

vFORUM **2019**

AP142

コンテナ入門

～これから始める人のためのコンテナ講座～

VMware株式会社

NTT - SE 部

ソリューションエンジニア 黒沢 勇

Make
Your
Mark

Agenda

1. なぜコンテナ？
2. なぜ Docker から Kubernetes(k8s)？
3. VMware はコンテナにどう取り組んでいるか？

本セッションのゴール

3つのポイントがわかればゴール達成です




仮想マシン
との違い

仮想マシンとコンテナの
違いを理解する



Kubernetes
の必要性

Kubernetes の
必要性を理解する



課題を
理解する

コンテナ環境の課題を
理解する

なぜコンテナ？

仮想マシンとの違い

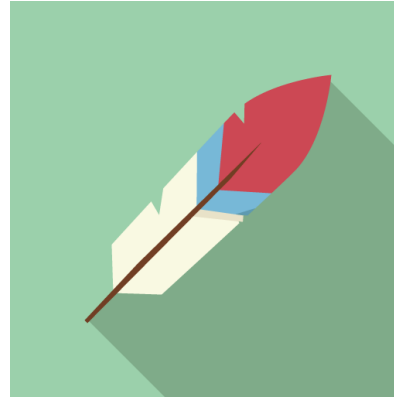
コンテナ技術の利点

なぜ注目を集めているのか？



高速

簡単 / 高速に
起動可能



軽量

効率的な
リソース利用

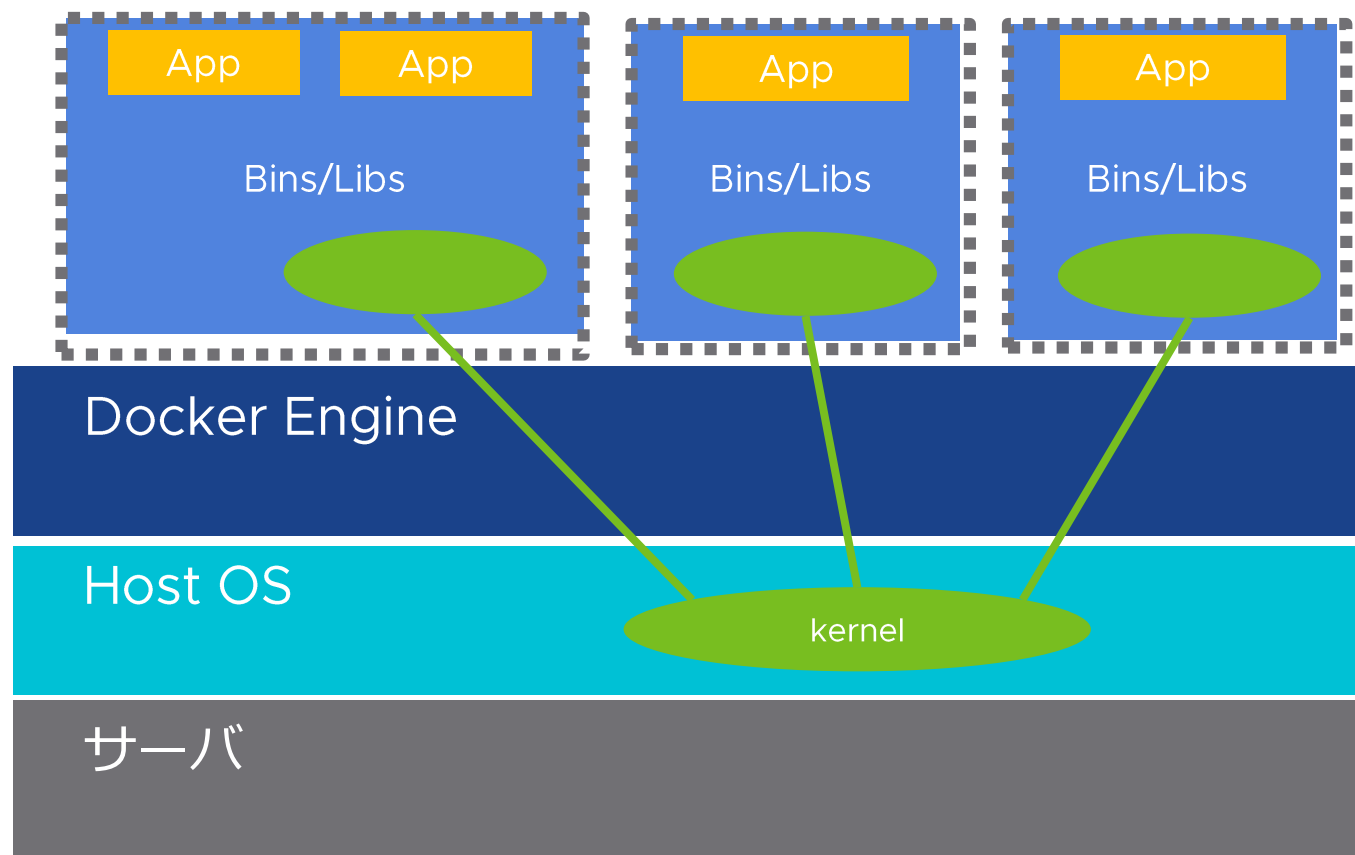


可搬性

アプリケーションを
どこでも起動可能

コンテナの基本的な仕組み

Docker の例



Namespace

- ✓ プロセス、リソースを隔離
(NW / ユーザ / プロセスなど)



