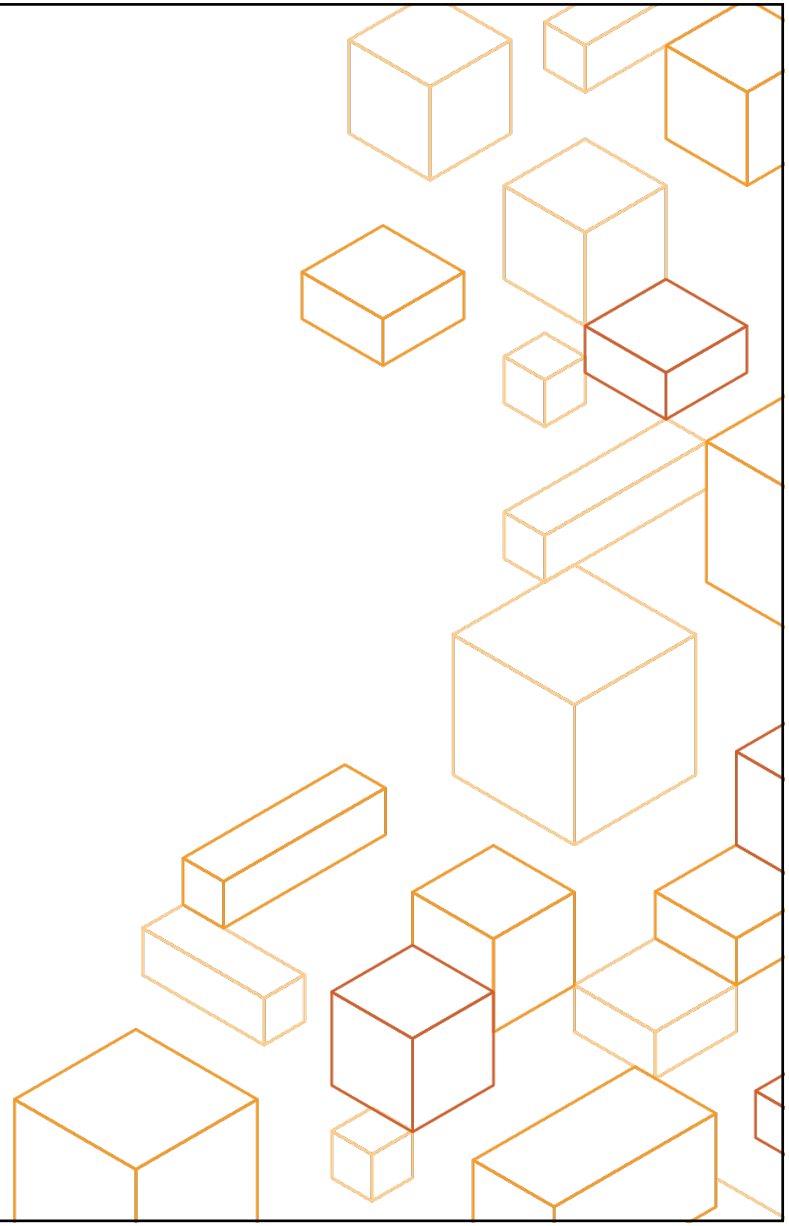


サーバーレスのおさらい

主要サービス、利用パターン、事例

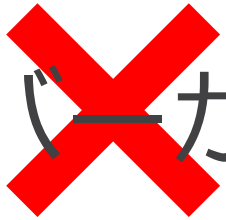
2021/09

アジェンダ

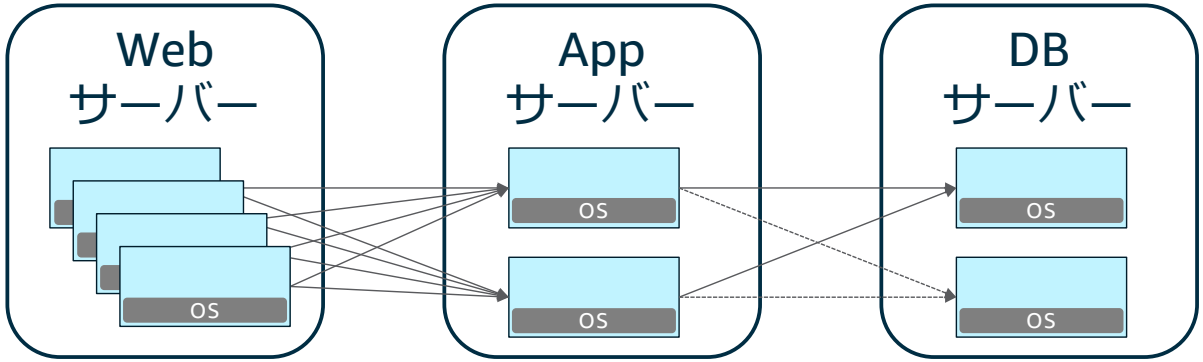
- サーバーレスとは
- 主要コンポーネントと機能的な特徴
- パターンで考える

