
迅速なリリースを支えるDevOps最新技術動向

2017/3/9

株式会社 日立製作所 研究開発グループ
システムイノベーションセンタ システム生産性研究部

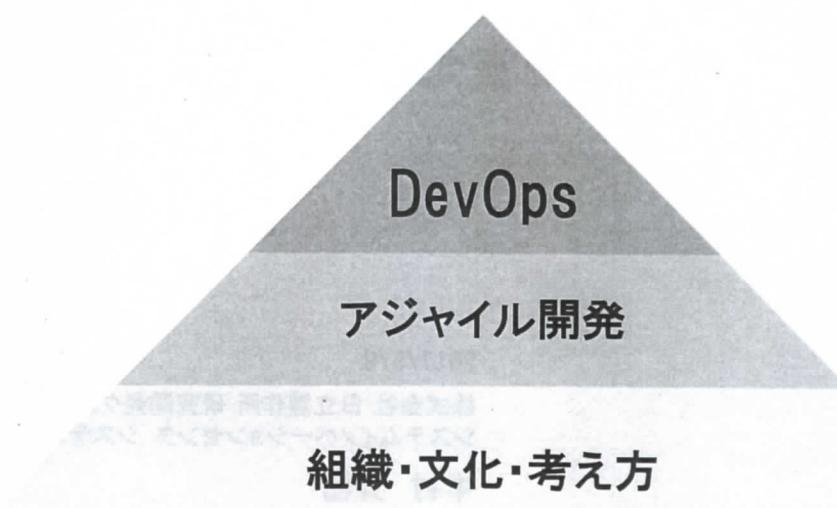
中村 秀樹

© Hitachi, Ltd. 2017. All rights reserved.

Contents

1. DevOps概要と必要となる技術要素
2. アーキテクチャの動向とコンテナ仮想化
3. Dockerおよび関連ツールの概要と動向
4. Dockerの課題
5. まとめ

- Dev(開発)とOps(運用)が密に協調・連携して、ビジネス価値を高めようとする取り組みや文化
- DevOpsはアジャイル開発を前提にしており、そのための組織・文化・考え方方が重要



© Hitachi, Ltd. 2017. All rights reserved.

2

1-2. 前提となる文化・考え方

- 8つの習慣
 - ✓ マイクロソフト DevOpsテクニカルエバンジェリスト牛尾剛氏とJapan Intercultural Consulting Rochelle Kopp氏が考案したAgile/DevOps の導入を成功させるための文化的習慣

- ① Be Lazy
- ② リスクや間違いを快く受け入れる
- ③ 不確実性を受入れる
- ④ サーヴァント・リーダーシップ
- ⑤ セルフマネジメントチーム
- ⑥ 従業員への信頼
- ⑦ 個人の自信
- ⑧ 階層関係のパワーバランス

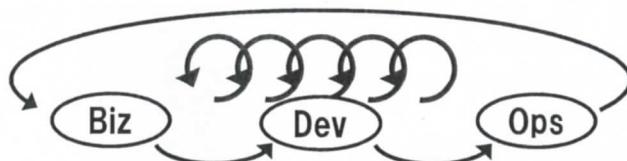
① ソフトウェアデリバリの効率化(BizからOpsまでの流れを効率化)



② フィードバックの効率化(OpsからBizへの流れの効率化)



③ 継続的実験、学びのサイクルの実現



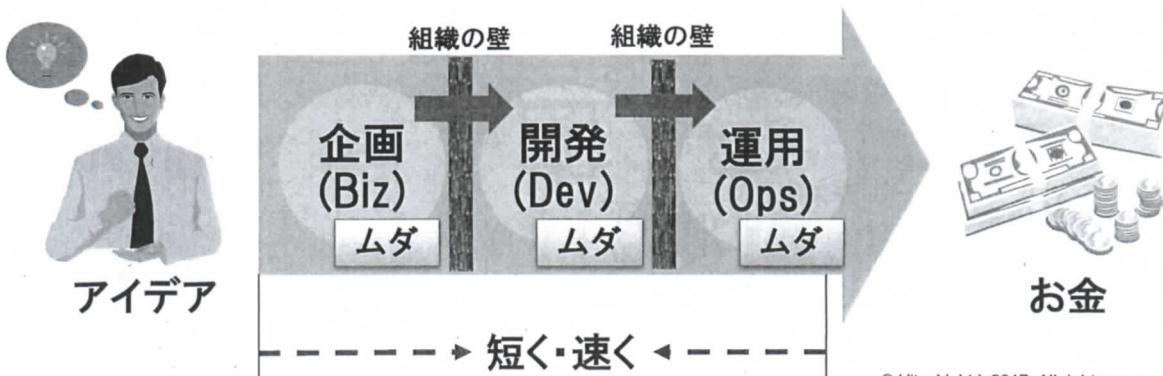
※Gene Kim氏の"The Three Ways:The Principles Underpinning DevOps"を参考に修正

© Hitachi, Ltd. 2017. All rights reserved.

4

1-3-1. ステップ1:ソフトウェアデリバリの効率化

- ・ アイディアからデリバリ(お金に換える)までのリードタイムを短縮
 - アジャイルや継続的デリバリーの考えを元に、クラウド・インフラ自動化技術が重要
 - ✓ 開発作業を自動化
 - ✓ 継続的インテグレーション(CI)、自動テスト
 - ✓ バージョン管理
 - ✓ Infrastructure as Code、リリーススマネージメント
 - ✓ 運用の効率化
 - ✓ 自動デプロイ、自動スケーリング
 - ✓ 上記を効率よく行うアーキテクチャ
 - ✓ マイクロサービスアーキテクチャ、サーバレスアーキテクチャなど



© Hitachi, Ltd. 2017. All rights reserved.

