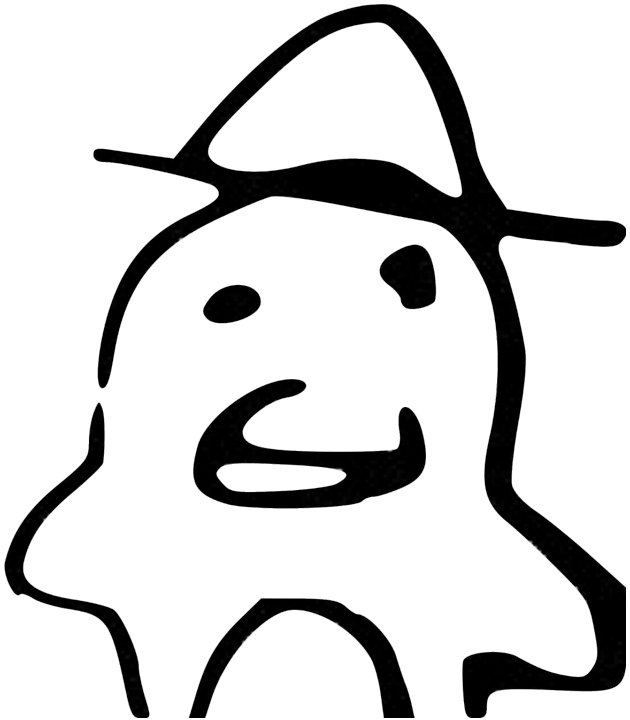
Google Cloud

はてな広告配信システムの クラウドネイティブ化への道のり

徳網 亮輔
株式会社はてな
ビジネスプラットフォームチーム
アプリケーションエンジニア / テックリード



Hatena

徳網 亮輔

id:pokutuna

アジェンダ

- はてなと広告配信システムについて
- 移転前のシステムと課題
- Google Cloud 移転の道のり
  - 1. 技術選定・構築
  - 2. レポート集計の実装と検証
  - 3. 負荷試験
  - 4. 本番環境の切り替え
  - 5. 移転後の暮らし

