

Reproducible Scientific Computing and Data Analysis

Nadia Marounina, Henry Lütcke
Scientific IT Services, ETH Zurich

March 11, 2026

Slides & Materials: https://siscourses.ethz.ch/reproducible_computing/



Overview of today's workshop



Setting the Scene



Managing your Source Code



Managing Dependencies & Computing Environments



Virtualizing Computing Environments



Interactive Computational Notebooks

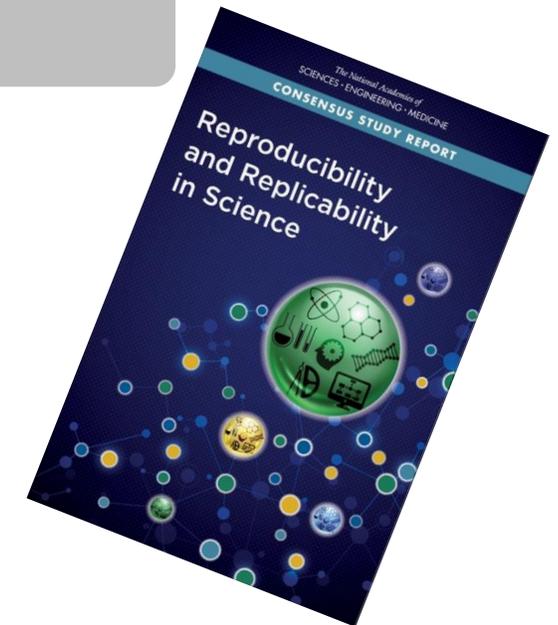
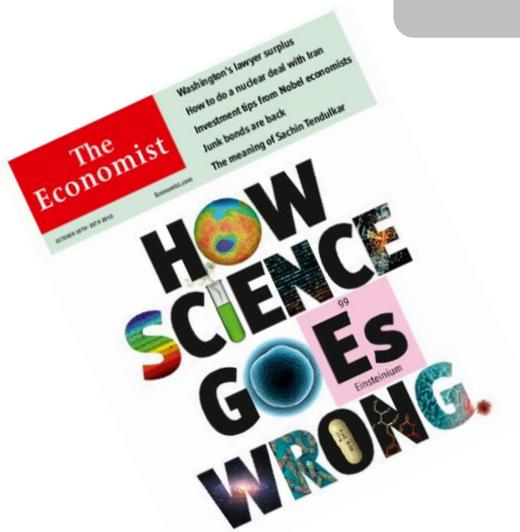


Reproducible Computing Platforms



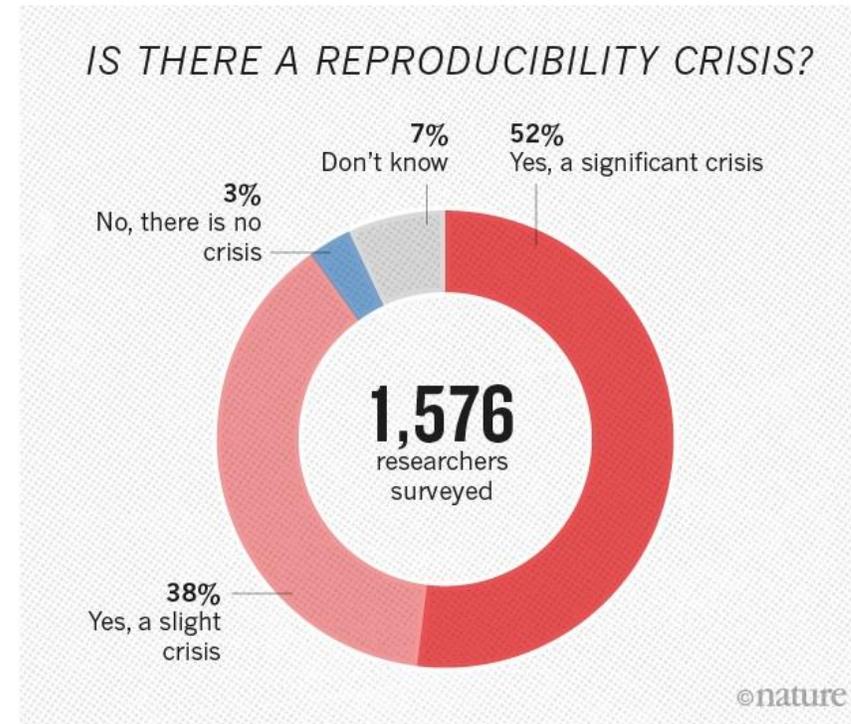
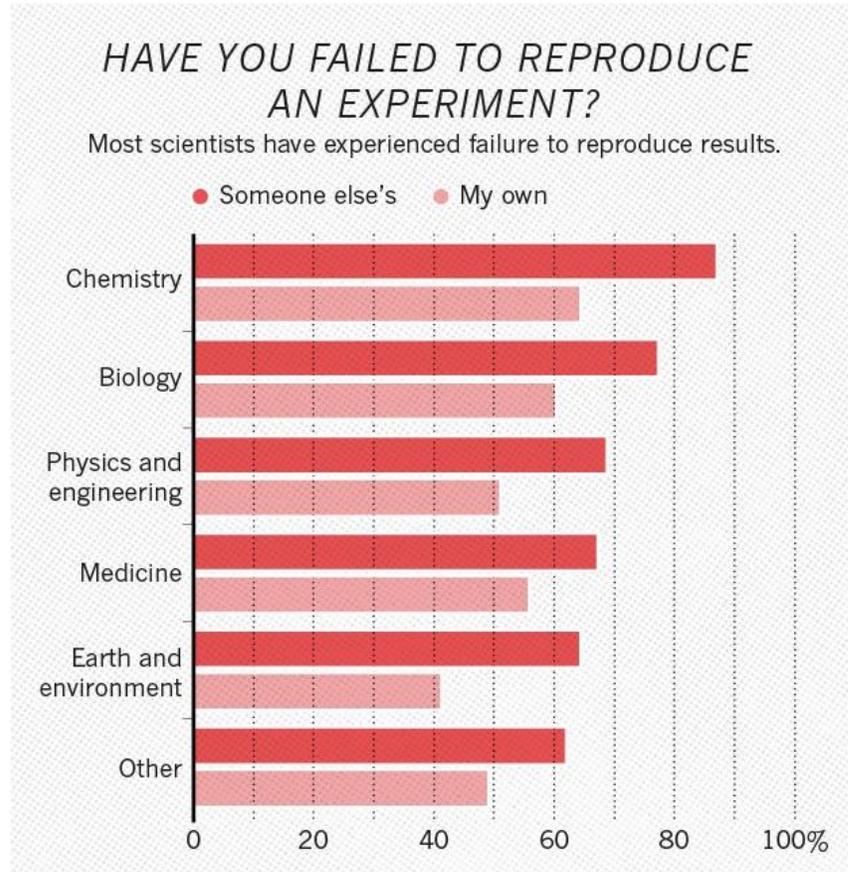


Setting the Scene



Reproducibility & Replicability in Science

Nature survey on reproducibility across all scientific domains



[Nature 533, 452–454 \(26 May 2016\) doi:10.1038/533452a](https://doi.org/10.1038/533452a)

Reproducibility & Replicability in Science

RESEARCH ARTICLE

Estimating the reproducibility of psychological science

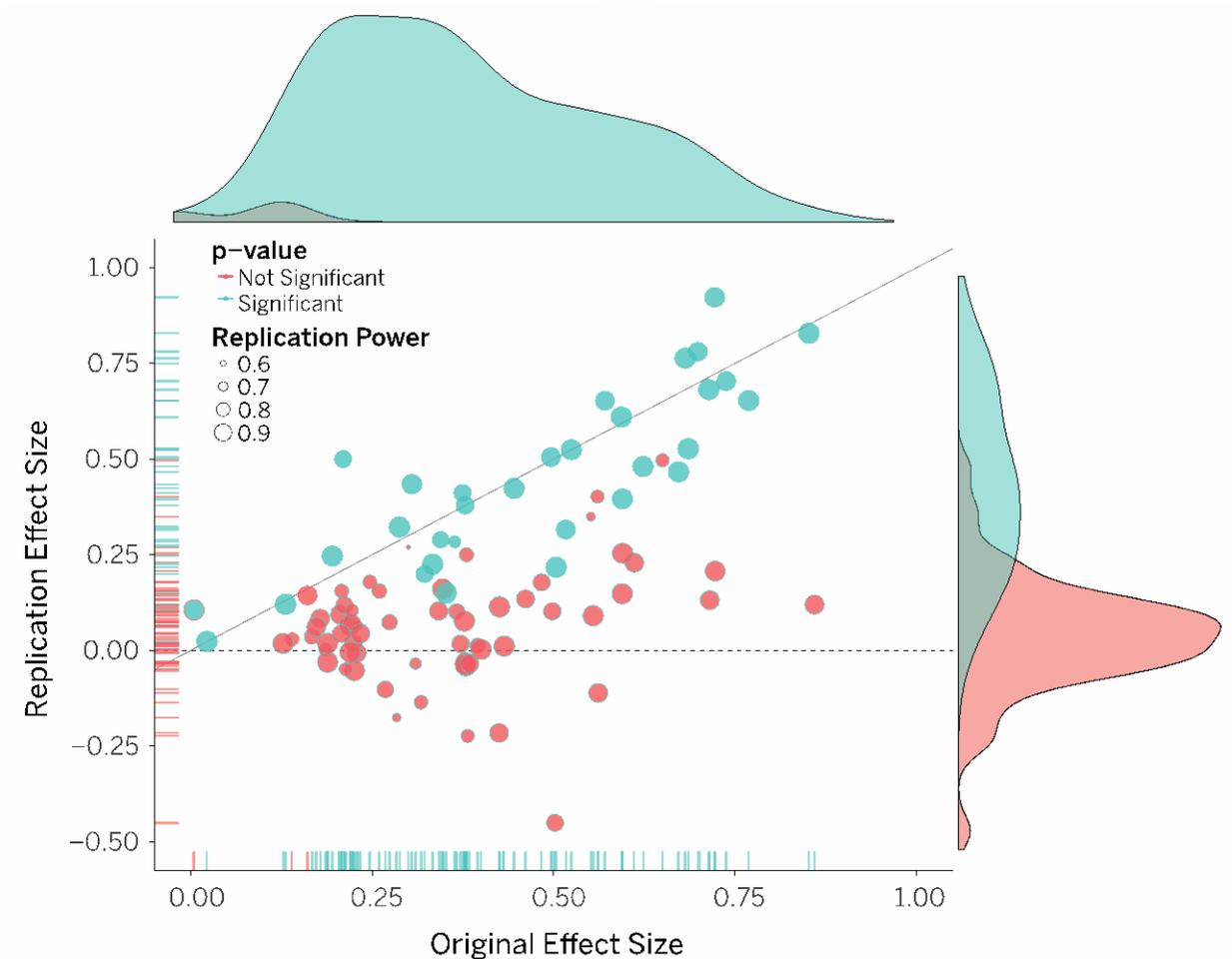
Open Science Collaboration^{*,†}

⁺ See all authors and affiliations

Science 28 Aug 2015:
Vol. 349, Issue 6251, aac4716
DOI: 10.1126/science.aac4716

The *Reproducibility project*

- Replicate 100 experiments published in top psychology journals
- One-half to two-thirds of original findings could not be observed in the replication study



Reproducibility & Replicability in Science

RESEARCH ARTICLE

Estimating the reproducibility of psychological science

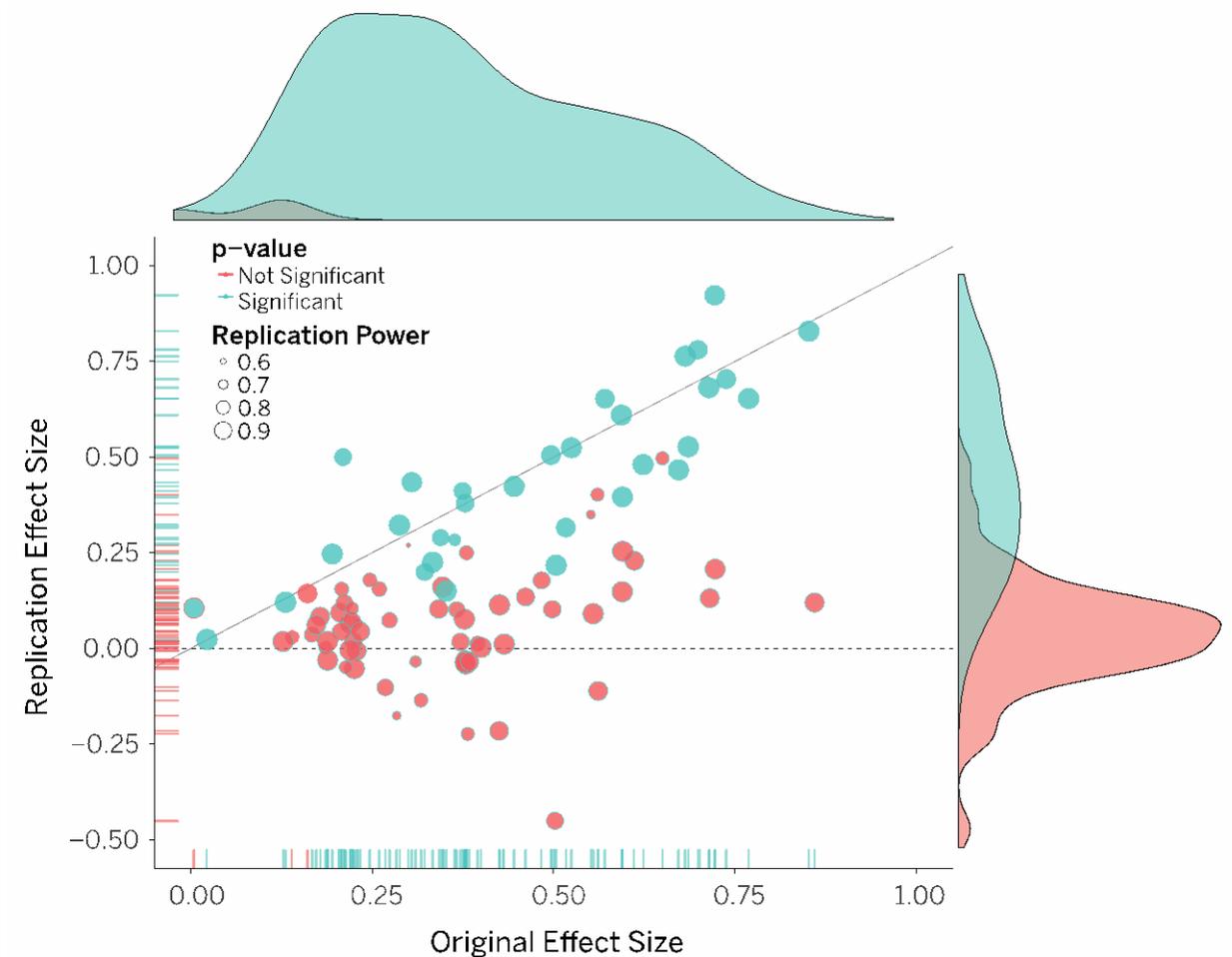
Open Science Collaboration^{*†}

⁺ See all authors and affiliations

Science 28 Aug 2015:
Vol. 349, Issue 6251, aac4716
DOI: 10.1126/science.aac4716

The **Reproducibility project**

- **Replicate 100 experiments** published in top psychology journals
- One-half to two-thirds of original findings could not be observed in the **replication study**



