ORACLE

Developer Days

未来を創造する最新テクノロジーを今、あなたの手に。

GraalVM + Helidon + OKEで構築するマイクロサービスアプリケーション

仁井田 拓也

日本オラクル株式会社
テクノロジー・クラウド・エンジニアリング本部
テクノロジー・クラウド・エンジニア

#OraDevDays

ORACLE

Developer Days

未来を創造する最新テクノロジーを今、あなたの手に。

仁井田 拓也
日本オラクル株式会社

- 仁井田 拓也
- 日本オラクル株式会社
  テクノロジー・クラウド・エンジニアリング本部
- 前職は某SIer
- Oracle歴：1年8ヶ月
- Cloud Native歴：1年半
- Kubernetesもここ1年で触り始めました(CKA取得)
- ジブリ大好き

@takuya_0301

#OraDevDays

アジェンダ

1. マイクロサービスの必要性と実装時の課題
2. マイクロサービス実装時の課題に対する解決策
3. GraalVM/Helidon/OKEのご紹介
4. デモンストレーション
5. まとめ

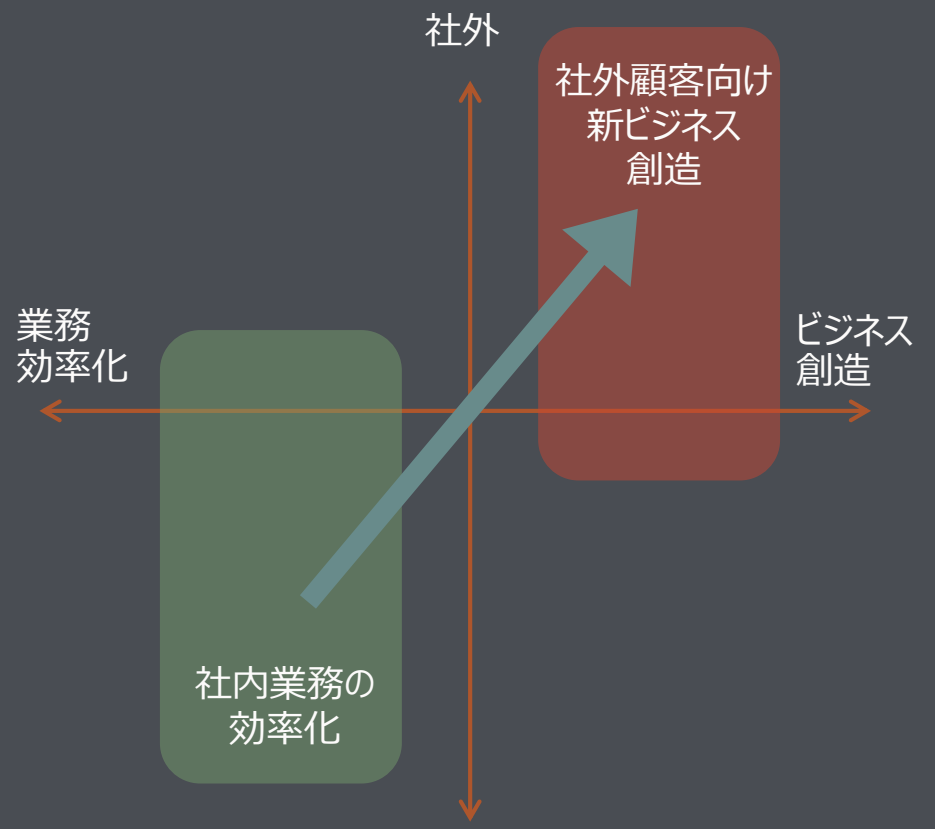
マイクロサービスの必要性と課題

GraalVM + Helidon + OKEで構築する
マイクロサービスアプリケーション

企業システムに求められるニーズの変化

- ビジネスには進化のスピードと品質が求められる
  - サービスの早期リリース
  - 市場の動向/反応を早期フィードバックして対応
  - サービス停止による機会損失をなくす

- システムは変化するビジネス要件に合わせ短い時間で高頻度にリリースすることが求められる
  - アプリの変更をすぐに本番環境に適用
  - 変化を許容できるシステムを構築
  - 停止時間の短縮と運用作業の効率化

