



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]

Serverless モニタリング

ソリューションカットシリーズ

Solutions Architect 下川 賢介
2019/8/20

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



過去資料
<https://amzn.to/JPArchive>



自己紹介

□ 名前

下川 賢介

□ 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

Serverless Specialist Solutions Architect



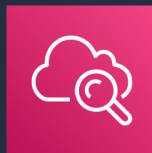
□ 好きなAWSのサービス



AWS Lambda



Amazon API Gateway



Amazon CloudWatch

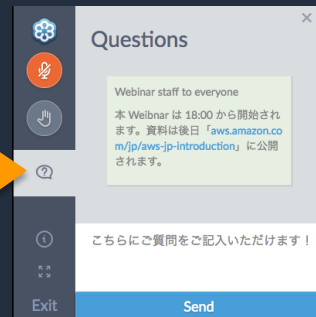
AWS Black Belt Online Seminar とは

「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、Amazon ウェブサービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

質問を投げることができます！

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

- ① 吹き出しをクリック
- ② 質問を入力
- ③ Sendをクリック



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

内容についての注意点

- 本資料では2019年8月20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.

本日のアジェンダ

- モニタリングについて
- Metricsの理解
- Log運用と可視化
- Traceを使った可視化

モニタリングについて



モニタリングの目的

担当者のロールによって、モニタリングをする意味も大きく異なってきます。

ユーザ体験の把握

監査対応

品質向上

不正アクセス検知

コンプライアンス
準拠

コスト効率

性能劣化検知

Inventory and classification



AWS Systems Manager



AWS Config




AWS X-Ray



Amazon CloudWatch

Monitoring and analytics

Service request



AWS Service Catalog




AWS CloudTrail

Packaging and delivery

Provisioning and orchestration




AWS OpsWorks



AWS CloudFormation

Cost management and resource optimization




AWS Trusted Advisor

Cloud migration, backup, and DR



AWS Snowball



AWS DMS



AWS SMS

security and compliance



多くの 3rdParty ツールとの連携が可能





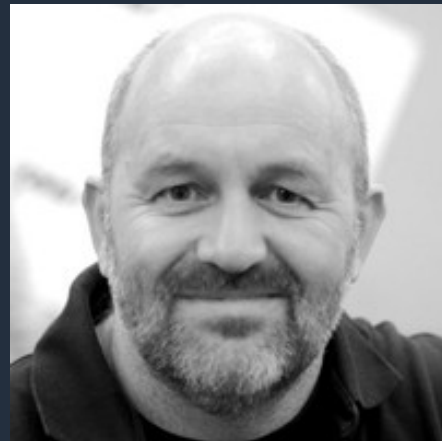
誰がモニタリングするのか

伝統的なシステムでは開発と運用の間に壁がある。
そんな壁はぶん投げてしまえ。
忘れてしまってもいい。Amazonでは不要だ。

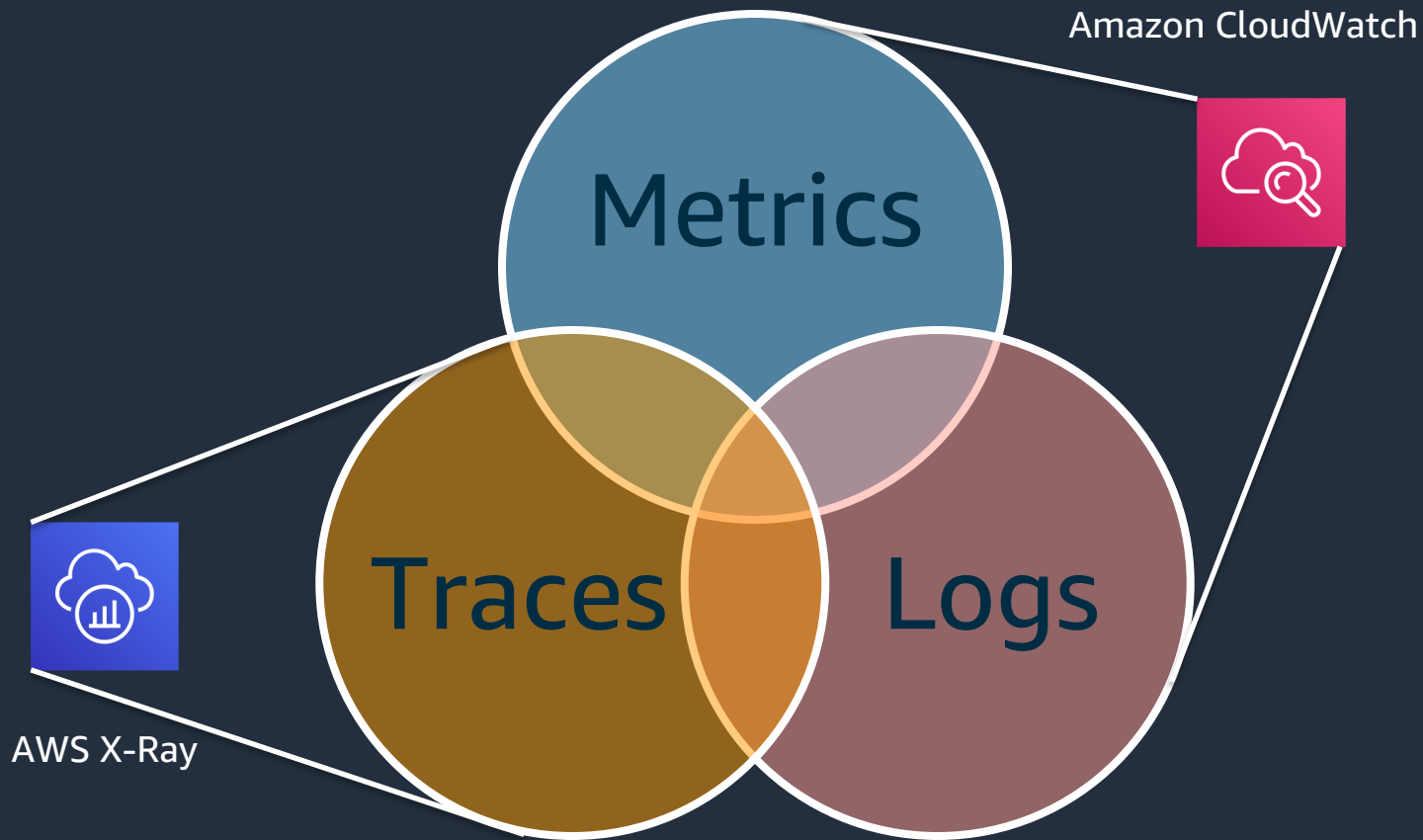
you build it, you run it

開発者も日々の運用に参加しなさい。
そうすればカスタマーのフィードバックが得られる。
これが、サービスの質を高めるのだ。

-- Werner Vogels in May 2006 / Amazon.com CTO

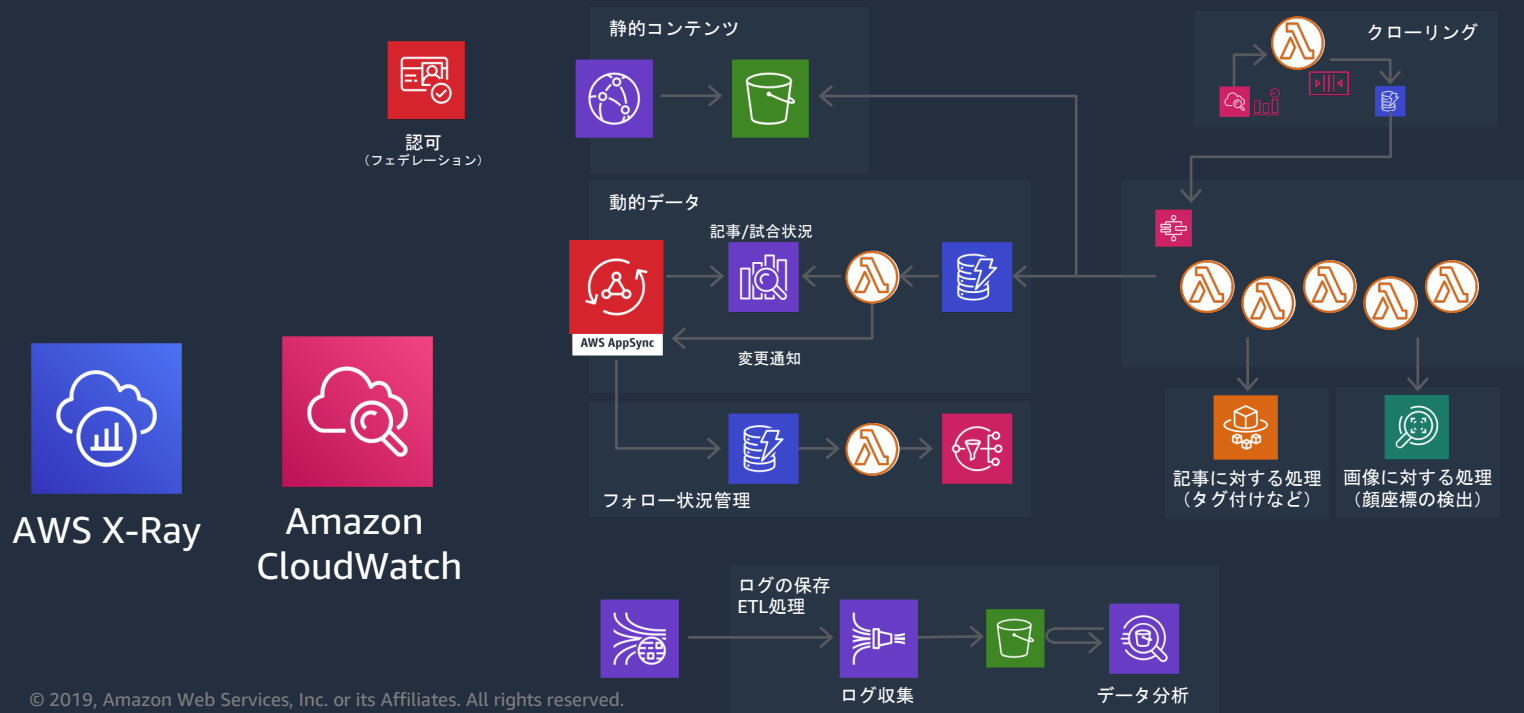


モニタリングサービス

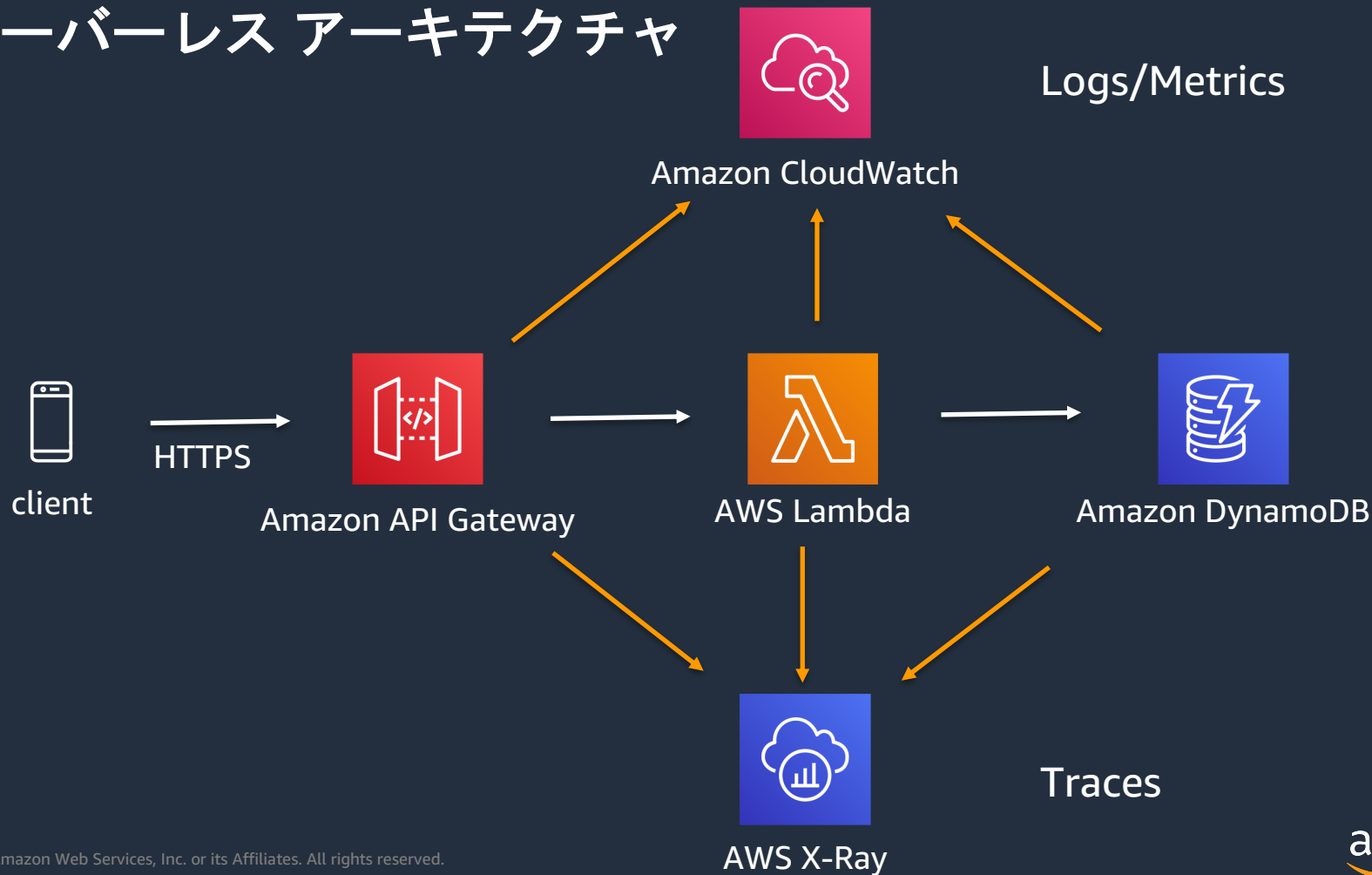


サーバーレスのモニタリング

サーバーレス アーキテクチャは、多くの場合分散しており、個々の分散パーツがモニタリングを必要とします。



サーバーレス アーキテクチャ



Metricsの理解





AWS LambdaのMetrics

AWS Lambda の Metrics

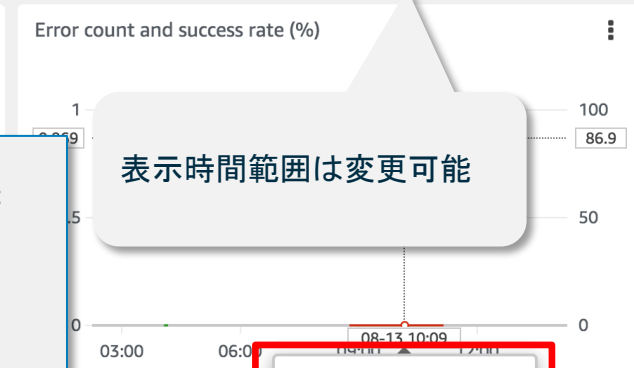
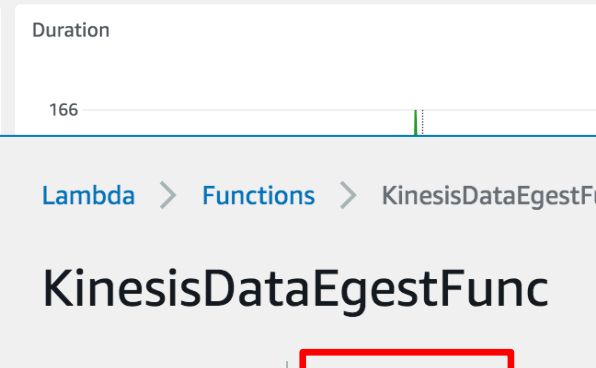
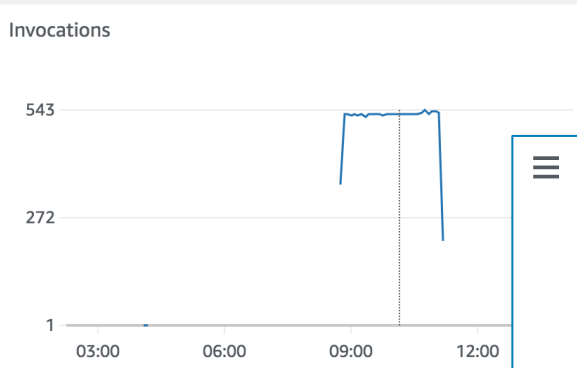
項目	単位	意味
Invocations	count	Lambdaの実行回数を計測したもの。=課金カウント ≠同時実行数
Duration	ms	Lambdaの実行時間を計測。（課金は100msで切り上げる） ※コールドスタートのLambda起動までの時間は含まない
Errors, Availability	count	Lambdaが正常終了しなかった回数を計測したもの Availabilityは%で表示される
Throttles	count	Throttleの発生回数 <u>アカウントにおける</u> Lambdaの同時起動数超過が発生している
IteratorAge	ms	ストリーム(Kinesis/DynamoDB)でのみ利用。バッチサイズ分取得したレコード終端の時刻とLambdaがイベントとして受信した時刻差で表示される
DeadLetterErrors	count	DLQを設定し、そのDLQへの書き込みが失敗すると増加する。

