Docker入門

ENOG32 Meeting

Hayato Imai

Dockerとは

Dockerの背景

- ２０１３年３月にOSSとして公開
- dotCloud社（現Docker社）が開発
- コンテナ型サーバー技術のコアコンポーネント

Dockerの目的

- アプリケーションの構成が多岐にわたる
- 開発、ステージング、本番など、環境が多数
- オンプレ、クラウド
- これらのアプリケーションの管理やデプロイの課題を解決することが目的

Dockerの役割

- アプリケーションイメージの構築
- アプリケーションイメージの配布
- アプリケーションの実行

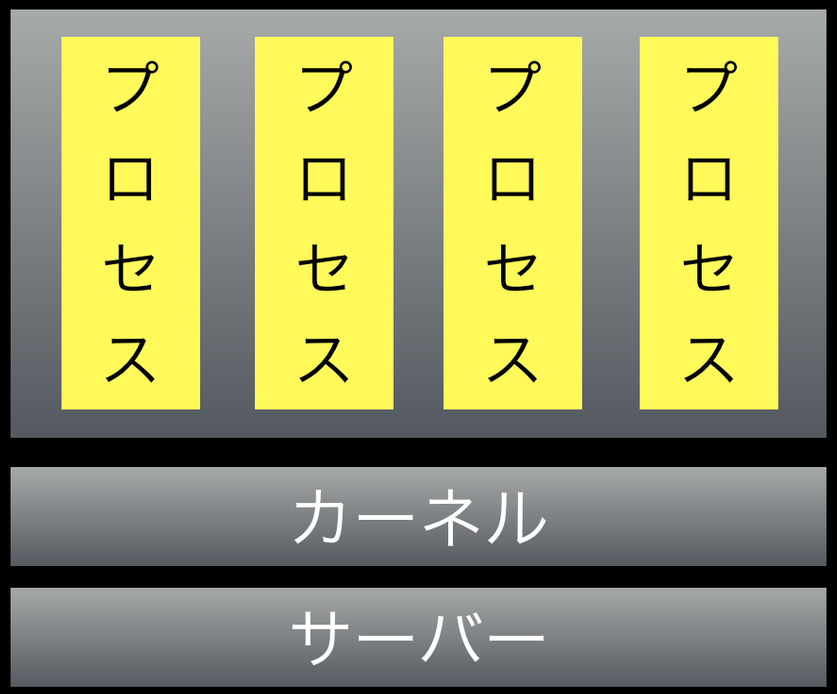
コンテナ型仮想化

VM型仮想化と
コンテナ型仮想化

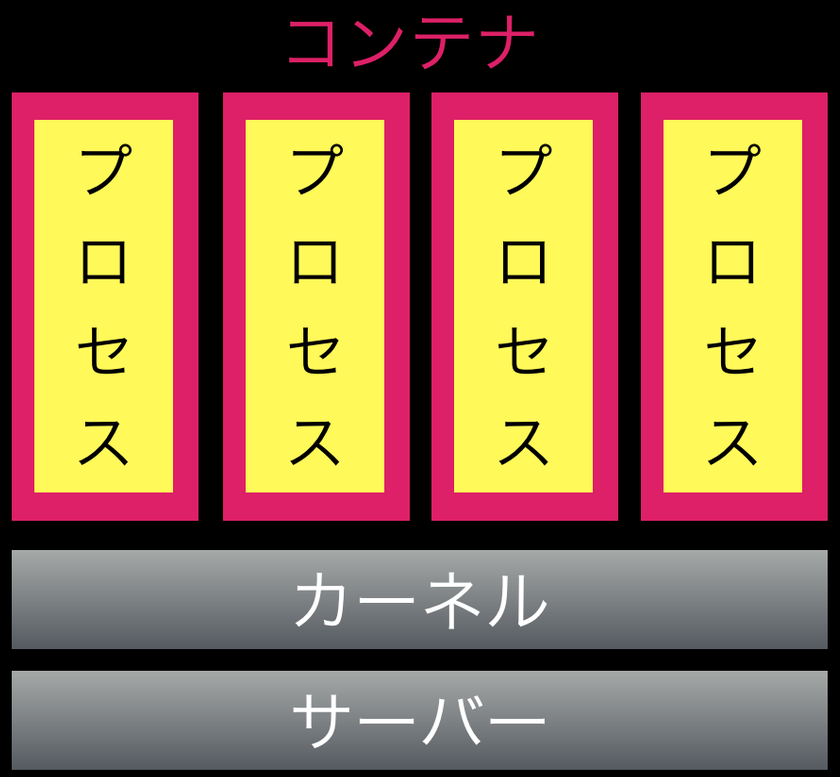
VM型仮想化

コンテナ型仮想化

コンテナによる環境の分離



通常の環境



コンテナで隔離した環境

コンテナに割り当てられる主なリソース

リソース	説明
CPU	CPUコア数やCPU使用率を設定する
メモリ	使用可能なメモリ量を制限する
プロセステーブル	コンテナ内のプロセスだけが見えるようにする
ファイルシステム	ルートディレクトリの内容を変更する
ネットワーク	仮想NICを割り当てる

コンテナ型仮想化の
メリットとデメリット

メリット	デメリット
・パフォーマンス劣化が少なく高速に起動できる ・ハードウェア固有の機能を必要としないため仮想マシン上でも動作可能	・カーネルは共通のため異なるOS同士を同時に動作できない ・ホストOSがクラッシュするとすべてのコンテナは道連れ

はじめてのDocker

Dockerの
インストール

CentOS 7

```
$ sudo yum install docker
$ sudo systemctl start docker
$ sudo systemctl enable docker
```

CentOS 6.5

```
$ sudo yum install epel-release
$ sudo yum install docker-io
$ sudo service docker start
$ sudo /sbin/chkconfig docker on
```

https://docs.docker.com/installation/

Dockerイメージのダウンロード

