

# Software Environments, Notebooks and Containers

(for Bioinformatics and Computational Biology)

Naples, November 26th, 2024

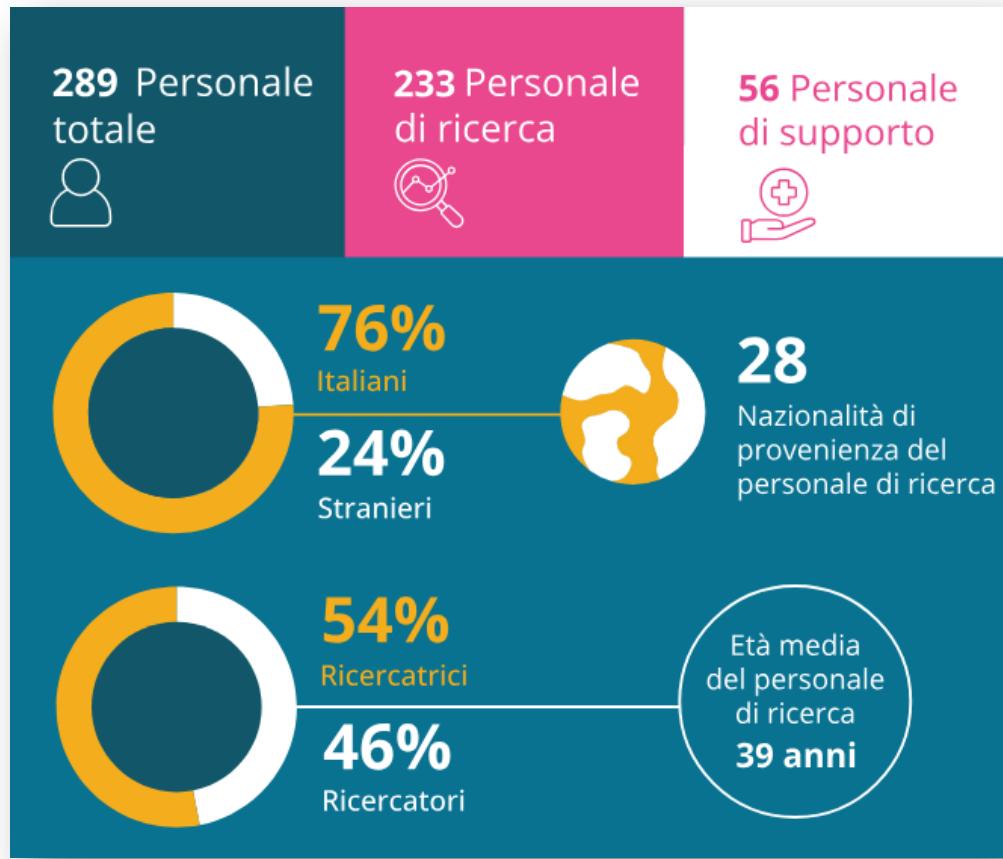


# Overview

- Let's get introduced
- Reproducibility in (computational) Science
- Software environments: Conda
- Go interactive with Jupyter Notebook
- Freeze across space and time with Containers
- Best practices & collaboration
- hands on through out the presentation

# IFOM - ETS

## The AIRC Institute of Molecular Oncology



# Research Computing & Data Science

**Raoul J.P. Bonnal**  
Head of RCDS



@raoulbonnal

**Cristiano Petrini**  
Bioinformatics Engineer



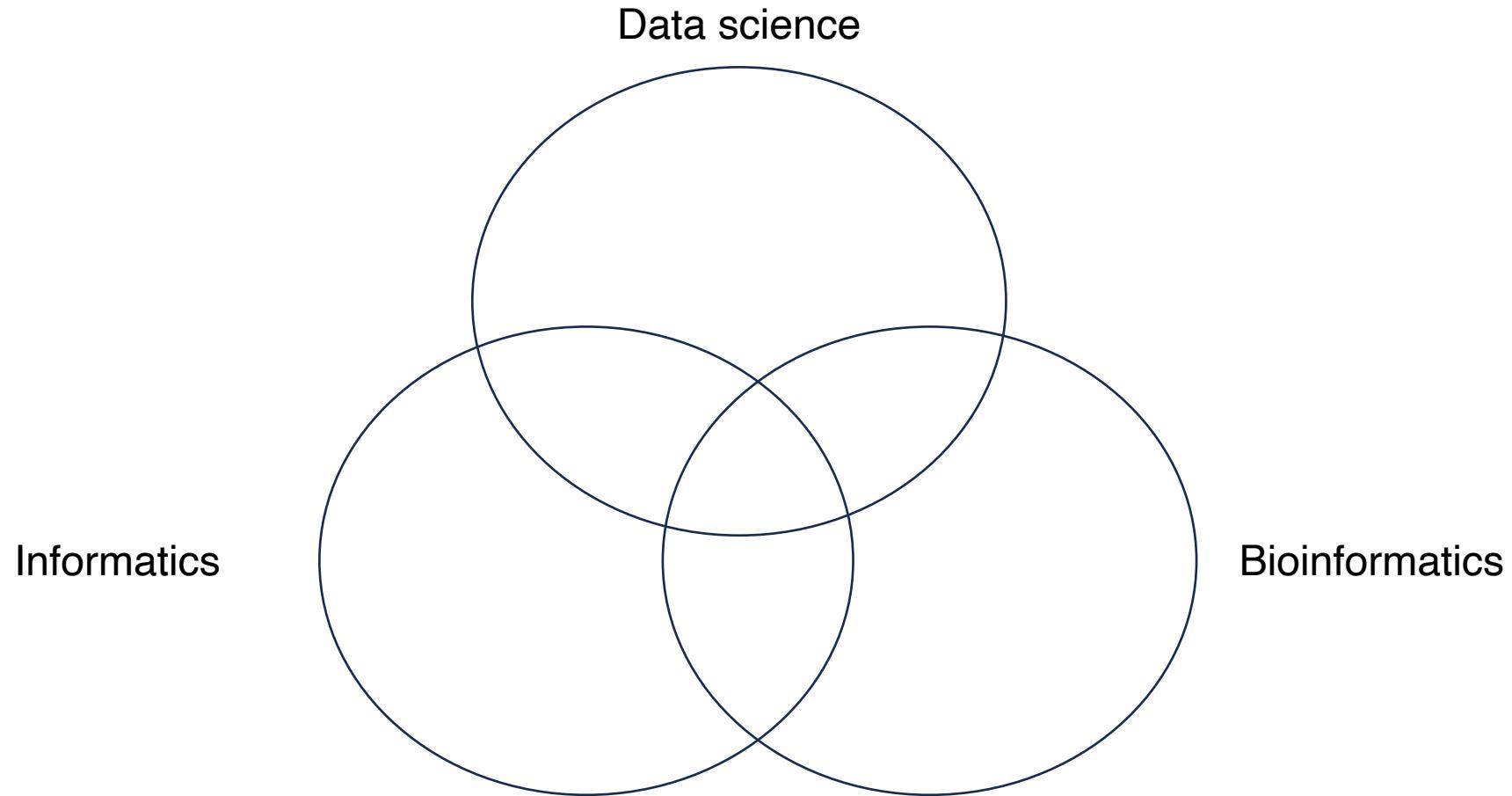
@cristiano-petrini-clkrs

**Riccardo L. Rossi**  
Bioinformatics Scientist



@ricrossi

# Tell us about you and where you are!



# Reproducibility in *(computational)* Science

The challenge of reproducible research

# Reproducible research

The term reproducible research refers to **the idea that scientific results should be documented in such a way that their deduction is fully transparent**. This requires a detailed description of the methods used to obtain the data and making the full dataset and the code to calculate the results easily accessible.



Wikipedia

[https://en.wikipedia.org › wiki › Reproducibility](https://en.wikipedia.org/wiki/Reproducibility) ::

[Reproducibility - Wikipedia](#)

**Your research is reproducible if  
- and only if - you can  
repeatedly and easily  
reproduce it**

**(and you are *another* person in 6 months time)**

# Reproducibility in Computational Science

- Unique Challenges of Code and Algorithms: Complex workflows that may **not be adequately documented**.
- Reproducibility as a Computational Requirement: Need for consistent outputs given **identical inputs**.
- **Software Environment Variability**: Differences in libraries, dependencies, tools, OS, hardware,...



# Software Environment Variability

## 1. Operating System Differences

- Scripts and tools often behave differently across operating systems (Linux, macOS, Windows).
- This variability introduces challenges when sharing workflows between researchers using different platforms.

## 2. Dependency Management Issues

- Software tools and workflows depend on specific versions of libraries, frameworks, and dependencies.
- Variations in versions can lead to differences in outputs, even with the same dataset and code.

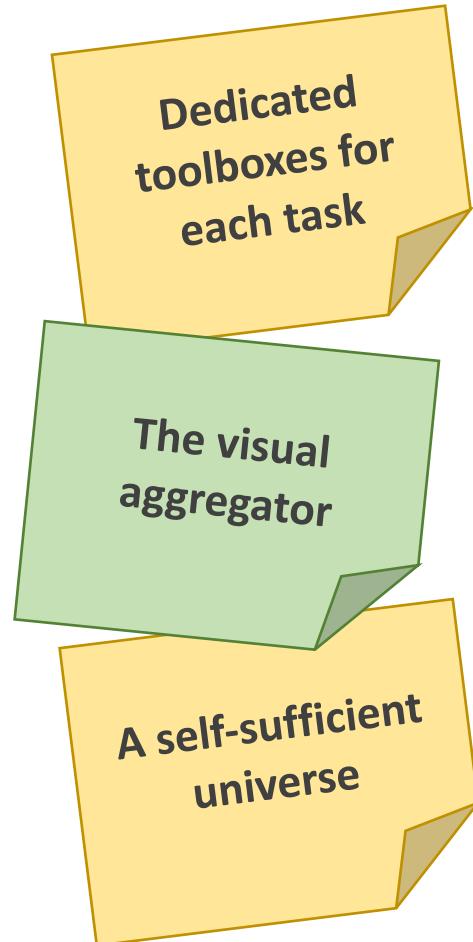
## 3. Hardware-Specific Variability

- Computational performance and precision may depend on the hardware used.
- Some analyses might fail or provide inconsistent results if moved to a less or differently capable system.

## 4. Network and Access Issues

- Cloud services or remote dependencies may change over time, making workflows break without notice.
- External API calls or dependency links can become unavailable or version-incompatible.

# Tools to mitigate variability



- **Software environments**

- Tools like **Conda** and virtual environments ensure dependency versions are pinned and reproducible.
- Configuration files like `requirements.txt` or `environment.yml` standardize environment setup.

- **Notebook Environments**

- Platforms like **Jupyter** provide a controlled, shareable, and interactive environment, reducing variability.

- **Containerization**

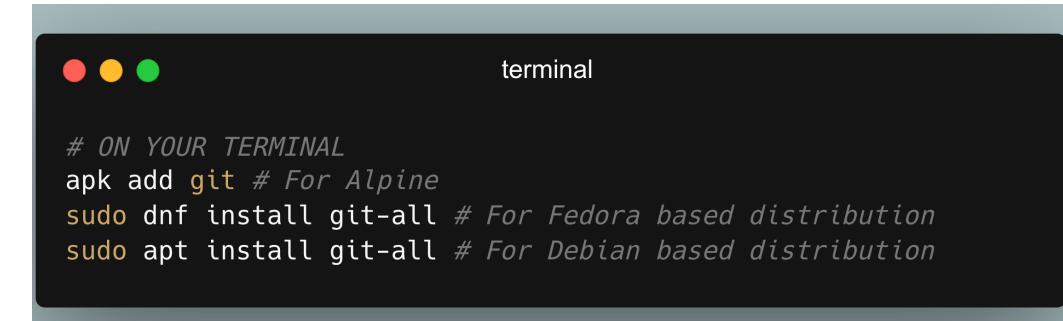
- Tools like **Docker** encapsulate the complete software environment, including dependencies, operating systems, and configurations.



# GitHub, get the workshop material

<https://github.com/ifom/elixir-italy-bbcc24-reproducibility>

The screenshot shows a GitHub repository page for 'ifom/elixir-italy-bbcc24-reproducibility'. The left sidebar lists files uploaded via Dockerfile, including 'Instruction to dockerfile', 'README.md', 'imdb\_R-based.ipynb', 'imdb\_analysis.R', 'imdb\_python-based\_analysis.ipynb', 'saccaromices\_mile\_correlation\_anl...', 'saccharomyces\_mile.csv', 'small\_title\_basics.csv.zip', and 'title.ratings.tsv'. The right sidebar displays repository statistics: 0 stars, 2 watching, 0 forks, and a report repository button. It also shows sections for 'Releases' (no releases published) and 'Packages' (no packages published), with links to 'Create a new release' and 'Publish your first package'. A 'Languages' section at the bottom shows Jupyter Notebook at 99.7% and R at 0.3%. The main content area features the text 'This is the course repository!'.



Dedicated  
toolboxes for  
each task

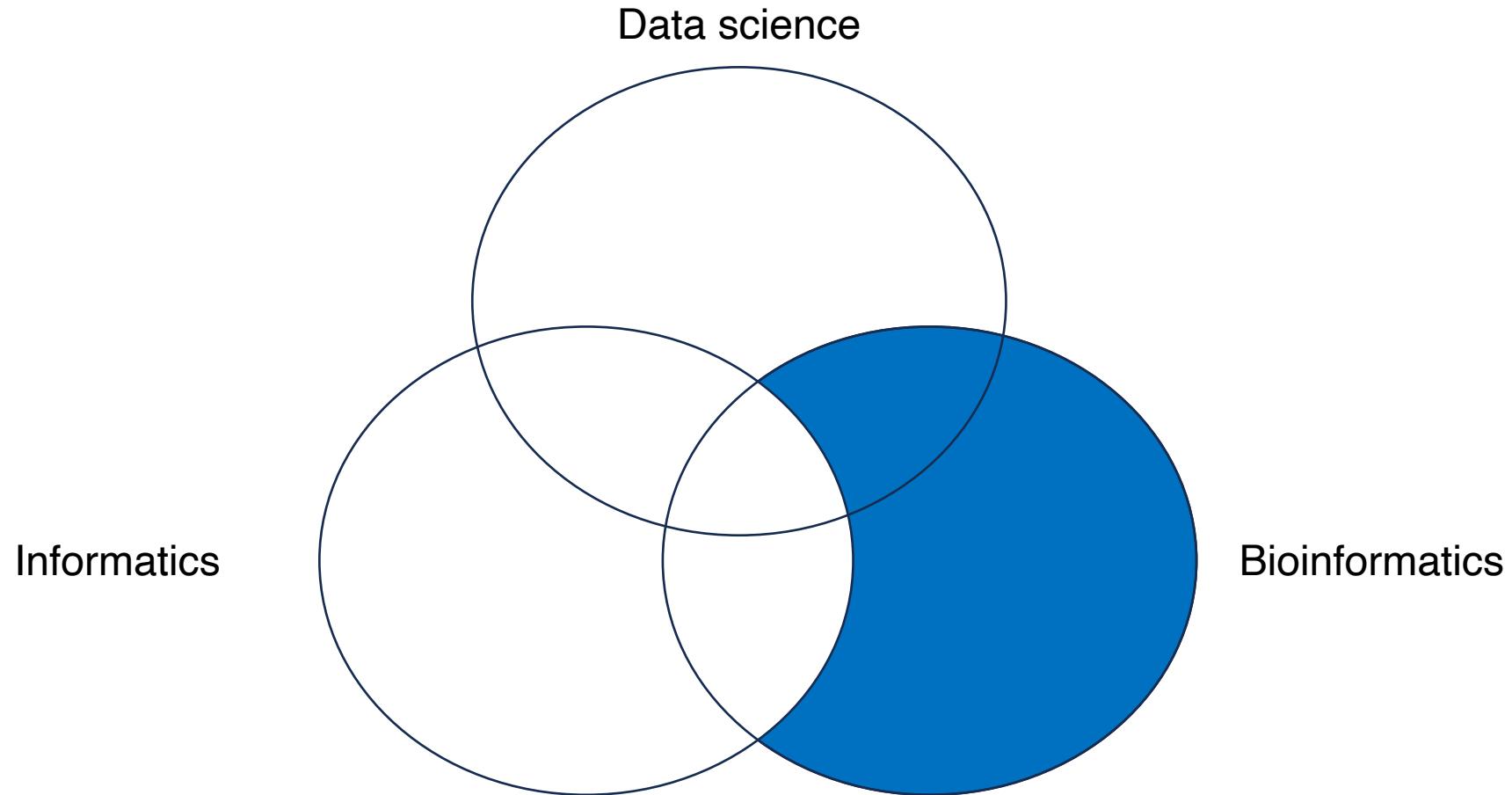
# Conda environments

Conda & companions

# Install and organize software

BBCC2024, RCDS

# Topic domain: bioinformatics



# Bioinformaticians are not system engineers!

- Bioinformaticians need software
- Software do not need bioinformaticians

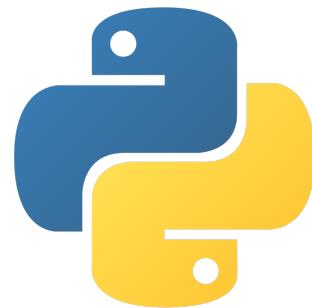
# Bioinformaticians are not system engineers!

- Bioinformaticians need software
- Software ~~do~~<sup>should</sup> not need bioinformaticians

# Languages often have their own repos



CRAN, Bioconductor,  
GitHubs,...

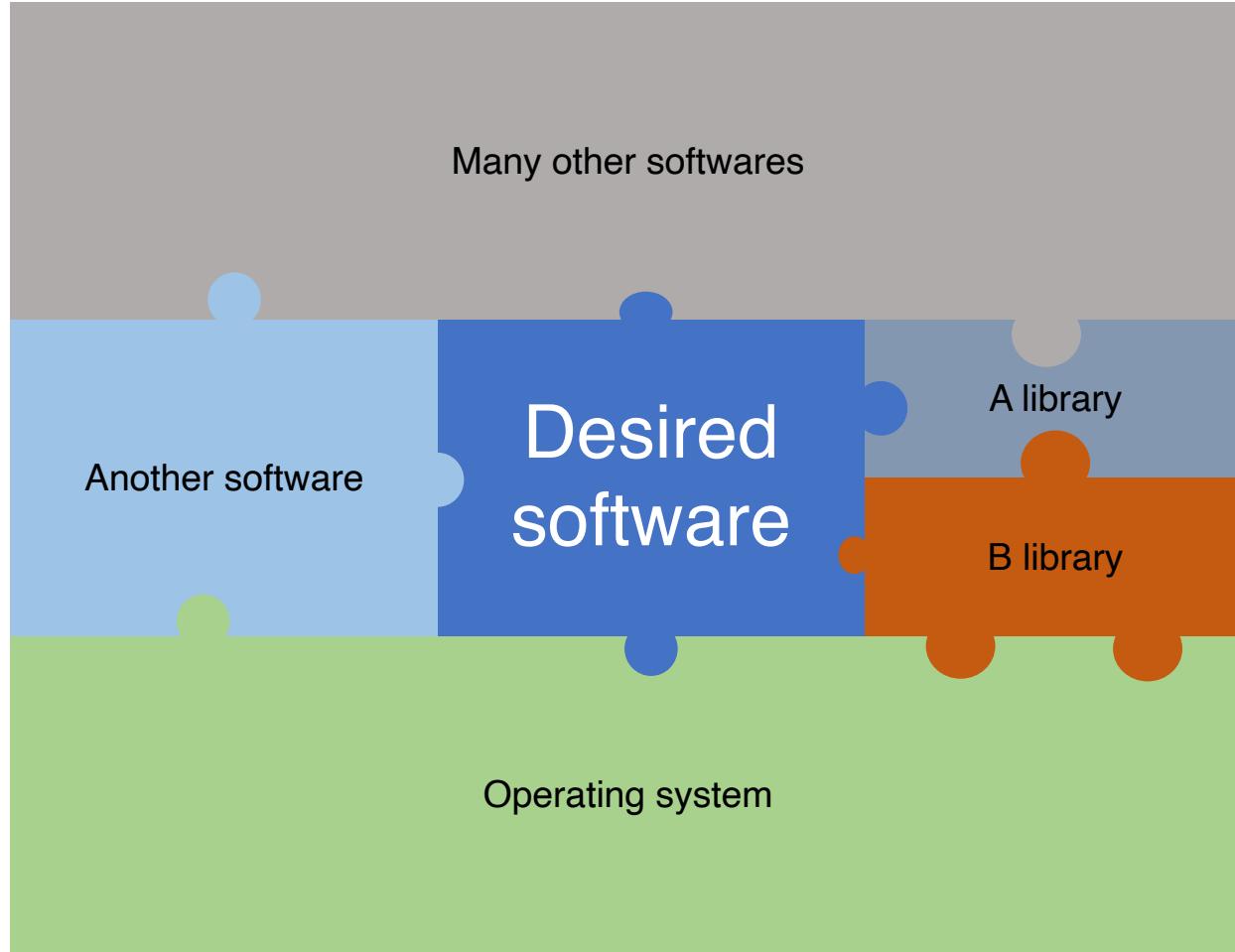


Pip, GitHubs, ...

# Installing software



# Installing software



# Installing software

*It may require another software to properly work*

Many other softwares

Another software

Desired software

A library

B library

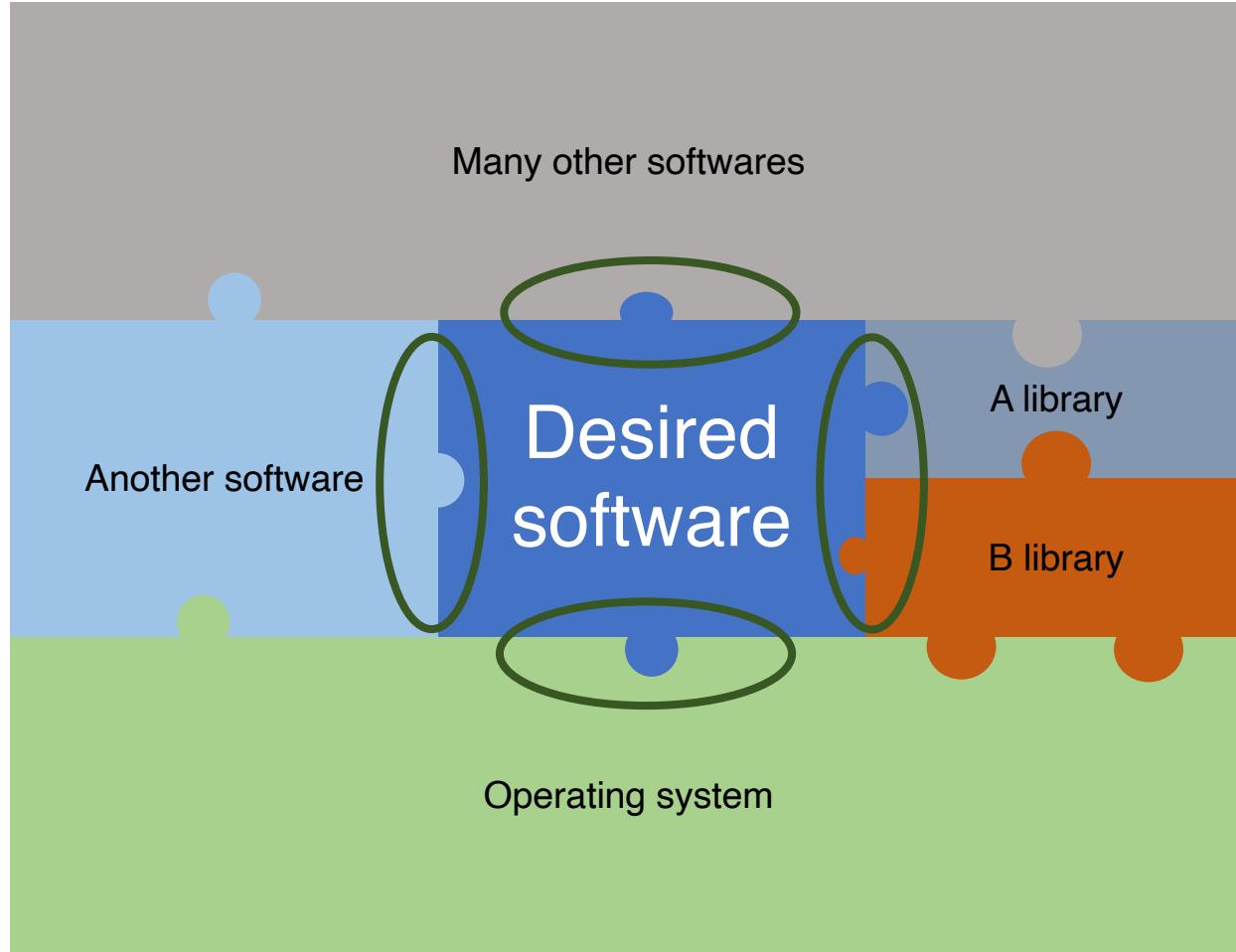
Operating system

*It should be compatible with other software*

*It may require installing other libraries or they should centrally installed*

*Operating system compatibility*

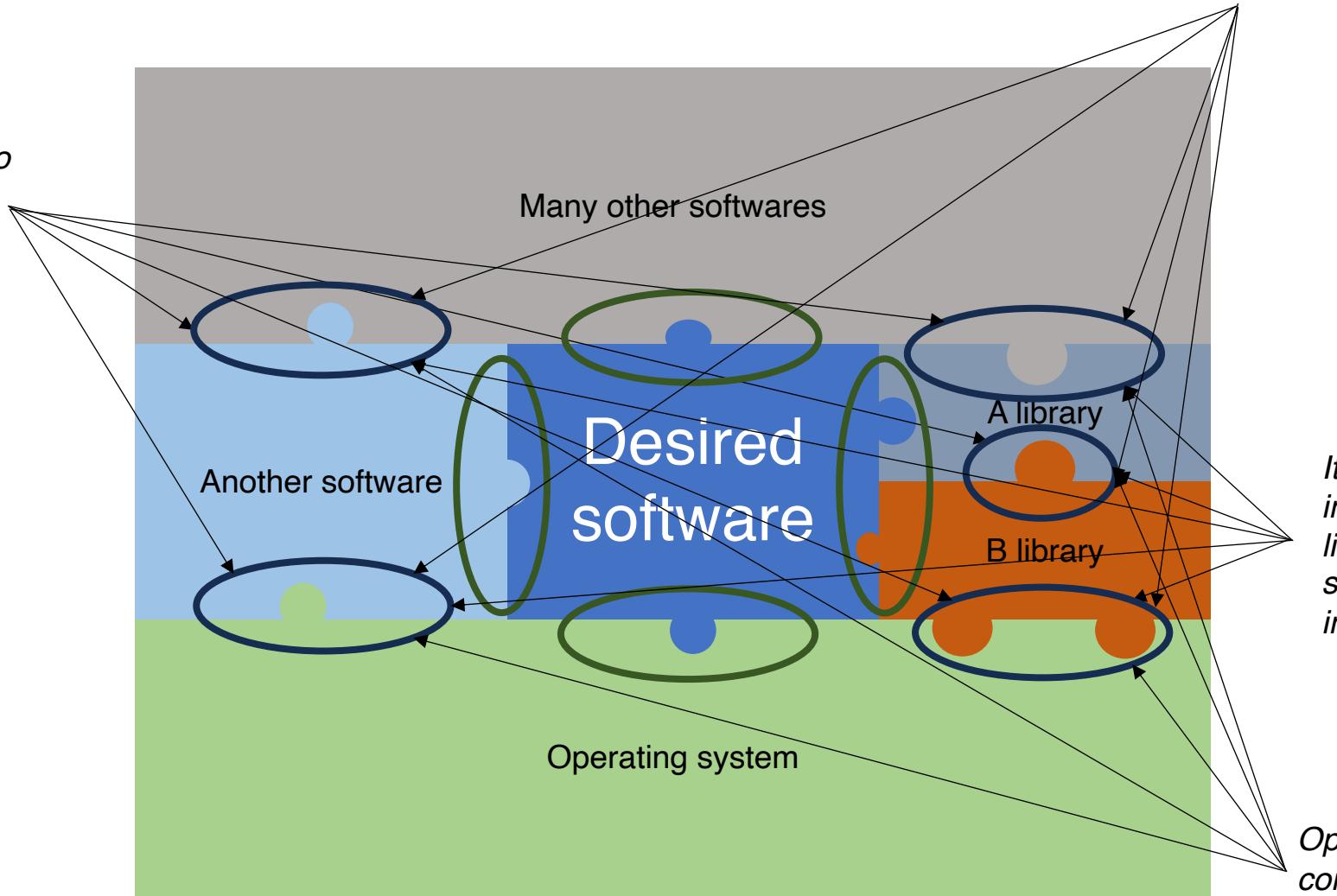
# Installing software



# Installing software

*It may require another software to properly work*

*It should be compatible with other software*



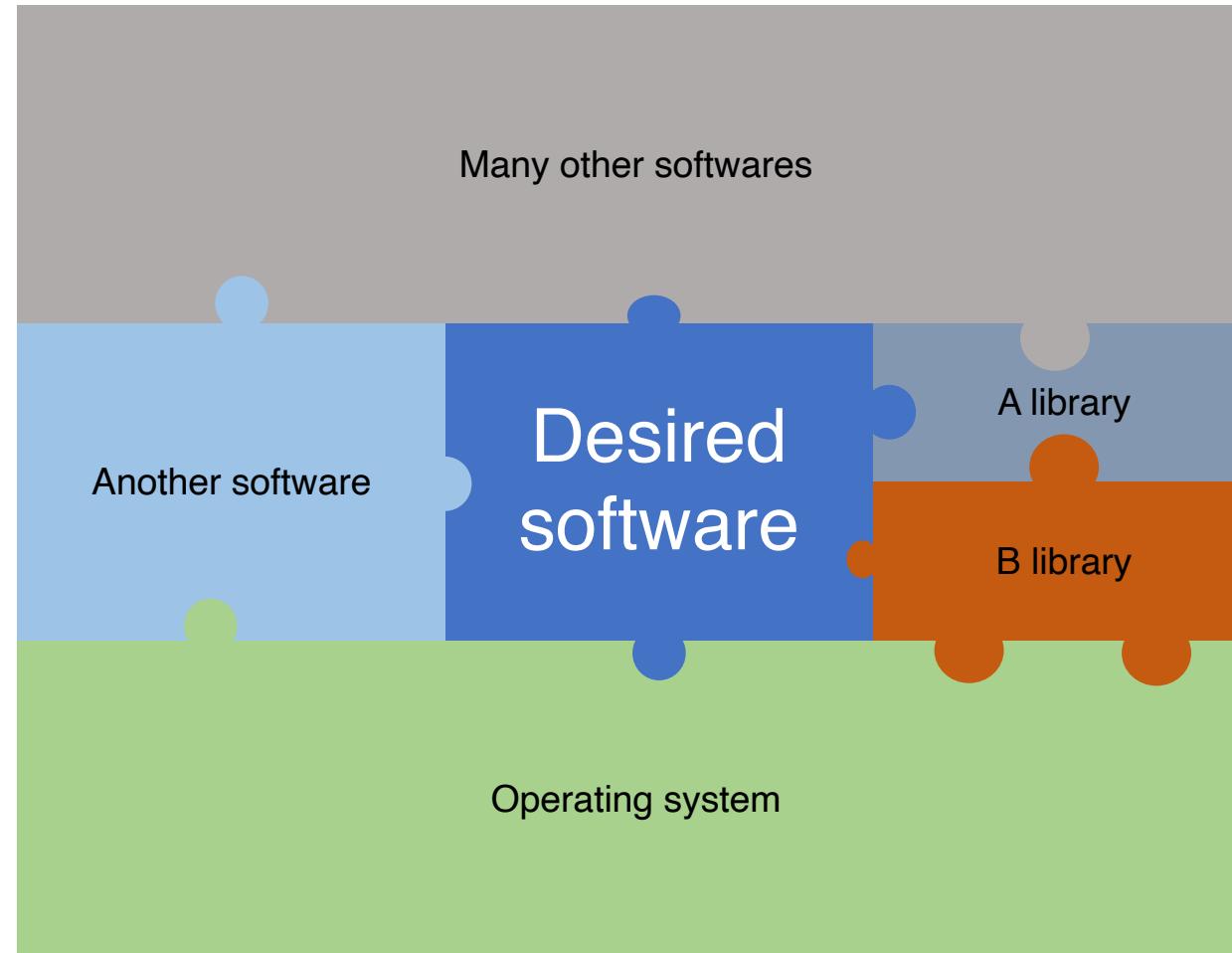
*It may require installing other libraries or they should centrally installed*

*Operating system compatibility*

# Installing software: conda

Conda is an open source **package management** system and **environment management** system for installing multiple versions of software packages and their dependencies and switching easily between them.

It works on Linux, OS X and Windows, and was created for Python programs but can package and distribute any software.

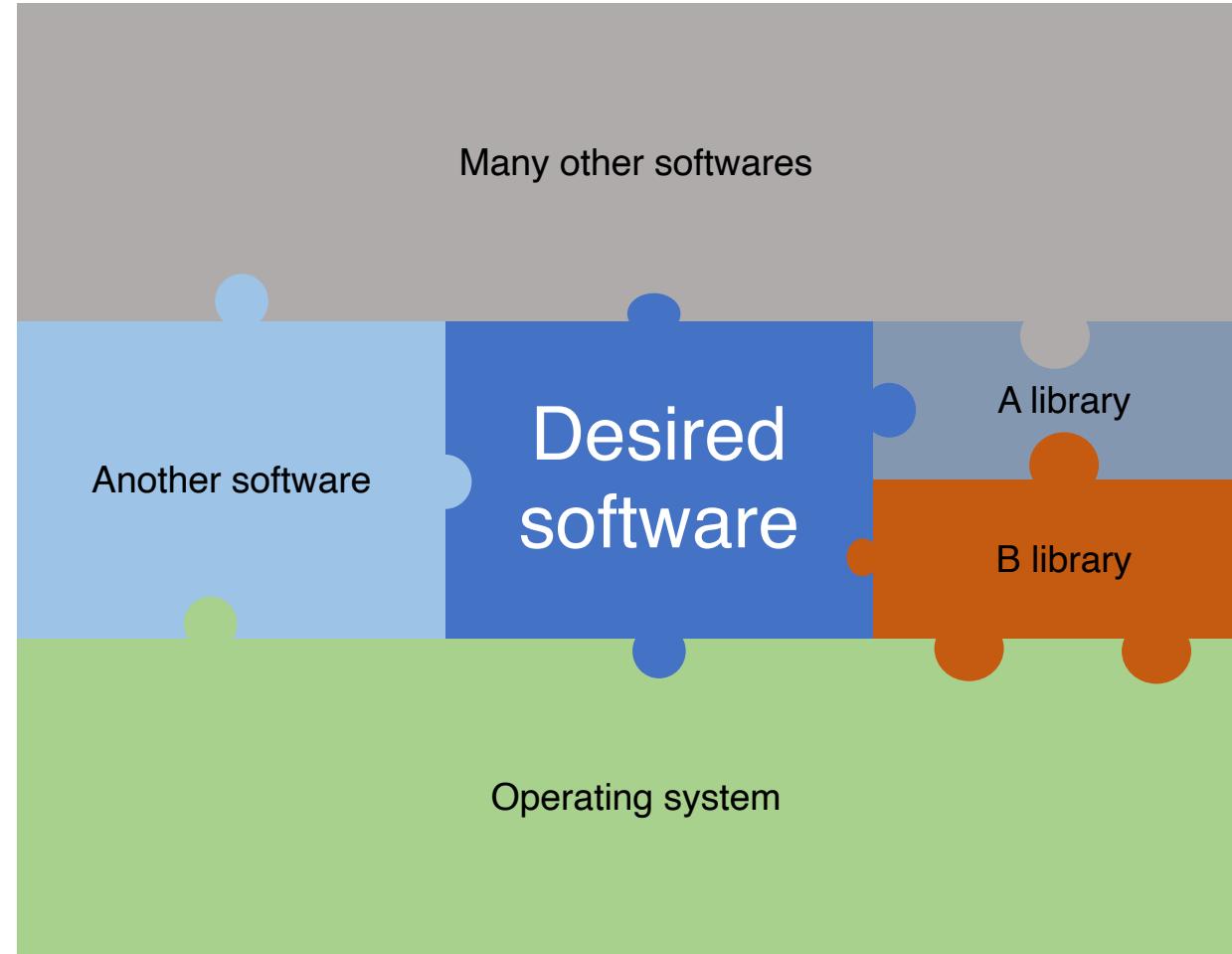


- Conda:*
- *solves software compatibilities*
  - *installs missing softwares*
  - *installs required libraries*

# Installing software: conda

Conda is an open source package management system and environment management system for installing multiple versions of software packages and their dependencies and switching easily between them.

