Containers

A.Comte & J.Orjuela

Science Reproducible Training SG 2023

Introduction

What is a Container?

Containers are **packages of software that contain all of the necessary elements to run in any environment**.

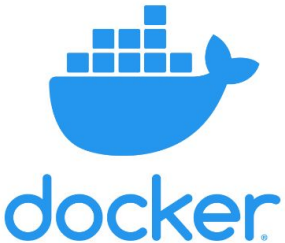
→ Software or code functions in exactly the same way regardless of where it's executed (mac/windows, cluster/local)

→ Containers are in many ways similar to virtual machines but more lightweight.

→ Containers can be used to allow others to reproduce a complete analysis

→ Containers can also be used to define software environments and settings for benchmarking studies

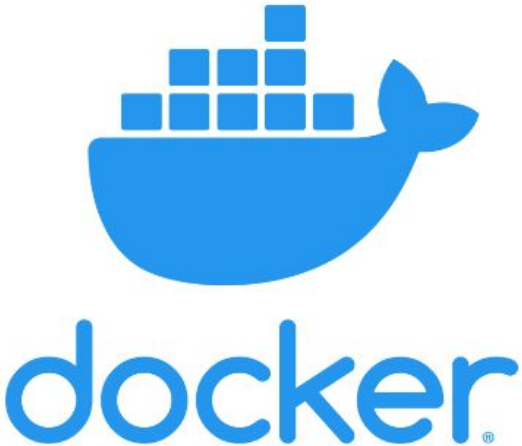
Making self-contained, distributable projects with



Singularity

The battle

	
runs as a daemon process with superuser privileges	runs as regular user
images stored centrally	image files that you can move around. No layers!
isolates the host and container file system by default	containers have access to host file system
well supported on Mac/Linux/Windows	limited support on Mac/Windows


docker

Standardized packaging for software and dependencies

Docker lets you create and run applications securely isolated in a container, packaged with all its dependencies and libraries.

Docker nomenclature:

A **Docker file** is a recipe used to build a Docker image

A **Docker image** is a standalone executable package of software

A **Docker container** is a standard unit of software run on the Docker Engine.

Docker Hub is an online service for sharing docker images

Setup Environment

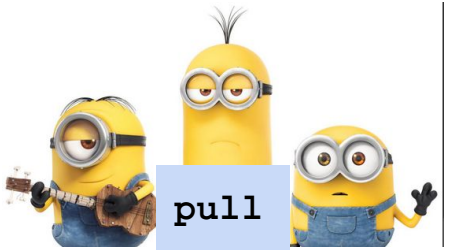
First let's create a dedicated folder for this tutorial:

```
mkdir -p  ~/training-reproducible-research-area/containers

cd ~/training-reproducible-research-area/containers

cp -r
~/training-reproducible-research-area/training_reproducible_research/tu
torials/containers/* .
```

The basics



Downloading containers

```
docker pull ubuntu:latest
docker image ls
```

Running containers

```
docker run ubuntu uname -a
```

Running interactively

```
docker run -ti ubuntu

# -t terminal connection btw shell
# -i interactif
# -v bind mount volume
# -w workdir inside the container
…
```

All dependances are into the containers

```
shell:
    """
    bowtie2-build tempfile intermediate/{wildcards.genome_id} > {log}
    """
```

nextflow
Snakemake

bowtie2-build is called directly from workflow rules

Containers inside scripts

```
mkdir -p $PWD/analysis
cd analysis

curl -o NCTC8325.fa.gz URL_to_file.fa.gz
gunzip -c NCTC8325.fa.gz > tempfile

docker run -v $(pwd)/analysis:/home
quay.io/biocontainers/bowtie2:2.5.0--py310h8d7afc0_0 bowtie2-build
/home/tempfile /home/NCTC832
```

Building a docker image

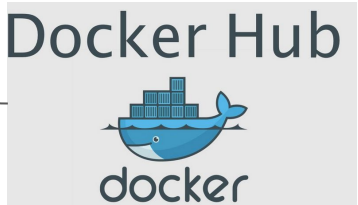
Format

Here is the format of the `Dockerfile` :

```
# Comment
INSTRUCTION arguments
```

FROM
LABEL
MAINTAINER
SHELL
WORKDIR
RUN
ENV
EXPOSE
CMD
COPY
…

FROM ubuntu:16.04



FROM my_docker_image:latest

\+ layers !!!

