Docker Practical Session

Celine Hernandez, Claire Toffano-Nioche, Thomas Denecker

28/06/2021

1 General information

Version:

- 0.1 (15/02/2017) Cours Université de Singapour
- 1.0 (28/08/2020) Cours IFB 2020
- 1.1 (30/08/2020) Version Markdown par Diane & Clémence
- 2.0 (28/06/2021) Cours IFB 2021

Contacts:

- Thomas Denecker Thomas.DENECKER@france-bioinformatique.fr (mailto:Thomas.DENECKER@france-bioinformatique.fr)
- Celine Hernandez celine.hernandez@i2bc.paris-saclay.fr (mailto:celine.hernandez@i2bc.paris-saclay.fr)
- Claire Toffano-Nioche claire.toffano-nioche@universite-paris-saclay.fr (mailto:claire.toffano-nioche@universite-paris-saclay.fr)

Websites

- Docker : https://www.docker.com/ (https://www.docker.com/)
- DockerHub : https://hub.docker.com/ (https://hub.docker.com/)

Resources

- Docker
    - Understanding Docker : [https://docs.docker.com/get-started/overview/ (https://docs.docker.com/get-started/overview/)]

    - Get started with Docker : [https://docs.docker.com/get-started/ (https://docs.docker.com/get-started/)]
    - More on Dockerfiles : [https://docs.docker.com/develop/develop-images/dockerfile_best-practices/ (https://docs.docker.com/develop/develop-images/dockerfile_best-practices/)]
- FASTQC
    - Official website : https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/)
    - Source code : https://github.com/s-andrews/FastQC (https://github.com/s-andrews/FastQC)

2 Part 1: using an available image

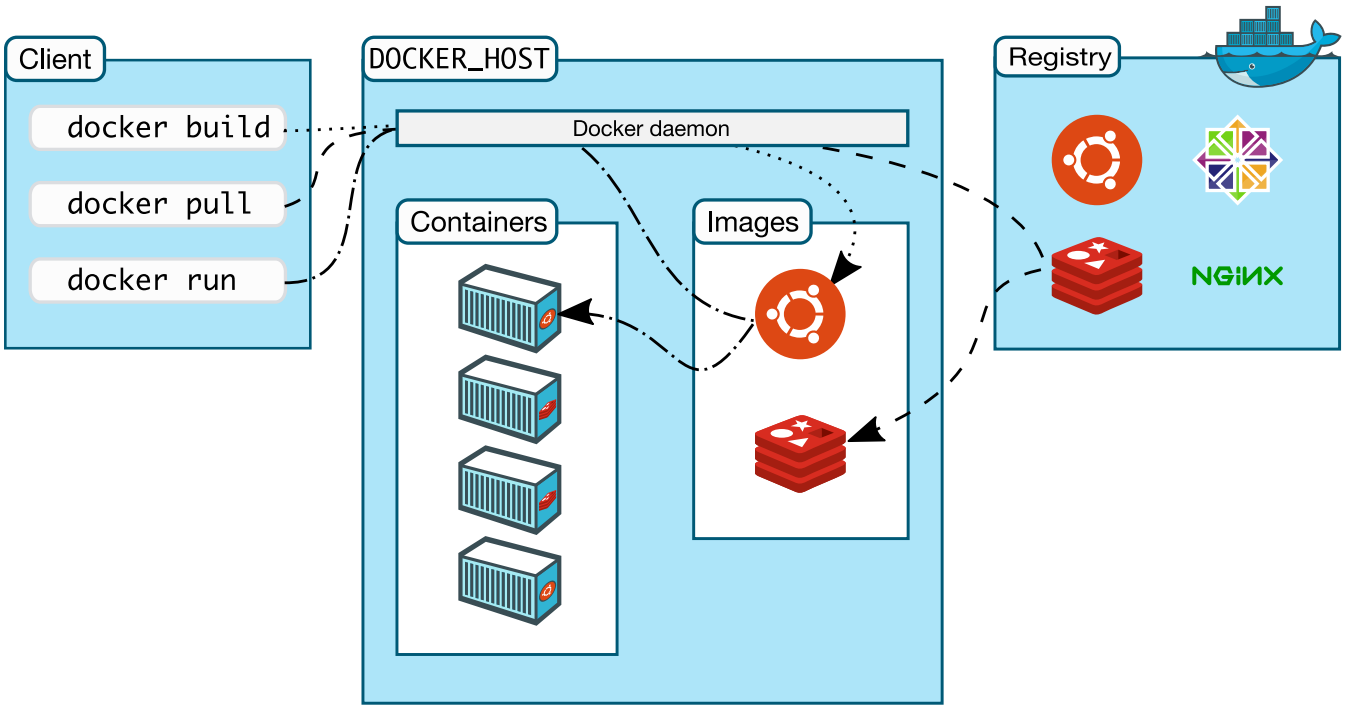
2.1 Introduction to Docker

> Goal: review the Docker structure

Docker is a command line-based software allowing to manipulate images and create application containers.

