3SHAKE

実践 SRE

株式会社スリーシェイク
手塚 卓也

自己紹介

3-SHAKE

手塚 卓也

自治体やデータベースマーケティング会社でのインフラ設計 /構築/運用を主に経験し、2018 年 10 月に 3-Shake に Join。

3-Shake Join 後は Google Cloud / AWS / kubernetes / ServiceMesh など様々な技術的アプローチを駆使し、
大手からベンチャー等規模を問わず様々な組織に対して SREのコンサルティングや実践を行っている。

趣味はボクシング観戦
※ 日曜日の昼間は一人で WOWOW 見ながら一人で熱狂してます ...

アジェンダ

1. About 3-shake
2. SRE 実践のためのアプローチ
3. Google Cloud での SRE の実践
4. まとめ

1. About 3-Shake

About 3-Shake

3-SHAKE

Sreake

SRE 支援 / 技術支援サービス

Sreake

Sreake は金融・医療・動画配信・AI・ゲームなど技術力が求められる領域で豊富な経験を持つ SRE が集まったチームによる技術支援サービスです。戦略策定から設計・構築・運用、SaaS 提供まで、幅広い領域をサポートします。

Sreake Security

セキュリティ脆弱性診断サービス

Sreake Security

Sreake Security は経験豊富なセキュリティ専門家が貴社の課題に合わせたセキュリティ対策を支援するサービスです。

Reckoner

クラウド型 ETL / データ パイプライン サービス

Reckoner

Reckoner はクラウド型 ETL / データパイプラインサービスです。

データベース・ストレージ・アプリケーションなど、あらゆるデータを統合・連携することで、データを活用したビジネス変革に貢献します。

Relance

ハイスキル フリーランス紹介サービス

Relance

Relance は、プロの現役エンジニア集団が最適なエンジニアを紹介するフリーランスエンジニア紹介サービスです。

技術支援や、1on1 での定期フォローなど、参画後も高いパフォーマンスを維持し続けられる体制を提供しています。

Sreake のエンジニアによる技術支援

3-SHAKE

技術戦略から設計、構築、運用までワンストップ支援する　技術支援サービス

技術戦略
コンサルティング

システム
設計

構築 / 実装
支援

アセスメント
（パフォーマンス / セキュリティ）

運用支援

- Multi Cloud や Cloud Native な先進的技術及び大規模なサービス運用に強みを持つエンジニアによる技術支援
- ベンダー的な役割ではなく「お客様の チームメンバー」という立ち位置で最新技術の提案から運用支援までをトータルご支援

Sreake SRE 導入 / 定着化支援

3-SHAKE

Enterprise を中心に様々な業種/業界でSRE を実践してきたナレッジを活かしたSRE チームの導入及び定着化を行います

- SRE の組成/SRE 文化の定着化の為の支援
- お客様の組織に適合した SRE の導入を支援

本日お話する内容

3-SHAKE

話すこと

私達が展開している SRE の実践方法について

Google Cloud の活用による SRE 実践

話さないこと

SRE の概念に関するお話

SRE のスキルセットや詳細なシステムアーキテクチャ

SRE の基本について知りたい方は以下がおすすめです

3-SHAKE

- **SRE 本**
  https://sre.google/sre-book/table-of-contents/
  https://sre.google/workbook/table-of-contents/

- **SRE をわかりやすく理解出来るコンテンツ**
  Google Cloud Days SRE の基本と組織への導入 ～ サービスレベル目標やエラー予算などサービスの信頼性に対する考え方～
  https://cloudonair.withgoogle.com/events/google-cloud-day-digital-21?talk=d3-infra-01

  3-shake SRE Tech Talk #2 (Google Cloud 篠原様による登壇)
  https://youtu.be/g-WMeetjoRM?t=218

3SHAKE

2. SRE 実践のためのアプローチ

Sreake 流 SRE Roadmap

3-SHAKE

組織組成するとき/組織改革するときに失敗するパターン

3-SHAKE

SRE チームに限らずあらゆる組織を立ち上げる時は様々な苦労や困難な壁にぶつかります。

重要なのは**組織組成や改革は当事者や周りが冷めてしまったら終わり**だということです。

その為に生まれたての組織は尚更慎重に活動していく必要があります。

- 組織の役割が曖昧でメンバーに目的が浸透していない
- 活動が他のチームや上層部から理解が得られない
- 現実と折り合いを付けずに到達困難な高すぎる目標を立ててしまう
- 新しい活動に割けるリソースがなく、コミット量が少ない

class SRE implements DevOps

3-SHAKE

class SRE は DevOps を実装するとありますが ...

※ SRE には、必ずしもDevOps インターフェースの一部ではない、追加のプラクティスや推奨事項が含まれている場合があります。

DevOps	SRE
Reduce organization silos 組織のサイロ化を無くす	Share ownership with developers by using the same tools and techniques across the stack 同じツールやテクニックを使用して、開発者とオーナーシップを共有する
Accept failure as normal 失敗を当たり前のように受け入れる	Have a formula for balancing accidents and failures against new releases 事故や失敗と新製品とのバランスをとるための公式を持つ
Implement gradual change 徐々に変化させる	Encourage moving quickly by reducing costs of failure 失敗したときのコストを減らすことで、迅速な行動を促します。
Leverage tooling & automation ツールと自動化の活用	Encourages "automating this year's job away" and minimizing manual systems work to focus on efforts that bring long-term value to the system システムに長期的な価値をもたらす取り組みに集中するために、「今年の仕事を自動化する」ことを奨励し、手作業によるシステム作業を最小限に抑える
Measure everything すべてを測定する	Believes that operations is a software problem, and defines prescriptive ways for measuring availability, uptime, outages, toil, etc. オペレーションはソフトウェアの問題であると考え、アベイラビリティー、アップタイム、アウテージ、トイルなどの測定方法を規定しています。

参照: https://cloud.google.com/blog/products/gcp/sre-vs-devops-competing-standards-or-close-friends

class SRE implements DevOps

3-SHAKE

class SRE は DevOps を実装するとありますが ...

※SRE には、必ずしもDevOps インターフェースの一部ではない、追加のプラクティスや推奨事項が含まれている場合があります。

DevOps	SRE
Reduce organization silos 組織のサイロ化を無くす	Share ownership with developers by using the same tools and techniques across the stack 同じツールやテクニックを使用して、開発者とオーナーシップを共有する
ツールと自動化の活用	efforts that bring long-term value to the system システムに長期的な価値をもたらす取り組みに集中するために、「今年の仕事を自動化する」ことを奨励し、手作業によるシステム作業を最小限に抑える
Measure everything すべてを測定する	Believes that operations is a software problem, and defines prescriptive ways for measuring availability, uptime, outages, toil, etc. オペレーションはソフトウェアの問題であると考え、アベイラビリティー、アップタイム、アウテージ、トイルなどの測定方法を規定しています。

こういった手法を受け入れるだけの組織的土壌がありますか？
なければ作れますか？
他チームから理解を得て協働が可能ですか？

参照: https://cloud.google.com/blog/products/gcp/sre-vs-devops-competing-standards-or-close-friends

SRE はあらゆるレイヤーに密接に関わります

3-SHAKE

DevOps の「実装者」としての役割をもつ SRE は当然
開発 や これまでの運用といったものと密接に関わります。
また、SLO の定義や SLA の定義を行うためには当然 SRE の
みで終止するものでは無いため、ビジネスサイドとも密接に関
わります。

SRE は単一組織として自分たちのタスクに終始すれば良いも
のではなく、様々な組織や役割と密接に連携し合いながら実
践していく必要があると考えています。

SRE あるある言いたい

3-SHAKE

- SLI / SLO の定義 = SRE
- 障害対応 = SRE
- 自動化の推進 = SRE
- 振り返りの文化を作る = SRE
- CI/CD の整備 = SRE
- 監視環境の整備 = SRE
- インフラの環境構築 = SRE
- アーキテクトレビュー = SRE