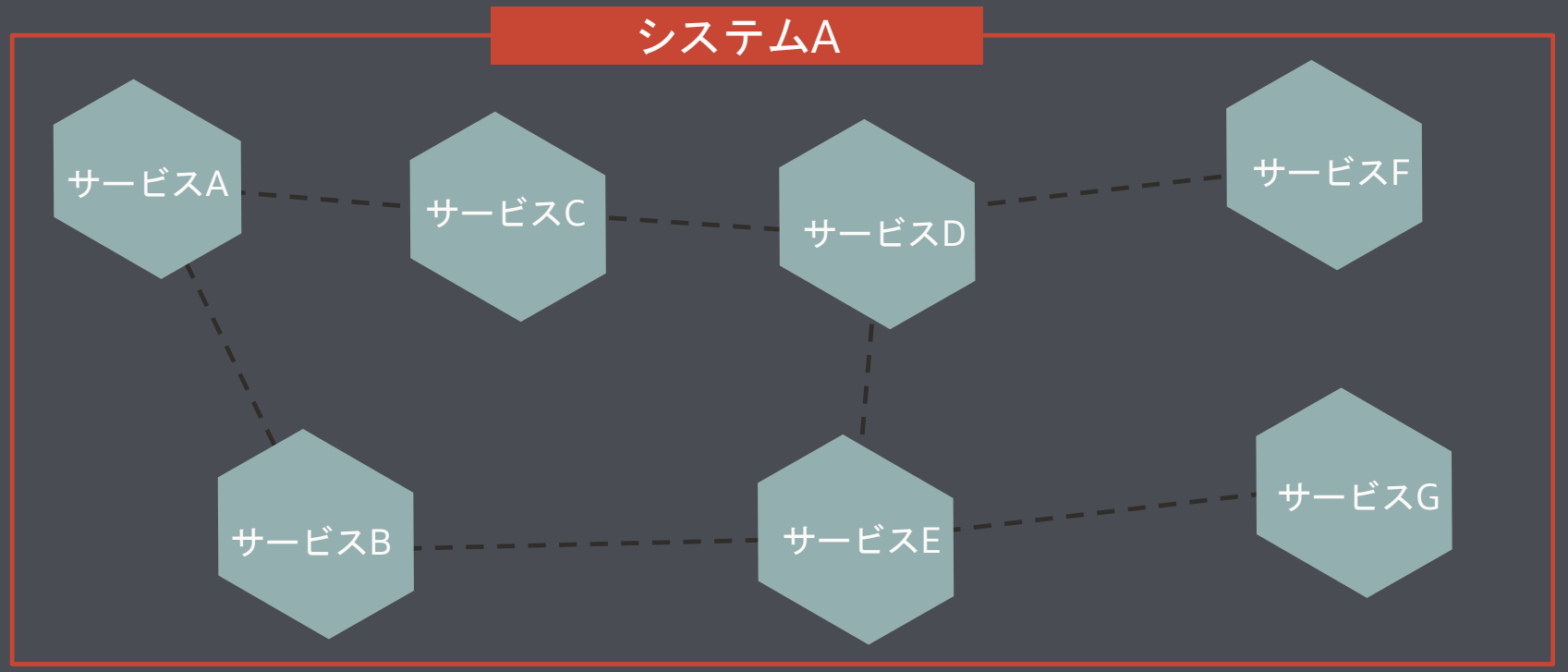


高頻度かつ即時に本番に反映可能なアプリケーションが必要

マイクロサービス・アーキテクチャとは

マイクロサービス・アーキテクチャ

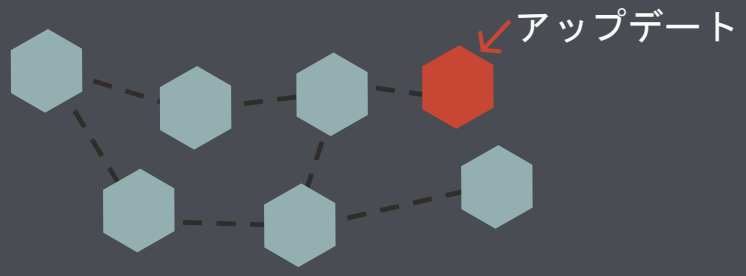
- 大規模なシステムを疎結合な複数のサービスの組み合わせで実現する設計方式
  - アップデートの容易性
  - スケールの容易性
  - 高可用性