Understanding from Dockerfiles

```
FROM ubuntu:16.04

LABEL description = "Minimal image for the NBIS reproducible research course."
MAINTAINER "John Sundh" john.sundh@scilifelab.se

# Use bash as shell
SHELL ["/bin/bash", "-c"]

# Set workdir
WORKDIR /course

# Install necessary tools
RUN apt-get update && \
   apt-get install -y --no-install-recommends bzip2 \
                                              ca-certificates \
                                              curl \
                                              [...]
                                              unzip \
                                              wget \
                                              && apt-get clean
```

```
FROM ubuntu:16.04

LABEL description = "Minimal image for the NBIS reproducible research course."
MAINTAINER "John Sundh" john.sundh@scilifelab.se

# Use bash as shell
SHELL ["/bin/bash", "-c"]

# Set workdir
WORKDIR /course

# Install necessary tools
RUN apt-get update && \
    apt-get install -y --no-install-recommends bzip2 \
                                               ca-certificates \
                                               curl \
                                               [...]
                                               unzip \
                                               wget \
                                               && apt-get clean
```


```
# Install necessary tools
RUN apt-get update /!\

RUN  apt-get install -y
--no-install-recommends bzip2 \
                                               ca-certificates \
                                               curl \
                                               [...]
                                  pd               unzip \
                                               wget

RUN  apt-get clean
```

```
# Install Miniconda and add to PATH
RUN curl -L https://repo.continuum.io/miniconda/Miniconda3-4.7.12.1-Linux-x86_64.sh -O && \
    bash Miniconda3-4.7.12.1-Linux-x86_64.sh -bf -p /usr/miniconda3/ && \
    rm Miniconda3-4.7.12.1-Linux-x86_64.sh && \
    /usr/miniconda3/bin/conda clean -tipsy && \
    ln -s /usr/miniconda3/etc/profile.d/conda.sh /etc/profile.d/conda.sh && \
    echo ". /usr/miniconda3/etc/profile.d/conda.sh" >> ~/.bashrc && \
    echo "conda activate base" >> ~/.bashrc

# Add conda to PATH and set locale
ENV PATH="/usr/miniconda3/bin:${PATH}"
ENV LC_ALL en_US.UTF-8
ENV LC_LANG en_US.UTF-8

# Configure Conda channels and install Mamba
RUN conda config --add channels bioconda \
    && conda config --add channels conda-forge \
    && conda config --set channel_priority strict \
    && conda install mamba \
    && mamba clean --all
```

```
# Open port for running Jupyter Notebook
EXPOSE 8888

# Start Bash shell by default
CMD /bin/bash
```

And now it's your turn!

TP : Building from Dockerfiles

TP : Creating your own Dockerfile

Building from Dockerfiles

```
docker build -f Dockerfile_slim -t my_docker_image .

-f dockerfile_recipe_name
-t tag
. /this/path
```

Creating your own Dockerfile

1. Create the file `Dockerfile_conda`.

2. Set `FROM` to the image we just built.

3. Install the required packages with Conda. We could do this by adding `environment.yml` from the Conda tutorial, but here we do it directly as `RUN` commands. We need to add the conda-forge and bioconda channels with `conda config --add channels <channel_name>` and install `fastqc=0.11.9` and `sra-tools=2.10.1` with `conda install`. The packages will be installed to the default environment named `base` inside the container.