サーバーレスとは
- おさらいとビジネス効果

サーバーレスとは

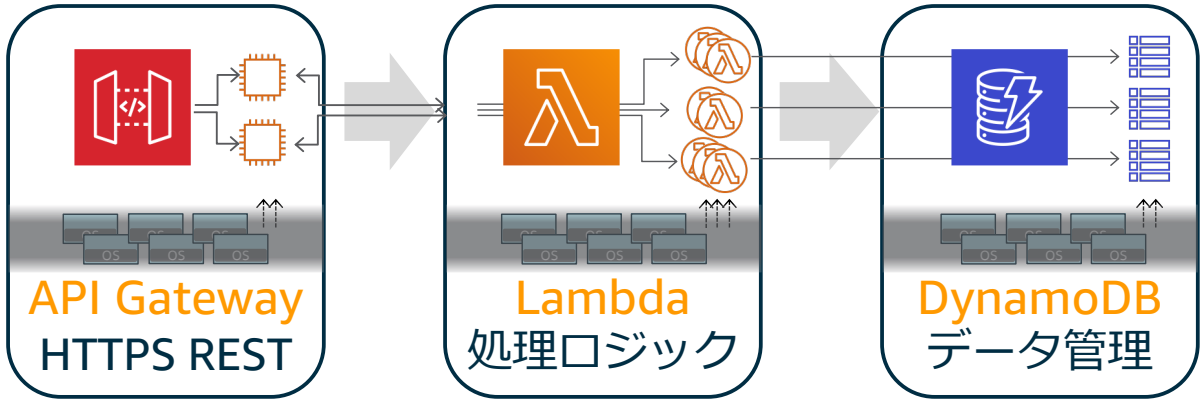


これまでの方式との対比



サーバー/OSの準備・構成
設定・開発作業

+ 規模の見積もり
+ 可用性設計
+ データ保全の検討

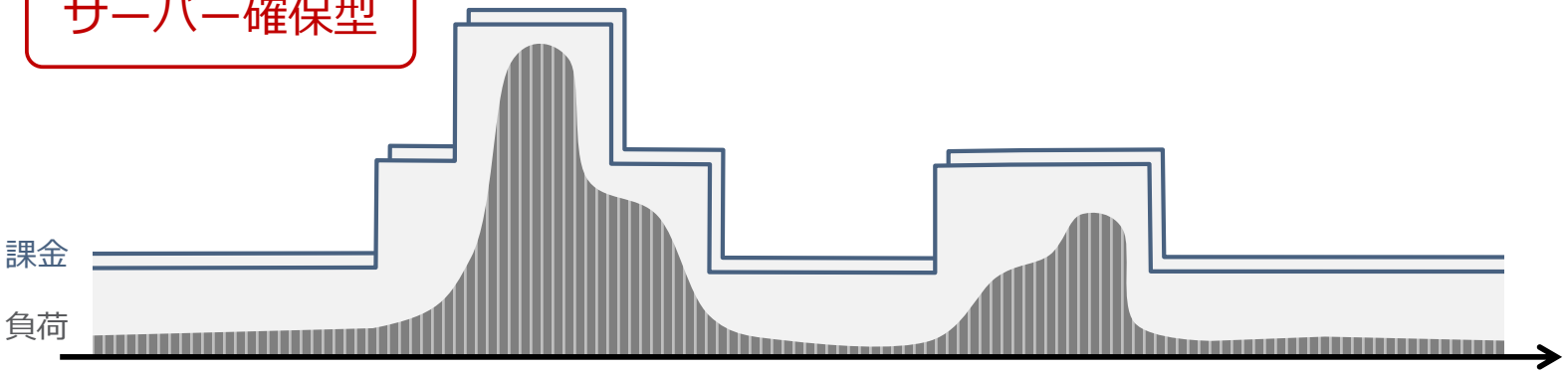


~~サーバー/OSの準備・構成~~
設定・開発作業

✓ リクエスト量に応じて自動スケール
✓ 設計済みのリトライ
✓ データ信頼性

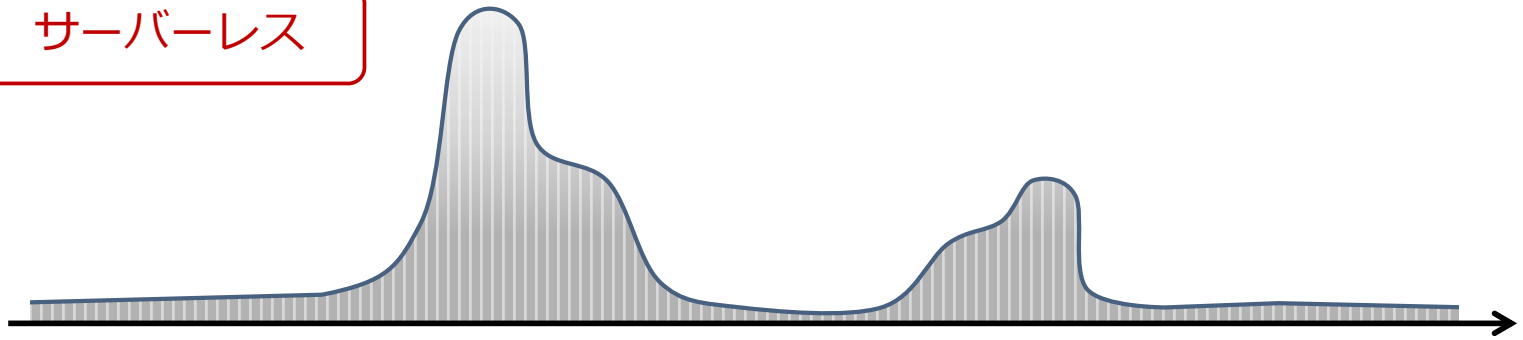
サーバーレスの利用費の構造

サーバー確保型



- 処理量を予測して環境を確保
- 確保分の課金
- 使わないときは（意識して）解放
- 自分で冗長化

サーバーレス



- 処理要求に応じて自動で環境を確保
- 負荷なし = ゼロ課金
- ms 単位の実行時間課金（AWS Lambda）
- 自動で冗長化

サーバーレスに対するお客様の認識

初期段階の思惑

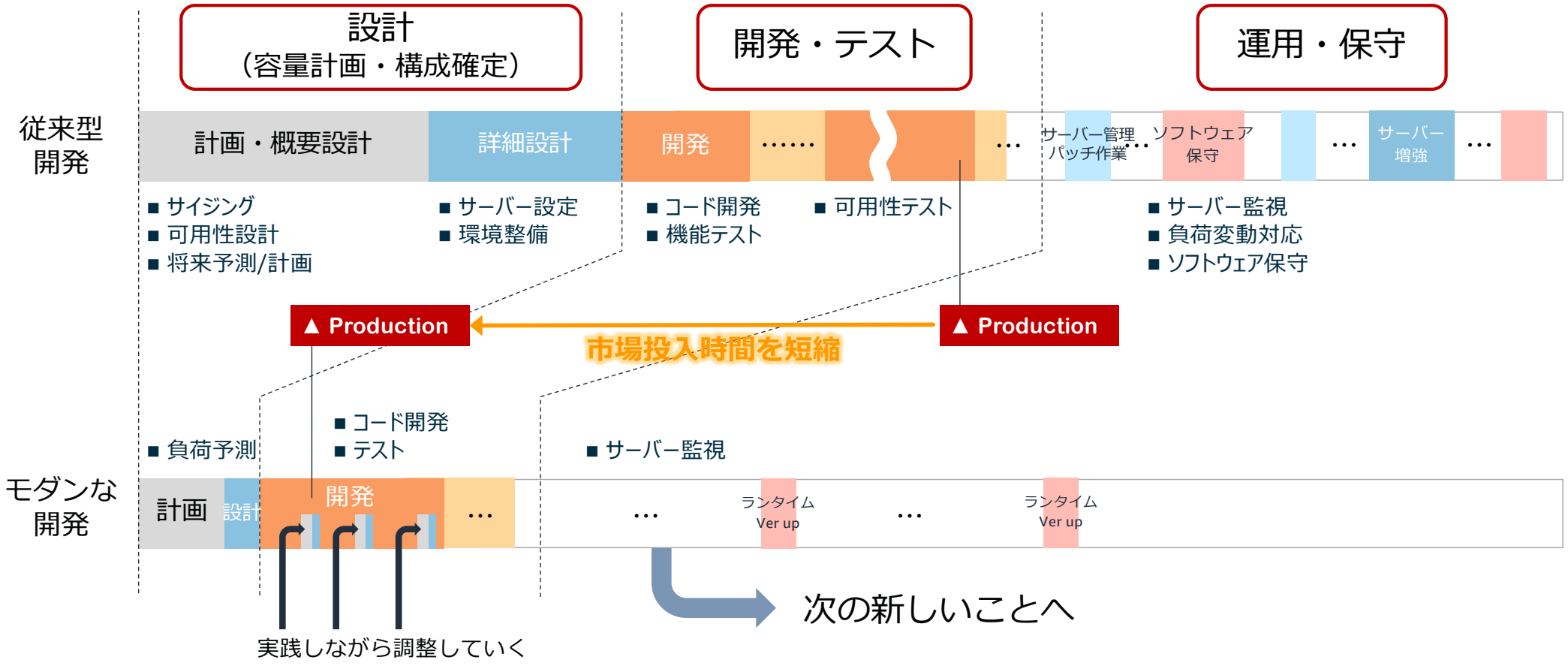
- 利用費を下げたい
  - アイドル時のマシン確保の費用をなくしたい

実際に経験した方の声

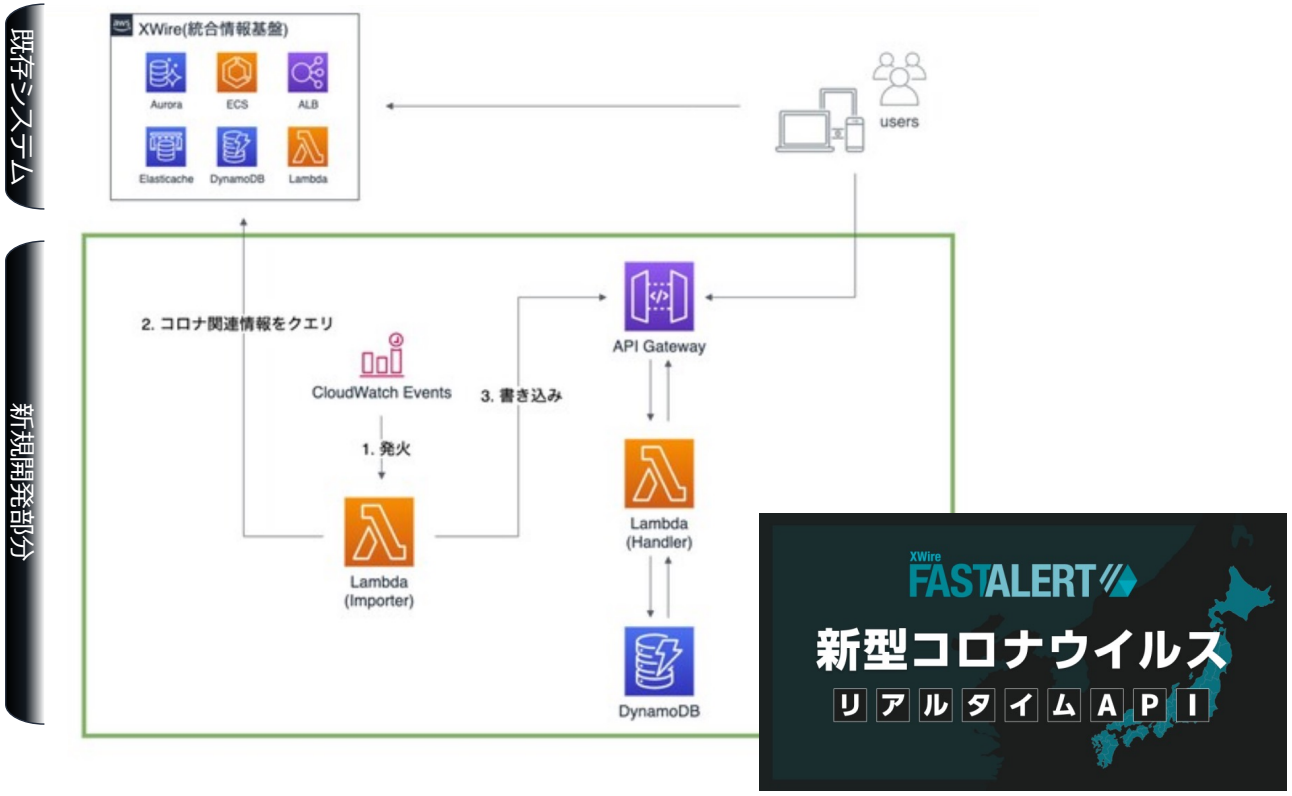
- バージョンアップ作業が減る
- サーバー枯渇を気にしなくなる
- セキュリティパッチ対応が減る
- 冗長設計作業、障害テストが減る

