Title: How can I link an image created volume with a docker-compose specified named volume?

Post Body:

I have been trying to user docker-compose to spin up a postgres container container, with a single, persisted named volume.

The goal is to have different postgres containers share the same persisted data (not concurrently!) - one container dies or is killed, another takes it place without losing previously persisted data.

As I understand 'named volumes' are supposed to replace 'Data Volume Containers'. However, so far either one of two things happen:

1. The postgres container fails to start up, with error message 'ERROR: Container command not found or does not exist.'
2. I achieve persistance for only that specific container. If it is stopped and removed and another container started, we start with a blank slate.

SO, as far as I understand, the postgres image does create it's own volume, which is of course bound to that specific container. Which would be fine, if I could just get THAT volume aliased or linked or something with the named volume.

Current incarnation of docker-compose.yml:

```
version: '2' services:  db:    image: postgres    restart: allways    volumes:      - myappdb:/var/lib/postgresql/data/    env
```

Am I doing something stupidly wrong, or attempting something that is simply not supported?

- Docker version 1.10.3, build 20f81dd
- docker-compose version 1.6.0, build d99cad6

Accepted Answer: None

Highest Rated Answer:

Ok, after a lot of trial and error, things are now working as they should (meaning I am able to run `docker-compose down` and then `docker-compose up` and my data is in the state where it was left with the `down` command).

In general, a few things:

1. Don't use the PGDATA environment option with the official postgres image
2. If using spring boot (like I was), and docker compose (as I was) and passing `environment` options to a service linked to your database container, do not wrap a profile name in double quotes. It is passed as-is to the Spring as-is, resulting in a non-existing profile to be used as the active profile.

I had some subtle and strange things incorrectly configured initially, but I suspect the killer was point 2 above - it caused my app,when running in a container, to use in-mem H2 database instead of the linked container database. So everything functioned (almost) perfectly - until container shutdown. And, when running from IDE, against container DB (with ports exposed to host), all worked perfectly (including persistence), since the active profile parameter was correctly set in the IDE launcher (NO quotes!).

Live and learn I guess (but I do feel a LOT of egg on my face).