AWS Builders Online Series

サーバー確保型からサーバーレスへ
～ サーバーレスを理解する最初の一歩 ～

杉 達也

アマゾン ウェブ サービス ジャパン合同会社
AWS プラットフォーム事業開発部
シニア事業開発マネージャ サーバーレス担当

aws

- 杉 達也
  - アマゾン ウェブ サービス
    事業開発マネージャ（サーバーレス）
- 仕事: サーバーレスの普及および
  お客様にとっての障壁となることを
  理解し、解決のために働きかける
- AWS入社以前は外資ソフトウェアベンダーで
  Java関連製品のビジネス開発に従事

サーバーレスへの期待・注目が高まっている背景

- **DX 推進へのプレッシャー**
  - 対面型ビジネスの見直し、新しいデジタルチャネル
  - 通知・伝達のデジタル化
  - B2B 連携強化

- **従来型のシステムに対する課題感**
  - 変化対応力の課題
  - 競争力強化の必要性
  - 2025年の崖

感染症

働き方変革

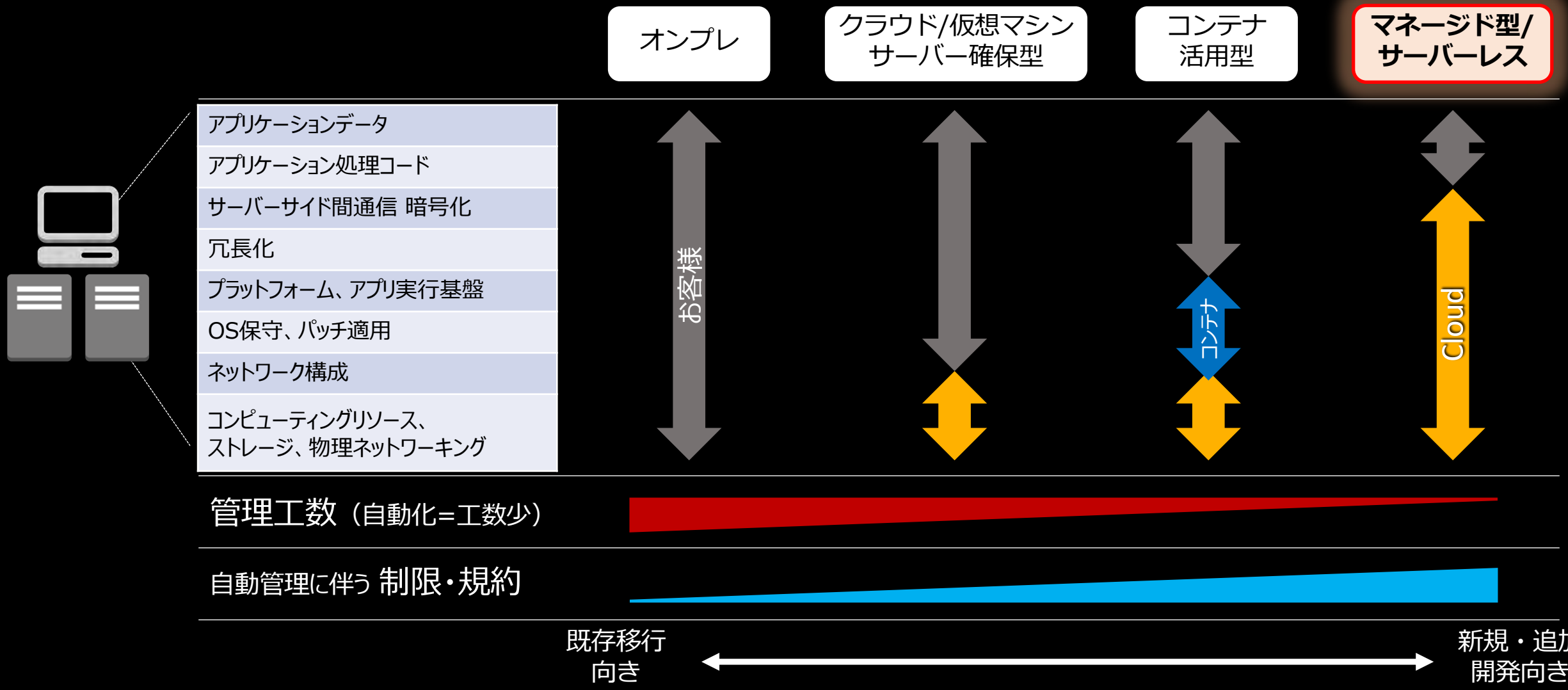
世界情勢

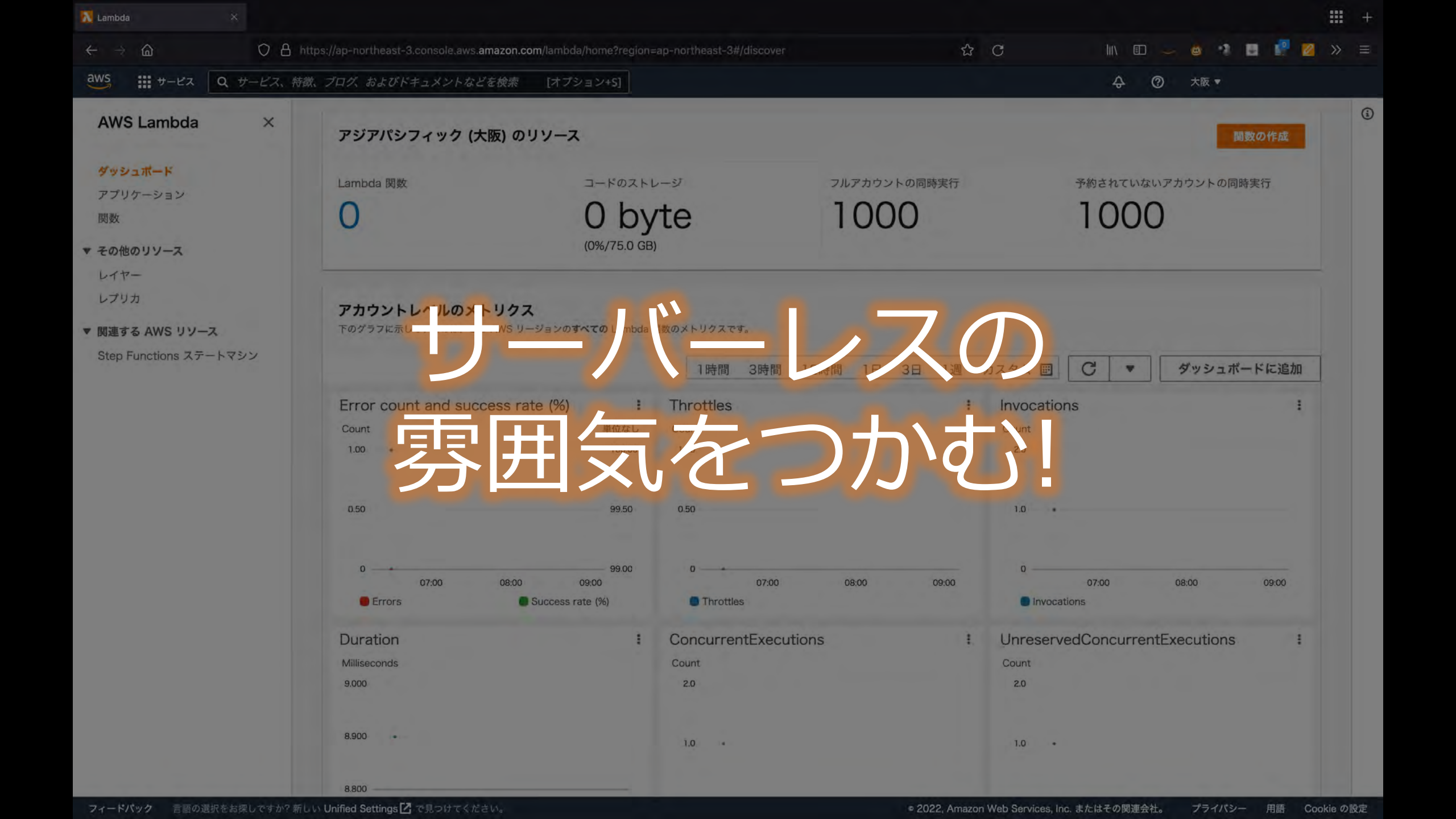
サーバーレスとは

サーバーがない?

