

東京電機大学 コンテナ型仮想化

2019/12/11

Minehiko Nohara

nohara-m@macnica.net

Agenda

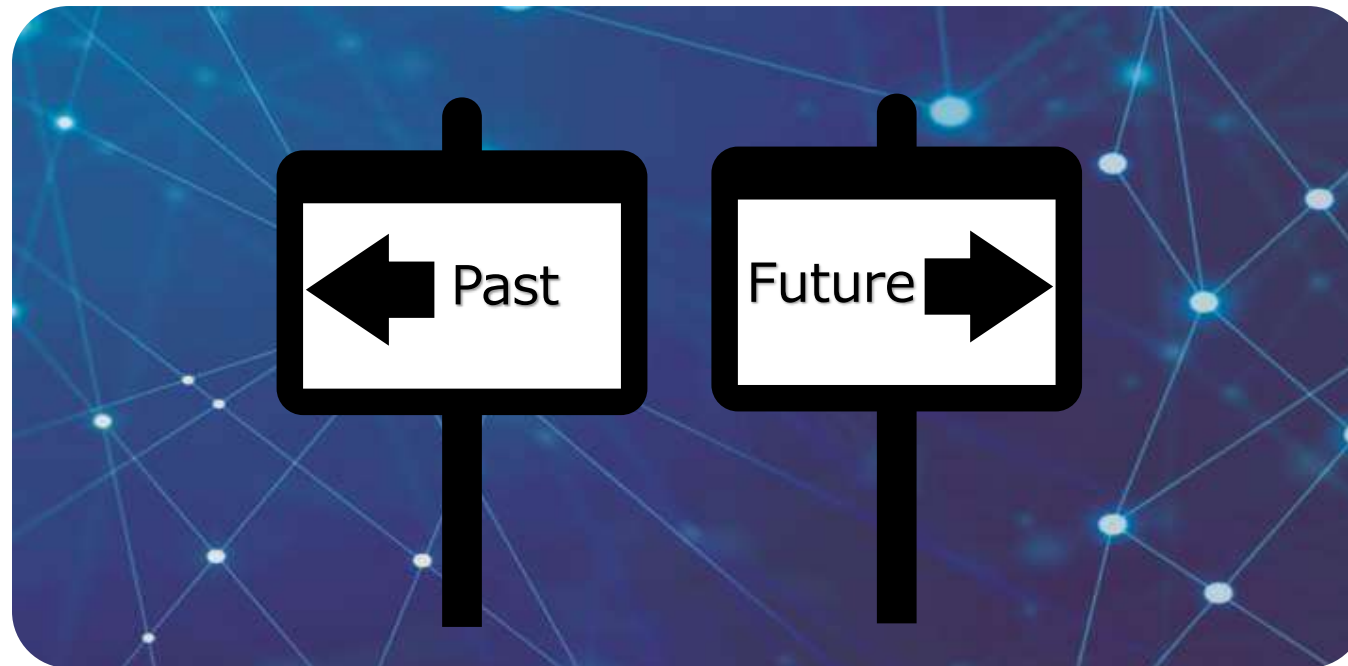
- 基本解説
 - 仮想化の歴史
 - コンテナの歴史と技術概要
 - ユースケース
- ハンズオン
 - dockerインストールの解説
 - 基本コマンド及びDocker Hubのアカウント作成
 - Dockerでアプリケーションをデプロイする
 - Dockerイメージをビルドする

自己紹介

- なまえ： 野原 峰彦
- 2006年にマクニカグループに入社
 - 自社SaaSサービス開発担当
 - Eメールアーカイブ製品担当
 - Web Analytics製品担当
 - 仮想分散型Firewall製品担当
 - オーバーレイ型SDN製品担当
 - Hyper Converged System製品担当
 - Web分離製品
 - コンテナ用Hyper-converged製品
 - 現在はMSCでDevOps系の商材全般の販売に関わる業務

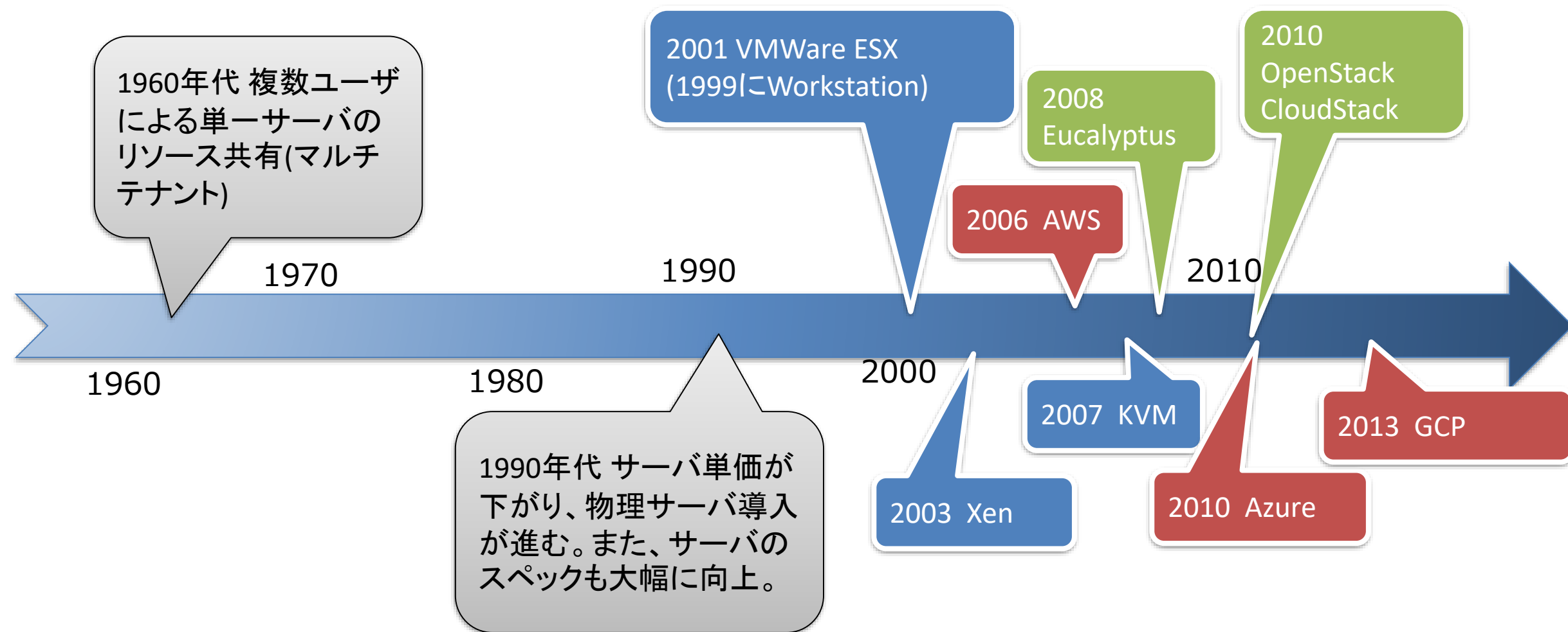


- パラダイムシフト ([英](#): paradigm shift) とは、その時代や分野において当然のことと考えられていた認識や思想、社会全体の価値観などが革命的にもしくは劇的に変化することをいう。パラダイムチェンジともいう。(出典: Wikipedia)



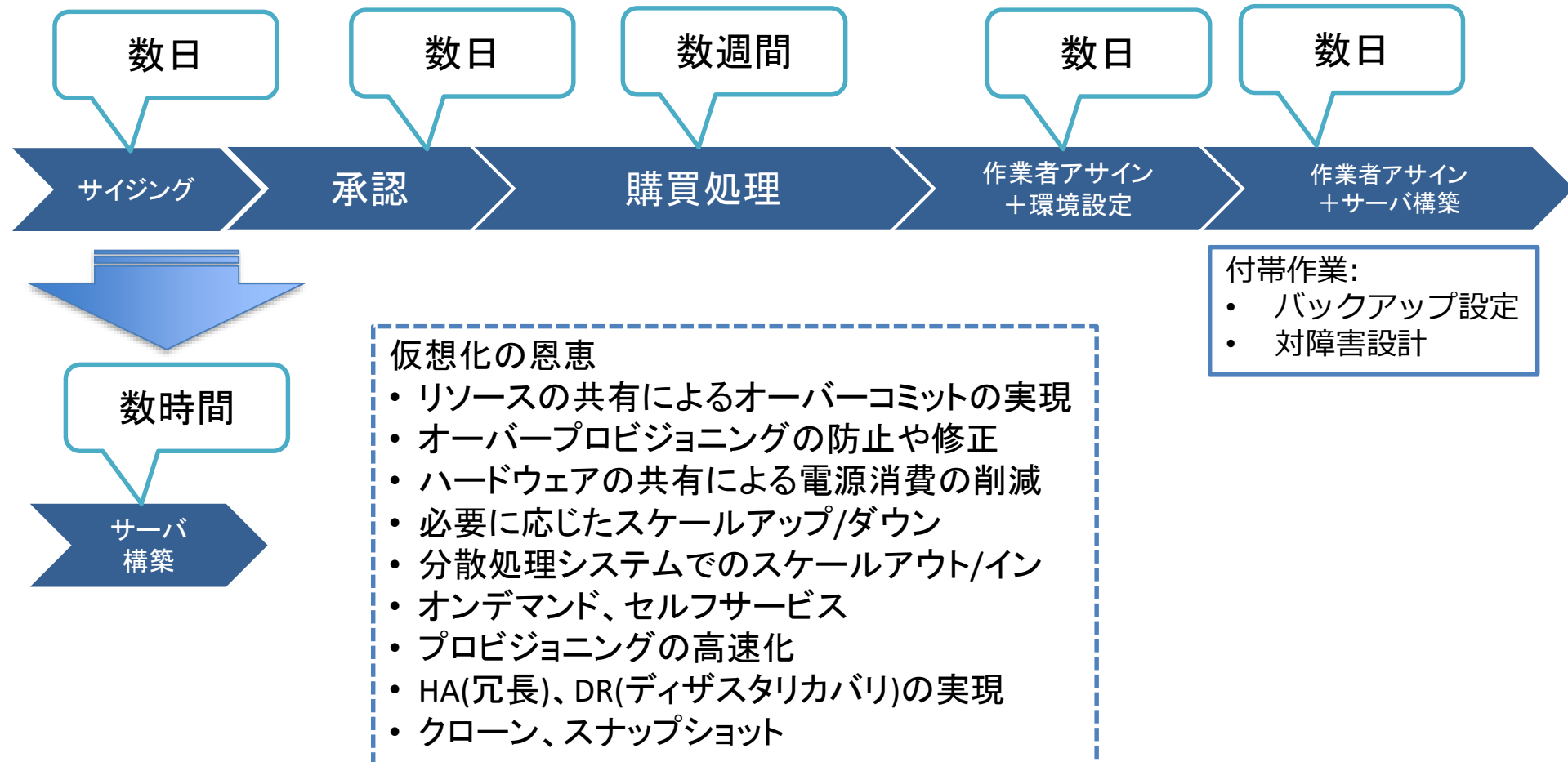
最初のパラダイムシフト/仮想化の歴史を振り返る

仮想化の歴史(サーバ仮想化系)



