MIDDLESEX Community College

Tools and Technologies for Tech Writers 2020

Git Cheatsheet

Notices

This document was prepared as a handout for the Middlesex Community College Tools and Technologies for Technical Writers class, Winter semester 2022.

Prepared by Zoë Lawson, course instructor.

Contents

Formatting conventions

When describing code examples, this document uses the following conventions.

Format	Convention
`monospace text`	A command to run
<variable name>	Something that needs to be replaced with an actual value. For example, *<your name>* should be replaced with whatever your name is.
`code to run`	An example command that you should enter at a command prompt (or GitBash window)
`messages and stuff`	An example of what a command prompt returns, such as status or error messages

Command Line

A short reference of common (and uncommon) Git commands.

Common Git Commands

Here are some of the basic commands you use with Git daily.

Command	Description	Specific examples

`git pull` *<remote repo> <branch>*	Get the changes from the remote repository into the branch.
	You can use a URL or a nickname for the *remote repo*. Use a URL to get from the repository you forked (e.g. ZoeLawson/mcc_tools_tech). Use `origin` for your repository.
		Get changes from the repository you forked from into your local Winter2020 branch
		`git pull https://github.com/ZoeLawson/mcc_tools_tech.git Winter2020`
		Get changes from your fork in GitHub into your local Winter2020 branch
		`git pull origin Winter2020`

`git status`	Get the current status of your local repository.	`git status` There are a lot of different results from the `git status` command. See Git Status Results on page 8.
`git add` *<files or folder>*	Add or stage the changed files in your local git repository. You can name specific files or folders, or just use `.` to indicate all the files in this directory and all sub-directories. This command is relative to the directory you are in. If you are in a "Homework" directory, it will only stage the files in the Homework directory.	Stage all the files you've changed. `git add .`
`git commit -m` *"Witty comment here"*	Saves the staged files to your local repository.	`git commit -m "Week 3 homework"`

Command	Description	Specific examples

		`[Winter2020 1ac2849] in progress`
		`1 file changed, 116 insertions(+), 3 deletions(-)`
		
		The first line gives you the name of the branch, the ID number of the commit, and the commit message.
		
		The second line lists technical details about the changes.

`git push` *<remote repo>* *<branch>*	Get changes from your local repository to the remote repository.	`git push origin Winter2020`
		
		`Enumerating objects: 18, done.`
		`Counting objects: 100% (18/18), done.`
		`Delta compression using up to 8 threads`
		`Compressing objects: 100% (15/15), done.`
		`Writing objects: 100% (15/15), 3.33 KiB | 1.67 MiB/s, done.`
		`Total 15 (delta 10), reused 0 (delta 0)`
		`remote: Resolving deltas: 100% (10/10), completed with 3 local objects.`
		`To https://github.com/ZoeLawson/mcc_tools_tech.git`
		`6259a47..1ac2849 Winter2020 -> Winter2020`
		
		The results are a lot of technical details about the data being moved from your local repository to the remote repository.

Uncommon Git Commands

Here are a few more git commands you use occasionally.

Command	Description	Specific Example
`git clone` *<remote repository URL>*	Make a copy of a Git repository on your local system in such a way the two repositories know about each other.	`git clone https://github.com/ZoeLawson/mcc_tools_tech.git` This command clones the ZoeLawson/mcc_tools_tech repository to your local system. You should use the fork you created. The URL would include your GitHub user name instead of "ZoeLawson".
`git checkout` *<Branch Name>*	Switch between branches. By default, Git always starts with the master branch. In general, you don't want to work in the master branch. In this class, we will constantly be working in Winter2020, so you only need to change branches once.	`git checkout Winter2020` This changes your local branch to the Winter2020 branch.

`git fetch`	When you clone a repository, you get all the current information about that repository. If people add more branches to the remote repository, your local repository doesn't know about it, until you run a `git fetch`.
	This is similar to the `git pull` command. However, the pull can only get branches the local repository knows about.
		`git fetch`
		`remote: Enumerating objects: 14, done.`
		`remote: Counting objects: 100% (14/14), done.`
		`remote: Total 35 (delta 14), reused 14 (delta 14), pack-reused 21`
		`Unpacking objects: 100% (35/35), done.`
		`From https://github.com/ZoeLawson/mcc_tools_tech`
		`cdff0ee..43f1608 Winter2020 -> origin/Winter2020`
		`8fb1229..4cdfe9a master -> origin/master`

Git Status Results

`git status` is a command that helps you understand what's happening in your local Git repository.

There are no changes locally:

```
On branch working
nothing to commit, working tree clean
```

The first line gives you the name of the branch you are on. The second line tells you there's nothing to do.

If you have files changed:

```
On branch working
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Week02-ProgressiveInfo/week2_progressiveinfo.html
```

The first line tells you which branch you are on. The Changes not staged for commit lists all the files that have changed, but are not added. The modified files are usually in red text. The text in parenthesis are suggested commands to help you.

If you have files staged (i.e. you've run `git add .`):

```
On branch working
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   handouts/usingGit/git_cheatsheet.ditamap
        new file:   handouts/usingGit/git_commands.dita
```

The first line tells you which branch you are on. The Changes to be committed lists the files you have added/staged, but have not officially told Git to remember. The staged files are often green. The text in paraethesis are suggested commands to help you.

Note: When you're doing a lot of work, you may see both unstaged and staged files when you run the `git status` command.

