

# Introduction to Github Activity

The following set of instructions will walk you through the process that you should follow when making changes to your project files.

## Prerequisites

This activity assumes that you have already installed git on your computer or that you have a portable version on your thumbdrive. The portable version is recommended as it can be “installed” on a thumb drive and allows you to use git on any computer you plug the drive into.

Both the full installation and the portable version of git can be downloaded [here](#). Note that only Windows has a portable version. Typically OSX and Linux will have git installed already.

## Step 1: Download “clone” the repository

First you’ll need the URL where your remote repository is located which is on the github site. The URL will look like the following where *repository-name* is the name of your team.

<https://github.com/mcc-robotics/ repository-name>

Once you have the URL you can clone the repository by simply adding **.git** to the end of the URL. Open the git bash console, navigate to the directory where you want to store your repository and run the following command.

```
$ git clone https://github.com/mcc-robotics/ repository-name.git
```

The repository should now be stored locally on your computer or thumb drive in a folder with the same name as your repository. You can navigate to the folder using the operating system, you only really need to use git to upload and download files from github, then you can edit them just as you would any other file on a computer.

### Note

If you don’t know how to navigate in the git console here are the few commands you will need.

```
$ cd folder-name Opens a folder, just change folder-name to the folder you want to open
```

```
$ cd .. Goes back one folder
```

```
$ ls -l Lists the files and folders within the current location.
```

## Step 1.1: Pull changes

Cloning is only necessary the first time you download a repository. Once you have cloned the repository you can update your repository with git pull. Pulling will update your files with any changes your

teammates have made. You should **always** call git pull before you start working on your project files in the repository.

```
$ git pull
```

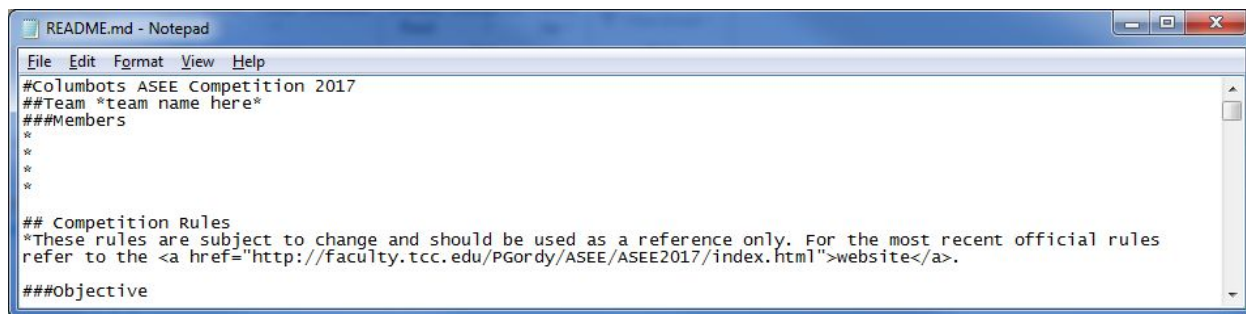
Go ahead and run this command now, you will simply get a message that says everything is up to date since we just cloned. If a teammate changed something it will display a list of any commits that have taken place and update your files for you.

## Important

Always use the git pull command before working on files in a repository. If you do not update your local files before working on them you could have conflicts when you try to commit. Conflicts can occur when multiple teammates have altered the same file, if you both edit the same portion of the file, git will not know whose updates to use. This is definitely fixable but it can be confusing to new users so it is best to avoid having conflicts in the first place.

## Step 2: Edit file(s)

Once the repository is cloned or updated you can make changes to the files. Open the README.md file located in the main repository folder. At the top you should see a section that looks like the following screenshot.



Don't be concerned with the odd characters, you will simply be adding your team name and/or member name and leaving the rest alone.

If another member has not already changed the team name, go ahead and replace *\*team name here\** with your actual team name.

Now next to one of the blank lines with an asterisk (\*), add your name, this is going to be a list of your team member names that shows up on github. Each member will add their own name as part of this activity.

Once you've made your changes, save and close the README.md file.

## Step 3: Check the status of our local repository

At any time you can use the following command to see what, if any, changes exist. Run the command to see what has changed. Note that in the console you need to enter the folder for your repository.

```
$ git status
```

You should see something like the following.

```
gber1@GBERLOL MINGW64 /c/Users/gber1/repos/mcc-robotics/Initial Team Repo (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Line by line this is what we can find out about our current state

1. At the end of the first line you'll notice the blue text "(master)". This is indicating to us that the git console recognizes that this is a repository and that we are on the master branch. We won't discuss branches just yet but it's important to note for now.
2. This line is simply the command that has been entered, in this case "git status"
3. Git tells us that we are on branch master.
4. Git tells us that we have changes that aren't staged for commit. What this means is that we have made changes but we haven't added them to the staging area. Whenever we make changes, in order to save them in git, we need to add them to the staging area and commit them.
5. Git is telling us that we can use the "git add" command to add the changes we've made.
6. Git is telling us we can use the "git checkout" command to clear our changes and go back to the way things were before making any changes. This command is helpful if you try a new idea on some code and it doesn't work or if you make a mistake and just need to undo your changes, you can simply revert back and you can revert a single file, multiple files or all files.
7. <blank>
8. Git is telling us that we've modified a file named "README.md" and it's colored in red because it is not staged. Here git may also tell us if we've added a new file.
9. <blank>
10. Git is telling us, there aren't any changes in the staging area to commit but we can use "git add" to add them. It is not recommended to use "git commit -a", there are many shortcut commands in git but unless you know how to use them they can cause more harm than good.

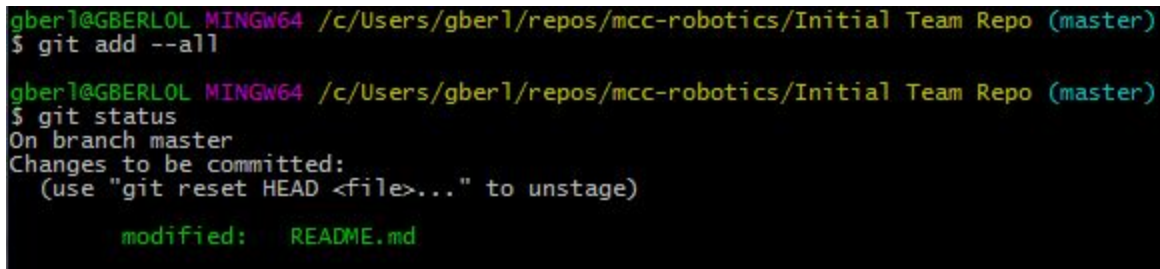
## Step 4: Add changed file(s)

After making changes to files, we need to tell git that we want git to track the changes we've made. To tell git to track the changes we've made we have to add them.

We add changes using the add command, some examples are below

