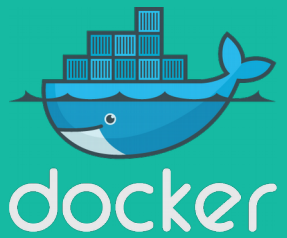
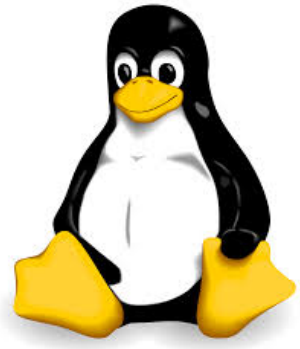
Docker入門



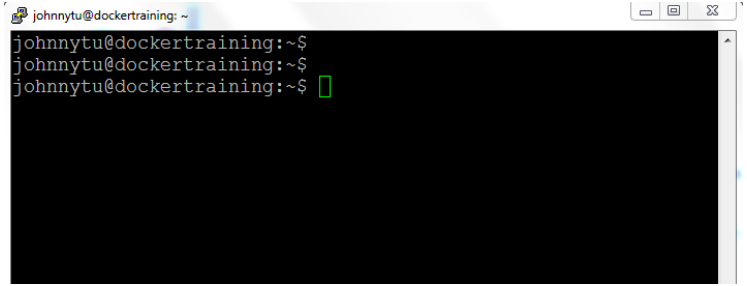
セッションの方針と前提条件

- 2日間に質問時間と演習時間を含みます
- おおよそ1時間ごと10分間の休憩を取ります
- お昼に1時間の休憩を取ります
- いつでも質問を受け付けます



前提条件

- Dockerの経験は不要です
- 演習用のLinux仮想マシンを提供します
- Linuxのコマンドラインの慣れが必要です



トレーニング環境について

- Ubuntu 14.04の仮想マシンを提供します。
- ログイン方法の詳細は別途お知らせします。
- 演習はこの仮想マシン上で実施します。

注意: 演習環境はAmazon Web Servicesを想定しています。
しかし、演習はDigital Ocean等の任意の環境でも可能です。

トレーニング環境にアクセス

- Ubuntu仮想マシンにログインします。
  ログイン方法の詳細は講師に確認してください。
  - MacやLinuxであればターミナル上からSSHを使います。
  - WindowsユーザはPuTTYやTeraTerm等を使います。

アジェンダ

- コンテナ入門
- Docker のインストール
- Docker の概念と用語
- イメージ入門
- コンテナの実行と管理
- イメージの構築
- イメージの管理と配布
- コンテナのボリューム
- コンテナのネットワーク機能
- Docker で継続的インテグレーション

1章:
コンテナ入門

docker

章の目的

本章で扱う内容:

- Dockerプラットフォームの概説
- コンテナを基盤とした仮想化概念の入門
- 仮想マシンを上回るコンテナの利点
- 見込み所要時間: 60分

Dockerとは?

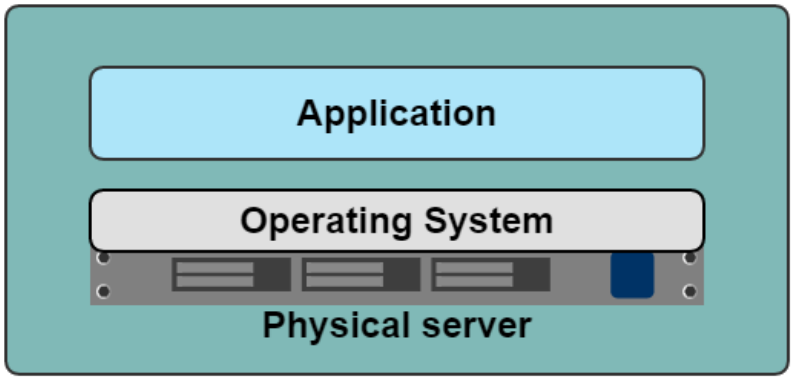
Docker とは、コンテナ技術を使ってアプリケーションを開発・出荷・実行するプラットフォームです。

- Dockerプラットフォームは、複数の製品やツールから構成されています。
  - Docker Engine
  - Docker Hub
  - Docker Trusted Registry
  - Docker Machine
  - Docker Swarm
  - Docker Compose
  - Kitematic

歴史の勉強

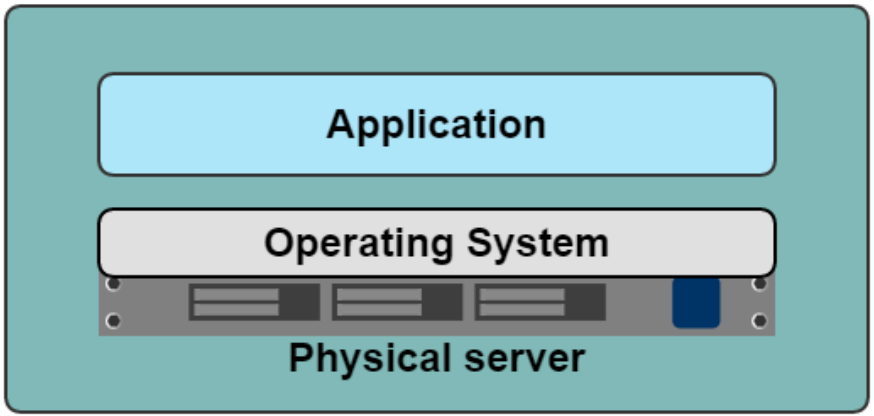
暗黒時代

1台の物理サーバに1つのアプリ



アプリケーションのデプロイの歴史的な限界

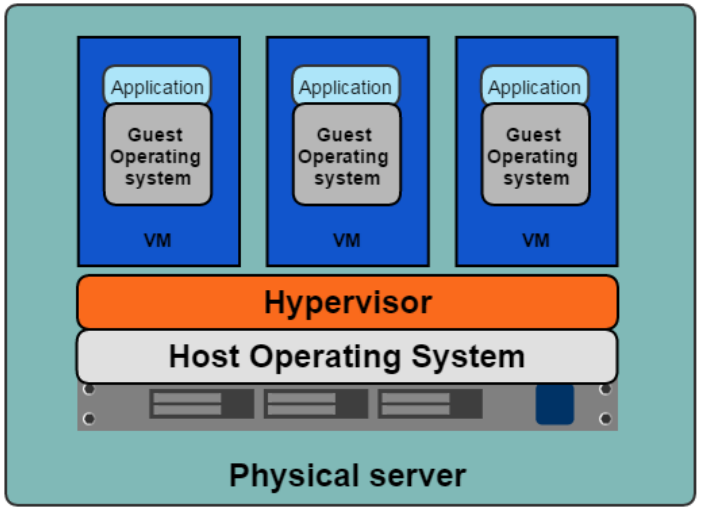
- 遅いデプロイ時間
- 大きなコスト
- リソースの無駄遣い
- スケールが困難
- マイグレーションが困難
- ベンダーロックイン



歴史の勉強

ハイパーバイザ型の仮想化基盤

- 1台の物理サーバで複数のアプリケーションを動かせるようになった。
- 各アプリケーションは仮想マシン(VM)上で動作。



仮想マシンの利点

- より良いリソースの共有
  - 1つの物理マシンを複数の仮想マシンに分割
- スケールが容易(システムの増加・減少)
- クラウド上の仮想マシン
  - 素早く、融通が効く
  - 使ったぶんだけ支払うモデル