- 開発・生産的な作業時間が増える
- 改善サイクルが回るようになる

作業量の削減 + 時間の短縮



JX通信社様
FASTALERT 新型コロナウィルス 特設ページ



法人向けのニュース情報収集・
配信サービスの特設ページ

高生産性

5日で開発、サービス開始。 API の
クォータ、スロットリングを迅速実装。

**マネージド
自動リソース管理**

今後の利用拡大予測せずともサービス
展開可能。問題なく稼働中。

コスト最適化

無料枠 + 利用されただけの課金
→ エンジニアの判断で進めやすい

サーバーレス の効能

作業量の削減 + 時間の短縮 + 利用費の適正化

サーバーレスによるお客様の効果例

5x 従来より生産性が向上
アプリ展開を加速化

1/6 安定した定常稼働により
運用の労力を大幅に短縮

1/3 コード量の減少（従来比）
= 生産性向上、保守改善

1人 運用を 1 人で楽に実施
機能改善に注力可能

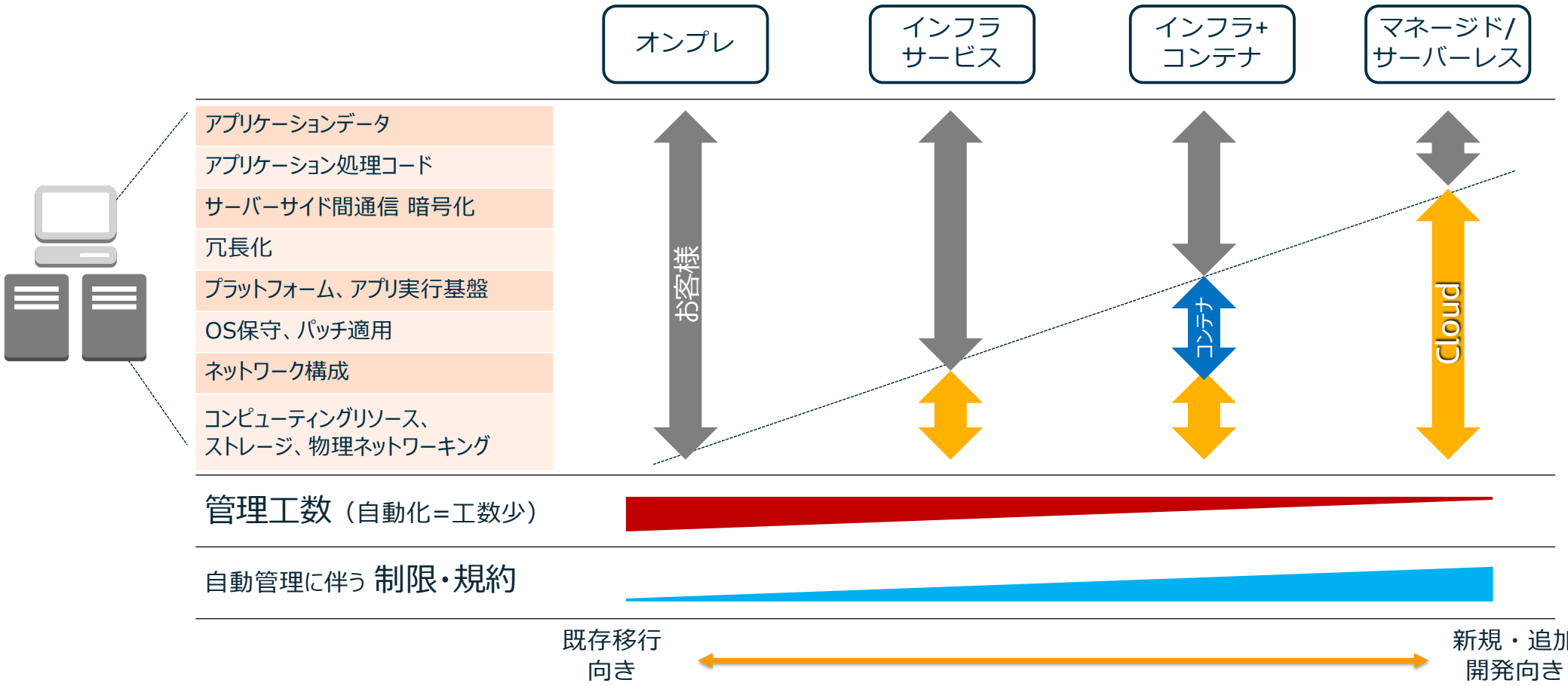
2ヶ月 スケール、冗長化などの
考慮不要で短期実装可能

9:1 “開発:保守/運用” の作業
比率が 1:9 から大きく改善

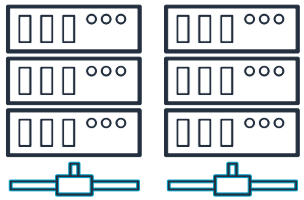
1-2日 簡易な機能追加は短期で
実装・デプロイ可能

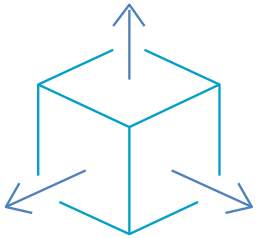
1/10 アイドル時間のリソースが
解放され、利用費が最適化

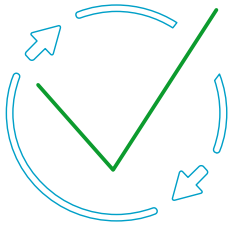
サーバーレスとは



サーバーを意識しないことで...







サーバー管理が不要（準備、OS保守 etc）

柔軟なスケーリング（拡張/縮退）

十二分に考慮された高可用性

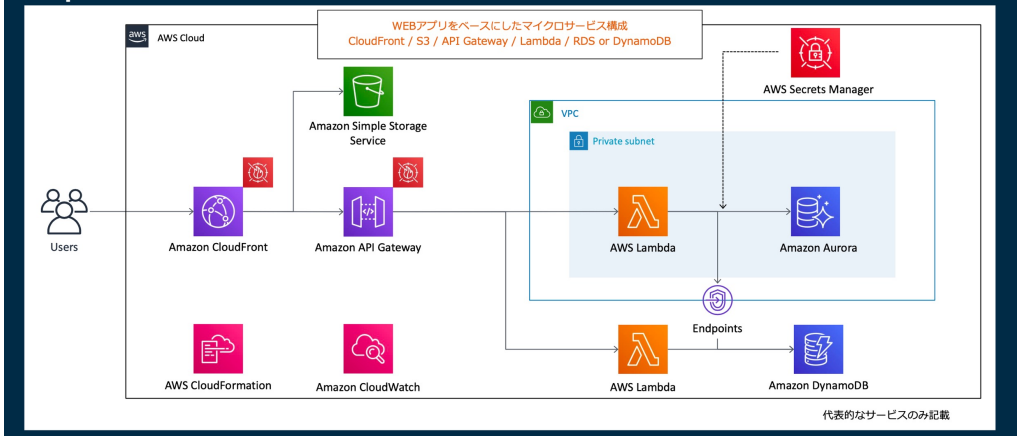
アイドル時のリソース確保が不要

- ユーザーの責任領域を小さくしそこだけに注力
  - 高生産性
  - マネージド業務注力
  - 変更容易性
  - エンジニア意識改革
- 実際の処理負荷に応じて自動で拡張/縮退
  - スケーラビリティ（機会損失防止）
  - マネージド自動リソース管理
  - リアルタイム（付加価値/機能差別化）
  - コスト最適化