If you've committed files, but haven't pushed them:

```
On branch working
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

The first line tells you what branch you're on. The next line tells you that you have changes locally that you haven't pushed up to the remote repository (aka origin)

GitHub Desktop

There are many different applications you can use to interact with Git besides the command line. One option is the GitHub Desktop.

You can download GitHub Desktop from https://desktop.github.com/.

The help for GitHub Desktop is available from https://help.github.com/en/desktop.

Clone Repository

This is generally a one-time task to make a copy of the remote repository in GitHub on your local system.

You must have the repository in your GitHub account before you begin. This can be a new repository you create for your own project, or a fork of an existing project.

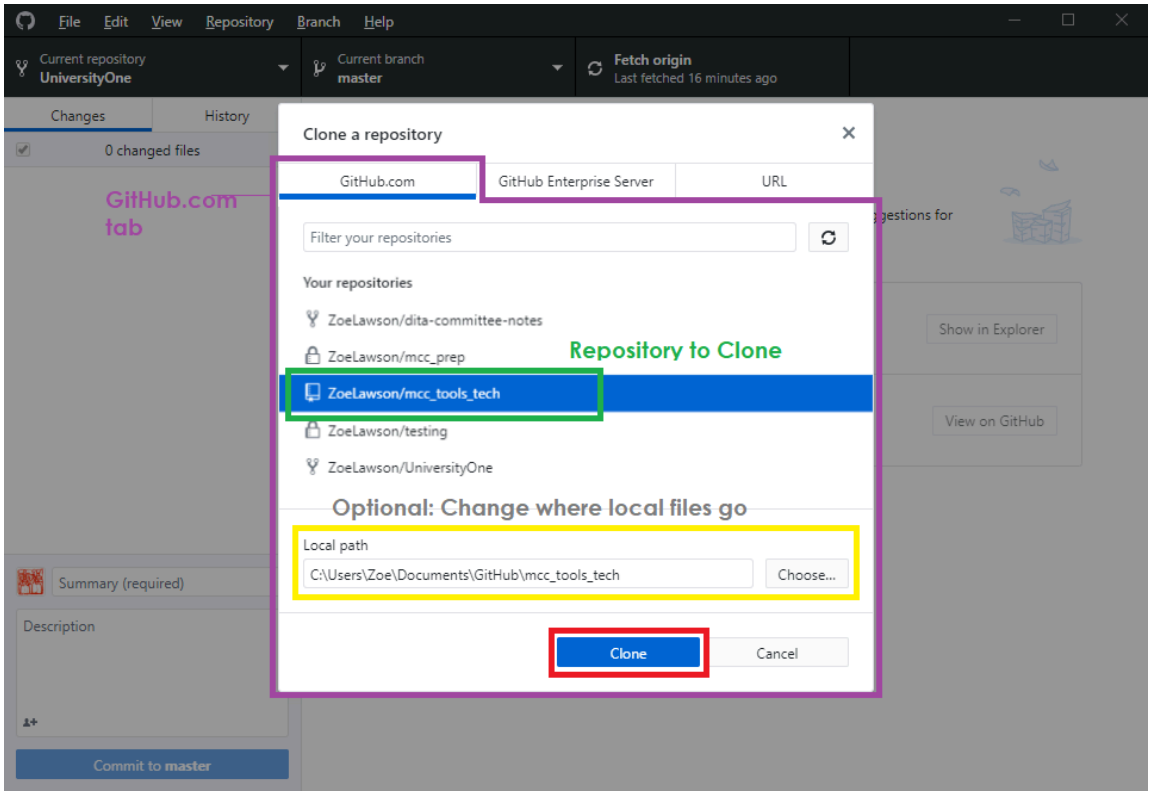
In this class, you will be working with a forked repository. See Fork a GitHub repository on page 17.

1. Select **File** > **Clone repository**.

2. Select the repository you want to clone.

   For this class, you want to select your fork of the mcc_tools_tech repository on the GitHub.com tab.

   This example is from my account, so the name is ZoeLawson. Your version of GitHub Desktop should show your GitHub account name.



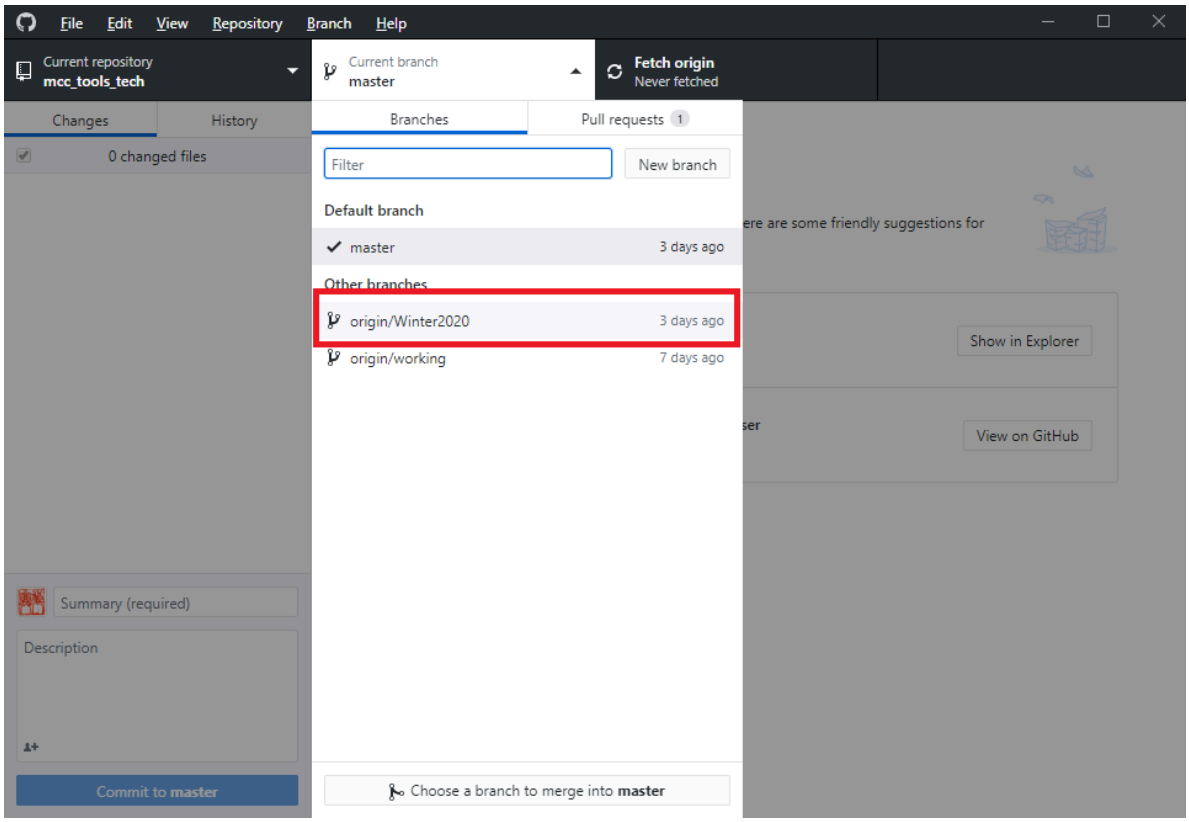
3. Change where the files are stored locally by changing the **Local path**.
4. Click **Clone**.

Change Branch

By default, Git starts in the master branch after you clone a repository.

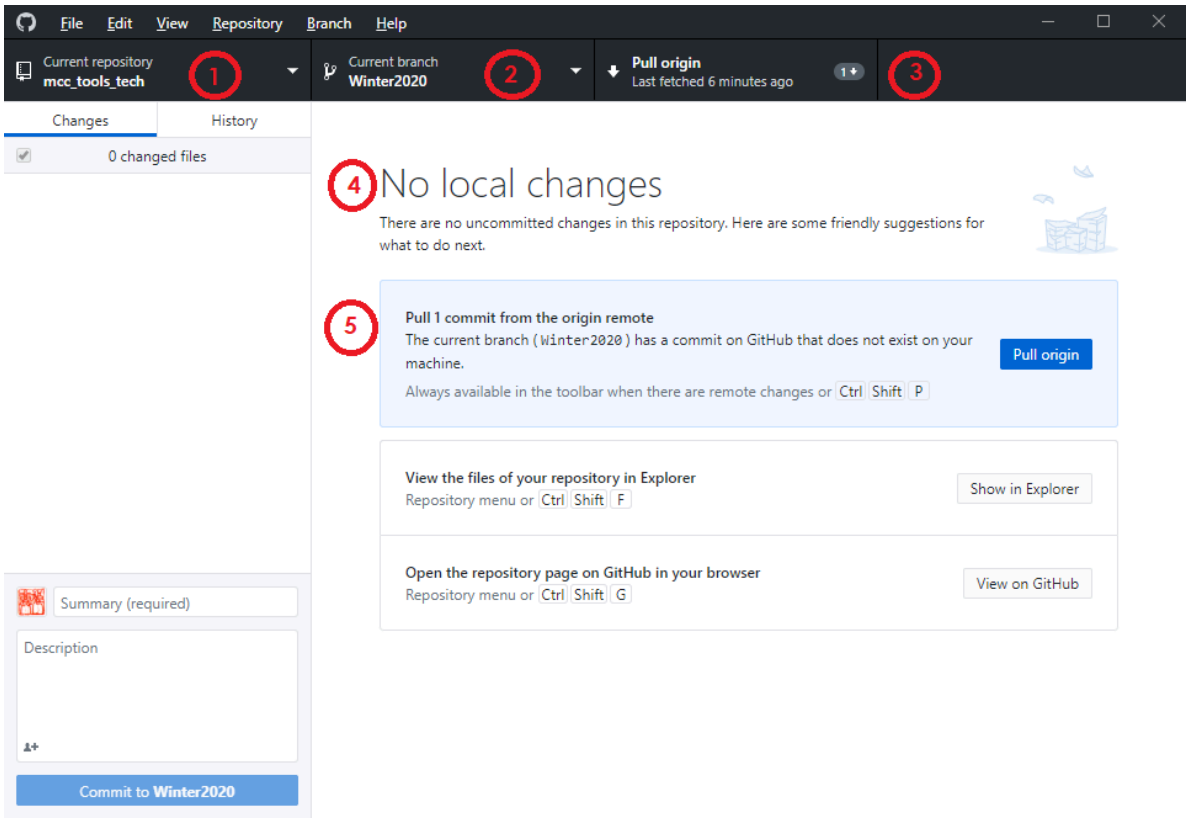
In this class, you want to work in Winter2020.

In GitHub Desktop, select **origin/Winter2020** from the drop-down under **Current branch**.