仮想マシンの限界

- 各仮想マシンに必要なもの
  - CPU割り当て
  - ストレージ
  - メモリ
  - ゲストOSが動作するために必要なすべて

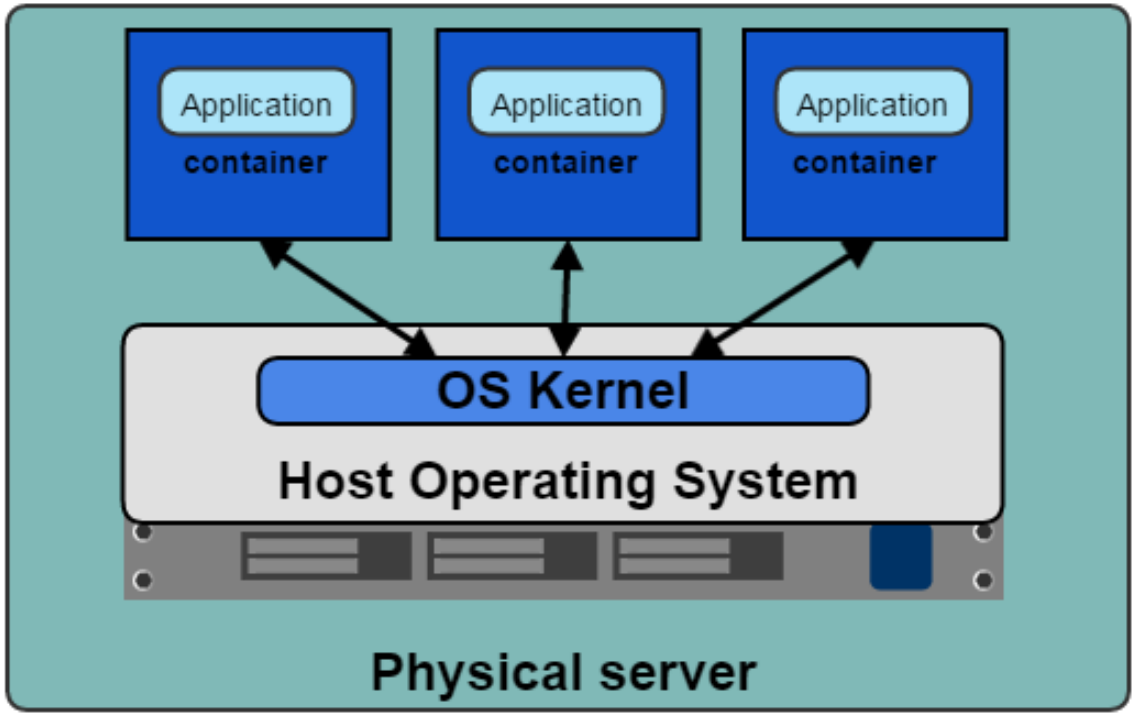
- 多くの仮想マシンを実行するにはより多くのリソースが必要
- ゲストOSはリソースを浪費
- アプリケーションのポータビリティ(可搬性)を保証しない

コンテナ入門

「コンテナ化」では、複数のルートファイルシステムを実行するためにホストOSのカーネルを使います。

- 各ファイルシステムを「コンテナ (container)」と呼ぶ
- 各コンテナはそれぞれがリソースを持つ
  - プロセス
  - メモリ
  - デバイス
  - ネットワークスタック

コンテナ


Application
container
Application
container
Application
container
OS Kernel
Host Operating System
Physical server

コンテナ vs 仮想マシン

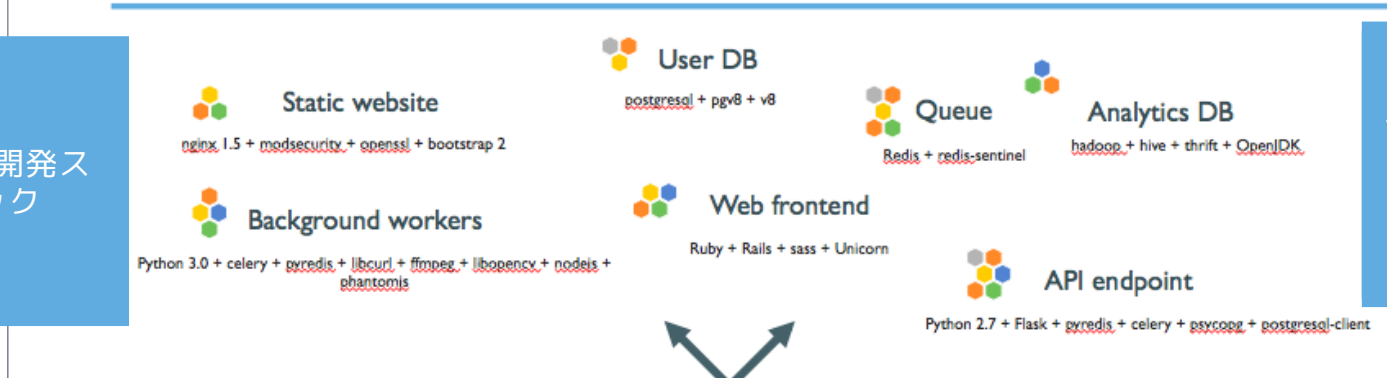
- コンテナは仮想マシンより軽量
- ゲストOSのインストールが不要
- 必要とするCPU 、メモリ、ストレージが少ない
- 物理マシン上に、仮想マシンより多くコンテナを起動できる
- 高いポータビリティ(可搬性)
- コンテナでは共通のOSを共有するように容易に管理できる
  - 単一OS上で複数の負荷を共有できる
- コンテナは仮想マシンと比較してマイクロサービスの開発・デプロイのより優れた方法

なぜDockerを使うのか?

- アプリケーションはもはや巨大な一枚岩(モノリシック)構造ではない
- サービス指向アーキテクチャ(SOA)とは、複数のアプリケーションをデプロイする必要があることを意味する
- サービスの分割により、反復性とスケールアウトをもたらす
- デプロイが複雑な手順になりがち

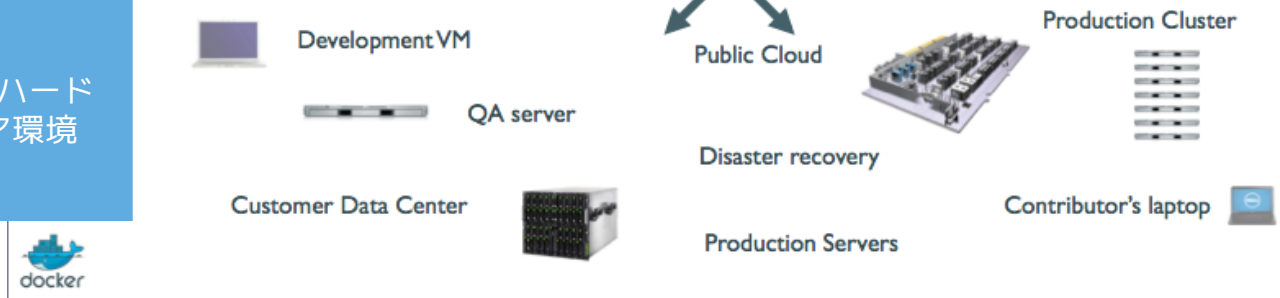
デプロイの悪夢

複数の開発スタック



サービスとアプリを適切に相互通信するには?

