Working in software teams

Data Sciences Institute
University of Toronto

Simeon Wong

Asking questions

- Zoom chat during class
  - Feel free to post and answer questions at any time
  - I will pause for questions occasionally, and review questions from the chat
- Pre- / Post-class office hours with Tong
- Email
  - simeonm.wong@mail.utoronto.ca
  - tong.su@mail.utoronto.ca

Daily scrum

Write in the chat:

1. How you'll use one thing you've learned so far
2. One thing you're struggling with / wish the course addressed

How to write robust software in a team that we, our colleagues, and the public can trust and use with confidence.

Previously at the DSI...

- Testing code
  - Write code with testing in mind
  - `pytest` and GitHub Actions for automated testing

- Python packages
  - Python code is usually distributed as packages on PyPI
  - Use `setuptools` to describe dependencies
  - Uploading and installing packages from GitHub using `pip`

Course overview

1. Configuration files & Environment variables
2. Using and writing Application Programming Interfaces (APIs)
3. Handling errors
4. Testing software
5. Building Python packages
6. **Working in software teams using GitHub features**

Today’s learning objectives

- I can track and manage software projects using GitHub
- I can ensure code is reviewed and tested using Pull Requests
- I know the principles of software project governance

Review

Why Git?

Scenario: Alex is a developer for a business-intelligence dashboard project.

- Alex's current big project is upgrading a financials chart on the dashboard home page
- Halfway through, Alex's colleague Jordan publishes a bug-fix on one of the analysis functions Sam uses for his chart
- Alex needs to integrate the updated analysis function into his work before shipping his upgraded chart to ensure it is correct.

Review

Git: Track changes for software

- Addresses unique challenges with working on software collaboratively
- Tracks changes to code over time
  - Backup of your work
  - Revert to older snapshots / point-in-time
- Merge changes from different features developed simultaneously

Review

Git commands

Setup

- git init
- git clone

Making edits

- git add
- git commit

Navigating through branches and commits

- git checkout
- git switch

Getting info

- git status
- git log

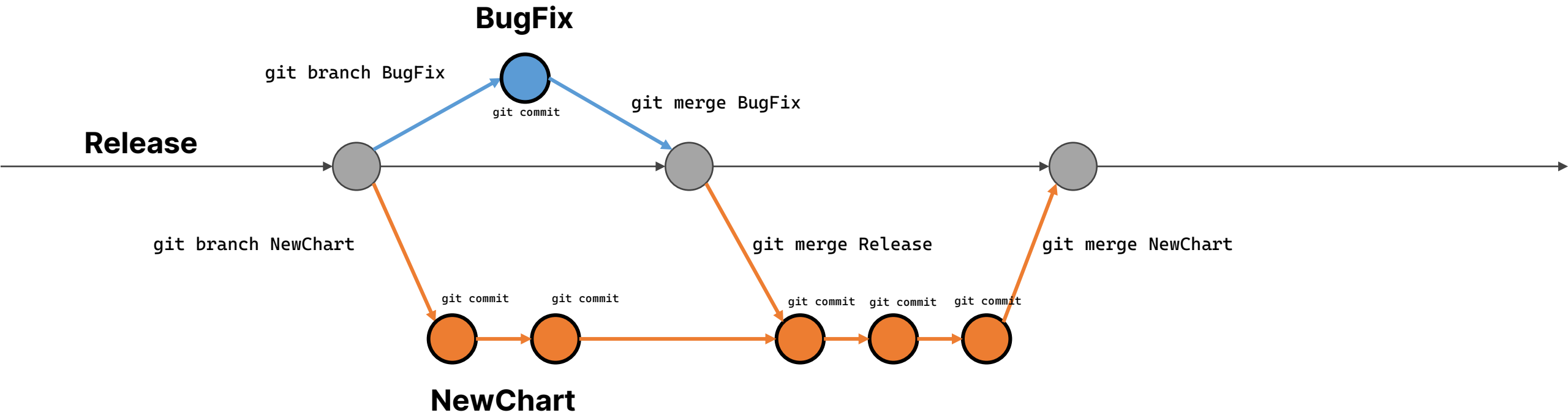
Incorporating changes

- git pull
- git merge

Review

Alex's new feature

1. Alex starts working on NewChart
2. Jordan starts working on BugFix
3. Jordan ships BugFix
4. Alex integrates BugFix
5. Alex keeps working on NewChart
6. Alex ships NewChart



Why software project management?

Scenario: Alex is a developer for a business-intelligence dashboard and is upgrading a financials chart on the dashboard home page

- Alex wants to gather ideas and feedback:
  - How the chart should look
  - How to make the analysis engine the most efficient
- Alex needs a place to track major discussion points on this feature

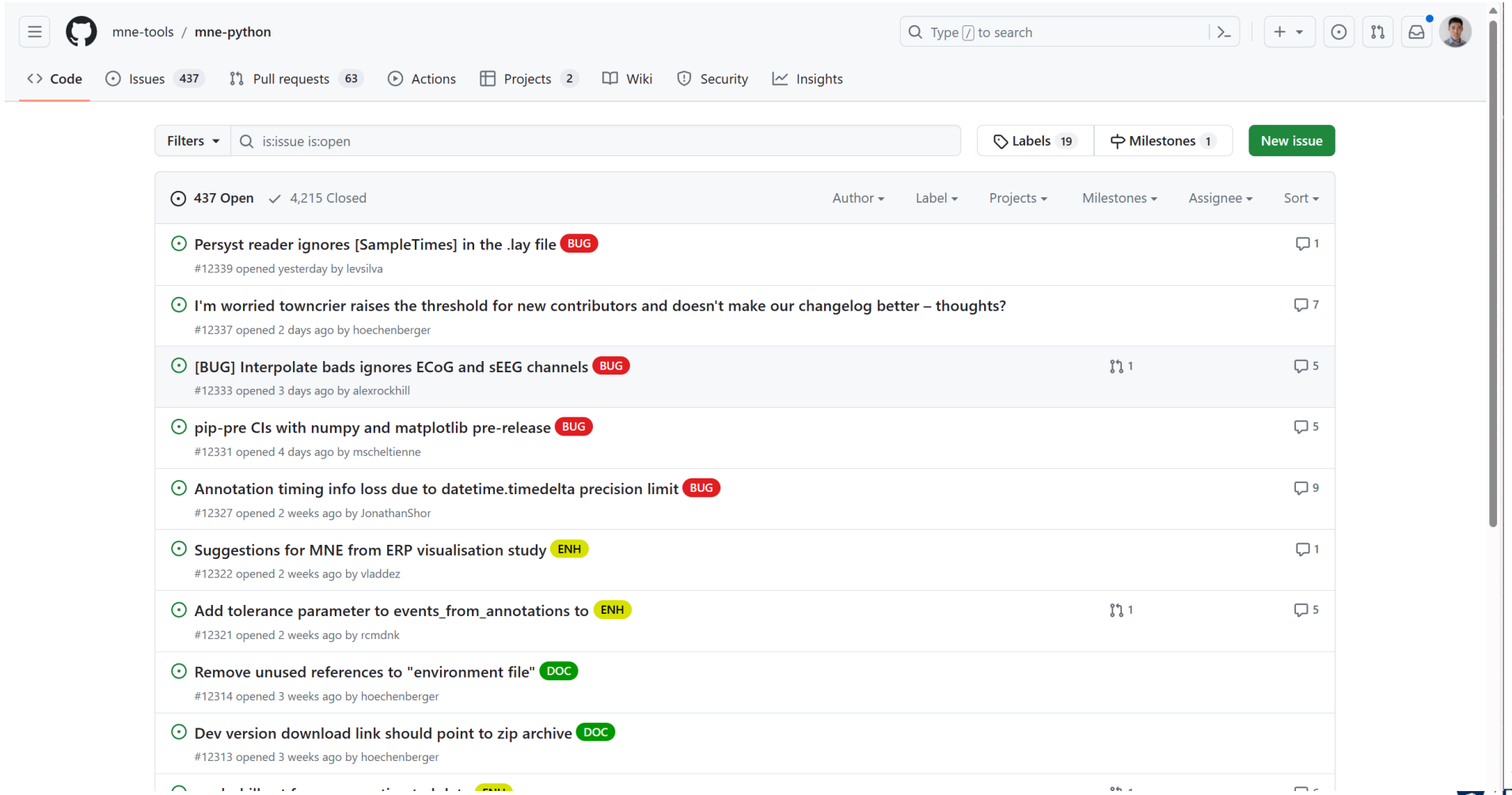
Working in software teams

GitHub

- Centralized hosting for Git repositories
- Project management features
  - Issues
    - Description of proposed or requested changes: update, features, bug-fixes, etc..
    - Can associated one or more pull-requests (proposed code changes)
    - Discussion thread
  - Projects
    - Linked to GitHub Issues
    - View Issues in tabular / Gantt chart view

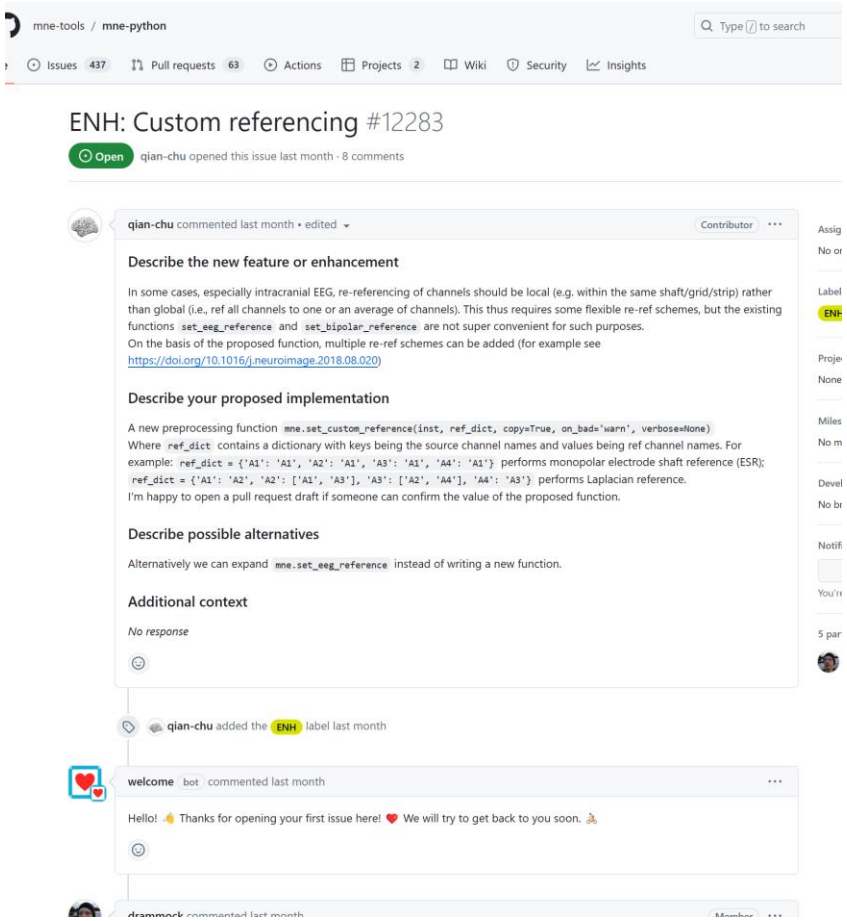
Working in software teams

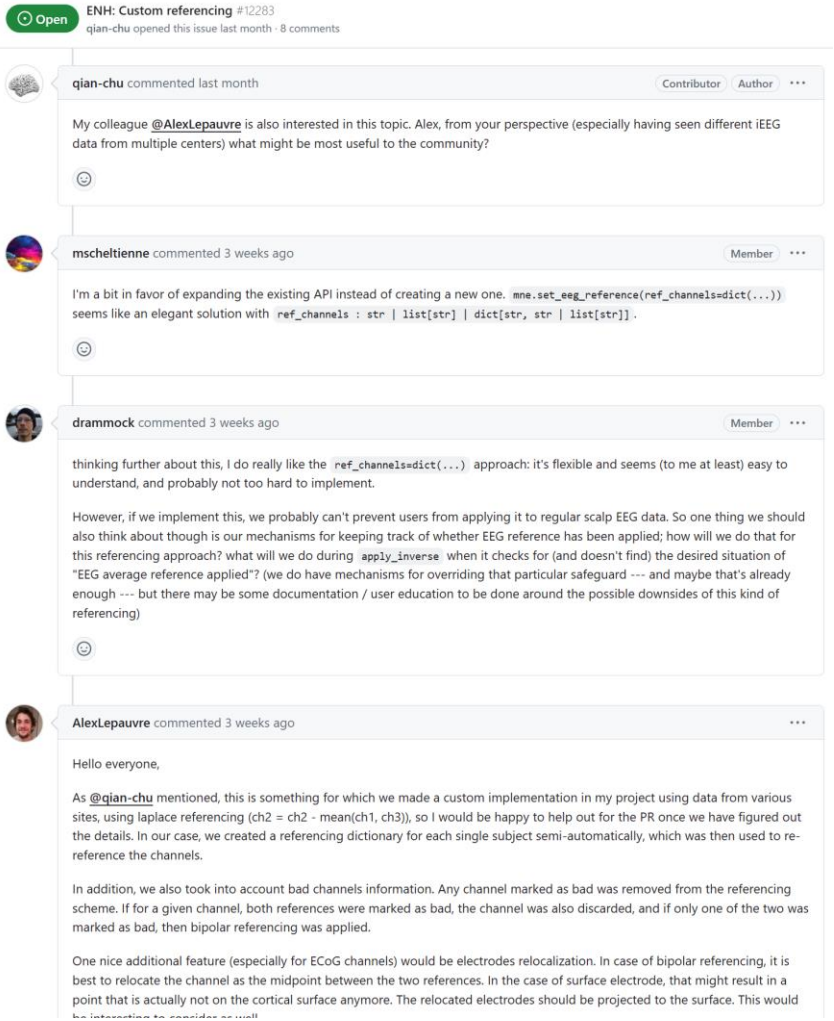
GitHub Issues tracks proposed changes



Working in software teams

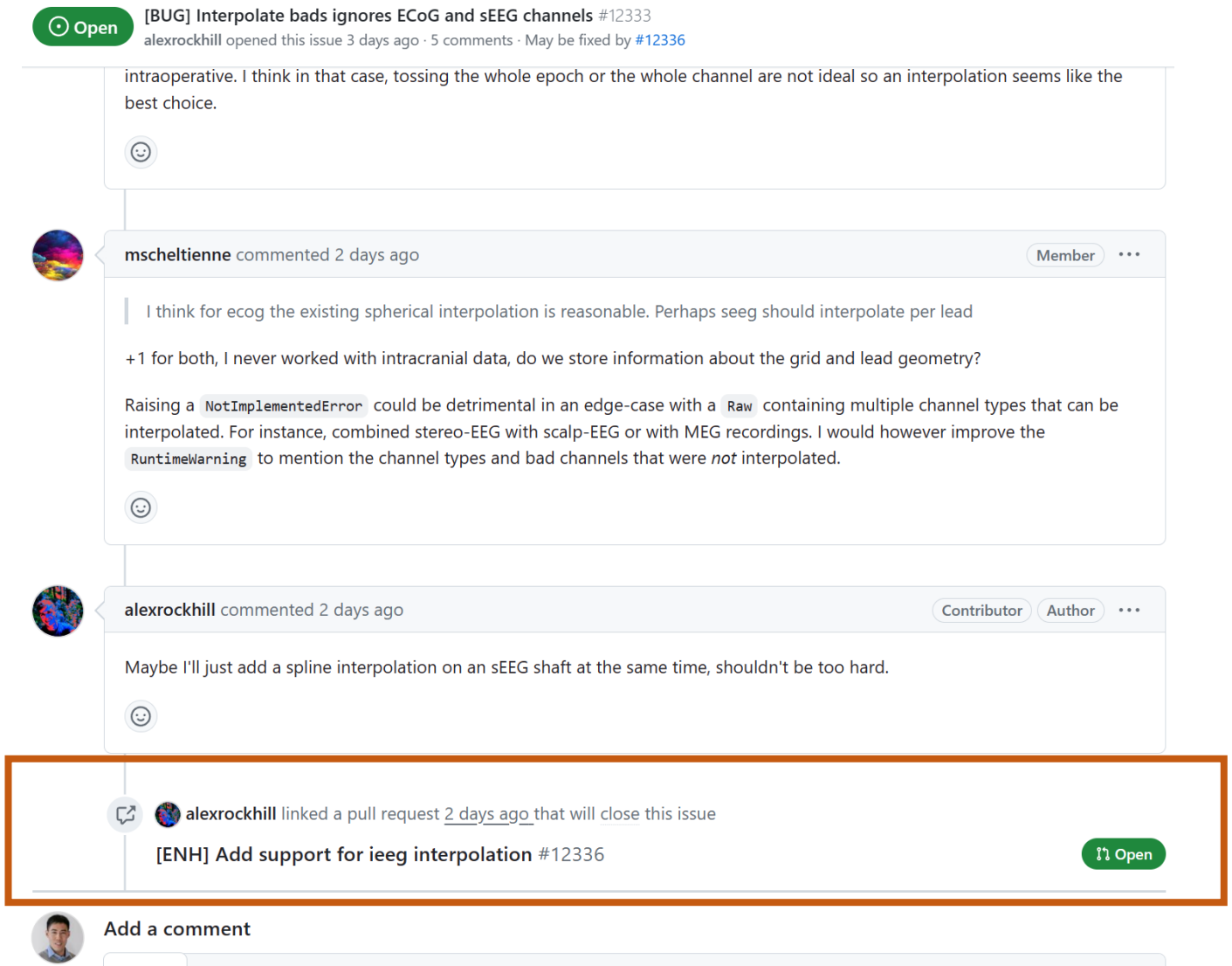
GitHub Issues allows for discussion



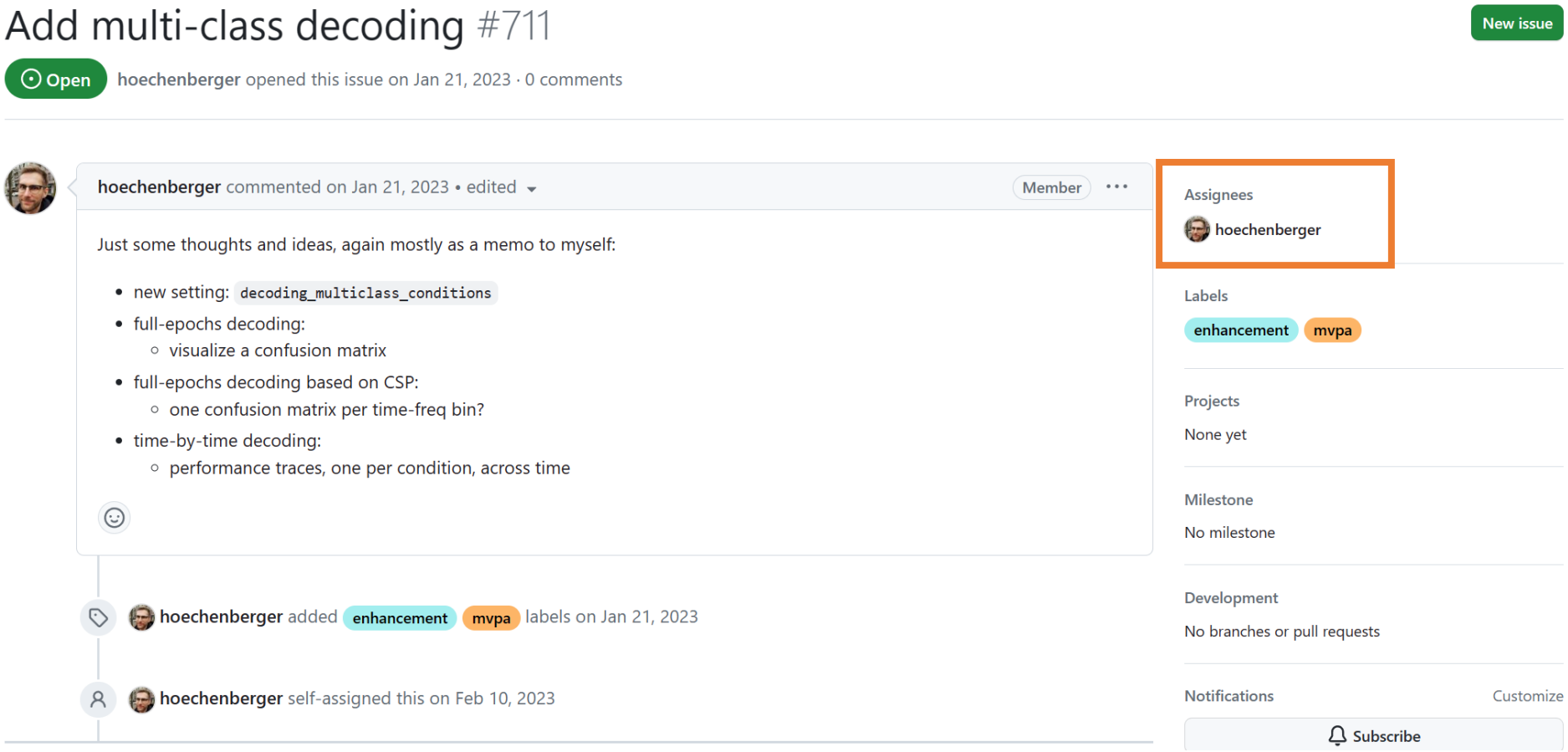


Working in software teams

GitHub Issues + Pull Requests

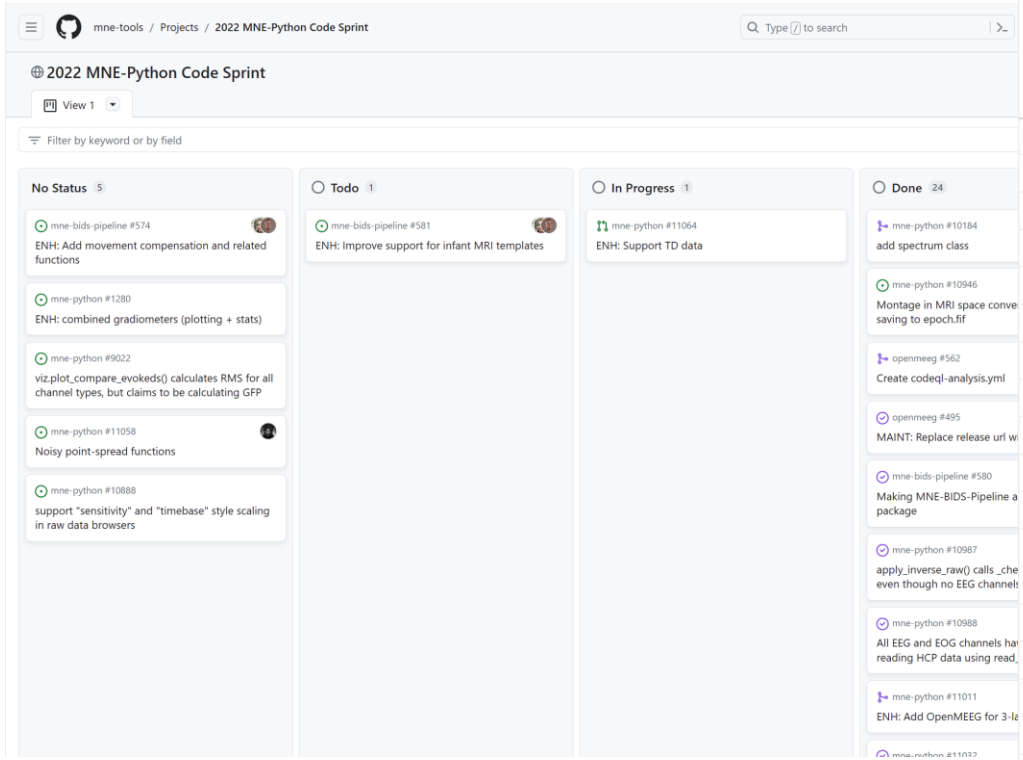


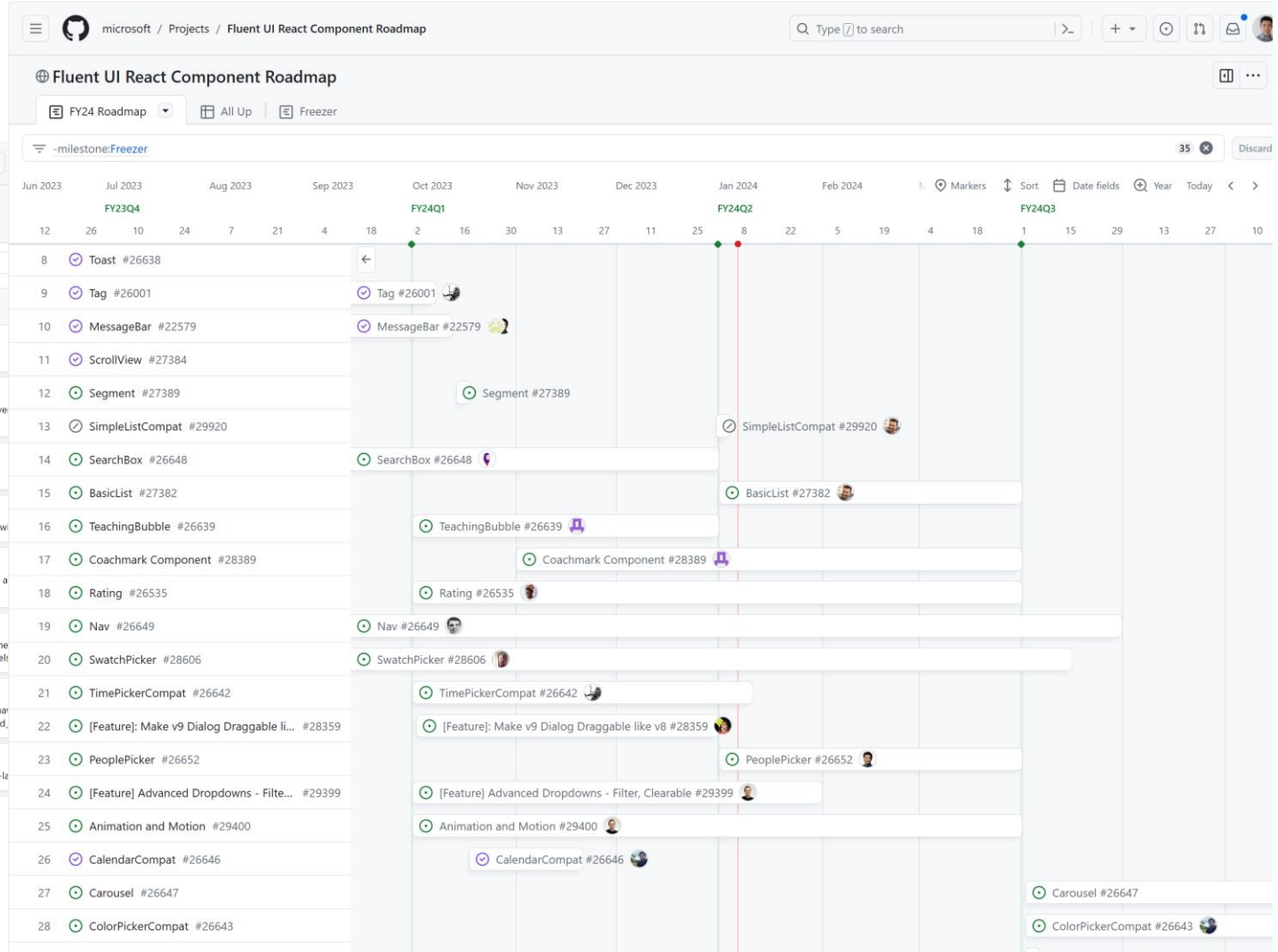
GitHub Issues: Assignees



Working in software teams

GitHub Projects





Why software project management?

Scenario: Alex is a developer for a business-intelligence dashboard project.

- Alex's finishes his big project: upgrading a financials chart
- Avery, the engineering manager wants to make sure:
  - Alex's feature has been peer-reviewed
  - The new charting code integrates well into the dashboard
- Alex wants to make sure his colleagues can understand how to use his code

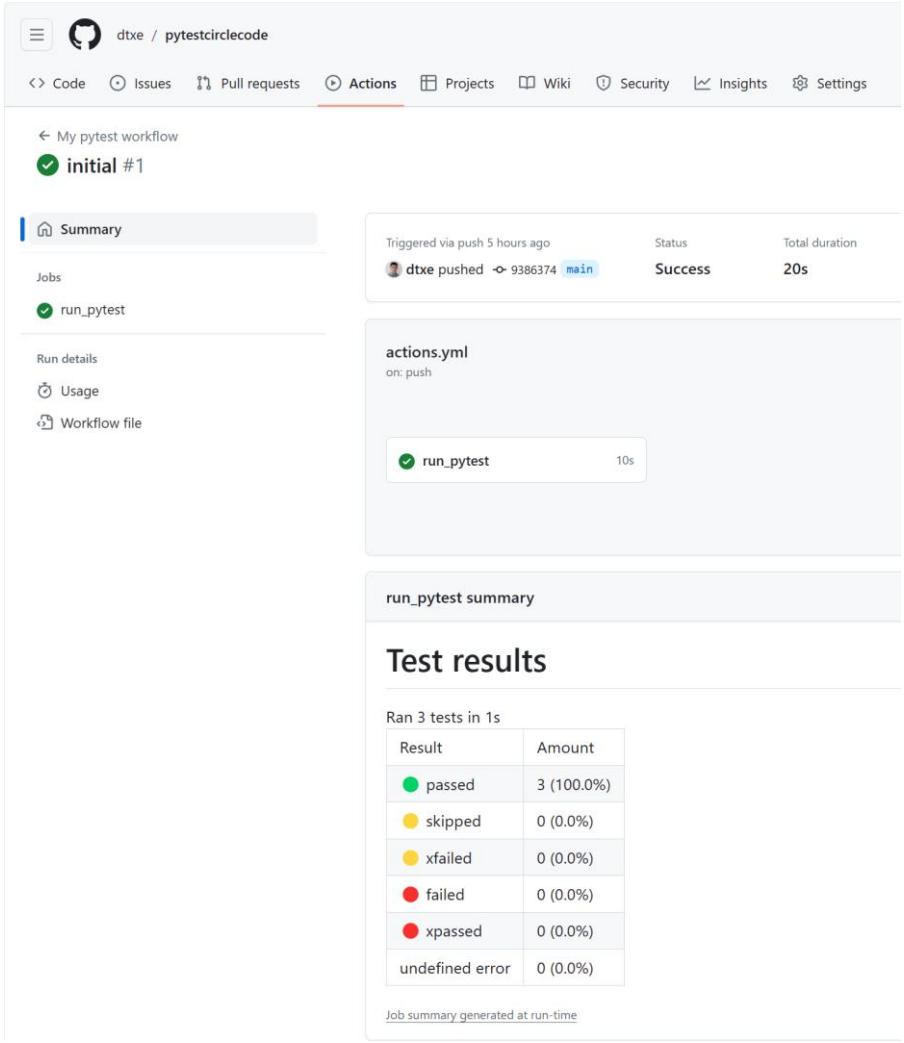
Working in software teams

GitHub

- Continuous Integration
  - Alex's code is automatically tested with `pytest` on GitHub actions
- Code review and approvals
  - Alex can request a colleague to peer review his code and make comments overall, or even line-by-line
- Granular read/write access controls
  - Pull requests on the main repository can have requirements
  - e.g. must be approved by one or more colleagues, must pass all unit tests

