Git Quick Start Guide

For this class, we will be using the Git (http://git-scm.com/), a distributed source control system, to manage homework submission. This document will introduce a quick guide to the commands and tools that you will need to know to be able to successfully use the system in this class.

Installation

If you don't already have Git configured on your machine, you may download the latest version here: http://git-scm.com/downloads. But, if you have a MAC, I find it's easier to just install the Xcode command-line tools.

We recommend using the command-line for Git commands (Git Bash on Windows) unless you are already familiar with another client.

Setup

The following will configure the name/email that will be displayed in your commit history.

```
$ git config --global user.name "Harry Q. Bovik"
$ git config --global user.email bovik@cs.cmu.edu
```

Cloning your repository

```
$ git clone https://github.com/cmu-webapps/[ANDREW-ID].git
```

You will be prompted for your GitHub username and password. This will create a directory on your local machine with your Andrew ID as the name. You will do all your work in this directory.

If you have set up two-factor authentication on GitHub, you can use the above clone command, but you will need to provide a personal access token instead of your password. You can create a token on GitHub. (Click top-right on your picture and then Settings -> Developer Settings -> Personal access tokens -> Generate new token.)

However, the best way to authenticate to GitHub is by using SSH. If you have an SSH Public Key (typically in ~/.ssh/rsa_id.pub) and have registered this public key with GitHub, then you can clone your repo (and any of the class repos), without having to provide your user id and password, using URLs with a different structure:x

```
$ git clone git@github.com:cmu-webapps/[ANDREW-ID].git
```

If you would like to register an SSH Public Key with GitHub, see the Using SSH Credentials, below.

Uploading homework

There are two steps needed to upload your homework to GitHub: committing and pushing.

Committing saves a snapshot of the selected changes you have made since the last commit. The process of committing consist of two commands; **`git add`** tracks the specified file and adds it to the *staging area*. You can check what files have been edited or are in the staging area with **`git status`**. **`git commit`** saves all of the changes in the files added to the staging area.

```
$ git add file1 file2 ... % Adds file1 and file2 to staging area.
$ git commit -m "Fix bug concerning user password reset"
```

If you omit the **`-m`** flag, then your default text-editor (typically ViM or Emacs) will be opened after the **`git commit`** for you to edit your commit message. Save and quit to complete the commit.

You should make sure to **commit often** (typically each time you complete a feature) and write **descriptive commit messages**.

Pushing saves all the commits created since the last time you've pushed onto the remote server that you cloned from (in this case, the GitHub repository).

```
$ git push
```

To check that your homework has been successfully submitted, visit your GitHub repository at `https://github.com/cmu-webapps/[ANDREW-ID]` and make sure that the files on the web page match the state of the files you would like to submit.

Grading homework

You must grade your own homework using the class AutoGrader. To do this, go to https://grader.cmu-webapps.org and click on “grade”. The AutoGrader will pull your repo from GitHub and grade the version you have just committed (main branch only). You may fix your errors, add, commit, push, and regrade as often as you wish (unless you’re unreasonably consuming too many server resources). Note that your “submission” time is based on when you click grade (or regrade) and not based on when you commit or push your repo. (See the Syllabus for details on late homework penalties.)

Misc.

All commits are saved in the Git log. The log contains information on all commits ever saved in the repository. To access this log, type:

```
$ git log
```

As mentioned above, you can always check which files have been edited/staged using **`git status`**.

```
$ git status
```

Using SSH Credentials

We highly recommend using SSH credentials to prove your identity to GitHub. This way you will not need to use your GitHub password when cloning (or pulling or pushing) repos.

You can see the SSH public keys someone has registered with GitHub by going to:

```
https://github.com/<github_username>.keys
```

For example, check mine out at `https://github.com/je0k.keys`. (Note: My GitHub username happens to be the same as my Andrew ID.)

Here are the steps to create and register a SSH public key:

- See if you already have an SSH Public Key in `~/.ssh/id_rsa.pub`. If not, generate one with the following command:
  ```
  ssh-keygen
  ```
  Note: You may leave all the passwords and passphrases blank.
- You should now be able to display your public key by printing the contents of:
  ```
  ~/.ssh/id_rsa.pub
  ```
- Then log into GitHub and tell it what your public key is by clicking at the top-right on your picture and going to the Settings -> SSH and GPG -> New SSH Key and then pasting the contents of your `~/.ssh/id_rsa.pub` file Key textbox (and also provide a description).
- You should now be able to visit `https://github.com/<github_username>.keys` and see your public key.

Now you can use SSH-style URLs when accessing repos, without providing a username or password:

```
git clone git@github.com:cmu-webapps/[ANDREW-ID].git
git clone git@github.com:cmu-webapps/django-intro.git
```

Team Repos

Later in the course, when you work on your group project, we will provide you with a team repo. Everyone on your team will be able to pull and push this team repo, so you will any to follow additional procedures to avoid conflicts as you and your teammates concurrently create and modify files.

You will clone your team repo with one of these commands:

```
$ git clone https://github.com/cmu-webapps/[TEAM].git
```

Or if using SSH:

```
$ git clone git@github.com:cmu-webapps/[TEAM].git
```

Then you will do the development for your project in the directory created by cloning.

Branches

When you have multiple people developing code together, you should make your changes in separate branches. This allows you to make changes insolation from your teammates and then once you have you changes finished, you can merge them into the main branch in a controlled manner. GitHub has support for merging code and often the code merges can be done automatically if there are no conflicts.

To create a new branch run this command:

```
$ git branch [name_of_new_branch]
```

Then switch to using this new branch:

```
$ git checkout [name_of_new_branch]
```

You can run `git status` and it will show you that you are working on this new branch (instead of the main branch). Develop your code as usual, including making periodic intermediate commits. When you're ready to merge your code into the main branch, you will need to push it to GitHub wirh this command:

```
$ git push --set-upstream origin [name_of_new_branch]
```

Once you've used the above command to set up the push, you can just use `git push` in the future. Also, you will now be able to see your new branch on GitHub. When you're ready to merge your changes into your main branch, create a pull request for your branch. GitHub will often be able to merge the files automatically (in particular, if the files you changed do not conflict with the files your teammates have changed).

Additional resources

Listed below are some additional resources for Git that you may find helpful:

- **Code School - Try Git** (https://try.github.io). 15-minute tutorial covering the Git's main features.
- **Pro Git** by Scott Chacon (http://git-scm.com/book). Excellent free online book that contains an overview of all the functionalities and tools of Git.