




docker初心者の方が 知っておいた方がよい基礎知識



TIS株式会社
戦略技術センター
森元 敏雄

今回のプレゼンテーションの目的

- docker**未経験の方**、**初心者の方**に
dockerについて**なんとなく理解して頂く**
- 今回以下のような参加者の方もおられますが
 - dockerを業務で使いこなしている
 - dockerの周辺ツールを使って運用できる
 - dockerのサポートを仕事にしている



しばし**ご休憩**下さい

1. **docker**とは？

2. **VM**と**docker**の違いとは？

3. dockerの**使われ方**は？

① 開発環境の事例

② 情報提供サイト運用の事例

③ 大規模コマースサイトの事例

4. まとめ



docker



vmware



1. **docker**とは？

2. **VM**と**docker**の違いとは？

3. dockerの**使われ方**は？

- ① 開発環境の事例
- ② 情報提供サイト運用の事例
- ③ 大規模コマースサイトの事例

4. まとめ



docker



vmware



1. dockerとは？

Docker（ドッカー[2]）はソフトウェアコンテナ内のアプリケーションのデプロイメントを自動化するオープンソースソフトウェアである。Linuxカーネルにおける「libcontainer」と呼ばれるLinuxコンテナ技術[3]とaufsのような特殊なファイルシステムを利用してコンテナ型の仮想化を行う[4]。VMware製品などの完全仮想化を行うハイパーバイザー型製品と比べて、ディスク使用量は少なく、インスタンス作成やインスタンス起動は速く、性能劣化がほとんどないという利点を持つ。dockerfileと呼ばれる設定ファイルからコンテナイメージファイルを作成可能という特性を持つ。一方で、コンテナOSとしてはホストOSと同じLinuxカーネルしか動作しない。

出展[Wikipedia] <https://ja.wikipedia.org/wiki/Docker>



出展[<http://blog.oukasoft.com/OS/>]

dockerの特長を3つ上げると

- ① コンテナ型の超軽量サーバ仮想化製品
- ② コンテナ内は実行環境として独立している
- ③ コンテナはリソース消費量が少ない

dockerの概念の理解にはこちらの記事がお勧めです。

Think IT 「Dockerを理解するための8つの軸」

<https://thinkit.co.jp/story/2015/07/29/5382>

※この記事お読みいただいたら、このプレゼンいらないかも

1. dockerとは？

① コンテナ型の超軽量サーバ仮想化製品

- dockerのドライバであるlibcontainerにより、Kernelの一部を分離した実行環境を作成できる
- この実行環境がコンテナであり、ベースOSと独立したファイルシステムやリソースやデバイスを利用できる
- コンテナはKernelやベースOSの機能を共有するため、OSの全機能を乗せる必要がない

「Docker 0.9: introducing execution drivers and libcontainer」

<https://blog.docker.com/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/>

なにができるのか

- OSの内部に**独立したアプリケーションの実行環境 = コンテナ**を生成することができる
- リソース消費量が非常に少なく **1台に物理サーバに多くのコンテナを稼働**させられる

1. dockerとは？

① コンテナ型の超軽量サーバ仮想化製品

● CentOS 7を実際に動かしてみた

➤ メモリ使用量

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	2257	0.0	0.0	11780	2924	pts/1	Ss+	10:59	0:00	/bin/bash
root	2326	0.0	0.0	11780	2984	pts/2	Ss+	11:04	0:00	/bin/bash
root	2369	0.0	0.0	11780	2812	pts/3	Ss+	11:04	0:00	/bin/bash
root	2412	0.0	0.0	11780	2820	pts/4	Ss+	11:04	0:00	/bin/bash
root	2456	0.0	0.0	11780	2808	pts/5	Ss+	11:04	0:00	/bin/bash
root	2498	0.0	0.0	11780	2808	pts/6	Ss+	11:04	0:00	/bin/bash

1コンテナ当
たり3MB弱で
動作している

ベースOSから
はbashの1プ
ロセスに見え
ている

➤ ディスク使用量

12K	348cadde08e0da06bc3a2966385cebd323a3462ccb09d3e67c25314840c2586e
28K	348cadde08e0da06bc3a2966385cebd323a3462ccb09d3e67c25314840c2586e-init
4.0K	a098168aaa9a5c70d240d71771f036f0f0ca8f0ccca00c2586e
4.0K	b984ff074f4526ea4544b4d020144b86b4e59e3d980c0de202f8cc55253731f0
4.0K	cbdf4f6bfe7e0d26b8d30c477663cc1aad1fa6c37798be82910088ee6d5baed
208M	f8b2fd51f64484885fc719fad86b0aaae7bf536d8493b4b3edf00c2620a6d7ca

