【大阪開催・AWS共催セミナー】
システム開発でのクラウド利用！最新動向と成功のポイント！！

AWSを活用した業務システム開発の実際とこれから

株式会社鈴木商店
塩飽　展弘

アジェンダ

- 鈴木商店のご紹介
- AWSを活用した業務システム開発の実際
  - フェーズ1：オンデマンドなリソース/既存環境との分離
  - フェーズ2：ネットワーク接続性/運用アウトソース
  - フェーズ3：クラウドならではサービス
- AWSを活用した業務システム開発のこれから

本日のポイント

- **業務システムは、従来の多層アプリケーションが多かった**
  - オンプレミスの延長線上にあるシステムの需要
  - AWSを活用することで価格低減やできることが拡大する
  - AWSはシステムの運用アウトソースを柔軟にする
- **今後はクラウドネイティブな要素を含んだ業務システムが増加する**
  - クラウドネイティブな考えを早い段階で取り入れることが競争力を上げる

鈴木商店のご紹介

挑戦する お客様の力に

今まで3日かかっていた処理が 1秒で出来る。ボタンひとつで数千万人に情報が伝わる。
知りたい情報が瞬時に見つかる。

システムには破壊的な力があります。
私はそんなシステムのちからに取り憑かれました。 **大好き♡**

しかし、システムで出来ることは情報を伝えることだけです。
美味しいものを食べたり、壮大な自然に感動したり、
家族と大切な時間を過ごしたり、病で苦しむ人を少しでも楽にしたり。
残念ながら、そのような **リアルなことはシステムには出来ません。**

でも、リアルなことに **挑戦している人たちの力にはなれる。**
しかも私達が大好きなシステムで **その力を何万倍** にもできる。
そんな思いでシステム開発会社を立ち上げました。

挑戦する人たちの力になるためには、
システムをもっと身近で手軽なものにする必要があります。

これまでのシステム開発の常識を捨て、**システム開発と正面から向き合い** 、
低価格で高品質なシステムの実現 に私達は挑戦してまいります。

代表取締役 鈴木史郎

鈴木商店のご紹介

とことん使いやすさにこだわった
システム開発を行う「ものづくり集団」です。

クラウドを活用し、フルスクラッチによる業務システムを中心に開発を実施しています。

90%以上の案件で活用

aws partner network

Standard Consulting Partner

salesforce registered consulting partner

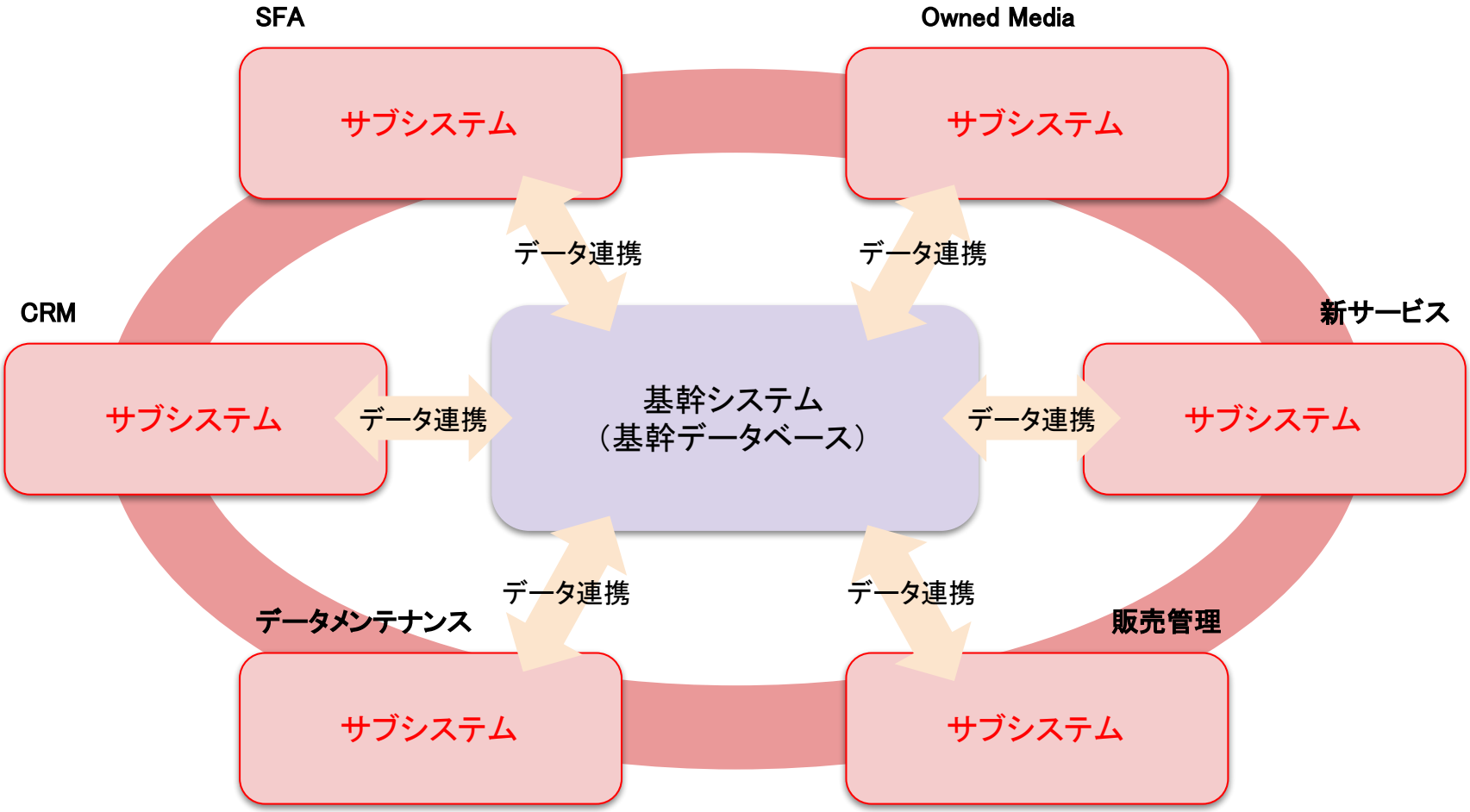
Google Cloud Partner

twilio

鈴木商店のご紹介

AWSを活用した業務システム開発の実際

鈴木商店の主なシステム開発領域



※この限りではありません

鈴木商店の主なシステム開発領域

サブシステム

- **業務主管部門主導のシステム開発**
  - 動きやすさ
  - 基幹システム（基幹DB）との分離
  - スピード
  - 柔軟性
  - 運用管理のアウトソース

