Docker入門

Dockerを使ってみよう。

自己紹介

- 重本　尚志
- 略歴：
  - 徳島大学工学部卒業（2008-03）
  - 独立系IT企業に新卒として入社（2008-04）
  - C#やJavaを中心（クラサバ多め）に案件を転々とする。
  - 退職・独立（2017-01）
- 趣味：トレーディングカード収集
- 好きな食べ物：奈良漬、味噌ラーメン

目次

- Dockerとは？
- 今回の利用環境
- Hello World
- コンテナを作って動かしてみる

Dockerとは？

- コンテナ型のアプリケーション実行環境
  - コンテナとは、アプリケーションを実行するための一揃いのリソース
  - コンテナを仮想環境のように扱うことができる
- イメージを共有することにより、複数台に同一環境を構築できる
  - DockerHubでイメージの共有が可能
- Dockerfileに構成を記述することにより、構成内容の把握ができる

Dockerとは？

どんな時に利用するか

- 複数環境をプロジェクトで構築するような場合
  - ①Python2.7 + MySQL
  - ②Python3.0 + PostgreSQL
  - ③Python3.5 + SQLite
- 多人数で同一環境を構築する場合（開発環境、動作確認用環境）

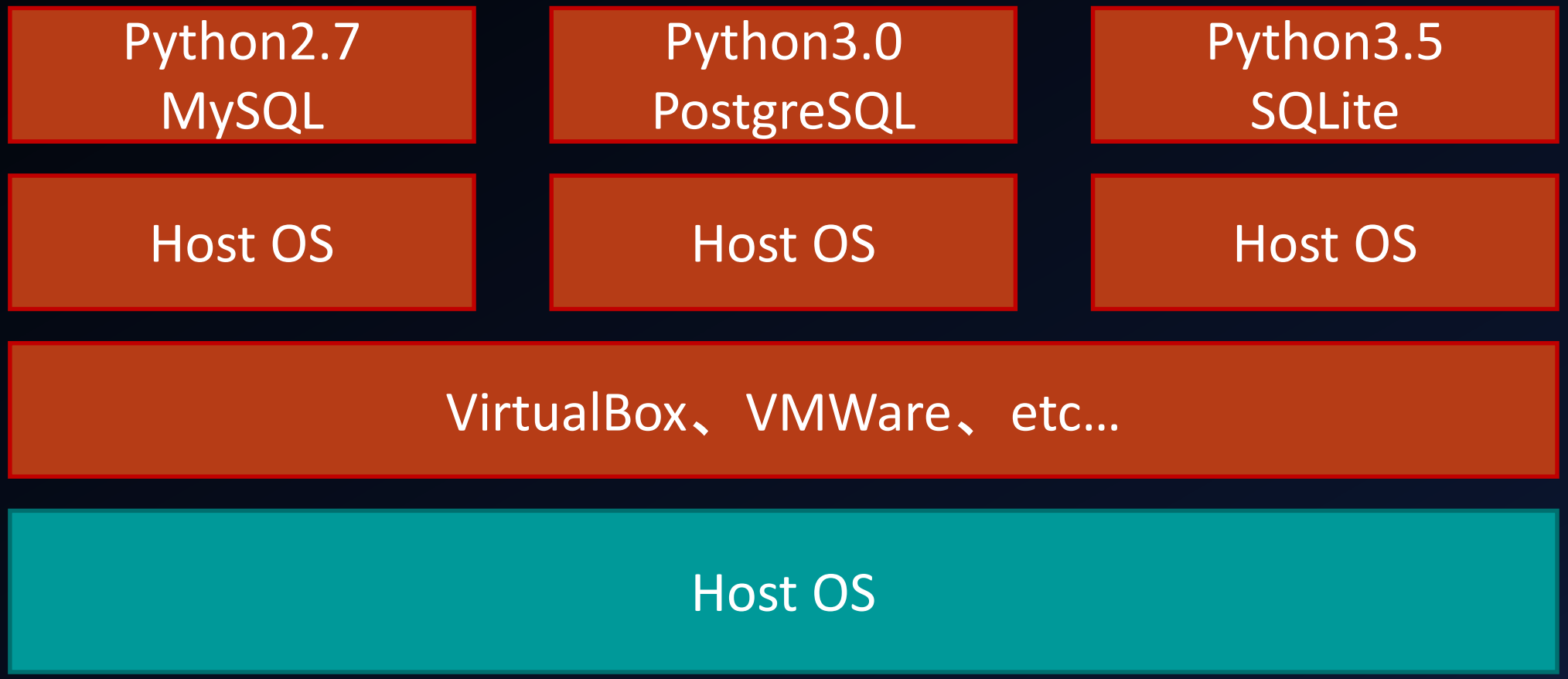
従来であれば仮想サーバー上にそれぞれで構築していた

→Dockerを利用すれば1つのホストOS上で構築できる。

