



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

# [AWS Black Belt Online Seminar]

## Amazon CloudWatch Container Insights で 始めるコンテナモニタリング入門

サービスカットシリーズ

Solutions Architect

水馬 拓也

2019/11/27

AWS 公式 Webinar

<https://amzn.to/JPWebinar>



過去資料

<https://amzn.to/JPArchive>

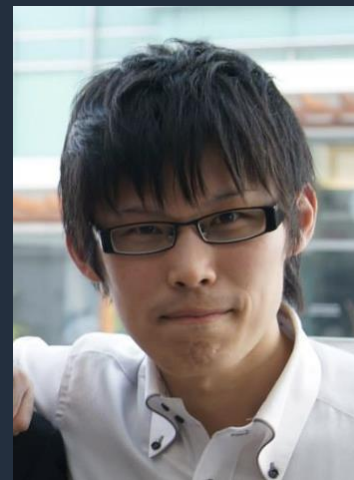


# 自己紹介

## 水馬 拓也 (みずま たくや)

- 所属

アマゾン ウェブ サービス ジャパン 株式会社  
技術統括本部 ソリューション アーキテクト



- 好きなサービス



AWS Fargate



AWS Lambda

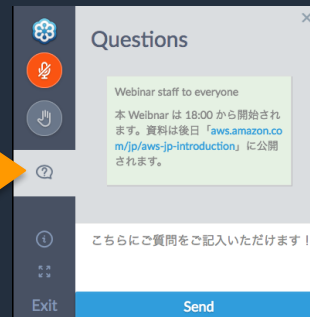
# AWS Black Belt Online Seminar とは

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

質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- いただいたQ&Aをピックアップしてblogにご紹介させていただく場合がございます
- 今後のロードマップに関するご質問は  
お答えできませんのでご了承下さい

- ① 吹き出しをクリック
- ② 質問を入力
- ③ Sendをクリック



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

# 内容についての注意点

- 本資料では2019年11月27日時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(<http://aws.amazon.com>)にてご確認ください。
- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。
- 価格は税抜表記となっております。日本居住者のお客様が東京リージョンを使用する場合、別途消費税をご請求させていただきます。
- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at <http://aws.amazon.com/agreement/>. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

# はじめに

- 想定聴講者:
  - 今後、Production環境でのコンテナ運用を検討されている方
  - コンテナワークロードにおけるモニタリング手法についてお悩みを持たれている方
- 本セッションで学べること:
  - コンテナワークロードのモニタリングにおいて必要な観点
  - Container Insightsを用いたコンテナワークロードのモニタリング方法、及び課題解決の手法

# 本日の内容

- **前提知識の整理**
  - モニタリングのスコープについて
  - コンテナオーケストレーション、Amazon ECSの主要要素
  - Container Insights登場前のコンテナモニタリング
- **Container Insightsの紹介**
  - Container Insightsの概要
  - 開始方法
  - メトリクスの表示方法
  - 収集されるパフォーマンスログの詳細
  - 具体的なユースケース
  - 料金について

# 本日の内容

- **前提知識の整理**

- モニタリングのスコープについて
- コンテナオーケストレーション、Amazon ECSの主要要素
- Container Insights登場前のコンテナモニタリング

- **Container Insightsの紹介**

- Container Insightsの概要
- 開始方法
- メトリクスの表示方法
- 収集されるパフォーマンスログの詳細
- 具体的なユースケース
- 料金について

# モニタリングのスコープについて



# 本セッションにおけるモニタリングのスコープ

- モニタリングといっても目的によって意味することが様々
- 本セッションではコンテナワークロードにおけるモニタリングに絞る

障害検知

性能劣化検知

コスト効率

不正アクセス検知

コンプライアンス  
準拠

監査対応

ユーザ体験の把握

..etc

# コンテナオーケストレーションについて

# コンテナオーケストレーションツールについて

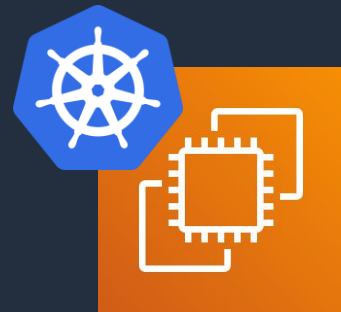
- 本セッションでは内容を簡潔にするために、コンテナオーケストレーションツールは Amazon ECSを前提にする



Amazon Elastic Container  
Service



Amazon Elastic  
Kubernetes Service



Kubernetes  
on  
Amazon EC2

# Amazon ECSとは？

## ■ コントロールプレーンとして提供



Amazon ECS

Linux & Windows



コンテナレベルの  
ネットワーク構成



高度なタスク配置  
戦略



他のAWSサービスと  
の連携



ECS CLI



グローバル展開



強力なスケジュールエンジン



オートスケーリング



CloudWatch 連携  
(ログ/メトリクス/イベント)



ロードバランサー

# ECSの構成要素

クラスタ

サービス

タスク



コンテナ



コンテナ

タスク



コンテナ



コンテナ



Elastic Load  
Balancing



タスク

- コンテナ(群)の実行単位
- タスクおよび各コンテナのCPUとメモリ上限を指定し、それを元にスケジュールされる

サービス

- 指定されたタスク数の維持
- ELBとの連携
- メトリクスに応じたオートスケール

クラスタ

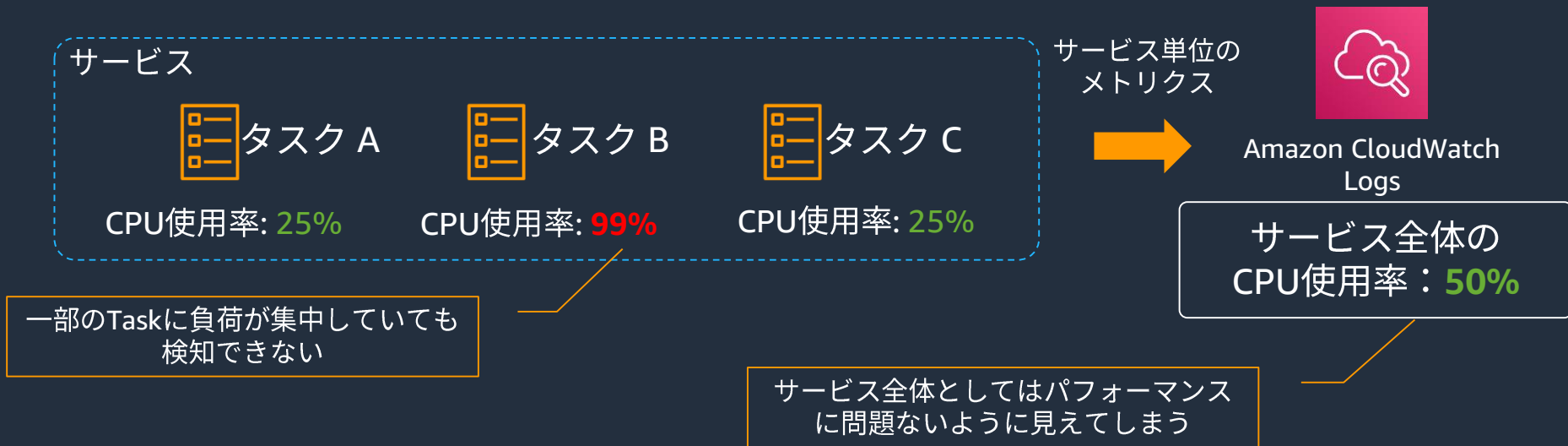
- コンテナ実行環境である論理的なグループ

# Container Insights登場前の コンテナモニタリング

# Container Insights登場前のコンテナモニタリングの例

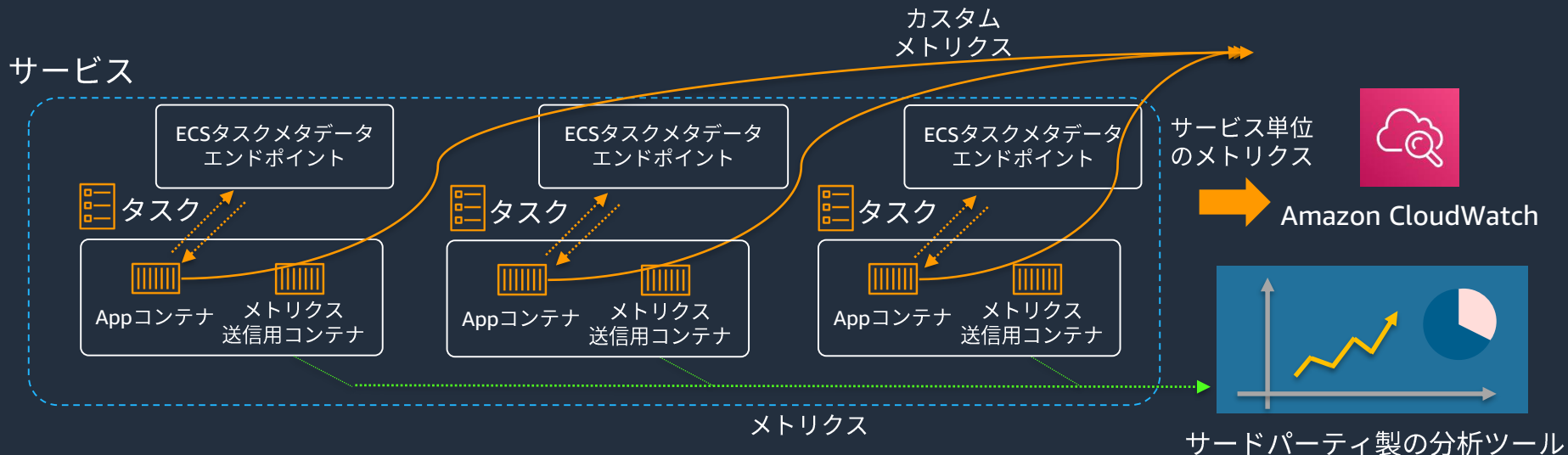
- CloudWatchのデフォルトの設定ではタスク、コンテナ単位のメトリクスが取得できなかった
- タスク、コンテナ単位レベルの問題を検知する際に、より詳細なメトリクス情報が必要となる場合があった

タスク、コンテナ単位レベルの問題の検知が難しい例)