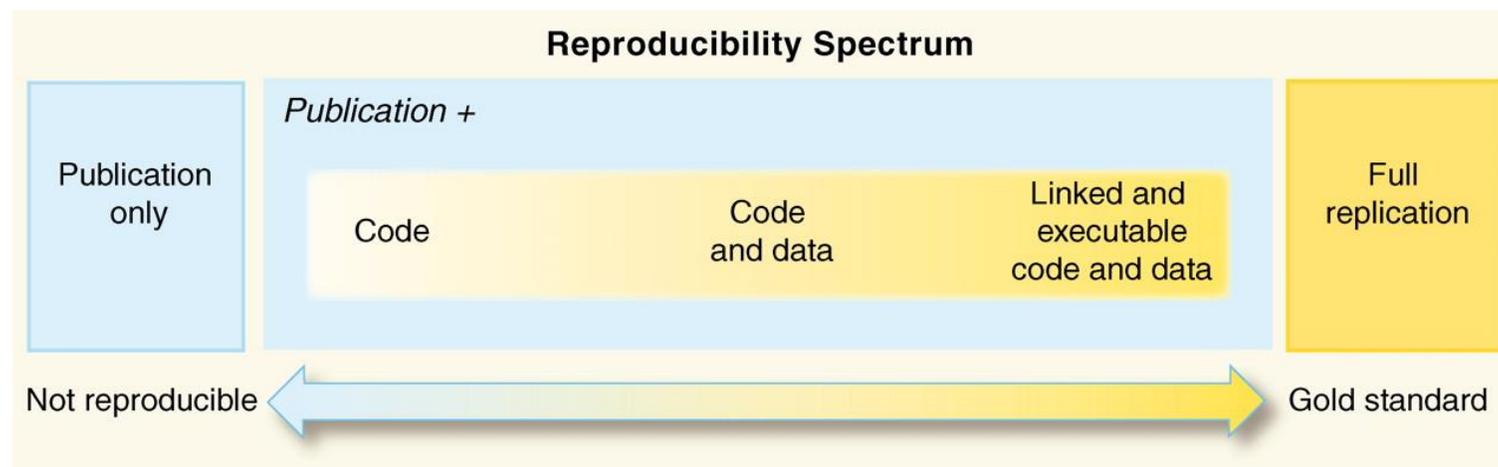
Reproducibility & Replicability in Science

Replication:

new data and / or new method in independent study = same finding

Reproducible research:

same data + same method = same results



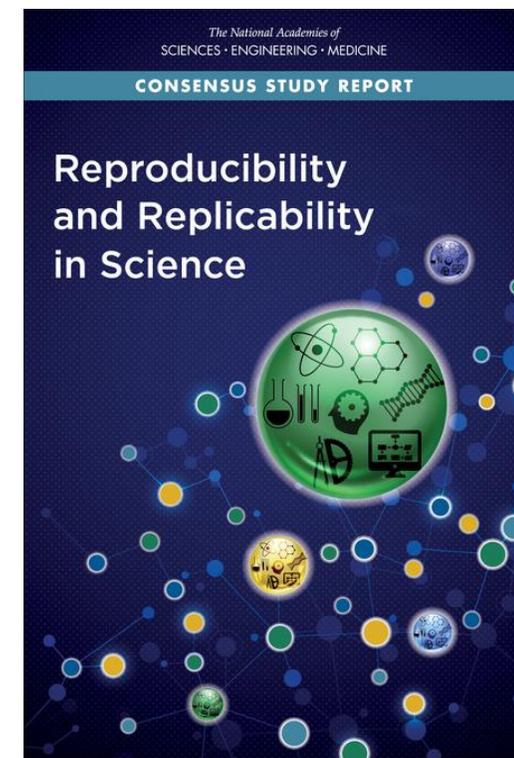
Peng (2011). [doi:10.1126/science.1213847](https://doi.org/10.1126/science.1213847)

Defining the Scope: Computational Reproducibility

«**Reproducibility** is obtaining consistent results using the same input data, computational steps, methods, and code and conditions of analysis. The term is synonymous with "computational reproducibility"... »

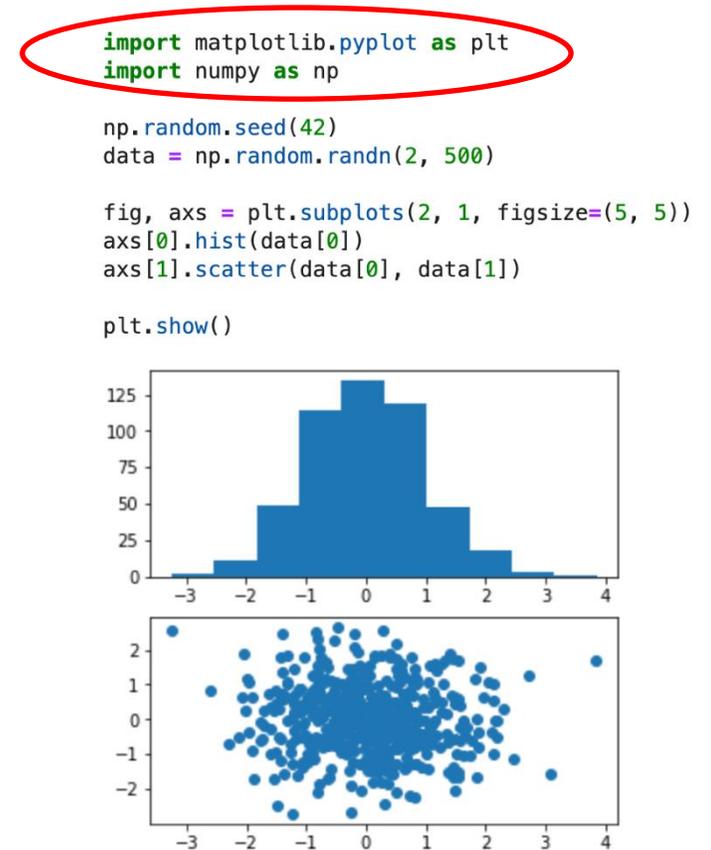
«To help ensure the reproducibility of computational results, researchers should **convey clear, specific, and complete information about any computational methods and data products that support their published results in order to enable other researchers to repeat the analysis**, unless such information is restricted by non-public data policies. That information should include the data, study methods, and **computational environment**. »

National Academies of Sciences, Engineering, and Medicine (2019). <https://doi.org/10.17226/25303>



Computational Reproducibility: What can go wrong?

- Code only runs on specific **operating system**
 - Examples: Windows / Linux scripts, special programs (e.g. *SigmaPlot*)
- Code has specific **external dependencies**
 - Example: wget <https://zenodo.org/record/1234567/files/dataset.zip>
- Code has specific **internal dependencies** (libraries, modules etc.)



Computational Reproducibility: What can go wrong?

- Code only runs on specific **operating system**
 - Examples: Windows / Linux scripts, special programs (e.g. *SigmaPlot*)
- Code has specific **external dependencies**
 - Example: wget <https://zenodo.org/record/1234567/files/dataset.zip>
- Code has specific **internal dependencies** (libraries, modules etc.)
- Code has specific **version dependencies**
- Code may rely on availability of specific **software licenses**
 - Example: fastaread function in the MATLAB Bioinformatics Toolbox

```
import numpy as np

print("Using Numpy %s" % np.__version__)

rng = np.random.default_rng(42)
rng.dirichlet((0.04, 0.03), 2)

Using Numpy 1.18.1
array([[2.10122596e-01, 7.89877404e-01],
       [1.99456813e-22, 1.00000000e+00]])
```

```
import numpy as np

print("Using Numpy %s" % np.__version__)

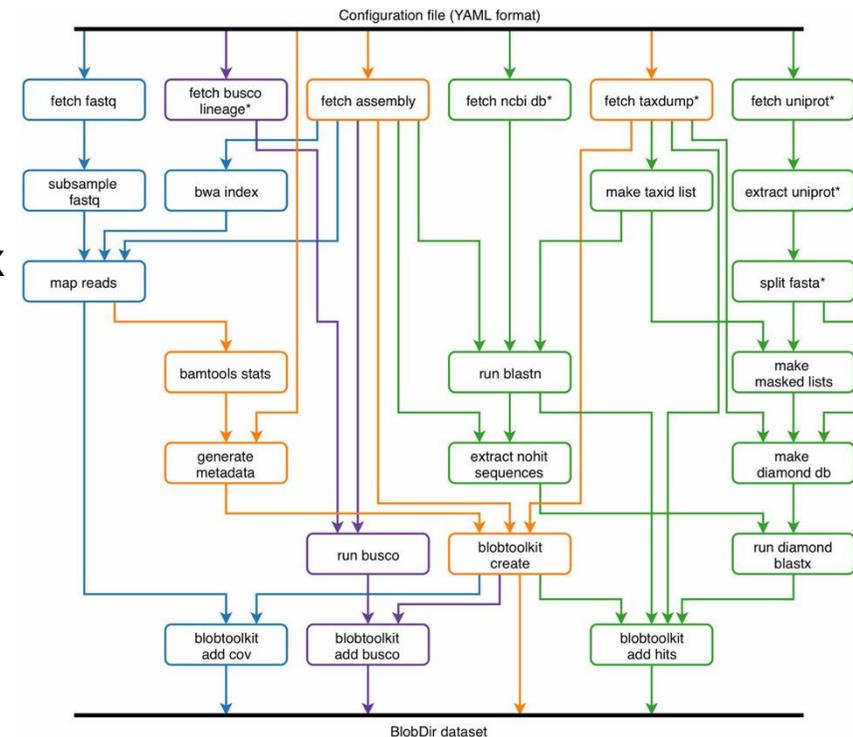
rng = np.random.default_rng(42)
rng.dirichlet((0.04, 0.03), 2)

Using Numpy 1.20.2
array([[9.99999999e-01, 7.24826532e-10],
       [9.99726345e-01, 2.73654825e-04]])
```

See <https://numpy.org/doc/stable/release/1.19.0-notes.html#changed-random-variate-stream-from-numpy-random-generator-dirichlet>

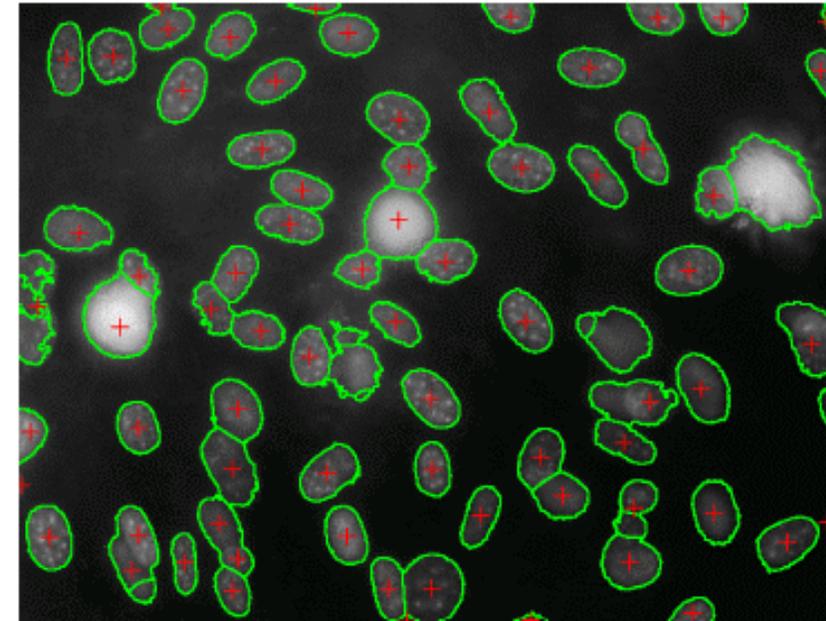
Computational Reproducibility: What can go wrong?

- Code only runs on specific **operating system**
 - Examples: Windows / Linux scripts, special programs (e.g. *SigmaPlot*)
- Code has specific **external dependencies**
 - Example: wget <https://zenodo.org/record/1234567/files/dataset.zip>
- Code has specific **internal dependencies** (libraries, modules etc.)
- Code has specific **version dependencies**
- Code may rely on availability of specific **software licenses**
- Code may be **incomprehensible** (complex, undocumented workflows)



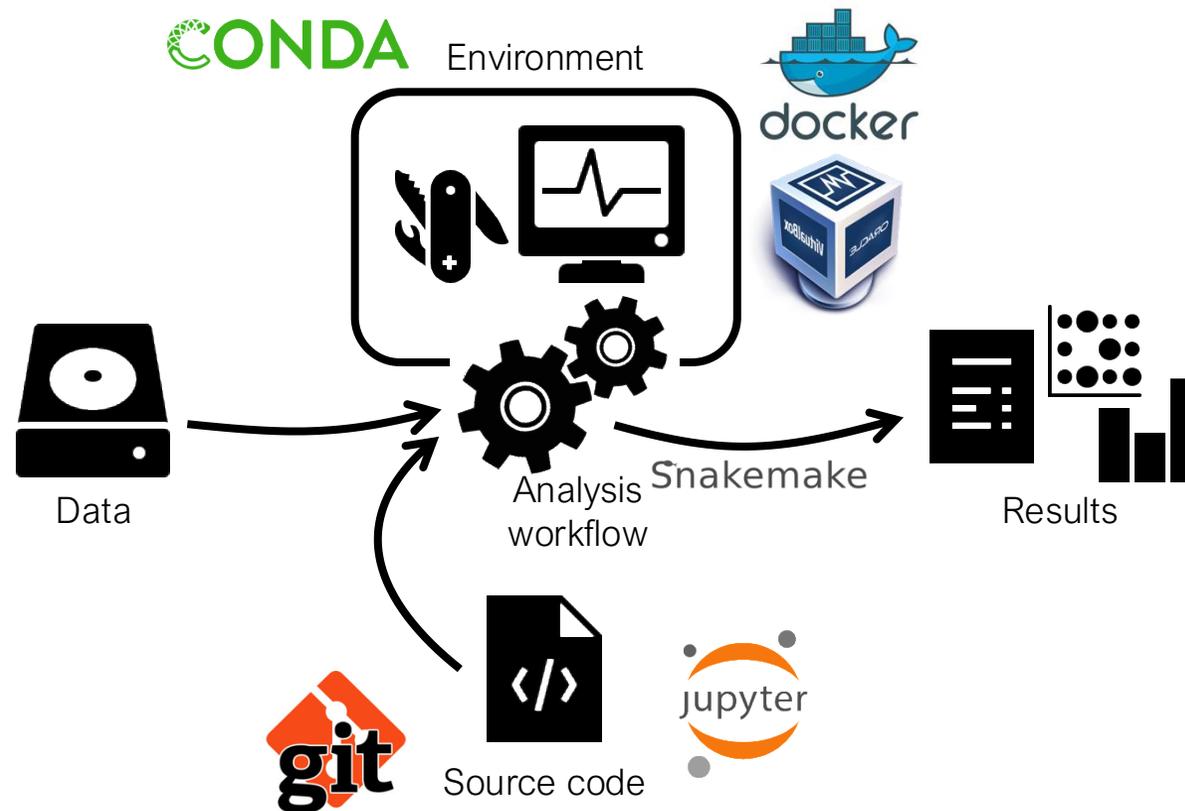
Computational Reproducibility: What can go wrong?

