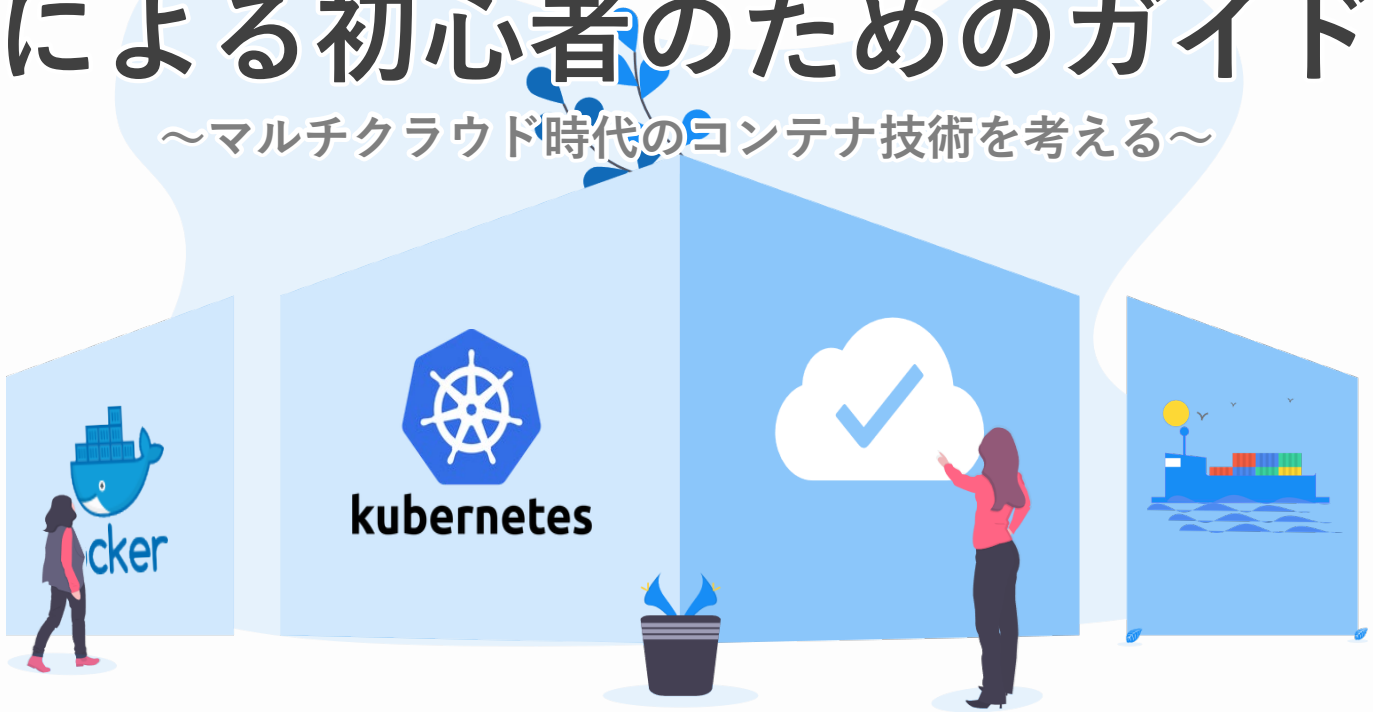
# 分科会発表会

# コンテナへの第一歩!
# 初心者による初心者のためのガイドライン

～マルチクラウド時代のコンテナ技術を考える～




2022年2月10日

# アジェンダ

- ✓はじめに
- ✓コンテナ技術とは一体何か
- ✓コンテナ使うと何ができる？
- ✓コンテナが苦手なこと
- ✓コンテナやりたい、何をやればいい？
- ✓コンテナ技術を導入して開発する為に必要なこと
- ✓コンテナ運用にあたって考えること
- ✓導入事例
- ✓さいごに

# 分科会構成メンバー

## ✓中日本と東日本の合同メンバーで構成

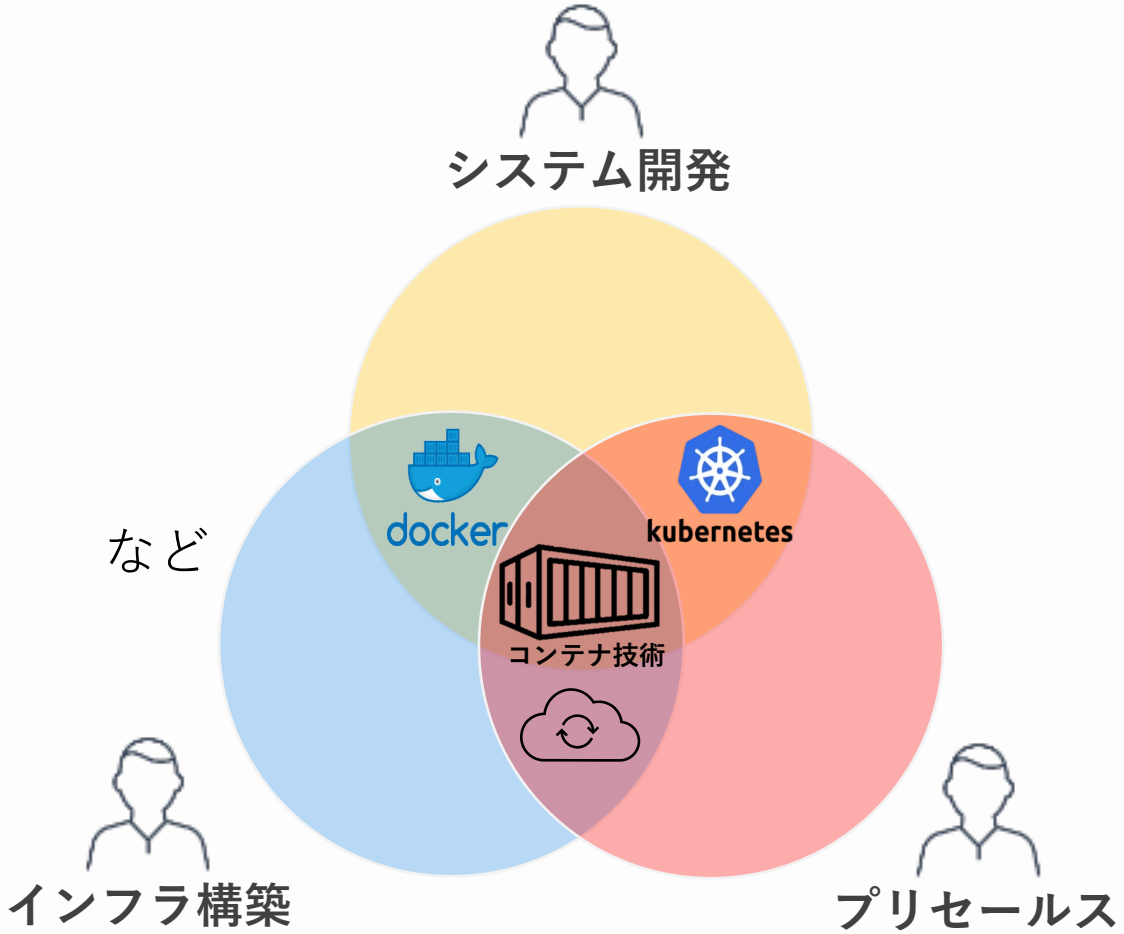
| ロケーション | 氏名/役割 | 会社名 |
|---|---|---|
| 中日本 | 曽我 周平/リーダー | エスツーアイ株式会社 |
| | 松宮 望 | リコージャパン株式会社 |
| | 杉浦 有香 | アビームシステムズ株式会社 |
| | 加藤 弘也 | 株式会社アシスト |
| | 杉山 優介/事務局窓口 | 株式会社アシスト |
| 東日本 | 釼地 勇気/サブリーダー | 三菱ケミカルシステム株式会社 |
| | 奥牧 義己 | 株式会社IHIエスキューブ |
| | 長井 悠徒 | 清水建設株式会社 |
| | 金 雨淵 | 株式会社ミントウェーブ |
| | 小山 雄貴 | 株式会社アシスト |
| | 岡村 卓也/事務局窓口 | 株式会社アシスト |