AWSのメリットを享受しやすい

AWSを選定される理由（業務システムの場合）

なんでもいい！

新規システムなので
クラウドでやりたい
（やってみたい）

クラウドなら
実現できるのでは？

既存のオンプレ環境に
手を入れてほしくない

当然クラウドでしょう

オンデマンドなリソース /
既存環境との分離

ネットワーク接続性 /
運用アウトソース

クラウドならではサービス

オンプレの延長線

クラウドネイティブ


EC2

RDS

Direct
Connect

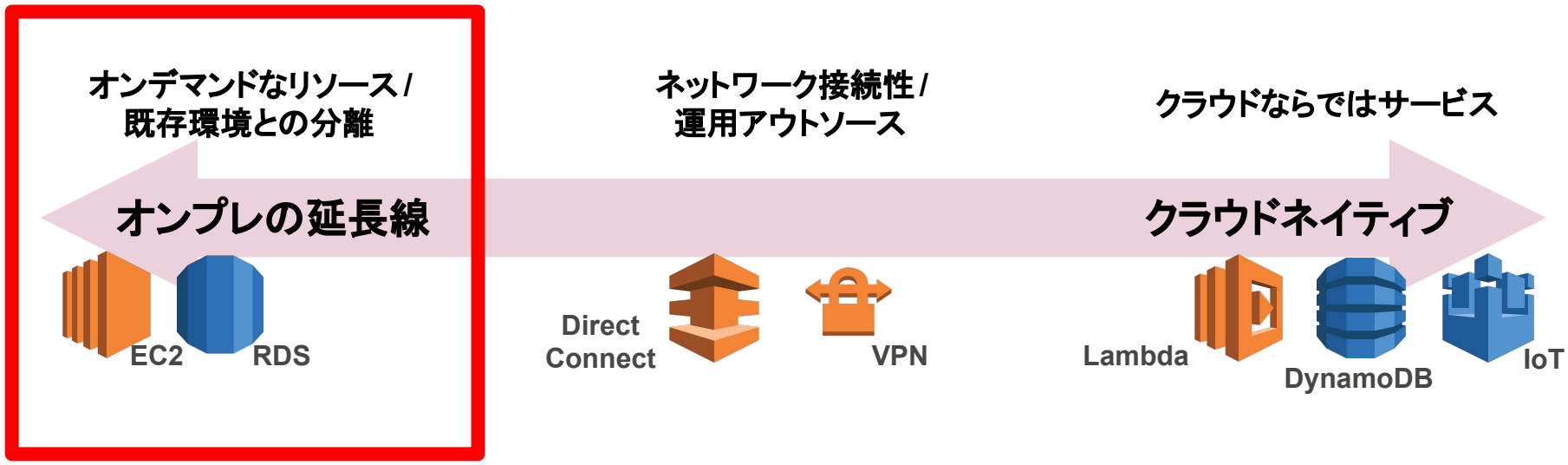

VPN

Lambda

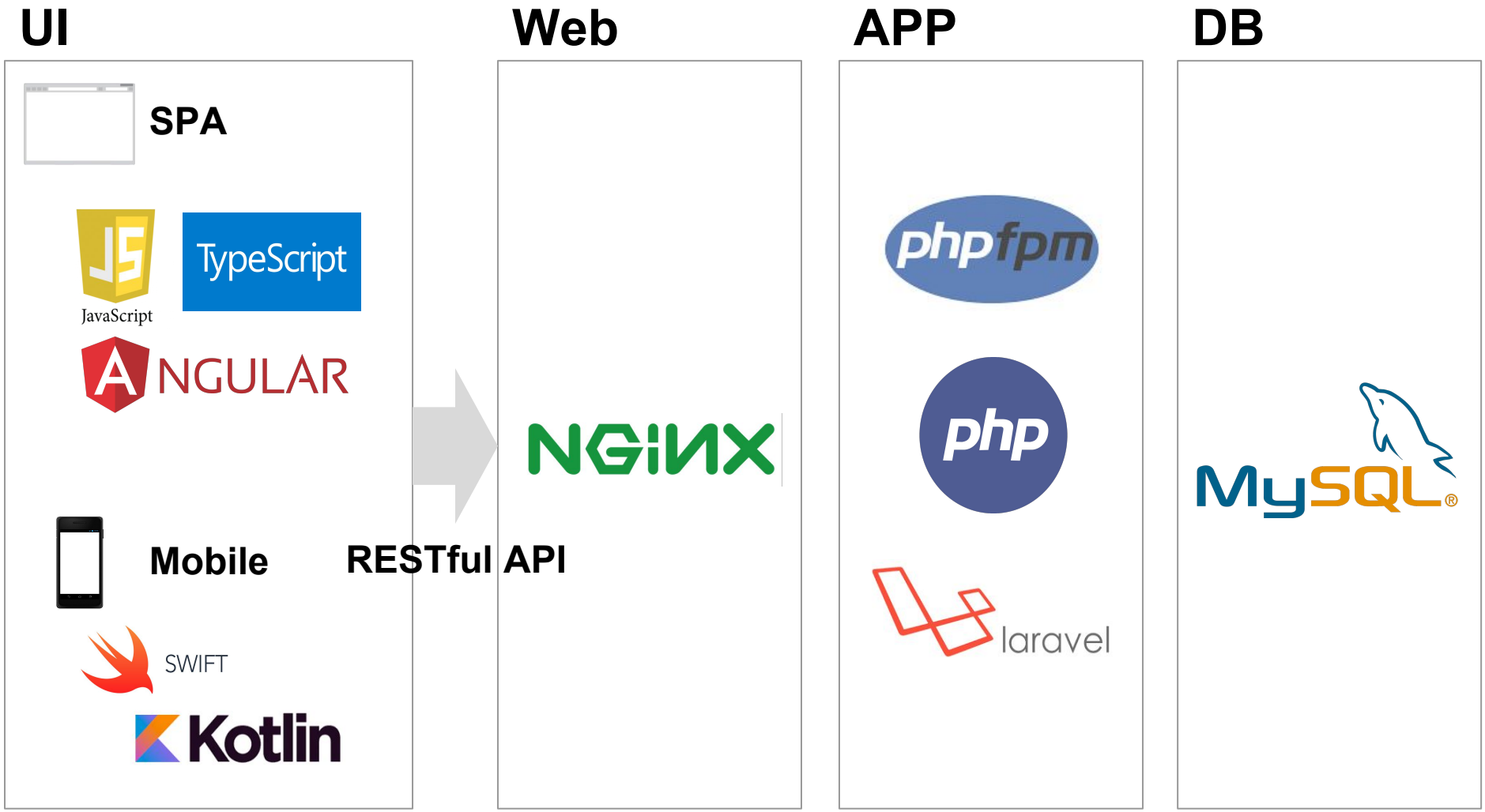

DynamoDB


IoT

フェーズ1



Webでの業務システム開発

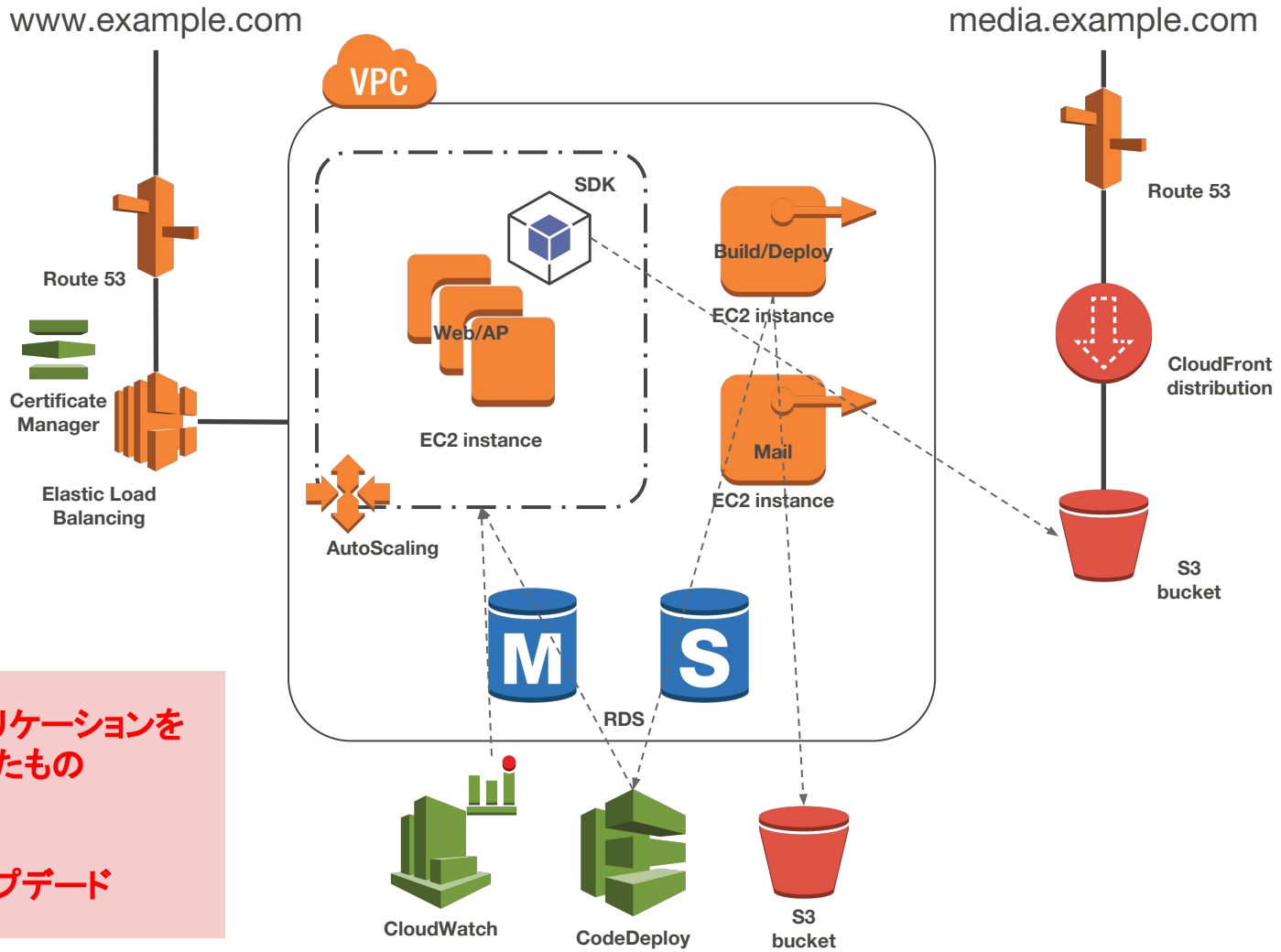


Webでの業務システム開発

- **データ構造が複雑**
  - リレーショナルデータベースで複雑なJOINが必要
  - リレーショナルデータベースの扱いに適しているWebアプリケーションフレームワークを使用
  - 基幹システム（DB）との連携

