Internet Week 2018

Docker コンテナの必要性と基礎

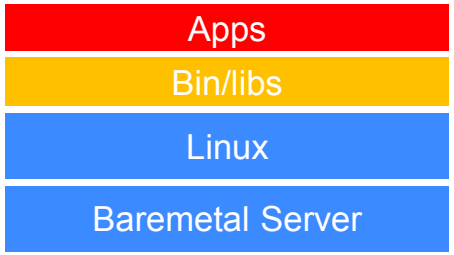
2018年11月28日

日本アイ・ビー・エム株式会社
クラウド事業本部　高良 真穂

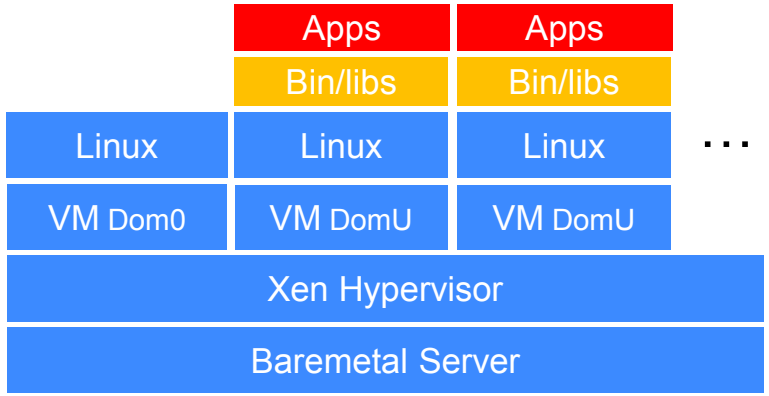
みなさんは Dockerコンテナを 使ってますか？

Dockerコンテナとは

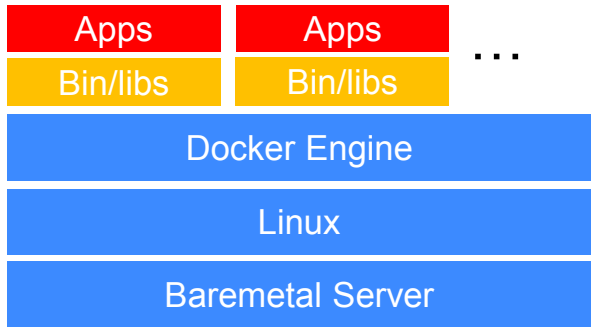
- コンテナの起動は早い、軽量
- 必要なコードセットをOSレベルで独立にまとめ、可搬性がある
- 一つのコンテナホスト上に複数を同居できるので、効率が良い



