



## 1 Requirements

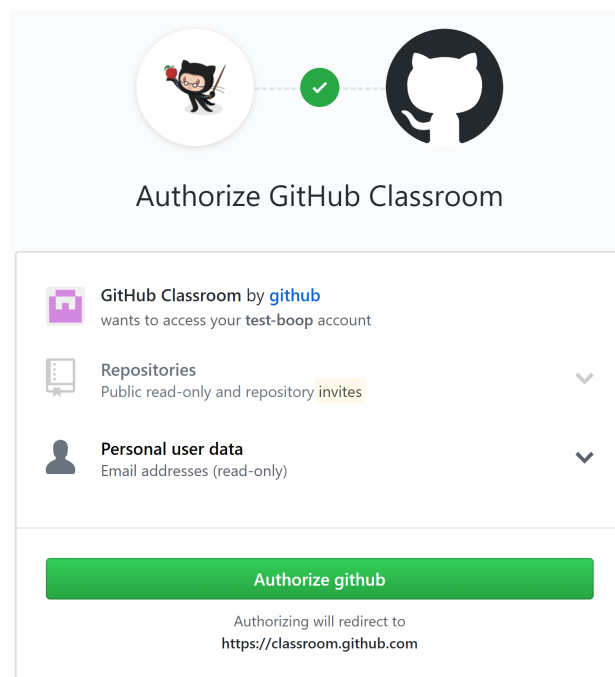
- (1) To submit your work with GitHub classroom, you will need a GitHub account. If you do not already have one, you can create an account for free at [github.com](https://github.com)

*Note:* Although not required, any student can get GitHub pro for free. If you are interested, go to [education.github.com](https://education.github.com) for more information.


- (2) The Eng-Grid computers already have git installed, so you can ssh into them and upload your code from there. If you want to upload your code from your own machine, you will need to download and install git from [git-scm.com/downloads](https://git-scm.com/downloads)

## 2 Creating your repository

We will give you a link to each assignment or lab. Once you visit the link, if this is the first time you are pushing your code on GitHub through GitHub classroom, you will be presented with an authorization screen like the one below.




Accept the authorization, which should lead to a new screen asking you to link your GitHub account to the emails that we have specified on the class roster. **Do not skip this step, and make sure that you choose the right email, since otherwise we will not be able to link your work to you.** If you choose the wrong email accidentally, then ask one of the instructors for help.

 Join the classroom roster


Your teacher has configured this classroom to pair GitHub accounts with identifiers. Please select yourself from the list below. You can also [skip](#) this step for now.

Identifiers
 

 ulskis@outlook.com

Skip

Once all of the authorization is done, or if you were already authorized, you need to accept the assignment.

 Accept the **Test Assignment** assignment

Accepting this assignment will give you access to the **test-assignment-test-boop** repository in the [@EC327-Fall-2019](#) organization on GitHub.

Accept this assignment

GitHub classroom will automatically create a unique repository for you, which you will be emailed a link to.

Your GitHub repository was created.


Your assignment repository is being setup. This might take a while.

Creating repository
 

Done

Importing starter code
 

Done

 Accepted the **Test Assignment** assignment

**You are ready to go!**

You may receive an invitation to join [@EC327-Fall-2019](#) via email invitation on your behalf. No further action is necessary.

Your assignment has been created here: <https://github.com/EC327-Fall-2019/test-assignment-test-boop>

### 3 Creating your Local Repository

If you have not created any code yet, or you need to modify a repository that already contains existing code, continue to section **3.1**. If you already have a folder on your computer that holds your code, skip down to section **3.2**.

### 3.1 Cloning the Repository

Now that you have a repository connected to GitHub Classroom, you need to start adding your own changes to it. You can clone your remote repository to your local files by running the following command, where the repo link is the unique link that was generated for your remote repository:

#### Cloning a Repository

```
git clone https://github.com/EC327-Fall-2019/your-repo
```

After cloning your repository, you should see that a new folder with the same name as your repository was created. Any existing code that may have been in your repository will also be in this folder. You can now edit or add files to this folder, which you can upload later to your remote GitHub repository. You will get a warning if the repo is empty, but this is fine.