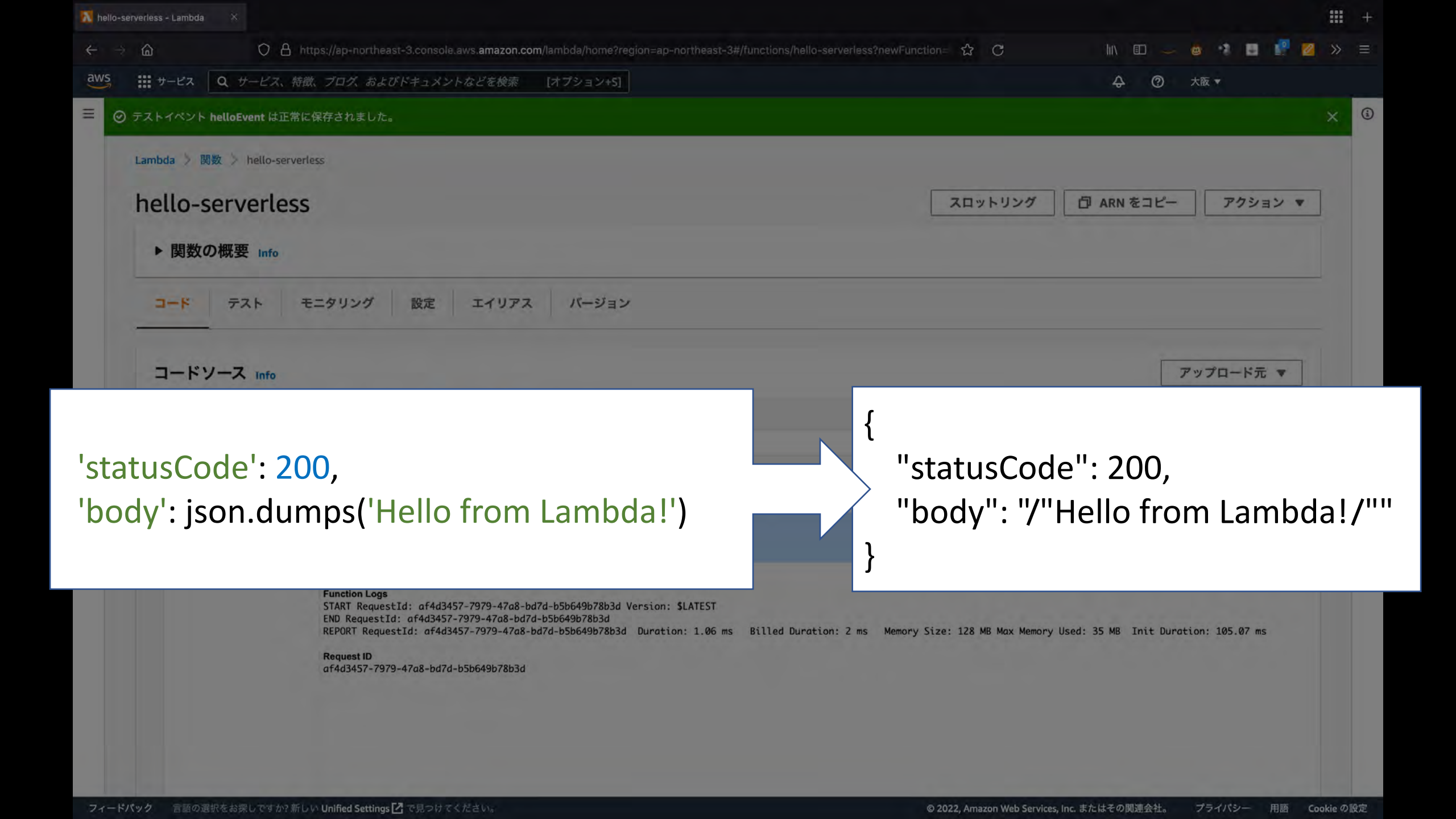
サーバーの存在を意識しない

サーバーレスとは

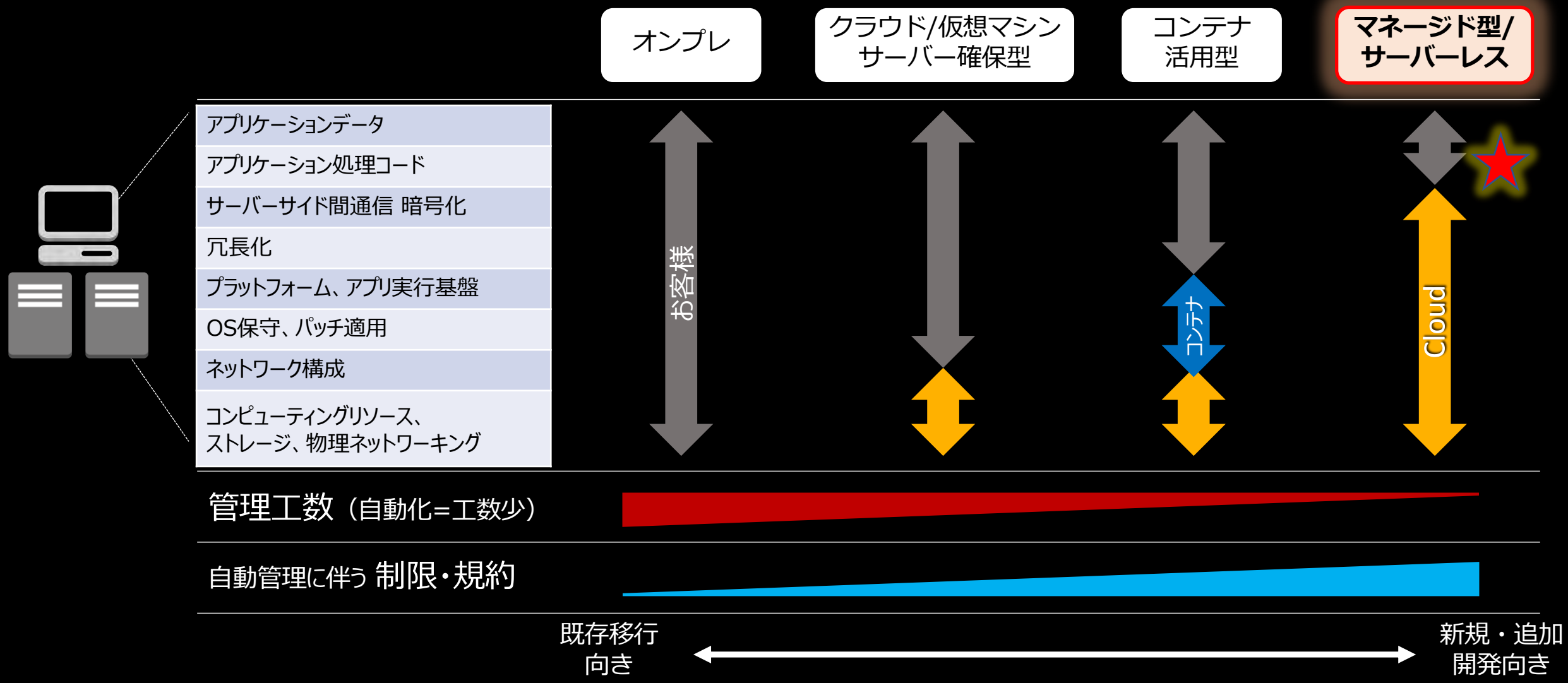




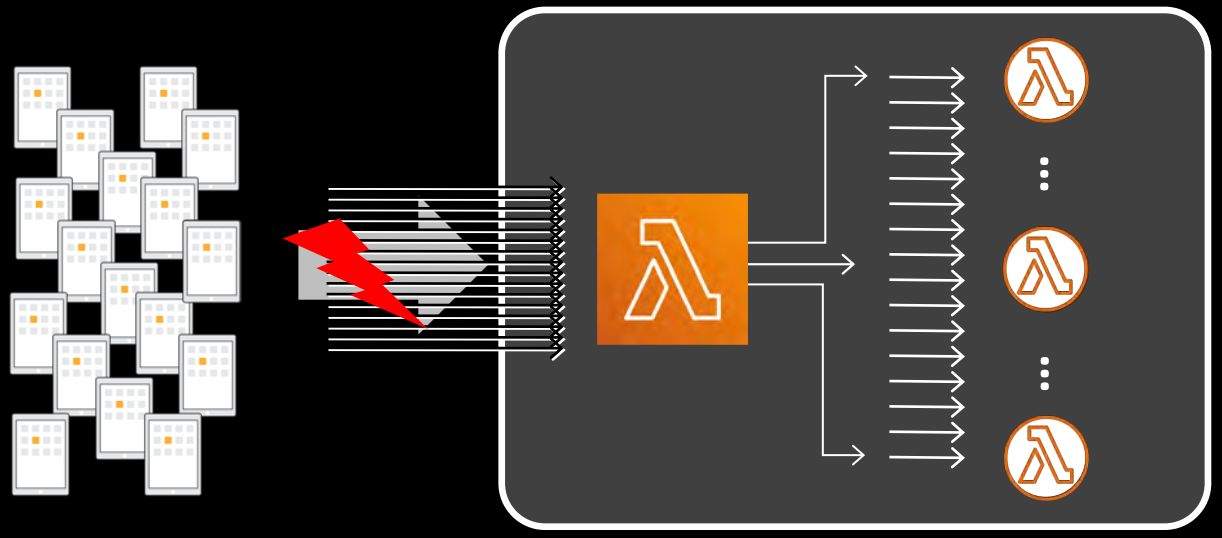
サーバーレスの雰囲気をつかむ!


hello-serverless - Lambda
https://ap-northeast-3.console.aws.amazon.com/lambda/home?region=ap-northeast-3#/functions/hello-serverless?newFunction=
aws
サービス
サービス、特徴、ブログ、およびドキュメントなどを検索
[オプション+S]
大阪
テストイベント helloEvent は正常に保存されました。
Lambda > 関数 > hello-serverless
hello-serverless
スロットリング
ARN をコピー
アクション
関数の概要 Info
コード
テスト
モニタリング
設定
エイリアス
バージョン
コードソース Info
アップロード元
'statusCode': 200,
'body': json.dumps('Hello from Lambda!')
{
"statusCode": 200,
"body": "/"Hello from Lambda!/""
}
Function Logs
START RequestId: af4d3457-7979-47a8-bd7d-b5b649b78b3d Version: $LATEST
END RequestId: af4d3457-7979-47a8-bd7d-b5b649b78b3d
REPORT RequestId: af4d3457-7979-47a8-bd7d-b5b649b78b3d Duration: 1.06 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 35 MB Init Duration: 105.07 ms
Request ID
af4d3457-7979-47a8-bd7d-b5b649b78b3d
フィードバック
言語の選択をお探しですか? 新しい Unified Settings で見つけてください。
© 2022, Amazon Web Services, Inc. またはその関連会社。
プライバシー
用語
Cookie の設定

サーバーレスとは [再掲]



