

参考

<https://qiita.com/curseoff/items/a9e64ad01d673abb6866>

<http://www.atmarkit.co.jp/ait/articles/1407/08/news031.html>

<https://qiita.com/k1tajima/items/bf8a714e39e857963abc>

<https://qiita.com/silane1001/items/973f32e05defe254408c>

<https://qiita.com/haveAbook/items/09cc4f1e4b8c05364dad>

<https://andrewlock.net/creating-multi-arch-docker-images-for-arm64-from-windows/>

インストール

CentOS7

yum でインストール

```
yum install docker
```

グループ作成

```
groupadd docker  
gpasswd -a user docker
```

dockerroot グループが自動で作成されるが、docker グループは作成されない。
docker グループがないと、

```
/var/run/docker.sock
```

のグループが root になるので root 以外のユーザが docker コマンドを実行できない。

サービス

```
$ sudo systemctl start docker  
$ sudo systemctl enable docker
```

proxy を使う場合

systemd の起動ファイルで環境変数を定義する

```
# cp /usr/lib/systemd/system/docker.service /etc/systemd/system/
```

docker.service

```
Environment="HTTP_PROXY=http://user:pwd@proxy.example.com:8080"
```

サービス再起動

```
# systemctl daemon-reload  
# systemctl restart docker
```

使い方

イメージ管理

イメージの検索

```
docker search ubuntu
```

イメージ取得

```
docker pull centos
```

ローカルイメージ一覧

```
docker images  
docker image list
```

イメージの CMD を確認する

```
docker inspect imageID
```

の CMD の行。

```
docker inspect imageID | grep CMD
```

でも良い。

コンテナ管理

コンテナ起動

bash 起動

```
docker run -it centos:7 /bin/bash
```

バックグラウンド起動

```
docker run -itd centos:7 /bin/bash
```

プロセスを終了せずにコンテナから抜ける

```
ctrl+p, ctrl+q
```

コンテナ確認

起動中のコンテナ

```
docker ps
```

全てのコンテナ

```
docker ps -a
```

起動中のコンテナの端末取得

```
docker attach コンテナ ID または、コンテナ名
docker attach TEST
```

コンテナ削除

```
docker rm コンテナ ID または、コンテナ名
```

全て削除する場合は

```
docker rm $(docker ps -aq)
```

ネットワーク関連

ネットワークの種類

```
docker network ls
```

ネットワーク指定

```
docker run -it --net=host centos:6 /bin/bash
```

docker network ls の NAME を --net=xxx に指定する。

ネットワークなし、ブリッジ、ホストのネットワークを共有などが選択できる。

ネットワーク (ポート転送)

```
docker run -it -p 8080:80 centos:6 /bin/bash
```

ボリューム関連

ホストとコンテナのマッピング

```
docker run -it -v /var/www:/var/www centos:6 /bin/bash
docker run -it -v /var/www:/var/www:Z centos:6 /bin/bash
```

:Z は SELinux が有効な場合に適切なラベルを付与する

docker -v の uid,gid は読書した OS のユーザの uid,gid を使う。

そのため、ホストとコンテナで uid,gid を合わせたほうが無難。

いっそ、ユーザのマッピングをしたい場合は ssh をマウントするほうが手っ取り早いかも

コンテナからイメージ作成

```
docker commit コンテナ ID リポジトリ名: タグ
docker commit コンテナ ID test:1
```

コンテナのエクスポート、インポート

エクスポート

```
docker export コンテナ ID または コンテナ名 > mycontainer.tar
```

インポート

```
cat filename.tar | docker import - REPOSITORY[:TAG]
```

イメージのセーブ、ロード

セーブ

```
docker save IMAGE > filename.tar
```

ロード

```
docker load < filename.tar
```

コンテナ内の systemd 環境でサービスを起動する

コンテナ内の systemd 環境でサービスを起動しようとするとうエラーになることがある

```
docker run --privileged --name centos -d centos /sbin/init
```

