

Dockerとコンテナ超入門

免責事項

- 初心者向けの内容
- わかりやすさ重視
- 内容が一部不正確な可能性あり
- ~~—(マサカリ投げないで...)~~

おしながき

- Docker概要
- 仮想化振り返り
- コンテナとその技術的側面
- Docker実践
- Docker Compose

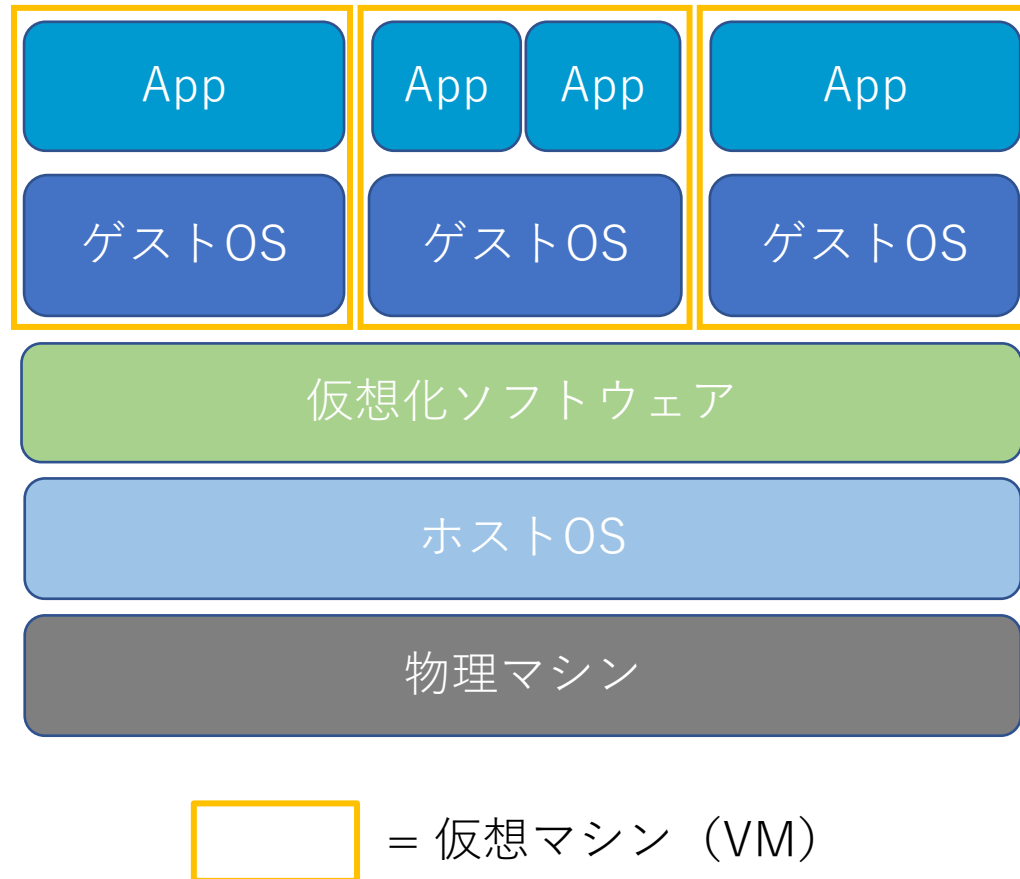
Docker is 何？

- Docker, Inc. (dotCloud, Inc.) が開発・提供するコンテナ型仮想化のプラットフォーム
- インフラ、ソフトウェア開発などで（今では当たり前）利用されている

