

Docker: Host Volumes and Host User/Group identifiers

Sep 20, 2015

Hard to understand

Even for proficient and experienced Linux/Unix users it is sometimes not easy to become a Docker-savvy.

Although Docker is all about putting software in a box, it equally requires you to think outside the box you create and consider the host context. Especially, if you are aiming for as portable docker containers as possible.

A difficulty in creating portable containers is mounting writable host volumes into containers and understand how the mapping of the user/group identifiers within the container and the user/group identifiers of the host works. Please be aware that the best approach is the [data volume container approach](#). However, by default – and therefore on almost all hosts – the maximum size of a container is 10GB. You only need host mounted volumes in cases where you are regularly handling data of sizes beyond 10GB.

In a nutshell, I think the most important aspects to know are:

- Volumes are created **ONCE** during container creation
- The host has its own [user identifiers](#) (uid) and [group identifiers](#) (gid) totally independent from the container's uids and gids
- For bind-mounts the host's uids and gids are taken 'as is' in the container and the container writes with the uids and gids of the user in the container. This is by default the root user who has uid=0 and gid=0.

Easy practical approach since Docker 1.8

After digging through several approaches, the one that works best for me is what I call the "Container's Host Common Group Approach" to ensure that files written by the container process seamlessly match the host settings.

That is:

1. Create one or many groups on the host with a certain gid
 - I like to name the host group [dockmaster](#) and give it gid 1496 ([one of the ISO](#)

[standard number for real intermodal containers ;\)](#))

2. Create a directory on host with dockmaster group and add the "[set group ID upon execution](#)" (`setgid`) bit
3. Start a docker container mounting the host directory and assign the host group id to the user within the container by using `--group-add` option
 - This feature is only available since docker 1.8 ([a feature not announced by docker!!](#)). But it is documented in docker man page and also see [pull request for details](#).

That's about it.

Detailed example

```
# step 1
# creating common host group
$ sudo useradd -M --uid 1496 --user-group dockmaster

# step 2
# creating host directory
$ mkdir -p /tmp/common-group-host-dir
# setting setgid bit and mke it writable
$ sudo chmod g+ws /tmp/common-group-host-dir/
# make dockmaster owner
$ sudo chown dockmaster:dockmaster /tmp/common-group-host-dir/
$ ls -lha /tmp/common-group-host-dir/

total 16K
drwxrwsr-x  2 dockmaster dockmaster 4.0K Sep 20 13:47 .
drwxrwxrwt 11 root          root      12K Sep 20 13:48 ..

# step 3
# starting container
sudo docker run \
-v /tmp/common-group-host-dir:/tmp/common-group-container-dir \
--user 1496 \
--group-add 1496 \
-ti --rm \
debian:jessie /bin/bash

# now in container
I have no name!@d2165449e550:/$ touch /tmp/common-group-container-dir/test-file

# check directory content from within container
I have no name!@d2165449e550:/$ ls -lha /tmp/common-group-container-dir/
total 8.0K
```

```
drwxr-sr-x 2 1496 1496 4.0K Sep 20 11:47 .
drwxrwxrwt 3 root root 4.0K Sep 20 11:47 ..
-rw-r--r-- 1 1496 1496 0 Sep 20 11:47 test-file
```

now back to host

I have no name!@d2165449e550:/\$ exit

on host

\$ ls -lha /tmp/common-group-host-dir/

total 16K

```
drwxr-sr-x 2 dockmaster dockmaster 4.0K Sep 20 13:47 .
drwxrwxrwt 11 root root 12K Sep 20 13:48 ..
-rw-r--r-- 1 dockmaster dockmaster 0 Sep 20 13:47 test-file
```

Inn step 1 we create a user and group with name *dockmaster* and uid/gid 1496. We then create a directory */tmp/common-group-container-dir/* make it writable for the group and add the setgid bit with *chmod g+ws*. Lastly, we make *dockmaster* the user and group owner of that directory.



The final 3rd step demonstrates the effect of running a container with *--group-add 1496* and *--user 1496*. Two apsects are important here:

1. Only *--group-add 1496* is necessary here
2. The gid must be given as the numeric number. If you provide a symbolic group name docker searches in posix compliant way for the group name in */etc/group* of the container, which does not exist in this case.
3. *--user 1496* is not necessary. This option demonstrates that one can add a user on the fly and hence run a container as non-root. This follows a [basic security advice](#) and [docker best practice](#).

Summary

With this solution the host operator can create directories and let container users create and write in it – whoever the container user is – by just passing the group id of the host directory via the *--add-group* option. From the container perspective: the user can write into volumes without the need to perform any user mapping activities nor the need to know anything about the host configuration. The permissions are fully controlled by the host operator running docker containers.

Thought.out.printf()

 [renzok](#)
 [renzokott](#)

Becoming a literate scientific developer,
building common understanding.