→Dockerイメージを共有することで、環境差異なく素早く構築できる。

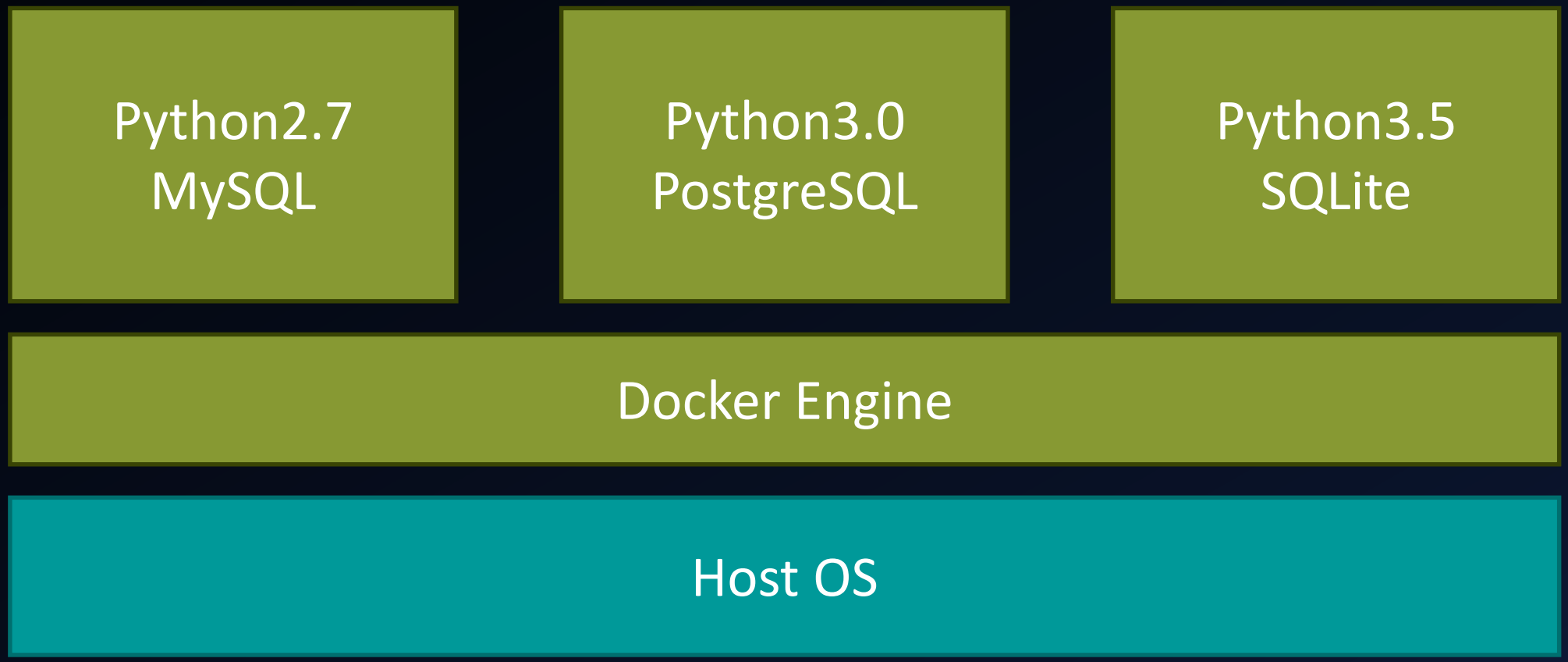
Dockerとは？

仮想技術を利用した場合



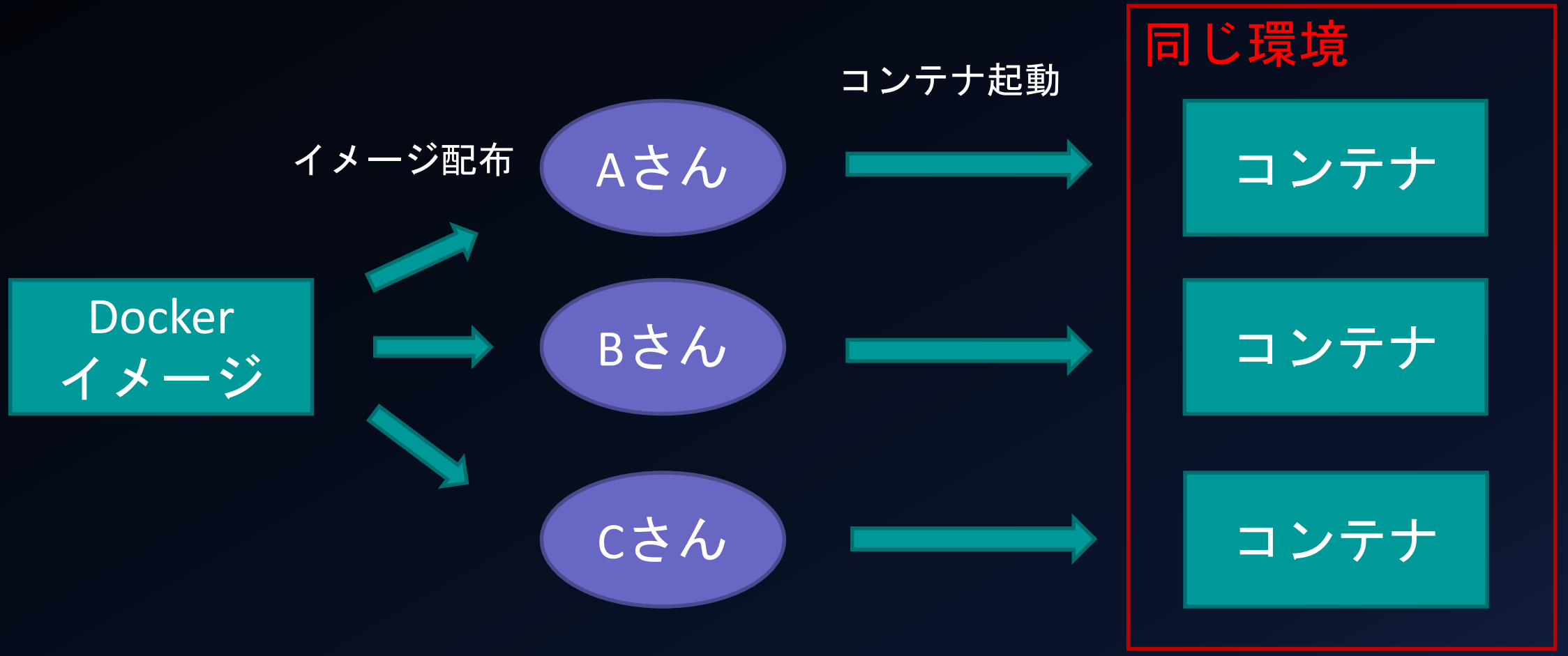
Dockerとは？

Dockerを利用した場合



Dockerとは？

- 多人数で利用する場合（Dockerイメージの共有）



Dockerとは？

Dockerを利用するメリット

- 仮想マシンを起動するより軽快（メモリ、起動時間短縮）
- イメージの共有（DockerHub、またはローカル）をすることによって、複数のマシンに簡単に同一環境を構成できる
- ツールのバージョン差異は複数コンテナを用意することで対応できる

