ORACLE

OCI Cloud Native

Oracleが提供するクラウドネイティブサービスのご紹介

2024年8月

日本オラクル株式会社

アジェンダ

- クラウドネイティブの定義
- アプリケーションのモダン化とは
- OCI におけるクラウドネイティブ
  - コンテナ化
  - 開発自動化
  - 運用自動化
- Q&A

クラウドネイティブの定義

Cloud Native Definition

クラウドネイティブとは？

Cloud Native Computing Foundation（CNCF）におけるクラウドネイティブの定義

クラウドネイティブ技術は、パブリッククラウド、プライベートクラウド、ハイブリッドクラウドなどの近代的でダイナミックな環境において、スケーラブルなアプリケーションを構築および実行するための能力を組織にもたらします。このアプローチの代表例に、コンテナ、サービスメッシュ、マイクロサービス、イミュータブルインフラストラクチャ、および宣言型APIがあります。

これらの手法により、回復性、管理力、および可観測性のある疎結合システムが実現します。これらを堅牢な自動化と組み合わせることで、エンジニアはインパクトのある変更を最小限の労力で頻繁かつ予測どおりに行うことができます。

Cloud Native Computing Foundationは、オープンソースでベンダー中立プロジェクトのエコシステムを育成・維持して、このパラダイムの採用を促進したいと考えてます。私たちは最先端のパターンを民主化し、これらのイノベーションを誰もが利用できるようにします。

Cloud Native Computing Foundation(CNCF) Cloud Native Definition v1.0 | https://github.com/cncf/toc/blob/master/DEFINITION.md

“拡張性・柔軟性に優れたアプリケーションをクラウドの特性を活かし
最小限の労力で構築および実行できるようにすること”

Cloud Native Definition

結局のところ、クラウドネイティブとは？

人的関与を極力減らして価値の向上を目指して、
それをエンドユーザに届けるマインドセットを持つこと

Cloud Native Definition

クラウドネイティブを実現する道しるべ

コンテナ化

1. **CONTAINERZATION | コンテナ化**
   - 仮想マシンとは異なり、環境差異が無く、軽量なイメージで可搬性が高く、リリースサイクルを速め、**機敏さと信頼性を兼ね備えたアプリケーションリリースを実現**

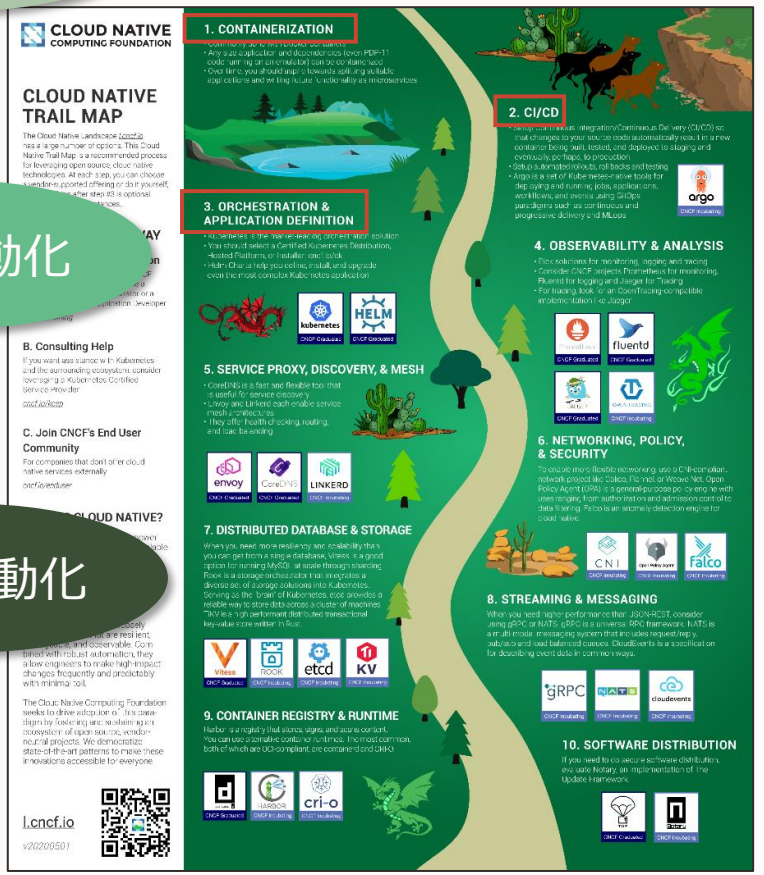
開発自動化

2. **CI/CD | 継続的インテグレーション/デリバリー**
   - ソースコードの変更を契機に、**自動的なビルド、テスト、そしてステージング環境および本番環境へのデプロイ**につながるCI/CD環境を構築する

運用自動化

3. **ORCHETRATION & APPLICATION DEFINITION | オーケストレーションツール、アプリケーション定義**
   - コンテナ化されたアプリケーションを**Kubernetes基盤で自律的に運用**
   - Helmを利用して、**マニフェストを効率的に管理**

拡張性・柔軟性に優れたアプリケーションを**最小限の労力**で構築/実行



CNCF Trail Map | https://github.com/cncf/trailmap

アプリケーションのモダン化とは

ガバメントクラウドにおける背景と目的：当社の理解

クラウドの特性を理解し最適化させることによりセキュリティ向上と全体的なコストメリットを享受

[背景と目的]「政府情報システムにおけるクラウドサービスの適切な利用に係る基本方針」より

参照元：「政府情報システムにおけるクラウドサービスの適切な利用に係る基本方針」[2022年12月28日版]

参照元：「政府情報システムにおけるクラウドサービスの適切な利用に係る基本方針」[2022年12月28日版] －「３．５ アプリケーションとシステム刷新について - 5)クラウド上で稼働するアプリケーションについて」より引用

ガバメントクラウドにおけるアプリケーションについて

クラウド上で稼働するアプリケーションの条件

#	項目	説明
1	**モダンアプリケーションとする**	クラウドならではの考え方とする 【例】マネージドサービスの組合せだけでシステムを構成、自らサーバを構築せずにシステムを構成する
2	**オンプレミス時代の旧来技術・運用を単純に踏襲しない**	現行踏襲せず、モダンな技術・運用で再設計を行う 【旧来技術・運用の例】：クライアントサーバ方式、専用端末のシンクライアント（VDI）、踏み台サーバ、閉域ネットワークのみに依存したセキュリティ対策、ビジネス要求やシステム価値につながらない監視ツール、メンテナンスを目的とした定期的なシステム（サービス）の停止、夜間に実施する必要のない夜間バッチ、オンプレミス用ミドルウェア等
3	**オンプレミス時代の人海戦術的な方式を踏襲せず自動化する**	クラウドの機能を活用して以下を行う ：インフラ環境構築の自動化（IaC）と CI/CD パイプライン化、インフラテストの自動化、システム監視や運用の自動化、セキュリティ監視の自動化
4	**単なるシステム監視ではなく定量的計測を行う**	クラウドの機能を用いた定量的計測により業務レベルでのサービス改善につながる監視/運用を行う
5	**セキュリティ対策もクラウドに最適化させる**	クラウドに最適化したセキュリティ対策を行う
6	**開発プロセスをクラウドに最適化させる**	クラウドの機能を活用しコスト効率の良い手法を選択する 【例】実機検証による効率化、アジャイル手法の採用、ドキュメントの自動生成 等
7	**稼働日で完成ではなく日々の運用で改善していく**	本番稼働後の継続的なサービス改善の実施 【例】クラウドから提供されるサービスのアップデート対応については、義務的な改修負担としてイベント的に捉えるのではなく、通常のアップデートと捉えて日常的に対応を行う

参照元：「政府情報システムにおけるクラウドサービスの適切な利用に係る基本方針」[2022年12月28日版] －「３．５ アプリケーションとシステム刷新について - 5)クラウド上で稼働するアプリケーションについて」より引用

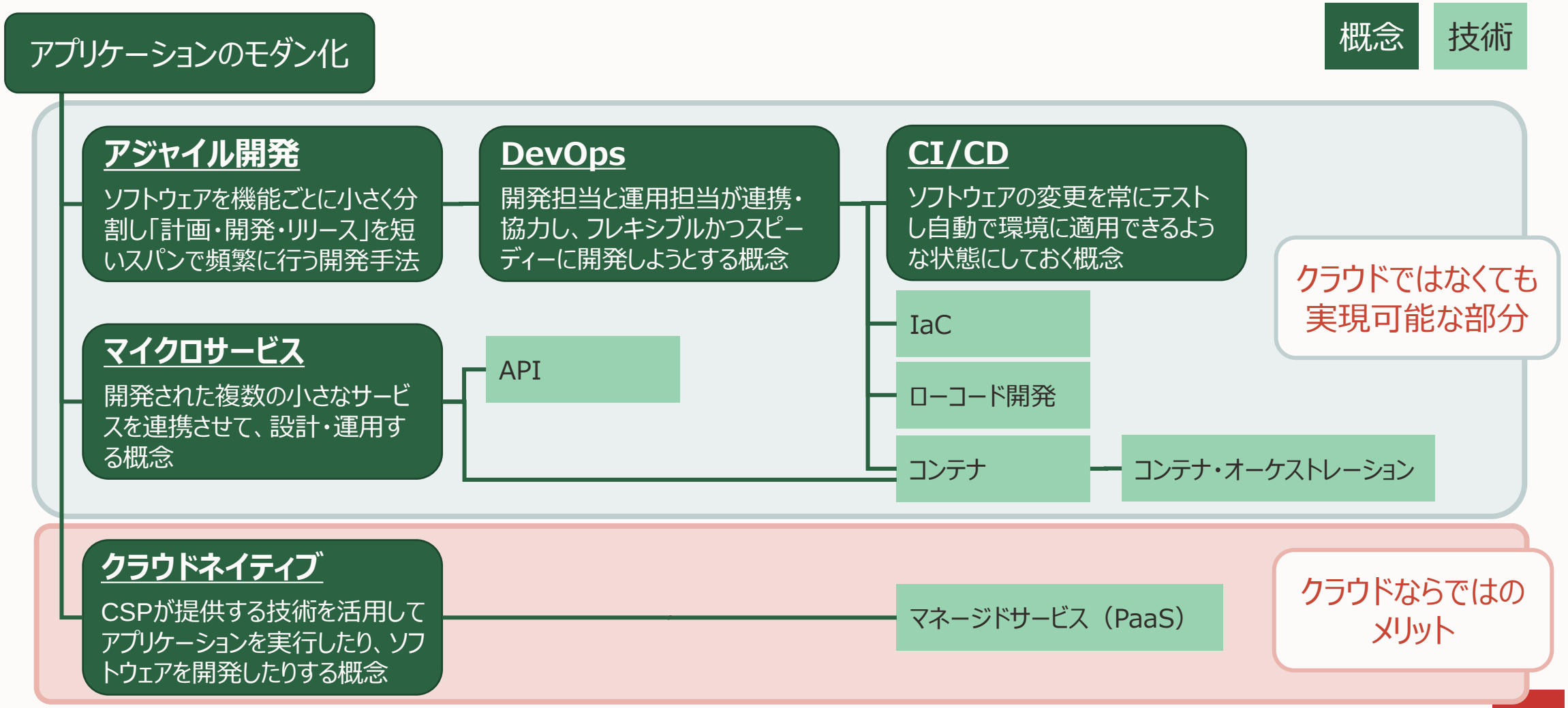
ガバメントクラウドにおけるアプリケーションについて

クラウド上で稼働するアプリケーションの条件

#	項目	説明
1	**モダンアプリケーションとする**	クラウドならではの考え方とする 【例】マネージドサービスの組合せだけでシステムを構成、自らサーバを構築せずにシステムを構成する
2	**オンプレミス時代の旧来技術・運用を単純に踏襲しない**	現行踏襲せず、モダンな技術・運用で再設計を行う 【旧来技術・運用の例】：クライアントサーバ方式、専用端末のシンクライアント（VDI）、踏み台サーバ、閉域ネットワークのみに依存したセキュリティ対策、ビジネス要求やシステム価値につながらない監視ツール、メンテナンスを目的とした定期的なシステム（サービス）の停止、夜間に実施する必要のない夜間バッチ、オンプレミス用ミドルウェア等
3	**オンプレミス時代の人海戦術的な方式を踏襲せず自動化する**	クラウドの機能を活用して以下を行う： インフラ環境構築の自動化（IaC）と CI/CD パイプライン化、インフラテストの自動化、システム監視や運用の自動化、セキュリティ監視の自動化
4	**単なるシステム監視ではなく定量的計測を行う**	クラウドの機能を用いた定量的計測により業務レベルでのサービス改善につながる監視/運用を行う
5	**セキュリティ対策もクラウドに最適化させる**	クラウドに最適化したセキュリティ対策を行う
6	**開発プロセスをクラウドに最適化させる**	クラウドの機能を活用しコスト効率の良い手法を選択する 【例】実機検証による効率化、アジャイル手法の採用、ドキュメントの自動生成 等
7	**稼働日で完成ではなく日々の運用で改善していく**	本番稼働後の継続的なサービス改善の実施 【例】クラウドから提供されるサービスのアップデート対応については、義務的な改修負担としてイベント的に捉えるのではなく、通常のアップデートと捉えて日常的に対応を行う

