

# Introduction to GitHub

Kelly Mahaffy, The University of Connecticut  
kelly.mahaffy@uconn.edu

*These exercises are in part inspired by the [Software Carpentry Version Control with Git](#) lesson modules and were adapted to fit the current workshop.*

## Before We Begin:

- If you see <pointy-brackets>, it means you need to replace them with something unique to your account, item, or repo! (often a name)
- One drawback to Zoom is that it makes communicating with those outside of your breakout room difficult. I will be popping in and out of rooms, but please feel free to text me at () or to add comments to this google doc if you need help and I will pop over to you ASAP!

## Exercise 1: Collaboration At Its Finest

To begin, decide who will be the owner and who will be the collaborator first. Don't worry, we will do this twice so you will both get an opportunity to do both roles!

Step 1- Owner: Create a repository and add your collab

- a. Log in to github.com, find the small plus in the upper right hand corner, and select "New Repository"
- b. Decide on a repository name, give it a short description if you want, decide if it will be public or private, and check the box next to "Initialize this repository with a README"

## PROTIPS!

- No Spaces- when deciding on any file names in github (including repo names) do not use spaces. This is because, for collaborators who may be using the command line, spaces are read as part of the syntax and they will not be able to access your files if there are spaces in the name.
- Public vs. Private- public repos can be seen by anyone with a GitHub account, while private repos are reserved only for yourself **and** your team. They can only be shared through invitation. You can change the settings on your repo multiple times throughout a project.
- README- a README file is a document where you can give a detailed description of your project, the work you have done, how to use the code in the repo, etc. This document is standard and will be expected, and used, by anyone who visits your repo.
- .gitignore- Adding this extension will keep this file from being accessible to others, even if your repo is public. This is a way to hide what you are currently working on, etc.


Now, you are ready to click “Create repository”!

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).


---


**Owner \*** **Repository name \***

 kellymahaffy1 ▾

Great repository names are short and memorable. Need inspiration? How about **congenial-octo-happiness**?

**Description (optional)**

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

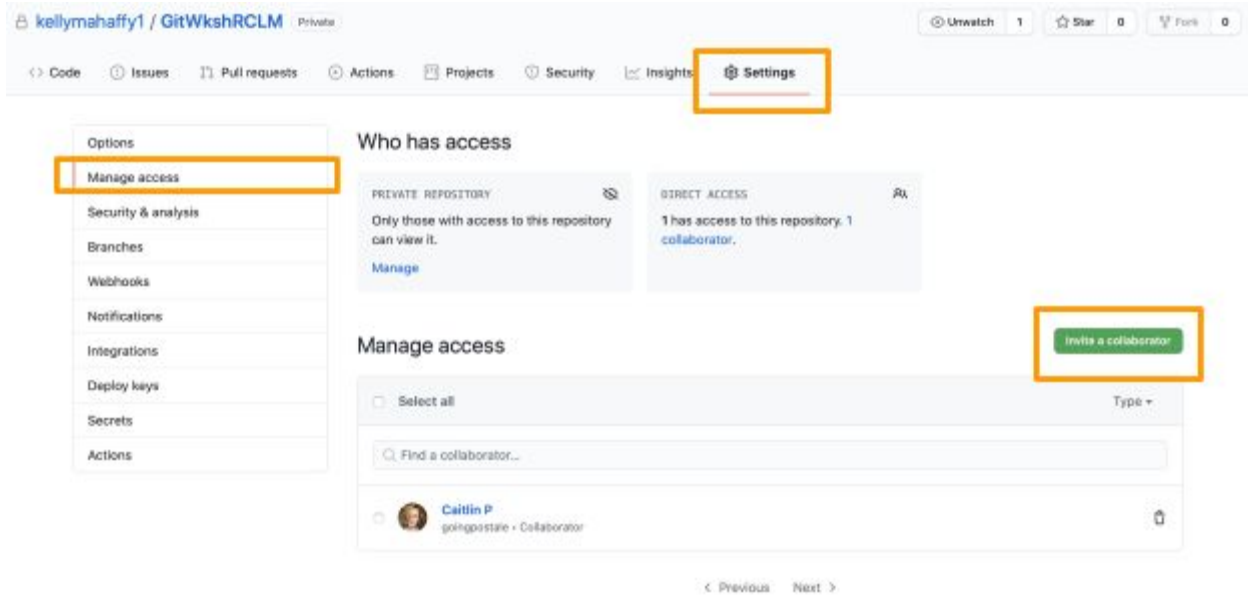
☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