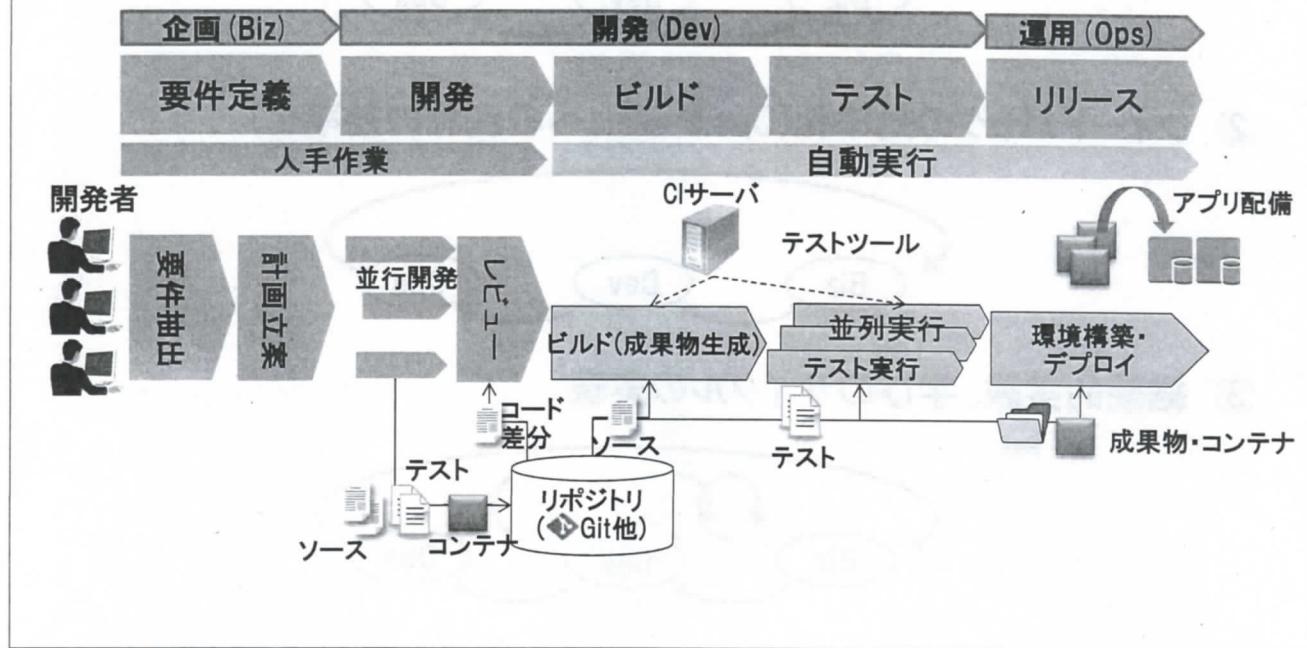
5

1-3-2. ソフトウェアデリバリの効率化を実現するOSS

HITACHI
Inspire the Next

- ・ ビルド以降の工程を自動化してデリバリの効率化を実現
- ・ 工程毎、環境に合わせて、最適なOSSを組み合わせて実現

DevOps



© Hitachi, Ltd. 2017. All rights reserved.

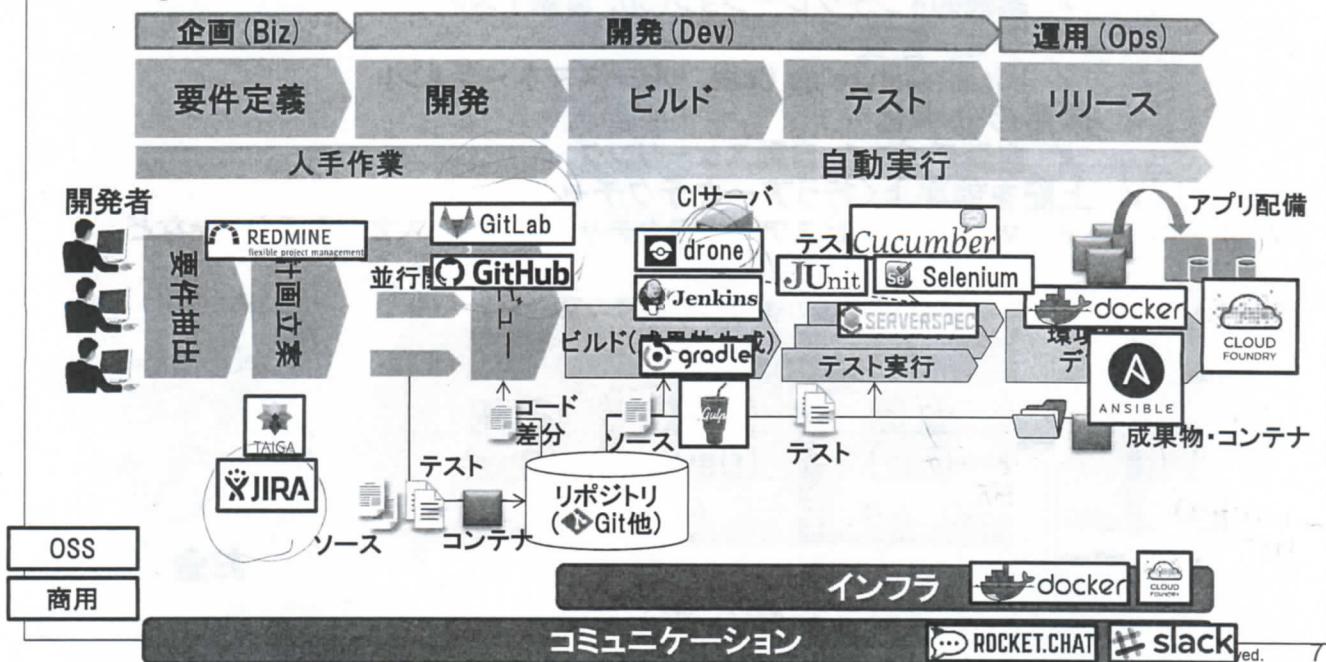
6

1-3-2. ソフトウェアデリバリの効率化を実現するOSS

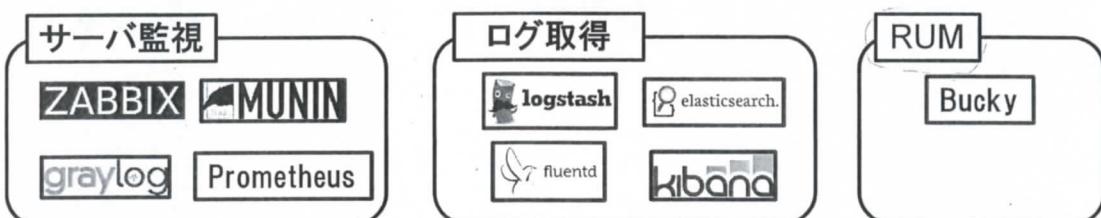
HITACHI
Inspire the Next

- ・ ビルド以降の工程を自動化してデリバリの効率化を実現
- ・ 工程毎、環境に合わせて、最適なOSSを組み合わせて実現

DevOps



- 本番環境やユーザから如何に、効率良く学べるようにするか
 - 実運用の状況把握と、問題発生時の対処を容易化する仕掛け作り
 - ✓ アプリケーションログ取得
 - ✓ アプリケーションパフォーマンスの監視
 - ✓ 可用性監視
 - ✓ 自動回復(ロールバックとロールフォワード)
- フィードバックの効率化を実現するOSS



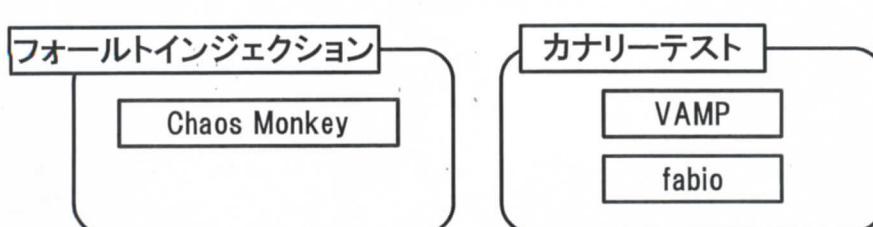
RUM:Real User Monitoring

© Hitachi, Ltd. 2017. All rights reserved.

