



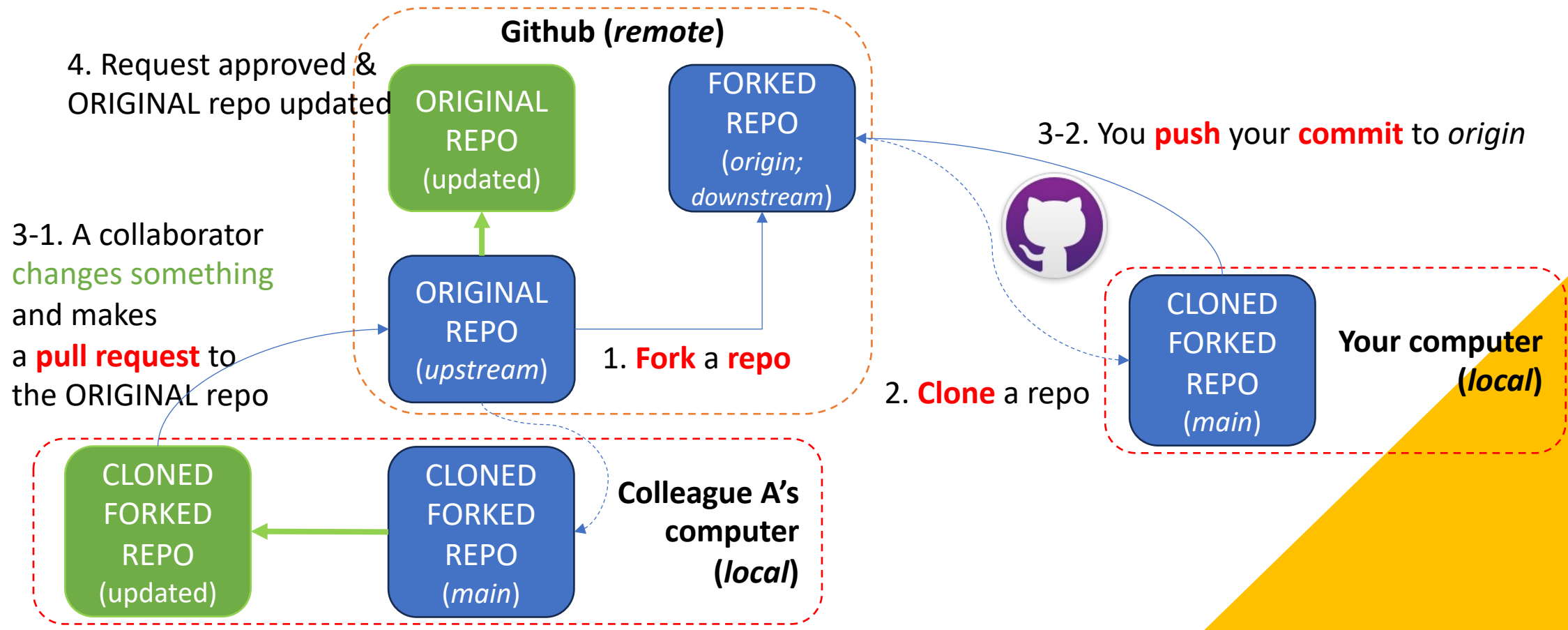
Week 2: Tools for reproducible science II

ReproRehab POD1, Week 2

Agenda

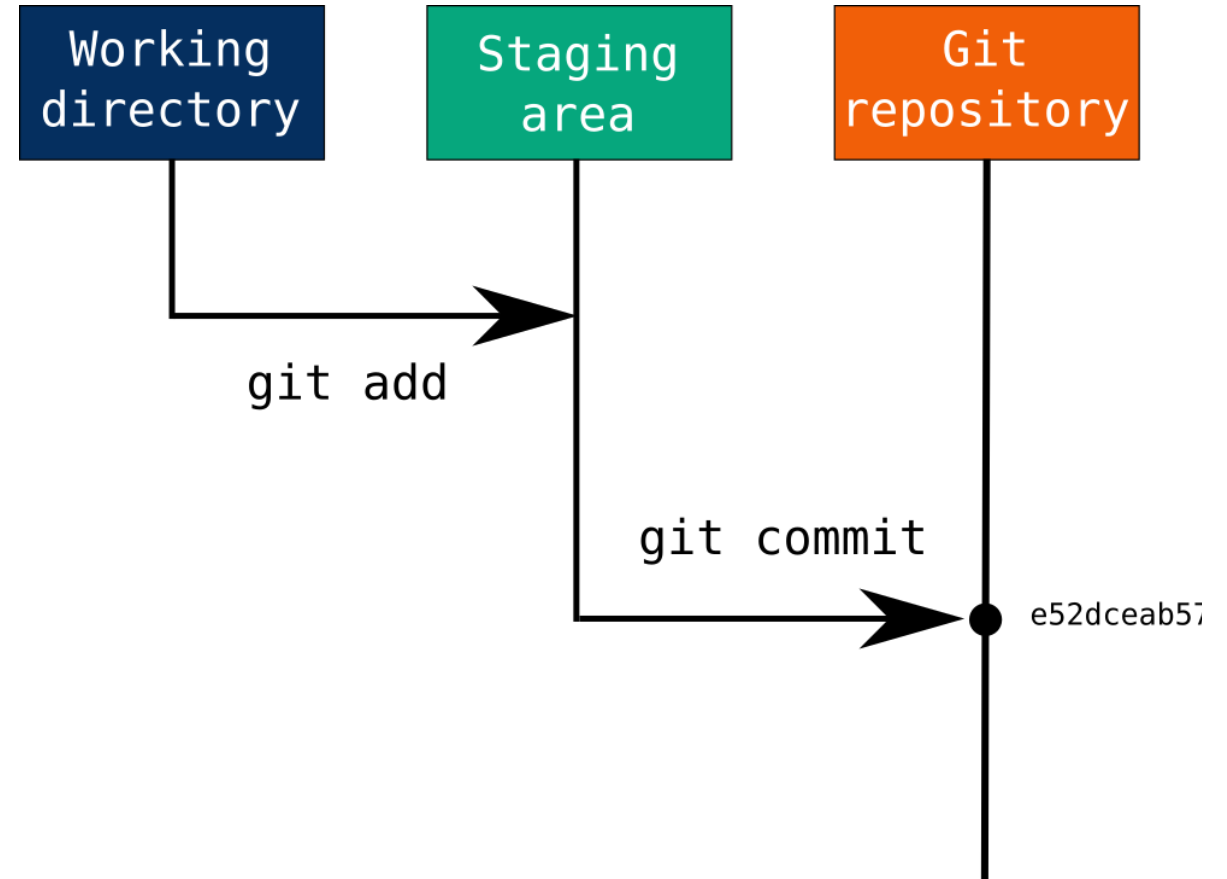
- Review: collaborating with Git/Github
- Use Github Desktop to:
 - 1) clone your forked repository to your computer
 - 2) upload (*add*, *commit* and *push*) files to your remote repository
 - 3) make a new repository, make a new branch, and merge branches
 - public vs. private repository
 - inviting collaborators to work together
- Data cloning using Datalad / Repos to share data

