Make uid & gid configurable for shared volumes #7198

New issue

Closed **digital-wonderland** opened this issue on Jul 23, 2014 · 126 comments

digital-wonderland commented on Jul 23, 2014

Please provide an option to make ownership of shared volumes configurable.

For example my current use case is to have logstash-forwarder running within a container, that has /var/lib/docker shared read-only as volume from the host.

Since /var/lib/docker is set to 0700 root:root on the host I can't access the volume as non root user.

What I would like to have is something like with NFS where one can map uid & gid from the host to users & groups on the client.

I.e. docker run -v /var/lib/docker:/var/lib/docker:ro:$user:$group would make the volume available in the container as read only, belonging to $user:$group.

👍 7

thaJeztah commented on Jul 24, 2014 Docker member

I don't know *how* they implemented this, and if a comparable feature exists on Linux, but on OS X, a feature exists to 'ignore ownership' on a volume. Effectively this will make *any* user see the files/directories as if they were the owner.

Although this won't serve all cases, it may be a nice addition

elgalu commented on Jul 28, 2014

+1

Related: http://stackoverflow.com/questions/23544282/what-is-the-best-way-to-manage-permissions-for-docker-shared-volumes

cpuguy83 commented on Jul 28, 2014

gh#5910 kind of handles this from the SELinux side.

riaanvddool commented on Aug 5, 2014

link: #5910 ;)

rhatdan commented on Aug 5, 2014

The SELinux change is actually changing the labels on the content. Potentially you could do the change on the host. I know of no other way to do this for UID/GID. But doing a chown -R $UID:$GID on the mount point.

Or you could add this type of access using ACLs. But I think you will need to change every INODE on the mount point.

brauner referenced this issue on Aug 25, 2014

Make uid & gid configurable for --devices #7722 Closed

nazar-pc commented on Oct 1, 2014

I also need this feature.

For example, I want to create web container, and attach volumes with websites and configurations to it, so that container will be completely universal for any number of websites.

However, I need git access for pushing code to website repository. Since I want to have my apps to be isolated - I want to have each website directory to be owned by separate user/group, and it would be great if files written by Docker container into volume will be owned by that separate user/group.

OlivierA commented on Oct 12, 2014

+1 for this feature.
I don't understand how read/write volumes can work without it. Expecting the guid/uid to be the same on the image and the host is a strong requirement incompatible with Docker's isolation principles.

I'm personally working around this with ugly and slow useradd/groupadd commands for my Dockerized development tools: https://github.com/ndless-nspire/Ndless/blob/master/ndless-sdk/bin-docker/nspire-docker

soltysh referenced this issue in **openshift/origin** on Oct 14, 2014

Bug 1150921 - Problem with pushing images to public repo caused by missing .dockercfg file in $HOME. #191 Merged

ElessarWebb commented on Nov 6, 2014

I might be completely missing the point. But I was struggling with a similar issue where i want to ensure that the http user has write permissions on /var/log, which is a volume and is likely from the host with root:root as owner.

I solved it by setting an entrypoint that ensures that the logdirectories are created and have the right permissions. I guess this works because the entrypoint script runs as root.

This was referenced on Nov 16, 2014

Vagrant setup missing `http.cors.allow-origin` for Elasticsearch 1.4? digital-wonderland/docker-logstash-forwarder#6 Closed

Get access to container files docker-library/tomcat#3 Open

smyrman commented on Dec 13, 2014

(removed comment -- wrong tab, sorry)

hgl referenced this issue in **boot2docker/boot2docker** on Jan 16, 2015

1.3.0 - Only root can write to OSX volumes / Can't change permissions within #581 Open

jjrv commented on Jan 27, 2015

I hacked around this in Ubuntu outside Docker. Install package bindfs and bind the directory with volume contents to another path while mapping UID and GID to ones used inside the container:

sudo bindfs -u UID -g GID oldpath newpath

Then use newpath as a docker volume. Oldpath still shows ownership correct for the host, newpath for the guest.

👍 1 ❤️ 1

cpuguy83 commented on Jan 27, 2015

@jjv the problem is bindfs is REALLY REALLY slow.

jjrv commented on Jan 27, 2015

@cpuguy83 yes it's far from optimal but maybe helps someone in a similar situation. This was to get things working inside a development virtual machine (vmhgfs doesn't allow setting UID/GID) while in production the UID and GID still have to match between host and guest...

jsternberg commented on Jan 27, 2015

It would actually be nice if a type of bindfs functionality was used when docker implements this, assuming it doesn't cause too much of a performance hit. That way, you wouldn't have to make sure the container was being run as the correct user. It should also be possible to use logical names instead of the literal uid/gid.

cpuguy83 commented on Jan 27, 2015

@jsternberg It is a TREMENDOUS performance hit. Pretty much akin to using vbox shared folders.

cynipe commented on Jan 29, 2015

+1

for the local development use-cases, I think Docker definitely need this feature. And in such case, I want this feature to support the both Windows and OSX.

Vagrant seems to support this by mapping host user's UID/PID to vagrant user's though. But for the developing purpose, I really want to use Docker instead of Vagrant, since it's much lightweight than Vagrant to running multi-host applications.

krisgraham commented on Jan 31, 2015

Please tell me what I'm missing here (I don't have any experience with Go), but doesn't the Go Mount() function accept flags? Couldn't we allow for a command like
`-v host/folder:container/folder -mount-as user:group`
Couldn't you just get the uid/gid with lookup (https://golang.org/src/os/user/lookup_unix.go)
and then pass them (uid=1,gid=1) as flags into Mount()? (https://golang.org/src/syscall/syscall_linux.go?s=19154:19249#L754)

cpuguy83 commented on Jan 31, 2015

@krisgraham bind mounts don't supprt setting uid/gid like that.

rhatdan commented on Jan 31, 2015

Also separating the -v option from the --mount-as option causes confusion when there are multiple -v option

berfarah commented on Feb 9, 2015

What's a good workaround for this? I'd love to use Docker for active development, and not having a mounted volume of some sort isn't really an option as I'd have to rebuild every time I make a change in my code.