従前からの多層アプリケーションが開発のベース
多層アプリケーションに適したクラウドサービスの活用

業務システムでよくあるアーキテクチャ



通常の多層アプリケーションを
AWS上に実装したもの
＋CDN
＋AutoScaling
＋ローリングアップデード

オンプレミスと同じなのか？
何かメリットはあるのか？

構築が早い

サービスを
止めない

キャパシティ

違います。

柔軟性
拡張性

変動費

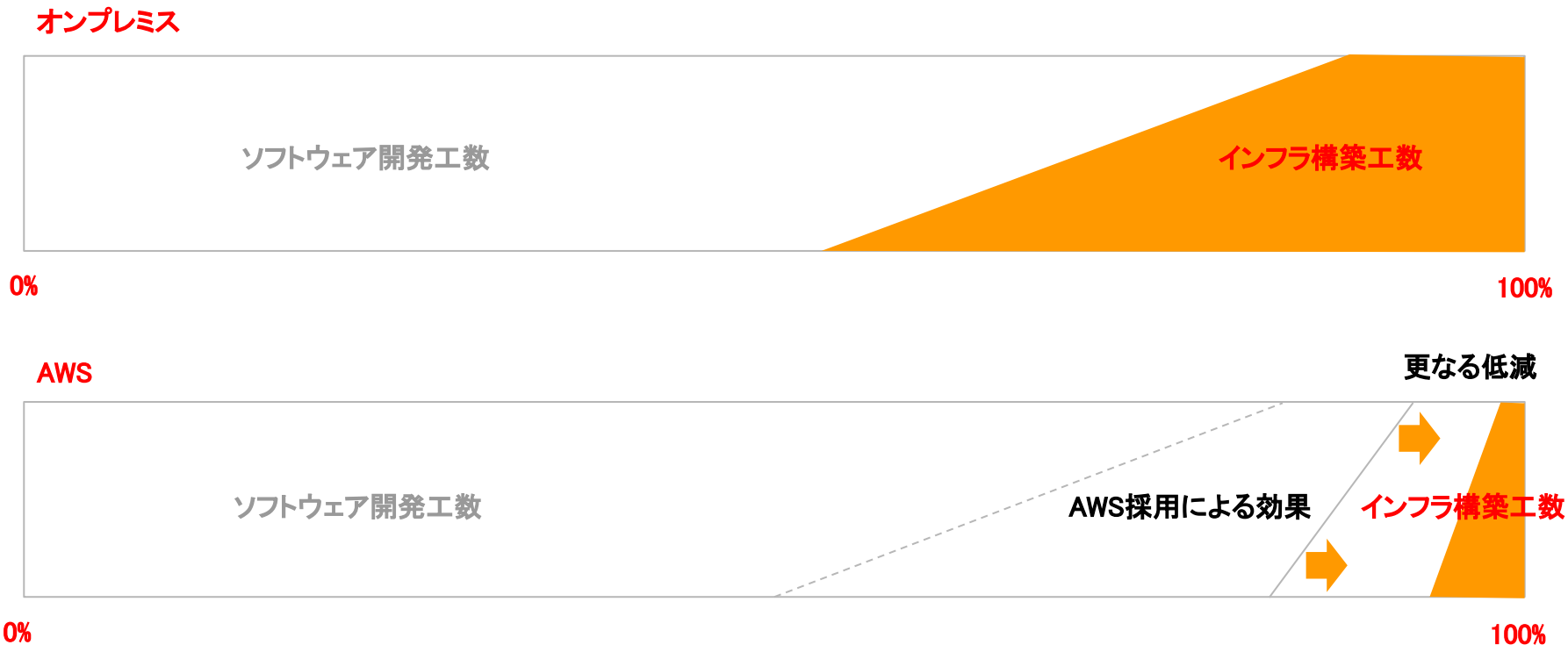
物理HWなし

他、多数

システム開発におけるインフラ構築工数の割合イメージ

構築が早い

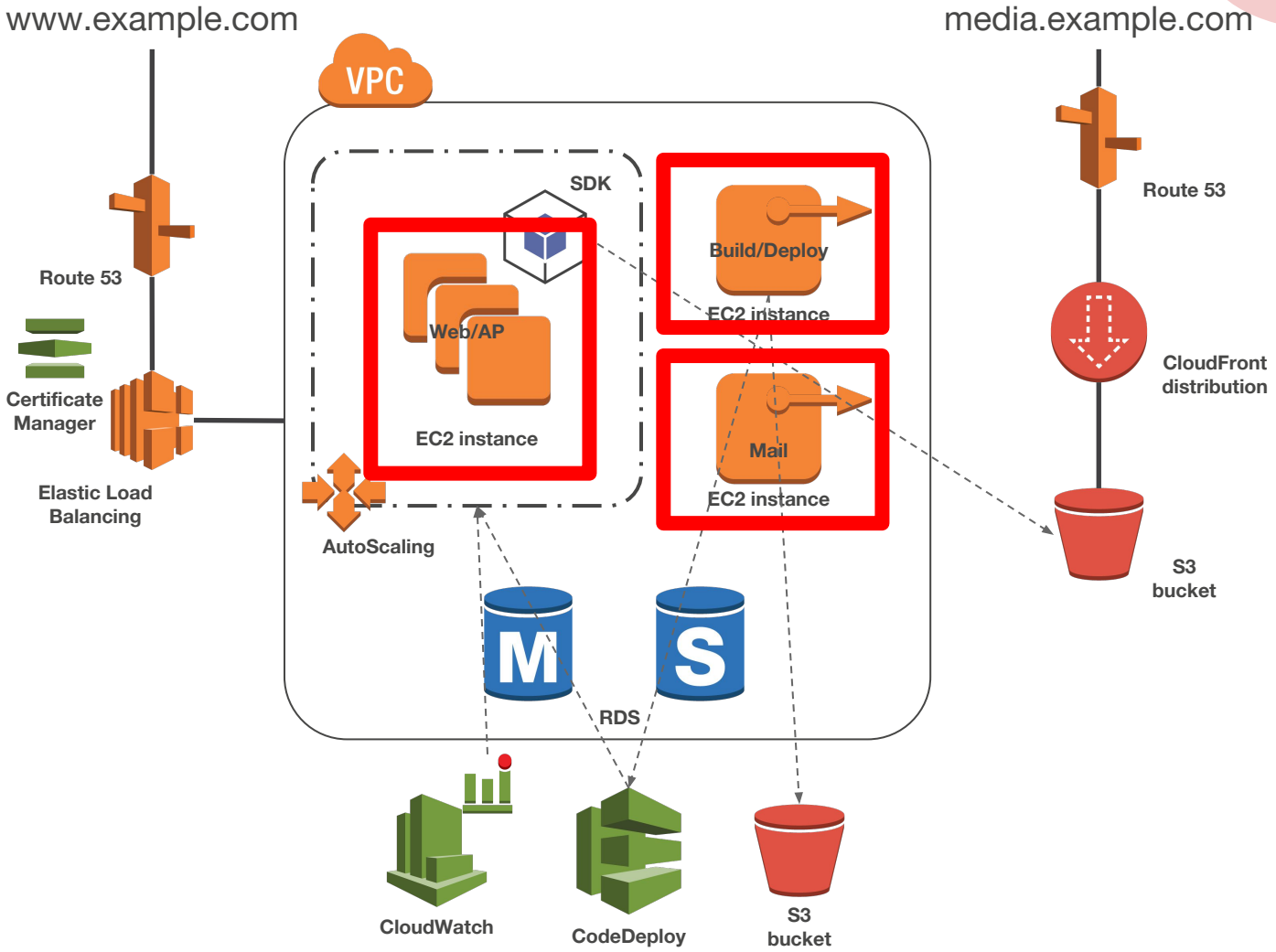
新規で業務システムを構築する場合の工数割合（弊社が構築するシステムの場合）



※弊社経験則によるものであり、システムにより大きく異なる場合もあり、内容を保証できるものではございません。

更なる工数低減

構築が早い

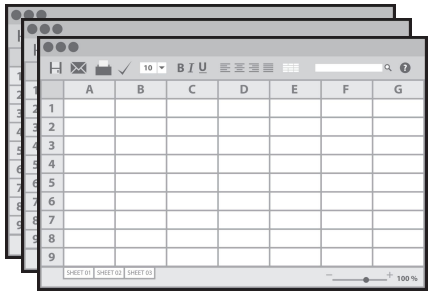


更なる工数低減

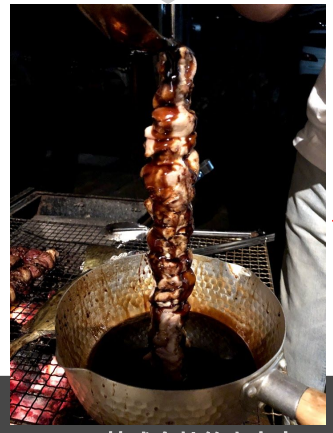
- **OS関連設定**
  - タイムゾーン
  - ロケール
  - ユーザ
  - ディレクトリ
  - などなどなどなど
- **ソフトウェアのインストール**
  - nginx
  - php
  - php-fpm
  - postfix
  - opendkim
  - td-agent
  - codedeploy-agent
  - DBクライアント
  - nodejs
  - などなどなどなど
- **ソフトウェア関連の設定**
  - ・・・・

構築が早い

パラメータシート



構築手順書

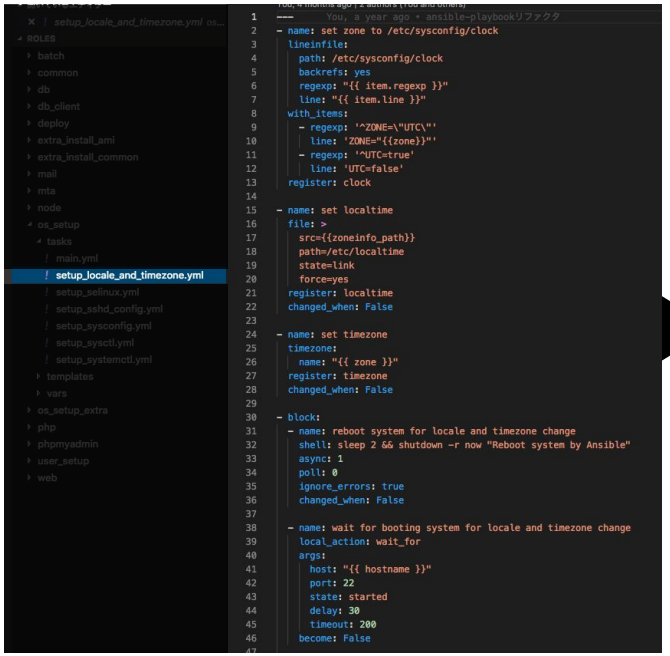


秘伝のタレ化

更なる工数低減

構築が早い

playbook(yaml)




```yaml
---  You, a year ago • ansible-playbookリファクタ
- name: set zone to /etc/sysconfig/clock
  lineinfile:
    path: /etc/sysconfig/clock
    backrefs: yes
    regexp: "{{ item.regexp }}"
    line: "{{ item.line }}"
  with_items:
    - regexp: '^ZONE=\"UTC\"'
      line: 'ZONE="{{zone}}"'
    - regexp: '^UTC=true'
      line: 'UTC=false'
  register: clock

- name: set localtime
  file: >
    src={{zoneinfo_path}}
    path=/etc/localtime
    state=link
    force=yes
  register: localtime
  changed_when: False

- name: set timezone
  timezone:
    name: "{{ zone }}"
  register: timezone
  changed_when: False

- block:
  - name: reboot system for locale and timezone change
    shell: sleep 2 && shutdown -r now "Reboot system by Ansible"
    async: 1
    poll: 0
    ignore_errors: true
    changed_when: False

  - name: wait for booting system for locale and timezone change
    local_action: wait_for
    args:
      host: "{{ hostname }}"
      port: 22
      state: started
      delay: 30
      timeout: 200
    become: False
```