# Container Insights登場前のコンテナモニタリングの例

- 各コンテナからECSタスクメタデータエンドポイントを呼び出し、カスタムメトリクスとしてCloudWatchにメトリクスを送信する
- サードパーティ製のメトリクス送信用のコンテナをサイドカーとして配置する



タスクやコンテナレベルのメトリクスを取得するには何かしらの工夫が必要だった



# 本日の内容

- **前提知識の整理**
  - モニタリングのスコープについて
  - コンテナオーケストレーション、Amazon ECSの主要要素
  - Container Insights登場前のコンテナモニタリング
- **Container Insightsの紹介**
  - Container Insightsの概要
  - 開始方法
  - メトリクスの表示方法
  - 収集されるパフォーマンスログの詳細
  - 具体的なユースケース
  - 料金について

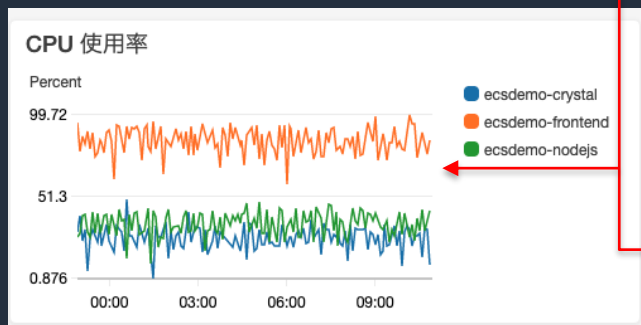
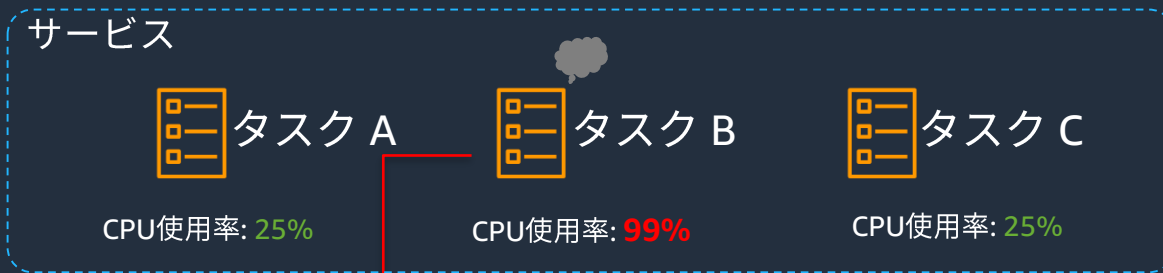
# Container Insightsの概要

# Amazon CloudWatch Container Insightsとは

- コンテナ化されたアプリケーションのメトリクスとログを収集、集計、要約できるCloudWatchの機能の一つ
- CloudWatchにてタスク、コンテナレベルでのモニタリングが可能
- Container Insights が収集するメトリクスは自動的に作成されるダッシュボードに集約され、より鋭い洞察を行うことが可能
- AWSが提供するコンテナオーケストレーションツールであるAmazon ECSや、Amazon EKS、および Amazon EC2 の Kubernetes プラットフォームでご利用可能
  - ※ 2019/11/27 現在、AWS Batch で Container Insights はサポートされていません。

# Amazon CloudWatch Container Insightsの概要

- Amazon CloudWatchと統合された、タスクやコンテナレベルでメトリクスやログを取得することが可能



Task performance (9)

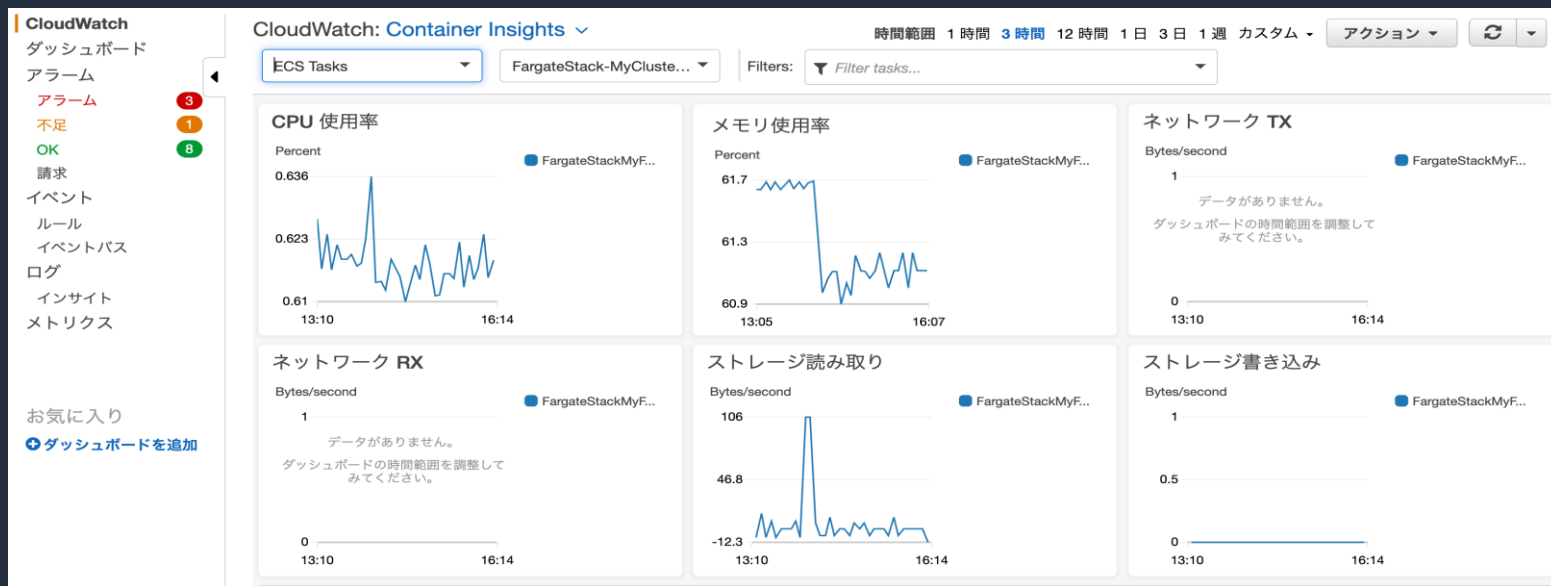
Q ecsdemo-frontend X

< 1

<input type="checkbox"/>	Task definition	サービス	平均 CPU (%)	平均メモリ (%)
<input type="checkbox"/>	ecsdemo-frontend	ecsdemo-frontend	1.6258	16.0156
<input type="checkbox"/>	ecsdemo-frontend	ecsdemo-frontend	99.574	96.0156
<input type="checkbox"/>	ecsdemo-frontend	ecsdemo-frontend	1.5725	16.0156

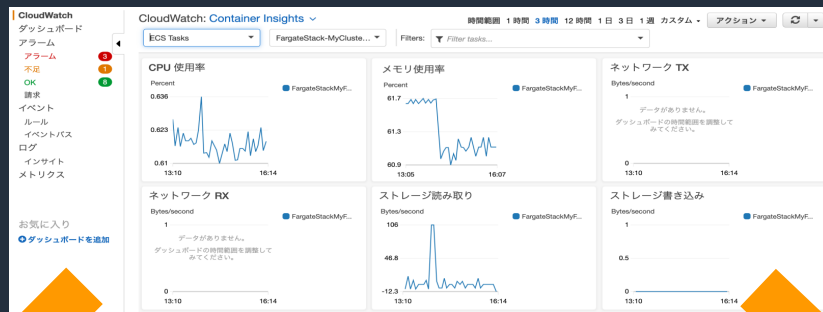
# Amazon CloudWatch Container Insightsの概要

- Container Insights が収集するメトリクスは自動的にダッシュボードに集約され、可視化を行うことが可能



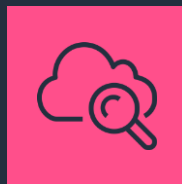
# Amazon CloudWatch Container Insightsの概要

- CloudWatch Logs InsightsやX-Rayとも統合されている
- Container Insightsのダッシュボードを起点により詳細な分析が可能