モダン化に関する用語

モダン化、クラウド化を取り巻く、様々な「概念」と「技術」

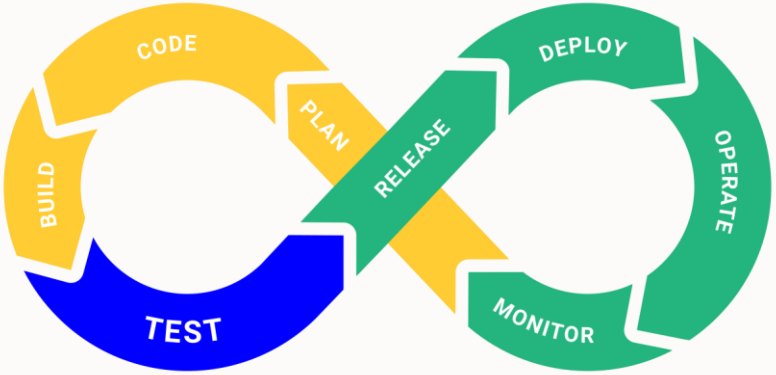


モダン化技術登場の背景

社会の変化の速さに対応するため、アプリケーション開発も効率化する必要が生じた

アプリケーション開発の効率化が求められる背景

- 変化の激しい社会情勢やニーズに対応する必要がある
- 企業価値の最大化を行い生き残っていくための変化（DX）が求められる
- 限りある人的リソースの中で生産性を高め効率化していく必要がある



素早い変化に対応するため、アプリケーション開発を効率化するアジャイル開発という開発手法

- 「計画、設計、実装、テスト」を機能単位に短サイクルで繰り返し開発を行う手法
- 開発期間を短縮することで、サービスインまでの時間を短縮できる
- ウォータフォール開発と異なり、仕様変更に強く、常に変化をさせていくビジネスや事業にマッチする考え方

短期間でのリリースを実現する手段として、開発と運用を一体として考えるDevOpsという考え

- 開発（Development）と運用（Operations）が互いに協力して開発・運用することで、アプリケーションの品質を高め、より早く、安全にリリースし、ビジネスの価値を高めていくという考え方
- そのために「**CI/CD**」という手法や 、「**コンテナ**」、「**コンテナ・オーケストレーション**」といった**テクノロジー**を活用

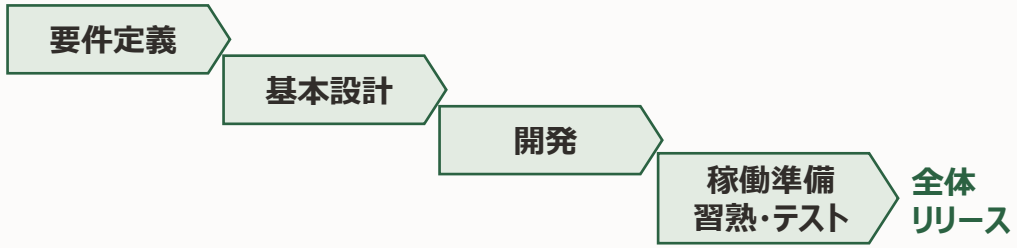
概念

2つの開発手法の整理　-ウォータフォール開発とアジャイル開発-

近年アジャイル開発が注目、ただし、それぞれにメリットとデメリットがある

ウォータフォール開発

机上でドキュメントをもとに合意形成を図りながら、各工程を滝のように上から下へと一方向に流す開発手法
（原則工程完了後に前工程への戻りは行わない）



メリット

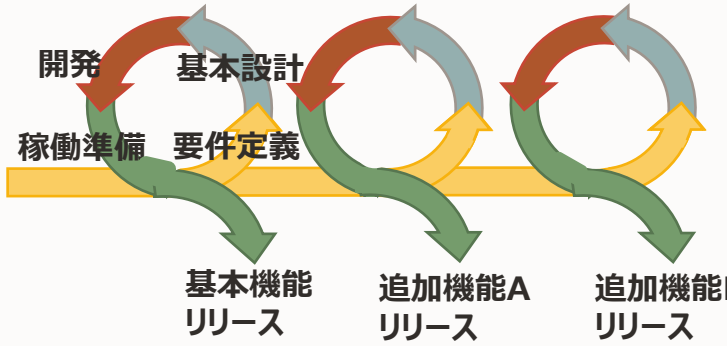
- 完成イメージに従って計画通りに開発を進めやすいためスケジュールや品質が期待した通りになりやすい
- スケジュール、工数、金額が見積もりやすい

デメリット

- ビジネスの事情で仕様変更が生じた場合、大きな手戻りが発生するため、ビジネス要件の変化が激しいシステムや、試しながら進めるDXの取り組みには向かない

アジャイル開発

システムを小さな単位に分けて、開発とテストを繰り返すことで、開発期間を短縮することができる開発手法。
頻繁にリリースを繰り返し、成果物を見ながらプロダクトをブラッシュアップしていく



メリット

- 小さな単位に分けることで開発期間を短縮可能
- 開発途中で成果物を見ながらプロダクトをブラッシュアップしていくことができるため、変化に対応しやすい

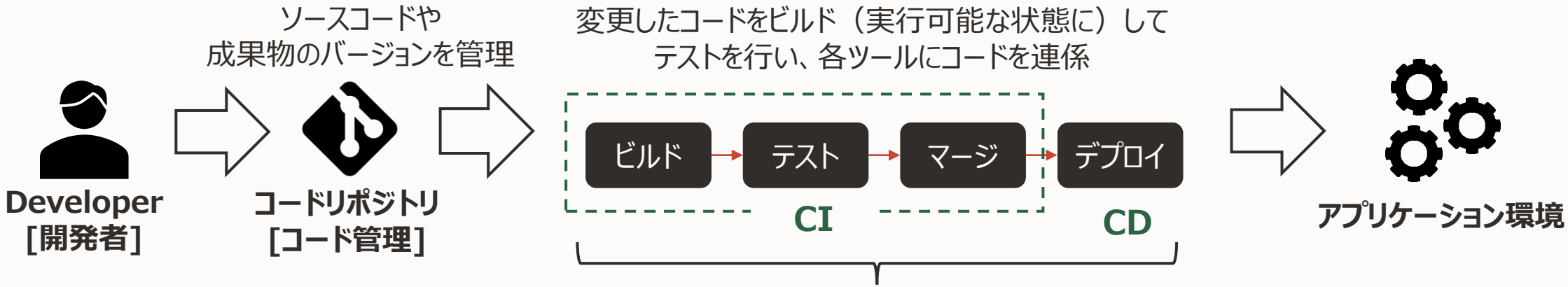
デメリット

- 都度要件を確認しなければならならないため、スケジュールや進捗具合がコントロールしにくくなる

概念

CI/CDとは

ソフトウェアの変更を常にテストし、できるかぎり高速に本番環境に適用できるような状態にしておく開発手法



CI (Continuous Integration)：継続的インテグレーション

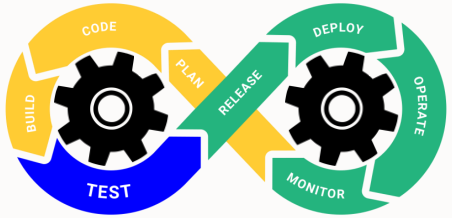
ソースコードの修正とテストを自動化し、反復的に行う手法。

早期にバグを発見、対処できることでアプリケーションの品質を高めることが可能

CD(Continuous Delivery)： 継続的デリバリー

CIによって生成された成果物をステージング環境などに自動的に配置(デプロイ)する手法。

リリース作業を簡素化し、安全、スピード、品質を向上させることが可能



CI/CD（CI/CDパイプライン）とは：ソフトウェアの変更を常にテストし、できるかぎり高速に本番環境に適用できるような状態にしておく開発手法

概念

CI/CDを活用するメリットとデメリット

CI/CDを活用するメリット

- 手作業でビルド・テスト・デプロイをする従来の手法に比べ、開発スピードを向上させ、生産性を向上
- アプリケーションリリースにおけるリスクを低減し、運用コストとリスクの削減につなげることができる
- 早く、正確に、より良いサービスを提供できることにより、アプリケーションの品質を向上し、職員様、国民利用者の満足度向上につなげることができる

CI/CDを活用するデメリット

- 自動化を可能にするための作業負担が発生する
- 継続的な自動化コードのメンテナンスが必要になる
- 多種多様で豊富なCI/CDツールの技術選定

CI/CDを有効に活用できるケース

- アジャイル開発のように、短期サイクルで機能ごとに開発とリリースを繰り返していくケース

（参考）CI/CDを実現する代表的なツール例

種類	機能	ツール例
コードリポジトリ	ソースコードや成果物のバージョンを管理	Jenkins Travis CI AWS Code Commit
CIツール	コードに変更があると、ビルドやテストに連携	GitHub/Git AWS Code Pipeline
構成管理	システム基盤の構成を管理	Ansible Puppet AWS Code Artifact
ビルド	ビルド（実行可能な状態にすること）を自動化	Ant Gradle Maven AWS Code Build
テスト	テストを自動化	UFT One Test Complete

OCIでは「DevOps」としてCI/CDのプラットフォームを提供
各機能を一元的に提供し、一気通貫の
開発・運用プロセスを実現

マイクロサービスとは

機能ごとに独立した小さなサービスを組み合わせて1つのシステムを構築する概念、ソフトウェアの開発手法

従来のシステム構成：モノリス（モノリシック）

すべての機能を一つにまとめたシステム構成

モノリスのメリット：

- **シンプル**：コードやデータが一か所にまとまっているため、規模が大きくなっても比較的管理しやすい
- **高速**：マイクロサービスのようにネットワークを介さないため、ネットワーク負荷が低い

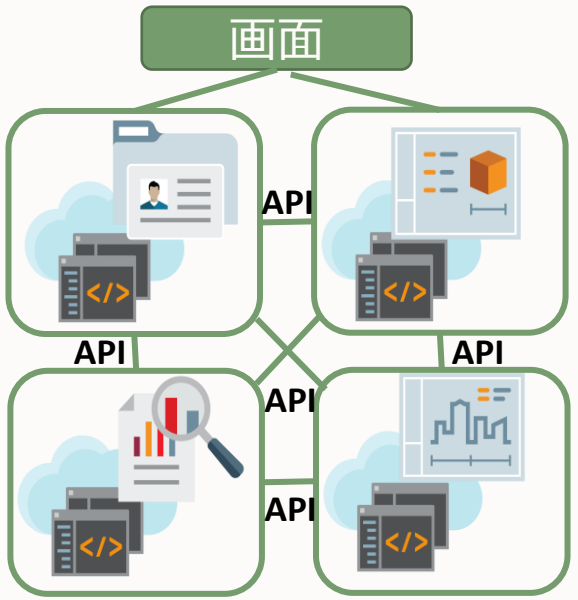
モノリスのデメリット

- **柔軟性に欠ける**：アプリケーションの拡張、変更や管理などの面で柔軟性が低い。一部を変更するためにも全体に影響が及ぶ可能性

対義

近年注目されている構成：マイクロサービス

単体で動作する機能を組み合わせたシステム構成、
またはシステムをそのように作ること



独立したプロセスで開発、運用される各サービスはAPIなどのリモート通信で連携する

マイクロサービスのメリット：

- **開発スピード**：開発単位が小さいため開発スピードが向上
- **柔軟**：機能の追加・変更がしやすい。一つのシステムの中で、各機能に適した異なる技術を使用できる
- **可用性**：障害や問題の範囲を限定的にし、リスクを軽減

マイクロサービスのデメリット

- **複雑**：システム全体の設計が複雑化し、難易度が高まる。機能間でデータの一貫性を維持しにくくなる

マイクロサービスとモノリスのトレードオフ

どちらが良いというものではなく、業務・システムの特性や目指すべき姿に合わせて最適な構造を検討すべき

	モノリス(従来)		マイクロサービス	
	変更の影響範囲が把握しにくいため、頻繁に更新できない	更新頻度	影響範囲がサービスにとどまるため、高頻度で更新できる	
	障害の影響がシステム全体に及びやすい	耐障害性	障害の影響範囲をサービスに留めるように設計できる	
	通信のオーバーヘッドが無いため、従来通りのパフォーマンスが出やすい	パフォーマンス	サービス間通信のオーバーヘッドのため、パフォーマンスが落ちやすい	
	基本的にはデータベースがひとつなのでデータの一貫性を保ち易い	データの一貫性	サービスをまたぐトランザクションは実現困難	
	管理対象が少ないための、運用・管理が比較的シンプル	運用・管理性	サービスが増える分だけ管理対象が増加	

どちらにも得手、不得手があるため、システム特性をふまえた見極めが必要

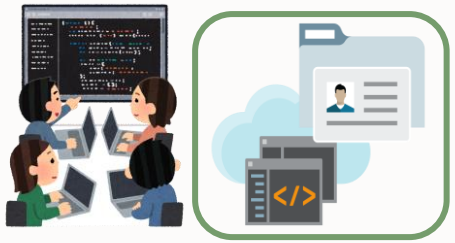
マイクロサービスを検討する上でのもう一つの要素「組織」

マイクロサービスを効率的に活用するには、「最適な組織体制」や「責任の持ち方」を検討する

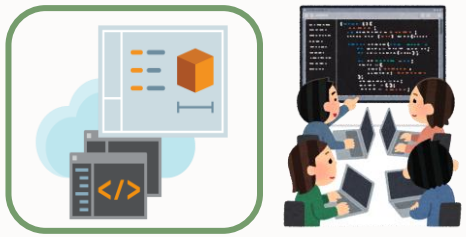
マイクロサービスでの開発に適した組織とは

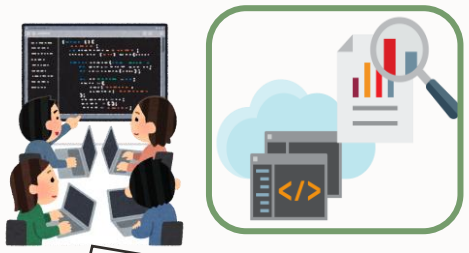
マイクロサービスで開発を進めるにあたっては、**「チーム」**に分けて開発を進めていくことが望ましい