仮想化とは？

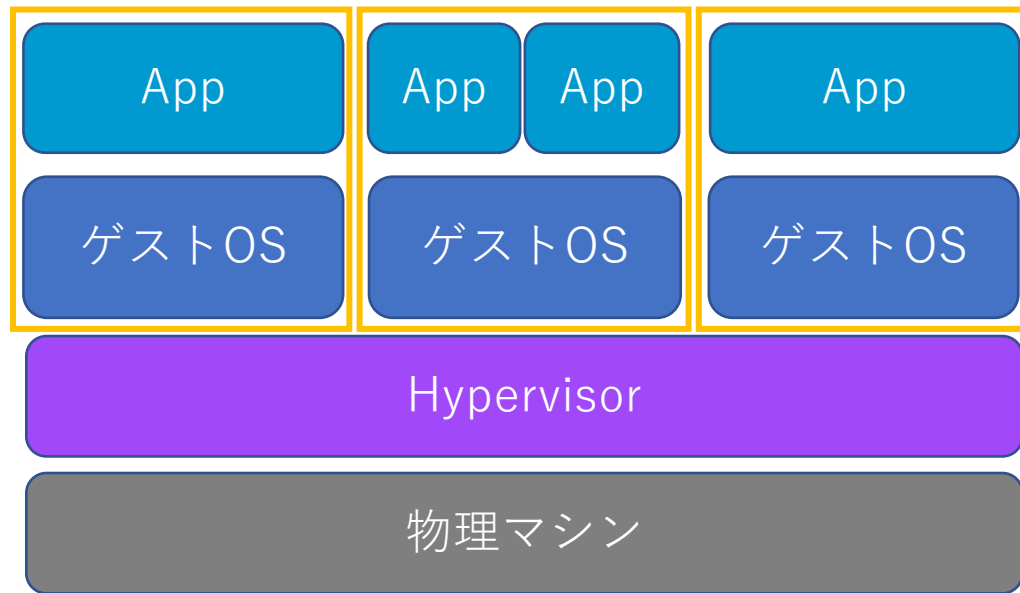
- ホスト型仮想化
- ハイパーバイザー型仮想化
- コンテナ型仮想化


ホスト型仮想化



- 仮想化ソフトウェアを入れるだけで動作
- ホストOSを動作させる分のリソースが必要
- VMの動作においてオーバーヘッド大
- VMware Workstation, Oracle VM VirtualBox, etc..

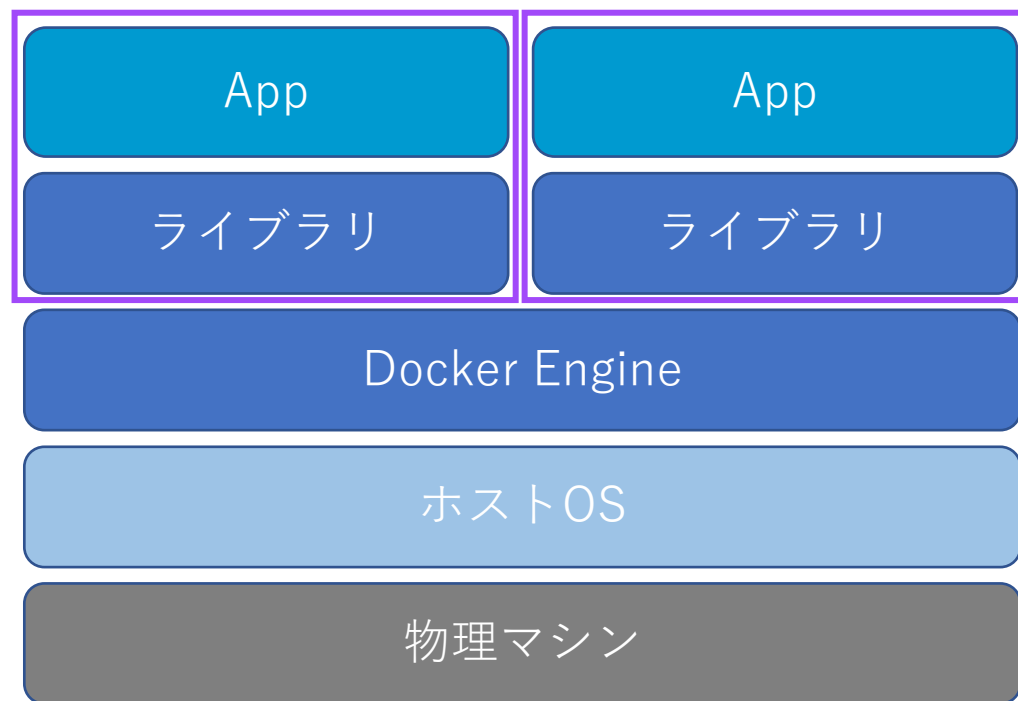
ハイパーバイザー型仮想化




 = 仮想マシン (VM)

- ハードウェアを直接制御
- ホスト型より高速
- VMware ESXi, KVM, Xen, etc..

コンテナ型仮想化 (Docker)



 = コンテナ

- ホストOSとコンテナはカーネルを共有
- オーバーヘッドは他二つより小さい
- 起動速い
- Docker, Kubernetes, etc...

