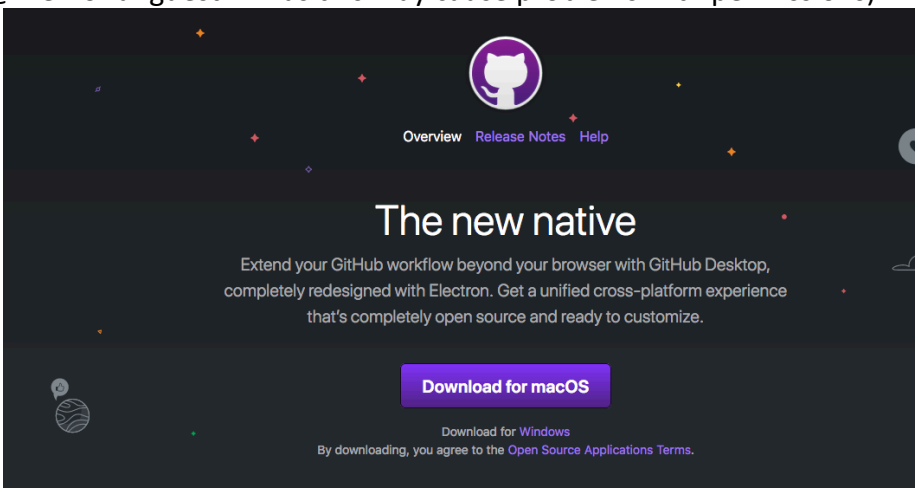
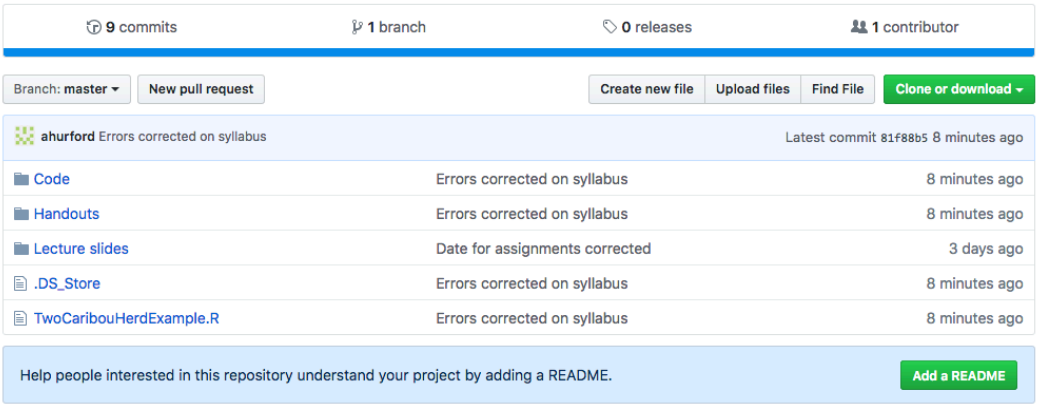
1. Installing Github Desktop

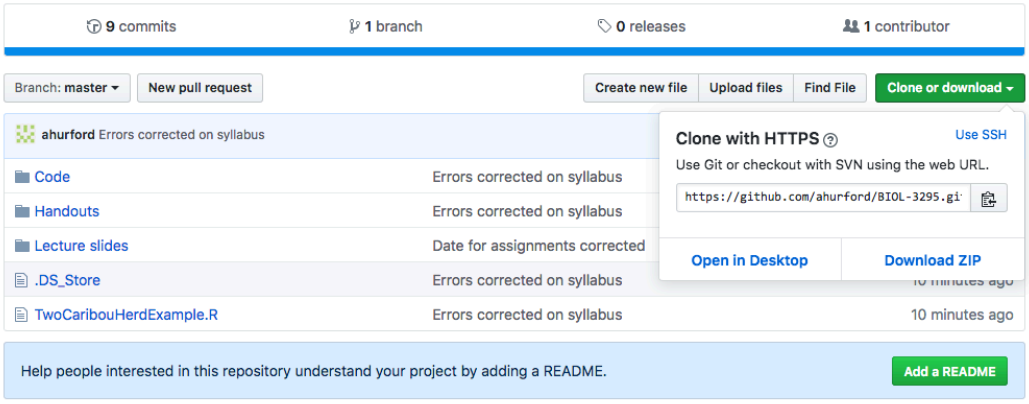
1. Install Github Desktop https://desktop.github.com/ (Note do not do with over the @memorial-guest wifi as this may cause problems with permissions)



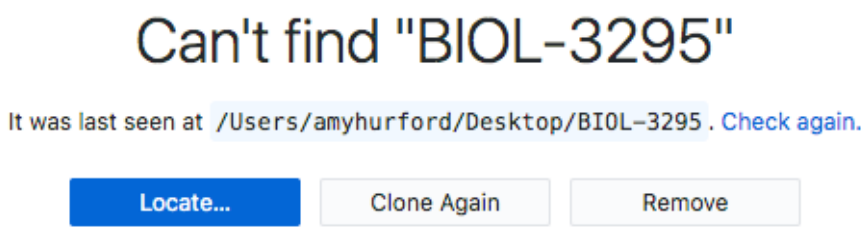
2. Go to the class git repository https://github.com/ahurford/BIOL-3295
3. Click on the green **'Clone or download'** button



4. Click the left option **'Open in Desktop'** (if you have not installed Github Desktop as per step 1, then you may choose 'Download ZIP' or you can just view the files on the Github website, .pdf versions of the file have been created for this purpose).



