Google Cloud

Cloud Run と周辺システムで、アプリの内製化を加速させていくために

Google Cloud
アプリケーション モダナイゼーション スペシャリスト
賴兼 孝幸

Table of Contents

01

内製化をはじめる際に気をつけるポイント

内製化のトレンド

“

ユーザー企業が技術的負債の解消と本格的な DX の実現に向けて、協調領域の**クラウド・共通プラットフォーム活用や競争領域の内製化**を進めることなどにより、企業のバリュー アップに資する IT 投資や経営の俊敏さが向上し、ユーザー企業で活躍する IT 人材が増加することが期待される

出典：我が国における IT 人材の動向 - 経済産業省
https://www.meti.go.jp/press/2020/12/20201228004/20201228004-2.pdf

SRE（サイト信頼性エンジニアリング）プラクティスの実践

1. **小さく始めて、繰り返す**

   関連アプリやチームを洗い出し、概念実証から始め、失敗から学び、そして反復する

2. チームを支援する

   社内のスキルアップ、チームの新規採用、拡充といったイネーブルメント戦略を検討

3. 学んだことをスケールする

   少数のチームで SRE プラクティスが確立されたら、SRE コミュニティと組織全体で形式化されたプロセスを作る

4. **データドリブン型のマインドセットを具体化する**

   データドリブン型を目指すため、組織に測定カルチャーを作ることがきわめて重要

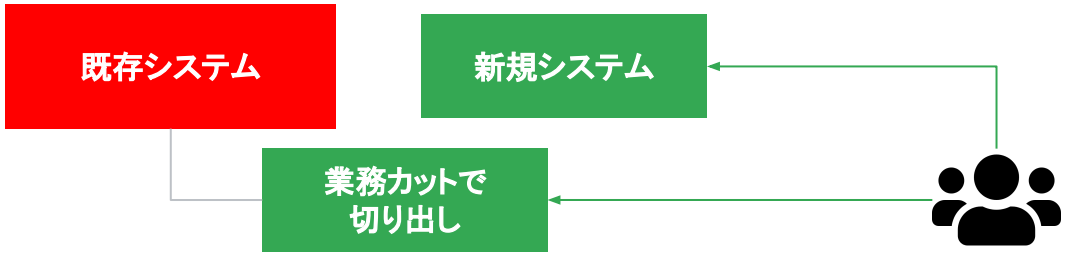
参考 : Google Cloud Blog, “SRE プラクティスを促進させるための 4 つのステップ”,
https://cloud.google.com/blog/ja/products/devops-sre/four-steps-to-jumpstarting-your-sre-practice