- Code only runs on specific **operating system**
 - Examples: Windows / Linux scripts, special programs (e.g. *SigmaPlot*)
- Code has specific **external dependencies**
 - Example: wget <https://zenodo.org/record/1234567/files/dataset.zip>
- Code has specific **internal dependencies** (libraries, modules etc.)
- Code has specific **version dependencies**
- Code may rely on availability of specific **software licenses**
 - Example: fastaread function in the MATLAB Bioinformatics Toolbox
- Code may be **incomprehensible** (complex, undocumented workflows)
- Analysis workflow may rely on **manual steps**



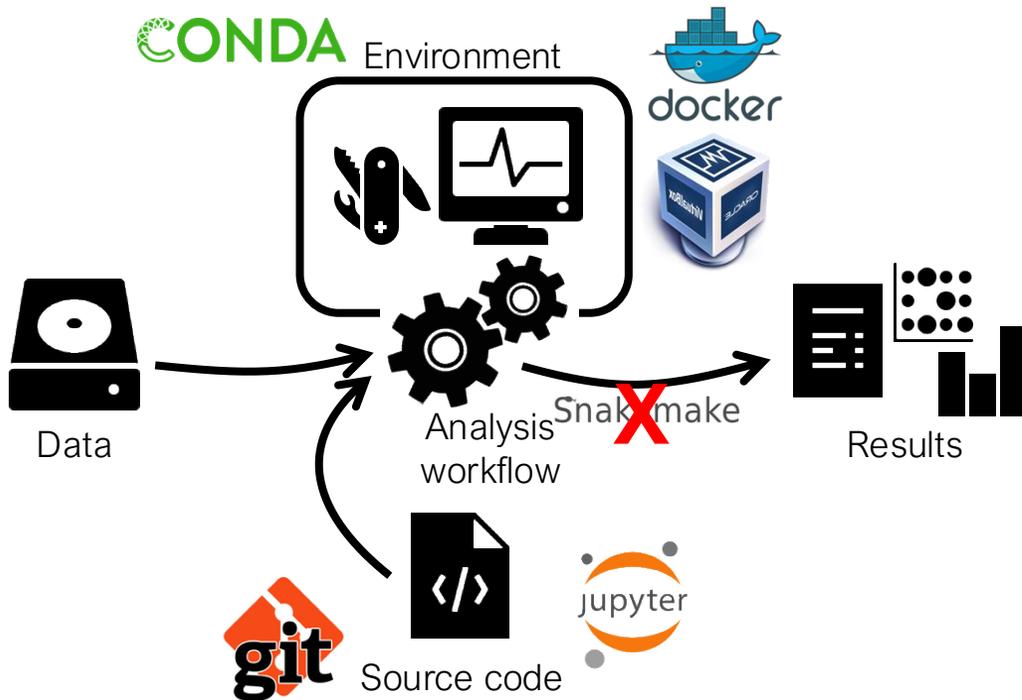
Computational Reproducibility: Pieces of the Puzzle

All parts of a computational analysis have to be reproducible!



Computational Reproducibility: Pieces of the Puzzle

What is covered in today's workshop?



Part 1 **Version control**
Track and backup your project history



Part 2 **Environment management**
Setup and manage your project environment



Part 3 **Virtual machines / containers**
Make your project self-contained and distributable



Part 4 **Notebooks**
Document your exploratory analysis



Part 5 **Reproducible computing platforms**
Out-of-the-box computational reproducibility



Computational Reproducibility: Questions?



Tell us a bit about yourself



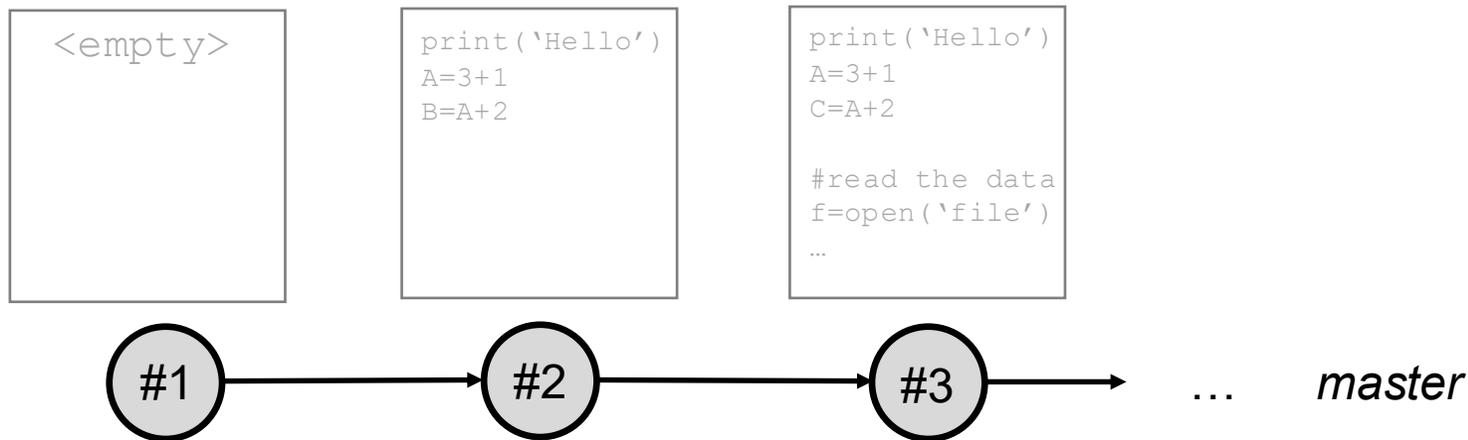
Managing your Source Code



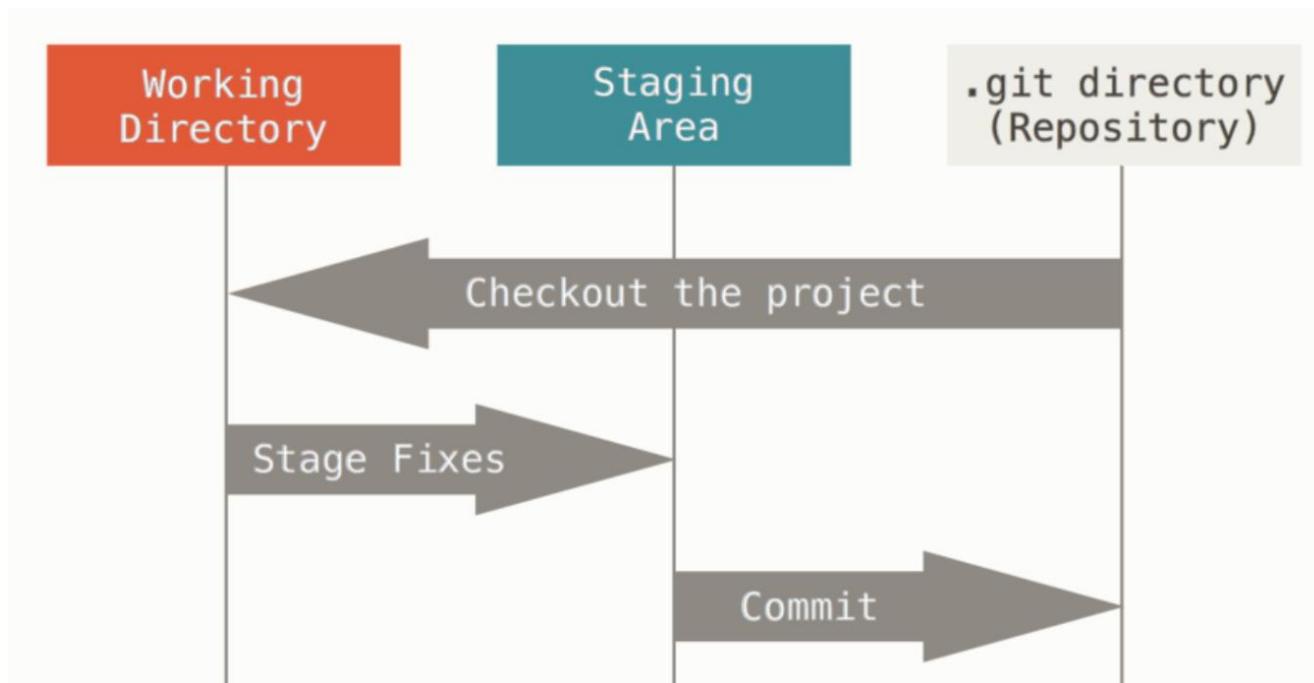
Code Management



- Code management is the process of handling changes in source code
- Proper code management is essential to ensure **reproducible results**
- Professional code management relies on **Version Control Systems (VCS)**
 - Version control: tracking changes made to text files over time
- **Git** is by far the most popular version control system used world-wide in the software community



How do I track the changes in my code with git?



The basic Git workflow

- Modify files in your working directory
- Selectively stage the changes you want to be part of your next commit, adding **only** those changes to the staging area
- Make a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your .git directory

[demo]

Test case : a program that takes in three files and print their content.
Text_1.txt contains the string “one”, text_2.txt “two”, etc

```
git_demo 13:58:33 >>ls
```

```
total 32
```

```
-rw-r-xr-x  1 nmarounina  staff  49 Mar  7 13:57 print_all.sh
```

```
-rw-r--r--  1 nmarounina  staff   4 Mar  7 13:54 text_1.txt
```

```
-rw-r--r--  1 nmarounina  staff   4 Mar  7 13:54 text_2.txt
```

```
-rw-r--r--  1 nmarounina  staff   6 Mar  7 13:54 text_3.txt
```

```
git_demo 13:59:00 >>./print_all.sh
```

```
one
```

```
two
```

```
three
```

```
git_demo 13:59:02 >>
```

Start with git :

```
git_demo 13:59:20 >>git init #initialises git
```

```
Initialized empty Git repository in /Users/nmarounina/Desktop/git_demo/.git/
```

```
git_demo 13:59:24 >>
```

```
git_demo 13:59:34 >>git add * #adds all files to the staging
```

```
git_demo 13:59:40 >>git status #prints information about the current staging area
```

```
On branch main
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file:   print_all.sh
```

```
new file:   text_1.txt
```

```
new file:   text_2.txt
```

```
new file:   text_3.txt
```

```
git_demo 13:59:50 >>
```

First commit :

```
git_demo 13:59:52 >>git commit -m "Initial commit" #creating the first commit/snapshot
```

```
[main (root-commit) d5badf3] Initial commit
```

```
4 files changed, 5 insertions(+)
```

```
create mode 100755 print_all.sh
```

```
create mode 100644 text_1.txt
```

```
create mode 100644 text_2.txt
```

```
create mode 100644 text_3.txt
```

```
git_demo 14:00:16 >>git log #lists all of the commits for this project
```

```
commit d5badf3593de0e511005eee061132d77cdde0823 (HEAD -> main)
```

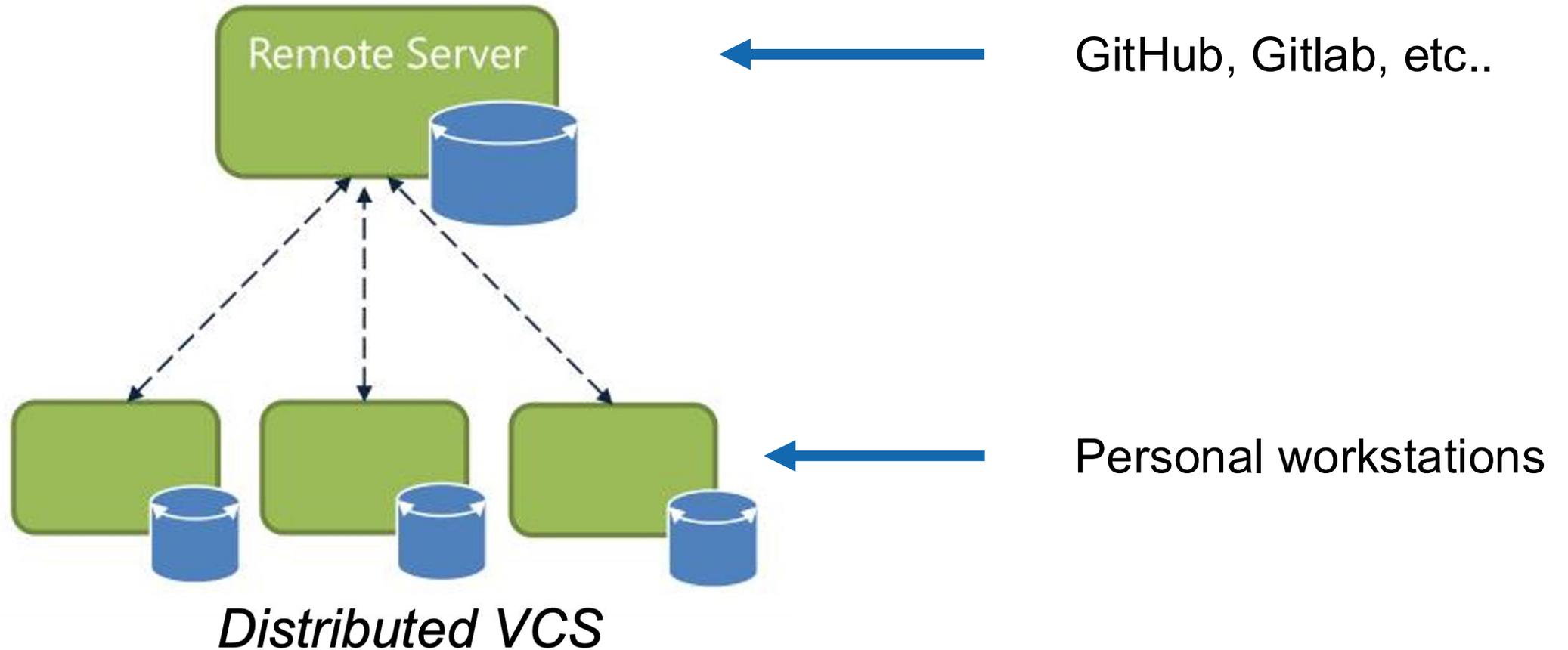
```
Author: Nadia Marounina <nmarounina@ethz.ch>
```

```
Date: Thu Mar 7 14:00:10 2024 +0100
```

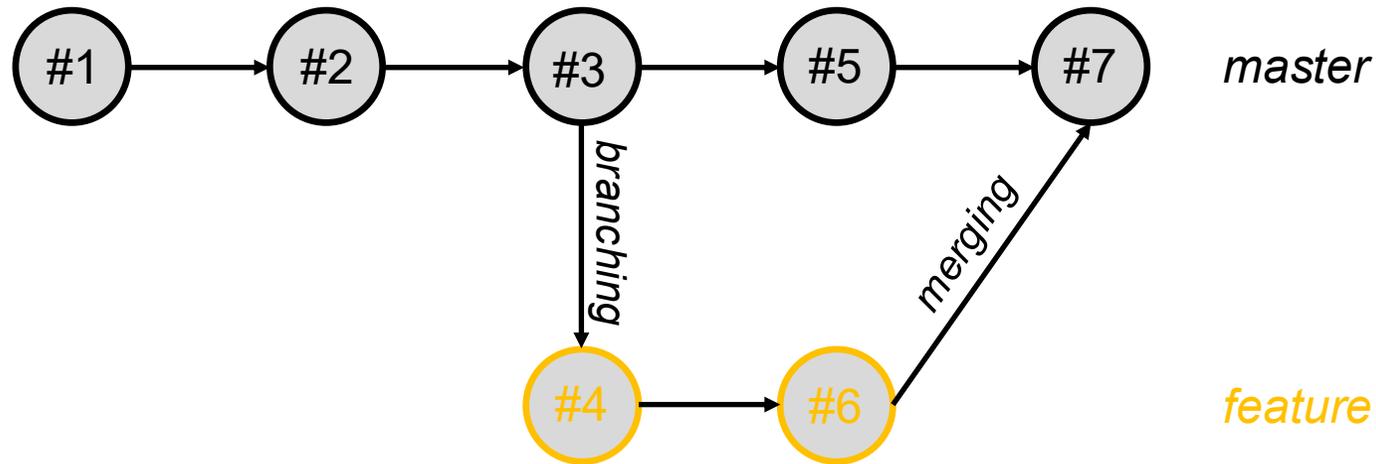
```
Initial commit
```

```
git_demo 14:00:20 >>
```

Git : How to share my code with others ?