SOMPOホールディングス様

AWS DevDay 2020 にて講演
「DX時代における最適な開発手法 サーバーレスとDB選択の勘所」

Sprint Teamが選んだ構成



In Partnership with intel

標準構成を準備して得られた恩恵

事例：車両変更受付フォームの構築

コロナ禍でコールセンターが休業となり、保険契約者が新しい自動車に
契約内容を変更するための、申請受付を行うサイトを開発する必要が発生。

通常であれば二ヶ月以上は掛かる見込み…

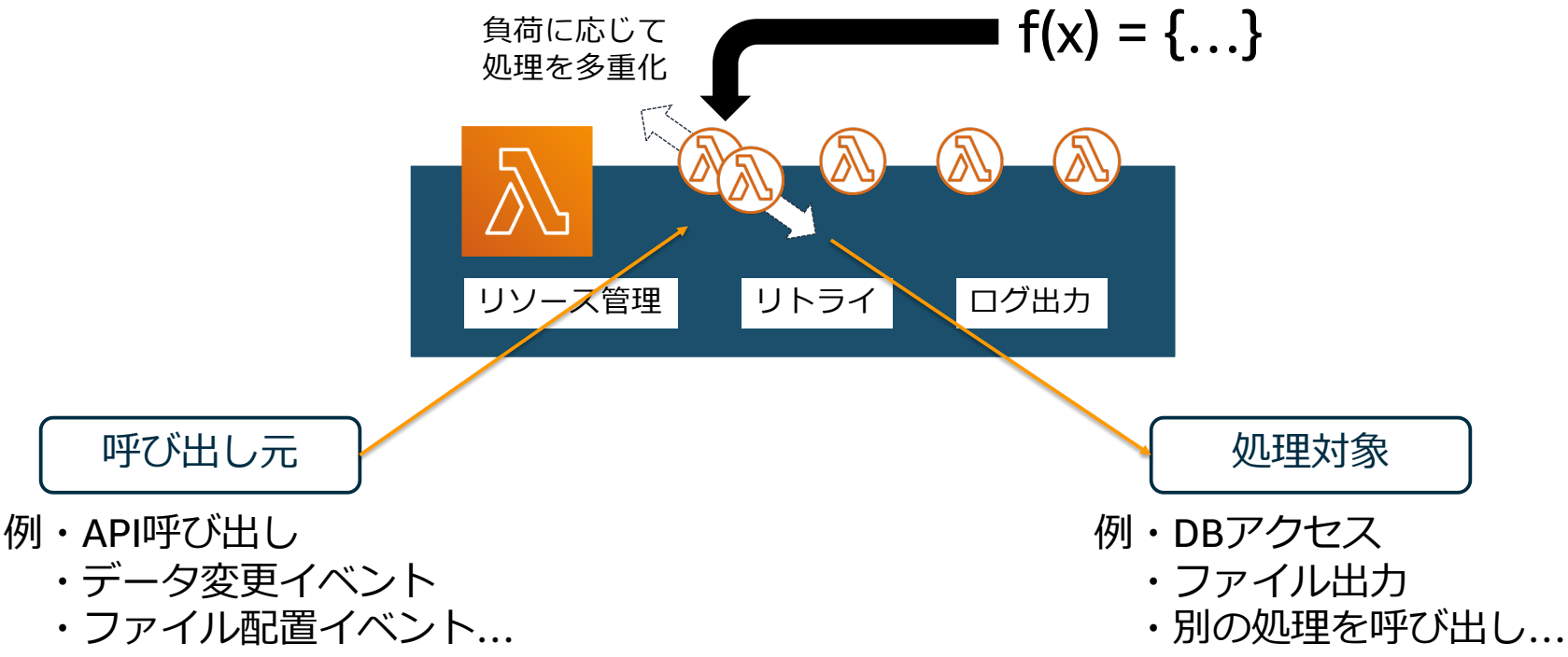
標準構成を利用して開発期間を短縮
三週間でのリリースを実現

In Partnership with intel

主要コンポーネントと機能的な特徴

Function as a Service: AWS Lambda

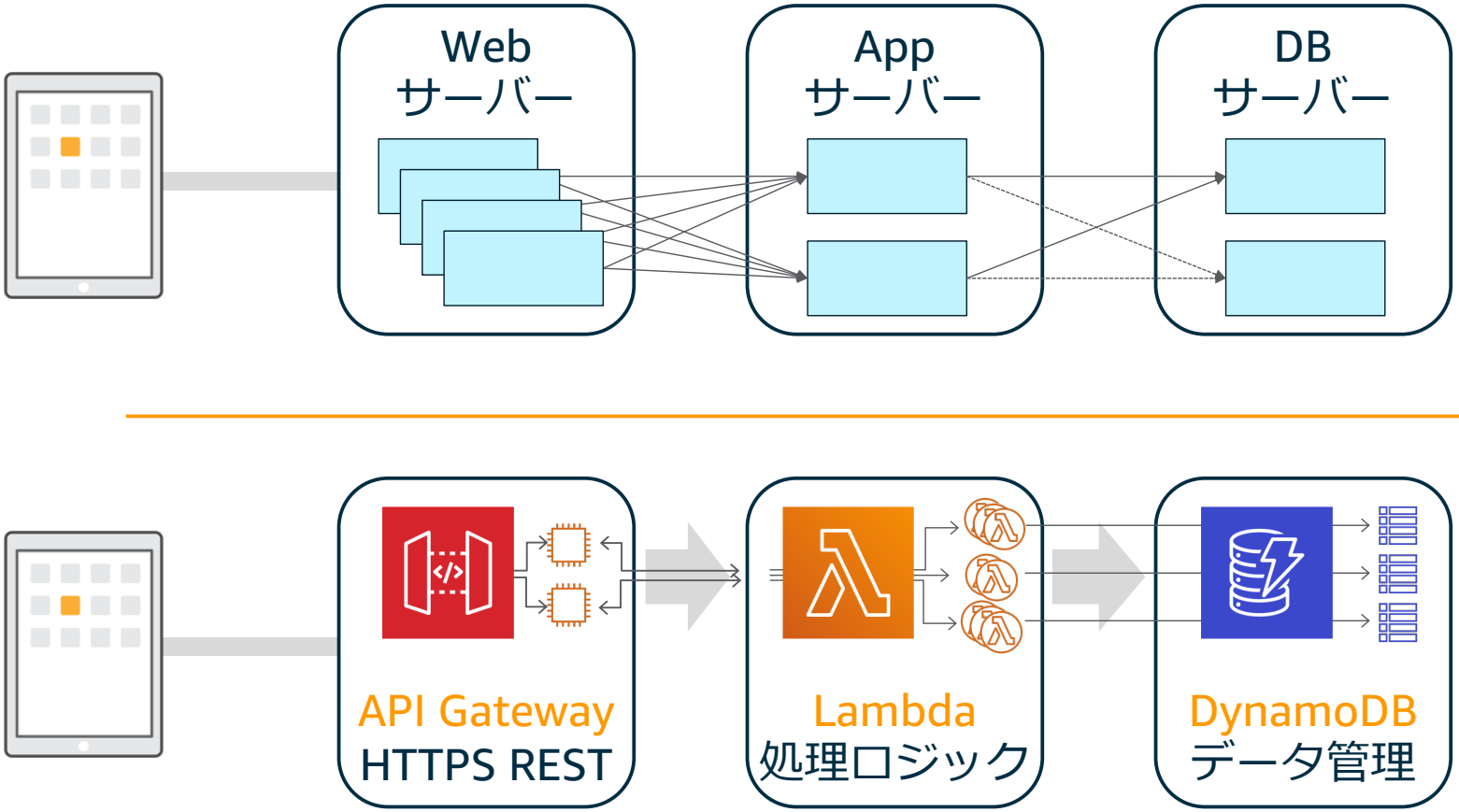
コードを用意 → 実行時に適切にインフラを確保して処理を実行



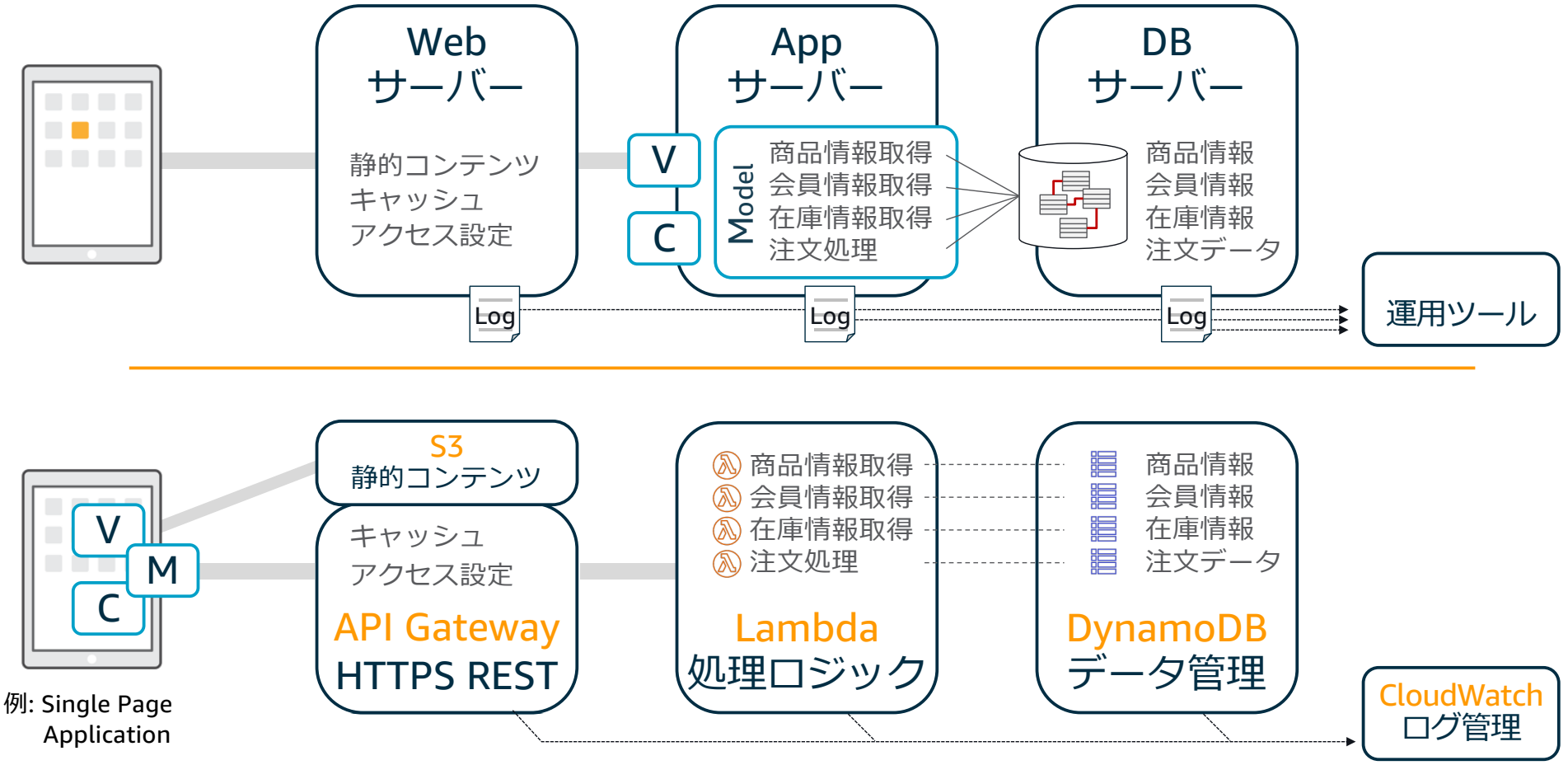
例・API呼び出し
・データ変更イベント
・ファイル配置イベント…

例・DBアクセス
・ファイル出力
・別の処理を呼び出し…

これまでの方式との対比（再掲）：物理構成



これまでの方式との対比: アプリ設計



設定の実際

https://xxxxx.execute-api.
<<region>>.
amazonaws.com/Prod/

※ デフォルトURL
カスタムURL設定可能

構成・設定

- URL
- 認証
- キャッシュ
- 関数紐付け

getAllItemsFunction

```
// DynamoDB へのアクセス
const dynamodb = require('aws-sdk/clients/dynamodb');
const docClient = new dynamodb.DocumentClient();

// テーブル名を環境変数から取得
const tableName = process.env.SAMPLE_TABLE;

exports.getAllItemsHandler = async (event) => {
    const { httpMethod, path } = event;
    if (httpMethod !== 'GET') {
        throw new Error( 'getAllItems: GETである必要があります' );
    }

    // ログへの出力
    console.log('received:', JSON.stringify(event));

    // 全件取得
    const params = { TableName: tableName };
    const { Items } = await docClient.scan(params).promise();

    // 出力の準備
    const response = {
        statusCode: 200,
        body: JSON.stringify(Items),
    }
    return response;
};
```

