GIT AND GITHUB IN DATA SCIENCE

BAN400

Håkon Otneim

Brief review of the video lectures

- In the video lectures related to this session we give a brief introduction to the main functionality of Git and Github.

Git

- Git is an open source, light-weight, and powerful system for *version control*.
- Git keeps track of the state of all the files in a project over time and works essentially as a time machine. We can easily navigate the history of the entire project and return to an earlier point if needed.
- Git lets us work on several *branches*. This lets us develop experimental features without touching a working/stable project, which can be merged into the main branch when tested and ready.
- Git comes with tools for handling merge conflicts.

Github

- Online (commercial, owned by Microsoft) service hosting git repositories.
- Comes with additional features such as *project management and team collaboration*, *releases*, *forks* and *pull requests*.
- Students get free access to GitHub Pro, which includes the possibility of keeping private repositories: https://education.github.com/pack.
- There are alternatives, most notably **GitLab** which is in orbit around (but not owned by) Google.
- … and many more: Wikipedia: comparison of source-code-hosting facilities

Interface

- We use the terminal to work with git and SSH authentication. This is pretty standard, and stable over time: You can expect that this workflow will work for a long time.
- The terminal provides the same interface for all platforms.
- There are GUIs, paid (i.e. GitKraken) and free (i.e. the GitHub Desktop).
- There are other options as well

Basic terminal commands

`ls`	List files
`cd`	Change directory
`pwd`	Print working directory
`..`	One step up

Basic git commands

Most of these commands take additional arguments:

`git init`	Initialize git repository in this folder
`git add .`	Move all files in working directory to the staging area
`git commit -m "Message"`	Commit to history, with commit message
`git log`	Log of recent commits
`git pull`	Update offline rep with the latest version from the online repo
`git clone`	Download repo from online source
`git push`	Push latest commit to online repo
`git branch`	Make a new branch
`git merge`	Merge branch into the current branch
`git reset`	Go back in time and restore an earlier version of the project

Workflow in data science projects

- Does Git/Github fit into your workflow?
- What kind of projects/tasks fit into at git-based workflow?
- What kind of projects/tasks **do not** fit into at git-based workflow (at least as far as we understand it now)?
- Some Git best practices::
  - "Commit often, perfect later, publish once."
  - "Don't change published history."
  - "Release tagging"

Moving on: forks and pull requests

- One thing is to manage a small coding project, either on our own or in a small group, using Git and Github.
- Git and Github (and similar services) also provide tools to really unleash the possibility of open source development.
- We will look specifically at
  - **Forking**: Create a linked copy of a Github repository on your own account. You can do whatever you want with this, but also pull the latest version and merge that with your own work.
  - **Pull request**: Request to merge your changes back into the main repository.