Google Cloud

モバイルゲームバックエンドのGKE + Cloud SQL への移行事例

岩立 稜佑

株式会社グリフォン

自己紹介

岩立 稜佑
株式会社グリフォン
SRE

2019 年

- 株式会社サイバーエージェントへ新卒入社
- ゲーム事業部 株式会社グリフォンへ配属
- 運用中のプロダクトを担当

2020 - 2021 年

- Google Cloud 移設へ携わり、2021 年 10 月に移設済み

現在

- Kubernetes を用いたマルチ テナント インフラ基盤の運用を担当

アジェンダ

- AWS から Google Cloud への移行事例について
  - 移設の検討理由
  - Google Cloud を採用した理由
  - 移設前・移設後のアーキテクチャ
  - 移設に対する検証
- 移行時に使用した Database Migration Service について
  - DMS の概要
  - 実際のオペレーション内容
  - DMS のメリット・デメリット

なぜ移設を検討したのか

- 運用している間に課題が発生し、リアーキテクチャを検討
  - クラウドの機能を使いこなせていない
  - 部分的にコード化が出来ていない所がちらほら
    - 設定を変更するたびに確認の手間が増加
- リアーキテクチャするのであれば **クラウドから検討し直しも可**

コンテナ オーケストレーション ツールの検討

- 大前提としてアプリケーションをコンテナ化
- Kubernetes のエコシステムの恩恵を活かしたい
- プロダクトごとにインフラを個別で運用できる体制ではない
    - マルチテナント基盤を運用したい

Kubernetes を使用することに決定

使用するクラウドサービスの見直し

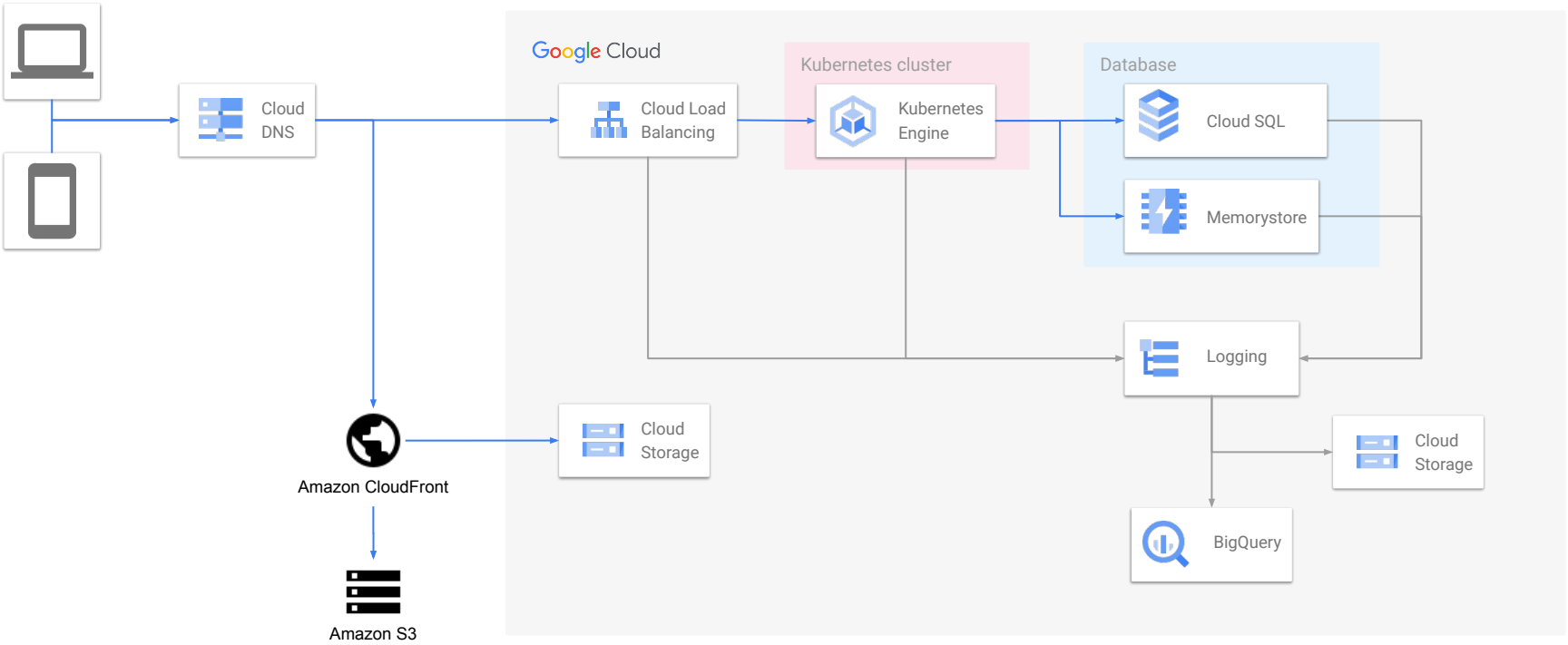
- 昔に Google Cloud を使っていた経験有
  - プロダクトの開発、運用や IaaC も経験済み
- BigQuery を既に使用している
  - AWS からデータを転送し、データ分析に使用中
- GKE には先進的な機能が提供されている