コンテナランタイム

- ✓ Docker に代表される
コンテナを動かすソフトウェア



カーネルをコンテナとホストで共有

- ✓ 高効率に配置可能
- ✓ アプリがどこでも起動可能

仮想マシンとコンテナの違い

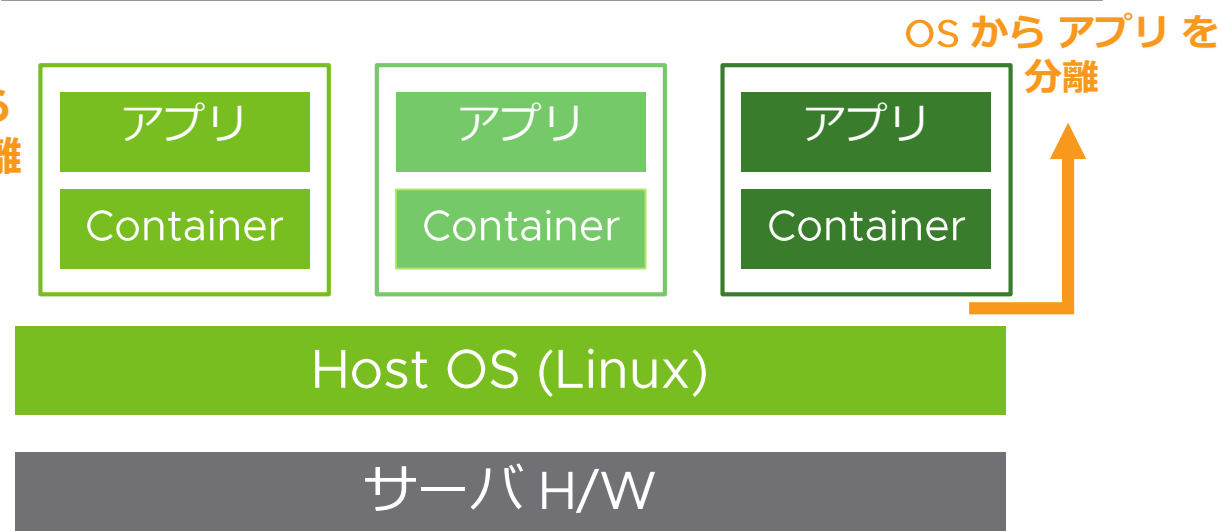
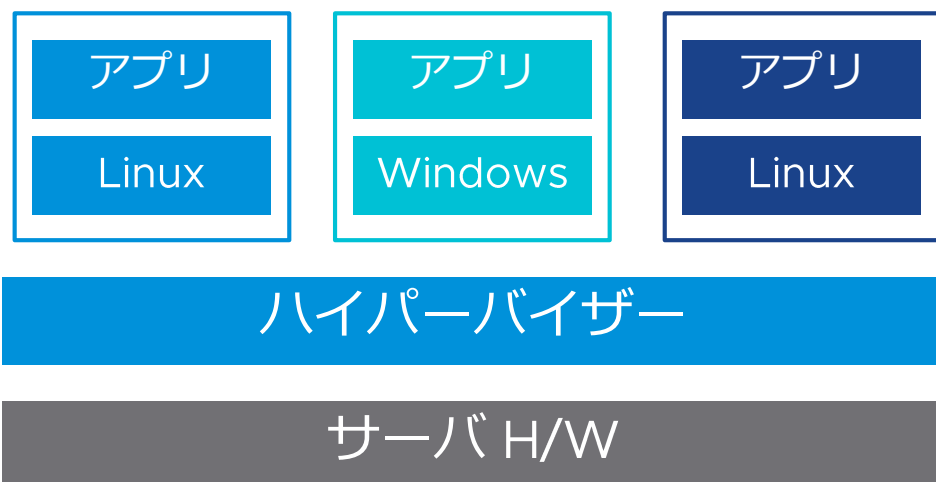
分離レイヤの違い

仮想マシン

- 複数の OS で H/W リソースを共有
- 物理で可能な機能はほぼ実現可能
- OS を問わずに仮想マシンが稼働可能
- 物理と同様に IP 管理をすることが可能

Linux コンテナ

- ホスト OS の カーネル を共有
カーネルの機能を活用、OS とアプリを分離
- オーバーヘッドなしでリソースを有効活用
- API を豊富に提供しているので自動化可能
- IP アドレスは自動設定、外部とは NAT で接続