AWS Lambda の Metrics



Amazon CloudWatch
Dashboards
に追加可能

Add to dashboard | 1h 3h **12h** 1d 3d 1w | Refresh



Lambda > Functions > KinesisDataEgestFunc

KinesisDataEgestFunc

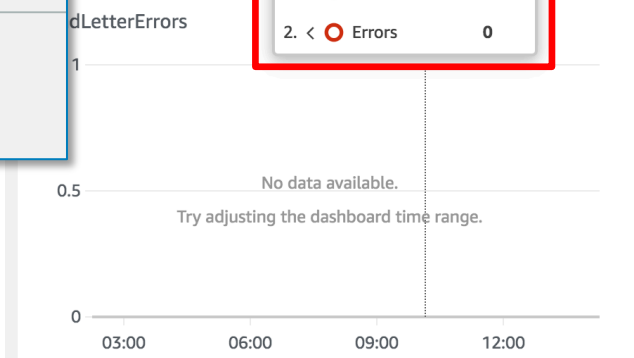
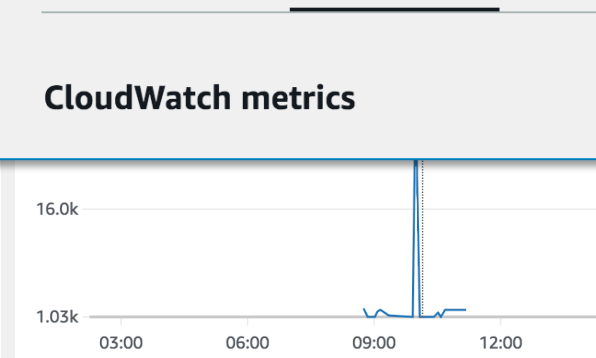
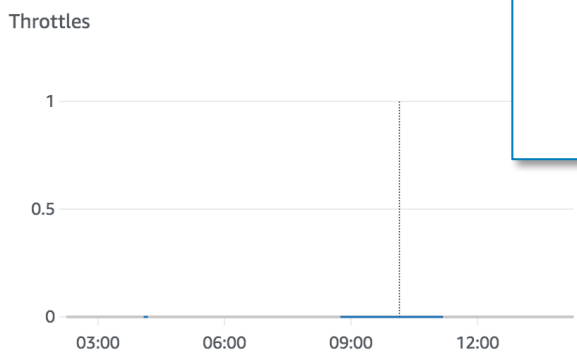
Configuration | **Monitoring**

CloudWatch metrics

表示時間範囲は変更可能

2019-08-13 10:10 UTC

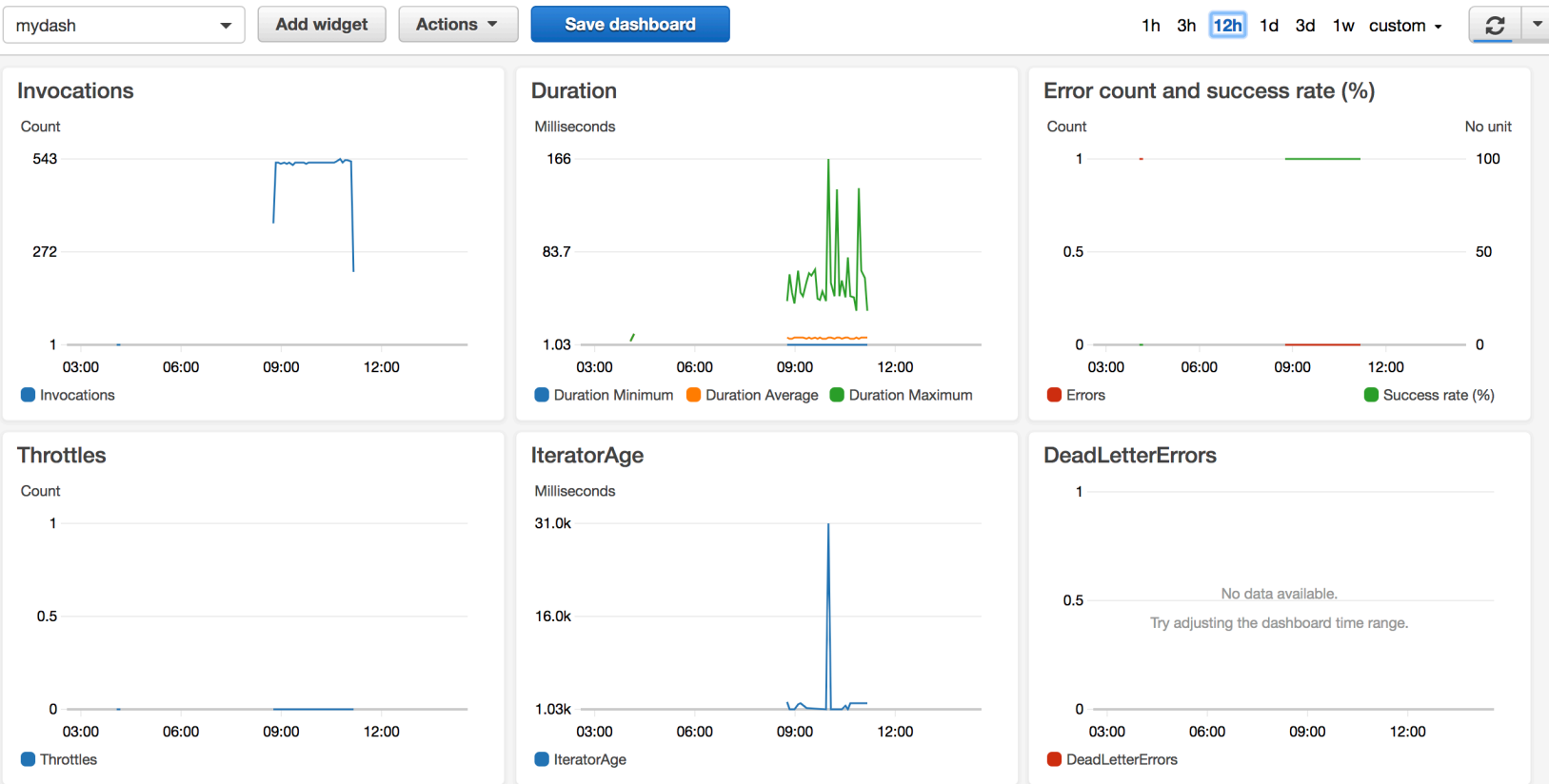
- 1. > Success rate (%) 100
- 2. < Errors 0



Amazon CloudWatch Dashboardsに追加された Metrics



- CloudWatch
- Dashboards**
- + mydash
- Alarms
 - ALARM 0
 - INSUFFICIENT 0
 - OK 6
- Billing
- Events
- Rules
- Event Buses
- Logs
- Insights
- Metrics
- Settings
- Favorites
- [+ Add a dashboard](#)



AWS Lambda の Metrics

項目	単位	意味
Invocations	count	Lambdaの実行回数を計測したもの。=課金カウント ≠同時実行数
Duration	ms	Lambdaの実行時間を計測。（課金は100msで切り上げる） ※コールドスタートのLambda起動までの時間は含まない
Errors, Availability	count	Lambdaが正常終了しなかった回数を計測したもの Availabilityは%で表示される
Throttles	count	Throttleの発生回数 <u>アカウントにおける</u> Lambdaの同時起動数超過が発生している
IteratorAge	ms	ストリーム(Kinesis/DynamoDB)でのみ利用。バッチサイズ分取得したレコード終端の時刻とLambdaがイベントとして受信した時刻差で表示される
DeadLetterErrors	count	DLQを設定し、そのDLQへの書き込みが失敗すると増加する。

AWS Lambdaの起動パターン

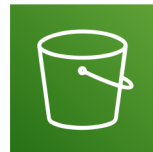
Invocations

RequestResponseベース



Amazon API Gateway

非同期Eventベース



Amazon Simple Storage Service(S3)



Amazon Simple Notification Service(SNS)

Stream Pollingベース



Amazon Kinesis Data Streams



Amazon DynamoDB Streams

Queue Pollingベース



Amazon Simple Queue Service (SQS)

□ RequestResponseベース



request数 : Lambda実行数 = n : n
(実行数の比)

AWS Lambdaの起動パターン

Invocations

□ 非同期Eventベース



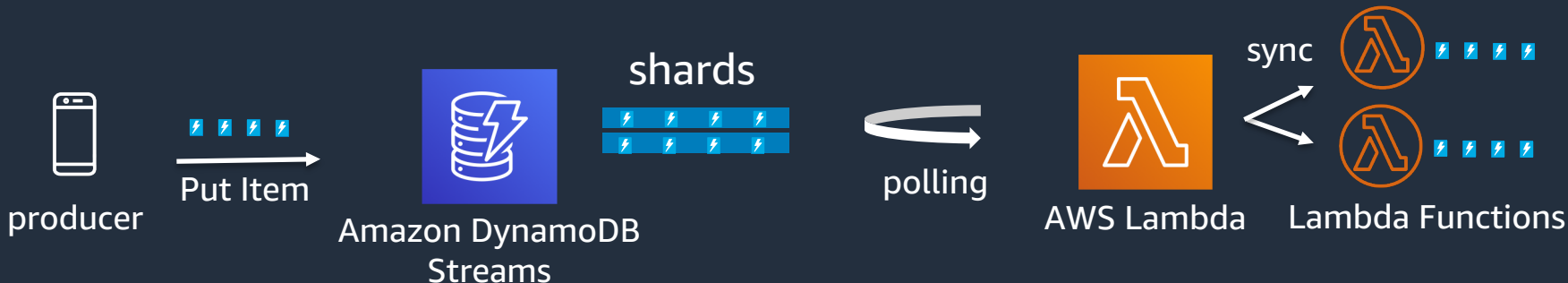
Event数 : Lambda実行数 = n : n
(実行数の比)

AWS Lambdaの起動パターン

Invocations

□ Stream Pollingベース

自動polling
指定したbatch size分のデータ取得
最大1万件まで指定可能



Record数 : Lambda実行数 = $n : n/\text{batch}$ 処理数平均 (実行数の比)
Shard数 : Lambda同時実行数 = $m : m$ (同時実行数の比)

□ Queue Pollingベース



message数 : Lambda実行数 = $n : n / \text{batch}$ 処理数平均 (実行数の比)
Queue内のmessage数に応じてLambda 起動数がauto scale (同時実行数)

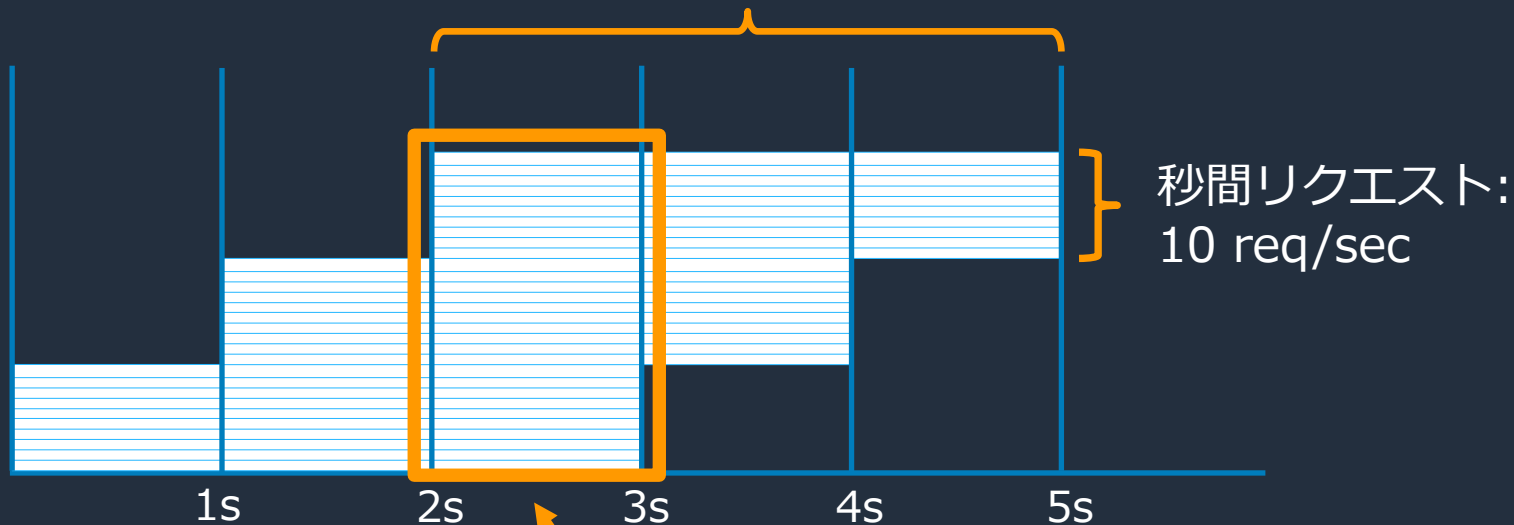
AWS Lambda の Metrics

項目	単位	意味
Invocations	count	Lambdaの実行回数を計測したもの。=課金カウント ≠同時実行数
Duration	ms	Lambdaの実行時間を計測。（課金は100msで切り上げる） ※コールドスタートのLambda起動までの時間は含まない
Errors, Availability	count	Lambdaが正常終了しなかった回数を計測したもの Availabilityは%で表示される
Throttles	count	Throttleの発生回数 <u>アカウントにおける</u> Lambdaの同時起動数超過が発生している
IteratorAge	ms	ストリーム(Kinesis/DynamoDB)でのみ利用。バッチサイズ分取得したレコード終端の時刻とLambdaがイベントとして受信した時刻差で表示される
DeadLetterErrors	count	DLQを設定し、そのDLQへの書き込みが失敗すると増加する。

同時実行数とは

Throttles

関数の平均実行時間: 3s / exec



同時実行数 = “同時”に実行されているタイミング
= 3 × 10 = 30

Throttlingの対策としては

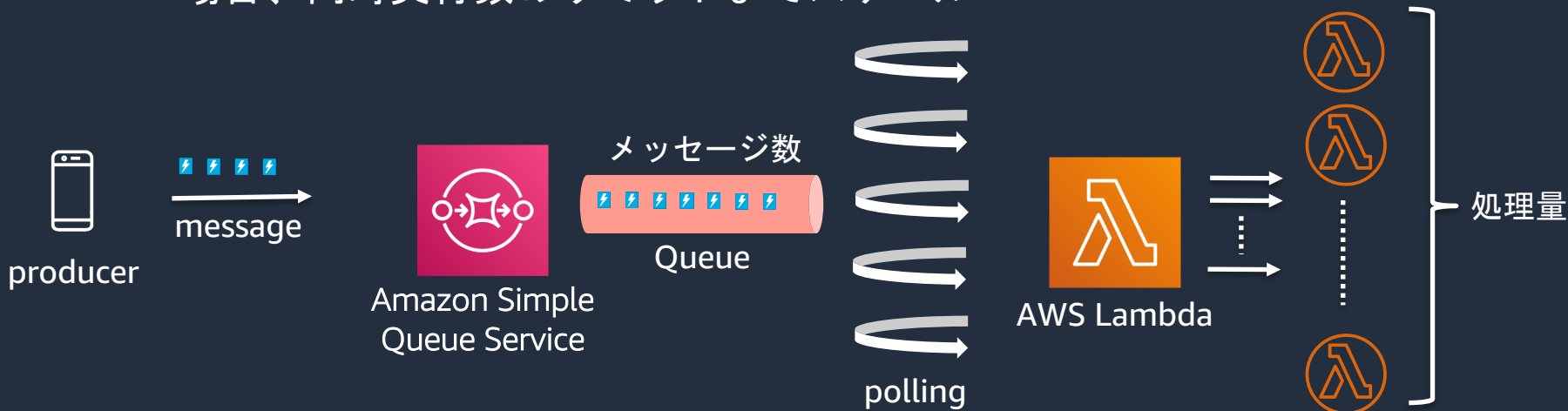
- アカウントの同時実行数の制限緩和を申請する
- SQSなどを用いて、非同期にリクエストを処理するようにアーキテクチャを変更する
- Lambda関数をチューニングして平均実行時間を下げる

Queue Pollingベースにおける注意事項

Throttles

メッセージ到達時の挙動

- 5つのパラレルロングポーリング接続を使用してSQSのQueueをポーリング
 - メッセージ数 > 処理量 の傾向が続き、最終的に処理が追いつかない場合、同時実行数のリミットまでスケール



Queue Pollingベースにおける注意事項

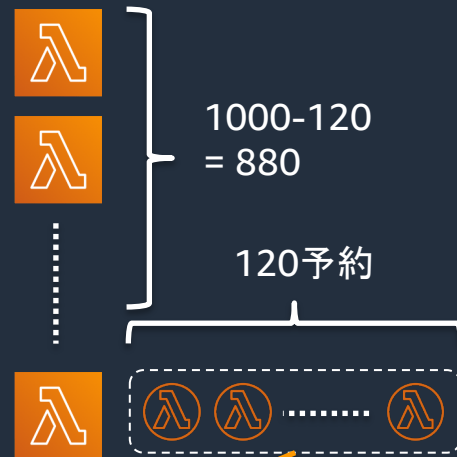
自動スケール

- Lambda関数単位に同時実行数を制限しない場合
 - アカウントの上限までスケールする可能性
 - 他のLambda関数の起動を妨げる恐れがある

対処方法

- 大規模に使用する際には同時実行数の設定も検討

AWS Account : 1000 in Tokyo



Concurrency

Unreserved account concurrency **880**

Use unreserved account concurrency

Reserve concurrency

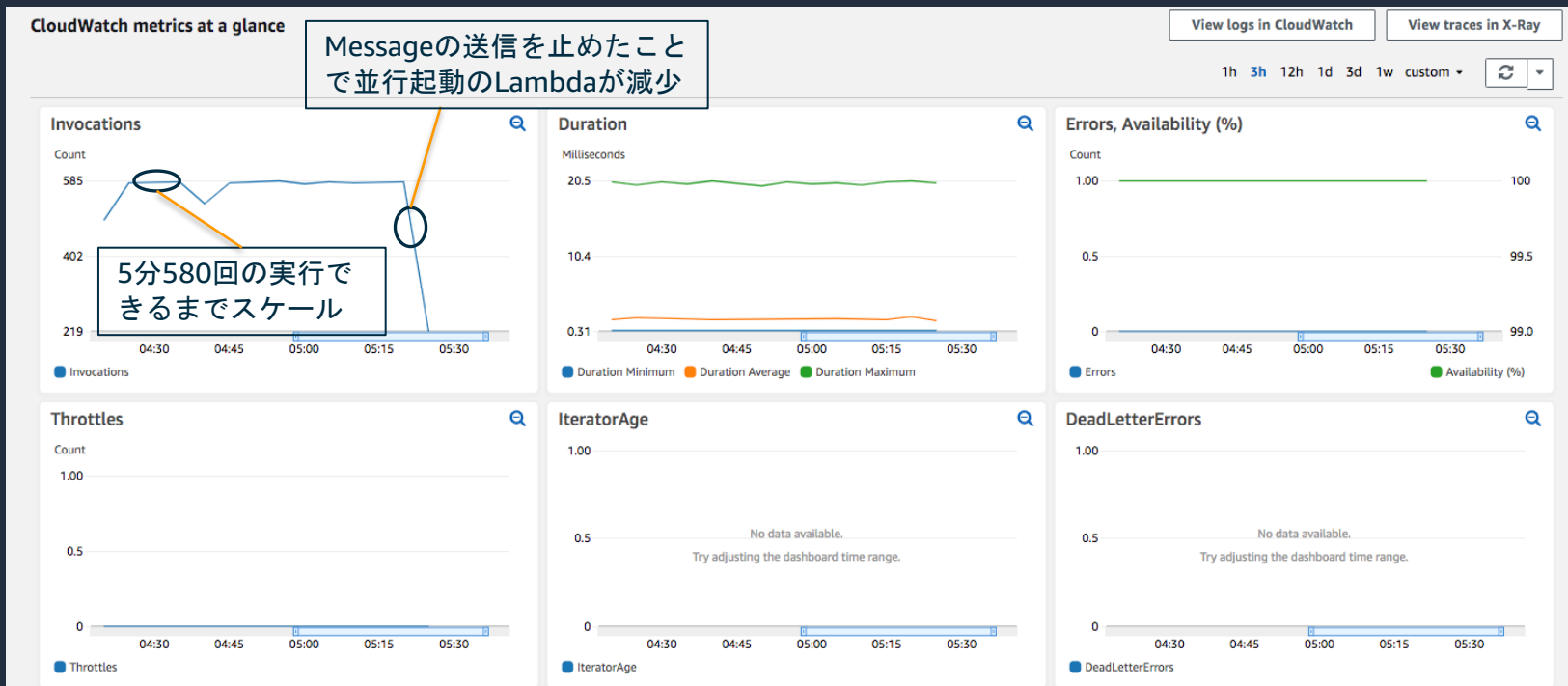
120

Throttles

Queue Pollingベースでの監視事項

Throttles

- 0.5秒間隔でMessageをQueueに送信
(metricsは標準の5min)