マイクロサービス・アーキテクチャのメリット

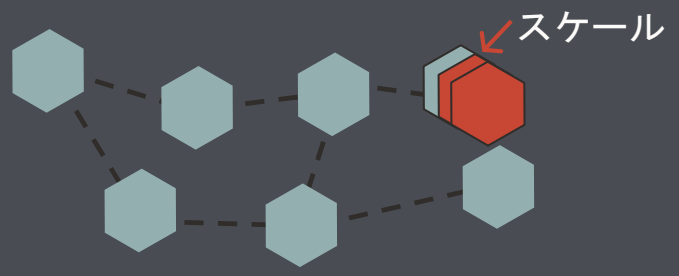
アップデートの容易性

- 他のサービスへの影響を極小化した形で、アップデート対象のサービスのみをアップデート可能
- 高頻度にサービスのアップデートが可能



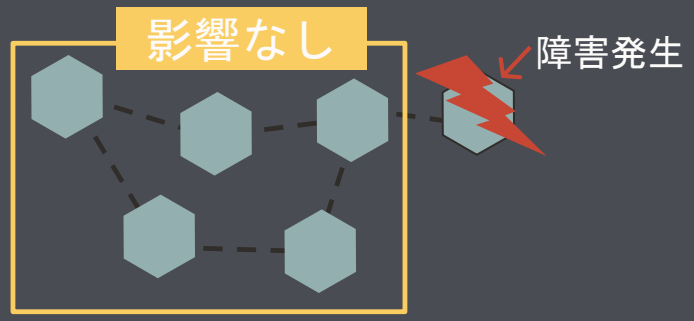
スケールの容易性

- 他のサービスに影響を与えることなく、必要なサービスのみをスケールアウト可能
- 使用するリソースを最適化



高可用性

- サービスが独立しているため、障害の影響を極小化
- 残ったサービス群でサービス提供を継続していくことが可能



マイクロサービス実装で発生する課題

メモリフットプリント

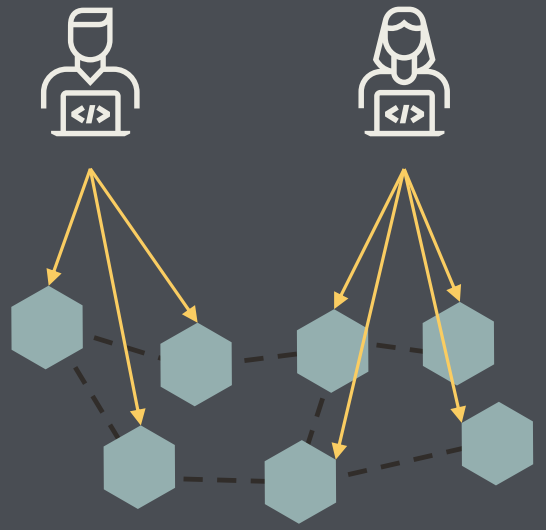
- 複数アプリケーションが動作することでメモリフットプリントが増大
  - パフォーマンスが劣化する可能性

アプリケーションフレームワーク

- 従来通りのフレームワークでは機能過多
  - パフォーマンスが劣化する可能性

アプリケーションライフサイクル

- 多数のサービスを一括で運用する必要性
  - 従来通りの手法では追従が困難



マイクロサービス実装時の課題に対する解決策

GraalVM + Helidon + OKEで構築する
マイクロサービスアプリケーション

マイクロサービス実装で発生する課題

メモリフットプリント

- 複数アプリケーションが動作することでメモリフットプリントが増大
  - パフォーマンスが劣化する可能性

アプリケーションフレームワーク

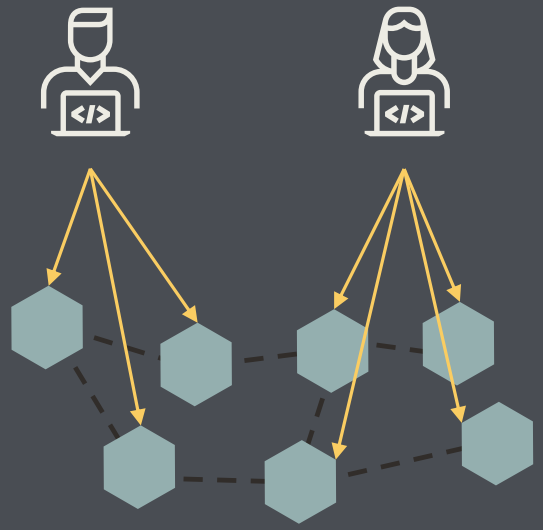
- 従来通りのフレームワークでは機能過多
  - パフォーマンスが劣化する可能性