運用体制に目をむける

既存システムが大きい場合、**内製化のスコープを決めて実施する（小さく始める）**とリスクが軽減

業務にあった運用体制をベースにマイクロサービス化させ、内製化のメリットを考える
（内製化やマイクロサービス化すること自体を目的化させない）

新規システムであれば、**あるべき姿として、新しくアーキテクチャ設計する**ことも大事



内製化による業務改善が期待できる

運用可能な内製化を実現するために

- 自社リソースの体制では、開発と運用が難しい

  ⇒ **業務ベースで考え、内製化メリットのある範囲から小さく始める**

- 実装や構築の専門スキル不足

  ⇒ **マネージド サービスを有効活用し、学習コストを最小限にする**

- ビジネスにフォーカスしたい。非ビジネス要件は外注したい

  ⇒ **サーバーレス環境を利用し、フォーカスすべきでない範囲の工数を省く**

  （外注して、人でカバーしていた範囲を、技術で置き換える ）

アプリケーションの
モダナイズ

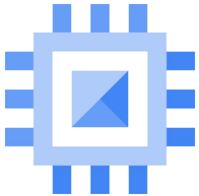
Cloud の活用

02

Cloud Run の活用

Compute プロダクト一覧

- Google Cloud では計 5 つの Compute プロダクトを用意している



Compute Engine

仮想マシン

Google Kubernetes Engine

フルマネージド Kubernetes

Cloud Run

サーバーレス コンテナ実行環境

App Engine

