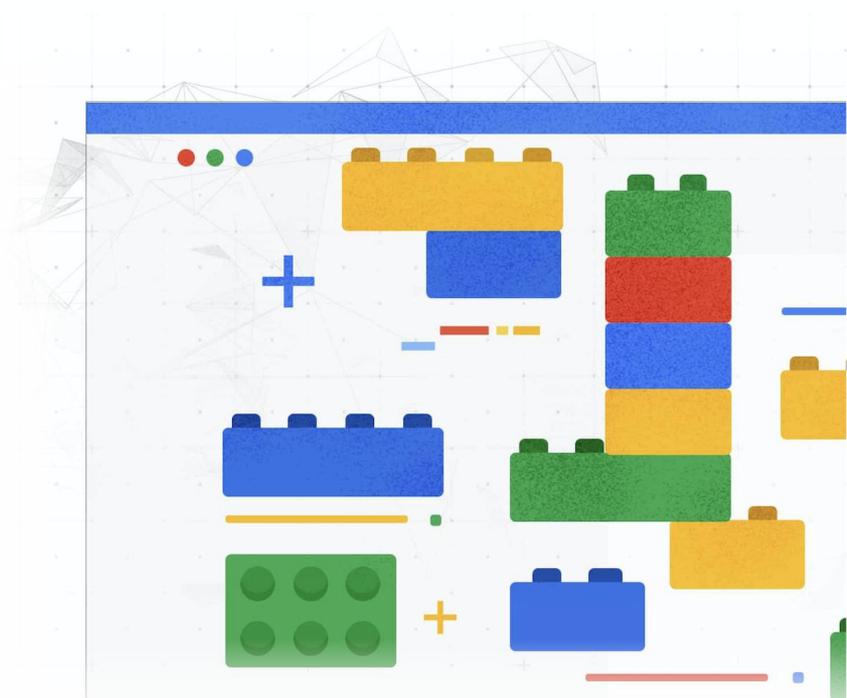


# Google Cloud で 実現する コンテナ時代の オブザーバビリティ

グーグル・クラウド・ジャパン 合同会社  
ソリューション アーキテクト  
長谷部 光治



# Who am I?

## 長谷部 光治

グーグル・クラウド・ジャパン 合同会社  
ソリューション アーキテクト

Areas of technology:

- ✓ Hybrid / Multi-cloud
- ✓ Application Development
- ✓ Infrastructure



# アジェンダ



1. コンテナ時代のオブザーバビリティ
2. ロギング
3. モニタリング
4. トレーシング
5. SLO モニタリング
6. まとめ

# コンテナ時代の オブザーバビリティ



# オブザーバビリティの主要 3 要素

## ログ

特定の時点における特定の作業の状態を表す **追記専用のファイル**  
メタデータの持ち方により構造化、非構造化の形式がある

## メトリクス (指標)

システム内で測定され、システムの状態を **測定可能な方法で表現** する  
通常数値で表され、カウンタ、分布、ゲージの形を取る

## トレース

分散システムでイベントを追跡するために使用され、**スパンで構成**される  
ウォーターフォール グラフで可視化され各処理時間がひと目で把握できる

<https://cloud.google.com/architecture/devops/devops-measurement-monitoring-and-observability>

# コンテナ

アプリケーション コードまたはバイナリと

その依存ファイル群を一つのユニットとしてまとめたもの

開発者とインフラ管理者の **責任分界点を明確化** できる

- **ポータビリティ**

オンプレミス、プライベートクラウド、パブリック

クラウド等、異なる実行環境間の移動が容易になる

- **軽量**

仮想マシンに比べてデータサイズが小さく、

起動にかかる時間も短い

またリソース使用量も少なく軽量



コンテナイメージ

アプリケーションコード

依存ファイル群

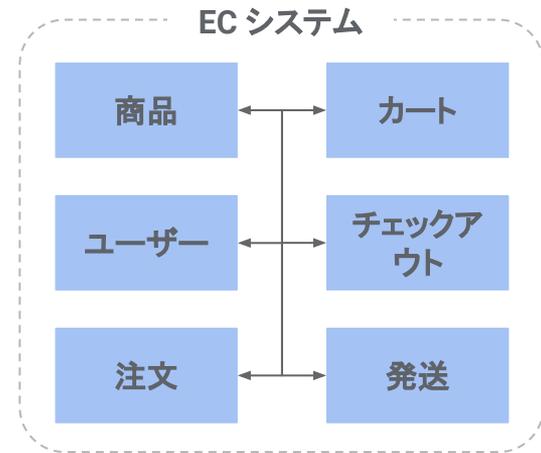
# マイクロサービス アーキテクチャ

## 特長

- 機能毎に独立したアプリケーション (サービス) に分割
- 各サービスは単一の役割をもつ
- 分散システム、サービス間は疎結合で軽量な API などで行き取り

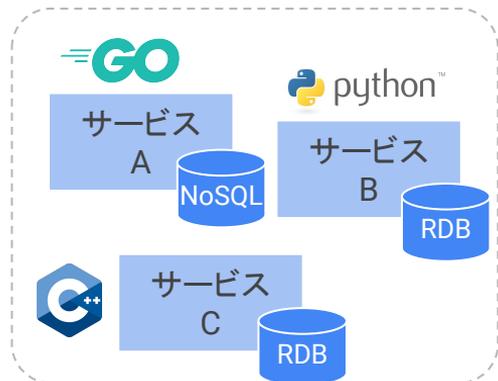


モノリシック



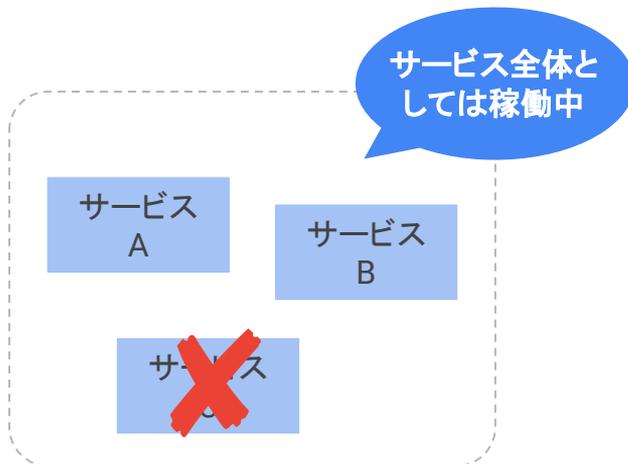
マイクロサービス

# マイクロサービス アーキテクチャ



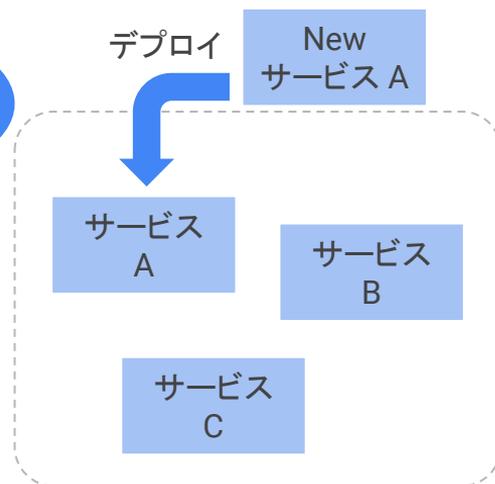
## 技術的異質性

各マイクロサービスに  
技術の選択肢を提供



## 回復力

あるサービスが  
停止してもサービス  
全体への影響を局所化



## 独立したデプロイ

サービス全体への  
影響無しに  
各サービスをデプロイ

マイクロサービスアーキテクチャ、コンテナは  
まず導入の**目的を明確化**し、メリット、デメリットを精査すべき

しかし、**無視はできない存在になってきている**

# コンテナ時代のオペレータビリティにおけるチャレンジ

## 運用対象リソースの 爆発的な増加、 環境の多様化

---

大量のコンテナで  
システムが構成される  
また複数のクラウド、  
オンプレミスを併用する  
ケースも増えている

## システムの複雑化

---

パブリック クラウドの  
マネージド サービス、SaaS、  
用途に合わせた  
ミドルウェアなど  
構成要素が増えている

## 権限の複雑化

---

各サービスごとに少人数の  
メンバーで開発することが  
増えており、チームごとに  
必要な権限のみを割り振る  
必要がある

# 今、オブザーバビリティソリューションに求められるもの

## 統合運用管理

ログだけ、メトリクスだけではなく、トレース、その他、問題発見、修正に役立つ情報を **統合して収集し、横断して可視化、分析** できる

## 自動化

手間がかからないよう収集、可視化、分析、また必要な場合はアラートを送る **プロセスが最大限自動化、省力化** されている

## 柔軟な 権限設定

ログ、メトリクス、トレース、またダッシュボードなどの情報を閲覧、操作できる **権限を柔軟に設定** できる

# Cloud Operations

Google Cloud 上で実行されるアプリケーションやシステムの  
一元管理を実現するための **統合管理ソリューション スイート**



Cloud Logging  
- ログ管理



Cloud Monitoring  
- メトリクス管理



Cloud Trace  
- トレース管理



Error Reporting  
- エラー分析



Cloud Profiler  
- プロファイリング



Cloud Debugger  
- リアルタイム デバッグ

※ 今回取り上げる主要プロダクト

# Google Cloud におけるコンテナ実行基盤



Google Kubernetes Engine (GKE)