SRE チームやること/頼まれること多くなって本質的なこと出来なくなりがち

注: SRE には、必ずしも DevOps インターフェースの一部ではない、追加のプラクティスや推奨事項が含まれている場合があります。

参照: https://cloud.google.com/blog/products/gcp/sre-vs-devops-competing-standards-or-close-friends

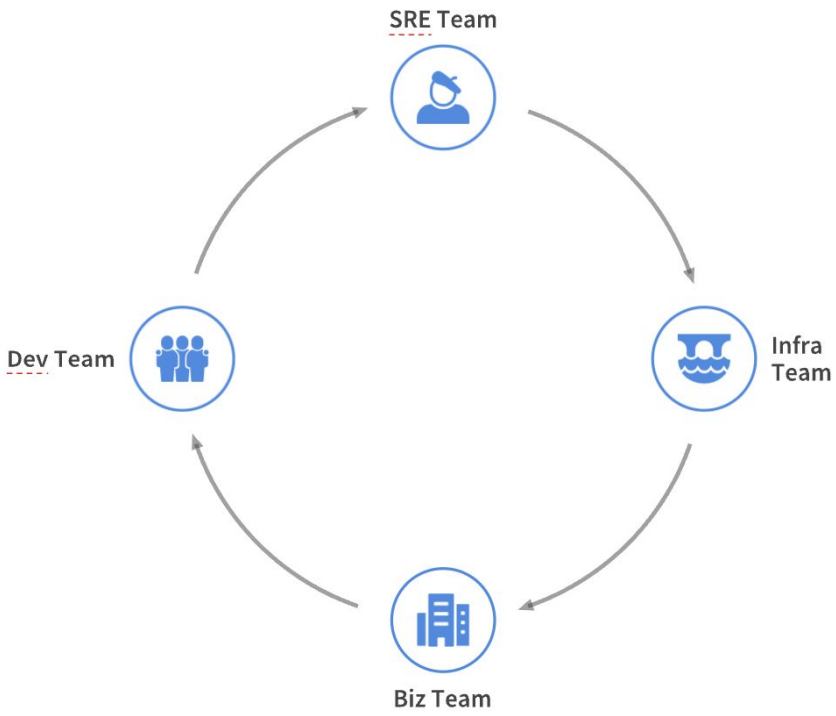
他チームとの協働の重要性

3-SHAKE

SRE チームの役割を明確にして理解を得よう

SRE は SRE チームだけで成立するものではありません。他チームとの協働があってこそ成立するものです。
そのためにも SRE の役割を明確にして有機的に協働する必要があります。

また、Google の SRE のカタチに囚われすぎない事も大事かもしれません。
今の組織やシステム構成にあわせた形で SRE の定義を行わなければ理想だけで上手く行かないと考えています。



Sreake 流 SRE Roadmap

3-SHAKE

SRE の実践内容 (例)

3-SHAKE

- 監視基盤導入
  - Logging
  - Monitoring
  - APM
- SLI / SLO の定義
- 運用体制整備
  - インシデント対応・管理
  - 効果的なアラート
- IaC (Infrastructure as Code)
  - 構成管理の品質チェック
  - GitOps
- CI/CD 導入
  - デプロイの自動化
  - コード品質・脆弱性の検査
  - DevSecOps
- アプリケーションのパッケージ化
  - コンテナ
- パフォーマンス分析
  - 分散トレーシング
  - 負荷試験
  - カオスシナリオ試験

SRE の実践内容 (例)

3-SHAKE

- 監視基盤導入
  - Logging
  - Monitoring
  - APM
- CI/CD 導入
  - デプロイの自動化
  - コード品質・脆弱性の検査
  - DevSecOps

これを複数のサービスにまたがって一気に全部実行するの相当困難。
本音を言うと不可能に近いです…

- IaC (Infrastructure as Code)
  - 構成管理の品質チェック
  - GitOps
  - 負荷試験
  - カオスシナリオ試験

ひとつのサービスから始める

3-SHAKE

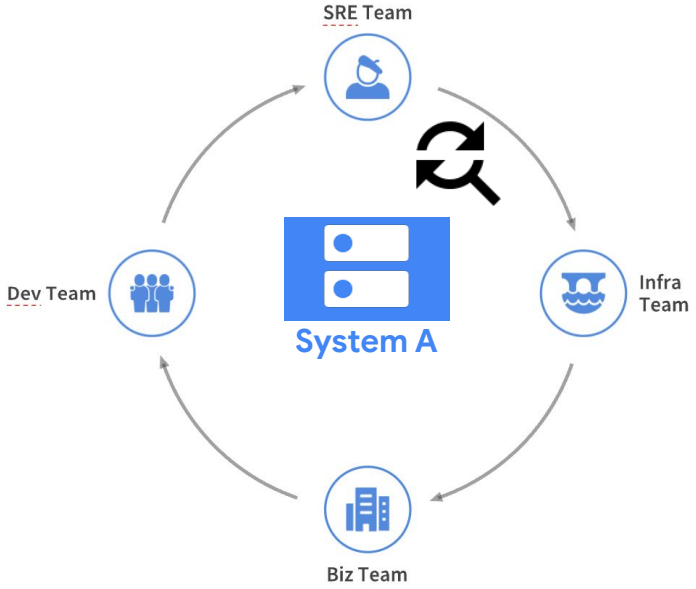
小さく始めてSRE チームとして成功体験を得よう

そのためには、多数のシステムを一気に対象としないこと。
まずはひとつのサービスを対象にSRE を始めて徐々に広げていきましょう。

なお且つ、出来れば新規サービスが望ましい。

理由

- 新しく SLI / SLO 設定する元となる情報を取得する為の基盤導入が難しい。
- 既存の運用がある為、新しい活動への導入が難しいケースがある
- そもそも手を加える前提で作られていない …



ひとつのサービスと
それに関わるチームからSRE の活動を始める

あらゆる情報を収集し始めよう

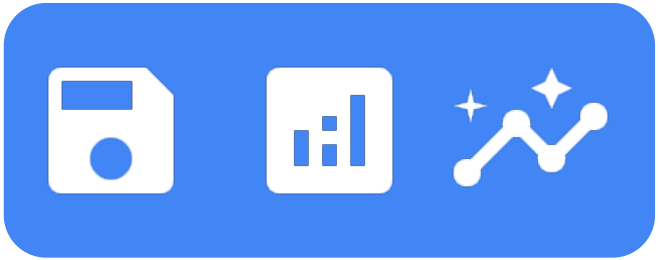
3-SHAKE

SRE を始めるにあたっては、元となるあらゆるデータが揃っていることが前提です。

結局は**データドリブンな世界**にする必要があり、データがない状態では SLI / SLO の設定はそもそも不可能です。

具体的には監視基盤の見直し等を通じてとにかくまずはデータを収集し、可視化出来るところまで持っていきましょう。

闇雲に戦う世界から



データを元に意思決定が出来る世界へ

Sreake 流 SRE Roadmap

3-SHAKE

SLI・SLO・SLA について

3-SHAKE

- **SLI (Service Level Indicator) サービスレベル指標**
  - 何をもとにシステムの良し悪しを判断するかの指標となるもの
  - SLO を定義する時に利用する
- **SLO (Service Level Objective) サービスレベル目標**
  - サービスレベルに対する内的な目標値
- **SLA (Service Level Agreement) サービスレベル保証**
  - サービスレベルに対する対外的な保証値