Collaborating with Git/Github

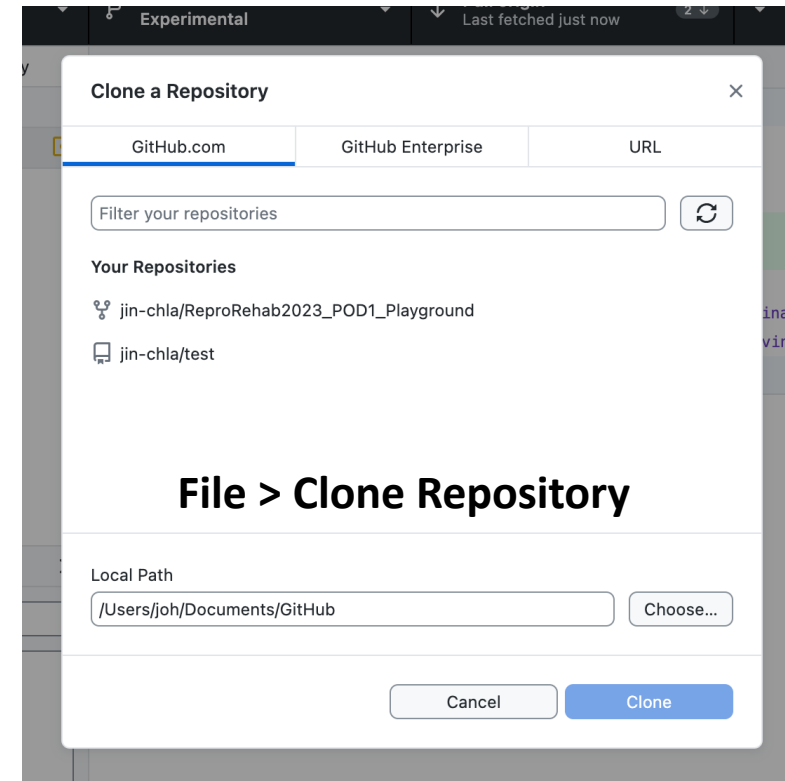
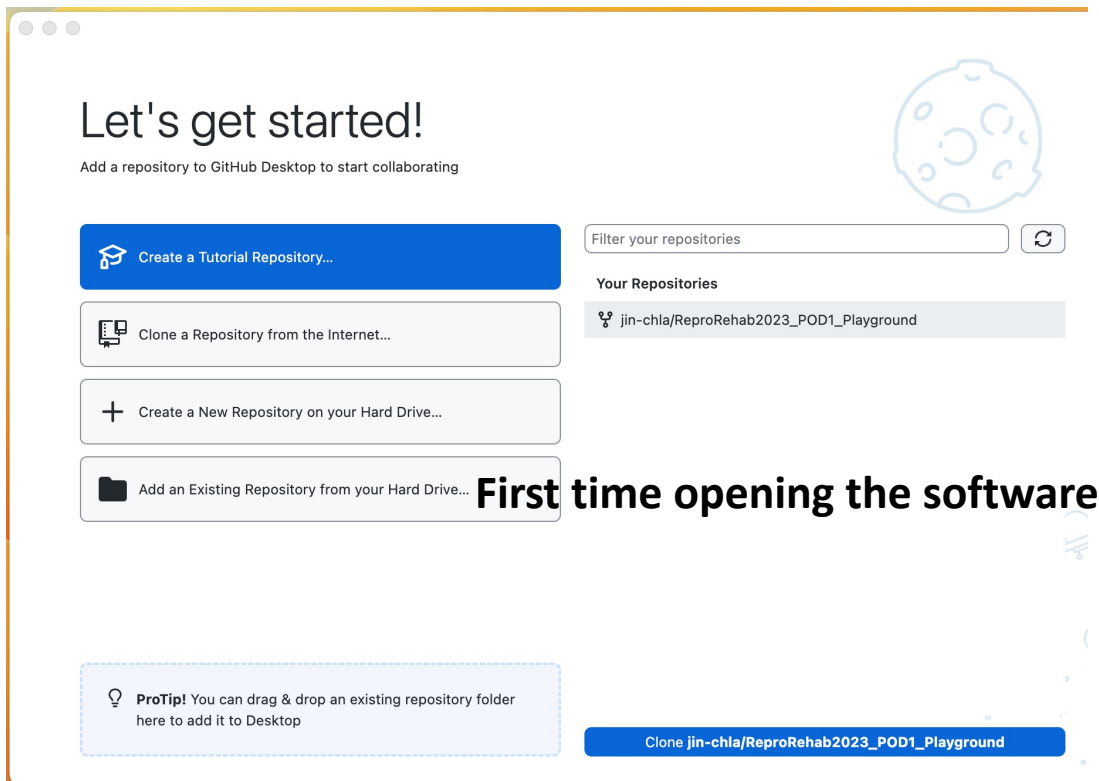


Add, commit and push changes

- Technically, a repository is a folder named **.git** (This is hidden, so not easy to access)
- Changes you make on your computer is added to the staging area (done automatically in Github Desktop)
- You then commit the staged changes to your local git repository
- Finally you push your commit from the local to remote repository (the one in Github.com)

