Title: Docker volume - need permissions to write to database

Post Body:

I'm working on a flask server (in a virtualenv with python 3.5) which is used as a REST API (only for development as suggested for flask). In the beginning it connects to a local sqlite database and it will commit any db changes as soon as possible. Now I wanted to run everything in a docker container and I was wondering how I can access the database because the sqlite file is located in the container.

So I created a volume in a docker-compose file which points to the dockerfile building the application.

Dockerfile:

```
FROM python:latest ENV HOME /home/parkrep WORKDIR $HOME ADD requirements.txt $HOME/requirements.txt RUN pip install -r require
```

.dockerignore

```
__pycache__ venv .gitignore .dockerignore README.md Dockerfile docker-compose.yml
```

docker-compose.yml

```
version: '2' services:   parkrep:     build:       context: ./       dockerfile: Dockerfile     volumes:       - ./output:/hom
```

If I run `docker-compose up` I get the following

```
parkrep_1  |   File '/home/parkrep/db_connector.py', line 45, in _connect_db parkrep_1  |     self._connection = sqlite3.conne
```

If I create the database at output/reports.db and start docker-compose again, it returns the following error:

```
parkrep_1  | sqlite3.OperationalError: attempt to write a readonly database
```

So obviously I don't have permissions to write to the file. I tested this behavior by writing to a test file which is mounted like this:

```
... volumes:   - ./output:/home/parkrep/output   - ./test.txt:/home/parkrep/text.txt command: bash -c 'echo 'hallo' > test.txt
```

Error message:

```
parkrep_1  | bash: text.txt: Permission denied
```

Let's see who owns this file:

```
parkrep_1  | drwxr-xr-x 7 root root 4.0K Dec 19 10:45 . parkrep_1  | -rw-rw-r-- 1 root root  143 Dec 12 15:08 config.yaml park
```

It turns out that there is no user 4262 in the container but on the host machine my user account has this id. So I think I know what the problem is now, but I have no clue how to get access to these files. I tried adding ':rw' to the volume definitions but I still don't have write permissions. How can I tell docker to not change the file/directory owner if a volume is defined.

I'm thinking of a problem with my local volume driver, but maybe someone else already had this problem and can tell me how to configure my image to get the required permissions.

Greetings, Thomas

```
docker info
```

```
     Containers: 1  Running: 0  Paused: 0  Stopped: 1 Images: 19 Server Version: 1.12.5 Storage Driver: aufs  Root Dir: /var/l
```

```
docker-compose -v
```

```
docker-compose version 1.9.0, build 2585387
```

```
lsb_release -a
```

```
No LSB modules are available. Distributor ID: Ubuntu Description:    Ubuntu 16.04.1 LTS Release:    16.04 Codename:   xenial
```

Accepted Answer:

Hot fix

I fixed the problem by giving writer permission to everyone.

```
mkdir output touch output/reports.db output/database.log chmod a+rw output output/*
```

This will give the user on the host machine and the root user in the docker machine permissions, no matter who owns the files. This is just a dirty fix, because I had to hurry up. Any process could access the and edit/delete the files. It would be better If only the docker user get's writing permission, but I couldn't give writer permission to the root user on the host machine.

Better approach

In this post they're using another user (www-data) in the container. After building the image they get the id of the user and replace the current file owner with this id. If you start the container as this user (www-data), the mount will copy the files with permissions and owner information, so the user can read and write to these files.

It would be a more secure way because you make sure that only the docker user can change the database/files. Because I couldn't make this work for me (seems to not work for root user with id=0), but I wanted to point out, that there is a better solution.

If you only need the data in the end after the docker stopped you might have a look at docker cp.

Highest Rated Answer: None