

Setting up your Git repository

Git is a widely used, open source software for file management and version control. Version control software such as Git enables you to keep track of changes to your code over time in an organized way, and enables multiple developers to work together on complex software without stepping on each others' toes. GitHub is an online code hosting platform that includes Git version control.

For our course, we will be using the CCIS Enterprise GitHub to submit homework and lab assignments.


This class has an [organization](#) on the CCIS GitHub server called **[CS5009-Seattle-Campus-Spring-2022](#)**. You will store your code in repositories in this organization. I will add you as members of the organization, using your CCIS user name. If you have not done so, please provide your CCIS user name then you could be added into this organization.

Creating a repository on GitHub

You will create a repository on GitHub and link it to a local repository that you create on your own computer. This will enable you to sync your work between your own computer and GitHub.

To create the repository on GitHub follow these steps:

1. Navigate to **[CS5009-Seattle-Campus-Spring-2022](#)**. It will look something like this (with the current semester's organization name, of course):



CS5009-Seattle-Campus-Spring-2022

Part of Northeastern University

[Repositories 17](#)
[People 58](#)
[Teams](#)
[Projects](#)
[Settings](#)

Type
Language
Sort

Customize pins
New

Liang-Huang-CS5009 Private

C++
0
0
0
0
Updated 4 minutes ago

SiqiLei Private

0
0
0
0
Updated 7 minutes ago

Student_Xinyi_Cheng Private

C++
0
0
0
0
Updated 8 minutes ago


CS5008_Jiawen_He Private

C++
0
0
0
0
Updated 10 minutes ago

Top languages

C++

People 58 >



Add someone

- Click "New" to create a new repository. Fill in the form as shown below, with the name of the repo `student-YourName` (without the angle brackets and with your own name in place of YourName). Check private and do *not* check "Initialize with README". Finally, click "Create Repository":

Create a new repository

A repository contains all project files, including the revision history.

Owner *



CS5009-Seattle-Campus-Spring-2022 ▾

Repository name *

/ student-SiLi ✓

Great repository names are short and memorable. Need inspiration? How about **furry-invention**?

Description (optional)



Public

Any logged in user can see this repository. You choose who can commit.



Internal

Northeastern University [enterprise members](#) can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.



Add a README file

This is where you can write a long description for your project. [Learn more.](#)