担当機能に集中して開発することで効率化



CI/CDのサイクルを早めることができる





全体規模が大きくなっても柔軟性やアジリティを維持

New

新しいプロジェクトも少人数で始めやすい

ただし…

単に組織を細分化するだけでは、生産性の向上や期待するような迅速かつ高品質な開発にはつながらない

- 連係しあう仕組みの場合チーム間の連携が必要
- 異なるチーム間の方針や方式を取りまとめる必要
- ある領域の変更が他に及ぼす影響の確認
- セキュリティや可用性レベルの統一

どのような単位でマイクロサービスに分ければいいのだろうか

複数のグループやベンダーをどのように統括し、まとめていけばいいのだろうか

モノリスであれば、1社に依頼すれば良かったのに・・・

コンテナとは
マイクロサービスに最適なアプリケーション動作基盤

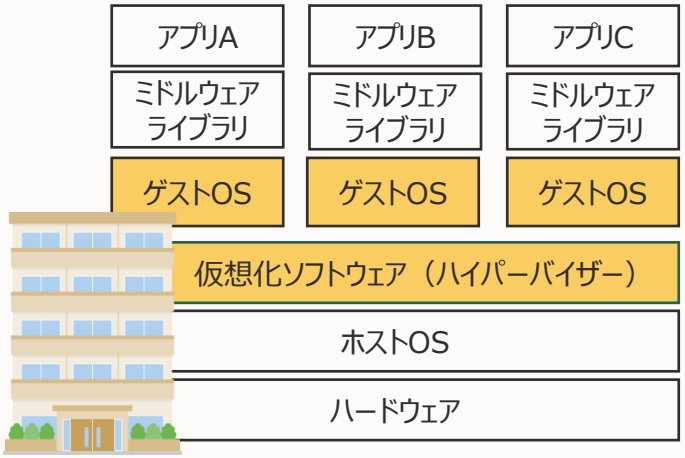
コンテナとは：アプリケーションやWebの開発・管理を効率的に行えるようにするOSレベルの仮想化技術
従来の構築法に比べて軽量であり、構築、移行がしやすいため、マイクロサービスに向く仮想化基盤と考えられる

ITインフラを個別のハードウェア上に構築

アプリ毎に物理サーバを分けて構築する。個別構築、アプリごとに余裕を見たインフラ調達、個別管理が必要であるため、ITインフラコストの高止まり、運用管理の煩雑化が課題となりやすい。

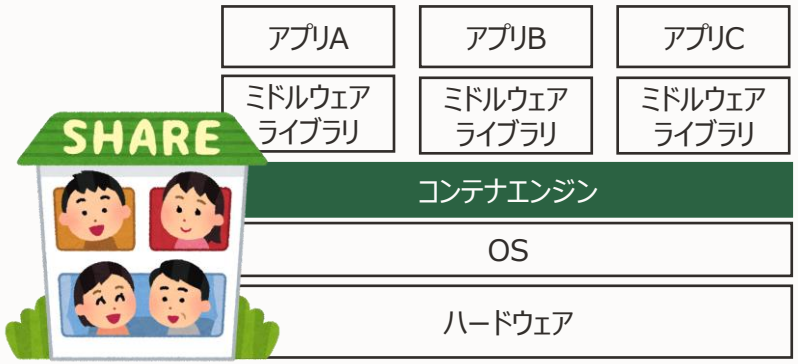
サーバ仮想化（VMWareなど）

物理サーバ上に仮想化ソフトウェアを入れ、その上にアプリケーション用の個別のOSを構築する。少数のハードウェアで複数の実行環境を構築できるが、仮想サーバごとにOSが動くため容量が大きく、OSごとの管理も必要。



コンテナ化：OS仮想化

OS上に「コンテナエンジン」を配置し、コンテナ上で複数のアプリ実行環境を構築する。OS等の共有できるリソースは共通で利用することでリソースを節約し、軽量に動作させることが可能。

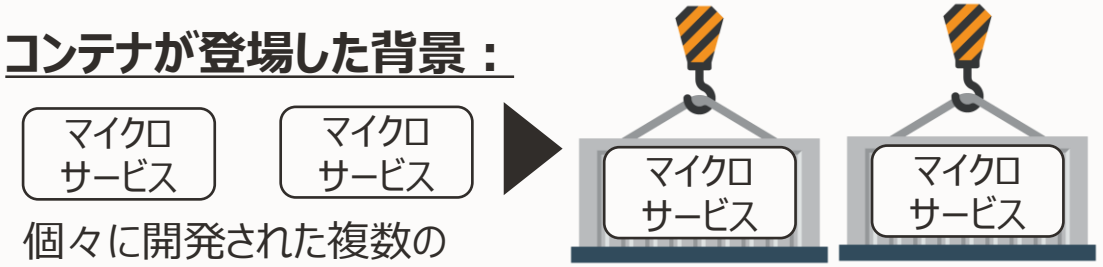


技術

コンテナの特徴と、利用するメリットとデメリット

マイクロサービスに最適なアプリケーション動作基盤

コンテナが登場した背景：



個々に開発された複数の小さなサービスを連携させて、設計・運用する概念

小さなアプリケーションを動かすための軽量な基盤

コンテナの特徴：

軽量：ホストOSのカーネルを共有することで、CPU、メモリ等のリソースを節約できる

手軽：アプリケーションの実行に必要なもの（ミドルウェアやOSライブラリ等）があらかじめコンテナ内にパッケージ化されているため、どのような環境でもすぐに実行可能

可搬性：コンテナエンジンがあればどの環境でも動作するため、別の環境に移行しやすい

※マイクロサービスを動かすうえでコンテナが必須なのではなく、マイクロサービスに向いている基盤

コンテナを利用するメリット：

開発者にとって
- 環境構築の工数削減等による作業効率向上
- 少ないリソースで開発環境を用意できる

運用者にとって
- OSの管理が不要なため管理工数を削減できる
- 柔軟/迅速なスケーリングでコストを最適化できる

コンテナを利用するデメリット：

開発者にとって
- 複数のOSを使った運用はできずOS依存が発生
- コンテナに関する専門知識が必要

運用者にとって
- 運用が複雑化しやすく、トラブル対応の難易度が高い

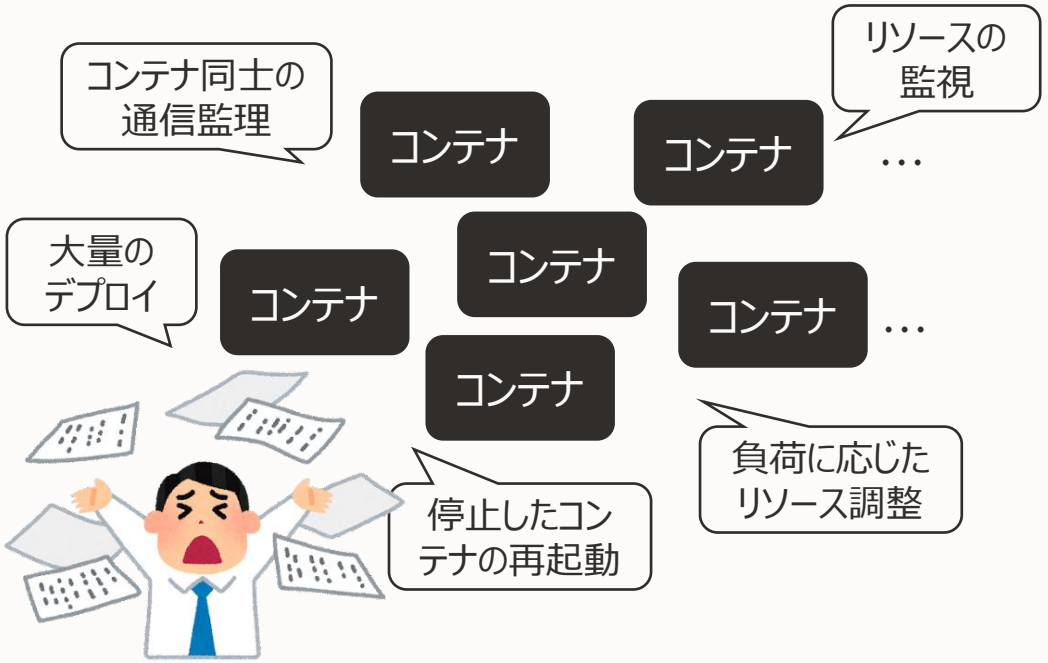
技術

コンテナの運用を効率化する「コンテナ・オーケストレーション」

コンテナの運用における課題と、コンテナ・オーケストレーションの活用

実際にコンテナを運用しようとすると…

大量のコンテナに対してオペレーションを行う
必要があり、手動で行うのは現実的ではない



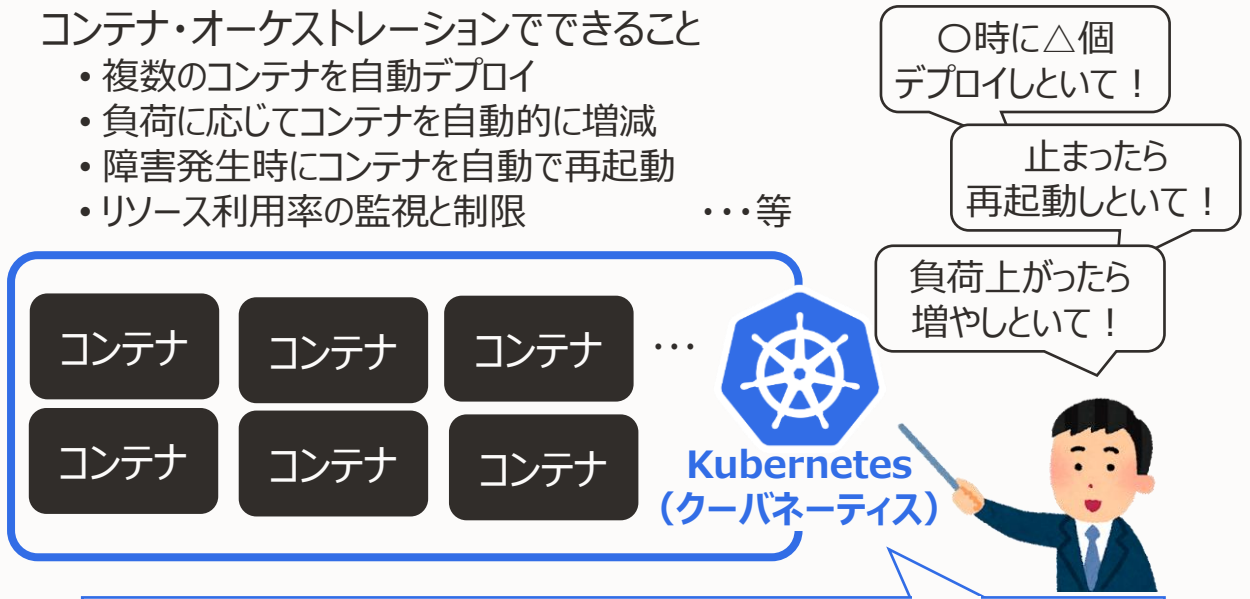
効率化

コンテナ・オーケストレーションの活用

コンテナを一括して管理・運用する、コンテナに関わる調整と
管理を自動化する技術

コンテナ・オーケストレーションでできること
- 複数のコンテナを自動デプロイ
- 負荷に応じてコンテナを自動的に増減
- 障害発生時にコンテナを自動で再起動
- リソース利用率の監視と制限

…等



コンテナ・オーケストレーションのデファクトスタンダード・ツール
各CSPがKubernetesを利用できるマネージドサービスを提供

技術

インフラ環境構築の自動化（IaC）とは

インフラの構成情報を“コード“として管理し、インフラ環境構築を自動化することができるテクノロジー

従来のインフラ環境構築

調達したサーバやクラウド環境に対して、
構築手順書をもとに人が様々な設定作業などを実施

各種設定

起こりえる問題

- 人的作業ミス
- 構築に毎回工数がかかる
- 更新漏れによる実環境との差異

コンテナ

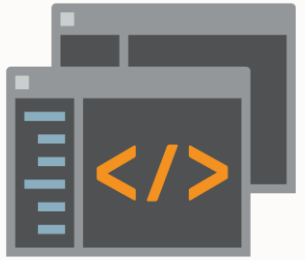
コンテナ

マイクロサービスやコンテナの活用により、構築頻度や管理対象は増加しさらに煩雑に…

自動化

Infrastructure as Code（IaC）の活用

インフラの構成情報を“コード“として管理し、インフラ環境構築を自動化することができる技術
設定情報等をコードで記載し保存しておき必要な時にコピーして実行することが可能



代表的なIaCツール
Terraform、Docker、Ansible、Chef　等

IaCを活用するメリット

- 人的ミスの低減による品質向上
- 環境構築に係るコストを低減可能
- サービスの拡充や迅速な提供が可能
- 正確で一貫性のある環境構築が可能
- 保守性の向上

IaCを活用する上での考慮点

- スキル/ナレッジが必要
- IaCを利用しても全てが自動化できるわけではない
- IaCのセキュリティ問題が環境全体の問題に発展する可能性

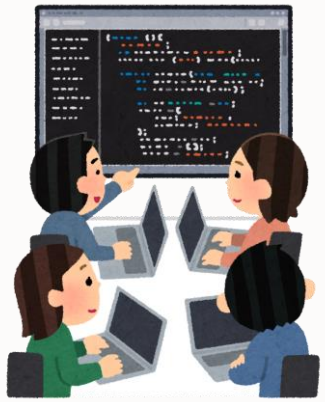
目的とスキル/時間等により見合った効果があるかは要検討

ローコード/ノーコード開発とは

可能なかぎりコードを書かずに、アプリケーション作成を自動化することで開発生産性を向上

スクラッチ開発

開発言語でコードをゼロから書き起こして、アプリケーションやシステムの開発を行う
高度な知識や技術が求められるため、一般的に外注を行うケースがほとんど



開発に時間とコストがかかる

一度作り始めたら変更が難しい

変化の速い時代のDX対応には向かない

ただし、自由度はスクラッチ開発が最も高く、ローコードやノーコードでは細かいデザインや機能の要望に応えられない可能性がある

自由度：スクラッチ開発 ＞ ローコード開発 ＞ ノーコード開発

対義

ローコード開発

可能な限りコードを書かずにアプリケーションを開発する手法。
アレンジしたい部分だけコードで作りこむことも可能。

- ツール提供ベンダーの提供するGUIツールを活用して視覚的な操作（ドラッグアンドドロップ等）でアプリケーションを作成できるため、深い専門知識がなくても開発が可能
- ツール提供ベンダーの用意する画面部品やロジック部品を組み合わせてアプリケーションを作成するため、システムの開発効率を上げ、コストを抑えることが可能（アジャイル開発にも最適）
- テンプレートや部品をもとにするため、画面イメージを統一し、分かりやすいインタフェースにすることが可能。また、担当者が変わっても引き継ぎやすい

