

WLOラズパイ倶楽部 第18回

# Dockerで活用するRaspberry Pi



2019年3月26日

過去17回の  
「WLOラズパイ倶楽部」の  
資料、スクリプトが  
格納されています。

今後のイベントでも  
ここに資料を格納するので  
予習・復習にご活用ください。

<https://github.com/WLO-RaspiClub/>

<https://git.io/wlopi>

The screenshot shows the GitHub repository page for WLO-RaspiClub. The repository name is 'WLO-RaspiClub' and it is described as 'WLOで隔週開催される「WLOラズパイ倶楽部」の情報共有チームです。'. The repository has 18 repositories, 5 people, 0 teams, and 0 projects. The page lists three repositories:

- 20190110\_WinterHolidayLT**: 2019年1月10日に開催予定の「WLOラズパイ倶楽部 冬休み作品発表会 2019」のリポジトリ. Updated 3 days ago.
- 20181108\_NewRaspbianInstallLive**: 2018年11月8日に開催予定の「WLOラズパイ倶楽部 最新Raspbianインストールライブ」のリポジトリ. Updated on 8 Nov 2018.
- 20180712\_HeadlessPi**: 2018年7月12日に開催予定の「WLOラズパイ倶楽部 ヘッドレスで活用 Raspberry Pi」のリポジトリ. Updated on 12 Jul 2018.

On the right side, there are sections for 'Top languages' (Shell, Python) and 'People' (5 members). There is also an 'Invite someone' button.

<http://git.io/wlopi>



- ・ 前回(2019/1/10)以降の新情報
  - ・ 3/4に6周年、記念イベントが各地で開催

<https://www.raspberrypi.org/blog/raspberry-jam-big-birthday-weekend-2018/>



- ・ Raspberry Pi SHOP: 直営店オープン(2/7)