## CloudWatch Logs Insightsの要件の例

- グラフのより詳細な値を見たい
- ログに対し分析クエリを発行したい

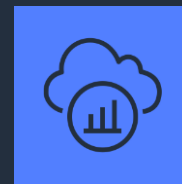


Amazon CloudWatch  
Logs Insights

## Container Insights

## X-Ray を使う要件の例

- タスク間の通信をトレースしたい



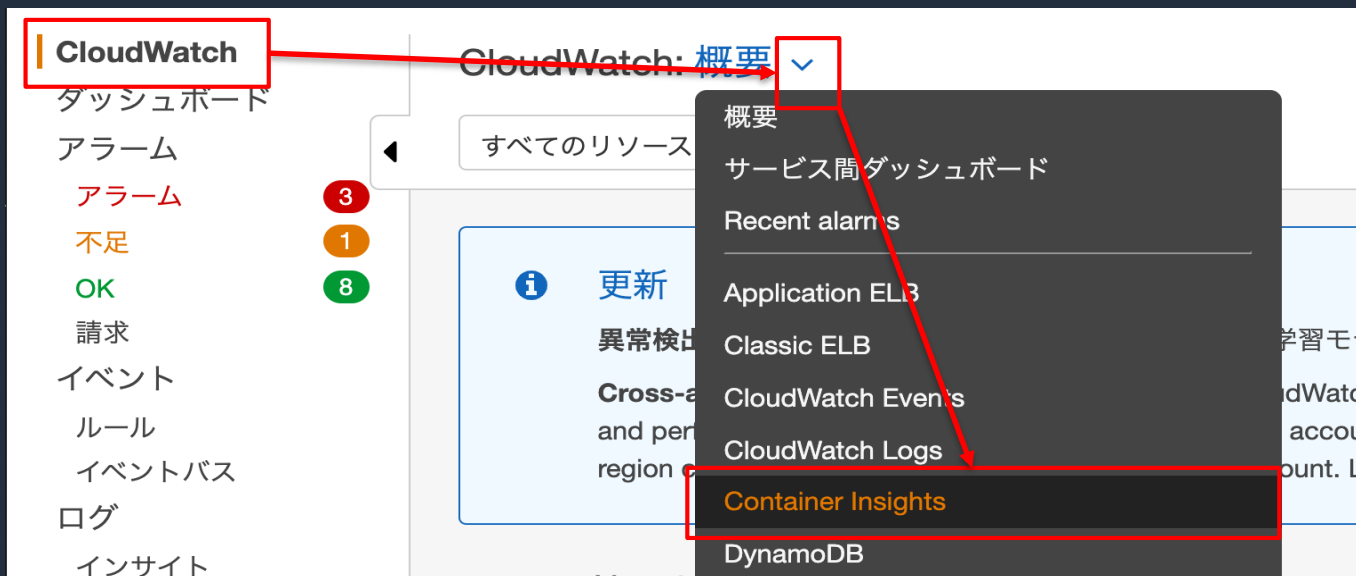
AWS X-Ray

# Container Insightsの使用方法



# Container Insightsのメトリクスの確認方法 (1)

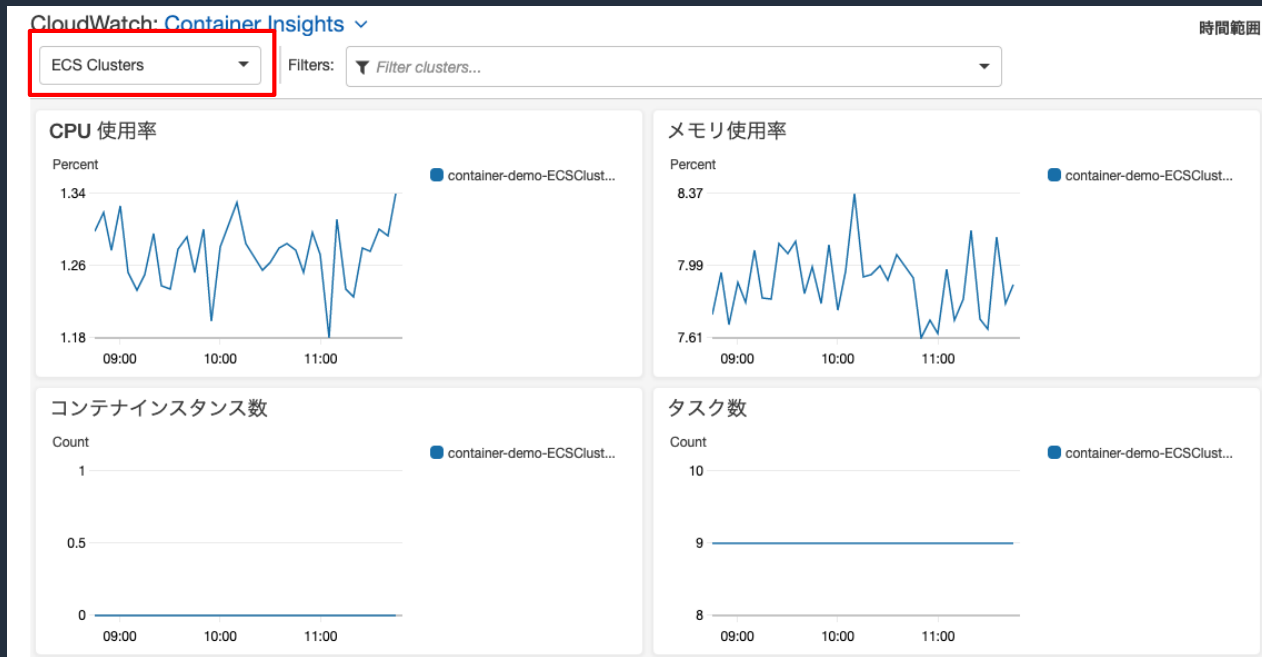
- CloudWatchのトップ画面を開き「概要」横のプルダウンからの「Container Insights」を選択
- 対象リージョンにContainer Insightsが有効なコンテナが起動していない場合はこの項目が表示されないので注意