はてなの事業

Hatena

コンテンツプラットフォーム

ユーザーがコンテンツを発信・拡散できるプラットフォームを提供

- Hatena Blog
- はてなブックマーク
- 人力検索はてな

コンテンツマーケティング

良質なコンテンツ発信を通したマーケティング活動の支援

- HATENA BLOG Media
- はてな MediaSuite
- ネイティブ広告

テクノロジーソリューション

サービス提供を通じて培った技術やノウハウをソリューションとして展開

- mackerel

受託・協業サービス

- イカリング２
- カクヨム
- ジャンプルーキー！
- あしたのヤングジャンプ

はてなの事業

Hatena

コンテンツプラットフォーム

ユーザーがコンテンツを発信・拡散できるプラットフォームを提供

Hatena Blog

はてなブックマーク

人力検索はてな

コンテンツマーケティング

良質なコンテンツ発信を通したマーケティング活動の支援

HATENA BLOG Media

はてな MediaSuite

ネイティブ広告

広告配信

テクノロジーソリューション

サービス提供を通じて培った技術やノウハウをソリューションとして展開

mackerel

受託・協業サービス

- イカリング２
- カクヨム
- ジャンプルーキー！
- あしたのヤングジャンプ

はてなブックマークとは

国内最大のソーシャルブックマークサービス

ユーザーのブックマーク情報を元にネットで話題のコンテンツが抽出される

1

ユーザーが気になるコンテンツを
ブックマーク（保存）する

2

ブックマーク数を参考として、
インターネット上で話題の
コンテンツが集まる

3

旬な話題を見に来るユーザーが、
コンテンツを拡散させていく

はてなのネイティブ広告

自然な誘導枠

自然な誘導先

広告配信システムの要件

1. 広告入稿

- 広告枠の編成・予約
- 広告の入稿・審査
- アカウント

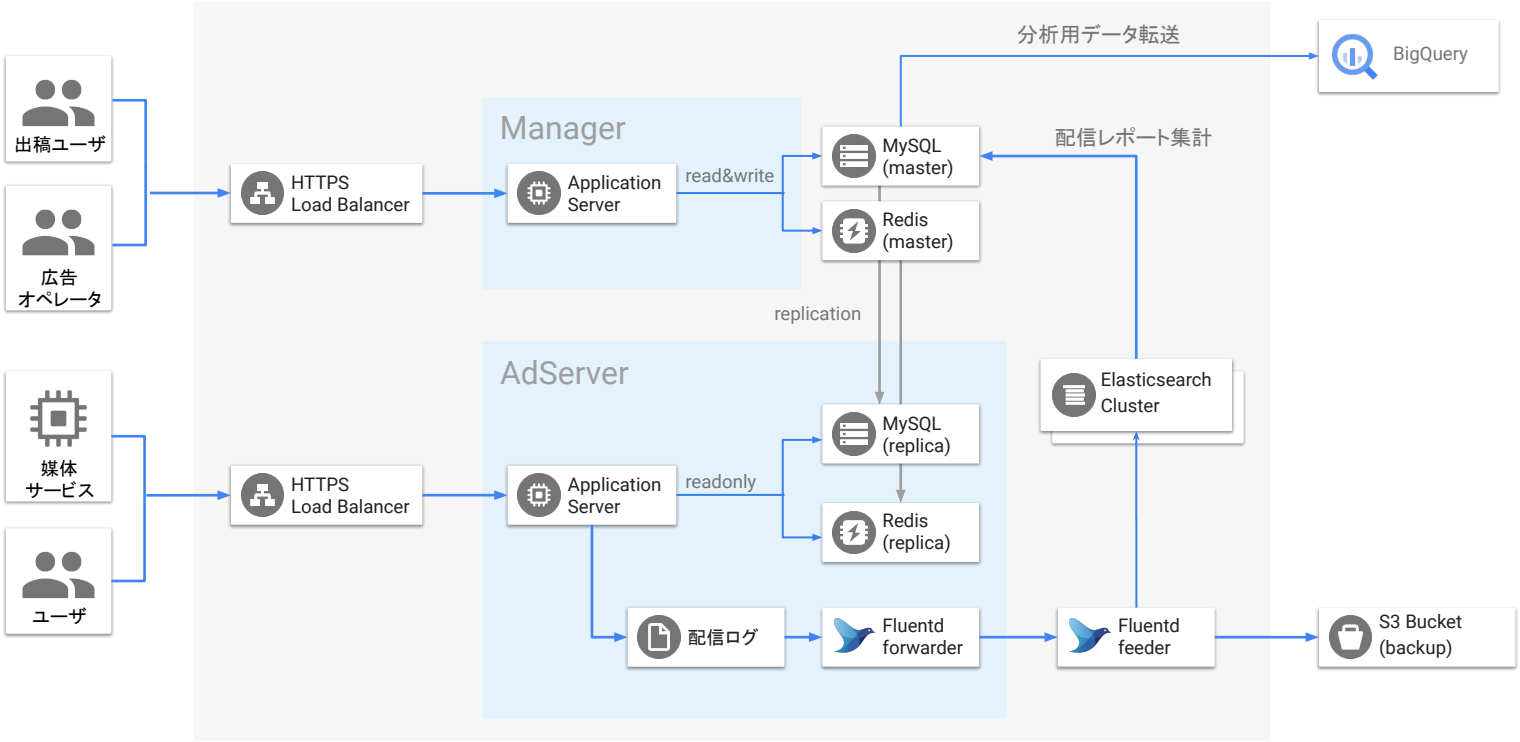
2. 広告配信

- 低レイテンシ
- キャッシュ

3. レポート

- 配信結果の確認
- 独自のコンバージョン

システム構成(移転前)

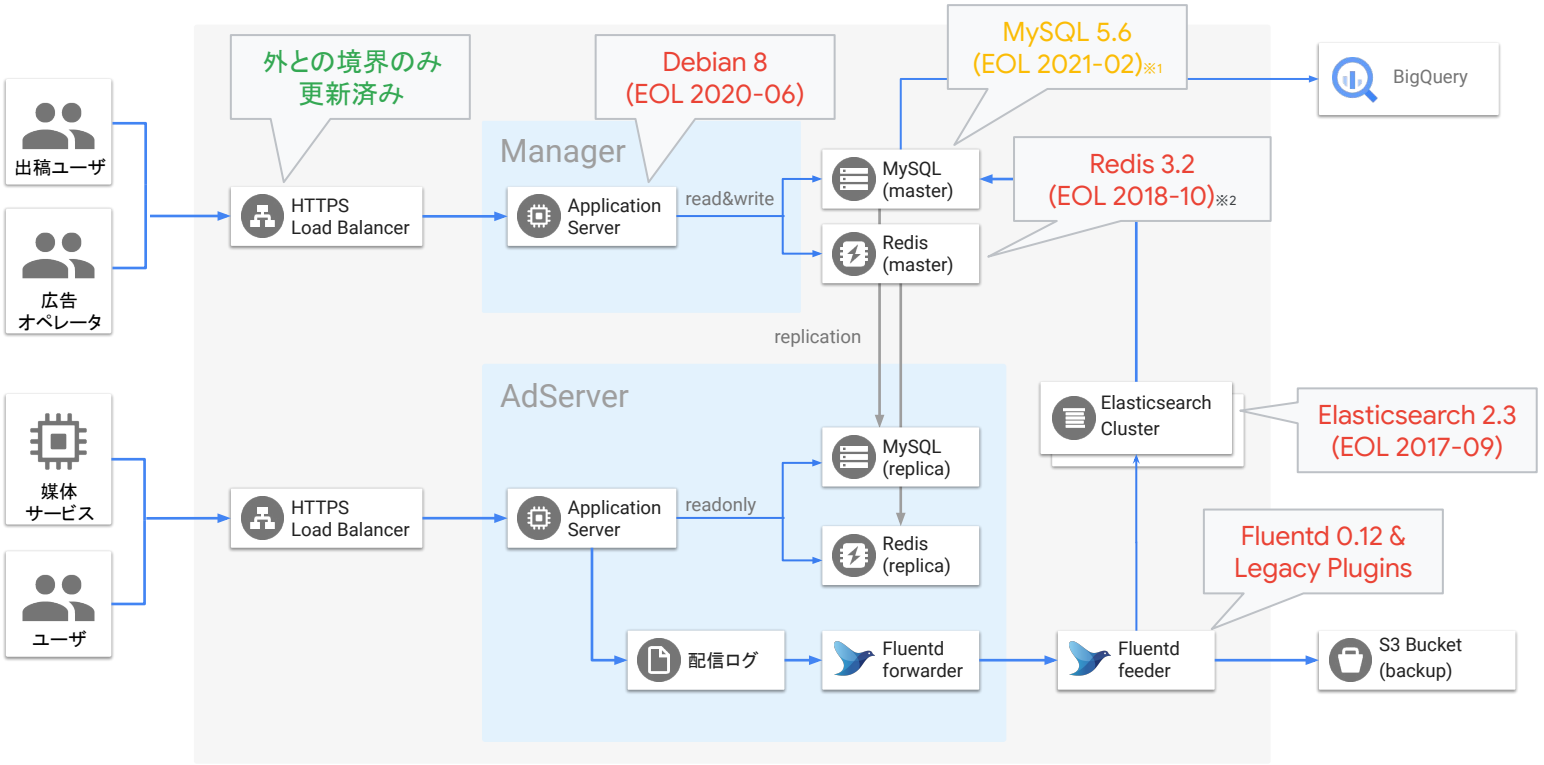

分析用データ転送
BigQuery
出稿ユーザ
広告
オペレータ
HTTPS
Load Balancer
Manager
Application
Server
read&write
MySQL
(master)
Redis
(master)
配信レポート集計
replication
AdServer
媒体
サービス
ユーザ
HTTPS
Load Balancer
Application
Server
readonly
MySQL
(replica)
Redis
(replica)
Elasticsearch
Cluster
配信ログ
Fluentd
forwarder
Fluentd
feeder
S3 Bucket
(backup)

