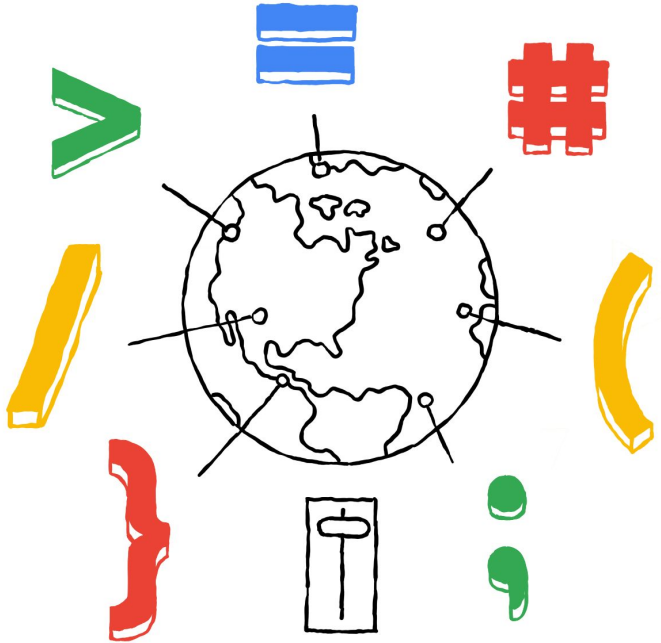
Google 流、理想的な CI / CD を導入するための考え方とその実践

Application Modernization Specialist
中丸 良



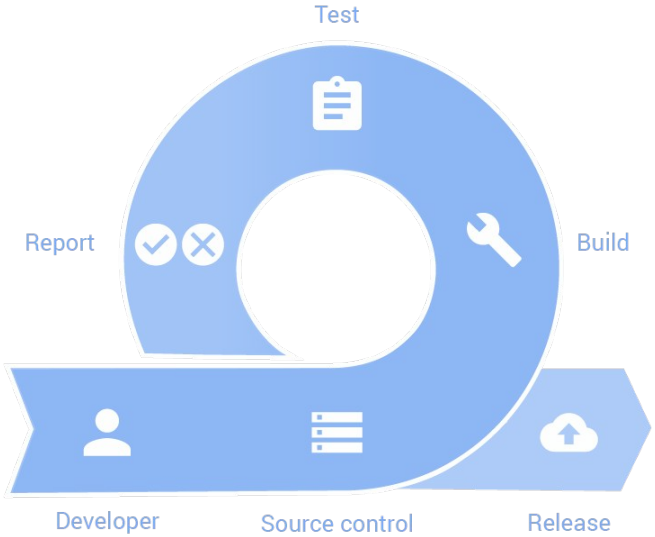
Google Cloud

CI: 継続的インテグレーション

とは　チームで開発したソースを頻繁に統合し、可能な限り速く**問題のある変更を**自動的に**発見すること。**

なぜ

- 一定以上の品質を安定して維持
- より迅速なフィードバック
- 変更そのものや問題の可視化
- etc..

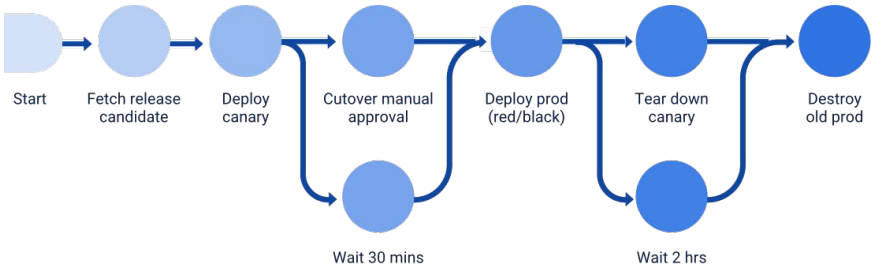


CD: 継続的デリバリー

とは　CI の成功など一定条件を満たす度に、CI で生成された成果物を、ポリシーに沿って本番環境にリリースすること。

なぜ

- デプロイにまつわるリスクの低減
- より早いユーザーフィードバック
- 信頼できる形での進捗
- etc..

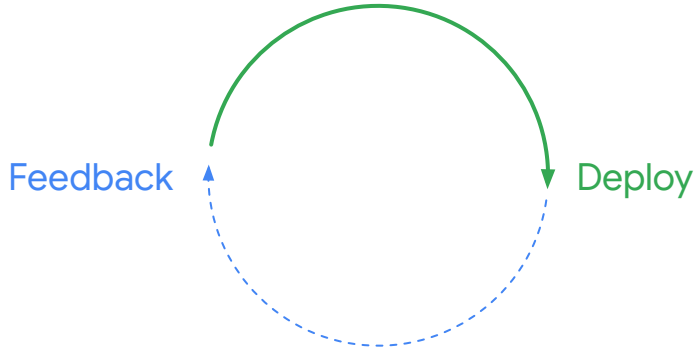


DevOps

とは　ソフトウェア開発サイクルの一連の流れを改善することによってこれらを実現する、**プラクティス / ガイドライン / 文化**

なぜ

- アイデアをマーケットに投入するまでの時間を短くする
- ユーザーからのフィードバックを早く得る / それに対するレスポンスも迅速に
- etc..



ウォーミングアップ

1. コードがコミットされてから本番環境で動作するまでの時間、どのくらいですか？
2. デプロイしたり、リリースしたりする頻度はどのくらいですか？
3. サービス障害や欠陥が発生した場合、復旧にどのくらい時間がかかりますか？
4. 本番環境への変更や製品リリースのうち、サービス低下が起こりその後修復が必要になる割合、どのくらいですか？

DevOps Research and Assessment（DORA）クイックチェック

https://cloud.google.com/devops

- "クイックチェック" へ
- さきほどのアンケート結果を入力してみてください！
- 業界における **DevOps パフォーマンス**を評価
  - エリート
  - ハイパフォーマー
  - ミディアムパフォーマー
  - ローパフォーマー
- CI / CD 導入の成果指標としても利用可