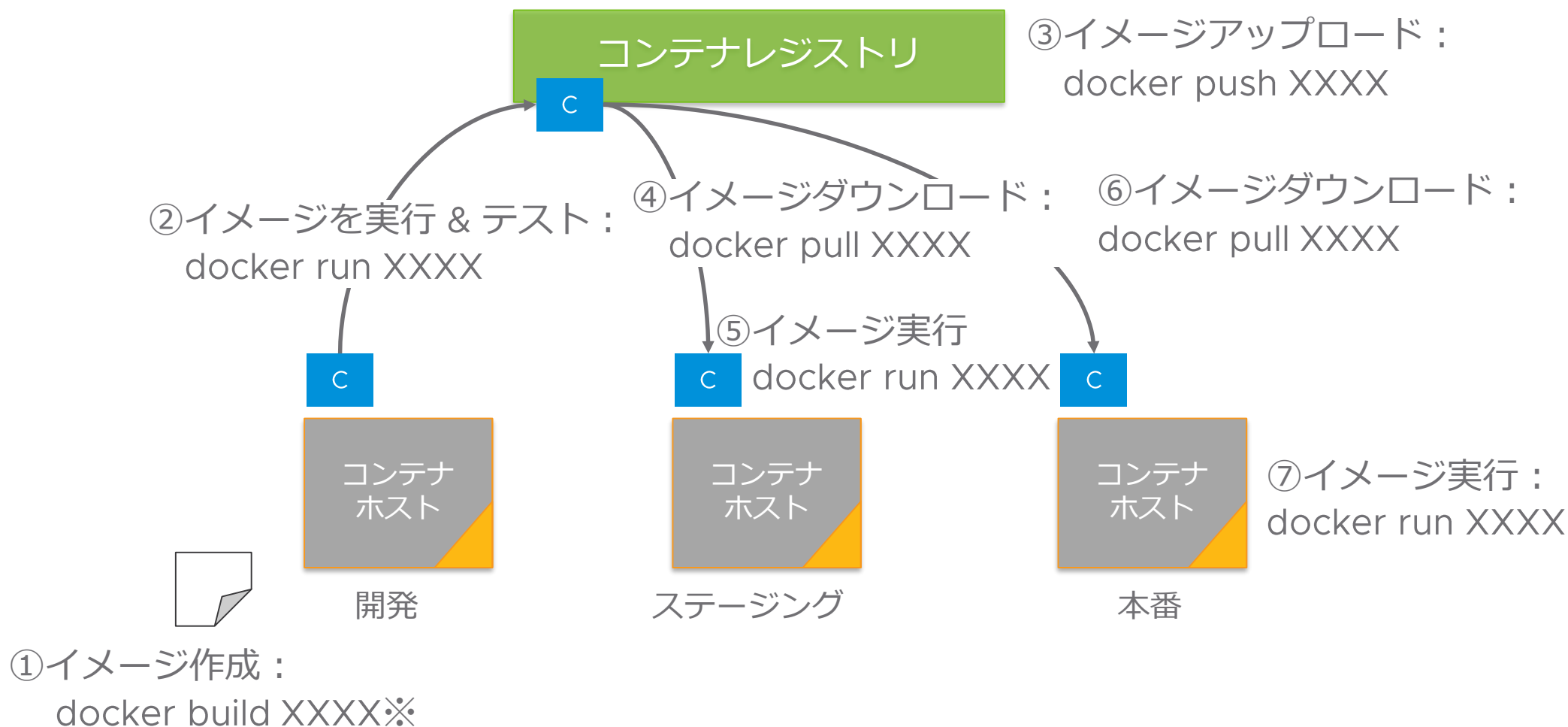
仮想マシンとコンテナを比較

4つの観点から

	仮想マシン	コンテナ技術
既存アプリ	殆どの 既存アプリ に コスト削減などの恩恵あり	既存アプリは 作り直し になるケースが散見
アプリ開発者視点	セルフサービス化などの インフラ提供者の工夫が必要	インフラを意識せずに、 素早く アプリ開発・提供が可能
セキュリティ	各 OS が 完全に分離 セキュリティの仕組みを流用	1つの OS で動くため セキュリティ要件 を検討
運用	既に枯れた運用のため 非常に安定 / 方策が整っている	アプリ側で検討が走り出し、 インフラ運用は置き去りになりがち

Docker の基本的な使い方

コンテナは Docker を使用するのがほぼデファクト



XXXX = dockerfile

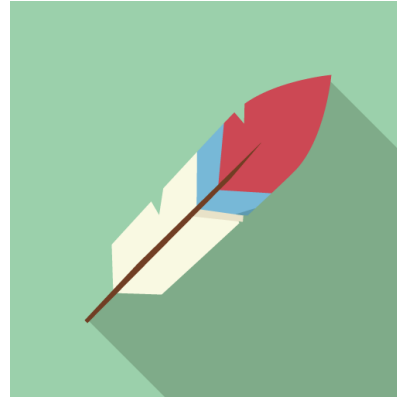
使い方からもDockerの良さ



高速

コマンド1つで

簡単・迅速に
起動可能



軽量

カーネル共有による

効率的な
リソース利用



可搬性

パッケージ化により

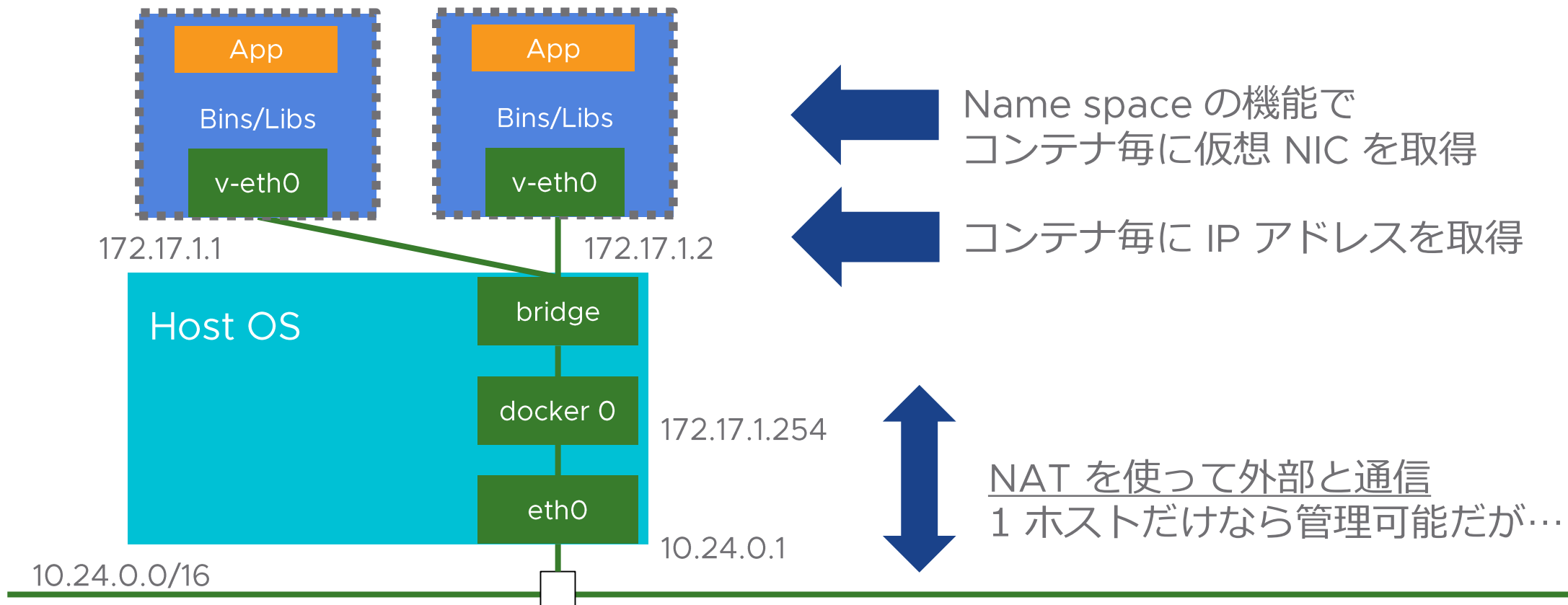
アプリケーションを
どこでも起動可能

なぜ Docker から Kubernetes(k8s) ?