**Add .gitignore:** None ▾ **Add a license:** None ▾ ⓘ

**Create repository**

Step 2- **Owner:** add collaborators to your project

1. Once on your repo, navigate to “Settings” (top right)
2. Find “Manage Access” from the menu on the left
3. Invite a collaborator via email, username, or name

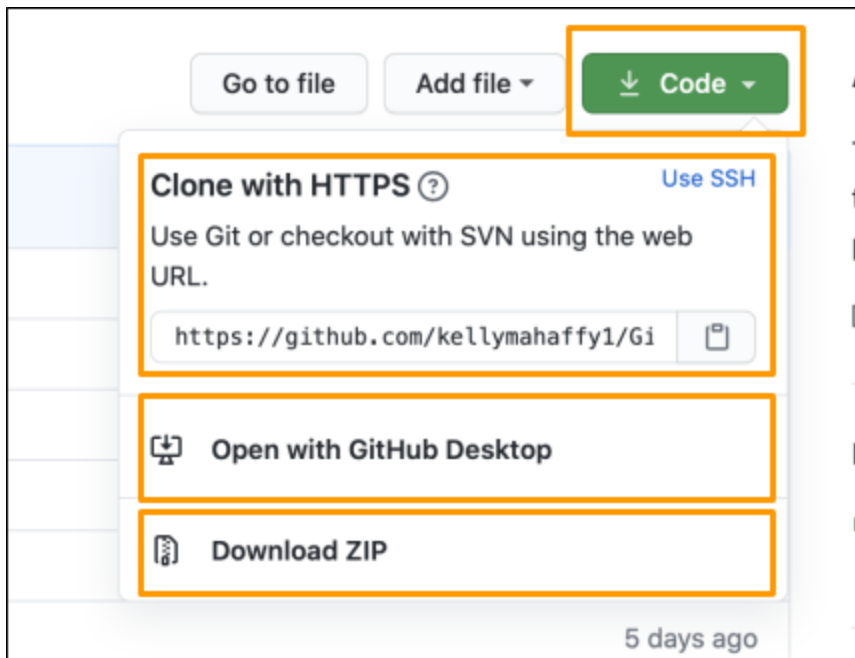


### Step 3- **Collab** accepting invitation

- Collaborators are not automatically added, they must accept the invitation.
- Check your email or go to <https://github.com/notifications> to accept access.

### Step 4- **Both**- Cloning to your computer **\*\*OPTIONAL\*\***

If you want to work on parts of your repo locally, you can either drag and drop to upload files or you can clone a repo to your computer. While this function is more important if you plan to use git from the command line, it might be useful for others as well.



#### Ways to clone:

1. **COMMAND LINE**- Click on the clipboard next to the url and navigate to the command line. Either navigate to a good directory (`cd <filepath>`) or make a directory (`mkdir <filename>`) and then use `git clone` and paste the url. Hit enter to clone.
2. **DESKTOP APP**- Simply click the "Open with GitHub Desktop" option

once you have downloaded desktop and logged in. The repo will be copied to your desktop and you can push and pull from your local device.

3. **DOWNLOAD A ZIP-** If you want to download the full repository over a single file, you can download a zip version which can then be unzipped and single files within can be accessed. Note, this does not automatically update! If you plan to use this, you will need to manually redownload after updates are made.