フルマネージドな Kubernetes サービス

- Kubernetes をシンプルに  
自動でデプロイ、スケーリング、管理する



Cloud Run

サーバーレスコンテナ 実行基盤

- コンテナをスケーラブルに  
より簡単な利用する

# 想定するペルソナ



開発者

- ログ、メトリクスはどのように出力する？
- バグはどこで起こっている？



インフラ担当者

- このログ、メトリクスは誰が閲覧可能？
- インフラに問題は起こっていない？



セキュリティ担当者

- 誰がどのような修正を行った？
- 不審なアクセスは来ていない？

ロギング



# Cloud Logging



収集から分析まで End-to-end で機能を提供する **フルマネージドなログ管理基盤**

## 収集

GKE、Cloud Run といった Google Cloud プロダクトからのログを**自動的に収集**  
構造化、非構造化、双方の形式をサポート

## ルーティング

ログルーターを經由して**Cloud Storage、Pub/Sub、BigQuery**  
または Splunk といった**サードパーティツール**にエクスポート

## 保存

システムログ、データアクセスログは**デフォルトで30日保存、最長10年まで延長可能**  
管理者ログは**削除禁止のストレージに400日保存**

## 分析

**ログエクスプローラ**からリアルタイムにログを分析  
**Error Reporting**でアプリケーションのエラーを自動的に分析

# アプリケーションからログを出力する

開発者

インフラ  
担当者

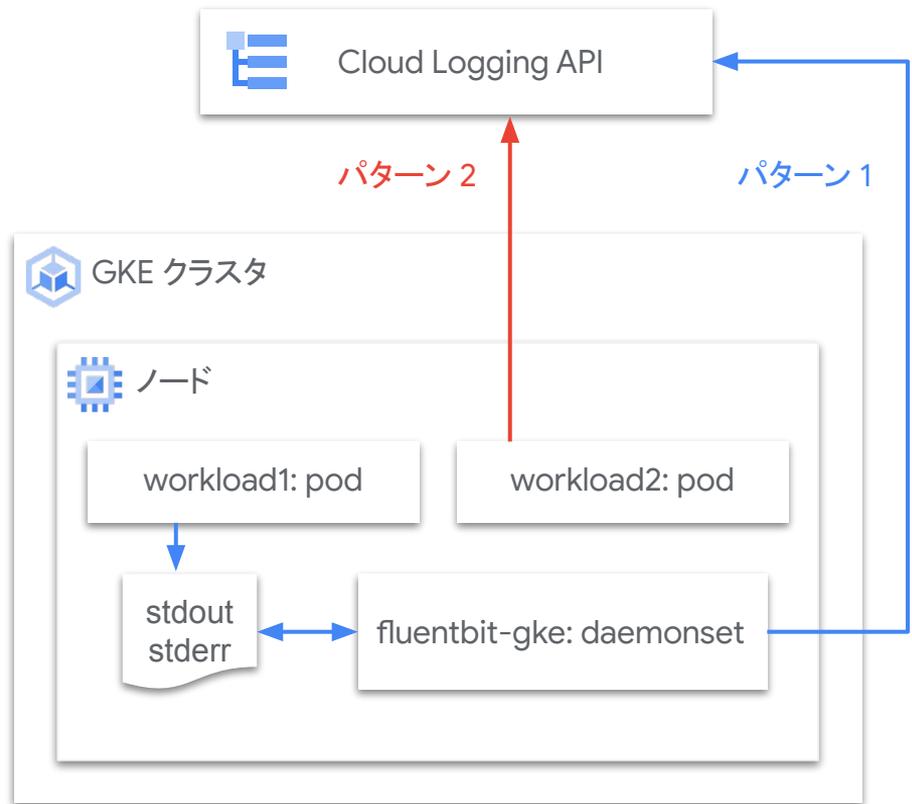
セキュリティ  
担当者

## GKE におけるログ出力の 2つの選択肢

1. 標準出力、標準エラー出力を利用する
2. 直接 Cloud Logging API を利用する

### → パターン 1 の方法をおすすめ

- アプリケーション コードにプラットフォーム特有の情報が入らない
- そのためアプリケーションのポータビリティが保たれる  
(ローカル、GKE でのロギングを同じコード、仕組みで実現できる)



# ログをリアルタイムで調査する

開発者

インフラ  
担当者

セキュリティ  
担当者

ログ エクスプローラ でリアルタイムで分析

## 主要機能

- ログの可視化と分析
  - ヒストグラム、ストリーム表示
- クエリの作成と絞り込み
  - 正規表現、期間での絞り込み
- コラボレーション
  - 共有クエリ

The screenshot displays the Google Cloud Log Explorer interface. At the top, there are navigation buttons for 'オプション' (Options), '範囲を絞り込む' (Filter range), and 'プロジェクト' (Project). Below this, a search bar contains a query: `resource.type="k8s_container":resource.labels.cluster_name="hello-cluster"`. A histogram shows the distribution of log entries over time, with a peak around 4:30 PM on February 13th. Below the histogram, a table lists log entries with columns for severity, timestamp, and log message. The interface also includes buttons for 'クエリを実行' (Execute query), 'クエリをクリア' (Clear query), '保存' (Save), 'ログをストリーム' (Stream logs), and 'リンクを共有' (Share link).

# アプリケーション エラーを自動的に集計する

開発者

インフラ  
担当者

セキュリティ  
担当者



アプリケーションで発生したエラーを  
リアルタイムに集計、表示

## 主要機能

- 複数回発生しているエラー  
(エラーグループ)を集計し  
修正が必要なものの可視化
- 新しくエラーグループが  
作成された際にアラートをトリガー

The screenshot shows the Error Reporting interface with the following elements:

- Header: Error Reporting | エラーグループ | 未解決, 確認済み | 自動再読み込み | すべてのサービス | すべてのバージョン
- Filter: エラーをフィルタ
- Table with columns: 解決ステータス, 発生回数 ↓, エラー, 発生場所

解決ステータス	発生回数 ↓	エラー	発生場所
未解決	5	Readiness probe failed: HTTP probe failed with statuscode: 503	k8s_pods
未解決	4	MountVolume.SetUp failed for volume "gke-metrics-agent-config-vol": failed to sync c...	k8s_pods

# ログ情報からアラートをトリガーする

開発者

インフラ  
担当者

セキュリティ  
担当者

## ログベースのアラート Preview

特定のログメッセージをトリガーに  
アラートを送信

## 利用例

- アプリケーションの重大なエラー
- 重要なインフラ設定の更新

## 設定手順

1. クエリを実行する
2. アラート作成をクリックする
3. アラートの詳細を入力する

ログ エクスプローラ オプション 範囲を絞り込む プロジェクト リンクを共有 学ぶ

クエリ 最近 (5) 保存済み (0) 候補 (1) クエリをクリア 保存 ログをストリーム クエリを実行

直近の1時間 severity=ERROR resource.type="k8s\_container" resource.labels.pod\_name="rule-evalua... クエリを編集

ログのフィールド ヒストグラム

ログのフィールド フィールドと値の検索

RESOURCE TYPE

- ✓ Kubernetes Container 消去 ×

SEVERITY

- ✓ Error 消去 ×

LOG NAME

stderr	6		
--------	---	--	--

PROJECT ID

kozzy-gmp-test01	6	> #	2022-02-
		> #	2022-02-
		> #	2022-02-

クエリ結果 6件

重大度	タイムスタンプ
	2022/02/2...

Alert details

Provide a name and description for this log alert.

Name example-alert

Choose logs to include in the alert

Create an inclusion filter to determine which logs are included in logs routing sink.

Alert query severity=ERROR resource.type="k8s\_container" resource.labels.pod\_name="rule-evaluator-6d7c5fc6d4-g4qjx"

Set notification frequency and autoclose duration

Configure the minimum amount of time between receiving notifications for logs that match this filter, and the duration to autoclose corresponding incidents.

Time between notifications 1時間  
Incident autoclose duration 7日

Who should be notified? (省略可)

When alerting policy violations occur, you will be notified via these channels.