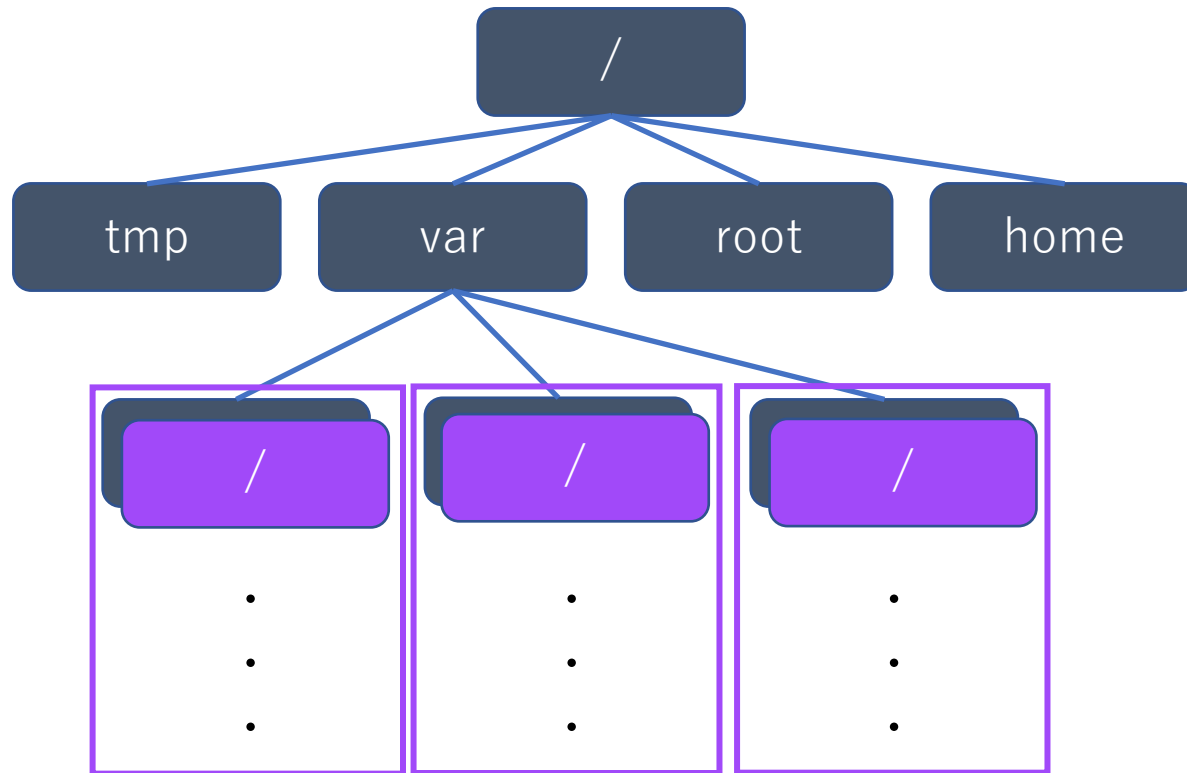
コンテナとその技術的側面

コンテナはホストOS上の隔離された領域

Linuxカーネルの機能を利用してうまい具合に作られている

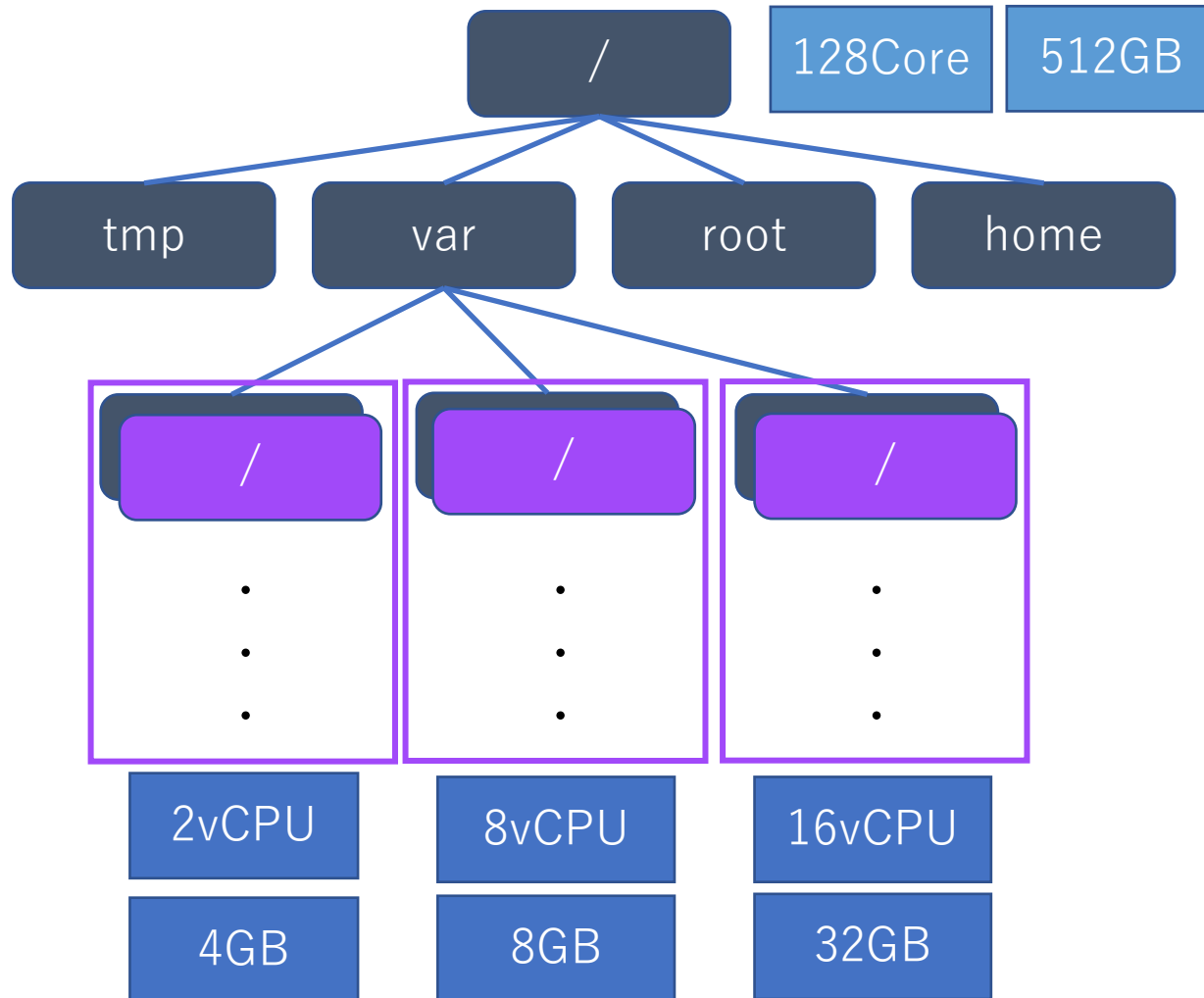
- chroot / pivot_root
- cgroups (control groups)
- namespace

chroot / pivot_root



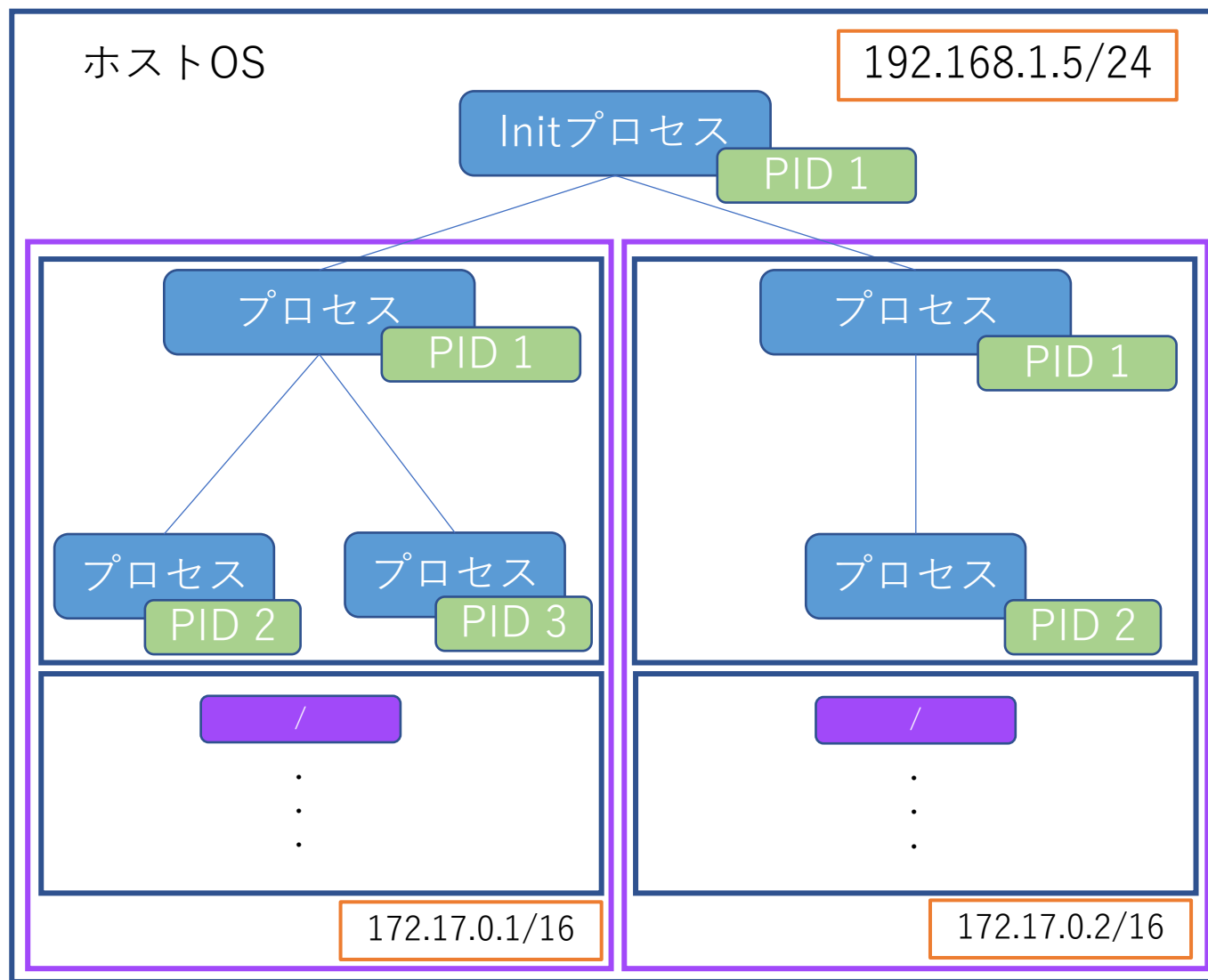
- 特定のディレクトリをルートディレクトリとして見立てて隔離する技術 (chroot監獄)
- プロセスを特定のディレクトリに隔離したり、プロセスのマウントポイントを入れ替えたり...

