CS 1302 – Lab 4, GitHub

Last update: 9.2.22, 12:46 pm

Contents

[1 Introduction 1](#_Toc78886525)

[2 Create a Repo & Upload Code 1](#_Toc78886526)

[3 Subsequent Upload to GitHub 5](#_Toc78886527)

[4 Delete a file from GitHub 7](#_Toc78886528)

[5 Branching 8](#_Toc78886529)

[6 Tips 10](#_Toc78886530)

[7 Submission 10](#_Toc78886531)

To make this document easier to read, it is recommended that you turn off spell checking and grammar checking in Word:

1. Choose: File, Option, Proofing
2. At the very bottom, check: “Hide spelling errors…” and “Hide grammar errors…”

# Introduction

Almost all software development is collaborative. It is the age-old engineering design principle: divide and conquer. Each team member works on different classes that are dependent one another. Thus, for an efficient and accurate development, a team must have a central location to store their code. We call such a location a *repository.* GitHub is the most widely used website for hosting and maintaining repositories (repos) of code.

This Lab will introduce you to GitHub. However, I don’t require the use of GitHub in this class. The idea is to introduce it to you now and hopefully you will either use it for this class and/or for side projects. You will have a much fuller introduction in CS 4321 where you will have 12+ Labs on Git[[1]](#footnote-1) and GitHub.

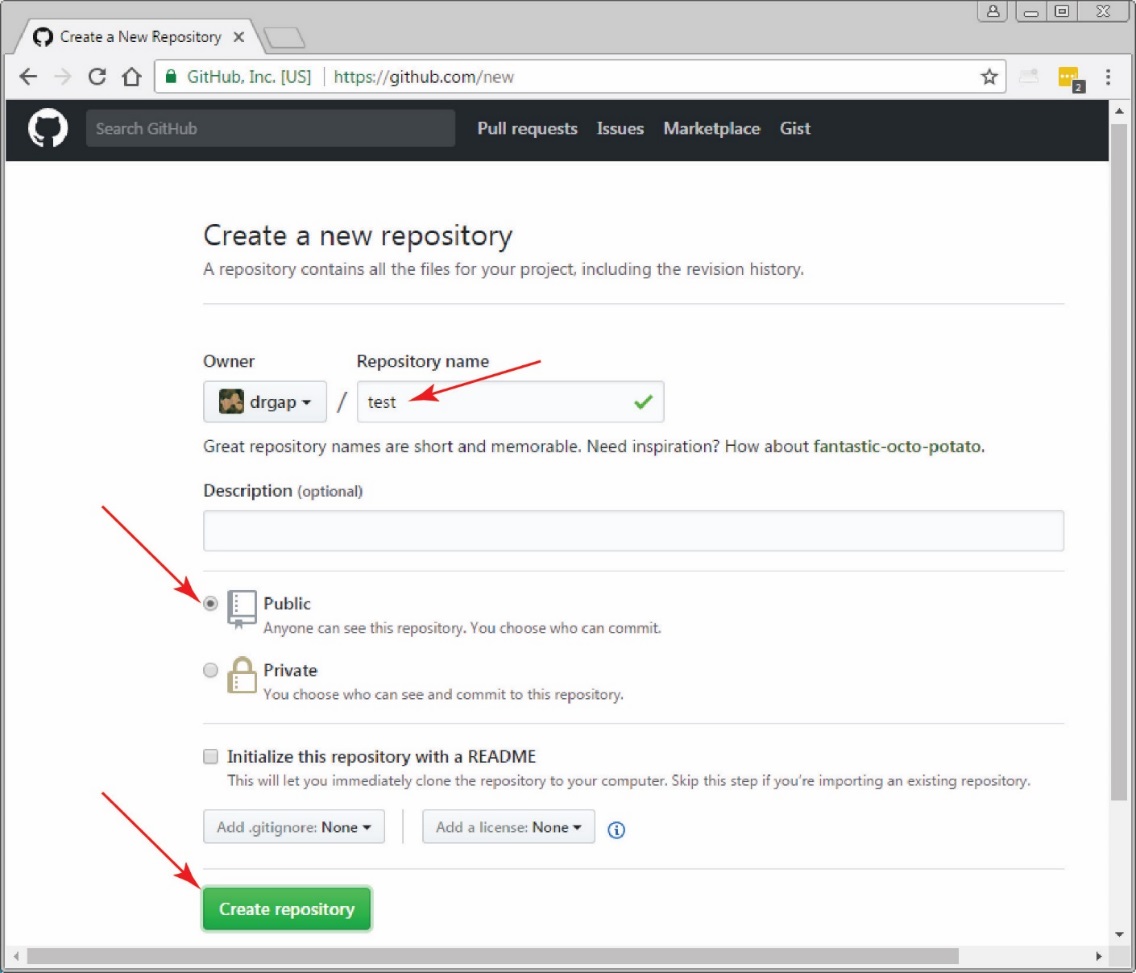
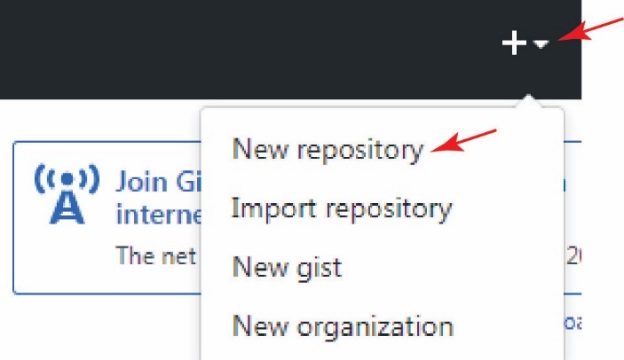
Do the following:

1. Create a GitHub account. You will create a username, a password, and specify your email address. [https://github.com/](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2F&data=02%7C01%7Cdgibson%40valdosta.edu%7C4dad75e3ebdc46ab47e208d7d03041a0%7C25a5d3408abc4053b4bddc1213280353%7C0%7C0%7C637206777989958306&sdata=PZKNTqsemFRE6xTle2Uu3aBxyyQYOdF2ToYFrtq3sUo%3D&reserved=0)

All repos are public. Anyone can see them (you have to pay to have private repos). Many companies will look at your GitHub repos when you apply for a job, no matter whether you have listed your *id* on your resume or not. Thus, I suggest creating a username that is not too far out, controversial, etc. Just something innocuous but meaningful to you.

1. Copy the *ver1* and *ver2* packages from your Lab 3 to a different location. You just need the two packages.

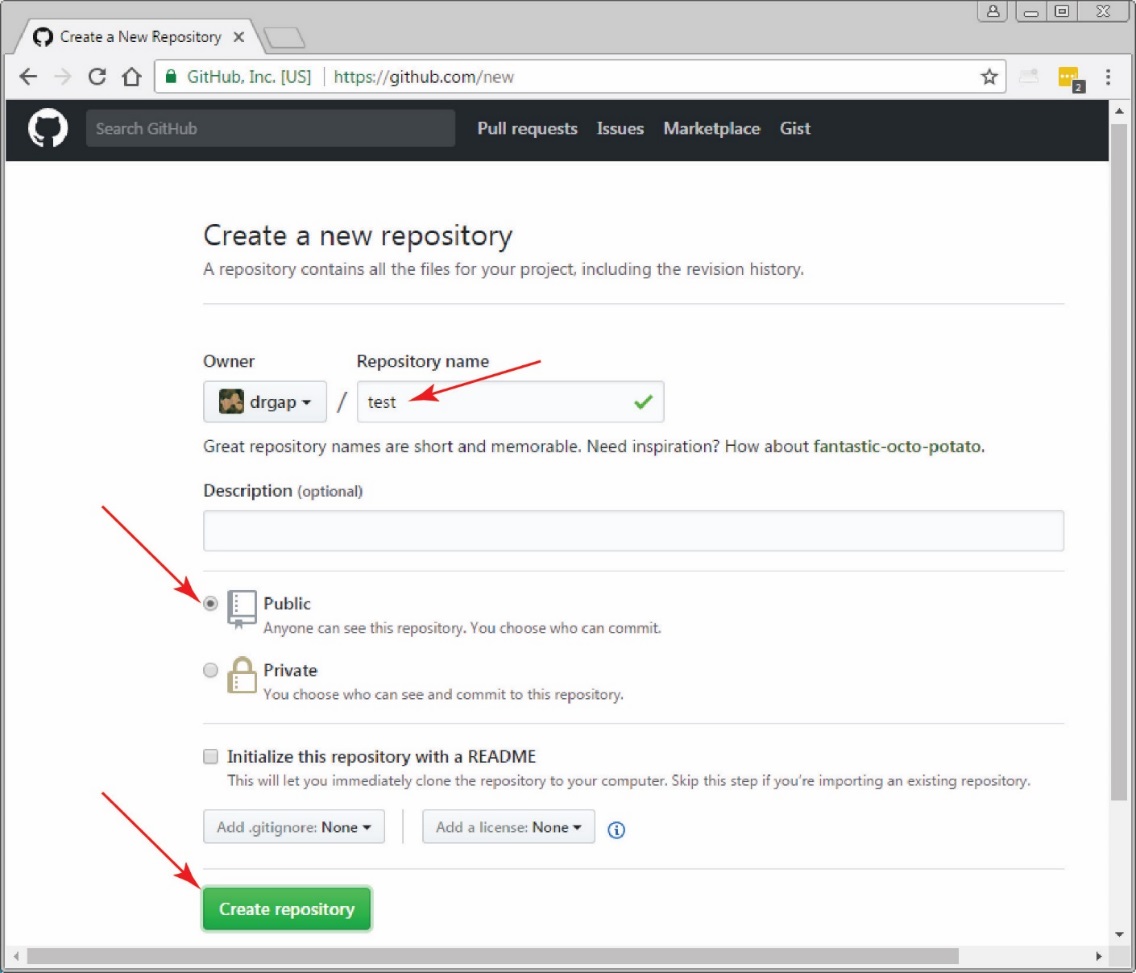
# Create a Repo & Upload Code

In the figures that follow, I’m using my personal GitHub account with the user name: *drgap,* and the gravatar:  which you will see in a few figures below.