SLI・SLO・SLA について

3-SHAKE

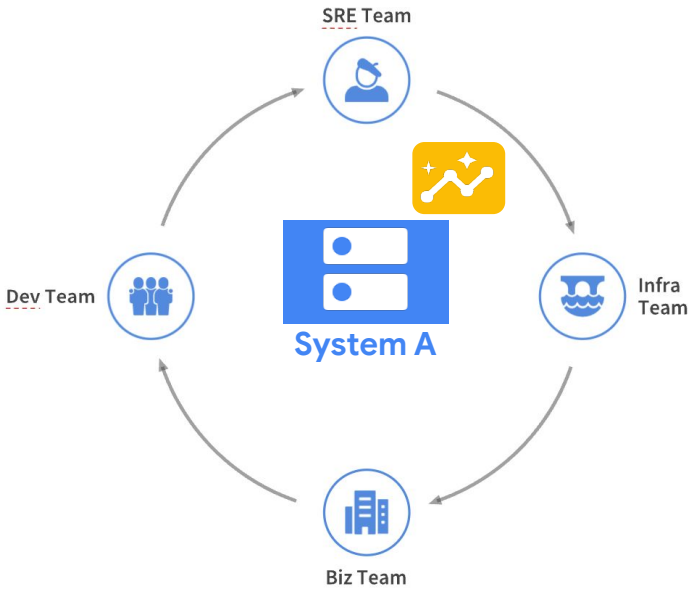
- SLI (Service Level Indicator) サービスレベル指標
  - 何をもとにシステムの良し悪しを判断するかの指標となるもの
  - SLO を定義する時に利用する

上質な SLO を設定するためにもまずは あらゆる指標となりうるあらゆるメトリクスを収集し、可視化ができるようにしよう

SLO の設定について

3-SHAKE

- CUJ (クリティカルユーザージャーニー)を検討していく事から始める
- ただ最初の SLO 設定はエイヤーで決めてしまうのも一つ。
  - SLO はサービスの成長に応じて変わっても良い。
  - ただし、SLO 100% はアンチパターンなので、そういう設計はしない事
- それより大事なのが、**Biz / Dev を置き去りにせず共通認識の獲得を意識しながら設定していくこと**
  - 定期的に振り返りながら組織一丸となって、一緒に高品質なプロダクトを目指していく事が大事。



ポストモーテムの実施

3-SHAKE

ポストモーテムとは、インシデント、その影響、インシデントを軽減または解決するために取られたアクション、根本原因、およびインシデントの再発を防止するためのフォローアップアクションを書面で記録することです。

- 振り返りから学ぶ文化を作る為にポストモーテムを行います。
- その為には障害解析に対して追跡が出来るようにしておかなければなりません。
- フォーマットについてはまずは SRE 本にて提示されている Example を流用するのがおすすめ。
- それと何よりも大切なのが **特定個人を非難しない文化** であることです。
  ※ あくまで、システムや仕組みの問題であると考えます。

参考: https://sre.google/sre-book/postmortem-culture/
https://sre.google/sre-book/example-postmortem/

Toil の削減

3-SHAKE

Toil とは、手作業、繰り返される、自動化が可能、戦術的、長期的な価値がない、サービスの成長に比例して増加する、といった特徴を持つ作業です。

- Dev Team からの依頼対応
- 障害対応
- 割り込みタスク
- SRE としての定型業務
- 定期的に発生する メンテナンス作業

まずはこれらを すべて測定することから

どういう作業がどの程度の頻度発生し、どれだけの工数をかけてどういう効果が得られたのかを取得する

参考: https://sre.google/sre-book/eliminating-toil/

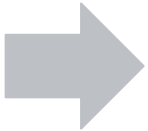
Toil 削減をするために

3-SHAKE

測定、振り返り、自動化のサイクルを回していく

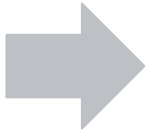
測定

Jira 等のプロジェクト管理ツールを駆使して定形業務や割り込みタスクを測定する。



振り返り

SRE チーム内外含めて定期的な振り返りを通じてToil 特定のを行う



自動化

振り返りによって特定されたToil を自動化によって削減していく

Sreake 流 SRE Roadmap

3-SHAKE

SREの発展と継続

3-SHAKE

- SRE の文化を浸透させていく
  - ひとつの成功体験を得られたら他のサービスにも展開していく
  - 定期的な振り返りを通じてさらなる改善を行っていく
  - 計測したモノからToil 削減の為の自動化の促進を進めていこう

- Error Budget を元に業務をコントロールする
  - そのためには十分な体制やSLI・SLO の定義が必要
  - また、サービスのビジネス的な意味合いによるところもあるので、現実的にはこの運用は相当難しい…

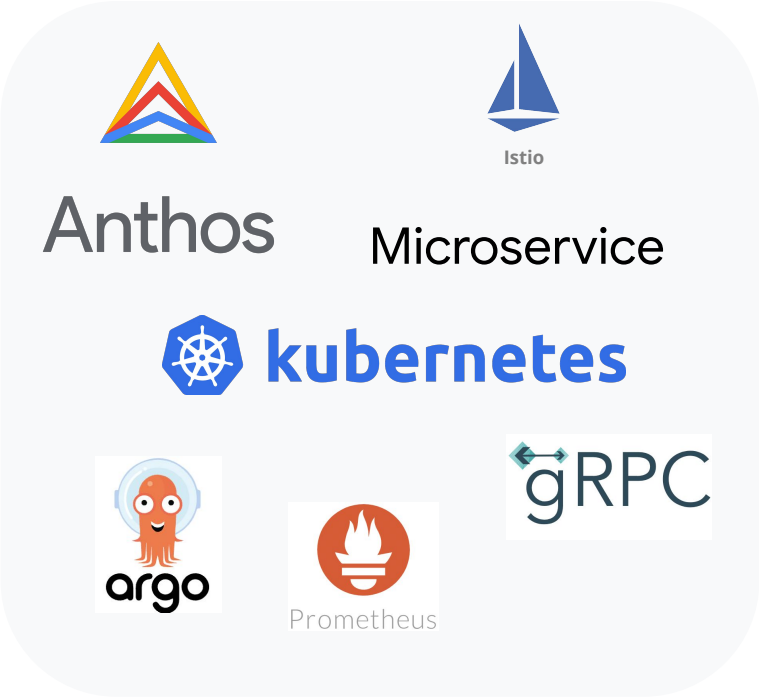
3SHAKE

3. Google Cloud での SRE の実践

アーキテクトにまつわるよくある現象 ※ 主観です

3-SHAKE

- **Cloud や「CloudNative な」技術領域にノウハウが少ない　エンジニアが集まった時に起きる現象**
  - Cloud サービスや Managed サービスを十分に使いこなせない
  - Application Modernization が出来ない
  - 最悪のケースでは可用性・信頼性などに問題が生じる
- **強すぎるエンジニアが集まった時に起きる現象**
  - 要件に対して技術的にオーバーキルしてしまう問題
  - 楽しくなっちゃっていろんな技術を触りたくなってしまって逆に複雑化させてしまう
  - 実はそんなに複雑でなくて良いのにMicroService にしてしまう等



アーキテクトにまつわるよくある現象 ※ 主観です

3-SHAKE

- **Cloud や「CloudNative な」技術領域にノウハウが少ない　エンジニアが集まった時に起きる現象**
  - Cloud サービスや Managed サービスを十分に使いこなせない
  - Application Modernization が出来ない
  - 複雑化してしまう
  - 実はそんなに複雑でなくて良いのにMicroService にしてしまう等

Modern で Open でシンプルで様々な指標が可視化されて、
且つ構成変更が容易なシステム構成が至高
非常に難しいが、苦闘しながらでもここを目指したい

Workload

3-SHAKE

Google Cloud には多様な Workload 実行基盤の選択肢がありますが ...

機能要件及び非機能要件は十分に考慮に入れる必要がありますが Manged なサービスでの実現性を始めに検討した上で徐々に Unmanaed なサービスへ降りていくようなイメージでWorkload を選定するのも一つ。

Workload

3-SHAKE

Google Cloud には多様な Workload 実行基盤の選択肢がありますが ...

