



redhat.

日産自動車株式会社 御中

ASP3 Mode2 Platform

検証ご支援サービス ご提案資料

19/07/2016

Red Hat

Draft版

管理番号 #xxxxx

アジェンダ

- ご支援の目的
- 日産様におけるMode2 Platformの位置付け
- 当プロジェクトのゴール
- 想定するユースケースに対する検証方針
- 開発側の課題について
- 体制案
- 会議体(案)
- 検証全体スケジュール(想定)
- 検証のロードマップ
- 検証の進め方
 - Phase1
 - Phase2
 - Phase3(参考)
 - Phase4(参考)
- 工数の見積りと作業報告
- 前提事項
- Appendix

ご支援の目的

- 日産様次期共通基盤(ASP3)における、新しいアプリケーションアーキテクチャ、開発プロセスの標準化のため、Linuxコンテナを活用した開発実行基盤の検討を、Red Hat OpenShift EnterpriseをベースにしたPoCを通じてご支援します

日産様におけるMode2 Platformの位置付け

日産様資料(20160610_mode2_redhat.pptx)より抜粋:

- 2017年末でCP2がEOSとなるため、後継となる次世代共通プラットフォームとしてASP3の検討に着手
- FY16にAlliance検討を行い、Bi-modalコンセプトを適用すること、および各ModeのS/W、H/W仕様を確認した
- 今年度、NMLでは上記を踏まえ、ASP3の基本構想を進め、2017年度に構築を行う予定
- CP2アプリの受け皿として、現行アーキを継承したMode1 Platformを用意するとともに、今後DevOpsの拡大に向け、Mode2 Platformをサービス提供する
- 一方で、FY16年度、IB2 CoE活動のなかで、次世代生産管理システムのアーキテクチャ策定を行い、今年度意向順次アプリ再構築に着手する計画がある
- Mode2 Platform のArchitecture、および仕様策定については、上記IB2アプリをClientとして、Pilotを行うこと形で進める予定

当プロジェクトのゴール

- ASP3において、既存システムからの移行時に、従来のアプリケーションアーキテクチャを採用するか、コンテナによるマイクロサービスアーキテクチャを採用するかをプロジェクト側が選べるようにすべくITサービスを準備する必要があり、そのため、OpenShift を含めたマイクロサービスのためのシステムを採用し、ASP3に組み込んでいけるかどうか、インフラおよびの両面からパイロットを通じて評価、判断をする
- ここでいうサービスとは、以下を意味する
 - インフラ基盤
 - 標準化された開発ツール
 - 標準化された開発手法

想定するユースケースに対する検証方針

- マイクロアプリケーション開発基盤
 - Docker/OpenShiftとCI/CDツールを連携し、開発、QA環境における開発フローの自動化、効率化、高品質化を図る
- マイクロサービスアプリケーション実行基盤
 - 上記に加え、Production環境へのアプリケーションのデプロイ、サービス提供の基盤としても活用する

上記ユースケースがどこまで適用可能、かつ利用するか否かについては、パイロットののち判断をする。当プロジェクトとしては上記いずれも適用可能かどうかの検討を行う

開発側の課題について

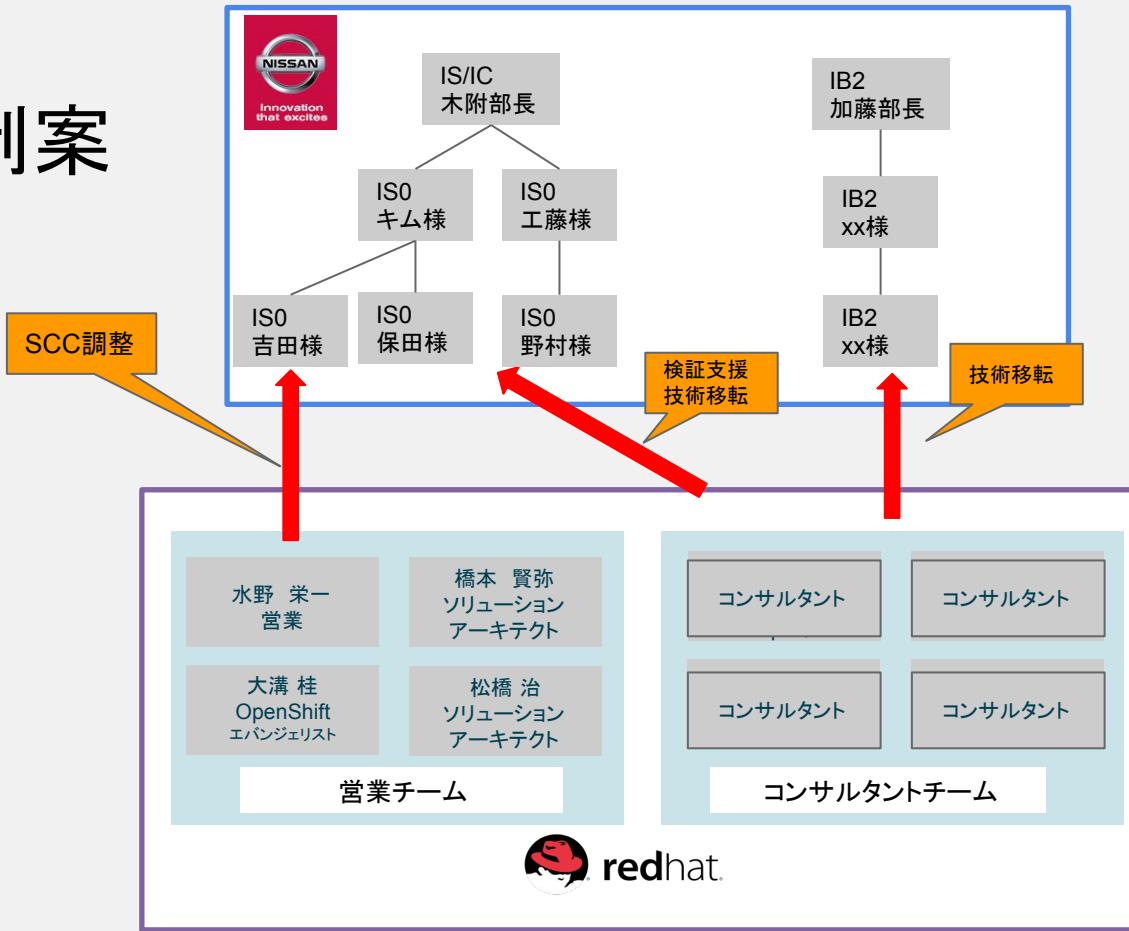
日産様資料(20160610_mc)

当フェーズにおいては、1に焦点を当てる。2についてはAuto Scalingの検証は実施せず、Planned Scalingについてのみ検討する

Container
DevTools(CI/CD)
OpenShift

	Target Issue	Goal(M/W)	Expected Infra	Cost
1	<ul style="list-style-type: none">・変更対応に時間がかかりすぎている・ダウンタイムが発生している	<ul style="list-style-type: none">・MicroService単位で、開発/テスト/Deployができるようになっている事・ダウンタイムなしにDeployができる事	Container	TBD
2	<ul style="list-style-type: none">・業務量が増加すると、システム処理が遅延する・突発なイベント毎の度に環境準備・増強が必要	<ul style="list-style-type: none">・処理量や負荷状態に応じてAuto Scalingができる事	<ul style="list-style-type: none">・Container・Programmable platform・API Management	TBD

体制案



会議体(案)

- 定例会 weekly @xxxxx(開催日について要確定)
 - 目的: プロジェクトの進捗、課題の整理
- 技術検討会 ad-hoc or bi-daily @ xxx(開催日について要確定)
 - 目的: 技術的設計や課題の議論
- オンラインコミュニケーションad-hoc @ remote
 - メール
 - 電話会議の検討
- その他

検証全体スケジュール(想定)



今回のご提案範囲

DevOps Workshop

- kaisai

検証のロードマップ

今回のご提案範囲

	Phase1	Phase2	Phase3	Phase4
目的	App開発検証	運用検証	パイロットプロジェクトへの適用	PaaS基盤提供
実施内容	検証環境構築 App開発標準化・自動化検証 Docker/OpenShift技術習得	運用検証 SLAの定義 APPデプロイ方式検証	パイロット用本番環境構築 ガイドライン策定 テンプレート作成 App開発者向け Docker/OpenShiftの教育	App開発者向け Docker/OpenShiftの教育
期間	1.25ヶ月	1~2ヶ月	4ヶ月	
Red Hat コンサル ティング	環境構築支援 スキルransファー Q&A	スキルransファー Q&A	環境構築支援 スキルransファー Q&A	Q&A
Red Hat GLS	DO-280 OpenShift Administration			

検証の進め方

Step1. 現状理解

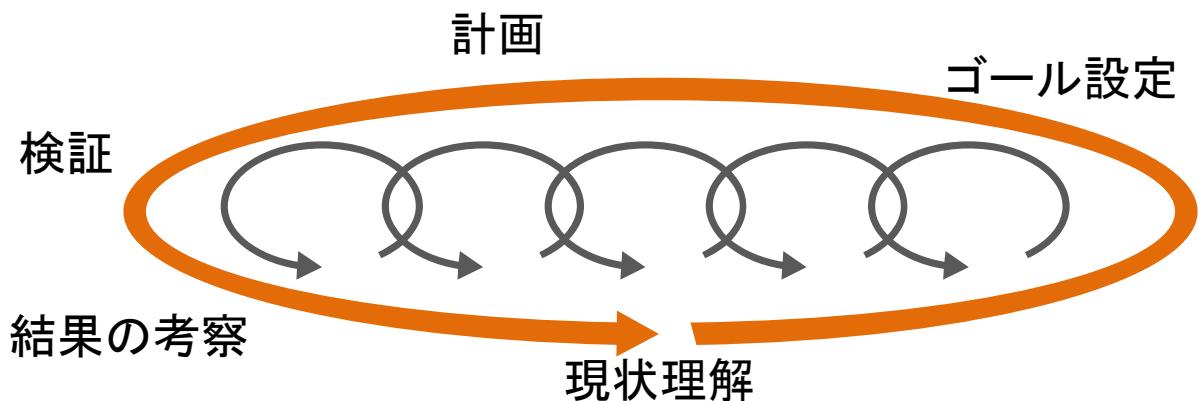
Step2. ゴール設定(今フェーズにおけるゴール、最終的なゴール)

Step3. 計画(今フェーズでの作業の具体化)

Step4. 検証

Step5. 結果の評価と考察

Step 6. ロールアウト計画の作成



Phase1 App開発検証

- 目的
 - OpenShift Enterprise 3 を活用した、アプリケーション開発の標準化及び自動化の実現可能性の確認
- PoCの実施
 - OpenShiftの検証環境の構築及び、日産様標準環境想定であるJavaを使用し、JBoss EAP/Docker/OpenShift を活用したアプリケーション開発を検証します。
 - レッドハットの必要と考えるツールをベースに、日産様の使用中/使用予定ツールとの突合せを実施し、**目的を果たすため必要最低限のツール群を使用します**
- Exit Criteria
 - OpenShift 3およびレッドハットが想定する標準開発支援ツールを使用した、基本的な開発 /テスト/ デプロイ環境の効率化、自動化の仕組みの動作がサンプルアプリケーションを使用して確認され、日産様とレッドハットが事前に合意した検証項目をクリアすることとします

作業項目

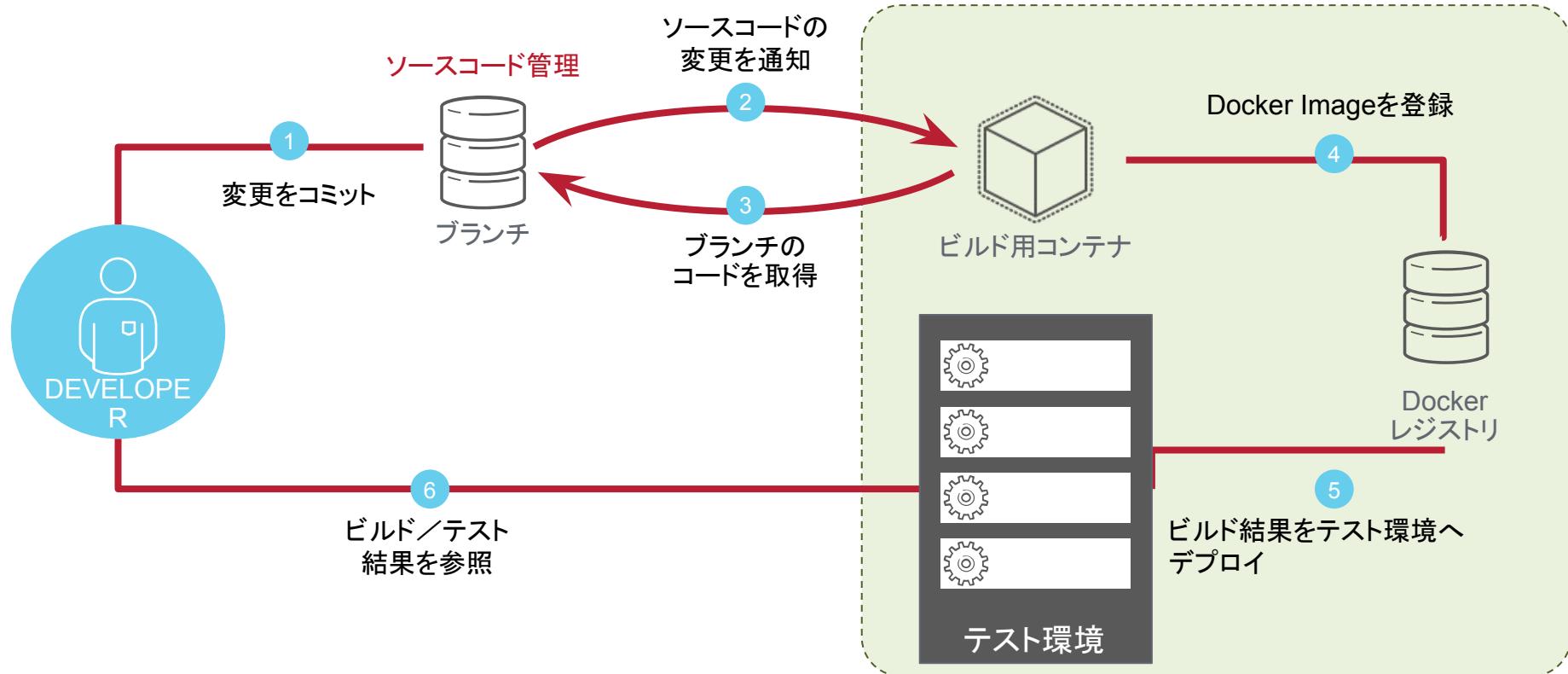
- OpenShift 検証環境構築支援
 - 環境ヒアリング
 - インストール(Master x 1, Node x 2,)
- サンプルアプリケーションデプロイ支援
 - テンプレート作成
 - 繙続的インテグレーション実現方式の定義
- 開発者向け導入ハンズオン
- 技術支援
 - 検証項目の策定支援
 - スキルトランスファー
 - Q&A