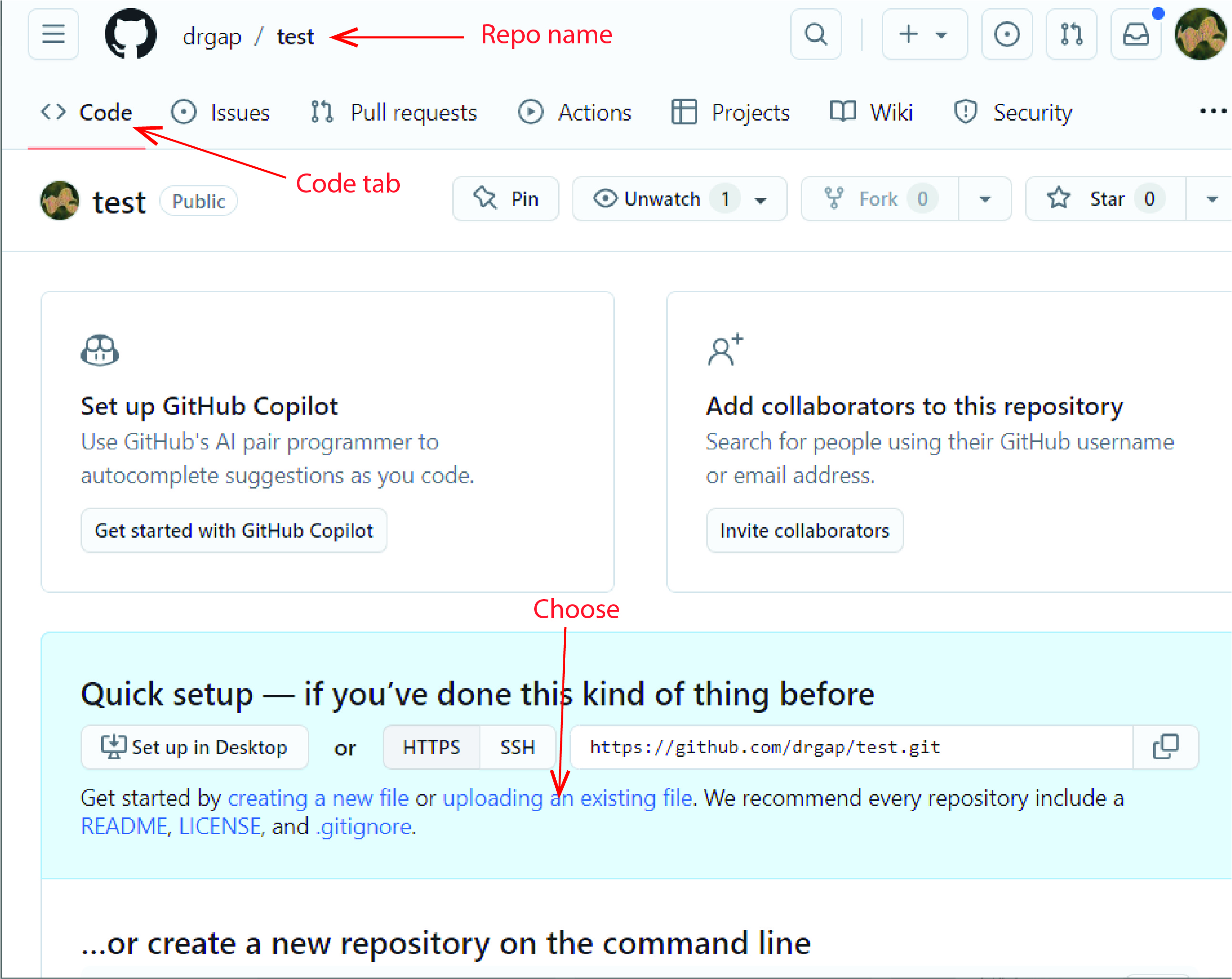
1. Follow these steps to create a new repository.
2. Login to GitHub
3. As shown in the figure on the right, select the drop-down beside the “+” sign in the upper right and choose, “New Repository”.