Git branching & merging



Git branches & merges

- The initial / default branch is typically called *master* or *main*
- Git manages branches very efficiently
- When merging merging branches, conflicts must be resolved carefully

[demo]

Creating a new branch:

```
git_demo 14:03:15 >>git branch numbers #creates a new branch named "numbers"
```

```
git_demo 14:04:00 >>git status
```

```
On branch main
```

```
nothing to commit, working tree clean
```

```
git_demo 14:04:03 >>git branch #list all branches for the project
```

```
* main
```

```
numbers
```

```
git_demo 14:04:35 >>git checkout numbers #switch to the new branch
```

```
Switched to branch 'numbers'
```

```
git_demo 14:04:53 >>
```

After changing the three text files in the new branch and committing it again :

```
git_demo 14:04:56 >>vi text_1.txt #vi is a text editor. Here I change 'one' to '1'...
git_demo 14:05:07 >>vi text_2.txt #... 'two' to '2'
git_demo 14:05:16 >>vi text_3.txt #... 'three' to '3'
git_demo 14:05:29 >>./print_all.sh
1
2
3
git_demo 14:05:37 >>git commit -m "Changed from text to number" #the change has been
committed

[... output excluded ...]
git_demo 14:05:51 >>
```

By switching branches, you change your files in your folder:

```
git_demo 14:06:39 >>git checkout main
```

```
Switched to branch 'main'
```

```
git_demo 14:07:29 >>./print_all.sh
```

```
one
```

```
two
```

```
three
```

```
git_demo 14:07:40 >>git checkout numbers
```

```
Switched to branch 'numbers'
```

```
git_demo 14:07:45 >>./print_all.sh
```

```
1
```

```
2
```

```
3
```

```
git_demo 14:07:46 >>
```

ETH Zurich GitLab Service



The screenshot shows the GitLab web interface for a project named 'experimental-project-1'. The browser address bar shows the URL <https://gitlab.ethz.ch/sis-rdm-training/experimental-project-1>. The page header includes the ETH zürich logo and navigation tabs for Projects, Groups, Activity, Milestones, and Snippets. A notification banner at the top states: "To receive notifications about scheduled maintenance, please subscribe to the mailing-list gitlab-operations@sympa.ethz.ch. You can subscribe to the mailing-list at <https://sympa.ethz.ch>".

The main content area shows the project details for 'experimental-project-1' (Project ID: 6107). It includes statistics for 7 Commits, 1 Branch, 0 Tags, and 9.5 MB Files. A commit history table is displayed below:

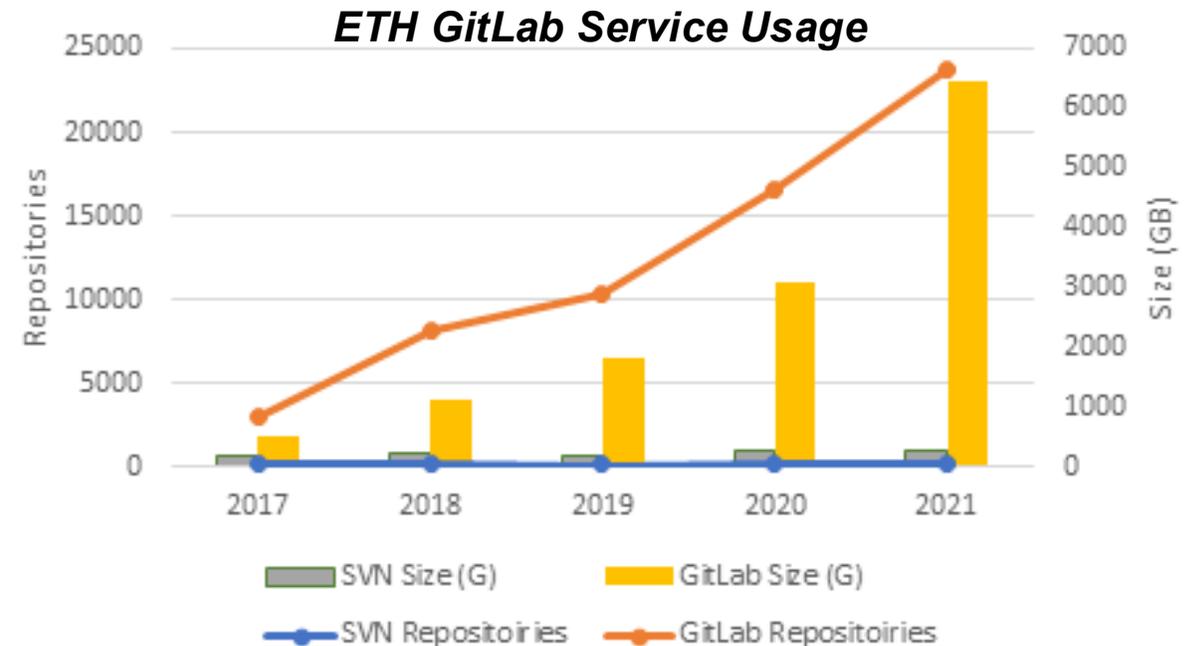
Name	Last commit	Last update
data	change image file size	4 months ago
.gitattributes	my first commit	5 months ago
analysis_code.py	change	4 months ago

<https://gitlab.ethz.ch>

ETH Zurich GitLab Service



- Integrated file, task and documentation management for individuals and / or groups
- Private, group and public repositories
- Built-in light-weight Wiki (protocols, list of materials etc.)
- Free for small repositories (< 2GB), otherwise yearly price of 250 CHF / TB / year
- Local and remote copies (off-site backup)
- Data can be exported (e.g. to Github)
- Built-in Container registry



Git – General Recommendations & Resources

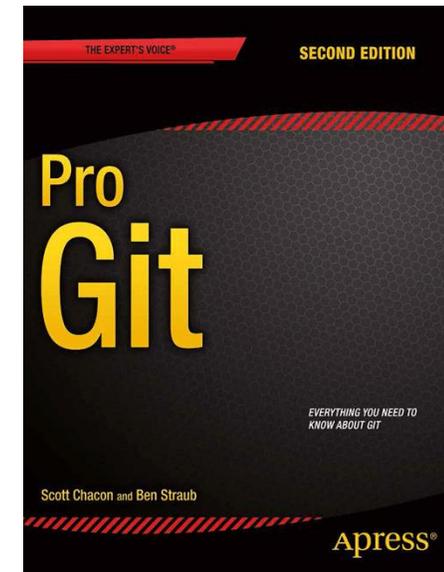


Recommendations for working with Git

- Commit early & often
- Provide short but meaningful commit messages
- Do not store large data files in Git repositories
 - e.g. images, movies, binary files
 - Use `.gitignore` file to exclude
 - Or consider tools such as [git-lfs](#) or [git-annex](#)
- Beware when resolving conflicts during *merge* or *pull* operations
 - A successful merge for Git may not be a successful merge for you

Resources for getting started with Git

- SIS can provide hands-on Git tutorials / workshops
- [Pro Git book](#) by S. Chacon & B. Straub
- Numerous tutorials available on the web / YouTube
 - [W3Schools Git tutorial](#)
 - [Software Carpentry Git course](#)
 - [Git tutorial for scientists](#)
- [List of Git GUI clients](#)



Management of source code: Questions?



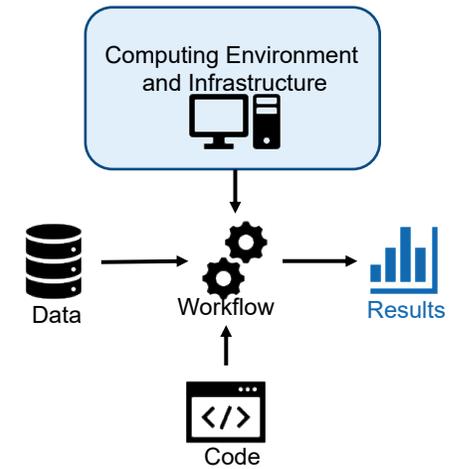
Managing Dependencies & Computing Environments



Reproducible Computing Environment

Problem:

Full reproducibility requires the possibility to recreate the system that was originally used to generate the results



Reproducible Computing Environment

Problem:

Full reproducibility requires the possibility to recreate the system that was originally used to generate the results

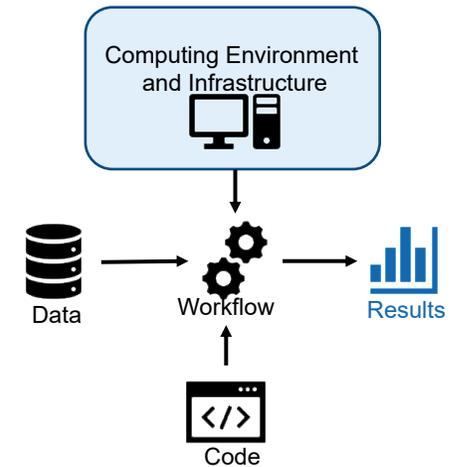
Solution:

Bundle your application and all dependencies

→ Environment Isolation & Dependency management

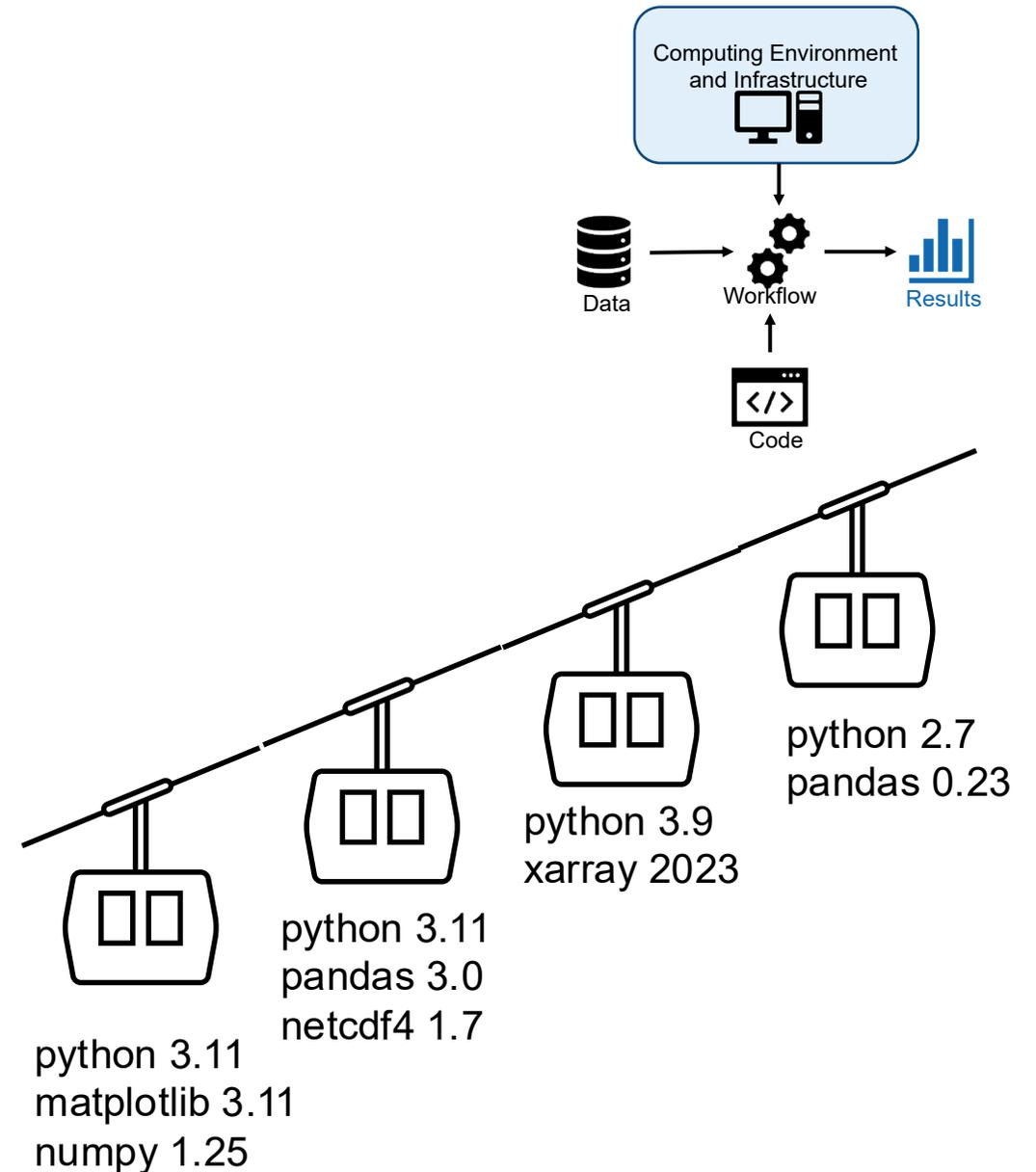
Tools:

- Application / software level: Conda, pip, virtualenv, renv, Devbox
- Containerization: Docker
- Virtualization (Virtual Machine, VM): VirtualBox, Vmware, Parallels



Virtual Computing Environments

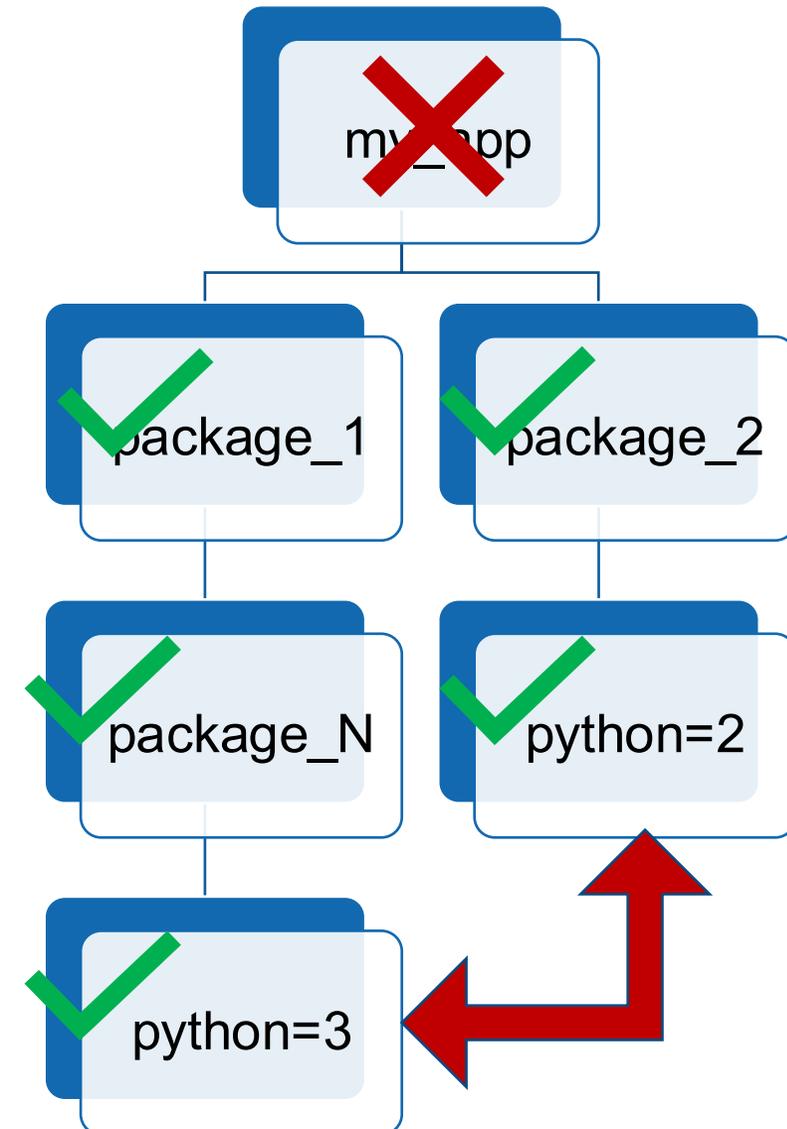
- Separated, self-contained builds of your coding language of choice (e.g. python, R, Julia, etc.)
- Great for projects that are solely in your coding language
- Some tools also work for projects with dependencies in multiple languages
- We recommend one virtual environment for each coding project