cgroups (control groups)




- プロセスグループでCPU, RAM, Disk I/Oを隔離・制限する機能

namespace



- プロセスとネットワークの名前空間を提供

 = namespace

Dockerの利点

- 起動速い

ホストOSとカーネルを共有するため実行速い

- コードで内容を定義

→可搬性高い, 環境の再現が容易, 共有も簡単

Create & Destroy!!

Infrastructure as Code (IaC) や ソフトウェア開発 に使うと楽しい&楽

例えば...

- Pythonで何か動かしたい...

ホストPCにPythonの実行環境をインストール



ソースコード用意



実行

手元のPCでは動くのに本番環境では
なぜか動かん

Pythonのバージョンは？
パッケージは？

例えば...

Pythonで何か動かしたい

>>環境構築つらい！！<<

入門

手元のPCでは動くのに本番環境では
なぜか動かん

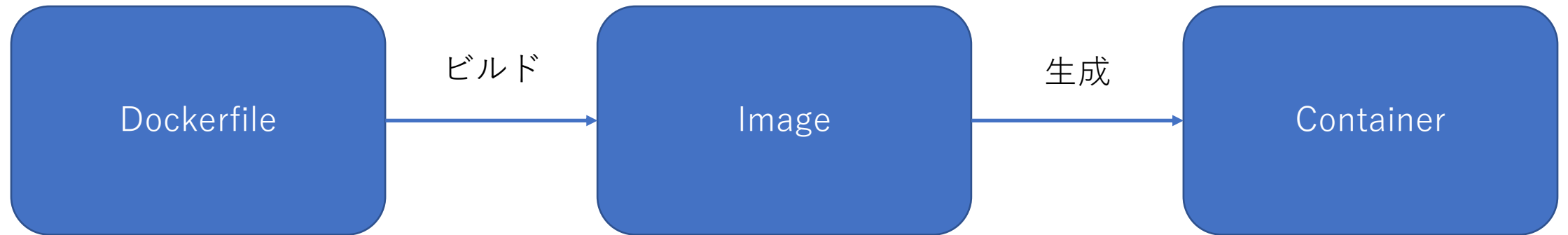
Pythonのバージョンは？
パッケージは？

Docker使おう！

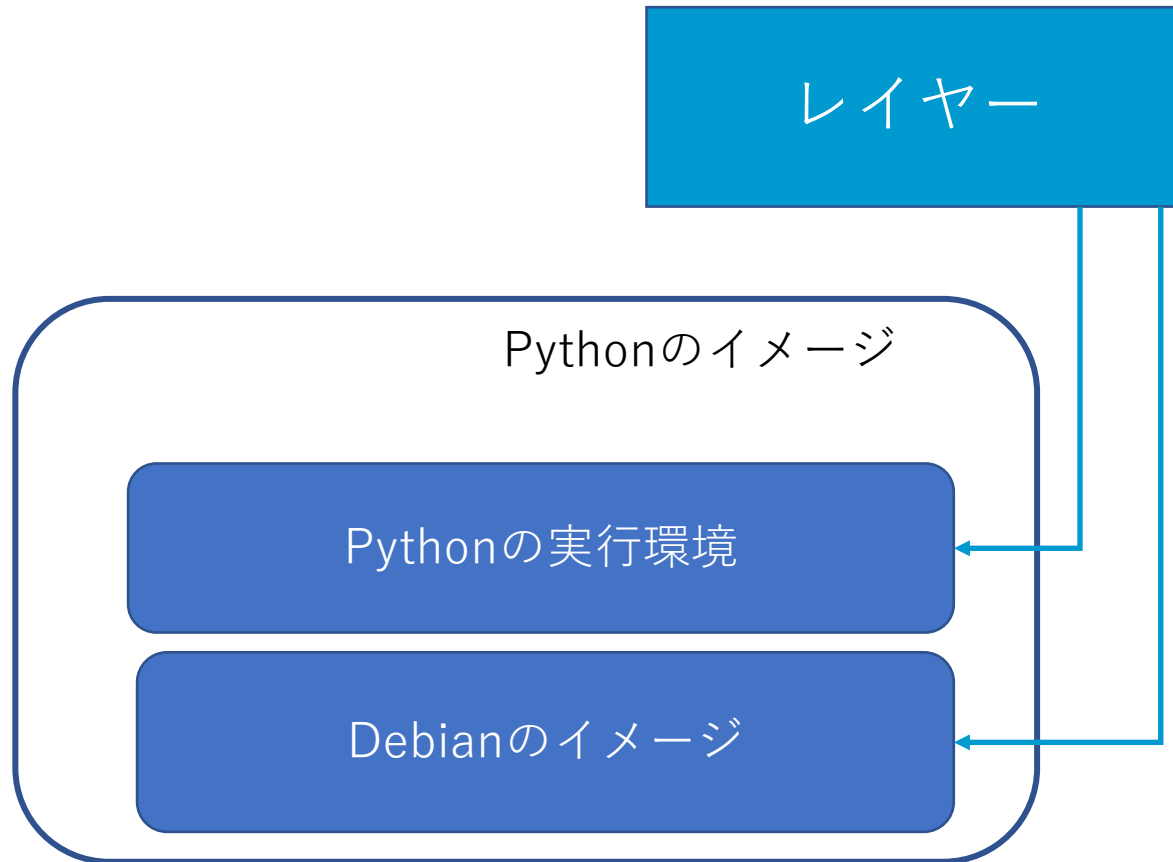
用意するもの

- Docker Desktop
→ 手順に従ってインストールする
- 教材
→ <https://github.com/quadseed/Docker-Tutorial>

Dockerfileとイメージ、そしてコンテナ



イメージ



- Docker Hubから拾ってくる or Dockerfile書いて自作
- Docker Hubでイメージを共有できる

Dockerfileの書き方の例

イメージのTagの名前

```
1 FROM python:3.9.6-slim-buster
2
3 RUN apt-get update
4 RUN apt-get -y install locales && \
5     localedef -f UTF-8 -i ja_JP ja_JP.UTF-8
6 ENV LANG ja_JP.UTF-8
7 ENV LANGUAGE ja_JP:ja
8 ENV LC_ALL ja_JP.UTF-8
9
10 COPY requirements.txt .
11 RUN pip install --upgrade pip
12 RUN pip install -r requirements.txt
```

- FROM: ベースにするイメージ
- RUN: 実行するコマンド
- ENV: 環境変数
- COPY: ディレクトリ内のファイルをイメージ内にコピー

