Git for STS 98

Git is a *version-control system*: a computer program for keeping track of files with multiple versions. Just as you might write several drafts of an English paper, a programmer might write several drafts of a program. Git keeps track of which draft is which, and also makes it easy to share the files with others. A git *repository* is a collection of files being tracked by git.

GitHub (`github.com`) is a service for hosting git repositories online. All handouts and assignments will be posted to the class GitHub repository. You'll submit your assignments to your own GitHub repository. We'll also use GitHub repositories for collaboration during group projects.

These notes explain how to use git. Also see `https://git-scm.com/book/en/v2` for an intro book on the subject. Chapters 1–2 apply to this class. The information in chapter 3 is good to know if you plan to do more computing.

If you run into problems, ask for help on Piazza or in office hours. Be specific about what didn't work and what error messages you saw.

Getting Started

You need to install git before you can start using it. You can download an installer for your operating system from `https://git-scm.com/download`.

Once the download is finished, run the installer to install git. Please don't change the default installation settings unless you know what you're doing!

The Command Line

The *command line* is a program that lets you give commands to your computer by typing them in. Long ago, before the mouse was invented, this was the most popular way to control a computer. Modern programmers still use the command line because once you get used to it, it's more efficient than using a mouse. If you find that hard to believe, ask yourself, "Have I ever used a keyboard shortcut?" Sometimes the command line is also called the *terminal*.

We'll use git on the command line. How you open the command line is different for each operating system:

Windows Click on the start menu (the Windows logo in the lower left corner of the screen) and search for "git bash".

Mac OS X Click on Spotlight (the magnifying glass in the upper right corner of the screen) and search for "Terminal".

Okay, now let's go over some command line basics. You're probably familiar with using folders on your computer. Imagine the command line as a fluffy animal that moves around your computer, doing whatever you say. The command line can only be in one folder at a time. The folder it's currently in is called the *working directory*.[1]

To check the name of the working directory, enter the command `pwd`. The command line will respond with something like

```
$ pwd
/home/noodles
```

The first line starts with `$` to show the command you entered. The second line is a list of folders separated by `/`. This is called the *path* to the working directory. You'll probably see a different path. This path means the working directory is the `noodles` folder, which is inside the `home` folder.

To get a list of folders and files inside the working directory, enter the command `ls`. The command line will respond with something like

```
$ ls
ramen  spaghetti  vermicelli
```

The output may be color-coded to indicate which are folders and which are files.

Finally, to change the working directory, enter the command `cd` followed by a path. There is no output for this command. For example,

```
$ cd ramen
```

changes the working directory to the `ramen` folder. Use the path `..` to change to the directory above the working directory. This is how you move around your computer with the command line.

[1] *Directory* is just a fancy word for folder.

Git Config

You need to tell git your name and email, so you get credit for the work you do. On the command line, enter

```
$ git config --global user.name "YOUR NAME"
$ git config --global user.email "YOUR EMAIL"
```

Now you're ready to use git.

Pulling Lecture Notes

The course lecture notes are at https://github.com/2016-ucdavis-sts98/notes.

To get a copy of the lecture notes, open the command line and enter

```
$ git clone https://github.com/2016-ucdavis-sts98/notes.git
```

The `git clone` command downloads a repository from the web to a folder inside the working directory. In this example, the notes are downloaded to a folder called `notes`.

From time to time, the repositories you've cloned will be updated online. This includes the lecture notes. You don't need to clone a repo again to get the updates. Instead, open a command line and change the working directory to the repo folder. For the lecture notes, you'd change to the `notes` folder. Then enter

```
$ git pull
```

The `git pull` command downloads all of the updates to the repo. If you make your own changes to the repo, pulling updates will not always work correctly.

Pushing Assignments

When a new assignment is ready, a link will be posted on Piazza. Accepting the assignment will create a new repo for you to work in. The steps for cloning the repo, making changes, and then submitting your changes will be similar for every assignment.

Let's go through an example for Assignment 1, assuming your GitHub username is "stsrocks". When you go through these steps, you should use your real GitHub username. After accepting the assignment, your new repo will be called `assignment-1-stsrocks`. To start working, clone the repo:

```
$ git clone https://github.com/2016-ucdavis-sts98/assignment-1-stsrocks.git
```

Instructions for the assignment are in the file `README.md`. You can also view this file on GitHub. Follow the instructions to complete the assignment.

There are a few git commands you need to know to save your work. The `git add` command tells git about changes you've made to a file. You can tell git about several files at once by separating their names with spaces. For example,

```
$ git add radish.txt celery.txt
```

tells git about changes to `radish.txt` and `celery.txt`. The `git status` command will tell you which changes git is tracking. Please **DO NOT** tell git to track data files (`.rds`, `.csv`), because this makes it take much longer to clone your repo.

After running `git add`, the `git commit` command tells git to create a commit. A *commit* is a checkpoint for your work that you can go back to at any time. When you make a commit, you should include a message describing what you changed. Here's how:

```
$ git commit -m "Wrote about radishes and celery."
```

Don't forget to put quotes around the message and `-m` before it.[2]

Finally, you need to publish your work to GitHub. Enter

```
$ git push origin master
```

The `git push` command uploads your work to the web. You should do this last, after committing all of your work.

[2] If you do forget `-m`, git will open the Vim text editor for you to write a commit message. Vim is useful but challenging to learn. To quit Vim, press the keys `Esc` `:` `q` `!` `Enter` in order. This will also cancel the commit.