The reason I want to use Docker for active development is so that it's consistent with my production environment.

cpuguy83 commented on Feb 9, 2015

@berfarah I use docker for active development every day.
There is rarely the case when I need to mess around with perms.
If you are using boot2docker on OSX, then make sure your working dir lives within /Users and you should be fine.

berfarah commented on Feb 9, 2015

@cpuguy83 Thanks for the quick reply. I'm having permissions issues with a rails environment where logs can't be written, and there are occasionally points of failure because of permissions. This is due to my services having a different UID to the one of the files.

ryneeverett commented on Feb 9, 2015

@berfarah My workaround is to write my dockerfiles such that the container-users/groups that own the code have the same UID/GUID as my host user. E.g., since my host user UID is 1000:

```
RUN \
    groupadd code_executor_group && \
    useradd code_executor_user -g code_executor_group -u 1000
```

👍 1

cpuguy83 commented on Feb 9, 2015

@berfarah Have logs to go stdout/stderr for RAILS_ENV=development?

ncjones commented on Feb 9, 2015

@cpuguy83 This issue does not affect OSX; **thaJeztah** commented on 24 Jul 2014:

> on OS X, a feature exists to 'ignore ownership' on a volume. Effectively this will make any user see the files/directories as if they were the owner.

thaJeztah commented on Feb 9, 2015 Docker member

@ncjones actually, the same applies to OS X. The "volumes" I was talking about there, are the harddisks / partitions (volumes) used on OS X itself. I doubt that makes a difference for working with Boot2Docker, but I'm not sure.

My comment was meant to inform if something *similar* was possible in Linux (thus inside the Boot2Docker VM)

Sorry for the confusion there.

berfarah commented on Feb 9, 2015

@ryneeverett Thanks, that's helpful. ~~Do you then just end up modifying permissions to 775 from 755 and 664 from 644 respectively where you need to?~~ edit: Reading skills!

@cpuguy83 Thanks! That seems like a more limiting fix than **@ryneeverett**'s solution, as it isn't quite something I can just carry forward to all projects.

ryneeverett commented on Feb 9, 2015

@berfarah Glad you found that helpful. Yeah, I just make sure that my host files have the correct permissions and Docker preserves them in the volumes.

muuki88 referenced this issue in **sbt/sbt-native-packager** on Feb 10, 2015

Created Volumes in Docker plugin, are not owned by daemon. #485 Closed

syncomm commented on Mar 9, 2015

+1

This is something that would be extremely useful for shared volumes through docker, and addresses a security concern I have:

- Shared volumes where the container uid and gid accidentally map to a privileged non-root user on the host. (ex. the files created by the container on the host are mapping to a user which can sudo w/o passwd, access otherwise restricted content, etc.)

Fixing this would be very convenient for enterprise folks running a ton of containers w/ Apache or some middleware and sharing out the logging volumes and/or various content/upload directories. I've also personally had some headaches with these permissions while containerizing end-user apps in Linux (like syncomm/spotify). Many of the workarounds today are themselves problematic. Ultimately, this has to be fixed inside docker. I especially don't feel comfortable running root shell scripts as an entrypoint, particularly when this problem highlights how the 0:0 uid/gid in the container will map to root on my host. I like the initial proposal "docker run -v /var/lib/docker:/var/lib/docker:ro:$user:$group".

cpuguy83 commented on Mar 9, 2015

@syncomm I would take the exact opposite approach.
Docker cannot make assumptions about the ownership of your data.

The approach you highlight at the bottom of your comment would be doable if we were to automagically map these real files to a fuse-based fs where we can change uids/gids... and this would incur a significant performance impact.

And soon uid 0 in the container will not be uid 0 on the host... which also makes file ownership even trickier.

md5 commented on Mar 9, 2015

Is this a duplicate of #2259?

syncomm commented on Mar 9, 2015

@cpuguy83 Thanks for the feedback, although I'm not sure what you mean by the exact opposite approach. Cloud you explain? I would agree that the ownership of data shouldn't have to be assumed by docker, but I believe providing a consistent mapping from container to host certainly makes the job of policing this data a lot easier for docker consumers.

I agree, that like the bindfs workaround, making it a fuse wrapper would incur some profound overhead. However, there has to be a way out of this conundrum without huge penalties. Essentially the container is behaving correctly (as if it were a separate unique machine from the host), and when we mount a host's directory as a volume it is (correctly) seeing the POSIX filesystem there, permissions and all. Unfortunately, that makes it really tough to share data between the two "hosts" in a consistent way. Things like nfs, cifs, etc. are used to this and support uid and gid mapping -- I would think there could be a parallel here in solving the problem. Honestly, I need to dig into the repo more to figure out where this is happening in the code and understand it better.

When will the changes you mentioned to file ownership drop? The uid 0 NOT being the same for container and host would actually make me feel a lot more comfortable making an entrypoint of a root shell script. Then it is just a matter of performing the suggested workaround of passing the correct uid/gid as an env and doing an adduser on container launch.

 1

cpuguy83 commented on Mar 10, 2015

@syncomm Unfortunately, not too my knowledge, not without wrapping them in some other kind of filesystem that can be mounted with uid/gid settings.

cyberco referenced this issue in **sameersbn/docker-postgresql** on Mar 11, 2015

Wrong owner of mounted data directory #22 Closed

thaJeztah referenced this issue on Apr 7, 2015

Automatically created host directories for mounts take permissions for daemon umask #12061 Closed

soulrebel commented on Apr 9, 2015

Ultimately this looks like something that requires kernel support for full bidirectional remapping of UIDs and GIDs, and doesn't even look easy to manage.

For non performance-sensitive scenarios maybe something could be done using FUSE.