AWS Lambda の Metrics

項目	単位	意味
Invocations	count	Lambdaの実行回数を計測したもの。=課金カウント ≠同時実行数
Duration	ms	Lambdaの実行時間を計測。（課金は100msで切り上げる） ※ コールドスタートのLambda起動までの時間は含まない
Errors, Availability	count	Lambdaが正常終了しなかった回数を計測したもの Availabilityは%で表示される
Throttles	count	Throttleの発生回数 アカウントにおける Lambdaの同時起動数超過が発生している
IteratorAge	ms	ストリーム(Kinesis/DynamoDB)でのみ利用。バッチサイズ分取得したレコード終端の時刻とLambdaがイベントとして受信した時刻差で表示される
DeadLetterErrors	count	DLQを設定し、そのDLQへの書き込みが失敗すると増加する。

Lambda Function Lifecycle

Duration



- コンテナ生成
- S3からのZIPダウンロード
- ZIPファイルの展開
- Durationには含まれない

Lambda Function Lifecycle

Duration



- 各ランタイムの初期化処理
- グローバルスコープ処理
- **Durationには含まれない**

Lambda Function Lifecycle

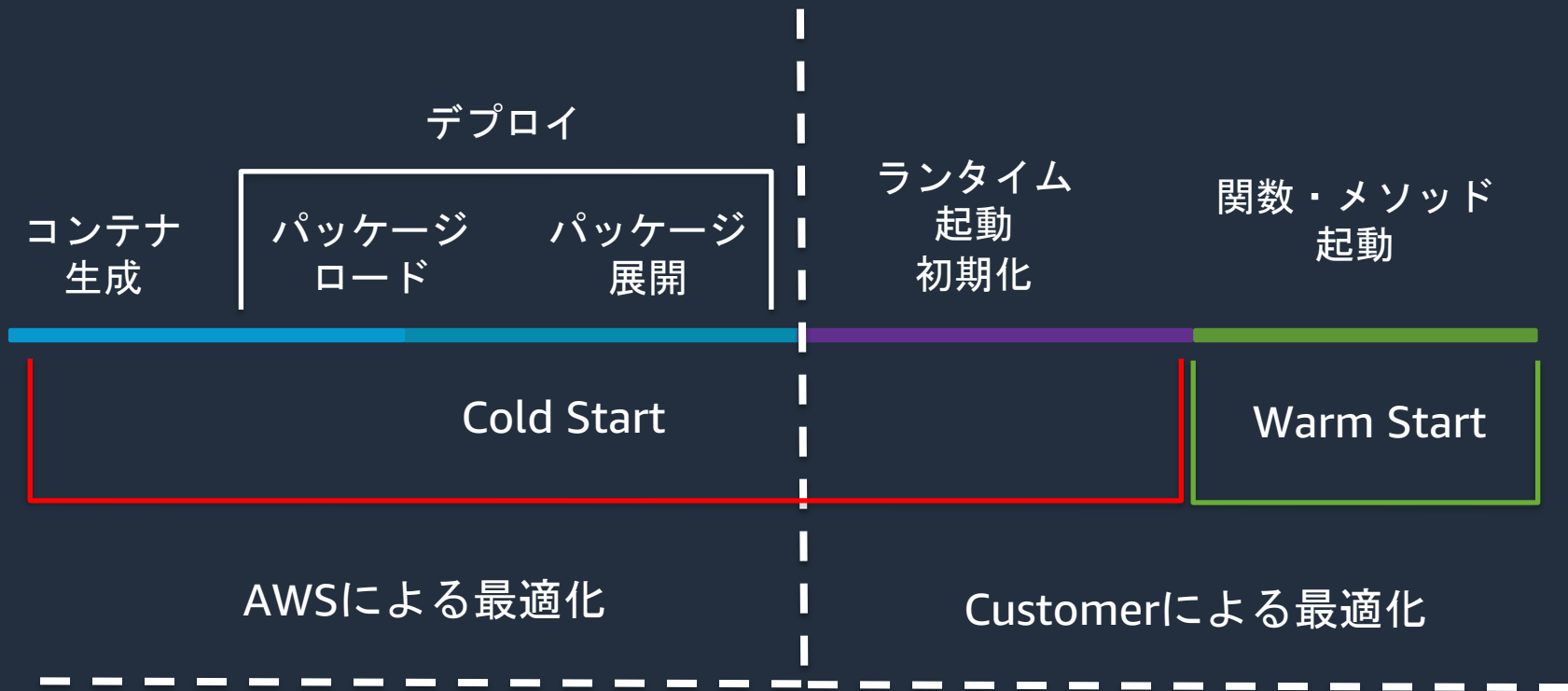
Duration



- ハンドラーで指定した関数/メソッドの実行
- **Duration値は実行時間**

Lambda Function Lifecycle

Duration



VPC Lambda Function Lifecycle

Duration



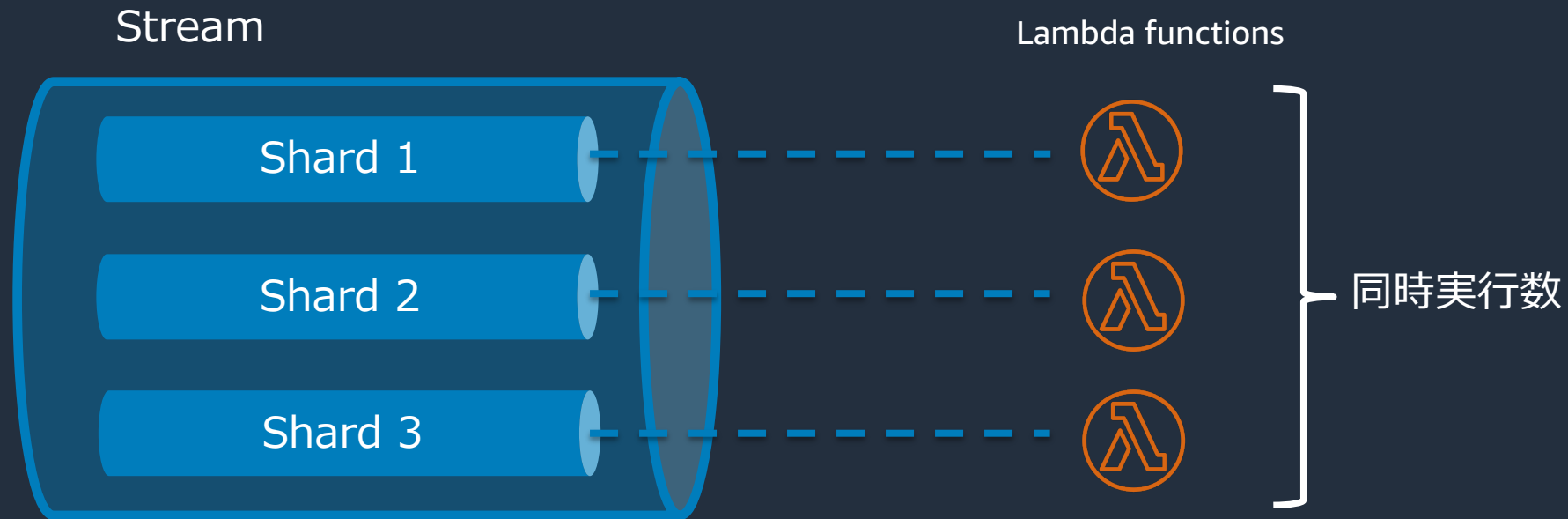
- VPCを利用する場合だけ
- 10秒～30秒かかる
- Durationには含まれない

AWS Lambda の Metrics

項目	単位	意味
Invocations	count	Lambdaの実行回数を計測したもの。=課金カウント ≠同時実行数
Duration	ms	Lambdaの実行時間を計測。（課金は100msで切り上げる） ※コールドスタートのLambda起動までの時間は含まない
Errors, Availability	count	Lambdaが正常終了しなかった回数を計測したもの Availabilityは%で表示される
Throttles	count	Throttleの発生回数 <u>アカウントにおける</u> Lambdaの同時起動数超過が発生している
IteratorAge	ms	ストリーム(Kinesis/DynamoDB)でのみ利用。バッチサイズ分取得したレコード終端の時刻とLambdaがイベントとして受信した時刻差で表示される
DeadLetterErrors	count	DLQを設定し、そのDLQへの書き込みが失敗すると増加する。

Stream PollingベースのLambda同時実行数

Errors



Stream PollingベースのErrors Metrics監視

Errors

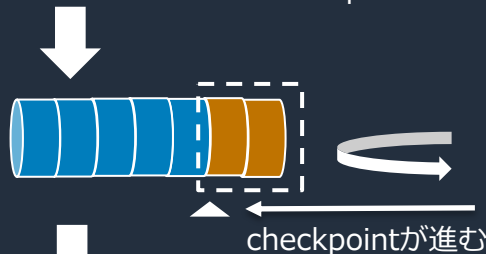
Q: AWS Lambda は Amazon Kinesis ストリームおよび Amazon DynamoDB ストリームからのデータをどのように処理しますか?

(snip)

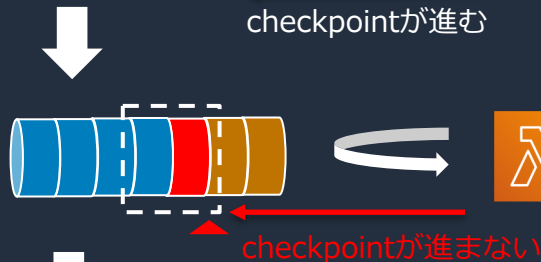
あるレコードに対する呼び出しがタイムアウトになったり制限されたりした場合、またはその他のエラーが発生した場合、Lambda は成功するまで (またはレコードが 24 時間の有効期限切れになるまで) 次のレコードに移動しません。異なるシャードにあるレコード間の順序は保証されず、各シャードの処理は並行して行われます。



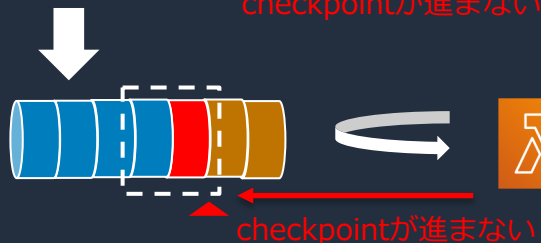
Batch Size : 2



2件read、正常終了



次の2件read、1件目のデータで異常終了



同checkpointからread (有効期限まで)

<https://aws.amazon.com/jp/lambda/faqs/>

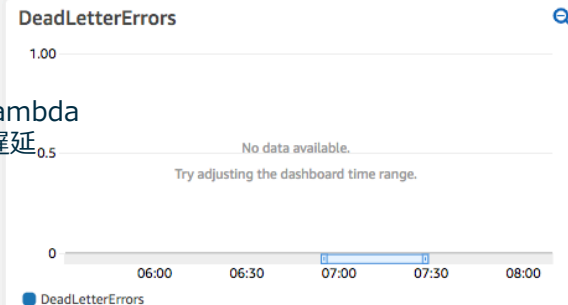
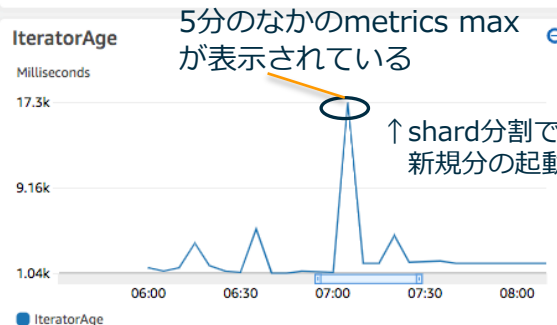
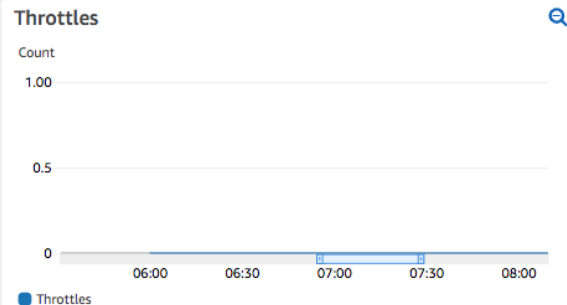
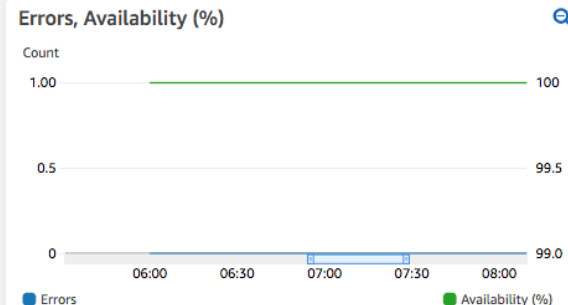
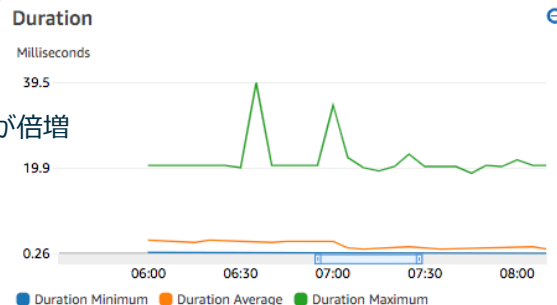
Errors metricsが上がった場合は、**処理がブロック**されている恐れがある。

- Lambdaのプログラム改修
 - Streamの場合、Errorsを挙げないように設計/テストをすることが重要
 - CloudWatch Logsにエラーレベルで記録
 - SQSに異常レコードを登録して、Shardを進める
- LambdaをトリガーするStarting positionをTRIM_HORIZONから、LATEST / AT_TIMESTAMPに変更（データを読まずに捨てることになるので注意）
 - TRIM_HORIZON – Shard内の最も古いレコードから読み込み
 - LATEST – Shard内の最新レコードから読み込み
 - AT_TIMESTAMP – 指定したtimestampポジションから読み込み

Stream Pollingベースでの監視事項

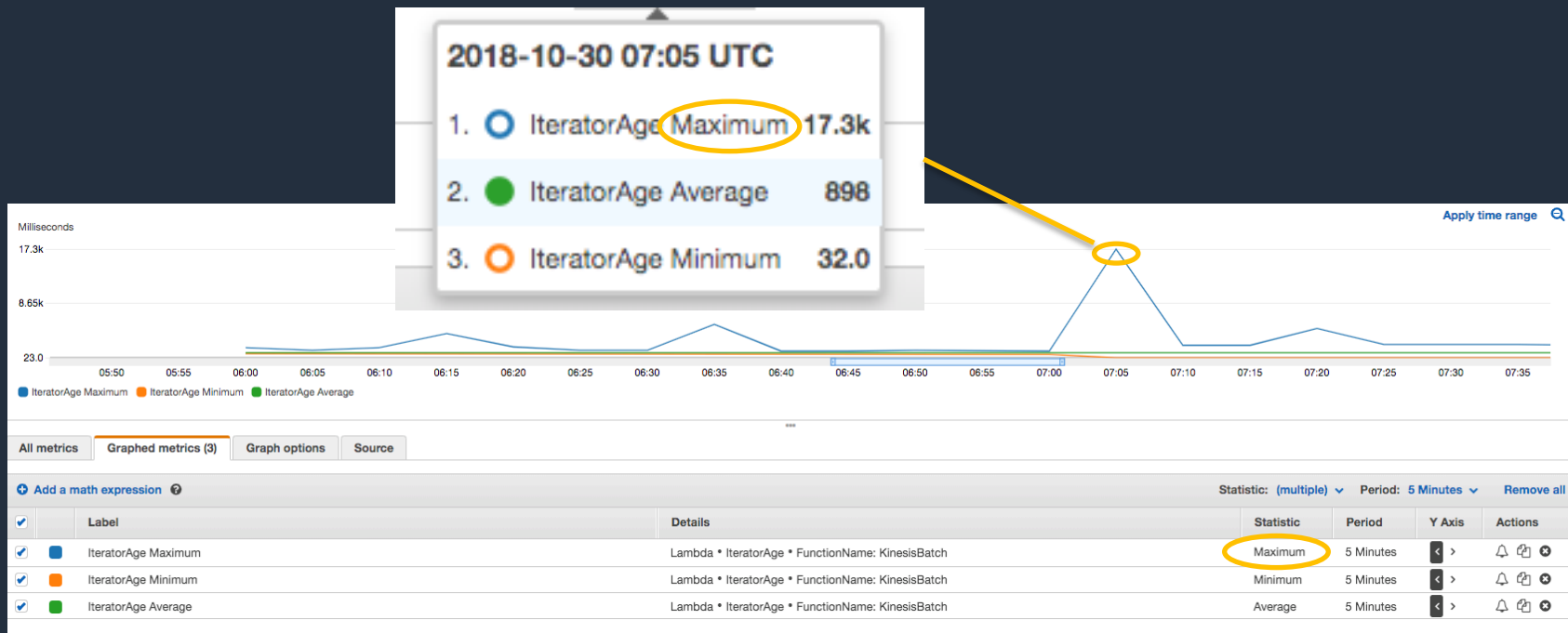
IteratorAge

1. 0.3秒間隔でrecordをKinesis Streamsへ送信
2. 途中でshard分割を実行



IteratorAgeをCloudwatch metricsでみると

IteratorAge



「5分のなかのmetrics Maximumが表示されている」

IteratorAgeの”Average”が大きくなっている場合は処理遅れが想定される。

- Lambda実行環境のスペック向上により実行時間の短縮を考える
 - メモリ増強など
- Kinesis Streamsのshardを分割
 - shardを分割することで並行処理数が増える
 - 東京リージョンの場合、1 streamあたり、200shards分割まで可能
https://docs.aws.amazon.com/ja_jp/streams/latest/dev/service-sizes-and-limits.html

※デフォルトのShard上限数は、AWS リージョン 米国東部（バージニア北部）、米国西部（オレゴン）、および 欧州（アイルランド）で 500 Shardsです。その他のリージョンのデフォルトのシャード制限はすべて 200 Shardsです。

AWS Lambda の Metrics

項目	単位	意味
Invocations	count	Lambdaの実行回数を計測したもの。=課金カウント ≠同時実行数
Duration	ms	Lambdaの実行時間を計測。（課金は100msで切り上げる） ※コールドスタートのLambda起動までの時間は含まない
Errors, Availability	count	Lambdaが正常終了しなかった回数を計測したもの Availabilityは%で表示される
Throttles	count	Throttleの発生回数 <u>アカウントにおける</u> Lambdaの同時起動数超過が発生している
IteratorAge	ms	ストリーム(Kinesis/DynamoDB)でのみ利用。バッチサイズ分取得したレコード終端の時刻とLambdaがイベントとして受信した時刻差で表示される
DeadLetterErrors	count	DLQを設定し、そのDLQへの書き込みが失敗すると増加する。

Dead Letter Queue(DLQ)とは

- 最大リトライ回数終了後に、LambdaのEventペイロードをSQSの標準Queue/SNSトピックに送信する機能
- 非同期イベントでLambdaが呼び出された場合に利用可

DeadLetterErrorsとは設定済みのDLQに書き込むことができない場合に、増加する

Debugging and error handling

DLQ resource [Info](#)
Choose the AWS service to send the event payload to after maximum retries are exceeded.