# メンバー共通の思い

## ✓担当業務は異なるものの分科会テーマに対する思いが一致

- **多種多様な担当業務**
  - システムの受託開発
  - システムインフラの構築、運用
  - 社内向けシステムの開発、管理
  - ソフトウェアのプリセールス　　など

- **テーマに対する共通の思い**
  - いずれコンテナ技術を利用したい
  - 技術習得のハードルが高いと感じている　　など



# 研究の目的

## ✓目的

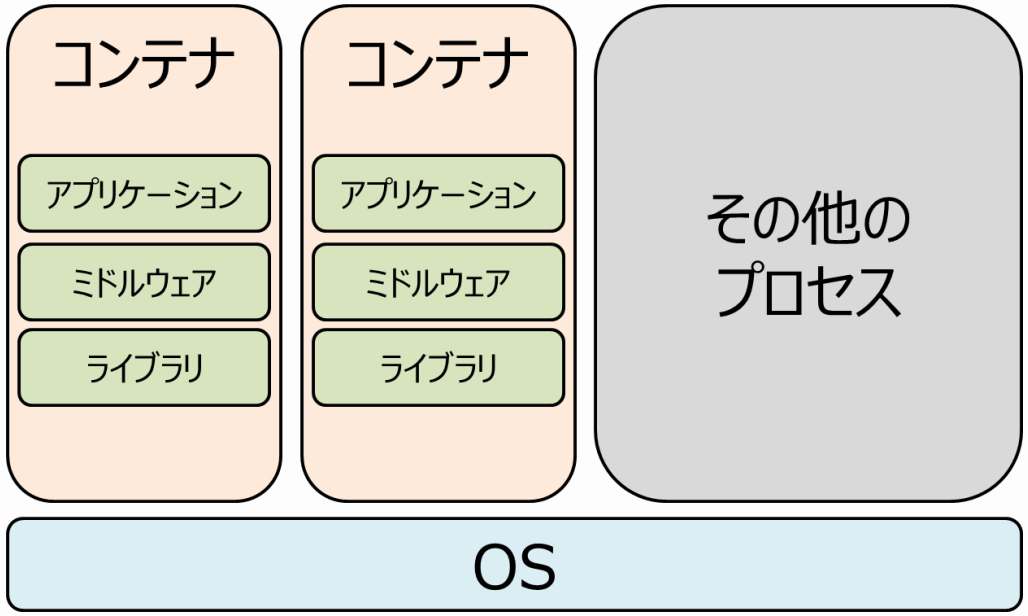
- **ガイドラインの作成**を目的として設定

- メンバーと同様の思いを持つ技術者を読者に想定し、
  **基礎学習の助け**とすることが狙い

# コンテナについて

## ✓コンテナとは

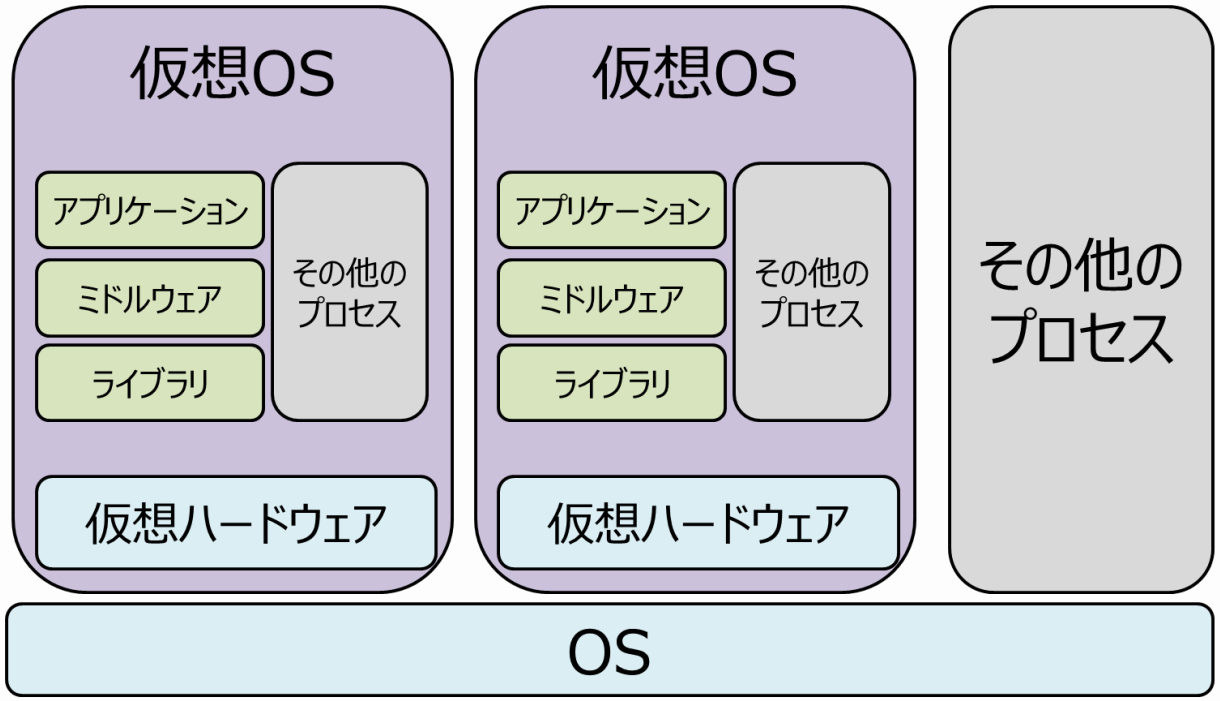
- アプリケーションおよびミドルウェア等の依存物をパッケージ化。
- アプリケーションの実行に必要となるモノがまとまっており、
  環境による影響を抑えることが可能。
- ホストOSのハードウェアリソース等を利用して動作する。
- 各コンテナは独立しており干渉しない。



# 仮想マシン(サーバ)について

## ✓仮想マシン(サーバ)とは

- ハードウェアやOS等を仮想化し、ホストOS上に端末として作成。
- 端末そのものを仮想化しているため、ホストOSへの影響を抑えることが可能。
- アプリケーションは、仮想OSのハードウェアリソース等を利用して動作する。
- 各仮想マシンは独立しており干渉しない。