5. My computer then gives a 'Launch in Application' window and I select 'GitHub Desktop' and 'Open link', but you may get something different depending on your computer.
6. My Github Desktop now displays the following screen:

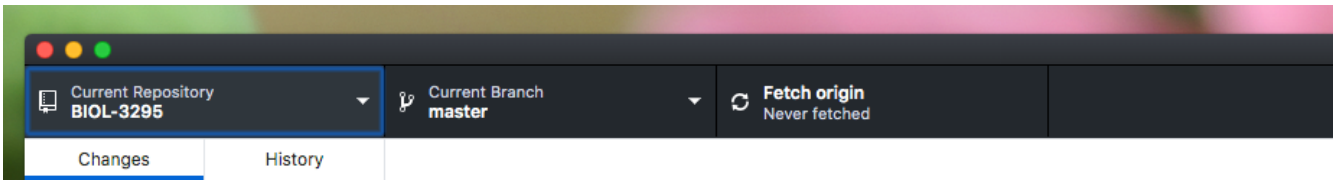


This is because I deleted my BIOL-3295 folder for the purpose of writing this guide. If you haven't downloaded before you'll likely have a different screen.

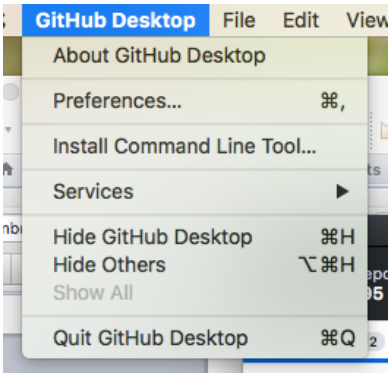
- "Locate…" means the software suspects that I may have moved the "BIOL-3295" folder from my Desktop (where this folder was last seen), and Github Desktop asks if I would like to guide it to the new directory where I moved the folder.
- **"Clone Again"** is the option I will choose. This will copy all the folders from the Github repository to a folder on my Desktop (I believe I had previously specified Desktop as the location that I wanted these files to appear in).

7. After I click **"Clone Again"** a folder "BIOL-3295" appears on my Desktop and this folder contains all the latest files for the class. If it is not clear where your files have downloaded to then search for the folder "BIOL-3295" on your computer as you would search for any other files.

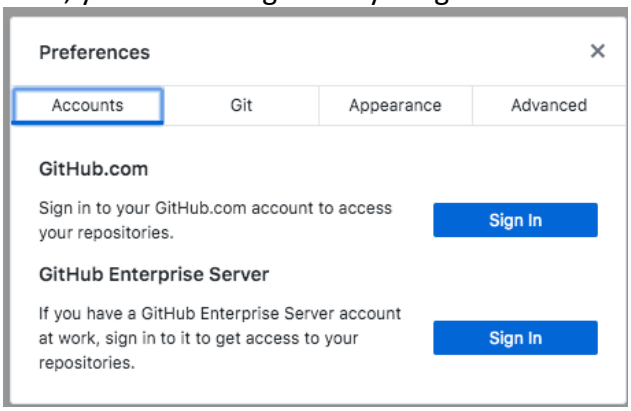
8. The usefulness of Github Desktop will become apparent later, as it is very easy to get the most recent course material using Github Desktop. With the current repository selected as "BIOL-3295" and the Current Branch as "master", click on **"Fetch origin"** and any new files will be downloaded to your BIOL-3295 folder.



9. The final step is to link your Github account to your Github Desktop. Sign-up for Github here: https://github.com/join. Next in the **"Github Desktop"** menu, select **"Preferences"**.



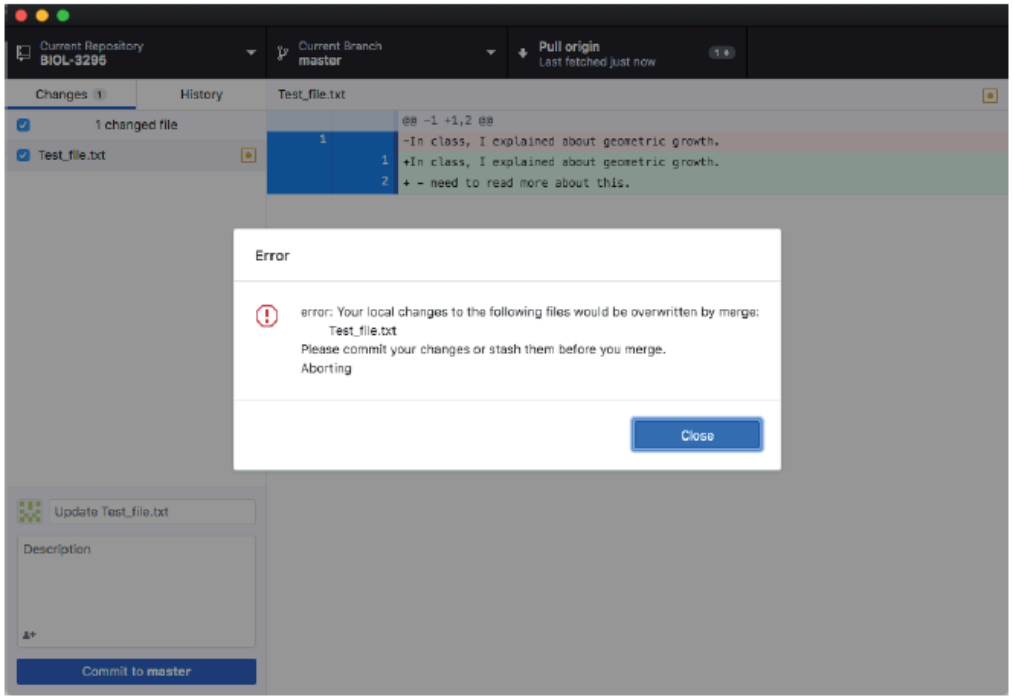
10. Next, you want to sign in to your github.com account using your credentials:



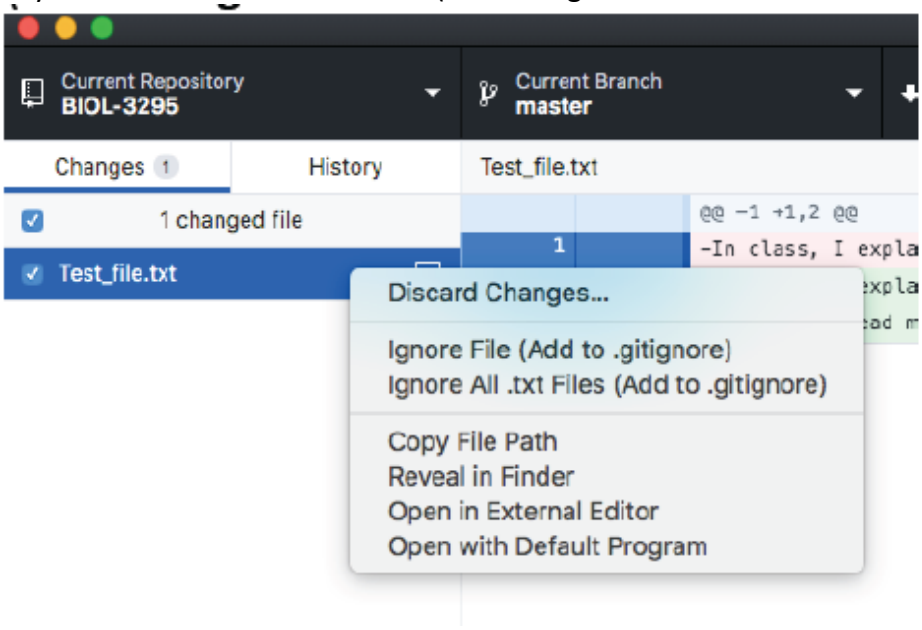
(I'm not certain that ,at this stage of the class this is necessary, but for Labs 5 and 8 you will need to submit your own files to the class repository, and you will need to be logged in for this).

2. When you make changes to your local files (which causes conflicts)

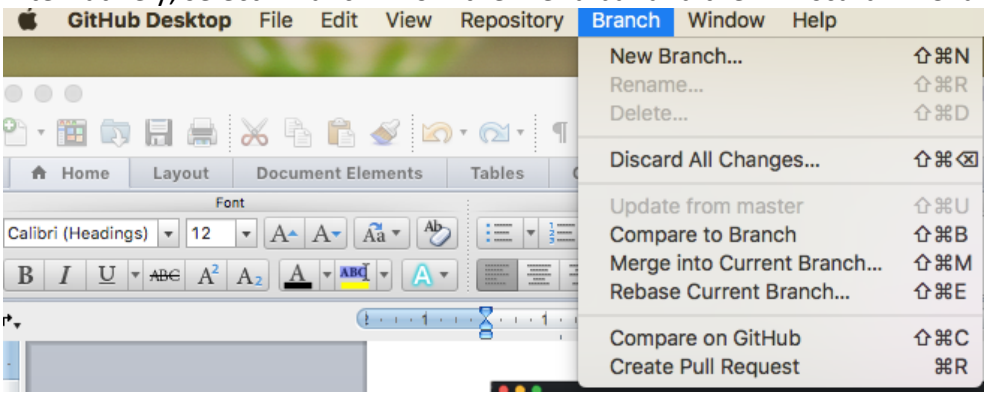
You might add notes to your local copy of the lecture slides, or you might edit an R script while working on a lab. This may cause conflicts as the version of the files on the remote server doesn't match your local versions, and this will prevent you from fetching the latest course files.



On the left hand side in the above photo we see that there are conflicts on the file 'Test_file.txt'. We can right click (or control click) on the yellow square with the circle inside and then select 'Discard changes…'. After you have discarded all of your local changes, the conflict will be removed, and you can now sync to the latest class material ('Fetch origin' with Current branch as 'master').



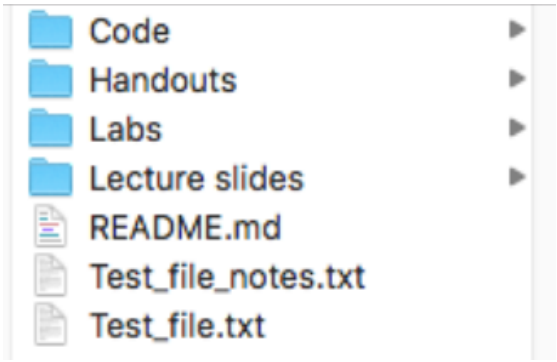
Alternatively, select "Branch" from the menu bar and then "Discard All Changes…."