ノーコード開発

プログラミングに関する専門知識不要で開発を行うことができ、ローコードよりも敷居が低い（誰でも開発しやすい）
一方、ローコードよりさらに自由度は低い

OCIにおけるクラウドネイティブ

クラウドネイティブ開発における3つの課題

ベンダーロックイン

- クローズドなAPIやサービス
- 可搬性のないソリューションでロックインされる可能性

OSSの乱立・DIYの限界

- セットアップに高度なノウハウが必要
- スキルを持った人員の不足
- パフォーマンス、セキュリティ、拡張性、信頼性の確保が困難

ツールの不足による追加コスト

- 不足するツールは独自に導入
- 管理が複雑化し維持コストが増加

クラウドネイティブ開発の課題へのOracleのアプローチ

Managed

マネージドサービス

- マネージドなサービスを
  エンタープライズグレードな
  OCI上でご提供
- DIY不要

Inclusive

開発から運用まで
End to End のサービス

- コンテナベースのランタイムから
  運用管理まで必要なサービスを
  全てご提供

Open

コミュニティ主導・オープン

- プロプライエタリなテクノロジーや
  利己的なOSS拡張には基づか
  ない

OCIでご提供しているCloud Native サービス

これまでご紹介してきた技術をはじめ、モダン・アプリケーションの開発・運用を支援するサービスを広く展開

OCI Cloud Native Landscape

アプリケーション実行基盤

- Container Engine for Kubernetes (Managed/Virtual Nodes)
- Container Instances
- Oracle Functions
- WebLogic Server for OCI/OKE

API 管理

- API Gateway

メッセージング

- Queue
- Streaming

サービス間連携

- Notifications
- Service Connector Hub
- Events

オブザーバビリティ

- Monitoring
- Logging
- Application Performance Monitoring

開発フレームワーク

- Helidon MP/SE
- Micronaut

開発・運用管理ツール

- OCI DevOps
- OCI Registry
- Resource Manager
- OCI Service Mesh
- OCI Workflow*
- MicroTx (Free版)

OCI Core Infrastructure

Compute | Database | Storage | Networking

*: Coming Soon!

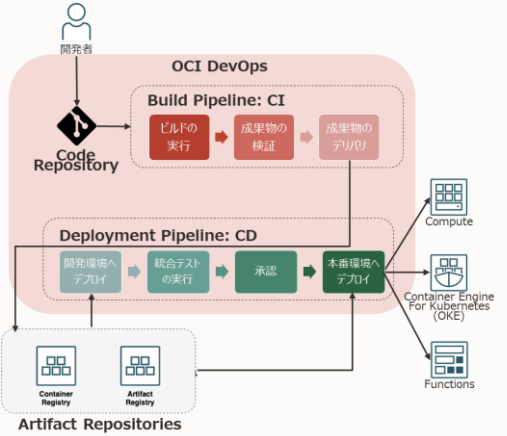
今回ご紹介した技術に対応するOCI Cloud Native サービス例

ご紹介した内容	OCIでご提供するサービス
DevOps	OCI DevOps Visual Builder Studio
CI/CD	
コンテナ	Container Instances
コンテナ・ オーケストレーション	Container Engine for Kubernetes WebLogic Server for OKE
IaC	Resource Manager
ローコード開発	Oracle Application Express

OCI DevOps

コード管理、ビルド・パイプライン、アーティファクト管理、デプロイメント・パイプラインから構成される各機能を用いた、一気通貫の開発・運用プロセスを実現

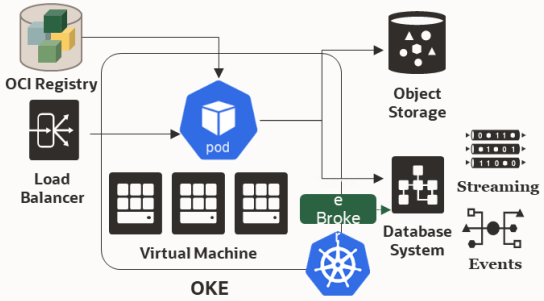
■価格：無料 (コード管理のストレージは別途)



Container Engine for Kubernetes (OKE)

コンテナを活用して、高頻度なリリースを掛けてゆくモダンアプリケーションの開発に最適

■価格：利用するリソース（ストレージ等）の料金のみ



Resource Manager

業界標準のTerraformベースで定義を記述するチーム開発に適したマネージドなIaC

■価格：無料

OCI Cloud Native サービスの強み

他社と比べたOCIサービスの強み

性能と価格

- OCI上で動作しているため、OCIの性能や価格面のメリットを享受可能
- 多くのサービスが無償、もしくはストレージとサーバ（Compute）のリソース料金のみ
- Oracle FunctionsはAWSより15%安価で無償枠も２倍

OSSをはじめオープンな技術を活用したサービス

- コンテナベースの技術では、OSSをもとに、可搬性のある技術を提供（他社クラウドやオンプレミスでも動作可能）
- 最新の技術をOSSにフィードバックしたり、他社クラウドに対しても提供
  - マイクロサービス開発向け軽量Javaアプリケーションフレームワーク
  - MySQL HeatWave…等

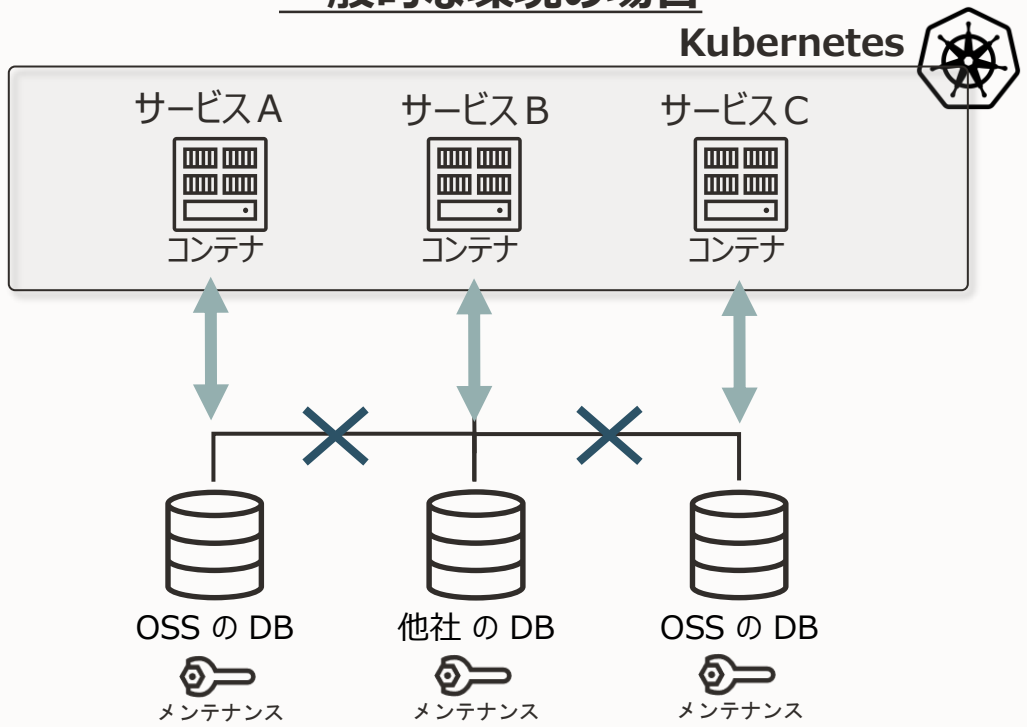
データ管理分野の強み

- マイクロサービスにおいてよくあるデータの分散、連携、管理に関する課題を解決
  - 様々なタイプのデータを管理できる統合型データベース
  - データベースのコンテナ化
  - 分散トランザクション管理

エンタープライズシステムでCloud Native型開発に取り組むのに最適なデータ管理

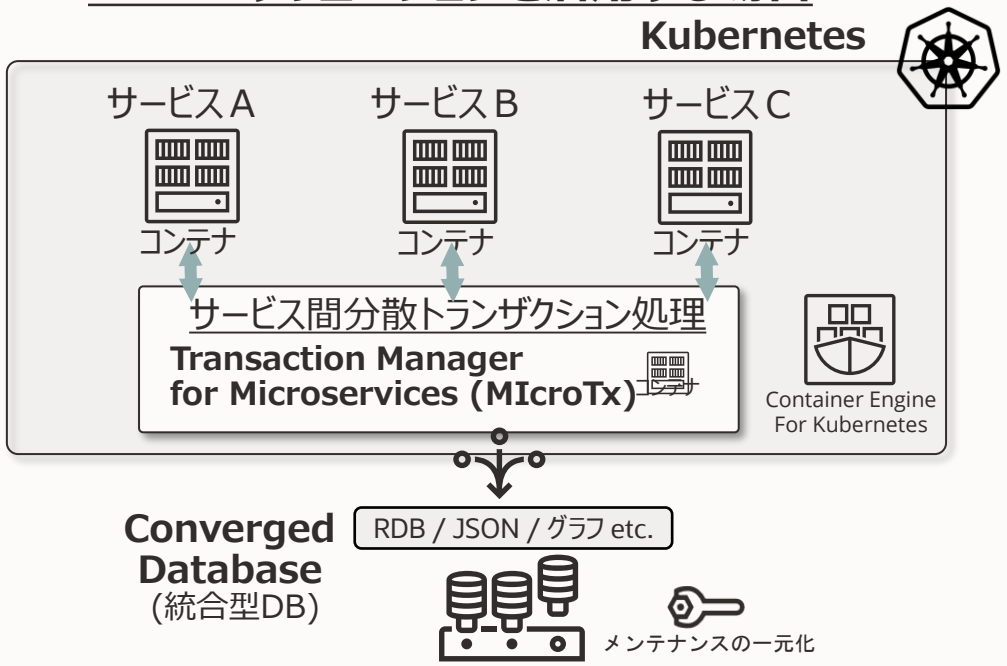
業界標準のオープンなテクノロジー + データマネジメント分野でのオラクルソリューションの強み

一般的な環境の場合



- サービス間分散トランザクション処理は現実的に不可
- サービスごとの個々のDBのメンテナンスが必要（どこにどのデータを置くかの設計、管理、データの重複等）

Oracleソリューションを活用する場合



- サービス間分散トランザクション処理が可能
- 運用やポリシーを一元管理しつつ、コンテナ（PDB）でマイクロサービスごとにデータを分離
- RDB/JSON/グラフなど様々なタイプのDBを利用可能

コンテナ化

コンテナ化に関するサービス

OCI Cloud Native Landscape

アプリケーション実行基盤

- Container Engine for Kubernetes (Managed/Virtual Nodes)
- Container Instances
- Oracle Functions
- WebLogic Server for OCI/OKE

API 管理

- API Gateway

メッセージング

- Queue
- Streaming

サービス間連携

- Notifications
- Service Connector Hub
- Events

オブザーバビリティ

- Monitoring
- Logging
- Application Performance Monitoring

開発フレームワーク

- Helidon MP/SE
- Micronaut

開発・運用管理ツール

- OCI DevOps
- OCI Registry
- Resource Manager
- OCI Service Mesh
- OCI Workflow*
- MicroTx (Free版)

OCI Core Infrastructure

Compute | Database | Storage | Networking

*: Coming Soon!

