

Google Cloud

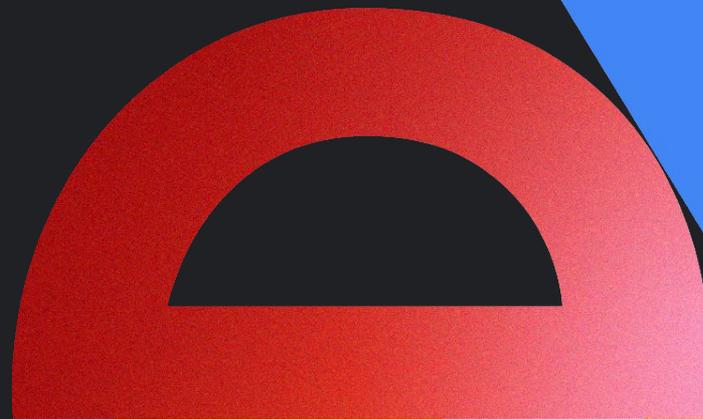
Next

Tokyo

Google Cloud

モニタリング

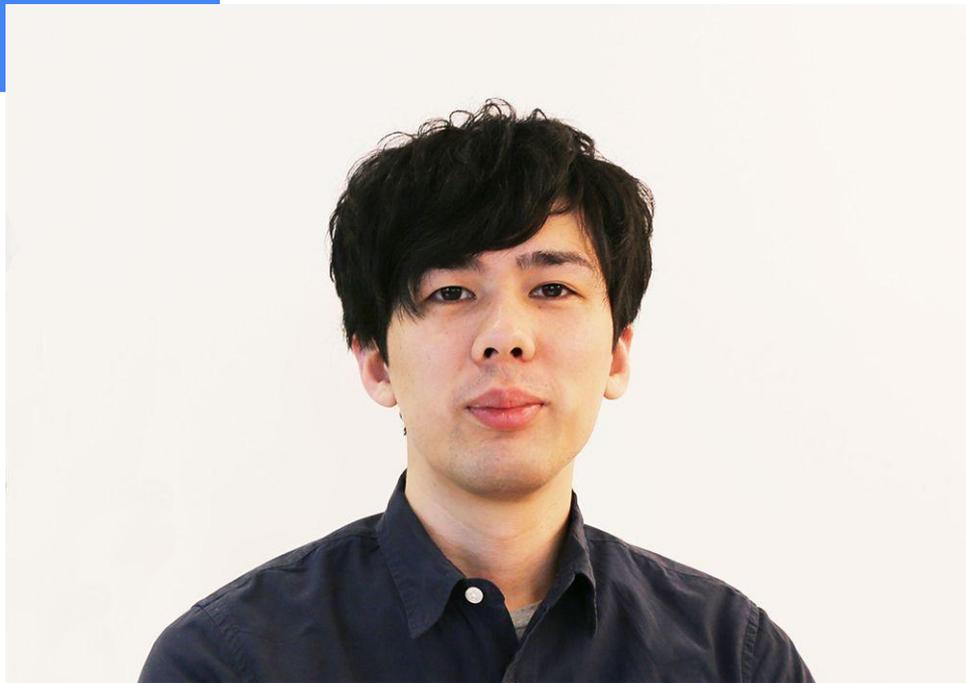
ベストプラクティス



坂田 純

Ubie 株式会社

Head of Platform Engineering



アジェンダ

- 01 Ubie について
- 02 Cloud Monitoring
- 03 GMP
- 04 Ubie での実践



Ubie について

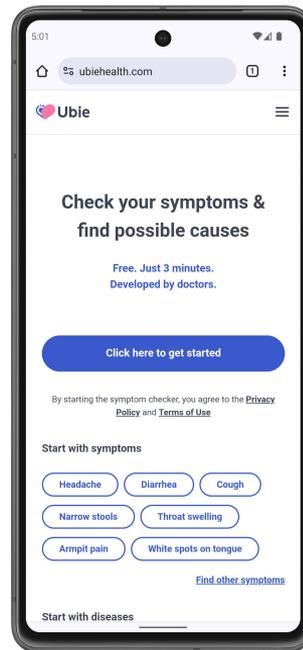


会社について

医師とエンジニアで創業

ヘルステックのスタートアップ

日本とアメリカでサービスを提供





プロダクト(生活者向け)

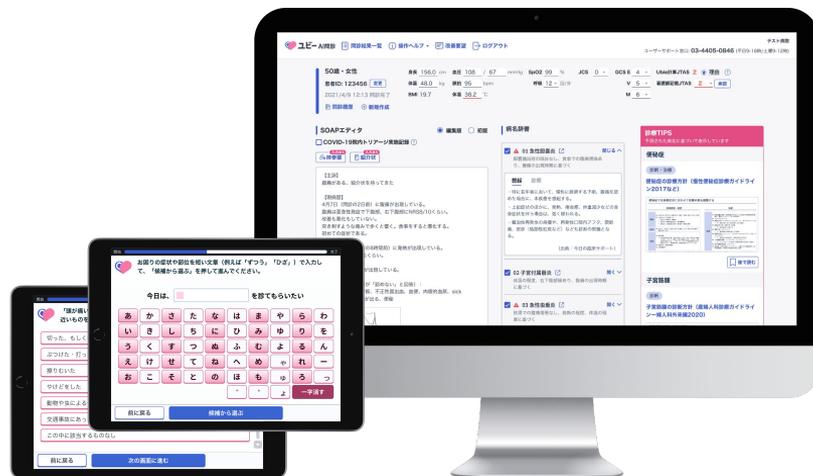
自分の症状を答えるだけで、
参考病名や近くの医療機関等
「受診の手がかり」が調べられます





プロダクト(医療機関向け)