# Container Insightsのメトリクスの確認方法 (2)

- メトリクスを表示する取得単位を選択する



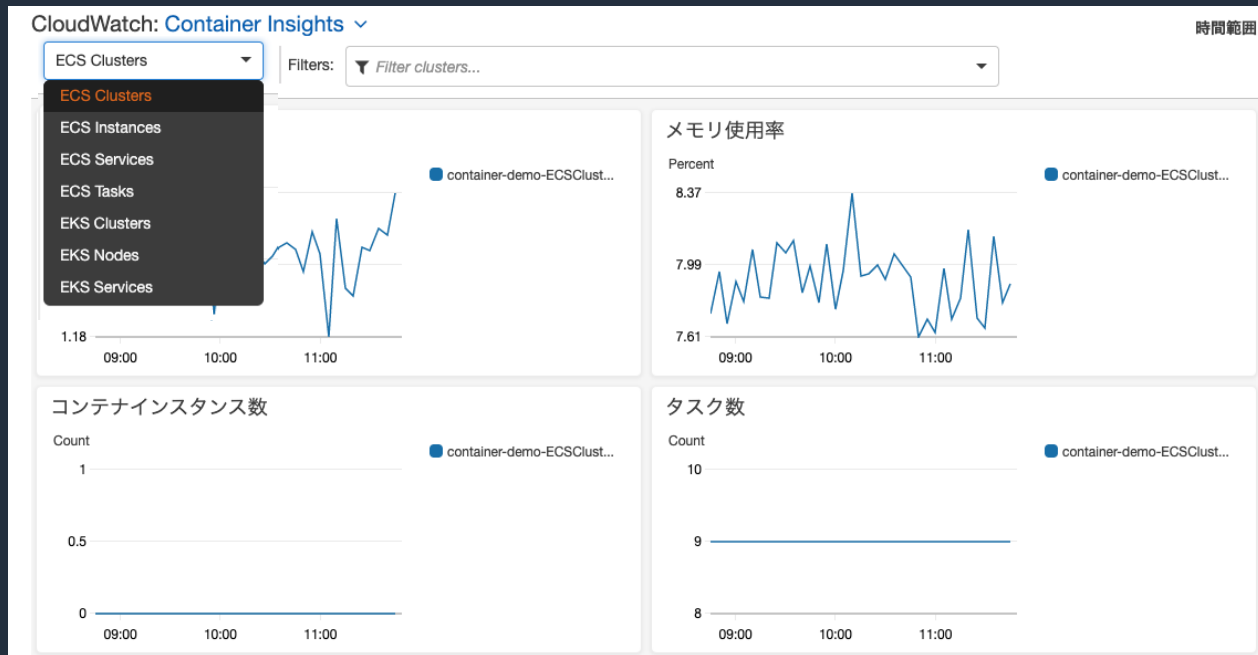
ECSにおける選択可能な単位

- ECS Clusters
- ECS Instances \*1
- ECS Services
- ECS Tasks

\*1 Fargate 起動タイプでタスクを起動している場合、ECS Instances メトリクスは取得されません

# Container Insightsのメトリクスの確認方法 (2)

- メトリクスを表示する取得単位を選択する



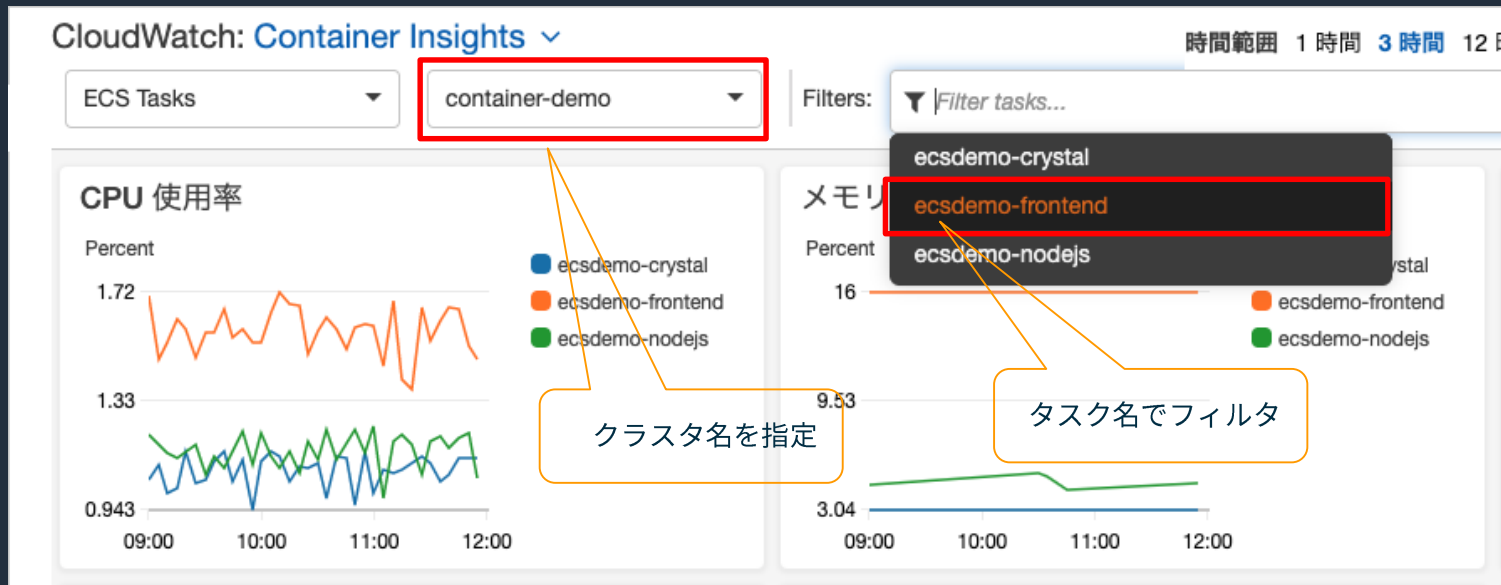
ECSにおける選択可能な単位

- ECS Clusters
- ECS Instances \*1
- ECS Services
- ECS Tasks

\*1 Fargate 起動タイプでタスクを起動している場合、ECS Instances メトリクスは取得されません

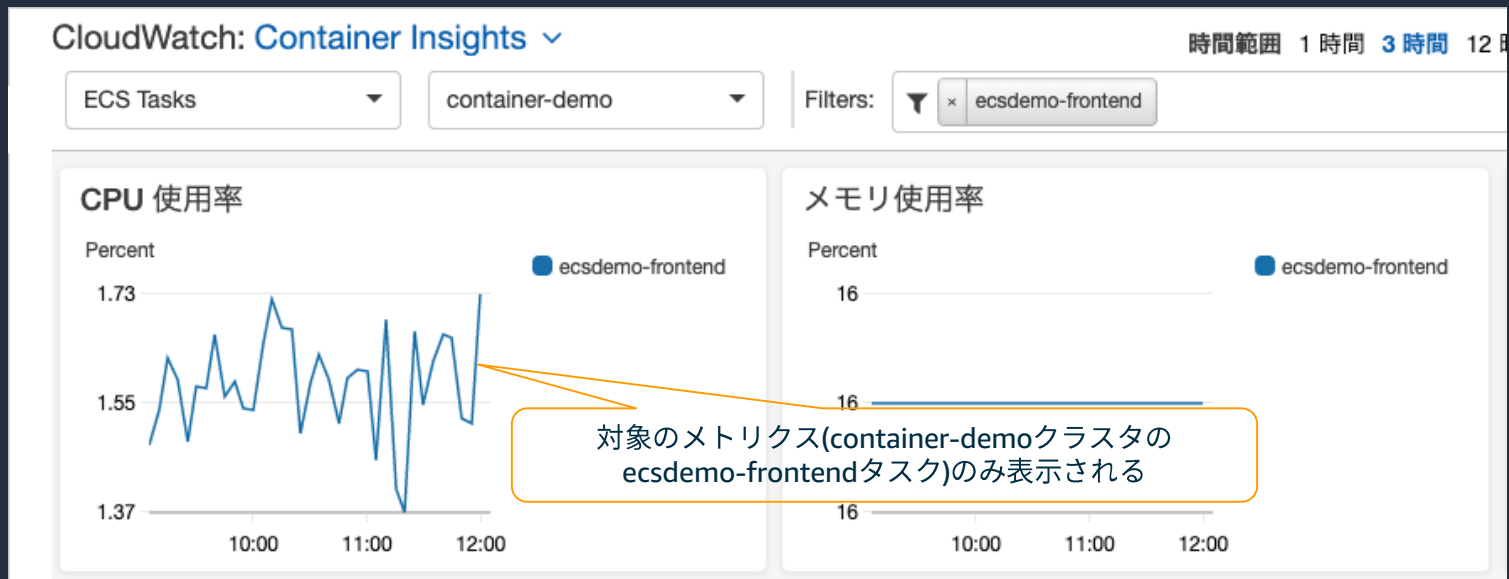
# Container Insightsのメトリクスの確認方法 (3)

- 特定のメトリクスのみ表示させるようフィルタをかける事が可能



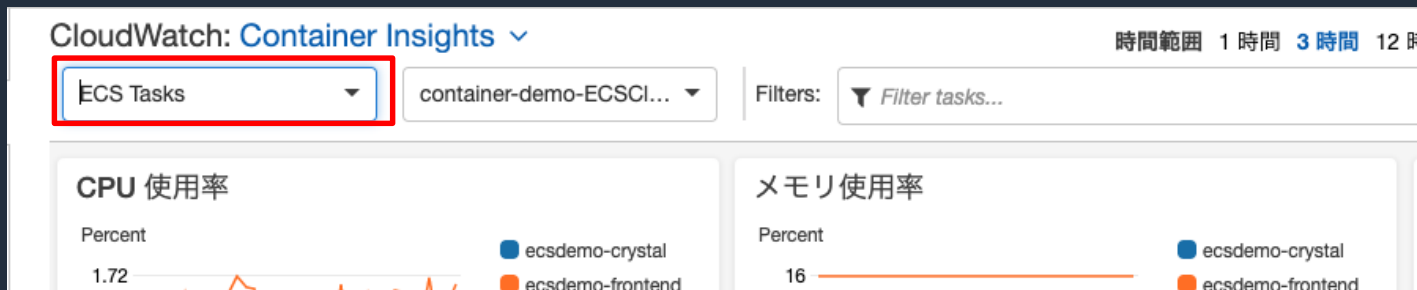
# Container Insightsのメトリクスの確認方法 (3)

- 特定のメトリクスのみ表示させるようフィルタをかける事が可能



# Container Insightsのメトリクスの確認方法 (4)

- コンテナ単位のメトリクスを取得するには「ECS Task」を指定する



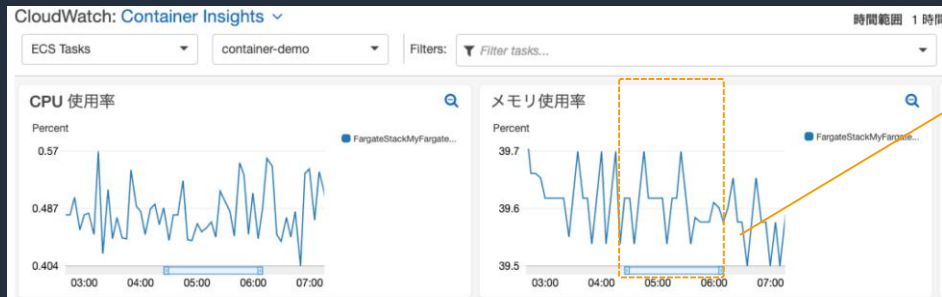
画面下部に移動

The screenshot shows the 'Container performance (9)' section of the CloudWatch Container Insights interface. It features a search bar with the placeholder text 'Filter task definitions, services, tasks...'. Below the search bar is a table with the following columns: 'Task definition', 'サービス', '平均 CPU (%)', and '平均メモリ (%)'. The table contains three rows of data for 'ecsdemo-frontend' tasks. The '平均 CPU (%)' and '平均メモリ (%)' columns are highlighted with a red box.

Task definition	サービス	平均 CPU (%)	平均メモリ (%)
ecsdemo-frontend	ecsdemo-frontend	1.6092	16.0156
ecsdemo-frontend	ecsdemo-frontend	1.5862	16.0156
ecsdemo-frontend	ecsdemo-frontend	1.5504	16.0156

# タスク単位のドリルダウンを行う

- Task performanceからCloudWatch Logs Insights、X-Rayの画面へ遷移
- タスクやコンテナを選択し「アクション」からより詳細な分析が可能



Task performance (14)

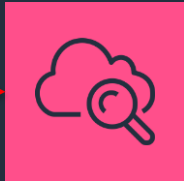
ecsdemo-nodejs

Task definition	サービス	平均 CPU (%)	平均メモリ (%)
<input type="checkbox"/>	ecsdemo-nodejs	6.4649	4.8314
<input checked="" type="checkbox"/>	ecsdemo-nodejs	95.1297	94.9085
<input type="checkbox"/>	ecsdemo-nodejs	5.0975	4.4271
<input type="checkbox"/>	ecsdemo-nodejs	4.5607	3.8574
<input type="checkbox"/>	ecsdemo-nodejs	4.5090	5.0148

アクション

- アプリケーションログの表示
- AWS X-Ray トレースの表示
- パフォーマンスログの表示

取得するログの期間を指定可能



- グラフのより詳細な値を確認
- ログに対し分析クエリを発行
- アプリケーションログの確認

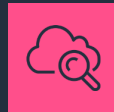
Amazon CloudWatch  
Logs Insights



- タスク間の通信をトレース

AWS X-Ray

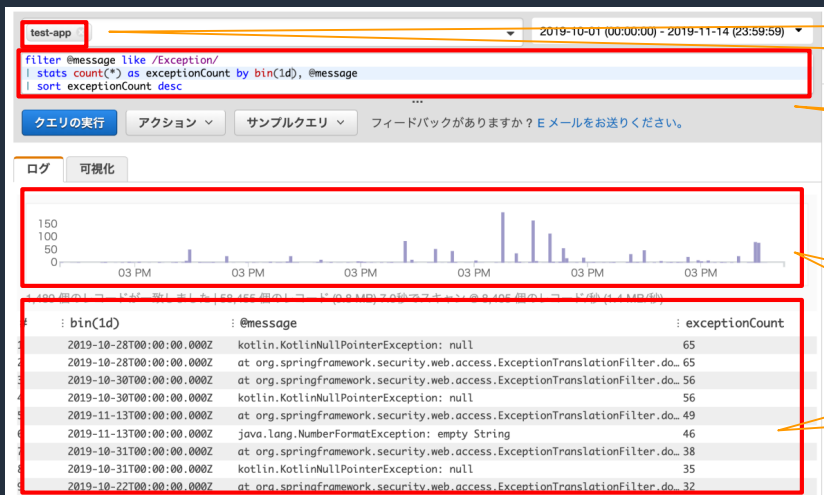
# アプリケーションログの表示



Amazon CloudWatch  
Logs Insights

- Cloudwatch Logs Insightsの画面に遷移し、対象タスクが出力するアプリケーションログやエラーログに対し抽出条件を指定したクエリを発行する事ができる
- 期間を絞った検索や、統計情報の取得も可能

例)アプリケーションログから「Exception」という文字を含むログを集計し、時系列で件数を可視化



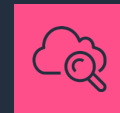
対象のタスク、コンテナが指定された状態

クエリの発行

ヒットしたログの数を時系列で表示

ヒットしたログの表示

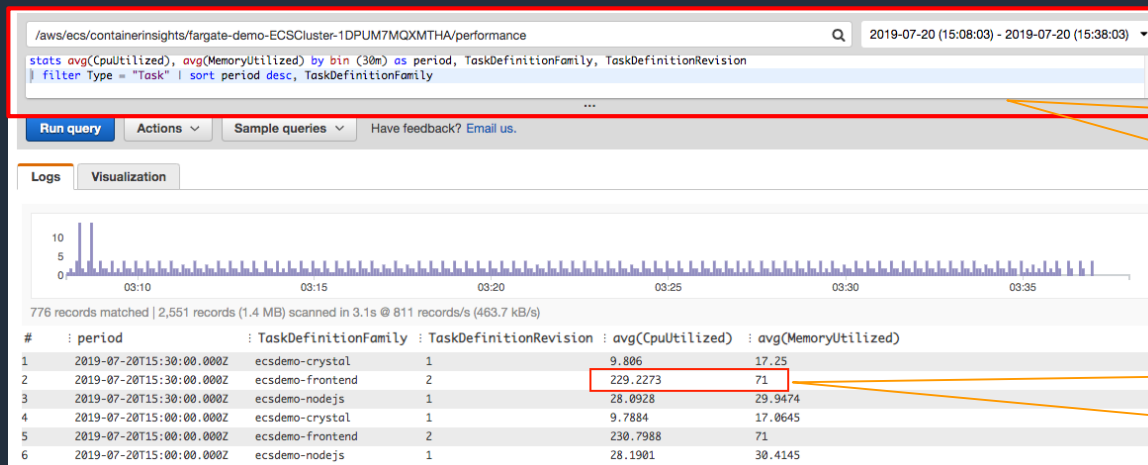
# パフォーマンスログの表示



Amazon CloudWatch  
Logs Insights

- CloudWatch Logs Insightsの画面に遷移し対象タスクが出力するパフォーマンスログに対し抽出条件を指定したクエリを発行することができる
- 自動ダッシュボード画面で表示されていたメトリクスをさらにドリルダウンして分析が可能