Notification Channels

SAVE CANCEL

# ログ情報からメトリクスを作成する

開発者

インフラ  
担当者

セキュリティ  
担当者

## ログベースの指標 (メトリクス)

特定のログエントリの情報を  
メトリクスに変換する

## 利用例

- ログに出力される特定の処理の  
実行回数をカウント
- ログに出力されたレイテンシ情報を  
抽出しメトリクスに変換

### ← ログの指標の作成

以下の設定を構成して、ログベースの指標を定義および作成します。

**指標タイプ**

Counter  
Counts the number of log entries matching a given filter  
[Learn more](#)

Distribution  
Collects numeric data from log entries matching a given filter  
[Learn more](#)

**詳細**

ログ指標の名前 \*

説明  
この指標の説明を入力します (省略可)

単位  
この指標に適用される測定単位 (バイト、秒など)。カウンタ指標では、空白のままにするか、数字「1」を押入します。分布指標の場合は、「s」、「ms」などの単位をオプションとして入力できます。 [詳細](#)

**フィルタの選択** ログをプレビュー

ログベースの指標の定義

**フィルタの作成 \***  
ユーザー補助機能のオプションを表示するには、Alt+F1 キーを押します。

1
---

**ラベル**  
ラベルを使用すると、ログベースの指標に複数の時系列を含めることができます。 [詳細](#)

+ ラベルを追加

**指標を作成** キャンセル

## GUI から必要情報を 入力するだけ

Google Cloud

# ログの取り扱いをコントロールする

## Cloud Logging の主要要素

### ログ ルーター (どのログをどこに送る)

- ログのルーティング先 (別プロジェクトも選択可能)、フィルタ設定

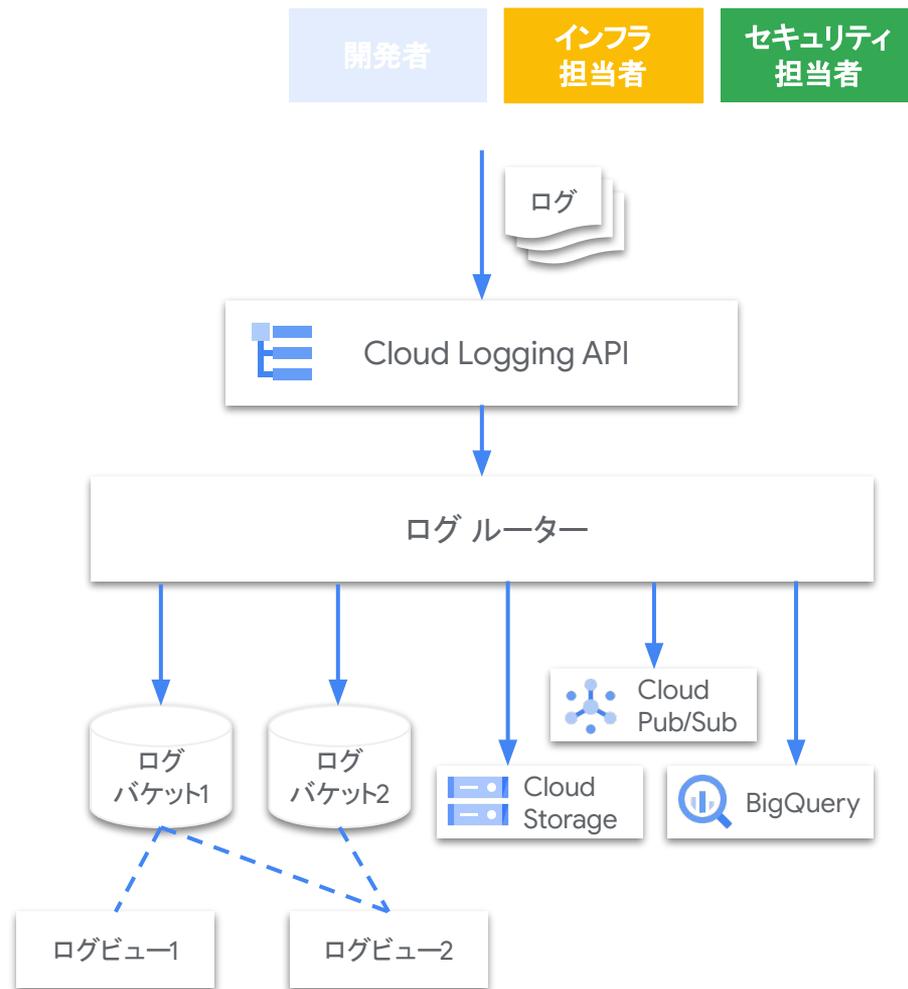
### ログバケット

(どのリージョンにどれだけ長く保存する)