spring

STRUTS

アプリケーションライフサイクル

- 多数のサービスを一括で運用する必要性
  - 従来通りの手法では追従が困難



解決策(メモリフットプリント)

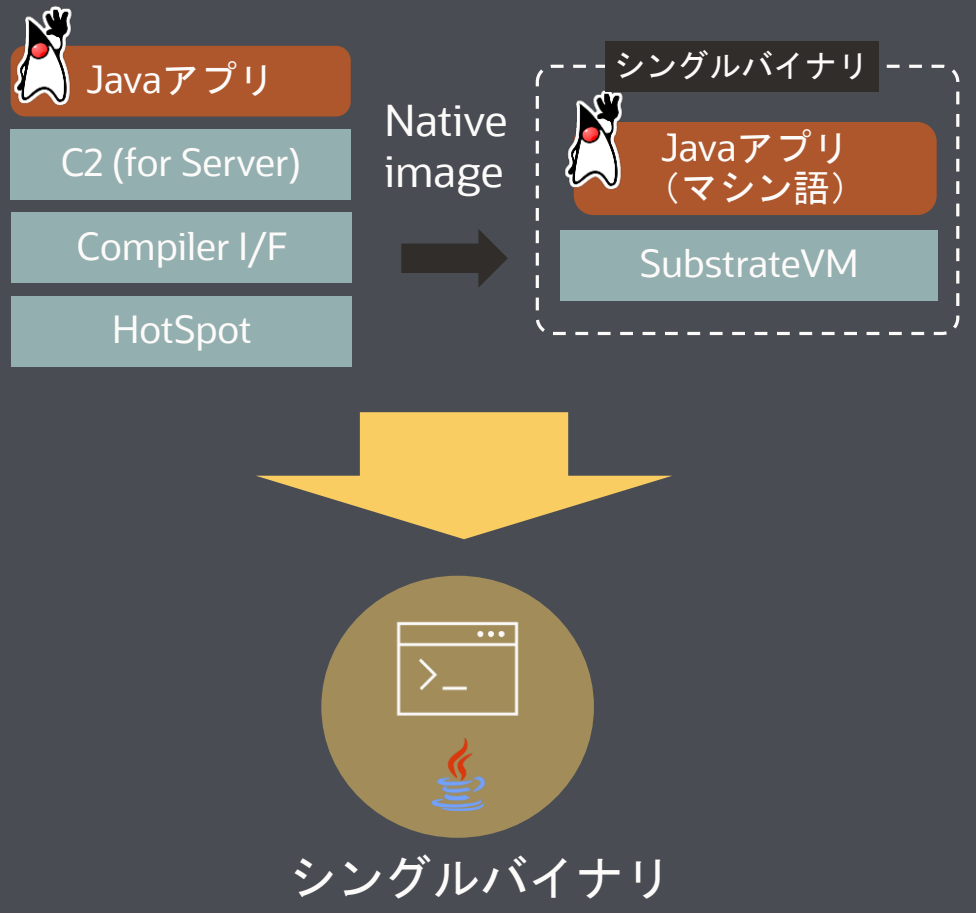
メモリフットプリントの軽量化(Native Image化)

Javaコード(JVMベースの言語)を事前コンパイルし、スタンドアローンで実行可能な形にコンパイルする

- 具体的には以下を含んでいる
  - 依存関係にあるアプリケーションクラス群
  - 実行時に利用するJDKクラス群(ランタイム環境)
  - 静的にリンクされたJDKのネイティブコード

・
・
・

- バイナリとして単体で実行可能
  - ランタイム起動時間の短縮
  - メモリフットプリントの極小化
  - セキュリティの向上(ランタイム環境の隔離)



解決策(アプリケーションフレームワーク)