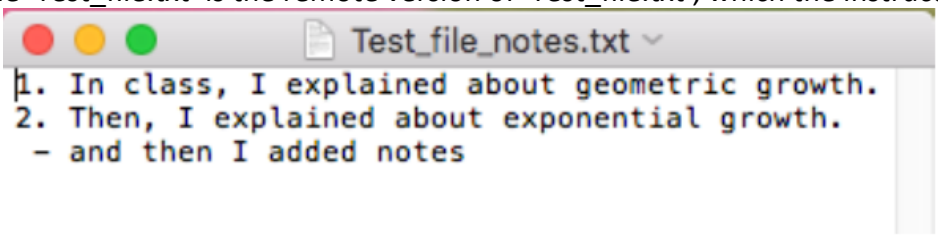
This solution allows to fetch the latest class material, but you have lost all of your local changes by choosing to "Discard Changes..". The next to sections discuss how to avoid losing your changes.

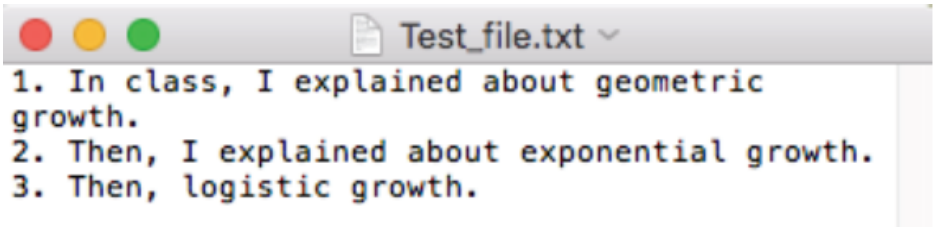
2a. Avoiding conflicts by renaming files

Suppose, we instead rename the local file with our edits. This is one option that appears to work.



Here, 'Test_file_notes.txt' is my local copy of 'Test_file.txt', which I have added some personal notes to, while 'Test_file.txt' is the remote version of 'Test_file.txt', which the instructor may modify.

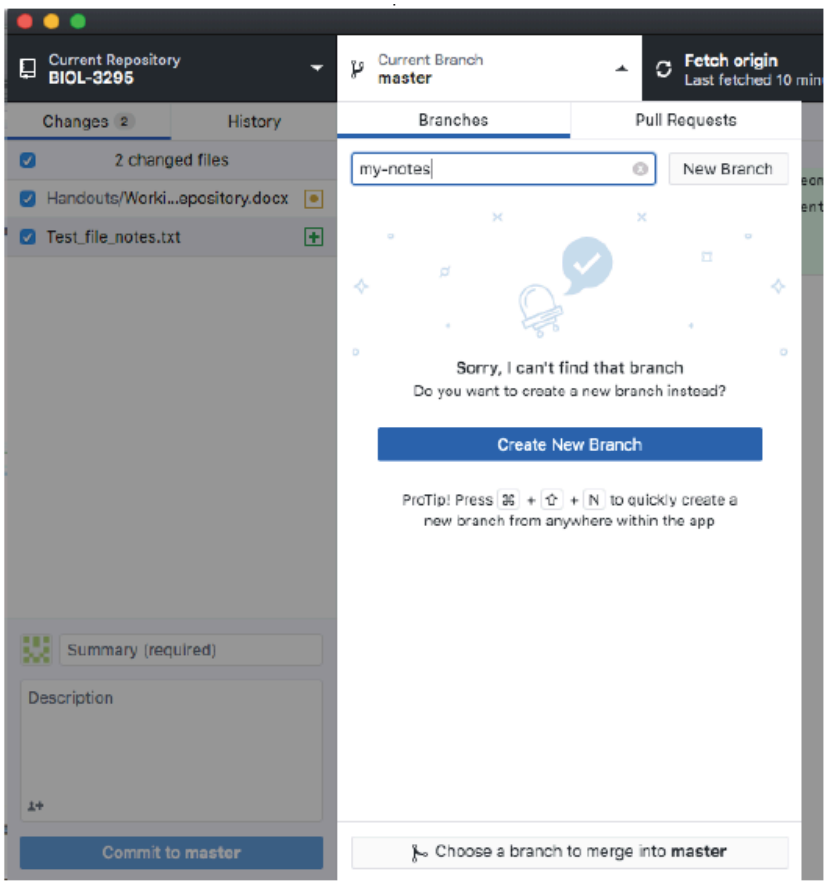


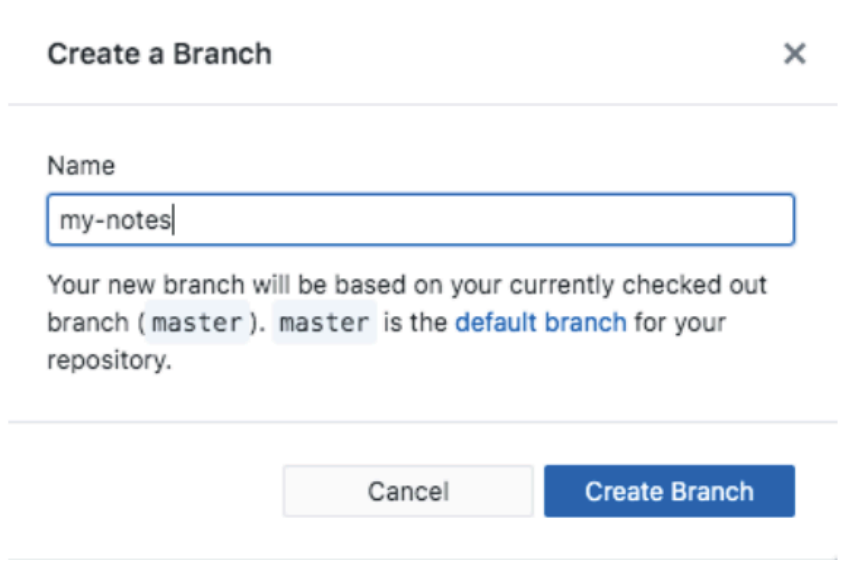


And so we can see that the remote (and local) version (Test_file.txt) has had "3. Then, logistic growth." added since you wrote your note "- and then I added notes", which only appears on the file 'Test_file_notes.txt'

