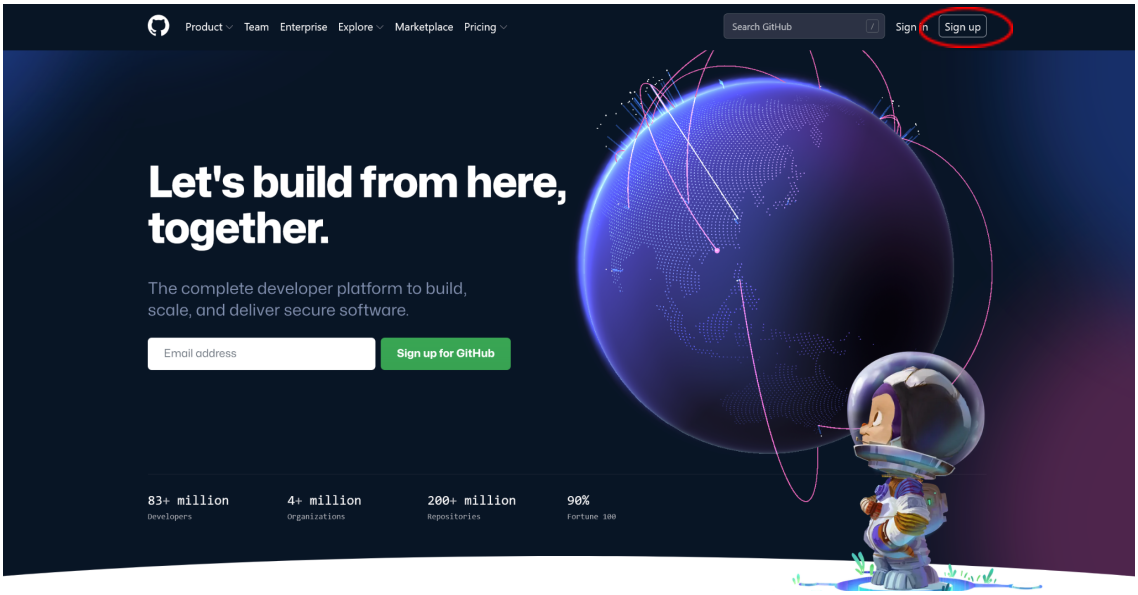
Git is a version control system created by Linus Torvalds in 2005. Using Git, developers can collaborate easily through branching and merging. **GitHub** offers Internet hosting service for software development and version control using Git. Getting your academic/personal projects on github is a way to build your own portfolio and it makes the recruiters see your accomplishments more directly!

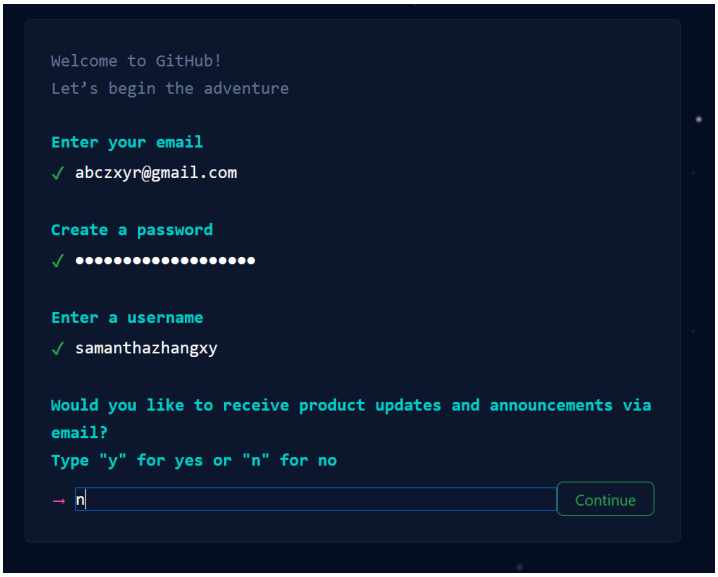
Step 1: sign up & set up repository

1. Sign up or sign in:

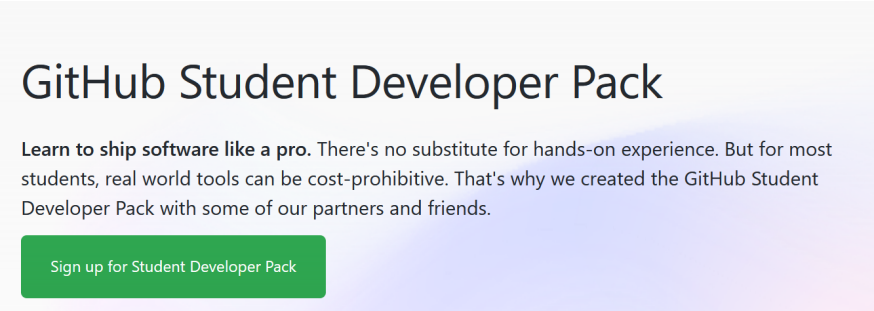
    a. Go to github's official website and click sign up on the top right if you don't already have an account.
    https://github.com/



    b. Enter email, passwork, etc to create the account:

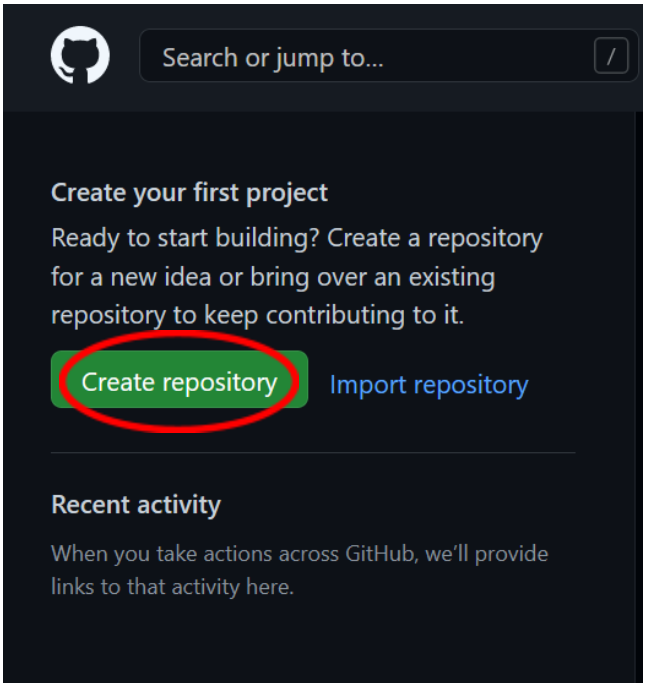


c. As UC Berkeley students, we are eligible to get the student developer pack which includes free GitHub Pro while you are a student, many GitHub Student Developer Pack partner offers, and GitHub Campus Expert training for qualified applicants.