イギリスのケンブリッジに  
財団直営ショップがオープン。  
マグカップやキーボード・マウスなど  
限定グッズも販売している。

<https://www.raspberrypi.org/raspberry-pi-store/>

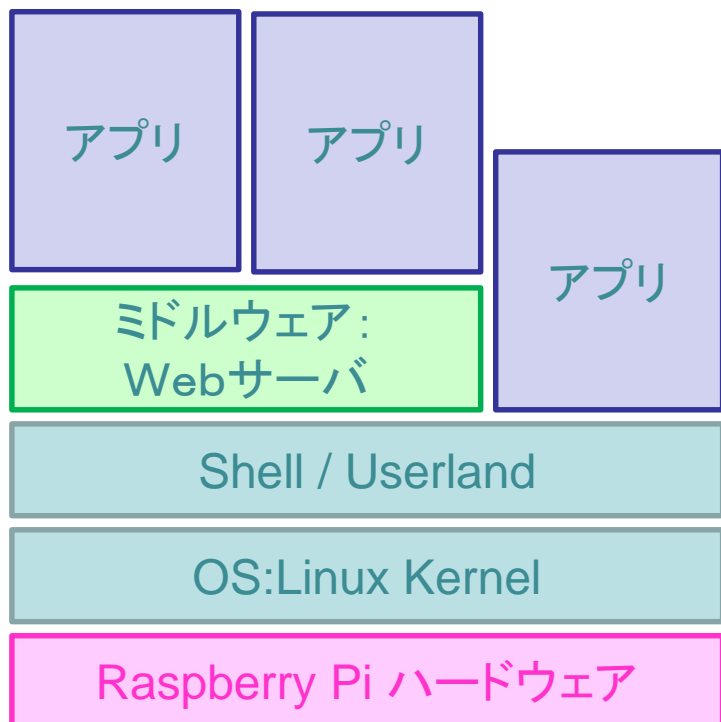
<http://git.io/wlopi>

- ・ Dockerの概要を理解する
- ・ Raspberry PiでDocker環境を構築する
- ・ 複数のコンテナを作成・実行・削除してみる

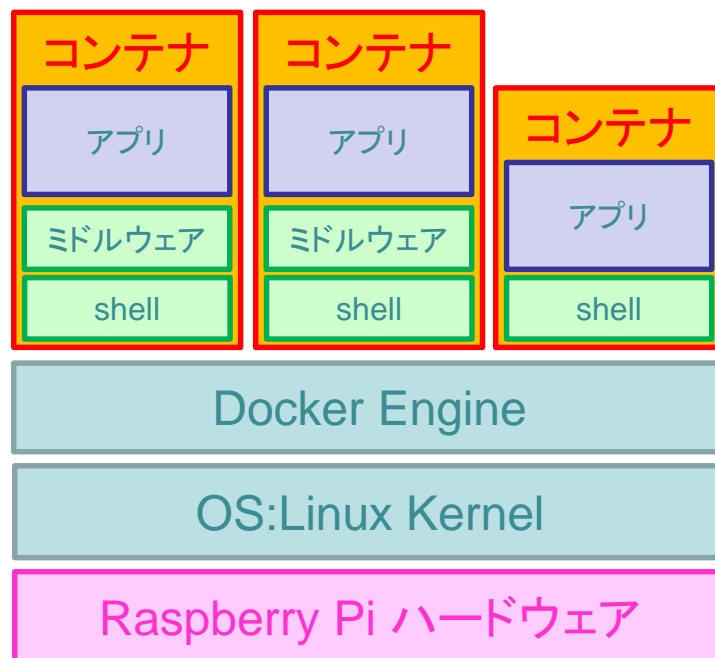
- ・ひとことで言うと：コンテナ型のアプリケーション実行環境

OSとアプリがインストールされた環境をそのまま「コンテナ」に閉じ込めて、いつでもその環境を再現することができます。

- ・ Dockerを使わないとき



- ・ Dockerを使うとき



- ・ **よく使われるアプリの導入手順をコード化して再現**

ベースとなるコンテナイメージに対してDockerfileという手順を適用することで目指すアプリが稼働する環境を手早く構築することができる。

- ・ **起動時間が早い**

構築済みイメージからアプリの起動が早い

- ・ **1つのホストで複数のアプリを起動できる**

1台のホストに複数のコンテナを動作させることができる

- ・ **一度構築すると再利用できる**

構築済みイメージを取っておくことで止めたり入れ替えたりできる

- ・ **Docker HUB で公開されたイメージ/Dockerfileを使える**

公開されたイメージやDockerfileで簡単に環境構築

## ・構築済みイメージの活用

複数のソフトウェアパッケージからなる大きなアプリケーションを導入する場合、CPUやファイルI/Oが遅いRaspberry Piでは構築に時間がかかることがある。

→構築済みDockerイメージを使うと、ダウンロードして実行するだけで使い始められる

例：ownCloud (パーソナルクラウドサーバ) Redmine (開発用BTS)

各種開発ツールなど

## ・1台のRaspberry Piを様々な用途に活用する

Dockerを使わない場合、用途別にmicroSDを複数使うなどすることで実現できるが毎回OSから導入する必要があり時間がかかる

→Dockerイメージを保持しておくことで構築済み環境をすぐ再現できる

## ・複数のRaspberry Piで同じ用途のサーバ構築する

構築済みDockerイメージをつかって、ネットワーク上のRaspberry Piに同じ環境をつくれる

## ・ゲストOSとして様々なOSを試す ※Linuxに限る

Raspbianの上でUbuntuやcentos(のuserland)を試すことができる

## ・まず、Docker Engine、関連ツールを導入する

導入スクリプトによる導入：公式Blogで紹介されている導入方法

<https://www.raspberrypi.org/blog/docker-comes-to-raspberry-pi/>

```
# インストール.  
$ curl -sSL https://get.docker.com/ | sh  
  
# バージョン確認  
$ sudo docker -v  
  
# root ユーザ以外で docker を使用できるように  
# docker グループに pi ユーザを追加.  
$ sudo usermod -aG docker pi  
  
# 自動起動設定.  
$ sudo systemctl enable docker  
  
# 一度再起動する  
$ sudo reboot  
  
# piユーザでツールが動作するか確認  
$ docker -v
```



## • hello world

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cab9c9fde470971e499788
Status: Downloaded newer image for hello-world:latest
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

<https://hub.docker.com/>

For more examples and ideas, visit:

<https://docs.docker.com/get-started/>

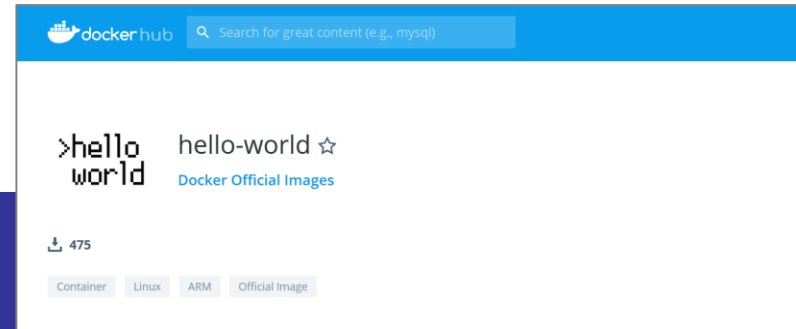
## ・ docker run hello-worldはなにをやっているのか

[https://hub.docker.com/\\_/hello-world](https://hub.docker.com/_/hello-world)

### 1: docker hubからイメージを取得(docker pull)

公式リポジトリであるdocker hubから  
イメージを取得する

```
pi@raspberrypi:~ $ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
c1eda109e4da: Pull complete
Digest: sha256:2557e3c07ed1e38f26e389462d03ed943586f744621577a99efb77324b0fe535
Status: Downloaded newer image for hello-world:latest
```



```
pi@raspberrypi:~ $ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	618e43431df9	2 months ago	1.64kB

### 2:取得したイメージからコンテナを作成(docker create)

```
pi@raspberrypi:~ $ docker create 618e43431df9
02dea1477005e37b109960b927b504212d00dcd65f5d2738ef42eab0e860ff11
```

```
pi@raspberrypi:~ $ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
02dea1477005	618e43431df9	"/hello"	17 seconds ago	Created		jovial_pike

### 3:イメージを読み込んで実行(docker start)

```
pi@raspberrypi:~ $ docker start -i jovial_pike

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

## • Docker イメージ

OS/ミドルウェア/アプリを導入済みの「コンテナの元」

作成方法は3種類

- Docker Hubからダウンロードして入手  
(実行対象のホストOS/CPUアーキテクチャに合わせたイメージ: Raspberry PiはLinux/ARM)
- 構築済みコンテナからイメージを生成
- Baseイメージから、Dockerfileをつかってカスタマイズして生成

一度生成されたイメージは編集できない。

→イメージを基にしてさらにミドルウェアやアプリを追加して新しいイメージを生成できる

## • Docker コンテナ

イメージから生成したアプリの実行環境

- イメージから生成 (docker create)
- Docker Engineで起動 (docker run)
- 一時停止/再開/再起動
- 停止/削除

コンテナ生成時に名前を付けることができる (ホスト内でユニーク・つけないと自動生成)

ホストOSの上で複数実行できる

ホストOSのストレージをマウントできる (マウントしていないとコンテナ削除時に消える)

ホストOSのネットワークポートを接続できる

## ・ Raspberry Piで実行できるイメージの検索

docker hubでLinux/ARMにチェックを入れて検索

<https://hub.docker.com/>

The screenshot shows the Docker Hub search results page. The top navigation bar includes the Docker Hub logo, a search bar, and an 'Explore' button. Below the navigation bar, there are tabs for 'Docker EE', 'Docker CE', 'Containers', and 'Plugins'. The main content area displays search results for images, filtered by 'Linux' and 'ARM' architecture. The results list includes 'redis', 'ubuntu', and 'busybox'. Each result shows the image name, update time, description, and a list of supported architectures. The 'redis' image is updated 29 minutes ago and supports architectures: Container, Linux, Windows, PowerPC 64 LE, 386, ARM 64, ARM, x86-64, IBM Z, and D. The 'ubuntu' image is also updated 29 minutes ago and supports architectures: Container, Linux, 386, ARM 64, ARM, x86-64, IBM Z, PowerPC 64 LE, and Base Images. The 'busybox' image is updated 29 minutes ago and supports architectures: Container, Linux, 386, ARM 64, ARM, x86-64, IBM Z, PowerPC 64 LE, and Base Images. On the right side, there are filters for 'Operating Systems' (Linux, Windows) and 'Architectures' (ARM, ARM 64, IBM POWER, IBM Z, PowerPC 64 LE, x86, x86-64).

Operating Systems

- ☒ Linux
- ☐ Windows

Architectures

- ☒ ARM
- ☐ ARM 64
- ☐ IBM POWER
- ☐ IBM Z
- ☐ PowerPC 64 LE
- ☐ x86
- ☐ x86-64

## ・ 軽量OS alpine を導入してみる

docker hubでLinux/ARMにチェックを入れて検索

<https://hub.docker.com/>

The screenshot shows the Docker Hub search results for the 'alpine' image. The left sidebar contains filters for 'Filters (3)', 'Docker Certified', 'Images', 'Categories', 'Operating Systems', and 'Architectures'. The 'Operating Systems' filter is checked for 'Linux', and the 'Architectures' filter is checked for 'ARM'. The search results show 1 - 7 of 7 available images. The 'alpine' image is highlighted, showing it is an 'OFFICIAL IMAGE' with over 10M downloads and 5.1K stars. The description states: 'A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!'. The image also shows the 'alpine' logo and the text 'Docker Official Images'. Below the description, there are tags for 'Container', 'Linux', '386', 'ARM 64', 'ARM', 'x86-64', 'IBM Z', 'PowerPC 64 LE', 'Featured Images', and 'Base Images'. The 'alpine' image is also shown in a larger preview window, displaying the same information and tags.

Filters (3) [Clear All](#)

Docker Certified [?](#)

☐ ☒ Docker Certified

Images

☐ Verified Publisher [?](#)  
Docker Certified And Verified Publisher Content

☐ Official Images [?](#)  
Official Images Published By Docker

Categories [?](#)

☐ Analytics

☐ Application Frameworks

☐ Application Infrastructure

☐ Application Services

☐ Base Images

☐ Databases

☐ DevOps Tools

☐ Featured Images

☐ Messaging Services

☐ Monitoring

☒ Operating Systems

☐ Programming Languages

☐ Security

☐ Storage

Operating Systems

☒ Linux

☐ Windows

Architectures

☒ ARM

1 - 7 of 7 available images.

Most Popular

**ubuntu**

Updated 2 minutes ago

Ubuntu is a Debian-based Linux operating system based on free software.

Container Linux 386 ARM 64 ARM x86-64 IBM Z PowerPC 64 LE Base Images Operating Systems

**alpine**

Updated 2 minutes ago

A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!

Container Linux PowerPC 64 LE 386 ARM 64 ARM x86-64 IBM Z Featured Images Base Images

**alpine** ☆

**Docker Official Images**

A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!

10M+

Container Linux 386 ARM 64 ARM x86-64 IBM Z PowerPC 64 LE Featured Images Base Images

Operating Systems Official Image

**fedora**

Updated a few seconds ago

10M+ 777 Downloads Stars

<http://git.io/wlopi>

## ・ 導入手順

alpineではデフォルトシェルはbashではなくash