複数のハードウェア環境



素早くスムーズにマイグレーションできる?

デプロイの悪夢 (続き)

	Developmen t VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor' s laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

出荷の例

複数の種類の商品

商品の相互関係は大丈夫? (例: コーヒー豆は香辛料の隣に置く)

複数の配達方法

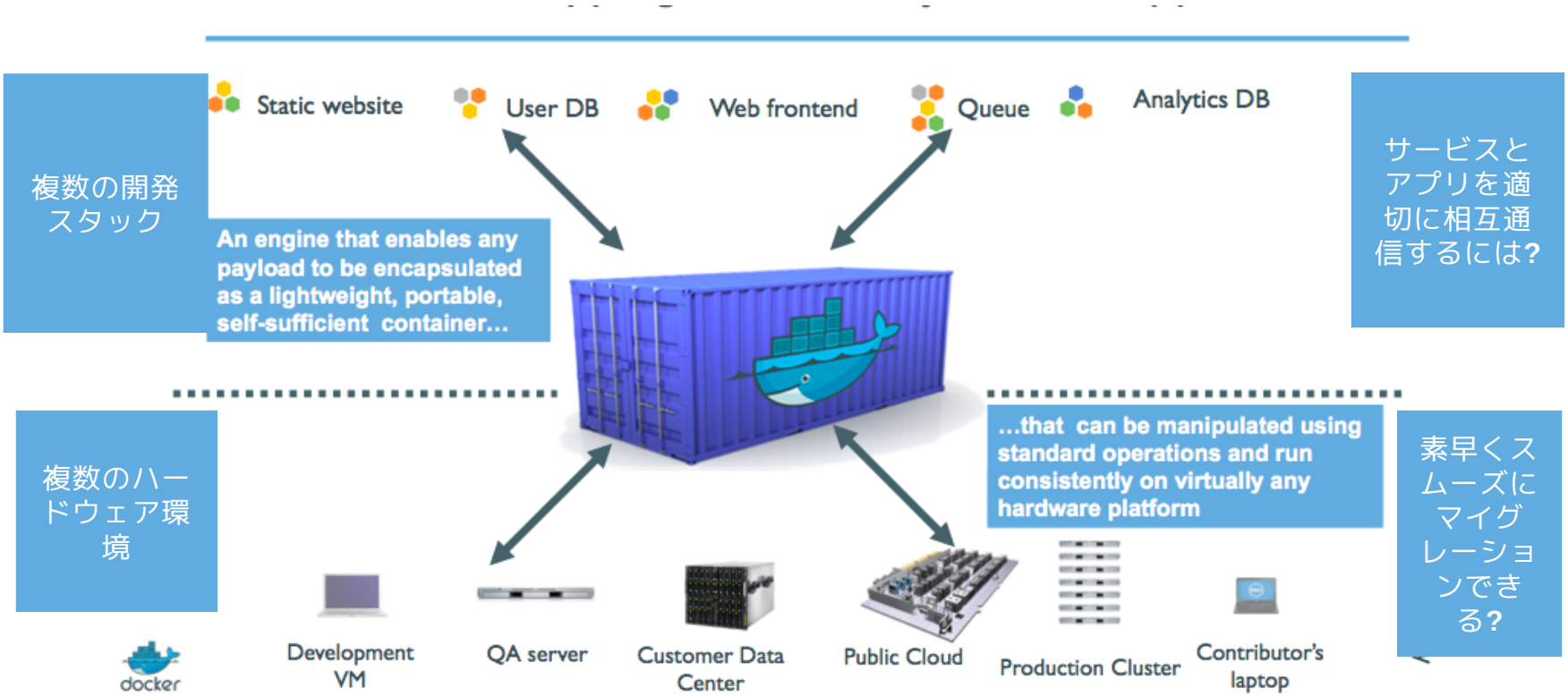
素早くスムーズに配達できる? (例:船から列車への荷揚げ)

PRESENTED BY CREATIONLINE,INC. 2017

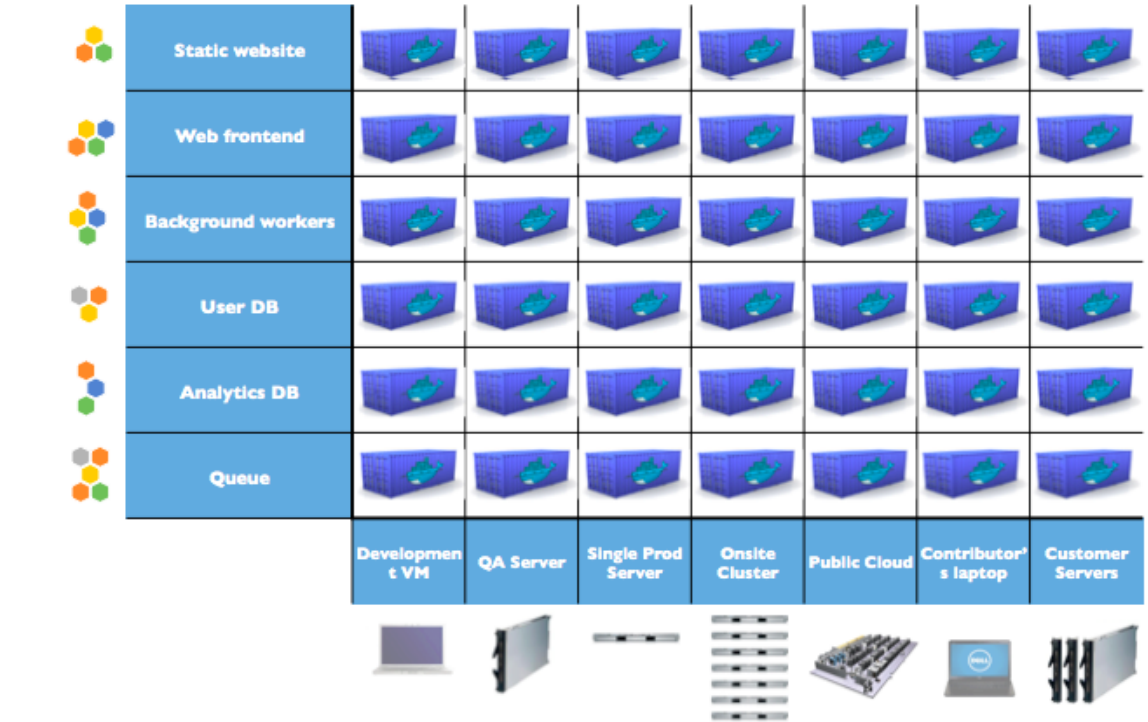
コンテナの出荷


複数の種類の商品
A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.
商品の相互関係は大丈夫? (例: コーヒー豆は香辛料の隣に置く)
…in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another
複数の配達方法
素早くスムーズに配達できる? (例:船から列車への荷揚げ)
docker