A few notes on GitHub Desktop

A few tips on what the GitHub Desktop is trying to show you. This screen shows if you have no local changes.



1. The name of the repository you are currently looking at. For this class, you only need to point to one repository, mcc_tools_tech. If you play around on GitHub, you may have many more.
2. The name of the branch you are currently working with. This should be Winter2020 for this class.
3. The last button changes as the status of the remote repository changes.
   - You can **Fetch** changes from the remote repository (called "origin"). This is how GitHub Desktop figures out if there are changes available in remote repository.
   - You can **Pull** updates from the remote repository (called "origin") down to your local system. This is how you get updates from the remote repository. (This is different from syncing a forked repository. See Sync a forked repository on page 15.)
4. Gives you the status of local changes on your system.
5. Suggested command based on the current file situation. This example shows that there are changes in the remote repository you should get locally. Therefore, it recommends you **Pull origin**. If you had local changes, it suggests to **Push** files to origin.

Pull changes from remote repository

If there are changes to the remote repository, you need to pull them into your local repository to make sure you have the latest and greatest changes.

This is different than syncing a forked repository. See Sync a forked repository on page 15.

GitHub Desktop offers three different methods to pull changes into your local repository. At least one of these options is always available. If you can't find the blue **Pull origin** button, you can use the **Repository** menu command.

