Reproducible education and research: Introduction to Docker

Jan Aerts

KU LEUVEN STADIUS

:: Why? ::

1. Teaching

[I0U19A] Management of Large-Scale Omics Data

- Lambda architecture
- Data storage: key/value stores, graph databases (neo4j, ...), document-oriented databases (mongodb, ...)
- Data processing: hadoop, spark

Need running (linux) system for demo and exercises.

Previous years:

- Amazon AWS EC2
- Grant from Amazon 2013-2015: $3,000
- Heavily annotated setup script
  - Install software
  - Create user accounts
  - Allow remote logins
  - ...

In 2016:

- Grant from Amazon to run server: $75 (seventy-five)
- Fed up with spending time debugging software installation

2. Software distribution

NGS Logistics?

Endeavour?

"An Introduction to Docker for Reproducible Research" (Carl Boettiger)

Challenges:

- dependency hell
- imprecise documentation
- code rot
- barriers to adoption and reuse in existing solutions

Current approaches to solve barriers to adoption:

.1. *workflow software* (e.g. `make`), but:

- no ownership and control of tools
- cannot meet need of every researcher
- standards-based => slower development

.2. *virtual machines*, but:

- black box => bad for reproducibility
- cannot be used as building block for downstream analysis

:: How? ::

Docker = lightweight runtime and packaging tool built from existing components of the linux kernel

Docker = development workflow and ecosystem

Terminology:

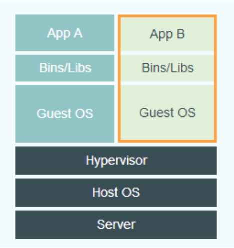
- *image* = immutable description of a system
- *container* = running instance of an image

Possible uses:

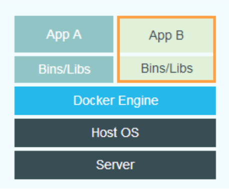
- micro-services (e.g. neo4j)
- commands that return immediately (e.g. pandoc)
- interactive commands

What does the system look like?