サーバーレス アプリケーション 実行環境

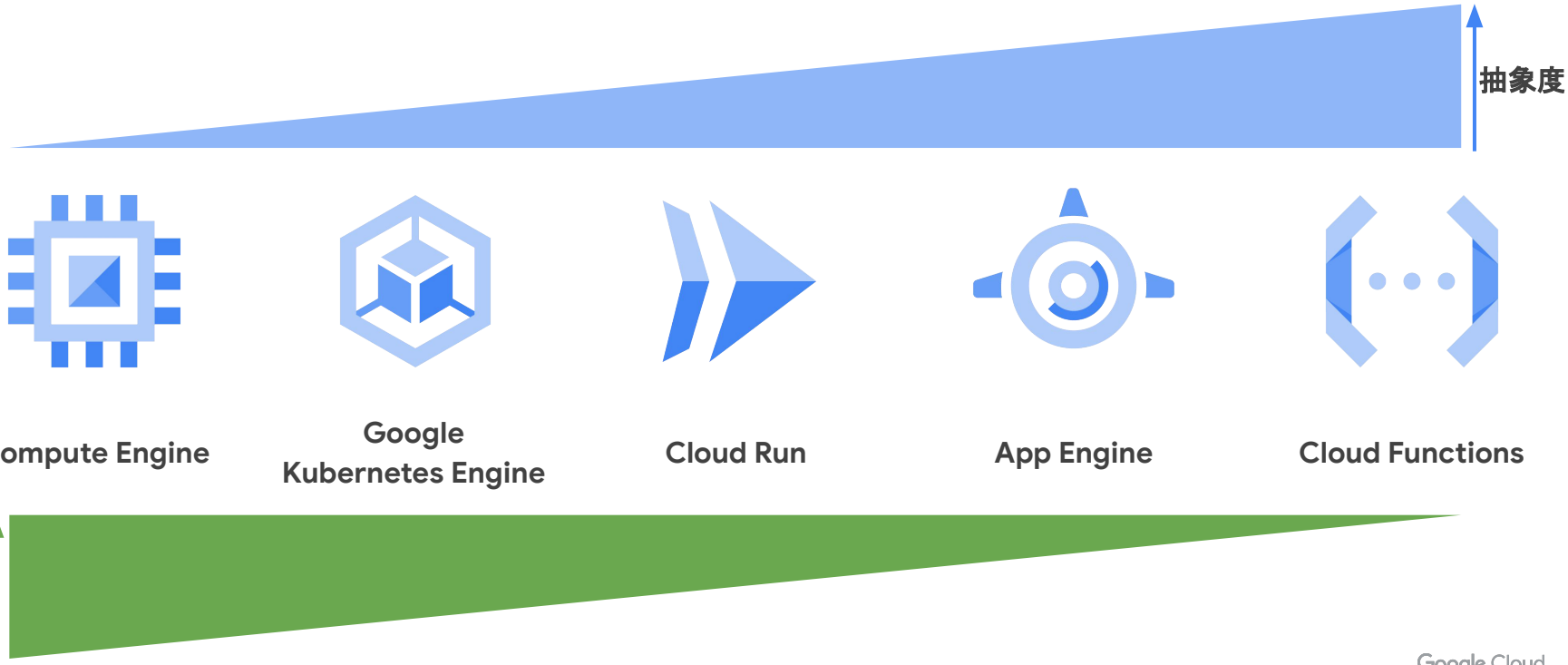
Cloud Functions

サーバーレス Functions as a Service

Compute プロダクト一覧

- 抽象度と柔軟性のトレードオフ

サーバーレス環境で小さくはじめる



Cloud Run について

サーバーレスのコンテナ実行環境

- **コンテナをデプロイするだけで** 外部から到達可能な **URL が発行**される
- 0 ～ N へトラフィックに応じて **高速にスケーリング**
- Eventarc と連携し**イベント駆動**で処理を実行
- HTTP/2、WebSocket、gRPC への対応
- **高度なトラフィック管理** が統合

Cloud Run Jobs **New**

- HTTP リクエストに依らない実行
- 分散タスクの実行で、より長時間の実行が可能
- 明示的な並列処理や、リトライ回数を定義

Eventarc

gRPC

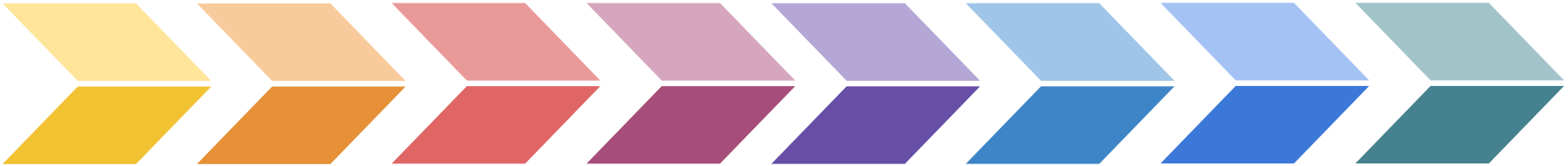
サーバーレス製品における、Cloud Run の優位性

インフラ管理不要で、制約も少なく、スケール性能が高い

- 言語やライブラリの制約なし
- WebSocket や gRPC サポート
- App Engine フレキシブル環境よりもスケール性能が高い
- マルチリージョンのサービス展開
- イベント駆動
- コンテナ化や CI / CD サポート（Buildpacks、ソースコードからのデプロイ）