※ Node.js の場合

DynamoDB

- テーブル
- データ

or

RDS

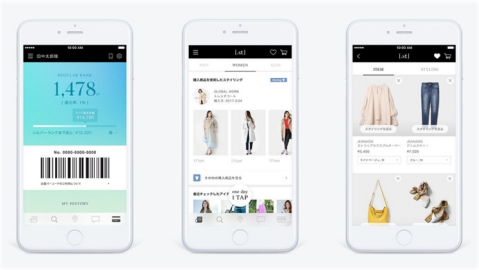
- テーブル
- データ

アダストリア様
サーバーレスによるモバイルバックエンド

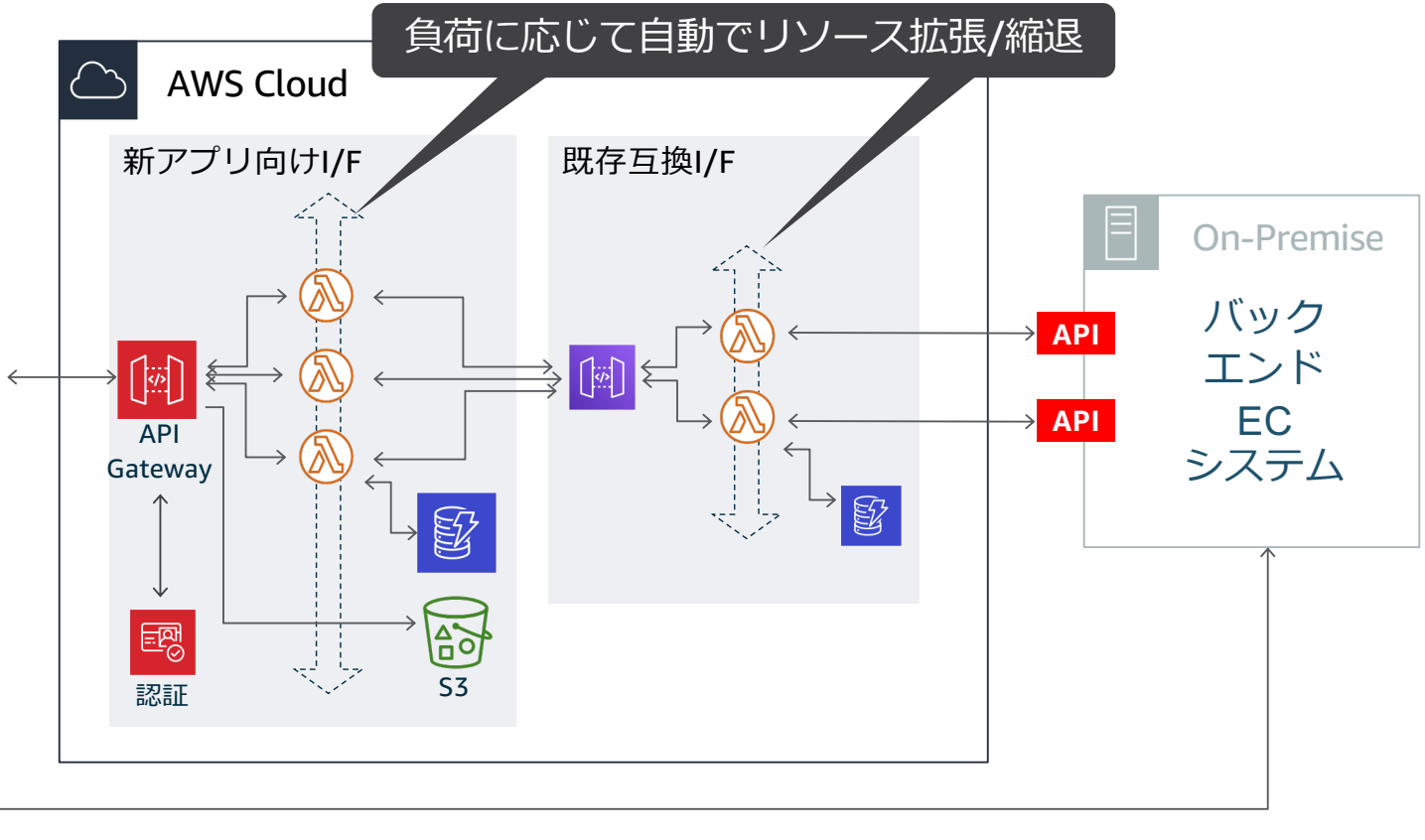
マネージド
自動リソース管理

マネージド
業務注力

- モバイルアプリ「.st」



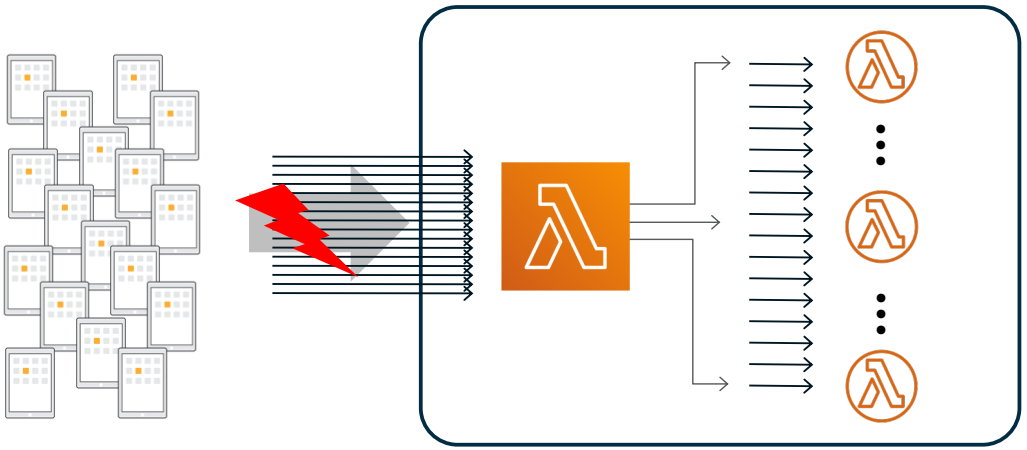
- PC Web用 ECサイト



機能的な特徴

- サーバーレスにおけるリソース管理
- サーバーレスにおけるデータベース

サーバーレスにおけるリソース管理

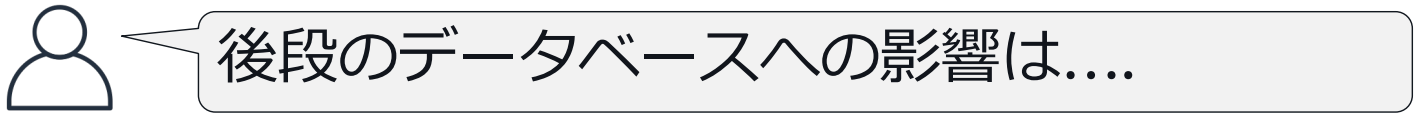


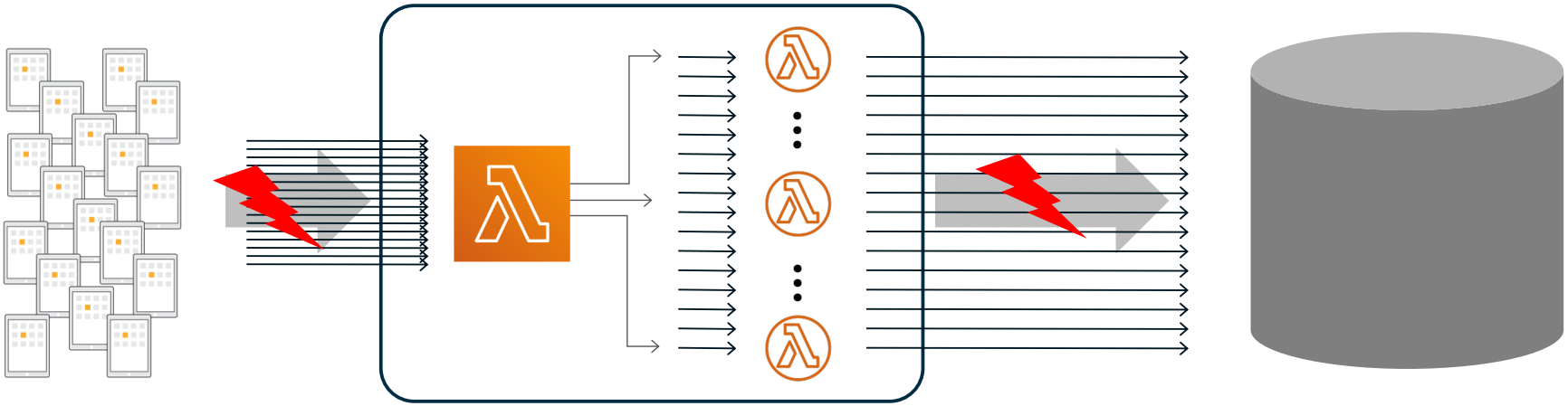
1. 呼び出し要求に応じて
   ファクションを実行
   （処理リソースを確保）
2. リクエスト増加
   → 必要に応じて処理
   リソースを追加確保

不適切なリソース利用の防止
= 同時実行数による制限・保護
（デフォルト1,000） 2021/09現在

効率的なリソース再利用の促進
= 処理タイムアウトの設定
（最大15分） 2021/09現在

サーバーレスにおけるデータベース





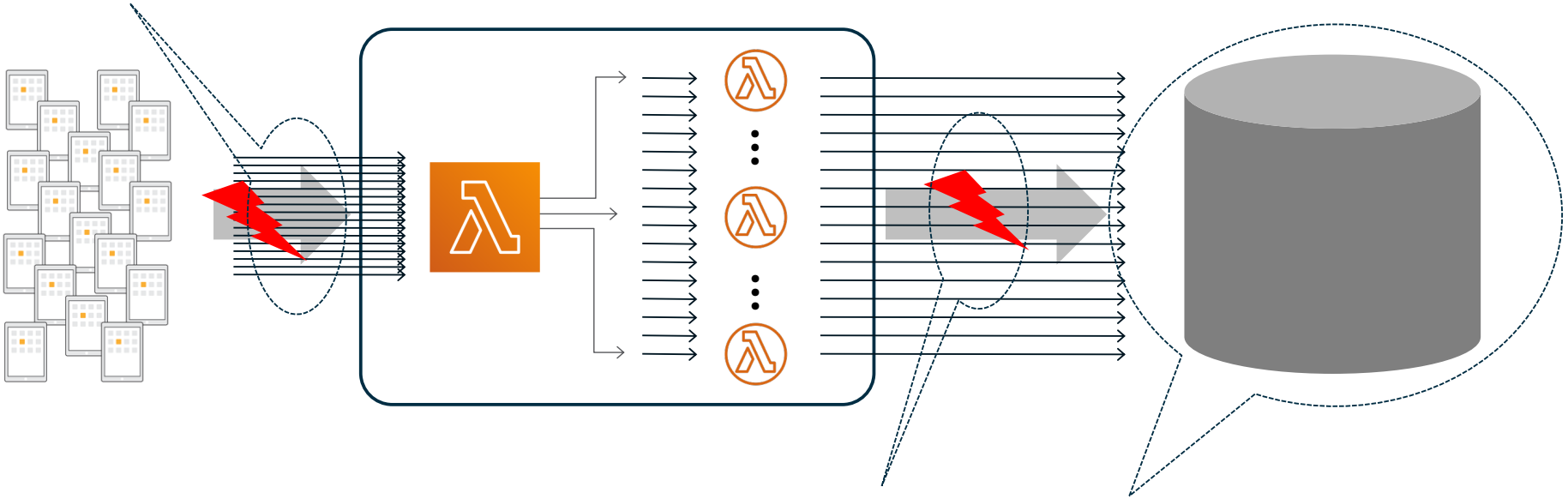
大量のDB接続リクエストが発生する可能性!!

