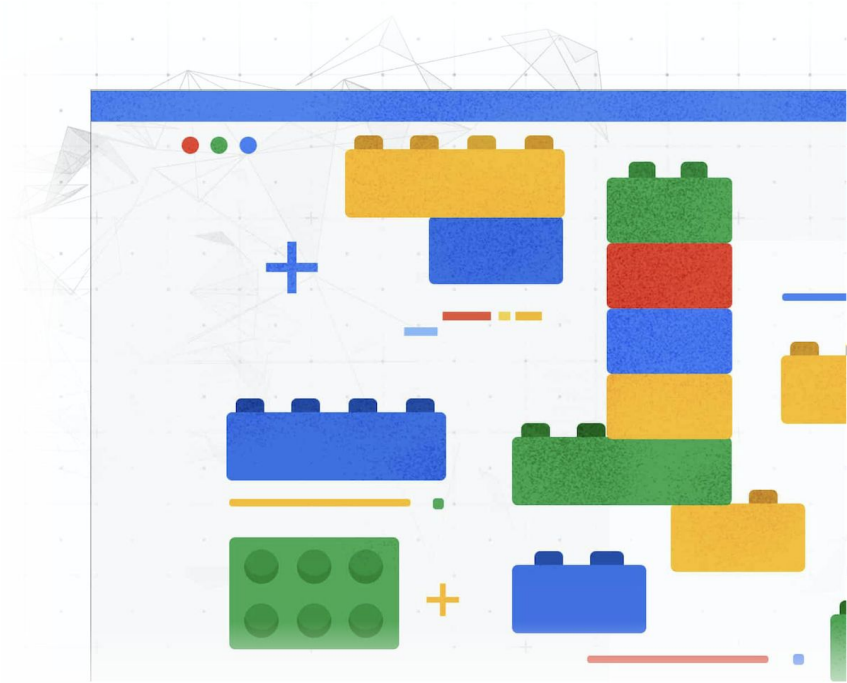
Google Cloud で実現するコンテナ時代のオブザーバビリティ



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

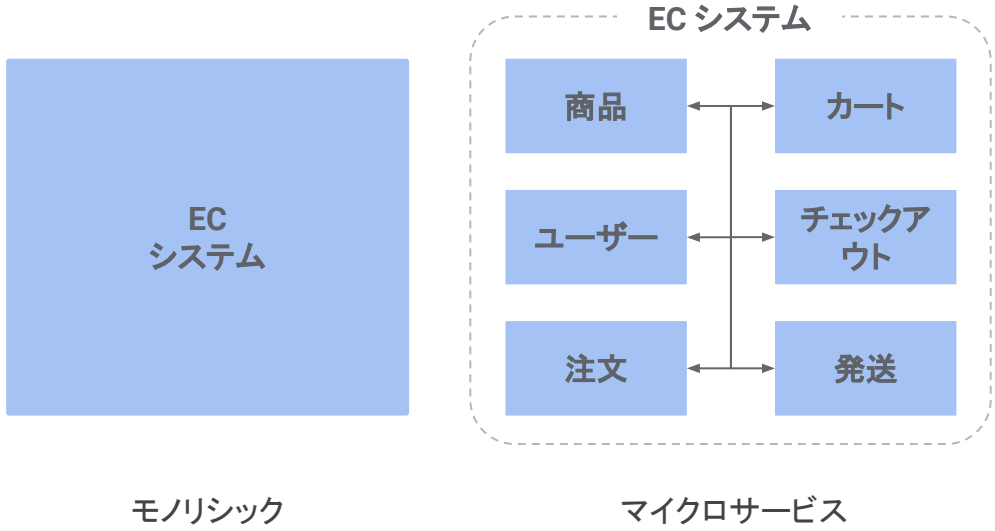
開発者とインフラ管理者の **責任分界点を明確化** できる

- **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

マイクロサービス アーキテクチャ

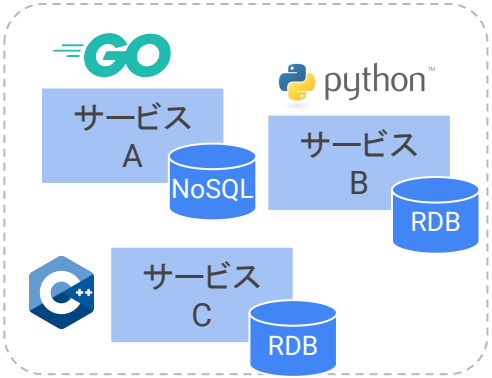
特長

- 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 などでやり取り



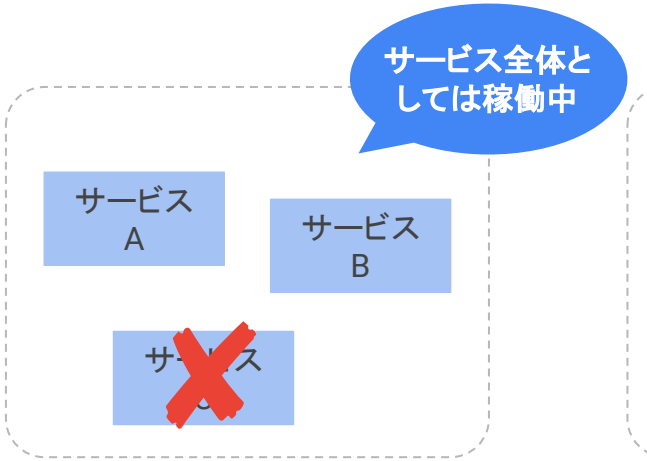
モノリシック　　マイクロサービス

マイクロサービス アーキテクチャ



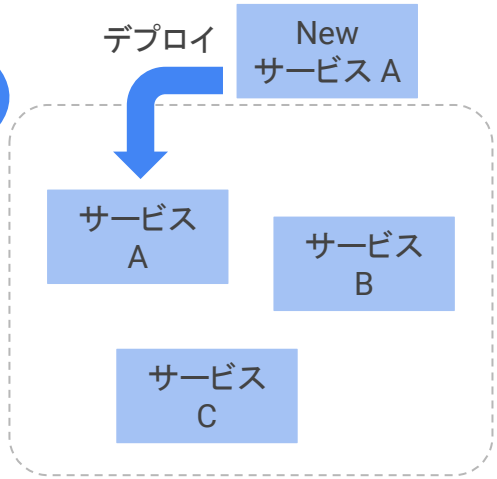
技術的異質性

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るべき

しかし、**無視はできない存在になってきている**

コンテナ時代のオブザーバ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

02

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

ログルーターを経由して**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

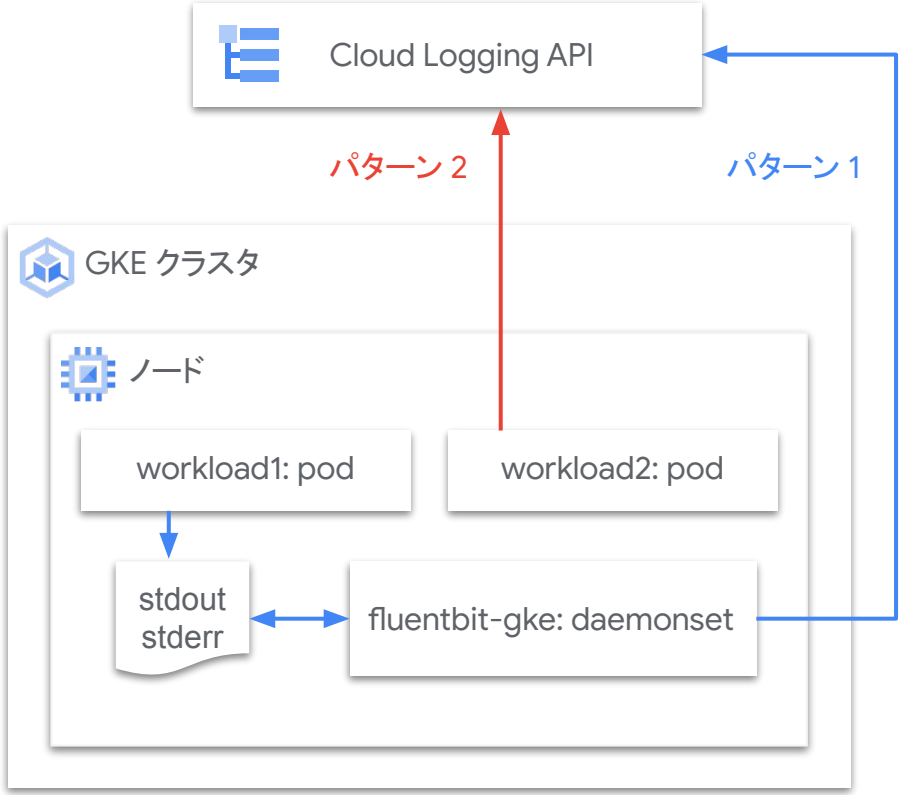
GKE におけるログ出力の **2 つの選択肢**

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)