新しい機能も頻繁にリリースされている

アプリをコンテナ化、デプロイ、ログやメトリクスの監視



Build	タグ	Push	テスト	デプロイ	変更監視	ログ監視	メトリクス監視
成果物としてコンテナイメージをビルド	各成果物が特定できるようタグで管理	レジストリへイメージをプッシュ	作られたイメージが期待通りであることを検証	クラスタへデプロイし期待通り動作することを確認	ソースファイルが変更されたらビルドして再デプロイ	アプリケーションが出力するログをリアルタイム監視	可用性や、レイテンシなどのメトリクスを継続的に監視、アラート

ローカル環境におけるコンテナ開発

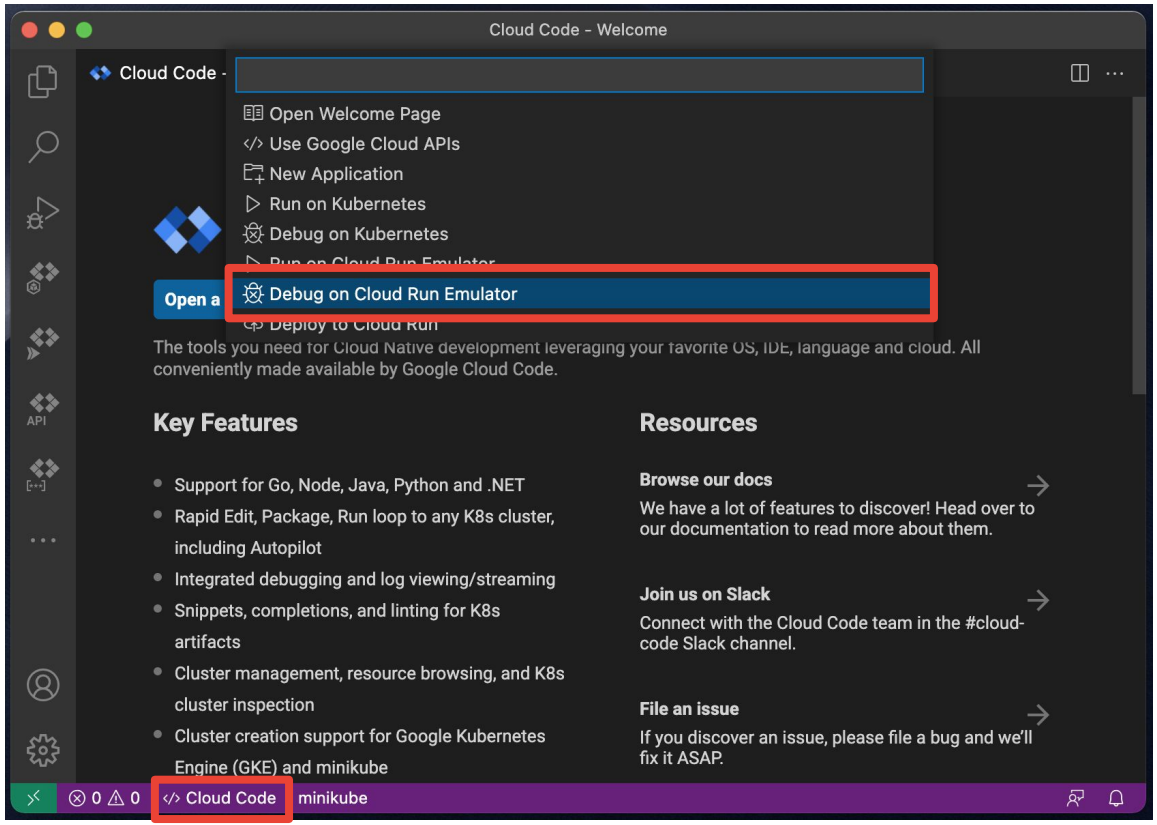
Cloud Code の利用

コンテナ、特に Kubernetes / Cloud Run ベースの
アプリケーション開発、運用を支援する
VS Code、IntelliJ、Cloud Shell エディタ**対応のプラグイン**

- ✓ **開発**: ビルド、起動、ポートフォワード、ファイル変更監視、デバッグ、ログ出力..
- ✓ **運用**: デプロイ、状況確認、ターミナル操作..

