CodeRefinery in a day

This course provides an introduction to essential practices in research software engineering, with a strong focus on collaboration, reproducibility, and sustainability. Participants will learn the fundamentals of version control and how to effectively collaborate on texts in GitHub. The course also explores key concepts in reproducible research, including organizing projects, a brief overview of environments and containers. In addition, it covers the social aspects of coding, such as software licensing, citation, and open collaboration. By the end, learners will know about best practices for developing and sharing research software more efficiently and transparently.

This course is a condensed version of the CodeRefinery workshop

⚙ Prerequisites

To follow the course, you will need a GitHub account: Instructions

Why is this important?

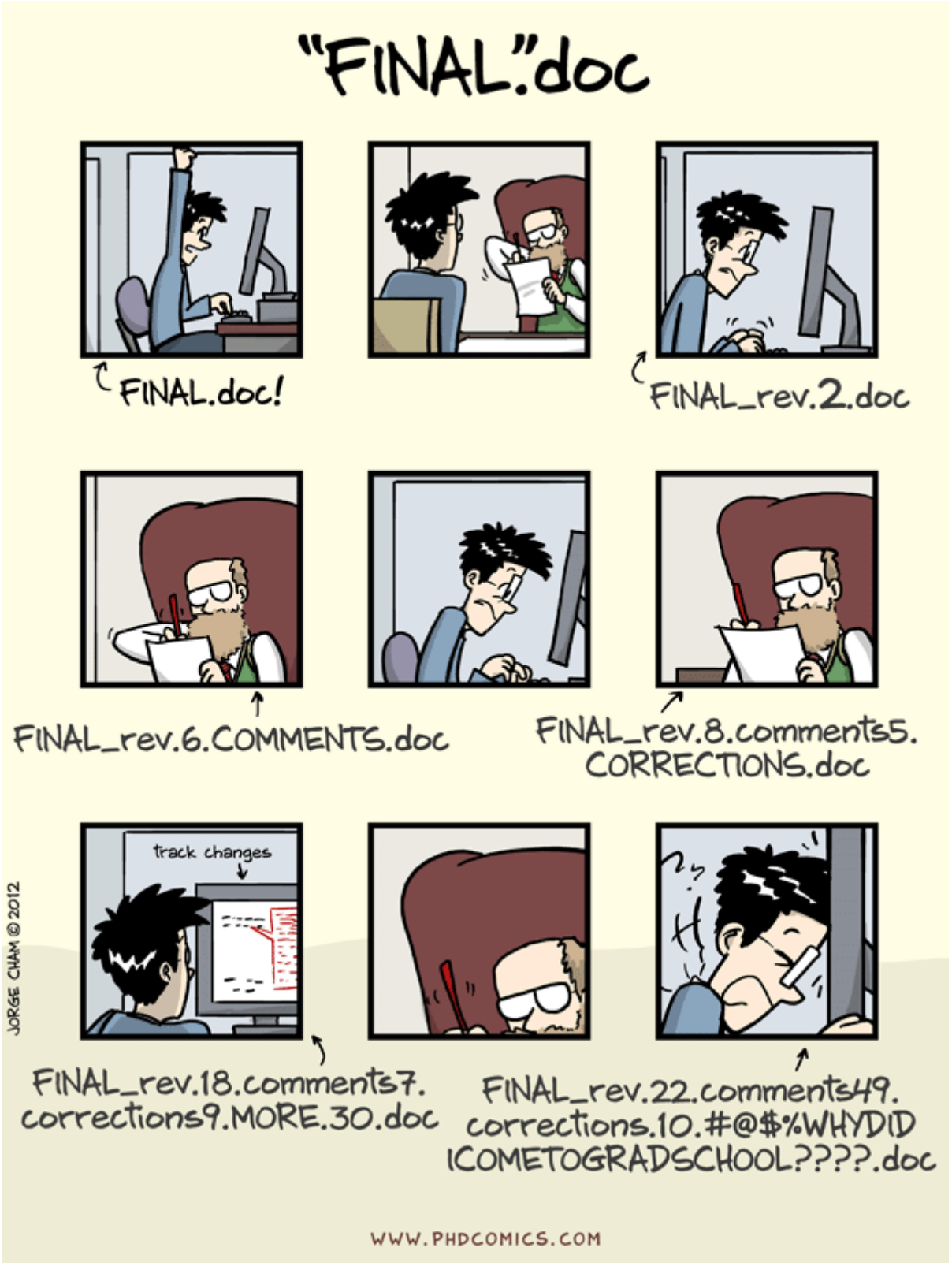
When in doubt if what you are doing is important in research, always reflect using the Allea European Code of Conduct for Research Integrity (2023)

The four Allea principles are:

- **Reliability** in ensuring the quality of research, reflected in the design, methodology, analysis, and use of resources.
- **Honesty** in developing, undertaking, reviewing, reporting, and communicating research in a ***transparent***, fair, full, and unbiased way.
- **Respect** for colleagues, research participants, research subjects, society, ecosystems, cultural heritage, and the environment.
- **Accountability** for the research ***from idea to publication***, for its management and organisation, for training, supervision, and mentoring, and for its wider societal impacts.

Transparency and accountability from idea to publication are fundamental requirements of reproducible research. What we will cover in this course are all the necessary steps needed so that the *computational/methodological/digital* part of the research work is done transparently and allows others to verify and reproduce it.

However, we are not *just* doing this to follow research integrity principles, we are also doing this to improve how we work with other collaborators, and especially **we do this to help our future self** so that it can always be possible to travel back in time and see what was done at a certain part of the research process.



(Source)

Schedule

Please note that some of the materials are hosted in other repositories. Make sure to always come back here to find the next link.

9:15-10:00	Introduction to Version Control motivation and sharing
10:00-10:10	Break
10:10-11:00	Collaborating with git
11:00-12:00	*Lunchbreak*
12:00-12:50	Collaborating with git exercise
12:50-13:00	Break
13:00-14:00	**Reproducibility and code**:
5 min	Reproducible research intro
10 min	Reproducible research motivation
15 min	Organizing projects
15 min	Environments and containers
14:00-14:10	Break
14:10-14:45	**Social coding**:
10 min	Social coding
10 min	Software licensing with cake
5 min	Software licensing
10 min	Software citation
14:45-15:00	Wrapup

Getting ready for this course

GitHub account

In this course, we use the public GitHub service and you need an account at https://github.com and a supported web browser. Basic GitHub accounts are free.

If you are concerned about the personal information to reveal to GitHub, for example how to keep your email address private, please review these instructions for keeping your email address private provided at GitHub.

We are trying to make it possible to follow this course also with other Git hosting services but this is work in progress.

Create a GitHub account

1. Go to https://github.com.
2. Click on the “Sign up” at the right-top corner.

3. Enter your username of your choice (if it is already used, you will get some suggestions), email address, and password.
4. Follow further instruction and verify your account.

GitHub may require you to enable multi-factor authentication (MFA). This is generally a good thing, but may take some time to set up. Luckily, you probably don't have to do this immediately. If you are prompted to enable MFA before the end of CodeRefinery, follow GitHub's instructions since they are usually pretty good.

How to verify that this worked

If you can log in to https://github.com, you should be good to go.

Using git to collaborate

Objectives

- Understanding the distributed nature of git collaborations
- Learning about the key terms
- Being able to suggest changes to repositories you do not own

Terminology: Commits, branches, repositories, forks, clones

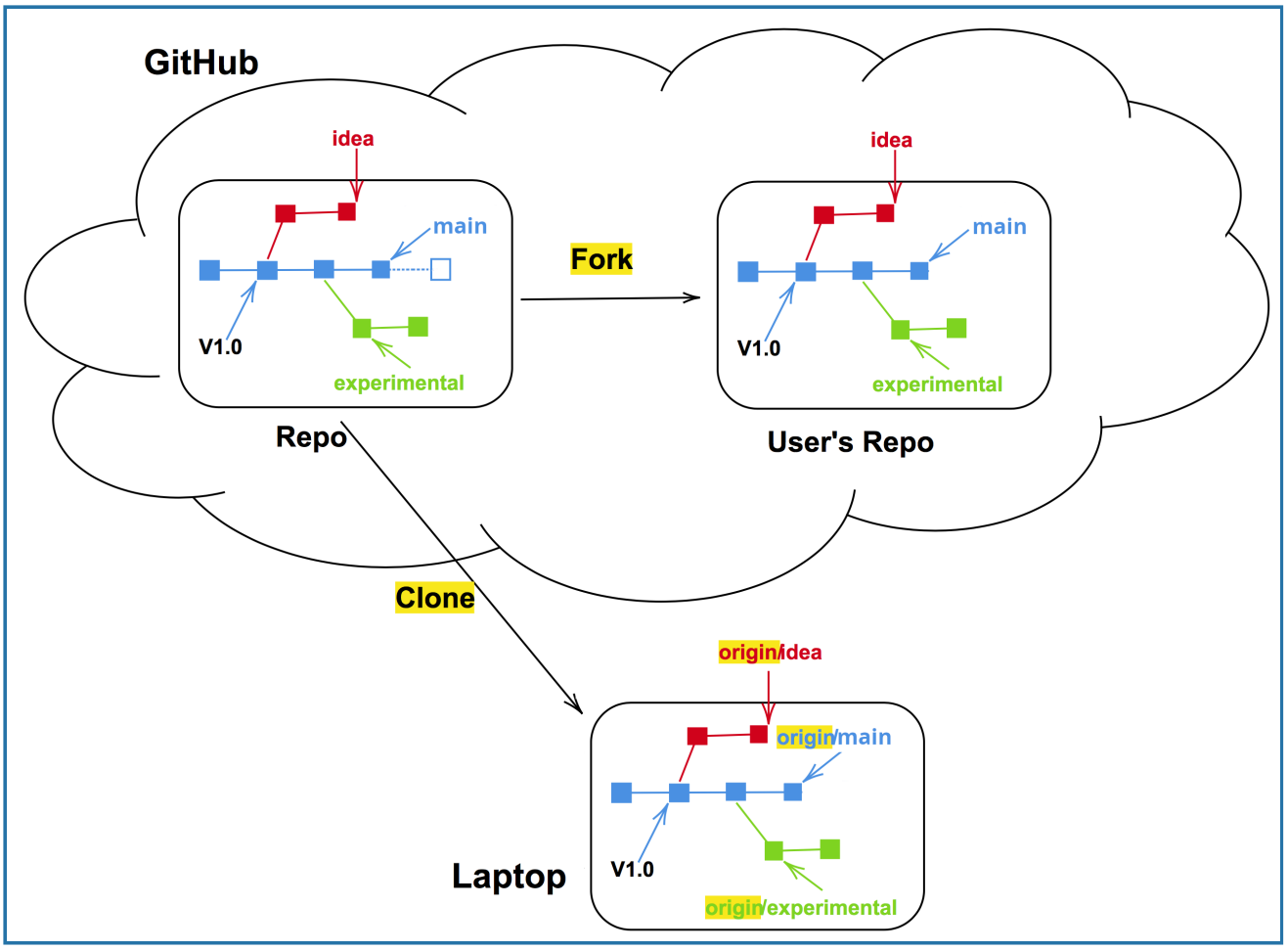
- repository: The project, contains all data and history (commits, branches, tags).
- commit: Snapshot of the project, gets a unique identifier (e.g. `c7f0e8bfc718be04525847fc7ac237f470add76e`).
- branch: Independent development line. The main development line is often called `main`. Technically, a branch in Git is implemented as a pointer to a commit (imagine a sticky note with a branch name on it).
- tag: A pointer to one commit, to be able to refer to it later. Like a commemorative plaque that you attach to a particular commit (e.g. `phd-printed` or `paper-submitted`).
- cloning : Copying the whole repository to your laptop - the first time. It is not necessary to download each file one by one.
- forking : Taking a copy of a repository (which is typically not yours) - your copy (fork) stays on GitHub/GitLab and you can make changes to your copy.

Cloning a repository

In order to make a complete copy a whole repository, the `git clone` command can be used. When cloning, all the files, of all or selected branches, of a repository are copied in one operation. Cloning of a repository is of relevance in a few different situations:

- Working on your own, cloning is the operation that you can use to create multiple instances of a repository on, for instance, a personal computer, a server, and a supercomputer.

- The parent repository could be a repository that you or your colleague own. A common use case for cloning is when working together within a smaller team where everyone has read and write access to the same Git repository.
- Alternatively, cloning can be made from a public repository of a code that you would like to use. Perhaps you have no intention to change the code, but would like to stay in tune with the latest developments, also in-between releases of new versions of the code.



Forking and cloning

Forking a repository

When a fork is made on GitHub/GitLab a complete copy, of all or selected branches, of the repository is made. The copy will reside under a different account on GitHub/GitLab. Forking of a repository is of high relevance when working with a Git repository to which you do not have write access.

- In the fork repository commits can be made to the default branch (`main` or `master`), and to other branches.
- The commits that are made within the branches of the fork repository can be contributed back to the parent repository by means of **pull or merge requests**.

Demo - forking a famous repository

A demo with forking a famous repository.

Exercise - suggesting a change on a repository you do not own

This exercise is the typical case where you find something that could be improved in a repository and you suggest a change. Compared to other scenarios (like a wiki) you do not need to have permissions to suggest a change. Furthermore, **your change will not overwrite anything done by others**, until the change is **reviewed and approved**.

✍️ Exercise: Add a new recipe to somebody else's cookbook

We work with this recipe book repository. This will be the **upstream** repository.

Exercise tasks:

1. Log in to GitHub.com and open the link above
2. Click on **Issues** and then click the button **New issue** to create a new issue in the **upstream repository**. In the issue, you describe the change you want to make. Example: "Title: avocado pasta. Content: I will add a recipe for pasta with avocado". Take note of the **issue number**: you will need it later.
3. Click on **Code** To see the repository structure. Find a subfolder where you want to add your new recipe and click on the subfolder. For a pasta recipe you will visit the subfolder **pasta**
4. Click on **Add file** and then "+ Create new file".
5. You will be asked to **fork** this repository to propose changes. Click on the green button **Fork this repository**
6. A file editor now opens and it will say "You're making changes in a project you don't have write access to. Submitting a change will write it to a new branch in your fork **eglereanwebdev/data-stewards-recipe-book**, so you can send a pull request."
7. Give your new file a name and write the recipe
8. Click the green button "**Commit changes...**". The commit window will pop up. Describe the change you did.
9. GitHub will redirect you to a comparison of the changes you did in your **forked repository** versus the **upstream** repository.
10. You can now open a pull request towards the upstream repository by clicking the green button **Create pull request**
11. In the page "Open a pull request" you will asked to give a title and explain what you did. Make sure you add "Closes #N" in the text, with `N` being the number of the issue you opened to explain what you were going to do. When ready, click the green button "**Create pull request**"
12. The project owner will merge the pull requests OR will ask for changes.
13. (Optional) After few pull requests are merged, yuo can update your fork with the changes from others.
14. (Optional) Check that in your fork you can see changes from other people's pull requests.
15. (Optional) In step 4, you could have also edited a file instead of adding a new file. You can look for typos and suggest improvements.

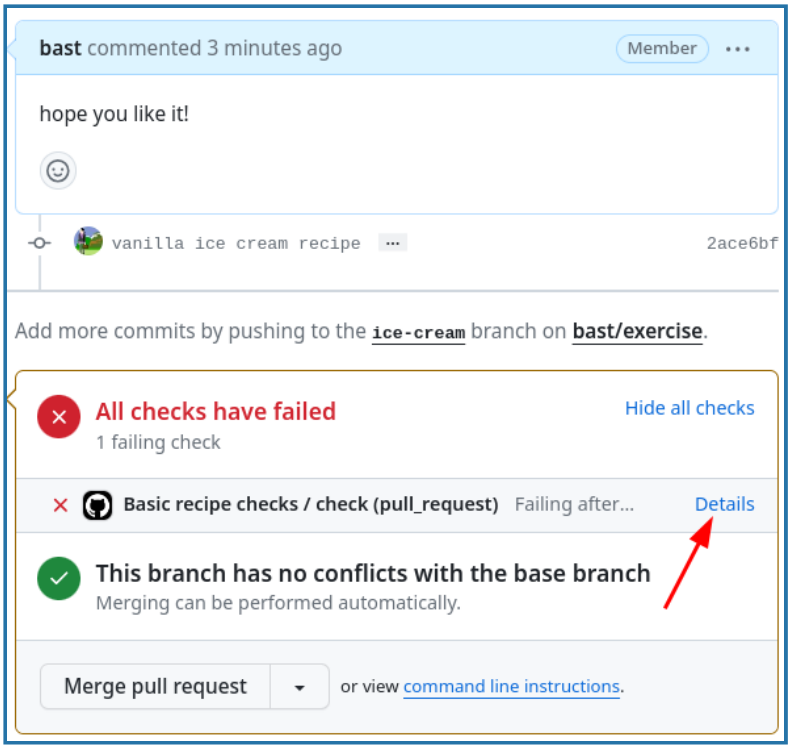
Advanced topics

Pull requests can be coupled with automated testing

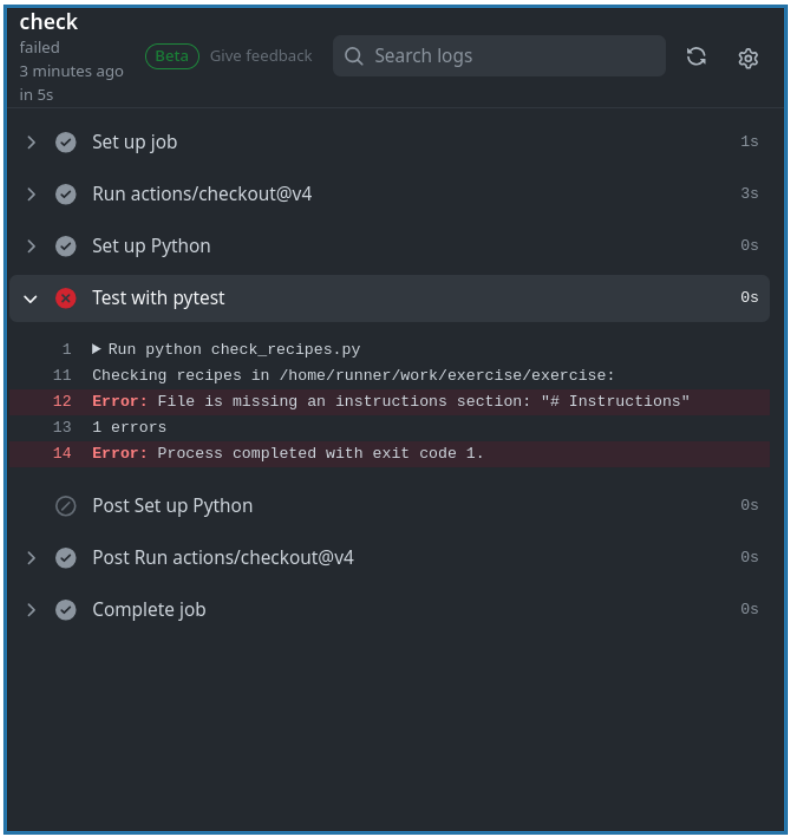
We added an automated test here just for fun and so that you see that this is possible to do.

In this exercise, the test is silly. It will check whether the recipe contains both an ingredients and an instructions section.

In this example the test failed:



Click on the “Details” link to see the details of the failed test:



How can this be useful?

- The project can define what kind of tests are expected to pass before a pull request can be merged.
- The reviewer can see the results of the tests, without having to run them locally.

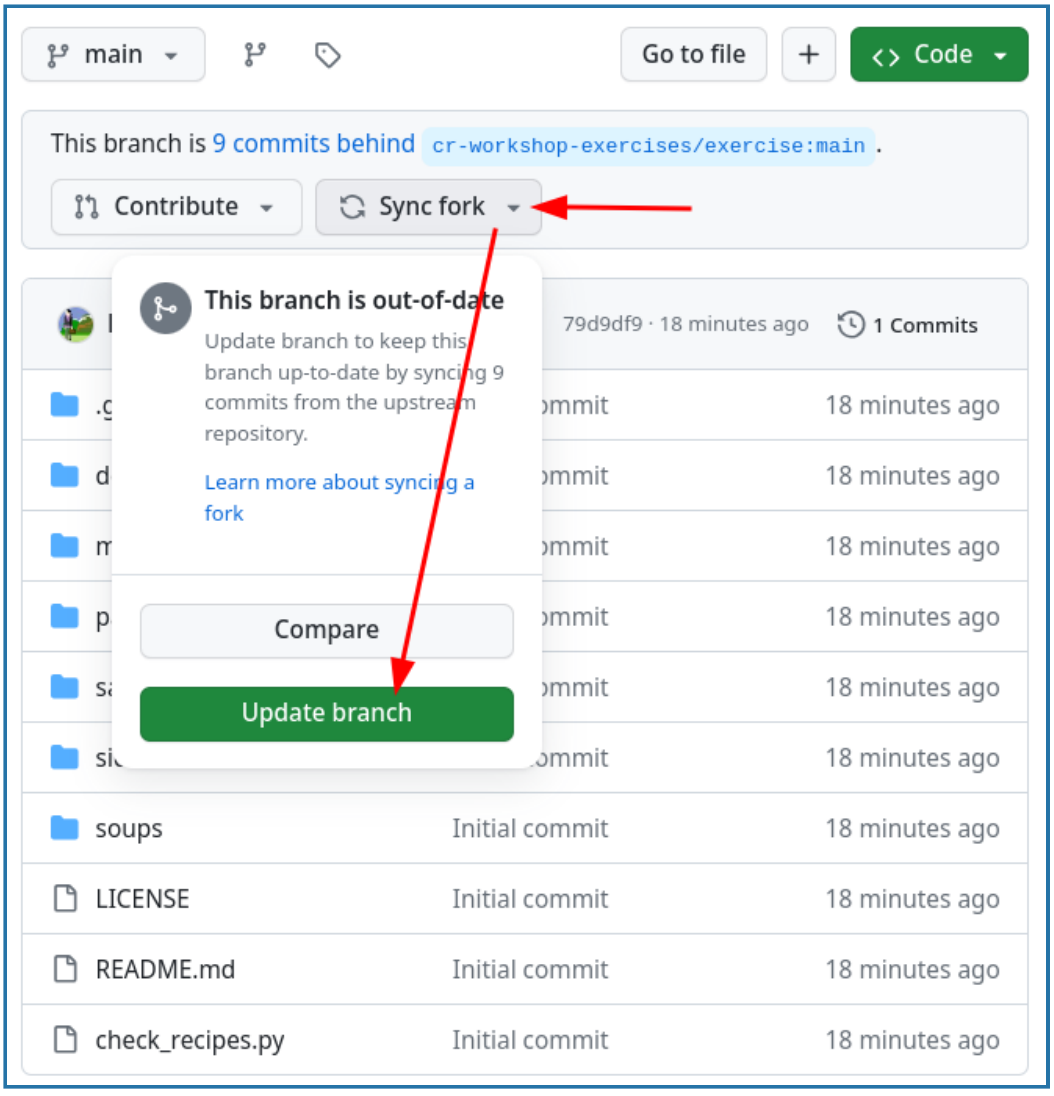
How does it work?

- We added a GitHub Actions workflow to automatically run on each push or pull request towards the `main` branch.

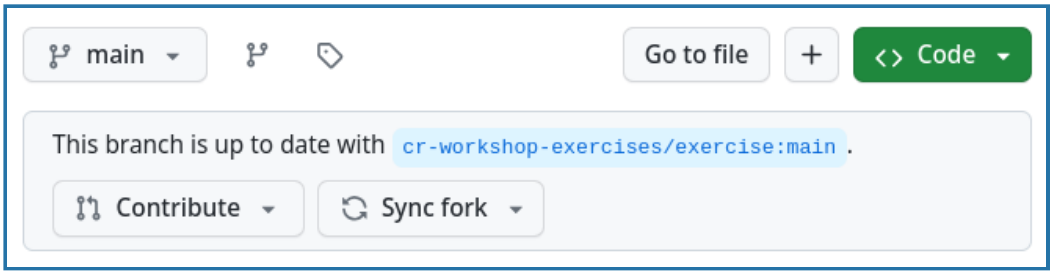
What tests or steps can you image for your project to run automatically with each pull request?

How to update your fork with changes from upstream

This used to be difficult but now it is two mouse clicks: Navigate to your fork and notice how GitHub tells you that your fork is behind. In my case, it is 9 commits behind upstream. To fix this, click on “Sync fork” and then “Update branch”:



After the update my “branch is up to date” with the upstream repository:



How to approach other people's repositories with ideas, changes, and requests

Contributing very minor changes

- Clone or fork+clone repository
- Create a branch
- Commit and push change
- Open a pull request or merge request

If you observe an issue and have an idea how to fix it

- Open an issue in the repository you wish to contribute to
- Describe the problem
- If you have a suggestion on how to fix it, describe your suggestion
- Possibly **discuss and get feedback**
- If you are working on the fix, indicate it in the issue so that others know that somebody is working on it and who is working on it
- Submit your fix as pull request or merge request which references/closes the issue

Motivation

- **Inform others about an observed problem**
- Make it clear whether this issue is up for grabs or already being worked on

If you have an idea for a new feature

- Open an issue in the repository you wish to contribute to
- In the issue, write a short proposal for your suggested change or new feature
- Motivate why and how you wish to do this
- Also indicate where you are unsure and where you would like feedback
- **Discuss and get feedback before you code**
- Once you start coding, indicate that you are working on it
- Once you are done, submit your new feature as pull request or merge request which references/closes the issue/proposal

Motivation

- **Get agreement and feedback before writing 5000 lines of code** which might be rejected
- If we later wonder why something was done, we have the issue/proposal as reference and can read up on the reasoning behind a code change

Summary

- This forking workflow lets you propose changes to repositories for which **you have no write access**.
- This is the way that much modern open-source software works.
- You can now contribute to any project you can view!

Wrap up and where to go from here

Explore other courses and networks in scientific computing

Aalto Scientific Computing: https://scicomp.aalto.fi/

- Daily session for questions
- Research Software Engineers provide close support for anything we have covered in this workshop: https://scicomp.aalto.fi/rse/

CSC - IT Center for Science (Finland): https://research.csc.fi/

- Weekly user support session
- Training calendar
- Check out the Scientific Computing Ambassadors
- Check out the **data support network**
- Join the data support network

The Carpentries: https://carpentries.org/

- Public reusable lesson programs & workshops: Data -, Software and Library Carpentry
- Explore e.g. Software lesson curriculum: https://software-carpentry.org/lessons/
- Many more lessons available in the incubator
- Instructor- and lesson development training available for a fee

CodeRefinery: https://coderefinery.org/

- Bi-yearly workshops on Research Software Development practices; next workshop autumn '25
- Materials available for self-learning and reuse: https://coderefinery.org/lessons/
- Instructor training materials

Follow CodeRefinery and spread the word <3

- Mastodon: @coderefinery@fosstodon.org.
- BlueSky: @coderefinery.org
- LinkedIn: CodeRefinery
- **Become a CodeRefinery ambassador** by signing up to our ambassador mailing list
- **Join as an Team Leader / local host.** Bring your group/classroom, learn together.
- **Become part of the team** CodeRefinery lives from **in-kind** contributions by organizations (your organization sponsors your worktime to the project)
- CodeRefinery community lives in the **Zulip chat**: https://coderefinery.zulipchat.com
- Follow what we do by signing up to our **newsletter**: https://coderefinery.org/#newsletter

Got interested in Research Software Development?

Join the Nordic Research Software Engineers (Nordic-RSE)

A **Research Software Engineer** combines research knowledge with software development experience, just like we have learned here.

- Community, seminars, and other events.
- Chat (part of CodeRefinery chat): https://coderefinery.zulipchat.com, `#nordic-rse`
- LinkedIn: https://www.linkedin.com/company/nordic-rse/
- Mastodon: https://fosstodon.org/@nordic_rse
- BlueSky: https://bsky.app/profile/nordic-rse.bsky.social

Join us for second in-person conference May 20.+21. 2025 in Gothenburg Sweden: https://nordic-rse.org/nrse2025/

Quick Reference

Instructor's guide

To be filled soon.

Why we teach this lesson

Intended learning outcomes

Timing

Preparing exercises

e.g. what to do the day before to set up common repositories.

Other practical aspects

Interesting questions you might get

Typical pitfalls

Who is the course for?

This course is for you, if you are in a position where you might want to support researchers with programming and its reproducibility.

This course is not designed for “professional software engineers” or to make you one.

About the course

In this course, you will get a quick overview of tools and best practices for research software development. This course will not teach a programming language, but we aim to show you some tools and techniques to be able to guide researchers to do programming well and avoid common inefficiency traps. The tools we teach are practically a requirement for any researcher who needs to write code. The main focus is on using Git for efficiently writing and maintaining research software or other content.

See also

All lessons of a full CodeRefinery workshop

You can also find recordings of the full lessons on Youtube:

- Version Control: Intro pt 1 , intro pt 2, collaborative version control
- Reproducible research
- Social coding

Full playlist on youtube, inlcuding also Documentation, Jupyter, modular code develeopment and automated testing.

Credits