CNBF Meetup

2020.1.29

「家族アルバム みてね」における Amazon EKSへの移行の取り組み

Isao Shimizu

mixi, Inc.

自己紹介

清水 勲 （しみず いさお） @isaoshimizu

所属

- 株式会社ミクシィ Vantageスタジオ みてね事業部 開発グループ SREチーム

経歴

- SIerで約8年間、受託開発、プロダクト開発を経験後、2011年に株式会社ミクシィへ入社
- SNS運用、モンスターストライクのSREを経て、**現在「家族アルバム みてね」のSRE**
  2019年5月 AWS Summit Tokyo
  コンテナ移行ってこんなに大変？ ～「家族アルバム みてね」を支えるインフラの裏側～
- 2020年1月 SRE NEXT 2020
  SREがセキュアなWebシステムを構築、維持するためにやれることはなにか
- その他「SRE Tech Talks #2」「hbstudy #76」「Internet Week 2017」等

アジェンダ

1. 「家族アルバム みてね」について
2. はじめに
3. みてねにおけるシステムの概要と課題
4. Amazon EKS（Kubernetes）を使うことで得られること
5. Kubernetesを導入する前に考えたいこと
6. まとめ

1 家族アルバム みてね

1.「家族アルバム みてね」について

子どもの写真・動画を、無料・無制限に共有できるアプリ

2015年サービスリリース

現在、**国内外**の利用者数 **600**万人以上

多言語対応（日本語、英語、韓国語、繁体字）

App Store レビュー **★4.7**

Google Play ストア レビュー **★4.7**

みてね　FamilyAlbum　家家相簿

1.「家族アルバム みてね」について

海外での受賞歴

2019 THE WEBBY AWARDS

国際デジタル芸術科学アカデミー (IADAS) によって毎年主催され、優れたインターネットに贈られる賞

2019 NATIONAL Parenting Product(NAPPA) AWARDS

家族にフォーカスした賞（おもちゃや家族向けの商品に特化）

1.「家族アルバム みてね」について

「みてねプレミアム」を2019年4月リリース

月額課金型の有料プラン

月額 480円

- ブラウザから写真や動画のアップロードが可能に
- 1秒動画の**毎月&年間版**配信
- 全商品が**送料無料**（家族全員）
- 公開範囲を細かく設定可能
- 動画アップロードの時間制限を延長 3分から**10分**へ

みてねをお使いの方、ぜひお試しください！

2

はじめに

2. はじめに

この発表について

- 「家族アルバム みてね」のインフラにおいて、コンテナやKubernetesの導入前後での変化、導入にあたって考えるべきことについてお話します。
- コンテナやKubernetesの詳細な技術、技法については紹介しません。
- ハッシュタグ **#cnbfmeetup**

