



コンテナ/OpenShiftの概要紹介

Red Hat K.K.

2023.12

Agenda

- コンテナはなぜ使われるのか？
- コンテナ利用に必要なもの
- コンテナはどのように使われるのか？
- なぜKubernetesが必要なのか？
- Red Hat OpenShiftの主な機能

コンテナはなぜ使われるのか？

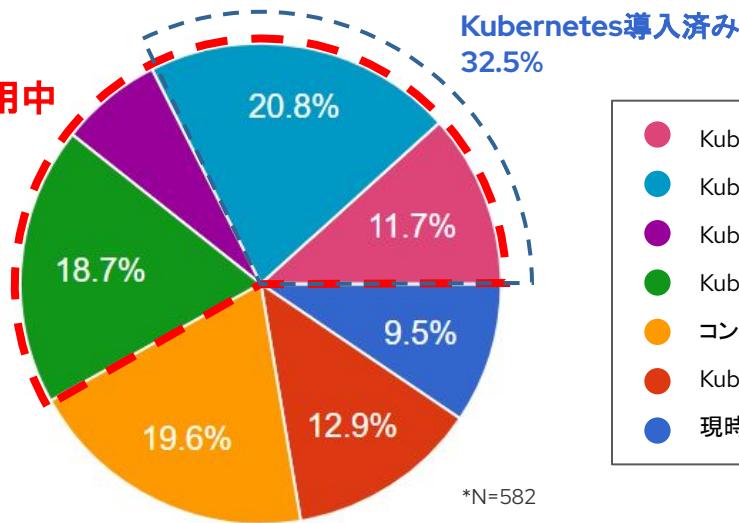
多くの企業で利用が進むコンテナ

半数を超える企業でコンテナが使用され、
約1/3はKubernetes(コンテナ基盤ソフトウェア)を使用しています。

あなたの企業(もしくは支援先企業)ではKubernetesを利用していますか?

※Kubernetesのプロダクトは問いません。

コンテナ使用中
58.0%



- Kubernetesの本番利用を行っている
- Kubernetesの検証、導入構築を行っている
- Kubernetesの利用を計画/検討している
- Kubernetesに関する情報収集は行っている
- コンテナ(Dockerなど)は使用/検討中だがKubernetesはこれから
- Kubernetes/コンテナに関してはよく知らない
- 現時点では使うことは考えていない

なぜコンテナを使うのか

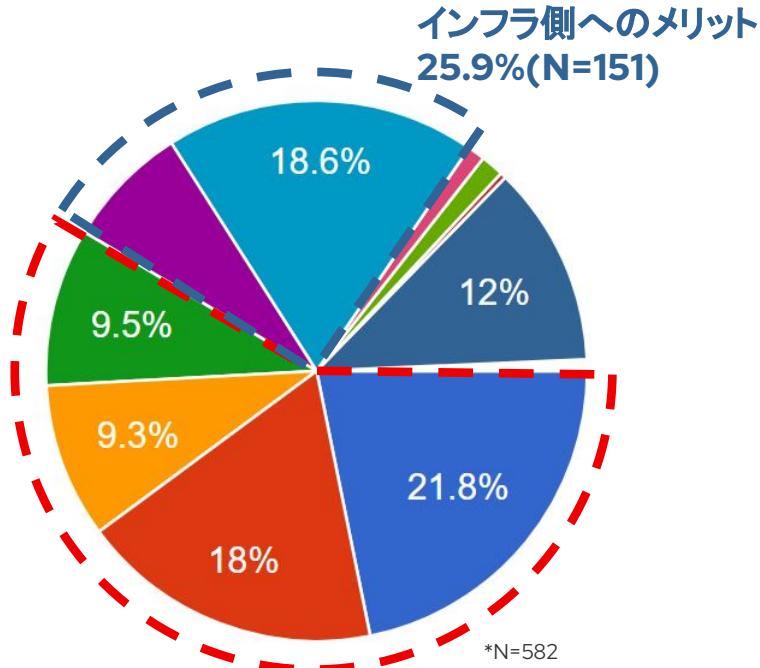
コンテナをうまく使うことで、**ビジネスメリット**を得られるから

コンテナ導入の期待値

既存システムと比べ、コンテナ(Kubernetes)導入に期待する一番のビジネスメリットはなんですか？

アプリ側へのメリット
58.5%(N=341)

- 開発生産性の向上
- 運用の効率化
- ...

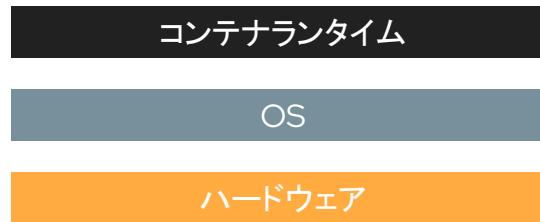
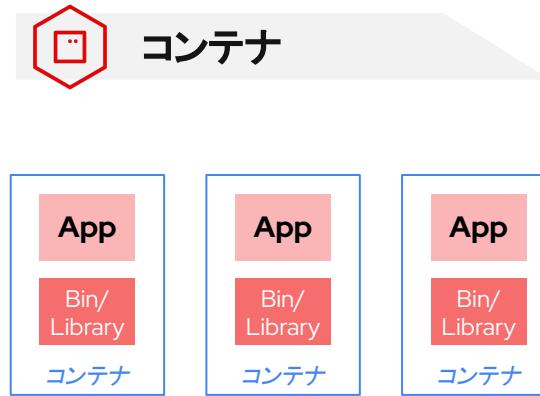
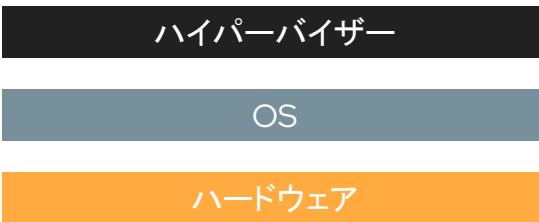


- アプリケーション開発の生産性向上、アジリティ向上
- アプリケーション運用の効率化、コスト削減
- アプリケーションリリースサイクルのスピード向上
- アプリケーションのポータビリティ(可搬性)
- インフラリソースの集約率向上、コスト削減
- インフラリソース管理の運用自動化、プロセス改善
- SoE(IoTやAI、機械学習など)の促進
- SoRのアプリケーションモダナイゼーション
- SoRのデータ利活用/マイグレーション
- まだよくわからない