Google Cloud を使用することに決定

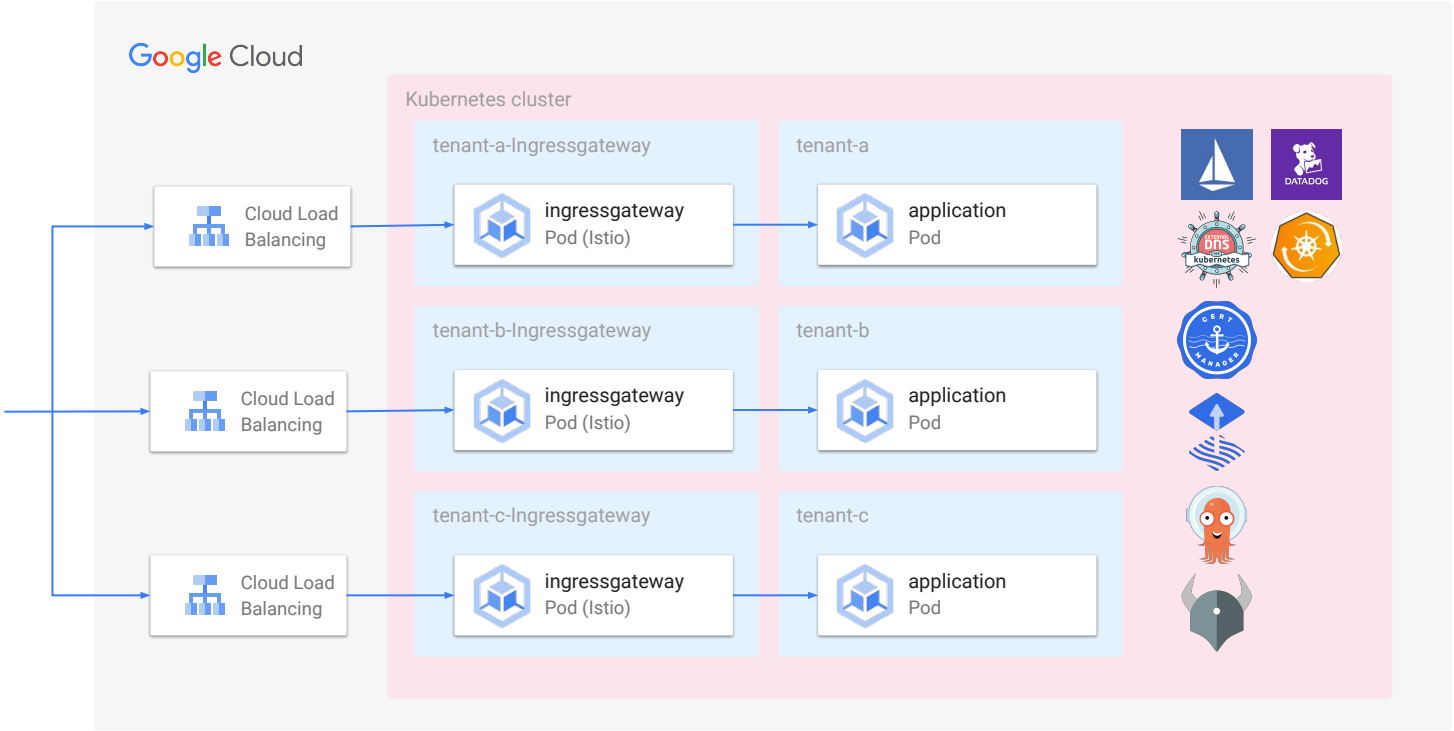
移設前のアーキテクチャ - AWS

- ネットワーク
  - Route53, ELB
- バックエンドサーバ
  - EC2, Aurora, ElastiCache
- 静的配信
  - CloudFront, S3
- ログ
  - Kinesis Data Firehose, Athena, Cloud Watch
- デプロイ
  - Code Deploy

現在運用中のアーキテクチャ - Google Cloud



現在運用中のアーキテクチャ - Kubernetes



移設に際して行った検証

- Kubernetes を基盤とした運用が可能か
  - エコシステムを利用したアーキテクチャの検討
- Cloud SQL による運用が可能か
  - 性能、強制メンテナンス、周辺機能など
- 負荷試験
  - GKE + Cloud SQL + Memorystore の構成でパフォーマンスがどの程度出せるか
- コスト面
  - N1, N2, C2インスタンスの比較など

移設に際して行った検証の結果

- Kubernetes を基盤とした運用が可能か