It works on Linux, OS X and Windows, and was created for Python programs but can package and distribute any software.



## Conda:

- *solves software compatibilities*
- *installs missing softwares*
- *installs required libraries*

# Conda is a *package manager*

- **Why Conda Was Created**

- Managing dependencies for Python and scientific libraries was challenging.
- Anaconda Inc. (formerly Continuum Analytics) introduced Conda in 2012 to address this issue.

- **Goals of Conda**

- Provide a cross-platform package manager.
- Handle complex dependencies and allow reproducible environments.

# Evolution and Development of Conda

- Expansion Beyond Python
  - Initially Python-focused, now supports R, Ruby, Lua, and other languages.
- Integration of Data Science Tools
  - Became the backbone for scientific computing environments.
  - Widely adopted in bioinformatics and computational biology.
- Key Milestones in Development
  - Introduction of the **Conda Forge** repository.
  - Development of tools like **Miniconda** for lightweight installations.

# Then there are *package repositories*

- **Anaconda**

- <https://anaconda.org/anaconda>

- **Bioconda**

- <https://anaconda.org/bioconda>

- **Conda-forge**

- <https://anaconda.org/conda-forge>

**Profile**

Anaconda's internal mirroring channel



**Organization** created on Sep 18, 2015

Anaconda, Inc.  
Austin, Texas

The packages on this channel are covered by the Anaconda repository Terms of Service. Among other things, the ToS prohibits heavy commercial use and mirroring by any third party for commercial purposes.

**Profile**

bioconda



**Organization** created on Sep 11, 2015

Bioconda is a distribution of bioinformatics software realized as a channel for the versatile Conda package manager.

**Profile**

conda-forge



**Organization** created on Apr 11, 2015

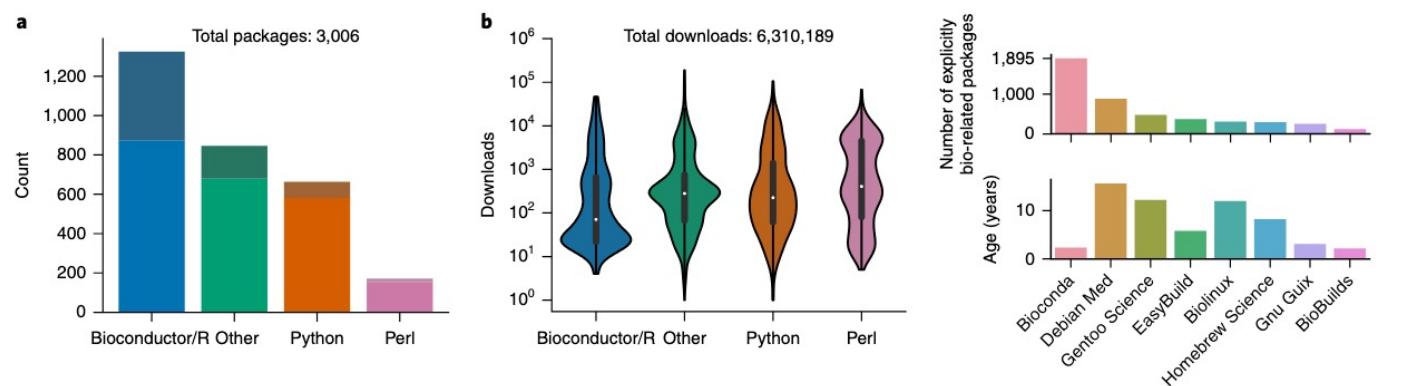
A community-led collection of recipes, build infrastructure, and distributions for the conda package manager.

Correspondence | Published: 02 July 2018

## Bioconda: sustainable and comprehensive software distribution for the life sciences

[Björn Grünig](#), [Ryan Dale](#), [Andreas Sjödin](#), [Brad A. Chapman](#), [Jillian Rowe](#), [Christopher H. Tomkins-Tinch](#), [Renan Valieris](#), [Johannes Köster](#)✉ & [The Bioconda Team](#)

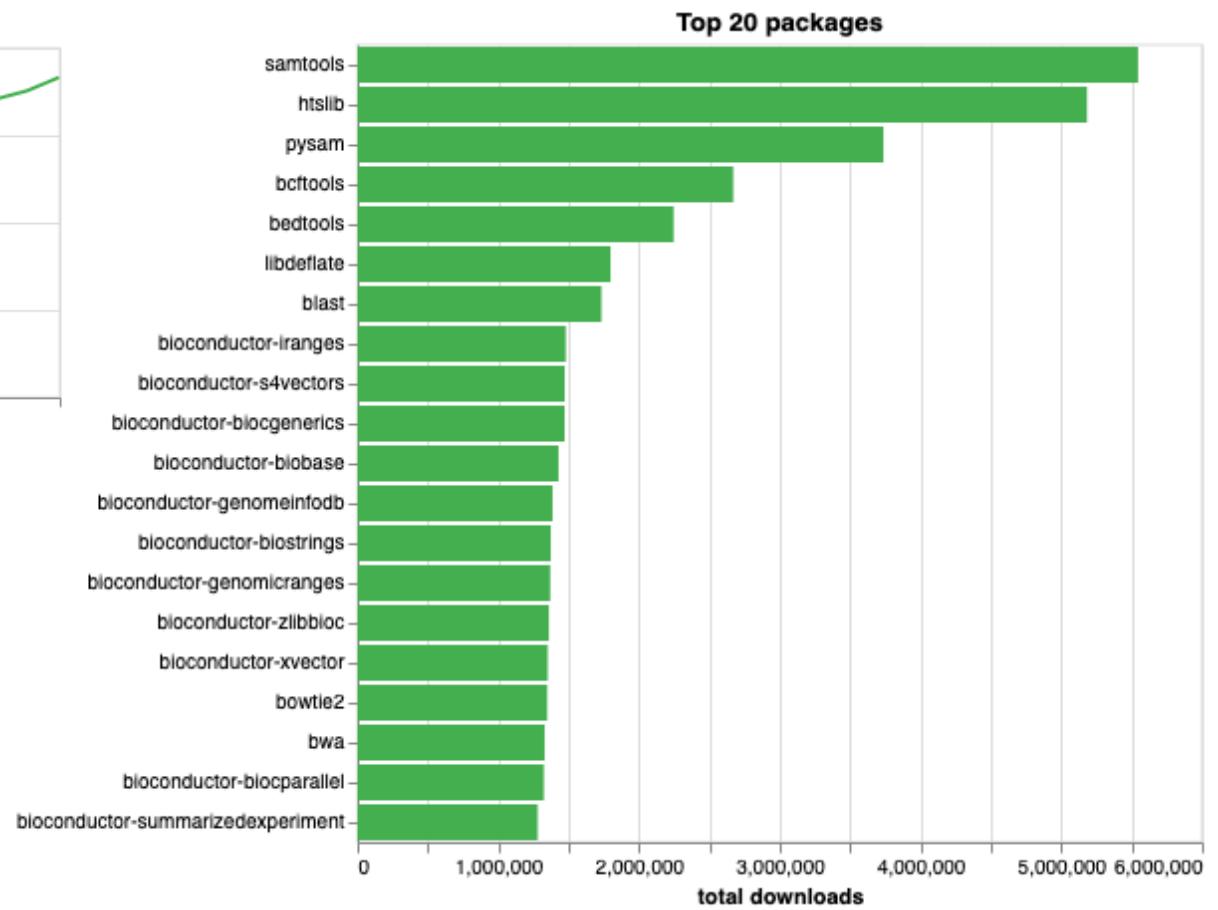
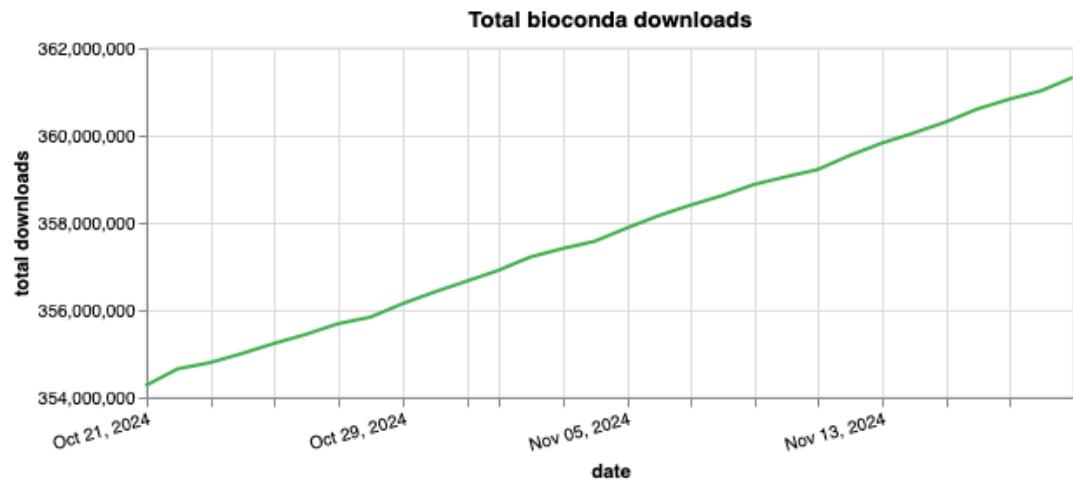
*Nature Methods* 15, 475–476 (2018) |



**Fig. 1 | Package numbers and usage.** **a**, Package count per language ecosystem (saturated colors on the lower portions of the bars represent explicitly life-science-related packages). **b**, Distribution of per-package downloads, separated by language ecosystem. The term “other” encompasses all packages that do not fall into one of the specific categories named. White dots represent the mean; dark bars represent the interval between upper and lower quartiles. **c**, Comparison of the number of explicitly life-science-related packages in Bioconda with that in Debian Med (<https://www.debian.org-devel/debian-med>), Gentoo Science Overlay (category sci-biology; <https://github.com/gentoo/sci>), EasyBuild (module bio; <https://easybuilders.github.io/easybuild>), Biolinux<sup>6</sup>, Homebrew Science (tag bioinformatics; <https://brew.sh>), GNU Guix (category bioinformatics; <https://www.gnu.org/s/guix>), and BioBuilds (<https://biobuilds.org>). The lower graph shows the project age since the first release or commit. Statistics obtained 25 October 2017.

# BIOCONDA®

Bioconda lets you install thousands of software packages related to biomedical research using the `conda` package manager.



# Always read the fine print (TOS)

The screenshot shows a website header with the JÜLICH Forschungszentrum logo and the JuRSE navigation bar. The main content is an article titled "Beware: the anaconda is squeezing us". A yellow callout box with the text "Just one example of many" is overlaid on the top right. A red circle highlights the title. A red box highlights the sentence "government entities and non-profit entities with over 200 employees or contractors" in the text below.

**Beware: the anaconda is squeezing us**

If you are a Python user, you know you should always (I mean it!) use [virtual environments](#) to keep things separate, reproducible and [sane](#). There are plenty of options to choose from such as [venv](#) or [virtualenv](#). A somewhat broader tool for this is [conda](#), "a community supporting a language-agnostic, multi-platform package management ecosystem for projects of any size and complexity". There are a lot of things you can do with this, but in particular it is very well suited for working with Python environments.

Sounds good? It is, but there is a catch. The main driver of using conda in data science with Python is [Anaconda Inc.](#) and while being community-driven, they need to make money at some point. To do that, they changed their Terms of Service (ToS) in 2020 (already), insisting that "government entities and non-profit entities with over 200 employees or contractors" must pay. That includes

The good news is: there are alternatives and actually very good ones. A quite prominent one is [mamba, The Fast Cross-Platform Package Manager](#). While primarily targeting speed and reliability, this package/environment manager has `conda-forge` as default channel to pull the packages from. This community-driven channel is independent of Anaconda and avoids any ToS breaches, while keeping the great features of such a package/environment system based on `conda`. Ways to use `mamba` include [mini-forge](#) or [micromamba](#), for example. Be aware to remove the default channel also from existing environments and do not use other Anaconda products like the Anaconda Distribution.

**Bottom line:** you have a choice and this is good news. Use virtual environments when working with Python and when you rely on `conda`, **do not use Anaconda services/channels**. If you need help, get in touch (see last FAQ below)!

## Anaconda Threatens Legal Action Over Licensing Terms

By Paul Mah    August 21, 2024

<https://www.cdotrends.com/story/4173/anaconda-threatens-legal-action-over-licensing-terms>



## Anaconda puts the squeeze on data scientists now deemed to be terms-of-service violators

Academic, non-profit organizations told to start paying up – or else

Thomas Claburn

Thu 8 Aug 2024 / 12:26 UTC

**UPDATED** Research and academic organizations are just now finding out that they will have to pay for software made by Anaconda, when for years these groups were under the impression it could be used at no cost.

[https://www.theregister.com/2024/08/08/anaconda\\_puts\\_the\\_squeeze\\_on/](https://www.theregister.com/2024/08/08/anaconda_puts_the_squeeze_on/)

IN THE UNITED STATES DISTRICT COURT  
FOR THE DISTRICT OF DELAWARE

ANACONDA, INC., )  
Plaintiff, ) C.A. No. \_\_\_\_\_  
v. )  
INTEL CORPORATION, )  
Defendant. )

**DEMAND FOR JURY TRIAL**

**COMPLAINT**

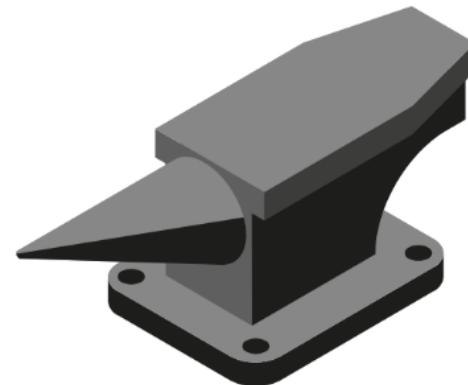
Plaintiff Anaconda, Inc. (“Anaconda” or “Plaintiff”) brings this complaint against Defendant Intel Corporation (“Intel” or “Defendant”) and alleges as follows:

**NATURE OF THE ACTION**

1. This is a civil action for the infringement, under the copyright laws of the United States, 17 U.S.C. § 101, *et seq.*, of proprietary material registered under U.S. Copyright No. TX 9-407-381 (“Asserted Copyright”) in violation of 17 U.S.C. § 501.
2. Anaconda developed a technology that has become critical to the booming artificial intelligence (“AI”) industry—one that is used by over 40 million users at over one million companies worldwide. Intel is one of those companies. For years, Intel bought licenses allowing

# Alternatives exist

- Community led
- Free to use
- Latest (*dev* also) releases



**Community-led recipes,  
infrastructure and  
distributions for conda.**

[Explore conda-forge](#)

[Download Installer](#)

## About conda-forge

[conda-forge](#) is a GitHub organization containing repositories of conda recipes.

# Hands-on: install miniforge3\*



terminal

```
curl -L -O "https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-$(uname)-$(uname -m).sh"  
bash Miniforge3-$(uname)-$(uname -m).sh
```

\**Why miniforge3 after all the slides on conda?*

After 2020, **Anaconda Inc.** (the company that owns conda) **set up a fee** on all the companies/non-academic institutes that use their repositories. The access to the conda **miniforge** repositories **instead** is still **free**.

Installing miniforge3 instead of miniconda3 (conda) we guarantee that the Anaconda Inc. repositories are not in the default repositories used to download software.

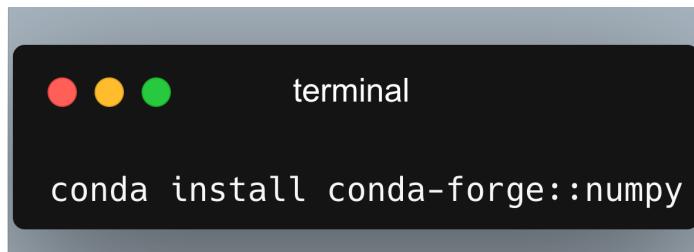
# Hands-on: install *software with miniforge3*



terminal

```
conda install conda-forge::numpy
```

# Hands-on: install software with miniforge3



```
cpetrini@ ~ % conda install conda-forge::numpy
Channels:
- conda-forge
Platform: osx-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /Users/cpetrini/miniforge3

added / updated specs:
- conda-forge::numpy

The following packages will be downloaded:

  package          | build
  --:: --
libcblas-3.9.0    |25_osx64_openblas      15 KB  conda-forge
numpy-2.1.3       |py312hfc93d17_0        7.2 MB  conda-forge

Total:           7.2 MB

The following NEW packages will be INSTALLED:

  libcblas          conda-forge/osx-64::libcblas-3.9.0-25_osx64_openblas
  numpy             conda-forge/osx-64::numpy-2.1.3-py312hfc93d17_0

Proceed ([y]/n)?
```

# Hands-on: install software with miniforge3



terminal  
conda install conda-forge::numpy



Repository collecting the required softwares

```
(base) cpetrini@ ~ % conda install conda-forge::numpy
Channels:
- conda-forge
Platorm: osx-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /Users/cpetrini/miniforge3

added / updated specs:
- conda-forge::numpy

The following packages will be downloaded:

  package          | build
  --:: --
libcblas-3.9.0    |25_osx64_openblas      15 KB  conda-forge
numpy-2.1.3        |py312hfc93d17_0       7.2 MB  conda-forge

Total:           7.2 MB

The following NEW packages will be INSTALLED:

  libcblas          conda-forge/osx-64::libcblas-3.9.0-25_osx64_openblas
  numpy            conda-forge/osx-64::numpy-2.1.3-py312hfc93d17_0

Proceed ([y]/n)?
```

# Hands-on: install software with miniforge3

```
terminal
conda install conda-forge::numpy
```

Packages info

```
(base) cpetrini@ ~ % conda install conda-forge::numpy
Channels:
- conda-forge
Platform: osx-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /Users/cpetrini/miniforge3

added / updated specs:
- conda-forge::numpy

The following packages will be downloaded:


| package       | build             | Total: |             |
|---------------|-------------------|--------|-------------|
| libblas-3.9.0 | 25_osx64_openblas | 15 KB  | conda-forge |
| numpy-2.1.3   | py312hfc93d17_0   | 7.2 MB | conda-forge |


The following NEW packages will be INSTALLED:

libblas      conda-forge/osx-64::libblas-3.9.0-25_osx64_openblas
numpy       conda-forge/osx-64::numpy-2.1.3-py312hfc93d17_0

Proceed ([y]/n)?
```

# Hands-on: install *software with miniforge3, specific version*



terminal

```
conda install conda-forge::r-base=4.3.2
```

# Hands-on: install *software with miniforge3, specific version*



A screenshot of a Mac OS X terminal window titled "terminal". The window shows the command "conda install conda-forge::r-base=4.3.2" entered by the user. The version number "4.3.2" is highlighted with a red oval. A red arrow points from the text "Requested a specific version" below the terminal window to the circled version number.

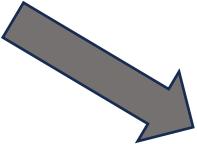
Requested a specific version

# Hands-on: install *software with miniforge3, specific version*



terminal

```
conda install conda-forge::r-base=4.3.2
```



```
The following packages will be DOWNGRADED:
```

cairo	1.18.0-h37bd5c4_3 → 1.18.0-h99e66fa_0
harfbuzz	9.0.0-h098a298_1 → 8.3.0-hf45c392_0
icu	75.1-h120a0e1_0 → 73.2-hf5e326d_0
libglib	2.82.2-hb6ef654_0 → 2.78.4-hab64008_0
libxml2	2.13.4-h12808cf_2 → 2.12.7-hc603aa4_3
pcre2	10.44-h7634a1b_2 → 10.42-h0ad2156_0
texlive-core	20230313-hca7b749_15 → 20230313-ha022968_11

```
Proceed ([y]/n)?
```

# Hands-on: install *software with* miniforge3, R libraries



terminal

```
conda install conda-forge::r-tidyverse
```

or

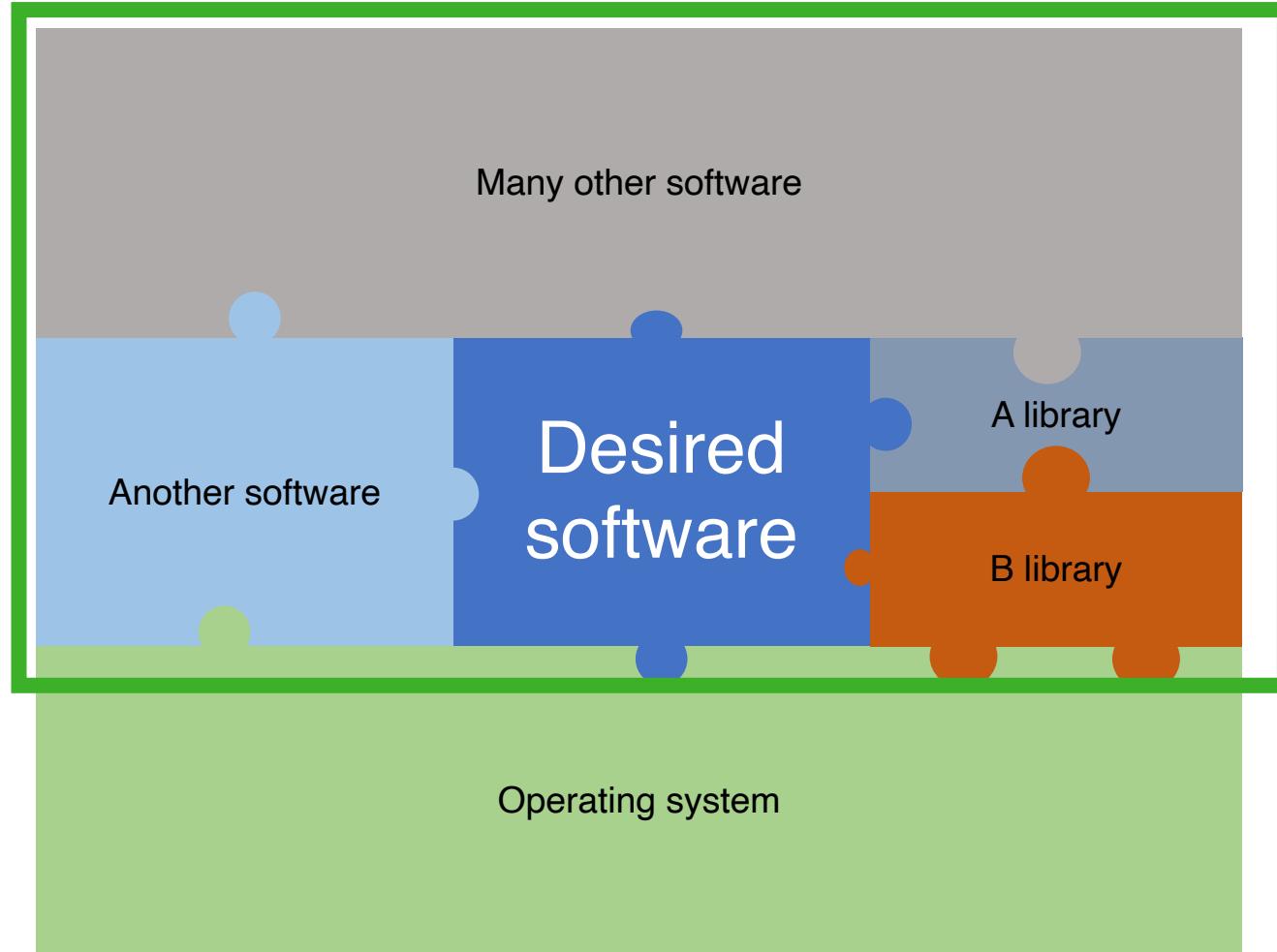
```
# Install from CRAN  
install.packages("tidyverse")
```



What's the best strategy  
to install an R library?

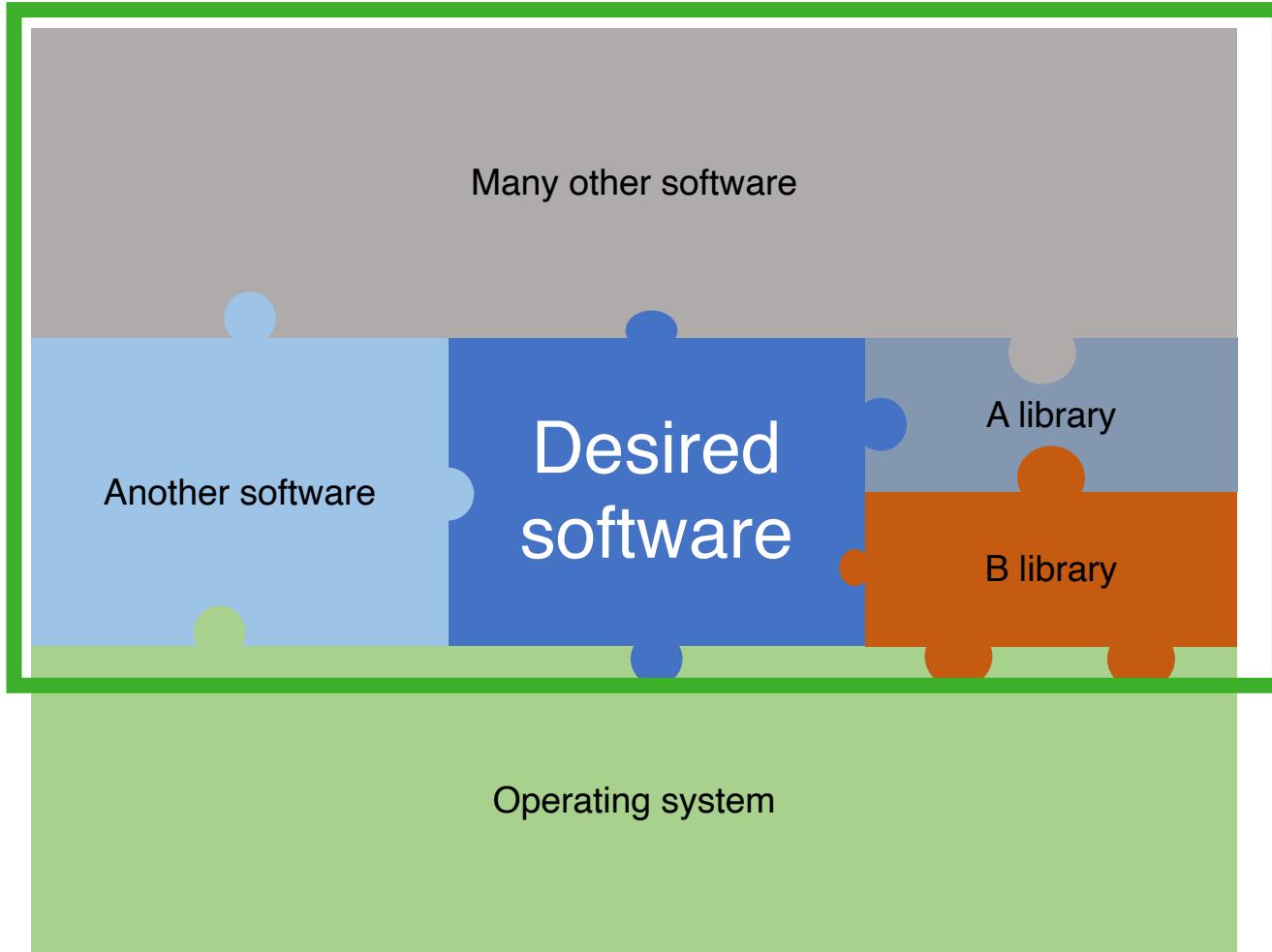
# Software environments with `conda` miniforge

Software environment



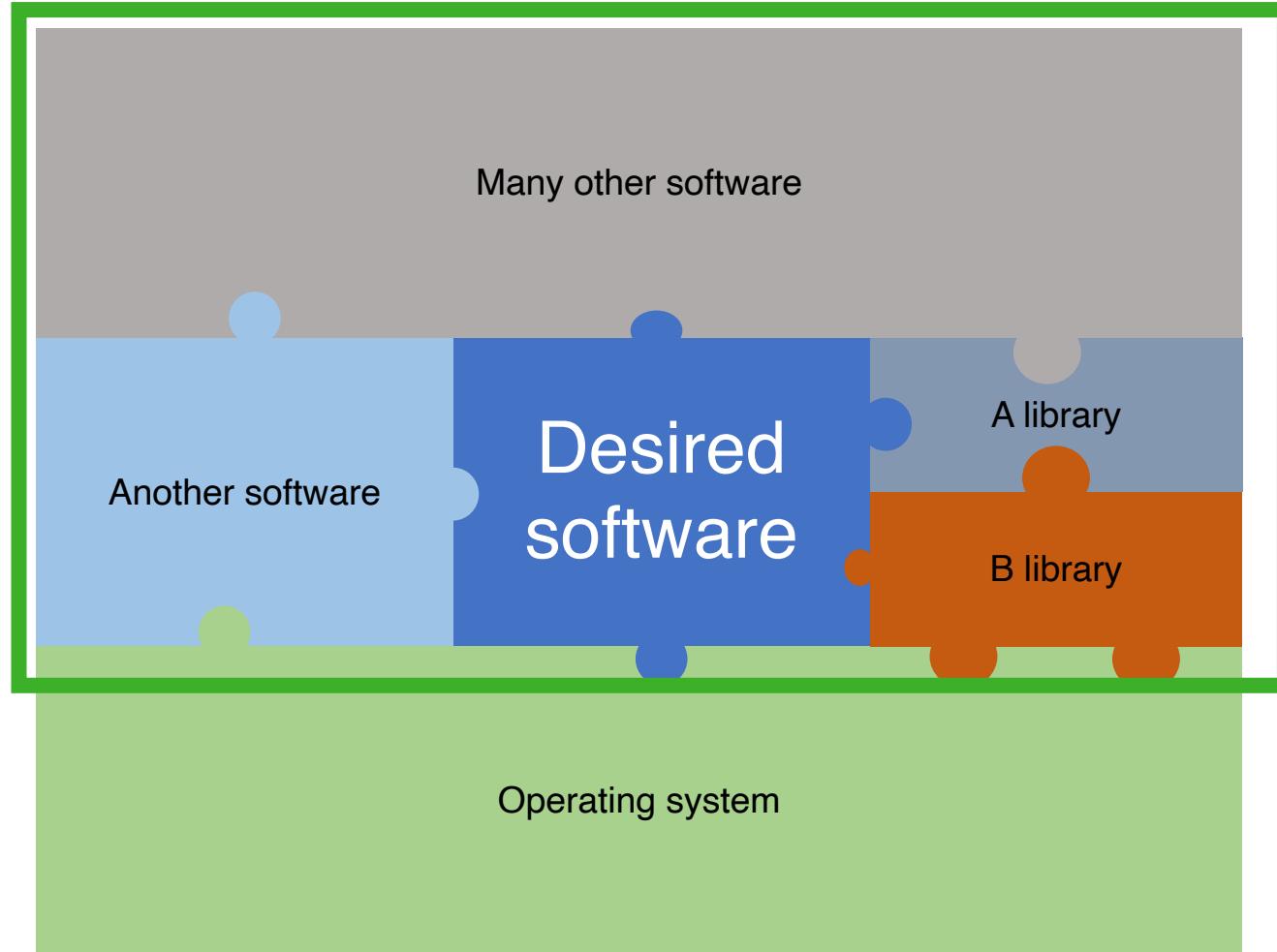
# Software environments with `conda` miniforge

Software environment



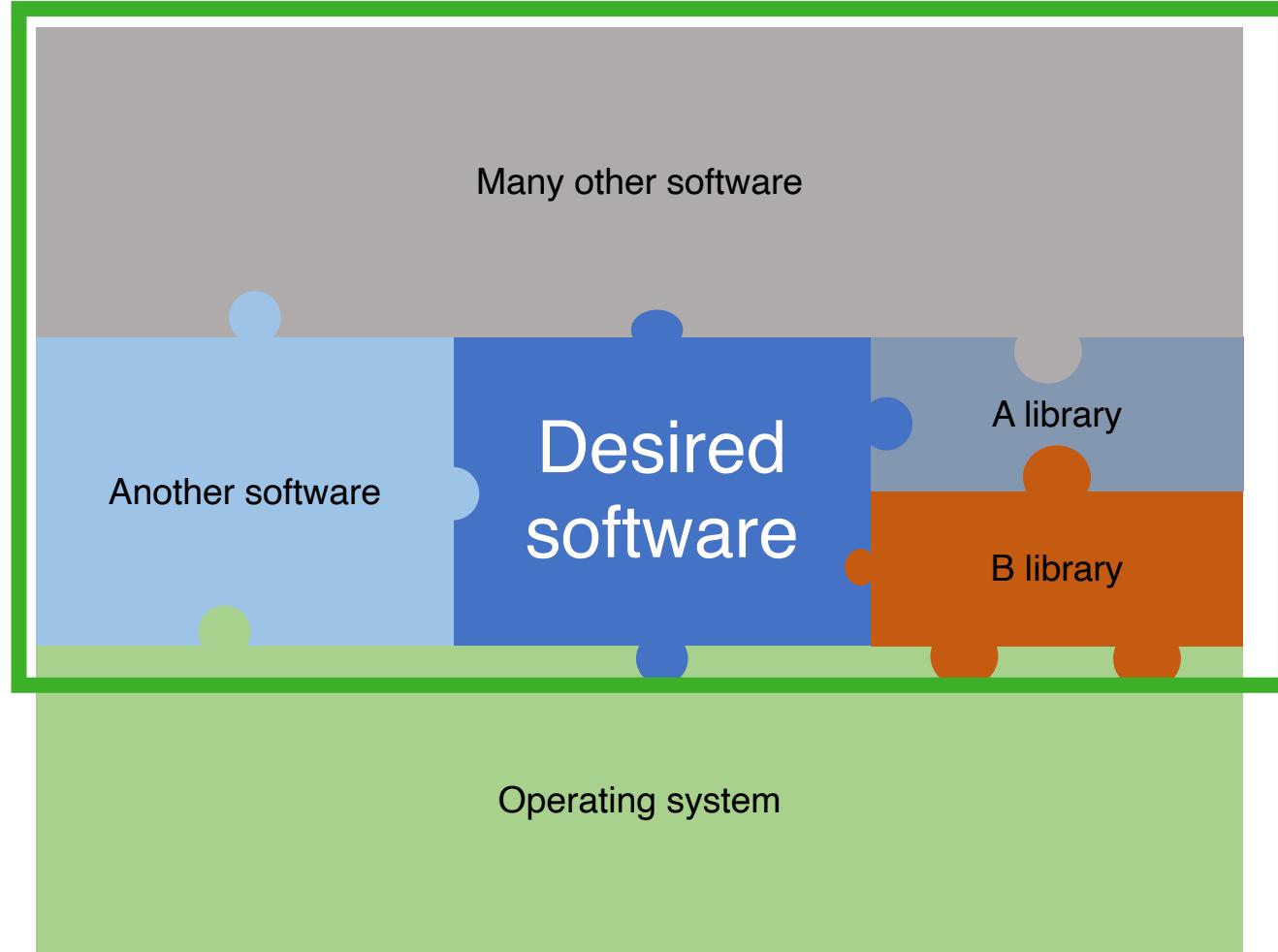
# Software environments with `conda miniforge`

Software environment



# Software environments with `conda` miniforge

Software environment

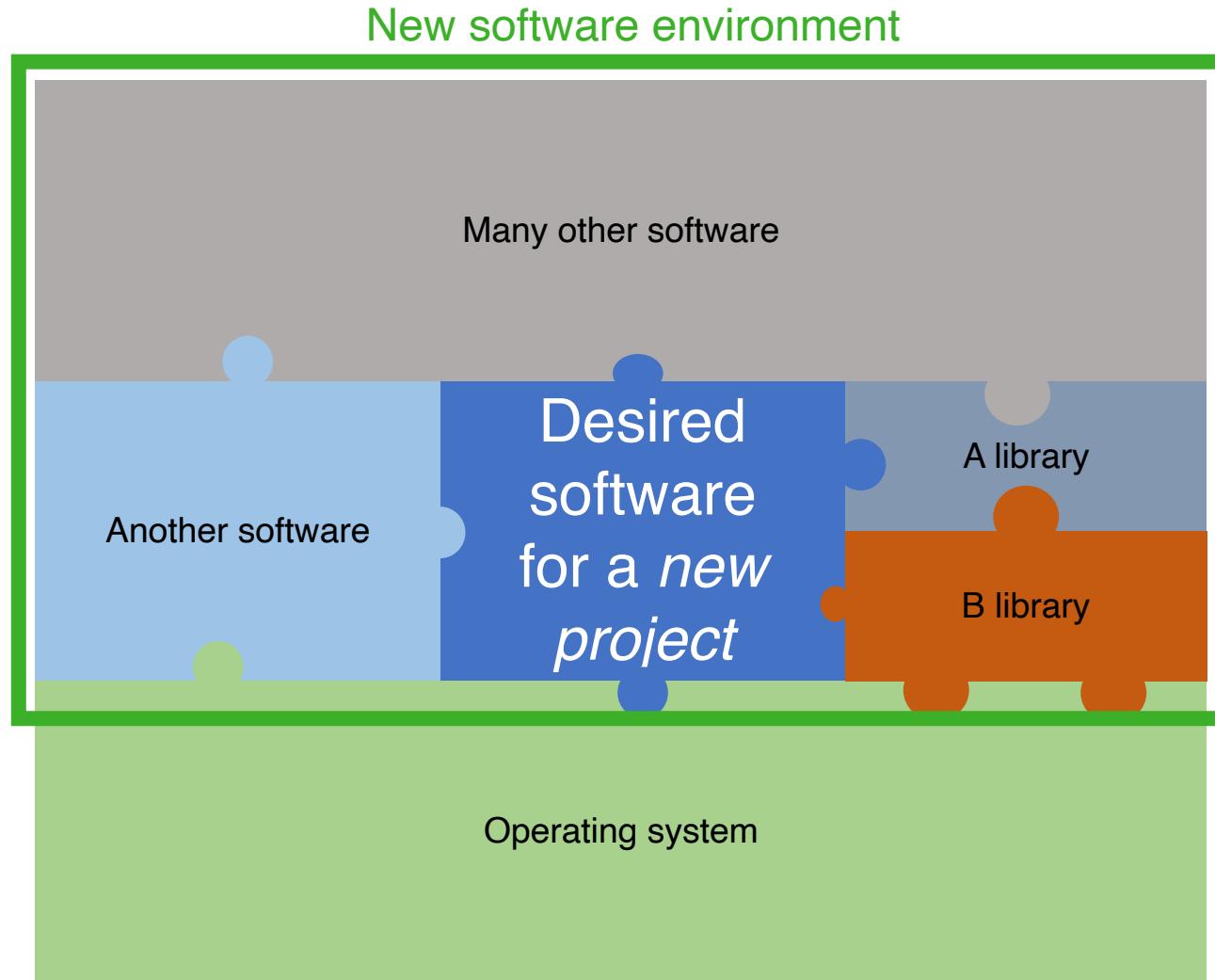
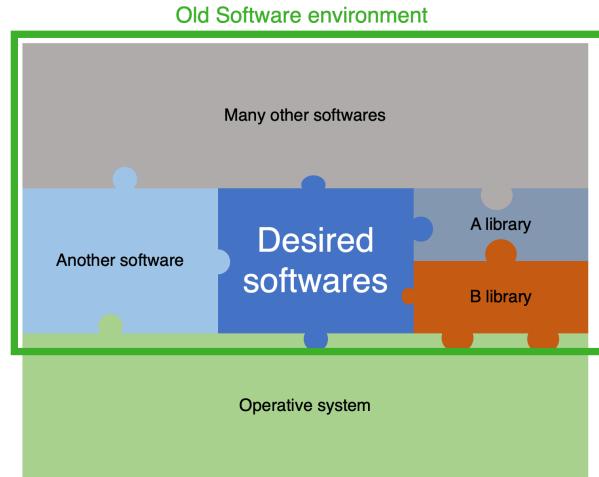


Another  
desired  
software for a  
*new project*

High chance of error!