As presented in the course, Docker consists of two elements:

- a client, to receive commands from the user
- a server, to execute commands and manage images and containers



Docker architecture

Typing this command will give the Client and Server versions available on your computer.

```
docker version
```

2.2 Skeleton of a Docker command

> Goal : see the basic structure of a Docker command and how to display the related documentation.

Usage, options and a full list of available commands can be accessed through the command line in a terminal.

Type the following command

```
docker --help
```

The general usage of a Docker command line is as follows:

```
docker [OPTIONS] COMMAND [arg...]
```

If you consider the first line, a Docker command is made of four parts.

- The 'docker' keyword
- [OPTIONS] : optional parameters to run the client
- COMMAND : the name of a command to be run by the client
- [arg…] : parameters for the chosen command

A COMMAND is a specific task that can be executed by Docker. In this tutorial, we are particularly interested in the commands pull, run, rm/rmi, build and save. All commands have a dedicated documentation page.

Try to access the documentation of one specific Docker command

```
docker pull --help
```

Questions :

- How many arguments are absolutely required by the command 'docker pull' ?
- Do you remember what a registry is?

2.3 Download a pre-defined image available on the DockerHub

> Goal: On the Docker hub, find and download the 'FastQC' image made by the BioContainers initiative.

Exercise :

- In a web browser, navigate to the DockerHub : https://hub.docker.com/ (https://hub.docker.com/)
- In the top search bar, type : biocontainers

This organization made many images available to the scientific community. We will have a look at the available images.

Question :

- How many times was the samtools image downloaded ?

Exercise :

- Among the list of images, find FastQC.
- Click on the name to access the description page for this image.

Note that we could also have searched for biocontainers/fastqc in the search field.

Question :

- On the image description page, can you find the 'pull' command ?

Exercise :

- Copy the pull command
- Execute the command inside a terminal.

```
docker pull biocontainers/fastqc
```

```
Using default tag: latest
Error response from daemon: manifest for biocontainers/fastqc:latest not found: ma
nifest unknown: manifest unknown
```

You will get an error as this image has no default tag ("latest"). So we need to specify one in the command line.

Exercise :

- Go to the "Tags" tab and copy the pull command of version v0.11.9_cv8

```
docker pull biocontainers/fastqc:v0.11.9_cv8
```

Question :

- How many times do you see 'Pull complete' displayed ? Why ?

You can find more information on layers here: https://docs.docker.com/storage/storagedriver/#images-and-layers (https://docs.docker.com/storage/storagedriver/#images-and-layers)

Note that if you execute the pull command a second time, Docker will not re-download it. Instead, you will see a message about its status.

```
docker pull biocontainers/fastqc:v0.11.9_cv8
```

Exercise :

Now, to be sure that the image was correctly pulled, let's see the list of all available downloaded images inside our workspace.

```
docker images
```

Question :

- What is the size of the biocontainers/fastqc image ?

Optional exercise :

- Display the detailed description of the image.

```
docker inspect biocontainers/fastqc:v0.11.9_cv8
```

What is displayed on the terminal is a description of the image in JSON format. You can find more information on the Docker website: https://docs.docker.com/engine/reference/commandline/inspect/ (https://docs.docker.com/engine/reference/commandline/inspect/).

2.4 Perform a task using a pulled image

> Goal: Run a container from a pulled image.

Among the Docker commands, we will now use the 'run' command.

Question :

- What are the options and parameters of the 'run' command ?

```
docker run --help
```

As displayed in the terminal, the description of the command is 'Run a command in a new container'.

Question :

- What is the difference between an image and a container ?

See the Docker documentation https://docs.docker.com/get-started/overview/#docker-objects (https://docs.docker.com/get-started/overview/#docker-objects)

FastQC has been installed as a global program in the image we pulled. Consequently, it is directly accessible when interacting with the image.