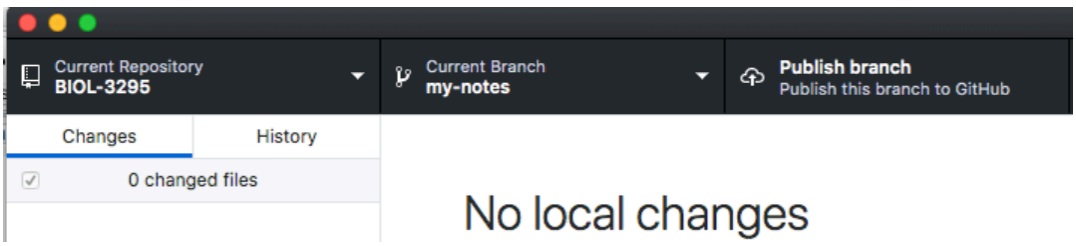
2b. Avoiding conflicts by creating a New Branch

Instead of renaming files you can fork the repository by creating a branch. Click on the Branch tab:

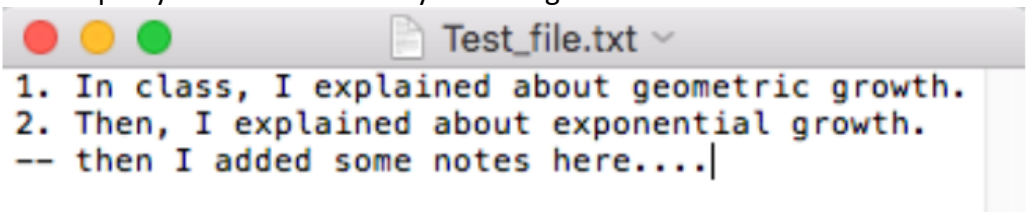




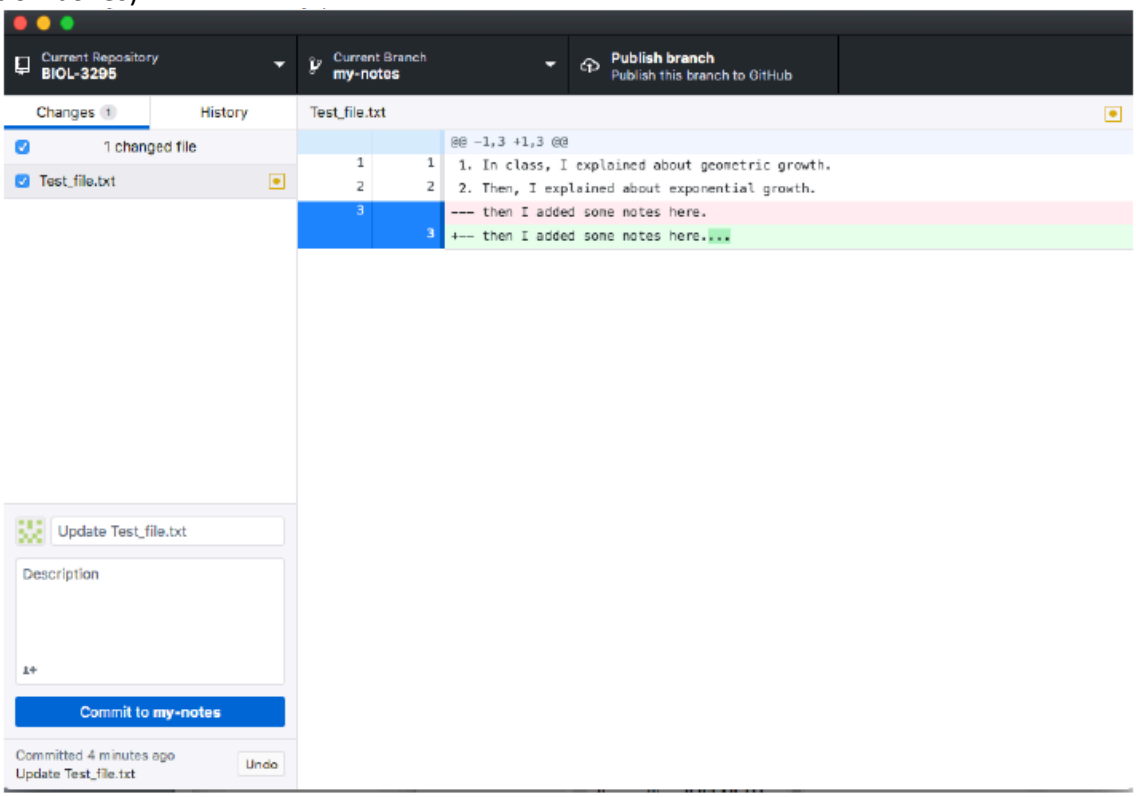
After creating your branch, make sure that your Current Branch is 'my-notes' when you start editing your local files. See the centre tab below is set to **my-notes**:



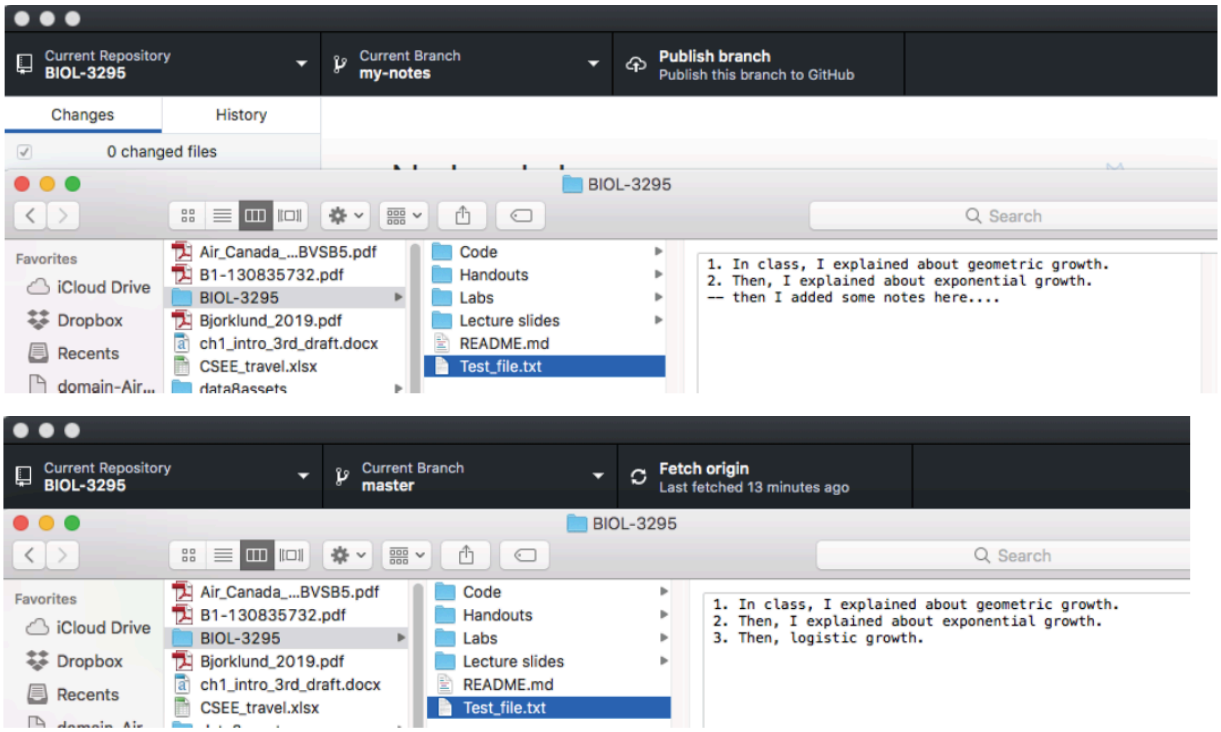
Now you can open your files and make your changes:



To save these changes to your branch, you want to commit them, by clicking the blue **"Commit to mynotes"** button in Github Desktop. (If this isn't available to click, you may need to fill in the title and description boxes).



Now you have 'forked' the repository - you have two local versions (or branches): 'master', which is an exact copy of the remote version (i.e. the instructor provided version) and 'my-notes' which is your personal copy with your notes added. To view either use the middle tab "Current Branch" in GitDesktop and then open the files as you would normally (outside of Github Desktop).



See how in the two photos above the version of Test_file.txt is different depending on the branch I have open in the middle tab of Github Desktop?

To summarize you have two ways to add notes to your course files: 1) rename the file with the notes, or 2) create a branch and commit the changes. But perhaps you are still wondering “but why don’t I just add my notes to the master branch?” This will create conflicts where, when you try to sync to new course material, there will be confusion about which version should take precedence (your local copy with your notes, or the newer remote version) and you will have to resolve this conflict before proceeding.

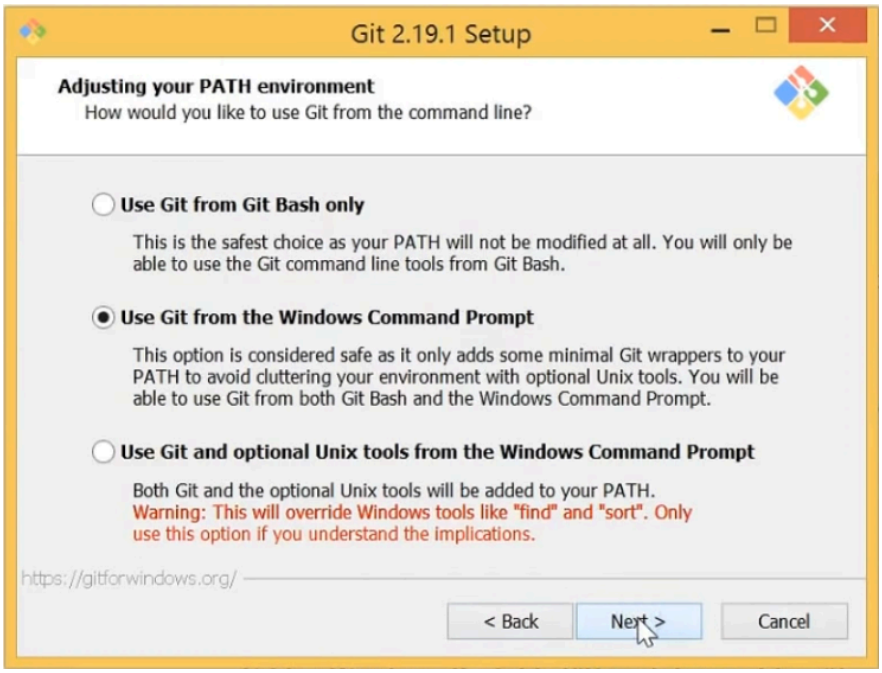
Or perhaps you are wondering “why don’t I just commit my changes to master after I add my notes?” As a student in the course, you don’t have this option because I am the administrator of “master”. Instead, you will be forced to create a branch with your changes, and then you can make a “pull request” to have your changes be considered for “master”. As the administrator, I would be notified of the “pull request” and would have to approve your changes before they could be added to master. In short, “master” is the version that is everyone’s copy, and so it doesn’t make sense for you personally to change the master branch, you only want to change your own copy, so you make a branch for that purpose and commit your changes on it.