2. はじめに

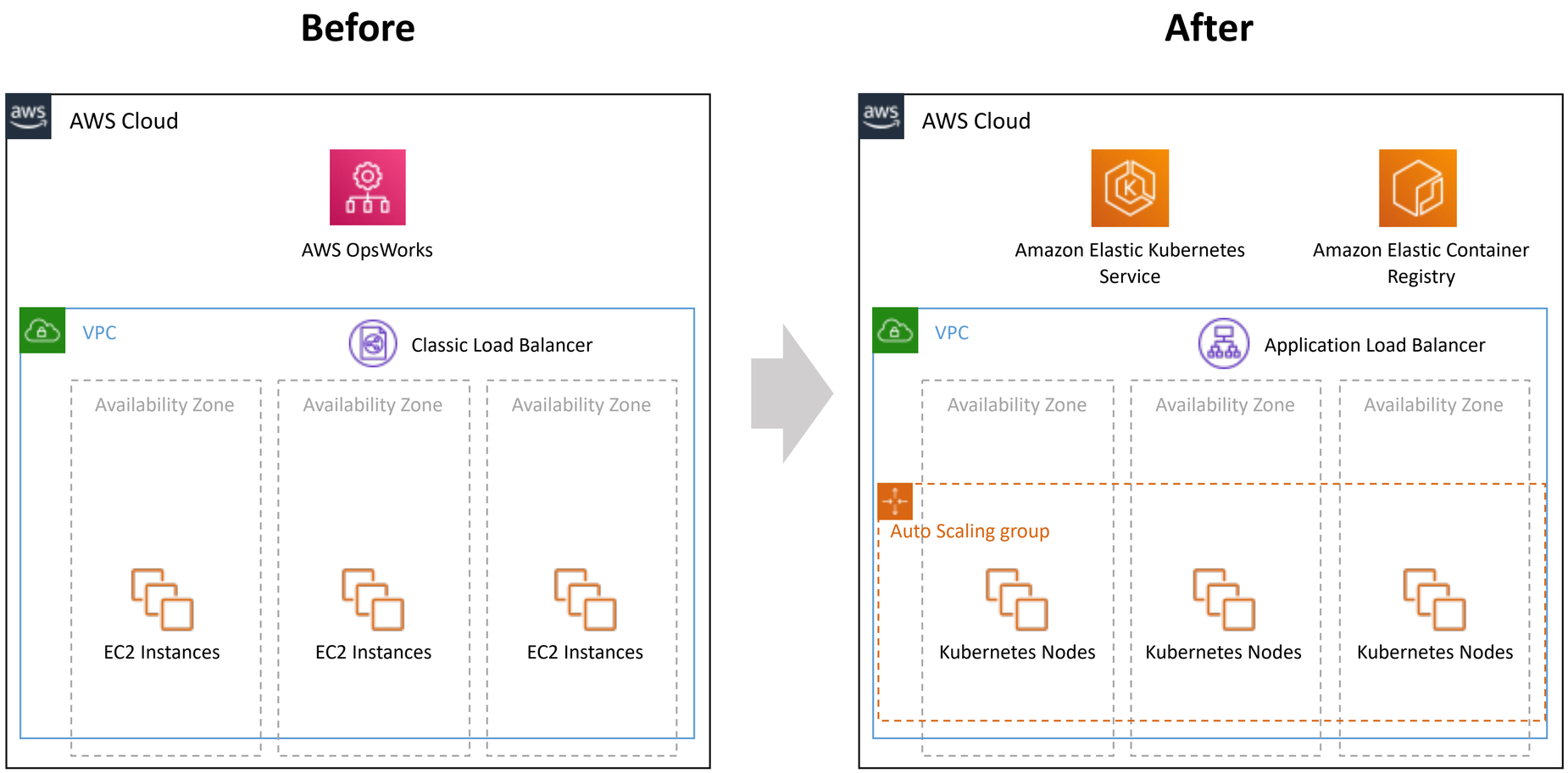
本日お持ち帰りいただきたいこと

- コンテナやKubernetesを導入することで得られるもの
- コンテナやKubernetesを導入する前に必要なこと

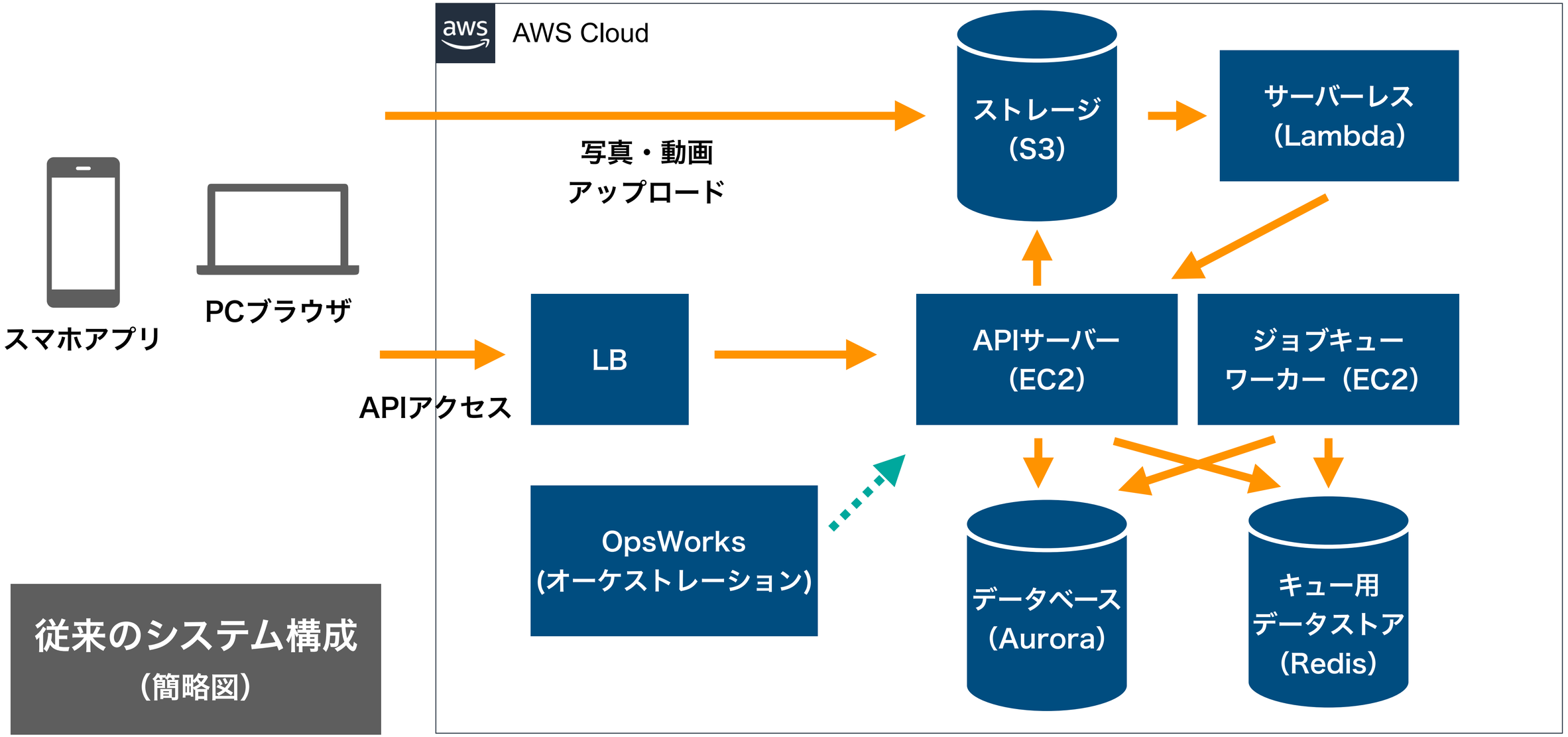
3 みてねにおけるシステムの概要と課題

3. みてねにおけるシステムの概要と課題

みてねのインフラはKubernetes（Amazon EKS）への移行中



3. みてねにおけるシステムの概要と課題



従来のシステム構成
（簡略図）

3. みてねにおけるシステムの概要と課題

従来のシステム構成

- クライアント
  - iOS、Android、PCブラウザ（一部機能）
- サーバー
  - すべてAWS
  - Ruby on Railsアプリケーション
  - クライアント・サーバー間はREST API（CloudFront経由）
  - データベースはAmazon Aurora MySQL
  - ジョブキュー（Sidekiq）を使った非同期処理
  - 写真や動画はAmazon S3へ保存（アップロード/ダウンロードはCloudFront経由）
  - オーケストレーションはAWS OpsWorksを利用