# コンテナと仮想マシンとの違い

## ✓コンテナと仮想マシンを比較

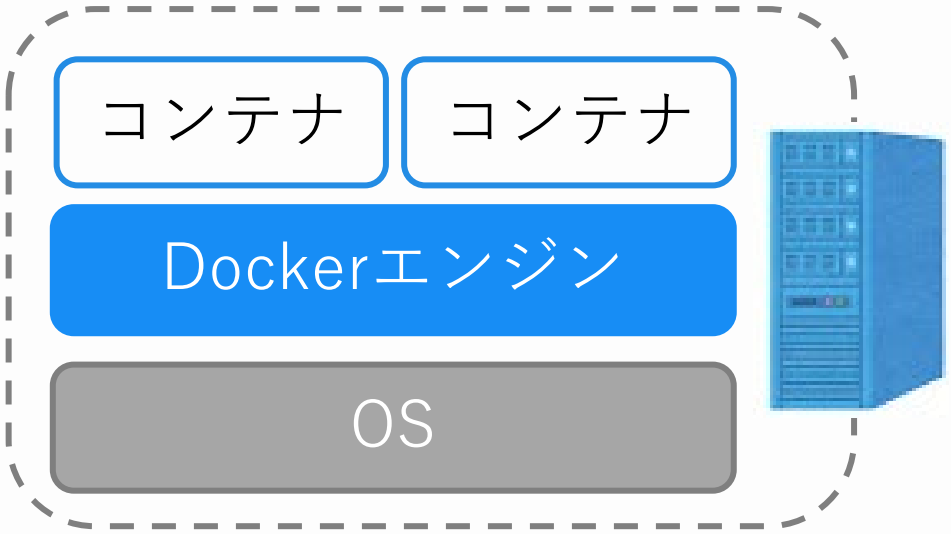
- コンテナ　：アプリケーションの実行を目的とした実行環境の仮想化。
- 仮想マシン：端末そのものの仮想化。

| 比較内容 | コンテナ | 仮想マシン(サーバ) |
|---|---|---|
| 利用用途 | アプリケーションの実行 | 複数のアプリケーションの実行など |
| 仮想化する対象 | アプリケーションの<br>実行に必要な環境 | 端末そのもの |
| 自由度 | 小 | 大 |
| リソース消費量 | 小 | 大 |

# Dockerとは

## ✓Docker

- コンテナを簡単に管理するためのプラットフォーム




サーバーにインストールした
Dockerエンジンの上でコンテナが動作する

### Dockerコンテナ実行の流れ


定義ファイルからイメージをビルド
(イメージ＝コンテナの設計図)

