GitHub Classroom

GitHib Education and GitHub Classroom is a product and service offered by GitHub for students and educators to learn and teach version control, and also to facilitate training in professional software development. **All** programming courses will use GitHub Classroom and version control. Students will be required to maintain private student repositories and submit all code via GitHub.

[Prepare | Tutorial | Process | Instructions]

Prepare

[back-to-top]

You will need a GitHub account to participate in programming courses. Any GitHub account will do, you do not need a special GitHub account nor do you need the student developer pack, however, you may wish to get access to the free student tools for your own purposes.

⚠ If you use your school email to sign up for an account, you should add an extra email to your account in case you lose access to your school email. ⚠

General Tutorial

[back-to-top]

Once you are set up with GitHub, read and complete the tutorial . This is usually covered in class as well (or a recorded lecture will be provided). The tutorial is general introduction to basic version control operations and concepts. The tutorial assumes you start local and push remote. Assignments for class will be the reverse, they start remote and pull local. **You need to know how to do both to be a developer**. The specifics of how GitHub is used for assignments is detailed after the following general description.

General GitHub Process for Assignments

[back-to-top]

The following is the **general** process for working with GitHub Classroom. Read this to understand the *basic concepts* of what will be done for assignments, and then the *specific* instructions on how to do this is given below this description.

- For **each assignment** you will be given an invitation link for the assignment, usually one per assignment.
- Accepting the invitation will create a private *remote* repository with starter code.
- You can get back to all your repos anytime by going to **https://github.com/ACC-Computer-Science-*semester*-*year*** where you replace *semester* and *year* with the current semester and year. For example, **https://github.com/ACC-Computer-Science-Fall-2000**
- You will create a local area on your computer to work (i.e. create a new directory for each assignment unless instructed to do otherwise).
- In that local area you will initialize git, connect your repo, and you will pull code back to your local development area from your remote repo.
  You will work in your local area, *frequently* committing code, and using version control and your repo normally. While working on assignments, **you must** commit often, small, and smart! This means you commit with every small, logical, well-test block of code. For example, when you have completed one function or method, and tested it very well, commit it. Another example, you fix one bug, you test the fix very well, commit it. Your commit messages need to be informative and explain fully what you just added/fixed. For example...

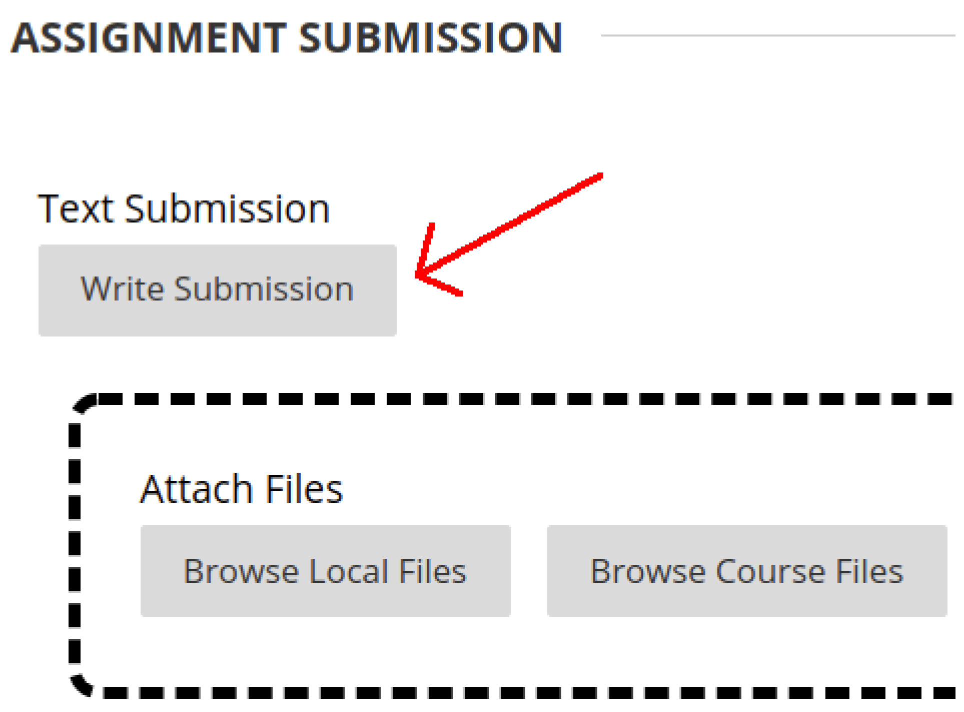
  "*Fixed bug in function get_array(). There was a one-off error causing the function to overrun the array. Fixed it by* *changing the <= to < in the for loop.*"

Do not make commit messages like, "done" or "committing code" or "fixed bug" or "final commit." Commit messages should tell a "story" about what you did, how you did it, your thought process, and why it was necessary to perform each step. Follow this guide to learn professional repository habits...

- **Commit often**: With each well tested feature or bug fix.
- **Commit small**: One function, one bug fix, one stub, etc.
- **Commit smart**: Fully explain what you did since the last commit.

Failure to commit often, small, and smart will likely result in your assignment being rejected and being assigned a grade of zero. This is a **required** part of every assignment.

- When you are ready for grading, you will push all your code to the remote repo and submit the link to your repo on Blackboard using the "write submission" area (not the comment section).



Sending your link via Blackboard is your signal that you are ready for grading. Note that the link is **NOT** the SSH link, and **NOT** the .git URL. The link you want is the actual Web URL from your browser's address bar that is the link to your repo. It should start with https:// and end **without** a ".git".

- While working, you can push as much or as little as you like, however for grading purposes you must push all the code you want graded. All code is graded on GitHub. **If it's not there, it will not be graded.**
- **CHECK YOUR REPO!** When you are ready to submit, double and triple check your repo is complete and up to date. The best way to do this is download your own repo into an empty directory and run the application you downloaded and make sure it works. If you do not do this, and if your repo is not complete when you submit, you will get a zero.
- If you make changes after you submit your link on Blackboard but before it's graded, that's fine. In that case DO NOT resubmit your link, just update your repo.
- Once your assignment is graded you will usually have an opportunity to resubmit *once* for a higher grade. See submission guidelines for details on this.

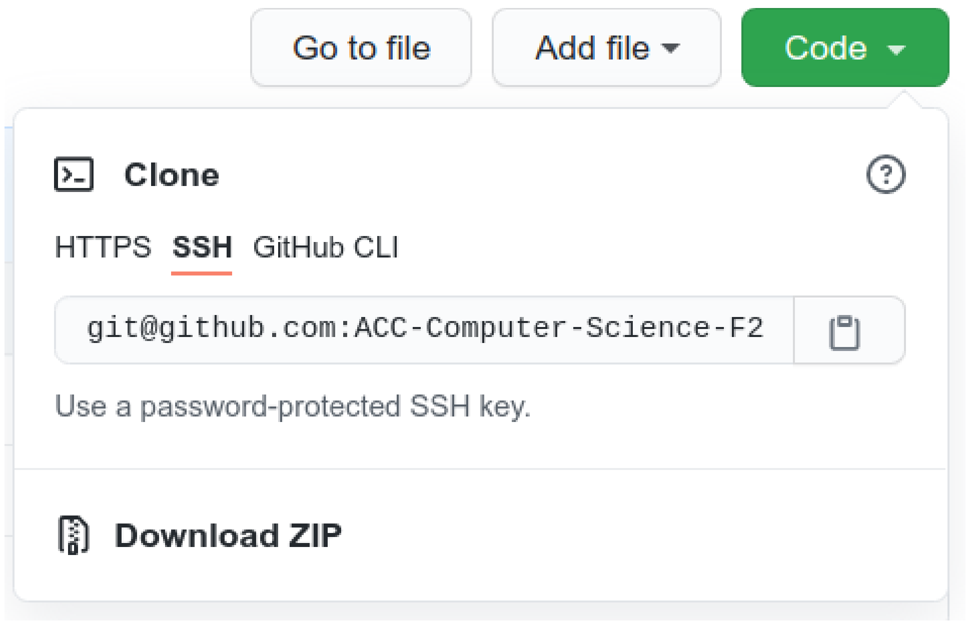
Assignment Specific Instructions

[back-to-top]

The following are the specific instructions/commands which will accomplish the process explained above. Follow these instructions for each assignment.

- Click the assignment link given to you with each assignment. A repo with starter files will be created for you on GitHub. This is your *remote* repo.
- Create an **empty** project directory locally. Make sure this directory does not have another Git instance in its path. i.e. No parent directory and no child directory may also have a git instance.
- Initialize git in your project directory with the command **git init**
- Add your remote repo locally by using the command **git remote add origin <your-repo-url>** Get your specific url from your repo by clicking the "*code*" button in your repo and then using the copy button. **Do not clone or download**, just click the copy button so you have access to the SSH link to copy/paste into your *git add origin*

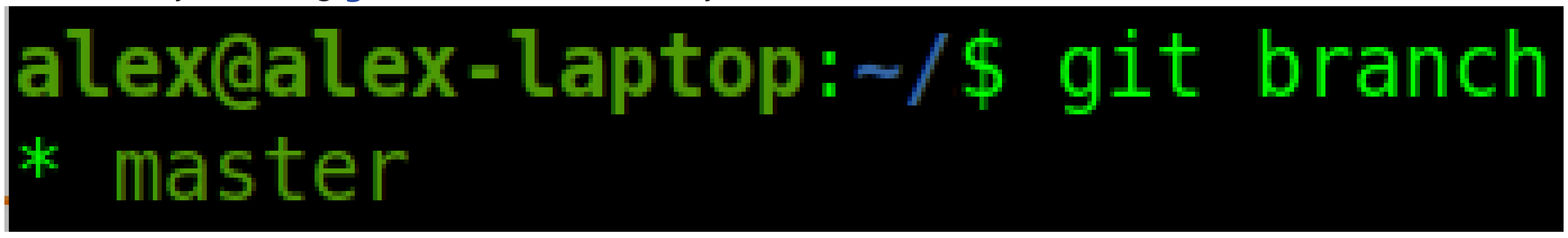
command. Notice the SSH tab is selected. You need to use SSH.



```
git@github.com:ACC-Computer-Science-F2
```

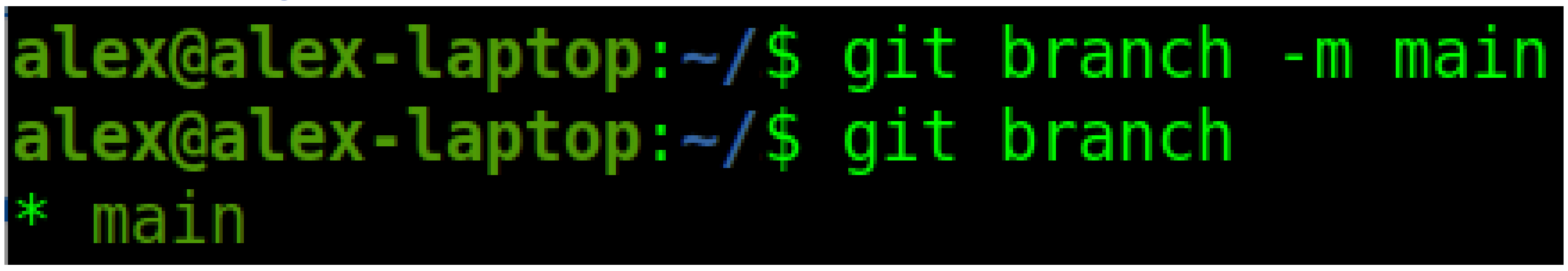
Use a password-protected SSH key.

- Pull your remote code local by using the command **git pull origin main**
- **NOTICE**: Because of the recent GitHub change to the default branch being "main" instead of "master" (this is also explained in the tutorial) there is an extra step at this point that *may* be needed. This is *not* part of the normal process, but is a "fix" until the "master" label goes away completely.
  - You pulled 'main' but your local git init may have made a 'master' branch. This will cause a conflict so you need to fix it now.
  - Check this by executing **git branch** command. If you see this



```
alex@alex-laptop:~/$ git branch
* master
```

then proceed to the next step to fix it.
  - Execute the command **git branch -m main** to rename the "master" branch to "main". Check you were successful with the **git branch**



```
alex@alex-laptop:~/$ git branch -m main
alex@alex-laptop:~/$ git branch
* main
```

  - Now proceed with the normal instructions in the next step.
- If it's not already given to you, create or modify the **.gitignore** file to ignore **everything** except your code files and what is **needed** for submission and grading. This will be discussed more in class.
- *If* you created or modified the .gitignore, commit it now. Do this by executing the following commands:
  1. **git add .gitignore**
  2. **git commit -m "initial commit"**

Now you are ready to begin working.

- While working on your assignment, commit your code with each logical and well tested change. For example:
  - After you create a new file, put your comment header in it, save it, and commit it with the message "initial commit."
  - After you complete **one** function and you are sure it works commit it with a description of what you did and how the function works.
  - After you fix **one** bug and you are sure it's fixed, commit it with a description of what you did to fix it.
- To commit files do the following:
  - Add the files ready for commit: **git add file.ext**
  - Commit the file(s) with a message: **git commit -m "some smart message"**
- Never use a complete wild card when you git add! For example never do this:
  - **git add .**
  - **git add ***

- When you want to sync your local code to your repo push it with the command: **git push origin main**
- When you are ready for grading, do your final commits, do a final push, and submit your repo link on blackboard under the assignment using the "*write submission*" feature.
- **THIS ENDS THE NORMAL INSTRUCTIONS**: The next two items are extra information for the case when you want to change something remotely (i.e. directly on GitHub). You should not change things remotely! It's dangerous and bad practice, and you are strongly advised not to do it. However, if you do...
  - Before you make any remote change make sure your repo is in sync (i.e. do a push).
  - Immediately after you make a remote change sync your local code (i.e. do a pull).
  - WARNING: Never make a remote *and* local change without pushing/pulling *between* the changes to stay in sync! If you do, you will have to reconcile the conflicts which is usually so difficult you will probably end up deleting your repo and starting over.

[back-to-top]