イメージの選び方

TAG

latest

Log4Shell CVE not detected

Last pushed 15 hours ago by dojanky

docker pull python:latest

DIGEST	OS/ARCH	COMPRESSED SIZE
b08026f64969	linux/386	339.3 MB
fa1fc98f0f8c	windows/amd64	2.38 GB
fc57996b63b	windows/amd64	2.13 GB
+7 more...		

TAG

3.9.12-slim-bullseye

Log4Shell CVE not detected

Last pushed 10 hours ago by dojanky

docker pull python:3.9.12-slim-bulls...

DIGEST	OS/ARCH	COMPRESSED SIZE
9684973aea05	linux/386	45.68 MB
4277d13b9392	linux/amd64	45.02 MB
023d0ad9f52a	linux/arm/v5	42.28 MB
+5 more...		

TAG

3.9.12-slim

Log4Shell CVE not detected

Last pushed 10 hours ago by dojanky

docker pull python:3.9.12-slim

DIGEST	OS/ARCH	COMPRESSED SIZE
9684973aea05	linux/386	45.68 MB
4277d13b9392	linux/amd64	45.02 MB
023d0ad9f52a	linux/arm/v5	42.28 MB
+5 more...		

TAG

3.9-slim-bullseye

Log4Shell CVE not detected

Last pushed 10 hours ago by dojanky

docker pull python:3.9-slim-bullseye

DIGEST	OS/ARCH	COMPRESSED SIZE
9684973aea05	linux/386	45.68 MB
4277d13b9392	linux/amd64	45.02 MB
023d0ad9f52a	linux/arm/v5	42.28 MB
+5 more...		

- latest: その時々 の最新
- slim: 一部コンポーネント削減版
- bullseye: Debian v11ベース
- alpine: 軽量Linuxベース

パッケージのバージョンを固定する
ためにバージョン指定を推奨
(不具合防止&冪等性確保)