コンテナとは一言で言えば何か？

アプリケーション本体と、
アプリケーションの実行に必要なライブラリ・依存関係など、
必要最小限の要素をひとつにパッケージした姿

コンテナと仮想マシンの違い



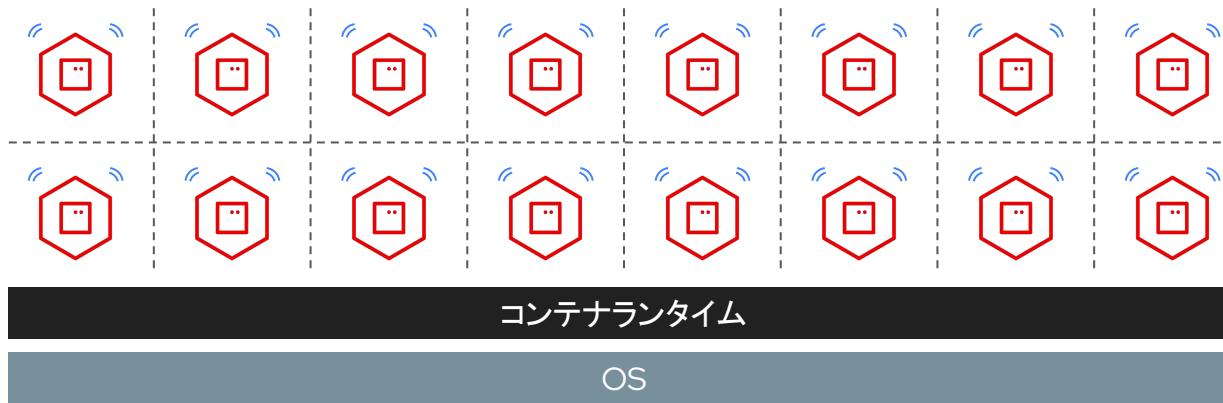
コンテナの特徴

▶ OS上で稼働

- カーネルが持つ機能を利用する。
- 1つのホストの上で複数のコンテナを同時に稼働できる。

▶ 隔離性

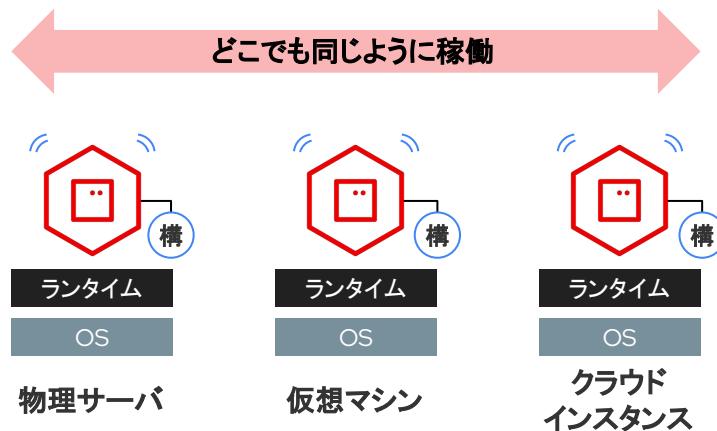
- ホストのカーネルを共有するが、コンテナ同士は隔離され互いに競合しない。
 - コンテナ同士で通信可能にはできる。



コンテナの特徴

▶ 可搬性 (Portable)

- どの環境でも同じように稼働する。
 - 環境に依存する構成情報はコンテナとは別で持つ。



▶ 軽量

- OSが無く、必要最小限の要素のみ持つ。

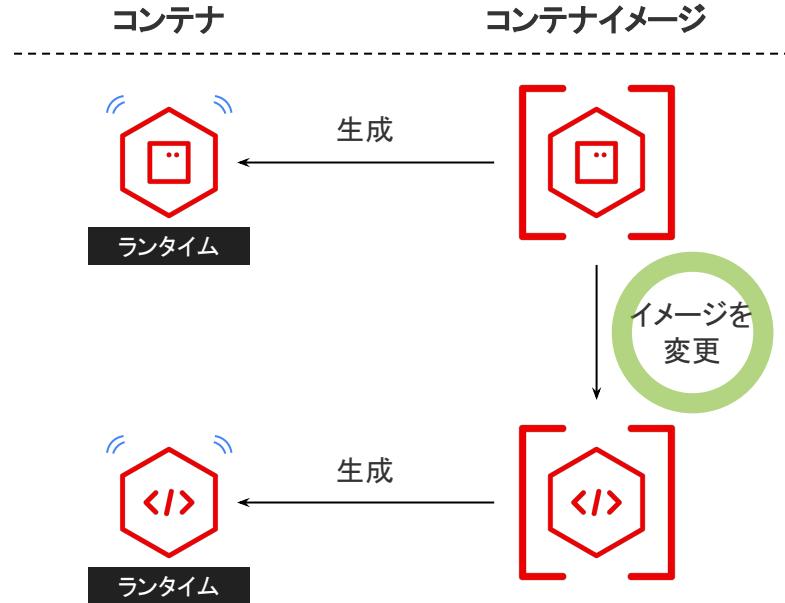
▶ 起動が高速

- OS起動時間を省略できる。

	仮想マシン	コンテナ
容量	1桁 ~ 2桁 GB	2桁 MB ~ 1桁 GB
起動時間	数分	数秒

コンテナの特徴

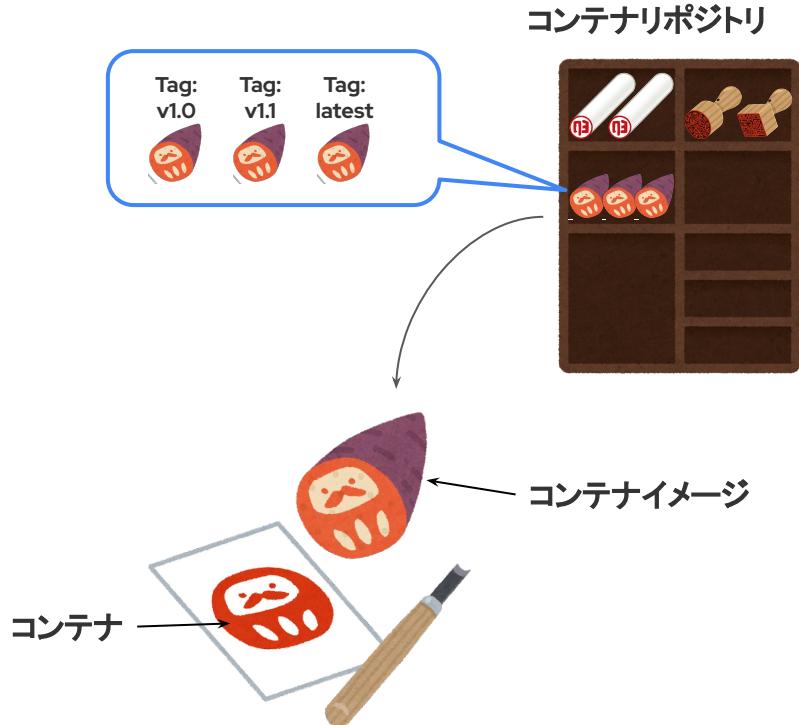
- ▶ **イメージから生成**
 - コンテナイメージから複製して作られる。
- ▶ **不变性 (Immutable)**
 - 同じコンテナイメージから起動したコンテナは、毎回必ず同じものとなる。
- ▶ **揮発性 (Ephemeral)**
 - コンテナに加えた変更は、コンテナが停止すると失われる。
 - コンテナ自身に永続性は無い。
 - コンテナに変更を加えたい場合は、コンテナイメージを変更して新しくコンテナを起動する。
 - 古いコンテナは破棄する。



コンテナを使うために 必要なもの

コンテナイメージ

- ▶ **コンテナの素**
 - あらゆるコンテナはイメージから作られる。
- ▶ **コンテナリポジトリで管理**
 - 同じイメージは系列立てて管理される。
 - それぞれのイメージはタグで区別される。
- ▶ **利用できるイメージ**
 - パブリックなイメージ
 - Web上で公開されている。
 - プライベートなイメージ
 - ユーザーが独自に作る。
- ▶ **イメージの作り方(イメージのビルト)**
 - 既存イメージに追加変更を加えてビルトする。
- ▶ **セキュリティには要注意**
 - 特にパブリックなイメージは「信頼できる提供元のイメージか」「脆弱性は無いか」など注意すること。



