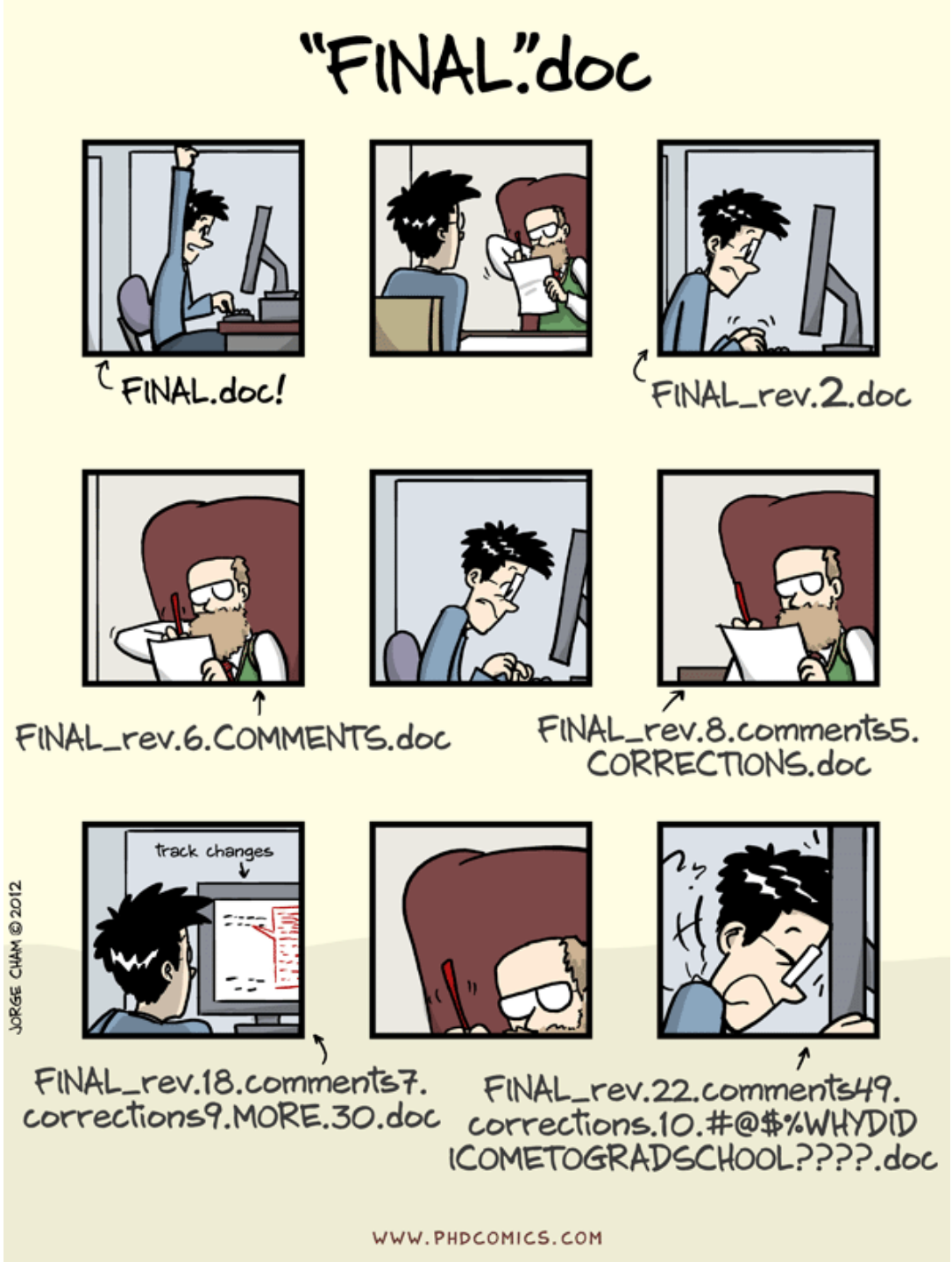
Version control and GitHub

Why use version control?



Version control and Git

- Main features of a version control system:
  1. Saves each new set of changes sequentially
  2. Keeps track of different versions of a document/project
  3. Able to merge changes from multiple versions

- **Git** is a specific version control **system**
  - Think “track changes” in Word + Dropbox, but much more general and powerful

- A whole new system to learn. Is it worth the effort?
  - Maybe not when working alone
  - But critical to avoid disaster when collaborating on complex projects
- The gold standard in the tech sector – used EVERYWHERE

GitHub

- GitHub is a specific **website** that uses Git to host projects in the cloud
- We will use GitHub at a few points in this course
  - Lecture slides
  - Assignment 2
  - Term project
- Why?
  - To start building habits of using version control
  - To get you used to the basic terminology and actions of Git and GitHub
- “Real” developers & data scientists use Git at the command line
  - I’m not going to require you to do that now
  - But I encourage you to learn it on your own

Getting set up with GitHub

1. Create a GitHub account
2. Download GitHub Desktop
3. Connect GitHub Desktop to your GitHub account

1. Create a GitHub account

If you don’t already have one:

- Go here and fill out the forms: https://github.com/

No need to apply for the GitHub Student benefits (though you can if you want to)