仮想化がもたらした業務効率の改善の例

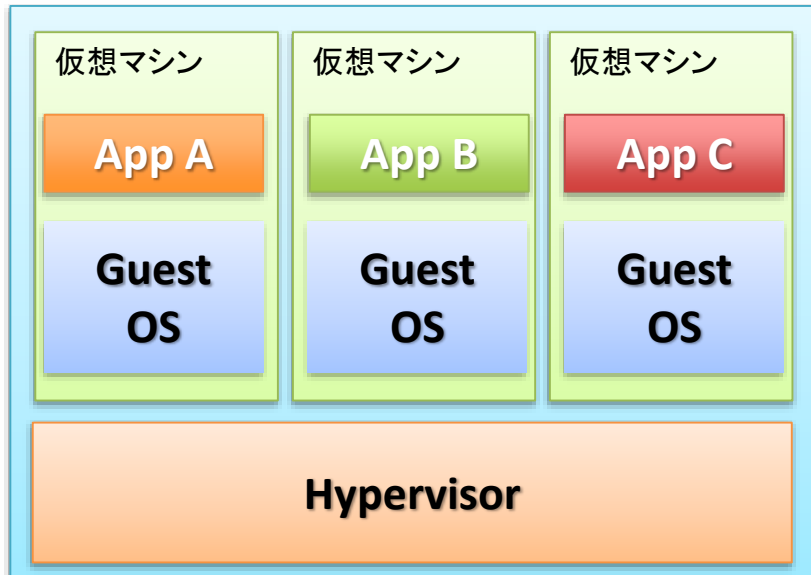
- サーバ購入、ネットワーク利用申請など、数週間に及ぶプロセスが、仮想化によって数時間で完結する時代に！



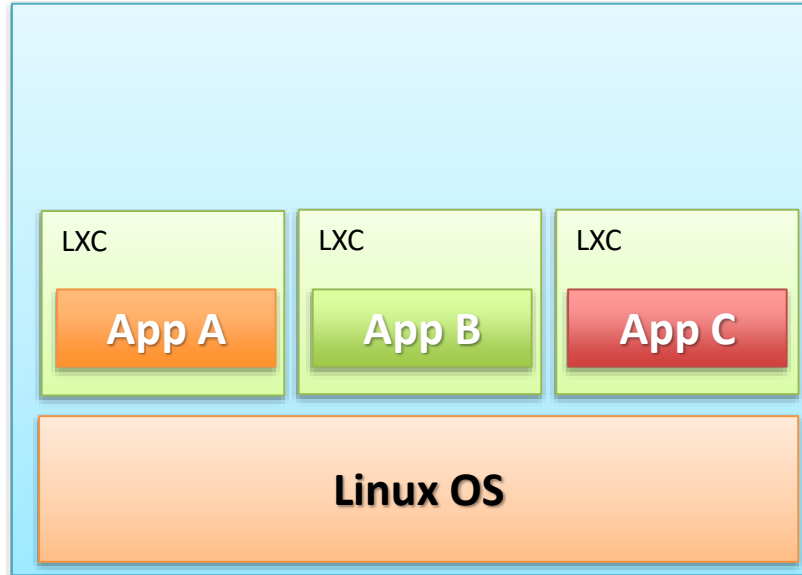
仮想化の歴史(サーバ仮想化系)



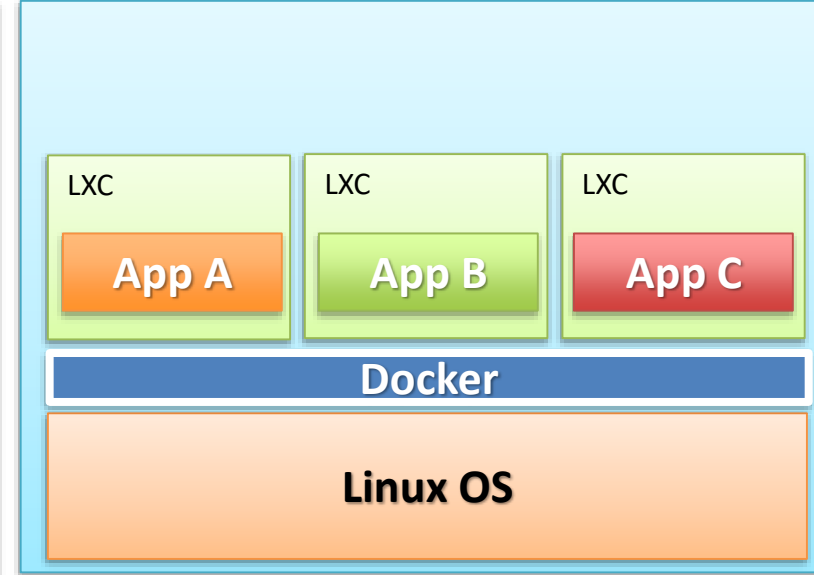
一般的な仮想化



LXCを用いた仮想化



Docker



仮想化の課題

■ ポータビリティ

- そもそもの容量が大きく、数～数十、数百GB単位のデータになる
- ハイパーバイザーごとの仮想マシンフォーマットやハードウェアのアーキテクチャに依存する

■ スケールアウトの処理速度とクローン

- リンククローンなど、ハイパーバイザー側の機能でカバー

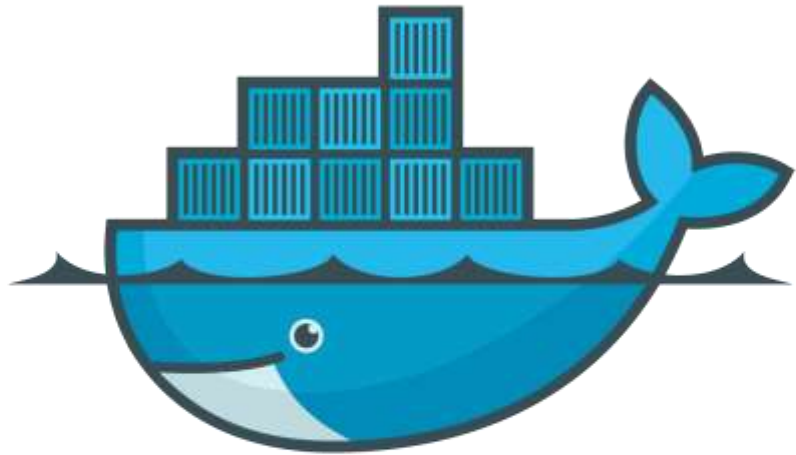
■ 起動とシャットダウンに要する時間

- OSの起動、終了に要する時間

■ 運用監視は物理環境と同様

- 仮想環境のオーケストレーターにより、各仮想マシンの一元管理は実現できるものの、運用監視のアプローチ自体は物理環境と同じ
- 特定のベンダーやテクノロジーにロックインされがち
- アーキテクチャ上、削減できる所要時間には限界がある
- 運用監視作業は大きくは改善できない

Docker



docker

dockerを
していますか？

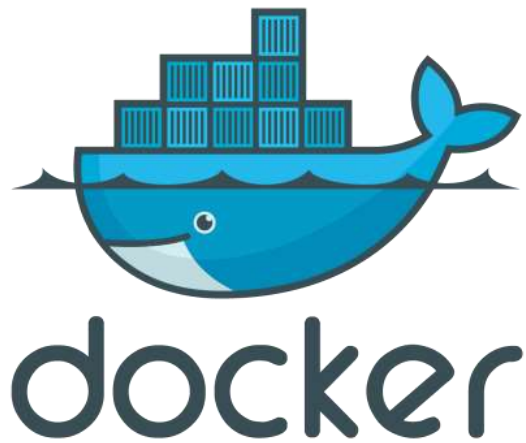
dockerを使ったことは
ありますか？

- DotCloud社(現docker社)が2013年のPyCon(Python Conference)で発表。その時点でのユーザーは5人。
- コンテナを利用するための統合管理ツールであり、従来のLXCにさまざまな付加価値を追加して提供している。
- オープンソースソフトウェアの「docker」として公開され、その使い勝手の良さから、多くの開発者、IT部門の管理者で利用されはじめ、特に2015年以降、爆発的にユーザーが増加している。
- 現在はコンテナの代名詞。
 - 世界コンテナ利用の八割以上がDockerを利用していると言われる。