例) パフォーマンスが悪化しているタスクのボトルネックを調査

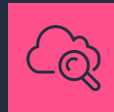


調査クエリの発行

パフォーマンスのボトルネックを調査



# クエリ構築のためのヘルプ機能



Amazon CloudWatch  
Logs Insights

- 一般的なクエリを「サンプルクエリ」という雛形として提供
- 画面右の「クエリのヘルプ」からコマンド構築していくことも可能

The screenshot shows the Amazon CloudWatch Logs Insights console. The main query editor contains the following query:

```
filter @message like /Exception/  
| stats count(*) as exceptionCount by bin(5m)  
| sort exceptionCount desc
```

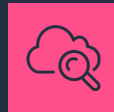
Below the query editor, the 'クエリの実行' (Execute Query) button is visible. To its right, the 'アクション' (Action) dropdown menu is open, and the 'サンプルクエリ' (Sample Query) option is selected. A dropdown menu is displayed below 'サンプルクエリ', listing various query types:

- Lambda のクエリ
- VPC フローログのクエリ
- CloudTrail のクエリ
- 一般的なクエリ
- Route 53 クエリ
- AWS AppSync クエリ

The '一般的なクエリ' (General Query) option is highlighted with a red box. A tooltip is shown for this option, containing the text: '最近追加された 25 件のログイベント' (25 recently added log events) and '5 分ごとにログに記録される例外の数' (Number of exceptions recorded in the log every 5 minutes). The second line of the tooltip is also highlighted with a red box.

On the right side of the console, the 'クエリのヘルプ' (Query Help) section is visible, with a link to '詳細はこちら' (Details here). Below this, the 'コマンド' (Commands) section lists various query commands: fields, filter, stats, sort, limit, parse. The '検出されたフィールド' (Detected Fields) section shows a search bar and a list of fields with their usage percentages: @ingestionTime (100%), @logStream (100%), @message (100%), @timestamp (100%), and CloudWatchMetrics.0.Dime... (100%).

# クエリ構築のためのヘルプ機能



Amazon CloudWatch  
Logs Insights

- 一般的なクエリを「サンプルクエリ」という雛形として提供
- 画面右の「クエリのヘルプ」からコマンド構築していくことも可能

The screenshot shows the Amazon CloudWatch Logs Insights interface. At the top, there's a search bar with the path `/aws/ecs/containerinsights/FargateStac...` and a date range of `2019-11-26 (11:11:23) - 2019-11-26 (14:11:23)`. The main query area contains the following query:

```
filter @message like /Exception/  
| stats count(*) as exceptionCount by bin(5m)  
| sort exceptionCount desc
```

Below the query, there are buttons for `クエリの実行` (Execute Query), `アクション` (Action), and `サンプルクエリ` (Sample Query). The `サンプルクエリ` dropdown menu is open, showing a list of query templates:

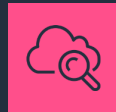
- Lambda のクエリ
- VPC フローログのクエリ
- CloudTrail のクエリ
- 一般的なクエリ
- Route 53 クエリ
- AWS AppSync クエリ

Two callout boxes provide additional information:

- The first callout box explains that the query is based on one or more conditions and lists supported operators: `=`, `!=`, `>`, `>=`, `<`, `<=`, `and`, `or`, and `not`. It also shows a sample query: `filter @message like /(?!)(Exception|error|fail|5\d\d)/`.
- The second callout box highlights the `一般的なクエリ` (General Query) option, which is described as `最近追加された 25 件のログイベント` (25 most recently added log events) and `5 分ごとにログに記録される例外の数` (Number of exceptions recorded in the log every 5 minutes).

On the right side, the `クエリのヘルプ` (Query Help) section is visible, with a `filter 追加` (Add filter) button highlighted in red. Below it, a list of fields is shown, including `@ingestionTime`, `@logStream`, `@message`, `@timestamp`, and `CloudWatchMetrics.0.Dime...`.

# 分析のためのサンプルクエリ for EKS



Amazon CloudWatch  
Logs Insights

## コンテナ名ごとの CPU 使用率(パフォーマンスログ)

```
stats pct(container_cpu_usage_total, 50) as CPUPercMedian by kubernetes.container_name  
| filter Type="Container"
```

## ノードの平均CPU使用率(パフォーマンスログ)

```
STATS avg(node_cpu_utilization) as avg_node_cpu_utilization by NodeName | SORT  
avg_node_cpu_utilization DESC
```

## クラスタノードの障害数(パフォーマンスログ)

```
stats avg(cluster_failed_node_count) as CountOfNodeFailures | filter Type="Cluster" | sort @timestamp  
desc
```

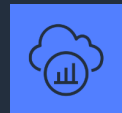
## アプリケーションログエラー (アプリケーションログ)

```
Count by container name: stats count() as countoferrors by kubernetes.container_name | filter  
stream="stderr" | sort countoferrors desc
```

### ■ Container Insights メトリクスの表示

[https://docs.aws.amazon.com/ja\\_jp/AmazonCloudWatch/latest/monitoring/Container-Insights-view-metrics.html](https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/Container-Insights-view-metrics.html)

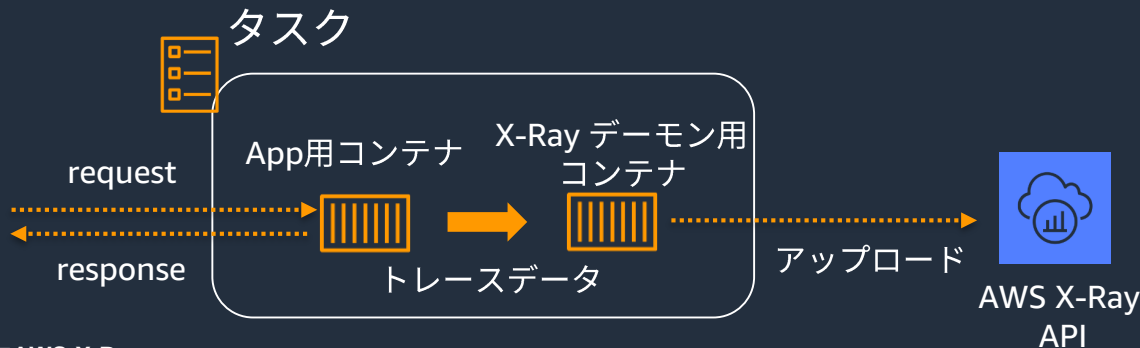
# AWS X-Rayトレースの表示



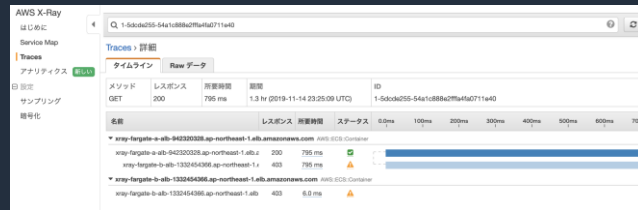
AWS X-Ray

- リクエスト単位の情報にトレース可能
  - リクエストのエラー率
  - レスポンスタイムの統計情報 ..etc
- Container InsightsでX-Rayのトレース情報の取得するにはアプリケーション用コンテナのサイドカーとしてX-Rayデーモンを実行するコンテナを配置する

例) X-Rayデーモン用コンテナをサイドカーとして配置する



## AWS X-Rayコンソールで可視化



AWS X-Ray

[https://docs.aws.amazon.com/ja\\_jp/xray/latest/devguide/aws-xray.html](https://docs.aws.amazon.com/ja_jp/xray/latest/devguide/aws-xray.html)

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# Container Insightsの開始方法

# Container Insightsを有効にする ～ ECS編 ～

新規のクラスタに対し有効化する

- マネジメントコンソールの「ステップ 2: クラスターの設定」画面にて「Container Insights を有効にする」にチェックを入れる



- AWS CloudFormationやAWS Cloud Development Kit(CDK)から作成する場合
  - アカウント単位、IAMロール単位、IAMユーザ単位のいずれかでContainer Insightsがオプトインされている必要がある

# Container Insightsを有効にする ～ ECS編 ～

## ・ Container Insightsのオプトインについて

### ■ Container Insightsは以下の単位でデフォルトで有効化することが可能

- ・ アカウント全体
- ・ IAMロール
- ・ IAMユーザ

Amazon ECS  
クラスター  
タスク定義  
**Account Settings**  
Amazon EKS  
Clusters  
Amazon ECR

### アカウント設定

root ユーザーとしてログインします。アカウントのデフォルト設定の変更、あるいはアカウントで特定の IAM ユーザーまたは IAM ロールを変更できます。

設定の適用範囲 特定の IAM ユーザー用の設定

IAM ユーザー

- ルート用設定 (アカウントのデフォルト設定を上書き)
- 特定の IAM ユーザー用の設定
- 特定の IAM ロール用の設定

画面下部へ移動

リソース	アカウントデフォルト設定を上書き
Container Insights	<input checked="" type="checkbox"/>

適用範囲でContainer Insightsがデフォルトで有効になる

# Container Insightsを有効にする ～ ECS編 ～

- Container Insightsのオプトインについて
  - Container InsightsのオプトインはAWS CLIからも実行可能

アカウント単位でのオプトイン

```
aws ecs put-account-setting-default --name containerInsights --value enabled --region us-east-1
```

ユーザ単位でのオプトイン

```
aws ecs put-account-setting --name containerInsights --value enabled --principal-arn arn:aws:iam::aws_account_id:user/userName --region us-east-1
```

