Version Controlling Your Thesis With Git

University of Virginia Graduate Seminar

Christopher J.R. Lloyd

December 3, 2020

1 Installing

- For Windows download and install from https://git-scm.com/download/
- On Mac you can check if it is already installed by opening the terminal (to open finder open the /Applications/Utilities folder, then double-click Terminal) and running:

  ```
  git --version
  ```

  If it is installed it will tell you the version, otherwise it will ask you if you would like to install it.
- On Debian-based Linux (including Ubuntu) you can install it via

  ```
  sudo apt install git-all
  ```

2 Accessing Git

Once Git has been installed one can access it from the terminal / command line. On Windows the command line (CMD) is accessed easily by holding "Windows Key" and "r" together to open the run menu. Then type in "cmd" and press "enter". While on Mac the terminal is opened as explained above. One can check to see if git is properly installed by running

```
git --version
```

In my case this outputs

```
git version 2.24.3
```

The way we will interact with git is by calling commands of the form

```
git command
```

where command is some valid git command. For instance, we have already seen version is valid git command. The next useful command is help:

```
git help
```

this will show a list of the valid commands.

3 Setting Up a Repository

Suppose you have a LaTeX project folder called `test` containing a LaTeX file `test.tex`. To turn this folder into a git repo, navigate to the folder inside of the terminal using the cd (which stands for change directory), and then run the command `git init`. For me (on Mac) this looks like

```
chris test % git init
Initialized empty Git repository in /Users/chris/Desktop/test/.git/
chris test %
```

Now a git repo has been created and we can start tracking files. Really this means a hidden folder named .git has been created inside of the test folder where git will keep all of the necessary data to track our project.

4 The First Commit

Making a commit in git is like etching the current status of the project into stone. It's not that you cant make changes after this, but there will always bee a record of exactly how this folder was at the instance of commit stored in this git repo. To look around at what is going on inside of the git repo one uses the git status command:

```
chris test % git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.aux
        test.log
        test.pdf
        test.synctex.gz
        test.tex

nothing added to commit but untracked files present (use "git add" to track)
```

This tells us what is and what isn't being tracked by the Git repository. We haven't started tracking anything yet. Notice there are a bunch of files generated by LaTeX that we really don't care about. We never want to start tracking these files since they are auto-generated. We really just want to track the test.tex file. We can tell Git to start caring about this file via git add test.tex and then running git status to see what has changed:

```
chris test % git add test.tex
chris test % git status
On branch master

No commits yet

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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.aux
        test.log
        test.pdf
        test.synctex.gz
```

Now we see that the test.tex is now ready to be committed. You only ever need to use the git add command once to tell the git about the file, after which Git will never forget.

Now we use the git stage test.tex command to tell git we are about to commit this file. This will seem slightly redundant for our single file example, but this becomes important when one is committing multiple files that all have a related changes, then one runs the git commit –m "Message". The extra option –m specifies the message of the commit, which can be whatever you want. This is useful for when you are looking back at the changes over

time. If one fails to specify the message here, they will be dropped into the default system text editor, probably vim or emacs which could be unpleasant.

```
chris test % git stage test.tex
chris test % git commit -m "First Commit"
[master (root-commit) 88e2b3c] First Commit

 1 file changed, 5 insertions(+)
 create mode 100644 test.tex
chris test %
```

In the above my message was "First Commit". We can now inspect the previous commits using git log:

```
chris % git log
commit 88e2b3cce2f264d3d658785b5bcad876b2c1342a (HEAD -> master)
Author: Chrstopher Lloyd <chris>
Date:   Thu Dec 3 14:26:46 2020 -0500

    First Commit
```

This is the most basic use of git. Now after you make changes to your file, stage the file, and commit it again with the relevant message, and this will add to your repo. Notice the long string of numbers and letters, this is the unique id of this commit. It is generated via a hashing function of the input (which in itself is mathematically interesting).

5 Pushing and Pulling and GitHub

So far this repo is only stored locally on your computer inside of the test folder. We wish to store copies of our repo somewhere else. A good choice is GitHub. Now after Microsoft bought GitHub any user can store private repositories. After making account on https://github.com/ you can make a new repository on the website, set it to private, and maybe name it say "test". Once this has been done you can push the local repository that was just created to github via the commands they present:

```
git remote add origin https://github.com/cjl8zf/test.git
git branch -M master
git push -u origin master
```

Note: In Fall 2020 GitHub changed the default branch name of the repos to main instead of master, but the git still uses master as the default so here we had to change main to master to be consistent with thee defaults of Git itself.

```
chris test % git remote add origin https://github.com/cjl8zf/test.git
chris test % git branch -M master
chris3 test % git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 279 bytes | 279.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/cjl8zf/test.git
   [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

This will prompt you for your GitHub password. This will only need to be done one time! Now you should see the tracked file has appeared on GitHub.

The real utility of Git comes from the way it allows multiple authors / programmers to collaborate on projects without over-writing other peoples work. To pull changes from the GitHub you would run git pull (only do this

after you have committed everything you are working on currently) this will then download any changes and incorporate them into your repository. Although, there will be no changes to our test.tex repo. Suppose we have made another change, committed it, and then want to add it to the GitHub repo. We would then use the git push command. For example after making a small change in git test

```
chris test % git stage test.tex
chris test % git commit -m "Made a change"
[master 4787a59] Made a change
Committer: Chrstopher Lloyd <chris>

1 file changed, 3 insertions(+), 1 deletion(-)
chris test % git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 330 bytes | 330.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/cjl8zf/test.git
   88e2b3c..4787a59  master -> master
```

Now the changes we have made are backed up safely on GitHub!

6 Cloning a Repo and Grabbing a Thesis Template

There are many interesting projects on GitHub and due to the distributed nature of git you can copy the entire project directly onto your computer with one command: git clone. For us we will download a copy of a nice UVA thesis template made by Chao Gu available at https://github.com/asymmetry/uva-thesis-template. Run the command:

```
git clone https://github.com/asymmetry/uva-thesis-template
Cloning into 'uva-thesis-template'...
remote: Enumerating objects: 30, done.
remote: Total 30 (delta 0), reused 0 (delta 0), pack-reused 30
Unpacking objects: 100% (30/30), done.
```

this will download a working thesis template. You can now edit the files as you wish and commit to this project. You would setup a new GitHub project and set that as the origin by following the GitHub setup steps above.

The source code for these notes can be cloned via:

```
git clone https://github.com/cjl8zf/uva-grad-sem-git-guide
```

7 Overleaf

There is background integration of Git inside of Overleaf. In the left side bar there is a Git option which allows you to clone the Overleaf project to your computer. You can then push to the origin (which will already be set up for you)!