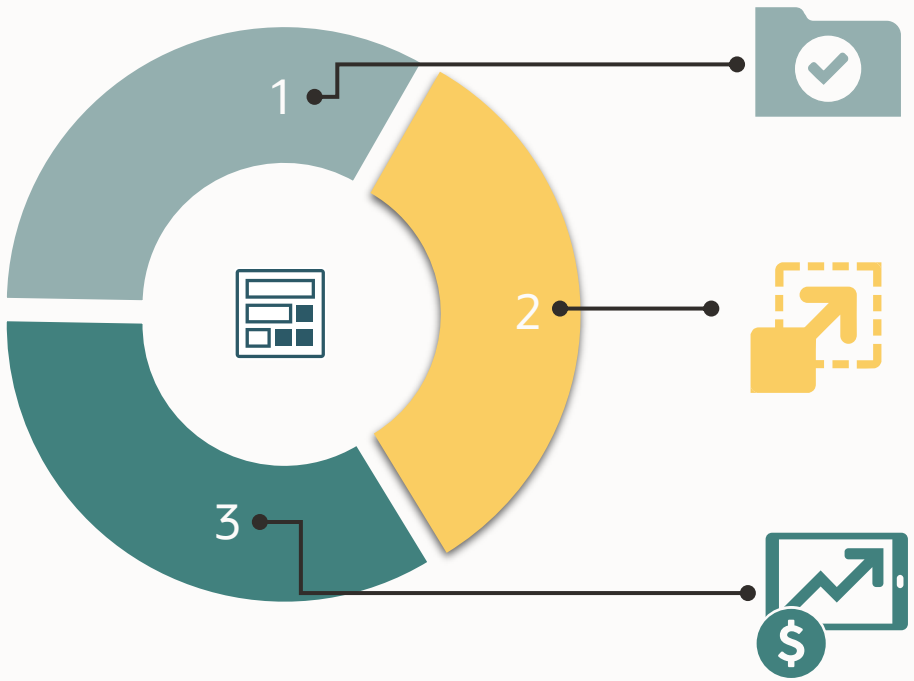
OCI Functions

拡張性・カスタマイズ性の高いオープンソース・ベースのフルマネージドFaaS

オープンソースのFaaSエンジン
“Fn Project”を採用

コンテナ技術をベースとした
高いカスタマイズ性

Oracle Cloud Infrastructure
が提供する
豊富なサービス群との連携



Pay per use

- 実稼働分のみ料金が発生
- アイドルタイムには課金が発生しない

Autonomous

- リクエスト量に応じた自動スケール
- プロビジョニングや管理のためのサーバは不要

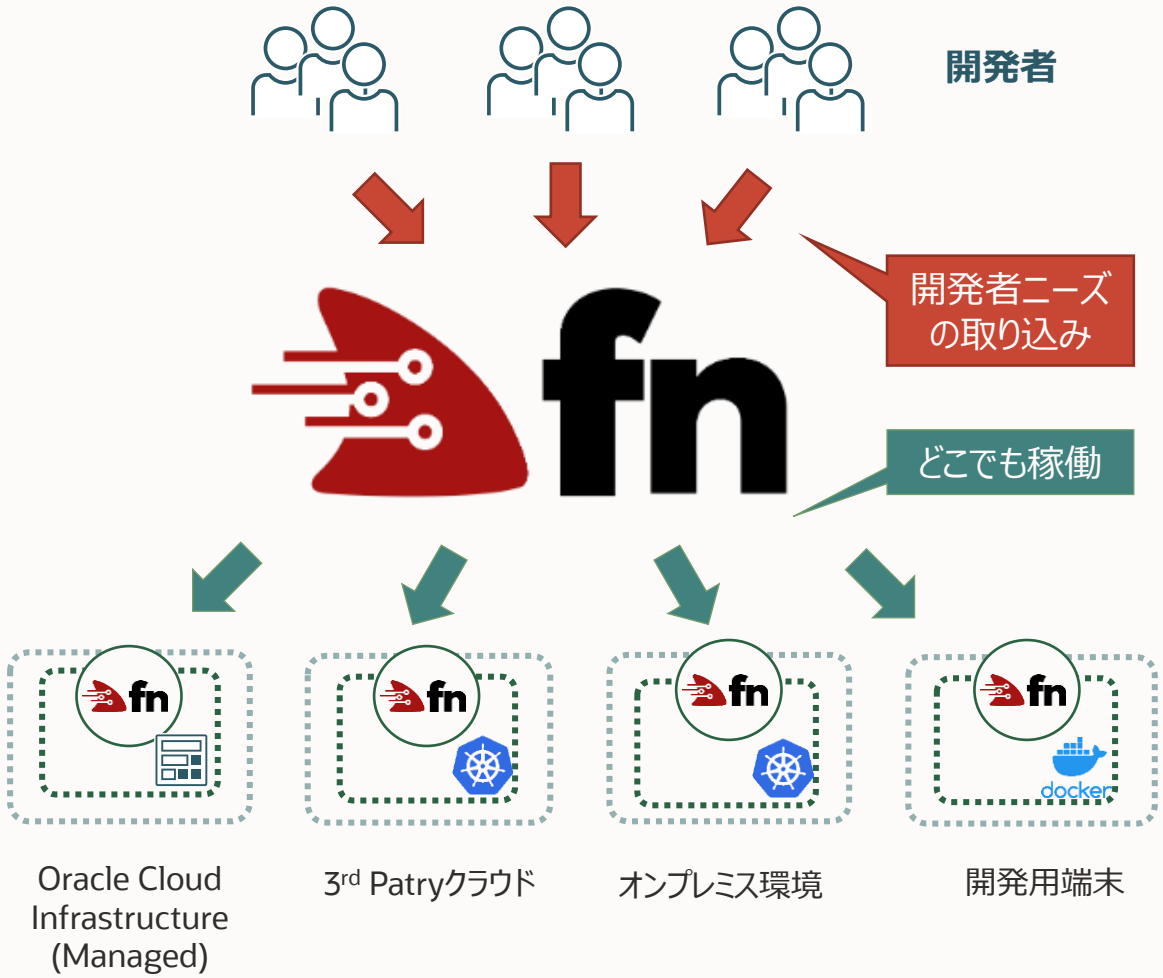
Event-driven

- OCIのトリガーによってコードを実行
- API Gateway, Service Connector Hub, Events, Notifications

コンテナベースのオープンソース “Fn Project”
どこでも稼働する可搬性の高いFaaSアプリケーションを実現

OSSベースのエコ・システム
- https://fnproject.io/
- 開発者のニーズを積極的に反映
- ベンダーロックインの懸念不要

場所を束縛されないポータビリティ
- クラウドからオンプレまでどこでも稼働
- 開発用PCでもそのまま稼働



OCI Functions

フルマネージド、高スケーラビリティ、実行時のみ課金のサーバーレス実行環境

■ **ユースケース**

- ビジネスロジック実装/イベント・ドリブン型開発の効率化

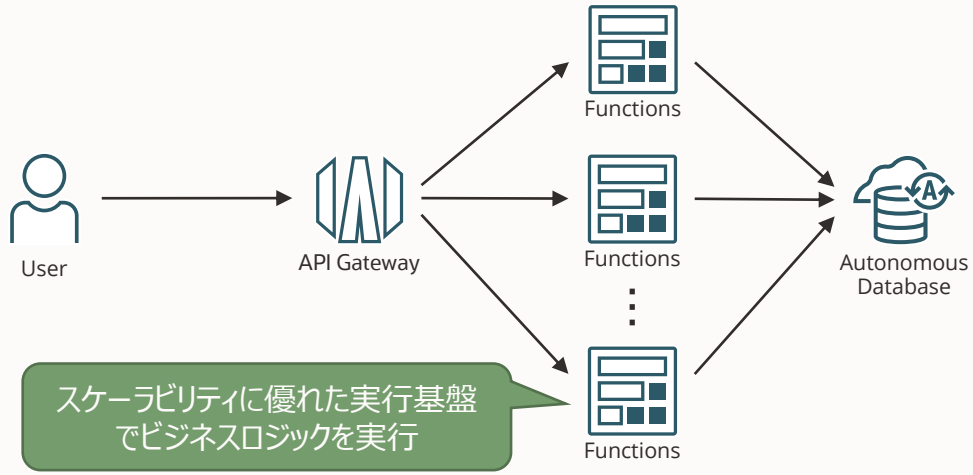
■ **特徴**

- フルマネージドなサーバレス実行基盤(FaaS)
- オープンソースの Fn Project がベースでベンダーロックインなし
- 複数のプログラミング言語をFDKとしてサポート
- 最大5分間の実行時間をサポート
- Provisioned Concurrencyによる初回実行時間の大幅な改善
- シンプルな開発フローとテストハーネスの提供

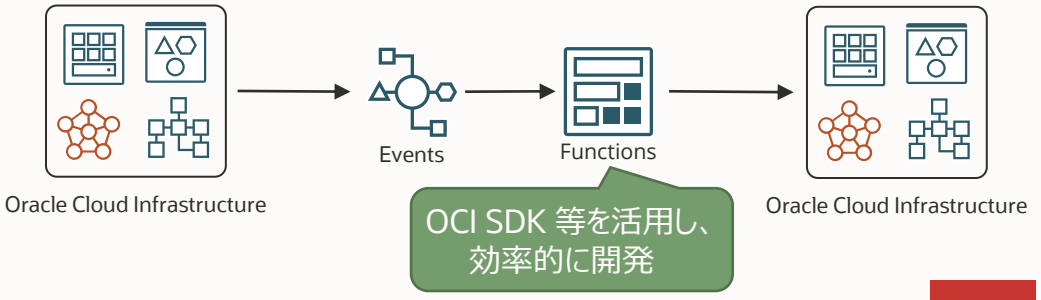
■ **価格**

- 最初の200万リクエストは無料、以降100万リクエストごとに¥28.00
- 最初の400,000 GBは無料、以降10,000 GB-秒ごとに¥19.838
  （※Provisioned Concurrencyを適用した、未使用のFunctionリソースに対しては、25%のリソース利用料が発生）

Web, Mobile, IoTのビジネスロジック実行基盤として活用



クラウドサービス間の連携やマイクロバッチとして活用



OCI Container Instances

インフラ管理不要(サーバレス)のコンテナ実行環境

■ ユースケース

コンテナオーケストレーション(Kubernetes)を必要としない
コンテナアプリケーションのデプロイ

例)API、Web アプリ、継続的インテグレーション・デリバリー
(CI/CD)の ジョブ、開発・テスト環境など

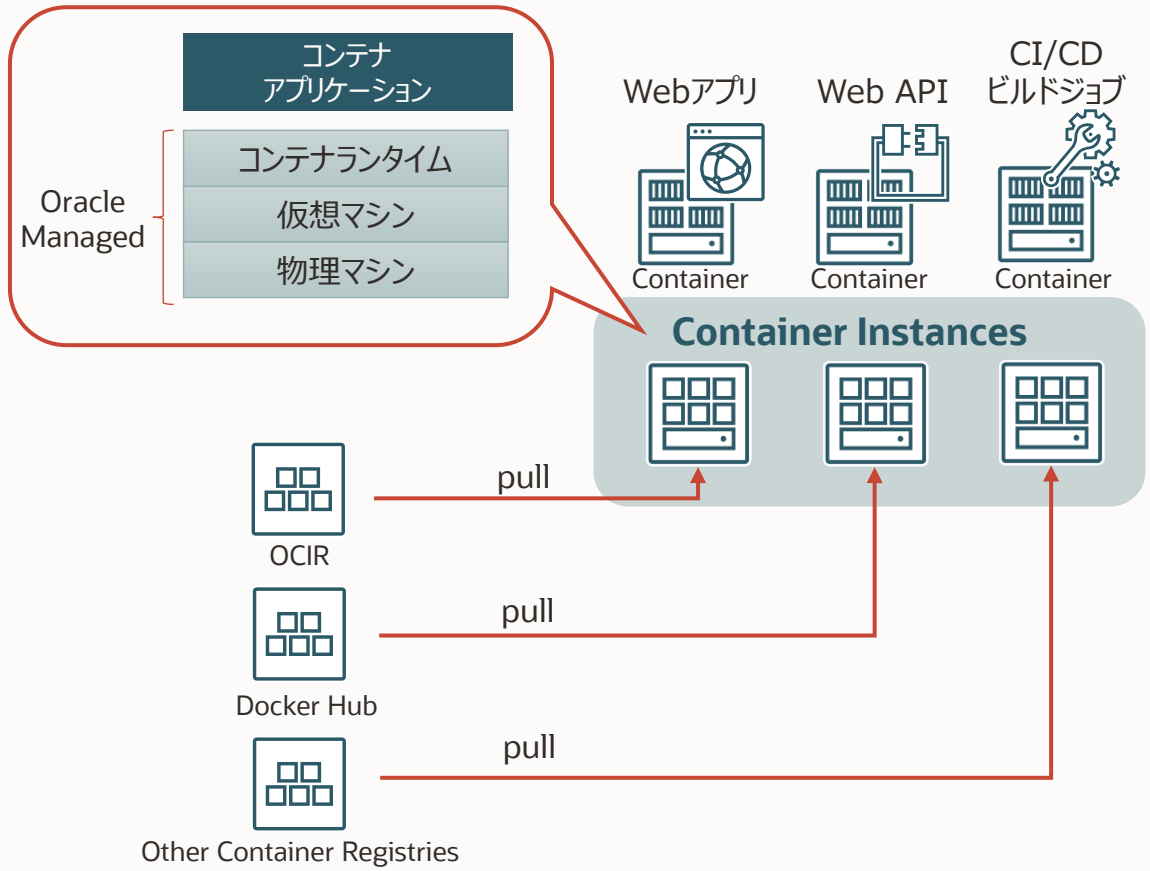
■ 特徴

- セキュアに分離、最適化されたコンテナ実行環境
- ユーザによる仮想マシンの管理、パッチ適用、トラブル
  シューティングが不要
- コンテナを即座に実行可能

■ 価格

通常の Compute と同様(OCPU/メモリ課金)

※ Serverless Container Instances に対する課金はなし



開発自動化

開発自動化に関するサービス

OCI Cloud Native Landscape

アプリケーション実行基盤

- Container Engine for Kubernetes (Managed/Virtual Nodes)
- Container Instances
- Oracle Functions
- WebLogic Server for OCI/OKE

API 管理

- API Gateway

メッセージング

- Queue
- Streaming

サービス間連携

- Notifications
- Service Connector Hub
- Events

オブザーバビリティ

- Monitoring
- Logging
- Application Performance Monitoring

開発フレームワーク

- Helidon MP/SE
- Micronaut

開発・運用管理ツール

- OCI DevOps
- OCI Registry
- Resource Manager
- OCI Service Mesh
- OCI Workflow*
- MicroTx (Free版)

OCI Core Infrastructure

Compute | Database | Storage | Networking

*: Coming Soon!