Cloud Code | Google Cloud
https://cloud.google.com/code?hl=ja

Debug on Cloud Run Emulator



1. minikube が起動して
2. アプリがビルドされ
3. **Cloud Run の設定に基づいて**
   デプロイされ
4. サービスの待機ポートが
   localhost にフォワードされ
5. ログも連携され
6. デバッガがアタッチされ
7. ファイル変更が監視されて
   リビルド & 再デプロイ

ビルドして、コンテナイメージをデプロイ

一般的なコンテナ イメージのデプロイ手順

1. Dockerfile を作成
2. ソースコードに合わせて、インストールなどの手順を Dockerfile に定義
3. CI パイプラインを記述（Cloud Build、Jenkins、etc...）
    a. docker build コマンドで、コンテナ イメージを作成、タグ付け
    b. docker push コマンドで、コンテナ イメージをレジストリに登録
4. gcloud run deploy コマンドや、terraform などで該当イメージを Cloud Run へデプロイ

Buildpacks でコンテナイメージを楽に作成する方法も

指定したディレクトリ以下を解析し、

Dockerfile なしに適切なコンテナへビルドします

サポート言語

- Go 1.10 +
- Node.js 10 +
- Python 3.7 +
- Java 8 +
- .NET Core 3.1 +
- Ruby 2.6 +

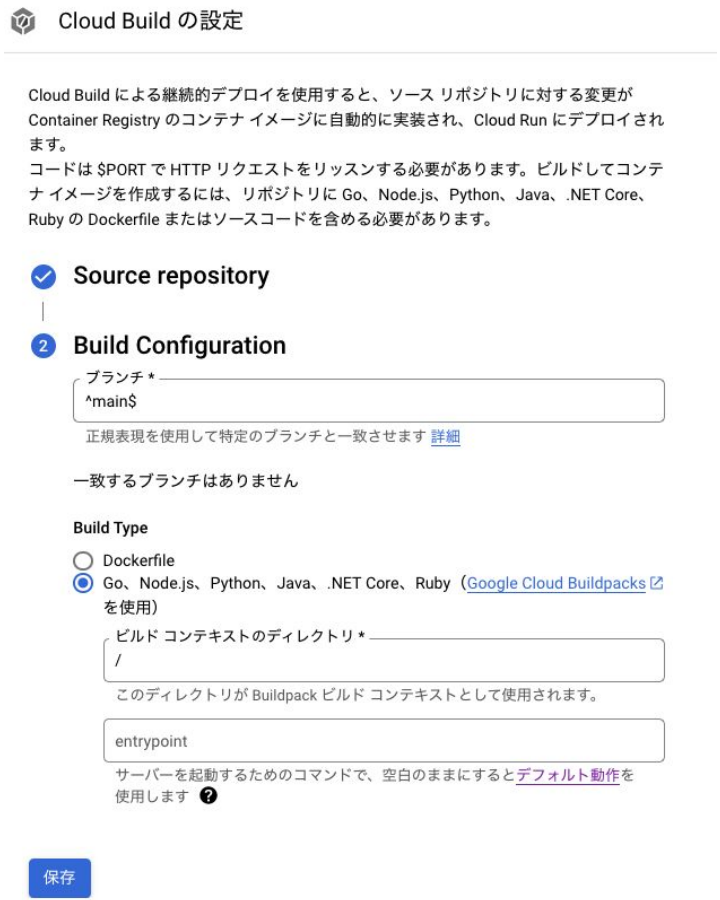
```
$ ls
index.js  package.json

$ gcloud builds submit \
    --pack image=gcr.io/my-project/my-app
```

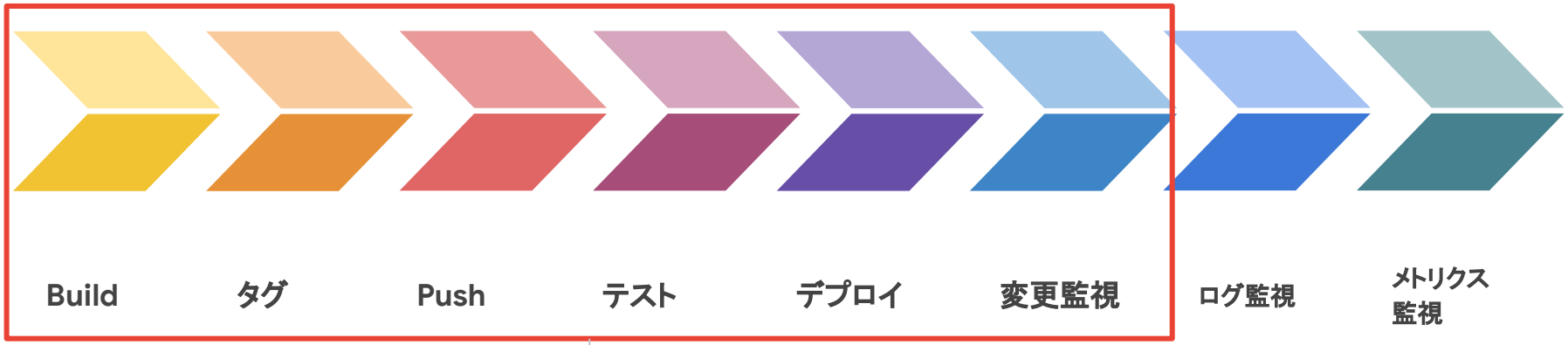
“Google Cloud Platform/buildpacks: Builders and buildpacks designed to run on Google Cloud's container platforms”
https://github.com/GoogleCloudPlatform/buildpacks