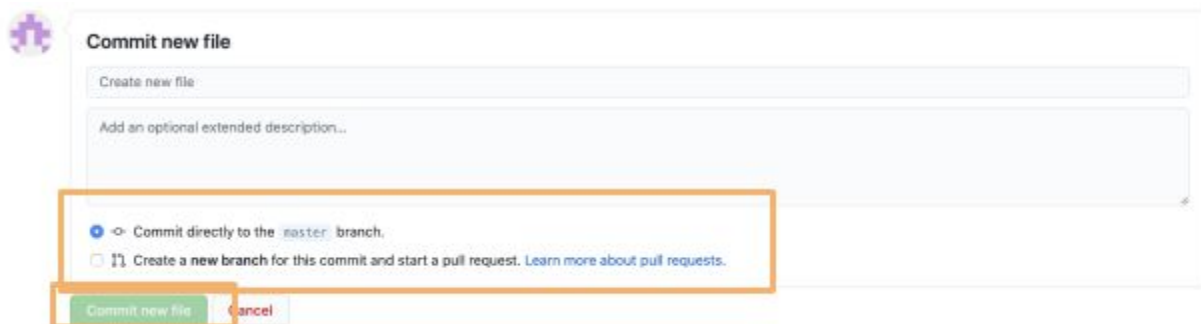
## Exercise 2: Create and Conflict

FOR THOSE ON THE WEBSITE:

Step 1- **Owner-** Create a new txt file by navigating to the “Add File” button (top right) and then selecting “Create new file”

- Protip- This is a great time for the **Collab** to either locate a good txt editor on their computer (I like atom, available here- <https://atom.io>) or to practice uploading random documents into the repo using the “Add File” button and selecting “Upload Files”
  - If you have them, try uploading code files to explore how they render. If you don’t, have a look at Kelly’s shared repo for this workshop at some point to see how R files, etc. render in GitHub

Step 2- **Owner-** Commit your file (after naming it in the top left corner) by scrolling to the bottom of the screen and clicking “Commit new file”



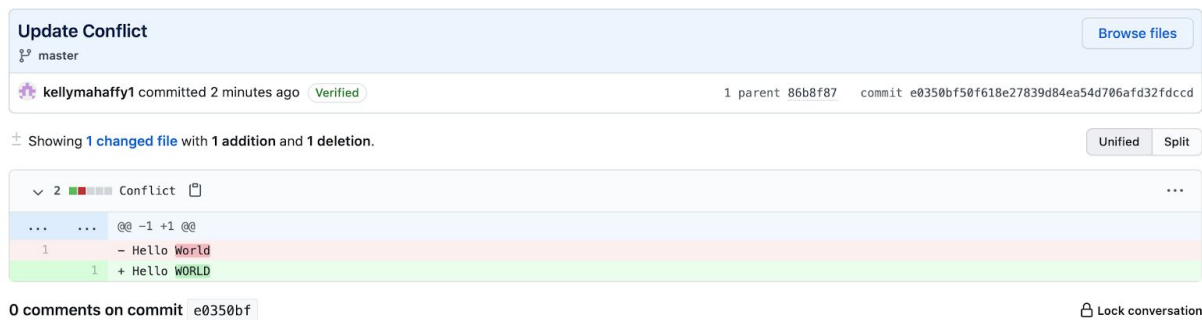
- Protip- You can commit (save) your document to the master repo or create a branch, like a small section of a larger repo, for the folder. If you create a branch, it will generate a pull request (We will get to this soon!)

Once you have committed your changes, head to your repo to view the document!

Step 3- **Both Owner and Collab-** Open the file that was just created by the owner and begin typing on the same line. Your goal here is to model what it would be like for two collaborators to have both edited a project, so you are trying to create a conflict between files.

- Both people should commit your document to the same branch (or master!).

- Go to your “Issues” tab (or click the file name, or look for a pop up, this hides in different places depending on if you are using the online, desktop, or command line plugin versions) and look at what GitHub has highlighted as the problem
  - Protip- Github will try to make this as easy as possible to spot, so anything that is added will be highlighted green, anything that has been erased will be highlighted red, and there will be the addition of some markers, tags, or hashes (often things like >=====, etc.) to point out the conflict to you.