  - 複数のエコシステムを組み合わせることで可能 ✅
- Cloud SQL による運用が可能か
  - 強制メンテナンスは年に 2,3回あるかどうか → 許容可能と判断✅
  - 性能や周辺機能は十分 ✅
- 負荷試験
  - AWSと同等程度のパフォーマンスを確認 ✅
- コスト面
  - C2インスタンスを採用することで同程度の料金で運用可能 ✅
    - N1,N2より値段は高いが、クロック周波数が高いため最終的に N1,N2より安くなった

Cloud SQLへの移設の検証

- mysqldump を用いたフルダンプからのリストア
  - 多くの操作を手動で行う必要があった
  - 移行に時間がかかり、その間サービスを停止する必要があった
    - 約 1 TBの DB で 116 時間 (約 5 日)

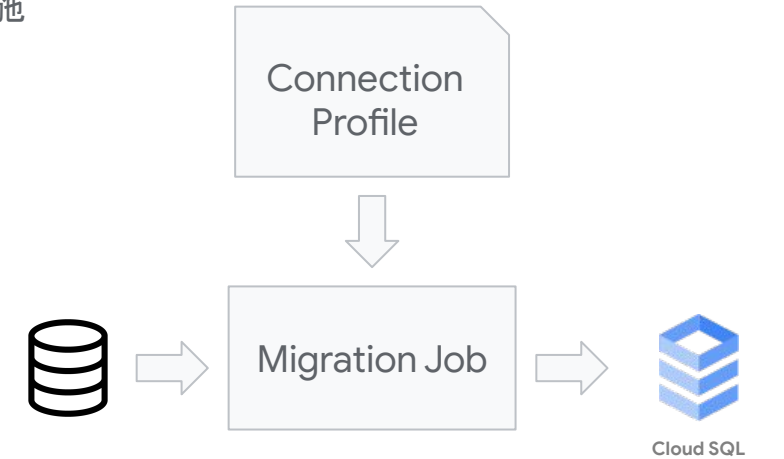
Data Migration Service

Database Migration Service

- Cloud SQL への移行を簡単に行えるサービス
- GUI で設定するだけで移行が可能
- フルダンプ + リストア + 差分追従を自動で実施
- 移行時の切り替え時間を最大限短縮