エンタープライズ環境での課題

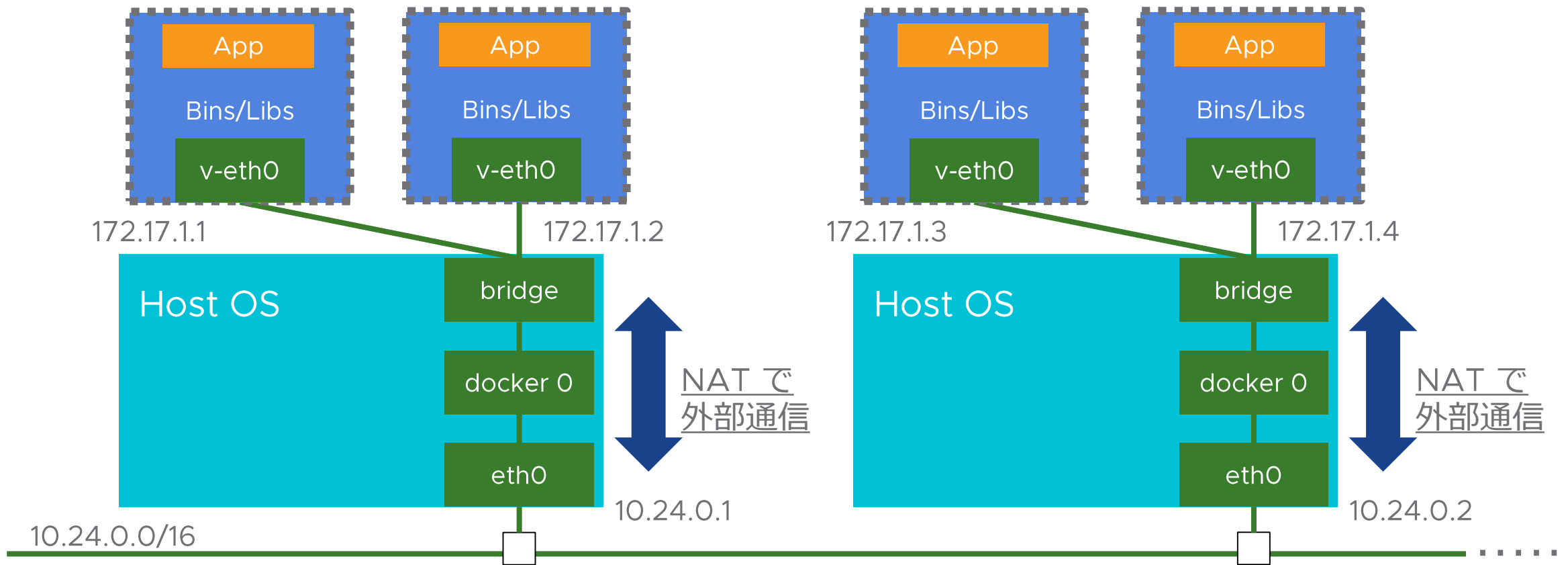
Docker 環境の課題例：ネットワーク

コンテナのネットワークアーキテクチャ



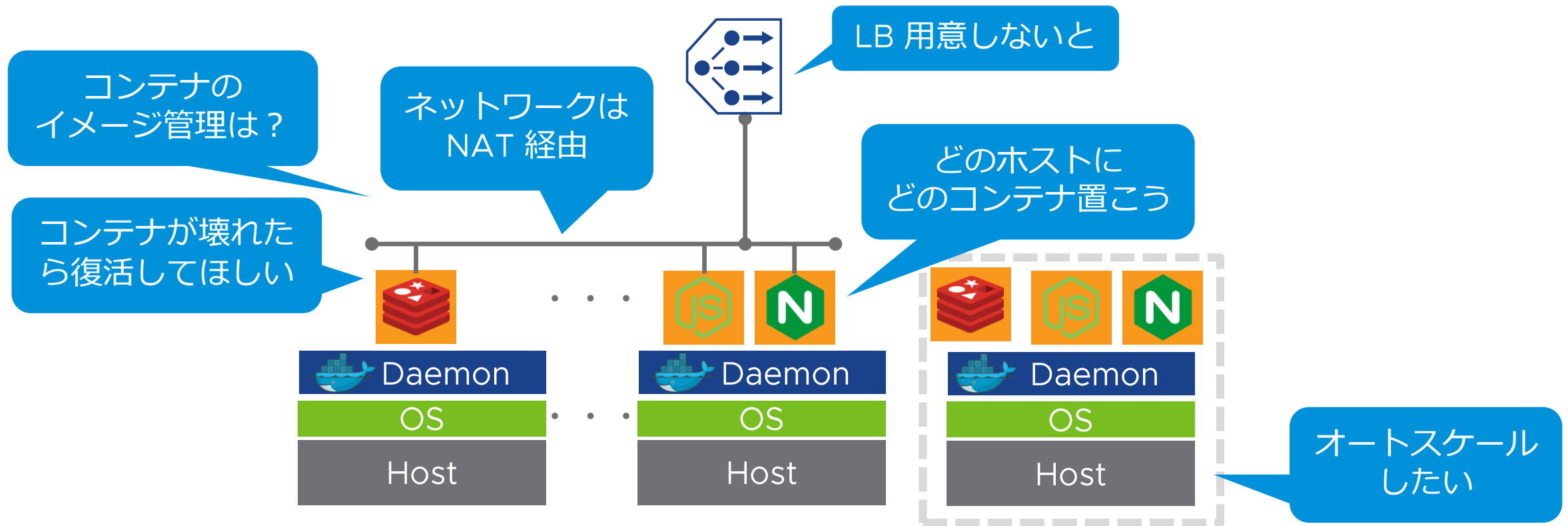
Docker 環境の課題例：ネットワーク

コンテナのネットワークアーキテクチャ



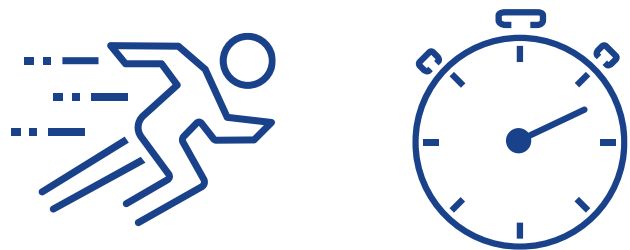
複数ホストでサービスを連携する場合、ネットワークが複雑な形に

Docker 単体環境の課題



エンタープライズ環境ではいろいろな悩みが出てくる

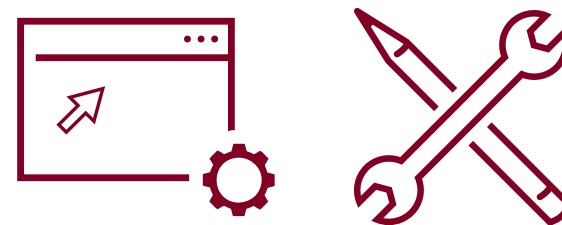
コンテナを開発側と管理者の両面から考える



