

GitHub Desktop Tutorial

1. **Create GitHub Account:** Go to github.com and create an account. You should use a professional name and email address here, as this name and email address will be attached to your publicly viewable commits. If you keep this professional, you can use this repository as part of your portfolio in the future.
2. **Create a Repository:** Create a repository by clicking the plus sign in the upper right corner next to your avatar. Name it something sensible like "Test Repository". You can leave everything else as default for now.
3. **Open GitHub Desktop:** Download and open the GitHub Desktop application (desktop.github.com). Sign in with your new GitHub account.
4. **Configure Git:** You will be asked to "Configure Git" - use the same name and email you used when creating your account in step 1.
5. **Clone Your Repo:** You will now have three options: Create new repository, Add local repository, and Clone repository. We already created our repository online, so we just need to clone it.

Choose "Clone Repository". From the resulting window, select your new repository from the list.

Clone a repository

GitHub.com

Enterprise

URL

Thomas

Your repositories

Jiyambi/ThomasWasLate

Local path

C:\Users\sarahherzog\Documents\GitHub

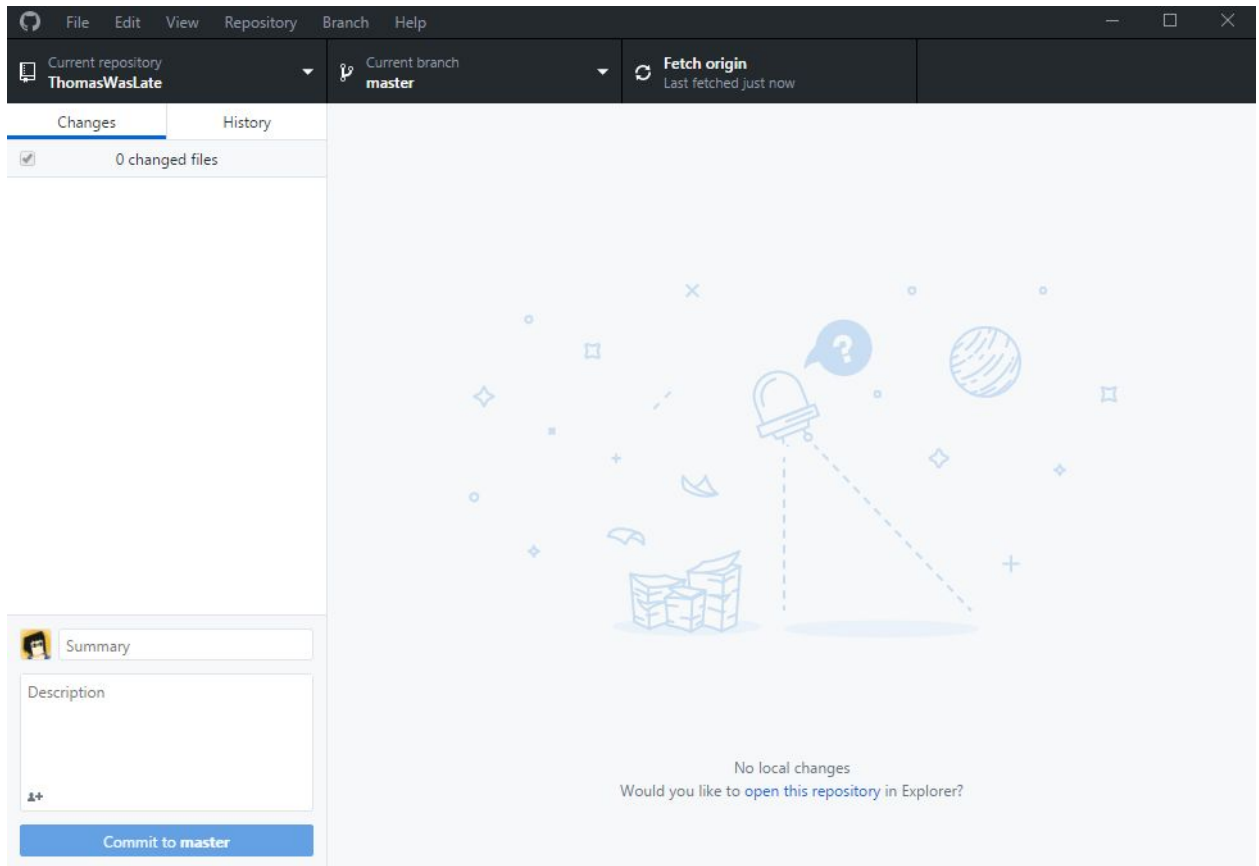
Choose...