Cloud Run のコンソール画面から、Cloud Build を利用した CI/CD と統合可能

- 解析するリポジトリ、ブランチ、ディレクトリを指定するだけ（GitHub なども可）
- Cloud Build で CI pipeline を定義する yaml ファイルが自動的にトリガーとして設定される
- 設定した後は、Git commit / push するだけで、最新ソースが Cloud Run に継続的にデプロイされるように



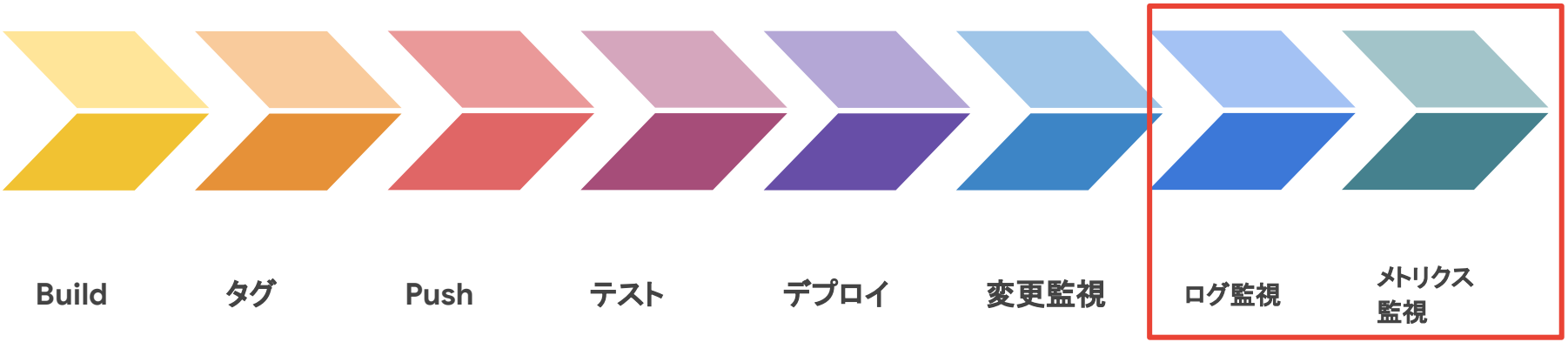
簡潔にデプロイする CI/CD パイプラインをすぐに構築



Cloud Run のコンソール画面で入力するだけで、
ここまでをまとめて設定！（テストは除く）

テストなど、CI パイプラインをカスタマイズしたい場合は、
作成された yaml ファイルを変更していけば良い

アプリをコンテナ化、デプロイ、監視などの運用



Cloud Run には、**Cloud Logging や Cloud Monitoring など、運用に必要なプロダクトが統合** されている

SRE の思想に基づいた **SLO モニタリング**も可能

Cloud Logging の統合

コンテナアプリから、標準出力でログを出力するだけで、Cloud Logging 上でログが確認可能

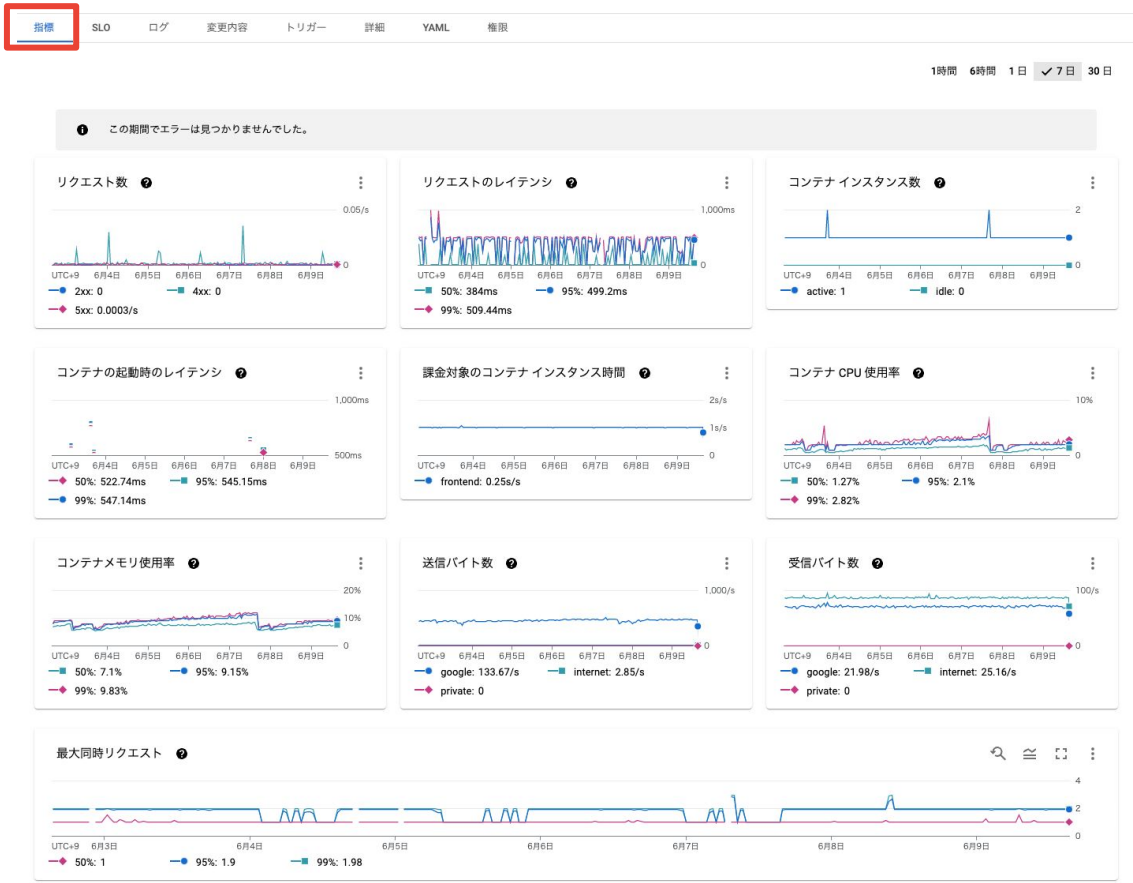
Cloud Run のコンソール画面にも統合されて、該当サービスのログが自動的にフィルタされる