```
$ docker pull debian:jessie
latest: Pulling from debian
3cb35ae859e7: Download complete
41b730702607: Download complete
...
Status: Downloaded newer image for debian:jessie
```

Dockerコンテナの起動

```
$ docker run debian:jessie cat /etc/debian_version
8.0
```

Dockerイメージの構築（1）

Dockerfileの準備

```
$ cat <<EOF >Dockerfile
FROM debian:jessie
RUN apt-get update && \
  apt-get -y install libapache2-mod-php5 && \
  apt-get clean && \
  rm /var/www/html/index.html
COPY index.php /var/www/html/index.php
VOLUME /var/log/apache2
EXPOSE 80
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
EOF
```

Dockerイメージの構築（2）

index.phpの準備

```
$ cat <<EOF >index.php
<?php
print "Hello World"
EOF
```

Dockerイメージの構築（2）

Dockerイメージをビルド

```
$ docker build -t enog32-hello-world .
Sending build context to Docker daemon 3.072 kB
Sending build context to Docker daemon
Step 0 : FROM debian:jessie
 ---> 41b730702607
...
Successfully built ab82e309b5e9
```

Dockerイメージの構築（3）

ビルドしたDockerイメージからコンテナを起動

```
$ docker run -p 80:80 -d enog32-hello-world
$ curl localhost/index.php
Hello World
```