でコンテナを起動して、

```
docker exec -ti centos bash
```

で、起動したコンテナで bash を起動する。

vim 等が文字化けする場合

```
export LANG=en_US.UTF-8
```

または、

```
localedef -f UTF-8 -i ja_JP ja_JP
export LANG=ja_JP.UTF-8
```

debian 系の場合

```
apt install language-pack-ja-base language-pack-ja
```

vim ~/.bashrc

```
export LANG=ja_JP.UTF-8
```

Dockerfile から build する

```
docker build
```

プロキシを使う場合は

```
docker build --build-arg HTTP_PROXY=http://10.20.30.2:1234 .
```

コンテナとホストのユーザマッピング

参考

<https://docs.docker.jp/engine/security/usersns-remap.html#rootless-distribution-specific-hint>

<https://docs.docker.com/engine/security/usersns-remap>

<https://stackoverflow.com/ja/q/9729190>

<https://unskilled.site/%E3%83%A6%E3%83%BC%E3%82%B6%E3%83%BC%E5%90%8D%E5%89%8D%E7%A9%BA%E9%96%93%E3%81%A7docker%E3%82%B3%E3%83%B3%E3%83%86%E3%83%8A%E3%81%A8%E3%81%A8%E3%83%9B%E3%82%B9%E3%83%88%E3%81%AE%E3%83%A6%E3%83%BC/>

ディストリビューションごとの設定

ユーザ名前空間を利用するにはディストリビューションごとに設定が必要なことがある。

Ubuntu

- ・特に設定は不要
- ・Ubuntu 16.04、18.04、20.04 は動作確認済

CentOS 8

- ・ `sudo dnf install -y iptables` が必要になる場合があります。

CentOS7

- ・ `/etc/sysctl.conf` (または `/etc/sysctl.d`) に `user.max_user_namespaces=28633` を追加し、`sudo sysctl --system` を実行。
- ・CentOS 7.7 で動作確認済

CentOS7.6 以前

- ・COPR パッケージ `vbatts/shadow-utils-newxidmap` をインストール。

CentOS7.5 以前

`sudo grubby --update-kernel=ALL --args="user_namespace.enable=1"` の実行と、有効化のために再起動が必要。

subuid、subgid の編集

コンテナの uid とホストの uid のマッピングを定義する。

コンテナの 0(root) とホストの 1000 番以降 65536 までマッピングする。

ホスト	ゲスト
1000	0

1001	1
...	...
66536	65536

/etc/subuid

```
user1:1000:65536
```

/etc/subgid

```
user1:1000:65536
```

docker の起動オプションを変更

2 つ方法がある。どちらでもいい。

方法 1 daemon.json を編集

/etc/docker/daemon.json

```
{
  "userns-remap": "user1"
}
```

方法 2 /etc/sysconfig/docker を編集

/etc/sysconfig/docker

OPTIONS に

```
--userns-remap=user1:user1
```

を追加。

確認

docker サービスを再起動すると

```
/var/lib/docker
```

に

```
1000.1000
```

等の uid.gid のディレクトリができる。

Docker の IP アドレス

デフォルトの IP アドレス

docker の IP アドレス割り振りはデフォルトで下記のようにになっている

```
基底 IP アドレス : 172.16.0.0/12
```

ゾーン割当サイズ : /16 ごと

もし、172.16/12 と同じネットワークが存在すると通信に支障がでる

デフォルトの IP アドレスを変更する

Docker 標準のブリッジ docker0 の IP アドレスと networks のブリッジの IP を変更するには以下のようになる。

```
{
  "bip": "192.168.0.1/24",
  "default-address-pools": [
    {
      "base": "192.168.0.0/16",
      "size": 24
    }
  ]
}
```

bip が docker0 の設定、default-address-pools は networks の設定。

Dockerfile

コンテナ作成、実行を自動化できる。

例えば、php 用の apache サーバを立てるときは以下のようにする。

Dockerfile

```
From php:7.1-apache-stretch
RUN echo "RUN は build 時に実行 "
RUN apt update
RUN a2enmod rewrite
RUN docker-php-ext-install pdo_mysql
CMD echo "CMD は run 時に実行 (一つだけ。複数ある場合は最後のものだけが実行される。"
```