2. Download GitHub Desktop

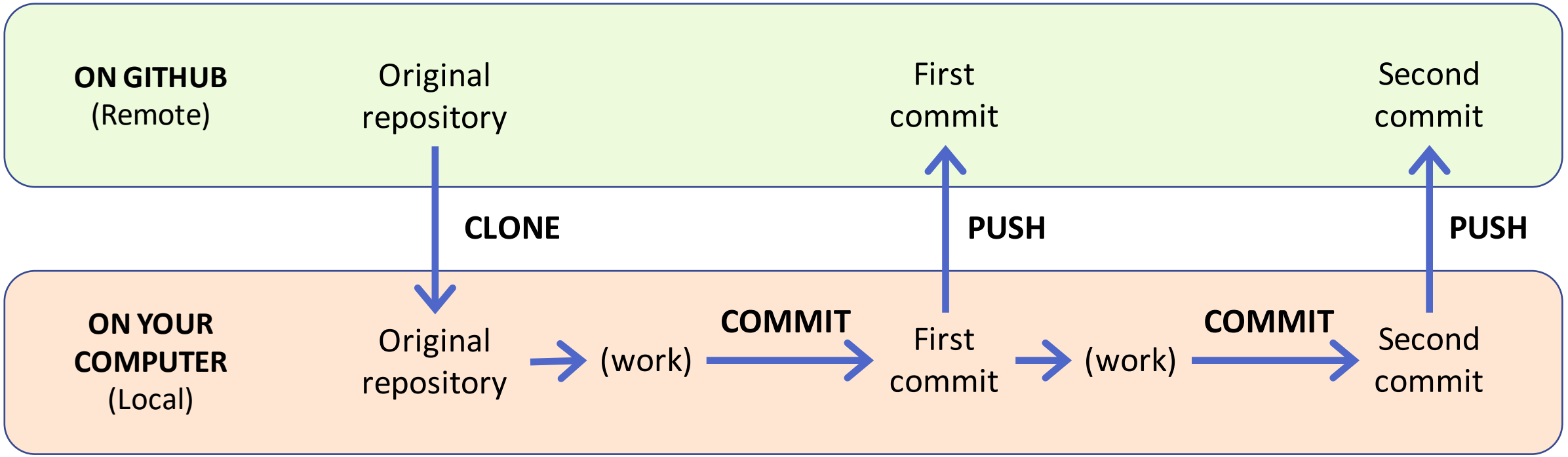
- Go here: https://desktop.github.com/

- GitHub Desktop is a standalone app for using Git and GitHub through a graphical user interface (GUI).
- Recommend but not strictly required
  - You can make changes directly through the GitHub website, but it will end up being harder in many ways
  - You can use Git at the command line (shell) if you already know it or want to learn

3. Connect GitHub Desktop to your GitHub account

- Open GitHub Desktop and go to File -> Options
- If you need help, try this: https://docs.github.com/en/desktop/installing-and-configuring-github-desktop/installing-and-authenticating-to-github-desktop/setting-up-github-desktop

Basic workflow (only 1 contributor)

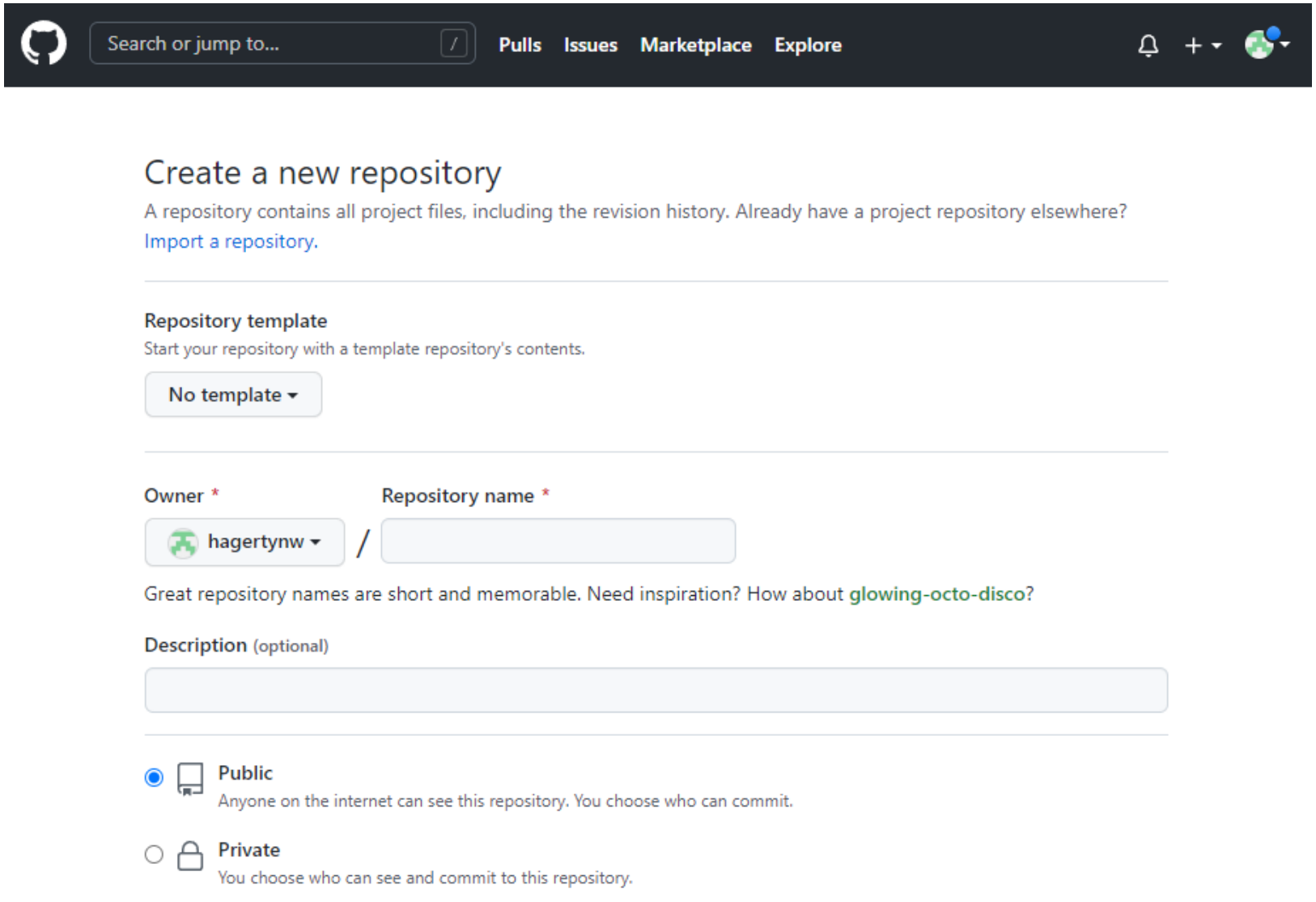


Workflow for your project

1. On GitHub.com, create a new repository
2. **Clone** this repository to your local machine
3. Do some work (edit the repository)
4. **Commit** changes (i.e., save a draft)
5. **Push** your commit to GitHub (back it up to the cloud)