Clone

Cancel

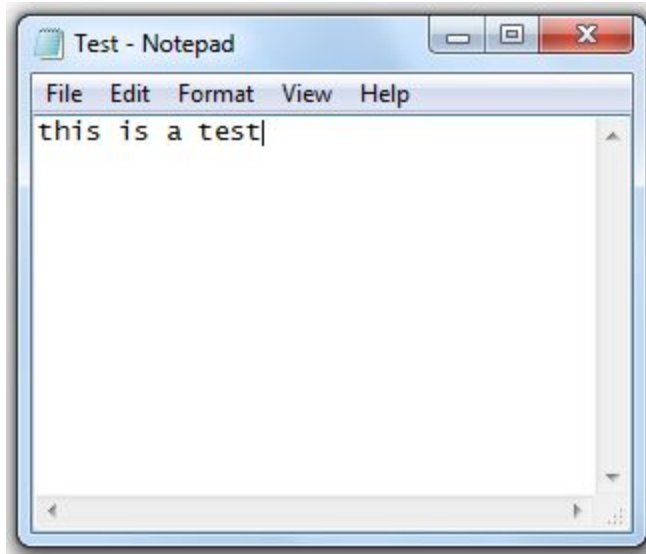
Wait a moment while the program pulls down your repository. This would take longer if you had anything in the repository :)

You should now see something like the following:

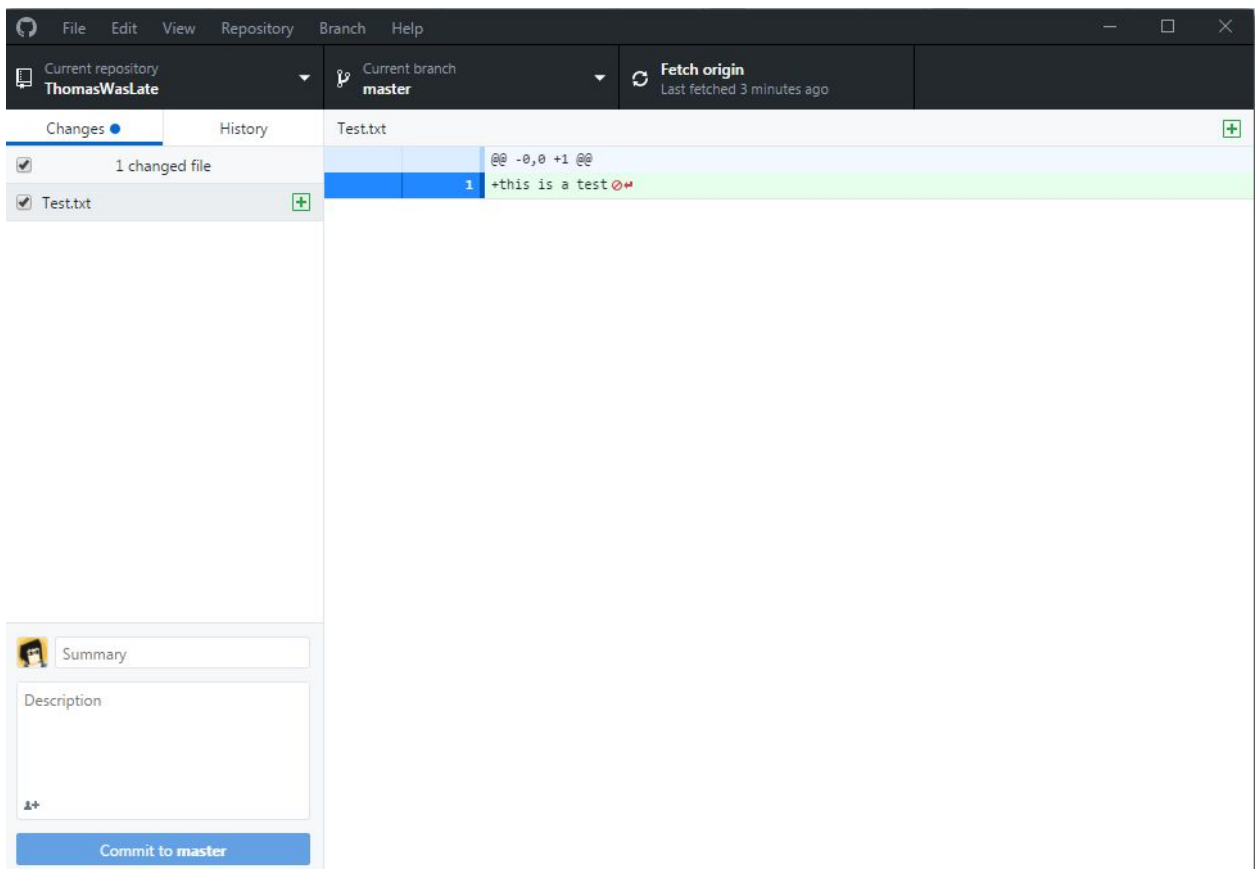


You can see that there are no changes currently. Your history has only one commit in it, an initial commit adding the C++ .gitignore file.

6. **Browse To Repository Location:** Let's test out our repo. Find where your repository is located by clicking on the Repository memory at the top, and click "Show in Explorer".
7. **Add Test File:** Add a plain text file using notepad (not Word!) with just one line, "this is a test". Call the file "Text.txt".



Save your file and head back to GitHub for Desktop. You will see the new file added to the “Changes” tab, and a notice the new lines are shown on the right if you select this new file:



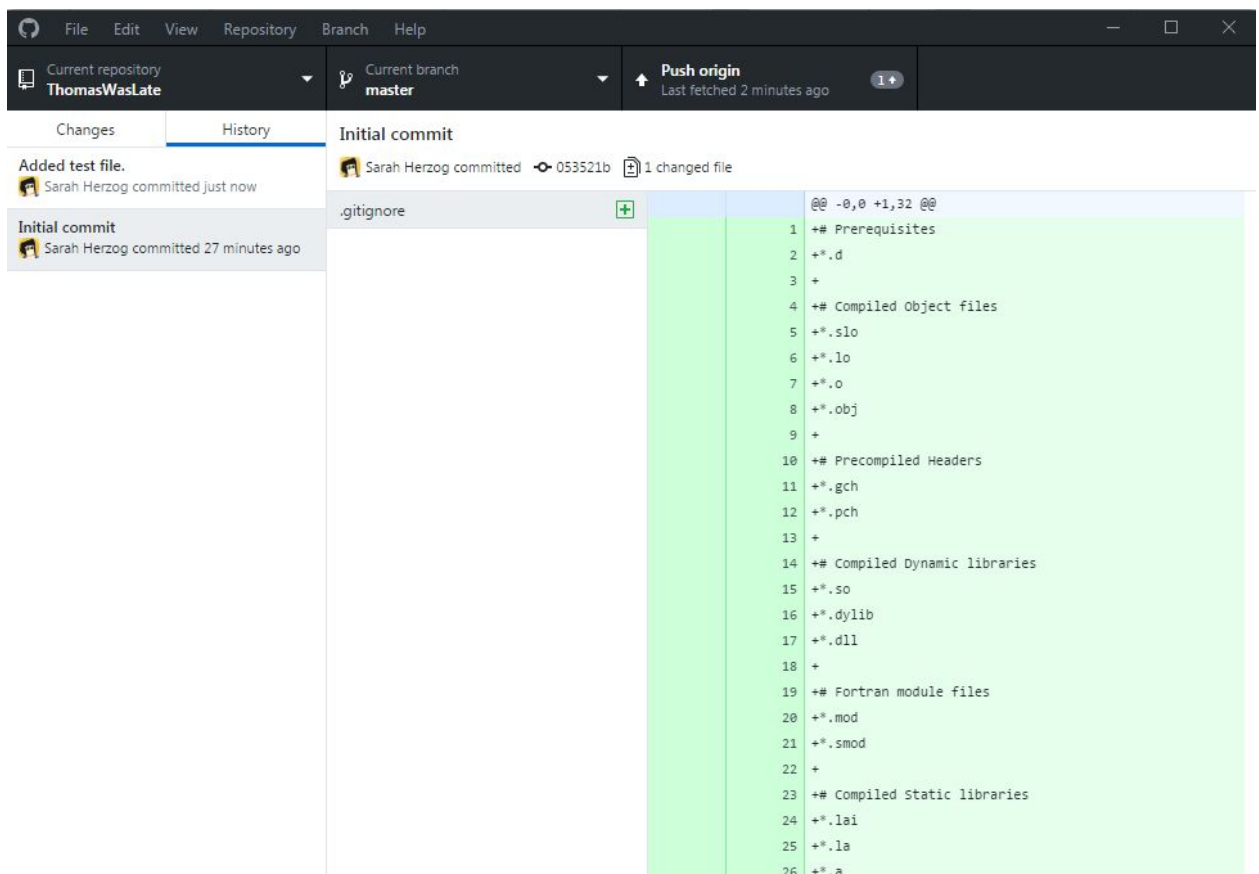
8. **Commit New File:** GitHub Desktop sees that the file is changed, but we haven't yet made our "snapshot" - we need to make a commit.