Still +1 for development use cases, since I end up with root-owned files in my home.

thaJeztah added the kind/feature label on Apr 13, 2015

ibukanov commented on Apr 17, 2015

For my use case it would be enough if docker could just apply custom uid/gid and selinux label to the host volume when it creates it leaving the permission and labels alone if the directory already exists. Currently I workaround that by having a separated entry point for the image that runs as root just to fix the volume permission.

cpuguy83 commented on Apr 17, 2015

@ibukanov it can't apply uid/gid, but selinux label is coming.

ibukanov commented on Apr 17, 2015

@cpuguy83 what is the technical reason for inability to apply a custom uid/gid when creating a host volume directory?

cpuguy83 commented on Apr 17, 2015

@ibukanov See above comments.

ibukanov commented on Apr 18, 2015

@cpuguy83 I just do not see from the comments what the issue is. For my user case I do not need any remapping of uid/gid. Currently inside the container when running as a root I can `chown uid:gid /host_volume && chmod 770 /host_volume`. What would be nice if the docker can do it on its own when it creates the directory on the host behind /host_volume. This way I would not need a hack of providing an extra entry point in the container just to perform the above operation as a container root and can always run the code using non-root account.

cpuguy83 commented on Apr 19, 2015

@ibukanov Ah, that's a different thing.
Docker does not, quite on purpose, make changes to host dirs.
If we do #9092, then if docker created the dir, it'll treat it like a normal volume (ie, copy everything at the

volume path in the container out onto the host and chmod/chown as it was in the container).

ibukanov commented on Apr 19, 2015

@cpuguy83 - well, docker does change something on the host when it creates the missing host volume. But #9092 sounds as if it would solve my use case indeed.

dato commented on May 17, 2015

@cpuguy83 wrote:

> And soon uid 0 in the container will not be uid 0 on the host... which also makes file ownership even trickier.

Is there an existing issue to track this? I was unable to find one.

Thanks!

thaJeztah commented on May 17, 2015 Docker member

> Is there an existing issue to track this? I was unable to find one.

@dato Look for "User Namespace"; an initial implementation can be found here: #12648 and here #11253

But there are various discussions / issues predating that :)

This was referenced on May 20, 2015

[Proposal] paths.php based on the environment laravel/framework#780 Closed

Data file permissions haugene/docker-transmission-openvpn#13 Closed

docker container loop to start and exist activemq disaster37/activemq#5 Closed

ayeo commented on Jun 8, 2015

+1

MrMMorris commented on Jun 9, 2015

just so I know I'm in the same boat as everyone:

I am using the nginx image and mounting a volume to /usr/shared/nginx/html for local development. Nginx runs as `www-data`, but the volume gets mounted as `1000:staff` and so I get `403 forbidden`.

ncjones commented on Jun 10, 2015

@MrMMorris I don't think that's the same boat - this issue is when files created within a container (on to a mounted volume) are not readable outside the container without sudo access. Your www-data user only needs read access to the mounted files so you should be able to just `chmod a+r` on your files.

👎 1

MrMMorris commented on Jun 10, 2015

@ncjones chmod/chown seems to have no effect on permissions in the container

I'm so confused because I swear this worked at multiple points during my use...

MrMMorris commented on Jun 10, 2015

Alright well this is embarrassing...

It seems either after upgrading to compose 1.3.0rc-1 (probably not) OR deleting all my containers/images and running `docker-compose up` again has fixed everything.... No more `403 forbidden`

I knew it was working before somehow so I took the ol' `rm -rf *` approach..

alercunha commented on Jun 14, 2015

I also need this issue resolved somehow. I have a docker image which I use to execute some heavy compilation work which must be isolated. So the container must go up, receive some external input, process then it will produce the binaries (output). Without proper permission management it has been a nightmare trying to get this to work. My images must be portable so I cannot assume much about the host.

motin commented on Jun 14, 2015

@alercunha In your case it sounds fine to use a single user inside the docker container, ie compile as root

rysiekpl referenced this issue on Jun 15, 2015

[1.7.0-dev] regression: using a bindfs mount as a volume ignores bindfs #13937 Closed

ncjones commented on Jun 15, 2015

@alercunha
We've worked around this issue in a build setting by having the Docker container chown the build output. Our Dockerfile will contain something like:

```
env USER_ID 1000
env GROUP_ID 1000
cmd bash -lc '\
  build.sh && \
  chown -R $USER_ID:$GROUP_ID build \
'
```

The actual uid/gid of the Docker host's user can then be provided to the container when run. For example, a Jenkins build command will use something like:

```
docker run \
  -e USER_ID=`id -u` \
  -e GROUP_ID=`id -g` \
  -v $basedir:/workspace
```

👍 4

alercunha commented on Jun 15, 2015

@motin Building as root is something I'm trying to avoid since I am using docker/buildroot to build several third party projects. The idea is to have some protection against that source code messing up with my build environment. So building as root doesn't sound like a good idea.

alercunha commented on Jun 15, 2015

@ncjones Thanks! That is actually a good idea, might work well as a solution in my particular case.

motin commented on Jun 15, 2015

@alercunha In that case it might be worth considering using separate images/containers for each third party project. That could also benefit the open source community better, for instance if you submitted official docker-images for building each of those third-party projects.

ayeo commented on Jun 24, 2015

To grant all privileges to www-data you can use:

```
RUN usermod -u 1000 www-data
```

👍 1

stanislavb commented on Jul 9, 2015

Encountered an issue when running a job in docker on a mounted volume. Resulting files were owned by root and not managable by the user running the docker command. Current workaround is to run following after the job to fix permissions.

```
docker run -t --rm \
    -v $PWD:/usr/src/app \
    -w /usr/src/app \
    debian:wheezy chown -R $(id -u):$(id -g) ./
```

mig-foxbat commented on Jul 11, 2015

is it possible to have docker change the permission of host based mounted volumes to be owned by the user specified in the USER command of the DockerFile? Clearly the permissions has to be configurable but we can have this has the default behaviour.

ibukanov commented on Jul 11, 2015

@mig-foxbat
For my purposes it would be sufficient if docker just sets ownership and permission on the host directory from the mount point in the image when the host directory is created for the first time. If the host directory already exists, the docker could leave it alone.

digital-wonderland commented on Jul 14, 2015

@mig-foxbat & **@ibukanov** the mandatory prerequisite for me, when opening this issue, was that the permissions on the hosts file system do **not** get changed (as in have some virtual in container mapping similar to what one can do with *NFS*).