```
pi@raspberrypi:~ $ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
d8d287cbc574: Pull complete
Digest: sha256:644fcb1a676b5165371437feaa922943aaf7afcfa8bfee4472f6860aad1ef2a0
Status: Downloaded newer image for alpine:latest
pi@raspberrypi:~ $ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alpine	latest	ef872b76f64d	2 weeks ago	3.7MB
debian	latest	101cc557c042	2 weeks ago	86.2MB
hello-world	latest	618e43431df9	2 months ago	1.64kB
raspbian/stretch	latest	82ca4d7b3224	11 months ago	165MB

```
pi@raspberrypi:~ $ docker create -it --name alpine ef872b76f64d ash
063dc545faecf8c112118efb0b38b296ce2bec84b37f69b2e6a2885134bf091b
pi@raspberrypi:~ $ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
063dc545faec	ef872b76f64d	"ash"	9 seconds ago	Created
02dea1477005	618e43431df9	"/hello"	2 hours ago	Exited (0)

```
pi@raspberrypi:~ $ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
--------------	-------	---------	---------	--------

## ・ 起動確認

alpineではデフォルトシェルはbashではなくash

```
pi@raspberrypi:~ $ docker start alpine
alpine
pi@raspberrypi:~ $ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
063dc545faec        ef872b76f64d       "ash"              29 seconds ago     Up 3 seconds
c68b5e690a97        82ca4d7b3224       "/bin/sh -c 'tail -f..." 2 hours ago        Up 2 hours
pi@raspberrypi:~ $ docker exec -it alpine ash
/ # cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.9.2
PRETTY_NAME="Alpine Linux v3.9"
HOME_URL="https://alpinelinux.org/"
BUG_REPORT_URL="https://bugs.alpinelinux.org/"
/ # exit
pi@raspberrypi:~ $ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 9 (stretch)"
NAME="Raspbian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
```