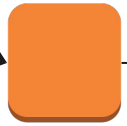
開発環境

Amazon Linux

コマンド1つ

ステージング・本番環境

コマンド1つ



EC2 Instance

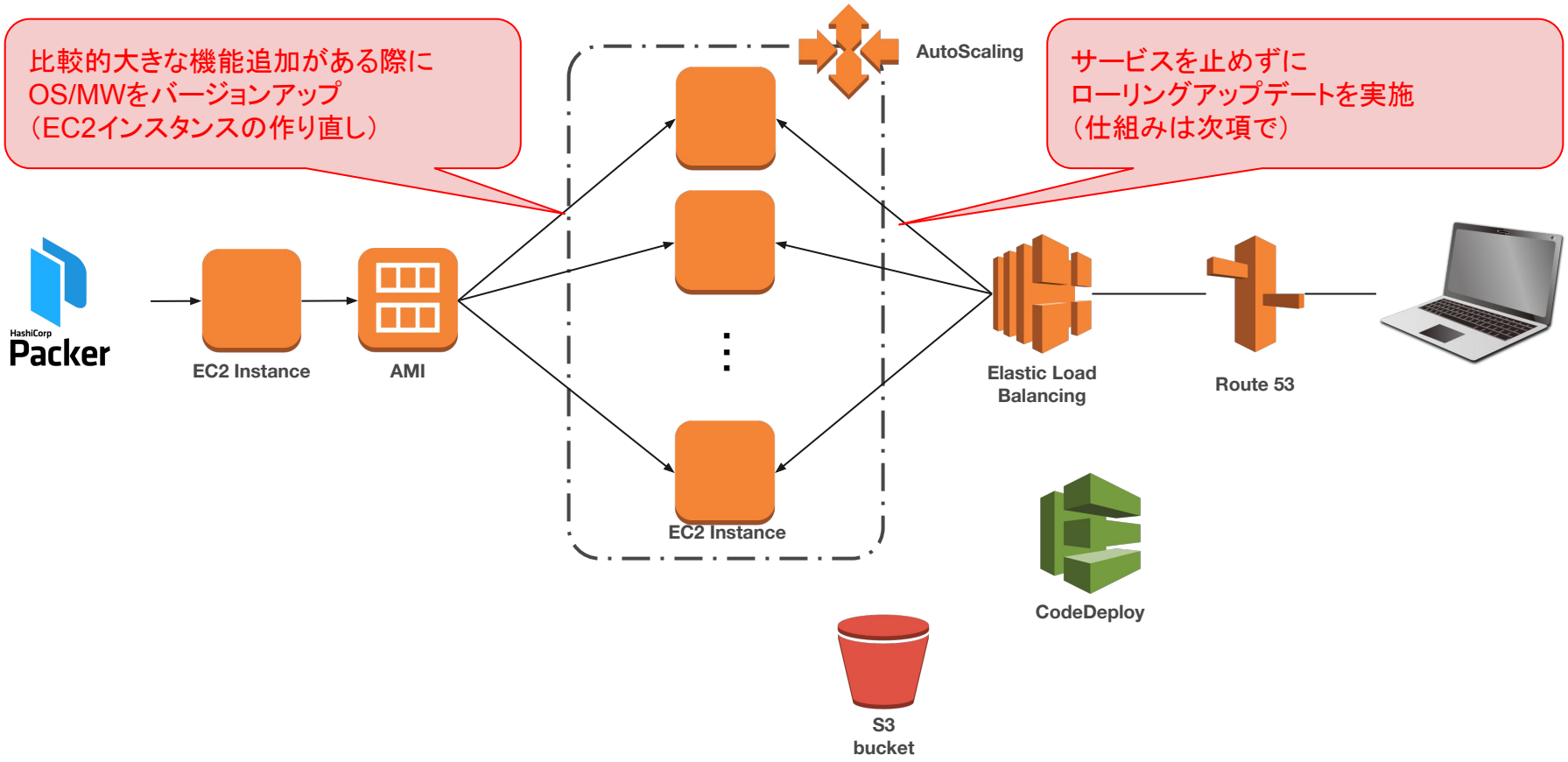
AMI

- 早い
- 秘伝のタレにならない
  - 作り直せる（これも大きい）

作りなせるサーバ＆ローリングアップデート

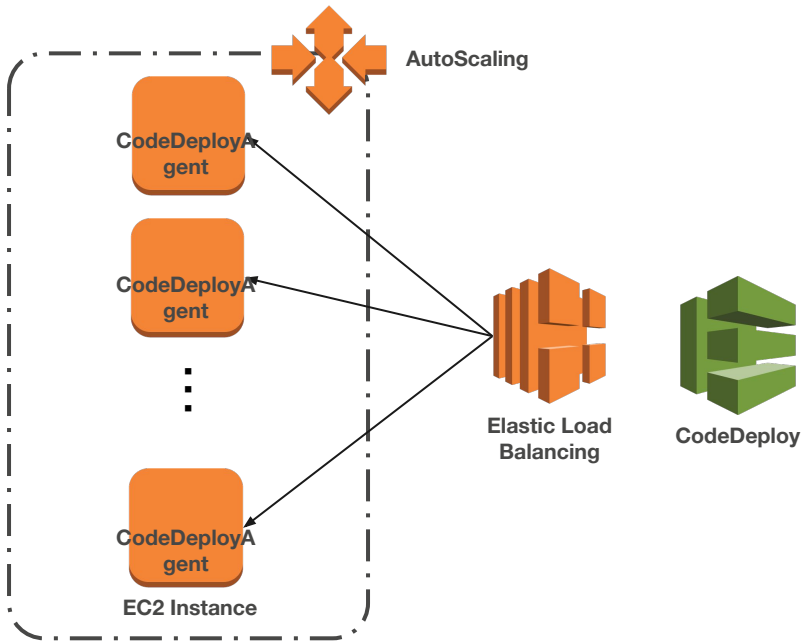
サービスを
止めない

スモールスタートで、徐々に機能追加をしていくような開発をすると、
サービスを止めずに、サーバも入れ替えることで定期的なアップデートが可能です。
（CIまではいらない、というような受託開発で有効）



CodeDeployを用いたローリングアップデート

サービスを
止めない



アプリケーションの
最新リビジョンを配置

デプロイの
イベントフックリスト

ApplicationStop
- ELBからEC2インスタンスをデタッチ
- Webサーバ（nginx）等を停止

DownloadBundle
- S3から最新リビジョンを取得

BeforeInstall
- 現在のリビジョンを削除

Install
- 最新リビジョンを展開

AfterInstall

ApplicationStart
- Webサーバ等を再開
- ELBにEC2インスタンスをアタッチ
- ステータス確認

ValidateService

インプレース
デプロイ

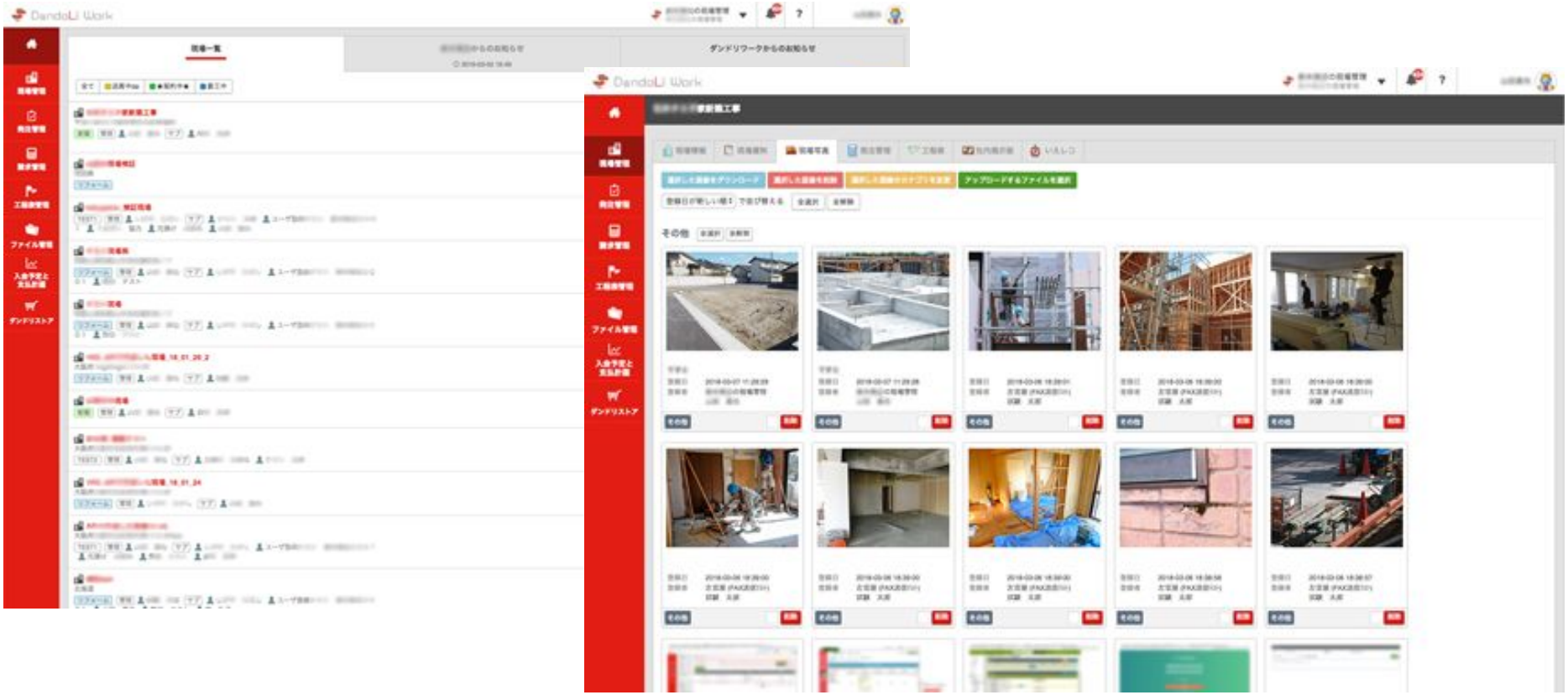
&

OneAtATime
or
HalfAtATime

事例：建築業界向けの現場管理システム

建築業界向けの現場管理システム

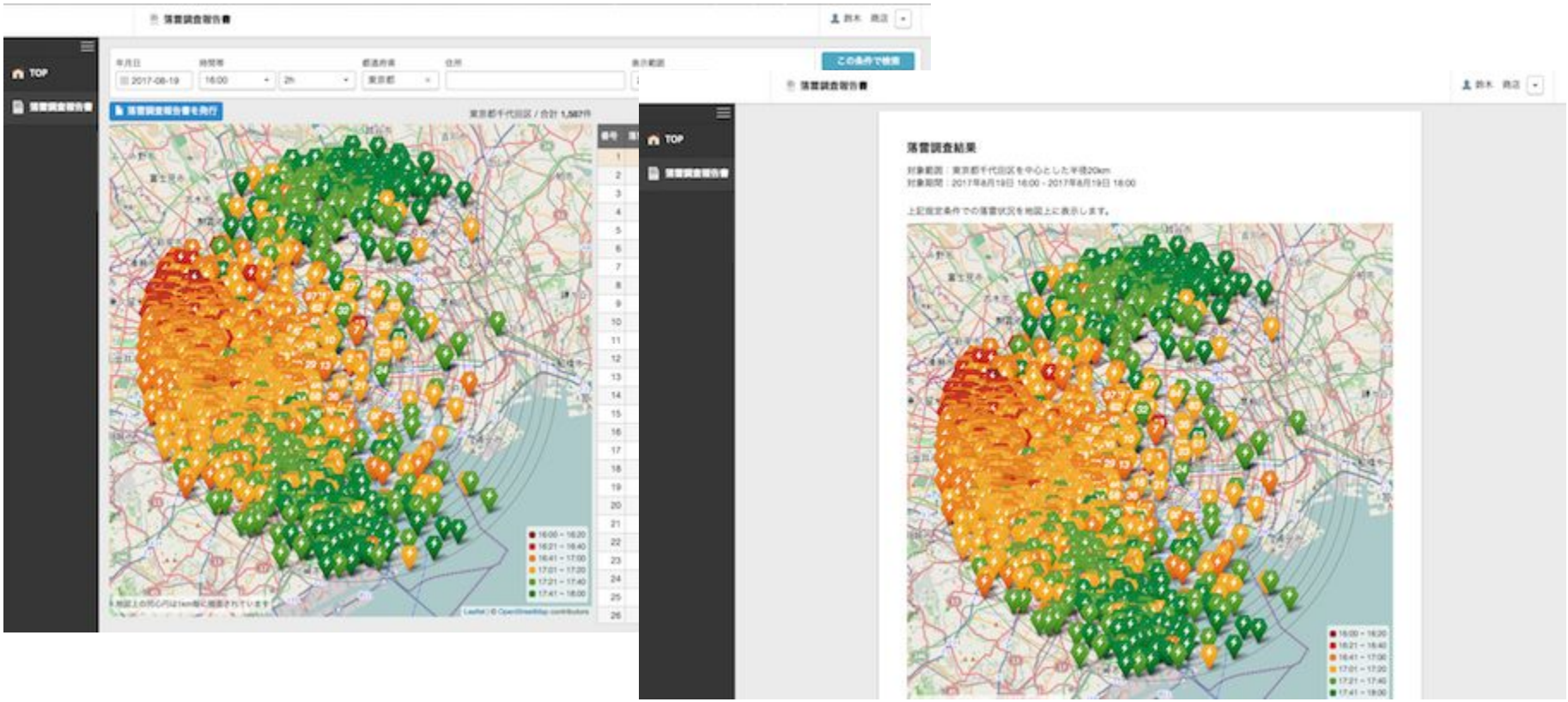
建築現場に関する全ての情報を一元化して業務効率を改善する、クラウド型情報共有システム