＝

RDB 側のリソースがパンクする

サーバーレスにおけるデータベース

1 同時リクエスト数がそこまで高くない or
入り口でスロットリングする

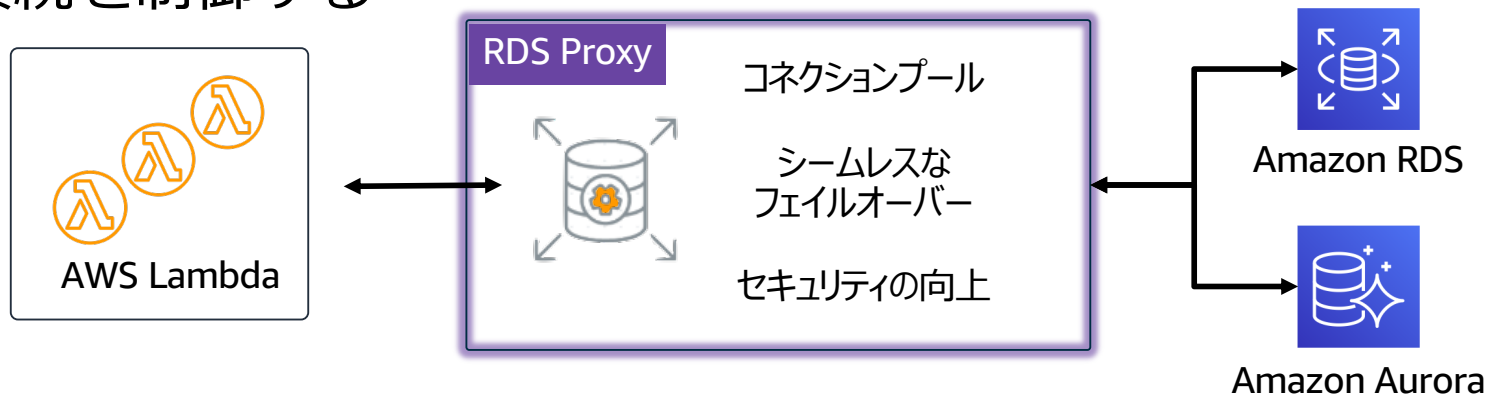


2 DB接続を
制御する

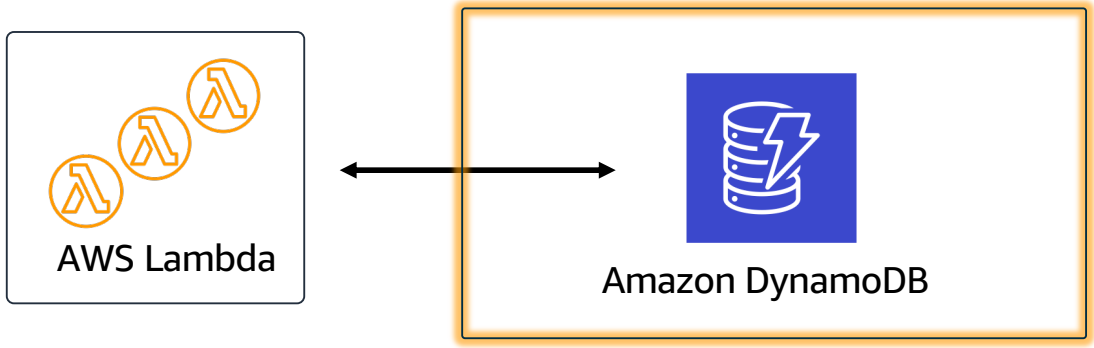
3 分散型の
DBを選ぶ

サーバーレスにおけるデータベースの選択

2 DB接続を制御する



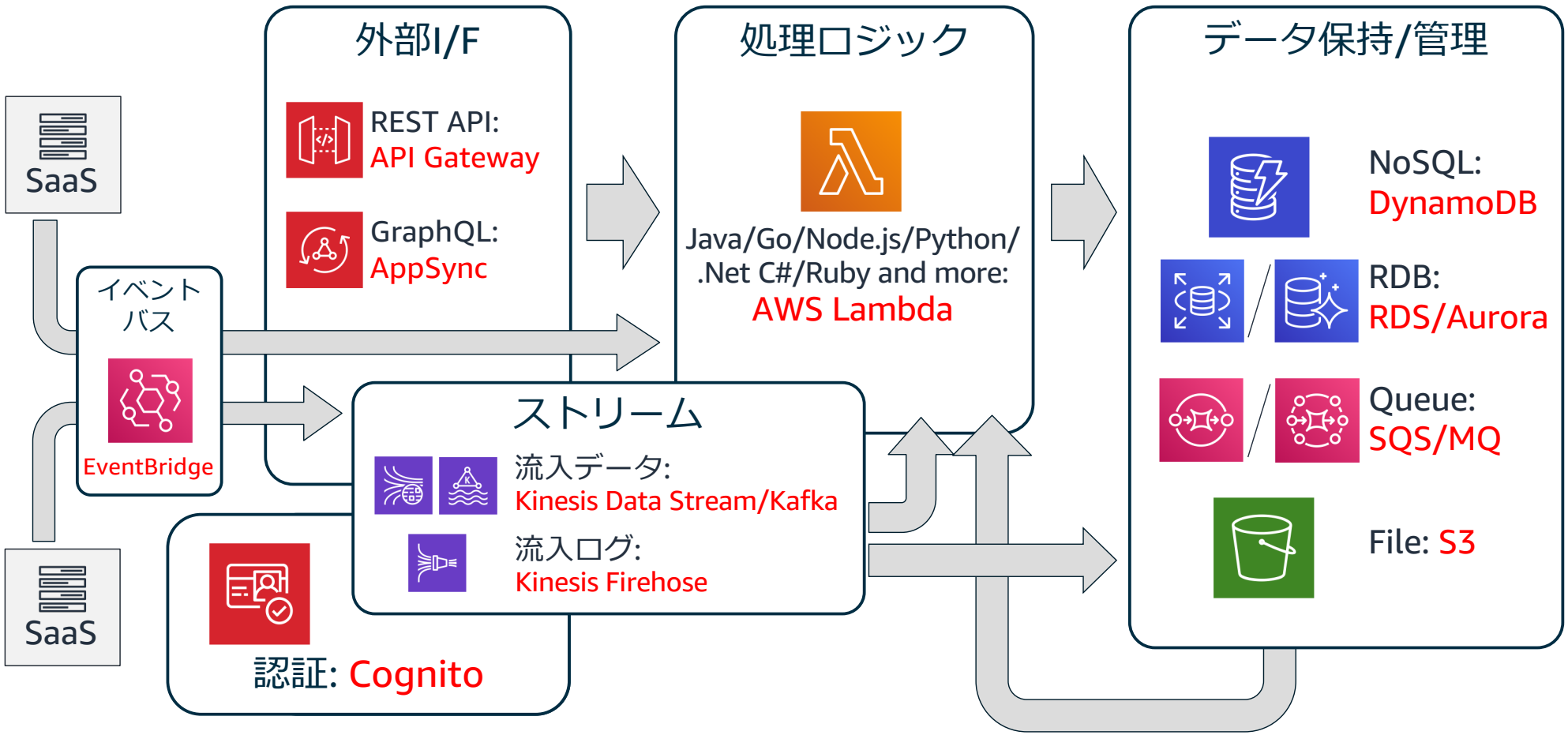
3 分散型の DBを選ぶ



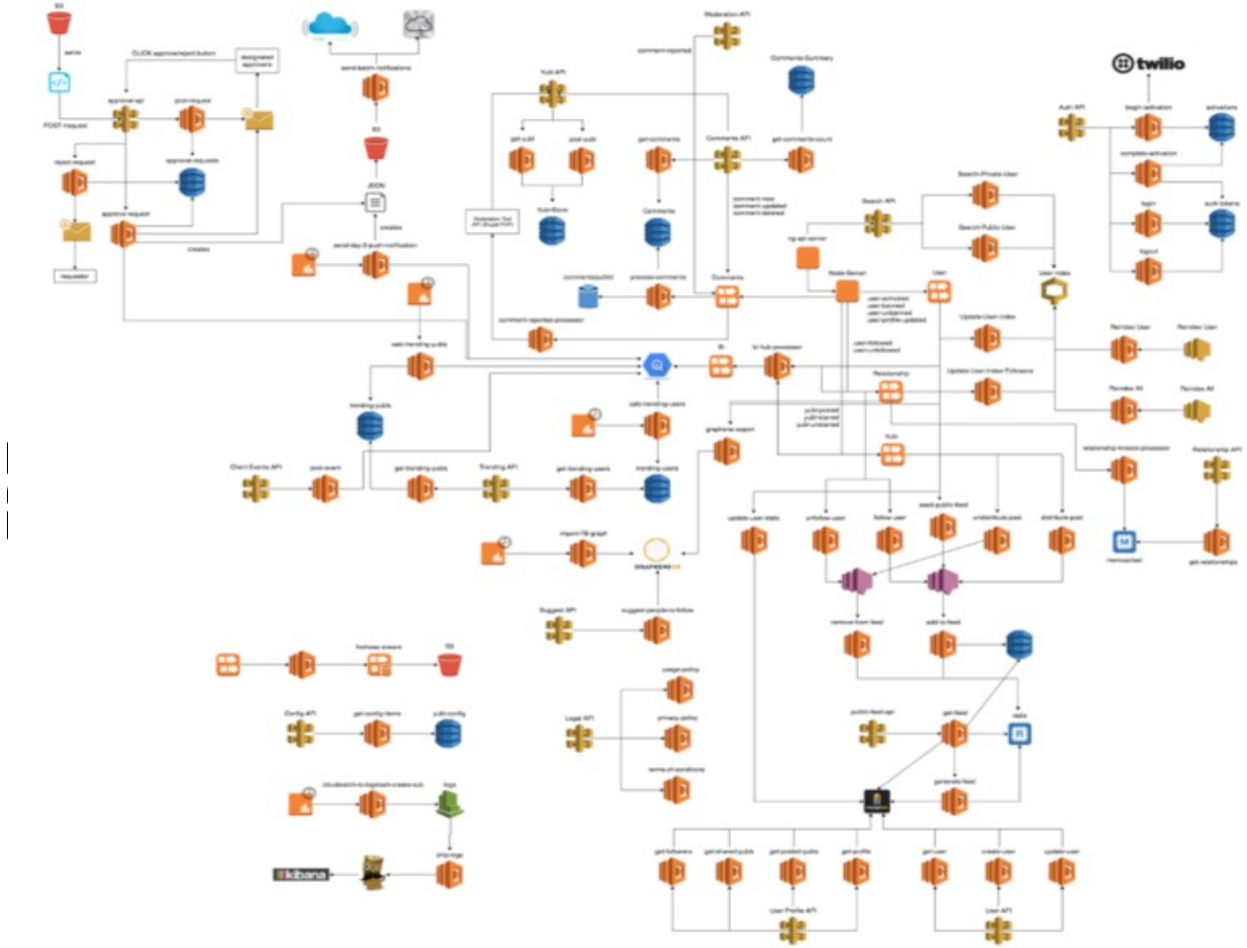
考慮点
- アプリから見た I/F
  - SQL? API?
- データ設計
  - RDB型? Key-Value?
- DBの可用性設計/
  スケーラビリティ


パターンで考える

サーバーレス: 主要コンポーネント



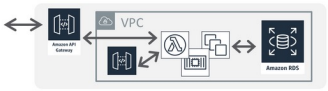
サーバーレス設計の行き着く先 ... 複雑化?