Amazon SQS

SQS Queue
The name of the existing queue or topic to serve as the DLQ.

LambdaDLQ

Enable AWS X-Ray [Info](#)
Enable AWS X-Ray to trace and monitor capabilities for your Lambda function

https://docs.aws.amazon.com/ja_jp/lambda/latest/dg/dlq.html

Metricsからは読み取れないこと

Eventベース/StreamベースのAWS Lambda

- At Least Once
 - AWS Lambdaは呼び出されれば最低一回は実行することを保証している。
 - 呼び出し側が呼び出さない
 - 呼び出し側が複数回呼び出す
 - この事象はAWS Lambdaでは監視/検知ができない
 - 全体の仕組みで検討する必要がある

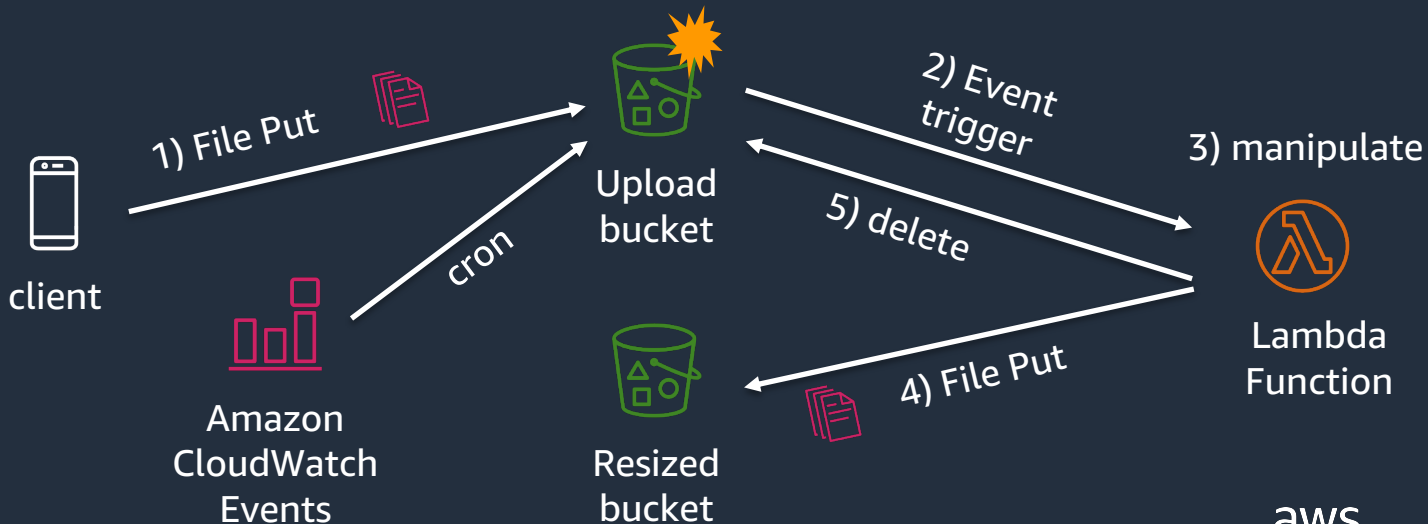
EventベースのAWS Lambda

- S3がLambdaを呼び出さない

https://docs.aws.amazon.com/ja_jp/lambda/latest/dg/with-s3-example-deployment-pkg.html

対処例

- Event発火用バケットと処理済みバケットを作成することで、Event発火漏れを検知する。



Event 発火 漏れ 対策 例

```
import boto3
s3_client = boto3.client('s3')

def lambda_handler(event, context):
    for record in event['Records']:
        bucket = record['s3']['bucket']['name']
        key = record['s3']['object']['key']
        download_path = '/tmp/{}'.format(uuid.uuid4(), key)
        s3_client.download_file(bucket, key, download_path)

        // do something to the image. (ex. thumbnail)

        s3_client.upload_file(download_path, 'upload-lambda-resized', key)
        s3_client.delete_object(Bucket=bucket, Key=key)

    return
```

Eventベース/StreamベースのAWS Lambda

- 複数回呼び出す

対処例

- Lambda Codeでの冪等性の確保
 - DynamoDB ConditionExpression パターン
 1. 入力イベントの固有属性の値を取得。(購入IDなど)
 2. 固有値がTABLEに存在するかどうかを確認します。
 - 存在する場合、(処理をSKIPし) Lambdaを正常に終了します
 - 存在しない場合、固有値をPUTして通常の実行を継続します。
 3. 関数の動作が正常に終了した場合、TABLEに固有値が含まれます。
 4. 関数の動作が異常に終了した場合、TABLEから固有値を消します。

<https://aws.amazon.com/jp/premiumsupport/knowledge-center/lambda-function-idempotent/>

複数回呼び出し対応のためのロック例

```
def lock(id):  
    try:  
        dynamodb.put_item(  
            TableName=table_name, Item={"id": {"S": id}},  
            ConditionExpression='attribute_not_exists(id)'  
        )  
        return True  
    except ClientError as e:  
        if e.response['Error']['Code'] == 'ConditionalCheckFailedException':  
            return False  
        else:  
            raise
```

AWS LambdaのMetrics監視のまとめ

Throttlesを観測した場合

- どのAWS Lambdaが大量に使っているかの特定
 - CloudWatchのアカウント単位と個別Lambda単位の使用状況を確認
 - アカウント単位で飽和している => Limit Increaseの申請
 - Lambda単位で飽和している => Reserve Concurrency設定変更

Errorsが増加している場合

- IAMでの権限設定漏れ
 - IAM Roleを他の関数と共有している場合、設定変更が行われたことを疑ってみる
- Queue/Streamは、batch sizeと全体処理時間とtimeout(min/sec)を確認
 - 単位処理時間 x batch size = 全体処理時間 > timeout
 - これも処理が進まないパターンになる

AWS LambdaのMetrics監視のまとめ

Dead Letter Queueが増え続ける

- 後段処理、連携システムのダウンなど、システム系としての障害の確認などを実施

リトライ処理の考慮

- 非同期型であれば1回の実行、2回のリトライが行われ、すべて失敗した場合に、（設定していれば）Dead Letter Queueへ送られる



Amazon API Gateway Metrics

Amazon API Gateway の Log・Traceの有効化

dev ステージエディター ステージの削除 タグの設定

URL の呼び出し: https://[redacted]execute-api.ap-northeast-1.amazonaws.com/dev

設定 **ログ/トレース** ステージ変数 SDK の生成 エクスポート デプロイ履歴 ドキュメント履歴 Canary

ステージのログギングおよびトレース設定を指定します。

CloudWatch 設定

CloudWatch ログを有効化 ⓘ

ログレベル

リクエスト/レスポンスをすべてログ

詳細 CloudWatch メトリクスを有効化 ⓘ

カスタムアクセスのログ記録

アクセスログの有効化

CloudWatch グループ ⓘ

ログの形式

入力の例:

[ログ変数のリスト](#)

X-Ray トレース [詳細はこちら](#)

X-Ray トレースの有効化 ⓘ [X-Ray サンプルングルールの設定](#)

Amazon API Gateway の Log・Traceの有効化

dev ステージエディター ステージの削除 タグの設定

URL の呼び出し: https://[redacted]execute-api.ap-northeast-1.amazonaws.com/dev

設定 **ログ/トレース** ステージ変数 SDK の生成 エクスポート デプロイ履歴 ドキュメント履歴 Canary

ステージのログギングおよびトレース設定を指定します。

CloudWatch 設定

CloudWatch ログを有効化 ⓘ

ログレベル

リクエスト/レスポンスをすべてログ

詳細 CloudWatch メトリクスを有効化 ⓘ

カスタムアクセスのログ記録

アクセスログの有効化

CloudWatch グループ ⓘ

ログの形式

入力の例:

[ログ変数のリスト](#)

X-Ray トレース [詳細はこちら](#)

X-Ray トレースの有効化 ⓘ [X-Ray サンプルングルールの設定](#)

Amazon API Gateway の Log・Traceの有効化

dev ステージエディター ステージの削除 タグの設定

URL の呼び出し: https://[redacted]execute-api.ap-northeast-1.amazonaws.com/dev

設定 **ログ/トレース** ステージ変数 SDK の生成 エクスポート デプロイ履歴 ドキュメント履歴 Canary

ステージのログギングおよびトレース設定を指定します。

CloudWatch 設定

CloudWatch ログを有効化 ⓘ

ログレベル

リクエスト/レスポンスをすべてログ

詳細 CloudWatch メトリクスを有効化 ⓘ

カスタムアクセスのログ記録

アクセスログの有効化

CloudWatch グループ ⓘ

ログの形式

入力の例:

[ログ変数のリスト](#)

X-Ray トレース [詳細はこちら](#)

X-Ray トレースの有効化 ⓘ [X-Ray サンプルングルールの設定](#)

Amazon API Gateway の Log ・ Traceの有効化

dev ステージエディター ステージの削除 タグの設定

URL の呼び出し: [https://\[redacted\].execute-api.ap-northeast-1.amazonaws.com/dev](https://[redacted].execute-api.ap-northeast-1.amazonaws.com/dev)

設定 **ログ/トレース** ステージ変数 SDK の生成 エクスポート デプロイ履歴 ドキュメント履歴 Canary

ステージのログギングおよびトレース設定を指定します。

CloudWatch 設定

CloudWatch ログを有効化 ⓘ

ログレベル

リクエスト/レスポンスをすべてログ

詳細 CloudWatch メトリクスを有効化 ⓘ

カスタムアクセスのログ記録

アクセスログの有効化

CloudWatch グループ ⓘ

ログの形式

入力の例:

[ログ変数のリスト](#)

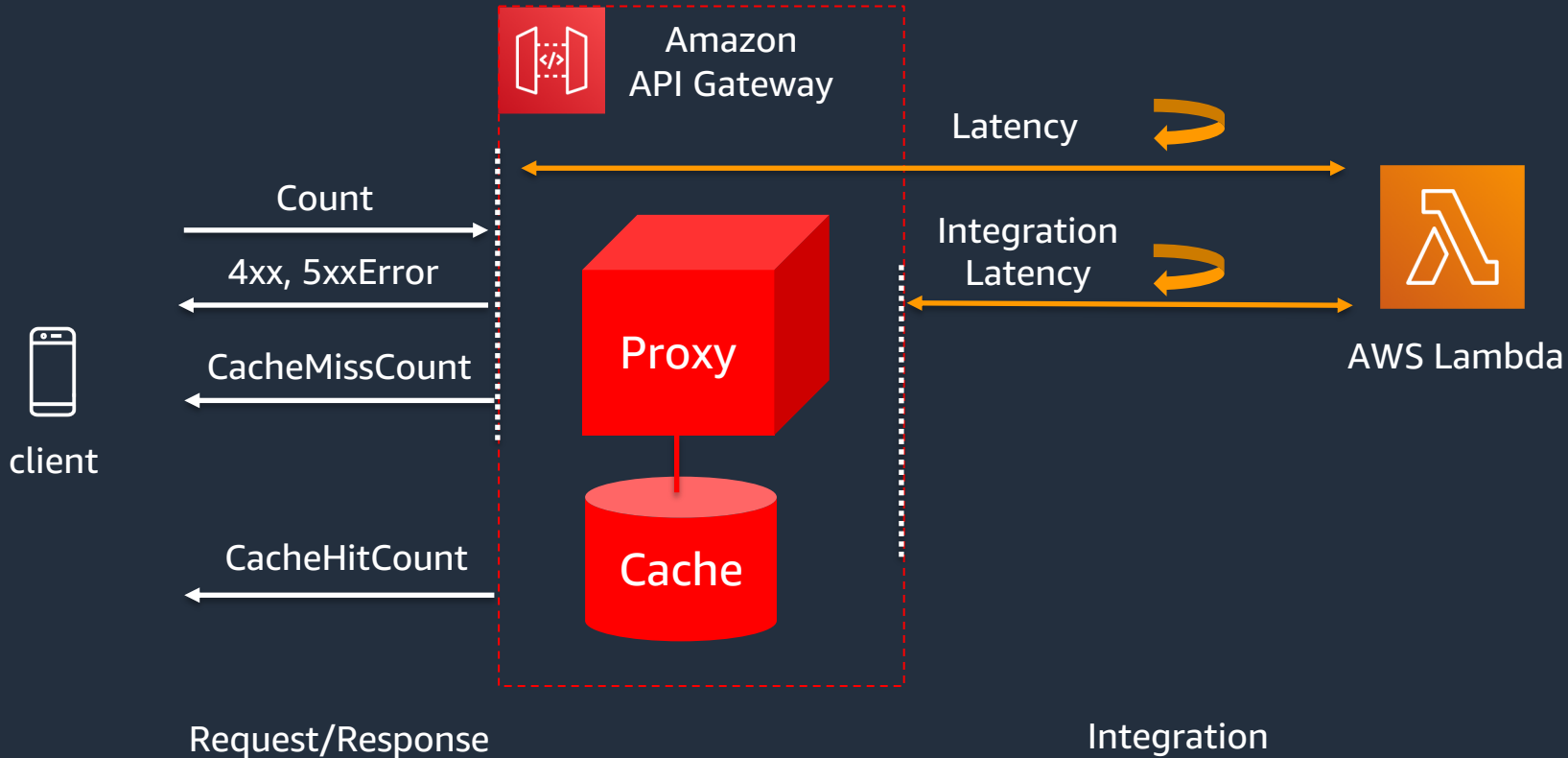
X-Ray トレース [詳細はこちら](#)

X-Ray トレースの有効化 ⓘ [X-Ray サンプルングルールの設定](#)

Amazon API Gateway の Metrics

項目	単位	意味
4xxError	count	Sumは4xxエラーの合計数、Averageは4xxエラー率
5xxError	count	Sumは5xxエラーの合計数、Averageは5xxエラー率
CacheHitCount	count	APIキャッシュからレスポンスした、 Sumはキャッシュヒットした数、Averageはキャッシュヒット率
CacheMissCount	count	キャッシュを有効にしているが、バックエンドから応答した数 Sumはキャッシュミスヒット数、Averageはキャッシュミス率
Count	count	APIのリクエスト数
IntegrationLatency	ms	API Gatewayが、バックエンドへリクエストして、バックエンドからレスポンスが返却されるまでの時間
Latency	ms	API Gatewayがクライアントからリクエストを受け取り、クライアントに返却するまでの時間。API Gatewayのオーバヘッドも含まれる

Amazon API Gateway Metrics



API GatewayのCache設定

The screenshot displays the AWS API Gateway console interface. On the left, a navigation sidebar lists various API components, with 'Stages' highlighted. The main area shows the 'prod Stage Editor' for a stage named 'prod'. The 'Cache Settings' section is active, showing the 'Cache status' as 'AVAILABLE' and a 'Flush entire cache' button. A red box highlights the 'Enable API cache' checkbox, which is checked. Below this, a yellow warning banner states 'Enabling API cache increases cost and...'. Other settings include 'Cache capacity' set to 0.5GB, 'Encrypt cache data' unchecked, and 'Cache time-to-live (TTL)' set to 30. The 'Invoke URL' is shown as 'https://t'.

APIs

- Baseball Team
 - Resources
 - Stages**
 - Authorizers
 - Gateway Responses
 - Models
 - Resource Policy
 - Documentation
 - Dashboard
 - Settings
- DynamoDBQueryAPI
- FileOperation
- Hello World

Stages **Create**

- prod

prod Stage Editor

Invoke URL: <https://t>

Settings Logs/Tracing Stage Variables SDK Generation Export

Cache Settings

Cache status AVAILABLE **Flush entire cache**

Enable API cache

Enabling API cache increases cost and...