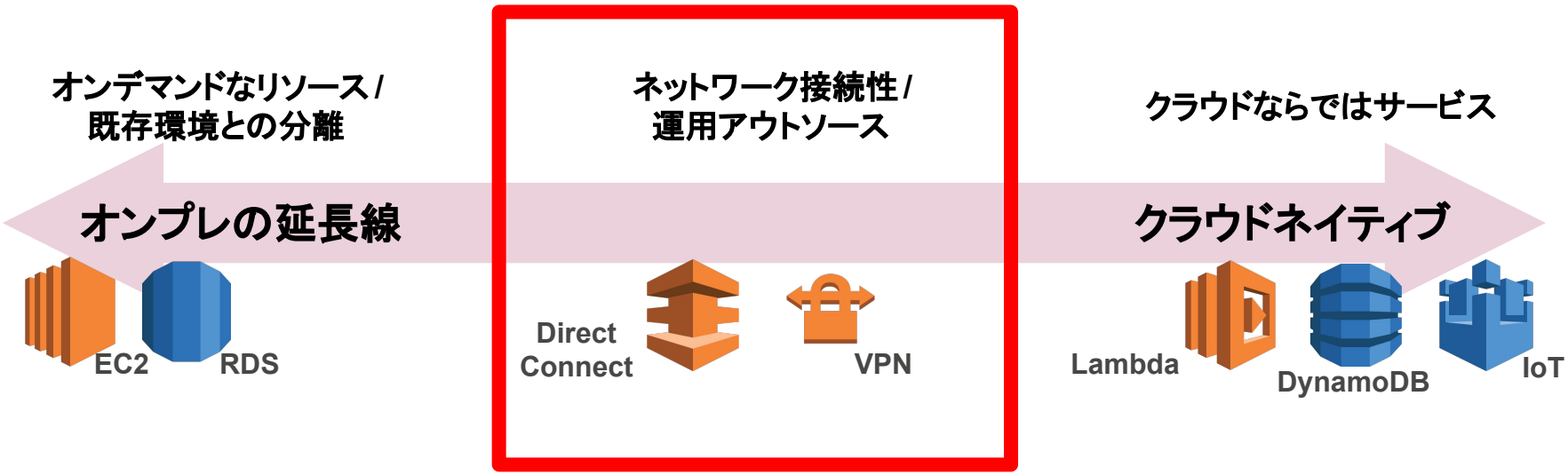
事例：落雷調査報告書発行システム

落雷調査報告書発行システム

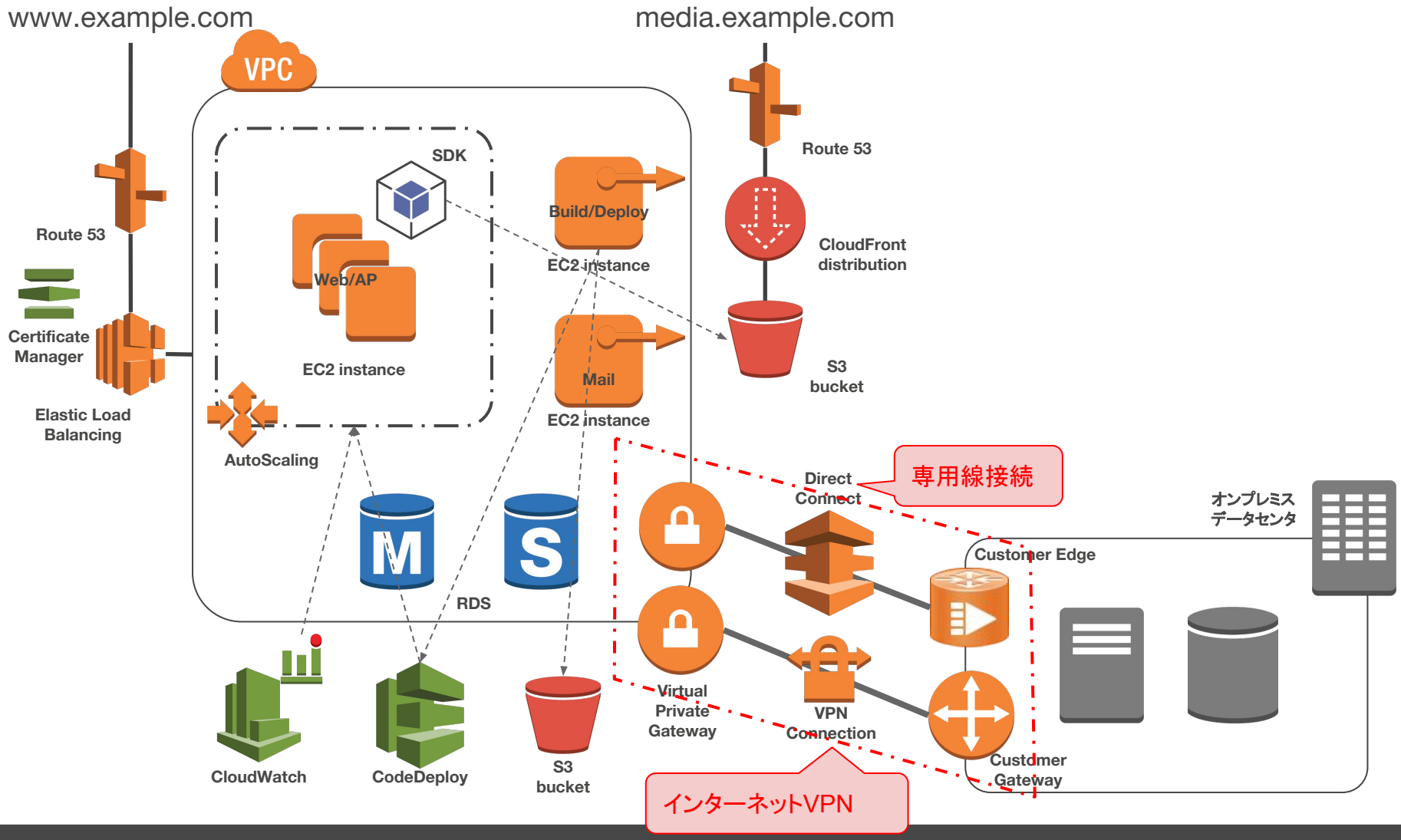
気象庁の雷監視システムが提供する落雷データを活用するシステム



フェーズ2

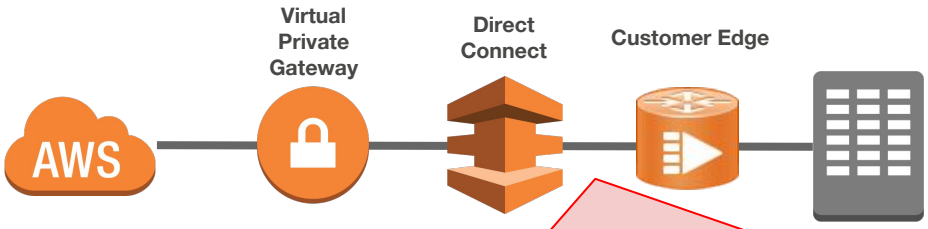


業務システムでよくあるアーキテクチャ（フェーズ2）



AWSとオンプレミスの接続方式

- **専用線での接続**
  - セキュア
  - 帯域確保
  - 低遅延
  - 冗長構成が容易

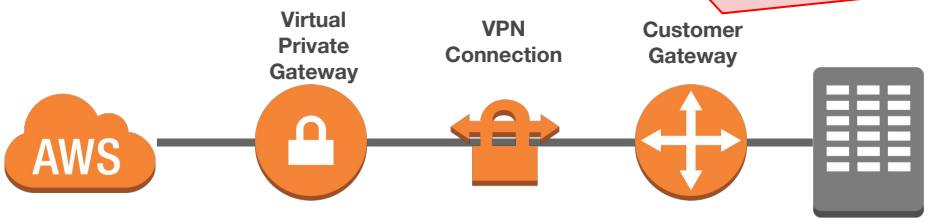


AWS Direct Connectをサポートするパートナー様とともにご提案させて頂きます

https://aws.amazon.com/jp/directconnect/partners/

- **インターネットVPNでの接続**
  - 安価
  - 比較的簡単に接続可能
    - 要固定グローバルIPアドレス

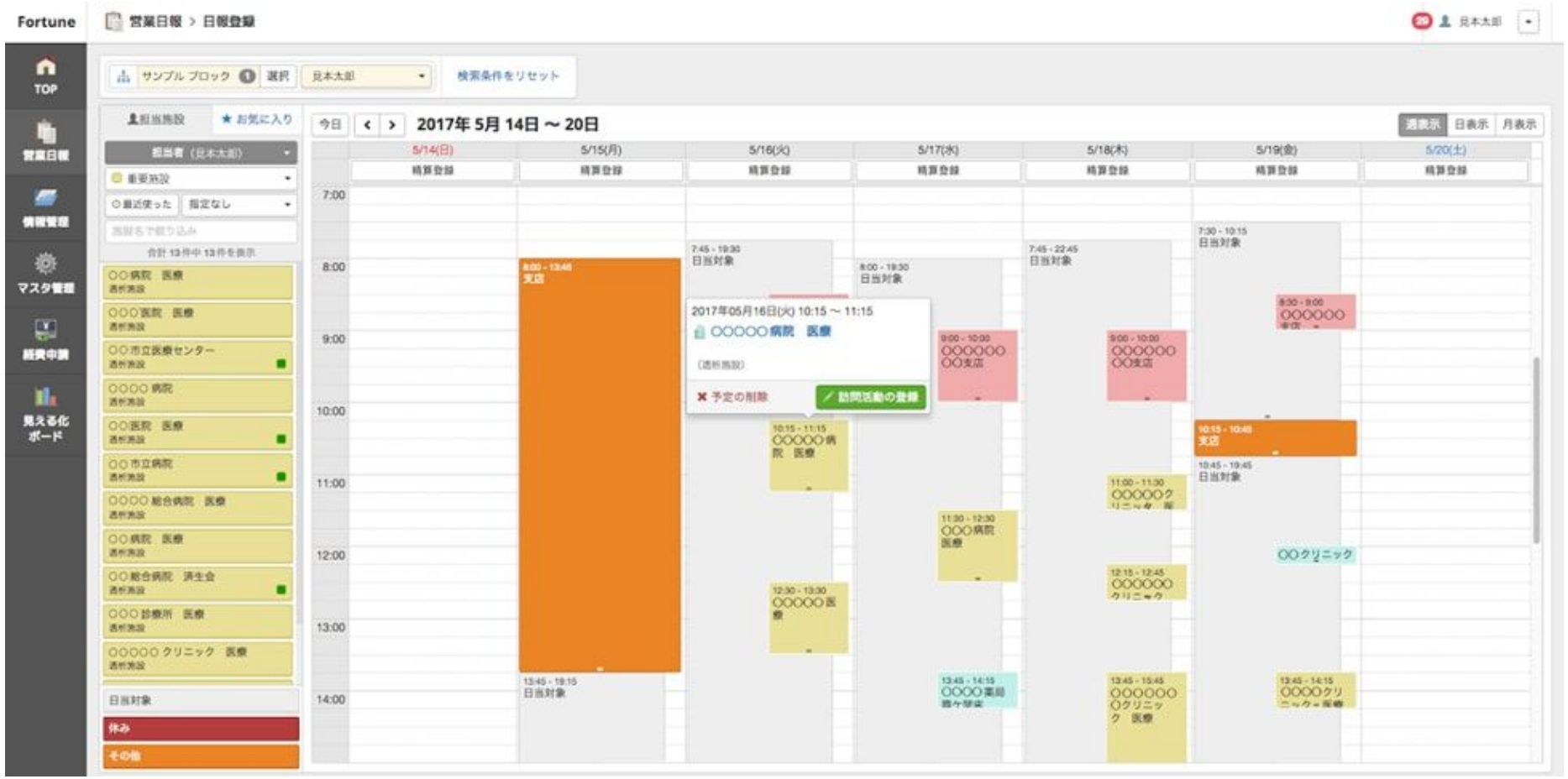
Cisco(ASA, ISRなど）
SonicWall
F5 Networks BIG-IP
Fotigate
Juniper(SSGなど)
YAMAHA RTX
などのサンプルコンフィグを提供致します。