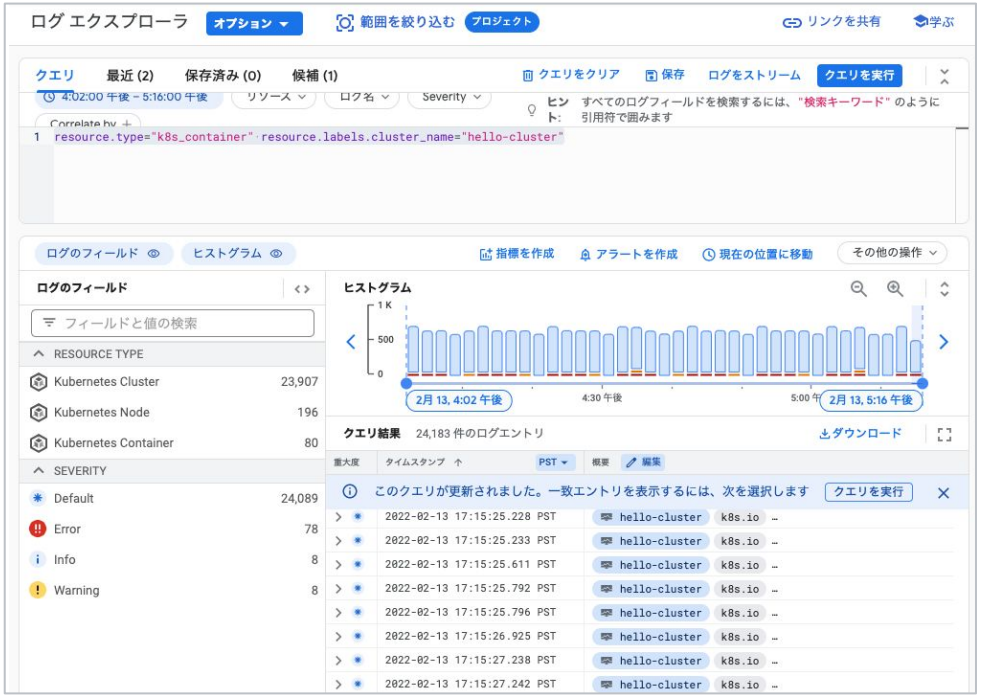
ログをリアルタイムで調査する

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



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

Error Reporting

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

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

GUI から必要情報を
入力するだけ

ログの取り扱いをコントロールする

開発者 | インフラ担当者 | セキュリティ担当者

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

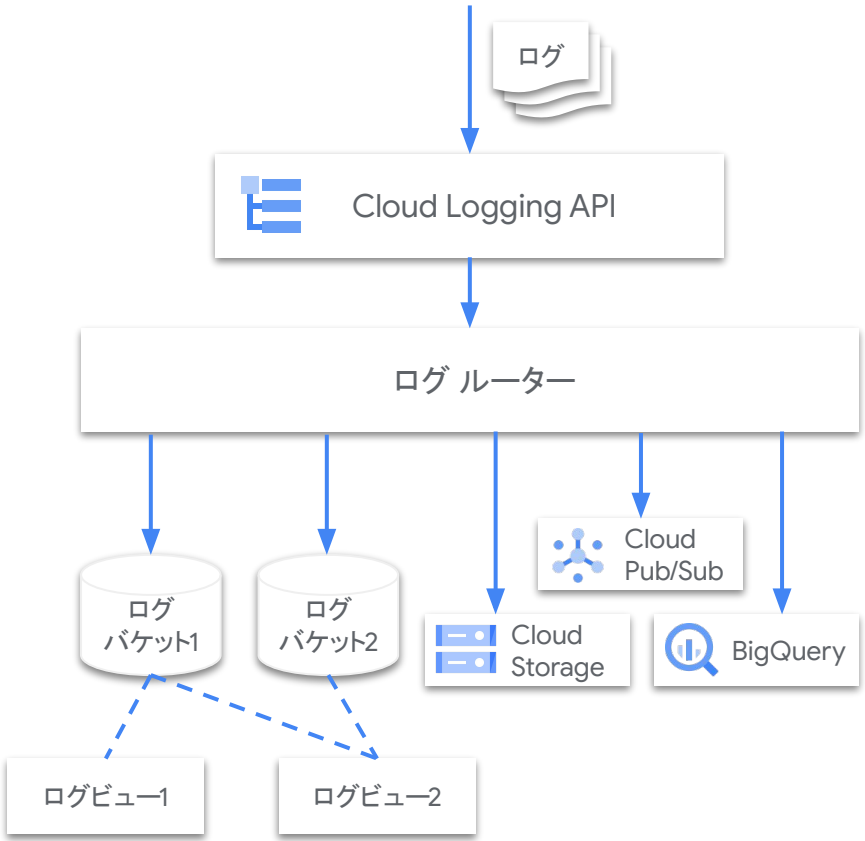
ログバケット