3. みてねにおけるシステムの概要と課題

従来の運用

- AWS OpsWorksの仕様に依存したオペレーション
- Web UI（マネジメントコンソール）による手作業
- オートスケールはVM（EC2）単位
- Chefを利用したプロビジョニング、デプロイ
- オンデマンドタイプのインスタンスのみの利用

従来の運用の課題はどこにあるか

3. みてねにおけるシステムの概要と課題

従来の運用

- AWS OpsWorksの仕様に依存したオペレーション
  - **AWS独自の仕様の把握が難しい、ドキュメントを見てもわからないことがある。**
  - **サポートへ問い合わせないとわからないこともある。**
- Web UI（マネジメントコンソール）による手作業（CLIやAPI操作は可能）
  - **属人化しやすい、作業漏れが起きやすい、手順書を書いても更新されにくい。**
- オートスケールはVM（EC2）単位
  - **VM（EC2）の起動が遅い（プロビジョニングとデプロイとも絡む問題）。**
- Chefを利用したプロビジョニング、デプロイ
  - **CookbookやRecipeの記述が必要、適用速度が遅い。**
- オンデマンドタイプのインスタンスのみの利用
  - **スポットインスタンスを活用したい（できないわけではないがハードルが高い）。**

4

Amazon EKS（Kubernetes）を使うことで得られること

Amazon EKSとは

4. Amazon EKS（Kubernetes）を使うことで得られること

Amazon EKSとは

- 2018年6月リリース
- KubernetesのMaster（コントロールプレーン）ノードをAWSが管理、提供
- Kubernetesのエコシステム、OSSやツールがそのまま使える
- AWSの各サービスと連携できる
- 本番環境のワークロードを実行できるように設計されている
- 今後のロードマップを常時公開
  https://github.com/aws/containers-roadmap/
- 2020年1月21日 コントロールプレーンの利用料が従来の半額に

Amazon EKS（Kubernetes）で
何が変わるか

従来の運用との比較

4. Amazon EKS（Kubernetes）を使うことで得られること

従来の運用

- AWS OpsWorksの仕様に依存したオペレーション
  - **仕様の把握が難しい、ドキュメントを見てもわからないことがある。**
  - **サポートへ問い合わせないとわからないこともある。**

Kubernetesにおける運用

- Kubernetesの仕様に依存したオペレーション
  - **ドキュメントを見て、ソースコードを読んで理解できる。**
  - **事例やノウハウが急増中。**
  - **Meetupやカンファレンス等のコミュニティが数多くある。**

4. Amazon EKS（Kubernetes）を使うことで得られること

従来の運用

- Web UI（マネジメントコンソール）による手作業（CLIやAPI操作は可能）
  - **属人化しやすい、作業漏れが起きやすい、手順書を書いても更新されにくい。**

Kubernetesにおける運用

- CLI、Infrastructure as Code、CIOps、GitOpsが基本
  - **さまざまなツールが用意されている。新たなツールもどんどん生まれている。**
  - **コード化によって属人化を最小限に。手順書は最低限に。**

4. Amazon EKS（Kubernetes）を使うことで得られること

従来の運用

- オートスケールはVM（EC2）単位
  - **VM（EC2）の起動が遅い（プロビジョニングとデプロイとも絡む問題）**

Kubernetesにおける運用

- オートスケールはコンテナもしくはVM（EC2）単位
  - **VMのリソースを効率良く使い、高速にコンテナを起動できる。**
  - **VM単位のスケーリングは、最小限のVMを使うことで従来より高速に。**

4. Amazon EKS（Kubernetes）を使うことで得られること

従来の運用

- Chefを利用したプロビジョニング、デプロイ
  - **CookbookやRecipeの記述が必要、適用速度が遅い**

Kubernetesにおける運用

- Dockerfileの内容をベースにあらかじめイメージを作成し、イメージをデプロイ
  - **イメージをそのまま利用できるため都度プロビジョニングは不要。**
  - **デプロイ時間のほとんどはイメージのダウンロード時間（キャッシュ可能）。**