Docker コンテナ


複数の開発スタック
Static website
User DB
Web frontend
Queue
Analytics DB
サービスとアプリを適切に相互通信するには?
An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…
…that can be manipulated using standard operations and run consistently on virtually any hardware platform
複数のハードウェア環境
素早くスムーズにマイグレーションできる?
docker
Development VM
QA server
Customer Data Center
Public Cloud
Production Cluster
Contributor's laptop

デプロイの組み合わせ問題を解決



Docker社について

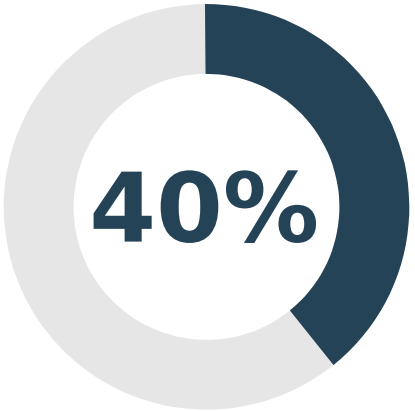
商用**Docker**ソリューションの提供

- 構築・出荷・実行を統合するソリューション
- 公式プロバイダによる商用技術サポート (Docker, IBM, HPE)

Dockerプロジェクトのスポンサー

- オープンソース版Dockerに対する主要な貢献および保守
- 10億以上のイメージをダウンロード
- 1500人以上の貢献者
- 200,000以上のDocker化アプリケーション

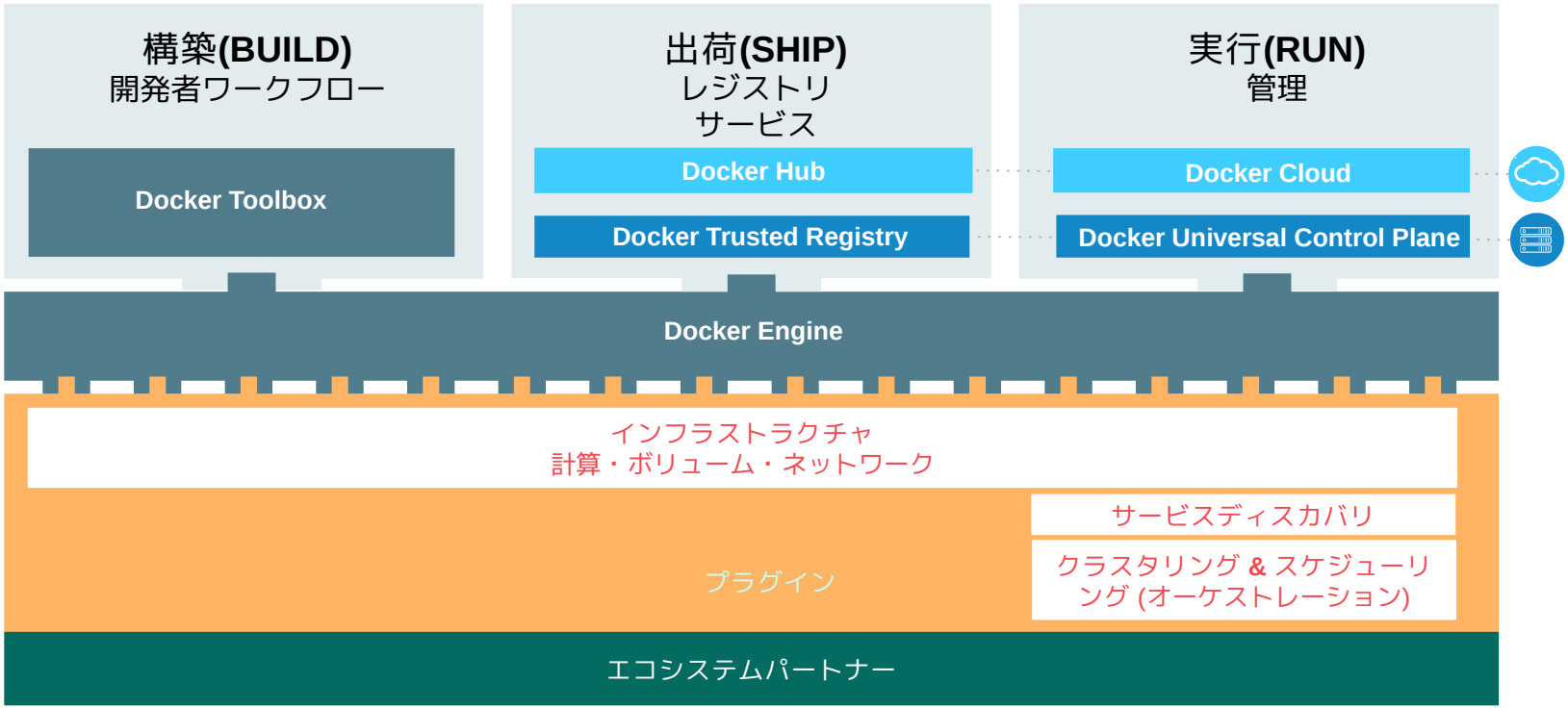
既に本番環境でDockerを利用しているユーザ



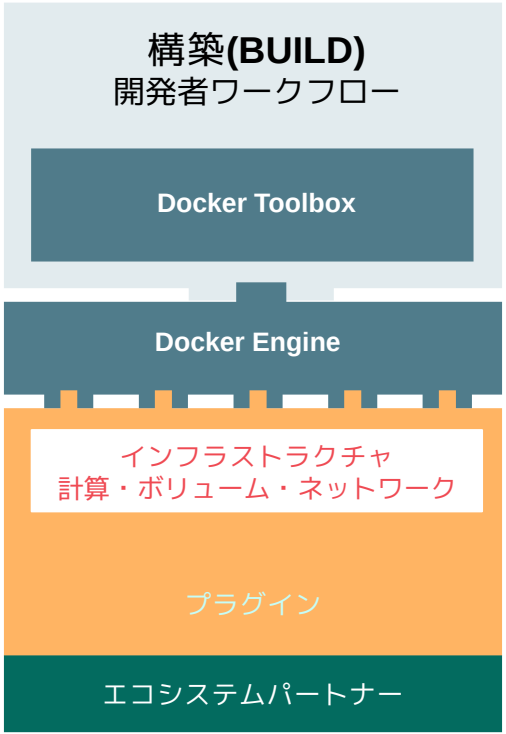
Gerber, Anna. “The State of Containers and the Docker Ecosystem: 2015” O’Reilly, September 2015

Docker社のミッション

Dockerのプラットフォーム



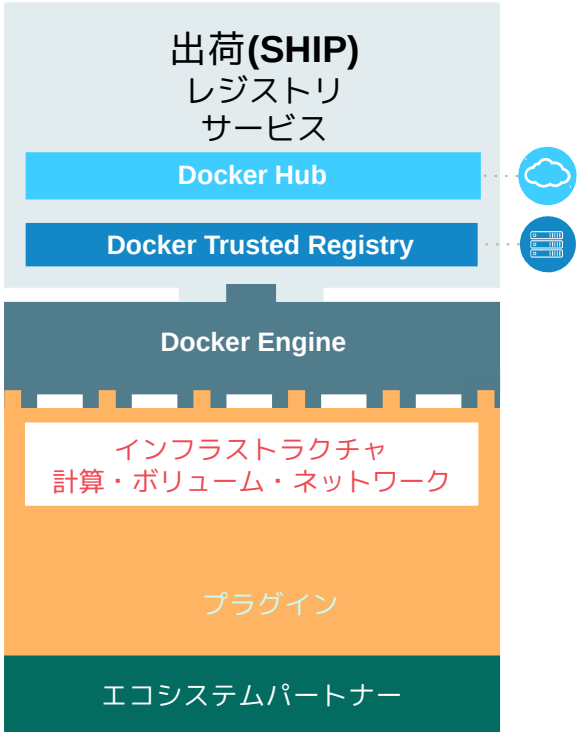
Docker化アプリを開発するための開発者のワークフロー



Docker Toolboxインストーラ を使えば、数クリックでMacやWindows環境に開発環境が整います:

- **Docker Engine** はローカルで仮想マシンを実行
- **Docker Compose** は複数のコンテナアプリを定義
- **Docker Swarm** はDocker Engineをクラスタリングし、コンテナをスケジューリング
- **Docker** クライアント でコマンドライン操作
- **Kitematic** でグラフィカル操作(GUI)

開発者とIT運用者のセキュアな共有と協調



- イメージをクラウドやオンプレミスに格納
- 自ネットワーク内もしくはクラウド上のイメージレジストリに**Docker**イメージを格納
- ユーザや組織によるアクセス権の管理
- **Content Trust**でイメージに署名
- **Web UI**でレポジトリ内の検索やユーザと設定の管理
- 自動化ワークフローをCIやCDシステムと統合

Trusted Registry
- LDAP/AD 統合
- 柔軟なストレージサポート
- ユーザ監査ログ
- イメージのソフトデリート
- ガベージコレクション

Hub
- ウェブフックとトリガー
- 自動構築

Docker化アプリをさまざまな場所へデプロイ・管理

- クラウドとオンプレミスに対応した管理コンソール
- ハイブリッドな基盤の管理
- **GUI**の管理ダッシュボードによる可視化
- コンテナと**Compose**アプリのデプロイと管理
- **Docker Engine**クラスタ**(Swarm)**のデプロイと管理
- イメージ、ネットワーク、ボリュームの統合管理
- **Hub**や**Trusted Registry**を通したレジストリの統合
- モニタリング、ログ、サービスディスカバリ等のシステムメトリック

Universal Control Plane
- LDAP/AD 統合/RBAC
- 高可用性
- TLS サポート

Docker Cloud
- IaaSクラウドとのセキュアな連携
- ホストとクラスタのプロビジョニング

Dockerの利点

- 懸念事項の分離
  - 開発者はアプリケーション構築に集中
  - システム管理者はデプロイ作業に集中
- 高速な開発サイクル
- アプリケーションのポータビリティ
  - ある環境で構築し、別の環境に出荷
- スケーラビリティ
  - 必要なときに新しいコンテナを簡単に用意可能
- 1つのマシン上で多くのアプリを実行

2章:
Dockerの概念と用語

docker

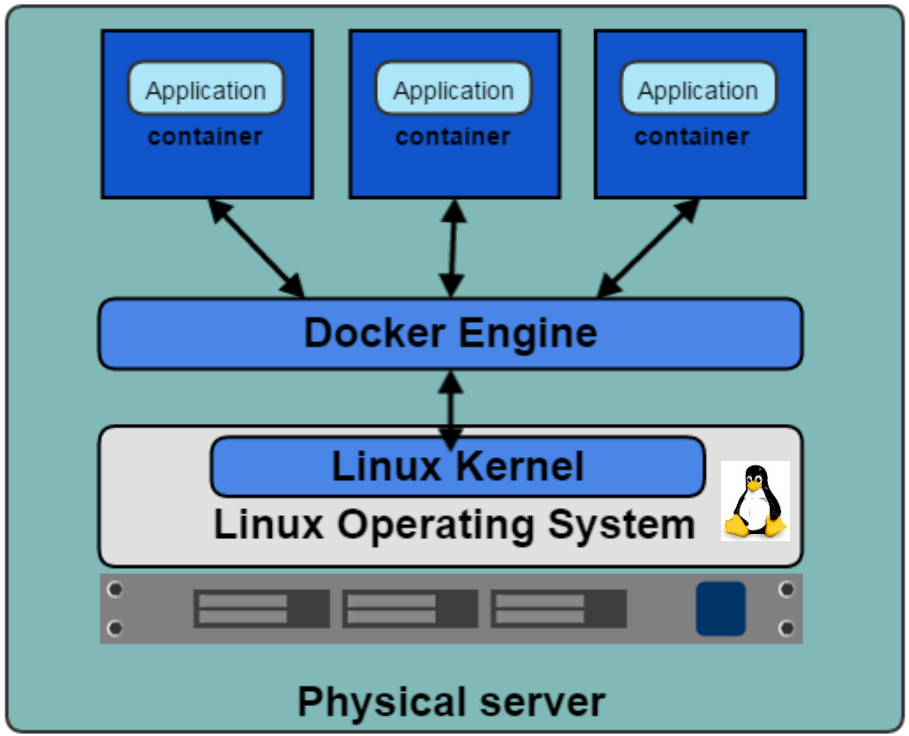
章の目的

本章で扱う内容:

- Dockerを構成するすべての要素を解説
  - Docker Engine
  - Dockerクライアント
  - コンテナ
  - イメージ
  - レジストリとレポジトリ
  - Docker Hub
  - オーケストレーション
  - Docker Toolbox / Machine
- 見込み所要時間: 30分

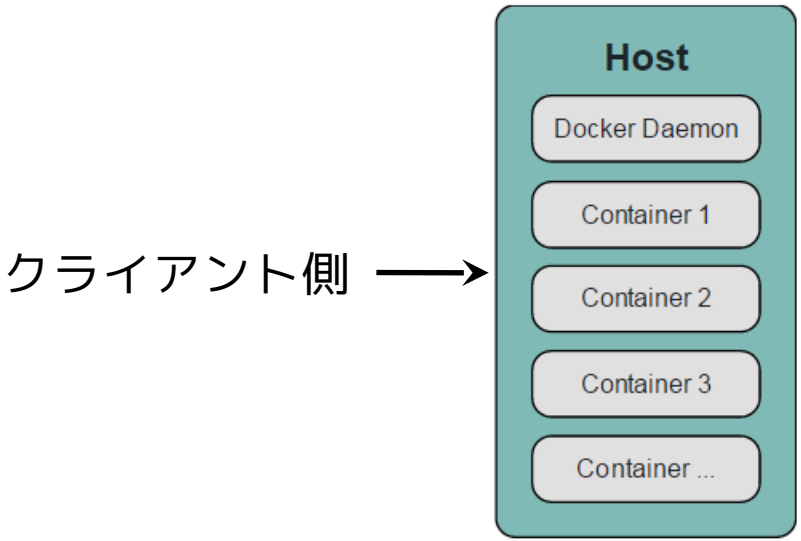
DockerとLinuxカーネル

- **Docker Engine**はコンテナを配布・実行するためのプログラム
- Docker EngineはLinuxカーネルのネームスペース(namespace)とコントロールグループ(cgroup)を使う
- ネームスペースによりワークスペースを分離する



Dockerクライアントとデーモン

- クライアント・サーバ型アーキテクチャ
- クライアントはユーザからの入力を受け付け、デーモンに対し送信
- デーモンはコンテナの配布と実行を行う
- クライアントとデーモンは同じホスト上でも異なったホスト上でも実行できる
- クライアントにはCLIとGUI(Kitematic)がある



