B. Git and Local Repositories

In 61B, you'll be required to use the git version control tool, which is wildly popular out in the real world. Unfortunately, the abstractions behind it are fairly tricky to understand, so it is likely that you will encounter significant frustration at some point as you learn to use git.

Before you proceed, read sections A-C of Sarah Kim's Using Git Guide

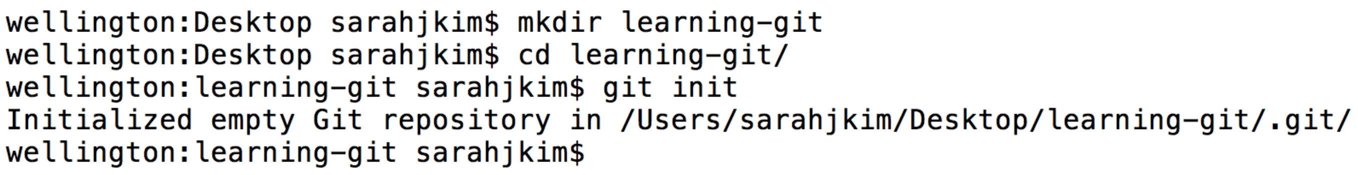
STOP! Do not proceed until you have read sections A through C of the Git Guide.
You do not need to read section D or later.

Git Exercise

Now that you've read the first 3 sections of the git guide, you're now ready to start using Git! Follow along with Sarah's example below. If you typed out all the commands from the tofu example, you may skip this exercise.

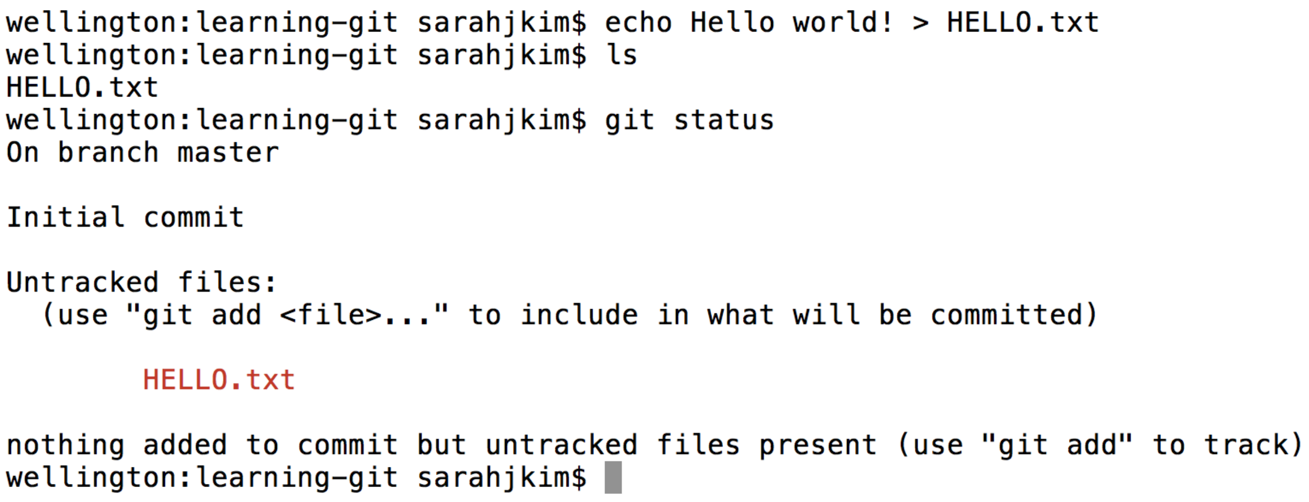
If you'd like more of a challenge, read the direction for each step and guess what the command should be before looking at the screenshots/running the command.