**Or, you may just see a “New” button, use that in this case.**

1. As shown below, name the repository, “test” and then choose “Create repository”

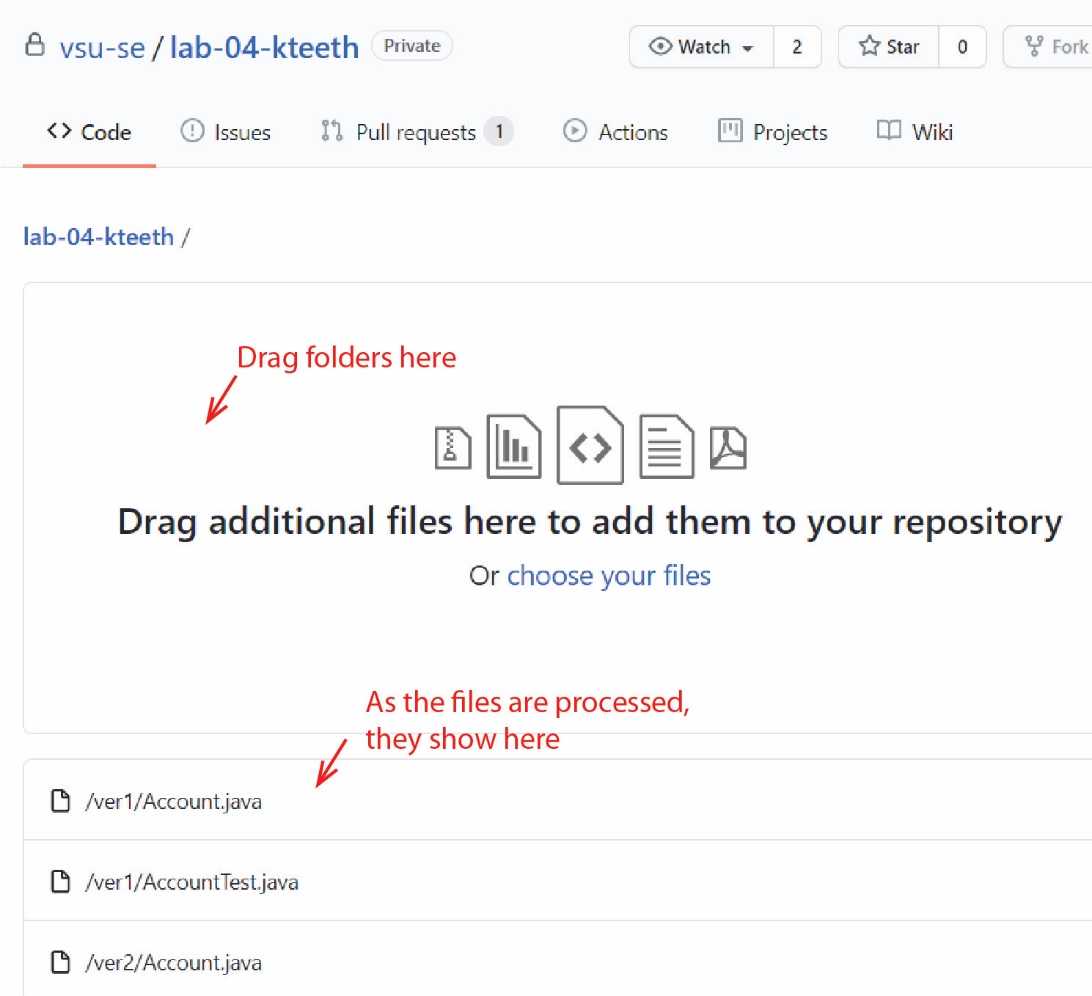


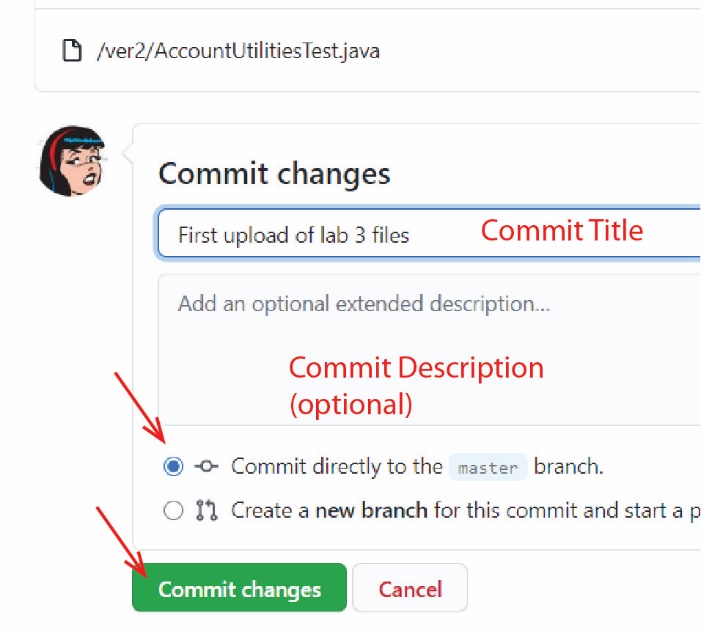
1. Upload some code to the repo
2. Your repo is shown as in the figure below with the Code tab active. Choose: Add file, Upload files.