（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

開発者 | インフラ担当者 | セキュリティ担当者

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

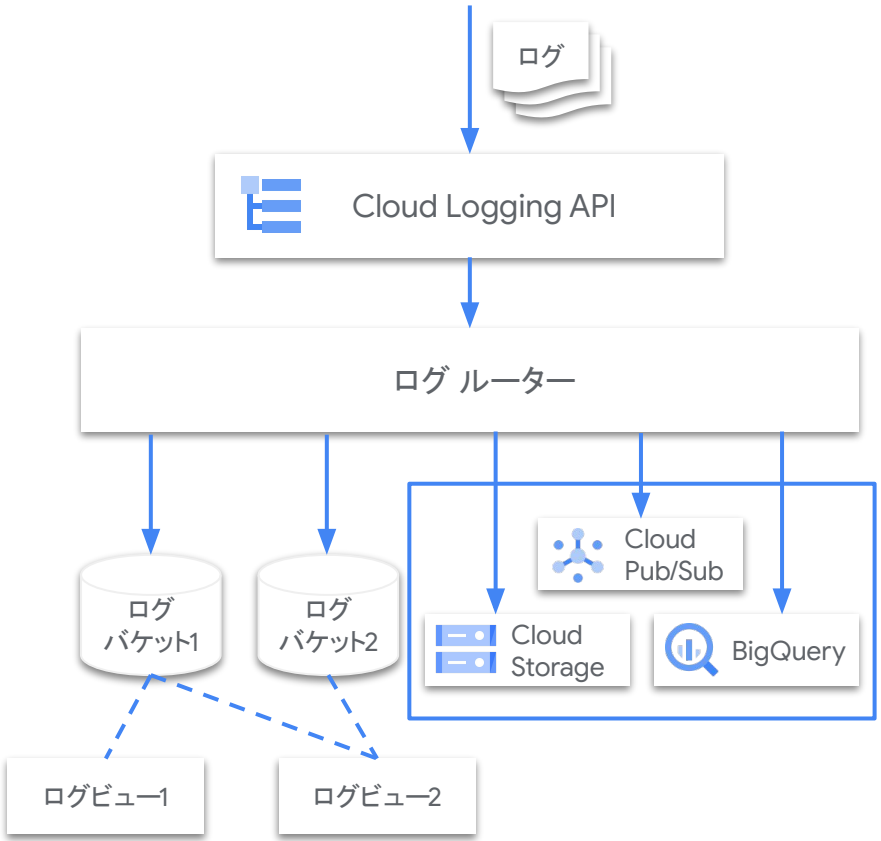
- スケーラブルなメッセージ配信サービス

  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

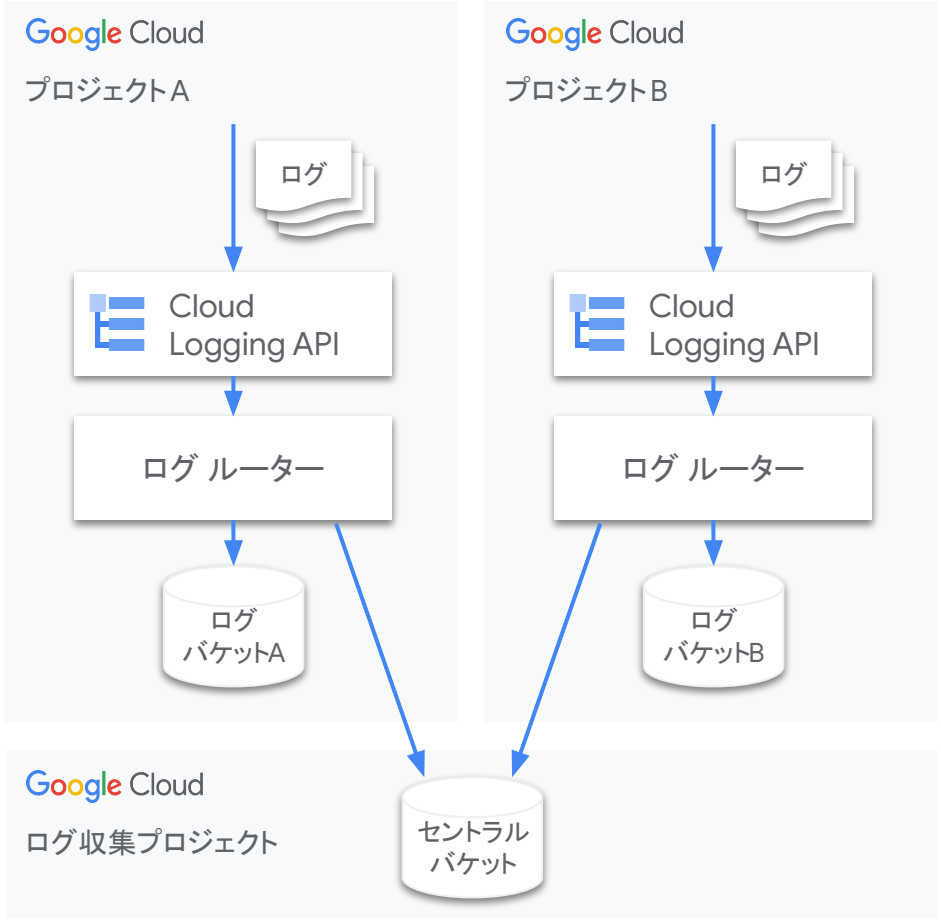


すべての環境からログを一元的に収集する

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

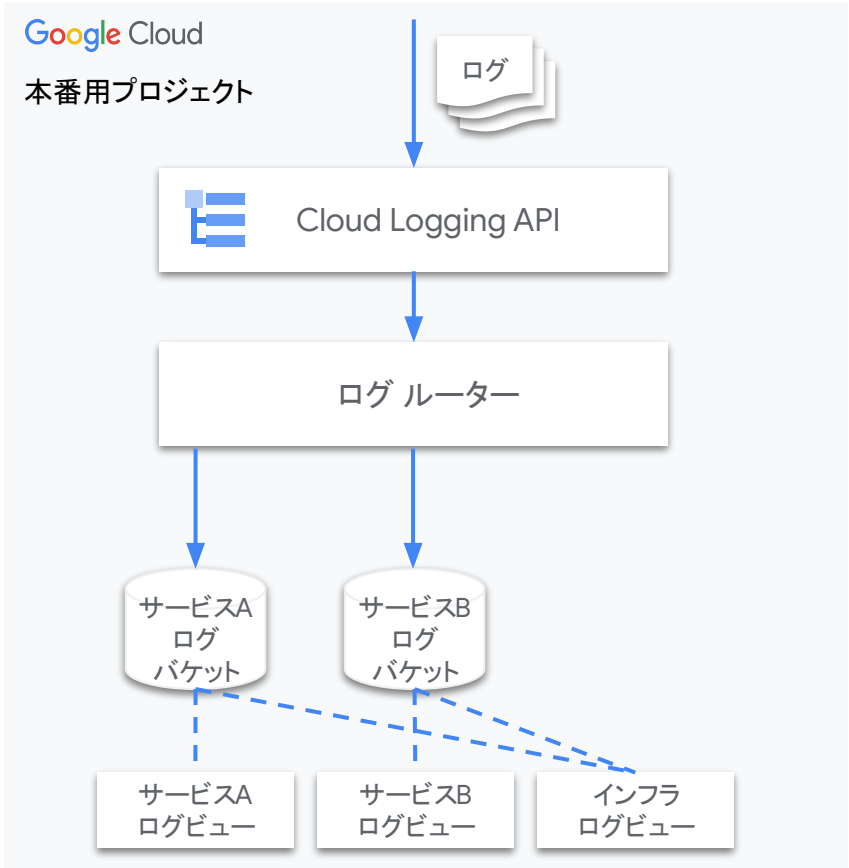


マイクロサービスごとにログを管理する

開発者 | インフラ担当者 | セキュリティ担当者

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