dockerによるアプリケーションのデプロイメント

■ Wordpressをインストールする

- 面倒くさそう...
- Dockerをつかうとたった2行で完了！



インストール手順の詳細

手順 1: ダウンロード・準備

手順 2: データベースのインストール

手順 3: MySQLのインストール

手順 4: WordPressのインストール

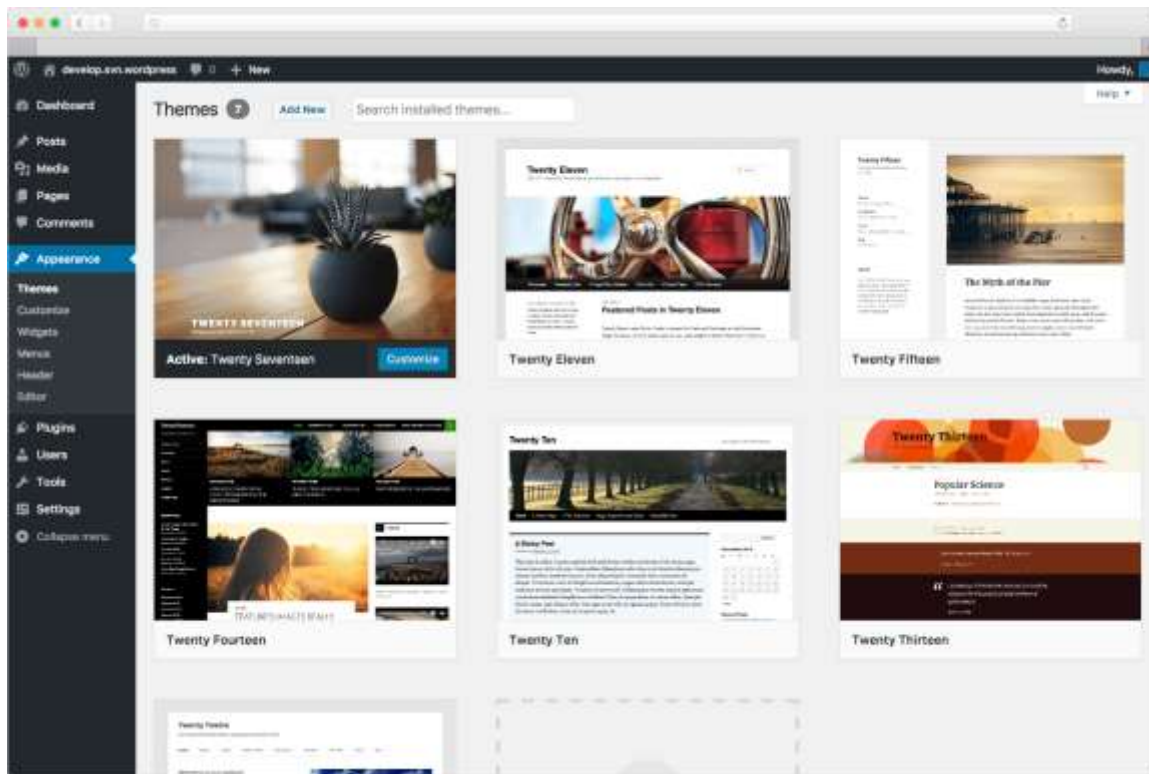
手順 5: インストールスクリプトの実行

インストール完了

インストールスクリプトを実行すると、WordPressのインストールが完了します。この時点で、WordPressのインストールが完了し、データベースの接続も正常に行われています。これで、WordPressのインストールが完了しました。

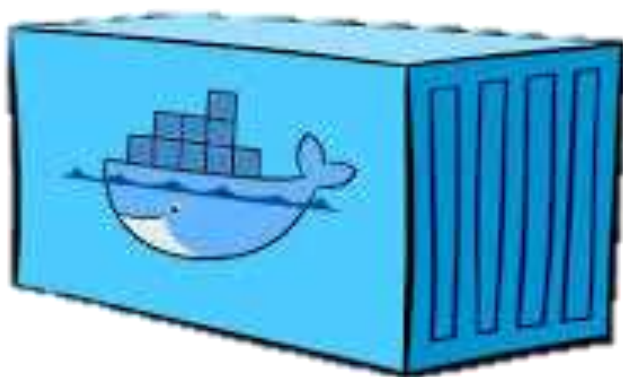
Wordpress

- ブログやニュース掲載などに使われる、非常にポピュラーなシステム
- 世界中で利用され、Web上の35%のサイトが使っているとされる
- データベースが必要と組み合わせて利用する



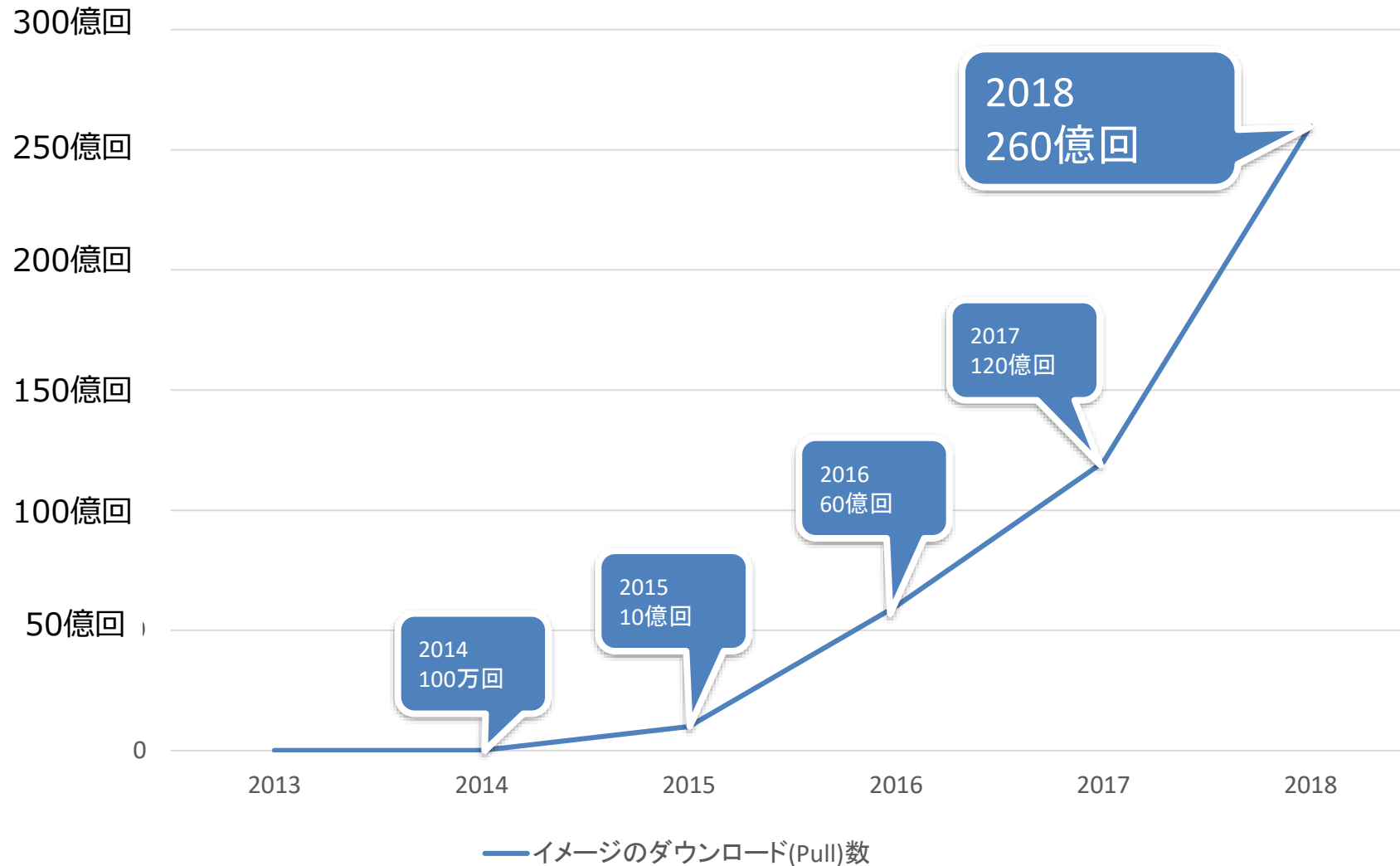
dockerの使い方、覚えるのが大変？

- **dockerには独特のコマンドがあり、独特の運用のクセがあるが:**
 - 比較的シンプルなので、簡単に覚えられます。
 - さまざまな3rd パーティツールを組み合わせでより簡単に利用することも可能です。