物理サーバー



仮想サーバー



コンテナ

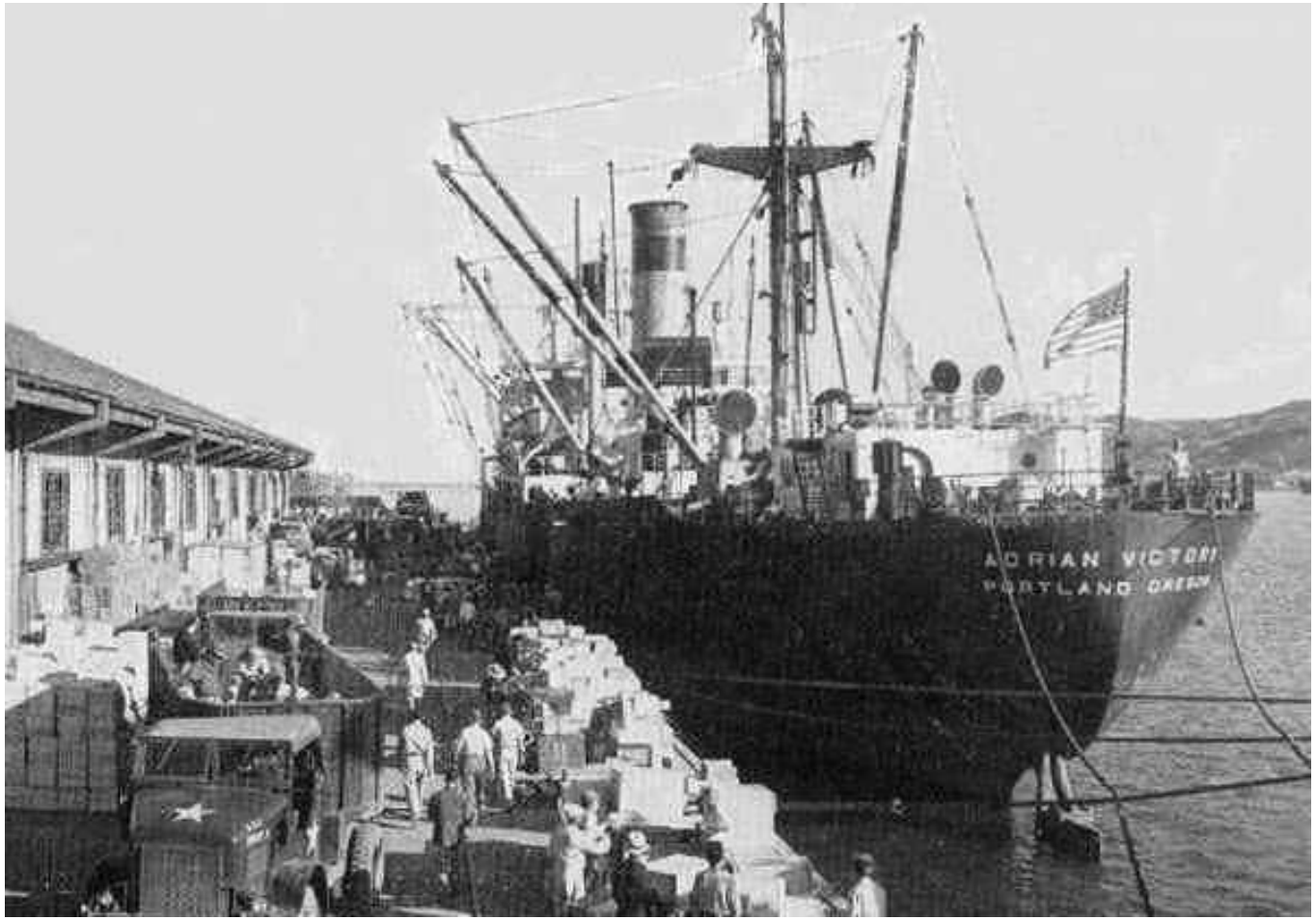
賢者は歴史に学び
愚者は経験に学ぶ

コンテナ以前の様子

大勢の作業員

倉庫に一時保管

物流の最大の
ボトルネック



http://military.wikia.com/wiki/Battle_of_Pusan_Perimeter_logistics

コンテナ革命以前の港湾荷役

物流コンテナによる革命

コンテナの発明者は、全米有数のトラック運送会社のオーナーとなったマルコム・マクリーン（Malcom P. McLean）

国際貨物輸送の分野に
海陸一貫輸送という大変革

1970年代には世界の主要航路のコンテナ化がほぼ完了した。わずか10年程度でこれほど急激な形態の変化が起こったのは海運史上でも他に例がない。

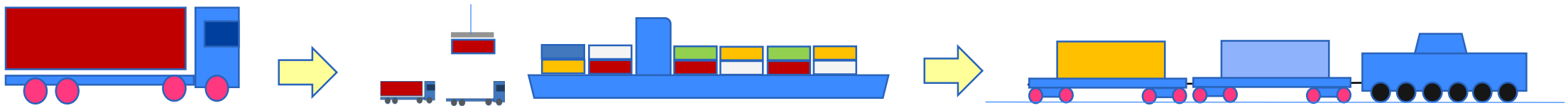
https://www.jsanet.or.jp/seminar/text/seminar_177.html

