

# ReproのImport/Exportを支える サーバーレスアーキテクチャ

Repro株式会社 橋立友宏 (@joker1007)

# Agenda

- コンテナ/サーバーレスの利点
- Reproというサービスの特性
- システム構成、アーキテクチャ紹介
- Amazon ECSとAWS Lambdaの使い分け
- AWS Step Functionsの利点
- 今後の展望

はじめに

そもそも、コンテナ/サーバーレスの何が嬉しいのか

***Repro***

# コンテナの大きな利点

## 環境自体のパッケージング

- 環境の再現性が高い
- ポータブル
- ミドルウェアを含めたアプリケーションセット自体をデプロイ

# サーバーレスコンポーネントの大きな利点

- 構築の手間が少ない
- 運用負荷が少ない
- リソースコントロールが柔軟

今回は特にリソースコントロールに注目する。

# 自己紹介

- 橋立 友宏
- Repro株式会社 執行役員 Chief Architect
- id: @joker1007
- パーフェクトRuby, パーフェクトRailsなどを共著
- 最近の仕事はデータエンジニアリング、Kafka、Kafka Streams

Reproのサービスと特性の紹介

**Repro**

デジタルマーケティングを総合的に支援する  
マーケティングオートメーションサービス。

**Repro**

# Push Notification

Repro

く プッシュ通知

## 新しいプッシュ通知の作成

キャンペーン

キャンペーン名

キャンペーン名

キャンペーンのゴール (任意)

キャンペーンのゴールは、作成後には変更できません。

なし

キャンペーンの狙いと実施後の結果 (任意) ⓘ

キャンペーンの狙いと実施後の結果

☐ 通知をニュースフィードとして使う ⓘ

このプッシュ通知のタイトル、本文、画像などの内容をAPI経由で取得できるようになります。 [詳細](#)  
タイトルは必須です。プッシュ通知配信後は設定を変更できません。

メッセージ

パターン

パターン1

+ 追加

プッシュ通知の種類

スタンダード

カスタム (JSON)

タイトル (任意)

iOS 8.4以上、もしくはAndroidで表示されます。

タイトルをここに書いてください

本文

本文をここに書いてください

最大表示文字数 (目安) : 80~160文字

リッチ通知メディア (任意) ⓘ

画像、GIF  
URLもしくはアップロード

動画  
URLのみ

音声  
URLのみ

メディアを表示できるのはiOS 10 以上、Android 4.1 以上です。 [詳細](#)

ディープリンクもしくはURL (任意) ⓘ

myApp:// or https://example.com

追加設定

このオプションが適用されるのはiOSのみです。

☐ バッジを表示する ⓘ  
バッジを消すためにはアプリ側で実装が必要です。 [詳細](#)

☐ プッシュ通知のサウンドを変更する

通知プレビュー

iOS

Android

プレビュー

プレビューと実際の通知の表示は異なることがあります。配信時には実際のアプリに設定されているアプリアイコンとアプリ名が使用されます。

ReproDev (development)



# Popup Message

Repro

## 新しいアプリ内メッセージの作成

### キャンペーン

#### キャンペーン名

#### キャンペーンのゴール (任意)

キャンペーンのゴールは、作成後には変更できません。

#### キャンペーンの狙いと実施後の結果 (任意) <sup>?</sup>

#### ☐ キャンペーンをニュースフィードとして使う <sup>?</sup>

このキャンペーンで配信するメッセージの概要を、ユーザーごとの配信履歴としてAPI経由で取得できるようになります。 [詳細](#)  
キャンペーン公開後は設定を変更できません。