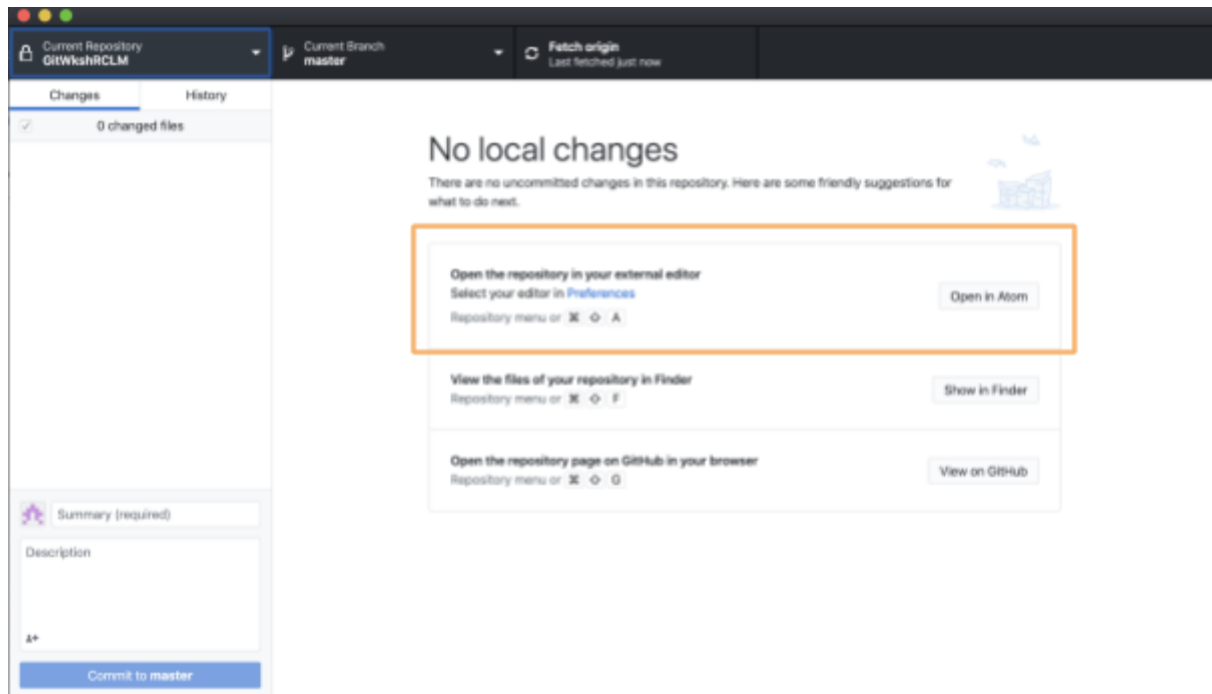


- So, this photo shows that the lowercase World was deleted and replaced with WORLD-- version control is cool!
- Now, resolve the conflict by either editing the text yourself, or using the blue highlight bars on the left to select changes that you want to keep.
  - If you are using the highlight bars, anything that is in blue WILL BE KEPT, while things that you do not select will not be kept!

FOR THOSE USING THE DESKTOP APP:

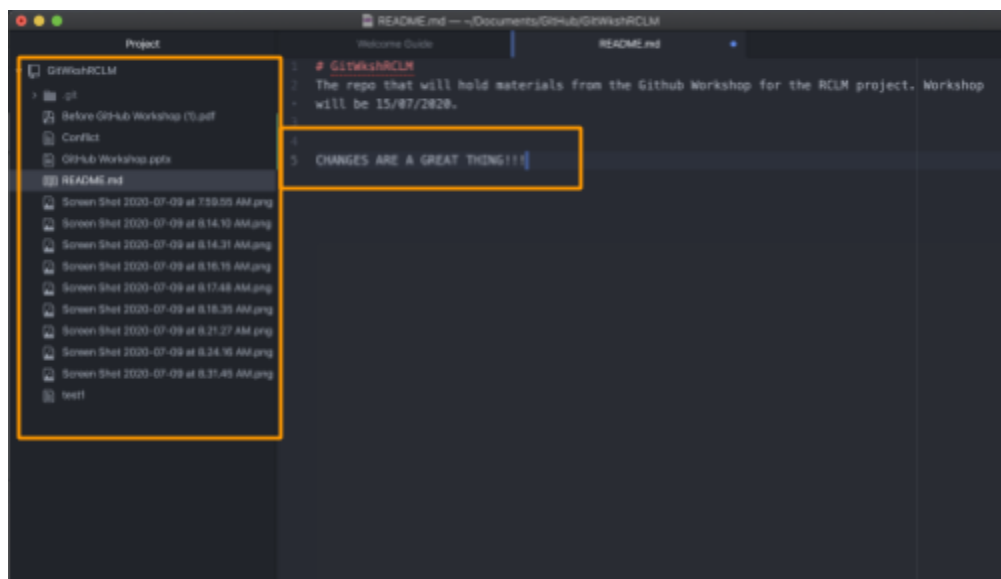
Step 1- **Both-** Be sure you have cloned the repo to your desktop (see above)

Step 2- **Both**- Select the option to open the repo in a text editor, for me it is Atom (my designated txt editor)

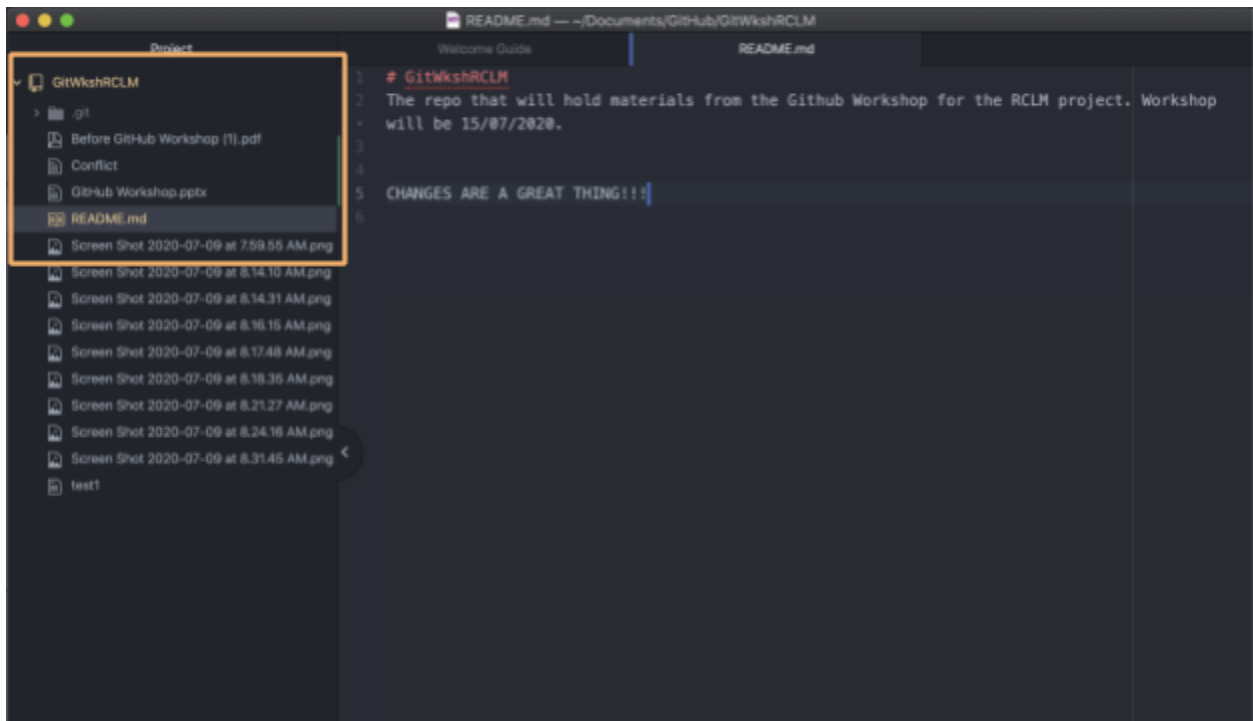


- Note- If you are creating a new document, you will need to either create it from within your repo (which you can do easily by using the “Show in Finder” option below, or drag (or save it) into your repo).

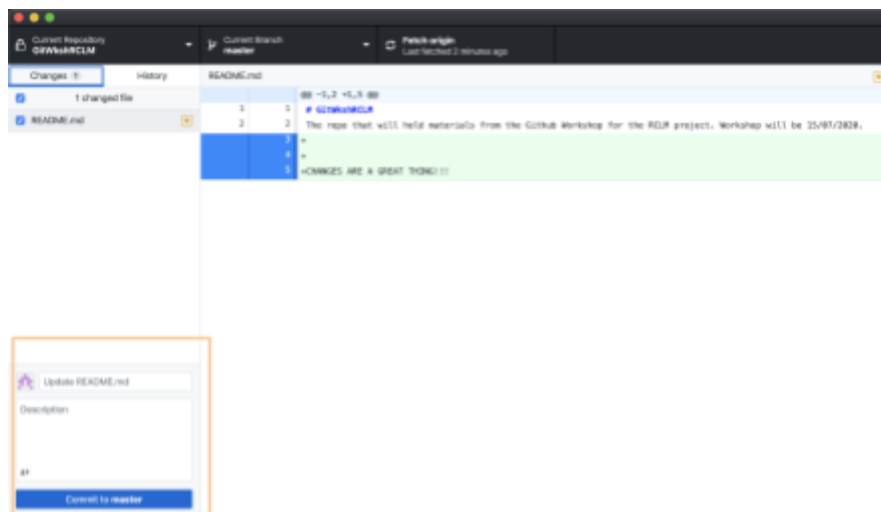
Step 3- Edit and save your changes to the document. You can select a document already in your repo on the side bar!



- Protip- In Atom and many other text editors, it will automatically highlight any changes saved locally but not committed to your repo for you! In this case, in orange on the side bar.

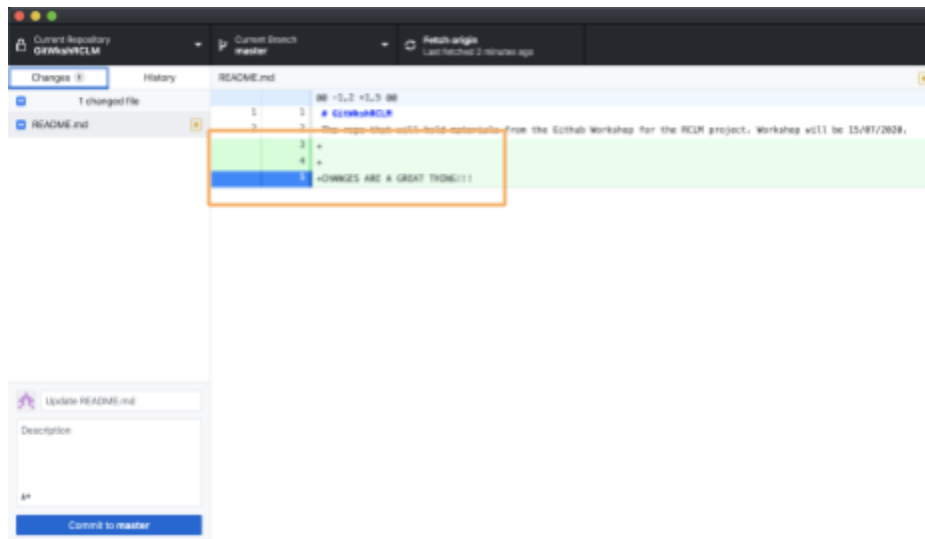


Step 4- Commit your changes to your repo by closing your text editor (make sure you have saved there, first) and clicking back into your repo. Your changes should automatically populate on the screen. You can commit them with the “Commit to Master” button in the bottom left of the screen (You may want to see the Protip below before committing changes!).



