

# Research Software Engineering

DAY 1



DIGITAL RESEARCH  
ACADEMY

Dr. Johanna Bayer



The Journal of  
Open Source Software



THE  
CARPENTRIES



# Dr. Johanna Bayer



**Postdoctoral Fellow** at the Donders Institute for Brain, Cognition and Behaviour, the Netherlands

**DRA Trainer** (focus on software)

**Repronim Fellow**

**Editor** at **JOSS**

**The Carpentries** instructor and maintainer

**The Turing Way** core contributor, maintainer and governance member

**Datatacks.club** podcast producer

**Cat mom**

**Who are you and why are you here?**

# Outline

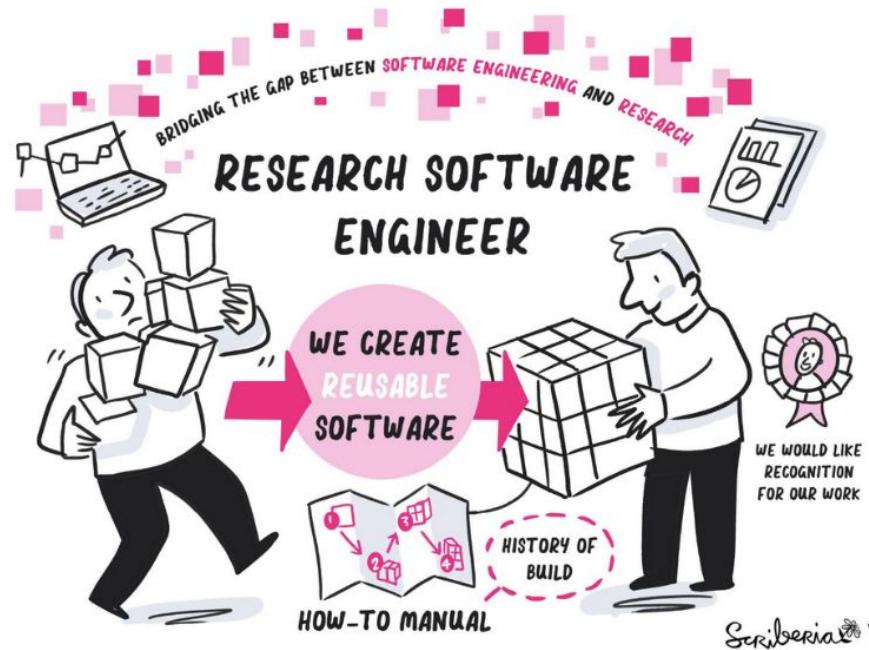
DAY 1				
Topic	Time	-	Duration	
Welcome + Introduction: Why are we here?	9:00	-	9:15	0:15
What is a research software engineer?	9:15	-	10:05	0:50
Setting up a research code project repository structure, naming, README				
<b>Activity 1 (20 min)</b>				
Break	10:05	-	10:20	0:15
Activities with Git Theoretical Background (30 min)	10:20	-	10:55	0:35
<b>Activity 2 (15 min)</b>				
Break	10:55	-	11:10	0:15
<b>Activity 3 (15 min)</b>	11:10	-	11:45	0:35
Theory and activities: collaborating effectively using issues, branches, forks, pull requests, merges and code reviews				
Wrap-up	11:45	-	12:00	0:15

# Discussion 1

## What is a Research Software Engineer?



# What is a Research Software Engineer?



# What is a Research Software Engineer?

Definition is quite imprecise and notes from the original meeting where the term was coined (UK, 2012), include more a description of what it NOT is:

*Roles: “research officer” – considered support don’t want to be servants*

*NOT permanent postdoc*

*Not a lab technician*

*Calling them software engineers is not good enough – domain specialised*

*“Researcher” has pressure to publish*

# What is a Research Software Engineer?

... the **research software engineer**. These individuals combine a *professional attitude* to the exercise of software engineering with a *deep understanding of research topics*.

*Computational work must reflect the committed attitude of experimentalists towards caring about precise, professional, repeatable, meticulous work – no-one with the same casual attitude to experimental instrumentation as many researchers have to code would be allowed anywhere near a lab. This is striking considering how often research results now depend on software.*

Baxter et al. 2012

# What is a Research Software Engineer?

F1000Research

F1000Research 2024, 13:1429 Last updated: 27 NOV 2024



Check for updates

OPINION ARTICLE

## Foundational Competencies and Responsibilities of a Research Software Engineer

[version 1; peer review: awaiting peer review]

Florian Goth <sup>1</sup>, Renato Alves <sup>2</sup>, Matthias Braun<sup>3</sup>, Leyla Jael Castro <sup>4</sup>, Gerasimos Chourdakis <sup>5,6</sup>, Simon Christ <sup>7</sup>, Jeremy Cohen <sup>8</sup>, Stephan Druskat <sup>9</sup>, Fredo Erxleben<sup>10</sup>, Jean-Noël Grad <sup>11</sup>, Magnus Hagdorn <sup>12</sup>, Toby Hodges<sup>13</sup>, Guido Juckeland <sup>10</sup>, Dominic Kempf <sup>14</sup>, Anna-Lena Lamprecht <sup>15</sup>, Jan Linxweiler <sup>16</sup>, Frank Löffler <sup>17</sup>, Michele Martone <sup>18</sup>, Moritz Schwarzmeier <sup>19</sup>, Heidi Seibold <sup>20</sup>, Jan Philipp Thiele <sup>21,22</sup>, Harald von Waldow <sup>23</sup>, Samantha Wittke<sup>24</sup>

# What is a Research Software Engineer?

Training/teaching

Work in research environment

Produce software

Communicators/Translators

self-identify

Produce papers

## Research Software Engineer

Research

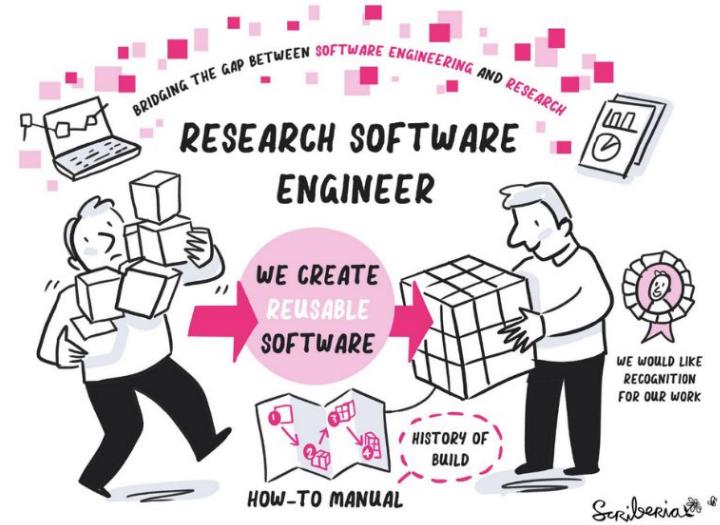
funding

Software Developer

Non-technical complexities of the  
research environment

# What is Research Software

Foundational algorithms, the software itself, as well as scripts and computational workflows that were **created during the research process or for a research purpose**, across all domains of research.



# Setting up a (research) code project



DIGITAL RESEARCH  
ACADEMY

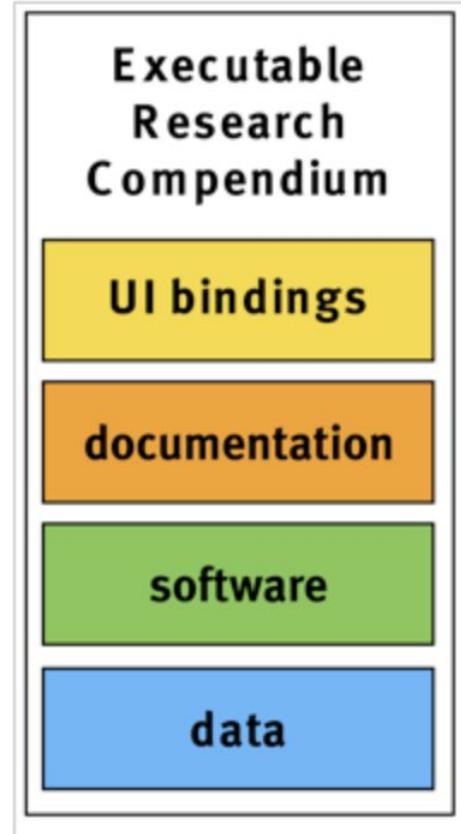
# The Research compendium



This image was created by [Scriberia](#) for The Turing Way community and is used under a CC-BY licence ([DOI 10.5281/zenodo.3332807](https://doi.org/10.5281/zenodo.3332807)).

# The Research Compendium

- A research compendium is a collection of all digital parts of a research project including data, code, texts (protocols, reports, questionnaires, meta data). The collection is created in such a way that reproducing all results is straightforward.
- A research compendium accompanies, enhances, or is a scientific publication providing data, code, and documentation for reproducing a scientific workflow.



Nuest, D., Konkol, M., Pebesma, E., Kray, C., Schutzeichel, M., Przibytzin, H., & Lorenz, J. (2017). Opening the Publication Process with Executable Research Compendia. *D-Lib Magazine*, 23(1/2).  
[10.1045/january2017-nuest](https://doi.org/10.1045/january2017-nuest)

# The research compendium/the repro pub

Published July 15, 2019 | Version v1

Conference paper

 Open

The ReproPub: A hybrid research object for supporting publication-level re-execution and generalization of neuroimaging research findings

Kennedy, David N<sup>1</sup> 

Show affiliations

- Idea: create a workflow that re-creates a publication from raw data to pdf

# Neurolibre

- FULLY reproducible paper
- Living preprint: reviewer comments are visible; interactive document

A reproducible benchmark of resting-state fMRI denoising strategies  
using fMRIPrep and Nilearn

Jupyter Notebook | Python | Submitted 13 May 2023 • Published 19 June 2023

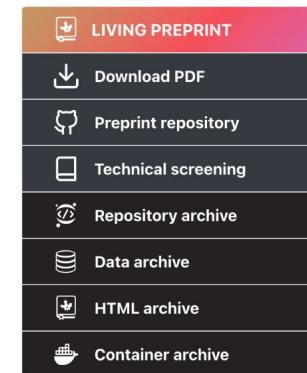
The screenshot shows a Jupyter Notebook interface for a preprint titled "A reproducible benchmark of resting-state fMRI denoising strategies using fMRIPrep and Nilearn". The authors listed are Hao-Ting Wang, Steven L Meisler, Hanad Sharmarke, Natasha Clarke, Francois Paugam, Nicolas Gensollen, Christopher J Markiewicz, Bertrand Thirion, and Pierre Bellec. The page includes links for "Download PDF", "Preprint repository", "Technical screening", "Repository archive", "Data archive", "HTML archive", and "Container archive". A red arrow at the bottom left points towards the screenshot.

NeuroLibre  
Reproducible Preprints

A reproducible benchmark of resting-state fMRI denoising strategies using fMRIPrep and Nilearn

Hao-Ting Wang<sup>1,7</sup>, Steven L Meisler<sup>2,3</sup>, Hanad Sharmarke<sup>1</sup>, Natasha Clarke<sup>1</sup>, Francois Paugam<sup>4</sup>, Nicolas Gensollen<sup>5</sup>, Christopher J Markiewicz<sup>6</sup>, Bertrand Thirion<sup>5</sup>, and Pierre Bellec<sup>1,7</sup>

DOI: [10.55458/neurolibre.00012](https://doi.org/10.55458/neurolibre.00012)  
Reproducible Preprint



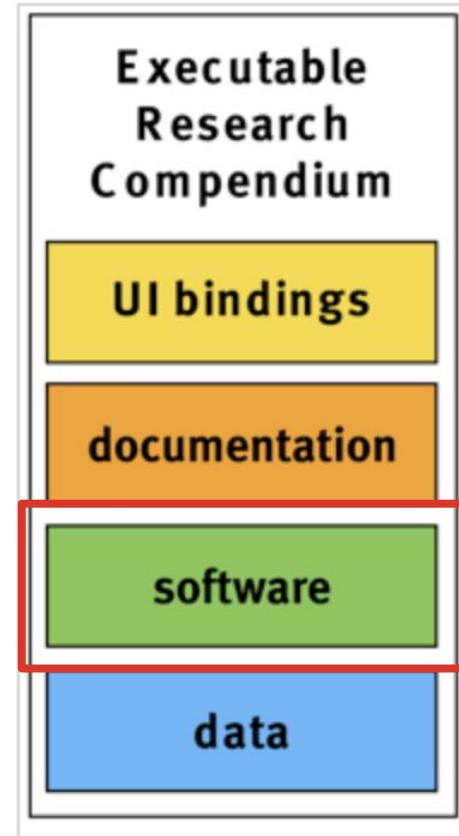
overview

# The Research Compendium

A very comprehensive website on research compendia can be found [here](#)

An example template to document your research can be found [here](#)

Reproducible science includes a good organisation of the software/code



# Setting up a code project - as part of a research compendium

## The folder structure

```
compendium/
  └── data
      └── my_data.csv
  └── analysis
      └── my_script.R
  └── DESCRIPTION
  └── README.md
```

project folder  
structure

```
compendium/
  └── CITATION
  └── code
      └── analyse_data.R
      └── clean_data.R
  └── data_clean
      └── data_clean.csv
  └── data_raw
      └── datapackage.json
      └── data_raw.csv
  └── Dockerfile
  └── figures
      └── flow_chart.jpeg
  └── LICENSE
  └── Makefile
  └── paper.Rmd
  └── README.md
```

# Setting up a code project

```
my-research-project/
    ├── .vscode/          # vs code settings
    │   └── launch.json    # Cleaned, derived data
    │       └── settings.json # Data documentation
    ├── src/              # Code (modules, scripts)
    ├── notebooks/        # Jupyter or MATLAB Live Scripts
    ├── data/
    │   ├── raw/           # Untouched raw data
    │   ├── processed/     # Cleaned, derived data
    │   └── README.md      # Data documentation
    ├── results/
    │   ├── figures/       # Plots/images generated
    │   └── tables/        # CSV, LaTeX tables, etc.
    └── paper/
        ├── manuscript.md  # Or .tex, .docx, etc.
        ├── bibliography.bib # References
        └── figures/         # Final figures for the paper
    ├── tests/             # Automated tests
    ├── requirements.txt   # or environment.yml / setup.py
    ├── Makefile           # Automate workflows (optional)
    ├── LICENSE
    ├── README.md          # What is your project about?
    └── .gitignore
```

project folder  
structure

# Cookiecutter project template

- Code and data version for good file organisation

project folder  
structure

```
johannabayer@Hanna my_project % tree
.
├── LICENSE
├── Makefile
├── README.md
├── data
│   ├── external
│   ├── interim
│   ├── processed
│   └── raw
├── docs
├── models
└── my_project
    ├── __init__.py
    ├── config.py  fix typo (#391)
    ├── dataset.py
    ├── features.py
    ├── modeling Remove config import from ini
    │   ├── __init__.py
    │   ├── predict.py
    │   └── train.py
    └── plots.py
├── notebooks
├── pyproject.toml
├── references
├── reports Start to flesh out READMEs
│   └── figures
├── requirements.txt
└── setup.cfg
Version 2 (#246)
```

# Project folder structure

- Cookie cutter project templates
- Cookie cutter data science (ccds)
  - Python-based project structure

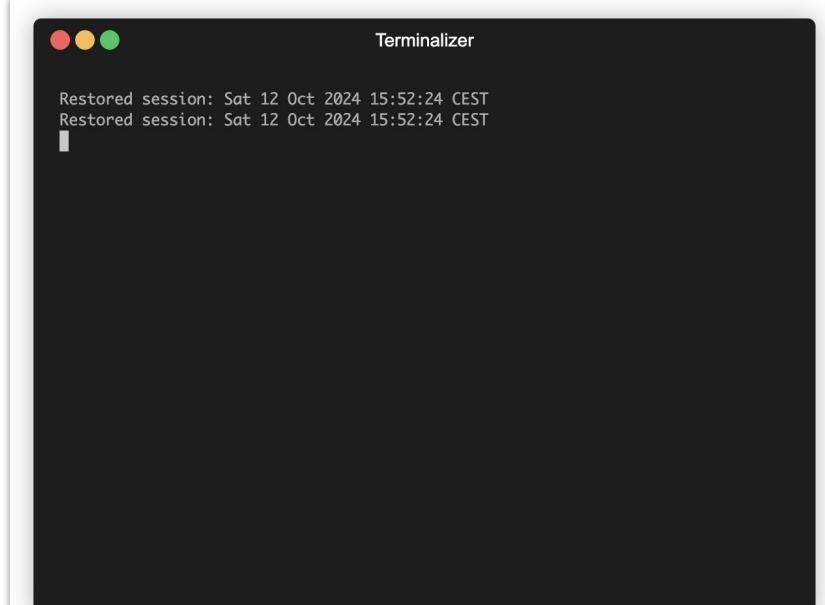
project folder  
structure

<https://cookiecutter.readthedocs.io/en/stable/>

[Home](#) / [Cookiecutter: Better Project Templates](#) [Edit on GitHub](#)

## Cookiecutter: Better Project Templates

Cookiecutter creates projects from [cookiecutters](#) (project templates), e.g. Python package projects from Python package templates.



Restored session: Sat 12 Oct 2024 15:52:24 CEST  
Restored session: Sat 12 Oct 2024 15:52:24 CEST

# Setting up a coding project - naming files

Files need to be

- Human readable
- Machine readable
- Consistent
- Descriptive
- Optional: Play with default ordering (i/e start your file with creation date  
YYYY-MM-DD).

# Naming files

(BAD) Examples:

- Myabstract.docx
- Joe's Filenames Use Spaces and Punctuation.xlsx
- figure 1.png
- fig 2.png
- JW7d^(2sl@deletethisandyourcareerisoverWx2\*.txt

 FINAL\_PACE400 with PNS variables v2 (brief PNS dataset\_PAS\_LatePace\_fulldates\_CAARMSPACE400\_oct2019\_original.sav



mergedPACE400 + LatePACE\_original.sav

# Naming files

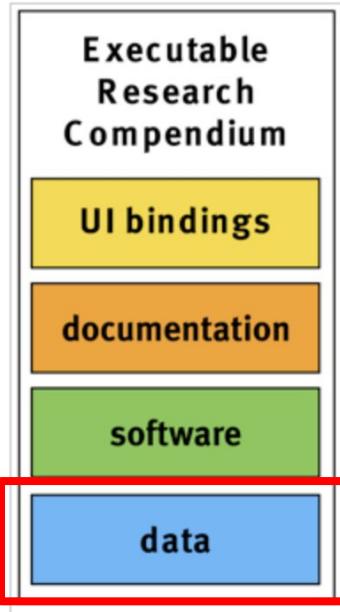
## Better examples

- YYYYMMDD\_OHBM\_abstract.txt
- Step\_1\_step\_description
- Step\_2\_step\_description
- Unique filenames:
  - Sub\_1
    - sub\_1\_fileA.txt
    - Sub\_1\_fileB.txt
  - Sub\_2
    - sub\_2\_fileA.txt
    - Sub\_2\_fileB.txt

## Follow the software example: versioning

- Generic version is the CURRENT/IMPORTANT version (eg. OHBM\_abstractc.txt)
- All older/other versions are versioned (e.g OHBM\_abstract\_Sept\_2024.txt)
- Similar to JAVA/Git - > main method/master is the one that compiles, not main\_2025, even if that's the latest version.

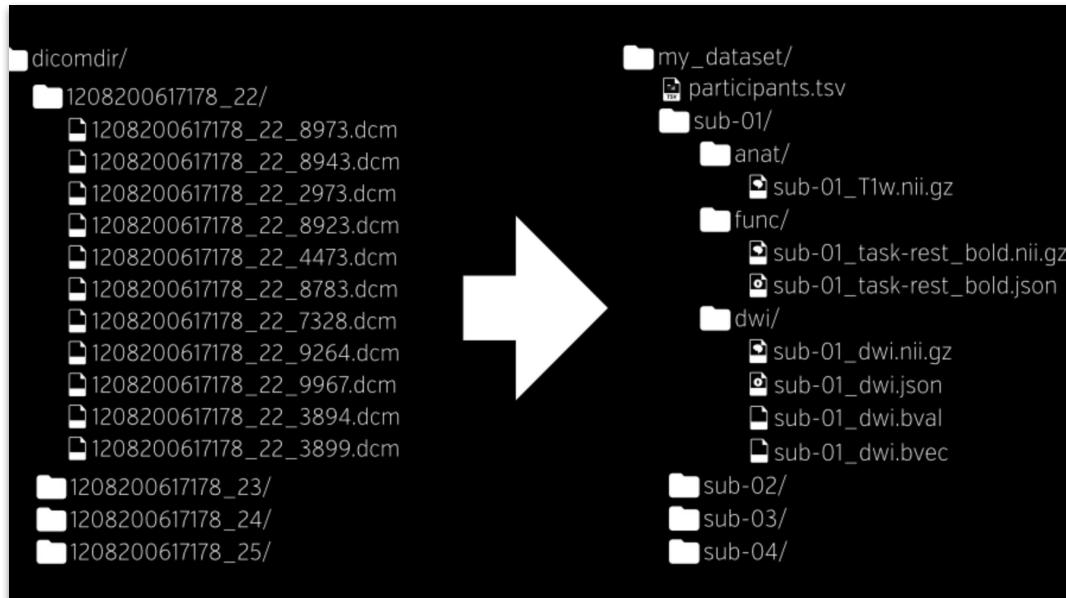
# Data organisation - does your field have a file standard?



```
my-research-project/
  └── .vscode/          # vs code settings
    └── launch.json      # Cleaned, derived data
    └── settings.json    # Data documentation
  └── src/              # Code (modules, scripts)
  └── notebooks/        # Jupyter or MATLAB Live Scripts
  └── data/             # Untouched raw data
    └── raw/             # Cleaned, derived data
    └── processed/       # Data documentation
    └── README.md
  └── results/
    └── figures/         # Plots/images generated
    └── tables/          # CSV, LaTeX tables, etc.
  └── paper/
    └── manuscript.md   # Or .tex, .docx, etc.
    └── bibliography.bib # References
    └── figures/         # Final figures for the paper
  └── tests/            # Automated tests
  └── requirements.txt  # or environment.yml / setup.py
  └── Makefile          # Automate workflows (optional)
  └── LICENSE
  └── README.md         # What is your project about?
  .gitignore
```

# Does your field have a file standard?

## Brain Imaging Derived Structures



# File organisation: (stealing from the) BIDS standard

## Brain Imaging Derived Structures

- modularizes data
- names files in a human AND machine friendly way
- specifies a folder structure
- uses standard interoperable file formats
- documents metadata
- minimize duplication



# File organisation: (stealing from the) BIDS standard

Brain Imaging Derived Structures

Key1-value\_key2-value\_suffix.extension

sub-001\_population-cross\_mri.nii

- **Suffix** preceded by an **underscore**
- Entities are composed of **key-label** pairs separated by **underscores**
- **Key** and **label** separated by **hyphen**
- **Keys, labels, suffixes** can only contain letters and / or numbers.
- Key-value combination can be repetitions from higher order folders to form unique filenames

# Adding a Readme.md

- Entry point/first item that visitor reads
- Follows Markdown format
- There are several (slightly differing) markdown formats, but we are going to focus on git markdown.

Readme.md

```
my-research-project/
└── .vscode/      # vs code settings
    ├── launch.json   # Cleaned, derived data
    └── settings.json # Data documentation
── src/          # Code (modules, scripts)
── notebooks/    # Jupyter or MATLAB Live Scripts
── data/
    ├── raw/         # Untouched raw data
    ├── processed/   # Cleaned, derived data
    └── README.md    # Data documentation
── results/
    ├── figures/     # Plots/images generated
    └── tables/      # CSV, LaTeX tables, etc.
── paper/
    ├── manuscript.md # Or .tex, .docx, etc.
    ├── bibliography.bib # References
    └── figures/      # Final figures for the paper
── tests/
── environment.yml # or requirements.txt / setup.py
── Makefile        # Automate workflows (optional)
── LICENSE
── README.md       # What is your project about?
── .gitignore
```

# Readme.md

Files ending in .md indicate the Markdown style. We are going to follow [Github Markdown](#).

A Readme.md should contain:

- Title and description of the project
- Author of the project and contact details
- Information about the license of the project - can everyone reuse your code?
- Structure of the project
- Anything that a user of the project should know

Examples: [pydantic](#)

# Thinking about the writing early on

```
my-research-project/
    └── .vscode/      # vs code settings
        ├── launch.json    # Cleaned, derived data
        └── settings.json   # Data documentation
    └── src/          # Code (modules, scripts)
    └── notebooks/     # Jupyter or MATLAB Live Scripts
    └── data/
        └── raw/          # Untouched raw data
        └── processed/     # Cleaned, derived data
        └── README.md      # Data documentation
    └── results/
        └── figures/       # Plots/images generated
        └── tables/         # CSV, LaTeX tables, etc.
    └── paper/
        ├── manuscript.md  # Or .tex, .docx, etc.
        ├── bibliography.bib # References
        └── figures/        # Final figures for the paper
    └── tests/          # Automated tests
    └── environment.yml  # or requirements.txt / setup.py
    └── Makefile         # Automate workflows (optional)
    └── LICENSE
    └── README.md       # What is your project about?
    └── .gitignore
```

# The repro pub - text editor and reference management system (based on reproducibility criteria)

Text editor	pro	con
MS Word/Google docs	<ul style="list-style-type: none"><li>- Plugins for most reference managers</li><li>- good (?) platform for getting feedback/review</li><li>- (Version controlled)</li></ul>	<ul style="list-style-type: none"><li>- Figures need to be imported each time</li></ul>
Markdown	<ul style="list-style-type: none"><li>-Integration with R (markdown, knitr)</li><li>-In theory, fully reproducible (code and paper)</li></ul>	<ul style="list-style-type: none"><li>- No review option</li></ul>
Overleaf (online)	<ul style="list-style-type: none"><li>- Bibtex/Latex</li><li>- Pro version has direct git and Dropbox link</li><li>- Math support, templates</li><li>- review option</li></ul>	<ul style="list-style-type: none"><li>- steeper learning curve</li></ul>
LaTeX (Desktop)	<ul style="list-style-type: none"><li>- Free</li><li>- Bibtex file</li><li>- Entirely version controllable</li><li>- Math support</li><li>- Create a figure folder that you save your figs in</li></ul>	<ul style="list-style-type: none"><li>-Very steep learning curve</li><li>-No review option</li></ul>

# Collaborative publishing using overleaf



- All benefits of pure latex but
  - Working on a document together & review mode (suggest changes)
  - Import citations from paperpile, zotero, mendeley and endnote
  - Templates for publications, journals, conference posters etc
  - Link an image folder (i.e in Dropbox)
  - Git integration

## Downside

- Internet dependency (except offline mode)

# Your code editor

```
my-research-project/
├── .vscode/      # vs code settings
│   ├── launch.json    # Cleaned, derived data
│   └── settings.json  # Data documentation
├── src/          # Code (modules, scripts)
├── notebooks/    # Jupyter or MATLAB Live Scripts
├── data/
│   ├── raw/         # Untouched raw data
│   ├── processed/   # Cleaned, derived data
│   └── README.md    # Data documentation
├── results/
│   ├── figures/     # Plots/images generated
│   └── tables/      # CSV, LaTeX tables, etc.
└── paper/
    ├── manuscript.md  # Or .tex, .docx, etc.
    ├── bibliography.bib # References
    └── figures/        # Final figures for the paper
├── tests/         # Automated tests
├── environment.yml # or requirements.txt / setup.py
├── Makefile        # Automate workflows (optional)
├── LICENSE
├── README.md       # What is your project about?
└── .gitignore
```

# Setting up your code editor

- VS code
  - Get the extensions for you code base

## Important files:

- Settings.json
  - Sets the behaviour of your vscode (linting, formatting etc)
- Launch.json
  - Specifies the behaviour of vs code running your programs
    - Debugging, which interpreter, which language

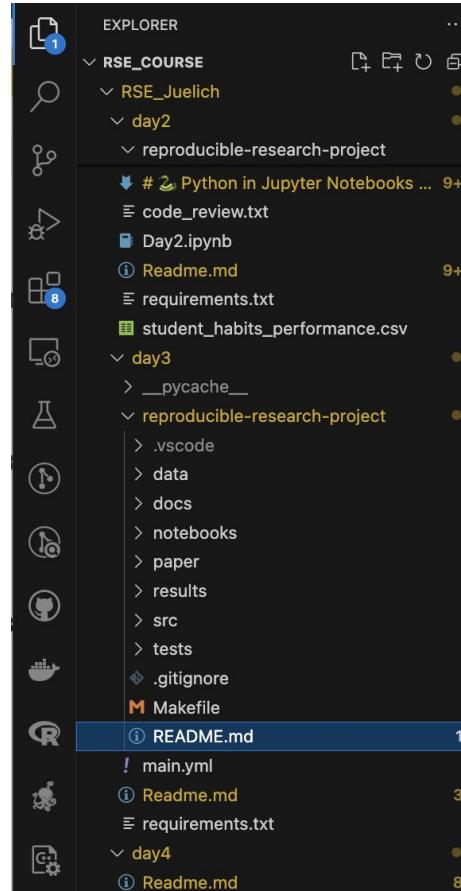
The screenshot shows the VS Code interface. The Explorer sidebar on the left displays a file tree for a project named 'RSE\_JUELICH'. It includes a '.vscode' folder containing 'settings.json', 'main.py', 'add.py', and 'test\_add.py'. Below this are 'day1', 'day2', 'day3', and a 'myproject' folder containing 'app', 'build', 'src', 'test', and 'fpm.toml'. The 'test' folder contains 'README.md', '.gitignore', and 'Readme.md'. The Editor pane on the right shows the contents of 'settings.json'. The JSON file defines testing arguments for Python, specifying command-line options like '-v', '-s', '-p', and test patterns, and enabling/unenabling different testing frameworks.

```
1  {
2      "python.testing.unittestArgs": [
3          "-v",
4          "-s",
5          "./day3",
6          "-p",
7          "test_*.py"
8      ],
9      "python.testing.pytestEnabled": false,
10     "python.testing.unittestEnabled": true
11 }
```

# Important VS code extensions

Extensions for your programming language;

- Python, Julia, etc IDE.
- GitLens
- (Docker)
- Jupyter notebook extension



# settings.json

**Purpose:** Configure **editor behavior, extensions, and general environment settings**

**Scope:** User-wide or workspace-specific

**Examples:**

- Tab size, font, auto-save, linting rules
- Default Python interpreter or Fortran compiler path
- Format-on-save, excluded files, theme

```
ielich > .vscode > {} settings.json > ...
{
  "python.testing.unittestArgs": [
    "-v",
    "-s",
    "./day3",
    "-p",
    "test_*.py"
  ],
  "python.testing.pytestEnabled": false,
  "python.testing.unittestEnabled": true,
  "python.linting.enabled": true,
  "editor.tabSize": 4,
  "files.exclude": {
    "**/__pycache__": true
  },
  "editor.formatOnSave": true
}
```

# launch.json

Specifies:

- Which script or executable to run
- What **command-line arguments** to pass
- Whether to **launch** a new process or **attach** to an existing one
- Where to show the output (terminal or debug console)
- Environment variables, working directory, debugging behaviour

```
luelich > .vscode > {} launch.json > ...
{
  "name": "Run analysis script",
  "type": "python",
  "request": "launch",
  "program": "${workspaceFolder}/day3/analysis.py",
  "console": "integratedTerminal",
  "args": ["--input", "data.csv", "--verbose"]
}
```

# Jupyter notebooks

- Can be installed via `pip install notebook` and then launched via `jupyter notebook notebook.ipynb` or `jupyter notebook`
- Allows to run notebooks in the browser
- Not super great for large research projects, but:
  - Operating system agnostic
  - Can be shared and opened in the browser via Google Colab
  - Can be run via vscode
  - A variety of kernels available

For research:

- Better: <https://marimo.io/>

Code editor



# Matlab environment

## Matlab projects

Use MATLAB Projects to manage files, paths, and settings.

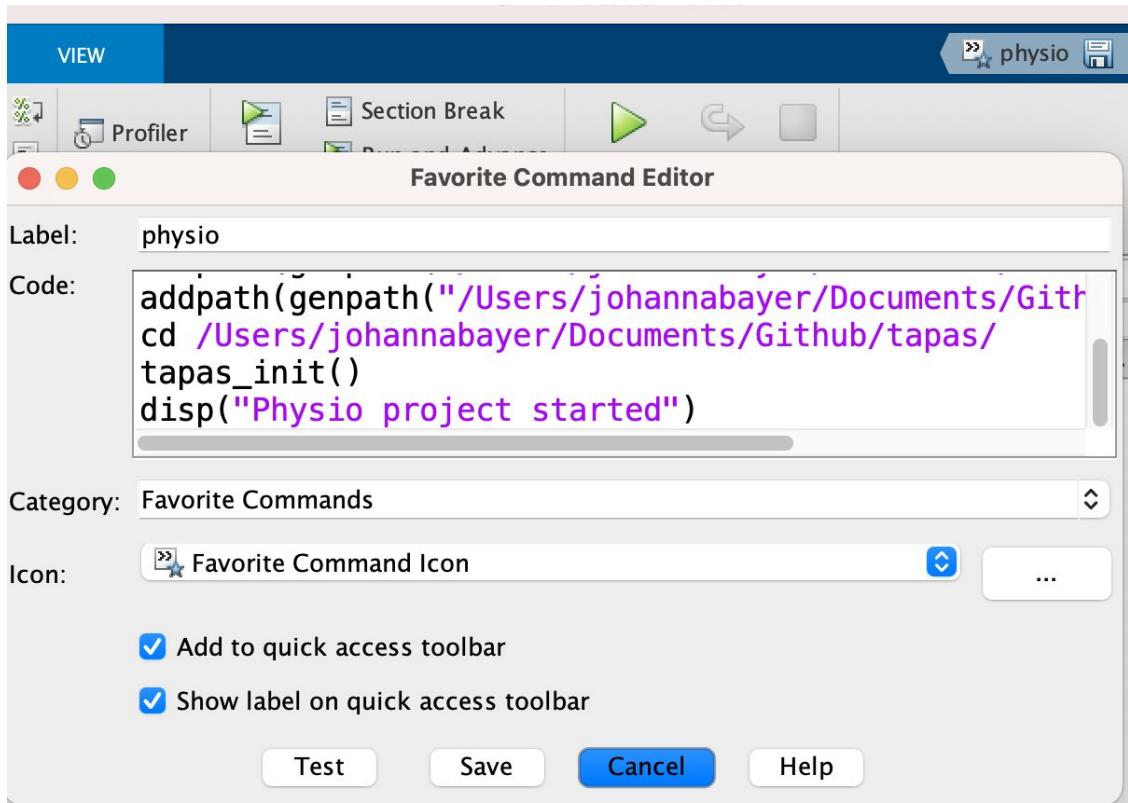
- Go to **Home > New > Project > From Folder**
- Choose your folder (e.g., your project-name/ directory)
- MATLAB creates a .prj file to manage paths, startup code, etc.
- 

## Configure startup paths:

- Add a **startup.m** file to automatically configure your environment on project open:

```
% startup.m
addpath(genpath('src'));
addpath('scripts');
disp('Project environment initialized.');
```

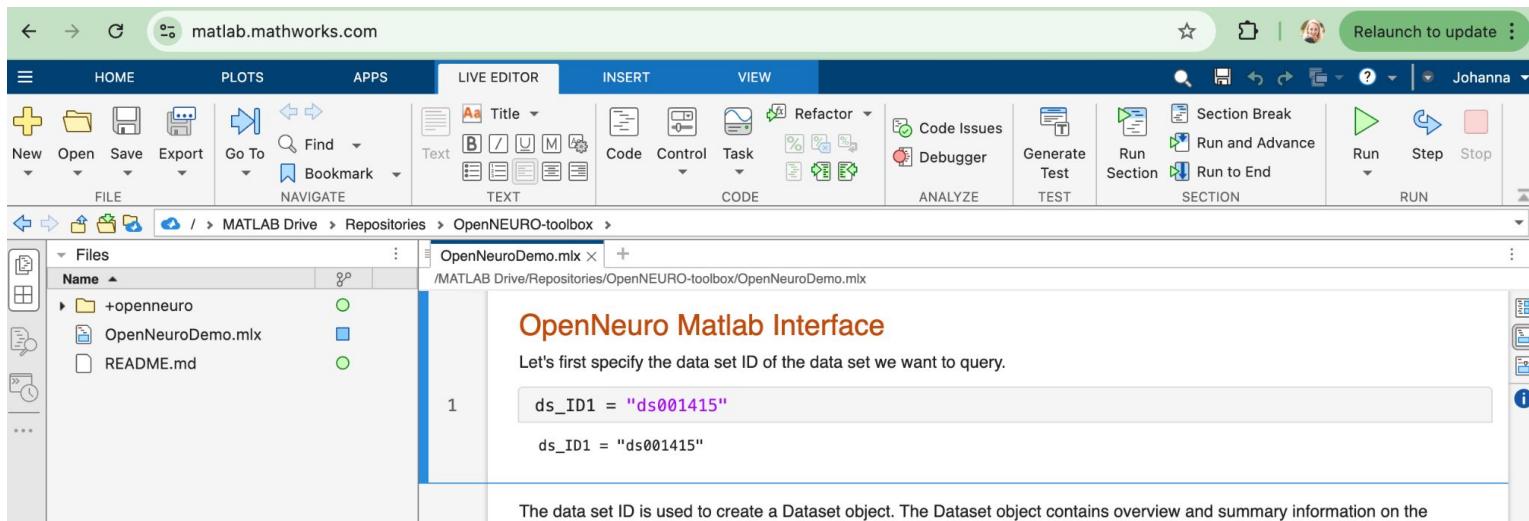
# Set favourite commands



Code editor

# Matlab Live Scripts

- Cell based matlab scripts (.mlx)
- Can be opened in the browser (Matlab online)
  - Allows to share and execute code
  - Still license needed



# Set up how to compile/run your code

Die Qual der Wahl ....

- Run via VS code
- Run via the command line/terminal
- Run via Jupyter notebook
- Run via a (other) IDE
- Run on a cluster/cloud ...

# Setting up a coding project on a cluster

- Rsync of folders between cluster and local (images, tables etc)
- Ssh into cluster via vscode
- Sbatch

# More resources on Research Compendia

- <https://github.com/topics/research-compendium/>
- <https://the-turing-way.netlify.app/reproducible-research/compendia>
- <https://research-compendium.science/>
- [https://selinazitrone.github.io/YoMos2020/slidy\\_presentation.html#\(1\)](https://selinazitrone.github.io/YoMos2020/slidy_presentation.html#(1))

```
johannabayer@Hanna my_project % tree
.
├── LICENSE
├── Makefile
├── README.md
├── data
│   ├── external
│   ├── interim
│   ├── processed
│   └── raw
├── docs
├── models
└── my_project
    ├── __init__.py
    ├── config.py
    ├── dataset.py
    ├── features.py
    ├── modeling
    │   ├── __init__.py
    │   ├── predict.py
    │   └── train.py
    └── plots.py
├── notebooks
├── pyproject.toml
├── references
├── reports
│   └── figures
├── requirements.txt
└── setup.cfg

2 directories, 14 files
```

15 min break

# (Software) version control



DIGITAL RESEARCH  
ACADEMY

# Version control

## Git and Github

GitHub is a web-based platform designed for version control, collaboration, and project management. It uses **Git**, a distributed version control system, to enable developers, researchers, and teams to track changes in code, manage projects, and collaborate on software and other digital projects.

GitHub is widely used in software development, open-source projects, research, and even non-code workflows due to its robust collaboration tools and integration capabilities.

Github's main purpose:

- Version control
- Collaboration



History of version  
control

# Git Github Gitlab and Co



Git: the Software:

Git is a distributed version control system (VCS) designed to track changes in files and coordinate work among multiple people. It allows developers to work on projects locally, create branches for experimentation, and merge changes back into the main codebase. Git is primarily a command-line tool, and it provides a set of commands for managing repositories and collaborating with others.



Github: the Platform:

GitHub is a web-based platform built around Git that provides a centralized hosting service for Git repositories. It adds a graphical user interface (GUI) on top of Git, making it easier to perform common Git operations like creating repositories, managing branches, and collaborating with others. GitHub also offers additional features such as issue tracking, pull requests, code review tools, and project management capabilities. It's widely used for open-source projects, personal projects, and team collaboration.



Gitlab: a different version

GitLab is another web-based platform that provides hosting for Git repositories, similar to GitHub. It offers similar features such as repository management, issue tracking, and pull requests. However, the focus is more on DevOps. It integrates Git repository hosting with CI/CD pipelines, project management, and other DevOps tools. GitLab also allows to host independent servers.

# A history of version control

Pre-version control (-1990)

## File based systems

### Features

- Versions of files were kept individually

### Challenges:

- Obvious

A STORY TOLD IN FILE NAMES:				
Filename	Date Modified	Size	Type	
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file	
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file	
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file	
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file	
data_2010.05.28_huh???.dat	7:20 PM 5/28/2010	30 KB	DAT file	
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file	
data_2010.05.29_aaarrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file	
data_2010.05.29_#\$@*!&!.dat	2:40 AM 5/29/2010	0 KB	DAT file	
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file	
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file	
data_2010.05.29_woohoo!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file	
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file	
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file	
ThesisOutline.doc	7:26 AM 5/29/2010	38 KB	DOC file	
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file	
JUNK...	2:45 PM 5/29/2010		Folder	
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file	

# A history of version control

## Centralized Version Control Systems (CVCS) (1990s)

### Features:

- Centralized repositories to coordinate changes between team members.
- Developers checked out files to work on them and checked them back into the central server.

### Challenges:

- Reliance on central server
- No ability to work offline

### Examples:

- Concurrent Versions System (CVS) and Subversion (SVN)

# A history of version control

## Distributed Version Control Systems (DVCS) (2005–Present)

### Features:

- Every developer has a full copy of the repository, including history.
- Allows offline work and faster operations.
- Resolves many limitations of CVCS.

### Examples:

- Git
- Mercurial
- Bazaar (Ubuntu)

# A history of version control

## **Web-Based, collaborative Version Control Systems:**

With the rise of the internet and collaborative development, web-based version control systems emerged. They provided a platform for hosting repositories and added web interfaces to simplify collaboration and code management. GitHub (2008), GitLab (2011), and Bitbucket (2008) are popular examples of such platforms.

### **Examples:**

- Github
- Gitlab
- Bitbucket

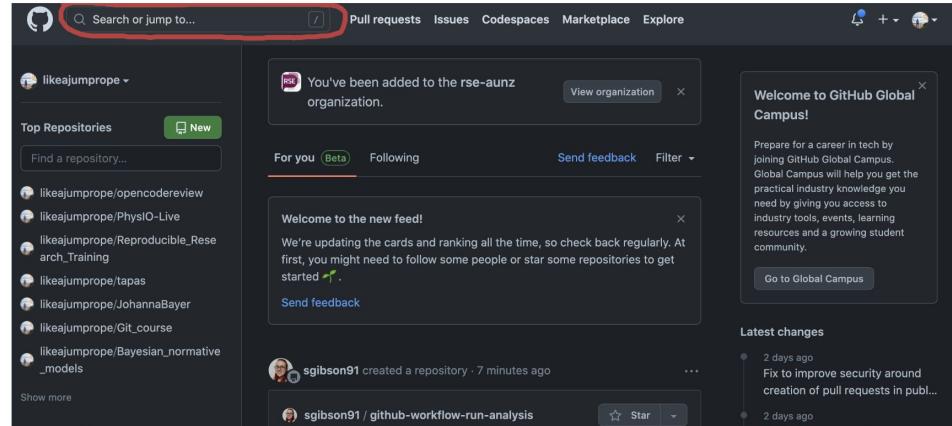
# Version control using Github/Gitlab



Community, T. T. W., & Scriberia. (2020). Illustrations from the Turing Way book dashes.  
Zenodo. [10.5281/ZENODO.3695300](https://zenodo.org/record/3695300)

# Getting to know the Github interface

1. Log into your github account
2. The login page will get you to your dashboard. Here you find:
  - a. An overview of your dashboard
  - b. Your profile
  - c. An overview of your repositories



# Getting to know the Github interface

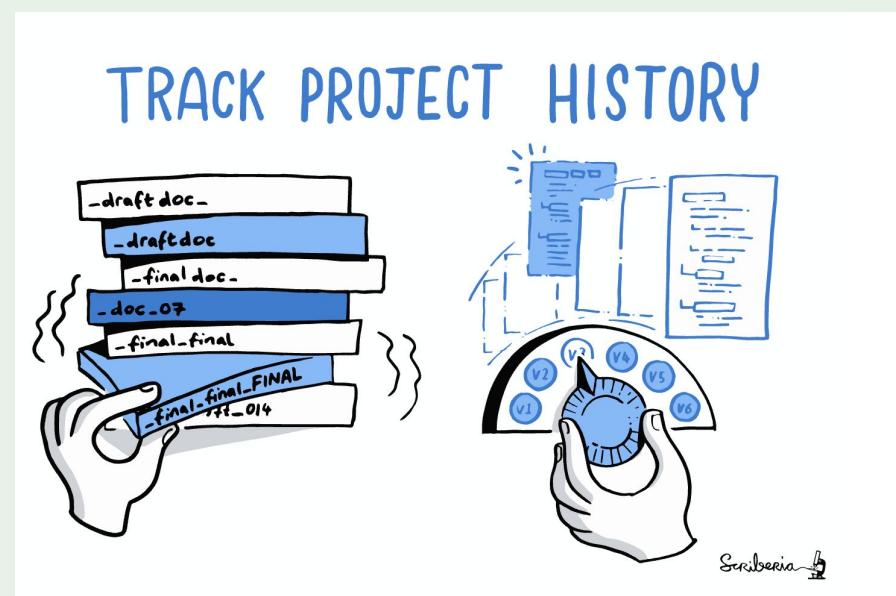
Repositories on Github are divided into:

- Those that you have write access to
- Those you can only view (the great majority)

You can use the search function to query all public repositories on Github

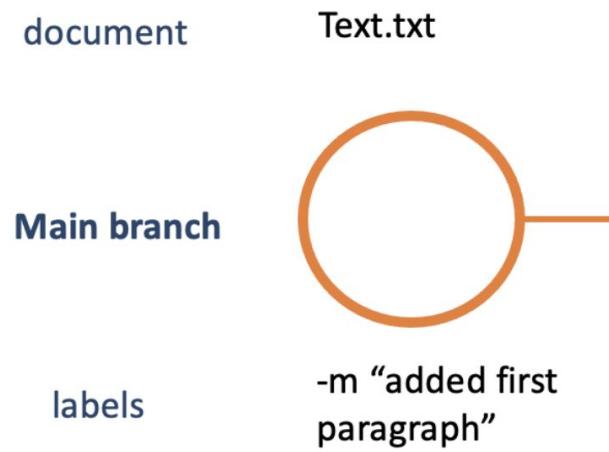
The screenshot shows the GitHub search interface with the query "Open science". The sidebar on the left includes filters for Code (6.4M), Repositories (3.7k), Issues (55k), Pull requests (38k), Discussions (2k), Users (3k), and More. Below that is a list of languages: Jupyter Notebook, Python, HTML, JavaScript, R, TeX, and Java. The main search results page displays 3.7k results from 293 ms. The first result is "datasciencemasters/go" by "The Open Source Data Science Masters", which has 23.8k stars and was updated on 19 Aug. The second result is "jonathan-bower/DataScienceResources" by "Open Source Data Science Resources.", which has 3.9k stars and was updated on 15 Aug. The third result is "CenterForOpenScience/osf.io" by "Facilitating Open Science", which has 652 stars and was updated 20 hours ago. This repository uses Python, as indicated by the highlighted tag in the screenshot. The fourth result is "DataScienceResearchPeru/OpenSource-RoadMap-DataScience" by "¡Camino a una educación autodidacta en Ciencia de Datos!", which has 0 stars and was last updated on 19 Aug. It uses data-science, deep-learning, and dsrp tags.

# Committing



# What is a commit?

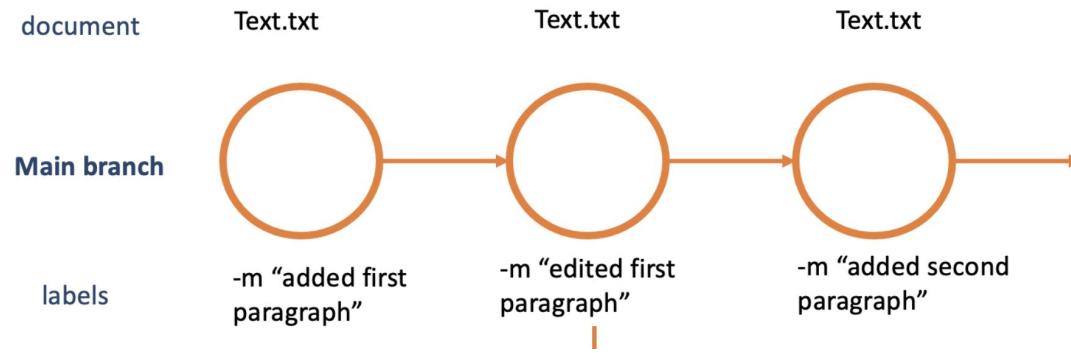
- **Commits:** annotated snapshot of changes in a document



Commits

# What is a commit?

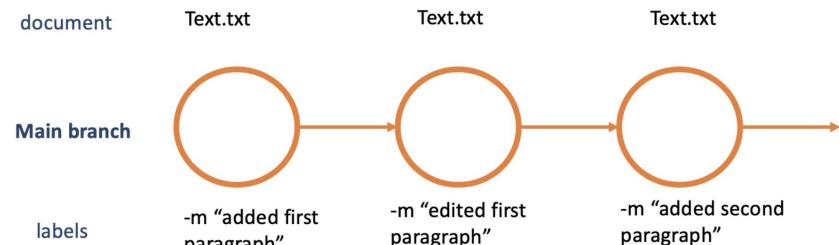
- **Commits:** annotated snapshot of changes in a document



Commits

# Git and Github: tracking your files, scripts and code

- Version control for files and scripts
- Ability to revert back to a previous commit



A screenshot of a GitHub repository interface showing a list of commits. The commits are listed in chronological order from top to bottom:

- Commits on Nov 8, 2024:
  - add GINI impurity**  
likeajumprope committed 3 weeks ago
- Commits on Nov 7, 2024:
  - more xgboost**  
likeajumprope committed 3 weeks ago
  - Update GPU usage**  
likeajumprope committed 3 weeks ago

At the bottom left, there is a red arrow pointing to the word 'Commits'.

# Git and Github: tracking your files, scripts and code

- ## - Commit history

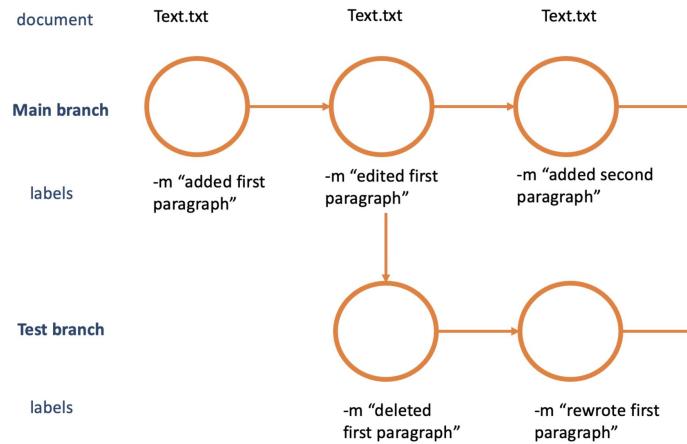


## Commits

# Git and Github: tracking your files, scripts and code

## Collaboration (github online)

- **Branch:** Diversion point of repository and history

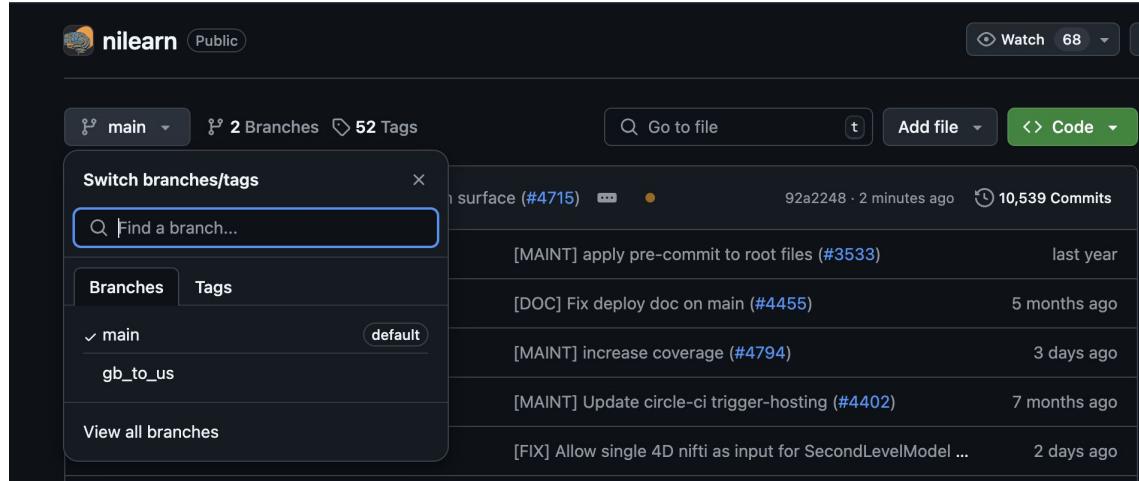


Branched

# Git and Github: tracking your files, scripts and code

## Collaboration (github online)

- **Branch:** Diversion point of repository and history

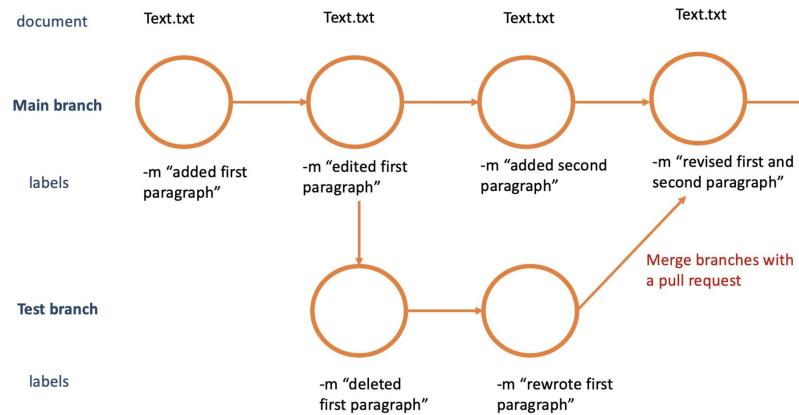


Branches

# Git and Github: tracking your files, scripts and code

## Collaboration (github online)

- **merging:** Unifying divergent branches, integrating changes
- Merging between branches is performed using a **pull request**.

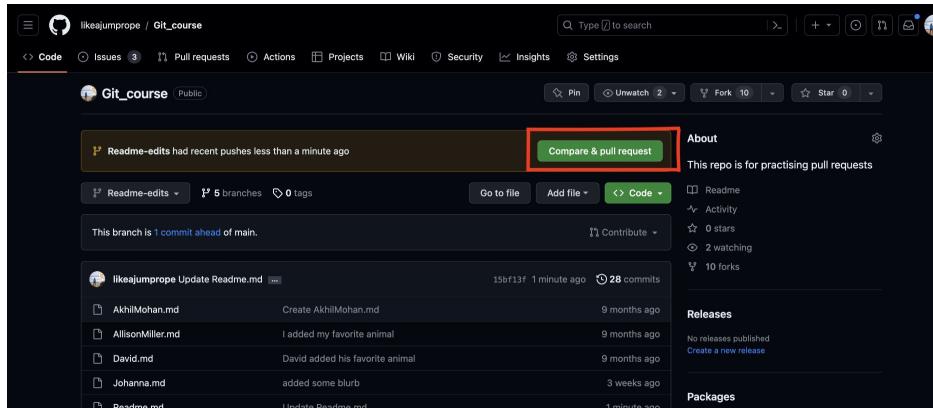


Merging

# Git and Github: tracking your files, scripts and code

## Collaboration (github online)

- **merging:** Unifying divergent branches, integrating changes
- Merging between branches is performed using a **pull request**.



Merging

# Git and Github: tracking your files, scripts and code

## Collaboration (github online)

- **Fork:** make a copy of someone else's repository under your account.
- Repositories on Github are divided into two categories:
  - Yours (you have writing permission)
  - Other people's (you don't)
- **Forking** creates a copy of someone else's repository that you own (aka you have writing permission)



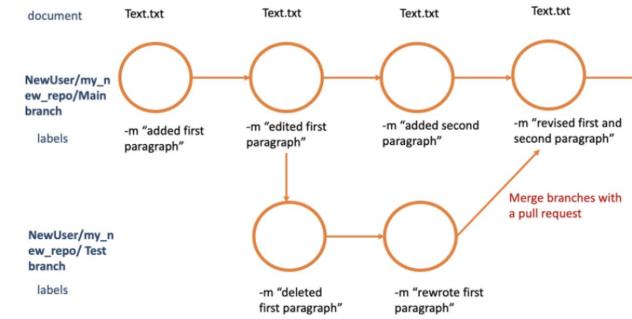
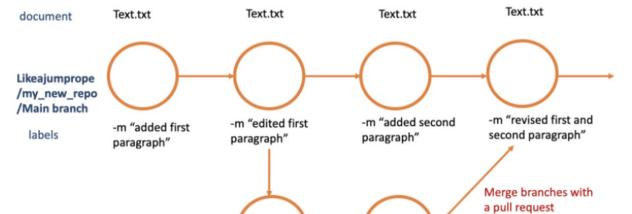
Forking