Cache capacity 0.5GB

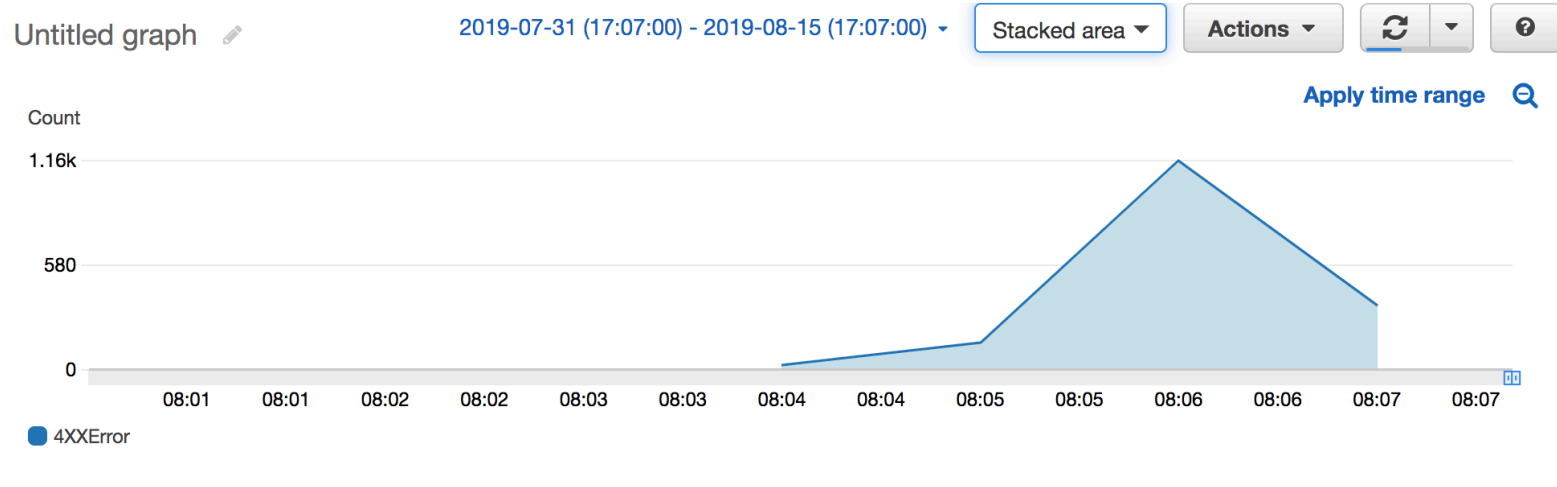
Encrypt cache data

Cache time-to-live (TTL) 30



API Gatewayのスロットリング監視








MetricsとしてThrottlesが定義されていないためHTTP 429 のレスポンスを確認する (HTTP 429 = too many requests)

- CloudWatch
- Dashboards
- Alarms
 - ALARM 0
 - INSUFFICIENT 0
 - OK 6
- Billing
- Events
- Rules
- Event Buses
- Logs
- Insights
- Metrics**
- Settings
- Favorites
- [+ Add a dashboard](#)

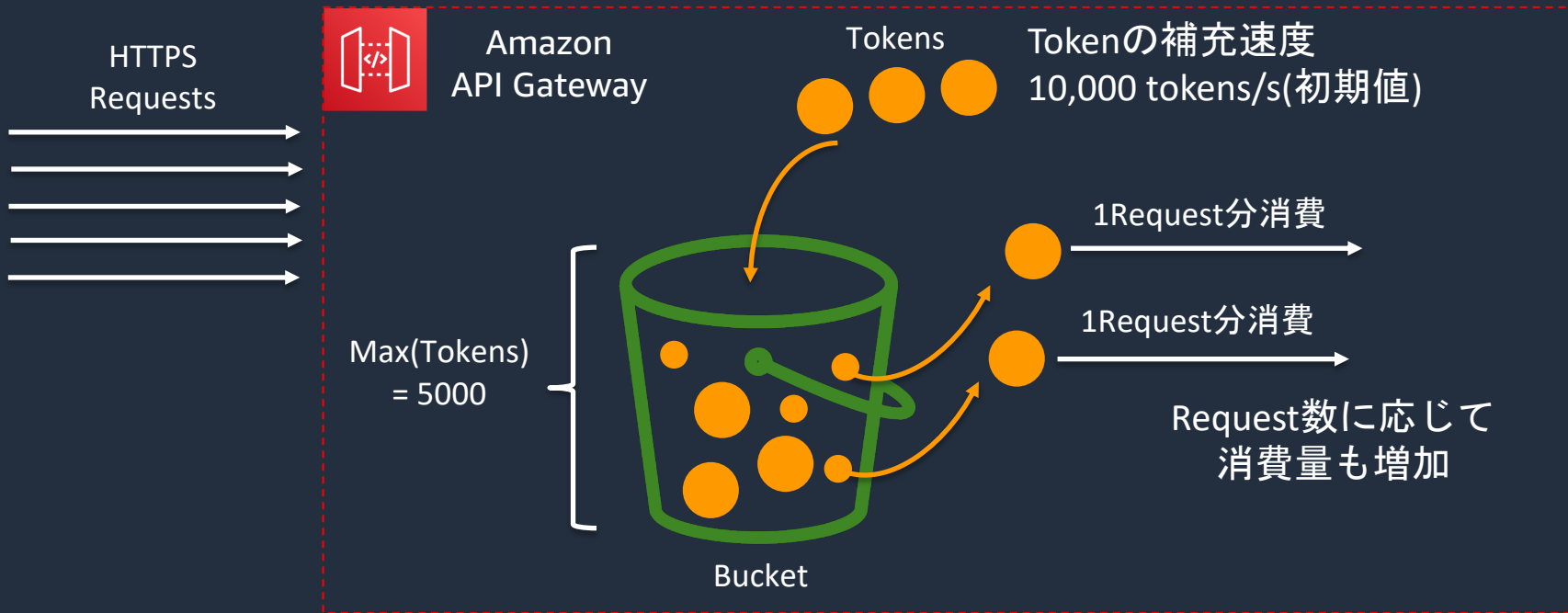


All metrics **Graphed metrics (1)** Graph options Source

[+ Add a math expression](#)  [Dynamic labels](#)  Statistic: Sum Period: 1 Second [Remove all](#)

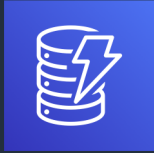
<input checked="" type="checkbox"/>	Label	Details	Statistic	Period	Y Axis	Actions
<input checked="" type="checkbox"/>	 4XXError	ApiGateway • 4XXError • ApiName: Hello ...	Sum	1 Second	 	   

API Gatewayのスロットリングを理解する



バースト = BucketのTokenの最大量 = 5000/account (初期値)

https://docs.aws.amazon.com/ja_ip/apigateway/latest/developerguide/api-gateway-request-throttling.html



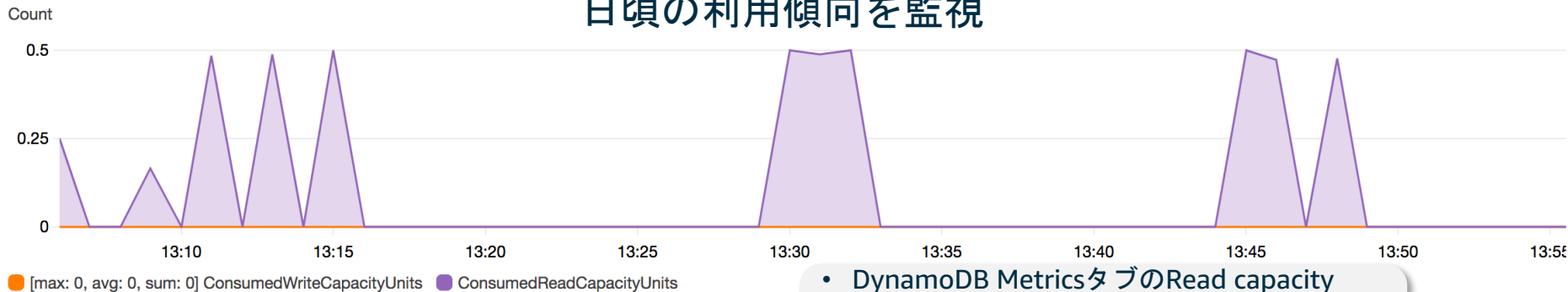
Amazon DynamoDB Metrics

Amazon DynamoDBの Metrics

項目	単位	意味
ConsumedReadCapacityUnits	Count	指定の期間内に消費された読み取り容量ユニット (RCU) の数
ConsumedWriteCapacityUnits	Count	指定の期間内に消費された書き込み容量ユニット(WCU)の数
ReadThrottleEvents	Count	テーブルまたはグローバルセカンダリインデックスのプロビジョニングされた読み取り容量ユニットを超える DynamoDB へのリクエスト
WriteThrottleEvents	Count	テーブルまたはグローバルセカンダリインデックスのプロビジョニングされた書き込み容量ユニットを超える DynamoDB へのリクエスト。
SystemErrors	Count	指定された期間中に HTTP 500 ステータスコードを生成する、DynamoDB または DynamoDB ストリームへのリクエスト。通常、HTTP 500 は内部サービスエラーを示します。

指定期間に消費された読み取り/書き込み容量ユニット監視

日頃の利用傾向を監視



- DynamoDB MetricsタブのRead capacity Unit/Secondは秒単位の表示
- CloudWatch ConsumedReadCapacityUnits は分単位の表示(CloudWatchの設定に依存)

All metrics

Graphed metrics (2/5)

Graph options

Source

+ Add a math expression ?

Dynamic labels v

Statistic: Average

1 Minute v

Remove all

<input type="checkbox"/>	<input type="checkbox"/>	Label	Details	Statistic		Y Axis	Actions
<input type="checkbox"/>	<input type="checkbox"/>	[max: \${MAX}, avg: \${AVG}, su...	DynamoDB • ConditionalCheckFailedRequests • TableName...	Average		< >	📈 🔔 📄 ✖
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[max: \${MAX}, avg: \${AVG}, su...	DynamoDB • ConsumedWriteCapacityUnits • TableName: s...	Average	1 Minute	< >	📈 🔔 📄 ✖
<input type="checkbox"/>	<input type="checkbox"/>	[max: \${MAX}, avg: \${AVG}, su...	DynamoDB • ProvisionedWriteCapacityUnits • TableName: ...	Average	1 Minute	< >	📈 🔔 📄 ✖
<input type="checkbox"/>	<input type="checkbox"/>	[max: \${MAX}, avg: \${AVG}, su...	DynamoDB • ProvisionedReadCapacityUnits • TableName: ...	Average	1 Minute	< >	📈 🔔 📄 ✖
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ConsumedReadCapacityUnits	DynamoDB • ConsumedReadCapacityUnits • TableName: s...	Average	1 Minute	< >	📈 🔔 📄 ✖

DynamoDBのスロットリング監視

Throttled read events/Throttled write eventsの監視

- テーブルにプロビジョニングされた読み取り/書き込み容量 (RCU/WCU)を指定期間で超えたリクエストで、テーブルにプロビジョニングされたスループット制限をどのリクエストが超えたかを判断できる

https://docs.aws.amazon.com/ja_jp/amazondynamodb/latest/developerguide/metrics-dimensions.html

Capacity: table

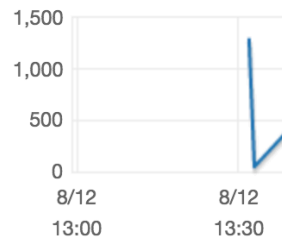
Read capacity Units/Second - 1 min avg ⓘ



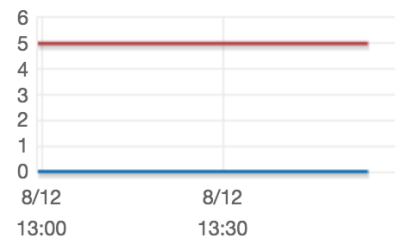
Throttled read requests Count ⓘ



Throttled read events Count ⓘ



Write capacity Units/Second - 1 min avg ⓘ



Provisioned Consumed

Get Scan Query Batch get

Provisioned Consumed

DynamoDBのシステムエラー監視

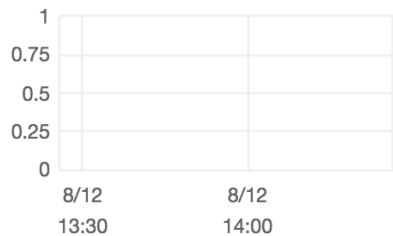
System errors read/writeを監視

- エラーが発生したかどうかを判断
 - 通常これは0であるべき値
- システムエラーの数 = HTTP Status 500 の数
 - DynamoDB / DynamoDB Streams へのエラーリクエスト数

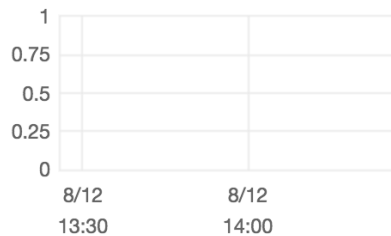
Errors

通常は0となっているはず

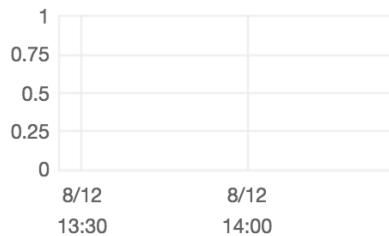
System errors read Count



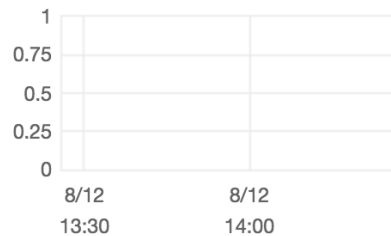
System errors write Count



User errors Count

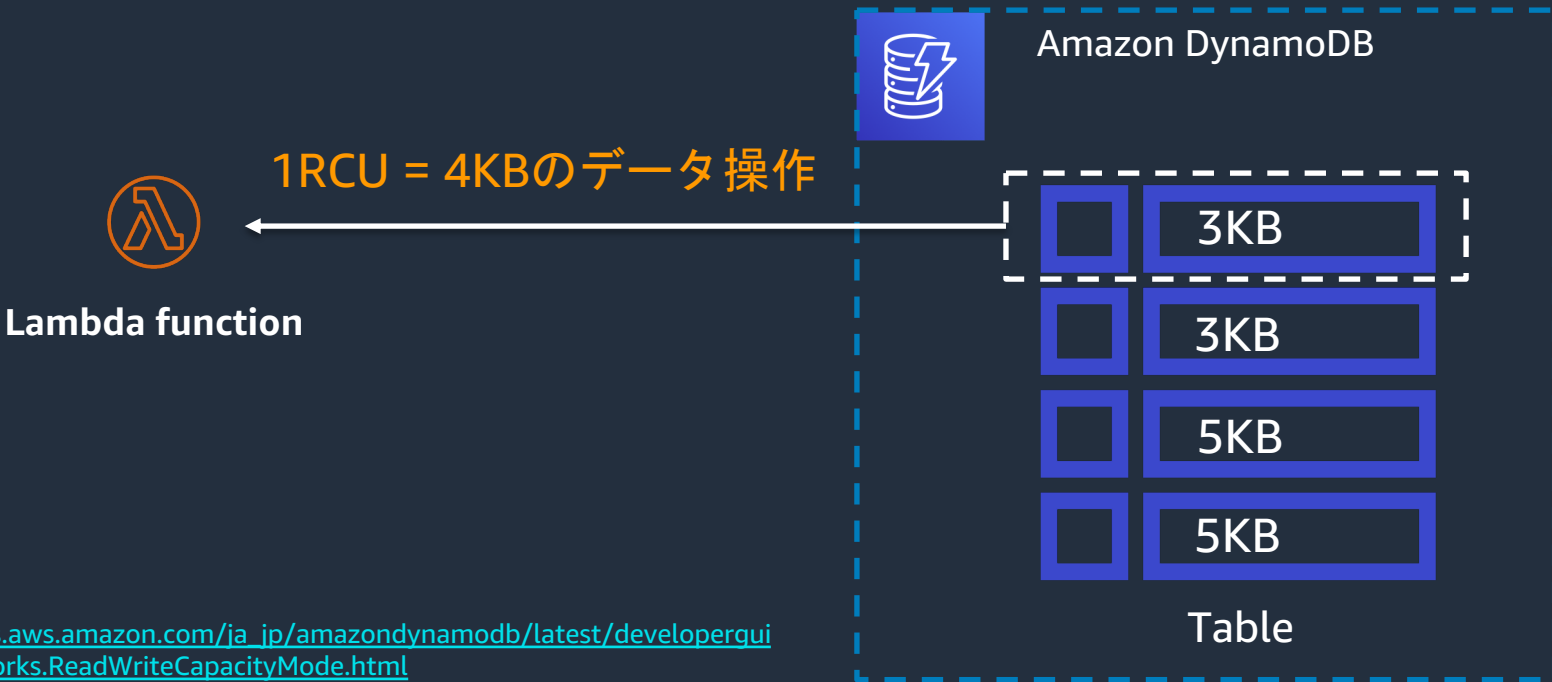


Conditional check failed Count



APIとRead Capacity Unit(CU)の消費

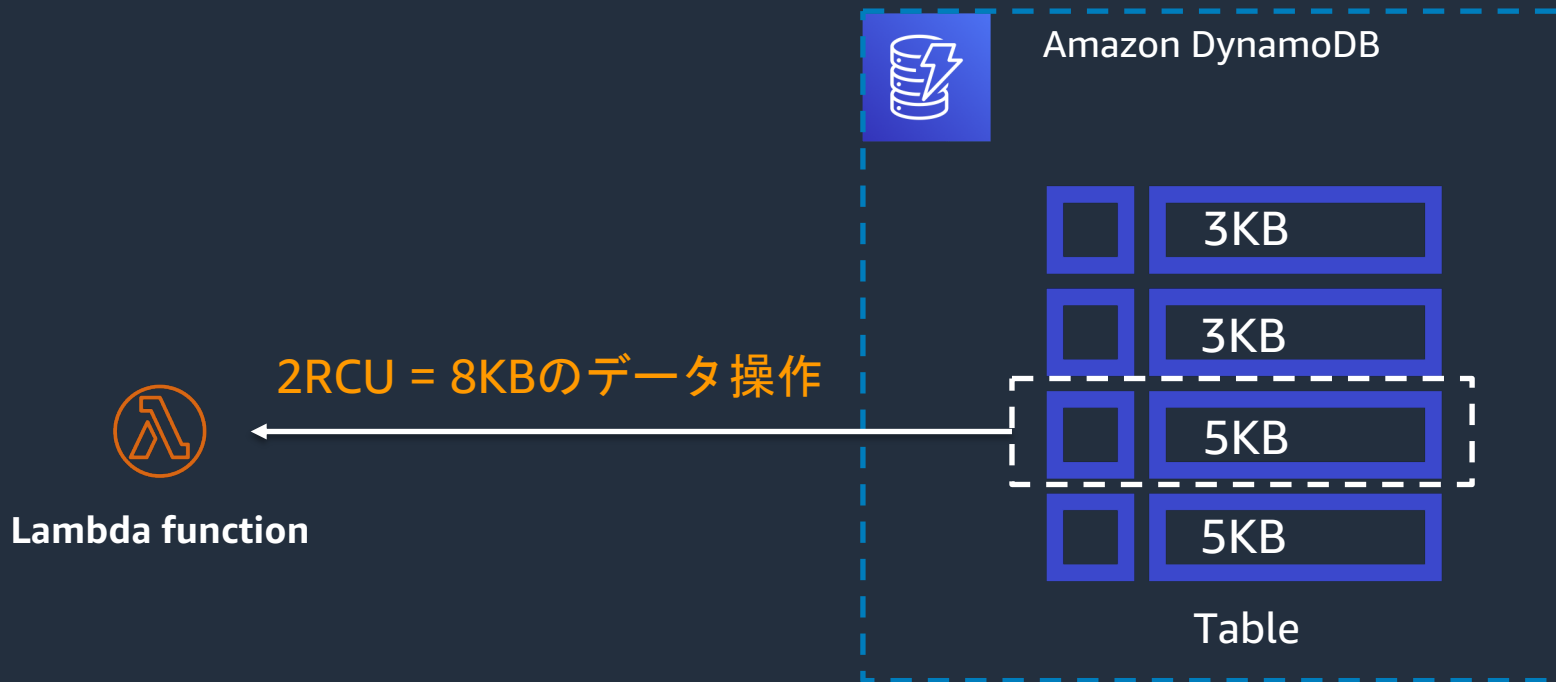
READの操作、query, scan, get_itemはリクエスト単位のデータサイズで算出



https://docs.aws.amazon.com/ja_jp/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html