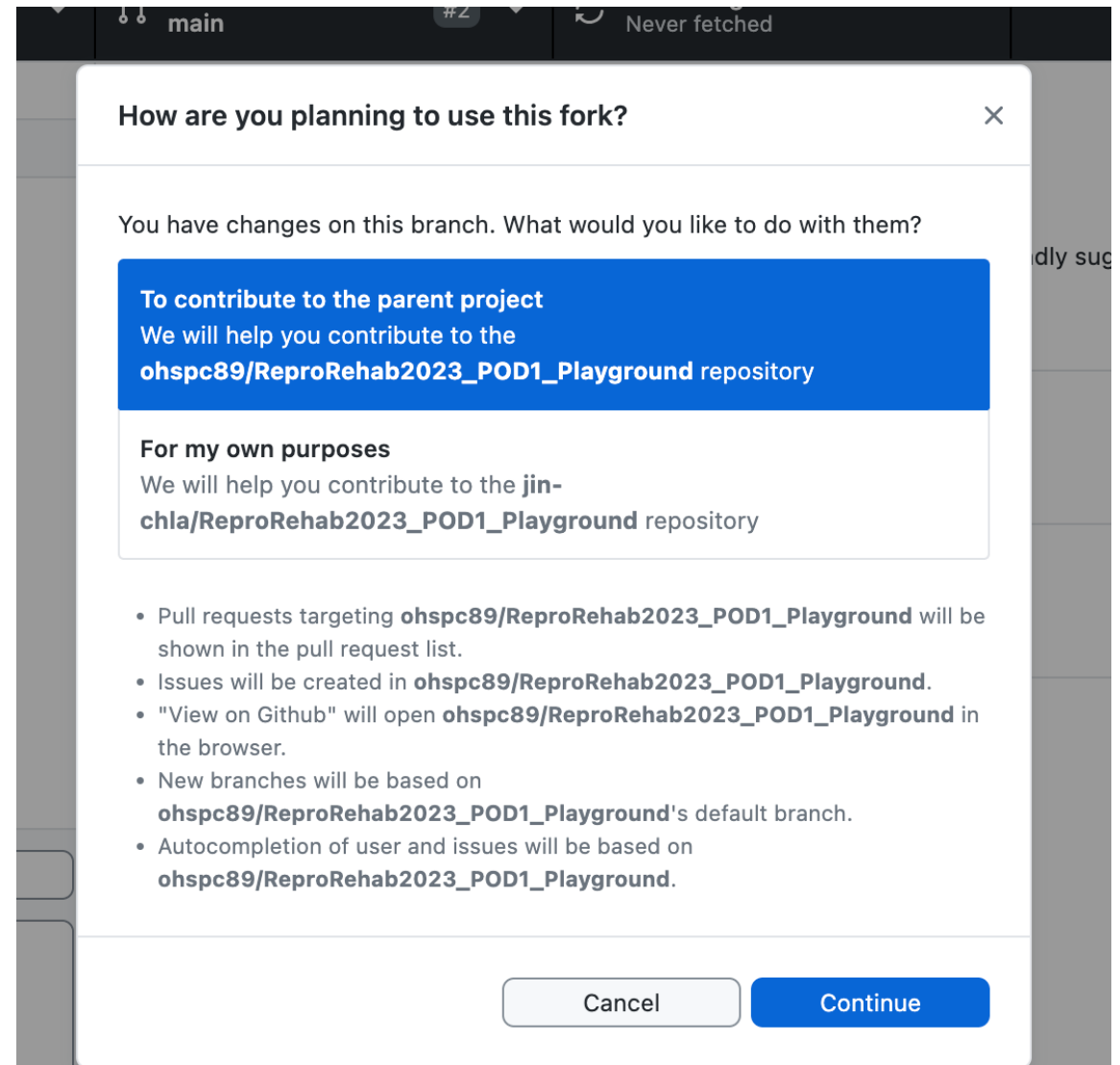


Cloning a repository

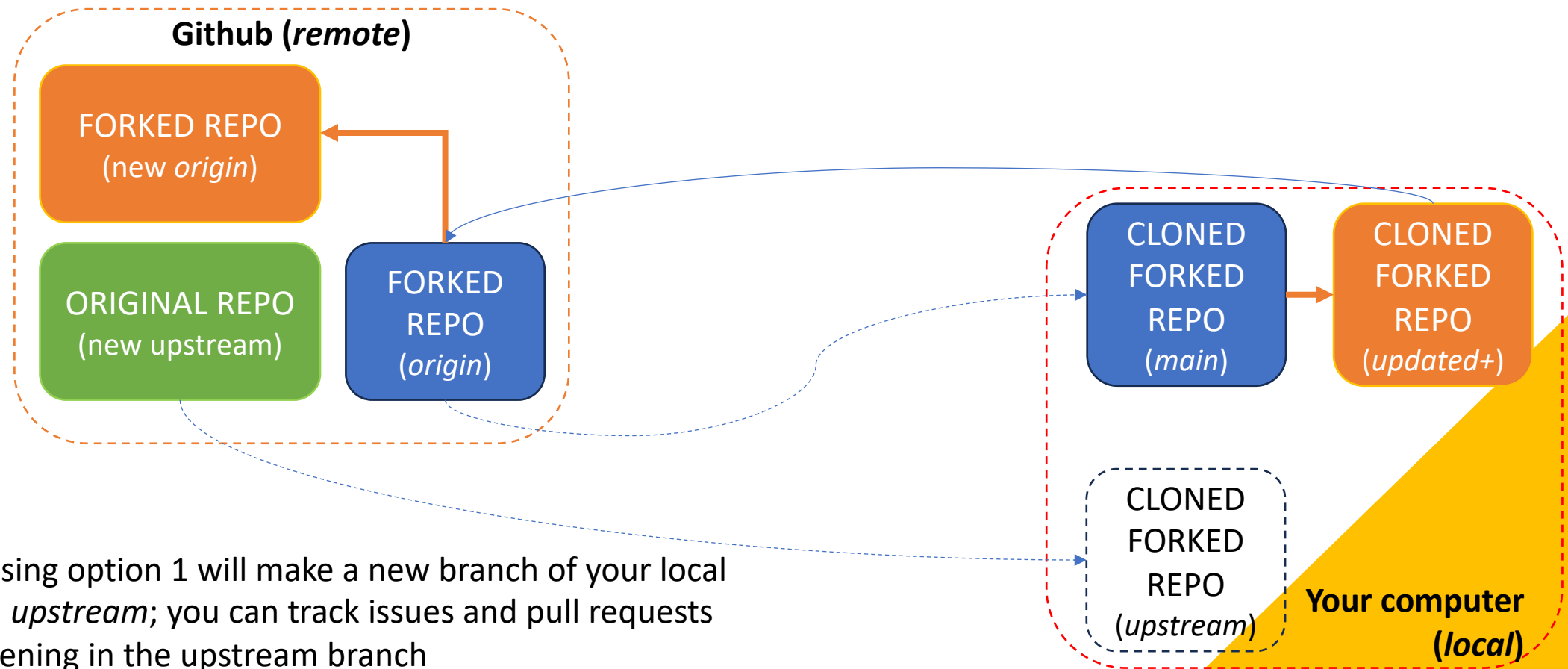


Cloning a repository

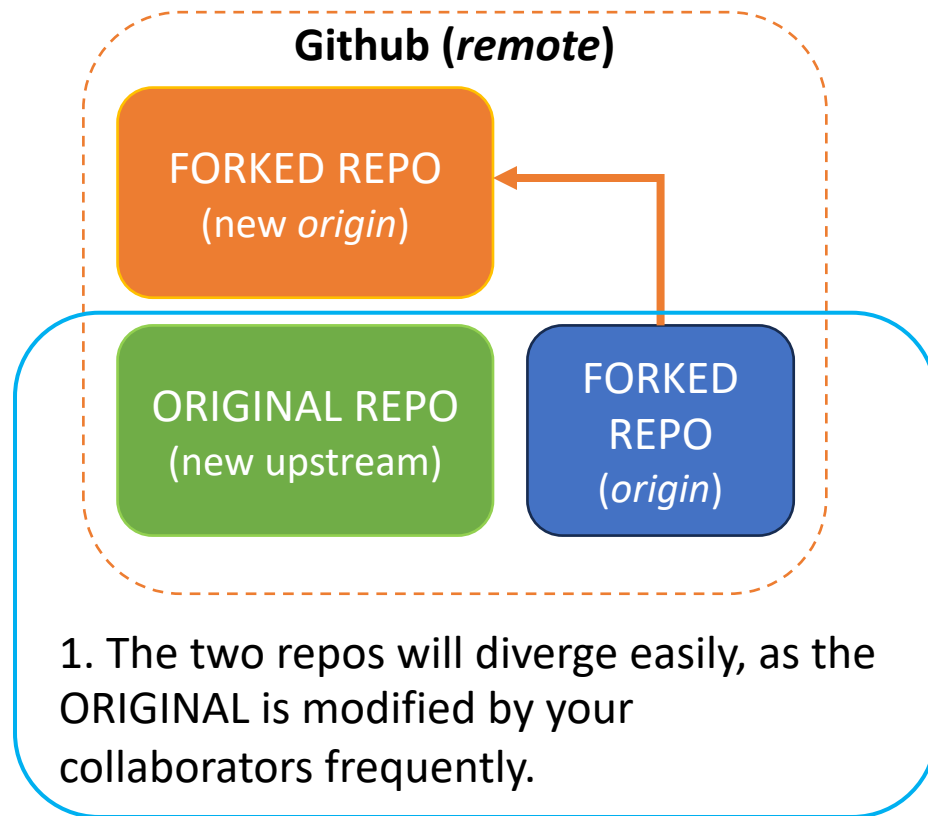
- If you clone a *forked* repository, you will see the screen on the right.
- See the next slide for more details.