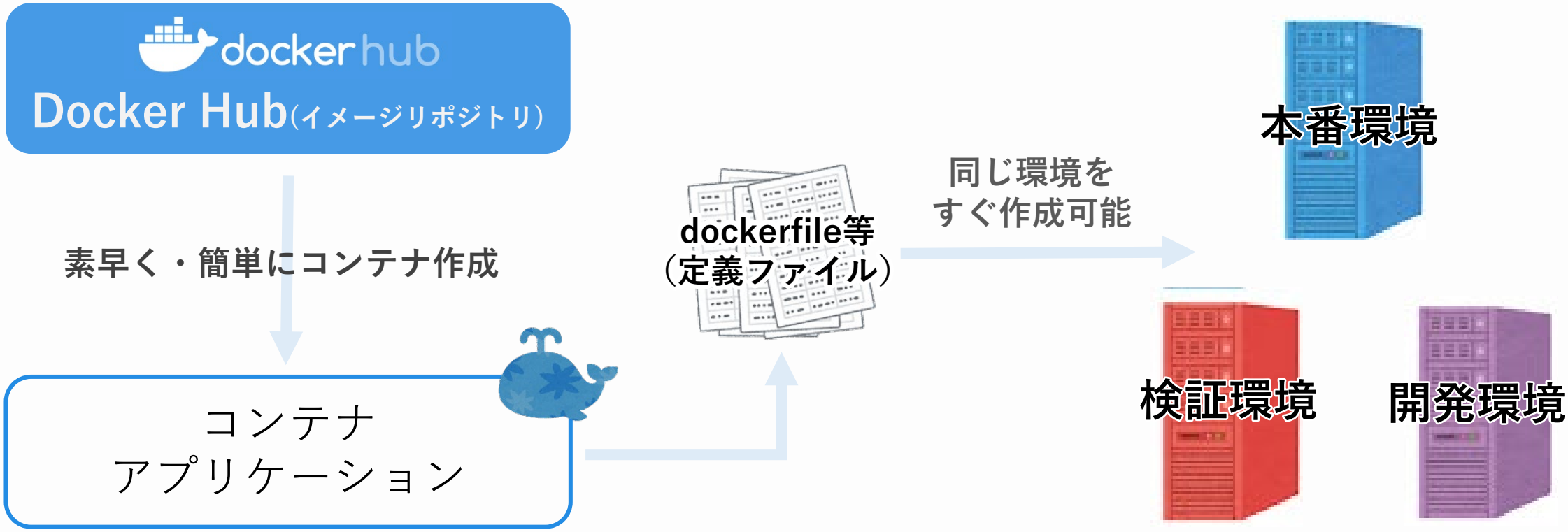
イメージ

イメージからコンテナを作成して実行
実行するたびに作成する

コンテナ

# コンテナ技術のメリット（開発）

## ✓環境構築作業の改善

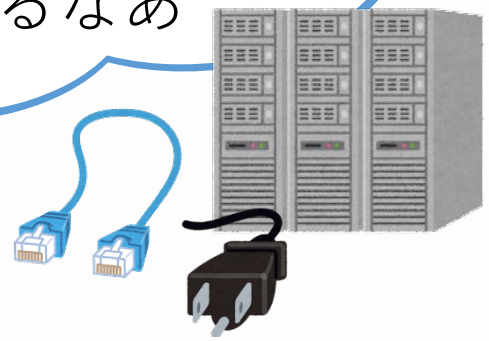



**環境差異によるトラブルを削減できる**

# コンテナ技術のメリット（開発）

## ✓開発時に発生するインフラ部門との連携を改善

機器発注、電源/NW/ラック確認、
サーバ基本設定、
アプリインストール・・・
時間かかるなあ



インフラ担当

コンテナ
アプリケーション

コンテナなら
アプリ担当だけで
アプリケーション
作成ができる！

アプリ担当

**工数の削減につながる**

# コンテナ技術のメリット（運用）

## ✓デプロイ作業の改善

### 物理サーバ・仮想サーバ

OSもあるから
起動に時間がかかるなあ

### コンテナアプリケーション

OS含んでないから
すぐ起動できる

**CI/CDと相性が良くビジネススピードアップ**

# コンテナ技術のメリット（運用）

## ✓インフラ業務の簡略化

### 物理サーバ・仮想サーバ

保守作業、HW障害対応、
構築がほぼ手作業で大変

インフラ担当

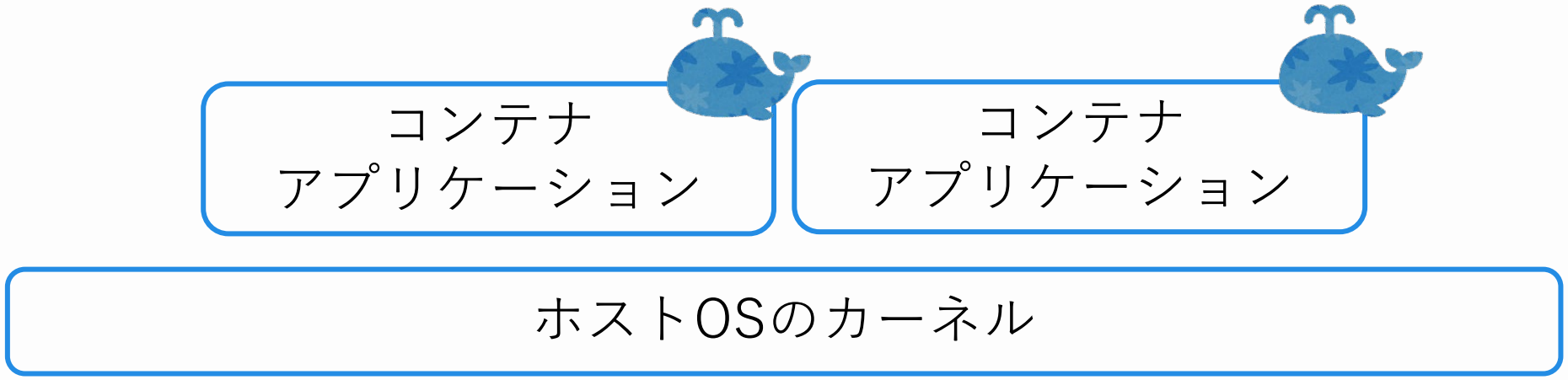
### コンテナアプリケーション


定義ファイルで管理
↓
工数削減、品質を保てる！

**品質向上、インフラ工数削減**

# コンテナ技術のメリット（運用）

## ✓サーバ資源の節約




CPU、メモリ等のリソースをコンテナ間で柔軟に利用可能