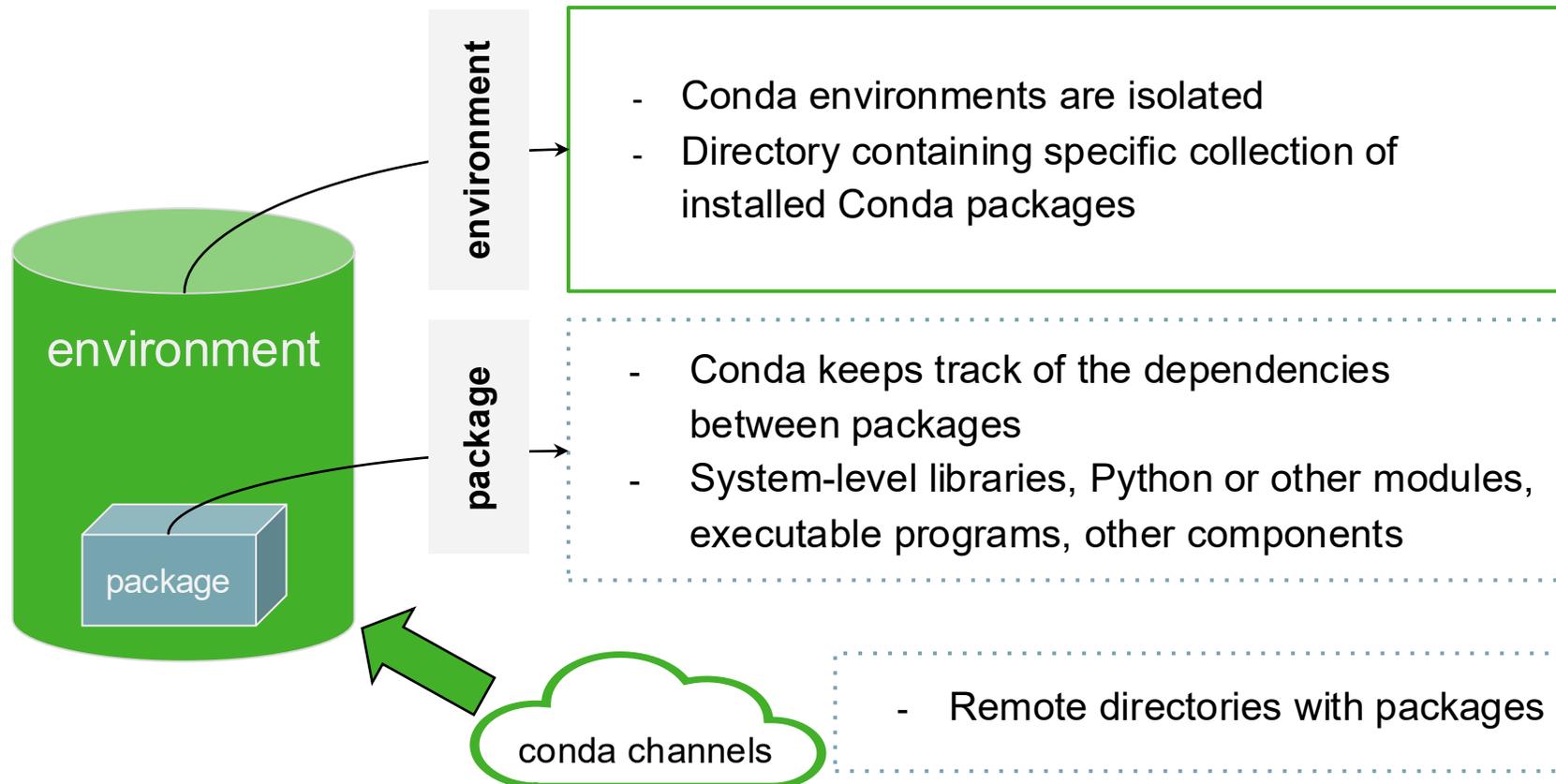
Reproducible Environment for R and Python



- Open source: Anaconda, Miniconda, Miniforge
- Commercial support: Anaconda Enterprise
 - **Note:** *certain functionality requires a paid license outside education / academia*
- Multi-platform: Windows, macOS, Linux
- Environment Management System
 - Isolated computing environments on the same system
 - Documentation of the computing environment
- Package Management System
 - Supported programming Languages: Python, R, ...
 - System libraries shipped in binary format
 - Resolve dependencies & conflicts between packages



Conda in a Nutshell



environment.yml

```
channels:  
- conda-forge  
  
dependencies:  
- python=3.8  
- jupyterlab
```

Conda automatically creates an environment file with packages and dependencies

Environment and Package Management Systems

Language	Environment Management	Package Management	Comments
Python 2 (not supported)	virtualenv, conda	pip, conda	
Python 3	venv, virtualenv, pipenv, poetry, uv, conda	pip, pipenv, poetry, uv, conda	Not all can install different Python versions
R	renv, conda	renv, conda	only conda can install different R versions
Julia	Pkg, conda	Pkg, conda	conda provides outdated Julia versions
Matlab	N/A	Add-on manager, Matlab Package Manager (unofficial)	Matlab search path determines dependencies



Alternatives to Conda are emerging!

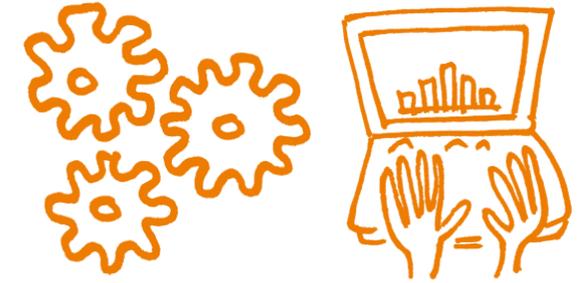


[pixi](#)



[Devbox](#)

Conda Hands-on Session



https://siscourses.ethz.ch/reproducible_computing/Conda.slidy.html



Home

Environments

Projects (beta)

Learning

Community

Documentation

Developer Blog

Feedback

Search Environments

Installed

Channels

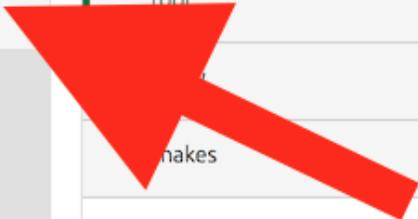
Update index...

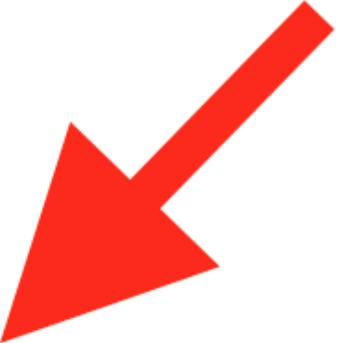
Search Pac...

Name	T	Description	Version
alabaster	○	Configurable, python 2+3 compatible sphinx theme	0.7.10
anaconda	○		custom
anaconda-client	○	Anaconda.org command line client library	1.6.3
anaconda-project	○	Reproducible, executable project directories	0.6.0
anyqt	○	Pyqt4/pyqt5 compatibility layer.	0.0.8
appnope	○		0.1.0
appscript	○		1.0.1
asn1crypto	○		0.22.0
astroid	○	Abstract syntax tree for python with inference support	1.4.9
astropy	○	Community-developed python library for astronomy	1.3.2
babel	○	Utilities to internationalize and localize python applications	2.4.0
backports	○		1.0
backports.shutil-get-terminal-size	○		1.0.0
beautifulsoup4	○	Python library designed for screen-scraping	4.6.0
bitarray	○		0.8.1

200 packages available

Create Clone Import Remove



Conda - What can go wrong?

- The package metadata (dependency list) is updated (not very likely)
- The package is deleted by the owner
- The package is not available under another platform
- There is no conda package for what you are looking for
- Complex dependencies may fail or take a long time to resolve

Virtualizing Computing Environments



Conda - What can go wrong?

- The package metadata (dependency list) is updated (not very likely)
- **The package is deleted by the owner**
- **The package is not available under another platform**
- **There is no conda package for what you are looking for**
- Complex dependencies may fail or take a long time to resolve

Reproducible Environment

Problem:

Full reproducibility requires the possibility to recreate the system that was originally used to generate the results

Solution:

Bundle your application and all dependencies

→ Environment Isolation & Dependency management

Tools:

- Application / software level: Conda, pip, virtualenv, renv
- Containerization: Docker
- Virtualization (Virtual Machine, VM): VirtualBox, VMware, Parallels

Reproducible Environment – Virtual Machines

- A virtual machine (VM) is an operating system (“guest”) that runs inside another computing environment (“host”).
- **Advantages:**
 - Allows multiple OS environments on a single physical computer
 - VMs are widely available and are easy to manage, maintain and distribute
 - Offers application provisioning and disaster recovery options
- **Drawbacks:**
 - They are not as efficient as a physical computer because the hardware resources are distributed in an indirect way.
 - Multiple VMs running on a single physical machine can deliver unstable performance



Source: <https://searchservirtualization.techtarget.com/definition/virtual-machine>

Reproducible Environment – Containers

- A mini, more flexible, virtual machine
- Great for projects that require specific system requirements / applications / tools / OS
- Larger than a virtual environment but even better for long-term reproducibility
- Docker / Podman / Colima / Apptainer are programs that help you create, run and deliver containers

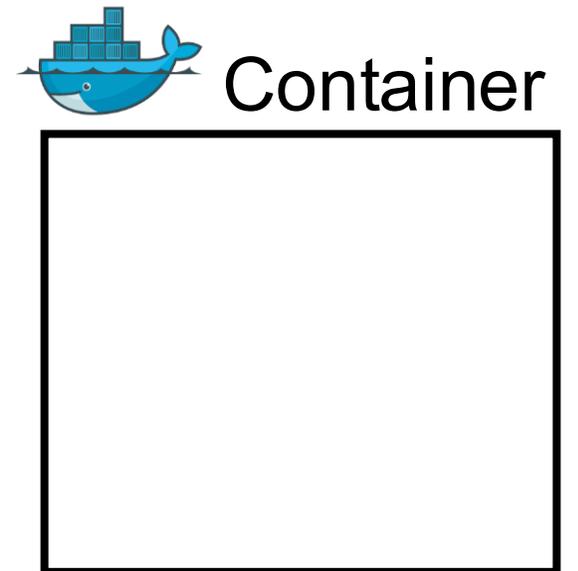
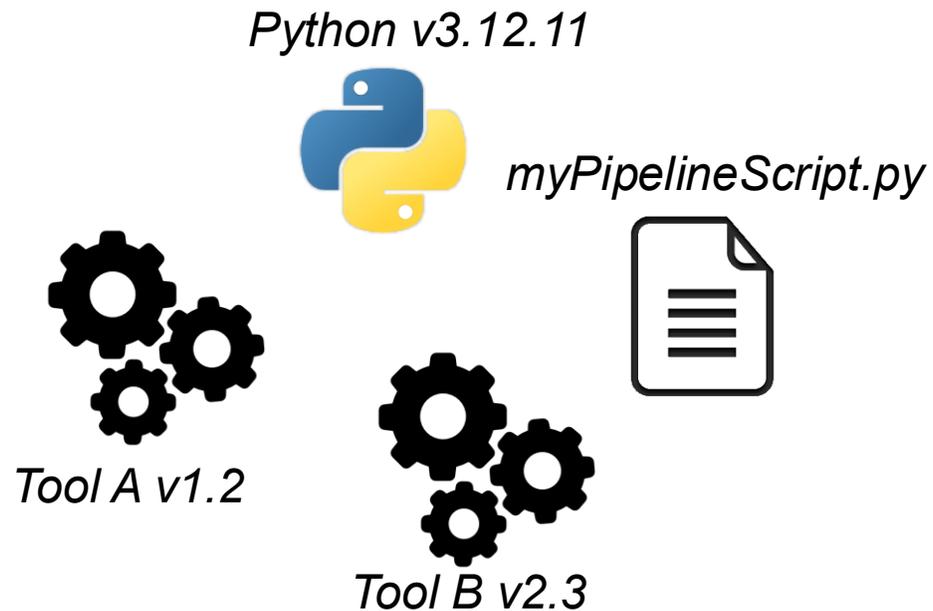


podman



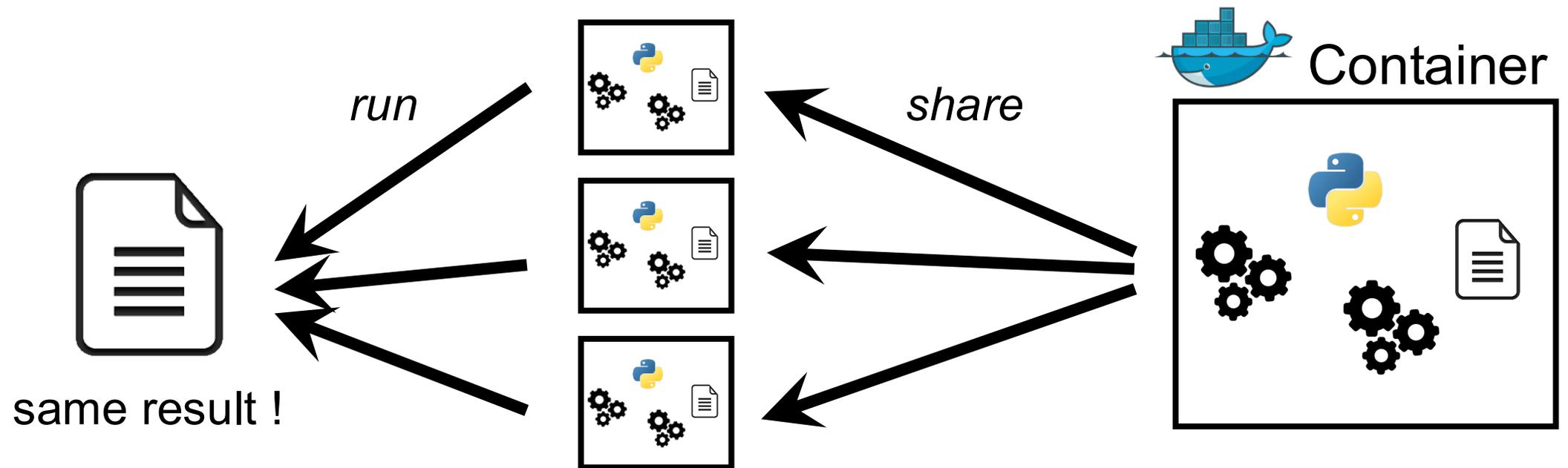
Reproducible Environment – Containers

- **Containers** allow you to **package** your software / pipeline with the **dependencies** inside a **reproducible**, easy to **share**, **runnable** file
- Example: **Docker containers**