4. Add `run_qc.sh` to the image by using the `COPY` instruction. The syntax is `COPY source target`, so in our case simply `COPY run_qc.sh .` to copy to the work directory in the image.

5. Set the default command for the image to `bash run_qc.sh`, which will execute the shell script.

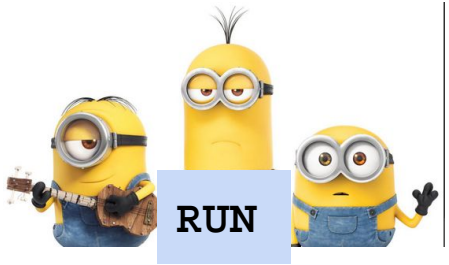
Creating your own Dockerfile

```
FROM my_docker_image:latest
RUN mamba install -n base fastqc=0.11.9
RUN mamba install -n base sra-tools=2.11.0
COPY run_qc.sh .
CMD bash run_qc.sh
```

Managing containers

```
docker run my_docker_conda

# -t terminal connection btw shell
# -i interactif
# -v bind mount volume
# -w workdir inside the container
…
```




```
docker run -d --rm --name my_container my_docker_conda
# -rm remove docker image when run is finished
# -d detached
```


```
docker containers ls --all

# -ls : list
# --all : show all the containers
```

check if container is running …

exec
ps

```
docker exec -it my_container_ID /bin/bash
```

```
docker ps
CONTAINER ID   IMAGE          COMMAND   CREATED          STATUS          PORTS   NAMES
a36ad8e57f0a   ubuntu:jammy   "bash"    15 minutes ago   Up 15 minutes           elated_haslett
```

```
docker exec a36ad8e57f0a echo "toto"
toto
docker exec elated_haslett echo "coucou"
coucou
```

Bind mounts

-v

```
docker run --rm -v HOST:DOCKER_PATH my_docker_conda
```

```
mkdir -p fastqc_results
docker run --rm -v $(pwd)/fastqc_results:/course/results/fastqc my_docker_conda

## /course/results/ is created by run_qc.sh
```

```
docker run -it --rm -v $(pwd):/course/ my_docker_conda /bin/bash
```

You can enter into the container and modify script *`run_qc.sh`* *and run it*
BUT
modifications in *`run_qc.sh`* *are not saved when you exit*

Distributing your images

[Docker Hub]

[Quay]

[Biocontainers]

[Rocker]

[Jupyter containers]

register into https://hub.docker.com/

Distributing your images

```
# login you (once)
docker login -u your_dockerhub_id
# build image:1.3.0 can be pushed into dockerhub
# … run it
docker run -it image:1.3.0
# and recovery image_ID (fd8a0825c9c9 in this example)
docker ps
# commit !
docker commit -m "First push" -a "Julie Orjuela" fd8a0825c9c9
your_dockerhub_id/image:1.3.0
# and push
docker push your_dockerhub_id/image:1.3.0
# Now image can be download/pull for collaborators from dockerhub
docker pull your_dockerhub_id/image:1.3.0
```

Packaging the case study

MRSA case

Multiresistant bacteria MRSA. Here we will build and run a Docker container that contains all the work we've done so far.

- We've set up a GitHub repository for version control and for hosting our project.
- We've defined a Conda environment that specifies the packages we're depending on in the project.
- We've constructed a Snakemake workflow that performs the data analysis and keeps track of files and parameters.
- We've written a R Markdown document that takes the results from the Snakemake workflow and summarizes them in a report

```
# explore it :
ls training_reproducible_research/tutorials/containers
drwxrwxr-x 2 orjuela orjuela 4096 juin   5 10:34 code
-rw-rw-r-- 1 orjuela orjuela 1569 juin   5 10:34 config.yml
-rw-rw-r-- 1 orjuela orjuela 2292 juin   5 10:34 Dockerfile
-rw-rw-r-- 1 orjuela orjuela 1824 juin   5 10:34 Dockerfile_slim
-rw-rw-r-- 1 orjuela orjuela  515 juin   5 10:34 environment.yml
-rw-rw-r-- 1 orjuela orjuela  765 juin   5 10:34 run_qc.sh
-rw-rw-r-- 1 orjuela orjuela 6577 juin   5 10:34 Snakefile

# Look the Dockerfile
it install the conda packages listed in environment.yml
CMD => it will run the whole Snakemake workflow
```