https://en.wikipedia.org/wiki/Malcom_McLean

コンテナ革命は業務改革

港湾荷役のボトルネックを解消
物流コストの削減とスピードアップ

陸海一貫輸送



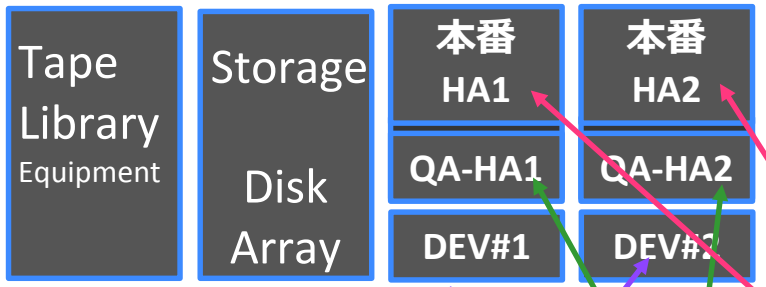
陸送（トラック）　港湾荷役　海上輸送 コンテナ船　港湾荷役　陸送（鉄道）

いま、IT業界のコンテナ革命が始まる

コンテナ以前のシステム

大量ドキュメントと俗人的な努力によるシステム開発&運用

コンテナ以前の
港湾荷役の様な人手作業



オンプレ or クラウド

システム管理者
スーパーバイザ

環境維持
パッチ適用
不具合修正
セキュリティ
チューニング
負荷増対応、資源配分

環境利用
スケジュール

環境取り合い

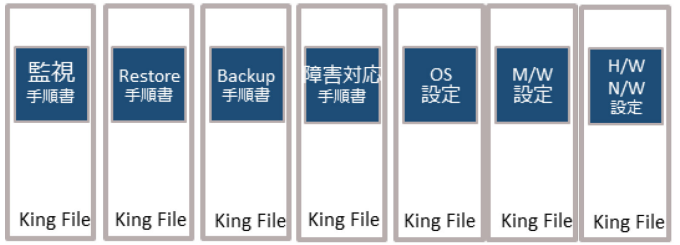
開発者

監視
アプリ・ジョブ運用
基盤運用
監視ソフトの設定
作りこみ

システオペ

ITIL準拠の手順書を作成

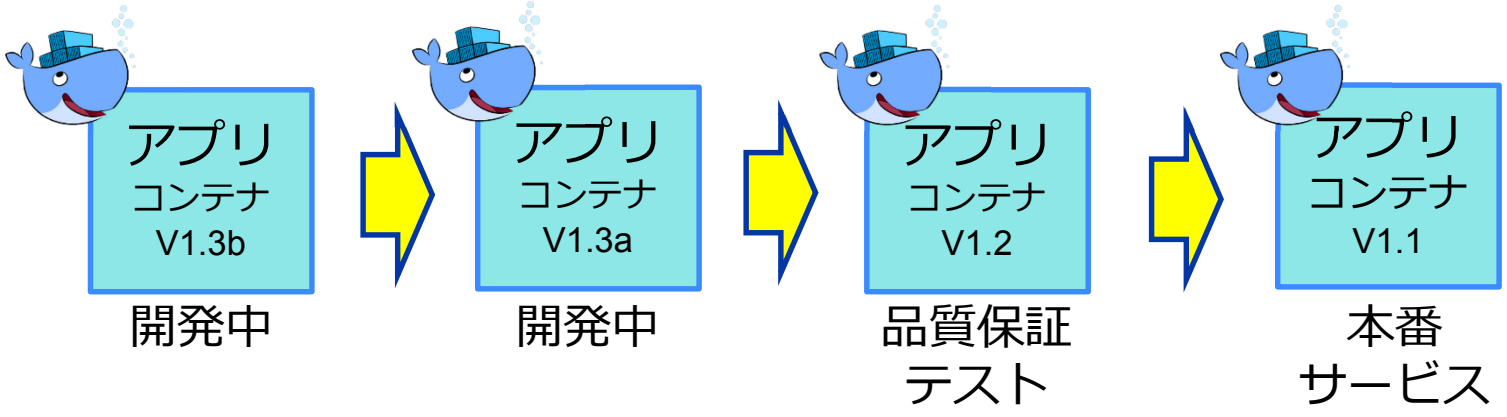
ドキュメント



think Japan

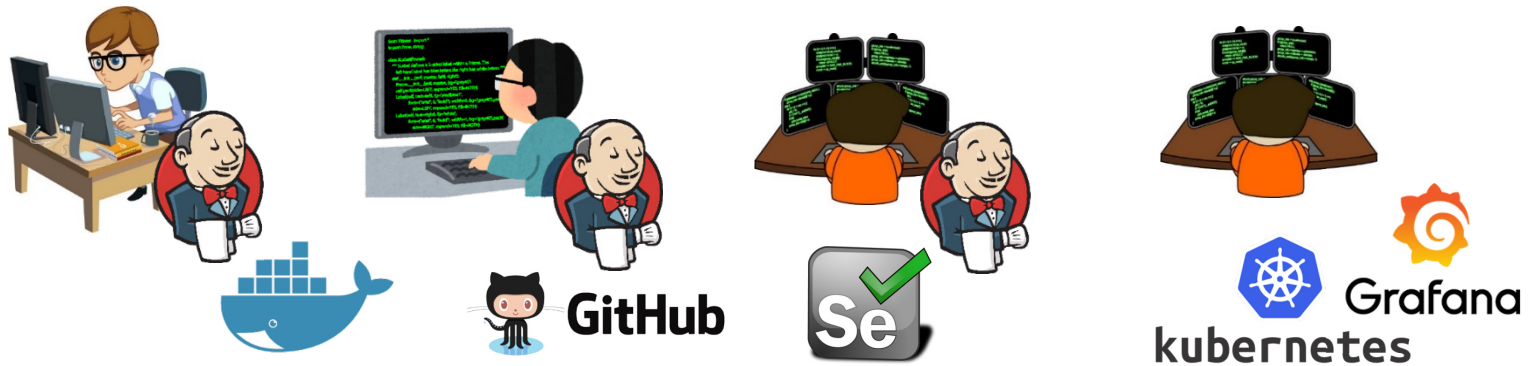
ITのコンテナも業務効率を劇的に改善

デリバリ・パイプライン



https://ja.wikipedia.org/wiki/%E3%82%B3%E3%83%B3%E3%83%86%E3%83%8A#/media/File:Dole_container_is_placed_onto_a_truck.jpg