開発者の目線からみたコンテナの魅力

- 展開 / クローンが高速
- ミドルウェア層を含めた制御が可能
- 少ないリソース オーバーヘッド
- 高い可搬性

**開発環境では コンテナ の
メリットを最大限に享受できる**



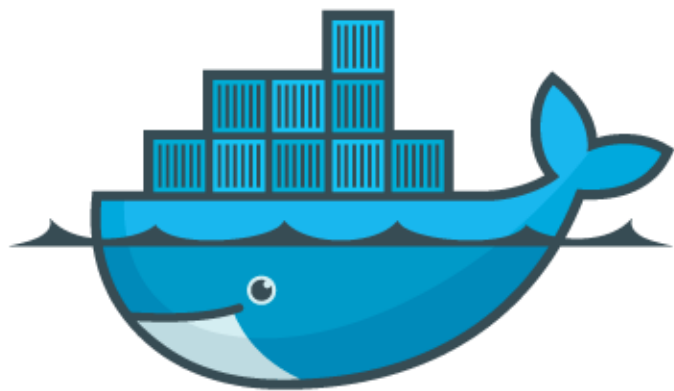
管理者の目線からみたコンテナの懸念点

- セキュリティ
- NAT ベースのネットワーク
- 可用性
- 監視、バックアップなど

**本番環境ではコンテナに関する
懸念点がまだまだ存在する**

Kubernetes 台頭の時代へ

オーケストレーションを活用してコンテナを本番環境で運用



ホスト1つ1つで
コンテナイメージをデプロイ

負荷分散の仕組みは自前で構築

オートスケールも自前で構築