システム構成(移転前)

- Manager と AdServer
- データストアは MySQL と Redis
  - AdServer はレイテンシ軽減のためローカルに replica を持つ
- 配信ログを Elasticsearch に収集してレポートを集計
  - impression(広告表示), click (広告クリック), conversion (目標達成) のログ

- 2016 年に実装された Perl アプリケーション
- 契約データセンターで稼働
  - Xen による仮想マシン + chef で構築

時は 2020 年



※1: MySQL 5.6 Community Edition EOL
※2: Redis 5 のリリースによりサポート対象から外れる

Google Cloud へ移転を決意

- バージョンアップの手間を減らしたい
  - マネージドサービスに寄せる
  - アプリケーション実行環境のコンテナ化
- Elasticsearch を BigQuery に置き換えたい
  - Elasticsearch のマネージドサービスは費用面で厳しい
  - データ分析を低コストに
- クラウドネイティブ化

Google Cloud 移転プロジェクト

開発メンバー

アプリケーション
エンジニア 1 名

SRE 2 名 (兼務)

開発期間

2020-06 ~ 2021-07

2021-06-27 リリース

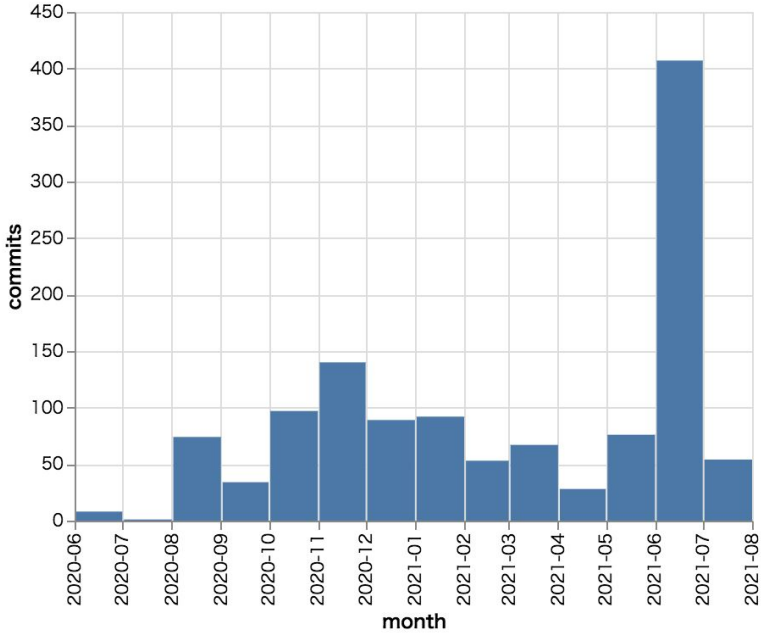
コミット数

1,220 commits

265 pull requests

1年プロジェクト？

- 他に優先度の高いプロジェクトと平行
  - 合間に進める
- SRE 1名の退職予定を機に加速
- 4~5ヶ月の規模感



Google Cloud 移転の道のり

1. 技術選定・構築

- アプリケーション実行環境
- Perl から Google Cloud を使う
- CI/CD

2. レポート集計の検証と実装

- 配信ログを先行して転送
- 集計のユニットテストをBQ で
- Logging シンクに任せる選択
- コンバージョンイベントの伝播

3. 負荷試験

- Locust による負荷試験
- レプリカ同居構成の見直し

4. 本番環境の切り替え

- AdServer だけ無停止作戦
- 切替時のレポート集計

移転完了

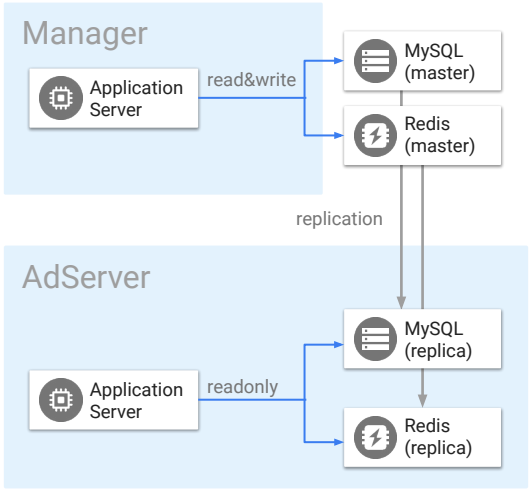
5. 移転後の暮らし

- 分析環境の向上
- SLO ダッシュボード
- エラー収集

アプリケーション実行環境

- Google Kubernetes Engine を採用
- k8s のやり方にのっかる
  - Cloud Scheduler vs. CronJob
- 必要なら複雑な構成も可能