Exercise :

Now, to run the application, execute the following command:

```
docker run biocontainers/fastqc:v0.11.9_cv8 fastqc --help
```

Question :

- What was displayed on the terminal ? Is it a message from Docker or from FastQC ?

> Congratulations! You just successfully downloaded and used your first Docker image !

2.5 Bind a local folder into a container

> Goal: map a local folder and run FastQC on a provided FastQ file.

Running FastQC without parameters was interesting as a demonstration of Docker's features. But if we want to really run FastQC, we also need to provide parameters and, most importantly, input files. Because of encapsulation, this functionality is not available by default. We need to explicitly map a local folder (a "volume") to a corresponding one inside the container (usually /home/), binding the two separated

“worlds” together.

To achieve this mapping, we use the –v option of the run command. Before trying it, you can check the command options once again.

```
docker run --help
```

Download and unzip the relevant files: https://zenodo.org/record/3997237 (https://zenodo.org/record/3997237) inside your current folder. You should have a folder named Data, containing fastq.gz files (among other files).

Exercise : find the paths to bind

- To bind our current folder to the /data/ folder located inside a container, we first need the absolute path of the current folder, obtained through the unix pwd command.

```
pwd
```

This path will be used in further commands through ${PWD}.

- Then we need to know where to bind it, i.e. what are the available Volumes of the image. Inspect the image to know which volume is available for binding.

```
docker inspect --format='{{.Config.Volumes}}' biocontainers/fastqc:v0.11.9_cv8
```

- Instead of running the fastqc command, we will now just list the content of the /data/ folder inside the container.

```
docker run biocontainers/fastqc:v0.11.9_cv8 ls /data
```

If nothing appears, it normal: the folder is empty and only serves as a “branching point”.

We now have the paths of the two folders we want to bind together.

Exercise : bind the two paths

To perform the folder mapping between the current folder and /data inside the image, the syntax is simple. The –v option takes one parameter made out of the two paths we want to map concatenated together, with the character : as separator.

```
docker run -v ${PWD}:/data/ biocontainers/fastqc:v0.11.9_cv8 ls /data/
```

Question :

- Is the displayed list the same as what is in your current folder?

```
ls
```

Finally, we can run FastQC on a FastQ file located in the Data folder. Change the name of the file to any of the provided files.

```
docker run -v ${PWD}:/data/ biocontainers/fastqc:v0.11.9_cv8 fastqc /data/Data/SRR
3105699_chr18.fastq.gz
```

2.6 Conclusion

> Congratulations, you know now how to run a command using a Docker image!

NB: DockerHub is the first but not the only registry available. You can also explore:

- Quay.io : https://quay.io/ (https://quay.io/)

```
docker pull quay.io/biocontainers/fastqc:0.11.9--hdfd78af_1
```

- GitHub Packages : https://docs.github.com/en/packages (https://docs.github.com/en/packages)

3 Part 2 : adapting an image to your needs

Sometimes, available images are just not enough. You need to build upon an existing image by adding a new functionality. In order to do so, there are multiple strategies: either save a new image from a modified one, or build it from a Dockerfile.

3.1 Pull a miniconda image

> Goal : search and pull a publicly available image where conda is available.

For this exercise you can either * Find on DockerHub the repository for MiniConda 3, by Continuum.io and get the pull command for version 4.9.2-alpine * Or, you can also search from the command line and build the pull command yourself.

```
docker search miniconda3
```

A table with five columns is now displayed. In the first column are the names of the available images. We will use these names to specify which image we are interested in. In the second column are descriptions for each image. The third column is the number of stars given by users to this particular image. The fourth and fifth columns indicate if a given image is an official release, and if it has been automatically created from a repository (GitHub, for instance).

```
docker pull continuumio/miniconda3:4.9.2
```

Now check that the image is present in your list.

```
docker images
```

Question :

- What is the size of the downloaded image? Compare that to the 780Mb required for a functional Debian Stretch installation (https://www.debian.org/releases/stretch/i386/ch02s05.html.en (https://www.debian.org/releases/stretch/i386/ch02s05.html.en)). What do you think?

3.2 Install FastQC in a container

> Goal: using Conda, install FastQC inside a container made out of a publicly available image

Remember the picture from the course with all the computers in a room? Let's "switch one on", and connect to it !