DMS に登場するコンポーネント

- 接続プロファイル、ジョブの 2 要素で構成
- 接続プロファイルで DB への認証情報などを設定
- プロファイルを元にジョブで Cloud SQL への移行を実施

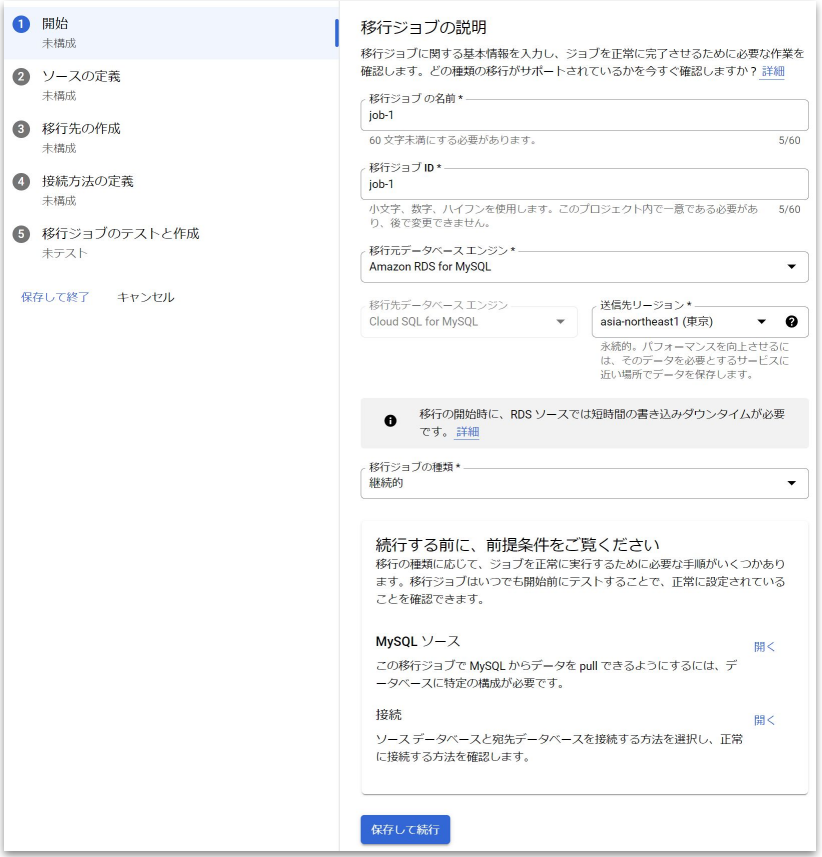


DMS の設定項目 - 接続プロファイル

- 移設元 DB の情報
  - データベースの種類
    - MySQL, Amazon RDS, Cloud SQL が選択可
  - 接続情報
  - 暗号化種別

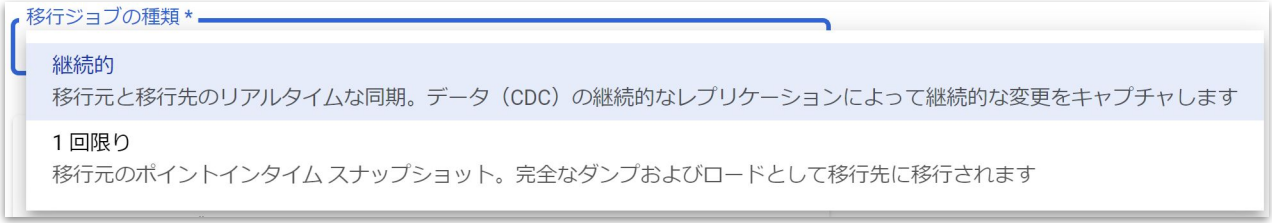
DMS の設定項目 - 移行ジョブ

- 移行元 DB の設定
- 移行ジョブの種類
- 移行先 DB の設定
  - Cloud SQL インスタンスの設定
- 接続テストの実施
  - 要件を満たしていない場合はエラー