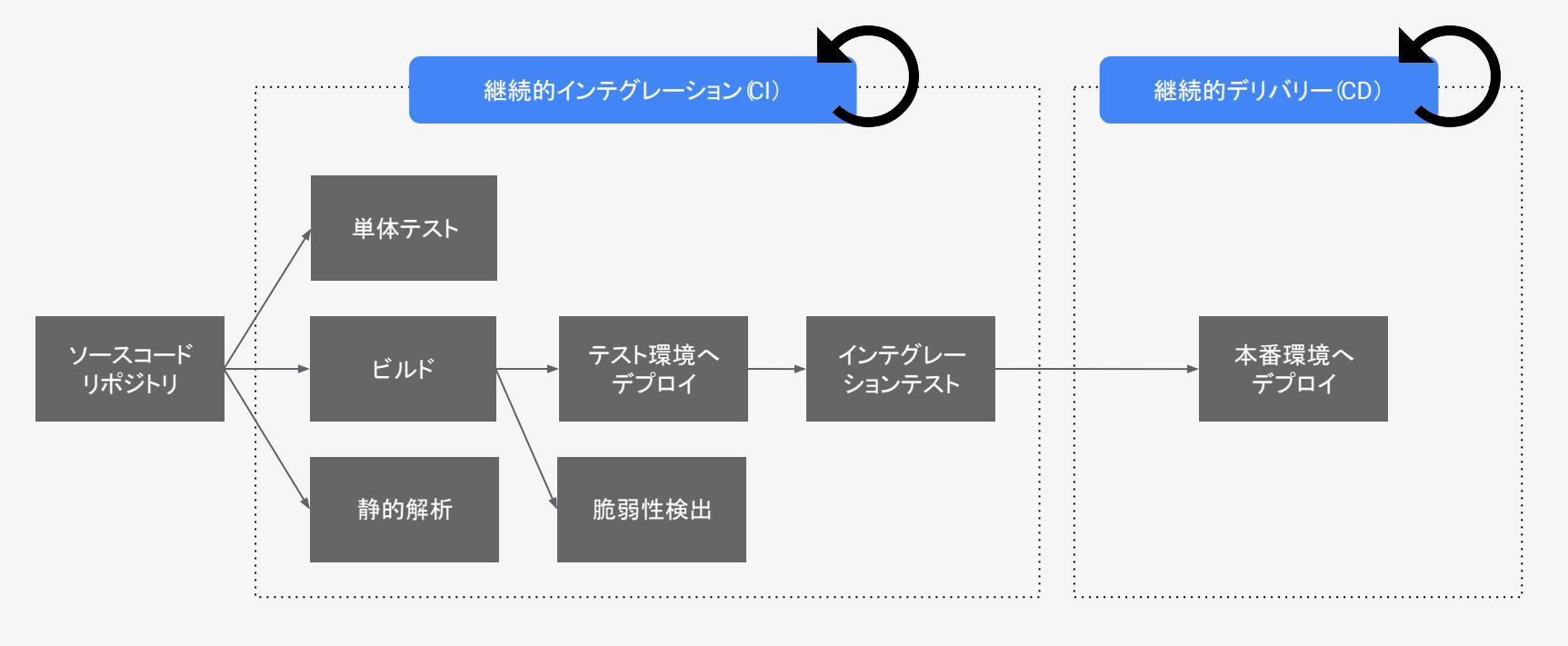
アジェンダ

1. CI / CD 導入の狙いと計測
2. CI / CD の設計
3. CI / CD の始め方

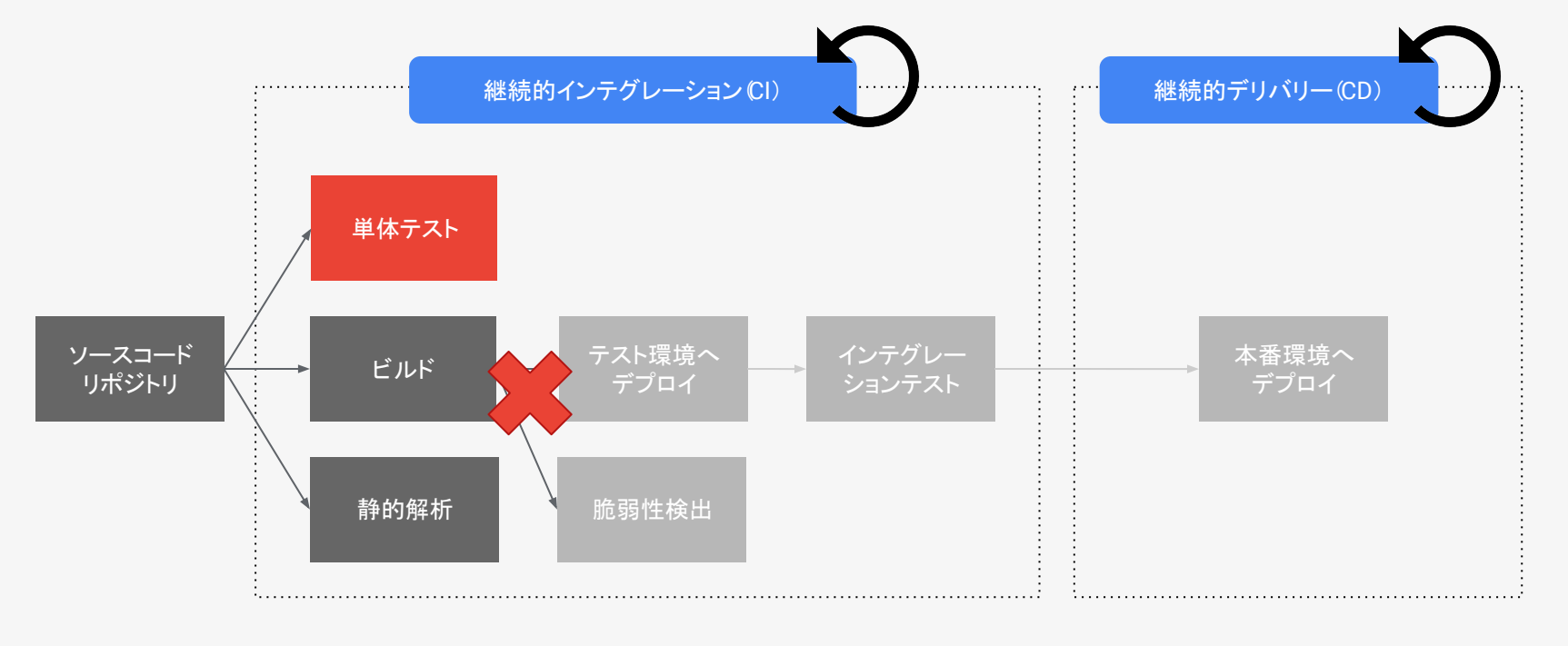
CI / CD
導入の狙いと計測

Google Cloud

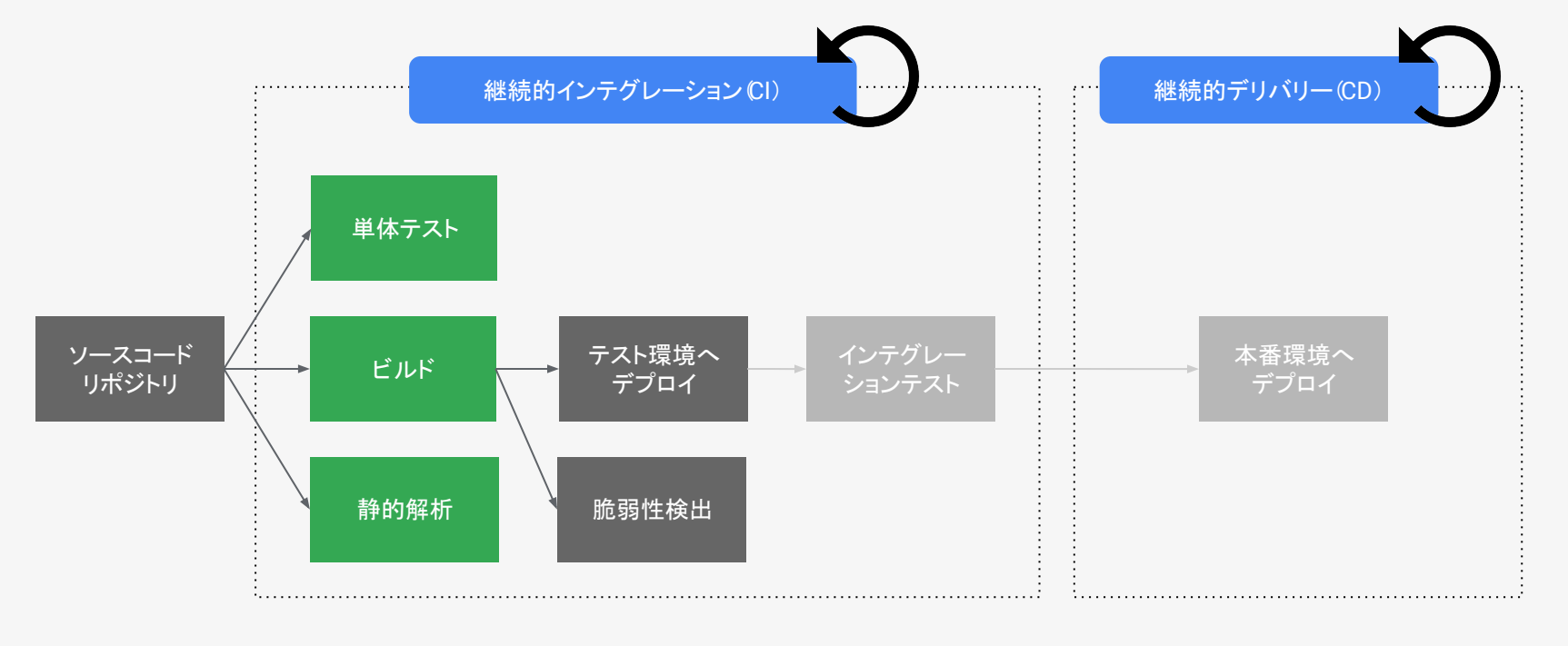
CI / CD とは


継続的インテグレーション(CI)
継続的デリバリー(CD)
単体テスト
ソースコード
リポジトリ
ビルド
テスト環境へ
デプロイ
インテグレー
ションテスト
本番環境へ
デプロイ
静的解析
脆弱性検出

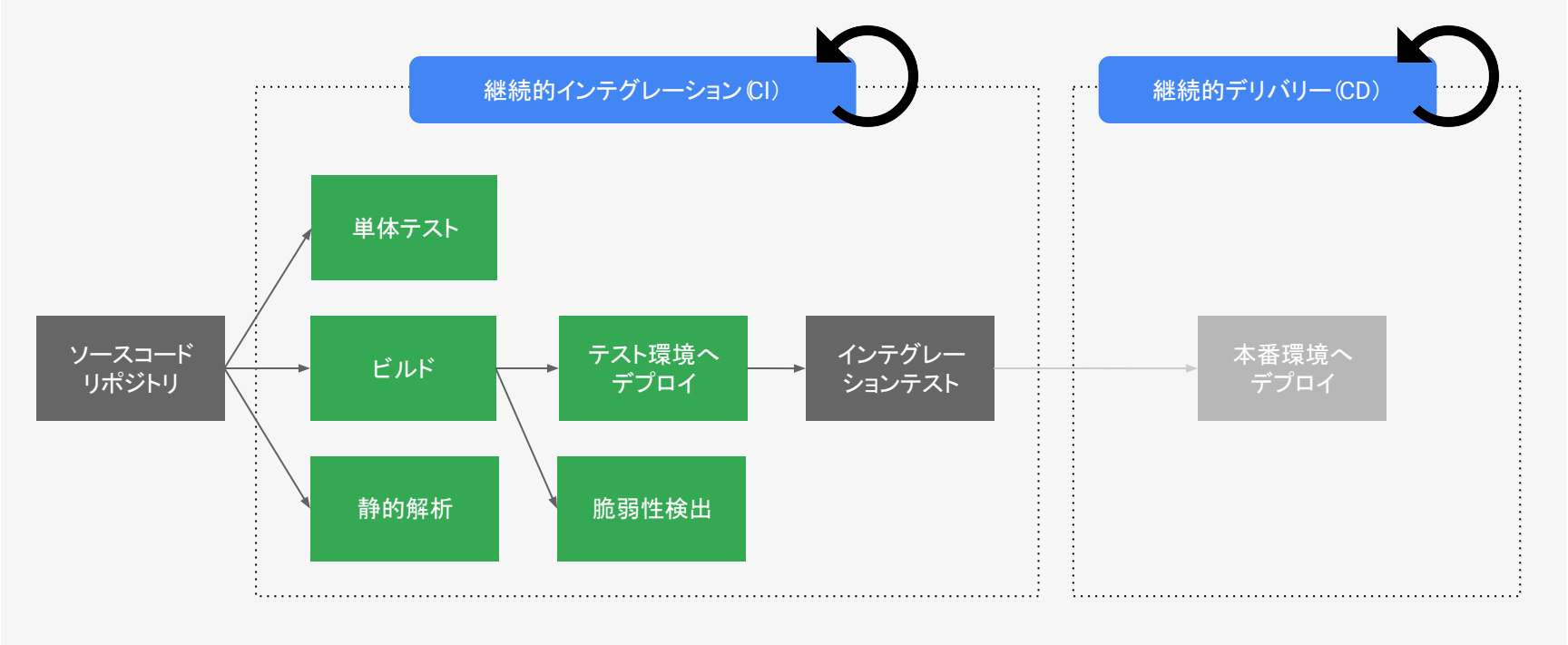
CI / CD とは


継続的インテグレーション(CI)
継続的デリバリー(CD)
単体テスト
ソースコード
リポジトリ
ビルド
テスト環境へ
デプロイ
インテグレー
ションテスト
本番環境へ
デプロイ
静的解析
脆弱性検出

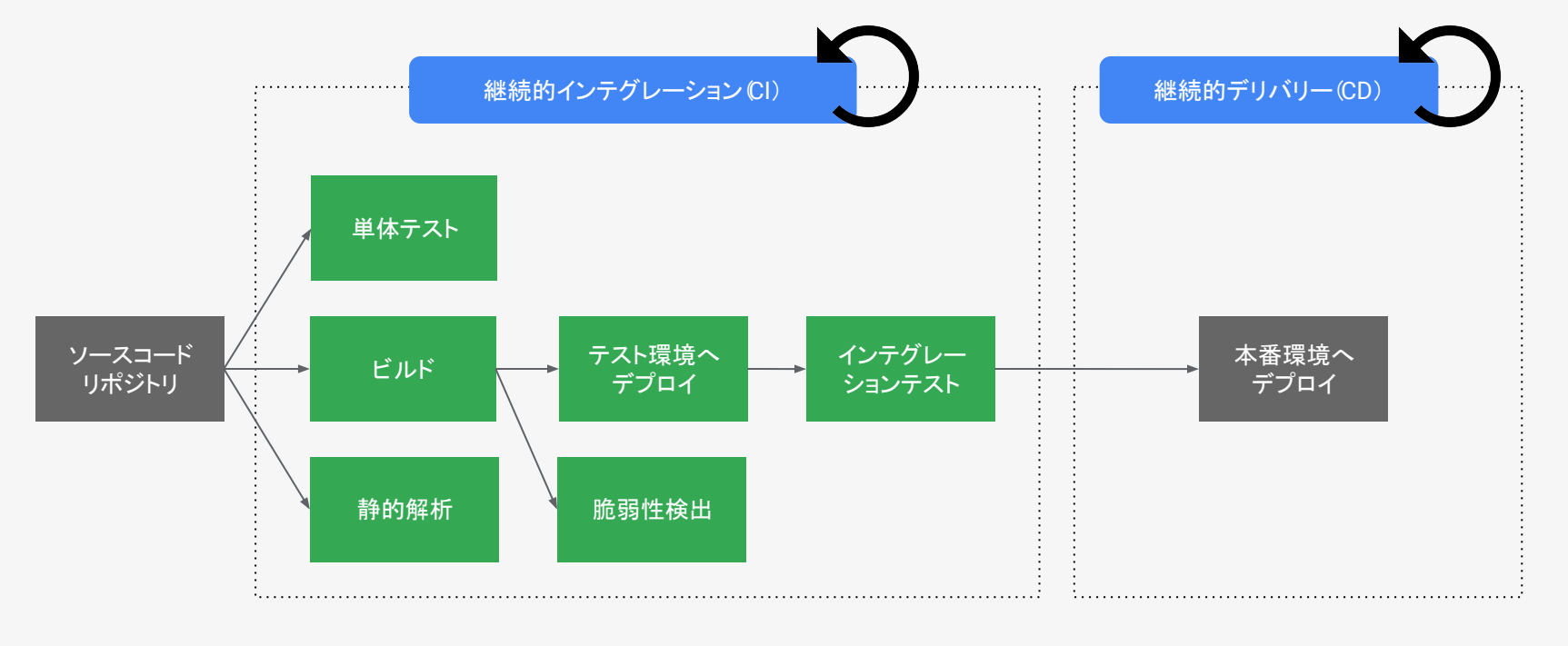
CI / CD とは


継続的インテグレーション(CI)
継続的デリバリー(CD)
単体テスト
ソースコード
リポジトリ
ビルド
テスト環境へ
デプロイ
インテグレー
ションテスト
本番環境へ
デプロイ
静的解析
脆弱性検出