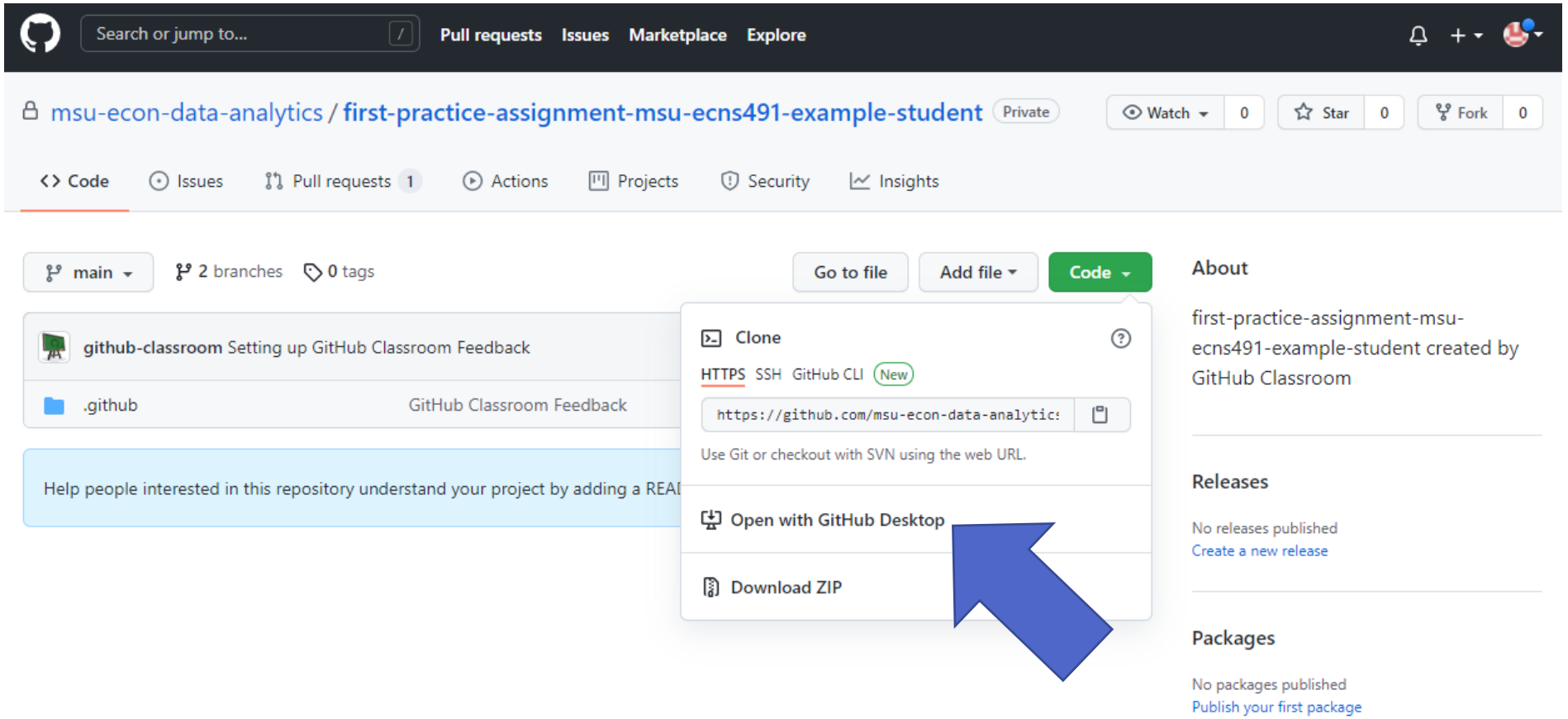
1. Create a new repository

A repository (**repo**) is the full record of a project folder and all its changes ever.



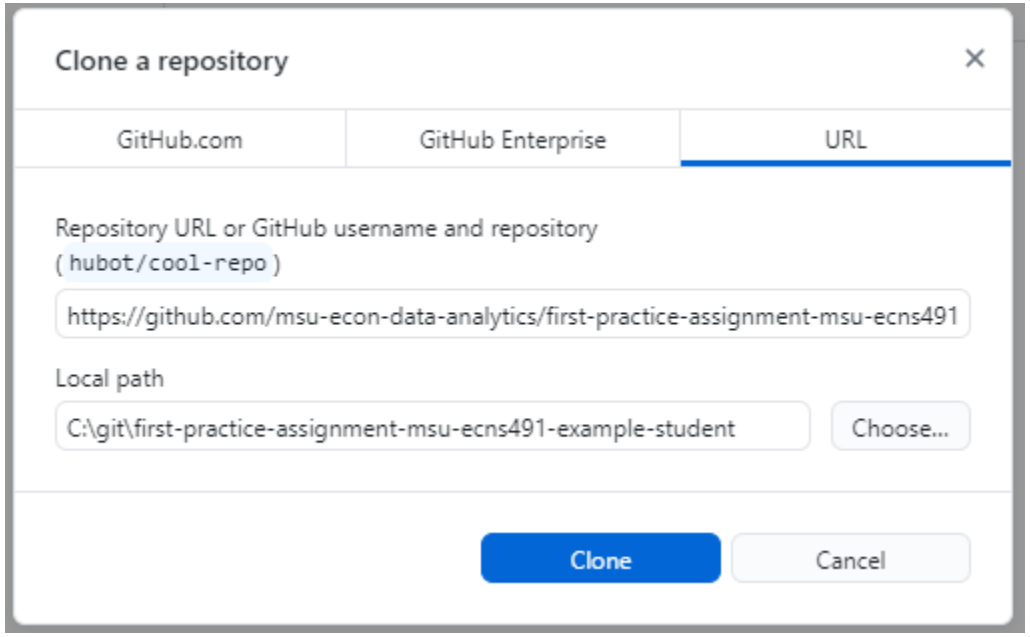
2. **Clone** the repo to your local machine