All ecosystems will be potentially updated

# Software environments with `conda miniforge`



# Hands-on: environments miniforge3

terminal

```
cpetrini@p-cpetrini ~ %
cpetrini@p-cpetrini ~ % echo "This is my prompt"
This is my prompt
(base) cpetrini@p-cpetrini ~ %
(base) cpetrini@p-cpetrini ~ % echo "This is my prompt after installing miniforge"
This is my prompt after installing miniforge
```

# Hands-on: environments miniforge3



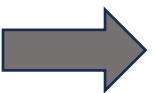
The terminal window shows the following session:

```
cpetrini@p-cpetrini ~ %
cpetrini@p-cpetrini ~ % echo "This is my prompt"
This is my prompt
(base) cpetrini@p-cpetrini ~ %
(base) cpetrini@p-cpetrini ~ % echo "This is my prompt after installing miniforge"
This is my prompt after installing miniforge
```

A red arrow points from the text "Give information about the environment in use" to the word "(base)" in the second-to-last line of the terminal output.

Give information about the environment in use

# Hands-on: create a new environment



```
(base) cpetrini@p-cpetrini ~ % conda create -n a_new_env_for_projX
Channels:
- conda-forge
Platform: osx-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /Users/cpetrini/miniforge3/envs/a_new_env_for_projX

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate a_new_env_for_projX
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

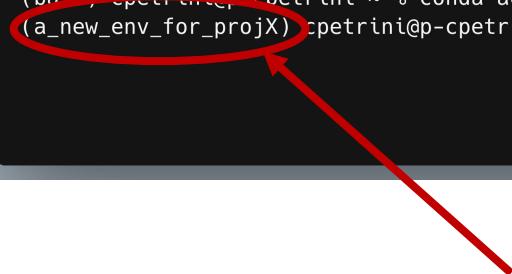
# Hands-on: create a new environment



A screenshot of a macOS terminal window titled "terminal". The window has three colored window control buttons (red, yellow, green) at the top left. The terminal text area shows two commands entered by the user:

```
(base) cpetrini@p-cpetrini ~ % conda create -n a_new_env_for_projX  
(base) cpetrini@p-cpetrini ~ % conda activate a_new_env_for_projX
```

# Hands-on: create a new environment



```
terminal  
  
(base) cpetrini@p-cpetrini ~ % conda create -n a_new_env_for_projX  
(base) cpetrini@p-cpetrini ~ % conda activate a_new_env_for_projX  
(a_new_env_for_projX) cpetrini@p-cpetrini ~ %
```

You are not in “base” anymore

# Hands-on: create a new environment

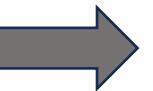


A screenshot of a macOS terminal window titled "terminal". The window shows the following command history:

```
(base) cpetrini@p-cpetrini ~ % conda create -n a_new_env_for_projX
(base) cpetrini@p-cpetrini ~ % conda activate a_new_env_for_projX
(a_new_env_for_projX) cpetrini@p-cpetrini ~ %
(a_new_env_for_projX) cpetrini@p-cpetrini ~ % conda install bioconda::samtools
```

# Hands-on: create a new environment

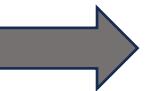
```
terminal  
  
(base) cpetrini@p-cpetrini ~ % conda create -n a_new_env_for_projX  
(base) cpetrini@p-cpetrini ~ % conda activate a_new_env_for_projX  
(a_new_env_for_projX) cpetrini@p-cpetrini ~ %  
(a_new_env_for_projX) cpetrini@p-cpetrini ~ % conda install bioconda::samtools
```



```
(a_new_env_for_projX) cpetrini@p-cpetrini ~ % conda install bioconda::samtools  
  
Channels:  
- conda-forge  
- bioconda  
Platform: osx-64  
Collecting package metadata (repodata.json): done  
Solving environment: done  
  
## Package Plan ##  
  
environment location: /Users/cpetrini/miniforge3/envs/a_new_env_for_projX  
  
added / updated specs:  
- bioconda::samtools  
  
The following packages will be downloaded:  
  
package | build  
---|---  
htslib-1.21 | hec81eee_0 2.8 MB bioconda  
libdeflate-1.21 | hfdf4475_0 69 KB conda-forge  
samtools-1.21 | h94387ee_0 457 KB bioconda  
  
Total: 3.3 MB  
  
The following NEW packages will be INSTALLED:
```

# Hands-on: create a new environment

```
terminal  
  
(base) cpetrini@p-cpetrini ~ % conda create -n a_new_env_for_projX  
(base) cpetrini@p-cpetrini ~ % conda activate a_new_env_for_projX  
(a_new_env_for_projX) cpetrini@p-cpetrini ~ %  
(a_new_env_for_projX) cpetrini@p-cpetrini ~ % conda install bioconda::samtools
```



```
(a_new_env_for_projX) cpetrini@p-cpetrini ~ % conda install bioconda::samtools  
  
Channels:  
- conda-forge  
- bioconda  
Platform: osx-64  
Collecting package metadata (repodata.json): done  
Solving environment: done  
  
## Package Plan ##  
  
environment location: /Users/cpetrini/miniforge3/envs/a_new_env_for_projX  
  
added / updated specs:  
- bioconda::samtools  
  
The following packages will be downloaded:  
  
package | build  
---|---  
htslib-1.21 | hec81eee_0 2.8 MB bioconda  
libdeflate-1.21 | hfdf4475_0 69 KB conda-forge  
samtools-1.21 | h94387ee_0 457 KB bioconda  
---|---  
Total: 3.3 MB  
  
The following NEW packages will be INSTALLED:
```



```
(a_new_env_for_projX) cpetrini@p-cpetrini ~ % samtools  
  
Program: samtools (Tools for alignments in the SAM format)  
Version: 1.21 (using htslib 1.21)  
  
Usage: samtools <command> [options]  
  
Commands:  
-- Indexing  
dict create a sequence dictionary file  
faidx index/extract FASTA  
fqidx index/extract FASTQ  
index index alignment
```

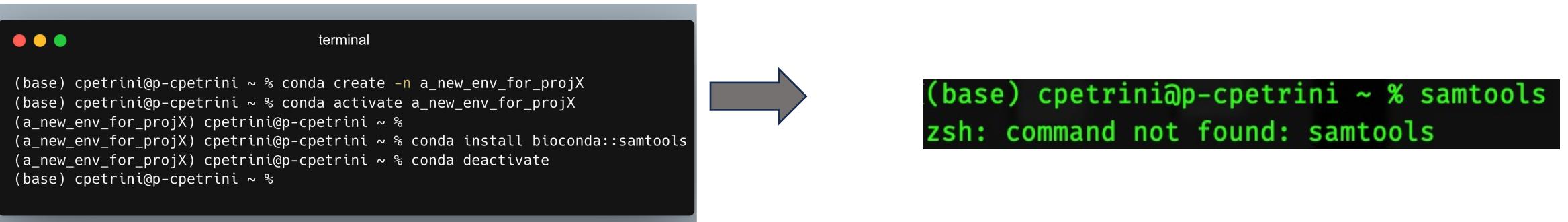
# Hands-on: create a new environment



terminal

```
(base) cpetrini@p-cpetrini ~ % conda create -n a_new_env_for_projX
(base) cpetrini@p-cpetrini ~ % conda activate a_new_env_for_projX
(a_new_env_for_projX) cpetrini@p-cpetrini ~ %
(a_new_env_for_projX) cpetrini@p-cpetrini ~ % conda install bioconda::samtools
(a_new_env_for_projX) cpetrini@p-cpetrini ~ % conda deactivate
(base) cpetrini@p-cpetrini ~ %
```

# Hands-on: create a new environment



```
terminal  
  
(base) cpetrini@p-cpetrini ~ % conda create -n a_new_env_for_projX  
(base) cpetrini@p-cpetrini ~ % conda activate a_new_env_for_projX  
(a_new_env_for_projX) cpetrini@p-cpetrini ~ %  
(a_new_env_for_projX) cpetrini@p-cpetrini ~ % conda install bioconda::samtools  
(a_new_env_for_projX) cpetrini@p-cpetrini ~ % conda deactivate  
(base) cpetrini@p-cpetrini ~ %  
  
(base) cpetrini@p-cpetrini ~ % samtools  
zsh: command not found: samtools
```

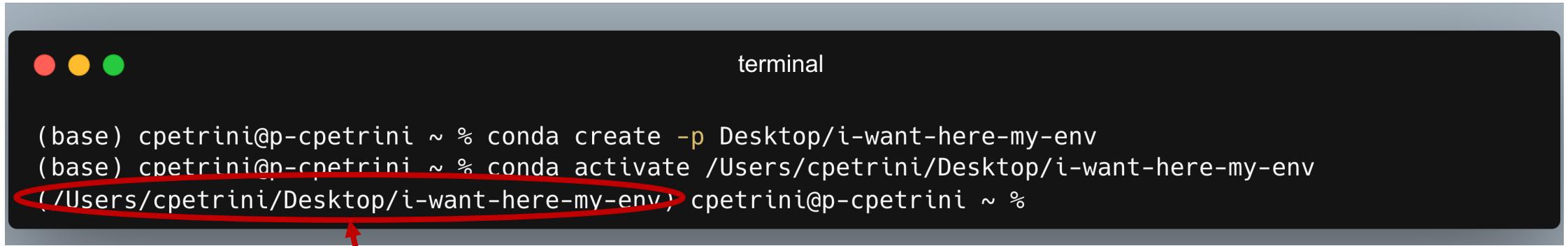
# Hands-on: create a new environment on a specific path



terminal

```
(base) cpetrini@p-cpetrini ~ % conda create -p Desktop/i-want-here-my-env
(base) cpetrini@p-cpetrini ~ % conda activate /Users/cpetrini/Desktop/i-want-here-my-env
(/Users/cpetrini/Desktop/i-want-here-my-env) cpetrini@p-cpetrini ~ %
```

# Hands-on: create a new environment on a specific path



The screenshot shows a terminal window with the following command history:

```
(base) cpetrini@p-cpetrini ~ % conda create -p Desktop/i-want-here-my-env
(base) cpetrini@p-cpetrini ~ % conda activate /Users/cpetrini/Desktop/i-want-here-my-env
(/Users/cpetrini/Desktop/i-want-here-my-env) cpetrini@p-cpetrini ~ %
```

A red oval highlights the path `/Users/cpetrini/Desktop/i-want-here-my-env`, and a red arrow points from this highlighted text down to the explanatory text below.

Now all the software installed for this environment will be located here

# Hands-on: create a new environment on a specific path

Environments in a specific path is the key for reproducibility,  
stay tuned!



```
terminal  
  
(base) cpetrini@p-cpetrini ~ % conda create -p Desktop/i-want-here-my-env  
(base) cpetrini@p-cpetrini ~ % conda activate /Users/cpetrini/Desktop/i-want-here-my-env  
(/Users/cpetrini/Desktop/i-want-here-my-env) cpetrini@p-cpetrini ~ %
```

Now all the software installed for this environment will be located here

# Back to reproducibility

Your research is reproducible if  
- and only if - you can  
repeatedly and easily  
reproduce it

(and you are another person in 6 months time)

# The case of brain's microbiome

nature > articles > article

Article | Published: 11 March 2020

**RETRACTED ARTICLE: Microbiome analyses of blood and tissues suggest cancer diagnostic approach**

Gregory D. Poore, Evgenia Kopylova, Qiyun Zhu, Carolina Carpenter, Serena Fraraccio, Stephen Wandro, Tomasz Kosciolek, Stefan Janssen, Jessica Metcalf, Se Jin Song, Jad Kanbar, Sandrine Miller-Montgomery, Robert Heaton, Rana Mckay, Sandip Pravin Patel, Austin D. Swafford & Rob Knight 

*Nature* 579, 567–574 (2020) | [Cite this article](#)

107k Accesses | 660 Citations | 977 Altmetric | [Metrics](#)

 This article was [retracted](#) on 26 June 2024

 This article has been [updated](#)

A **controversial** paper that generated much buzz in the field notwithstanding the high-impact journal.

Was later retracted **by the editors** (not by the authors) after the buzz substantiated into data **irreproducibility**

> 900 citations

# Retraction came after...

- Steven Salzberg, one of the most influential bioinformatics scientists took on a multi year battle
- Salzberg and team spent massive computational resources/time to re-analyse the whole data.



# The biggest problem with reproducibility

Since there is a **stigma** that comes with not being reproducible

=>

- Hi pressure in not **admitting** errors
- Huge effort to **prove** something is wrong

# The Reproducibility Crisis in Science

- **Reproducibility:** The ability to replicate the results of a study using the same methods, data, and analysis.
- **Challenges in traditional research:** Issues like incomplete documentation, inaccessible datasets, and variability in methods.
- **Impact on scientific progress:** Loss of trust, slower advancement, and difficulties in building on previous research

# Lesson learnt

- Exaggerated claims are the new norm!
- We have **a credibility problem in science**: there is NO COST to being wrong, so people are wrong more often than ever
- As a scientist you should be extraordinarily careful
- Don't build a career on nonsense
- Redo the analysis in papers you want to build upon

The visual  
aggregator

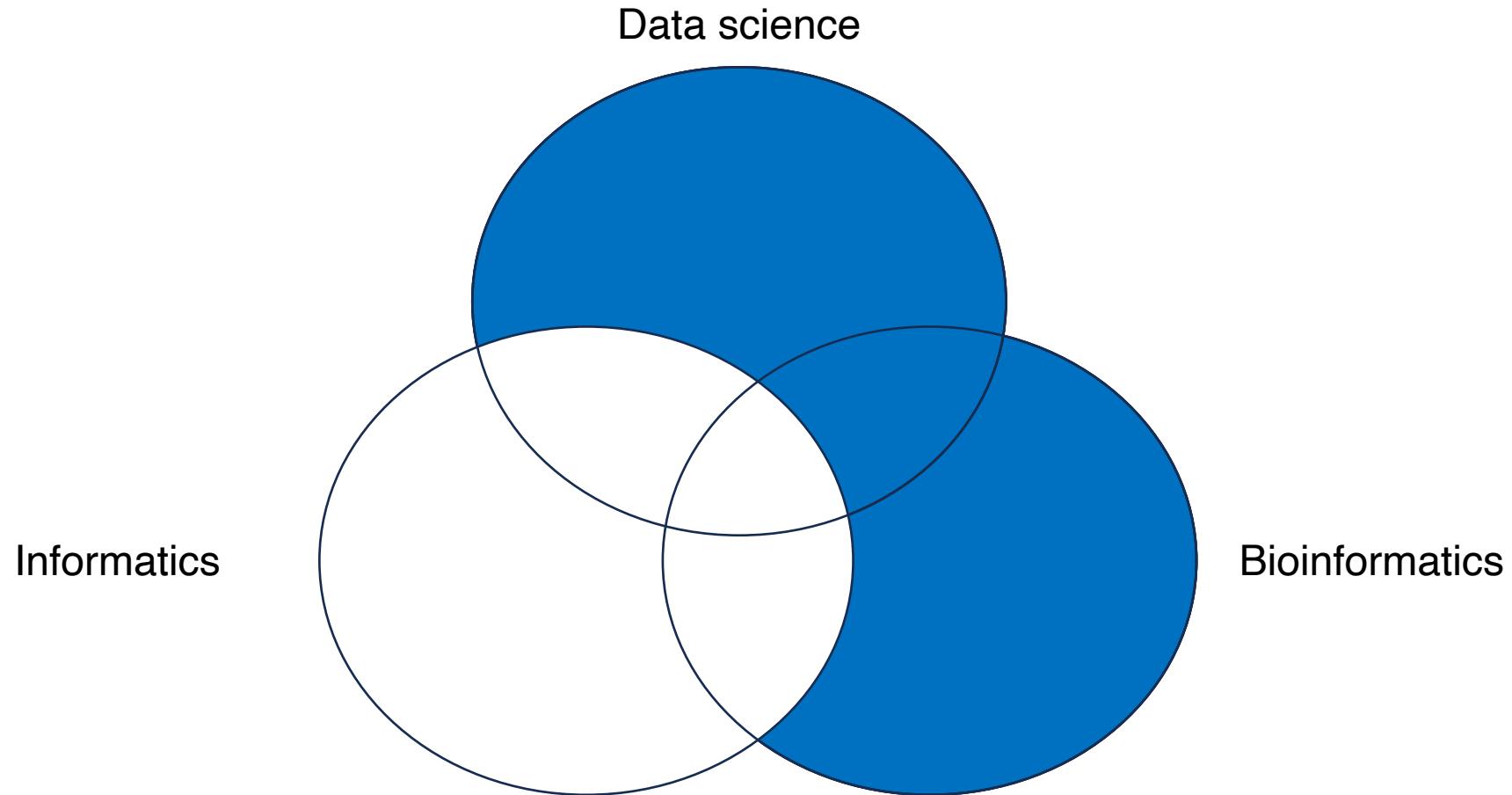
# Notebooks

Is seeing believing? Go interactive with Jupyter Notebooks

# Work reports and data visualization

BBCC2024, RCDS

# Topic domain: bioinformatics



# The problem: keep constantly track of what you are doing!

Bioinformaticians are required to **be conscious** of the work they perform and performed in the past

*That means:*

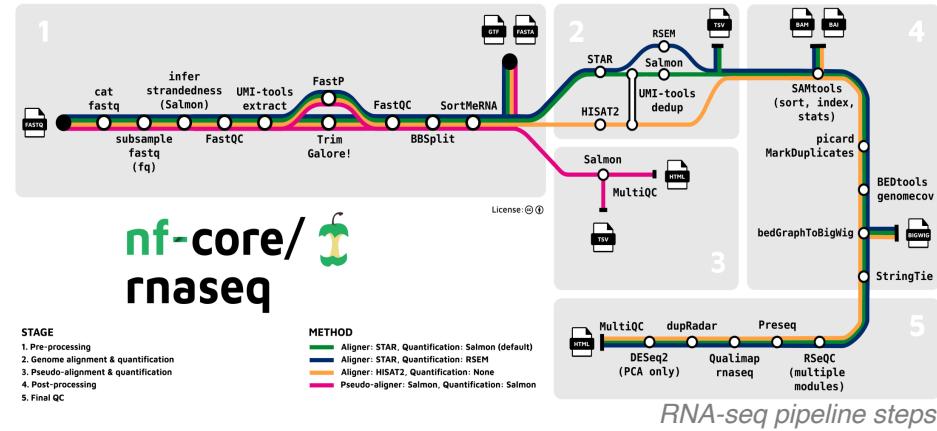
- Create detailed and easily accessible **reports** of analyses
- **Be aware** of the data processed and managed

# Why creating a report?

**The analyses are usually based on:**

- many parameters
- many steps

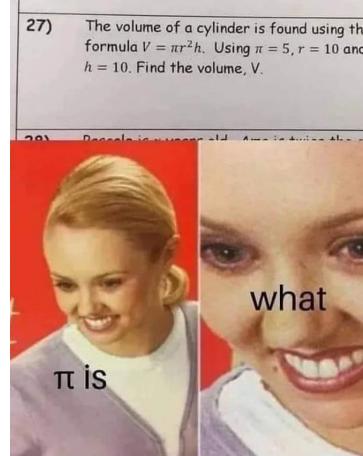
remembering everything after days/months/years could be completely unreliable.



**At the same time:**

- PI
- colleague
- collaborator
- reviewer
- a random interested person

**could ask you about it** and you should be able to answer correctly and quickly



Your PI after you attempt to guess a parameter you forgot

```

{
  "version": "3.12.0",
  "name": "rnaseq",
  "description": "A standard nf-core RNA-seq pipeline for processing transcriptome sequencing data. It includes pre-processing, alignment, quantification, and post-processing steps using various tools like STAR, HISAT2, RSEM, Salmon, and StringTie. The pipeline also includes quality control (MultiQC) and differential expression analysis (DESeq2).",
  "url": "https://nf-co.re/rnaseq/3.12.0",
  "git_sha": "3.12.0",
  "date": "2023-09-26T14:00:00Z",
  "contributors": [
    {
      "name": "nf-core/rnaseq"
    }
  ],
  "dependencies": {
    "nf-core/rnaseq": "3.12.0",
    "nf-core/tools": "3.12.0",
    "nf-core/test-datasets": "3.12.0",
    "nf-core/shared": "3.12.0"
  },
  "parameters": {
    "fastq": "FastQ",
    "bam": "BAM"
  }
}
  
```

A portion of JSON describing RNA-seq required parameters

# Why be aware of data managed?

*An example:*

To: you  
Cc:  
Subject: Project decision  
Message Size: 15 KB

Hi my dear student,

we just got this dataset of *Saccharomyces* *Mile* growth rate from a group of wet-lab researchers.

They measure the yeast growth rate at a certain time point. Now, they ask if there is any correlation between time and growth rate.

Are these data meaningful according to your analysis?  
We may need an answer to decide whether to collaborate or not with them.

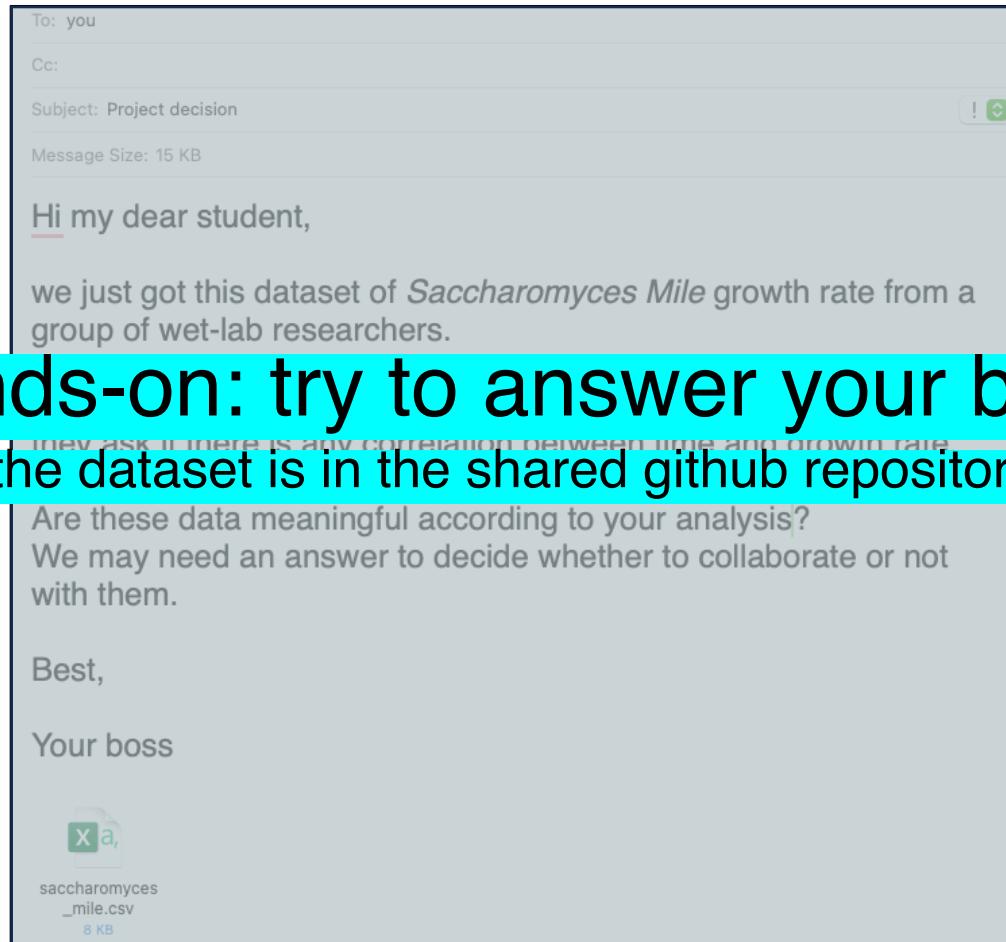
Best,

Your boss

 saccharomyces\_mile.csv  
8 KB

# Why be aware of data managed?

*An example:*

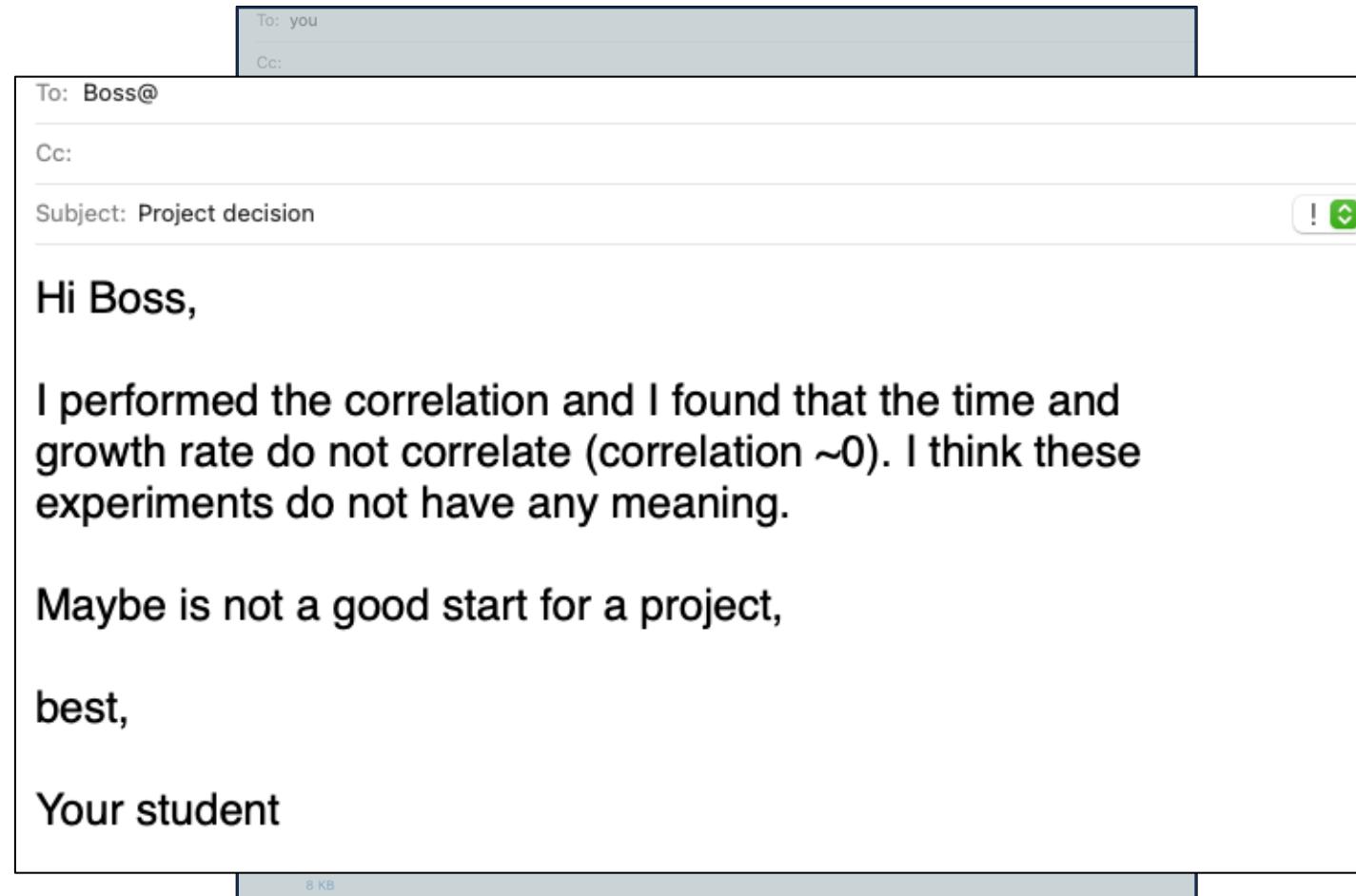


**Hands-on: try to answer your boss!**  
(the dataset is in the shared github repository)

*R correlation function: cor();  
python correlation function  
corrcoef() from Numpy library*

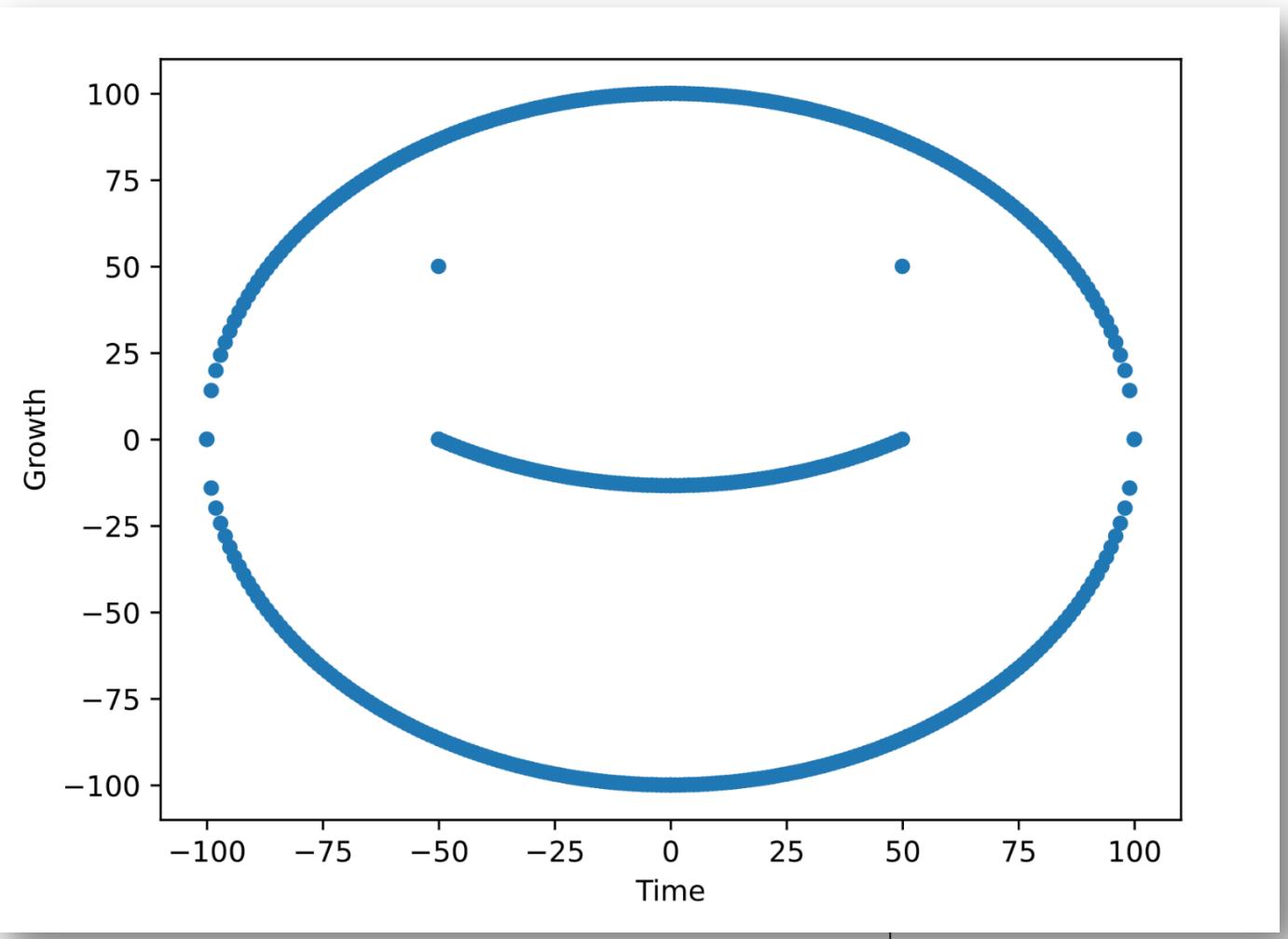
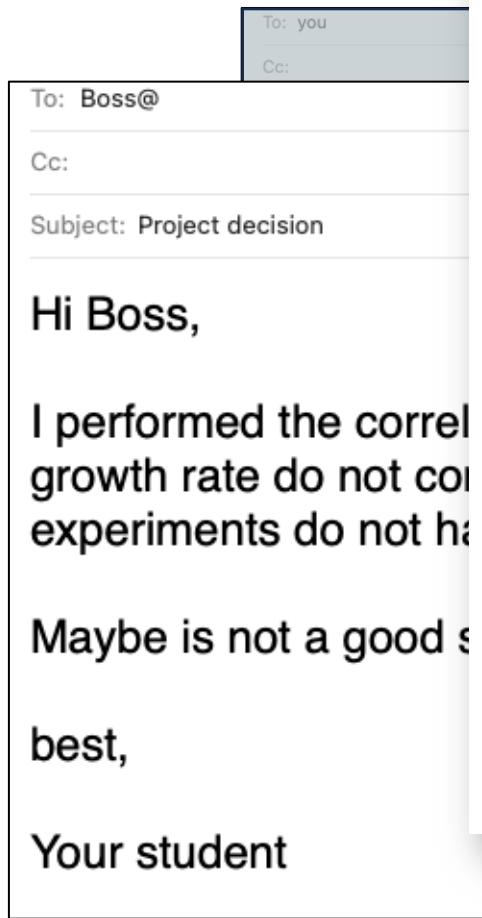
# Why be aware of data managed?

*An example:*

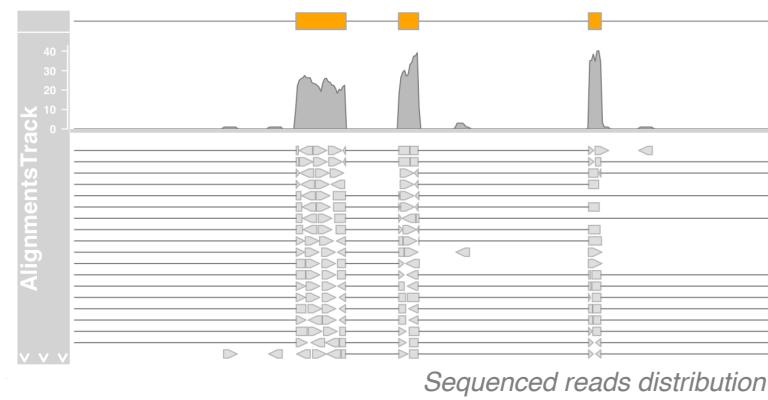
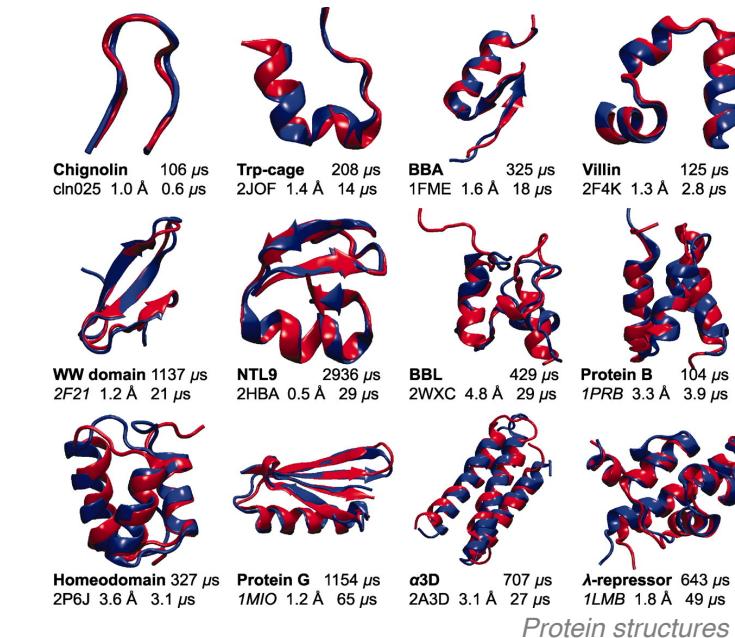
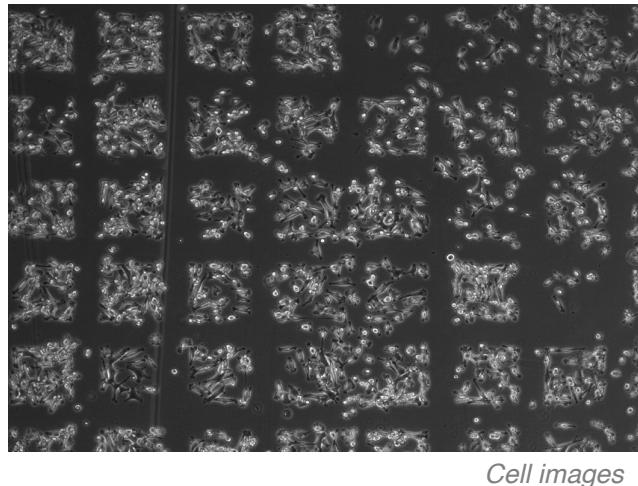
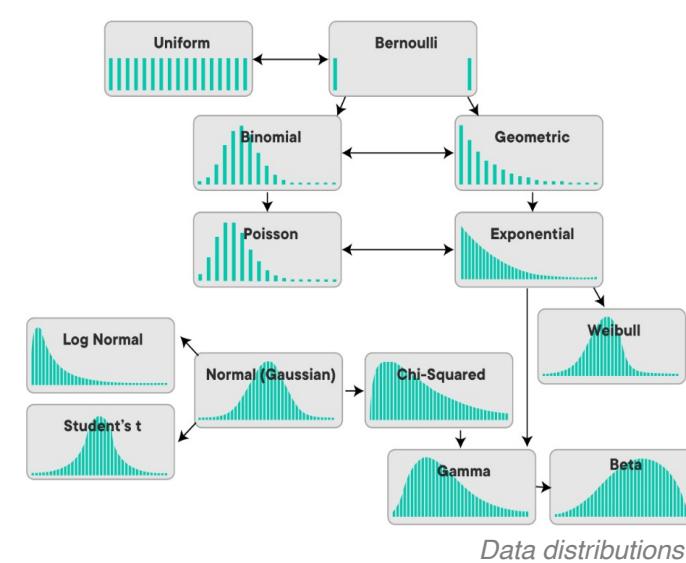


# Why be aware of data managed? Did you plot the data?

*An example:*



# Why be aware of data managed? Other examples



# Introducing JupyterLab & notebooks



<https://jupyter.org/>

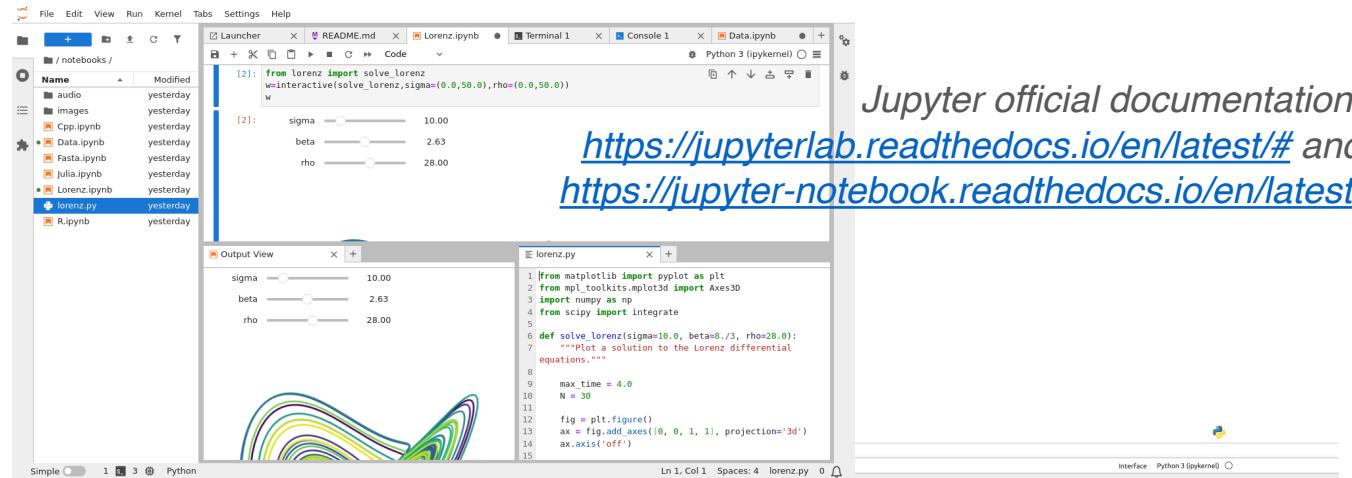
Free software, open standards, and web services for **interactive** computing  
**across all programming languages**

# The problems solver: JupyterLab

## JupyterLab is an IDE that combines

- coding,
- code processing,
- work reports
- data visualization

in a single file (called a Jupyter Notebook).



Jupyter official documentation:  
<https://jupyterlab.readthedocs.io/en/latest/#> and  
<https://jupyter-notebook.readthedocs.io/en/latest/>

Example of JupyterLab interface  
Running Code

First and foremost, the Jupyter Notebook is an interactive environment for writing and running code. The notebook is capable of running code in a wide range of languages. However, each notebook is associated with a single kernel. This notebook is associated with the Python kernel, therefore runs Python code.

Code cells allow you to enter and run code

Run a code cell using Shift+Enter or pressing the  button in the toolbar above:

```
[1]: a = 10
[2]: print(a)
10
```

There are two other keyboard shortcuts for running code:

- Alt+Enter: runs the current cell and inserts a new one below.
- Ctrl+Enter: run the current cell and enters command mode.

Managing the Kernel

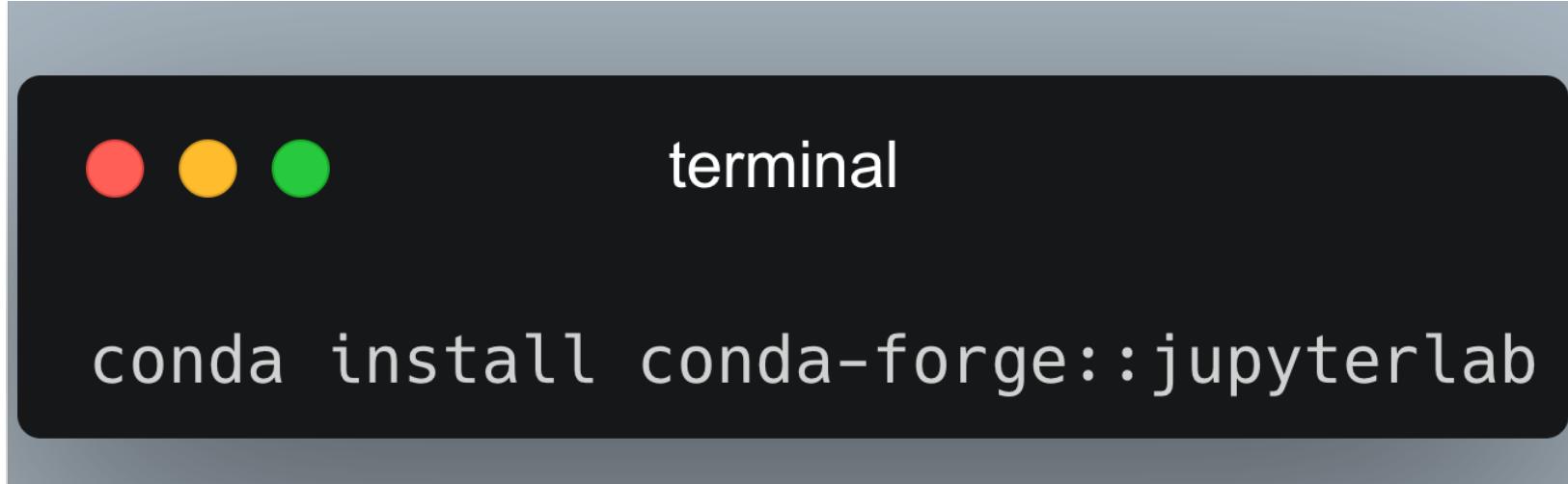
Code is run in a separate process called the Kernel. The Kernel can be interrupted or restarted. Try running the following cell and then hit the  button in the toolbar above.

```
[3]: import time
time.sleep(10)
```

If the Kernel dies you will be prompted to restart it. Here we call the low-level system `libc.time` routine with the wrong argument via ctypes to segfault the Python interpreter.

Example of Jupyter Notebook

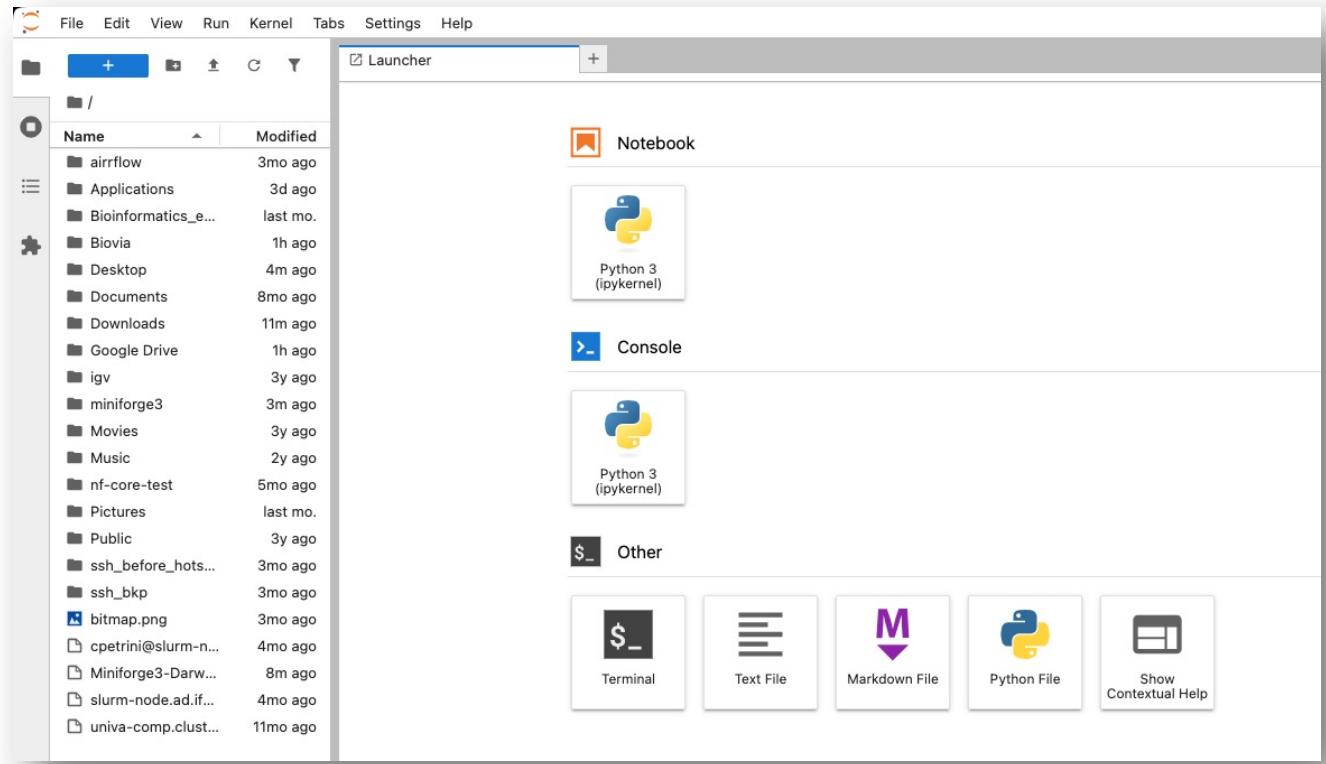
# Hands-on: install JupyterLab



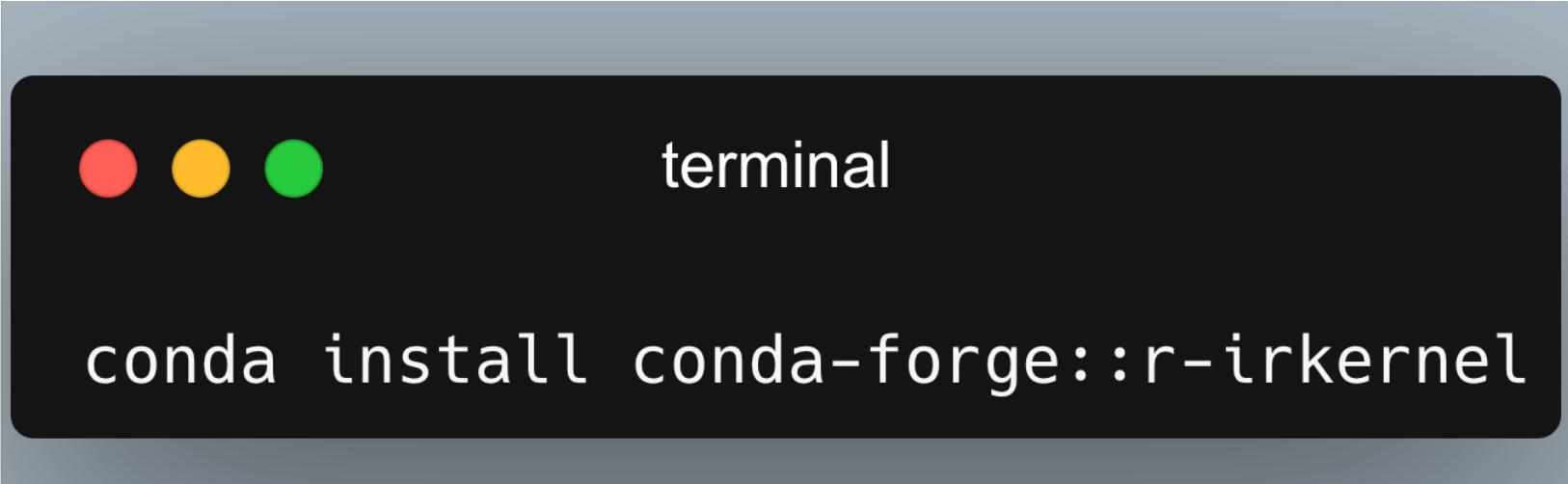
# Hands-on: run JupyterLab



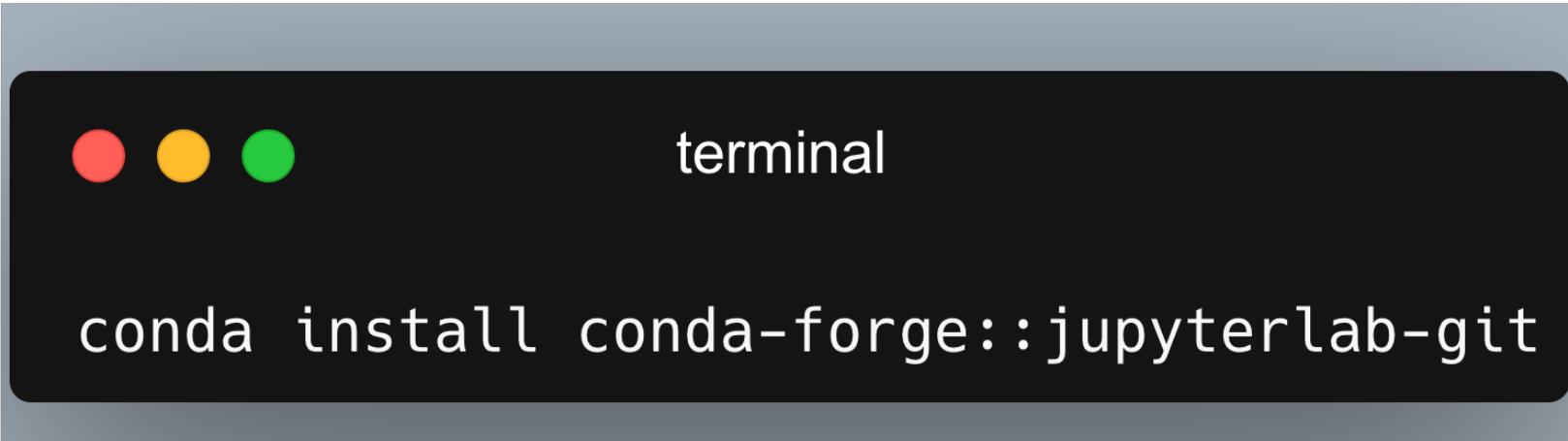
*It automatically  
opens the  
browser\**



# Hands-on: install R kernel



# Hands-on: install jupyter git extension



# Hands-on: install jupyter with R and Jupyter lab

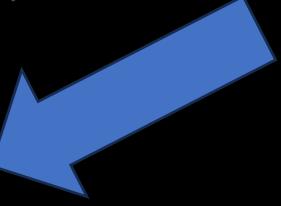
```
conda create --name tidyR
conda activate tidyR
conda install conda-forge::r-dplyr
conda install conda-forge::r-readr
conda install conda-forge::r-ggplot2
conda install conda-forge::r-irkernel
conda install conda-forge::jupyterlab
```

Launch JupyterLab after activating the conda env

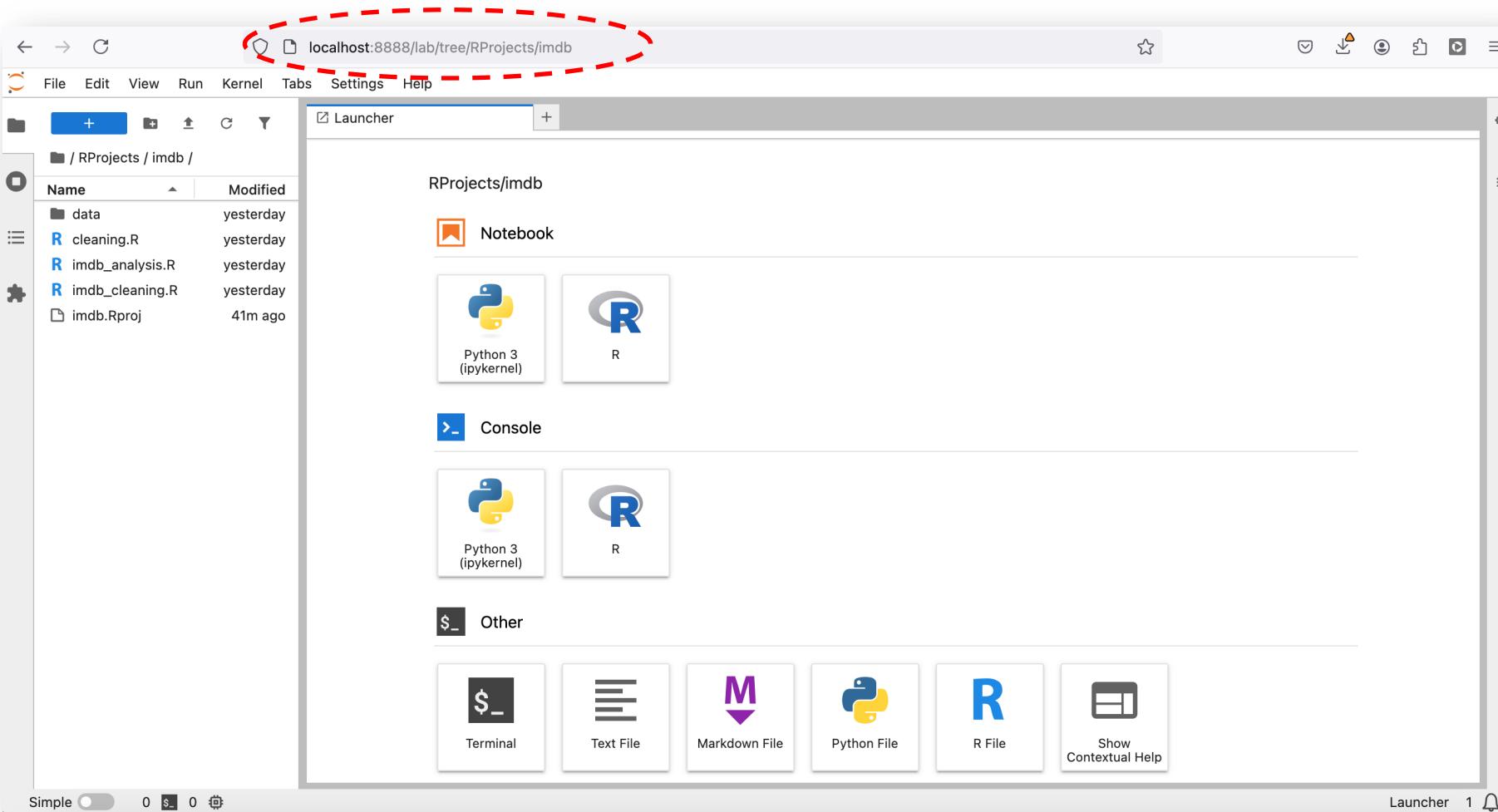
```
conda activate tidyR
jupyter lab
```

# Hands-on: launch the jupyter lab environment

```
python3.13
json5-0.9.28          | 28 KB    | #####| 100%
pure_eval-0.2.3        | 16 KB    | #####| 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(tidyR) rrossi@ ~ % jupyter lab
[I 2024-11-22 18:36:42.862 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2024-11-22 18:36:42.866 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2024-11-22 18:36:42.870 ServerApp] jupyterlab | extension was successfully linked.
[I 2024-11-22 18:36:42.873 ServerApp] Writing Jupyter server cookie secret to /Users/rrossi/Library/Jupyter/run
```



# Hands-on: shift to the browser and start working



# Exercise: analyze the IMDB data



Developer

## IMDb Non-Commercial Datasets

<https://developer.imdb.com/non-commercial-datasets/>

### IMDb Data Files Available for Download

Documentation for these data files can be found on [IMDb](#).

The data files available on this page are provided for non-commercial use only. These files have not been modified and there has been no change in location or schema, but if you encounter any issues please let us know.

[name.basics.tsv.gz](#)

[title.akas.tsv.gz](#)

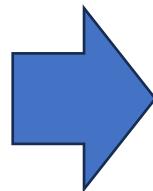
[title.basics.tsv.gz](#)

[title.crew.tsv.gz](#)

[title.episode.tsv.gz](#)

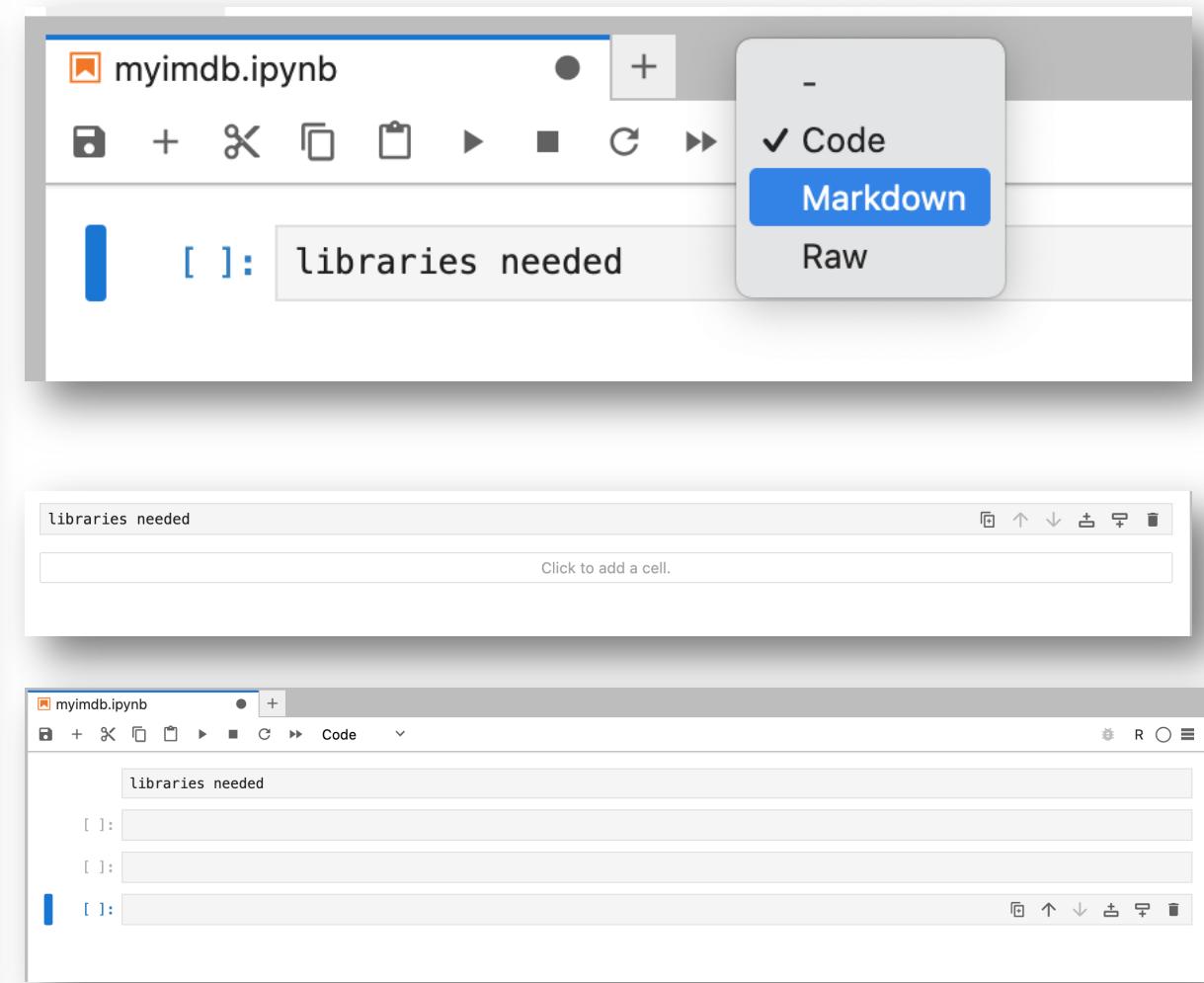
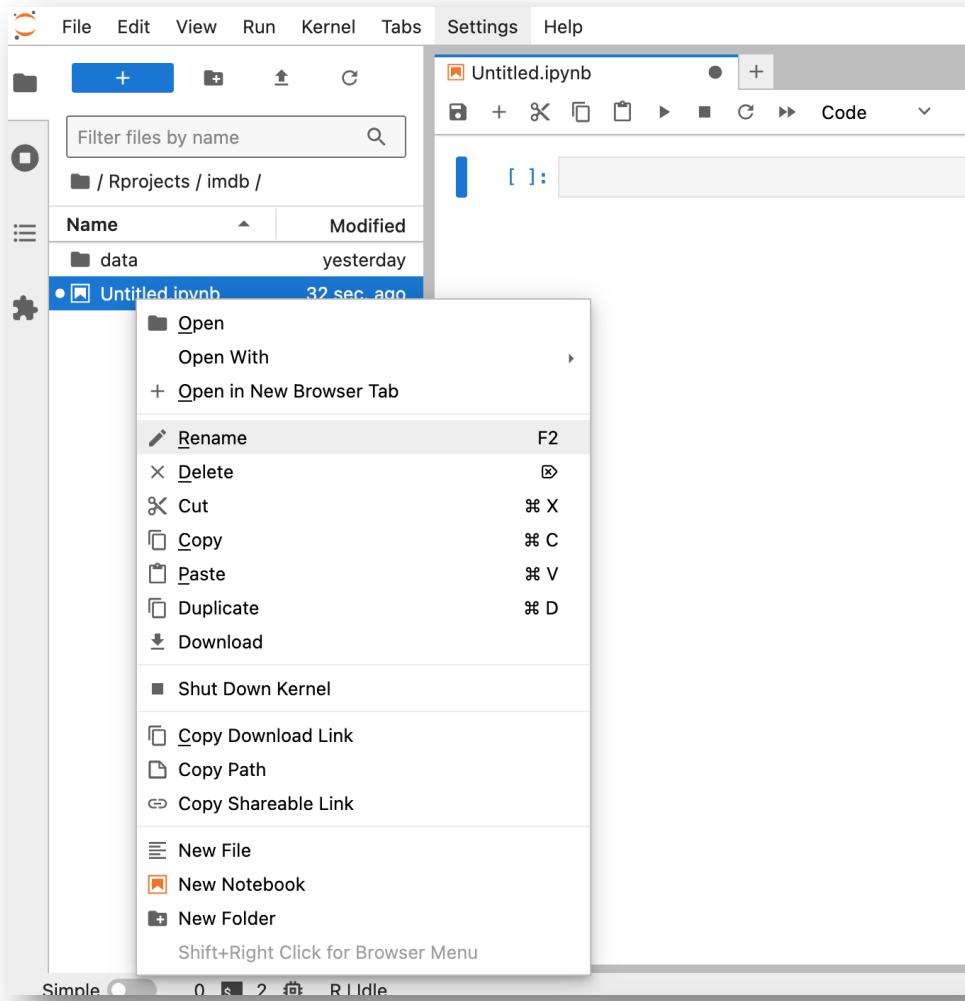
[title.principals.tsv.gz](#)

[title.ratings.tsv.gz](#)

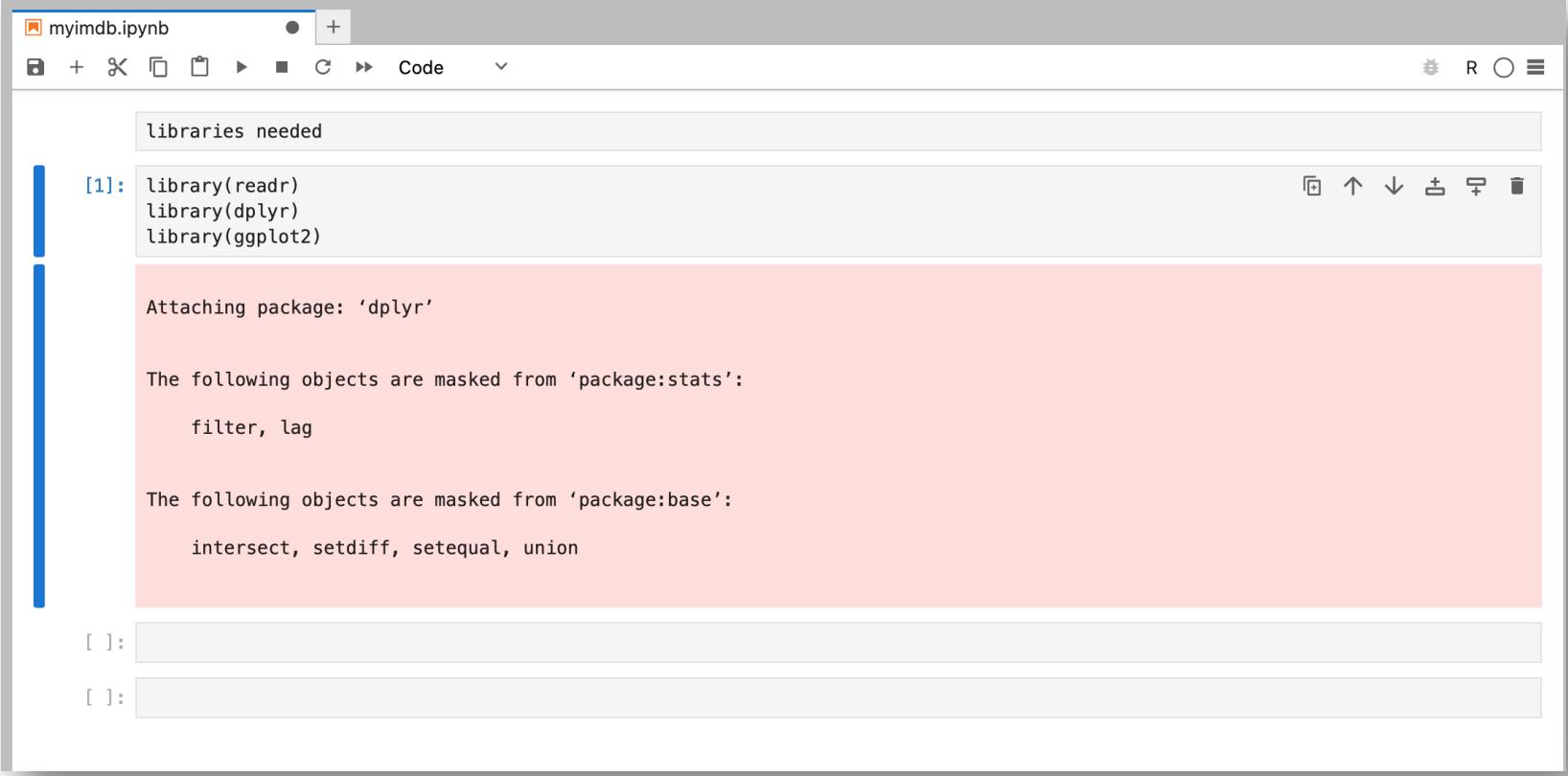


You will use the simplified  
data on the workshop  
GitHub repo instead

# Hands-on: opening and renaming, code/markdown chunks



# Hands-on: load libraries



The screenshot shows a Jupyter Notebook interface with the title "myimdb.ipynb". In the code cell [1], the user has entered:

```
library(readr)  
library(dplyr)  
library(ggplot2)
```

The output area displays the results of the library loading:

```
Attaching package: 'dplyr'  
  
The following objects are masked from 'package:stats':  
  filter, lag  
  
The following objects are masked from 'package:base':  
  intersect, setdiff, setequal, union
```

# Hands-on: read-in data files and inspect them

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** myimdb.ipynb
- Toolbar:** Includes icons for file operations (New, Open, Save, etc.), cell type (Code), and other notebook settings.
- Text Cell:** "READING FILES  
please change reading path accordingly"
- Code Cell [2]:**

```
# read the main title table (reduced and cleaned)
title_basics <- read_csv("data/small_title_basics.csv")
head(title_basics)
```
- Output Cell:** Shows the structure of the `title\_basics` dataset:
  - Rows: 333760 Columns: 9
  - Column specification:
    - Delimiter: ","
    - chr (5): tconst, titleType, primaryTitle, originalTitle, genres
    - dbl (3): isAdult, startYear, runtimeMinutes
    - lgl (1): endYear
  - A note: "Use `spec()` to retrieve the full column specification for this data."
  - A note: "Specify the column types or set `show\_col\_types = FALSE` to quiet this message."
  - A tibble: 6 × 9

tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
<chr>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<lgl>	<dbl>	<chr>
tt0011801	movie	Tötet nicht mehr	Tötet nicht mehr	0	2019	NA	NA	Action,Crime
tt0035423	movie	Kate & Leopold	Kate & Leopold	0	2001	NA	118	Comedy,Fantasy,Romance
tt0062336	movie	The Tango of the Widower and Its Distorting Mirror	El tango del viudo y su espejo deformante	0	2020	NA	70	Drama
tt0067758	movie	Simón, contamos contigo	Simón, contamos contigo	0	2015	NA	81	Comedy,Drama
tt0069049	movie	The Other Side of the Wind	The Other Side of the Wind	0	2018	NA	122	Drama

# Hands-on: errors or warnings

The screenshot shows a Jupyter Notebook interface with three code cells:

- Cell 3:**