Reproducible Environment – Containers

- **Containers** allow you to **package** your software / pipeline with the **dependencies** inside a **reproducible**, easy to **share**, **runnable** file
- Example: **Docker containers**



Reproducible Environment – Containers

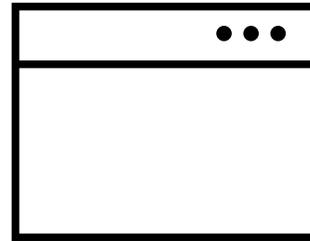
Pull back the curtain on containerization



Dockerfile

Small text file with instructions

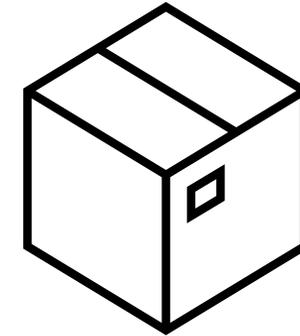
Build



Docker image

Packaged snapshot of everything
(OS, packages, code) ready to be used

Run



Docker container

A running instance of your image,
Package of software

*an instance because you can run
multiple containers from one image

Reproducible Environment – Containers

Pull back the curtain on containerization

- Write or use an existing Dockerfile

```
FROM python:3

WORKDIR /usr/src/app

COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD [ "python", ".scripts/wind_speed_at_dorthys_house.py" ]
```

Reproducible Environment – Containers

Pull back the curtain on containerization

- Write or use an existing Dockerfile

Dockerfile for pflotran

Starting with Ubuntu

Installing various programs like git, gfortran, python

Setting up a virtual environment

```
# Base image and setup
FROM ubuntu:22.04 AS base
LABEL org.opencontainers.image.ref.name="ubuntu"
LABEL org.opencontainers.image.version="22.04"
LABEL org.opencontainers.image.authors="Pin Shuai"
LABEL org.opencontainers.image.description="Docker image for running PFLOTRAN"

ARG NB_USER=aggie
ARG NB_UID=1000
ARG NB_GID=100
ARG PETSC_VERSION=v3.21.4 # Replace with the desired PETSc version so it work
ARG pflotran_branch=v6.0 # Replace with the desired PFLOTRAN branch
ENV DEBIAN_FRONTEND=noninteractive
ENV NB_USER=$NB_USER NB_UID=$NB_UID NB_GID=$NB_GID
ENV SHELL=/bin/bash
ENV PATH=/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

ENV PETSC_DIR=/scratch/petsc \
    PETSC_ARCH=petsc-arch

# Add fix-permissions script
COPY fix-permissions /usr/local/bin/fix-permissions
RUN chmod +x /usr/local/bin/fix-permissions

# Install prerequisites
RUN apt update -qq && \
    apt install -y --no-install-recommends \
    git make cmake gcc gfortran g++ wget nano vim python3 python3-pip local \
    apt clean && \
    rm -rf /var/lib/apt/lists/*

# Create user and set permissions. --m creates the home directory, --s sets the
RUN useradd --m --s /bin/bash --H -u $NB_UID $NB_USER && \
    fix-permissions /home/$NB_USER

# Conda and Python setup
USER $NB_USER
WORKDIR /home/$NB_USER

# Install jupyter and Install Python packages
RUN pip3 install -U virtualenv
RUN pip3 install -U jupyterlab h5py matplotlib hydroeval numpy scipy pandas && \
    ENV PATH=/home/$NB_USER/.local/bin:$PATH

# Add JupyterLab configuration
USER root
COPY jupyter_server_config.py /etc/jupyter/
COPY jupyter_notebook_config.py /etc/jupyter/
RUN chmod +x /etc/jupyter/jupyter_server_config.py

COPY jupyter_notebook_config.py /etc/jupyter/
RUN chmod +x /etc/jupyter/jupyter_server_config.py

# Set up the build environment
WORKDIR /scratch

# Expose ports and final configuration
ENV JUPYTER_ENABLE_LAB=yes
EXPOSE 8888

# Build MPICH
RUN wget https://www.mpich.org/static/downloads/4.1/mpich-4.1.tar.gz --no-check-certificate && \
    tar -xzf mpich-4.1.tar.gz && \
    cd mpich-4.1 && \
    ./configure --prefix=/scratch/build-mpich && \
    make && \
    make install && \
    mv /scratch/build-mpich/bin/* /scratch/build-mpich/bin/ && \
    rm -rf /scratch/build-mpich

# Copy and build PETSc
COPY ./build-petsc.sh /scratch/build-petsc.sh
RUN chmod +x /scratch/build-petsc.sh && \
    ENV PATH=/scratch/mpich-4.1/install/bin:$PATH

# Verify PETSc environment
RUN echo "PETSc version = ${PETSC_VERSION}" && \
    echo "PETSc directory = ${PETSC_DIR}" && \
    echo "PETSc arch = ${PETSC_ARCH}"

# Clone PFLOTRAN
RUN git clone https://bitbucket.org/pflotran/pflotran.git /scratch/pflotran && \
    cd /scratch/pflotran && \
    checkout ${pflotran_branch} && \
    "PFLOTRAN branch = ${pflotran_branch}" && \
    cd /scratch/pflotran/src/pflotran && \
    clean && \
    j4 \
    _code_coverage=1 \
    _runtime_checks=1 \
    _warnings_as_errors=1 \
    && \
    cd /scratch/pflotran/src/pflotran/bin && \
    ./pflotran

# Set environment variables
ENV NB_UID=$NB_UID NB_GID=$NB_GID --from=root:root --R /scratch/pflotran/src/pflotran \
    _code_coverage=1 \
    _runtime_checks=1 \
    _warnings_as_errors=1 \
    && \
    cd /scratch/pflotran/src/pflotran/bin && \
    ./pflotran

# back to user home
USER $NB_USER
WORKDIR /home/$NB_USER

# Set environment variables
ENV NB_UID=$NB_UID NB_GID=$NB_GID --from=root:root --R /scratch/pflotran/src/pflotran \
    _code_coverage=1 \
    _runtime_checks=1 \
    _warnings_as_errors=1 \
    && \
    cd /scratch/pflotran/src/pflotran/bin && \
    ./pflotran

# Set environment variables
ENV NB_UID=$NB_UID NB_GID=$NB_GID --from=root:root --R /scratch/pflotran/src/pflotran \
    _code_coverage=1 \
    _runtime_checks=1 \
    _warnings_as_errors=1 \
    && \
    cd /scratch/pflotran/src/pflotran/bin && \
    ./pflotran
```

Configuring jupyter notebook

Setting up pflotran

<https://groundwater.usu.edu/>

Reproducible Environment – Containers

Pull back the curtain on containerization

- Write or use an existing Dockerfile
- Build the image
 - Or use an existing one:
<https://hub.docker.com/search?badges=official>

```
docker build -t tornado_analysis
```

Reproducible Environment – Containers

Pull back the curtain on containerization

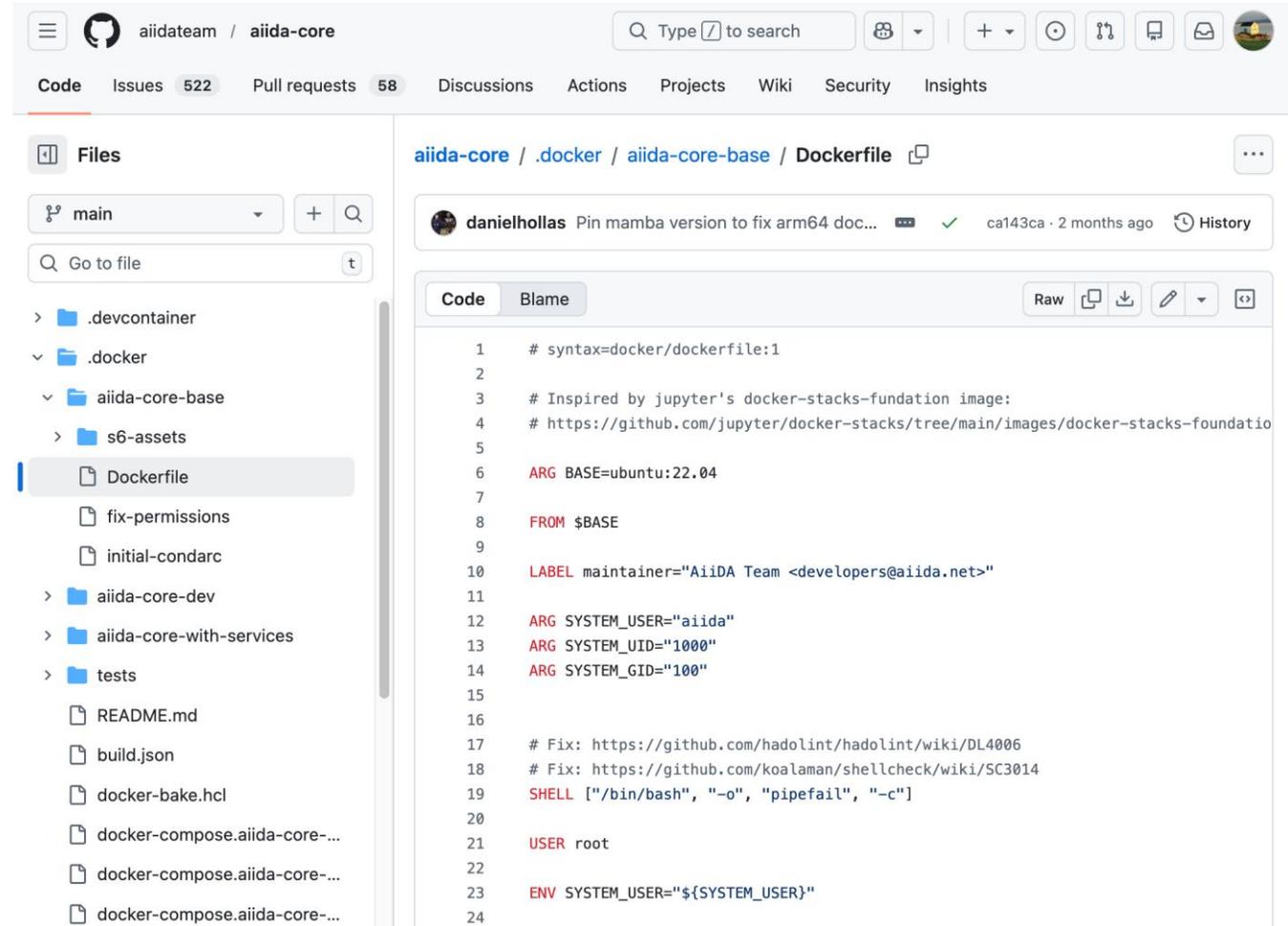
- Write or use an existing Dockerfile
- Build the image
 - Or use an existing one:
<https://hub.docker.com/search?badges=official>
- Run the container

```
docker run tornado_analysis
```

Reproducible Environment – Containers

Pull back the curtain on containerization

- Write or use an existing Dockerfile
- Build the image
 - Or use an existing one:
<https://hub.docker.com/search?badges=official>
- Run the container
- Share your Dockerfile when you share your code



The screenshot shows a GitHub repository for 'aiida-core' by the 'aiidateam'. The file browser on the left shows the directory structure, with the 'Dockerfile' file selected under the '.docker' directory. The main content area displays the Dockerfile code, which is a multi-stage build for an AiiDA workflow manager container. The code includes comments about the image's inspiration and fixes, and defines environment variables for the base image, user, and group.

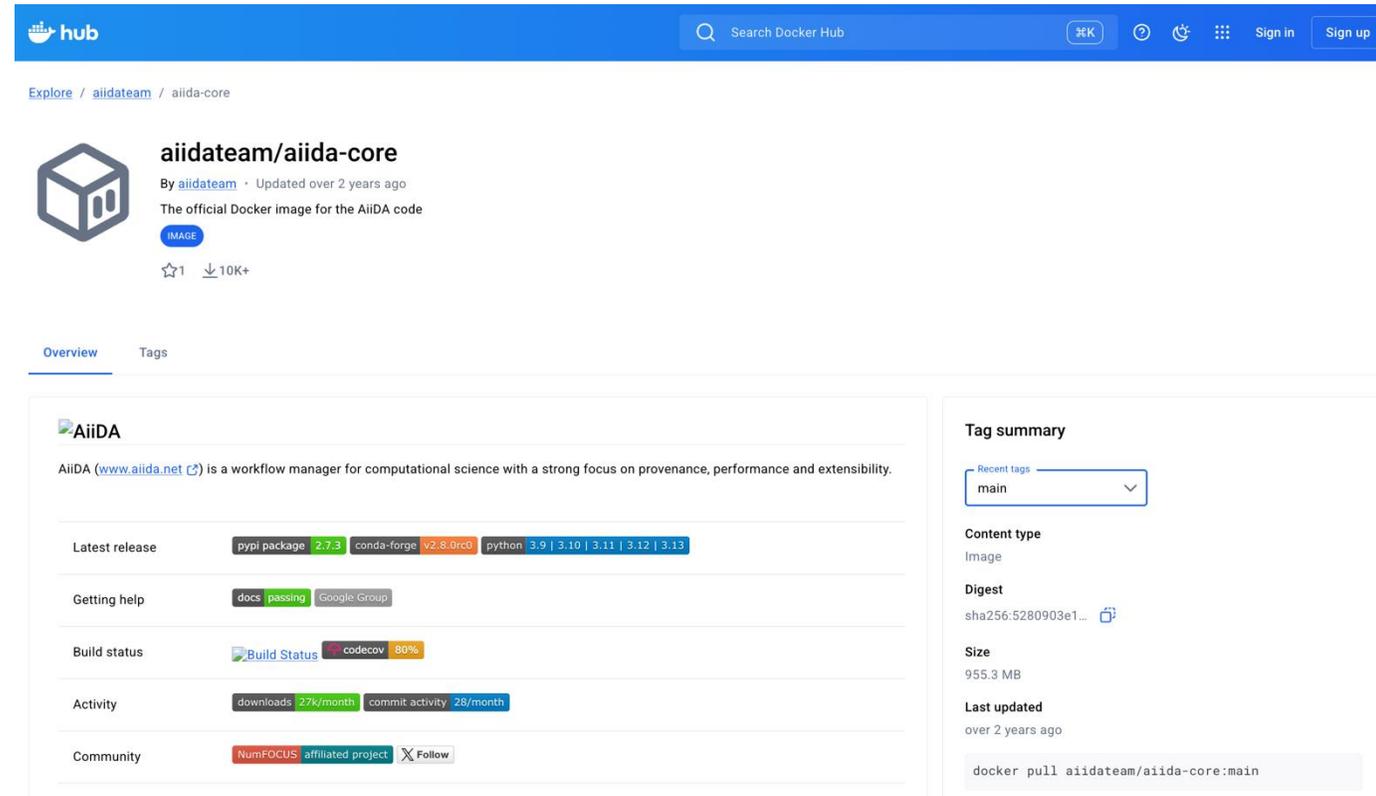
```
1 # syntax=docker/dockerfile:1
2
3 # Inspired by jupyter's docker-stacks-foundation image:
4 # https://github.com/jupyter/docker-stacks/tree/main/images/docker-stacks-foundatio
5
6 ARG BASE=ubuntu:22.04
7
8 FROM $BASE
9
10 LABEL maintainer="AiiDA Team <developers@aiida.net>"
11
12 ARG SYSTEM_USER="aiida"
13 ARG SYSTEM_UID="1000"
14 ARG SYSTEM_GID="100"
15
16
17 # Fix: https://github.com/hadolint/hadolint/wiki/DL4006
18 # Fix: https://github.com/koalaman/shellcheck/wiki/SC3014
19 SHELL ["/bin/bash", "-o", "pipefail", "-c"]
20
21 USER root
22
23 ENV SYSTEM_USER="${SYSTEM_USER}"
24
```

Dockerfile for [AiiDA workflow manager](#) on Github

Reproducible Environment – Containers

Pull back the curtain on containerization

- Write or use an existing Dockerfile
- Build the image
 - Or use an existing one:
<https://hub.docker.com/search?badges=official>
- Run the container
- Share your Dockerfile when you share your code
- Archive your Docker **image** for long-term preservation



The screenshot displays the Docker Hub interface for the repository `aiidateam/aiida-core`. The page includes a search bar at the top, navigation links, and a detailed overview of the repository. The overview section features several badges: 'Latest release' with links to 'pypi package 2.7.3', 'conda-forge v2.8.0rc0', and 'python 3.9 | 3.10 | 3.11 | 3.12 | 3.13'; 'Getting help' with 'docs passing' and 'Google Group'; 'Build status' with 'Build Status' and 'codecov 80%'; 'Activity' with 'downloads 27k/month' and 'commit activity 28/month'; and 'Community' with 'NumFOCUS affiliated project' and 'Follow'. The 'Tag summary' section on the right shows the 'main' tag, which is an 'Image' with a digest of 'sha256:5280903e1...' and a size of '955.3 MB'. It was last updated 'over 2 years ago'. A terminal snippet at the bottom right shows the command `docker pull aiidateam/aiida-core:main`.