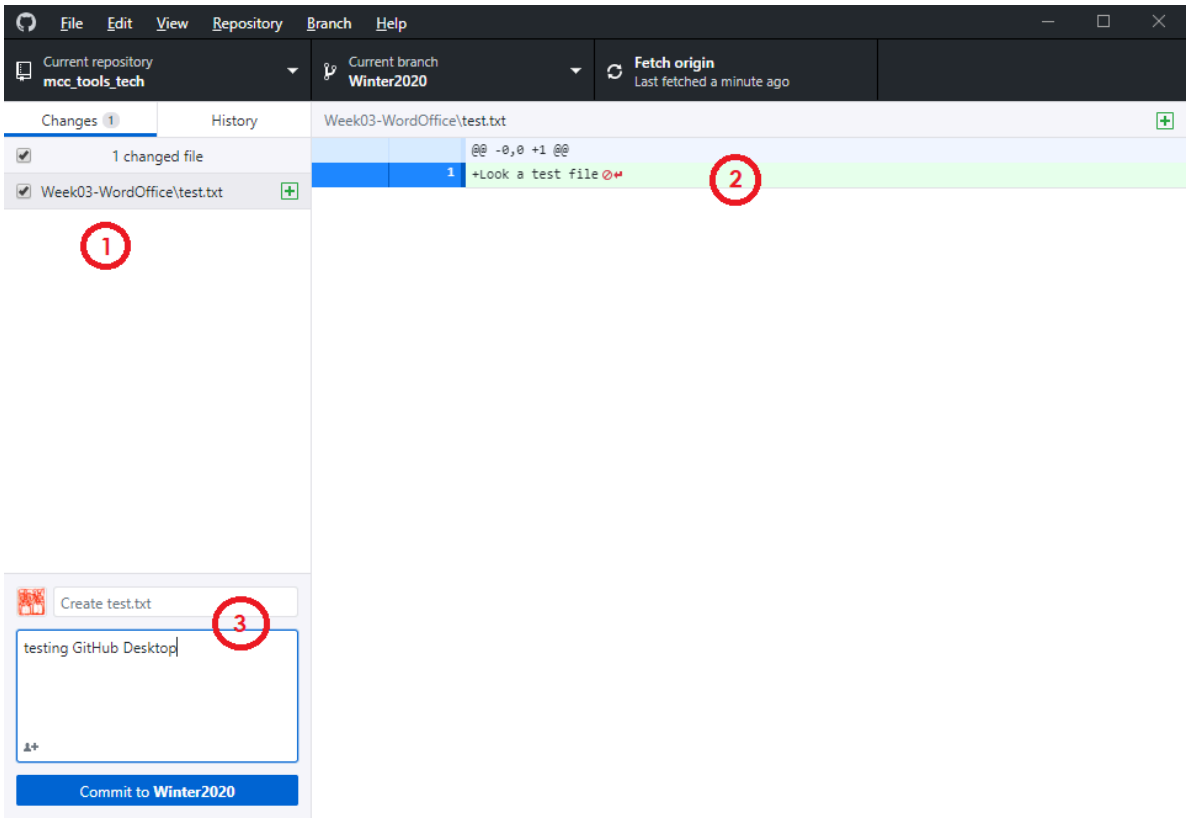
- Click the blue **Pull origin** button.
- Click the black **Pull origin** button.
- Select **Repository** > **Pull**

GitHub Desktop updates your local files with the latest and greatest files from the remote repository (origin).

If there are no changes to pull, the buttons may say **Fetch origin**. After you click **Fetch origin**, if there are changes, the buttons should change to **Pull origin**.

Commit changes

GitHub desktop automatically determines when you've added or changed files in your local copy of the repository. You can easily Commit the changes to your local repository.



1. The Changes panel lists all the files that have changed. There will be different indicators for new , changed , or deleted files.
2. If the file is a type of file GitHub Desktop knows how to handle, such as a text file, it will show the contents of the file and mark the changes.

3. Where you can provide a summary of changes (in this example `Create test.txt`) and optionally, provide more details (in this example, `testing GitHub Desktop`).

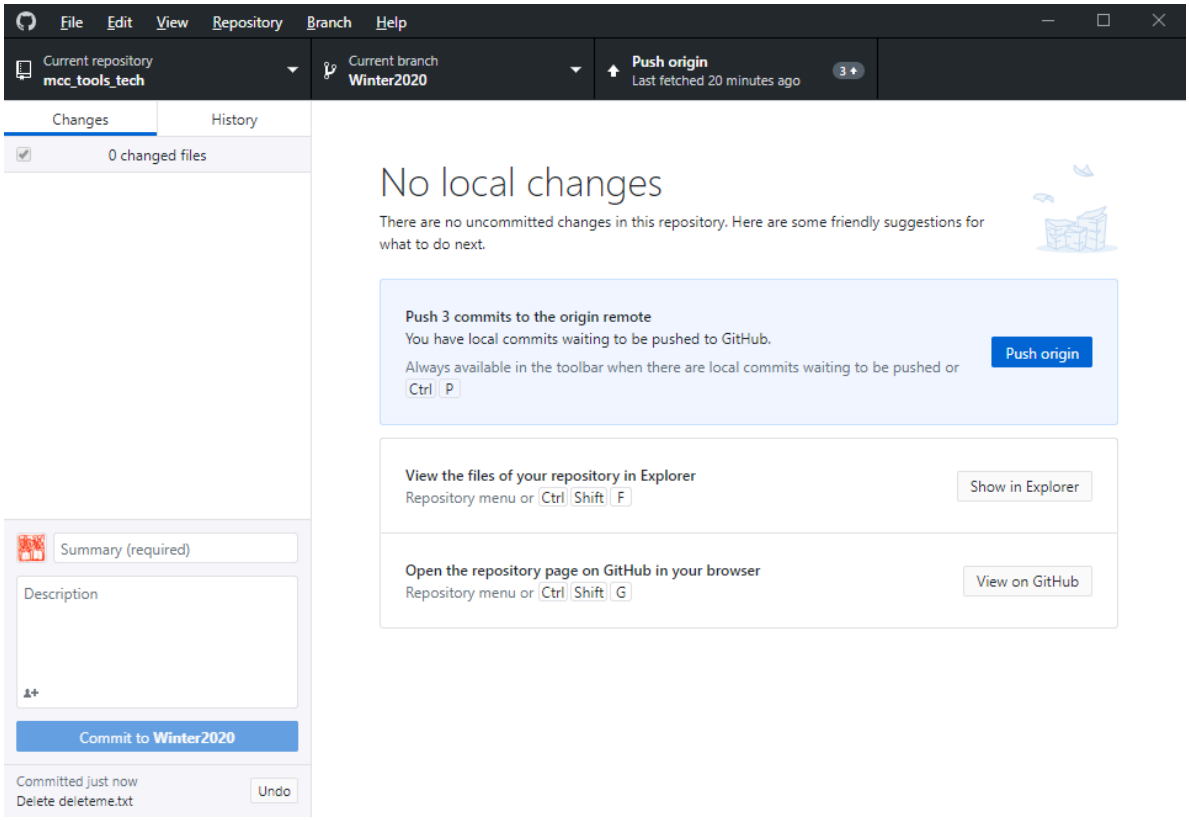
Use GitHub Desktop to commit changes to your local repository.

1. Make some change to your local copy of the repository.

   In this example, created a new file called `text.txt` with the content `Look a test file`.

   When you go to GitHub Desktop after making a change to your local files, you will see the changed files listed in the Changes panel. As you select the files in the Changes panel, their contents display in the right panel.
2. Provide a summary of the change. You can also add more details in the larger text box.
3. Click **Commit to Winter2020**.