8

1-3-3. ステップ3:継続的実験、学びのサイクルの実現

- 仮説に基づく本番環境やユーザで検証の実現と、高速化
 - リーンスタートアップの考え方をベースにした取り組み
 - ✓ 仮説に基づく開発
 - ✓ 本番環境でのテスト
 - ✓ フォールトインジェクション
 - ~~✓ 仕様状況監視/ユーザテレメトリ~~
 - ✓ A/Bテスト、カナリーテスト
 - ✓ 仮説駆動のバックログ
 - ✓ 上記を簡単に実現する環境
- 継続的実験を実現するOSS



- DevOps実現には、インフラの構築・破棄を高速かつ柔軟に行う基盤が必須
 - Software Definedで高速かつ柔軟に所望のシステムを構築するには、IaaS/PaaS/コンテナ仮想化基盤が必要
 - ✓ ユーザの利用状況に合わせた柔軟なリソース割り当て
→Scale-out/Scale-in
 - ✓ システムを止めないデプロイ:BlueGreen Deploy
 - ✓ A/Bテスト、カナリーテスト:複数システムを柔軟に提供
 - 高度なDevOpsを行う際には、構築・破棄を多く実施するため、起動が高速なコンテナ仮想化はほぼ必須になりつつある

© Hitachi, Ltd. 2017. All rights reserved. 10

Contents

- 1 DevOps概要と必要な技術要素
2. アーキテクチャの動向とコンテナ仮想化
- 3 Dockerおよび関連ツールの概要と動向
- 4 Dockerの課題
- 5 まとめ

- 単一マシンから大規模分散システムへ
 - ムーアの法則の終焉
 - ✓ 単一CPUの性能向上は頭打ちで、分散処理による性能向上
 - クラウドの登場により、オンデマンドなリソース割り当て
 - 様々なシステムが連携する大規模なシステムへ(IoT, CPS)
- 個々のサーバは大規模・複雑から小規模・単純へ
 - 分散させる対象がより上位へ(システム→ワークロード)
- 大規模分散システムのアーキテクチャ
 - マイクロサービスアーキテクチャ
 - ✓ 上位概念(機能・組織・文化的視点)でのアーキテクチャ
 - サーバレスアーキテクチャ
 - ✓ 実装視点(どのように開発するか)でのアーキテクチャ

2-2. マイクロサービスアーキテクチャ

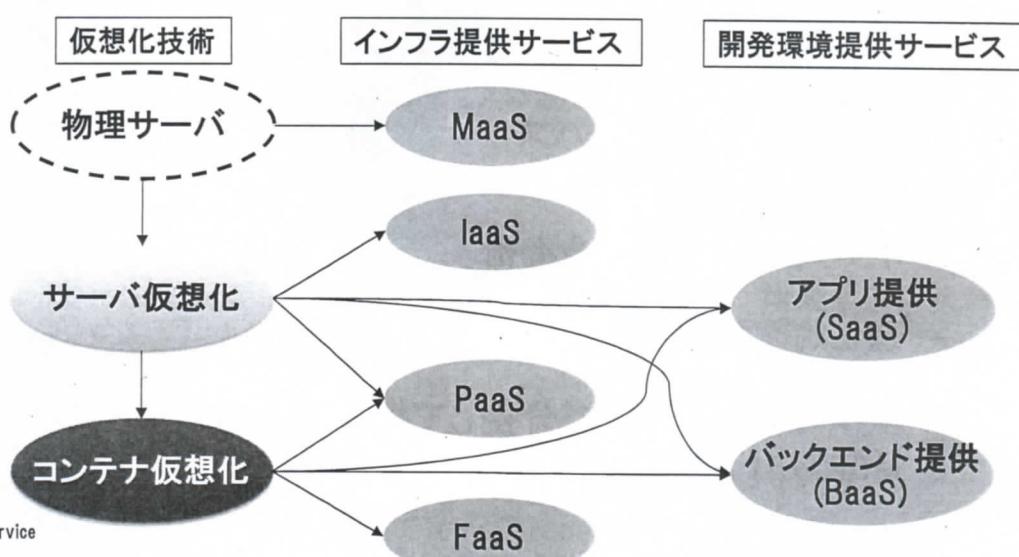
- 2014年にJames Lewis/Martin Fowlerが提唱したアーキテクチャ
 - Webサービスのベストプラクティスとしてまとめたもの
- 一つのアプリケーションを小さなサービス群の組合せで構築
- 9つの特徴
 - サービスによるコンポーネント化
 - ビジネスケイパビリティに基づく組織化
 - プロジェクトドリブンではなく、プロダクトドリブン
 - 貢いエンドポイントとバス
 - 分散ガバナンス
 - 分散データ管理
 - インフラストラクチャ自動化
 - 障害設計
 - 進化的設計
- SOAとMSAとの違い
 - SOA:トップダウンで分割して構築(封建制、君主制)
 - MSA:ボトムアップで構築(民主的)

- 2012年に、開発者がサーバを意識せずに開発する”Serverless”が提案された
- 以後、以下の3つのコンテキストで捉えられている
 - BaaS(Backend as a Service)
 - ✓ クライアントのアプリケーション(SPA)が、BaaSが提供するID基盤やデータストアを直接利用するアーキテクチャ
 - ✓ AWSでは2-Tierアプリケーションと呼ばれる
 - FaaS(Function as a Service)
 - ✓ アプリケーション単位ではなく関数(Function)単位で起動し、関数の組合せにより、アプリケーションを実現するアーキテクチャ
 - ✓ 昨今のサーバレスアーキテクチャはFaaSを指すことが多い
 - リアクティブシステム
 - ✓ 中央集権的なサーバの従来のアーキテクチャと異なり、複数のコンポーネントが非同期なイベント(メッセージ)で接続し、全体として機能を実現するアーキテクチャ

© Hitachi, Ltd. 2017. All rights reserved. 14

2-4. 仮想化の動向

- 仮想化で扱う単位が、より上位レイヤーに、より小さな単位となり、より軽量な仮想化が求められるようになってきた
- これらを実現する技術が”コンテナ仮想化“



IaaS: Infrastructure as a Service
 MaaS: Metal as a Service
 PaaS: Platform as a Service
 SaaS: Software as a Service
 BaaS: Backend as a Service
 FaaS: Function as a Service

© Hitachi, Ltd. 2017. All rights reserved. 15

- 1つのOS上に隔離空間を作成し、その上でアプリケーションを動作させる仮想化方式
- 1つのアプリが1つのプロセスとして管理

	一般的なサーバ仮想化	コンテナ仮想化
構成		
起動時間	遅い(30秒～1分)	速い(1秒未満)
消費リソース	多い	少ない
アクセス性能	遅い	速い

© Hitachi, Ltd. 2017. All rights reserved.