ディスク使用
量も210MBで
動作している

※Ubuntu 15.10 + docker 1.10.2, build c3959b1で検証

1. dockerとは？

② コンテナ内は実行環境として独立している

- ベースOSとコンテナ内のファイルシステムは分離されている
- コンテナ内のリソース（ネットワークやポート）もベースOSから分離されている
- コンテナ間も同様に分離されている

なにができるのか

- コンテナを利用することでOS内に**多面の実行環境を構築**することができる
- 環境も独立しているため、コンテナで分離していればファイルやバージョンの競合や、設定やポートの**競合が回避**できる

1. dockerとは？

② コンテナ内は実行環境として独立している

● Ubuntu15.10内でCentOS 7コンテナを実行

```
# cat /etc/lsb-release
```

```
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=15.10
```

ベースOSは
Ubuntu

```
# docker ps
```

CONTAINER ID	IMAGE NAMES	COMMAND	CREATED	STATUS
edb01aa1cee	centos	"/bin/bash"	52 minutes ago	Up 52 minutes
suspicious_leavitt				

```
# docker attach edb01aa1cee
```

```
[root@edb01aa1cee /]# cat /etc/redhat-release
CentOS Linux release 7.2.1511 (Core)
```

コンテナの中は
CentOS 7.2

```
[root@edb01aa1cee /]# df -h
```

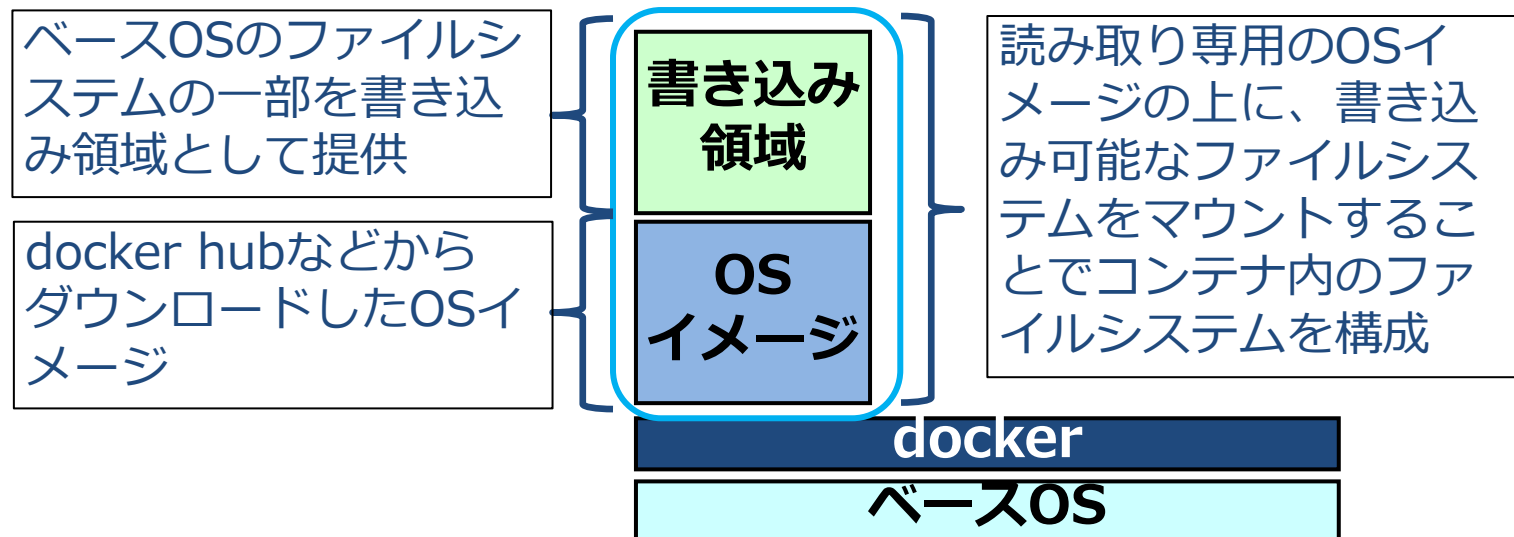
Filesystem	Size	Used	Avail	Use%	Mounted on
none	41G	2.2G	37G	6%	/
tmpfs	3.9G	0	3.9G	0%	/dev
tmpfs	3.9G	0	3.9G	0%	/sys/fs/cgroup
/dev/dm-0	41G	2.2G	37G	6%	/etc/hosts
shm	64M	0	64M	0%	/dev/shm