イメージ作成

```
docker build -t test.tag -f Dockerfile .
```

設定ファイル名が Dockerfile の場合は、省略可。

コンテナ実行

```
docker run --rm -it test.tag /bin/bash
```

実行時に引数が無ければ image の CMD が実行される。

image の CMD を確認したいときは

```
docker inspect imageID
```

の

```
CMD
```

の行で確認できる。

Docker Compose

複数のコンテナを作成、起動ができる。

例えば、cakephp の環境を作成したいときに、web サーバ、DB サーバ (mysql)、DB 管理用 (phpMyAdmin) をまとめて作成、起動ができる。

コンテナからホストのサービスにアクセスする

以下の設定をすることで、host.docker.internal でホストにアクセスできる

```
services:
  SERVICE:
    image: IMAGE
    extra_hosts:
      - host.docker.internal:host-gateway
```

コマンドライン補完

<https://docs.docker.jp/compose/completion.html>

```
curl -L https://raw.githubusercontent.com/docker/compose/$(docker-compose version --short)/contrib/completion/bash/docker-compose > /etc/bash_completion.d/docker-compose
```

サンプル 1

web サーバと mysql、phpmyadmin を作成

docker-compose.yml

```
version: '3'
services:
  web:
    build:
      context: .
      dockerfile: ./docker/web/Dockerfile
    # user: "1000:1000"
    # tty: true
    # 最後の :z は selinux のラベル書き換え
    volumes:
      - ./src:/var/www/html:z
    ports:
      - 80:80
    depends_on:
      - mysql
  mysql:
    image: mysql:5.7
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - TZ=Asia/Tokyo
      - LANG=C.UTF-8
    ports:
      - 3306:3306
    volumes:
      - dbdata:/var/lib/mysql
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    environment:
      - PMA_ARBITRARY=1
      - PMA_HOST=mysql
      - PMA_USER=root
      - PMA_PASSWORD=root
    depends_on:
      - mysql
    ports:
      - 81:80
volumes:
  dbdata:
```

./docker/web/Dockerfile

```
From php:7.1-apache-stretch
RUN echo "RUN は build 時に実行 "
RUN apt update
RUN a2enmod rewrite
RUN docker-php-ext-install pdo_mysql
```

サンプル 2

web、phpadmin、metabase 等複数の web サービスを立てつつ、reverse-proxy を立てる

docker-compose.yml

```
version: '3'
services:
  reverse-proxy:
    image: nginx:latest
    ports:
      - 80:80
    volumes:
      - ./docker/reverse-proxy/default.conf:/etc/nginx/conf.d/default.conf:z
    depends_on:
      - web
      - phpmyadmin
      - metabase
  web:
    build:
      context: .
      dockerfile: ./docker/web/Dockerfile
    # user: "1000:1000"
    # tty: true
    # 最後の :z は selinux のラベル書き換え
    volumes:
      - ./src:/var/www/html:z
    ports:
      - 8080:80
    depends_on:
      - mysql
  mysql:
    image: mysql:5.7
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - TZ=Asia/Tokyo
      - LANG=C.UTF-8
    ports:
      - 3306:3306
    volumes:
      - dbdata:/var/lib/mysql
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    environment:
      - PMA_ARBITRARY=1
      - PMA_HOST=mysql
      - PMA_USER=root
      - PMA_PASSWORD=root
    depends_on:
      - mysql
    ports:
      - 81:80
  metabase:
    image: metabase/metabase
    ports:
      - "3000:3000"
    depends_on:
      - mysql
volumes:
  dbdata:
```

./docker/web/Dockerfile

```
From php:7.1-apache-stretch
RUN echo "RUN は build 時に実行 "
RUN apt update
```

```
RUN a2enmod rewrite
RUN docker-php-ext-install pdo_mysql
```

./Docker/reverse-proxy/default.conf

```
server {
    listen      80;
    server_name localhost;

    location /phpmyadmin/ {
        proxy_pass http://phpmyadmin:80/;
        proxy_set_header Host $http_host;
    }

    location /metabase/ {
        proxy_pass http://metabase:3000/;
        proxy_set_header Host $http_host;
    }

    location / {
        proxy_pass http://web:80/;
        proxy_set_header Host $http_host;
    }
}
```