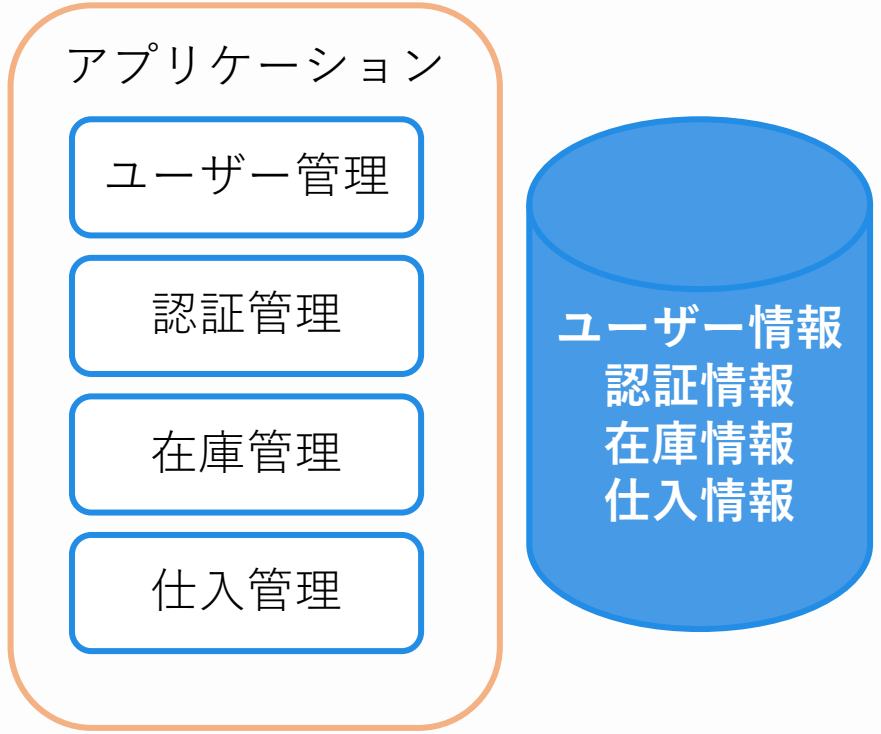
**リソース集約によるコスト削減**

# コンテナが苦手なこと（開発）

## ✓マイクロサービスの設計が難しい

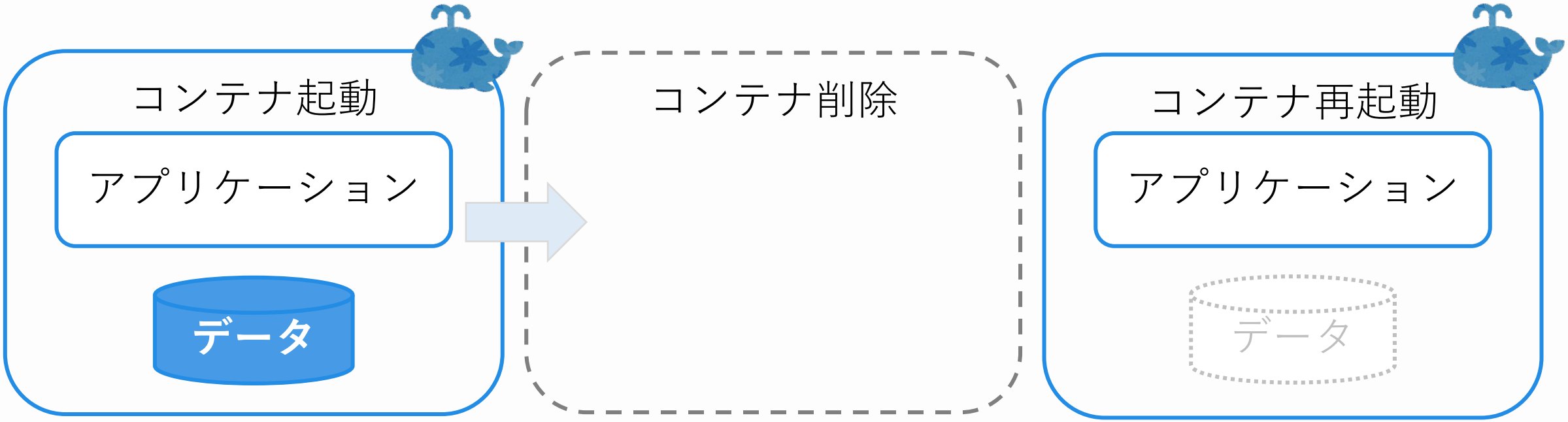
### マイクロサービスアーキテクチャ


### モノリシックアーキテクチャ



# コンテナが苦手なこと（開発）

## ✓データの永続化が苦手




**コンテナが削除されるとコンテナ内のデータも消える**

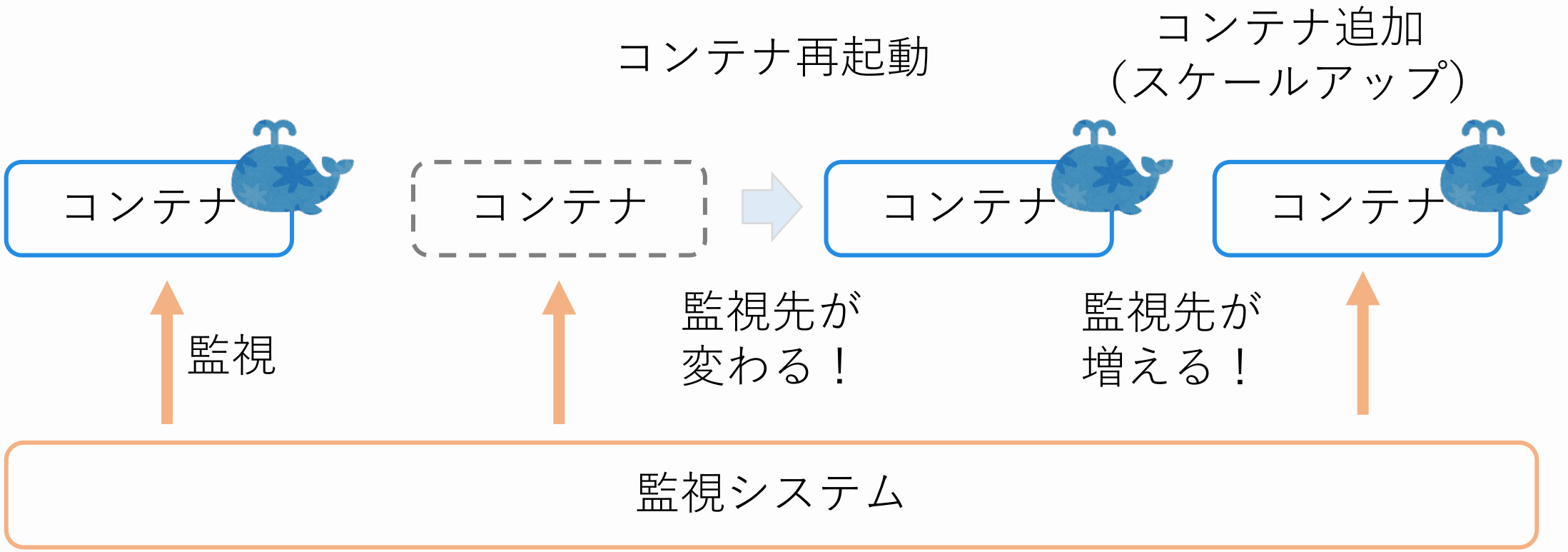
# コンテナが苦手なこと（開発）

## ✓データの永続化が苦手


**永続化したいデータはコンテナ外部に保存する必要がある**

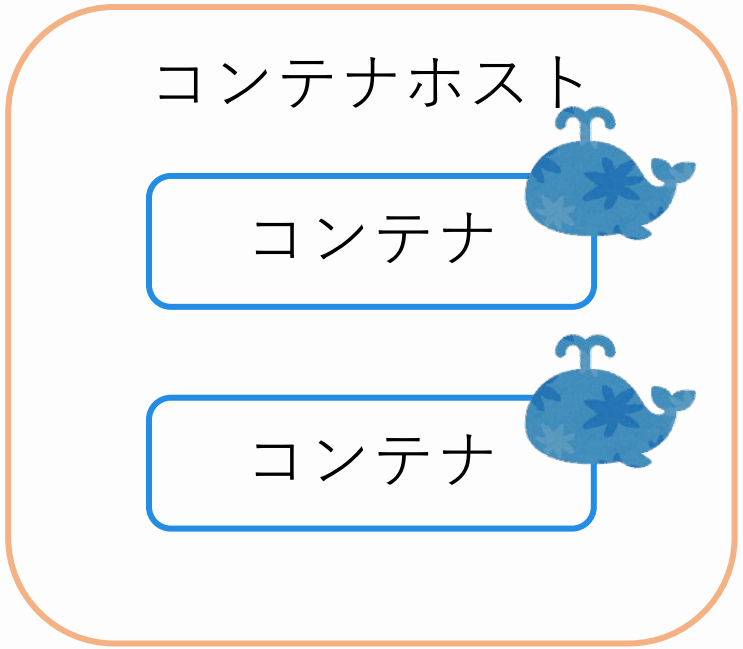
# コンテナが苦手なこと（運用）

## ✓運用監視の見直しが必要

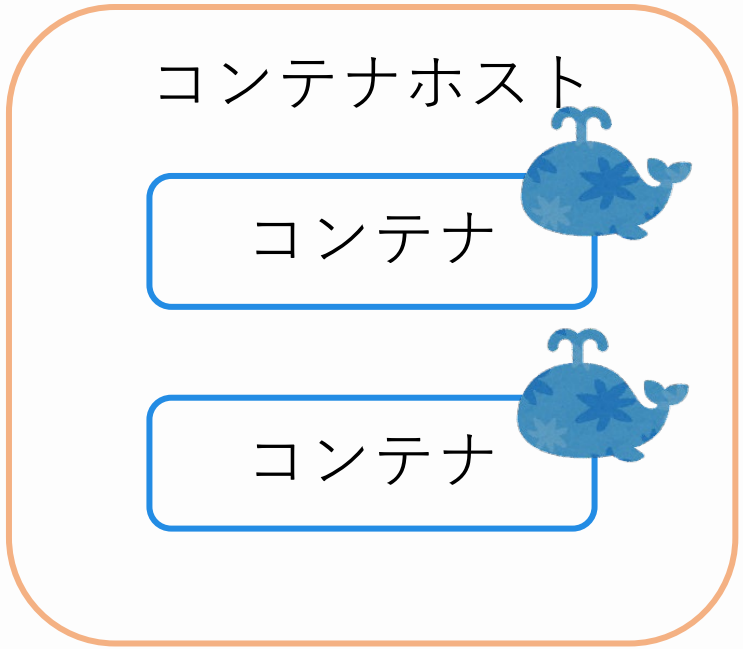


# コンテナが苦手なこと（運用）

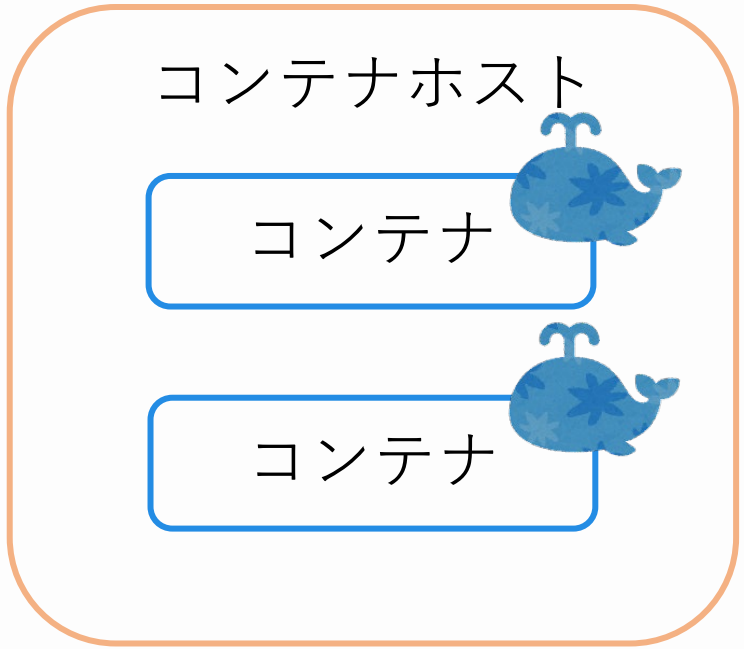
## ✓運用監視の見直しが必要




コンテナはOSを
含まない




1ホストに複数の
コンテナがある



# コンテナが苦手なこと（運用）

## ✓学習コストが高い

- 多岐にわたる知識が必要

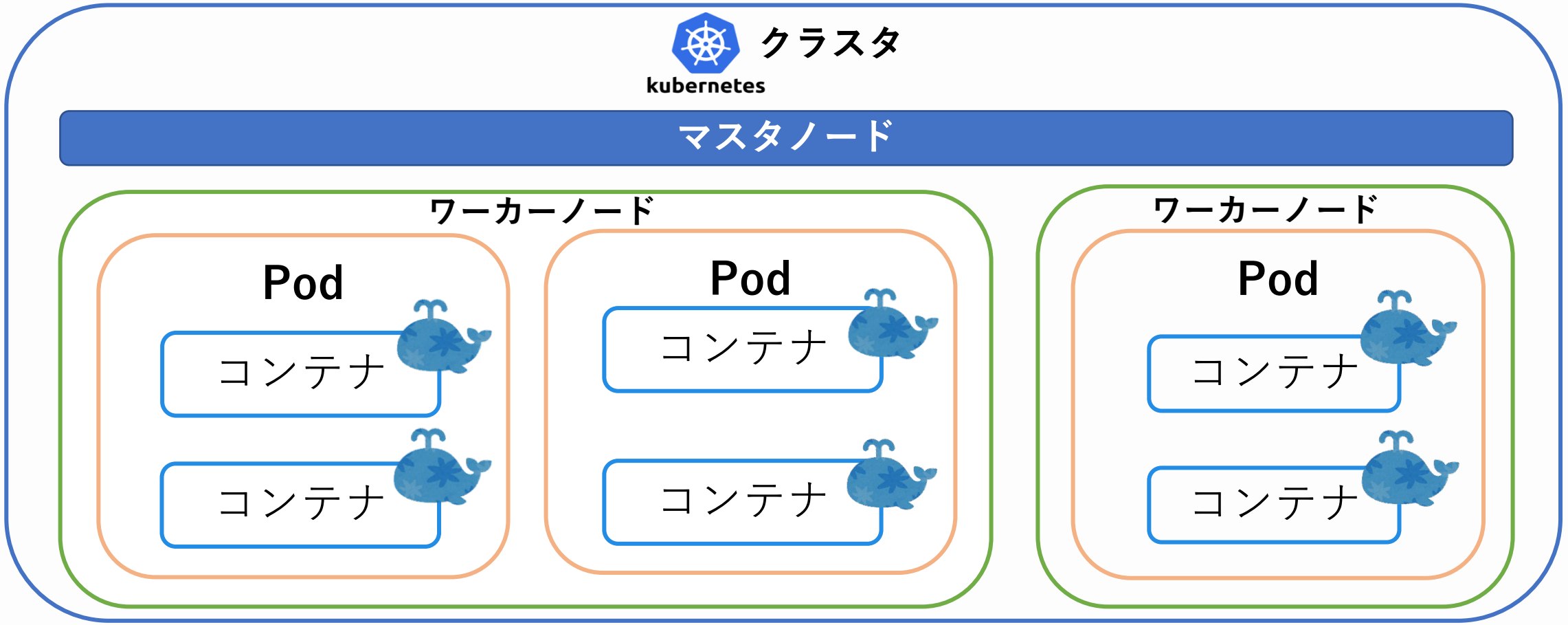
- コンテナ間通信
- 冗長化
- 負荷分散
- オートスケールの管理
- 監視
- バックアップ
- セキュリティ