実戦でよく使われるユースケースパターン

動的 Web / モバイルバックエンド
関連資料 | Tutorial | Tutorial（中級編）
テンプレートから始める → こちら

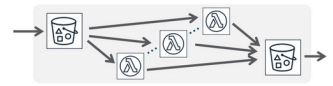
リアルタイムモバイル / オフライン対応
AWS マンガ | **[New]** ワークショップ
[New] Solution リンク



業務系 API / グループ企業間 API
Tutorials | Private API 記事 | 関連事例
OpenAPI の利用（REST | HTTP API）

Push 配信系・インタラクティブ API
関連リンク | AppRepositoryサンプル
解説動画（英語）

画像処理 / シンプルなデータ加工
Tutorial | 関連事例 | Solution リンク
テンプレートから始める → こちら



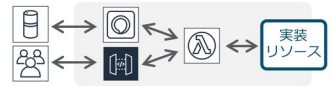
分散並列処理（like MapReduce）
関連事例1 | 関連事例2
RefArch

イベント駆動の業務処理連携
SNS-SQS Tutorial | SQS-Lambda 連携
テンプレートから始める → こちら

アプリケーションフロー処理
Tutorial（Workflow / エラー処理）
短時間・高速フロー処理向け Express

流入データの連続処理
関連資料 | RefArch
Tutorial1 | Tutorial2

IoT バックエンド
関連資料 | 関連事例 | RefArch
関連 Solution1 | Solution2



チャットボット / Alexa スキル
Alexa スキル開発 | RefArch
Solution リンク

データ変更トリガー処理
活用例 | Tutorial

ログデータ収集処理
関連事例 | Solution1 | **[New]** Solution2
データ変換ブループリント

データレイク周りのデータ加工
[New] Solution リンク | DB Loader
or より包括的なソリューション

機械学習/ETLデータパイプライン
関連記事1 | 関連記事2
機能紹介動画 | 関連事例

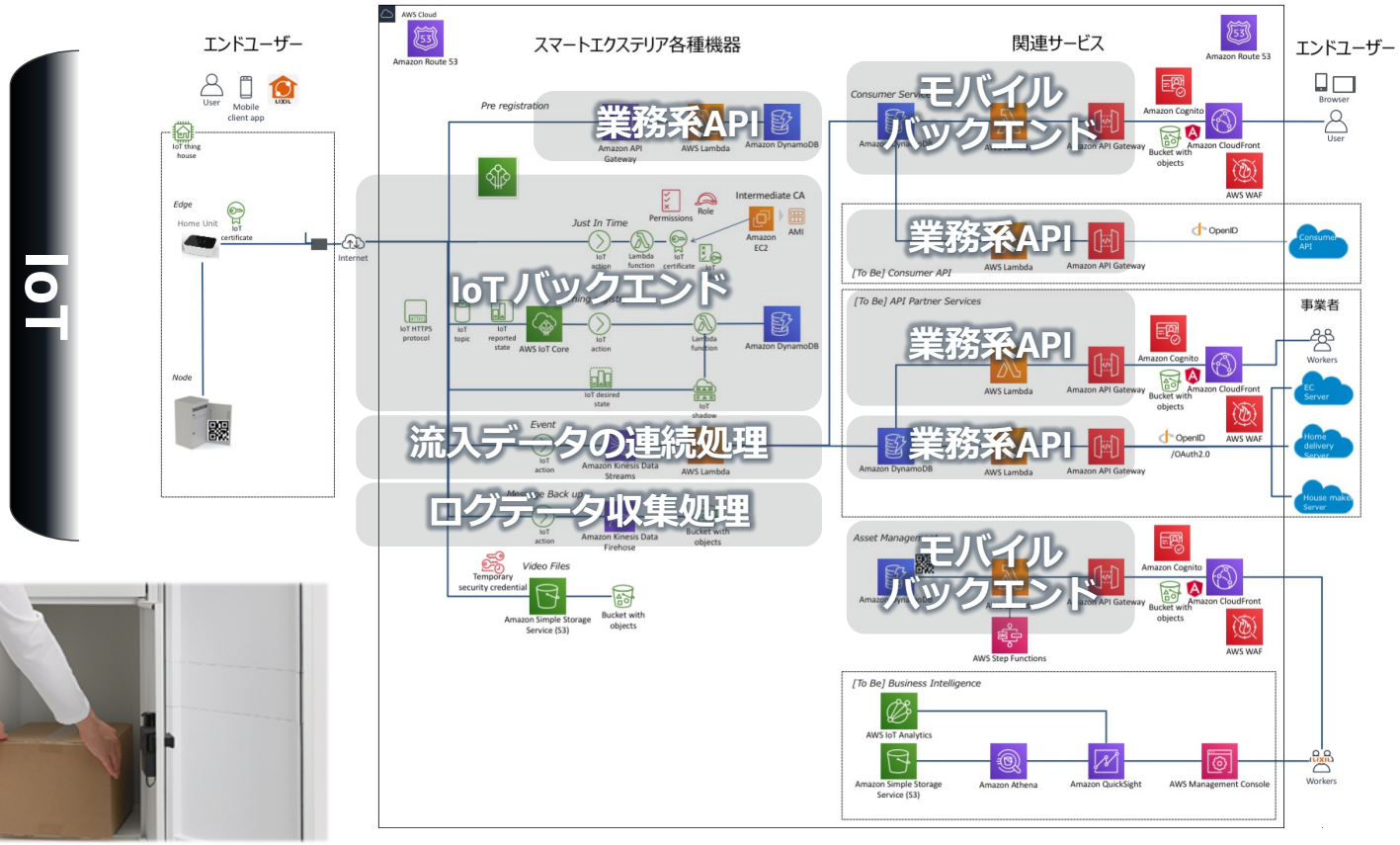
スケジュール・ジョブ / SaaS イベント
関連 Doc / Template | 関連 Tutorial
テンプレートから始める → こちら

サーバーレス パターン

https://aws.amazon.com/jp/serverless/patterns/serverless-pattern/

LIXIL 様
スマート宅配ポストサービス

変更容易性　マネージド リソース自動管理　コスト最適化



NTT東日本様

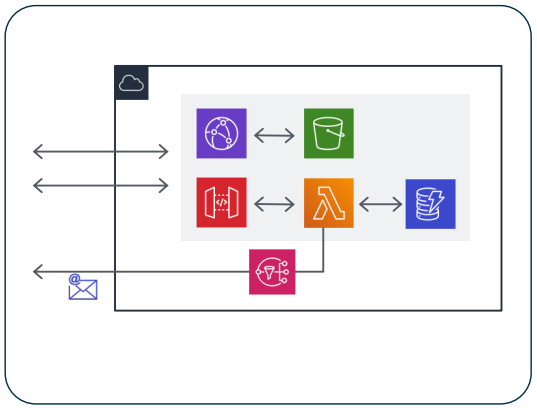
AWS DevDay 2020 にて講演

コロナウィルス影響で
急務のシステム依頼 !!

本格
検討へ

今回はこれを選択

実装

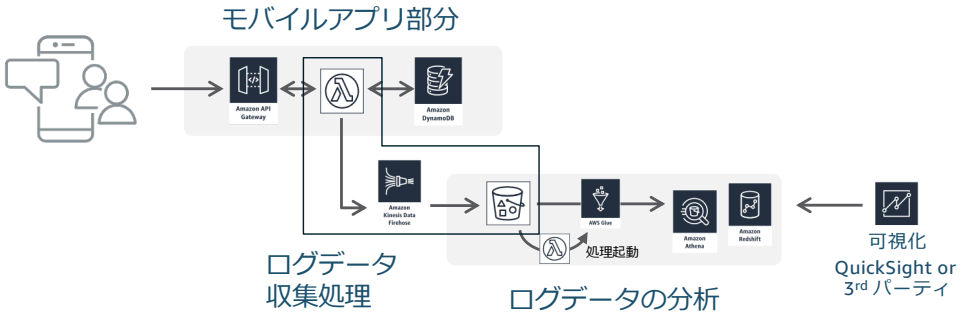


- 勉強会で爆速開発の
  話を聞いていた
- Webで同内容を確認できた

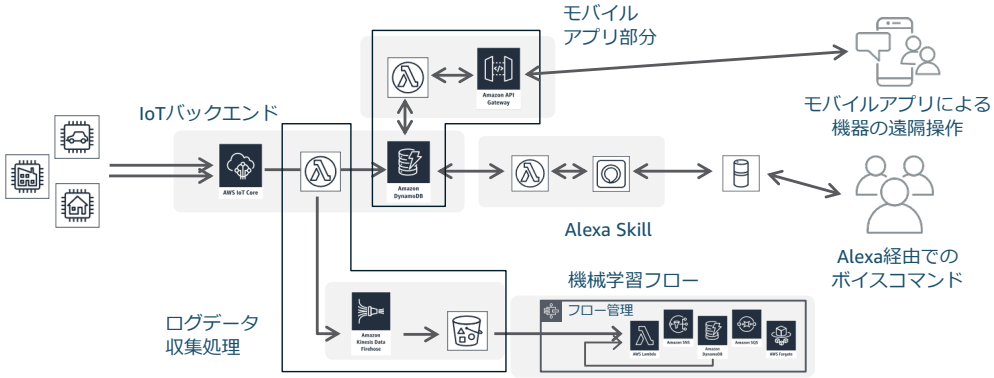
- Webでやりたいこと/形を選び
  それをベースに実装を開始

- 5日で実装、緊急リリース
  (+SNS上限緩和実施)