コンテナレジストリ

▶ コンテナイメージの保管場所

- コンテナイメージはレジストリで保管され、共有される。
 - コンテナレジストリからイメージをダウンロードすることを “Pull” と呼ぶ。
 - コンテナレジストリにイメージをアップロードすることを “Push” と呼ぶ。

▶ 使用できるコンテナレジストリ

- パブリックなレジストリ
 - [Docker Hub](#) や [Quay.io](#) など、Web上で公開されたレジストリ。
- プライベートなレジストリ
 - イメージを非公開にするため、独自に構築したり、クラウドのサービスを使うなどして準備する。

コンテナレジストリ



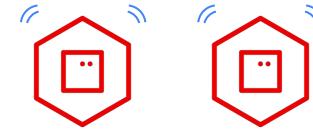
コンテナランタイム

- ▶ **コンテナが稼働するために必要なソフトウェア**
 - コンテナの作成・実行・停止・削除などのライフサイクルの管理をする。
 - ランタイムがなければコンテナは動かない。

▶ 有名なランタイム

- ローカル環境で使うとき
 - [Docker](#) (RHEL7で採用)
 - [Podman](#) (RHEL8, 9で採用)
- コンテナ基盤(Kubernetesなど)で使うとき
 - [cri-o](#) (Red Hat OpenShiftで採用)
 - [containerd](#)

※ ランタイムの役割や機能はおおむね同じだが、思想・内部アーキテクチャは変わるため微妙な違いはある。



ランタイム

ランタイムの役割

- コンテナのライフサイクル管理
 - ハードウェアリソースの分離
 - モニタリング・ロギング
 - コンテナイメージのPull・管理
 - コンテナイメージのビルド・Push
- ...etc

コンテナは
どのように使われるのか？

アプリケーション開発現場でのコンテナのうまい使いかた

▶ アプリケーション本番稼働の問題

○ 環境への依存

- 微妙な環境の違いによって、開発環境とステージング・本番環境で挙動が違うなどの問題が起こりえる。

○ システム基盤への依存

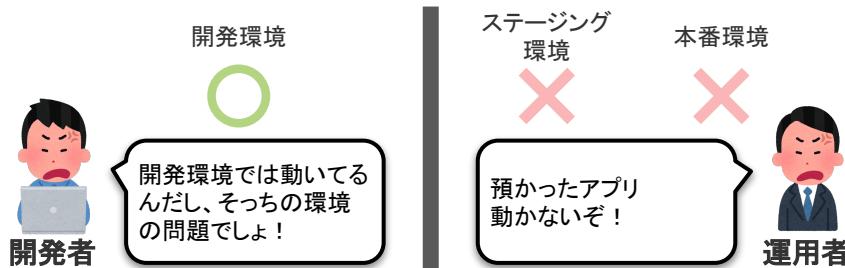
- 別の基盤(例えばオンプレミスからクラウド)へのアプリケーション移行が難しい。

▶ コンテナを使うことによる解決

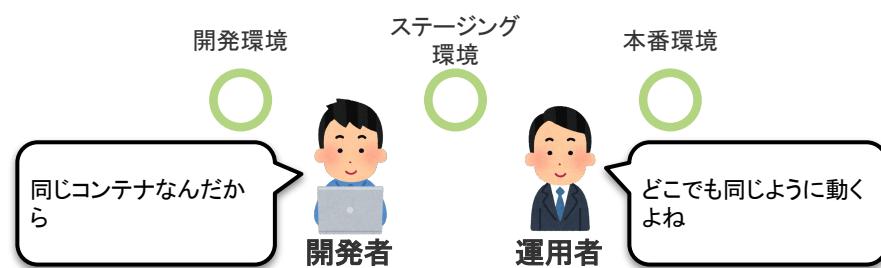
○ 環境やシステム基盤への依存を極小化

- コンテナは基盤への依存性が低いため、環境ごとに挙動が違うことを防ぐことができる。
- 基盤ごとの違いはコンテナ自体とは別で吸収できるため、アプリケーションの移行が比較的容易である。

コンテナ化されていないアプリケーション



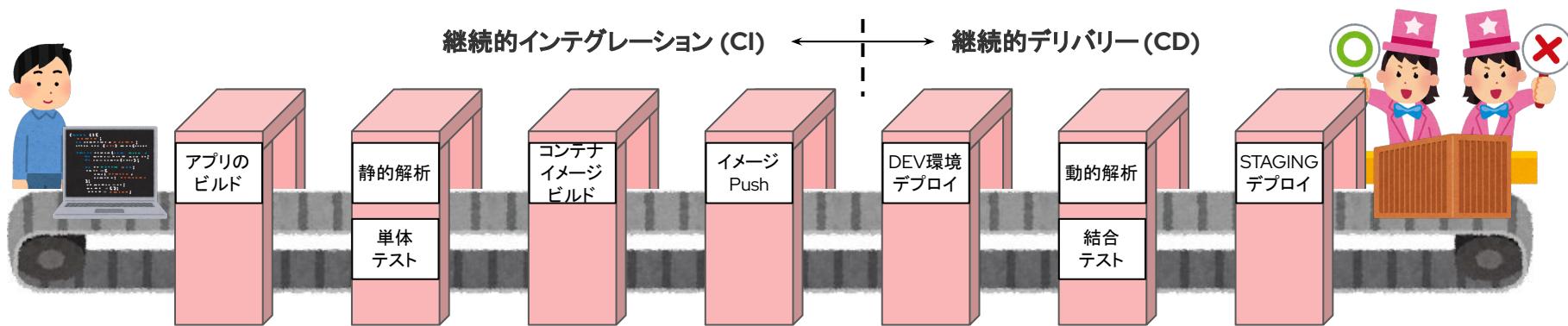
コンテナ化されたアプリケーション



開発～デリバリーまでのコンテナのうまい使いかた

▶ 継続的インテグレーションと継続的デリバリー(CI/CD)

- アプリケーションのビルドからコンテナのビルド、デプロイまでを自動的に行うようとする。
 - 途中で行うコードの解析やテストなども自動化する。
 - CI/CD用のソフトウェアを利用して、“パイプライン”を構成する
- 人の作業を「アプリケーションのソースコードの投入」と「デプロイの承認」など最小限にして極力自動化することで、アプリケーションデリバリーまでの作業品質を均一化する。



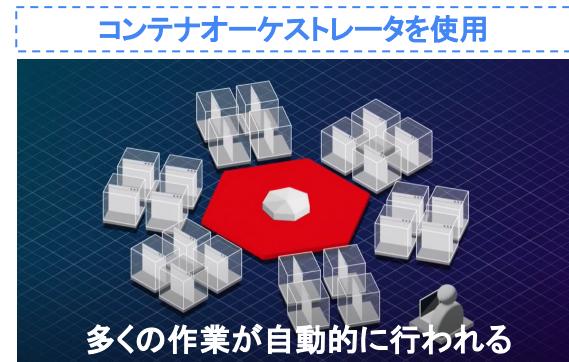
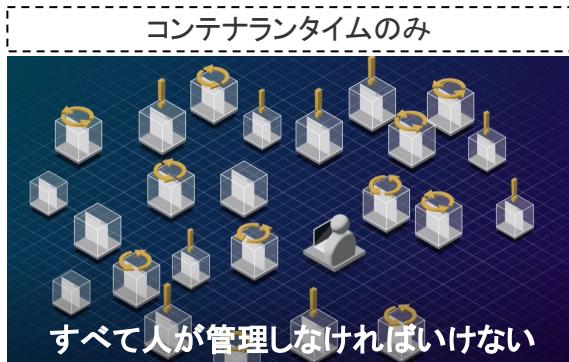
なぜKubernetesが必要なのか？