問診業務効率化や認知向上など、患者さんとのコミュニケーション設計を通じ、診察の質向上を支援



Ubie のアーキテクチャ

分散サービス アーキテクチャ

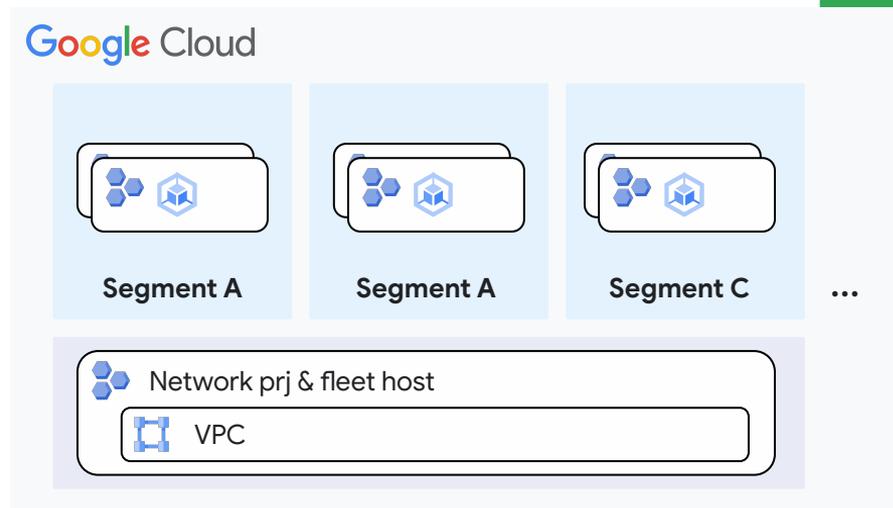
- マイクロサービスとモジュラーモノリスの組み合わせ
- (過度なマイクロサービス化から方針変更)
- ただし大小 50 以上サービスが存在
- 日本とアメリカは独立した環境
- 基本的に Google Kubernetes Engine (GKE) を利用



サービスメッシュ

プロジェクトを細かく分離

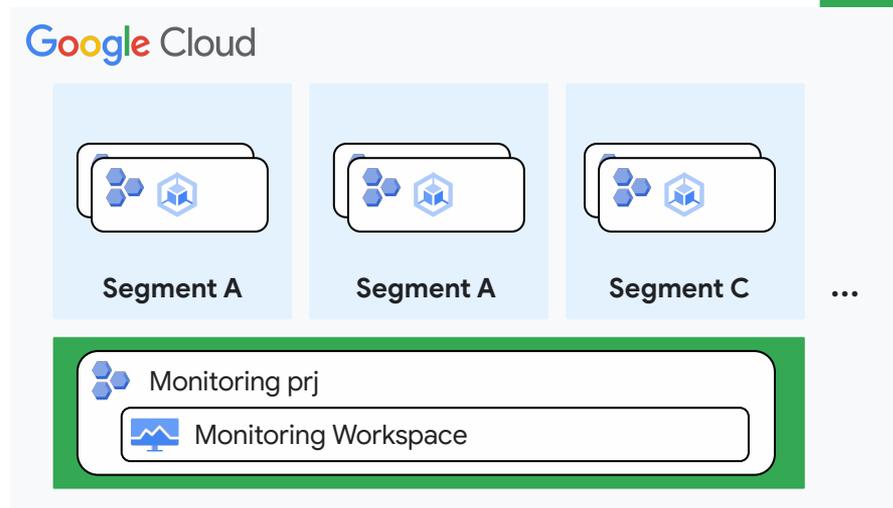
- データの特性やセグメントを定義
- セグメントをプロジェクトレベルで分離
- Shared VPC と Anthos Service Mesh を
利用し一つのメッシュを構成



モニタリング専用プロジェクト

モニタリング専用プロジェクトを用意

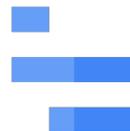
アラートの設定やダッシュボードなどを一元化するため、環境ごとにモニタリングプロジェクトを作りメトリクスを集約



Cloud Monitoring

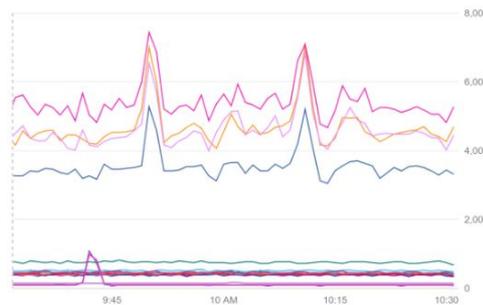
Google Cloud's Observability

- Cloud Monitoring
- Cloud Logging
- Cloud Trace
- Cloud Profiler



Cloud Monitoring

- Google Cloud のネイティブの監視ソリューション
- 各プロダクトのメトリクス、カスタムメトリクスをダッシュボードとして可視化、アラート設定が可能



SLO monitoring

SRE のプラクティスを実践できる環境

しきい値ベースのアラートではなく、SLO やそれをベースとしたアラートの設定も可能

SRE 本にあるようなプラクティスを実践できる

Define SLI details

Refine the metric target for your SLI.

Performance Metric

Certain metrics cannot be used in an SLO. [Learn more](#)

Metric name	Metric kind	Value type
prometheus.googleapis.com/apiserver_request_duration_seconds/histog...	Cumulative	Distribution

Performance metric

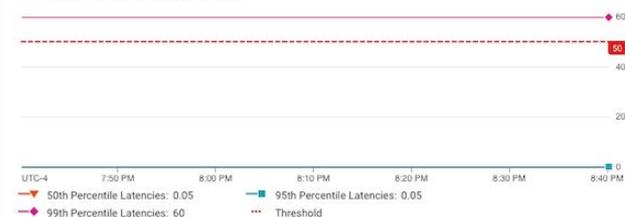
To refine the selected metric, add a resource type and additional filters. You can preview inputs in the chart below.