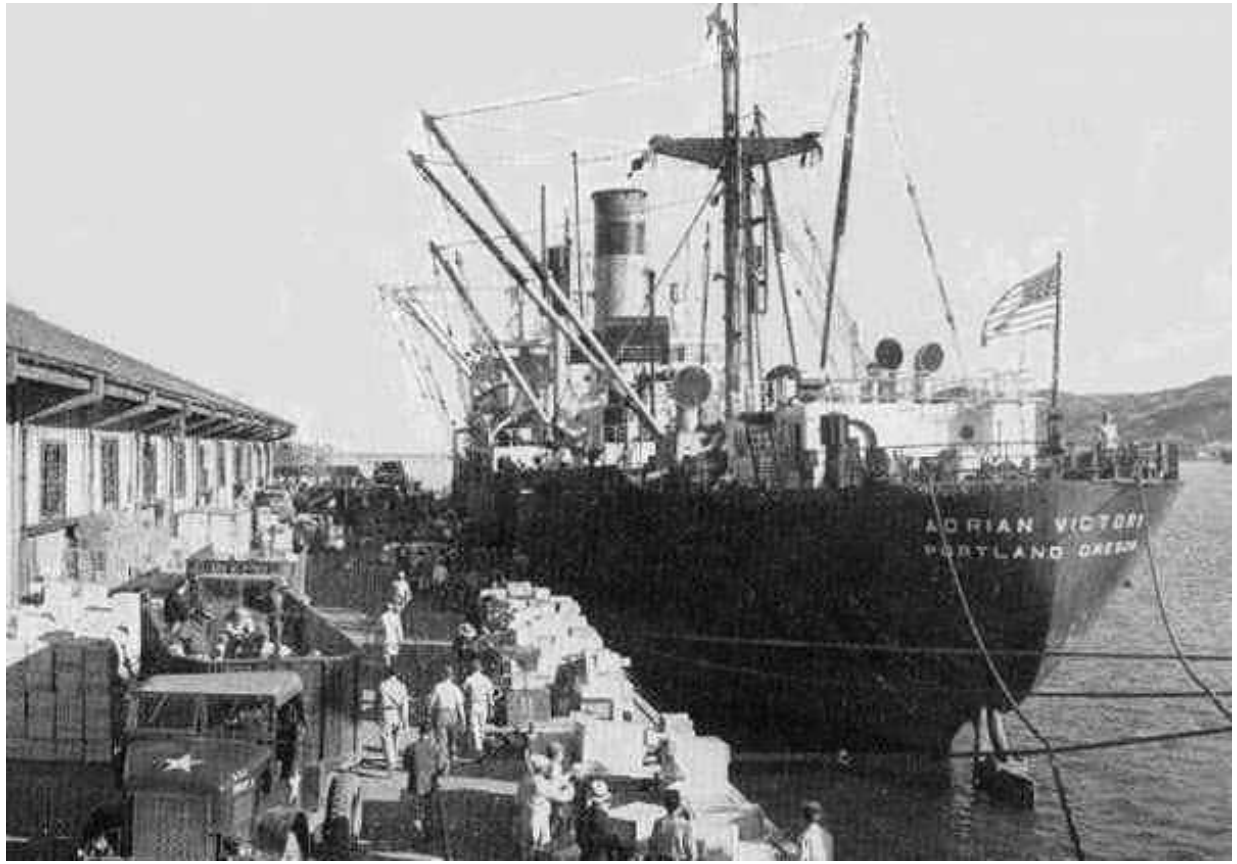
デベロッパー　デベロッパー　テストエンジニア　SRE



高度に最適化された
現代のコンテナ
配送システム
に例えることができる

IT業界で働く
あなたは

どちらを目指しますか？



http://military.wikia.com/wiki/Battle_of_Pusan_Perimeter_logistics

https://ja.wikipedia.org/wiki/%E3%82%B3%E3%83%B3%E3%83%86%E3%83%8A#/media/File:Dole_container_is_placed_onto_a_truck.jpg

think Japan

なんで
コンテナ?