- データストアは
  Cloud SQL & Memorystore を採用

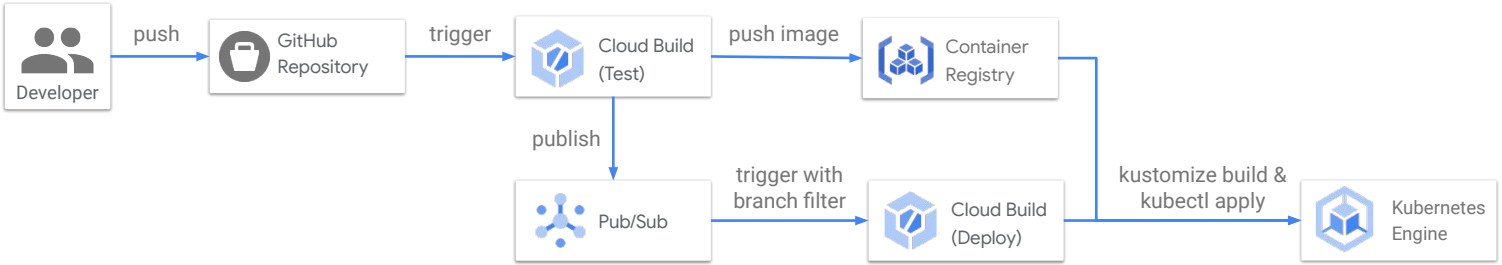


Perl から Google Cloud を使う

- Perl に公式のクライアントライブラリは無い
- 認証は Application Default Credentials フローを実装
  - ローカルでは gcloud の認証情報
  - Google Cloud 上では Metadata Server (Workload Identity)
- REST API リファレンスを読み必要な部分だけ自前で実装
  - BigQuery クライアント
  - Pub/Sub push の JWT の検証

CI / CD

- Google Cloud Build で実装
  - Spinnaker や ArgoCD、Tekton を検討したが見送り
  - 当時 Cloud Deploy (2021/09 Preview) の噂を聞いた
- 素朴な GitOps




1. 技術選定・構築
2. レポート集計の
検証と実装
3. 負荷試験
4. 本番環境の
切り替え
5. 移転後の暮らし

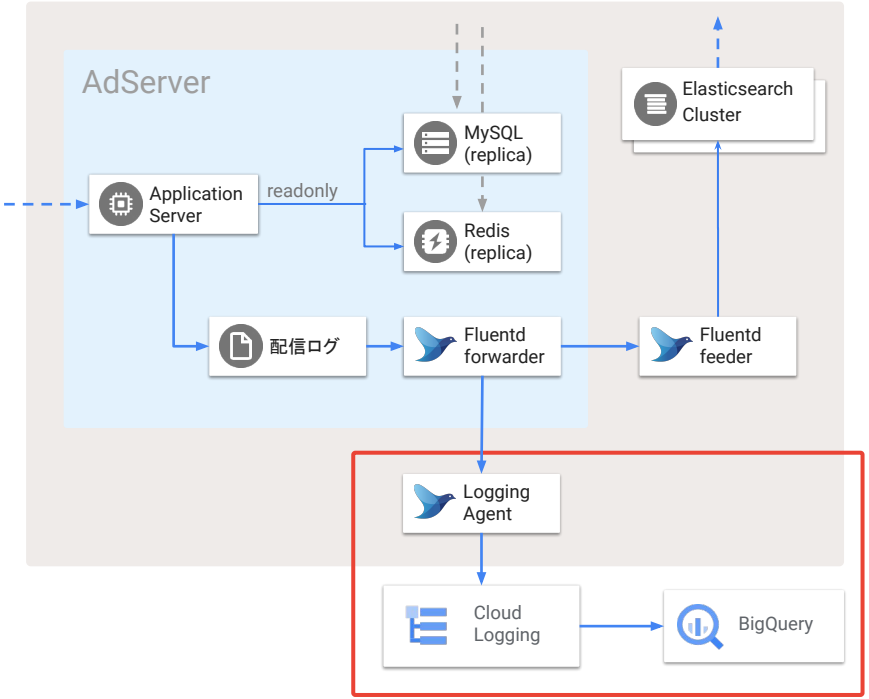
レポート集計の移転

- 広告における重要機能
- Elasticsearch から BigQuery への置き換え
  - 集計の再実装
- 本番環境の切り替え時に不整合が起きないように

広告配信ログを先行して転送

- Logging Agent を立てたホストに配信ログを forward
  - 既存環境の変更を最小限に
- Cloud Logging のログルーターで BigQuery へ転送