What you are trying to do can easily be done today by running *chown* in some startup script within your container.

terrisgit commented on Jul 14, 2015

I love the upcoming volume plug-in architecture but I would much rather the Docker team add this feature. Docker works well until you use volumes with non-root user in your container. Then when things (e.g., logging) don't work you either throw your laptop through the Starbucks window or you find the stackoverflow item referenced above.

👍 1

This was referenced on Jul 14, 2015

How to enable logging jtgasper3/docker-shibboleth-idp#10 — Closed

Chown contents on volume mount if user specifies :u #14632 — Closed

dhbaird commented on Jul 16, 2015

+1 I would want to have containers running applications as non-root user, which becomes complicated due

to this issue.

Why not implement this feature using LXC's id_map capability?

dhbaird commented on Jul 16, 2015

I just noticed the answer to "id_map" (aka User Namespace support) is mentioned above already by **@thaJeztah**

mitchcapper commented on Jul 16, 2015

While it takes a bit of extra work to set permissions we are running dozens of containers as non-root and avoiding the permission problems. Most of our data containers start with something like:
ENV CONTAINER_USER consul
ENV CONTAINER_UID 312312
RUN adduser -D $CONTAINER_USER -u $CONTAINER_UID
RUN mkdir -p /var/consul/
CMD chown $CONTAINER_USER.$CONTAINER_USER /var/consul/

With the actual run container being similar in terms of setting the UID/useradd info. This is some duplicate work (having to set the UID on both the data container and the run container) however its fairly minor. Having docker change permissions or map permissions on the fly would add some decent complexity and users would have to be more careful from a security perspective. I am also for whatever helps reduce the complexity of the command line, so if everything is self contained in the dockerfile its a bit more straightforward what is happening.

Now there is the note above of trying to access host files in a container with some sort of mapping. I think this is certainly a risky proposition. Often trying to share files without direct host mapping is probably the best solution, as otherwise you are somewhat inverting docker security. Rather than a container having less permissions than it would otherwise you are talking about a container running as non-root having root level access (presumed often use case) to host files. Group permissions or other solutions I feel should win out in most situations.

iadknet commented on Jul 25, 2015

I worked around this with an entrypoint shell script that chowns the mounted directory and the executes the subcommand via su.

An example in a solr container where the /data directory is mounted in as a volume from the host:

```
#!/bin/bash

chown -R $SOLR_USER:$SOLR_USER /data

su -c "$@" -m $SOLR_USER
```

This was referenced on Aug 8, 2015

- **Setting UID/GIDs** dokku/dokku#1376 — Closed
- **Switch to official Logstash Docker image** CiscoCloud/mantl#596 — Closed
- **Work out persistent data locations and mount as volumes instead** gavinsaunders/blendcluster#7 — Open
- **Code coverage ownership** JulienBreux/phpunit-docker#13 — Closed
- **Persistent data** itzg/dockerfiles#27 — Closed

GordonTheTurtle commented on Aug 28, 2015

USER POLL

The best way to get notified when there are changes in this discussion is by clicking the Subscribe button in the top right.

The people listed below have appreciated your meaningfull discussion with a random +1:

@jcercurati
@iangelov
@scue
@razvanphp
@ghost
@andrerocker
@anandnalya
@ddopson
@hason
@hadim
@iyn
@cgrantcharovtp
@sandytrinh
@gomex
@DrBenton
@tehmaspc
@segphault
@avaz
@edavism
@ayeo
@stanislavb
@smebberson
@tony-kerz
@msierks
@pdonorio
@samsong8610
@qm78
@joshughes
@roelvanduijnhoven
@vladimirbright
@ava-dylang

ava-dylang commented on Aug 31, 2015

Saying anything on these threads subscribes a user automatically, so yes all those people are subscribed unless they've additionally, intentionally unsubscribed. (also not sure *who* you're referring to as `your`)

bobef commented on Sep 7, 2015

This is definitely a must. Docker must remap UIDs properly. Otherwise everything created from the container to the host OS volume is owned by root. The workaround is terrible - changing users inside the container which leads to all sorts of complications, having to chown everywhere, the Dockerfile grows with ugliness. And worst of all hardcoding UIDs make the containers unportable!

👍 3

k0pernikus commented on Sep 7, 2015

+1 This feature would be highly appreciated, as I want to avoid switching between docker container and my host system, just to avoid overwriting users id on files.

dhbaird commented on Sep 7, 2015

Getting UID customization is possible right now, but within the following constraints:

- Uid customization is possible by making the container accept the uid(s) it needs via environment variable. This gives the host has full control over the uid(s) the container uses, and has an added benefit: multiple variables may be set to the same uid value. Uid remapping is unable to support multiple container uids mapping to a single host uid, since the mapping has to be 1:1. For example, if the desired mapping was in fact that the application should run as uid=0, you could do that by injecting with variables, but not via uid remapping. See **@ncjones** and **@mitchcapper** solution above.
- The root uid=0 notably however cannot be changed to a different value, except by remapping: #12648.

This was referenced on Sep 10, 2015

Extending docker-compose file to add volumes, point to logstash config file spujadas/elk-docker#7 — Closed

Create jovyan in Dockerfile, reset UID at startup jupyter/docker-stacks#31 — Merged

Document how to overwrite solrconfig.xml and schema.xml makuk66/docker-solr#10 — Closed

phpguru commented on Sep 21, 2015

Almost all modern web development frameworks have a logs folder and a cache folder that is inside the project, defined in project config and shared via the mounted volume. I used vboxfs first but was too slow. I found docker-machine-nfs to create NFS mounts and it's fast, but now my web files in containers are owned by 502:dialout and can't be chmod or chown so my web app won't run. I tried docker vagrant box but it requires vagrant with all different env vars and configs, and doesn't yet work with docker-machine. Even if its a manual process for the short term, is there a workaround to add user 502 to www-data or something? That's all I need to be in business!