- Clone downloads a full copy of the repo from GitHub to file storage on your computer



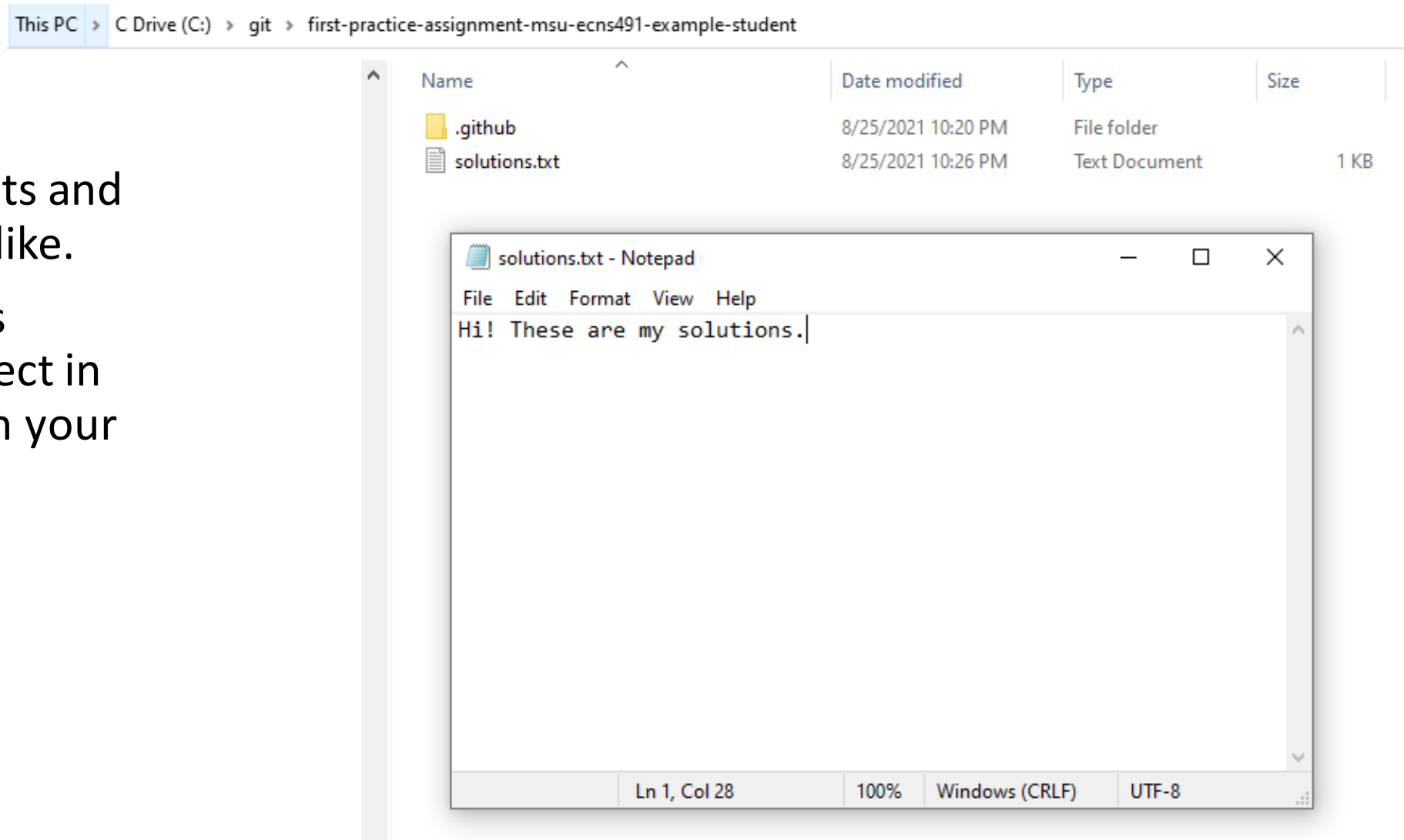
2. **Clone** the repo to your local machine

- GitHub Desktop should now come up
- Choose where you want to store the repo on your computer (the default location is probably fine)



