Git Q&A

Q: Is there a point in using Git when working on code alone ?

A: there’s a few advantages, like

- work ***back-up*** online
- retaining work ***history*** (aka go back whenever needed) offline

- ***transparency*** (often reviewers ask for code and I have read my fair share of reviews that criticize poorly documented code) online
- ***visibility*** (especially for younger scientists applying for jobs) online
-
- preparation for the time you’ll actually need to work on a collaborative project offline

- & you have the honor to help train AI without your consent :P online

Q: Recommendations for some Git usage habits from regular Git users & Useful tips that make the usage of Git convenient

A: Try to use

- ***branches*** after developing the core functionality of your repo

  (experiment without breaking stuff)
- short, but ***informative commit messages***

  (which can be helpful when looking back in a file's history)
- one, or multiple informative ***README***(s)
- wiki

  (on website – simple markdown with internal links to files)

  (if you think your project is too convoluted and needs to be documented)

Q: How to organize your workspace in Git to be convenient later for code sharing?

A: Depends on the complexity of your repository.

- try to maintain at least one high-level README
- use ***tags*** when you reach important milestones

  (e.g. exact scripts version used for a publication)
- use ***issues*** and/or ***milestones*** (on website) in order to keep track of your progress and plan

  (you can make this even more specific with the use of ***labels***)

- use directories and/or git *submodules* if your repo contains many files

Q: How to "link" your results (text info, tables, figures to submitter script versions and organize it in a structured way?

A: Depends on how automated you want the process to be.

- easy: use commit messages with a version scheme of your own and look for it in the history
- moderate: use tags when having something that you think is important; switch between tags to browse different versions of your results
- advanced: every commit has a unique identifier; you could write a script that first computes the corresponding *hash* and appends it to the names of some files, before you commit.

Q: How to check and combine upstream and downstream changes?

A: Use

- ***git rebase*** <upstream> <downstream>

  (also see https://medium.com/@topspinj/how-to-git-rebase-into-a-forked-repo-c9f05e821c8a)
- if there are any conflicts (i.e. code parts changed in both versions) you will need to resolve them manually before finishing the merging

Q: What are the Git functionalities that are not readily available using Github and that may be of interest for researchers?

A: No answer on research-specific functionalities, sorry XD; but

- more generally, all services allow you to perform basic, every-day operations, but not more complicated actions
- therefore, when something goes wrong and you will need to search for info, read & understand command line arguments
- it’s gonna be easier if you understand how git works and where to find documentation