実際のリクエスト量に応じて対応

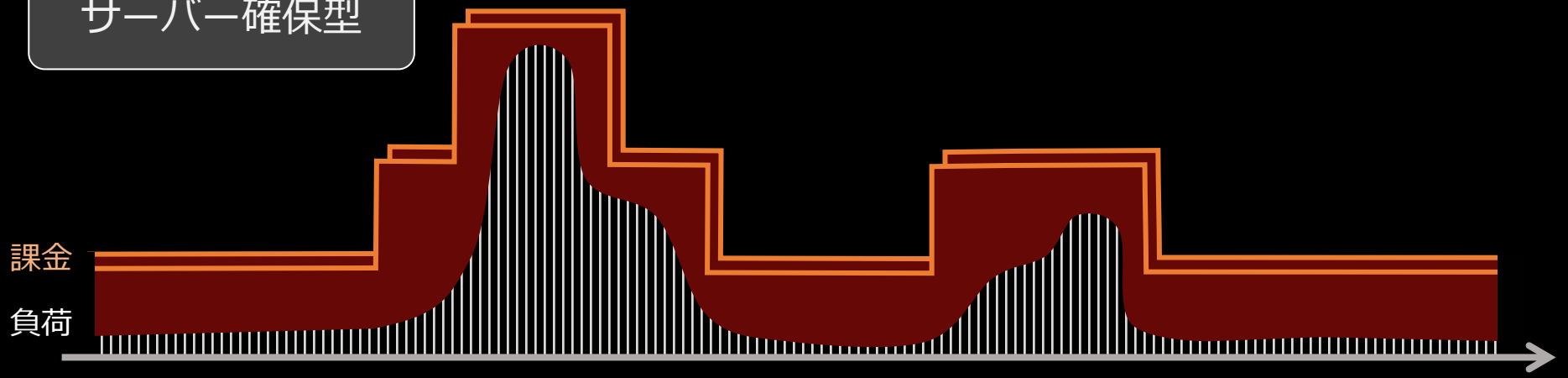


- リクエストがなければ
  ゼロ利用
- 1リクエストきたら
  それを処理
- 大量に来たら
  その分を処理する
  インスタンスを自動起動

処理に必要な実行リソースの確保はクラウドにおまかせ

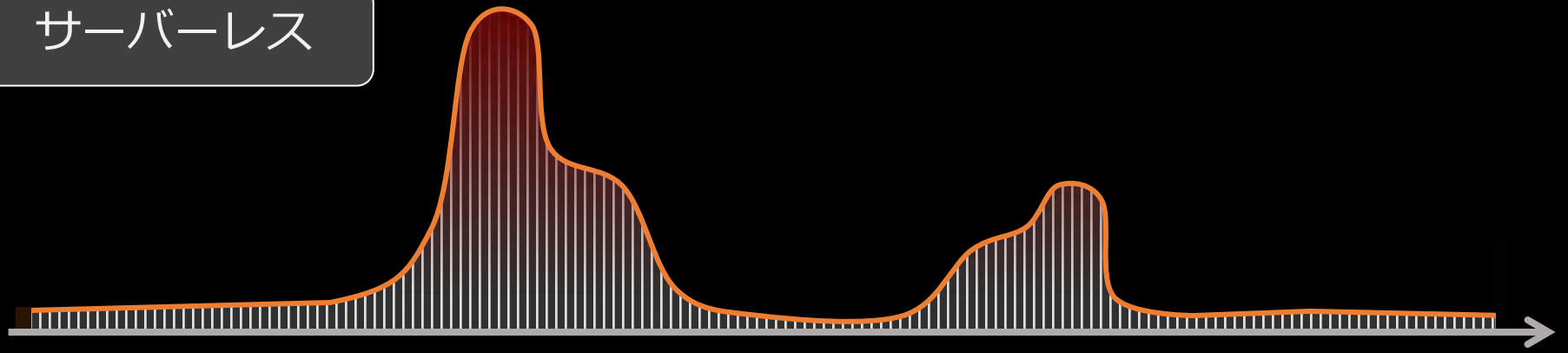
サーバーレスの利用費の構造

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期待と結果

初期段階の思惑

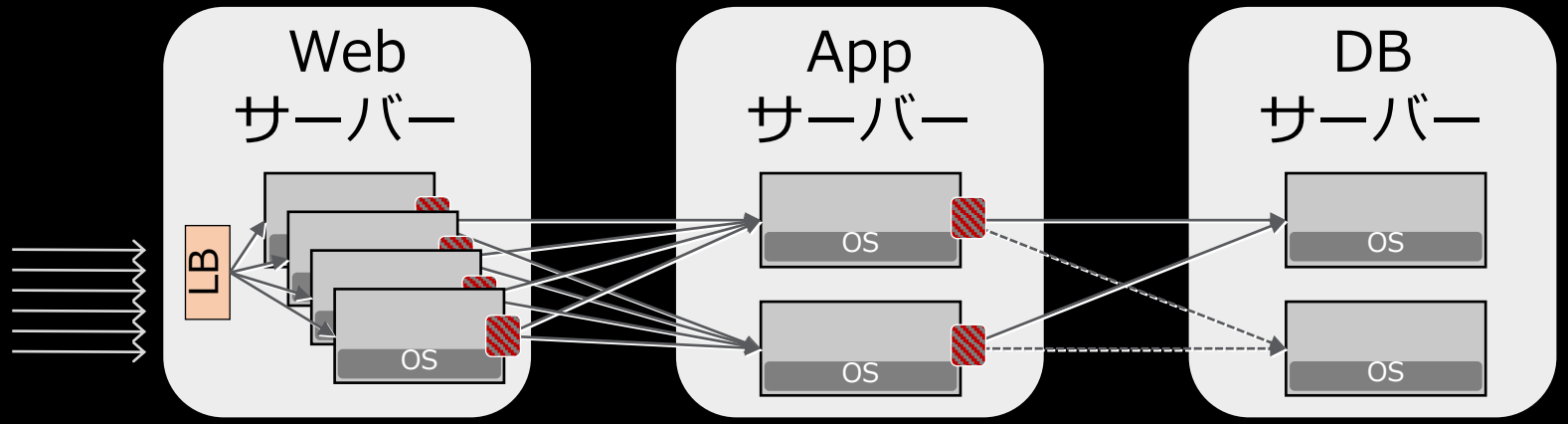
- 利用費を下げたい
  - 余分に確保している部分のサーバー費用をなくしたい

実際に経験した方の声

- 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
- バージョンアップ作業が減る

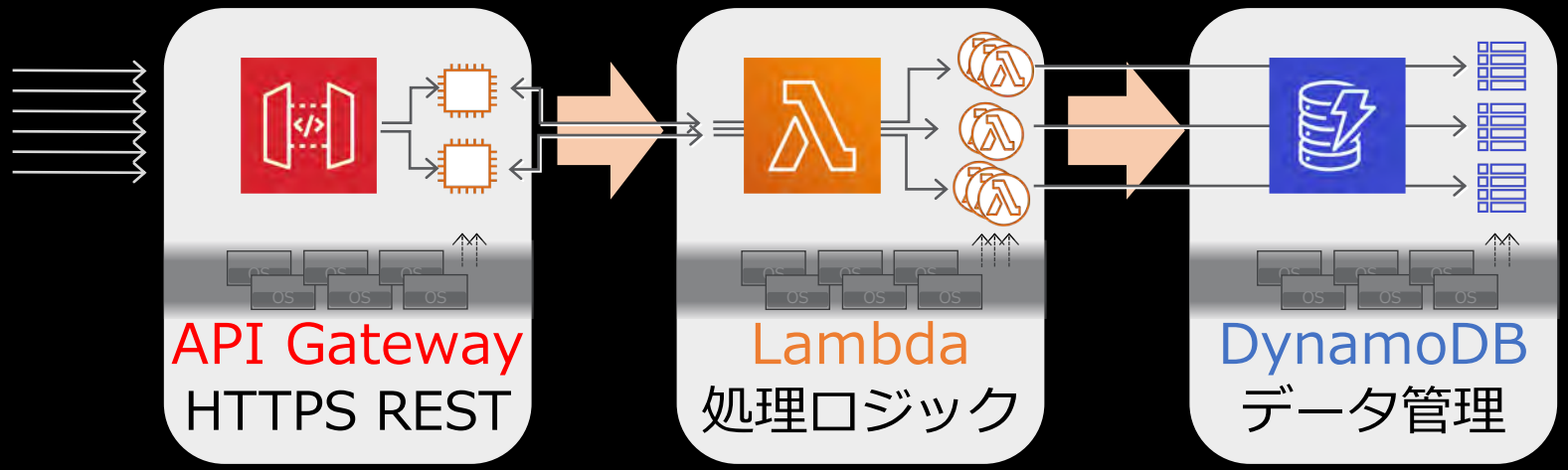
- 開発・生産的な作業時間が増える
- 改善サイクルが回るようになる

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

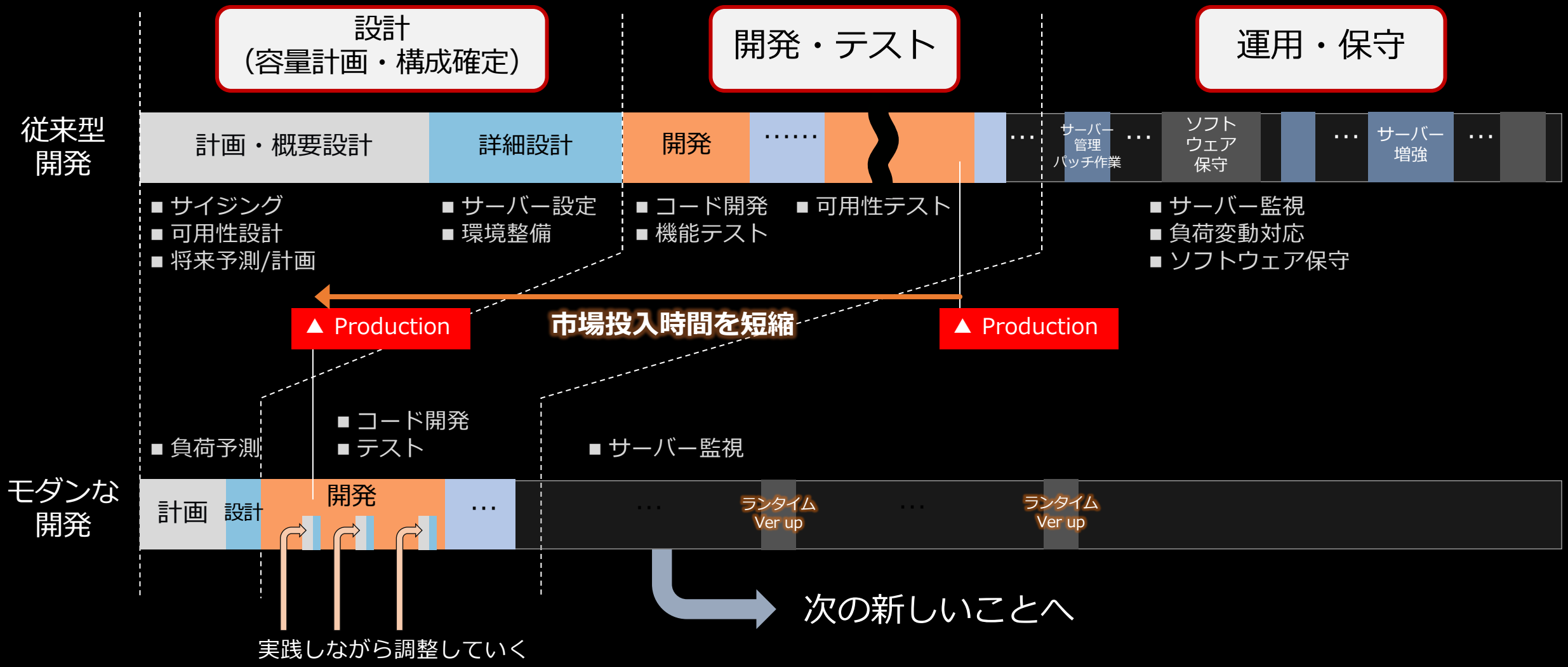
+ 規模の見積もり
+ 可用性設計
  ロードバランス設定
  (LB + 割り振り設定)
+ データ保全の検討