- ログの保存

### ログビュー (誰がどのログを閲覧する)

- ログの閲覧権限を制御



# ログを活用する

## ログ ルーターから用途に応じて転送する

### BigQuery

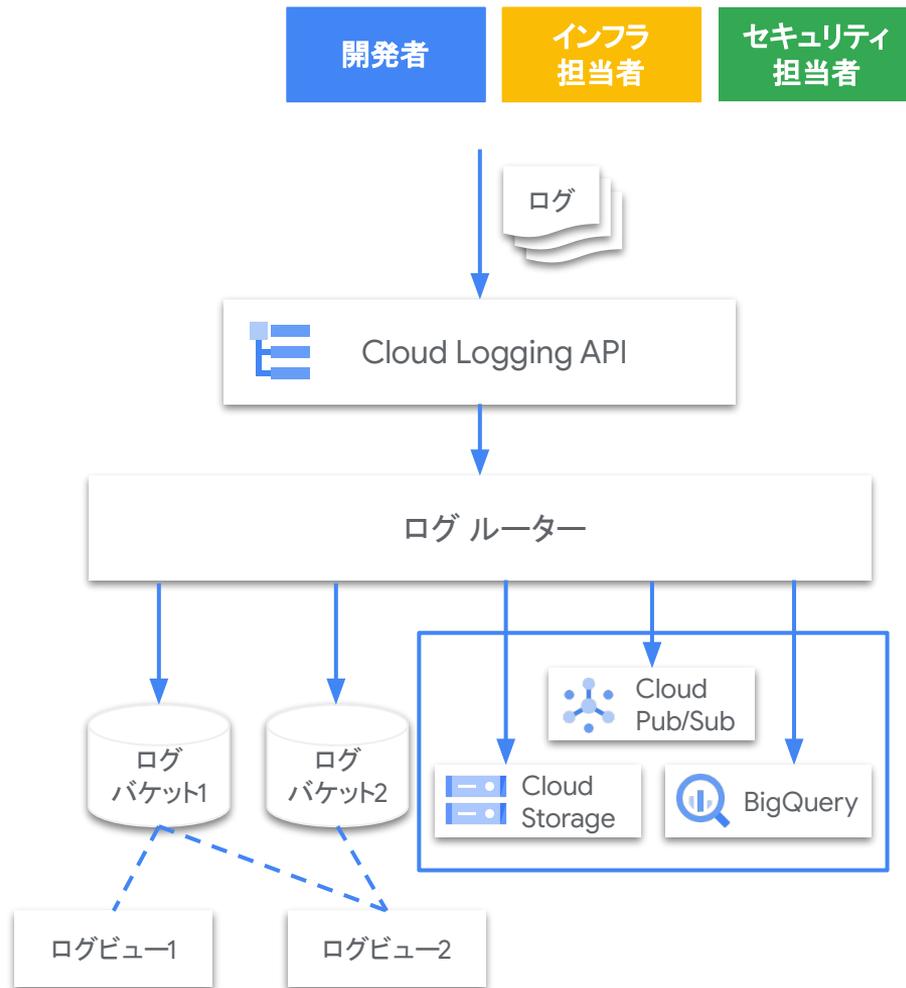
- サーバーレスなデータウェアハウス  
ログを分析し、インサイトを得る

### Cloud Pub/Sub

- スケーラブルなメッセージ配信サービス  
外部サービスにログを連携する

### Cloud Storage

- 高信頼なオブジェクト ストレージ  
ログを長期保存



# すべての環境からログを一元的に収集する

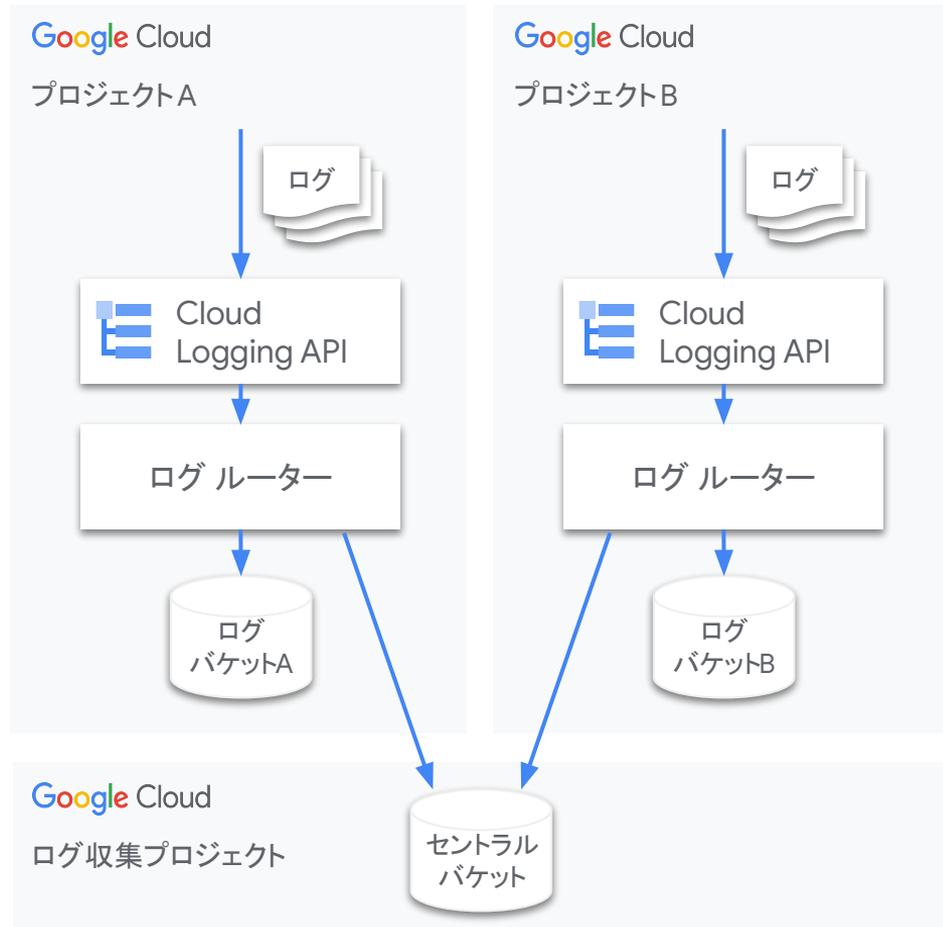
開発者

インフラ  
担当者

セキュリティ  
担当者

## 専用の Google Cloud プロジェクトに ログをまとめて保存する

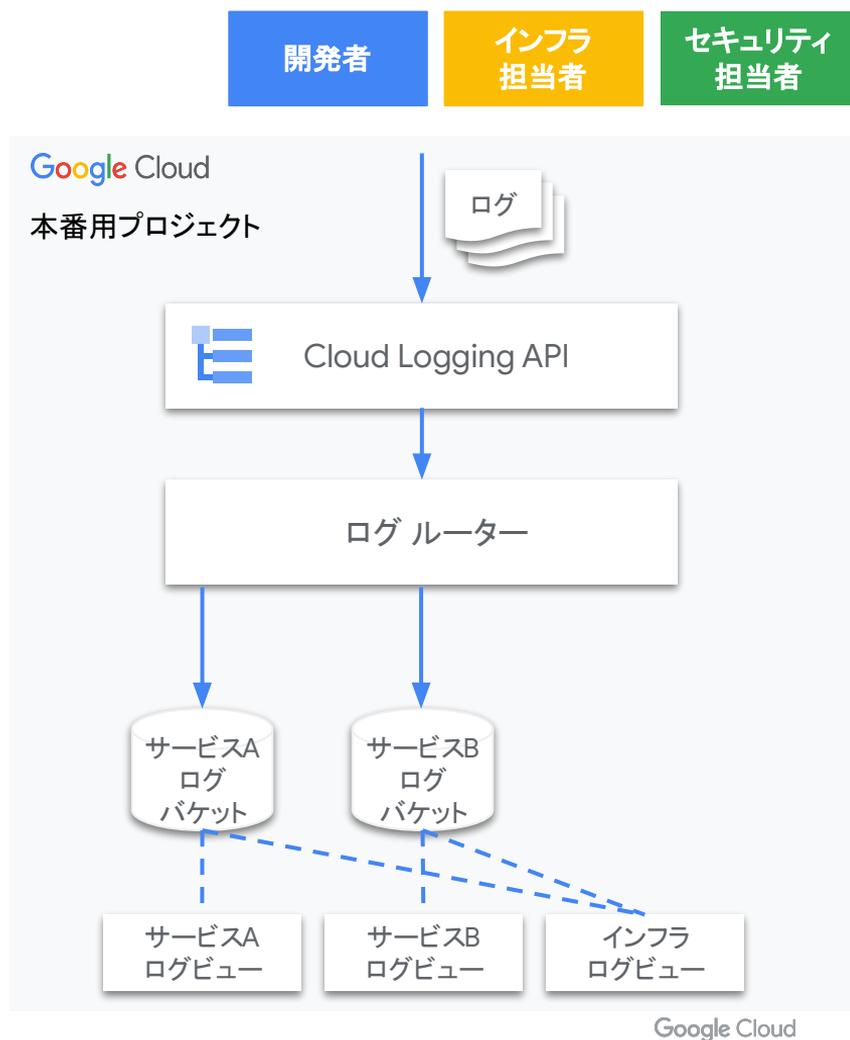
1. ログ収集用プロジェクトを用意
2. ログバケット (右図で示した  
セントラル バケット) を作成
3. 収集対象プロジェクトのログルーターで  
ログの転送先に手順 2 で作成した  
バケットを追加



# マイクロサービスごとにログを管理する

## 各マイクロサービスごとにログの管理を 制御する (1 Google Cloud プロジェクト構成)

1. マイクロサービスごとに  
専用のログバケットを作成
2. ログルーターのフィルタで各マイクロ  
サービスのログを専用のバケットに出力
3. 各ログバケットごとにログビューを作成
4. それぞれのログビューに開発者を割り当て



# ログを監査する

## 監査ログをセキュリティ担当者が管理

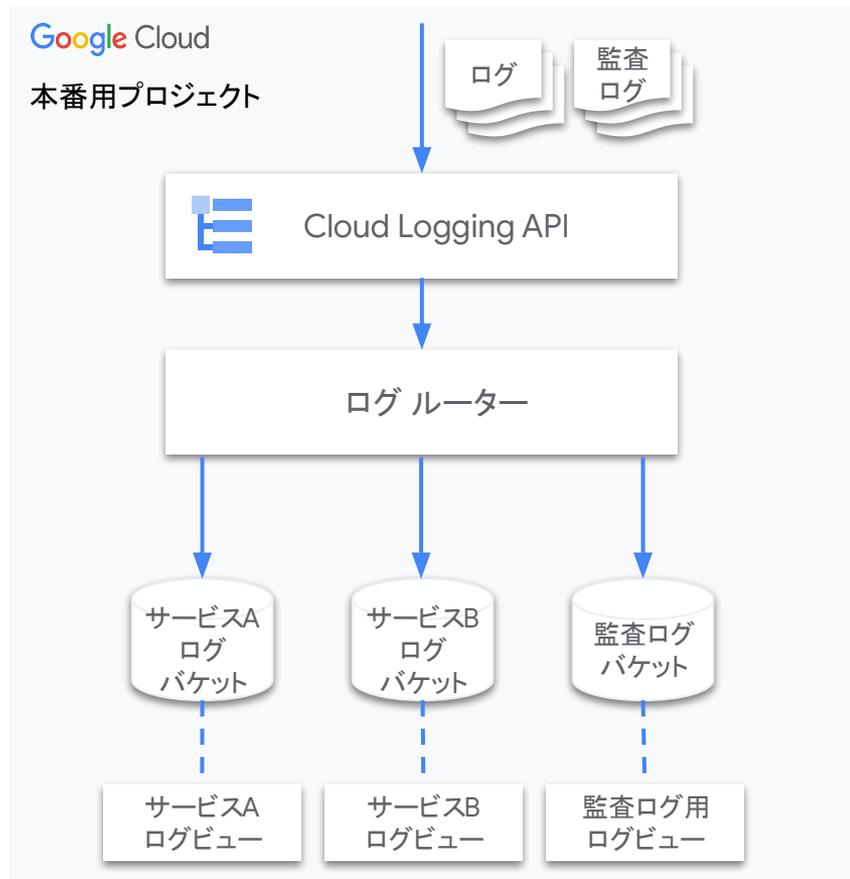
### 1. 監査対象のサービスで監査ログを有効化

タイトル ↑	管理読み取り	データ読み取り	データ書き込み	除外
<input type="checkbox"/> Cloud Run Admin API	-	-	-	0
<input type="checkbox"/> Kubernetes Engine API	-	-	-	0

### 2. 監査ログ用のログバケットを作成

### 3. ログルーターで監査ログを作成した ログバケットに転送する

### 4. 監査ログ用ログビューを作成、 セキュリティ担当者をアサイン



開発者

インフラ  
担当者

セキュリティ  
担当者

モニタリング



# Cloud Monitoring



アプリケーションとインフラストラクチャのパフォーマンス、可用性、健全性を可視化する

## 収集

5000 を超える Google Cloud に関するメトリクスを自動的に収集  
Google Cloud だけではなく、オンプレミス、他クラウドのメトリクスも収集可能