軽量アプリケーションフレームワークの採用

- 小さなサービスを構築/デプロイするための軽量なフットプリント
  - 利用するリソースを極小化してリソースを最適化
- 複数言語が利用されるマイクロサービス環境下での相互運用性
  - Node.jsやPythonなどで実装された他サービスとも容易に連携可能
- RESTサービスなどのサービス間通信を効率的に実装可能
- CI/CDプロセスやツールでの扱いやすさ
- データ永続化対応における柔軟性
  - RDB/NoSQL/Kafkaなど

解決策(アプリケーションライフサイクル)

コンテナ

- 実行環境の隔離と可搬性の高さ(CI/CDの容易性)
  - Immutable Infrastructure
    - アプリケーションの実行環境をコンテナとしてパッケージ化
    - 同一環境を簡単に構築可能

オーケストレーション

- 大量のコンテナを一括で管理(コンテナオーケストレーション)
  - それぞれのコンテナに対するアップデート、障害対応、監視などを一括で実施可能
  - 現時点でのデファクトスタンダードはKubernetes

kubernetes

GraalVM/Helidon/OKEのご紹介

GraalVM + Helidon + OKEで構築する
マイクロサービスアプリケーション

マイクロサービス実装時の課題を解決するOracle Cloudソリューション

メモリフットプリント
- メモリフットプリントの軽量化

GraalVM™

アプリケーションフレームワーク
- 軽量アプリケーションフレームワークの採用


helidon.io

アプリケーションライフサイクル
- コンテナオーケストレーション(コンテナ)

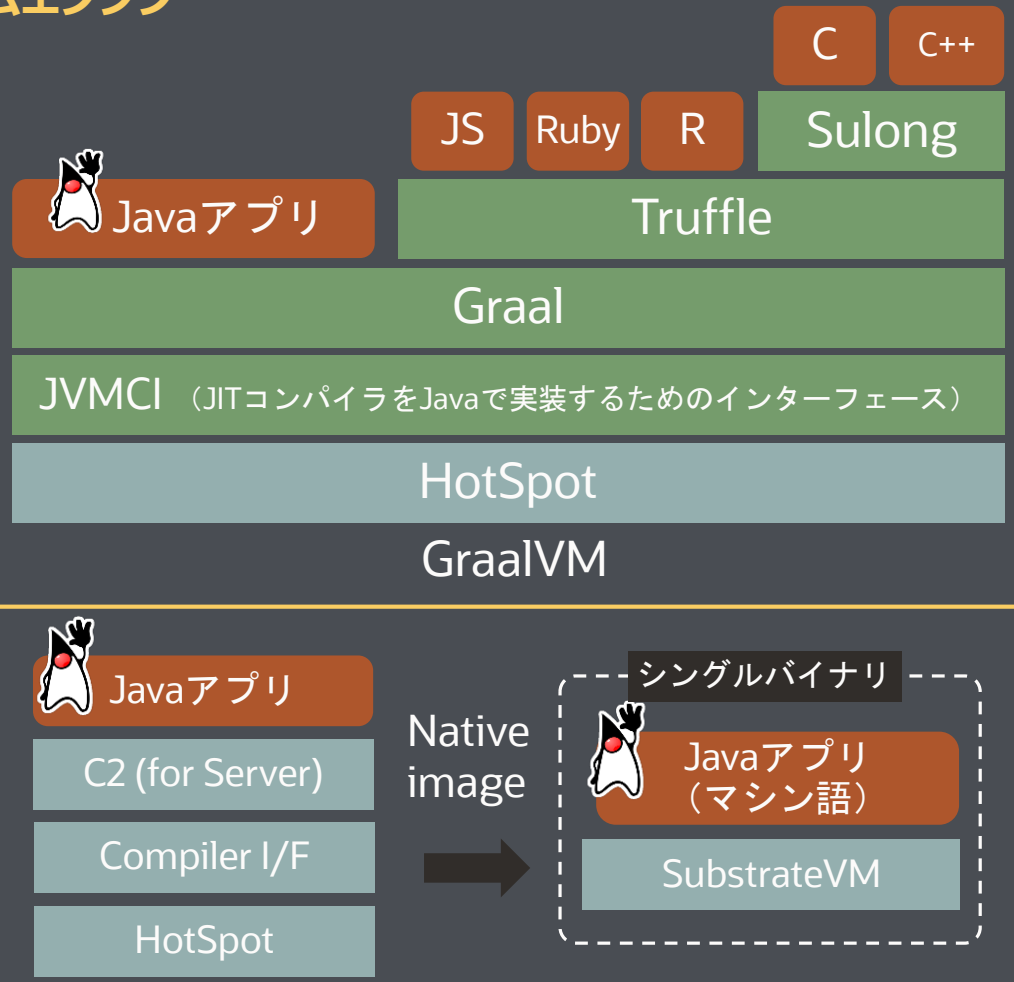

Oracle Container Engine for Kubernetes(OKE)