実稼働するシステムでコンテナを使うために

コンテナオーケストレータ = Kubernetes を使う

コンテナオーケストレータはコンテナの管理・運用を自動化し、ランタイムだけでは足りないシステムの可用性や運用性を提供します。

▶ コンテナの管理・運用を自動化



【人が行う作業】

- 属人的な障害復旧オペレーション
- 手動によるコンテナ変更作業
- アプリケーションごとの設定管理
- 定期的な監視作業

【人が行う作業】

- ビジネス変化に応じた適切なリソース調整

様々なクラウドでのコンテナアプリの実行

クラウドプロバイダごとに異なる実装やサービスの詳細を知る必要がなく、
コンテナアプリの実行状態を記述して、宣言的に実行できます。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment # デプロイの名前
spec:
  replicas: 3 # レプリカ数 (Podの数)
  template: # 作成される Pod のテンプレート
    metadata:
      labels: # Pod 管理に利用されるラベル
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports: # 利用するポート番号
            - containerPort: 80
```

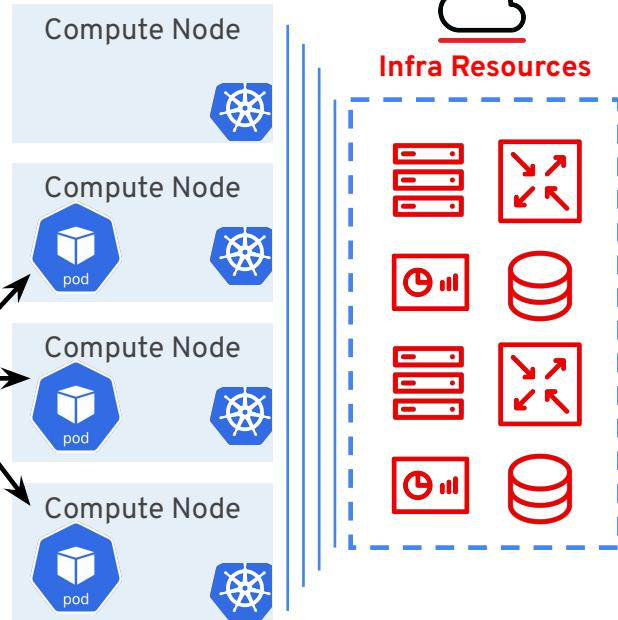
※ Pod: Kubernetes 上でのコンテナアプリの実行単位。

コンテナや、コンテナが利用する外部ストレージの設定などをまとめたもの。
基本的には、1 Pod = 1 コンテナアプリ、を想定。



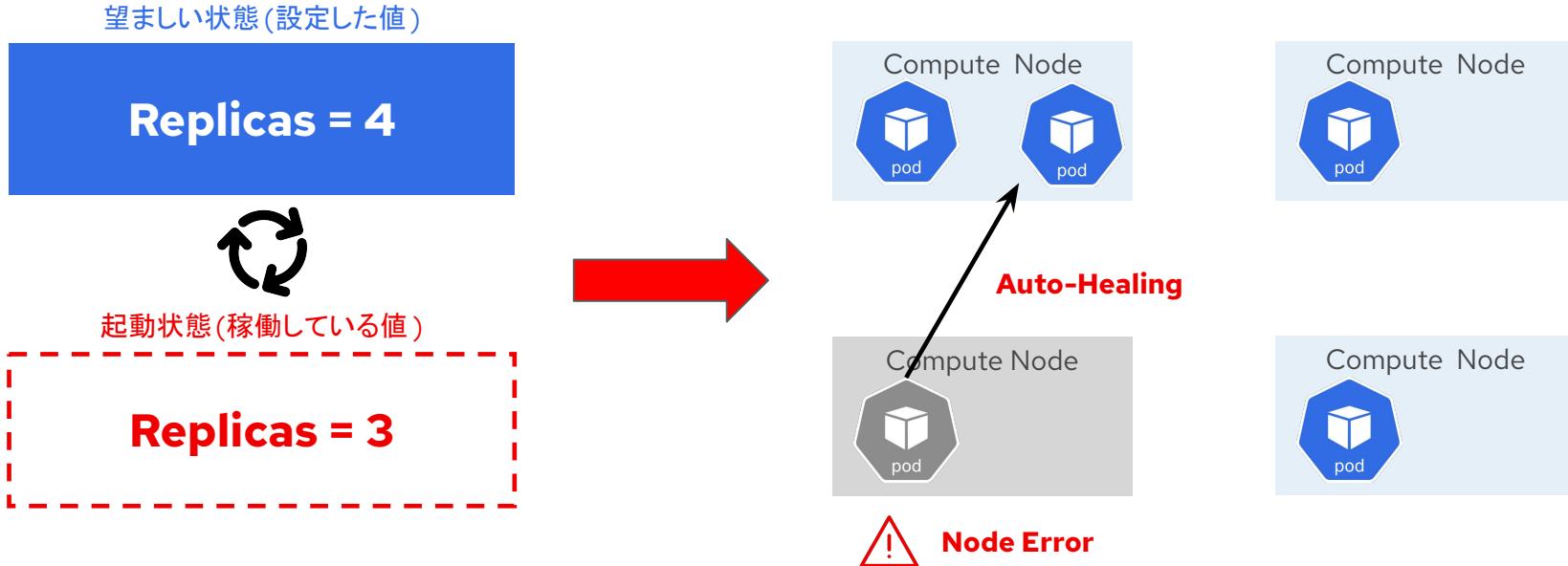
kubernetes

Deploy



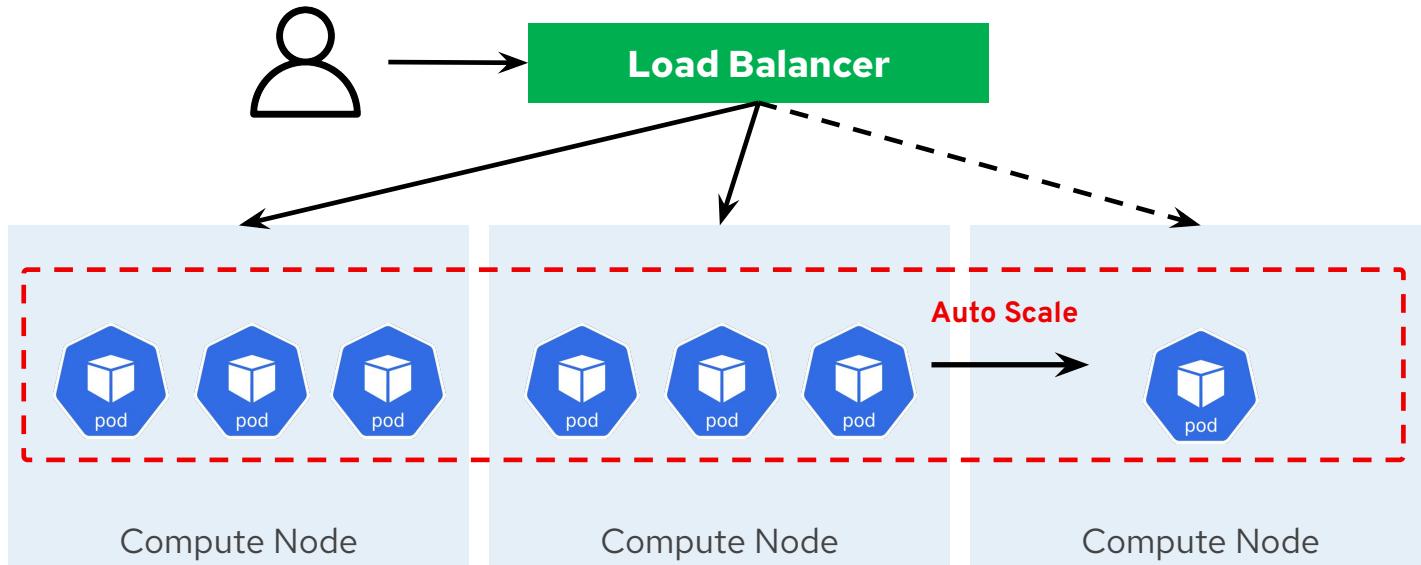
コンテナアプリの冗長性の担保 (セルフヒーリング)

Kubernetesは、現在のアプリケーションが望ましい状態に一致するように動作します。



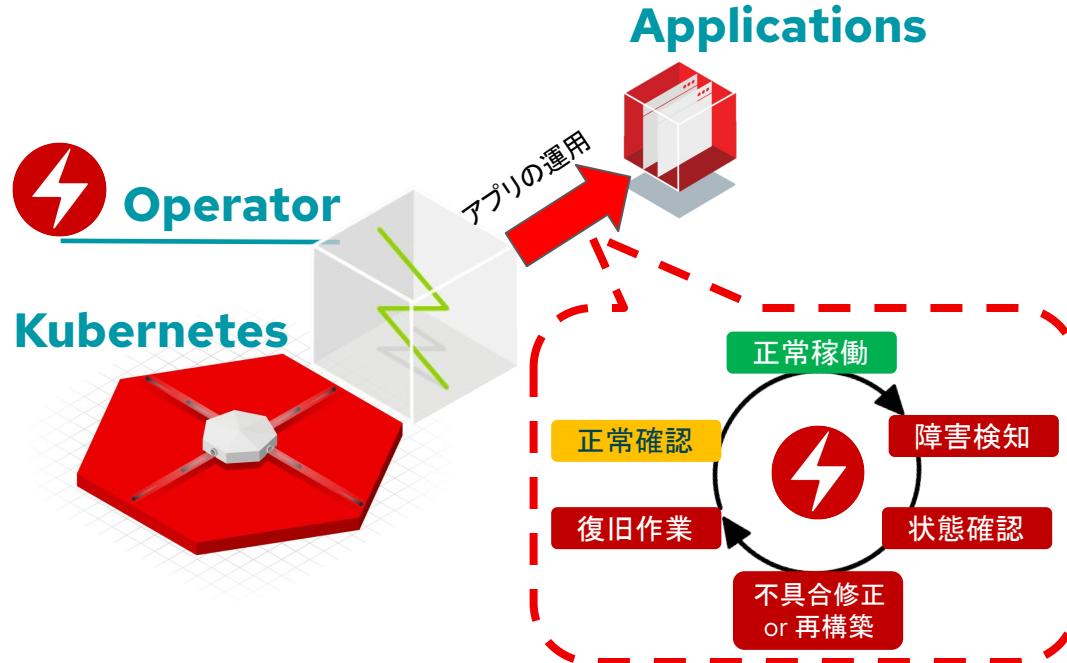
負荷状況に応じた動的なスケールアウト(オートスケーリング)

負荷状況に応じたスケールアウトや、不要なリソースの縮退を、動的に実行できます。



Operator(運用専用コンテナ)で、コンテナアプリの運用自動化

運用の知見やマニュアルをコード化し、ステートフルアプリケーションの運用を自動化



- アプリ運用のマニュアルをコード化及びパッケージングして、アプリの **運用専用コンテナ** として実行
- アプリ運用に必要となる、下記のような作業を自動的に実行
 - インストール
 - コンフィグデプロイ
 - アップデート
 - リソーススケーリング
 - バックアップ、リカバリー
 - モニタリング
 - ロギング
- アプリを利用したシステム構築や、障害復旧などの自動化を支援
- Operatorの開発フレームワークも提供

Kubernetesだけではできないこと

Kubernetesはコンテナの管理、運用に役立つ機能を提供しますが、それ単体だけではできないこともあります。コンテナのビルドやミドルウェアの管理には、Kubernetes以外のツールの連携が必要です。

Kubernetesでは提供されない機能

コンテナの動的
ビルド/デプロイ

ミドルウェア
の管理

クラスタの
ロギングや監視

コンテナの
セキュリティ

クラスタ
アップグレード



kubernetes



Bare metal



Virtual



Private cloud



Public cloud



Edge

Red Hat OpenShift

エンタープライズに求められる機能を Kubernetes に付随し、サポートすることで、ビジネス価値に直結する機能を提供しています。**アプリケーション開発の効率化に重きを置く点** が、インフラ運用の効率化に取り組むことを目的とした Kubernetes 単体利用と大きく異なる点です。



コンテナの動的
ビルド/デプロイ

ミドルウェア
の管理

クラスタの
ロギングや監視

コンテナの
セキュリティ

クラスタ
アップグレード



kubernetes



Bare metal



Virtual



Private cloud



Public cloud



Edge

Red Hat OpenShift の主な機能

OpenShiftで提供されるセルフ開発の利用例

Gitリポジトリのソースコードを利用して、OpenShift環境にNode.jsアプリをデプロイ

1

コンテナの動的
ビルド/デプロイ

Import from git

Git

Git Repo URL *

<https://github.com/jankleinert/concession-kiosk-backend>

› Show Advanced Git Options

Builder

Builder Image *



Builder Image Version *

IST 10

開発者は自身がアプリ開発に利用している
GitリポジトリのURLを指定

Node.jsのバージョンを選択

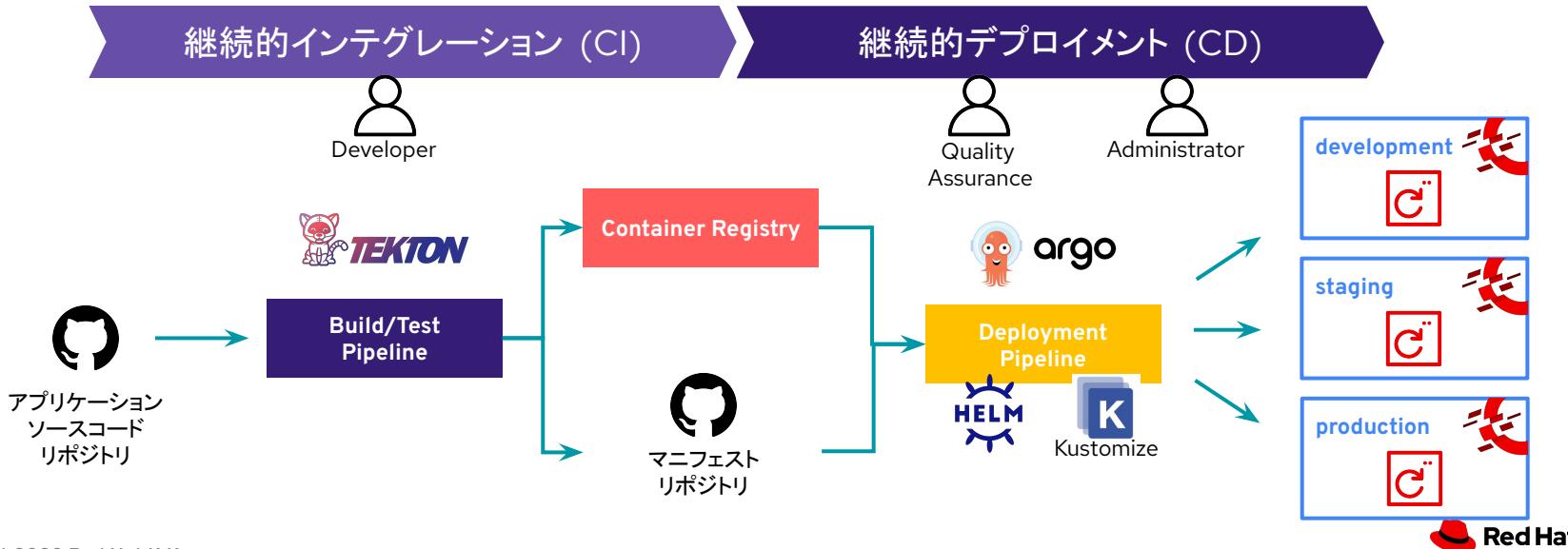
OpenShift上にデプロイするコンテナアプリ用
のビルダーイメージを選択 (この例では
Node.jsアプリをビルト・デプロイするために予
めOpenShiftで用意されている、専用のコンテ