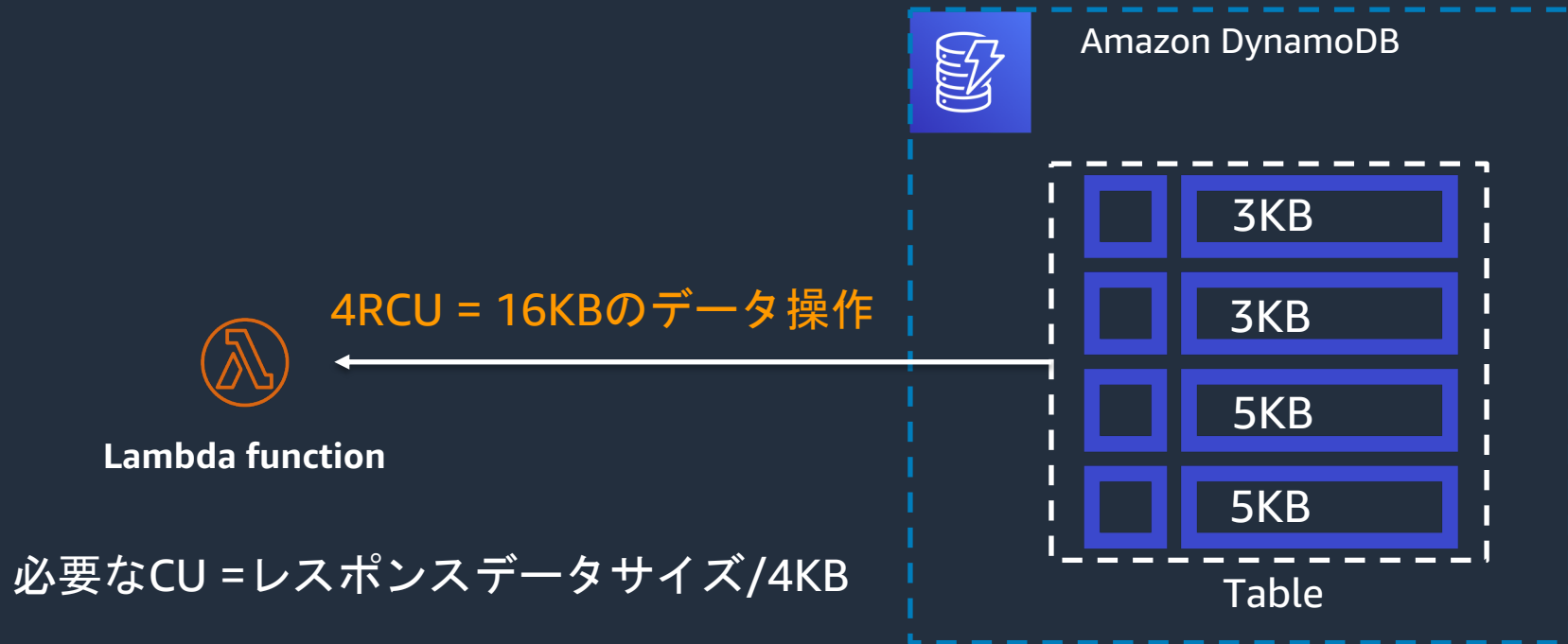
APIとRead Capacity Unit(CU)の消費

READの操作、query, scan, get_itemはリクエスト単位のデータサイズで算出

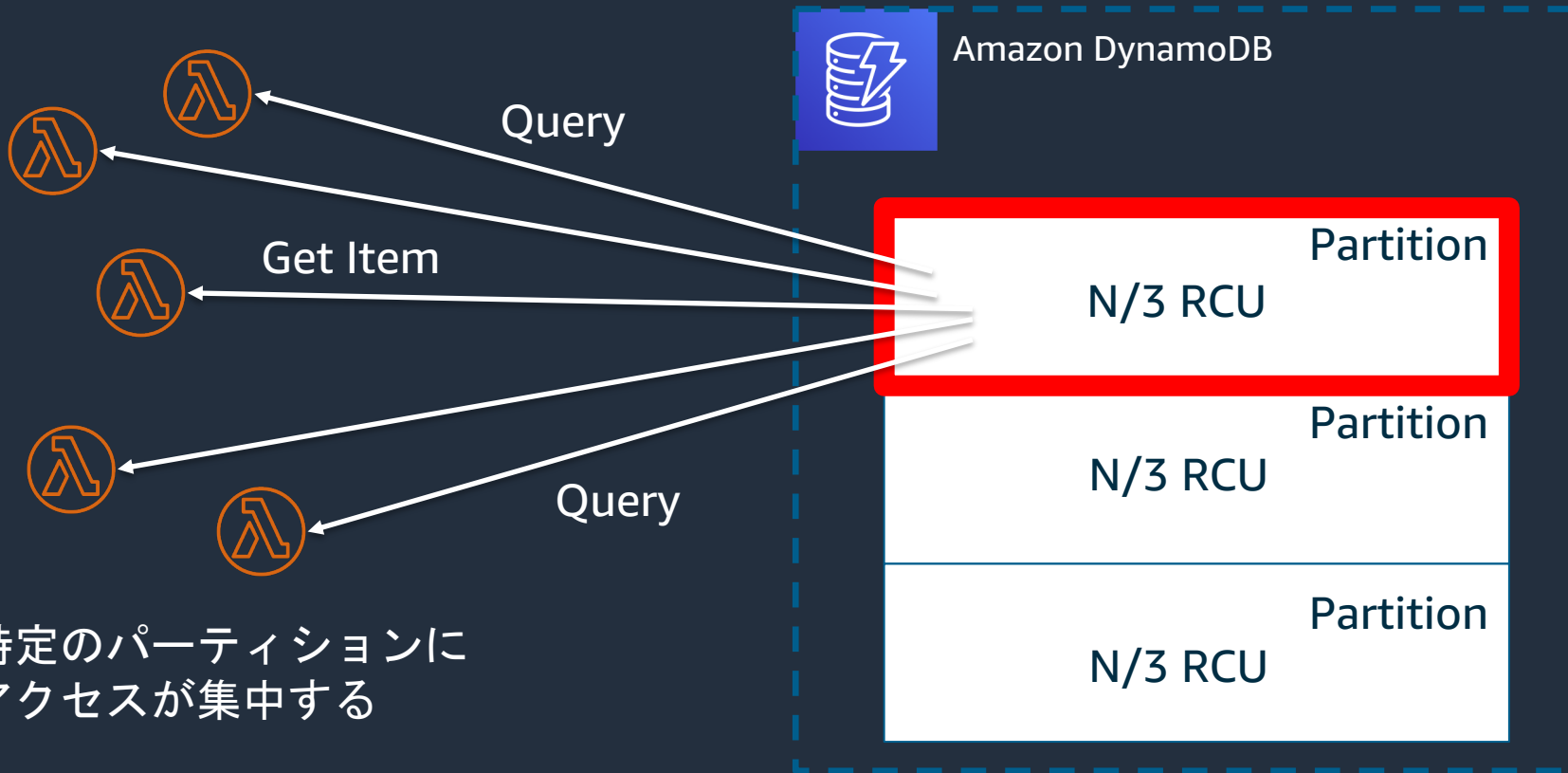


APIとRead Capacity Unit(CU)の消費

READの操作、query, scan, get_itemはリクエスト単位のデータサイズで算出



ホットパーティションとは



TTLの活用

TTLはCUを消費しない

- TTLを利用しない削除の場合は、削除対応のデータをquery/scanしてからdeleteするオペレーションとなり、瞬間的に大きなR/W CUを設定する必要がでたりなど運用上CUの管理が必要
- TTL期限が過ぎてすぐに削除されるわけではなく、48時間以内に削除となっているので、アプリケーション的な考慮は必要
 - queryでTTL切れのアイテムが取れる可能性がある
- TimeToLiveDeletedItemCountでTTLにより削除されたアイテム数をカウントすることも可能

https://docs.aws.amazon.com/ja_jp/amazondynamodb/latest/developerguide/TTL.html

オートスケーリングを利用しているのにCU不足

オートスケールを利用しても、スパイクアクセスの完全対応が難しい

- 時刻によるスパイク
- イベントによるスパイク

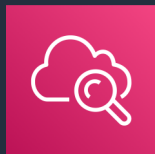
APIコントロール（もしくはオートスケールと併用)が良い場合もある

オートスケールを設定したテーブルも初期設定は RCU/WCUともに5が初期値となるので、この点も注意

https://docs.aws.amazon.com/ja_jp/amazondynamodb/latest/developerguide/AutoScaling.CLI.html#AutoScaling.CLI.BeforeYouBegin



Log運用と可視化



Amazon CloudWatch でのログ管理

Amazon CloudWatch Logsの使い方を知る

CloudWatchは簡単にLog出力でき、とても便利です。

CloudWatch

Dashboards

Alarms

ALARM

INSUFFICIENT

OK

Billing

Events

Rules

Event Buses

Logs

Insights

Metrics

Settings

CloudWatch > Log Groups > St...

Search Log Group

Create Log Stream

Delete Log Stream

Filter: Log Stream Name Prefix

Log Streams

Last Event

<input type="checkbox"/>	2019/08/12/[\$LATEST]a9ffa3f08823042129b1f0e97543684af	2019-08-	ist-1:4
<input type="checkbox"/>	2019/08/12/[\$LATEST]6...d1a2044fca7613b03b4129499	2019-08-	ist-1:4
<input type="checkbox"/>	2019/08/12/[\$LATEST]4...350c849b6a802e9cd74cc7af1	2019-08-12 10:...	arn:aws:logs:ap-northeast-1:4
<input type="checkbox"/>	2019/08/12/[\$LATEST]2...adc154cc1af65e565942c5f33	2019-08-12 10:...	arn:aws:logs:ap-northeast-1:4
<input type="checkbox"/>	2019/08/12/[\$LATEST]4...3b4ef4ea3ac251414580e2799	2019-08-12 18:13 UTC+9	arn:aws:logs:ap-northeast-1:4
<input type="checkbox"/>	2019/08/12/[\$LATEST]3...3a4c847389a782ba3ea08667a	2019-08-12 18:11 UTC+9	arn:aws:logs:ap-northeast-1:4
<input type="checkbox"/>	2019/08/12/[\$LATEST]6...23f34c80a77bee1df2a48721	2019-08-12 18:10 UTC+9	arn:aws:logs:ap-northeast-1:4
<input type="checkbox"/>	2019/08/12/[\$LATEST]222d1cb2f6604f229b1a14d3e0f168d9	2019-08-12 17:59 UTC+9	arn:aws:logs:ap-northeast-1:4

ログの内容を横断的に確認する機能が欲しい

Lambdaのコンテナ起動単位でロググループが分かれている。

フィルターパターンの利用例

- ログが正規化されていなくても一致する文字列の検索も可能。
- アプリケーションが出力するログフォーマットと利用ルールを明確化することでmetric filterも作成できる。
- printで出力するのではなく、logger関数などを利用してログ出力することがおすすめ

Time (UTC +00:00)	Message
2018-10-30	
11:13:24	Loading function
11:13:24	Get record count:
11:13:24	1
11:13:24	Decoded payload: {"Tmp": 16, "TimeStamp": 1540898004111, "ID": "test1"}
11:13:24	END RequestId: e13b5b1b-70e6-459b-a85e-b13ec98ec010
11:13:24	REPORT RequestId: e13b5b1b-70e6-459b-a85e-b13ec98ec010 Duration: 0.48 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory
11:13:26	START RequestId: 370c60b5-2458-4ac9-8844-fb8c68aabf34 Version: \$LATEST
11:13:26	{u'Records': [{u'eventVersion': u'1.0', u'eventID': u'shardId-000000000002:495896627707678205167181931411716199584615236037320376
11:13:26	Loading function
11:13:26	Get record count:
11:13:26	6
11:13:26	Decoded payload: {"Tmp": 15, "TimeStamp": 1540898004419, "ID": "test1"}
11:13:26	Decoded payload: {"Tmp": 22, "TimeStamp": 1540898004728, "ID": "test1"}
11:13:26	Decoded payload: {"Tmp": 25, "TimeStamp": 1540898005038, "ID": "test1"}
11:13:26	Decoded payload: {"Tmp": 24, "TimeStamp": 1540898005349, "ID": "test1"}
11:13:26	Decoded payload: {"Tmp": 25, "TimeStamp": 1540898005657, "ID": "test1"}
11:13:26	Decoded payload: {"Tmp": 20, "TimeStamp": 1540898005969, "ID": "test1"}
11:13:26	END RequestId: 370c60b5-2458-4ac9-8844-fb8c68aabf34
11:13:26	REPORT RequestId: 370c60b5-2458-4ac9-8844-fb8c68aabf34 Duration: 0.66 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory
11:13:26	START RequestId: 694f20f9-55fd-4a32-a99e-eb1a540573bd Version: \$LATEST
11:13:26	{u'Records': [{u'eventVersion': u'1.0', u'eventID': u'shardId-000000000002:495896627707678205167181931415705654789343513688240292
11:13:26	Loading function
11:13:26	Get record count:
11:13:26	1
11:13:26	Decoded payload: {"Tmp": 23, "TimeStamp": 1540898006586, "ID": "test1"}
11:13:26	END RequestId: 694f20f9-55fd-4a32-a99e-eb1a540573bd
11:13:26	REPORT RequestId: 694f20f9-55fd-4a32-a99e-eb1a540573bd Duration: 0.42 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory

Define Logs Metric Filter

Filter for Log Group: /aws/lambda/KinesisBatch

You can use metric filters to monitor events in a log group as they are sent to CloudWatch Logs. You can monitor and count specific terms or extract values from log events and associate the results with a metric. [Learn more about pattern syntax.](#)

Filter Pattern

[Show examples](#)

Select Log Data to Test

[Clear](#) [Test Pattern](#)

START RequestId: 52afba3f-539c-4335-9468-5c61125abfc3 Version: \$LATEST
{u'Records': [{u'eventVersion': u'1.0', u'eventID': u'shardId-000000000002:49589662770767820516718189
Loading function
Get record count:
33
Decoded payload: {"Tmp": 25, "TimeStamp": 1540883102840, "ID": "test1"}
Decoded payload: {"Tmp": 19, "TimeStamp": 1540883103150, "ID": "test1"}

Results

Found 3 matches out of 50 event(s) in the sample log.

[Show test results](#)

[Cancel](#) [Assign Metric](#)

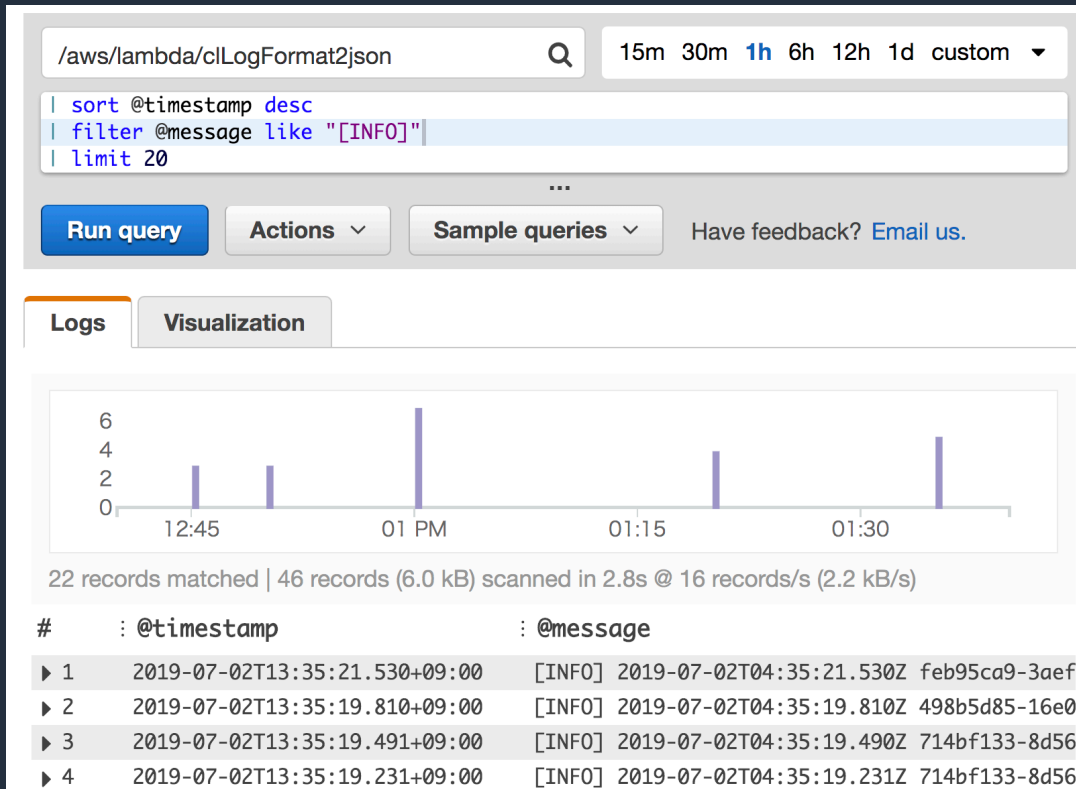
CloudWatch Logs filter pattern

filter patternの仕様

- 大文字、小文字は区別される
- 正規表現は利用できない
- 簡単なパターンマッチのみ利用可能
- JSON形式、スペース区切りは利用可能

CloudWatch Logs Insightsの活用

```
fields @timestamp, @message  
| sort @timestamp desc  
| filter @message like "[INFO]"  
| limit 20
```