Range type: Less Than

[Add filter](#) Resource = prometheus_target AND Metric = prometheus.googleapis.com/apiserver_request_duration_seconds/histogram

Preview

Based on current parameters, using historical data



GMP

Google Cloud Managed Service for Prometheus

Prometheus

オープンソースのモニタリング / アラートニングツール

- もともとは Sound Cloud 社から登場
- Kubernetes 同様に CNCF のプロジェクトへ
- 特にクラウドネイティブな環境では**モニタリングのデファクト** となっており、様々なオープンソースのプロジェクトで prometheus が使われている



Prometheus

- 一派的なモニタリングのエージェントと異なり、ワークロードからメトリクスをスクレイプするアーキテクチャを採用
- 各ワークロードは Prometheus の形式でメトリクスを出せば良い



Prometheus の課題

大規模環境で Prometheus を運用する場合、ストレージやスケーリングの問題

ストレージ

長期保存によるストレージの増加やクエリの実行パフォーマンスの最適化を踏まえたストレージの選定

スケーリング

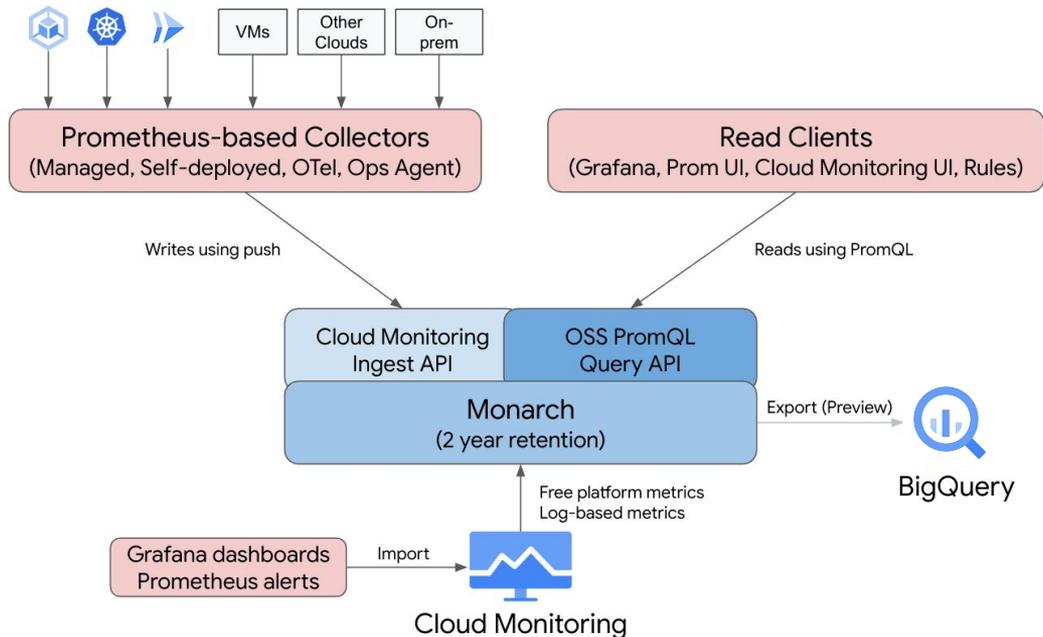
単一の Prometheus 構成ではスクレイピングなどの性能限界などの考慮、メトリクスの集約

Google Cloud Managed Service for Prometheus (GMP)

Cloud Monitoring が直接サポートする Prometheus

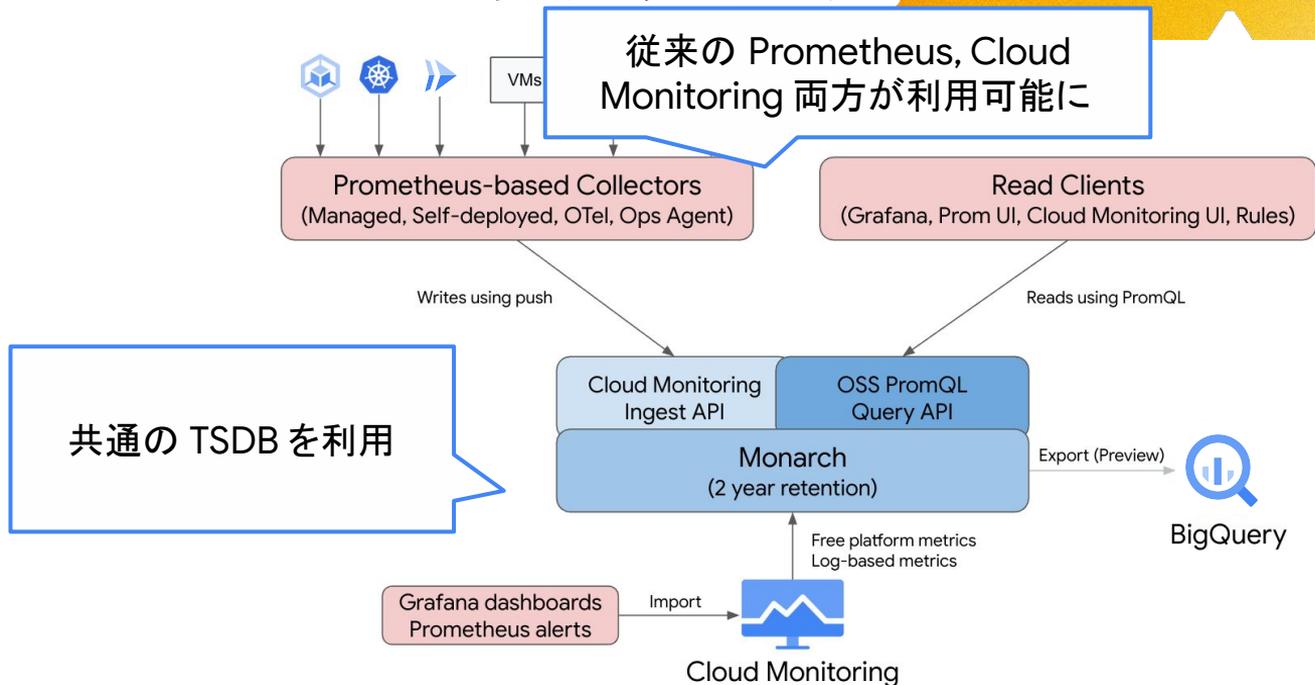
- GMP 以前も Prometheus を含め Cloud Monitoring にメトリクスを投入する方法は存在していたが、Prometheus シリーズ数など制限が多く、実用段階ではなかった
- GMP は Cloud Monitoring でもともと使われている Monarch というバックエンドを活用し、それらの制限が緩和された
- Kubernetes の CRD なども用意しユーザビリティも向上