Dockerとは？

Dockerを利用するデメリット

- コンテナで利用できるOSがLinuxベース

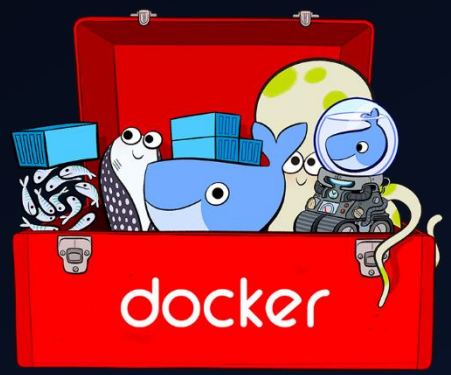
- 環境に変更をかけた場合、コマンドを実行して変更を保存しないと保持されない

Dockerとは？～まとめ～

- Dockerはコンテナを仮想環境と見立ててそれぞれのコンテナで色々な環境を構築できる。
- イメージの共有ができる
- 利用できるOSはLinuxベース
- 環境を変更した場合、コマンドで差分を保存する必要がある

今回の利用環境

- Docker Toolbox（Windows版）
  - インストールすると、VirtualBox上にDocker環境を構築してくれる。
  - インストールされるツールはCUI（Docker QuickstartTerminal）、GUI（Kitematic）がある。
- Virtual Box（Docker Toolboxインストール時にインストールされる）



Hello World

内容・目的

- 「Hello World」と表示だけして終了するコンテナを起動する
- DockerイメージにはUbuntuを利用する
- とりあえず動かしてみる

Hello World

- Docker QuickstartTerminalを起動し、コマンドを実行する。
  - docker run –it ubuntu:latest /bin/echo 'Hello world‘

```
sarut@DESKTOP-A5ITNO6 MINGW64 ~
$ docker run -it ubuntu:latest /bin/echo 'Hello world'
Hello world
```

Hello World

- コマンドの説明

docker run –it ubuntu:latest /bin/echo ’Hello world ‘

- docker run ： コンテナ実行コマンド
- –it ： 対話的にコンテナとやり取りする
- ubuntu:latest ： Dockerイメージ
- /bin/echo ‘Hello World’ ： 実行コマンド

Hello World～まとめ～

Dockerを動かしてみるには

1. DockerToolboxのインストール
2. docker runコマンドの実行

を行うだけ。

Dockerイメージは、下記の順で検索される。

1. ローカル
2. DockerHub

コンテナを作って動かしてみる

内容・目的

- 複数種類のDockerイメージを作成し、同時実行する。
- イメージの作成はDockerfileを使用する。
- それぞれのコンテナに対してコマンドを実行し、バージョン差異（複数の環境）が1ホストOS上で成立していることを確認する。