機能要件及び非機能要件と最大限向き合う事を前提にしつつも、
「**いかに楽を出来るか**」という観点から考える事も重要です。
それによりそもそもToil の生産量を減らすことが出来るかもしれません。

Ex. 何でもかんでもKubernetes が良いという訳ではない

Managed	Unmanaged

機能要件及び非機能要件は十分に考慮に入れる必要がありますがManged なサービスでの実現性を始めに検討した上で徐々にUnmanaed なサービスへ降りていくようなイメージでWorkload を選定するのも一つ。

CI/CD (Release Engineering)

3-SHAKE

CI/CD を行う目的とは？

❖ **CI**

- ➢ テストを自動化したい
- ➢ コードの品質を高めたい
- ➢ Release Ready な状態にしたい

❖ **CD**

- ➢ 簡単にデプロイ出来るようにしたい
- ➢ 何かあればすぐ切り戻したい
- ➢ 承認など段階を踏んだデプロイをしたい

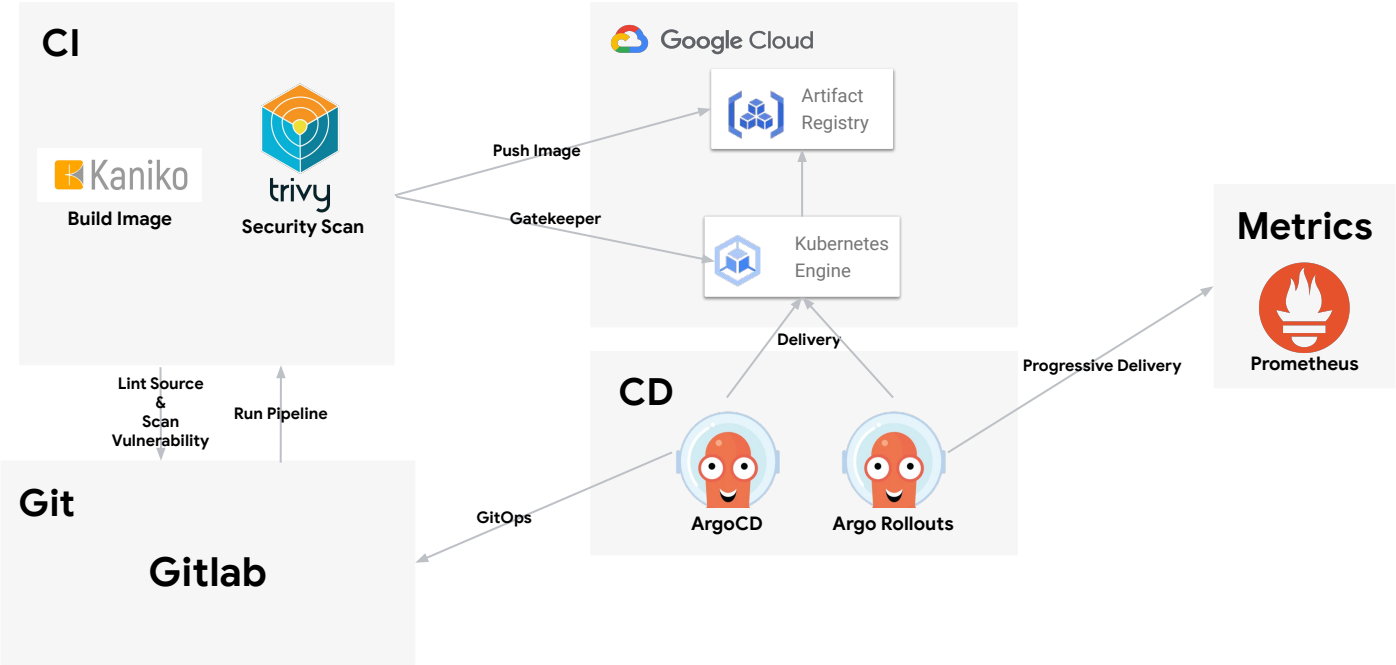
障害発生の多くは人間が手を加えたとき(新しいバージョンのリリース時)に起きると言われています。

CI/CD はリリースサイクルの向上と信頼性の両方を担保する為のアプローチであるべきだと考えています。

CI/CD (Release Engineering)

3-SHAKE

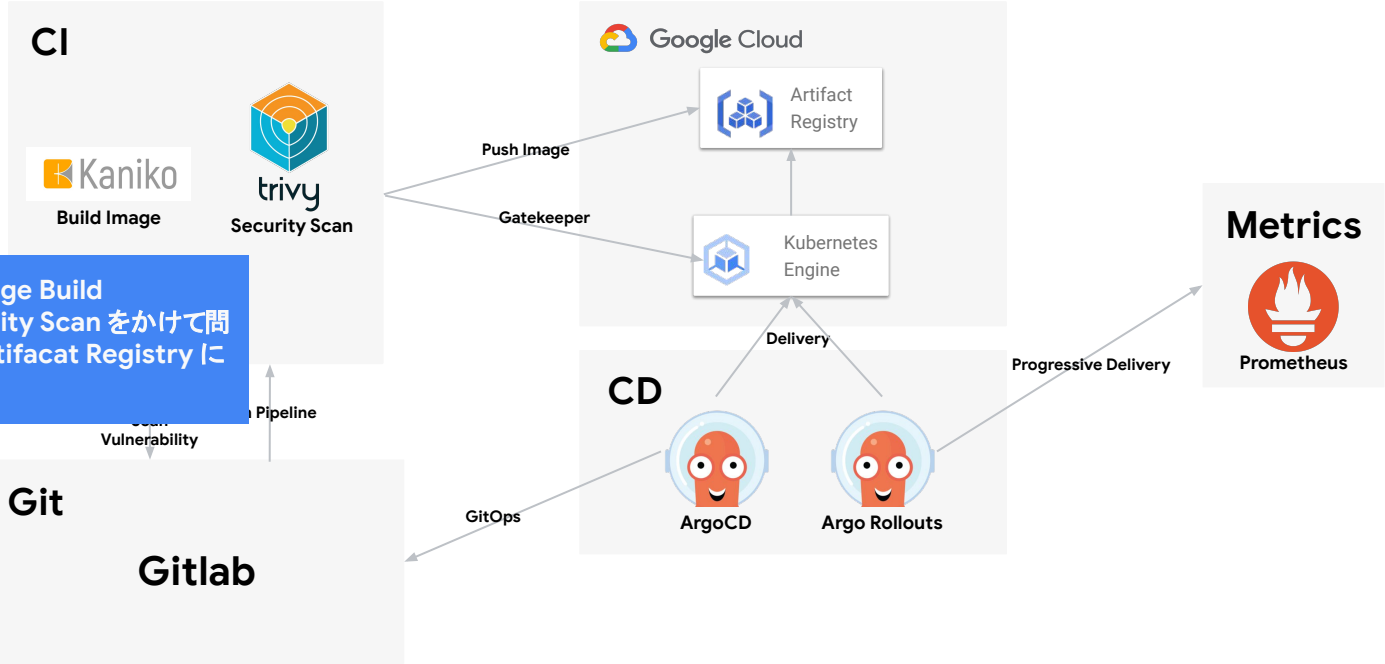
Kubernetes での CI/CD の実践



CI/CD (Release Engineering)