ナイイメージを選択)

コンテナアプリのビルト/テスト/デプロイ の完全自動化

1

コンテナの動的
ビルト/デプロイ

- OpenShiftに統合されたPipelines ([Tekton](#)) を利用可能
 - ソースコードの静的解析/ビルト、単体/インテグレーションテストを実施 (CI部分を担当)
 - サーバレスなアーキテクチャ
- OpenShiftに統合されたGitOps ([Argo CD](#)) を利用可能
 - GitリポジトリをSingle Source of Truthとし、デプロイ先の状態変化を自動検知して、定義された状態を維持
- TektonとArgo CDは、Operatorによって運用を自動化



WebブラウザベースのIDEを提供

様々な開発環境のテンプレートを利用可能 (OpenShift Dev Spaces)

The screenshot shows the Red Hat CodeReady Workspaces interface. On the left, there's a sidebar with navigation links like 'Get Started Page', 'Workspaces', 'RECENT WORKSPACES', and '+ Create Workspace'. The main area is titled 'Getting Started with CodeReady Workspaces' and has a sub-section 'Select a Sample' with a dropdown menu and a 'Filter by' input. Below this, there's a grid of 12 workspace templates:

- JBoss**: Java with JBoss EAP XP 2.0 Bootable Jar
- JBoss**: Java with JBoss EAP XP 2.0 Microprofile
- JBoss**: Java stack with JBoss EAP 7.3
- Red Hat Fuse**: Red Hat Fuse stack with OpenJDK 8, Maven 3.5 and JBoss EAP 7.3
- Tooling for Apache Camel K**: Tooling to develop Integration projects with Apache Camel K
- Java Gradle**: Java stack with OpenJDK 8, Maven 3.6.3, and Gradle 6.1
- Quarkus**: Java Quarkus
- Java Vert.x**: Java stack with OpenJDK 11, Maven 3.6.3 and Vert.x booster
- Java Maven**: Java stack with OpenJDK 11, Maven 3.6.3 and Vert.x demo application
- Java Spring Boot**: Java stack with OpenJDK 8, Maven 3.6.3 and Spring Boot Petclinic demo application
- NodeJS ConfigMap Express**: NodeJS stack with NPM 6.14.6, NodeJS 12.18.4 and ConfigMap Web Application
- NodeJS MongoDB**: NodeJS stack with NPM 6.14.6, NodeJS 12.18.4 and MongoDB
- C++**: C and C++ Developer Tools stack with GCC 8.3.1, cmake 3.11.4 and make 4.2.1
- .NET**: .NET Core SDK 3.1.003, Runtime, C# Language Support and Debugger
- Go**: Stack with Go 1.12.2
- PHP CakePHP**: PHP Stack with PHP 7.3.5, Apache Web Server 2.4.37, Composer 1.8.4 and a quickstart CakePHP application for OpenShift
- PHP-DI**: PHP Stack with PHP 7.3.5, Apache Web Server 2.4.37 and Composer 1.8.4
- Python**: Python Stack with Python 3.9.0

On the right side, there's a large terminal window showing the output of a 'start native' command for a Quarkus application. The terminal output includes logs for Dockerfile build, Java compilation, and the start of a Quarkus application. A preview window on the far right shows a simple web page with the message 'Your new Cloud Native application is ready!'. At the bottom right, there's a Red Hat logo.