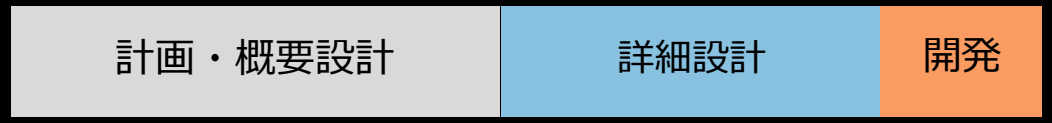
~~サーバー/OSの準備・構成~~
設定・開発作業

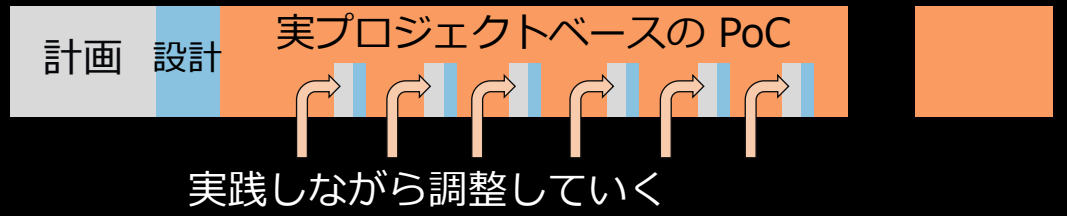
✓ リクエスト量に応じて
  自動スケール
✓ 設計済みのリトライ
  マルチ AZ 構成済
✓ データ可用性

プロジェクトのどの工程に効いてくるのか?



