@yama07
投稿日 2017年08月21日　更新日 2017年09月04日

Dockerでuid/gid指定可能かつsudo実行可能なユーザとしてコンテナを起動する

Docker

背景

dockerコンテナ内でちょっとした作業をしようとした際に、こんな問題に直面しました。

```
$ docker run --rm -it -v ${PWD}/work:/work ubuntu:14.04

< ちょっとした作業 >

root@02fcfbf664e1:/# mkdir /work/hoge
# コンテナ内で、マウントしたディレクトリ内にファイル作成
root@02fcfbf664e1:/# vi /work/hoge/container_de_sakusei

< 作業終了 >
root@02fcfbf664e1:/# exit
exit

# ホストで、さっき作成したファイルを(ディレクトリごと)削除
```

```
$ rm -rf ./work/hoge
rm: cannot remove ‘./work/hoge/container_de_sakusei’: Perm
```

☹️

ファイルが削除できない。

原因は単純で、コンテナ内でrootユーザとしてファイルを作成したので、オーナがrootになっていました。

```
$ ls -l ./work/hoge/
total 4
-rw-r--r-- 1 root root 10 Aug 18 02:02 container_de_sakuse
```

それなら、`-u`オプションをつけて実行ユーザを変えればいいと思ったんですが、また別の問題が発生しました。

```
$ docker run --rm -it -v ${PWD}/work:/work -u $(id -u):$(i
groups: cannot find name for group ID 1002
I have no name!@ef625fbaf1c8:/$
I have no name!@ef625fbaf1c8:/$ sudo ls
sudo: unknown uid 1001: who are you?
```

☹️

sudoが使えない。（まぁ当然なんですが。）

コンテナ内にユーザ／グループが存在していないので、`groups: cannot find name for group ID`が出たり、`I have no name!`と言われたりするのは仕方ないですが、`sudo`が使えないのは辛い。

やりたいこと

- uid/gidを指定して起動したい
- sudo使いたい

要するに、aptでパッケージを追加したり、コンテナ<->ホスト間でパーミッションに悩まされることなくファイル共有したいということです。

どう実現するか

言葉で説明するより、Dockerfileとentrypointのスクリプトを見ていただいた方が早いと思います。

Dockerfile

```
FROM ubuntu:14.04

# 一般ユーザ名を設定
ENV USER_NAME=developer

# sudoを使用できるようにする
RUN echo "${USER_NAME} ALL=(ALL) NOPASSWD: ALL" >> /etc/su

# 一般ユーザがユーザ/グループを追加できるようにする
# entrypoint.sh内でパーミッションを元に戻す
RUN chmod u+s /usr/sbin/useradd \
    && chmod u+s /usr/sbin/groupadd

COPY entrypoint.sh /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]
CMD ["bash"]
```

entrypoint.sh

```
#!/bin/bash -e

USER_ID=$(id -u)
GROUP_ID=$(id -g)

# グループを作成する
if [ x"$GROUP_ID" != x"0" ]; then
```

```
    groupadd -g $GROUP_ID $USER_NAME
fi

# ユーザを作成する
if [ x"$USER_ID" != x"0" ]; then
    useradd -d /home/$USER_NAME -m -s /bin/bash -u $USER_I
fi

# パーミッションを元に戻す
sudo chmod u-s /usr/sbin/useradd
sudo chmod u-s /usr/sbin/groupadd

exec $@
```

やっていることは以下の通りです。

1. “developer”ユーザがパスワードなしでsudoが実行できるように、sudoersを設定する
2. 一般ユーザが `useradd` コマンドと `groupadd` コマンドを実行できるように、SUIDを付与する
3. `-u` オプションで指定したuid/gidを持つ“developer”ユーザを作成する
4. 一般ユーザが `useradd` コマンドと `groupadd` コマンドを実行できないように、両コマンドのパーミッションを元に戻す（SUIDを取る）

1.と2.をDockerfile内で事前(`docker build`時)におこない、3.と4.をentrypointのシェルスクリプト内でコンテナ起動時(`docker run`時)におこないます。