- 導入済みイメージの表示: `docker images / docker image ls`
- 導入済みコンテナ一覧: `docker ps -a / docker container ls -a`
- 実行中コンテナ一覧: `docker ps / docker container ls`
  
- Docker Hub からイメージを導入: `docker pull イメージ名`
- イメージからコンテナ作成: `docker create -it --name コンテナ名 イメージID シェルコマンド`
- コンテナ実行: `docker start コンテナ名`
  
- 動作中コンテナに入る `docker exec -it コンテナ名 シェルコマンド`
  
- コンテナを停止 `docker stop コンテナ名`
- コンテナを削除 `docker rm コンテナ名`
- イメージを削除 `docker image rm イメージ名 / docker rmi イメージ名`

参考 : 「docker container / image コマンド新旧比較」

<https://qiita.com/zembutsu/items/6e1ad18f0d548ce6c266>

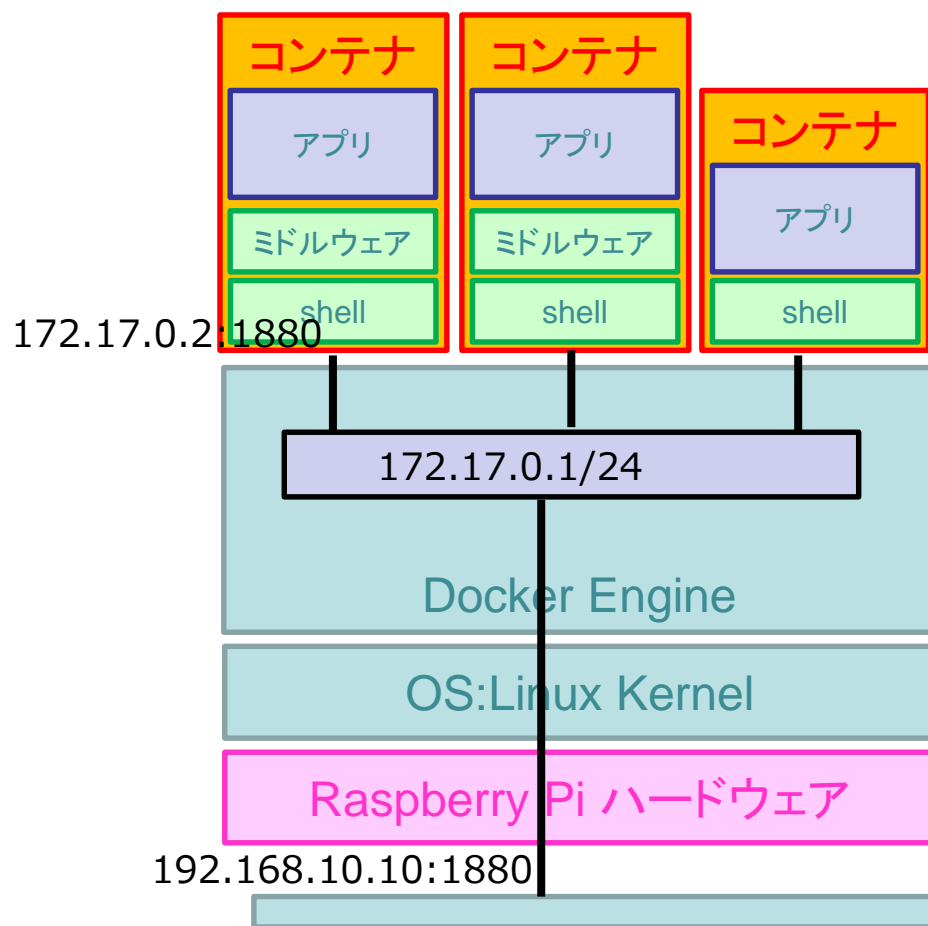


## ・コンテナとホストOSをつなぐ仮想ネットワーク

Docker Engine導入時に

Bridgeネットワーク、hostネットワーク、noneネットワーク

3種類が自動的に生成



Bridgedネットワークの場合、  
起動時に172.17.0.1/24の仮想ネットワークが  
生成され、コンテナに172.17.0.xのIPアドレス  
が割り振られる