1. Initialize a Git repository called `learning-git`.



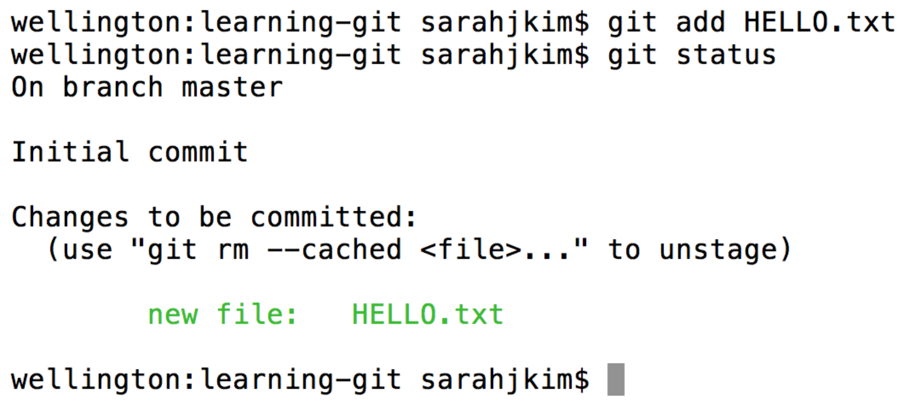
```
wellington:Desktop sarahjkim$ mkdir learning-git
wellington:Desktop sarahjkim$ cd learning-git/
wellington:learning-git sarahjkim$ git init
Initialized empty Git repository in /Users/sarahjkim/Desktop/learning-git/.git/
wellington:learning-git sarahjkim$
```

2. Add a file called `HELLO.txt`.



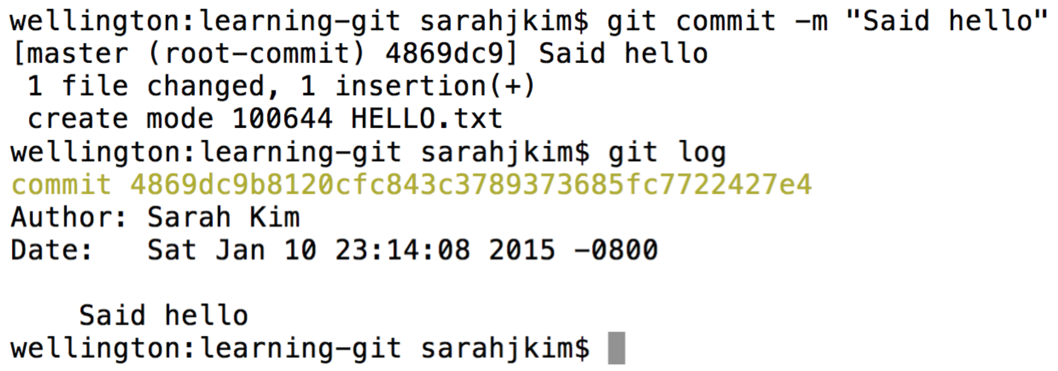
```
wellington:learning-git sarahjkim$ echo Hello world! > HELLO.txt
wellington:learning-git sarahjkim$ ls
HELLO.txt
wellington:learning-git sarahjkim$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        HELLO.txt

nothing added to commit but untracked files present (use "git add" to track)
wellington:learning-git sarahjkim$
```

3. Suppose we want to save the state of this file in git. First we stage it:



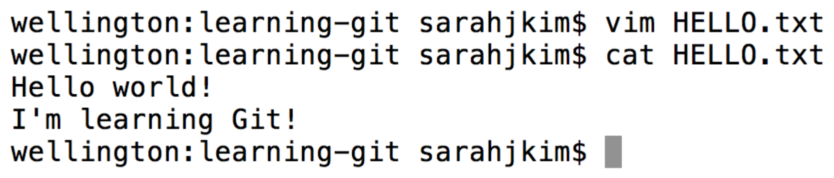
```
wellington:learning-git sarahjkim$ git add HELLO.txt
wellington:learning-git sarahjkim$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   HELLO.txt

wellington:learning-git sarahjkim$
```

4. Now that we have staged `HELLO.txt` , it will be included in our commit. Commit the file with a message of your choice.



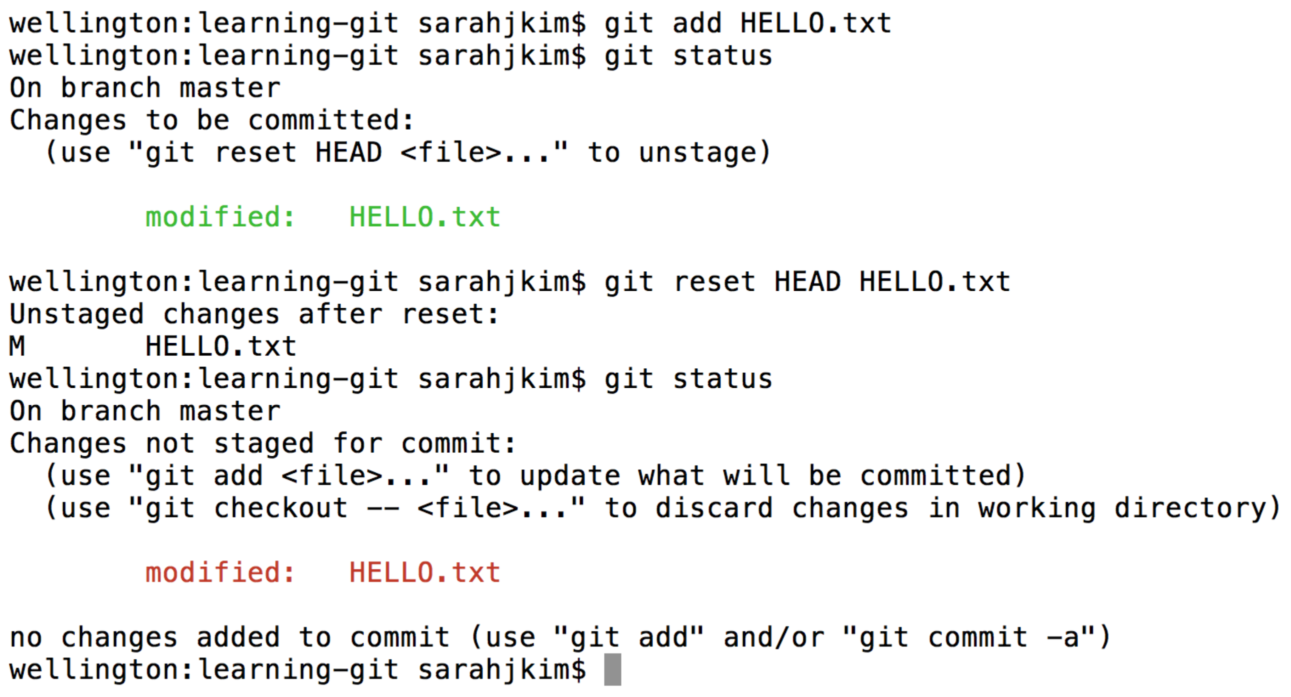
```
wellington:learning-git sarahjkim$ git commit -m "Said hello"
[master (root-commit) 4869dc9] Said hello
 1 file changed, 1 insertion(+)
 create mode 100644 HELLO.txt
wellington:learning-git sarahjkim$ git log
commit 4869dc9b8120cfc843c3789373685fc7722427e4
Author: Sarah Kim
Date:   Sat Jan 10 23:14:08 2015 -0800

    Said hello
wellington:learning-git sarahjkim$
```

5. Let's update our `HELLO.txt` . Here I used a text editor called vim to add some text to the file, and a tool called cat to show the file to you in the screenshot. You can use any text editor of your choice. If you've never used vim, use a different text editor (it's hard to use for beginners). If you get stuck in vim, try googling to see how to get out of it. Getting stuck in vim is a rite of passage.



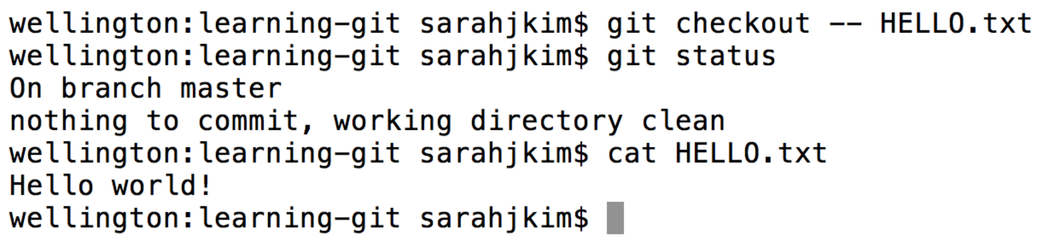
```
wellington:learning-git sarahjkim$ vim HELLO.txt
wellington:learning-git sarahjkim$ cat HELLO.txt
Hello world!
I'm learning Git!
wellington:learning-git sarahjkim$
```