コンテナ内には
CentOS 7.2の
ファイルシステ
ムが存在

1. dockerとは？

③ コンテナはリソース消費量が少ない

- 読み取り専用のコンテナのベースイメージ上に書き込み可能領域を重ねてmountすることでファイルシステムを作成している

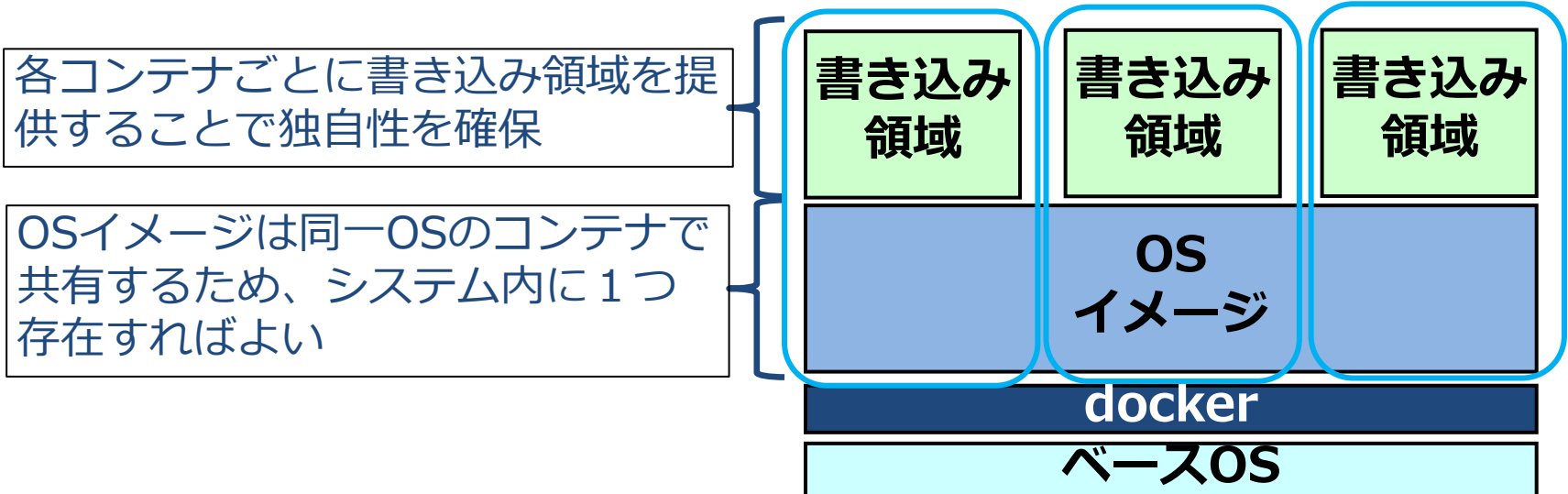


この方式を採用することで、**ベースOSのイメージを変更せず**に**コンテナ独自のファイルシステム**を実現

1. dockerとは？

③ コンテナはリソース消費量が少ない

- 同一OSのコンテナを多数起動する場合、OSイメージのファイルは各コンテナで共有される



同一OSのコンテナを多数起動する場合、各コンテナの差分データのみがハードディスク上に保持されるため、**ハードディスクの消費量が大きく削減**できる