3-SHAKE

Kubernetes での CI/CD の実践



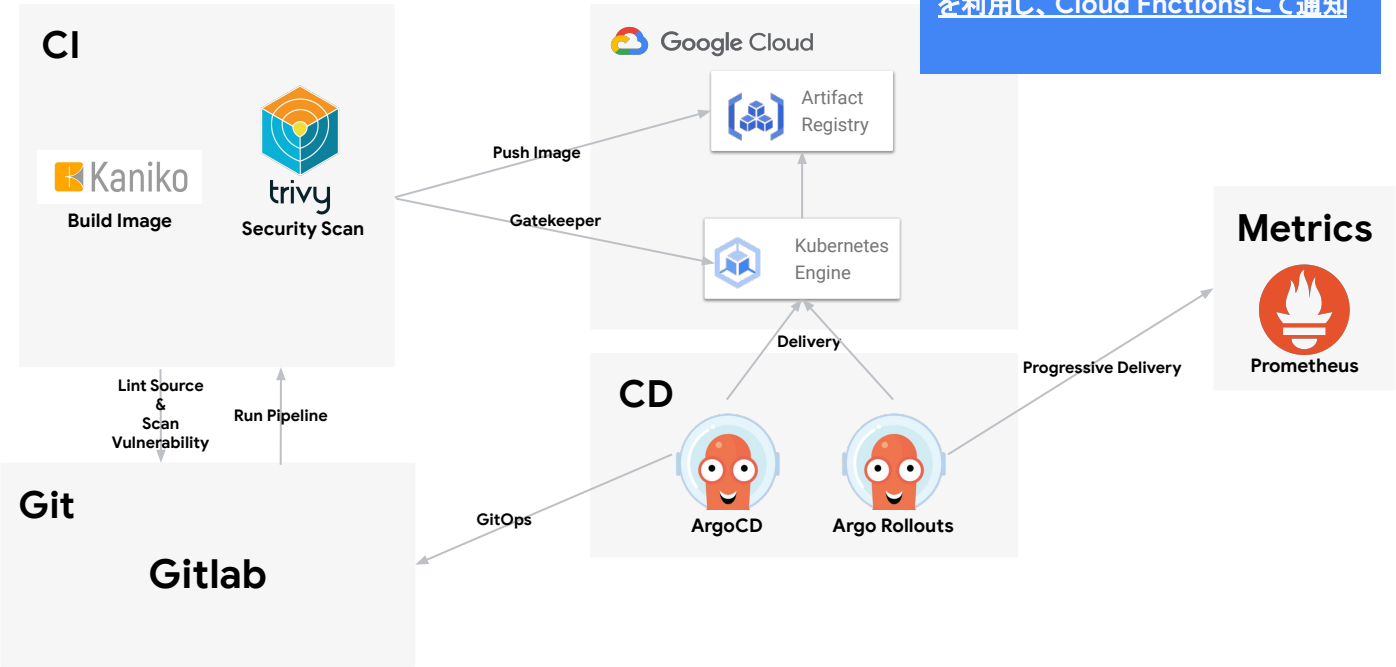
Kaniko にて Image Build
trivy にて Security Scan をかけて問題ない場合はArtifacat Registry に Push

CI/CD (Release Engineering)

3-SHAKE

Kubernetes での CI/CD の実践

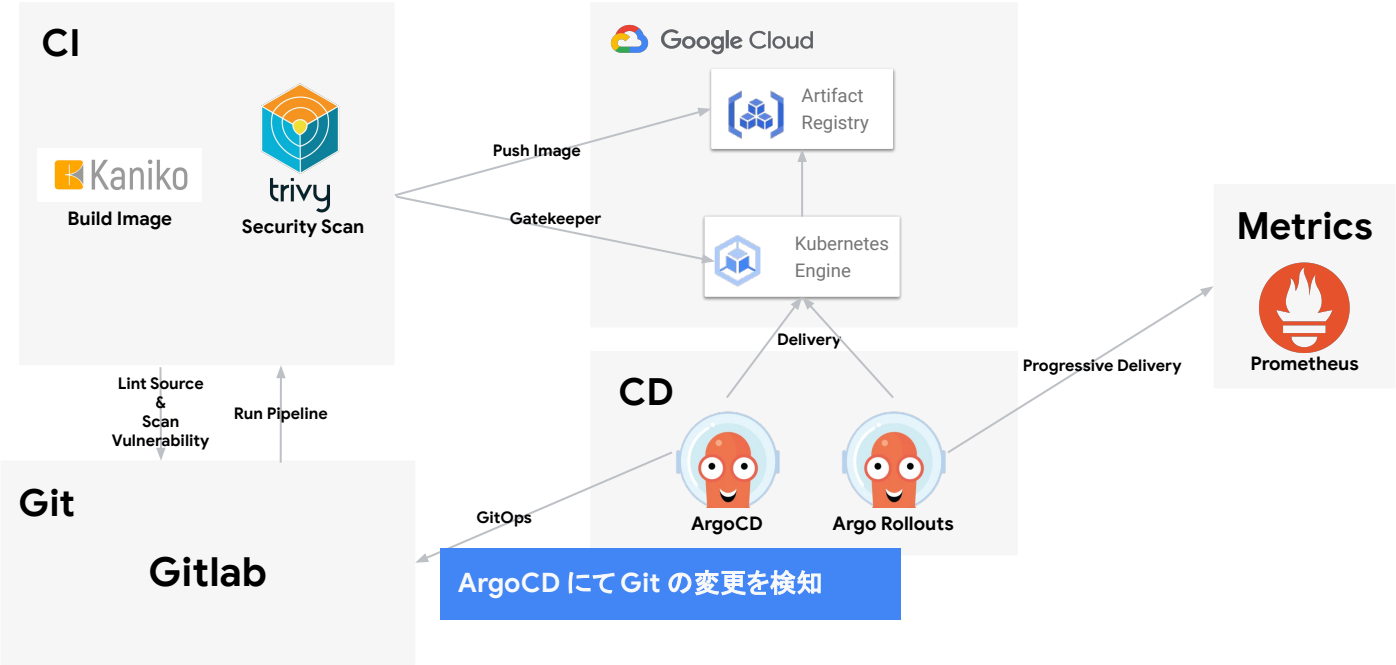
Push 済みの Image については Artifact Registry の Image Scan を利用し、Cloud Fnctionsにて通知



CI/CD (Release Engineering)

3-SHAKE

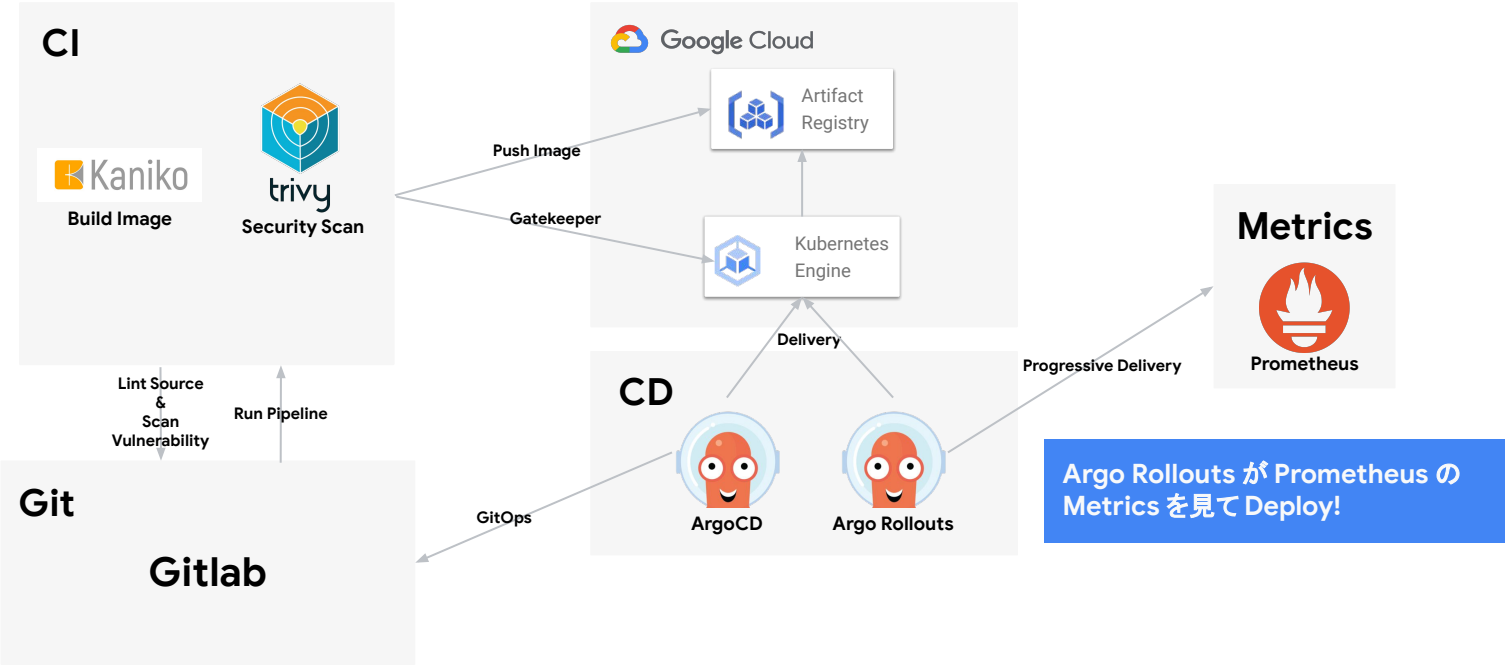
Kubernetes での CI/CD の実践



CI/CD (Release Engineering)

3-SHAKE

Kubernetes での CI/CD の実践



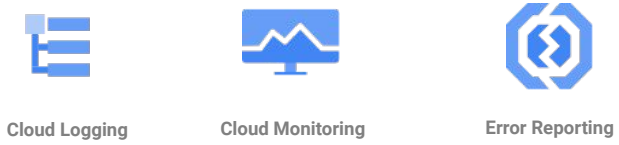
Monitoring & Logging

3-SHAKE

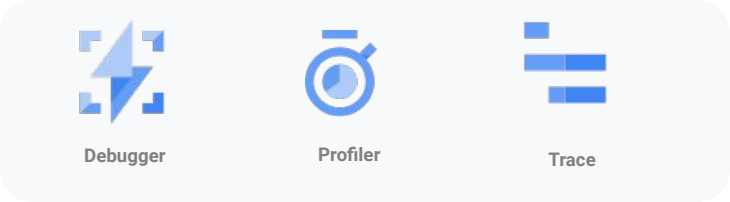
Cloud Operations

Google Cloud のあらゆるサービスとDefault で連携されているのが最大のメリット。
また、SaaS として監視基盤を利用する事が可能なので基盤運用を行う必要がなく、その分工数を減らせる。

※ ただし、可用性の要件によっては OSS や他の SaaS の利用も検討に入れる必要があります。



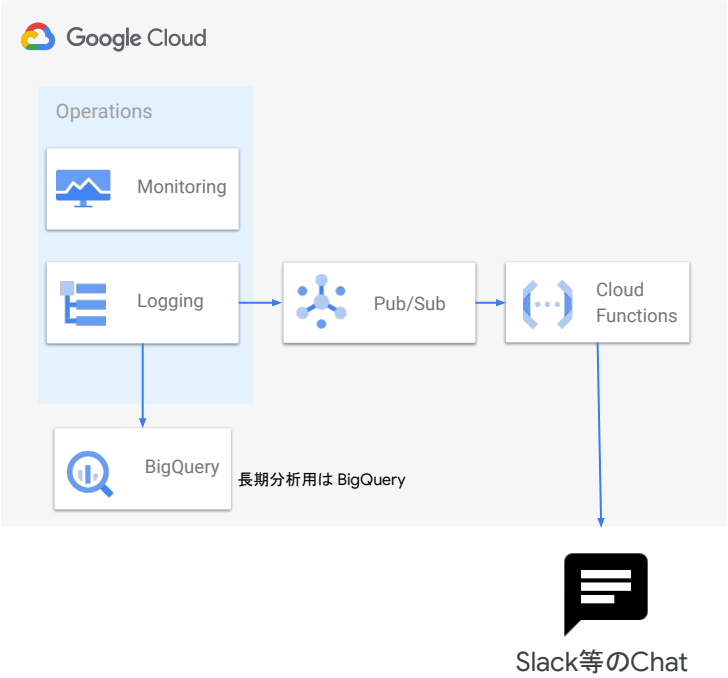
APM



Cloud Logging 通知利用時の Tips

Cloud Logging を利用する場合、ログベースの指標を使って通知可能だが、Log の中身までは通知する事が出来ない。そのため、Log の中身を含めた通知を行いたい場合は Pub/Sub と連携し、Cloud Functions 等を使って独自に実装して通知処理を行う必要がある。

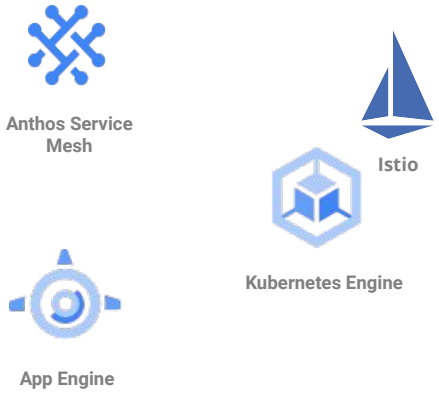
※ただし、個人情報などのセンシティブなデータが含まれていないかは要確認



3-SHAKE

SLO Monitoring の活用

Anthos Service Mesh、Istio on GKE、App Engine は自動的に
SLO Monitoring 対象のサービスとして検出される。
MicroService を運用する上で計測がしにくい各サービスのSLI を取得できるのは大きなメリット。
また、SLI としてログベースの指標やPrometheus のメトリクス等も利用可能なため、手軽にSRE の実践が出来る為、独自で作り込みも可能。



詳細については 弊勉強会でGoogle Cloud 篠原様がデモ付きで話されているのでぜひご参照ください
https://youtu.be/g-WMeetjoRM?t=218

