Docker Workshop Exercises

Ignacio Vidal - Abbas Sarvmeily

April 18, 2016

Introduction

This set of exercises is designed to get to know the basic usage of the docker client. If you are working from within a network, you may need to specify a custom *dns* configuration to the docker daemon by stopping and running it again with `service docker stop && docker -d --dns=IP_ADDRESS`.

The `feeg6003/scicomp:v1` docker image is already in the virtual machine to minimise any network failure during the workshop.

Exercise 1 Running a custom container

Go to `https://hub.docker.com/` and find the `feeg6003/scicomp:v1` docker image. Pull the image from the registry and create a bash container. You might:

- Create and run it with a single command.
- Give it a meaningful name using the `--name` switch.
- Check for `gcc` and the python interpreter. Check that those commands are not available from the host OS.
- Check the kernel version.
- Perform a stress check with the command `stress -c 4`. See that the host OS is using all the cores (You might not be able to see this if you are using a virtual machine).
- Open a new terminal window and stop the container from the host. Use `docker ps -a` to see your stopped container. Start it again.
- Use `attach` to connect to the container again.

Exercise 2 Data persistence

Each container manages its own data files. Others containers can not access it, even if the containers are launched using the same image. That is what is meant with *containerization*.

- Run a container from `feeg6003/scicomp:v1`.
- Create a text file with some funny joke inside.
- Install `cowsay` and make the cow say the joke from the file with `cat joke.txt|cowsay`.
- Run a second container, also using `feeg6003/scicomp`, and see that neither `cowsay` nor `joke.txt` are there.

Exercise 3 Acessing host data from a container

Connect the `/home/feeg6003-docker/data` directory in the host OS to the `/data` directory of a container running `bash`. Use the `feeg6003/scicomp:v1` image and run **two** different containers.

If you are not using the provided virtual machine, you can experiment with any directory you like.

From the second container, edit the `README.md` file. Check that the changes made in one container are reflected in the other.

Exercise 4 Recreate the `feeg6003/scicomp` image using commits

Run the `ubuntu:14.04` image and install as many software packages as you want. Once finished, stop the container and commit the changes using

```
docker commit -m "Commit message" -a "Author name" containerID user/newimagename:tag
```

Exercise 5 Recreate the `feeg6003/scicomp` image using a Dockerfile

Install the same software packages as in the exercise 4, but using a Dockerfile. You can find a Dockerfile template in the blog.

Exercise 6 Running a daemonized container

The other main application of docker is to ship software in a fast a simple way. Look for the official *Owncloud* docker image in the public docker registry `https://hub.docker.com/` and run it as a daemon. You will set your own personal cloud with a single command, including a webserver, database server and PHP installation.

Note: In order to use a server running within a container, you must map the network ports with the switch `-p PORT_IN_CONTAINER:PORT_IN_HOST`

Exercise 7 Managing containers and images

Docker provides us with commands to see running containers and manage images. Familiarise yourself with `docker ps` and `docker image`. Delete every container and image generated while doing the exercise with `docker rm` and `docker rmi`.