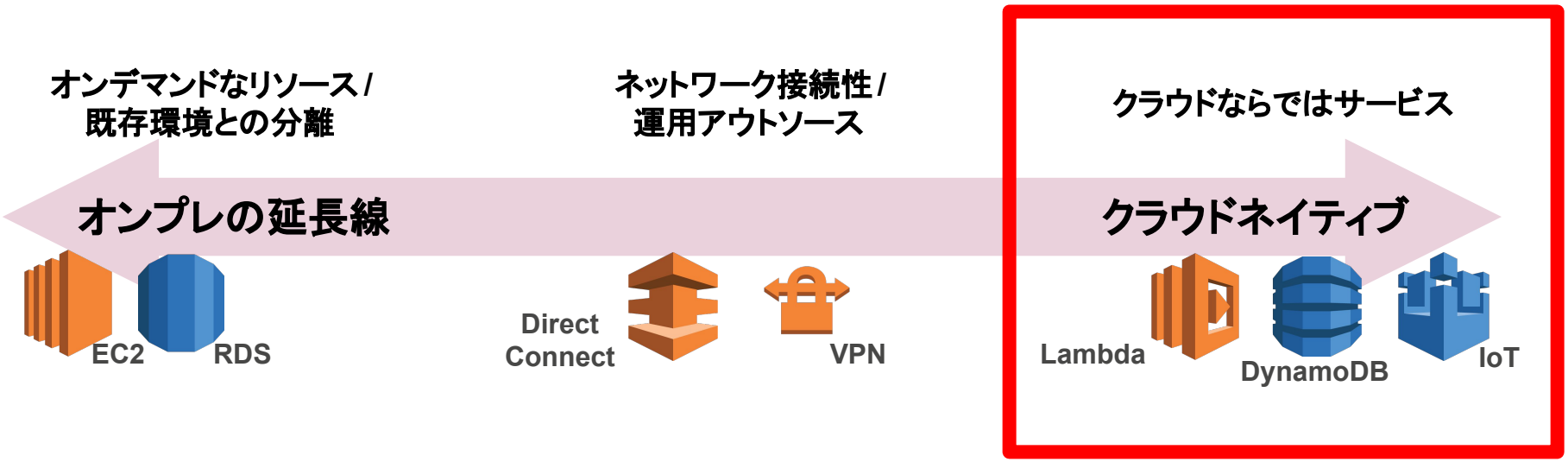
事例：製薬企業様向けSFAシステム

製薬企業様向けSFAシステム

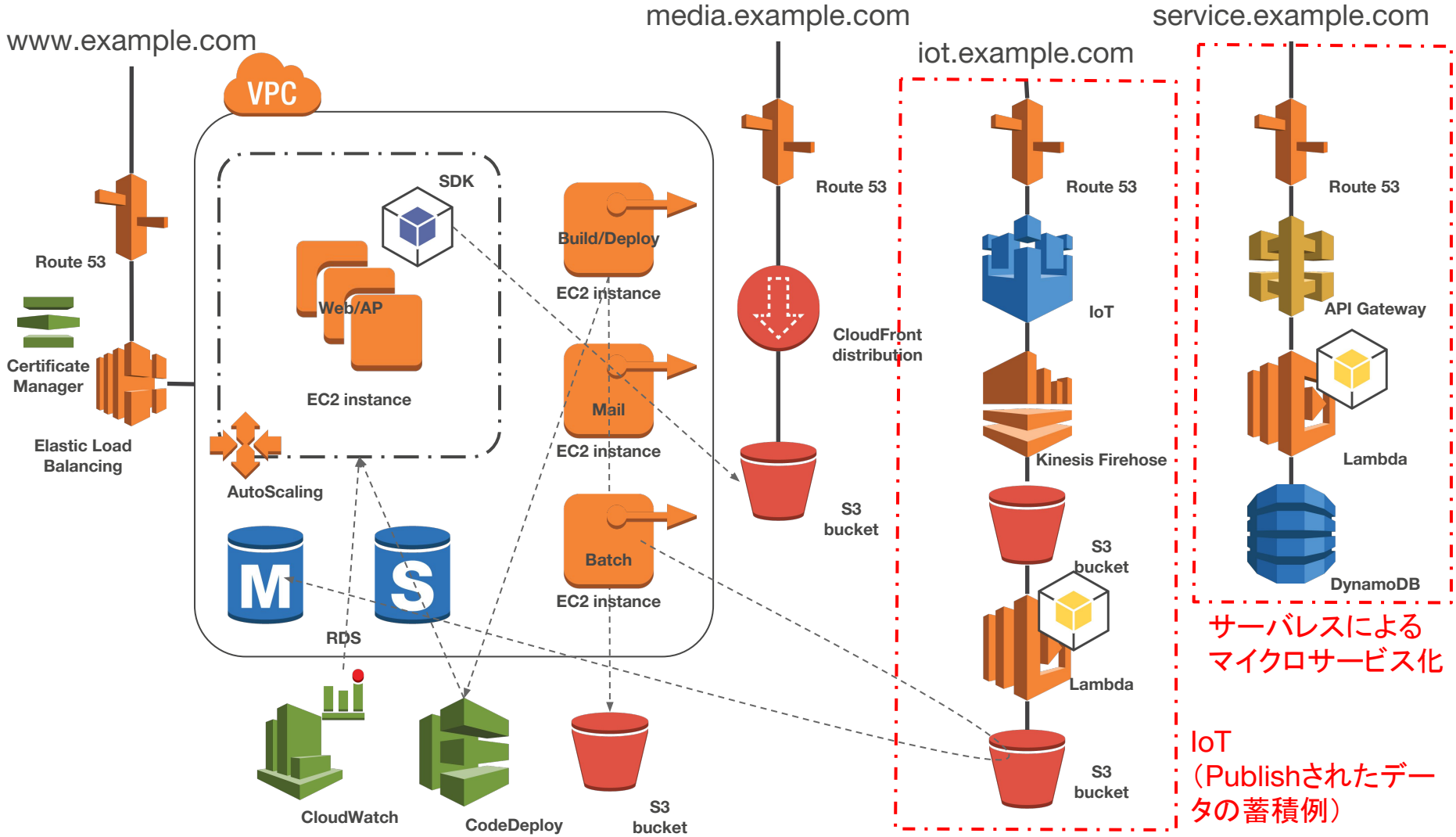
MRが医師等に対して実施するディティール情報を管理共有する、クラウド型システム



フェーズ3

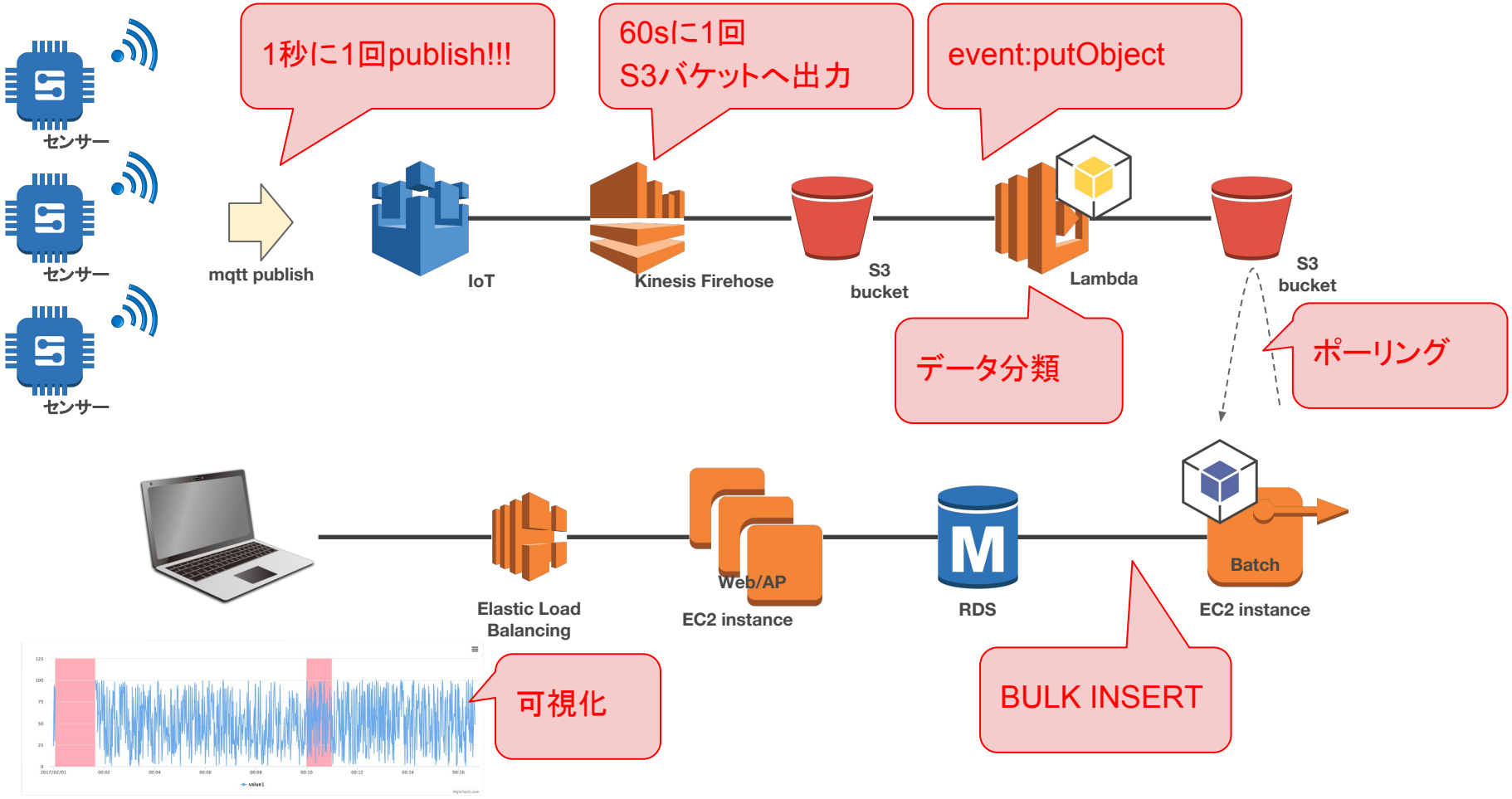


業務システムでよくある？アーキテクチャ（フェーズ3）



IoT（Publishされたデータの蓄積例）

※実験的な用途で使用しています



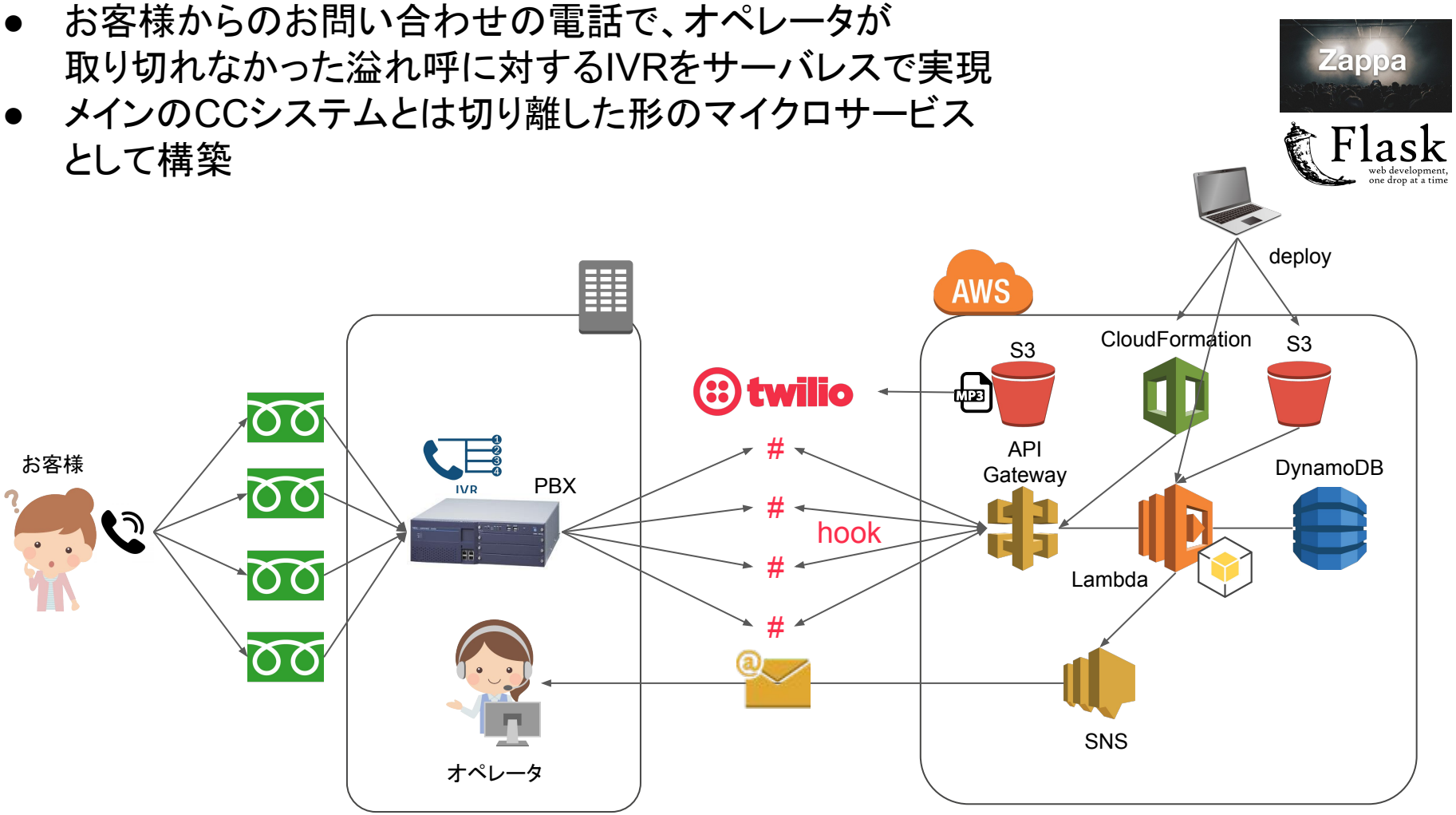
IoT（Publishされたデータの蓄積例）

- 軽量なプロトコル（MQTT）で容易にデータを収集
  - データ収集のみでなく、双方向のやりとりも可能です。
    - デバイスシャドウ

- 大量データの処理、蓄積