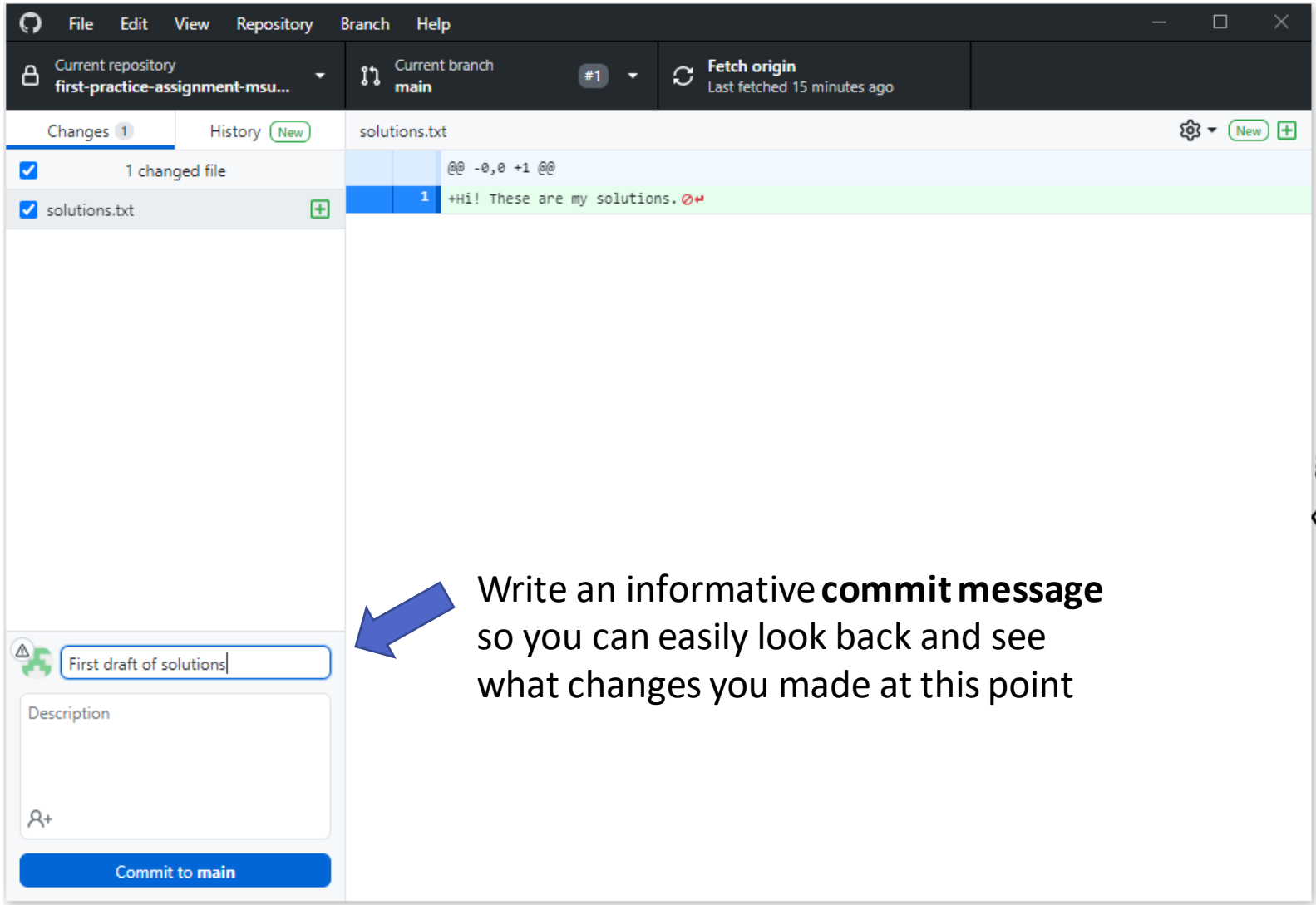
3. Work on your project (edit the repo)

- Create or edit scripts and documents as you like.
- Save all documents related to this project in the repo's folder on your computer.



4. **Commit** your changes

- Commit is like Save, but for your whole project
- It records a snapshot of your whole directory at this point
- Unlike Save (but like version history in Google Docs), you can go back to a particular commit later



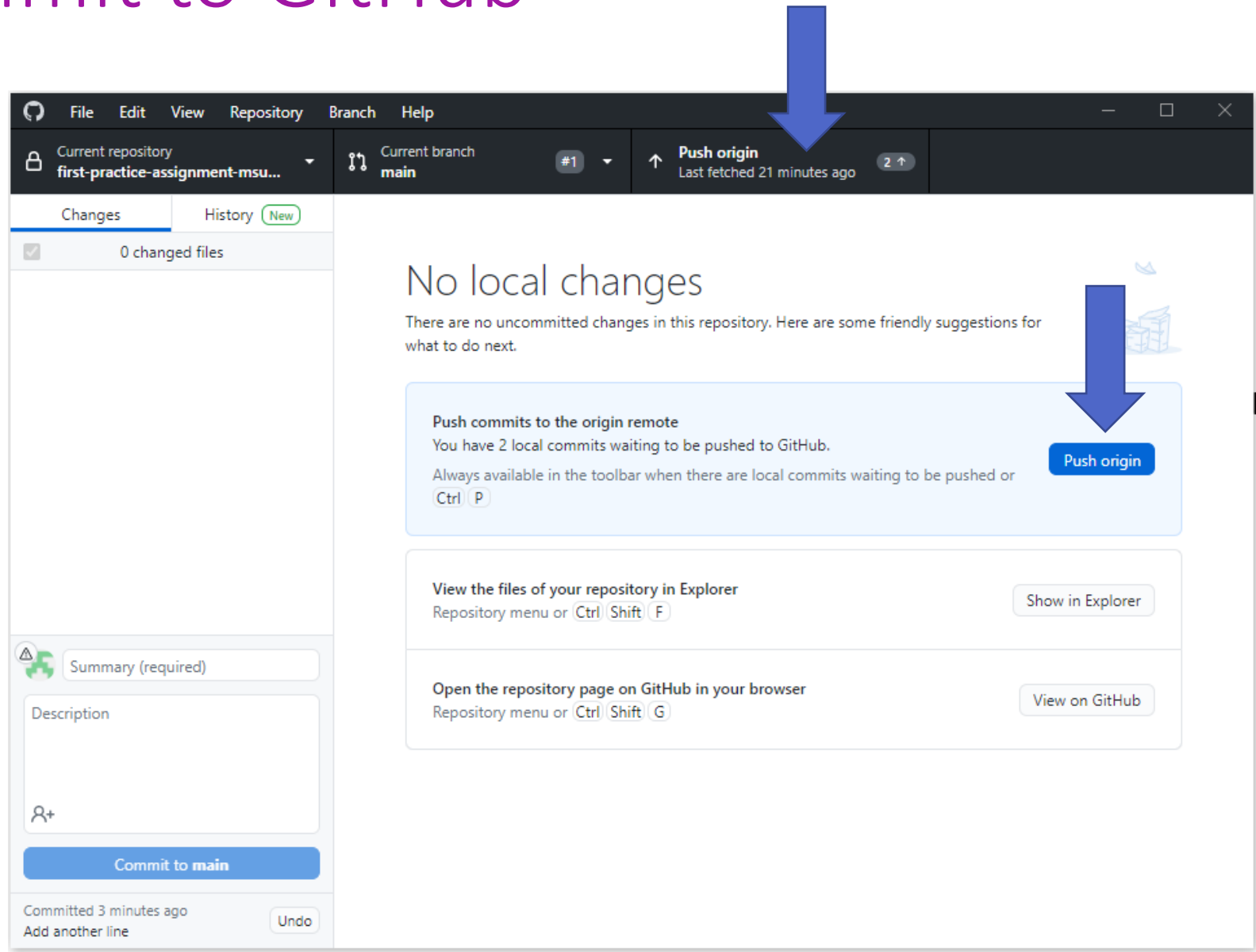
Write an informative **commit message** so you can easily look back and see what changes you made at this point