4. Amazon EKS（Kubernetes）を使うことで得られること

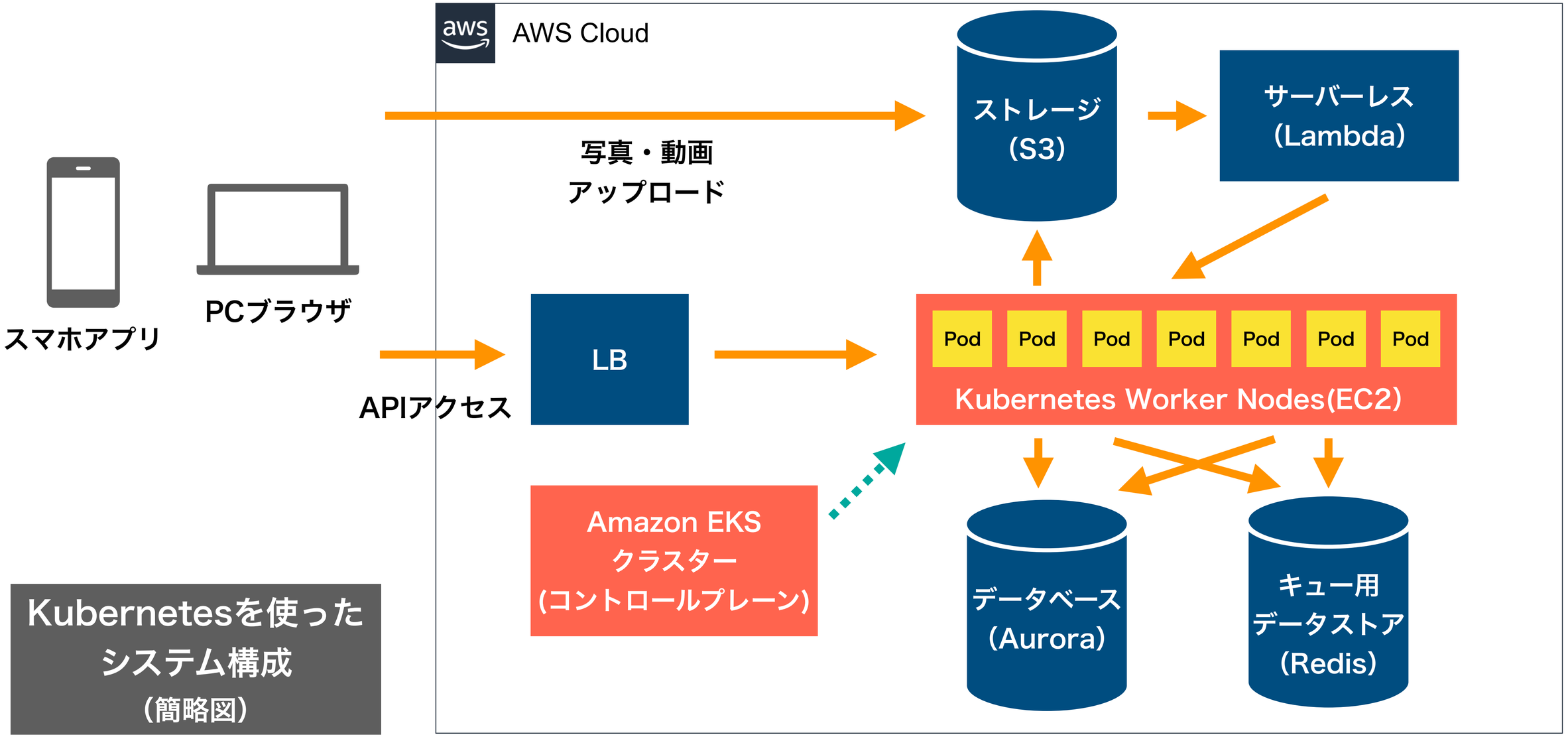
従来の運用

- オンデマンドタイプのインスタンスのみの利用
  - **スポットインスタンスを活用したい（できないわけではないがハードルが高い）。**

Kubernetesにおける運用

- スポットインスタンスを積極的に利用できる
  - **コンテナはディスポーザブル（廃棄可能）であるべきという原則。**
  - **スポットインスタンスの利用によって大幅なコスト削減につながる。**