OCI DevOps

ソースコード、ソフトウェアビルド、アーティファクト、リリースを一括管理

■ ユースケース

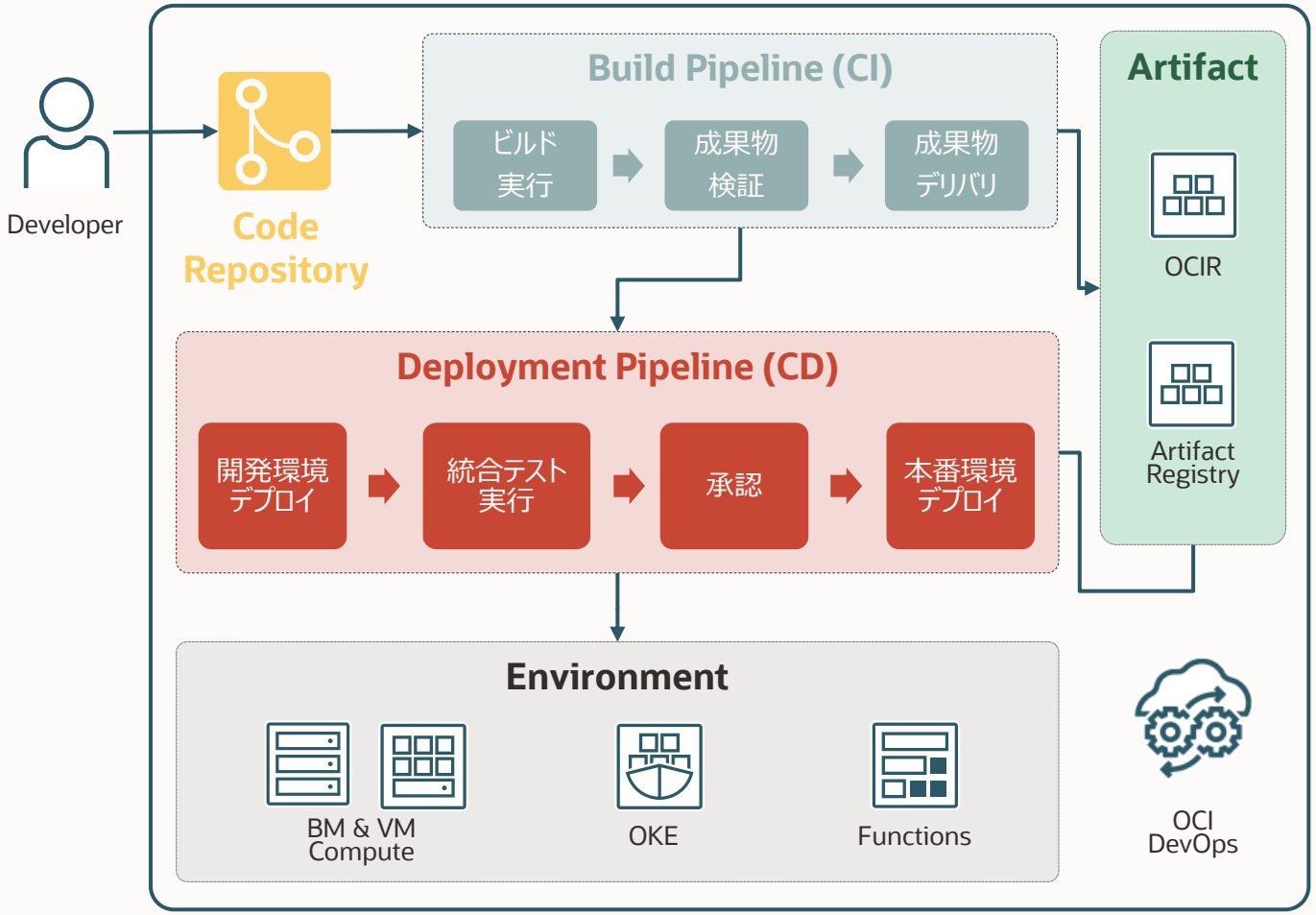
継続的インテグレーション／継続的デリバリー（CI/CD）の実現、アプリケーション開発のライフサイクルを想定したシステム開発および運用環境の整備

■ 特徴

- コード管理、ビルド・パイプライン、アーティファクト管理、デプロイメント・パイプラインから構成される各機能を用いた、**一気通貫の開発・運用プロセス**を実現
- アーティファクトサービス（OCIR,Artifact Registry）と連携して、実行環境を問わず、**幅広い環境へのデプロイ（Compute,OKE,Functions）**を実現
- デプロイ戦略として、**Blue/Greenデプロイ、カナリアリリース**に対応

■ 価格

無料 (※コード管理のストレージ、およびビルドのためのcomputeは別途必要)



Oracle Cloud Infrastructure Registry (OCIR)

コンテナ・イメージを管理するプライベート・レジストリ

■ **ユースケース**

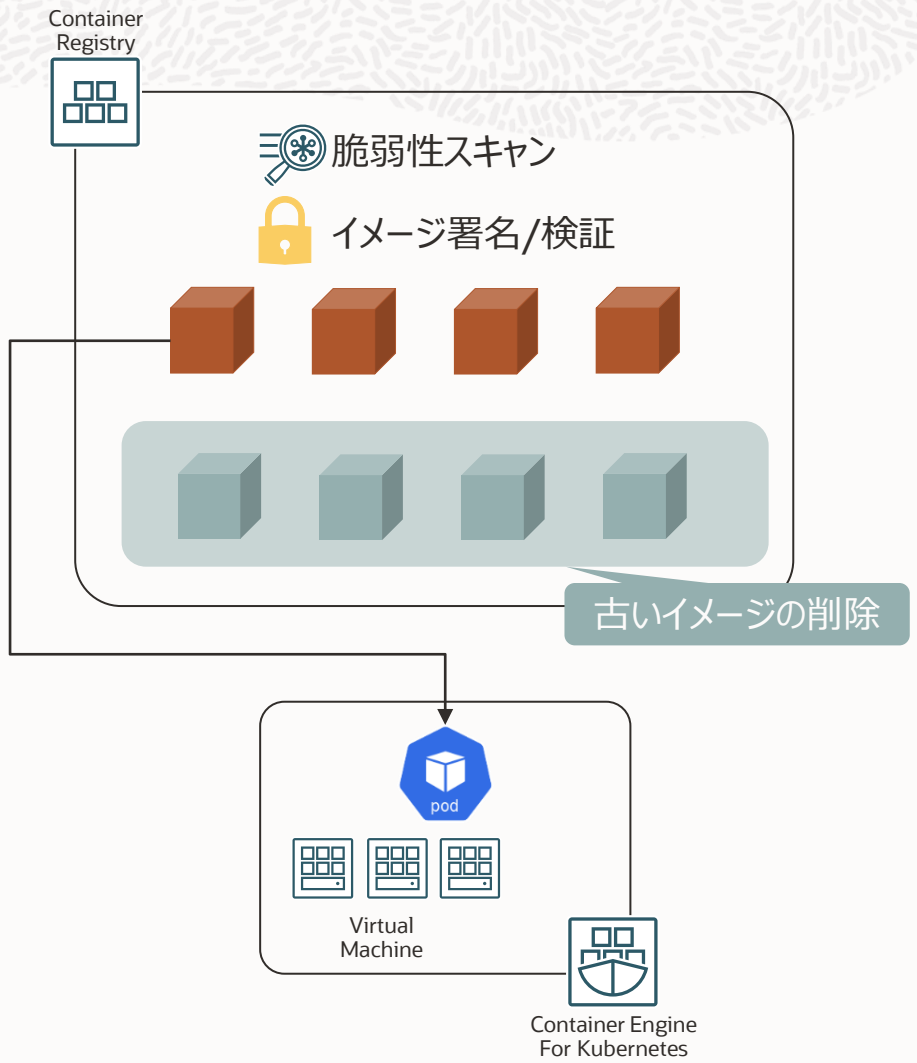
DockerやKubernetesで使用するコンテナ・イメージをセキュアに保管、安全性/信頼性の高いコンテナイメージの配信

■ **特徴**

- 単一のコンテナイメージ(タグ)で**OSやCPUアーキテクチャに依存しない保管**を実現(マルチアーキテクチャ対応)
- 保管しているコンテナイメージの定期的な脆弱性スキャンや署名/検証による**セキュアなイメージの維持を実現**
- イメージ保持ポリシーの適用による**意図しないレジストリサイズの増大防止や最新のコンテナイメージの維持**を実現

■ **価格**

無料(※Object Storage/Network利用分のみ課金)



運用自動化

運用自動化に関するサービス

OCI Cloud Native Landscape

アプリケーション実行基盤

- Container Engine for Kubernetes (Managed/Virtual Nodes)
- Container Instances
- Oracle Functions
- WebLogic Server for OCI/OKE

API 管理

- API Gateway

メッセージング

- Queue
- Streaming

サービス間連携

- Notifications
- Service Connector Hub
- Events

オブザーバビリティ

- Monitoring
- Logging
- Application Performance Monitoring

開発フレームワーク

- Helidon MP/SE
- Micronaut

開発・運用管理ツール

- OCI DevOps
- OCI Registry
- Resource Manager
- OCI Service Mesh
- OCI Workflow*
- MicroTx (Free版)

OCI Core Infrastructure

Compute | Database | Storage | Networking

*: Coming Soon!

Oracle Container Engine for Kubernetes (OKE)

高可用性と開発生産性を両立するKubernetesプラットフォーム

■ ユースケース

迅速なコンテナプリケーションのデプロイと可用性の高いKubernetesプラットフォームの実現、コンテナアプリケーション運用管理の省力化

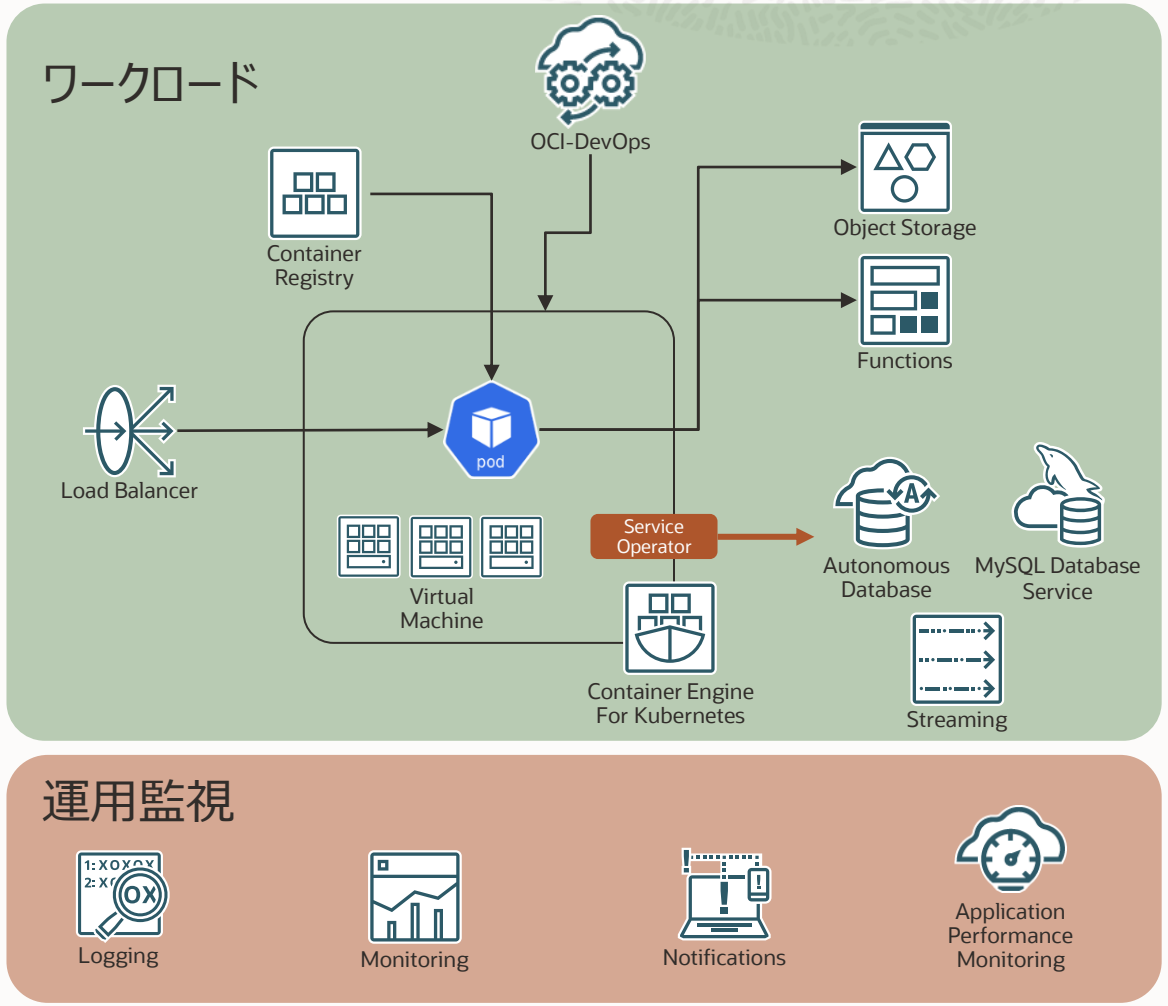
■ 特徴

- Oracle Databaseなど他のOCI周辺サービスとの親和性による**効率的なコンテナアプリケーション環境構築の実現**
- OCI Service Operator for Kubernetesを利用した**周辺サービスの効率的な運用管理**
- 仮想サーバ(VM)だけではなく、**ベアメタルサーバ、GPUやHPCなどを利用し、多彩なワークロードを実現**

■ 価格

Basic Cluster: 無料(※)

Enhanced Cluster(1Clusterあたり): ¥14/hour(※)
(※別途Compute/Block Volume/Network/Load BalancerなどのIaaSサービス)利用分を課金)