6. If we want to save the change we made in git, first we'll have to stage it. Stage it with the `add` command. Then, suppose we decide we no longer like the change we made, and we don't want to save it in Git. Unstage the file with the `reset` command. *Important*: The reset command does NOT change the actual HELLO.txt file. In terms of our panorama analogy, it only deletes the picture we took of this file!



```
wellington:learning-git sarahjkim$ git add HELLO.txt
wellington:learning-git sarahjkim$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   HELLO.txt

wellington:learning-git sarahjkim$ git reset HEAD HELLO.txt
Unstaged changes after reset:
M       HELLO.txt
wellington:learning-git sarahjkim$ 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:   HELLO.txt

no changes added to commit (use "git add" and/or "git commit -a")
wellington:learning-git sarahjkim$
```

7. Now suppose we dislike the changes we made and want to return the file to its state the last time we committed it -- that is, before we added the extra lines. Discard your changes to `HELLO.txt` since your first commit with the `checkout` command. Here, instead of specifiying a commit ID, we'll use the -- command, which uses the most recent commit by default.



```
wellington:learning-git sarahjkim$ git checkout -- HELLO.txt
wellington:learning-git sarahjkim$ git status
On branch master
nothing to commit, working directory clean
wellington:learning-git sarahjkim$ cat HELLO.txt
Hello world!
wellington:learning-git sarahjkim$
```

It is important that you understand every step of this example. Please ask for help if you are confused about any step.

D. Git and Remote Repositories

We're now ready to finish off the lab. But first...

STOP! Before you proceed, read section D of the Using Git Guide.

There is no need to read sections E or later. Those are for your later reference, and do not need to be read during this lab.

In 61B, you'll be required to submit your code to your personal GitHub repository. This is for several reasons:

- To spare you the incredible agony of losing your files.
- To submit your work for grading and to get results back from the autograder.
- To save you from the tremendous anguish of making unknown changes to your files that break everything.
- To ensure that we have easy access to your code so that we can help if you're stuck, and so that we can grade your code.
- **To dissuade you from posting your solutions on the web in a public GitHub repository**. This is a major violation of course policy!
- To expose you to a realistic workflow that is common on every major project you'll ever work on again.
- To enable safer, more equitable partner collaborations.

Before beginning this section ensure that the name of your GitHub repository in the Berkeley-CS61B organizationmatches your instructional account login. If this is not true, please let your TA know.

Note: You'll need to perform this series of steps to set up your Git repo on each computer you use (e.g. instructional computer, personal computer). If you know that you'll only be using your personal computer, feel free to do this only on your personal computer (and not your lab account).

1. Clone your Berkeley-CS61B organizationrepository.
   - Navigate to the spot in your folders on your computer that you'd like to start your repository.

     ```
     $ cd cs61b
     ```

   - Enter the following command to clone your GitHub repo. Make sure to replace the `**` with your own instructional account login/repo name.

     ```
     git clone https://github.com/Berkeley-CS61B/**.git
     ```

     Copy

     If you'd like to SSH instead of HTTP (and set up your own SSH key), feel free to also do that instead. If you don't know what we're saying, then using https is fine. The advantage of SSH is that you won't have to type in your GitHub password every time you use your repository.

- Move into your newly created repo! (Make sure you do this part, or the rest of the steps below will not work correctly.)

```
$ cd **
```

2. Add the `skeleton` remote repository. You will pull from this remote repository to get starter code for assignments. (Make sure that you are within the newly created repository folder when the continue with these commands.)

   - Enter the following command to add the `skeleton` remote.

```
git remote add skeleton https://github.com/Berkeley-CS61B/skeleton-sp17.git
```

Copy

   - Listing the remotes should now show both the `origin` and `skeleton` remotes.

```
$ git remote -v
```

Working on the Skeleton

1. You must now pull from the `skeleton` remote in order to get the starter code for lab 1. You will also do this when new projects and assignments are released. To do this, use the spookiest command in the whole git toolbox:

```
$ git pull skeleton master
```

What this does is grab all remote files from the repo named `skeleton` (which is located at https://github.com/Berkeley-CS61B/skeleton-sp17.git) and copies them into your current folder.

2. Move the `HelloWorld.java` and `HelloNumbers.java` that you previously created into the `lab1` directory. If you didn't create `HelloNumbers.java` , go back and do Exercise 1.1.2 (see part A of this lab).

3. Stage and commit `HelloWorld.java` and `HelloNumbers.java` .

```
$ git add lab1/*
$ git commit -m "Completed lab1"
```

4. Push these changes to the `master` branch on the `origin` remote repo.

```
$ git push origin master
```

You can verify that this has been successful by checking your repo ongithub.com.