開発で、こんな経験ありませんか？

✓テストはOKでも、本番では不具合が出る

✓デプロイしてみたら、動きませんでした

✓サーバーごとに違うバージョン環境

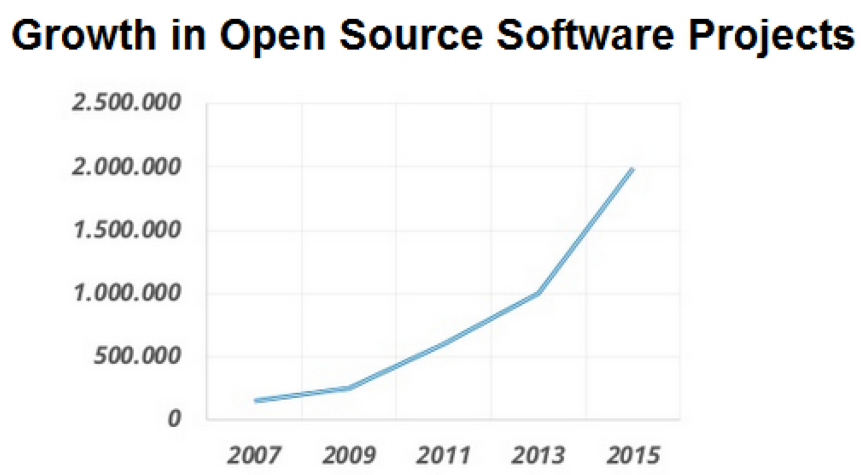
✓テスト環境のデータを占有させて…

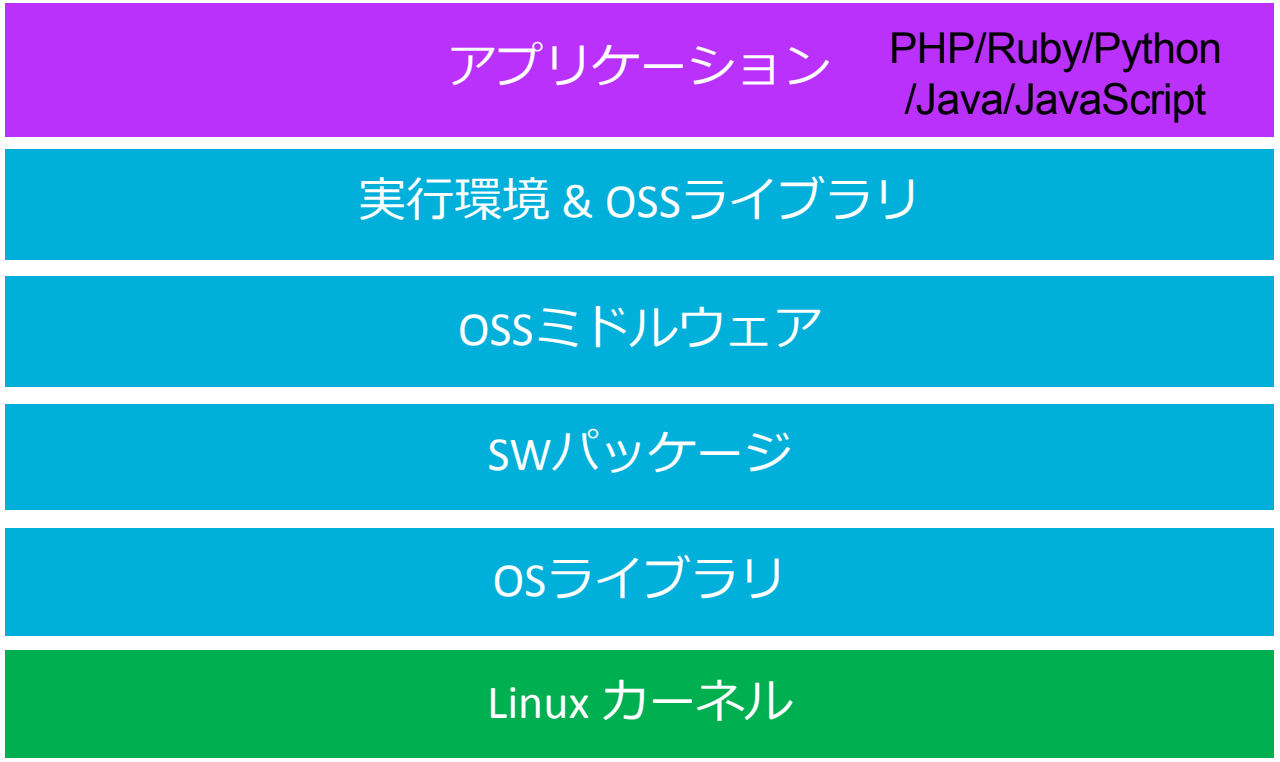
要するに
アプリの土台が
安定しないって事

土台＝プラットフォーム

アプリの土台とは？

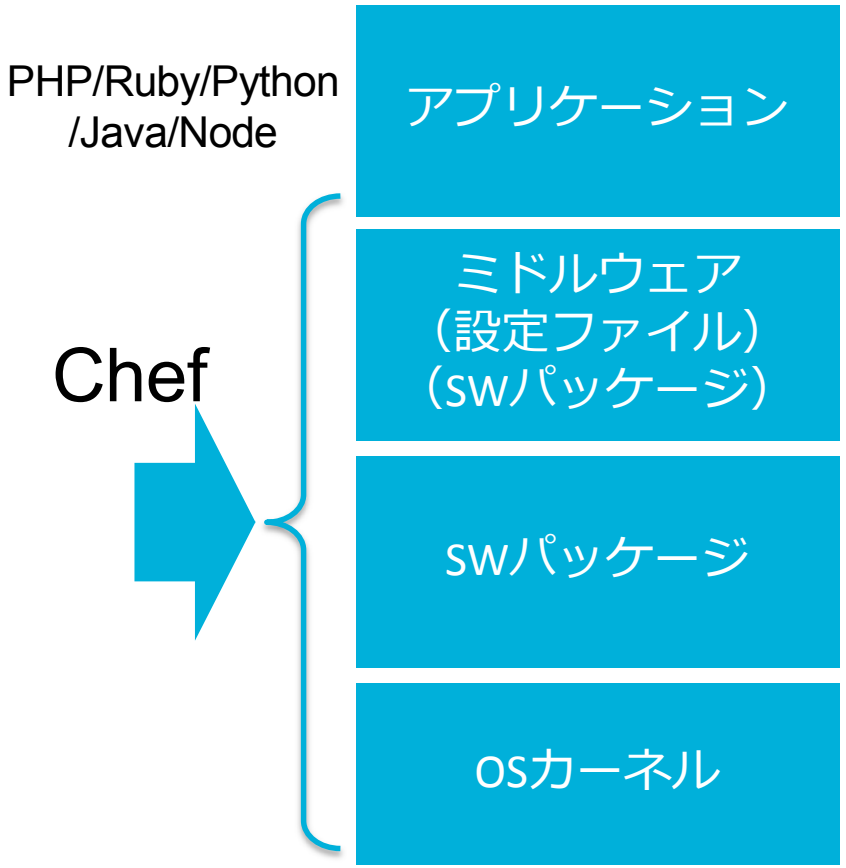
– アプリ開発で、OSSライブラリを抜きに開発が成り立たない
– OSSプロジェクトは、日々進化し、頻繁に更新される





アプリの土台とは？

–リストすると、こんな感じ
–もはや人の力では、把握できない！



/bin/sh
config(nginx) = 1:1.10.2-1.el7
libc.so.6()(64bit)
libc.so.6(GLIBC_2.10)(64bit)
libc.so.6(GLIBC_2.14)(64bit)
libc.so.6(GLIBC_2.2.5)(64bit)
libc.so.6(GLIBC_2.3)(64bit)
libc.so.6(GLIBC_2.3.2)(64bit)
libc.so.6(GLIBC_2.3.4)(64bit)
libc.so.6(GLIBC_2.4)(64bit)
libc.so.6(GLIBC_2.7)(64bit)
libcrypt.so.1()(64bit)
libcrypt.so.1(GLIBC_2.2.5)(64bit)
libcrypto.so.10()(64bit)
libcrypto.so.10(OPENSSL_1.0.1)(64bit)
libcrypto.so.10(OPENSSL_1.0.1_EC)(64bit)
libcrypto.so.10(libcrypto.so.10)(64bit)
libdl.so.2()(64bit)
libdl.so.2(GLIBC_2.2.5)(64bit)
libpcre.so.1()(64bit)
libprofiler.so.0()(64bit)
libpthread.so.0()(64bit)
libpthread.so.0(GLIBC_2.2.5)(64bit)
libssl.so.10()(64bit)
libssl.so.10(libssl.so.10)(64bit)
libz.so.1()(64bit)
nginx-all-modules = 1:1.10.2-1.el7
nginx-filesystem
nginx-filesystem = 1:1.10.2-1.el7
openssl
pcre
rpmlib(CompressedFileNames) <= 3.0.4-1
rpmlib(FileDigests) <= 4.6.0-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rtld(GNU_HASH)
systemd