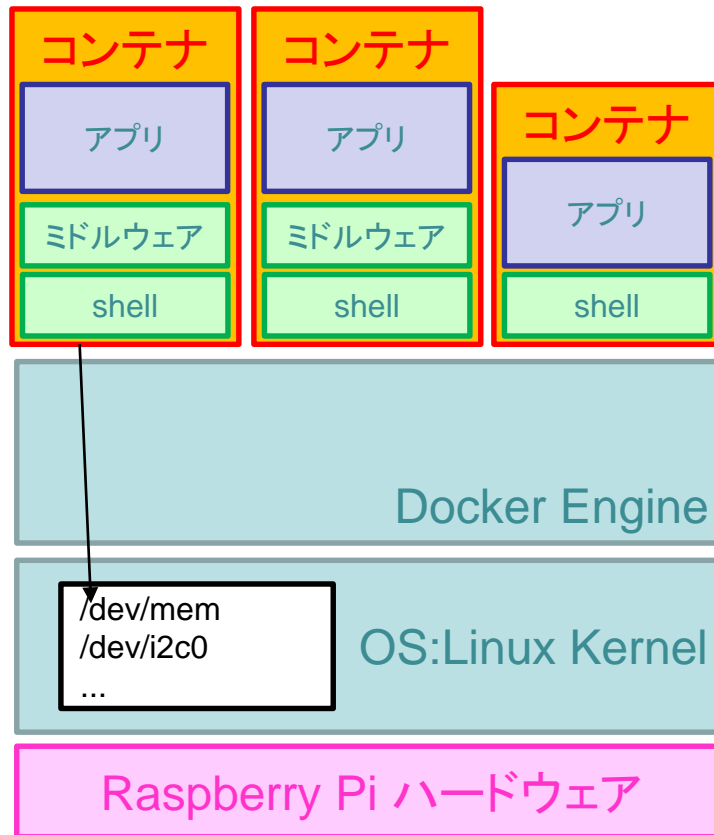
docker run/docker create時の引数で  
ポート番号を指定すると  
ホストOSのポートをコンテナのポートと  
接続することができる

例： -p 1880:1880 と指定した場合、  
ホストのポート1880が  
コンテナのポート1880に  
転送される

## ・ホストOSのデバイスツリー(/dev/\*)をコンテナから使う

標準では、コンテナからホストOSのデバイスを使うことはできない。

→GPIOやI2C,UART,カメラなどをコンテナから使えない。



docker run/docker create時の引数で  
--privileged  
を指定すると  
ホストOSのポートをコンテナのポートと  
接続することができる

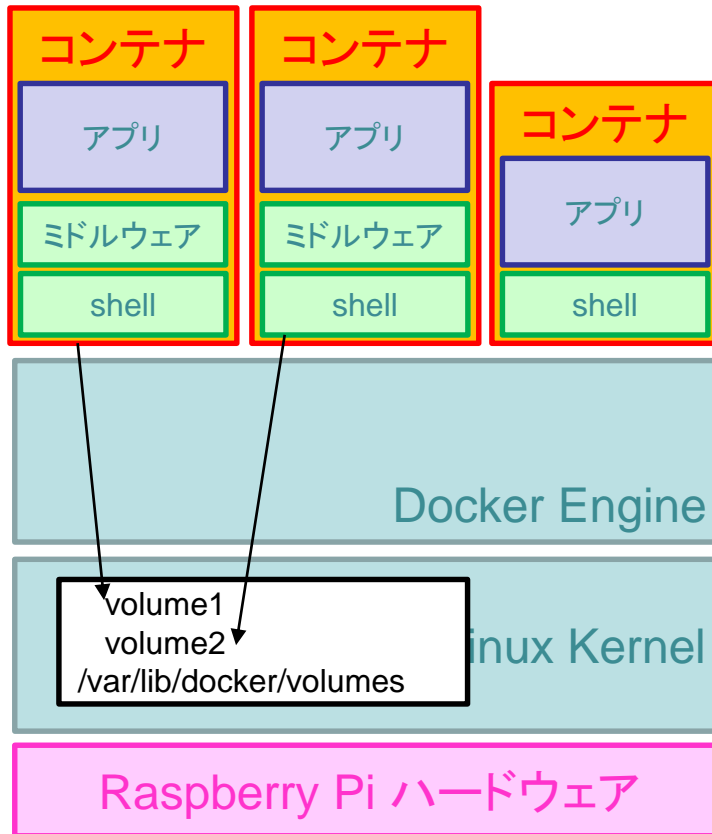
※複数のコンテナでprivilegedを指定すると  
高負荷になる場合があるようです。