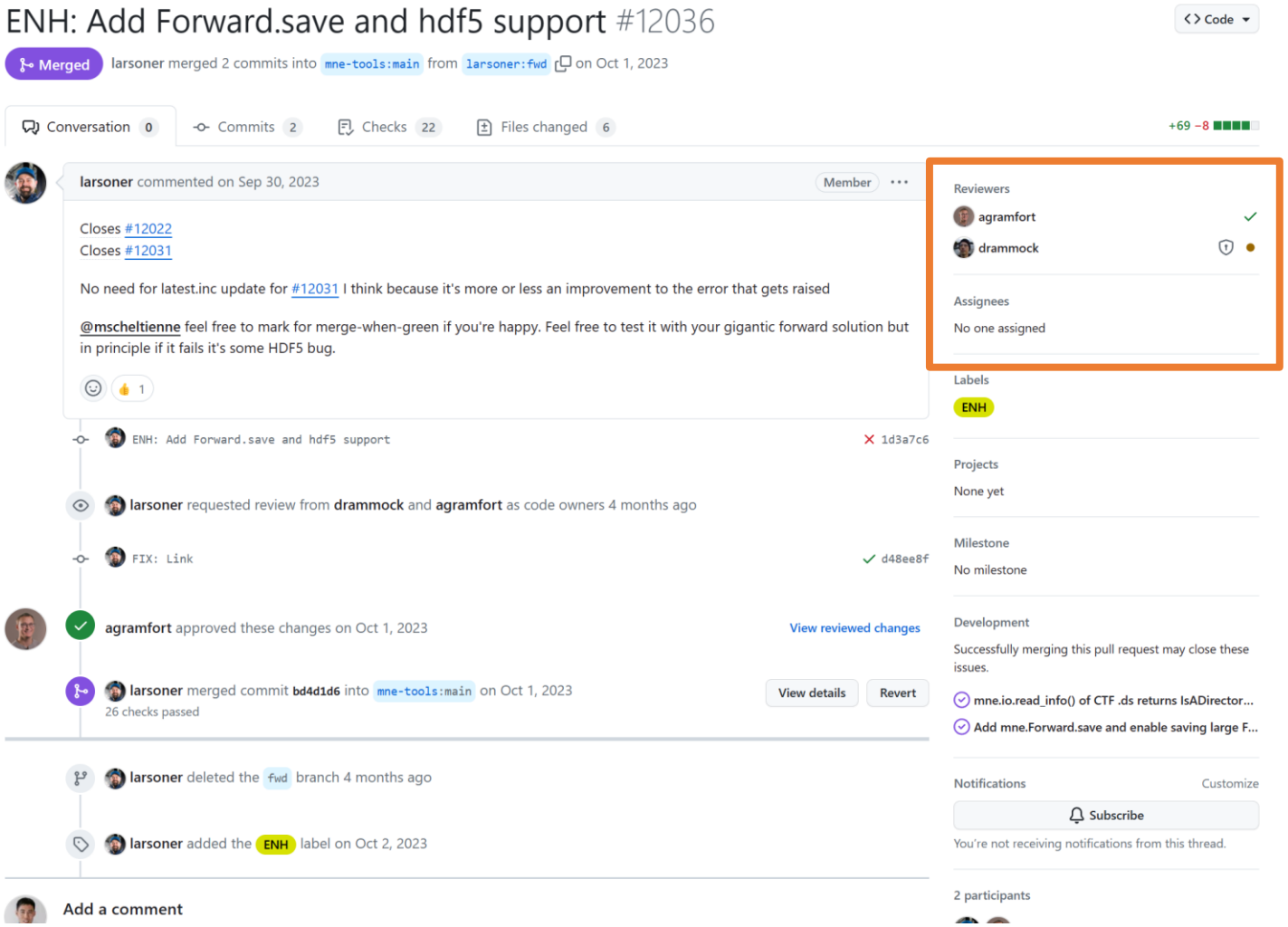
Working in software teams

Continuous integration code testing



Working in software teams

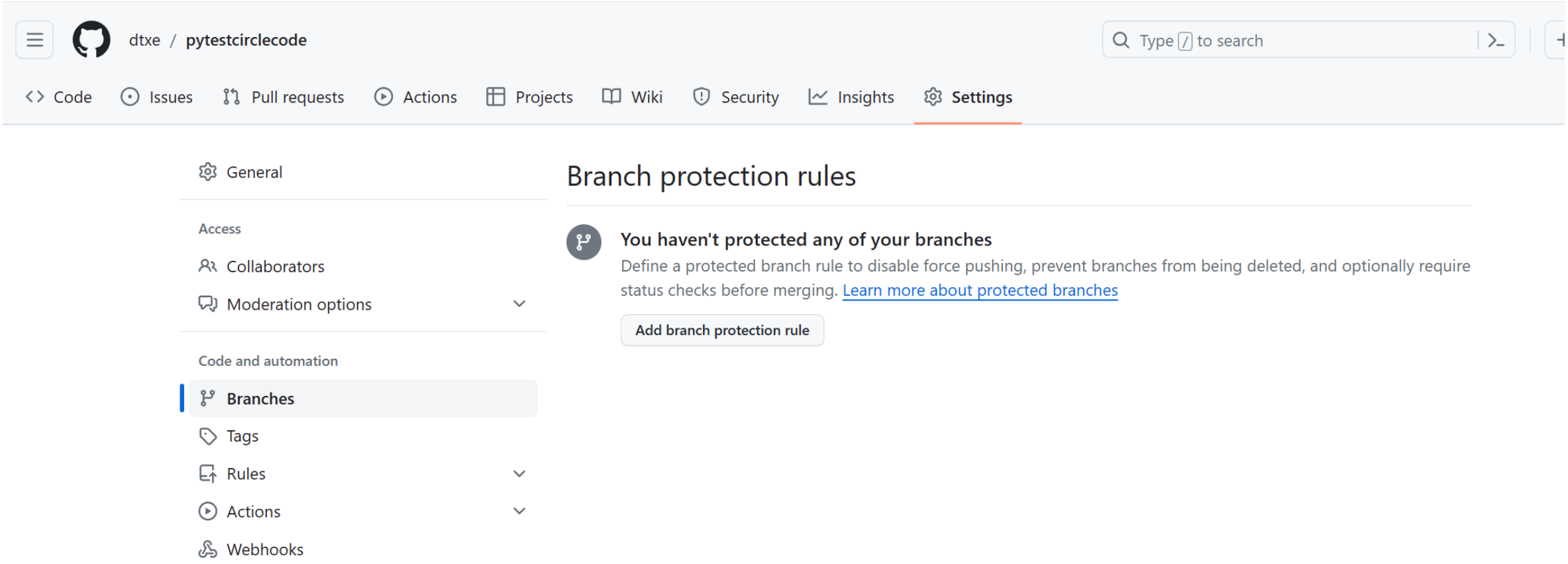
Assigned reviewers and approvers



Working in software teams

Branch protection rules

- Repository settings > Branches > Branch protection rules



Using GitHub features wisely

- GitHub features are part of a comprehensive code quality strategy
- **Recall from code testing:**
  Balance between impact/risk and engineering effort
  - Don’t overburden your contributors and teammates with hoops to jump through if they don’t contribute meaningfully to code quality

Other software project tools

Code repositories

- Atlassian Bitbucket
- GitLab
- AWS CodeCommit
- Azure Repos
- Google Cloud Source
- Self-hosted over SSH / WebDAV

Project tools

- Atlassian Jira
- GitLab
- Redmine
- ClickUp
- Monday.com
- Asana

What questions do we have?

Project governance

What makes up a project?

- A **dataset** being used by several research projects?
    - raw data, programs used to tidy the data, tidied data, text files describing license and provenance
- A **set of annual reports** written for an NGO?
    - jupyter notebooks, copies of html and pdf versions of the report, a text file containing links to datasets used in the report (which can’t be store don Github since they contain personal identifying information)
- A **software library** providing an interactive glossary of data science terms in both Python and R?
    - files needed to create a package, a Markdown full of terms and definitions, a Makefile with targets to check cross references, etc

What makes up a project?