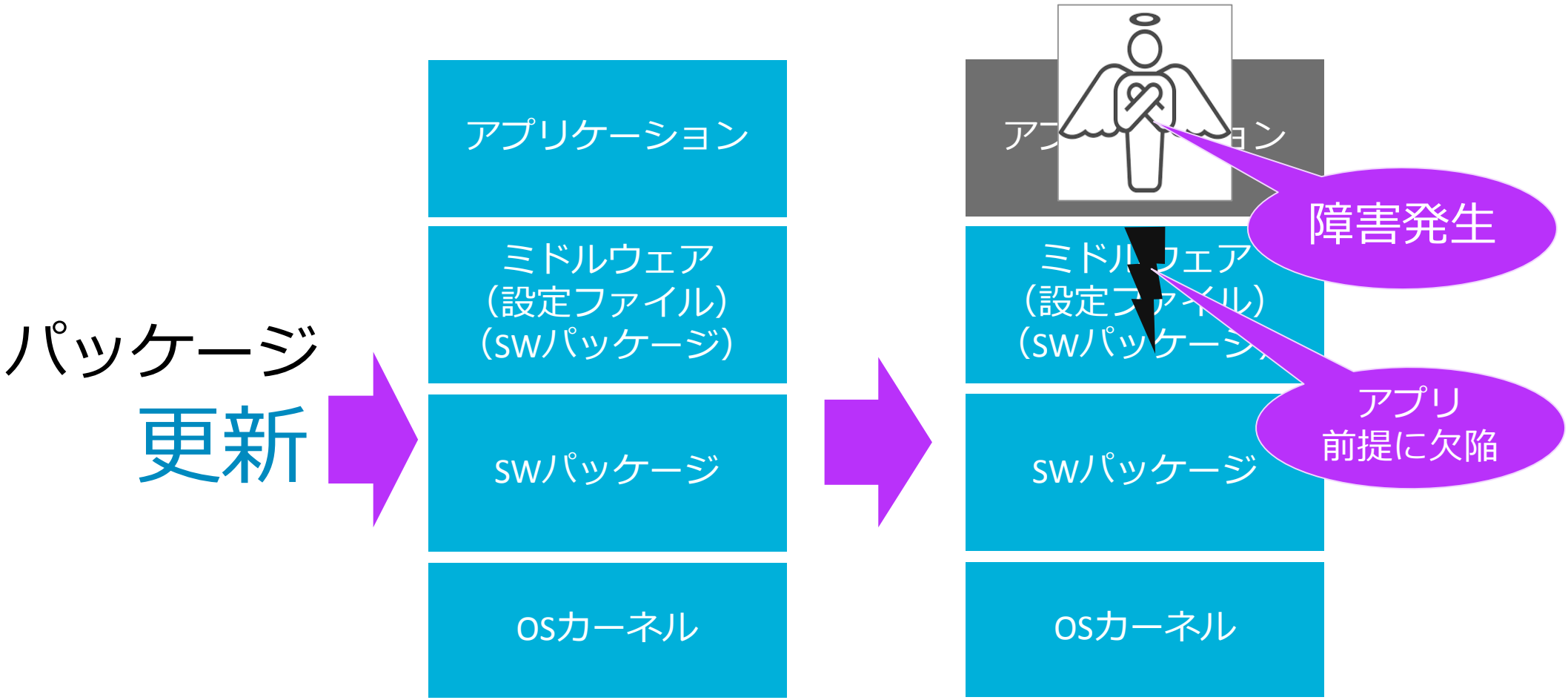
Nginxの依存パッケージ

httpd-mmn = 20120211x8664
php-common(x86-64) = 5.4.16-42.el7
php-cli(x86-64) = 5.4.16-42.el7
httpd
rpmlib(FileDigests) <= 4.6.0-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rpmlib(CompressedFileNames) <= 3.0.4-1
libbz2.so.1()(64bit)
libcom_err.so.2()(64bit)
libcrypto.so.10()(64bit)
libcrypto.so.10(libcrypto.so.10)(64bit)
libcrypto.so.10(OPENSSL_1.0.1)(64bit)
libcrypt.so.1()(64bit)
libc.so.6()(64bit)
libc.so.6(GLIBC_2.11)(64bit)
libc.so.6(GLIBC_2.14)(64bit)
libc.so.6(GLIBC_2.15)(64bit)
libc.so.6(GLIBC_2.2.5)(64bit)
libc.so.6(GLIBC_2.4)(64bit)
libc.so.6(GLIBC_2.3)(64bit)
libc.so.6(GLIBC_2.4)(64bit)
libc.so.6(GLIBC_2.7)(64bit)
libc.so.6(GLIBC_2.8)(64bit)
libdl.so.2()(64bit)
libdl.so.2(GLIBC_2.2.5)(64bit)
libgmp.so.10()(64bit)
libgssapi_krb5.so.2()(64bit)
libk5crypto.so.3()(64bit)
libkrb5.so.3()(64bit)
libm.so.6()(64bit)
libm.so.6(GLIBC_2.2.5)(64bit)
libnsl.so.1()(64bit)
libpcre.so.1()(64bit)
libresolv.so.2()(64bit)
libresolv.so.2(GLIBC_2.2.5)(64bit)
librt.so.1()(64bit)
libssl.so.10()(64bit)
libssl.so.10(libssl.so.10)(64bit)
libxml2.so.2()(64bit)
libxml2.so.2(LIBXML2_2.4.30)(64bit)
libxml2.so.2(LIBXML2_2.5.2)(64bit)
libxml2.so.2(LIBXML2_2.6.0)(64bit)
libxml2.so.2(LIBXML2_2.6.11)(64bit)
libxml2.so.2(LIBXML2_2.6.5)(64bit)
libxml2.so.2(LIBXML2_2.9.0)(64bit)
libz.so.1()(64bit)
rtld(GNU_HASH)
rpmlib(PayloadIsXz) <= 5.2-1