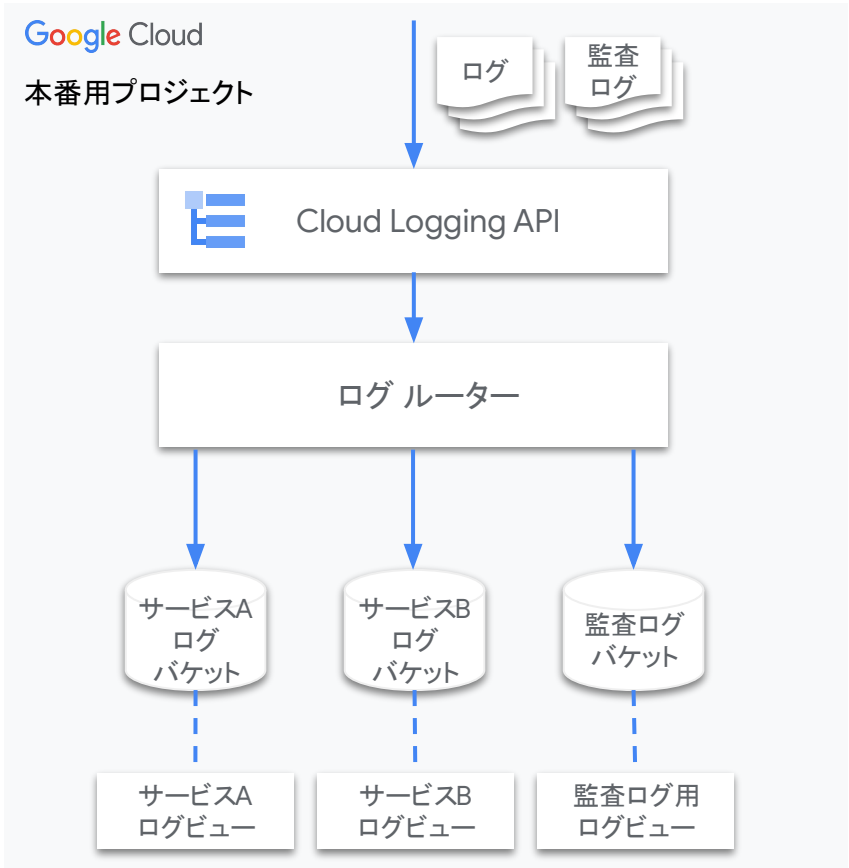
ログを監査する

開発者 | インフラ担当者 | セキュリティ担当者

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

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



モニタリング

03

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

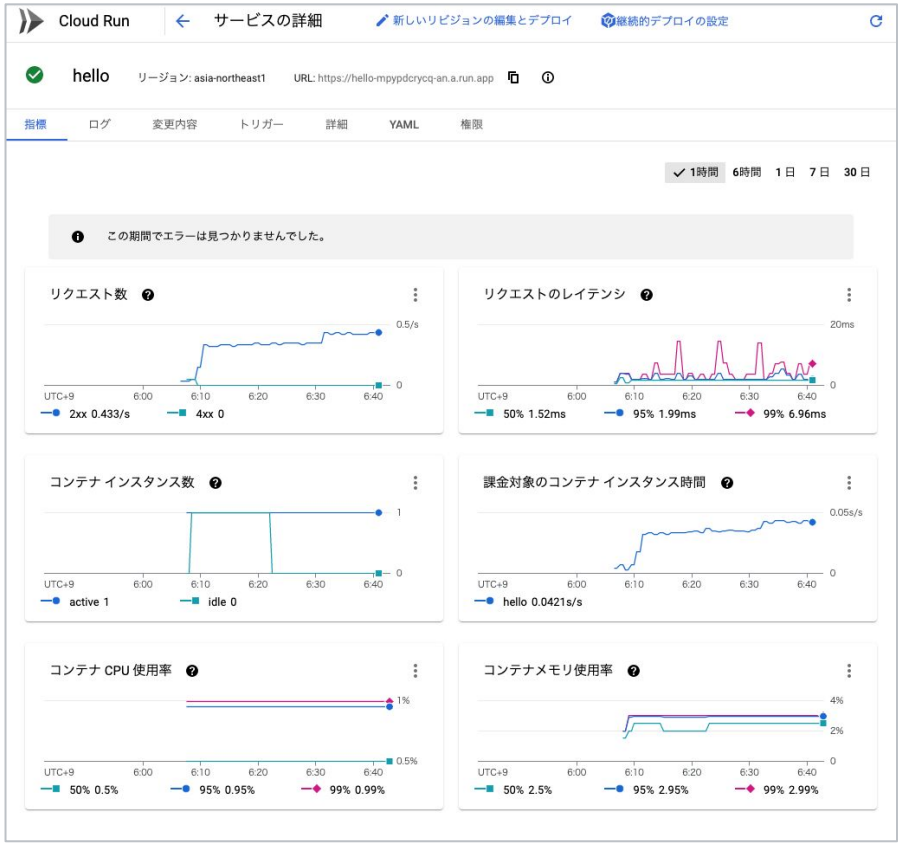
Google Cloud では自動的に
様々なメトリクスが取得されている

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

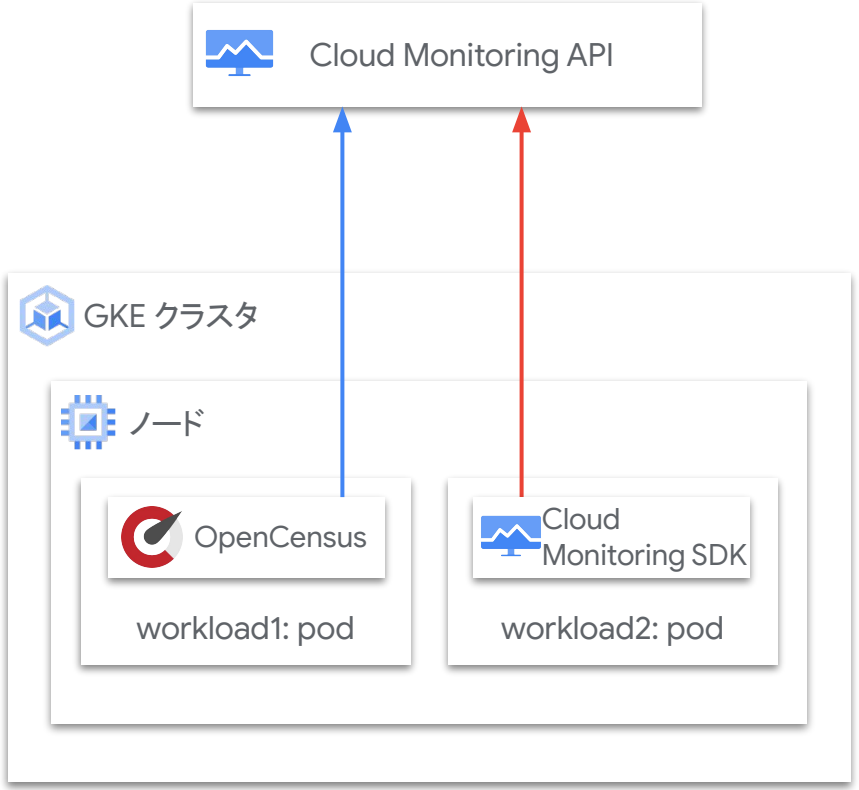
アプリケーションからカスタム メトリクスを Cloud Monitoring に送信する