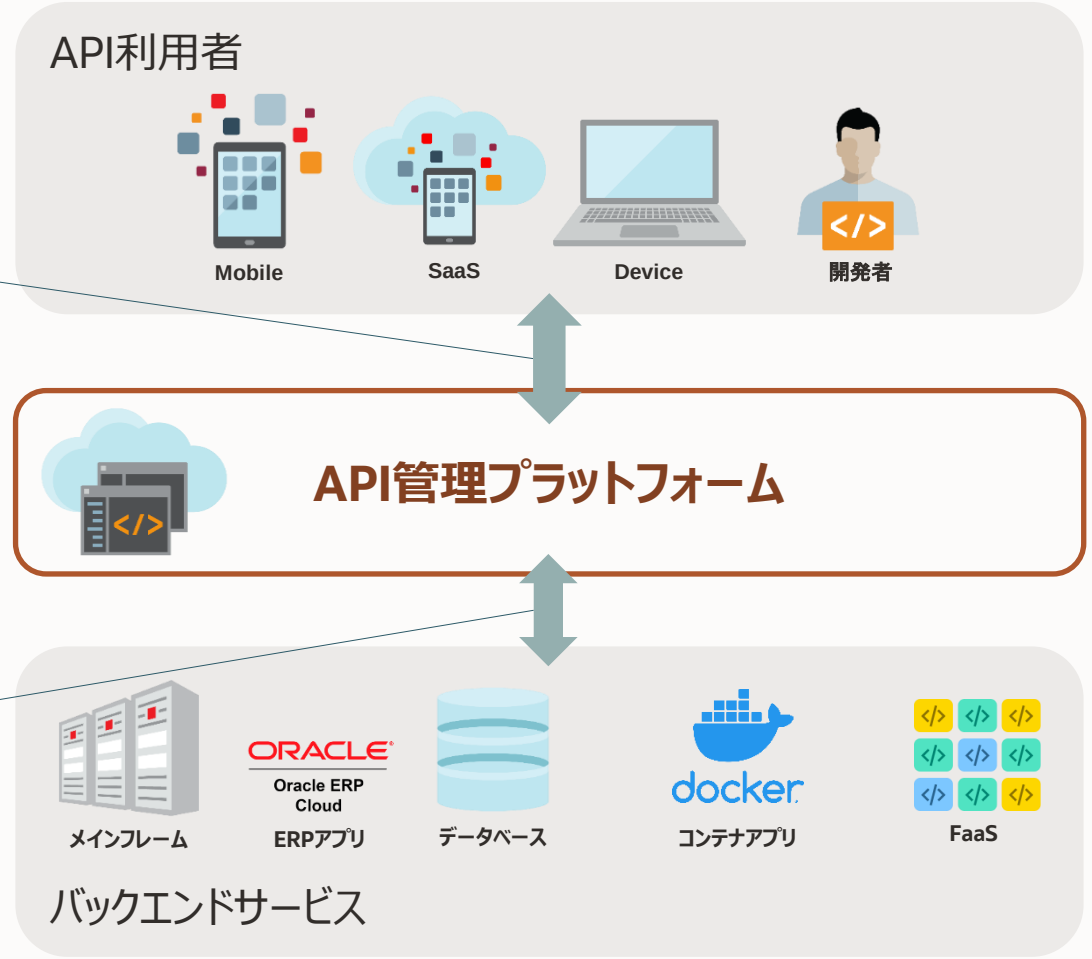


OCI API Gateway

プラットフォームを利用することによる、API管理の一元化

認証方法やアクセスポイントが統一されており、APIを利用する側の実装や管理がシンプルに

APIサービスの監視先やライフサイクルの管理先を一元化



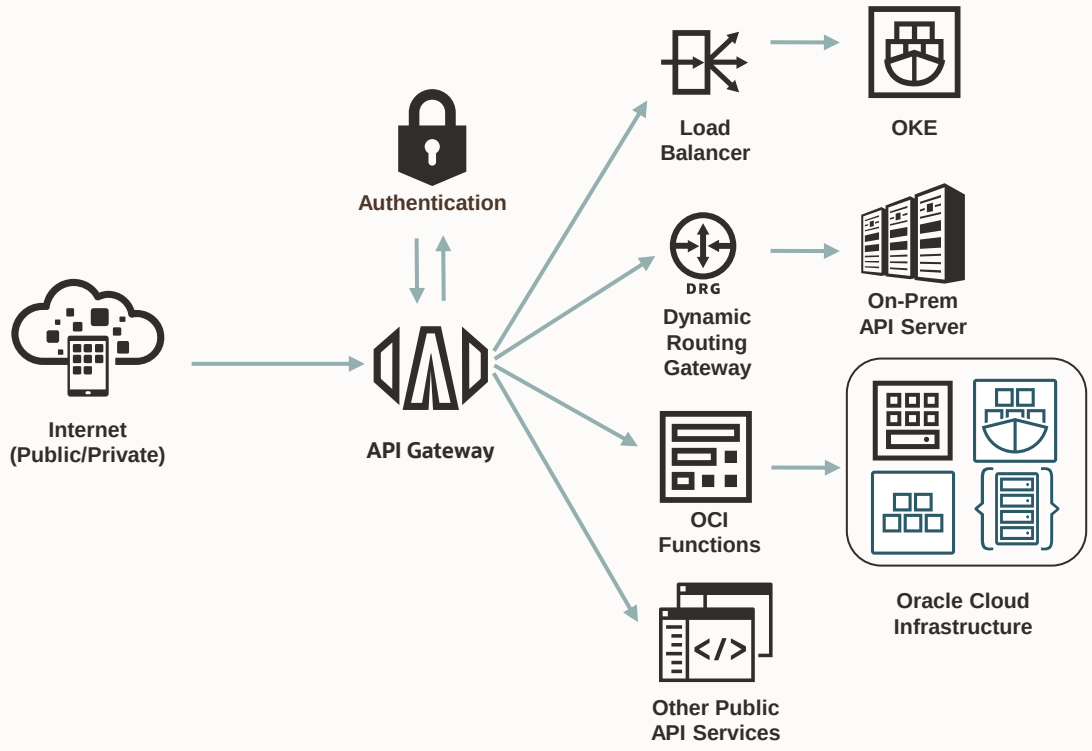
OCI API Gateway

バックエンドサービスの入り口として機能するセキュアなAPIエンドポイント

API Gatewayは、様々なバックエンド(OCI Functions/OKE/パブリックAPI/オンプレミス)のREST APIのエンドポイントになる

【現時点での提供機能】

- 堅牢性の提供
  - カスタム認証
  - JWT検証
  - 流入制限／CORSサポート
- ゲートウェイやAPIの監視（可観測性）
  - ログ／モニタリング／メトリクス
- デザイン管理
  - OpenAPI 2.0/3.0のサポート
- ライフサイクル管理
  - GUI/JSONなど用途に合わせたGatewayの操作



Resource Manager

■ ユースケース

- OCI 上のインフラストラクチャをコードとして管理し、ソフトウェアの提供を効率化したい
- 環境構築、変更時の人的ミスを削減したい

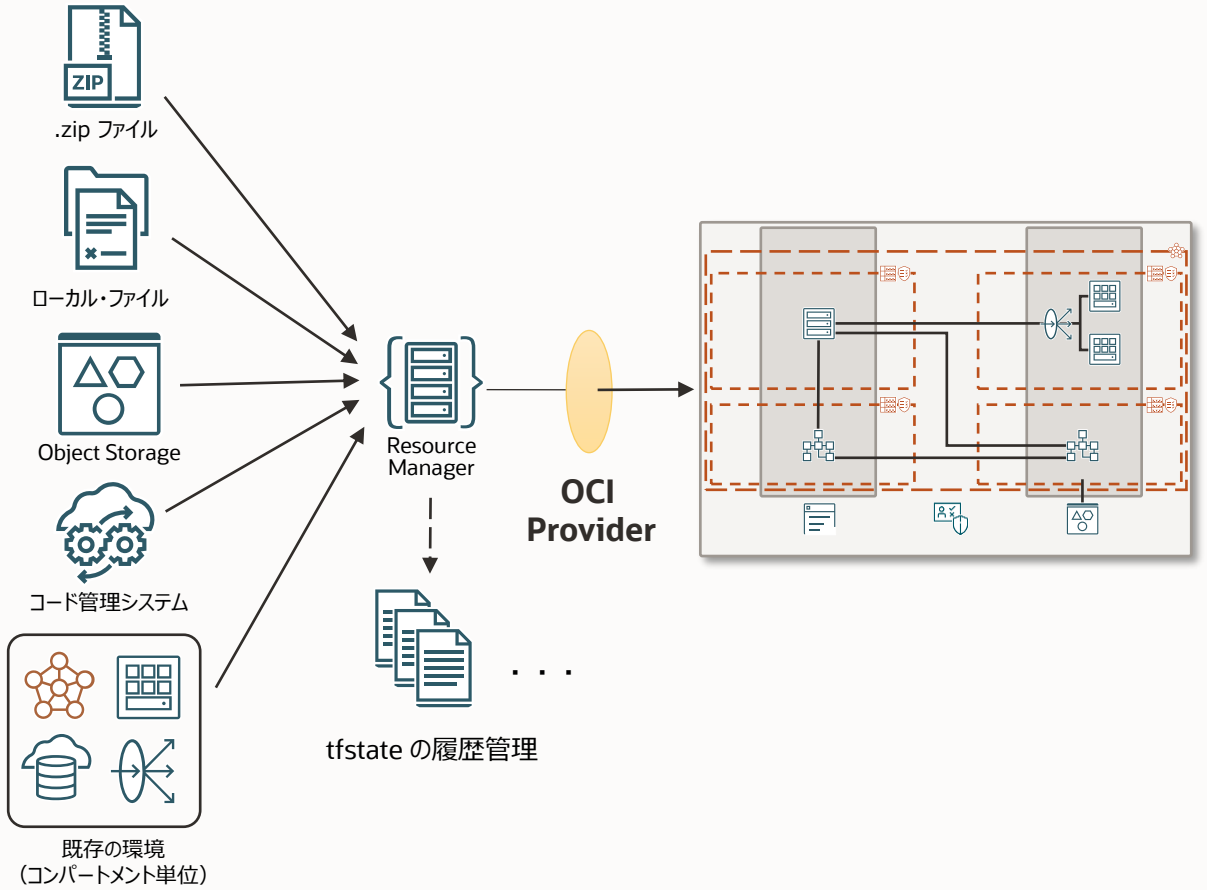
■ 特長

- Terraform をフルマネージドなサービスとして提供
- 状態管理ファイル(tfstate)の安全な管理
  - メンバー間の共有/履歴管理/IAM による権限管理
- Terraform 構成ファイルの格納場所をユースケースに応じて複数サポート
- スキーマ・ドキュメントを用いた OCI コンソールの拡張

■ 価格

- Resource Manager の使用は無料

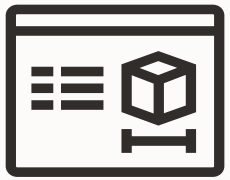
（※プロビジョニングされたインフラに対する課金のみ発生）



Oracle Cloud にて提供するデータベースサービス

多様なビジネス要件に対応する柔軟なサービスを提供

Enterprise and Standard Database Services



Exadata Cloud Infrastructure

Exadata Cloud@Customer

Shared

ADB on Dedicated Infrastructure

ADB on Cloud@Customer

Base Database Service

Exadata Database Service

Autonomous Database

Automation

シングル/小規模なワークロード	あらゆるワークロードに対応する最高のパフォーマンス、拡張性、統合性		
Oracle管理のインフラストラクチャとお客さま管理のデータベース		Oracleによるフルマネージド	
Oracle Cloud Infrastructure	お客さまのデータセンター	Oracle Cloud Infrastructure	お客さまのデータセンター

Autonomous Database
新時代のデータベース・サービス

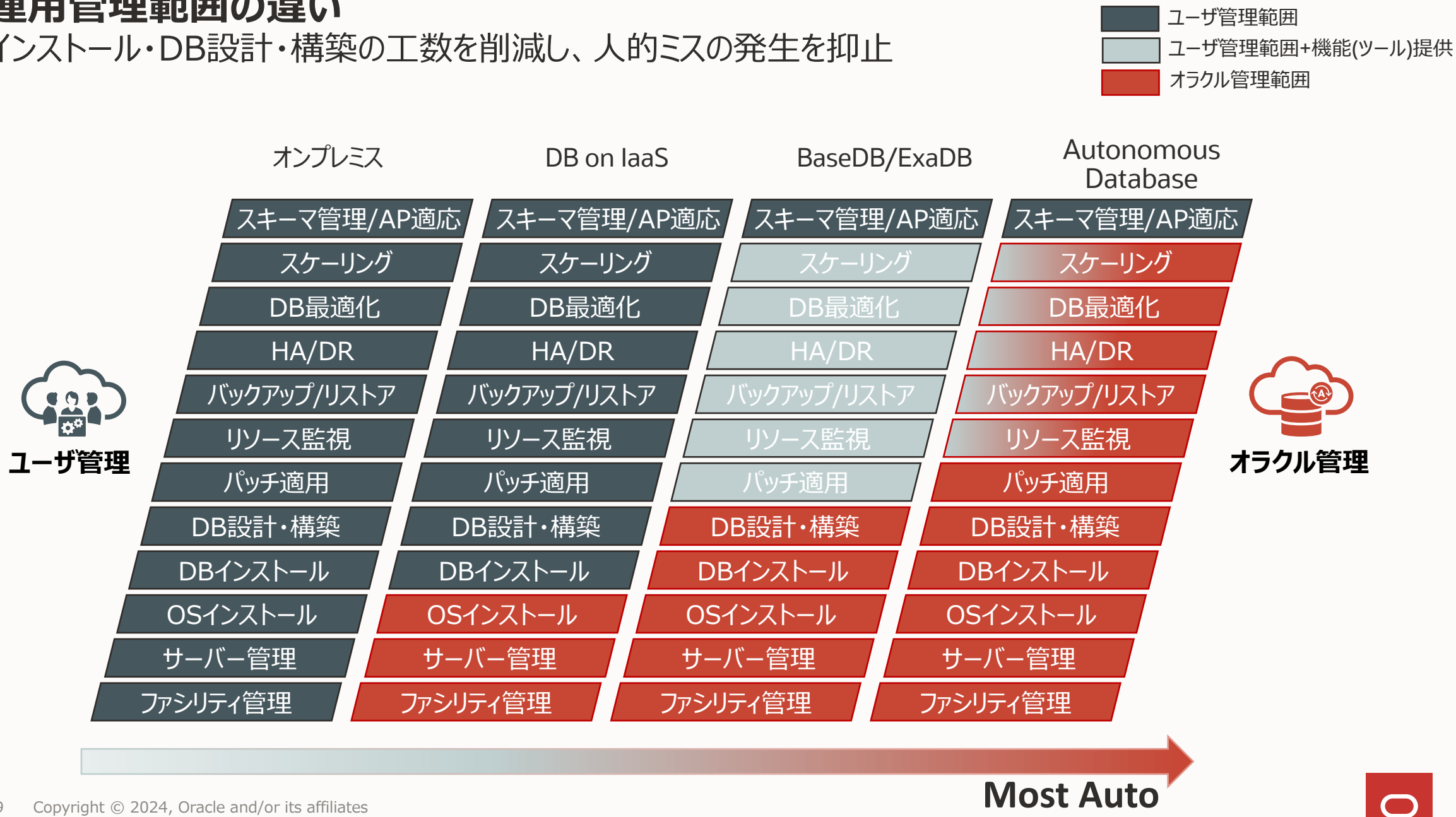
完全な マネージド サービス	**実績のあるOracle Database / Exadata が基盤** 高性能・高可用性・高セキュリティが実装済み **AI/機械学習を利用した完全自動運用** **様々なツールが無償で同梱、DBで完結**
完全な 柔軟性	**CPUを無停止で増減可能** **ワークロードに応じた自動増減も可能** **CPU/ストレージは1秒単位で課金**
完全な マルチモデル	**あらゆるワークロード(OLTP/分析/混在)** **あらゆるデータタイプ(構造化/JSON/グラフ等)** **1つのデータベースで対応可能**