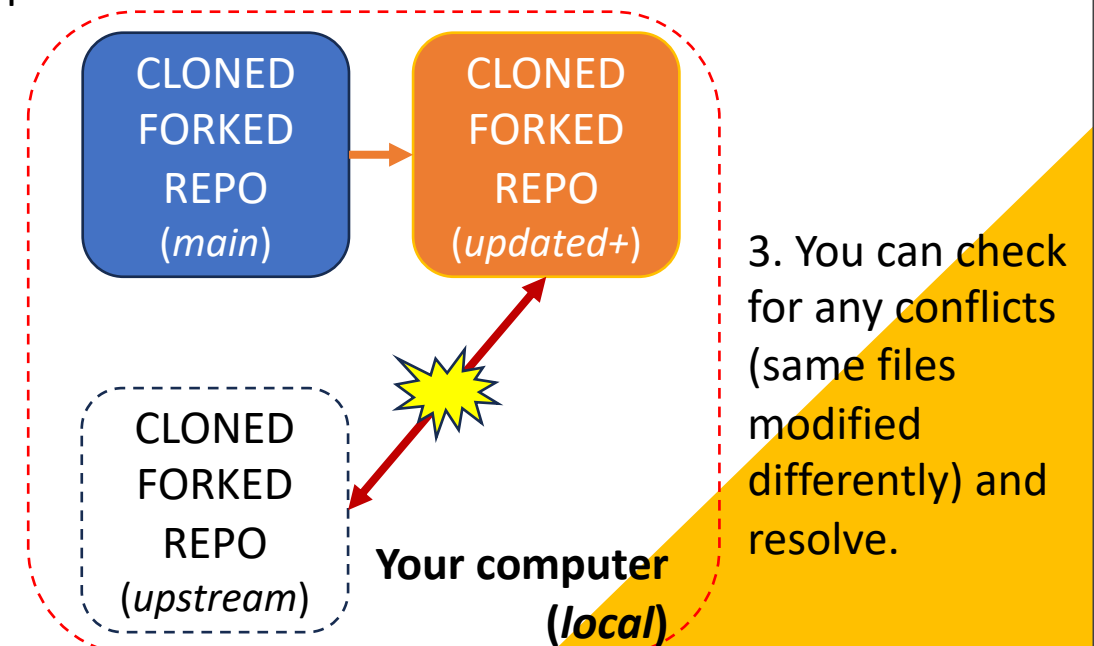
Fork option 1: prioritize the parent (upstream)



Fork option 1: prioritize the parent (upstream)



2. While working on your local machine, you can try *merging* branches to catch up with others



Example of conflict

- README.md file

[Original version in upstream]

“This is a repository prepared for ReproRehab2023 POD 1 learners. Please fork this repository, make edits in your branch, and make pull requests!”

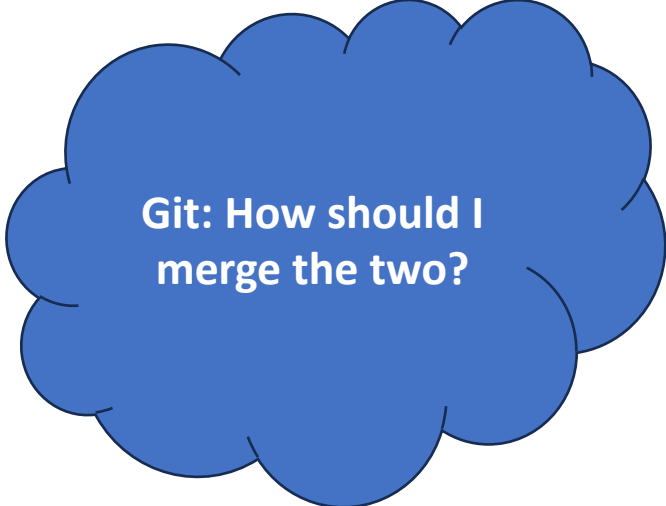
[Your colleague’s edit saved on upstream]

“This is a repository prepared for ReproRehab2023 POD 1 learners. Please fork this repository, make edits in your branch, and make pull requests!”

I hope you enjoy working with Git/Github!

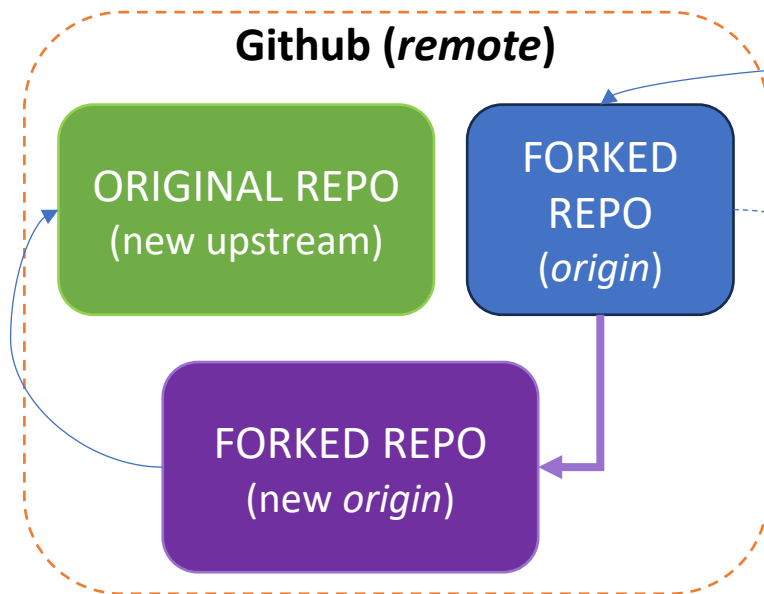
[Edits you made in your local main branch]

“This is a repository prepared for ReproRehab2023 POD 1 learners. Please fork this repository, make edits in your branch, and make pull requests! \n\n Date added: 10/5/2023 \n\n Adding anotherline here \n\n LALALALALA”



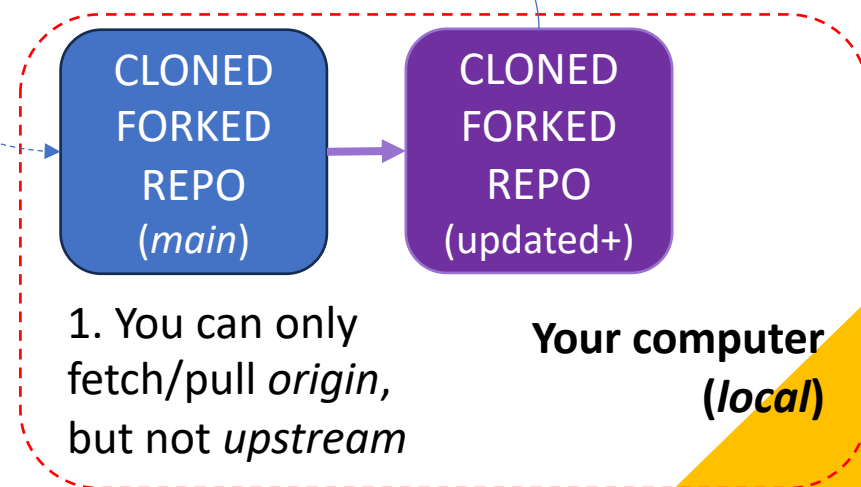
Git: How should I merge the two?