OperatorHub.io

<https://operatorhub.io/>

KubernetesのOperatorのカタログを掲載するポータルサイトです。Red HatとMicrosoft、Google、Amazon Web Servicesと共に2019年にローンチしました。



Red Hat Ecosystem Catalog

<https://catalog.redhat.com/software>

Red Hat製品が稼働するハードウェア・ソフトウェアの認定製品を検索できるポータルサイトです。ソフトウェア認定では、サードパーティのOperator認定とコンテナの認定があり、既に多くのソフトウェアが登録されています。



Red Hat Marketplace

<https://marketplace.redhat.com/>

Red Hat認定のOperatorを検索し、購入・デプロイ・管理を容易にするオープンクラウドマーケットプレイスです。Red Hat OpenShift環境で利用できます。

Red Hat Marketplace

2

ミドルウェアの管理

- Red Hat製品だけでなく、様々なパートナーのOperator(数百以上)を掲載
- OpenShiftではOperatorHubという形式で統合されており、Red Hat Marketplaceにある様々なOperatorを簡単にデプロイすることが可能

The screenshot shows the Red Hat Marketplace homepage with the following details:

- Header:** Red Hat Marketplace logo, navigation menu (Solutions, Sell with us, Blog, Docs, Support), search bar, and user options (Log in, Create account).
- Section:** All products (Viewing 214 products).
 - Sort by:** By relevance.
 - Product Grid:** A grid of 20 Operator cards, each with a thumbnail, name, provider, description, and rating.
 - E.D.D.I.** By Lab4ai: Scalable Open Source Chatbot Platform to build multiple Chatbots with NLP.
 - anchore** By Anchore, Inc.: Provides deep container image inspection to generate vulnerability analysis, software dependency analysis, and more.
 - FP-PREDICT[®]** By Findability Sciences: FP-Predict™, an Automated, self learning, and Multi-Cloud Container Vulnerability Analysis Platform.
 - Ivory Service Architect** By GT Software, Inc.: True drag and drop mainframe integration that quickly connects mainframe services to cloud.
 - Modeling Tool** By PerceptLabs, Inc.: A visual modeling tool for machine learning at warp speed.
 - Densify** By Densify: Patented machine learning analytics that enables automated management of complex infrastructure.
 - Red Hat Single Sign-On** By Red Hat: Based on the Keycloak project, Red Hat Single Sign-On enhances security by providing a single sign-on experience across multiple applications.
 - Red Hat JBoss Enterprise Application Platform** By Red Hat: Fully supported in traditional, container, and cloud environments.
 - Jobet DX** By Jobet, Inc.: An open source no-code / low-code application platform.
 - dynatrace** By Dynatrace: Provides software intelligence to simplify enterprise cloud complexity and accelerate innovation.
 - Lightbend** By Lightbend, Inc.: Akka Platform Operator.