ロール単位でのオプトイン

```
aws ecs put-account-setting --name containerInsights --value enabled --principal-arn arn:aws:iam::aws_account_id:role/roleName --region us-east-1
```



# Container Insightsを有効にする ～ ECS編 ～

既存のクラスタに対し有効化する

- Container Insights は、既存の Amazon ECS クラスタ及び、新規に作成したクラスターで有効にできる
- 既存のクラスタでContainer Insightsを有効にするには、AWS CLIから以下のコマンドを発行

```
$ aws ecs update-cluster-settings --cluster <myCluster> --settings  
name=containerInsights,value=enabled
```

※ AWS CLIのバージョンが1.16.200以降が必要

📖 Amazon ECS での Container Insights のセットアップ

[https://docs.aws.amazon.com/ja\\_jp/AmazonCloudWatch/latest/monitoring/deploy-container-insights-ECS.html](https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/deploy-container-insights-ECS.html)

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# Container Insightsを有効にする ～ ECS編 ～

- Container Insightsが有効かどうかは、ECSのクラスター画面で「Container Insights」に緑色のチェックが入っているかどうかで確認できる

The screenshot shows the AWS ECS console interface for a cluster named 'container-demo'. At the top, there is a 'CloudWatch モニタリング' (CloudWatch Monitoring) section with a red box highlighting a green checkmark and the text 'Container Insights'. Below this, the cluster is divided into two sections: 'FARGATE' and 'EC2'. The 'FARGATE' section displays three metrics: '3 サービス' (3 Services), '9 実行中のタスク' (9 Running Tasks), and '0 保留中のタスク' (0 Pending Tasks). The 'EC2' section displays three metrics: '0 サービス' (0 Services), '0 実行中のタスク' (0 Running Tasks), and '0 保留中のタスク' (0 Pending Tasks). Additionally, there are two grey boxes labeled 'No data' for 'CPUUtilization' and 'MemoryUtilization', and a final metric '0 コンテナインスタンス' (0 Container Instances).

Category	Service	Running Tasks	Pending Tasks	CPU Utilization	Memory Utilization	Container Instances
FARGATE	3	9	0			
EC2	0	0	0	No data	No data	0

# Container Insightsを有効にする ～ EKS、Kubernetes on EC2編 ～

Container Insightsを有効にするための前提条件を確認

- EKS / Kubernetes on EC2 で必要な条件
  - Container Insightsが利用可能なリージョンでクラスタが起動している
  - kubectl がインストール、権限設定が完了しており、kubectl apply が実行可能な状態になっている
- Kubernetes on EC2 で必要な条件
  - Kubernetes クラスターで、ロールベースのアクセス制御 (RBAC) が有効になっている
  - kubelet で Webhook 認証モードが有効になっている
  - コンテナランタイムがDockerになっている

📖 Amazon EKS と Kubernetes で Container Insights をセットアップする 前提条件の確認

[https://docs.aws.amazon.com/ja\\_jp/AmazonCloudWatch/latest/monitoring/Container-Insights-prerequisites.html](https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/Container-Insights-prerequisites.html)

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# Container Insightsを有効にする ～ EKS、Kubernetes on EC2編 ～

## Container Insightsを有効にする手順

- 必須手順
  - CloudWatch にメトリクスを送信するCloudWatchエージェントをDaemonSetとしてセットアップ
  - CloudWatch Logsにログを送信するDaemonSetとしてFluentDをセットアップ
- オプション手順
  - Amazon EKS コントロールプレーンのログ記録を有効する(コントロールプレーンのログをCloudWatch Logsで監査する必要がある場合)
  - StatsD エンドポイントとして CloudWatch エージェントをセットアップする(StatsDを使用している場合)

📖 Amazon EKS と Kubernetes で Container Insights をセットアップする

[https://docs.aws.amazon.com/ja\\_jp/AmazonCloudWatch/latest/monitoring/deploy-container-insights-EKS.html](https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/deploy-container-insights-EKS.html)

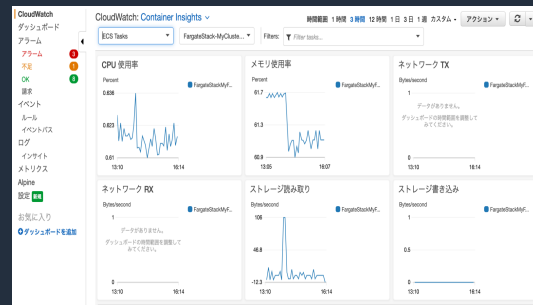
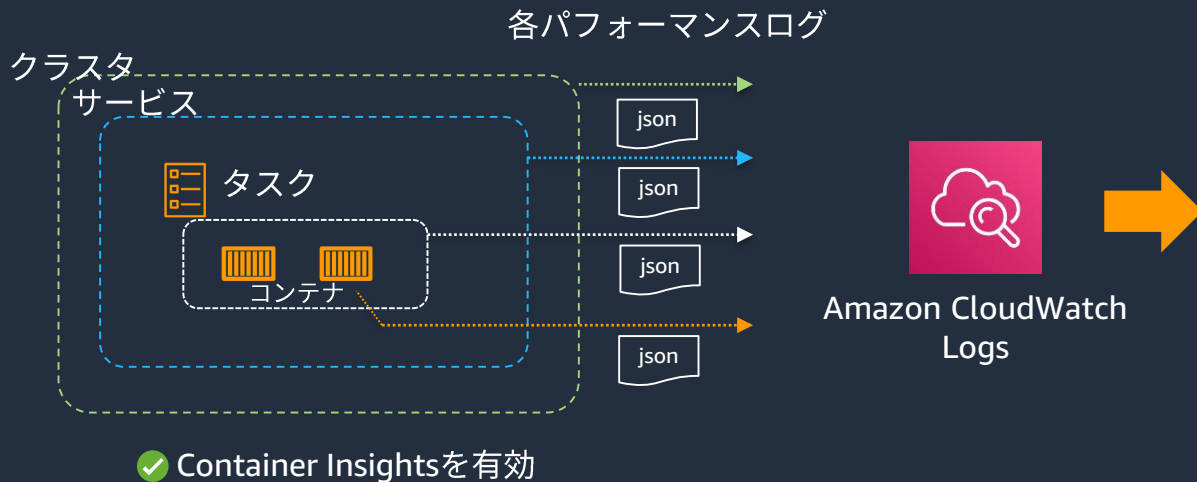
© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# 収集されるパフォーマンスログの詳細

# パフォーマンスログとは？

- Container Insightsの機能を有効にするとカスタムメトリクスとしてパフォーマンスログがタスク単位、サービス単位、コンテナ単位でCloudWatch Logsに出力される
- パフォーマンスログをもとに自動ダッシュボード上で可視化が行われる
- 通常のカスタムメトリクス同様、ロググループやCloudWatch Logs Insightsからクエリで分析することも可能



# パフォーマンスログの内容

- パフォーマンスログの各項目のデータ型をドキュメントで公開

```
{
  "Version": "0",
  "Type": "Container",
  "ContainerName": "mysql",
  "TaskId": "7c7bce41-8f2f-4673-b1a3-0e57ef51168f",
  "TaskDefinitionFamily": "hello_world",
  "TaskDefinitionRevision": "2",
  "ContainerInstanceId": "12345678-db61-4b1b-b0ec-93ad21467cc9",
  "EC2InstanceId": "i-1234567890123456",
  "ServiceName": "myCIService",
  "ClusterName": "myCICluster",
  "Timestamp": 1561586749104,
  "CpuUtilized": 1.8381139895790504,
  "CpuReserved": 30.0,
  "MemoryUtilized": 295,
  "MemoryReserved": 400,
  "StorageReadBytes": 10817536,
  "StorageWriteBytes": 2085376000,
  "NetworkRxBytes": 6522,
  "NetworkRxDropped": 0,
  "NetworkRxErrors": 0,
  "NetworkRxPackets": 83,
  "NetworkTxBytes": 3904,
  "NetworkTxDropped": 0,
  "NetworkTxErrors": 0,
  "NetworkTxPackets": 44
}
```

## コンテナのデータ型

コンテナが存在する  
タスク ID

コンテナが配置されている  
サービスの名前

使用されている  
CPUユニット数

使用されているメモリ(MB)

$\text{MemoryUtilized} / \text{MemoryReserved} * 100$   
 $= 295 / 400$   
 $\approx 75\%$  [メモリ使用率]

■ Amazon ECS の Container Insights パフォーマンスログイベント

[https://docs.aws.amazon.com/ja\\_jp/AmazonCloudWatch/latest/monitoring/Container-Insights-reference-performance-logs-ECS.html](https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/Container-Insights-reference-performance-logs-ECS.html)

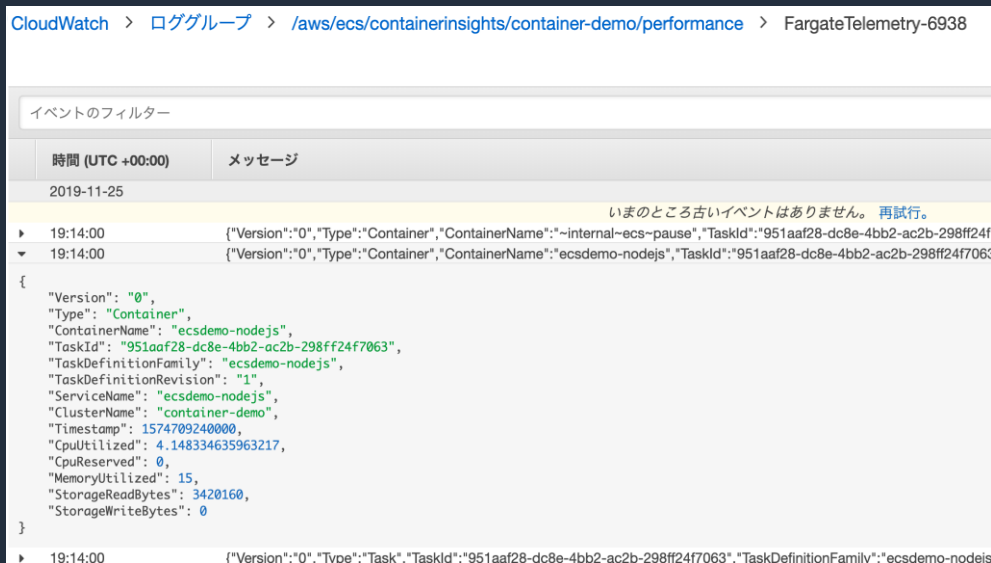
■ Amazon EKS の Container Insights パフォーマンスログイベント