GraalVM

Oracle主導で開発されるOSSの多言語の単一かつ高速なランタイムエンジン

- Javaアプリを安定・高速実行
  - Javaで書かれた新たなJITコンパイラ
- JavaアプリのNative Image化
  - 事前にマシン語にコンパイル(Ahead-of-Time Compilation)
  - 起動時間の極小化とメモリフットプリントの削減
- 複数プログラミング言語を実行可能なランタイム
  - Java, JavaScript, Ruby, Rust, C, C++…
  - 単一プログラム内で、複数言語を絡めた記述が可能

- Community Edition(GPLv2)とEnterprise Edition
  - EEは最適化されたパフォーマンスとサポートを提供
  - EEはOracle Cloud Infrastructureで無償利用可能



Helidon

Javaの軽量アプリケーションフレームワーク

- オープン・ソース (github.com/oracle/helidon)
  - 最新版2.1.0(2020/12現在)
- マイクロサービスを開発するためのJavaライブラリの集合体
- JVM上の単体アプリケーションとして動作
- 従来からの一般的なツール・基盤で開発＆デプロイ可能
  - Java SE, Maven/Gradle, Docker, Kubernetes, etc.
- Java以外で開発されたサービスに対する相互運用性
- WebLogic Server/Coherence(EE/Grid)ユーザへのサポートの提供

helidon SE

- マイクロ・フレームワーク
- 超軽量フットプリント
- 関数型
- Reactive Web Server

フットプリント重視

helidon MP

- MicroProfile 3.2
- 軽量フットプリント
- 宣言型
- Java EEサブセット + マイクロサービス関連機能

機能性重視

Oracle Container Engine for Kubernetes(OKE)

エンタープライズ品質の性能と可用性

- 次世代インフラストラクチャによるばらつきのない高性能な分散アプリケーション環境
- アプリケーション環境と管理ノードの冗長化を自動的に構成し高可用性システムを実現

マネージド環境による開発への注力と費用対効果

- 複雑なKubernetes環境の構築を不要としアプリケーション開発を即座に開始
- 追加費用を一切不要とする標準機能としてのマネージドKubernetes環境