- Protip- The blue bars next to the highlighted area show changes. Things highlighted in green were added and things highlighted in red were deleted. If you do not want to keep

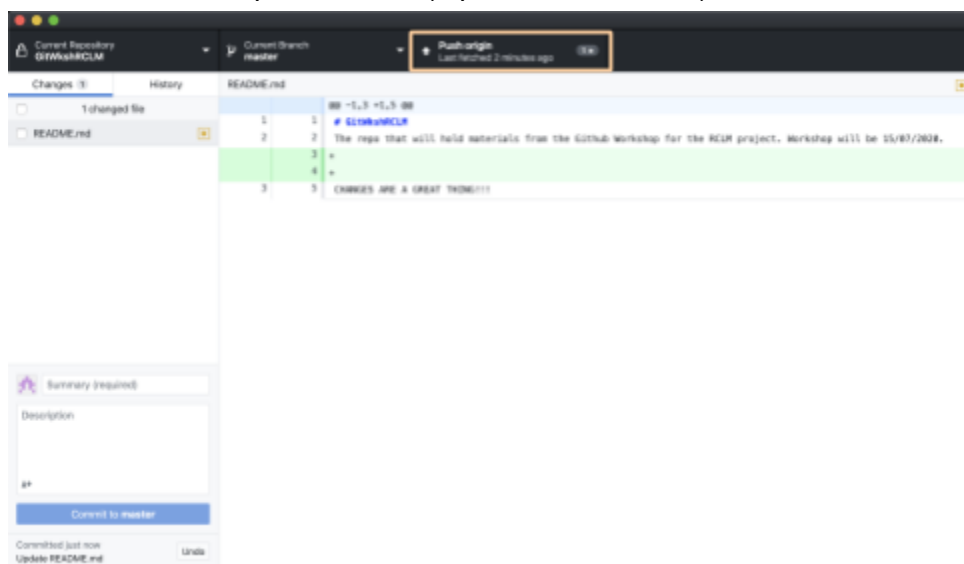
something, you can click so that the blue bar next to it is no longer blue, then it will not save. So, for example, if I do not want to commit the extra lines between my original text and my changes, my screen will look like this:



- An important reminder for this protip- this works to not save the changes that GitHub suggests, so if you click off of a line that was removed it will actually save that line. It can be a bit wonky to think about, so feel free to practice by deleting things!

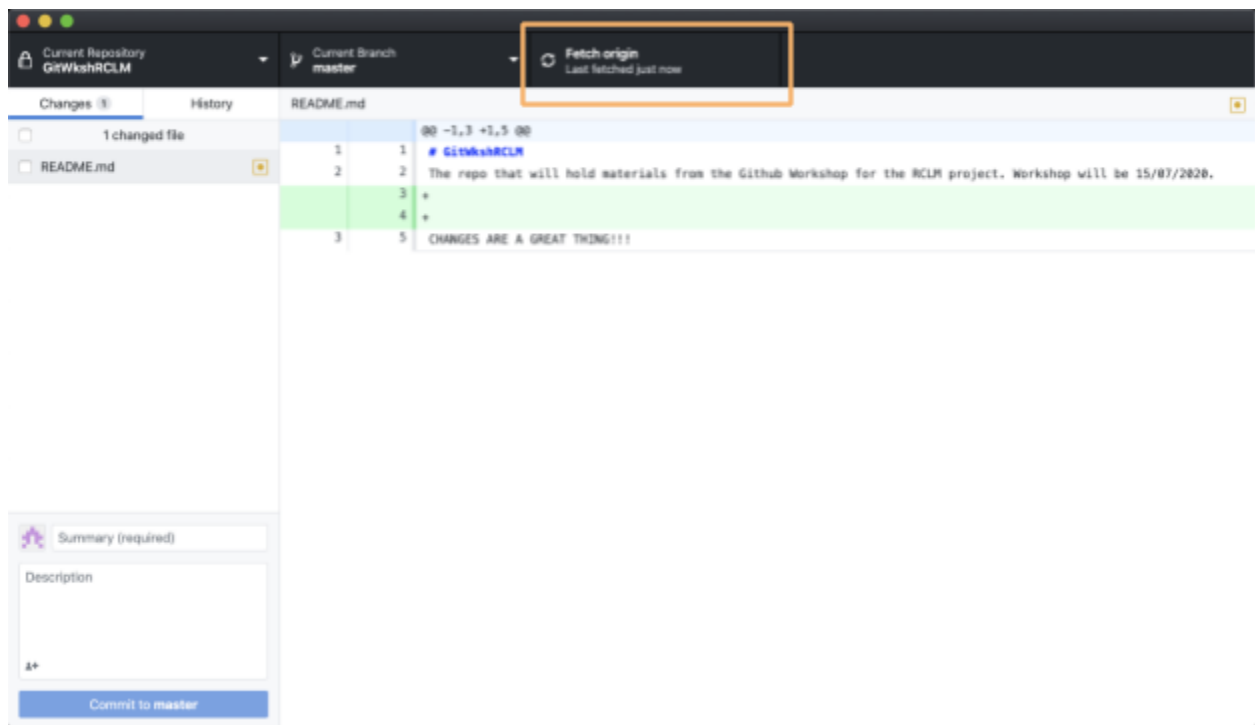
Step 5- You aren't quite done yet! Your changes are now saved locally, but you need to **push** them back to the version of the repo stored in the cloud so that your collaborators have them. THIS IS A SUPER IMPORTANT STEP!! Push things back often! Like, really often!