- 実際の配信結果を使って Elasticsearch と BigQuery の集計結果を比較しつつクエリを実装



既存のユニットテストを BigQuery で実行

- 従来はローカルの Docker 上で Elasticsearch のテストを実行していた

- テストプロセスごとにログバッファを用意
- 集計クエリ実行メソッドをフック
  - WITH 句でログテーブルを再現
  - 集計テーブル名を置換

既存のテストがそのまま BigQuery で通るように

実行クエリ例

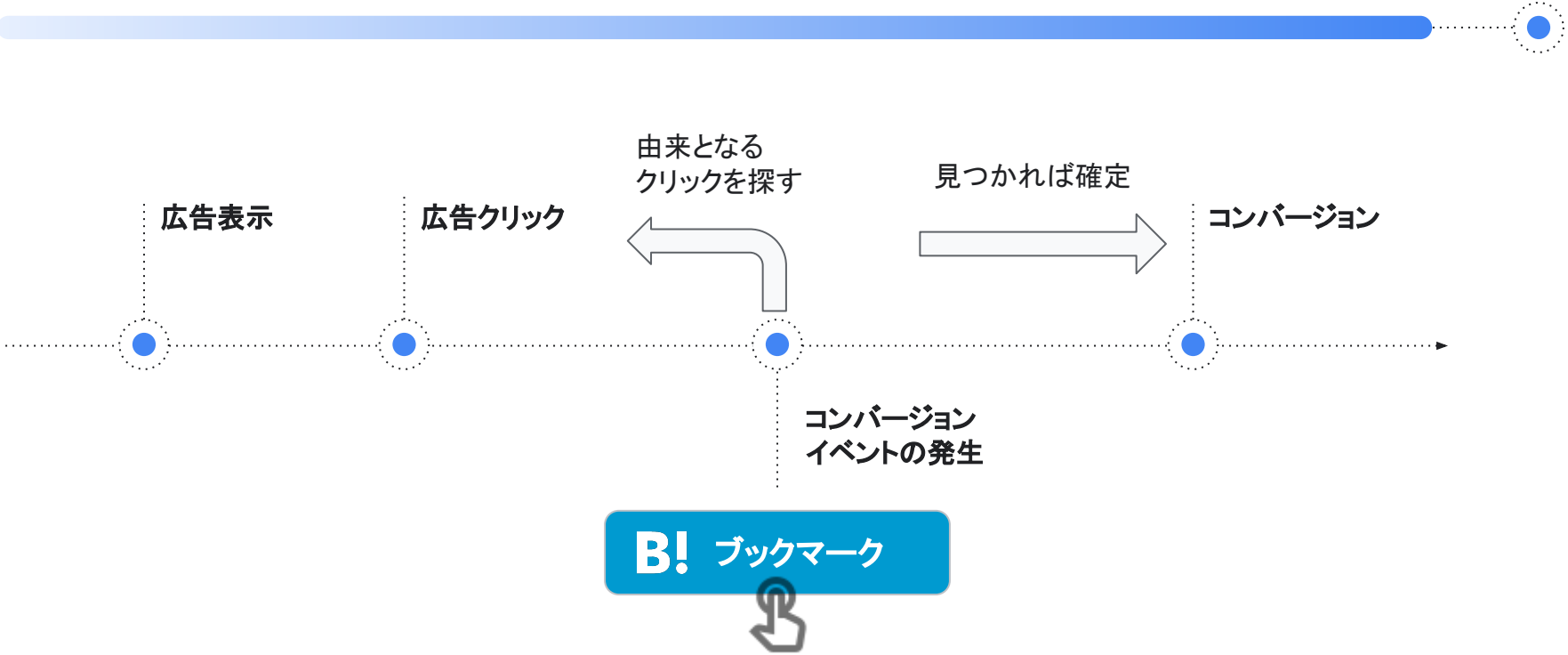
```
WITH log_buffer AS (
  # ログバッファに書かれたJSON ログをSQL に詰める
  SELECT * FROM UNNEST([
    '{"logged_time":"1645349221","_type":"impression","_id":"ABC","creative":{"id":"123", "url":"https://example.com/1"}, ...}',
    '{"logged_time":"1645349222","_type":"impression","_id":"DEF","creative":{"id":"456", "url":"https://example.com/2"}, ...}',
    ...
  ]) AS line
), adserver_logs AS (
  # 配信ログテーブルのスキーマにあわせてJSON を展開
  SELECT
    TIMESTAMP_SECONDS(CAST(JSON_VALUE(line, "$.logged_time") as INT64)) AS timestamp,
    STRUCT<
      _type STRING,
      _id STRING,
      creative STRUCT<id STRING, url STRING>,
      ...
    >(
     JSON_VALUE(line, "$._type"),
     JSON_VALUE(line, "$._id"),
     STRUCT(JSON_VALUE(line, "$.creative.id"), JSON_VALUE(line, "$.creative_redirect_to")),
     ...
    ) AS jsonPayload,
    ...
  FROM log_buffer
)

# レポート集計クエリ
SELECT ...
FROM adserver_logs # テーブル名を置換
WHERE ...
```

Logging から BigQuery へのシンクを活用する

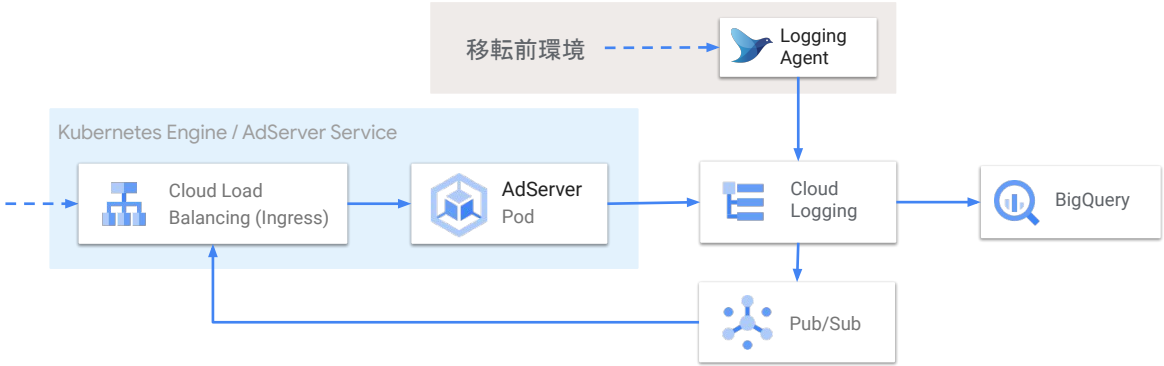
- 広告配信ログを stdout に書く
  - GKE の DaemonSet にある fluentbit が Logging へ転送
  - JSON を書くと jsonPayload 以下に入る
- BigQuery へ転送するシンクを作成
  - テーブル名やスキーマは自動で決まる
    - ログ JSON 中の型を一貫させておく
    - 数値はデフォルトで FLOAT64 になる
  - ログ配送経路を意識しなくてよくなる