👍 1

pro100coder commented on Oct 13, 2015

+1

thaJeztah commented on Oct 13, 2015 — Docker member

For those following this; there's currently a PR for an experimental feature that will recursively change ownership of bind-mounted files. It should be used with care though, because that feature will change the files on the host (which may not be what you want in all cases); see #14632 (and #16767)

This was referenced on Oct 14, 2015

Permission denied error ches/docker-kafka#9 — Open

pyenv does not run using sudo yyuu/pyenv#79 — Closed

krikkiteer commented on Nov 10, 2015

+1

kvaps commented on Nov 10, 2015

By the way, I solved this problem at using ACL

posita commented on Nov 10, 2015

@kvaps, care to elaborate?

 **MaksymBilenko** referenced this issue in **MaksymBilenko/docker-oracle-xe-11g** on Nov 10, 2015

Latest image not running #5 — Closed

denydias commented on Nov 12, 2015

@posita, my bet is **@kvaps** is talking about POSIX ACLs. If I'm right, that's not the same as the OP request.

kvaps commented on Nov 12, 2015

@posita, **@denydias**, sorry for long answer. Yes, I am talk about this.
I have many containers like owncloud, samba and minidlna. Everyone works with different uid and gid.

They all operate with the same files. What would everyone can read and write files, I mounted filesystem with `acl` option into host machine, and gave everyone uid and gid rights for this files just simple, like `chown` command:

```
# give access for owncloud (apache uid 33):
setfacl -R -m "u:33:rwx" /data
# give access for samba (uid 1000):
setfacl -R -m "u:1000:rwx" /data
# give access for minidlna (uid 997):
setfacl -R -m "u:997:r-x" /data
# preserve this permissions for new files and folders:
setfacl -R -d -m "u:33:rwx" /data
setfacl -R -d -m "u:1000:rwx" /data
setfacl -R -d -m "u:997:r-x" /data
```

bigeasy added a commit to bigeasy/homeport that referenced this issue on Nov 15, 2015

Rename `uid` and `gid` on boot. ... 3b66be0

digital-wonderland referenced this issue in **digital-wonderland/docker-logstash-forwarder** on Nov 17, 2015

[aufs] Not picking up /etc/logstash-forwarder.conf within containers #23 Open

pro100coder commented on Nov 18, 2015

@kvaps, command `setfacl` not work in `Dockerfile`. Example:

```
FROM nginx

ADD data/conf /etc/nginx

RUN mkdir -p /etc/nginx/sites-enabled

VOLUME /etc/nginx

RUN setfacl -dR -m u:1000:rwx /etc/nginx && setfacl -R -m u:1000:rwx /etc/nginx
```

Result:

```
root@3975ac4fba98:/etc/nginx# getfacl sites-enabled/
# file: sites-enabled/
# owner: root
# group: root
user::rwx
group::r-x
other::r-x
```

ACLs works only after mount on host machine and run `setfacl`. Docker version:

```
Client version: 1.7.1
Client API version: 1.19
Go version (client): go1.4.2
Git commit (client): 786b29d
OS/Arch (client): linux/amd64
Server version: 1.7.1
Server API version: 1.19
Go version (server): go1.4.2
Git commit (server): 786b29d
OS/Arch (server): linux/amd64
```

dhbaird commented on Nov 18, 2015

This is very simple to do right, and also very simple to screw up. So do not screw it up!
The right way to do this is: The first time the container is run, the host injects which uids the container should use (as already described above).
The worse way to do this is: expect the container to impose its uid requirements on the host. This is wrong, wrong, wrong. For example, consider two containers that share a common mount: one container writes files

to serve, and another container runs httpd to serve the files. Now, if both of these containers have competing definitions for uid, this system will be broke at worst, or deficient at best. The ACLs being set won't make any sense to the host, and they'll likely be more wide than necessary, which means that it's now a security problem. Please do not do this!! Do it the right way.

LaurensRietveld commented on Nov 19, 2015

Agreed. Leaving the mapping between UID to the container is a good enough solution. Just wanted that docker leads the way by example, and supports such UID/GID mappings in the official docker images.

pa-de-solminihac commented on Nov 21, 2015

Solved at **run** time, without having to `chown`.

I've used something close to **@ncjones** ' solution, but I don't want to `chown` files because I don't want them to be modified on the host. So I chose to change the UID at container startup.

I create a dedicated user in my `Dockerfile`:

```
RUN adduser --no-create-home --disabled-login --gecos "" username --uid 1000
```

I define a startup script in my `Dockerfile`:

```
CMD ["/run.sh"]
```

I have this line in my `/run.sh` script :

```
usermod -u $USER_ID username
```

Now I can choose `USER_ID` when container starts:

```
docker run -e USER_ID=$(id -u)
```

brthor commented on Nov 25, 2015

Managed to solve this using the new dockerfile args. It doesn't require doing anything special after the container is built, so I thought I'd share. (Requires Docker 1.9)

In the Dockerfile:

```
# Setup User to match Host User, and give superuser permissions
ARG USER_ID=0
RUN useradd code_executor -u ${USER_ID} -g sudo
RUN echo 'code_executor ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers
USER ${USER_ID}
```

Then to build:

```
docker build --build-arg USER_ID=$(id -u)
```

Everything else is as normal :)

👍 1

pdonorio commented on Nov 26, 2015

Edit:
i wrote this post on the wrong issue (it was another one related to boot2docker).
Sorry.

riquito commented on Nov 27, 2015

@pa-de-solminihac `usermod -u $USER_ID username` will change the user id of `username` and will trigger a ownership change of the files in `username`'s HOME, but every file previously owned by `username` outside of his HOME will probably become unreadable/unwriteable since they now belong to a different user

 **FrenchBen** referenced this issue in **docker/machine** on Nov 30, 2015

Volumes access permission issue #2426 — Closed

pa-de-solminihac commented on Dec 1, 2015

@riquito I use it with a dedicated user, created in the `Dockerfile`:

```
adduser --no-create-home --disabled-login --gecos "" username --uid 1000
```

Therefore there are no files previously owned by `username`. So I guess there is no problem ;)

This was referenced on Dec 7, 2015

Trouble with volume permissions when container kicked off by plugin jenkinsci/docker-plugin#353 — Open

Permission denied - /var/jenkins_home/copy_reference_file.log jenkinsci/docker#177 — Closed

posita commented on Dec 14, 2015

FYI, `setfacl` does *not* work on volumes tracing back to OS X:

```
root@0da3c867240d:~# setfacl --default --recursive --modify u:500:rwx --modify g:500:rwx /opt/test
setfacl: /opt/test: Operation not supported
```

(In this case, `/opt/test` is hosted from OS X via Docker machine and Boot2Docker. See also boot2docker/boot2docker#581.)

