



# Git Good

## an Introduction to GitHub for Collaboration
















**Cassandra Gould van Praag**  
Executive Director, RCM Cooperative  
TOSI Open Scientist in Residence 2025



**Emma Karoune**  
Principle Researcher, Alan Turing Institute  
Co-Founder, RCM Cooperative

### Contributors ✨

Thanks goes to these wonderful people ([emoji key](#)):

 <a href="#">Cassandra Gould van Praag</a> 📖 ✅ 📧	 <a href="#">Hari Sood</a> 📖 ✅ 📧	 <a href="#">Sophia Batchelor</a> 📖 ✅ 📧	 <a href="#">Esther Plomp</a> 📖 ✅ 📧	 <a href="#">Anne Lee Steele</a> 📖 ✅ 📧	 <a href="#">Emma Karoune</a> 📖 ✅ 📧	 <a href="#">GabinWK</a> 📖 ✅ 📧
 <a href="#">Malvika Sharan</a> 📖 ✅ 📧	 <a href="#">Eirini Zormpa</a> 📖 ✅ 📧	 <a href="#">Ayesha Magill</a> 📖 ✅ 📧	 <a href="#">Alexandra Araujo Alvarez</a> 📖 ✅ 📧	 <a href="#">Vicky Hellon</a> 📖 ✅ 📧	 <a href="#">andrepiza</a> 📖 ✅ 📧	

*All illustrations by Scriberia for  
The Turing Way  
used under CC-BY 4.0 License*

This template has been  
created by **Slidesgo**

This project follows the [all-contributors](#) specification. Contributions of any kind welcome!



# Housekeeping

- Slides and list of links available at:
  - <https://github.com/the-turing-way/gitgood>
- Recording: **Camera off / chat** if you don't want to be attributed
- Add questions to **zoom chat**
- Use **raise hands to speak** and **use zoom reactions** to let us know when you have finished an activity
- We are operating under the McGill Faculty of Medicine and Health Sciences **Code of Conduct**
  - "...demonstrate respect and appreciation for [your] colleagues..."
  - <https://www.mcgill.ca/medhealthsci/about/our-vision-mission-values/code-conduct>
  - See Appendix 1 for reporting procedures
- You will need a **GitHub account** to follow along with the tutorials
  - **Make one now if you haven't got one already!**
- All exercises will use the **online GitHub interface**
  - Does not require git to be installed or security set up
- Julia Pfarr (ORIGAMI Lab) Git/Hub 101 at Mtl BrainHack 2026:
  - [https://julia-pfarr.github.io/git-gitlab\\_tutorial/](https://julia-pfarr.github.io/git-gitlab_tutorial/)
- Kendra Oudyk (ORIGAMI Lab) Intro to Git/GitHub OSHO Feb 2025
  - <https://www.youtube.com/@OpenScienceOfficeHours>



# Aims

- Feel more confident in:
  - Interacting with GitHub;
  - Contributing to other peoples GitHub materials;
  - Setting up your own repositories to receive contributions;
- Know where to find help with any of this!

## Agenda

- Part 1: What is Git & Github?
- Exercise 1: Finding an Issue and contributing to a project
  - You will practice: Using issues to contribute and document decisions, markdown
- Part 2: What is open source?
- Part 3: Using GitHub online and in the terminal (demo)
- Exercise 2: Editing a file and making a Pull Request
  - You will practice: Markdown; contributing, pull requests
- Part 4: What can we make with Github? (Websites and Project Management)
- Exercise 3 (take home): Make your own website



# What is GitHub?



- **Sounds like it's only for programmers!**
- GitHub is also great for version control of anything with letters/numbers/characters (and images) in and even better for working collaboratively!
- GitHub enables:
  - Making and sharing of something which looks professional
  - Tracking of changes as your project grows
  - Integration of contributions from others
  - Maintenance of an open "To Do" list
  - Licensing and DOI'ing materials, making them reusable and citable



# Git vs GitHub



- Software which enables for version control.
- Free and open source!
- Often pre-installed on your computer



- A popular service for hosting and sharing projects you have been tracking with git.
- Secure and flexible control of access to materials.







# Collaborative docs



- Multiple people contributing to the same information from different times/places.

Hand/Chat:  
What might be  
challenges  
with distributed  
collaboration?

Hand/Chat:  
How could  
version control  
help?





# Making distributed collaboration possible



Version control



Reviewing & updating



Attribution to contributors



Transparency & accountability



Project management



Share globally



# Navigating


- Files:
  - README
  - CONTRIBUTING
  - CODE\_OF\_CONDUCT
  - LICENSE
- Issues
- Pull requests
- Projects
- Contributors
- File views:
  - Preview
  - Code
  - Blame
  - History
  - Commits
  - Diff
- Repo vs organization
- Organisation members
- Repo collaborators

Chat: Any  
buttons you  
want to see  
or  
understand?





# Exercise 1: Commenting

- Go to the practice space repo **issues** and complete issue #3
- Take a look at the **markdown** cheat sheet to add some formatting to your comment
- Use zoom reaction  when you are done

Collaboration  
practice!



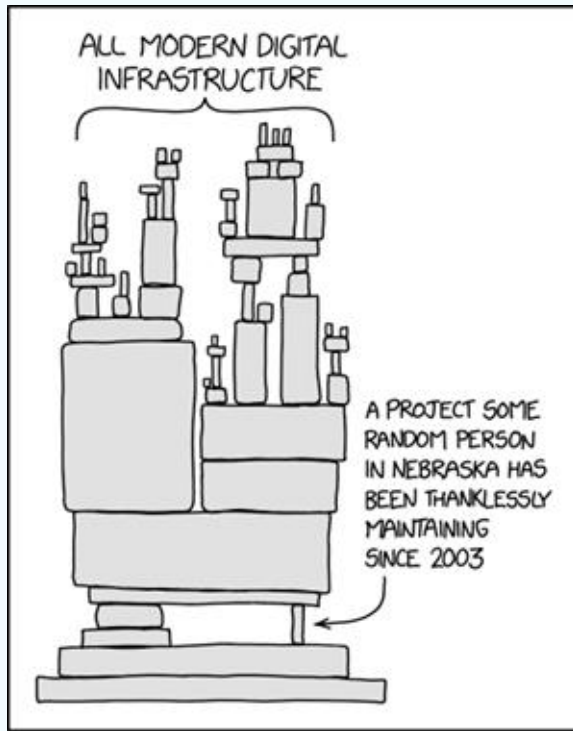
# Discussion

- What are the pros and cons of using a platform like GitHub for:
  1. Your own code and documents?
  2. Shared (private) code and documents?
  3. Shared (public) code and documents?
- **What are your questions?**

Hands /  
Chat



## Part 2: Open Source



Chat: What open source software do you use?

*xkcd*, 'Dependency': <https://xkcd.com/2347/>

### Open Source Initiative (OSI) Legal definition:

- The software is free to redistribute
- The source code is included
- "Derived works" (remixes) are allowed
- There is no discrimination against persons, groups, or fields of endeavour (e.g., you cannot ban a specific company or country from using it)
- Roots in free software movement
- Developed into free and open source software (F/OSS) movement
- Influenced other 'open' movements: open knowledge, open data, open access, open science



# Why do we need licenses?

## The Legal Default is "Closed"

- By international copyright law, any original work (code, text, or data) is **automatically protected by "All Rights Reserved"** the moment it is created.
- If you put materials on GitHub without a license, other researchers can look at it, but they cannot **legally** download, run, or build upon it.
- A license tells the world, **"I own this, but I am giving you legal permission to use it under these specific conditions."**

## Safeguarding "Safe Sharing"

- Researchers often hesitate to share because they fear their work will be used in ways they didn't intend. Licenses create a contract.
- Liability Protection: Almost all open licenses include a "Disclaimer of Warranty." which protects you from being sued if someone else uses your code and it fails or produces an error.

## Credit as The Currency of Academia

- Academia runs on citation and reputation.
- In traditional publishing, we have established norms for citing papers. For code and documentation, licenses like MIT or CC-BY turn "attribution" from a polite suggestion into a legal requirement.
- By licensing your "non-paper" outputs (datasets, software, tutorials), you are forcing the academic ecosystem to treat those contributions as valuable, citable objects.



# Licenses & copyright



## Copyright

All Rights Reserved

Reuse requires the permission from the copyright owner

## Open License

Some Rights Reserved

Re-use is permitted without permission under the specific limitations of the license

## Public Domain

No Rights Reserved

May be used without permission

- **Permissive Licenses**

- "You can do whatever you want with this, as long as you give me credit (attribution) and don't hold me liable."
- Examples: **MIT, Apache 2.0, BSD, CC-BY-4.0**
- Good: Encourage maximum adoption. No issues with IP or license stacking
- Bad: Changes can be relicensed as closed

- **Copyleft (reciprocal) Licenses**

- "If you use my open materials, you must make your derived materials open too"
- Examples: **GNU GPL (v2 or v3), AGPL, CC-BY-SA**
- Good: Prevents enclosure of the commons
- Bad: Hesitancy to force openness

Go look for a software license





# Licenses for docs

- When you publish in Neuroimage, you are asked to choose from one of Elsevier's approved "open access user licenses":
  - Creative Commons Attribution (**CC BY**)
  - Creative Commons Attribution-NonCommercial (**CC BY-NC**)
  - Creative Commons Attribution-NonCommercial-NoDerivatives (**CC BY-NC-ND**)
  - Copyright Elsevier or a Society
  - Elsevier user license

Which is most / least permissive?

**creative commons**

**Licenses**

**Icons**

**Terms of the Licenses**

**Public Domain Dedication (CC0)**  
This is considered a dedication to the public domain, and thus the creator(s) associated with this item have waived all their rights to the work worldwide under copyright law.

**Attribution (BY)**  
Others can copy, distribute, display, perform and remix the work if they credit/cite the creator/author.

**Derivative Works (ND)**  
Others can only copy, distribute, display or perform verbatim copies of the work. (No modifications allowed.)

**Share Alike (SA)**  
Others can distribute the work only under a license identical to the one attached to the original work.

**Non-Commercial (NC)**  
Others can copy, distribute, display, perform or remix the work but only for non-commercial purposes.

This work is a CC0 Public Domain Dedication work.

Ohio State University, CC0



# Licenses for docs

Why do docs have different licenses than code?

## "Software" vs. "Literary Works"

- Code Licenses (MIT, GPL): are designed for functional works. They deal with things like "compilation," "object code," and "execution."
- Creative Commons (CC): Are designed for literary and artistic works. They cover "reproduction," "adaptation," and "display."
- Most code licenses don't explicitly address the rights needed for documentation, such as the right to translate a book into another language or the right to print and sell physical copies.

## The "Attribution" Problem

- In the MIT license, the requirement is to keep the "copyright notice and permission notice" in all copies of the software (LICENSE.md)
- In documentation, CC-BY gives you more flexibility on how you give credit (e.g., a "Credits" page or a footer).
- Using MIT for a book might legally require you to print the entire MIT license text on every single page or diagram.

How would  
you like your  
research  
outputs to be  
licensed?



# Licenses for hardware

- Follow similar permissive < > copyleft spectrum

## The "Right to Manufacture"

- "Copying" involves buying components, soldering, and assembly.
- A hardware license gives you the legal right to manufacture a physical version of the design.
- Without a hardware-specific license, someone might have the right to look at your circuit diagram (protected by copyright), but they might not technically have the right to build and sell the physical item if the "utility" isn't covered by the license.

## The Patent "Safety Net"

- Copyright protects how ideas are written down. Patents protect how things work.
- If you share a new sensor design under a documentation license, a company could file a patent on the "invention," and then sue you for making your own device.
- Licenses like CERN-OHL or Solderpad include "Patent Grants." This means anyone who contributes to the project effectively says: "I promise not to sue anyone for using any patents I have that are part of this design."

## The "Physical" Attribution

- Hardware licenses often include clauses about "Notice Retention" on the physical object.
- Hardware licenses specify where/how the license is identified on the physical item.



# Summary slide



- We want to **share our work** so our research can have the greatest impact!
- Sharing on GitHub enables the use of a well tested mechanism to receive the input of others
  - More contributions been **more robust** and potentially more impactful
  - Contributions are **attributed and version controlled**
- When we share, we should always apply a license so we can control **how our outputs are used on by whom**.
  - Licenses can be **more or less restrictive**
  - Software documentation and hardware all have specific license types which **legal terms appropriate** to different outputs





# Part 3: Using GitHub

- For some repos you might have a 'local' copy on your computer and a 'remote' copy on GitHub
- Personally, I have a local copy where I want to do more complicated editing

## Pushing your local changes to your remote

- Directory must first be initialised for git (ready to be tracked)

> git status

> git add [filename]

> git commit -m "[describe the change]"

> git push

```
gitgood --zsh -- 111x38
(base) cassandragouldvanpraag@Mac gitgood % git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   CODE_OF_CONDUCT.md
        modified:   README.md
        modified:   workshops/202602_0SH0/list-of-links.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        tmp.md
        workshops/202602_0SH0/202603_GitGood_0SH0.pptx
        workshops/202602_0SH0/gitgood-new-new.pptx
        workshops/202602_0SH0/gitgood-new.pptx
        workshops/202602_0SH0/gitgood.pptx

no changes added to commit (use "git add" and/or "git commit -a")
(base) cassandragouldvanpraag@Mac gitgood % git add README.md
(base) cassandragouldvanpraag@Mac gitgood % git commit -m "update repo checklist"
[main 48f9fa9] update repo checklist
 1 file changed, 1 insertion(+), 1 deletion(-)
(base) cassandragouldvanpraag@Mac gitgood % git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 10 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 306 bytes | 306.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:the-turing-way/gitgood.git
   20aed3f..48f9fa9  main -> main
(base) cassandragouldvanpraag@Mac gitgood %
```

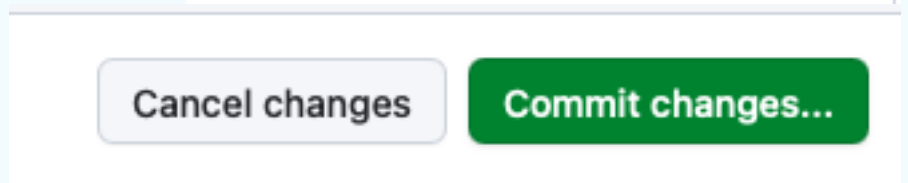
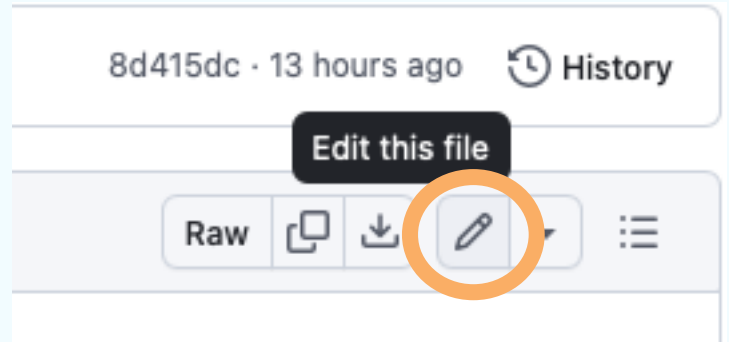




# Editing online

## Making online changes to your remote

1. Find the file
2. <Edit this file>
3. Make the change
4. <Commit changes>



Commit changes

Commit message

demo remote edit

Extended description

Add an optional extended description...

☒ Commit directly to the main branch

☐ Create a new branch for this commit and start a pull request

[Learn more about pull requests](#)

Cancel

Commit changes



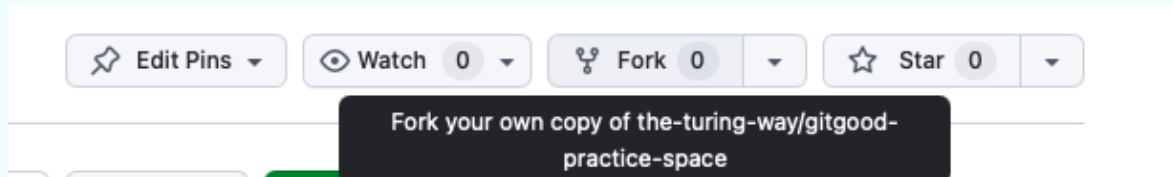
# Collaborating on someone else's repo

1. Make a copy of their repo (**fork**)
  2. Make your changes in *your* online copy (**commit**)
  3. Share your changes with the repo owner (**pull request**)
  4. Changes are incorporated in the original repo (**merge**)
- Additional step of making your own copy of the repo preserves the fidelity of the original and give you space to play!
  - We're going to practice this online only!



# Collaborating on someone else's repo

1. Make a copy of their repo (**fork**)



**Create a new fork**

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (\*).

Owner \*  Repository name \*

gitgood-practice-space is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

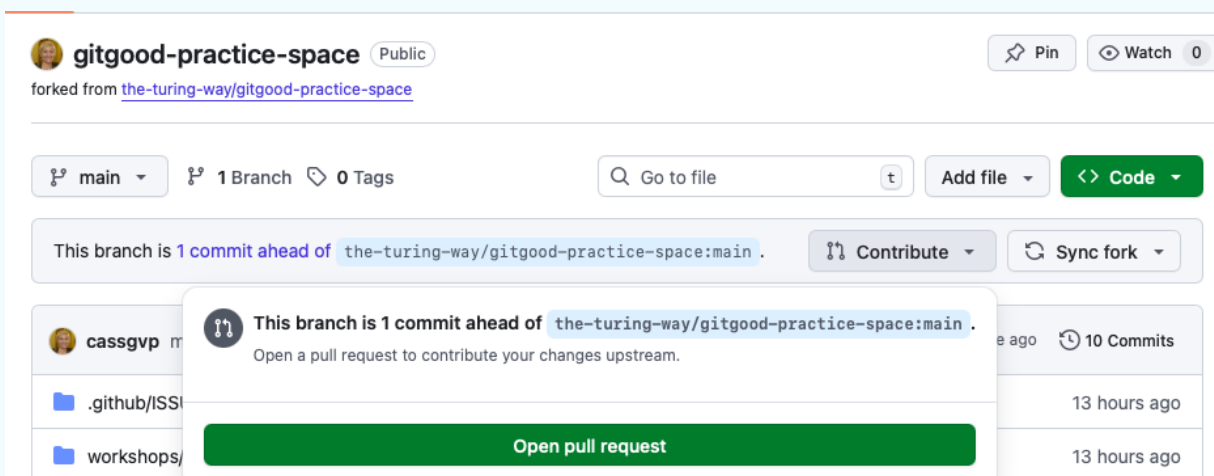
Description  
  
80 / 350 characters

☒ Copy the `main` branch only  
Contribute back to the-turing-way/gitgood-practice-space by adding your own branch. [Learn more](#).

① You are creating a fork in your personal account.

[Create fork](#)

2. Make your changes in *your* online copy (**commit**)
3. Share your changes with the repo owner (**pull request**)





# Collaborating on someone else's repo

**Comparing changes**

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about pull requests](#).

base repository: the-turing-way/gitgood-prac... base: main ← head repository: cassgvp/gitgood-practice-sp... compare: main

✓ **Able to merge.** These branches can be automatically merged.

**Add a title**

make a small change in my copy

**Add a description**

Write Preview H B I

Add your description here...

☐ Markdown is supported ☒ Paste, drop, or click to add files

☒ Allow edits by maintainers **Create pull request**

- Changes are reviewed by repo original repo owner (**merge**)

**Commit message**

Merge pull request #4 from cassgvp/main

**Extended description**

make a small change in my copy

This commit will be authored by 43407869+cassgvp@users.noreply.github.com.


**Confirm merge** **Cancel**

Still in progress? [Convert to draft](#)



# Exercise 2: Pull request

**We're all going to submit a pull request to the practice repo!**

1. Choose a line to edit from the framapad
2. Make a copy of the practice repo (**fork**)
3. Add your name; date today; a link to your research profile or some of your work to `exercise-2.md` (**commit**)
4. Share your changes with the repo owner (**pull request**)
5. Emma will review and incorporate your changes (**merge**)
6. Use zoom reaction  when your changes have been merged!

**We'll look out for any accident merge conflicts to review together**

Collaboration  
practice!







# Discussion

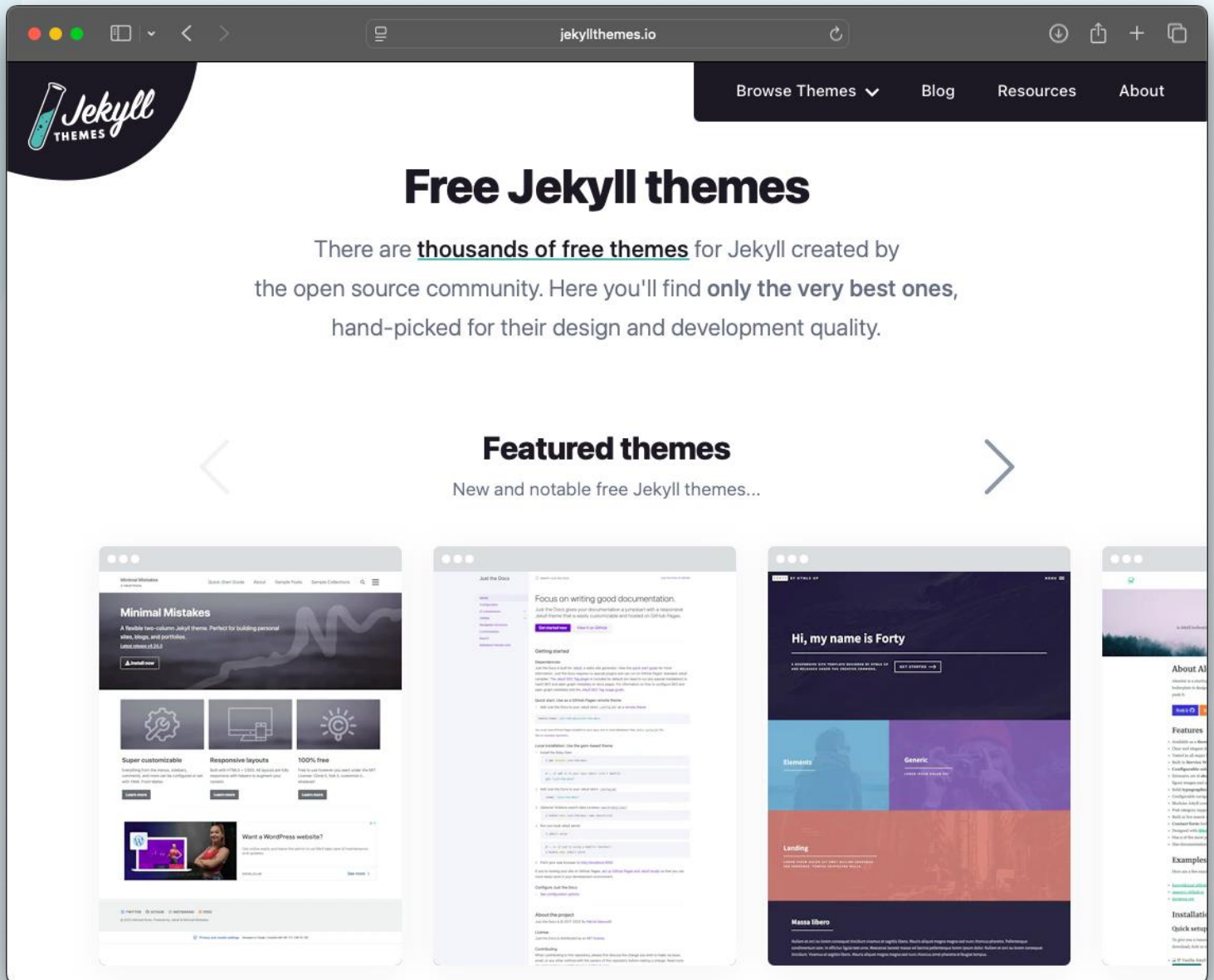
- What are your fears / concerns about working on someone else's repo?
- How would you like people to interact with you / your repos? (CONTRIBUTING.md)
- **What are your questions?**

Hands /  
Chat



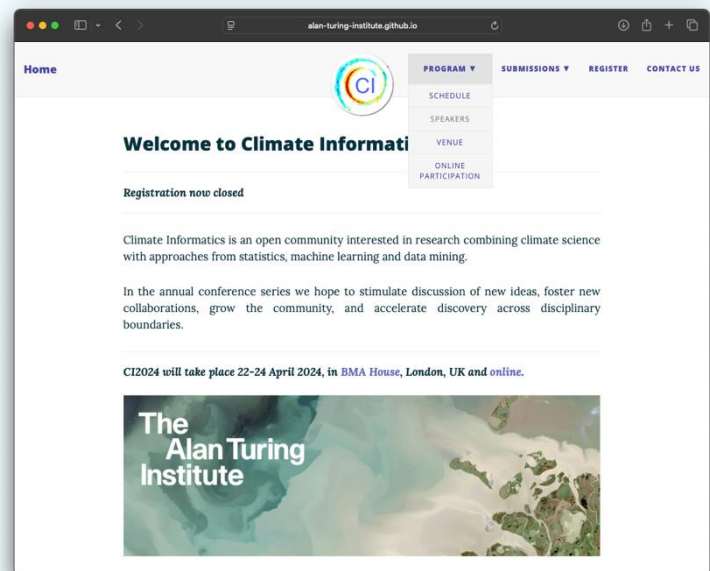
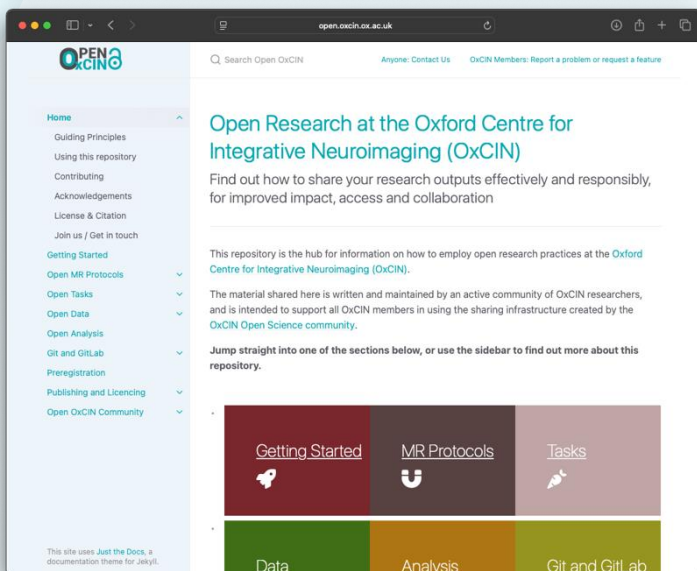
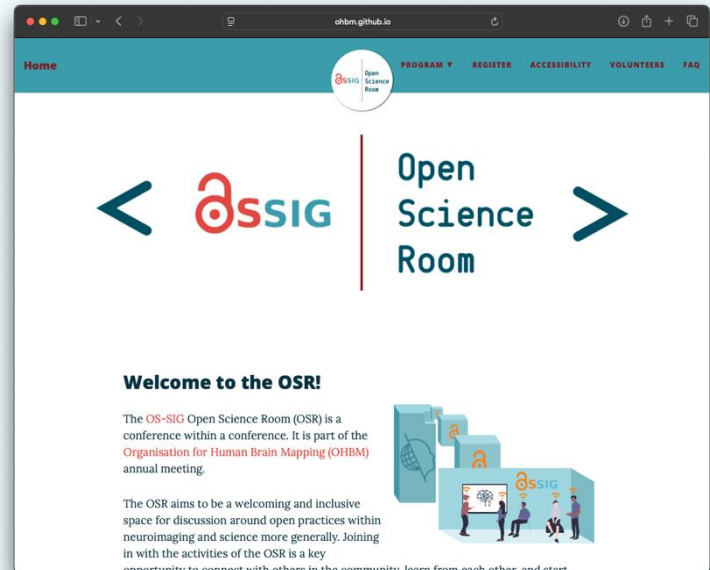
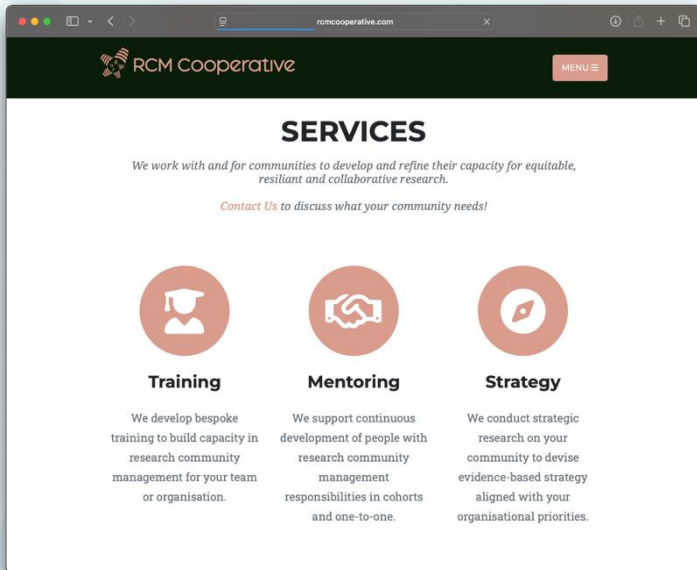
# Part 4: What else can you do with GitHub

## GitHub Pages (github.io)





# GitHub Pages



- Collaborative, version controlled, reproducible!



# GitHub Pages

## 1. Enable Pages

The screenshot shows the GitHub repository settings page for 'cassgvp / gitgood-practice-space'. The 'Settings' tab is selected and circled in orange. In the left sidebar, the 'Pages' option is also circled in orange. The 'General' section is active, showing the repository name 'gitgood-practice-space' and the default branch 'main'. The 'Releases' section is partially visible at the bottom.

## 2. Choose a branch

The screenshot shows the 'Build and deployment' section of the GitHub repository settings. The 'Source' dropdown is set to 'Deploy from a branch'. The 'Branch' section indicates that GitHub Pages is currently disabled and prompts the user to select a source branch. A 'Select branch' dialog box is open, showing a search bar and a list of branches. The 'main' branch is highlighted and circled in orange.

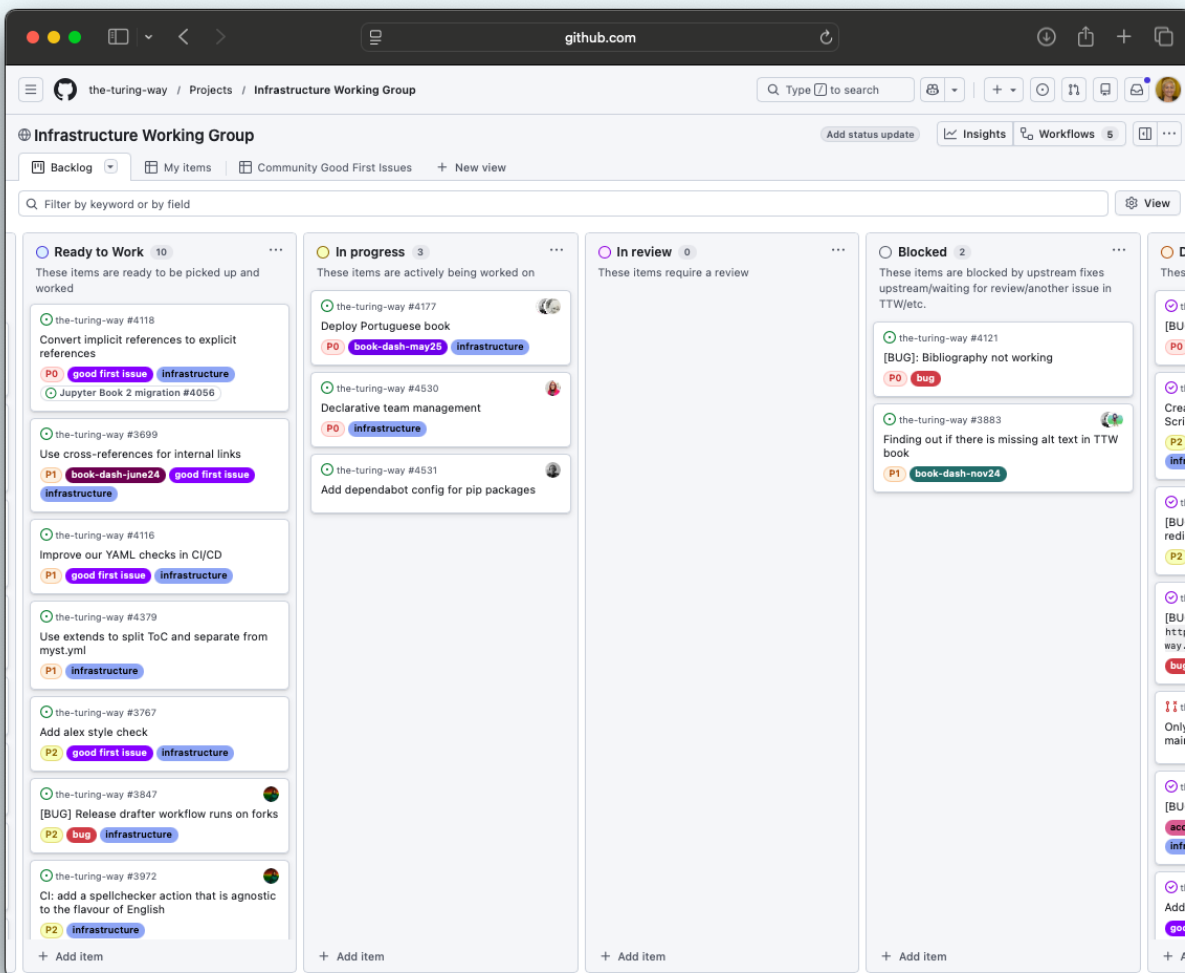
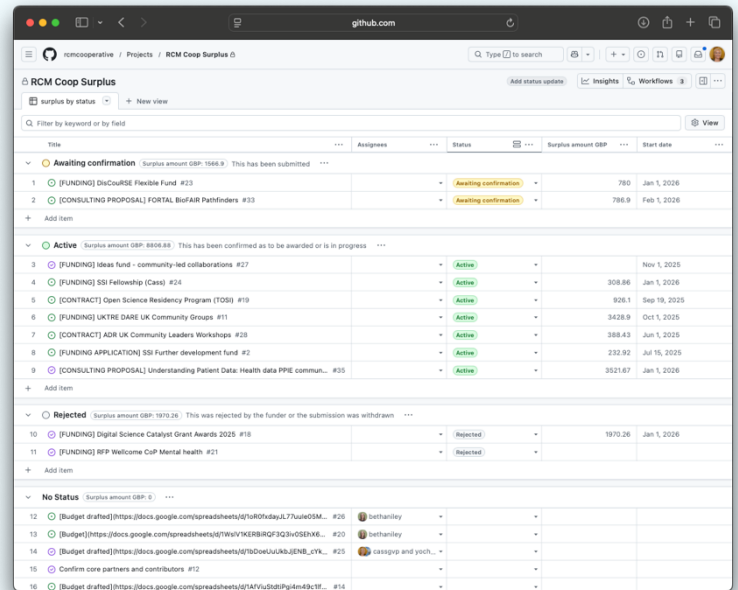
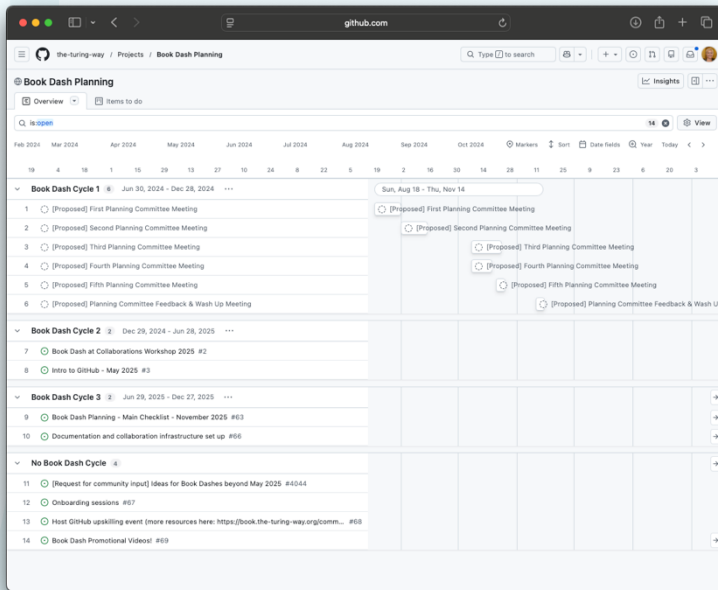
## 3. Apply a (remote) theme

The screenshot shows the 'Branch' section of the GitHub repository settings. It states that the GitHub Pages site is currently being built from the 'main' branch. Below this, there are dropdown menus for selecting the branch ('main') and the directory ('/ (root)'), followed by a 'Save' button. A link is provided to learn how to add a Jekyll theme to the site.





# Project Management





# Discussion

- What are you going to try next with GitHub?
- What are your questions?
- **Please fill in the feedback form now 😊**

Hands /  
Chat



# Summary slide



- We want to **share our work** so our research can have the greatest impact!
- Sharing on GitHub enables the use of a well tested **mechanism to collaborate with attribution**
- We know why and how to **license our materials**
- We know how to **submit and receive contributions**
- We know that we can **publish collaborative webpages** with GitHub
- We know that GitHub can be used to **manage collaborative projects**





# Closing

- Thanks to TTW GitGood contributors and TTW infrastructure team!
- Thanks to OSHO for brining us together!
- Thank you for engaging!
- Book time with Cass to discuss anything related to this workshop, open science / reproducibility, collaboration / research community management!