To do that, we need to enter a "commit message". These **must always** be as descriptive as possible of what you changed. Writing good commit messages is vital to make version control a useful tool.

For our example, write "Added a test text file" in the Summary box. As long as your Summary is detailed enough, you can leave the Description box blank. Then click "Commit to master".

Your first snapshot (commit) has been made! If you check the History tab, you will see your new commit above the initial one.

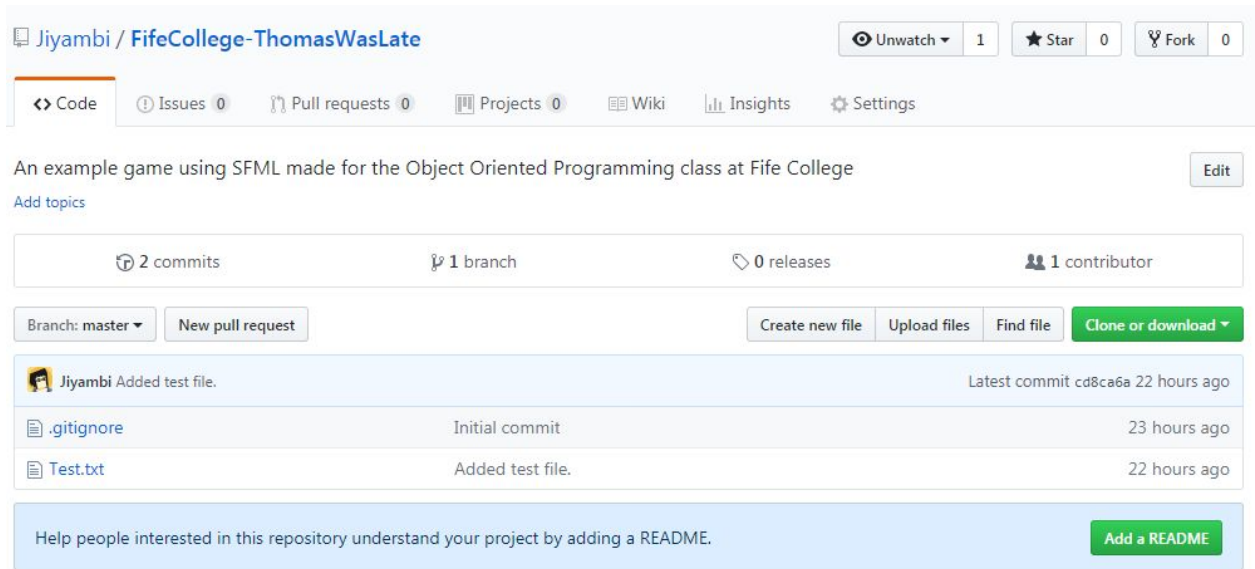
9. **Push Changes:** If you go to the GitHub website, you won't see your new commit. That's because you haven't yet **pushed** your changes from your computer up to the online repository. To do this, click on the button in the top right of the screen that says "Push" - if you have any local changes to push, it will have an up arrow and a number on it.



Now if you go to the GitHub website for your repo, you should see your new file and the

commit you just made in the repo's history.

