Getting started on Git/GitHub

Ricardo B. Lourenço — McMaster Remote Sensing Lab

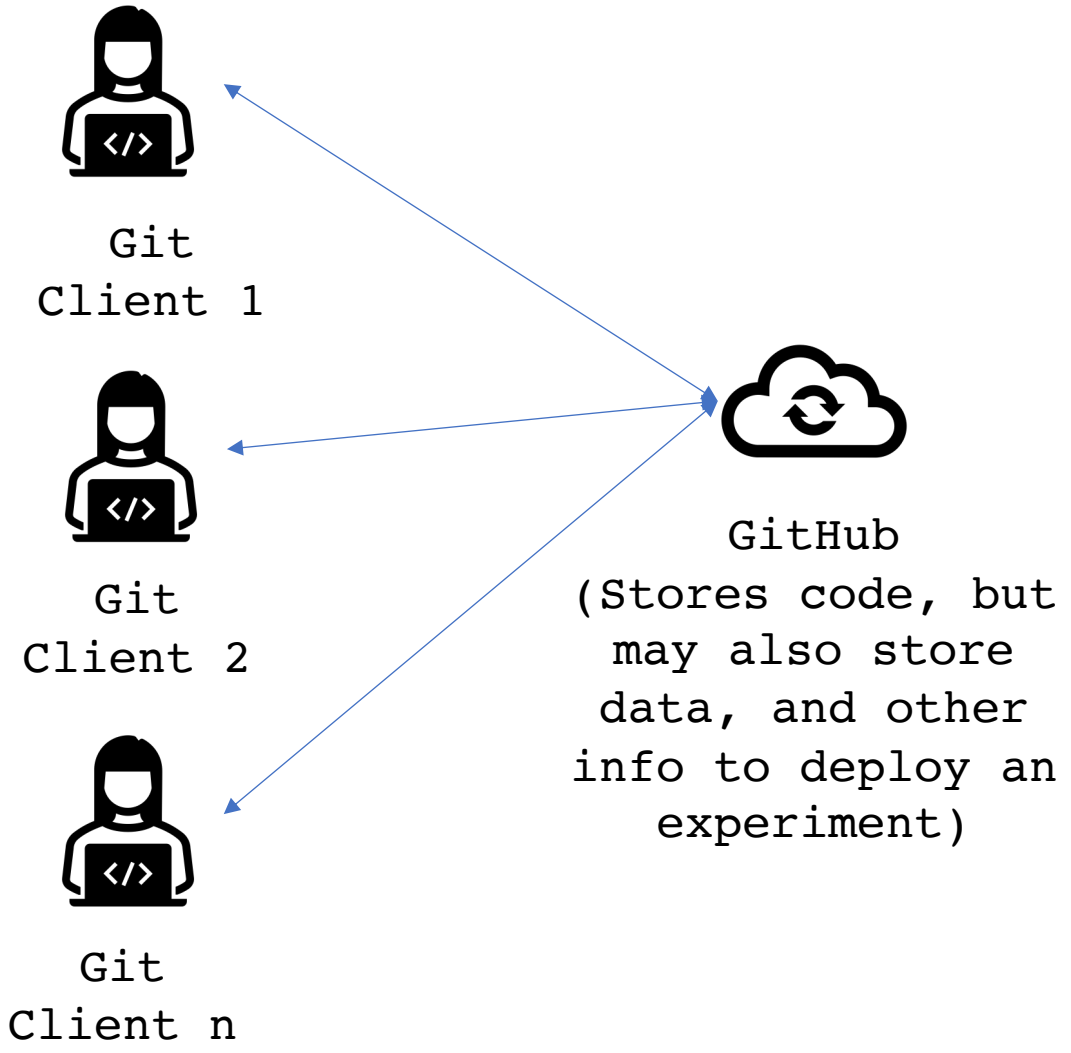
Feb 18th, 2022

Scope

- This presentation will briefly cover the Git commands used to interface with a GitHub account;
  - A proper course may take some weeks for an in-depth coverage;
- We are not covering much the culture of reproducible science, but the git relation to such a culture;
- The setup of the GitHub account, and other more complex arrangements (CI/CD ; GitHub Pages; Compute Canada integrations; etc.) will be covered at the **CCPrimer** webpage:
  - https://ricardobarroslourenco.github.io/CCprimer/
- These slides were based on GitHub´s cheat-sheet, which can be handy when you are started:
  - https://training.github.com/downloads/github-git-cheat-sheet/

Outline of the system

- Multiple users contributing simultaneously;
- Centralized cloud;
- Each project = repository;
- Managing conflit is difficult;
- Created for Computer Science, not for other scientific domains, but Scientists are adapting as needed;
- Each research community has its own standards;
- Usually the norm is sharing code (as part of Open Science efforts, but nowadays entire experiments are shared (ex.: Google Earth Engine JS code; GIScience; ML community; Pangeo; etc.)