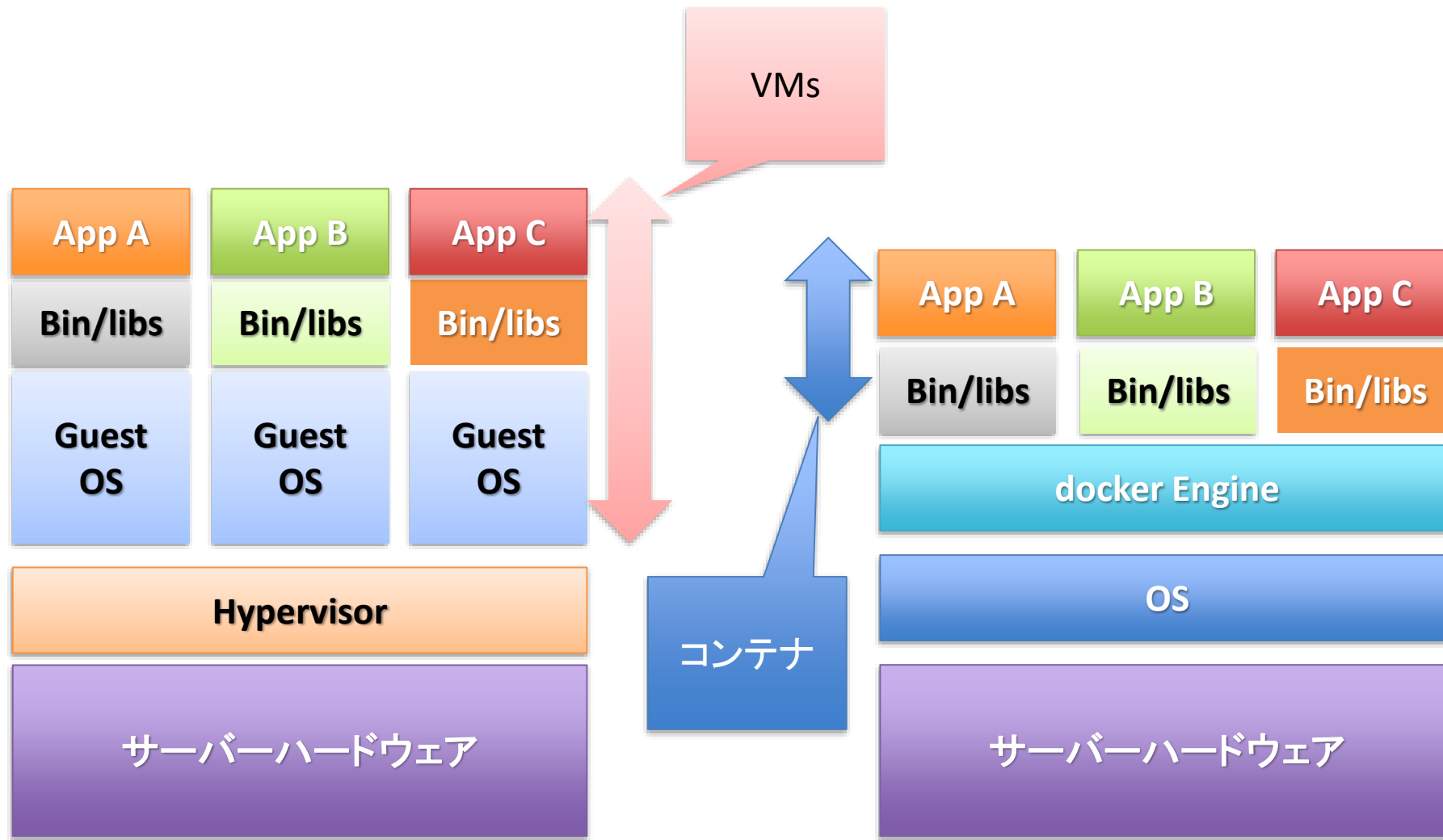


docker hub イメージダウンロード(Pull)数の推移

イメージのダウンロード(Pull)数

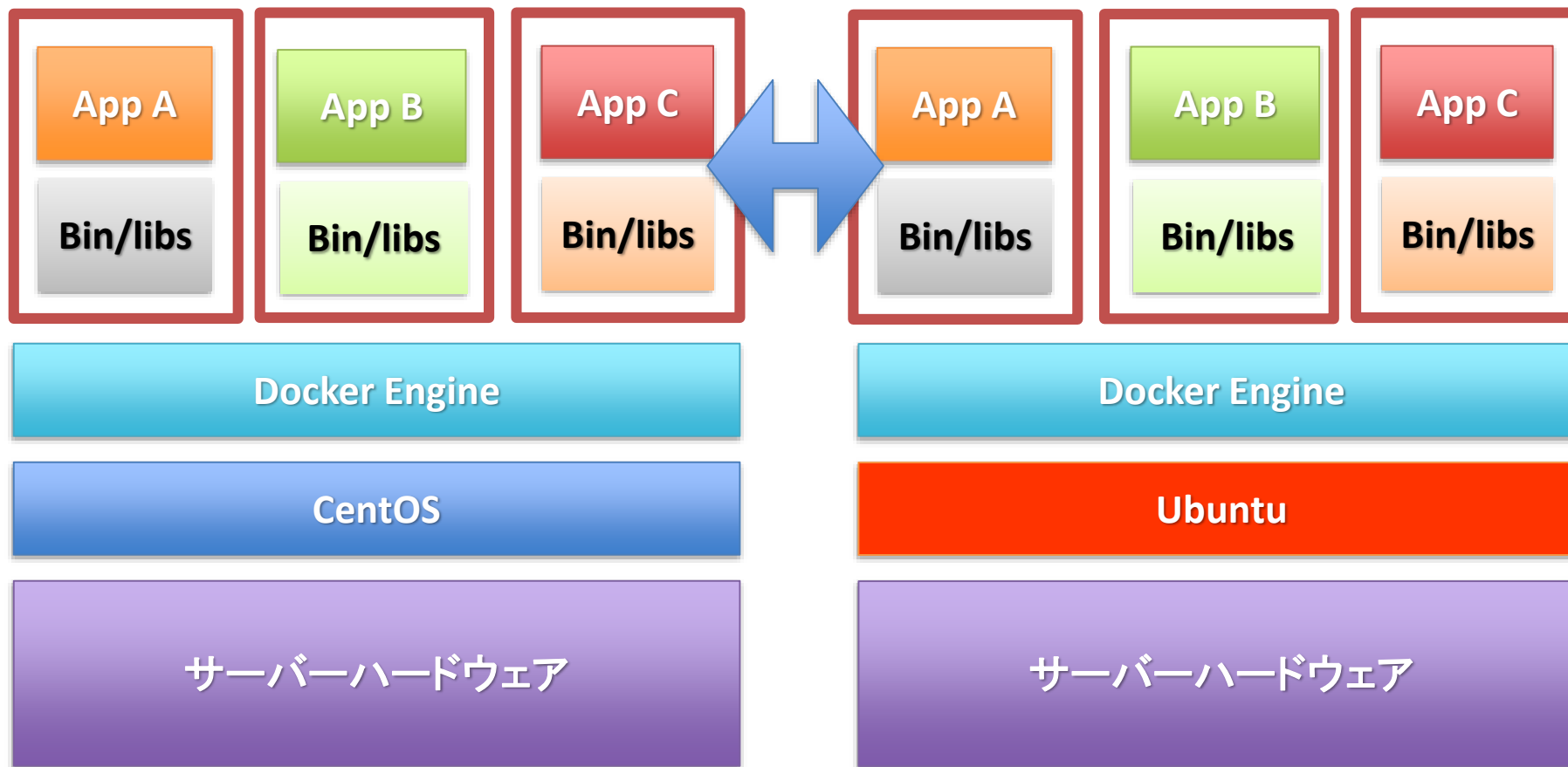


コンテナは軽量



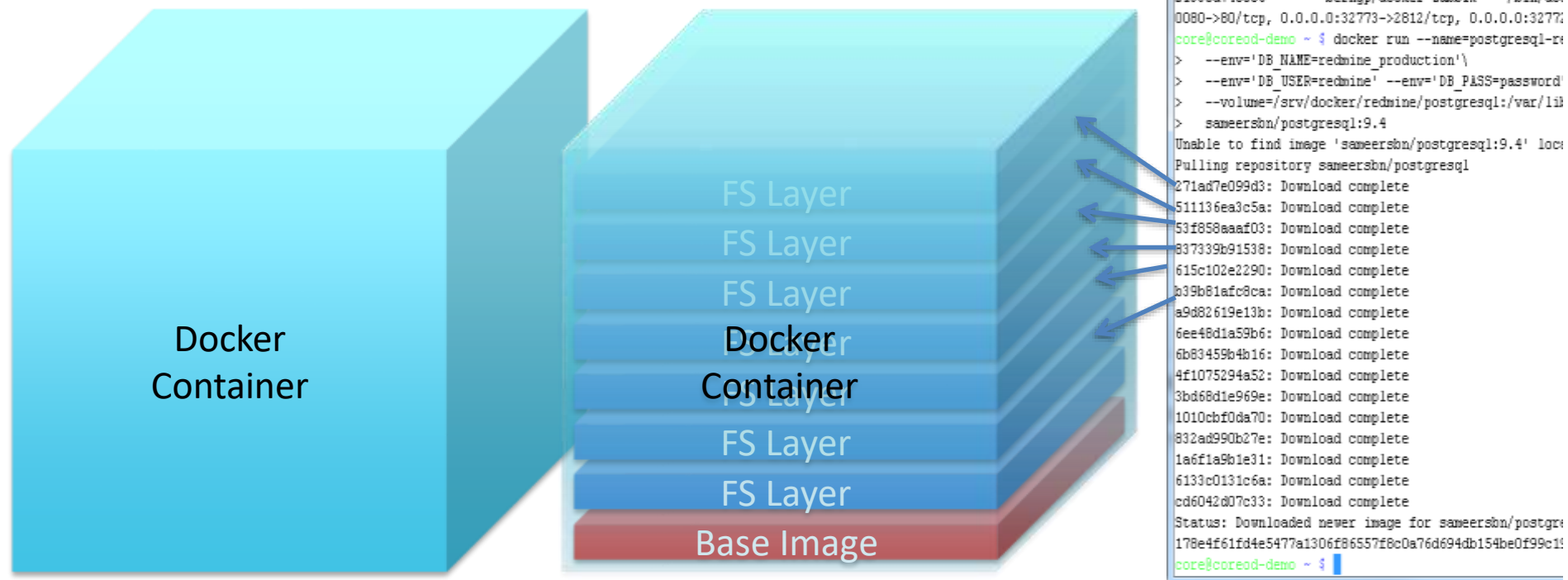
ポータビリティ

実行環境、Dependency問題を気にしないポータビリティ

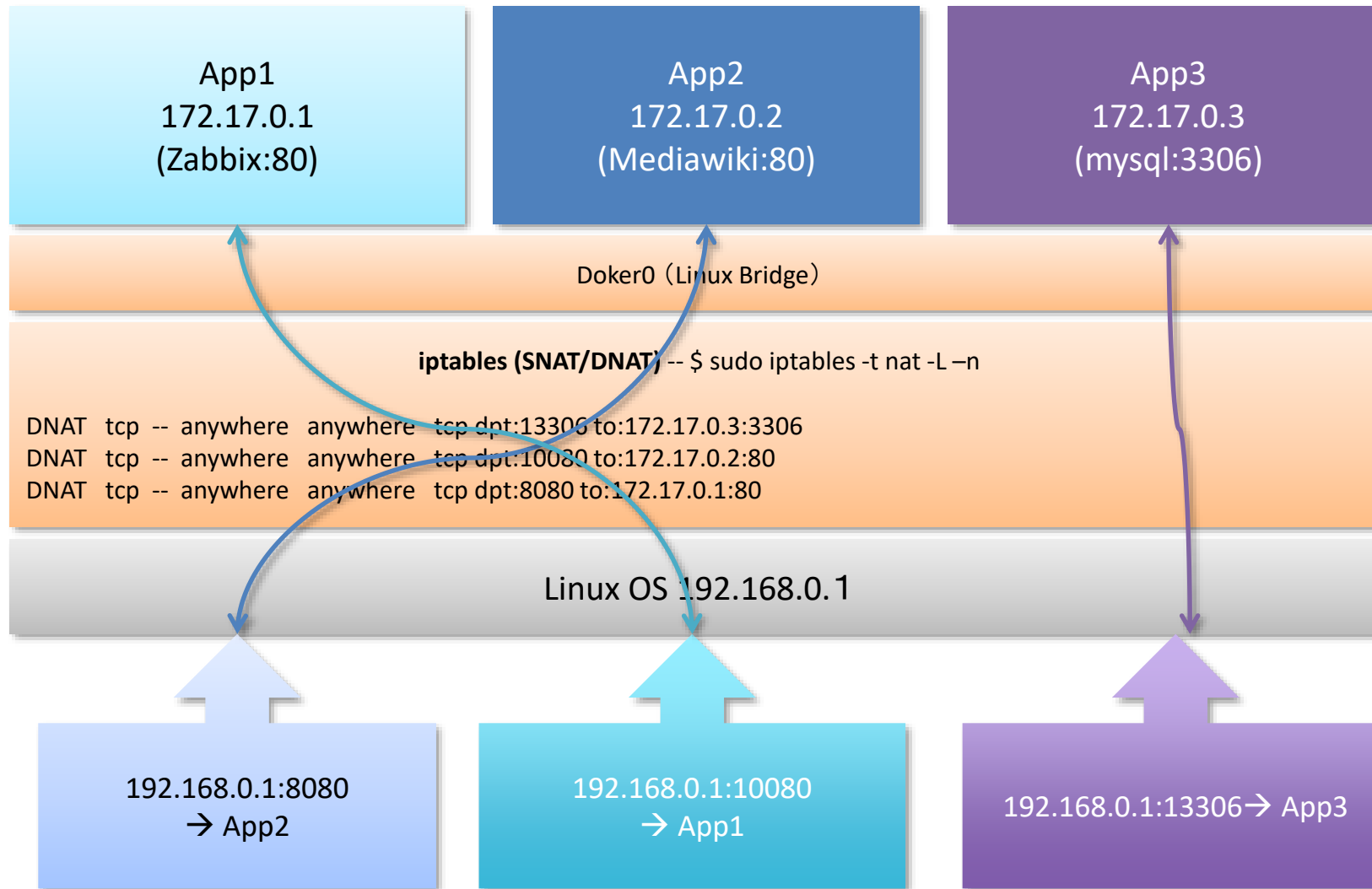


dockerコンテナの構成

Dockerコンテナは小さいイメージの集合体による、パッケージ化されたアプリ実行環境



一般的なdockerコンテナの接続イメージ



さまざまなコンテナRuntime

■ containerd

- <https://containerd.io/>

■ rkt

- <https://coreos.com/rkt/docs/latest/>

■ cri-o

- <http://cri-o.io/>

■ Google Cloud gVisor

- <https://github.com/google/gvisor>

■ Intel Clear Containers

- <https://clearlinux.org/containers>

■ kata

- <https://katacontainers.io/>

■ lxd

- <https://www.ubuntu.com/containers/lxd>

■ pouch

- <https://github.com/alibaba/pouch>

■ OCI (Open Container Institute)

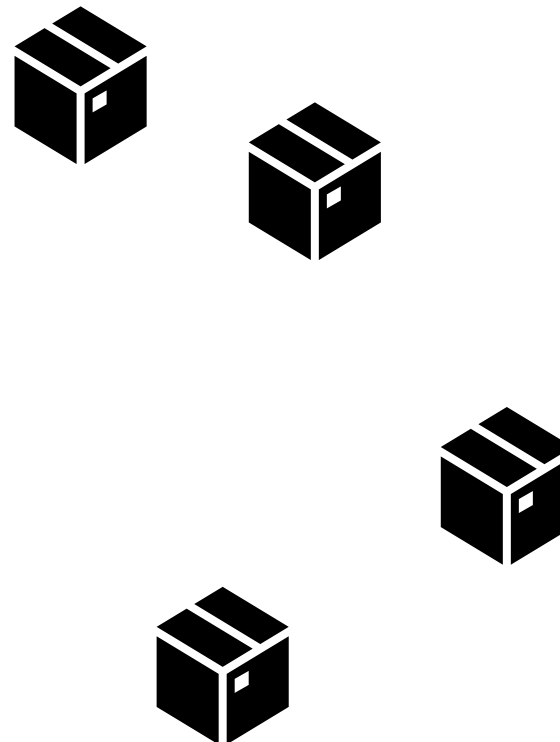
- <https://www.opencontainers.org/>

■ runV

- <https://github.com/hyperhq/runv>

■ Singularity