Cloud Native JavaソリューションやOracle Databaseとの親和性

- マイクロサービス化に対応するソリューションと共に活用可能
- 既存資産／スキルを活用したCloud Nativeアプローチによる開発

デモンストレーション

GraalVM + Helidon + OKEで構築する
マイクロサービスアプリケーション

デモの概要

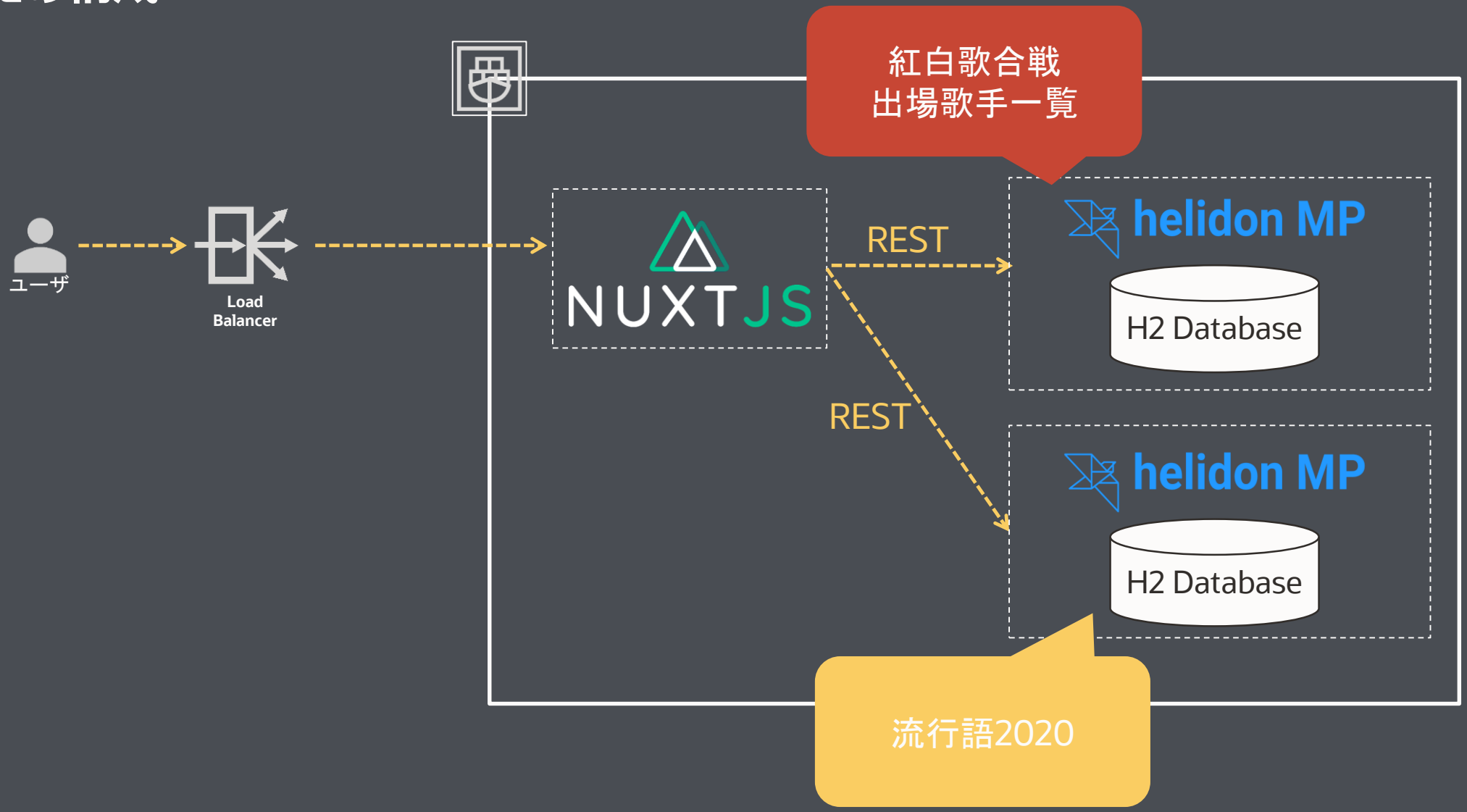
GraalVM/Helidon/OKEでマイクロサービスを動かしてみる

- フロントエンドアプリケーション: Node.js
  - フレームワークはNuxt.js
  - 今回はおまけでGraalVM(20.1.0)のnpm(node)バイナリでビルドして動作