4. **Commit** your changes

- Commit early and often!
  - Every time you make a major change, or take a break from working
  - If you make a big mistake, you can use GitHub Desktop to roll back to an earlier commit

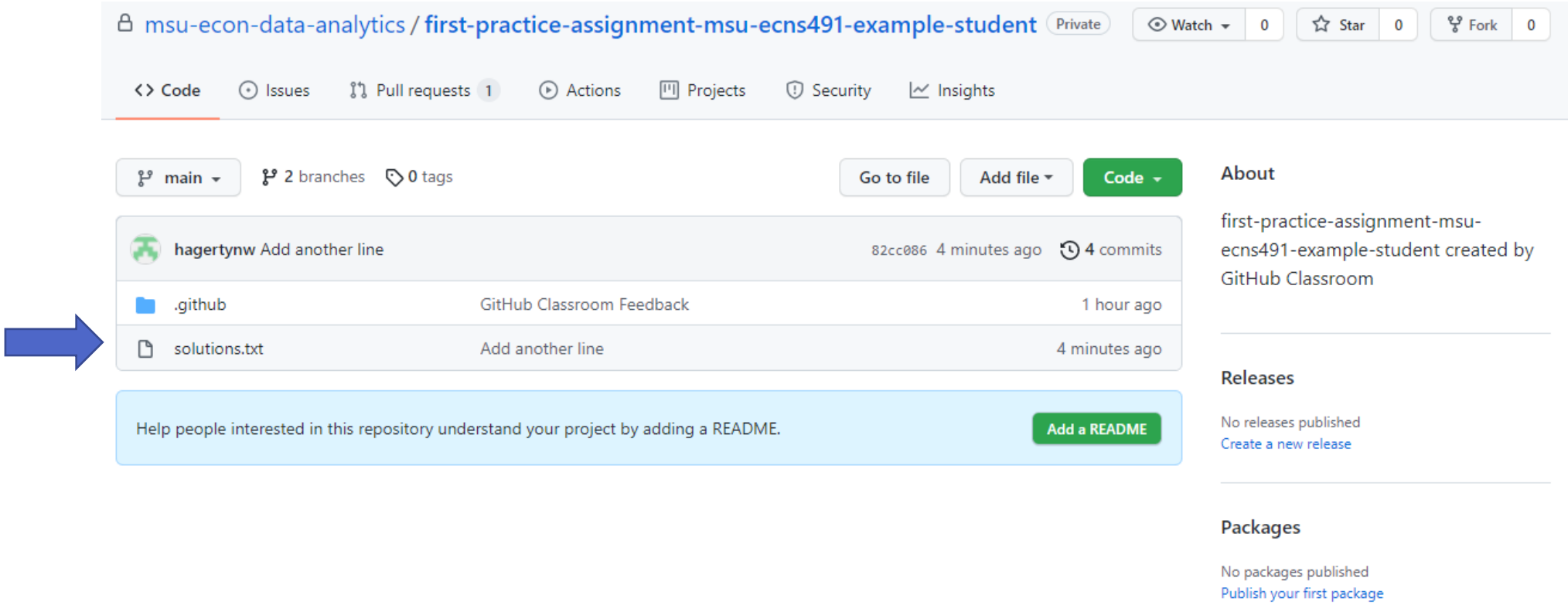
5. **Push** your commit to GitHub

- Commit is only local (your changes aren't on GitHub yet)
- Now we need to **push** the commit(s) to the remote GitHub repository
- Push uploads your changes to the cloud (GitHub)

