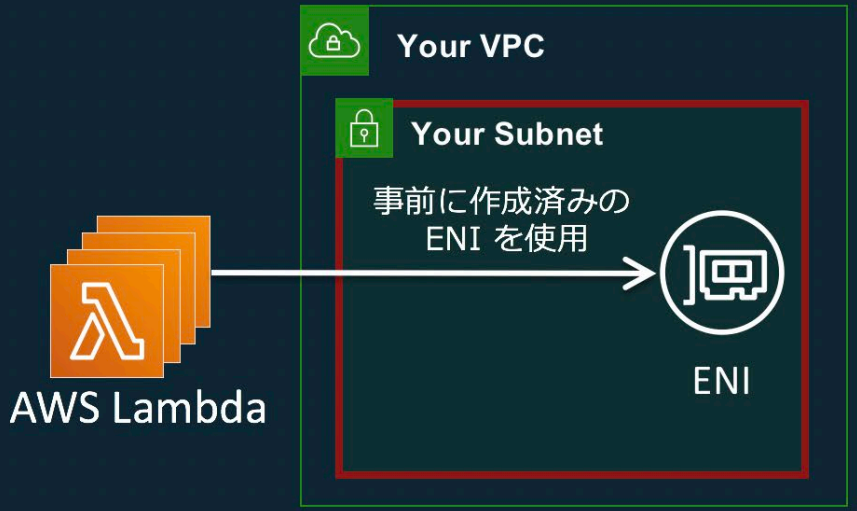
AWS Lambda VPC接続の改善

改善前



Lambda 関数の実行時、必要に応じて動的に
ENI が作成される
→ そのとき**10 秒～３０秒**ほどかかっていた

改善後

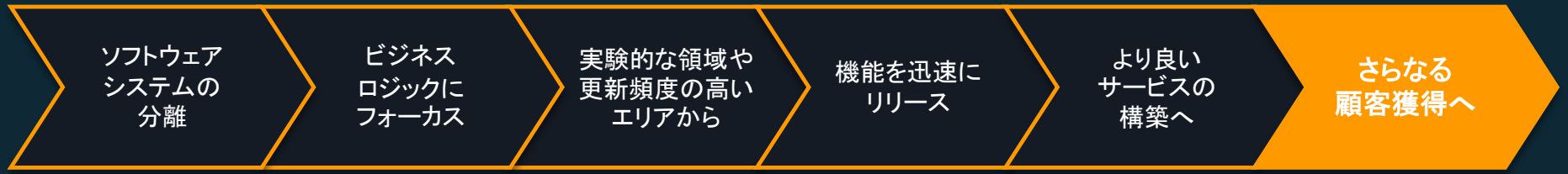


Lambda 関数の作成時または VPC 設定有効化時に
あらかじめ ENI を作成し、以降はそれを使える

最後に、、、

今、まさにパラダイムシフトのとき!

75% *の組織は今後 2 年でサーバーレステクノロジーの利用を計画をしている*



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

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

@_kensh

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

過去資料
https://amzn.to/JPArchive