

春の Observability 祭り 2025 ~進化する Amazon CloudWatch 基礎から最新機能まで完全解説~

# Amazon CloudWatch の進化

Mitsuaki Tsugo

Solutions Architects

Amazon Web Service Japan G.K.



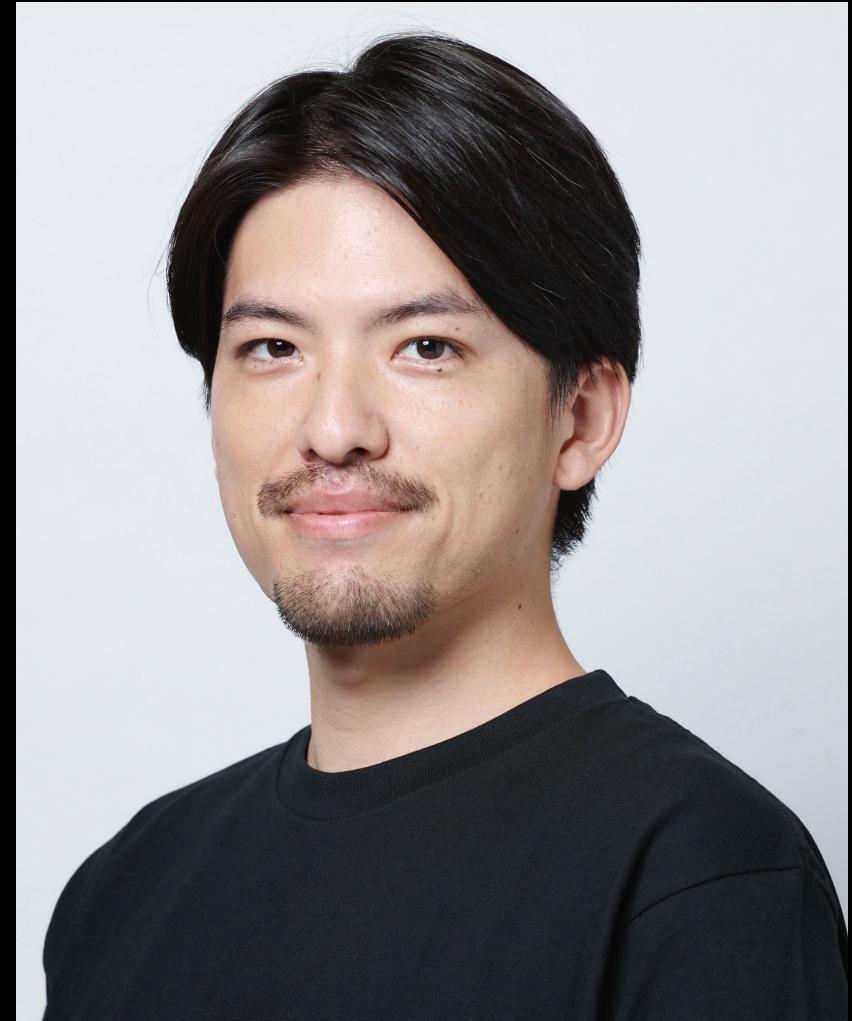
# 自己紹介

## 津郷 光明 Mitsuaki Tsugo

アマゾンウェブサービスジャパン  
ソリューションアーキテクト

製造業のお客様を中心にご支援しています。

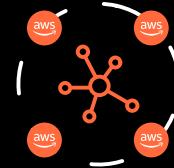
Observability / IaC x GenAI が最近のテーマ



# Observability helps you



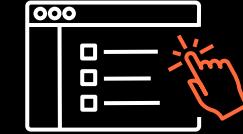
サービス状態  
(正常 / 異常) の把握



パフォーマンスと  
可用性の向上



運用コストの  
削減



顧客体験の  
向上

System

Business

**Booking.com**

ウェブサイトのリアルユーザー  
パフォーマンスを収集・測定



**CloudPassage**

潜在的なバグの発見と修正を高速化、  
機能開発が加速



**mapbox**

サードパーティシステムのセットアップ、  
構成、学習の運用負荷を軽減

**regoconsulting**

顧客からの通知前に問題を解決



# Observability でよく利用されるテレメトリデータ

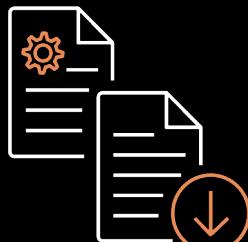
## メトリクス



時間間隔で計測されたデータの数値表現

傾向の把握、予測に役立つ

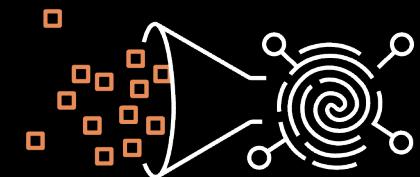
## ログ



タイムスタンプが記録された、時間の経過とともに起こったイベントの記録

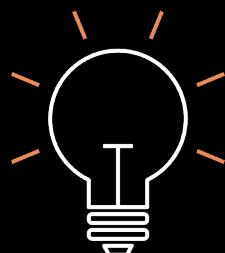
予測不可能な振る舞いの発見に役立つ

## トレース



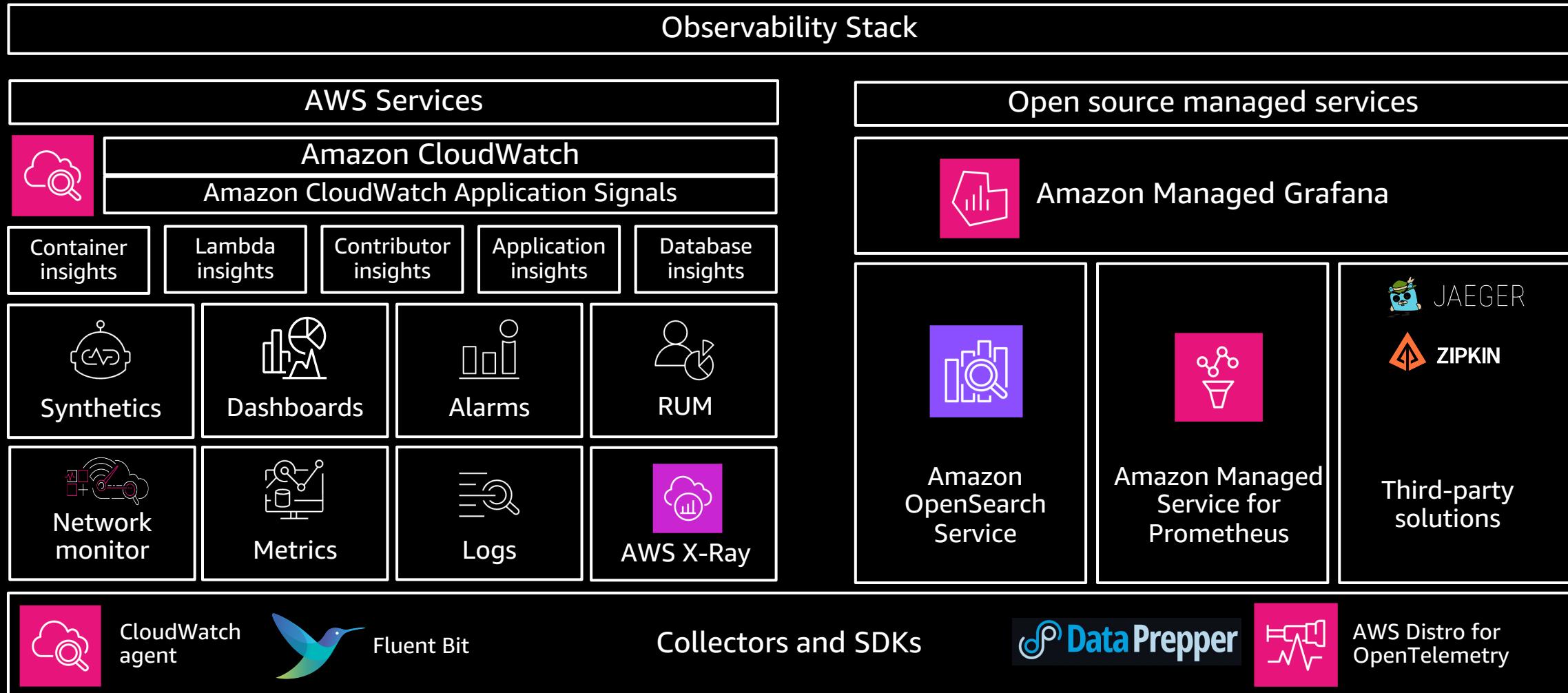
エンドツーエンドのリクエストフローの記録

リクエストの流れと構造の両方を可視化することで因果関係の追跡に役立つ



Observability

# AWS Observability Services





# Amazon CloudWatch

AWS のリソース、アプリケーション、  
オンプレミスのモニタリングサービス



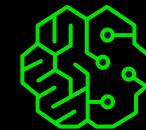
Auto-scaling and serverless



監視の集約

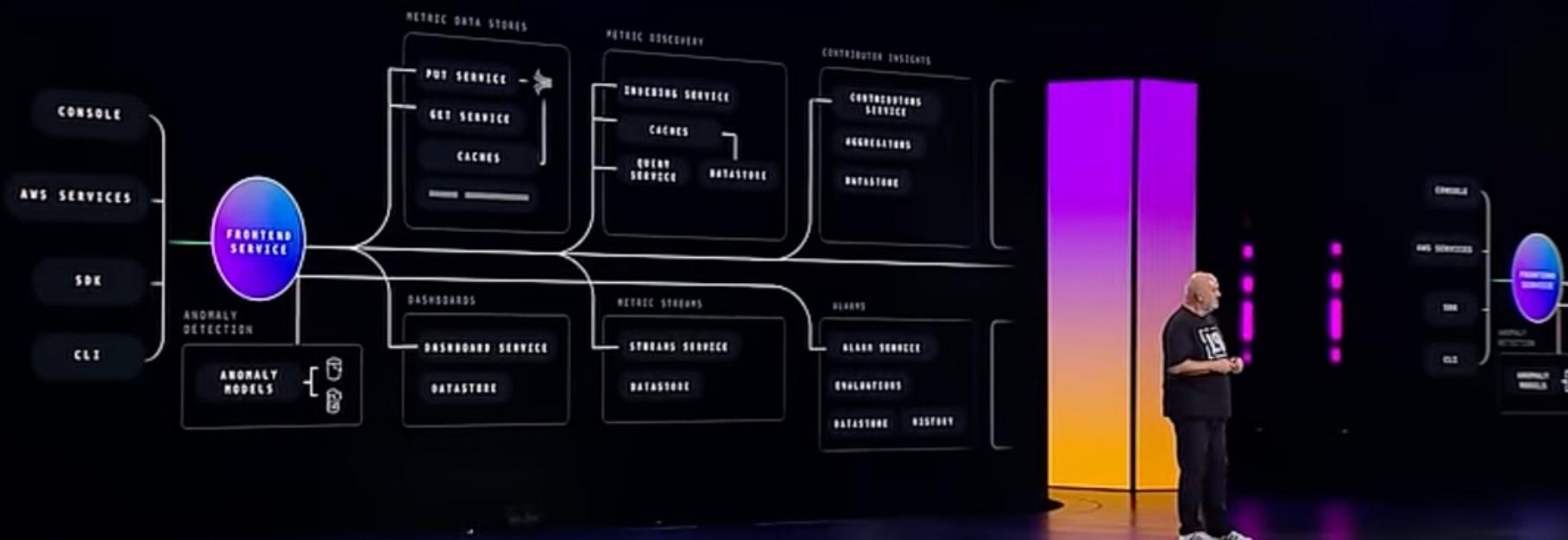


分析・トラブルシュート



生成 AI を利用した状況把握・運用改善  
自動アクション

# Amazon CloudWatch は大きく投資され、進化している

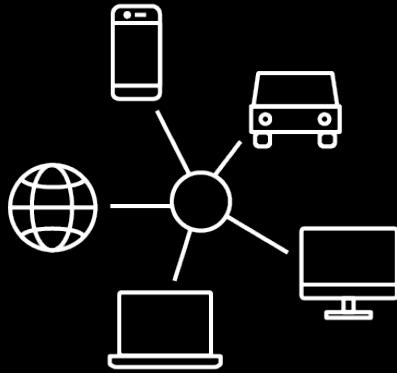


分離されたサービスの中で機能だけでなく、コードやライブラリも柔軟に選定され開発されている

AWS re:Invent 2024

Dr. Werner Vogels, VP and CTO at Amazon.com Keynoteより

# Amazon CloudWatch の何が良いのか



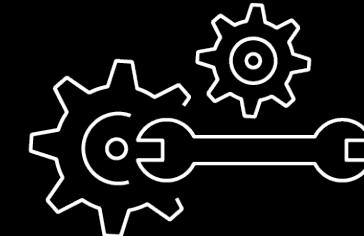
---

広いカバー領域



---

導入の簡易さ



---

柔軟性

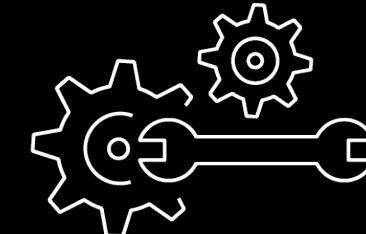
# Amazon CloudWatch の何が良いのか



広いカバー領域

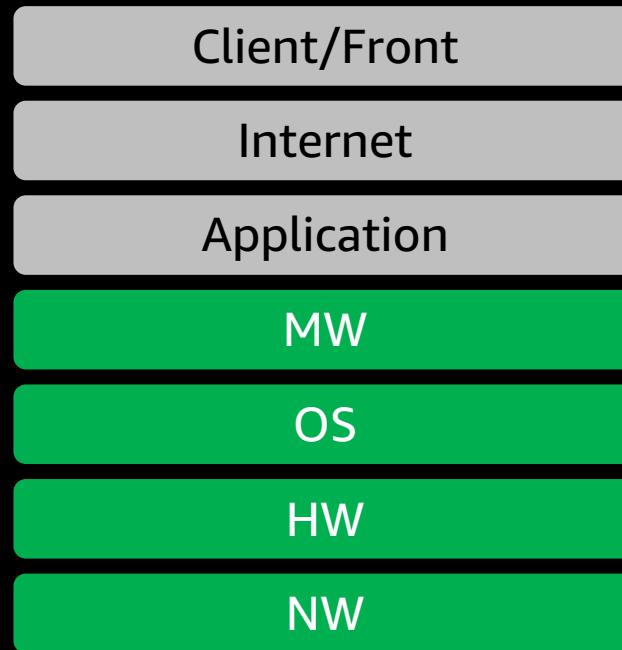


導入の簡易さ

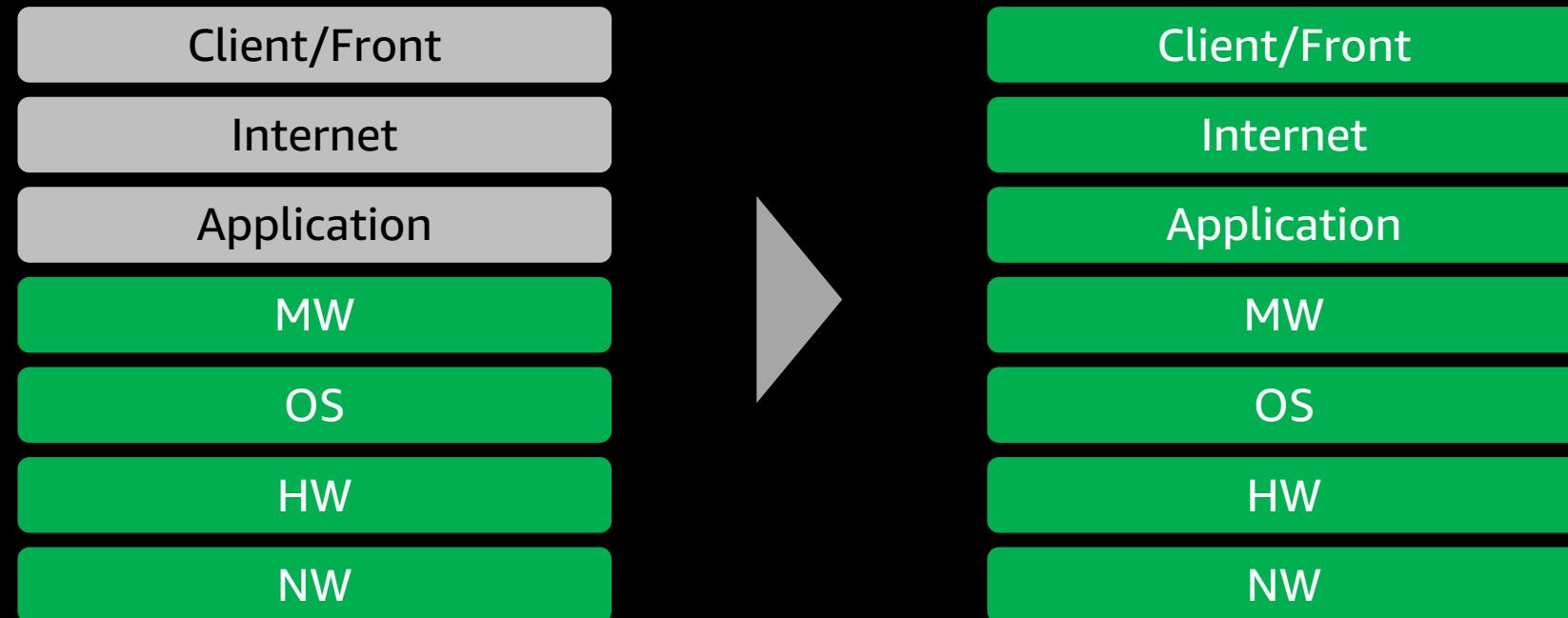


柔軟性

サービス状態や実際の利用状況をモニタリングする  
障害の把握や性能改善、機能改善につながる洞察を得る



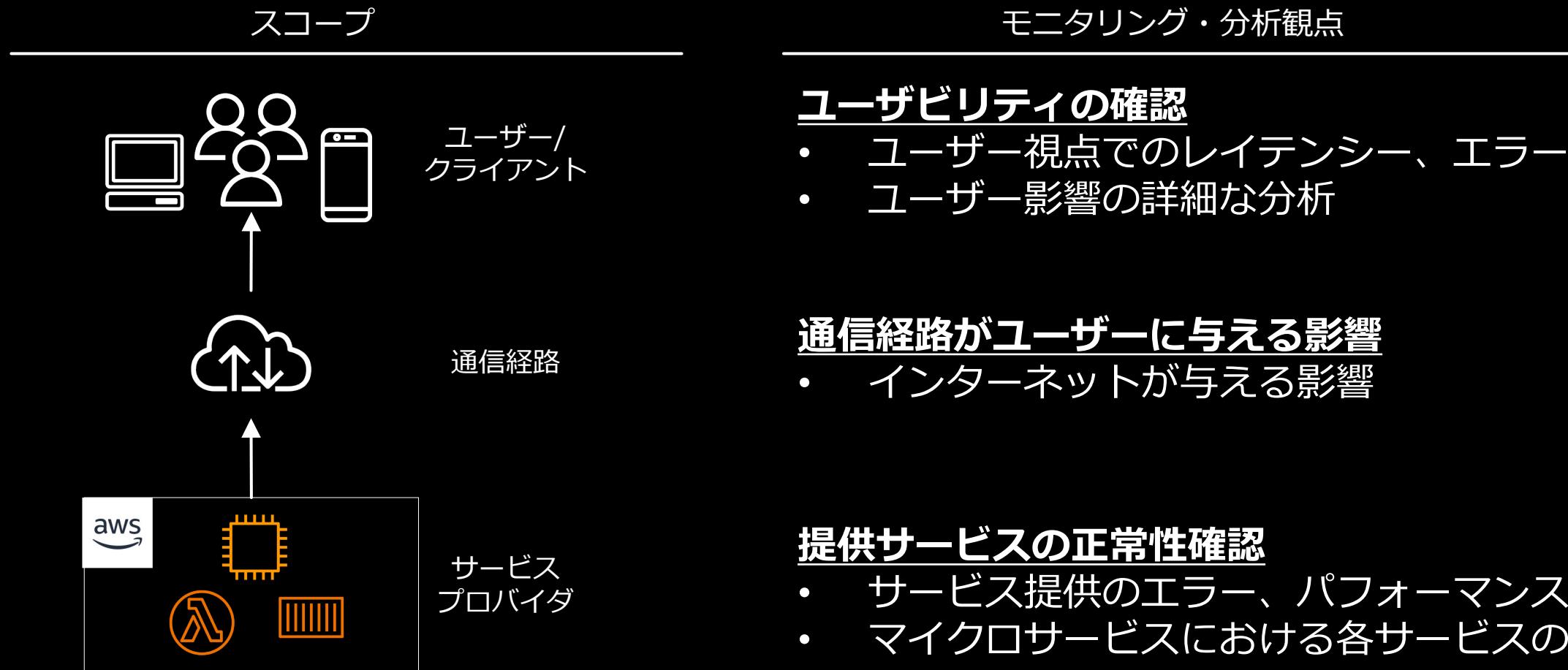
サービス状態や実際の利用状況をモニタリングする  
障害の把握や性能改善、機能改善につながる洞察を得る



インフラに限定せずアプリケーションまで含めて  
サービスを“総合的に”評価する必要がある

# テレメトリデータを取得する対象は拡大している

## 全てのレイヤーでテレメトリを取得し、監視・分析する



テレメトリデータを取得する対象は拡大している  
全てのレイヤーでテレメトリを取得し、監視・分析する



### モニタリング・分析観点

#### ユーザビリティの確認

- ユーザー視点でのレイテンシー、エラー
- ユーザー影響の詳細な分析

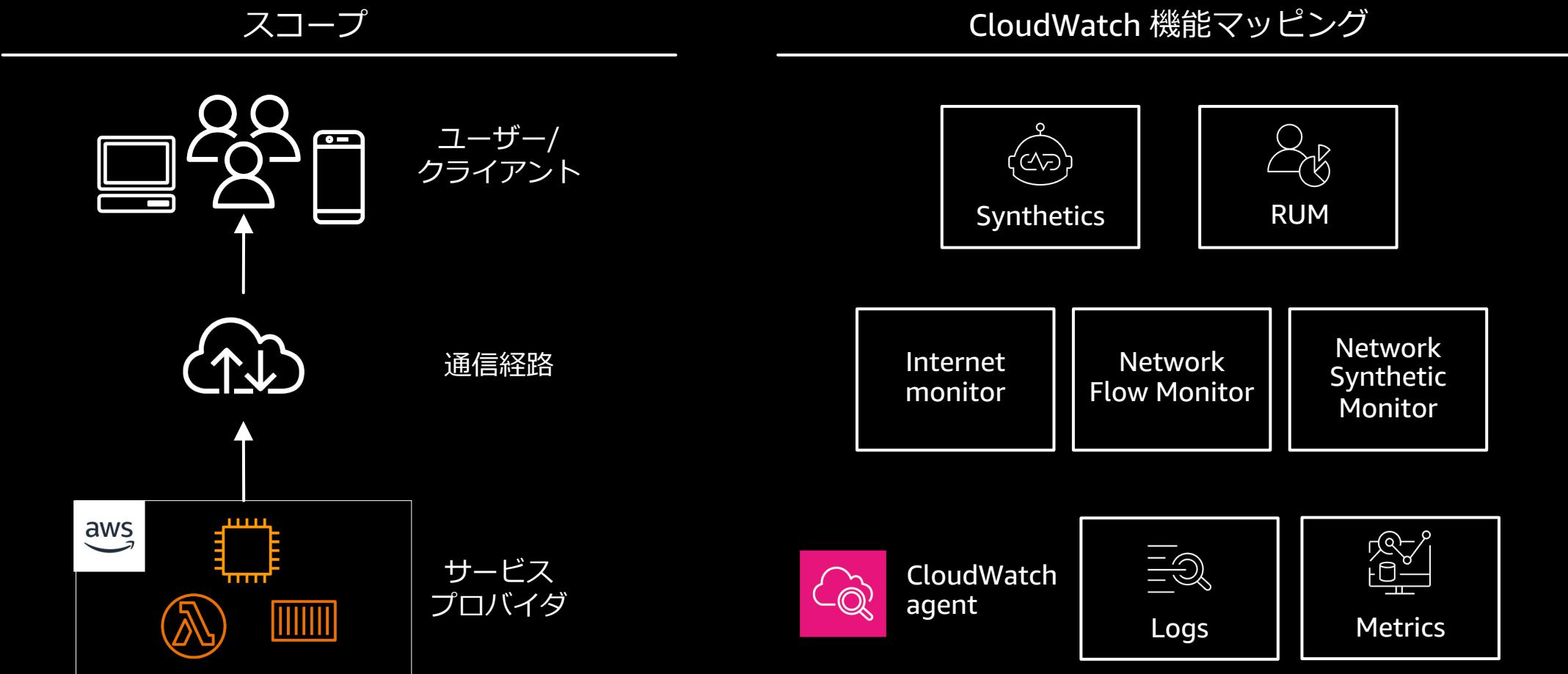
#### 通信経路がユーザーに与える影響

- インターネットが与える影響

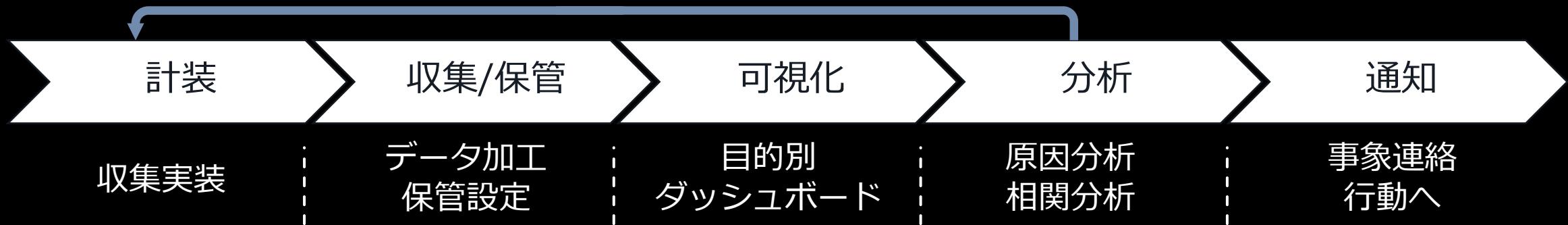
#### 提供サービスの正常性確認

- サービス提供のエラー、パフォーマンス
- マイクロサービスにおける各サービスの状態

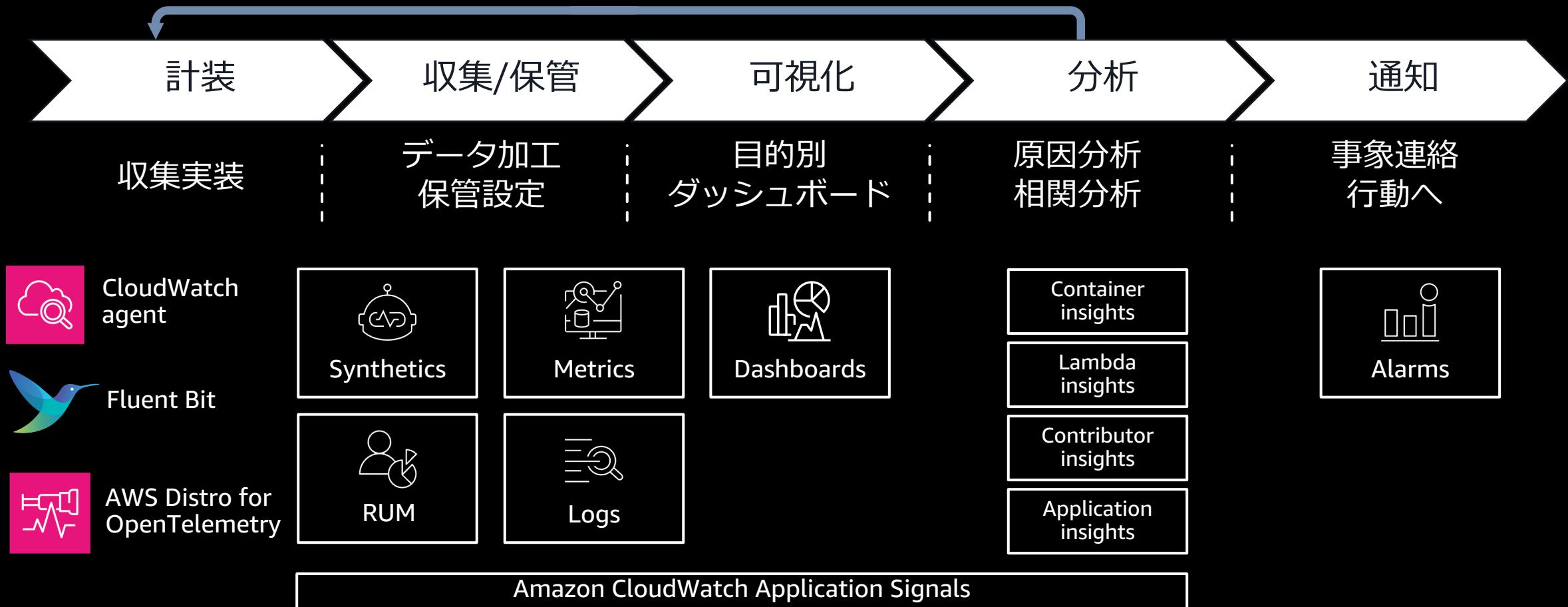
# CloudWatch は全てのレイヤーを網羅する



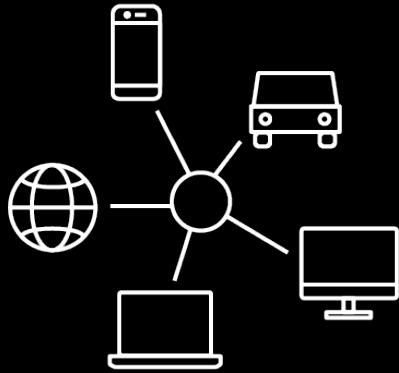
# CloudWatch はテレメトリ収集から分析まで網羅する



# CloudWatch はテレメトリ収集から分析まで網羅する



# Amazon CloudWatch の何が良いのか

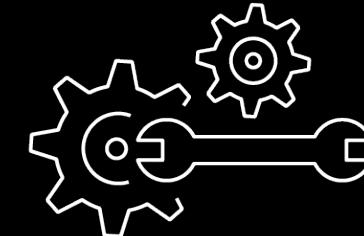


---

広いカバー領域



導入の簡易さ



---

柔軟性

# Observability でよく利用されるテレメトリデータ

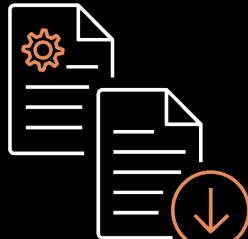
## メトリクス



時間間隔で計測されたデータの数値表現

傾向の把握、予測に役立つ

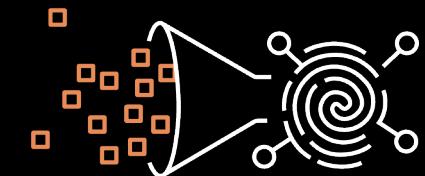
## ログ



タイムスタンプが記録された、時間の経過とともに起こったイベントの記録

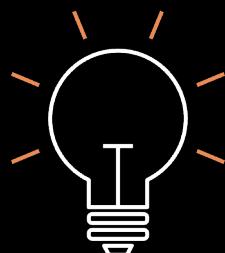
予測不可能な振る舞いの発見に役立つ

## トレース



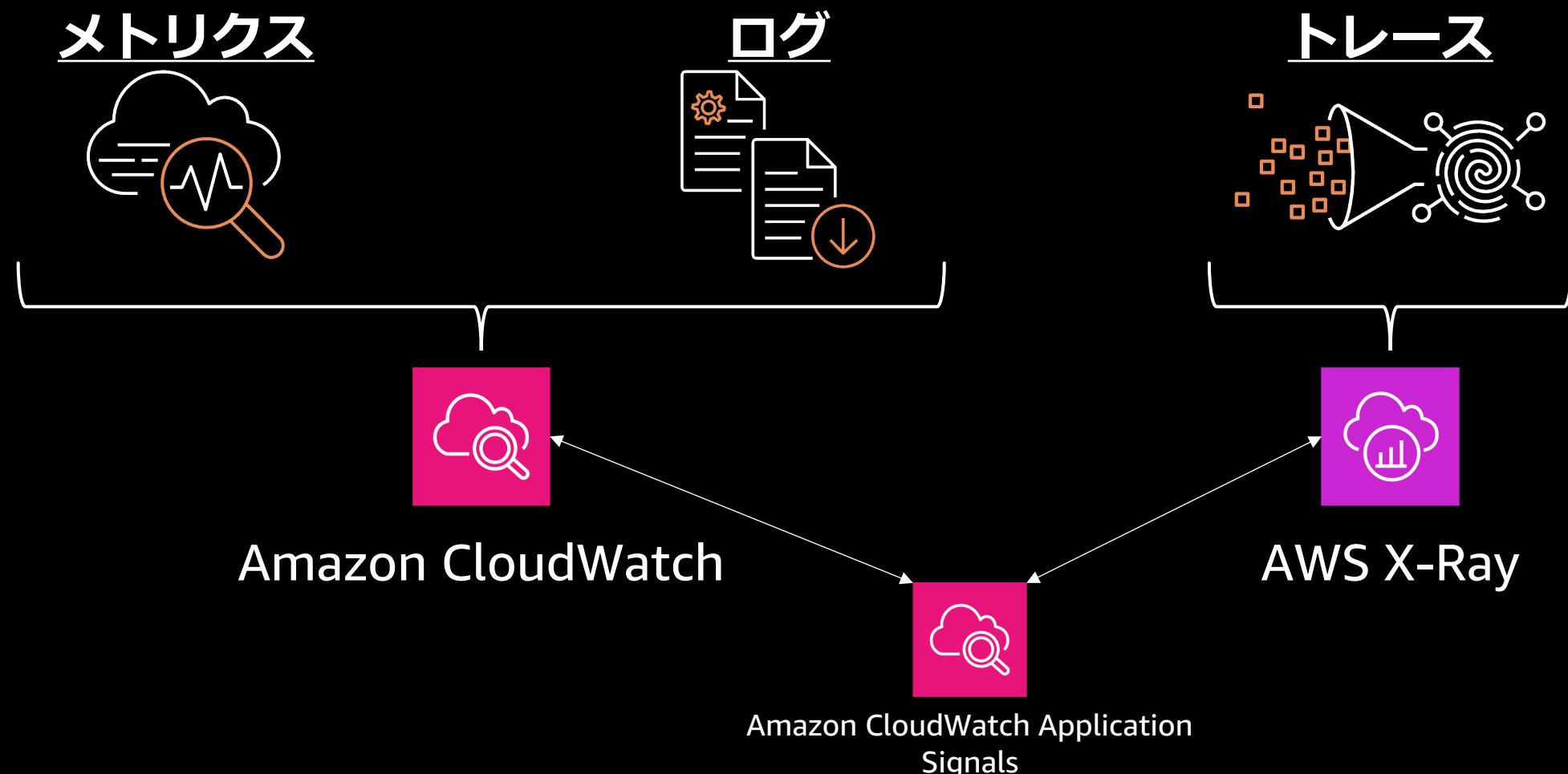
エンドツーエンドのリクエストフローの記録

リクエストの流れと構造の両方を可視化することで因果関係の追跡に役立つ



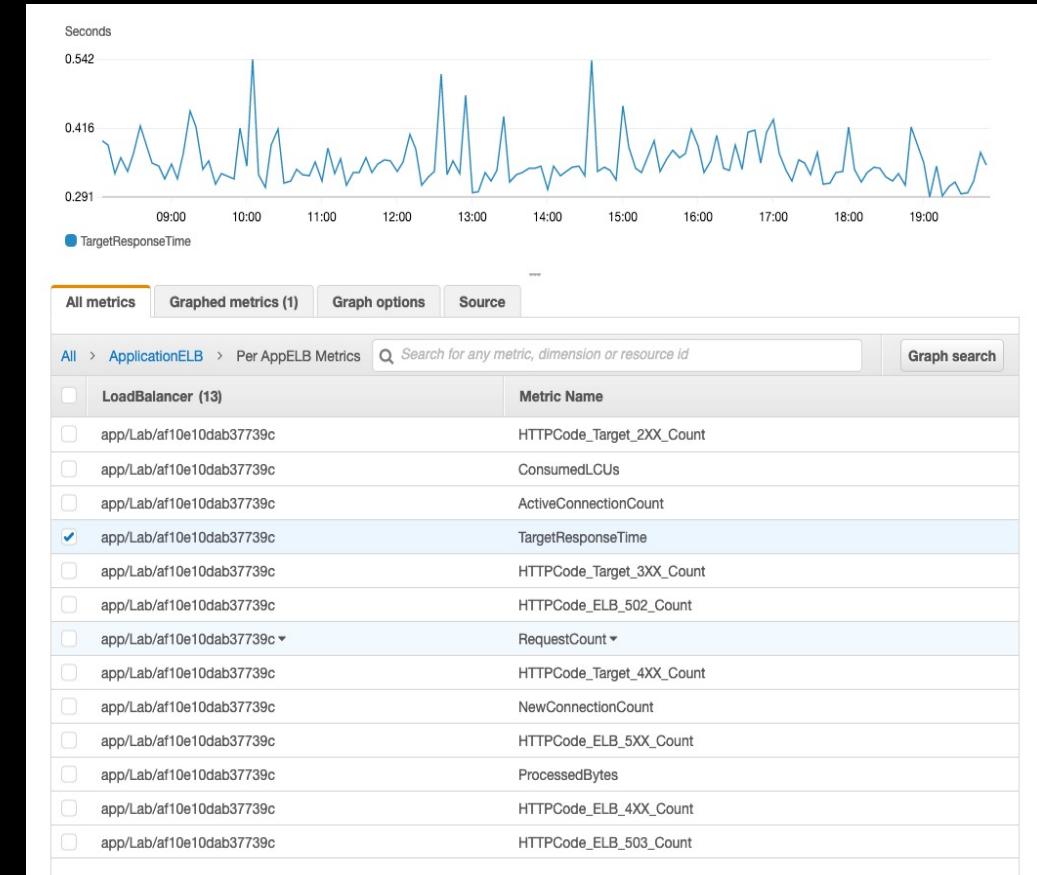
## Observability

# CloudWatch / X-Ray は最初の選択肢



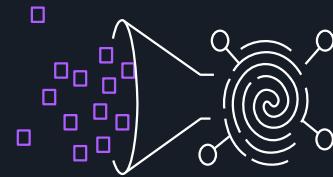
# CloudWatch / X-Ray は最初の選択肢

- AWS サービスに対する**ビルトインのメトリクスやログ、トレースが利用可能**  
特に準備不要でテレメトリデータの収集を始められる  
e.g. EC2、RDS、Lambda、ECS、S3、DynamoDB...more
- AWS サービスとの**インテグレーションが容易**  
テレメトリデータの S3 へのバックアップ  
Alarm からのオートメーション処理  
...more
- 1 プラットフォームで完結できる
- **サーバーレスで管理不要**