- <https://singularity.lbl.gov/>



オーケストレーションツールの登場

- 複数ホストのクラスタ化
- アプリケーション(POD)のオートヒーリング
- 認証基盤
- マルチテナント
- ホスト間のネットワーキング
- トラフィックのバランシング etc.



kubernetes

コンテナテクノロジーによってもたらされる変化の例

■ 開発者作業の効率化と自動化

- ビルドの自動化
- テストの自動化
- パッケージングの自動化
- コンテナ単位で任意の言語、任意のツールセットを利用可能

■ 運用者作業の効率化と自動化

- 消費リソースの削減
- インフラおよびアプリのデプロイメントのコード化による自動化
- 障害発生時の自動復旧
- 運用監視用に標準のAPIが利用可能

■ 環境の変化 (新たなツールや新たな学習コスト)

- コンテナの開発とデプロイメントのツールやフローの変更
- コンテナのライフサイクル管理と監視
- コンテナのアーキテクチャに適した新たなセキュリティ対策の導入

ユースケースの紹介

コンテナのユースケース例

■ 特定サービスの軽量化(マイクロサービス)

- アプリケーションのコアサービスやDBのインターフェイスを細分化し、独立したコンテナで提供する
 - リソース管理の効率化
 - インストール、アップデートの簡素化
 - 容易なスケール、迅速なエラー処理や再起動

■ Web Server/スマホアプリ

- アクセスの上下 変動が激しく、画一サービスを提供するコンテンツ
- 軽量なため、数百、数千単位までスケールさせて利用するケースも
- 問題がでたものは随時削除してデプロイしなおしを行う

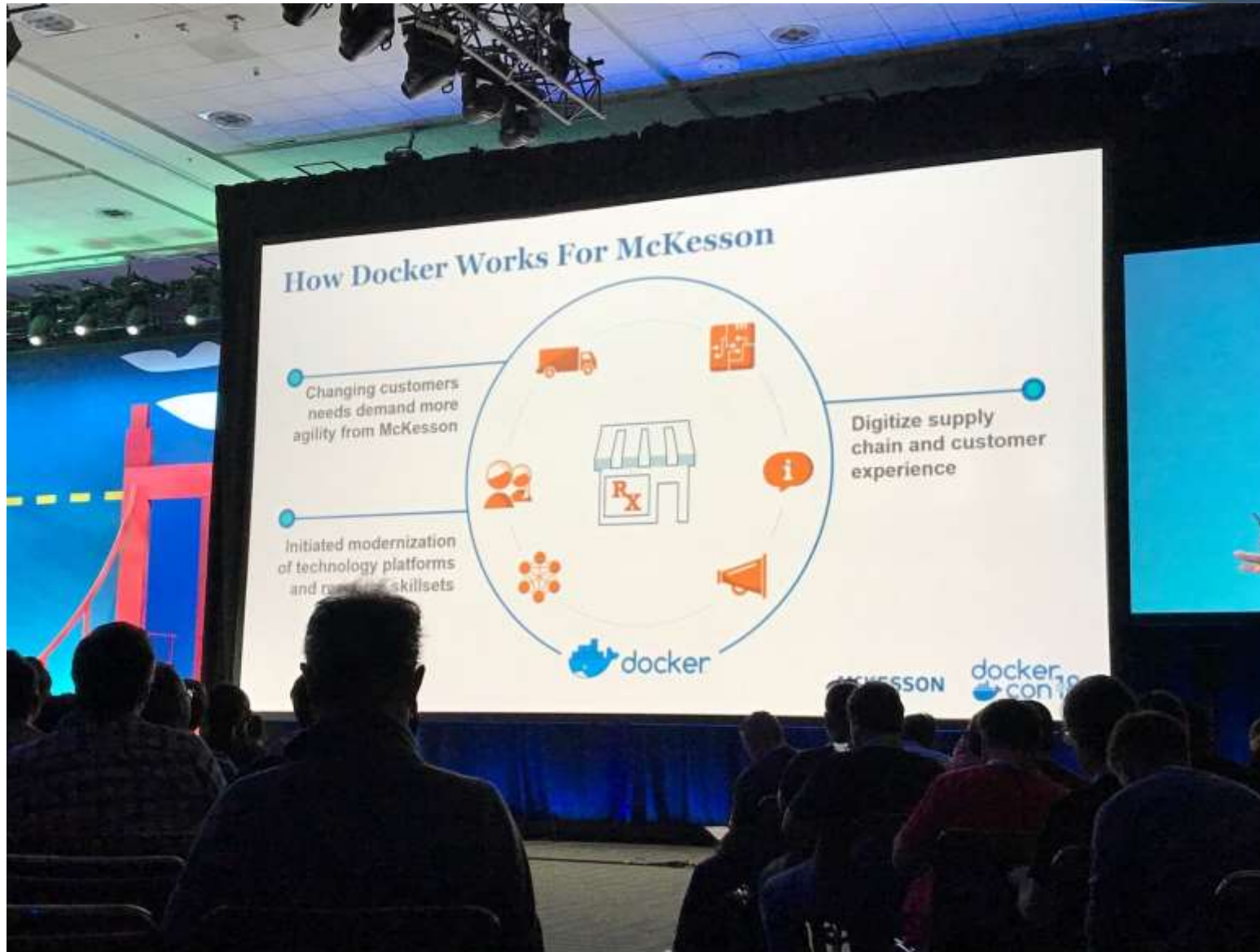
■ DevOps実現ツールのひとつとして

- 環境依存問題を大幅に削減できるため、テスト効率がよい
- Dockerのようにポータビリティ性のあるコンテナは、あらゆるプラットフォームにおいて同一の環境を共有することが可能
- マイクロサービス化によって疎結合されたコンポーネント単位でのテストやアップデート