kubernetes

複数ホストで
コンテナイメージを展開

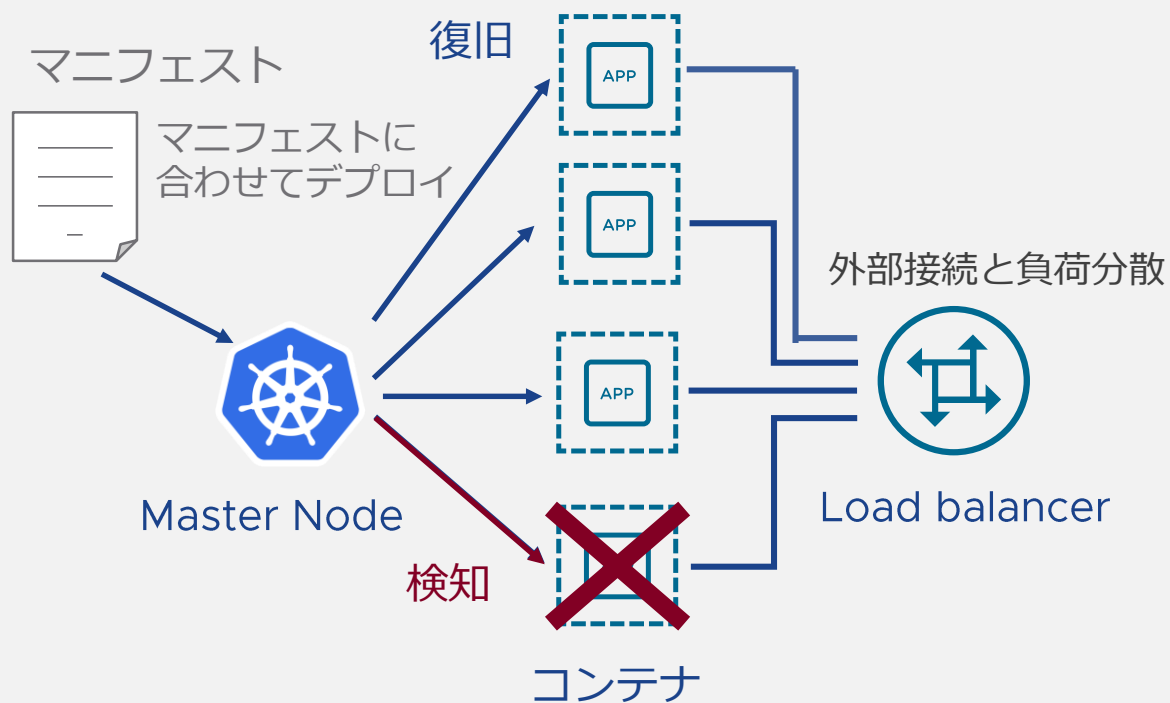
ロードバランサを自動配備

オートスケールの仕組みを持つ

Kubernetes とは？

コンテナ管理を自動的に行ってくれるオーケストレーションシステム

Kubernetes 動作イメージ



Kubernetes ができること

複数コンテナをポリシー準拠でデプロイ

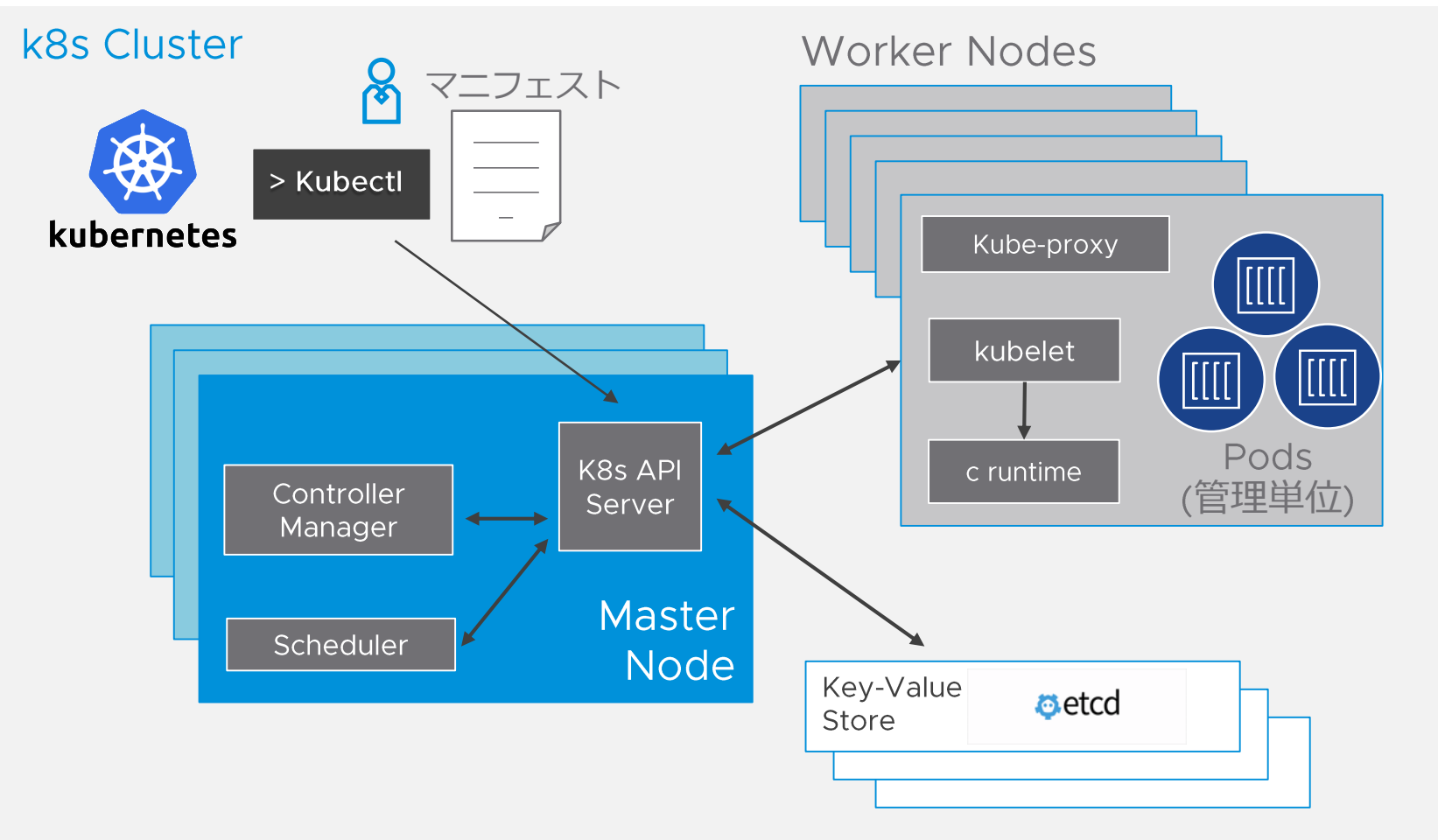
負荷分散機能の提供

コンテナの外部接続ネットワーク提供

障害時はコンテナを自動スケーリング

Kubernetes アーキテクチャ / コンポーネント

内部構造



k8s Cluster は
