Training 3 - Docker compose

Loïc Delestra

During the last training course you used simple docker command and Dockerfiles. This week you will use multiple containers connected together with network links. You will also use docker-compose to create multiple containers application.

A/ Use link

Docker container can use network. In the previous training you have experiment the port mapping. Now you will use networking between containers. To make containers link your will need to use named containers and alias (cf slides and docker documentation)

1. Create a mysql container, from the official mysql repository, and name it "database".
2. Create a phpmyadmin, from the unofficial image `corbinu/docker-phpmyadmin`, name it "phpmyadmin' and link it to"database" with the alias "mysql".
   You can found pretty helpful documentation in the corbinu docker-phpmyadmin docker hub page.

3. Use `docker exec` to open an interactive shell into the phpmyadmin running container. Display the file /etc/hosts. What is the ip address of your mysql container?
4. Login to the phpmyadmin page

B/ All in one file.

You can run and connect multiple containers manually in command line, but it's not very effective and not scalable. Imagine you doing that with a 5, 10 or 15 containers application.

Docker provide a tool for that: docker-compose

Check docker-compose is present in your system with "`docker-compose --version`". If not you can install it with:

```
curl -L https://github.com/docker/compose/releases/download/1.5.1/docker-compose-`uname -s`-`uname -m` > \
/usr/local/bin/docker-compose ;\
chmod +x /usr/local/bin/docker-compose
```

B.1

5. Write a Dockerfile to build an image "pinger" from ubuntu with the entrypoint "ping target"
6. Write a docker-compose.yml file with a mysql container named "mysql" and the "pinger" image who link with the "mysql" with the alias "target"
7. Run your multiple container application with one command and check that "pinger" in pinging the target container.

B.2

8. Into a new docker_sql folder, create a new docker-compose.yml file. Recreate the same architecture as the A/ part with a mysql and a phpmyadmin containers.

C/ DAMP

You certainly already know LAMP (Linux Apache Mysql PHP). You will recreate this architecture with three container:
- "apache" with apache server and php module.
- "mysql" for the database.
- "phpmyadmin" for the database administration.

9. Create a Dockerfile for your "apache" container (you can use a httpd base image).
10. Create a docker-compose file to build and run your **3 containers**
11. Your apache container must have a minimalistic web site who display a list of blog post stored in your mysql container.

- your blog post must have two fields.
  - id INT
  - content TEXT

ps: libapache2-mod-php5

D/ (bonus) github and dockerhub

12. If you doesnt have a github account already, create one.
13. In this github create a new repository named "my-site". This repository will contain a Dockerfile and project sources.
14. Create a dockerhub account.
15. Create a new "automated" repository named "my-site" and link it with your github repo.
16. Commit new content in your github repository. A new image will be build in your docker hub.
17. Use docker pull to update your image and run your container. Check its up to date.