1. dockerとは？

2. VMとdockerの違いとは？

3. dockerの**使われ方**は？

- ① 開発環境の事例
- ② 情報提供サイト運用の事例
- ③ 大規模コマースサイトの事例

4. まとめ



docker



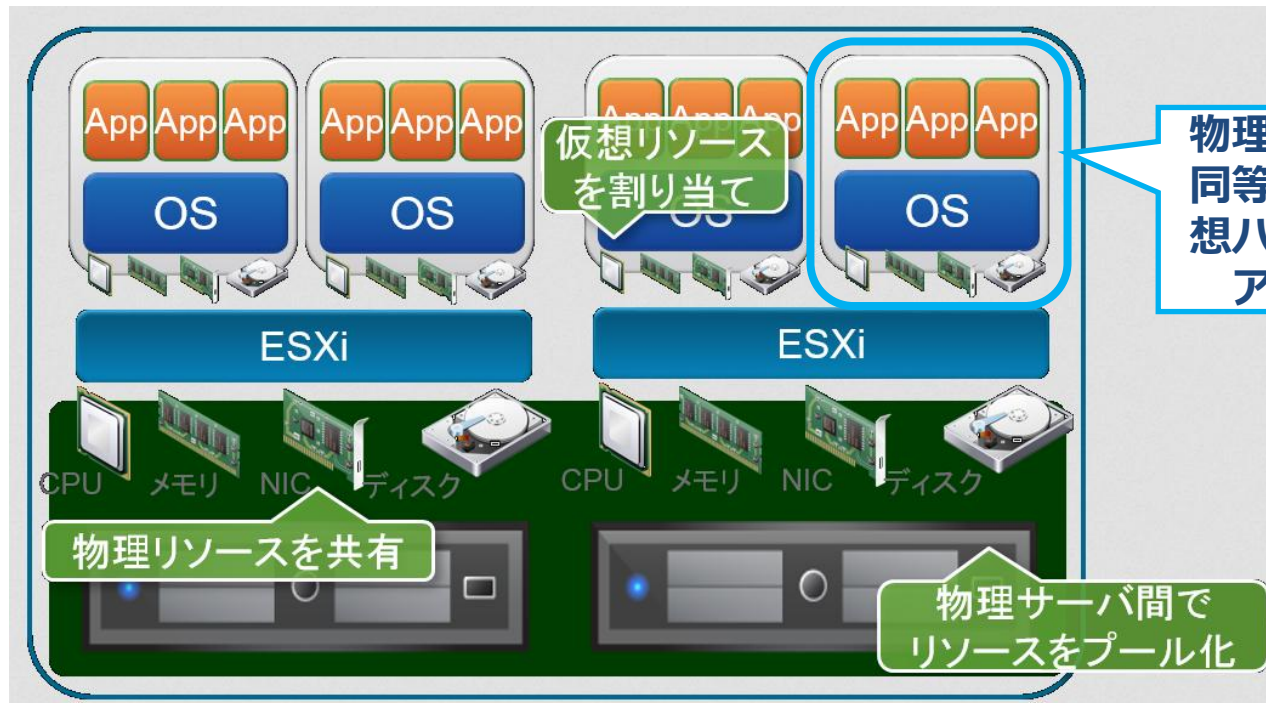
vmware



2. VMとdockerの違いとは？

① VMware/KVM/Xenのサーバ仮想化

- 物理マシンのハードウェアやネットワークスイッチなどハードウェアを完全にエミュレーションできる環境



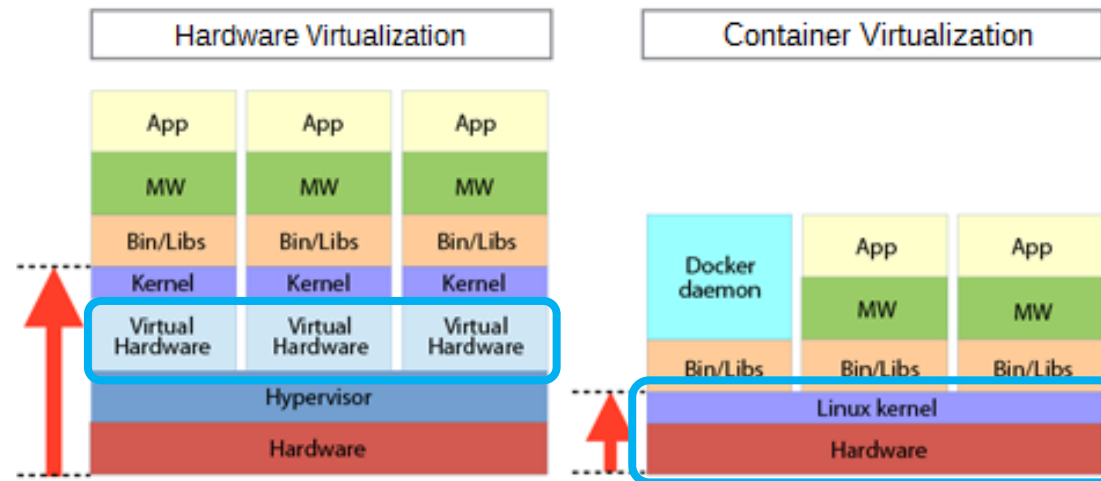
出典:vmware BLOGS
https://blogs.vmware.com/jp-cim/2014/10/vsphere_kiso07.html