GMP のアーキテクチャ



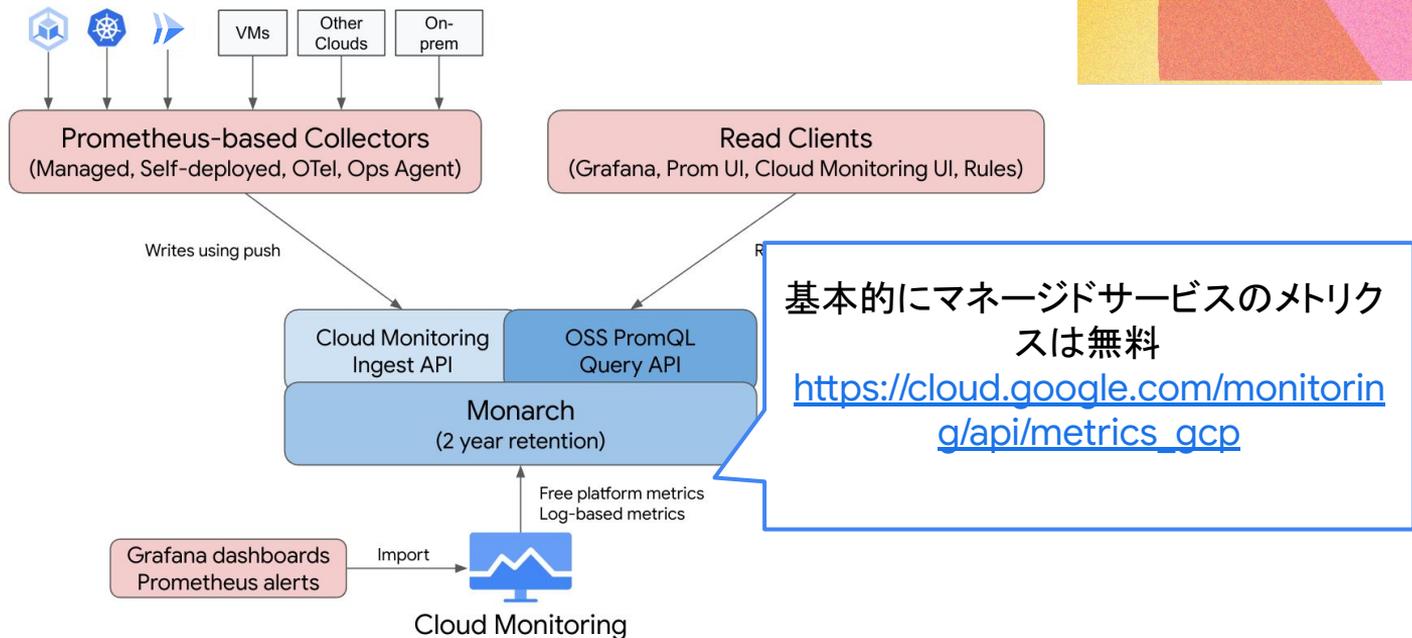
<https://cloud.google.com/stackdriver/docs/managed-prometheus>

GMP のアーキテクチャ



<https://cloud.google.com/stackdriver/docs/managed-prometheus>

GMP のアーキテクチャ



<https://cloud.google.com/stackdriver/docs/managed-prometheus>

[Home](#) > [Publications](#) >

Monarch: Google's Planet-Scale In-Memory Time Series Database

Colin Adams · Luis Alonso · Ben Atkin · John P. Banning · Sumeer Bhola · Rick Buskens · Ming Chen · Xi Chen · Yoo Chung · Qin Jia · Nick Sakharov · George T. Talbot · Adam Jacob Tart · Nick Taylor · VLDB Endowment(2020), 3181–3194

[Download](#)[Google Scholar](#)[Copy Bibtex](#)

Abstract

Monarch is a globally-distributed in-memory time series database system in Google. Monarch runs as a multi-tenant service and is used mostly to monitor the availability, correctness, performance, load, and other aspects of billion-user-scale applications and systems at Google. Every second, the system ingests terabytes of time series data into memory and serves millions of queries. Monarch has a regionalized architecture for reliability and scalability, and global query and configuration planes that integrate the regions into a unified system. On top of its distributed architecture, Monarch has flexible configuration, an expressive relational data model, and powerful queries. This paper describes the structure of the system and the novel mechanisms that achieve a reliable and flexible unified system on a regionalized distributed architecture. We also share important lessons learned from a decade's experience of developing and running Monarch as a service in Google.