# Amazon CloudWatch Application Signals

アプリケーションの状態を把握・分析するための情報を自動で収集・可視化



テレメトリデータ  
自動収集



サービスの検出/  
トポロジー可視化



ダッシュボード提供  
トレース・メトリクスの  
確認

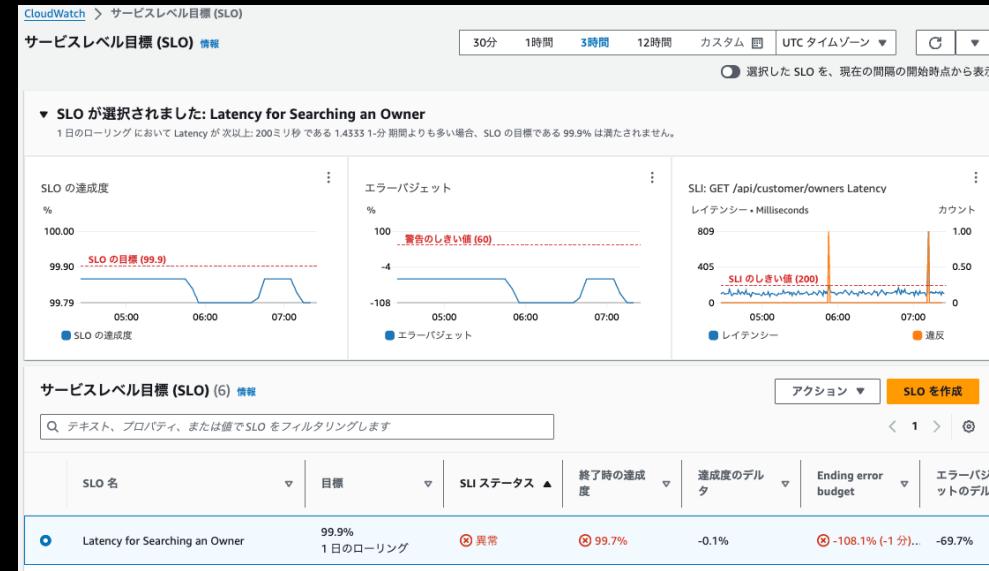


SLOモニタリング



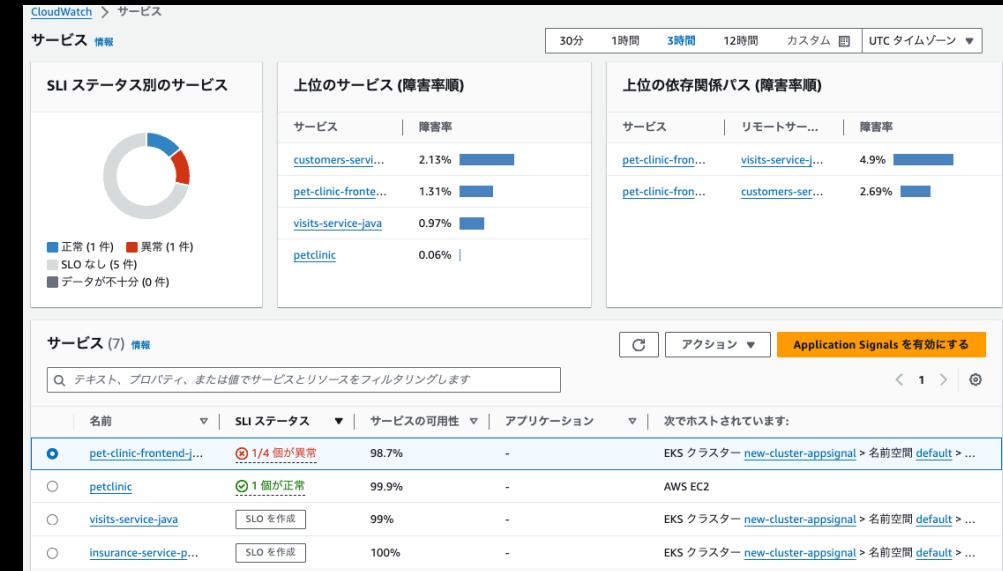
CloudWatch Synthetics /  
CloudWatch RUM  
との連携

# Services Dashboard

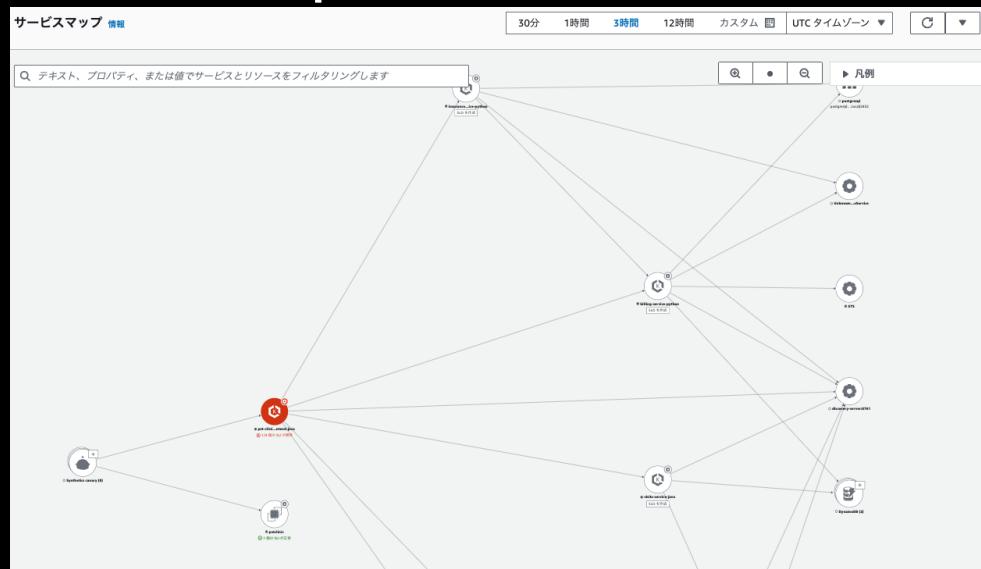


計装・可視化の簡易さ

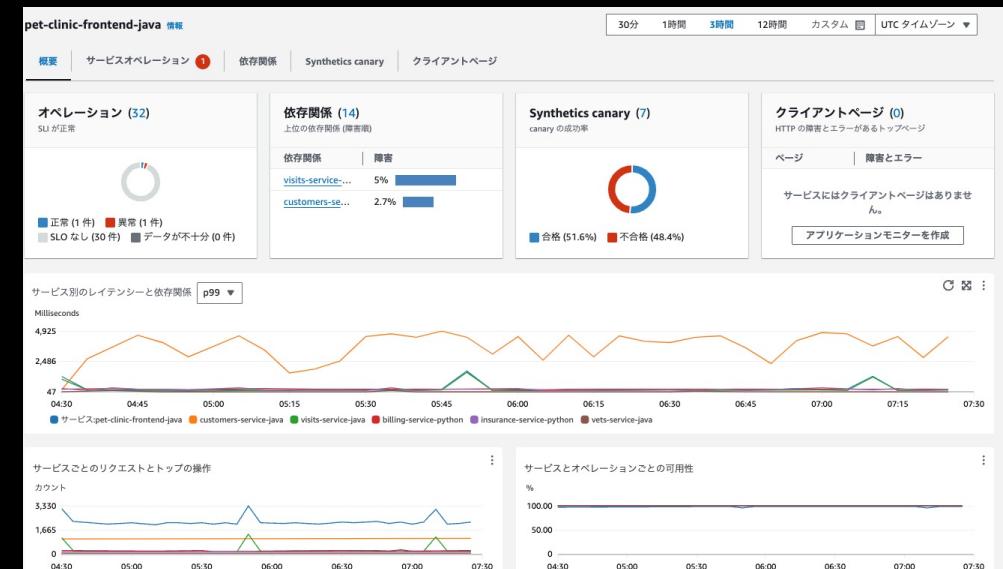
# SLO Dashboard



# Service Map



# Service



# 自然言語でデータ分析クエリ

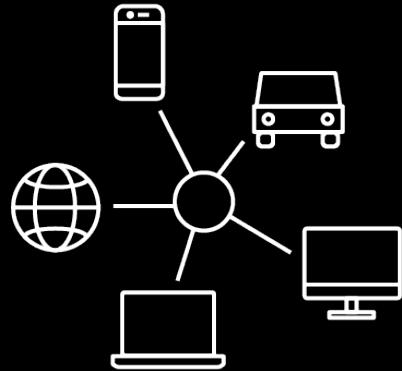
CloudWatch Logs Insight と Metrics Insights で  
生成 AI により自然言語でクエリを生成

- 自然言語の質問からクエリ生成
- 生成したクエリの解説も付与  
クエリ構文の学習を支援
- 自然言語を利用して既存クエリを改良  
繰り返し実行するクエリの継続的な改善に有益
- 英語のみサポート

専門知識がなくても分析が可能に



# Amazon CloudWatch の何が良いのか



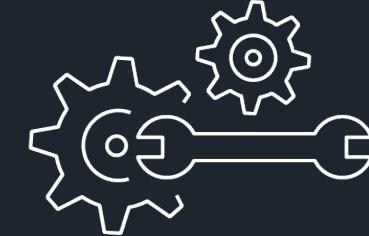
---

広いカバー領域



---

導入の簡易さ



---

柔軟性

# 進化した CloudWatch Agent

- OpenTelemetryに対応
- OpenTelemetry Protocol (OTLP) をサポート
- X-Ray SDK のデータ転送先として利用可能に



CloudWatch Agent



- 旧来の CloudWatch Agent
- X-Ray daemon
- OpenTelemetry Collector の役割

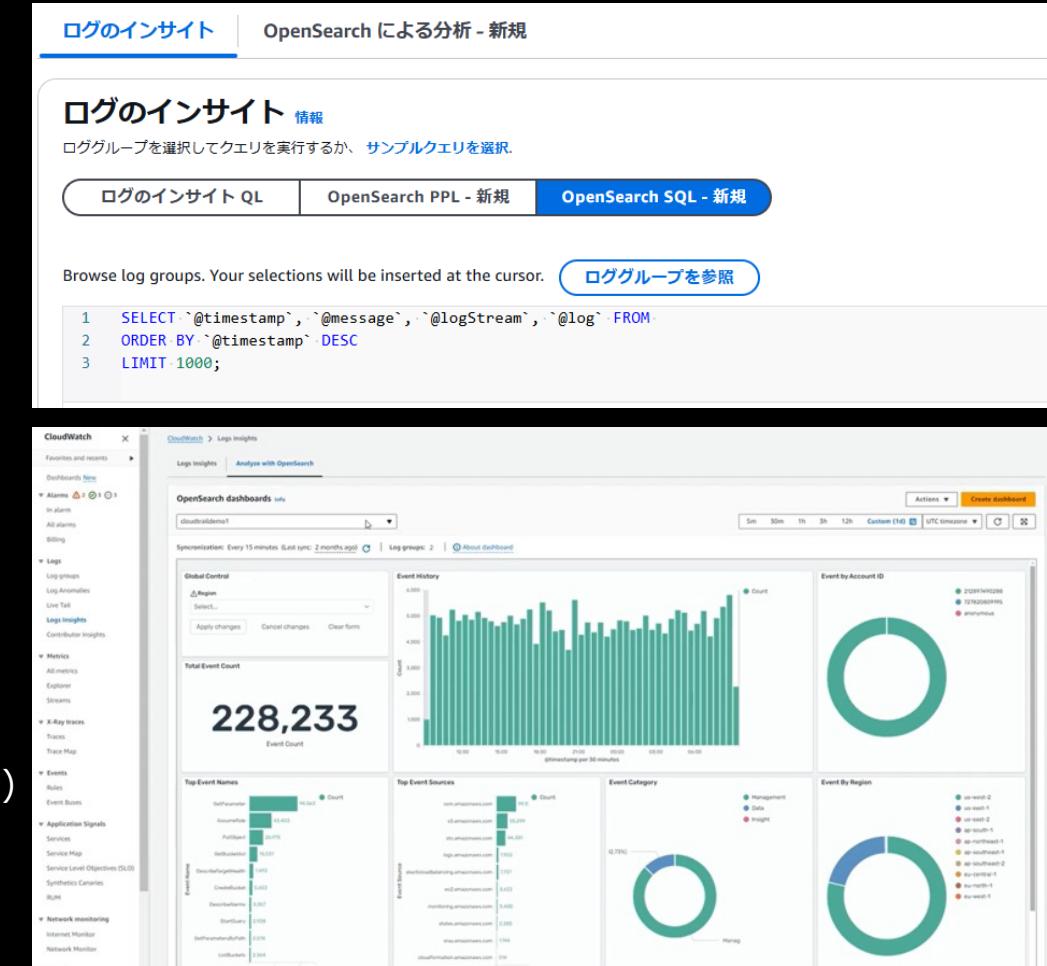
CloudWatch x OpenTelemetry の  
実装が可能に

# CloudWatch Logs と OpenSearch が Zero-ETL 統合

- CloudWatch Logs Insights で、OpenSearch SQL と PPL (パイプ処理言語) が利用可能に
- CloudWatch Logs Insights 上で以下のログの OpenSearch のダッシュボードを設定なしに作成可能
  - VPC Flow logs / WAF / CloudTrail のログ
- Zero-ETL 統合によりデータをコピーせずに CloudWatch のログを OpenSearch 上で分析可能
- OpenSearch から CW Logs へのダイレクトクエリをサポートしているリージョンで利用可能 (東京リージョン含む)

[https://docs.aws.amazon.com/ja\\_jp/opensearch-service/latest/developerguide/direct-query.html#direct-query-cloudwatch-logs-regions-table](https://docs.aws.amazon.com/ja_jp/opensearch-service/latest/developerguide/direct-query.html#direct-query-cloudwatch-logs-regions-table)

<https://aws.amazon.com/jp/blogs/news/new-amazon-cloudwatch-and-amazon-opensearch-service-launch-an-integrated-analytics-experience/>

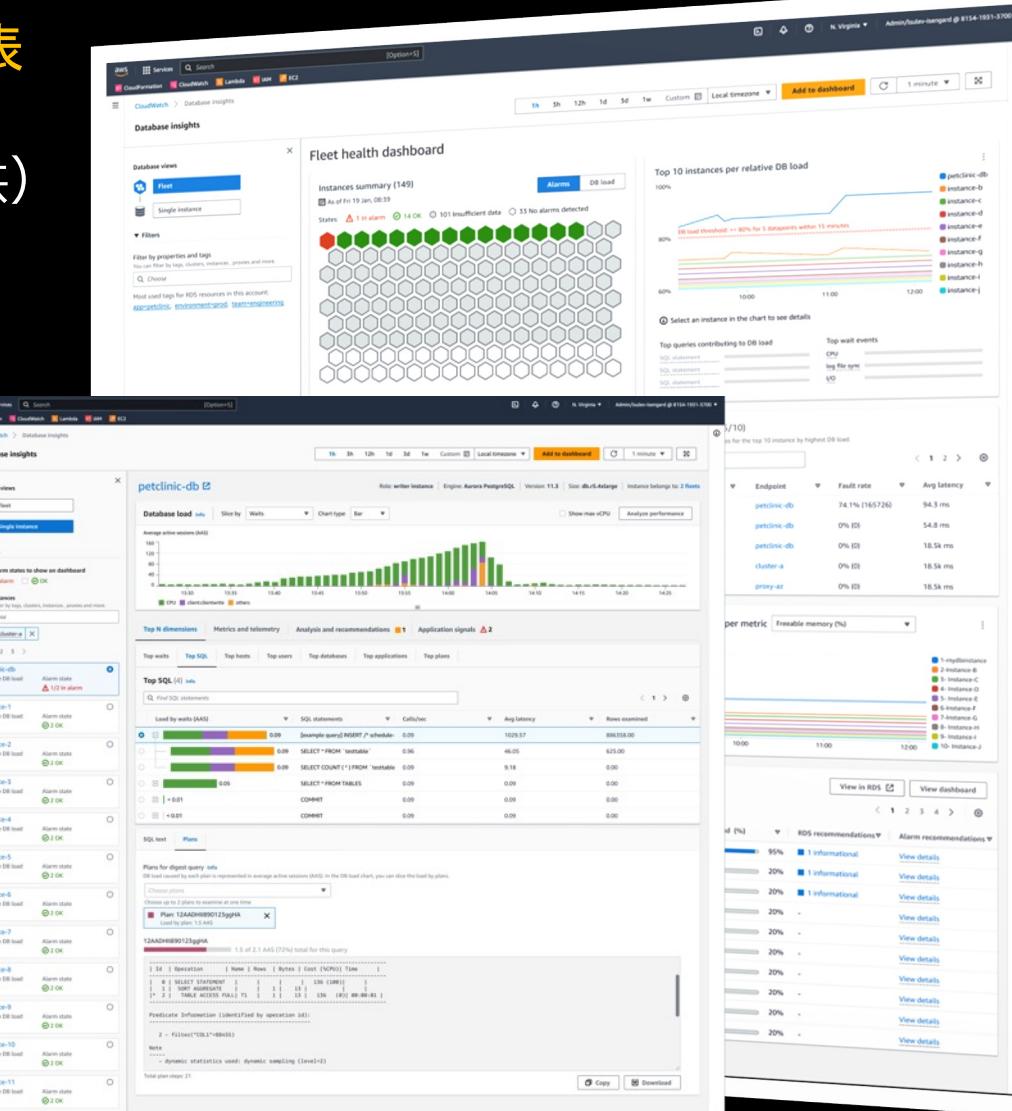


# Amazon CloudWatch Database Insights

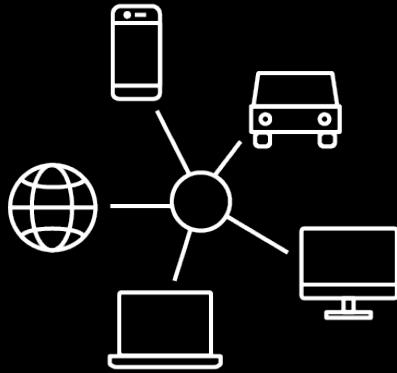
## Amazon Aurora データベースオブザーバビリティ機能を発表

- すべてのデータベースステレメトリを CloudWatch の統合ビューで可視化（事前構築済みの以下 2 つのビューを提供）
  - Fleet Health : DB フリート全体の状態把握のためのビュー
  - Database Instance : DB インスタンス毎の詳細なビュー
- 有効時に Standard または Advanced モードを選択（2つモードの違いは [こちら](#) を参照）  
※
- Amazon Aurora MySQL / Postgres でサポート
- 東京、大阪を含む全商用リージョンで利用可能

※Advanced モード有効には Performance Insights 有効化が必要



# Amazon CloudWatch の何が良いのか



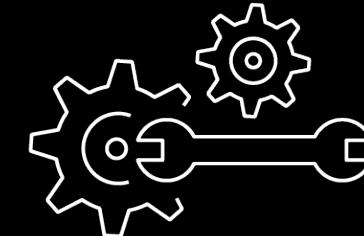
---

広いカバー領域



---

導入の簡易さ



---

柔軟性

ここで紹介したものは概要と一部機能のみ  
この後のパートにご期待ください

# まとめ

- Amazon CloudWatch は継続的な進化を続けており機能も拡大
- CloudWatch の良さは”広いカバー領域” ”導入の簡易さ” ”柔軟性”
- 最近は OpenTelemetry にも対応



# Thank you!

Mitsuaki Tsugo  
Solutions Architect



春の Observability 祭り 2025

# Amazon CloudWatch を使ってネットワーク監視を行うには

宮崎 友貴

アマゾンウェブサービスジャパン合同会社  
ソリューションアーキテクト

# 自己紹介

## 宮崎 友貴

アマゾンウェブサービスジャパン  
ソリューションアーキテクト

通信業のお客様を中心にご支援しています。



好きな AWS の observability サービス  
Amazon CloudWatch Metrics  
Amazon Managed Grafana

# Agenda

- ネットワークオブザーバビリティとは
- Network Flow Monitor とは
- Network Synthetic Monitor とは
- Internet Monitor とは
- まとめ

# ネットワークのオブザーバビリティとは

ネットワークの健全性、パフォーマンスを包括的に把握  
迅速な検出と解決を効率的に行えることで障害発生時の影響を最小化

## 検出する

問題が発生したらすぐに特定して、調査と修正ができるだけ早く開始できます



## 調査する

障害発生時のテレメトリーを調べて問題の性質を把握します



## 修復する

問題による顧客への影響を一時的または恒久的に軽減するための対策を講じます



## パフォーマンス

RTT (Round Trip Time) やパケットロスなどのパフォーマンス指標を継続的に測定して異常を特定します



## 位置特定する

メトリックガイドによるネットワーク障害の故障個所の特定により、トラブルシューティングを迅速化します



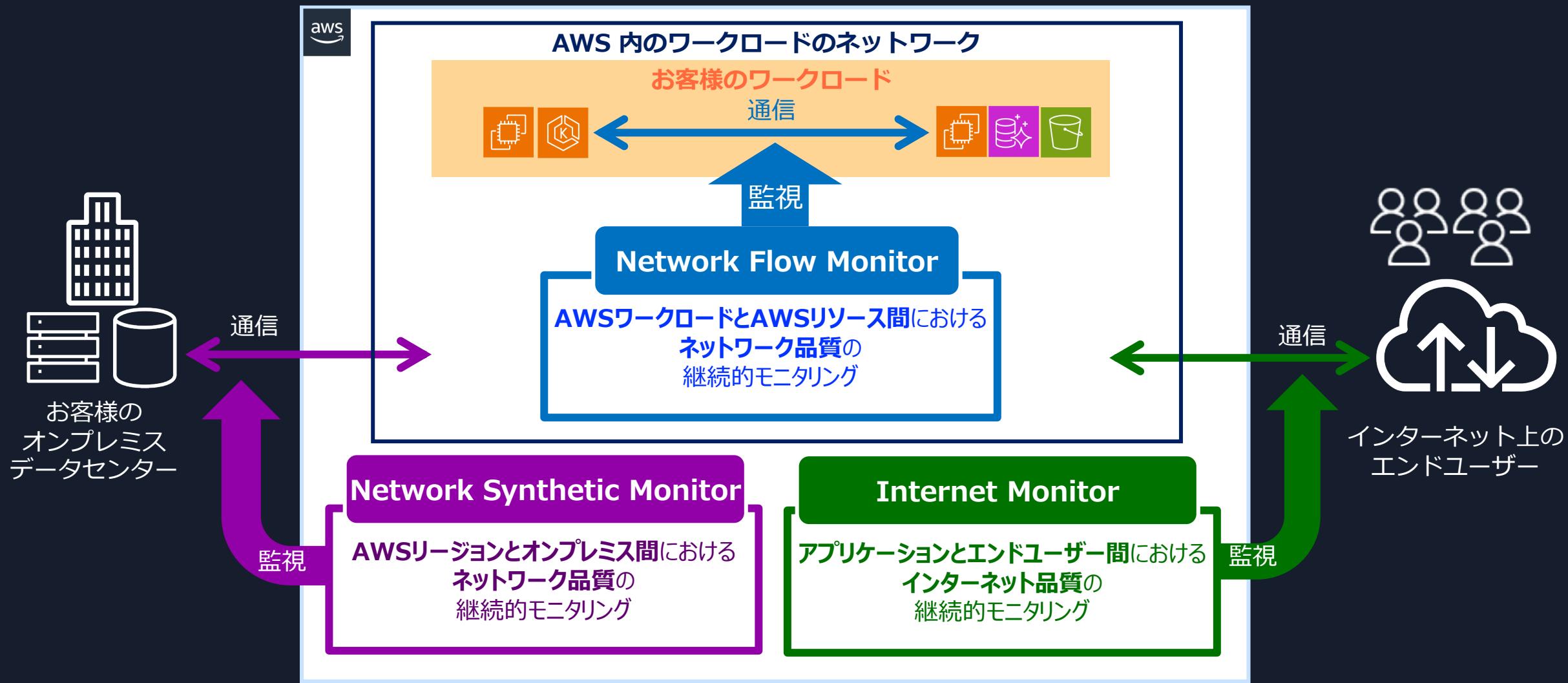
## 経路変更

障害が発生したネットワークセグメントを迂回するよう、アラート生成し、トラフィックをシフトします。



# ネットワークパフォーマンス監視を提供する AWS マネージドサービス

それぞれモニタリング対象が異なる 3つの サービス を提供  
目的に応じて 使い分け や 組み合わせ



# Network Flow Monitorとは

# Network Flow Monitor とは

AWSワークロードとAWSリソース間のネットワークを監視するマネージメントサービス

- Amazon EC2 や Amazon EKS などのコンピューティングインスタンスとAWS サービス間のトラフィックを監視
- エージェントを使用してパケットロス(%)とレイテンシー(ms)などのメトリクスをほぼリアルタイムで可視化
- ネットワークヘルスインジケーター (NHI) により AWS ネットワークの正常性を確認可能
- AWSサポートにも提供されるため、ネットワークの問題のトラブルシューティングを迅速化

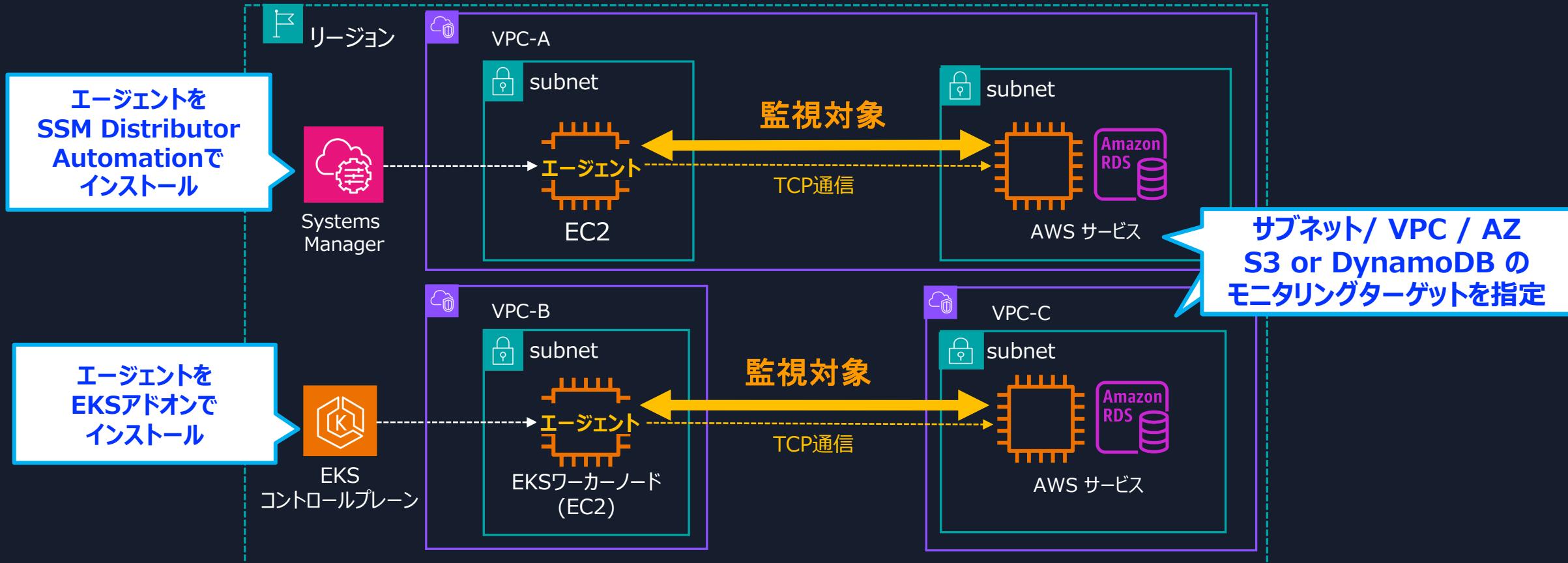


AWS サービス間のネットワークパフォーマンスを監視、可視化

障害時にAWS側かアプリケーション側かの問題の切り分けにより迅速な解決が可能に

# Network Flow Monitor のアーキテクチャ

Network  
Flow  
Monitor



- エージェントから指定したサブネット/VPC/AZ の範囲内リソースもしくは S3/DynamoDB への TCP 通信を監視
- エージェントは、TCP 接続に関連するイベントを受信し、eBPF を使用して TCP トラフィックを分析
- パフォーマンスマトリクスを Network Flow Monitor サービスのバックエンドに約30秒ごとに送信



# Network Flow Monitor の料金

Network Flow Monitor は、以下2つの料金がかかります。

## Amazon CloudWatch Network Flow Monitor 料金

### ①監視リソース(エージェント)

監視リソースあたり  
**0.0069 USD/時間**

※「監視リソース」は  
稼働中のエージェント毎に  
1つ計上されるリソース

### ②CloudWatchメトリクス

最初の10,000  
メトリクスまで  
**0.30 USD/metric/月**

- ※1つのモニターは  
5つのメトリクス を発行
- DataTransferred
  - Retransmissions
  - タイムアウト
  - RoundTripTime
  - ヘルスインジケータ(NHI)

### (関連サービス利用料)

Amazon CloudWatch  
(アラーム, …)

+  
AWS Lambda

+  
⋮

# デモ：Network Flow Monitor を使ってみよう

マネージドエージェントをインストールするだけで簡単に開始可能

Step1

インスタンスリソースにエージェントをデプロイ

EC2はSSMを、EKSはEKSアドオンを使用して簡単にデプロイ

Step2

ワークロードインサイトを確認

エージェントが自動収集したワークロードのパフォーマンスを確認

Step3

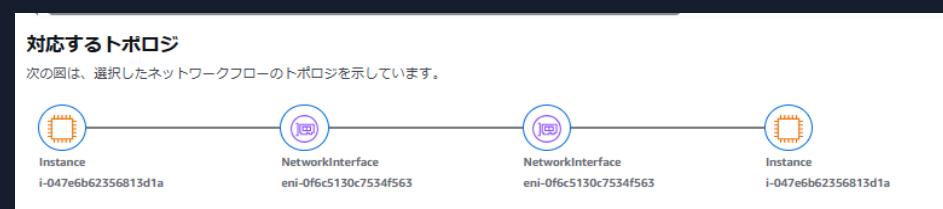
モニターを作成

特定のワークロードのネットワーク品質を継続的にモニタリング

Step4

Historical Explorer から再送信の発生を確認

ネットワークフローのトポロジを表示し、トラブルシューティング

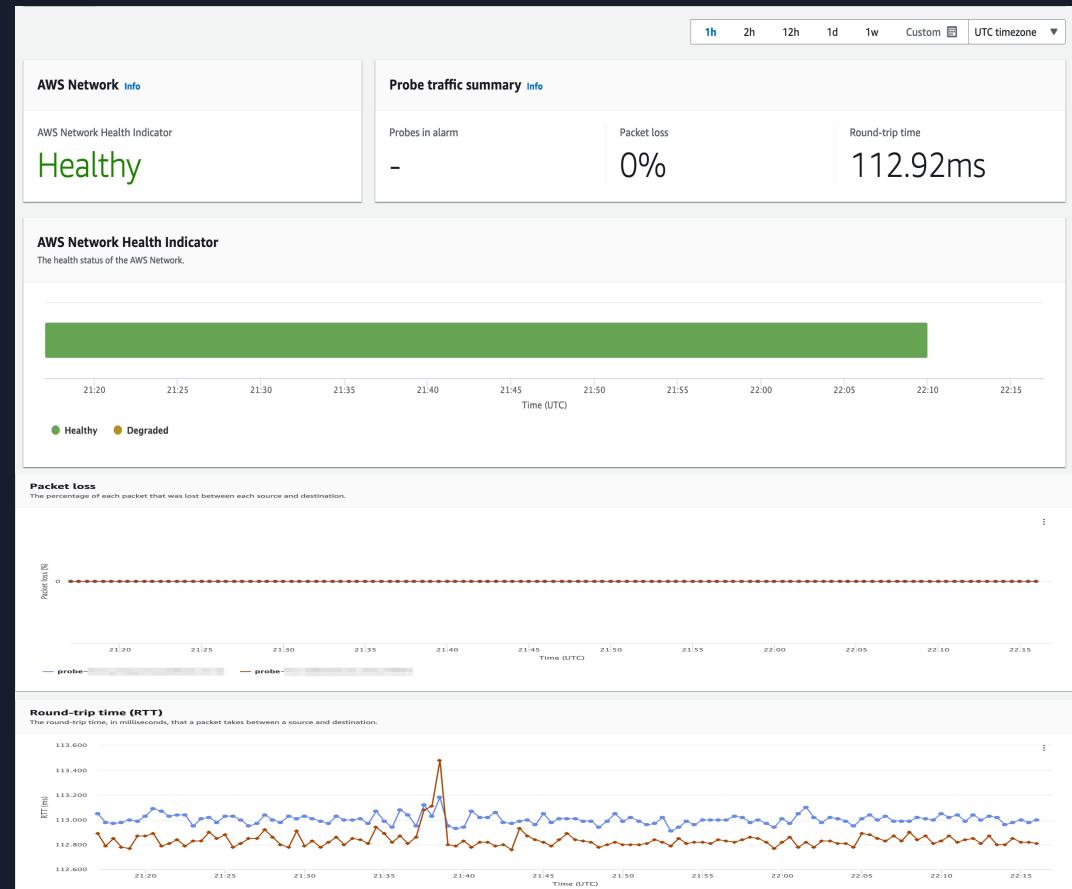


# Network Synthetic Monitor とは

# Network Synthetic Monitor とは

DirectConnect や VPN 等を経由した AWS とオンプレミス間のネットワーク品質を監視

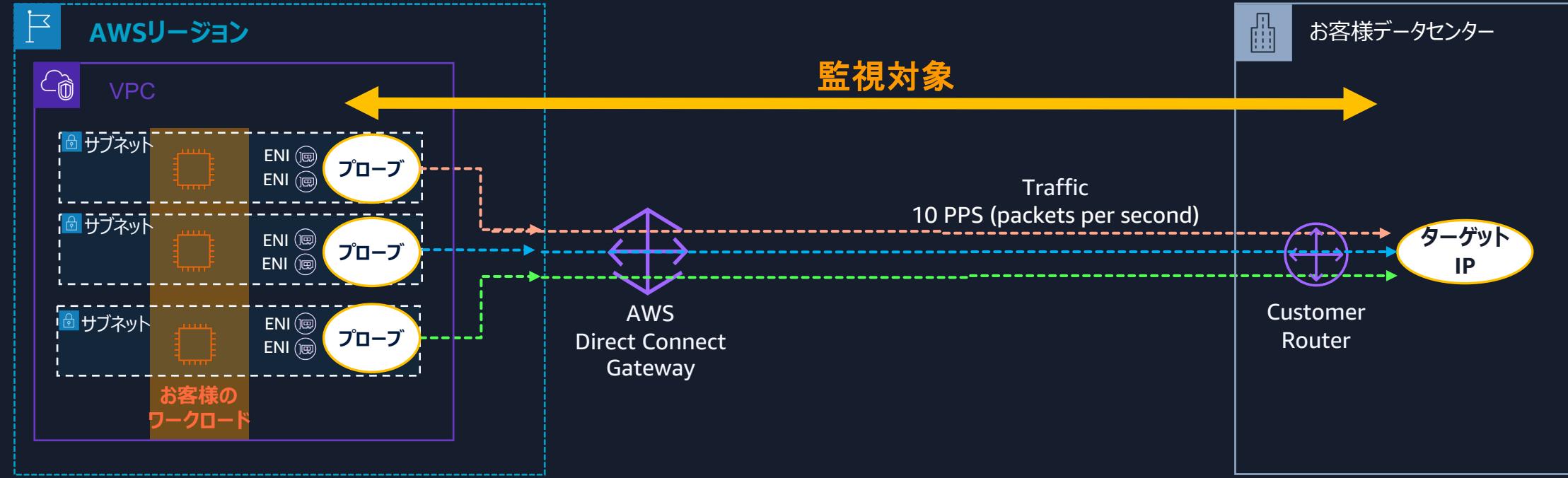
- パケットロスとレイテンシー(RTT) メトリクスをリアルタイムで可視化
- AWS 管理のネットワークの正常性を示す AWS Network Health Indicator (NHI) を使用して問題を切り分け
- エージェントレスで、VPC のサブネットとエンドポイントを指定するだけで開始可能
- サブネットに配置したプローブにより、10 PPS (packets per second) でトラフィックが生成
- TCP/ICMPプロトコルをサポート



AWS とオンプレミス間のネットワークの状態を監視、通知、可視化

AWS管理のネットワークの問題かどうかの切り分けが可能に

# Network Synthetic Monitor のアーキテクチャ



- 送信元サブネット、宛先IP、アドレスファミリ (IPv4/IPv6)、プロトコル (ICMP/TCP)、パケットサイズのすべての組み合わせに対して、**プローブ(AWS ホストリソースからオンプレミスの宛先 IP アドレスに送信されるトラフィック)** を作成
- エンドツーエンドの監視を行うには、ミッションクリティカルなワークロードが設定されている各AZに**プローブ**を配備し、ワークロードの通信先であるオンプレミス内の宛先(IP Address)をターゲットに設定