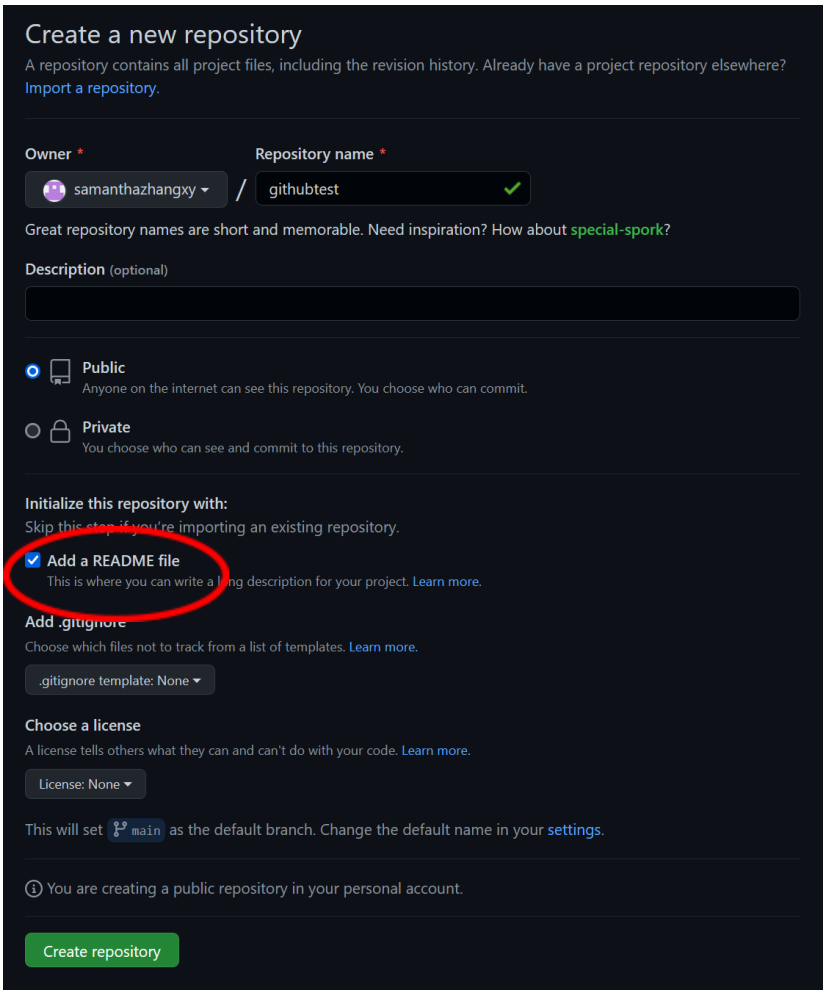


Apply with your berkeley email address.

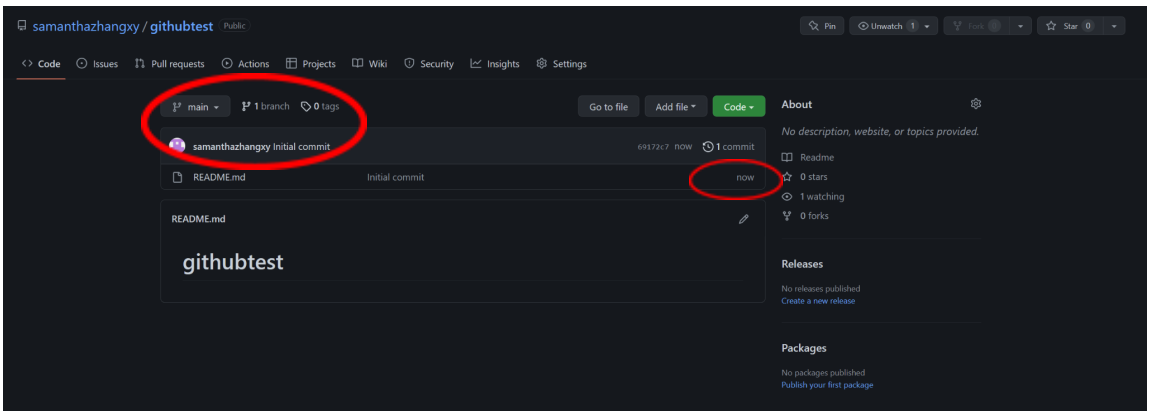
2. At the landing page after sign in/sign up, the left side bar shows your recent github repositories. These can be the ones that you created or it can be the ones that other people added you to as a collaborator.
    a. Now we will create a new repository by clicking on the create repository/new button.



    b. It's always good to have something in your repository, so we can check the Add a README file, which basically acts as an abstract for your project. Also, when we create a new repository, we can always link our local files to the repository and the following tutorial will show you how to achieve that.



c. Now, we can see that we have a new github test repo set up. This is the central portal of the repo where we can manage our branches, files, commits!