```
# read the ratings table
title_ratings <- read_tsv("data/title.ratings.tsv")
```

Rows: 1501791 Columns: 3  
— Column specification —  
Delimiter: "\t"  
chr (1): tconst  
dbl (2): averageRating, numVotes  
  
i Use `spec()` to retrieve the full column specification for this data.  
i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.
- Cell 4:**

```
head(title_ratings)
```

A tibble: 6 × 3  
tconst averageRating numVotes  
<chr> <dbl> <dbl>  
tt0000001 5.7 2102  
tt0000002 5.6 282  
tt0000003 6.5 2121  
tt0000004 5.4 182  
tt0000005 6.2 2852  
tt0000006 5.0 200
- Cell 5:**

```
# read the principals table (containing data about actors, reduced and cleaned version)
title_principals <- read_csv("data/small_title_principals.csv")
# head(title_principals)
```

Error: 'data/small\_title\_principals.csv' does not exist in current working directory ('/storage/home/rrossi/Rprojects/imdb').  
Traceback:  
1. read\_csv("data/small\_title\_principals.csv")  
2. vroom::vroom(file, delim = ",", col\_names = col\_names, col\_types = col\_types,  
· col\_select = {  
· {

# Hands-on: processing objects



The screenshot shows a Jupyter Notebook interface with the title bar 'myimdb.ipynb'. The notebook contains two code cells:

```
# PROCESSING OBJECTS

[ ]: # add ratings to title_basics (join the tables)
title_basics_ratings <- left_join(title_basics, title_ratings, by = "tconst")
head(title_basics_ratings)

[ ]: # keep only shorter movies with at least 10 votes
df_movies <- filter(title_basics_ratings, runtimeMinutes < 180, numVotes >= 10)
dim(df_movies)
head(df_movies)
```

The code uses the dplyr package for data manipulation, specifically joining 'title\_basics' and 'title\_ratings' tables on the 'tconst' column, and filtering movies with runtime less than 180 minutes and at least 10 votes.

# Hands-on: plotting

The screenshot shows a Jupyter Notebook interface with a tab titled "myimdb.ipynb". The notebook contains three code cells:

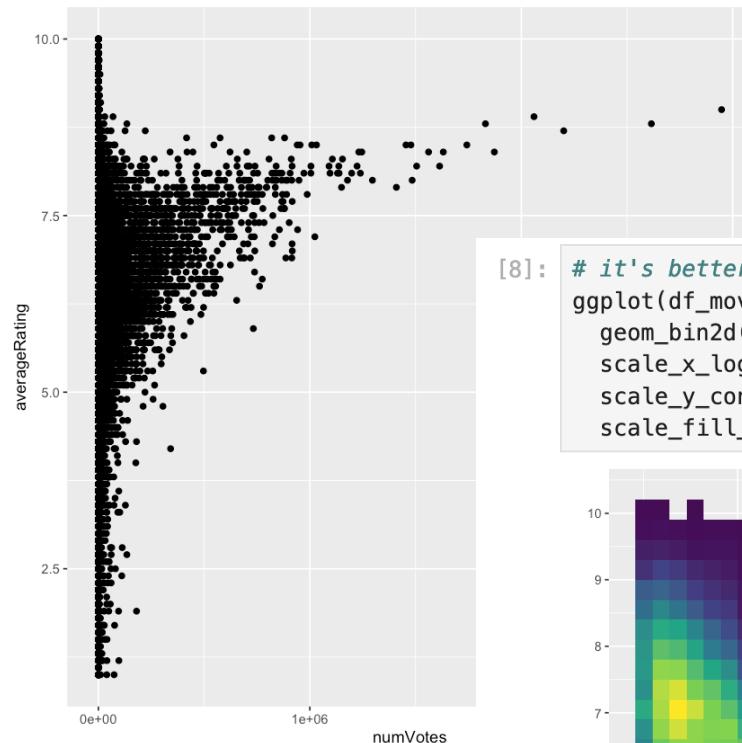
- # PLOTTING FILES**
- [ ]: 

```
# Let's see the ratings vs number of votes distribution
ggplot(df_movies, aes(x = numVotes, y = averageRating)) +
  geom_point()
```
- [ ]: 

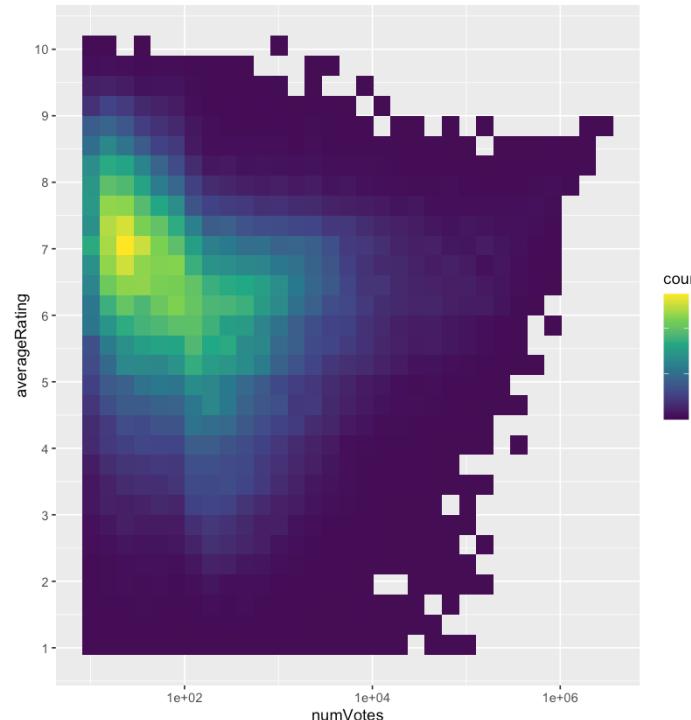
```
# it's better to look at it with a heatmap on log10 scale
ggplot(df_movies, aes(x = numVotes, y = averageRating)) +
  geom_bin2d() +
  scale_x_log10() +
  scale_y_continuous(breaks = 1:10) +
  scale_fill_viridis_c()
```
- [ ]: 

```
# what about the rating as a function of lenght of the movie?
ggplot(df_movies, aes(x = runtimeMinutes, y = averageRating)) +
  geom_bin2d() +
  scale_x_continuous(breaks = seq(0, 180, 60), labels = 0:3) +
  scale_y_continuous(breaks = 0:10) +
  scale_fill_viridis_c(option = "inferno") +
  theme_minimal() +
  labs(title = "Relationship between Movie Runtime and Average Movie Rating",
       subtitle = "Data from IMDb",
       x = "Runtime (Hours)",
       y = "Average User Rating",
       fill = "# Movies")
```

```
[7]: # Let's see the ratings vs number of votes distribution
ggplot(df_movies, aes(x = numVotes, y = averageRating)) +
  geom_point()
```

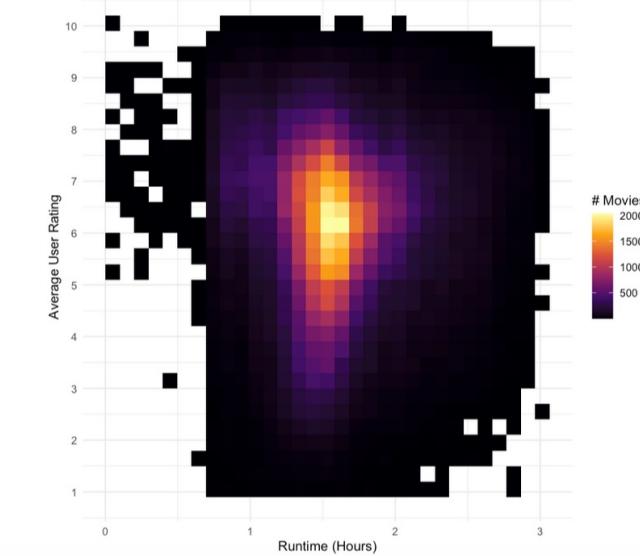


```
[8]: # it's better to look at it with a heatmap on log10 scale
ggplot(df_movies, aes(x = numVotes, y = averageRating)) +
  geom_bin2d() +
  scale_x_log10() +
  scale_y_continuous(breaks = 1:10) +
  scale_fill_viridis_c()
```



```
[9]: # what about the rating as a function of lenght of the movie?
ggplot(df_movies, aes(x = runtimeMinutes, y = averageRating)) +
  geom_bin2d() +
  scale_x_continuous(breaks = seq(0, 180, 60), labels = 0:3) +
  scale_y_continuous(breaks = 0:10) +
  scale_fill_viridis_c(option = "inferno") +
  theme_minimal() +
  labs(title = "Relationship between Movie Runtime and Average Movie Rating",
       subtitle = "Data from IMDb",
       x = "Runtime (Hours)",
       y = "Average User Rating",
       fill = "# Movies")
```

Relationship between Movie Runtime and Average Movie Rating  
Data from IMDb



# Hands-on: meanwhile, in the background...

