

# FAIR\_bioinfo : Open Science and FAIR principles in a bioinformatics project

How to make a bioinformatics project more reproducible

C. Hernandez<sup>1</sup>    T. Denecker<sup>2</sup>    J. Sellier<sup>2</sup>    G. Le Corguillé<sup>2</sup>  
C. Toffano-Nioche<sup>1</sup>

<sup>1</sup>Institute for Integrative Biology of the Cell (I2BC)  
UMR 9198, Université Paris-Sud, CNRS, CEA  
91190 - Gif-sur-Yvette, France

<sup>2</sup>IFB Core Cluster taskforce

June 2021

# General information

## Practical information:

- Dates: June 28th - 30th
- Location: Institut des Systèmes Complexes, 113 rue Nationale, 75013-Paris
- Courses: 9:00 to 17:30
- Meal: 12:30-14:00
- Pauses: 10:30-11:00 + 15:30-16:00
- 2 days of courses + 1 day of course building

## Round table:

- Teachers
- Learners

## Ressources:



- GitLab
- L<sup>A</sup>T<sub>E</sub>X

# Training schedule

## Day 1:

- Introduction to reproducibility
- History management (3 Practical Sessions, git, GitHub)
- Control your development environment (1 PS, CONDA)
- Encapsulation (2 PS, docker)

## Day 2:

- Workflow (2 PS, SNAKEMAKE)
- Traceability with notebooks (2 PS, jupyter, zenodo)
- IFB resources (2 PS, slurm, singularity)
- Sharing and disseminating ( GitHub, zenodo)
- Conclusion

## Day 3:

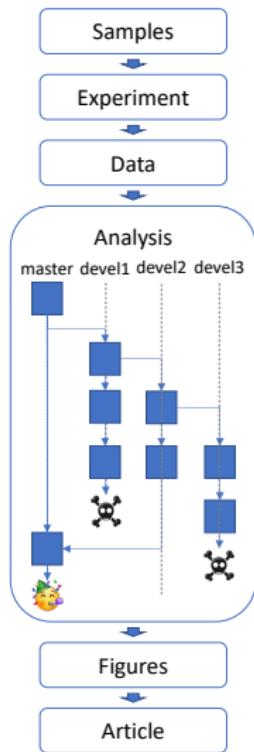
- Empowerment and improvement of resources

# Table of contents

- 1 Introduction to reproducibility
- 2 History management
- 3 Control your development environment
  - Introduction
  - Encapsulation
  - The Conda stack
  - What is Docker?
  - Practical session
- 4 Workflow
- 5 Tracability with Notebook
- 6 IFB resources
- 7 Sharing and dissemination
- 8 Conclusion
- 9 3rd Day

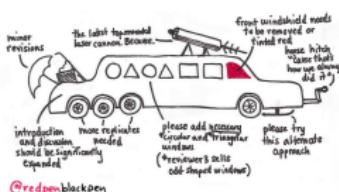
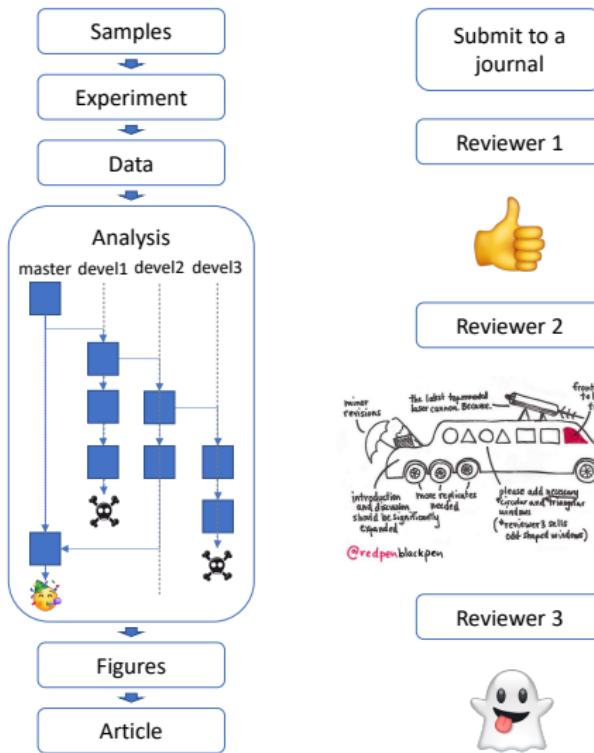
# Introduction

## A (not-so-uncommon) nightmare



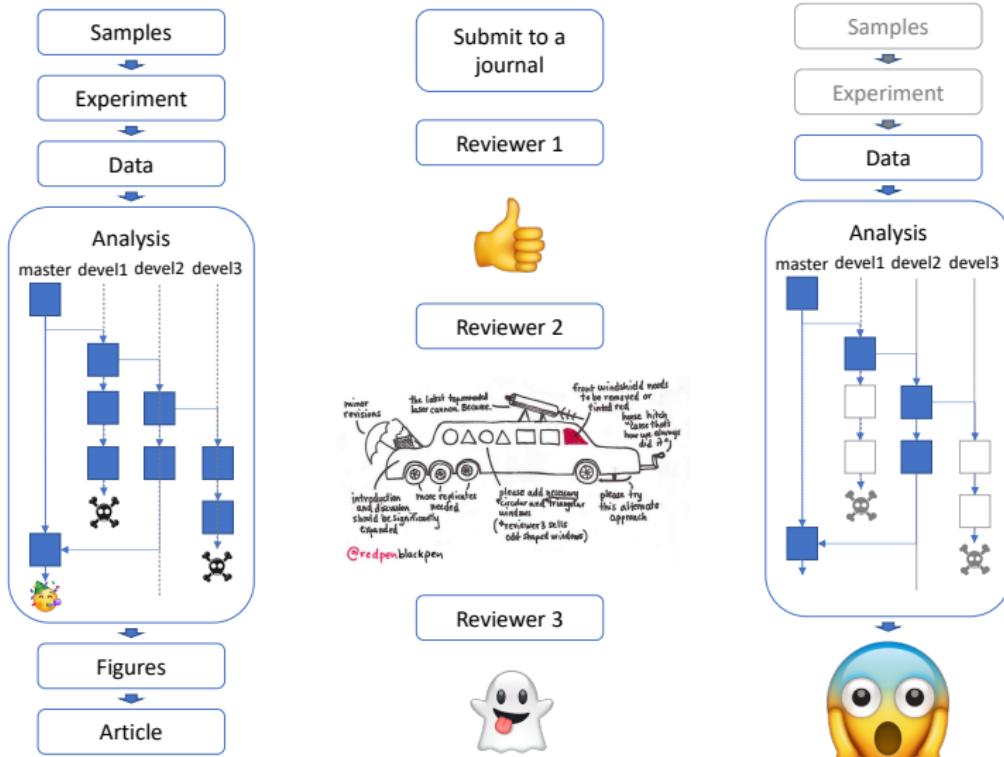
# Introduction

## A (not-so-uncommon) nightmare



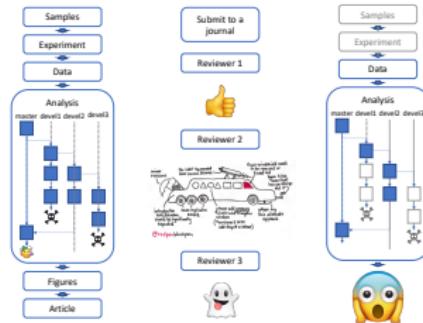
# Introduction

## A (not-so-uncommon) nightmare



# Introduction

## A (not-so-uncommon) nightmare



What changed?

- Package
- Software
- Libraries
- Environment variables
- OS version
- Computer
- ..?

# Different levels of encapsulation

Goal : capture the system environment of applications (OS, packages, libraries,...) to control their execution.

- Hardware virtualisation (virtual machines) 
- OS virtualisation (images and containers) 
- Environment management **CONDA**

# Encapsulation

Let's say we want to install RStudio...

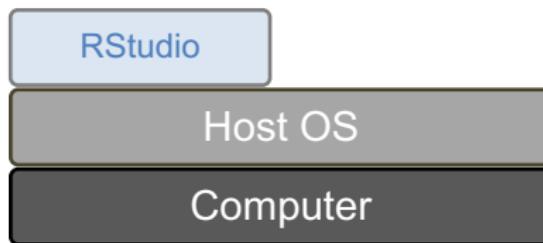


# Encapsulation

We started with a computer using a specific OS...

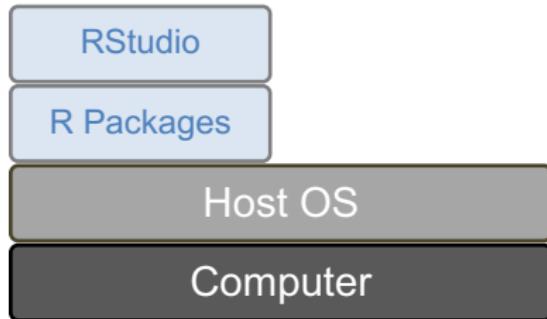


# Encapsulation



We started with a computer using a specific OS...  
And inside this environment, we installed a new application.

# Encapsulation



We started with a computer using a specific OS...  
And inside this environment, we installed a new application.  
Applications rely on dependencies,  
e.g. external libraries.

# Encapsulation



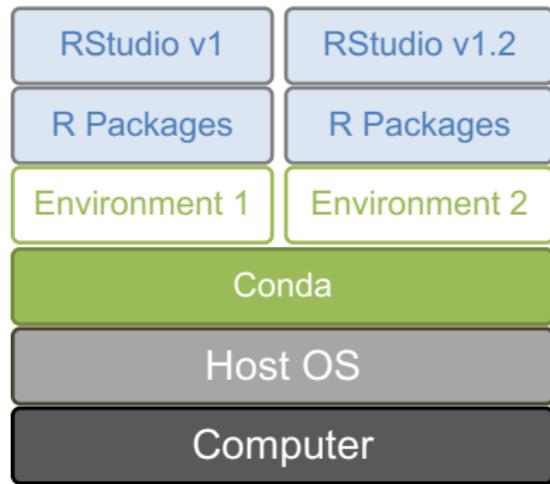
Usually dependencies of different applications don't interfere.  
But what if we want to test the latest version of our favourite tool?  
There might be conflicts...

# Encapsulation



Usually dependencies of different applications don't interfere.  
But what if we want to test the latest version of our favourite tool?  
There might be conflicts...

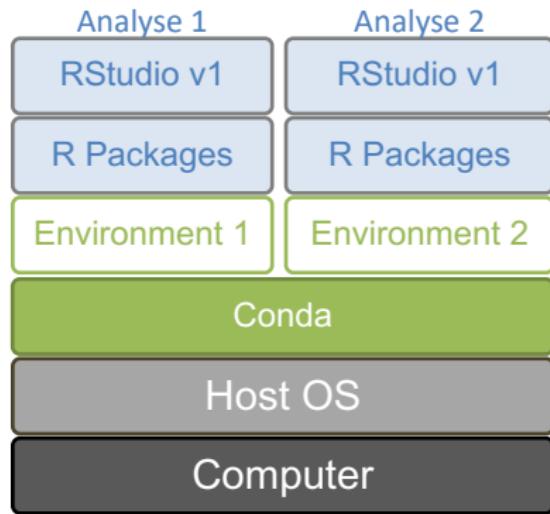
# Encapsulation : managing environments



Idea : create separated environments  
for each application.

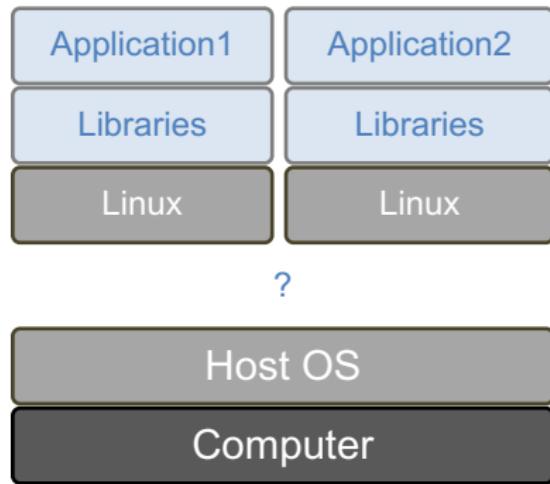
**CONDA**

# Encapsulation : managing environments



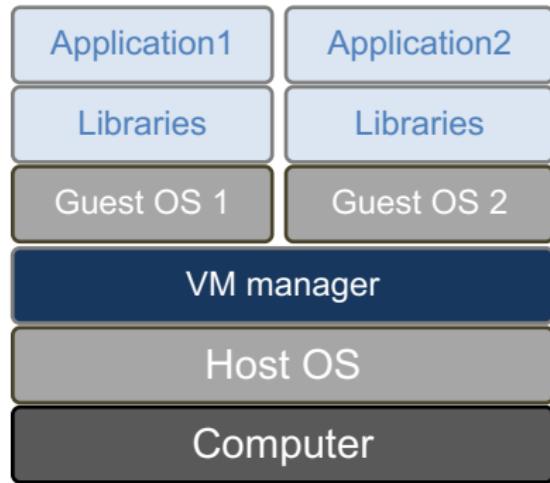
Idea : create separated environments for each application.  
More versatile: create a new environment per analysis.

# Encapsulation : hardware virtualisation



But what if we want to install a software from a different OS?

# Encapsulation : hardware virtualisation



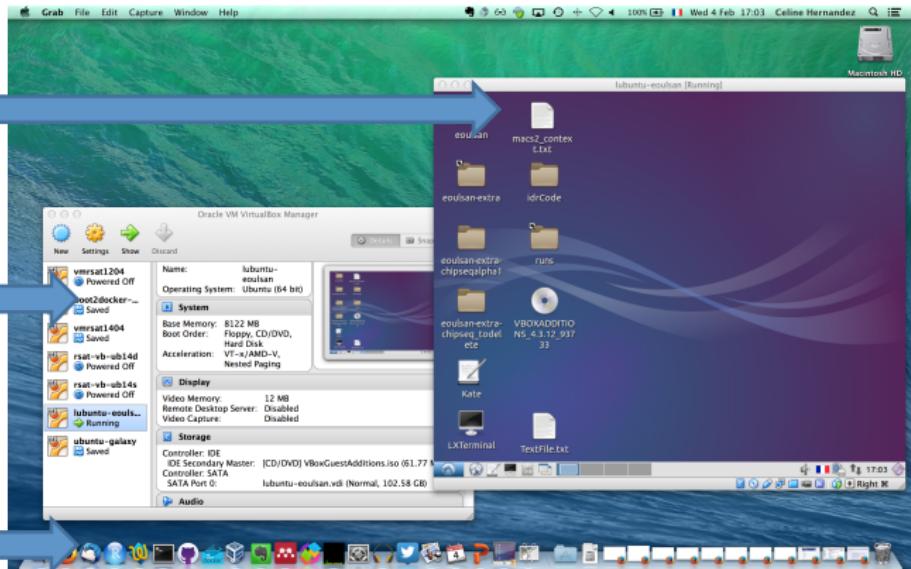
Idea: use virtual machines

Pros:

- Each application gets a completely different and independent environment
- Virtual machines can be transferred to another computer (using the same manager)

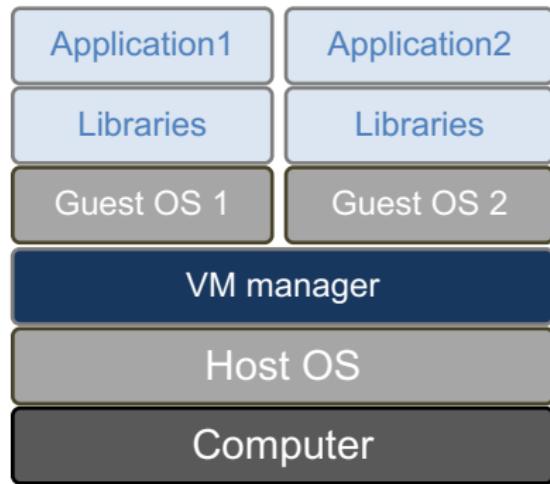
# Encapsulation : hardware virtualisation

Ubuntu



MacOS

# Encapsulation : hardware virtualisation



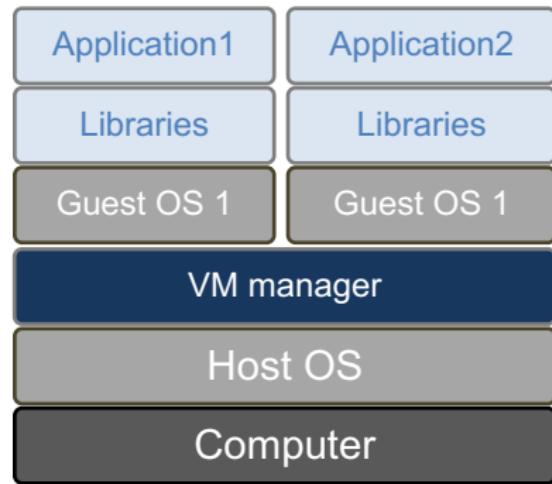
Idea: use virtual machines

Pros: transferable independent environments

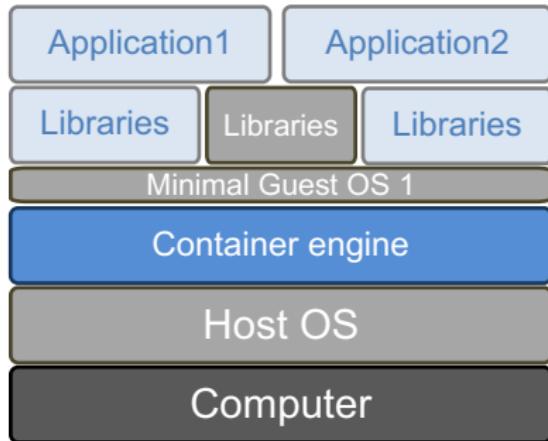
Cons:

- Redundancy between VMs
- Heavy to set up
- No automation

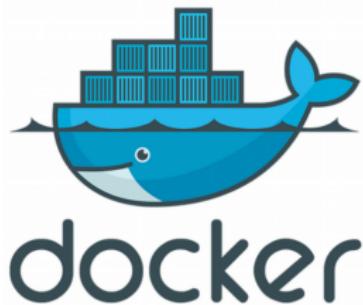
# Encapsulation : OS virtualisation



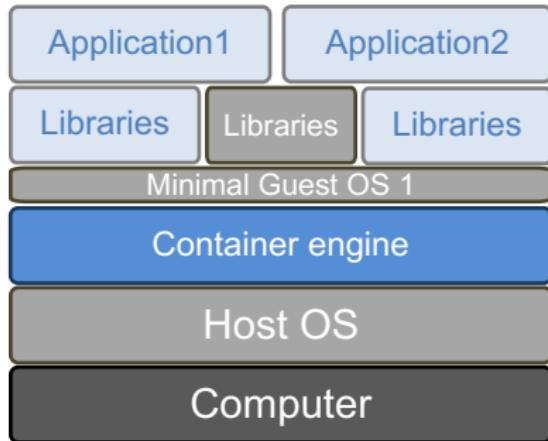
# Encapsulation : OS virtualisation



Idea: "trick" applications into believing that they are in a different OS than the host's  
Avoid redundancy.



# Encapsulation : OS virtualisation

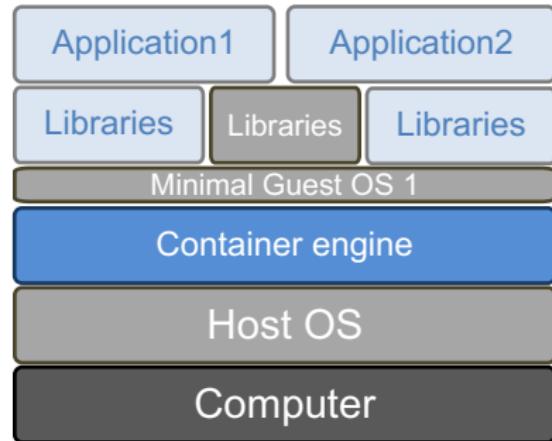


OS virtualisation vs hardware virtualisation

Pros:

- Speed
  - ▶ Installation is faster
  - ▶ No boot time
- Lightweight
  - ▶ Minimal base OS
  - ▶ Minimal libraries and application set
- Easy sharing of applications

# Encapsulation : OS virtualisation



Cons:

- Singularity to use images on a cluster
- Changes of policies of the Docker company

# Docker policy

## Update of the Docker Image retention policy (13/08/2020)

### What is a container image retention limit and how does it affect my account?

Image retention is based on the activity of each individual image stored within a user account. If an image has not either been pulled or pushed in the amount of time specified in your subscription plan, the image will be tagged "inactive." Any images that are tagged as "inactive" will be scheduled for deletion. Only accounts that are on the **Free** individual or organization plans will be subject to image retention limits. A new dashboard will also be available in Docker Hub that offers the ability to view the status of all of your container images.

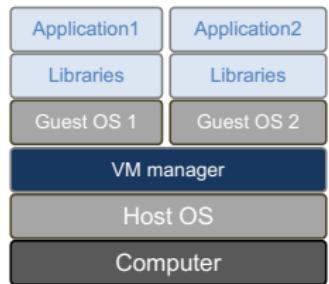
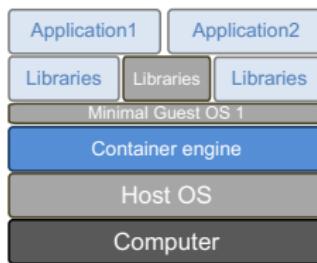
### What are the new container image retention limits?

Docker is introducing a container image retention policy which will be enforced starting November 1, 2020. The container image retention policy will apply to the following plans:

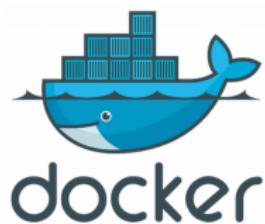
- Free plans will have a 6 month image retention limit
- Pro and Team plans will have unlimited image retention

<https://www.docker.com/pricing/retentionfaq>

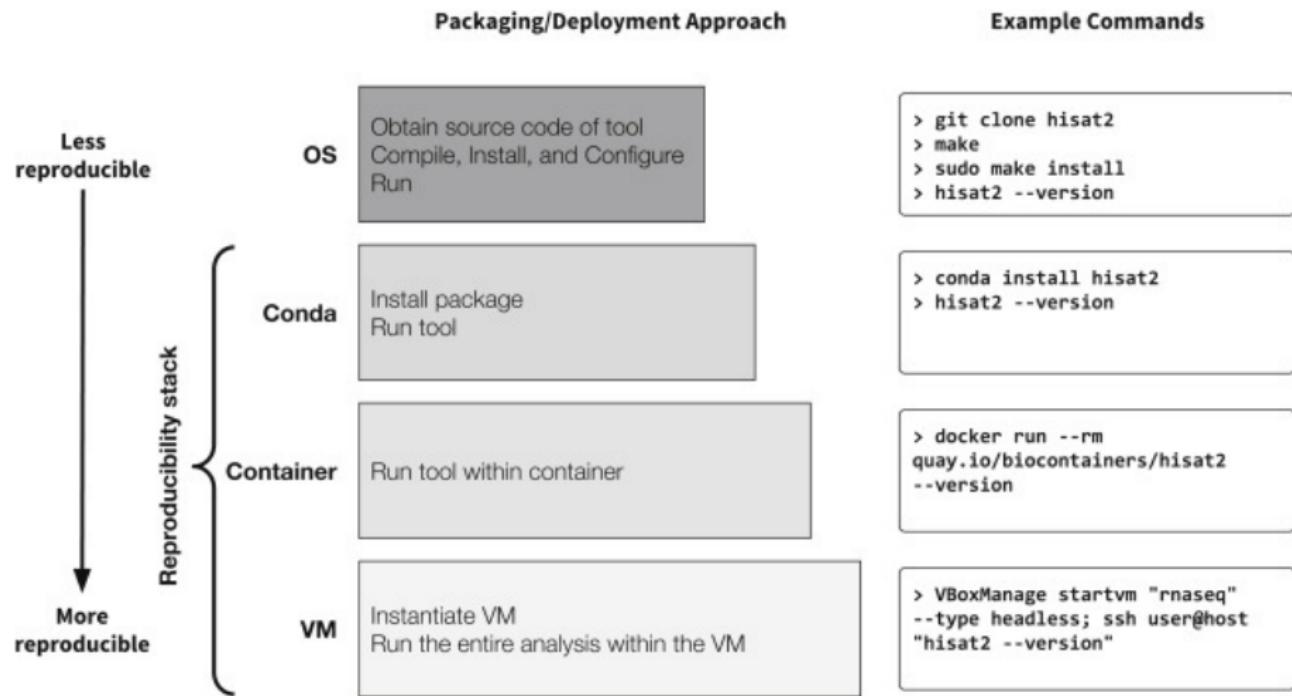
# Encapsulation



CONDA



# Encapsulation and reproducibility stack



Practical Computational Reproducibility in the Life Sciences - Björn  
Grüning et al (2018)

# Conda

# CONDA: an environment manager

## Conda definitions

- Environment: a set of packages/tools in a directory (added to our PATH)
- Conda: an open source package + a general-purpose environment management system (installation, execution, upgrade). For any programming language, multi-platform (Windows, MacOS, Linux).
- Conda package: a compressed tarball of a tool

## Why using an environment manager?

- avoid compilation and dependencies problems: an environment manager will take care of everything!
- have several environments in parallel each with their own set of tools
- useful when cross-tools dependencies are incompatible with each other

## Conda distribution

- Anaconda: a data science platform, comes with a lot of packages
- Miniconda: come without installed packages

## Anconda cloud, the "conda hub"

- [Anaconda cloud](#) (private company) relies on the community of developers, concerns many domains (Machine Learning, Data Visualization, Dashboarding-web, Image Processing, Natural Language Processing, etc)
- Anaconda cloud: made up of channels/owners. Each channels contains one or more conda packages
- be careful when downloading any packages from an untrusted source, always inspect before installation

## Some conda channels

- default
- conda-forge: many popular python packages (analogous to PyPI but with a unified, automated build infrastructure and more peer review of recipes)
- bioconda: bioinformaticians' contributions
- private

## Channels list order

- when different channels have the same package ⇒ collisions
- collisions resolved following the order of your channels list ⇒ put supplemental channels at the bottom of your channel list

# CONDA R, mamba

## Conda and R

The R interpreter is included in the `r-essentials` package ( 200 r packages). Add `r-` before the regular r package name (eg. `r-ggplot2`)

◆ Favorites ◆ Downloads ◆ Package (owner / package)				Platforms
18	304565	 <a href="#">r / r-essentials</a> 3.6.0	Some essential packages for working with R	linux-32 linux-64 osx-64 win-32 win-64  conda
11	198513	 <a href="#">skyblued / r-essentials</a> 3.5.1	Some essential packages for working with R	linux-64 osx-64 win-32 win-64  <a href="#">copy</a> conda
1	66845	 <a href="#">conda-forge / r-essentials</a> 4.1	Some essential packages for working with R. This was migrated from the 'r' channel.	linux-64 noarch osx-64  conda

## Mamba

A fast drop-in alternative to conda, using libSolv for dependency resolution

```
1 conda install -c conda-forge mamba
```

Next, replace `conda` by `mamba` to use it

# CONDA command

## simple commands

```
1 conda create env -n myenv # creation of a conda environment
2 conda info --envs # list environments (* for the active one)
3 conda activate myenv # active the myenv environment
4 conda list # list packages (only in an active environment)
5 conda install package # installation of a tool/package
6 conda remove package # suppress the tool from the
    environment
7 conda env remove -n myenv # suppress the myenv environment
8 conda deactivate # inactivate the environment
```

## miniconda3

With the miniconda3 distribution and by default, environments are installed in a miniconda3/envs/ repository



# CONDA 2 modes

## interactive

- create an environment
- activate the environment
- install some conda packages

## configuration file

- list all conda packages in a configuration file (yaml or json format)
- create the environment based on the configuration file (option -f)
- activate the environment

## reproducibility

- good practice: use a configuration file
- specify a precise version of a package:  
`<channel>::<package>=<version>`

# Conda Exercise

# Conda setup

## How to access conda?

- Conda is so used that it could even be installed by default to your machine. To test this: `conda --version`
- if not, may install it or got it by a docker image:

```
1 docker run -i -t -v ${PWD}:/data continuumio/miniconda3
```

- already activated on the IFB cluster (otherwise with module:  
`module load conda`)

# How to access tools?

## Manage Conda environment

- ➊ create the working environment:

```
1 conda create env -n myenv
```

- ➋ activate it:

```
1 conda activate myenv
```

- ➌ if not yet done, install packages (specify the channel):

```
1 conda install -c bioconda bowtie2
```

- ➍ work with the tools

- ➎ quite the environment:

```
1 conda deactivate
```

# Install snakemake with conda

## Objective

Create a conda configuration file to install the snakemake tool.

## Hint

- Search its channel in the Anaconda cloud web pages
- the "minimal" environment is sufficient

# Install snakemake with conda

## condaEnvSmk.yml

```
1 channels:  
2   - conda-forge  
3   - bioconda  
4   - main  
5 dependencies:  
6   - snakemake-minimal=6.5.0
```

## run

```
1 conda env create -n condaEnvSmk -f condaEnvSmk.yml  
2 conda activate condaEnvSmk  
3 snakemake --version
```

# Docker

# What is Docker?

Docker is not very “old”

- First commit January 2013
- First version March 2013
- Version 1.0 in June 2014

But its adoption was fast

- Officially packaged in Ubuntu since 2014 (v14.04)

# What is Docker?

Image



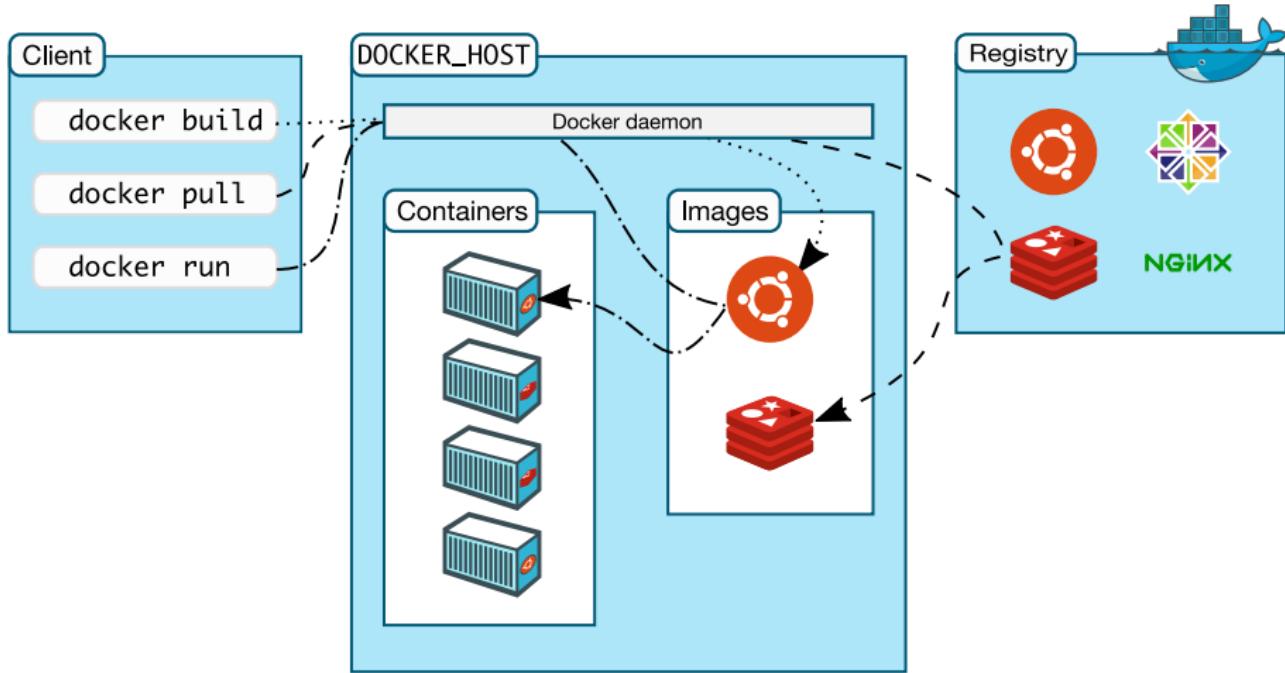
Container



- Set of libraries and functions
- Fixed. Cannot be modified
- Can be stored/shared online
- Can be automatically built

- "Active image"
- Can be modified (interactive)
- Can be turned into an image
- One image, many containers

# What is Docker?



(<https://docs.docker.com/get-started/overview/>)

# What is Docker?

## DockerHub

The screenshot shows the Docker Hub homepage with the URL <https://hub.docker.com/explore/>. The page title is "Docker Store is the new place to discover public Docker content. Check it out →". It features a search bar and navigation links for "Explore", "Help", "Sign up", and "Sign in". Below the header, there's a section titled "Explore Official Repositories" with a list of popular Docker images:

Image	Name	Owner	Stars	Pulls	Details
	nginx	official	5.3K	10M+	<a href="#">DETAILS</a>
	redis	official	3.4K	10M+	<a href="#">DETAILS</a>
	busybox	official	924	10M+	<a href="#">DETAILS</a>
	ubuntu	official	5.5K	10M+	<a href="#">DETAILS</a>
	registry	docker	1.3K	10M+	<a href="#">DETAILS</a>

(<https://hub.docker.com/>)



# What is Docker?

## Usermade images (1/2)

The screenshot shows the Docker Hub user profile for `genomicpariscentre`. The profile page includes a large placeholder image for a profile picture, the user's name (`genomicpariscentre`), the organization name (`Genomic Paris Centre`), location (`Paris`), website (`http://genomique.biologie.ens.fr`), and the date joined (`June 2014`). Below this, there is a section titled "Repos" which lists eight Docker images:

Image Name	Description	Stars	Pulls	Details
<code>genomicpariscentre/star</code>	public   automated build	1	1.2K	<a href="#">DETAILS</a>
<code>genomicpariscentre/bcl2fastq</code>	public   automated build	0	1.2K	<a href="#">DETAILS</a>
<code>genomicpariscentre/blast2</code>	public   automated build	0	765	<a href="#">DETAILS</a>
<code>genomicpariscentre/bcbio-nextgen</code>	public   automated build	0	451	<a href="#">DETAILS</a>
<code>genomicpariscentre/fastqc</code>	public   automated build	0	404	<a href="#">DETAILS</a>
<code>genomicpariscentre/bowtie2</code>	public   automated build	0	308	<a href="#">DETAILS</a>
<code>genomicpariscentre/samtools</code>	public   automated build	0	304	<a href="#">DETAILS</a>
<code>genomicpariscentre/eulools</code>	public   automated build	2	231	<a href="#">DETAILS</a>

(url`https://hub.docker.com/u/genomicpariscentre/`)



# What is Docker?

Usermade images (2/2)

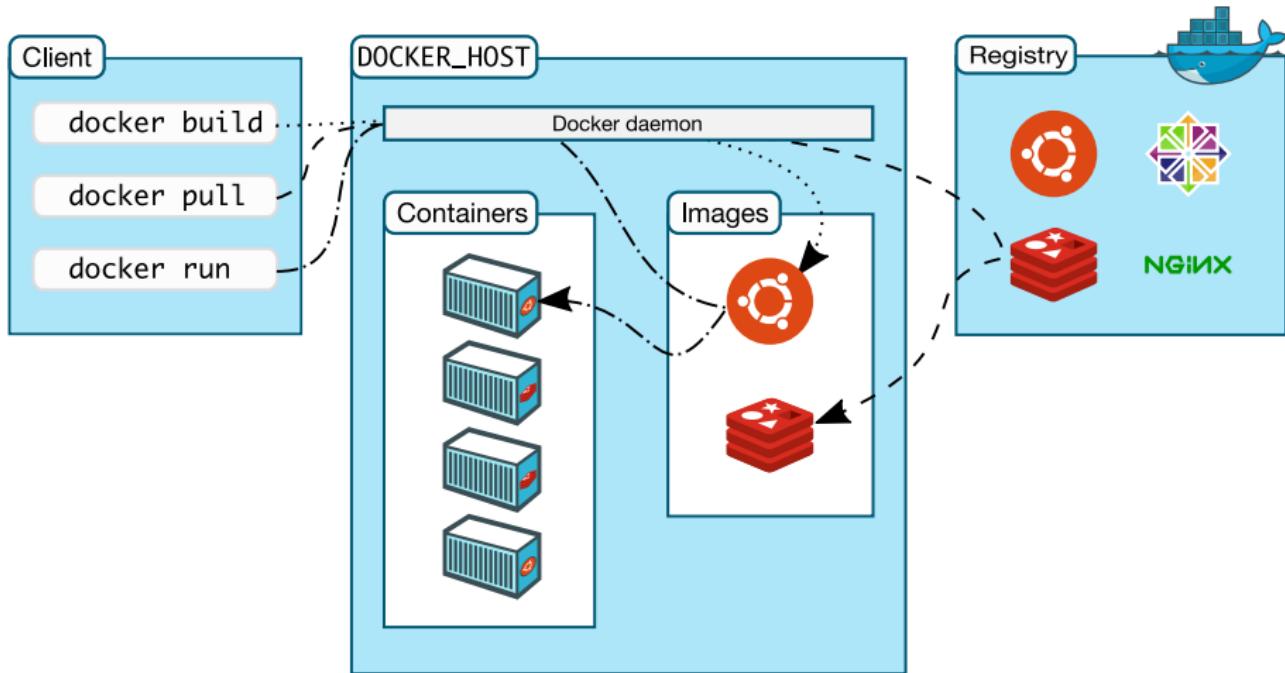
Be critical!

The screenshot shows the Docker Hub interface for the repository `genomicpariscentre/samtools`. The page title is "genomicpariscentre/samtools". The repository is described as "PUBLIC | AUTOMATED BUILD". It was last pushed 2 years ago. There are tabs for "Repo Info", "Tags", "Dockerfile", and "Build Details". Under "Repo Info", there are sections for "Short Description" (Samtools is a processor of sequence alignments for SAM and BAM formats) and "Full Description" (Samtools is a processor of sequence alignments for SAM and BAM formats). To the right, there is a "Docker Pull Command" field containing `docker pull genomicpariscentre/samtools`, an "Owner" section showing the profile of `genomicpariscentre`, and a "Source Repository" section linking to `GenomicParisCentre/dockerfiles`.

(<https://hub.docker.com/r/genomicpariscentre/samtools/>)



# What is Docker?



(<https://docs.docker.com/get-started/overview/>)

# What is Docker?

Other commands :

- docker images : list images available locally
- docker ps : status of containers
- docker rm : delete a container
- docker rmi : delete an image
- ...

(More details during the practical session.)