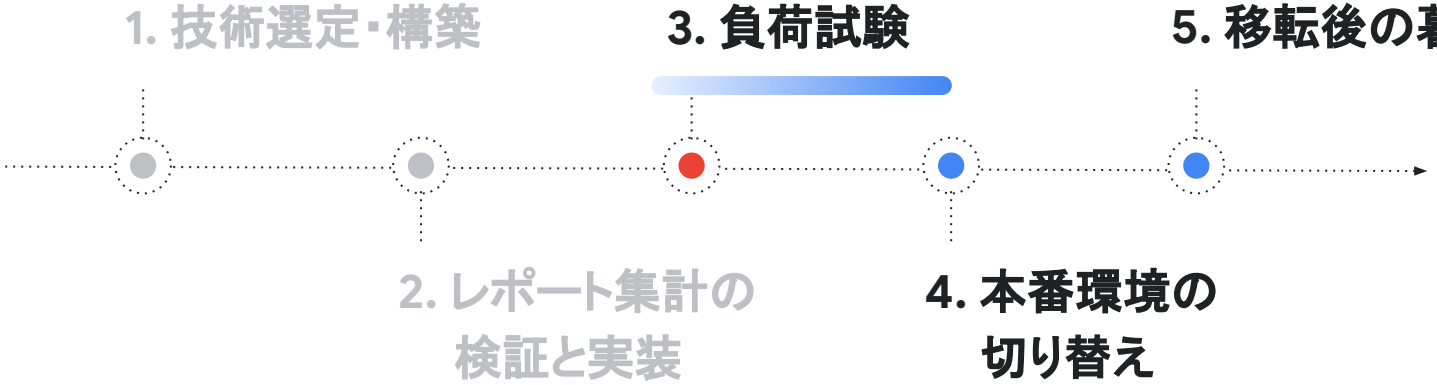
コンバージョンの集計


広告表示
広告クリック
由来となる
クリックを探す
見つかれば確定
コンバージョン
コンバージョン
イベントの発生
B! ブックマーク

コンバージョンイベントの伝播

- 移転までAPIリクエストが来ない＆切替え時のロストがある
- Logging → Pub/Sub で移転前から移転先へフィードバック



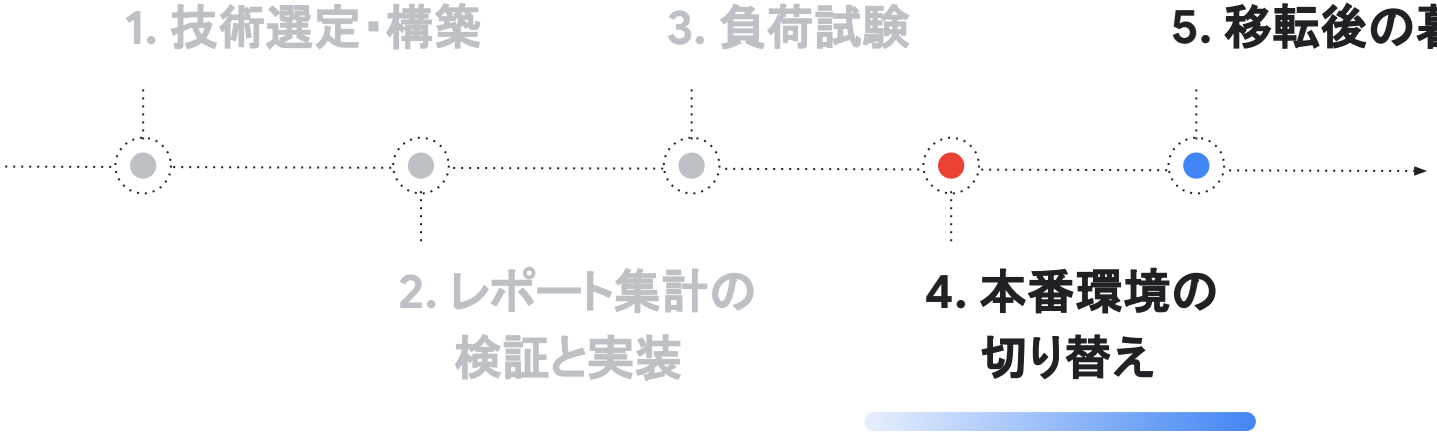

1. 技術選定・構築
2. レポート集計の
検証と実装
3. 負荷試験
4. 本番環境の
切り替え
5. 移転後の暮らし

Locust による負荷試験

- 本番環境と同等の構成のGKE クラスタで負荷試験
- Manager に負荷試験用 API を実装
  - 負荷試験用の初期データを返し、シナリオ側で利用
- アプリケーションイメージを切り替えつつ負荷試験
  - プロファイラの有無
  - Pod 内に Redis レプリカを置く/ 置かない
- サーバーパラメータを調整しチューニング

レプリカ同居構成を見送り

- Redis レプリカ同居の有無を比較
  - 10~20% 程度のレスポンスタイムの改善
- アーキテクチャのシンプルさを選ぶ
  - 同居なしで平均 15ms 程度、改善幅は大きくない
  - MySQL の同居はそれ以下の見立て


1. 技術選定・構築
2. レポート集計の
検証と実装
3. 負荷試験
4. 本番環境の
切り替え
5. 移転後の暮らし

本番環境の切り替え

- AdServer は無停止 / Manager を停止
  - 休日はほぼアクセスなし
  - 入稿や配信設定がリリース作業中にされなければよい
- レポート集計
  - しばらく新旧 2 つのログテーブルをUNION ALL して集計
- DNS レコードの切り替えによるリリース
- ロールバック手段も用意しておく
  - Logging のログを Elasticsearch に入れる形式に変換するDataflow