マネージド/サーバーレスによる変化





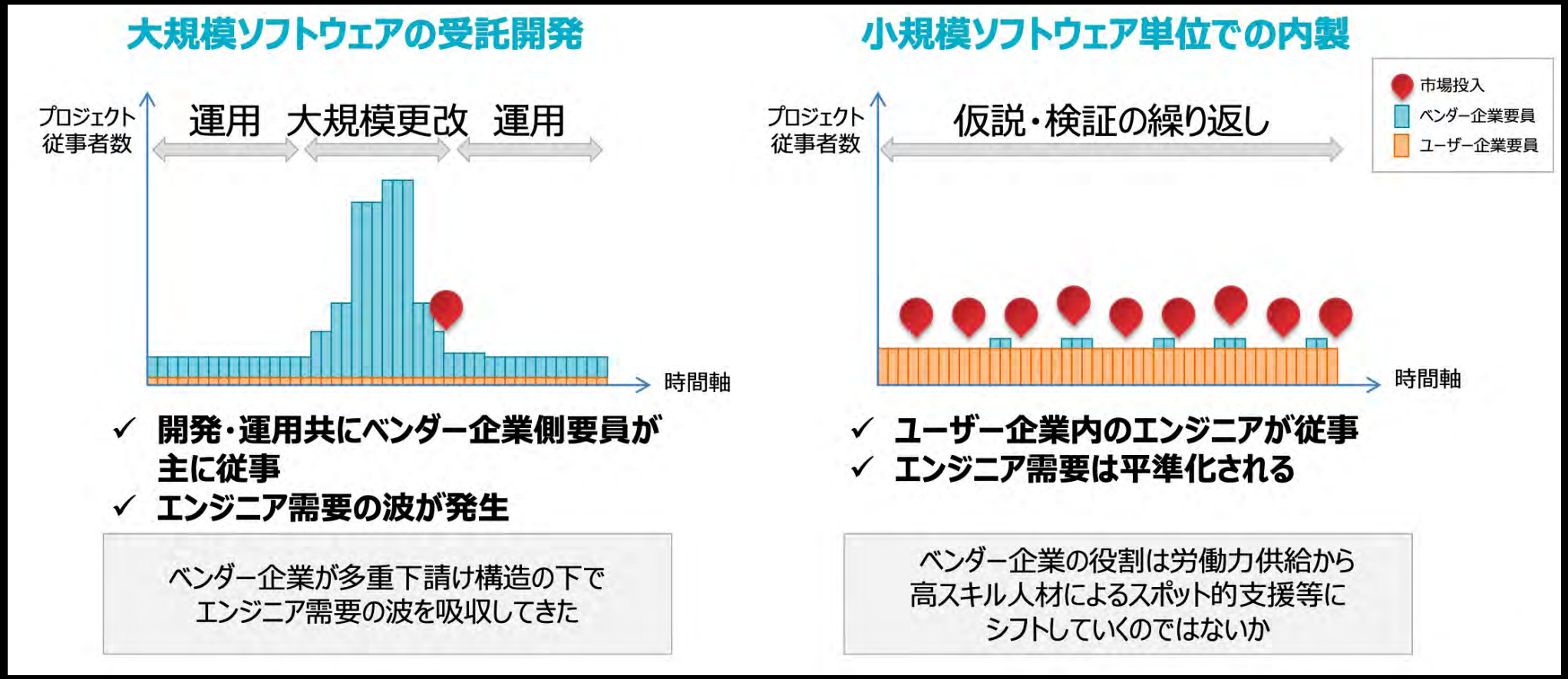
従来型

- ウォーターフォール
- 十分な検討を経て実施
- アプリとインフラチーム
- 利用費: 確保型
- システム品質: それぞれに確保

クラウドネイティブ型

- アジャイル
- Try & Error
- アプリチームの拡充化
- 利用費: 実績ベース
- システム品質: 一定レベルを担保

参考: DX 推進に向けた推奨事項

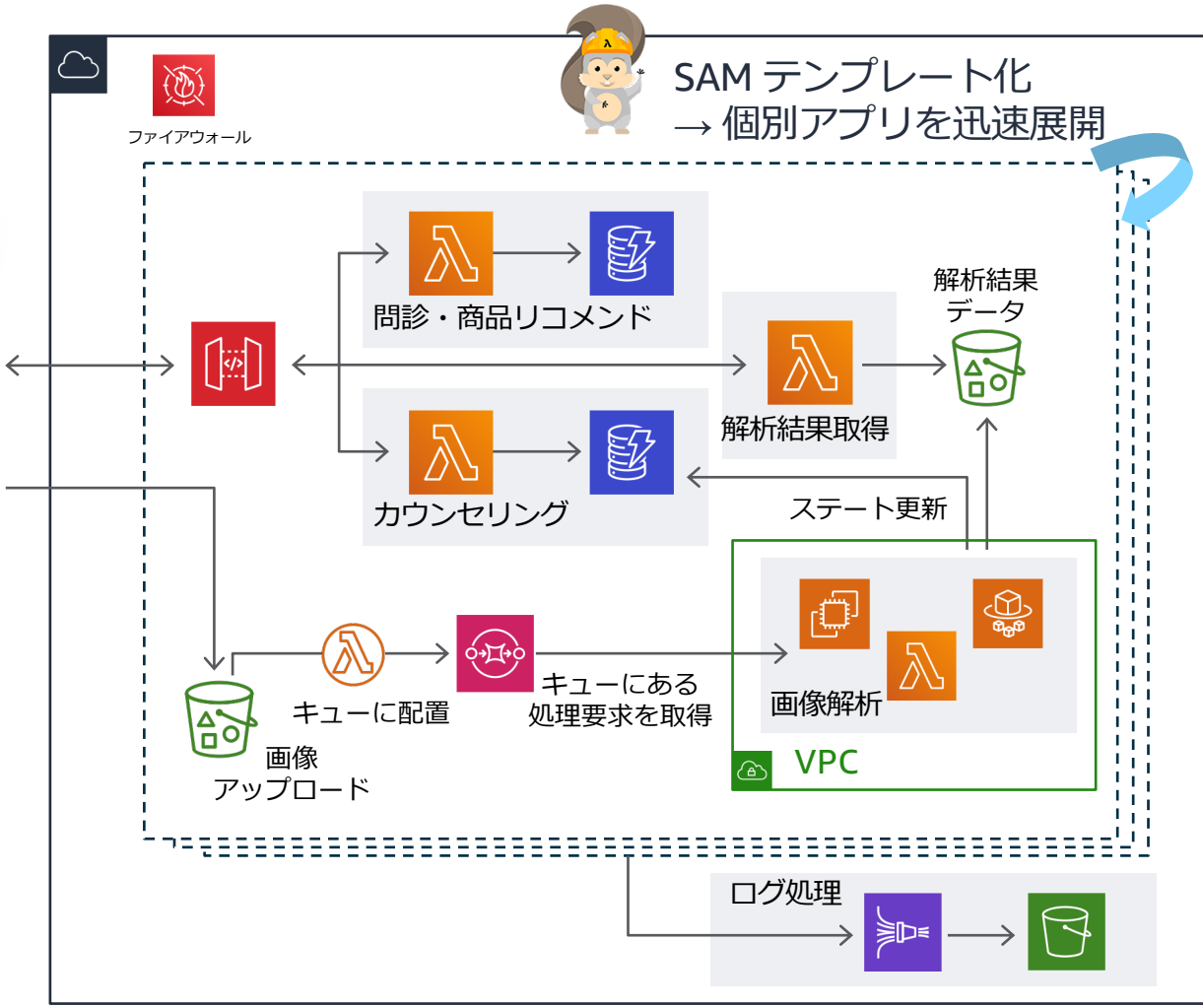


出典「DXレポート2」 経済産業省 2020/12/28

コーセー様
複数の個別カウンセリングアプリを短期間で開発

コーセー
HADA mite

SEKKISEI
透明感測定
：



顧客体験向上のための診断機能付き
個別アプリを**半年で 7 本**リリース

高生産性

共通機能を SAM テンプレート化し
個別アプリをそれぞれ **1、2ヶ月**で構築

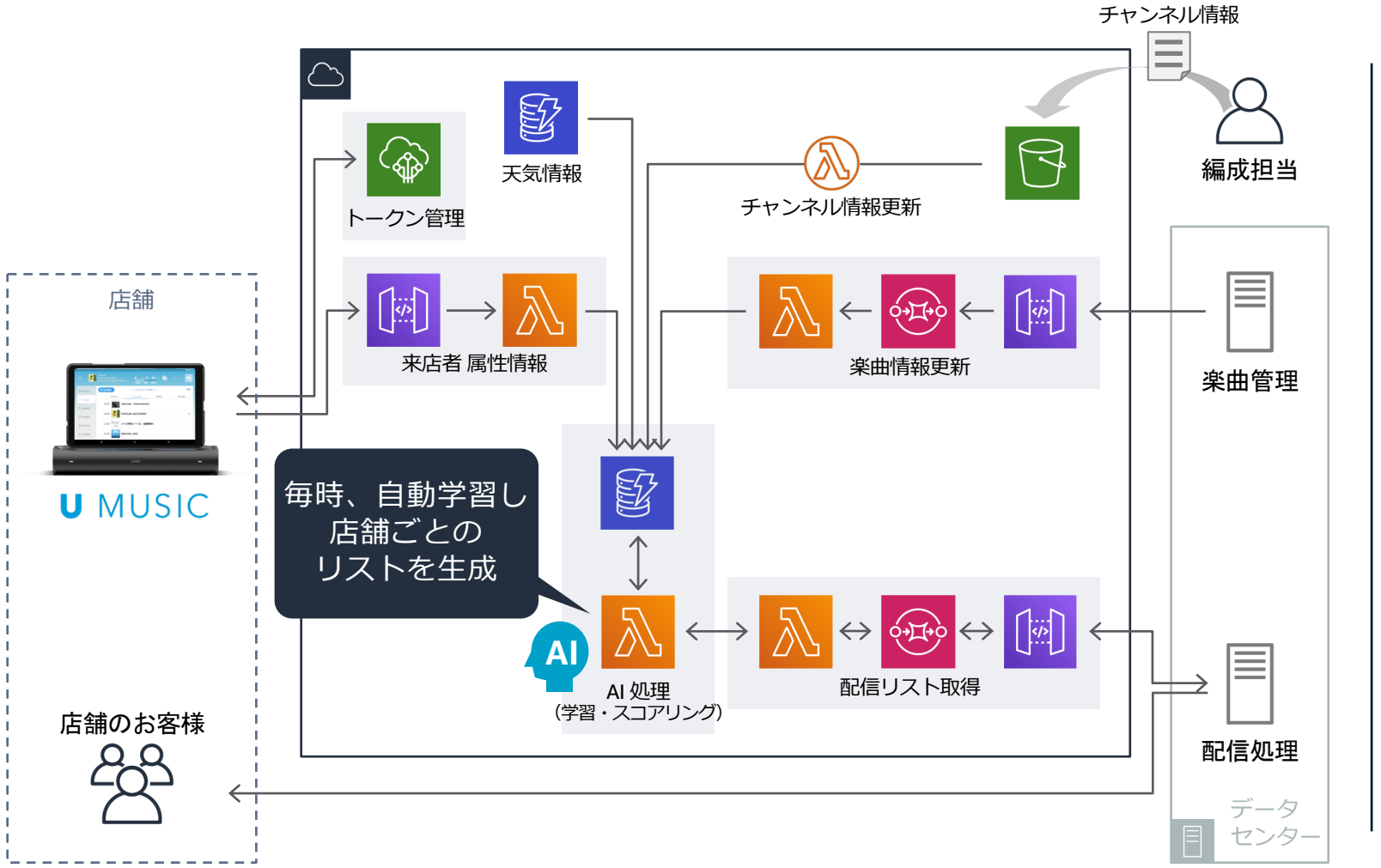
スケーラビリティ

お客様の利用が増加しても対応可能
（容量見積もりの時間短縮に貢献）

マネージド
業務注力

顧客体験の向上・改善に集中
アプリの運用・維持をクラウドに移譲

USEN様
「U MUSIC」AI によるチャンネル選曲とプレイリスト配信サービス



店舗DX :お店の雰囲気、時間帯、
天候に合わせた最適な BGM を提供

スケーラビリティ

数十万店舗のプレイリストの作成を
並列化し、1時間ごとに **5分** で完了

高生産性