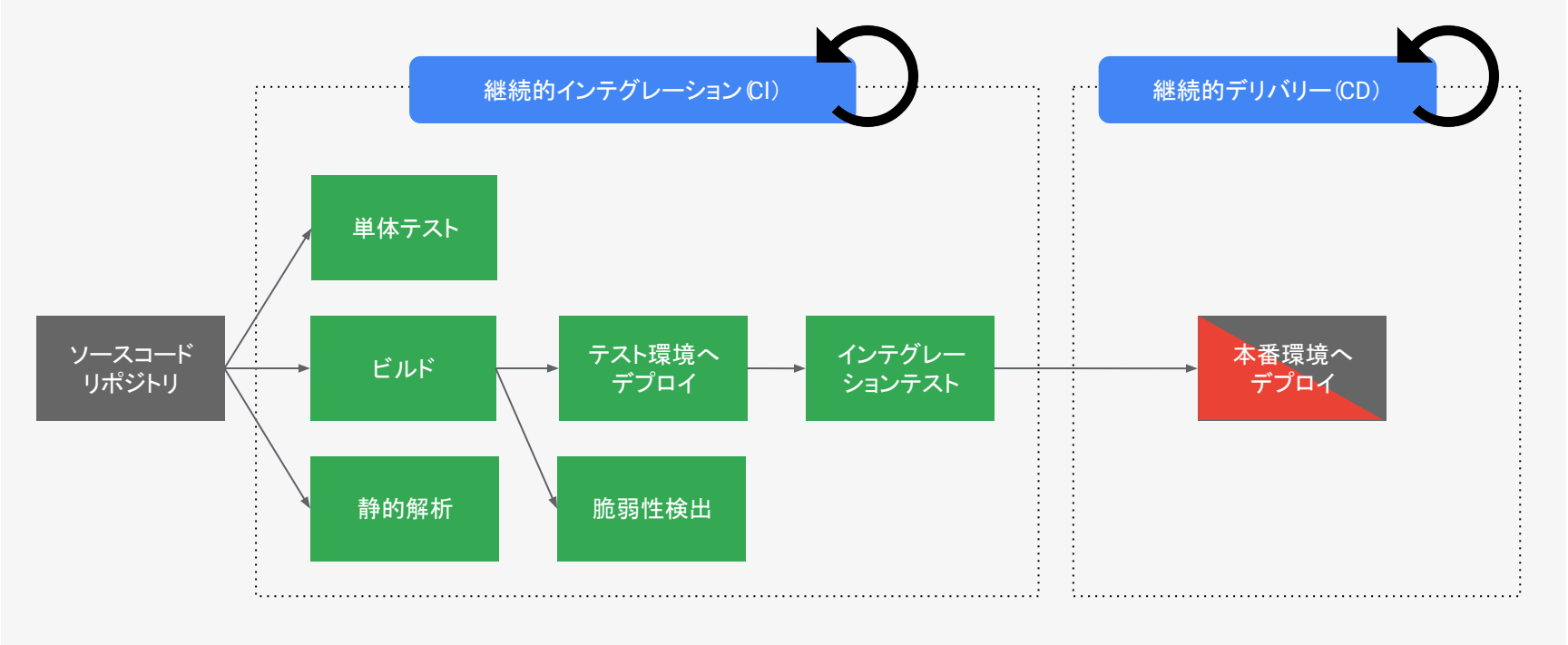
CI / CD とは


継続的インテグレーション(CI)
継続的デリバリー(CD)
単体テスト
ソースコード
リポジトリ
ビルド
テスト環境へ
デプロイ
インテグレー
ションテスト
本番環境へ
デプロイ
静的解析
脆弱性検出

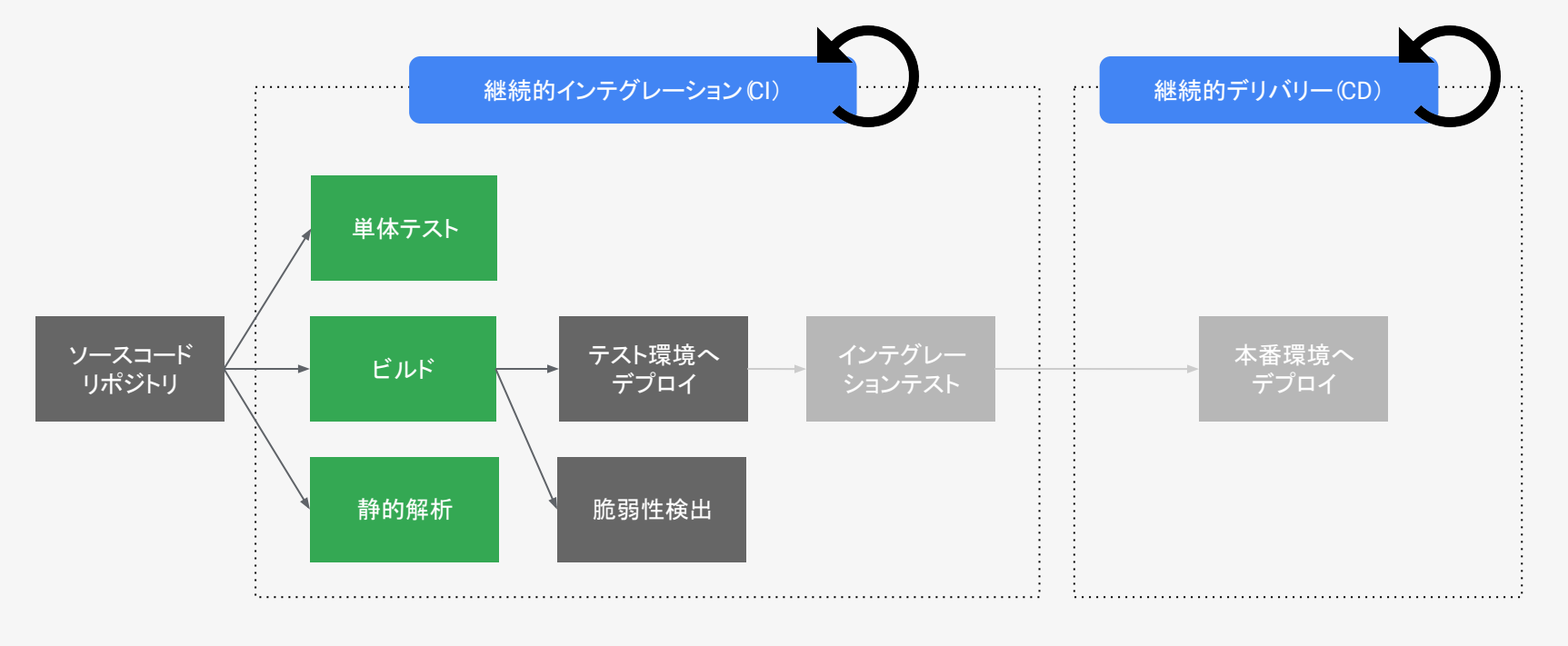
CI / CD とは


継続的インテグレーション(CI)
継続的デリバリー(CD)
単体テスト
ソースコード
リポジトリ
ビルド
テスト環境へ
デプロイ
インテグレー
ションテスト
本番環境へ
デプロイ
静的解析
脆弱性検出

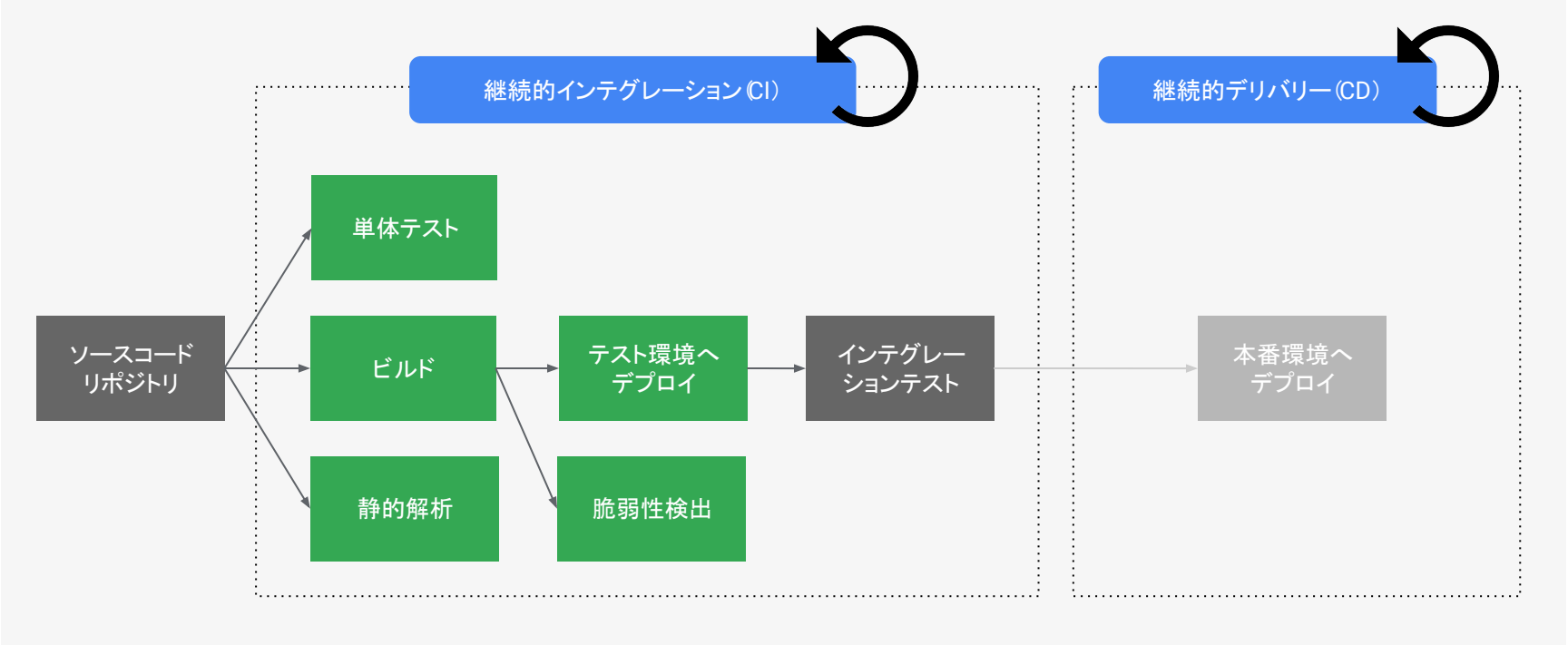
CI / CD とは


継続的インテグレーション(CI)
継続的デリバリー(CD)
単体テスト
ソースコード
リポジトリ
ビルド
テスト環境へ
デプロイ
インテグレー
ションテスト
本番環境へ
デプロイ
静的解析
脆弱性検出

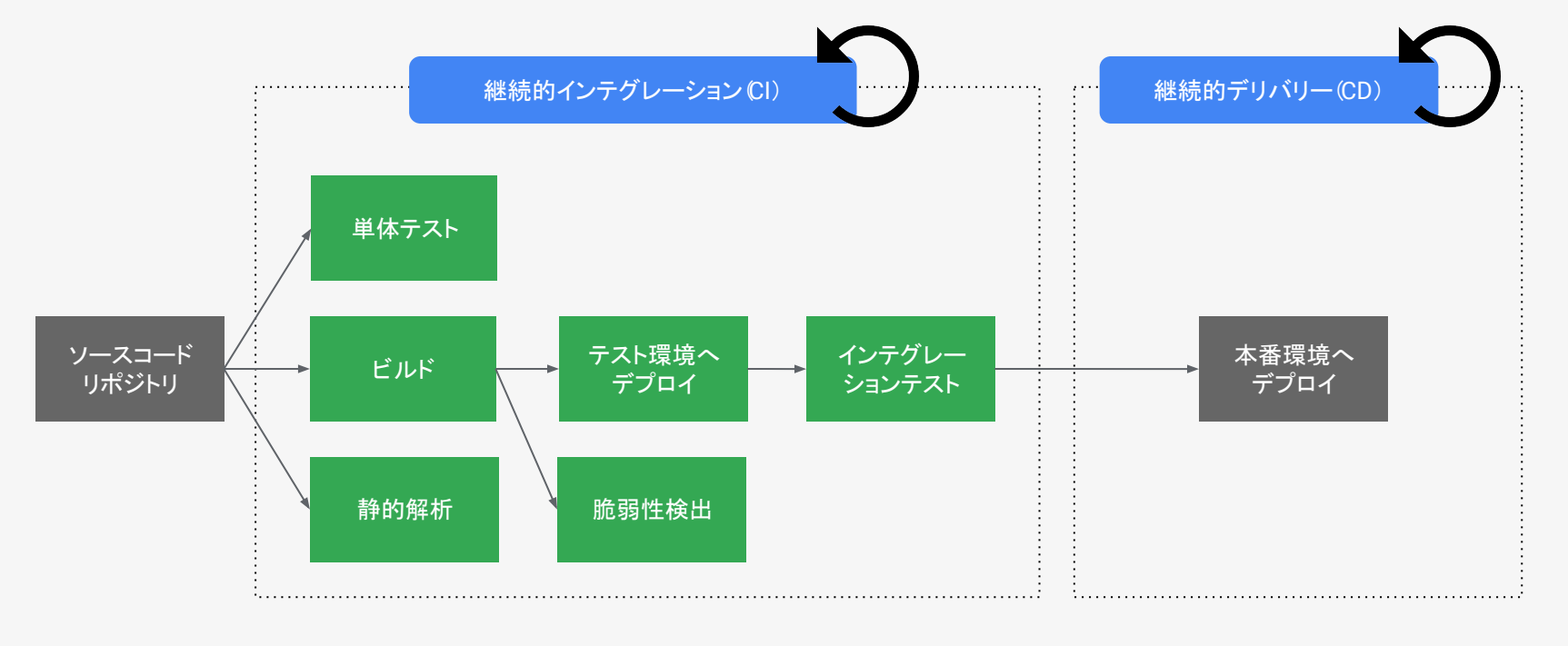
CI / CD とは


継続的インテグレーション(CI)
継続的デリバリー(CD)
単体テスト
ソースコード
リポジトリ
ビルド
テスト環境へ
デプロイ
インテグレー
ションテスト
本番環境へ
デプロイ
静的解析
脆弱性検出