 - Anaconda Team Edition** By Anaconda: An enterprise-ready clearinghouse for build artifacts along with Phusion Gigaspaces InsightEdge.
 - GigaSpaces InsightEdge** By GigaSpaces Technologies, Inc.: Runs services and machine learning models in production, at extreme scale and with zero downtime.
 - Hazelcast Jet** By Hazelcast, Inc.: Build fault-tolerant and distributed data pipelines that transform and analyze data in real time.
 - Couchbase Server Enterprise Edition** By Couchbase: Develop with agility and deploy on any cloud, at any scale.

OpenShiftに統合されたロギング

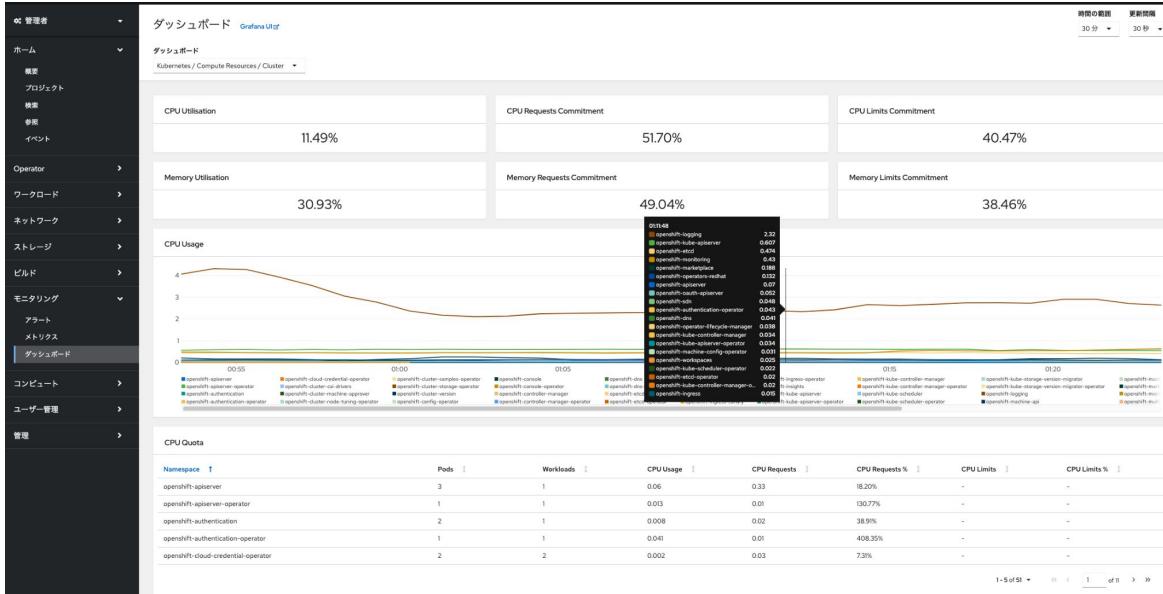
3

クラスタの
ロギングや監視

- ロギングには、LokiStackログストア/Vectorコレクターを利用
 - ElasticsearchログストアとFluentdコレクターは非推奨扱い
- ユーザは自分が利用するプロジェクト上のアプリログ、管理者は全てのシステムログを参照可能
- ログは外部システムへの転送が可能
 - rsyslogを利用するログ集積システム、Apache Kafkaのブローカーなど

OpenShiftに統合されたモニタリング

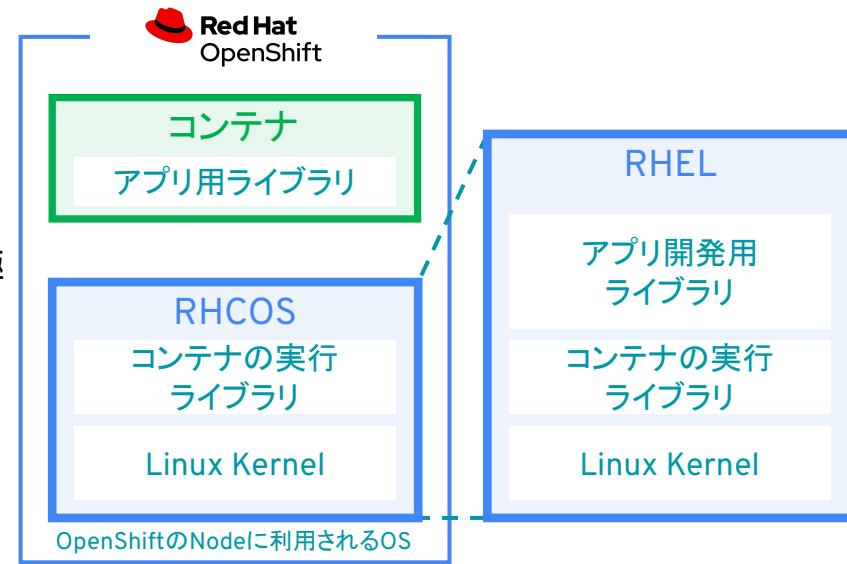
- モニタリングには、Prometheusを利用
- 管理者はOpenShift環境全体のリソース利用量や、API呼び出しのパフォーマンスなどを参照可能
- モニタリングによるアラートは、PagerDuty/Webhook/Email/Slackを利用した通知設定が可能



コンテナ利用に最適化された RHCOS(Red Hat Enterprise Linux CoreOS)を利用することによって、より安全かつ安定したコンテナ環境を提供します。

RHCOSは、RHELのKernelを利用しコンテナ実行に必要なライブラリだけを載せたコンテナ専用軽量OSです。
従来のRHELと同等の利用でサポートされます。

- OpenShiftと連携し、動的なUpgradeをOne-Clickで実現
- ライブラリが少ないため、セキュリティホールを生む可能性が極めて低い
 - そのため、多くのプログラムをOSの中で動かすことができない



ベースイメージの提供

4

コンテナの
セキュリティ

- コンテナベースイメージ(アプリケーションランタイム / SDK)をRed Hat Ecosystem Catalogにて提供
- セキュリティ脆弱性診断にも対応しており、**安心してコンテナイメージを利用可能**

The screenshot shows the Red Hat Ecosystem Catalog interface. On the left, there's a search bar with 'python' and a 'Search' button. Below it are filters for 'Provider' (Couchbase, IBM, New Relic, Red Hat, Inc.) and 'Category' (Application Delivery). The main area displays three container images for Python:

- Red Hat** Python 2.7: Platform for building and running Python 2.7 applications. Updated 20 days ago.
- Red Hat** Python 2.7: Platform for building and running Python 2.7 applications. Updated 21 days ago.
- Red Hat** Python 3.8: Platform for building and running Python 3.8 applications. Updated a month ago.

An orange arrow points to the detailed security analysis of the Python 3.8 image. The detailed view shows a 'Health index' from A to F, where A is green and F is red. The text states: "This image does not have any unapplied Critical or Important security updates. The Container Health Index analysis is based on RPM packages signed and created by Red Hat, and does not grade other software that may be included in a container image." There's also a green checkmark icon and a note: "This image does not contain known unapplied security advisories." On the right, there are sections for 'Release category' (Generally Available), 'Advisory' (RHBA-2021:3113), and 'Privilege mode' (Unprivileged).