Create a GitHub account

- To create an account is simple:
  - Go to GitHub pricing page, select the Free tier, and click *Join for Free*;
  - Just follow the prompts to create your personal account.
  - You should receive an e-mail from them on the registered e-mail account, to verify your identity. If you fail to do so, your account would be basically useless.
- Note: Since I have created my account some years ago, I am just using GitHub's user documentation as reference. If things get complicated in this step, please let me know.

Install a Git client on your machine (or where you want to connect with GitHub)

- On Windows/MacOS you may install a GitHub Desktop (which has a GUI to control)
  - https://desktop.github.com/
- On other machines: a git client!
  - https://git-scm.com/
- There are several other options! Refer to CCPrimer for details (ex.: Using it with PyCharm; Rstudio; VSCode; MacOS native git, etc.)

Setting up your client

- Too brief description, more details at CCPrimer;
- Skip this if you use GitHub Desktop or IDE´s;
- If using git, open your terminal (command line or Bash, for ex.) and set:
- Global user name:
  - git config --global user.name "[name]”
- E-mail address attached to the commits:
  - git config --global user.email "[email address]

Creating repositories (or, starting a project)

- You may create a repo using the GitHub webpage (it is easier);
- To create via a Git client, you need to be on the terminal, at the root folder of the codebase you want to create a repo over and:
  - `git init`
- This just do the creation locally, now you need to upload your creation:
  - `git remote add origin [url]`
- If you are actually starting to contribute on a repo already available on GitHub (or if you created the repo on the GitHub website first), you just need a local sync copy:
  - `git clone [url]`

Branches

- On a GitHub repository you have branches, which are a subdivision of versioning;
- When you create a new repo, it will only comes with a master branch;
  - Often is created a development branch which is used for code that is not production ready;
  - *You may create as many branches as needed*, each with a custom name;
- In scientific usage, this may not make much sense (*remember this was meant for software engineering*), but one may envision that branches can be used to research steps of a project (and even GitHub provides planning tools such as a Kanban, in which you can control the stage of development)
- Details on branches will be provided at CCPrimer;

Making changes

- To add a new file to a pre-existing repository (locally):
  - The file must be already saved in the root, ot a subfolder of the root, locally;
  - git add filename.extension
- If you change (editing) a pre-existent local file, you don´t need to add, because it is already on the tree structure, and committing will be enough to synchronize with a GitHub repo;
- If needed to remove a file from the project;
  - With also a deletion of the file on disk:
    - git rm filename.extension
  - Without deleting the file:
    - git rm --cached filename.extension
- After all local structure changes, you need to persist them locally:
  - git commit -m ”message describing what you have done so far”
- When needing to synchronize changes with the GitHub repo:
  - git push origin branch_name

Synchronize Changes

- These commands are meant to synchronize your local to the global repository (a.k.as GitHub);
- Updating your local branch with all commits ahead of you (if available on the GitHub repo) and conciliating them with your changes:
  - git pull
    - Pulling is actually a combination of fetch and merge commands, and it is highly recommended, therefore we won´t cover the other options (look into CC Primer if needed)
- As saw previoulsly, uploading your local changes to the main GitHub repo:
  - git push

When things get complicated

- When working alone is easy ☺ Establishing a collaboration culture is harder.
- Common cases
  - When multiple people are submitting code at a same file, at a same portion of that file;
  - When different people makes too many changed without syncing to the main repo, and other people are doing the same thing;
- Solutions
  - Combine with people where will you be editing;
  - Try to commit and push changes often, but meaningful ones;
  - Try to review those commits promptly;
- Technical solutions:
  - git merge : combines branches into a unified solution, often needing editing
  - git rebase (not recommended): do the same as merge, but does not preserve the provenance of the changes;
  - git reset (extreme solution): wipe out changes related to a certain commit point

References

- CCPrimer: Reference guide I am writing for the lab with all things computational related to Compute Canada (on the go, and feedback is welcome ☺)
  - https://ricardobarroslourenco.github.io/CCprimer/
- Prof. Paez course on Reproducible Research (highly recommended)
  - https://github.com/paezha/Reproducible-Research-Workflow
- Atlassian material on Git:
  - https://www.atlassian.com/git/tutorials

Thanks/Obrigado ☺