Phase1 App開発検証

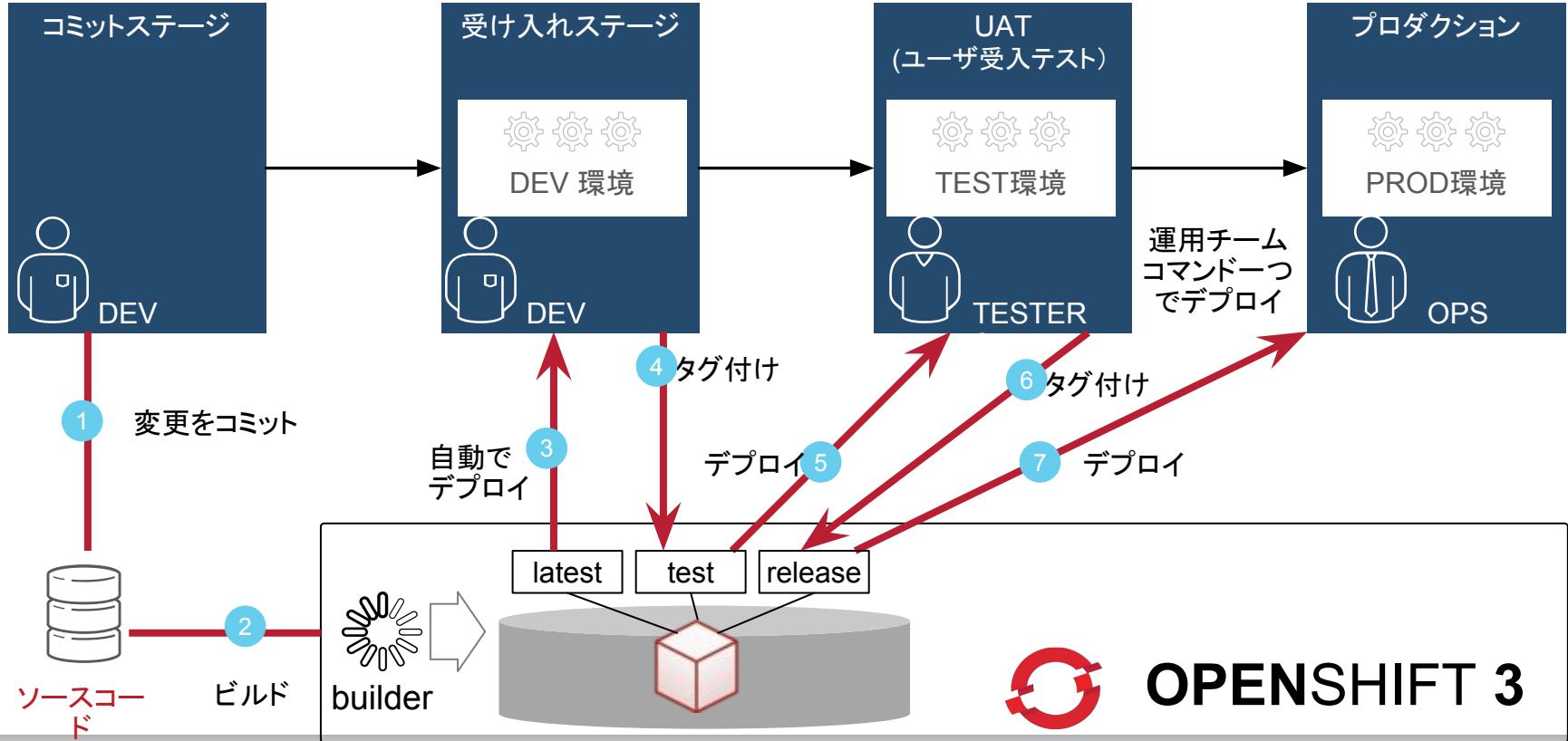
App開発検証項目概要(例) (今後プロジェクト内で精査)

No.	項目	概要	検証方法	Criteria	備考
1	CI	サンプルアプリケーションを用い、開発支援ツールと連携した基本的自動ビルト/テスト/デプロイプロセスの実装/実施	ツール(git,jenkins,and more)とOpenShiftを使用したCI/CDフローの構築	動作すること 懸念事項をリストできること	
2	CD	B/G Deployなど、サービスに影響を与えないデブロイ手法の確認	OpenShiftに用意された機能によるテストを実施	動作すること	
3	Management	ソースコードバージョン管理、Dockerイメージ管理、リリース管理、権限管理	各ツールとOpenShiftのどれが担うのかを確認	対象が網羅できること	
4	Security	アプリ/コンテナセキュリティについて、CI/CDフロー内で自動チェックが可能か	手段を調査検討、実装、テスト	チェックを自動化できるもの/できないものが明らかになると	
5	App Arch	RDB接続構成(OpenShift外)のアプリを想定した検証	OpenShiftの外に準備したRDBとの接続するためのサンプルアプリでのテスト	方法論について確認、実証できること	phase3
6	Container	テンプレート標準化と個別構成/設定の流し込み。	DockerとOpenShiftでの手法について、確認、テストを実施	方法論について確認、実証できること	
N	App Arch	既存アプリの並列処理化によるパフォーマンス向上の有無			phase3

参考イメージ: 自動化継続的インテグレーションの実現



参考イメージ: 自動化:CI/CDフロー



Phase2: 運用検証

- 目的
 - OpenShift Enterprise 3 を活用した、アプリケーション開発基盤またはサービス提供基盤の運用機能を確認し、運用設計の足がかりとします。
 - PoCの実施
 - 検証にあたっては、App開発検証にて利用した環境及びアプリケーションを利用します。
 - **OpenShiftの冗長構成の検証を実施する場合は、環境構築を新たに行うため、Phase1で最初から冗長構成での構築をするかどうか要確認**
 - 本フェーズは、OpenShiftとしてあるべき冗長化構成に基づき評価を行います。Phase2の結果を元に、Phase 4移行にて、本来の本番構成の設計が行われることを想定しています。
 - Goal/Exit Criteria
 - 適用対象業務に対する想定 SLAを元にした、OpenShift環境の構成が確認できる
 - 障害発生時動作を確認し、問題判別に取りかかる手法を定義する
- 作業項目
- 運用者向け導入ハンズオン
 - 技術支援
 - 検証項目の策定支援
 - Q&A

Phase2 運用検証

運用検証項目概要(例) (今後プロジェクト内で精査)

No.	項目	概要	検証方法	Criteria	備考
1	OpenShift Service	OpenShift自身の起動や停止方法について	手法について整理、実施する	実施方法が確認され、動作すること	
2	Maintenance	OpenShift 修正の適用方法について確認 (Rolling Updateなどの有無、手法)	手法について整理、実施する	実施方法が確認され、動作すること	
3	Architecture	OpenShiftの冗長化設計 , Persistent Storage設計	整理、実施する	実施方法が確認され、動作すること	
4	Monitoring /Trouble	OpenShiftの障害時動作、リカバリ方法	手法について整理、実施する	実施方法が確認され、動作すること	
5	Self Service/Auth	OpenShiftの認証、セルフサービス、テナントマネジメントについて、できること、できないことを確認	手法について整理、実施する	実施方法が確認され、動作すること	
6	Security	OS, Docker, Kubernetes, OpenShiftそれぞれにおけるHardning手法/選択肢の有無確認	手法について整理、実施する	方法論について確認、できること	
N	Scaling	OpenShiftのノード追加と、追加後のサービス展開方法について	手法について整理、実施する	実施方法が確認され、動作すること	

参考: Phase3 パイロットプロジェクトへの適用

目的

OpenShift Enterprise 3 を活用したパイロットプロジェクトへの適用、及び、パイロットプロジェクトからのフィードバックに対する改善策の策定をご支援します。

パイロットプロジェクトは PCS/CATS アプリケーションを想定しています。パイロット用アプリケーション自身の Containerization については別途事前に状況を確認し支援の必要性を検討いただきます

今回の構築は QA 環境を想定しており、真の本番環境 (ASP3) の環境構築計画については別途ディスカッションをさせていただきます。Phase1/2で使用した環境をパイロット環境として使うことを想定するため、環境構築の部分は optional になる可能性があります

作業項目

- OpenShift 本番環境(パイロット用)構築支援
 - 環境ヒアリング
 - 環境構築
- アプリケーションデプロイ支援
 - テンプレート作成
- 開発者向け導入ハンズオン
- 技術支援
 - スキルransファー
 - Q&A

参考: 日産様

参考: Phase4 PaaS基盤提供

目的

OpenShift Enterprise 3 のPaaS基盤活用時のQ&Aを実施致します。

作業項目

- Q&A

App開発検証 - 作業詳細

Category	Description	Duration	Deliverables
環境構築支援	OpenShiftのインストール環境をヒアリングし、基本的な構成での環境を構築します。	5.0日	
サンプルアプリケーションデプロイ支援	サンプルの構成をヒアリングし、サンプルテンプレートを作成致します。	3.0日	サンプルテンプレート
開発者向けハンズオントレーニング	OpenShiftを利用したアプリケーション開発をハンズオン形式で実施します。	1.0日	ハンズオン資料
技術支援	アプリケーション開発の標準化及び自動化を実現するための検証項目の策定支援、Q&Aを実施致します。	11.0日	技術メモ Q&A票
合計		20.0人日	

サービス提供にあたり、コンサルタントからの適切な技術移転を実施するために、担当者の方に OpenShift Enterprise Administrationの研修(DO280 Open Shift Enterprise Administration 3日コース)を受講いただくことを強くお願い致します。

運用検証 - 作業詳細

Category	Description	Duration	Deliverables
運用者向け導入ハンズオン	OpenShiftを利用したApp開発及びOpenShiftの設定などをハンズオン形式で実施します。	2.0日	ハンズオン資料
技術支援	検証項目の策定支援、 Q&A を実施致します。	18.0日	Q&A票
合計		20.0人日	

サービス提供にあたり、コンサルタントからの適切な技術移転を実施するために、担当者の方に OpenShift Enterprise Administrationの研修(DO280 Open Shift Enterprise Administration 3日コース)を受講いただくことを強くお願い致します。

パイロットプロジェクトへの適用 - 作業詳細

Category	Description	Duration	Deliverables
環境構築支援	OpenShiftのインストール環境をヒアリングし、環境を構築します。	5.0日	
アプリケーションデプロイ支援	アプリケーションデプロイに必要なテンプレート作成を支援します。	5.0日	サンプルテンプレート
開発者向けハンズオントレーニング	OpenShiftを利用したアプリケーション開発をハンズオン形式で実施します。	1.0日	ハンズオン資料
技術支援	パイロットプロジェクトからのフィードバックへの対応策の検討、Q&Aを実施します。	9.0日	技術メモ Q&A票
合計		20.0人日	

PaaS基盤提供- 作業詳細

Category	Description	Duration	Deliverables
Q&A	リモート及びオンサイトでのQ&Aを実施します。	10.0日	技術メモ Q&A票
合計		10.0人日	

工数の見積りと作業報告

- 期間

- 2016/09/01 - 2016/11/30(想定: 契約期間とは別)
- 第2フェーズの内容/期間は2016/09に再度確認

- 時間

- 第1フェーズ: 160時間
- 第2フェーズ: 160時間
- 参考: 第3フェーズ: 160時間 (今後精査)
- 参考: 第4フェーズ: 80時間 (今後精査)

- 作業報告形式

- 月次でRHコンサルタントプロジェクトチームにご報告(報告書の提出)

前提事項

- 本契約はタイムアンドマテリアル方式による準委任契約です
- レッドハットはお客様にナレッジを提供するためのドキュメントを、必要に応じ作成しますが、お客様社内のプロジェクト成果物としてドキュメントは作成しません
- 作業場所はレッドハットオフィス(恵比寿)または日産様NGIC(本厚木)とします
- 稼働時間は平日09:00-18:00とします。休日及び時間外は2倍の工数を消費します
- その他コンサルタントの作業に関する前提是契約書に準じます

検証環境構築における確認/前提事項

- OpenShift環境 ネットワーク接続
 - OpenShiftのインストール及びテストにおいては、インターネット接続が必須となります。作業前にコンサルタントの会話のうえ、ご準備をお願い致します
- 検証環境のインフラ設計について
 - OpenShiftの冗長化構成について、Phase1検証環境とPhase2検証環境を別に構築するかどうかについての相談をさせていただきます。
- OpenShiftの前提スキル構築について
 - サービス提供にあたり、コンサルタントからの適切な技術移転を実施するために、担当者の方にOpenShift Enterprise Administrationの研修(DO280 Open Shift Enterprise Administration 3日コース)を受講いただくことを強くお願い致します
 - i. 2016/08/08 ~ 2016/08/10 (品川御殿山会場)
 - ii. 2016/09/26 ~ 2016/09/28 (品川御殿山会場)

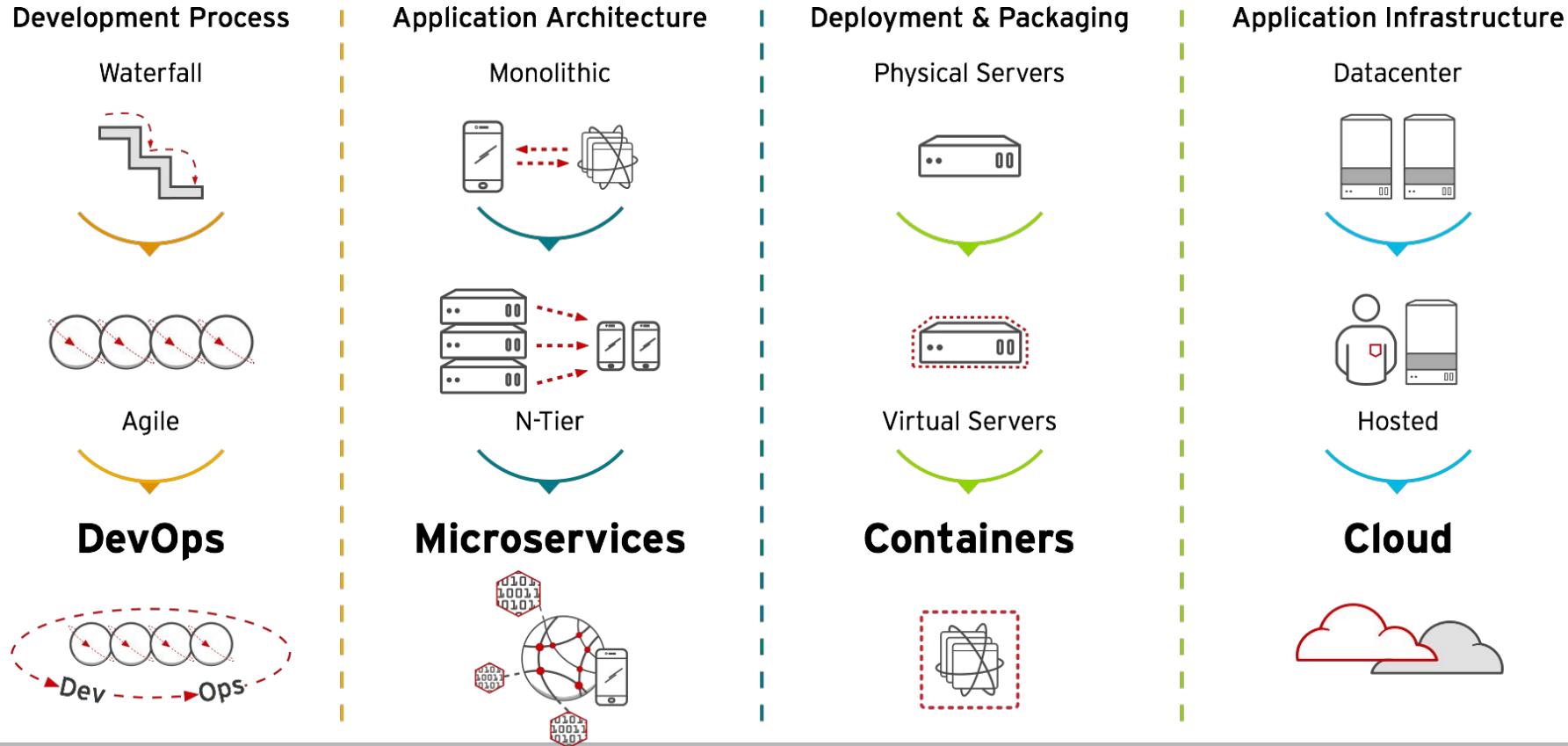
<https://www.redhat.com/ja/services/training/do280-openshift-enterprise-administration>

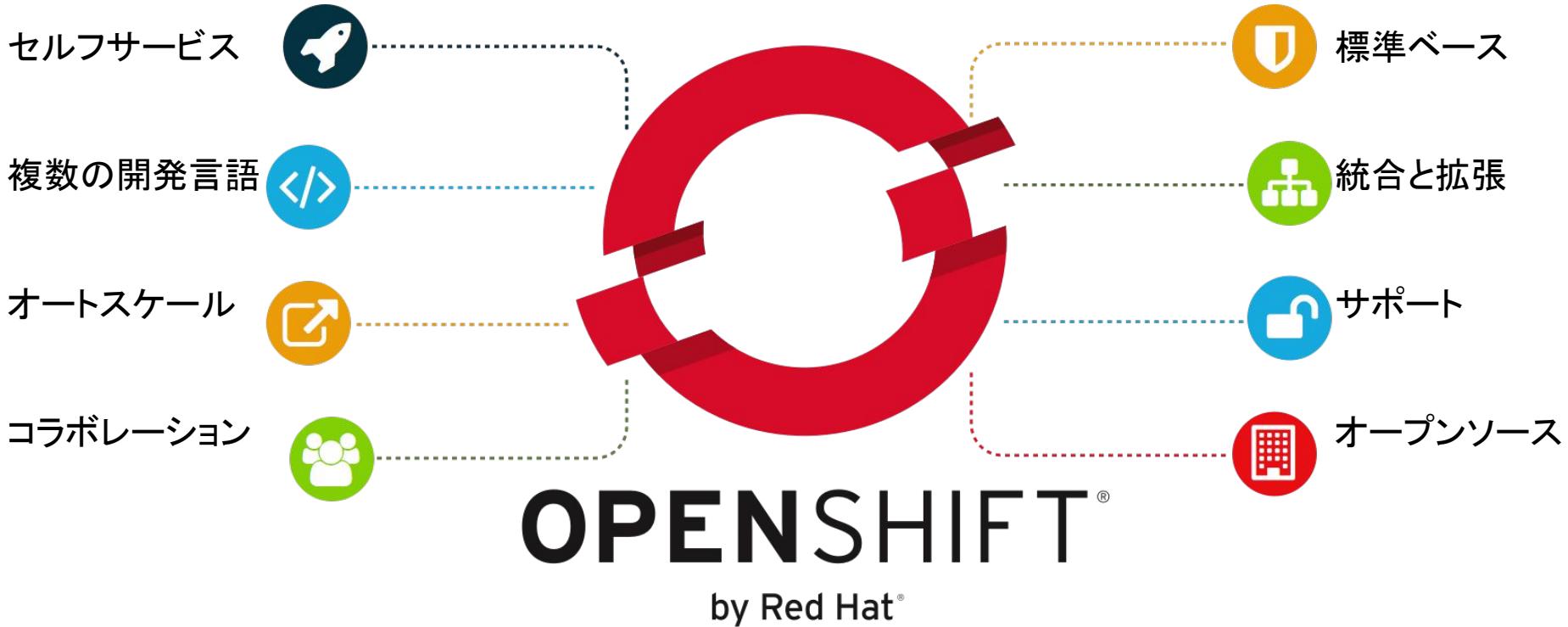


redhat.

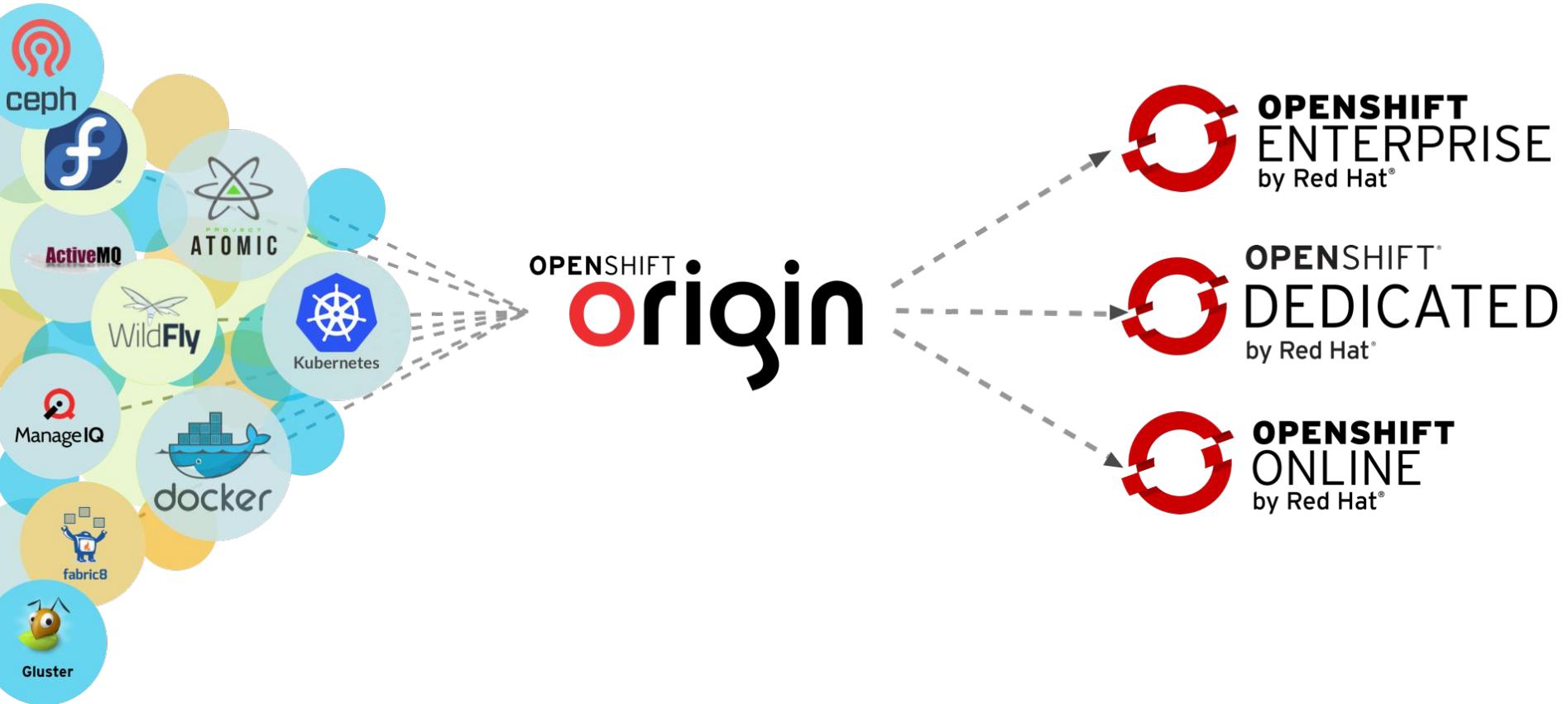
Appendix

必要なのはビジネスの要求に応えられる情報システム

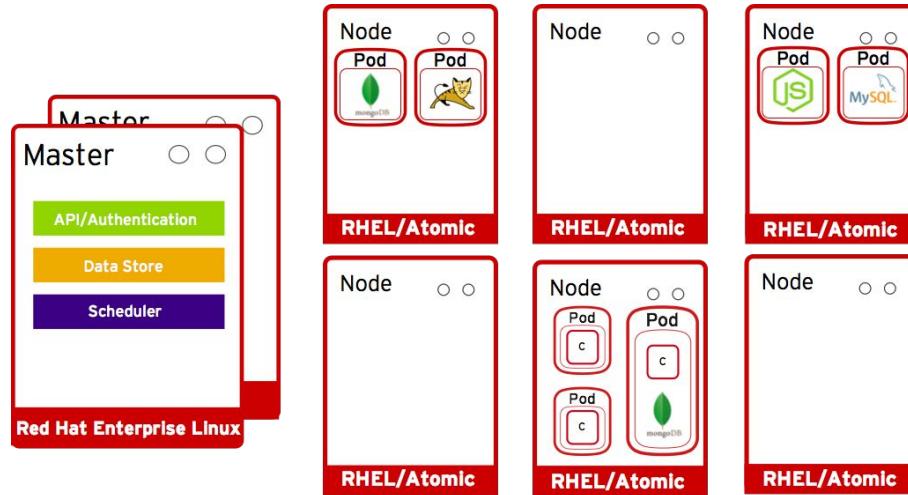




Community Powered Innovation

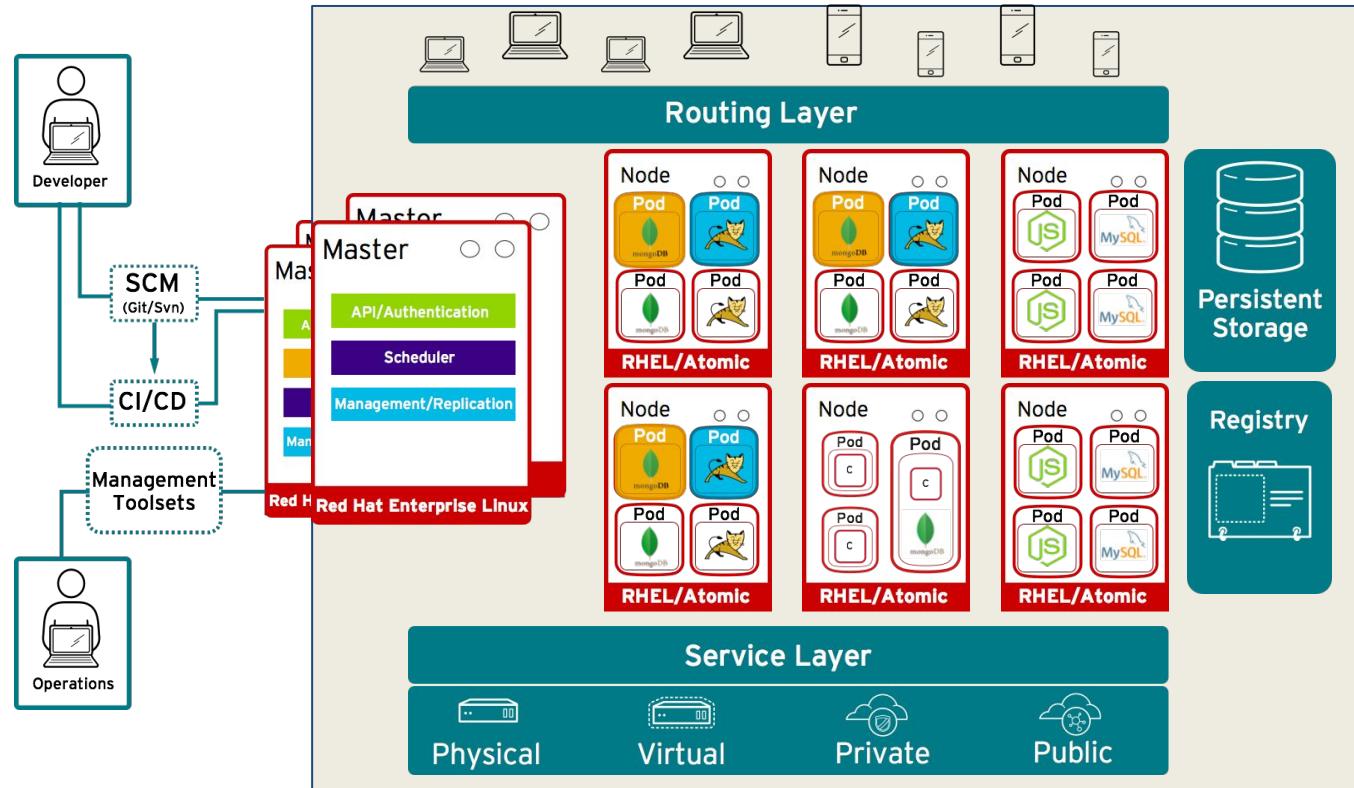


Docker & Kubernetes

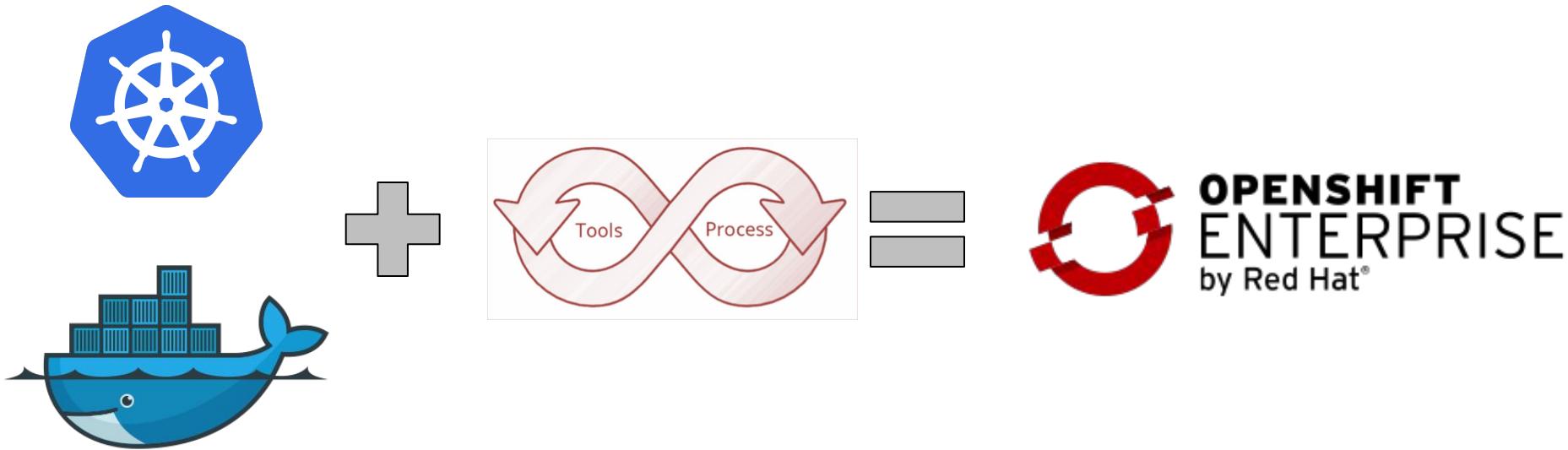


OpenShift Enterprise 3

- DevOps**
- ✓ SCMとの連携
 - ✓ APPのライフサイクル管理
 - ✓ ロギング
 - ✓ メトリクス取得
 - チームでの利用**
 - ✓ ユーザ管理



Docker, Kubernetes だけでは実現できない価値を提供



OpenShift Enterprise 3 の構成



OPENSIFT 3

OpenShift Web UI

様々なPaaSサービス
複数開発言語、JBoss, DB, CI/CD, DevOps

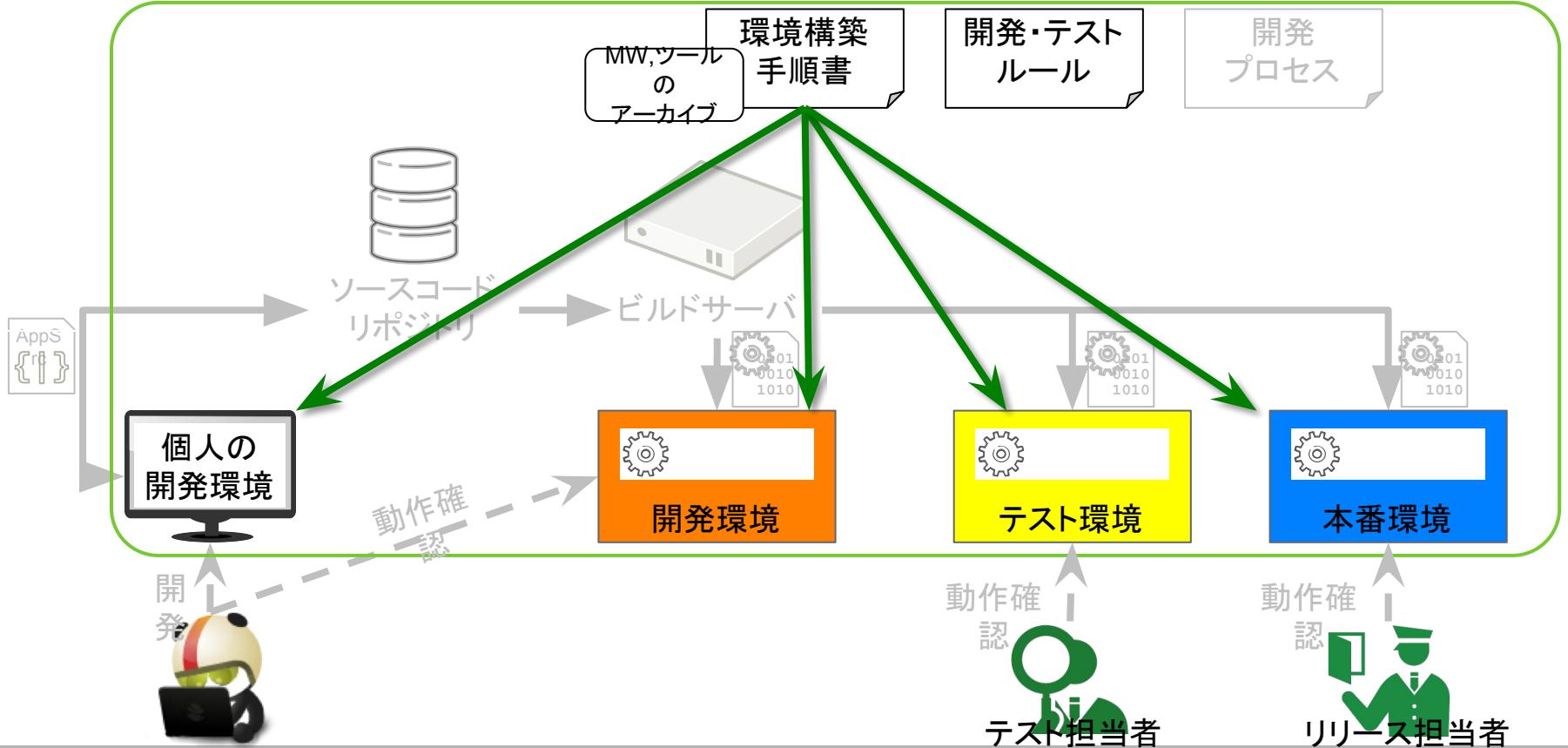
コンテナ管理 & オーケストレーション
Kubernetes

コンテナ API
Docker

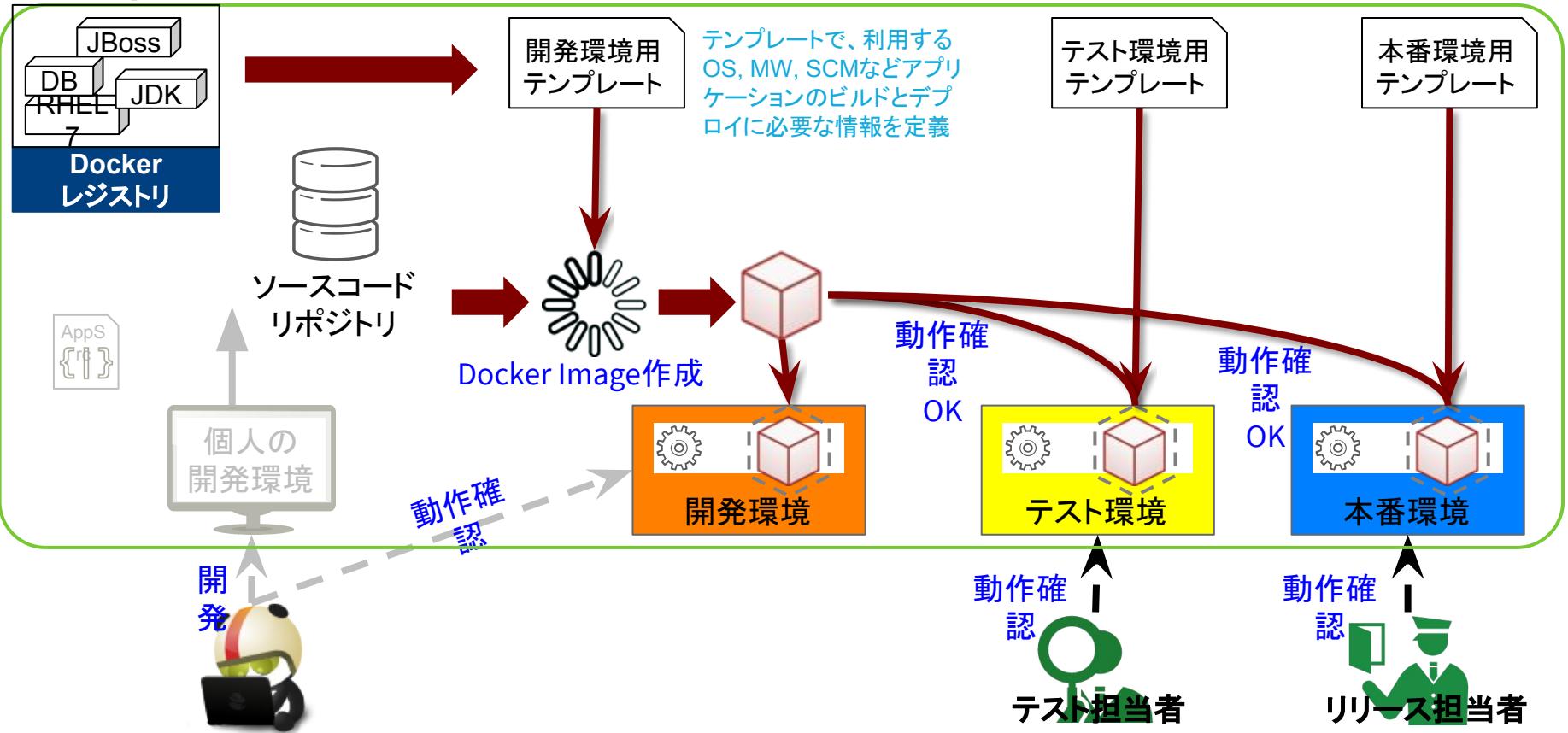
コンテナOS
RHEL 7.1+

- ユーザエクスペリエンス
 - Web UI, CLI, REST API
 - IDE連携
- 開発の効率化
 - プログラミング言語
 - ミドルウェア
 - DevOpsを促進させる機能
- 環境の標準化
 - 豊富なテンプレート
- 運用の利便性向上
 - de fact standardな技術
 - Docker
 - Kubernetes
- 安心の実行環境
 - Red Hat Enterprise Linux 7

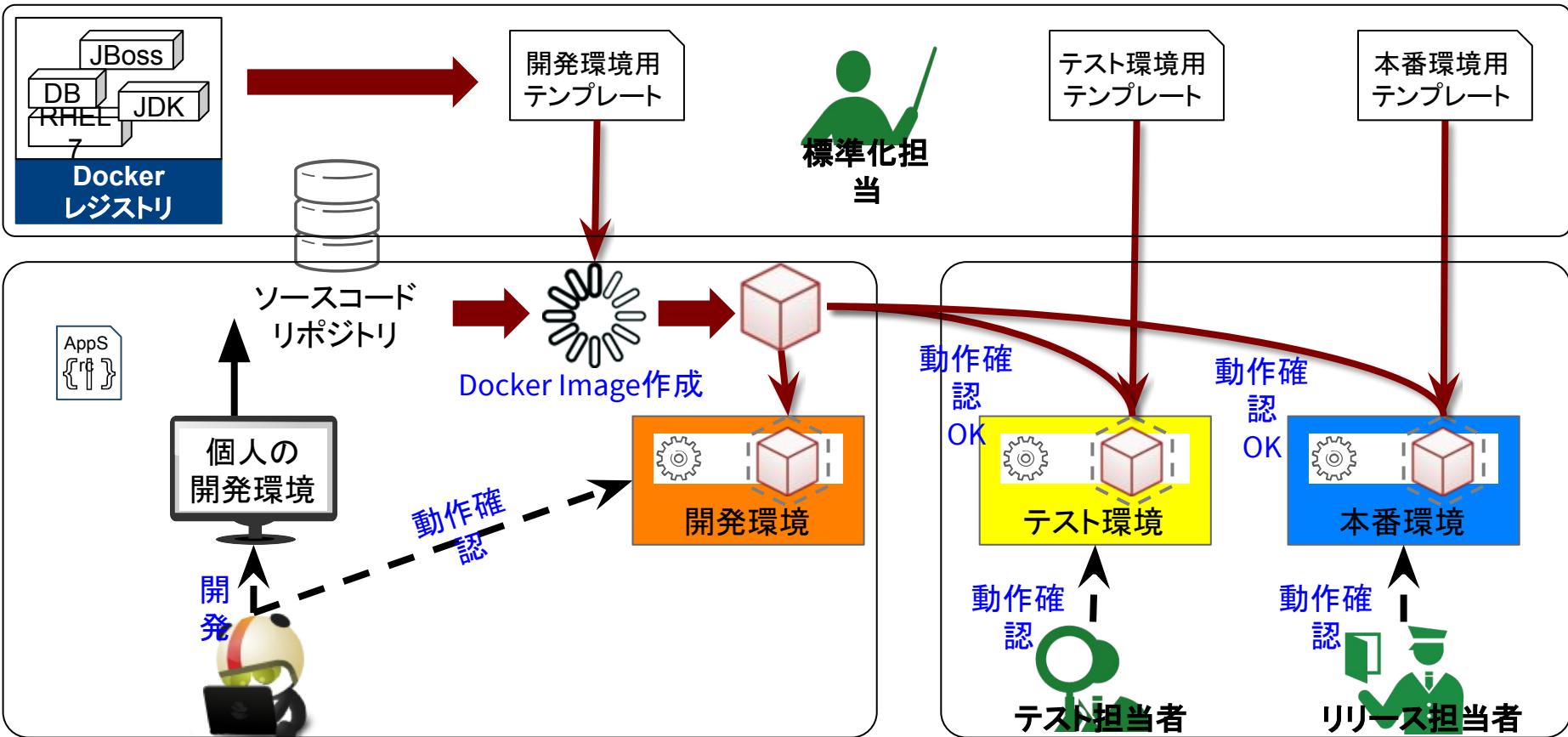
従来の技術の標準化



OpenShiftで実現する標準化



OpenShiftを活用する場合の役割分担



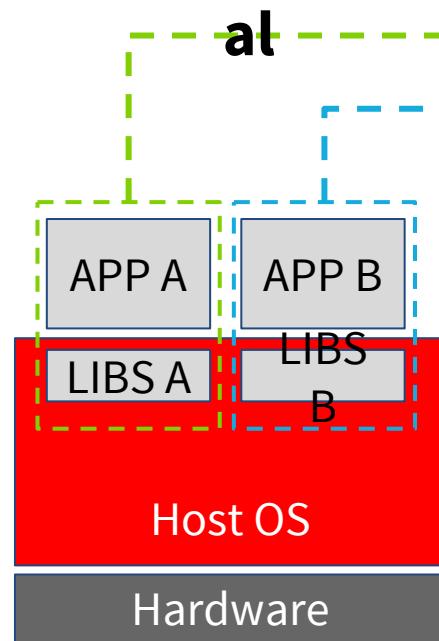
OpenShiftのメリット

- コンテナ型アプリケーション
- 標準化
- UX、セルフサービス
- 自動化
- 効率化

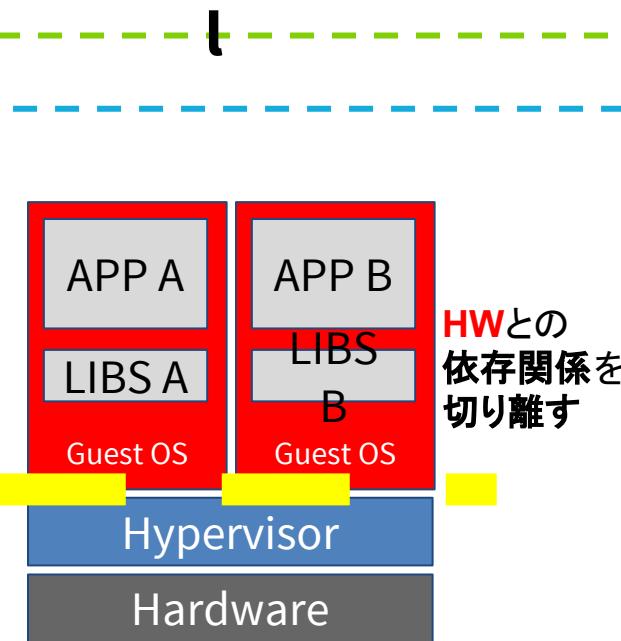
コンテナ:アプリケーションの仮想化

OSとの依存関係を切り離す
→ リソース利用の効率化

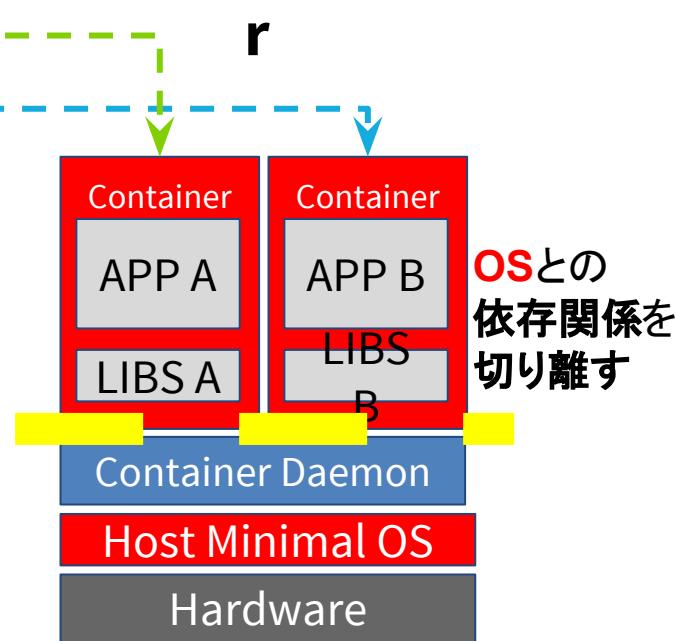
Tradition



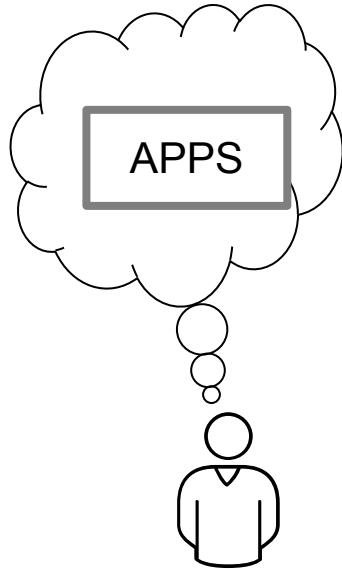
Virtua



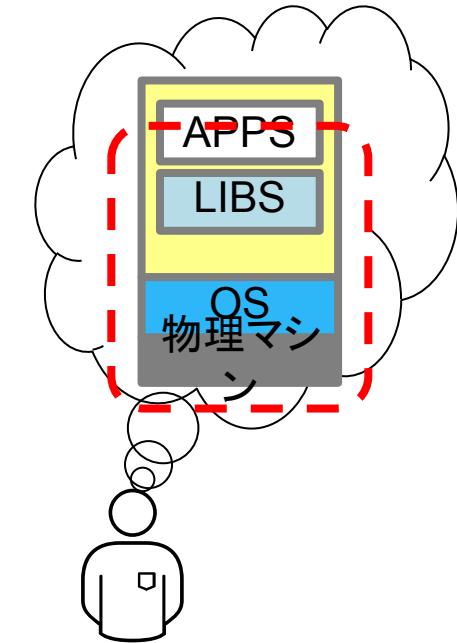
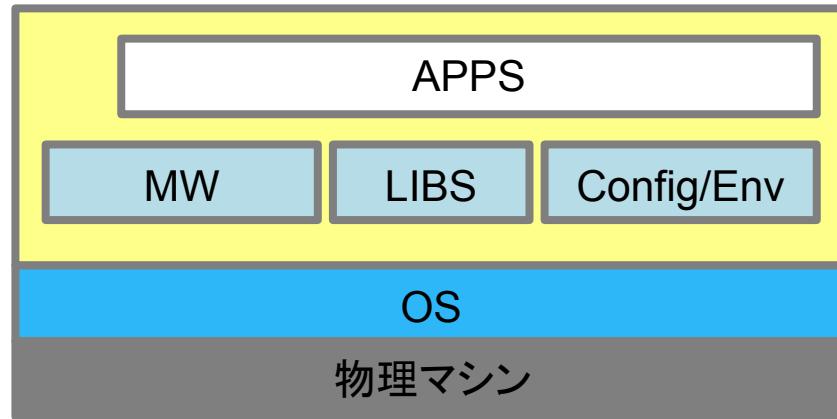
Contain