The screenshot shows the AWS CloudWatch Logs Insights interface. At the top, a search bar contains the path `/aws/lambda/clLogFormat2json`. To the right, there are time range filters: 15m, 30m, 1h (selected), 6h, 12h, 1d, and custom. Below the search bar, a query editor contains the following query:

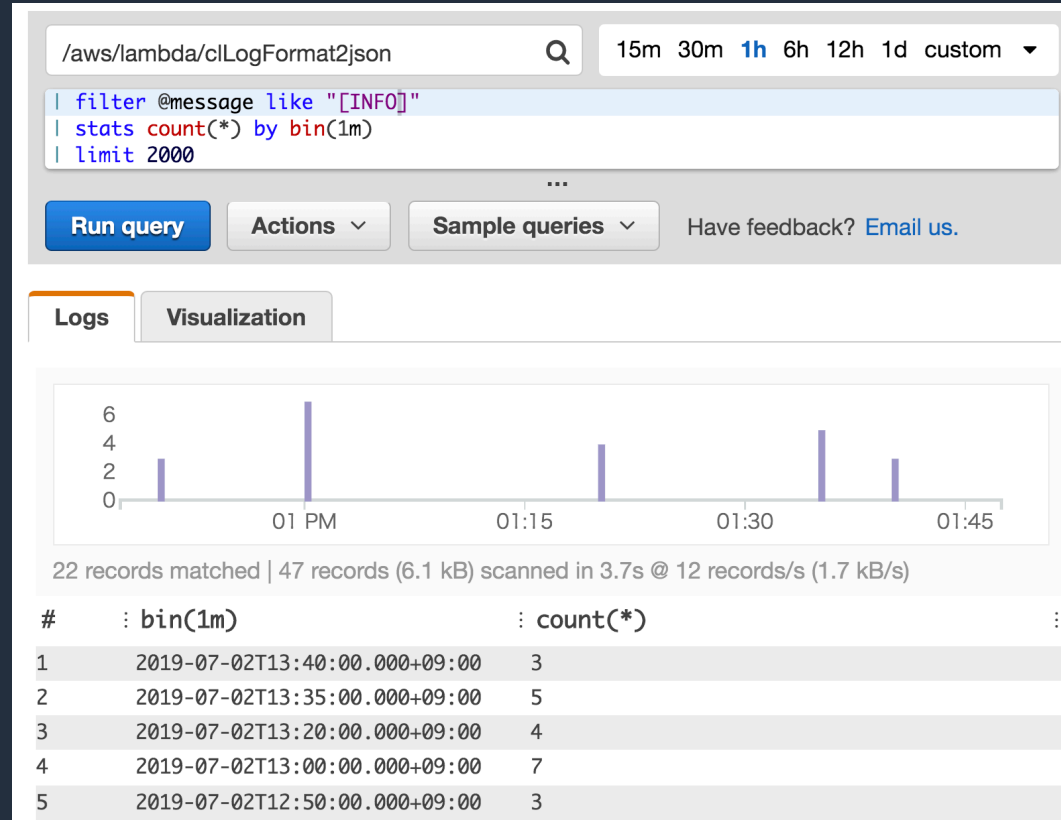
```
| sort @timestamp desc  
| filter @message like "[INFO]"  
| limit 20
```

Below the query editor, there are buttons for "Run query", "Actions", and "Sample queries", along with a link to "Have feedback? Email us." Below this, there are two tabs: "Logs" (selected) and "Visualization". The "Visualization" tab shows a bar chart with the y-axis ranging from 0 to 6 and the x-axis showing time intervals: 12:45, 01 PM, 01:15, and 01:30. The chart shows four bars with heights of approximately 3, 3, 6, and 5 respectively. Below the chart, the text reads: "22 records matched | 46 records (6.0 kB) scanned in 2.8s @ 16 records/s (2.2 kB/s)". Below this, there is a table with the following columns: #, @timestamp, @message, and a truncated @message field.

#	@timestamp	@message	@message
▶ 1	2019-07-02T13:35:21.530+09:00	[INFO]	2019-07-02T04:35:21.530Z feb95ca9-3aef
▶ 2	2019-07-02T13:35:19.810+09:00	[INFO]	2019-07-02T04:35:19.810Z 498b5d85-16e0
▶ 3	2019-07-02T13:35:19.491+09:00	[INFO]	2019-07-02T04:35:19.490Z 714bf133-8d56
▶ 4	2019-07-02T13:35:19.231+09:00	[INFO]	2019-07-02T04:35:19.231Z 714bf133-8d56

CloudWatch Logs Insightsの活用

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like "[INFO]"
| stats count(*) by bin(1m)
| limit 2000
```



ログ保管期間

- Lambdaのログ保管期間はデフォルトが無期限
- 必要な期間に圧縮することを忘れがち

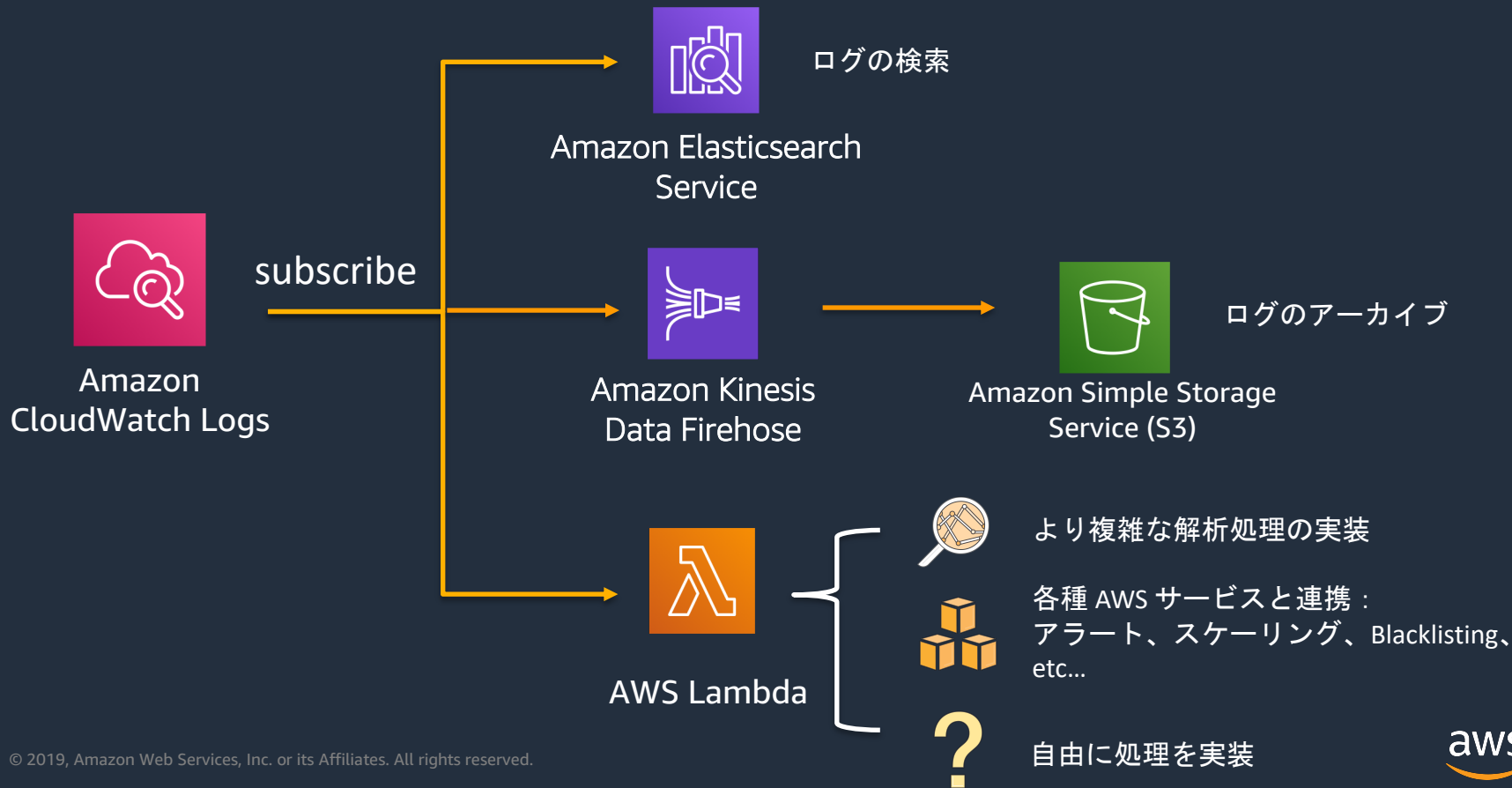
CloudWatch > Log Groups

Create Metric Filter Actions

Filter: Log Group Name Prefix

Log Groups	Insights	Expire Events After	Metric Filters
<input type="radio"/> /aws/apigateway/welcome	Explore	Never Expire	0 filters
<input type="radio"/> /aws/lambda/ConditionFunc	Explore	Never Expire	0 filters
<input type="radio"/> /aws/lambda/CreateCognitoUser	Explore	Never Expire	0 filters
<input type="radio"/> /aws/lambda/CreateCognitoUserjs	Explore	Never Expire	0 filters
<input type="radio"/> /aws/lambda/DLQfunc	Explore	Never Expire	0 filters
<input type="radio"/> /aws/lambda/DynamoDBreadfunc	Explore	Never Expire	0 filters
<input type="radio"/> /aws/lambda/DynamoDBtoLambda	Explore	Never Expire	0 filters
<input type="radio"/> /aws/lambda/JWTFumction	Explore	Never Expire	0 filters
<input type="radio"/> /aws/lambda/KinesisDataEgestFunc	Explore	Never Expire	0 filters
<input type="radio"/> /aws/lambda/KinesisDataIngestFunc	Explore	Never Expire	0 filters
<input type="radio"/> /aws/lambda/KinesisDataIngestLoopFunc	Explore	Never Expire	0 filters

他サービスに連携してログの活用





Amazon Elasticsearch Service との連携



- Dashboard
- My domains
 - logsearch
- Reserved instances

logsearch

- Configure cluster
- Modify access policy
- Manage tags
- Delete domain
- Upgrade domain

- Overview
- Cluster health
- Instance health
- Indices
- Logs
- Upgrade history

Domain status	Active
Elasticsearch version	6.7
Endpoint	https://search-logsearch-██████████-vanbgdpy.ap-northeast-1.es.amazonaws.com
Domain ARN	arn:aws:es:ap-north-██████████:domain/logsearch
Kibana	https://search-logsearch-██████████-vanbgdpy.ap-northeast-1.es.amazonaws.com/_plugin/kibana/
Availability zones	1
Instance type	t2.small.elasticsearch
Number of instances	1
Storage type	EBS
EBS volume type	General Purpose (SSD)
EBS volume size	10 GB
Encryption at rest	Disabled

CloudWatch

Dashboards

Alarms

ALARM

INSUFFICIENT

OK

Billing

Events

Rules

Event Buses

Logs

Insights

Metrics

Settings

Favorites

[+ Add a dashboard](#)

0

0

4

CloudWatch > Log Groups

Amazon CloudWatch Logs



Create Metric Filter

Actions ▾

Filter: hello

Log Groups

helloworld_loggroup

Create log group

Delete log group

Export

Export data to Amazon S3

View all exports to Amazon S3

Subscriptions

Stream to AWS Lambda

Stream to Amazon Elasticsearch Service

Remove Subscription Filter



Log Groups 1-1

Metric Filters

Subscriptions

ers

None

Step 1: Choose Destination

Step 2: Configure Log Format and Filters

Step 3: Review

Step 4: Confirmation

Start Streaming helloworld_loggroup to Amazon Elasticsearch Service

You are about to start streaming data from your "helloworld_loggroup" log group to an Amazon Elasticsearch Service cluster. Any new log data sent to this log group will be sent to the cluster you choose.



Select account This Account

Another Account

Amazon ES cluster

logsearch



Lambda Function

CloudWatch Logs uses Lambda to deliver log data to Amazon ES. You must specify an IAM role that grants Lambda permission to make calls to Amazon ES. You can choose an existing role or create an IAM role that automatically has the required permissions. To deliver log data to another account, you must specify the Elasticsearch Domain ARN and Elasticsearch Endpoint of other account and ensure permissions are granted to be able to publish to that ARN.

Lambda IAM Execution Role *

LambdaElasticServiceRole



Cancel

Previous

Next

Start Streaming

[Step 1: Choose Destination](#)

[Step 2: Configure Log Format and Filters](#)

[Step 3: Review](#)

Step 4: Confirmation

Success

You have started streaming log data from your "helloworld_loggroup" log group to your "logsearch" Amazon Elasticsearch Service cluster.

If you have set an access policy for your Amazon Elasticsearch Service cluster that allows access from your browser, you can start interacting with your log data using the following links:

- [Kibana 3](#)
- [Kibana 4](#)
- [Elasticsearch API](#)

The following Lambda function transforms your incoming CloudWatch Logs data to index documents and submits them to your Elasticsearch cluster. You can modify this function to add custom data transformations (for example, converting IP addresses to geo-locations):

- [LogsToElasticsearch_logsearch](#)

If you are streaming any of the following AWS log data sources you can import sample Kibana 3 dashboards by downloading the relevant dashboard file below. [Learn how](#) to import dashboard files in Kibana 3.

- [Amazon VPC Flow Logs](#)
- [AWS Lambda](#)
- [AWS CloudTrail](#)

You can cancel the streaming of your log data to Amazon Elasticsearch Service any time by removing the subscription filter associated with your log group.


[Cancel](#)

[Previous](#)

[Next](#)

[Start Streaming](#)



 Kibana[Index Patterns](#)

Saved Objects

Advanced Settings

Create index ...

No default index pattern. You must select or create one to continue.

cwl-*

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

 Include system indices

Step 1 of 2: Define index pattern


Index pattern

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

[> Next step](#)

✓ **Success!** Your index pattern matches **1 index**.

Rows per page: 10

 Kibana[Index Patterns](#)

Saved Objects

Advanced Settings

Create index ...

No default index pattern. You must select or create one to continue.

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

 Include system indices

Step 2 of 2: Configure settings

You've defined `cwl-*` as your index pattern. Now you can specify some settings before we create it.

Time Filter field name

Refresh



The Time Filter will use this field to filter your data by time.

You can choose not to have a time field, but you will not be able to narrow down your data by a time range.

[> Show advanced options](#)[< Back](#)

Create index pattern

Kibana

[Index Patterns](#)

Saved Objects

Advanced Settings

Create index ...

★ cwl-*

★ cwl-*



Time Filter field name: @timestamp

This page lists every field in the **cwl-*** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch [Mapping API](#)

Fields (29)	Scripted fields (0)	Source filters (0)
-------------	---------------------	--------------------

Filter

All field types ▾

Name	Type	Format	Search...	Aggre...	Exclu...
@id	string		●		
@id.keyword	string		●	●	
@log_group	string		●		
@log_group.keyword	string		●	●	
@log_stream	string		●		

Kibana interface showing a search query and a histogram visualization.

Search Bar: >_ 200

Visualize Panel: cw1-*

Selected fields: ? _source

Available fields: t @id, t @log_group

Top 5 values in 50 / 50 records: helloworld_loggroup 100.0%, t @log_stream, t @message

Visualization: Histogram of @timestamp per millisecond. The x-axis ranges from 21:45:20.800 to 21:45:20.890. The y-axis is Count (0 to 25). A red box highlights the peak at approximately 21:45:20.840.

Table:

Time	_source
▶ August 11th 2019, 21:45:20.841	@message: 13.231.21.130 -- [11/Aug/2019:12:45:20 +0000] "GET /helloworld HTTP/1.1" 200 20 e129c3fb-bc35-11e9-a4a8-71d913e77613 date: 11/Aug/2019:12:45:20 +0000 request: GET /helloworld HTTP/1.1 ident: - bytes: 20 e129c3fb-bc35-11e9-a4a8-71d913e77613 host: 13.231.21.130 authuser: - status: 200
▶ August 11th 2019, 21:45:20.843	@message: 13.231.21.130 -- [11/Aug/2019:12:45:20 +0000] "GET /helloworld HTTP/1.1" 200 20 e12a121e-bc35-11e9-a4a8-71d913e77613 date: 11/Aug/2019:12:45:20 +0000 request: GET /helloworld HTTP/1.1 ident: - bytes: 20 e12a121e-bc35-11e9-a4a8-71d913e77613 host: 13.231.21.130 authuser: - status: 200

>_ e12a121e

Options

Refresh

- Discover
- Visualize
- Dashboard
- Timelion
- Alerting
- Dev Tools
- Management

Add a filter +

cwl-*

Selected fields

? _source

Available fields

t @id

t @log_group

Top 5 values in 1 / 1 records

helloworld_loggroup 100.0%

t @log_stream

t @message

Top 5 values in 1 / 1 records

13.231.21.130 -- [11... 100.0%

t @owner

@timestamp

August 11th 2019, 21:45:20.817 - August 11th 2019, 21:45:20.897 — Auto



Time	_source
▶ August 11th 2019, 21:45:20.843	bytes: 20 e12a121e-bc35-11e9-a4a8-71d913e77613 @message: 13.231.21.130 -- [11/Aug/2019:12:45:20 +0000] "GET /helloworld HTTP/1.1" 200 20 e12a121e-bc35-11e9-a4a8-71d913e77613 date: 11/Aug/2019:12:45:20 +0000 request: GET /helloworld HTTP/1.1 ident: - host: 13.231.21.130 authuser: - status: 200

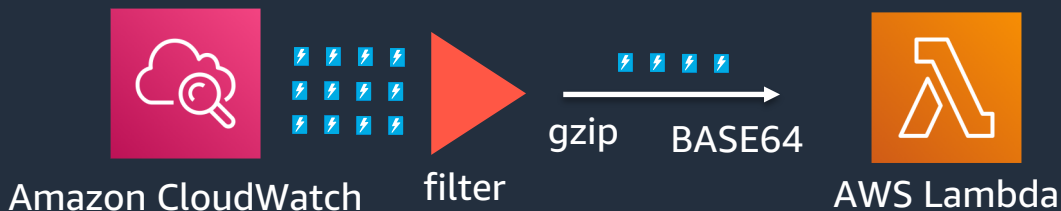


AWS Lambdaとの連携

CloudWatch LogStreamをLambdaのEvent引数で受け取る