### メッセージ

#### パターン

パターン 1

+ 追加

#### オーバーレイ

背景色 

#### ダイアログ

背景色 


#### 見出し (任意)

見出し  0 / 39

#### 本文

本文  0 / 255

#### アクションボタン 1

ボタン  0 / 16

#### ディープリンクもしくはURL (任意) <sup>?</sup>

#### トラッキングイベント名 (任意) <sup>?</sup>

#### プレビュー

プレビューと実際のメッセージの表示は異なることがあります。



# 柔軟なユーザーオーディエンスの抽出

**Repro**


## 新しいオーディエンスの作成

### オーディエンス

オーディエンス名

Audience1

### 対象ユーザー

フィルター 

× フィルターを消去

カートに追加

を

3

日以内に

1

回以上

実行した

+ OR

And

And not

年齢

が


25

以上

+ OR

×

+ フィルターを追加

対象ユーザー数 0 

オーディエンスを保存

# Import/Export処理の特性

数千万規模のユーザー情報を任意のタイミングでImport/Exportできる。

- パターン化できない負荷
- それなりの実行時間と負荷
- 任意のタイミングで並列実行

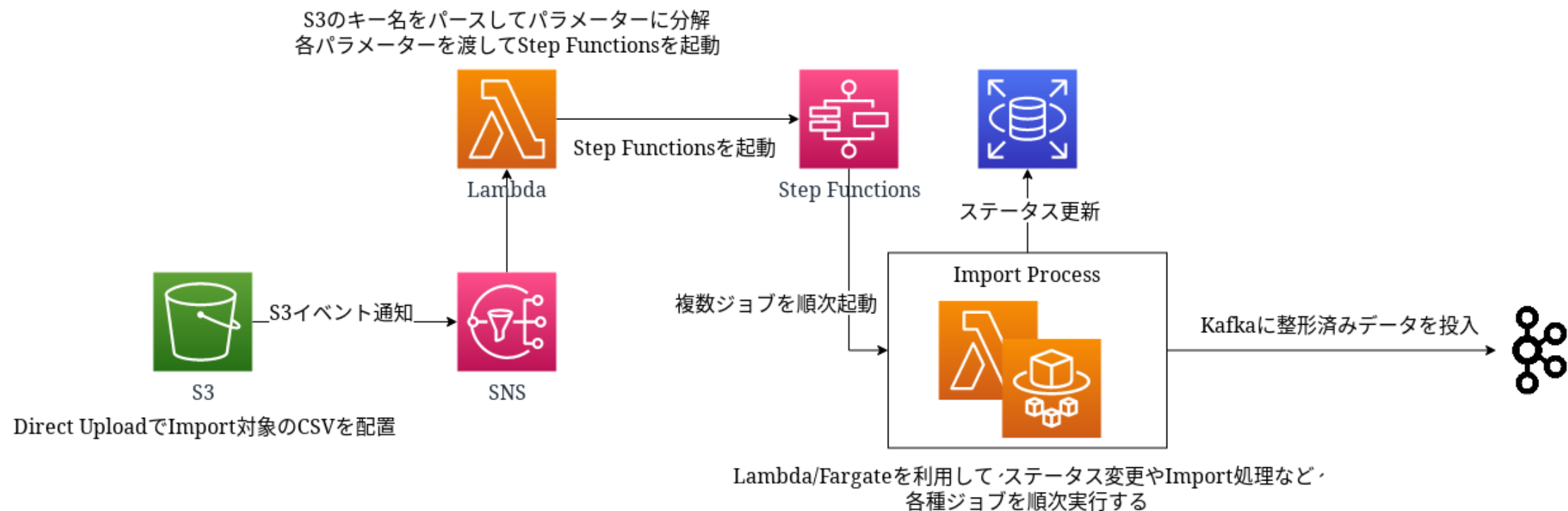
言い換えると

- 突発的に高い負荷がかかる
- 理論上のピークは非常に高くスケーラビリティが必要
- 負荷が無い時は全く無い

計算資源の柔軟なコントロールが重要

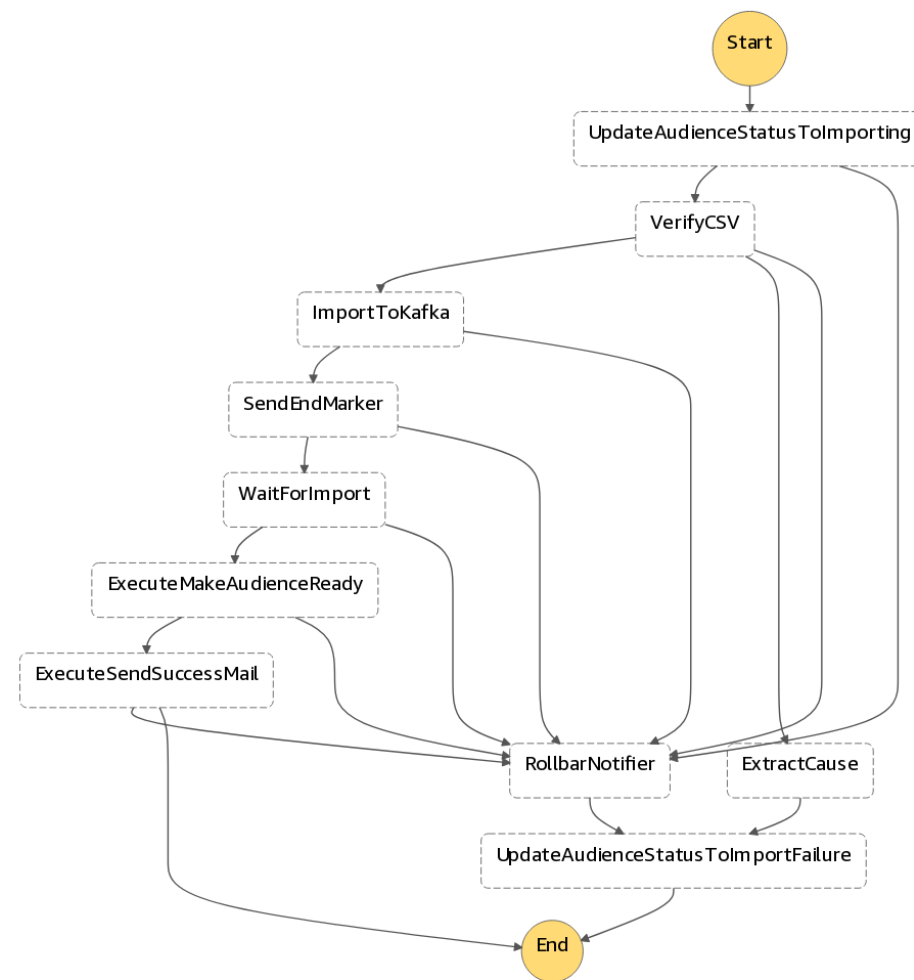
**Repro**

# システム構成



# AWS Step Functions 詳細

実際のシステムを省略して主要な処理のみを記載しています



# AWS Fargateによるリソースコントロール

AWS Fargateで必要な時にだけembulkを実行するリソースを確保することで、リクエストが無い時のコストを0にしつつ素早くジョブを実行できる。

注意点として、AWS Fargateの起動には1分～2分程プロビジョニングにかかるオーバーヘッドが発生する。

今回の要件では、そもそも実行時間が数分から数十分かかるので、起動時のオーバーヘッドは許容できるレベル。

# AWS Lambdaとの棲み分け

- 実行時間が15分を越える可能性がある場合
- マルチスレッドを活用するCPUバウンドな処理を含む場合
- ディスクI/Oを伴う場合 (特にI/Oが多い場合はAmazon EC2ベースを利用)

こういったケースではAWS Lambdaを利用できない、もしくは推奨できない。

ECSでは、AWS Fargateの上限を越える様なリソースが欲しい場合は、Capacity ProviderとAuto Scalingの組み合わせに切り替えることで、同一の基盤を利用しつつスケールアップにも対応できる。

# AWS Lambdaのコンテナイメージサポート

2020年末頃に追加されたAWS Lambdaの新機能により、コンテナイメージをアップロードしてそのまま実行できる様に。

- 最大10GBまでと十分なサイズのイメージをサポート
- デプロイに一定の準備時間がかかるが、その後の起動は早い
- コンテナと同一の基盤でコード管理が出来る
- ライブラリのバージョンを含めた複雑な環境をコントロールできる



# 複雑なアプリケーションの現実

アプリケーションの一機能をAWS Lambdaで構成する時、以前の環境ではコードベースが共有できないケースが多かった。

特にLambda関数実行時のシステム構成が隠蔽されていたので、RDBに接続するだけでもライブラリのバージョンや配置方法で一捻り必要だった。

コンテナイメージをそのまま利用できる様になったので、複雑なアプリケーションフレームワークの環境を丸ごとAWS Lambdaで動かせる様になった。

# Ruby on Rails on AWS Lambda

Railsのアプリケーションコードを通常のWebサービスのコードベースのままAWS Lambdaで動作させ、アプリケーションドメインのロジックの完全な共通化を実現。