- コンテナホスト側アップデート

**技術習得者が少ないと**
**コンテナ運用が属人化するリスク！**

# 苦手を補完するKubernetes

✓Kubernetesを用いることで苦手としていたことを補完します。

✓コンテナの監視や自動復旧やオートスケール、ストレージ管理など



# コンテナを利用するための準備

✓コンテナを利用するにあたりどのような準備が必要だろうか
　まず、オンプレミスを前提に見ていきましょう。

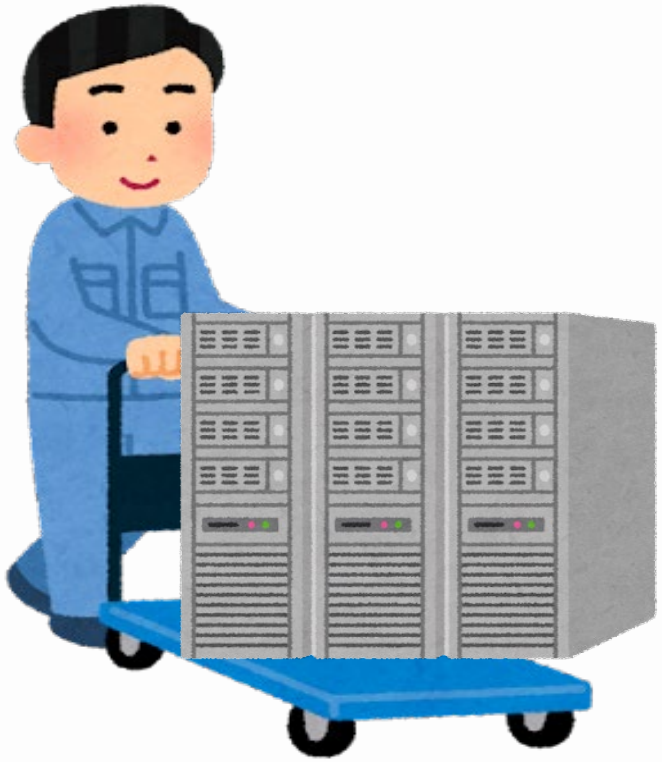

※ここではKubernetesを利用することを前提とする

# コンテナを利用するための準備

✓初めに、必要になるHWの設置や各種設定が必要になります。

**例**

- HW設定
- DC手配
- NW手配
- 電源工事
- ラッキング



| | | | | |
|---|---|---|---|---|
| **HWセットアップ** | **OSセットアップ** | **Kubernetes セットアップ** | **アプリ開発** | **運用** |

# コンテナを利用するための準備

✓Kubernetesを導入する前段階として、OSの各種インストールや設定なども必要です。

**例**

・各種OS設定
　-各種PKGインストール
　-セキュリティ設定

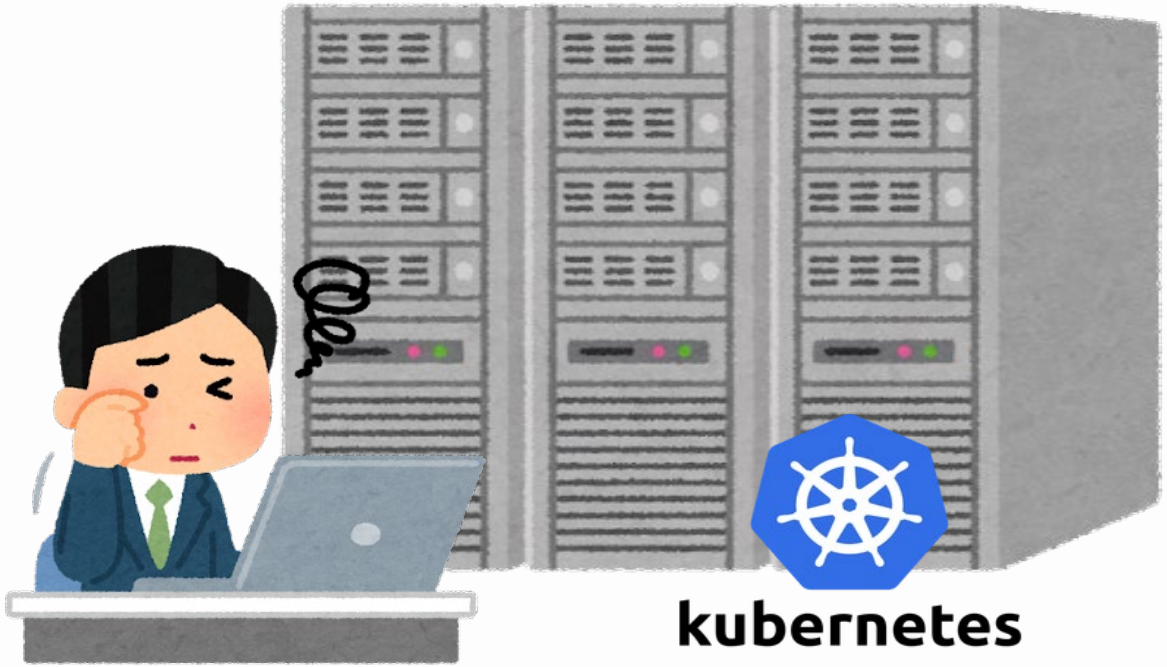
| | | | | |
|---|---|---|---|---|
| **HWセットアップ** | **OSセットアップ** | **Kubernetes セットアップ** | **アプリ開発** | **運用** |