PHPの依存パッケージ

アプリの土台は膨大に

– 膨大な量のスタックの上で、アプリが稼動している事実
– スタックの更新で障害が出るか検証不可能、管理不可能、予知不可能



コンテナ技術はどの様に課題を解決する？

Dockerによる解決策

– 膨大な量のSWスタックを管理するのは無理なので、アプリのテストがパスしたら、その後はサーバーのソフトウェアに変更を加えない。

–Immutable Infrastructure （不変の基盤）

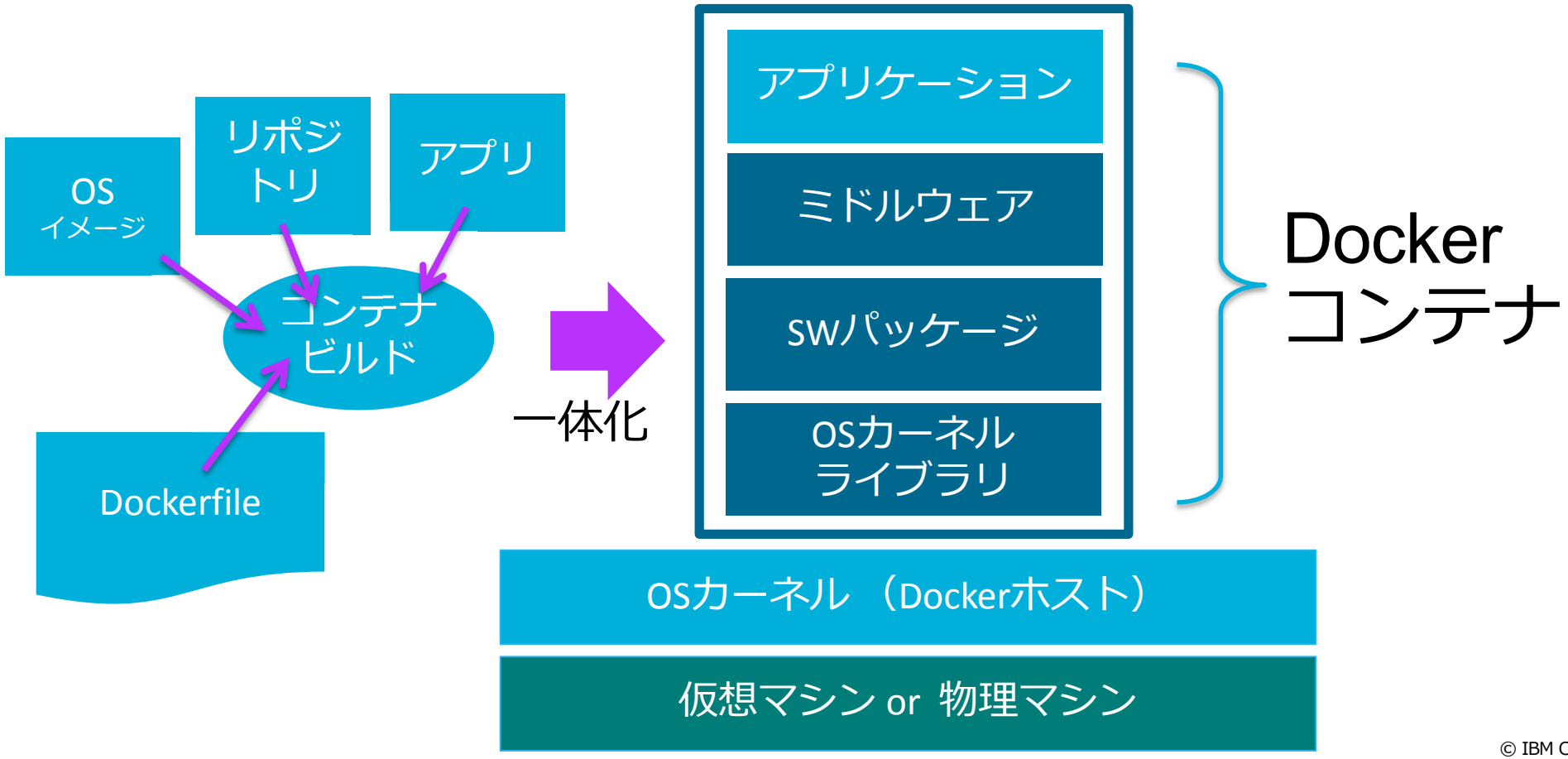
アプリの土台を

凍結

Immutable

Dockerによる解決策

– Dockerfile (コンテナ仕様書）からアプリが含まれたコンテナを作成
– 変更が必要な場合は、コンテナごと再ビルド
– コンテナは、OSカーネルのSW更新の影響を受けない



コンテナのメリット　（もうちいど）

軽い

コンテナの起動は、
秒単位である

ポータブル

必要なコードセットを
OSレベルで独立にまとめる
事ができる

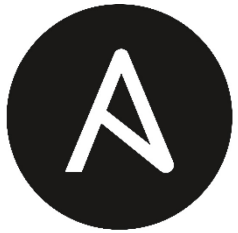
効率が良い

仮想サーバーや
物理サーバー上に
同居できる

大雑把に整理すると

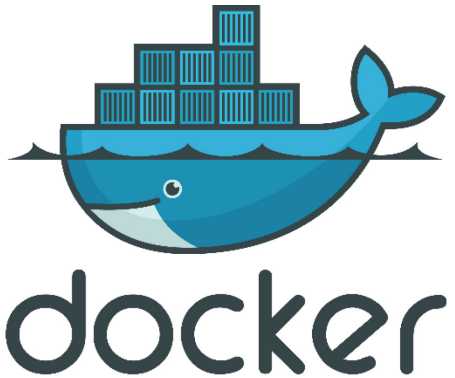
サーバー自動設定ツール CHEF と コンテナ・ランタイム環境 Dockerの整理