 **posita** referenced this issue in **Parallels/docker-machine-parallels** on Dec 14, 2015

Feature request: support NFS as alternative means to mount /Users #29 — Closed

ava-dylang commented on Dec 15, 2015

@posita Are you using virtualbox to host your boottodocker image? If so that may actually be a limitation with the way virtualbox does shared folders. I've had a lot of trouble doing any kind of regular permeability though virtual box shared folder.

posita commented on Dec 15, 2015

@ava-dylang, that's a good point. The above is via Docker Machine with the Parallels driver, which uses Parallels' native shared folders implementation, which appears similarly limited. (See also this docker/machine#13 (comment) regarding a proposal for those types of environments.)

This was referenced on Dec 21, 2015

volume ownership based on -u user #18825 — Closed

User and group ID mappings docker-library/mongo#73 — Closed

System user and group stilliard/docker-pure-ftpd#15 — Open

JonathonReinhart commented on Jan 5

FWIW, I was having trouble with file ownership in my utility, Scuba.

I worked-around the issue by adding an "init" script that creates a user in the container with the same UID/GID as the invoking user in the host. See JonathonReinhart/scuba#11 and JonathonReinhart/scuba#13.

Now files created in the container look exactly like they'd been created on the host.

Update: That caused problems (JonathonReinhart/scuba#22). I instead fixed it by generating my own `/etc/passwd` and friends, based on the uid/gid of the host user, and injected it into the container (see JonathonReinhart/scuba#24).

JonathonReinhart referenced this issue on Jan 8

Add option to `docker run` to set umask in container #19189 Open

starkovv commented on Jan 9

+1

To make it more maintainable and secure when host runs in multi-tenant mode.

DJviolin commented on Jan 15

I had permission issues with such Wordpress installs, when you share the entire Wordpress from the host to the container with a volume.

In my stack, I have a base image, which is a debian with my basic modifications and every other image will be built from this image.

In the base image, I have this part:

```
### Start of Nginx WEBSERVER setup
RUN mkdir -p /var/www
# Modify www-data user and set UID, GID to 500
# https://muffinresearch.co.uk/linux-changing-uids-and-gids-for-user/
RUN groupmod -g 500 www-data \
    && usermod -u 500 www-data \
    #&& `find / -user 33 -exec chown -h 500 {} \;` \
    #&& `find / -group 33 -exec chgrp -h 500 {} \;` \
    && usermod -g 500 www-data \
    && chown -R www-data:www-data /var/www \
    && chmod g+s /var/www
### End of Nginx WEBSERVER setup
```

`www-data` is not created by php or nginx installs. It is a default defined user/group in Debian and maybe other distros. Some PHP, Nginx installs suggest use this user in their config files.

If your host user's UID/GID is 500 (most default linux first non-root user's UID/GID is 500 or 1000), than this script change the `www-data` user's UID/GID to 500 from 33. This way if you share anything from the host, Docker thinks those files and folders owned by `www-data` user!

In your PHP-FPM setting files set the user and group to `www-data` also.

In your nginx Dockerfile, you also have to set this:

```
# Allow Nginx to access /var/run/php-fpm/php-fpm.sock
RUN usermod -aG www-data nginx
```

This way `nginx` user can access the files owned by `www-data` (you can define the nginx's user name in the nginx config files).

After this hack, my Wordpress install not have ANY permission issues! All files resides on the host + updating Wordpress works flawlessly!

jantman commented on Jan 16

@DJviolin I'm not sure that an issue like this is the proper place for a Wordpress tutorial, and I hesitate to expound on it here. However, for the hapless newbies who might somehow stumble on this:

1. What you posted makes a lot of assumptions about installation - i.e. UID/GID values, user and group names, etc. It's not terribly portable across distributions.
2. More importantly, giving world-writable (0777) access to your entire Wordpress installation is extremely dangerous, and explicitly recommended against in the official WP documentation. Moreover, with all of this mounted from the host, you've just allowed arbitrary uploads from the Internet to write to the *host* filesystem.

DJviolin commented on Jan 16

@jantman Sorry, my bad. Looks like the first hack solves the permission issues as well. There is no need to change any default permission.

Obviously everyone need to find out their host user UID:GID and change the configuration according to this. Plus connect with an already existing or new user in the quest. Mine is a working example for a living stack.

Until docker not giving a solution for this problem, the assumptions will stay.

My solution is the same as **@JonathonReinhart** or **@pa-de-solminihac** ones.

This was referenced on Jan 29

How can we go inside individual container? itzg/dockerfiles#56 Closed

uid and guid inside docker volumes kvaps/docker-letsencrypt-webroot#2 Closed

Mounted datafile is not getting picked up by the database MaksymBilenko/docker-oracle-xe-11g#11 Closed

calavera commented on Mar 3

You can do this by creating local volumes, it just landed in master #20262.

Closing this as fixed.

calavera closed this on Mar 3

keywan-ghadami commented on Mar 8

Hi i also had to make an workaround in my dockerfile:

```
...
COPY start.sh /root/start.sh
CMD /root/start.sh
```

and then in start.sh

```
usermod -u $USER_ID www-data
exec php-fpm
```

As $USER_ID can be injected as environment parameter it is possible to use that docker image without modification.

Anyway i think there should be more built in functionality for that kind of stuff and am wondering how to use that local volumes suggested by **@calavera** , can someone provide an example?

sourcejedi commented on Mar 14

@keywan-ghadami I was puzzling over local volumes too. The problem is you have to create the filesystem first (or use tmpfs). So it doesn't use /var/lib/docker. In fact you can't use a sub-directory of *any* existing FS because bind mounts don't support the `uid` option. I don't know what local volumes are for; you could equally create and mount the FS yourself and then use a normal host volume (`-v my-created-fs:container-mount-point`).