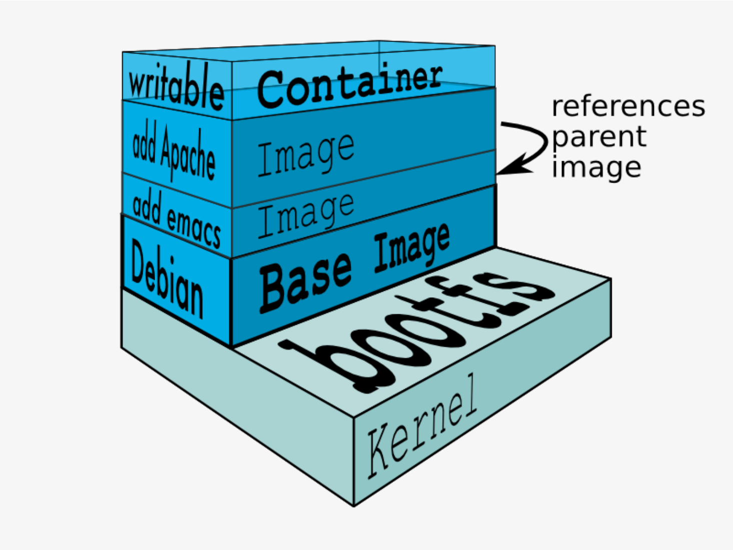
Virtual machines vs docker



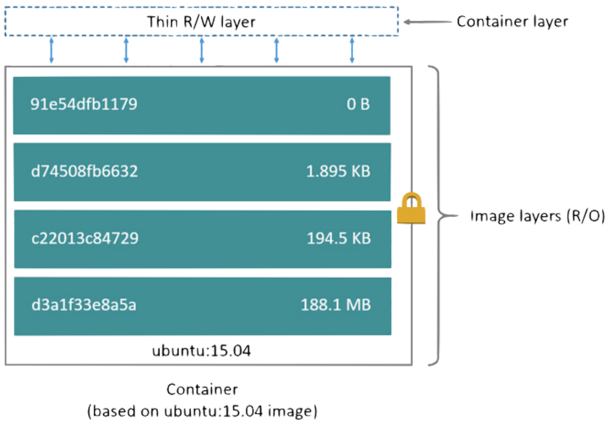
Docker



Docker layers



"Union File System" across layers



Description of a docker image: `Dockerfile`

```
FROM neo4j:2.3
MAINTAINER Jan Aerts <jan.aerts@kuleuven.be>
EXPOSE 7474

RUN mkdir -p /startup
ADD docker-startup.sh /startup/docker-startup.sh
ADD gene-nodes.txt /startup/gene-nodes.txt
ADD disease-nodes.txt /startup/disease-nodes.txt
ADD gene-disease_relationships.txt /startup/gene-disease_relationships.txt
ADD gene-gene_relationships.txt /startup/gene-gene_relationships.txt
RUN chmod a+x /startup/docker-startup.sh

CMD ["/bin/sh", "/startup/docker-startup.sh"]
```

`Dockerfile` commands

- `FROM`: set the base image
- `RUN`: execute command in a new layer on top of the current image
- `ADD`: copy file from local directory into image
- `CMD`: default command to run when container is created
- ...

https://docs.docker.com/engine/reference/builder/

Docker commands

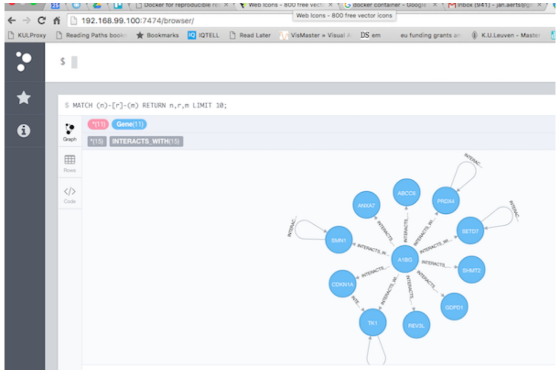
`docker build`

```
docker build --rm -t jandot/neo4j-i0u19a .
```

```
Sending build context to Docker daemon 15.25 MB
Step 1 : FROM neo4j:2.3
2.3: Pulling from library/neo4j
8b87079b7a06: Pull complete
a3ed95caeb02: Pull complete
1bb8eaf3d643: Pull complete
8b814800df49: Pull complete
8819a60acbef: Pull complete
1be1b08f002b: Pull complete
192853c43a20: Pull complete
9cebd99651f4: Pull complete
4e875535e701: Pull complete
beacf1089488: Pull complete
43ecb2670ec8: Pull complete
6de76c08a945: Pull complete
Digest: sha256:272442ce02990019a11690813f5e0853f5adea1c7b5177ab097c2427a019df4b
Status: Downloaded newer image for neo4j:2.3
 ---> c575eeb7b57a
Step 2 : MAINTAINER Jan Aerts <jan.aerts@kuleuven.be>
```

`docker run`

```
docker run -d -p 7474:7474 jandot/neo4j-i0u19a
```



```
$ MATCH (n)-[r]-(m) RETURN n,r,m LIMIT 10;
```

`docker exec`

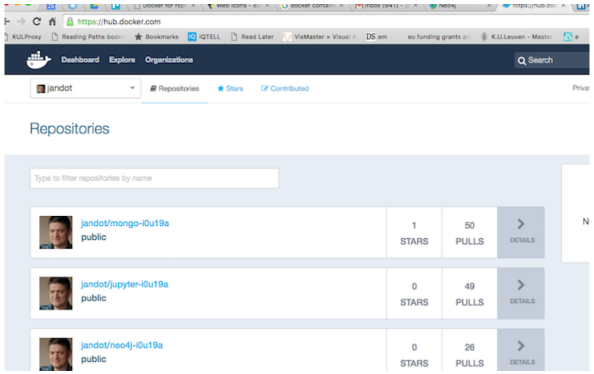
```
docker exec -it <id> /bin/bash
```

to enter a running image

docker push

```
docker push jandot/neo4j-i0u19a
```

hub.docker.com



Repositories

Type to filter repositories by name

jandot/mongo-i0u19a public	1 STARS	50 PULLS	DETAILS
jandot/jupyter-i0u19a public	0 STARS	49 PULLS	DETAILS
jandot/neo4j-i0u19a public	0 STARS	26 PULLS	DETAILS

How to build a `Dockerfile`?