**In the figures that follow, I’m now using a sample student that I made up, that has real gmail and GitHub account. Her name is, “Katie Teethtone”, and her GitHub account name is *kteeth*. You’ll also see her gravatar, which is an image of Veronica from the Archie comics. Sorry for the confusion, this lab was pieced together from other artifacts.**

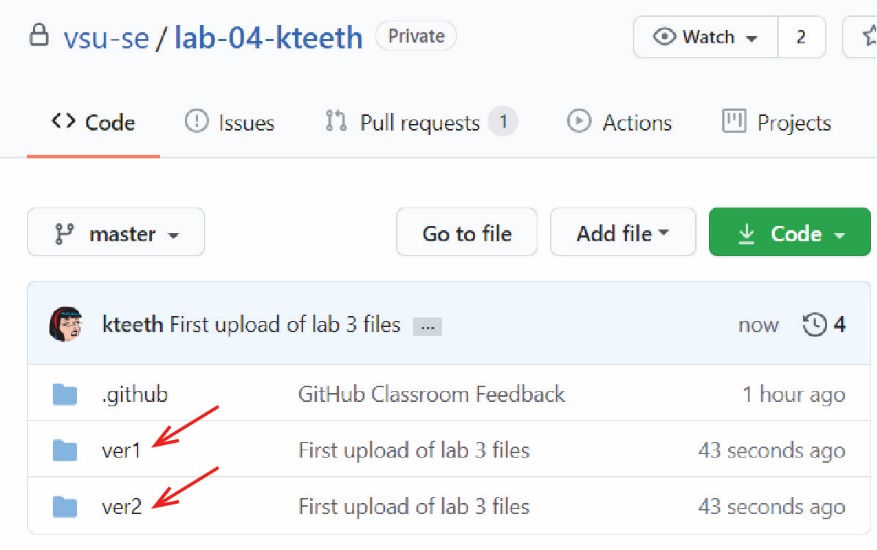
1. Do the following as indicated in the figure below:
2. Drag the package folders from Lab 03 (*ver1* and *ver2*) into the region provided for upload.



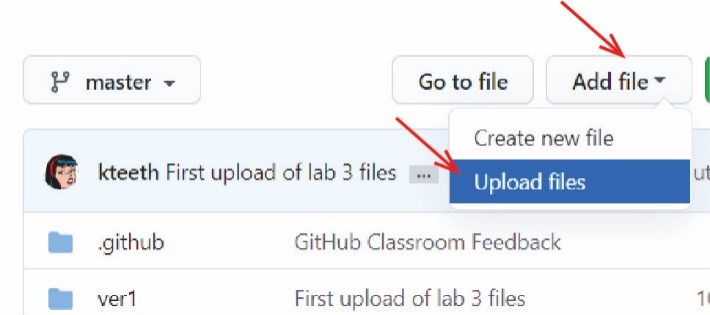
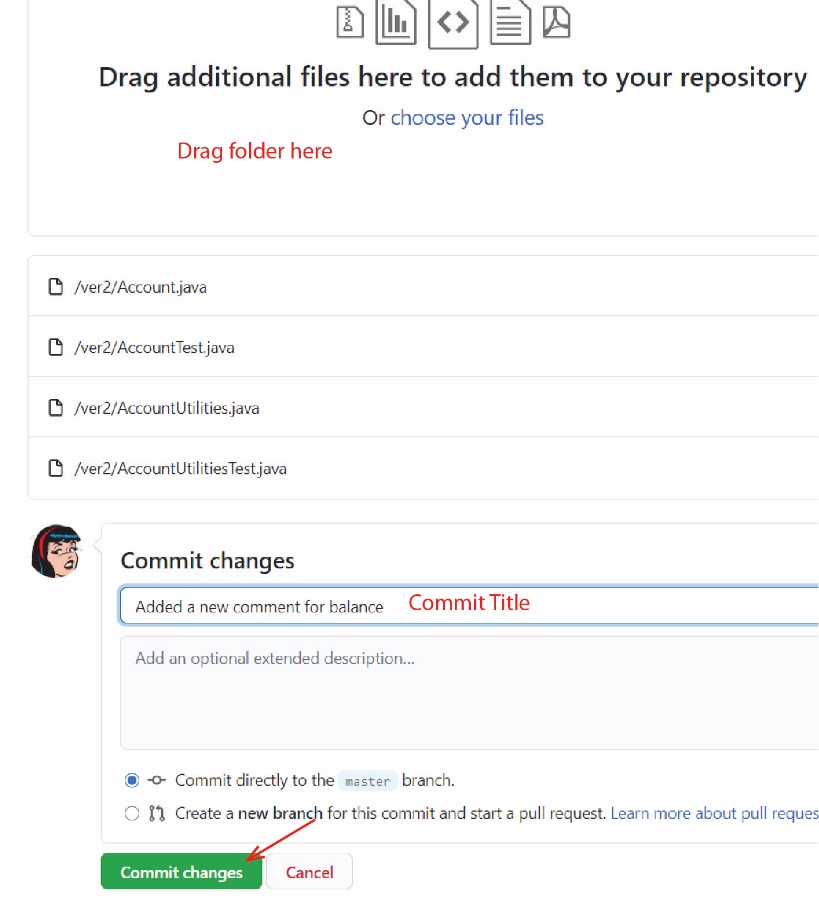
1. Type a commit title
2. Type a commit description (optional)
3. Press: Commit changes