その場合は

--security-opt label:disable --cap-add SYS\_ADMIN

## ・ホストOSのストレージをコンテナから使う

標準では、コンテナを停止すると、コンテナ内のファイルは消えてしまう  
→ホストOSにvolumeを作成し、それをコンテナにマウントすることで  
コンテナが停止してもファイルを永続化することができる



まずホストOS側で  
docker volume create volume名  
でvolumeを作成する。  
(ホストOSの/var/lib/docker/volumesに  
生成される)

docker run/docker create時の引数で  
--mount source=volume名,target=マウント先  
でコンテナにmountされる

※複数のコンテナで同じvolumeをmountして  
ファイルを共有することができる

## • Docker HUBの raspbian/stretchが使えます

イメージの取得 : `docker pull raspbian/stretch`

イメージの確認 : `docker images`

1つ目のコンテナの作成 : `docker create -it --privileged --name raspbian1 イメージID`

1つ目コンテナの実行 : `docker start raspbian1`

1つ目のコンテナに入ってみる : `docker exec -it raspbian1 bash`

1つ目のコンテナでwiringpiを導入してみる : `apt update -y; apt install wiringpi`

1つ目のコンテナでwiringpiを試してみる : `gpio readall; gpio mode 29 out; gpio write 29 1`

wiringpi導入後のコンテナから、イメージを作る : `docker commit raspbian1` リポジトリ名:タグ

2つ目のコンテナの作成 : `docker create -it --privileged --name raspbian2 イメージID2`

2つ目コンテナの実行 : `docker start raspbian2`

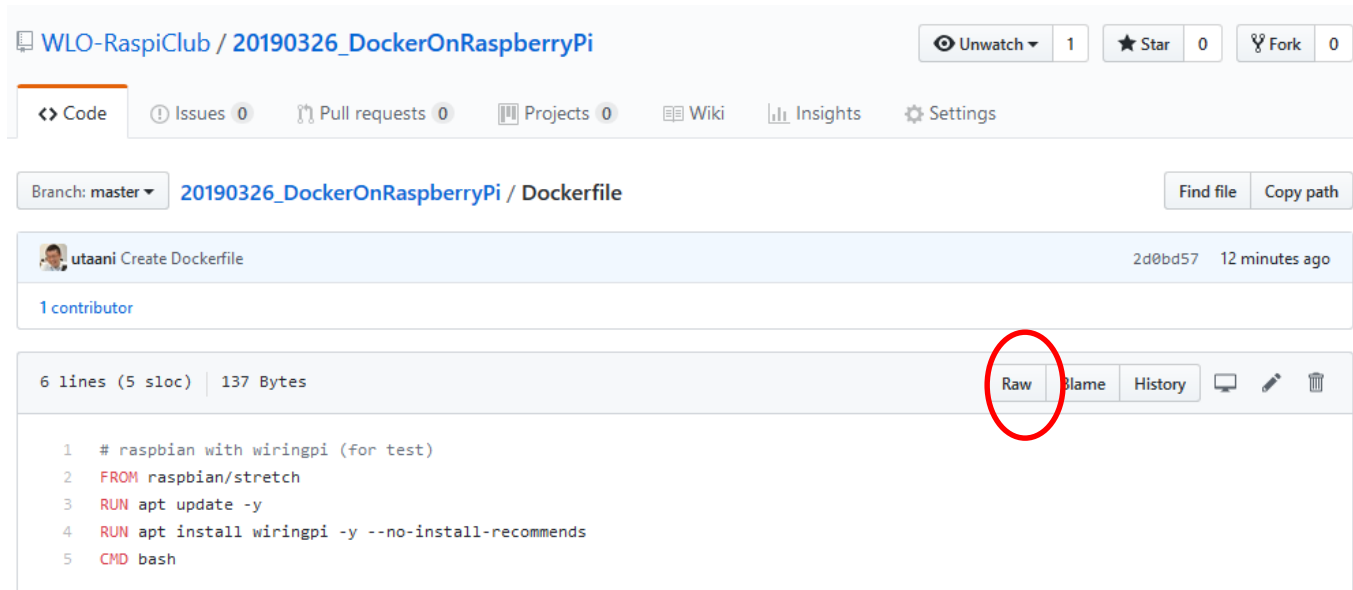
2つ目のコンテナに入ってみる : `docker exec -it raspbian2 bash`