Your changes are commited to the local repository.



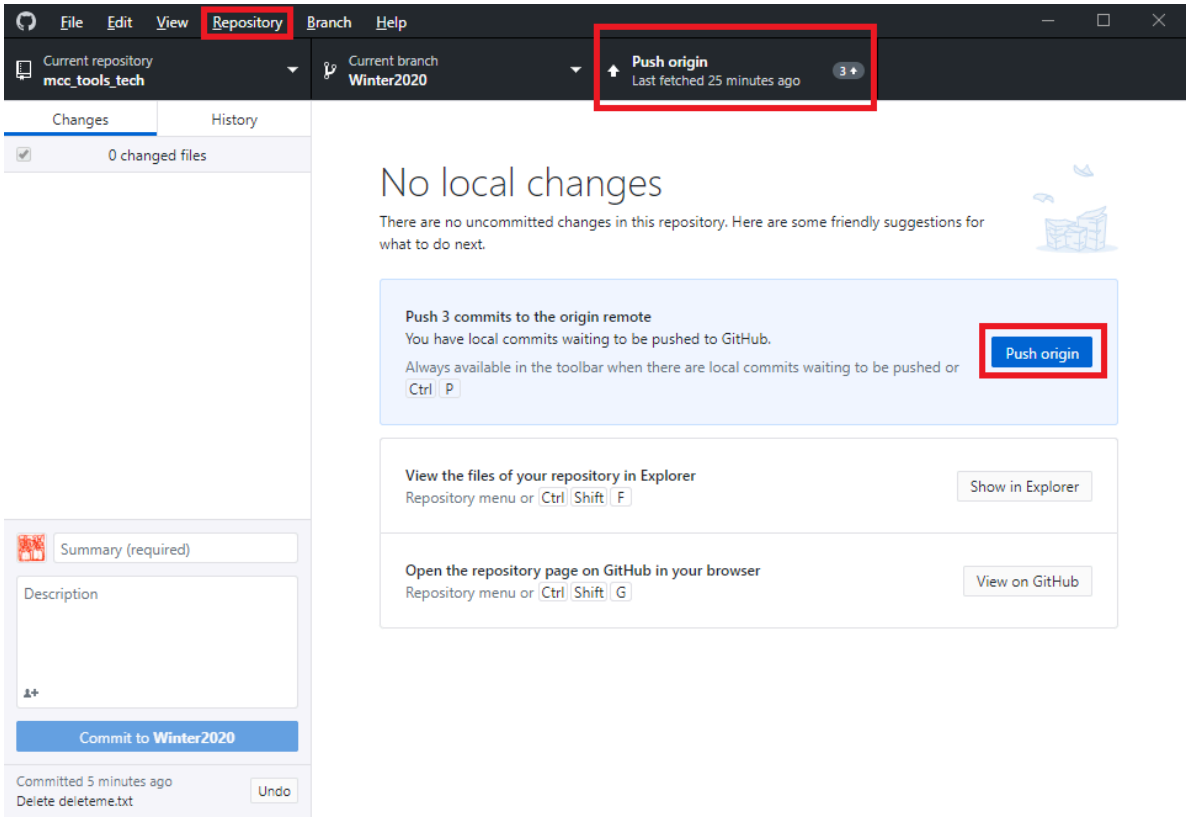
You can now Push changes to remote repository on page 14.

Push changes to remote repository

After you've committed changes to your local repository, you need to push them to the remote repository.

You usually need to pull any changes into your local repository before you can push changes to the remote repository. See Pull changes from remote repository on page 12.

GitHub Desktop provides three different methods to push content.



- Click the blue **Push origin** button.
- Click the black **Push origin** button.
- Select **Repository** > **Push**.

Whichever method you select, the files are pushed to the remote repository (origin).

You are now ready to make a pull request to get your changes in the remote repository to the original forked repository (ZoeLawson/mcc_tools_tech). See Make a pull request on page 17.

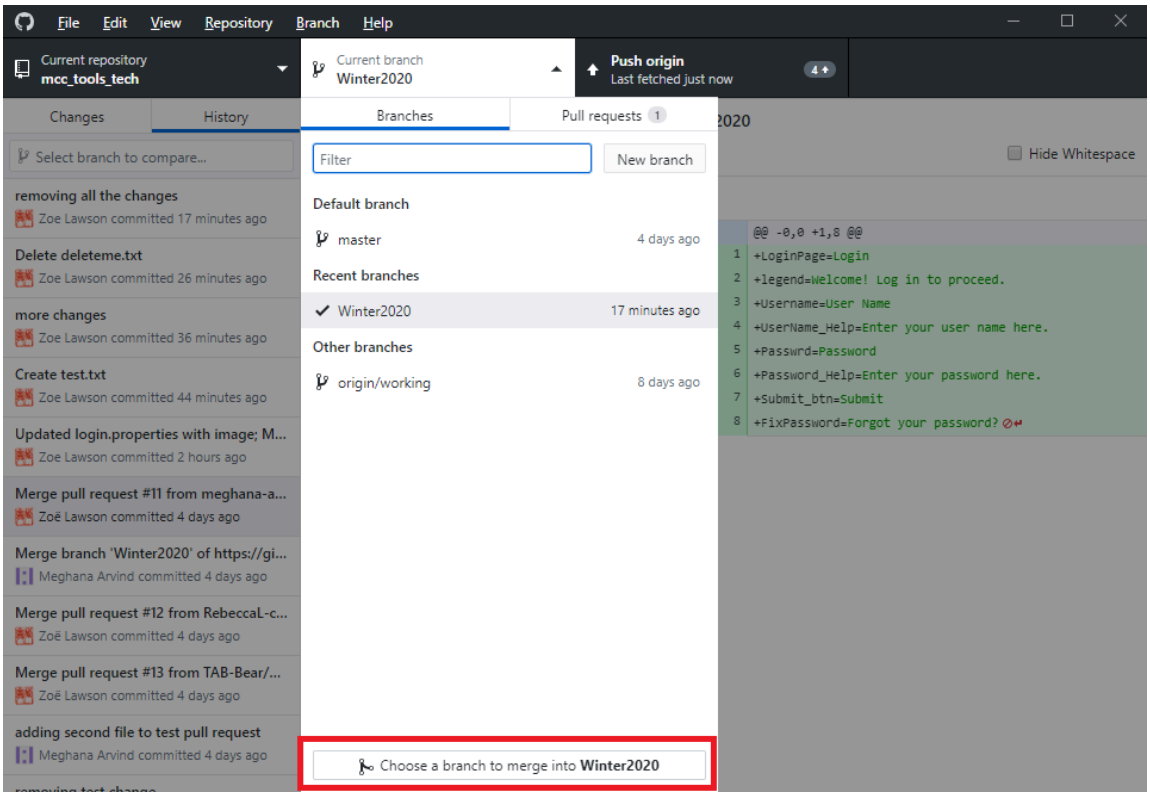
Sync a forked repository

You forked the ZoeLawson/mcc_tools_tech repository. When I make changes to the original ZoeLawson/mcc_tools_tech repository, you need to sync the files to get the changes into your forked version of the repository.

For example, the Week 4 homework is not yet availalble. When I update the ZoeLawson/mcc_tools_tech repository with the Week 4 homework, you need to sync the repositories to get the changes into yours to do the homework.

Unfortunately, GitHub does not provide an "easy" sync button. Nor does the user interface use any words like "sync" to help guide you. It follows the Git concept that you are merging changes in from a different branch in a different repository.

1. Select the drop down for the **Current Branch** and select **Choose a branch to merge into Winter2020**.