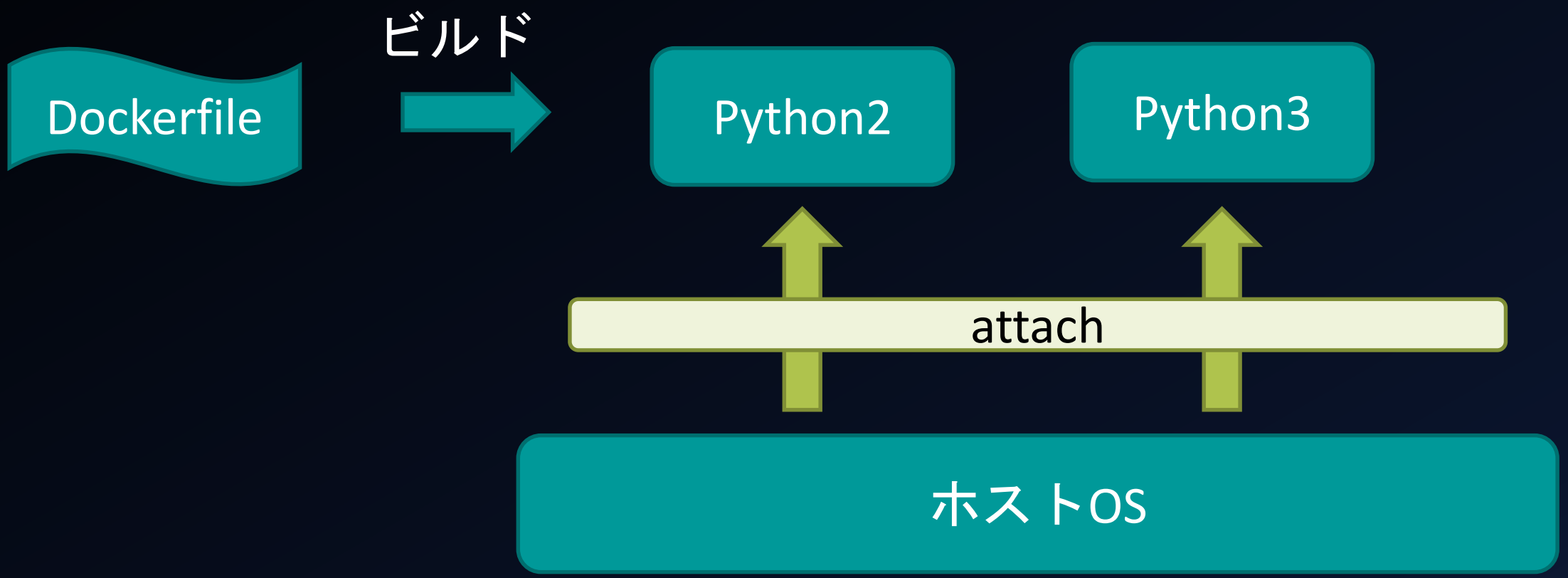
コンテナを作って動かしてみる

今回用意したDockerfile

- Python2.7
- Python3.5

上記2種類のコンテナを今回は利用して、それぞれのコンテナ上でPythonファイルを実行する。

コンテナを作って動かしてみる



コンテナを作って動かしてみる

- Dockerfile置き場
  - https://github.com/PUreatio/study_3/tree/master/Dockerfile

- Pythonファイル置き場
  - https://github.com/PUreatio/study_3/tree/master/Python

- イメージのビルド

＞docker build –t {作成するイメージのタグ} {Dockerfileパス}

コンテナを作って動かしてみる

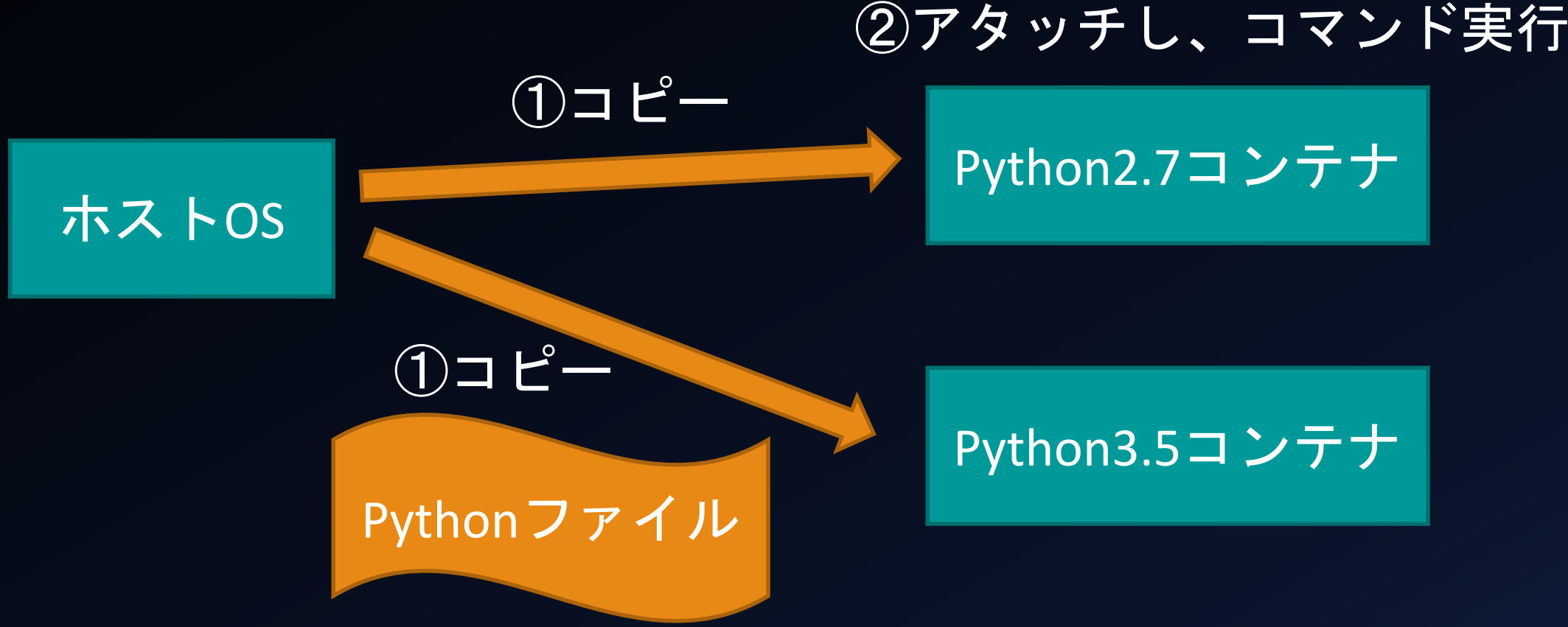
- 単純にPythonコマンドをそれぞれの環境で打ってみる
  - Python2.7 → python -V
  - Python3.5 → python3 -V

- 結果

ホストOS	Python2.7	Python3.5
• Python2.7→失敗 • Python3.5→失敗	• Python2.7→成功 • Python3.5→失敗	• Python2.7→失敗 • Python3.5→成功