Dockerイメージを
レジストリに登録

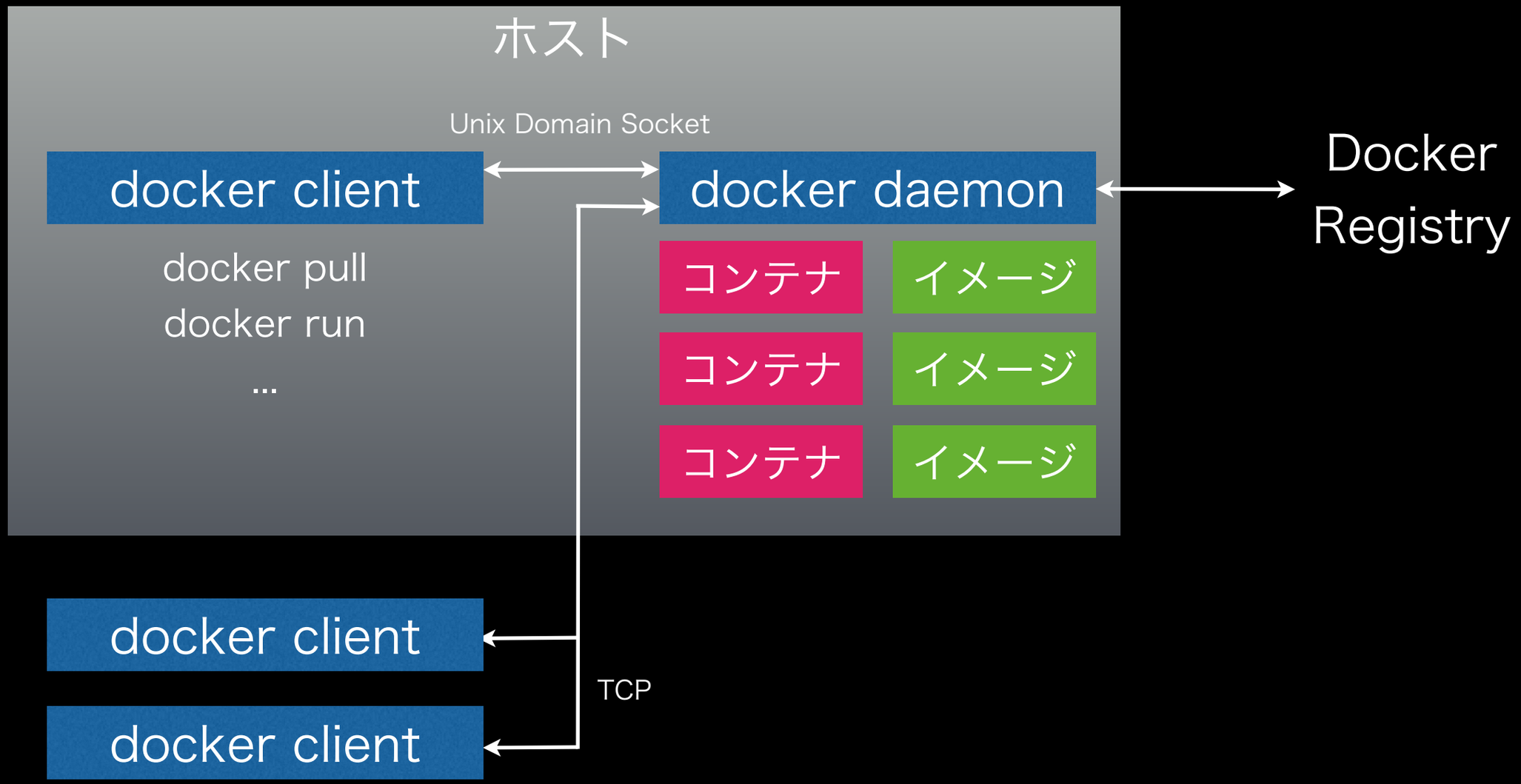
https://registry.hub.docker.com/

```
$ docker login
Username: username
Passowrd:
...

$ docker build -t username/enog32-hello-world .
...

$ docker push username/enog32-hello-world
...
```

Dockerの

アーキテクチャ

クライアント/サーバー


ホスト
Unix Domain Socket
docker client
docker daemon
docker pull
docker run
...
コンテナ
イメージ
コンテナ
イメージ
コンテナ
イメージ
Docker
Registry
docker client
docker client
TCP