Master と Worker で役割を分割

Master

- コンテナの管理

Worker

- コンテナの実行

マニフェスト

- 動かし方を定義
- k8s リソース設定

etcd

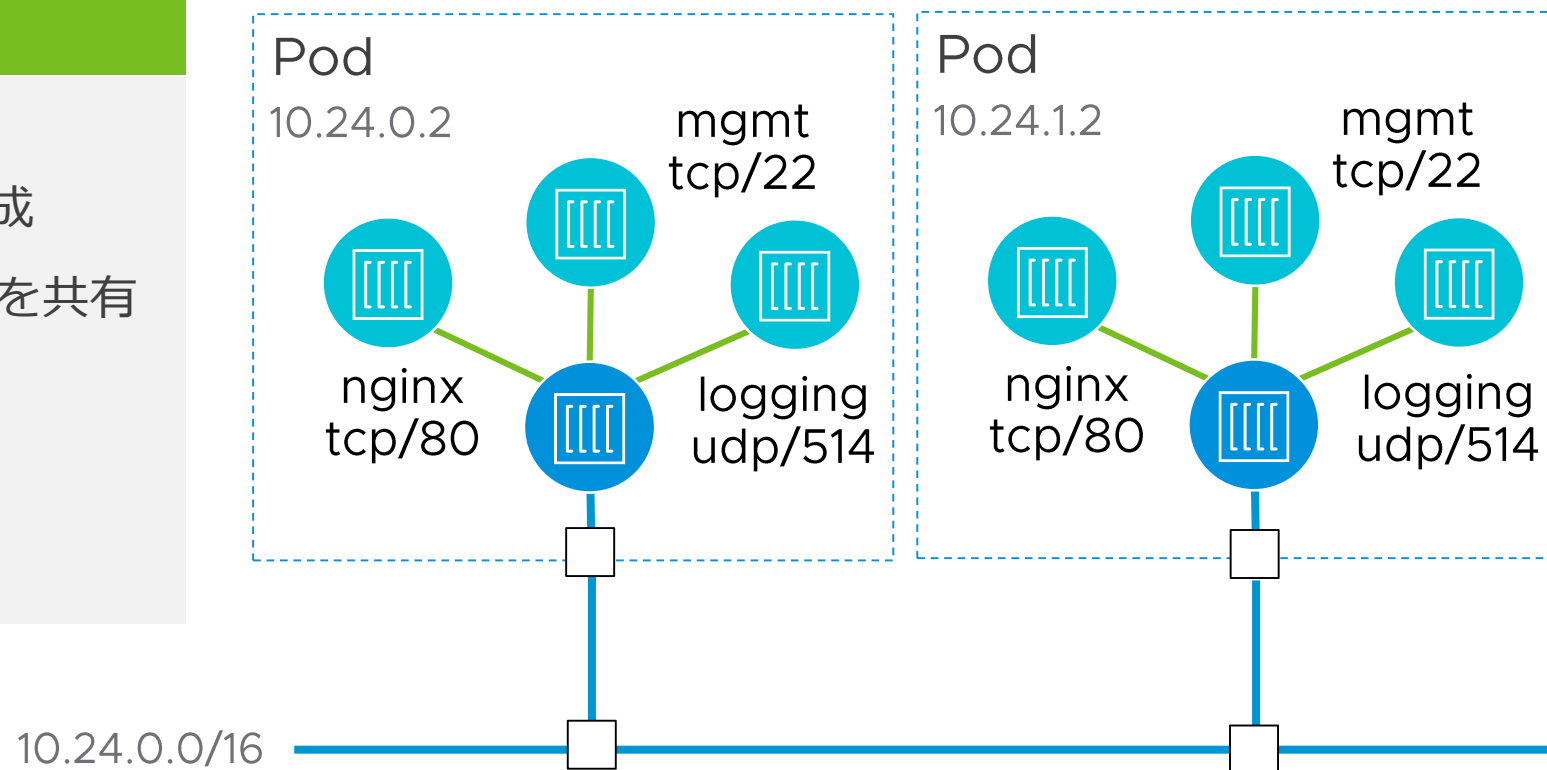
- Key-Value Store
- k8s のネットワーク設定を保存

Pod とは？

Pod 概要

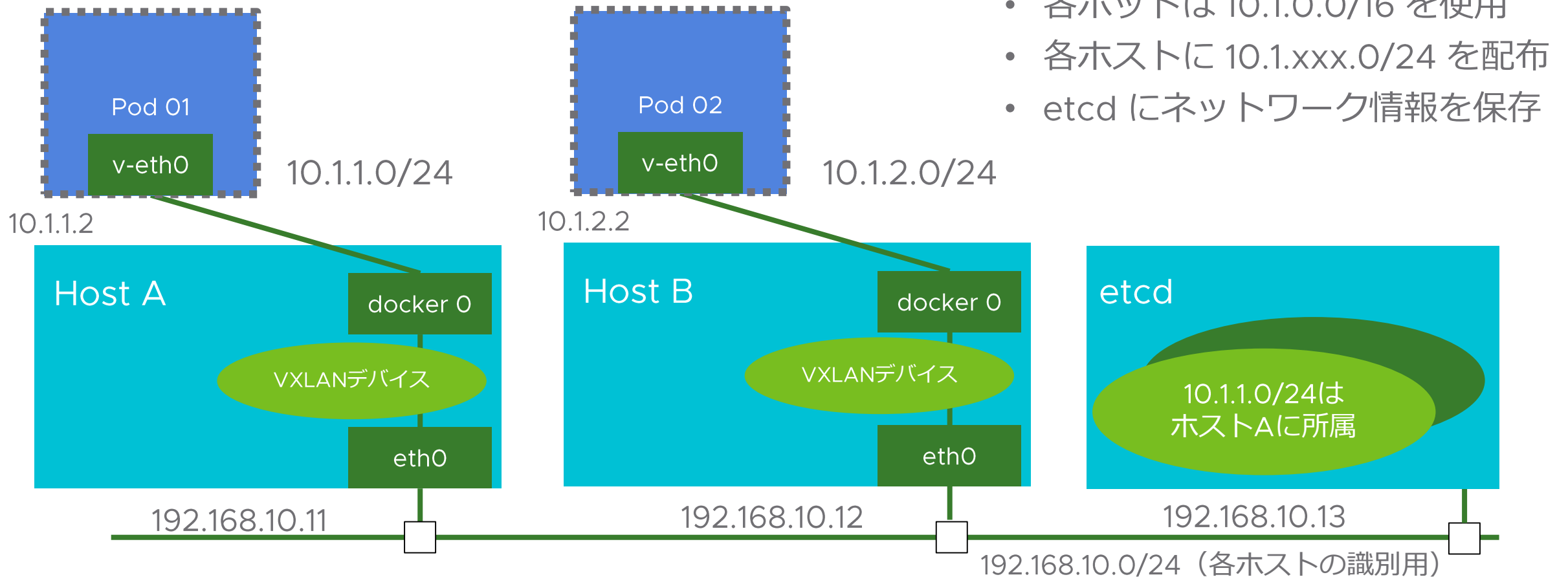
- ◆ k8s のコンテナ最小管理単位
- ◆ 1つ、または複数のコンテナで構成
- ◆ 同一 pod のコンテナはリソースを共有
 - ✓ ネットワーク
 - ✓ ホスト名
 - ✓ IPC (inter-process communication)
 - ✓ Storage

