MAYO CLINIC

E5883

CONTRIBUTING TO OPEN-SOURCE SOFTWARE

HOW TO BE PART OF A LARGE SOFTWARE PROJECT

Eric A. Borisch, MS
MR Laboratory, Mayo Clinic

ISMRM & SMRT Annual Meeting & Exhibition
May 19, 2021

What you will find in the full presentation…

TOPICS COVERED IN PRESENTATION

1. Why you should contribute
2. Getting started – working with a project
3. Ways to contribute
4. Collaborating with *git*: key concepts
5. Contributing to a GitHub project
6. “Live” demonstration
7. Final thoughts, references & further reading

How to add your own information to the Small World project

CONTRIBUTING

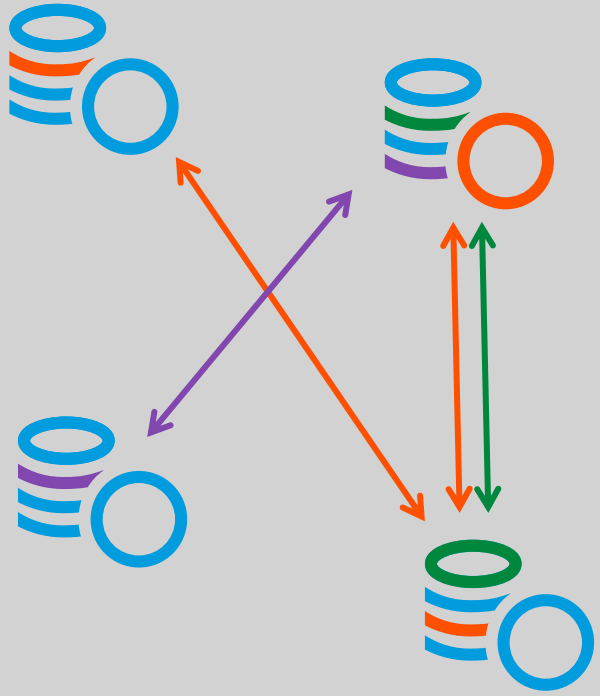
- Any level of user can meaningfully contribute to an open-source project.

Anyone can contribute!

- End users (not necessarily programmers)
  - Documentation improvements or additions [Pull requests]
  - Report issues, or provide feature requests [Issues or Discussions]
- Developers (users of a software library, for example)
  - Provide bug fixes [Pull requests]
  - Implement new features [Pull requests]
- Team members (Originators or collaborators on the project)
  - Review and address bug reports / feature requests [Issues & Pull requests]
  - Discuss and plan future developments [Discussions / Wiki]

GIT: *DISTRIBUTED* VERSION CONTROL

- Built from the ground-up for distributed version control
- *git* helps you manage and distribute your changes, but doesn't impose many restrictions
- Some best practices when collaborating with others:
  - Perform development on a local branch rather than *main (master)*
  - Rebase your development branch against *upstream* frequently
  - Stage commits to upstream in a *feature branch* (the first step to a *pull request*)