## アラート

収集されたメトリクスから条件を組み合わせアラートポリシーを作成  
E メール、SMS、Slack など複数の通知手段を用意

## ダッシュボード

20 を超えるすぐに使えるダッシュボード  
JSON 形式でダッシュボードをコードで管理

## 稼働時間の監視

世界中のポイントからサービスエンドポイントの可用性を監視  
プレビューでプライベートエンドポイントの監視も可能に

# システムメトリクスを利用する

開発者

インフラ  
担当者

セキュリティ  
担当者

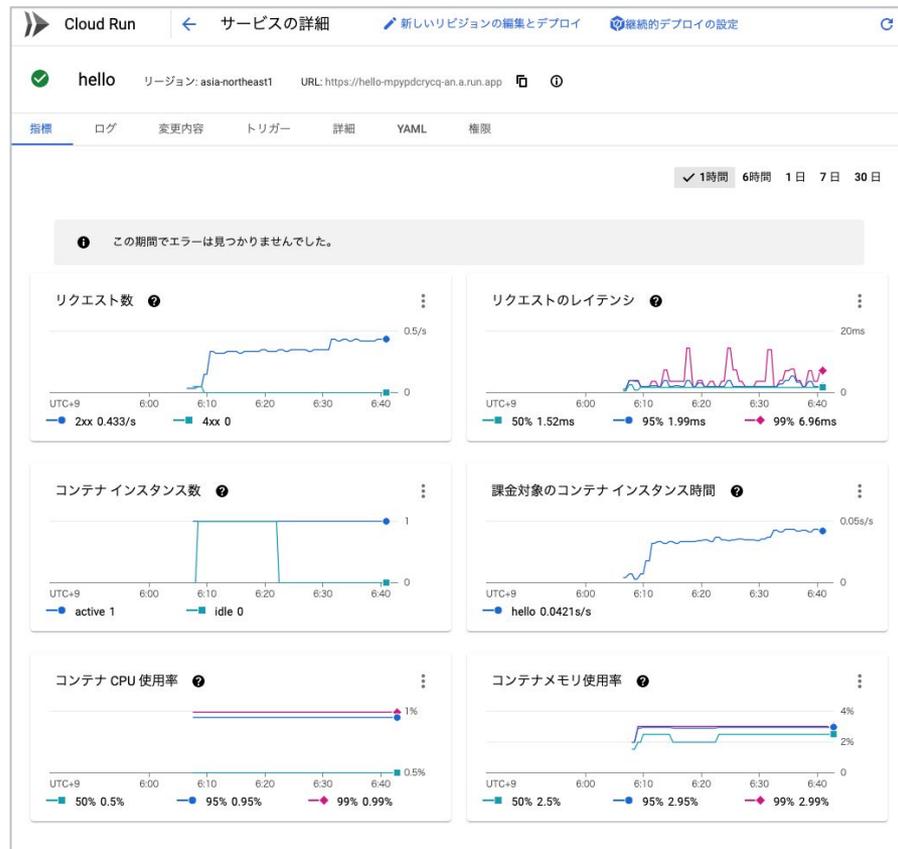
Google Cloud では自動的に  
様々なメトリクスが取得されている

## GKE

- コンテナごと、ノードごとの CPU、メモリ、ディスク、ネットワーク状況など

## Cloud Run

- リビジョンごとのインスタンス数、CPU、メモリ、ネットワーク状況、またリクエスト数、レイテンシなど



# アプリケーションからメトリクスを出力する

開発者

インフラ  
担当者

セキュリティ  
担当者

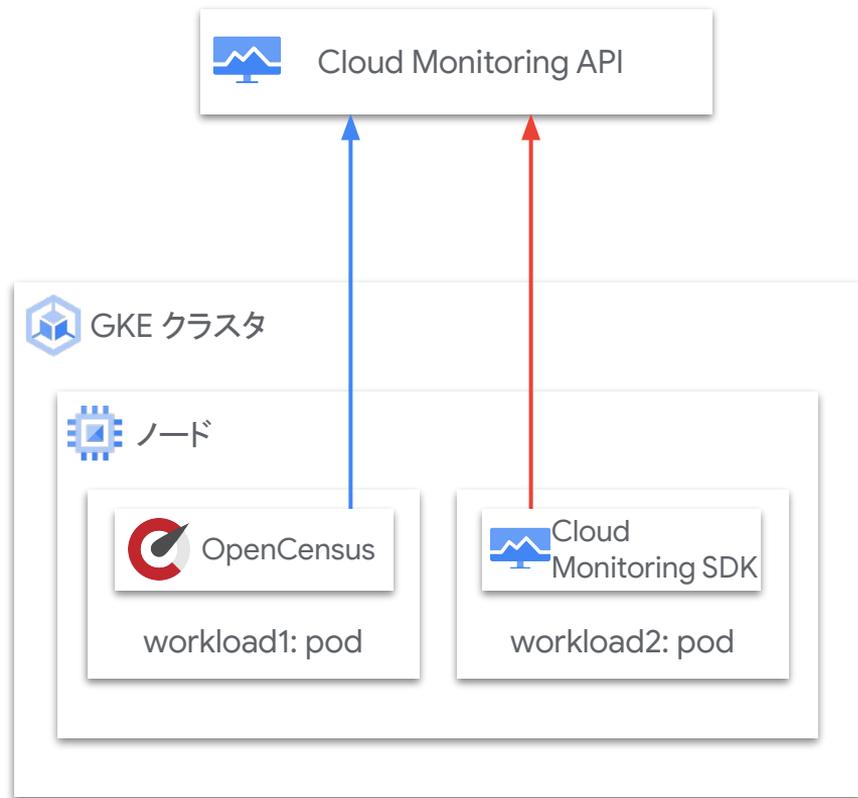
アプリケーションからカスタム メトリクスを  
Cloud Monitoring に送信する

OpenCensus (おすすめ)

- 汎用的な API を提供

Cloud Monitoring API

- SDK を使いアクセスできるが、  
低レベルかつ独自 API となっている



# メトリクスを Prometheus 形式で公開する

開発者

インフラ  
担当者

セキュリティ  
担当者

目的に合わせた 2 つの選択肢



最新アップデート  
2/28 に GA に  
なりました！

Google Cloud Managed Service  
for Prometheus (GMP)

Prometheus をスケーラブルに  
展開する Google 推奨の方法



Cloud Monitoring &  
GKE ワークロード指標 [Preview](#)