[https://docs.aws.amazon.com/ja\\_jp/AmazonCloudWatch/latest/monitoring/Container-Insights-reference-performance-logs-EKS.html](https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/Container-Insights-reference-performance-logs-EKS.html)

# パフォーマンスログの確認方法

- パフォーマンスログはCloudWatch Logsのロググループからも確認可能
  - ECSの場合：`/aws/ecs/containerinsights/<クラスタ名>/performance`
  - EKSの場合：`/aws/containerinsights/<クラスタ名>/performance`

例) container-demo clusterに配置されているcontainer のパフォーマンスログ



The screenshot shows the AWS CloudWatch Logs console interface. The breadcrumb navigation at the top reads: CloudWatch > ロググループ > /aws/ecs/containerinsights/container-demo/performance > FargateTelemetry-6938. Below the navigation is a search bar labeled 'イベントのフィルター'. The main content area displays a table of log events. The first event is a message: 'いまのところ古いイベントはありません。再試行。'. The second event is expanded, showing a JSON log entry for a container named 'ecsdemo-nodejs' at 19:14:00. The JSON entry includes fields for Version, Type, ContainerName, TaskId, TaskDefinitionFamily, TaskDefinitionRevision, ServiceName, ClusterName, Timestamp, CpuUtilized, CpuReserved, MemoryUtilized, StorageReadBytes, and StorageWriteBytes.

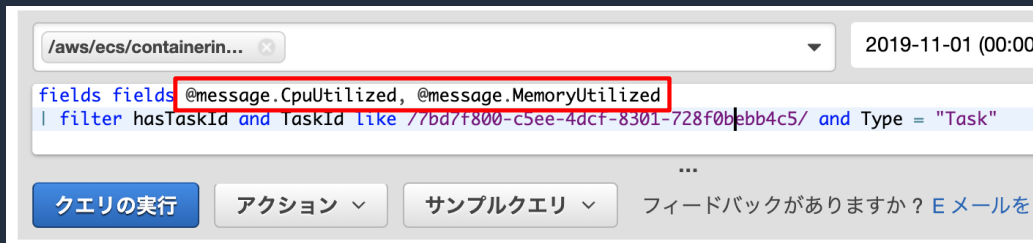
時間 (UTC +00:00)	メッセージ
2019-11-25	いまのところ古いイベントはありません。再試行。
▶ 19:14:00	{"Version":"0","Type":"Container","ContainerName":"~internal-ecs-pause","TaskId":"951aaf28-dc8e-4bb2-ac2b-298ff24f7063"}
▼ 19:14:00	<pre>{   "Version": "0",   "Type": "Container",   "ContainerName": "ecsdemo-nodejs",   "TaskId": "951aaf28-dc8e-4bb2-ac2b-298ff24f7063",   "TaskDefinitionFamily": "ecsdemo-nodejs",   "TaskDefinitionRevision": "1",   "ServiceName": "ecsdemo-nodejs",   "ClusterName": "container-demo",   "Timestamp": 1574709240000,   "CpuUtilized": 4.148334635963217,   "CpuReserved": 0,   "MemoryUtilized": 15,   "StorageReadBytes": 3420160,   "StorageWriteBytes": 0 }</pre>
▶ 19:14:00	{"Version":"0","Type":"Task","TaskId":"951aaf28-dc8e-4bb2-ac2b-298ff24f7063","TaskDefinitionFamily":"ecsdemo-nodejs"}



# CloudWatch Logs Insightsからのパフォーマンスログの確認方法

- パフォーマンスログ内容は@messageフィールドに保持される
- CpuUtilized項目を取得したい場合は@message.CpuUtilizedで取得が可能

例) タスクのCpuUtilizedとMemoryUtilized項目を取得する



# 具体的なユースケース



Big Data

# ユースケース1.

## ECSでタスクに配置するコンテナのリソースを適切なサイズにチューニングしたい



# ユースケースの詳細

- コンテナごとの適切なリソースの配分は非常に重要
- 適切でないリソース配分はパフォーマンスの劣化や、過剰なリソースの確保、リソース不足によるプロセスの停止につながる可能性がある

例) リソースが適切に配分できていない例

- タスク全体ではリソースに余力があるが、片方のコンテナでリソースが足りていない
- どちらのコンテナにも余力があるが、タスク全体でリソースを効率的に使えていない状態



タスク

CPUユニット数: 1024ユニット  
メモリ: 1024MB

を割り当てる



Nginxコンテナ

128 / 768MiB

128 / 768ユニット



Railsコンテナ

250 / 256MiB

250 / 256ユニット



タスク

CPUユニット数: 2048ユニット  
メモリ: 2048MB

を割り当てる



Nginxコンテナ

128 / 1536MiB

128 / 1536ユニット



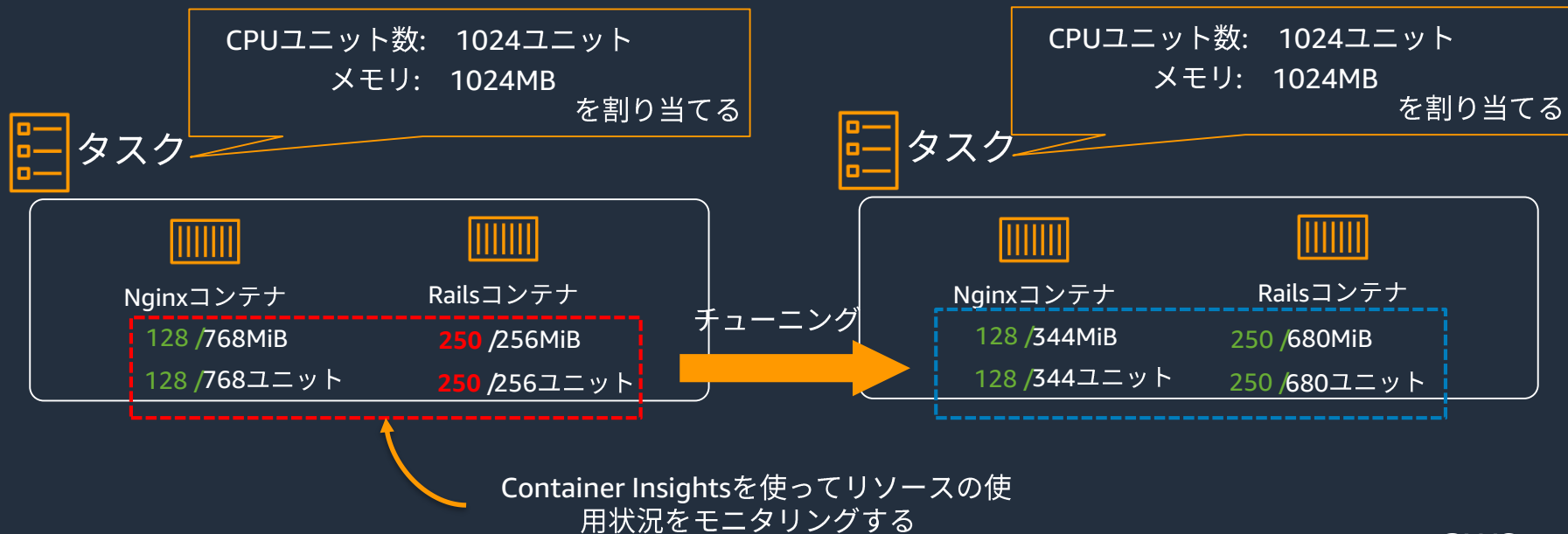
Railsコンテナ

250 / 512MiB

250 / 512ユニット

# Container Insightsを使って適切なリソース配分を行う

- Container Insightsはコンテナ単位のリソースモニタリングが可能であるため、各コンテナに適切なリソース配分が行われているかを確認できる
- 適切なリソース配分により、効率的かつ安全に運用できるリソースを割り当てる



# ステップ1. Containerごとのメトリクスを確認する

- Container Insightsの自動ダッシュボードから「ECS Tasks」を選択し、Containerごとのパフォーマンスを確認する



Container performance (4) アクション

Filter task definitions, container names, tasks...

Task definition	コンテナ名	平均 CPU (%)	平均メモリ (%)
task-demo	container-a	89.1227	85.6548
task-demo	container-b	9.6542	11.6954

片方のコンテナに負荷が偏っていることが判明した！

# ステップ.2 Containerリソース配分を変更する

- ECSコンソールのタスク定義からタスク内のコンテナリソース配分を変更する

container-a : メモリ256MB/CPU 128ユニット

container-b : メモリ256MB/CPU 128ユニット

container-a : メモリ384MB/CPU 192ユニット

container-b : メモリ128MB/CPU 64 ユニット

タスクサイズ

タスクサイズにより、タスクの固定サイズを指定できます。Fargate 起動タイプを使用したタスクにはタスクサイズが必須で、EC2 起動タイプではオプションです。タスクサイズが設定されている場合、コンテナレベルのメモリ設定はオプションです。タスクサイズは Windows コンテナではサポートされません。

タスクメモリ (GB)

0.25 vCPU の有効なメモリ範囲: 0.5GB - 2GB。

タスク CPU (vCPU)

0.5 GB メモリの有効な CPU: 0.25 vCPU

コンテナメモリの予約用のタスクメモリの最大割り当て

コンテナへの CPU の最大割り当て

コンテナの定義

コンテナの追加

コンテナ名	イメージ	ハード/ソフトウェア...	CPU ユニ...	GPU	Inference Accelerat...	基本	
container-a	888727018440.dkr.ec...	256/256	128			true	⊕
container-b	888727018440.dkr.ec...	256/256	128			true	⊕



タスクサイズ

タスクサイズにより、タスクの固定サイズを指定できます。Fargate 起動タイプを使用したタスクにはタスクサイズが必須で、EC2 起動タイプではオプションです。タスクサイズが設定されている場合、コンテナレベルのメモリ設定はオプションです。タスクサイズは Windows コンテナではサポートされません。

タスクメモリ (GB)

0.25 vCPU の有効なメモリ範囲: 0.5GB - 2GB。

タスク CPU (vCPU)