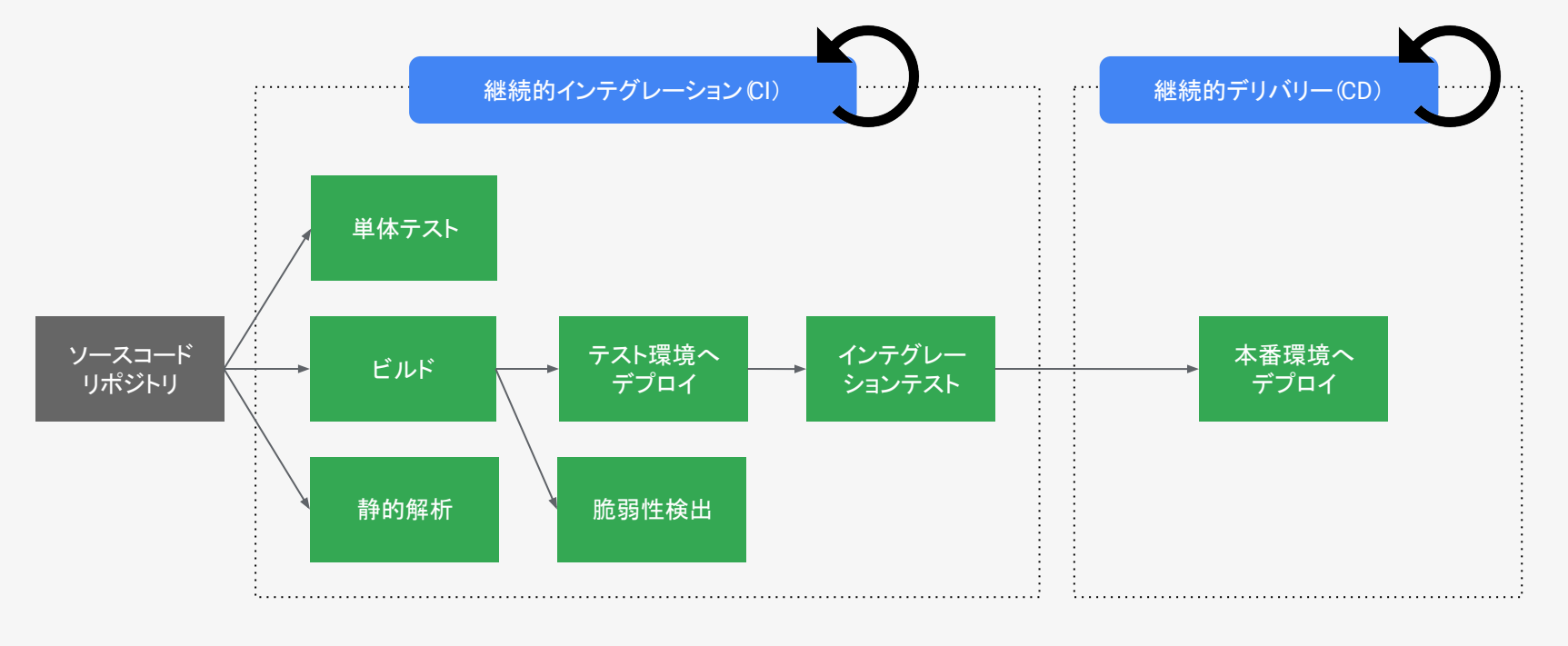
CI / CD とは


継続的インテグレーション(CI)
継続的デリバリー(CD)
単体テスト
ソースコード
リポジトリ
ビルド
テスト環境へ
デプロイ
インテグレー
ションテスト
本番環境へ
デプロイ
静的解析
脆弱性検出

CI / CD とは

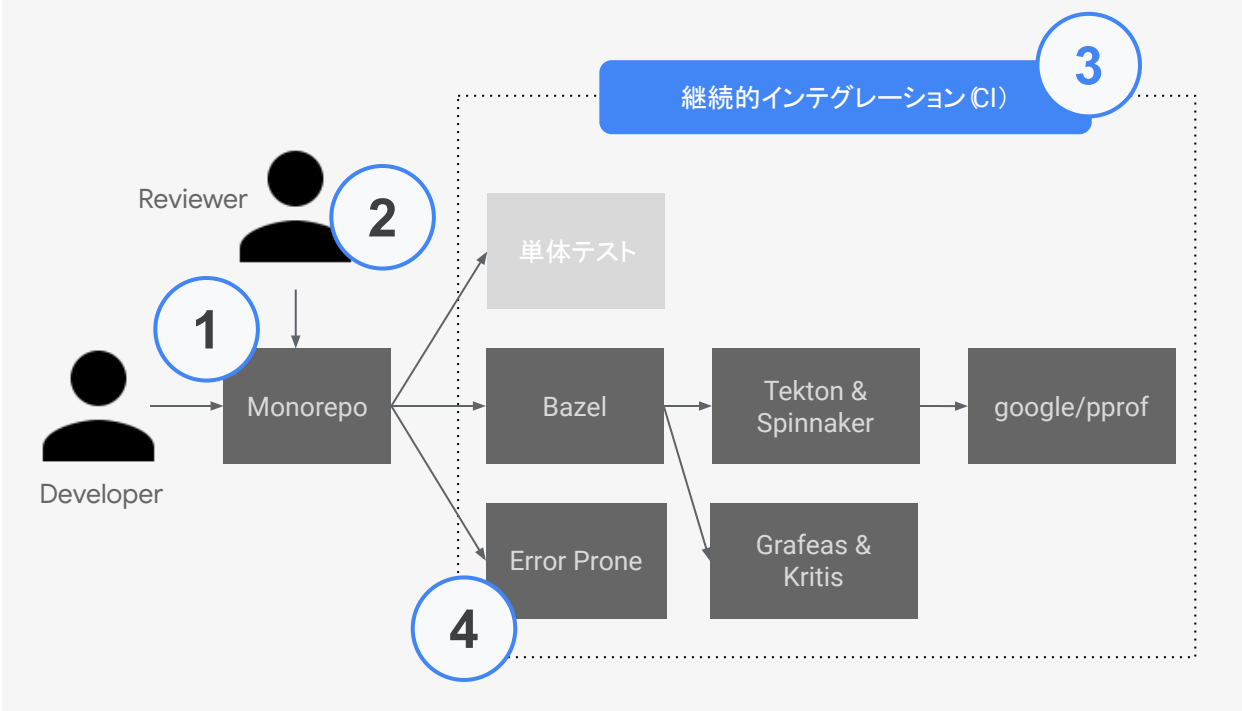

継続的インテグレーション(CI)
継続的デリバリー(CD)
単体テスト
ソースコード
リポジトリ
ビルド
テスト環境へ
デプロイ
インテグレー
ションテスト
本番環境へ
デプロイ
静的解析
脆弱性検出

CI / CD とは


継続的インテグレーション(CI)
継続的デリバリー(CD)
ソースコード
リポジトリ
単体テスト
ビルド
静的解析
テスト環境へ
デプロイ
脆弱性検出
インテグレー
ションテスト
本番環境へ
デプロイ

Google の CI / CD

CI に関する Google の論文や OSS



論文・エッセイ

1. Advantages and Disadvantages of a Monolithic Codebase (2018)

2. Modern Code Review: A Case Study at Google (2018)

3. Taming Google-Scale Continuous Testing (2017)

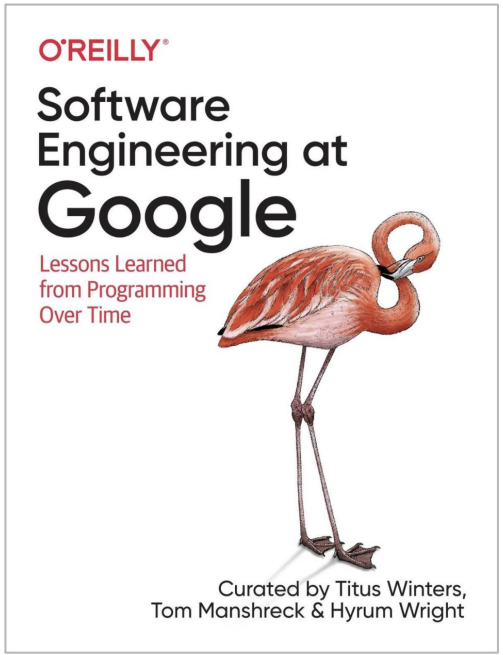
4. Lessons from Building Static Analysis Tools at Google (2018)

CI に関する Google の論文や OSS

	概要	関連 OSS / ガイド
1. Advantages and Disadvantages of a Monolithic Codebase (2018)	モノレポを使うと CI / CD に関する**一連のツールや依存関係が整理され**、バージョン管理や一貫性、品質、エンジニアに対しての透明性など複数の課題が解消される、など。	Bazel
2. Modern Code Review: A Case Study at Google (2018)	「他の開発者が理解できるコードを書かせること」からスタート。Google における 9 百万個変更履歴を分析、その紹介も。**小さな変更を頻繁に行うことで開発効率が向上。**	Code Review Developer Guide
3. Taming Google-Scale Continuous Testing (2017)	ソフトウェアの依存を追跡する仕組み、**とある変更が影響する** すべての**ターゲットを** 非同期に、効率的に並列 **テストし直す仕組み** など。（AFFECTED フラグとマイルストーン）	
4. Lessons from Building Static Analysis Tools at Google (2018)	いくつかの失敗プロジェクトを経て、**ビルドそのものを失敗させる** 仕組みへ。開発者がコードレビューを依頼した際、人間のレビューが入る前に自動ビルドするするという順序が重要。	Error Prone

Software Engineering at Google

Lessons Learned from Programming Over Time



8 章 Style Guides and Rules

9 章 Code Review

11 章 Testing Overview

16 章 Version Control and Branch Management

18 章 Build Systems and Build Philosophy

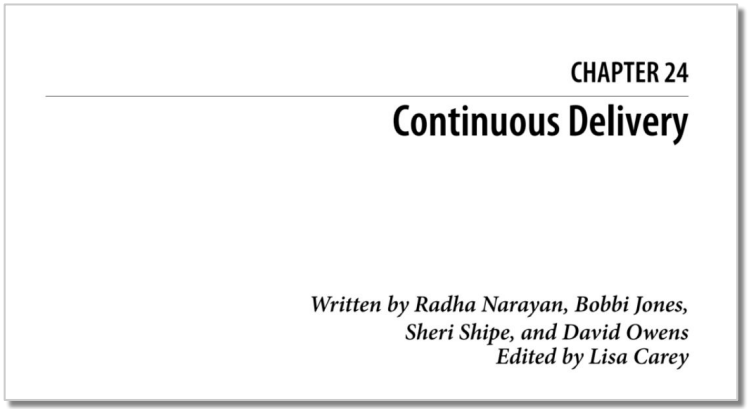
20 章 Static Analysis

23 章 **Continuous Integration**

24 章 **Continuous Delivery**

CI も CD も、まずはフィードバックループ

- 企業が長期的に成功する鍵は
  - アイディアを可能な限り速くユーザーに届け
  - フィードバックに素早く反応すること
- **Launch and iterate:** リリースは早め・頻繁に
  - リリースは終わりではなく学習の始まり
- **Faster is safer**: 小さな変更ほど高い品質となる