16

2-6. サーバ仮想化とコンテナ仮想化との違い

- 物理サーバ、サーバ仮想化とは異なる振る舞い
 - OSの起動シーケンスは実行されない
 - Daemon起動も基本的には行われない
- サーバの振る舞いをそのまま再現というより、プログラムを起動するだけ
- ただし、各コンテナ内で依存関係は解決しており、他の影響を受けない

- コンテナ仮想化を実現するプラットフォームは以下2つあるが、現段階では、Product-ReadyなものはDockerのみ。
 - Docker
 - Rkt(Rocket)

© Hitachi, Ltd. 2017. All rights reserved. 18

HITACHI
Inspire the Next

Contents

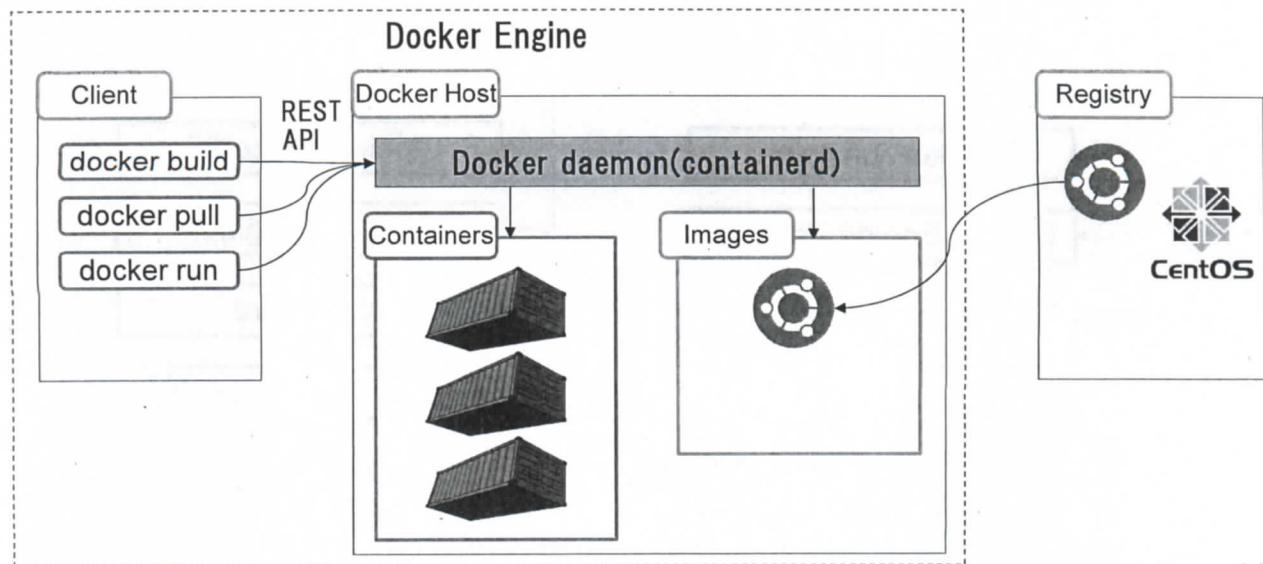
1. DevOps概要と必要となる技術要素
2. アーキテクチャの動向とコンテナ仮想化
3. Dockerおよび関連ツールの概要と動向
4. Dockerの課題
5. まとめ

- ・ コンテナ仮想化技術をベースにした、コンテナ内のアプリケーションの管理・実行を行うプラットフォーム
 - ・ コンテナ仮想化はLinuxの機能をそのまま利用
 - ・ 名前空間(net, ipc, user, mnt, pid, uts)
 - ・ 名前空間毎のハードウェア資源(CPU, MEM等)割当
- ・ Dockerによってもたらされたもの
 - ・ コンテナのイメージ配布フォーマット
 - ・ イメージが持ち運べるように
 - ・ 差分階層形式のイメージ管理ツール
 - ・ イメージ作成が簡単に
 - ・ エコシステムとビジネスモデル

© Hitachi, Ltd. 2017. All rights reserved. 20

3-2-1. Dockerの基本構成

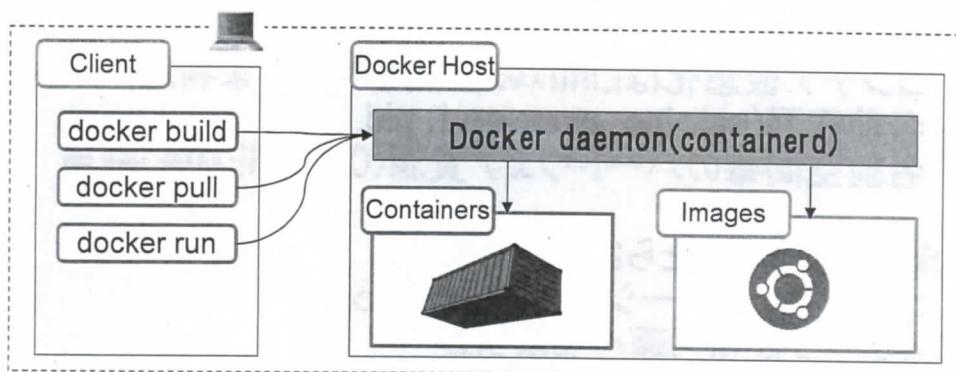
- ・ Docker Engine:コンテナを動作させるコアモジュール
 - ・ Client: CLI(Command Line Interface)
 - ・ Server:Docker daemon(containerd)
 - ・ ClientからServerへはREST APIでアクセス
- ・ Registry:共有するコンテナイメージの保管場所



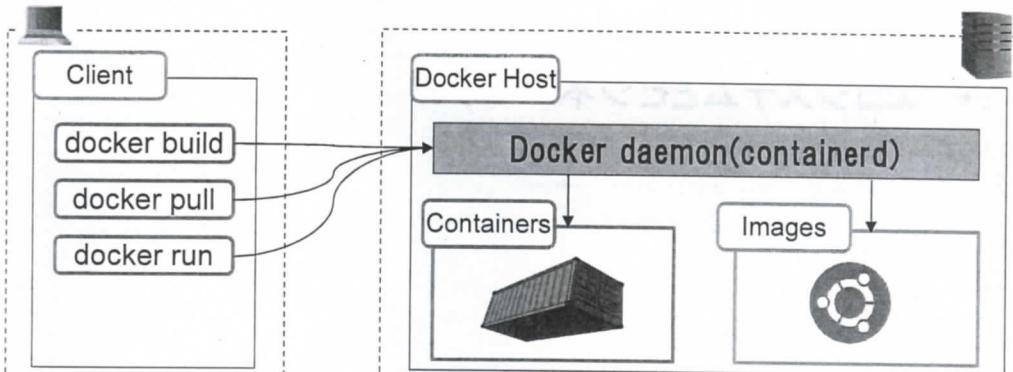
© Hitachi, Ltd. 2017. All rights reserved. 21

1台構成でもClient-Server構成でも、どちらでも動作させることが可能

1台構成



Client-Server
構成



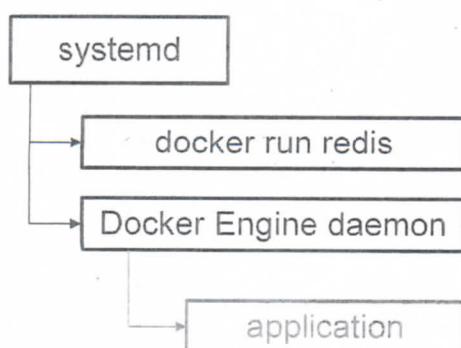
© Hitachi, Ltd. 2017. All rights reserved.