# Network Synthetic Monitor の料金

Network Synthetic Monitor は、以下2つの料金がかかります。

## Amazon CloudWatch Network Synthetic Monitor 料金

### ①監視リソース(プローブ)

4つのプローブあたり  
**0.11 USD/時間**

※「監視リソース」は  
稼働中のエージェント毎に  
1つ計上されるリソース

### ②CloudWatchメトリクス

最初の10,000  
メトリクスまで  
**0.30 USD/metric/月**

- ※1つのプローブは  
3つのメトリクスを発行
- ラウンドトリップ時間
  - パケットロス
  - Network Health Indicator (NHI)

### (関連サービス利用料)

Amazon CloudWatch  
(アラーム, …)

+

AWS Lambda

+

:

# Network Synthetic Monitor を使ってにみよう

ソースとなるAWS側のサブネットとオンプレミス側の宛先IPを指定するだけで簡単に開始可能

Step1

## モニターを作成

AWS側のソースとオンプレミス側の宛先を指定し、プローブを作成

Step2

## 自動生成されたダッシュボードを確認

プローブにより収集されたメトリクスを確認および継続的に監視

Step3

## 問題のトラブルシューティング

問題の原因のトラブルシューティングに役立てる

### ソースと宛先を選択

ここで指定した送信元 VPC サブネットと宛先 IP アドレス間のメッシュネットワークを監視します。

#### AWS ネットワークソース 情報

##### サブネット

1つ以上のサブネットを追加します。

##### サブネットを選択

subnet-05c24df1f99aed2b9

オーナー: 157823861179 アベイラビリティーゾーン: ap-northeast-1d 使用可能な IP アドレス: 4079 CIDR: 172.31.16.0/20

subnet-0b414e9bfd3da75fb

オーナー: 157823861179 アベイラビリティーゾーン: ap-northeast-1c 使用可能な IP アドレス: 4086 CIDR: 172.31.0.0/20

subnet-0a8ded6db50f17eb0

オーナー: 157823861179 アベイラビリティーゾーン: ap-northeast-1a 使用可能な IP アドレス: 4069 CIDR: 172.31.32.0/20

#### 送信先1 情報

##### IP アドレス

IPv4 アドレスまたは IPv6 アドレスを入力します。

172.31.21.0

##### ▶ 詳細設定

##### 宛先を追加

あと 3 件の宛先を追加できます。

# Internet Monitor とは

# Internet Monitor

アプリケーションとエンドユーザー間のインターネットパフォーマンスを監視するマネージメントサービス

- 継続的にインターネット通信（送受信）時の可用性とレイテンシ( RTT )などのメトリクスを収集
- インターネット天気図で過去24時間以内に発生したインターネットの問題を世界地図で可視化
- アプリケーションを利用するクライアントのインターネットパスを監視し、パフォーマンスを可視化  
監視対象：VPC/Workspaces/CloudFront/NLB
- クライアントやアプリケーションへの影響はなし
- パフォーマンスを最適化する方法を提案
- 測定値は CloudWatch Logs に発行されるので詳細の追跡や分析が可能



過去 24 時間に世界中で発生した主要なインターネットイベントの概要

インターネット経由の通信状態を一か所で監視、可視化、通知

問題の原因がインターネット、AWS ネットワーク、またはエンドユーザー・アプリケーションか、切り分け可能

# Internet Monitor を使ってみよう

監視対象とするリソースを指定するだけで簡単に開始可能

Step0

## インターネット天気図を確認

インターネットの可用性とパフォーマンス問題を確認

Step1

## モニターを作成

監視対象となるリースを指定

Step2

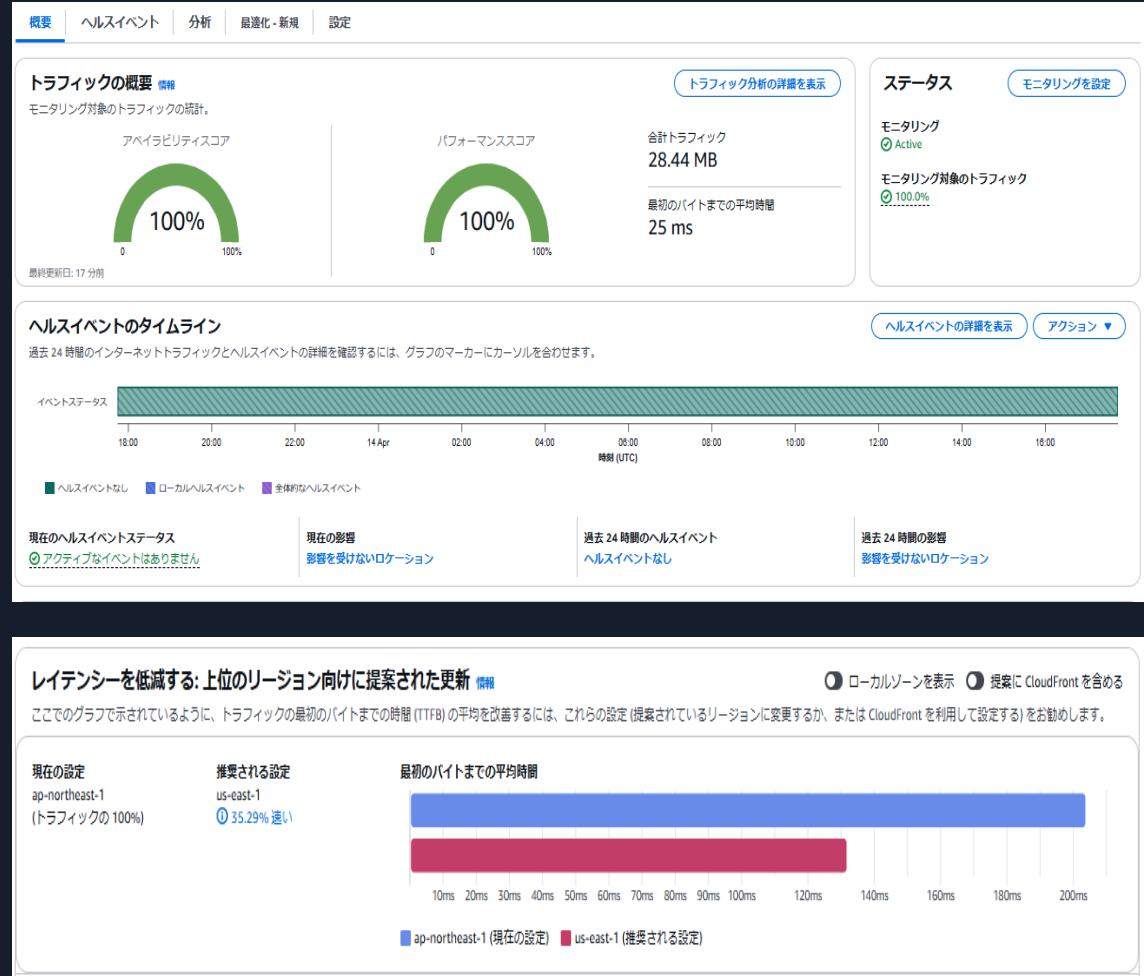
## ダッシュボードを確認

インターネットトラフィックの分析結果を確認

Step3

## レイテンシーを改善するための提案を取得

クライアントが最適なパフォーマンスを得ることのできる  
提案を確認



# Internet Monitor の料金

Internet  
Monitor

Internet Monitor は、以下3つの料金がかかります。

## Amazon CloudWatch Internet Monitor 料金

### ①モニタリング対象リソース

監視対象リソースあたり  
**0.01 USD/時間**

※「監視対象リソース」は  
VPC, CloudFrontディストリビューション,  
WorkSpaces ディレクトリなどの  
モニタリングするアプリケーションリソース

### ②モニタリング対象都市ネットワーク

10,000 個の  
モニタリング対象の  
都市ネットワークあたり  
**0.74 USD/時間**

※都市ネットワークとは、  
クライアントがアプリケーションにアクセスする都市  
およびインターネットサービスプロバイダーなどの  
ネットワーク (ASN) のこと

### ③CloudWatch Logs の料金

収集するデータ量に対して  
**0.76 USD/GB**  
クエリされたデータ量に対して  
**0.0076 USD/GB**

※モニターは、トラフィック量上位の  
都市ネットワーク (最大 500) の診断ログを  
5 分ごとに CloudWatch Logs に生成

※収集はスタンダード料金  
アーカイブ保存やLive Tail分析など  
使用する場合は別途料金がかかります

+

+

※2025/4 時点での東京リージョンの場合

<https://aws.amazon.com/jp/cloudwatch/pricing/>



# まとめ

# まとめ - ネットワークモニタリングを提供するAWSサービス

それぞれモニタリング対象が異なる 3つの サービス を提供  
目的に応じて 使い分け や 組み合わせ

ネットワークのモニタリング  
マネージドサービス

CloudWatch  
Network Flow Monitor

CloudWatch  
Network Synthetic Monitor

CloudWatch  
Internet Monitor

AWSワークLOADとAWSリソース間における  
ネットワーク品質の  
継続的モニタリング

AWSリージョンとオンプレミス間における  
ネットワーク品質の  
継続的モニタリング

アプリケーションとエンドユーザー間における  
インターネット品質の  
継続的モニタリング

春の Observability 祭り 2025 ~進化する Amazon CloudWatch 基礎から最新機能まで完全解説~

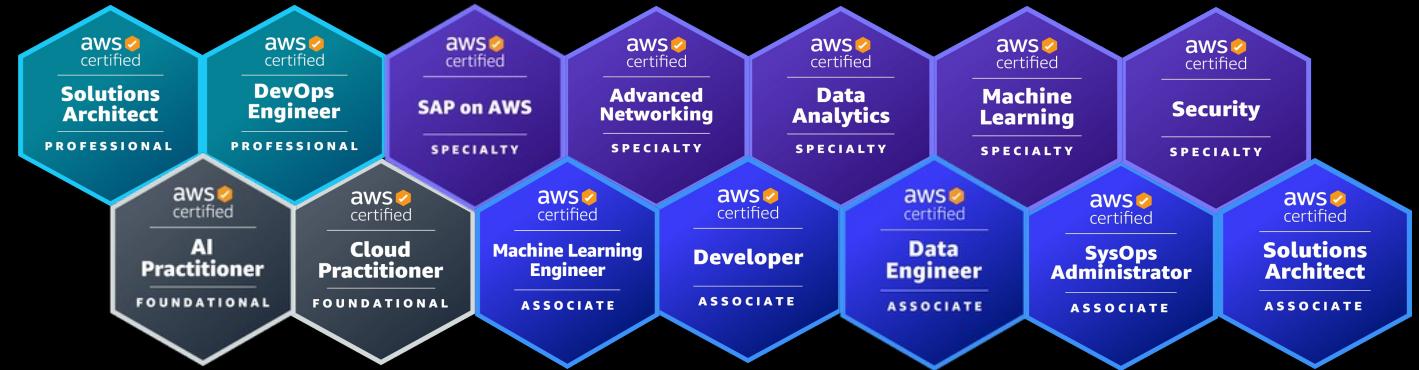
# Amazon CloudWatch で始める エンドユーザーエクスペリエンスのモニタリング

日平 大樹

アマゾンウェブサービスジャパン合同会社  
テクニカルアカウントマネージャー



# 自己紹介



氏名

日平 大樹 (ひびら たいき)

役職

エンタープライズサポートをご契約頂いたお客様を支援する  
テクニカルアカウントマネージャー

好きな  
サービス

Amazon CloudWatch  
Amazon EC2

**“ Nearly 70% of consumers admit that page speed influences their likeliness to buy.**

Unbounce (2019), “Think Fast: The Page Speed Report, Stats & Trends for Marketers,”

<https://unbounce.com/page-speed-report>.

# エンドユーザー体験モニタリングの重要性



サービスやアプリの健全性をチェックする



アプリケーションを元の状態に回復する



トラブルの原因を調査する



ユーザ行動を分析する



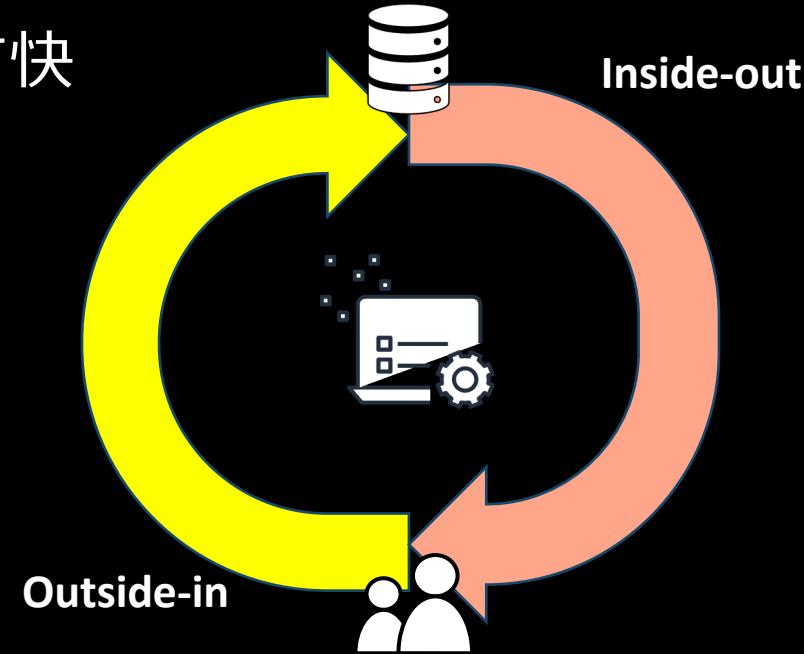
キャパシティを分析する

# フルスタックのオブザーバビリティ戦略

**エンドユーザー** にとって快適か？

例えば

- 表示時間
- アクション応答時間
- 購入失敗
- JavaScript エラー



**バックエンドアプリケーション** がどのような状態にあるか？

例えば

- スロークエリ
- インテグレーションヘルス
- コンテナ再起動
- レイテンシー

ワークフローの完全な把握には、エンドユーザーの体験を含む、すべての層を紐付け可視化することが重要

# ユーザー体験のモニタリング方法

## Real User Monitoring

※ Passive monitoring の一種

- 実ユーザーアクセスの情報を取得して計測
- 中長期のユーザー傾向把握用途に好適



Amazon CloudWatch RUM

## Synthetic Monitoring

※ Active monitoring の一種

- 定常的に一定間隔で計測プログラムがアクセスし、パフォーマンスを計測
- 短期のパフォーマンス計測用途に好適
- 一般に「外形監視」と訳される

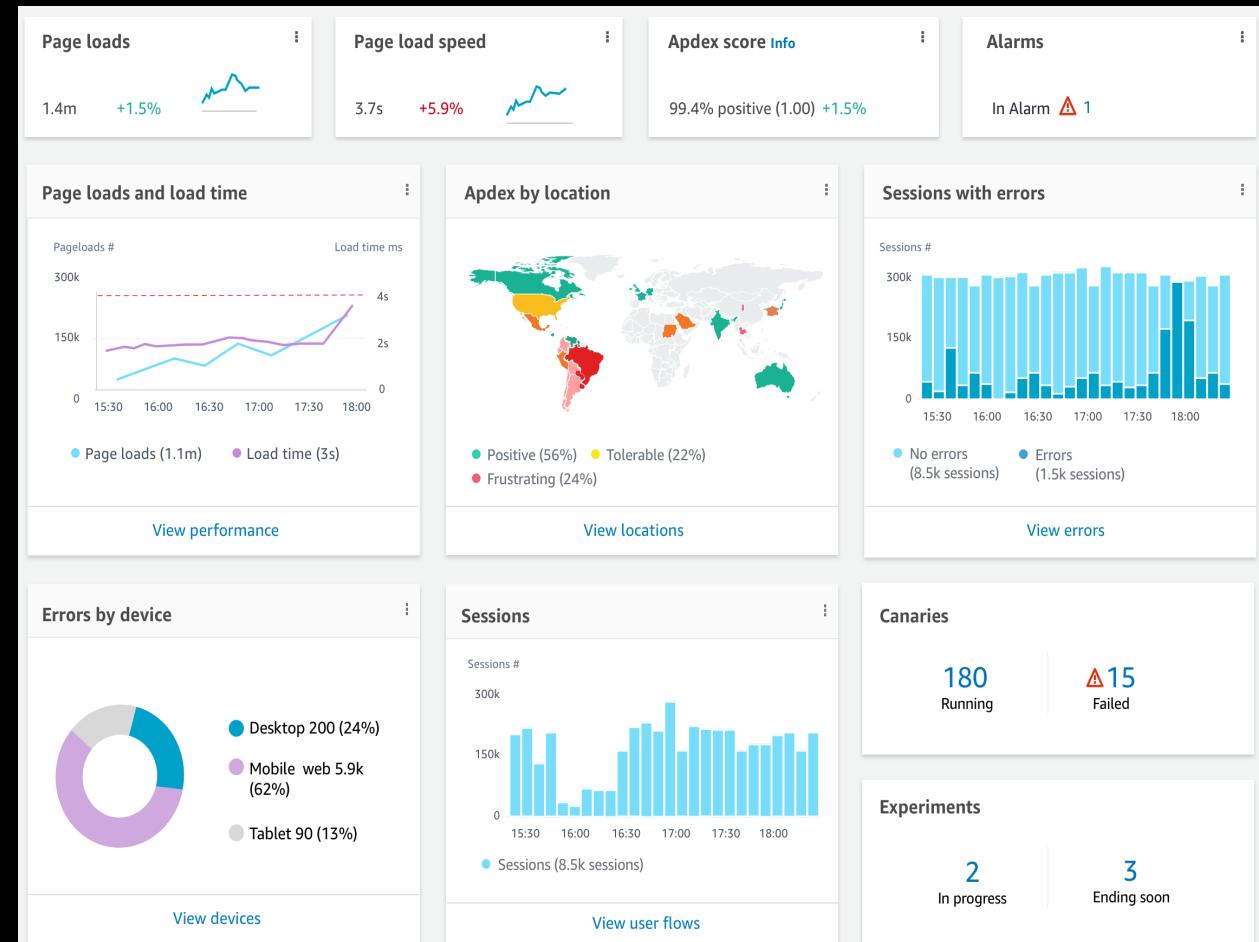


Amazon CloudWatch Synthetics

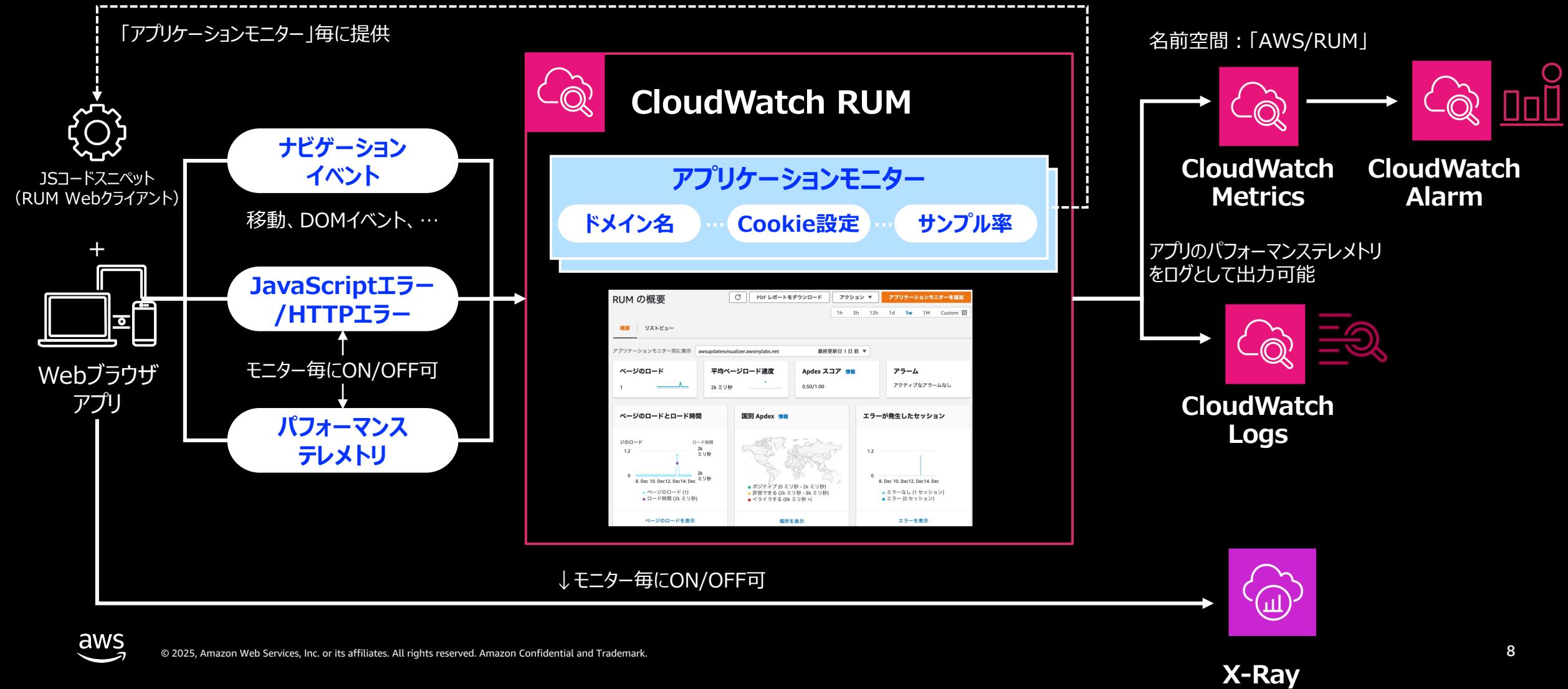
# Amazon CloudWatch RUM (Real User Monitoring)

アプリのパフォーマンスに関するクライアントサイドのデータをリアルタイムで取得し、ユーザー体験を最適化

- リアルユーザーのパフォーマンス、ブラウザやデバイスの種類、位置、ネットワークの接続性の問題などを把握可能
- ページの読み込み順序や JavaScript / HTTP レスポンスのエラーなど、パフォーマンス問題に関する情報を可視化
- 同じ問題の影響下にあるユーザーセッション数から優先順位を付けることも容易
- CloudWatch Logs や X-Ray と連携も可能



# Amazon CloudWatch RUM の動作イメージ



# Amazon CloudWatch RUM の導入

The screenshot shows the 'Code Snippet' configuration step of the CloudWatch RUM setup wizard. It includes:

- A navigation bar with steps: Step 1 (Application Monitor added), Step 2 (Code Snippet selected), and Step 3 (Code Snippet).
- A note about the code snippet: "サイトから CloudWatch RUM サービスにデータを送信するには、アプリケーションに CloudWatch RUM ウェブクライアントをインストールする必要があります。" (You must install the CloudWatch RUM web client in your application to send data to the service.)
- An information box: "① Javascript コードスニペットによってダウンロードおよび設定されるウェブクライアントは、Cookie (または同様の技術) を使用してエンドユーザーデータの収集に役立てます。コードスニペットを使用する前に、Amazon CloudWatch RUM のデータプライバシーとデータ保護をご覧ください。 詳細はこちら [リンク]" (The web client installed via the code snippet uses cookies (or similar technologies) to collect end-user data. Please review the data privacy and data protection for Amazon CloudWatch RUM before using the code snippet.)
- A sample code section with tabs for HTML, TypeScript, and JavaScript. The HTML tab is selected, showing the following code:

```
1 <script>
2   (function(n,i,v,r,s,c,x,z){x>window.AwsRumClient={q:[],n:n,i:i,v:v,r:r,c:c};window[n]=
3     'cwr',
4     '29ad76ec-2e13-4ec4-ab06-0906ea2bb952',
5     '1.0.0',
6     'us-west-2',
7     'https://client.rum.us-east-1.amazonaws.com/1.19.0/cwr.js',
8     {
9       sessionSampleRate: 1,
10      identityPoolId: "us-west-2:088e5a07-f306-4f1d-a459-060fc54de774",
11      endpoint: "https://dataplane.rum.us-west-2.amazonaws.com",
12      telemetries: ["performance", "errors", "http"],
13      allowCookies: true,
14      enableXRay: false
15    }
16  );
17 </script>
```

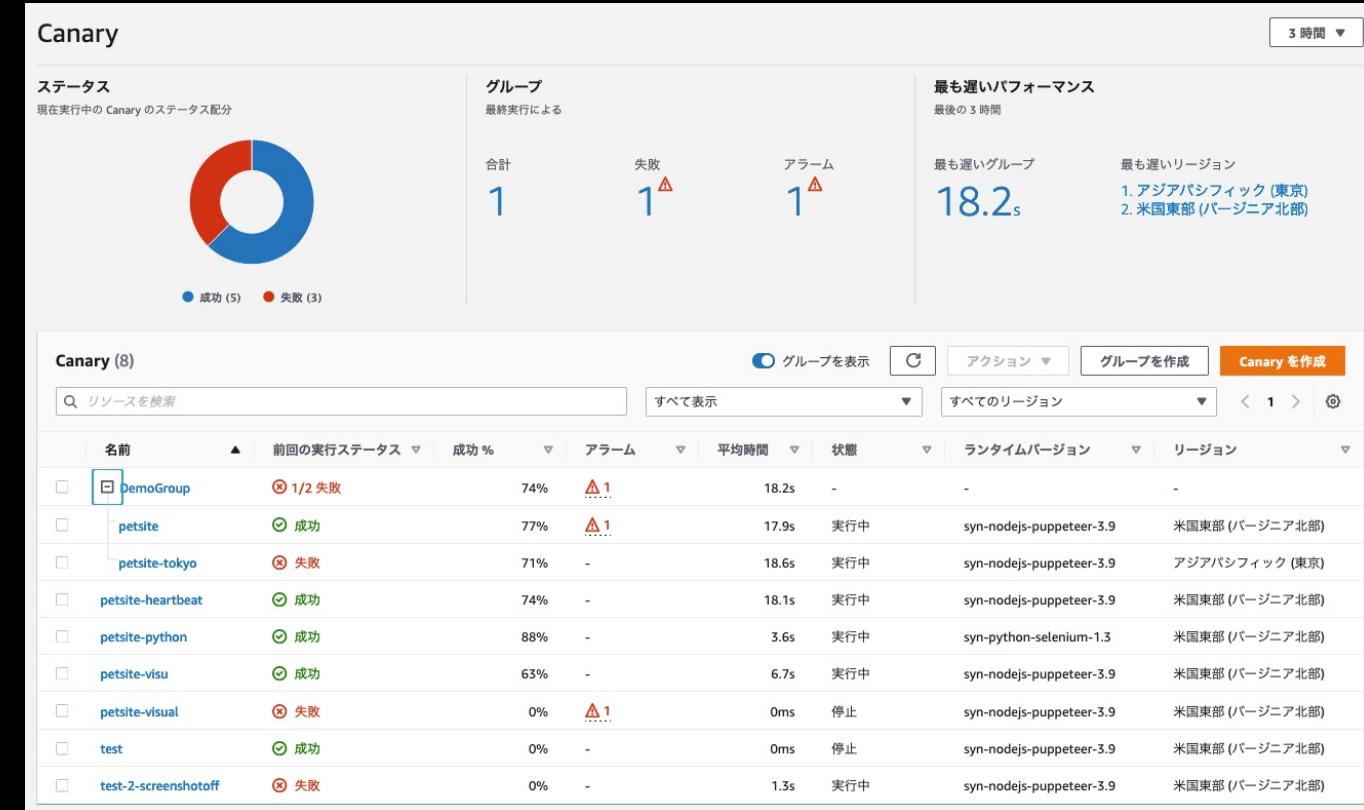
At the bottom, it says: "詳細については、以下を参照してください。 Amazon CloudWatch RUM web client [リンク]" (For more details, please refer to the Amazon CloudWatch RUM web client.)

- 導入はアプリケーションモニターを作成し、**コードスニペットをアプリケーションに挿入するだけ**
- コードスニペットには TypeScript、JavaScript、HTML のサンプルコードが用意されている
- (注) CloudWatch RUM が生成するトレースをサーバー側トレースと紐付ける場合はウェブクライアントの [addXRayTraceIdHeader](#) オプションで true を設定してください。

# Amazon CloudWatch Synthetics

## 実際のユーザートラフィックがなくてもエンドポイントを継続的にテスト

- CloudWatch Synthetics は、Web アプリケーションと API を簡単に監視できるようにするサービス
- 実際のユーザートラフィックがなくてもエンドポイントを継続的にテストが可能
- 設定したしきい値に基づいてエラーがあればアラート
- X-Ray との連携も可能



# Amazon CloudWatch Synthetics Canary 全体構成イメージ



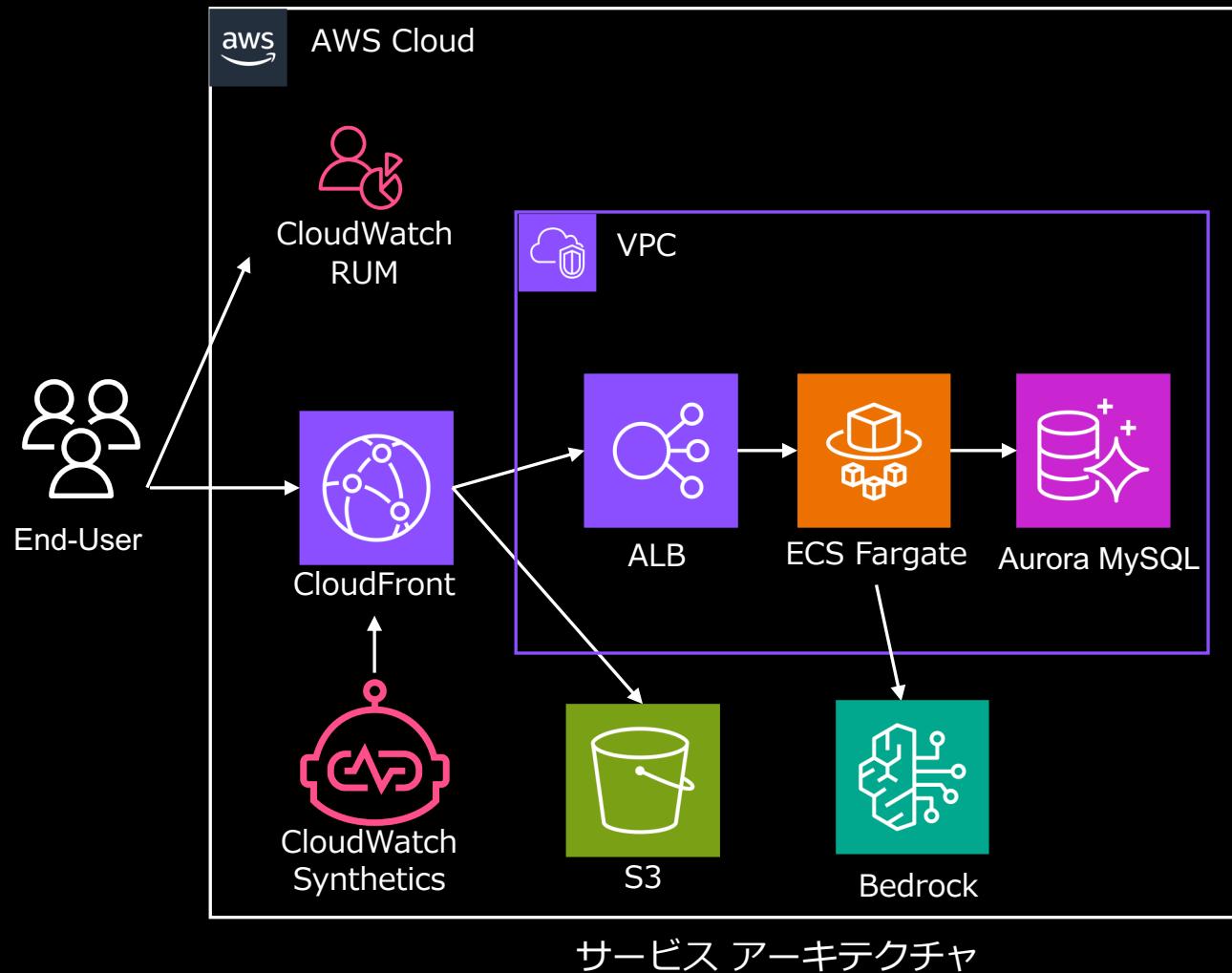
# Amazon CloudWatch Synthetics Canary 作成



- Blueprint が用意されており、  
基本的なユースケースにおいては、**ノーコードでテストを作成可能**
  - ハートビートのモニタリング
  - API Canary
  - リンク切れチェッカー
  - Canary レコーダー
  - GUI ワークフロービルダー
  - ビジュアルモニタリング
- 独自のスクリプトを利用することもできるため、**既存のテスト資産も利用可能**

# ユースケース

- チームではリリースのテスト効率を上げるために、CloudWatch Synthetics Canary を利用し、ユーザーの擬似操作テストを自動化しています。
- また、実際のユーザーの体験を把握するために CloudWatch RUM を導入しています。
- 今回、新しい機能をリリースしたところ、CloudWatch RUM で Javascript エラーが検知されました。また、CloudWatch Synthetics Canary の失敗を検知したため調査を開始しました。



# Amazon CloudWatch RUM の導入

```
ts 01.rum.ts M X
web > plugins > ts 01.rum.ts > [o] default > [o] defineNuxtPlugin() callback > [o] config > [o] sessionEventLimit
1 import { AwsRum, type AwsRumConfig } from 'aws-rum-web';
2
3 export default defineNuxtPlugin(_nuxtApp) => {
4   const runtimeConfig = useRuntimeConfig();
5   if (runtimeConfig.public.rumEnabled) {
6     const config: AwsRumConfig = [
7       sessionSampleRate: 1,
8       identityPoolId: runtimeConfig.public.rumIdentityPoolId,
9       endpoint: runtimeConfig.public.rumEndpoint,
10      telemetries: ["performance", "errors", { 'http': { addXRayTraceIdHeader: true, recordAllRequests: true, } }],
11      allowCookies: true,
12      enableXRay: true,
13      sessionEventLimit: 0,
14      releaseId: runtimeConfig.public.rumReleaseId
15    ];
16
17    const APPLICATION_ID: string = runtimeConfig.public.rumApplicationId;
18    const APPLICATION_VERSION: string = '1.0.0';
19    const APPLICATION_REGION: string = runtimeConfig.public.rumApplicationRegion;
20
21    const awsRum: AwsRum = new AwsRum(
22      APPLICATION_ID,
23      APPLICATION_VERSION,
24      APPLICATION_REGION,
25      config
26    );
27
28    return {
29      provide: {
30        awsRum: awsRum
31      }
32    };
33  };
34}
```

今回、フロントは Nuxt で作成したシングルページアプリケーションです。  
RUM Client 導入は Package に “aws-rum-web” を追加し、  
マネジメントコンソールで表示された TypeScript コードスニペットのファイルを  
plugins 以下に作成するだけです。

```
TS authUser.ts M ●
web > composables > TS authUser.ts > ...
1 // アプリケーションの認証情報取得
2 export const useAuthUser = async () => {
3   const data: any = await useNuxtApp().$api('/api/v1/auth/validate_token', {
4     method: 'GET',
5   });
6   if (data.success) {
7     data.user = data.data
8     useNuxtApp().$auth.setData(data)
9     useNuxtApp().$awsRum.addSessionAttributes({
10       uid: data.user.uid
11     });
12
13   // TypeScript ではなく、HTMLへの埋め込みの場合はこちらを利用します
14   // cwr('addSessionAttributes', {
15   //   uid: data.user.uid
16   // })
17 }
```

RUM イベントのセッションにユーザーを識別できるように  
ログイン情報から User ID (uid) をRUM セッション属性に追加しました。



# Amazon CloudWatch RUM (Errors)

The screenshot illustrates the Amazon CloudWatch RUM Errors interface, highlighting how it tracks user errors across different sessions and devices.

**Left Panel (Main View):**

- CloudWatch RUM Overview:** Shows a bar chart of errors over time (13:00 to 14:15) and a table of errors by message ("Cannot read properties of undefined (reading 'id')").
- Filtering:** A red box highlights the "Errors" tab under "Pages" and the search/filter bar at the top.
- Session Details:** A red box highlights the session ID "41f39d8b-fd6e-489d-9f0c-5c4cb3b4ec83" in the "Sessions with selected error" table.

**Middle Panel (Sessions with Selected Error):**

Session ID	Selected error count	Total error count	Duration	Browser	Device
41f39d8b-fd6e-489d-9f0c-5c4cb3b4ec83	2	3	0:20	Chrome Headless v126.0.6478.126	Desktop Linux, x86_64
53d592ca-3a9f-4a31-93f0-3f98e7dcc777	2	3	0:19	Chrome Headless v126.0.6478.126	Desktop Linux, x86_64
6c392620-2424-4a32-a4e9-773f26bc456f	2	3	0:25	Chrome Headless v126.0.6478.126	Desktop Linux, x86_64
6d4b62c8-cd7e-41a1-b718-1850dcfb714c	2	3	0:25	Chrome Headless v126.0.6478.126	Desktop Linux, x86_64
8e7a6f39-03eb-4981-81f8-14449428d8fc	2	3	0:20	Chrome Headless v126.0.6478.126	Desktop Linux, x86_64

**Bottom Panel (Event Detail):**

An event detail view for "com.amazon.rum.js\_error\_event" is shown, with a red box highlighting the "uid: thibra@amazon.co.jp" metadata entry. The event occurred on Mar 17, 2025, at 1:53:27 PM GMT+9.