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

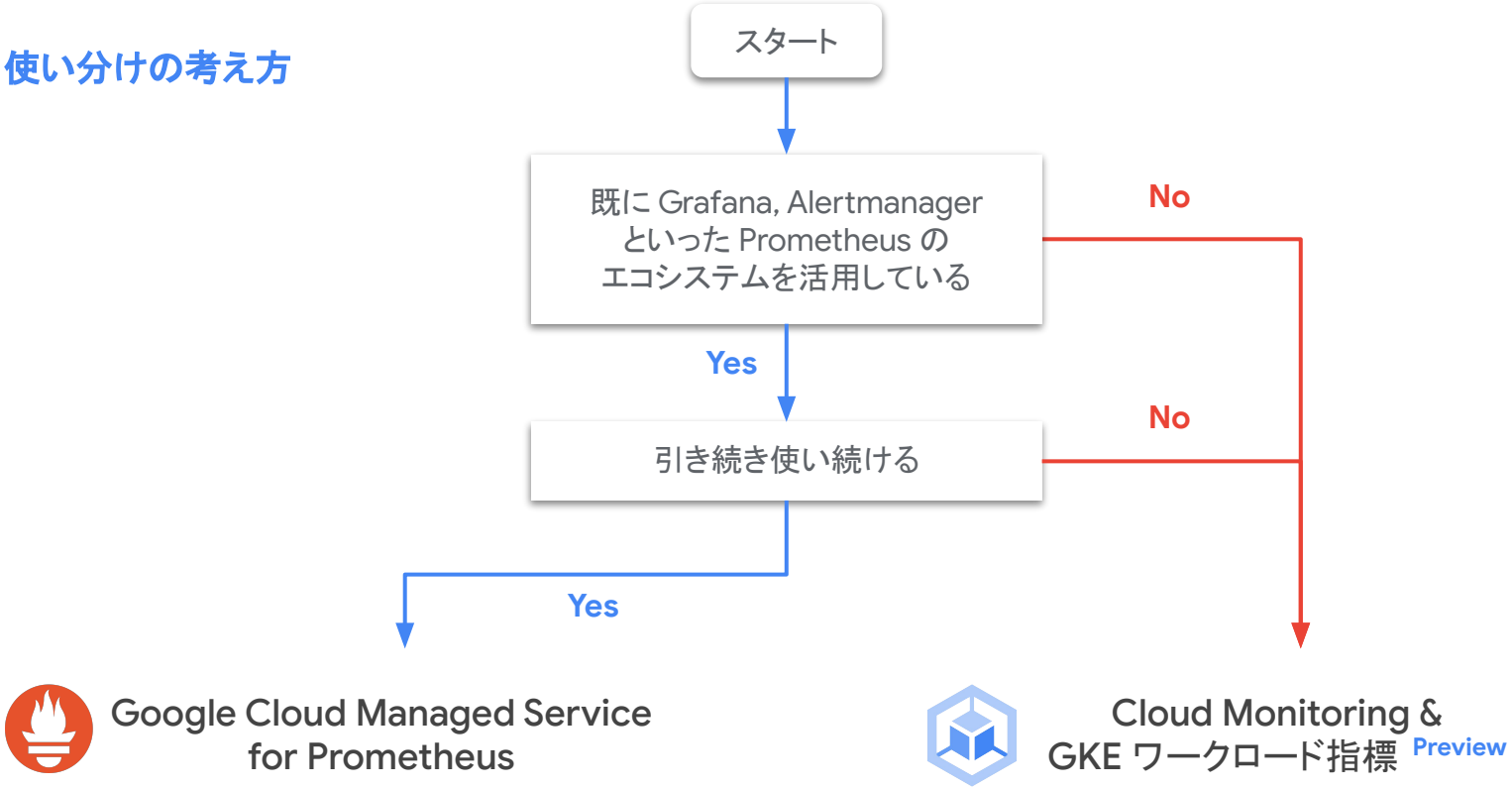
Cloud Monitoring & GKE ワークロード指標 Preview

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



GMP のメリット

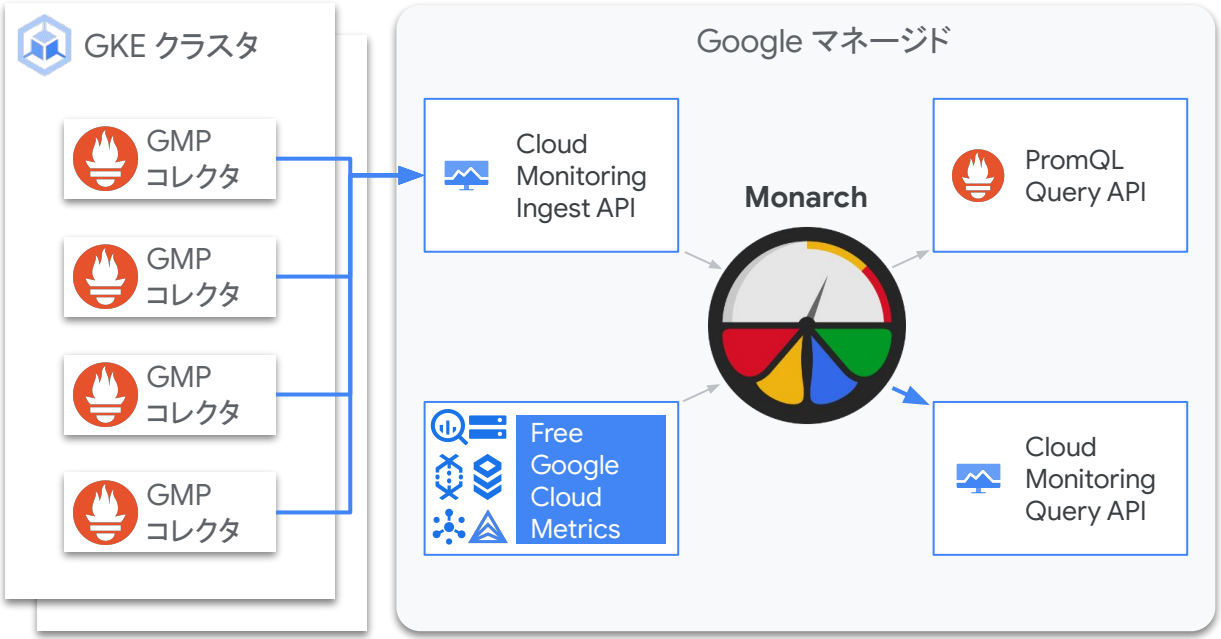
開発者 | インフラ担当者 | セキュリティ担当者