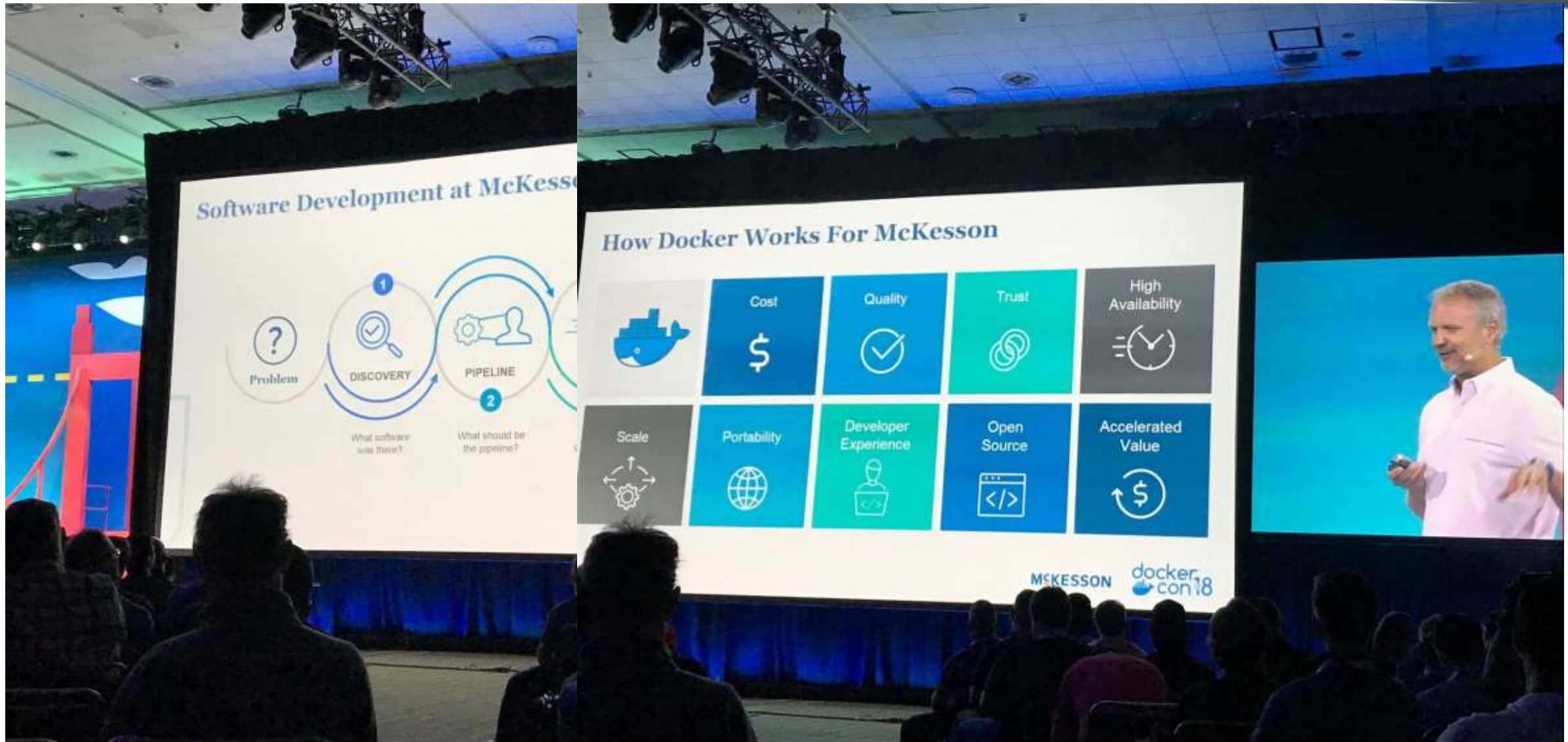
■ 試験用、評価用ツールとして

- すぐに使ってみたいアプリケーションをイメージからインポート
- 短時間で簡単に構築。リソース消費も極すくない
- バックアップも容易に小容量で取得。マイグレーションも容易。

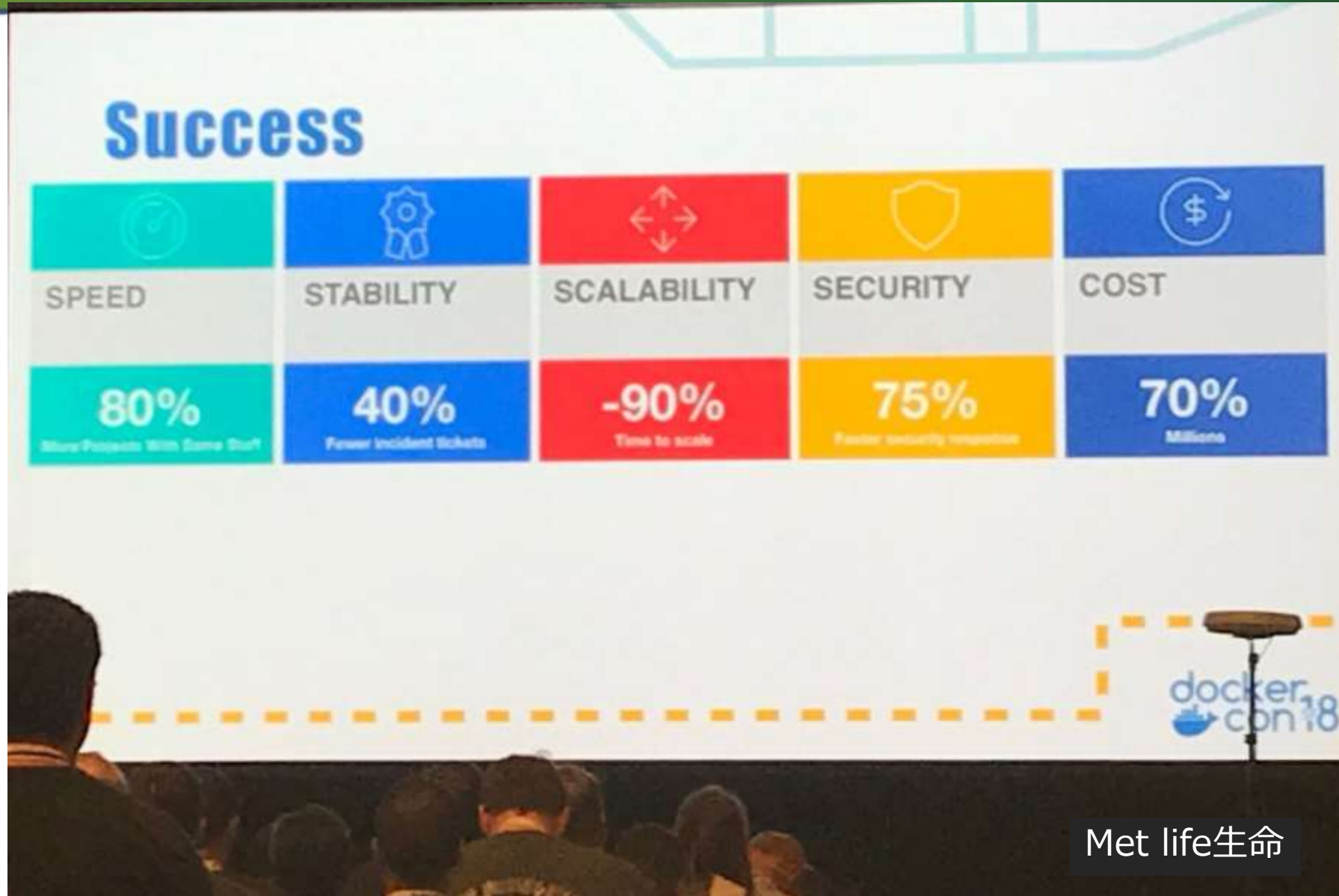
ユースケース紹介1 McKesson



McKesson 開発パイプラインの自動化



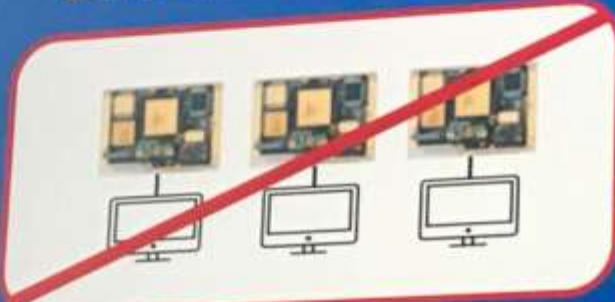
ユースケース紹介2 Met Life



ユースケース紹介3 NASA

docker con 18

What Are We Trying to Solve?

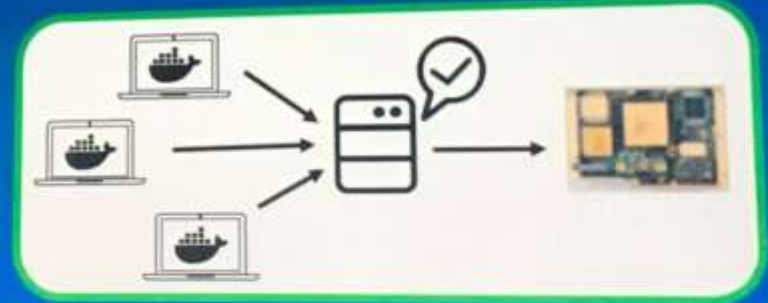


Hardware Scarcity

- Testbeds cost > \$300K
- Configuration management is painful
- Every developer/subsystem wants one

What is the holy grail?

- Hardware emulation!
- Develop in software land
- Test on real hardware
- CD to other teams/real spacecraft