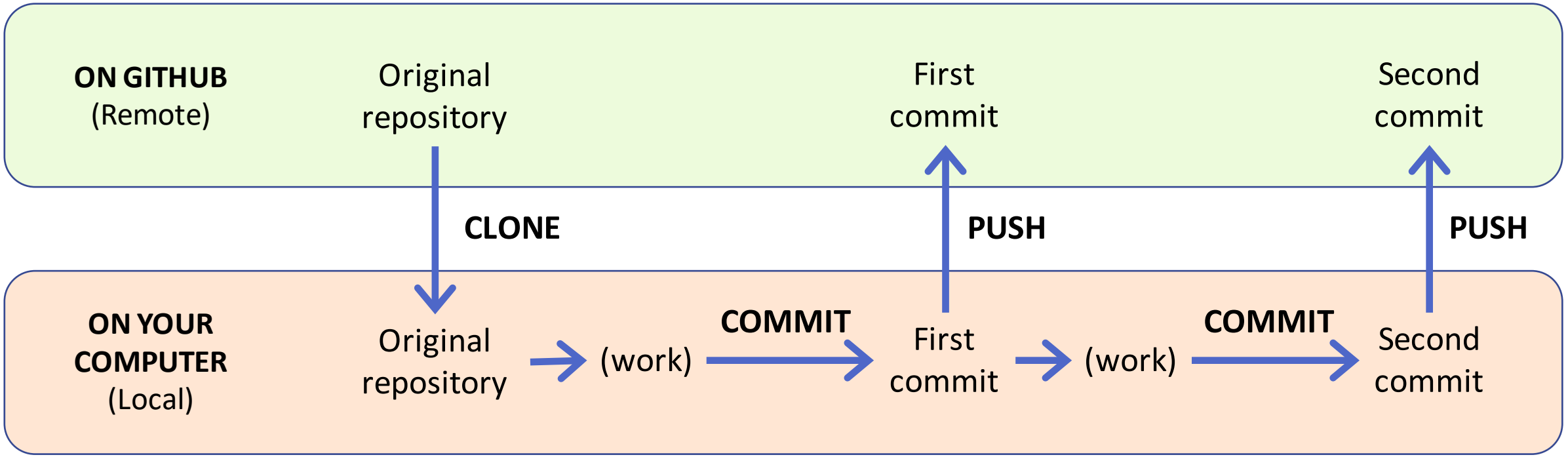


5. **Push** your commit to GitHub

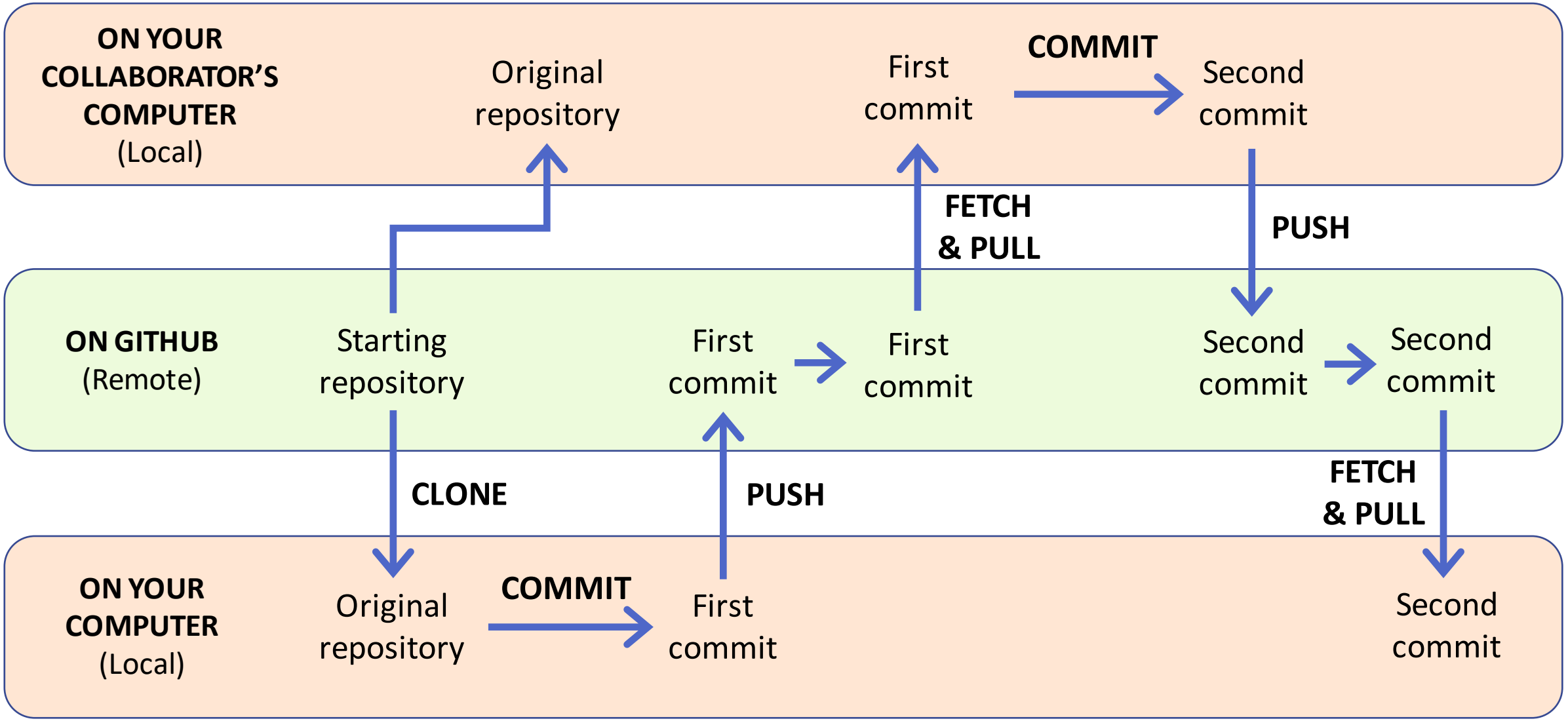
- Now, back on GitHub, you can see the new files you added



Basic workflow (only 1 contributor)