OpenShiftのサブスクリプションに 含まれるベースイメージ

4

コンテナの
セキュリティ

「Software Collections(for RHEL7)」および「Application Streams (for RHEL8, RHEL9)」の
コンテナイメージのサポートが OpenShift のサブスクリプションに含まれています。

よく利用されることが多い、開発系の言語やデータベースなどのソフトウェアのバージョンの
サポート提供期間を短くする代わりに、バージョン更新頻度を高くしています。

Red Hat Enterprise Linux 7 Software Collections Product Life Cycle

Note

<https://access.redhat.com/ja/node/4654951>

Red Hat Enterprise Linux 8/9 Application Streams

Note

<https://access.redhat.com/support/policy/updates/rhel-app-streams-life-cycle>

Application Streamsの一部抜粋

Application Stream	Release Date	Retirement Date
mariadb 10.5	May 2021	May 2026
postgresql 13	May 2021	May 2026
python 3.9	May 2021	May 2024
redis 6	May 2021	May 2024
dotnet 5.0	Dec 2020	Jan 2022
nginx 1.18	Nov 2020	Nov 2022
perl 5.30	Nov 2020	Nov 2023
php 7.4	Nov 2020	May 2029

Over The Air (OTA) アップデート

5

クラスタアップグレード

- Cluster Version Operator (CVO) によるOpenShift 環境のアップデートの自動化
- CVOがネットワーク経由で、OpenShiftの有効なアップデート情報をチェックし、管理者に提示
- Web Consoleから簡単にアップデートを実行可能
 - 全てのコントローラ、コンピュートノードを、簡単にアップデート可能 (Self-ManagedのOpenShiftでは、一部のコンピュートノードをアップデート対象から外すことが可能)
 - コンピュートノードで Podが起動している場合、Podの停止→コンピュートノードのアップデートと再起動→Podの起動、を順番に実施
- 下記がアップデート対象
 - ホストOS (RHCOS)
 - OpenShiftのクラスタ管理サービス (Kubernetes, Monitoring, Networkなどを各種 Cluster Operatorによって管理)

クラスター設定

詳細 ClusterOperators グローバル設定



サブスクリプション

[OpenShift Cluster Manager](#)

クラスターID

a129b13a-d986-4ad9-93a9-7cacf186c374

必要なリリースイメージ

quay.io/openshift-release-dev/ocp-release@sha256:d74b1cfa81f8c9cc23336aee72d8ae9c9905e62c4874b071317a078c316f8a70

クラスターバージョンの設定

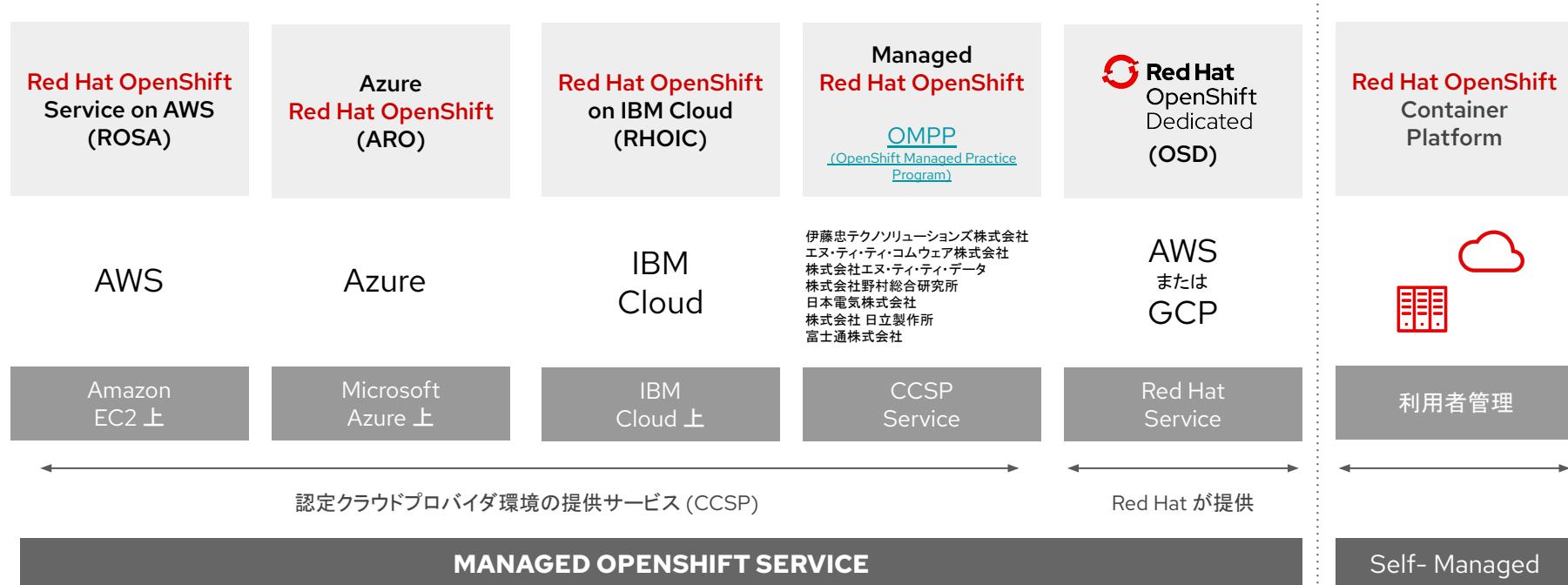
[CV version](#)

Cluster Autoscaler

[Machine Autoscaler](#)

Red Hat OpenShift Everywhere

OpenShiftは、Azure上でも、AWS上でも、Google Cloud上でも、Red Hat OpenStack上でも、VMware上でも、オンプレのベアメタルサーバ上でも、[テス^ト済み](#)なので、場所を選ばずにどこでも同じ知識で運用を回す事ができるというのが大きな特徴です



まとめ

- コンテナは、アプリケーション側に、より大きなビジネスメリットを提供
- コンテナは仮想マシンと同じように見えるが、様々な異なる特徴がある
- コンテナをうまく使うことで、アプリケーションの開発や運用を効率化可能
- 実稼働するシステムでは可用性や運用性が必要となるため、Kubernetesといった、コンテナオーケストレータを使うことが求められる
- Red Hat OpenShiftは、アプリケーション開発を効率化する様々な機能を提供
 - コンテナの動的ビルド/デプロイ
 - ミドルウェアの管理
 - クラスタのロギングや監視
 - コンテナのセキュリティ
 - クラスタアップグレード
- Red Hat OpenShiftは、様々なオンプレ/クラウド環境で実行可能



Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



twitter.com/RedHat