

# Docker 勉強会 2回目

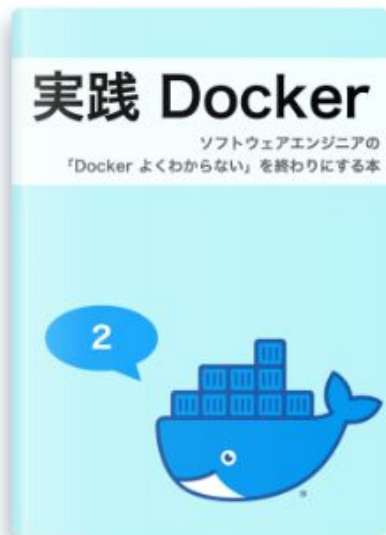
2024/2/7

# 事前準備

- Dockerを使える状態にしておく
  - [インストール方法\(Ubuntu20.04\)](#)
- (Docker Hubのアカウント作成)

# 参考書

実践 Docker – ソフトウェアエンジニアの「Docker よくわからない」を終わりにする本



## Chapters

- 1部
  - 導入
- 2部
  - 基礎
- 3部
  - 応用

## この資料について

1. 基本的には**参考書に準拠**して解説する
2. 取り扱わないトピックについては、~~このように表記する~~
3. (今回の目的において)**特に大事なところは**ページ全体を**赤枠で覆う**



# はじめに

- **目的**

- Dockerを普段から使うための簡単な理解をする

- **目標**

- 複数のROSバージョンによる煩わしさから解放される
- ROS2勉強会のための環境をDockerで構築する

## 結局 何がしたいのか(便利なのか)

- **ROSの複数のワークスペースやバージョンへの対応**
  - 開発環境を統一しやすい(人へ簡単に配布できる)
  - apt installしたものまでバックアップできる(依存関係)
    - apt install ros-<distro>-\* や ~msgsなど入れ直す必要がない
- **CUDA 複数のバージョンを同じホストで共存できる**
  - ドライバ, フレームワーク, ライブラリのバージョン管理が楽
  - ホストにCUDA環境を構築する必要がない

→簡単に壊せて, どのマシンでも同じ環境を容易に再現できる.

# 実習のための準備

**docker pullでイメージをダウンロード(2.08GB)**

```
$ docker pull masakifujiwara1/ros2:humble-orne-box3-sim
```

## 前回のおさらい

Ubuntu 22.04 のコンテナを起動してください

- オプションでターミナル操作可能
- それ以外のオプションは自由

```
$ docker ps [option]
```

```
$ docker stop ubuntu
```



## 前回のおさらい

イメージを指定するにはTAGを使う

```
$ docker run [option] <image> [command]
```

ubuntu:latest == ubuntu:22.04

REPOSITORY : TAG

```
$ docker run --rm ubuntu cat /etc/lsb-release  
$ docker run -it --rm ubuntu bash
```

# 2部

## 11: Dockerfileの基礎

- イメージをビルドする ( docker image build)
- ~~イメージのレイヤーを確認する ( docker image history)~~
- Dockerfileの必要性和有用性
- Dockerfileの基本の命令
- Dockerfileの作成
  - FROM: ベースイメージを指定する
  - RUN: 任意のコマンドを実行する
  - COPY: ホストマシンのファイルをイメージに追加する
  - CMD: デフォルト命令を指定する
  - 確認
- イメージをビルドする
- ~~Dockerfileを複数扱うには~~
- レイヤーを確認する
- ~~RUNをいくつかのレイヤーにするか~~
- ~~DockerHubのレイヤー情報を読み解く~~

## 2部 11: Dockerfileの基礎

### イメージをビルドする(docker image build)

```
$ docker build [option] <path>
```

オプション	意味	用途
<code>-f, --file</code>	Dockerfile を指定する	複数の Dockerfile を使い分ける
<code>-t, --tag</code>	ビルド結果にタグをつける	人間が把握しやすいようにする

## 2部 11: Dockerfileの基礎

### Dockerfileの必要性和有用性

これまでに...

- コンテナ内で行った操作は, コンテナ終了とともに無かったことになる
- イメージは, レイヤーという情報が積み重なったものである

しかし, 用意されているものでは不十分な場合がある！

例)ubuntu:22.04

- vimもgitもROSも入っていない

→ 自分専用のカスタマイズしたイメージが欲しい

## 2部 11: Dockerfileの基礎

### Dockerfileの基本の命令

命令	効果
FROM	ベースイメージを指定する
RUN	任意のコマンドを実行する
COPY	ホストマシンのファイルをイメージに追加する
CMD	デフォルト命令を指定する

例えば...