Docker images for [AiiDA workflow manager](#) on Docker Hub

Reproducible Environment – Containers

Difference between Dockerfile, Image and Container

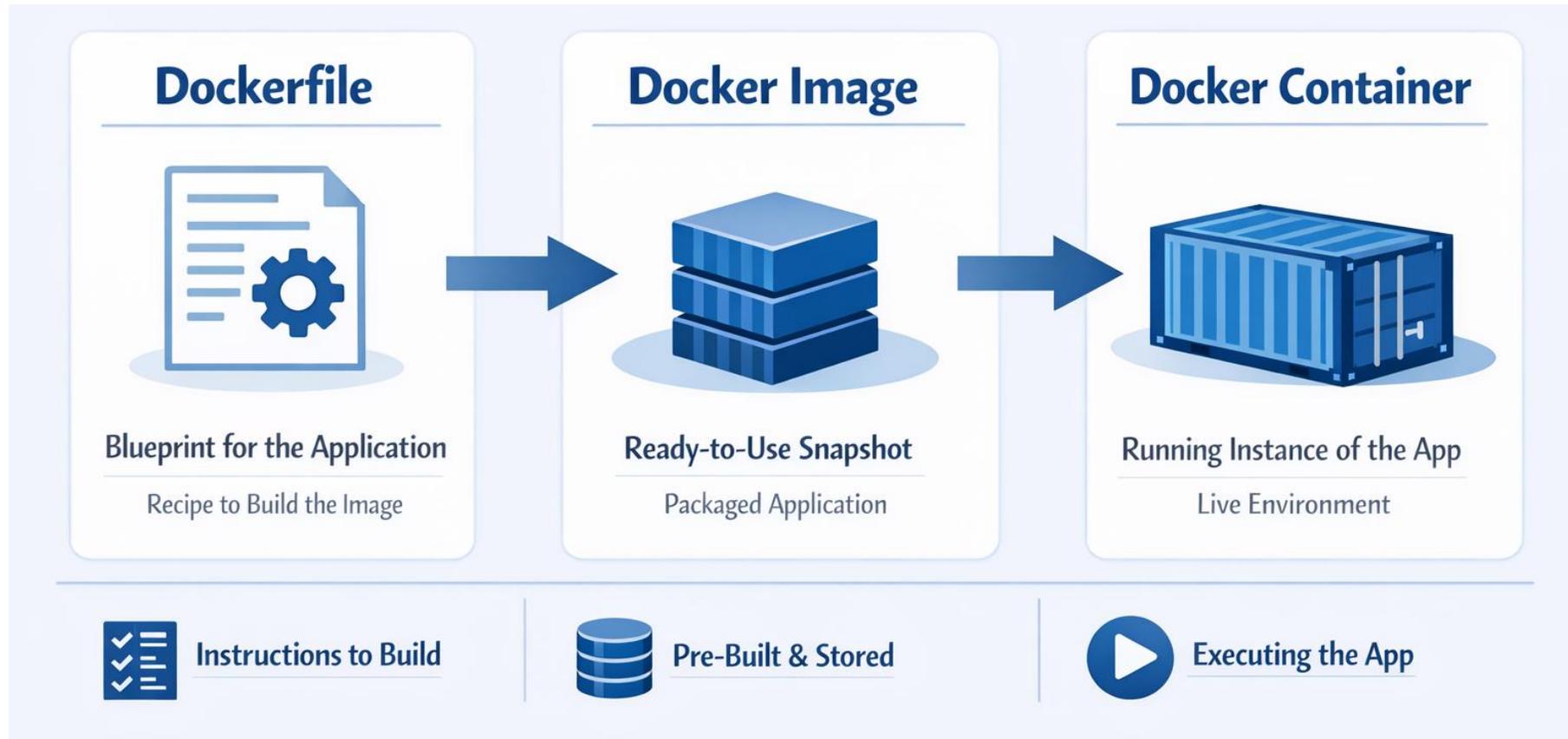


Image generated with DALL·E (OpenAI), March 2026.

Virtual Machines vs Containers

	VMs (Virtual Box)	Containers (Docker)
Use case	Complex Apps (GUI, ...)	Data Analysis Scripts, Simple Apps, Microservices, Continuous Integration
Virtualization	Hardware-level	OS-level
Size	GB	MB
Startup time	Minutes	Seconds
Guest OS	Windows, macOS, Linux	Primarily Linux-based
Host OS	Windows, macOS, Linux	Linux, Windows 10 / macOS with hypervisor
Overhead (RAM, CPU)	High - reduced performance	Low - close to native performance
Security	Better (fully isolated)	Poorer (shared kernel)
How to use	Easy if you know to install OS	New things to learn
Getting started	www.virtualbox.org/manual/ch01.html	https://docs.docker.com/get-started/

Reproducible computational environment: Questions?





File Edit View Run Kernel Tabs Settings Help

Lorenz.ipynb Python 3

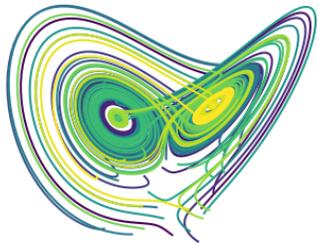
We explore the Lorenz system of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

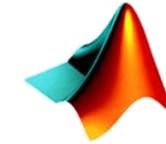
Let's change (σ, β, ρ) with ipynwidgets and examine the trajectories.

In [2]: `from lorenz import solve_lorenz`
`w=interactive(solve_lorenz,sigma=(0.0,50.0),rho=(0.0,50.0))`
`w`

sigma 10.00
beta 2.67
rho 28.00



```
6 def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
7     """Plot a solution to the Lorenz differential equations."""
8
9     max_time = 4.0
10    N = 30
11
12    fig = plt.figure()
13    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
14    ax.axis('off')
15
16    # prepare the axes limits
17    ax.set_xlim((-25, 25))
18    ax.set_ylim((-35, 35))
19    ax.set_zlim((5, 55))
20
21    def lorenz_deriv(x,y,z, t0, sigma=sigma, beta=beta, rho=rho):
22        """Compute the time-derivative of a Lorenz system."""
23        x, y, z = X,Y,Z
24        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
25
26    # Choose random starting points, uniformly distributed from -15 to 15
27    np.random.seed(1)
28    x0 = -15 + 30 * np.random.random((N, 3))
29
30    # Solve for the trajectories
31    t = np.linspace(0, max_time, int(250*max_time))
32    x_t = np.asarray([integrate.odeint(lorenz_deriv, x0i, t)
33                    for x0i in x0])
34
35    # choose a different color for each trajectory
36    colors = plt.cm.viridis(np.linspace(0, 1, N))
37
38    for i in range(N):
39        x, y, z = x_t[i, :, :].T
40        lines = ax.plot(x, y, z, '-', c=colors[i])
41        plt.setp(lines, linewidth=2)
42    angle = 104
43    ax.view_init(30, angle)
```



MATLAB
Live Editor



WolframAlpha
NOTEBOOK EDITION™

Interactive Computational Notebooks



Interactive Notebooks

- Applications that combine documentation, code, input and output generated by the code, e.g. graphs, plots ([Nature 515, 151–152](#))
- Useful for exploratory data analysis, sharing, and reproducibility



- Open source + commercial edition
- Mainly for development in R but other languages supported



- Open source
- > 40 languages supported (Python, R, Julia, Matlab, IDL, etc.)



- Commercial
- Used in mathematical fields



- Commercial
- Used in scientific, engineering, mathematical fields

Interactive Notebooks: Jupyter

- **Jupyter notebook:** web-based interactive computational environment
- **JupyterLab:** web-based interactive development environment for notebooks, code, and data

The screenshot displays the JupyterLab interface. On the left is a file browser showing a directory structure with files like 'data', 'README.md', and 'requirements.txt'. The main area is a Jupyter notebook titled 'Image Analysis with Python and Jupyter'. The notebook content includes:

Image Analysis with Python and Jupyter

This Jupyter notebook demonstrates the analysis of a simple image with Python.

First, make sure you have the image `blobs.tif` in the `data` folder of your notebook dashboard.

Import required modules

```
[1]: import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imread
from skimage.morphology import reconstruction, remove_small_objects
from skimage.measure import label, regionprops

%matplotlib inline
```

Read and display the data

```
[2]: img = imread('data/blobs.tif')

[3]: fig = plt.figure(figsize=(10,5))
fig.add_subplot(121)
plt.imshow(img, cmap='gray')
plt.title('blobs')
fig.add_subplot(122)
plt.hist(img.ravel(), bins=10)
plt.title('Histogram of blobs')
plt.show()
```

The notebook output shows two plots: a grayscale image of 'blobs' and a histogram titled 'Histogram of blobs'.

On the right side of the interface, there are two files open: 'requirements.txt' and 'README.md'. The 'requirements.txt' file contains the following text:

```
1 # This file may be used to create an environment using:
2 # $ conda create --name <env> --file <this file>
3 # platform: osx-64
4 matplotlib
5 numpy
6 pandas
7 scikit-image
8 scipy
9
```

The 'README.md' file contains the following text:

```
1 # Jupyter-Demo-RDM
2
3 Demo of Jupyter notebook for ETH ARDM workshops
4
5 \[\[Binder\]\] \(https://mybinder.org/badge\_logo.svg\)
6 (https://mybinder.org/v2/gh/hluetck/Jupyter-Demo-
7 RDM/master)
8
```

Interactive Notebooks – what can go wrong?

- **Versioning**

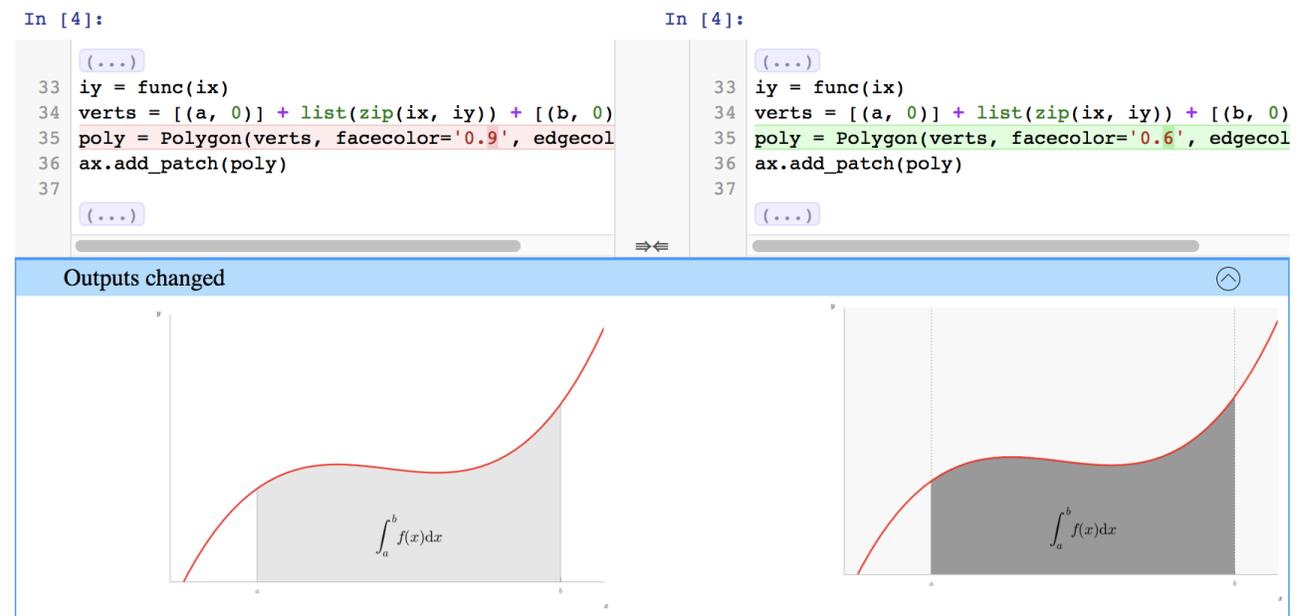
- Version control of even moderately complex NBs is challenging
- Tracking NB history is harder than for traditional source code
- Some tools may help (e.g. [nbdime](#), [JupyterText](#), [nbstripout](#), [nbconvert](#))

```
$ diff a.ipynb b.ipynb
76,77d75
<     "plt.rc('axes', grid=False)\n",
<     "plt.rc('axes', facecolor='white')\n",
90c88
<     "image/png": "iVBORw0KGGoAAAANSUgAABLkAAAMQCAYAAADLj7d\lAAAABHNCSVQICAgIFAhki
AAAAAlwSFlz\nAAAWJQAAFiUBSVIk8AAAIABJREFUJzsvXeYZFd57b12h0maPNJII2lG0aCAKEBCFgozIxBAP
lY\nlwaDyDZg8MX+zMU2F4Mx1x8PwAwxmBjg4yNi2BfQMa20iiAQFkIjXKWRtJIE3tSz3TXuX+8vV2n\nqyucv
N+9z/o9zzynprvq1D6nqqtr1prbRNFQghhBBCCCGEEEEII8Zkh1wMghBBCCCGEEEEIIISQv\nFLkIIYQQQgghhB
BCiPdQ5CKEEEEIIYQQQggh3k0RixBCCCGEEEEIIYR4D0UuQgghhBBCCCGEE0I9\nFLkIIYQQQgghhBBCiPdQ5CK
EEEEIIYQQQggh3k0RixBCCCGEEEEIIYR4D0UuQgghhBBCCCGEE0I9\nFLkIIYQQQgghhBBCiPdQ5CKEEEEIIYQQ
Qggh3k0RixBCCCGEEEEIIYR4D0UuQgghhBBCCCGEE0I9\nFLkIIYQQQjzEGH0JMaZljPmo67EkZWq8D7keByGEE
ELChCIXIYQQQirDGPOmKaFj3BhzkMNx/H/G\nmG3GmP/pagwFEbkeQJUYY75gjNljHmD67EQQgghRB8UuQghhB
BSJe+DCDMjAH7L4TjeAmA+gLc5\nHEMRGNcDqJi3AVgI4DddD4QQQggh+qDIRQghhJBKMMacCuBMAFsg4sy7jTH
DjobzZwBuBvBxR/dP\nnsVERADcC+LTrgRBCCCFEHxS5CCGEEFIVH4C4uP4SILQcBOD1LgYSRVEz iqIXR1H0frf3
T7IRRdFf\nRlH0K1EUXe96LIQQQgjRB0UuQgghhJSOMWYpgP8BoAXg7wH8HcTN9Tsux0UIIYQQQsKBHchhBBC\
nguBdA0YAuDyKoscBfByAlgBnGwDe73PkhBBCCCFkCChyEUTTTaRUHjDFGUHjTf0PxcjK7eDMB3n65C\nnNyxchhBBC
```

Interactive Notebooks – what can go wrong?