Monitoring & Logging

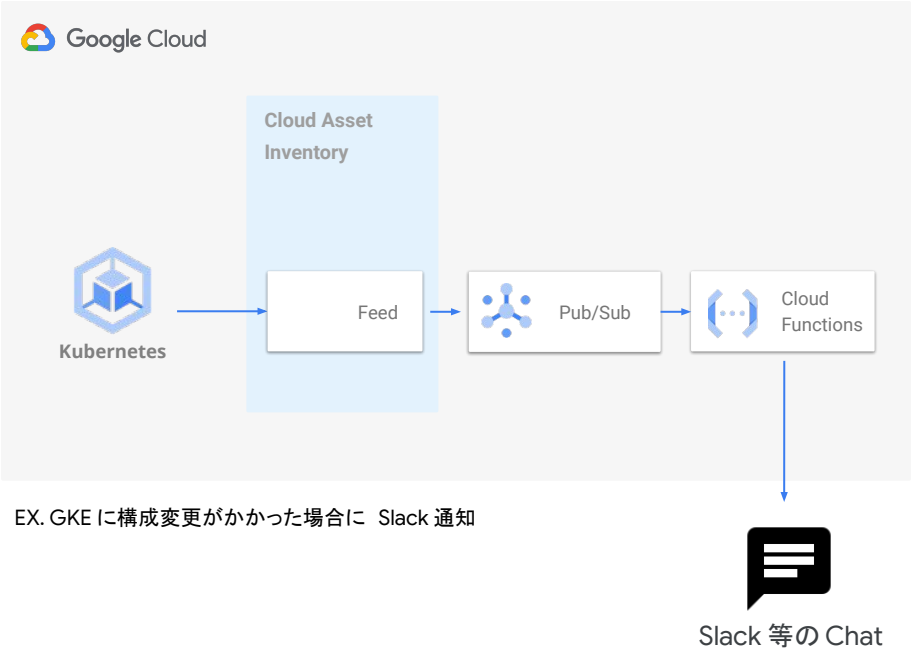
3-SHAKE

Cloud Asset Inventory の活用

プロジェクトとサービスで使用されているGoogle Cloud と Anthos のアセットをすべて表示、モニタリング、分析できるサービス。

ダッシュボードも用意されていて、想定していないリージョンでリソースが使われていないかなども確認できる。

また、アセットの構成変更に関するリアルタイム通知を受け取る事も可能で、変更監視としての通知や想定外の変更に対する戻し等も作り込めば可能。



EX. GKE に構成変更がかかった場合に Slack 通知

Alerting & Incident Management

3-SHAKE

Google Cloud と PagerDuty の組み合わせで SRE を加速させる

オンコール機能

- ❖ 適切なオンコール体制を作る為のスケジューリング機能

実践的な Alerting

- ❖ EventRule を始めとしたインテリジェンスなアラート制御

インシデント管理

- ❖ 様々なインテグレーションに対応しているので、包括的にインシデント管理が可能

障害発生時の対応の追跡

- ❖ ユーザーのアクションを自動で記録

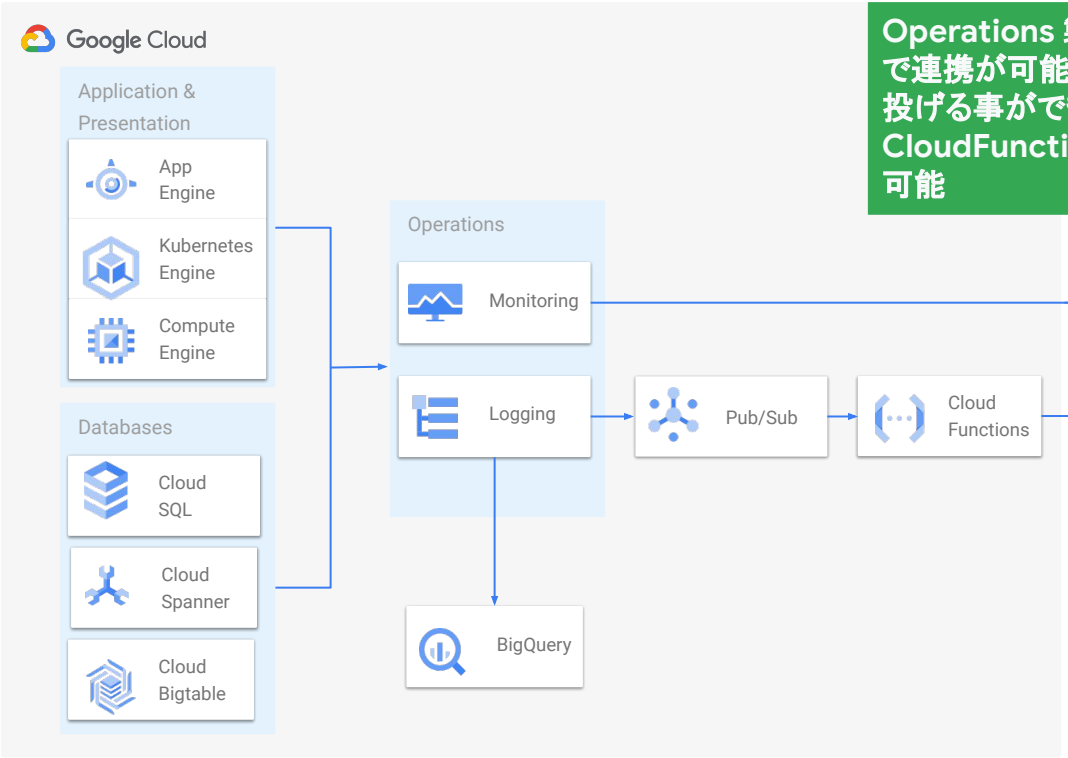
ポストモーテム

- ❖ 組み込みでポストモーテムの機能があり、失敗から学ぶ文化をすぐに作れる

Google Cloud　PagerDuty