22

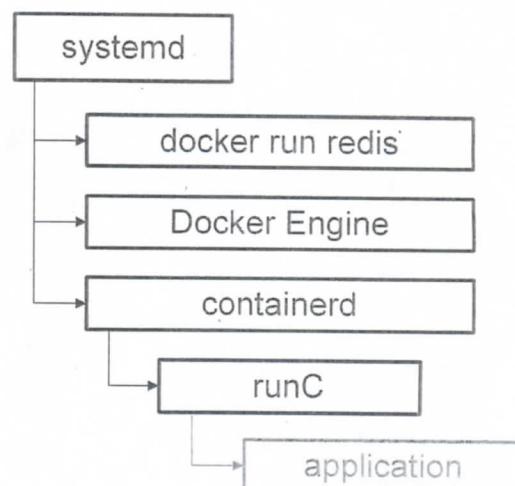
3-3. Dockerのプロセスモデル

- 1.11.0より以前:Docker Engine daemonから起動
- 1.11.0以降:containerd => runCから起動

Docker <1.11.0

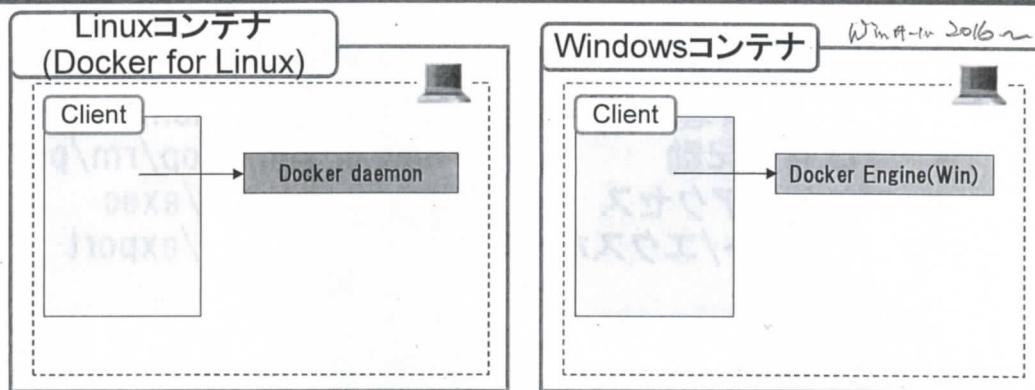


Docker ≥1.11.0

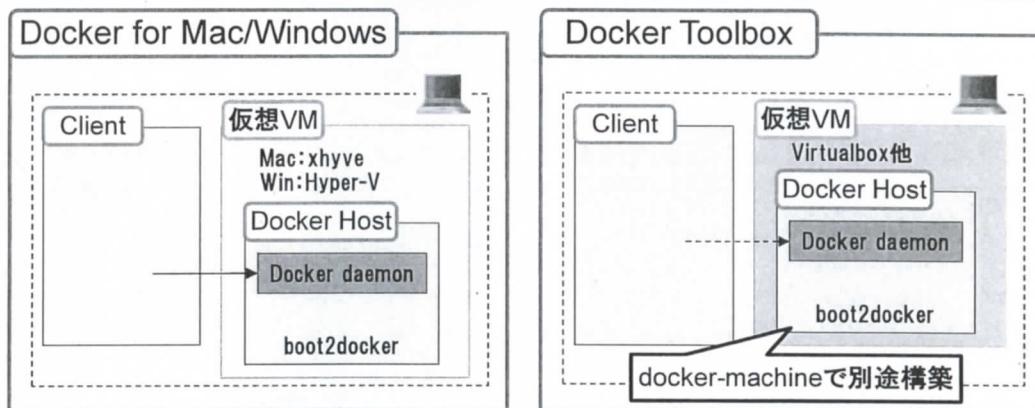


3-4. Dockerのリリース形態

直接利用



仮想VM経由
利用



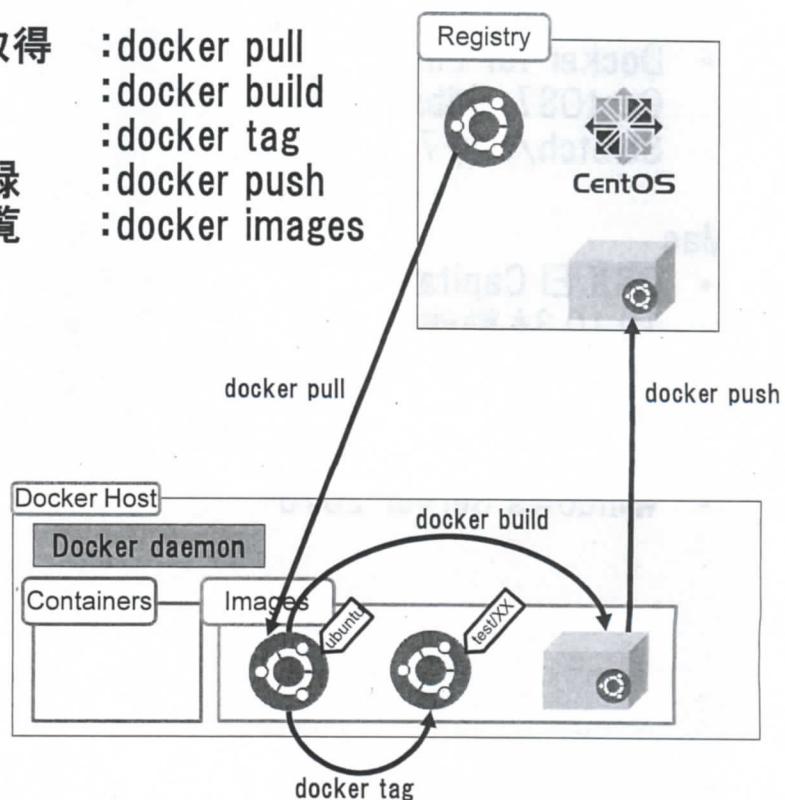
3-5. Dockerの動作環境

- **Linux**
 - Docker for Linux: 64bit Ubuntu 14.04/16.04/16.10、64bit CentOS7、64bit RHEL、64bit Fedora 24/25、64bit Debian Stretch/8.0/7.7他
- **Mac**
 - OSX El Capitan 10.11、macOS Sierra以降、OSX Yosemite 10.10.3も動作するがサポートに制限あり
- **Windows**
 - 64bits Windows 10 Pro/Enterprise/Education+Hyper-V
 - Windows Server 2016