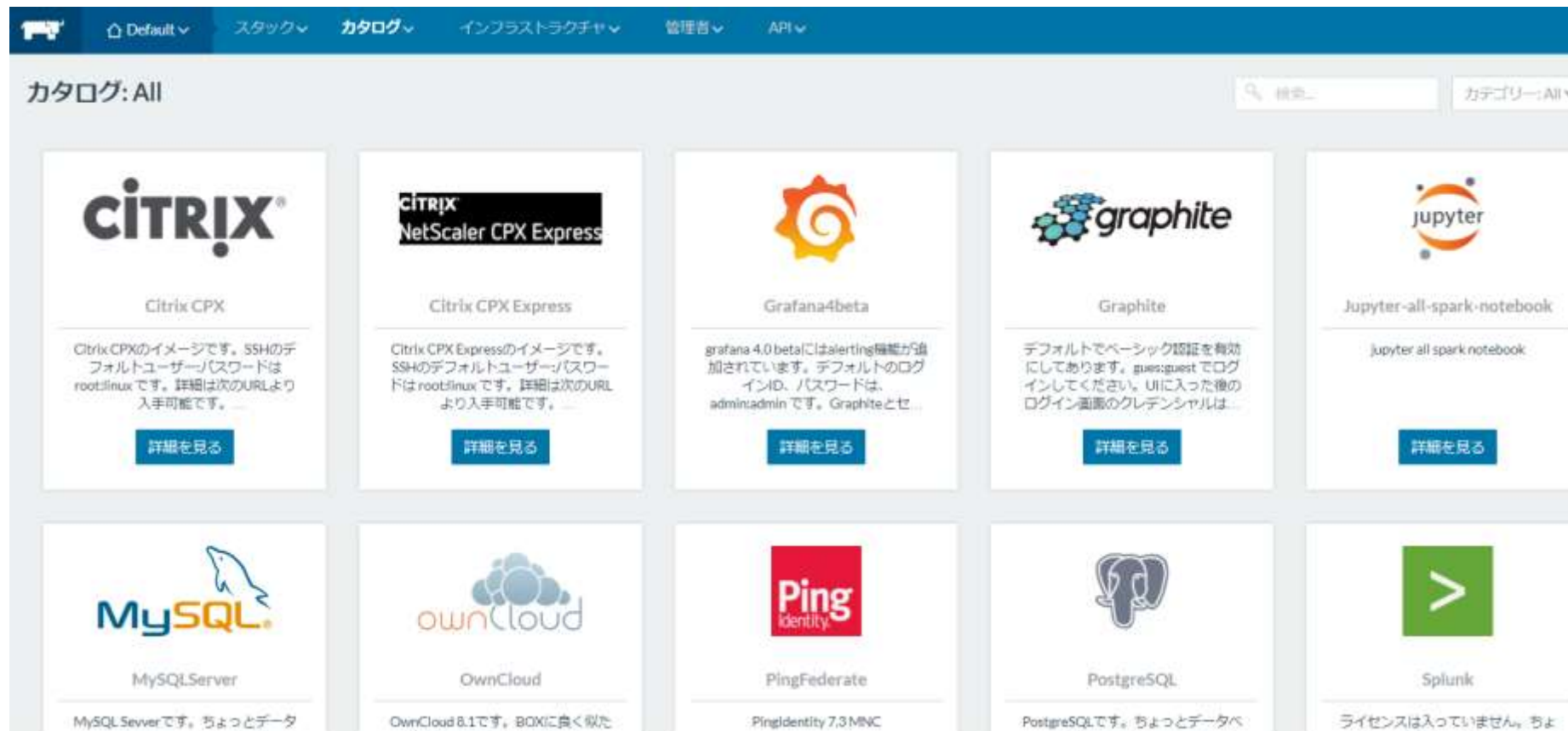


ユースケース紹介4 Solo.io



ユースケース5 マクニカネットワークスの利用事例

- Rancherというソフトウェアを利用したApplicationのマーケットプレイス。
- 従来各エンジニアが仮想マシンなどで都度構築していたアプリをオンデマンドで提供。



ハンズオン

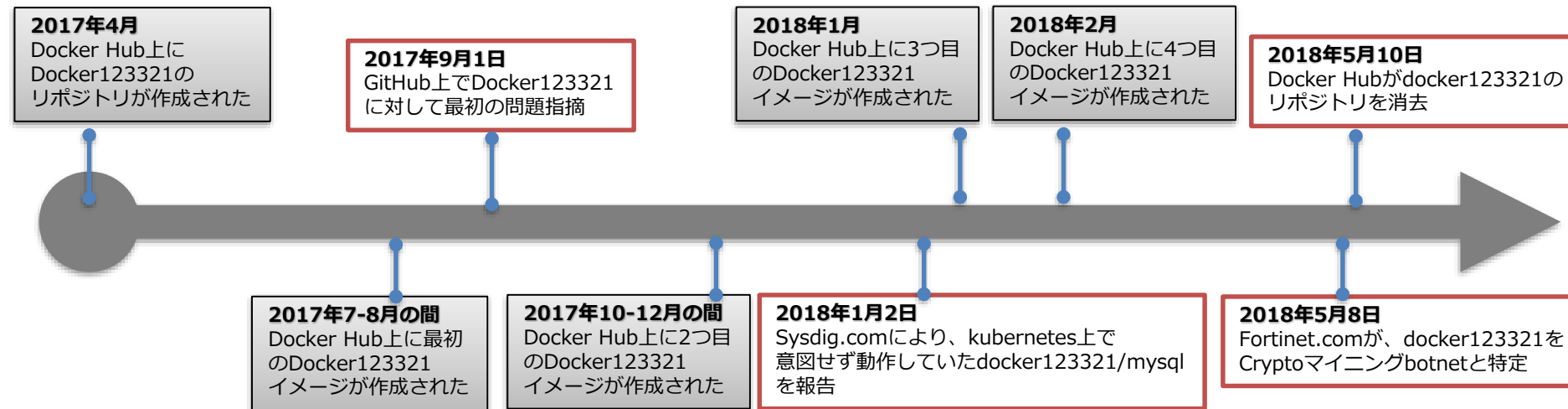
■ ハンズオンテキスト

<http://bit.ly/36jtyiR>

コンテナのセキュリティ

【参考】コンテナに関連したセキュリティニュース1

- 2017年3月時点で、Docker Hubのイメージの24%に重大な脆弱性との診断
- 2018/6 Docker Hubでマイニングスクリプトが仕込まれたイメージがニュースに
 - 9万ドル相当の仮想通貨マイニングが行われたと推定される
 - 2019/5にも同様のマイニングスクリプトを仕込まれたリポジトリが問題となった



【参考】コンテナに関連したセキュリティニュース2

■ 2018/12 kubernetesの深刻な脆弱性が公表された(CVE-2018-1002105)

- Aggregated APIサーバーの実行環境で、本来フルアクセス権限が与えられていないユーザーを権限昇格させることを可能とする脆弱性
- 次のバージョンが影響対象
 - Kubernetes v1.0.x-1.9.x
 - Kubernetes v1.10.0-1.10.10 (v1.10.11で修正)
 - Kubernetes v1.11.0-1.11.4 (v1.11.5で修正)
 - Kubernetes v1.12.0-1.12.2 (v1.12.3で修正)
- オリジナル情報
<https://github.com/kubernetes/kubernetes/issues/71411>

■ 2019/2 RunCに深刻な権限昇格の脆弱性(CVE-2019-5736)

- 本脆弱性を悪用して細工したコンテナをユーザが実行した場合、ホスト上のruncバイナリが意図せず上書きされ、コンテナが起動しているホスト上でroot 権限でコマンドが実行される恐れがある
- 影響対象
 - runc1.0 rc6 およびそれ以前
- JPCert情報
<https://www.jpccert.or.jp/at/2019/at190007.html>

コンテナ環境におけるセキュリティ

■ コンテナ環境にもセキュリティが必要！！

- コンテナの本番環境運用において、セキュリティは特に難しい問題です。共有される**ホストOSカーネルの完全性**と、その上で実行される**コンテナの完全性と分離状態**は非常に重要です。セキュリティが強化され、パッチが適用され、**最小限のOSをホストOSとして使用**し、脆弱性やマルウェアが**継続的に監視**され、信頼できるサービスが確実に提供されるようにする必要があります。

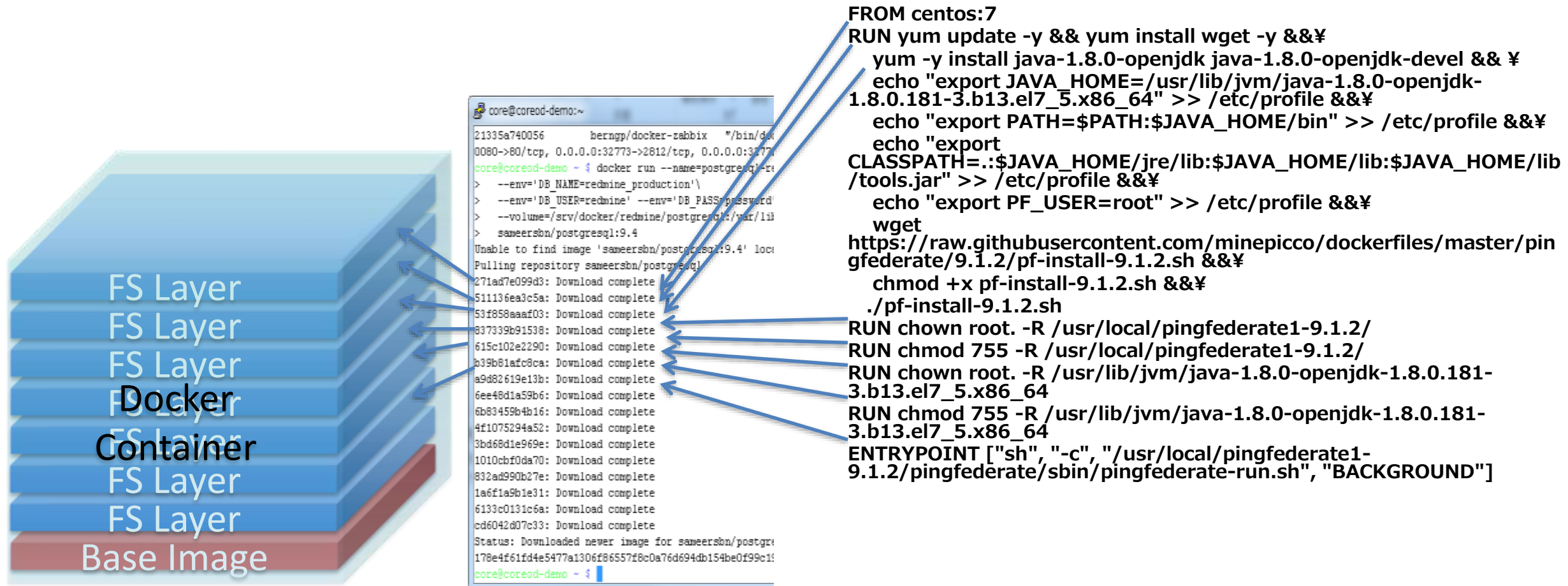
出典: <https://www.gartner.com/smarterwithgartner/6-best-practices-for-creating-a-container-platform-strategy/>



コンテナ、コンテナ実行環境の特性

コンテナ(docker)のアーキテクチャ(docker)