  - 今回はRDSやS3を使用しましたが、ご利用方法に応じて
    アーキテクチャを検討させて頂きます。
    - リアルタイムなデータ処理、アクション
    - データのサマライズ、変換処理
    - NoSQLデータベース（DynamoDB）による、
      より大量のデータ蓄積　等

サーバレスによるマイクロサービス化

- お客様からのお問い合わせの電話で、オペレータが取り切れなかった溢れ呼に対するIVRをサーバレスで実現
- メインのCCシステムとは切り離した形のマイクロサービスとして構築



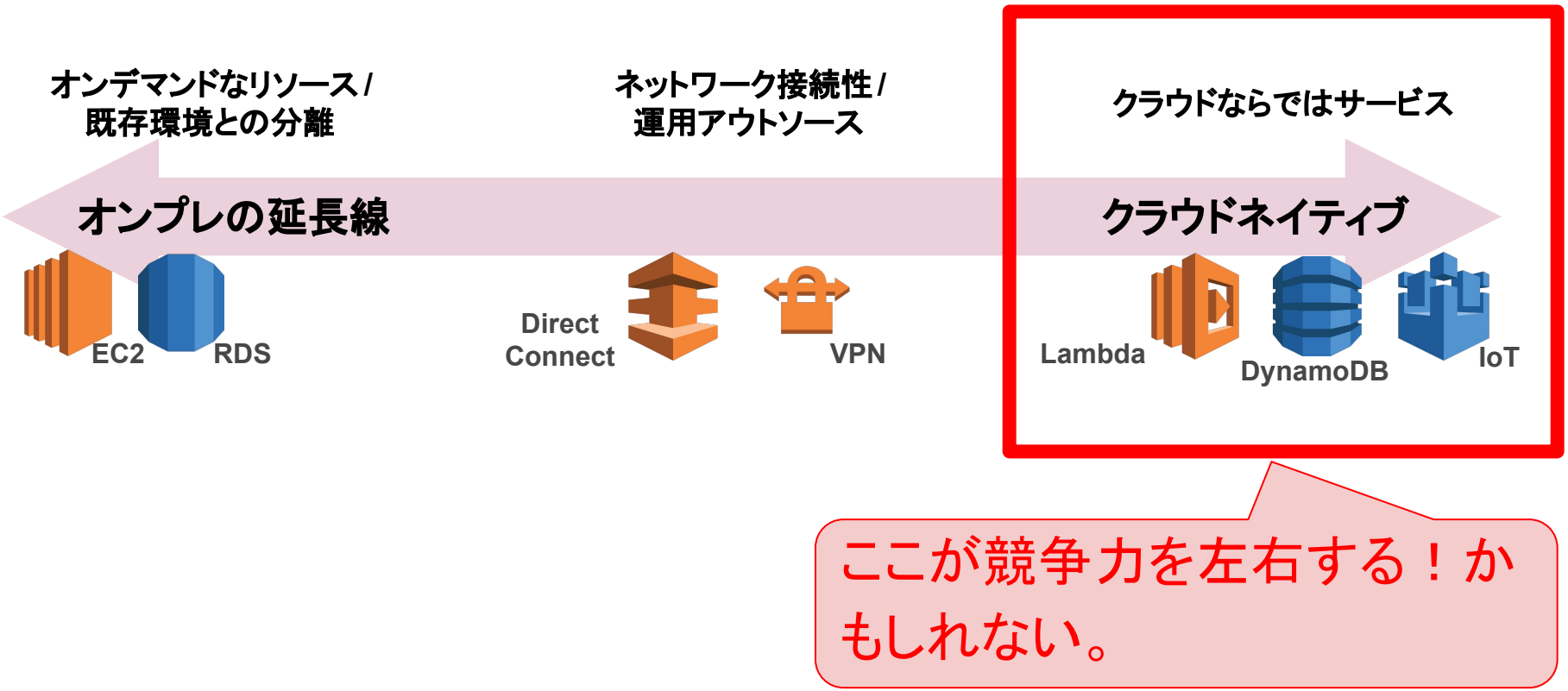
サーバレスによるマイクロサービス化

- スピード vs 将来像
  - （ものによるが）開発スピードは早い
  - 将来的にCTIなどと統合されるか、
    その際のインターフェースとして十分か、
    などを検討し、
    - シンプルで
    - 疎結合でOK（がいい）