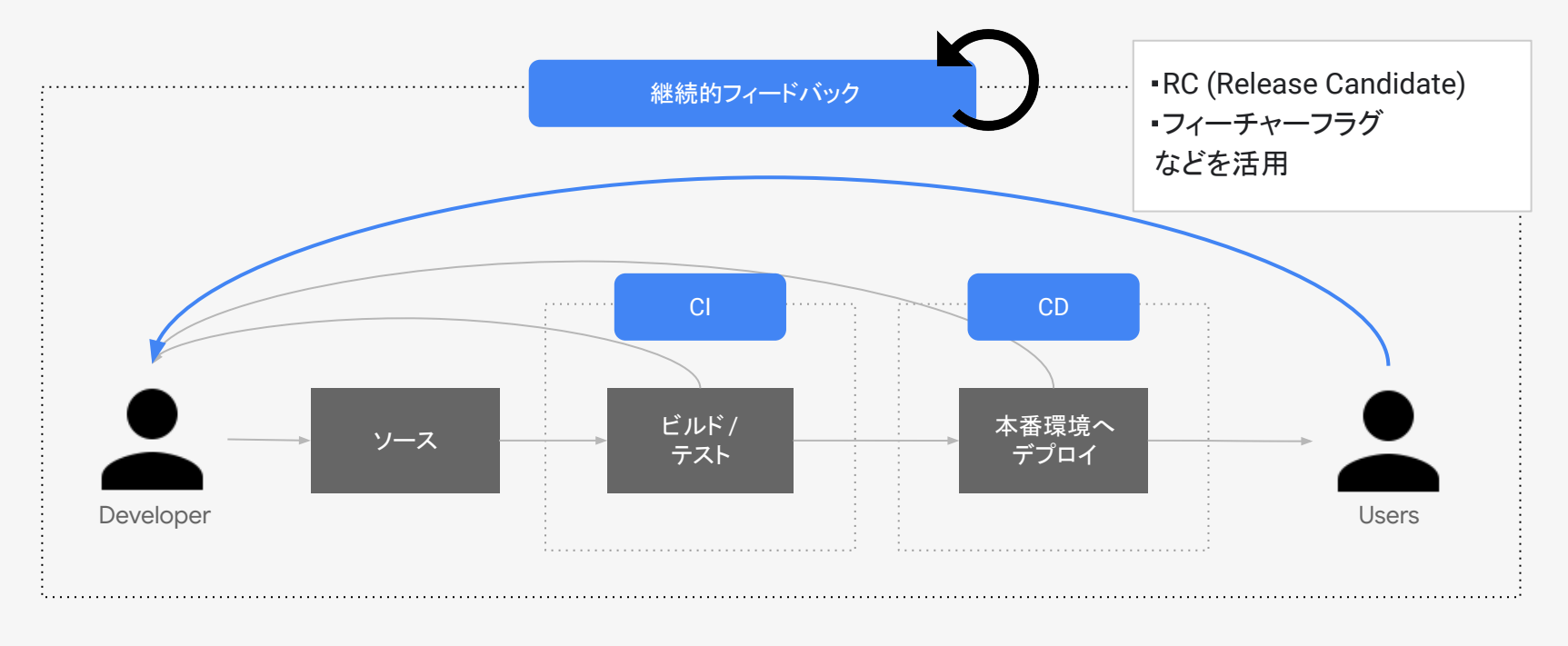

CHAPTER 24

Continuous Delivery

Written by Radha Narayan, Bobbi Jones, Sheri Shipe, and David Owens
Edited by Lisa Carey

参考: Software Engineering at Google

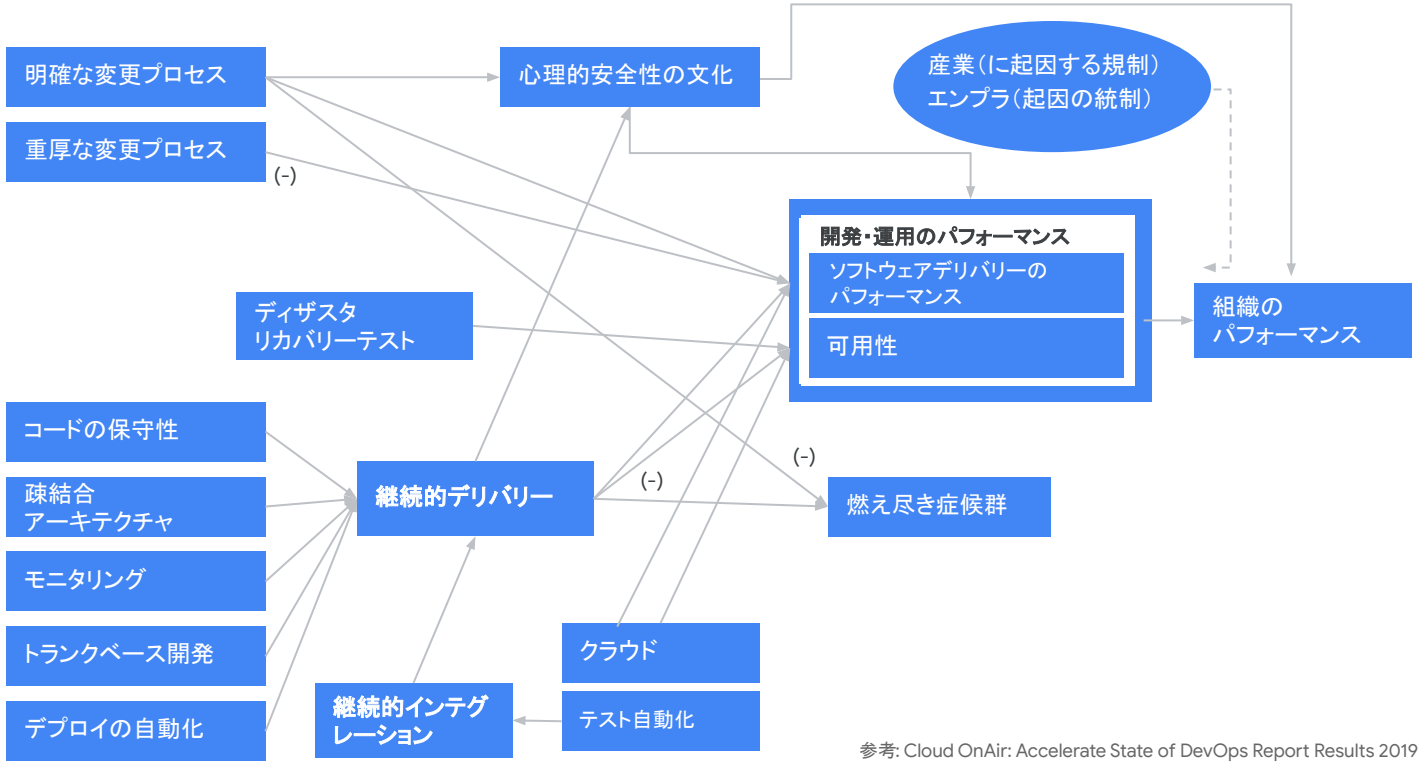
リリース後も重要、フィードバックループ



参考: Software Engineering at Google

何を目的に導入し、どう計測する？

CI / CD 導入の全体像: DORA の "DevOps 研究モデル"



参考: Cloud OnAir: Accelerate State of DevOps Report Results 2019
https://www.youtube.com/watch?v=Fzu5Ry5lU8c

CI / CD の達成度を測る指標の例

CI / CD そのものを継続的に改善するために。

速度

- デプロイ頻度向上
- 変更のリードタイム短縮
- サービス復旧時間の短縮

品質

- 変更時の失敗率低減
- サービスレベル向上
- 欠陥の早期発見

セキュリティ

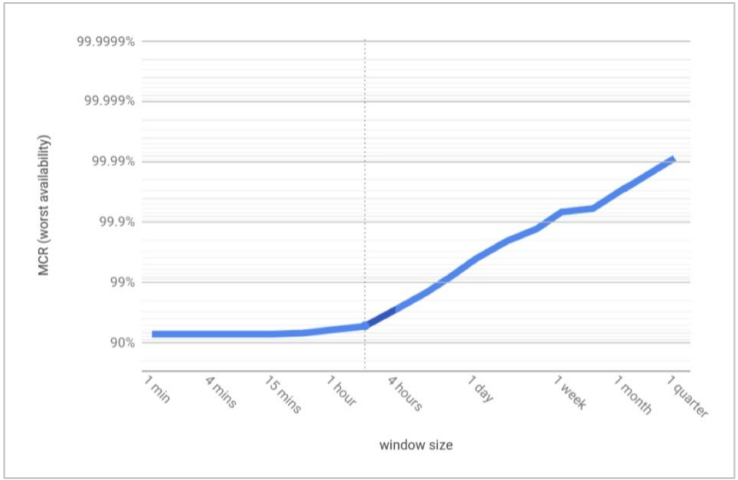
- 脆弱性の検知数
- パッチ適用速度の改善
- 静的解析での発見数

参考: 継続的インテグレーションhttps://cloud.google.com/solutions/continuous-integration?hl=ja
継続的デリバリーhttps://cloud.google.com/solutions/continuous-delivery?hl=ja
The 2019 Accelerate State of DevOps https://services.google.com/fh/files/misc/state-of-devops-2019.pdf
Building Secure & Reliable Systems https://landing.google.com/sre/resources/foundationsandprinciples/srs-book/

サービスレベルの向上: 意味のある可用性とは何か

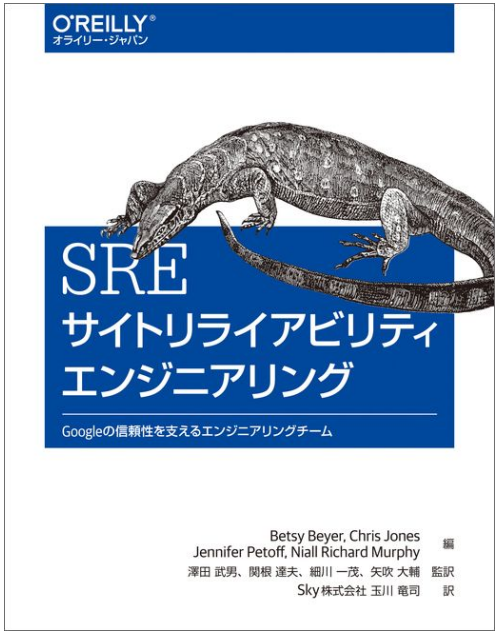
- 2020/02 に G Suite チームが、ユーザーにとっても開発者にとっても意味のある “可用性” とは何かを論文として発表
- *“**優れた可用性の指標は**、ユーザーの経験を捉えた **意味のあるもの** で、彼らの **体験と比例** しており、原因がわかり **対処可能** でなければならない。”*
- 上記要件を満たす指標を模索、結果

$$\mathrm{MCR}(w) \equiv \min_{\substack{\mathbf{w}\subseteq[T_1,T_2] \\ |\mathbf{w}|=w}} \left(\frac{u(\mathbf{w})}{t(\mathbf{w})} \right).$$



参考: Meaningful Availability https://www.usenix.org/system/files/nsdi20spring_hauer_prepub.pdf

Site Reliability Engineering
サイト信頼性エンジニアリング



1 章 イントロダクション

4 章 サービスレベル目標

7 章 Google における自動化の進化

8 章 **リリースエンジニアリング**

17 章 **信頼性のためのテスト**

27 章 大規模なプロダクトの**ローンチにおける信頼性**

https://landing.google.com/sre/books/

CI / CD の設計

Google Cloud

テスト / デプロイの戦略を考える

アプリケーション

マイクロサービスアーキテクチャでは、サービスは独立してバージョン管理されることもあれば、一連のサービスを集めて一緒にバージョン管理されることもある。

バージョン

一緒にデプロイされるべきコードと設定を一意に特定する ID。
"Kubernetes 0.18.2" は "kube-scheduler 937915d" でもある

環境

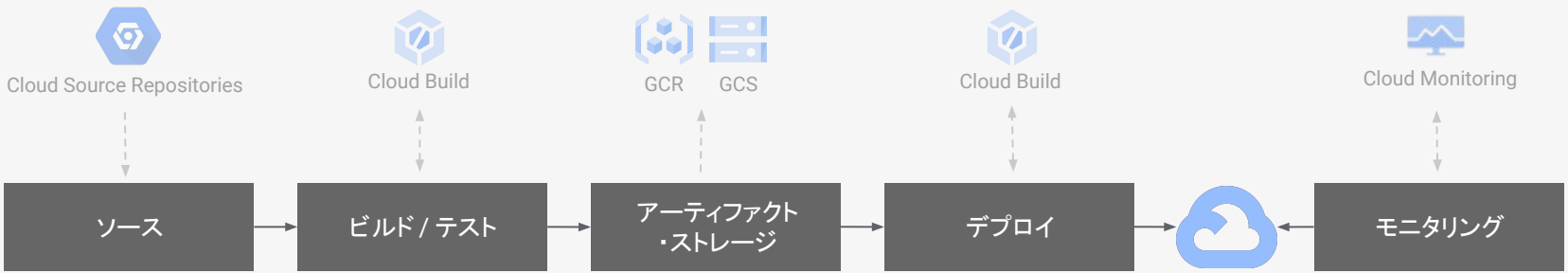
アプリケーションバージョンをデプロイする先となるインフラ、またはサービス。
テスト環境, 本番環境 / VM, コンテナ, Google Play Store ..