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

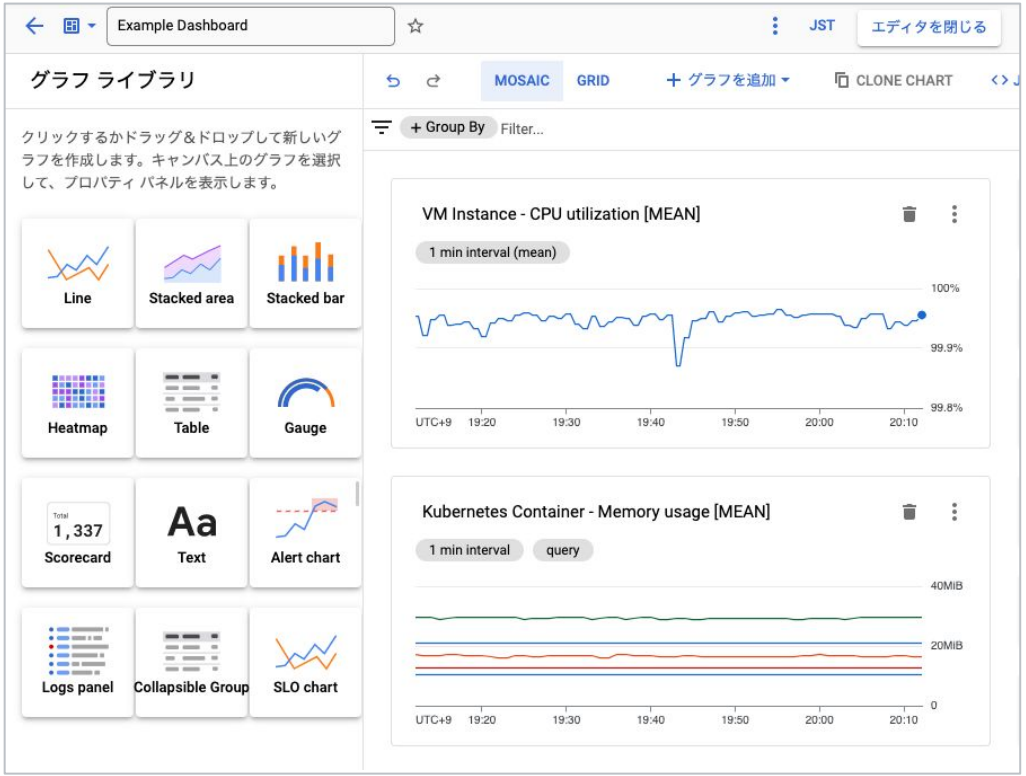


グラフ、ダッシュボードを作成する

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

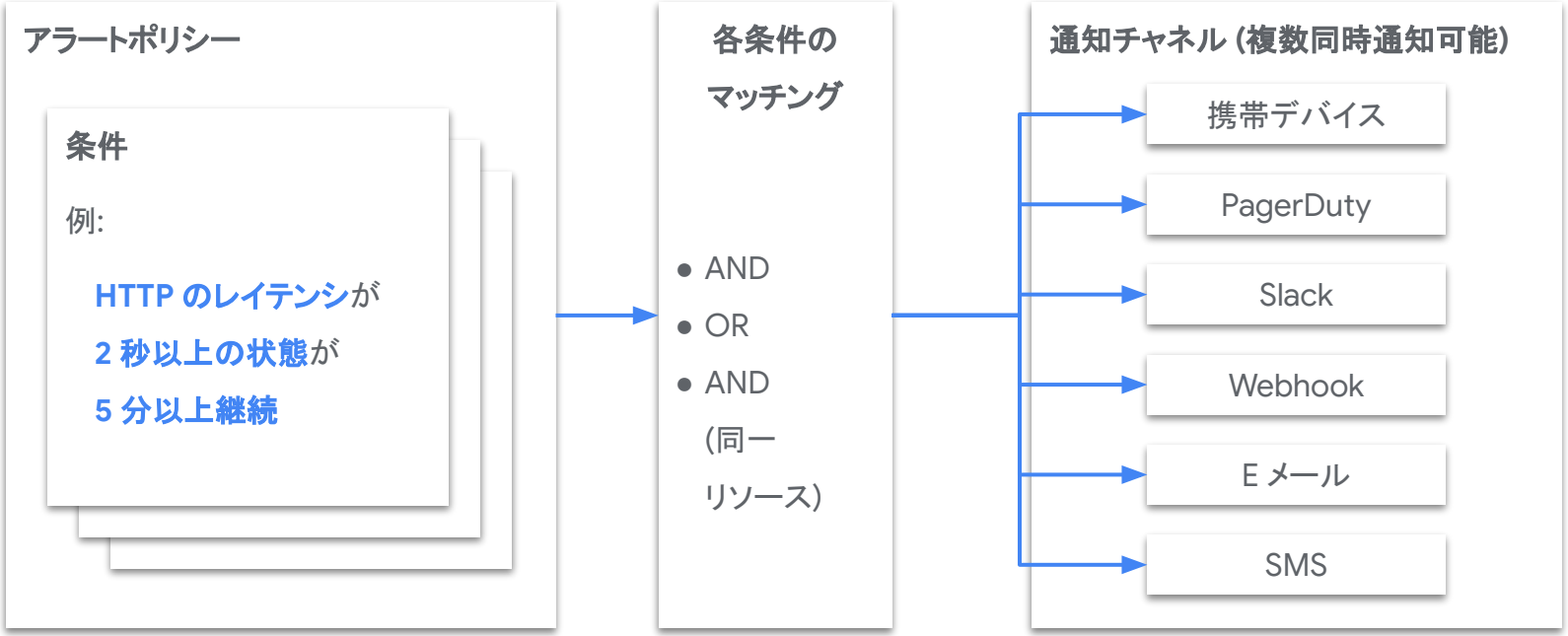
- コピー
- 共有
- 変更、世代管理

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 ●

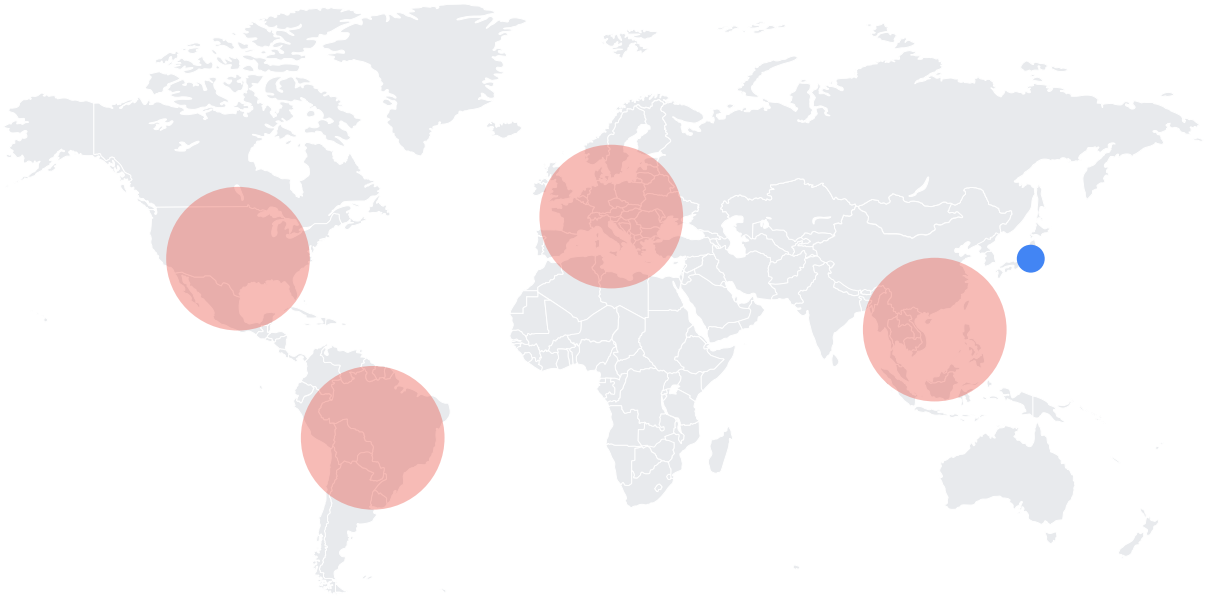