```
$ git add file where file is replaced with the filename we want to add, wildcards are accepted.  
$ git add *.txt here we've specified that we want to add all .txt files  
$ git add --all here we've specified that we want to add all changes.
```

Git allows you to add specific changes down to the filename but for ease you should use the --all option. Another screenshot is shown as an example of what to expect.



```
gberl@GBERL0L MINGW64 /c/Users/gberl/repos/mcc-robotics/Initial Team Repo (master)  
$ git add --all  
  
gberl@GBERL0L MINGW64 /c/Users/gberl/repos/mcc-robotics/Initial Team Repo (master)  
$ git status  
On branch master  
Changes to be committed:  
  (use "git reset HEAD <file>..." to unstage)  
  
    modified:   README.md
```

The git add command doesn't actually trigger a message but if we use git status we can see what's going on. Rather than labeling line by line, since most of the lines are the same format you should simply note the differences from the previous screenshot.

- The message now states "Changes to be committed."
- Git tells us we can undo our add command by using the "git reset" command
- Finally, the message regarding the modified file is now in green because the changes are being tracked. Only changes that have been staged (added) are going to be tracked by git.

## Step 5. Commit staged changes

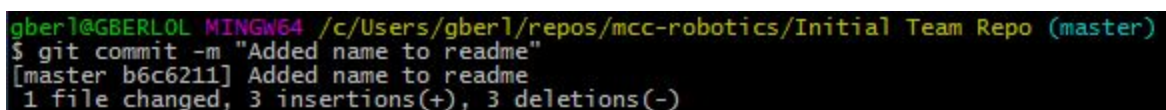
After adding changes to the staging area, you can still make more changes but if you want to include them in the commit you'll have to remember to add them to the staging area as well. Once you've made all of the changes for a particular commit you are ready to make the commitment.

To commit your changes, use the following command.

```
$ git commit -m "Added name to readme"
```

You should notice that we used the -m option which tells git that you want to add a message. All commits require a message, if you do not use the -m option an editor will open where you must enter a commit message. The editor can be confusing so it is recommended that you use -m so you can enter the message in the same line.

Here is another screenshot for reference



```
gberl@GBERL0L MINGW64 /c/Users/gberl/repos/mcc-robotics/Initial Team Repo (master)  
$ git commit -m "Added name to readme"  
[master b6c6211] Added name to readme  
1 file changed, 3 insertions(+), 3 deletions(-)
```

In summary, git is telling us that the commit was made, one file was changed with three insertions and three deletions. Three lines were changed in the README.md file so this is why we see these numbers.

## Note

If you forget to use -m and you get stuck in the editor, use the following sequence to get out.

- Enter a colon using shift+;
- You should see a flashing cursor at the bottom of the window
- Enter the letter 'q' and press the 'enter' key

## Important:

Commit messages should be short but provide enough information to give an idea of what was changed. This makes it easier to find a particular change when looking back at your commits. Also, commits should not contain a large number of changes and changes should all relate to the same topic. If you need to make software changes to navigation and the collector you should break it into two parts and make two commits so that if either of those changes should fail you can easily undo only that small part that was affected.

Commits should also only include small changes because it reduces the chance of having a conflict because you are keeping the remote repository updated often. There is a saying in the git community “commit early and often” and rightly so.

## Step 6. Push committed changes to github

The final step in making a change to project files is to push your changes to github. Pushing makes your changes available for your teammates to see and to update the files they have on their computer or thumb drive.

To push, use the following command

```
$ git push
```

If you encounter any failure messages from git you may need to use the following command the first time.

```
$ git push -u origin master
```

The “-u origin master” option is a way of telling git that from now on when you call git push you want to push to the master branch on the origin repository (origin is the name of the repository on github).

## Note

If you are having trouble grasping the idea of adding, committing and pushing think of this analogy. When a company ships a package, the process might be something like this:

### Git add

- COMPANY: A packer will add (git add) each item to the box (think of the box as the staging area) but they don't want to close (git commit) the box until all items (changes) have been added (git add).
- GIT USER: When you make changes you want to add each item to the staging area, generally you add them all at once but you have the option to add more in case you forgot something.

### Git commit

- COMPANY: After the packer has finished adding (git add) all of the items for shipment, they will seal the box and put a label on it (git commit -m "message").
- GIT USER: After all changes have been added they need to be committed with a message

### Git push

- COMPANY: The box then has to go to the purchaser's home (remote repository) so they ship (git push) it through the postal service (internet) to the purchaser (remote repository).
- GIT USER: This commit has to go to github so you git push to send your commit to the remote site.