.1. Start from a base image `Dockerfile`

```
FROM jupyter/datascience-notebook
```

(see https://hub.docker.com/r/ipython/notebook/)

.2. Build and run the base image

```
docker build -t ipython-dev-env .
docker run -it --rm -p 8888:8888 ipython-dev-env
```

Navigate to http://localhost:443

.3. Install an extra python module into the notebook server

```
!pip3 search gensim
!pip3 install gensim
```

.4. See if it works

```
import gensim
```

.5. If it works: add command to `Dockerfile`

```
FROM jupyter/datascience-notebook
RUN pip3 install gensim
```

.6. Rebuild and re-run

Installing docker

- linux: through official packages
- OSX & Windows: need lightweight VM => boot2docker or beta

How does this solve the challenges?

- dependency hell
- imprecise documentation
- code rot
- barriers to adoption and reuse in existing solutions

Dependency hell

- docker images

Imprecise documentation

- `Dockerfile`

Code rot

- docker image versions

```
docker build --rm -t jandot/neo4j-i0u19a .
```

vs

```
docker build --rm -t jandot/neo4j-i0u19a:1.0 .
```

Barriers to adoption and reuse

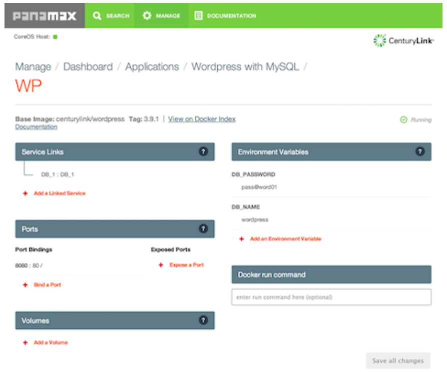
- build once, run everywhere (on student's laptop, ...)
- integrating into local development environments
- portable computation & sharing
- re-usable modules
- versioning
- fast

Complete applications

docker-compose.yml

```
mongo:
  image: mongo:2.6.11
  ports:
   - "27017:27017"
application:
  build: .
  command:  node --debug=5858 app.js --color=always
  ports:
   - "3000:3000"
   - "5858:5858"
  volumes :
  - ./:/app
  links:
   - mongo
```

panamax.io



CoreOS Host:

Manage / Dashboard / Applications / Wordpress with MySQL /

WP

Base Image: centurylink/wordpress Tag: 3.9.1 | View on Docker Index
Documentation

Running

Service Links

DB_1 : DB_1

+ Add a Linked Service

Ports

Port Bindings: 8080 : 80 /

+ Bind a Port

Exposed Ports

+ Expose a Port

Volumes

+ Add a Volume

Environment Variables

DB_PASSWORD: pass@word01

DB_NAME: wordpress

+ Add an Environment Variable

Docker run command

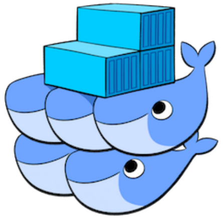
enter run command here (optional)

Save all changes

Deploying applications on cluster

Docker Swarm

www.docker.com/products/docker-swarm



- Evaluate Swarm in a sandbox
- Try Swarm at scale

Kubernetes

kubernetes.io

- Getting started
- Turn-key cloud solutions on Google Compute Engine, AWS, Azure