- **Versioning**

- Version control of even moderately complex NBs is challenging
- Tracking NB history is harder than for traditional source code
- Some tools may help (e.g. [nbdime](#), [JupyterText](#), [nbstripout](#), [nbconvert](#))



Interactive Notebooks – what can go wrong?

- **Versioning**
 - Version control of even moderately complex NBs is challenging
 - Tracking NB history is harder than for traditional source code, especially with “classical” git
 - Some jupyter-targeted tools may help
- **Reproducibility**
 - Interactive working mode can result in hard-to-reproduce notebooks
 - Discipline is needed! Regular pruning & refactoring; *“Restart kernel & Run all”* is your friend
- **Security**
 - Data confidentiality & access controls may be problematic



Reproducible Computing Platforms



Reproducible Computing Platforms

- Integrated, **web-based** solutions for **reproducible** and **collaborative** data analysis and **computing**
- Usually built upon **proven open-source technologies** (Git, Conda, Docker etc.)
- Technical **complexity hidden** from user (or made easily accessible)
- Platforms provide **low entry barrier** access to fully reproducible computing
- **Commercial platforms**
 - Examples: [Code Ocean](#), [Google Colaboratory](#), ...
 - Costs are incurred by usage of underlying cloud infrastructure (storage, compute, data transfer!)
 - Beware of data ownership, licensing issues and general T&Cs
- **Community platforms**
 - Examples: [mybinder](#), [Renkulab.io](#)
 - Usually free of charge but resources are limited

Reproducible Computing Platforms: *renkulab.io*

- [Renkulab](#) is a **platform for collaborative data science** from the [Swiss Data Science Center](#) (SDSC)

The screenshot shows the Renkulab.io website interface. At the top left is the 'renku' logo, which consists of the word 'renku' in a green, lowercase, sans-serif font followed by a stylized green and red grid icon. In the top right corner, there are links for 'Help' and 'Login', and a hamburger menu icon. The main content area features the text 'Connecting data, code, compute, and *people*.' in a large, white, sans-serif font. Below this is the text 'One seamless platform powering collaboration in your project, team, and community.' in a smaller, white, sans-serif font. At the bottom left, there are two buttons: a green button with white text 'Create an account' and a white button with a black border and black text 'Explore a project'. On the right side of the interface, there is a grid of icons representing various data science and computing concepts: a plus sign in a circle, a folder, a stack of data disks, a code editor icon, a globe, a person icon, a server rack, a cloud with a chip, and a 3D cube.

Reproducible Computing Platforms: *renkulab.io*

- [Renkulab](#) is a **platform for collaborative data science** from the [Swiss Data Science Center](#) (SDSC)

The screenshot shows the Renkulab.io interface for a project titled "N2O Pathway Analysis". The interface includes sections for Sessions, Data, and Code Repositories. Overlaid on the screenshot is a diagram with three main green circular nodes: "Compute", "Data", and "Code".

- Compute** node: Includes icons for Docker, Switch, and Jupyter.
- Data** node: Includes icons for ETH Research Collection, ETH polybox, Amazon S3, SWITCHdrive, and Azure.
- Code** node: Includes icons for GitHub and GitLab.

Learning more about Renkulab

- Login with your Switch edu-ID (or create a new account)
- [Getting Started Tutorial](#)
- More [Renku Tutorials](#)
- Renku [How-To Guides](#)

Reproducible Computing Platforms: *mybinder.org*

- [Binder](#) converts a Git repository into a collection of interactive notebooks
- Notebooks open in an executable environment → code becomes reproducible by anybody, anywhere

Build and launch a repository

GitHub repository name or URL

GitHub

Git ref (branch, tag, or commit) File to open (in JupyterLab) File

Fill in the fields to see a URL for sharing your Binder.

Badges for your README

Build Logs

How it works

1

Enter your repository information

Provide in the above form a URL or a GitHub repository that contains Jupyter notebooks, as well as a branch, tag, or commit hash. Launch will build your Binder repository. If you specify a path to a notebook file, the notebook will be opened in your browser after building.

2

We build a Docker image of your repository

Binder will search for a dependency file, such as requirements.txt or environment.yml, in the repository's root directory ([more details on more complex dependencies in documentation](#)). The dependency files will be used to build a Docker image. If an image has already been built for the given repository, it will not be rebuilt. If a new commit has been made, the image will automatically be rebuilt.

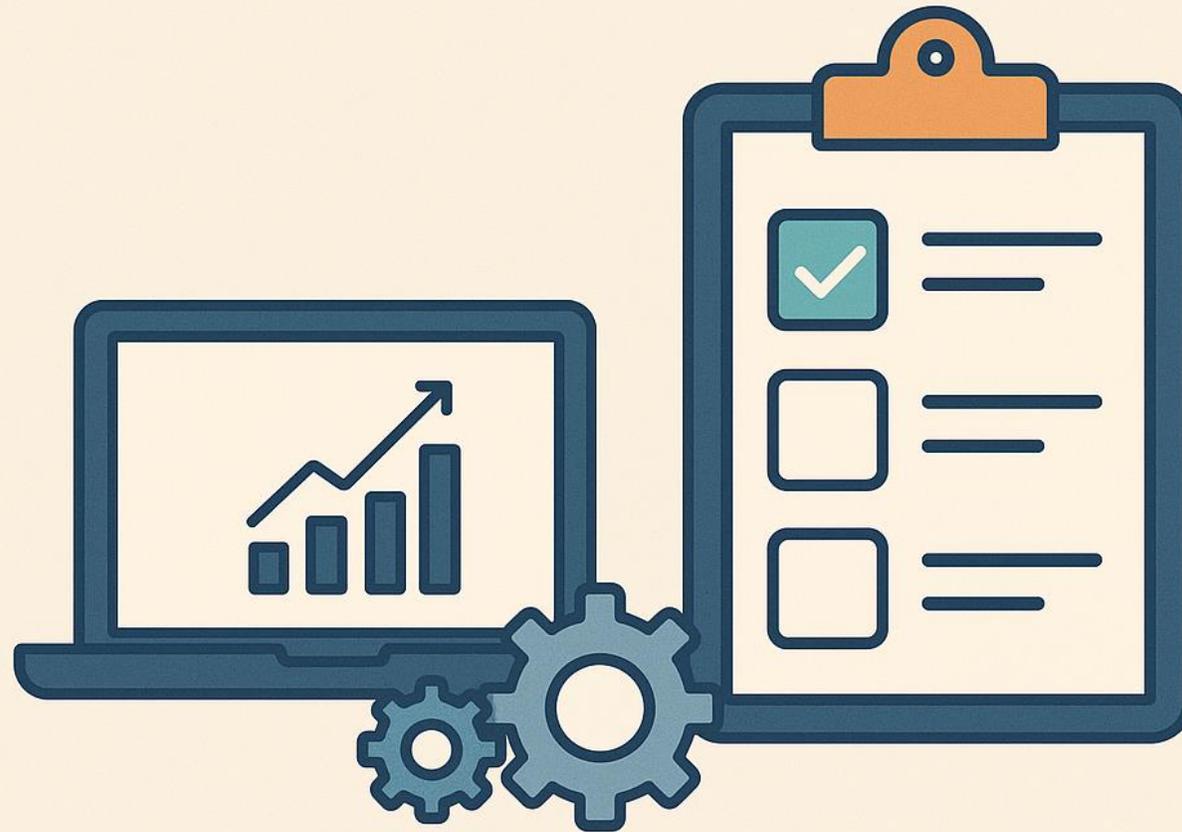
3

Interact with your notebooks in a live environment!

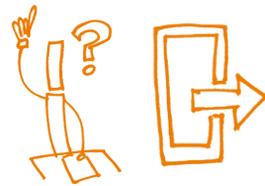
A [JupyterHub](#) server will host your repository's contents. We offer you a reusable link and badge to your live repository that you can easily share with others.

- Live example: <https://github.com/hluetck/mybinder-demo-project>

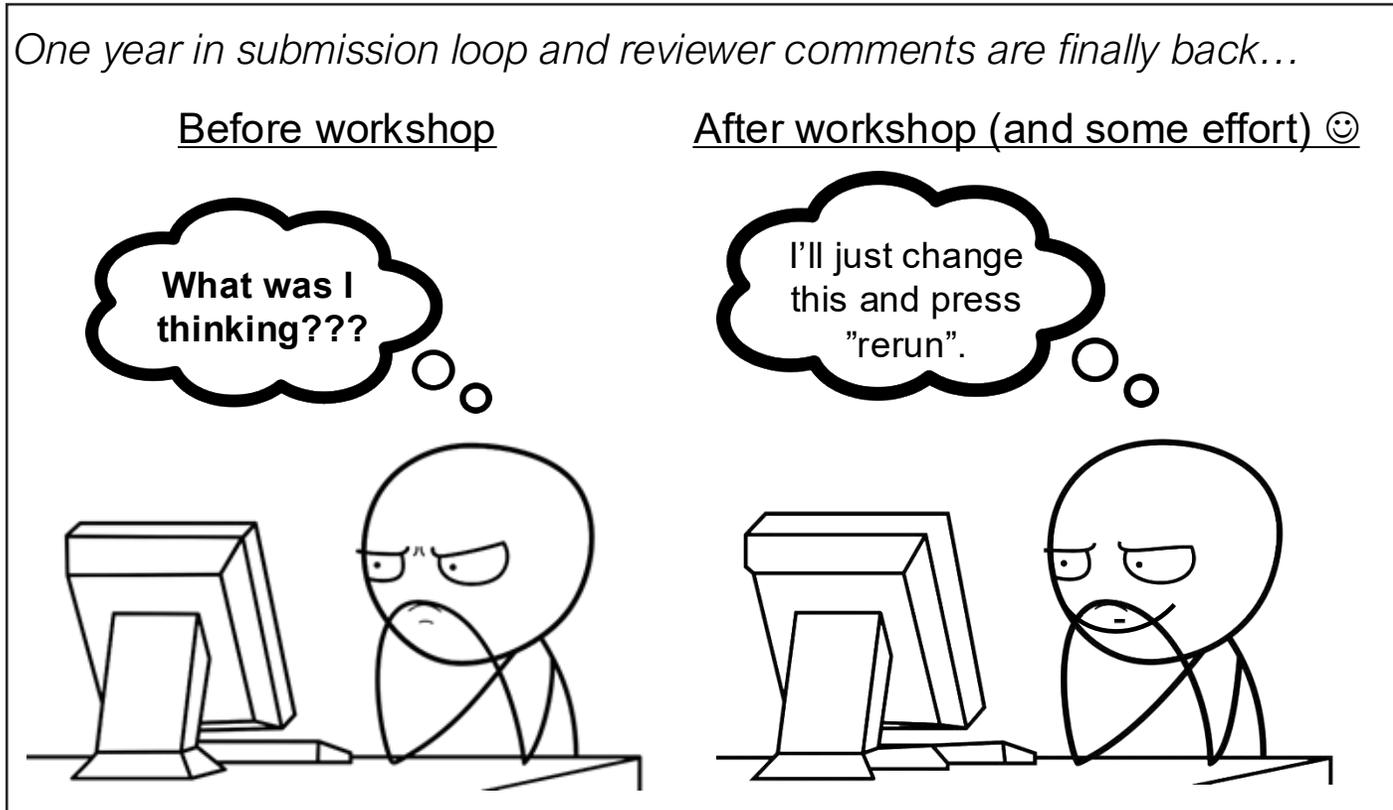
QUIZ TIME ...



Wrap-up & Discussion



What's in it for me?



At the start of the project

- Forced to think about scope and limitations
- Improved structure and organization

During the project

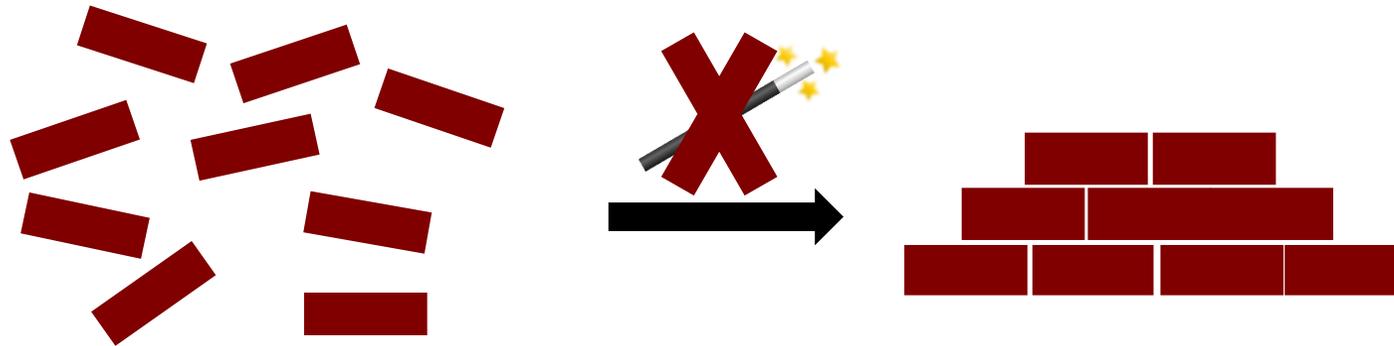
- Easier to rerun experiments and analysis
- Closer interaction between collaborators
- Much of the manuscript "writes itself"

After the end of the project

- Faster resumption of research by others (or your future self), thereby increasing the impact of your work
- Increased visibility in the scientific community

What's in it for me?

- Aim for improvement, not perfection!
- RDM requires **WORK & TIME**, but the time spent on this is an **investment** for the future!



Contact us for consultations / trainings on data management, version control, reproducible computational workflows or data science support

sis.helpdesk@ethz.ch





Open Research Data Portal
The ORD Program of the ETH Domain

Search ...

Home Projects **Training** RDS Explorer Documents About

Infos

Module Overview

This training offers comprehensive e-learning modules, designed to address key aspects of RDM and ORD, as well as to cater to various levels of familiarity with RDM and various research stages.

We are developing and releasing modules progressively, with the first one published in Feb. 2025. New modules, along with their respective OER, are added regularly. All modules are expected to be available by Dec. 2025.

E-Learning Modules

<p>Data Documentation and Metadata</p> <p>Available</p>	<p>Data Organisation and Management</p> <p>Available</p>	<p>Data Publishing and Long-Term Preservation</p> <p>Available</p>
---	--	--

Contacts

Nadia Marounina

nadejda.marounina@id.ethz.ch

Henry Lütcke

henry.lutcke@id.ethz.ch

sis.helpdesk@ethz.ch

<https://sis.id.ethz.ch/>

Feedback: <https://www.umfrageonline.ch/c/scientificcomputing>



[https://siscourses.ethz.ch/
reproducible_computing/](https://siscourses.ethz.ch/reproducible_computing/)



Any final questions on what we have discussed this morning?



[https://siscourses.ethz.ch/
reproducible_computing/](https://siscourses.ethz.ch/reproducible_computing/)



Feedback: <https://www.umfrageonline.ch/c/scientificcomputing>