Kubernetes (GKE) アプリケーションを  
監視する Google 推奨の方法



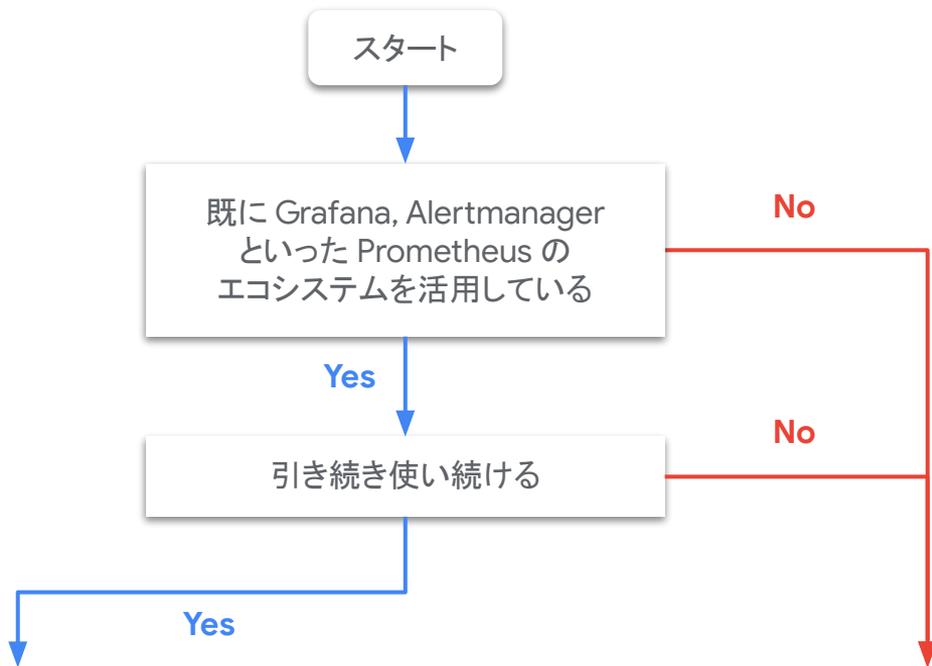
# メトリクスを Prometheus 形式で公開する

開発者

インフラ  
担当者

セキュリティ  
担当者

## 使い分けの考え方



Google Cloud Managed Service  
for Prometheus



Cloud Monitoring &  
GKE ワークロード指標 Preview

# GMP のメリット

開発者

インフラ  
担当者

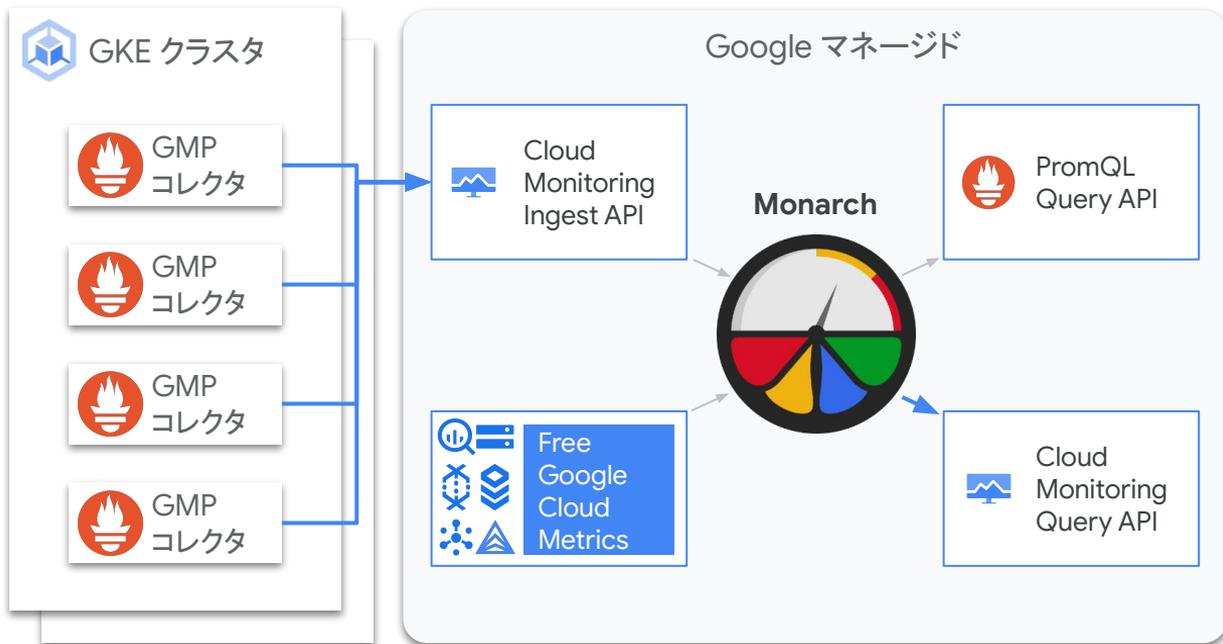
セキュリティ  
担当者

データストアに Google Managed の  
Monarch が使われている

GMP により

- 大規模な Prometheus の  
運用、管理
- データストアの運用、管理

から解放される



# グラフ、ダッシュボードを作成する

開発者

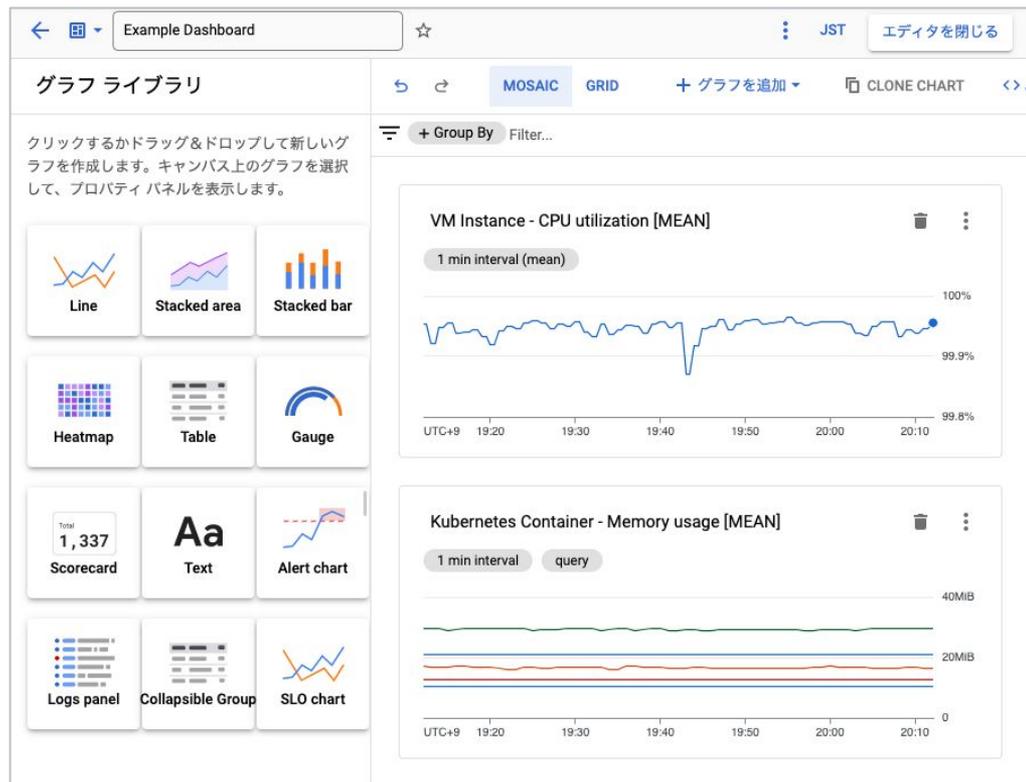
インフラ  
担当者

セキュリティ  
担当者

## メトリクスから

## グラフ、ダッシュボードを作成する

- 折れ線グラフ、ヒートマップ、表など複数の表示形式
- GUI コンソール、Monitoring Query Language (MQL) など複数の方法でグラフを作成



# グラフをポータブルな形で表現する

開発者

インフラ  
担当者

セキュリティ  
担当者

## Monitoring Query Language

表現力の高いテキストベースのインターフェース

以下のような複雑なグラフを作成できる

- 現在の値と過去の値の比率を計算する
- 時系列のサンプルをランダムに選択する
- パーセンタイル値を任意 (99.9% など) に指定する

東京リージョン (asia-northeast1-b) の GKE で稼働している Pod のメモリ (non-evictable) 使用量 トップ 5

```
fetch k8s_container
| metric 'kubernetes.io/container/memory/used_bytes'
| filter
    (resource.location == 'asia-northeast1-b')
    && (metric.memory_type == 'non-evictable')
| group_by 1m, [value_used_bytes_mean:
mean(value.used_bytes)]
| every 1m
| group_by [resource.pod_name],
    [value_used_bytes_mean_aggregate:
aggregate(value_used_bytes_mean)]
| top 5
```

# ダッシュボードをコードで管理する

開発者

インフラ  
担当者

セキュリティ  
担当者

複数のグラフを含んだダッシュボードを

**JSON 形式**で作成、管理できる

以下のようなことも可能になる

- コピー
- 共有
- 変更、世代管理

## ダッシュボード コード例

```
{
  "category": "CUSTOM",
  "displayName": "Sample dashboard01",
  "mosaicLayout": {
    "columns": 12,
    "tiles": [
      {
        "height": 4,
        "widget": {
          "title": "Memory usage top 5 in Tokyo",
          "xyChart": {
            "chartOptions": {
...以下、省略...
```