コンテナを作って動かしてみる

今回コンテナを利用して行う操作



コンテナを作って動かしてみる

- ファイルコピー

ホストOS側のファイルをコンテナ側にコピーする場合

＞docker　cp　{コピー元パス}　{コピー先コンテナ}:{コピー先パス}

コンテナ側のファイルをホストOS側にコピーする場合

＞docker　cp　{コピー元コンテナ}:{コピー元パス}　{コピー先パス}

Dockerを作って動かしてみる

- コンテナへのアタッチ

＞docker attach {コンテナ名 or コンテナID}

- コンテナからのデタッチ

＞exit

＞Ctrl + P、Ctrl + Q

コンテナを作って動かしてみる

- Pythonファイルを実際に動かす
  - 2.7、3.5それぞれで動く内容が同じPythonファイルをそれぞれのコンテナで動かしてみる

- 結果

Python2.7	Python3.5
• Python2.7⇒動く • Python3.5⇒動かない	• Python2.7⇒動かない • Python3.5⇒動く

コンテナを作って動かしてみる～まとめ～

- 1つのホストOSから複数環境を管理できる。
- コンテナを実行してもホストOSに環境がインストールされているわけではない
- 各コンテナをそれぞれ独立した仮想環境のように扱える

ご清聴ありがとうございました