Giting Started with GitHub

Last Edited Jan 20, 2022

**Introduction**

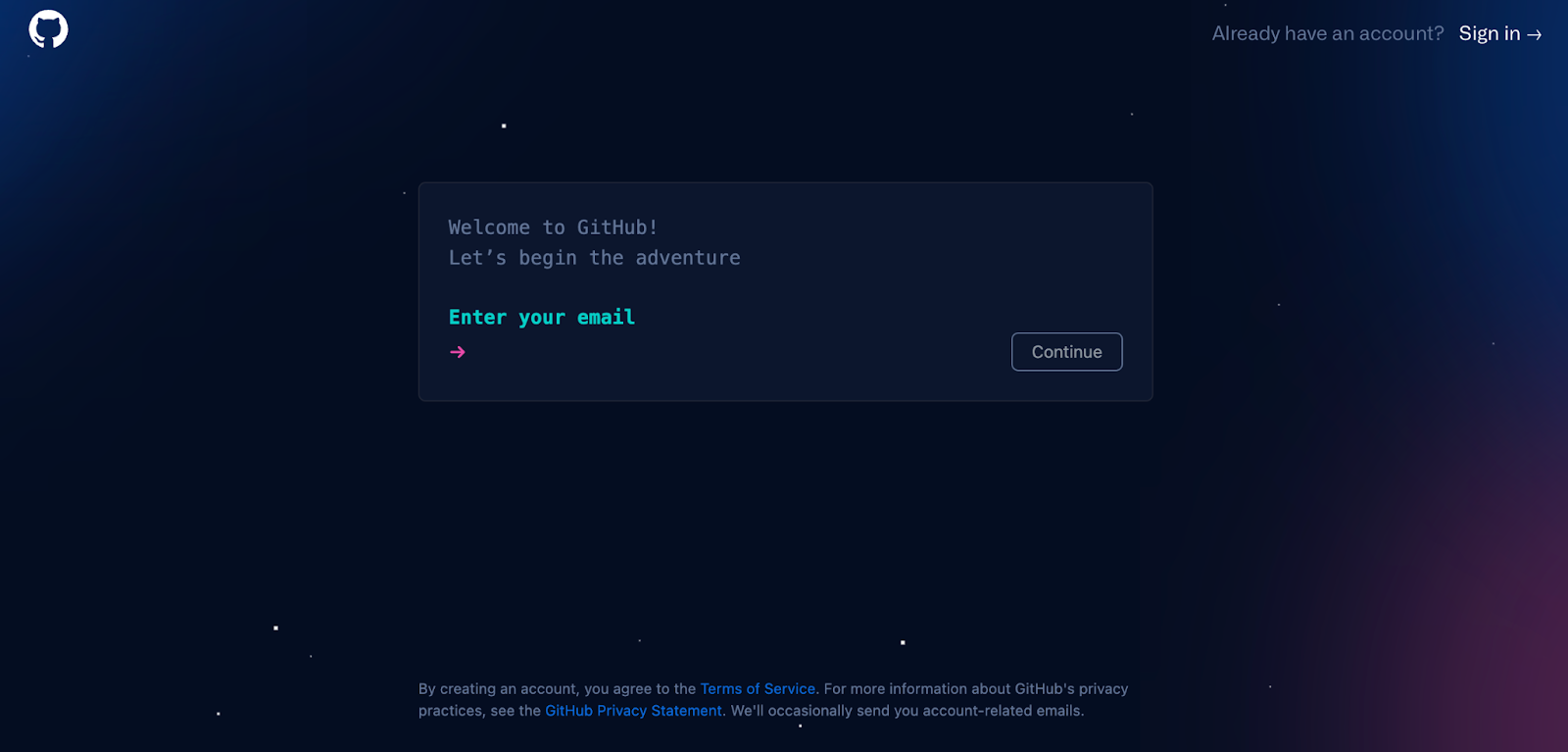
Interacting with the Erdős Institute’s educational materials requires the ability to get that content onto your computer. We distribute our materials through code repositories stored on our organizational GitHub account, <https://github.com/TheErdosInstitute>. In this written tutorial we will cover the basics of GitHub and git version control. If you are familiar with these, feel free to skip this tutorial.

*Note: If you already have a GitHub account and are familiar with git version control software you can skip this tutorial.*

**Getting a GitHub Account**

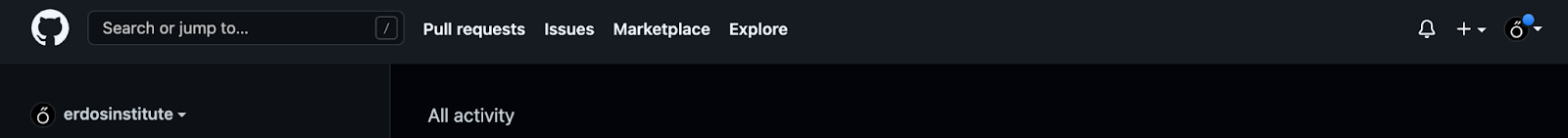
Since we host our repositories on the GitHub website you will need to get a GitHub account, you can do so by clicking on this link: <https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home>.

You should see something like the following:



Follow the on screen instructions and work your way through GitHub’s sign-up process.

Once you have an account and are signed in you should see something like the following:



With your username and a generic profile picture in place of *erdosinstitute* and the double acute ő icon.

**Repositories**

GitHub hosts *repositories,* which are *nice* folders where you can store files, typically related to a project involving some kind of code or data. Here *nice* refers to the repository’s *version control* features.

For those of you unfamiliar with *version control*, we can compare it to the process of writing a draft document. In the past while writing a paper on a text editor, you may have created multiple versions of the document as you worked your way through the editing process. For example, paper1.doc, paper2.doc, paper3.doc, paper\_final\_edits.doc, paper\_final\_edits\_typo\_fixed.doc, paper\_final\_edits\_typo\_fixed\_final\_version\_for\_real.doc, etc.. Such a process can get quite messy, and that’s just with a single person controlling the file names. Now imagine you have a team of people all editing the same file in different spots all saving their own version which is slightly different from the version you’re editing. Version control software aids in this process by tracking every change you make to the files within the repository, while making sure that you are able to recover any older version of the file that you had saved previously.

The standard open-source version control software is known as *git*, <https://git-scm.com/>. In this tutorial we will be covering the bare minimum of what you will need for our content. This includes:

* *Cloning* an existing repository,
* *Pulling* updates,
* Creating a new repository,
* Making commits to update your own repository and
* Pushing the changes for everyone to see.

Please note that this means we will not be touching on other git topics like:

* Branches,
* Merging,
* Forks and
* More.

These are concepts important to working with git in teams, but are beyond the scope of this tutorial.

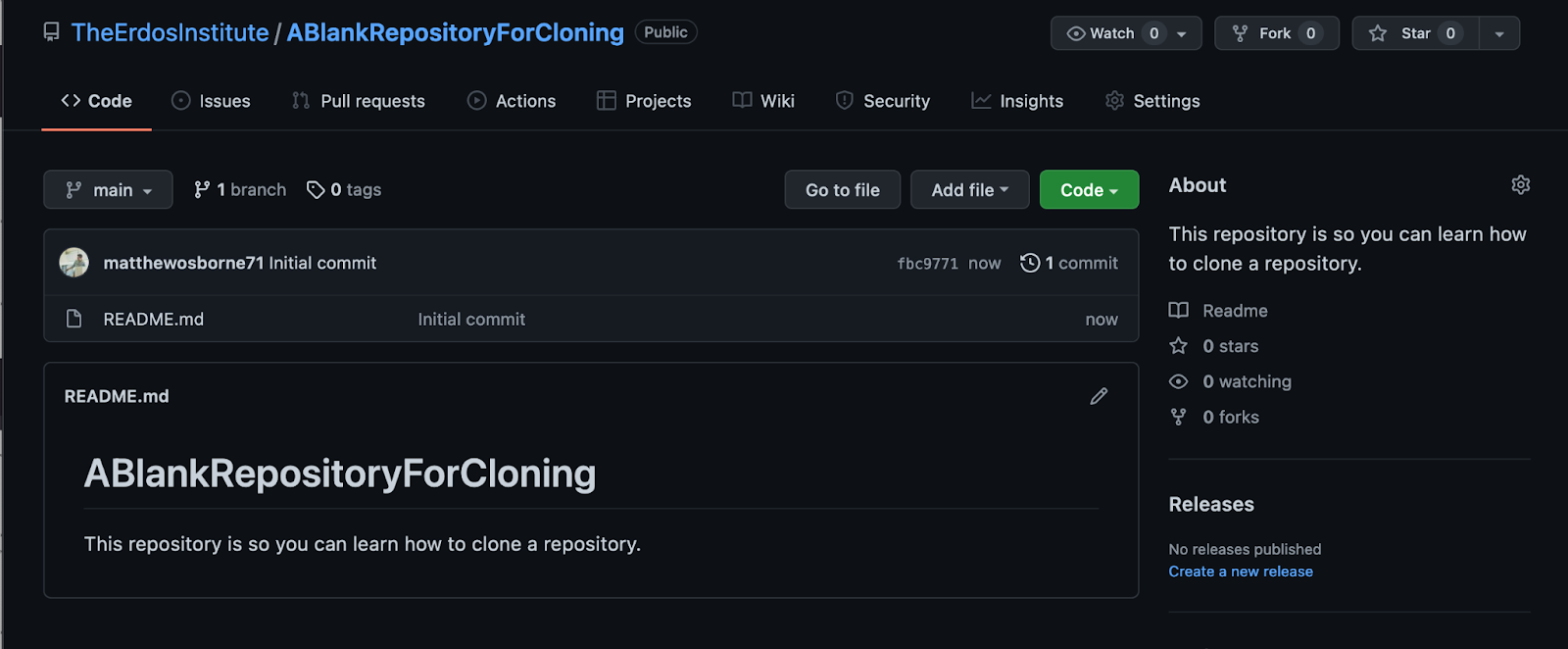
Command Prompt/Terminal vs GitHub Desktop

The following git tasks can be accomplished by using git directly in the command prompt (Windows machines)/terminal (Mac OS and Linux) or by using the GitHub desktop app. If you are comfortable typing commands into the command prompt/terminal I suggest you use the git approach. To use git directly in the command prompt/terminal follow the installation instructions found here, <https://git-scm.com/downloads>, assuming you have not already installed git. If you would prefer to have a user interface, I suggest using the GitHub desktop app, which can be installed by following the directions here, <https://desktop.github.com/>.

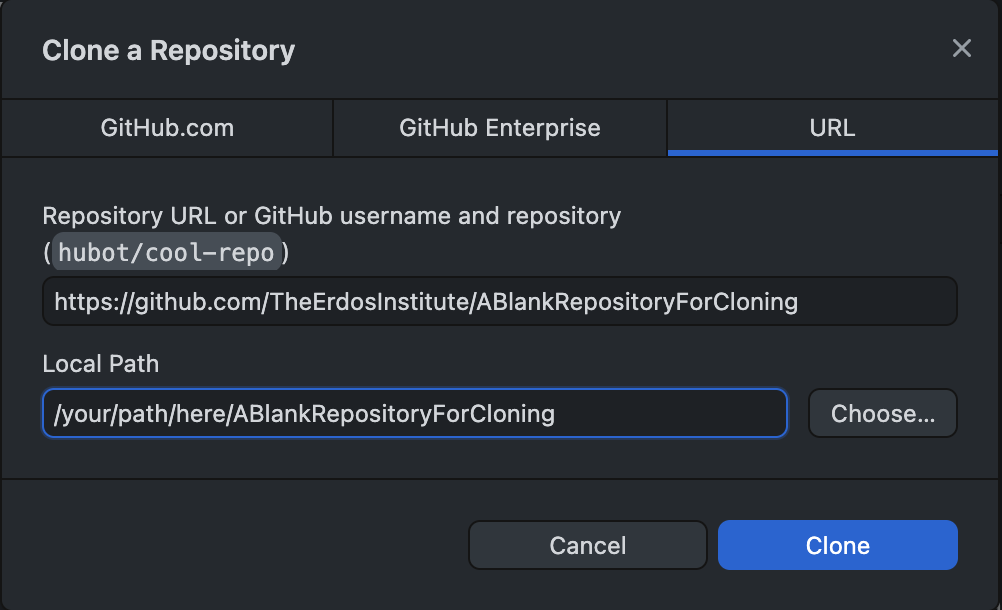
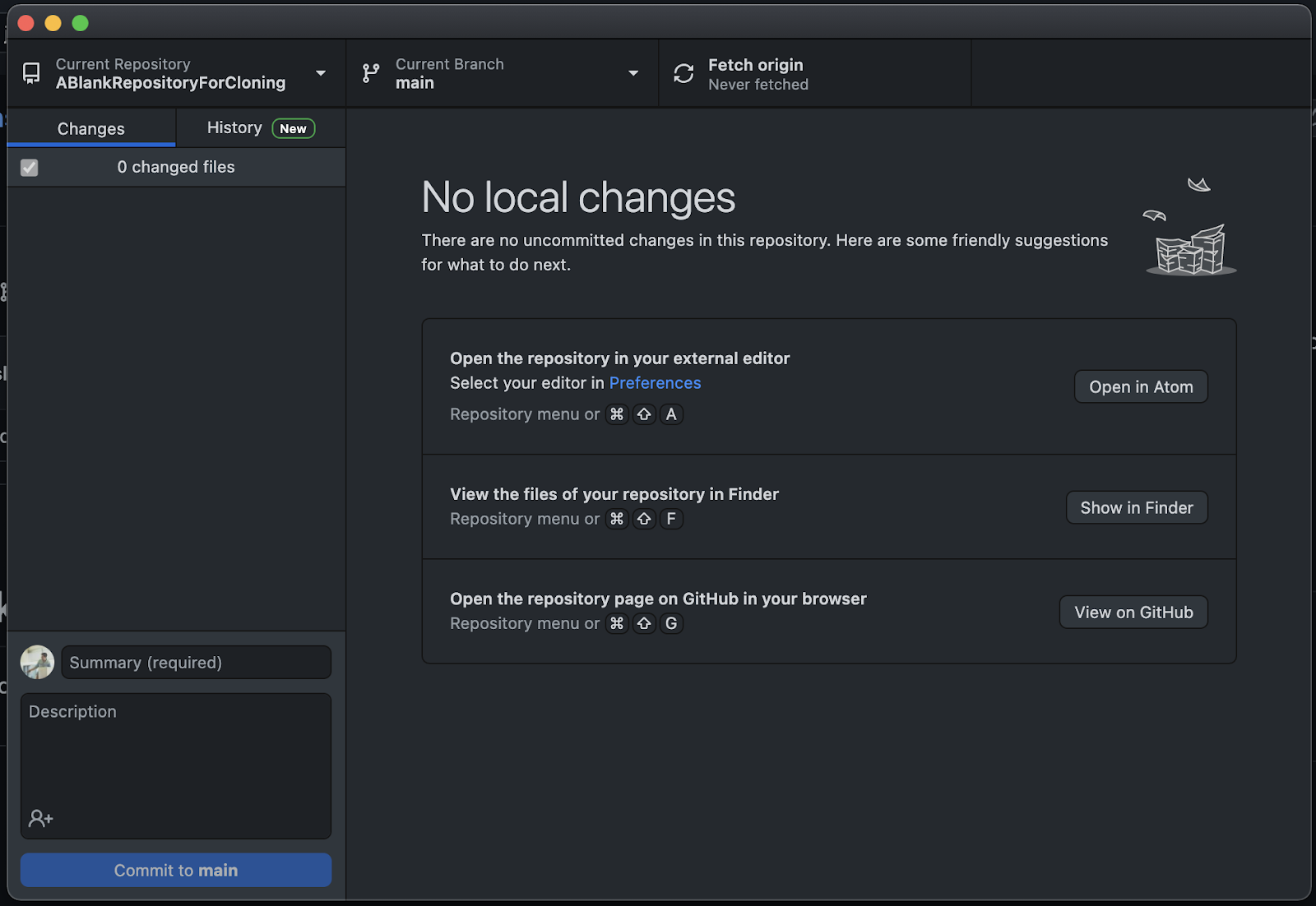
*Cloning an Existing Repository*

The first thing we will do is *clone* an existing repository. *Cloning* is the process of copying a repository onto your computer in a way that allows you to use git in the future to update the repository when changes are made to it.

We will clone the following repository, <https://github.com/TheErdosInstitute/ABlankRepositoryForCloning>.



GitHub Desktop Instructions

1. Open your GitHub desktop application
2. Copy the URL for the repository (i.e. <https://github.com/TheErdosInstitute/ABlankRepositoryForCloning>)
3. Click the “Clone a Repository from the Internet…” button (see image below)
4. Paste the repository’s URL in the “Repository URL” spot
5. Type the path of the location where you’d like to store the repository, this should be a folder in your computer where you are comfortable keeping the repository.
6. You should now see something like the following

Congratulations, you have successfully cloned a repository onto your laptop with the GitHub desktop app. You should now have a folder labeled “ABlankRepositoryForCloning” at the desired location in your computer.

Terminal/Command Prompt Instructions

1. Open a new terminal or command prompt window.
2. Change the current directory to the directory where you want to clone the repository.
3. Click the green “Code” button on the repository’s webpage.
4. Execute the command git clone repository\_url, where you should replace the text repository\_url with what is copied when you click the copy button highlighted with a yellow box in the image below.
5. If you are prompted to log into your GitHub account in the terminal do so by following the prompts in your command line.

Congratulations, you have successfully cloned a repository onto your laptop with the GitHub desktop app. You should now have a folder labeled “ABlankRepositoryForCloning” at the desired location in your computer.

*Pulling an Update to a Repository*

Sometimes the repository you cloned will have been updated and you will want to update your local copy of the repository to match the original. The process of updating your local copy is known as *pulling* updates.

*Note:* When you pull an update from the original repository any changes to files that already existed in the older version of the repository will automatically be made to your local copy (the one stored on your computer). If you want to keep an unaltered version of a specific file, you may want to create a hard copy of the repository in your computer. For example, if there was a file in “ABlankRepositoryForCloning” called “FileILove” that you didn’t want to change you could create a new file “FileILove\_Copy” that is a copy of “FileILove” prior to pulling the updates. You could also copy the entire repository on your computer so you would have both “ABlankRepositoryForCloning” folder and a “ABlankRepository\_Copy” folder on your computer prior to pulling the update. For those of you thinking there has to be another way to do this using git, you may be interested in learning more about forks and branches.

GitHub Desktop Instructions

When there is an update to pull you should receive a notification on your app, that looks like so:

Click the “Pull origin” button.

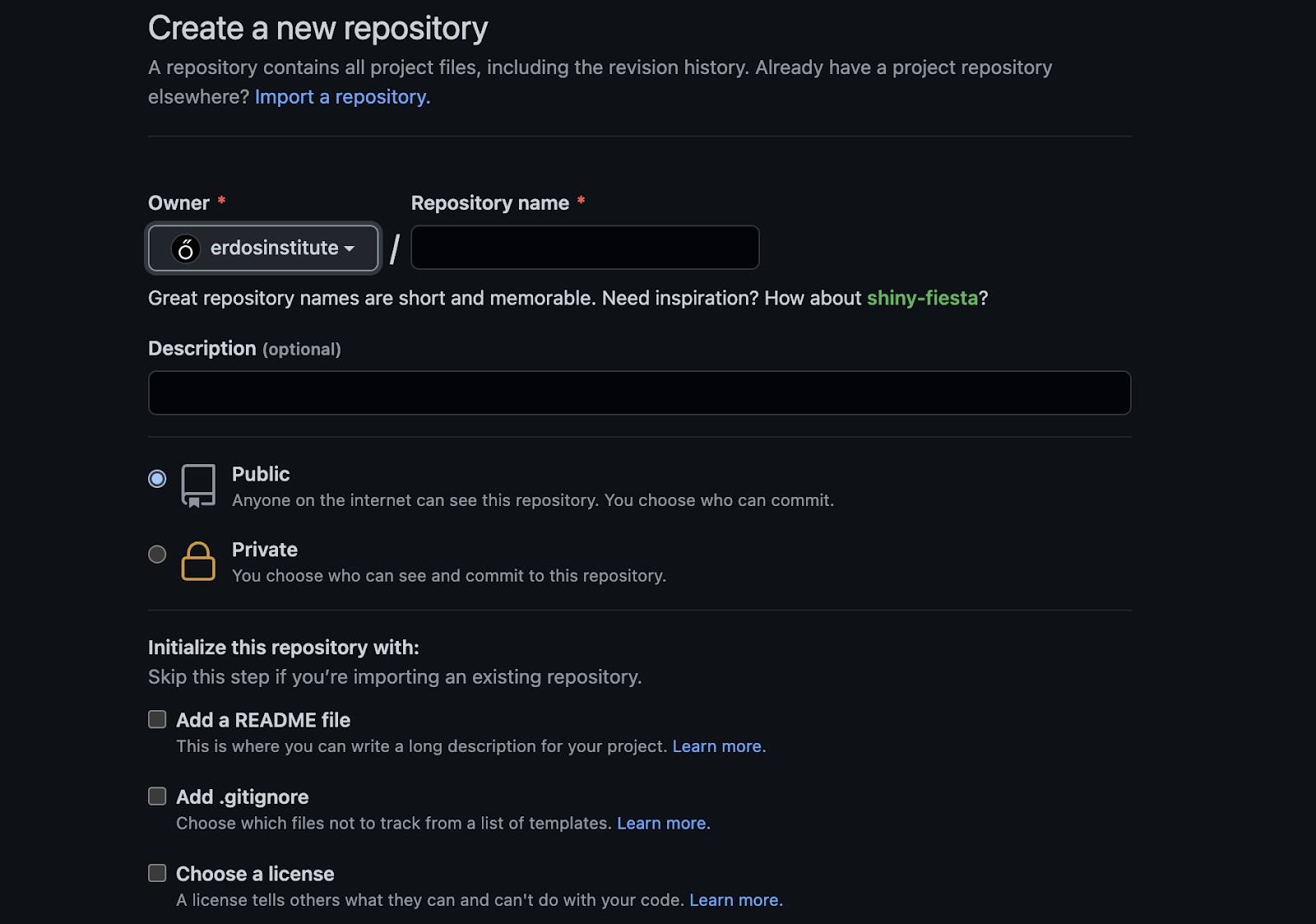
Terminal/Command Prompt Instructions

1. Open a terminal window or command prompt
2. Change your directory to your local copy of the repository
3. Execute the command git pull origin main

*Making a GitHub Repository*

At some point you will want to create your own repository on GitHub. Doing so is straightforward.

Instructions

1. Go to <https://github.com/> and sign into your account
2. Click the “New” button to the right of the “Repositories” section of your login page (As of January 2022 this is on the left hand side of the webpage)
3. You should now see something like this:
   1. Choose a repository name
   2. Give a short description if you want to
   3. Choose whether you want your repository to be :
      1. Public - Anyone can see and clone it
      2. Private - Only you and your collaborators can interact with it
   4. Add a README file (optional but strongly recommended)
      1. A README is a file providing some amount of documentation for your repository. When someone goes to your repository’s URL they will see this file displayed along with the files contained within the repository. *It is good practice to have a nicely written README file*.
   5. Add a .gitignore file (optional but recommended, note the leading dot)
      1. A .gitignore file is a git system file that tells git what files to ignore when you *push* updates to the GitHub servers from your local copy of the repository. It is a simple text file where each line corresponds to a file that you **do not** want added to the GitHub servers. For example, if you had a .gitignore file with grocery\_list.txt in it, the file grocery\_list.txt would not be added in any updates. Files storing sensitive data such as passwords or very large files produced as output of code are common types of files you might want to ignore.
   6. Add a licence file (optional)
      1. A license tells others what they can and can't do with your code. You may or may not find this necessary depending on the specific project. If you think you might want a licence file, GitHub likely provides a generic license template that will work for your project.
   7. When you are ready click the “Create Repository” button at the bottom

*Pushing Updates to your Repository*

After you have made some edits to your local repository’s files or maybe even created new files in your local repository, you could be ready to make these changes in the GitHub version of the repository as well. The process of making these changes with git is known as *pushing* an update.

The first step is to add the file to staging, then commit the changes to the local copy of your repo. This tells git that you want the changes you’ve made locally when saving the file to be part of the repository itself. Finally, changes are pushed up to the GitHub copy of the repository.

GitHub Desktop Instructions

1. In the “Changes” panel select the changes you want to push by clicking the relevant checkboxes.
2. You then have to *commit* the changes with a small message describing what has changed. This message should be short but still descriptive enough that someone reading it can understand what the update was. Type your message in the “Description” box then click the “Commit to…” button
3. When you are ready to *push* the updates click the “Push …” button.

Terminal/Command Prompt Instructions

1. Open your command prompt or terminal.
2. Set the present working directory as the repository.
3. For each file you want to change, run the command git add file\_name
4. For each file you want to change, execute the command git commit -m “Type a short message”, where you should replace file\_name with the relevant file’s name and type a short but descriptive message of the update in place of “Type a short message”.
5. When you have made all of your *commits* execute the command git push.

**Conclusion**

This tutorial has provided a brief introduction to git version control software as well as the git repository hosting site GitHub. *For the purposes of the Erdős Institute data science boot camps you will mostly need to clone a repository and pull updates as boot camp materials are added.* However, when working on a group project or as a part of a data science team you may want to know more about git. Additional resources on *branches, forks and merging* can be found in the Additional Resources section below.

**Additional Resources**

* The git documentation, <https://git-scm.com/doc>
* <https://coderefinery.github.io/git-intro/>
* <https://www.atlassian.com/git>

*This tutorial was written for the 2022 Erdős Institute Cőde Data Science Boot Camp by Matthew Osborne, Ph. D.*