**Text Overlay:**

「uid」を記録しているため、エラーが起きたセッションのメタデータを確認すると、どのユーザーでエラーが起っているかを把握できる

# Amazon CloudWatch RUM (Errors)

▼ Latest raw event

```
{ "event_timestamp": 1743048645000, "event_type": "com.amazon.rum.js_error_event", "event_id": "79cd4fa0-24fd-4c8e-89a6-c953d11e7aad", "event_version": "1.0.0", "log_stream": "2025-3-27T13", "application_id": "845cf90d-d5ca-48ab-bfbf-e0e3c891f9fa", "application_version": "1.0.0", "metadata": { "version": "1.0.0", "browserLanguage": "ja", "browserName": "Chrome", "browserVersion": "134.0.0.0", "osName": "Mac OS", "osVersion": "10.15.7", "deviceType": "desktop", "platformType": "web", "pageId": "", "interaction": "0", "title": "ピクチャーズ", "domain": "d2q37f0j8y2l.cloudfront.net", "aws_client": "arw-module", "aws_clientVersion": "1.21.0", "aws_releaseId": "4d33188", "uid": "tibiro@amazon.co.jp", "countryCode": "JP", "subdivisionCode": "13" }, "user_details": { "userId": "1503Sead9-9b6f-4c7d-ace4-220793298283", "sessionId": "dfe57cd8-e376-4169-b044-87fd7a3de2ce" }, "event_details": { "version": "1.0.0", "type": "TypeError", "message": "Cannot read properties of undefined (reading 'id')", "filename": null, "lineno": null, "colno": null, "stack": "TypeError: Cannot read properties of undefined (reading 'id')\n at https://d2q37f0j8y2l.cloudfront.net/_nuxt/CUT7zfTe.js:1:241\n at Object.r\n(https://d2q37f0j8y2l.cloudfront.net/_nuxt/CTTSocTQ.js:1:413961)\n at Proxy.<anonymous>\n(https://d2q37f0j8y2l.cloudfront.net/_nuxt/B9MNzK8z.js:1:4172)\n at Xo\n(https://d2q37f0j8y2l.cloudfront.net/_nuxt/CTTSocTQ.js:15:33458)\n at Pi.W\n(https://d2q37f0j8y2l.cloudfront.net/_nuxt/CTTSocTQ.js:15:24630)\n at Pi.run\n(https://d2q37f0j8y2l.cloudfront.net/_nuxt/CTTSocTQ.js:10:1808)\n at oe\n(https://d2q37f0j8y2l.cloudfront.net/_nuxt/CTTSocTQ.js:15:24272)\n at D\n(https://d2q37f0j8y2l.cloudfront.net/_nuxt/CTTSocTQ.js:15:23939)\n at p\n(https://d2q37f0j8y2l.cloudfront.net/_nuxt/CTTSocTQ.js:15:20793)\n at Be\n(https://d2q37f0j8y2l.cloudfront.net/_nuxt/CTTSocTQ.js:15:26347)", "unminifiedStack": "TypeError: Cannot read properties of undefined (reading 'id')\n at ./././.pages/index.vue:1:2:0\n at Object.r\n(./././.node_modules/vue/runtime-core/dist/runtime-core.esm-bundler.js:692:15)\n at Proxy.<anonymous>\n(./././.node_modules/vuetify/lib/components/VGrid/VCol.mjs:125:7)\n at Xo\n(./././.node_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:6502:15)\n at renderComponentRoot\n(./././.node_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:5397:25)\n at Pi.run\n(./././.node_modules/@vue/reactivity/dist/reactivity.esm-bundler.js:225:18)\n at oe\n(./././.node_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:5258:17)\n at updateComponent\n(./././.node_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:5193:22)\n at n1\n(./././.node_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:4701:12)\n at n1\n(./././.node_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:5598:10)" } }
```

"unminifiedStack": "  
TypeError: Cannot read properties of undefined (reading 'id')  
at (./././.pages/index.vue:172:0)  
at Object.r (./././.node\_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:692:15)  
at Proxy.<anonymous> (./././.node\_modules/vuetify/lib/components/VGrid/VCol.mjs:125:7)  
at Xo (./././.node\_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:6502:15)  
at renderComponentRoot (./././.node\_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:5397:25)  
at Pi.run (./././.node\_modules/@vue/reactivity/dist/reactivity.esm-bundler.js:225:18)  
at oe (./././.node\_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:5258:17)  
at updateComponent (./././.node\_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:5193:22)  
at n1 (./././.node\_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:4701:12)  
at n1 (./././.node\_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:5598:10)"

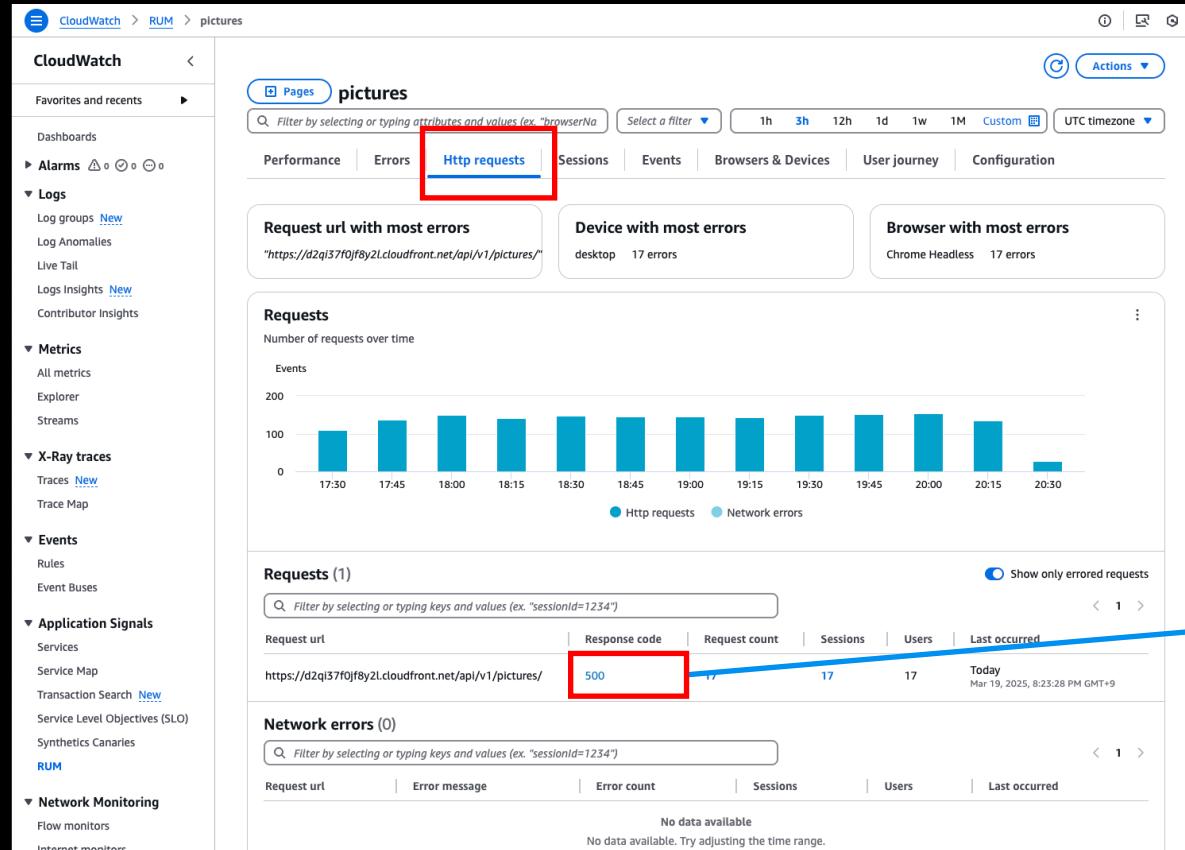
2025年3月18日にリリースされたソースマップ対応を利用することで、圧縮前のスタックトレースを確認可能  
実際のソースコードと見比べることで、エラー発生箇所を素早く把握できる

```
# pages/index.vue より抜粋  
<v-col v-for="(picture, index) in pictures" :key="picture.id">  
<v-card link :to=`/pictures/${picture.id}`>  
...  
</v-card>  
</v-col>
```

picture がundefined  
なため エラーになつて  
いると分かった



# Amazon CloudWatch RUM (Http requests)



Request url	Error message	Type	Error count	Sessions	Users	Last occurred
<a href="https://d2qj37f0jf8y2l.cloudfront.net/api/v1/pictures/">https://d2qj37f0jf8y2l.cloudfront.net/api/v1/pictures/</a>	500	HTTP	18	18	18	Today Mar 19, 2025, 8:33:26 PM GMT+9

**Error details**

Request url: https://d2qj37f0jf8y2l.cloudfront.net/api/v1/pictures/ | Error message: 500: | Type: HTTP | Error count: 18 | Sessions: 18 | Users: 18 | Last occurred: Today Mar 19, 2025, 8:33:26 PM GMT+9

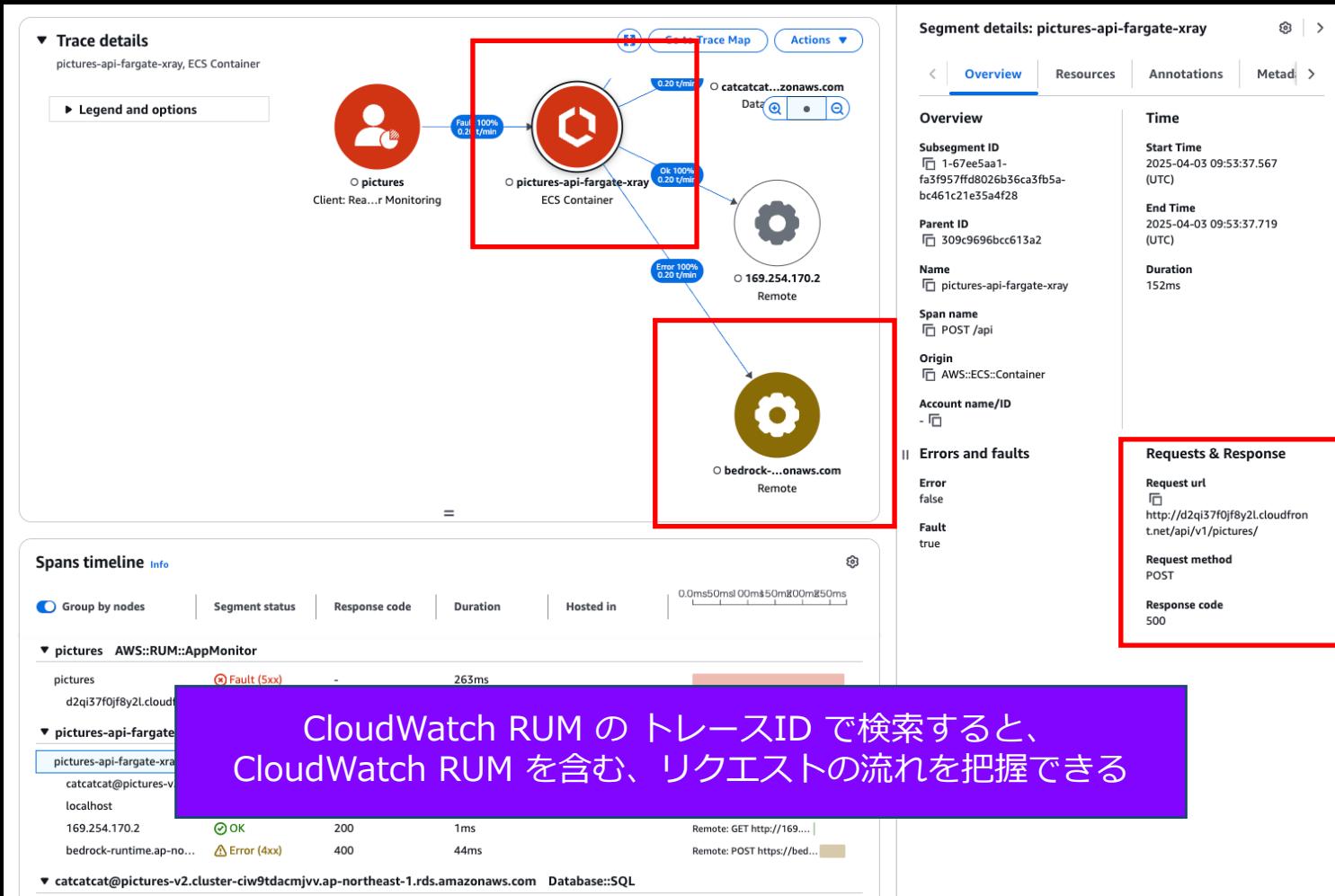
**Latest raw event**

```
{ "event_timestamp": 1742384006000, "event_type": "com.amazon.rum.http_event", "event_id": "98bc550d-9990-4908-a8ad-b5868b749e5b", "event_version": "1.0.0", "log_stream": "2025-3-19T20", "application_id": "845cf90d-d5ca-48ab-bfbf-e0e3c891f9fa", "application_version": "1.0.0", "metadata": { "version": "1.0.0", "browserLanguage": "en-US", "browserName": "Chrome Headless", "browserVersion": "126.0.6478.126", "osName": "Linux", "osVersion": "x86_64", "deviceType": "desktop", "platformType": "web", "pageId": "/", "parentPageId": "/auth/signin", "interaction": "2", "title": "ピクチャーズ", "domain": "d2qj37f0jf8y2l.cloudfront.net", "aws_client": "arw-module", "aws_client_version": "1.21.0", "uid": "thibiro@amazon.co.jp", "countryCode": "JP", "subdivisionCode": "13" }, "user_details": { "userId": "9e923e45-b221-4886-8374-beb2b20bcbdd", "sessionId": "8b22407f-94fe-4241-9833-2e5e33164634" }, "event_details": { "version": "1.0.0", "traceId": "1-67daab85-03bf8bc831e51c8981b8ee92", "segmentId": "79eh7zed2d37n511", "request": { "method": "POST", "url": "https://d2qj37f0jf8y2l.cloudfront.net/api/v1/pictures/" } }
```

Http requests タブより、APIサーバーとの通信でエラーが発生していることを把握  
Http request には トレースID が付与されており、X-Ray から詳細を確認できる



# Amazon CloudWatch RUM × AWS X-Ray



# Amazon CloudWatch Synthetics

CloudWatch > Synthetics Canaries > pictures-internal-test...

Last updated: 7:34 PM. Auto-refreshes every minute.

pictures-internal-test Info

Summary  
Latest run: Failed

Issues in selected time range: 2 issue(s)

Success % in selected time range: 88%

State: Running

Search for resources View in Service Map Actions 1 minute

Availability | Monitoring | Configuration | Groups | Tags

Issues (2) In selected time range

No test result returned. Connection timed out... April 11, 2025 19:30

No test result returned. Connection timed out... April 11, 2025 19:20

Canary runs Learn more View Canary troubleshooting documentation for additional information.

Each point represents a canary run. Click each data point for details. 30m 1h 3h 12h 3d 1w Custom Local timezone

Passed Failed

Screenshots Logs Traces

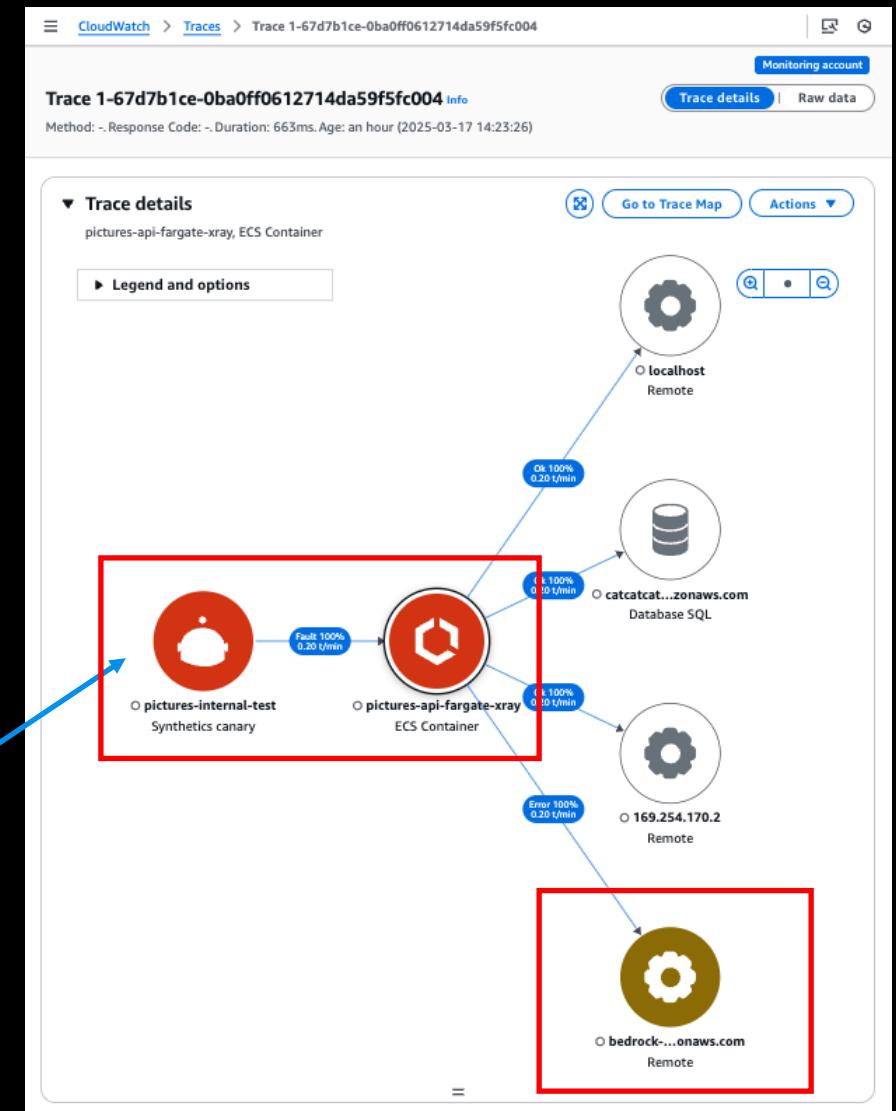
Show errors only Go to trace map

Traces (26)

Trace ID	Trace status	Response code	Response time	URL address
1-67f8ef13-2cc15ab8b0959d7777fcb7e7	Error	500	0.69s	<a href="http://d2qj37f0jf8y2l.cloudfront.net">http://d2qj37f0jf8y2l.cloudfront.net</a>
1-67f8eff-2c4dcbcc1b69f26119cfb837	OK	-	0.07s	-

Traces

Synthetics Canary テストの失敗を検知  
「Traces」タブの「GO to trace map」からエラーリクエストの X-Ray を確認することが可能



# AWS X-Ray で Dive Deep

**指定したモデルが誤っているエラーが発生**

The screenshot shows the AWS X-Ray interface. On the left, the Spans timeline displays a sequence of events from a CloudWatch Synthetics test. A specific event in the middle of the timeline is highlighted with a red box, showing a POST request to `https://bedrock-runtime.ap-northeast-1.amazonaws.com:443/model/us.anthropic.claude-3-7-sonnet-20250219-v1:0/converse` with a status code of 400 (Error). This event is connected by a blue arrow to the right-hand Segment details panel.

**Segment details: pictures-api-fargate-xray**

**Exceptions**

Path	message	type	Cause
/usr/local/bundle/ruby/3.4.0	The provided model identifier is invalid.	Aws::BedrockRuntime::Errors::ValidationException	-

**Logs**

```
# :@log :@timestamp :@message
# 1 466398113874:[ecs/pictures-api] 2025-03-17T05:23:26.6932 [1-67d7b1ce-0ba0ff0612714d059f5fc004] Started POST "/api/v1/pictures/" for 3.172.52.100 at 2025-03-17 05:23:26 +0000
# 2 466398113874:[ecs/pictures-api] 2025-03-17T05:23:26.6942 [1-67d7b1ce-0ba0ff0612714d059f5fc004] Processing by Api::V1::PicturesController#create as */
# 3 466398113874:[ecs/pictures-api] 2025-03-17T05:23:26.6952 [1-67d7b1ce-0ba0ff0612714d059f5fc004] Parameters: {"file" => #ActionDispatch::Http::UploadedFile:0x000fffff78d1fb8 @tempfile=<Tempfile:/tmp/RackMultipart20250317-43-5vmtg.png>, "text" => "東京リージョンに対し、「us.anthropic.claude-3-7-sonnet-20250219-v1:0」を利用していることが原因と判明"}
# 4 466398113874:[ecs/pictures-api] 2025-03-17T05:23:26.6952 [1-67d7b1ce-0ba0ff0612714d059f5fc004] Completed 500 Internal Server Error in 184ms (ActiveRecord: 0.9ms (1 query, 0 cached) | GC: 0.0ms)
# 5 466398113874:[ecs/pictures-api] 2025-03-17T05:23:26.8782 [1-67d7b1ce-0ba0ff0612714d059f5fc004] Completed 500 Internal Server Error in 184ms (ActiveRecord: 0.9ms (1 query, 0 cached) | GC: 0.0ms)
# 6 466398113874:[ecs/pictures-api] 2025-03-17T05:23:26.8782 [1-67d7b1ce-0ba0ff0612714d059f5fc004] [Aws::BedrockRuntime::Client: #00 0.172947 0 retries] converse(model_id:'us.anthropic.claude-3-7-sonnet-20250219-v1:0',messages:[{"role":"user",content:"東京リージョンに対し、「us.anthropic.claude-3-7-sonnet-20250219-v1:0」を利用していることが原因と判明"}])
# 7 466398113874:[ecs/pictures-api] 2025-03-17T05:23:26.8802 [1-67d7b1ce-0ba0ff0612714d059f5fc004] Aws::BedrockRuntime::Errors::ValidationException: The provided model identifier is invalid.
# 8 466398113874:[ecs/pictures-api] 2025-03-17T05:23:26.8802 [1-67d7b1ce-0ba0ff0612714d059f5fc004] app/models/bedrock_client.rb:12:in `BedrockClient#tags'
# 9 466398113874:[ecs/pictures-api] 2025-03-17T05:23:26.8802 [1-67d7b1ce-0ba0ff0612714d059f5fc004] app/controllers/api/v1/pictures_controller.rb:16:in `Api::V1::PicturesController#create'
# 10 466398113874:[ecs/pictures-api] 2025-03-17T05:23:26.8802 [1-67d7b1ce-0ba0ff0612714d059f5fc004]
# 11 466398113874:[ecs/pictures-api] 2025-03-17T05:23:26.8802 [1-67d7b1ce-0ba0ff0612714d059f5fc004]
```

**Requests & Response**

**Request url**

`https://bedrock-runtime.ap-northeast-1.amazonaws.com:443/model/us.anthropic.claude-3-7-sonnet-20250219-v1:0/converse`

**Request method**

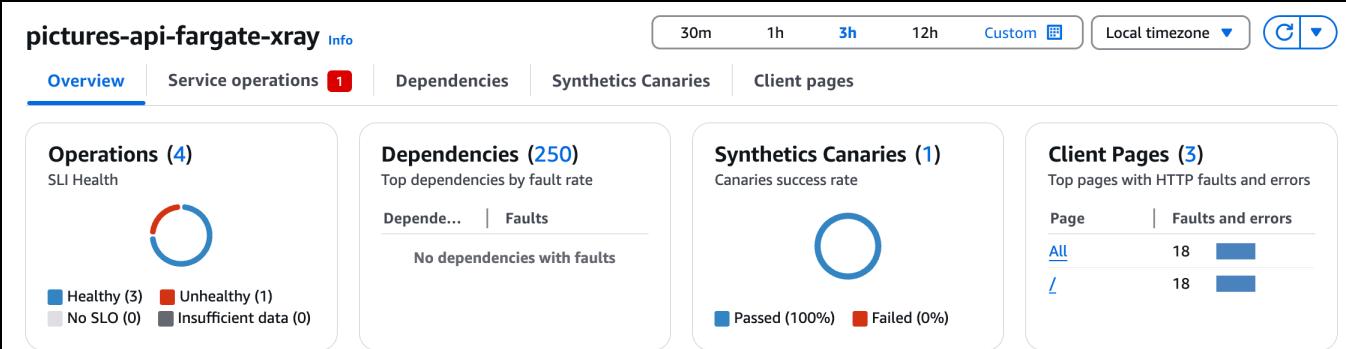
POST

**Response code**

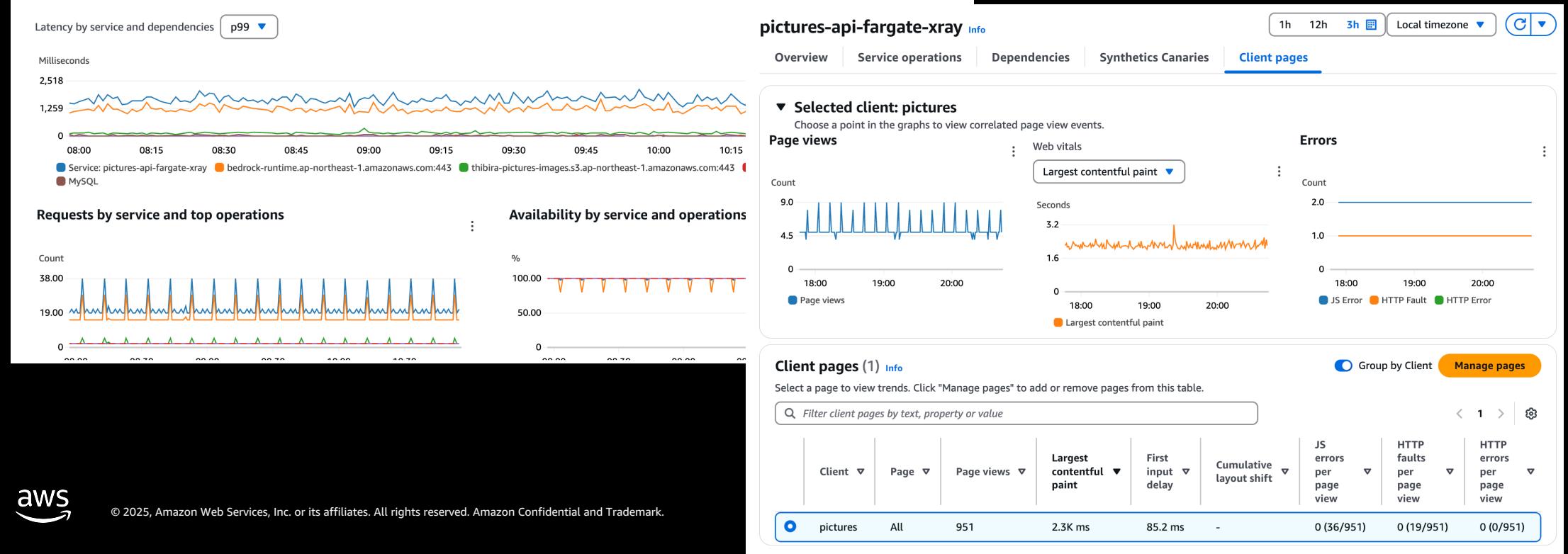
400

**東京リージョンに対し、「us.anthropic.claude-3-7-sonnet-20250219-v1:0」を利用していることが原因と判明**

# Application Signals との統合



CloudWatch Synthetics と CloudWatch RUM  
は Application Signals と 統合  
これにより、サービスの状況を 1 つのダッシュ  
ボードですぐに確認が可能



# まとめ

- このセッションではユーザー体験モニタリングの重要性と CloudWatch を利用したユーザー体験モニタリングを説明
- Application CloudWatch Synthetics や Application CloudWatch RUM と X-Ray を利用した調査を紹介

**CloudWatch を利用して  
ユーザー体験を向上しましょう！**



# Thank you!

Taiki Hibira

[www.linkedin.com/in/taiki-hibira](https://www.linkedin.com/in/taiki-hibira)

## [Appendix] 注意 ①

CloudWatch Synthetics canary は実際のエンドユーザーの操作ではないため、CloudWatch RUM は CloudWatch Synthetics canary の操作を記録しません。

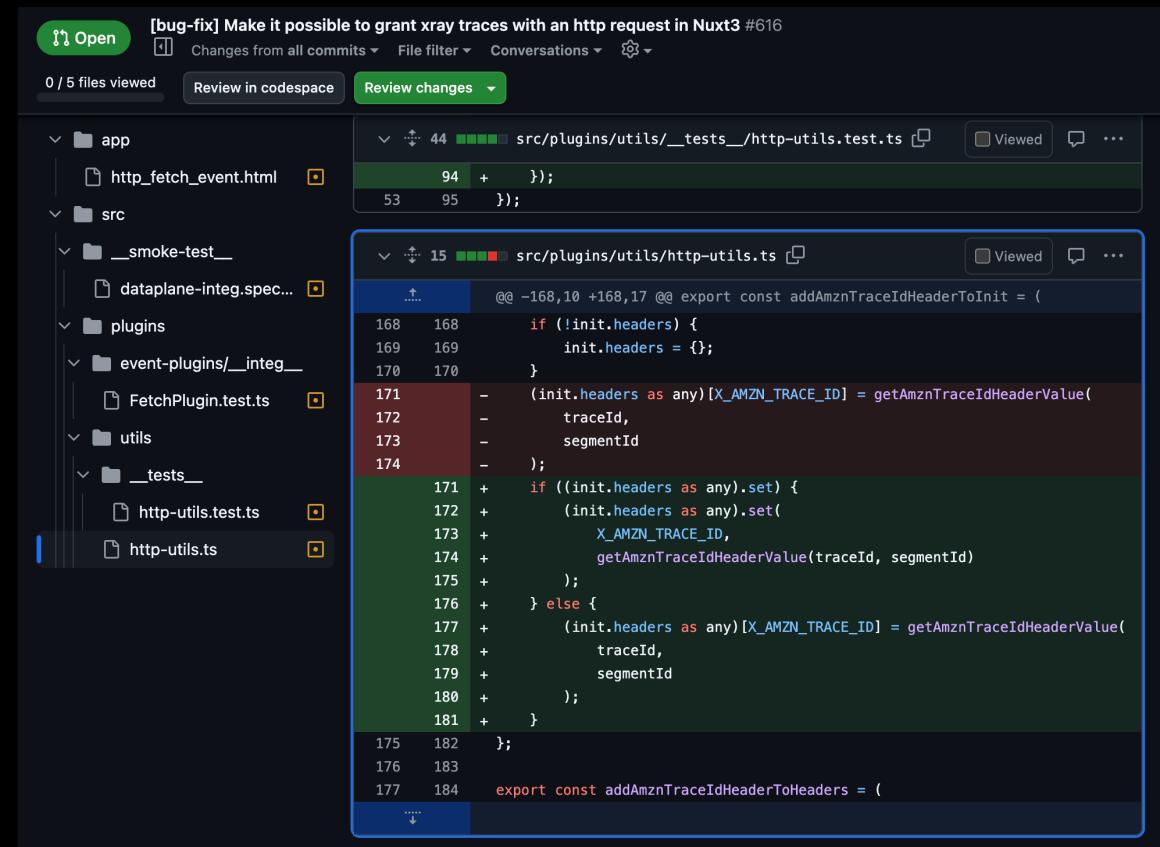
ただし、CloudWatch Synthetics canary で利用するヘッドレスブラウザの User-Agent を変更する事で、CloudWatch RUM で記録することが可能です。

今回のユースケースでは Synthetics クラスの [async addUserAgent\(page, userAgentString\)](#) を使用し、CloudWatch Synthetics canary のユーザー操作を CloudWatch RUM に記録しました。

```
let page = await synthetics.getPage(options);
await synthetics.addUserAgent(page, "myuseragent");
```

# [Appendix] 注意 ②

- このデモでは フロントエンドに Nuxt を利用していますが、RUM Client の問題により Http Request のヘッダーにトレース ID が正しく反映されません。
- 現在、GitHub に PR を提出し、レビュー待ちです。
- Nuxt をご利用の場合はご注意ください。



The screenshot shows a GitHub pull request titled "[bug-fix] Make it possible to grant xray traces with an http request in Nuxt3 #616". The code changes are displayed in a dark-themed code editor. The changes are focused on the `addAmznTraceIdHeaderToInit` function in `src/plugins/utils/http-utils.ts`. The changes add logic to handle both `set` and `get` operations for the `X\_AMZN\_TRACE\_ID` header.

```
diff --git a/src/plugins/utils/http-utils.ts b/src/plugins/utils/http-utils.ts
@@ -168,10 +168,17 @@ export const addAmznTraceIdHeaderToInit = (
 169   init.headers = {};
 170 }
 171 - (init.headers as any)[X_AMZN_TRACE_ID] = getAmznTraceIdHeaderValue(
 172 -   traceId,
 173 -   segmentId
 174 - );
 175 + if ((init.headers as any).set) {
 176 +   (init.headers as any).set(
 177 +     X_AMZN_TRACE_ID,
 178 +     getAmznTraceIdHeaderValue(traceId, segmentId)
 179 +   );
 180 } else {
 181   (init.headers as any)[X_AMZN_TRACE_ID] = getAmznTraceIdHeaderValue(
 182     traceId,
 183     segmentId
 184   );
 185 }
```

# [Appendix] 注意 ③

```
"unminifiedStack": "
TypeError: Cannot read properties of undefined (reading 'id')
at ../../../../pages/index.vue:172:0)
UnminifyLineFailure: Source map file 0cec3b1/[as default] _nuxt/DCLV2sy0.js.map not found at xxx-sorcemap for accountId XXX.
at Proxy.<anonymous> (../../../../node_modules/vuetify/lib/components/VGrid/VCol.mjs:125:7)
at Xo (../../../../node_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:6502:15)
UnminifyLineFailure: Source map file 0cec3b1/[as fn] _nuxt/DCLV2sy0.js.map not found at xxx-sorcemap for accountId XXX.
at Pi.run (../../../../node_modules/@vue/reactivity/dist/reactivity.esm-bundler.js:225:18)
at oe (../../../../node_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:5258:17)
at updateComponent (../../../../node_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:5193:22)
at n1 (../../../../node_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:4701:12)
at n1 (../../../../node_modules/@vue/runtime-core/dist/runtime-core.esm-bundler.js:5598:10)"
}
```

- スタックトレースの内容によって、Unminified が正しく動作しない場合があります。
- この問題はすでにサービスチームにも共有し、対応を予定しています。
- もし問題を発見された場合は、ぜひ貴社担当のアカウントチームやサポートへご連絡ください。

# [Appendix] 2024 年～2025 年のアップデート

- **Amazon CloudWatch Synthetics のアップデート**
  - Amazon CloudWatch Synthetics が canary 実行に関する 30 日間の履歴データのサポートを開始 (Mar 15, 2024)
  - Amazon CloudWatch Synthetics が、関連する AWS Lambda リソースへの Canary タグのレプリケーションのサポートを開始 (2024年9月19日)
  - Amazon CloudWatch Synthetics が NodeJS で Canary を作成するための Playwright ランタイムをサポート (2024 年11月21日)
  - Amazon CloudWatch Synthetics で Canary に関連付けられた Lambda リソースが自動的に削除されるように (2024年11月21日)
  - Amazon CloudWatch Synthetics が IPv6 サポートを追加 (2025年1月30日)
- **Amazon CloudWatch RUM のアップデート**
  - CloudWatch RUM PutRumEvents API が AWS CloudTrail のデータイベントロギングのサポートを開始 (2024年7月25 日)
  - CloudWatch RUM が、パーセンタイル集計と、ウェブバイタルメトリクスによる簡単なトラブルシューティングのサポートを開始 (2024年11月19日)
  - Amazon CloudWatch RUM がデータインジェストアクセスに関するリソースベースポリシーのサポートを導入 (2025年3月3日)
  - CloudWatch RUM が、エラーのデバッグを容易にする JavaScript ソースマップのサポートを開始 (2025年3月18日)
  - Amazon CloudWatch RUM が単一のアプリケーションモニタによる複数ドメインのモニタリングのサポートを開始 (2025年3月18日)

春の Observability 祭り 2025 ~進化する Amazon CloudWatch 基礎から最新機能まで完全解説~

# CloudWatch大好きなSAが語る CloudWatch キホンノキ

Miki Tsuwazaki

Solutions Architect

Amazon Web Service Japan G.K.

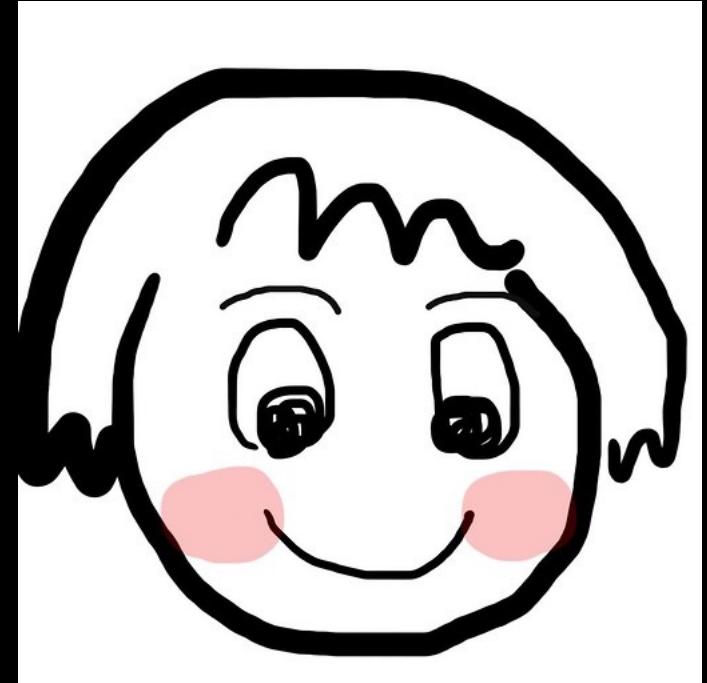


# 自己紹介

津和崎 美希 Miki Tsuwazaki

アマゾンウェブサービスジャパン  
ソリューションアーキテクト

通信業のお客様を中心にご支援しています。



## ■大好きなサービス



Amazon CloudWatch

# Amazon CloudWatchの好きなところ

- ①ある意味サーバーレスなところ
- ②スケーラビリティが求められる企業の事例があるところ
- ③初心者に優しいところ
- ④ログ保管用のストレージなど管理に頭をいためなくていいところ
- ⑤監視するシステムの進化にあわせて進化するところ

# CloudWatchの始め方

## ★まずはダッシュボードを見てみよう

CloudWatchをマネージメントコンソールで今使っているAWSサービスの状況を確認してみる。

The screenshot shows the CloudWatch Management Console with the 'CloudWatch' sidebar open. Under 'ダッシュボード', the '自動ダッシュボード' tab is selected. A callout bubble points from the sidebar to the '自動ダッシュボード' tab with the text: '利用中のサービスの自動ダッシュボードがあらかじめ用意されている。' (Automatic dashboards for currently used services are pre-prepared.)

①ダッシュボードをClick

②自動ダッシュボードをClick

カスタムダッシュボード | 自動ダッシュボード

自動ダッシュボード (30) 情報

リソースグループ別フィルタ条件

任意のリソースグル... ▾

名前

アラーム状態

△ アラーム状態

Application ELB

AutoScaling

Billing

DynamoDB

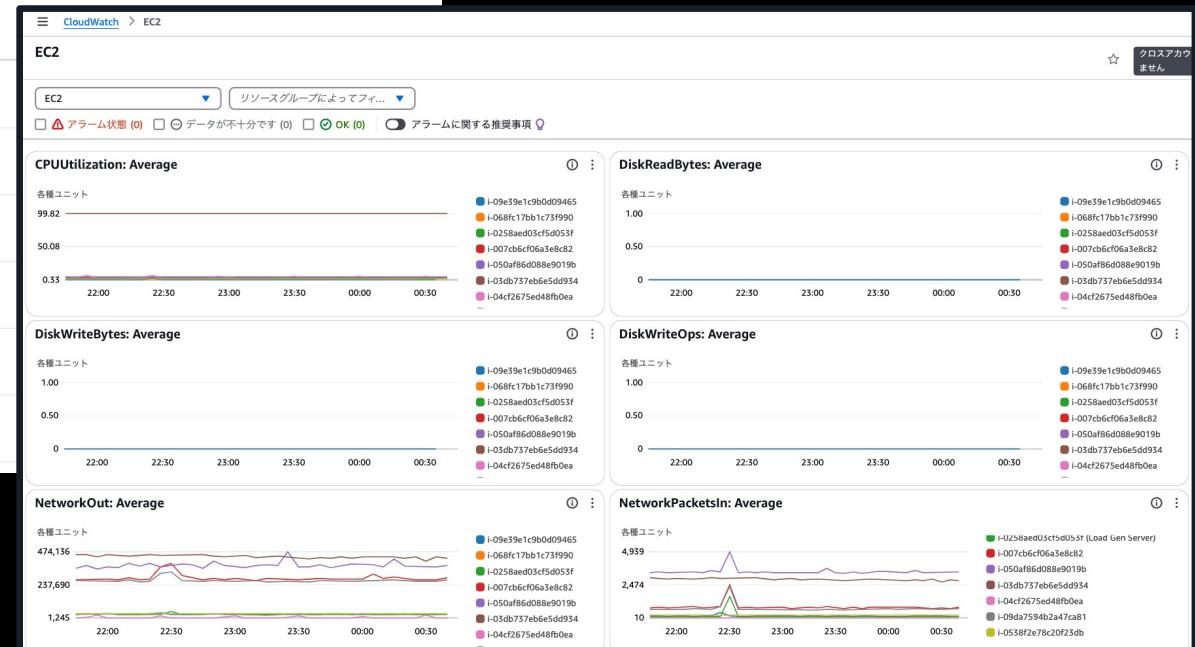
Elastic Block Store (EBS)

EC2

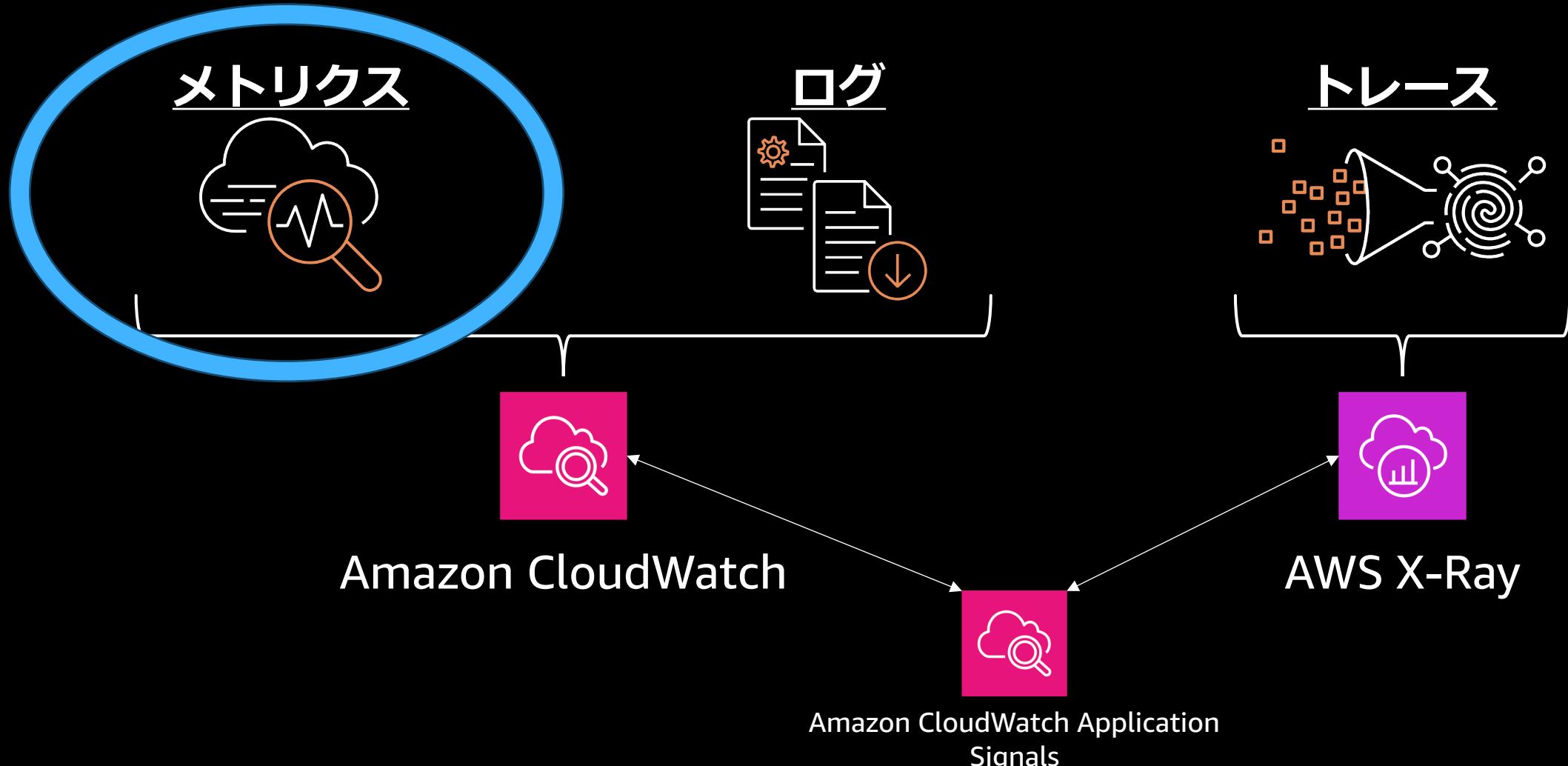
※この操作にかかる料金は無料 (2025/04/11時点)

- 基本のモニタリングメトリクス (デフォルトで AWS サービスから送信されるメトリクス) は無料
- すべての自動ダッシュボードは無料

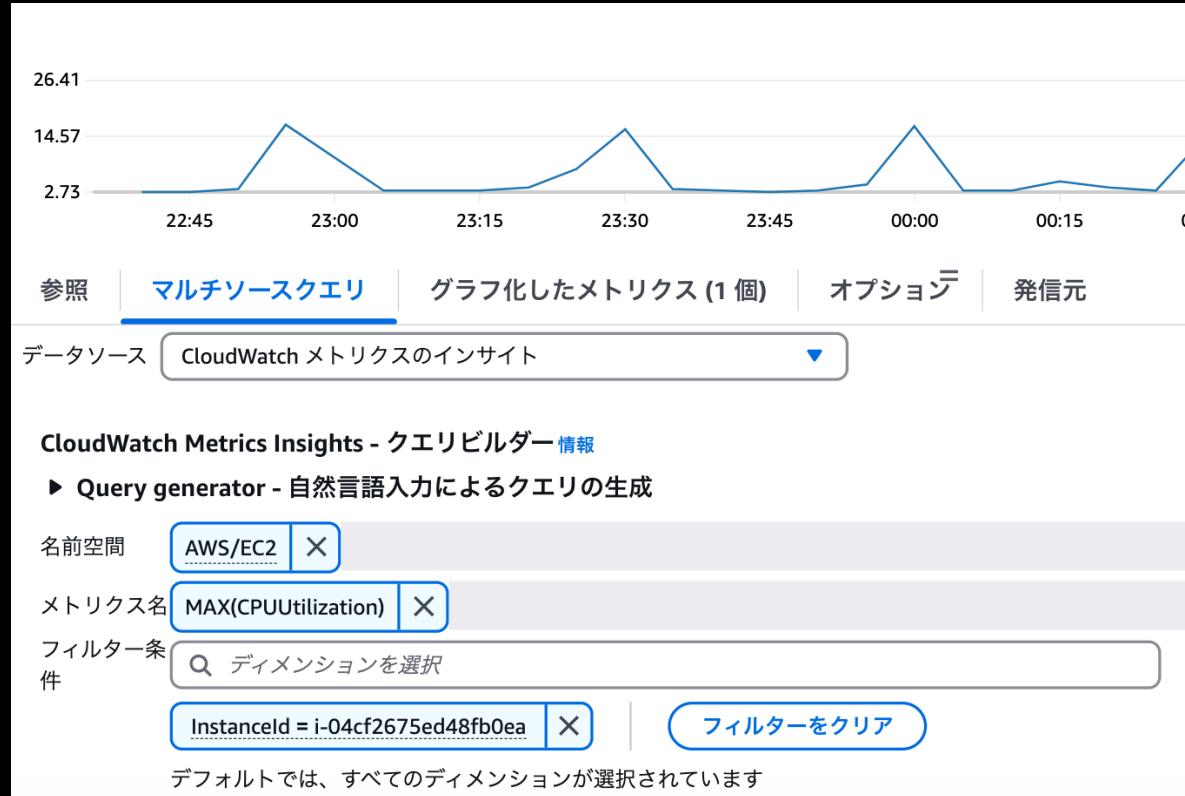
料金については、AWS公式Webサイトにてご確認ください  
<https://aws.amazon.com/jp/cloudwatch/pricing/>



# 再掲：CloudWatch / X-Ray は最初の選択肢



# CloudWatch メトリクスとは？



## メトリクス

- CloudWatchに発行された時系列のデータポイントのセット
- データポイントはタイプスタンプと（オプションで）測定単位を保持
- メトリクスは作成されたリージョンにのみ存在
- **メトリクスは名前空間、メトリクス名、ディメンションで一意に定義される。**
- メトリクスは削除不可で[保持スケジュール](#)に従い期限切れになる

## 名前空間

- CloudWatch メトリクスのコンテナ
- 異なる名前空間のメトリクスは相互に切り離される
- **AWSサービスでは AWS/<service>が使用される(例: AWS/EC2)**

## ディメンション

- **メトリクスを一意に識別する名前/値のペア**  
(例：InstanceId=i-12345678)

# ★メトリクスを可視化してみよう - DEMO

The screenshot shows the AWS CloudWatch Metrics Insights interface. On the left, there's a sidebar with navigation links for CloudWatch, AI Operations, Alarms, Logs, Metrics, X-Ray Traces, Events, and Application Signals. The main area is titled "タイトルなしのグラフ" (Untitled Graph) and displays three numerical values: 99.89, 50.11, and 0.34. A message says "CloudWatch グラフがブランクです。ここに表示するメトリクスをいくつか選択します。" (The CloudWatch graph is blank. Select the metrics to display here.) Below this is a timeline from 03:00 to 05:45. At the bottom, there are tabs for "マルチソースクエリ" (Multi-source query), "グラフ化したメトリクス" (Graphed metrics), "オプション" (Options), and "発信元" (Source). A dropdown menu for "データソース" (Data source) is set to "CloudWatch メトリクスのインサイト". On the right, there are buttons for "Builder" and "Editor". A modal window titled "CloudWatch Metrics Insights - クエリビルダー 情報" (Information) is open, showing a "Query generator - 自然言語入力によるクエリの生成" (Query generator - Generation by natural language input) section with a search bar and a list of AWS services: AWS/ECR, AWS/ECS, AWS/EKS, AWS/ELB, AWS/Events, AWS/Firehose, AWS/Glue, and AWS/HealthLake. The "AWS/ECR" item is currently selected.

# CloudWatch カスタムメトリクス

## 標準メトリクス以外の独自メトリクスも監視可能

- AWS CLIの“put-metric-data”もしくは“PutMetricData” APIでデータを登録
- CloudWatch Logsに“Embedded Metrics Format”で書くことで、カスタムメトリクスをLogに埋め込んで送ることも可能
- 詳細度が 1秒のデータを含む高解像度なメトリクスを発行可能 →1分未満のアクティビティを迅速に把握

```
$ aws cloudwatch put-metric-data --metric-name RequestLatency  
  --namespace "GetStarted"  
  --timestamp 2014-10-28T12:30:00Z  
  --value 87  
  --unit Milliseconds  
  --dimensions InstanceId=1-23456789,InstanceType=t3.small  
  
$ aws cloudwatch put-metric-data --metric-name RequestLatency  
  --namespace "GetStarted"  
  --timestamp 2014-10-28T12:30:00Z  
  --statistic-value Sum=60,Minimum=15,Maximum=105,SampleCount=5
```

← 単一値の登録

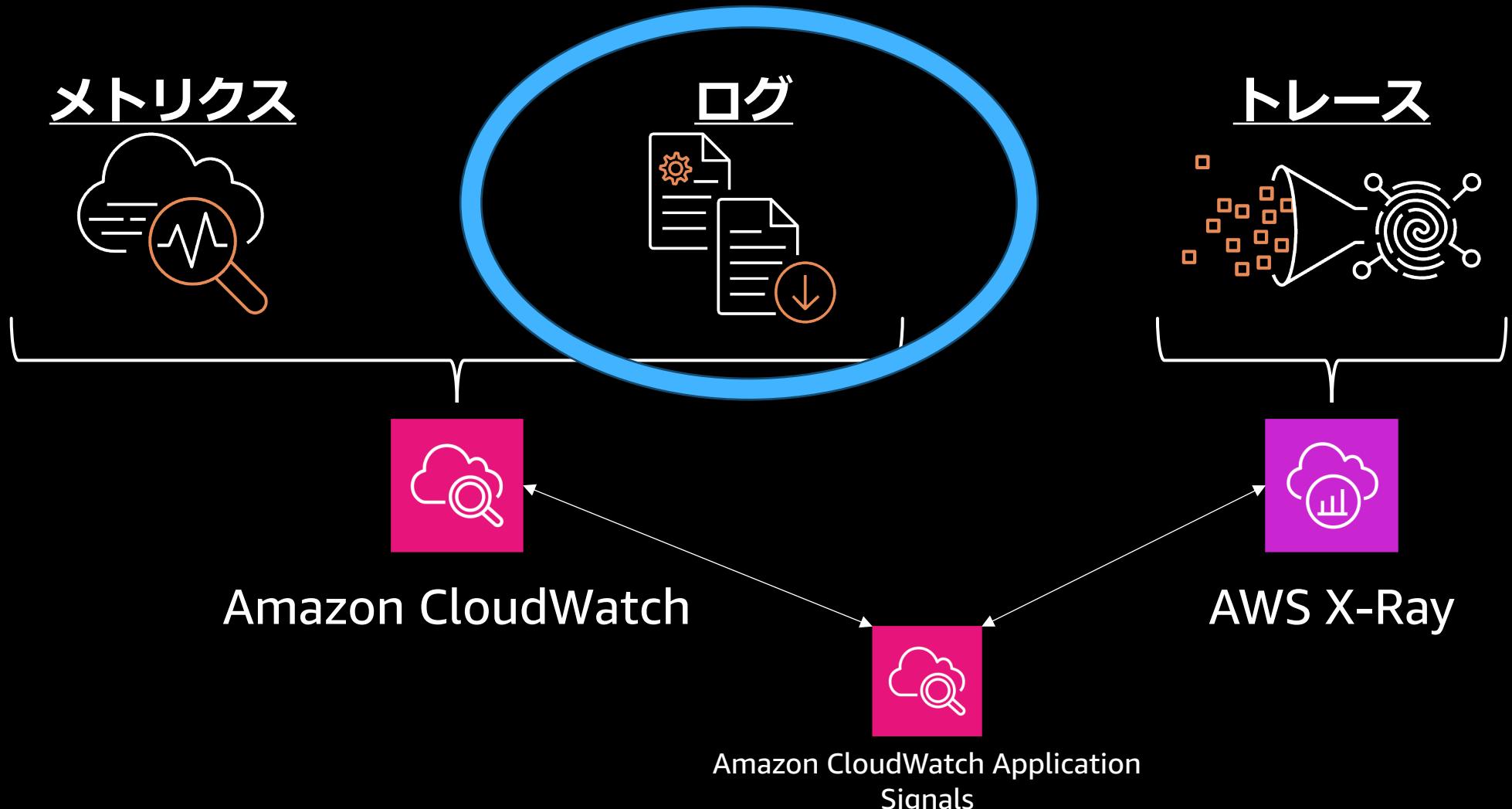
← 統計セットの登録

## 高解像度にメトリクスを発行する場合、APIコール時のスロットリング対策も忘れずに（上限緩和申請も検討）

- 単一のput-metric-dataに複数のデータポイントを入れる
- 統計セットを利用する
- リトライ処理をする



# 再掲：CloudWatch / X-Ray は最初の選択肢



# CloudWatch Logsの例

The screenshot illustrates the AWS CloudWatch Logs interface, specifically for a Lambda function named 'Summit\_Ad'.

**Log Group Details:** The top left shows the log group details for '/aws/lambda/Summit\_Ad'. It includes fields like ARN, creation time, retention, and log size.

**Log Stream Management:** The bottom left shows a list of 8 log streams, each corresponding to a Lambda execution. A blue arrow points from the text 'Lambda関数を実行するとログストリームが自動で追加される' (A log stream is automatically added when a Lambda function is executed) to this list.

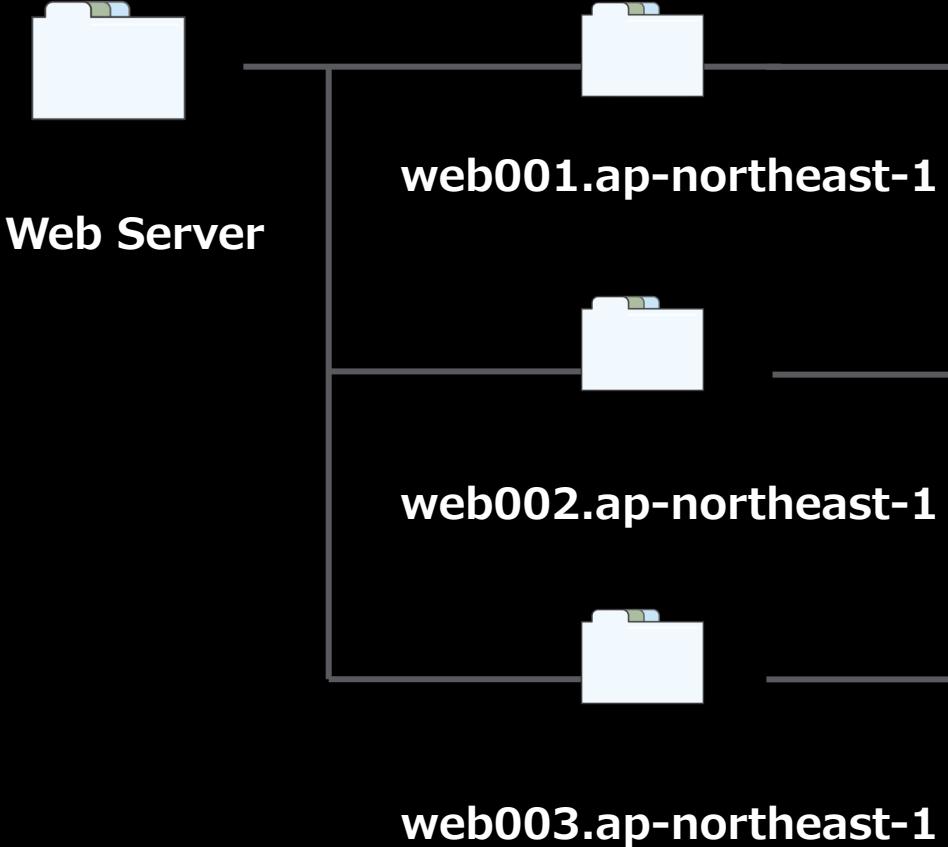
**Log Events:** The bottom right shows a detailed view of log events for a specific execution. A blue arrow points from the text 'ログイベントで生のイベントが確認できる' (Events can be checked directly in the log event) to the event list. The events show the runtime environment and request details.

**Central Summary:** A central blue box contains the text: 'ロググループが Lambda関数ごとにを作成される' (Log groups are created for each Lambda function).



# CloudWatch Logsの仕組み

## ロググループ    ログストリーム    ログイベント



### ログイベント

- 1つのログエントリ
- モニタリングしているリソースによって記録されたアクティビティのレコード
- イベント発生時のタイムスタンプおよび生のイベントメッセージで構成

### ログストリーム

- 同じソースを共有する一連のログイベント
- モニタリングしているリソースのタイムスタンプ順でイベントを表す

### ロググループ

- 複数のログストリームで構成
- 保持、監視、アクセス制御について同じ設定を共有するログストリームのグループを定義

# ★ログを可視化してみよう - DEMO

Lambda > 関数 > Summit\_Ad

## Summit\_Ad

スロットリング ARN をコピー アクション ▾ Infrastructure Composer へのエクスポート ダウンロード ▾

▼ 関数の概要 情報

ダイアグラム テンプレート

Summit\_Ad

Layers (0)

+ トリガーを追加 + 送信先を追加

説明

-

最終更新 1か月前

関数の ARN arn:aws:lambda:ap-northeast-1:100446803103:function:Summit\_Ad

関数 URL 情報

-

コード テスト モニタリング 設定 エイリアス バージョン

Monitor 情報

ロード中

Filter metrics by 関数 ▾

a)

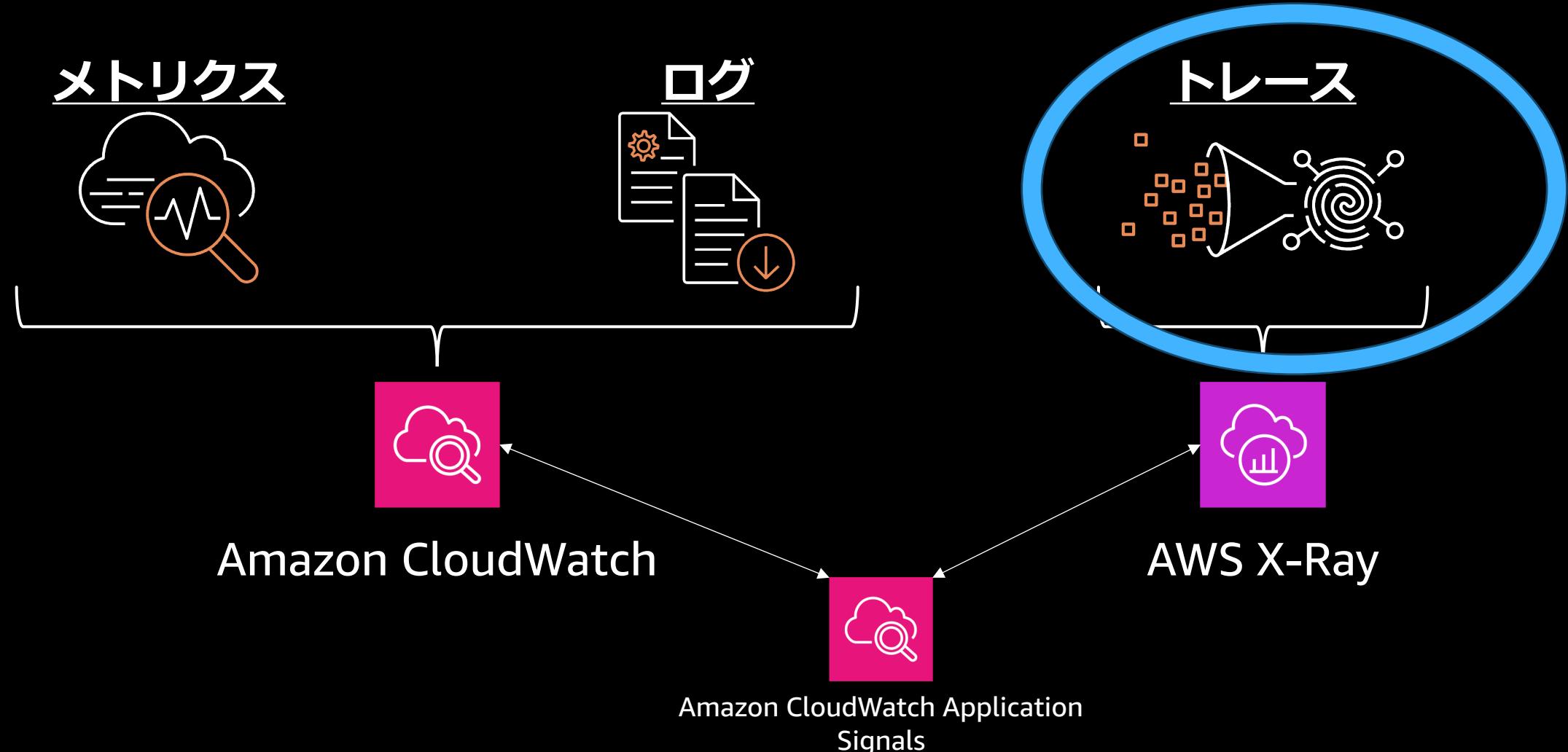
# CloudWatch Logs Insightsを活用して ロググループからエラーLOGを抽出して可視化

The screenshot illustrates the workflow for generating and executing a CloudWatch Logs Insights query:

- ① Natural Language Query Generation:** The user is in the "Logs Insights" section, generating a query in English: "Tell me the number of error messages every 10 minutes".
- ② Generated Query Execution:** The generated query is displayed in the main query editor:

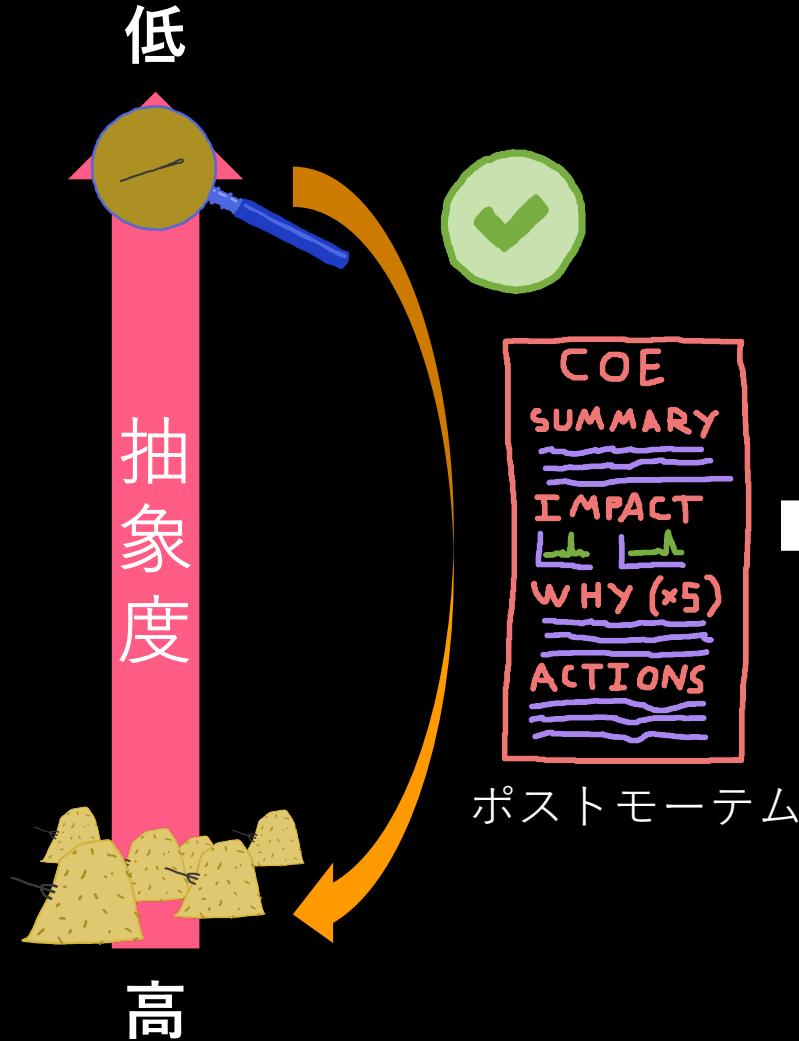
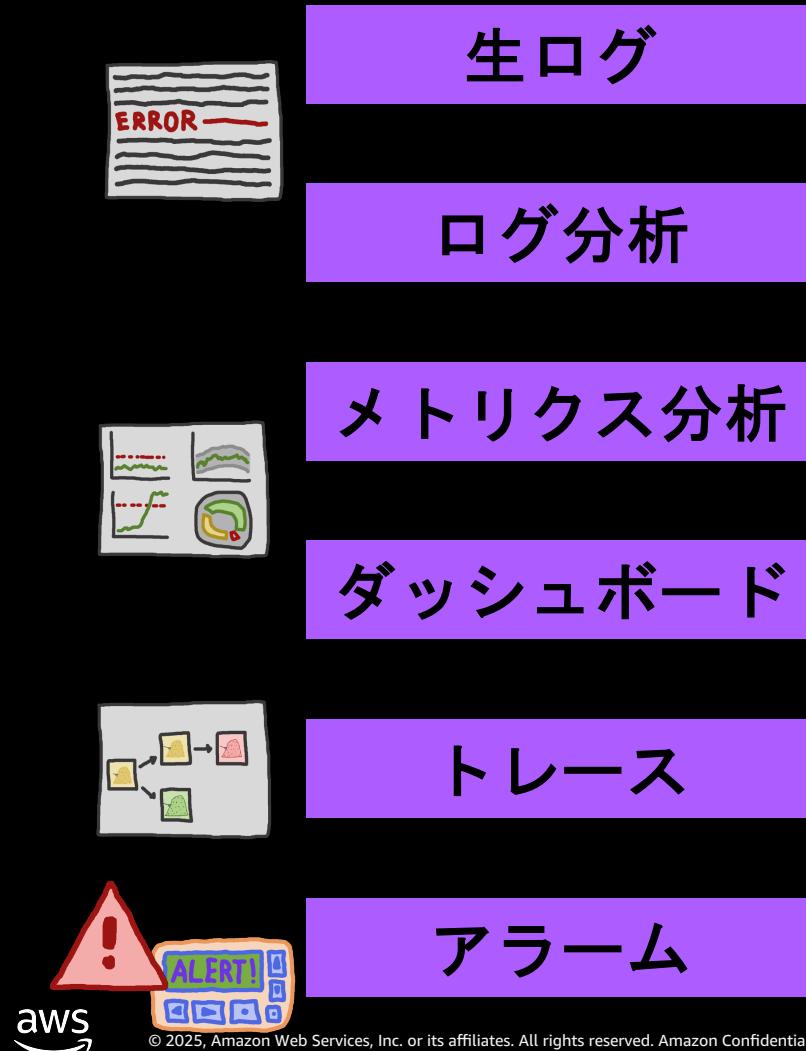
```
1 fields @timestamp, @message
2 | filter @message like /(?i)ERROR/
3 | stats count(*) as ctr by bin(10m)
```
- Execution Results:** The results show 23 matching log entries over a 10-minute window, with a total of 62 records (9.2 KB) processed at 51 records/s (7.7 KB/s).
- Cost Summary:** A sidebar provides a breakdown of costs for the query execution, including:
  - Logs (Logs Insights): USD 0.0076/スキャンされたデータの GB
  - 検出およびマスキング (データ保護): USD 0.12/スキャンされたデータの GB
  - 分析 (Live Tail): 0.01 USD/分
- Log Details:** The bottom pane shows the raw log entries in table format.

# 再掲：CloudWatch / X-Ray は最初の選択肢



# 参考) AmazonでのObservability best practice

## Amazon での対応例



なぜビジネスに影響がでたのか？アラームはなったのか？の質問に答えるために計装を追加していく

\*計装とは：  
ログ、メトリクス、トレースなどのデータを取得し、外部に送信できるようシステムに組み込むこと



運用を可視化するためのダッシュボードの構築



# ★トレースでできること

# CloudWatchにテレメトリーデータを集めるには？

- 統合CloudWatchエージェントを使えば、CloudWatch Logs・Metrics・Traceに対応している
- クラウドでもオンプレミスでもコンテナでも利用可能。
- Linux, Windowsの両方で稼働。

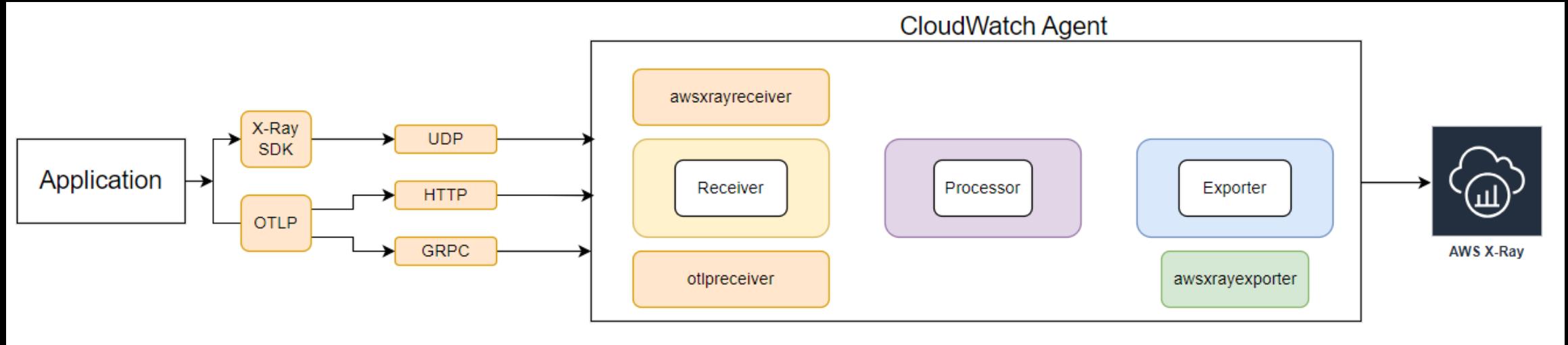
★サーバーへのインストール方法は大きく以下3パターン

1. コマンドラインで実行（複数サーバーに行うには工夫が必要）→コマンドラインを使用した CloudWatch エージェントのインストール
2. SSMを活用（SSMなら多数のサーバーにも簡単導入）→AWS Systems Manager を使用した CloudWatch エージェントのインストール
3. IaCで（CFnで併せてAgentも導入）→AWS CloudFormationを使用してCloudWatch エージェントのインストール

★コンテナ向けには、CloudWatch エージェントコンテナイメージを提供

→ [https://docs.aws.amazon.com/ja\\_jp/AmazonCloudWatch/latest/monitoring/ContainerInsights.html](https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/ContainerInsights.html)

# 参考：CloudWatch Agentでx-Rayにトレースを送る



AWS X-Ray と OpenTelemetry SDK の両方からのトレースを収集して AWS X-Ray に送信するよう CloudWatch エージェントを設定できるようになっている。

# CloudWatch Alarmsでメトリクスをアクションにつなげる

メトリクス

アクション



Alarm



Amazon  
CloudWatch

## ①静的閾値に基づくアラーム

- 例：5分間平均のCPU使用率が90%を6つの評価期間に2回以上発生  
= 30分の評価間隔で閾値超え2回ならばアラーム



Amazon SNS



Amazon EC2



Amazon EC2  
Auto Scaling



AWS Systems Manager  
OpsCenter



AWS Systems Manager  
Incident Manager

## ②Metrics Insights クエリに基づく CloudWatch アラーム

- 例：フリートのすべてのインスタンスの CPU 使用率を監視する  
→アラーム作成後の追加インスタンスにも監視が自動で適用される

## ③CloudWatch 異常検出に基づいたアラーム

- 例：最大2週間分のErrorCountメトリクスに機械学習アルゴリズムを適用して、メトリクスの想定値のモデルから“正常の範囲”を超えてErrorがある場合、突然の変化として検出する

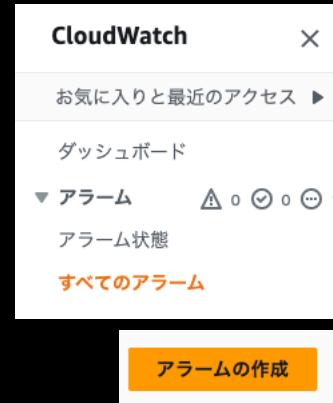
## ④複合アラーム

- (例：①と③両方ALARM状態のときにSNSトピックに通知する  
※EC2アクション未対応)

※Event Bridgeにも  
アラーム状態変化は  
配信される

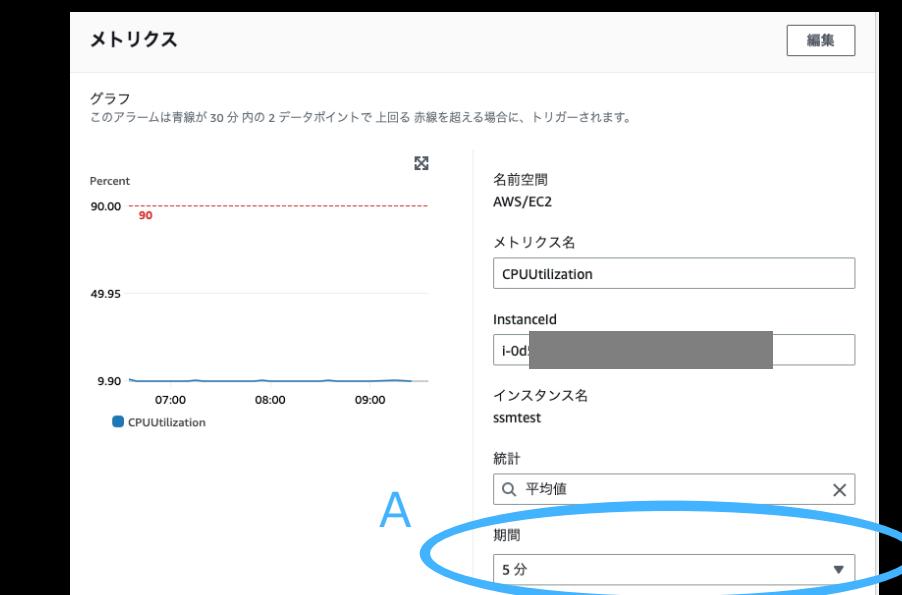


# アラームを作成する例



1.ナビゲーションペインで、[アラーム]、[すべてのアラーム] の順に選択

2.[アラームの作成] を選択



- A [期間] は、アラームの各データポイントを作成するためにメトリクスや式を評価する期間  
B [評価期間] は、アラームの状態を決定するまでに要する直近の期間(データポイント)の数  
C [Datapoints to Alarm (アラームを実行するデータポイント)] は、アラームが ALARM 状態に移るためにしきい値を超過する必要がある評価期間内のデータポイントの数

5分間平均のCPU使用率が90%を6つの評価期間に2回以上発生(欠落データはない場合)  
≒30分の評価間隔で閾値超え2回ならばアラーム

The 'Conditions' screen shows the 'しきい値の種類' section with '静的' selected. It defines an alarm condition where 'CPUUtilization' is greater than or equal to 90 for 2 consecutive periods of 5 minutes. The 'Actions' screen shows the '通知' section with 'OK' selected, which triggers an SNS notification to the 'Monitoringhandson' topic.