2. Select **upstream/Winter2020** from the Other branches.
3. Click **Merge upstream/Winter2020 into Winter2020**.

   This gets the changes in the "upstream" (the original repository you forked) into your local repository.

   You can check your local files to confirm there are some new changes. For example, look for homework from other students or the next week's homework.
4. Click **Push origin** to get the changes into your remote repository.

GitHub Web Interface

The GitHub web interface is a quick way to look at your repositories in GitHub. You can browse files, make pull requests, and gather other important information from the GitHub web interface.

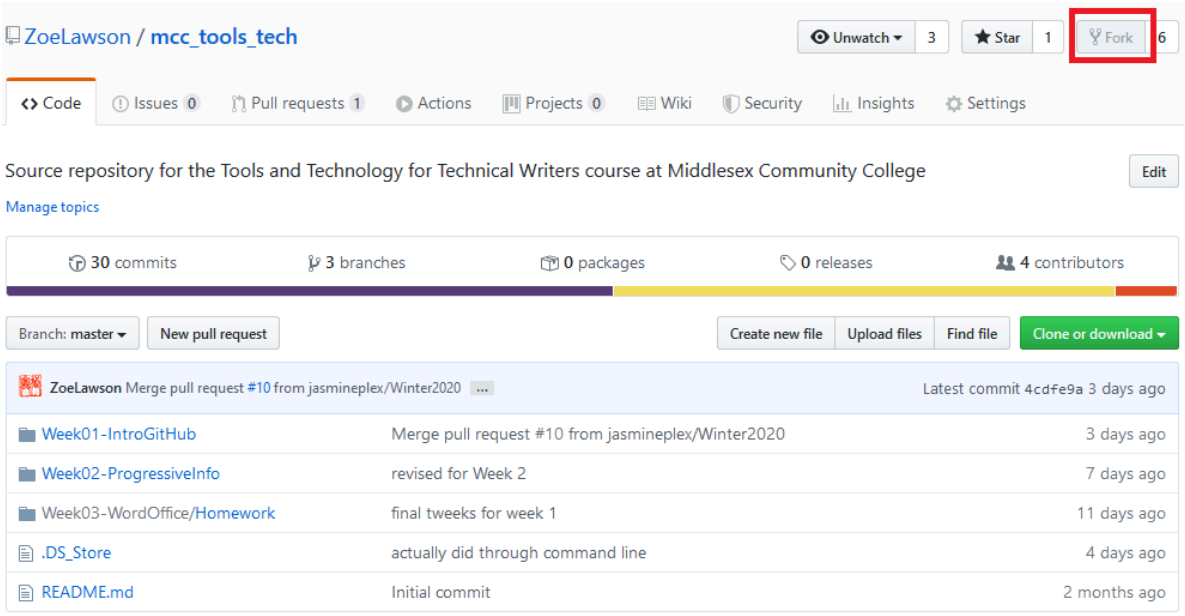
There are a ton of features in GitHub that this class is not using. You can use the Git Help at https://help.github.com/en to learn more.

Fork a GitHub repository

Fork the mcc_tools_tech repository into your repository so you can make changes.

"Fork" is a concept in GitHub. It is not a basic command in Git. In GitHub, you may not know the owner of the content you want to work with. You may not be able to be a contributor to that repository. So you "fork" or make a copy of the repository in your GitHub account. You can then work in the fork, and make a pull request into the original repository.

Go to https://github.com/ZoeLawson/mcc_tools_tech and click **Fork**.



Make a pull request

When you have changes in your remote repository you want to get to the original forked repository, you make a pull request. You are asking the owner of the original repository to pull your changes into their repository.

1. Go to the fork of your repository in the GitHub web interface.
2. Make sure you are in the **Winter2020** branch.
3. Click a **New pull request** button.

4. Confirm that the **base** is the original repository (ZoeLawson/mcc_tools_tech) and the **compare** is your branch in your fork (*Your GitHub Account Name*/mcc_tools_tech).
5. Add some description of the changes you're requesting to have merged in.

   The more descriptive you are, the better for troubleshooting later. In the future, you may not know the person reviewing the pull request. Even if you know the person, they may not be intimately aware of whatever you're working on.

   I review the pull requests for my team of writers. All of our books are in a single repository. We've had issues with bad push/pull practices in the past where folks have overwritten other people's work. Therefore I try to check if coworker A's pull request only contains files for the books they're currently working on. But, I don't always know everything they're working on. So I ask that people include the book they're working on in the description. Therefore, if they say they're checking in the user guide, but I see changes to files in the installation guide, I can stop and ask questions and try to avoid disaster.
6. Click **Create Pull Request**.

The owner of the original repository gets an email indicating that there's a pull request. You can also see it on the Pull Request tab in the GitHub web interface.. I will then review your request and most likely merge the change. You should receive an email when the request is merged, or if I comment on it.

Sync a forked repository

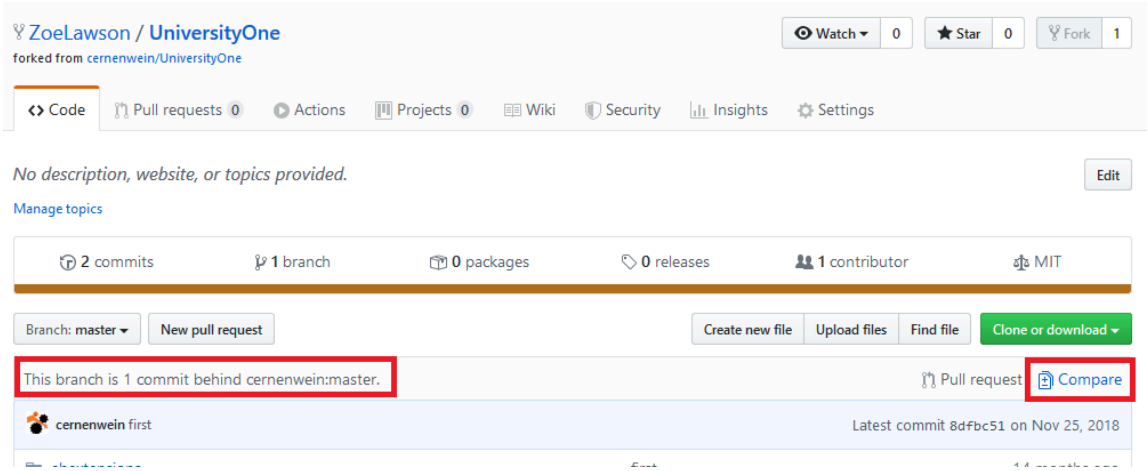
You can also sync a forked repository using the GitHub web interface.

See Sync a forked repository in GitHub Desktop for more information about what syncing means.