0.5 GB メモリの有効な CPU: 0.25 vCPU

コンテナメモリの予約用のタスクメモリの最大割り当て

コンテナへの CPU の最大割り当て

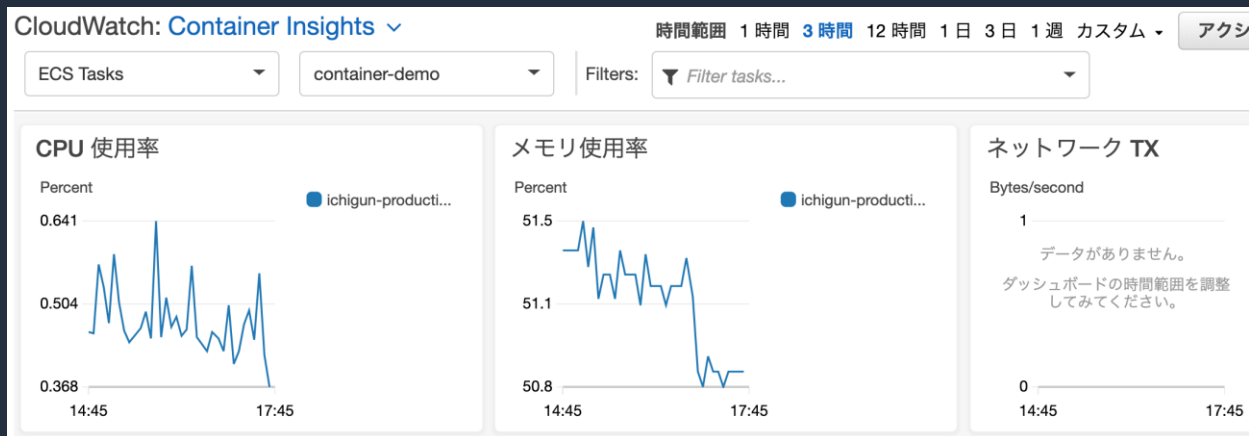
コンテナの定義

コンテナの追加

コンテナ名	イメージ	ハード/ソフトウェア...	CPU ユニ...	GPU	Inference Accelerat...	基本	
container-a	888727018440.dkr.ec...	384/384	192			true	⊕
container-b	888727018440.dkr.ec...	128/128	64			true	⊕

# ステップ3. 再びContainerごとのリソースを確認

- あらためて Container Insights からコンテナごとのリソース使用状況を確認する



Container performance (4)

Task definition	コンテナ名	平均 CPU (%)	平均メモリ (%)
task-demo	container-a	45.3721	50.2441
task-demo	container-b	42.9787	44.1687

適切なリソース割り当てができたことを確認！



Big Data

## ユースケース2.

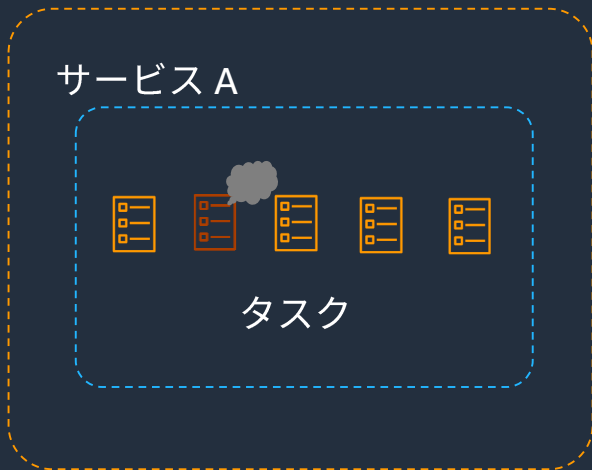
# 特定のタスクだけで発生している問題の調査を行いたい



# ユースケースの詳細

- サービスで起動している複数のタスクのうち、特定のタスクで問題が発生した場合に調査を行う

クラスタ



## 調査例

- 不定期に特定のリクエストのレスポンスが悪化する
- 定期的にタスクの再起動が発生している

# ステップ1. タスクごとのパフォーマンスを確認

- 特定のタスクのみリソースを著しく消費している
- 問題の特定のため、アプリケーションログを確認する
- 「アクション」 > 「アプリケーションログの表示」

Task performance (14)

アクション

Q ecsdemo-nodejs

<input type="checkbox"/>	Task definition	サービス	平均 CPU (%)	平均メモリ (%)	
<input type="checkbox"/>	ecsdemo-nodejs	ecsdemo-nodejs	6.4649	4.8314	
<input checked="" type="checkbox"/>	ecsdemo-nodejs	ecsdemo-nodejs	95.1297	94.9085	
<input type="checkbox"/>	ecsdemo-nodejs	ecsdemo-nodejs	5.0975	4.4271	
<input type="checkbox"/>	ecsdemo-nodejs	ecsdemo-nodejs	4.5607	3.8574	
<input type="checkbox"/>	ecsdemo-nodejs	ecsdemo-nodejs	4.5090	5.0148	

アプリケーションログの表示  
AWS X-Ray トレースの表示  
パフォーマンスログの表示

クラスタ

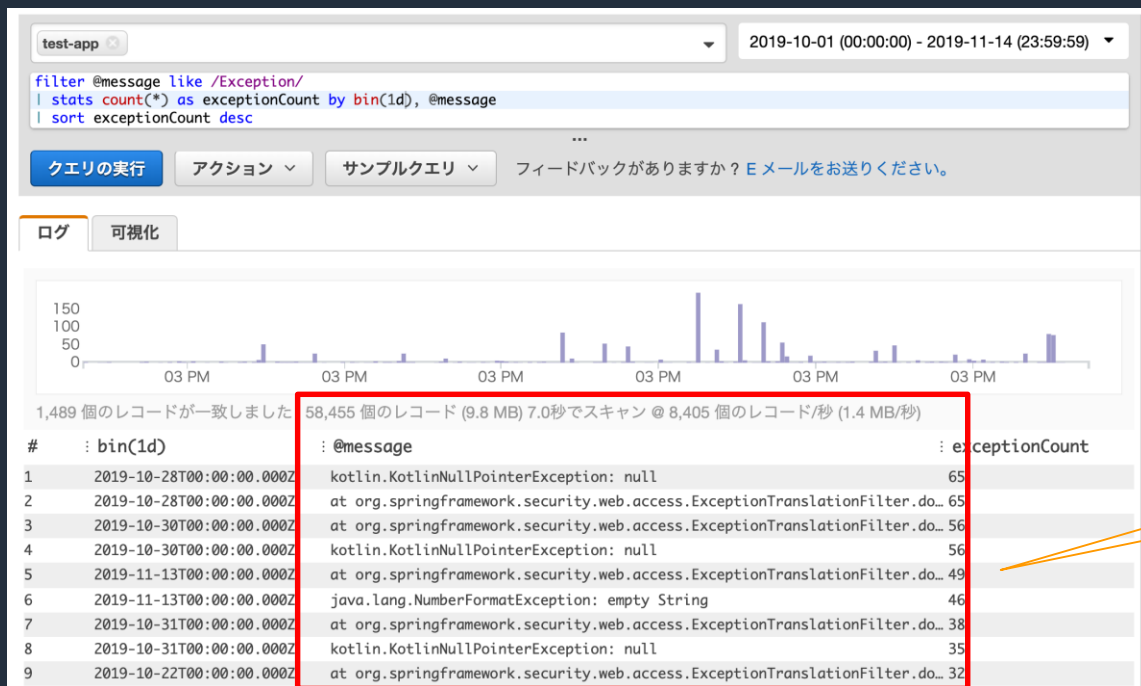
サービス A

タスク

リソース表示の観点

# ステップ2. CPU、メモリ使用率の高いタスクのログを確認

- アプリケーションログから異常が発生していないかを確認
- 異常が発生していない他のタスクとの比較も有効



@messageフィールド  
にログの内容が表示  
されている

# 料金について

# 料金について

Container Insights は以下の項目で課金される

## ECSのContainer Insightsの料金の例

前提:

10 個の Amazon EC2 インスタンス、50 個の平均的な実行中のコンテナ、20 個の一意的タスク名、5 個の一意的サービス名をもつ 1 つのコンテナクラスターをモニタリングする場合

CloudWatchカスタムメトリクス費用:

8 個のクラスターメトリクス + (6 個のタスクメトリクス \* 20 個の一意的タスク名) + (11 個のサービスメトリクス \* 5 個の一意的サービス名) = 183 CloudWatch メトリクス

183 \* 0.30USD = **54.90 USD**

CloudWatch Logs費用:

Amazon ECS では平均して 1 時間あたりメトリクスごとに 13 KB が取り込まれる  
(13 KB/1024/1024) GB \* 183 メトリクス \* 月平均 730 時間 = 月間 1.66 GB

1.66 GB \* 0.50USD = **0.83USD**

**合計: 月額55.73 USD**

■ AWS CloudWatchの料金

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

# まとめ

- Container Insightsはコンテナ化されたアプリケーションのメトリクスとログを収集、集計、要約できるCloudWatchの機能の一つ
- Container Insightsの登場により、サードパーティ製のツールや自前のカスタムメトリクスを使用しなくても、タスク、コンテナレベルでのモニタリングが可能になった
- CloudWatch 自動ダッシュボードを起点にタスク、コンテナレベルでのログ分析やパフォーマンスモニタリングなどの統合的な分析が可能

# Q&A

ご質問については、

AWS Japan Blog 「<https://aws.amazon.com/jp/blogs/news/>」

にて資料公開と併せて、後日掲載します



# AWS の日本語資料の場所「AWS 資料」で検索



日本担当チームへお問い合わせ サポート 日本語 ▼ アカウント ▼

コンソールにサインイン

製品 ソリューション 料金 ドキュメント 学習 パートナー AWS Marketplace その他 🔍

## AWS クラウドサービス活用資料集トップ

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

[AWS Webinar お申込 »](#)

[AWS 初心者向け »](#)

[業種・ソリューション別資料 »](#)

[サービス別資料 »](#)

<https://amzn.to/JPArchive>



# AWS Well-Architected 個別技術相談会

毎週”W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に  
対策などを相談することも可能

• 申込みはイベント告知サイトから

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

AWS イベント

で[検索]



# ご視聴ありがとうございました

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



過去資料  
<https://amzn.to/JPArchive>