We will now start a terminal (bash) session inside a container based on the image we just downloaded.

```
docker run -ti --name myfastqc continuumio/miniconda3:4.9.2 /bin/bash
```

```
(base) root@5dfee6f14dff:/#
```

Note: Check the documentation or https://docs.docker.com/engine/reference/commandline/run/ (https://docs.docker.com/engine/reference/commandline/run/) for the meaning of options -t -i.

Normally, the first part of the terminal line should have changed to root@ followed by a series of letters and numbers. This series is the unique identifier for your container named *myfastqc*, which has been randomly generated at the creation of the container. You are now connected to a lightweight version of Debian with miniconda installed.

Next step is to run the command that will install **FastQC** using **conda** inside this container.

Here is the official page of the fastqc package on the Anaconda repository: https://anaconda.org/bioconda/fastqc (https://anaconda.org/bioconda/fastqc) We will use the bioconda channel to get FastQC as a Conda package. Type y when prompted "Proceed ([y]/n)?".

```
conda install -c bioconda fastqc
```

Let's run **FastQC** to check that it works, and finally exit the container:

```
fastqc --help
```

It's not the goal of this exercise, but if you had started this container with the -v option, you could have directly checked the quality of your FastQ files from inside the container. But don't do that now. Simply exit the session and the container.

```
exit
```

We can check the status of the latest container created:

```
docker ps -l
```

This will display a table with seven columns. In the column titled STATUS, you can check that we "Exited" the latest container created a certain time ago. Note that as we exited the container, it has now stopped. To use again the same picture, the computer is now "switched off".

NB: do not try to copy huge data files inside a container to work "locally". You would get very big containers. Instead, always use folder binding with the -v option.

3.3 Restart and detach a container

> Goal : learn how to re-use a container where you installed something

Exercise : restart the container

- Use the start command to restart the container created in the last exercise

```
docker start –ti myfastqc /bin/bash
```

- Go back to the container using the exec command instead of the run command. Is FastQC still there?

```
docker exec –ti myfastqc /bin/bash
```

NB: If you run the exec command after exiting the container, you will get an error.

```
Error response from daemon: Container 9aef6012878628ae297910e6aa6d30c8224fe42f522b
829ffa75d0254ec437a5 is not running
```

Question :

- What happens now when you exit the container? Is it stopped? Check with:

```
docker ps –l
```

In fact, the container keeps on running. This is because re-starting a container turns it into a “detached process” running in the background.

Alternatively, we could have added the -d option to the first docker run command, creating directly a detached container. Or we could have exited the container by pressing Ctrl-P followed by Ctrl-Q, instead of using ‘exit’.

- Finally, you can stop the container.

```
docker stop myfastqc
```

3.4 (Optional) Turn a container into an image, export and reload it

> Goal : saving our container into a stable image.

Images are stable objects. They can be pulled, deleted, pulled again, you will always get the same object. On the contrary, each container is unique as it’s the result of all the modifications and commands you entered while connected to it. If we would type again the ‘docker run’ command of the previous exercise, we would get a brand new container, **without FastQC** installed. So how do we turn a container into an image?

Exercise : saving a container as an image

If you want to save a given container and turn it into an image, you can perfom a “commit” of the container **myfastqc**. This will save it as an image named *test/myfastqc:1.0*.

```
docker commit myfastqc test/myfastqc:1.0
```

Exercise : export and re-load the image

Finally, you can export your newly created image as a separated file, re-loadable as needed. More information on the 'docker save' command is available here, or from the command line: https://docs.docker.com/engine/reference/commandline/save/ (https://docs.docker.com/engine/reference/commandline/save/)

```
docker save test/myfastqc:1.0 > ./savedfastqcimage.tar
```

This archive can be reloaded as needed. First delete the current image.

```
docker rmi test/myfastqc:1.0
```

Then, load the image from the archive.

```
docker load < ./savedfastqcimage.tar
```

You can now check that it has been correctly loaded using 'docker images'.

4 Part 3 : creating images from a configuration file (Dockerfile)

4.1 Create an image automatically

> Goal : use a Dockerfile to create the same image as in the previous part.

Creating a container ourselves was nice. But having to enter each installation command can be a bit tedious, especially if you have many commands to enter. In this part, we want to automate the installation process and directly saving the resulting object as a stable image.