→ Ubuntu:22.04

→ apt-get install vim

→ hoge.txt(ホストpc) >> hoge.txt(Docker)

→ bash, date, cat /etc/lsb-release

Hands-on !

## 2部 11: Dockerfileの基礎

### Dockerfileの作成

```
$ mkdir docker-practice  
$ cd docker-practice  
$ touch Dockerfile
```

※Dockerfileは拡張子なし

# Hands-on !

## 2部 11: Dockerfileの基礎

FROM: ベースイメージを指定する

```
$ vim Dockerfile  
FROM ubuntu:22.04
```

RUN: 任意のコマンドを実行する (Linuxのコマンドを実行してその結果をレイヤーとして積み重ねる )

```
FROM ubuntu:22.04  
RUN apt update  
RUN apt-get install -y vim  
→ 1回保存 (:wq)
```

Hands-on !

## 2部 11: Dockerfileの基礎

COPY: ホストマシンのファイルをイメージに追加する

```
$ vim .vimrc (※docker-practice直下)  
set number  
:wq
```

```
$ vim Dockerfile  
COPY .vimrc /root/.vimrc (末尾に追加)
```



# Hands-on !

## 2部 11: Dockerfileの基礎

CMD: デフォルト命令を指定する

```
FROM ubuntu:22.04
```

```
RUN apt update
```

```
RUN apt-get install -y vim
```

```
COPY .vimrc /root/.vimrc
```

```
CMD date (末尾に追加, ※基本はbash推奨)
```

確認

```
$ tree -a .
```

## 2部 11: Dockerfileの基礎

### イメージをビルドする

```
$ docker image build [option] <path>
```

Hands-on !

```
$ docker image build --tag my-ubuntu:date .  
$ docker images  
$ docker run my-ubuntu:date (※Commandを省略)  
$ docker run -it --rm my-ubuntu:date bash (※Command == bash)  
root@~~~~~: /# vim /root/.vimrc
```

## 2部 11: Dockerfileの基礎

レイヤーを確認する(ローカルに存在するイメージのレイヤー情報を確認)

```
~/docker-paractice
>>> docker history ubuntu:22.04
```

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
fd1d8f58e8ae	12 days ago	/bin/sh -c #(nop) CMD ["/bin/bash"]	0B	
<missing>	12 days ago	/bin/sh -c #(nop) ADD file:99224b1f237763b30...	77.9MB	
<missing>	12 days ago	/bin/sh -c #(nop) LABEL org.opencontainers...	0B	
<missing>	12 days ago	/bin/sh -c #(nop) LABEL org.opencontainers...	0B	
<missing>	12 days ago	/bin/sh -c #(nop) ARG LAUNCHPAD_BUILD_ARCH	0B	
<missing>	12 days ago	/bin/sh -c #(nop) ARG RELEASE	0B	

```
~/docker-paractice
>>> docker history my-ubuntu:date
```

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
dd2258ac9c9d	9 minutes ago	/bin/sh -c #(nop) CMD ["/bin/sh" "-c" "date...	0B	
8dfb3af9682a	9 minutes ago	/bin/sh -c #(nop) COPY file:5744921fad1f5dd3...	11B	
fd1d8f58e8ae	12 days ago	/bin/sh -c #(nop) CMD ["/bin/bash"]	0B	
<missing>	12 days ago	/bin/sh -c #(nop) ADD file:99224b1f237763b30...	77.9MB	
<missing>	12 days ago	/bin/sh -c #(nop) LABEL org.opencontainers...	0B	
<missing>	12 days ago	/bin/sh -c #(nop) LABEL org.opencontainers...	0B	
<missing>	12 days ago	/bin/sh -c #(nop) ARG LAUNCHPAD_BUILD_ARCH	0B	
<missing>	12 days ago	/bin/sh -c #(nop) ARG RELEASE	0B	

Ubuntu:22.04と同じ！

## 2部

### まとめ(11: Dockerfileの基礎)

- **基本操作**

- FROM: ベースイメージを指定
- RUN: Linuxコマンドを実行
- COPY: ホストPCのファイルをイメージに追加
- CMD: デフォルト命令を指定