4. Amazon EKS（Kubernetes）を使うことで得られること



Kubernetesを使った
システム構成
（簡略図）

4. Amazon EKS（Kubernetes）を使うことで得られること

いいことばかりですね 😊

4. Amazon EKS（Kubernetes）を使うことで得られること

いいことばかりではない 😣

4. Amazon EKS（Kubernetes）を使うことで得られること

Kubernetesにおける運用のつらみの一例（個人差あります）

- ツールや手法の選定
  - 一般的な正解はたぶんありません。ワークロードやユースケース次第。
  - デプロイ、ログ転送、監視・モニタリングなど多種多様なツールが存在。
  - ツール自体のアップデートも頻繁におこなわれる。
- Kubernetesの進化（アップデート）への追従
  - 3ヶ月に1回くらいの頻度でアップデートがリリースされます。
- 学習することはそれなりにある
  - 全部を学習する必要はありません。

5

Kubernetesを導入する前に考えたいこと

本当にKubernetesでしか解決できないか

5. Kubernetesを導入する前に考えたいこと

本当にKubernetesでしか解決できないか

- 本質的な課題を再度見つめてみる
- 他にも選択肢がないかどうか
  - AWSの場合
    - Amazon ECS（AWS独自のコンテナサービス）
    - AWS Lambda（サーバーレス）
    - Amazon EC2 + Auto Scaling Groups など
- サービスやプロダクトの規模、成長度、ワークロードに合っているかどうか
- コンテナ化する前にやるべきことがあるのかどうか
  - たとえば、ステートレスな設計、実装が求められる

組織の文化や企業のワークフローと合っているか

5. Kubernetesを導入する前に考えたいこと

組織の文化や企業のワークフローと合っているか

- 高頻度なデプロイ
  - 毎日デプロイできる環境つくり
- 高頻度なアップデート
  - 古くなったソフトウェアを利用し続けるリスク
- 自動化されたユニットテスト
  - 当たり前に書かれ、CIツールによって自動で実行されている
- 目指すべきSLA（があれば）を考える
  - 100%はありえない
- サービス・プロダクトオーナーの理解
  - それなりの時間やコストを必要とすることに対しての理解

5. Kubernetesを導入する前に考えたいこと

自チームで運用できるかどうか

5. Kubernetesを導入する前に考えたいこと

自チームで運用できるかどうか

- なるべく属人的にならない運用
  - 「理解していません」「わかりません」の状態をできるだけ作らない
- Kubernetesの進化や変化へのキャッチアップ、追従
  - アップデートは必須
  - 都度、アップデートによって得られるもの、無くなるものを理解する必要がある
- モニタリングや監視の体制、障害時の体制
  - 自分たちで体制を作れるか
  - 発生した問題の原因を調査できるか
  - 必要なツール、SaaSなどを導入して、可観測性を向上できる体制か

6

まとめ

6. まとめ

- Kubernetesで解決できる課題は多いが、Kubernetes以外の選択肢もある。
- すべてにとって共通の正解というものはない（銀の弾丸などない）。
- 自分のサービス・プロダクトにフィットするかどうか。
- やってみないとわからないことも多い。
- より小規模なものを短いサイクルで数多くトライして、最適解を探す。
- 開発、運用する組織との相性も十分に考慮する必要がある。
- 未来永劫使える便利ツールというものはない。
- 変化に対応しやすい組織作りやコードを書くことが大事。