開発期間を **数ヶ月単位** で短縮
新サービスの迅速な展開を加速化

コスト最適化

初期投資を **約 6000万円** 削減
運用工数低減 → 中長期の**維持費も削減**

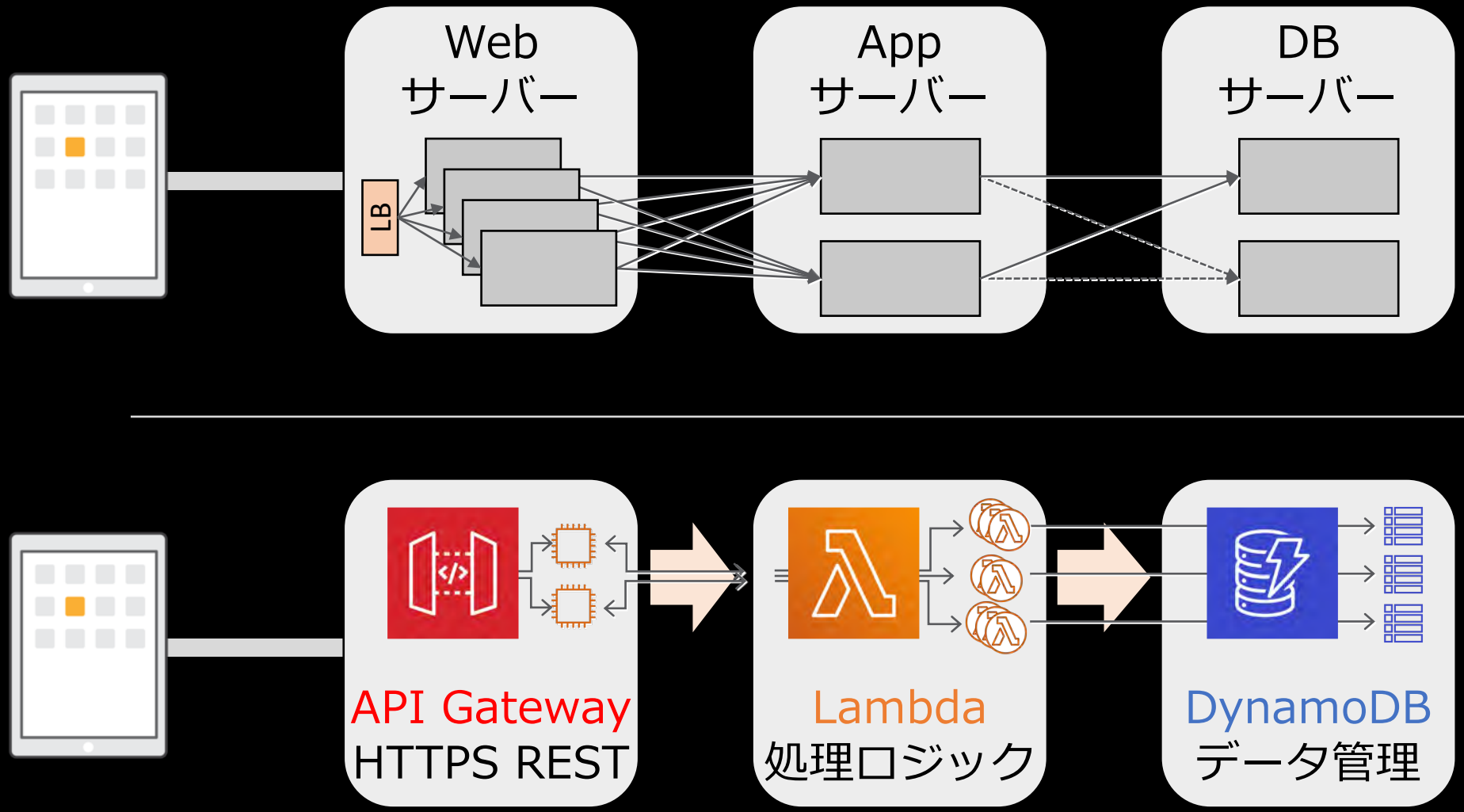
サーバーレス の効能

作業量の削減 ＋ 時間の短縮 ＋ 利用費の適正化

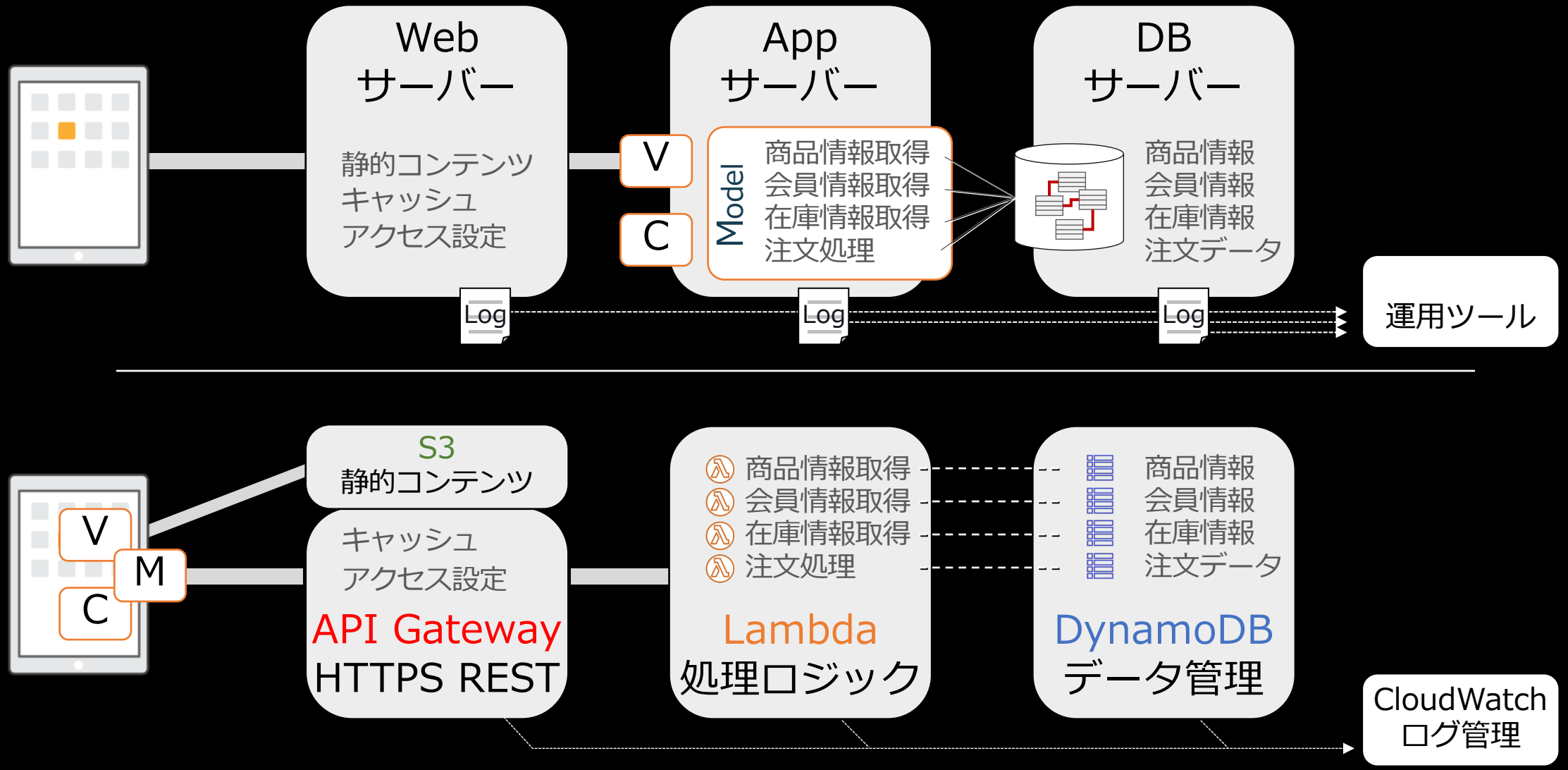
サーバーレスによるお客様の効果例

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

これまでの方式との対比: 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

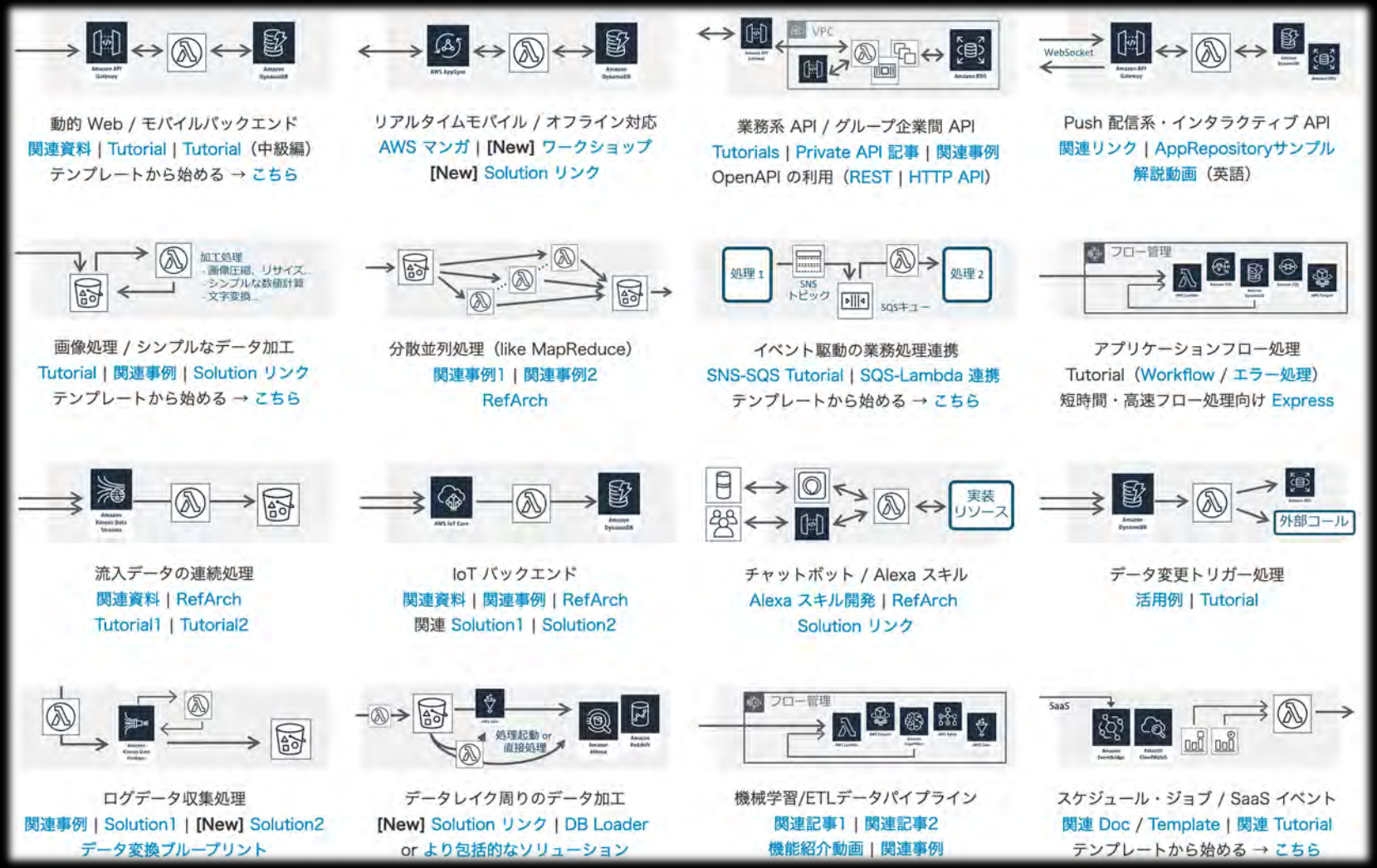
RDS

- テーブル
- データ

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

実戦でよく使われるユースケースパターン



サーバーレス パターン

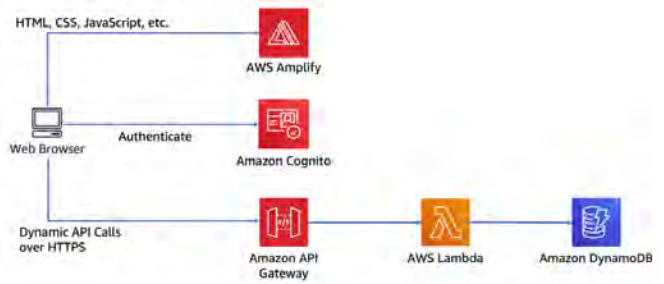
https://aws.amazon.com/jp/serverless/patterns/serverless-pattern/

サーバーレスの始め方（1/2）

2021.12 Edition

1 最初のトライ: サーバーの準備も実行環境構築も不要、いきなりアプリ開発を体験!

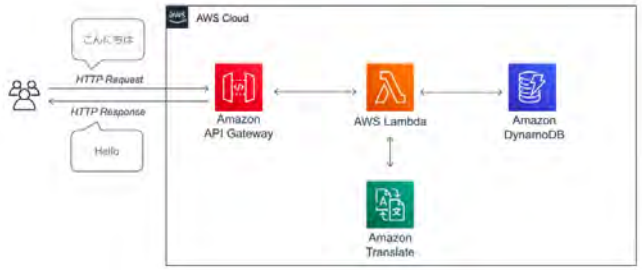