サンプル3

docker 内に Linux を同じ UID のユーザを作成する

initEnv.sh

```
echo UID=`id -u` > .env
echo GID=`id -g` >> .env
```

または、.env ファイルを作っておく

.env

```
UID=1000
GID=1000
```

docker-compose.yml

```
version: '3'
services:
  web:
    build:
      context: .
      dockerfile: ./docker/web/Dockerfile
    args:
      - UID=${UID}
      - GID=${GID}
    # user: "1000:1000"
    tty: true
    # 最後の :z は selinux のラベル書き換え
    volumes:
      - ./src:/var/www/html:z
    ports:
      - 80:80
    depends_on:
      - mysql
  mysql:
    image: mysql:5.7
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - TZ=Asia/Tokyo
      - LANG=C.UTF-8
    ports:
      - 3306:3306
    volumes:
```

```

    - dbdata:/var/lib/mysql
phpmyadmin:
  image: phpmyadmin/phpmyadmin
  environment:
    - PMA_ARBITRARY=1
    - PMA_HOST=mysql
    - PMA_USER=root
    - PMA_PASSWORD=root
  depends_on:
    - mysql
  ports:
    - 81:80
volumes:
  dbdata:

```

docker/web/Dockerfile

```

From centos:centos7

RUN echo "RUN は build 時に実行 "
# RUN yum -y install ¥
#   epel-release ¥
#   java-11-openjdk-devel ¥

RUN yum -y install ¥
    sudo
RUN echo '%wheel          ALL=(ALL)        NOPASSWD: ALL' >> /etc/sudoers

ARG UID
ARG GID
RUN useradd -m -u ${UID} -U web
RUN groupmod -g ${GID} web
RUN gpasswd -a web wheel
USER web

# CMD echo "CMD は run 時に実行 (一つだけ。複数ある場合は最後のものだけが実行される。"

```

サンプル 4

mate-terminal、meld、mozc を使えるようにした環境。
xserver を事前に立てる必要あり。

```

make build
make reset
make up

```

ssh で接続して

```

sh bin/start_terminal.sh

```

で mate-terminal が起動する。

サンプル 5

```

centos
ubuntu

```

mate-terminal、meld、mozc を使えるようにした環境を ssh を使わずに GUI を表示できるようにした環境

xserver を事前に立てる必要あり。

```

make build
make reset
make terminal

```

サンプル 6

centos

ubuntu

mate-terminal、meld、mozc を使えるようにした環境を WSLg で GUI を表示できるようにした環境

WSLg を使うので、xserver 不要

```
make build
make reset
make terminal
```

その他

デタッチ等のキーを変更する

https://qiita.com/hironobu_s/items/358b0ba5a7089521b60e

~/docker/config.json に detachKeys を追加する

```
{
  "detachKeys": "ctrl-q,ctrl-q"
}
```

とか

コンテナにまとめて設定を適用する

<https://matsuand.github.io/docs.docker.jp.onthefly/engine/reference/commandline/cli/>

ホストの ~/docker/config.json に設定を記載することでコンテナ作成、起動時に自動的に設定を適用する。

例えば、プロキシを設定する場合は以下のようにする。

```
{
  "proxies": {
    "default": {
      "httpProxy": "http://user:pass@example.com:3128",
      "httpsProxy": "http://user:pass@example.com:3128",
      "noProxy": "http://user:pass@example.com:3128",
      "ftpProxy": "http://user:pass@example.com:3128"
    }
  }
}
```

上記の設定をすることで、コンテナの HTTP_PROXY, HTTPS_PROXY, https_proxy, https_proxy などが設定される。

アーキテクチャの異なるイメージを作成する

```
sudo apt-get install -y qemu qemu-user-static
```

または

```
docker run --privileged --rm tonistiigi/binfmt --install all
```

で QEMU の設定をする。2 つ目の方がおすすめ。

```
docker buildx ls
```

で arm が追加されていることを確認。

```
docker build -t hello_arm64 --platform arm64 .
```