Alerting & Incident Management

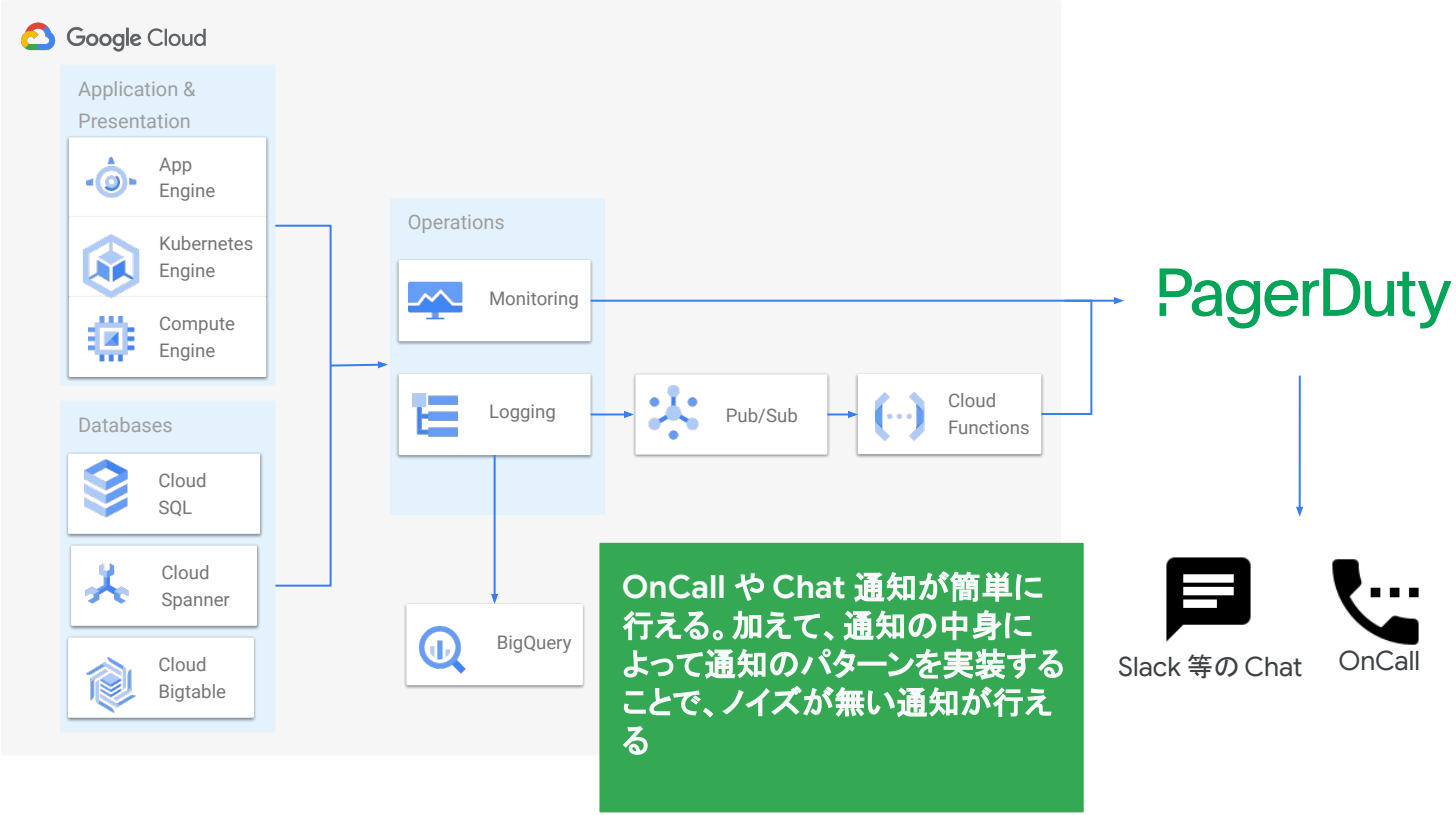
3-SHAKE



Operations 製品とワンクリックで連携が可能。Mail や API でも投げる事ができるため、CloudFunctions 等からも連携可能

PagerDuty

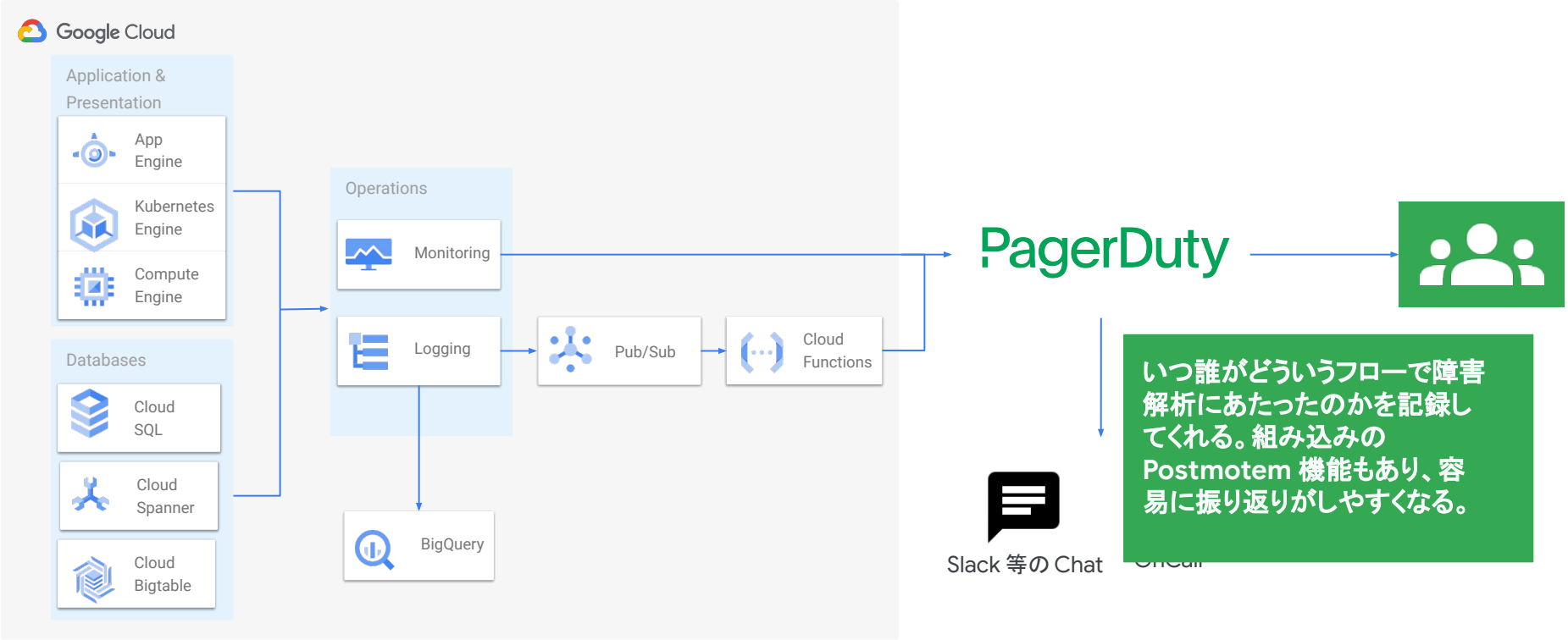
Alerting & Incident Management



OnCall や Chat 通知が簡単に行える。加えて、通知の中身によって通知のパターンを実装することで、ノイズが無い通知が行える

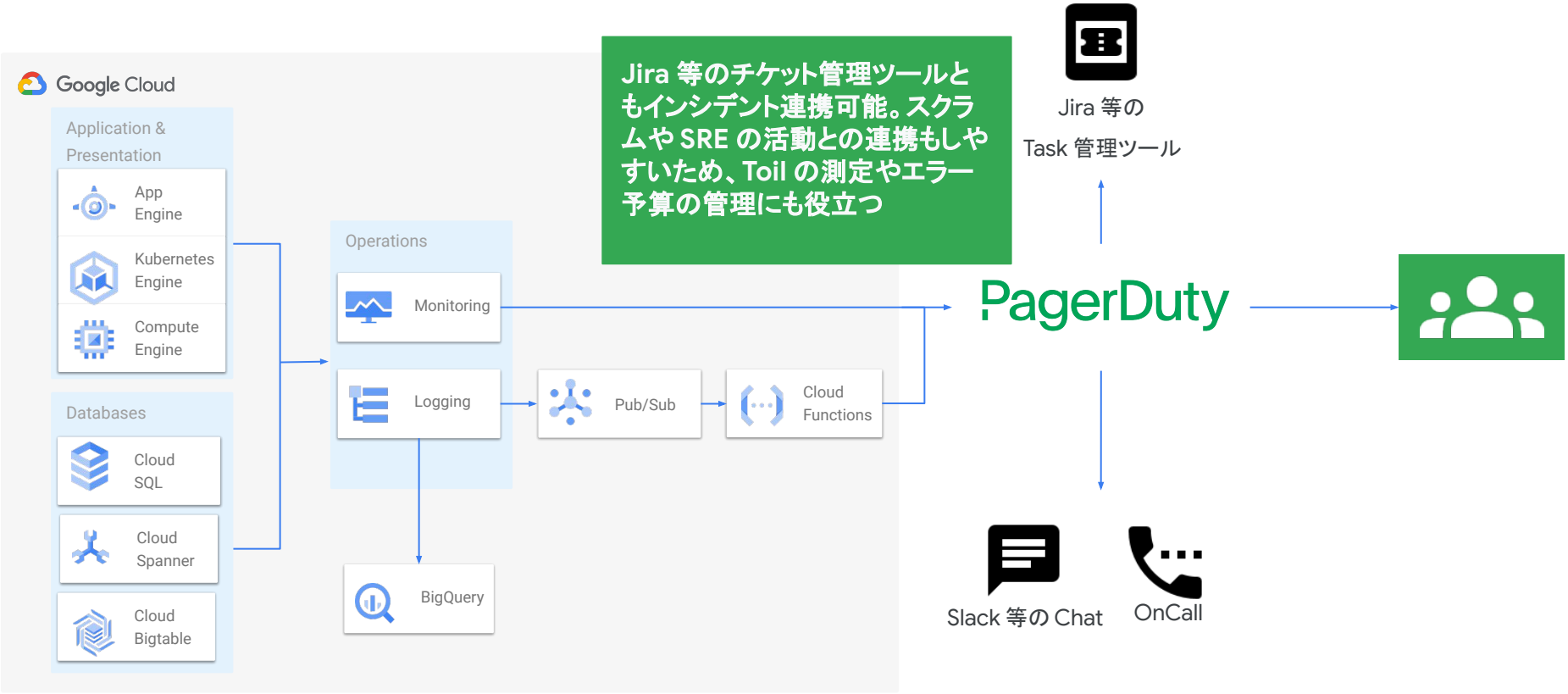
Alerting & Incident Management

3-SHAKE



Alerting & Incident Management

3-SHAKE



4. まとめ

Sreake 流 SRE Roadmap

3-SHAKE

まとめ

3-SHAKE

- Google の SRE は素晴らしい見本になる。でも、SRE はウェットな部分も多いのが現実。　自分たちの組織やメンバー構成を考えながら組織にあった最適な SRE を実現していこう。
- SRE を始める時はなるべく小さく、手を付けやすいところから始めよう
- アーキテクチャは極力「シンプル」に。 そもそも Toil を生まないアーキテクチャを目指そう
- Saas 製品を上手く組み合わせながら手を抜けるところは手を抜こう

3SHAKE

Thank you