基本的に物理マシンにできることは仮想マシンでもできる

2. VMとdockerの違いとは？

② Dockerのコンテナのサーバ仮想化

- アクセスが制限されたファイルシステムをマウントしたプロセス内で独立した実行環境を構築



出典: Research at <http://research.worksap.com/research/docker-20140724-2/>

ベースOSと完全に分離してないため、**セキュリティ上の制約**が存在する。さらに**ハードウェア**や**Kernel**に影響する**プロダクト**は**利用できない**場合がある

2. VMとdockerの違いとは？

できないことの一例

- fdiskコマンドやSoftware iSCSIやDRBDなどディスクデバイスに対して操作が発生する製品
 - dockerコンテナ内からでは/dev/sdxなどのデバイスを参照することも操作することもできない
- systemctlなどのサービスを制御するコマンド
 - RHEL 7.x系ではサービスの起動、停止を行うsystemctlコマンドがコンテナ内では使用できない
- rebootなどのサーバを再起動するコマンド
 - systemctl不可を再起動で回避することもできない

docker環境では**ハードウェアはベースOS上で管理**し、コンテナ内で利用するサービスは**コンテナ生成時にdockerfile等でインストール**を行うのが通常手段である

1. **docker**とは？

2. **VM**と**docker**の違いとは？

3. dockerの**使われ方**は？

- ① 開発環境の事例
- ② 情報提供サイト運用の事例
- ③ 大規模コマースサイトの事例

4. まとめ



docker



vmware



3. dockerの**使われ方**は？

① 開発環境の事例

● パッケージ製品の保守開発環境への利用事例

➤ 導入前の問題

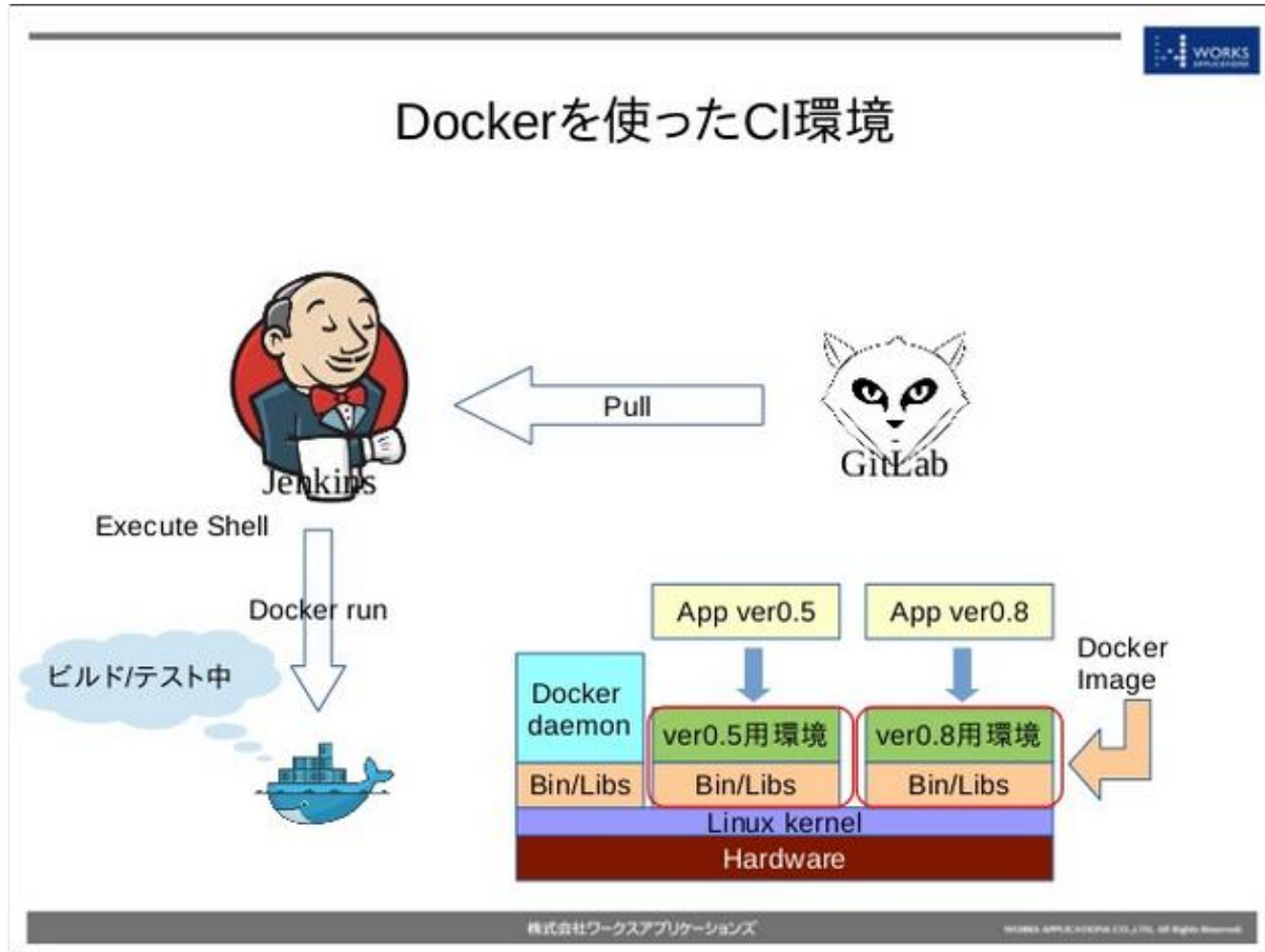
- 多数のパッケージ製品の開発検証環境が必要
- 1つのパッケージ製品に対しても新旧バージョンの保守・開発環境が必要
- 国内・海外の開発拠点の環境の共通化が必要