# コンテナを利用するための準備

✓その後、Kubernetesなどコンテナミドルの導入を行います。
初めての場合、構築経験があるベンダを探す必要があります。

**例**

- Kubernetes導入
  - -インストール
  - -各ノード設定
- Kubernetes設計
  - -通信IP/ポート
  - -ストレージ
  - -セキュリティ




| HWセットアップ | OSセットアップ | Kubernetes セットアップ | アプリ開発 | 運用 |
|---|---|---|---|---|

# コンテナを利用するための準備

✓そうしてやっとコンテナが利用可能な状態になります。

✓アプリケーション開発においても新たな考慮事項が多くあります。

**例**

- 標準化
- ストレージ
- 監視
- アプリ設定
  （マイクロサービス）
- バージョン管理
- セキュリティ

| HWセットアップ | OSセットアップ | Kubernetes セットアップ | アプリ開発 | 運用 |
|---|---|---|---|---|

# コンテナを利用するための準備

✓合わせて運用についても考慮する必要があります。

✓従来の物理/仮想と合わせてコンテナの運用も加わるため運用の煩雑化も考えられます。

**例**

- 各レイヤの監視
  (死活/リソース監視)
- ログ管理
- 監視
- バックアップ運用
- 障害対応等

| HWセットアップ | OSセットアップ | Kubernetes セットアップ | アプリ開発 | 運用 |
|---|---|---|---|---|

# コンテナを利用するための準備

✓Kubernetesは構成上大規模になり高額化しやすい傾向にあります。
企画し費用試算や効果試算を行うことや各種担当者との調整が
従来構成に比べて大変であると考えられます。

予算確保

ベンダ選定/要員確保

費用算出/ROI試算

各担当者への教育

# コンテナを利用するための準備

✓Kubernetesは構成上大規模になり高額化しやすい傾向にあります。
企画し費用試算や効果試算を行うことや各種担当者との調整が
従来構成に比べて大変であると考えられます。

予算確保

ベンダ選定/要員確保

**考慮事項が多く、準備がとても大変！**

費用算出/ROI試算

各担当者への教育

# コンテナ利用の救世主

✓クラウド（KaaS）を利用することでインフラの初期構築費用の削減やインフラの運用コストを下げることができ、ハードルを大幅に削減可能だと考えられます

✓KaaS導入時のオンプレミスからの考慮点削減


KaaS導入で大幅に手間が削減可能

KaaS導入で一部手間が削減可能

オンプレミスとあまり変わらない

# コンテナ利用の救世主

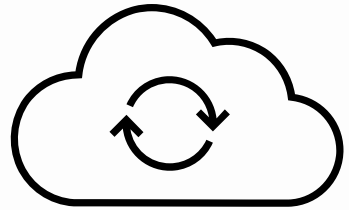
✓これまでのことを考慮すると

**コンテナを始めたい場合はクラウド利用がベスト**

という結論が、当分科会での１つの結論である。


≒


×



# マイクロサービスについての理解

## ✓マイクロサービス(マイクロサービス化)についての理解

- クラウド環境上で稼働するマイクロサービスの基盤技術として
  現在は「コンテナ」が主流となっている。
- コンテナ技術・マイクロサービスについて理解することで、
  コンテナを用いたアプリケーション開発時の課題や検討すべき点が明確になってくると考える。