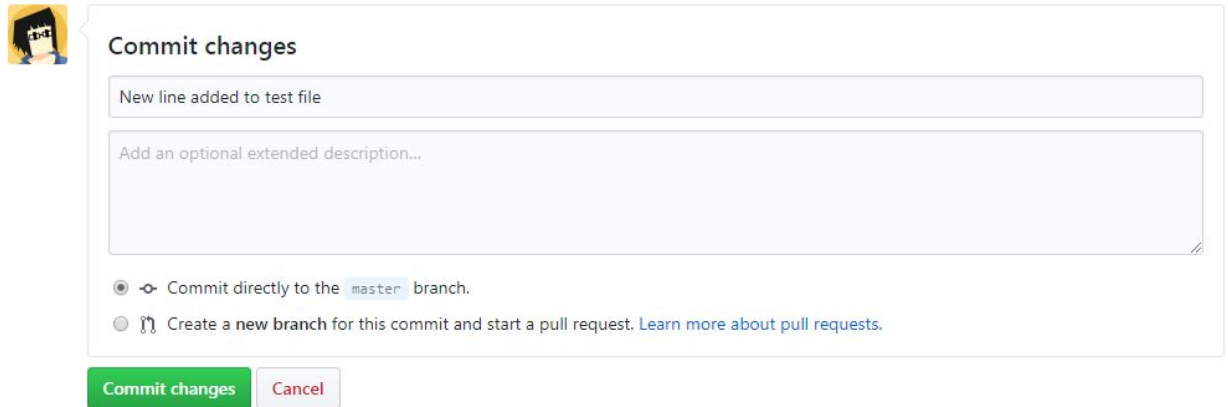
10. Edit File Online: We also need to learn how to pull changes down from our online repository. Let's edit our new text file in the web browser. Go to the text file on the GitHub website and click the pencil button to edit it directly.



Add a line to our file - "this is a new line!".

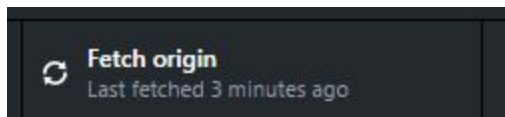
```
1  this is a test
2  this is a new line!
3
```

Just like when we make a change on GitHub Desktop, we also need to add a commit message when changing things online. Write "New line added to test file" in the commit message, and make the commit.

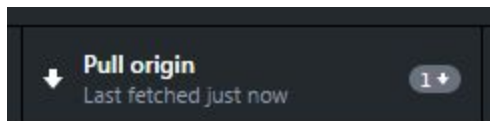


The image shows the GitHub 'Commit changes' dialog. It features a small avatar icon on the left. The main area has a text input field containing 'New line added to test file' and a larger text area below it with the placeholder 'Add an optional extended description...'. At the bottom, there are two radio button options: 'Commit directly to the master branch.' (which is selected) and 'Create a new branch for this commit and start a pull request.' with a link to 'Learn more about pull requests.'. Below the options are two buttons: a green 'Commit changes' button and a grey 'Cancel' button.








11. Pull Changes: Now head back to GitHub Desktop. You won't see your new change in the history yet. If you are lucky, you may now see the option to "Pull" your changes from the server ("origin") on the top right. This is because the application automatically checks the server to see if there are any updates every so often.



If you don't see "Pull", the button will say "Fetch" - click this to have the application check the server now. It should change to "Pull" after seeing there is a new change ready on the server.



Once you see "Pull", click the button to pull down your change to the text file. Now check your history and see the change has been added. Open up the text file to check and see that the new line has been added to it.

Changes	History	New line added to test file		
New line added to test file  Sarah Herzog committed a minute ago		 Sarah Herzog committed  06b1f7f  1 changed file		
Added test file.  Sarah Herzog committed a day ago		Test.txt		<div>@@ -1 +1,2 @@</div> <div>11 this is a test</div> <div>22 +this is a new line!</div>
Initial commit  Sarah Herzog committed a day ago				

12. Clean Up: Now that we've done our test, let's get rid of that text file and test what we've learned. Delete the text file, commit the change, and push to the server. Check on the GitHub website the text file is gone from the repo and the new commit is in the history.

That's it! You now know how to Clone, Commit, Push, Fetch, and Pull - the major operations for any version control system!

13. Further Learning: Git and GitHub can do a lot more than these basics. I suggest you look in to using .gitignore to ignore files that don't need to be backed up, such as intermediate files in the build process. We haven't covered how to best use Git for group projects, the branching and merging functionality, or reverting or rolling back to older commits. Have a read through some documentation online to get an idea of how these features work.