クライアントとデーモンのバージョン確認

- コマンド
  `docker version`

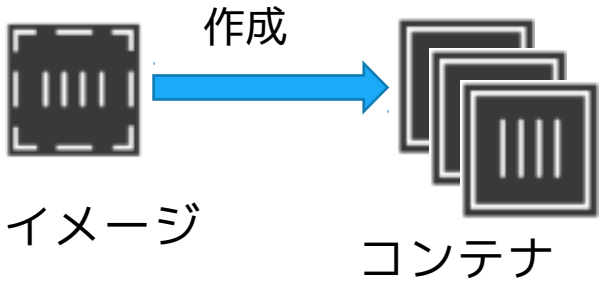
```
ubuntu@node-0:~$ docker version
Client:
 Version:      1.11.1
 API version:  1.23
 Go version:   go1.5.4
 Git commit:   5604cbe
 Built:        Tue Apr 26 23:30:23 2016
 OS/Arch:      linux/amd64

Server:
 Version:      1.11.1
 API version:  1.23
 Go version:   go1.5.4
 Git commit:   5604cbe
 Built:        Tue Apr 26 23:30:23 2016
 OS/Arch:      linux/amd64
```

クライアント側

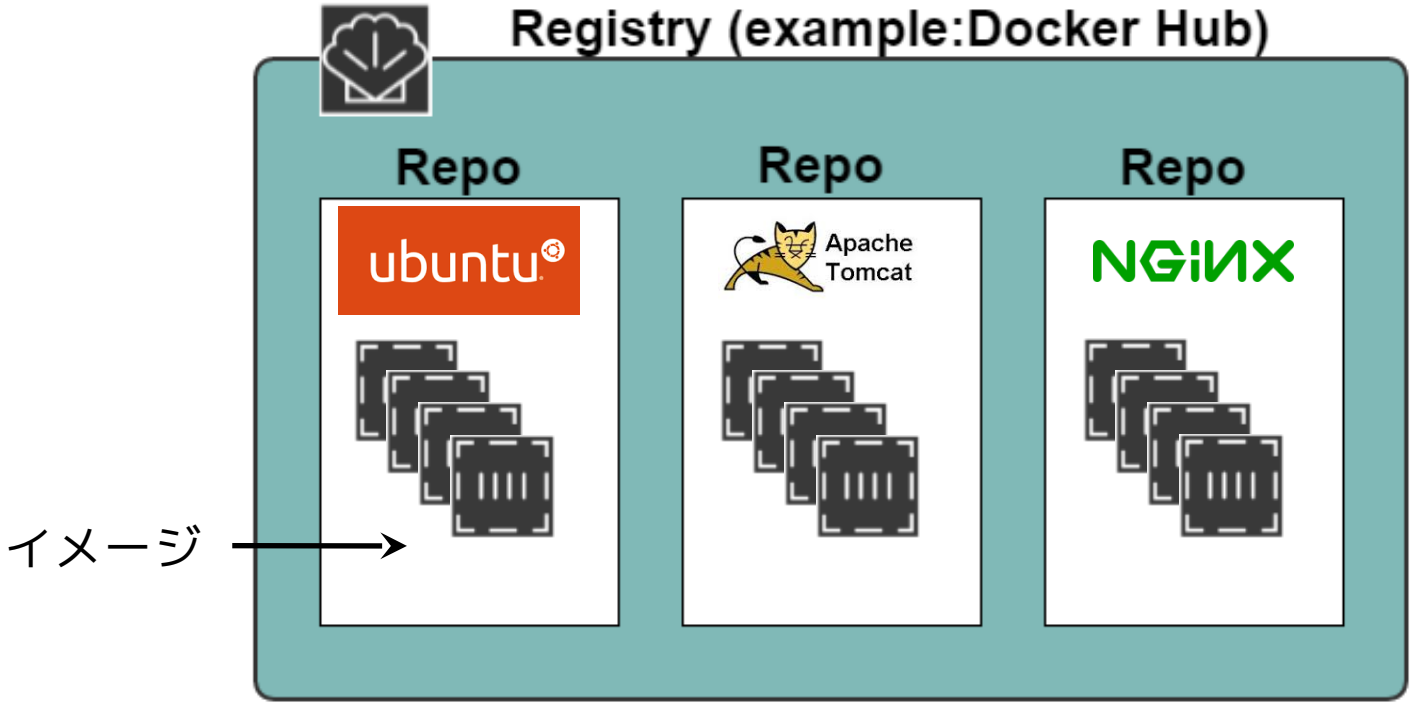
デーモン側

Dockerコンテナとイメージ

- イメージ
  - コンテナを作成するための読み込み専用テンプレート
  - 自分または他のDockerユーザが構築
  - Docker Hub、Docker Trusted Registry、あるいは自身のレジストリに格納
- コンテナ
  - 分離した(独立した)アプリケーションのプラットフォーム
  - アプリケーションの実行に必要なすべてを含む

