Training 2 - Docker

Loïc Delestra

A/ First steps with Docker

For the first part of this training classe you will manipulate and understand how Docker images and containers works.

1. Use the `docker run` command to start a container from `hello-world` image.
2. Has you can see the `hello-world` container display messages in your terminal.
   - Start your container in background (in detache mode).
   - Use a log command to display messages from your previous container.
3. Display local images.
4. Use the run command to start a container from `debian` image. Use options to get an interactive shell and explore the container.
   - Display container's processus (into the container). Look at the PID.
   - From a terminal in your host machine, use 'docker ps' command to list running containers.
   - Come back in your container shell and exit it. You must 'Be back!' in your host terminal, now list running containers and after that list all containers (even stopped ones).
   - Has you can see docker keep all stopped containers. **Remove** all of it!
5. Use the run command to start a container from `ubuntu` image with no interactive shell but a ping on '127.0.0.1'.
6. Run a container from the same image but in background this time.
   - Use `docker ps` and `docker logs` to check container and ping are running.
   - Then execute another command to open an interactive shell into this running container.
   - When you got a shell into the container, display the list of processus. What's your ping command PID?
   - Exit from the container, and kill it!
7. Run a 'nginx' container. Nginx image provide a default page, in order to look at this page in your host browser you have to enable port mapping.
   - Run an nginx container in background with auto port mapping. Find ports used, with the `docker ps` command, and check the page in your host browser.
   - You must see a 'Welcome to nginx!' page.
   - Stop and remove the container.
   - Start again a nginx container in background but this time you will map the container port 80 to the host port 8080.
   - Check nginx container is running and the page accessible with 8080 port.

B/ Build a docker image

For this part you will add packages in a container and save it as a new image.

8. Run a container from the official `ubuntu` image in interactive mode.
9. Use apt-get to install nmap package into your container.
10. When your new package are installed, check they are working (nmap iutweb.u-clermont1.fr).

11. Exit your container and find it's ID with `docker ps -a`. Build a new image from your stopped container, name it /nmap with the tag '1.0'.
    - What is the command used?
12. List all your local docker images and check your new image are listed.
13. Run a container from your image <yourname>/nmap:1.0 in interactive mode and install vim.
14. Exit this container and create a new image <yourname>/nmap:1.1 from this.
15. Use your new nmap image to scan the server iutweb.u-clermont1.fr .
    - How many open port do you find?

C/ Docker Buildfile

In this part you will create a new nmap image but instead of manualy install nmap package you will use Dockerfile.

16. Create a new directory docker_nmap.
17. In this directory create a new Dockerfile and edit it to get:
    - Ubuntu 14.04 as base image.
    - Install nmap and vim in two separate instruction.
    - Define `nmap -help` has default command (CMD).
18. Build your image as <yourname>/nmap:1.2 . How many steps in this build execution?
19. Test your new image without arguments, just run it. You must see the nmap help message.
    - Now run again a container from your nmap image but override the defaut command by "nmap iutweb.u-clermont1.fr".
    - Run again but override the default command by "ping iutweb.u-clermont1.fr -c 2".
20. You have a nmap image but have to specify to use nmap and you can use every command instead. You will change that to use nmap every time.
    - Edit your Dockerfile to add nmap as 'ENTRYPOINT' and 'iutweb.u-clermont1.fr' as defaut command.
    - Build a new <yourname>/nmap:1.3 image. How many steps and time this build take? why ?
    - Test it without arguments. Your container must, by default, scan iutweb.u-clermont1.fr for open ports.
    - Run a container from this 1.3 image to scan isima.fr .

C/ Application images

In this part you must use a Dockerfile to create two applications image. You want to package a website into an image.

21. Look at https://hub.docker.com/ to find the official 'httpd' repository. The repository info page give you tips to create an image with your website pages.
    - Create a docker_mysite folder that will contain your Dockerfile and web site pages.
    - Create a <yourname>/my-site image from 'httpd' image, hosting your personnal pages.
    - By default the httpd image expose the port 80. You just have to make port mapping.
    - Run a container and test it.
22. You must create a container with the Spring framework website 'sagan'. The Spring website is a demonstration and open source site for the Spring framework. You can checkout the code from github: https://github.com/spring-io/sagan.git .
    - You can build a jar from the sagan project.
    - Your can use a java 8 official image from dockerHub.
    - Use your imagination... (or the spring documentation for Docker)