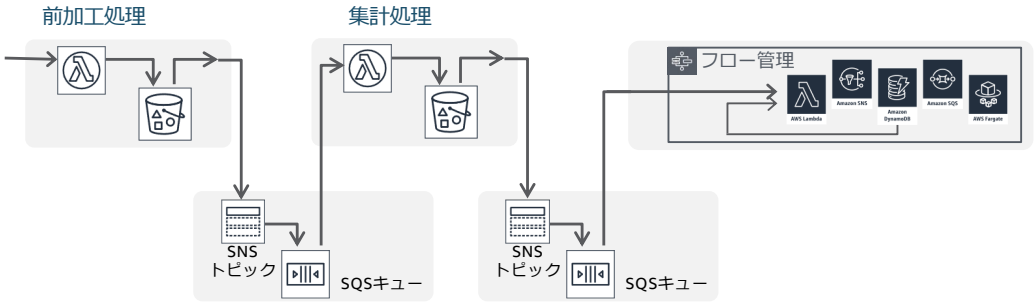
パターン組み合わせの例



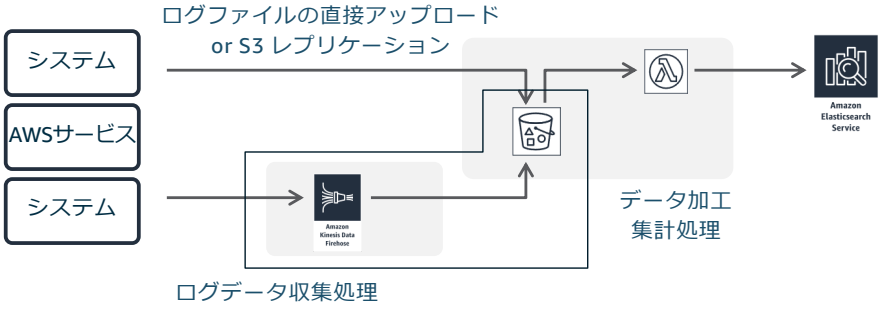
モニタリング型モバイルアプリ



IoTアプリケーション



データ連携処理パイプライン



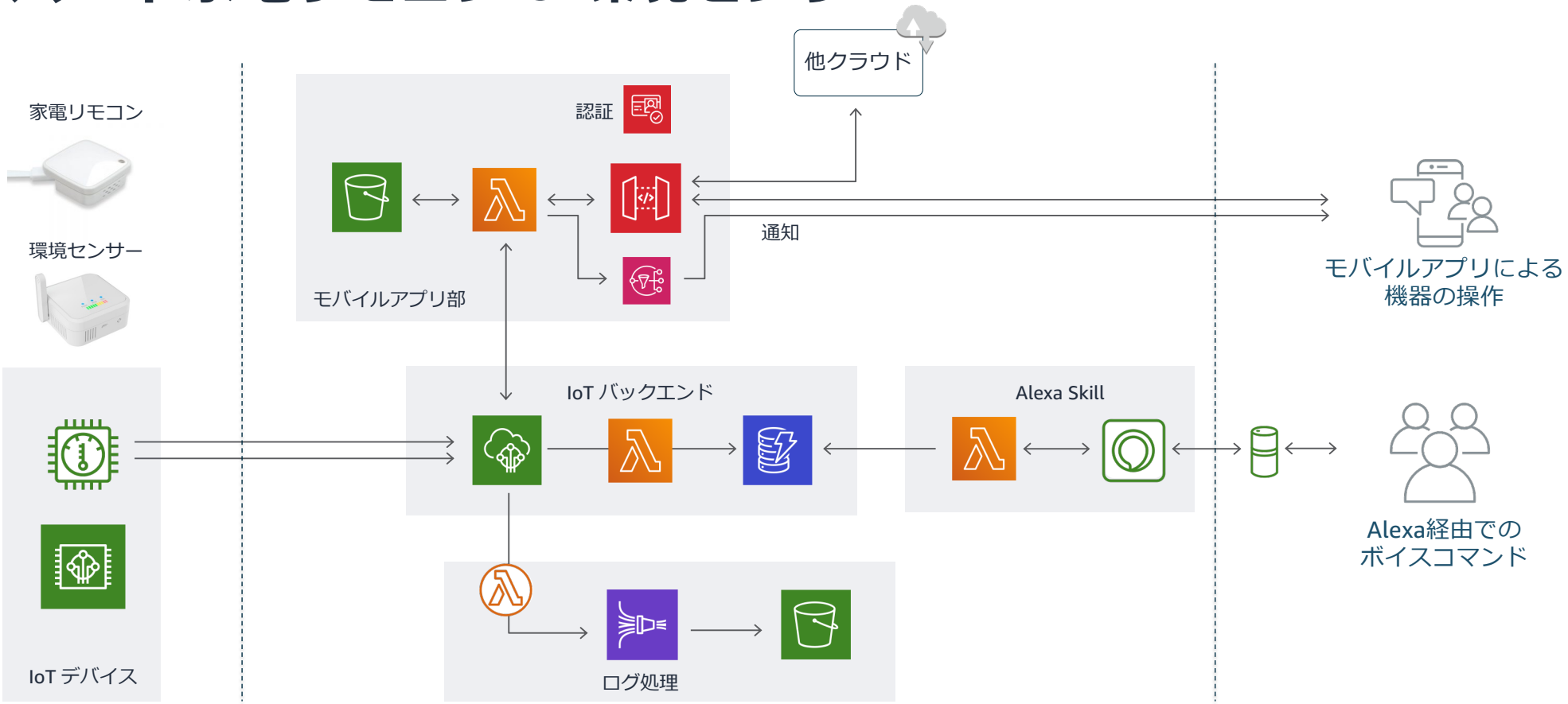
ログデータ処理、SIEM

ラトックシステム様
スマート家電リモコン & 環境センサー

マネージド
業務注力

スケーラビリティ
（機会損失防止）

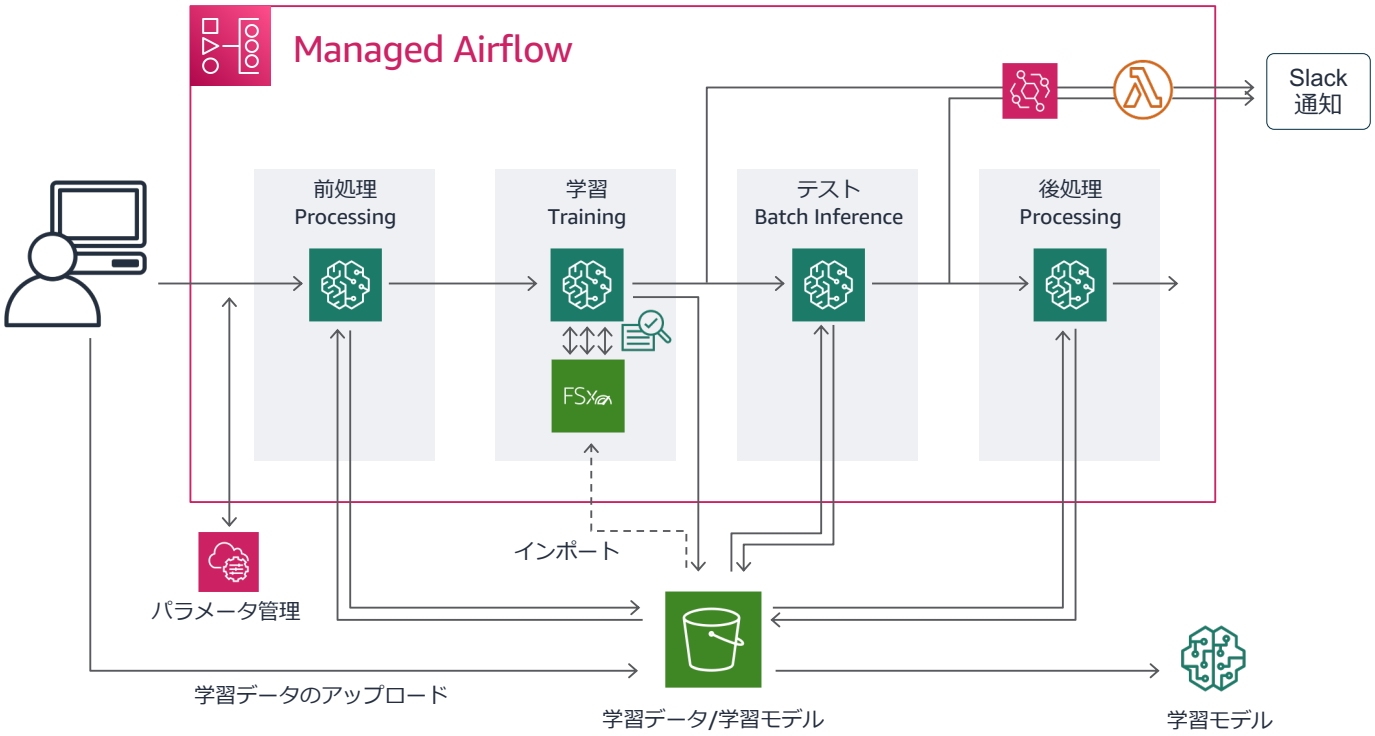
コスト最適化



コナミデジタルエンタテインメント様
機械学習パイプラインの自動化

マネージド
業務注力

マネージド
自動リソース管理



- 写真画像の判定/分類モデルの構築
  - 写真撮影された画像から種別を自動判定するモデルを学習・構築
  - 全体のML処理フローに Managed Airflow を活用（Airflow on EC2 からの移行）
    - ✓ Airflow 環境構成とNW配置を適正化
      - 構成検討工数を削減
      - 踏み台サーバーの排除
      - セキュアに Airflow UI 構成
    - ✓ Managed → 運用の属人化の排除

サーバーレステクノロジご利用の国内のお客様（抜粋）

画像加工処理

NIKKEI / ZOZO Technologies / Speee / Fleekdrive by ソルクシーズ / LightFile by アイデアマンズ

モバイル/Web バックエンド

顧客向け

LIFULL / ADASTRIA / WarranteeNow / CARTE by CyberAgent / cookpad TV / ダイスポ LOVE SPORTS

従業員向け

YAMAHA Revs Your Heart / Samantha Thavasa

サービスAPI/ 業務機能API

毎日新聞 / 朝日新聞 / 産経デジタル SANKEI DIGITAL INC. / SUNTORY SYSTEM TECHNOLOGY SUNTORY / BANDAI NAMCO Entertainment / ぐるなび / Game Server Services / kabu.com カブドットコム証券 / ALIS

業務データ処理（POS/商品/在庫...）

TOKYU HANDS / DAISO ダイソー

バッチ・分散並列処理

WORKS APPLICATIONS / Fringe / カオナビ

ストリーム処理

Zucks / cookpad / ABC

SaaS/ ソリューション基盤

ThemiStruct テミストラクト by オージス総研 / iret cloudpack

ログ処理/ サービス監視

RICOH / SmartNews / TOKIO MARINE NICHIDO 東京海上日動 / U-NEXT / GREE

データ前処理/機械学習パイプライン処理

sansan / Gunosy / トリップAIコンシェルジュ by リクルートライフスタイル

IoT関連 ユースケース

J!NS / SHARP / LIXIL / DAIKIN / OMRON / RATOC Systems, Inc. / e-kakashi by PSソリューションズ / Farmnote / 来栖川電算 Kurusugawa Computer Inc.

まとめ

サーバーレス型のアプリケーション設計

- やりたいことに合いそうなユースケースパターンから検討してみる

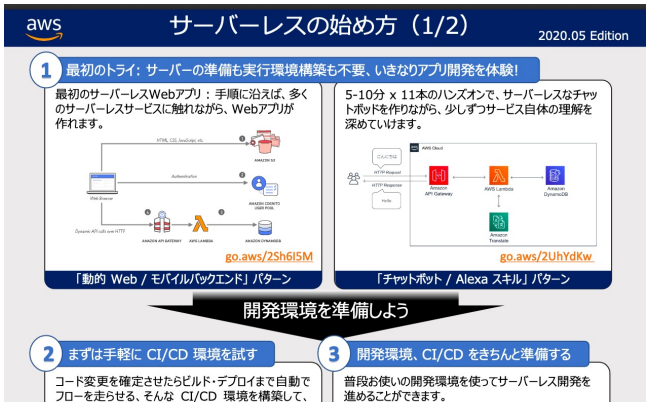
サーバーレス検討に向いている領域

- 新規プロジェクト、DXプロジェクト
- 既存システムに対する機能追加部分
- システム連携を担うデータ連携部分、
  バッチ処理の分解型実装

aws

サーバーレスを始める方のための情報源

これから始める人の
ための自習ガイド
- ハンズオン
- 典型的な 6 Step



amzn.to/2ZGL3ZS

アーキテクト向け
ユースケースパターン
- 16 のパターン
- 組み合わせ活用例

amzn.to/2UJT4bB

サーバーレス パターン

これからの開発者向け
サーバーレス技術情報
- ハンズオン・技術資料
- 開発環境、Tips...

amzn.to/2WeZuQl

サーバーレス 始めよう