- 公開エンドポイント
- プライベート エンドポイント Preview

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

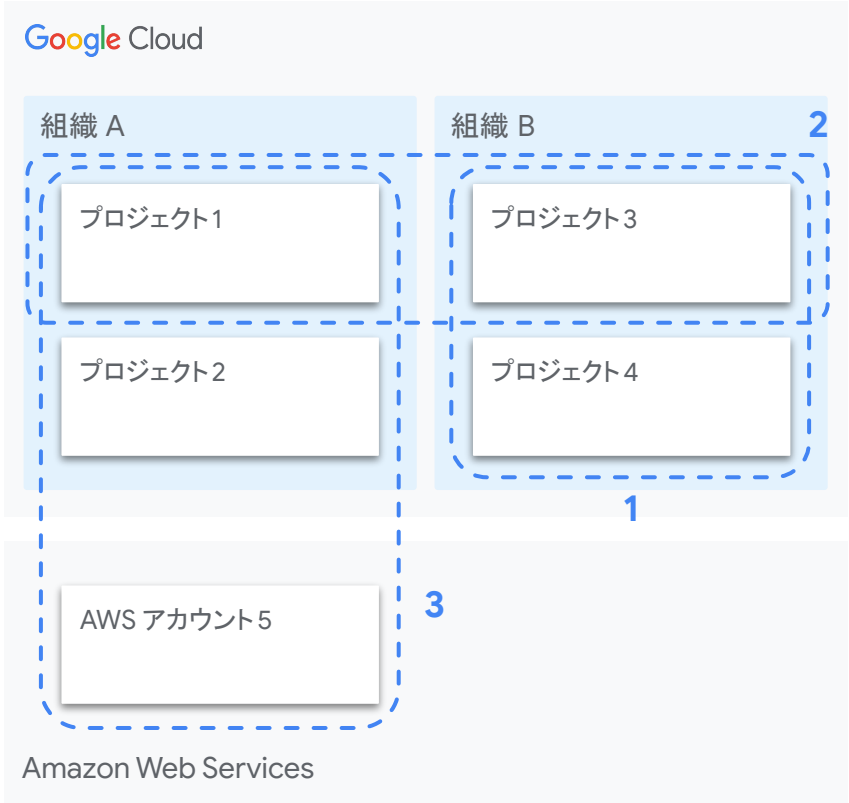
開発者 | インフラ担当者 | セキュリティ担当者

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

04

Cloud Trace

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)