 **ivotron** referenced this issue in **systemslab/srl-roles** on Mar 15

New model for checking permissions #3 Closed

alvinchevolleaux commented on Mar 20

I'm quite late to this thread but has this issue been resolved? It's not 100% clear from all the different issues referenced here. **@brthor** seems to have the best solution from what I can see but involves command line options that could more easily be just added into the Dockerfile and done behind the scenes.

For me, hard-coding UID and GID is a bad idea for portability between environments and I don't see the need to add command line arguments every time I boot up a container or build an image when it could be a simple option in the Dockerfile. Could there not be a simple option in either the Dockerfile or perhaps through docker-compose.yml where you can specify some sort of "map uid from host" option?

There may be a good solution out there already but I can't really figure this out from the documentation or this thread.

brthor commented on Mar 21

Thanks **@alvinchevolleaux** I can confirm we've been using the solution I posted above in the CI for https://github.com/dotnet/cli successfully for months now.

Do recommend it!

👍 2

alvinchevolleaux commented on Mar 21

@brthor Yep it's what I have gone with, I added `export USER_ID=$(id -u)` to my .bash_profile so it's all automatic for my various environments. Many thanks.

EronWright commented on Mar 24

As discussed above, when sharing content from your home folder into a container, add a user account with the same UID as yours. **Here's a trick to cope if your UID isn't 1000.** I assume that each user builds his own docker image from the Dockerfile.

Your `Dockerfile` should contain:

```
RUN useradd --uid 1000 -m vagrant
USER vagrant
```

The constant `1000` is substituted for your actual UID using a *git filter*. Run the following on your host:

```
git config filter.uidfix.smudge "sed s/1000/$UID/g"
git config filter.uidfix.clean "sed s/$UID/1000/g"
```

Finally, add a `.gitattributes` file to apply the git filter:

```
Dockerfile filter=uidfix
```

This works by replacing `1000` with your actual UID when you checkout the Dockerfile by *smudging* the file. Git will *clean* the file (putting back `1000`) when you commit. Any commands run in the container run as `vagrant` user with the correct UID.

👍 4

 **thomaszbz** referenced this issue in **thomaszbz/native-dockerfiles-typo3** on Mar 24

Apply git smudging concept #29 Open

keywan-ghadami commented on Mar 30

Seams this ticket should be reopend as all solutions provided here are only workarounds
I found a good documented and more complete solution in https://github.com/rocker-org/rocker/blob/master/rstudio/userconf.sh
It uses runtime arguments which makes it possible to use prebuild images which is not possible with the git filter

👍 1

docbill commented on Mar 31

@calavera I do not understand your solution.

e.g.

```
$ mkdir /tmp/test
$ sudo docker volume create --name=test -d local --opt type=nfs --opt device=/tmp/test:/data --opt
$ sudo docker run -t -i -v test:/tmp fedora bash -i
[.. $] touch /tmp/foo.txt
[.. $] exit
$ ls -la /tmp/test
```

When I look in the /tmp/test directory there are no files. It does not seem as if the /tmp/test folder is even being mounting in the container... Rather it is creating a container volume. Which is not really want I want.

I can find no documentation that tells me what are valid --opt options. So I am really guessing as to how this is suppose to work.

docbill commented on Mar 31

Please reopen this issue. The new local driver does not seem to at all address the issue of mounting local host directories with selected user id's.

👍 9

czerasz added a commit to czerasz/docker-fleetctl that referenced this issue on Apr 1

`Add git filter ...` `8e95a7d`

This was referenced on Apr 7

Data-only containers obsolete with docker 1.9.0? #17798 — Closed

Cannout mount container volumes mkobit/docker-nifi#5 — Open

Errors on file creation MaksymBilenko/docker-oracle-xe-11g#14 — Closed

jcfr added a commit to jcfr/docker-aosp that referenced this issue on Apr 20

`utils/aosp: Update script and Dockerfile to work with any host user u… ...` `b41731e`

jcfr added a commit to jcfr/docker-aosp that referenced this issue on Apr 20

`utils/aosp: Update script and Dockerfile to work with any host user u… ...` `6b55800`

jcfr referenced this issue in **kylemanna/docker-aosp** on Apr 20

Update `Dockerfile` and `utils/aosp` to work with arbitrary uid gid #11 — Merged

jcfr added a commit to jcfr/docker-aosp that referenced this issue on Apr 20

`utils/aosp: Update script and Dockerfile to work with any host user u… ...` `91ae4a8`

brthor referenced this issue in dotnet/cli on May 3

outdated packages on apt-get #2583 Open

brthor commented on May 3

@calavera
Can you please comment on how we can solve the issue of files being dropped as root with local host volumes? I have been seeing this issue crop up frequently, especially when using docker for CI scenarios.

Looking at the linked PR I don't see anything immediately obvious, but I'm sure I'm missing something 😄

docbill commented on May 4

I have been using variations of this workaround in my containers, such as docbill/docker-force . However, it occurs to me a cleaner solution is a container that is only responsible for doing the squashroot...

ktosiek commented on May 4

Using something like bindfs for local should help. You can run it with -o map=$(id -u)/root:@$(id -g)/**@root** (assuming no user namespace) and you should see files as yours outside of the container while they are owned by root inside of it.

👍 4

This was referenced on May 4

linuxbrew inside a docker container under root sjackman/docker-linuxbrew#7 Open

File save permissions linuxserver/dockergui#6 Open

JonathonReinhart commented on May 11

@ktosiek Thanks, that sounds like it has potential. Could you post a complete example `docker run` command or console session?

👍 4

Crafter6432 referenced this issue in extra84/mumble-container on May 23

cp: can't create '/data/murmur.ini': Permission denied #1 Closed

quinncomendant commented on Jul 6

Could someone please post a status update on this issue: was it closed as "won't fix" or has this actually been implemented? The thread is quite long, so please forgive me as can't see from a glance what the resolution is.