4.条件・アクションを設定

The 'Actions' screen shows the '通知' section with 'OK' selected. It also includes sections for 'アラーム状態トリガー' (with 'OK' selected), '次のSNSトピックに通知を送信' (with '既存のSNSトピックを選択' selected), and '通知の送信先' (with 'Monitoringhandson' selected). There are also sections for 'Eメール (エンドポイント)' and '通知の追加'.



# アラームを数多く作成するのが大変な時

The screenshot shows the AWS CloudWatch Metrics console for the Tokyo region. It displays 74 metrics across various instances. A tooltip is visible, providing information about alarm best practices and cost. A red annotation highlights the 'CloudFormation JSON' download option.

インスタンス名	InstanceId	メトリクス名	アラーム数	操作
db connection	i-0ea3665ceb...	CPUUtilization ⓘ	1 個のアラーム	詳細を表示
db connection	i-0ea3665ceb...	StatusCheckFailed ⓘ	1 個のアラーム	詳細を表示
ECS Instance - xray-sa...	i-051eedb9a2...	CPUUtilization ⓘ	1 個のアラーム	詳細を表示
ECS Instance - xray-sa...	i-051eedb9a2...	StatusCheckFailed ⓘ	1 個のアラーム	詳細を表示
test	i-0ac20c78f7...	StatusCheckFailed ⓘ	1 個のアラーム	詳細を表示
test	i-0ac20c78f7...	CPUUtilization ⓘ	1 個のアラーム	詳細を表示
test2dist1	i-00efcc3c25...	CPUUtilization ⓘ	1 個のアラーム	詳細を表示

アラームに関する推奨事項は、モニタリングのベストプラクティスに基づいています。アラームを作成する前に、推奨されるアラーム設定を確認してください。初期費用は発生せず、作成したアラームに対してのみお支払いいただきます。詳細については、「[AWS のサービスに関するアラームの推奨事項](#)」を参照してください。

アラームに関する推奨事項

アラームコードのダウンロード (8)

CloudFormation JSON

CloudFormation YAML

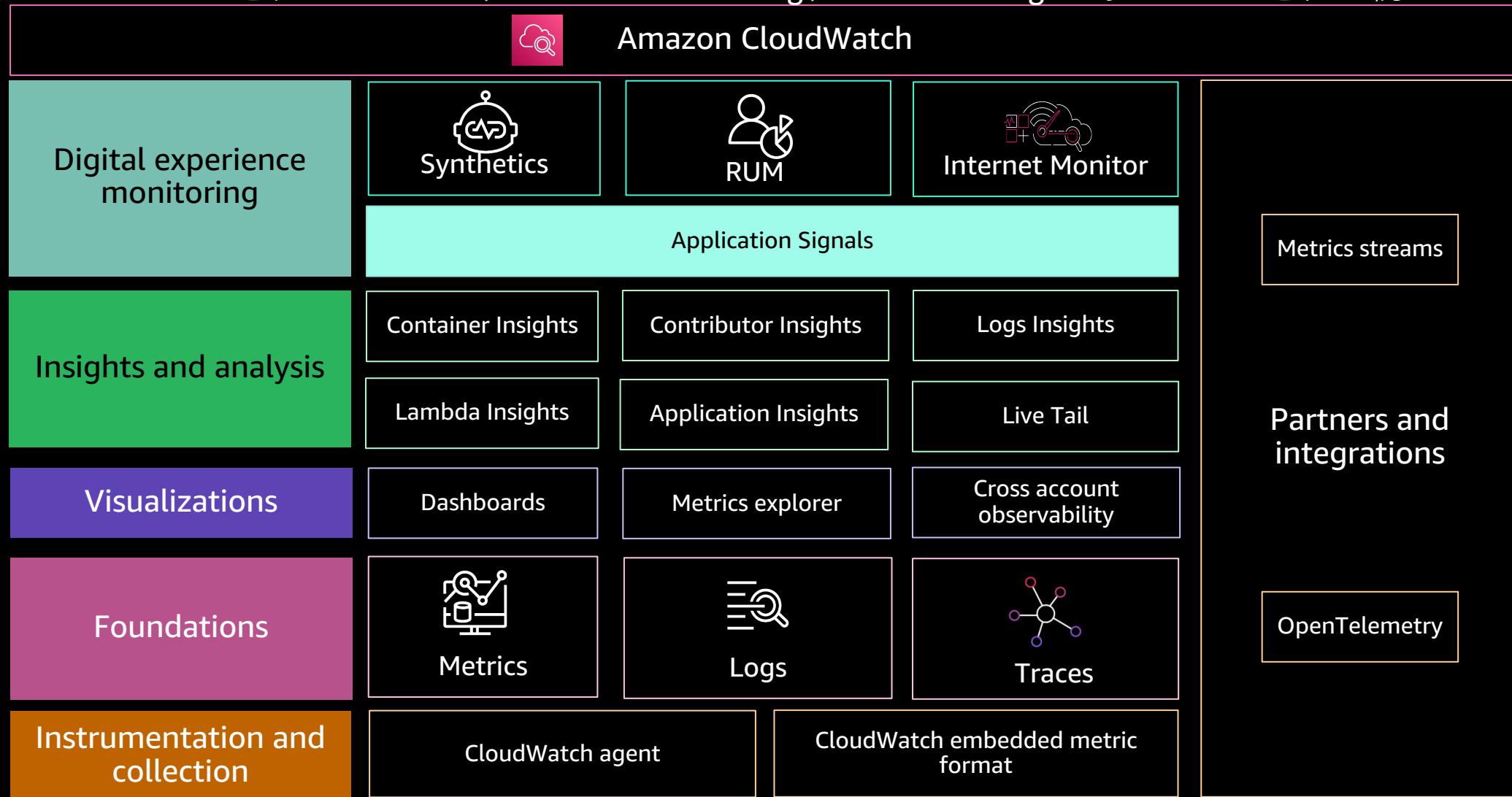
AWS CLI

Terraform

アラームに関する推奨事項とその設定のための IaC や CLI サンプルを AWS マネージメントコンソールからダウンロードして活用できます。

# CloudWatchの全体像

Application Signalsの進化に伴いアプリケーションモニタリングのカバレッジ向上に加え、監視するシステムの進化にあわせて、Network Monitoring、Database Insights等も加わり進化を続けていく。



# CloudWatch基本は簡単！まずは使ってみよう。

# Thank you



# CloudWatchの地味だけど強力な機能紹介！

2025/04/16

伊藤 威(Tsuyoshi Ito)

アマゾンウェブサービスジャパン合同会社



# 自己紹介

- 名前
  - 伊藤 威 (Tsuyoshi Ito)
- 所属
  - 主に通信業界のお客様をご支援
  - ソリューションアーキテクト
- 好きなサービス
  - Amazon CloudWatch



# アジェンダ

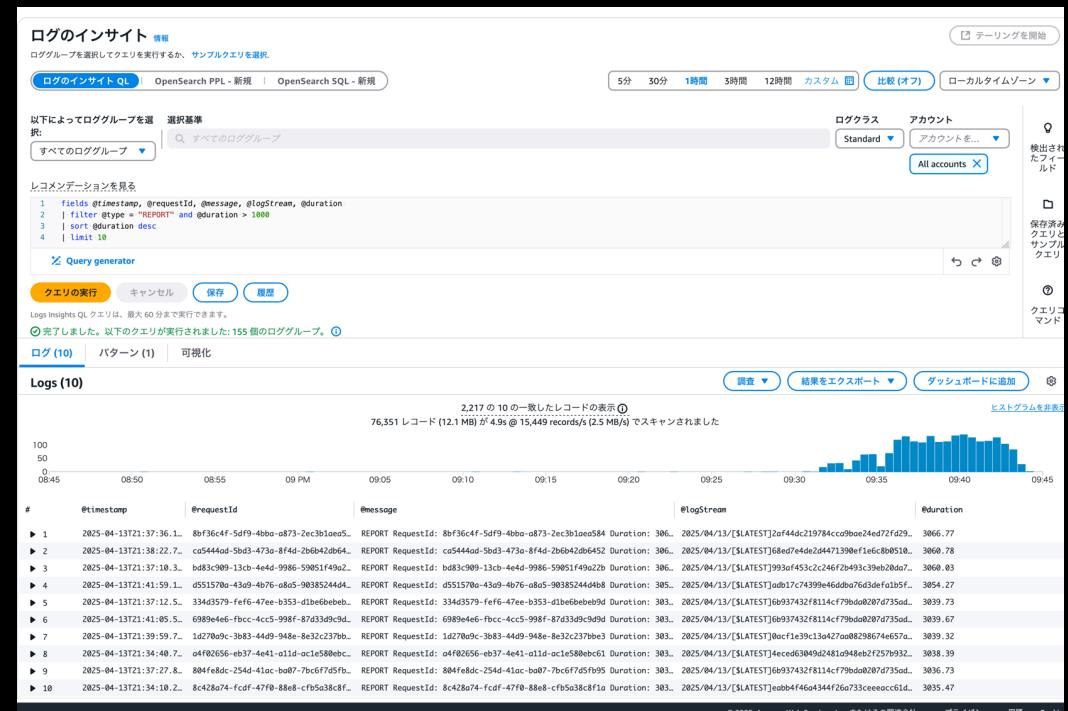
- CloudWatch logs insightsのフィールドインデックス機能
- クロスアカウントオブザーバビリティ×Live Tail
- Explore Related

# CloudWatch logs insightsの フィールドインデックス機能



# CloudWatch logs insights

- AWSが提供するログデータの分析サービス
- 専用クエリ言語(\*)でCloudWatch Logsから必要な情報を抽出可能
- 料金体系はスキャンされたログ量
  - USD 0.0076/スキャンされたデータの GB (東京リージョン)
- ユースケース
  - アプリケーションエラーの特定
  - パフォーマンス分析



(\*)直近のアップデートで OpenSearch PPL, SQLでのクエリも可能になりました



# CloudWatch Logs フィールドインデックス機能

- 課題
  - スキャン量による課金であるためログの規模が増えるとともに、クエリ時間コストも増加
- 実現できること
  - 重要なログ属性にインデックスを作成し、クエリパフォーマンスを向上
  - すべてのリージョンで利用可能
  - 追加料金の発生はしない

# そもそもフィールドとは？

- ログデータ内部で個別の情報要素を識別するための名前付き属性
  - システムフィールドはすべてのログに付与

フィールド名	説明
@message	解析されていない生のログイベント
@timestamp	ログイベントが発生した時刻
@ingestionTime	ログイベントを受信した時刻
@logStream	ログイベントが記録されるログストリーム
@log	account-id:log-group-name で表記される。ロググループ識別子
@entity	(option) Explore Related で使用するエンティティ情報

自動生成されたフィールドには先頭に @ がつく

#	@timestamp	@requestId	@message
▼ 1	2025-04-13T2...	52a52bbf-3e...	2025-04-13T13:09:38.713Z 52a52bbf-3e71-46b4-8e01-3de655be55a1 Task timed out after 1 second
フィールド		値	
@entity.KeyAttributes.Name		TimeFunctionsStack-TimeFunction87DF51133-2pCKWRjS2a2o	
@entity.KeyAttributes.Type		Service	
@entity.KeyAttributes.Environment		lambda:default	
@entity.Attributes.Lambda.Function		TimeFunctionsStack-TimeFunction87DF51133-2pCKWRjS2a2o	
@entity.Attributes.PlatformType		AWS::Lambda	
@ingestionTime		1744549778716	
@log		426501616617:/aws/lambda/TimeFunctionsStack-TimeFunction87DF51133-2pCKWRjS2a2o	
@logStream		2025/04/13/[LATEST]a483212b95164b75bce48a75ade35277	
@message		2025-04-13T13:09:38.713Z 52a52bbf-3e71-46b4-8e01-3de655be55a1 Task timed out after 1 second	
@requestId		52a52bbf-3e71-46b4-8e01-3de655be55a1	
@timestamp		1744549778716	

Lambda のログ



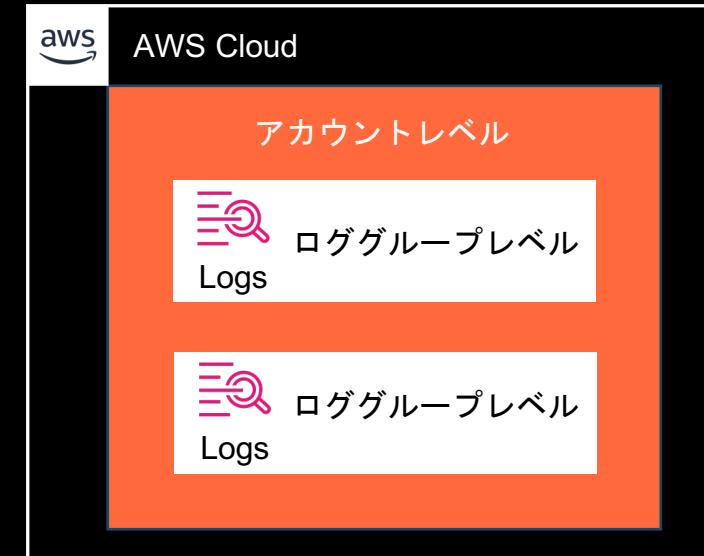
# フィールドインデックスを付与できるシステムフィールド

フィールド名	説明	補足
@ingestionTime	ログイベントを受信した時刻	システム
@logStream	ログイベントが記録されるログストリーム	システム
@requestId	Lambda などでリクエストごとに生成される一意の ID	Lambda 関連
@type	Lambda のログエントリタイプ(START, END, REPORT)	Lambda 関連
@initDuration	Lambda関数のINITセクション（メインハンドラーの外側のコード）の実行にかかった時間	Lambda 関連
@duration	クションLambda関数ハンドラーの合計呼び出し時間	Lambda 関連
@billedDuration	請求目的で丸められた時間	Lambda 関連
@memorySize	Lambda 関数に割り当てられたメモリ	Lambda 関連
@maxMemoryUsed	Lambda 呼び出し中に使用された最大メモリ量	Lambda 関連
@xrayTraceId	X-ray のトレース ID	X-ray 関連
@xraySegmentId	時刻X-ray セグメントID	X-ray 関連

@message などの多くの情報を含むフィールドは指定できない点には注意  
この表以外のカスタムフィールドも指定可能

# インデックスポリシー

- ・ フィールドインデックスを適用するためのポリシー
  - アカウントレベル: アカウントに含まれるロググループ全体
  - ロググループレベル: 指定した特定のロググループのみ
- ・ インデックスポリシーの制限
  - 1つのポリシーにつき**20**フィールド
  - ロググループレベルが優先

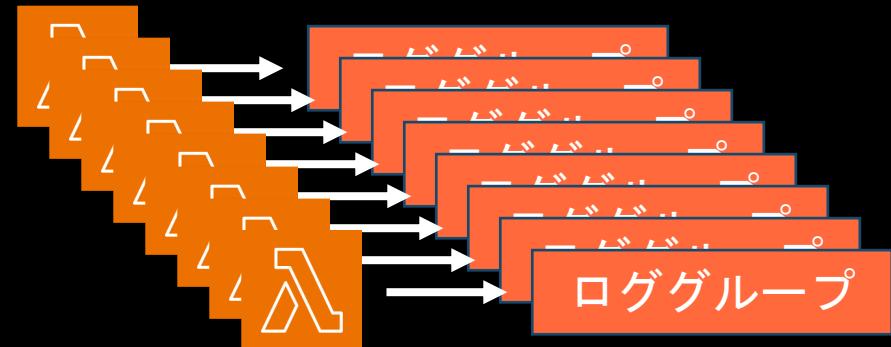


# デモを通じてフィールドインデックスを学ぶ

- Lambda のログをクエリを通して効果を体験

- 設定

- 10個の Lambda 関数
- 事前に1,000,000回以上呼び出し
- アカウントレベルで下記のフィールドにインデックスを作成
  - ✓ @type
  - ✓ @requestId



Lambda 関数とロググループの関係

# フィールドインデックスを設定

アカウントレベルのインデックスを編集  
このアカウントで選択したロググループのインデックスポリシーを編集します。

インデックスポリシーの詳細  
ポリシー名  
インデックスポリシーの作成後は、ポリシー名を編集できません。  
lambda-type  
ポリシー名はアカウント内で一意でなければならず、最大 256 文字です。

ロググループを選択 [情報](#)  
このインデックスポリシーを適用するロググループをこのアカウントに追加します。  
 すべての標準ロググループ  プレフィックスの一致でロググループを選択

① 複数のアカウントインデックスポリシーを設定できますが、曖昧さを回避するために、それぞれに異なるプレフィックスを指定する必要があることに留意してください。

プレフィックス名を入力  
インデックスポリシーごとに選択できるプレフィックスは 1 個だけです。  
/aws/lambda/ [X](#)  
インデックスポリシーは、次にのみ適用されます: [すべての標準ログクラス](#) このプレフィックスで始まる名前のロググループ(例: /aws/lambda/)

プレフィックスが一致したロググループをプレビュー [設定](#) [X](#)

フィールドインデックスの詳細  
フィールドバス [情報](#)  
@requestId [削除](#)  
@type [削除](#)  
フィールドバス  
最大でさらに 18 個のフィールドを追加できます。  
① 選択したフィールドインデックスは、最大 30 日間インデックス化されたままになります。

キャンセル [変更を保存](#)

プレフィックスを絞って  
インデックスを適用

/aws/lambda/TimeFunctionsStack-TimeFunction1971F84ED-XDdz9m3mSVJi  
アクション Logs Insights で表示 テーリングを開始 ロググループの検索

▼ ロググループの詳細  
ログクラス [情報](#)  
スタンダード  
ARN [X](#) arn:aws:logs:us-east-1:426501616617:log-group:/aws/lambda/TimeFunctionsStack-TimeFunction1971F84ED-XDdz9m3mSVJi:  
作成時刻 2 時間前  
保持 失効しない  
KMS キー ID  
異常検出 [設定](#)

データ保護  
機密データの数  
ファイルインデックス 2  
トランスマーフォーマー [設定](#)

メトリクスフィルター サブスクリプションフィルター 寄稿者のインサイト データ保護 フィールドインデックス - 新規 トランスマーフォーマー - 新規 [設定](#) [X](#)

① 複数のロググループにわたるインデックスを管理するには、アカウントレベルのインデックスポリシーを使用します。

Log group field indexes  
View and manage field indexes for this log group  
λ [X](#) 1 [@](#)

フィールドバス	ステータス	最初のイベント時刻	最終のイベント時刻	最後のスキャン時間
lambda-type	Active	31 分前	1 分前	27 分前
@requestId	Active	31 分前	1 分前	27 分前
@type	Active	43 分前	1 分前	25 分前

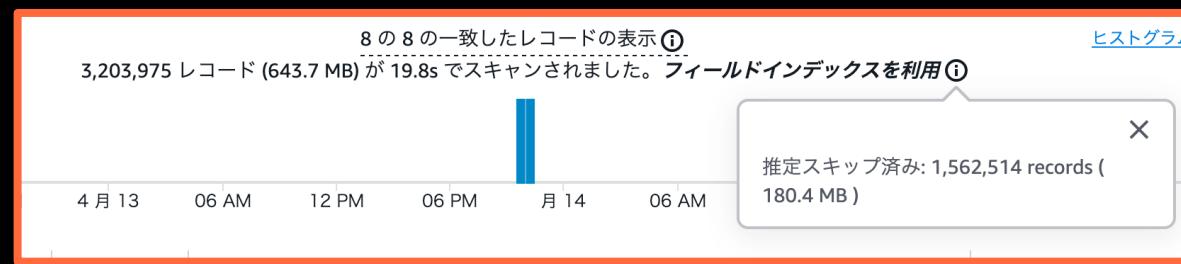
@type と @requestId を指定

アカウントレベルのポリシーがロ  
ググループに適用されている

# 特定の ID を抽出してみる

```
1 fields @timestamp, @requestId, @message, @logStream, @duration  
2 | filter @requestId IN ["52a52bbf-3e71-46b4-8e01-3de655be55a1", "b507ab31-2368-4e03-9b4f-f0132d92f095"]
```

全ログ 約 600 MB から 180 MB をスキップ



ログのインサイト 情報

ロググループを選択してクエリを実行するか、サンプルクエリを選択。

ログのインサイト QL | OpenSearch PPL - 新規 | OpenSearch SQL - 新規

30分 3時間 3日 回 比較(オフ) ローカルタイムゾーン ▾

すべてのロググループ ▾ 検出されたフィールド

ログクラス Standard アカウント All accounts X

レコマンデーションを見る

1 fields @timestamp, @requestId, @message, @logStream, @duration  
2 | filter @requestId IN ["52a52bbf-3e71-46b4-8e01-3de655be55a1", "b507ab31-2368-4e03-9b4f-f0132d92f095"]

Query generator

クエリの実行 キャンセル 保存 履歴

Logs (8) パターン (4) 可視化

8 の 8 の一致したレコードの表示 ①  
3,203,975 レコード (643.7 MB) が 19.8s でスキャンされました。フィールドインデックスを利用 ①

ヒストグラムを非表示

推定スキップ済み: 1,562,514 records (180.4 MB)

#	@timestamp	@requestId	@message	@logStream
1	2025-04-13T2...	52a52bbf-3e...	2025-04-13T13:09:38.713Z 52a52bbf-3e71-46b4-8e01-3de655be55a1 Task timed o...	2025/04/13/[\${LATEST}]a48321...
2	2025-04-13T2...	52a52bbf-3e...	END RequestId: 52a52bbf-3e71-46b4-8e01-3de655be55a1	2025/04/13/[\${LATEST}]a48321...
3	2025-04-13T2...	52a52bbf-3e...	REPORT RequestId: 52a52bbf-3e71-46b4-8e01-3de655be55a1 Duration: 3022.36 m...	2025/04/13/[\${LATEST}]a48321...
4	2025-04-13T2...	52a52bbf-3e...	START RequestId: 52a52bbf-3e71-46b4-8e01-3de655be55a1 Version: \${LATEST}	2025/04/13/[\${LATEST}]a48321...
5	2025-04-13T2...	b507ab31-23...	2025-04-13T12:50:16.934Z b507ab31-2368-4e03-9b4f-f0132d92f095 Task timed o...	2025/04/13/[\${LATEST}]69d059...



# フィールドインデックスが無効な場合

= 演算子を使用

ログのインサイト 情報

ロググループを選択してクエリを実行するか、サンプルクエリを選択。

ログのインサイト QL | OpenSearch PPL - 新規 | OpenSearch SQL - 新規

```
1 fields @timestamp, @requestId, @message, @logStream, @duration
2 | filter @requestId = "52a52bbf-3e71-46b4-8e01-3de655be55a1"
```

1 Fields @timestamp, @requestId, @message, @logStream, @duration  
2 | filter @requestId = "52a52bbf-3e71-46b4-8e01-3de655be55a1"

クエリの実行 キャンセル 保存 履歴

4 の 4 の一致したレコードの表示 ①  
3,198,946 レコード (643.2 MB) が 20.4s でスキャンされました。フィールドインデックスを利用 ①

ヒストグラム

推定スキップ済み: 1,562,533 records (180.4 MB)

6 PM 4月13 06 AM 12 PM 06 PM 月14 06 AM

推定スキップ済み: 1,562,533 records (180.4 MB)

#	@timestamp	@requestId	@message	@logStream
1	2025-04-13T2...	52a52bbf-3e...	2025-04-13T13:09:38.713Z 52a52bbf-3e71-46b4-8e01-3de655be55a1 Task timed o...	2025/04/13/[\${LATEST}]o48321
2	2025-04-13T2...	52a52bbf-3e...	END RequestId: 52a52bbf-3e71-46b4-8e01-3de655be55a1	2025/04/13/[\${LATEST}]o48321
3	2025-04-13T2...	52a52bbf-3e...	REPORT RequestId: 52a52bbf-3e71-46b4-8e01-3de655be55a1 Duration: 3022.36 m...	2025/04/13/[\${LATEST}]o48321
4	2025-04-13T2...	52a52bbf-3e...	START RequestId: 52a52bbf-3e71-46b4-8e01-3de655be55a1 Version: \${LATEST}	2025/04/13/[\${LATEST}]o48321

ログのインサイト 情報

ロググループを選択してクエリを実行するか、サンプルクエリを選択。

ログのインサイト QL | OpenSearch PPL - 新規 | OpenSearch SQL - 新規

```
1 fields @timestamp, @requestId, @message, @logStream, @duration
2 | filter @requestId like "52a52bbf-3e71-46b4-8e01-3de655be55a1"
```

1 Fields @timestamp, @requestId, @message, @logStream, @duration  
2 | filter @requestId like "52a52bbf-3e71-46b4-8e01-3de655be55a1"

クエリの実行 キャンセル 保存 履歴

4 の 4 の一致したレコードの表示 ①  
4,761,452 レコード (823.7 MB) が 20.3s @ 234,901 records/s (40.6 MB/s) でスキャンされました

ヒストグラム

06 PM 4月13 06 AM 12 PM 06 PM 月14 06 AM 12 PM 06 PM 火15 06 AM

0 12 PM 06 PM 4月13 06 AM 12 PM 06 PM 月14 06 AM 12 PM 06 PM 火15 06 AM

#	@timestamp	@requestId	@message	@logStream
1	2025-04-13T2...	52a52bbf-3e...	2025-04-13T13:09:38.713Z 52a52bbf-3e71-46b4-8e01-3de655be55a1 Task timed o...	2025/04/13/[\${LATEST}]o48321
2	2025-04-13T2...	52a52bbf-3e...	END RequestId: 52a52bbf-3e71-46b4-8e01-3de655be55a1	2025/04/13/[\${LATEST}]o48321
3	2025-04-13T2...	52a52bbf-3e...	REPORT RequestId: 52a52bbf-3e71-46b4-8e01-3de655be55a1 Duration: 3022.36 m...	2025/04/13/[\${LATEST}]o48321
4	2025-04-13T2...	52a52bbf-3e...	START RequestId: 52a52bbf-3e71-46b4-8e01-3de655be55a1 Version: \${LATEST}	2025/04/13/[\${LATEST}]o48321



# 注意事項

- ・ インデックスが適用される演算子は = と IN のみ
  - 正規表現を使用可能な like 演算子、不等号 <, > には適用されない
- ・ ポリシー作成前のログにインデックスは付与できない
  - 事前にクエリするフィールドをインデックス化する必要がある

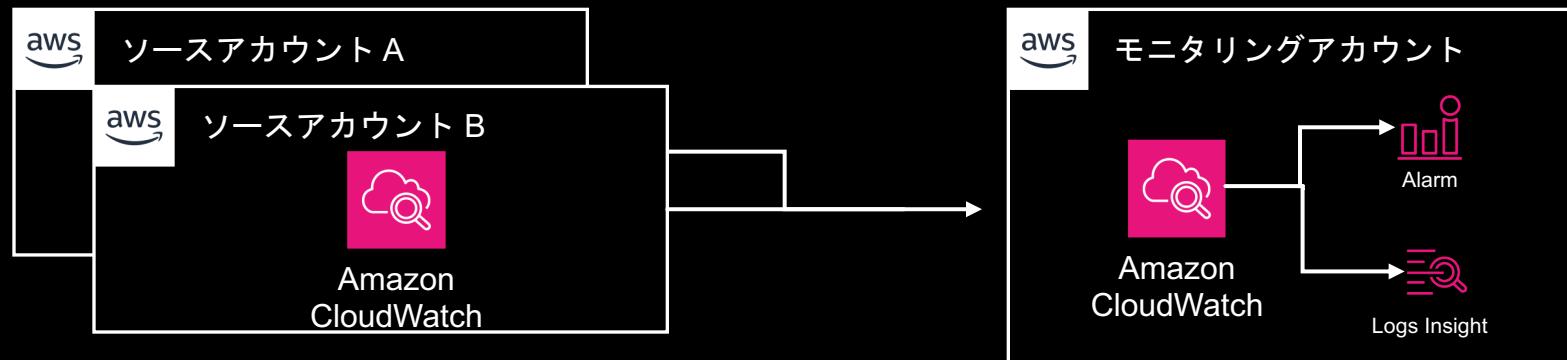
# クロスアカウントオブザーバビリティ×Live Tail



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

# CloudWatch のクロスアカウントオブザーバビリティとは

- 課題
  - マルチアカウント管理をする場合、横断的に分析するのが困難
- 実現すること
  - 「メトリクス」「ログ」「トレース」などを一つのアカウントへ共有
  - モニタリングアカウントで、ソースアカウントのテレメトリに対して分析が可能



\* ソースアカウントとモニタリングアカウントは同一リージョン

CloudWatch のクロスアカウントオブザーバビリティ [https://docs.aws.amazon.com/ja\\_jp/AmazonCloudWatch/latest/monitoring/CloudWatch-Unified-Cross-Account.html](https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/CloudWatch-Unified-Cross-Account.html)

## クロスアカウントオブザーバビリティで共有できるテレメトリ

- テレメトリ
  - ログ
  - メトリクス
  - トレース
  - Application Insights – Applications
  - Internet Monitor – モニター
  - Application Signals – サービス、サービスレベル目標(SLO)
- ログとメトリクス、Application Signals の追加コストは発生せず、最初のトレースコピーも無料(複数のモニタリングアカウントに接続すると料金発生)

# 共有方法

AWS Organizations に属しているアカウント(推奨) or 個別のアカウント

- モニタリングアカウント側
  - ソースアカウントの org-id か、個別の AWS アカウント ID を指定
  - 共有受け入れるテレメトリを指定
- ソースアカウント側
  - モニタリングアカウントのシンク ARN を入手
  - 共有を許可するテレメトリを指定

意図しないアカウントからの意図しないテレメトリの共有を防ぐ

[https://docs.aws.amazon.com/ja\\_jp/AmazonCloudWatch/latest/monitoring/CloudWatch-Unified-Cross-Account.html](https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/CloudWatch-Unified-Cross-Account.html)



# 共有されたテレメトリはシームレスに分析可能

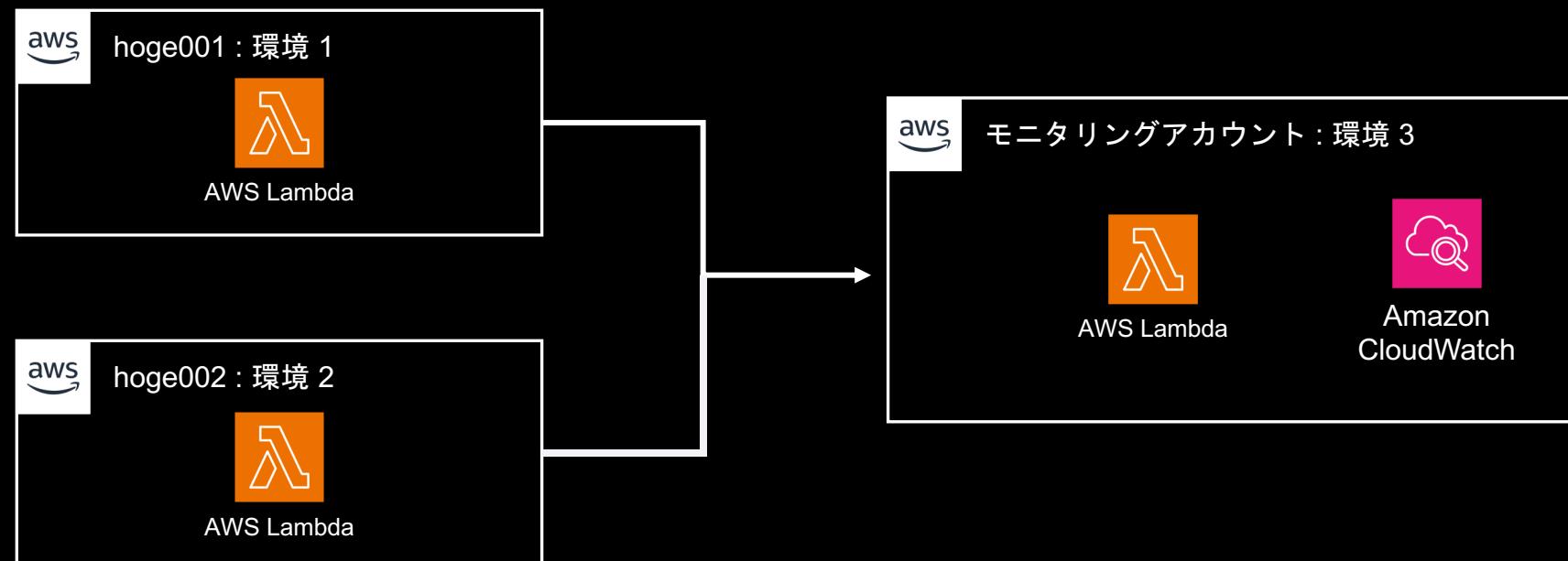
## Live Tail 機能で試してみる

- Live Tail とは?
  - CloudWatch Log ロググループに配信されるログをリアルタイムで可視化
  - 料金
    - ✓ 一ヶ月あたり 1,800 分までは無料
    - ✓ 0.01 USD/分

The screenshot shows the 'Live Tail' interface for CloudWatch Metrics Insights. At the top, there's a search bar with the word 'error' and a timestamp '00:00:36'. Below it is a table with columns for 'タイムスタンプ(ローカル)', 'メッセージ', and 'ロググループ'. The table lists numerous log entries from a Lambda function, each with a timestamp like '2025-04-16T02:53:57.992' and a message ID like '3784'. To the left of the table is a detailed sidebar with sections for 'フィルター', 'ロググループを検索して選択する', '選択したロググループを少なく表示', 'ログストリームを選択 - オプション', 'ブレイクダ운スを入力', and 'ログイベントのフィルタリング'. The sidebar also includes a 'フィルターを適用' button.

# クロスアカウントオブザーバビリティ × Live Tail

異なる環境で動いてる同一の処理のエラーのばらつき具合を確認



# Live Tail で可視化

モニタリングアカウントとして分析

エラー箇所を描画

The screenshot shows the AWS CloudWatch Live Tail interface. On the left, a sidebar titled "Live Tail 情報" contains a "期間を強調表示" section with a dropdown set to "最大 5 つの用語を強調表示 (大文字と小文字を区別しない)" and a "error" button. Below it is a "フィルター" section with "ロググループを選択" dropdown containing two entries: "/aws/lambda/MemoryConsumerStack-MemoryConsumerFunction9BC82A94-6alg7Y4sr" and "/aws/lambda/MemoryConsumerStack-MemoryConsumerFunction9BC82A94-hoge001". There are also checkboxes for "選択したロググループを少なく表示" and "ログストリームを選択 - オプション". At the bottom are "フィルターを適用" and "ログイベントのフィルタリング" buttons. The main area displays a table of log events with columns: "タイムスタンプ(ローカル)", "メッセージ", and "ロググループ". The "メッセージ" column shows log entries starting with account IDs like "990:/aws/Lambda/MemoryConsumerStack-MemoryConsumerFunction9BC82A94-6alg7Y4sr" and "517:/aws/lambda/memoryconsumerstack-memor...". Red arrows point from the "error" button and the account ID in the log entries to orange callout boxes.