## 3部

- 13: イメージのビルド
- 14: コンテナの起動
- ~~● 15: ボリューム~~
- 16: バインドマウント
- ~~● 17: ポート~~
- ~~● 18: ネットワーク~~
- ~~● 19: Docker Compose~~
- ~~● 20: デバッグノウハウ~~

# 3部

## 13: イメージのビルド

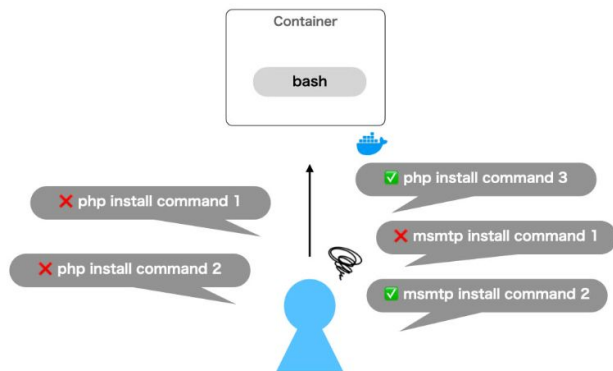
- Appコンテナについて
- DBコンテナについて
- Mailコンテナについて
- 構築
- DBイメージのビルド
- Mailイメージの選定
- 初めて構築するときは
- コンテナをイメージ化する (docker commit) ※参考書外

## 3部 13: イメージのビルド

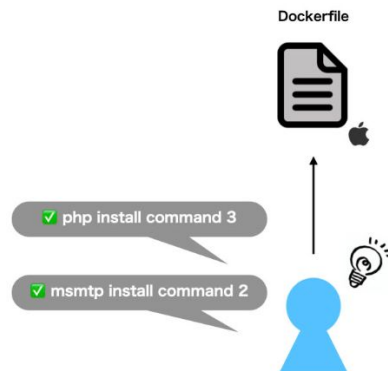
### 初めて構築するときは

先程まで: いきなりDockerfileを作成

→効率が悪い. 通常は, **bashで試行錯誤**してから!



bashでコマンドを確認

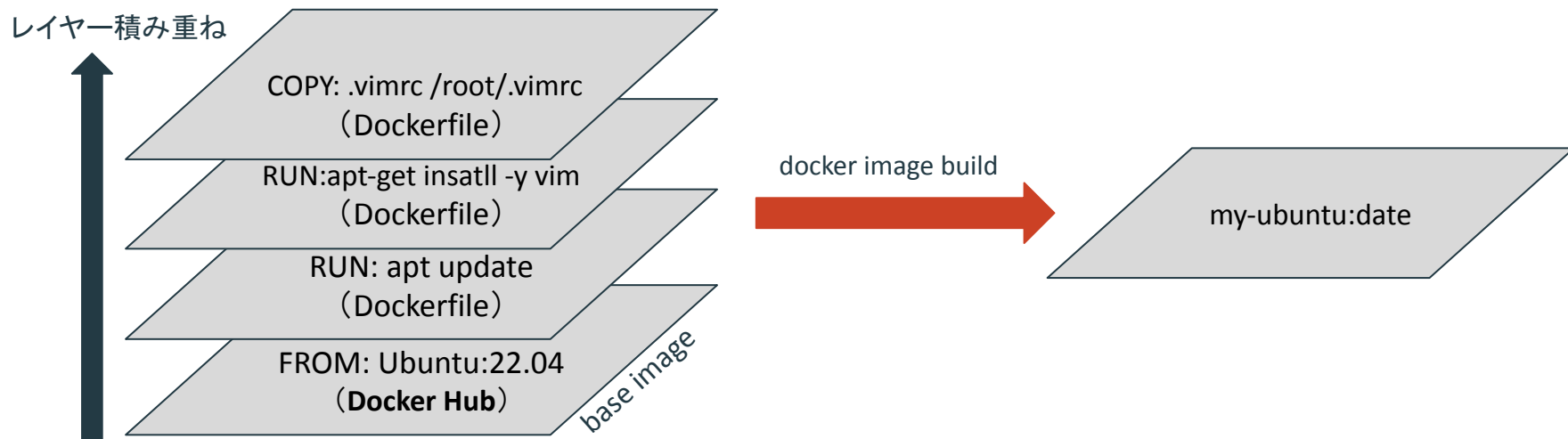


コマンドが通るかや手順が確定してからDockerfileを作成

## 3部 13: イメージのビルド

(bashで試行錯誤してからそのまま)コンテナをイメージ化する

今まで: ベースイメージにDockerfileでレイヤーを積み重ねてイメージを作成





## 3部 13: イメージのビルド

### コンテナをイメージ化する

```
$ docker commit <container ID or container name> <tag>
```

### Hands-on !

```
$ docker run -it --rm --name my-ubuntu1 my-ubuntu:date bash
root@~~~~~: /# echo "hoge" > /home/hoge.txt
$ docker commit my-ubuntu1 my-ubuntu:hoge
$ docker run my-ubuntu:hoge cat /home/hoge.txt
$ docker run my-ubuntu:date cat /home/hoge.txt
```