➤ 導入後の効果

- 各製品の各バージョンの検証環境が統一され、**製品の品質が安定**
- **開発環境維持コストを低減**できる
- 使用するリソース量が削減でき、開発者個々に**自由な開発環境を提供**

3. dockerの**使われ方**は？

① 開発環境の事例



引用：株式会社ワークスアプリケーションズ 遠藤博樹氏の講演資料
 (<http://www.slideshare.net/endhrk/docker-use-case-36473690>)

3. dockerの**使われ方**は？

② 情報提供サイト運用の事例

● 情報提供サイトの更新処理の自動化への利用事例

➤ 導入の為に実施した対策

- Webサイトのアプリケーションを更新する単位でコンテナ化
- システムの更新はコンテナの入れ替えで実施
- コンテナの変更は上位のnginxの設定変更で対応

➤ 導入後の効果

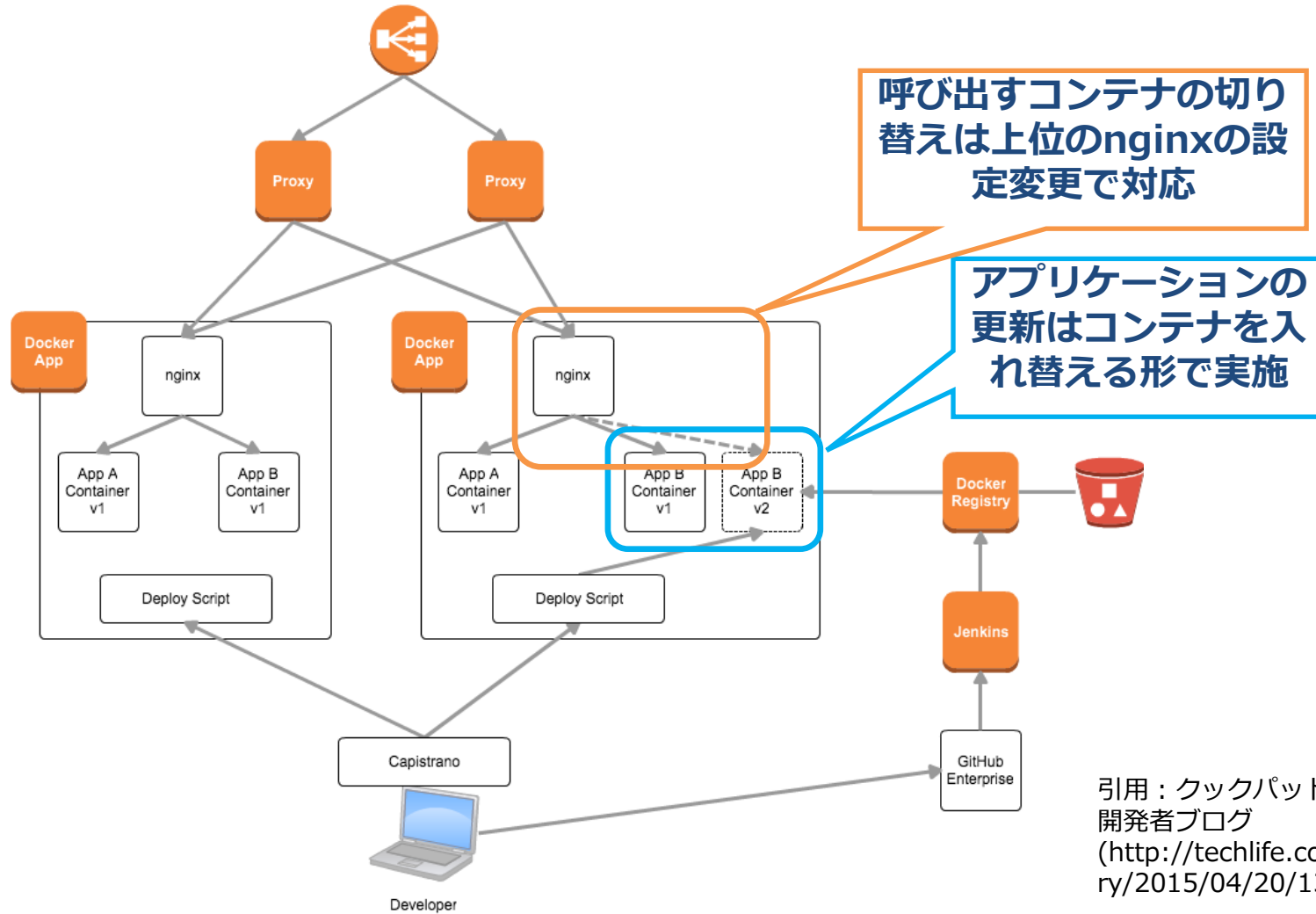
- **アプリケーションの更新および設定変更の自動化を実現**

➤ 今後の課題

- コンテナ配置のワークロードの自動化
- コンテナ増減によるオートスケールの実現

3. dockerの**使われ方**は？

② 情報提供サイト運用の事例



3. dockerの**使われ方**は？

③ 大規模コマースサイトの事例

● 10万テナントの大規模コマースサイトの運用



Docker at Shopify: How we built containers that power over 100,000 online shops



Graeme Johnson
November 18, 2014

This is the second in a series of blog posts describing our evolution of Shopify toward a Docker-powered, containerized data center. This instalment will focus on the creation of the container used in our production environment when you visit a Shopify storefront.