## ・ Dockerfile: 基になるイメージからコンテナを作成する手順を書いたレシピ

前ページの手順をDockerfileにしたらこんな感じ

```
FROM raspbian/stretch
RUN apt update -y
RUN apt install wiringpi -y --no-install-recommends
CMD bash
```

[https://github.com/WLO-RaspiClub/20190326\\_DockerOnRaspberryPi/blob/master/Dockerfile](https://github.com/WLO-RaspiClub/20190326_DockerOnRaspberryPi/blob/master/Dockerfile)



WLO-RaspiClub / 20190326\_DockerOnRaspberryPi

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master 20190326\_DockerOnRaspberryPi / Dockerfile Find file Copy path

utaani Create Dockerfile 2d0bd57 12 minutes ago

1 contributor

6 lines (5 sloc) 137 Bytes

Raw Blame History

```
1 # raspbian with wiringpi (for test)
2 FROM raspbian/stretch
3 RUN apt update -y
4 RUN apt install wiringpi -y --no-install-recommends
5 CMD bash
```

```
pi@raspberrypi:~ $ mkdir /home/pi/wlopi
pi@raspberrypi:~ $ cd /home/pi/wlopi
pi@raspberrypi:~ $ wget https://github.com/WLO-RaspiClub/20190326_DockerOnRaspberryPi/raw/master/

pi@raspberrypi:~ $ docker build -t raspbian-wiringpi2 /home/pi/wlopi/
Sending build context to Docker daemon 2.048kB
Step 1/4 : FROM raspbian/stretch
--> 82ca4d7b3224
Step 2/4 : RUN apt update -y
--> Running in 442e21c7016b
<中略>
Successfully built 7a2f01ab6f6b
Successfully tagged raspbian-wiringpi2:latest
pi@raspberrypi:~ $ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
raspbian-wiringpi2	latest	7a2f01ab6f6b	8 minutes ago	213MB
raspbian-wiringpi	latest	91b8de93b22a	24 minutes ago	213MB
raspbian/stretch	wiringpi	ad0d2b97b588	About an hour ago	301MB
arm32v7/node	10.15.3-stretch-slim	7e9baa71de19	2 weeks ago	118MB
alpine	latest	ef872b76f64d	2 weeks ago	3.7MB
debian	latest	101cc557c042	2 weeks ago	86.2MB
hello-world	latest	618e43431df9	2 months ago	1.64kB
raspbian/stretch	latest	82ca4d7b3224	11 months ago	165MB

```
pi@raspberrypi:~ $
```

## ・ローカルリポジトリ/ビルドサーバ

Docker HUBは外部のクラウドサービスなので、公開されたら困るコンテナやイメージを預けるのが不安な場合、ローカルネットワーク内にDocker HUBに代わるリポジトリを構築することができます。

その際、Raspberry Piのように非力なホストOSに代わってdocker buildを実行してイメージを作成してくれるビルドサーバを併設することができます。

(CPUアーキテクチャが違う場合でも、クロスコンパイル環境で高速にdocker buildできる)

## ・複数のコンテナからなるシステムを一度に構築

Docker Composeを使って、一度に複数のコンテナを作成、ネットワークやストレージの設定も実施して一度にシステム全体を構築できます。

## ・複数のホストからなる大規模なシステムを構築

Docker Swarmを使うことで、複数のホストから構成される多数のコンテナの導入・管理をサポートします。(オーケストレーション)

- Qiita記事「【図解】 Dockerの全体像を理解する 」

前編 : <https://qiita.com/kotaro-dr/items/b1024c7d200a75b992fc>

中編 : <https://qiita.com/kotaro-dr/items/88ec3a0e2d80d7cdf87a>

後編 : <https://qiita.com/kotaro-dr/items/40106f13d47bfcbc2572>



- Qiita記事「よく使うDockerコマンド」

[https://qiita.com/tera\\_shin/items/8a43e904bd15990d3129](https://qiita.com/tera_shin/items/8a43e904bd15990d3129)