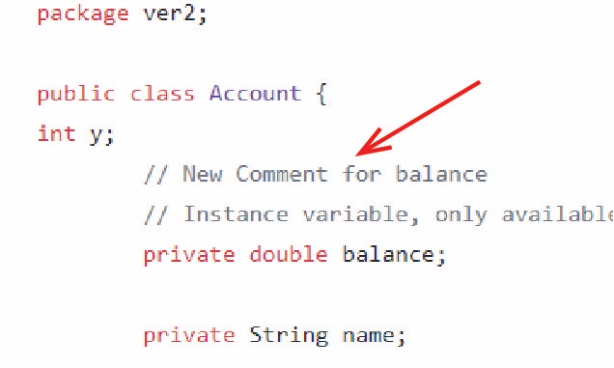
Note: this saves your files on the *master* branch.

1. The result will show your folders. You can drill down into them and see the files. You can edit them to. I don’t recommend that unless you learn other Git and GitHub concepts.



# Subsequent Upload to GitHub

1. Next, you will make changes to your copy of Lab 3 code on your local machine, and then upload to GitHub. Do the following on your computer, not GitHub:
2. Open, *Account.java* in the *ver2* folder. I just used Notepad, but you can use Eclipse if you want.
3. Add an additional comment on the *balance* instance variable. I added: // New Comment for balance
4. Save and close *Account.java*.
5. Return to GitHub and do the following:
6. Choose: Add file, Upload files.
7. Drag the *ver2* folder (since that was the only change) into the region provided for upload. Note, alternately, and probably better, you could have drilled down into the *ver2* folder on GitHub and then choose: Upload files. Then, you could simply drag the one modified file, *Account.java*. We do that later.
8. Type a *commit title*
9. Type a *commit description* (optional)
10. Press: Commit changes
11. The repo should update properly. Drill down into *ver2* and open *Account.java* and observe that the file was changed.



1. Next, we will essentially, repeat what we just did. Do the following on your computer, not GitHub:
2. Open, *Account.java* in the *ver2* folder.
3. Remove this constructor:

// Constructor

public Account(double balance) {

this("Unknown", balance);

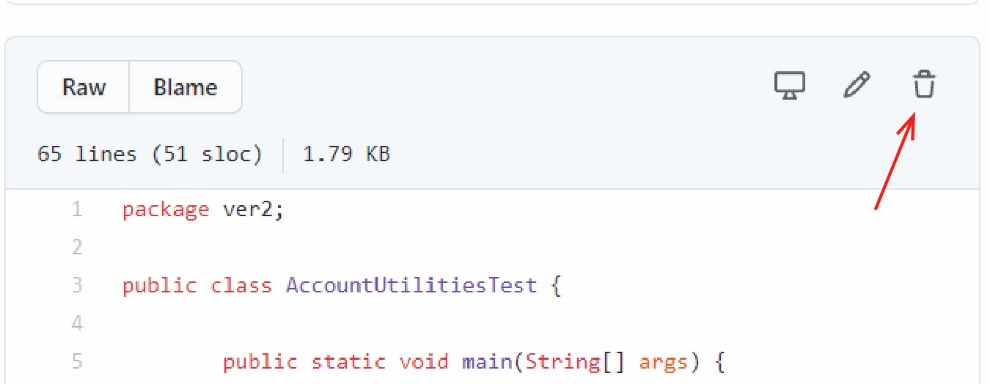
}

1. Save and close *Account.java*.
2. Return to GitHub and do the following:
3. Drill down into the *ver2* folder.
4. Choose: Add file, Upload files.
5. Drag *Account.java* from your computer into the dialog.
6. Type a Commit Title, and choose: Commit changes
7. The repo should update properly. Open *Account.java* and observe that the file was changed on GitHub.