2. Using Git through the command-line interface (Optional – Only do this if you do not want to use GitHub Desktop)

A. Set up. This will vary according to the Operating System that you use, follow the respective instructions below.

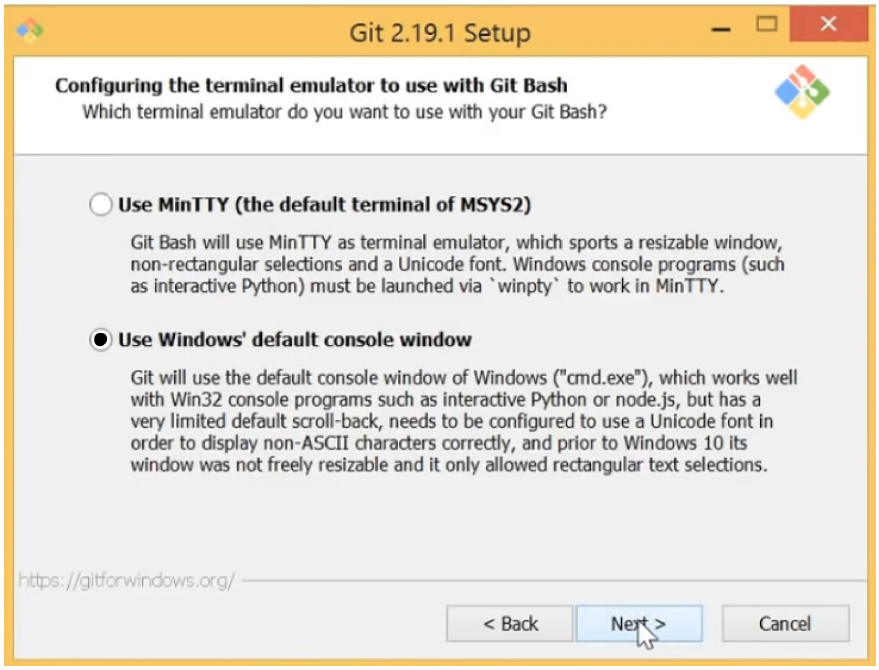
i. For Windows users:

1. Download the Git for Windows from https://gitforwindows.org/
2. Run the installer and follow the steps below:
    1. Click on "Next" four times (two times if you've previously installed Git). You don't need to change anything in the information, location, components, and start menu screens.
    2. Keep "Use Git from the Windows Command Prompt " selected and click on "Next". If you forget to do this rerun the installer and select the appropriate option.



3. Click on "Next" two more times.

4. **Select "Use Windows' default console window" and click on "Next".**



5. Click on "Next" two more times and then click on "Finish".

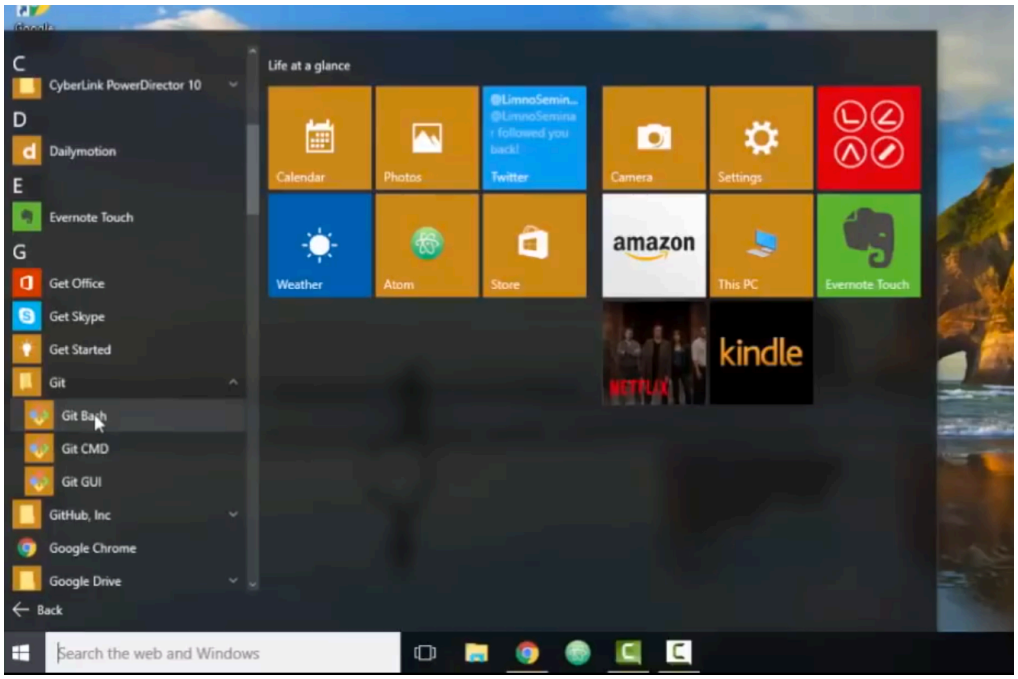
3. If your "HOME" environment variable is not set (or you don't know what this is):
    1. Open command prompt (Open Start Menu then type cmd and press Enter)
    2. Type the following line into the command prompt window **exactly** as shown:

        setx HOME "%USERPROFILE%"