GKE の場合

GKE ではオプションを有効化するだけで CRD が **DaemonSet** としてインストールされ簡単にメトリクスの収集が可能

- PodMonitoring
- ClusterPodMonitoring

※ Prometheus Operator に近いが独自規格

```
apiVersion: monitoring.googleapis.com/v1
kind: PodMonitoring
metadata:
  name: my-app
  namespace: my-app
spec:
  endpoints:
    - interval: 30s
      path: /metrics
  selector:
    matchLabels:
      app: my-app
```

Cloud Run も公式にサポート

サイドカーを利用することで Cloud Run のアプリケーションのメトリクスも取得

- もともと OpenTelemetry のサイドカーで取得可能であったが、公式のサイドカーがサポートされた
- <https://cloud.google.com/stackdriver/docs/managed-prometheus/cloudrun-sidecar>

GMP のアーキテクチャ

単純に Prometheus がホストされているわけではない

ほぼ無限にスケールするデータベースとメトリクスの収集機構が提供されている

そのため、本来の Prometheus で気をつける必要があったストレージや冗長性観点では気にする
必要がない

また、カーディナリティについても Prometheus ほどシビアではない

PromQL

Cloud Monitoring では全ての機能で PromQL が利用可能

- もとものの Cloud Monitoring のクエリ言語 MQL に加え、Prometheus の PromQL がすべての機能で利用可能に
- リリース当初はアラートの設定など MQL しか使えないなど制限があった
- PromQL / MQL を混在することも可能だが**今後は PromQL を利用していくと良い**
- 開発者観点でも Kubernetes のようによりオープンな規格を習得することにフォーカスできる

GMP まとめ

GMP は Google が 10 年以上運用してきた地球規模の分散 DB とオープンソースの Prometheus の規格を組みわせ、自在にメトリクスの収集・活用を可能にしている

Ubie での実践

ベストプラクティスを添えて

Ubic のメトリクス

メトリクスは大きく 2 つに分類

- 共通のメトリクス
 - 主にマネージドサービスのメトリクス
- アプリケーションやマイクロサービス固有のメトリクス
 - e.g. ログインの成功率

共通のメトリクス

- マネージドサービス(ASM, GKE, Spanner etc...)
- Prometheus Exporter
 - e.g. Elasticsearch, Fastly
- ログベースメトリクス

GMP の利用

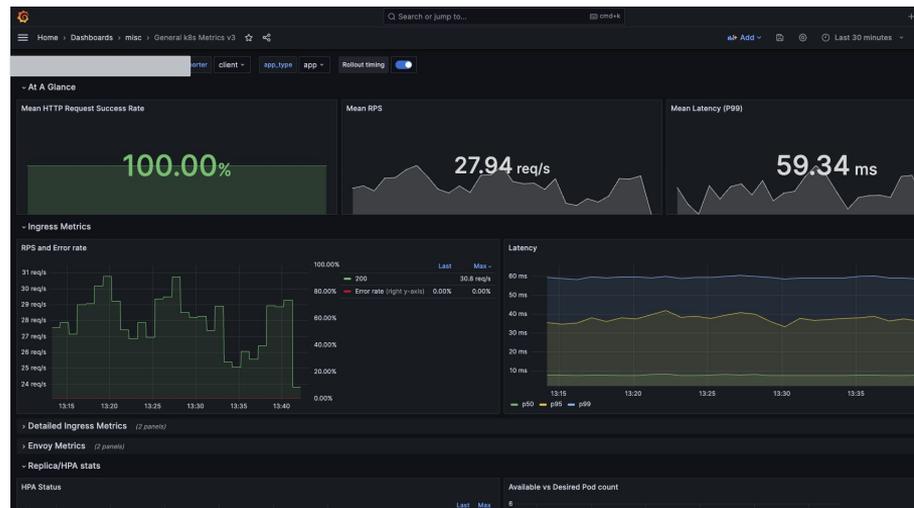
Ubic ではサービスマッシュの導入と合わせて Prometheus を導入

- もともとは Prometheus を自分たちでホスティング
- ストレージの永続化は行わずリアルタイムで確認できるところから開始
- あまりにも不便なので永続化のため Cloud Monitoring / 他社の Prometheus を検証、利用していた
- GMP がリリースされたので移行 (かなりのコスト削減インパクトがあった)

Grafana の利用

全エンジニアが同じダッシュボードを利用

- GMP をソースにメトリクスの提供
- プラットフォームが横断的にトラフィックなどのダッシュボードは提供
- 各開発チームも自由にダッシュボード作成が行える



コスト対策

プロダクトの値下げはあったもののコストコントロールは重要

- CRD で必要なメトリクスのみ取得
- メトリクス種別ごとにアラートの設定

August 08, 2023

CHANGED

The price of Managed Service for Prometheus samples ingested into Cloud Monitoring has dropped by 60 percent. For more information, see [Cloud Monitoring pricing summary](#), and for worked examples, see [Pricing examples based on samples ingested](#).

必要なメトリクスのみ取得

- 特にオープンソースで公開されているPrometheus Exporter は汎用的に作られているため**利用しないメトリクスも取得**してしまう
- **metricRelabeling** を使うとそういったメトリクスを取得しないように落とすことができる



```
apiVersion: monitoring.googleapis.com/v1
kind: PodMonitoring
metadata:
  name: my-app
  namespace: my-app
spec:
  endpoints:
    - interval: 30s
      path: /metrics
      # for cost reduction
      metricRelabeling:
        - action: drop
          sourceLabels: [__name__]
          regex: '(http_.*|jdbc_.*|zipkin_.*)'
  selector:
    matchLabels:
      app: my-app
```

アプリケーション

- 言語やフレームワークなどランタイムのメトリクス
- ビジネスロジックのメトリクス

テンプレートに組み込み

アプリケーションのテンプレートに実装を組み込み

- サービスの立ち上げをスムーズに
- ビジネスロジックに紐づくものは追加実装が必要だが、
言語レベルのものは追加実装不要でメトリクスを出力



Proprietary

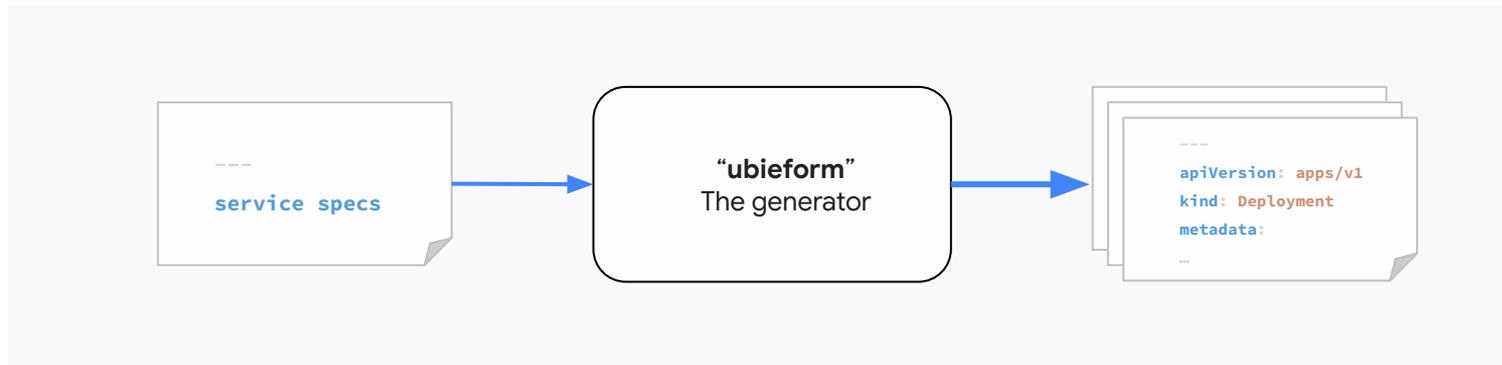
The screenshot shows a GitHub repository page for 'ubie-inc / go-backend-template'. The repository has 18 issues and 3 pull requests. A pull request by 'm-mizutani' is open, titled 'Add gRPC metrics'. The code view shows the following Go code:

```
1 package metrics
2
3 import (
4     grpc_prometheus "github.com/grpc-ecosystem/go-grpc-prometheus"
5     "github.com/m-mizutani/goerr"
6     "github.com/prometheus/client_golang/prometheus"
7     "github.com/prometheus/client_golang/prometheus/collectors"
8 )
9
```

サービスの生成

メトリクス取得の設定もインフラコード生成で自動化

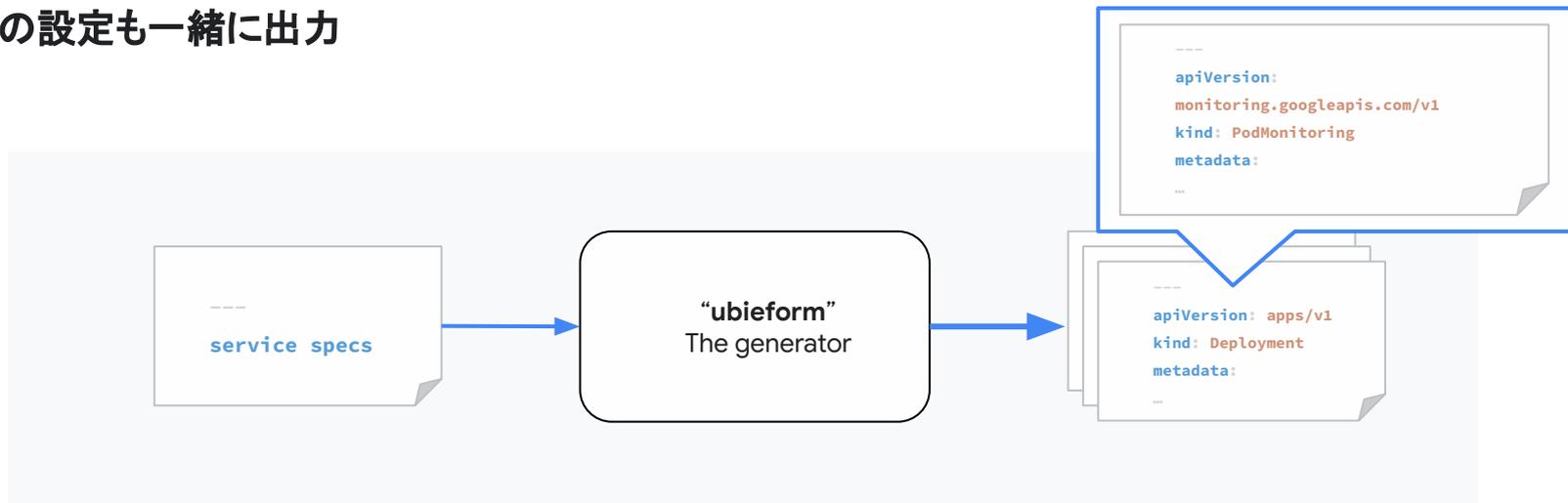
ubiform という社内ツールで Kubernetes のマニフェストを生成



サービスの生成

メトリクス取得の設定もインフラコード生成で自動化

GMP の設定も一緒に出力



サービスカタログの利用

開発者がサービスのメトリクスやログにたどり着けるように

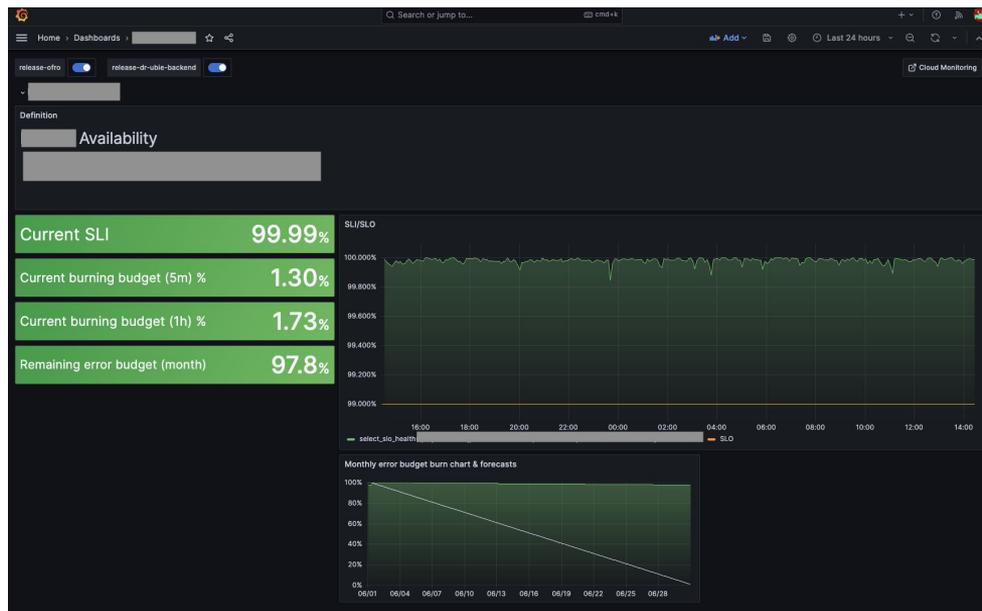
サービスごとの Grafana のダッシュボードや Cloud Logging のメトリクスへのリンクを backstage に自動連携



SLO

SLO Monitoring は Grafana でも利用可能

サービス/プロダクトごとに可視化



Future works

今後の展望

インタラクティブ プレイブック

インタラクティブにトラブルシューティング

見るだけでなくプレイブックとしてダッシュボードが利用可能

e.g. コンテナが起動しないときにどうしたらいいかなど

テキストでの手順の記載も可能

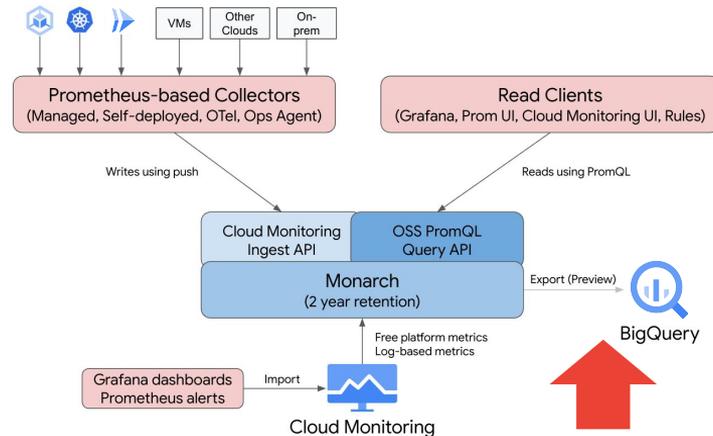
(正直あまり知られていないが結構便利)

The screenshot shows the GKE Interactive Playbook interface for 'Crashlooping Pods'. The interface is titled 'GKE Interactive Playbook - Crashlooping Pods' and includes a navigation bar with 'Cluster: internal (asia-northeast1)', 'Namespace', and 'Node'. The main content area is titled 'View Restarting Containers' and asks 'Which containers are restarting?'. Below this, there is a 'Dashboard' section with a 'Predefined' dropdown and a 'Container Restarts by Workload (Per 5 Minute Intervals)' chart. The chart shows a timeline of container restarts for various workloads, including 'internal config-management-system config-management-operator', 'internal cross-logs-system cross-logs-controller-manager', 'internal cp-internal-dto-docs dto-docs', 'internal cp-internal-elementary elementary', 'internal external-secrets external-secrets', 'internal external-secrets external-secrets-cert-controller', 'internal external-secrets external-secrets-webhook', 'internal gatekeeper-system gatekeeper-audit', 'internal gatekeeper-system gatekeeper-controller-manager', 'internal gke-gcp-system collector', 'internal gke-gcp-system gcp-operator', and 'internal istio-ingressgateway istio-ingressgateway'. Below the chart, there is a 'Next Steps' section with links to 'Investigate possible root causes', 'Jump to Identify Application Errors', 'Jump to Investigate Out Of Memory Issues', 'Jump to Investigate Node Disruptions', and 'Jump to Investigate Liveness Probe Failures'. There is also a section for 'Identify Application Errors' with a 'Dashboard' dropdown and an 'Application Logs' section with a 'Show query' dropdown and a 'Filter' search bar.

Observability Analytics

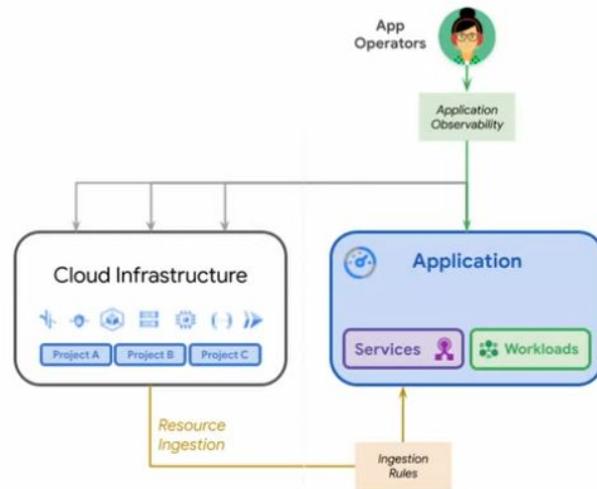
ログ・メトリクスが BigQueryからクエリ可能に

現状は Log Analytics を利用することでログが集計可能
これにメトリクスも加わり、SQL で包括的に Observability
がクエリ可能に



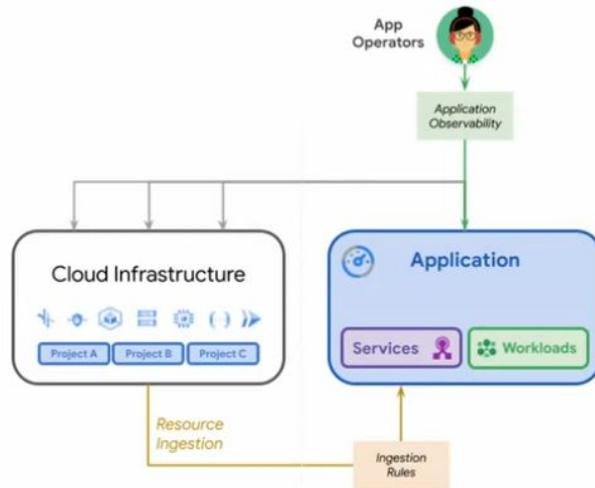
App Hub

- クラウドリソースやインフラではなく、**アプリケーション中心**にオペレーションとガバナンスを強化
- オペレーション面では機能がまだ不十分だがオブザーバビリティも今後期待



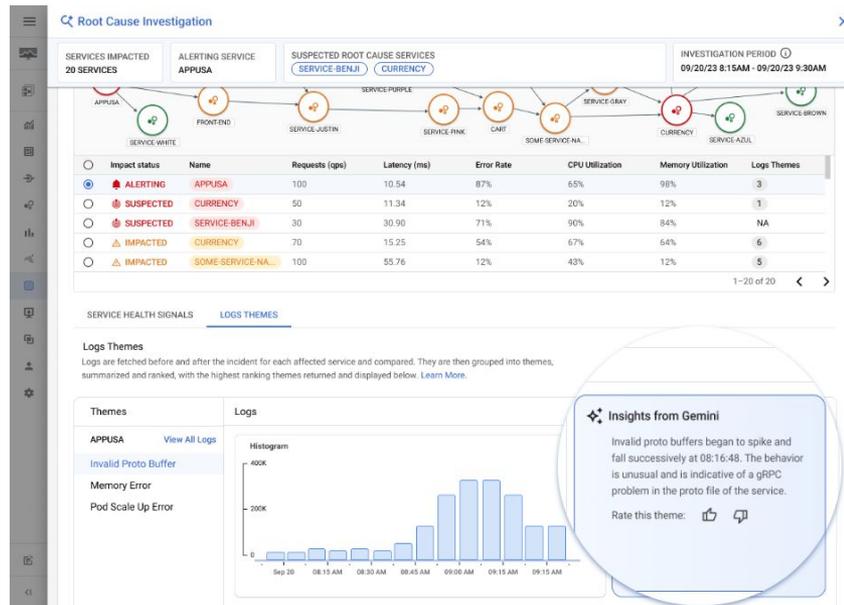
App Hub

現状は GKE, ASM, Cloud Run, Cloud Monitoring などそれぞれのプロダクトでメトリクスの確認や SLO の設定が可能だが、インフラに関わらず開発者が一番に利用する場所として期待
自分たちで backstage を管理しなくていいように



AI: Gemini Cloud Assist

- Next'24 で発表
- Gemini を Cloud Console に統合
- 一般的なことから実際に動いているリソースまで AI と対話が可能
- アラートやトラブルシューティング時に役立つことを期待



Takeaways

~まとめ~

まとめ



Cloud Monitoring

アラートやダッシュボードの作成だけでなく、SLO を使った実践的なモニタリングができる



Advanced Observability

Observability Analytics により SQL を駆使したログやメトリクスの分析も可能に



GMP

Prometheus と Cloud Monitoring 双方のメリットを享受しつつ、エンドユーザー固有のメトリクスを収集・可視化することが可能



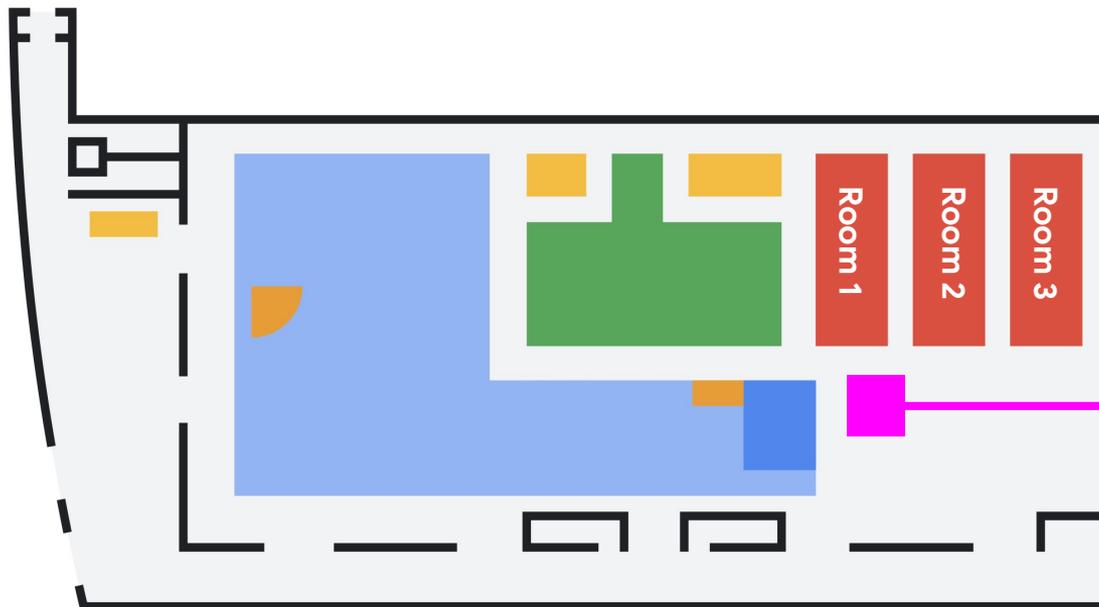
AI、Platform Engineering

Platform Engineering や AI の観点でより開発者が使いやすいプラットフォームへ

Ask the Speaker にぜひお越しください

セッションに関する質問にスピーカーが直接お答えします！

1F



Ask the
Speaker

Thank you