# アラートを管理する

開発者

インフラ  
担当者

セキュリティ  
担当者



# アプリケーションの稼働状況をチェックする

開発者

インフラ  
担当者

セキュリティ  
担当者

「稼働時間チェック」を利用して監視する

アラートとの連携も可能

## 監視対象 ●

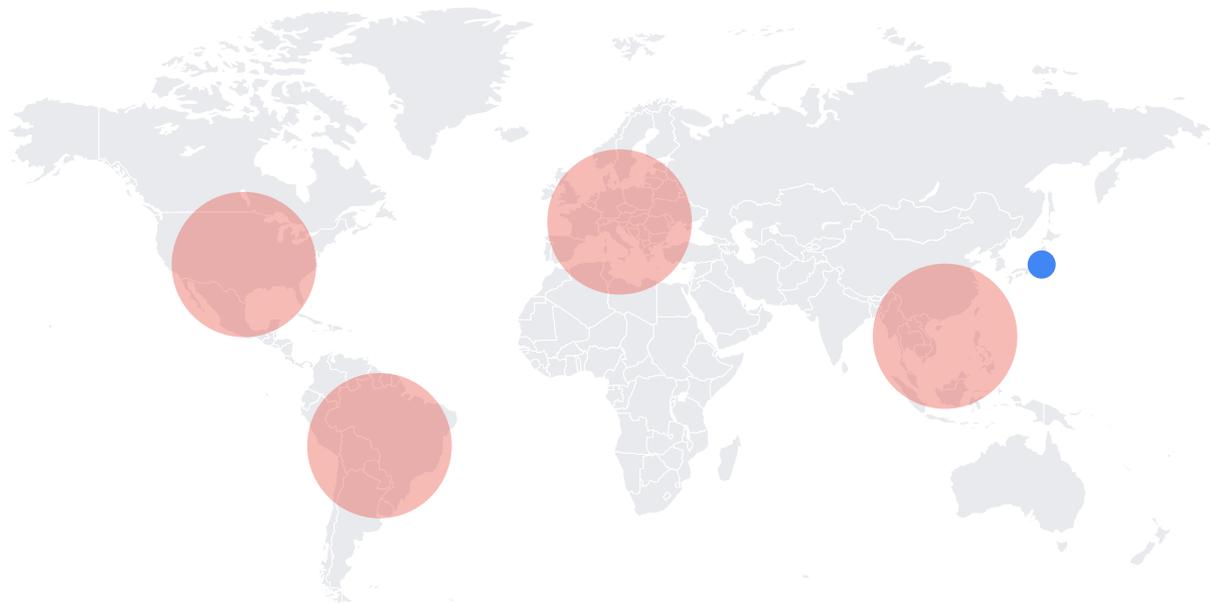
- 公開エンドポイント
- プライベート エンドポイント Preview

## 監視元 ●

- アジア, ヨーロッパ, 南北アメリカ

## メトリクスの生成

- 成功 / 失敗カウント, レイテンシ



# 複数プロジェクトのメトリクスを統合して管理する

開発者

インフラ  
担当者

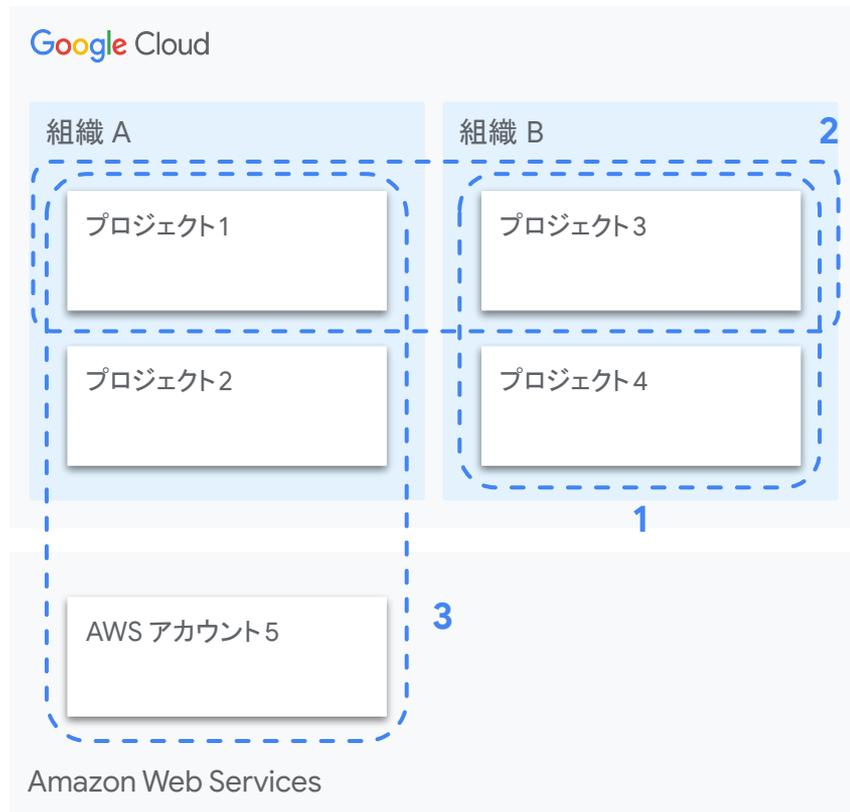
セキュリティ  
担当者

## メトリクス スコープにより

## 複数プロジェクトのメトリクスをまとめて表示

## パターン

1. 1つの組織内の複数のプロジェクト
2. 複数の組織に渡る複数のプロジェクト
3. 複数の Google Cloud プロジェクトと AWS アカウント



トレーシング





簡単に使用できる **フルマネージドな分散トレース システム**

## 収集

**OpenCensus、OpenTelemetry、クライアントライブラリ**からのトレースを収集  
Cloud Run、Cloud Functions、App Engine スタンダード環境からの**トレース自動収集**

## 表示

収集したトレーシングデータを複数の切り口から表示  
トレーシングデータから**ログ情報 (Cloud Logging)** へのシームレスな連携

## 分析

過去日とのパフォーマンス比較情報を含んだ**日次レポートを自動作成**  
**BigQuery にエクスポート**しトレース情報を分析 (Preview 機能)