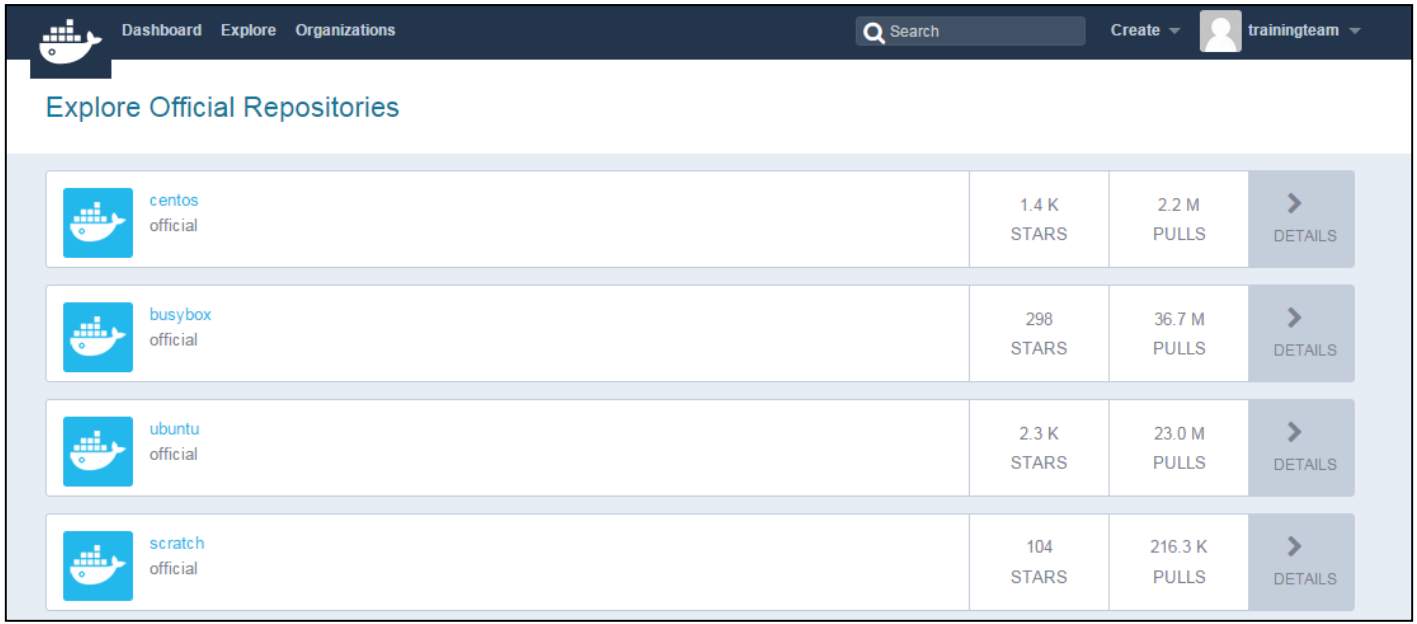


レジストリとレポジトリ



Docker Hub

- **Docker Hub** は、パブリックあるいはプライベートなレジストリ機能を提供する、Docker社が提供するクラウドホステッドサービスです。

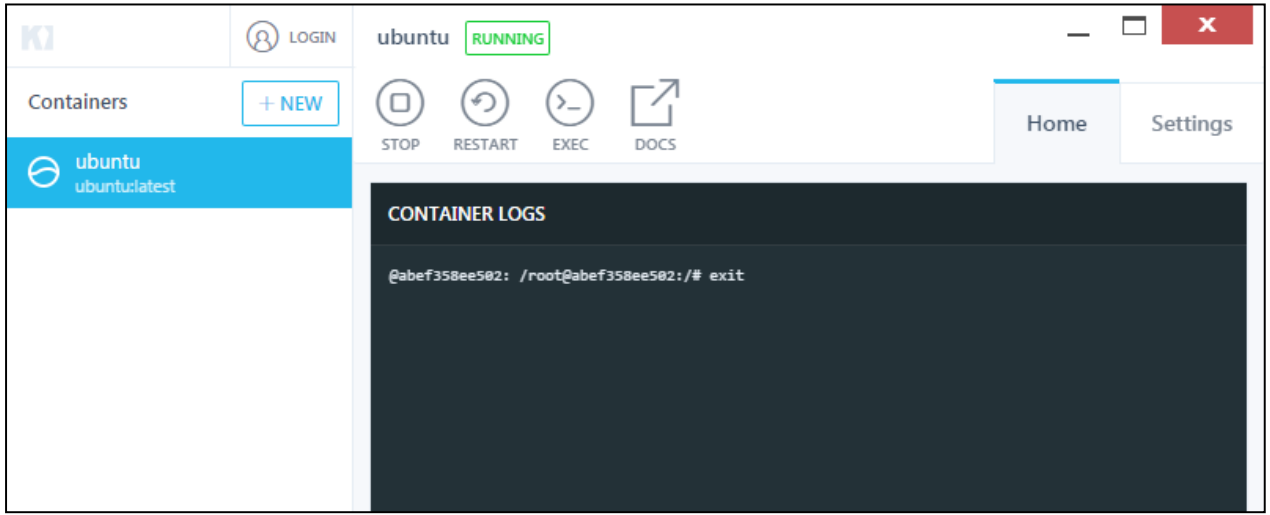


Dockerオーケストレーション

- Dockerには分散アプリケーションのオーケストレーションのための3つのツールがあります。
- Docker Machine
  - ホストをプロビジョニング(自動構築)し、そのホストにDocker Engineをインストールするツール
- Docker Compose
  - 複数のコンテナで構成するアプリケーションを作成・管理するツール
- Docker Engine swarm mode
  - Docker Engineのクラスタを管理するネイティブツール
  - Docker Engine 1.12から利用可能

Kitematic

- Docker用のGUI
- コンテナの起動と停止
- Docker Hubに格納したイメージへ簡単にアクセス
- Mac OS XとWindowsで動作
- 自動更新をサポート



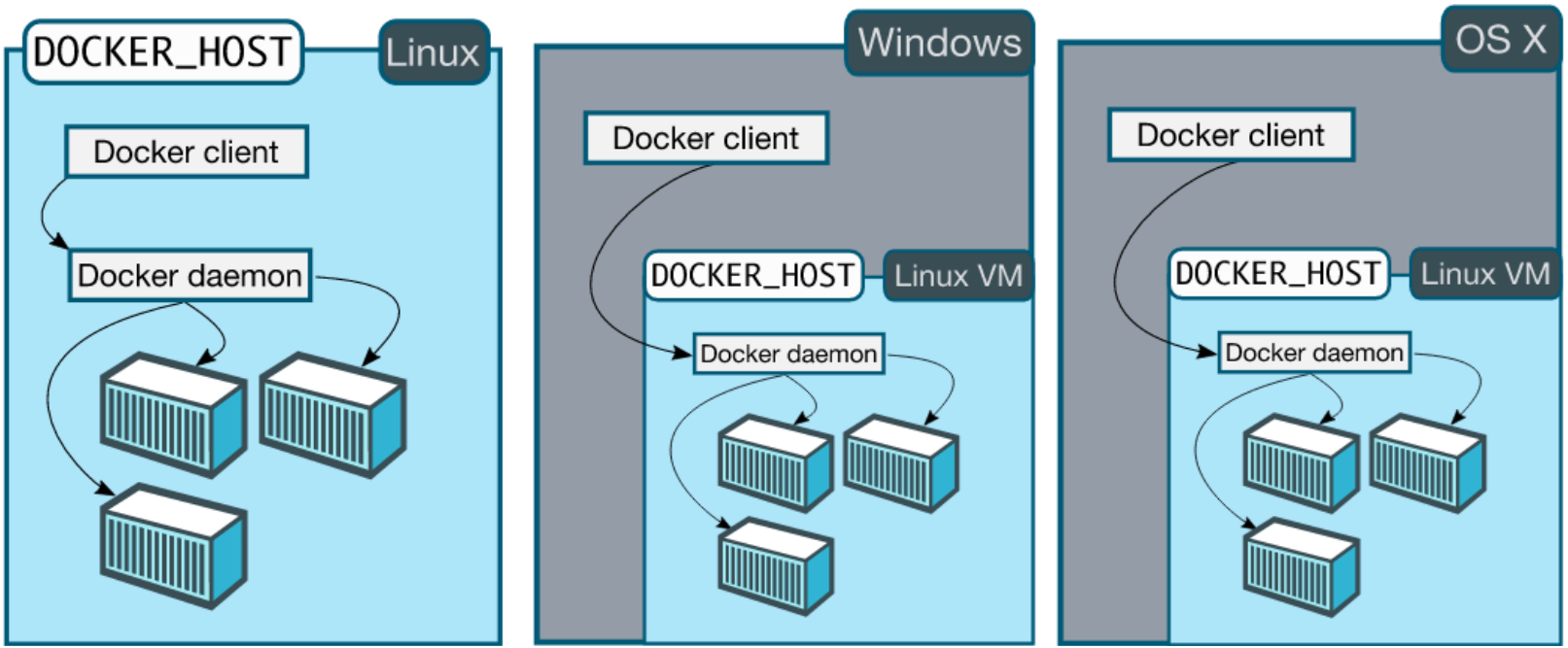
Docker Toolbox

- Docker EngineはWindowsとMac OS上では直接実行できません。
- Linux仮想マシンをセットアップしてから、Docker Engineをインストールする必要があります。
- Docker Toolboxは次のツールをセットアップします。
  - Oracle VirtualBoxで軽量なLinux仮想マシンを実行
  - Docker Machine
  - Docker Engine
  - Docker Compose
  - 仮想マシンをコマンドラインで操作するための設定済みシェル
  - Kitematic