## 3部 13: イメージのビルド

### コンテナをイメージ化する

docker commitのメリット, デメリット

- メリット
  - バックアップに最適
  - Dockerfileをいじらなくていい
  - 時間がかからない
    - 例) インストールやビルドをしなくて済む
- デメリット
  - マルチアーキテクチャに対応できなくなる
  - 何を変更したのか分からなくなる
    - 例) 説明書無しで作成した→ 全体像が分からないため, 変更が困難

# 3部

## 14: コンテナの起動

- 構築
- DBコンテナの起動
- Mailコンテナの起動
- MySQLデータベースの接続経路について
- コンテナが起動しないときは

## 3部 14: コンテナの起動

### コンテナが起動しないときは

コンテナが起動しない理由は大きく2つ

1. Dockerの状態や操作に不備がある
  - 起動コマンドの文法が間違っている(操作ミス)
  - コンテナ名の衝突(頻繁に起こり得る)

→ 大抵, docker run の出力を読めば分かる!

~~2. イメージや命令に不備がある~~

# 3部

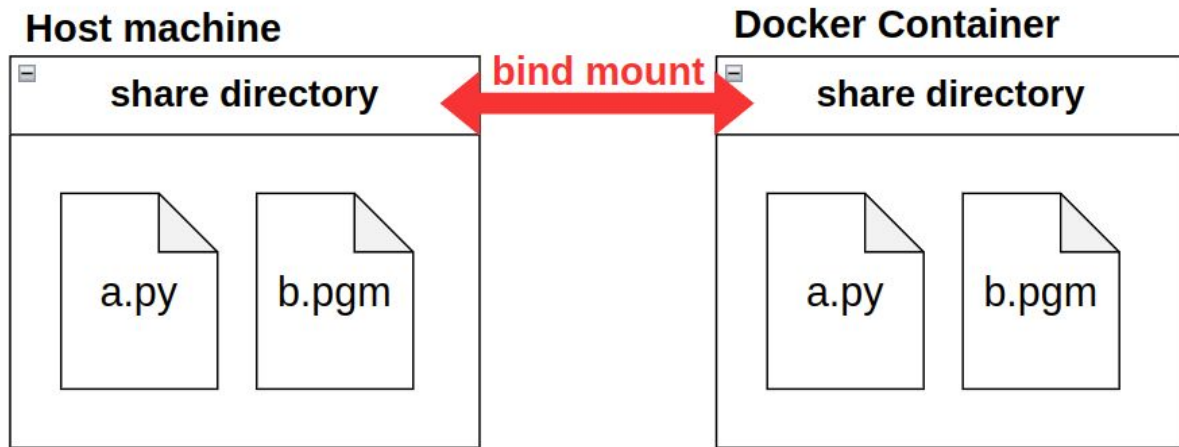
## 16: バインドマウント

- バインドマウントとは
- ~~構築~~
- ~~Appコンテナにソースコードをマウントする~~
- ~~DBコンテナに初期化クエリをマウントする~~
- バインドマウントの実体と注意
- ~~COPYとバインドマウントの使い分け~~

## 3部 16: バインドマウント

### バインドマウントとは

ホストPCの任意のディレクトリをコンテナにマウントする仕組み



# Hands-on !

## 3部 16: バインドマウント

バインドマウントとは

```
$ mkdir hoge && date > hoge/hoge.yaml  
$ docker run -it --rm -v ./hoge:/home/hoge my-ubuntu:hoge bash  
root@~~~~~: /# vi /home/hoge/hoge.yaml
```

## 3部 16: バインドマウント

### バインドマウントの実体と注意

バインドマウントはホストマシンのファイルシステム

つまり、管理をしているのは**ホストマシン**

ということは...

削除したものにバインドマウントしたディレクトリが含まれていると、

削除はホストマシンにまで波及する.