Best practices

Use docker containers during development

Write `Dockerfiles` instead of installing interactive sessions

Add tests or checks to the `Dockerfile`

Use and provide appropriate base images

Version everything in the `Dockerfile`

```
FROM jupyter/pyspark-notebook:latest
MAINTAINER Jan Aerts <jan.aerts@kuleuven.be>
RUN pip install pymongo
RUN pip install py2neo
RUN pip install bokeh
...
```

vs

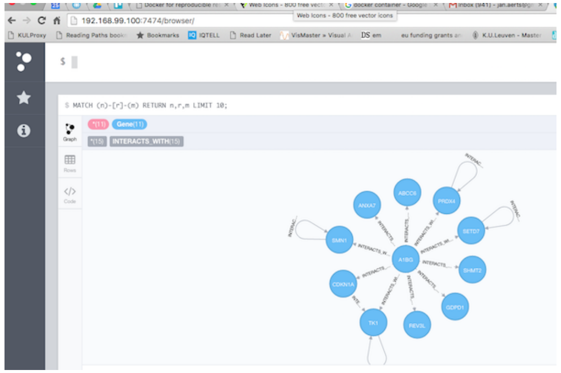
```
FROM jupyter/pyspark-notebook:2d878db5cbff
MAINTAINER Jan Aerts <jan.aerts@kuleuven.be>
RUN pip install pymongo==3.2.2
RUN pip install py2neo==2.0.8
RUN pip install bokeh==0.11.1
...
```

:: What? ::

Demo

neo4j

```
docker run -d -p 7474:7474 jandot/neo4j-i0u19a
```



192.168.99.100:7474/browser/

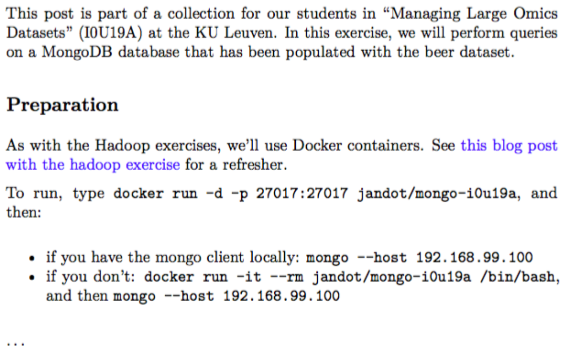
```
$ MATCH (n)-[r]-(m) RETURN n,r,m LIMIT 10;
```

pandoc

my-text.md

```
This post is part of a collection for our students in "Managing Large
Omics Datasets" (I0U19A) at the KU Leuven. In this exercise, we will
perform queries on a MongoDB database that has been populated with
the beer dataset.

## Preparation
As with the Hadoop exercises, weâll use Docker containers. See [this
blog post with the hadoop exercise](http://vda-lab.github.io/2016/04/hadoop-tutor
for a refresher.

To run, type `docker run -d -p 27017:27017 jandot/mongo-i0u19a`, and
then:

* if you have the mongo client locally: `mongo --host 192.168.99.100`
* if you don't: `docker run -it --rm jandot/mongo-i0u19a /bin/bash`,
and then `mongo --host 192.168.99.100`

...
```

```
docker run -v $(pwd):/source jandot/pandoc -f markdown -t latex /source/my-text.m
```



This post is part of a collection for our students in “Managing Large Omics Datasets” (I0U19A) at the KU Leuven. In this exercise, we will perform queries on a MongoDB database that has been populated with the beer dataset.

Preparation

As with the Hadoop exercises, we’ll use Docker containers. See this blog post with the hadoop exercise for a refresher.

To run, type `docker run -d -p 27017:27017 jandot/mongo-i0u19a`, and then:

- if you have the mongo client locally: `mongo --host 192.168.99.100`
- if you don’t: `docker run -it --rm jandot/mongo-i0u19a /bin/bash`, and then `mongo --host 192.168.99.100`

...

jupyter and neo4j

```
docker run -d -p 7474:7474 jandot/neo4j-i0u19a
docker run -d -p 8888:8888 jandot/jupyter-i0u19a
```

This presentation...

slides/slides.md

```
 --NEWH
Docker container =~ light-weight virtual machine

*image* = immutable description of a system

*container* = running instance of an image

 --NEWV
Possible uses:

* micro-services (e.g. neo4j)
* commands that return immediately (e.g. pandoc)
* not: interactive commands

Using both neo4j and mongodb in the same application?
...
```

Viewing the presentation

```
docker run \
    -p 8000:8000 \
    -v $(pwd)/images:/opt/presentation/images \
    -v $(pwd)/slides:/opt/presentation/slides \
    -d jandot/docker-presentation
```

Bioinformatics-specific examples:

- RNA sequencing pipeline: www.nextflow.io/example4.html
- Biodocker.org: BLAST, EMBOSS, bwa, picard, samtools, vcftools, ...
- Algorun - Docker-based container template for computational algorithms
- RStudio: `docker run -d -p 8787:8787 rocker/rstudio` => localhost:8787 (`rstudio/rstudio`)

≡

More information

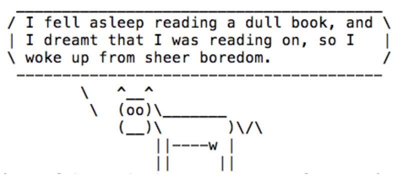
docker.io

docs.docker.com

hub.docker.com

≡

:: Exercises ::



```
 _________________________________________
/ I fell asleep reading a dull book, and \
| I dreamt that I was reading on, so I   |
\ woke up from sheer boredom.            /
 -----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

≡

To create the cow image, you'll need to have the `fortune` and `cowsay` commands installed on an ubuntu system.

≡

Exercises:

1. Running an image interactively
2. Running an image with default behaviour
3. Adding customizable quote at build stage
4. Adding customizable quote at run stage

1. Running an image interactively

- Start an interactive ubuntu docker image: `docker run -it --rm ubuntu /bin/bash`
  - What does each parameter mean?
- Update the software packages: `apt-get update`
- Install `fortune`: `apt-get install fortune`
- Install `cowsay`: `apt-get install cowsay`
- Try it out: `/usr/games/fortune | /usr/games/cowsay`
- When done, type `exit`

What do you need to do if you want to do this again?

2. Running an image with default behaviour: Let the cow say a random adage

If you want to generate a new image: need to run an ubuntu image again, install `fortune` and `cowsay`, and run the command.

Better: create specific image based on Dockerfile

Reference: see https://docs.docker.com/engine/reference/builder/

Steps:

- Create `Dockerfile` file
- Build docker images
- Run docker container

Dockerfile

```
FROM ubuntu:14.04
MAINTAINER YourName <youremail>

#Get up-to-date
RUN apt-get update && apt-get upgrade -y

#Install fortune and cowsay
RUN apt-get install -y fortune cowsay

#Set the default command
CMD /usr/games/fortune | /usr/games/cowsay
```

Build and run:

```
docker build -t <yourname>/exercise2 .
docker run <yourname>/exercise2
```

3. Adding customizable quote at build stage:
Let the cow say anything you provide

What if we want to tell the cow exactly what to say, instead of relying on `fortune`?

Create file `my-saying.txt` that contains our message.

Dockerfile

```
FROM ubuntu:14.04
MAINTAINER YourName <youremail>

#Get up-to-date
RUN apt-get update && apt-get upgrade -y

#Install cowsay
RUN apt-get install -y cowsay

#Copy the file with the saying
COPY my-saying.txt /tmp/my-saying.txt

#Set the default command
CMD /usr/games/cowsay < /tmp/my-saying.txt
```

Problem: changing contents of `my-saying.txt` is not reflected in what cow says, unless we rebuild the image

4. Adding customizable quote at run stage: Let the cow say anything you provide

New approach: *mount* local directory

Dockerfile

```
FROM ubuntu:14.04
MAINTAINER YourName <youremail>

#Get up-to-date
RUN apt-get update && apt-get upgrade -y

#Install cowsay
RUN apt-get install -y cowsay

#Make working directory
RUN mkdir /work

#Set the default command
CMD /usr/games/cowsay < /work/my-saying.txt
```

- `docker build -t <yourname>/exercise4 .`
- ``docker run -v `pwd`:/work <yourname>/exercise4``