# Delete a file from GitHub

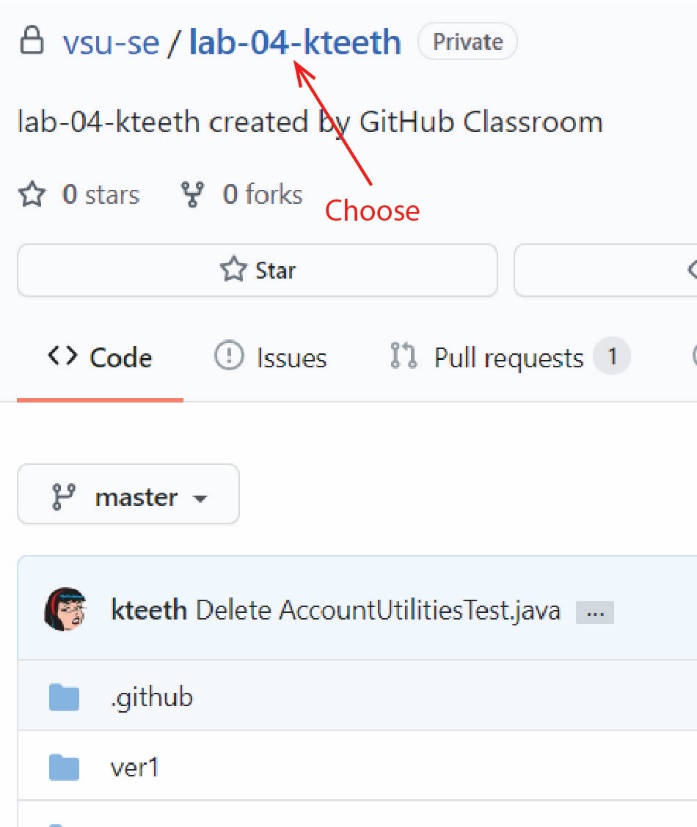
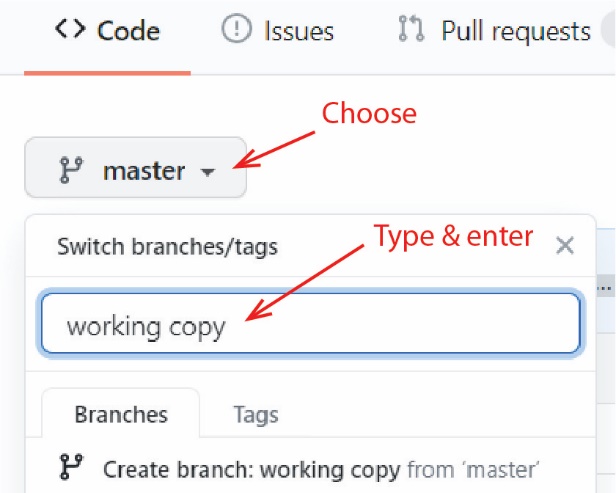
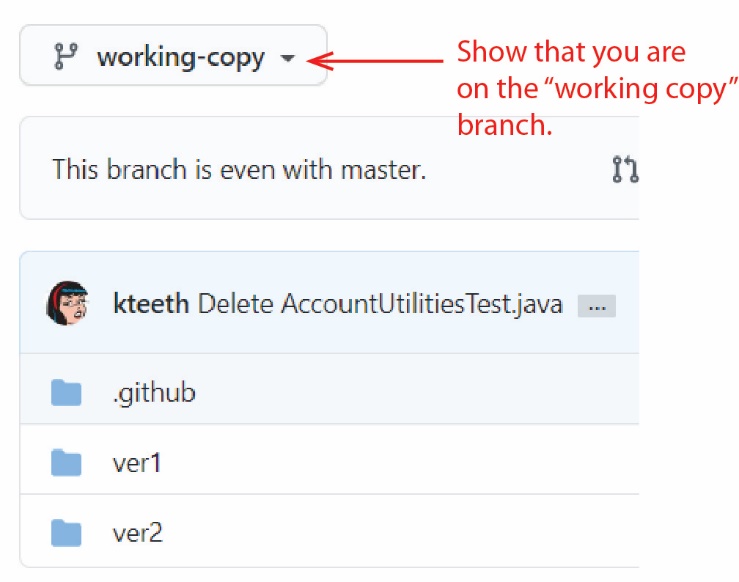
You may want to delete a file from GitHub. We do this next, and it generates a Commit dialog as it did when we added files. There is not a way to delete all files in the repo at once using the GitHub.

1. Do the following on GitHub:
2. Drill down into the *ver2* folder and display: *AccountUtilitiesTest.java*.
3. On the right, choose the trash can icon



1. Scroll down, and you will see that a Commit Title has been supplied for you. Choose: Commit Changes.
2. You can also edit a file on GitHub by choose the Pencil icon while the file is being displayed. Once saved, it will generate a Commit dialog.