テスト / デプロイ戦略

アプリケーション × 環境 ごとにその理想的な戦略は変わる 可能性が高い。

- テストは何を実施？
- デプロイ頻度は？
- どんなポリシーで デプロイする？

テスト / デプロイ戦略に基づき前後の流れも考える



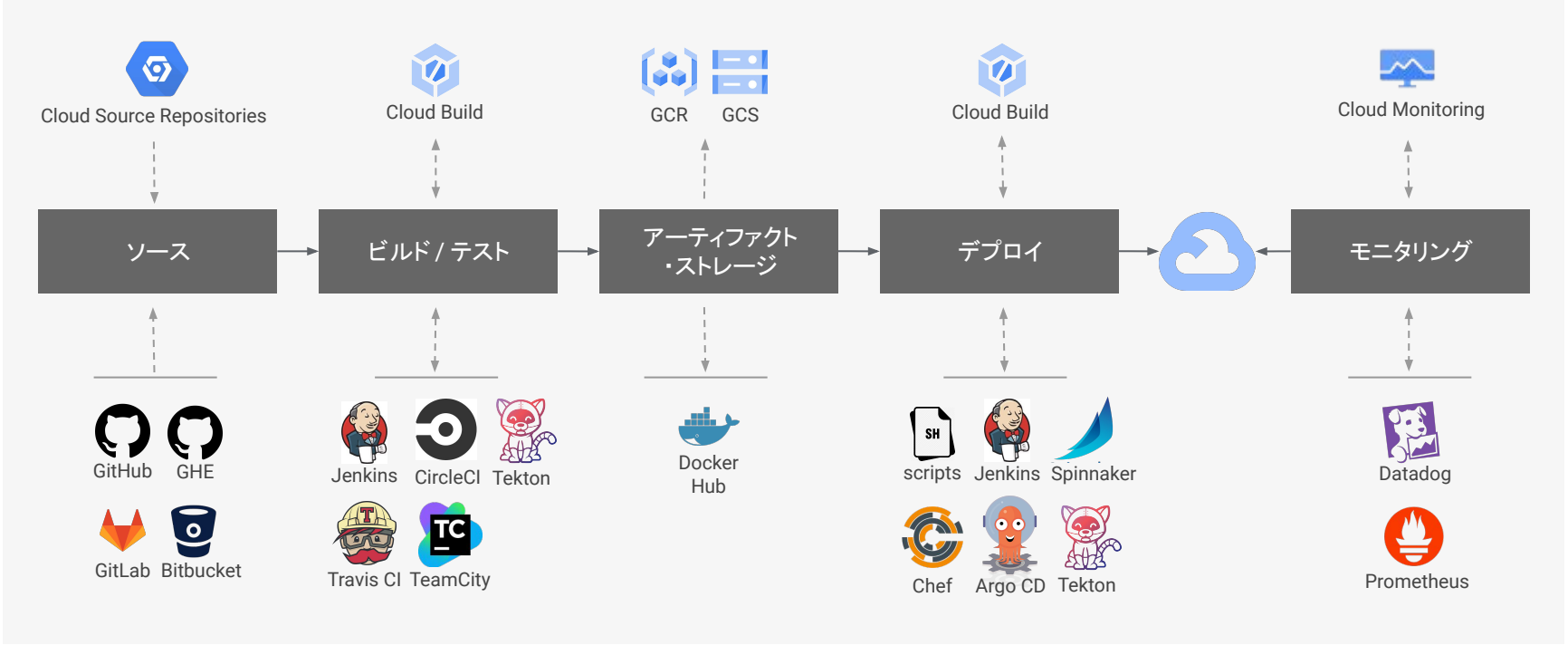
例）環境にデプロイするのはどのバージョンか、何を契機に CI / CD をスタートさせるのか

例）環境にデプロイするアーティファクトはいつビルドしたものか、結合テストは行うのか

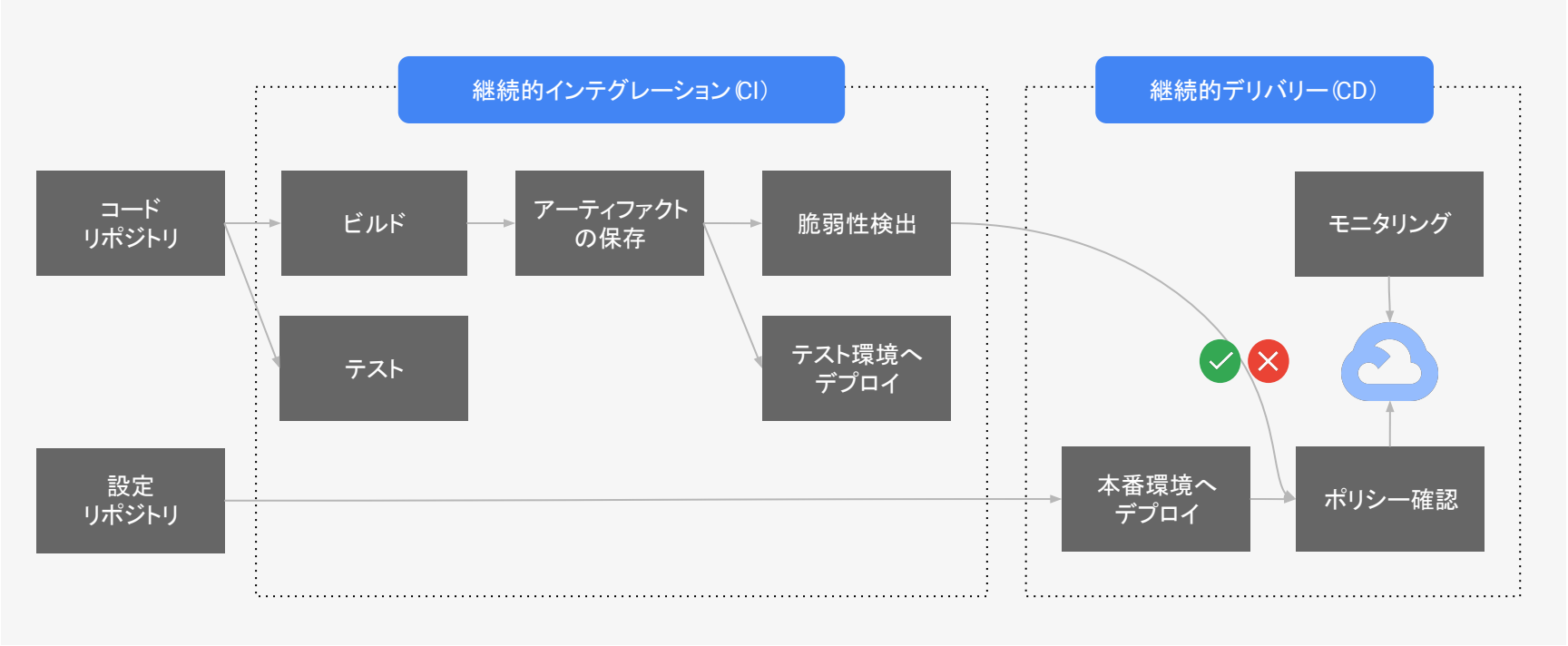
例）バージョン情報はどのようにもつのか、過去のデータはいつまで保持するのか

例）サービスレベルはどう計測するのか、ロールバックするための指標は必要か、CI / CD の達成度はどこで測るか

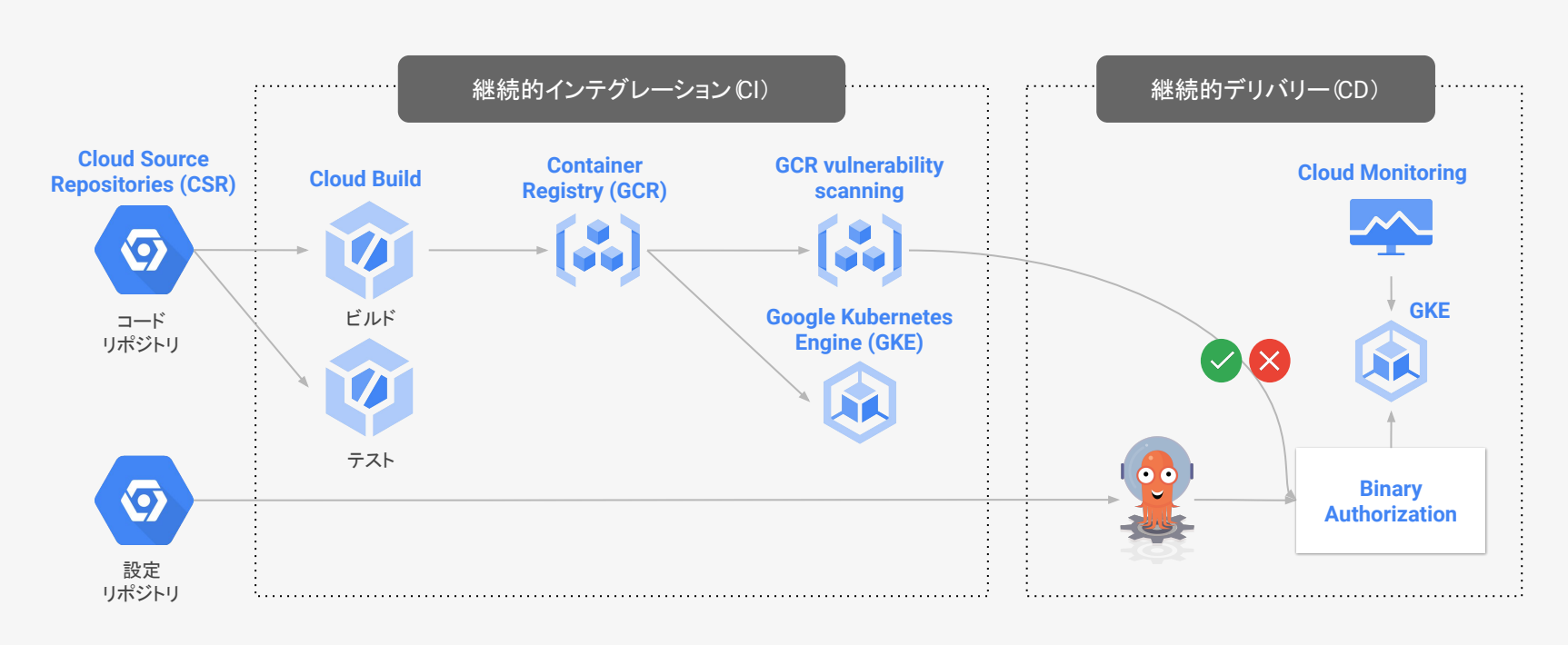
CI / CD 各フェーズでの利用候補例



デプロイ戦略に基づき流れを想定して



それぞれどのツールを使うか検討してみましょう



Google Cloud の CI / CD 支援サービス

Cloud Source Repositories (CSR)

フルマネージドでプライベートなgit リポジトリ

他の GCP 同様の可用性と耐久性

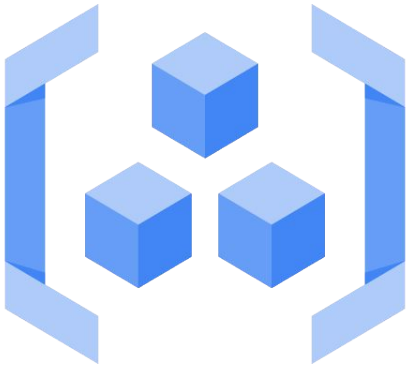
GitHub や Bitbucket からコードをミラーリング

あらゆる CLI ツールをビルドステップとして
組み込むことが可能

GCP サービス群との統合

App Engine や Cloud Functions へのデプロイ
デバッガを併用しアプリをリアルタイムで検査するなど

Google **Container Registry** (GCR)