### 3.2 Initializing a Local Repository

If you want to start adding existing code to your repository, then you can initialize a local repository on your computer and push your code from there. **For the purposes of this class, only do this if the repository is empty and you already have a folder on your computer with code that you wish to upload.** We can do this by creating a local repository on your machine that keeps track of your changes, and pushing everything to your GitHub Classroom repository. Using the terminal, navigate to the folder where your code is located, and execute the following commands.

#### Initializing a Local Repository

```
git init
git remote add origin https://github.com/EC327-Fall-2019/your-repo
```

- The *git init* command will create a local repository
- The *git remote add origin* command will add the remote URL of the unique link that was generated for your remote repository to your local repository, so it knows where to push your changes (upload your code) in the future

## 4 Pushing Your Changes

When you want to upload changes, there is a different process for pushing the first time and any time after this. When you first create your repo, the git client does not know where you want to push your code, so you add *-u origin master* to the end of the *git push* command to specify the master (main) branch as the default place to upload your code. Any time you push in the future, this will be the default branch, and you will not need to specify the *-u* flag or anything that is after it in the first push code block below.

#### Pushing Your Changes for the First Time

```
git add .
git commit -m "first commit"
git push -u origin master
```

#### Pushing Your Changes after the first commit

```
git add .
git commit -m "meaningful commit message relevant to code changes"
git push # pushes to whichever upstream branch was set previously
```

- Executing *git add .* will stage your files so that they can be committed. **Just doing git add will not upload your files.** The period after *add* tells the git client to add every file within the current directory and its sub-directories. If you want to add all your changes, make sure that you are in the root directory of your local repository.

- *git commit* will store/record your changes locally, and adding *-m "first commit"* will leave a message, or tag, on this commit called "first commit". Any future commits should have a message that is relevant to whatever changes you are adding to the repository
- *git push -u origin master* will push all of your code to the master branch of your repository (typically the main branch) and the *-u* flag specifies to set master as the upstream (default) branch to push to in the future.

## 5 Useful Commands

- (1) *git add* to stage files to be committed.

Example: Add files to git

```
git add <file1> <file2> # add 2 files named <file1> and <file2>
git add *.cpp # add all .cpp files
git add . # add all files
```

**Note:** Git add will only be applied to files within the directory that you are in and its sub-directories

- (2) *git status* to see which files are changed, and which are staged for a commit

Example: Check staged files with git status before committing

```
git add .
git status
git commit -m "meaningful commit message relevant to code changes"
git push
```

- (3) *git reset* to remove staged files (this will not remove changes, it will just "undo" the git add command)

Example: Remove files staged for commit

```
git add .
git status
# At this point you realize that you did not want to add all of the files
git reset
```

- (4) *git pull* to pull (download) changes to the code that are in the remote repository but not in your local repository
- (5) *git diff* to view changes to the code that are not yet committed in your local repository
- (6) *git checkout* to create and switch branches

Example: Remove files staged for commit

```
git pull # make sure your current branch is up to date first
git checkout -b [name_of_your_new_branch] # Create a branch locally
git push origin [name_of_your_new_branch] # push the changes to the remote repo
git branch -a # list all of the branches
git checkout master # Switch back to the master branch
```

**Note:** Although branches can be very useful, especially on larger software engineering projects, you do not have to use different branches for EC327

## 6 Things to Note & Resources

- You should typically never only push your code once for a project. It is much better practice to push your changes after each major update, with a commit message that details your changes in case you want to go back. We will not be grading you on this, but we highly recommend you follow good practices, especially for bigger assignments such as the PAs.

- Make sure that you push all of your changes before an assignment's due date. Every change that you add to your repository has a timestamp.
- There are many more commands and powerful applications of git, but for the purposes of this class you only need to know how to create repositories and commit/push your code.
- We recognize that git is new to many of you, so please ask questions during labs, post on piazza, or come to office hours if you are having trouble or need more clarification on something
- Your final grade for assignments will be based off of code that is on your master branch only.
- If you want to know some more git commands and look more into how they work, the following guide is a nice and simple rundown of git: [up1.github.io/git-guide/index.html](https://up1.github.io/git-guide/index.html)
- This site has some useful resources to learn git: [try.github.io](https://try.github.io)
- Here is a GitHub tutorial for beginners: [product.hubspot.com/blog/git-and-github-tutorial-for-beginners](https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners)