

GitHub

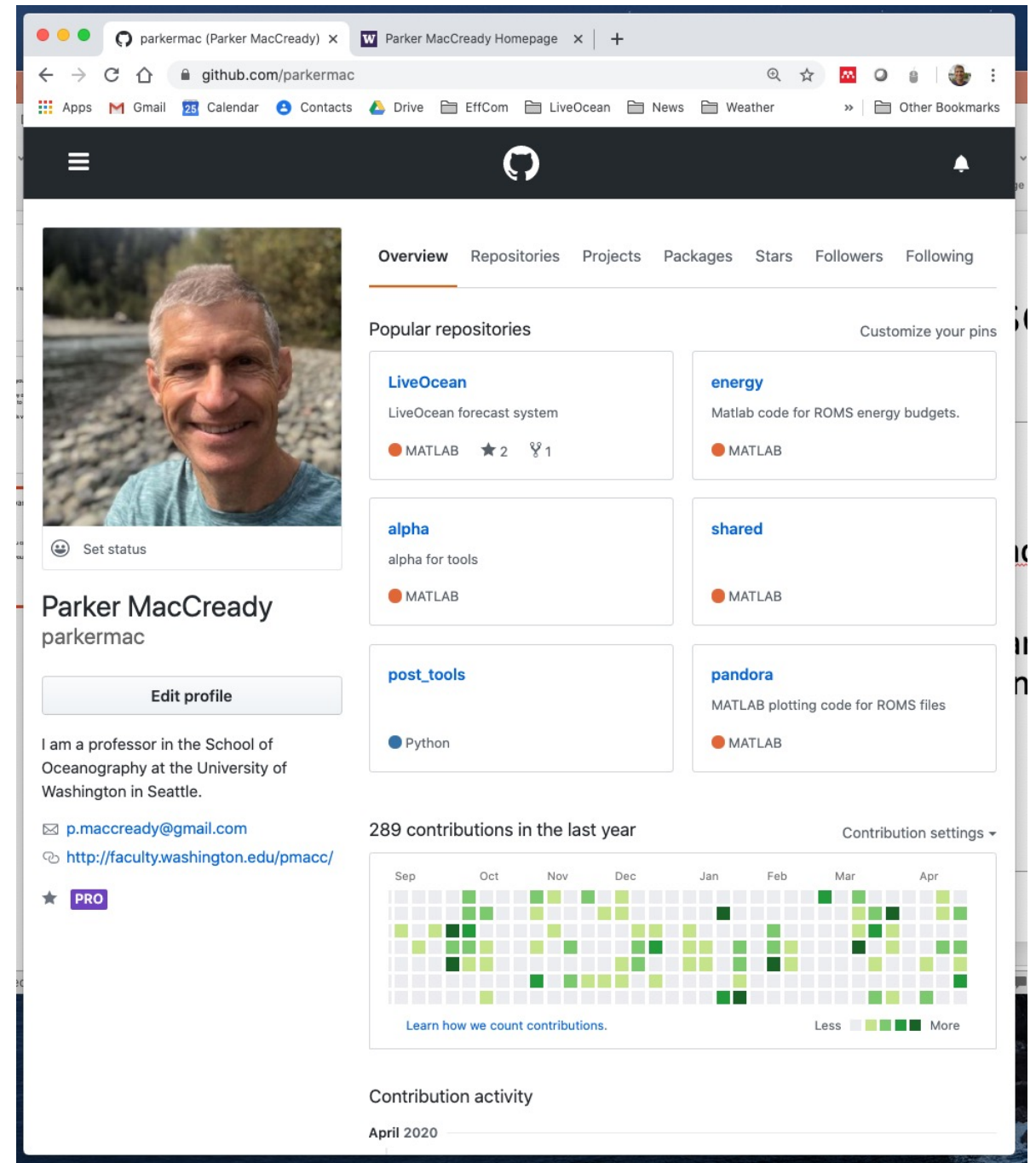
Keeping your code organized, backed-up, and easy to transport to any remote computer

What is GitHub?

- GitHub is a software system for keeping track of changes you make to a code project.
- It is also a place in "the cloud" where you store a safe copy of your code, and from which you can easily distribute your code to a remote machine.
- Even if you are only working alone on your own laptop it is valuable to incorporate GitHub into how you work.

First step: get an account on GitHub

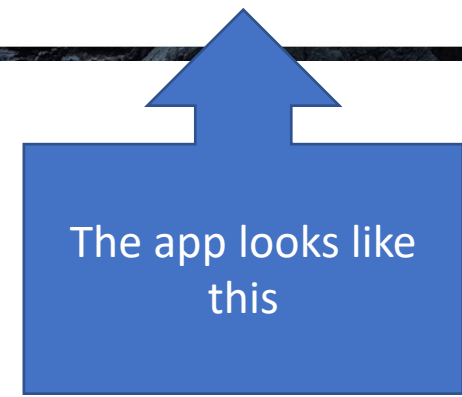
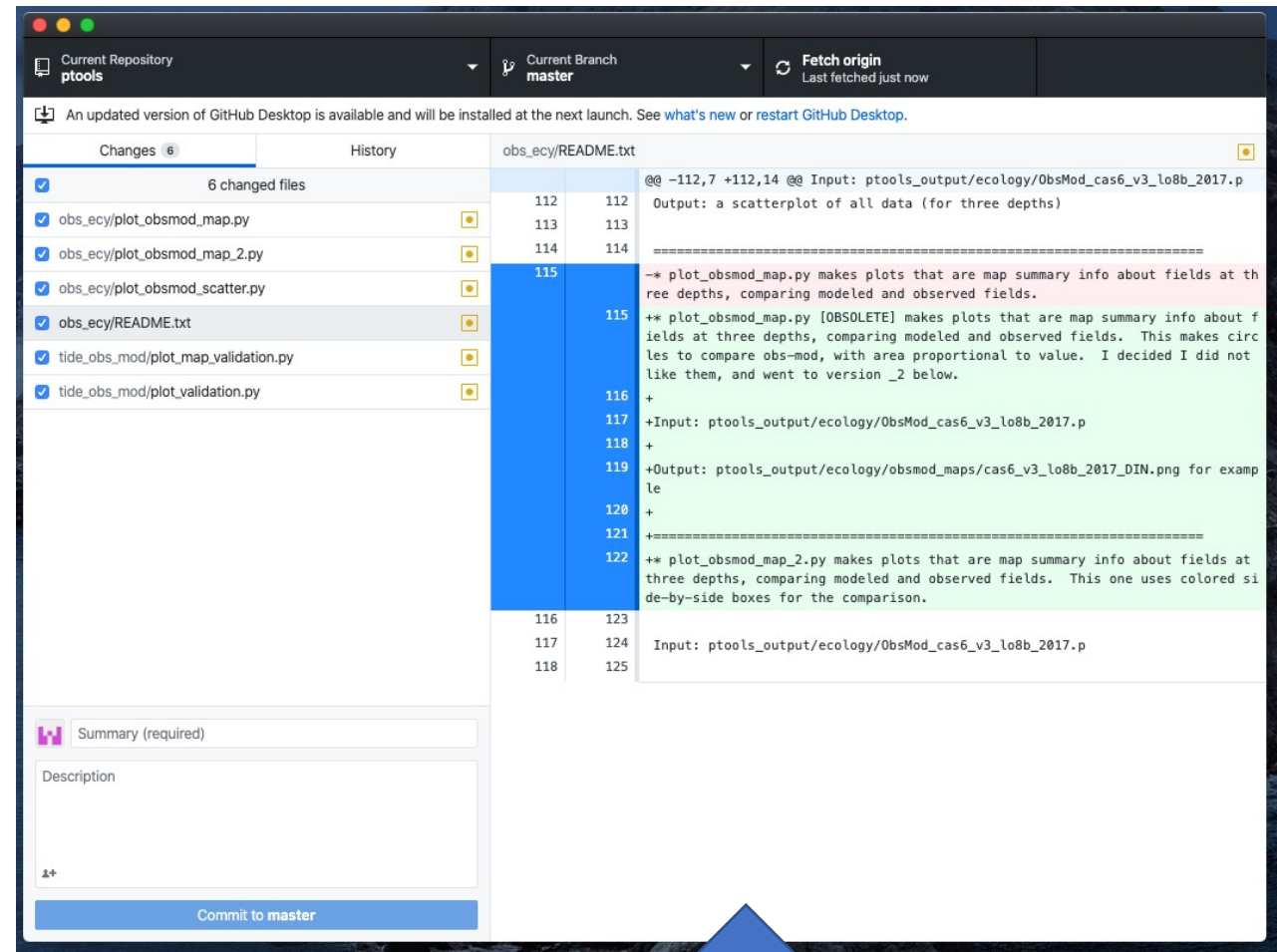
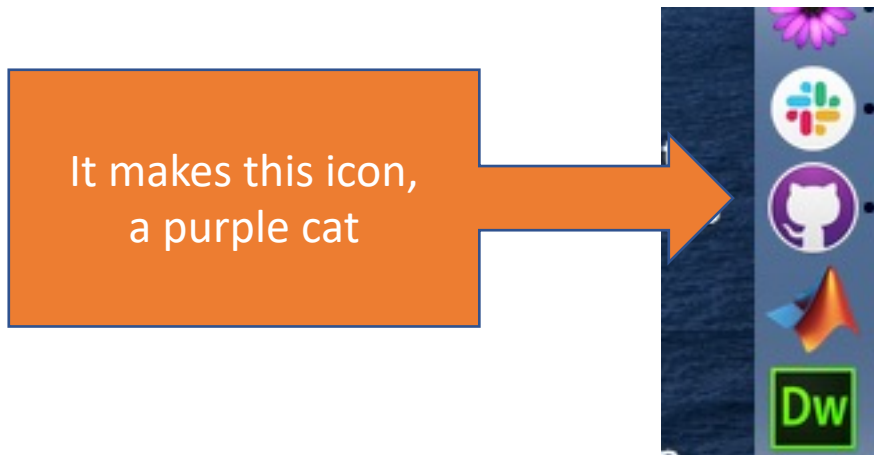
- Sign up for a GitHub account:
- <https://github.com/>
- You have to make up a username. I use "parkermac". You can add more information to your profile if you like.
- It's free!
- Note: by default the code on GitHub is publicly available.



The screenshot shows a web browser displaying the GitHub profile page for Parker MacCreedy (username: parkermac). The browser tabs include "parkermac (Parker MacCreedy)" and "Parker MacCreedy Homepage". The address bar shows "github.com/parkermac". The profile page features a profile picture of a man, a bio stating "I am a professor in the School of Oceanography at the University of Washington in Seattle.", and contact information including an email address "p.maccready@gmail.com" and a website "http://faculty.washington.edu/pmacc/". The profile is marked as "PRO". The right sidebar shows navigation tabs for "Overview", "Repositories", "Projects", "Packages", "Stars", "Followers", and "Following". Under "Popular repositories", several repositories are listed, including "LiveOcean" (MATLAB, 2 stars, 1 fork), "energy" (MATLAB), "alpha" (MATLAB), "shared" (MATLAB), "post_tools" (Python), and "pandora" (MATLAB). A "289 contributions in the last year" section displays a grid of green squares representing contributions from September to April. The "Contribution activity" section shows a timeline for April 2020.

Then get some software for your computer

- Download and install the "GitHub Desktop" software on your personal computer, available for both Mac and Windows.
- <https://desktop.github.com/>



Launch GitHub Desktop and...

- In GitHub Desktop -> Preferences, log into your GitHub account
- In GitHub Desktop also do "Install Command Line Tool..."
- (everything you do by clicking in the Desktop app you can also do from the command line, but for now we will stick to using the app).
- Now you have all the tools in place, but you still need to put a code project into Git...

Put your first code project into git

- Choose a folder that already has some code in it (OK for it to have subfolders - but it should all be code and text files, not data or output), or choose the name of a new folder that you will put code into later.
- **Launch the GitHub Desktop app on your computer**
- Do File -> New Repository...
- And you will get this box ==>

The screenshot shows the 'Create a New Repository' dialog box with the following fields and values:

- Name:** repository name
- Description:** (empty)
- Local Path:** /Users/pm7/Documents/Classes/2020 Effective C (with a 'Choose...' button)
- Initialize this repository with a README:**
- Git Ignore:** None
- License:** None

Buttons at the bottom: Cancel, Create Repository

1. Type the name of the existing folder, or the new folder you want to create

2. Add some words describing it

3. Use Choose... to navigate to where the folder is, or where you want it to be (the **parent** directory of the one where you project is). No need to initialize with a README.

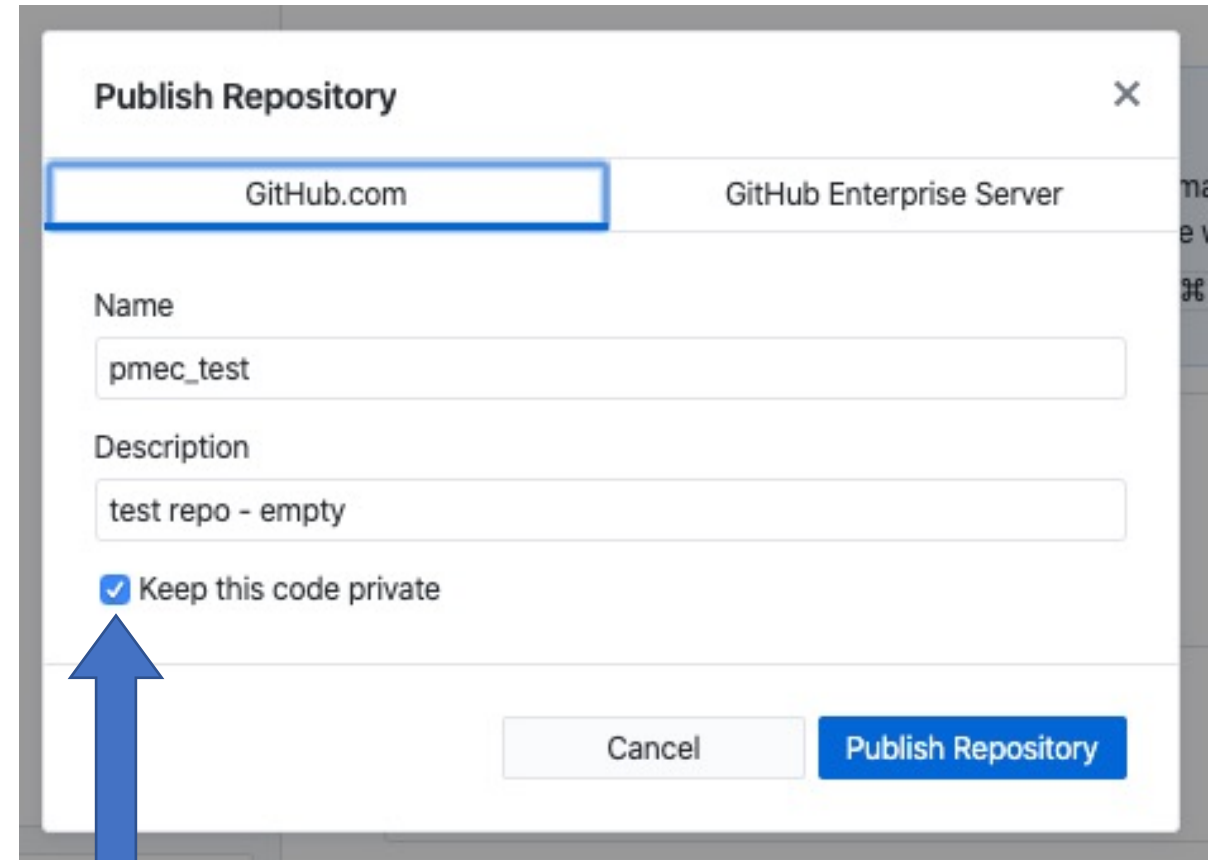
4. For Git Ignore choose: **Python**

5. For License choose: **MIT License**

6. Then click this!

public or private?

- When you make a new repo and publish it to GitHub you now can choose to make it public or private
- It used to be that you had to pay to make it private - that it no longer the case.
- I prefer to keep things public by default, except under circumstances where I really need them to be private.
- For this class I'd suggest you make your classwork repo public, so I can see it if needed, but the choice is up to you.



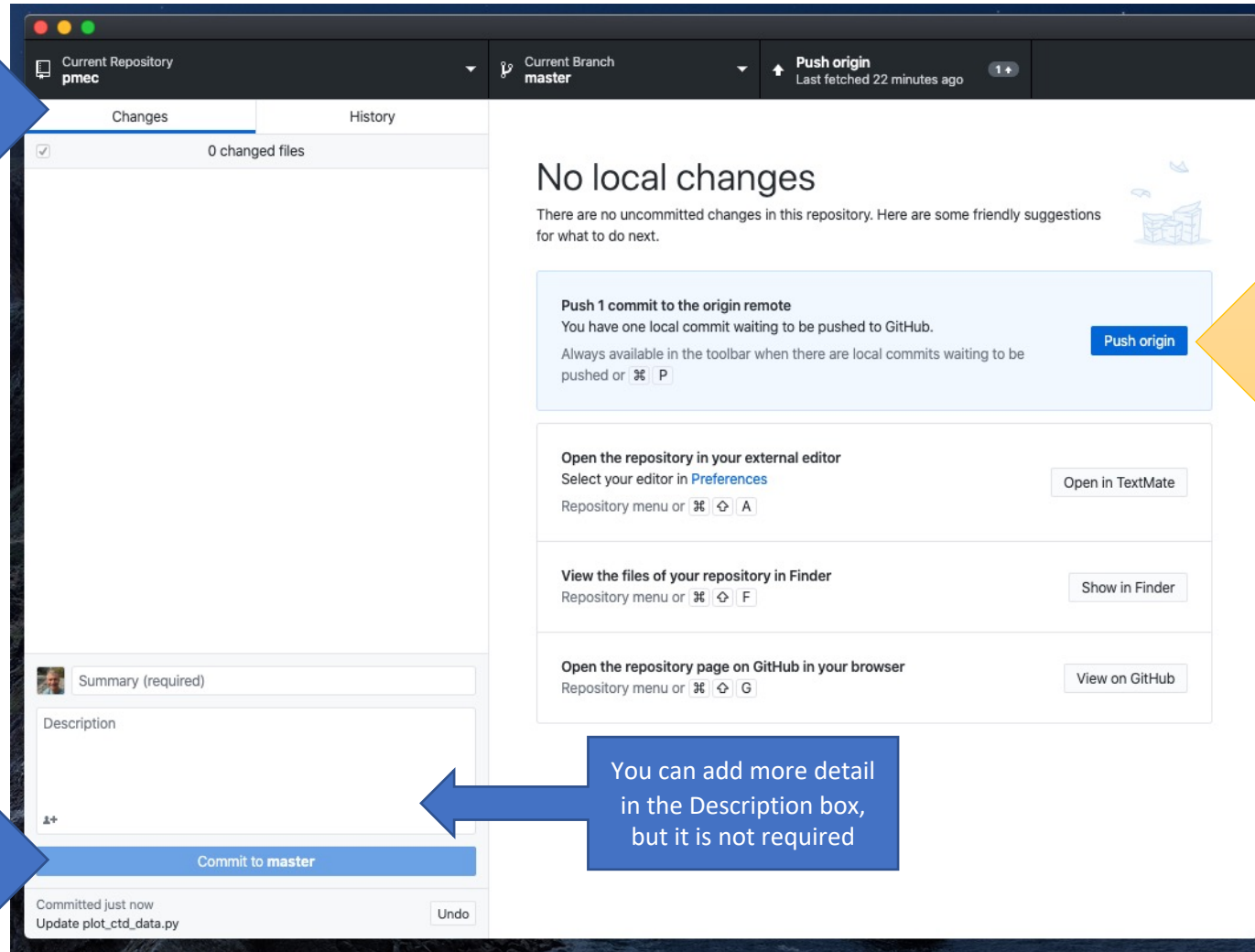
The screenshot shows the 'Publish Repository' dialog box in GitHub. At the top, there are two tabs: 'GitHub.com' (selected) and 'GitHub Enterprise Server'. Below the tabs, there are three input fields: 'Name' with the value 'pmech_test', 'Description' with the value 'test repo - empty', and a checked checkbox labeled 'Keep this code private'. At the bottom right, there are two buttons: 'Cancel' and 'Publish Repository'. A blue arrow points from the bottom of the dialog to the 'Keep this code private' checkbox, with a blue box containing the text 'Unclick?' at the base of the arrow.

What you did...

- You made Git on your computer know that you want it to keep track of the code project in this folder
- In that folder it created a hidden directory called **.git**. That is where it stores information on all the changes you make to code there. You never need to look in this.
- I did not suggest adding a README because it just adds an empty file in "Markdown" format. I'd prefer that you write your own README as a text file because it is simpler and more portable.
- You added a LICENSE text file that says anyone can use this code
- You added a hidden text file called **.gitignore** that has a long list of file types for which Git will not keep track of changes. One of these would be the ".pyc" files that Python automatically creates for any module you write - it is a compiled version of that module.
- If you had any code in the folder to begin with it also automatically "committed" them with the message "Initial commit"

Next you want to push this repository to GitHub in the cloud, using the GitHub Desktop app on your laptop

Make sure you are on the "Changes" tab (the History tab shows past changes)

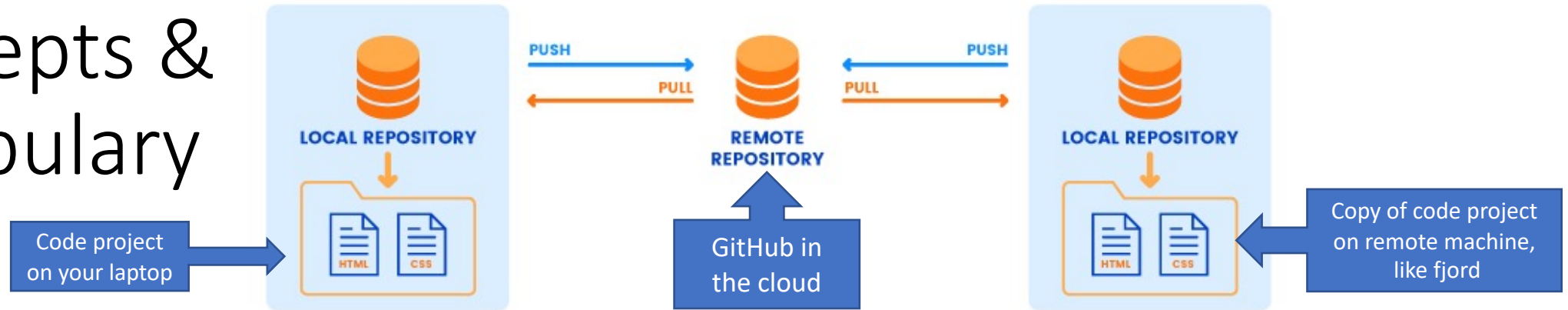


Then click on the "Push origin" button to send a copy of your code project to the GitHub cloud

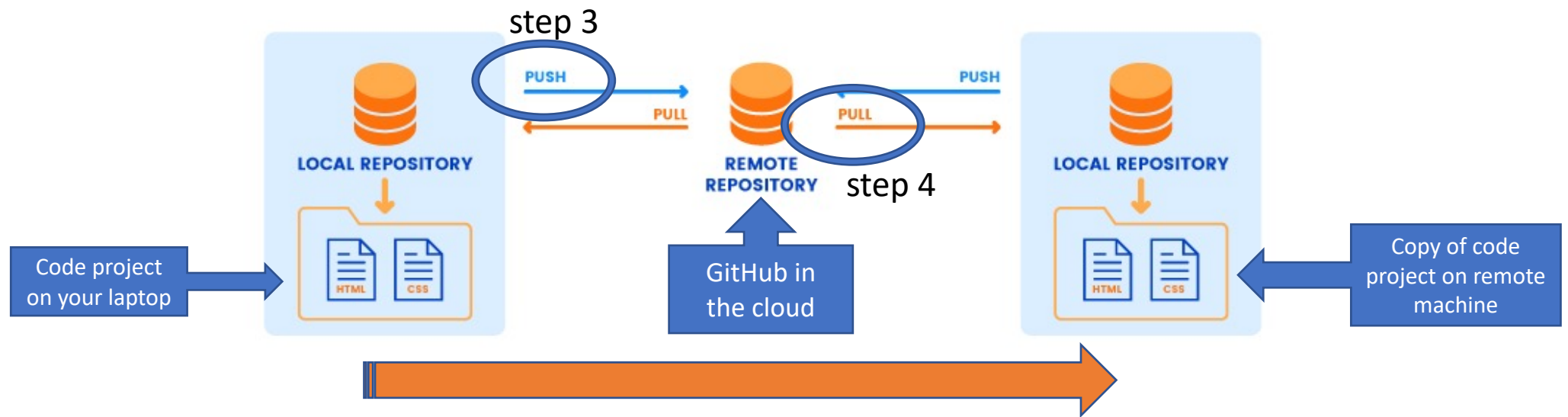
If you need to save any changes locally, you type in a Summary and then commit

You can add more detail in the Description box, but it is not required

Concepts & Vocabulary



- Your code project is just a folder (and any subfolders) with code and other text files.
- When you tell git to make this code project a "Repository" then git adds a hidden folder ".git" to your folder where it keeps a copy of your files and a history of all the changes you make. That is the "Local Repository" in the figure above.
- "repo" is the slang for Repository. Use it in casual conversation with friends so they will know you are cool.
- Every time you change a file in the folder (add, delete, rename, or edit) git will keep track, and then when you "commit" the changes they will be part of your local repository.
- Then you "push" your local repository to the cloud: your GitHub account. Mysteriously this remote repo is always referred to as "origin", even though the code actually originated from your laptop.
- Finally you can "clone" the remote repository to any other machine (like fjord) and it will appear as a folder with your code in it. If you want to update the code on the remote machine you just "pull" it from the remote repo by using the command: "git pull". You do this using the linux terminal and you have to have navigated to be inside the folder containing the code project.
- We'll go over the details of the clone step in a few slides.



- **A very simple "one-way" workflow consists of:**

1. edit code on your laptop and save the changes
2. commit the changes using GitHub Desktop
3. push the changes to the remote repo using the "Push origin" button in GitHub Desktop
4. on the remote machine update the code by using "git pull" from the command line
5. now you can run the code on the remote machine, confident that it is exactly the same as on your laptop. Of course for this to work your code has to be written to work on the remote machine.

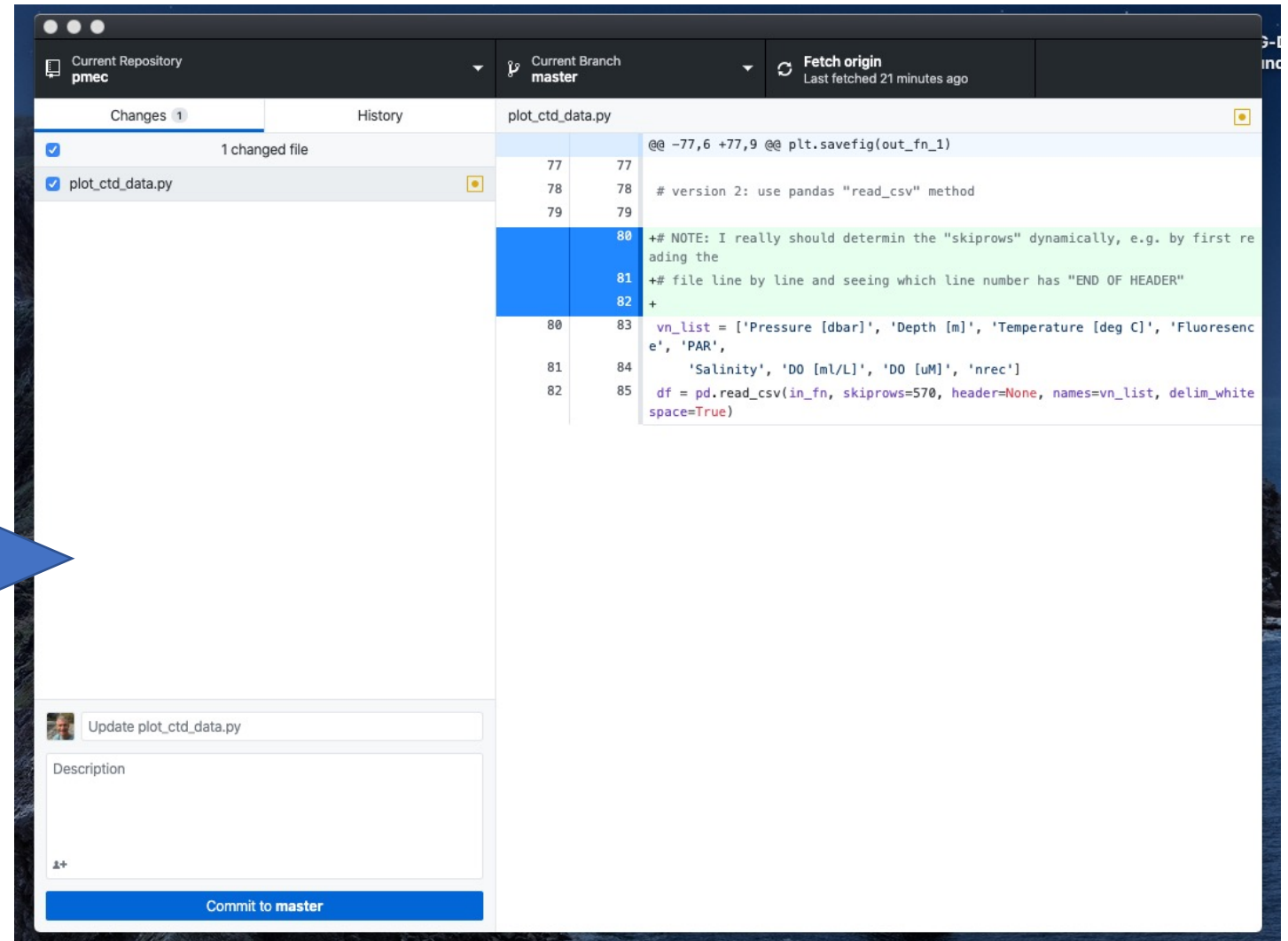
Every time you change a file in the folder (add, delete, rename, or edit) git will keep track

Here is what the app looks like after I made (and saved) an edit to one of my programs in the repo.

Note that in the right panel it shows a focus of exactly what lines changed.

If you feel you are done with making changes in your current editing session (maybe at the end of the day) you:

1. *add a summary of the changes*
2. *Click on "Commit to master"*
3. *"Click on Push origin"*



More git lingo: "master"

- The term "master" refers to the "branch" of the repo you are working on. In GitHub you can make other branches, e.g. to test out some new code while you still want the old code to be functional. We will not be using branches in this class, but you may find them useful sometime.
- For our purposes, you can just think of "master" as meaning *"the version of my code project that has the most recent changes committed"*.

cloning

- In order for "git pull" to work on the **remote machine** you first have to "clone" the repo to that location.
- Go to your github.com account and open your repo there

The screenshot shows the GitHub repository page for 'parkermac/pmec'. The repository is titled 'Code for the Effective Computing 2020 class'. It has 4 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. The current branch is 'master'. A 'Clone or download' button is visible in the top right of the repository area. Below the repository information, a list of files is shown, including 'shared', '.gitattributes', '.gitignore', 'first_python_program.py', 'first_script.sh', 'plot_ctd_data.py', 'test0.py', 'test_input_output.py', and 'test_my_module.py'. The 'Clone or download' button is highlighted with a red box.

Click on the Clone or download button

The close-up shows the 'Clone or download' dropdown menu. The 'Clone with HTTPS' option is selected, and the URL 'https://github.com/parkermac/pmec.git' is displayed. A clipboard icon is visible next to the URL. The 'Open in Desktop' and 'Download ZIP' options are also visible.

Then copy the link it makes by clicking on the clipboard icon

Cloning - last step

- Finally, logon to your remote machine (fjord) and cd to where you want the cloned repo to end up - e.g. /data1/effcom/[username]/
- and then type:
- `git clone [paste in the URL you copied]`
- and your directory will appear, full of your code!
- If you make new changes on your laptop, commit and push them, then all you have to do on fjord the next time is type:
- `git pull`
- from **inside the directory you made**, and then code will be updated to the most recent master version.
- Note: if you created your repo as **private**, then you will likely have to issue the command:
 - `unset SSH_ASKPASS`
 - before doing `git clone [...]`. Then it will ask for your GitHub password.
 - You can add this as a line in your `.bashrc` on fjord.

My own cloning screen shots, on fjord:

```
pm7 — parker@fjord:/data1/parker — ssh parker@fjord.ocean.washington.edu — 110x36
Last login: Sun Apr 26 08:07:08 on ttys001

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Parker-MacBook-Pro:~ pm7$ fjo
parker@fjord.ocean.washington.edu's password:
Last login: Fri Apr 24 13:00:11 2020 from c-73-221-128-168.hsd1.wa.comcast.net
[parker@fjord ~]$ cdpm
[parker@fjord parker]$ ls
LiveOcean  LiveOcean_output  ptools_data  tools  wcofs_data
LiveOcean_data  ptools  ptools_output  tools_output
[parker@fjord parker]$ git clone https://github.com/parkermac/pmec.git
```

Cloning

Note: if you put the repo in the wrong place, just delete it and start again. Git won't mind.

```
[parker@fjord parker]$ git clone https://github.com/parkermac/pmec.git
Initialized empty Git repository in /data1/parker/pmec/.git/
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 23 (delta 4), reused 22 (delta 3), pack-reused 0
Unpacking objects: 100% (23/23), done.
[parker@fjord parker]$ ls
LiveOcean  LiveOcean_output  ptools  ptools_output  tools_output
LiveOcean_data  pmec  ptools_data  tools  wcofs_data
[parker@fjord parker]$
```

Now the directory "pmec" exists

When I try to use "git pull" from inside pmec it tells me it is already up to date.

```
[parker@fjord parker]$ ls -la pmec
total 44
drwxr-xr-x. 4 parker parker 4096 Apr 26 12:15 .
drwxr-xr-x. 13 parker parker 4096 Apr 26 12:15 ..
-rw-rw-r--. 1 parker parker 560 Apr 26 12:15 first_python_program.py
-rwxrwxr-x. 1 parker parker 1473 Apr 26 12:15 first_script.sh
drwxrwxr-x. 8 parker parker 4096 Apr 26 12:15 .git
-rw-rw-r--. 1 parker parker 66 Apr 26 12:15 .gitattributes
-rw-rw-r--. 1 parker parker 16 Apr 26 12:15 .gitignore
-rw-rw-r--. 1 parker parker 2897 Apr 26 12:15 plot_ctd_data.py
drwxrwxr-x. 2 parker parker 25 Apr 26 12:15 shared
-rw-rw-r--. 1 parker parker 301 Apr 26 12:15 test0.py
-rw-rw-r--. 1 parker parker 1927 Apr 26 12:15 test_input_output.py
-rw-rw-r--. 1 parker parker 798 Apr 26 12:15 test_my_module.py
[parker@fjord parker]$
```

and pmec has my code in it

```
[parker@fjord parker]$ cd pmec
[parker@fjord pmec]$
[parker@fjord pmec]$ git pull
Already up-to-date.
[parker@fjord pmec]$
```


Resources

- This one is the best Git tutorial I have found, although it does everything from the command line. Nonetheless, very clear on the concepts:
 - <https://rubygarage.org/blog/most-basic-git-commands-with-examples>
- General advice in installing Git anywhere (e.g. in linux):
 - <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
- Another tutorial from Software Carpentry
 - <http://swcarpentry.github.io/git-novice/>
- And some thoughts about **collaborating** using GitHub (which we will get to in the second Git lecture):
 - <https://uoftcoders.github.io/studyGroup/lessons/git/collaboration/lesson/>