事前準備

- Google Cloud 環境リリース
- レポート集計 & コンバージョン記録を両環境で並行稼動
- DNS TTL を短く

従来環境リリース

- 広告配信設定を変更できないように & メンテナンス表示

データベースのコピー

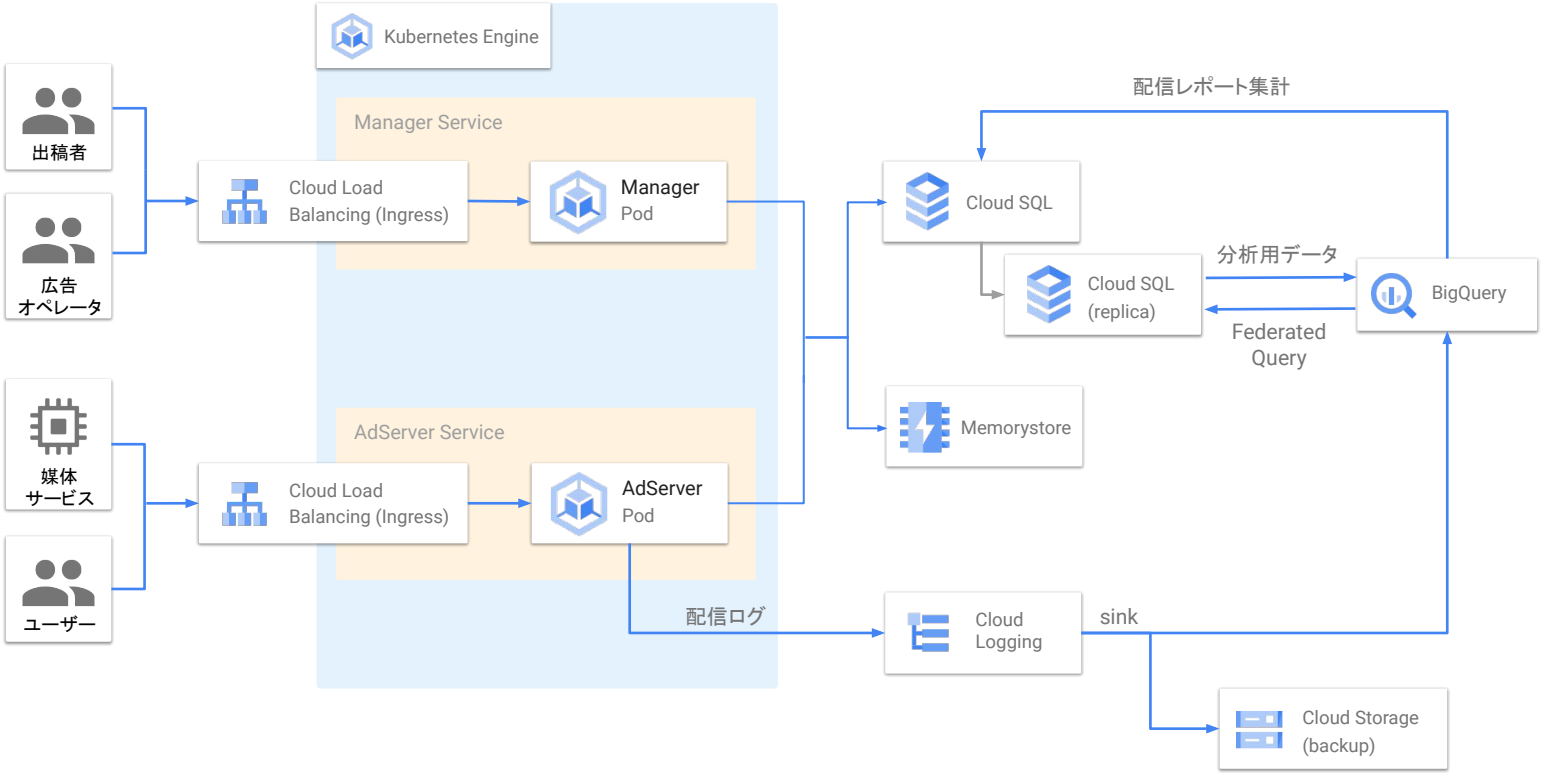
DNS レコードを更新

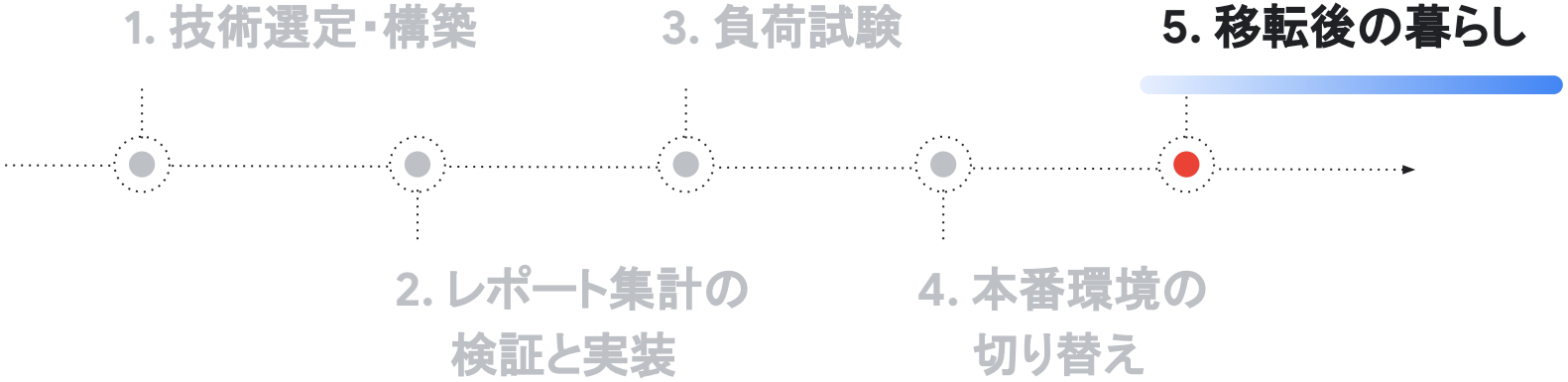
- しばらく両環境にリクエストがくる

Google Cloud 環境リリース

- メンテナンス表示を消す

移転後の構成


Kubernetes Engine
Manager Service
AdServer Service
出稿者
広告
オペレータ
媒体
サービス
ユーザー
Cloud Load
Balancing (Ingress)
Cloud Load
Balancing (Ingress)
Manager
Pod
AdServer
Pod
配信レポート集計
Cloud SQL
Cloud SQL
(replica)
分析用データ
Federated
Query
BigQuery
Memorystore
配信ログ
Cloud
Logging
sink
Cloud Storage
(backup)