# 実際の運用例

- GitHubでの管理
- DockerHubでの管理
- DockerHubへpush

# 実際の運用例

## GitHubでの管理

### 管理するもの

- Dockerfile
- COPY元のファイル
- シェルスクリプト

The screenshot shows the GitHub search interface with the search term 'docker'. The results are sorted by 'last updated' and show 7 repositories. The first five repositories are listed with their names, public status, shell icon, star count, and update date. Each repository has a green line graph showing activity over time.

Repository Name	Public	Shell	Stars	Updated
ros1_docker	Public	●	1	Updated 2 weeks ago
cuda_docker	Public	●		Updated 3 weeks ago
ros2_docker	Public	●		Updated on Dec 21, 2023
3d_mapping_docker	Public	●	1	Updated on Oct 27, 2023
tsukuba2023_humble_docker	Public	●		Updated on Sep 4, 2023

# Hands-on !

## 実際の運用例

### GitHubでの管理

```
$ git clone https://github.com/masakifujiwara1/ros2_docker.git
```

```
— Dockerfile
— README.md
— build.sh
— config
  — .bashrc
  — .tmux.conf
  — .vimrc
  — requirements.txt
— docker_share
  — hoge.txt
— login.sh
— run.sh
— run_gpu.sh
```

} コピー元のファイル  
} バインドマウント

# 実際の運用例

## GitHubでの管理

run\_gpu.sh

```
#!/bin/bash
eval "docker container run --network host --gpus all -it --name my-humble -e
DISPLAY=$DISPLAY --volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" -v
$PWD/docker_share:/home/host_files --privileged -v /dev:/dev --env="XAUTHORITY=$XAUTH"
-v "$XAUTH:$XAUTH" --env="QT_X11_NO_MITSHM=1" --ipc=host
masakifujiwara1/ros2:humble-orne-box3-sim"
```

ネットワーク設定, ディスプレイ設定, GPU設定, デバイス設定...  
→ シェルスクリプトに押し込める

# 実際の運用例

## GitHubでの管理

### login.sh

```
#!/bin/bash  
  
eval "docker container exec -it my-humble-orne-box3-sim bash"
```

### build.sh

```
#!/bin/bash  
  
eval "docker image build -t masakifujiwaral/ros2:my-humble-orne-box3-sim ."
```

# 実際の運用例

## GitHubでの管理

**ros2\_docker** Public

● Shell Updated 11 minutes ago

**ros1\_docker** Public

● Shell ☆ 1 Updated 2 weeks ago

**cuda\_docker** Public

● Shell Updated 3 weeks ago

**3d\_mapping\_docker** Public

● Shell 1 📄 BSD 3-Clause "New" or "Revised" License Updated on Oct 27, 2023

**tsukuba2023\_humble\_docker** Public

● Shell Updated on Sep 4, 2023

### ros2関連を管理

- humble
- foxy
- cuda117-pytorch2.0-humble
- tokuron
- humble-orne-box3-sim

### map作成用のパッケージを管理


- noetic
  - cartographer
- humble
  - cartographer
  - LIO-SAM
  - li\_slam\_ros2

# 実際の運用例

## DockerHubでの管理

REPOSITORY:tag

で分かりやすく管理




**masakifujiiwara1**  
Community User · Joined August 22, 2023

Repositories

Starred

Displaying 1 to 7 repositories




**masakifujiiwara1/office-robot**

By [masakifujiiwara1](#) · Updated 14 days ago

Image

2 · 0




**masakifujiiwara1/ros1**

By [masakifujiiwara1](#) · Updated 16 days ago

Image

7 · 0



**masakifujiiwara1/ros1** ☆

By [masakifujiiwara1](#) · Updated 16 days ago

Image

Pulls

Overview

Tags

Sort by

Newest

Filter Tags

TAG

[noetic-box-setup](#)

Last pushed 16 days ago by [masakifujiiwara1](#)

DIGEST

[cc6c3259efca](#)

OS/ARCH

linux/amd64

COMPRESSION SIZE

1.12 GB

docker pull masakifujiiwara1/ros1-

TAG

[noetic](#)

Last pushed 16 days ago by [masakifujiiwara1](#)

DIGEST

[e03c3734aace](#)

OS/ARCH

linux/amd64

COMPRESSION SIZE

1.07 GB

docker pull masakifujiiwara1/ros1-

# 実際の運用例

## DockerHubへpush

```
$ docker login
```

```
$ docker push my-ubuntu:hoge
```



## まとめ

1. DockerHubでimageを探してくる
2. imageをカスタム
3. docker runでコンテナ起動
4. 開発!!