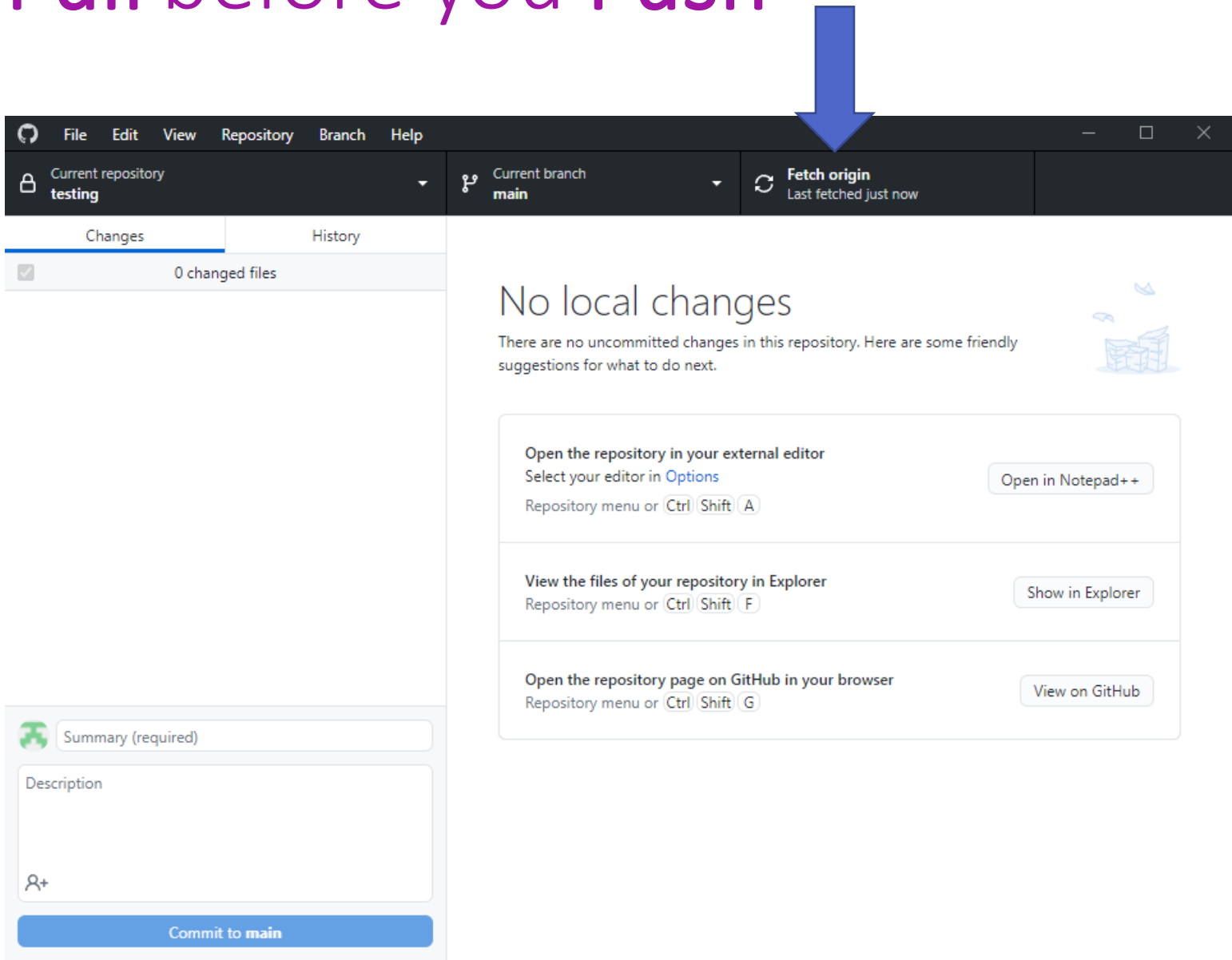


Example collaborative workflow



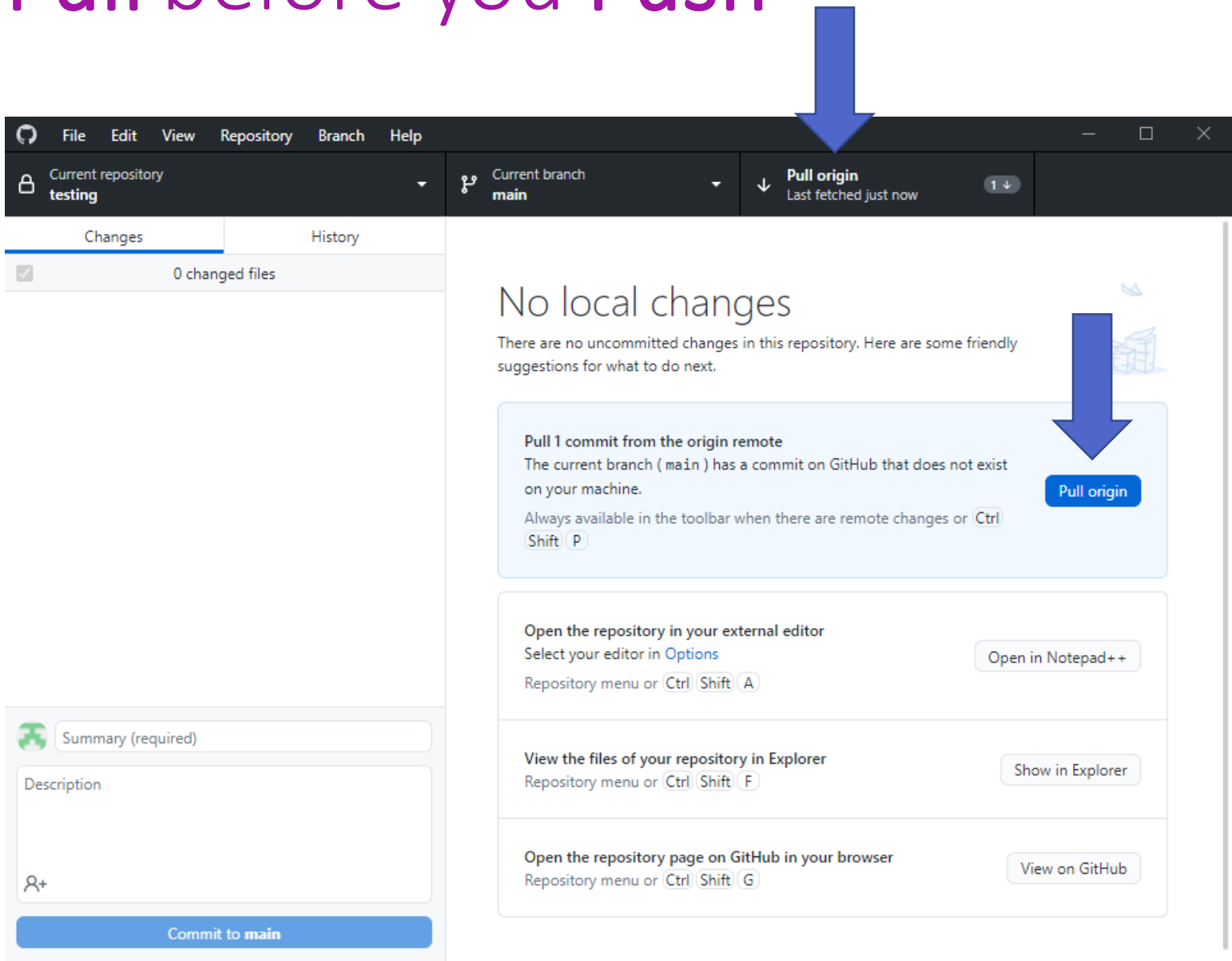
Always **Fetch** and **Pull** before you **Push**

- Your collaborator might have made changes since you last worked on it
- **Fetch** to check for changes



Always **Fetch** and **Pull** before you **Push**

- Your collaborator might have made changes since you last worked on it
- **Fetch** to check for changes
- **Pull** to download their changes
- Resolve any merge conflicts
- Now you can **push**!



Many more features & workflow options

(All optional, but very useful for collaborating)

- Forking and pull requests: https://guides.github.com/activities/forking/
- Branches and merges: https://guides.github.com/activities/hello-world/
- For much more, see the other “Git and GitHub” resources on the course resource list: https://github.com/msu-econ-data-analytics/course-materials#git-and-github