# Git and Github: tracking your files, scripts and code

## Collaboration (github online)

-

Forking



Forking

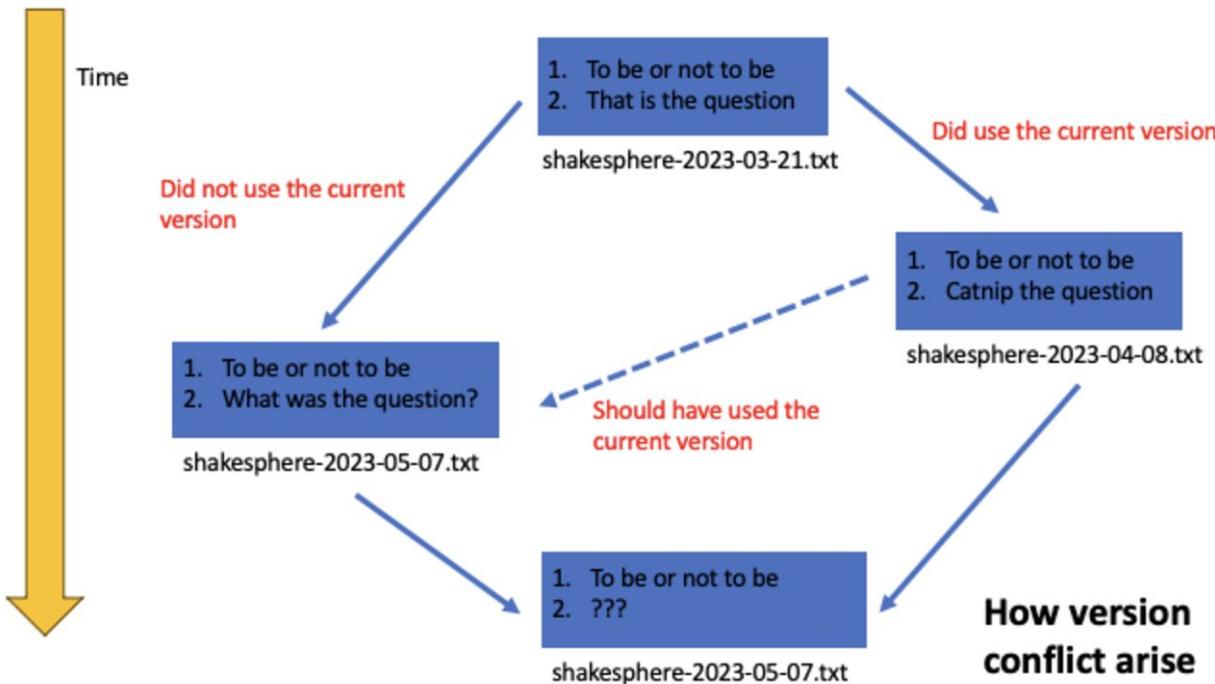
# Why is it called pull request?

- Remember the separation between your and other people's repositories.
- In order to get your changes into someone else's repository, you have to request for them to be pulled in. This action is not up to you.
- You request a pull.
- The other person accepts, and pulls.



Fig. 191 Making your first pull request on GitHub. The Turing Way project illustration by Scriberia. Used under a CC-BY 4.0 licence. DOI: 10.5281/zenodo.3332807.

# Merge conflicts



Merge conflicts

# Resolving a merge conflict

Update merge\_conflicts.md #19

Resolving conflicts between dev and main and committing changes → dev

1 conflicting file	merge_conflicts.md
 merge_conflicts.md merge_conflicts.md	1 Practising merge conflicts. 2 <<<<< dev 3 Bad idea. 4 ===== 5 Creating a merge conflict. 6 >>>>> main 7

Merge conflicts

# Issues on Github

**Issues** on GitHub are a built-in tool for tracking tasks, enhancements, bugs, and other project-related activities. They provide a way for developers, collaborators, and even external users to communicate and manage the work associated with a GitHub repository.

The screenshot shows the GitHub Issues page with a dark theme. At the top, there is a navigation bar with links: Code, Issues (which is underlined), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, a modal window titled "Label issues and pull requests for new contributors" is displayed. It contains the text: "Now, GitHub will help potential first-time contributors [discover issues](#) labeled with `good first issue`". There is a "Dismiss" button in the top right corner of the modal. Below the modal, there is a search bar with the query "is:issue is:open" and a "Filters" dropdown. To the right of the search bar are buttons for "Labels 9", "Milestones 0", and a green "New issue" button. At the bottom of the page, there is a large red arrow pointing to the left with the word "Issues" in white text.

Issues

# Issues and collaboration on Github online

Issues are the way of communicating on Github

- Issues follow markdown format
- PRs can be referenced in Issues (and vice versa)
- Usually one PR responds to an issue

The screenshot shows a GitHub issue page for a repository named 'Links'. The issue is titled '#2' and is labeled as 'Closed'. The issue was opened by a user named 'likeajumprope' 3 minutes ago. The issue body contains a link to 'nilearn/nilearn#5342'. There is a button to 'Create sub-issue'. Below the issue, a comment from 'likeajumprope' indicates they closed the issue as 'completed' now. The right side of the screen displays settings for assignees, labels, projects, and milestones, all currently set to 'No one', 'No labels', 'No projects', and 'No milestone' respectively. At the bottom, there is a red arrow pointing to the word 'Issues'.

Issues

# Activity 1 (20')

1.

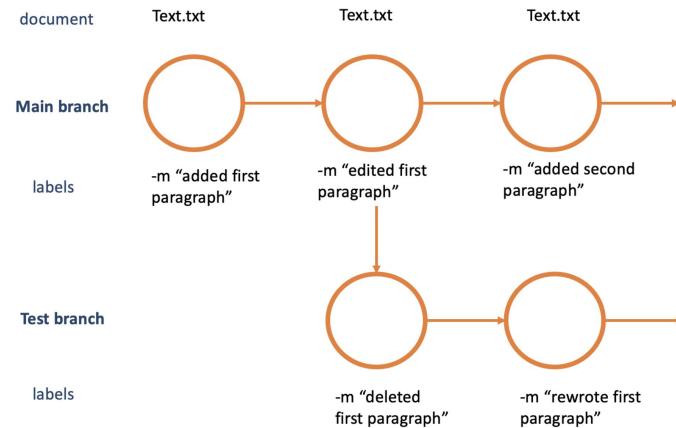
## Working on Github online

1. Log into your Github account.
2. Fork the course repo:  
[https://github.com/likeajumprope/RSE\\_Juelich](https://github.com/likeajumprope/RSE_Juelich)
3. In the folder for day1, you'll find a Readme.md.
4. Do Activity 1:
  - Create a file, add your favourite animal and make a PR.

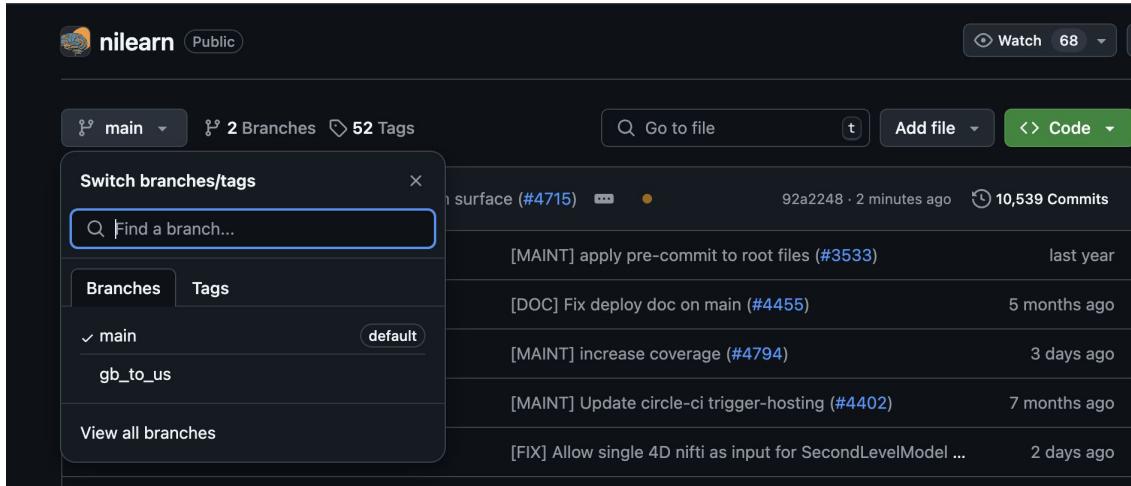
# Branches

## Collaboration (github online)

- **Branch:** Diversion point of repository and history



# Make branch and merge back into your code



Branches

# Activity 2 (15')

Branching

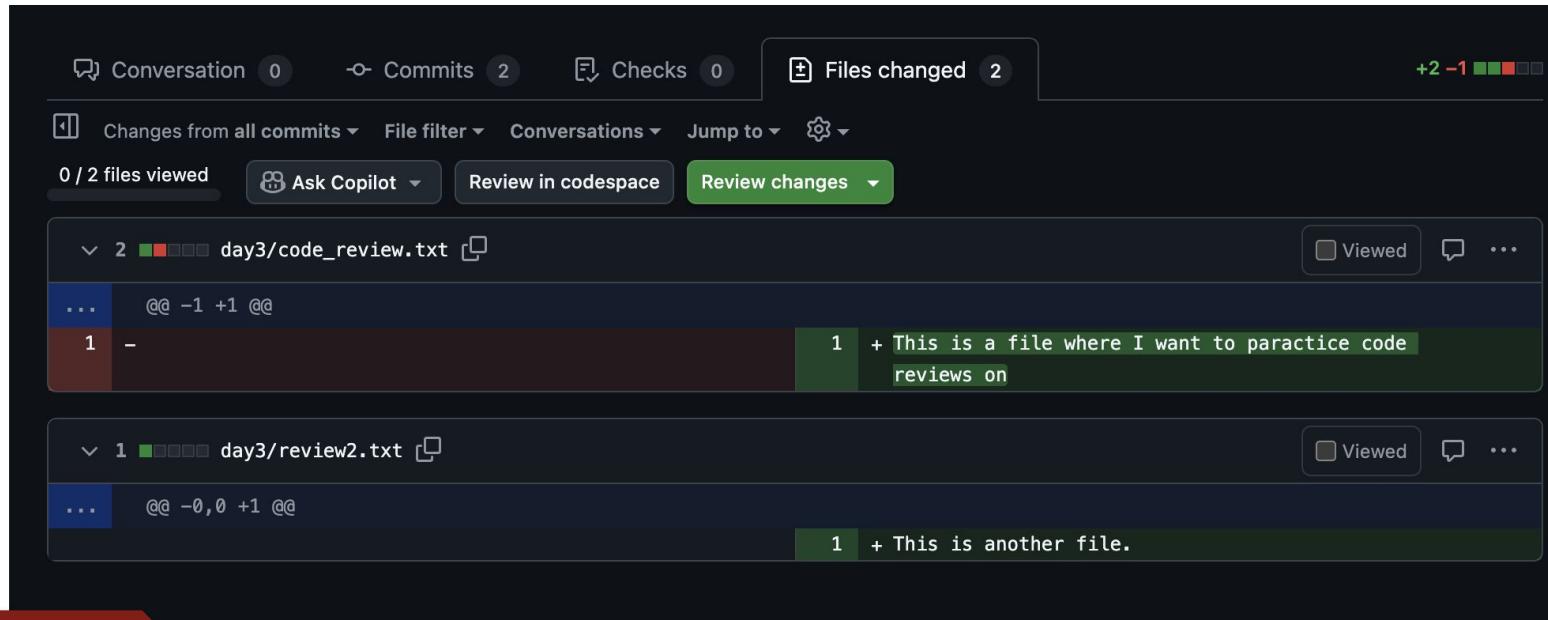
Branches

- Under the Readme.md section of day1, do Activity 2:
- Create a branch, make a change, merge it back into main

# Break (15 min)

# Review a PR online

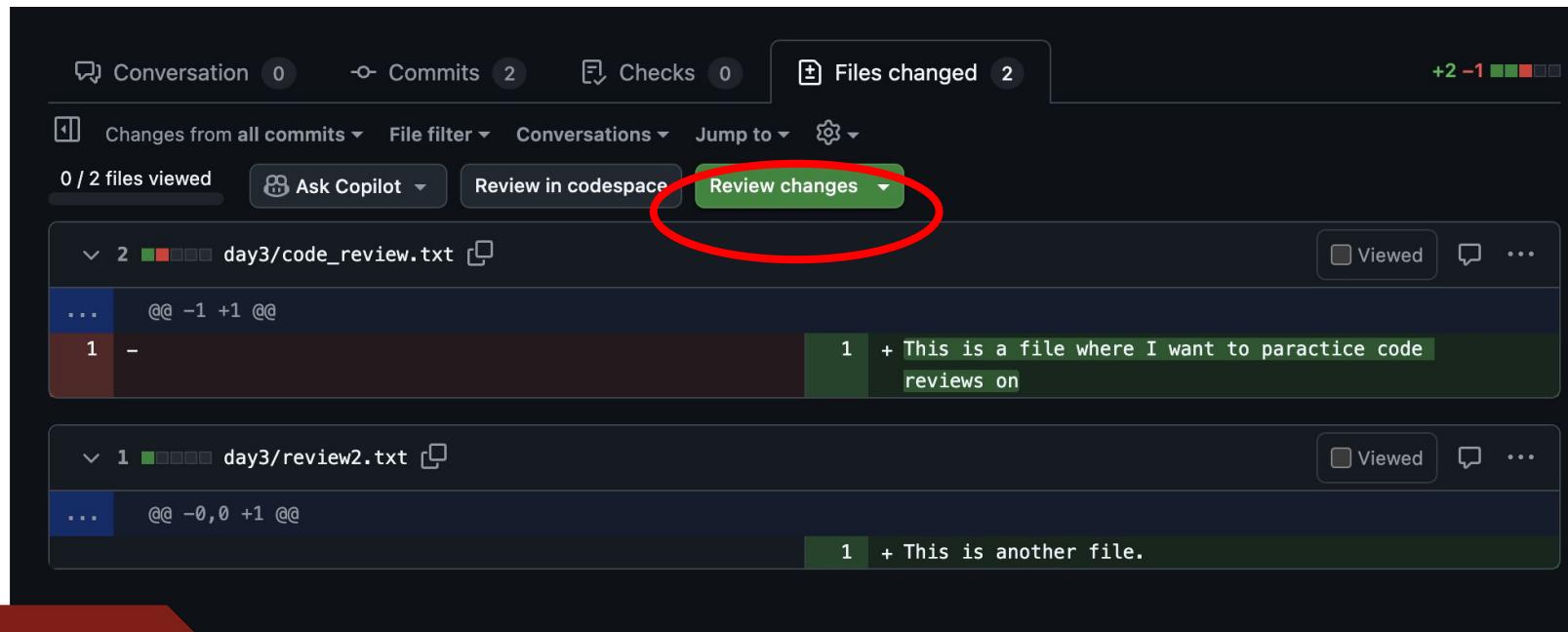
Github allows for a very efficient code review process



Code reviews

# Review a PR online

Click on “review changes”



Code reviews

# Review a PR online

Click on “review changes”

The screenshot shows a GitHub pull request interface with the following details:

- Header:** Conversation (0), Commits (2), Checks (0), Files changed (2). A status bar shows +2 -1.
- Toolbar:** Changes from all commits ▾, File filter ▾, Conversations ▾, Jump to ▾, ⚙ ▾.
- File List:** 0 / 2 files viewed.
  - day3/code\_review.txt:** 2 changes. The first change is @@ -1 +1 @@. The second change is 1 -. A red circle highlights the "Viewed" checkbox and the comment "This is a file where I want to practice code reviews on".
  - day3/review2.txt:** 1 change. The change is @@ -0,0 +1 @@. A red circle highlights the "Viewed" checkbox.

Code reviews

# **Activity 3 (15')**

## **Forking, PR's and code review**

Under the Readme.md section of day1, do Activity 3

- Fork each other's repo, make a PR, assign each other as reviewers and perform a code review

# Some more Github features

# Project management - boards on github

Project boards are a great way to organize a project and issues on github.

The screenshot shows the GitHub interface for the repository "community-management". The top navigation bar includes links for Code, Issues (17), Pull requests, Discussions, Actions, Projects (1), Wiki, Security, and Insights. The repository name "community-management" is displayed with a "Public" badge. Below the header, there are buttons for Edit Pins, Watch (6), Fork (1), and a dropdown menu. The main content area shows a list of recent commits:

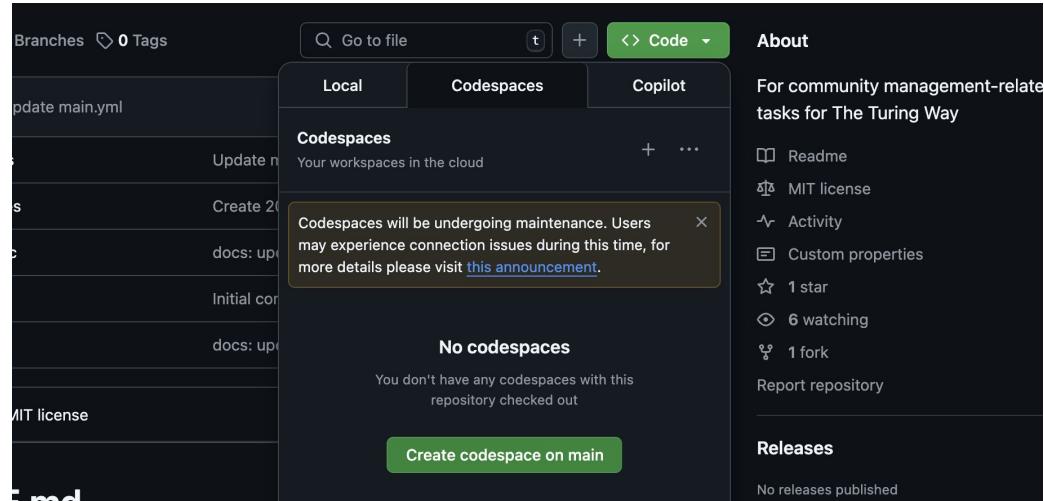
Author	Commit Message	Date
likeajumprope	Update main.yml	a501b92 · 2 days ago
.github/workflows	Update main.yml	2 days ago
wg-meeting-notes	Create 20250408-wg-notes.md	4 days ago
.all-contributorsrc	docs: update .all-contributorsrc	4 days ago
LICENSE	Initial commit	last year
README.md	docs: update README.md	4 days ago

On the right side, there is an "About" section with links to Readme, MIT license, Activity, Custom properties, 1 star, 6 watching, 1 fork, and Report repository. At the bottom, there are links for Project boards, Releases, and a red arrow pointing to the "Project boards" link.

# Codespaces

Github codespaces allows you to run files in Github repository in VS code instance in your browser.

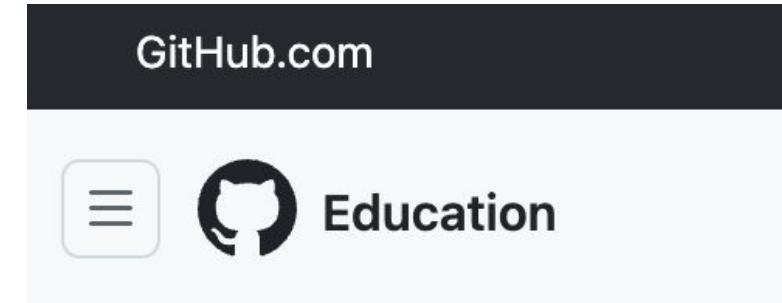
- Good for quickly checking things
- CAREFUL: You have some free quota (as student etc), but then it's a paid service
- Close your instance!!



Codespaces

# Github Education

- As a registered student, you can gain access to Github education
  - Free perks, such as:
    - Free Co-Pilot and Codespace
    - Full Notion AI plan
    - Licenses to Jetbrains IDE etc
    - Certifications



# Github Copilot

- Chat-GPT for CODE
  - Integratable with VS code
- 
- If you are a registered maintainer of an open source repository, you also get free access to codespaces and co-pilot.



What are your questions?

# Ressources

Baxter, R, Chue Hong, N, Gorissen, D, Hetherington, J & Todorov, I 2012, 'The Research Software Engineer', Digital Research 2012, Oxford, United Kingdom, 10/09/12 - 12/09/12.

The Missing Semester of Your CS Education: <https://missing.csail.mit.edu/>

Nuest, D., Konkol, M., Pebesma, E., Kray, C., Schutzeichel, M., Przibytzin, H., & Lorenz, J. (2017). Opening the Publication Process with Executable Research Compendia. *D-Lib Magazine*, 23(1/2). [10.1045/january2017-nuest](https://doi.org/10.1045/january2017-nuest)

BIDS standard: <https://bids.neuroimaging.io/index.htm>

Goth, F. *et al.* Foundational competencies and responsibilities of a Research Software Engineer. *F1000Res.* **13**, 1429 (2024).

<https://book.the-turing-way.org/collaboration/github-novice>

<https://book.the-turing-way.org/reproducible-research/vcs>