Fork option 2: prioritize the forked repo (origin)



3. Any conflict will be resolved later when you make a pull request here

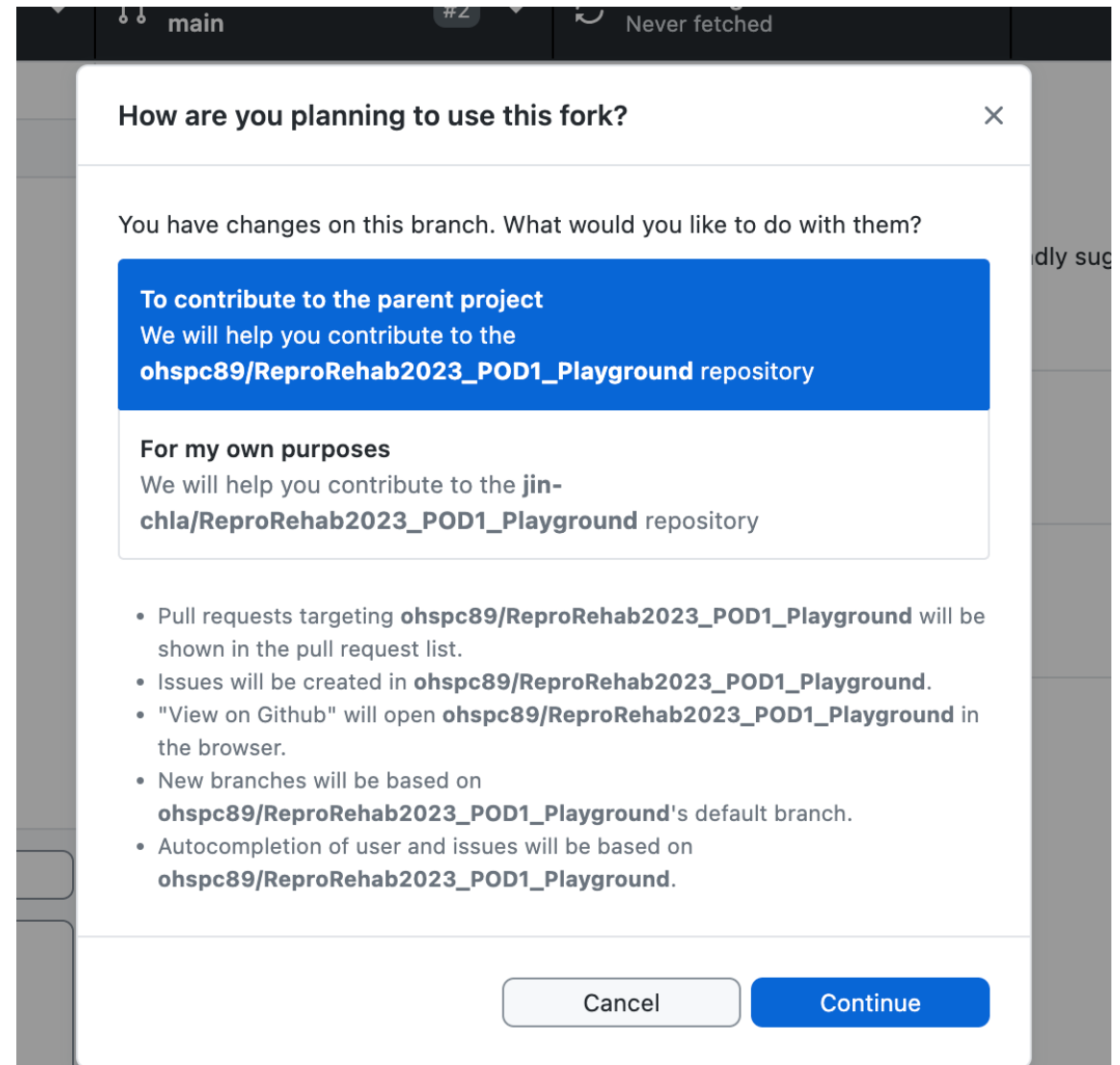
2. Again, pushing your commits to your forked remote repo is possible



1. You can only fetch/pull *origin*, but not *upstream*

Cloning a repository

- Option 1 allows you to keep the direct communication with the *upstream* repo open. This lets you work in a more *collaborative* fashion.
- Option 2 lets you work on your repo independently and try different features without messing with the *upstream* repo.
- For today's practice, let's go with the second option.



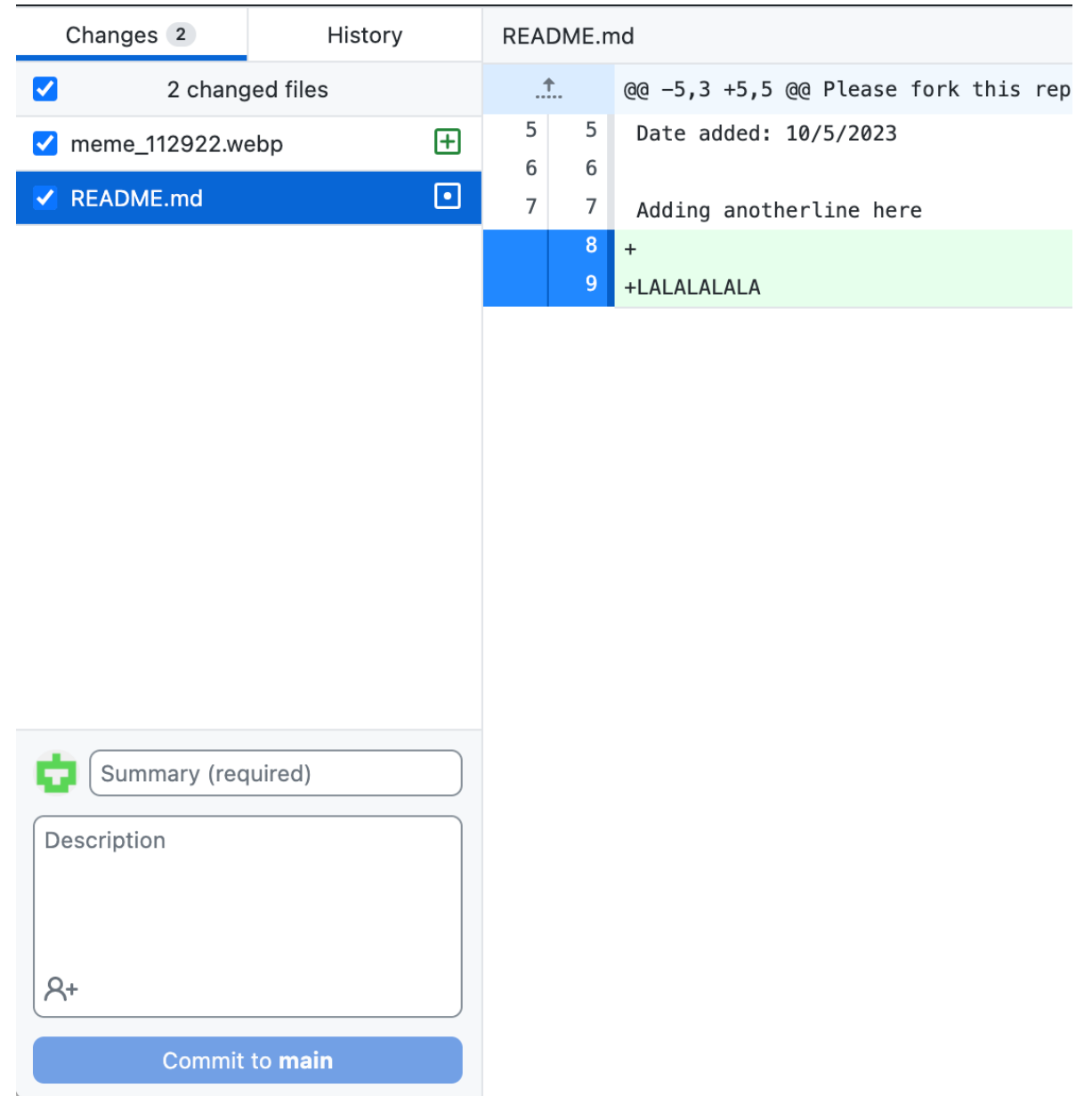
Working from your local machine

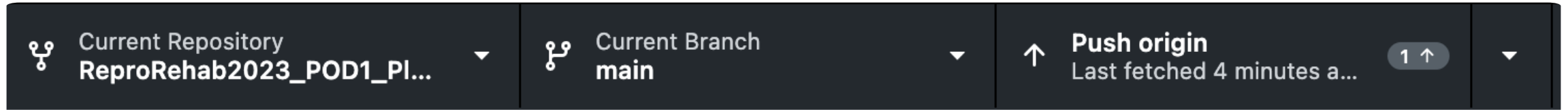
- For the sake of time, let's do a simple task. Choose Repository > Show in File Explorer (or what is the exact command on windows?)
- Open **README.md** using a text editor and make any changes.
- Copy a *meme* (in .jpg, .png, .webp...) to the opened folder.



