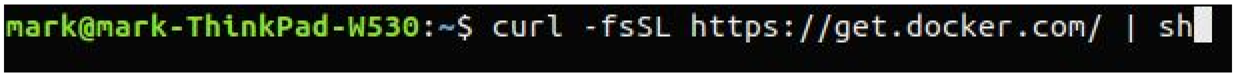
MariaDB Replication with Docker Containers

Introduction:

This guide will give a brief tutorial on how to setup MariaDB replication between two Docker containers. The steps here could be used for MariaDB installations on bare metal or on virtual machines, but there are some considerations that have been made for Docker. This includes networking, storage of configuration files, and storage of the database itself.

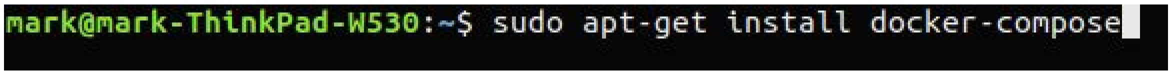
Requirements:

- Docker -
  - Windows and OS X users can download the Docker Toolbox from: https://www.docker.com/products/docker-toolbox. This will install virtualbox, a virtual machine hosting a minimal linux installation that will actually run docker, the docker command line terminal, docker compose, and kitematic: a GUI front end for managing docker containers.
  - Linux users can run the following command to install the latest version of docker:



```
mark@mark-ThinkPad-W530:~$ curl -fsSL https://get.docker.com/ | sh
```

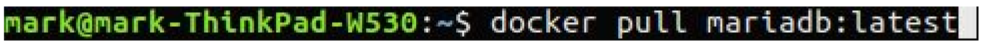
- Docker Compose -
  - Docker Compose will already be installed in the Docker Toolbox that comes with Windows and OS X.
  - Linux users will need to install Docker Compose via their system's package manager after having Docker installed.



```
mark@mark-ThinkPad-W530:~$ sudo apt-get install docker-compose
```

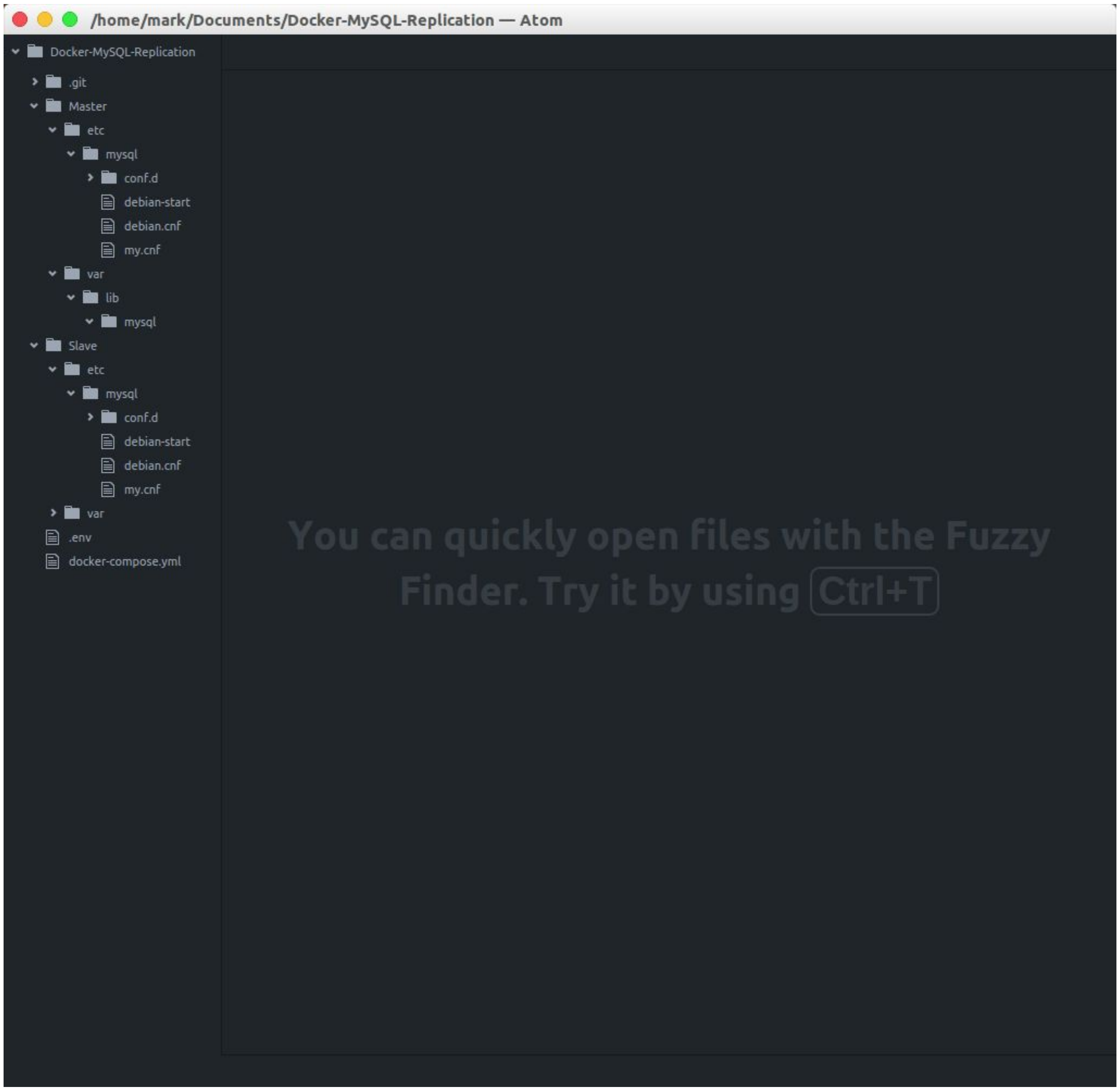
Setup:

I decided to use docker compose to spin up my containers, because it allows for granular control and definition of the containers through a yml “yaml” file. Compose is a tool for defining and running multi-container Docker applications. Then, using a single command, you create and start all the services from your yml file. Once both Docker and Docker Compose are installed, we are ready to get started. You will need to be connected to the internet to download the latest MariaDB Docker image. If you know you will be testing the replication somewhere without internet access, you can pull the image locally for use later.



```
mark@mark-ThinkPad-W530:~$ docker pull mariadb:latest
```

You will want to create a folder somewhere on your computer to work in. I recommend using the Atom text editor, as you can specify a folder as a working directory and navigate fluidly.



```
/home/mark/Documents/Docker-MySQL-Replication — Atom

Docker-MySQL-Replication
  .git
  Master
    etc
      mysql
        conf.d
        debian-start
        debian.cnf
        my.cnf
    var
      lib
        mysql
  Slave
    etc
      mysql
        conf.d
        debian-start
        debian.cnf
        my.cnf
    var
  .env
  docker-compose.yml
```

You can quickly open files with the Fuzzy Finder. Try it by using Ctrl+T

Networking:

The default method that docker uses for networking is to join containers to a bridge on the local system called docker0. Docker containers possess standard eth0 interfaces with an IP address in a private network range that is currently not in use by the host, typically 172.x.x.x. This default bridge mode will be sufficient for the docker containers to communicate with each other. There are methods for giving docker containers their own IP's on the network, but this proof of concept does not need those requirements.

Ports:

Storage:

We want to have control over the configuration files that MariaDB uses as well as the database files themselves. We can do this by using the volumes feature of docker, which will allow us to mount a folder from the local filesystem to a folder inside the filesystem of the docker container. This way, we can edit the configuration files before even starting up the container, or do so while the container is running. There are two directories in the docker containers that we want to mount to our local file system.

- /etc/mysql - Where configuration files are stored.
- /var/lib/mysql - Where the database files are stored.

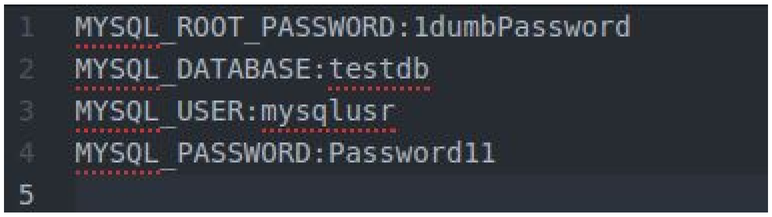
This can be achieved by using the -v flag with the docker run command, but we will be demonstrating the use of volumes in the docker compose yml file.