パッケージ化の中にアプリケーションや依存コンポーネントが入る



docker起動中のファイルシステム

展開されたファイルシステムはコンテナとの紐付けが困難

```
root@11284rancher2: /  
root@11284rancher2:/# ls -lah /var/lib/docker/volumes/  
total 380K  
drwx----- 77 root root 12K Sep 14 11:20 .  
drwx--x--x 15 root root 4.0K May 25 10:24 ..  
drwxr-xr-x 3 root root 4.0K May 22 19:28 032500152def76528fc692eb98fd6d094035a89d5453346fcecbbf560da17e0dd  
drwxr-xr-x 3 root root 4.0K May 9 15:13 047f3810c7729592a6aba2a3a80bd0ae5fb5a8587813a51803bf060c6896f614  
drwxr-xr-x 3 root root 4.0K May 15 09:19 0542fb8e57bb05edafe19168b650038ffcb6ad6428e6d20ad77cd7895b7590dc  
drwxr-xr-x 3 root root 4.0K May 22 20:28 07ead5e49819950809cd8c144c74331c259708ac32aaa89b9b9d531ea511e152  
drwxr-xr-x 3 root root 4.0K May 22 19:10 08956cd34da3a111c737959b02df15747d5dd0fe5348c894cc12b5467e35d0d8  
drwxr-xr-x 3 root root 4.0K May 8 20:38 09f519f01d20090f0dcbd1bfe45b15eaec1b494236137605e6c0a2696e65aae4  
drwxr-xr-x 3 root root 4.0K May 15 13:17 0a63286609bb62357ef4e9c3cdab70f35eac8a5b79e4fcacc08d044d3ea5c09e  
drwxr-xr-x 3 root root 4.0K May 14 17:36 0b33915d1309d1843e12a141e7d91296e0ff8779b0a1f4807c5e361953d711ee  
drwxr-xr-x 3 root root 4.0K May 9 15:33 103334132b14aa15c787752649cfead16846087d5703b92bfb8dcc22b9c470de  
drwxr-xr-x 3 root root 4.0K May 9 15:27 117d3c34613f7fa5bb88c93dc9229c649d3a4f5e7841c9e0896dc7d3f38b9654  
drwxr-xr-x 3 root root 4.0K Sep 14 11:20 1cb350e96c0c3341863b448bb80670f2da3f0ca90a245abc3333407f98e62262  
drwxr-xr-x 3 root root 4.0K May 14 17:35 1dffa0a0af6aa8bdeeeef073a42207b98f8de1f2d603dec39906d77b748ed1663  
drwxr-xr-x 3 root root 4.0K May 22 19:28 1f2d8cc838d501c30aa54d0bec16b81f60f23a844a402cc44429f2331ed9703  
drwxr-xr-x 3 root root 4.0K May 15 11:39 21dd0fa6b7c5121d6da724614499151659f486d620c7dacd67549d1ae536522a  
drwxr-xr-x 3 root root 4.0K May 22 19:10 2561ab3b78f3d54863ea7a3d620c5af66dba2e0090bd27bd9f86996761567ce3  
drwxr-xr-x 3 root root 4.0K Sep 6 14:27 260290746eeda3f4cdbaa33c4a1d6f5a218f4175b8b601588042e2633774140b  
drwxr-xr-x 3 root root 4.0K May 9 14:22 2960bb2fa717559f7080a2e131a66f4faa9c6a23d9f1f3852d0990ec790d204e  
drwxr-xr-x 3 root root 4.0K May 22 18:59 2dfbffa6b8eb21c7c3c094151d64972216682ab9b9e25ce5790f4cb4c563ed1a  
drwxr-xr-x 3 root root 4.0K May 15 09:19 303b56c5301589ccd4217d880522c554b75dede778e3f79350c90d2cb6a860cc  
drwxr-xr-x 3 root root 4.0K May 8 20:37 312820c0d18ed1dabab2d100c615bbd680c00422ad4ccc2fa6d5ab06c265dc04
```

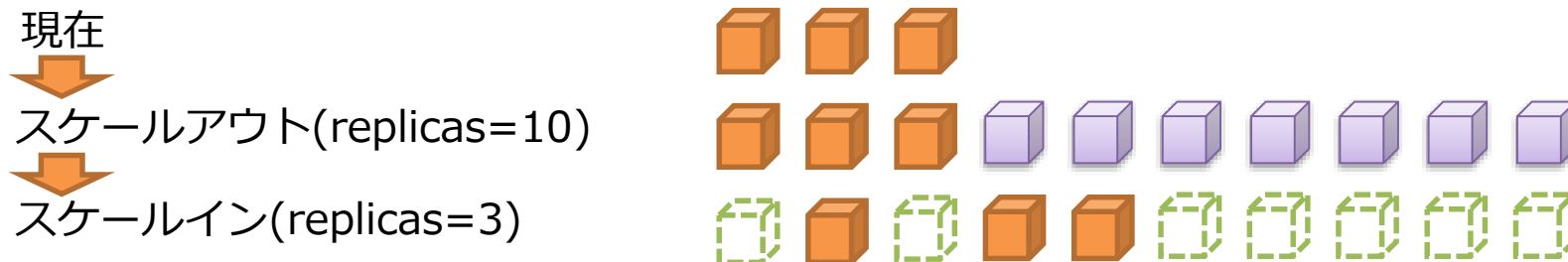

コンテナ環境のスケール

■ 自動/手動で任意の数にスケールアウト

- 配置先は明示的に指定しない限り自動にスケールアウト
- 運用またはスケールでフェイルがあれば、コンテナを再構築
- 原則動的なIPアドレスを利用

■ 自動/手動で任意の数にスケールイン

- どのコンテナがターミネートされるかはわからない



コンテナ環境のライフサイクル

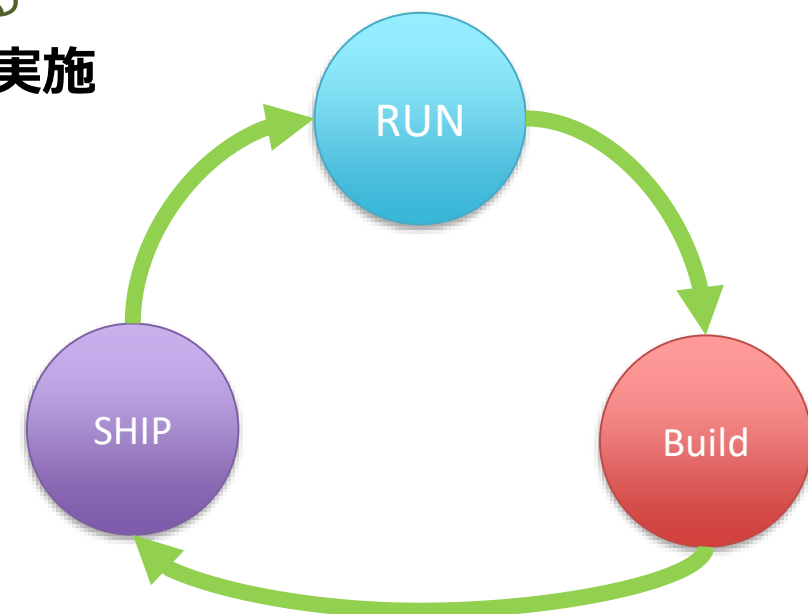
■ 設定変更/versionアップ方法が通常アプローチと異なる

- ソフトウェアバージョンのUpdateの方法は？
- パッチ適用/設定変更/機能追加/バグ修正の方法は？

元イメージをリビルドして入れ替える運用が一般的

※コンテナ環境は、画一的なイメージの利用を前提としている

※Blue/Greenやカナリーリリースにてサービス切り替えを実施



コンテナ実行環境の構成にも注意が必要

■ インフラ

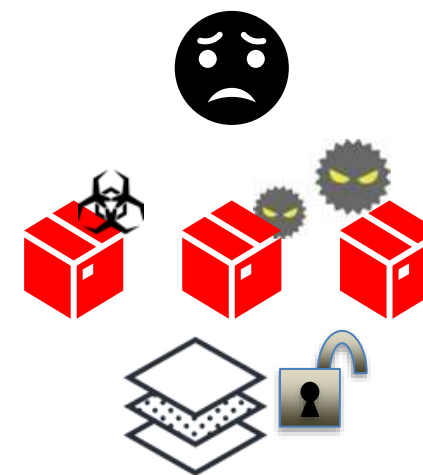
- ホストOSの選定や、その脆弱性への配慮
- ネットワークの構成

■ コンテナの構成

- Runtimeや関連ツールのバージョン
- docker.sockに対する権限
- ホストディレクトリのマウント状況
- 環境変数でコンテナに渡すクリアテキストのパスワード

■ コンテナオーケストレーター

- Kubernetesのバージョンや設定
- etcd、CoreDNSなどのコンポーネントのバージョンや設定
- Calico、Flannelなどのバージョンや設定など・・・



コンテナセキュリティのフレームワーク

コンテナセキュリティのフレームワークについて

■ NIST SP.800-190

- 2017年にリリースされた、コンテナ環境に対するセキュリティガイドライン
- ホストOS、コンテナの構成、特権ユーザー、ストレージ、ネットワークなど、コンテナ実行環境全体を対象としている

■ CISベンチマーク

- 様々な環境におけるセキュリティやコンプライアンスの状態を判断するための具体的な指標
- KubernetesやDocker、Linuxなど
- クラウド基盤を利用する場合は、クラウド基盤のCIS(AWS、Azure、GCPなど)を利用

- DevOpsをすすめると、コンテナの利用に行き着くケースは非常に多い
 - 親和性が高いツールであるため、利用するメリットは大きい
- コンテナ環境におけるセキュリティは、従来とは異なる
 - **NIST SP.800-190に即したセキュリティ**の検討が必要
- **DevSecOpsの実現**
 - Shift-leftで問題発生時のタイムラグを最小限に抑える
 - DevOpsへの影響を最小限に抑える
 - 運用環境のセキュリティポリシーを自動化することの検討
 - コンテナ本体は可能な限りシンプルに構成する
 - 問題発生時のデバッグをスムーズにし、影響範囲を限定的にする

NIST SP.800-190の一部抜粋紹介

NIST Special Publication 800-190

■ タイトル

- Application Container Security Guide

■ 著者

- Murugiah Souppaya (NIST)
- John Morello (Twistlock)
- Karen Scarfone (Scarfone Cybersecurity)



<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf>

- **docker(コンテナ)は今後のアプリケーション開発で利用されるテクノロジーの中心であり、スタンダードとなることが予想されます。**
 - 今後アプリケーション開発に関わる可能性のある方は、この周辺の技術を把握しておくことを強く推奨します！
- **ライフサイクル管理やアーキテクチャ上の違いにより、従来のセキュリティとは異なる対策が必要です。**

Thank you
nohara-m@macnica.net