タイムスタンプ(ローカル)	メッセージ	ロググループ
2025-04-16T02:53:57.992+09:00	2025-04-15T17:53:57.992Z 2654a056-72a6-45fb-8e7f-ce...	990:/aws/Lambda/MemoryConsumerStack-Memor...
2025-04-16T02:53:57.993+09:00	2025-04-15T17:53:57.993Z 2654a056-72a6-45fb-8e7f-ce...	990:/aws/Lambda/MemoryConsumerStack-Memor...
2025-04-16T02:53:59.303+09:00	RequestId: 2654a056-72a6-45fb-8e7f-ce6bfaf4cae Err...	990:/aws/Lambda/MemoryConsumerStack-Memor...
2025-04-16T02:53:59.303+09:00	END RequestId: 2654a056-72a6-45fb-8e7f-ce6bfaf4cae	378
2025-04-16T02:53:59.303+09:00	REPORT RequestId: 2654a056-72a6-45fb-8e7f-ce6bfaf4...	378
2025-04-16T02:53:58.381+09:00	END RequestId: 5a2950f2-f161-4c5d-a500-398d3e076125	426
2025-04-16T02:53:58.381+09:00	REPORT RequestId: 5a2950f2-f161-4c5d-a500-398d3e076...	426
2025-04-16T02:53:58.978+09:00	START RequestId: 73291750-10bd-49c8-956a-801f2a4be0...	426
2025-04-16T02:53:58.979+09:00	2025-04-15T17:53:58.979Z 73291750-10bd-49c8-956a-80...	426
2025-04-16T02:53:58.980+09:00	2025-04-15T17:53:58.980Z 73291750-10bd-49c8-956a-80...	426
2025-04-16T02:53:59.579+09:00	END RequestId: 73291750-10bd-49c8-956a-801f2a4be0e1	426
2025-04-16T02:53:59.579+09:00	REPORT RequestId: 73291750-10bd-49c8-956a-801f2a4be...	426
2025-04-16T02:54:00.187+09:00	START RequestId: 1534e44e-e8c4-4117-b75e-0e6cc6da06...	426
2025-04-16T02:54:00.189+09:00	2025-04-15T17:54:00.189Z 1534e44e-e8c4-4117-b75e-0e...	426
2025-04-16T02:54:00.190+09:00	2025-04-15T17:54:00.190Z 1534e44e-e8c4-4117-b75e-0e...	426
2025-04-16T02:54:00.963+09:00	RequestId: 1534e44e-e8c4-4117-b75e-0e6cc6da066a Err...	426
2025-04-16T02:54:00.963+09:00	END RequestId: 1534e44e-e8c4-4117-b75e-0e6cc6da066a	426
2025-04-16T02:54:00.963+09:00	REPORT RequestId: 1534e44e-e8c4-4117-b75e-0e6cc6da0...	426
2025-04-16T02:53:59.318+09:00	INIT_START Runtime Version: nodejs:18.v63 Runtime V...	378
2025-04-16T02:53:59.888+09:00	START RequestId: 2db7174f-9aef-b91-b3b5-e5f39a6b70...	378
2025-04-16T02:53:59.890+09:00	2025-04-15T17:53:59.890Z 2db7174f-9aef-b91-b3b5-e5...	378

ロググループの先頭にアカウントID



# Logs Insights で集計

The screenshot illustrates the use of AWS Logs Insights for log aggregation and analysis.

**Logs Insights Query Editor:**

- Log Group Selection:** The user has selected the log group `/aws/lambda/MemoryConsumerStack-MemoryConsumerFunction9BC82A94-6alg7Y4srD1Y`.
- Query:** The following Log Insights Query Language (QL) is displayed:

```
1 fields @message
2 | filter @message like /(?i)ERROR/
3 | stats count(*) as ctr by @log
```
- Execution Result:** The query has completed successfully, processing 3 log groups.

**Logs View:**

- Summary Table:** A table shows the count of errors for three log entries. The first entry is highlighted with a red border.

#	@log	ctr
1	039...08:/aws/lambda/MemoryConsumerStack-MemoryConsumerFunction9...	164
2	374...30:/aws/lambda/MemoryConsumerStack-MemoryConsumerFunction9...	209
3	426...17:/aws/lambda/MemoryConsumerStack-MemoryConsumerFunction9...	118
- Log Stream Preview:** The first log entry is expanded, showing its raw log message and timestamp.

**Annotation:** An orange callout points to the table with the text **@log でグループ化** (Grouped by @log).

# Explore Related



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

# Explore related で関連するテレメトリを可視化

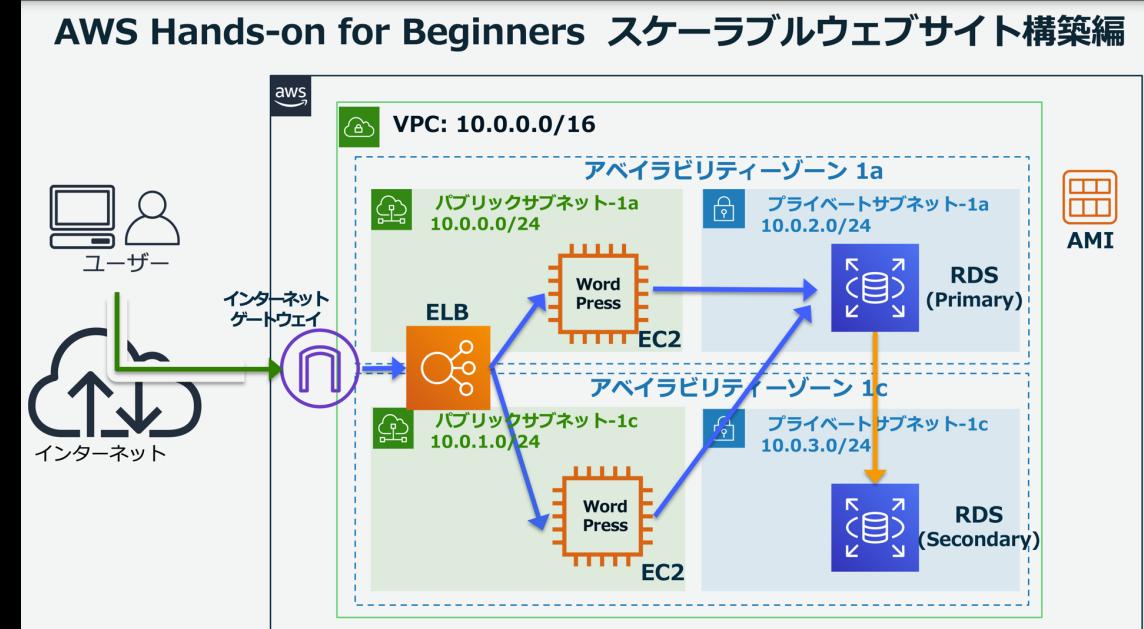
リソース間の関連とテレメトリをトポロジーマップで一元的に表現することで問題特定を迅速に

- 課題
  - 複数のテレメトリを調査するには多数のサービスページの遷移が必要だった
- 実現できること
  - Explore Related では ALB と EC2 など関連があるサービスを自動判定し、トポロジーマップとして可視化
  - 「メトリクスナビゲーション」「マネジメントコンソールのツールバー」など、様々な場所からアクセス可能



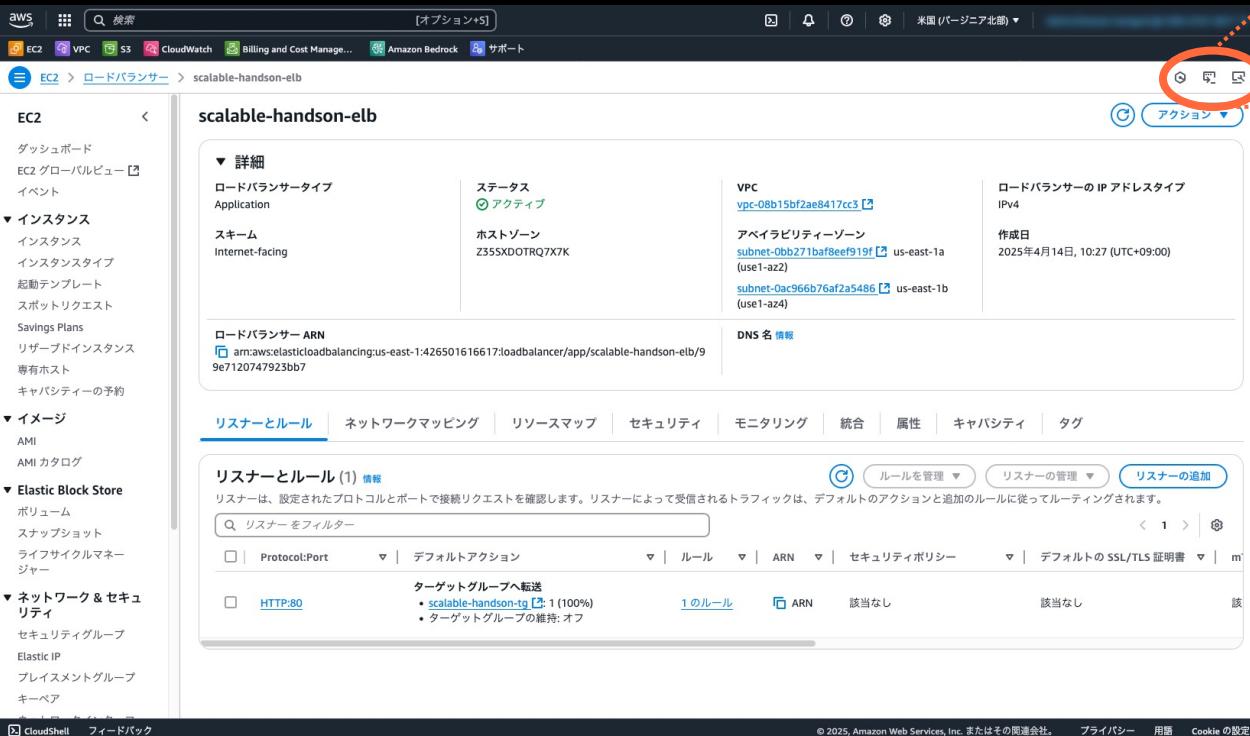
# トポロジーマップの使い所

- トレースマップがなくてもトポロジーマップでシステムのつながりを把握可能
  - 担当者のいないシステムの連携を探る場合
  - サービスの前後のつながりと流量を把握可能

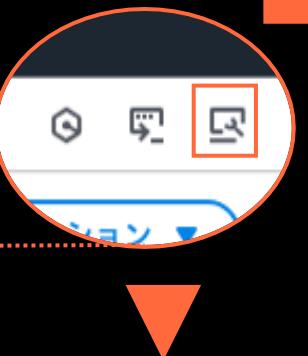


# Explore Related にアクセス

始点となるサービスに狙いを定める(今回はELB)



The screenshot shows the AWS EC2 Load Balancer console for the 'scalable-handson-elb'. The main page displays details like the load balancer type (Application), status (Active), VPC (vpc-08b15bf2ae8417cc3), and IP address type (IPv4). Below this, the 'Listeners and Rules' section is visible, showing a single rule for port 80 targeting the 'scalable-handson-tg' target group. The 'Actions' dropdown menu is highlighted with a red circle.



CloudWatch 以外のサービス  
からでもアクセス可能



The screenshot shows the 'Explore related' interface for the 'scalable-handson-elb' load balancer. The page title is 'scalable-handson-elb' and the tab 'Explore related' is selected. The main content area shows the same load balancer details as the previous screen. At the bottom, there is a search bar with the placeholder 'リソースを検索' (Search resource) and a 'Q' icon, which is highlighted with a red box.



# リソースの選択

ARN を頼りに始点となるリソースを指定する

The screenshot shows the AWS EC2 console interface. On the left, there's a navigation sidebar with sections like EC2, Instances, Images, and Elastic Block Store. The main content area displays details for a specific instance named "scalable-handson-elb". A red box highlights the "ロードバランサー ARN" field, which contains the value "arn:aws:elasticloadbalancing:us-east-1:426501616617:loadbalancer/app/scalable-handson-elb/99e7120747923bb7". To the right of the instance details, a modal window titled "リソースを検索" is open, also containing the same ARN value. The "Use" button at the bottom right of the modal is highlighted with a blue box.



# トレースマップでリソースのつながりを探索

「リソースの前後のつながり」 「それらに紐づくテレメトリ」を確認できる



# Explore related が自動でトポロジーを作る仕組み

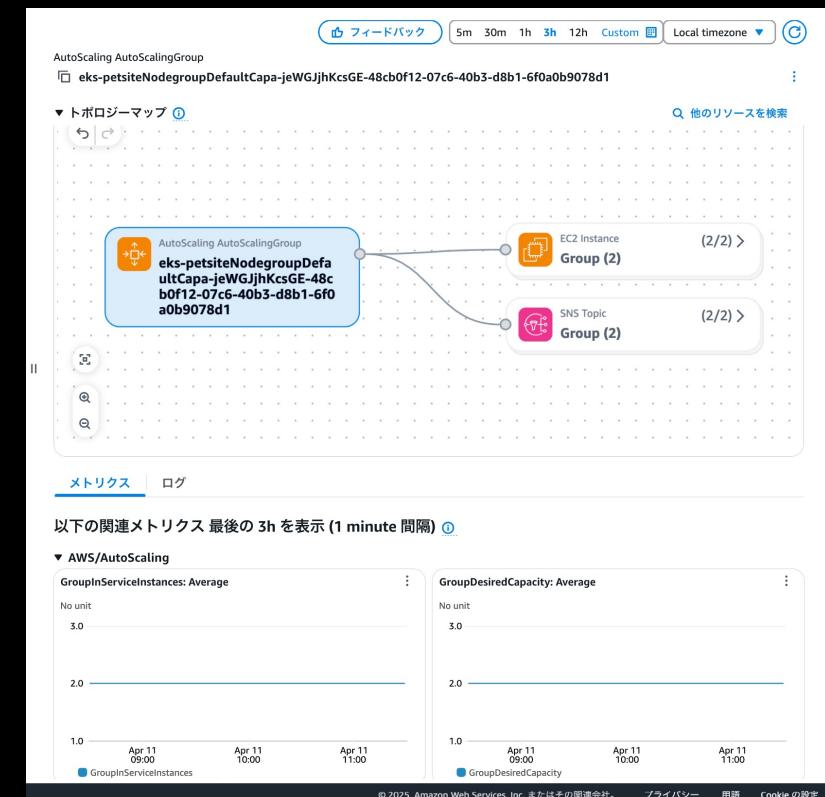
AWS リソースが発生させるメトリクスとログには、エンティティ情報が付与

Auto Scaling Group と関連付けられた EC2 インスタンスは  
トポロジーマップの表示可能

AWS Service	Resource	Metrics	Logs
DynamoDB Accelerator	AWS::DAX::Cluster	⌚	⌚
Amazon EC2	AWS::EC2::CapacityReservation	⌚	⌚
Amazon EC2	AWS::EC2::Instance	⌚	⌚
Amazon EC2	AWS::EC2::FlowLog	⌚	⌚
Amazon EC2	AWS::EC2::NATGateway	⌚	⌚
Amazon EC2	AWS::EC2::NetworkInterface	⌚	⌚
Amazon EC2	AWS::EC2::Subnet	⌚	⌚
Amazon EC2	AWS::EC2::TransitGateway	⌚	⌚
Amazon EC2	AWS::EC2::TransitGatewayAttachment	⌚	⌚
Amazon EC2	AWS::EC2::VerifiedAccessInstance	⌚	⌚
Amazon EC2	AWS::EC2::Volume	⌚	⌚
Amazon EC2	AWS::EC2::VPC	⌚	⌚
Amazon EC2	AWS::EC2::VPNConnection	⌚	⌚
Amazon EC2 Auto Scaling	AWS::AutoScaling::AutoScalingGroup	⌚	⌚
AWS Elastic Beanstalk	AWS::ElasticBeanstalk::Environment	⌚	⌚

エンティティ対応しているリソース

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/services-with-related-telemetry.html>



# 注意点

- すべてのサービスをEnd to End で見れるわけではない
  - 設定値に基づいて自動でトポロジーを作成しているため、前後関係しか見えない

# まとめ

- CloudWatch logs の フィールドインデックス機能
  - logs insights クエリのコストとパフォーマンス効率化
- クロスアカウントオブザーバビリティ × Live Tail
  - 一つのアカウントでテレメトリのシームレスな分析が可能
- Explore Related
  - サービスページを移動しなくてもテレメトリの探索が可能

# Thank you



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

# Amazon CloudWatch Application Signals ではじめるバーンレートアラーム

Yoshi Yamaguchi (@ymotongpoo)  
Senior Developer Advocate  
Amazon Web Services Japan, G.K.



# 自己紹介

山口 能迪 (やまぐち よしふみ)

アマゾンウェブサービスジャパン合同会社  
シニアデベロッパーアドボケイト

## 専門領域

- オブザーバビリティ
- SRE全般



@ymotongpoo





# アジェンダ

- バーンレート計測の重要性
- Application Signals による SLO モニタリング

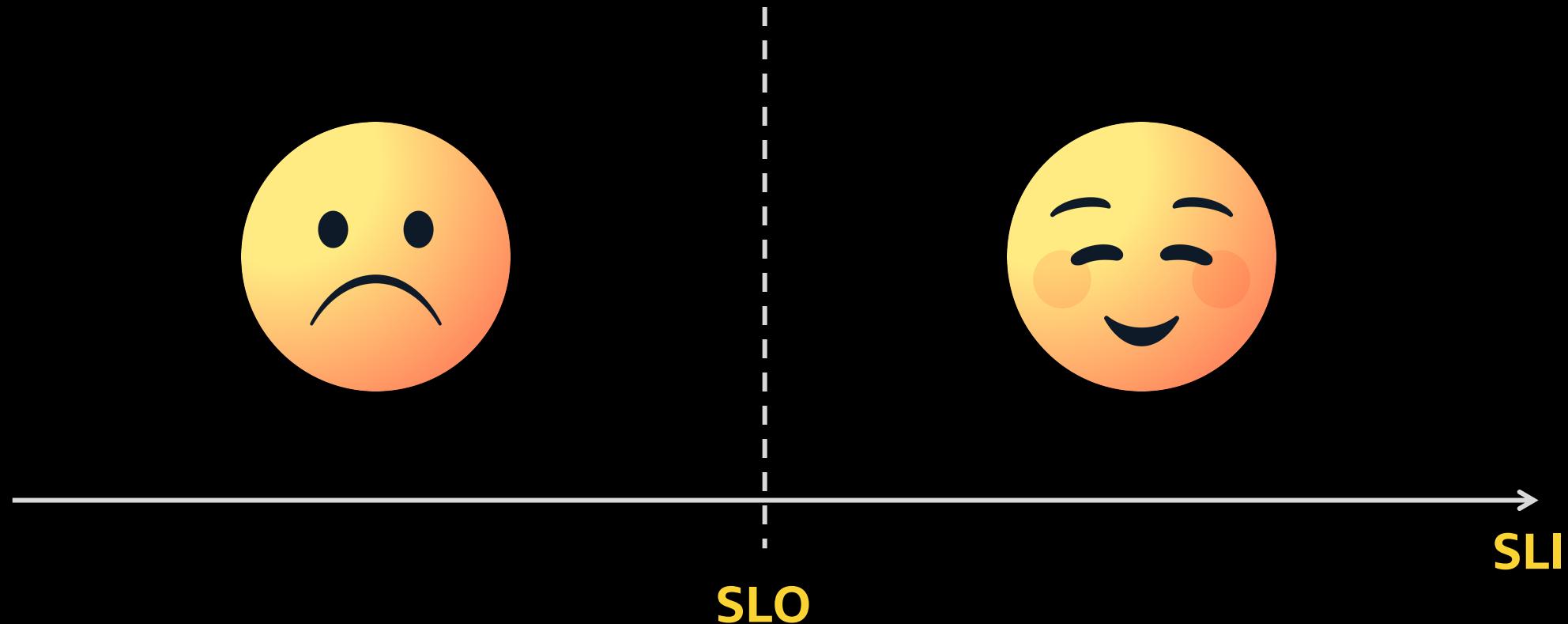
# バーンレート計測の重要性



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# サービスレベル指標 SLI ／ サービスレベル目標 SLO

信頼性 = ユーザーがサービスに期待する性能品質 = システムメトリクスではない



# サービスレベル指標 SLI / サービスレベル目標 SLO

**SLI** = ユーザーがサービスに期待する性能品質

$$\text{SLI} = \frac{\text{良いイベント} / \text{時間}}{\text{全イベント} / \text{時間}}$$

例1: チケットの予約にかかる時間 (レイテンシー)

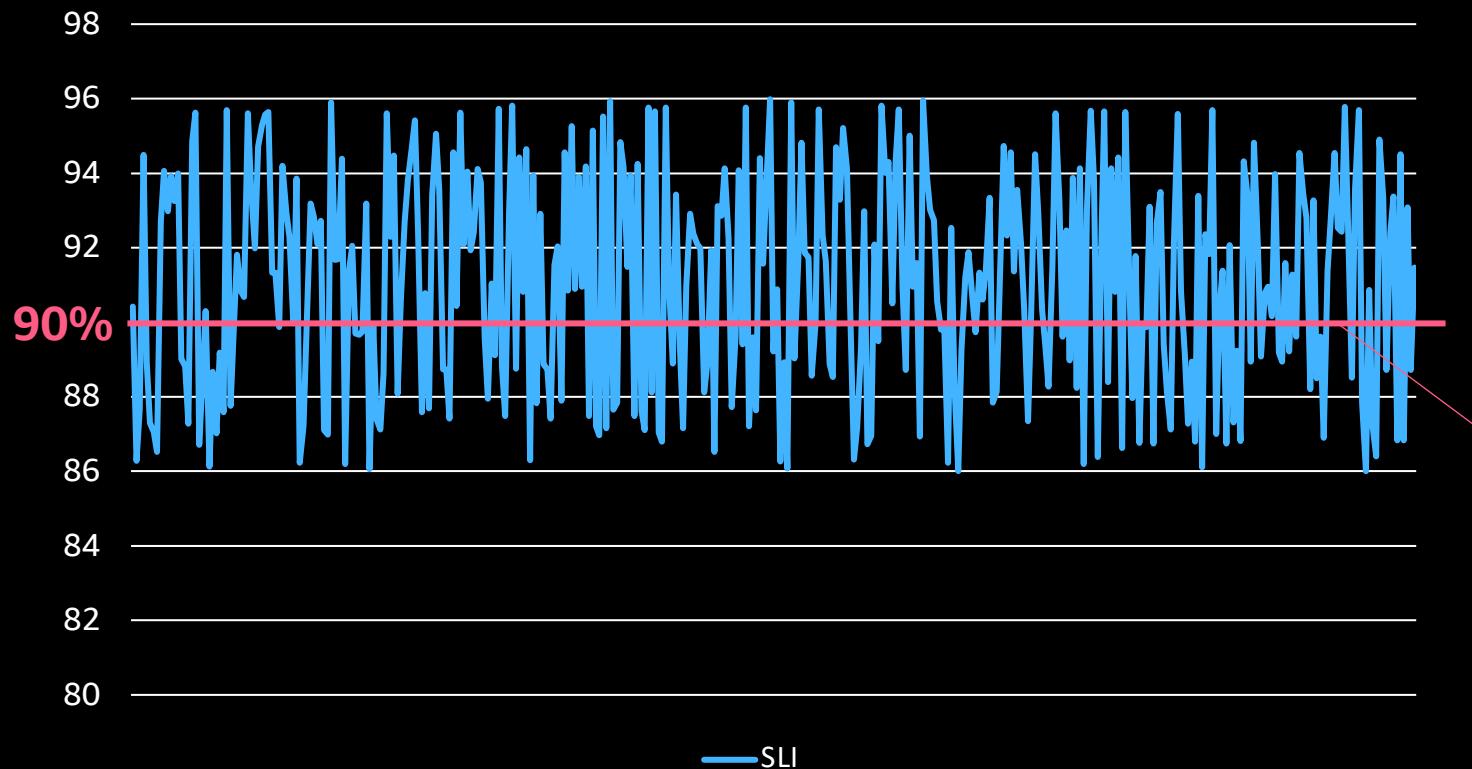
$$\frac{\text{500ms以内のレスポンスの数}}{\text{全レスポンスの数}}$$

例2: 予約ページにアクセスできる時間 (可用性)

$$\frac{\text{アクセスできてた時間}}{\text{計測中の全期間}}$$

# SLOだけ見てれば安心！なのか…？

良いレイテンシーの割合



**SLI:** レイテンシーが1800ms以下  
**SLO:** 28日間で90%

しきい値でアラームを設定してたら…

# アラーム疲れ alarm fatigue

重要なアラームが多すぎることで  
疲弊し、重要なアラームを見逃してしまった状況。

アラームはアクションが必要な状況でのみ発報すべき

[WAF: OPS08-BP04 Create actionable alerts](#)

c.f.

- Amazon DevOps Guru
- Amazon CloudWatch 複合アラーム



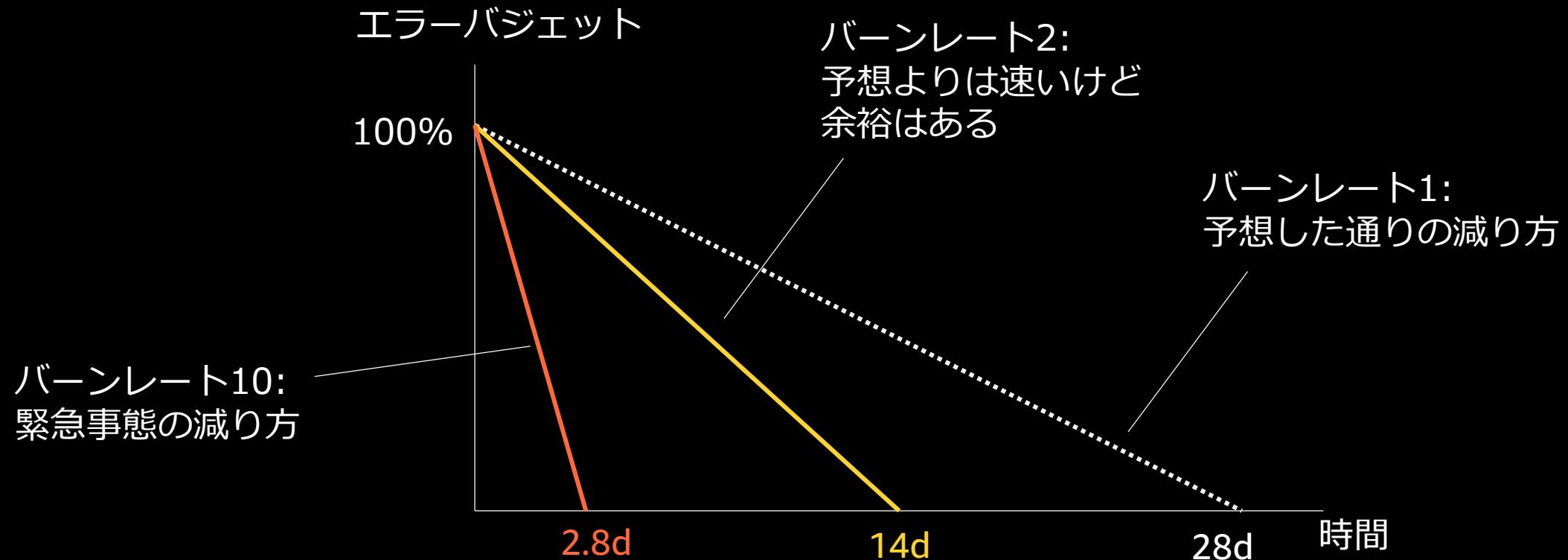
# エラーバジェット

エラーバジェット = 100% - SLO



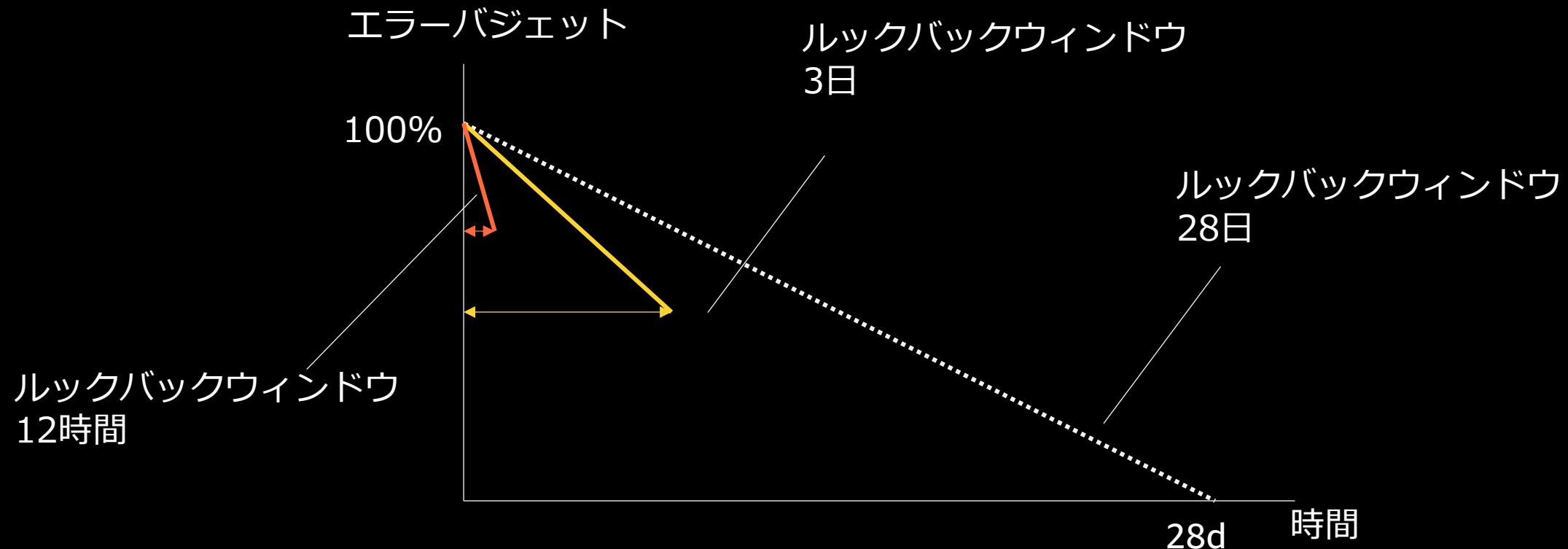
# バーンレートアラーム

思ってたよりエラーバジェットが速く減ってたら知らせる

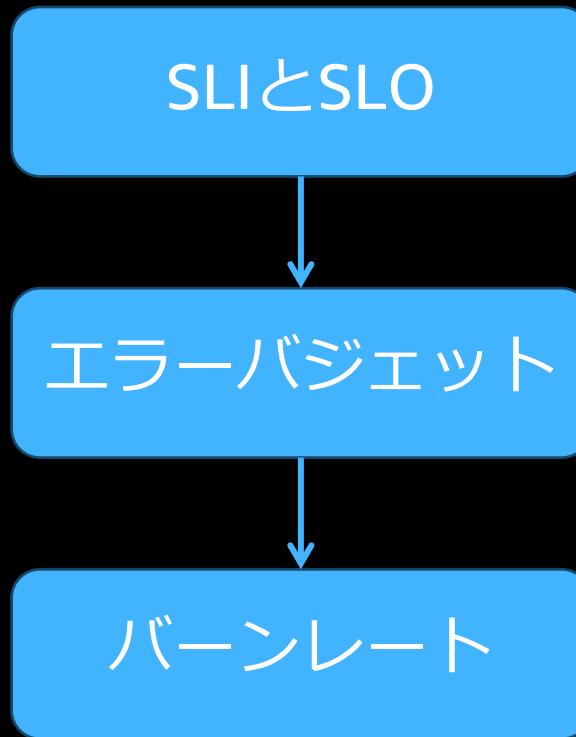


# ルックバックウィンドウ

消費速度が速い場合は短い時間で気が付かないと意味がない



# 関係図



指標と計測期間と目標値を決めるだけ  
(割り算の計算式を決めるだけ)

一定期間のデータをローリングで計算し続ける  
(1次計算)

エラーバジェットの差分をローリングで計算し続ける  
(2次計算)

バーンレートアラームを自分で実装するのは多少手間がかかる

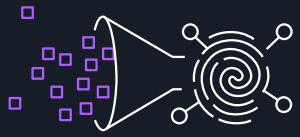
# Application Signals による SLO モニタリング



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Amazon CloudWatch Application Signals

アプリケーションの状態を把握・分析するための情報を自動で収集・可視化



テレメトリデータ  
自動収集



サービスの検出/  
トポロジー可視化



ダッシュボード提供  
トレース・メトリクスの  
確認



SLO  
モニタリング



CloudWatch Synthetics  
/ CloudWatch RUM  
との連携

# Application Signals がサポートされる環境

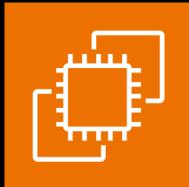
## アーキテクチャ



**Amazon EKS**  
EC2 / Fargate



**Amazon ECS**  
EC2 / Fargate  
awsvpc ネットワーク  
モードのみ



**Amazon EC2**  
コンテナ以外の  
アプリケーションも  
実装可能

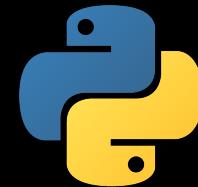


**AWS Lambda**

## プログラム言語



**Java**



**Python**



**Node.js**



**.NET**

8, 11, 17

3.8+

14, 16, 18,  
20, 22

.NET 6, 8

.NET Framework  
4.6.2+

# Application Signals の有効化の例

Lambda ではワンクリックで設定可能

ここをチェックするだけ →

The screenshot shows the 'Monitoring Tools' configuration page for a Lambda function named 'my-simple-sample'. The 'CloudWatch Applications Signals and AWS X-Ray' section is highlighted with a blue border. Inside this section, the 'Enable Application Signals' checkbox is checked (indicated by a blue outline). A blue arrow points from the text 'ここをチェックするだけ' (Check here) to this checkbox.

モニタリングツールを編集

Amazon CloudWatch 情報

デフォルトでは、Lambda 関数は CloudWatch Logs ストリームと標準メトリクスを生成します。

ログとメトリクス (デフォルト)

CloudWatch アプリケーションシグナルと AWS X-Ray 情報

CloudWatch アプリケーションシグナルと AWS X-Ray は、アプリケーションのパフォーマンスと状態を監視するのに役立ちます。

アプリケーションシグナル 情報

アプリケーションシグナルは、リクエスト、可用性、レイテンシー、エラー、障害などのアプリケーショントレースとメトリクスを収集するのに役立ちます。これらのトレースとメトリクスは [CloudWatch コンソール](#) で確認できます。

有効化

サポートされているランタイム: nodejs18.x, nodejs20.x, nodejs22.x, python3.10, python3.11, python3.12, python3.13, java11, java17, java21, dotnet6, dotnet8. (現在のランタイム: nodejs22.x)

Lambda サービストレース 情報

Lambda サービスの X-Ray トレースは、アプリケーション内のリクエストのパスを追跡するのに役立ちます。これらのトレースは [X-Ray コンソール](#) で確認できます。

有効化

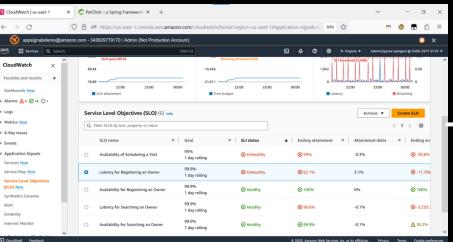
CloudWatch Lambda インサイト 情報

オンにすると、CPU 時間、メモリ、ディスク、ネットワーク使用量などのシステムレベルのメトリクスを収集できます。

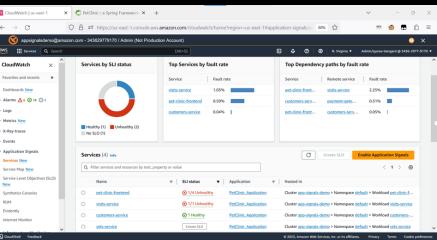
拡張モニタリング

# Application Signals のダッシュボード

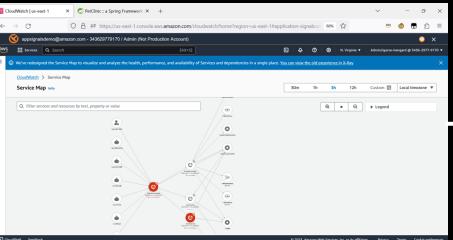
SLO  
モニタリング



サービス  
ダッシュボード



サービスマップ

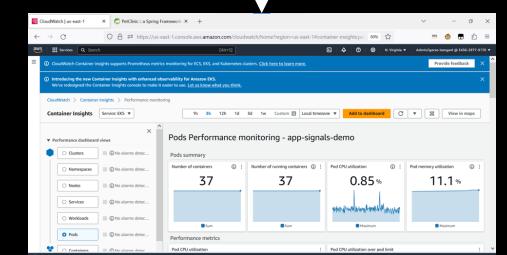


サービス

カナリア※

トレース

インサイト、ログ



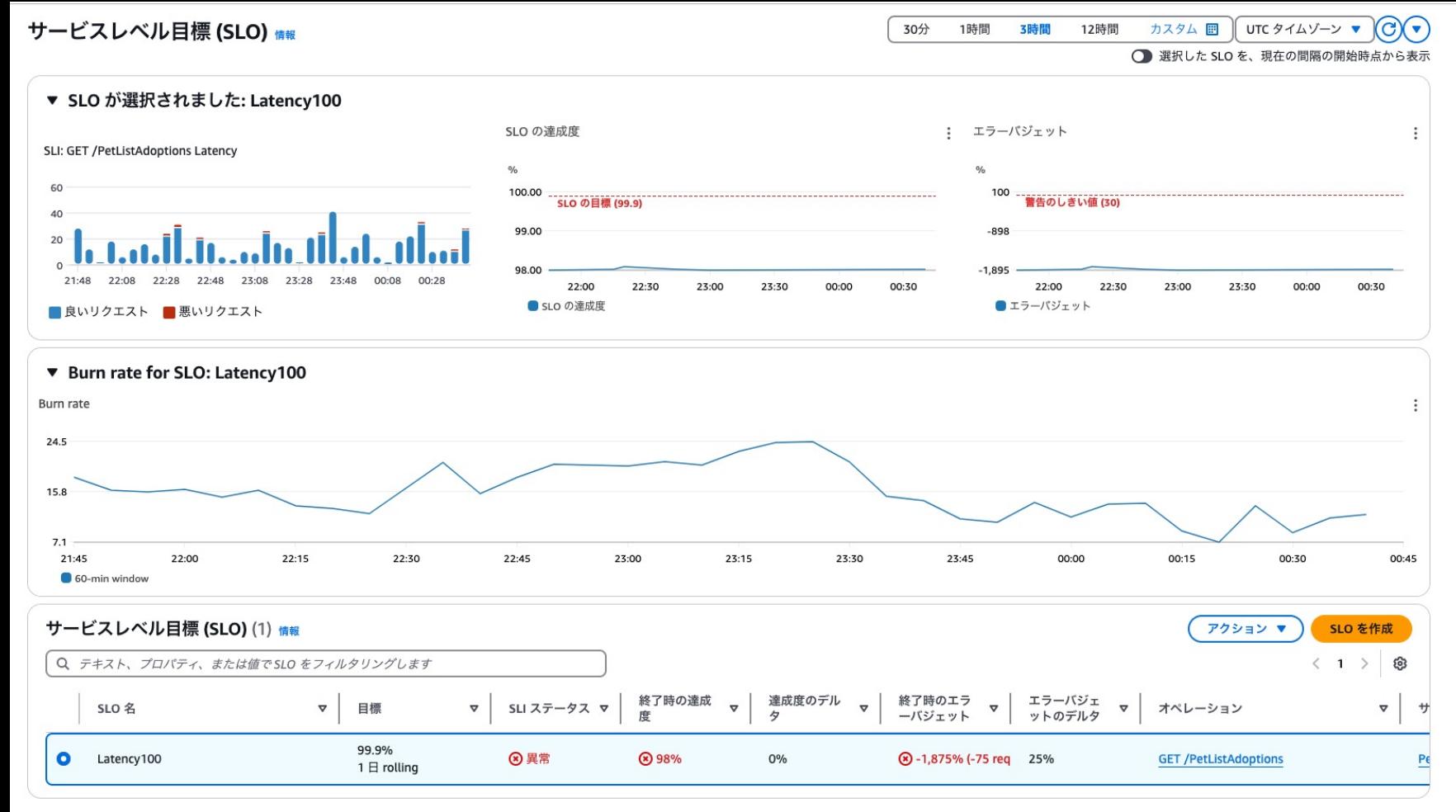
※必ず利用しなければならないわけではありません



# SLOダッシュボード



SLIのSLOに対する現状を把握、バーンレートも見える



# デモ: SLI→SLO→バーンレートアラーム

# SLI の設定

Application Signals ではサービスの操作に関わる指標を SLI とすることを想定

サービスレベル目標 (SLO)

> 「サービスレベル目標 (SLO)」セクション



サービスレベル目標 (SLO) (0) 情報

SLO 名 目標 SLI ステータス ▲ Latest attainment 達成度のデルタ ▼ Latest error budget

SLO なし  
SLO は作成されていません

SLO の詳細 [?] SLO を作成

アクション ▾ < 1 > ⚙️

サービス  
> 「サービス」セクション



サービス (2) 情報

名前 SLI ステータス サービスの可用性 アプ

<input checked="" type="radio"/> dotnet-cart-api	SLO を作成	36.3%	-
<input type="radio"/> dotnet-delivery-api	SLO を作成	100%	-



サービスレベル目標 (SLO)

> サービスレベル目標 (SLO) を作成

## サービスレベル指標 (SLI) を設定 情報

サービスレベルインジケータ (SLI) は、サービスレベル目標 (SLO) を達成しているかどうかを判断するために使用されるメトリックです。

### タイプ

サービスオペレーション  
検出されたサービスから得られたサービスオペレーションメントリクエストの使用に関する信頼性の目標を設定します。

Dependency  
Set a reliability target using a dependency metric from a discovered service.

