

GitHub Overview

What is Git?

1. Git is a version control system that allows us to track the changes we make to our project files.
2. We use git because it allows us to back up our project files on what is called a **Repository**.

What is a Repository?

1. A repository is just where our project files are being kept. These are the folders that you see on your computer.
2. There are two main types of repositories that we will be using in class.
 - a. The **Local Repository** is the project files that are kept on our computer.
 - b. The **GitHub Repository** is the project files that are stored on the GitHub servers.
3. For this class we will be using **GitHub Desktop** to keep track of our repositories.
 - a. **Using GitHub will be required for your group projects!**

GitHub and GitHub Desktop

1. First things first, make sure that you have a GitHub account.
 - a. If not, go to <https://github.com/> and create an account.
2. Next make sure that GitHub Desktop is installed on your computer.
 - a. If you need to install it, you can go here: <https://desktop.github.com/>

Creating a GitHub Repository


For this class, we are going to be creating various repositories that house the files for the projects we make in class. The repository we make for today's class will store the files for our next lesson.

1. Creating a repository is very simple. Go on to [GitHub.com](https://github.com), make sure you are logged in and then click on the green **New** button.



2. First you need to select the **Owner** and **Name** of the Repository.


Owner * Repository name *

 AsleepInTheBreakroom / UnityIntro

✔️ UnityIntro is available.

If you belong to different GitHub groups, you can choose what group the repository will be created in.

Owner * Repository name *

 gl-athemos / GitHubRefresher


✔️ GitHubRefresher is available.


- It is advised that you should give your repository a **Description**. This way you can say what the repository is for.


Description (optional)

Repository for the intro to unity lesson.


- You then chose the **type** of repository.

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

☒  **Internal**
Ohio University [enterprise members](#) can see this repository. You can choose who can commit.

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

For this project we will be setting our repository to **Private**.

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

- Lastly, it is **required** that you use a **.gitignore** file.

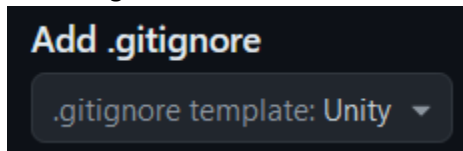
Add .gitignore

.gitignore template: None

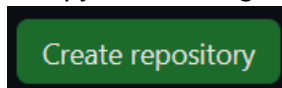
The .gitignore file tells github to ignore certain files when we upload them to github. We require using .gitignore files because without them some projects will not be able to be uploaded to github.

From the drop down list you can search up what type .gitignore file you will use for your project.

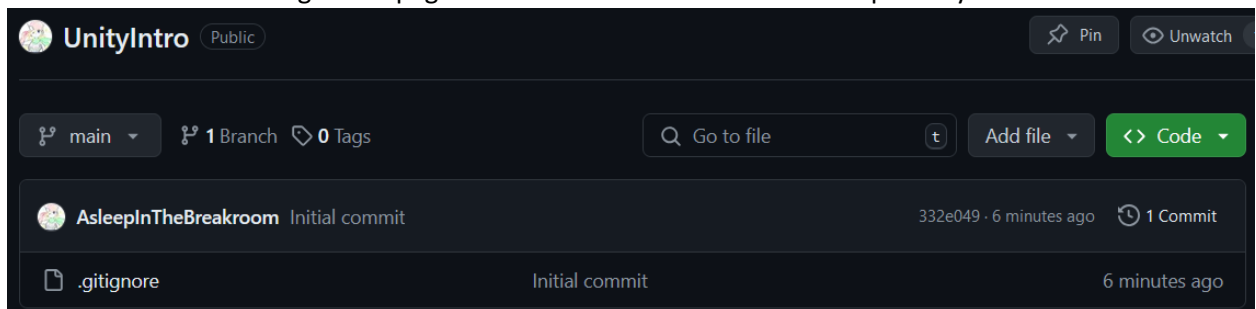
For our class projects we will use the **Unity** gitignore file but there should be ignore files for other engines such as Unreal or Godot.



6. Lastly just Click the green **Create Repository** button.



7. You should then be brought to a page that will show all the files of the repository.



We should only have the .gitignore file that we created along with the repository.

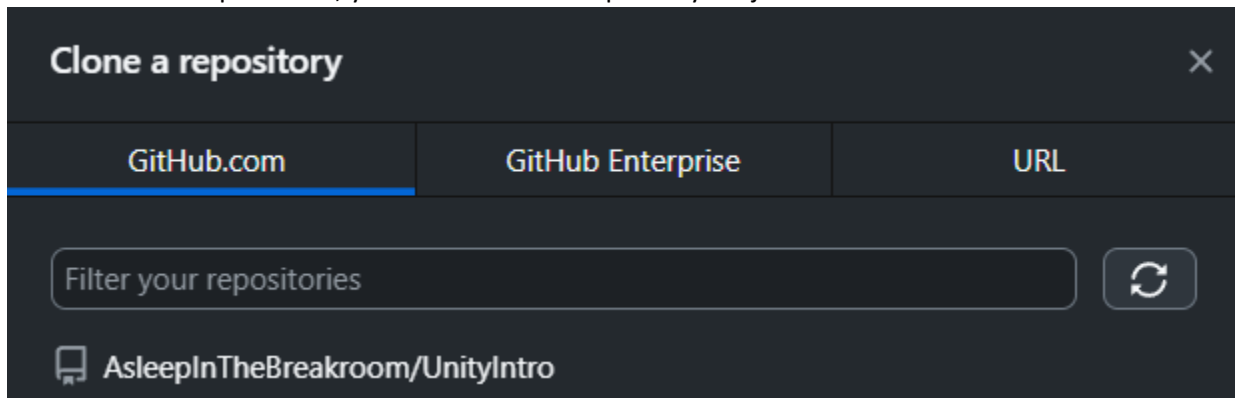
For group projects, I will be creating the repositories that will be used for them, so you do not need to worry about making those yourself.

This will require you to accept an invite from GitHub to be accepted into the project's repository.

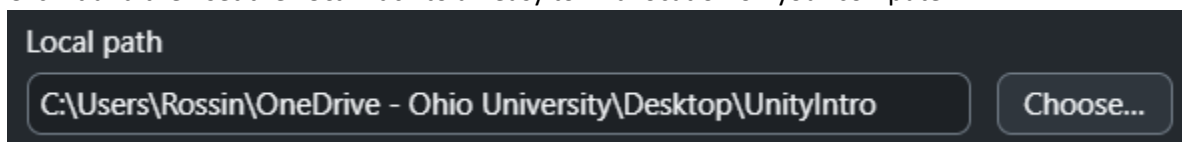
Cloning a Repository Via GitHub Desktop

1. Whenever we go to download our project, we need to make sure to do it via **GitHub Desktop**.
 - a. On GitHub you are given the option to download your project via a zip folder.
 - b. **DON'T DO THIS.** This will not sync up the files you download with the files that are on the GitHub servers.
2. To download of newly created repository, we need to open up **GutHub Desktop**, sign in and click **File > Clone Repository**.

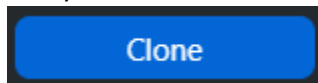
- From the list of repositories, you should see the repository we just created on GitHub.



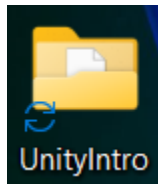
- Click it and then set the **Local Path** to an easy to find location on your computer.



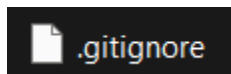
- Lastly click the **Clone** button.



- You should then see the repository on your computer.

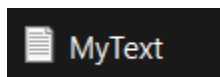


If we look inside we will also see that we have the .gitignore file as well.

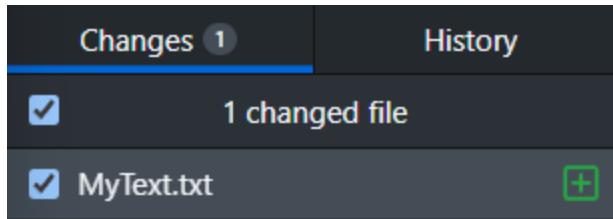


Pushing and Pulling

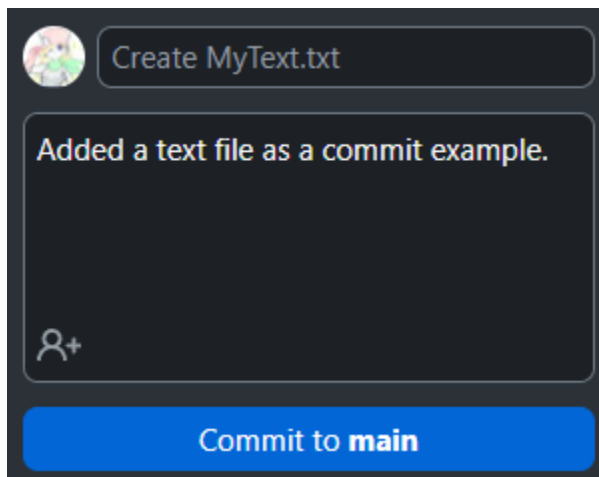
- Now that we have the repository on our computer lets **add a text file** to it.



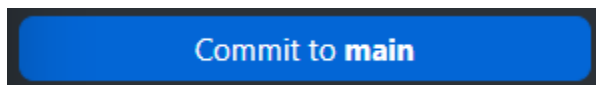
- Once we do, if we go back to GitHub Desktop, we will see that it has recognized the new file in the repository.



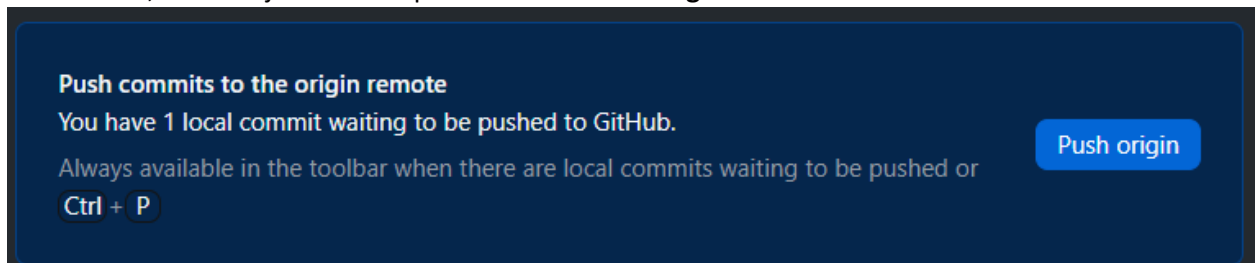
3. Now all we need to do is **Commit** the file.



We give a general description of the changes we made to the project. And then we can give a more detailed description if we choose to. We then just need to hit the blue **Commit to Main** button.

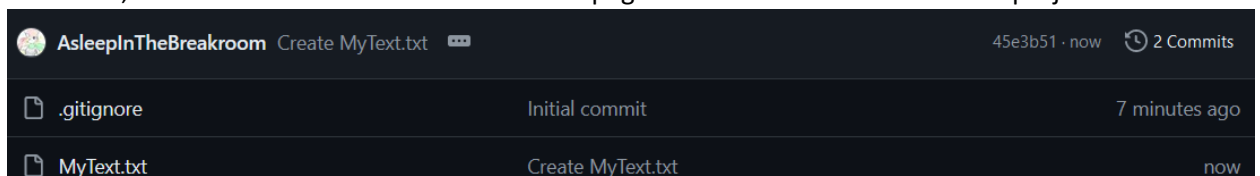


4. Once we do, we then just need to push the blue **Push Origin** button.

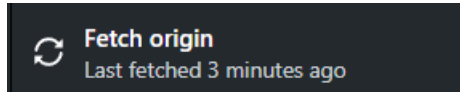


Push essentially just uploads the file from our local repository and puts them on the GitHub repository.

5. If we now, head back over to the GitHub website page we can see the text file in our project.



6. The counterpart to pushing is **Pulling**. Pulling allows us to take the files from the GitHub repository and download them to our local repository.
7. We can pull by going to GitHub Desktop and at the top click on the **Fetch Origin** button.



This shouldn't do anything since the github repository is the same as our local repository.

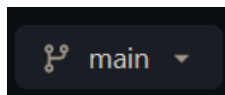
Important: It is a good idea to always **Pull** before you go to work on your team project. That way the project that you are working on is up to date. If you don't do this, you might run into conflicts with the project.

Important: Be sure to work on the same version of unity as the other on your team. If you use separate versions, you will run into conflicts.

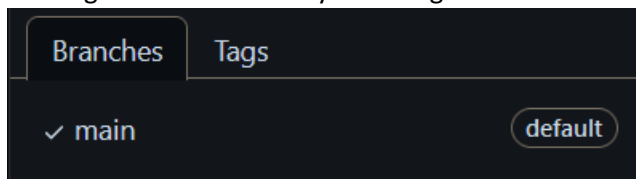
Important: Be sure to push your project often, that way the others in your group can stay up to date.

Branches

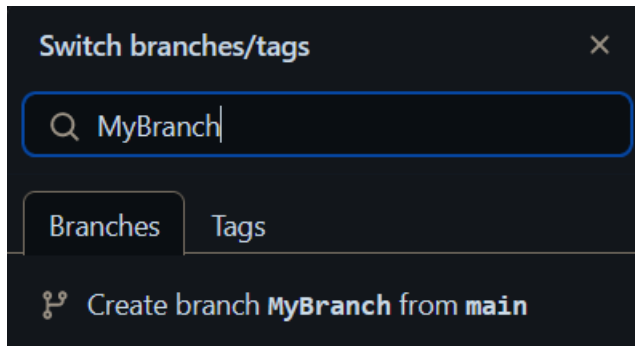
1. Let's take a minute to talk about **Branches**. A branch is just a duplicate copy of the project that you can make changes to without effecting the base project. Branches are a good way to break up your project and it helps avoid conflicts while within a group. You may find it useful if **each team member works on their own branch**.
2. To create a branch, go to the GitHub website and find the **Main** button.



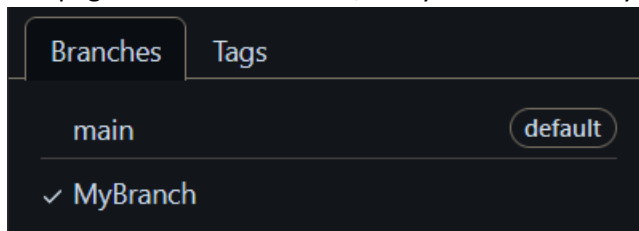
Clicking on it will result in you seeing all the available branches for the project.



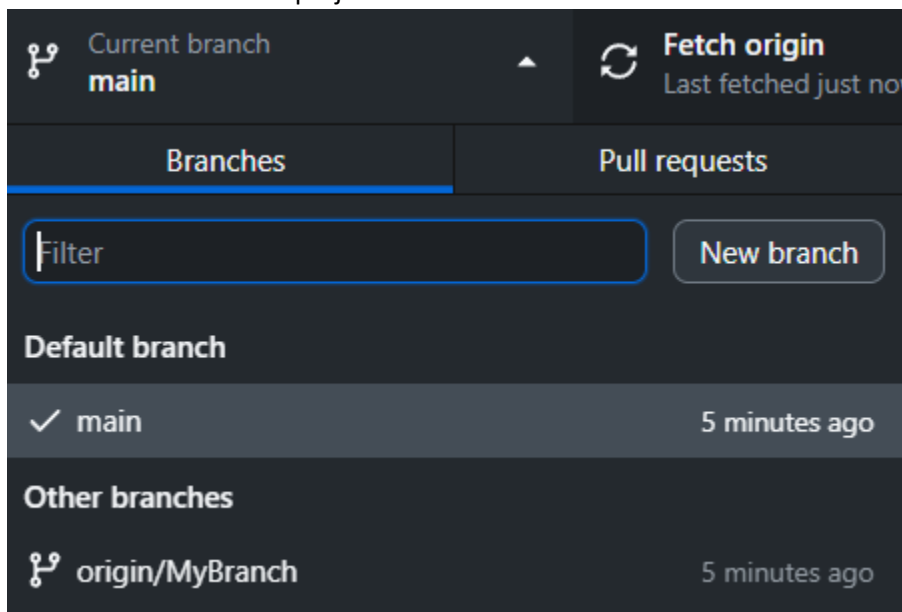
3. To create a new branch, in the search bar, type in the name of the branch and then hit the **Create Branch** button.



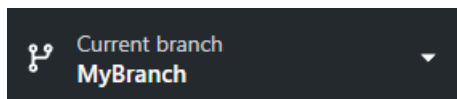
The page should then refresh, and you will see that you are in the branch that you just created.



- Let's head back to GitHub Desktop and change what branch we are currently working on. **Fetch Origin** to update our repository. We should be able to select the **Current Branch** button and then see the branches of our project.



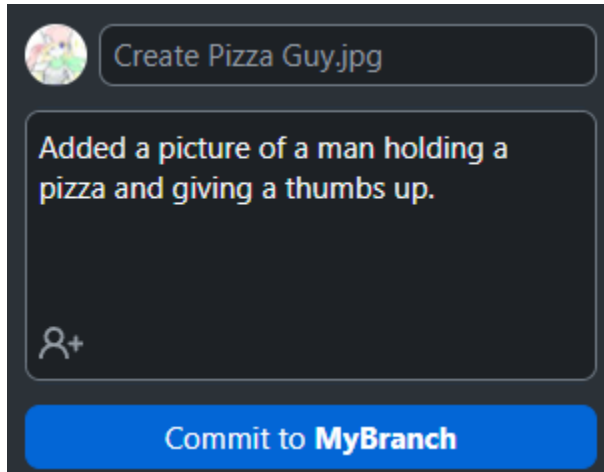
Let's then click on the branch that we want to be on to switch to that branch.



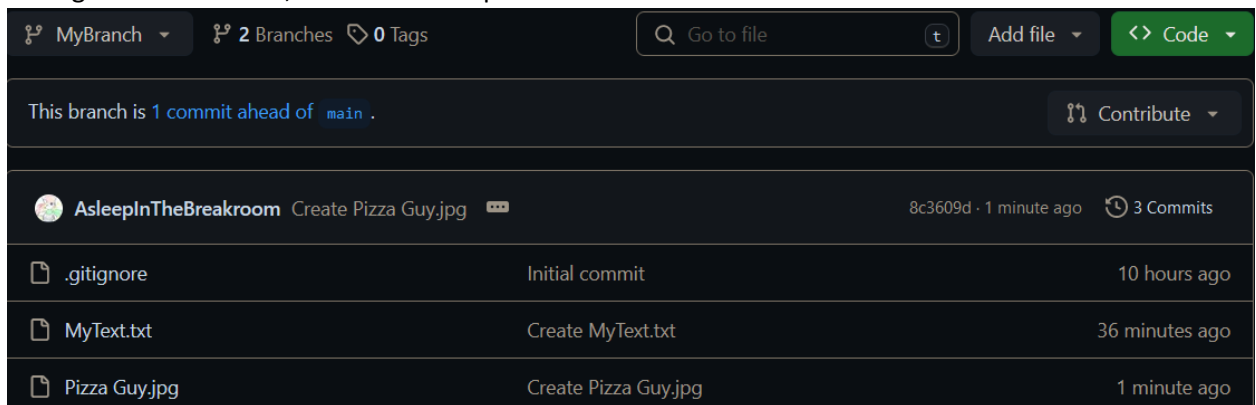
We will be able to see what branch we are on by checking the **Current Branch** button.

5. In our repository, let's add a picture that we grabbed from google.

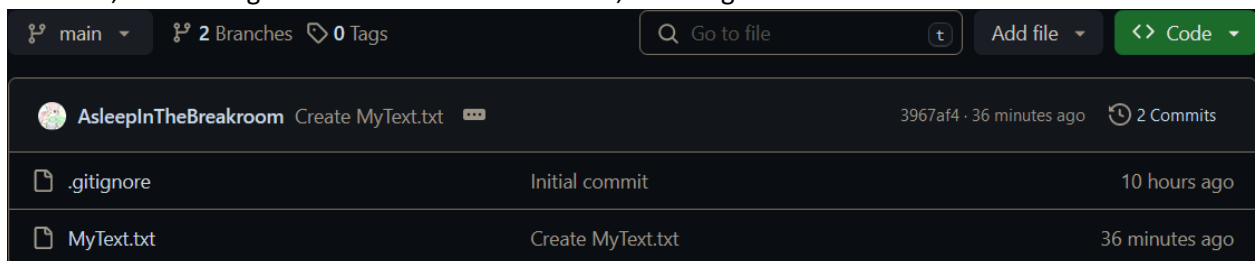
Once added, let's go ahead and commit and then push the file.



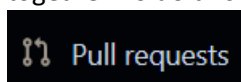
6. If we go back to GitHub, we will see the picture that we added.



However, if we change the branch back to the main, the image file will not be there.



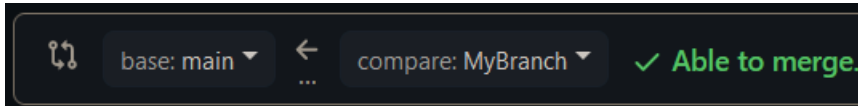
7. Even though the image was pushed because it is on a separate branch it will not be on the main branch. However, at some point in our projects we are going to need to merge branches together. To do this go to the **Pull Requests** tab at the top of the page.



8. Then click on the **Compare & Pull Request** button.



9. We then to set what branches will merge with each other.

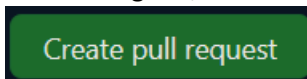


If there are no merge conflicts we should be able to merge without issue.

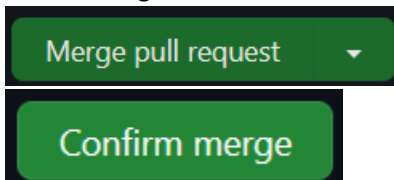
10. We will be able to add a title and a description of what is happening during the merge.



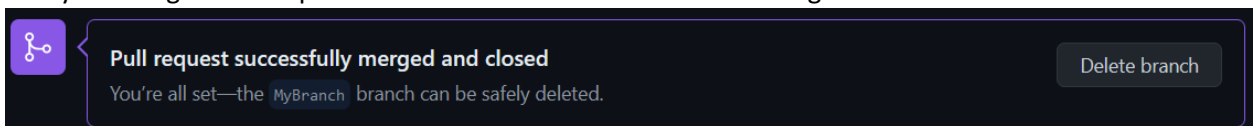
11. If all looks good, click on the **Create Pull Request** button.



12. After a few seconds of scanning, GitHub will allow us to merge the branches. All we need to do is click the **Merge Pull Request** button and then click on the **Confirm Merge** button. The branches will be merged.



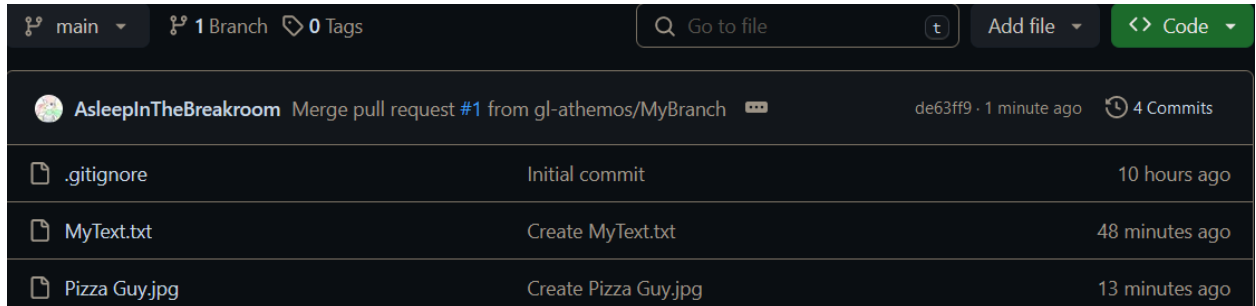
13. Lastly we are given the option to delete the branch since it was merged.



For this instance, let's go ahead and delete the branch since it is no longer in use.

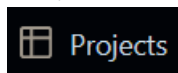
Important: For your projects, you will most likely keep your own branch and will only delete it when it is no longer in use.

14. If we then go to our main branch we can see that our two branches were successfully merged together.

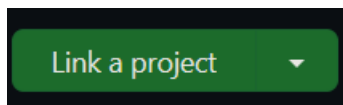


GitHub Projects

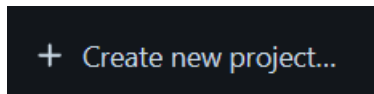
1. The last thing I want to cover is **GitHub Projects**. GitHub projects allow the members of the group to create a task board so they can keep track of what needs to be done for the project. There are other task management programs such as **Hack n' Plan**, **Notion** or **Trello**. But it might be nice to keep everything all in one place and use the program that GitHub provides.
 - a. **Using a task management program will be required for your group project!**
2. First, we need to click on the **Projects** tab at the top of the GitHub page.



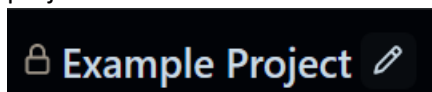
3. We can then click on the **Link A Project** button.



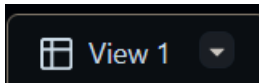
And then select the **Create New Project** from the drop down.



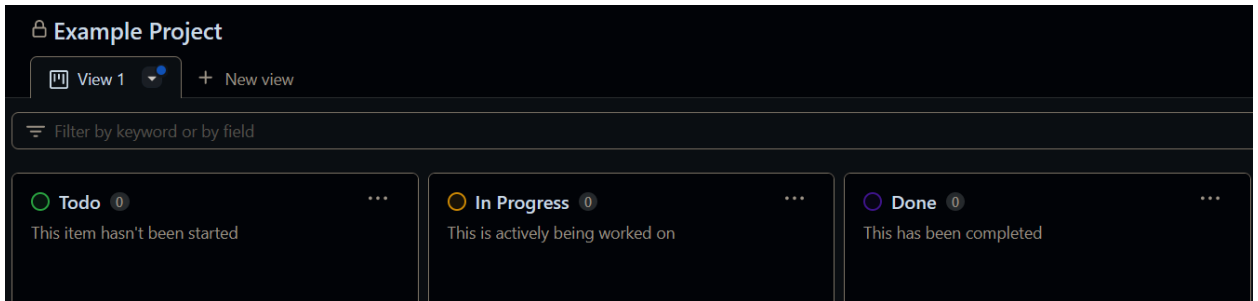
4. Once we do, we will be taken to the project page. Let's first start off by changing the name of the project. We can click on the **Pencil** icon next to the projects current name to edit it.



- Next let's switch the view of the project to a **Task Board**. Click the drop-down arrow next to the **View 1** tab and then select **Board**.

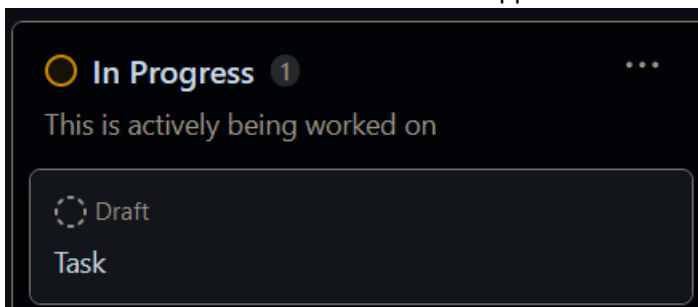


Upon doing so we will see three columns we can use to track the tasks for our project.

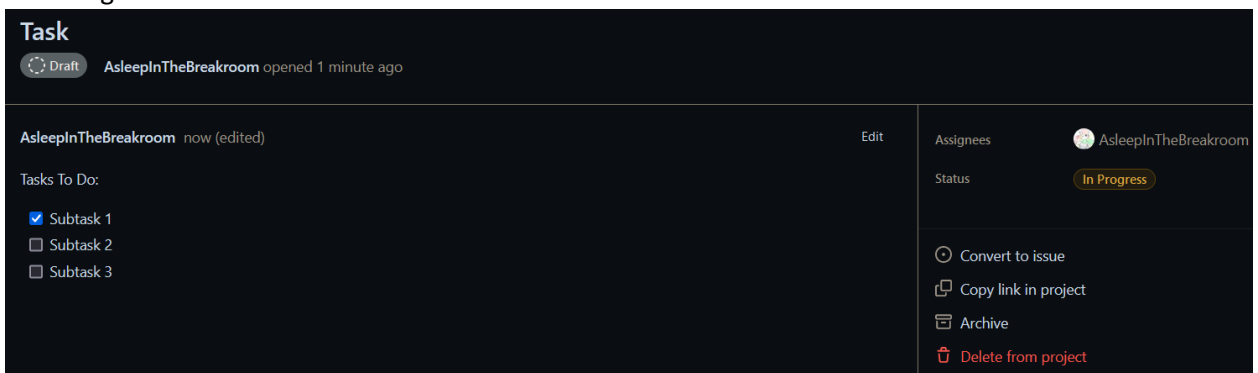


- If we click a **Plus Sign** at any of the bottom columns, we can add a task to one of the columns.

Once we enter a name of the task it will appear in the corresponding column.



- Clicking on the name of the task will allow us to edit it. We can change description, add subtasks, and assign members to the task.



- Once we have made tasks, be sure to click the green **Save** button on the right-hand side of the task window.