最小構成と月額費用：2 ECPU / 20 GB 、¥69,995.52 + ¥323.68 = ¥70,319.2

※2 ECPU は、1 vCPU 相当です。

運用管理範囲の違い

インストール・DB設計・構築の工数を削減し、人的ミスの発生を抑止



Oracle Cloud Observability and Management (O&M)

■ OCIに最適化された運用管理プラット・フォーム

すべての監視データを可視化し、シームレスに連携

リソースの稼働状況やアクセスログは常時モニタリングし
異常を素早く検知するフレームワーク

アプリケーションやデータベースのパフォーマンス改善を
様々なインフラスタックの観点からアドバイス

リモートワークに最適な接続の容易性と厳格な認証

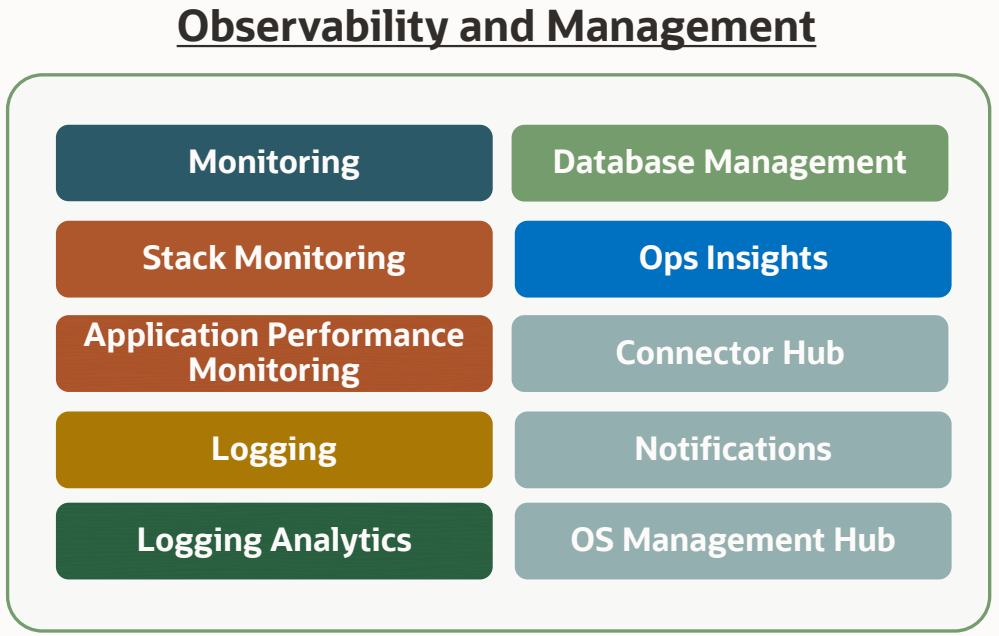
マルチクラウド、オンプレミスのリソース監視へ拡張可能

■ 最先端のテクノロジー

機械学習を用いたログ分析、将来必要となるリソースの
需要予測どのイノベーティブな機能を提供

■ オープン・スタンダート

クラウド・ネイティブに準拠し、OpenTracingやFluentd等の
オープンな技術を採用、既存ツールとの相互運用性を向上

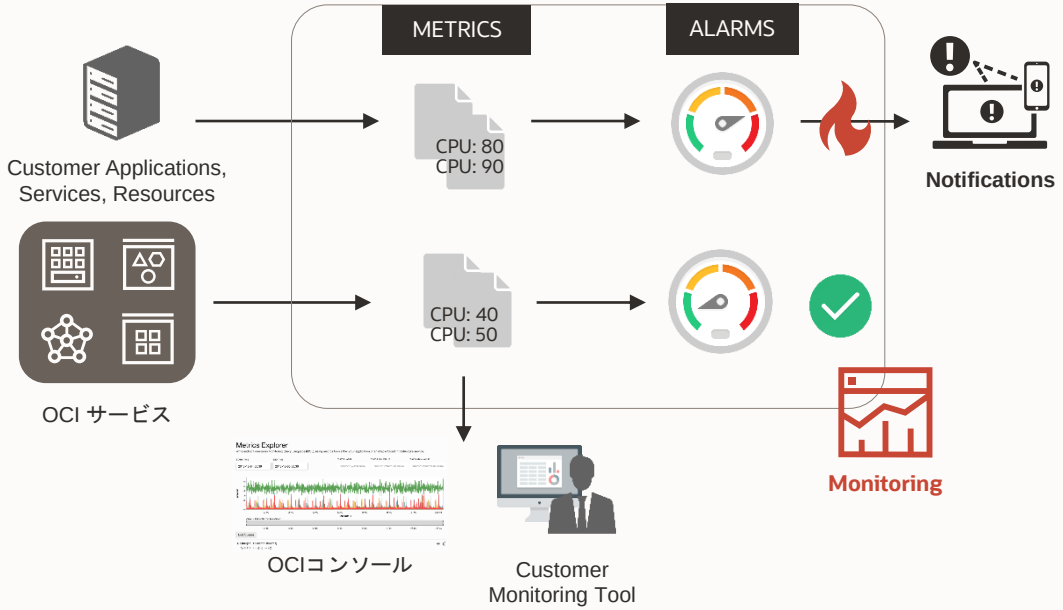


Monitoring

各OCIサービスやコンピュートのリソースを監視する基本監視機能

サービス概要/特徴

- コンピュート・インスタンス、VNIC、ブロック・ボリューム、ロードバランサーなどのサービスやリソースの性能や状態を監視
- 特別な設定は必要なく、標準機能として提供
- 事前定義済のビジュアライゼーション・ダッシュボードの提供
- ユーザー自身によるカスタム・メトリックも定義可能



ユース・ケース

- 以下のようなリソースの監視を行うとともに、異常が発生した場合には管理者に通知することで問題を迅速に解決
  - コンピュート・インスタンスのCPU, メモリ, Disk、ネットワーク
  - ロードバランサー, VPN, Fast Connectなどのネットワークリソースの稼働状況
  - ストレージやデータベース・サービスなどのスループット
  - OS上で動作するミドルウェアや特定のアプリケーションのプロセス監視
  - リソースの予期せぬ停止や過負荷などのしきい値を超えた際のアラーム通知

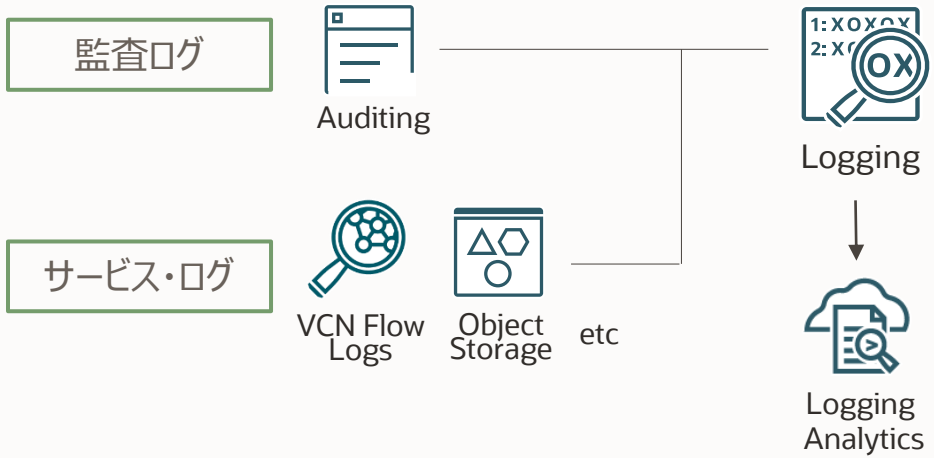
サービス価格

- 基本無料
- カスタムメトリックやメトリック・エクスプローラーでの独自の運用項目の追加時には、一部有料

Logging
監査ログやサービスログといったOCIの運用に不可欠なログ出力管理

サービス概要/特徴

- OCI内のログ出力を管理し、ログデータの簡易的な分析と連携管理を提供
- OCIのコンソール操作やログイン、API実行などの監査ログ
- VCN Flow Logs, Load Balancers, Functions Object Storage, WAFなど様々なサービス・ログ
- Logging Analyticsにログを転送させることで直感的なログのビジュアライズや高度な分析が可能



ユース・ケース

- OCIの監査ログから不正ログインや疑わしい操作がないかを定常的に監視
- Load BalancerやVNC Flow Logsといったネットワークのログから不正アクセスの兆候を検出
- オブジェクト・ストレージにある機密ファイルへのアクセスが適切に行われているかをチェック
- ログ保管のコンプライアンス対応として、Object Storageに長期的に安価にログを補完

サービス価格

- 10GBまで無料, 以降、1GBあたり ¥7/月
- 保存期間は最大6カ月

顧客事例：自治体向けパッケージ

クラウド・ネイティブの機能を利用してアプリケーションのモダン化を実現

株式会社RKKCSと日本オラクル、総合行政システムのガバメントクラウド移行に向け連携を強化

自治体向け基幹系パッケージをOCIで実装し、クラウド・ネイティブなアプリケーション開発を推進

東京—2023年4月13日

日本オラクル株式会社（本社：東京都港区、取締役 執行役 社長：三澤 智光）は本日、株式会社 RKKCS（本社：熊本県熊本市西区、代表取締役社長：金子 篤）と連携を強化し、同社の自治体向け基幹系パッケージ「総合行政システム」と関連システムにおける国が定める自治体システム標準化の対象業務を、ガバメントクラウドに採択された「Oracle Cloud Infrastructure (OCI)」で実装していく方針であることを発表します。RKKCSによるガバメントクラウドにおけるクラウド・サービス・プロバイダー（CSP）選定において、OCIが備えるランサムウェアを防ぎながらデータを保護する高度なセキュリティや高い可用性、そして優れたコスト・パフォーマンスを高く評価いただきました。今後、「総合行政システム」のOCI移行とクラウド・ネイティブなアプリケーション開発に対し協働して推進していきます。

クラウド・ネイティブ・テクノロジーを活用した新規アプリケーションの実装

国が求めるアプリケーションのモダン化に従い、クラウド・ネイティブなマネージド・サービス活用による運用コストの最適化を図ります。新規アプリケーションでは従来の「Web3層アプリケーション」ではなくクラウドに最適な「Web APIアーキテクチャ」を採用します。このアーキテクチャをベースに、アプリケーションをコンテナ化し、コンテナのCI/CD環境を構築することによって、リリース・サイクルの短縮や運用の効率化を図ります。具体的に、「Oracle Container Engine for Kubernetes」による柔軟なコンテナ実行基盤の構築、「OCI Service Mesh」、「OCI API Gateway」、「OCI Identity and Access Management」を用いたセキュリティと認証・認可機能の実装、「OCI DevOps」によるパイプラインの自動化、IaC（Infrastructure as Code）による基盤構築・構成管理の負荷軽減を目指します。

事例のポイント

OCIのマネージドサービスを利用してクラウドに最適な「Web APIアーキテクチャ」を採用しモダンアプリケーションを実現

- アプリケーションの**コンテナ化**
- 「OCI DevOps」によるパイプラインの**自動化**
- IaC（Infrastructure as Code）による**基盤構築・構成管理の負荷軽減**
- セキュリティと認証・認可機能の実装

まとめ

■ クラウドネイティブとは

拡張性・柔軟性に優れたアプリケーションをクラウドの特性を活かし最小限の労力で構築および実行できるようにすること

■ クラウドネイティブを実現するための3つのステップ

- **コンテナ化**
  軽量で構築、移行がしやすいOSレベルの仮想化技術
  機敏かつ信頼性の高いアプリケーションリリースを実現

Container Instances
Oracle Functions

- **開発自動化**
  自動的なビルド、テスト、デプロイが可能なCI/CD環境の構築
  開発スピードの向上とアプリケーションリリースにおけるリスクの低減

OCI DevOps
OCI Registry

- **運用自動化**
  コンテナ化されたアプリケーションをKubernetes基盤上で自律的に運用
  運用管理の自動化、マネージドサービスの利用

Oracle Container Engine
for Kubernetes
API Gateway
O&Mサービス
Autonomous Database


ORACLE