CloudWatch メトリクス  
CloudWatch メトリクスを使用して信頼性の目標を設定します。

### サービスを選択 情報

どのサービスについてメトリクスを設定しますか?

dotnet-cart-api

### オペレーションを選択 情報

どのサービスオペレーションの信頼性を測定しますか?

POST /cart

### 計算方法を選択 情報

カスタマーエクスペリエンスの測定にどの計算方法を使用したいですか?

リクエスト  
すべてのリクエストに対して適切なリクエストの数を計算します。

期間  
指定された時間間隔内の良好な期間の数を計算します。

### 条件を設定

許容できるエクスペリエンスであることを示すために、どのメトリクスとしきい値を使用しますか?

レイテンシー ▾ は、次以下である必要があります: 1800 ミリ秒.

POST /cart リクエストは、レイテンシーが 1800ミリ秒 に対して 次以下: である場合に、良好かつ正常であるとみなされます。

# SLO の設定

フォームに沿って入力していくだけ

**サービスレベル目標 (SLO) を設定** 情報

サービスレベル目標 (SLO) は、SLI を使用して、顧客の期待に対するサービスの信頼性を定義します。

**間隔を設定** 情報

どのくらいの間隔で SLO を測定したいですか？

28 ローリング 日 ▾

**達成目標を設定** 情報

この期間における SLO 達成目標はどのようなものですか？

90 %.

パーセンテージは 0~100 である必要があります。

**警告のしきい値を設定** 情報

次の場合に SLO を [警告状態] としてマークします: エラーバジェット は次以下です:

30 %

パーセンテージは 0~100 である必要があります。

# バーンレートアラーム

CloudWatch アラームまで設定

まずバーンレートを計算する期間を設定

バーンレートを設定 - オプション 情報

バーンレートは、エラーバジェットをどれくらいの速さで消費しているかを示します。

ルックバックウィンドウ(分単位)

2880

480

バーンレートをさらに追加

最大 10 個のバーンレートをセットアップできます



各ルックバックウィンドウでバーンレートアラームを設定

バーンレートアラームを設定 - オプション 情報

バーンレートが設定されたしきい値を超えたときに通知する CloudWatch アラームを作成します

自動的に作成するバーンレートアラームを選択します:

ルックバックウィンドウの分数分の 2880 バーンレートアラーム  
**Burn rate threshold**  
Set up threshold to alarm on the burn rate

消費予算のパーセンテージでバーンレートしきい値を設定

バーンレートしきい値を直接設定  
Will fire every time burn rate is larger than  
2  
Value must be a number larger than 1

SNS トピック

通知を受信する SNS (Simple Notification Service) トピックを定義します。

既存の SNS トピックを選択

新しいトピックを作成

新しいトピックを作成...

トピック名は一意である必要があります。

cart-api-post-root-latency-90-slow-burn

SNS トピック名は最大 256 文字で、英数字、ハイフン (-)、アンダースコア (\_) のみを使用できます。

通知を受け取る E メールアンドポイント...

メールアドレスのカンマ区切りリストを追加します。各アドレスは上記のトピックに対するサブスクリプションとして追加されます。

team-cart-api@example.com  
user1@example.com, user2@example.com.

# まとめ

バーンレートアラームは理由がわかりやすい！

Application Signals でバーンレートアラームを設定しましょう！



# Thank you!

Any questions?

**Yoshi Yamaguchi**  
@ymotongpoo



A LinkedIn profile card for Yoshi Yamaguchi. The background image is a sunset over a city skyline, likely Tokyo, with the Tokyo Tower visible. The profile picture shows a young man with glasses, wearing a dark t-shirt, resting his chin on his hand. To the right of the profile picture are three circular icons: a person icon, a search icon, and a 'Follow' button. Below the profile picture, the name 'Yoshi Yamaguchi' is displayed in bold, followed by the handle '@ymotongpoo'. A bio section includes the text 'Senior Developer Advocate at AWS | Observability, SRE/DevOps, Go | ex-Google | opinions=mine | bsky: [@ymotongpoo.com](#)'. Below the bio are two small icons: a briefcase for 'Cloud Services & Solutions' and a location pin for '日本 東京'. At the bottom of the card, it says 'Joined April 2007' and shows the follower count '947 Following' and '13.6K Followers'.

# サーバレス、コンテナ、データベース 特化型機能をご紹介。 CloudWatch をもっと使いこなそう！

堀 貴裕

ソリューションアーキテクト



# 自己紹介



氏名	堀 貴裕 (ほり たかひろ)
役職	主に製造業のお客様をご支援 技術担当 Solutions Architect
好きな分野 AWS サービス	オブザーバビリティ、Amazon CloudWatch

# アジェンダ

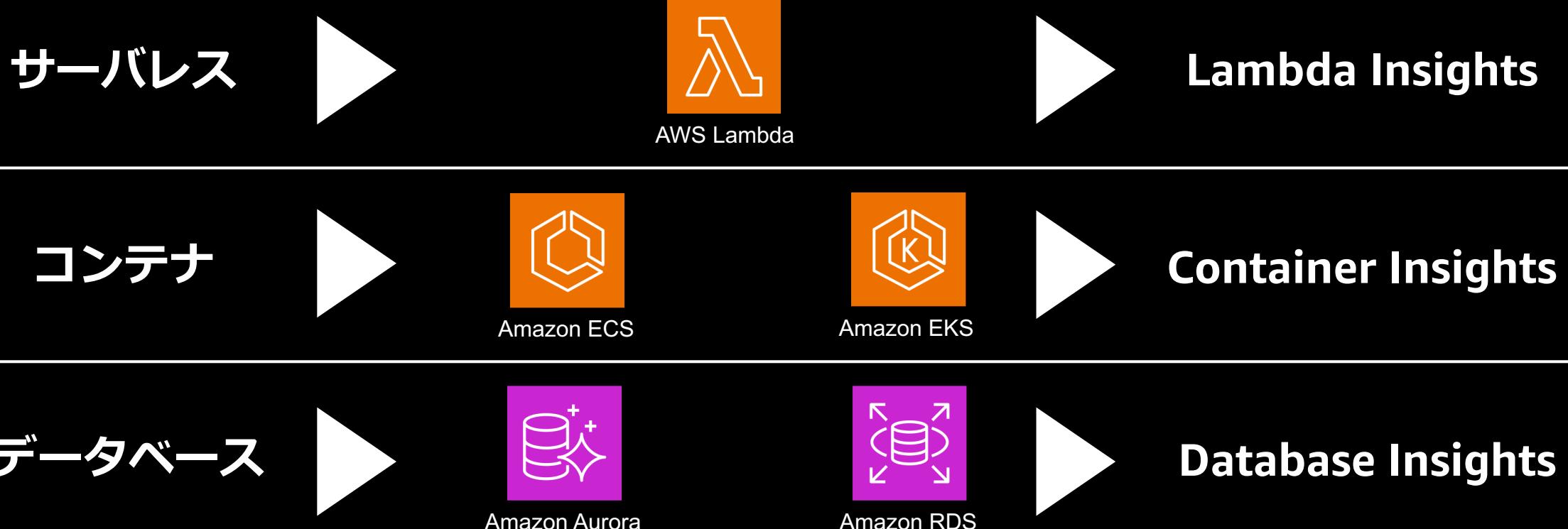
- 本日のゴール
- Amazon CloudWatch Lambda Insights
- Amazon CloudWatch Container Insights
- Amazon CloudWatch Database Insights
- まとめ

# アジェンダ

- 本日のゴール
- Amazon CloudWatch Lambda Insights
- Amazon CloudWatch Container Insights
- Amazon CloudWatch Database Insights
- まとめ

# 本日のゴール

- ・ サーバレス、コンテナ、データベースを運用する皆様がより深い障害分析、最適化を行うための CloudWatch の機能を知っていただく



# 3つの機能でできること

- ・ サーバレス、コンテナ、データベースでデフォルトのメトリクスやログでは分析しづらい情報を収集、可視化できる



「Lambda でコールドスタートが起きてるみたいだけど、いつ起きているのかログを仕込むのは大変そうだ、、」

➡ 共通で取得、可視化したい情報が存在するが実装が大変



「Lambda insights だと自動でコールドスタートの発生、遅延を計測し、ログとの紐付けもできます！」

➡ AWS が情報取得、可視化を機能としてご用意

# アジェンダ

- 本日のゴール
- Amazon CloudWatch Lambda Insights
- Amazon CloudWatch Container Insights
- Amazon CloudWatch Database Insights
- まとめ

# Lambda Insights のユースケース

- 主な活用シーン

- 1 パフォーマンス問題の特定と解決  
メモリリーク、CPU 使用の急増、レイテンシーの増大
- 2 コスト最適化  
過剰なメモリ割り当てや非効率な関数実行
- 3 アプリケーションの健全性監視  
異常な Lambda 関数の早期発見
- 4 トラブルシューティングの効率化  
依存するリソースの探索の効率化



# Lambda Insights の主要機能

- 主要機能

1

## 詳細なメトリクス収集

CPU 使用時間、メモリ使用率、/tmp の使用率など詳細なメトリクスを自動収集

2

## 詳細なパフォーマンス分析

コードスタート、エラー、関数実行コストなどの分析が可能

3

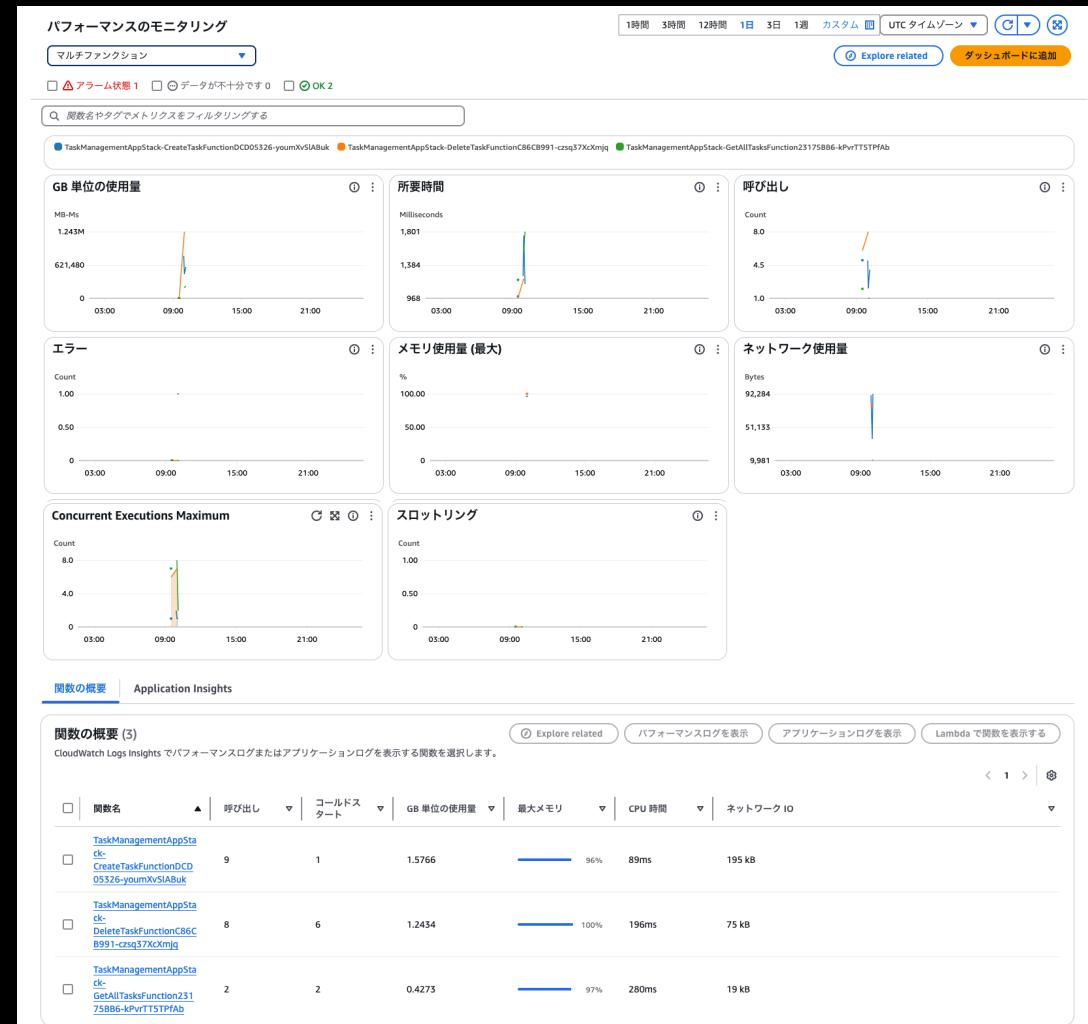
## Lambda 拡張機能として簡単に実装

Lambda レイヤーとして簡単に設定

4

## 自動ダッシュボード

事前に用意されたダッシュボードで分析最適化が可能



ダッシュボード例



# コードスタートの分析例

1

コードスタート  
発生の関数の特定

2

実行時間など概要を  
ダッシュボードで把握

3

初期実行時間など  
パフォーマンスログを  
分析

The figure consists of three vertically stacked screenshots from AWS CloudWatch Logs Insights and Metrics Insights.

**Screenshot 1: CloudWatch Logs Insights - Function Overview**

This screenshot shows a table of Lambda functions. The second function, "TaskManagementAppStack-DeleteTaskFunctionC86C", is highlighted with a red box. The table includes columns for Function Name, Invocation Count, Cold Start Count, GB Unit Usage, Maximum Memory, CPU Time, and Network IO.

関数名	呼び出し	コードスタート	GB 単位の使用量	最大メモリ	CPU 時間	ネットワーク IO
TaskManagementAppStack-CreateTaskFunctionDCD	9	1	1.5766	96%	89ms	195 kB
<b>TaskManagementAppStack-DeleteTaskFunctionC86C</b>	8	6	1.2434	100%	196ms	75 kB

**Screenshot 2: CloudWatch Metrics Insights - Performance Monitoring**

This screenshot shows a dashboard for TaskManagementAppStack-DeleteTaskFunctionC86C. It includes three line charts: "呼び出しとエラー" (Invocations and Errors), "所要時間" (Execution Time), and "スロットリング" (Throttling). The "所要時間" chart is highlighted with a red box. The x-axis represents time from 03:00 to 21:00.

**Screenshot 3: CloudWatch Logs Insights - Log Analysis**

This screenshot shows a log analysis interface with a table of logs. The first few log entries are:

#	@timestamp	@message	@logStream
2	2025-04-10T10:09:34.0...	END RequestId: 5f953dfa-139c-4b95-a139-3886b6f44b79	2025/04/10/\$LATEST d7c692ada76f4e0fa7627a166404db
3	2025-04-10T10:09:34.0...	2025-04-10T10:09:34.00002 5f953dfa-139c-4b95-a139-3886b6f44b79 ERROR ..	2025/04/10/\$LATEST d7c692ada76f4e0fa7627a166404db
4	2025-04-10T10:09:33.8...	2025-04-10T10:09:33.8572 5f953dfa-139c-4b95-a139-3886b6f44b79 INFO ..	2025/04/10/\$LATEST d7c692ada76f4e0fa7627a166404db
5	2025-04-10T10:09:33.8...	2025-04-10T10:09:33.8572 5f953dfa-139c-4b95-a139-3886b6f44b79 INFO T..	2025/04/10/\$LATEST d7c692ada76f4e0fa7627a166404db
6	2025-04-10T10:09:33.8...	START RequestId: 5f953dfa-139c-4b95-a139-3886b6f44b79 Version: \$LATE...	2025/04/10/\$LATEST d7c692ada76f4e0fa7627a166404db
7	2025-04-10T10:09:29.1...	REPORT RequestId: 53273a5c-b413-4b53-e6e2-fe10fae6b22a Duration: 266...	2025/04/10/\$LATEST d7c692ada76f4e0fa7627a166404db



# アジェンダ

- 本日のゴール
- Amazon CloudWatch Lambda Insights
- Amazon CloudWatch Container Insights
- Amazon CloudWatch Database Insights
- まとめ

# Container Insights のユースケース

- 主な活用シーン

1

様々な粒度での問題を迅速に特定

クラスター、インスタンス、サービス、タスク  
コンテナレベルでの問題の検知

2

コスト最適化

リソース割り当て、スケーリングの過不足を解消

3

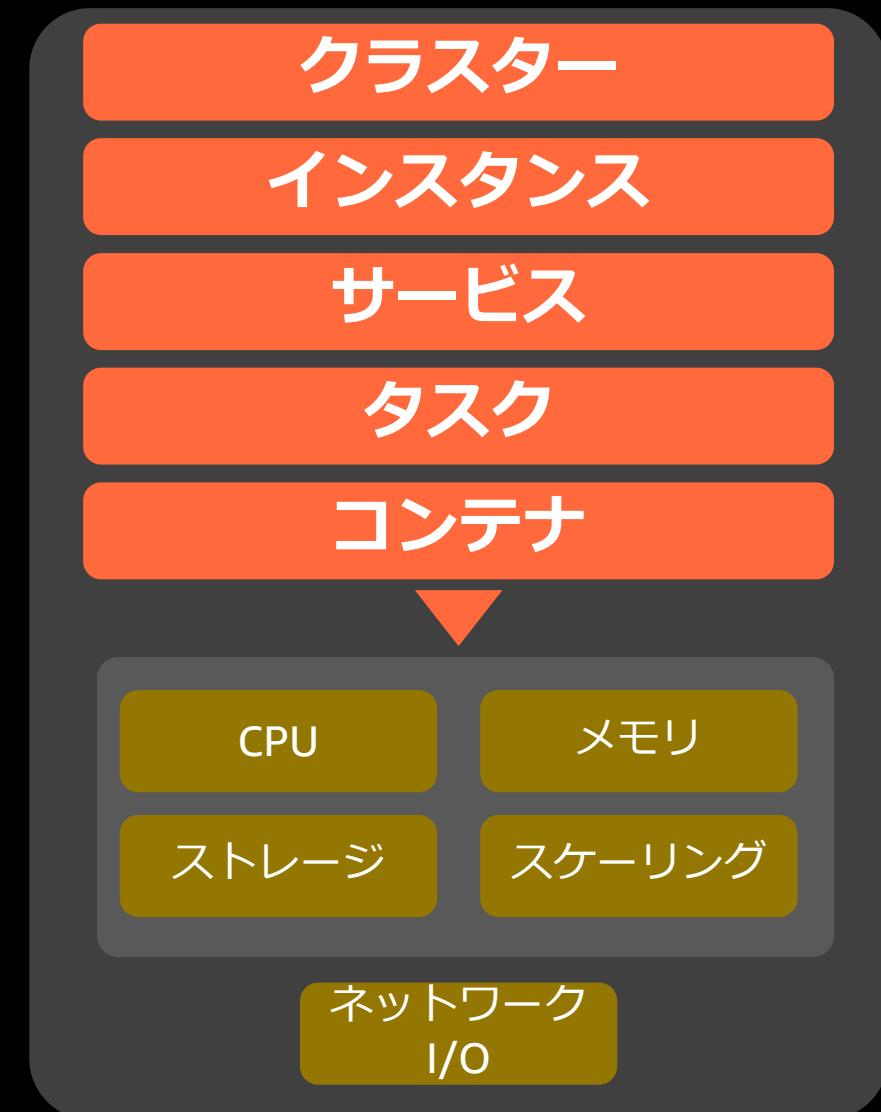
アプリケーションの健全性監視

異常なサービス (コンテナ) の早期発見

4

トラブルシューティングの効率化

サービス間の関連性を可視化し  
ボトルネックの発見



# Container Insights の主要機能

- 主要機能

1

## 複数の粒度でのメトリクス収集

クラスター、インスタンス、サービス、タスク  
コンテナレベルでのメトリクス自動収集

2

## 簡単セットアップで即時有効化

ECS, EKS (on Fargate, EC2) で数クリックで有効化  
Prometheus にも対応

3

## サービス間の相関の可視化

サービス間の相関とボトルネックをマップで表示

4

## 自動ダッシュボード

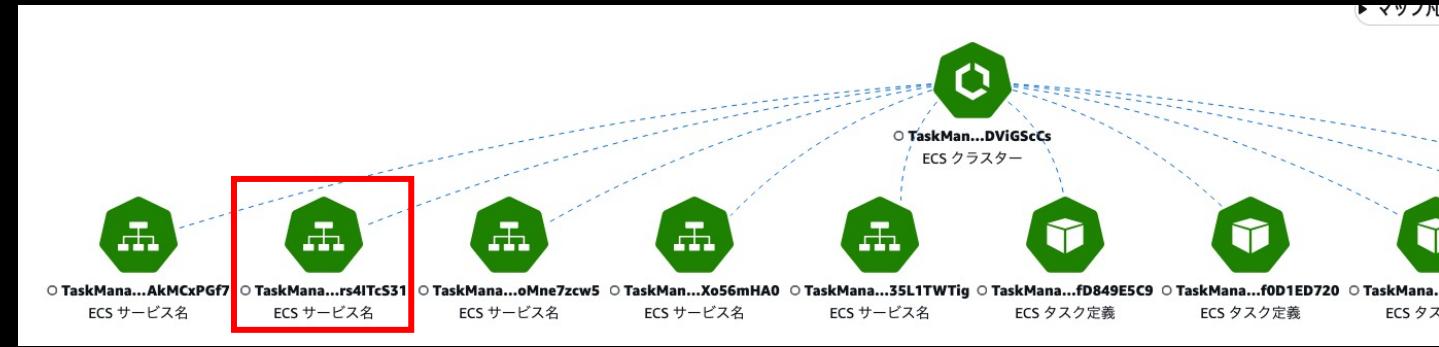
事前に用意されたダッシュボードで  
複数粒度での分析・可視化



# コンテナワーククロードでの問題特定例

1

コンテナマップで問題のあるサービスを特定



2

パフォーマンスダッシュボードで問題を特定



3

ログを分析し問題のある箇所を特定

A logs table showing 20 entries. The first entry, timestamped 2025-04-10T10:09:34.0, is highlighted with a red box. The table includes columns for #, @timestamp, @message, and @logStream.

#	@timestamp	@message	@logStream
▶ 2	2025-04-10T10:09:34.0...	END RequestId: 5f953dfa-139c-4b95-a139-3886b6f44b79	2025/04/10/[\$LATEST]d7c692ada76f4e0faf7627a166404d8 [2]
▶ 3	2025-04-10T10:09:34.0...	3035 04-10T10:09:34.0992 5f953dfa-139c-4b95-a139-3886b6f44b79	2025/04/10/[\$LATEST]d7c692ada76f4e0faf7627a166404d8 [2]
▶ 4	2025-04-10T10:09:33.8572	5f953dfa-139c-4b95-a139-3886b6f44b79 INFO E...	2025/04/10/[\$LATEST]d7c692ada76f4e0faf7627a166404d8 [2]
▶ 5	2025-04-10T10:09:33.8572	5f953dfa-139c-4b95-a139-3886b6f44b79 INFO T...	2025/04/10/[\$LATEST]d7c692ada76f4e0faf7627a166404d8 [2]
▶ 6	2025-04-10T10:09:33.8...	START RequestId: 5f953dfa-139c-4b95-a139-3886b6f44b79 Version: \$LATE...	2025/04/10/[\$LATEST]d7c692ada76f4e0faf7627a166404d8 [2]
▶ 7	2025-04-10T10:09:29.1...	REPORT RequestId: 53273a5c-b413-4b53-9e62-fe10faeb22a Duration: 266...	2025/04/10/[\$LATEST]d7c692ada76f4e0faf7627a166404d8 [2]



# アジェンダ

- 本日のゴール
- Amazon CloudWatch Lambda Insights
- Amazon CloudWatch Container Insights
- Amazon CloudWatch Database Insights
- まとめ

# Database Insights のユースケース

- 主な活用シーン

- 1 パフォーマンスのボトルネック特定  
どこがボトルネックなどか  
アプリケーションまで追跡
- 2 SQL クエリの最適化  
SQL クエリのパフォーマンス改善のための  
情報を提供
- 3 コスト最適化  
負荷や使用率からキャパシティプランニング
- 4 トラブルシューティングの効率化  
呼び出し元のサービスを特定し問題解決を  
迅速化



# Database Insights の機能

- 主要機能

1

## 詳細な SQL の分析

DB ロード、トップ SQL などボトルネックを可視化し、統計情報から詳細な分析が可能

2

## 簡単セットアップで即時有効化

Aurora, RDS で数クリックでセットアップ

3

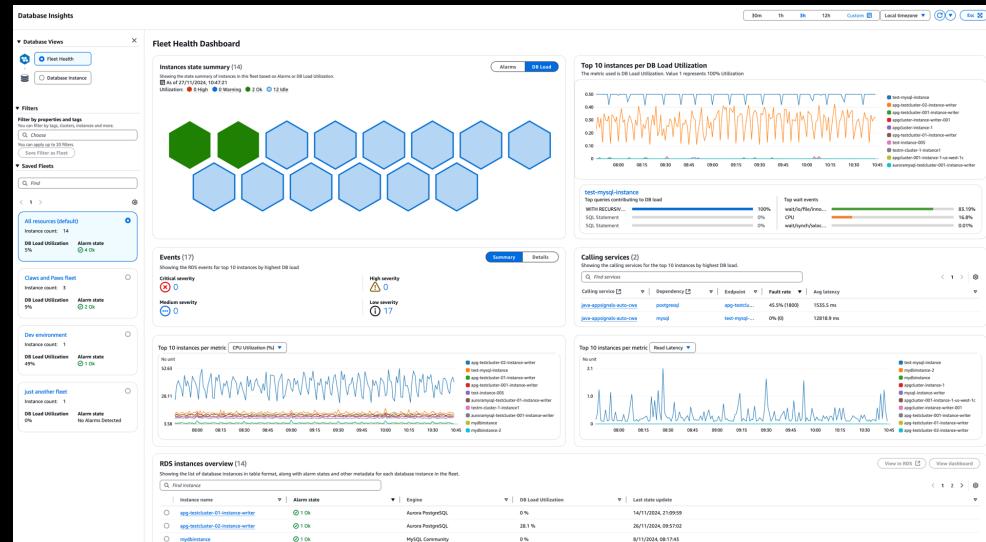
## 依存関係の可視化

依存するサービスとデータベースをマッピング

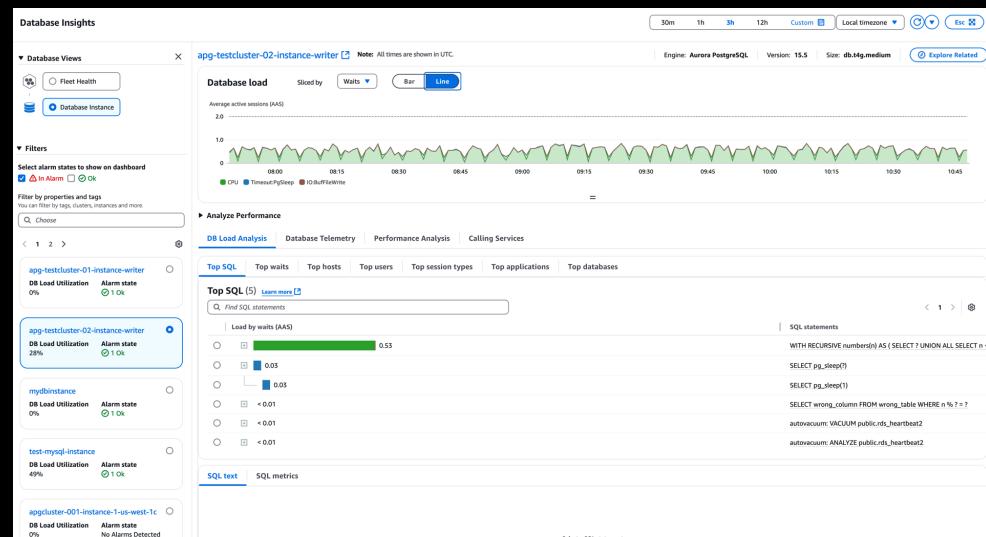
4

## 自動ダッシュボード

事前に用意されたダッシュボードで分析最適化が可能



Fleet View



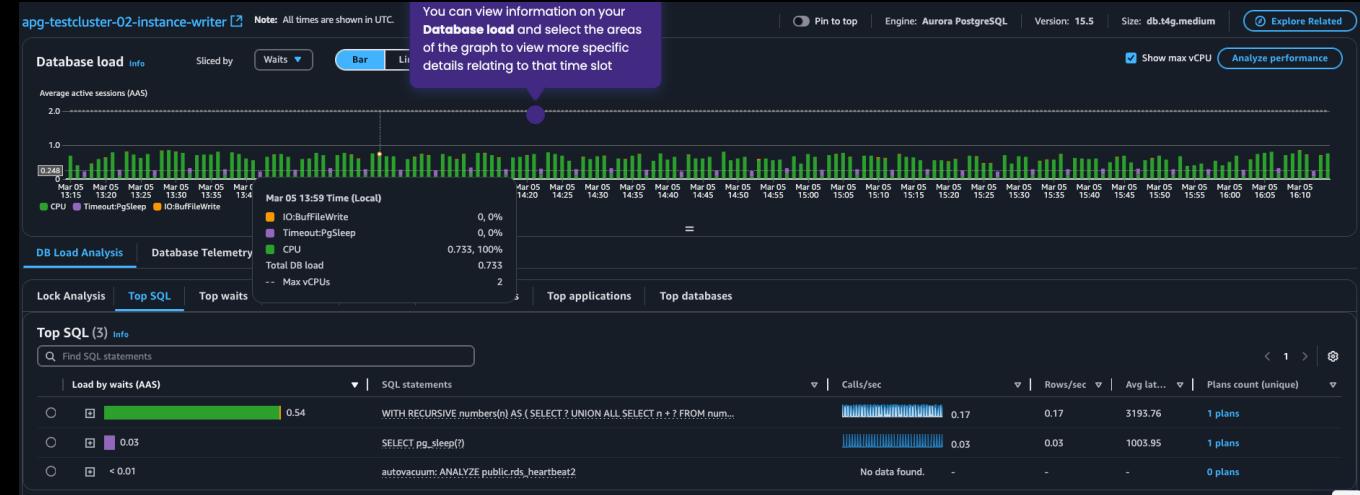
Instance View



# SQL の最適化例

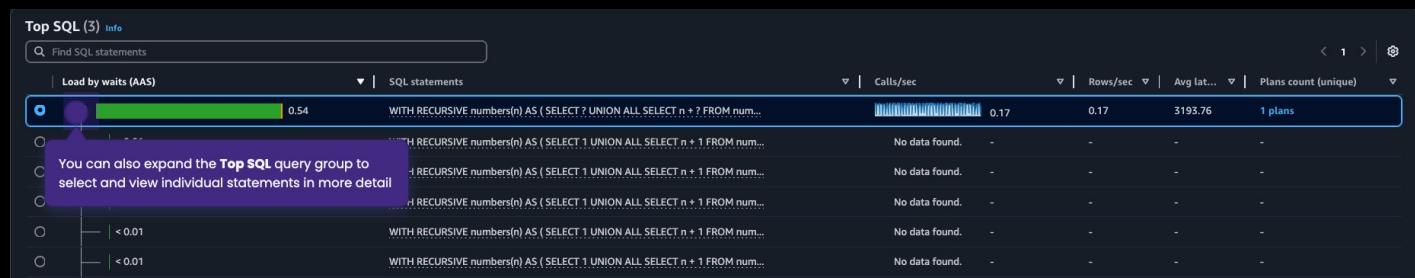
1

ボトルネックの  
トップ SQL を特定



2

待機イベントの分析



3

実行計画を分析し、SQL を  
改善



# アジェンダ

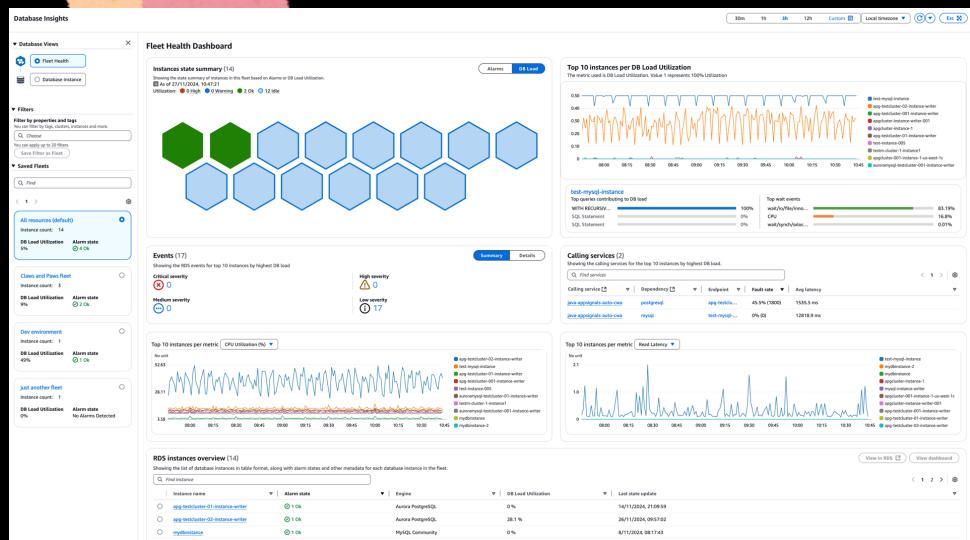
- 本日のゴール
- Amazon CloudWatch Lambda Insights
- Amazon CloudWatch Container Insights
- Amazon CloudWatch Database Insights
- まとめ

# コスト最適化から始めよう

- パフォーマンスをとるか、コストをとるか、判断材料に利用する

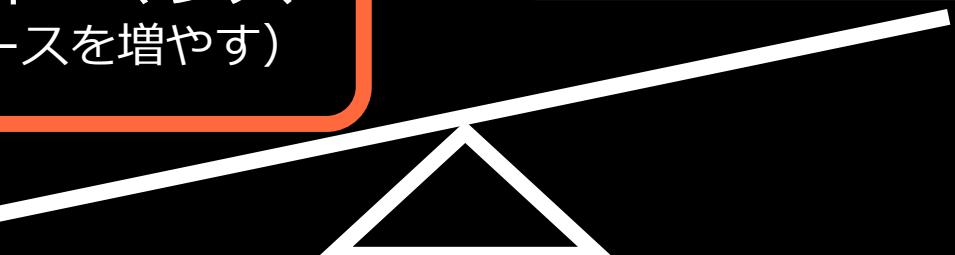


ワークロードが成長してきたので  
コスト最適化しよう！



パフォーマンス  
(リソースを増やす)

コスト  
(リソースを減らす)



# まとめ

- サーバレス、コンテナ、データベースの運用で  
「もっと細かく分析・最適化したい」方は  
Lambda Insights, Container Insights, Database Insights が便利です
- まずはワークロードが成長し、コスト最適化を行いたい際に導入して下さい

# Thank you!