移行ジョブの種類

- 継続的モード
  - フルダンプのロード後に変更されたデータをレプリケーションによって継続的に追従
- 1 回限りモード
  - フルダンプとリストアのみ行う。その後に変更されたデータに関しては追従しない



DMS の制約

- 移設元 DB のホスト名が 60 文字制限
  - 超過した場合は CNAME レコードで対応
- 移行先 DB は自動で作成されるインスタンス以外使用不可
- MySQL System Database は移設対象外のため、ユーザは移設後に作成し直す必要あり
- 移行元 DB で binlog を有効化する必要あり
- 継続的モードを選択した場合はレプリケーションを行うため、 gtid_mode の設定が必要
  - ON, OFF, OFF_PERMISSIVE のいずれかに設定。ON_PREMISSIVE はサポート外
- 移行ジョブを開始するタイミングで書き込みが停止している必要がある
  - 数十秒から数分で終了

グリフォンにおける DMS に対する要件

- 1TB弱のデータベースの移行
- ダウンタイムは当日数時間のメンテナンスで行える範囲内
  - レプリケーション遅延
  - オペレーションにかかる時間

要件に対する検証結果

- 1TB弱のデータベースの移行
  - 3 日で移行完了✅
- ダウンタイムは当日数時間のメンテナンスで行える範囲内
  - レプリケーション遅延
    - 1秒未満で影響無し✅
  - 当日のオペレーションにかかる時間
    - 切り離しにかかる時間は 1分未満✅

移設のオペレーション内容 - 移設 1 ヶ月前

- トラブルに備えて 1 ヶ月前からデータベースの移行を開始
  - Cloud SQL 側での作業
    - 接続プロファイル、移行ジョブの作成
  - Aurora 側での作業
    - Public Endpoint の払い出し
    - 移行先の Cloud SQL からアクセスを許可

移設のオペレーション内容 - 移設当日

1. サービス側のメンテナンス処理
2. **DMS の移行ジョブを Promote して Cloud SQL を Aurora から切断**
3. **Cloud SQL にユーザを作成**
4. Cloud SQL から Aurora へのレプリケーションを設定
5. サービスを AWS から Google Cloud へ切り替え

トラブルに備えて切り戻しの用意

- Google Cloud への移設に失敗した場合に備える必要がある
  - 切り戻し時のロールバックを防ぐために、Cloud SQL に行われた変更は Aurora にも反映
- DMS は Cloud SQL への一方通行で Cloud SQL ➡ Aurora への移設は不可
  - 今回は AWS の Database Migration Service を用いて切り戻し用レプリケーションを設定

DMS のメリット・デメリット

- メリット
  - Cloud SQL への移行は非常に楽に行える
  - 移設時のオペレーション時間を短縮出来る
- デメリット
  - Cloud SQL への移行のみにしか使えない
    - 切り戻し用の設定が不可能
  - 既存の Cloud SQL インスタンスが使用不可
    - IaaC の観点から Terraform 等によって生成したインスタンスを使用したい

まとめ

- Cloud SQL への移設に関する問題点を DMS によって解決
  - Cloud SQL への移設であれば満足に使用可能
  - 切り戻しは考える必要あり
- 今後はマルチクラスタに対応
  - CNI やクラスタ自体の変更時のブルーグリーンを考えて必要
  - プロダクトの性質によってはやむを得ず別クラスタにする必要性を考慮

Thank you.

Google Cloud