最初のサーバーレスWebアプリ：手順に沿えば、多くのサーバーレスサービスに触れながら、Webアプリが作れます。



go.aws/2Sh6I5M

「動的 Web / モバイルバックエンド」パターン

5-10分 x 11本のハンズオンで、サーバーレスな機能 API を作りながら、少しずつサービス自体の理解を深めていけます。



go.aws/2UhYdKw

「機能API」パターン

開発環境を準備しよう

2 まずは手軽に CI/CD 環境を試す

コード変更を確定させたらビルド・デプロイまで自動でフローを走らせる、そんな CI/CD 環境を構築して、そこで事前定義済みの典型的なサーバーレスアプリケーションのテンプレートから開発できるようにする機能が AWS Lambda には用意されています。まずは

3 開発環境、CI/CD をきちんと準備する

普段お使いの開発環境を使ってサーバーレス開発を進めることができます。

» 概要解説はこちら go.aws/2RUngS5

開発環境+ツールキットの設定

NEXT STEP

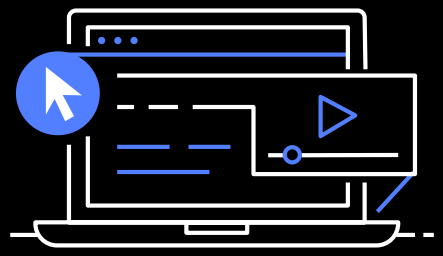
よりサーバーレスを理解するために

- Step-by-Step のハンズオンからスタート（Hands-on for Beginnerシリーズ）
- 全体感を理解する 6 Step
- できそうなところから手を付ける、もアリです

https://amzn.to/2ZGL3ZS

https://aws.amazon.com/jp/serverless/patterns/redirect-serverless-steps

AWS デジタルトレーニング



実力、自信、信頼性を高め、業界で認められた資格で差をつけよう

デジタル学習

- スキルビルダー – AWS のエキスパートが開発した数百のデジタルトレーニングを自分のスケジュールで学習できます
- Cloud Quest - AWS Cloud Quest は、実践的なクラウド経験を積み、AWSクラウドのスキルを身につけることができる、初めてで唯一のロールプレイングゲームです

認定試験準備ためのリソース

- Cloud Practitioner - AWS Certified Cloud Practitioner 取得に役立つリソースをご紹介します
- Developer – Associate – AWS Certified Developer – Associate 取得に役立つリソースをご紹介します

aws

aws.amazon.com/jp/training

AWS Builders Online Series に
ご参加いただきありがとうございます

楽しんでいただけましたか? ぜひアンケートにご協力ください。
本日のイベントに関するご意見/ご感想や今後のイベントについての
ご希望や改善のご提案などがございましたら、ぜひお聞かせください。


aws-apj-marketing@amazon.com


twitter.com/awscloud_jp


facebook.com/600986860012140


https://www.youtube.com/user/AmazonWebServicesJP


https://www.linkedin.com/showcase/aws-careers/


twitch.tv/aws

Thank you!