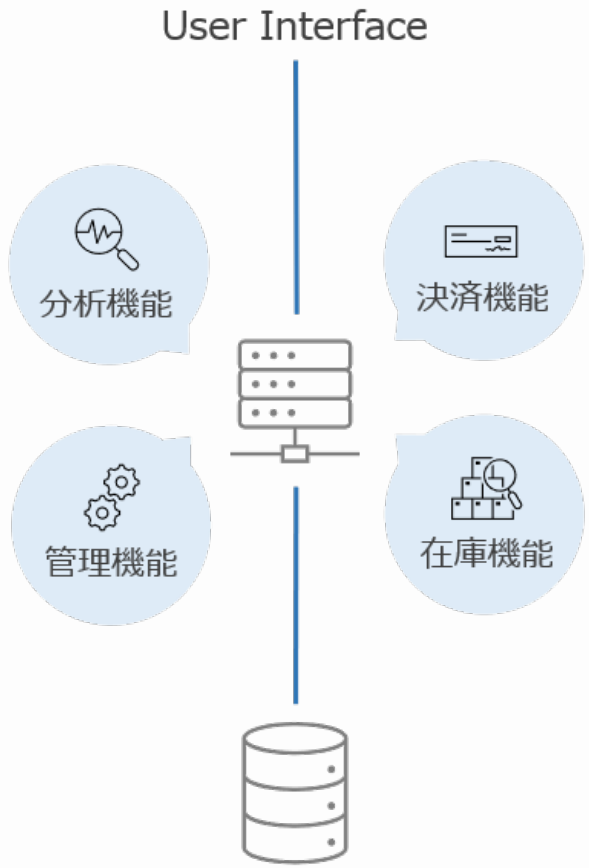
## ✓マイクロサービスとは？？

- 複数の規模の小さなサービス（システム）を組み合わせて、
  一つの大きなアプリケーションを構成するソフトウェア開発技法のこと。

- 具体的なイメージを次のスライドに示す。

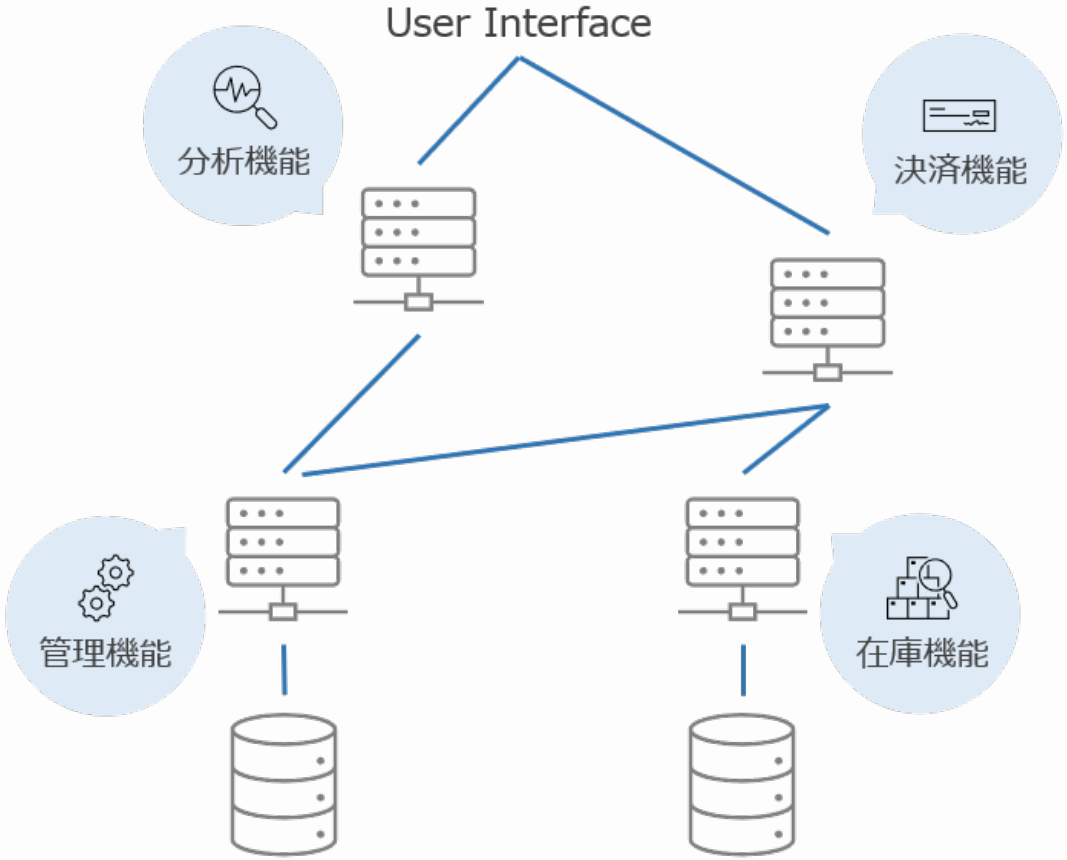
# マイクロサービスについての理解

## モノリシックアーキテクチャ



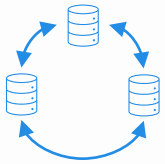

機能が複数に分かれていたとしても、
基本的には１つのマシン上で稼働している。

## マイクロサービスアーキテクチャ



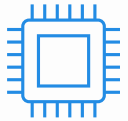

複数の機能が独立したマシン上で稼働する。
それぞれのネットワークを介して連携を行う。

# マイクロサービス化の課題(システム的)



## データの一貫性

サービスごとにデータストアが分かれる為、データの一貫性を保つことが難しくなる。



## パフォーマンス

複数サービスによるAPI連携によって通信が多発し、パフォーマンスが劣化する可能性がある。

## 設計/実装の高難度化

APIが基本となる為、「サービス同士の連携方法」、「API呼び出し失敗時の対応」などの特有の問題が発生する。

## 障害解析の高難度化

障害発生時、複数のサービスの中から原因の特定作業が必要。ログ解析、デバッグ等の作業も増加。

# 対策(システム的)

## ✓ HEROKU ：「The twelve-Factor App」の参照

- マイクロサービスはクラウドをベースに語られることが多い。
  クラウドネイティブを実現するためのデザインパターンとなる。

| No. | 項目名 | | |
|---|---|---|---|
| 1 | One codebase, one application（1 コードベース、1 アプリケーション） | 9 | Environment parity（環境一致） |
| 2 | **API first（API ファースト）** | 10 | Administrative processes（管理プロセス） |
| 3 | Dependency management（依存関係管理） | 11 | Port binding（ポートバインディング） |
| 4 | **Design, build, release, and run（デザイン、ビルド、リリース、実行）** | 12 | Stateless processes（ステートレスプロセス） |
| 5 | Configuration, credentials, and code（設定、機密情報、コード） | 13 | Concurrency（並行性） |
| 6 | Logs（ログ） | 14 | Telemetry（テレメトリ） |
| 7 | Disposability（廃棄容易性） | 15 | Authentication and authorization（認証/認可） |
| 8 | **Backing services（バッキングサービス）** | | |

# マイクロサービス化の課題(組織体制的)

- ✓ amazon：人数編成
  - チームサイズは“Two pizza rule”(1チームは2つのピザで満足できる人数)に抑えることが推奨されている。人数を制限することで、自律性とオーナーシップを持つことが可能。
- ✓ NETFLIX：自立組織化されたチーム
  - 各チームが担当システムの開発と運用に責任を持つことで、ソフトウェアサイクルからフィードバックを直接受けられ、改善サイクルを加速させることが可能。ある程度の裁量を持たせることが重要といえる。
- ✓ 適切な情報公開
  - 主となるAPIの仕様や、システムの仕組みなどについて、ドキュメントをきちんと作成・管理・共有することが非常に大切である。

# インフラ

## ✓ログ管理 


- Kubernetes を活用する上で、ログ管理を設計レベルで落とし込むことが非常に重要。

## ✓死活監視 

- 監視ツールや監視サービスを用いて定期的なステータスチェックを行う。

## ✓バージョン管理 


- マイナーバージョンが約3か月ごとにリリースされるため管理の簡易化が必要。

## ✓リソース管理(上限の考慮) 

- ポッドが消費するリソースの上限値を設定し、共存するポッド同士が悪影響を及ぼし合わないように制御することが必要。

# インフラ

## ✓コスト監視(クラウドサービス)

- オートスケールにより無尽蔵にポッドが増えて想定外の料金を請求されないように、コンテナの最大本数を制限する。

## ✓バックアップ

- Kubernetes クラスタの状態を保存している etcd（Kubernetes クラスタの心臓部）のバックアップが必要。

## ✓ネットワーク

- Kubernetes クラスタを稼働させるために、最初に各ノードへの IP アドレス割り当てが必要。

# インフラ

## ✓セキュリティ

- ①Kubernetes API にアクセスできるユーザーおよびサービスアカウントの権限管理
- ②データストアへのアクセス制御
- ③コンテナイメージへの署名
- ④コンテナの脆弱性確認
- ⑤不正なコンテナの侵入、改竄の検知

# アプリケーション

## ✓パフォーマンスの監視

- アプリケーションのレスポンスタイムや、コンポーネントごとの実行時間を監視し、システム全体の稼働状況を常に把握しておくことが推奨される。

## ✓システム稼働の維持

- 特定の HTTP エンドポイントを開発時から用意しておき、システムの稼働状況に応じてコンテナの再作成やルーティング制御などを自動実行するように実装する。

## ✓本番運用を見据えたテストの実施

- 本番環境で設計通りのパフォーマンスが維持できるか負荷テストを実施することが求められる。 また、データの差異による不具合なども本番環境で発生する恐れがある。

# 導入事例

## ✓クラウド事業者が提供するマネージドサービスの導入事例が多数

- **導入前の課題**
  - 利用者の増加によるレイテンシの発生
  - 既存システムの拡張性の低さ
  - 既存のコンピューティングリソースの不足
  - エンジニアの不足
  - デプロイ作業の煩雑化
- **利用した代表的なクラウドサービス**
  - 

：Amazon Elastic Kubernetes Service（Amazon EKS）
  - 

：Google Kubernetes Engine（GKE）
  - 

：Azure Kubernetes Service (AKS)

# 導入効果

## ✓コンテナ技術を導入する価値は大きい！

- サービス構築の迅速化
- コストの適正化
- ビジネス展開スピードの向上
- セキュリティ強化
- 開発生産性の向上
- レスポンス改善

**サービス品質向上**

# まとめ

✓コンテナは、導入した企業に様々なメリットを享受できる技術

✓たくさんの技術の中であくまでも選択肢の1つ

✓クラウド事業者のマネージドサービスでの利用を推奨

✓ガイドライン以上の部分も考慮しコンテナ技術を考えてください

# 最後に

**成果報告書も読んでみてください！**

こんなことも書いてます

- ✓ここまでの内容をより詳しく
- ✓コンテナ導入事例
- ✓学習のベストプラクティス
- ✓その他コンテナ関連のマネージドサービス