### **Docker Volumes**

#### Goals

- 1. Create a new volume in Docker
- 2. Run a postgres container with the volume mounted
- 3. Create a new database on the volume
- 4. Back up the volume

## Creating a Volume

Create a volume with the docker volume create command and check to see if it is there with the docker volume ls command:

```
$ docker volume create db-data
db-data
$ docker volume ls
DRIVER VOLUME NAME
local db-data
```



Docker volumes are stored in /var/lib/docker/volumes/<volume-name> . If you're not running Docker natively (in Linux) that means your volumes live in that folder in a virtual drive used by your Docker Linux VM. That makes them pretty tough to get to without using Docker!

# **■ Running PostgreSQL**

Now let's run a postgres container in the background with our volume mounted in <code>/var/lib/postgresql/data</code>, which is the path where the PostgreSQL database information is stored by default:

```
$ docker run -d --mount source=db-data,target=/var/lib/postgresql/data -e 🔪 💵 🛂
POSTGRES PASSWORD=changeme postgres
26618a503c688c19728c87190beb7497f68b6e7d350b09e3eae7f2f71b1c8738
$ docker ps 4
CONTAINER ID IMAGE
                          COMMAND
                                                   CREATED
                                                                    STATUS
                                                                                    PORTS \
NAMES
26618a503c68
               postgres
                          "docker-entrypoint.s..."
                                                   28 seconds ago
                                                                    Up 28 seconds
                                                                                    5432/tcp \
gifted_montalcini
```

- The \ character is a line continuation. It means that the text below it, starting with "POSTGRES\_PASSWORD", is actually on the same line but it couldn't fit on the screen.
- Here's an explanation of some of the options and arguments:
  - -d run in the background (daemon)
  - --mount source=db-data,target=/var/lib/postgresql/data mount the exercise4-db local volume in the /var/lib/postgresql/data directory of the container

**-e POSTGRES\_PASSWORD=changeme** set the environment variable fo the postgres password to changeme. PostgreSQL won't start without a password.

postgres run the postgres image

- **3** Your output may be slightly different if you don't have the postgres image available locally.
- This is just to confirm that it's running

You can also check the logs to see how things went with docker logs <CONTAINER ID> where <CONTAINER ID> is the id from your ps command.

## Creating a New Database

The easiest way to create a new database on our running container would be with the psql command. Fortunately the psql command is already on our container so we can exec it from there. Let's connect to our database and put some data in there:

• Here is an explanation of the options:

```
    exec run a command on an already running container
    -it run interactively
    <CONTAINER ID> your container id as listed in the ps output above
    psql run the psql command (command line SQL client)
```

- **--username=postgres** this argument is passed to the psql command, it says you want to connect as the default postgres user
- 2 These psql commands create a bit of data for us to use

# ■ Backing up the Database Directory

Now we will start another container, that uses the same volume as our postgres container. On the new container we will use the tar command to make a compressed archive the database directory:

- This is a long command, so lets take it argument-by-argument:
  - run start a new container and run a command on it
  - --rm don't archive this container when we are done, it's temporary
  - --volumes-from <CONTAINER ID> give us the same volumes at the same mountpoints as our running postgres container
  - -v "\$(pwd):/backup" bind mount the local working directory to /backup on the container we are creating ubuntu use the ubuntu base image
  - tar caf /backup/db-backup.tar.gz /var/lib/postgresql/data This is the command that is run on the container. It creates a compressed tar archive from the database directory and writes it to /backup/db-backup.tar.gz.
- 2 Here's our backup in the directory where we were working on the host machine.

#### Deliverables

Submit your db-backup.tar.gz file as your work for this exercise.

### Resources

- <u>Docker Documentation: Use volumes</u>
- <u>Docker Hub: postgres</u>