1. 技術選定・構築
2. レポート集計の
検証と実装
3. 負荷試験
4. 本番環境の
切り替え
5. 移転後の暮らし

よりリアルタイム & 低コストなデータ分析

- Federated Query で Cloud SQL レプリカを BigQuery から参照
  - BigQuery テーブルの洗い替えにも便利
- Google データポータルのダッシュボードがよりリッチ& リアルタイムに
- Google Spreadsheet コネクテッドシートの活用

オーダー ❶ ▲	集計開始 ❷ ▲	集計終了	終了	想定imp	達成imp	達成率
【21年4-6月期以降】はてなブックマークアプリ/ネイティブ広告枠	2022/02/22 11:00	2022/02/23 11:00	true	300,000	344,858	115.0%
【21年4-6月期以降】はてなブックマークアプリ/ネイティブ広告枠	2022/02/21 11:00	2022/02/22 11:00	true	300,000	356,784	118.9%
【21年4-6月期以降】はてなブックマークアプリ/ネイティブ広告枠	2022/02/22 11:00	2022/02/23 11:00	true	300,000	344,266	114.8%
【21年4-6月期以降】はてなブックマークアプリ/ネイティブ広告枠	2022/02/22 11:00	2022/02/23 11:00	true	300,000	344,742	114.9%
【21年4-6月期以降】はてなブックマークアプリ/ネイティブ広告枠	2022/02/23 11:00	2022/02/24 11:00	false	300,000	262,775	87.6%

Monitoring ダッシュボードによる SLO 運用

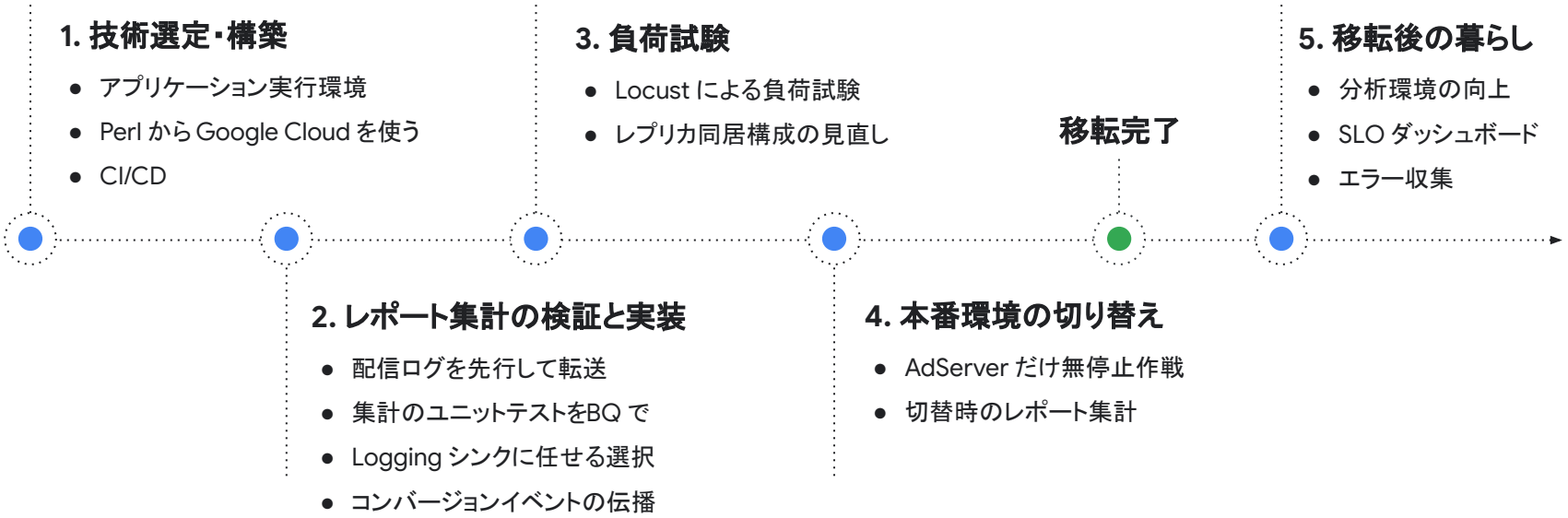
Error Reporting によるエラー収集

- 構造化ロギングでエラーを送信
  - ログの形式設定エラー | Error Reporting | Google Cloud
  - 以下を含む JSON を Logging へ出力

```
{
  "@type": "type.googleapis.com/google.devtools.clouderrorreporting.v1beta1.ReportedErrorEvent",
  "message": "<stack trace>"
  ...
}
```

- 依存ライブラリを追加したりAPI リクエストを管理しなくてよい

まとめ


1. 技術選定・構築
アプリケーション実行環境
Perl から Google Cloud を使う
CI/CD
2. レポート集計の検証と実装
配信ログを先行して転送
集計のユニットテストをBQ で
Logging シンクに任せる選択
コンバージョンイベントの伝播
3. 負荷試験
Locust による負荷試験
レプリカ同居構成の見直し
4. 本番環境の切り替え
AdServer だけ無停止作戦
切替時のレポート集計
移転完了
5. 移転後の暮らし
分析環境の向上
SLO ダッシュボード
エラー収集

Thank you.