These options may only be available if you can see a message in GitHub along the lines of `This branch is n commits behind ZoeLawson/Winter2020`.

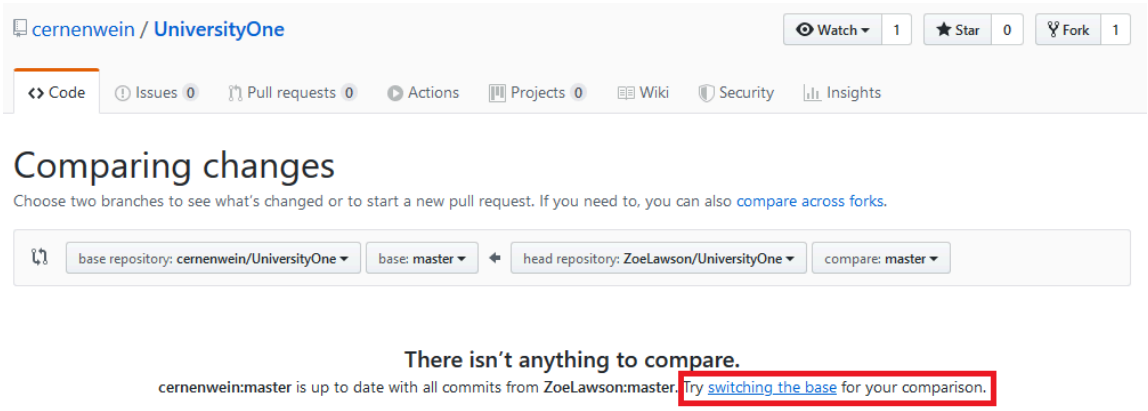
Note: Because I cannot fork my own repository, I cannot make screen shots of how it will actually look for you. I forked a different repository for my images.

1. From the Code tab of GitHub, click **Compare**.



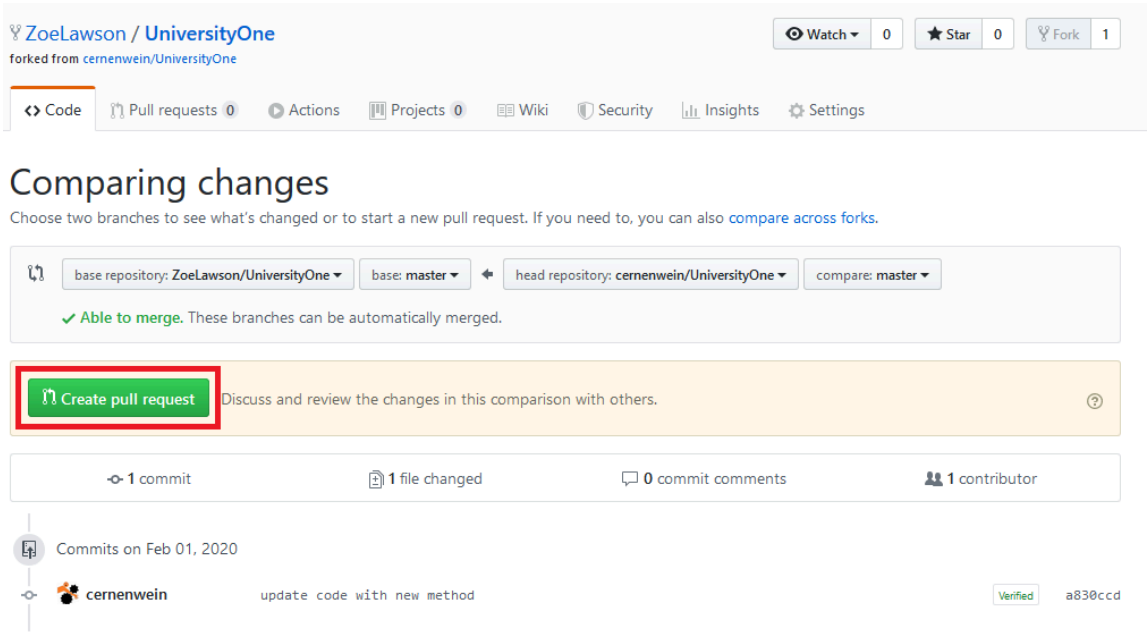
   Instead of ZoeLawson/UniversityOne, it should use *Your GitHub Account Name*/mcc_tools_tech. Instead of cernenwein:master, it should be **ZoeLawson:Winter2020**.

2. Click **switching the base**.



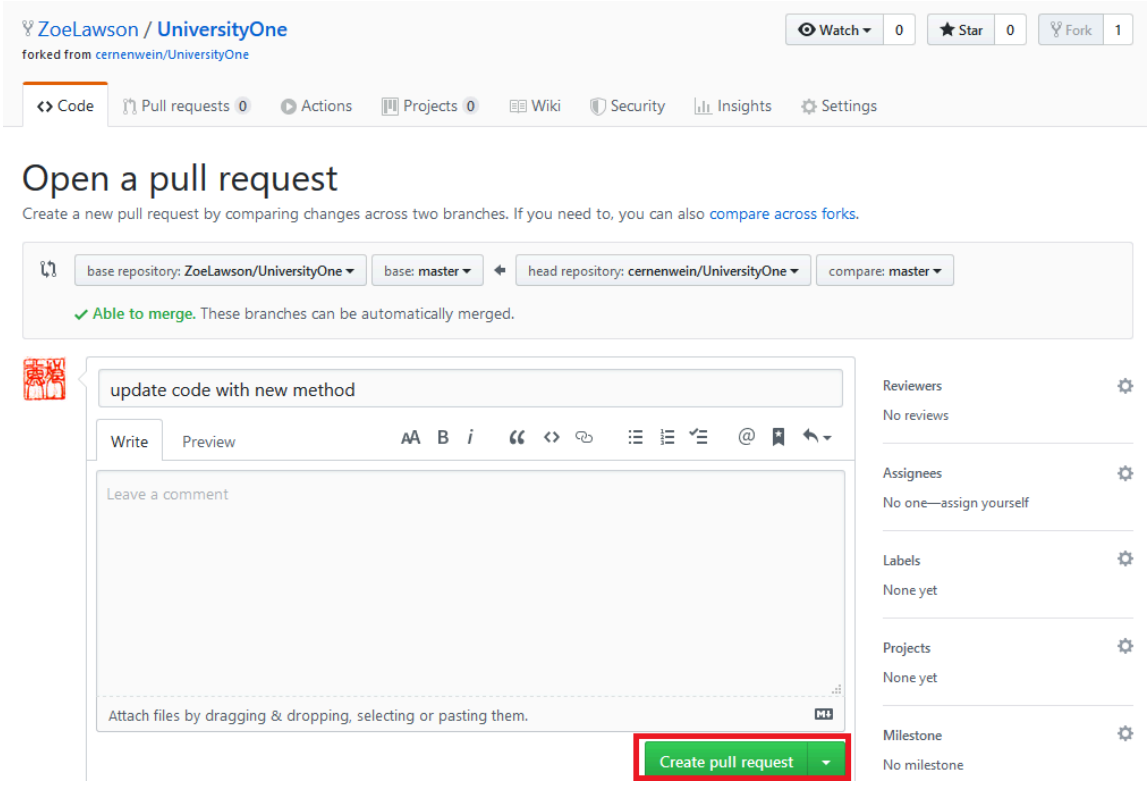
This will change the flow of information from the forked repo (*Your GitHub Account Name*/ mcc_tools_tech) to the original repo (ZoeLawson/mcc_tools_tech) to the opposite. Also, your branch to compare will be Winter2020, not master.

3. Click **Create pull request.**



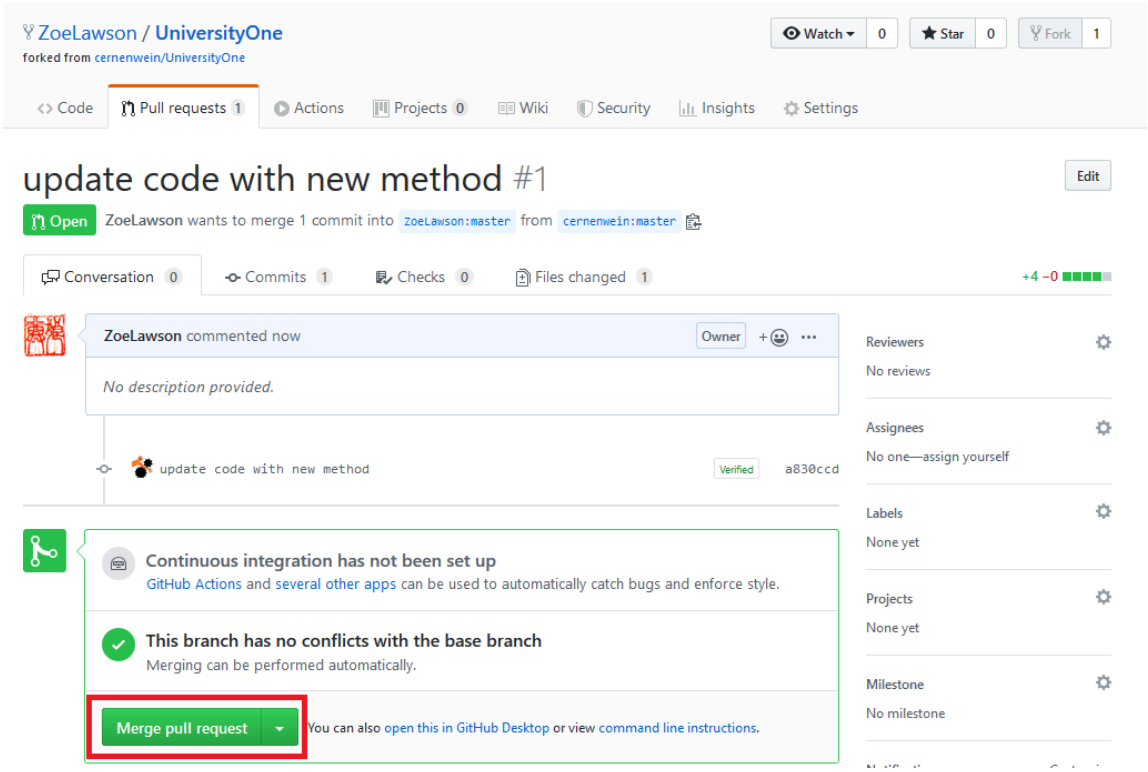
As you can see, the base repository is now the forked repository, and the head repository is the original repository. The arrow shows the data will be flowing from the original to the fork.

4. Provide some sort of summary. You can usually accept the default because it's the same commit messages from the original repository. Then click **Create pull request**.

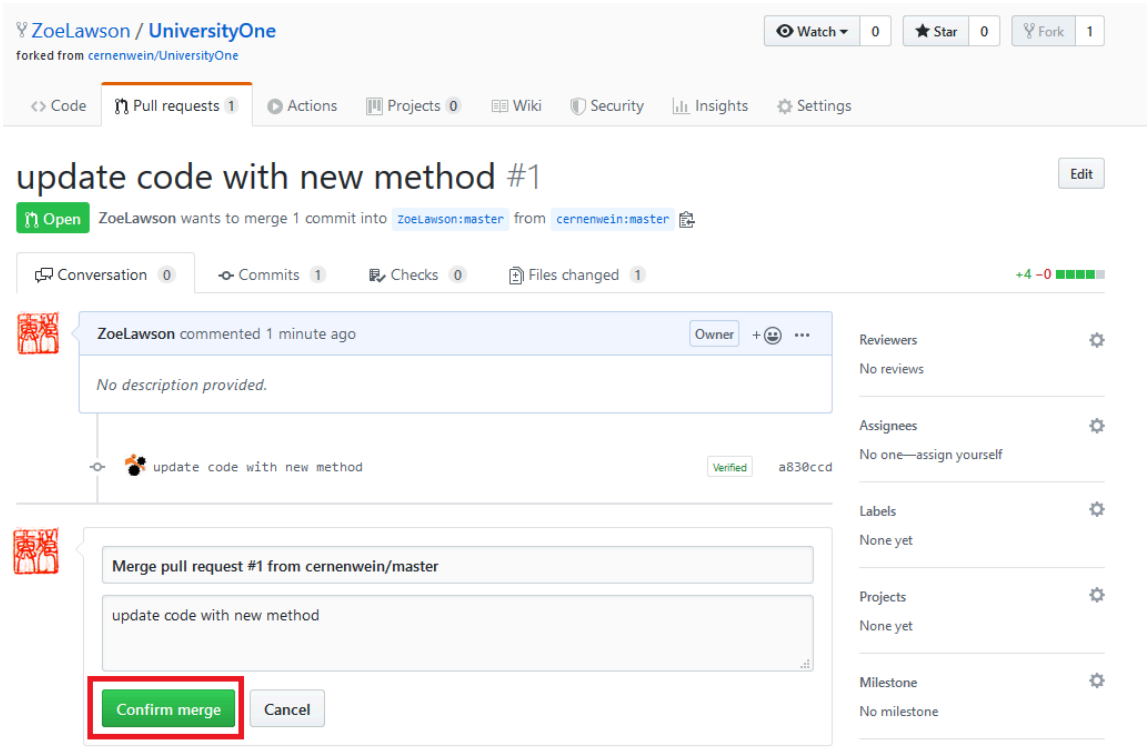


Yes, you are creating a pull request into your own repository.

5. Click **Merge pull request**.



6. Click **Confirm merge**.



These steps will merge the changes from the original repository into your remote repository. You will then need to Pull changes from remote repository on page 12 to get the files to your local system.