(Add) Commit and Push

- If you use Github Desktop, any changes you make to the folder is immediately added (*staged*) for commits.
- Put a short summary about the change you made. Try making it meaningful and memorable. Writing description is optional.
- Then press **Commit to main**





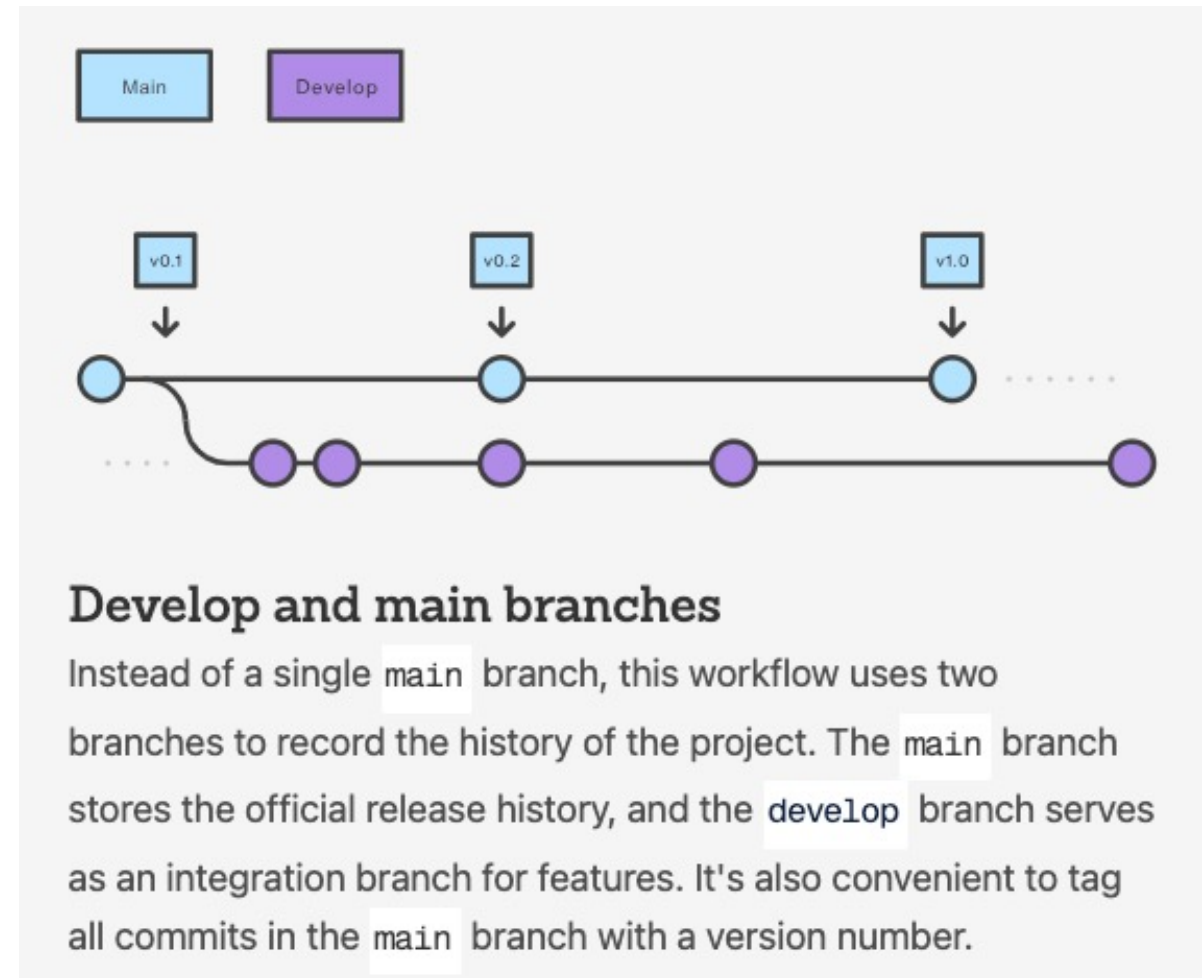
(Add) Commit and Push

- Then you can see the option **Push origin**. If you click that then these edits will be made to your FORKED repo.
- Please go to your online repo and check if all changes are made properly.



Multiple branches – why?

- The legacy workflow of software developers
- You don't want to release any unstable (or in-progress) version of your package.
- Therefore, you always store the clean and working version in the main branch and work *under the hood*.

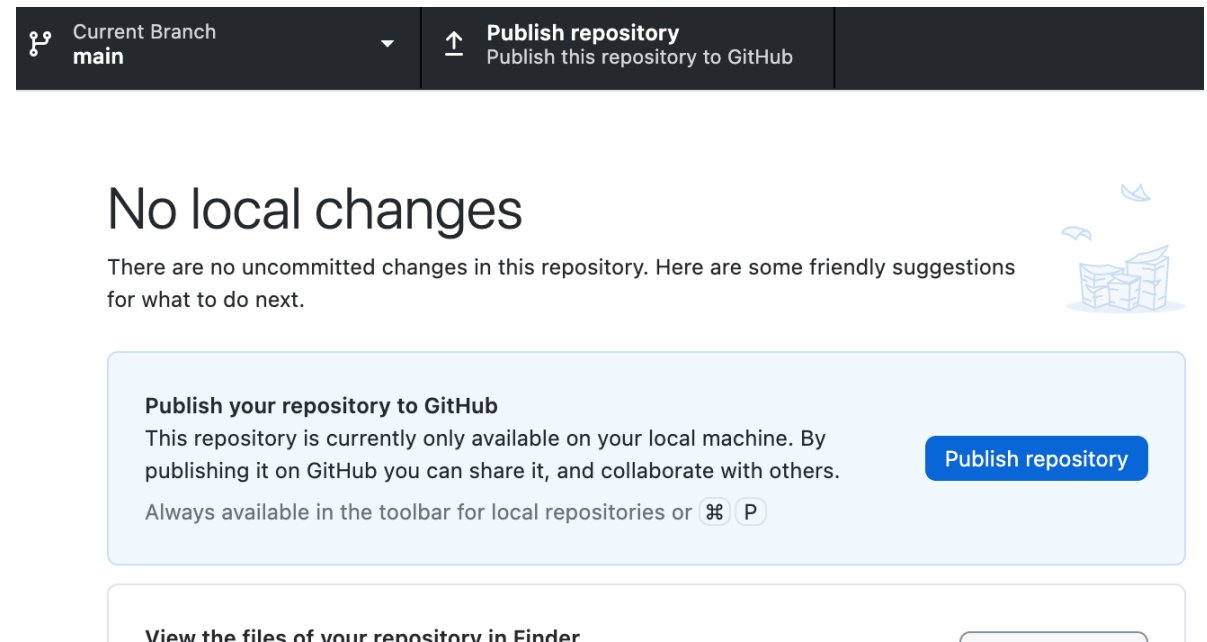


Make a new repository

- File > New Repository

Publish your repository to GitHub

- Yay! You just created your own repository

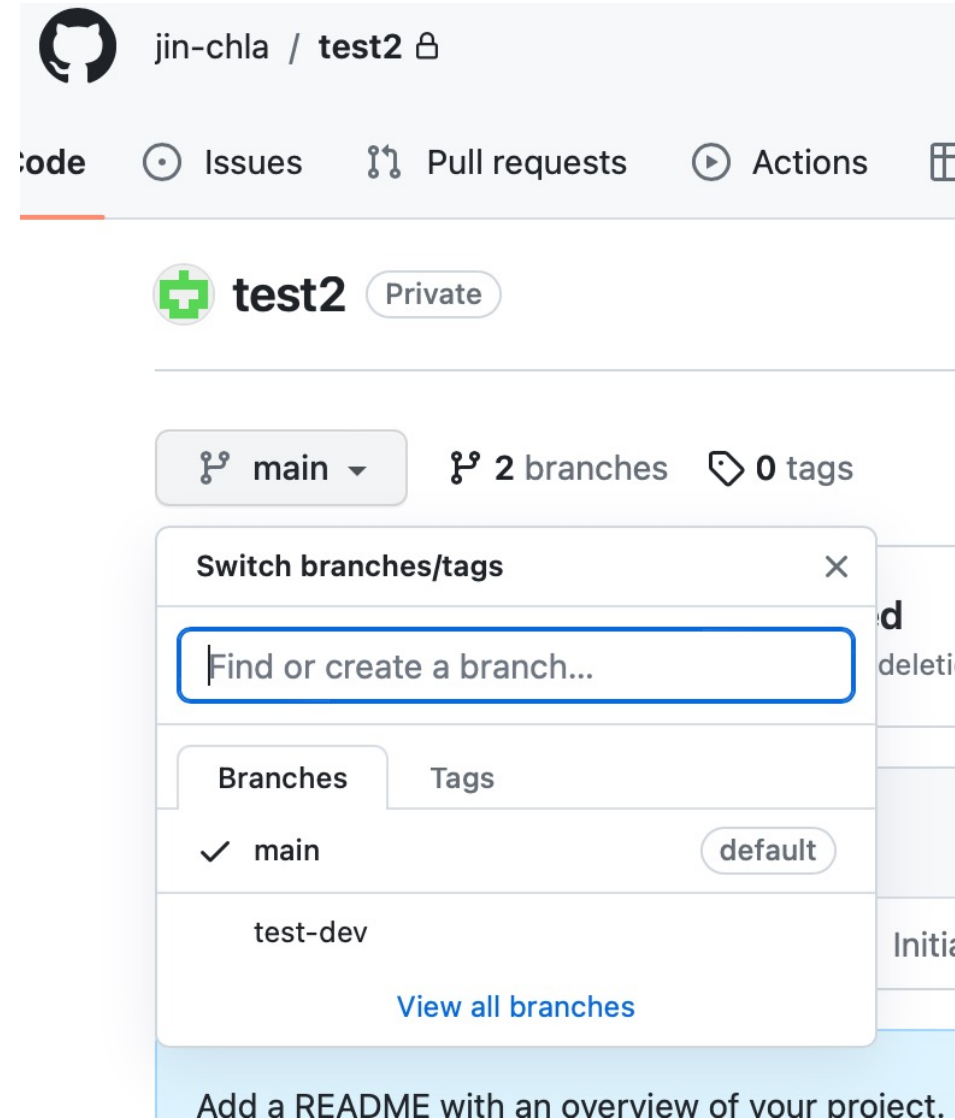


Make a new branch

- Branch > New Branch
- Name your new branch (e.g. test-dev)

Publish your branch to GitHub

- If you go to the online repository, you can switch between branches.

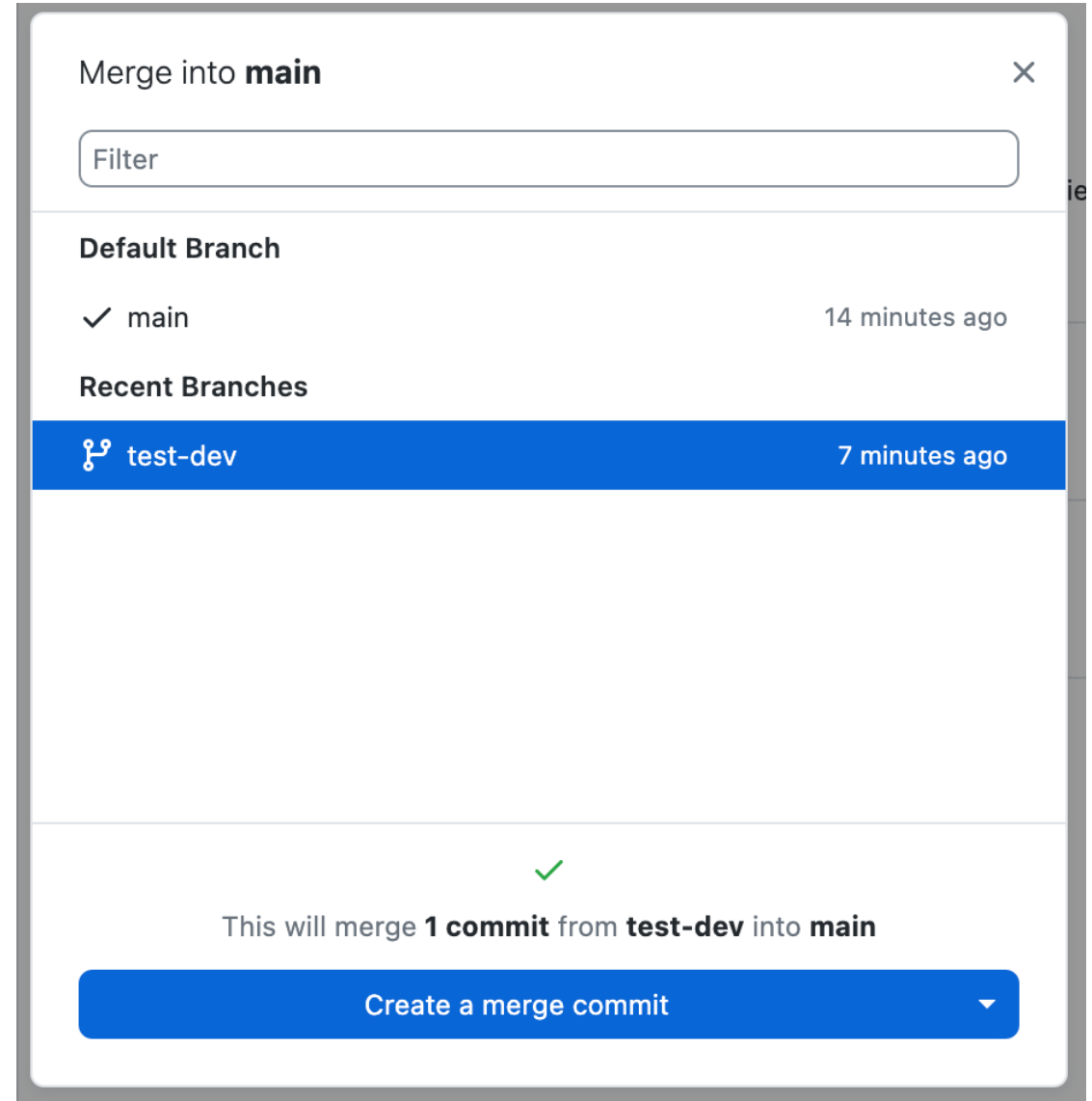


Merge branches

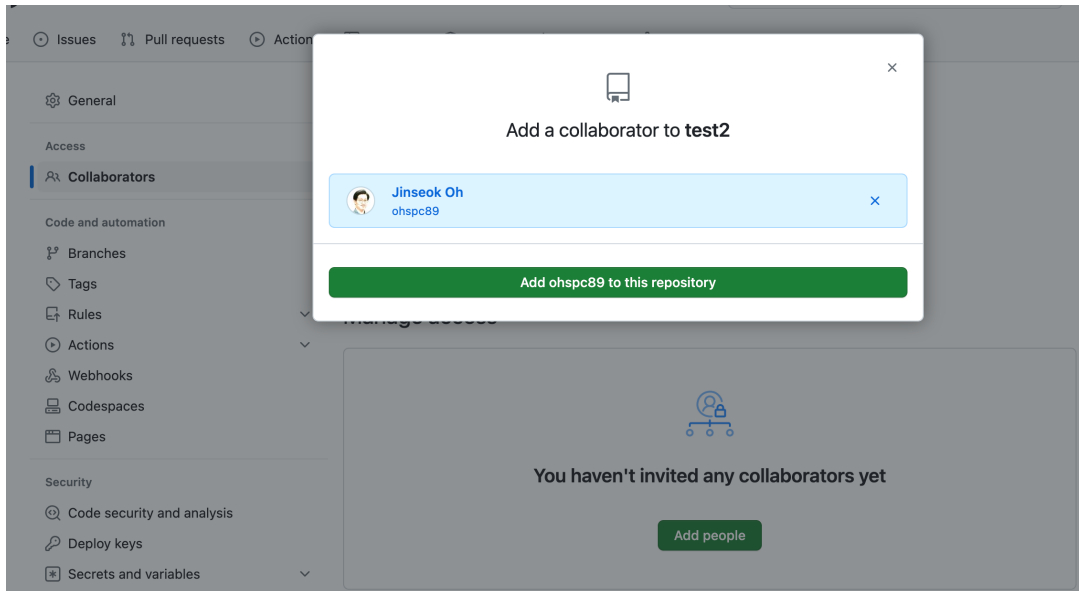
- Set your current branch as the new branch other than *main*.
- Place any file (ex. a txt file) in the folder. You can open the folder by Repository > Show in File Explorer
- Commit and Push.
- Then switch the current branch to *main*.

Merge branches

- Branch > Merge into current branch
- Select the new branch and create a merge commit.
- Push origin
- Go to your online repository and check branches.



Github: two more things!



- Public vs. Private repository (read more [here](#))
- Inviting collaborators (Settings > Collaborators > Add people)
- Collaborators can directly push to remote repository

Datalad

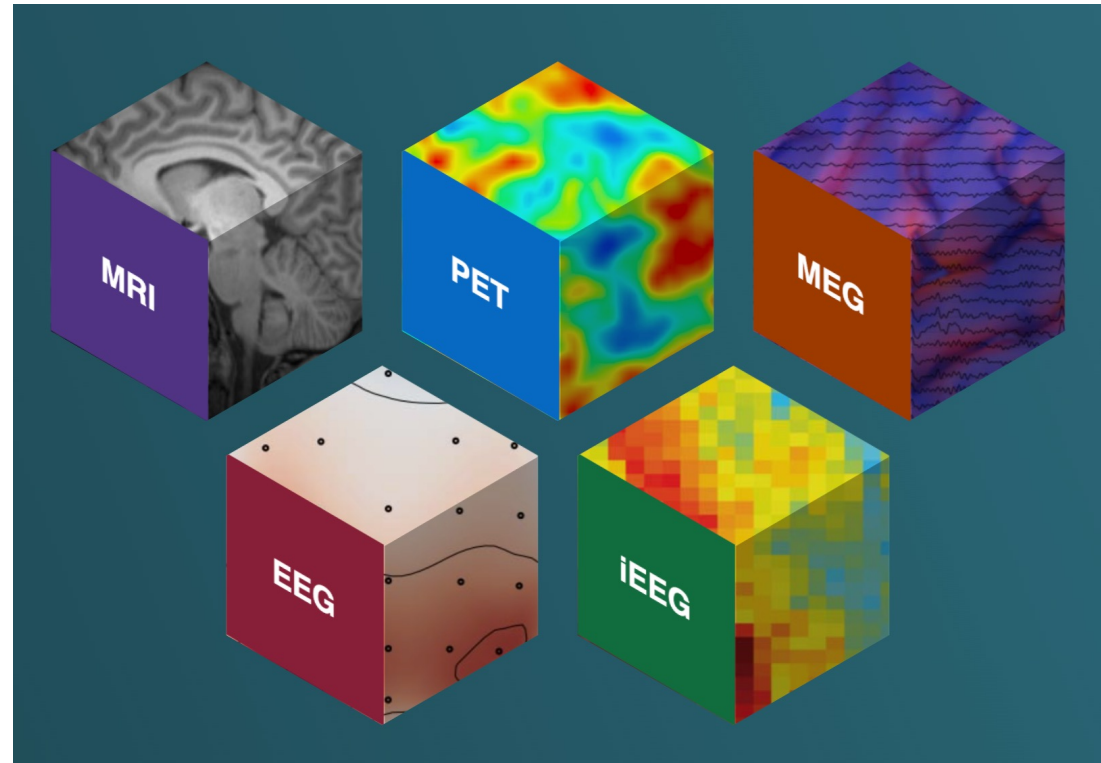
- “DataLad can *clone* a dataset to another location in a different computer.”
- Build on top of **Git** + α
- Allows version-controlling **data** and software alongside to code

YouTube: What is DataLad?



OpenNeuro

- A platform for sharing *BIDS-compliant* neuroimaging data
(BIDS: Brain Imaging Data Structure)
- Use datalad to download datasets from OpenNeuro



 Database Open Access

Motion and heart rate from a wrist-worn wearable and labeled sleep from polysomnography

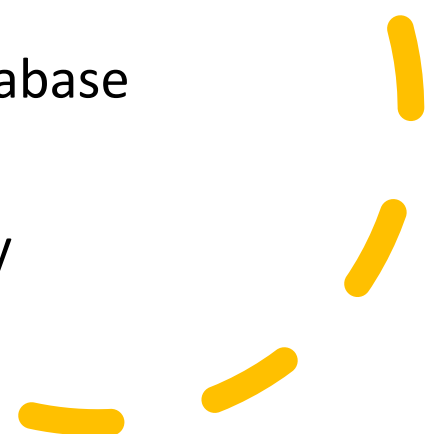
Olivia Walch 

Published: Oct. 8, 2019. Version: 1.0.0

Repositories of motion* data

*: Wearable sensor, EMG, Motion Capture...

- PhysioNet (physionet.org)
- CMU Graphics Lab Motion Capture Database (<http://mocap.cs.cmu.edu>)
- UC Irvine Machine Learning Repository (<https://archive.ics.uci.edu>)



If you want to share your code / data

- MATLAB File Exchange ([link](#))
- Your polished Github repository ([read more](#))
- Try Code Ocean ([read more](#)) to share your environment (see example [here](#))
- Check this NIH page to find repositories for sharing scientific data ([link](#))

Summary

- So... why bother learning Git/Github again?
 - Version control: keep track of **who** did **what** on **which** file **when**
 - Even if you never make use of this feature, you still can **clone** different repositories that have processing pipeline codes you may find useful
 - You may at one day wish to suggest a feature to an existing code or even prepare one by yourself and share it with others
 - Other version control software (ex. datalad) are based on Git – useful to know the basics