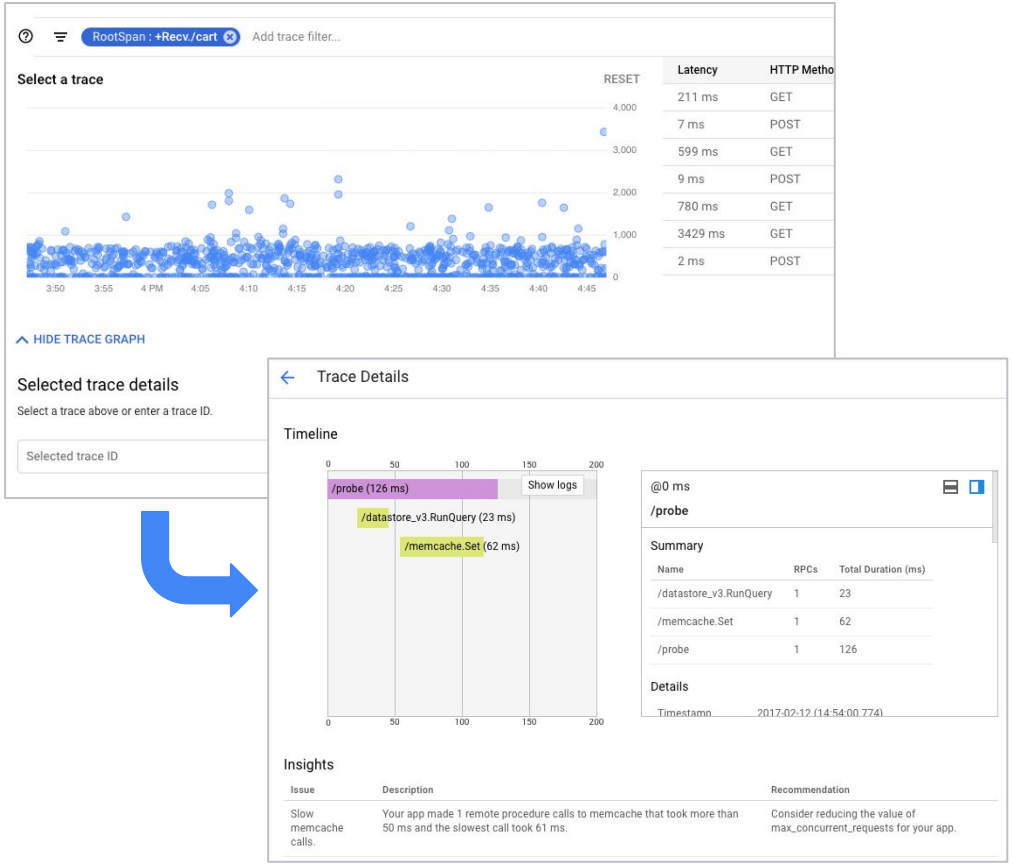
トレースデータをリアルタイムに確認する

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

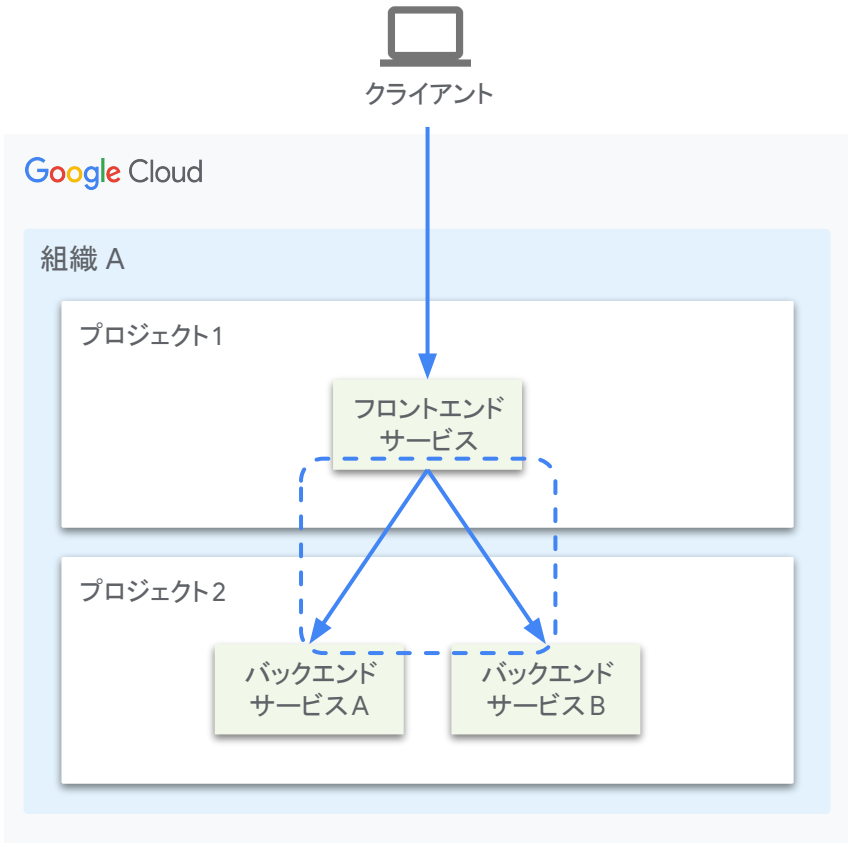


複数プロジェクトのトレースを一元管理

開発者 | インフラ担当者 | セキュリティ担当者

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



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

4 つのゴールデンシグナル

開発者 | インフラ担当者 | セキュリティ担当者

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

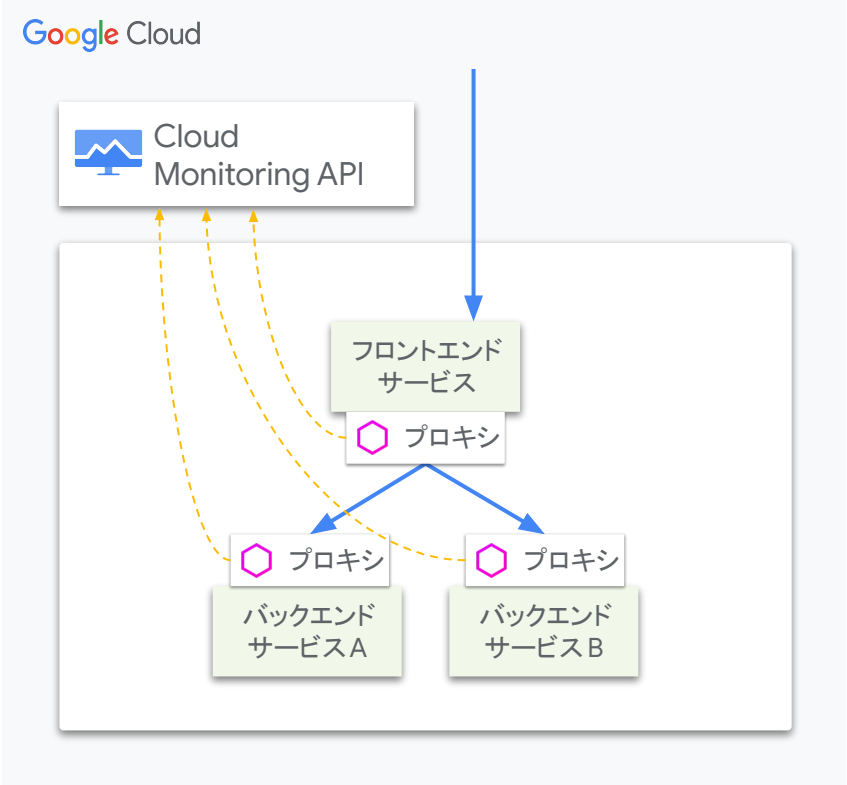
https://sre.google/sre-book/monitoring-distributed-systems/

サービスメ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

06

コンテナ時代に対応するオブザーバ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

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