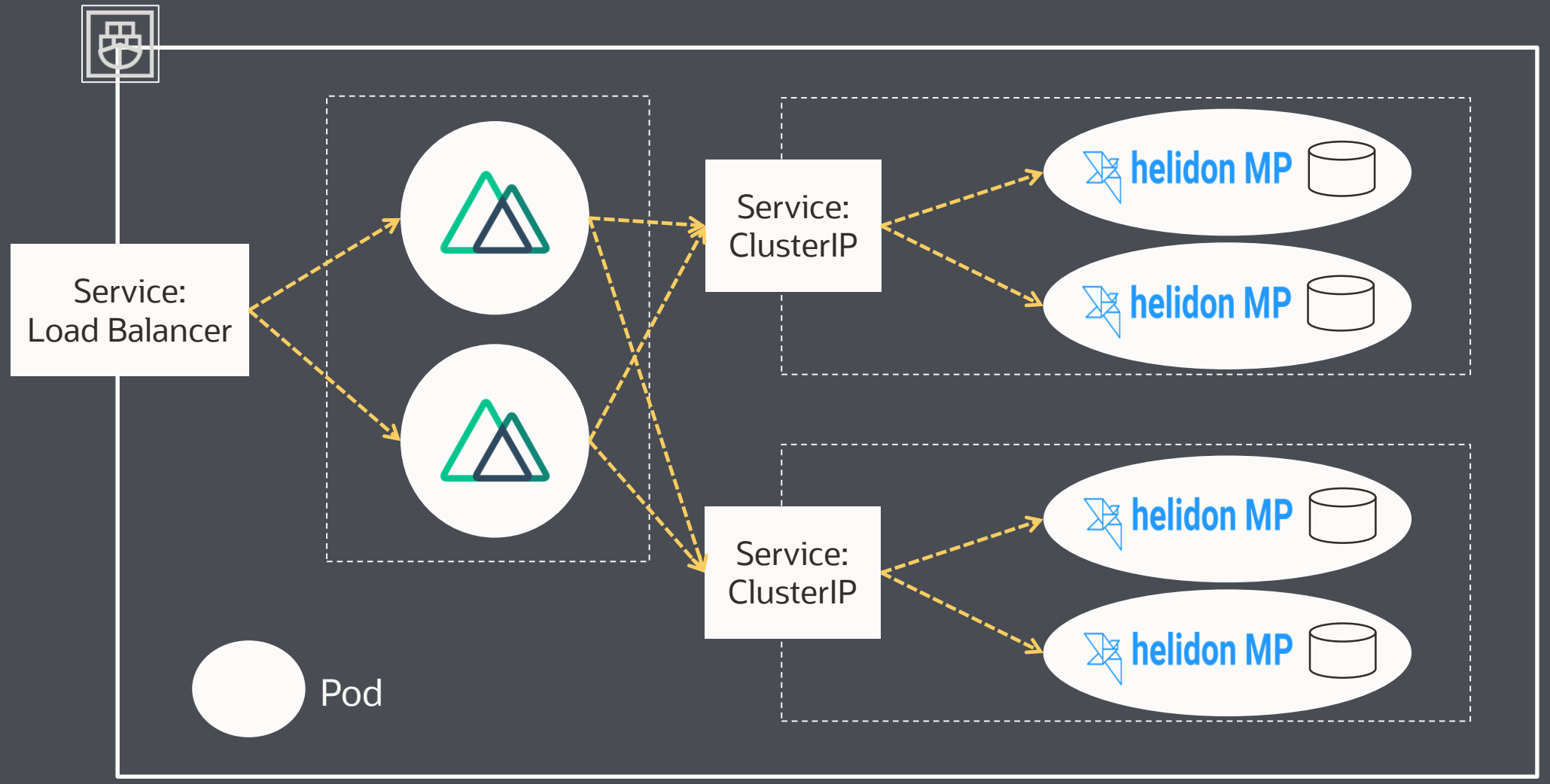
- バックエンドアプリケーション(Web API): Helidon MP(v2.1.0)
  - Native Imageとして稼働
  - データソースにはH2 Databaseを利用
  - H2 Databaseはサイドカーパターンで動作(詳細は後述)

- ランタイム環境: OKE(Oracle Container Engine for Kubernetes)
  - Kubernetesバージョン: v1.18.10

デモの構成



デモのアーキテクチャ(詳細)



まとめ

GraalVM + Helidon + OKEで構築する
マイクロサービスアプリケーション

まとめ

マイクロサービスの必要性

- 高頻度かつ即時に本番に反映可能なアプリケーションに最適
  - アップデートの容易性
  - スケールの容易性
  - 高可用性

マイクロサービス実装時の課題

- メモリフットプリントの増大
- 機能過多のアプリケーションフレームワーク
- アプリケーションライフサイクル運用の複雑さ

マイクロサービス実装時の課題を解決するソリューション

- GraalVM(Native Image)によるフットプリントの極小化
- 軽量アプリケーションフレームワークであるHelidonの利用
- OKEによるアプリケーションライフサイクルの最適化

GraalVM™

helidon.io

参考資料

Helidon : https://helidon.io/

GraalVM : https://www.graalvm.org/

Oracle Container Engine for Kubernetes(OKE) :
https://docs.cloud.oracle.com/ja-jp/iaas/Content/ContEng/Concepts/contengoverview.htm

OKEハンズオン :
https://oracle-japan.github.io/paasdocs/documents/containers/common/

本日のデモのソースコード :
https://github.com/oracle-japan/oracle-developer-days-2020-cloudnative-demo


ORACLE