Docker for MacおよびWindows

- Docker Toolboxを置き換えるツール
- WindowsあるいはMacのネイティブの仮想化を利用
- VirtualBoxを必要としない
- https://www.docker.com/products/docker#/windows
  - Windows 10 professional edition 64ビット版が必要
- https://www.docker.com/products/docker#/mac
  - Yosemite 10.10.3が必要

Linux vs WindowsおよびMac OS



Docker Cloud

本番環境でDocker化アプリケーションのデプロイと管理する最良の手法

開発者向け: DevOpsを主導し、分散アプリのデプロイと管理を自身でできるようになります

運用者向け: クラウドとプライベート基盤を横断する分散アプリのデプロイと管理を簡単にできるようになります

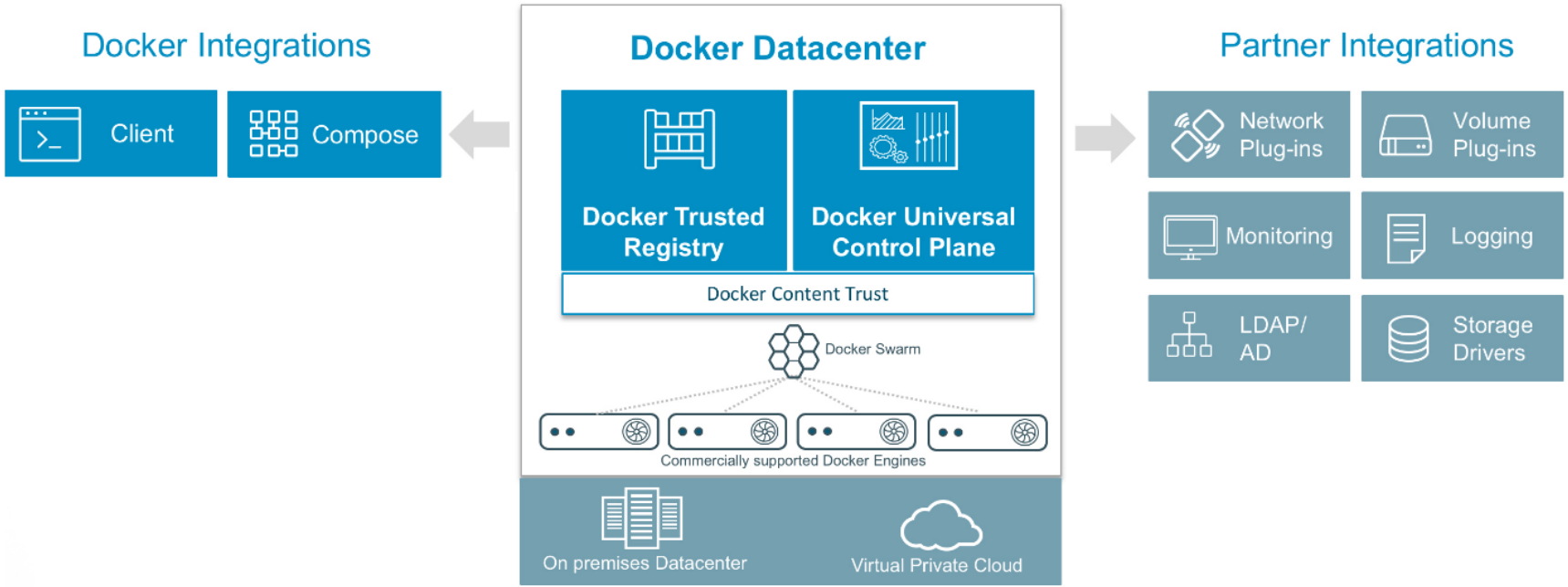
単一のプラットフォーム: 開発者と運用者が一貫したワークフローで開発から運用まで使えるようになります

Docker Datacenter

*Docker Datacenter*は、オンプレミス*CaaS*インフラストラクチャのためのオープンソースと商用製品の統合ソリューションです。

- Docker Toolboxによる開発者の統合
- イメージをDocker Trusted Registryに「出荷」できる
- Universal Control Planeを用いたコンテナのデプロイ
- Docker APIの完全なサポート
- 商用サポート版のDocker Engine (CS Engine)
- ロールベースのアクセス制御
- LDAP、監視、ログツールのようなエンタープライズシステムとの統合

Docker Datacenterの概要



Docker Trusted Registry

Docker Trusted Registry (DTR)は、あなたのインフラでイメージの格納と管理を安全に行うためのレジストリサーバです。

- DTRの機能
  - イメージを格納するためのレジストリ
  - プラガブルなストレージドライバ
  - 管理設定のためのウェブベースのGUI
  - 簡単で透過的なアップグレード
  - 組み込みのシステムメトリクスダッシュボード
  - ロギング

Docker Universal Control Plane

開発者の迅速さによる*IT*運用の制御、そして *Docker*化アプリケーションのポータビリティをもたらす初めてのエンタープライズソリューションです。

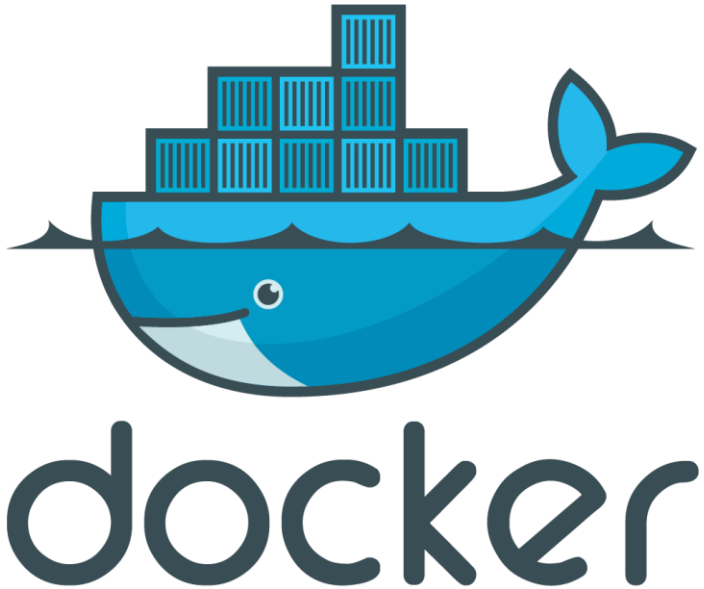
エンタープライズ対応: LDAP/AD 統合、オンプレミスなデプロイ、高可用性ソリューション

開発者は 独力でアプリの構築とデプロイが可能で、運用は アプリインフラのデプロイと管理が可能です。

オープンな**API**、プラグインアーキテクチャ、幅広いエコシステムをサポートする**Docker**ネイティブなソリューション

3章:
Dockerのインストール

docker



本資料は2017年3月時点での抜粋版資料です。
最新のDockerトレーニングについては、お問い合わせください。

ソフトバンク コマース＆サービス株式会社

DevOps問い合わせ窓口

サイト: https://licensecounter.jp/docker/

メール: SBCASGRP-DevOps@g.softbank.co.jp