All the steps that we manually did in the preceding exercise can be collected into a "recipe" called a Dockerfile, which is a simple text filewith commands. A Dockerfile can then be used to build an image on-demand.

Type the following command.

```
printf "FROM continuumio/miniconda3:4.9.2\nRUN conda install -c bioconda fastqc\n"
> Dockerfile
```

The Dockerfile created by the command should read:

```
FROM continuumio/miniconda3:4.9.2
RUN conda install -c bioconda fastqc
```

- **FROM** specifies the base image that should be used as a starting point to create our new image.
- **RUN** specifies any command that should be run for our purpose. In our case, we just need one command line.

You can find more information on what can be included in a Dockerfile at this URL: https://docs.docker.com /develop/develop-images/dockerfile_best-practices/ (https://docs.docker.com/develop/develop-images /dockerfile_best-practices/). Note that, for instance, it is usually a good practice to give also the name of a MAINTAINER, responsible for the writing and maintenance of the Dockerfile.

Now that we saw what is a Dockerfile, we will use it to build an image containing FastQC, and most importantly, save it as an image.

```
docker build –t test/myfastqcfile:1.0 .
```

Docker will execute one by one each instruction written inside the Dockerfile. We can now check that we have locally a new image called test/myfastqcfile with the tag 1.0.

```
docker images
```

Finally, we can run a simple command to check if FastQC is working and what is the installed version inside the image.

```
docker run test/myfastqcfile:1.0 fastqc –v
```

4.2 (Optional) Create an image based on RStudio

> Goal : create a Docker image containing RStudio and a set of R packages (CRAN or Bioconductor).

This is not a real exercise, more an example to illustrate where you can go from here with what you just learned.

As a data analyst, it can be difficult to keep a clean environment with stable packages and software. Docker provides a solution for that problem. A Dockerfile allows to write a recipe for an image with your favorite packages and RStudio. This image can be exported and archived for future reference.

Copy the following text into a file named Dockerfile.

```
#####################################################
#
# Dockerfile to use RStudio in Docker
# Based on Rocker/rstudio:4.0.2
#
#####################################################

# Set the base image
FROM bioconductor/bioconductor_docker:RELEASE_3_11

# Install needed R packages
# Don't forget to rebuild the image if this list changes

RUN R -e 'BiocManager::install("DESeq2")'

RUN install2.r --error --repos http://cran.rstudio.com/ \
    gplots \
    RColorBrewer

RUN R -e "installed.packages()[,c(3:4)]"
```

As in the previous exercise, build an image from the Dockerfile. Be patient, as the Rocker image is quite heavy to pull.

```
docker build -t analysis/test:1.0 . | tee docker_build_$(date '+%Y-%m-%d').log
```

And start a container.

```
docker run --rm -p 8787:8787 -e PASSWORD=LetMe1n -v ${PWD}:/home/rstudio -w /home/
rstudio analysis/test:1.0
```

Do not exit nor terminate the session (only when you're finished with the practical session). Open a browser and go to the URL: localhost:8787 User is: rstudio and password LetMe1n (see previous command).

This illustrates how to create your own workspace for your analyses, here with DESeq2 installed for RNA-seq analyses. The Rocker image used here is based on the server installation of RStudio. Rocker images are maintained by the community and exists in different versions. See :

- https://hub.docker.com/r/rocker/rstudio (https://hub.docker.com/r/rocker/rstudio) on the DockerHub.
- https://www.rocker-project.org/ (https://www.rocker-project.org/) for the Rocker project.

For Python users, see the Jupyter images available: https://hub.docker.com/u/jupyter (https://hub.docker.com/u/jupyter)

5 Conclusion

This is the end of the practical session. We hope you enjoyed it. Don't hesitate to ask any questions and feel free to contact us any time after the session!

6 List of commands

- Search the available versions of an image in the Docker registry:

```
docker search <keyword>
```

- Pulling an image

```
docker pull <name>:<version>
```

- Starting a container on a given image running a single command:

```
docker run -ti <image>:<version> <command>
```

- List all containers and their status

```
docker ps -l
```

- List all pulled images

```
docker images
```

- Removing one local container

```
docker rm <containerid>
```

- Removing one local image

```
docker rmi <name>:<version>
```

- Clean all containers

```
docker rm $(docker ps -aq)
```

- Clean all images (after cleaning the containers)

```
docker rmi $(docker images -aq)
```