- コンテナ管理API
 - イメージ管理 : docker pull/push/images
 - プロセス起動 : docker run/stop/rm/ps
 - プロセスアクセス : docker attach/exec
 - インポート/エクスポート : docker import/export
- リソース生成管理API
 - 仮想ネットワーク : docker network
 - ボリューム(ストレージ) : docker volume

3-6-1. コンテナ管理API-イメージ管理

- Registryからイメージ取得 : docker pull
- カスタムイメージ作成 : docker build
- タグ(名前)作成 : docker tag
- Registryへイメージ登録 : docker push
- ローカルのイメージ一覧 : docker images



- Dockerfile(イメージカスタマイズ手順)を元にカスタムイメージを作成
- Dockerではイメージにレイヤー構成
 - 仮想ネットワーク : docker network
 - ボリューム(ストレージ) : docker volume

3-6-2. Dockerfileの例

```

FROM debian:jessie

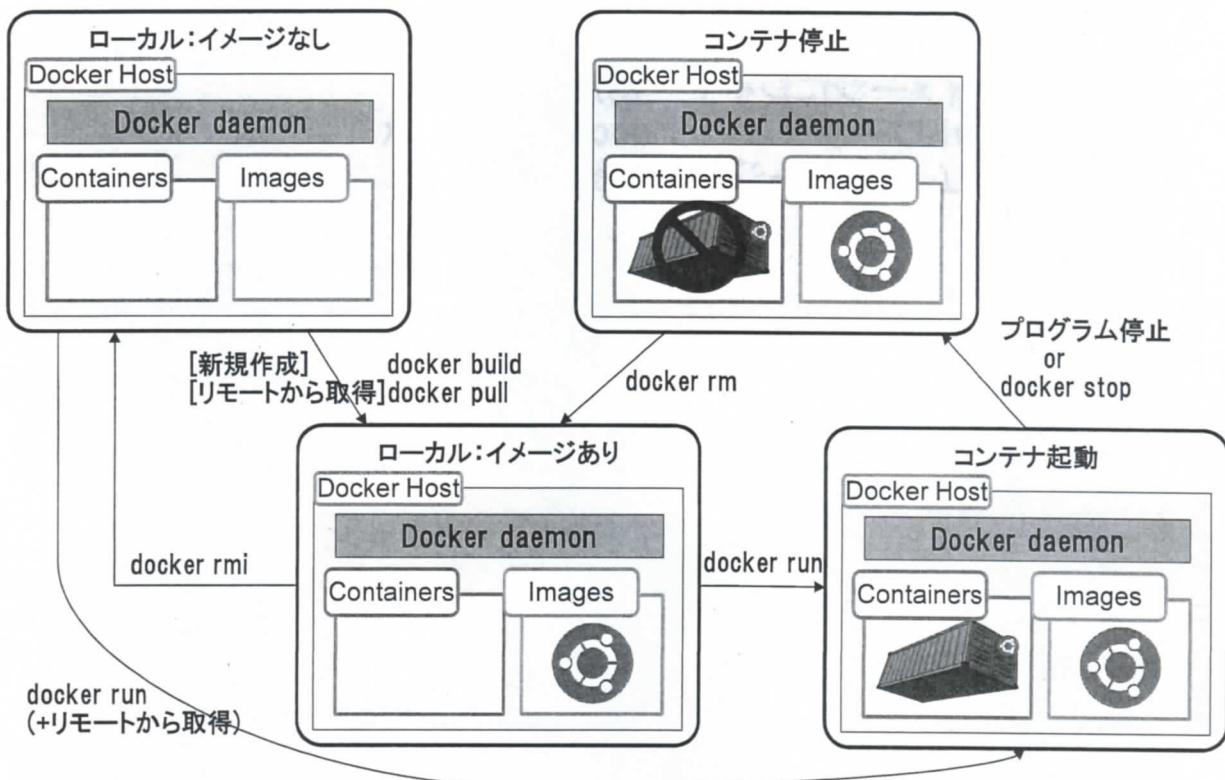
ENV NGINX_VERSION 1.11.10-1~jessie

RUN apt-key adv --keyserver hkp://pgp.mit.edu:80 --recv-keys
573BFD6B3D8FBC641079A6ABABF5BD827BD9BF62 \
& echo "deb http://nginx.org/packages/mainline/debian/ jessie nginx" \
>> /etc/apt/sources.list \
&& apt-get update \
&& apt-get install --no-install-recommends --no-install-suggests -y \
ca-certificates \
nginx=${NGINX_VERSION} \
nginx-module-xslt \
nginx-module-geoip \
nginx-module-image-filter \
nginx-module-perl \
nginx-module-njs \
gettext-base \
&& rm -rf /var/lib/apt/lists/*

RUN ln -sf /dev/stdout /var/log/nginx/access.log \
&& ln -sf /dev/stderr /var/log/nginx/error.log

EXPOSE 80 443
CMD ["nginx", "-g", "daemon off;"]

```

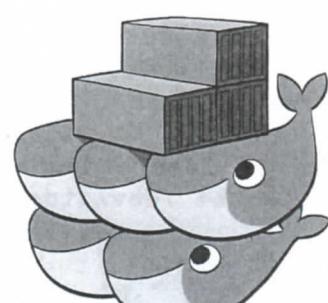
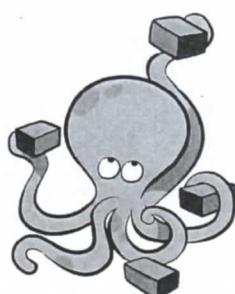