イメージをビルド

➤ `docker image build -t イメージ名[:tag名] Dockerfile`のパス

Ex) `docker image build -t [アカウント名/]test-image:0.1 .`

Docker hubに公開する場合はDocker Hubに登録したアカウント名が必須

公開しない場合はイメージ名単体でOK

データの永続化

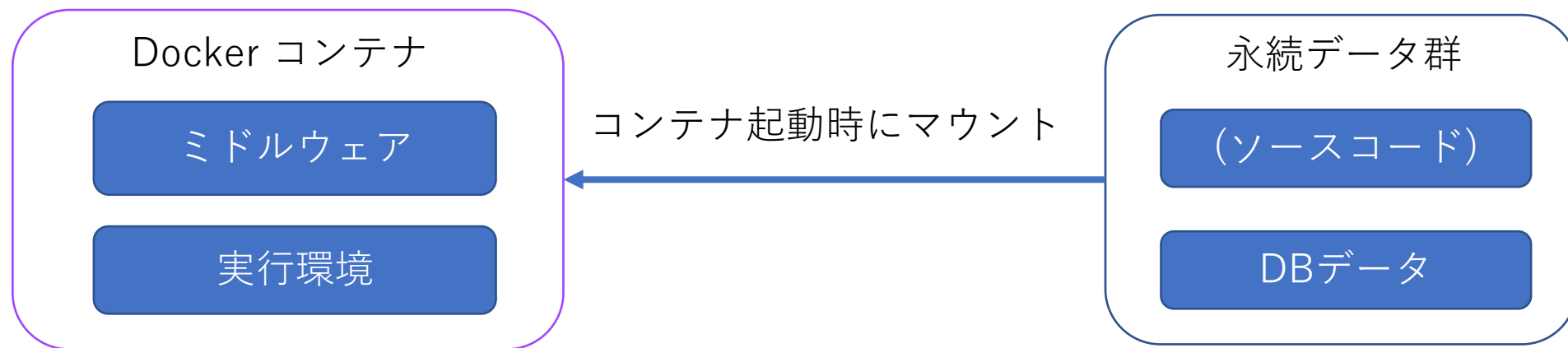
COPY でソースファイルをイメージに埋め込み

→ソースコード変更した場合、イメージ再生成 & コンテナ再構築が必要



ソースコードはコンテナにVolumeとしてマウントする（イメージにはライブラリ周りだけ用意）

データの永続化

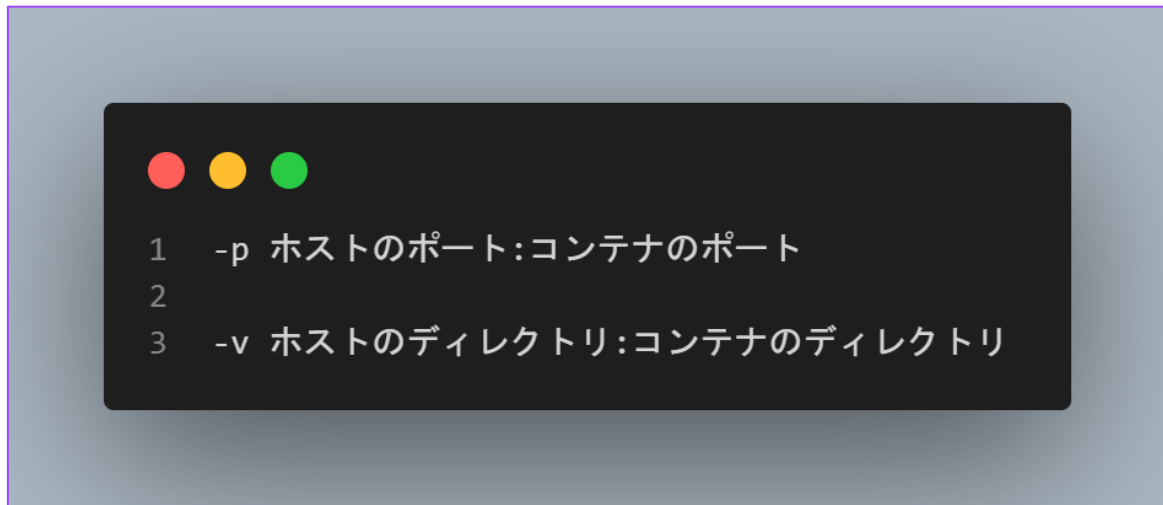


コマンドあれこれ

- `docker container run <イメージ名>`
→ コンテナ作成&起動

→ イメージのタグを指定した場合はtagも必須
→ `docker container create + docker container start`
- `docker container start <コンテナ名 or コンテナID>`
→ 既存コンテナの起動
- `docker container stop <コンテナ名 or コンテナID>`
→ 稼働中のコンテナの停止
- `docker ps`
→ 起動中のコンテナ一覧
- `docker image ls`
→ ローカルにあるイメージ一覧

引数あれこれ (docker container run)



-p ポートのバインド
-v ボリュームのマウント

-it コンテナ起動時に標準入出力と接続
-it = --interactive (-i) + --tty (-t)