- What people have meetings about
- **Governance:** If the same group needs to get together on a regular basis to talk about something, that *something* probably deserves its own repository
- **Continuity:** And if the list of people changes slowly over time but the meetings continue, that's an even stronger sign

Copyright licenses

- A license dictates how project materials can be used and redistributed
  - Can affect the willingness of contributors
  - The choice of license is crucial to the project's long-term sustainability
- Consider contributors' contractual obligations
  - e.g. students and faculty may have a copyright on the research work they produce, but university staff members may not.
- Including an explicit license avoids legal messiness and should be chosen early on
  - **Generally, licenses do not apply retroactively**

Copyright licenses

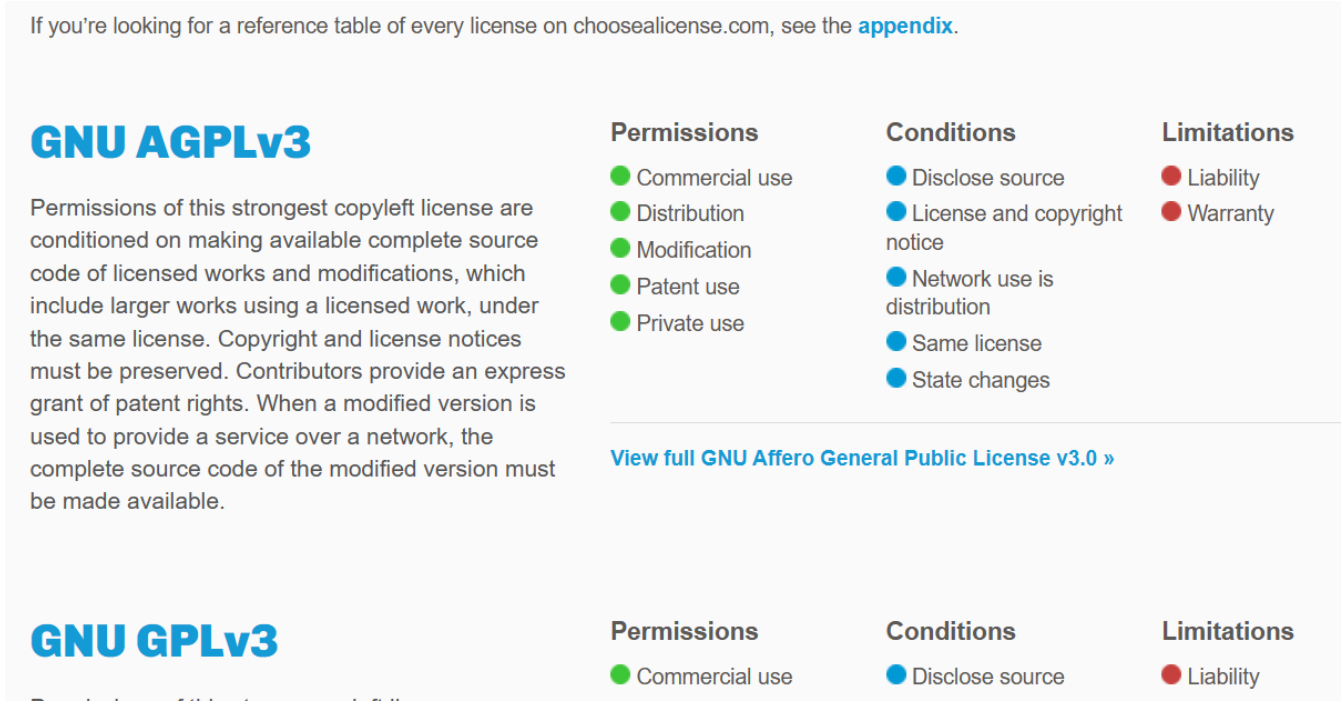
- Do we want to license the work at all?
- Is the content we are licensing source code?
- Do we require people distributing derivative works to also distribute their code?
- Do we want to address patent rights?
- Is our license compatible with the licenses of the software we depend on?
- Do our institutions have any policies that may overrule our choices?
- Are there any copyright experts within our institution who can assist us?

Project governance

Copyright licenses

Great summary here:

https://choosealicense.com/licenses/



The Code of Conduct

1. Promotes fairness within a group
2. Ensures all members that this project takes inclusion seriously
3. Ensures that everyone knows what the rules are
4. Prevents anyone who misbehaves from pretending that they didn't know what the did was unacceptable

Often listed in a file named `CONDUCT.md` in project directory.

Example: https://www.contributor-covenant.org/

Decision making

- Every team has a power structure:
  formal (accountable) or informal (unaccountable).
- Importance of explicit governance in groups larger than six people.
- **Objective**: Establish who makes decisions and how to reach consensus.

Project governance

Decision making: Martha's rules

- Anyone who wants to sponsor a proposal must file one at least 24 hours in advance. It must include:
  - a one-line summary
  - the full text of the proposal
  - any required background information
  - pros and cons
  - possible alternatives
- A quorum is established in a meeting if half or more of voting members are present
- Once a person has sponsored a proposal, they are responsible. The group may not discuss it unless the sponsor is present

Project governance

Decision making: Martha's rules

- After the sponsor presents the proposal, cast a sense vote:
  - Who likes the proposal?
  - Who can live with it?
  - Who is uncomfortable with it?
- If everyone likes or can live with it, it passes with no further discussion.
- If most of the group is uncomfortable, it is sent back to the sponsor for further work. (The sponsor can decide to drop it)
- If some members are uncomfortable, a time is set to discuss, moderated by the meeting moderator.
  - After 10 minutes or so, the moderator calls a yes or no vote.
  - If the majority is yes, it passes.
  - Otherwise, it is returned to sponsor for further work.

Project governance

Decision making: meeting tips

1. Decide if there actually needs to be a meeting.
2. Write an agenda.
3. Include timings in the agenda.
4. Prioritize.
5. Make one person responsible for keeping things moving.
6. Require politeness.
7. No interruptions.
8. No distractions.
9. Take minutes.
10. End early.

Project governance

Document everything!

Make sure your project description, licenses, code of conduct, decision making structures, meeting details are **discoverable** for newcomers!

What questions do we have?

How to write robust software in a team that we, our colleagues, and the public can trust and use with confidence.

Building software summary

1. Track your changes and work collaboratively in Git
2. Use configuration files
3. Handle and raise errors with meaningful messages
4. Document, test, and package your work for others to use

Go forth and build great software! 💪