Environment Variables:

We are using the official MariaDB docker image from the docker hub. (https://hub.docker.com/_/mariadb/) This image is created and supported by the MariaDB community, and provides some helpful instruction on how to use it. To use this image, it is necessary to define some environment variables before the container is started:

- MYSQL_ROOT_PASSWORD
- MYSQL_DATABASE
- MYSQL_USER, MYSQL_PASSWORD
- MYSQL_ALLOW_EMPTY_PASSWORD

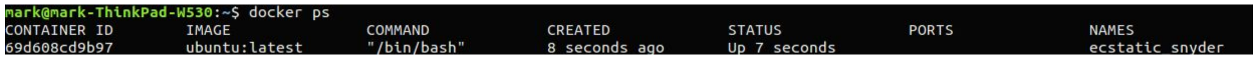
Luckily, docker allows us to define environment variables in a file, .env, and pass this to the container during initiation. Simply create a .env file in the root of your working directory and define your variables.



```
MYSQL_ROOT_PASSWORD:1dumbPassword
MYSQL_DATABASE:testdb
MYSQL_USER:mysqlusr
MYSQL_PASSWORD:Password11
```

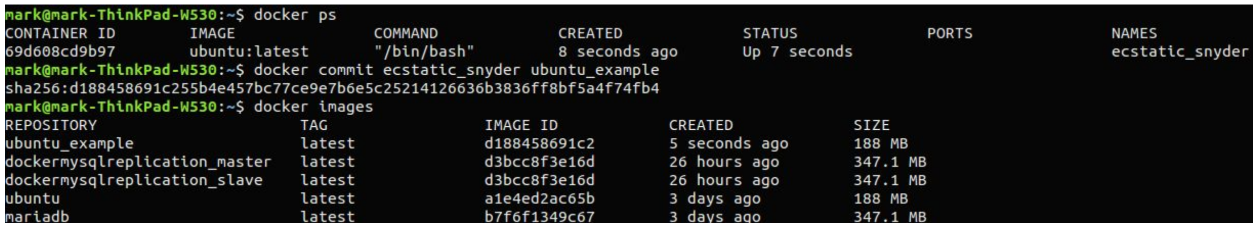
Persistence:

A running docker container can be saved or committed as it's own launchable image. To do this, you'll need to find the running docker containers on the system.



```
mark@mark-ThinkPad-W530:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
69d608cd9b97        ubuntu:latest       "/bin/bash"         8 seconds ago       Up 7 seconds                            ecstatic_snyder
```

Notice the name docker has given the container. Docker assigns each container a unique, yet psuedo-random name based on a list of fun verbs and nouns. You will be using this name to save the running docker container.



```
mark@mark-ThinkPad-W530:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
69d608cd9b97        ubuntu:latest       "/bin/bash"         8 seconds ago       Up 7 seconds                            ecstatic_snyder
mark@mark-ThinkPad-W530:~$ docker commit ecstatic_snyder ubuntu_example
sha256:d188458691c255b4e457bc77ce9e7b6e5c25214126636b3836ff8bf5a4f74fb4
mark@mark-ThinkPad-W530:~$ docker images
REPOSITORY                     TAG                 IMAGE ID            CREATED             SIZE
ubuntu_example                 latest              d188458691c2        5 seconds ago       188 MB
dockermysqlreplication_master  latest              d3bcc8f3e16d        26 hours ago        347.1 MB
dockermysqlreplication_slave   latest              d3bcc8f3e16d        26 hours ago        347.1 MB
ubuntu                         latest              a1e4ed2ac65b        3 days ago          188 MB
mariadb                        latest              b7f6f1349c67        3 days ago          347.1 MB
```

I've renamed the running container from ecstatic_synder to ubuntu_example. I can now run ubuntu_example anytime I want and it will be an exact copy from the moment I committed it. This was achieved by using the command: docker commit.

Consider when it would be best to commit a docker container. With our configuration, all possible changes to MariaDB and it's database will be stored on the host's filesystem, which could be used again and again to create more docker containers. It would be unnecessary to commit the container, if there were no changes elsewhere.

Master:

We finally get to start configuring our database instances for replication! The files in this repo were files I copied straight from the docker container running MariaDB. Open the my.cnf file located in Master/etc/mysql. Insert the following lines into the [mysqld] section:

```
[mysqld]
server_id=1
log-basename=master
log-bin
binlog-format=row
binlog-do-db=unixmen
[...]
```

Slave: