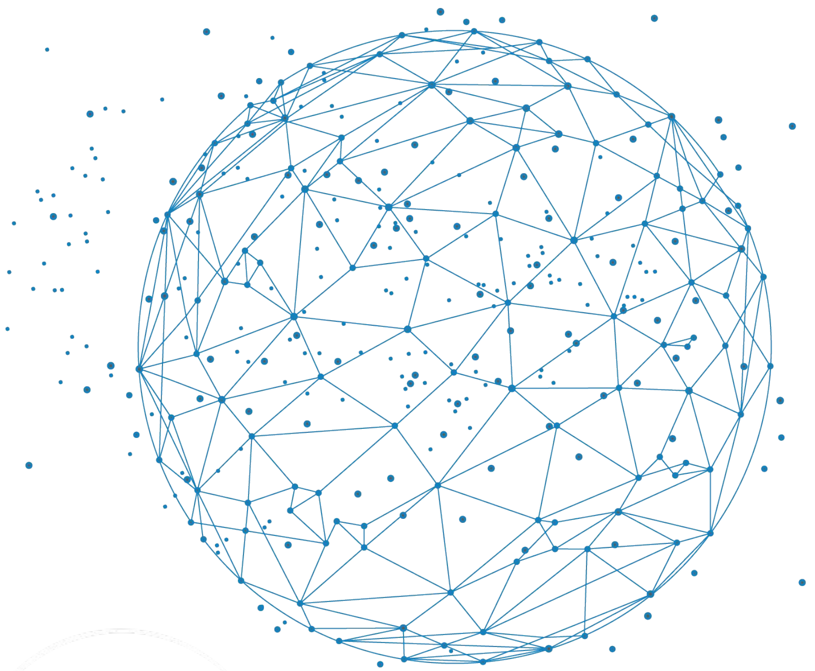
13 au 15 juin 2022, Institut des Systèmes Complexes, Paris



FAIR Bioinfo 2022
Les principes FAIR dans un projet de bioinformatique

Conclusion

Current schedule

- Introduction to FAIR & Open Science
- Project management (data, repository)
- Traceability with notebooks (jupyterlab)
- History management (Git)
- Introduction to encapsulation
- Environment management (Conda)
- Containerisation (docker)
- Share & disseminate, code & project (GitHub)
- HPC : cluster (Slurm)
- Analysis workflow (snakemake)
- HPC : containerisation (singularity)
- Conclusion

Current schedule

- Introduction to FAIR & Open Science
- Project management (data, repository)
- Traceability with notebooks (jupyterlab)
- History management (Git)
- Introduction to encapsulation
- Environment management (Conda)
- Containerisation (docker)
- Share & disseminate, code & project (GitHub)
- HPC : cluster (Slurm)
- Analysis workflow (snakemake)
- HPC : containerisation (singularity)
- Conclusion

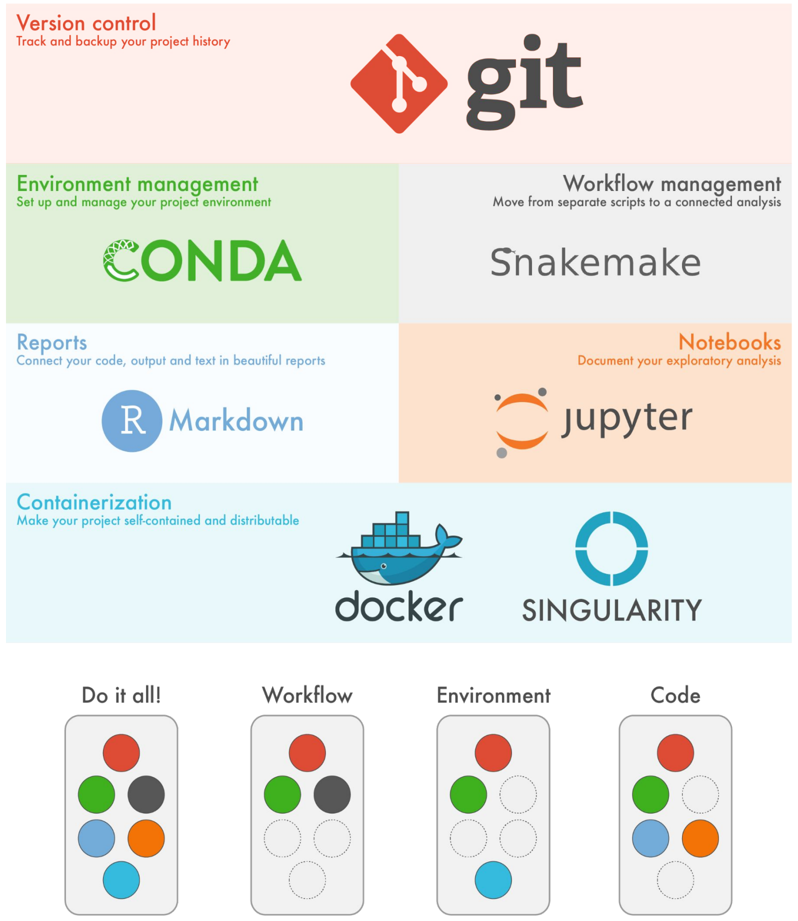
e-labbook

code development

HPC

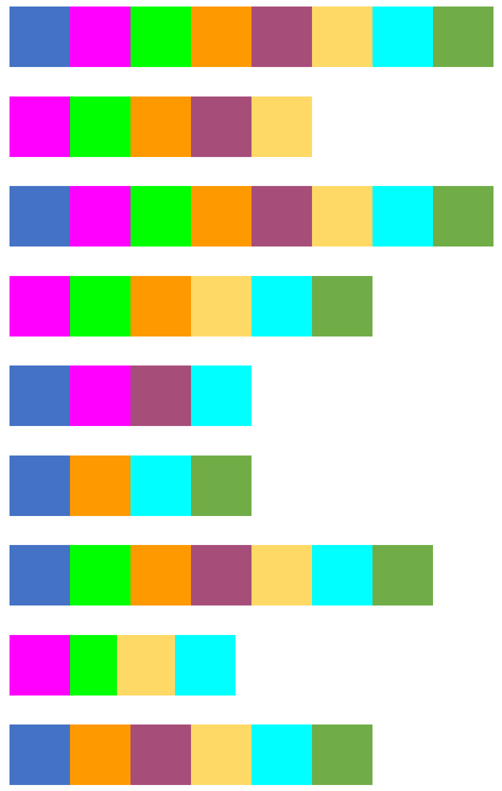
Current schedule

- Introduction to FAIR & Open Science
- Project management (data, repository)
- Traceability with notebooks (jupyterlab)
- History management (Git)
- Introduction to encapsulation
- Environment management (Conda)
- Containerisation (docker)
- Share & disseminate, code & project (GitHub)
- HPC : cluster (Slurm)
- Analysis workflow (snakemake)
- HPC : containerisation (singularity)
- Conclusion


Version control
Track and backup your project history
git
Environment management
Set up and manage your project environment
CONDA
Workflow management
Move from separate scripts to a connected analysis
Snakemake
Reports
Connect your code, output and text in beautiful reports
R Markdown
Notebooks
Document your exploratory analysis
jupyter
Containerization
Make your project self-contained and distributable
docker
SINGULARITY
Do it all!
Workflow
Environment
Code

Other use cases?

- Project management (data, repository)
- Traceability with notebooks (jupyterlab)
- History management (Git)
- Environment management (Conda)
- Containerisation (docker)
- Share & disseminate, code & project (Git***)
- HPC : cluster (Slurm)
- Analysis workflow (snakemake)
- HPC : containerisation (singularity)



Céline (NGS) | Claire (NGS analysis) | Emilie (NGS) | Gildas (admin)

Julien (Dev) | Pauline (WGA) | Hugo (Dev & NGS) | Thomas (Dev)

*** Lab !

Use case levels

- With a tool:
  - download, install, config, run
  - create new (env_config, snakefile, dockerfile)
  - participate to the development of the tool (bioconda)

FAIR_bioinfo tools

Findable

Accessible

Interoperable

Reusable

?

GitHub

GitHub Pages https://pages.github.com

zenodo doi

GitLab

docker

CONDA

GitHub

snakemake

docker

slurm

cc

CeCILL

jupyter

FAIR_bioinfo tools

Objective:

FAIR raw data

+

FAIR scripts

=

FAIR processed data

A virtuous cycle

Reproducibility checklist [1]

Code **avoid** workflows based on **point-and-click interfaces** (eg. Excel), enshrine computations and data manipulation in code

Document how code works, define parameters and computational environment required: comments, **notebooks** and **README**

Record key parameters (eg. the ‘seed’ values of a random-number generator)

Test functions using positive and negative control **data sets**, run those tests throughout development
Guide with master script (eg. ‘run.sh’) that downloads data sets and executes workflow

Reproducibility checklist [2]

Archive with long-term stability services such as Zenodo, Figshare and Software Heritage (GitHub is impermanent online repository). Track the project’s history with a **version-control** tools (eg. Git).

Note (tag) which version you used to create each result

Package with ready-to-use computational environments using **containerization** tools (eg. Docker, Singularity), web services (Code Ocean, Gigantum, Binder) or **virtual-environment** managers (Conda)

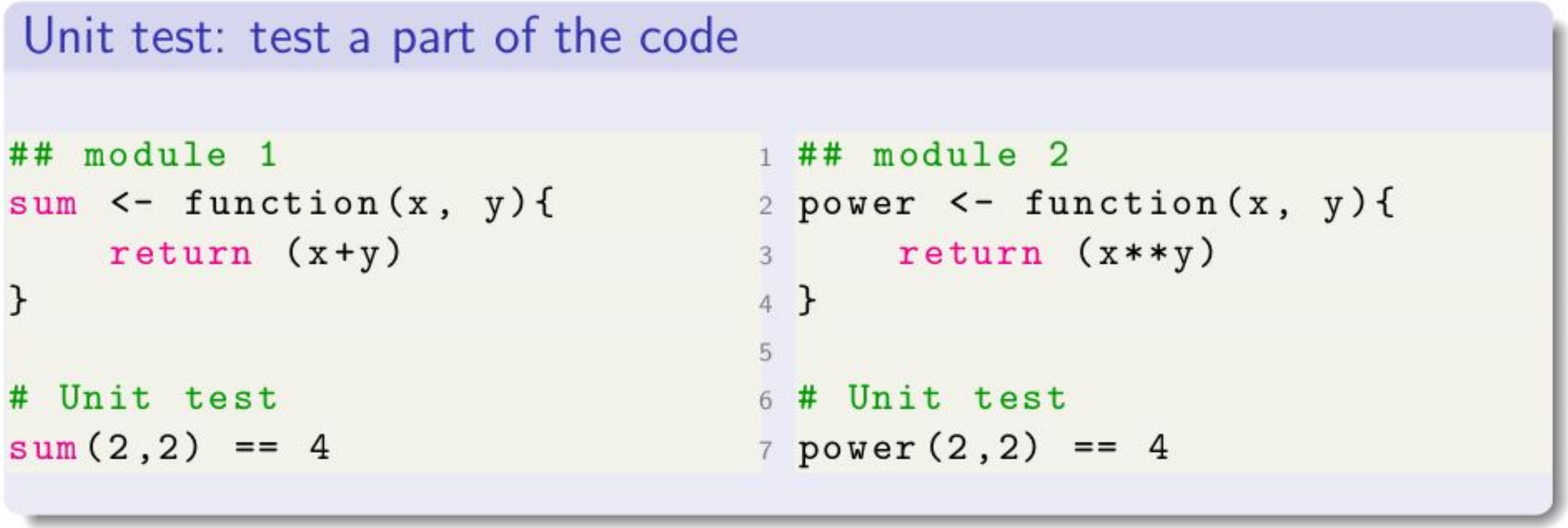
Simplify and avoid niche or hard-to-install third-party code libraries

Verify your code’s portability by running it in a range of computing environments

Automate the test of your code with **continuous-integration** services(eg. Travis CI)

https://www.nature.com/articles/d41586-020-02462-7

Tests - Continous Integration



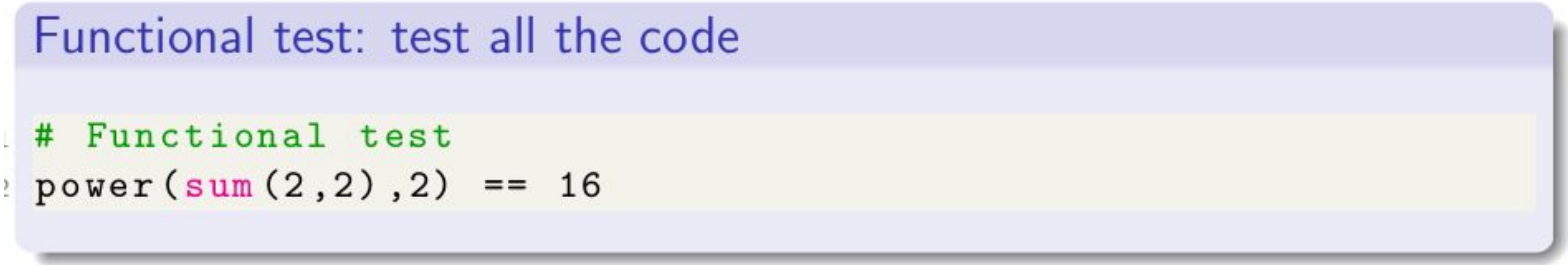
Unit test: test a part of the code

```
## module 1
sum <- function(x, y){
    return (x+y)
}

# Unit test
sum(2,2) == 4
```

```
## module 2
power <- function(x, y){
    return (x**y)
}

# Unit test
power(2,2) == 4
```



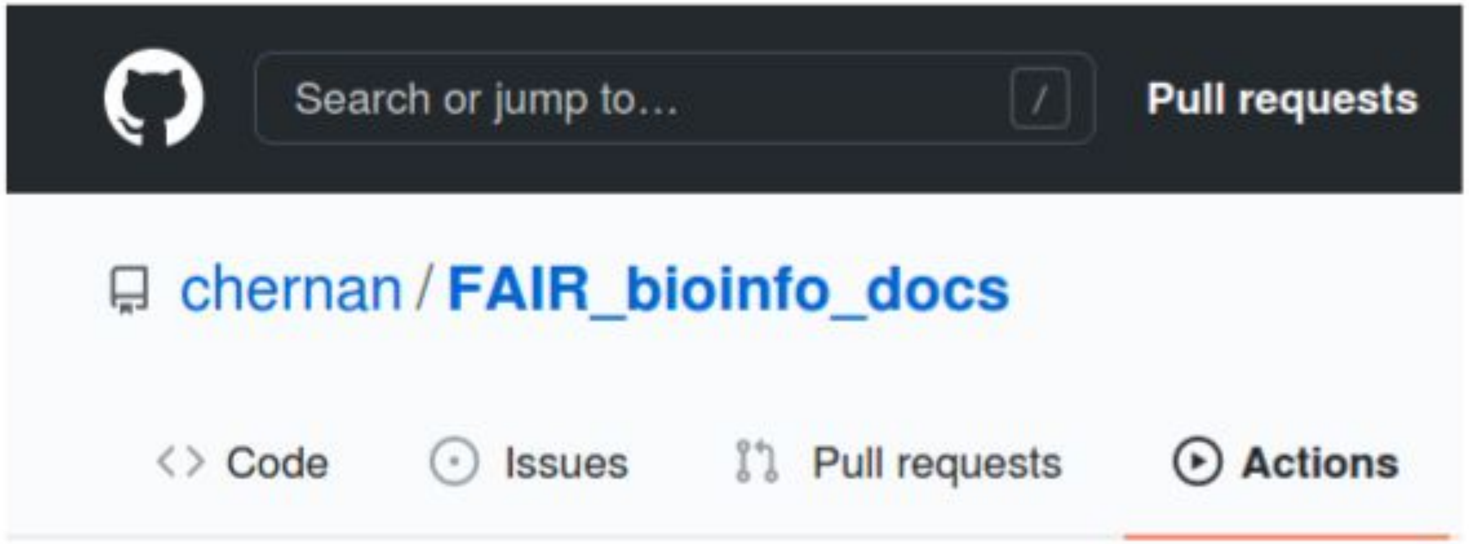
Functional test: test all the code

```
# Functional test
power(sum(2,2),2) == 16
```

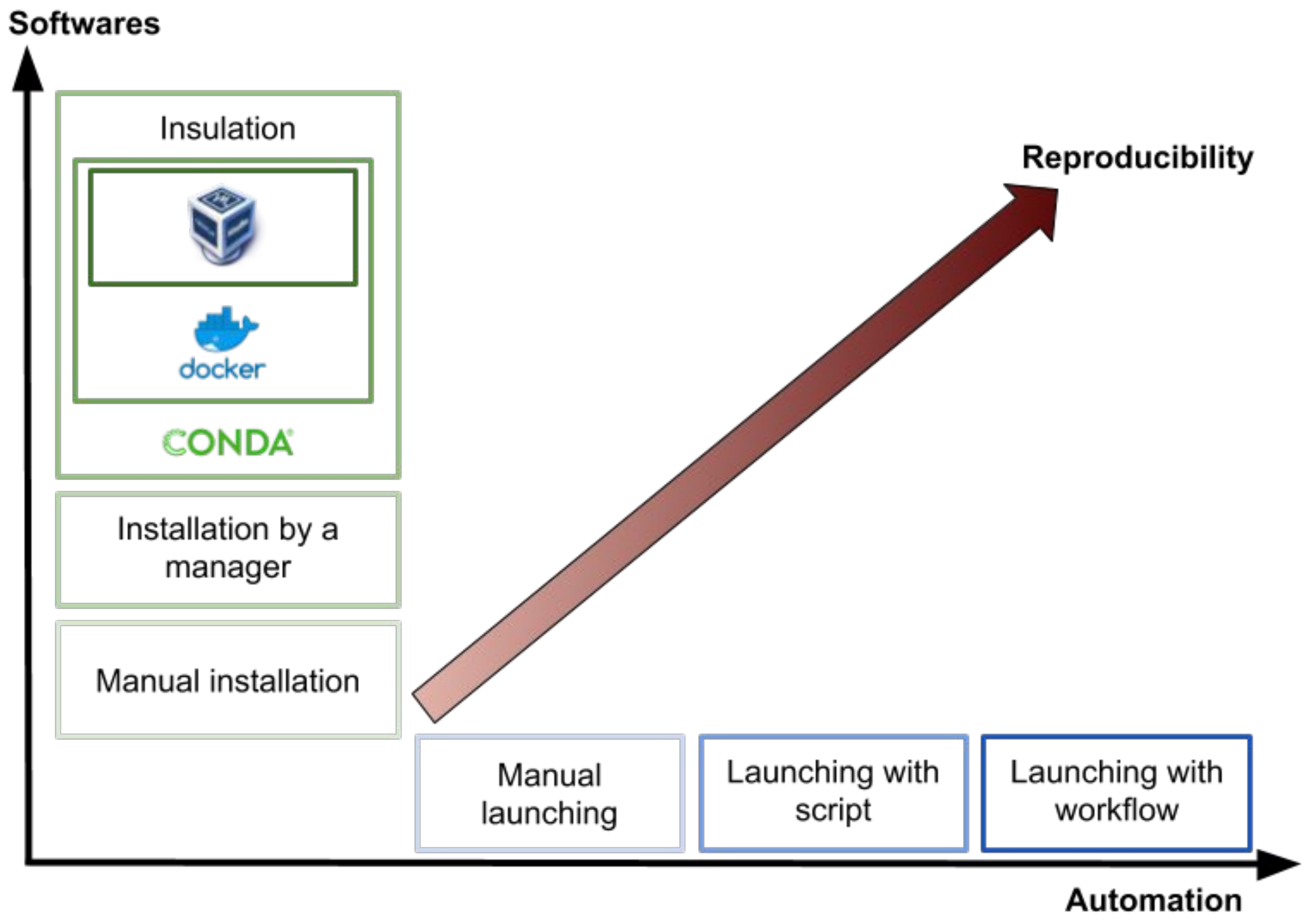
Continuous Integration

Automated verification each time the source code is modified that the modifications do not produce:

- any regression in the developed application
- any change in the results obtained


Search or jump to...
Pull requests
chernan / FAIR_bioinfo_docs

Reproducibility: a multidimensional & a multi-level process



FAIR_bioinfo limits

Automation

Manual command lines

↓

Write a shell script

↓

Use a workflow manager

↓

Tests and continuous integration (*)

User analysis (trial-and-error)

Offer a GUI (eg. with R-Shiny) (*)

↓

Save and re-import choices (*)

Softwares

Local installation

↓

Package manager

↓

Conda environment

↓

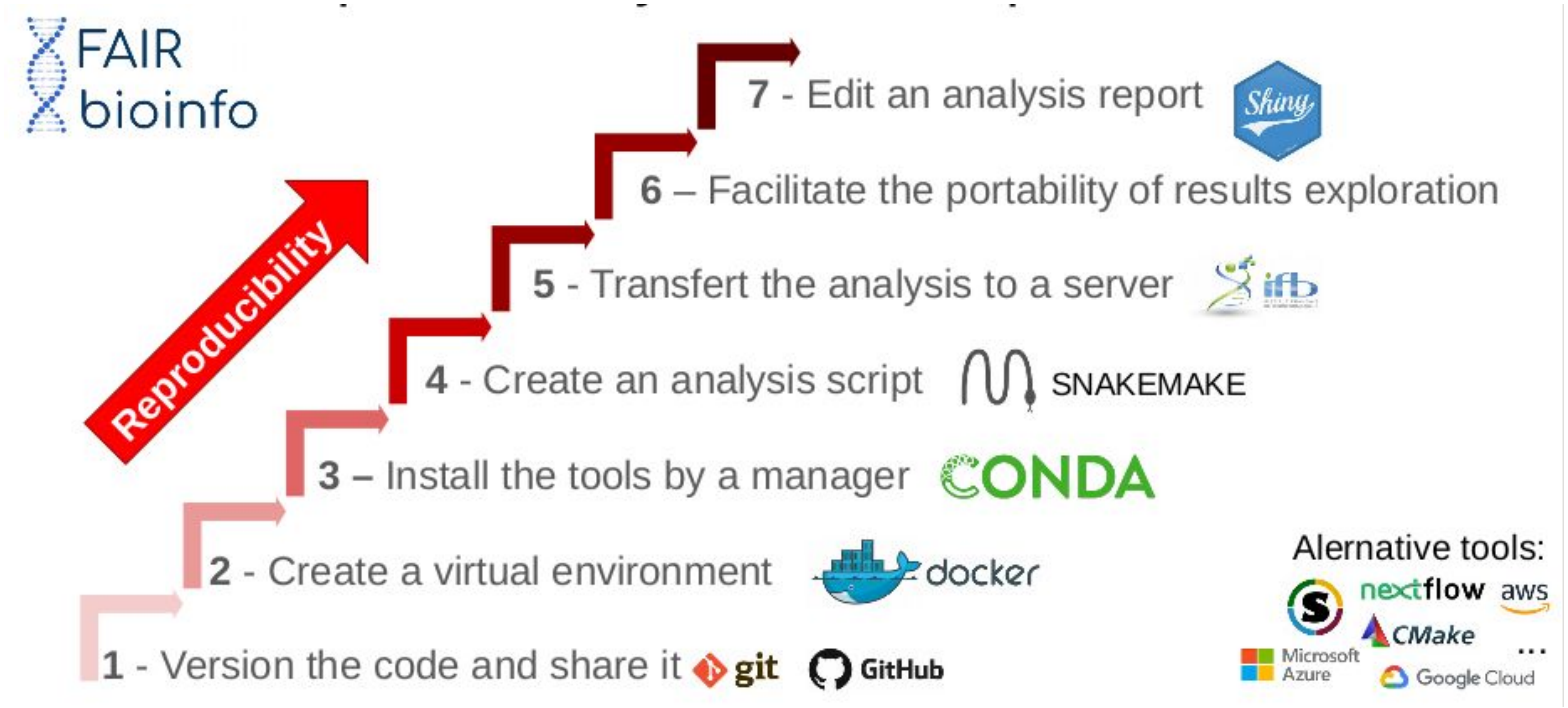
Image / container

↓

Virtual machine (*)

(*) not in the course

FAIR_bioinfo (v1)



FAIR
bioinfo
Reproducibility
7 - Edit an analysis report
Shiny
6 – Facilitate the portability of results exploration
5 - Transfert the analysis to a server
ifb
4 - Create an analysis script
SNAKEMAKE
3 – Install the tools by a manager
CONDA
2 - Create a virtual environment
docker
1 - Version the code and share it
git
GitHub
Alernative tools:
nextflow
aws
CMake
Microsoft Azure
Google Cloud
...

Reproducibility: how far?

Reproducibility to the exact bit?
container uses some resources of the support machine
⇒ version control of the env. (Nix, Guix)

HPC and parallelization?
loss of computational order, multithreading, identical hardware?
⇒ …

Usability, effort/cost & simplicity
Reproducibility to the exact bit

Thanks

Pedagogical team (our guardian angels): Yousra, Hélène

IFB Core Cluster taskforce: Julien, Gildas, and all those who provide in the shadows

Helpers: Emilie, Pauline, Hugo

Organisations: CNRS, INRAE, IFB, I2BC, Paris Saclay University

INSTITUT FRANÇAIS DE BIOINFORMATIQUE