```
python3.13
To access the server, open this file in a browser:
file:///Users/rrossi/Library/Jupyter/runtime/jpserver-633-open.html
Or copy and paste one of these URLs:
http://localhost:8888/lab?token=291fd06b8c39b1ad76356362373468b903ec4425fb459dac
http://127.0.0.1:8888/lab?token=291fd06b8c39b1ad76356362373468b903ec4425fb459dac
[I 2024-11-22 18:36:46.093 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server
[W 2024-11-22 18:36:46.686 LabApp] Could not determine jupyterlab build status without nodejs
[I 2024-11-22 18:39:30.698 ServerApp] Creating new notebook in /RProjects/imdb
[I 2024-11-22 18:39:30.764 ServerApp] Writing notebook-signing key to /Users/rrossi/Library/Jupyter/notebook_secret
/opt/miniconda3/envs/tidyR/bin/jupyter-lab:10: DeprecationWarning: Parsing dates involving a day of month without a year specified is ambiguous
and fails to parse leap day. The default behavior will change in Python 3.15
to either always raise an exception or to use a different default year (TBD).
To avoid trouble, add a specific year to the input & format.
See https://github.com/python/cpython/issues/70647.
    sys.exit(main())
[I 2024-11-22 18:39:32.444 ServerApp] Kernel started: 93362536-ee59-4dfa-8b7d-1b85f7a2b410
[I 2024-11-22 18:39:44.007 ServerApp] Connecting to kernel 93362536-ee59-4dfa-8b7d-1b85f7a2b410.
[I 2024-11-22 18:39:44.024 ServerApp] Connecting to kernel 93362536-ee59-4dfa-8b7d-1b85f7a2b410.
[I 2024-11-22 18:39:44.042 ServerApp] Connecting to kernel 93362536-ee59-4dfa-8b7d-1b85f7a2b410.
[I 2024-11-22 18:41:31.851 ServerApp] Saving file at /RProjects/imdb/Untitled.ipynb
[I 2024-11-22 18:43:31.875 ServerApp] Saving file at /RProjects/imdb/Untitled.ipynb
[I 2024-11-22 18:45:56.985 ServerApp] Saving file at /RProjects/imdb/myimdb.ipynb
[I 2024-11-22 18:47:57.032 ServerApp] Saving file at /RProjects/imdb/myimdb.ipynb
```

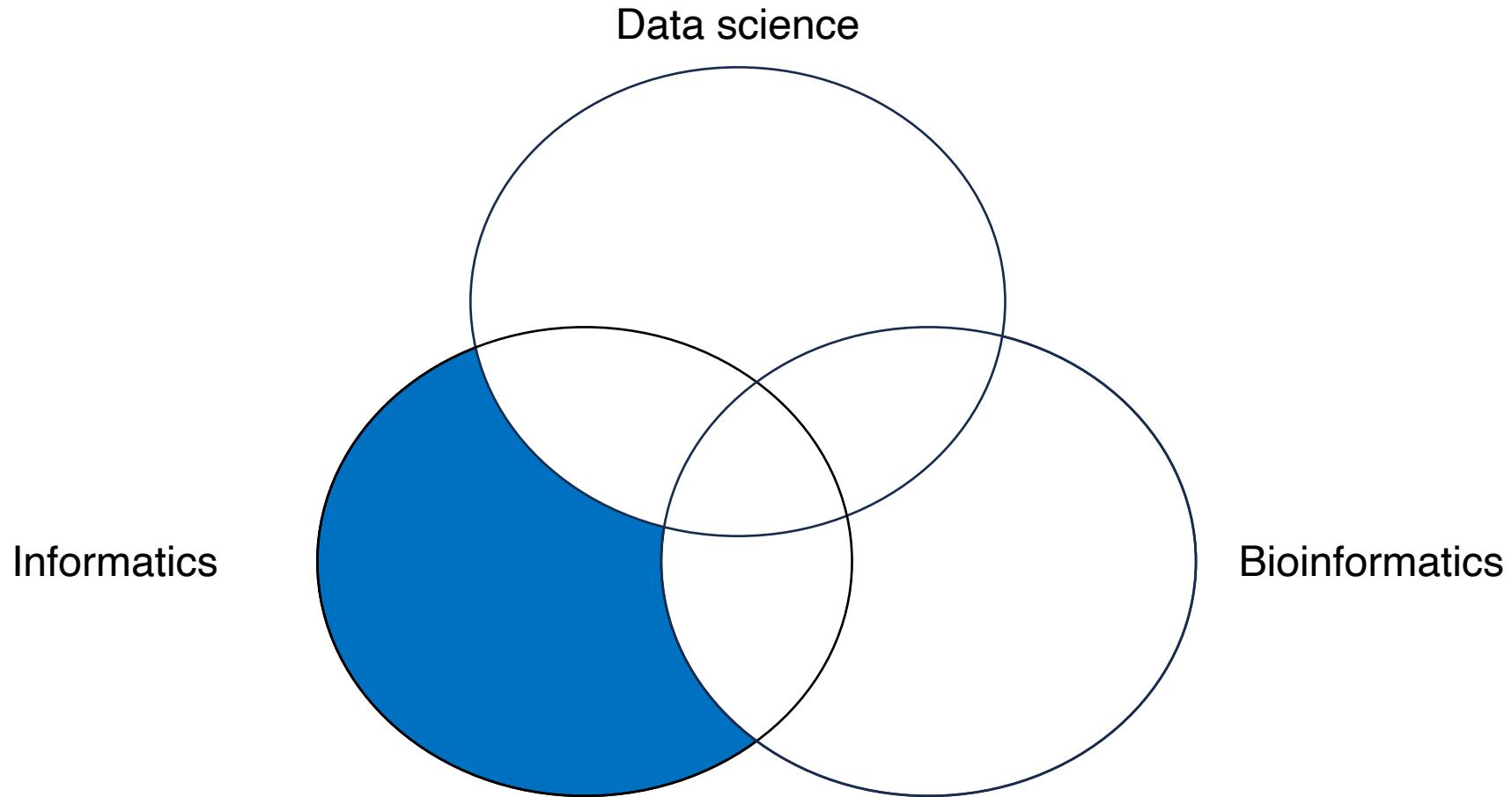
A self-sufficient  
universe

# Containers

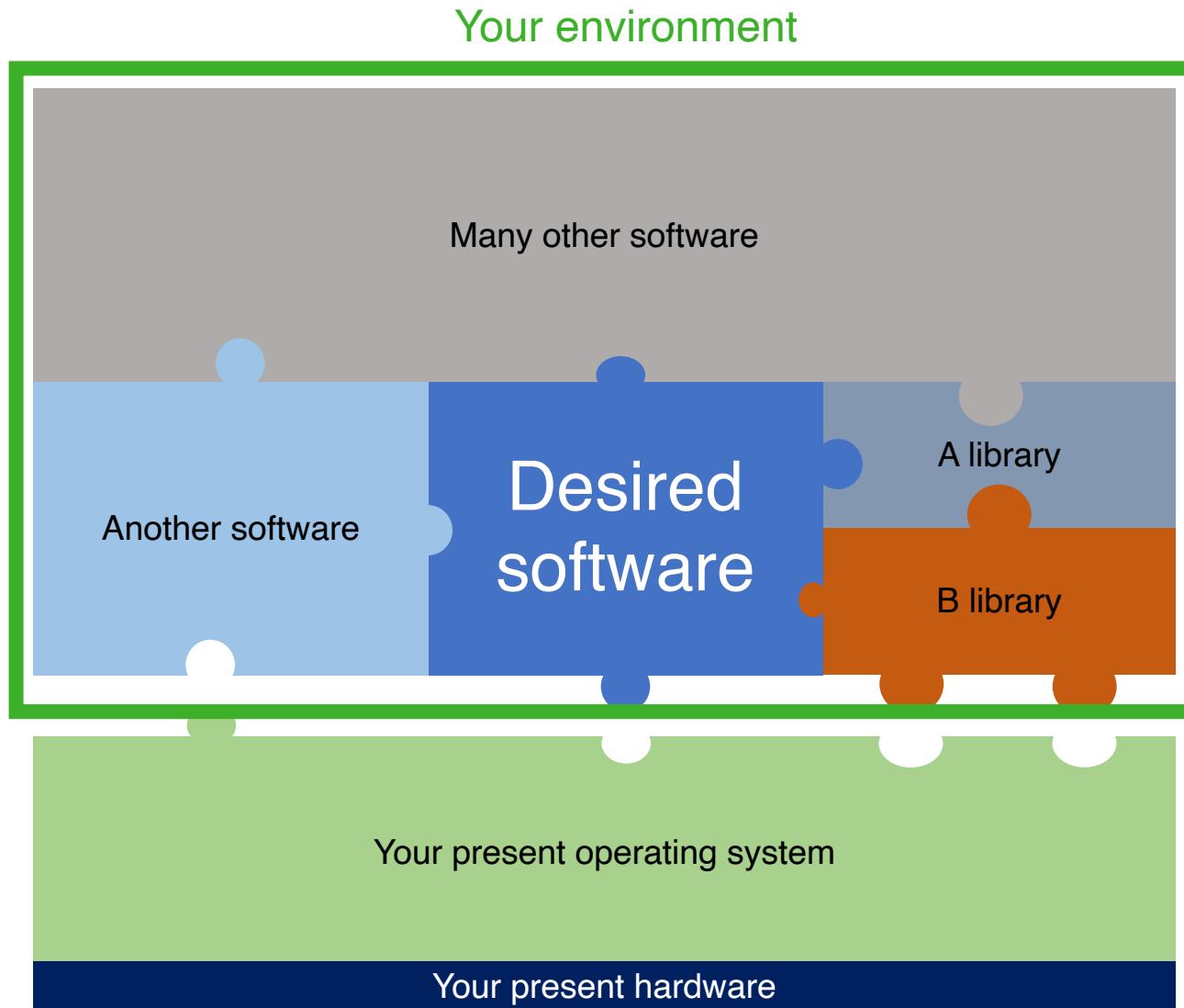
A step forward in reproducibility



# Topic domain: informatics



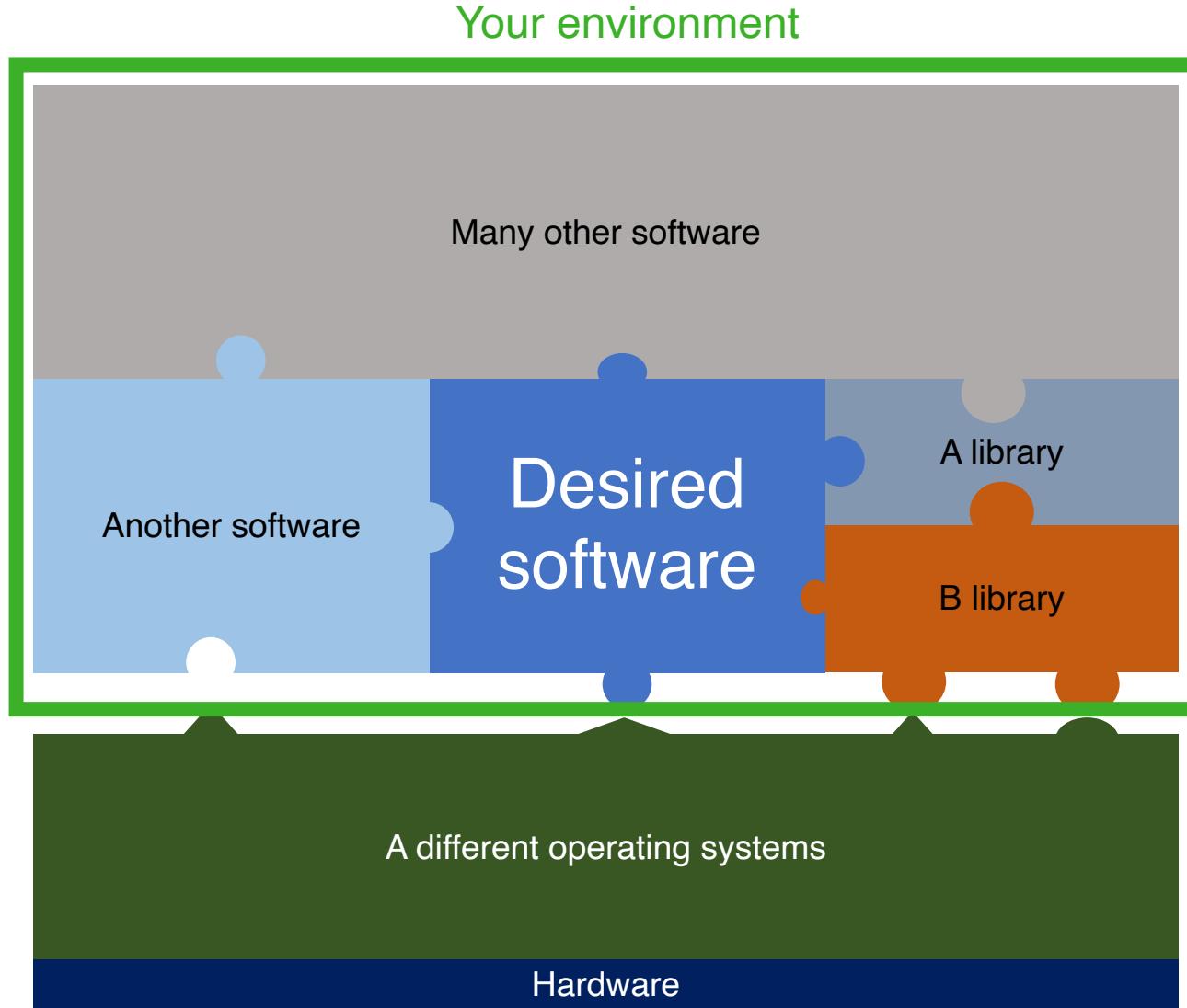
# The problem: reproducibility across time and space!



The operative systems:

- could be updated
- could be changed
- are many

# The problem: reproducibility across time and space!

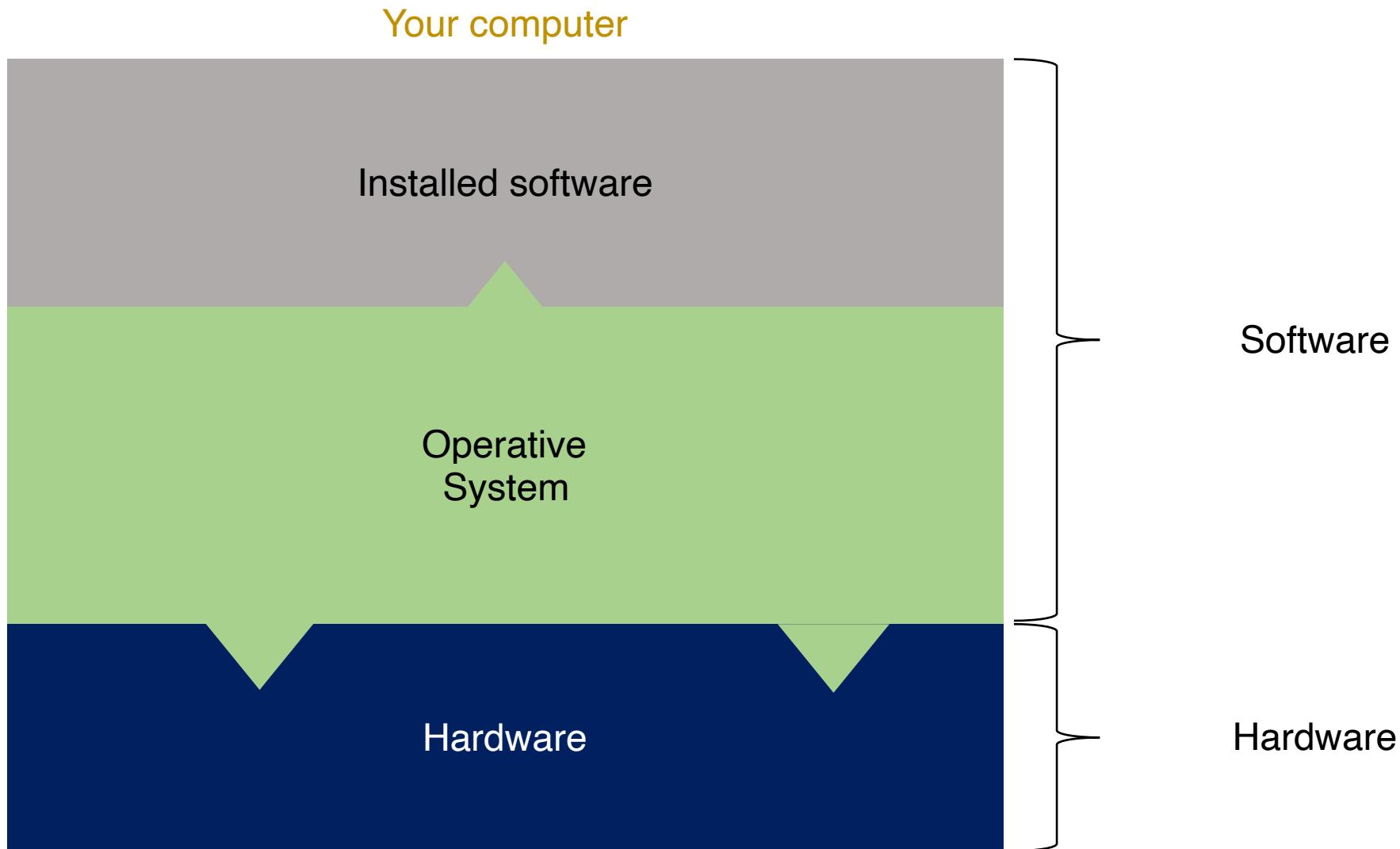


The operative systems:

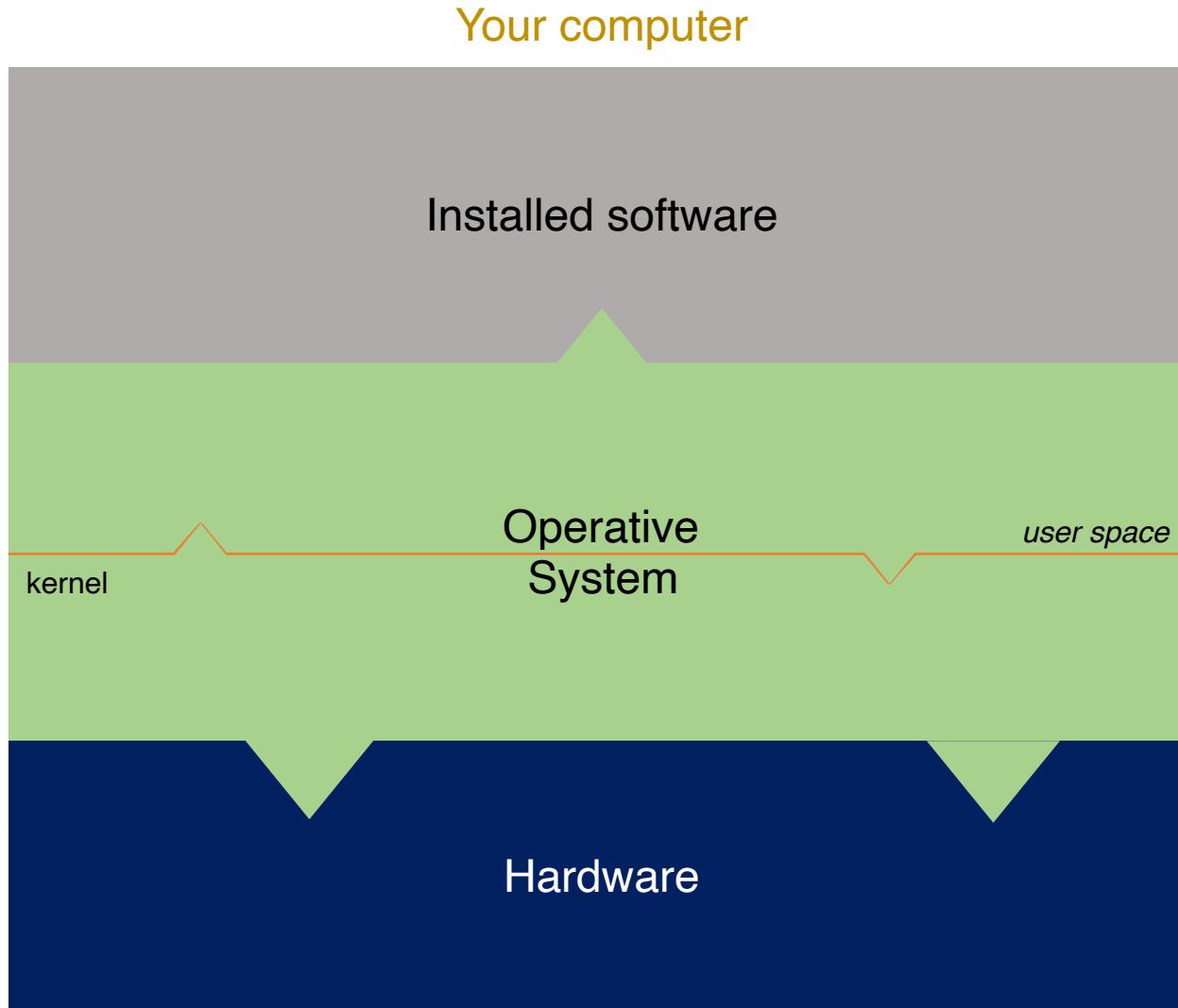
- could be updated
- could be changed
- are many

***Compatibility issue!***  
Reproducibility compromised!

# The solution: containers, freeze part of the operative system



# The solution: containers, freeze part of the operative system



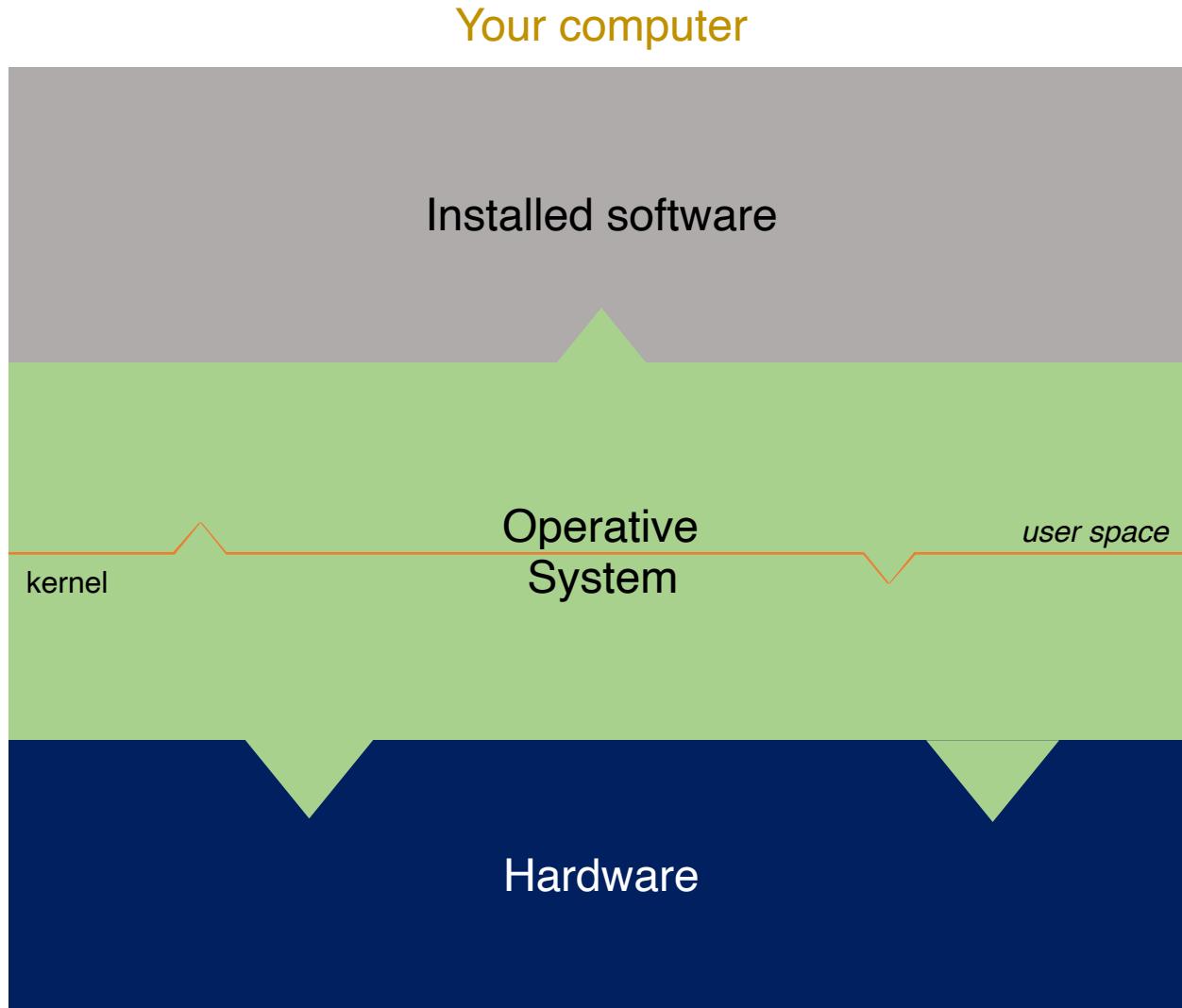
A Unix operating system is broken into two primary components:

- the kernel space
- the user space.

The Kernel talks to the hardware, and provides core system features.

The user space is the environment that most people are most familiar with. It is where applications, libraries and system services run.

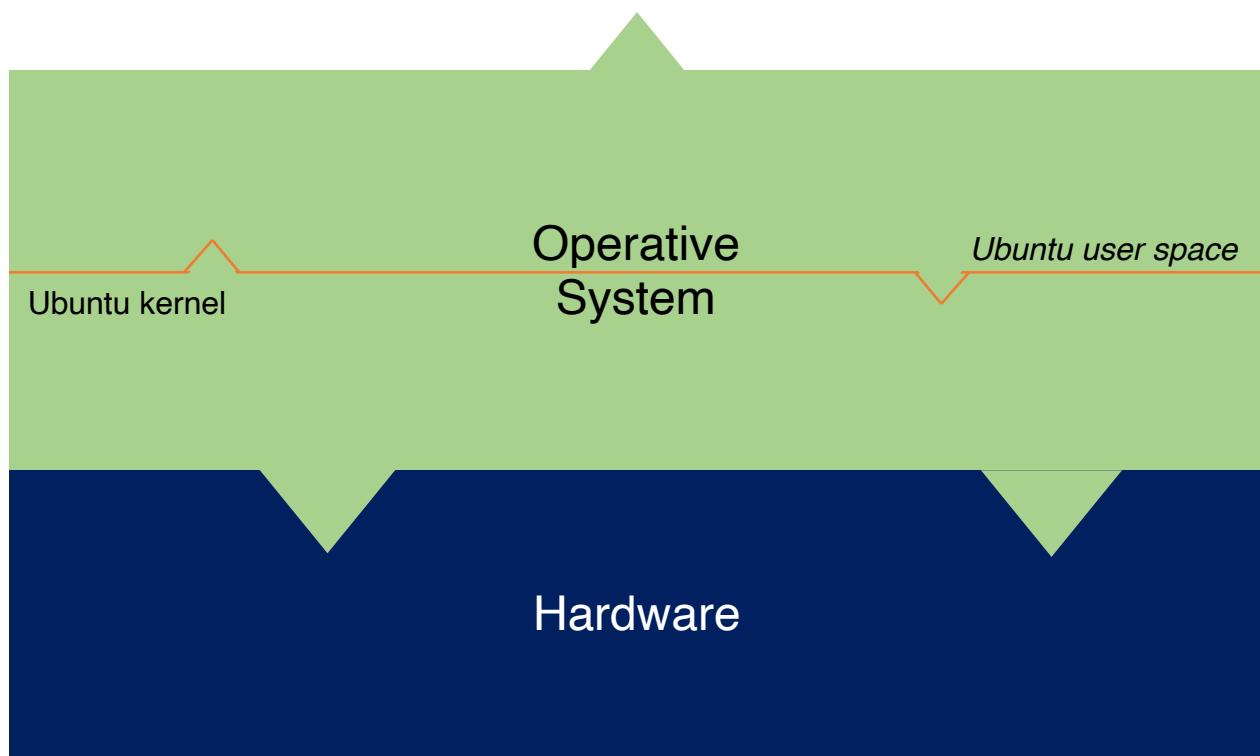
# The solution: containers, freeze part of the operative system



**Containers** change the user space into a swappable component.

This means that the entire user space portion of a Linux operating system, including programs, custom configurations, and environment can be independent of **whether your system is...**

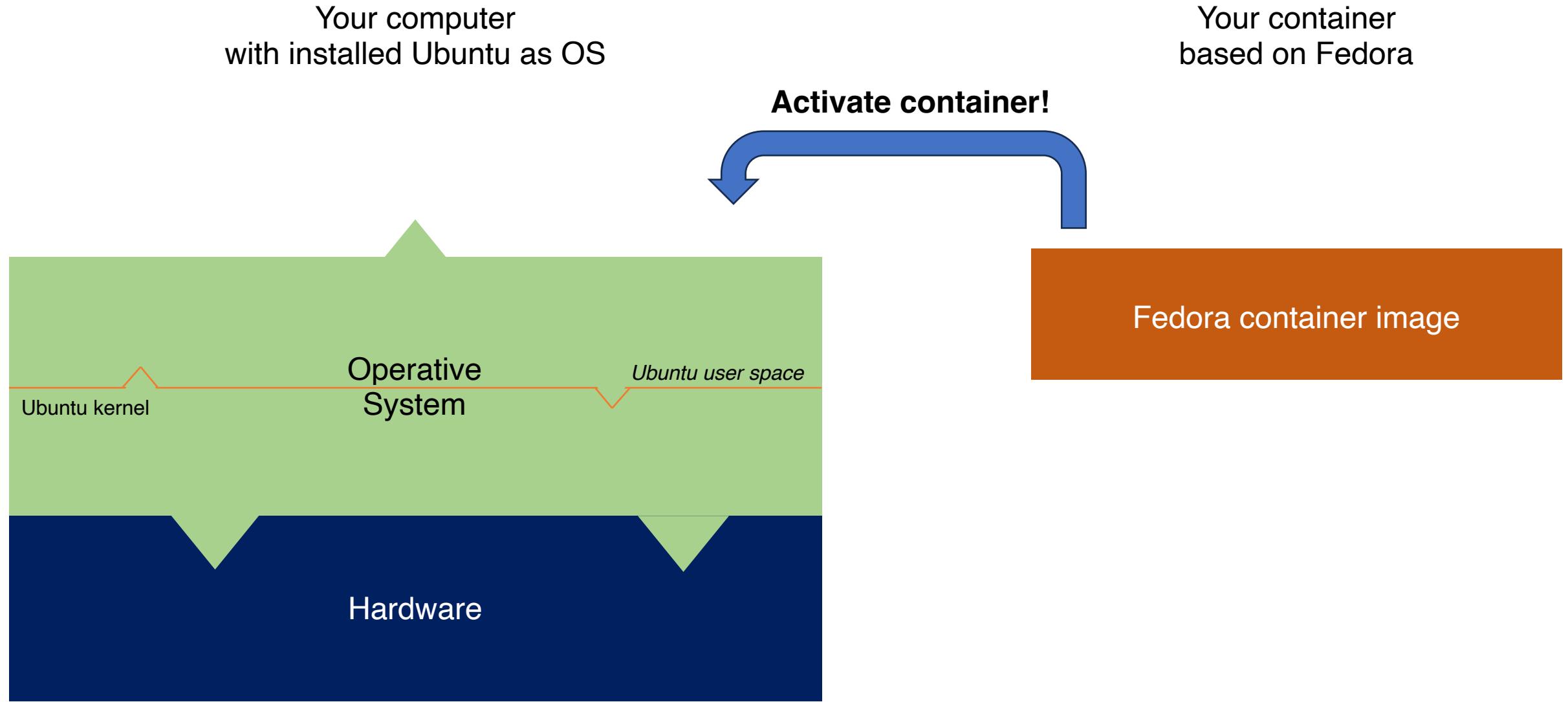
# An example



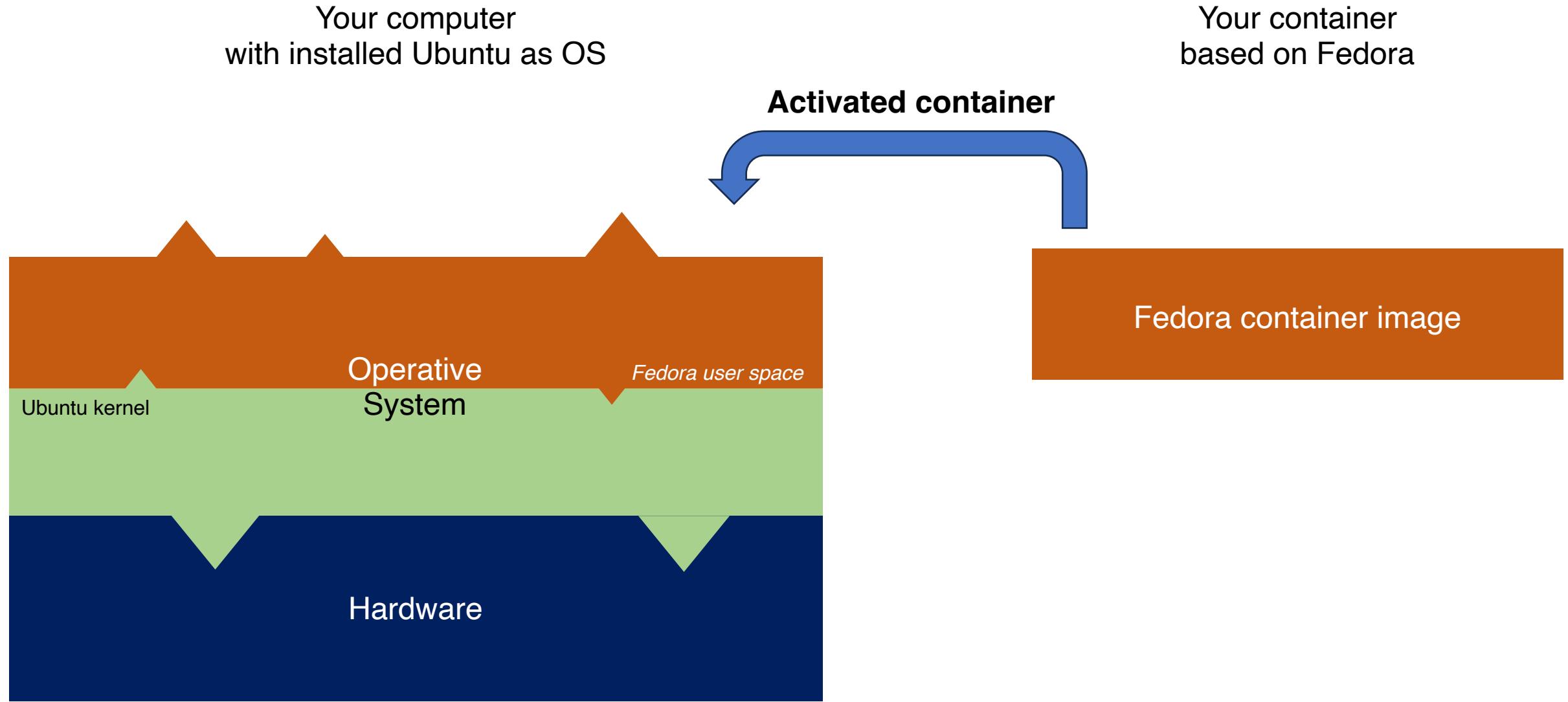
Your container  
based on Fedora



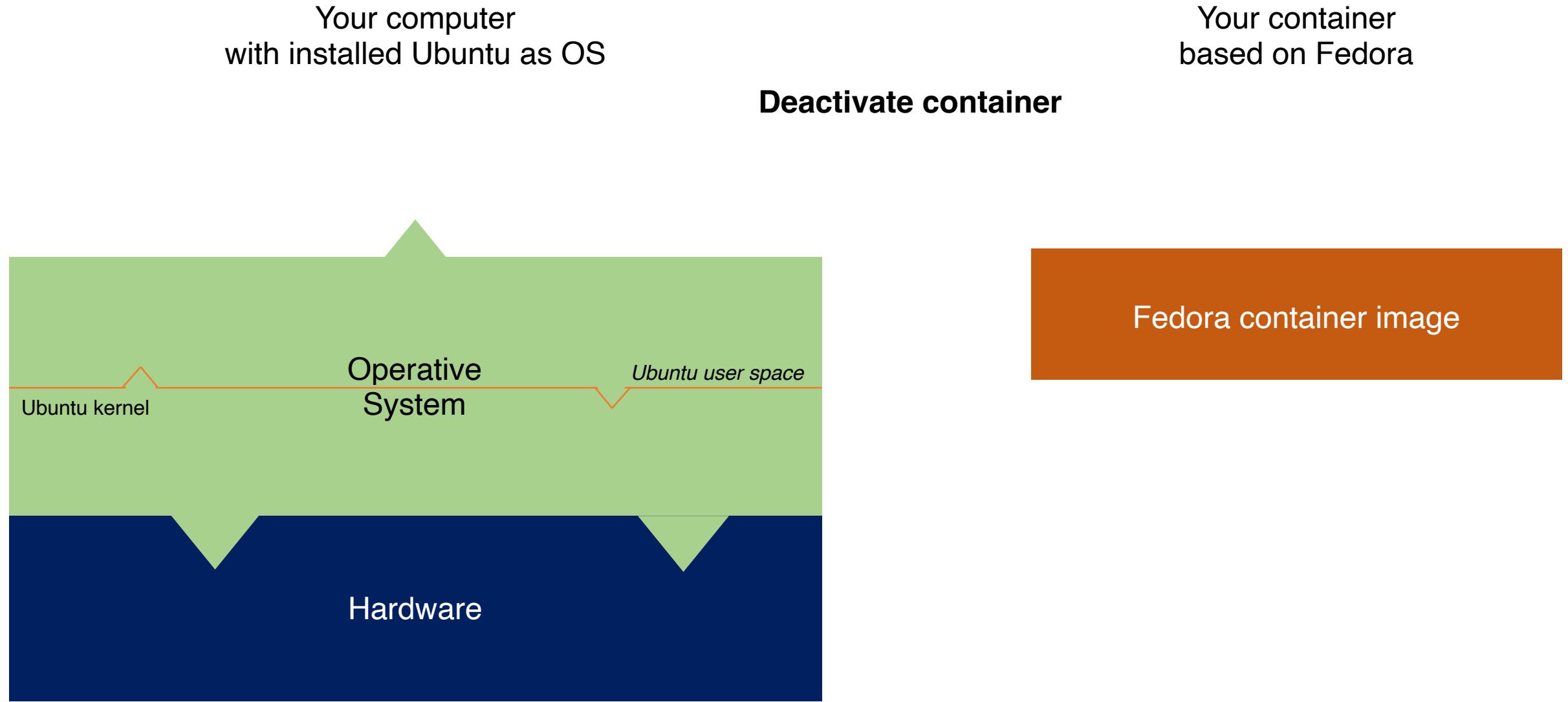
# An example



# An example

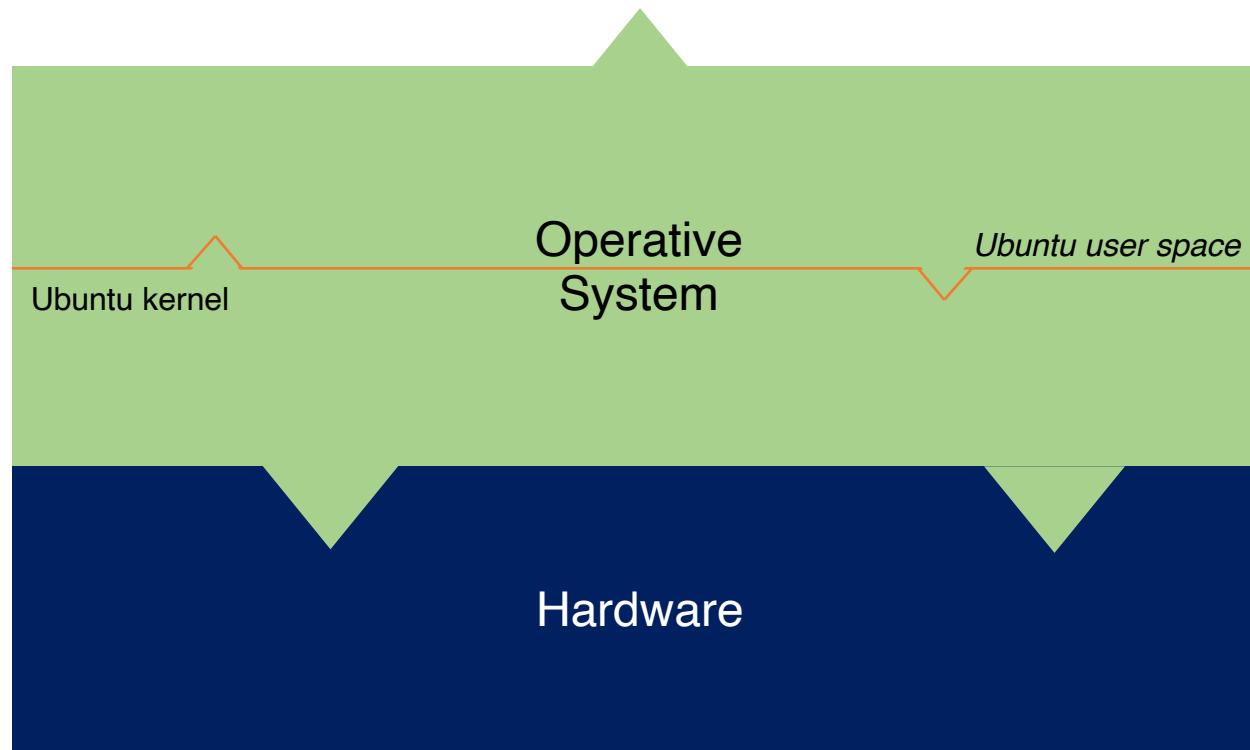


# An example



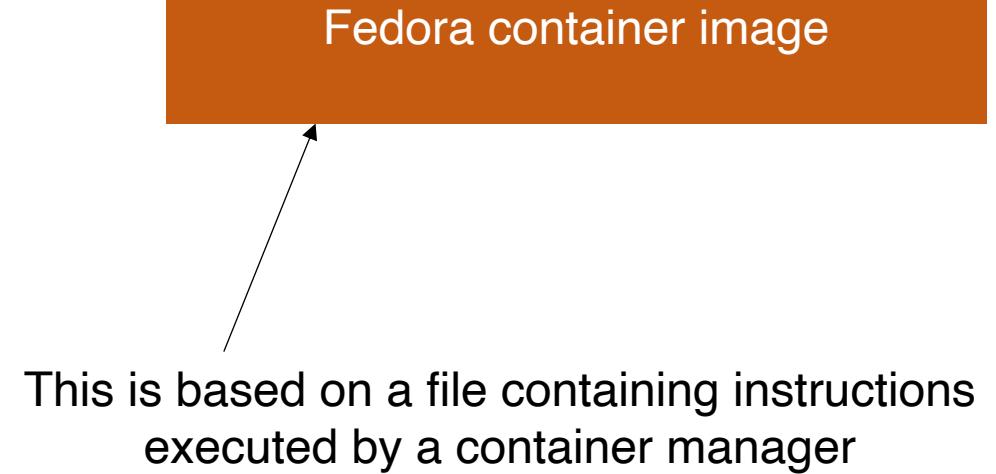
# An example, comments

Your computer  
with installed Ubuntu as OS



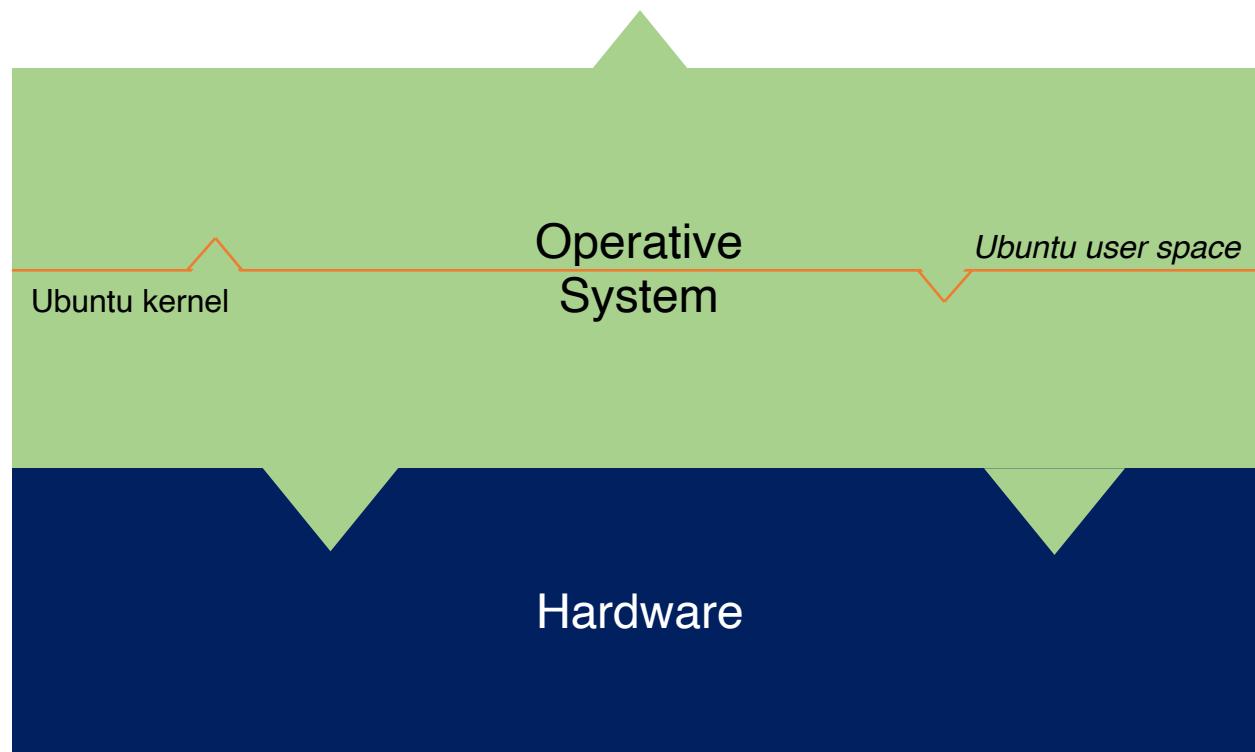
**Deactivate container**

Your container  
based on Fedora



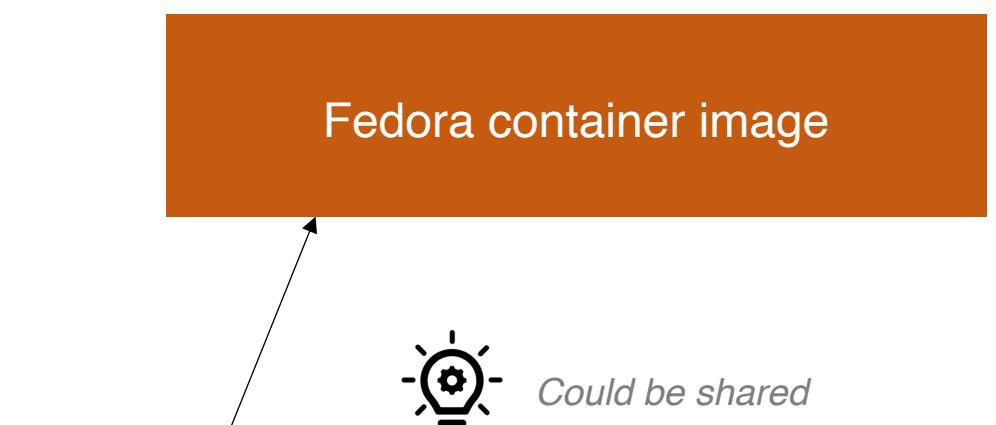
# An example, comments

Your computer  
with installed Ubuntu as OS



**Deactivate container**

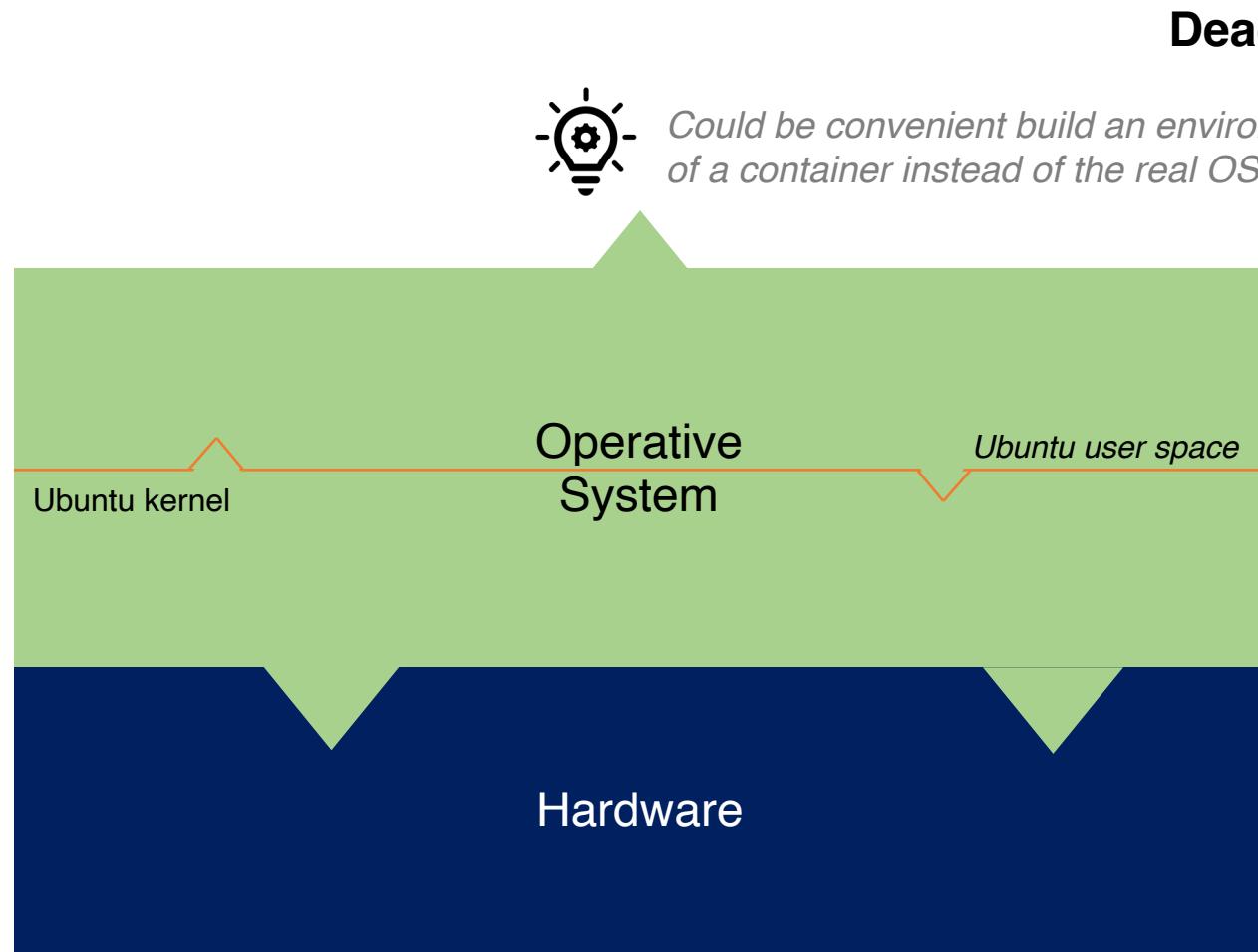
Your container  
based on Fedora



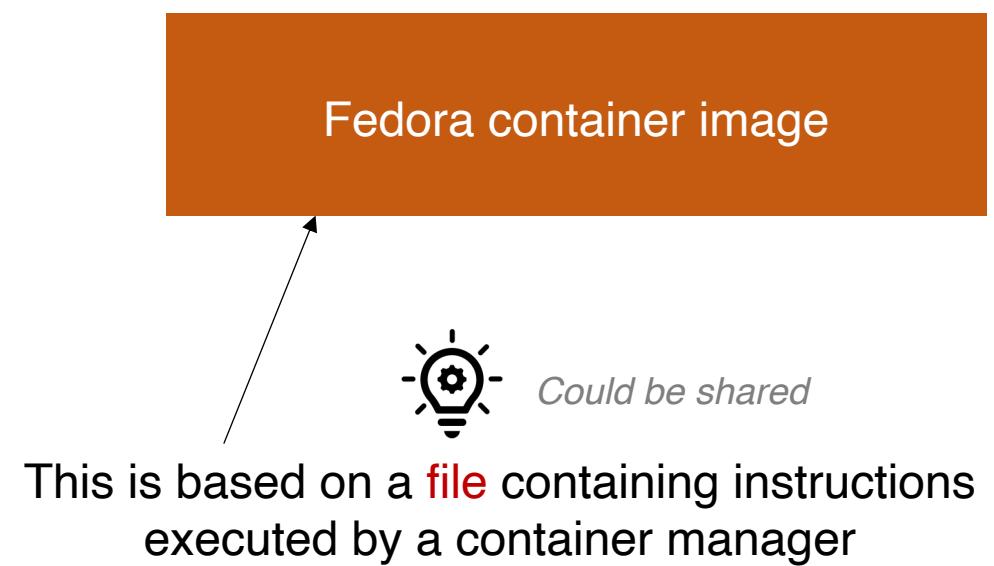
This is based on a **file** containing instructions  
executed by a container manager

# An example, comments

Your computer  
with installed Ubuntu as OS

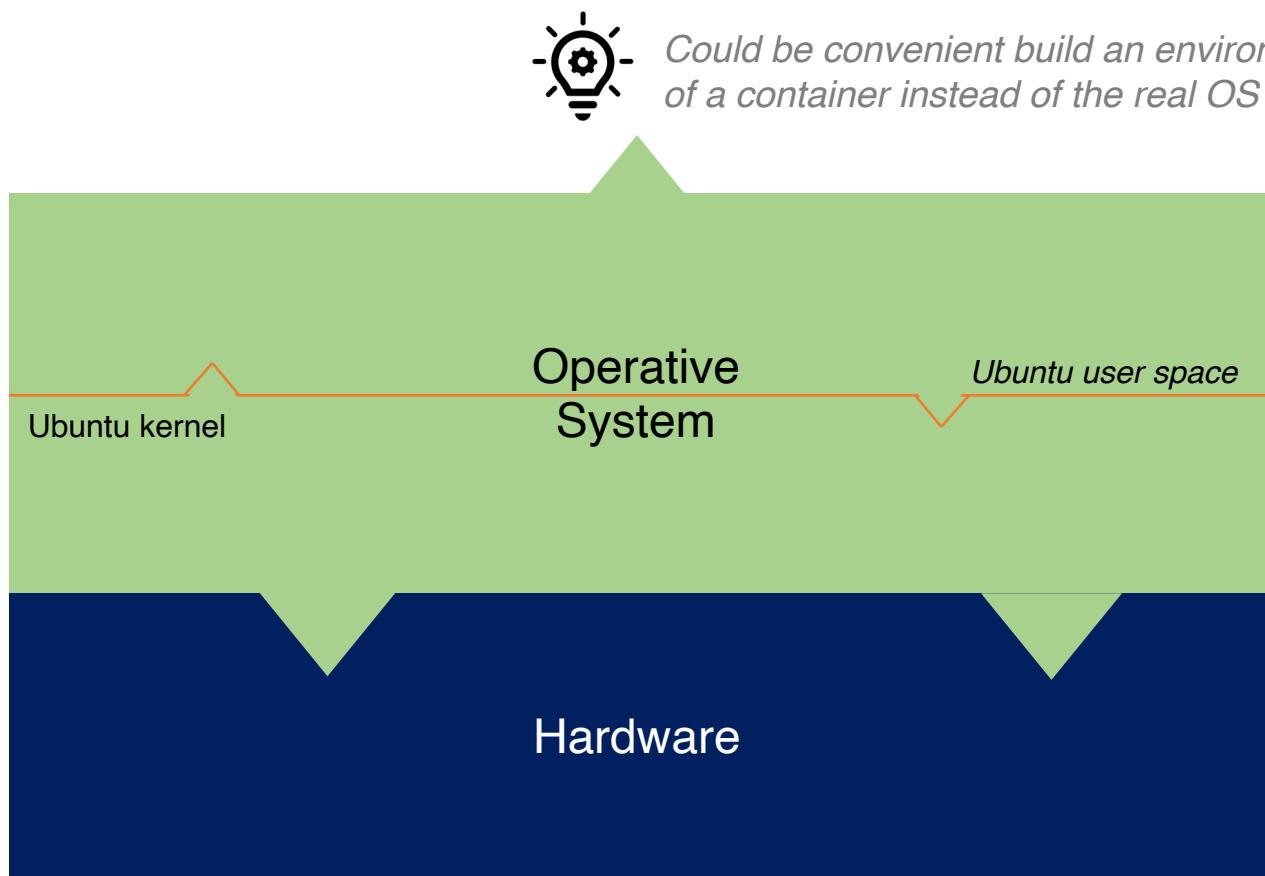


Your container  
based on Fedora



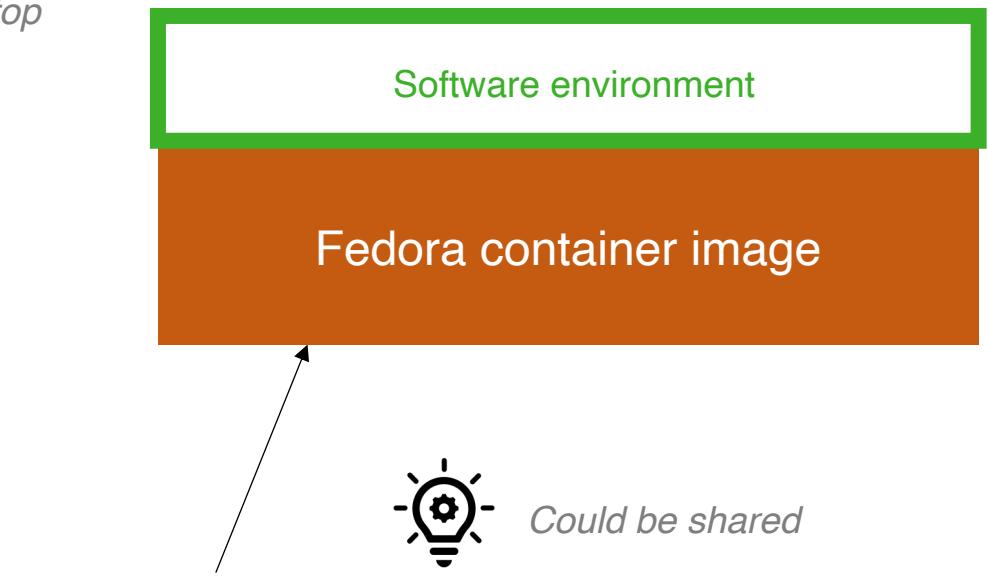
# An example, comments

Your computer  
with installed Ubuntu as OS



## Deactivate container

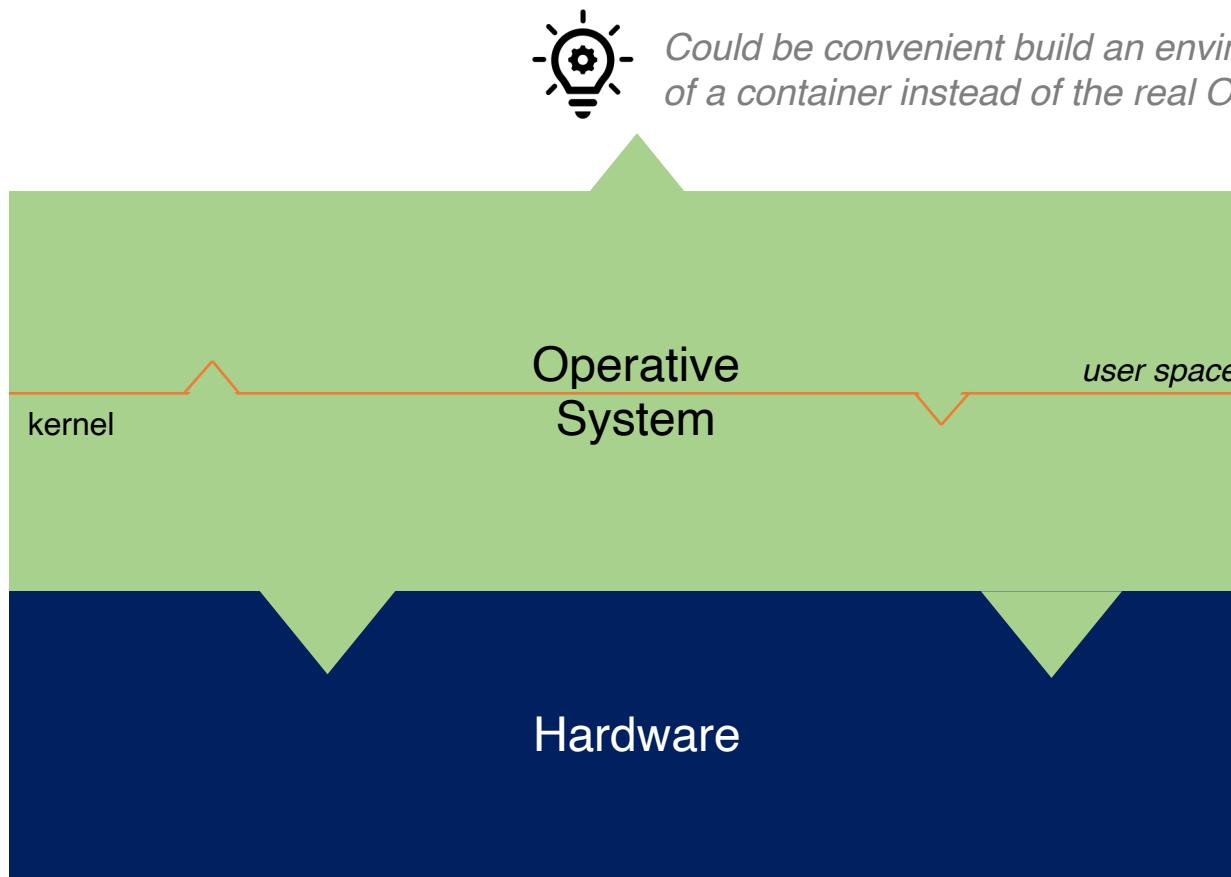
Your container  
based on Fedora



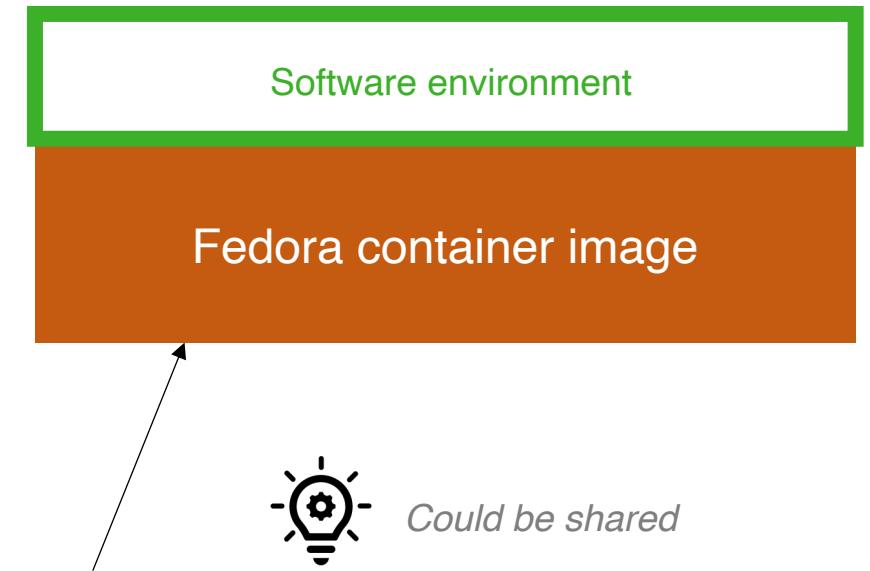
This is based on a **file** containing instructions  
executed by a container manager

# You can run whatever container on whatever operative system

Your computer



Your container



# Recapitulating...



Conda environment on top of your OS



Conda environment on top of a container

Dependent by:

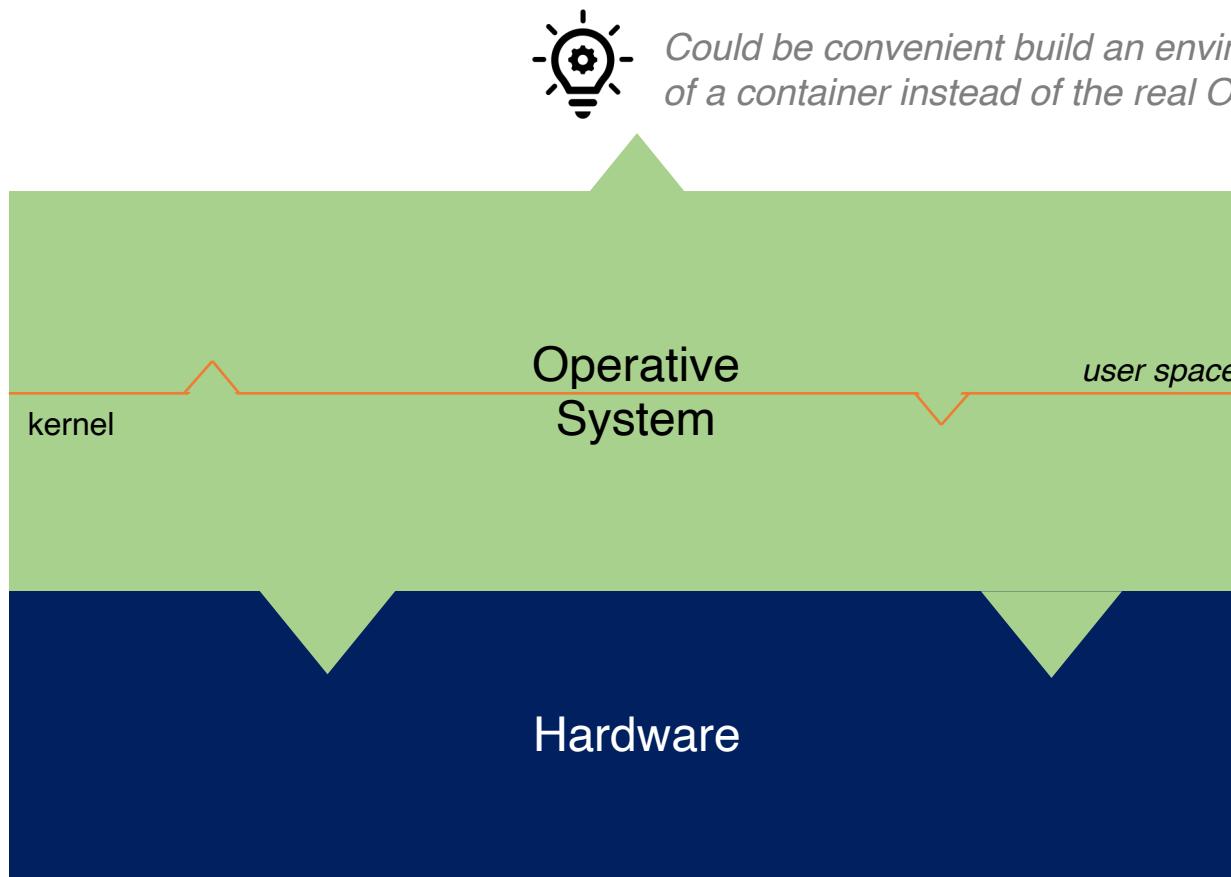
- your present hardware
- your present operative system

Reproducibility requires:

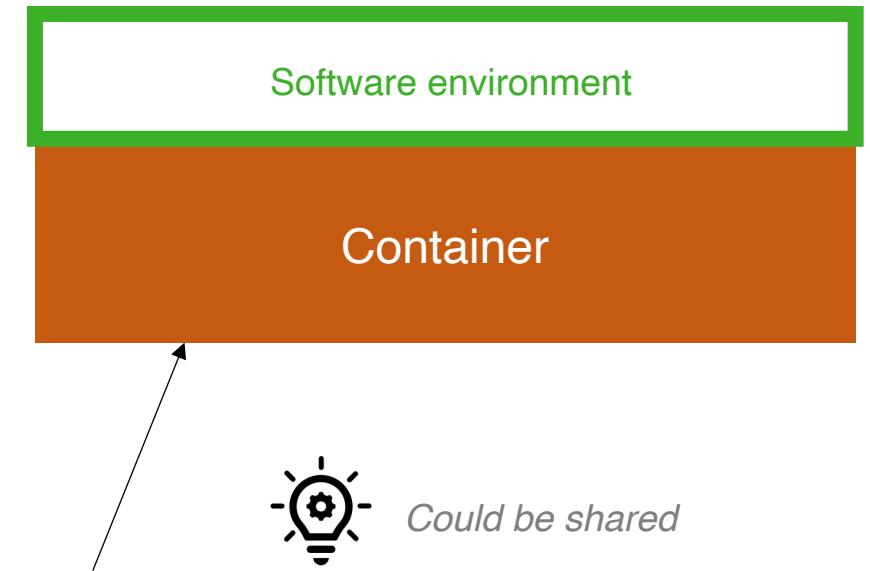
- instruction sharing
- container file

# You can run whatever container on whatever operative system

Your computer

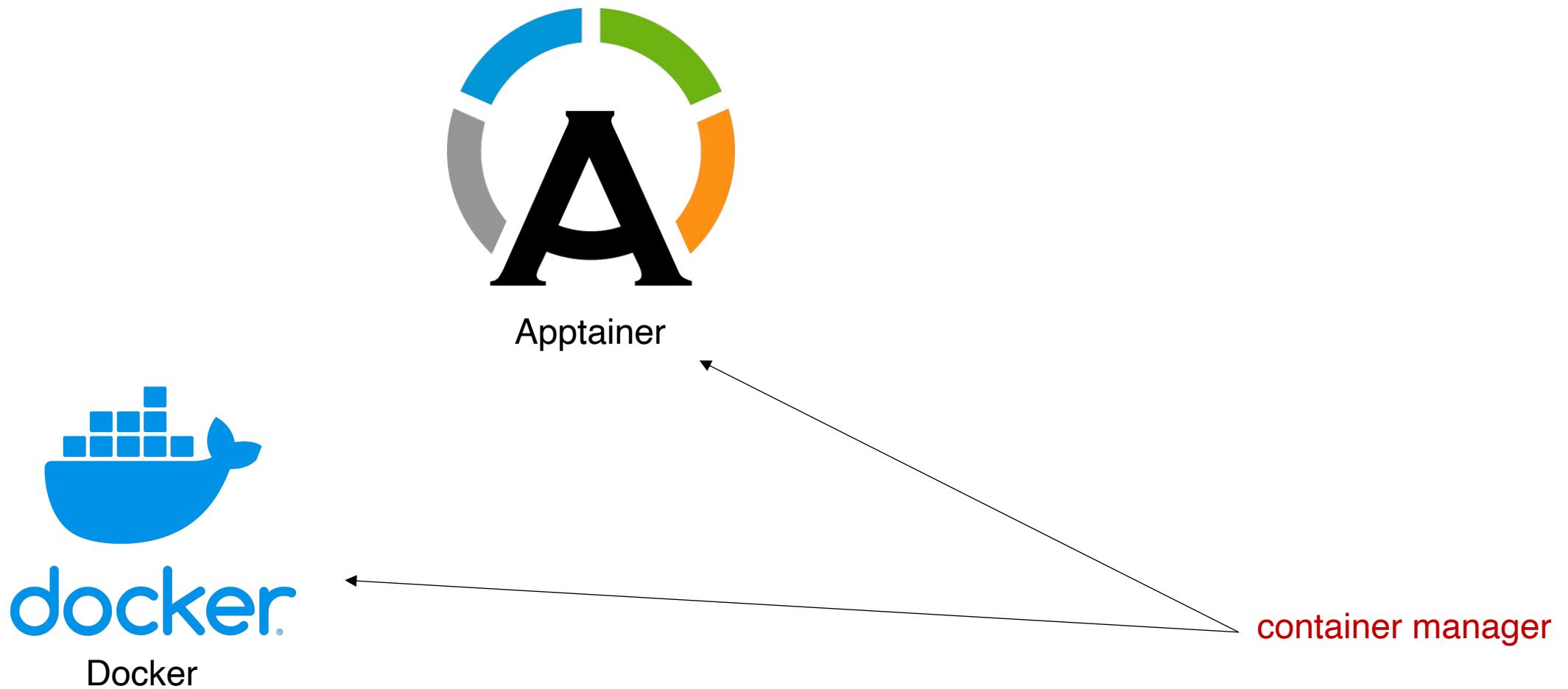


Your container

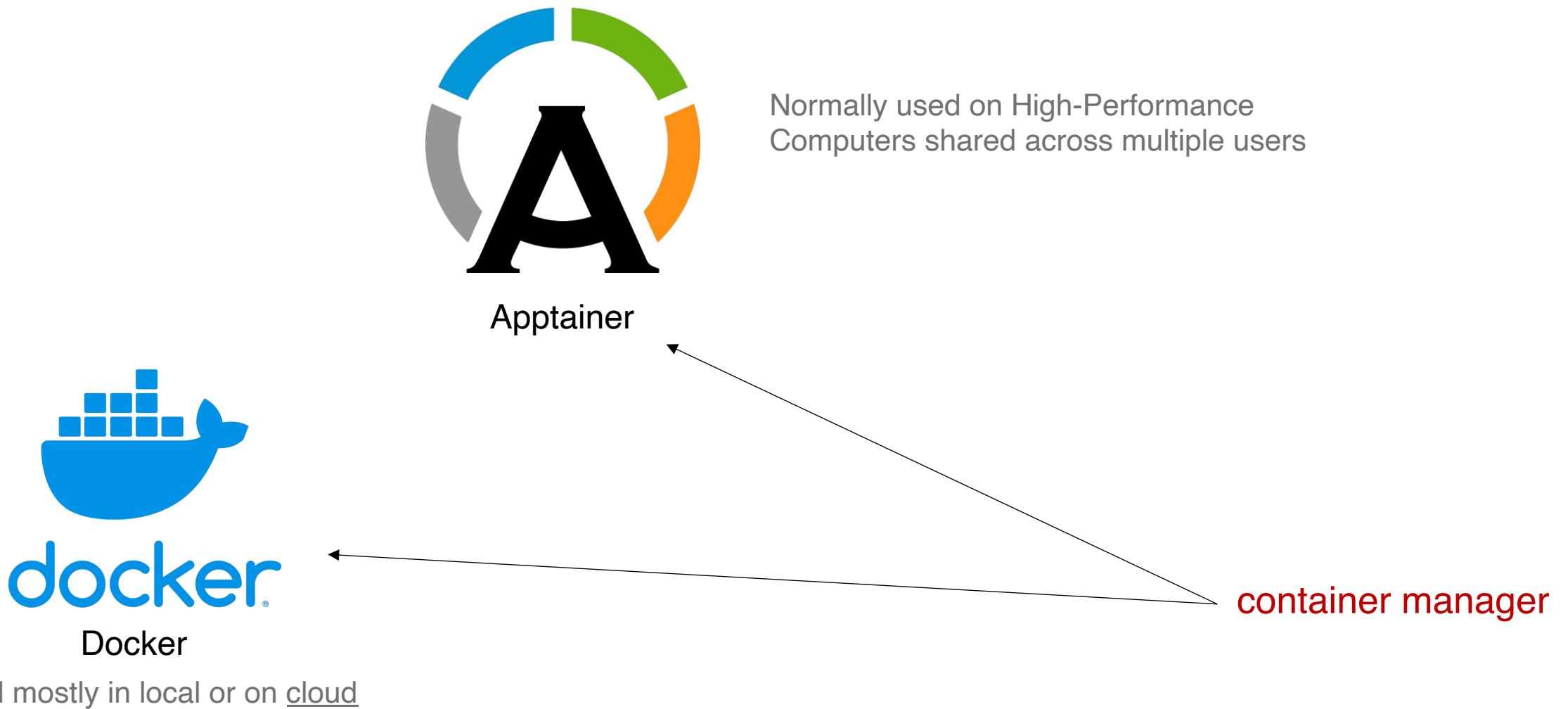


This is based on a file containing instructions  
executed by a **container manager**

# You can run whatever container on whatever operative system



# You can run whatever container on whatever operative system



# Hands-on: build your RedHat based container

# Hands-on: build your RedHat based container

0. Let's start creating our container instruction

# Hands-on: build your RedHat based container



terminal

