# Git Reference

*Spring 2017*

## Contents



## 1 Introduction: What is Git?

Git is version control system. Version control systems allow you to keep track of versions of your code (this can be invaluable when a mysterious bug appears in your code because you can roll back to a previous, bug-free version!), and allow you to easily share code with collaborators (to quote CS 32's Git lab from 2016, version control systems allow you to "easily share code without constantly exchanging emails with subjects like 'code', 'code new', 'code final', and 'code final for real this time, bro'"). CS 138's four projects are completed with a partner, so Git is used to easily share code amongst partners. Furthermore, Git is used as a tool for the CS 138 TAs to share project stencil code with students.

## 2 Setting Up Git

There are two components involved in setting up Git. One component is that you must have an account on a repository hosting service (that is, a website where you can put your code so your partner can access it). The repository hosting service we use in CS138 is called GitHub. The other component is the Git command line tools, which allow you to interface with Git locally.

1. **Creating a GitHub Account**
   To sign up for a GitHub account, open GitHub.com in a browser and choose your username, brown.edu email, and password. Then, select the "free" plan, and then you will be taken to your homepage. Now, you need to verify your email account. Click on the tool icon in the top right corner to get to account settings, then select emails, and click the "verify" button next to your email address. Then, go to your email to confirm.

2. **Installing Git Command-Line Tools (already done on department machines)**
   Follow this guide.

3. **Configuring Settings**
   Open up a terminal (on the machine where you downloaded the Git command line tools and would like to write your code) and run these two commands:

   ```
   git config --global user.name "Ryan Gosling"
   git config --global user.email ryan_gosling@brown.edu
   ```

   The first command sets your name. Remember to put it in quotes! The second sets your email. For this, use the email you used to create your Github account.

   The following two settings are optional, but could be useful:

   ```
   git config --global core.editor editor-of-your-choice
   git config --global color.ui true
   ```

   The first command changes the text editor that comes up when you need to create a commit message. Git's default text editor is Vim, which is a bit different from the traditional text editor. Therefore, if you don't know how to use vim, you should set it to an editor of your choice.

   The second command turns the Git color setting on. This is not necessary but some find that it is helpful and makes things easier to read.

## 3 Cloning a Repository

Git can be used in many ways, but in CS138 we'll be using a central repository (a central location where data is stored and managed) that you and your partner will both pull from and push to. You can make a clone of this repository to get a local copy to work from. Once you're ready to share your changes with your partner, you will be able to sync your changes with the central repository. If your partner has made any changes and synced them with the central repository, you will be able to update your local copy with their changes. The TAs will be creating a repository for you and your partner that will contain the stencil code for all of the projects. To begin working, you must clone this repository to your local repository.

When you have signed up to complete a project with a partner, the TA staff will create a repository for you on GitHub that only you, your partner, and the course staff can access. You should receive a link to this repository in your email, and you can also access it by going to the "Repositories" tab on the CS 138 GitHub page and clicking on the repository named *s17-login1-login2*. Once you are on your repository's page, above the list of files/folders on the right will be a green button thats says "Clone or download." Click this and copy the link for "Clone with HTTPS." In your terminal, navigate to where you want your work for CS138 to be and enter 'git clone [link from Github copied above]'. This will copy the directory structure and files from the Git repo into the location you chose.

# 4 Git Basics

In this section, we will go over some basic git commands and what they allow you to do.

- `git status` tells you the status of your local repository: what files have been added, deleted and changed since your last commit. It also tells you which changes are in staging and which ones aren't. When a change is in staging, this means that it will be included in the next commit. If you have text coloring on, staged changes will be green and unstaged changes will be red.
- `git add <file/folder>` stages the specified file or folder. You can use regular expressions for the argument to include multiple files or run `git add` multiple times on different files. If you stage a file but make a change to it after, it automatically gets unstaged, so you have to `git add` it again.
- `git commit` commits the changes that were staged. Think of this as logging a new version of your code with the set of staged changes. Git will open its text editor so that you can add a message to go along with your commit so that anyone reading the commit messages knows what changed with each commit. Writing thoughtful and informative commit messages can save your partner (and likely you in the future) a lot of headache when trying to figure out what was changed in previous commits. All commits only exist on your local copy of your code until you push.
- `git push` pushes all local commits since your last push to the remote repository. This is what lets your partner access your changes. If someone else has made changes to the remote repository while you were working, Git may insist that you perform a `git pull` before you can push your changes.
- `git pull` pulls all commits from the remote repository that your local version doesn't have. Basically, you're getting all the changes that anyone else who has been working on your project has pushed since the last time you pulled. You should commit all your changes before doing this to avoid losing them.

For the most part, even if you and your partner have been editing parts of your project concurrently, as long as you're not working on the same parts, this system will work well. However, if you do end up in a situation where you and your partner are concurrently working on the same parts of your code, when you perform a Git pull, you could encounter a merge conflict. Git will not complete the pull until you resolve these conflicts. Since Git doesn't know which of your two changes to keep, it creates a version of the file with both sets of changes and asks you to pick one. The syntax for this will be

```
<<<<<<<
(your changes)
=======
(other person's changes)
>>>>>>>
```

Open the file with the conflict, pick the changes you want, delete the special symbols, and then stage and commit the changes. Then the pull can go ahead as planned. One tip for avoiding these

situations is to pull often (maybe before you start working each day), and always pull before you push. This way conflicts won't pile up.

## 5 Conclusion

This guide provides the very barebones basics of Git that you will need for CS 138, but Git has a lot to offer past what is offered in this guide. For clarifications of the information in this guide and to learn more about Git, we recommend stopping by TA hours or search for Git resources online.

---

Please let us know if you find any mistakes, inconsistencies, or confusing language in this or any other CS138 document by filling out the anonymous feedback form:

`http://cs.brown.edu/courses/cs138/s17/feedback.html`.