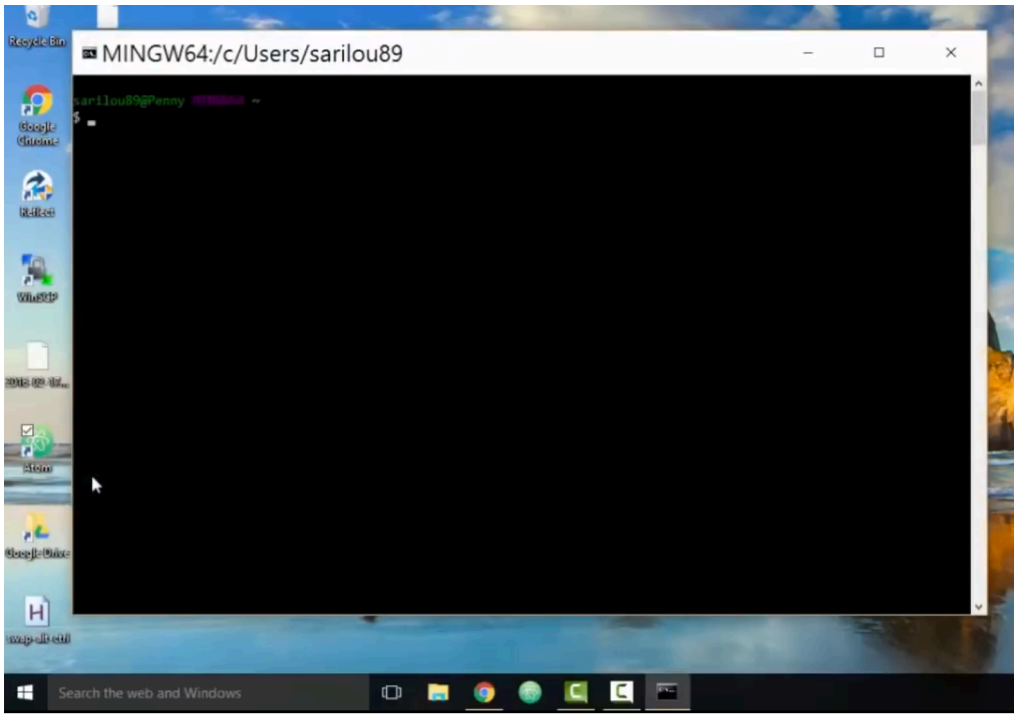
    3. Press Enter, you should see SUCCESS: Specified value was saved.
    4. Quit command prompt by typing exit then pressing Enter

This will provide you with both Git and Bash in the Git Bash program.

Once installed, you can open the **Git Bash** terminal from the Apps menu.



The Git Bash terminal should look like this:



ii. For MacOS users:

- **For OS X 10.9 and higher**, install Git for Mac by downloading and running the most recent "mavericks" installer from https://sourceforge.net/projects/git-osx-installer/files/ Because this installer is not signed by the developer, you may have to right click (control click) on the .pkg file, click “Open”, and click “Open” on the pop up window. After installing Git, there will not be anything in your /Applications folder, as Git is a command line program.
- **For older versions of OS X (10.5-10.8)** use the most recent available installer labeled "snow-leopard" https://sourceforge.net/projects/git-osx-installer/files/

You can open the terminal from /Applications/Utilities or from the Launchpad.

iii. For GNU/Linux users:

Open the terminal and install Git according to your distro's package manager:

- For Debian/Ubuntu run `sudo apt-get install git`
- For Fedora run `sudo dnf install git`

B. Basic Git commands

1. Open the terminal
2. Go to the directory where you want to work from (Desktop recommended)

```
cd Desktop
```

3. The **first time** that you use Git you will need to **clone** the project. This will set the REMOTE as “origin” (which is where the repository came from)

```
git clone https://github.com/ahurford/BIOL-3295
```

4. Change directories to your newly created repository

```
cd Desktop/BIOL-3295
```

5. Avoid conflicts by creating a branch called my-notes

```
git checkout -b my-notes
```

6. Switch to your new branch, so you can work on it

```
git checkout my-notes
```

7. You can check the status of any changes in the directory

```
git status
```

8. Commit your changes if you want to save them

```
git add MY-FILE-NAME
git commit -m "A SHORT COMMENT TO DESCRIBE THE INTENTION OF THE COMMIT"
```

9. Send changes to your account on github.com, this will push all your local commits to your remote fork of the repository

```
git push NAME-OF-YOUR-GITHUB-REPOSITORY my-notes
```

10. Alternatively, discard the changes that you made

```
git checkout -- MY-FILE-NAME
```

A basic workflow would be: changing repositories (4), switching branches (6), checking the status of the repository (7) and commiting or discarding your changes (8 or 10).

To keep your local repository up-to-date with the class repository (**it is important to do this before every lab to work on an up-to-date copy!**):

11. Switch to the master branch to pull the latest changes from there

```
git checkout master
```

12. Download the latest changes into the local repository

```
git pull origin master
```