- To do this, locate the “push” button (top middle of screen) and click it





- Once it has pushed, it will give you the option to **Fetch**. You should fetch every time you open your GitHub so that you know you are working with the most recent versions of everything!



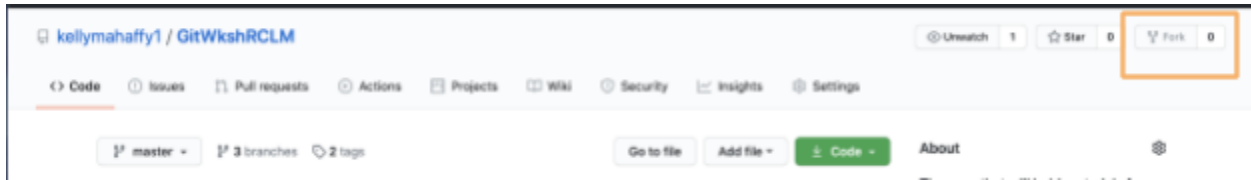
Whew, that is the hardest part of this tutorial. Take a stretch/pat on the back break!

### Exercise 3- Forking and Pulling

Forking and pulling are ways to collaborate on one project or to network and collaborate with those that you do not even know! This is a HUGE OPPORTUNITY for those who are publishing open access! With the right tags and description, you can get yourself hundreds of free copy editors!

Step 1: **Both**- Fork my RCLM repo

- Forking is when you clone an entire repository to your account. You can think of this like a fork, if the middle prong is the original repo you are simply copying an exact copy to a new prong. These repos are still connected in that your copied version still lists who originally created the repo and they still retain ownership over the original files, but you have your own version now that you can play along with
- First, visit <https://github.com/kellymahaffy1/GitWkshRCLM> and fork this repo to your own account via the fork button



## Step 2- Make changes

- Now that you have copied my repo, make some changes! I might suggest adding your name to the list of participants in the README, creating some conflicts in the Conflict file, or even adding a pic of your repo!

## Step 3- Pull Requests

- A pull request is a way of helping out the GitHub community, it is basically a way of asking the original author of a repo to pull your changes into their project.
  - One cool example- GitHub aficionado and general coding rockstar [Danielle Navarro](#) often adds her books to her GitHub and, as a result, they are some of the cleanest coding textbooks I have ever read!
- Once you have made changes, create a pull request by clicking “Create a new branch for this commit and start a pull request”



- PROTIP- Creating a pull request in a project that you are a collaborator on gives others a chance to review your work before it is added into the project. Creating a pull request of a cloned repo send that request to the original author, alerting them to some fix that you think needs to be made.
- Once a pull request is made, the owner of the repo can decide if they do or do not want to make the change suggested. If you get a pull request on one of your repos you will get an email and lots of notifications!

## You made it!!

I hope that you have seen some of the powerful ways that GitHub can be used as a collaborative and creative tool!

You can head back to the main room by leaving the breakout room (Leave, then click Leave Breakout Room) or text Kelly when you are done!