サーバー自動設定ツールは、 SWパッケージやソフトウェア設定を何度でも実行できるツール



冪等性
idempotence

Dockerはアプリの動作環境を変えないために、OSのSWスタックから隔絶（コンテナ化）して、稼動させるツール



不変性
Immutable

Dockerを本番サービスに適用できるの？

開発環境と本番環境を分離したら課題が発生

ー今までの本番デプロイは、アプリのモジュールを置き換えるだけ

アプリのデプロイは
紫色の部分だけ

開発環境

- アプリケーション
- ミドルウェア
- SWパッケージ
- OSカーネル
- 開発仮想マシン

本番環境

- アプリケーション
- ミドルウェア
- SWパッケージ
- OSカーネル
- 本番マシン

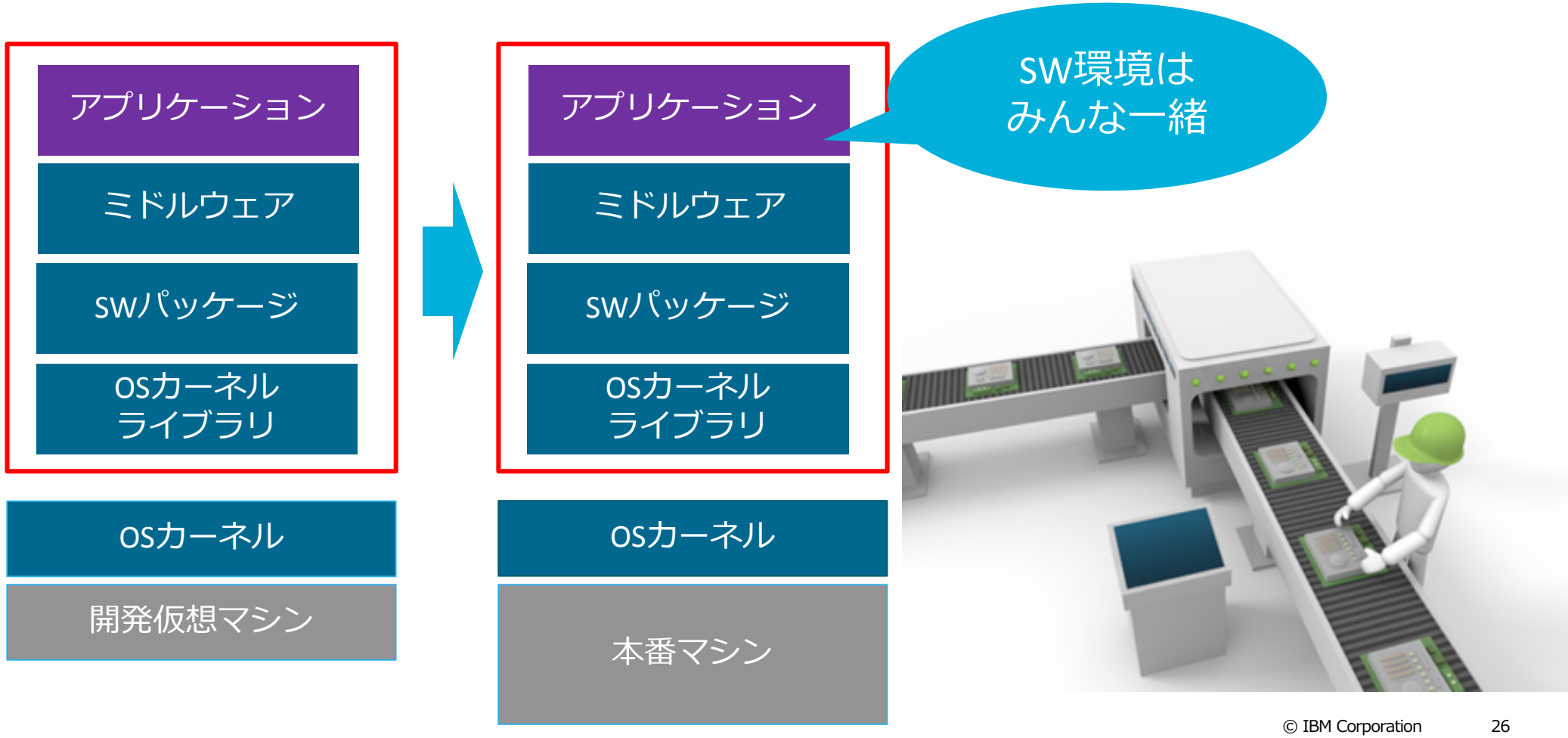
モジュールが
足りないよ！

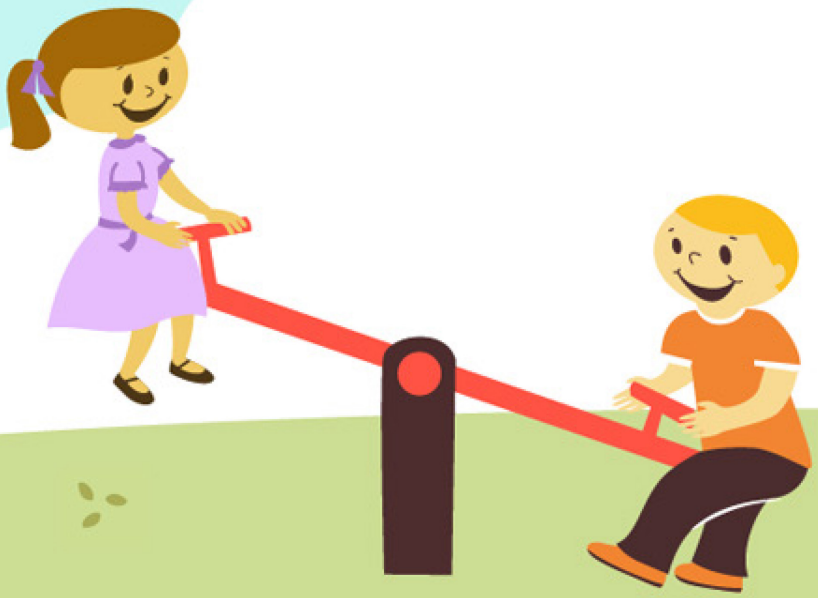
コンテナにより、アプリケーションの環境依存を最小化する

–コンテナではコンベアで移動する様に、開発から本番へ移動するだけ

– アプリの土台として依存するソフトウェアをコンテナとしてパッケージ化

「アプリの土台」から丸ごとコンテナとして本番へ移動させる





Docker ハンズオンへ

ハンズオンのファイルを開いてください。