コンテナ化されていない Dev と Ops

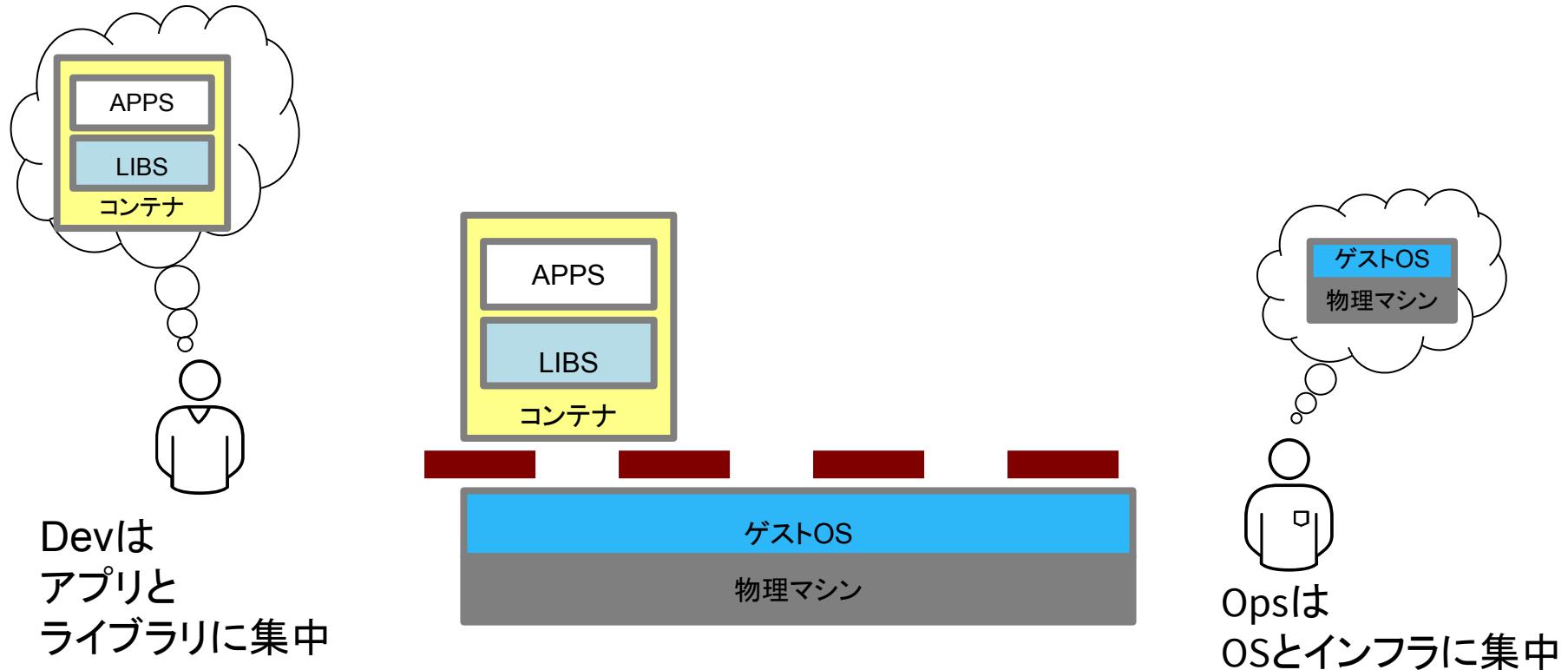


DevはAppはインフラ
には関与しない



Opsは基盤からAppが必要とする設定まで気にする必要がある

コンテナとDevOps



アプリケーションをコンテナ化すると



アプリケーションの
配布とデプロイ
が簡単

コンテナ化したアプリケーションのメリット

Step 1

Package

課題:
インストールや設定は複雑で時間がかかる
人為的ミスの発生
手順化(手順書やスクリプト)しても、陳腐化が早い

メリット:
複製が簡単
設定ミスがなくなる

Step 2

Share

課題:
環境毎に異なるパッケージング

メリット:
実行環境そのものがパッケージングされるので、開発、QA、運用チームで配布が簡単
新しいアプリケーションの検証がすぐに始められる

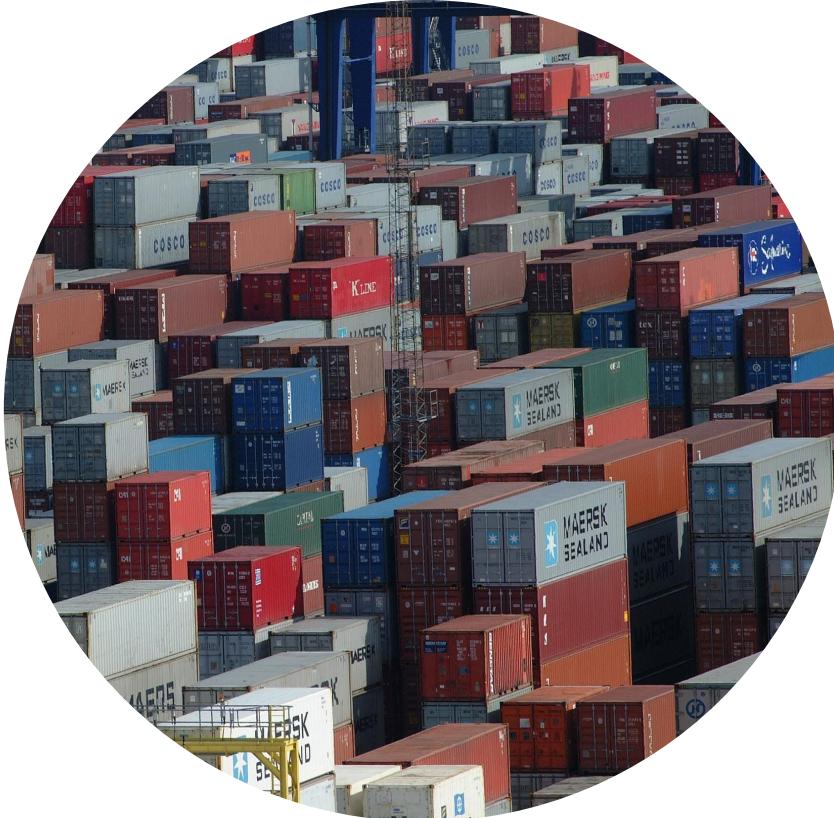
Step 3

Deploy

課題:
時間がかかる
頻繁なリリースができない

メリット:
通常のデプロイ時に行っていた作業はパッケージングで行うので、デプロイが短時間で可能

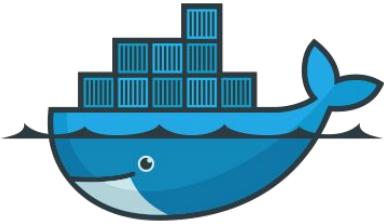
次に出てくる課題



大量のコンテナ化されたア
プリケーションを
パッケージング
デプロイ
オーケストレーション
管理

するのは、複雑で手間が
かかる

OpenShift Enterprise



ポータビリティの高い
Dockerフォーマット



オーケストレーション



コンテナの実行環境

+

コンテナのビルト、デプロイ、
複数コンテナの連携

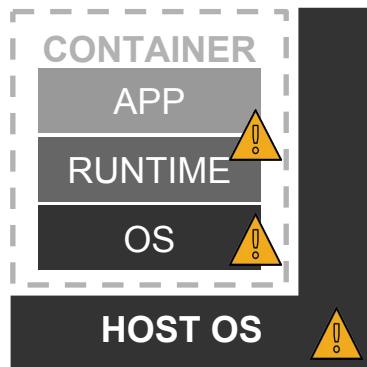
+

アプリケーションの
ライフサイクル管理

Red Hat Certified Container

Docker + DIY

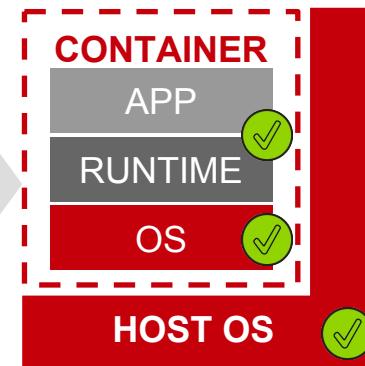
- ・信頼できないコンテナイメージ: コンテナとして何がパッケージングされているのかわからない
- ・脆弱: いつ、どうやってライブラリの更新をするかわからない
- ・ポータビリティ: 本当にポータビリティある?



- Certified Container Images
- Certification Middleware
- Red Hat Container Registry
- Container Development Kit
- Certification as a service

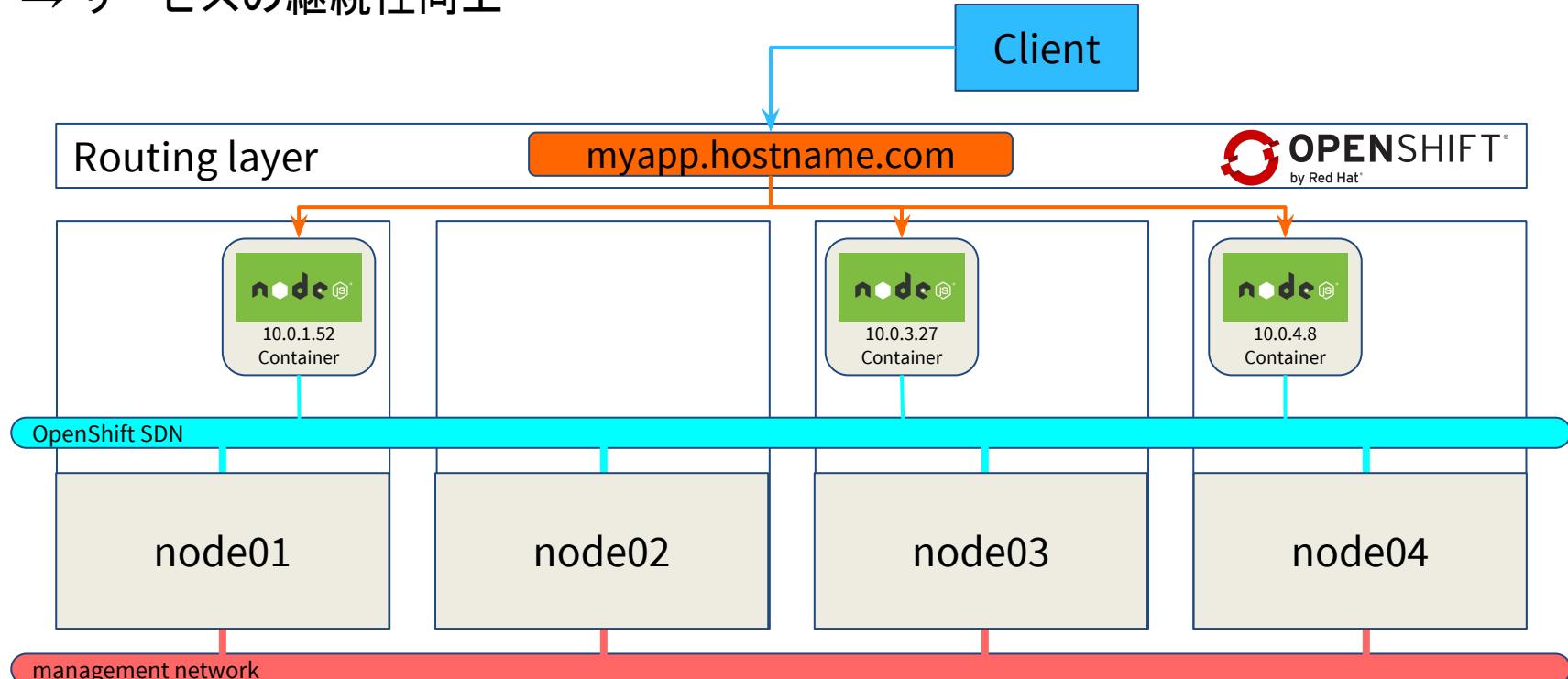
RED HAT CERTIFIED

- ・信頼できるコンテナイメージ
- ・コンテナにもセキュリティアップデートを提供
- ・高いポータビリティ

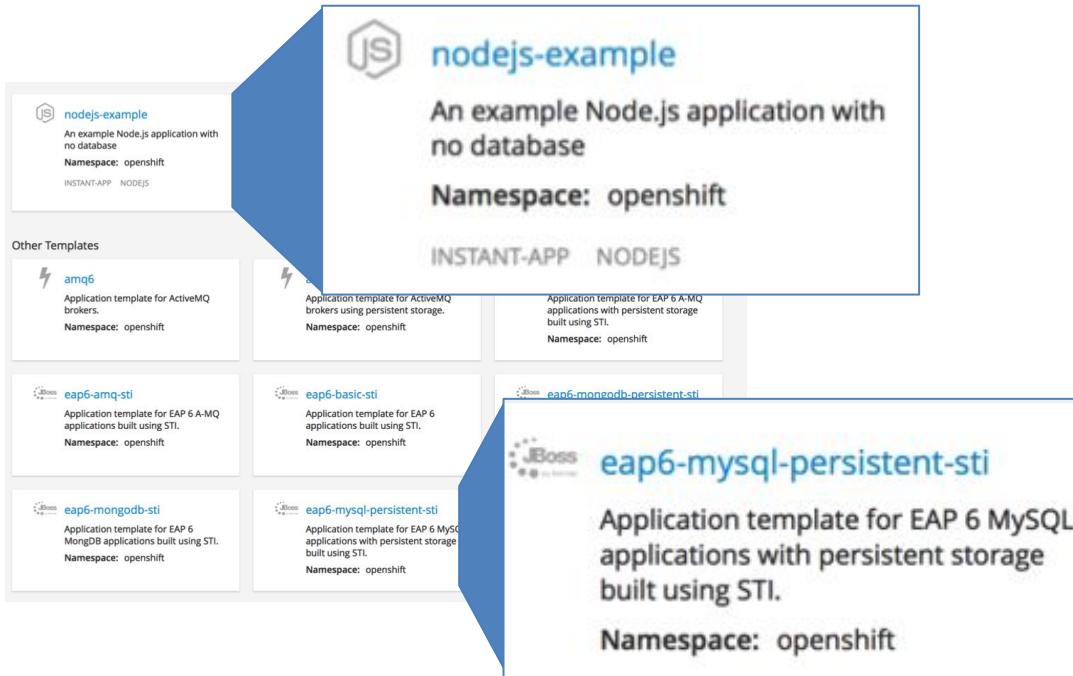


コンテナ:OpenShiftによるコンテナ化

アプリケーションがどこで(どのサーバで)動いているかを意識しない
→ サービスの継続性向上



標準化: アプリケーション実行環境



- **Template**
 - アプリケーションの標準的な実行環境の定義
 - Build, Deployのストラテジ
 - パラメータ
- よくあるパターンは Red Hat から提供
- プロジェクトなどに合わせて作成し、OPENSHIFTへ登録も可能

標準化:テンプレート

テンプレートで
ミドルウェアや
言語などを選択



ソースコードリポジトリを指
定

PHP 5.6
Build and run PHP 5.6 applications

* Name: hello

Used to uniquely identify within this project all the resources created to support the application.