Dockerfile

```
# Set workdir
WORKDIR /course

# Add project files
COPY environment.yml Snakefile config.yml ./
COPY code ./code/

# Install conda environment
# Configure Conda channels and install Mamba
RUN conda config --add channels bioconda \
        && conda config --add channels conda-forge \
        && conda install mamba \
        && mamba env update -n base -f environment.yml \
        && mamba install -c conda-forge jupyter \
        && conda clean --all

CMD snakemake -rp -c 1 --configfile config.yml
```

MRSA case

```
# Build it
docker build -t my_docker_project -f Dockerfile .

# OR pull it if they are in dockerhub
docker pull nbisweden/workshop-reproducible-research

# Validate
docker image ls
```

Singularity

Singularity Nomenclature

A **Singularity definition file (.def)** is a recipe used to build a Singularity image

A **Singularity image (.sif)** is the builded container

Singularity Nomenclature

	Docker	Singularity
Base	**FROM** ubuntu:16.04	**Bootstrap**: docker **From**: ubuntu:16.04
people	**MAINTAINER** "John Sundh" john.sundh@scilifelab.se	
description	**LABEL** description = "Minimal image for the NBIS reproducible research course."	**%labels** "Minimal image for the NBIS reproducible research course."
workdir	**WORKDIR** /course	**%post** mkdir /course cd /course

Singularity Nomenclature

	Docker	Singularity
Installations	**RUN** apt-get update && \ apt-get install -y --no-install-recommends bzip2 \ ca-certificates \ curl \ [...] \ vim \ && apt-get clean	**%post** apt-get update -y && \ apt-get install -y --no-install-recommends bzip2 \ ca-certificates \ curl \ [...] \ vim \ && apt-get clean -y
set environment	**ENV** PATH="/usr/miniconda3/bin:${PATH}" **ENV** LC_ALL en_US.UTF-8 **ENV** LC_LANG en_US.UTF-8	**%environment** PATH="/usr/miniconda3/bin:$PATH" LC_ALL=en_US.UTF-8 LC_LANG=en_US.UTF-8
execution	**CMD** /bin/bash	**%startscript** /bin/bash

Singularity commands

```
# Build a container
singularity build Singularity.sif Singularity.def

# Run a command from a container:
singularity run Singularity.sif echo toto

# Use a container interactively:
singularity shell Singularity.sif

#Get a singularity container from a docker image:
singularity pull mrsa_proj.sif
docker://nbisweden/workshop-reproducible-research
```

What/where to use ?

	Docker	Singularity
Limitations	To maintain the Docker engine, Docker daemon needs to be run in the background Docker daemon needs root privileges, which could potentially be a security concern.	Smaller community Not good for windows (a lot of dependencies needed)
Strengths	Docker is suitable for DevOps engineering Docker Hub provides a great number of pre-built Docker images, which is convenient to meet the needs of many applications. Big community Containers strictly isolated from the system	More suitable for scientific users Singularity containers are preferred when running applications in HPC systems. Singularity containers can be run without sudo. Unprivileged users can also use –remote or –fakeroot features to build Singularity containers. Singularity can convert Docker containers to Singularity

Warning!

Une image permet de faire de la reproductibilité **MAIS** un fichier de définition n'est pas forcément reproductible.

Si les versions ne sont pas fixées dans le fichier de définition, l'image reconstruite plusieurs mois/années plus tard sera différente.

En 1955:
Citroën DS:latest

En 2019:
Citroën DS:latest

En 2023:
Citroën DS:1955