Cloud Monitoring の統合

自動的に必要なメトリクスを収集

Cloud Run のコンソール画面で、
該当サービスのメトリクスを一覧で表示可能

アラートも Cloud Monitoring の
画面から設定可能



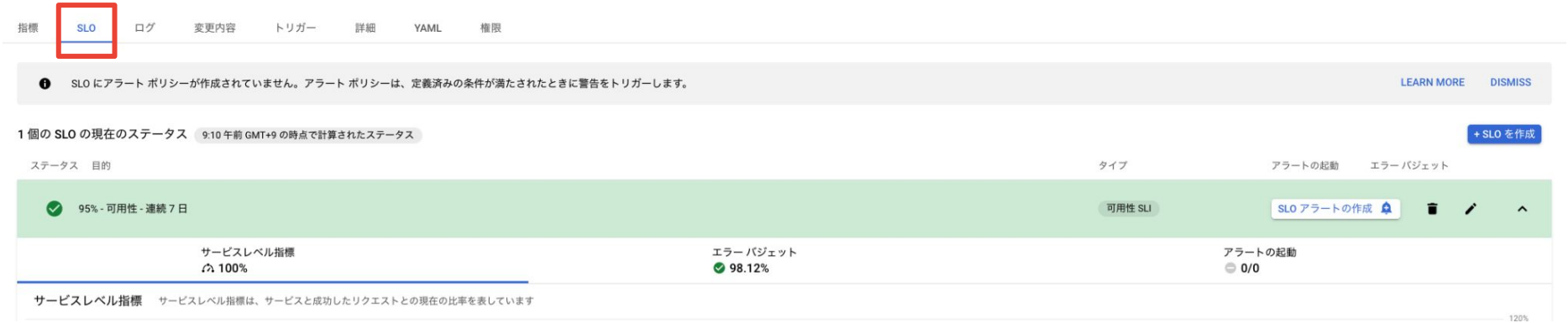
Cloud Run SLO モニタリング

SLO モニタリングとは

- 顧客の目に見える行動を監視する
- ユーザーに対しての保証の検証

エラー バジェットによる、速度と信頼性のバランスをとった開発

ユーザーに対しての保証が破られる方向にあるときのみ、警告アラートを出す



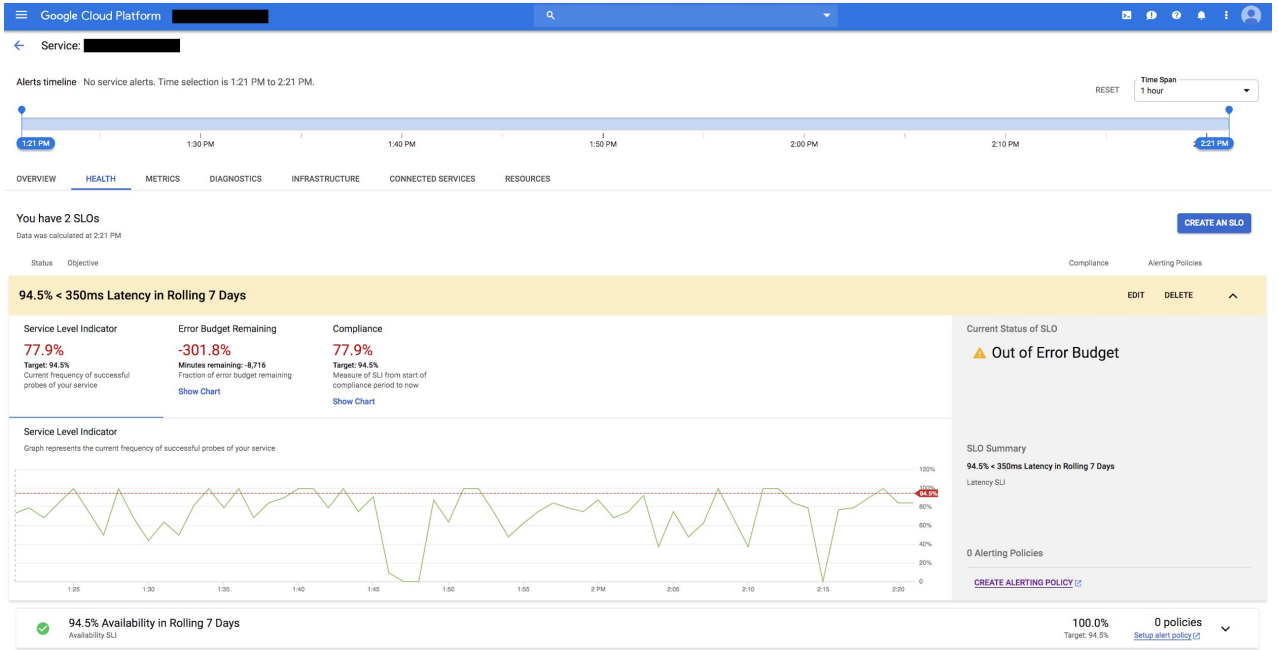
SLO モニタリングとは

顧客の目に見える行動を監視する

ユーザーに対しての保証の検証

エラー バジェットによる、速度と信頼性のバランスをとった開発

ユーザーに対しての保証が破られる方向にあるときのみ、警告アラートを出す



エラー バジェット

信頼性レベル	許容可能な不具合の期間		
	年間	四半期間	30 日間
90%	36.5 日	9 日	3 日
95%	18.25 日	4.5 日	1.5 日
99%	3.65 日	21.6 時間	7.2 時間
99.5%	1.83 日	10.8 時間	3.6 時間
99.9%	8.76 時間	2.16 時間	43.2 分
99.95%	4.38 時間	1.08 時間	21.6 分
99.99%	52.6 分	12.96 分	4.32 分
99.999%	5.26 分	1.30 分	25.9 秒

エラー バジェット
= 1 - SLO

ここで、信頼性を
許容できる不具合の量
と捉えると・・

Canary リリースを利用して、エラー バジェットを節約

信頼性レベル	許容可能な不具合の期間		
	年間	四半期間	30 日間
90%	36.5 日	9 日	3 日
95%	18.25 日	4.5 日	1.5 日
99%	3.65 日	21.6 時間	7.2 時間
99.5%	1.83 日	10.8 時間	3.6 時間
99.9%	8.76 時間	2.16 時間	43.2 分
99.95%	4.38 時間	1.08 時間	21.6 分
99.99%	52.6 分	12.96 分	4.32 分
99.999%	5.26 分	1.30 分	25.9 秒