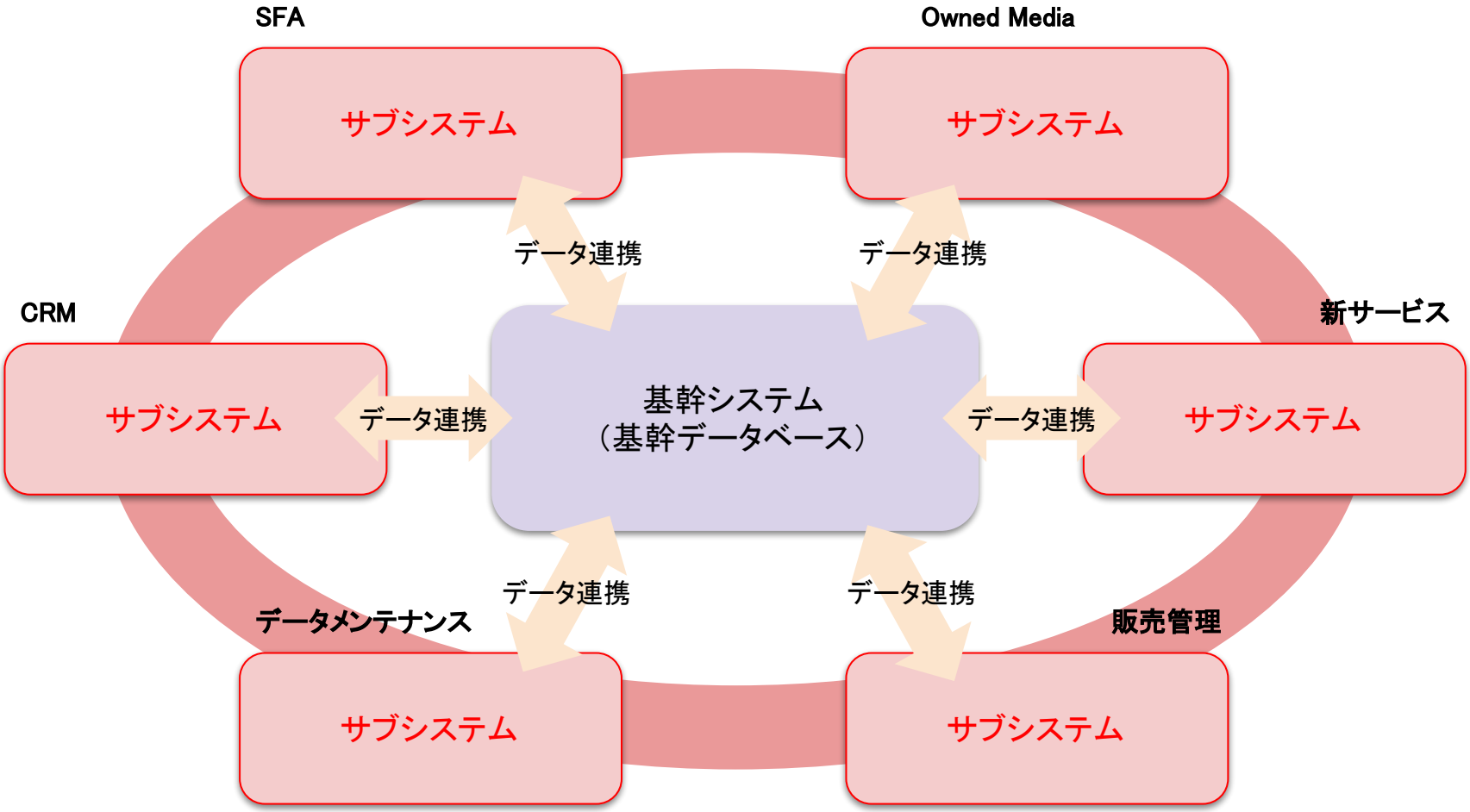
  であれば、十分選択肢に入ります

AWSを活用した業務システム開発のこれから

フェーズ3

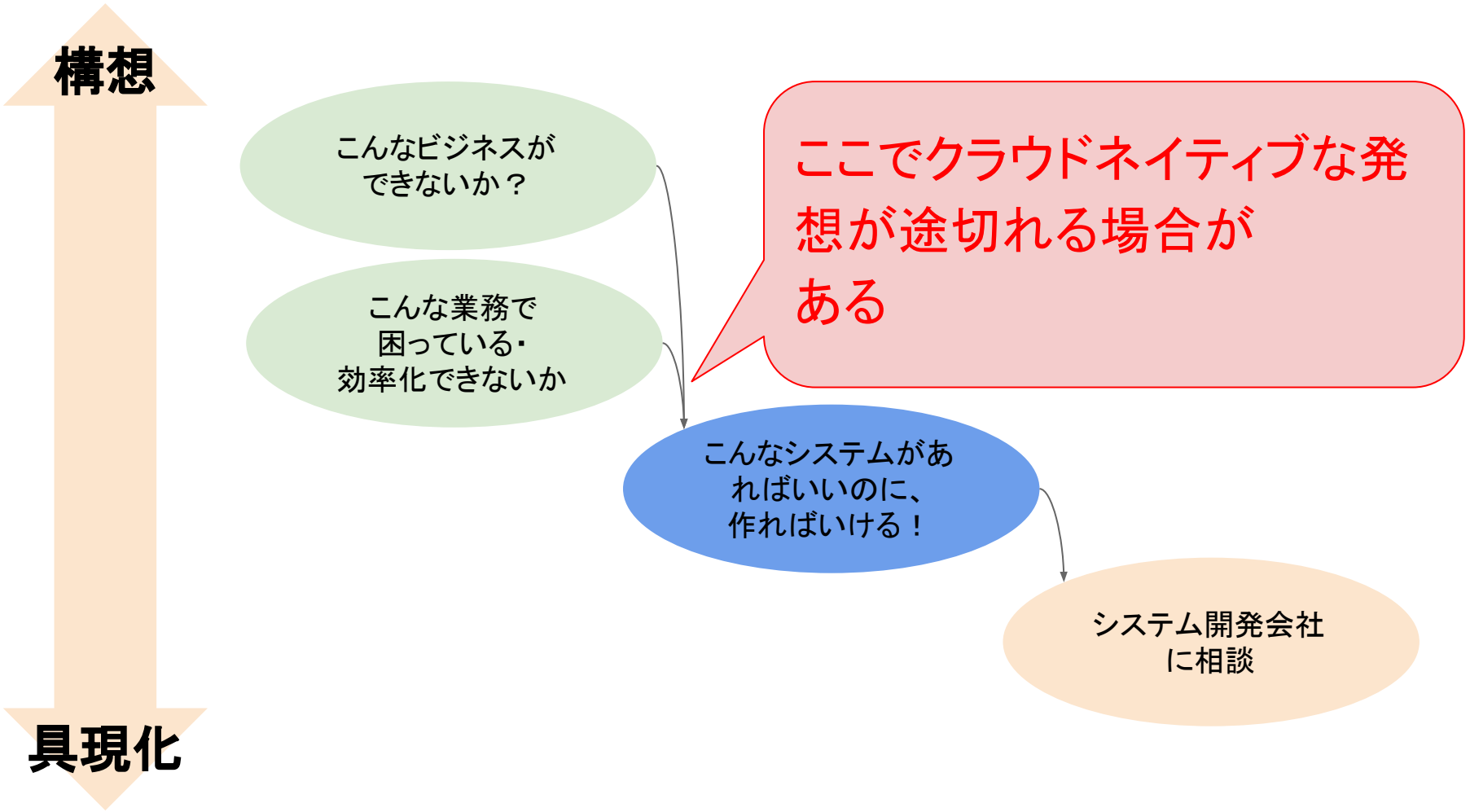


鈴木商店の主なシステム開発領域

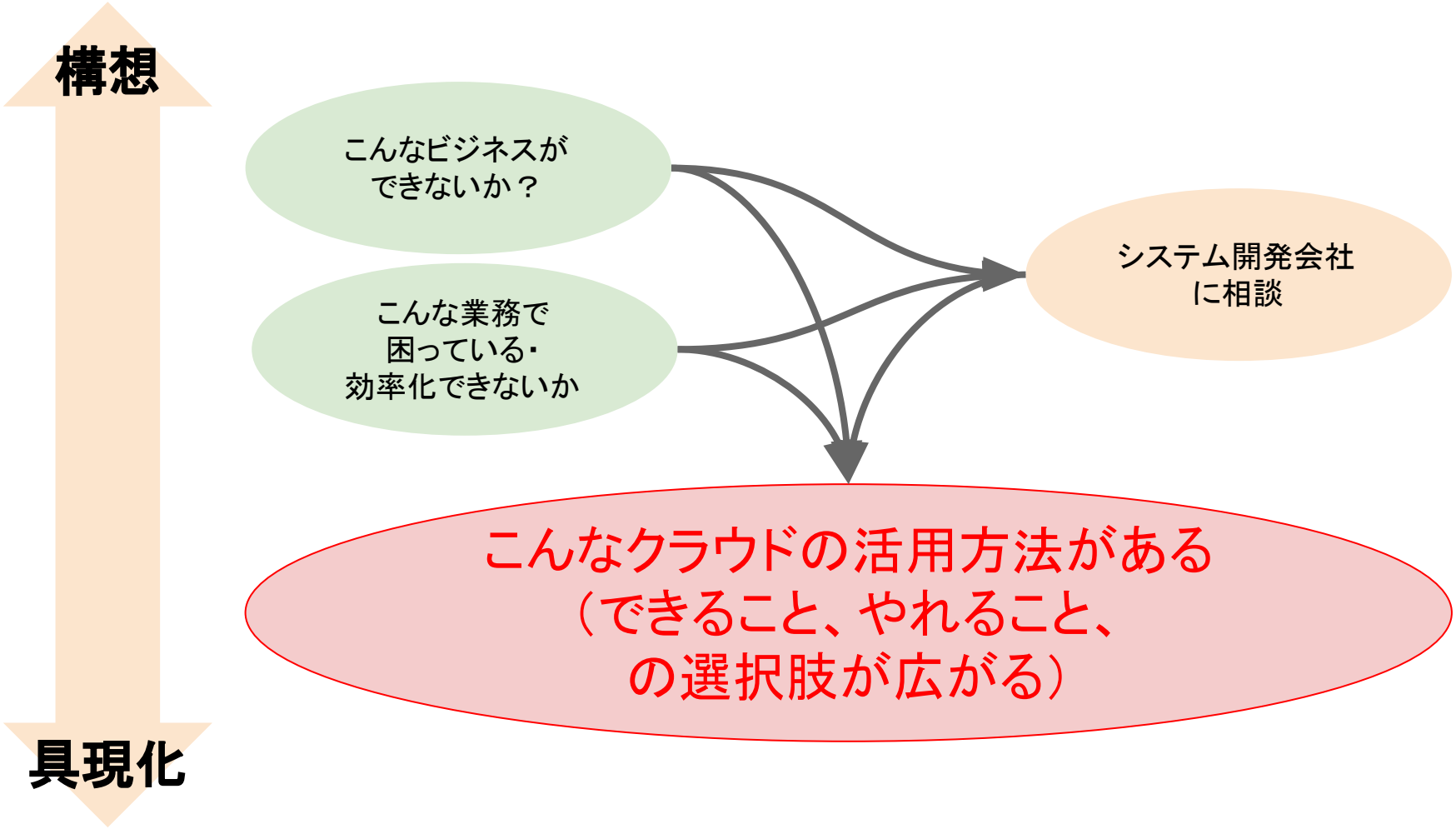


※この限りではありません

発想の転換



発想の転換



本日のポイント

- **業務システムは、従来の多層アプリケーションが多かった**
  - オンプレミスの延長線上にあるシステムの需要
  - AWSを活用することで価格低減やできることが拡大する
  - AWSはシステムの運用アウトソースを柔軟にする
- **今後はクラウドネイティブな要素を含んだ業務システムが増加する**
  - クラウドネイティブな考えを早い段階で取り入れることが競争力を上げる