高速でプライベートなDocker イメージリポジトリ

他の GCP 同様の可用性と耐久性

GCP サービス群との統合

GKE, Cloud Run, GAE, GCE への容易なデプロイ

IAM でのアクセス制御

GCP 内部からはもちろん、外部からも安全なアクセスが可能

Google **Cloud Build**

デベロッパーフレンドリー

CSR、GitHub または Bitbucket での変更をトリガーに

柔軟なビルドステップ

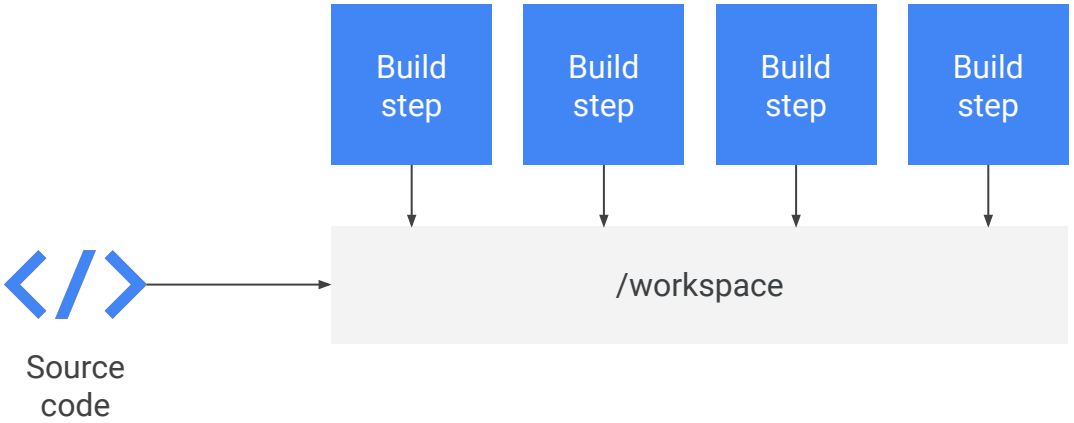
あらゆる CLI ツールをビルドステップとして
組み込むことが可能

フルマネージド CI / CD プラットフォーム

お客様が VM を用意したりキャパシティの管理をする必要はない

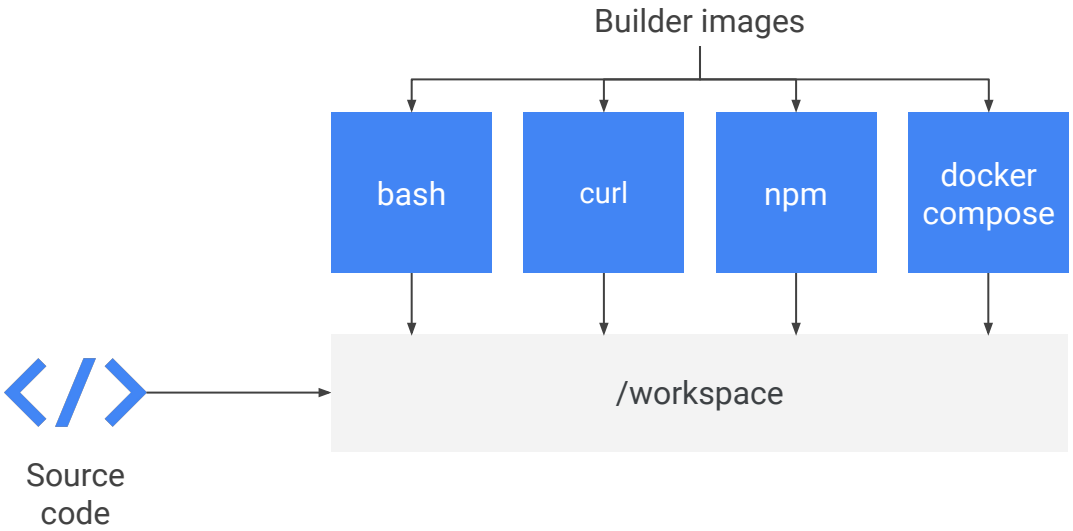
Google **Cloud Build**

ビルドステップとして テスト・ビルド・デプロイ など
柔軟に CI / CD プロセスを実行可能



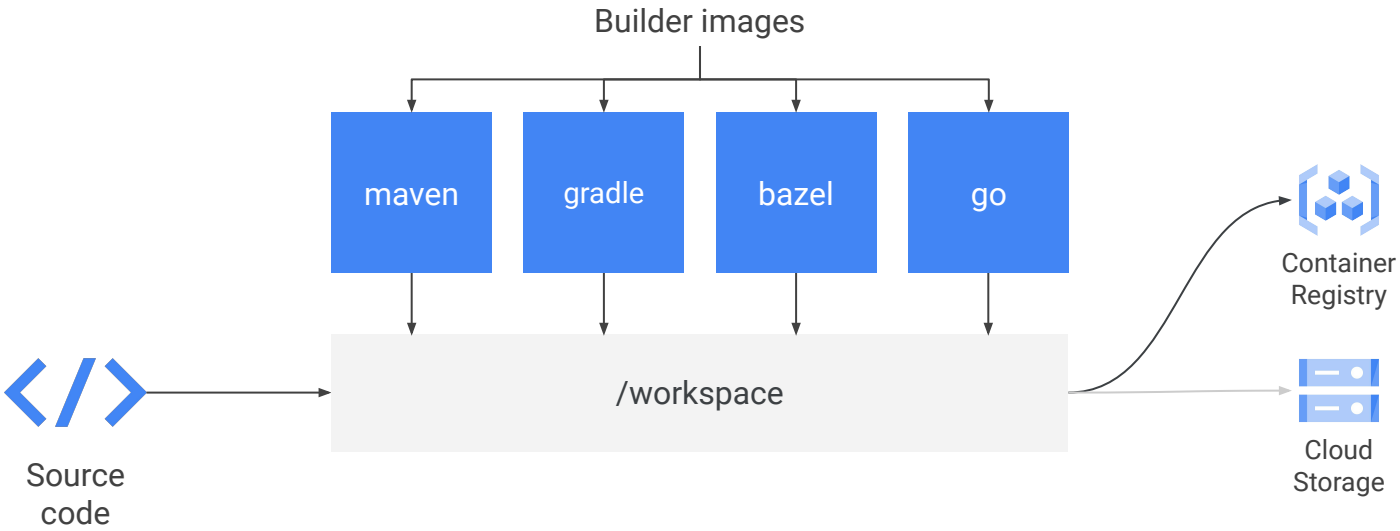
Google **Cloud Build**

テストしたり



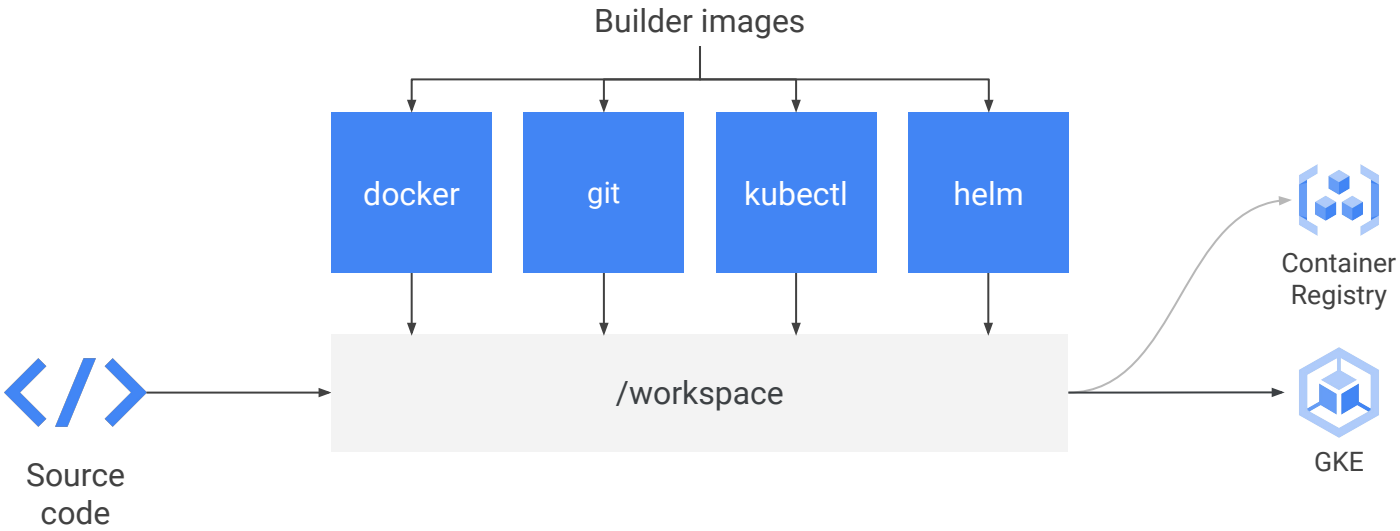
Google **Cloud Build**

ビルドや成果物の保存をしたり



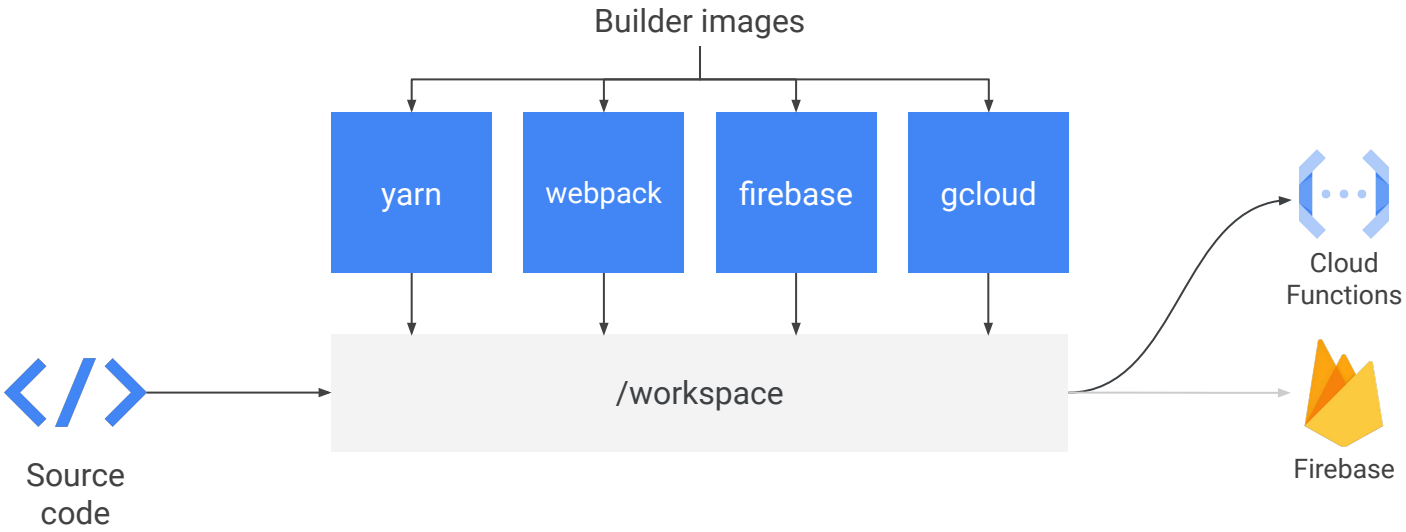
Google **Cloud Build**

Kubernetes にデプロイしたり

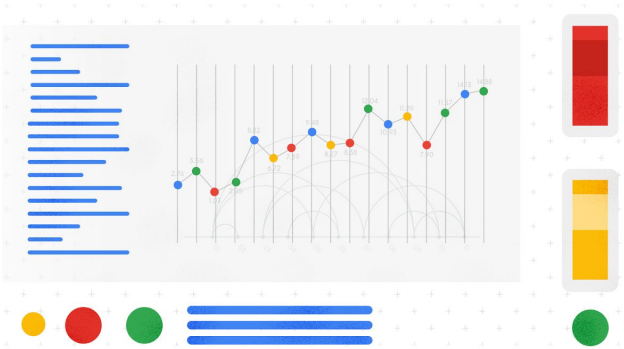


Google **Cloud Build**

サーバーレスにデプロイしたりできます



Operations



高度な故障分析機能

アプリケーション動作環境にまで踏み込んだデバッグや
サービス間のトレーシング

マイクロサービス間の可視化を容易に

Istio との連携によりマイクロサービス間を可視化

統合監視プラットフォーム

Monitoring や Logging、APM など豊富な機能を一括で提供

Operations の主な機能

Cloud Monitoring

Cloud Logging

Cloud Trace

Cloud Debugger

Error Reporting

Cloud Profiler

Cloud Monitoring

分析情報の視覚化、警告の通知

指標、イベント、メタデータを収集し分析します

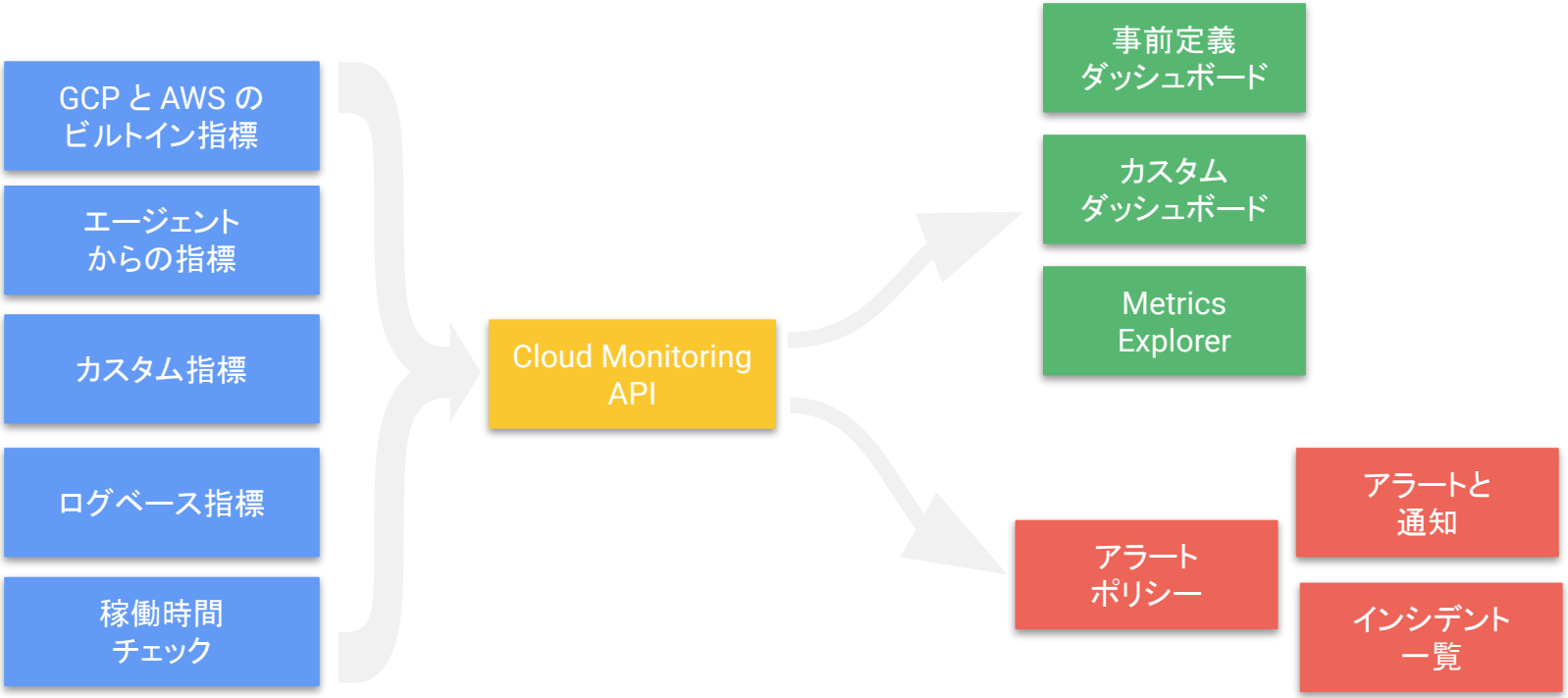
カスタム指標での柔軟な分析

アプリケーションだけでなく**ビジネスレベルのモニタリング** も

GCP サービス群、ロギングなどとの統合

Google Cloud のリソースとサービスを検出し、モニタリング
ログデータに基づいて指標を作成、可視化も容易に

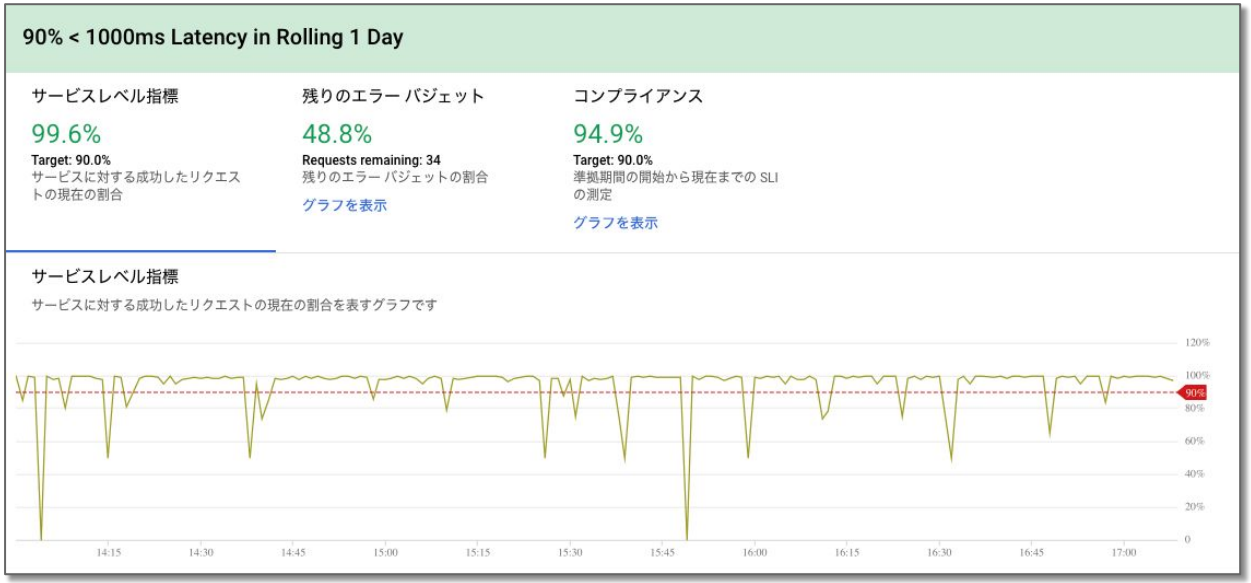
Cloud Monitoring



Beta 機能

サービスレベルのモニタリング

Cloud Monitoring を使えば、**可用性の評価に利用できる**サービスレベルの把握が容易に



Google Cloud

CI / CD の
始め方

Google Cloud

コミュニティを作る

HEATMAP OF DEVOPS TRANSFORMATION STRATEGIES BY PERFORMANCE PROFILE

	Low	Medium	High	Elite
Training Center	27%	21%	18%	14%
Center of Excellence	34%	34%	20%	24%
Proof of Concept but Stall	41%	32%	20%	16%
Proof of Concept as a Template	16%	29%	29%	30%
Proof of Concept as a Seed	21%	24%	29%	30%
Communities of Practice	24%	51%	47%	57%
Big Bang	19%	19%	11%	9%
Bottom-up or Grassroots	29%	39%	46%	46%
Mashup	46%	42%	34%	38%

参考: The 2019 Accelerate State of DevOps
https://services.google.com/fh/files/misc/state-of-devops-2019.pdf