```
mkdir -p containers # Create the folder where put your dockerfile
cd containers # Positionate yourself into the dockerfile folder
touch Dockerfile # Create the file containing the instruction to build your container
vi Dockerfile # Edit the Dockerfile, I am using vi, you can use whatever you like (e.g.nano, emacs, sublimetext)
```

## 0. Let's start creating our container instruction

# Hands-on: build your RedHat based container



terminal

```
$ cat Dockerfile
FROM registry.access.redhat.com/ubi9/ubi:9.2

CMD [ "bash"]
LABEL maintainer="IFOM"
ENV LANG=C.UTF-8 LC_ALL=C.UTF-8
```

1. Define the operative system (we will use RedHat 9.2)

# Hands-on: build your RedHat based container



```
$ cat Dockerfile
FROM registry.access.redhat.com/ubi9/ubi:9.2

CMD [ "bash"]
LABEL maintainer="IFOM"
ENV LANG=C.UTF-8 LC_ALL=C.UTF-8
```

**FROM** Create a new build stage from a base image.

1. Define the operative system (we will use RedHat 9.2)

# Hands-on: build your RedHat based container

A terminal window titled "terminal" displays the contents of a Dockerfile. The file starts with `FROM`, which is circled in red. Below it are `CMD`, `LABEL`, and `ENV` instructions, each also circled in red.

```
$ cat Dockerfile
FROM registry.access.redhat.com/ubi9/ubi:9.2
CMD [ "bash" ]
LABEL maintainer="IFOM"
ENV LANG=C.UTF-8 LC_ALL=C.UTF-8
```

**FROM** Create a new build stage from a base image.

1. Define the operative system (we will use RedHat 9.2)

<b>CMD</b>	Specify default commands.
<b>LABEL</b>	Add metadata to an image.
<b>ENV</b>	Set environment variables.

# Hands-on: build your RedHat based container

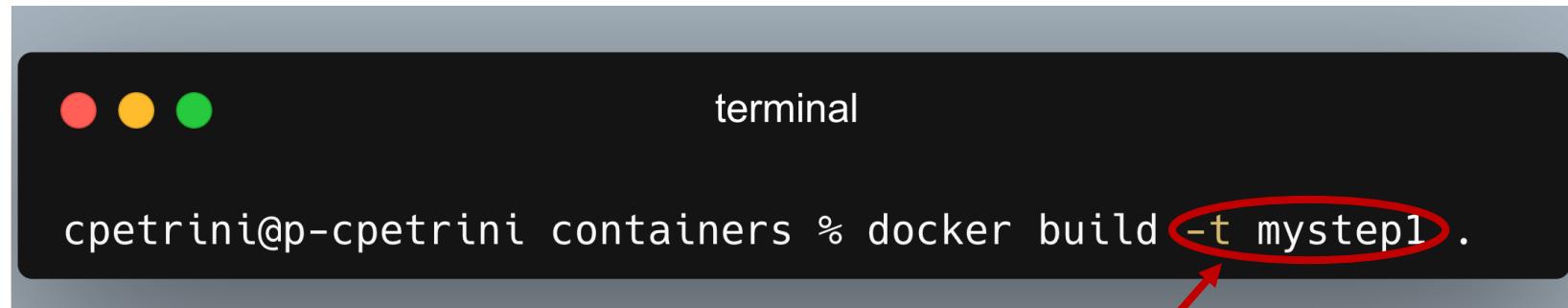


terminal

```
cpetrini@p-cpetrini containers % docker build -t mystep1 .
```

1. Define the operative system (we will use RedHat 9.2), **build it**

# Hands-on: build your RedHat based container

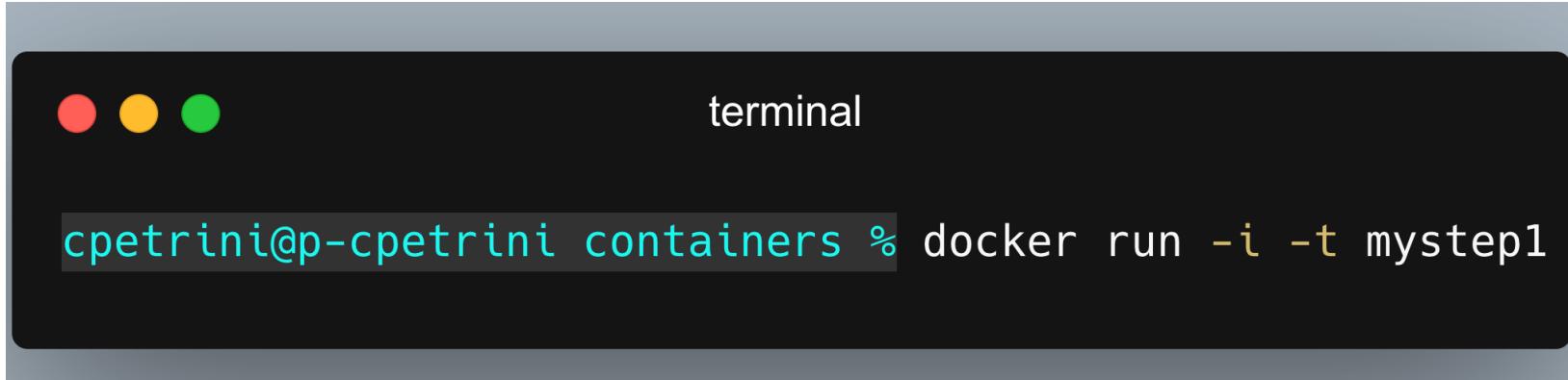


```
cpetrini@p-cpetrini containers % docker build -t mystep1 .
```

Describe the name of the *user space* that will run on top of the kernel

1. Define the operative system (we will use RedHat 9.2), **build it**

# Hands-on: build your RedHat based container



1. Define the operative system (we will use RedHat 9.2), build **and run it**

# Hands-on: build your RedHat based container



terminal

```
cpetrini@p-cpetrini containers % docker run -i -t mystep1
```



```
cpetrini@p-cpetrini containers % docker run -i -t mystep1  
[root@c8e614422846 /]#
```

1. Define the operative system (we will use RedHat 9.2), build **and run it**

# Hands-on: build your RedHat based container



terminal

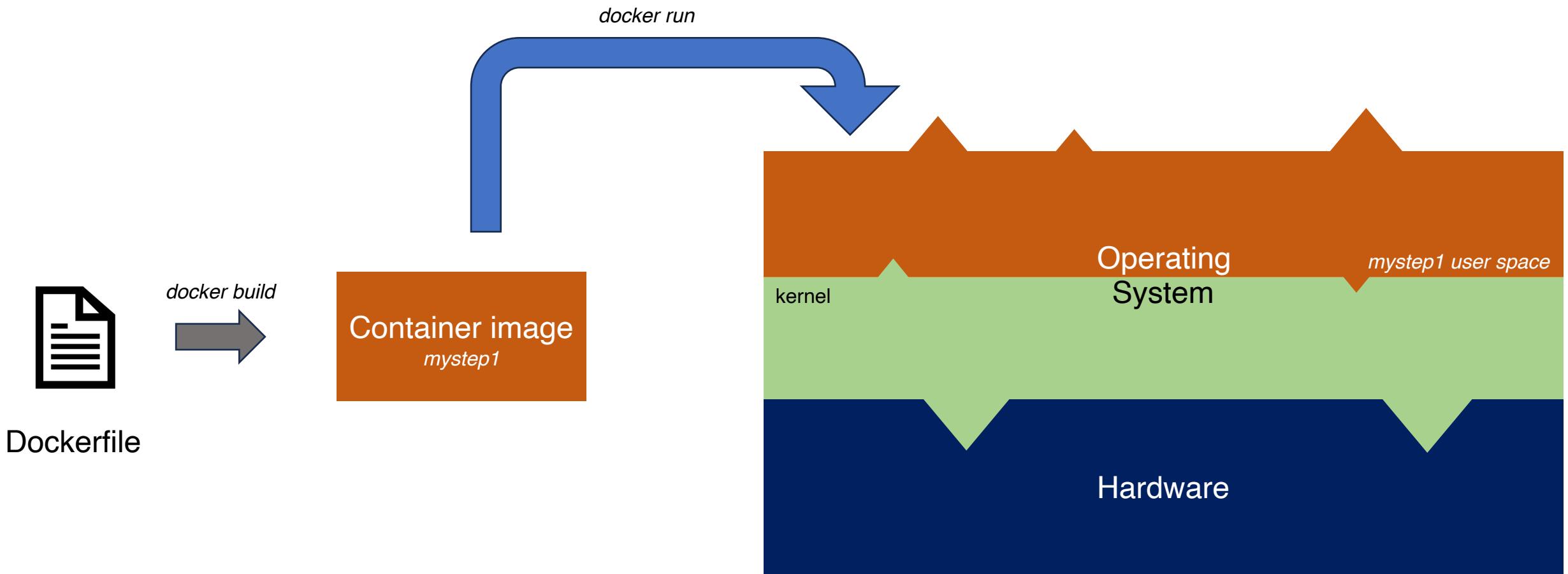
```
cpetrini@p-cpetrini containers % docker run -i -t mystep1
```



```
[root@c8e614422846 /]# less  
bash: less: command not found
```

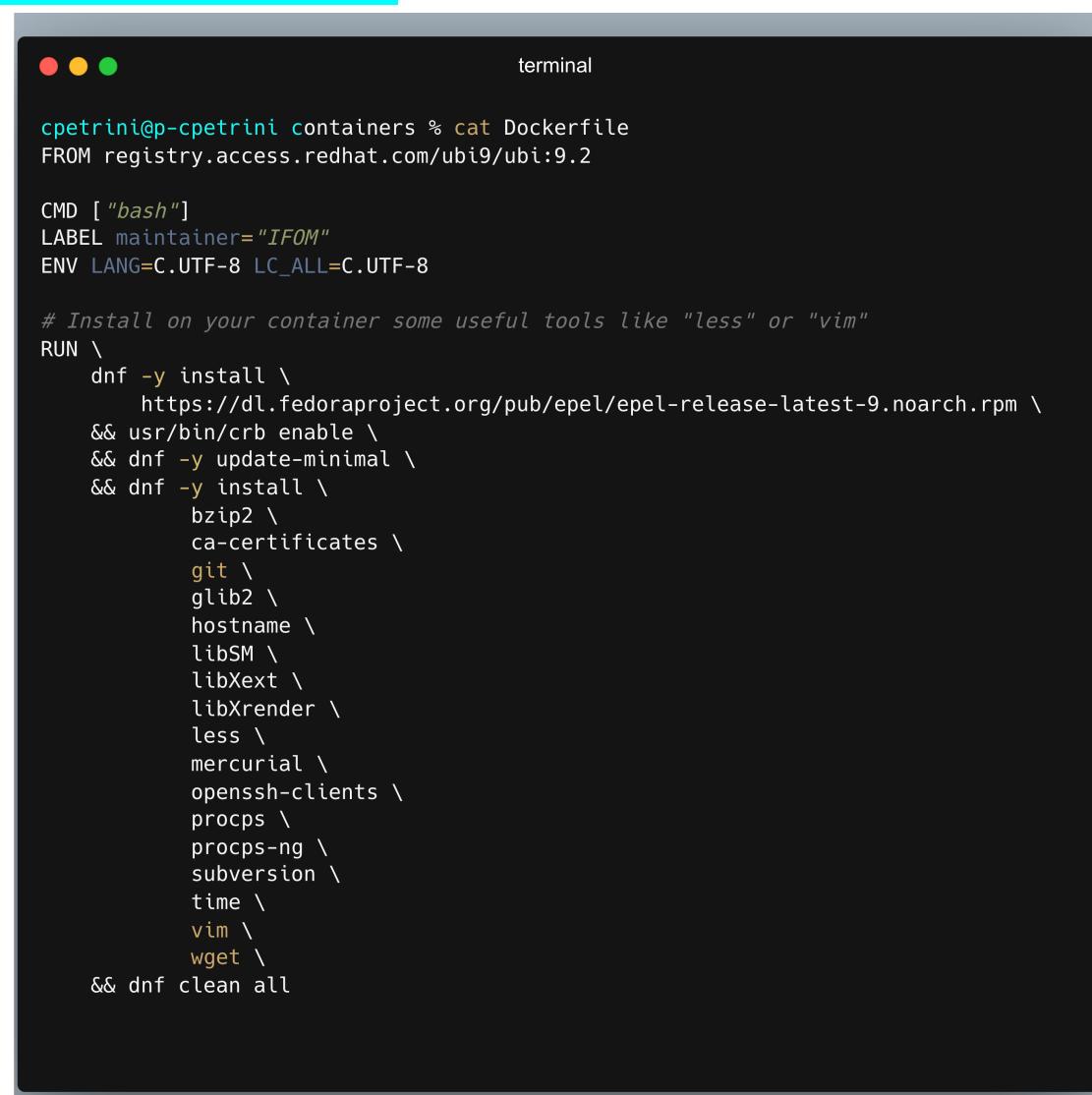
1. Define the operative system (we will use RedHat 9.2), build **and run it**

# What did it happen?!



# Hands-on: build your RedHat based container

2. Add some useful tools to our container (e.g. less)



The terminal window shows a Dockerfile being viewed. The file starts with a base image from the Red Hat registry and includes commands to install various utilities like less, vim, and wget.

```
cpetrini@p-cpetrini containers % cat Dockerfile
FROM registry.access.redhat.com/ubi9/ubi:9.2

CMD [ "bash"]
LABEL maintainer="IFOM"
ENV LANG=C.UTF-8 LC_ALL=C.UTF-8

# Install on your container some useful tools like "less" or "vim"
RUN \
    dnf -y install \
        https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm \
&& user/bin/crb enable \
&& dnf -y update-minimal \
&& dnf -y install \
    bzip2 \
    ca-certificates \
    git \
    glib2 \
    hostname \
    libSM \
    libXext \
    libXrender \
    less \
    mercurial \
    openssh-clients \
    procps \
    procps-ng \
    subversion \
    time \
    vim \
    wget \
&& dnf clean all
```

# Hands-on: build your RedHat based container



terminal

```
cpetrini@p-cpetrini containers % docker build -t mystep2 .
```

2. Add some useful tools to our container (e.g. less), **build it**

# Hands-on: build your RedHat based container



terminal

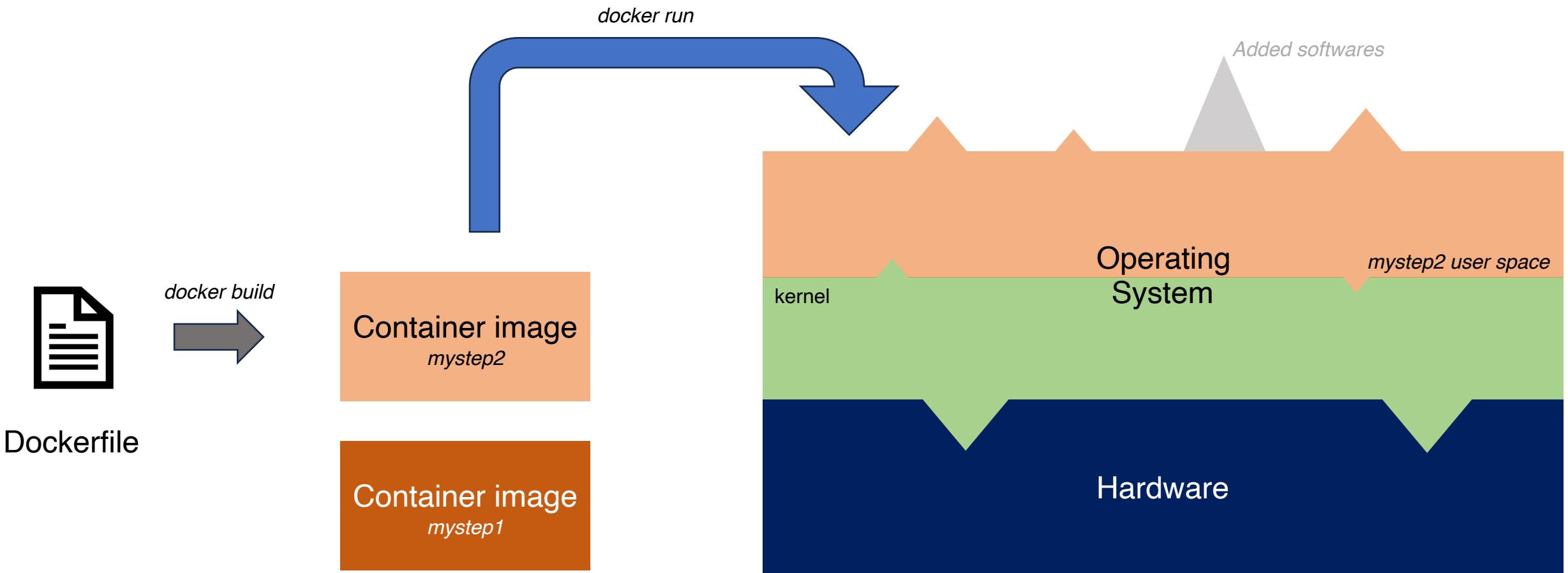
```
cpetrini@p-cpetrini containers % docker build -t mystep2 .
```



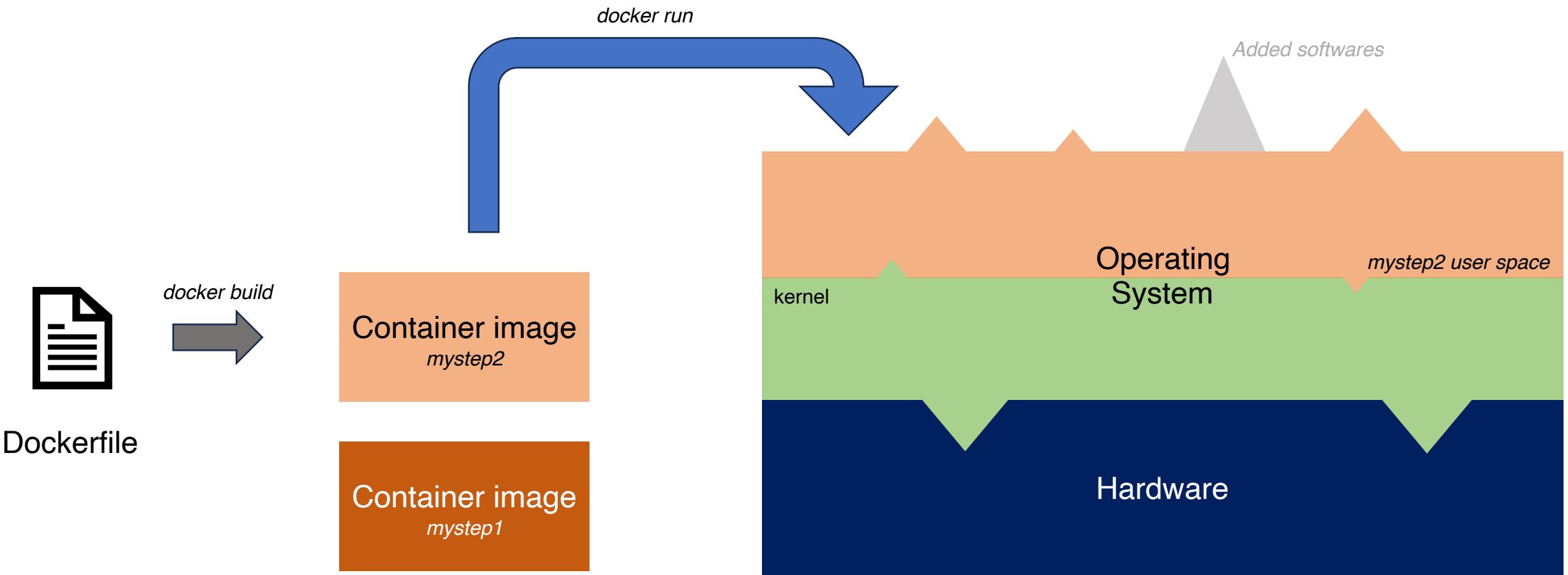
```
cpetrini@p-cpetrini containers % docker run -i -t mystep2
[root@5480c042ac14 /]# less
Missing filename ("less --help" for help)
[root@5480c042ac14 /]# █
```

2. Add some useful tools to our container (e.g. less), build **and** run it

# What did it happen?!

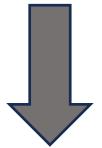
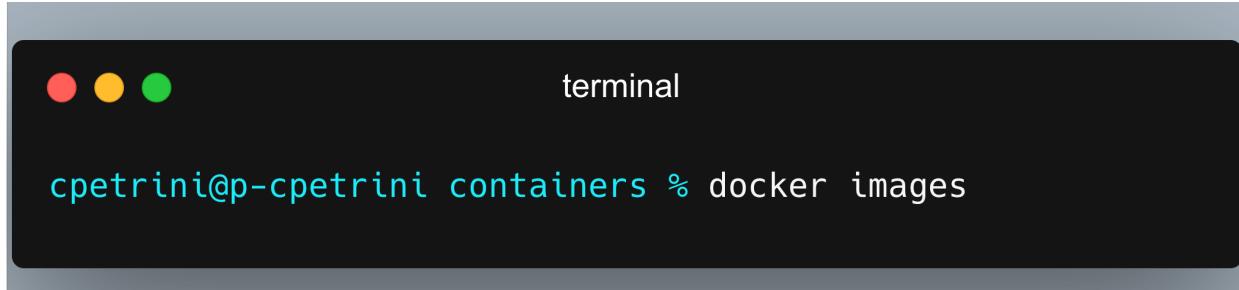


# What about *mystep1*!?



**It is still there and you can load it when you want!**

# Hands-on: build your RedHat based container

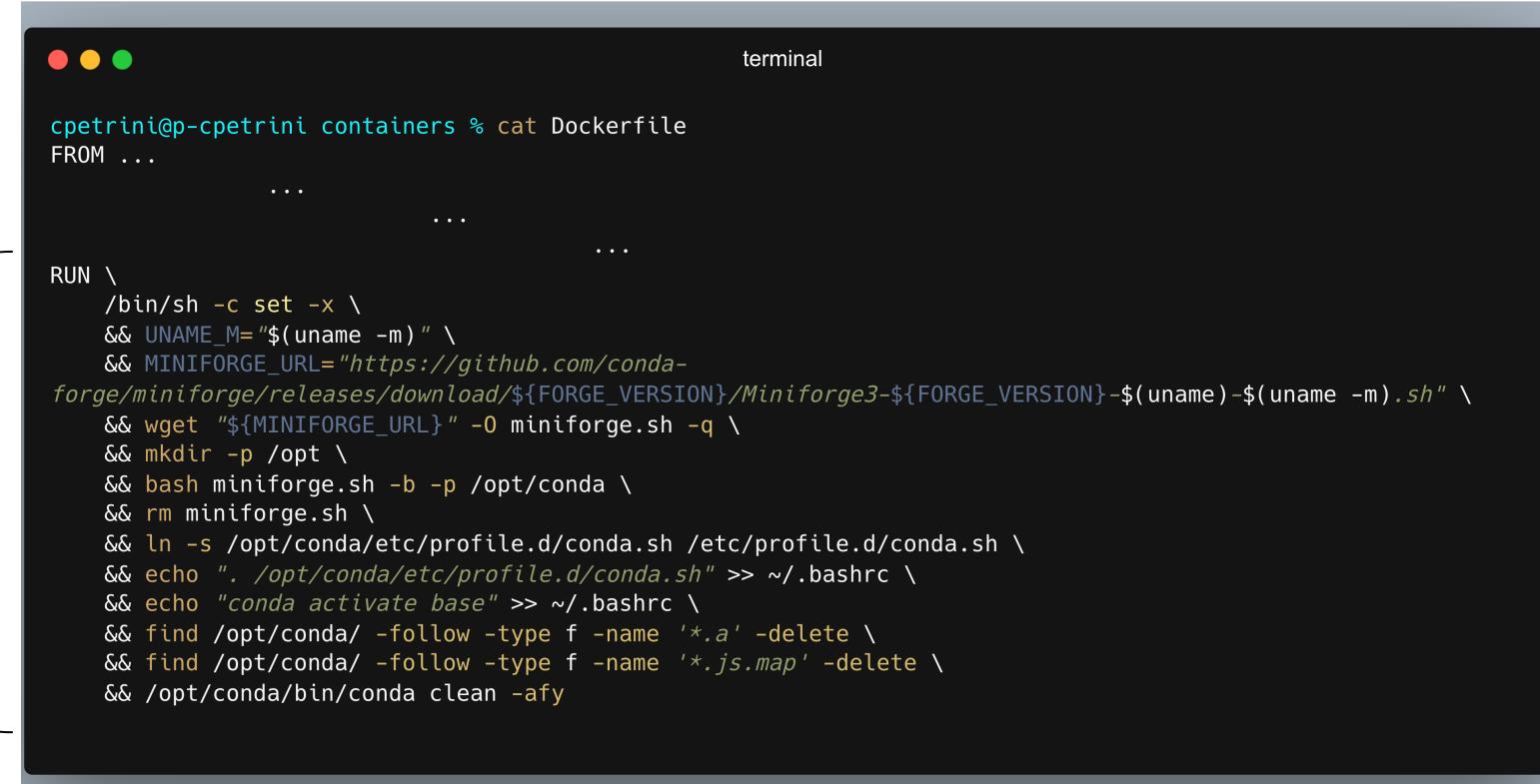


REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mystep2	latest	ec9b9409aa82	5 hours ago	578MB
mystep1	latest	b1dc540c596a	13 months ago	313MB
cpetrini@p-cpetrini containers %				

2bis. Check the available containers

# Hands-on: build your RedHat based container

## 3. Install miniforge inside container



```
cpetrini@p-cpetrini containers % cat Dockerfile
FROM ...