* Git Repository URL: <https://github.com/akubicharm/php-hello-world.git#demo>

Sample repository for php: <https://github.com/openshift/cakephp-ex.git> Try it ↗

Show advanced build and deployment options

Create Cancel



OpenShift が
Docker Imageを作成し、
Docker Containerを起動

OPENSHIFT ENTERPRISE

Overview

Browse

Settings

SERVICE : HELLO
hello-hello.apps.ose31

DEPLOYMENT: HELLO. #1

CONTAINER: HELLO

- Image: hello-hello (5bb3c95)
- Build: #1 from <<https://github.com/akubicharm/php-hello-world.git>>

Ports: 8080/TCP



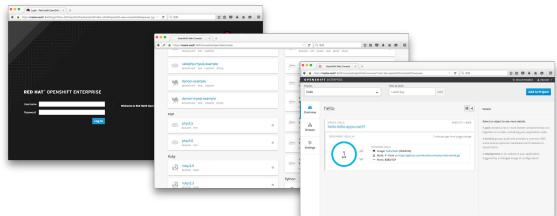
これはほ
んの
一例です



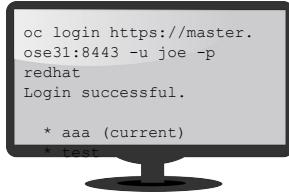
<http://bit.ly/komizo-nodejs>

UX:3つのユーザインターフェース

WebUI



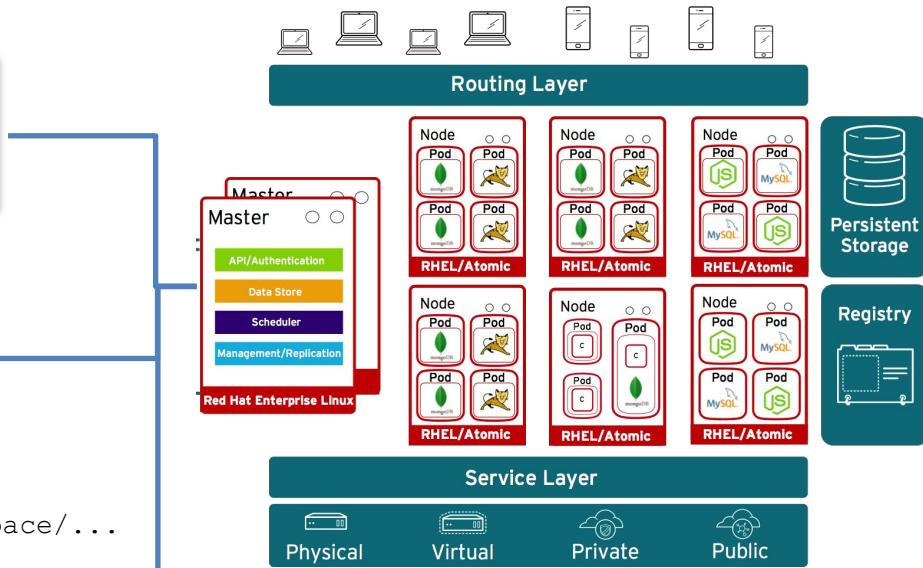
CLI



REST API

[http://master:8443/oapi/v1/namespaces/...](http://master:8443/oapi/v1/namespaces/)

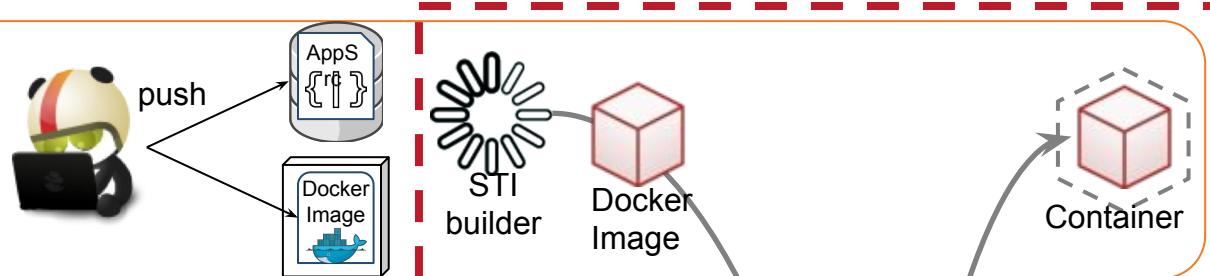
```
{  
    "type": "Generic",  
    "generic": {  
        "secret": "secret101"  
    }  
}
```