— IPC (プロセス間通信)
— 外部 IP トラフィック



Kubernetes のネットワーク

オーバーレイネットワークにより NAT 問題を解決※



- 各ポッドは 10.1.0.0/16 を使用
- 各ホストに 10.1.xxx.0/24 を配布
- etcd にネットワーク情報を保存

オーバーレイによるカプセリングにより各 pod 同士の通信をシンプルに

K8s は全てのコンテナの問題を解決するのか？

課題はまだある

コンテナが壊れたときの
可用性担保

LB による負荷分散

ネットワークを複数の
OSS で管理

コンテナのイメージ管理は？

そもそも k8s 構築が難しい

コンテナの配置関係の
可視化 / 管理

Master node の死活監視で
可用性を担保

LB の自動配備により
負荷分散

NW 運用の煩雑化

k8s 単体は提供せず

ツールは提供されているが
不十分な場合が多い

GUI や可視化ツールは
不十分

Docker 単体運用の問題は一部解決されたが、残っている課題 / 新たな課題が残る

VMware は コンテナにどう取り組んでいるか？

残った課題に対して VMware はどう取り組むか

課題と解決策

ネットワークを複数の
OSS で管理

コンテナのイメージ管理は？

そもそも k8s 構築が難しい

コンテナの配置関係の
可視化 / 管理



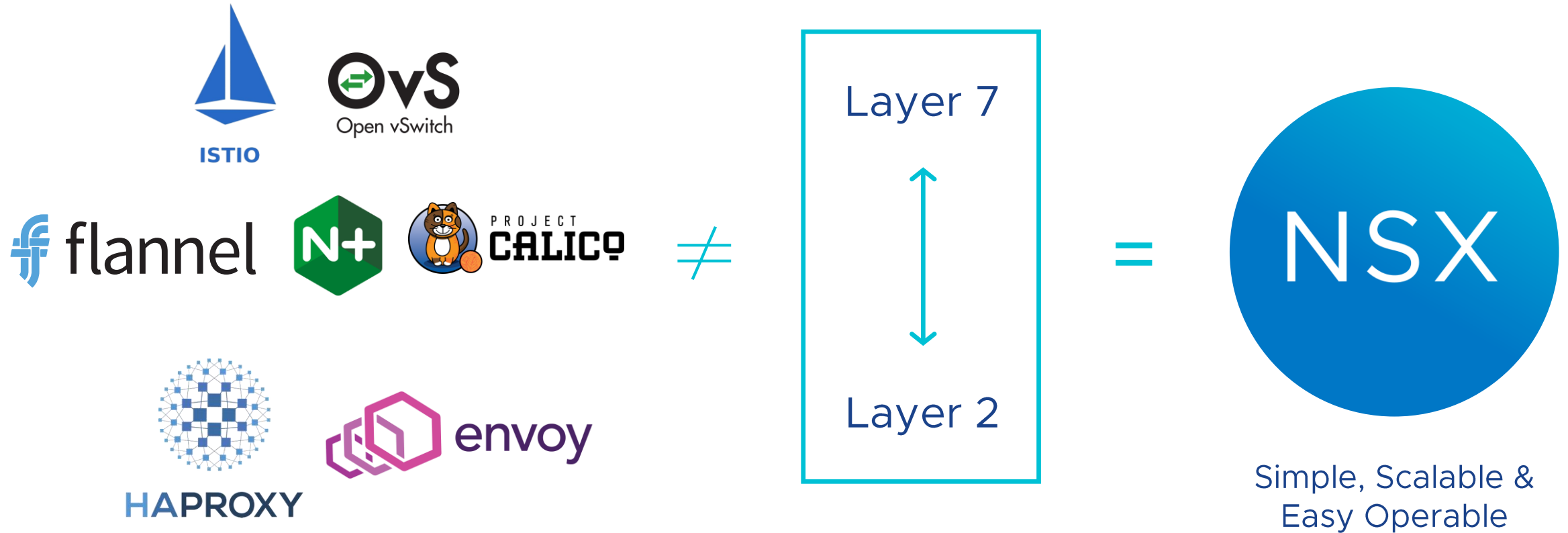
VMware NSX®

VMware Enterprise PKS

VMware Tanzu
Project Pacific

コンテナネットワーキングは NSX で最適化

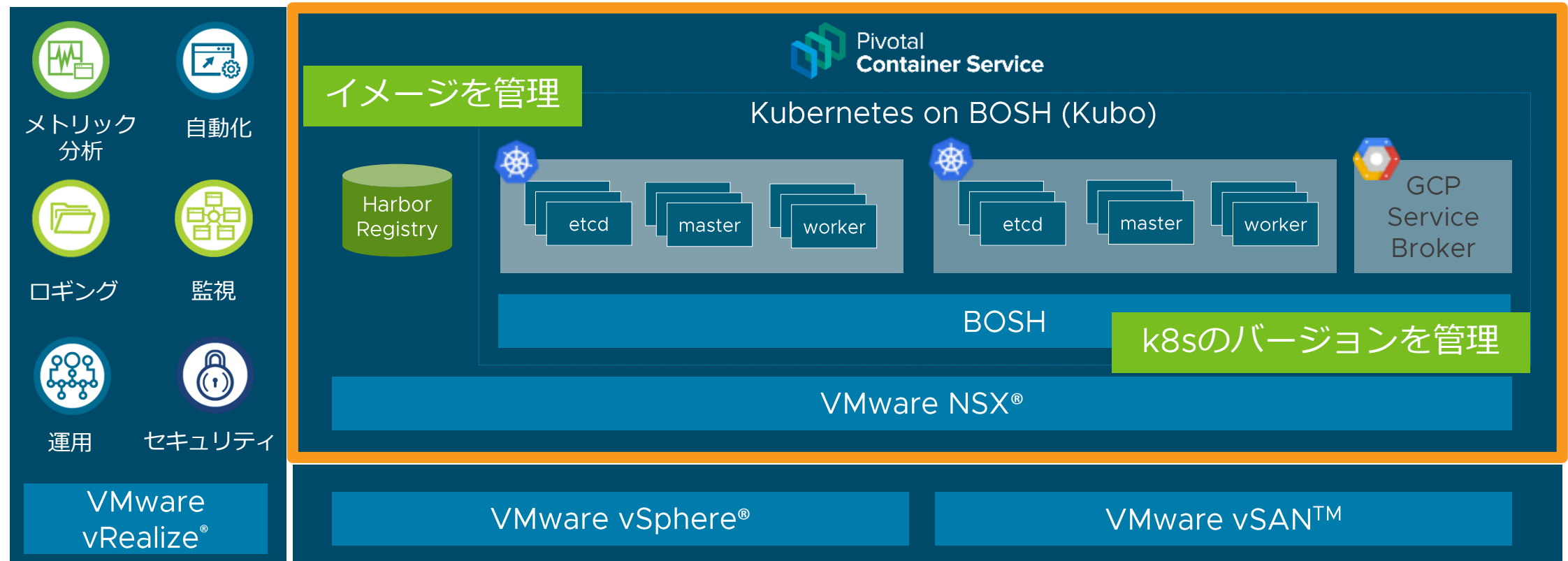
NSX により包括的なコンテナネットワーキングに対応



複数の OSS を使用する煩雑な世界から NSX のシンプルなコンテナネットワークの世界へ

VMware Enterprise PKS

k8s を簡単に構築 / 運用できるツールとして提供



k8s クラスタをコマンド1つでデプロイ

イメージやバージョン管理する仕組みを搭載

VMware Tanzu / Project Pacific

k8s の課題を ハイパーバイザー との統合で解消

VM ベースの k8s も、従来の VM も、ベアメタルコンテナも使用可能



Developer

k8s API を
セルフサービスで使用

コンテナアプリを
安定した基盤で Deploy



Kubernetes
Clusters



Virtual
Machines



Native
Pods



IT Operator

今までと同じ
やり方で k8s も管理

ストレージや
ネットワークも
今まで通り管理

VMware vSphere®

Supervisor Kubernetes Cluster

ESXi Cluster



Storage

Networking



VMware
vCenter®

Developer には k8s による
デプロイの迅速化を提供

IT Operator には k8s を
慣れ親しんだ形での運用を提供

本セッションのゴール

3つのポイントが分かればゴール達成でした



仮想マシン
との違い

OS レベルで分離するか、
アプリレベルで分離するかが
最大の違い



Kubernetes
の必要性

可用性、自動化の観点から
オーケストレーションが
不可欠



VMware
の取り組み

インフラ運用、体制変更、
スキルセットの取得などが課題
VMwareの取り組みで解決



Thank You