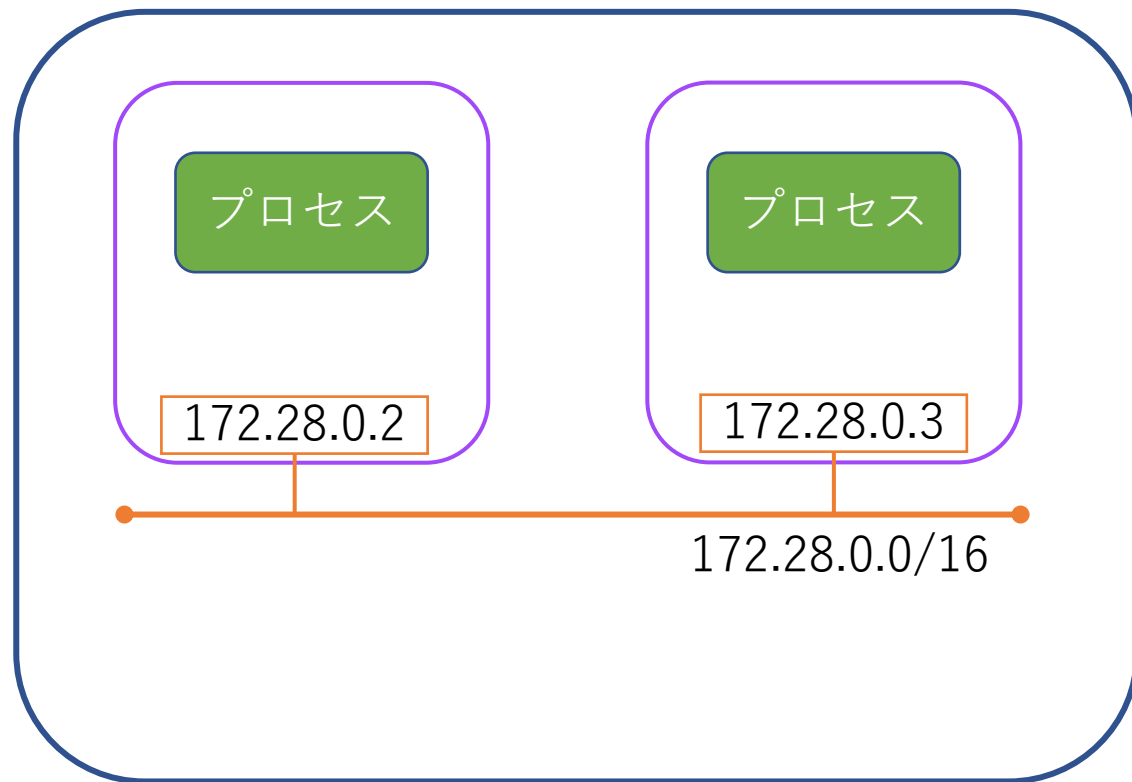
-d バックグラウンドで起動

稼働中のコンテナへの接続

- `docker attach <コンテナ名 or コンテナID>`
 - コンテナでシェルが動作している場合のみ
 - `exit` コマンドで抜けるとシェルが停止してコンテナも止まる
- `docker exec <コンテナ名 or コンテナID> /bin/bash`
 - 新たにシェルを起動する

Docker と Docker Compose

Docker Composeとは？



- 複数のコンテナを協調させることが可能
- コンテナの立ち上げ順序や環境変数、ポート設定やボリューム、コンテナネットワークの全てをyamlで定義できる

docker-compose.yml

```
1 version: "3"
2 services:
3   database:
4     image: mysql:5.7
5     volumes:
6       - ./db/mysql:/var/lib/mysql
7     restart: always
8     environment:
9       MYSQL_ROOT_PASSWORD: hogehege
10      MYSQL_DATABASE: database01
11      MYSQL_USER: user
12      MYSQL_PASSWORD: password
13
14   phpmyadmin:
15     image: phpmyadmin/phpmyadmin:latest
16     restart: always
17     ports:
18       - 8001:80
19     depends_on:
20       - database
21
22   wordpress:
23     image: wordpress:latest
24     volumes:
25       - ./wordpress/html:/var/www/html
26       - ./php/php.ini:/usr/local/etc/php/conf.d/php.ini
27     restart: always
28     ports:
29       - 8080:80
30     environment:
31       WORDPRESS_DB_HOST: database:3306
32       WORDPRESS_DB_NAME: database01
33       WORDPRESS_DB_USER: user
34       WORDPRESS_DB_PASSWORD: password
35
36     depends_on:
37       - database
```

- 各コンテナの設定をyaml形式で定義
- コンテナネットワークについても設定可能
- versionによって一部構文に差異がある

演習コマンドまとめ

- `docker image build -t python1 ./section1`
- `docker container run -it python1`

- `docker image build -t python2 ./section2`
- `docker container run -it python2`

- `docker image build -t python3 ./section3`
- `docker container run -v c:/Docker-Tutorial-main/section3/src:/project/src -it python3`

- `docker pull httpd`
- `docker container run -v c:/Docker-Tutorial-main/section4:/usr/local/apache2/htdocs -p 8080:80 -d httpd`
- `docker ps`

- `docker compose up -d`