# トレースデータをリアルタイムに確認する

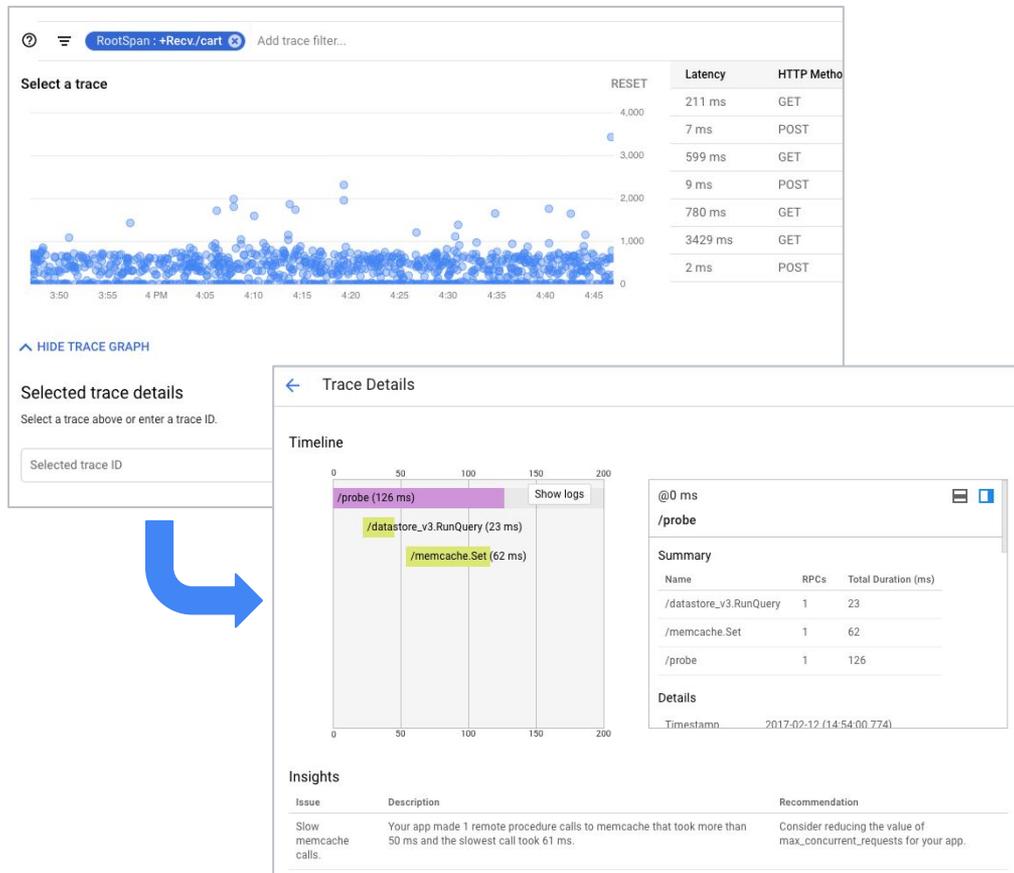
開発者

インフラ  
担当者

セキュリティ  
担当者

時間がかかっている処理を  
トレースデータを元に調査

1. トレースリストから  
遅いトレースを確認
2. トレースの詳細を確認し  
時間がかかっている処理を特定



# 複数プロジェクトのトレースを一元管理

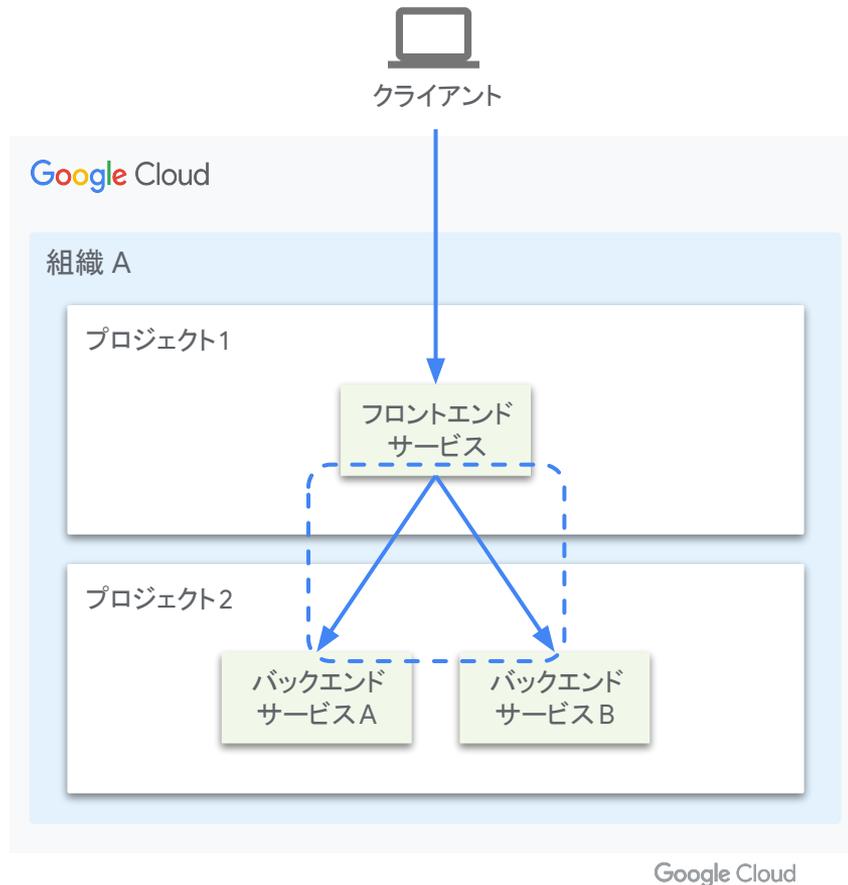
## 同じ組織内にあるプロジェクトの トレーススパンをまとめて表示

マイクロサービス アーキテクチャにおいて  
サービスごとに Google Cloud プロジェクトを  
分けている構成で有用

開発者

インフラ  
担当者

セキュリティ  
担当者



SLO モニタリング



# Google が本番システムを管理するプラクティス

開発者

インフラ  
担当者

セキュリティ  
担当者

## Google のサイト信頼性エンジニアリング (SRE) ではサービスレベル目標 (SLO) を用いて利用者の満足度を定量的に測定

- サービスレベル指標 (SLI)
  - パフォーマンスを測定する指標  
(0% ~ 100%)
- サービスレベル目標 (SLO)
  - パフォーマンスの目標値の定義
- エラーバジェット
  - パフォーマンスの実測値と目標値の差

例、

**SLI:** あるサービスの API のすべての HTTP レスポンスのうち 2xx (成功) を返す割合

**SLO:** 99%

**エラーバジェット:** 1% (100% - 99%)

# SLO モニタリングを簡単に実現する

開発者

インフラ  
担当者

セキュリティ  
担当者

## GUI から SLO モニタリングを設定

1. サービスを定義する
2. SLI を設定する
3. SLI の詳細を定義する
4. SLO を設定する
5. SLO を確認して保存する
6. (オプション) アラートを設定する

### サービスレベル目標 (SLO) の作成

- ✓ サービスレベル指標 (SLI) の設定  
パフォーマンス目標を設定するサービスヘルスの要素を選択します
- ✓ SLI の詳細を定義する  
選択した指標の詳細を指定します
- ✓ サービスレベル目標 (SLO) の設定  
サービスのパフォーマンス目標を設定します
- 4 確認と保存**  
詳細を確認し、SLO の名前を指定します

### 確認と保存

#### SLO の詳細

表示名

候補 [95% - 分布カット - 連続日](#)

#### SLO JSON プレビュー

現在のパラメータに基づいて、この JSON は S

```
1 {
2   "displayName": "95% - 正
3   "goal": 0.95,
4   "rollingPeriod": "86400s
5   "serviceLevelIndicator":
6     "requestBased": {
7       "distributionCut":
```

## 4つのゴールデンシグナル

SREプラクティスに示されているお客様向けシステムで  
最低限取得したほうが良い4つの重要なメトリクス

### レイテンシ

---

リクエストの処理に  
かかった時間  
成功と失敗したリクエス  
トのレイテンシは  
区別することが重要

### トラフィック

---

どれだけシステムに  
負荷がかかっているか  
Webサービスの場合通  
常秒間リクエスト数が使  
われる

### エラー

---

リクエストが失敗した割  
合。エラーには HTTP  
status-code 500 のよ  
うな明示的なものだけで  
はなく、暗黙的なものも  
含まれる

### 飽和度

---

システムがどれだけ  
飽和しているか  
ボトルネックとなる  
リソース (メモリの場合  
はメモリ使用量, I/O の  
場合は I/O 使用状況)  
を利用する

<https://sre.google/sre-book/monitoring-distributed-systems/>

# サービスマッシュでゴールデンシグナルを取得する

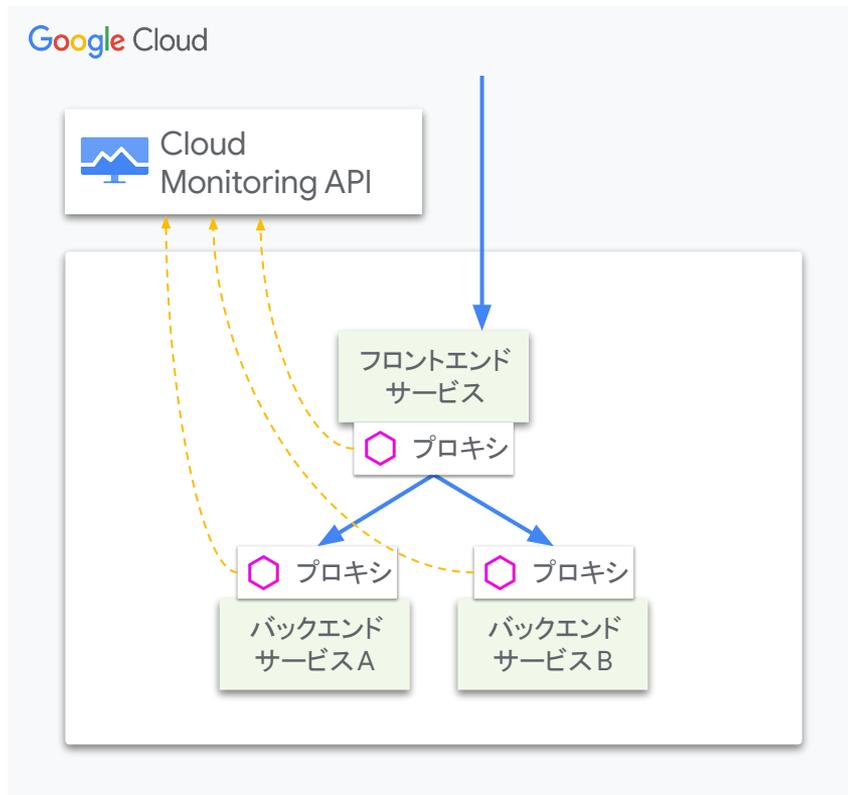
開発者

インフラ  
担当者

セキュリティ  
担当者

## Anthos Service Mesh を導入し ゴールデンシグナルを自動取得する

- サービスメッシュではプロキシがサイドカーとして導入される
- 導入されたプロキシがログ情報、メトリクスを出力する
- 結果、アプリケーションを修正せずに有用な情報が取得できる



まとめ



# コンテナ時代に対応するオペレータビリティを提供する Cloud Operations

- **フルマネージド**
- 各プロダクトが連携しており **複数の視点を用いて横断的に問題を発見、分析、修正**
- **自動的に取得される** 各種ログ、メトリクス、監査ログ
- **オンプレミス、他クラウド** も管理
- 組織、プロジェクト、サービス構成に合わせた **柔軟な権限設定**
- コードでダッシュボードを管理することによる **再利用性の向上**



Cloud Logging  
- ログ管理



Cloud Monitoring  
- メトリクス管理



Cloud Trace  
- トレース管理



Error Reporting  
- エラー分析



Cloud Profiler  
- プロファイリング



Cloud Debugger  
- リアルタイム デバッグ

# Google が培ってきたプラクティスを素早く取り込める SLO モニタリング

収集されたメトリクスを用いて

利用者の満足度を定量的に測定

コードに手を入れず

ゴールデンシグナルを取得できる

サービス メッシュ

## サービスレベル目標 (SLO) の作成

- ✓ サービスレベル指標 (SLI) の設定  
パフォーマンス目標を設定するサービスヘルスの要素を選択します
- ✓ SLI の詳細を定義する  
選択した指標の詳細を指定します
- ✓ サービスレベル目標 (SLO) の設定  
サービスのパフォーマンス目標を設定します
- 4 確認と保存**  
詳細を確認し、SLO の名前を指定します

## 確認と保存

SLO の詳細

表示名

候補 [95% - 分布カット - 連続日](#)

SLO JSON プレビュー

現在のパラメータに基づいて、この JSON は S

```
1 {
2   "displayName": "95% - 正
3   "goal": 0.95,
4   "rollingPeriod": "86400s
5   "serviceLevelIndicator":
6     "requestBased": {
7       "distributionCut":
```

Google Cloud

Google Cloud でワークロードを動かしているなら  
既に**多数の有用なログ、メトリクスが収集**されています。

それらを最大限活用し、**今日からオブザーバビリティを向上**させませんか？

**Thank you**