一人で導入を進める必要はない

社内・社外で同士を募る

"実践のためのコミュニティ' が有用！

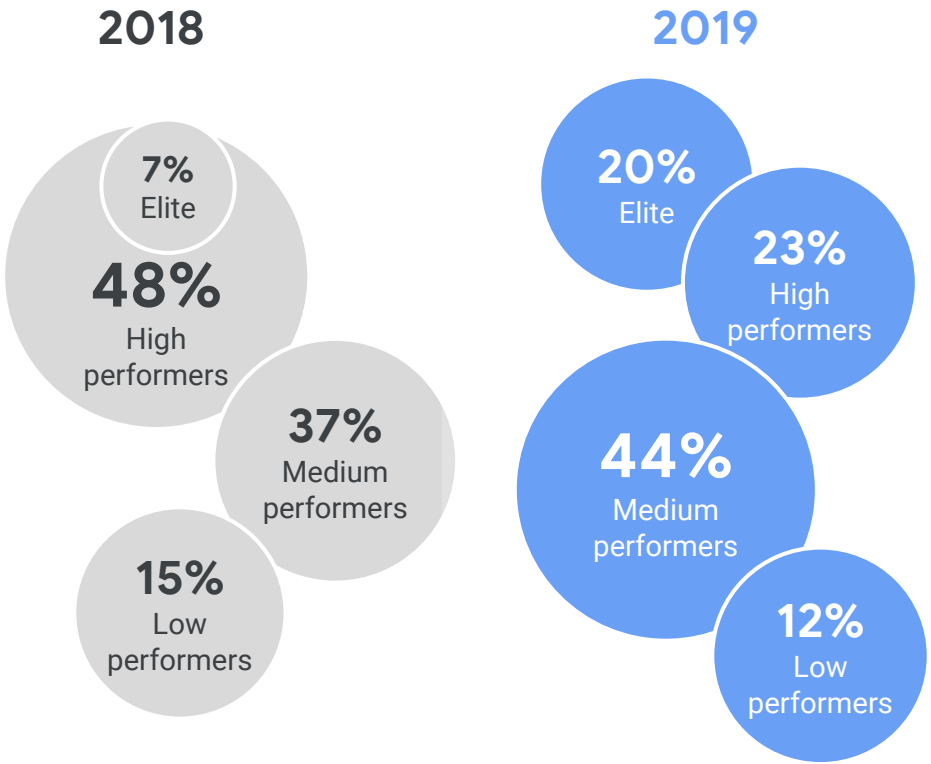
DORA の研究によると PoC も有意義
CoE は新たなサイロになるリスクも。

どこから取りかかるか

- **手作業のリスクは気にしながらも**
  - まずは自動化しようと考えずあれこれ試してみましょう
- **CI から**
  - 単体テスト
  - 静的解析
    - 論文 "Lessons from Building Static Analysis Tools at Google" にみる Google の失敗
    - 開発者のローカルに任せても一向に状況改善せず CI で問題を発見すると、そのビルドを失敗させるようにツールを設定CIを成功させるためには、開発者は必ず問題を解決する必要があるため、小さく始めるのがよい。
- **モニタリングから**
  - サービスレベルの計測
    - Cloud Monitoring を使えば SRE のベストプラクティスを比較的容易に開始できます。

改善し続ける CI / CD

- ハイパフォーマー減少 ..
  - システムの複雑化
  - **改善し続けることが大切**
    - もちろん CI / CD も
- "DevOps" はキャズムを超えました
  - エリート ≧ 20%



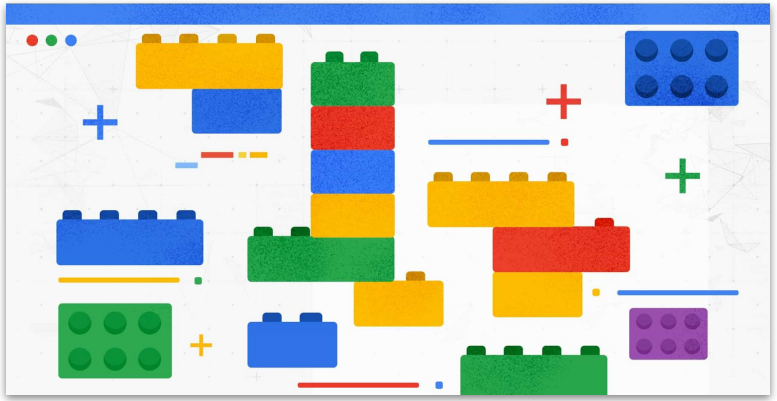
参考: The 2019 Accelerate State of DevOps
https://services.google.com/fh/files/misc/state-of-devops-2019.pdf

Demo

GKE への自動デプロイ

クイックスタート！
数クリックで GKE へのデプロイパイプラインを作る

- 2020/03/06 GKE "自動デプロイ" 機能 リリース
- *"数回のクリックで GKE の継続的デリバリーパイプラインを作成できる自動デプロイ機能。Kubernetes のデプロイに関して Google が蓄積してきた、すぐに使えるベストプラクティスを実装 "*
- メリット
  - エンドツーエンドの追跡が可能
  - プレビューデプロイによるシフトレフト
  - etc..



https://cloud.google.com/blog/ja/products/application-development/automated-deployment-pipelines-come-to-gke

まとめ

04

Google Cloud

まとめ

全体像の把握

CI / CD そのものだけでなく、組織全体として**何を実現したいのか**
またその達成度をどのように計測し、改善し続けるかを考えてみる

参考

DORA, SRE 本, "Software Engineering at Google"

設計

アプリケーションと環境に応じて**テスト / デプロイ戦略**を決め、
ソースの管理からサービスモニタリングまでどうするかを考える

Cloud Build での CI / CD,
Operations での評価指標

始め方

社内外に仲間を作り、**できるところから**始め、サービスレベルや CI / CD の成果指標が改善されるよう、**継続的に**取り組む

DORA, Google Cloud 各種
動画 / ドキュメント

Next Steps

- **DORA** の DevOps クイックチェックで現状分析をしてみましょう！
  https://cloud.google.com/devops?hl=ja

- 動画、記事、クイックスタートなど CI / CD について豊富な情報があります。ご覧ください
  https://cloud.google.com/docs/ci-cd?hl=ja

- Qwiklabs にもオンラインの ハンズオン があります。ぜひお試しください
  - Continuous Delivery with Jenkins in Kubernetes Engine（中級）
  - Continuous Delivery Pipelines with Spinnaker and Kubernetes Engine（上級）

次回セッションのご紹介

- 次回テーマは Cloud Run
  - 日程: 2020/06/12 20:00〜
  - Java にフォーカス！
- 今回触れなかった、Binary Authorization などセキュリティについては 06/19 の予定です

Thank you