上記の例では、ubuntu:14.04のイメージを使用していますが、ubuntu:16.04やcentosなどのイメージでもおそらく同様にできます。

SUID（Set User ID）は見慣れない方もいるかもしれませんが、特殊なアクセス権です。`passwd`コマンドの実行ファイルに付与されていたりします。
実行ファイルを実行する際は、通常、そのファイルを実行したユーザの権限で実行されますが、SUIDが設定されている実行ファイルの場合、その実行ファイルの所有者の権限で実行されます。
例えば、`passwd`コマンドは自身のパスワード設定のために一般ユーザであっても実行できますが、この際、rootでしかread/writeできないはずの/etc/shadowに暗号化されたパスワードが書き込まれます。つまり、一般ユーザが実行したのにもかかわらず、`passwd`コマンドの実行ファイルのオーナーであるrootによって実行されたように振る舞います。

今回の場合、`useradd`コマンドと`groupadd`コマンドにSUIDを設定することで、一般ユーザなのにrootユーザのようにユーザ／グループ追加をできるようにしています。

使い方

Dockerイメージのビルド

上記のDockerfileとentrypoint.shを作成して`docker build`するだけです。

```
# Dockerfileを作成
$ vi Dockerfile

# entrypointとなるシェルスクリプトを作成
$ vi entrypoint.sh

# 実行権限を付与
$ chmod a+x entrypoint.sh
```

```
# Docker imageをビルド
$ docker build -t ubuntu_sudo:14.04 .
```

Dockerコンテナの起動

オプションは必要に応じて変更してください。

```
$ docker run \
    --rm -it \
    -u $(id -u):$(id -g) \
    -v ${PWD}/work:/work \
    ubuntu_sudo:14.04

developer@08f6830cc60a:/$ id
uid=1001(developer) gid=1002(developer) groups=1002(develo

developer@08f6830cc60a:/$ sudo ls
bin  boot  dev  entrypoint.sh  etc  home  lib  lib64  medi
```

おまけ:その他の方法

やりたいことを実現する方法は、他にもいろいろあります。

自分専用のDockerイメージを作成する

ユーザ追加およびsudoの設定を済ませた自分専用のDockerイメージを用意する方法です。
他の方がやっていた例として一番多かった方法です。

Dockerfile

```
FROM ubuntu:14.04

ENV USER_ID=1000 \
    GROUP_ID=1000 \
    USER_NAME=developer

RUN groupadd -g $GROUP_ID $USER_NAME \
    useradd -d /home/$USER_NAME -m -s /bin/bash -u $USER_I
    echo "${USER_NAME} ALL=(ALL) NOPASSWD: ALL" >> /etc/su
```

この方法のイケてないところは、イメージをユーザ間で使い回せないことです。
例えば、RubyやPythonなどの開発環境がDockerイメージと

して用意されていた場合、それとは別に、各々が自ユーザのuid/guidを設定したイメージを管理しなければならなくなります。
また、複数マシンを跨ぐ場合、ホスト間でユーザのuid/gidが異なっていると、イメージの作り直しが必要となります。

ホストの認証情報を共有する

ホストが保有している認証情報を、そのままコンテナにマウントする方法です。

```
$ docker run \
    --rm -it \
    -u $(id -u):$(id -g) \
    -v /etc/group:/etc/group:ro \
    -v /etc/passwd:/etc/passwd:ro \
    -v /etc/shadow:/etc/shadow:ro \
    -v /etc/sudoers.d:/etc/sudoers.d:ro \
    ubuntu:14.04
```

この方法のイケてないところは、コンテナ起動(docker run)のコマンドオプションが長いことです。（そう思うのは私だ

けかもしれませんが）
未検証ですが、`:ro`(read only)をつけ忘れてコンテナ内でユーザ追加／削除などを行うと、ホストに影響を与えそうです。
また、コンテナ内の既存のシステムユーザ／グループとホストのシステムユーザ／グループのidがずれていた場合（あるかわかりませんが）、不具合が発生するかもしれません。
それと、ホストのsudoersの情報を引き継ぐので、ホストでsudoが使えなければ、当然コンテナ内でも使用できません。
これが一番致命的かもしれません。
ただし、既存のベースイメージをそのまま使えるという点では非常にイケてると思います。

応用

今回の内容をベースに、デスクトップ環境を作りました。以下記事を参照してください。
Dockerでuid/gid指定可能かつsudo使用可能なデスクトップ環境を構築する(XRDP編)