UX: アプリケーションのビルドとデプロイ

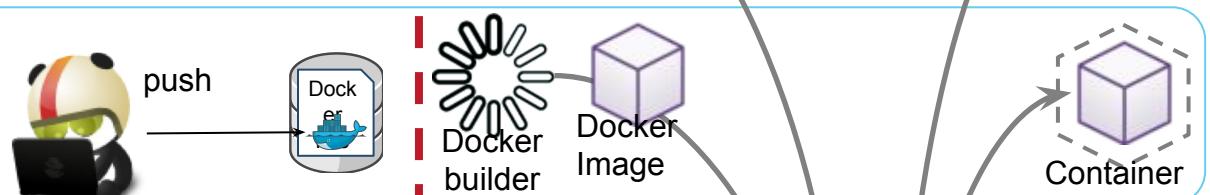
ソースコードからビルド

Source To Image build



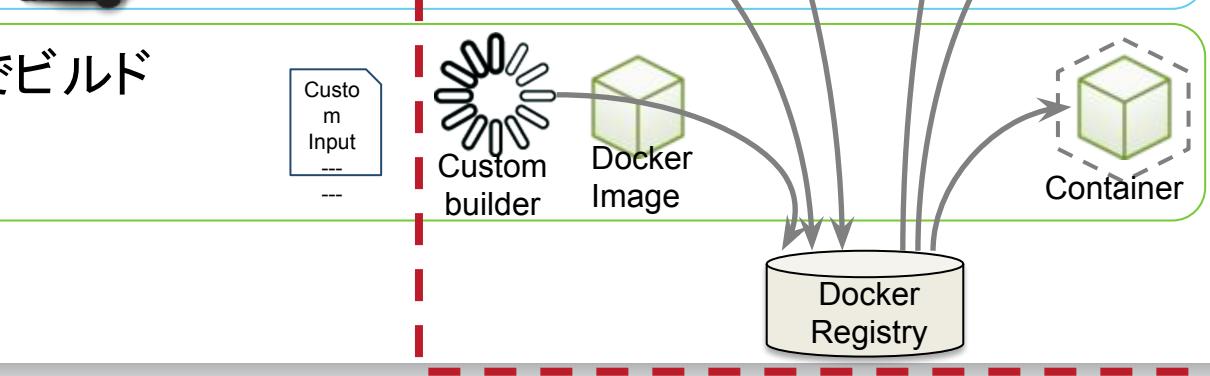
Docker Fileからビルド

Docker Build

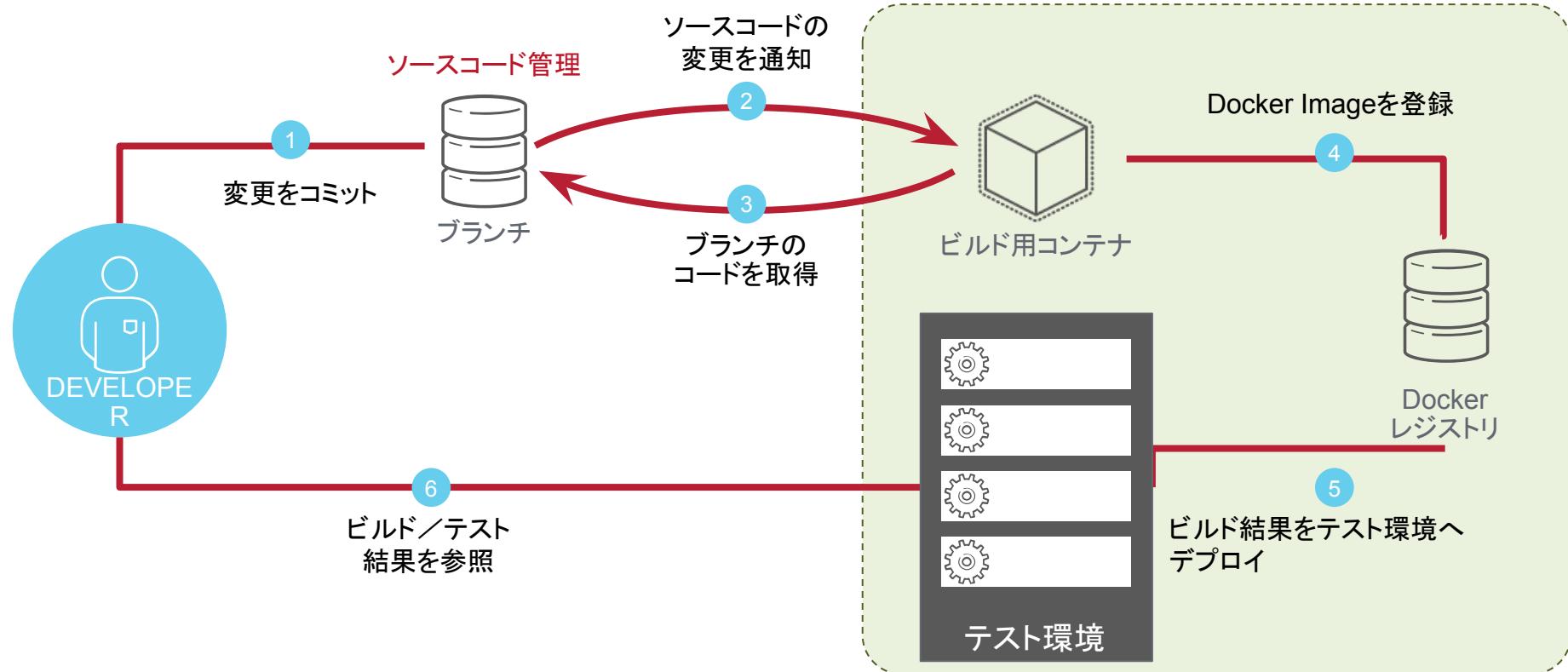


カスタム定義のビルダーでビルド

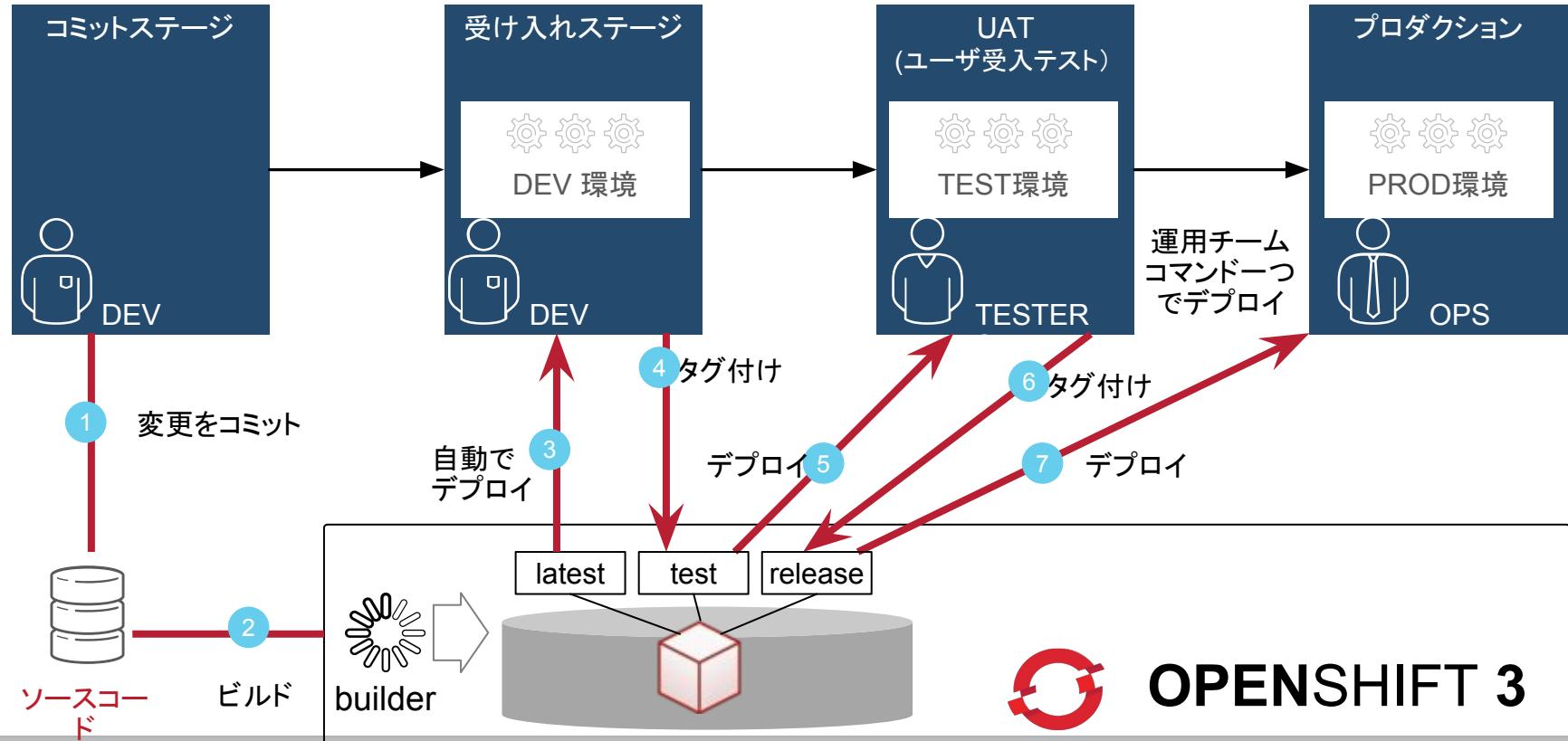
Custom Build



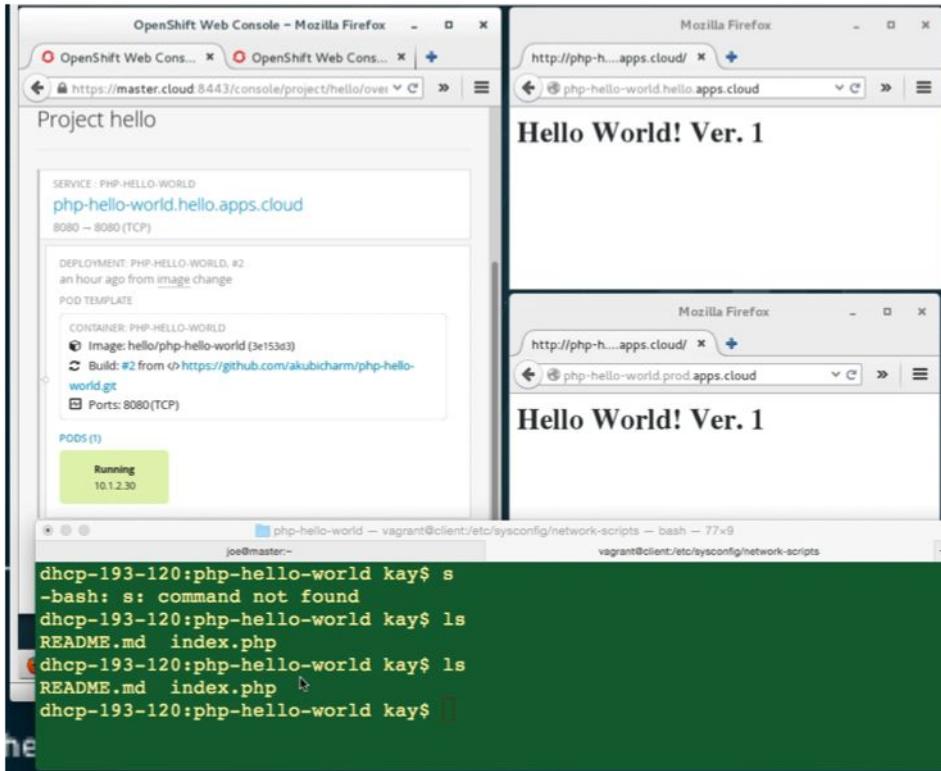
自動化:継続的インテグレーションの実現



自動化:CI/CDフロー

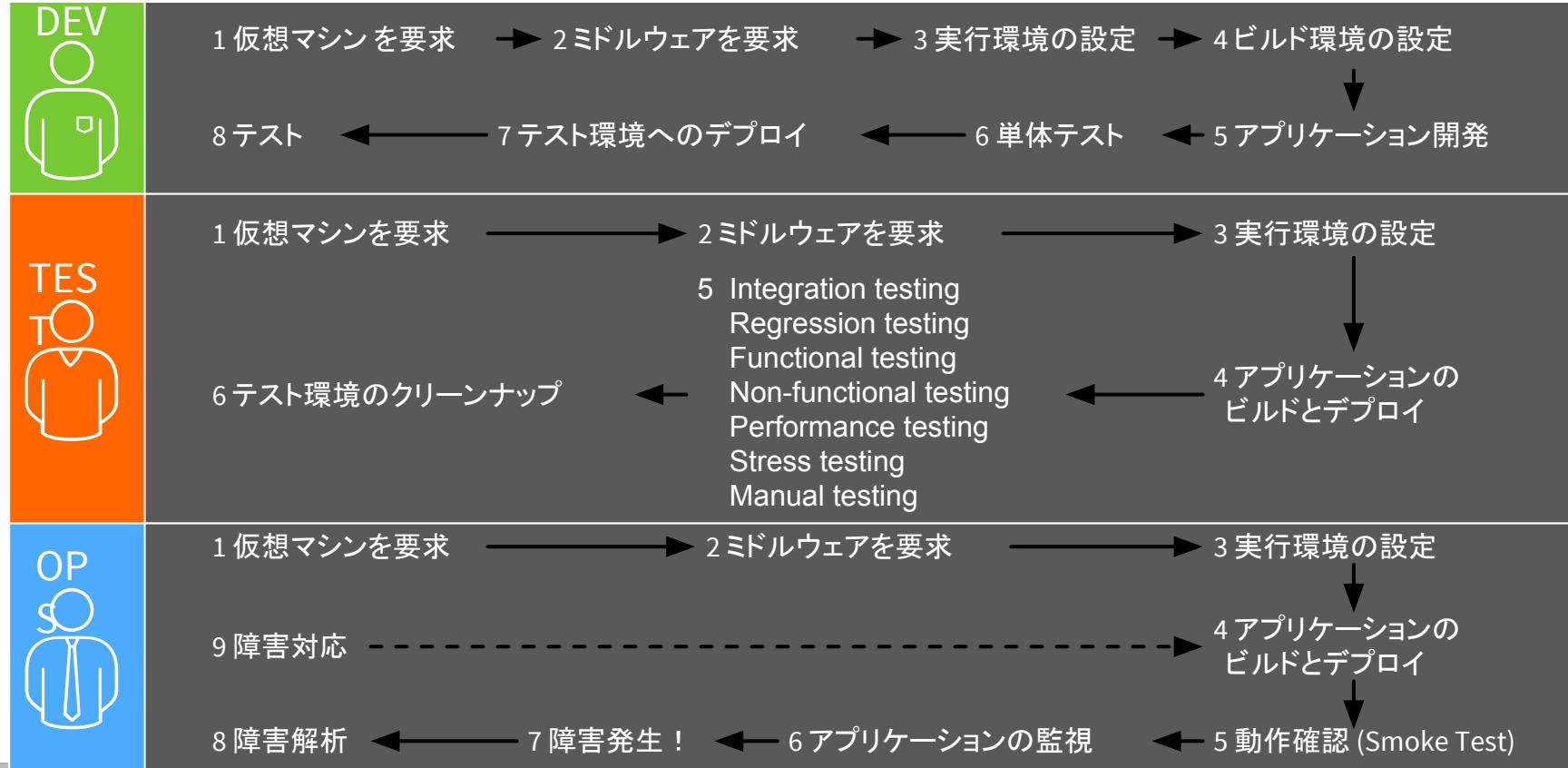


自動化:CI/CDの実現

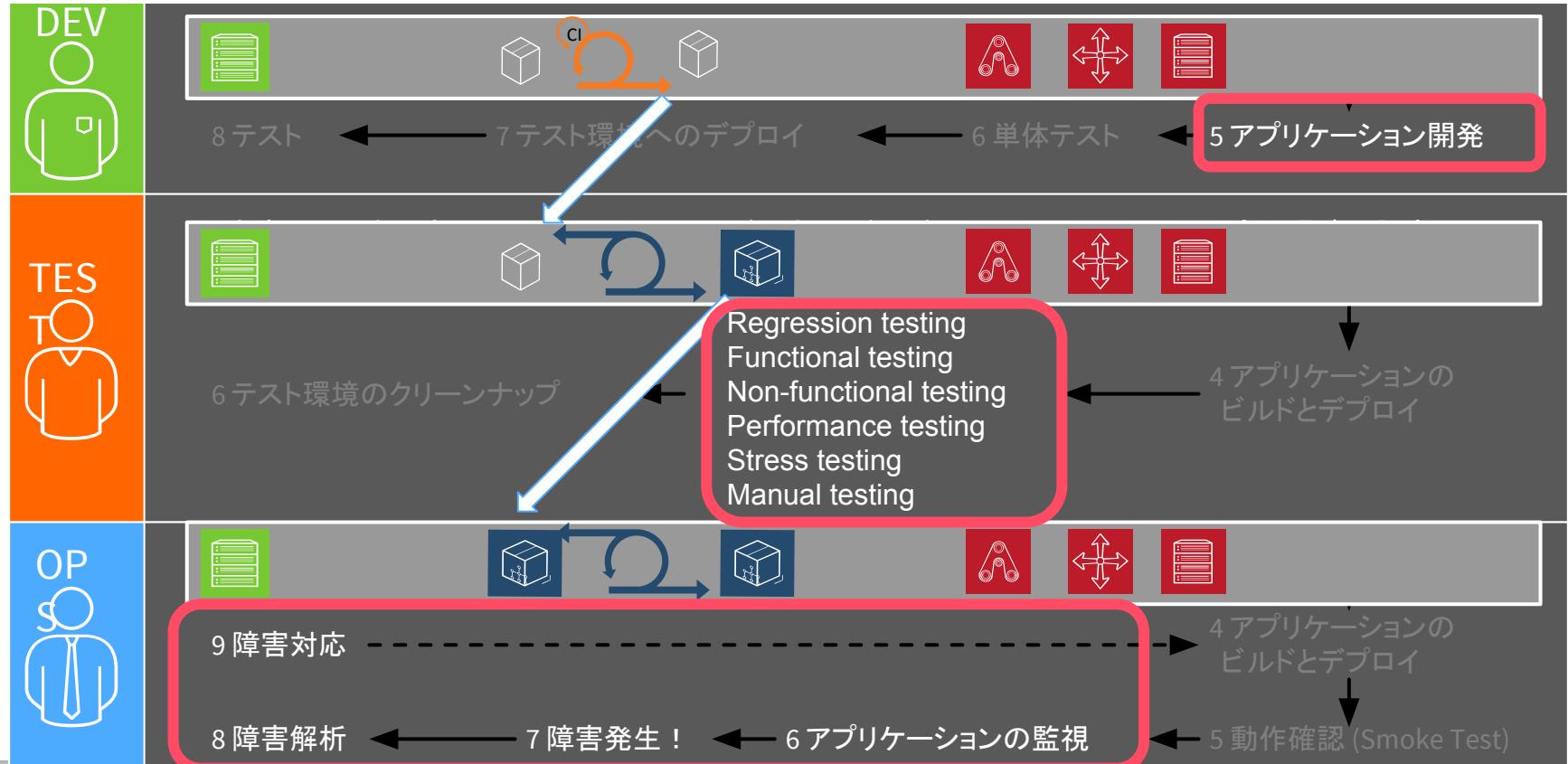


<https://www.youtube.com/watch?v=bsarIV1L5uk>

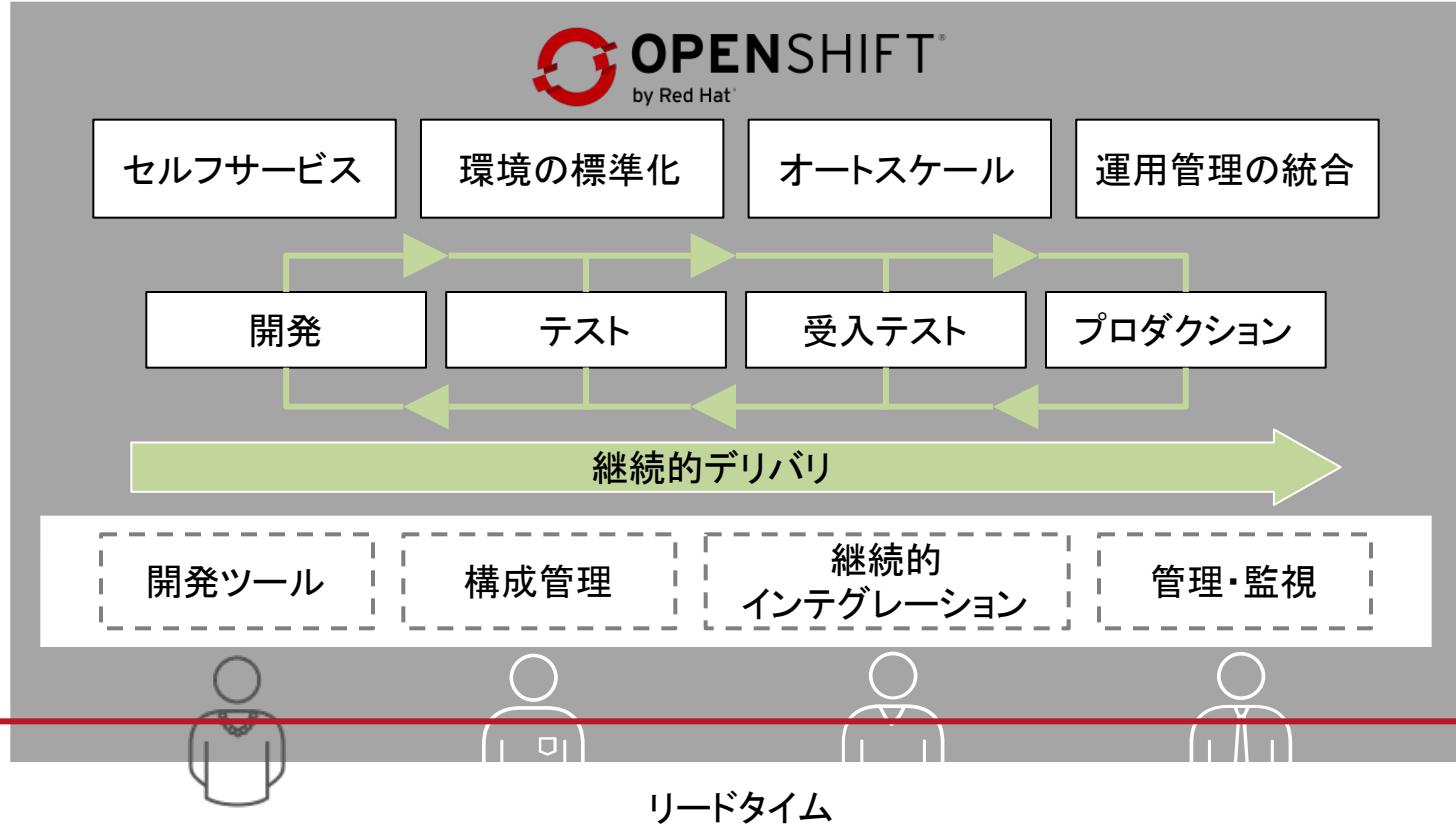
従来の開発～運用までのフロー



OpenShiftを活用してリードタイムの短縮



企画からサービス提供までのリードタイムを短縮





OPENSIFT
by Red Hat®