...
...
RUN \
/bin/sh -c set -x \
&& UNAME_M="$(uname -m)" \
&& MINIFORGE_URL="https://github.com/conda-forge/miniforge/releases/download/${FORGE_VERSION}/Miniforge3-${FORGE_VERSION}-$(uname)-$uname.sh" \
&& wget "${MINIFORGE_URL}" -O miniforge.sh -q \
&& mkdir -p /opt \
&& bash miniforge.sh -b -p /opt/conda \
&& rm miniforge.sh \
&& ln -s /opt/conda/etc/profile.d/conda.sh /etc/profile.d/conda.sh \
&& echo ". /opt/conda/etc/profile.d/conda.sh" >> ~/.bashrc \
&& echo "conda activate base" >> ~/.bashrc \
&& find /opt/conda/ -follow -type f -name '*.a' -delete \
&& find /opt/conda/ -follow -type f -name '*.js.map' -delete \
&& /opt/conda/bin/conda clean -afy
```

# Hands-on: build your RedHat based container

3. Install miniforge inside container

```
cpetrini@p-cpetrini containers % cat Dockerfile
FROM ...

...
...
RUN \
/bin/sh -c set -x \
&& UNAME_M="$(uname -m)" \
&& MINIFORGE_URL="https://github.com/conda-forge/miniforge/releases/download/${FORGE_VERSION}/Miniforge3-${FORGE_VERSION}-$(uname)-$uname.sh" \
&& wget "${MINIFORGE_URL}" -O miniforge.sh -q \
&& mkdir -p /opt \
&& bash miniforge.sh -b -p /opt/conda \
&& rm miniforge.sh \
&& ln -s /opt/conda/etc/profile.d/conda.sh /etc/profile.d/conda.sh \
&& echo ". /opt/conda/etc/profile.d/conda.sh" >> ~/.bashrc \
&& echo "conda activate base" >> ~/.bashrc \
&& find /opt/conda/ -follow -type f -name '*.a' -delete \
&& find /opt/conda/ -follow -type f -name '*.js.map' -delete \
&& /opt/conda/bin/conda clean -afy
```

It is just a sophisticated version of

saw on conda section

```
curl -L -O "https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-$(uname)-$uname.sh"
bash Miniforge3-$(uname)-$uname.sh
```

# Hands-on: build your RedHat based container

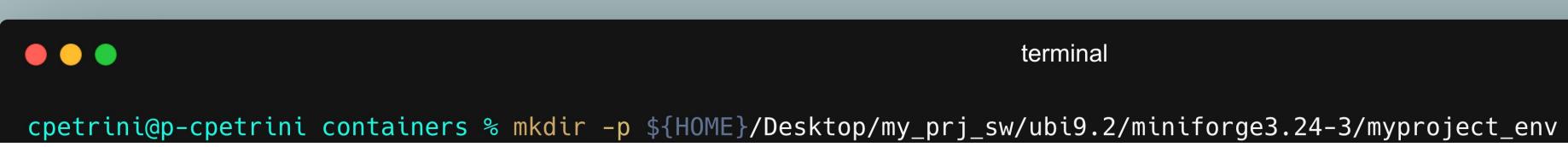


terminal

```
cpetrini@p-cpetrini containers % docker build -t mystep3 .
```

3. Install miniforge inside container

# Hands-on: create a conda environment on top of the container



terminal

```
cpetrini@p-cpetrini containers % mkdir -p ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env
```

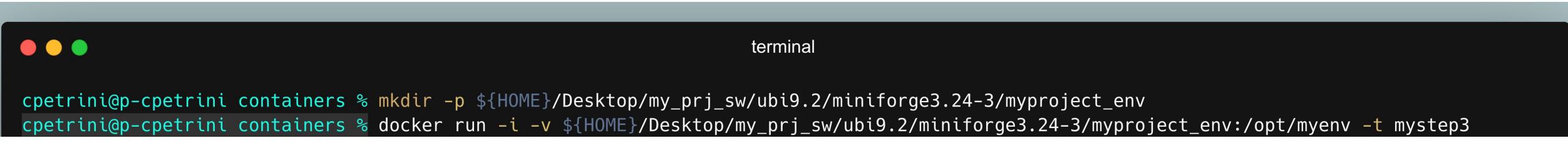
# Hands-on: create a conda environment on top of the container



terminal

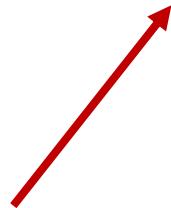
```
cpetrini@p-cpetrini containers % mkdir -p ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env  
cpetrini@p-cpetrini containers % docker run -i -v ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env:/opt/myenv -t mystep3
```

# Hands-on: create a conda environment on top of the container



A screenshot of a terminal window titled "terminal". The window has three colored window control buttons (red, yellow, green) at the top left. The terminal displays two commands in light blue font:

```
cpetrini@p-cpetrini containers % mkdir -p ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env  
cpetrini@p-cpetrini containers % docker run -i -v ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env:/opt/myenv -t mystep3
```



Mount a folder inside the container

# Hands-on: create a conda environment on top of the container

A screenshot of a terminal window titled "terminal". The terminal shows two commands:

```
cpetrini@p-cpetrini containers % mkdir -p ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env  
cpetrini@p-cpetrini containers % docker run -i -v ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env:/opt/myenv -t mystep3
```

The terminal has three red circular icons in the top-left corner.

Annotations explain the Docker command:

- A red arrow points from the text "Mount a folder inside the container" to the "-v" option in the command line.
- A bracket labeled "Folder we want to mount" covers the path \${HOME}/Desktop/my\_prj\_sw/ubi9.2/miniforge3.24-3/myproject\_env.
- A bracket labeled "position : where we want the folder inside the container" covers the path /opt/myenv.

Mount a folder inside the container

Folder we want to mount

position : where we want the folder inside the container

# Hands-on: create a conda environment on top of the container



terminal

```
cpetrini@p-cpetrini containers % mkdir -p ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env  
cpetrini@p-cpetrini containers % docker run -i -v ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env:/opt/myenv -t mystep3  
(base) [root@970e6bb7e8f3 /]# conda create -p /opt/myenv
```

# Hands-on: create a conda environment on top of the container



terminal

```
cpetrini@p-cpetrini containers % mkdir -p ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env  
cpetrini@p-cpetrini containers % docker run -i -v ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env:/opt/myenv -t mystep3  
(base) [root@970e6bb7e8f3 /]# conda create -p /opt/myenv  
(base) [root@970e6bb7e8f3 /]# conda activate /opt/myenv/
```

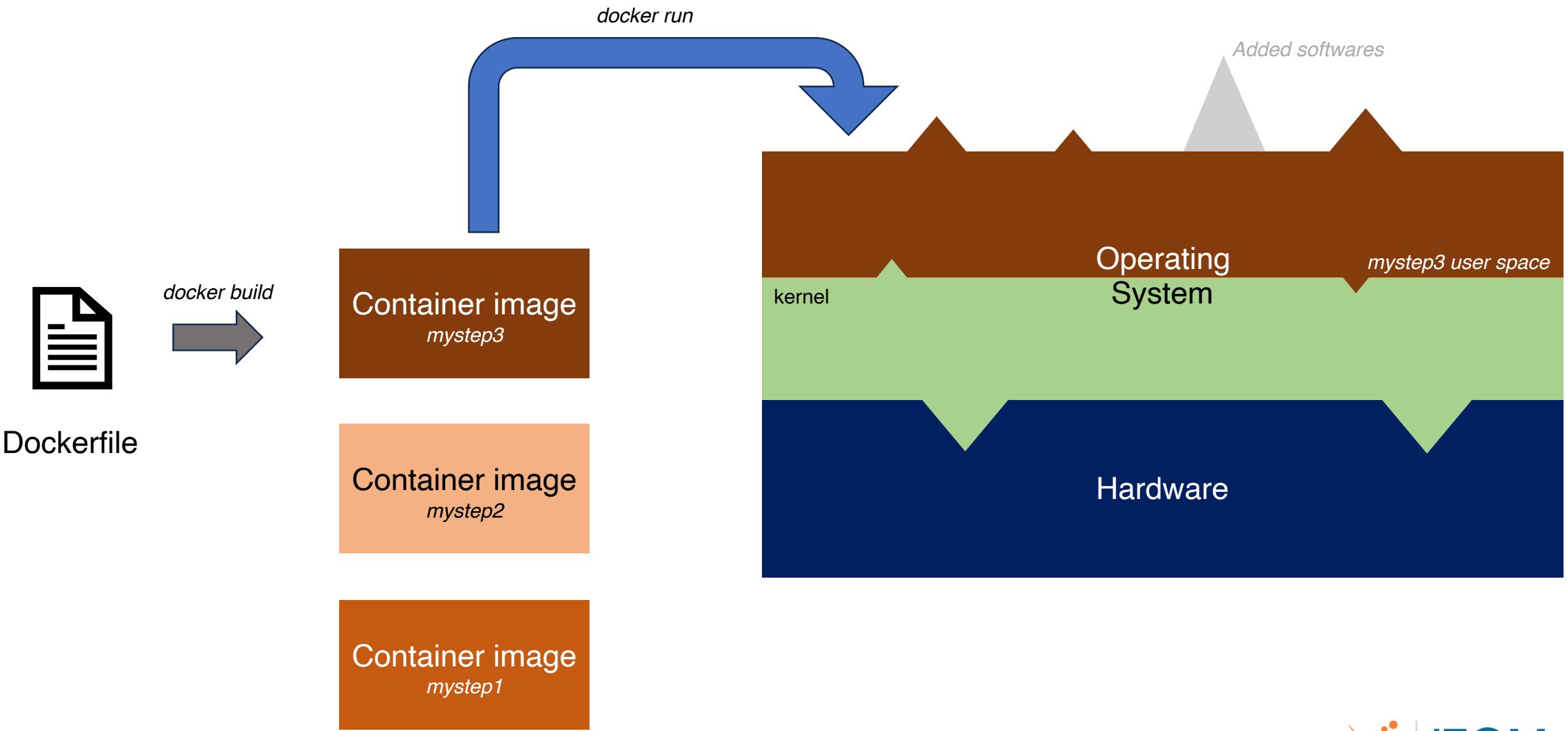
# Hands-on: create a conda environment on top of the container



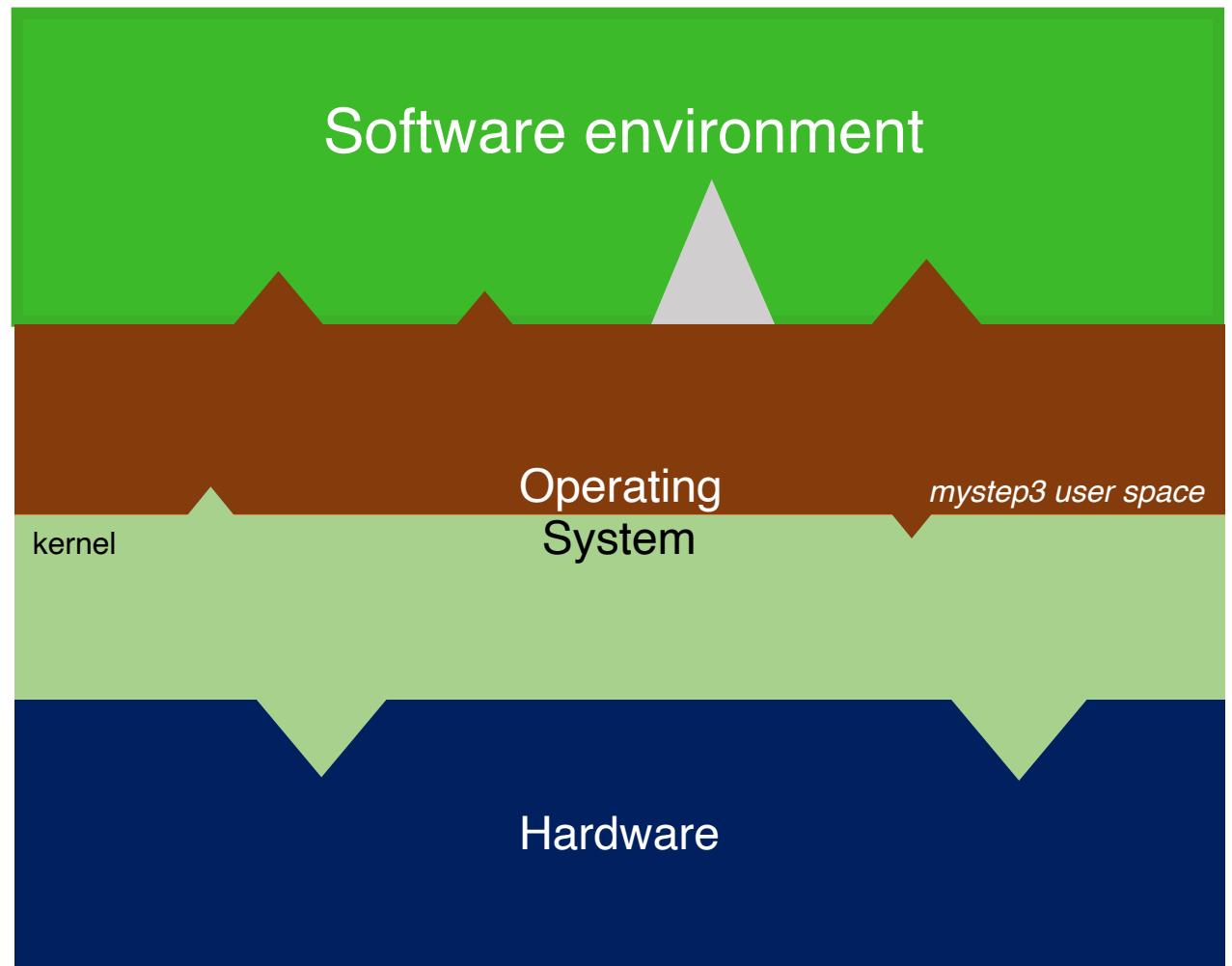
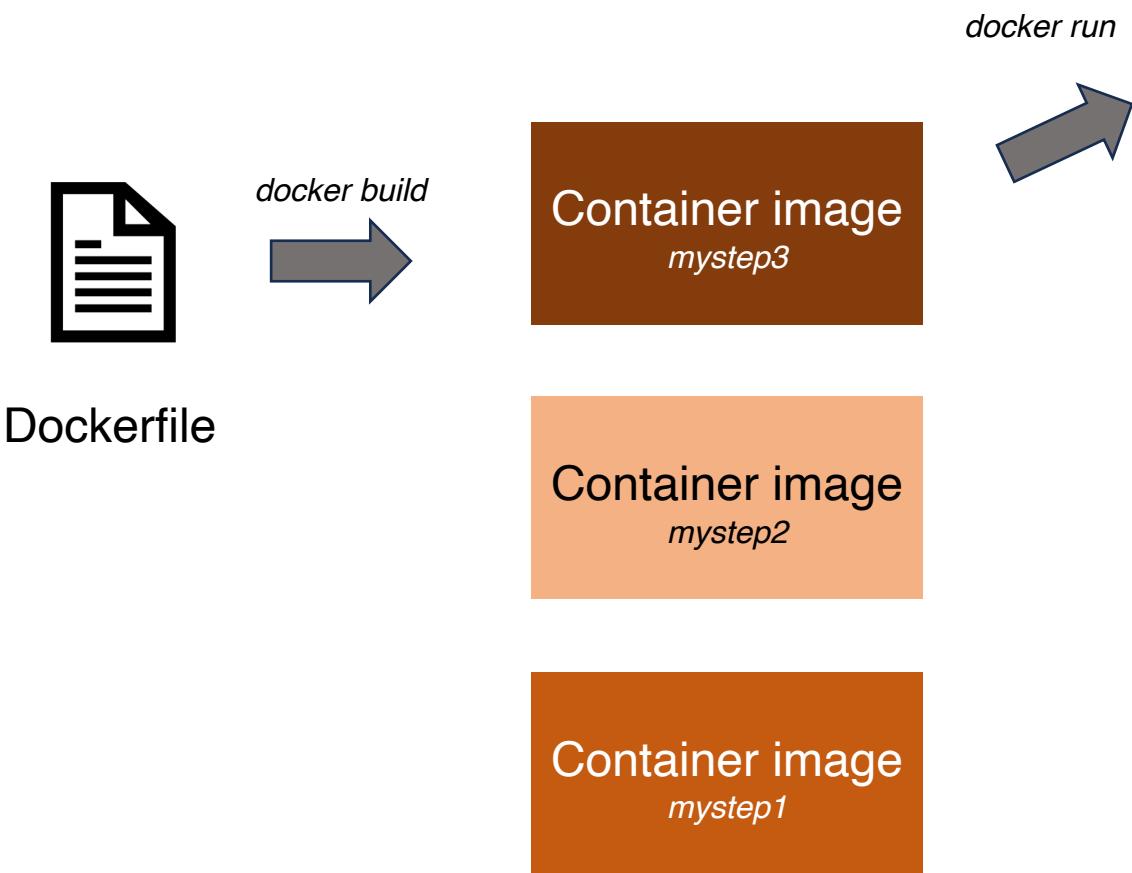
terminal

```
cpetrini@p-cpetrini containers % mkdir -p ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env
cpetrini@p-cpetrini containers % docker run -i -v ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env:/opt/myenv -t mystep3
(base) [root@970e6bb7e8f3 /]# conda create -p /opt/myenv
(base) [root@970e6bb7e8f3 /]# conda activate /opt/myenv/
(/opt/myenv) [root@970e6bb7e8f3 /]# conda install bioconda::samtools
```

# What did we?!



# What did we?!



# Hands-on: create a conda environment on top of the container



terminal

```
cpetrini@p-cpetrini containers % mkdir -p ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env
cpetrini@p-cpetrini containers % docker run -i -v ${HOME}/Desktop/my_prj_sw/ubi9.2/miniforge3.24-3/myproject_env:/opt/myenv -t mystep3
(base) [root@970e6bb7e8f3 /]# conda create -p /opt/myenv
(base) [root@970e6bb7e8f3 /]# conda activate /opt/myenv/
(/opt/myenv) [root@970e6bb7e8f3 /]# conda install bioconda::samtools
```

Do some tests yourself!

Recreate the environment built before:

- can you have access it from outside the container?
- does it works?
- what about JupyterLab?

*Bonus section!  
Best practices*

# Git & GitHub

Collaboration best practices

# What is git?

A Version Control System for Everyone: Git is a free, open-source version control system that **tracks changes in files**, allowing multiple people to collaborate on projects efficiently.

- It keeps a **history of your project**.
- **Safely** experiment with new ideas (branching).
- **Collaborate** with others without overwriting each other's work

*Real-Life Example: Think of it as "Track Changes" in Word but for code and complex projects.*

# Why Version Control is Essential?



Solve Collaboration and Tracking Issues (**i.e. files proliferation**)

It keeps a history of your project.

- Maintaining a history of changes.
- Enabling collaboration via branching and merging.
- Allowing **rollbacks** to any previous state.

# Git & GitHub

- Git is a piece of free software living on your computer, where you develop/write code



- GitHub (and others) are remote (web) repositories that collect your code and organize your work lines according to the rules of Git

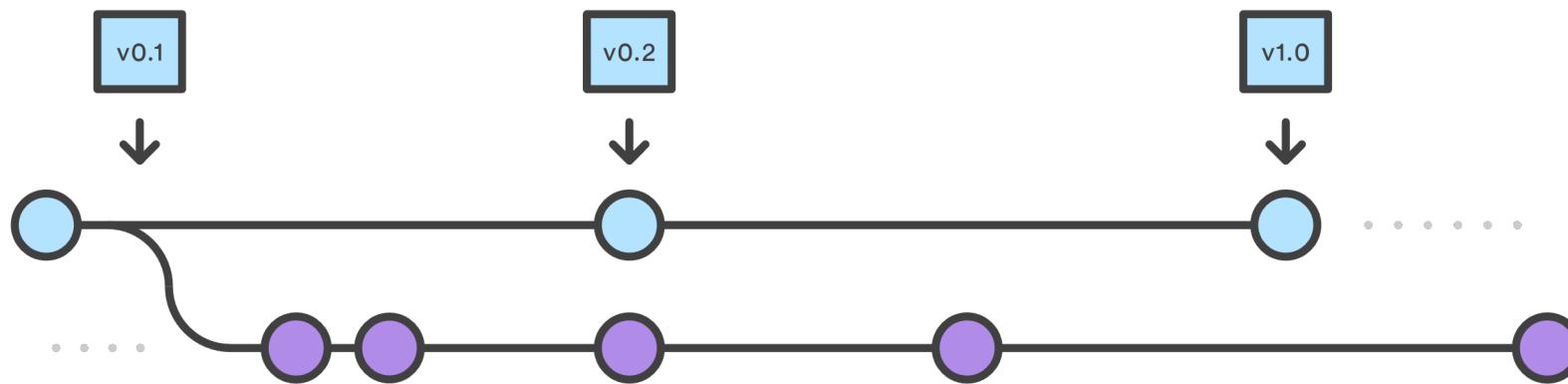


Bitbucket

# Branches: different worklines

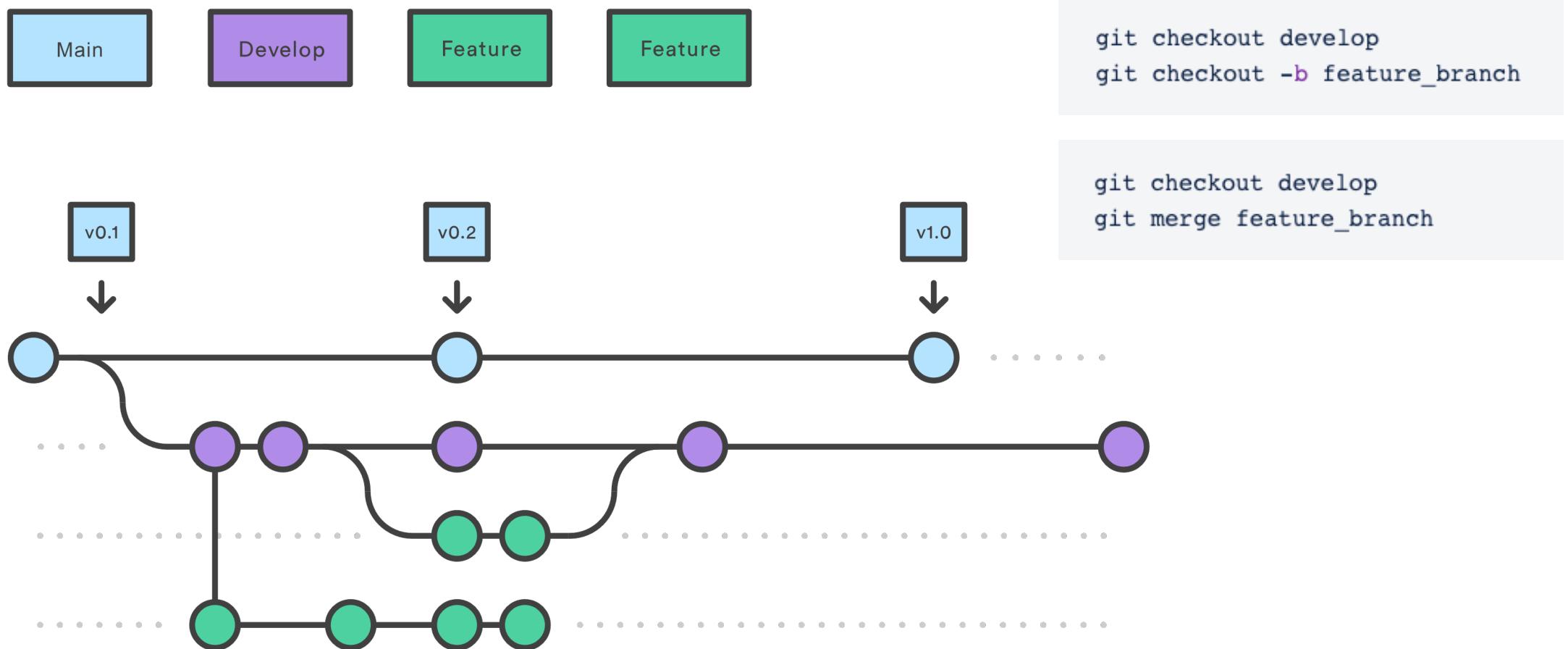


```
git branch develop  
git push -u origin develop
```

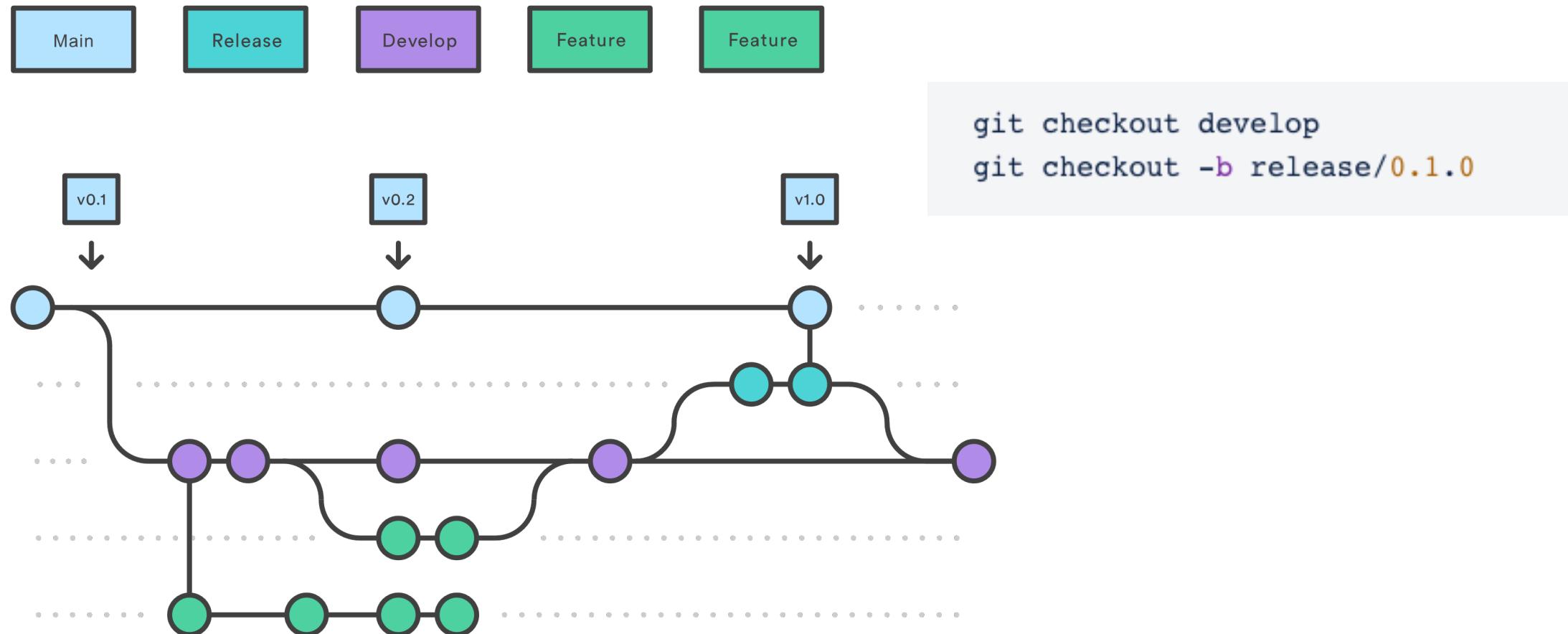


Credits: Atlassian, <https://www.atlassian.com/it/git/tutorials/comparing-workflows/>

# More features branches

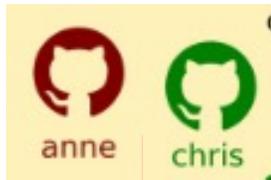


# Release

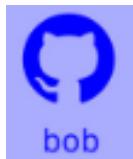


# The actors in a collaboration

Colleagues in a lab



A collaborator in another lab



All the others

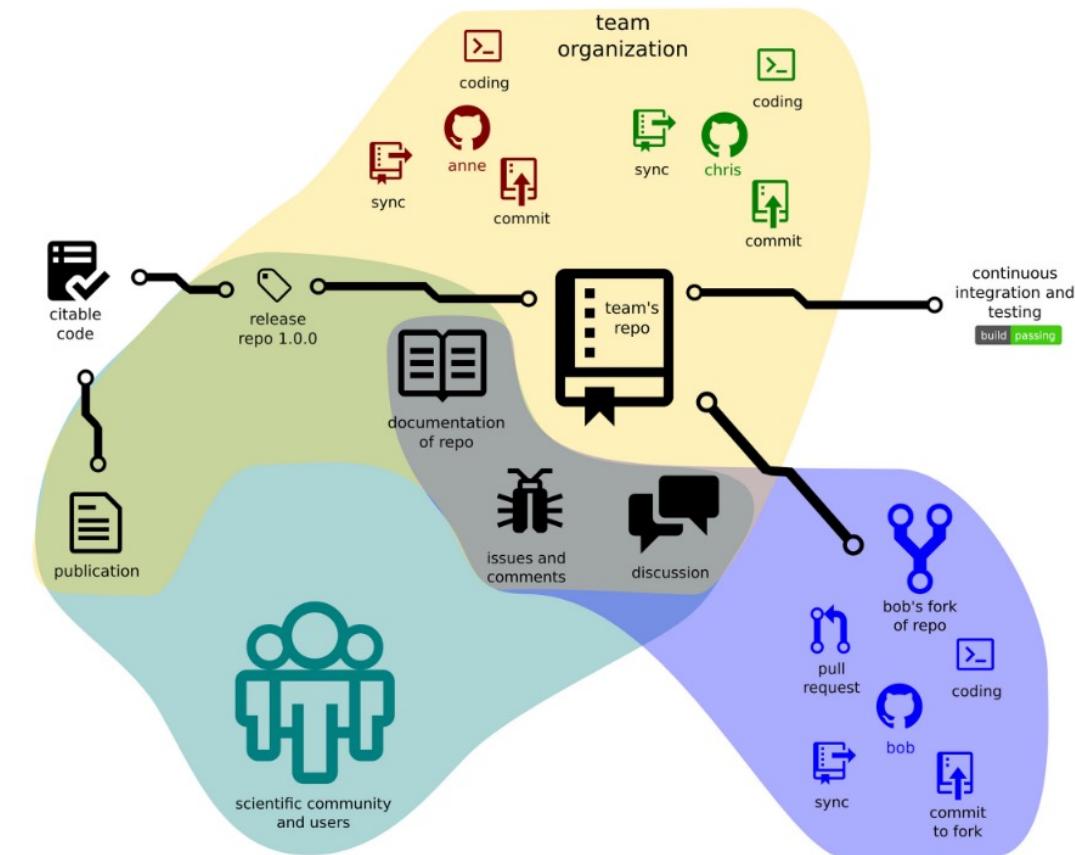


Fig 1. The structure of a GitHub-based project illustrating project structure and interactions with the community.

doi:10.1371/journal.pcbi.1004947.g001



COMPUTATIONAL  
BIOLOGY

---

EDITORIAL

## Ten Simple Rules for Taking Advantage of Git and GitHub

Yasset Perez-Riverol<sup>1\*</sup>, Laurent Gatto<sup>2</sup>, Rui Wang<sup>1</sup>, Timo Sachsenberg<sup>3</sup>, Julian Uszkoreit<sup>4</sup>, Felipe da Veiga Leprevost<sup>5</sup>, Christian Fufezan<sup>6</sup>, Tobias Ternent<sup>1</sup>, Stephen J. Eglen<sup>7</sup>, Daniel S. Katz<sup>8</sup>, Tom J. Pollard<sup>9</sup>, Alexander Konovalov<sup>10</sup>, Robert M. Flight<sup>11</sup>, Kai Blin<sup>12</sup>, Juan Antonio Vizcaíno<sup>1\*</sup>

## EDITORIAL

## Ten Simple Rules for Taking Advantage of Git and GitHub

Yasset Perez-Riverol<sup>1,\*</sup>, Laurent Gatto<sup>2</sup>, Rui Wang<sup>1</sup>, Timo Sachsenberg<sup>3</sup>, Julian Usakoreit<sup>4</sup>, Felipe da Veiga Leprevost<sup>5</sup>, Christian Fuzeau<sup>6</sup>, Tobias Ternent<sup>1</sup>, Stephen J. Eglen<sup>7</sup>, Daniel S. Katz<sup>8</sup>, Tom J. Pollard<sup>9</sup>, Alexander Konovalov<sup>10</sup>, Robert M. Flight<sup>11</sup>, Kai Blin<sup>12</sup>, Juan Antonio Vizcaíno<sup>1\*</sup>

# Ten simple rules

- Rule 1: Use GitHub to Track Your Projects
- Rule 2: GitHub for Single Users, Teams, and Organizations
- Rule 3: Developing and Collaborating on New Features: Branching and Forking
- Rule 4: Naming Branches and Commits: Tags and Semantic Versions
- Rule 5: Let GitHub Do Some Tasks for You: Integrate
- Rule 6: Let GitHub Do More Tasks for You: Automate
- Rule 7: Use GitHub to Openly and Collaboratively Discuss, Address, and Close Issues
- Rule 8: Make Your Code Easily Citable, and Cite Source Code!
- Rule 9: Promote and Discuss Your Projects: Web Page and More
- Rule 10: Use GitHub to Be Social: Follow and Watch

# Ten simple rules

Yasset Perez-Riverol<sup>1,\*</sup>, Laurent Gatto<sup>2</sup>, Rui Wang<sup>1</sup>, Timo Sachsenberg<sup>3</sup>, Julian Usakoreit<sup>4</sup>, Felipe da Veiga Leprevost<sup>5</sup>, Christian Fuzeau<sup>6</sup>, Tobias Ternent<sup>1</sup>, Stephen J. Eglen<sup>7</sup>, Daniel S. Katz<sup>8</sup>, Tom J. Pollard<sup>9</sup>, Alexander Konovalov<sup>10</sup>, Robert M. Flight<sup>11</sup>, Kai Blin<sup>12</sup>, Juan Antonio Vizcaíno<sup>1\*</sup>

- **Rule 1: Use GitHub to Track Your Projects**
- **Rule 2: GitHub for Single Users, Teams, and Organizations**
- **Rule 3: Developing and Collaborating on New Features: Branching and Forking**
- **Rule 4: Naming Branches and Commits: Tags and Semantic Versions**
- Rule 5: Let GitHub Do Some Tasks for You: Integrate
- Rule 6: Let GitHub Do More Tasks for You: Automate
- **Rule 7: Use GitHub to Openly and Collaboratively Discuss, Address, and Close Issues**
- Rule 8: Make Your Code Easily Citable, and Cite Source Code!
- Rule 9: Promote and Discuss Your Projects: Web Page and More
- Rule 10: Use GitHub to Be Social: Follow and Watch

**“Acta est fabula plaudite”**  
*Svetonio*

(The play has been performed; applaud!)

# Credits & References

- *Code picture created using:* <https://carbon.now.sh/>
- *Conda description from:* <https://anaconda.org/anaconda/conda>
- *Miniforge3 installing code from:* <https://github.com/conda-forge/miniforge>
- *Jupyter official documentation:* <https://jupyterlab.readthedocs.io/en/latest/#>  
and <https://jupyter-notebook.readthedocs.io/en/latest/>
- *Apptainer official documentation:*  
<https://apptainer.org/docs/user/latest/introduction.html>