```
{  
  "awslogs": {  
    "data": "H4sIAAAAAAAAAAI1SwYrbMBD9FaNr18mMJVmSb4  
            Em6WFLC/FtNwTZmnUMTpzacrdLyL934qXsaaESQ  
            (snip)  
            AsHJKorPaoEWGu77ZDv104ZsjdV3/2g9dODDazOh  
            wY81M8TuP8be6inGmc3lVB3G7721+9vgAVcwIAAA=="  
  }  
}
```

} Event引数



- LogStreamは、gzip圧縮され、BASE64エンコードされている
- 全streamを流すのではなく、フィルタする

CloudWatch

Dashboards

Alarms

ALARM

INSUFFICIENT

OK

Billing

Events

Rules

Event Buses

Logs

Insights

Metrics

Settings

Favorites

[+ Add a dashboard](#)

CloudWatch > Log Groups

Amazon CloudWatch Logs



Create Metric Filter

Actions ▾

Filter: hello

Log Groups	Insights	Expire Events After	Metric Filters	Subscriptions
<input type="radio"/> helloworld_loggroup	Explore	Never Expire	0 filters	None

Step 1: Define Pattern

Step 2: Assign Metric

Define Logs Metric Filter

Amazon CloudWatch Logs



Filter for Log Group: helloworld_loggroup

You can use metric filters to monitor events in a log group as they are sent to CloudWatch Logs. You can monitor and count specific terms or extract values from log events and associate the results with a metric. [Learn more about pattern syntax.](#)

Filter Pattern



[Show examples](#)

Select Log Data to Test

[Clear](#)

```
13.231.21.130 -- [11/Aug/2019:12:45:20 +0000] "GET /helloworld HTTP/1.1" 200 20 e129eaec-bc35-11e9-b37c
13.231.21.130 -- [11/Aug/2019:12:45:21 +0000] "GET /helloworld HTTP/1.1" 200 20 e150fb30-bc35-11e9-b37c
13.231.21.130 -- [11/Aug/2019:12:45:21 +0000] "GET /helloworld HTTP/1.1" 200 20 e158754e-bc35-11e9-b37c
13.231.21.130 -- [11/Aug/2019:12:45:21 +0000] "GET /helloworld HTTP/1.1" 200 20 e175e890-bc35-11e9-b37c
13.231.21.130 -- [11/Aug/2019:12:53:58 +0000] "GET /helloworld HTTP/1.1" 200 20 157e1c37-bc37-11e9-b37c
13.231.21.130 -- [11/Aug/2019:12:53:58 +0000] "GET /helloworld HTTP/1.1" 200 20 15859552-bc37-11e9-b37c
13.231.21.130 -- [11/Aug/2019:12:53:58 +0000] "GET /helloworld HTTP/1.1" 200 20 159091e4-bc37-11e9-b37c
```



Results

Found **8** matches out of 8 event(s) in the sample log.

[Show test results](#)

[Cancel](#)

Create Metric Filter and Assign a Metric



Filter for Log Group: helloworld_loggroup

Log events that match the pattern you define are recorded to the metric that you specify. You can graph the metric and set alarms to notify you.

Filter Name:



Filter Pattern:

Metric Details

Metric Namespace:



[Create new namespace](#)

Metric Name:



[Show advanced metric settings](#)

Cancel

Previous

Create Filter

Create function [Info](#)


AWS Lambda



Choose one of the following options to create your function.


Author from scratch

Start with a simple Hello World example.




Use a blueprint

Build a Lambda application from sample code and configuration presets for common use cases.



Browse serverless app repository

Deploy a sample Lambda application from the AWS Serverless Application Repository.



Blueprints [Info](#) Export

? < 1 >

Keyword : cloudwatch-logs-process-data ✕

cloudwatch-logs-process-data

A real-time consumer of log events ingested by an Amazon CloudWatch Logs log group.

nodejs · logs · cloudwatch

CloudWatch Logs trigger

Remove



Log group

Please select the CloudWatch Logs log group that serves as the event source. Log Events sent to the log group will trigger your Lambda function with the contents of the logs received.

helloworld_loggroup

Filter name

Choose a name for your filter.

helloworld

Filter pattern

Enter an optional filter pattern.

Lambda will add the necessary permissions for Amazon CloudWatch Logs to invoke your Lambda function from this trigger.

[Learn more](#) about the Lambda permissions model.

Enable trigger

Enable the trigger now, or create it in a disabled state for testing (recommended).



Function code [Info](#)

Code entry type

Edit code inline

Runtime

Node.js 8.10

The screenshot shows the AWS Lambda console's code editor. The menu bar includes File, Edit, Find, View, Go, Tools, and Window. The left sidebar shows the 'Environment' section with a folder named 'LogStreamTriggerFt' and a file named 'index.js'. The main editor area displays the following JavaScript code in 'index.js':

```
1 const zlib = require('zlib');
2
3 exports.handler = async (event, context) => {
4   const payload = new Buffer(event.awslogs.data, 'base64');
5   const parsed = JSON.parse(zlib.gunzipSync(payload).toString('utf8'));
6   console.log('Decoded payload:', JSON.stringify(parsed));
7   return `Successfully processed ${parsed.logEvents.length} log events.`;
8 };
9
```

The code uses the `zlib.gunzipSync` method to decompress a base64-encoded payload. The `'base64'` string and the `zlib.gunzipSync(payload)` call are highlighted with red boxes.

CloudWatch

Dashboards

Alarms

ALARM

INSUFFICIENT

OK

Billing

Events

Rules

Event Buses

Logs

Insights

Metrics

Settings

Favorites

[+ Add a dashboard](#)

CloudWatch > Log Groups

Amazon CloudWatch Logs



Create Metric Filter

Actions ▾

- 0
- 0
- 4

Filter: hello

Log Groups

helloworld_loggroup

- Create log group
- Delete log group
- Export
 - Export data to Amazon S3
 - View all exports to Amazon S3
- Subscriptions
 - Stream to AWS Lambda
 - Stream to Amazon Elasticsearch Service
 - Remove Subscription Filter



Log Groups 1-1

Metric Filters Subscriptions

None

Step 1: Choose Destination

Step 2: Configure Log Format and Filters

Step 3: Review

Lambda Function

Choose the Lambda function that should execute when a log event matches the filter you are going to specify. [Learn more about Lambda functions.](#)



Lambda Function *

LogStreamTriggerFunc



Warning

Streaming large amounts of CloudWatch Logs data to other destinations might result in high usage charges. We recommend that you create a Budget in the Billing and Cost Management console. For more information, see [Managing Your Costs with Budgets](#).

Cancel

Previous

Next

Start Streaming

Step 1: Choose Destination

Step 2: Configure Log Format and Filters

Step 3: Review

Review & Start Streaming to Amazon Lambda Service

CloudWatch Logs will stream data from the log group "helloworld_loggroup" to the Lambda function "LogStreamTriggerFunc".



Choose Destination

Edit

Lambda Function Name LogStreamTriggerFunc

Configure Log Format and Filters

Edit

Subscription Filter Name LambdaStream_LogStreamTriggerFunc

Filter Pattern [host, ident, authuser, date, request, status, bytes]

Cancel

Previous

Next

Start Streaming

Filter events

Message

2019-08-10 07:37:48

START RequestId: 7cab1ff5-6ea8-4a5c-afb0-a1ca7fe8a40f Version: \$LATEST

2019-08-10T07:37:48.810Z 7cab1ff5-6ea8-4a5c-afb0-a1ca7fe8a40f Decoded payload:

```
{
  "messageType": "DATA_MESSAGE",
  "owner": "████████████████████",
  "logGroup": "helloworld_loggroup",
  "logStream": "05d6d898fc9676b8468c3fe74ded9382",
  "subscriptionFilters": [
    "LambdaStream_LogStreamTriggerFunc"
  ]
},
{
  "logEvents": [
    {
      "id": "34910091598381004365381727989893135091847801358698938368",
      "timestamp": 1509422047800,
      "message": "27.0.3.145 - - [10/Aug/2019:07:37:27 +0000] \"GET /helloworld HTTP/1.1\" 200 20 b401dcd7-bb41-11e9-9070-6dac7fc19459",
      "extractedFields": {
        "date": "10/Aug/2019:07:37:27 +0000",
        "request": "GET /helloworld HTTP/1.1",
        "ident": "-",
        "bytes": "20 b401dcd7-bb41-11e9-9070-6dac7fc19459",
        "host": "27.0.3.145",
        "authuser": "-",
        "status": "200"
      }
    }
  ]
}
}
```



Traceを使った可視化

AWS X-Ray



リクエスト実行状況の確認

アプリケーションを構成する個々のサービスやリソースの実行結果ステータスを集計し、アプリケーションの実行状況をエンドツーエンドで確認可能



アプリケーションの問題の検出

アプリケーションの実行状況についての関連する情報を収集し、問題の根本原因を調査可能



AWSとの連携

Amazon EC2, Amazon ECS, AWS Lambda, AWS Elastic Beanstalk と連携



アプリケーションのパフォーマンス向上

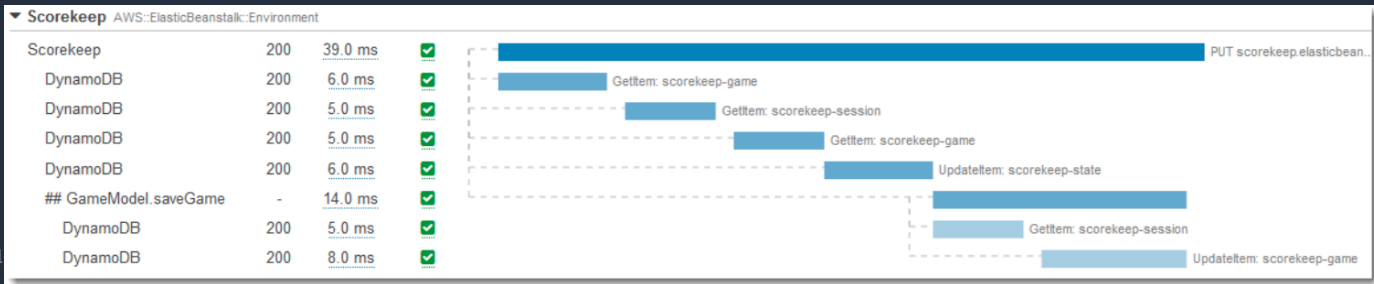
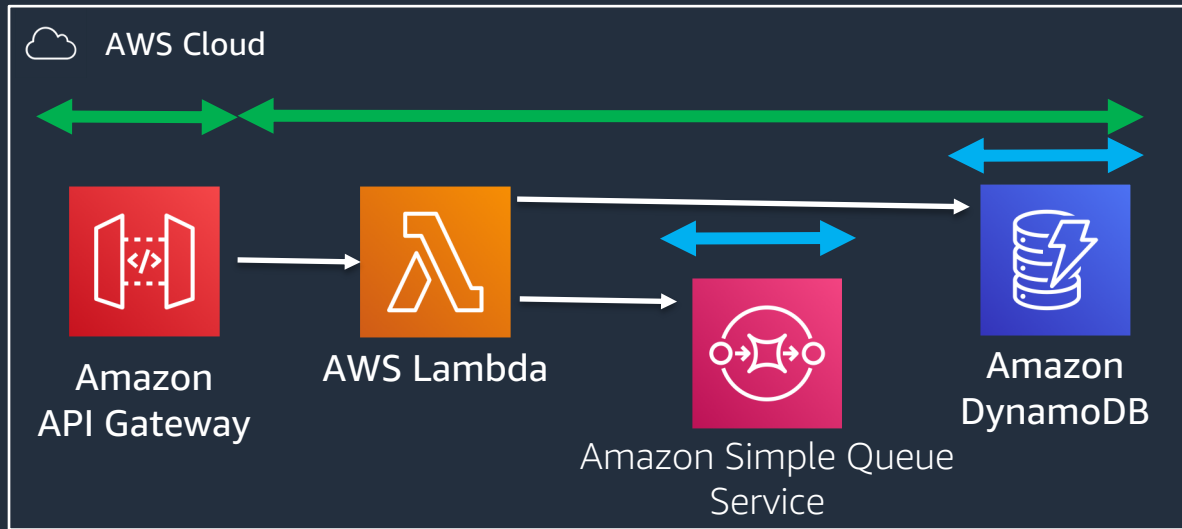
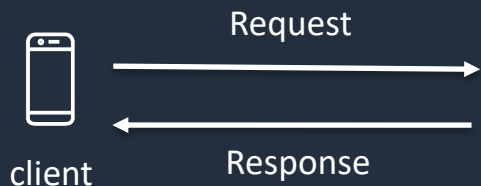
サービスやリソースの関係をリアルタイムで表示し、レイテンシ増加やパフォーマンス低下などのボトルネックを特定可能



さまざまなアプリケーション向けの設計

非同期のシンプルなイベント呼び出し、3層のウェブアプリケーション、数千のサービスから構成される複雑なマイクロサービスも分析可能

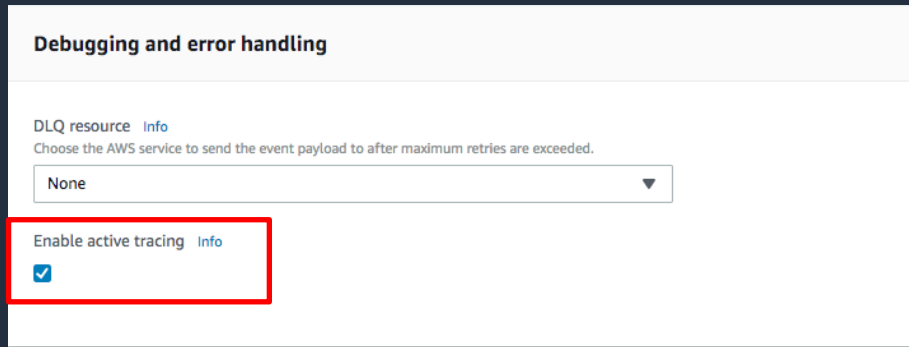
AWS X-Ray コンポーネント



AWS LambdaでX-Rayを利用するためには、

Lambda用パッケージにX-Ray SDKを追加し、アクティブトレースをONにする。
IAM roleも必要、Managed policyが用意されている

- Pythonの場合
 - Python 2.7, Python3.6以降
- Node.jsの場合
 - Node.js 4.3以降
- Javaの場合
 - Java8以降
- Goの場合
 - Go1.7以降
- .NETの場合
 - .NET Core 2.0以降



Debugging and error handling

DLQ resource [Info](#)
Choose the AWS service to send the event payload to after maximum retries are exceeded.

None ▼

Enable active tracing [Info](#)

CLIであれば、
`--tracing-config` オプション

実装例(Python)

```
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all
```

```
patch_all()
```

```
def main(event, context):
```

```
    xray_recorder.begin_segment('main segment')
```

```
    (main処理)
```

```
        xray_recorder.begin_subsegment('sub segment')
```

```
        (処理A)
```

```
        xray_recorder.end_subsegment('sub segment')
```

```
    xray_recorder.end_segment
```

```
    return
```

実装例(Python)

```
from aws_xray_sdk.core import xray_recorder  
from aws_xray_sdk.core import patch_all
```

patch_all()

対応しているライブラリにパッチ適用

pythonの場合

-botocore, boto3

-requests

-sqlite3

-mysql-connector-python

など

patch_allではなく patch('boto3')などとすることも可能

実装例(Python)

```
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all
```

```
patch_all()
```

```
def main(event, context):
```

```
    xray_recorder.begin_segment('main segment')
```

処理全体のセグメントを定義

```
    (main処理)
```

```
        xray_recorder.begin_subsegment('sub segment')
```

```
        (処理A)
```

```
        xray_recorder.end_subsegment('sub segment')
```

```
    xray_recorder.end_segment
```

```
    return
```

実装例(Python)

```
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all
```

```
patch_all()
```

```
def main(event, context):
```

```
    xray_recorder.begin_segment('main segment')
```

```
    (main処理)
```

```
        xray_recorder.begin_subsegment('sub segment')
```

```
        (処理A)
```

```
        xray_recorder.end_subsegment('sub segment')
```

```
    xray_recorder.end_segment
```

```
    return
```

処理の中でサブセグメントを作ることも可能

サービスグラフ

各ノードの呼び出しの結果を色で分類、割合を円グラフに
(サービスマップ)

- グリーン 成功した呼び出し
- レッド 5xx errors
- イエロー 4xx errors
- パープル 429 Too Many Requests (スロットリングエラー)

- ・ 平均レイテンシ (ms)
- ・ トレース数 (trace/min)
- ・ サービス名
- ・ サービスの分類



A group of people are gathered around a table in a meeting. One person is using a laptop, while another points at the screen. The scene is brightly lit, suggesting a modern office environment.

まとめ

まとめ

AWS サーバーレスサービスに関わる考え方やMetricsをご紹介してきました。

- アプリケーション開発、運用においては一般的な知識と経験はサーバーレスでも活用できます
- ログ設計なども通常のアプリケーション設計と変わらない
 - CloudWatch Logsや3rd partyツールなどの監視ツールも同様
- サーバーレスサービスの制約や仕様を理解し、これまでのアプリケーション開発経験と組み合わせることで従来のアプリケーションよりも、柔軟なアプリケーション開発/運用もできる可能性がある

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。

8月の Black Belt Online Seminar 配信予定

<https://amzn.to/JPWebinar>

~~08/06 (火) 12:00-13:00 AWS Glue~~

~~08/13 (火) 12:00-13:00 実践的 Serverless セキュリティプラクティス~~

~~08/14 (水) 18:00-19:00 AWS Serverless Application Model~~

08/20 (火) 12:00-13:00 Serverless モニタリング

08/21 (水) 18:00-19:00 AWS AppSync

08/28 (水) 18:00-19:00 Amazon Aurora with PostgreSQL Compatibility



DEV DAY

TOKYO

今年もやります！イノベーションをリードするDeveloperのための秋の祭典「**AWS DevDay Tokyo 2019**」

- ・今おさえておくべき技術領域を網羅し、新たな時代に活躍するDeveloperとなるための知識とスキルを身に着ける2日間
- ・豪華スピーカーによるゼネラルセッション
- ・50にのぼるセッションとワークショップで構成。CFP枠を拡大し、より実践的なコンテンツへ

日程：2019年10月3日（木）ー 4日（金）

会場：神田明神ホール（東京都千代田区 神田明神内）

主催：アマゾン ウェブ サービス ジャパン株式会社

協賛：インテル株式会社



今年の会場は「神田明神」



Save the dateはこちらへ
<https://amzn.to/devdaytokyo2019>

AWS の日本語資料の場所「AWS 資料」で検索



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 ▾

コンソールにサインイン

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 🔍

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>

