GitHub reference

This document provides a basic introduction and reference to the utilization of GitHub in an academic environment for collaboration on programming projects and/or providing course document to students in a course.

US Coast Guard Academy 2020

Table of Contents

[2 Introduction 2](#_Toc31959484)

[2.1 What is GitHub 2](#_Toc31959485)

[2.2 Setup 2](#_Toc31959486)

[2.2.1 Creating Your Repository 2](#_Toc31959487)

[2.2.2 Software 4](#_Toc31959488)

[3 Utilization of GitHub in the Classroom 8](#_Toc31959489)

[3.1 Sharing Documents With Peers and Students 8](#_Toc31959490)

[3.1.1 Putting Documents on the Repository 8](#_Toc31959491)

[3.1.2 Cloning Repositories (Students) 10](#_Toc31959492)

[3.1.3 Updating Documents 10](#_Toc31959493)

[3.1.4 What is Forking? 10](#_Toc31959494)

[3.2 Projects 11](#_Toc31959495)

[3.2.1 Structure 11](#_Toc31959496)

[3.2.2 Collaborators 12](#_Toc31959497)

[3.2.3 Committing Changes 13](#_Toc31959498)

[4 Project Management 14](#_Toc31959499)

# Introduction

## What is GitHub

GitHub is software that has been developed for primarily programmers. Whenever more than one programmer must collaborate on a project, there has always been the issue of sharing code from a project with others. GitHub allows for others to replicate a programming environment while maintaining a certain level of integrity to the layout of the paths and the current state of the project. If using the software, you can keep the most up-to-date replica of the files stored on the GitHub server for your repository. Any changes on GitHub will reflect on your filesystem. Likewise, any changes made on your filesystem will reflect on GitHub. Both of these require pushing and pulling to the origin which will be covered later in this guide.

What does this mean for us as programmers? Let’s take the example of having two programmers working on a project together. Whenever an update to code is made by one programmer, there is an issue of getting the updated code to his or her partner. They could maintain a string of emails; but, this quickly gets cluttered and disorganized. Using GitHub, when an update is made to the code, the programmer could commit his changes to the GitHub project and the other programmer could pull this change from the project origin and have a copy of the most recent code directly to his or her system.

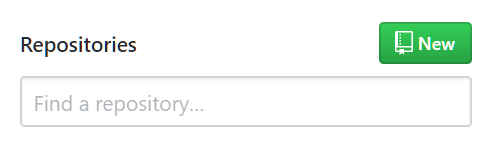
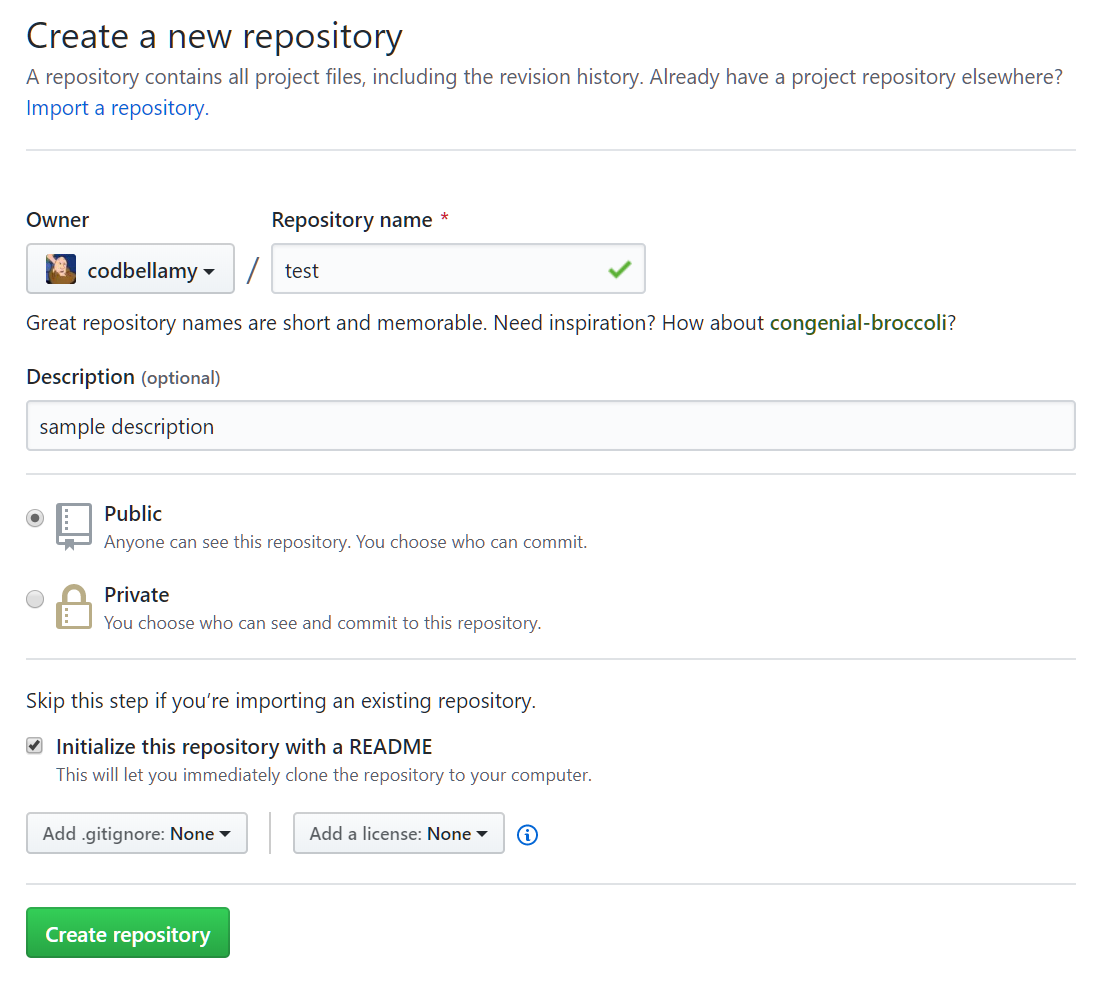
In an academic sense, this could be applied to a classroom setting. A professor could maintain administrative files in one folder and course materials such as code and power point presentations in another folder. If a student wanted a copy of these materials, the student could simply fork the branch to their own filesystem and have an exact copy of the entire project. This includes code, administrative documents, power points, and due dates.

## Setup

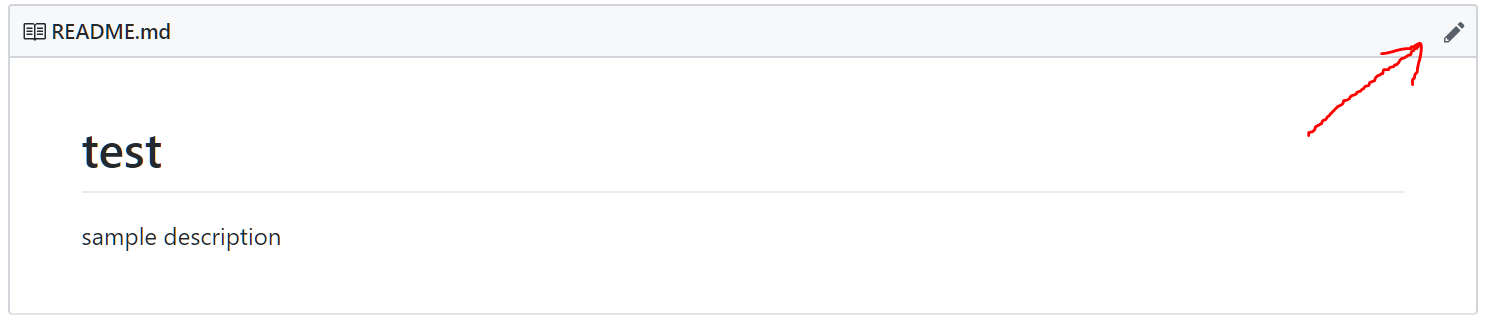
The following steps are to help you set up GitHub on your system. Although downloading the GitHub software is not necessary, it is highly recommended. Prior to completing any of these steps, it is mandatory for project collaborators to maintain an account. Standard users that will be cloning the project on their filesystems are not required to create an account. To create an account, go to <https://github.com/>.

### Creating Your Repository

Project collaborators have the highest authority when it comes to managing the project or course files. Like D2L, only teachers and TAs should be added as project collaborators.

1. Sign into your account
2. At the top left of the screen by the “Repositories” header, click “New.” 
3. Create a repository name, enter a basic description of the project, and select “Initialize this repository with a README” 
4. Select “Create repository”

That’s it! You’ve successfully created your repository in GitHub. The README file is a text file that typically contains information about the project/course. You can edit this by clicking the pencil icon on the top-right of the dedicated README box.

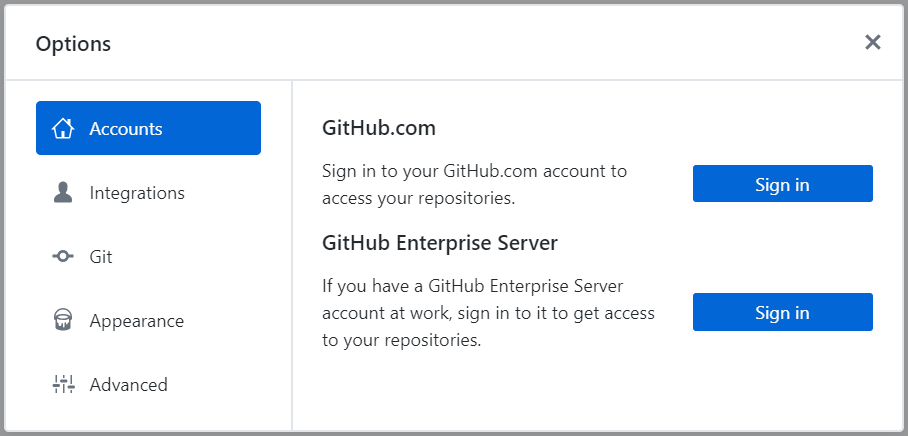
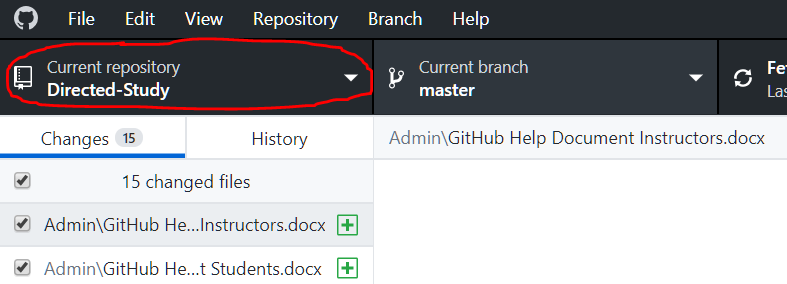


### Software

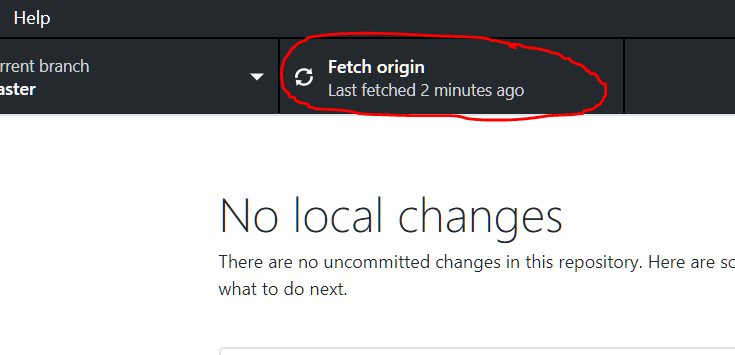
There is only one program that makes accessing, updating, and cloning projects marginally easier. This software is called “GitHub Desktop.” It is important to note that most of the tasks that can be done on the software can be done through GitHub’s website. The software can be used to keep an exact replica of the server’s files on your system. The software can be downloaded at <https://desktop.github.com/>. After installing GitHub Desktop, sign into your account if you are a project collaborator. All your project permissions will be translated to the software automatically. More in-depth tutorials on how to use the software will be covered in later sections.

#### Connecting GitHub to Your Repository

Connecting GitHub to your repository is very simple. Assuming you have created your repository as mentioned in 2.2.1, you can connect to your repository in a few simple steps.

1. Open GitHub Desktop
2. Go to File > Options > Accounts
3. Under “GitHub.com,” click the first “Sign In” option 
4. Enter your credentials and sign in
5. After being redirected to the main program interface, click the dropdown menu at the top-left to select a repository
6. Click “Add” > “Clone Repository”
7. Select the desired repository and click “Clone”

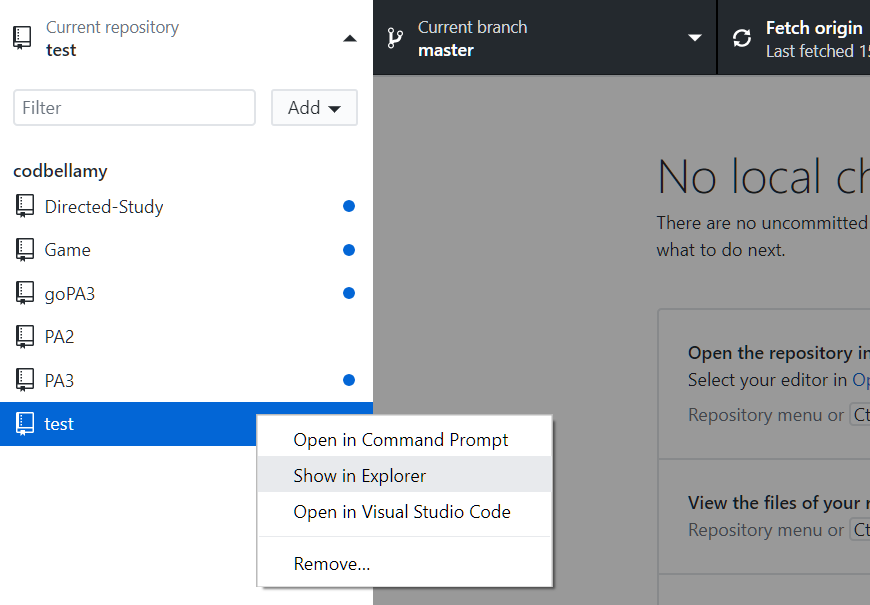
Since you are already signed in, a list of your current, active repositories with GitHub will be listed under the “GitHub.com” tab when selecting repositories to clone. You are now successfully connected to your repository.

1. At the top right, select “Fetch Origin” 

If there are any differences between the files posted on GitHub and your filesystem, a pull request will be generated. Click “Pull from origin” if prompted. A pull request is a request to the GitHub server to download any commit differences between your machine and the server. If your machine is behind, a pull request is generated and any files necessary to catch up will be downloaded upon clicking “Pull from origin.” If your machine is ahead of the origin, a push request will be generated. Push requests are covered in 3.1.1. If done successfully, your machine will have the exact same structure as the files stored on GitHub.

#### Accessing the Files (Windows)

It is important to note that the files that will be on your desktop will be a clone of the files in your repository. This means that any changes that are done on GitHub will reflect on your desktop. Likewise, any changes that are made on your desktop will reflect on your GitHub repository. To access the files on your machine, follow the steps provided.

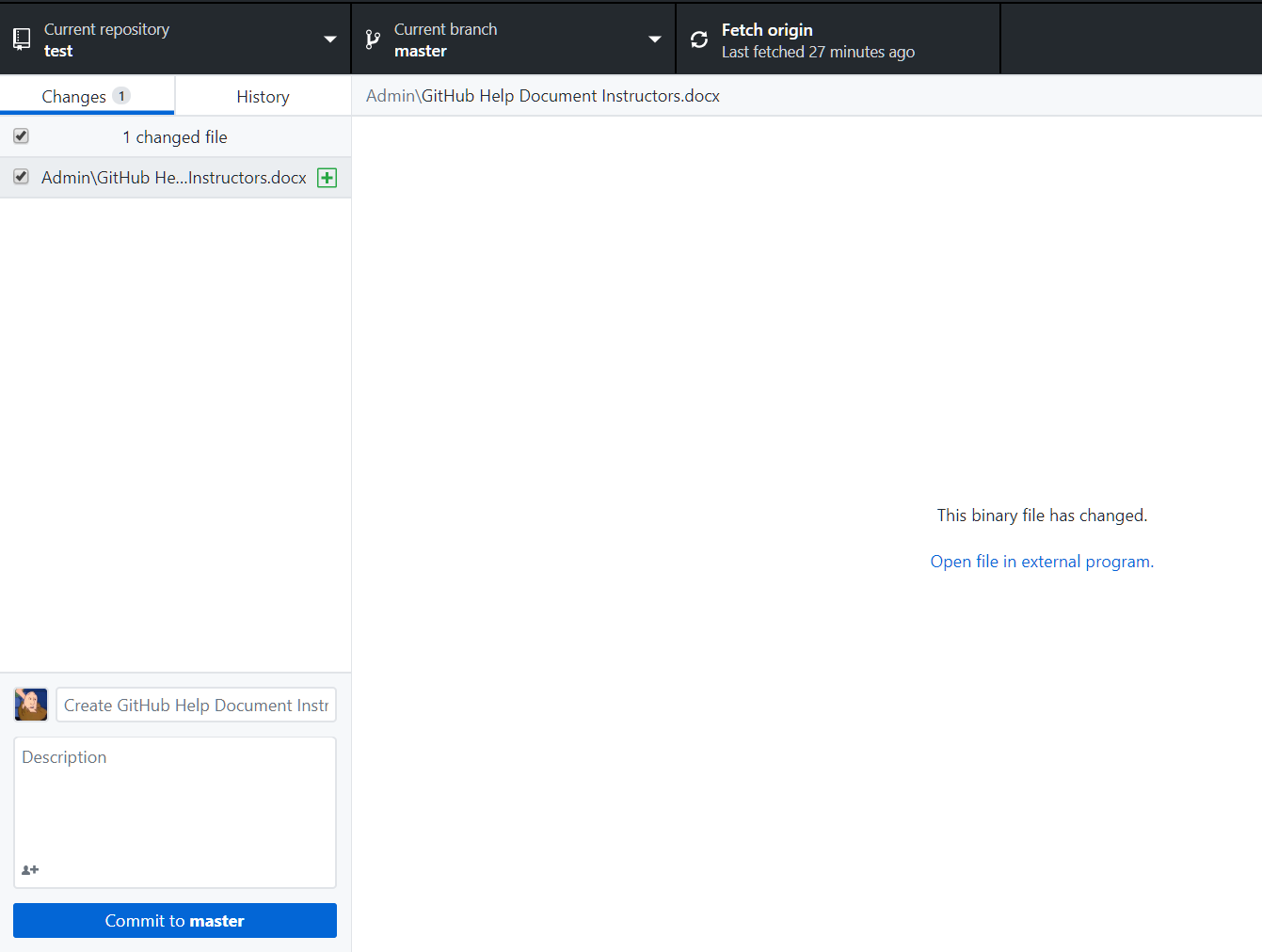
1. Click the “Current Repository” tab on the top-left.
2. Right click on the active repository.
3. Click “show in explorer” 

This will open the corresponding directory to the repository on your machine.

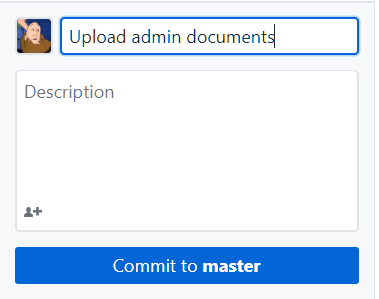
#### Maintaining Sync with GitHub

Now that you have successfully cloned your repository, we want to maintain a certain level of sync with your repository on GitHub’s servers. Any changes that are made on your filesystem **WILL NOT** automatically push to GitHub. In the following example, we will upload this document to our repository in a folder called “Admin.” These steps can be done to create any structure for your repository. All folders and files will maintain the same state that they are uploaded.

With our repository open, create a new folder and place the document inside. After doing this open the GitHub Desktop application. You will notice the changes that you’ve made reflect automatically within the software.



If you are satisfied with these changes, we need to commit these changes to master. To do this, we will create a title and an optional description in the bottom left of the screen. If there are any documents that you do not want to upload, you may simply uncheck them from the list on the left or remove them from the directory entirely.



After clicking “Commit to master,” we finally must push these changes to the server. This can be done by clicking “Push origin” on the top-right of your screen. Once completed, our changes will reflect on GitHub.com in our repository.

# Utilization of GitHub in the Classroom

## Sharing Documents With Peers and Students

Sometimes, it is useful to simply provide documents to your students like the format that D2L offers. When providing documents to students, it is not necessary for the students to have a GitHub account so long as your project is public. Create a repository strictly for your course and be sure to only add instructors and TAs as necessary as project collaborators. Your students will be able to view the files and download them as needed.

### Putting Documents on the Repository

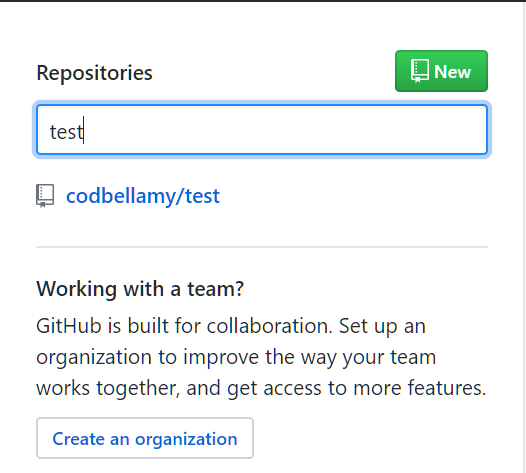
Once your repository is set up, adding documents to the repository is relatively straight forward barring managing the file structure (directory layout). The following steps provided will walkthrough creating a folder and adding documents to your repository.

#### GitHub Desktop

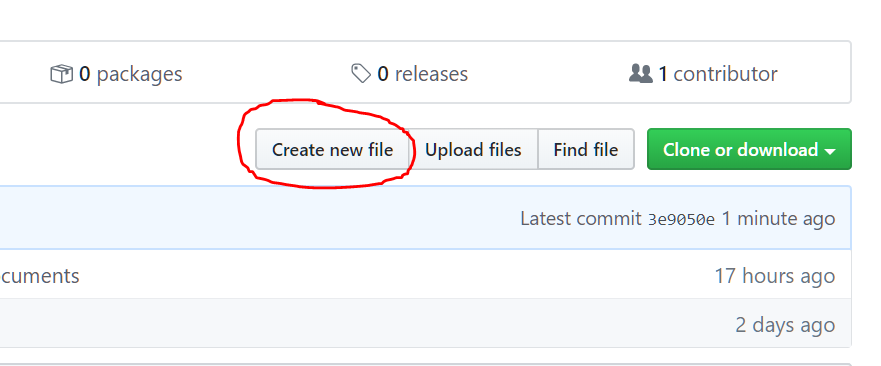
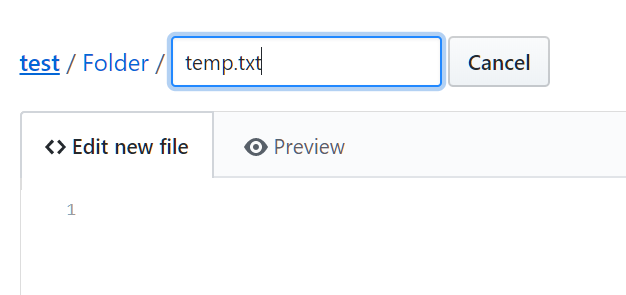
See 2.2.2.2-3 for a detailed description on how to upload folders and files to your repository.

#### Website

Adding files to your repository is slightly more complex compared to uploading documents with the software. Additionally, creating folders is even more non-intuitive. Again, I highly recommend using the software for convenience. If downloading the software is not possible (using a .MIL or some .EDU machines) then the following steps will show how to create a folder and upload a file.

1. Once on GitHub.com, sign into your account
2. On the top-left, search for the name of your repository
3. Click the repository link

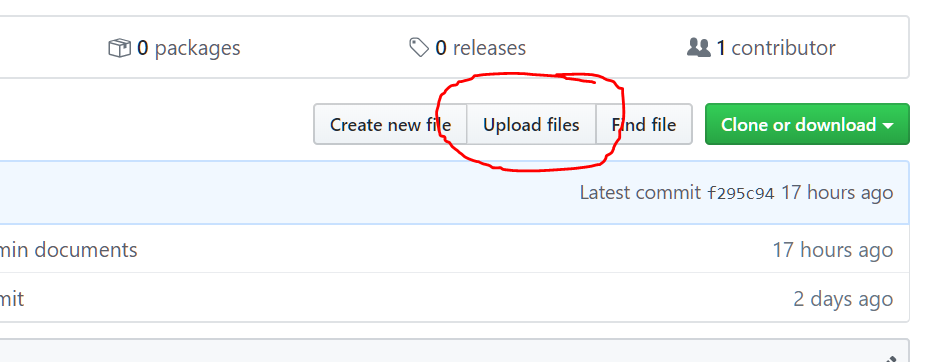
We will now create a folder to place these files. Creating a new folder is completely non-intuitive. Following these steps is currently the easiest way to create a folder when using the website. **If you only want to upload files to an existing folder**, navigate to the desired folder and continue to step 8.

1. At the top of your repository, click “Create new file” 
2. Leave the contents of this file empty
3. At the top of the page, name the file “Folder/temp.txt” where “Folder” is a placeholder and “temp.txt” is a temporary file we will place into our new folder. 

You will notice that each time you type a forward slash, the text will be moved to the left. This indicates that the file will be placed in that directory.

1. Click “commit changes” at the bottom of the webpage

After doing these steps, you will be redirected to the directory you have created. The following steps will illustrate uploading your desired content to this folder. After following these steps, click on your temporary text file to view its contents. Subsequently, delete this file since it is no longer needed and commit these changes. Note that if you delete the temporary file before uploading your content, the directory will be destroyed automatically. **Only delete this file AFTER** uploading the desired content.

1. At the top of your repository, click “Upload Files” 
2. Drag or click “choose your files” to upload necessary files
3. Make a title to commit these changes
4. Click “Commit changes” at the bottom

After your files have been uploaded, you will be redirected to the repository home page. You should notice the files you uploaded should be in the root directory.

### Cloning Repositories (Students)

After uploading your documents and setting up your repository, you may want to give access to your class. This can simply be done by sending your students the link to your repository. If they have GitHub desktop, they can clone the repository directly through the software using the link provided. Since they have not been added as collaborators, there is no worry of your students accidentally committing changes to your repository since they do not have the appropriate permissions. Thus, successfully avoiding a catastrophe. The benefit of using GitHub desktop is that students may keep very accurate and up-to-date changes to your repository.

Additionally, students may navigate to your repository using the link provided and download the contents directly from the website. This is very useful if there is no programming collaboration required and you are using GitHub simply as cloud storage for course files (similar to D2L). The benefit to downloading from the website is the ability to download individual files while ignoring content that is not required to be downloaded.

### Updating Documents

When using GitHub desktop, simply navigate to the directory where the files are stored. Any changes to these files will reflect as a change on the software. The user can simply commit and push these changes and the changes to the file will reflect on the repository.

If using the website, update the file locally. Then, delete the file on the repository and upload the replacement.

### What is Forking?

What is forking and how is it different from branching? For the purpose of using GitHub in a classroom environment, branching is hardly necessary. Branching is when collaborators on a project decide that a project can take two or more paths in development. Another use for branching is when you have two teams working on the same project. Both teams could be working on the exact same project, but one team’s code might be more efficient or elegant than the other team. In this case, the better of the two codes would be merged with the master branch and committed to the final project. In both cases, branching can cause confusion and adds more levels of difficulty than benefit for a classroom environment. In this document, we will not be branching repositories.

This brings us to forking. Forking a project or repository can be done by anyone with any level of privileges. The only time a project cannot be forked is when the project is set to private. Only those with specific read access to the repository can fork this. So what is forking? Forking is cloning the entire repository and taking full ownership privileges of the fork. A forked project cannot commit changes to the original repository unless the owner of the original repository gives permission via a pull request.

## Projects

When students need to work on a project there are a few decisions by the instructor to make. It is important to structure your GitHub appropriately depending on the needs of the course. Setting up the structure of this project could either make this the project easier or very inconvenient for both parties if not structured properly. We will cover all avenues in this section.

Some questions to consider when structuring this project are as follows:

* Will the students need prerequisite “skeleton” code?
* Will the students need data files?
* Is grading automated?
* Are variable and function names a requirement of this project?
* Is collaboration and equal participation encouraged?

### Structure

When designing a project for your students, the aforementioned questions are very important to consider. For example, assume groups of 4 students are to work on a project together and some skeleton code (such as some pre-designed classes) is provided. Then it would make sense for one student to fork your project repository (containing the skeleton code) and allow his or her classmates to collaborate on their fork. As an instructor, you can monitor the contribution of each student by viewing each commit. By viewing each commit, you can see exactly which lines of code were contributed from whom. Additionally, for the purpose of grading, the instructor can clone their fork, and test their code in the same environment that the students were programming in. Their file structure, variable and function names, and additional files are preserved. Below is a table that illustrates which structure is best suited for the project based on needs.

#### Clone Only

When creating a clone-only structure, it is important to note that the instructors and TAs will be the only collaborators on the repository. This means that the students cannot contribute to any files in this repository. They can, however clone or download any files necessary. When using a clone-only structure, it works best when a folder with the project contents are provided to the students from within the course’s main repository. No additional repositories need to be generated. Clone-only makes it difficult for a particular structure to be provided and to see precise commits and progress on the project.

#### Forking

Forking a project is probably the most collaborative-friendly way of allowing the students to work on a project. The instructor will create a new repository that contains a specific layout to the directory along with some skeleton code and/or some data files. The students can then fork this repository. Essentially, forking a project duplicates the repository in its entirety but allows the student to become the owner of their own fork. Now that the student is the owner of this fork, they can commit changes, add files, and invite others to be collaborators on their project. The instructor can monitor all commits made by each student within each fork.

The instructor also has the capability to access all forks made of their repository. More essentially, the student’s fork can be cloned by the instructor and their code can be run in the exact same environment that the student was programming in. This process eliminates any issues with variable names, function names, and file names.

To create a fork, go to the desired repository on GitHub.com. You will notice a fork button at the top-right of the page. If students are signed into their account, they can click this button to have a fork added to their list of repositories. From here, it is treated like any other repository that they own.

Next to the fork button, there is another link that shows a number. This number shows how many forks have been made of your repository. When navigating to this link, it will show all users who forked your repository as well as a link to their fork. You can oversee the progress of a project in this manner.

#### Full Privileges

The final structure for projects is giving students full or pseudo-full privileges to your repository. A new project repository can be created with some files. Then, all students (or instructors) involved will have full access to your repository. This is dangerous in a classroom environment since students can accidentally commit a change to the repository that could be detrimental to project in its entirety. Although these changes can be rolled back, it would generally be a hassle to have to roll back multiple commits throughout a project’s life. This can be managed through branch rules and merging to the master branch via pull requests. For the purposes of this document, we will avoid adding unnecessary steps which would complicate the use of GitHub for a classroom. This method is useful within a small group such as a Capstone project or directed study. Adding users as collaborators is covered in the next section.

### Collaborators

The term collaborator, in GitHub’s respect, is interchangeable with co-owner. Collaborators can commit unrestricted changes to the repository, create branches, add more collaborators, etc.

To add a collaborator to your repository, access the repository from GitHub.com. Under the “Settings” tab, click “Manage Access.” You might be prompted to enter your password. After verifying your credentials, you will see an option to add collaborators. You can search the user by email, full name, or email. After finding the user and selecting them, an email will be sent to the user notifying them of their invitation. Each user that is invited as a collaborator will not show up in the list of collaborators until they have accepted their invite from their email. Once the invite is accepted, they are now full collaborators on the repository.

### Committing Changes

# Project Management