Read the first post in this series [here](#).

引用 : Shopify Inc 技術ブログ(<https://engineering.shopify.com/17489060-docker-at-shopify-how-we-built-containers-that-power-over-100-000-online-shops>)

3. dockerの**使われ方**は？

③ 大規模コマースサイトの事例

➤ 導入方法

- コンテナ内部から不要な機能を除き、最小化
- コンテナの構築をDockerfileで自動化
- アプリケーションをDockerコンテナで動作させることを前提とする Containerizingを行った
- アプリケーションを1つのコンテナに集約した

➤ 導入後の効果

- 利用申し込みからテナントへのサービスの提供の自動化を実現
- サービス提供までの時間を実現
- リソース使用量の低減にも成功
- 現在は20万件以上のテナントにサービスを提供

4.まとめ

- dockerはOS内に独立した実行環境を構築できる
 - **実行環境をカプセル化**できる製品
- コンテナはリソース消費量が少ないので、**より多くの実行環境を同時に提供**できる
 - 同じものを繰り返し構築することに適している
- 仕様上できないことが結構多い
 - **アプリ・インフラ両面から解決**が必要
- 将来、**エンタープライズでの利用**が普通になる
 - dockerの開発も周辺ツールも開発もPaaS利用を前提としている
 - 実現には**フルスタックエンジニア**の参画が不可欠



TIS

IT Holdings Group

Go Beyond