© Hitachi, Ltd. 2017. All rights reserved. 30

3-7. システム構築支援ツール

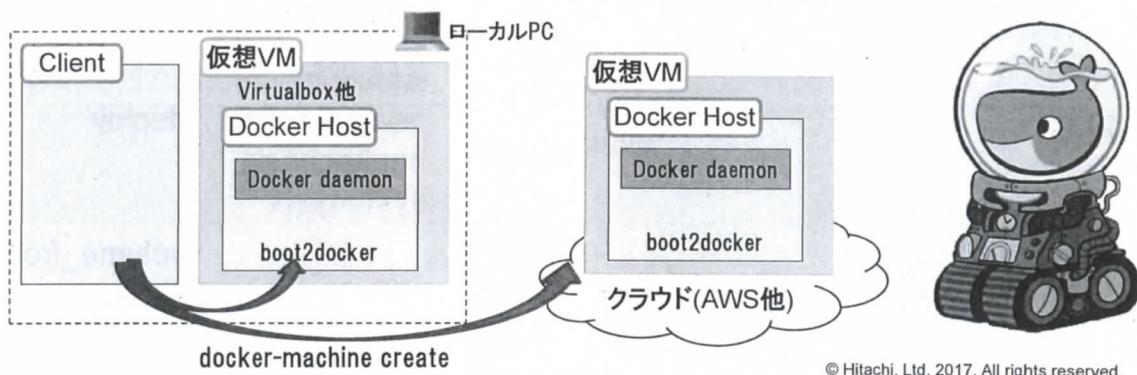
以下のようなニーズに対応したシステム構築支援ツールを提供

- ① Docker Hostを簡単に用意したい
 - **Docker Machine:** Docker Hostを簡単に作成するツール
- ② 複数のコンテナからなるシステムを簡単に構築したい
 - **Docker Compose:** 複数コンテナを使った構築自動化ツール
- ③ 複数サーバ(VM)に跨ったコンテナ実行環境を構築したい
 - **Docker Swarm:** 複数のDocker Hostを束ねクラスタリング



© Hitachi, Ltd. 2017. All rights reserved. 31

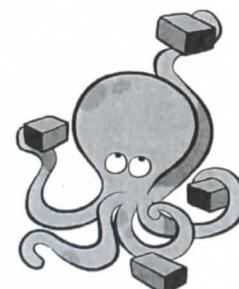
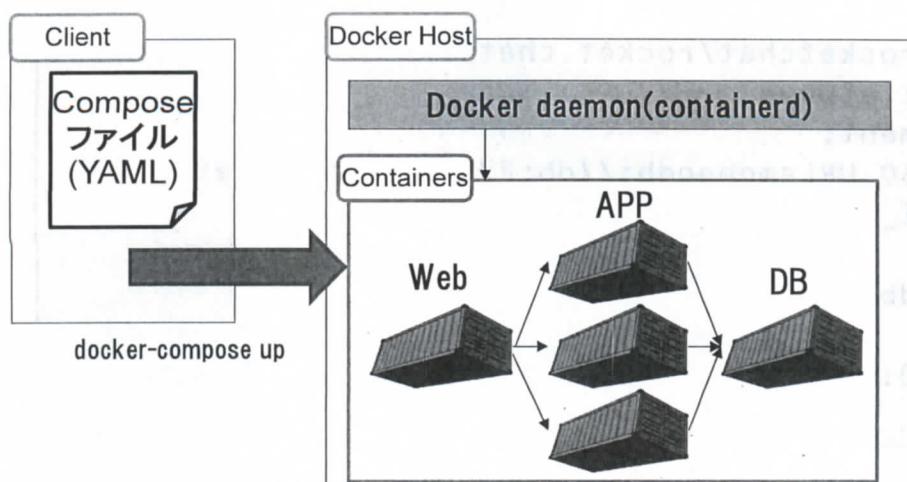
- Docker Hostの構築を簡単に行うツール
 - Driverにより様々な環境での構築が可能に
 - AWS, Azure, Digital Ocean, GCE, exoscale
 - OpenStack, Hyper-V
 - Virtualbox, VMware Fusion/vSphere/vCloud
 - Docker for Mac/Windows, Docker Toolboxをインストールすると自動的にインストール済
- コマンドを実行すると、仮想環境上にOSをインストールして自動的にDocker Hostを構築



© Hitachi, Ltd. 2017. All rights reserved. 32

3-9-1. Docker Compose

- 複数のコンテナを組み合わせ、複雑なシステムの構築を支援
 - Composeファイル: YAMLフォーマットで記載した設定ファイル
 - コンテナの依存関係も考慮した起動スケジューリング
 - コンテナ間のリンクにより簡単にアクセス可能



© Hitachi, Ltd. 2017. All rights reserved. 33

- 複数Versionがあり、記述できる範囲が異なる

	Version 1	Version 2	Version 2.1	Version 3
docker-compose	～1.6.X	1.6.0～	1.9.0～	× (docker stack deploy)
Docker Engine	-	1.10.0～	1.12.0～	1.12.0～ (Swarm Mode)
宣言	不要	version: "2"	version: "2.1"	version: "3"
概要	直service記述	service記述要		
追加	-	volumes, networks	link_local_ips, isolation, userns_mode, healthcheck, systemctls	deploy
削除	-	-	-	volume_from 他

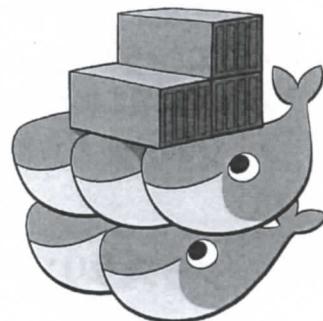
3-9-3. Composeファイルの例

```

version: "2"
services:
  db:
    image: mongo
    restart: always
  rocket:
    image: rocketchat/rocket.chat
    restart: always
    environment:
      - MONGO_URL=mongodb://db:27017/rocketchat
      - ROOT_URL=http://localhost:8000
  links:
    - db:db
  ports:
    - 8000:3000

```

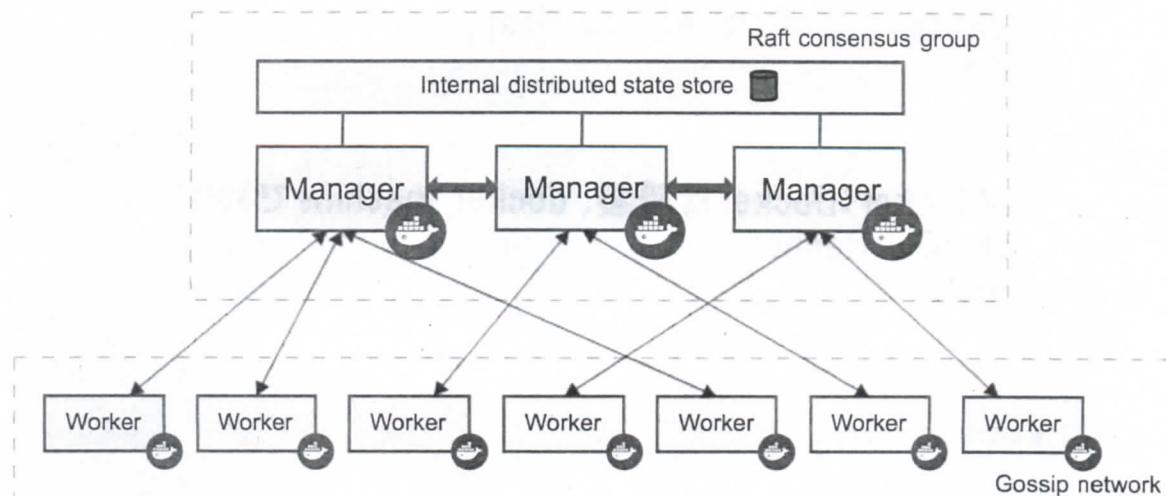
- Docker社謹製クラスタリング機能
- 1.12.0より、Docker Engineに統合(Swarm mode)
 - 1.11.X以前は、Docker Engineとは別にSwarmイメージを使ってクラスタリングを構築



© Hitachi, Ltd. 2017. All rights reserved. 36

3-10-2. Docker Swarmの構成

- Manager Node: 管理ノード
- Worker Node: タスク実行ノード



© Hitachi, Ltd. 2017. All rights reserved.

