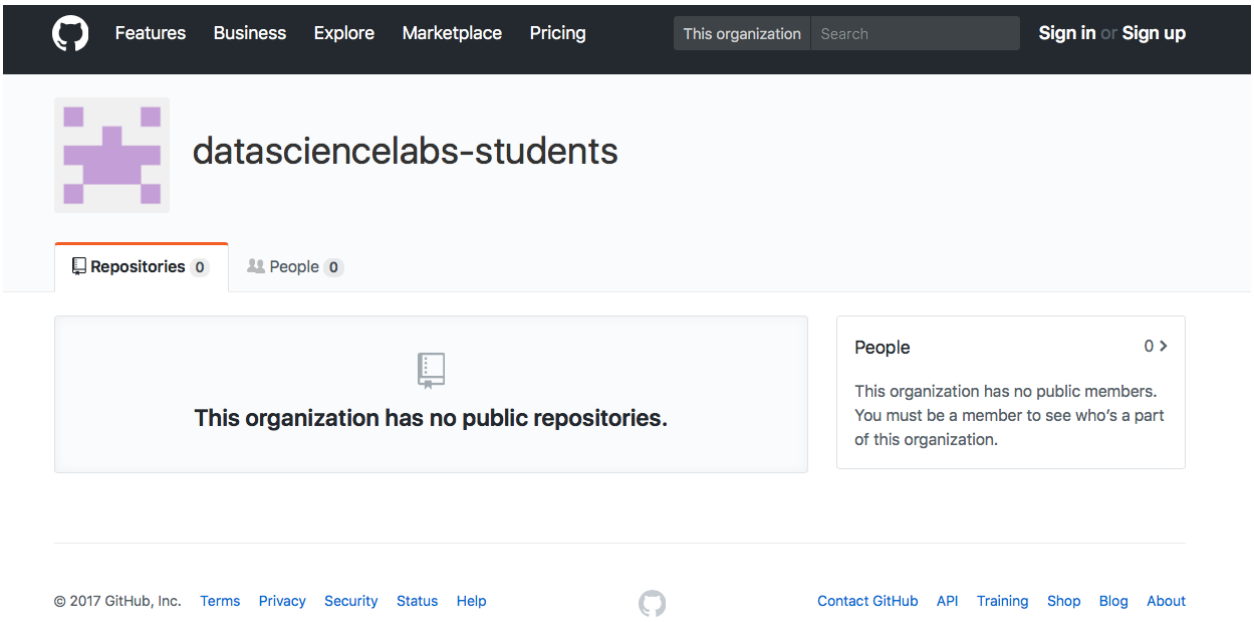
# Homework with Git and GitHub Using the Command Line

You will be using git and GitHub to get your homework assignments, work on your homework, and submit your homework solutions. This tutorial will walk you through that process using git and GitHub through the command line (see the other tutorial for using git and GitHub with RStudio).

## Getting and Working on Homework

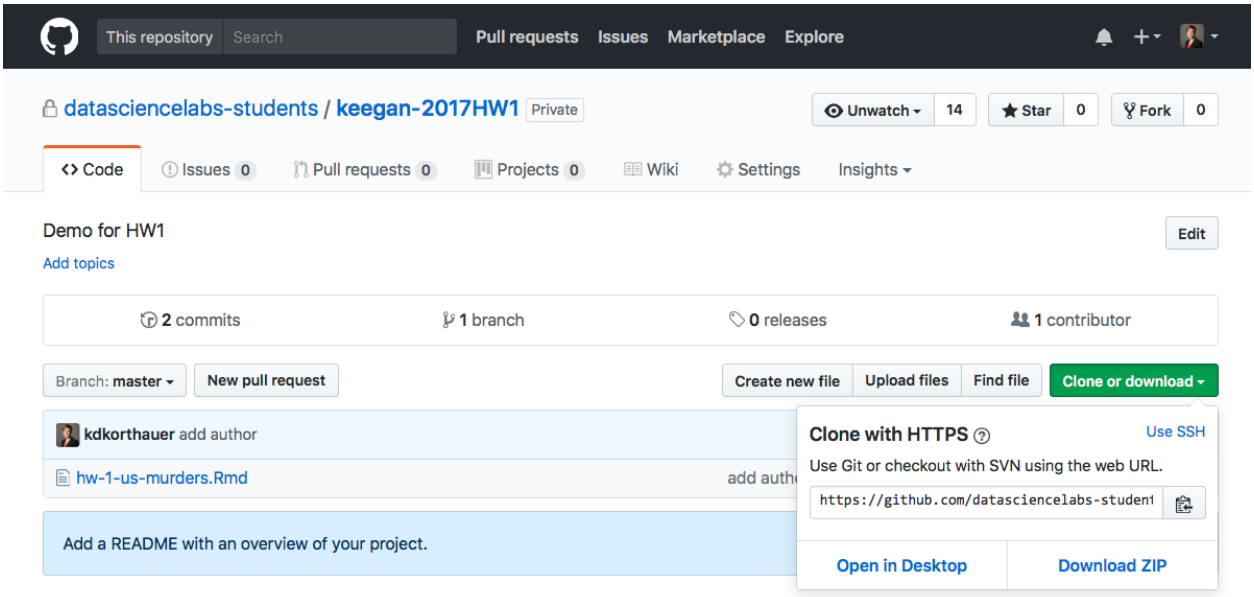
### Cloning your Homework repository

Each of you will be made members of the `datasciencelabs-students` organization on GitHub. This means that your homework repositories all technically belong to us. But you will be granted unlimited access throughout the course!




You will notice when you visit the Data Science Labs' Github page that you can only see repositories with your GitHub username on them. You will get one repository for each homework throughout the semester. When a new homework is released you can go to the corresponding repository to see what is in store. The work flow will be pretty simple.

1. Go to: https://github.com/datasciencelabs-students
2. Click on the repository you want to work on. For example `<your_GitHub_username>-2020HW1` for Homework 1.
3. Copy the link near the top of the page that is revealed after clicking 'clone or download'.




4. Go to your `Terminal` (on Mac) or `git bash` (on Windows), change directories into your BST260 folder.
5. Use `git clone` to clone the repository using the link from step 3. For example:

   `$ git clone https://github.com/datasciencelabs-students/<your_GitHub_username>-2020HW1.git`

```
keegan-3:~ keegankorthauer$ mkdir BST260
keegan-3:~ keegankorthauer$ cd BST260
keegan-3:BST260 keegankorthauer$ git clone https://github.com/datasciencelabs-st
udents/keegan-2017HW1.git
Cloning into 'keegan-2017HW1'...
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 1), reused 6 (delta 1), pack-reused 0
Unpacking objects: 100% (6/6), done.
Checking connectivity... done.
keegan-3:BST260 keegankorthauer$ cd keegan-2017HW1
```

6. You should now see a new directory called `<your_GitHub_username>-2020HW1`. Move into that directory with `cd` (shown in the last line of the previous image).
7. If you type `git status` it will give you the current status of your directory. It should look something like this:

```
keegan-3:keegan-2017HW1 keegankorthauer$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
keegan-3:keegan-2017HW1 keegankorthauer$
```

### Working on your homework

Once you have a local copy of your repository, it's time to get to work!

After writing some of your homework in an `Rmd` file, `knit` it, make pretty plots, and find out some cool stuff about the dataset, it's time to `add/commit/push`. After some work, if you head back to `Terminal` you will see that something has changed when you type `git status`:

```
keegan-3:keegan-2017HW1 keegankorthauer$ git status
On branch master
Your branch is up-to-date with 'origin/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:   hw-1-us-murders.Rmd

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

        hw-1-us-murders.html

no changes added to commit (use "git add" and/or "git commit -a")
keegan-3:keegan-2017HW1 keegankorthauer$
```

You will notice that there is one `untracked file`. In order to get git to track changes in this file we need to add it. So we type :

```
$ git add hw-1-us-murders.html
```

We also need to add the .Rmd file in order to `stage` it (so that it will be included in the next commit). So we type :

```
$ git add hw-1-us-murders.Rmd
```

```
keegan-3:keegan-2017HW1 keegankorthauer$ git add hw-1-us-murders.Rmd
keegan-3:keegan-2017HW1 keegankorthauer$ git add hw-1-us-murders.html
keegan-3:keegan-2017HW1 keegankorthauer$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   hw-1-us-murders.Rmd
        new file:   hw-1-us-murders.html

keegan-3:keegan-2017HW1 keegankorthauer$
```

Now you will notice that the files have turned green and are now labeled as changes to be committed. Now it's time to commit. This is equivalent to `save` in most programs. But what is special about `git` and other version control software is that we can track and revert changes! We also need to give what's called a `commit message`, which will help us keep track of the changes we made when we look at this in the future. Leave detailed messages so that future you will know what you did. Future you will thank you. We will get to this part later. Notice the `-am` flag. The `a` stands for *all*, as in all tracked files, and the `m` stands for *message*.

We do that by typing:

```
git commit -am "This is my commit message, it is very detailed."
```

```
keegan-3:keegan-2017HW1 keegankorthauer$ git commit -m "started answer to probel
m 1"
[master e98d180] started answer to probelm 1
 2 files changed, 178 insertions(+)
 create mode 100644 hw-1-us-murders.html
keegan-3:keegan-2017HW1 keegankorthauer$
```

Cool! Now we've saved our work on our local directory, we can now push our work to Github. Note, we can (and should) do this as many times as we want before the homework deadline. What is great about this is that it will make getting help from your TA easier as well as keeping a copy of your work in the cloud in case your computer crashes, or you accidentally delete something.

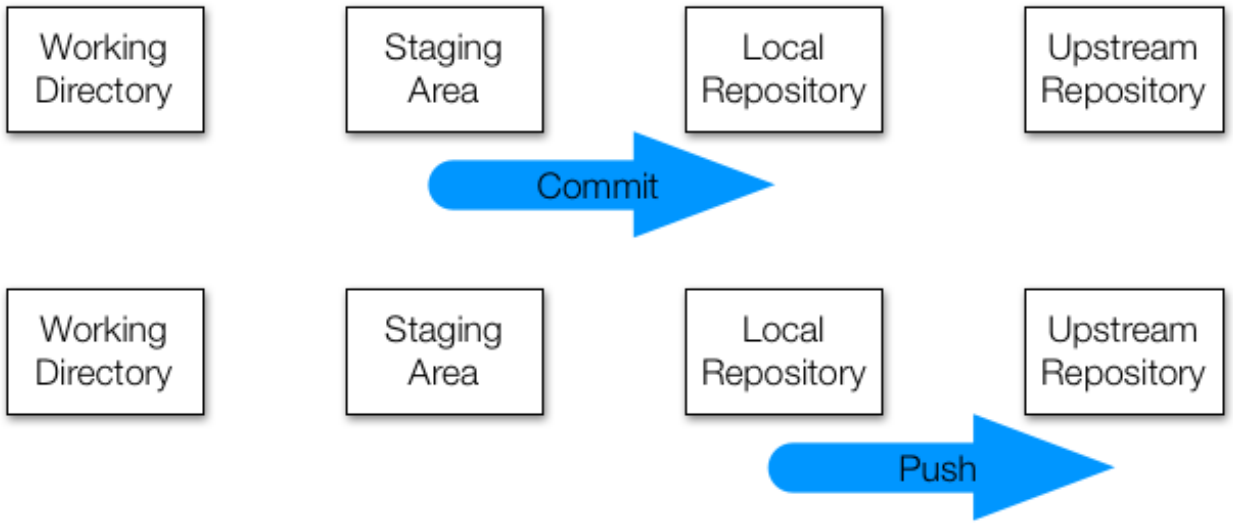
```
keegan-3:keegan-2017HW1 keegankorthauer$ git push
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 244.11 KiB | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/datasciencelabs-students/keegan-2017HW1.git
   9e9ce1c..e98d180  master -> master
keegan-3:keegan-2017HW1 keegankorthauer$
```

**Summary**

To summarize, it is important to do the following steps whenever you finish working on your homework to make full use of `git` and Github as well as generally having the best experience in this class.

1. Work on your homework
2. Add changes to track with: `git add`
3. Commit changes to your local repository: `git commit`
4. Push the changes to your github repo: `git push`

Generally keep this picture in mind whenever you want to do this loop, it is important to only add changed files you care about and nothing you do not care about. If certain files keep popping up in your git status that you will never want to add, e.g. `.Rhistory`, etc, add them to your `.gitignore` to simplify your life. This will keep those files from showing up here. For more info on this see the `version_control.Rmd`




## Late Day Policy

From the course web-page: > Each student is given six late days for homework at the beginning of the semester. A late day extends the individual homework deadline by 24 hours without penalty. No more than two late days may be used on any one assignment. Assignments handed in more than 48 hours after the original deadline will not be graded. We do not accept any homework under any circumstances more than 48 hours after the original deadline. Late days are intended to give you flexibility: you can use them for any reason no questions asked. You don't get any bonus points for not using your late days. Also, you can only use late days for the individual homework deadlines all other deadlines (e.g., project milestones) are hard.

We made this policy because we understand that you are all busy and things happen. We hope that this added flexibility gives you the freedom to enjoy the course and engage with the material fully.

### Some unsolicited advice

To be fair to all the students we have to enforce this late day policy, so we have put together a list of things to consider near the deadline.

Say the homework is due Friday at 11:59 pm.

1. If we do not see any more `commit`s after two days after the deadline we will take the last `commit` as your final submission.
2. Check that the final `commit` is showing on your Github repo page. "I forgot to `push`" is not an acceptable excuse for late work.
3. It may help to add a message like "This is my final version of the homework please grade this" but that's up to you.
4. We will assess the number of late days you used and keep track.
5. You **do not** need to tell us that you will take extra days, we will be able to see the time stamp of your last `commit`.
6. When you are done with the homework, do not `commit` or `push` any more. If you `commit` and `push` after the deadline you will be charged a late day. This is strict.

## Happy `git`-ing