👍 1

LK4D4 commented on Jul 6

@quinncomendant see #7198 (comment)
Report if it doesn't work.

quinncomendant commented on Jul 6

Thanks @LK4D4.

To summarize, the issue was closed with a fix that allows mount opts for the `local` volume driver.

As far as I can tell, there are no options for setting the user:group of a *host-directory mounted as a volume*

(please correct me if that option exists—it's what I came here to learn).

😕 1

thaJeztah commented on Jul 6 — Docker member

@quinncomendant correct; when bind-mounting a host-directory you want the container to use the files that are there as-is, which includes permissions. If you want to changes those permissions, you need to do so on the host, before bind-mounting them

👍 1

docbill commented on Jul 7

@quinncomendant Or in simplier terms this is a WON'T FIX.... Since the solution provided does not actually do anything to address the issue reported, nor is there plan to solve that issue.

...

boj commented on Jul 7 • edited

This does appear to be a WON'T FIX issue. It always seems like the *development* aspects of containerization continue to be overlooked and still require us to write hacks.

I specifically came across this issue because the developers I am supporting are running project file generating commands from inside a container and out to the host mounted volume (installing these tools on the host completely defeats the purpose of using a container, right?).

They want to quickly iterate on these files, yet as others in this thread have pointed out they get written out as the uid/gid the container is running as, which requires them to be chowned correctly on the host to be further manipulated (by an IDE for example).

There really needs to be a *docker specific* way to shim in a temporary uid/gid in these kinds of cases.

👍 5

JonathonReinhart commented on Jul 7

@boj This is the exact scenario that I ran into while developing Scuba. My current solution creates, during startup, a user in the container with the same uid/gid as the host user. It's not the simplest solution, but it does work well.

😕 1

boj commented on Jul 7 • edited

@JonathonReinhart Thanks, you gave me a little inspiration.

I ended up writing a script wrapper called from the host like this (they use django in this case):

```
# bin/manage.py
#!/bin/sh

docker run -v $(pwd):/usr/local/prj -it --entrypoint="/usr/bin/python3.4" -w /usr/local/prj -u $(id
```

👍 1

JonathonReinhart commented on Jul 7 • edited

@boj The potential problem with that solution (`-u $(id -u):$(id -g)`) is that there is no entry in

`/etc/passwd` or `/etc/group` for that uid/gid in the container: JonathonReinhart/scuba#11

boj commented on Jul 7

@JonathonReinhart Noted. In this particular case I only care about the files written out to the developer's host and that they have the same permissions as the host user.

No worries about what actually happens at runtime, this was all I really needed.

ibukanov commented on Jul 7

@JonathonReinhart

If the software in the container needs an entry in /etc/passwd or /etc/group, make those world-writable in the image and then add the necessary entries there on startup.

bamarni commented on Jul 7

So the solution mentioned by **@JonathonReinhart** (uid/gid mapping) solves this.

It seems like this is already supported in runc (https://github.com/opencontainers/runc/blob/8c9db3a7a5145f6b26c8051af319eee6f72c9ca8/libcontainer/configs/config.go#L19-24). In Docker there is the `userns-remap` config for the deamon, here we'd basically need to have it more fine-grained (container level instead of deamon level), is there any interest / plan to support this?

DJviolin commented 29 days ago

This `docker-compose.yml` doesn't work:

```
version: '2'

services:
  app:
    build: ./app
    container_name: myapp
    volumes:
      #- "../app:/root/www/myapp:rw"
      - myapp:/root/www/myapp:rw

volumes:
  myapp:
    driver: local
    driver_opts:
      o: uid=500
      device: ../app
```

Someone can tell me why? I following the official guidelines:
https://docs.docker.com/engine/reference/commandline/volume_create/#/driver-specific-options

Is it possible to attach named volumes from the host? With `driver_opts` you can define `uid` (and maybe `gid` ?).

DJviolin referenced this issue 28 days ago

Support mount opts for `local` volume driver #20262 Merged

lazyuser commented 25 days ago

+1

xiaods commented 22 days ago

+1

JonathonReinhart commented 20 days ago

@lazyuser @xiaods Please stop with the +1's. It accomplishes nothing other than spamming all of those --> participants.

@bamarni Yes, I think the new user-namespace feature can solve this, but as you said, it would need to be implemented per-container. The end result would be: A container is running as what it thinks is "root", but is actually the UID/GID passed on the `docker run` command line. Files would "come out" of the container owned by the appropriate user then.

👍 2

pa-de-solminihac commented 20 days ago

@lazyuser @xiaods @JonathonReinhart you should rather click the +1 button under the issue description

nalipaz commented 20 days ago

Or just click subscribe on the right if you are only wanting notifications...

bamarni commented 17 days ago

@JonathonReinhart : definitely, I've went through the doc again but unfortunately having the mapping daemon-wide instead of per-container was a conscious decision because of a limitation :

> Note: The single mapping per-daemon restriction is in place for now because Docker shares image layers from its local cache across all containers running on the engine instance. Since file ownership must be the same for all containers sharing the same layer content, the decision was made to map the file ownership on docker pull to the daemon's user and group mappings so that there is no delay for running containers once the content is downloaded. This design preserves the same performance for docker pull, docker push, and container startup as users expect with user namespaces disabled.

(https://docs.docker.com/engine/reference/commandline/dockerd/#/daemon-user-namespace-options)

π **NicolasCARPi** referenced this issue in **elabftw/elabftw** 15 days ago

Update from 1.2.0p1 to 1.2.4 on Ubuntu 14.04 LTS Server with PHP 5.5 #276 Closed

lazyuser commented 11 days ago

Dear **@JonathonReinhart**, **@pa-de-solminihac** and **@nalipaz** ,
Thank you for your effort to educate me and others not to leave comments unrelated to the issue by doing just that! FYI, Github does not allow searching for issues one is subscribed to without at least commenting on them. For more information see isaacs/github#283. Ironically Github's issue is almost the same age as Docker's one, and both seems to be prioritized similarly.

To all, sorry for spamming. The first time it was a workaround for the aforementioned github bug, and this time I just could not resist giving the irony of the situation.

👍 1