37

- Kubernetes
 - Google社中心が開発
 - Dockerではデファクトになっている
 - 機能： クラスタリング+デプロイツール
- Apache Mesos
 - 汎用的なクラスタリングツール(Docker以外の用途でも利用される)
 - 開発元のMesosphere社は、MesosにMarathon(デプロイツール)、Chronos(ジョブスケジューラ)他を組合せ、DC/OSとしてリリース
- Rancher/Cattle
 - エンタープライズ向け(ユーザ認証、Audit Log等)
 - 非常に構築が簡単
 - Docker Swarm/Kubernetes/Mesosにも対応

→ クラスタリングツールは乱立しており、要件に合わせ、選定必要

3-11. Dockerホスト用OS

- Dockerでシステムを構築すると、Dockerが動作する環境さえあれば事足りる。Dockerだけが動作するDockerホスト用OSが登場。
- 高機能
 - Atomic Project
 - 商用版: RHEL Atomic Host
 - OSS版: CentOS Atomic Host
 - CoreOS
- 軽量
 - Boot2Docker:Docker社謹製、docker machineで利用
 - RancherOS:Racher
 - BargeOS

	Boot2Docker	RancherOS	BargeOS	Moby(参考)
Release	2017/03/01	2017/02/23	2017/02/28	2017/03/06
Version	17.03.0-ce	0.8.1	2.4.0	17.03.0-ce-mac2 (15654)
Size	39 MB	54 MB	13 MB	61 MB
Kernel	4.4.52	4.9.12	4.9.13	4.9.12
User Land	Tiny Core Linux v7.2(glibc) + BusyBox v1.24.2	Buildroot(uClibc) + BusyBox v1.25.0	Buildroot(glibc) + BusyBox v1.26.2	Alpine Linux v3.5.0(musl libc) + BusyBox v1.25.1
Docker	17.03.0-ce (3a232c8)	1.12.6	1.10.3	17.03.0-ce (3a232c8)
Boot Time	15s	15s	4s	6s

※「メモ:Docker ホスト用軽量 OS の比較」(<http://qiita.com/A-l/items/3d2aedbad4f10141cef3>)を参考に作成

© Hitachi, Ltd. 2017. All rights reserved. 40

Contents

- 1 DevOps概要と必要となる技術要素
- 2 アーキテクチャの動向とコンテナ仮想化
- 3 Dockerおよび関連ツールの概要と動向
- 4. Dockerの課題**
- 5 まとめ

- docker pull問題
 - イメージの検証が甘い
 - <https://titanous.com/posts/docker-insecurity>
- Dockerfile問題
 - Dockerfileの仕様にクセがある
 - 10年後に同じコンテナイメージが作れる保証がない
- docker registry問題
 - セキュリティの問題でDocker Hubが信頼できない
- コンテナイメージ仕様問題
 - コンテナフォーマットが独自でありDockerにロックインされる
 - 統一的なフォーマットを定義する動きも

- docker imageが大きい
 - ubuntuでも数百MB
 - 軽量なLinuxが使われ始めている
 - busybox
 - Alpine
- docker host OSをどうするか?
 - dockerさえ動けば良い
 - CoreOS, RHEL Atomicなど
 - さらに軽量なOS
 - boot2docker, Barge OS

- 2017/3/2に企業向けサブスクリプションサービス”Enterprise Edition”的提供を開始
 - ✓ 無料:Docker Community Edition(Docker CE)
 - ✓ 有料:Docker Enterprise Edition(Docker EE)
 - BASIC: \$750/年・ノード
 - Docker CertifiedのPluginやイメージが利用可能
 - STANDARD: \$1,500/年・ノード
 - BASICに加え、イメージ管理、Docker DataCenter(コンテナアーリ管理、マルチテナント対応)が利用可能
 - ADVANCED: \$2,000/年・ノード
 - ADVANCEDに加え、イメージのセキュリティ検査が利用可能
- ✓ Red Hat Enterprise Linuxなど有償のOSは、Enterprise Editionのみサポートに変更
- ✓ なお、Windows Server 2016ユーザは追加費用なしにDocker EEが利用可能

4-4. エディションと利用可能なプラットフォーム

分類	Platform	Docker EE	Docker CE
Linux系	Ubuntu	✓	✓
	Debian		✓
	Red Hat Enterprise Linux	✓	
	CentOS	✓	✓
	Fedora		✓
	Oracle Linux	✓	
	SUSE Linux Enterprise Server	✓	
Windows	Microsoft Windows Server 2016	✓	
	Microsoft Windows 10		✓
Mac系	macOS		✓
Cloud系	Microsoft Azure	✓	✓
	Amazon Web Service	✓	✓

Contents

1. DevOps概要と必要となる技術要素
2. アーキテクチャの動向とコンテナ仮想化
3. Dockerおよび関連ツールの概要と動向
4. Dockerの課題
5. まとめ

© Hitachi, Ltd. 2017. All rights reserved. 46

5. まとめ

- DevOpsを適用する際に必要となる技術要素を紹介
 - ✓ 特にDevOpsを実現する技術のうち、重要なアーキテクチャ周辺の動向を紹介
 - ✓ マイクロサービスアーキテクチャ
 - ✓ サーバレスアーキテクチャ
- DevOpsを実現を容易にするインフラ技術として、注目されているDockerおよび関連ツールを紹介
 - ✓ Dockerはプロダクションレディの技術になっており、様々な企業で採用しはじめている
 - ✓ Docker以外の技術も乱立し、セキュリティ、課金体系等に課題もあるが、大規模分散システムの構築には必須の技術
→今後、技術動向を注視する必要がある

他社商標に対する表示

- ・Docker, Docker Hub, Docker Swarmは、Docker Inc.の商標または登録商標です。
- ・Ubuntuは、Canonical Limitedの商標です。
- ・CentOSは、Red Hat, Incの商標です。
- ・Kubernetesは、Linux Foundationの商標です。
- ・Apache Mesosは、Apache Software Foundationの商標です。
- ・Amazon Web Services, AWSは、AWSおよびその関連会社の商標です。
- ・Windowsは、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。
- ・その他の製品名称などの固有名詞は各社の商標、登録商標あるいは商品名称です。
また、本資料ではTM、[®]マークは表記しておりません。