といってもWebリクエストを受けている訳ではなく、Railsにおけるバックグラウンド処理の実行基盤としてAWS Lambdaが活用できるようになったということ。

デプロイが完了した後は、実行のオーバーヘッドはコールドスタートでも数秒以下でAWS Fargateの起動より圧倒的に高速。

# AWS Step Functionsによる統合

JSONでシンプルに記述できて、AWSの複数のサービスを柔軟に実行コントロールできる。

AWS Step Functionsという名前以上に様々なサービスに対応している。

今回のシステムでは、AWS FargateとAWS Lambdaを特性に合わせて切り替えつつ、実行プロセスのステップごとにエラーを可視化することに活用している。

# AWS Step Functionsの記述例 1

State定義の中のLambdaタスクを抜粋したもの。

本来はエラーハンドリング等を含むため、もう少し記述が多い。

```
"UpdateAudienceStatusToImporting": {
  "Type": "Task",
  "Resource": "arn:aws:states:::lambda:invoke",
  "Parameters": {
    "FunctionName": "rake-handler",
    "Payload": {
      "TASK_NAME": "imported_audience:update_status",
      "RAKE_ENV_USER_SEGMENTATION_ID.$": "$.user_segmentation_id",
      "RAKE_ENV_UPDATE_STATE_EVENT": "start_importing",
      "RAKE_ENV_ASSUMED_AFTER_STATE": "import_audience_importing",
      "RAKE_ENV_STEP_FUNCTION_EXECUTION_ARN.$": "$$.Execution.Id"
    }
  },
  "Next": "ImportToKafka"
```

# AWS Step Functionsの記述例 2

同様にECSタスクを抜粋したもの。

```
"ImportToKafka": {
  "Type": "Task",
  "Resource": "arn:aws:states:::ecs:runTask.sync",
  "Parameters": {
    "LaunchType": "FARGATE",
    "Cluster": "batch-worker",
    "TaskDefinition": "embulk",
    "Overrides": {
      "ContainerOverrides": [
        {
          "Name": "embulk",
          "Environment": [
            {"Name": "INSIGHT_ID", "Value.$": "$.insight_id"},
            {"Name": "USER_SEGMENTATION_ID", "Value.$": "$.user_segmentation_id"},
            {"Name": "S3_BUCKET", "Value.$": "$.s3_bucket"},
            {"Name": "S3_KEY", "Value.$": "$.s3_key"}
          ],
          "Command": ["../wrap.sh", "embulk", "run", "-b", ".", "configs/send_audience_to_kafka.yml.liquid"]
        }
      ]
    },
    "NetworkConfiguration": {
      "AwsvpcConfiguration": {
        "Subnets": ["subnet-xxxxxxx"],
        "SecurityGroups": ["sg-xxxxxxx"]
      }
    }
  },
  "Next": "SendEndMarker"
},
```

# マイクロサービスとAWS Step Functions

単機能の簡単な同期処理を複数組み合わせ、複雑なことを実現する。  
それを支援する基盤としてAWS Step Functionsは最適。  
冪等性を上手く考慮できれば、リトライやエラーハンドリングも非常に簡単。

## まとめ

- AWS FargateとAWS Lambdaをコンテナイメージという同一のパッケージで構成できる様になった
- それぞれの特性に合わせて実行基盤を切り替える
- 一つ一つのジョブは出来るだけシンプルな同期処理として実装する
- AWS Step Functionsを使ってシステムの実行フローを統合する

## 今後の展望

- Amazon EventBridgeからAWS Step Functionsが直接起動できる様になったので活用したい
  - Amazon S3からAmazon EventBridgeも可能になったので、Amazon S3 -> Amazon EventBridge -> AWS Step Functionsが可能
- AWS App Runnerが活用できる範囲があるか検討



# お知らせ

Reproは今日紹介した様に、大規模なサービス事業にも対応できるマーケティングオートメーションサービスを提供しています。サービスの詳細は以下のURLをご覧ください。

<https://repro.io/>

またRepro社では、フロントエンドエンジニアから分散処理基盤を扱うデータエンジニアまで様々な技術者を募集しています。採用情報の詳細は以下のURLをご覧ください。

<https://company.repro.io/recruit/>

本日はご静聴ありがとうございました。

**Repro**