Dockerの構成要素（構築）

Dockerイメージ

- Dockerコンテナを起動する際に利用されるテンプレート

- アプリケーション実行のためのすべてのファイルを含む

- 既存のイメージを利用して新しいイメージを作成することも可能

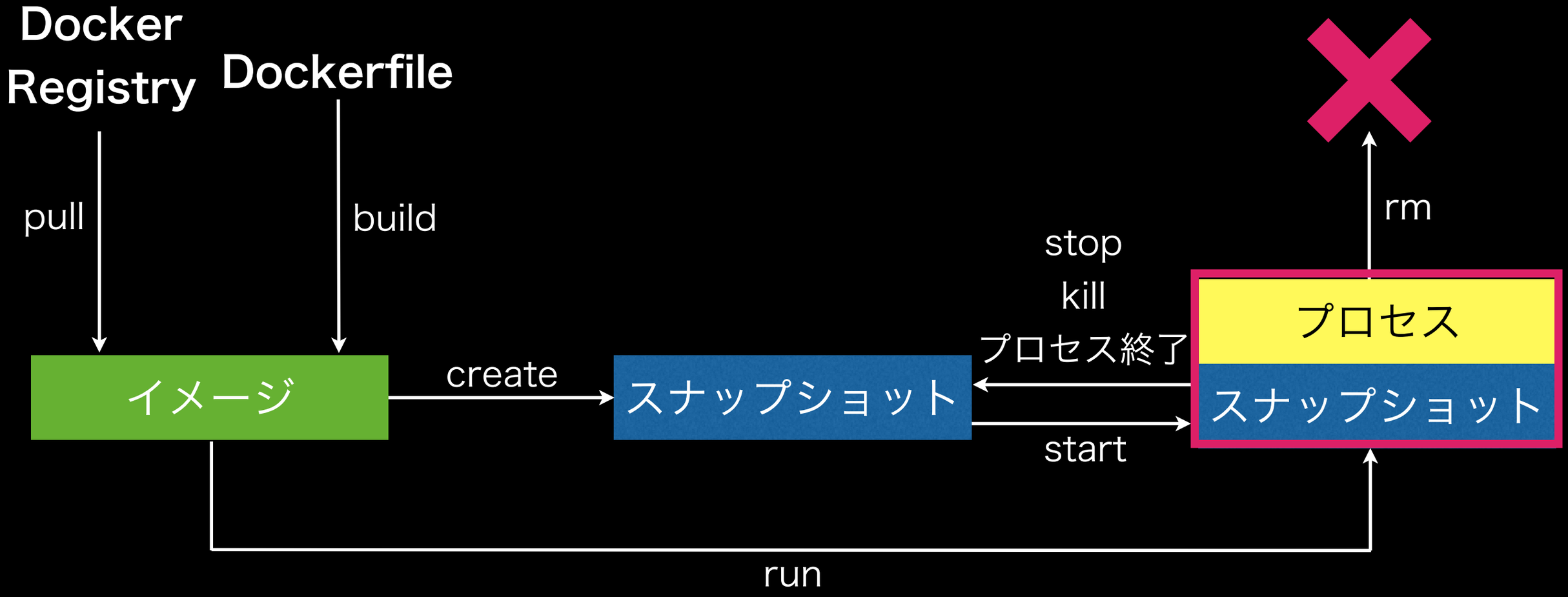
Dockerの構成要素（配布）
Dockerレジストリ

- Dockerイメージを管理する
- 公式のレジストリはDocker Hub Registry
- 自前で用意することも可能
- サードパーティーのレジストリサービスも有