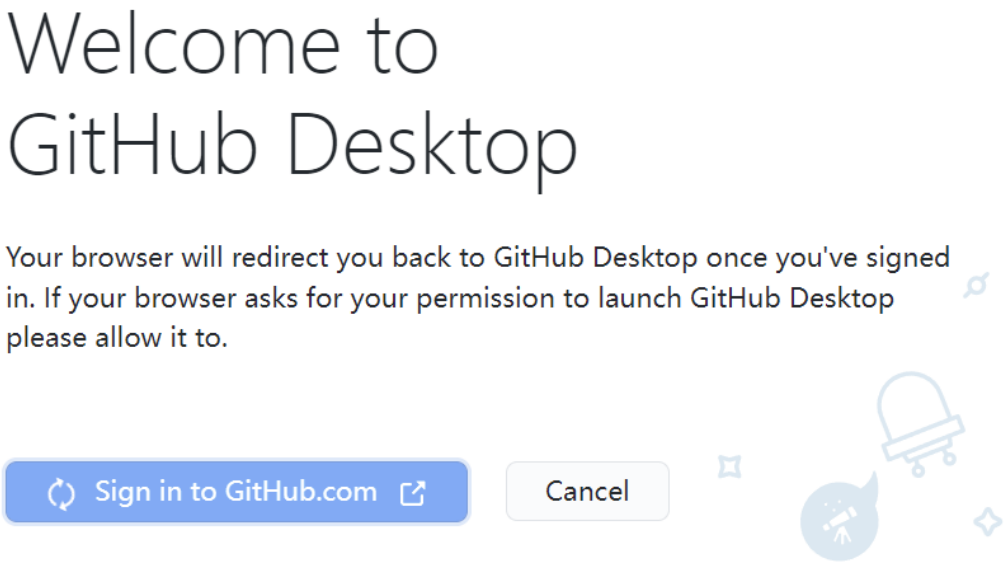


Step 2: Download Github Desktop:

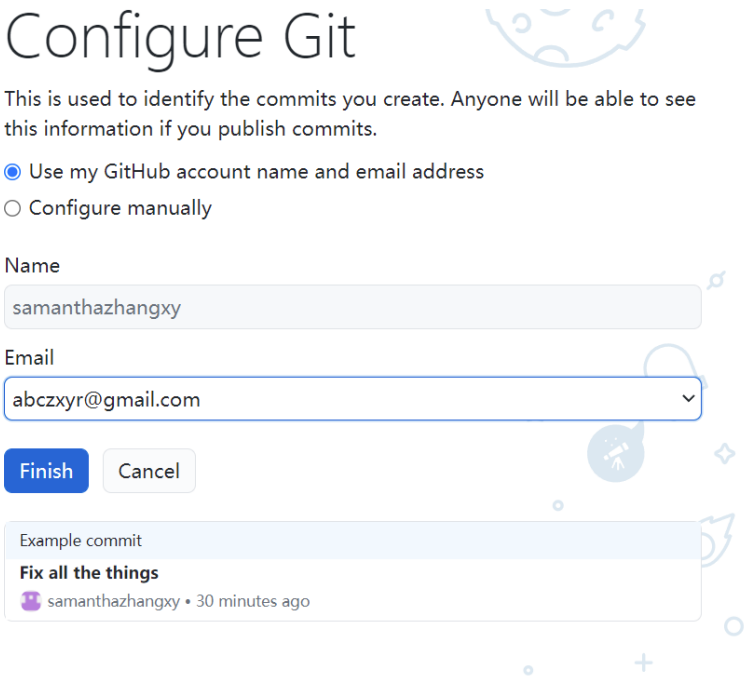
1. To manage our local file in github, we can download github desktop for easier access.

a. Download at https://desktop.github.com/ for both Windows and MaxOS:

2. After installing github desktop, click sign in and authorize the access of your github account through desktop.

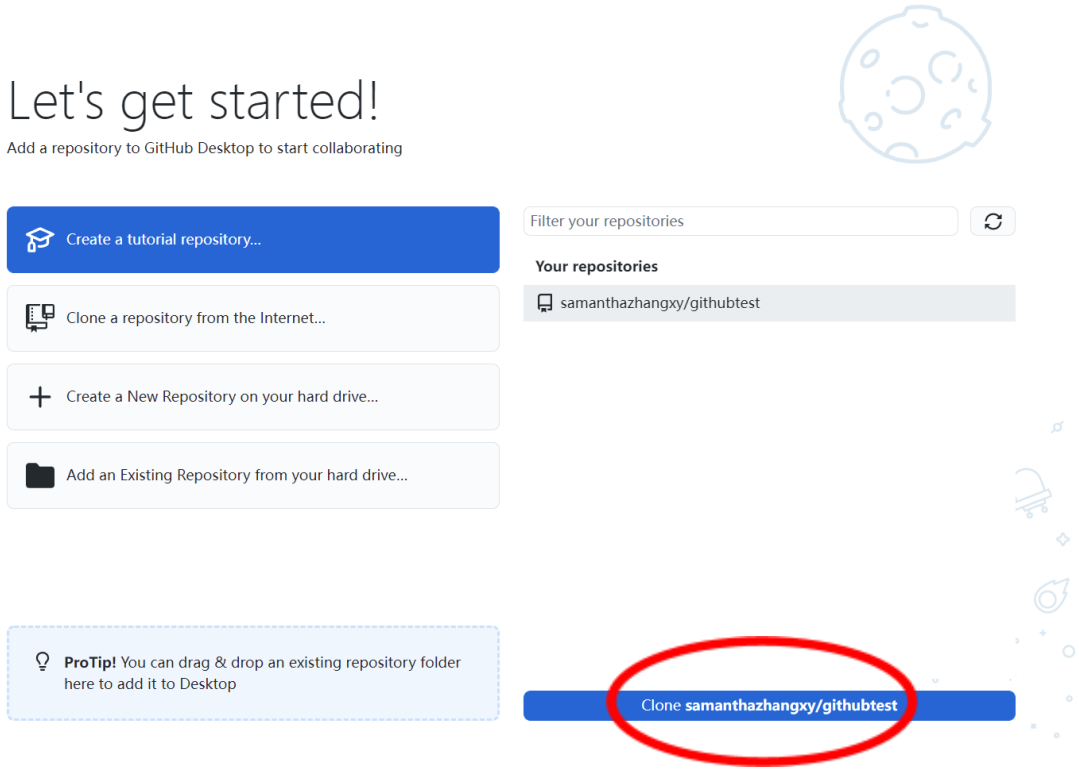


3. Going back to the Github Desktop, we are asked to do some configuration. GitHub Desktop allows you to set the name and email address you would like associated with the commits you make in your repositories. If your name and email address have already been set in the global Git configuration for your computer, GitHub Desktop will detect and use those values. GitHub Desktop also allows you to set a different name and email address for an individual repository. This is useful when you need to use a separate work email address for a specific repository.
    a. In the **Name** field, type the name you'd like to use for your Git configuration.
    b. Select the **Email** dropdown and click the email address you would like to use for your commits.

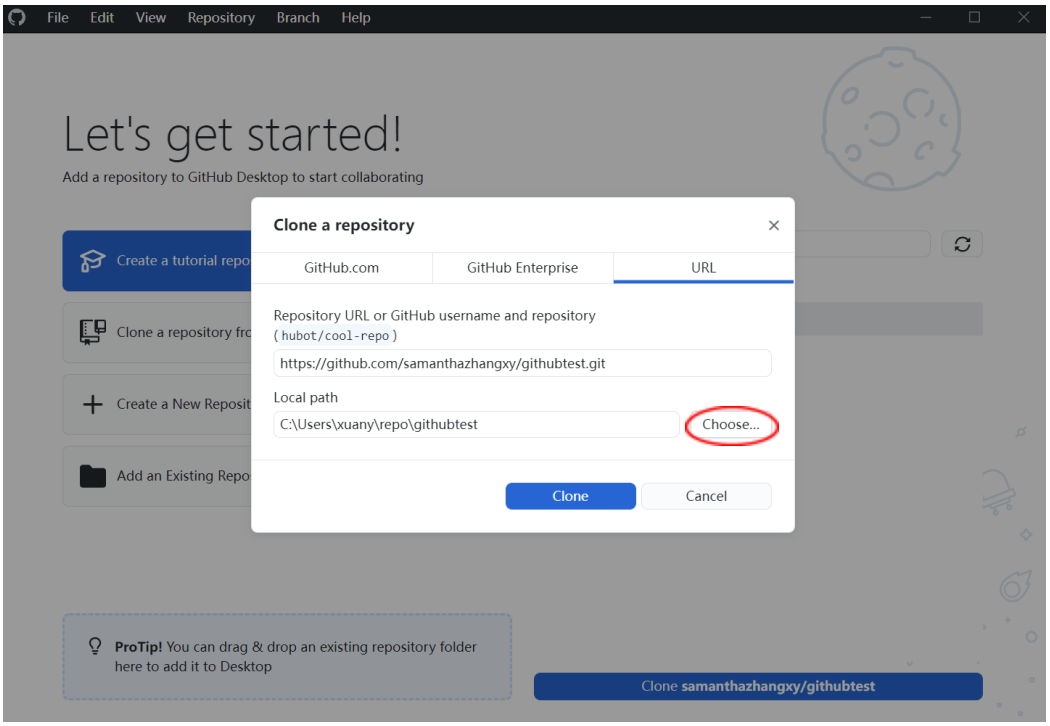


Step 3: Clone a github repository to local:

1. After we successfully set up github desktop, we can clone our test repo in github to our local computer.

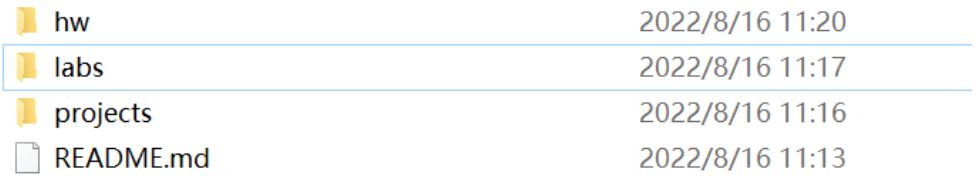


a. We can choose to create a new local repository or if you already have a local project repo, you can place it in the local repo instead.



Step 4: Making changes in the local repo and commit to github:

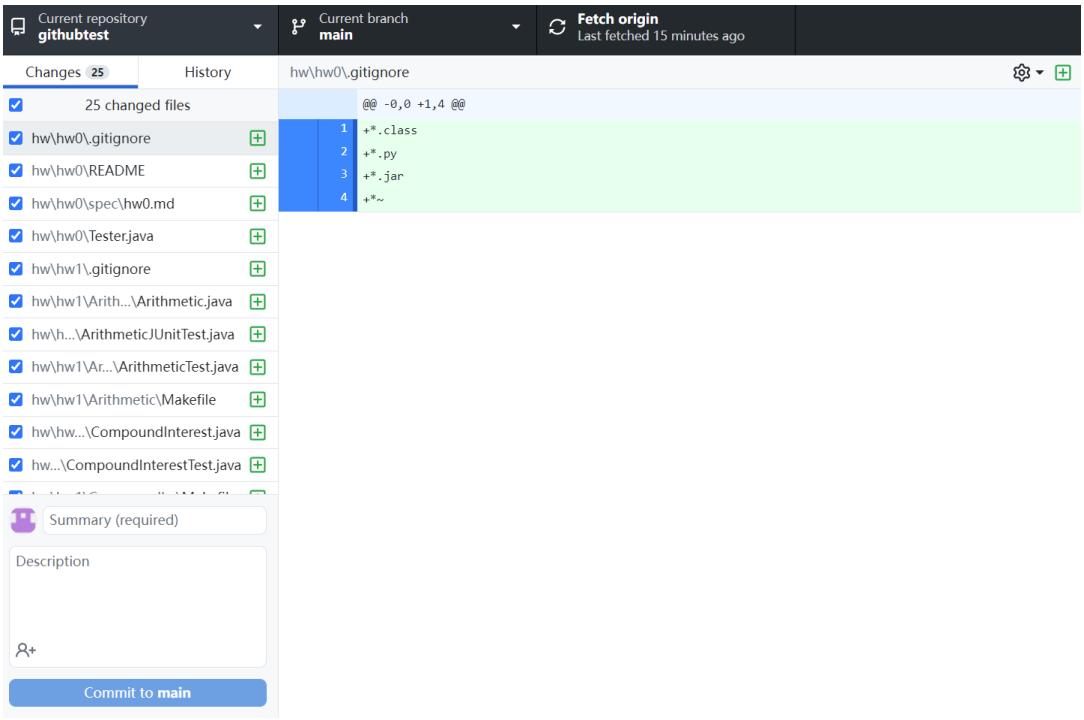
1. Now that we have our test repo cloned to local, we can start working on our projects in the local test repo clone. A great practice is to always have your class projects published online, so in this tutorial, we are going to show that.
    a. We can first create folders for hw, labs, and projects for the class that we want to have on github. We can create those folders locally or through Git Bash for Windows or Terminal for MacOS.



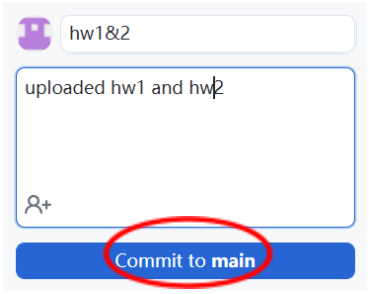
    b. After we create the folders, we can copy and paste our past work into these folders accordingly. Here, I copied two of my 61b hws into the hw folder.

    c. Now go back to github desktop, we can see all of the changes we made. Git doesn't record empty folders created, so the labs and projects folders we created are not shown here.
    Any additions we have in our code are shown in green, and deletions are shown in red.

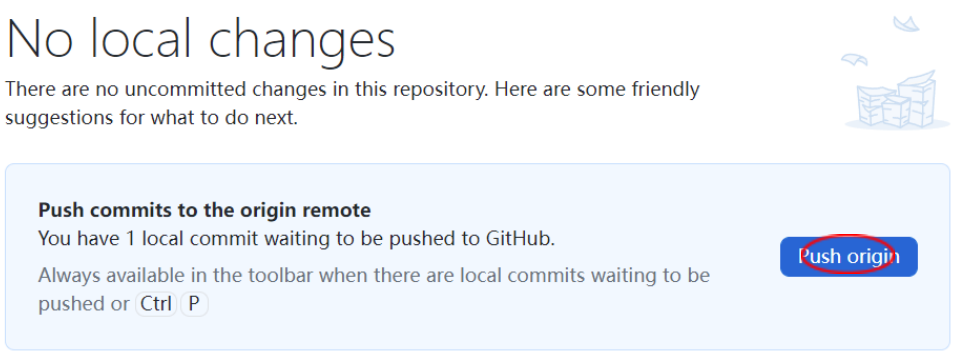
The changes bar functions as status in Git, where all of the changes are shown in the staging area but not yet committed or pushed to our cloud github repository.



d. As we know from using Git, it's always good to make commits in time whenever we make changes to our project files so that we have freedom to go back and forth between versions and won't risk losing our updates.
So in order to commit, we need to give it a name in the Summary box, and we can add a description too, if we want. Once we are ready, hit the commit to the main button.



e. After we have done all the changes we want in our local repo and committed all of the changes, we can push them to the origin remote. These commits will now be reflected in our githubtest remote repository.



f. Go back to our github online repo. We can see the commit we just made and track down which file was changed through the commit as well.