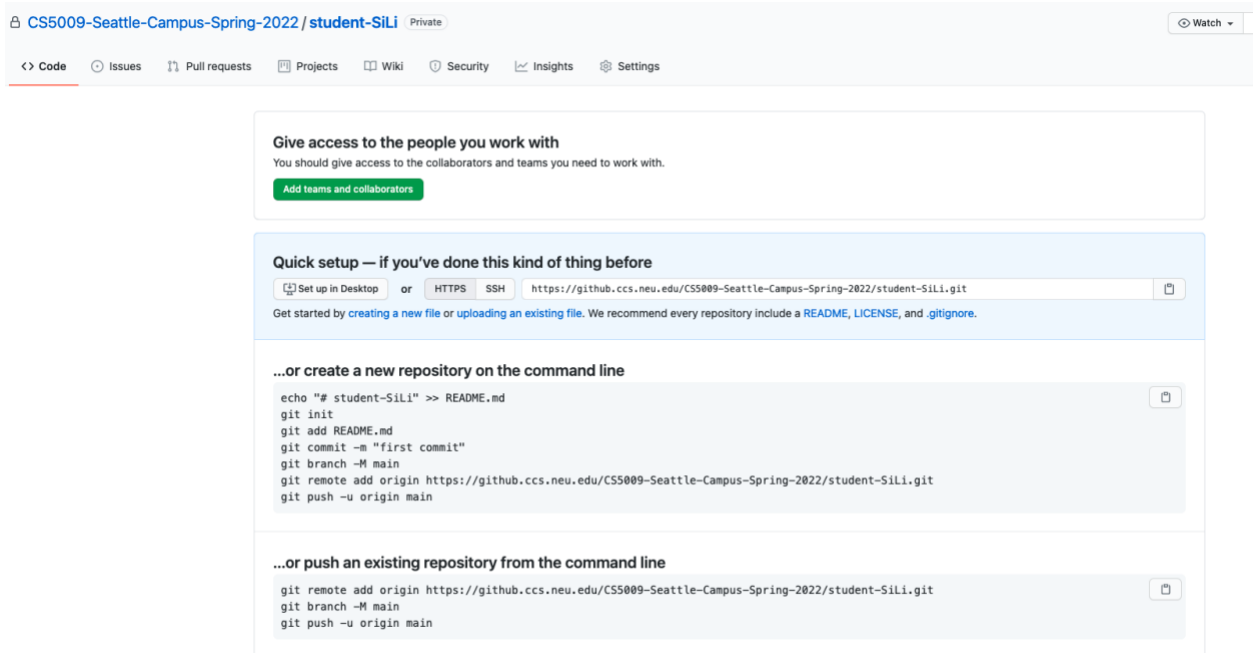


Add .gitignore

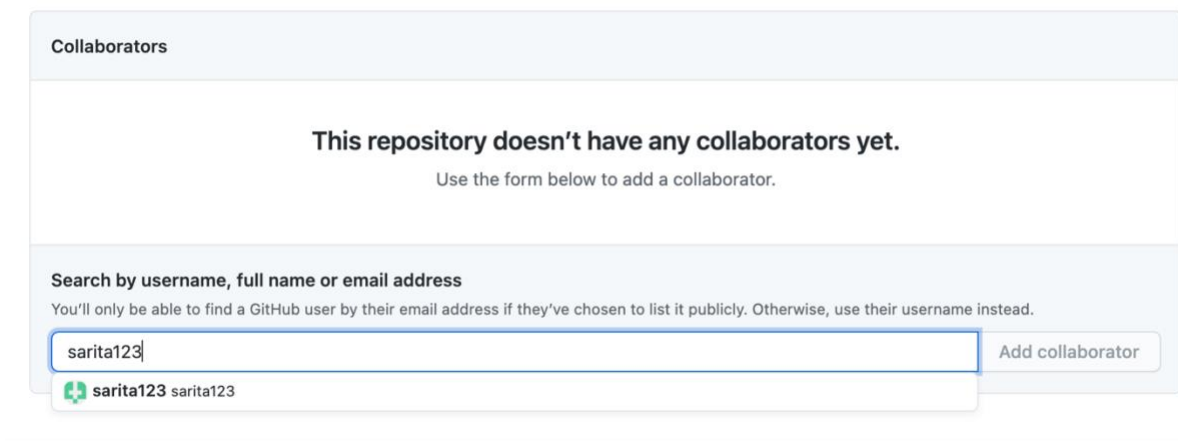
Choose which files not to track from a list of templates. [Learn more.](#)

Create repository

3. You'll now see a page like the one shown below. You're going to follow the second set of instructions in a moment, under "...or create a new repository on the command line" (the instructions are also repeated below in the next section), but first, click "Add teams and collaborators":



- Under the Settings menu, in the "Collaborators" area, search for professor and TAs' username `sarita123`, `binhuiyun`, `linusguo`, `lisils`, one by one as shown below, then click "Add collaborator".



- You can return to the instructions page by clicking on the repository name at the top of the page.

Creating your local repository and linking it to GitHub

Return to your terminal. Make sure that you're in the `cs5008` directory. You can check this by typing the command `pwd` at the prompt (stands for "present working directory"). If the output of `pwd` doesn't say `cs5008` (along with the full path to your directory) you'll need to change your working directory to `cs5008`. Do that by using the `cd` command in your command line.

From within `cs5008` carry out the following steps (these are the same as the instructions on your GitHub page)

1. Type the following line in your command line (replacing the "YourName" with your actual name as you did in the name of the repo), and press enter:

```
2. echo "# student-YourName" >> README.md
```

This creates a README.md file and puts some basic text in it.

3. Type the following line in your command line, and press return:

```
4. git init
```

This initializes a local Git repository in the `cs5008` directory on your computer. You only need to ever do this once for a project.

5. Type the following line in your command line, and press return:

```
6. git add .
```

This is a little bit different from what GitHub says, but it's more general, and it's what you'll use on other occasions. This "adds" all the changes you've made so far to prepare them to be committed. You've only made one change in your directory, (adding the `README.md` file), so that's all that's being "staged" for committing now. You can think of it as a formality you go through before committing changes.

7. Type the following line in your command line, and press return:

```
8. git commit -m "first commit"
```

This commits all recent changes to your project. "Commit" here means to make a permanent record that can be accessed and referred to in the future. You will always be able to go back and see exactly what changed in the code at this point. the "first commit" here is the message you associate with this commit. You should always write an informative [commit message](#) that briefly describes what you did since the previous commit.

9. At this point, you've successfully committed changes to your *local repository*. This means that your repository's history is maintained on your own computer. This can be useful for you, but it's not something that can be easily shared with other collaborators, and it doesn't take advantage of remote storage. So we'd like to put a synced duplicate of this repository on the cloud on GitHub. To do this, type the following line of code (with appropriate changes to the name of the GitHub repository) in your command line and press return:

```
10. git remote add origin https://github.ccs.neu.edu/CS5009-Seattle-Campus-Spring-2022/student-YourName.git
```

Here, you're telling Git to treat the URL as a *remote repository*, that is, an offsite mirror of this repository. The name "origin" is what this particular remote will be called (it is possible to have more than one remote repository for a local repository, so we need to give them names.)

11. Finally, we want to "push" our recent committed changes to the remote repository. We do this (the first time) with:

```
12. git push -u origin master
```

The `-u` tells Git to set the `origin` remote (i.e., the one on GitHub) to default. The "master" branch is the main branch of your repository (Branching is for when multiple areas of development on the same process need to be carried out in parallel without interfering with each other, for example when teams are working on experimental features that will later be incorporated into the main software. We'll only ever use the master branch in this course.)

At this point, you may be asked to log in to GitHub. You'll be prompted for a user name and password, which are the same CCIS credentials you used to log into GitHub.

In the future, you won't use the `-u` flag again when pushing changes to GitHub. You will only need to type `git push`.

13. And you're done! Go back to your browser and refresh the GitHub repository page. If all went well, the instruction page should be replaced by something looking like this:

Congratulations! You now have your repo up and running, both locally and remotely. For the rest of this class, you will do your work in the local directory of your repo. Whenever you write some code that you'd like to keep a record of, you'll do the following steps in the command line. (Making sure you are in the project directory in your command line):

```
git add .  
  
git commit -m "your informative comment"  
  
git push
```

And your changes will be pushed and stored in GitHub.

Committing and pushing your changes

You should stage and commit changes to your code frequently. Think of this as something like saving. You should commit each time you make a meaningful change to your code. (You *must* commit to submit your code for grading, but you may, and should, commit more frequently than that). Now, we'll stage, commit, and push the changes you've made since your first commit. These include creating a new directory (such as `lab01`) and creating and editing a new file (`lab01.cpp`).

1. In order to manage the git repository of the project, you should be in the project's *top level directory* in your command line. Since you're currently in a subdirectory (such as `lab01`) you can get to the project's top level directory (`cs5008`) by using the following command:

```
2. cd ..
```

This means change directory to the present directory's immediate parent.

When you do this, run `pwd` to double check that you're in the `cs5008` directory.

3. In `cs5008` run the following commands, one after another:

```
4. git add .  
  
5. git commit -m "Added lab directory and Hello World program"  
  
6. git push
```

7. And you're done! Go to your GitHub repository on the web and refresh the page. Poke around that repo and take a look at your code.

The commands in step 2 are all you'll ever need to do to push your changes to GitHub. You'll run exactly the same commands (aside from the message) each time you make changes to your code.

8. Make some more changes to `lab1.cpp` that include both adding and removing lines, and push those changes to GitHub again. Inspect the commits by looking at the "commits" link on your repo. What kind of information does GitHub maintain about each commit? What does it look like when you add lines of code? What does it look like when you remove lines of code?
9. If you haven't already, go back and add the TAs as collaborators on your repository.