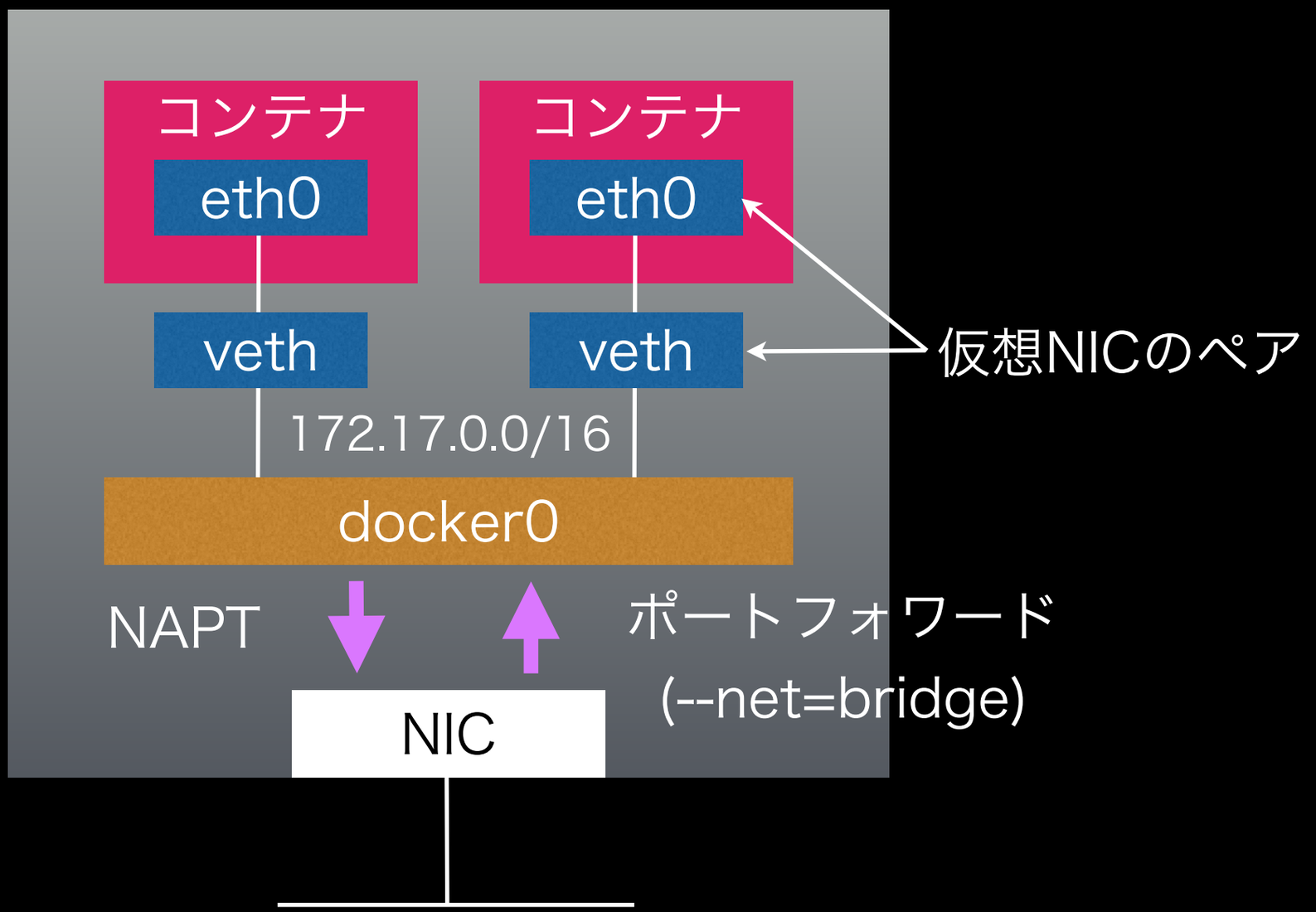
Dockerの構成要素（実行）
Dockerコンテナ

- Dokcerイメージを元にコンテナを起動しアプリケーションを実行する
- 独立したリソースが割り当てられる
- リソースの使用制限を設定可能

Dockerコンテナの
ライフサイクル



Dockerコンテナのネットワーク



Dockerの

ユースケース

PaaS

- git pushするタイミングで新規にコンテナを作成してアプリケーションをデプロイする（Herokuスタイル）
- Dokku, Flynn, DeisなどのDockerを利用したOSSのPaaS

アプリケーションの配布

- FFmpeg、ImageMagick、RRDToolなどの環境構築に手間がかかるアプリケーションをイメージ化して配布

- WordPressやRedmine、 Jenkinsなどのイメージを利用することで環境構築の手間がかからない

CI

- コンテナ内の無駄なモジュール、ライブラリがインストールれされていないクリーンな環境でアプリケーションのビルドやテストを行う
- Jenkins + Docker
- Docker対応のCIサービス

共有ホスティング環境

- ランタイムやミドルウェア、アプリケーションのコンテナを利用（Apache, PHP, MySQL, WordPress, etc.）
- データはホスト側で運用し、コンテナからはVolume機能で参照する
- suExecに比べ独立性が高くセキュア
- 実行環境の入れ替えや切り戻しが容易

ありがとうございました