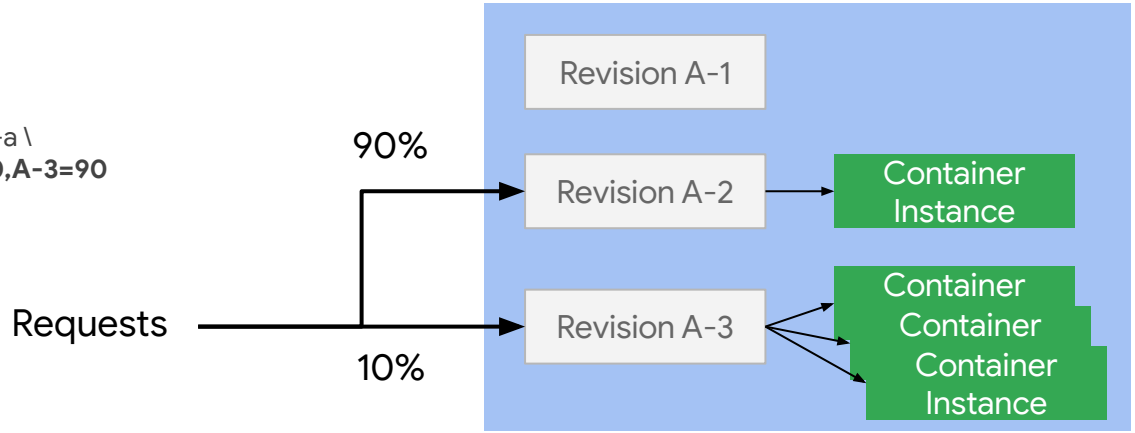
開発者と運用者で
同じ指標を目標にできる

エラー率	許容可能期間
100%	21.6 分
10%	3.6 時間
1%	36 時間
0.1%	15 日
<0.05%	ずっと

トラフィックを制御したデプロイ戦略

Update traffic 機能を使うことで、Blue / Green Deployment や Canary release、
ロールバックなどが簡単に可能に

```
$ gcloud run services \
update-traffic service-a \
--to-revisions=A-2=10,A-3=90
```



03

デモ

04

まとめ

Cloud Run を活用して内製化の懸念を払拭

- 自社リソースの体制では、開発と運用が難しい
  - ⇒ **業務ベースで考え、内製化メリットのある範囲から小さく始める**
  - ⇒ **切り出した業務範囲を Cloud Run のサービスで構築**
- 実装や構築の専門スキル不足
  - ⇒ **マネージド サービスを有効活用し、学習コストを最小限にする**
  - ⇒ **インフラ構築不要。CI/CD、運用まで統合機能でカバー**
- ビジネスにフォーカスしたい。非ビジネス要件は外注したい
  - ⇒ **サーバーレス環境を利用し、フォーカスすべきでない範囲の工数を省く**
  - ⇒ **ビジネス ロジックが入るアプリケーションにフォーカスできる**

Tech Acceleration Program

アプリケーション開発支援プログラム

Prototype

開発予定アプリケーションの
プロトタイプ開発の
ご支援

Architecture

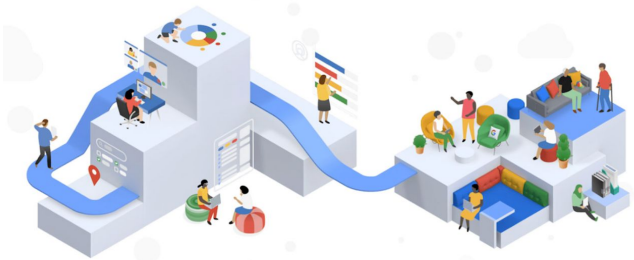
開発予定アプリケーションの
最適なアーキテクチャ設計の
ご支援

Security

クラウド開発における
セキュリティルール策定の
ご支援

内製化支援パートナー

内製化支援で DX を加速



ビジネスアセスメント	組織・体制作り支援	テーマ選定	技術アセスメント	開発支援	活用・運用支援
課題発掘から開発プロセスへの繋がりをコンサル	アジャイルな開発組織構築支援	どのようなユニットで何を開発するのか	ユーザー側の技術スキルレベルのアセスメント	PoC、Pilot 開発 DevOps との連携支援 ex) GitLab	定期的な活用支援

	ビジネスアセスメント	組織・体制作り支援	テーマ選定	技術アセスメント	開発支援	活用・運用支援
iret	○	○	○	○	○	○
CTC	○	○	○	○		
UL Systems, Inc.	○	○	○			
Cloud Ace	○	○	○	○	○	○
CREATIONLINE, INC.		○			○	
GI Cloud		○			○	○
Sreake					○	○
Centillion System		○		○	○	
DIGITAL VORN			○	○	○	○

Google Cloud パートナーが提供する内製化支援サービスにご興味のある方はこちらまで
https://goo.gle/naiseika-partners