# Branching

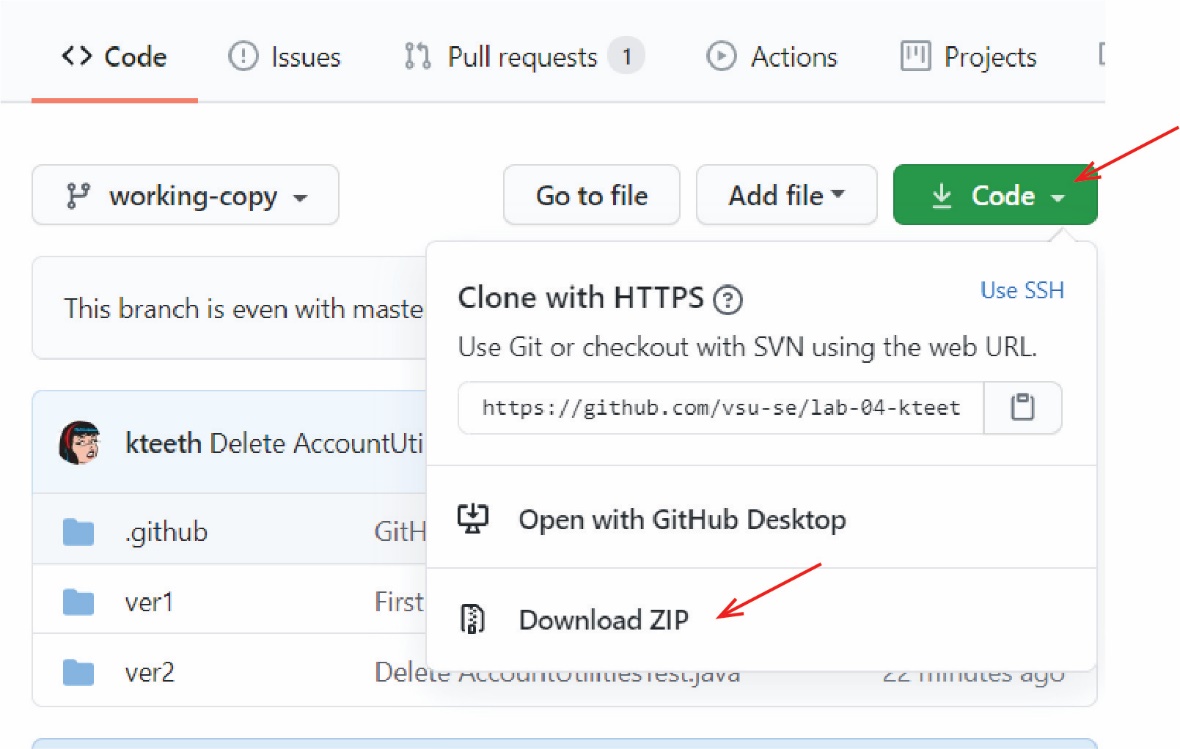
1. For our purposes, you can create a *branch* which is a copy of your code.
2. Select the name of your repo as shown in the figure on the right. This takes you back to your “code home”.
3. Do the following as shown in the figure on the right:
4. Notice that currently we are on the *master* branch.
5. Select: master
6. Type a name for the new branch (*working copy*) and Enter.
7. Notice that it now says we are on the *new-feature* branch and we see the package folders.
8. You can switch back and forth between the branches with the dropdown. You are welcome to experiment.
9. Note:

* In practice, the *master* branch always has working and tested code. All development takes place on a branch. When it is working and tested, then it is merged into *master.* Merging can be tricky when a “conflict” occurs, but if you are working by yourself it is usually easy.
* If you are working alone, you might choose to simply work off *master.* And, you might make a branch to simply make a copy of your code from time to time as you experiment on main. **Note: this is NOT the proper way to use GitHub, but it perhaps simpler for small assignments.**

1. Branching is part of the GitHub workflow (which we will not adhere to):
2. The *master* branch always contains working code.
3. When you want to change the code in *master*, you make a *branch* which is a copy of *master.*
4. You make your changes on the branch.
5. When tested and complete, you do a *New pull request*. This will step you through *merging* your code back into the *master* branch.
6. In general, with a team of 3 people assigned to a repo, for example, each person would have their own branch, writing the code they are responsible for. And when complete, each person merges their code back into *master* (via a pull request).
7. Merging can get tricky, so we will not consider that.

# Tips

1. **Download code from GitHub.** You can download all code in a zip file. To do this, with the Code tab active, press the green button that says, “Clone or download” and choose: Download ZIP



# Submission

In the Lab 4 drop box, type your GitHub Username (the username you use to login to GitHub) in the Text Submission box.

**If your GitHub Id is not correct, you will lose 25 points.**

Here’s a way to verify: type your username: Type it into the Text Submission box, and before submitting, select it, copy it, and then use it to login to GitHub.

**You are done!**

1. Git is a command line language for utilizing repo’s on a personal device and interacting with GitHub. [↑](#footnote-ref-1)