git and github

Usman Ayub Sheikh
PhD Researcher @ BCBL

version control

- git and github
- to keep track of different versions of our files,
- that we can revert back to if we want, or
- see the details of.

to keep track of any changes

- such that we can go back anytime we want
- what were those and when were they made
- helps a lot when something goes wrong, and
- you want to debug, find out what went wrong, and
- at which stage.

test changes without losing the original

- tried refactoring a working piece of code, BAM!
- one thing we can do is to make a copy before
- hard, actually, impossible to manage
- With git,
  - we can make the change,
  - test the change, and
  - merge it back only when we are sure it works

revert back changes to the older version

- made a change
- later on, found it to be superfluous, or
- unnecessary,
- you can always revert back to older version, and
- start working from there.

synchronization of code

- collaboration over github, or
- any other code hosting service for that matter
- git allows synchronization of code, and ensures that
- your code potentially matches with the code being written by your collaborators

what’s in it for non-programmers?

- can be used to keep track of all sorts of files
- you can really from all of its uses if you use it for plain text files: text (.txt), latex (.tex), markdown (.md) files
- you cannot use it for word documents, powerpoint slides; one of the reasons it’s so hard to convince writers to use git

setting things up for our first project

- git is an offline program
- the installation procedure is very simple
- github account
    - www.github.com
    - username, email ID and password

setting things up for our first project

- go to your home directory, and create a folder named, for example, my-project
- inside this directory,
  - create a file named paper.txt, and
  - a folder named, for example, source
  - source/analysis.py
- in the terminal, change the directory to ~/my-project

first three git commands

git init

- to initialize a project/repository for version control

first three git commands

```
git init
```

- to initialize a project/repository for version control

```
git add <file-1> <file-2>
```

- To tell git which files to include in the next revision

first three git commands

```
git init
```

- to initialize a project/repository for version control

```
git add <file-1> <file-2>
```

- to initialize a project/repository for version control

```
git commit -m “setting up project”
```

- to save the change as revision; hash, time, message

CREATED REPO AND MADE AN INITIAL COMMIT
SO I GUESS YOU COULD SAY THINGS ARE GETTING PRETTY SERIOUS

git log for history

- to see the history of revisions

```
git log
```

- first commit, not much history to look for

and git status for status

- to see the history of revisions

```
git log
```

- first commit, not much history to look for
- while we make a few more revisions,

```
git status
```

- whether the revisions we have made have already been added, committed or not?

git diff for difference

- what exactly does this revision adds to, or removes from the project
- remember, these are the changes yet to be committed

git diff for difference

- what exactly does this revision adds to, or removes from the project
- remember, these are the changes yet to be committed

git diff for difference

- what exactly does this revision adds to, or removes from the project
- remember, these are the changes yet to be committed
- how is this revision/commit different from that revision:
  - `git diff <hash-1> <hash-2>`
  - First six characters will work

where does github come in?

- fork the repository cooking-recipes
- forking allows us to:
  - have a version that we can experiment with,
  - without affecting the actual project
- when we want to propose changes to a project, or
- when we want to learn from it and use it as a starting point for our own project

git clone and git push

- `git clone <url of the fork>`
- this will clone the remote repository to a local folder of the same name
- you can make the changes you want to make

git clone and git push

- `git clone <url of the fork>`
- this will clone the remote repository to a local folder of the same name
- local vs. remote: origin or upstream
- once you are done making the changes, you would like these to be pushed to the remote
- `git push <url of the fork> master`
    - or ~~`<url of the fork>`~~ `origin`, and
    - `master` is the branch you want to push to

opening/creating a pull request

- so far, all the changes have been made to fork
- if you want to propose these changes to the actual library or toolbox or a manuscript
- you open a pull request (select branch; main page)
- the author/s review the changes, and merge them if there are no conflicts
- often leads to discussions about changes being bug-prone, features being necessary/unnecessary, style being good/awful

git pull

- `git pull <url of the upstream> master`
- `git push origin master`
- we already have an alias for url of the fork,
- we can create one for the upstream too.
  - `git remote -v`
  - `git remote add upstream <url of the upstream>`
- once we are sure all are synchronized, we can propose our changes, push them to the fork, and open a pull request for review.

git pull

- `git pull <url of the upstream> master`
- `git push origin master`
- we already have an alias for url of the fork,
- we can create one for the upstream too.
  - `git remote -v`
  - `git remote add upstream <url of the upstream>`
- once we are sure all are synchronized, we can propose our changes, push them to the fork, and open a pull request for review.

git checkout, git revert and git reset

- `git checkout <commit hash>`
- `git checkout file_name`
- `git revert <commit hash>`
- `git reset --soft <commit hash>`
- `git reset --hard <commit hash>`
- learn more about them through some useful resources like git documentation, and git immersion course

branching

- to create a new branch, you will run `git branch` with the name of the branch
- to enlist all branches, you will run `git branch`, without any argument, the highlighted one is the one active now
- to switch to a specific branch, you will run `git checkout` with the name of the branch

Thanks

Questions?