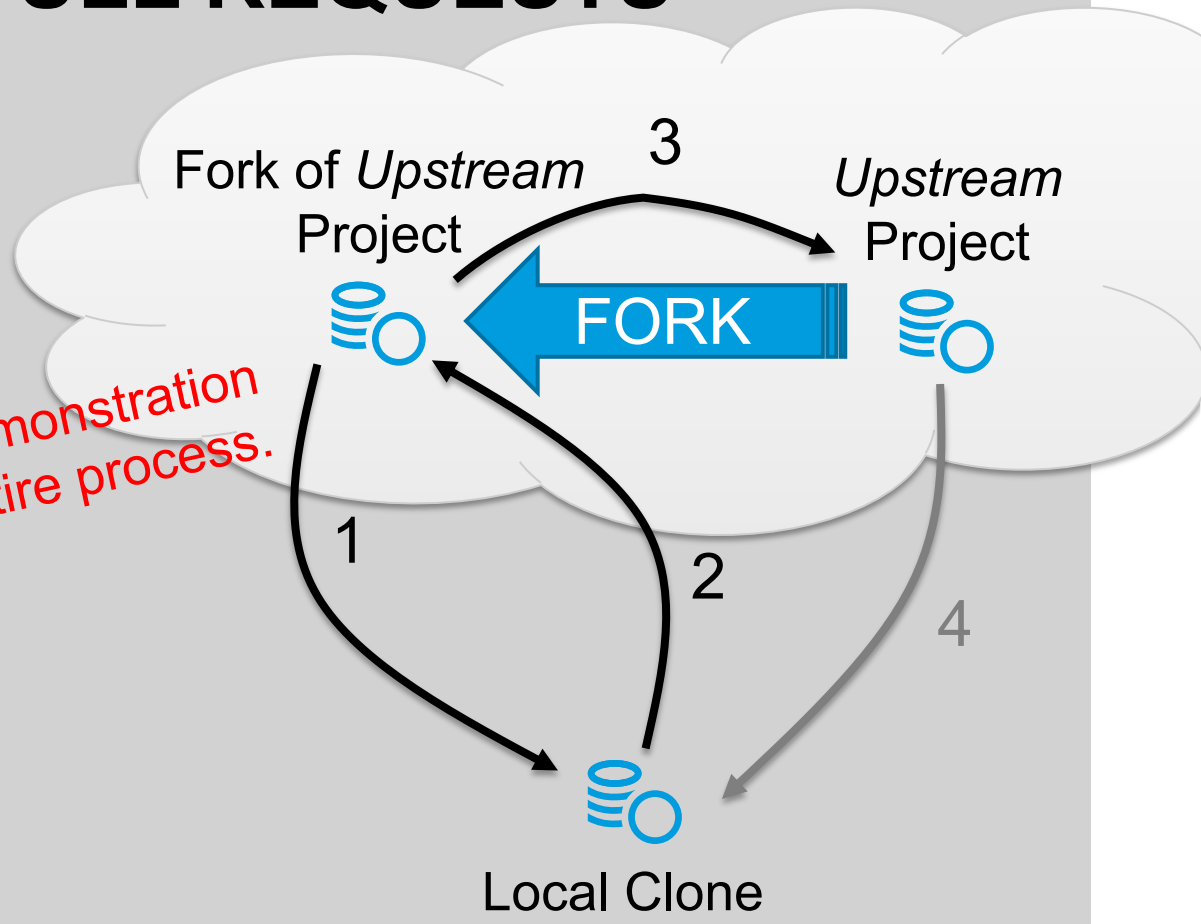
GITHUB CONTRIBUTIONS: PULL REQUESTS

"Fork" the project to your account on GitHub, then start with the same steps ...

1. Clone (the fork) to your local system
   - Create *feature* branch; modify; commit. (repeat modify/commit as needed)
2. Push *feature* branch to our fork
3. Create a *pull request* to *merge* your *feature branch* into the *upstream* project
   - Will become part of *upstream* when they choose to *merge* it
4. Add *upstream* remote to facilitate future *pulls* of the latest changes :

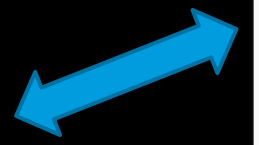
```
git remote add upstream <URL>
```

"LIVE" demonstration of this entire process.



FURTHER READING

- These slides:
  https://github.com/eborisch/ISMRM/blob/main/2021/Open-source.pdf
- Configuring SSH keys for working with GitHub
  https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh
- Online git reference documentation:
  https://git-scm.com/docs
- A nice overview on different ways to contribute to OSS:
  https://opensource.guide/how-to-contribute/
- The git “choose your own adventure” or “oops I did X, how do I fix it?”:
  http://sethrobertson.github.io/GitFixUm/fixup.html
- One of many “cheat-sheets” out there:
  https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
- Markdown format:
  https://guides.github.com/features/mastering-markdown/
- GUI interfaces to git:
  https://git-scm.com/downloads/guis



Questions?

@eborisch
borisch.eric@mayo.edu

And without these flashy animations.

OTHER USES FOR GIT: IT'S NOT JUST FOR SHARED CODE!

- Using git ≠ using GitHub; many benefits available entirely locally!!
- Anything text-based where you want to keep track of changes. Text files, LaTeX, Matlab scripts, etc.
- With some extra effort, you can `git diff` Word documents:
  https://front-matter.io/mfenner/using-microsoft-word-with-git
- Configuration files (or "dotfiles")
- Trivial to do; in your folder with files to track:

```
git init .
git add <filenames>
git commit -m "initial commit"
```


QUESTIONS AND INTERACTIVE DEMO TIME

DISCUSSION KICK-STARTERS?

Some git internals (to inform your mental model of what's happening)…

Working Directory	[`git add`] →	*Index*
Index	[`git commit`] →	*Repository*
Repository	[`git checkout`] →	*Working Directory*

- Working Directory: The documents present and editable in your directory
- Index: Staging area for changes to be added to the repository
- Repository: Actual storage of file contents over time

You can `git commit <path>` directly from Working to Repository

PARTING SHOTS

- Commits and branches are cheap; don't be afraid to use them.
- If are concerned about what a command is going to do, copy your work to a new branch, and try it there
- If you don't want to or aren't allowed to use GitHub, you can still use git!
  - Having a GitHub/GitLab server to push to isn't a *requirement*, although it does make sharing easier (thanks to the GUI for Pull Requests)
  - A shared filesystem within a lab can host a repository: `git clone /path/to/repository`

