

Git

Learning to use version control is absolutely essential for a programmer. If you are not using some kind of version control system already, then you should immediately start using one. Version control ensures that you can always retrieve an older version of your code in case you need to retrieve some code you have deleted. Storing your repositories in the cloud ensures that you won't lose your code, which should be one of your most precious possessions. I find this so convenient, I use version control not just for code but also for LaTeX papers and presentations I write, and for other documents I keep in my home directory.

Git is the preferred tool for many developers because it handles [distributed version control](http://git-scm.com/book/en/Getting-Started-About-Version-Control) [so well](http://git-scm.com/book/en/Getting-Started-About-Version-Control). Outside of university projects, it is most common to work on teams when writing code, so a tool like Git that provides easy collaboration is required. It is quickly becoming a de facto standard so it is useful to understand at least its basics. Git is supported by most online software repositories, like [GitHub](https://github.com/) [\(https://github.com/\)](https://github.com/) and [BitBucket](https://bitbucket.org/) [\(https://bitbucket.org/\)](https://bitbucket.org/).

Learning Git

The best place to learn Git is to use the free [Pro Git book](http://git-scm.com/book) [_\(http://git-scm.com/book\)_](http://git-scm.com/book). Read Chapters 1 and 2 and you will learn what you need to be productive for individual projects in this class.

You can also use this interactive [Git tutorial](https://try.github.io/levels/1/challenges/1) [_\(https://try.github.io/levels/1/challenges/1\)_](https://try.github.io/levels/1/challenges/1). Here is a useful [Git cheatsheet](https://www.git-tower.com/blog/git-cheat-sheet/) [_\(https://www.git-tower.com/blog/git-cheat-sheet/\)_](https://www.git-tower.com/blog/git-cheat-sheet/).

Configuration

To begin, I recommend doing some configuration for your name, email, and editor:

```
git config --global user.name "Daniel Zappala"
git config --global user.email daniel.zappala@gmail.com
git config --global core.editor <emacs or nano or vim>
```

The commands I use most often:

Create Repositories

- `git init` : initialize a new repository
- `git clone <url>` : clone a repository

Work With A Local Repository

- `git log` : view all previous commits

- `git log -p` : view all previous commits with the diffs
- `git status` : list changed and new files
- `git diff` : view changes to files
- `git add .` : add all new files to a repository
- `git commit -a` : commit all changes to all files
- `git tag <name>` : mark current commit with a tag

Update a Repository and Publish Changes

- `git remote add <remote> <url>` : add a remote repository
- `git pull <remote> <branch>` : download commits from a remote repository and merge them
- `git push <remote> <branch>` : upload commits to a remote repository
- `git push --tags` : push tags to a remote repository





In addition to these commands, it is useful to create a `.gitignore` file containing paths and file extensions you would like to ignore. For example, this might include backup copies created by your editor, configuration files that include important passwords, or (later in the course) node.js modules.



Branching and Workflow (Advanced Material)



Once you start participating in shared projects, then Chapters 3 and 5 of Pro Git will help you with branching and workflow models. I also recommend [this example workflow](http://nvie.com/posts/a-successful-git-branching-model/) (<http://nvie.com/posts/a-successful-git-branching-model/>), which you can trim down for smaller projects. There are some good [thoughts about workflows](https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow) (<https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow>) by Atlassian.



The commands I use most often:


- `git branch` : list all branches
- `git checkout <branch>` : check out an existing branch
- `git checkout -b` : create and check out a new branch








<https://byu.instructure.com/courses/5724/modules/items/362832>





<https://byu.instructure.com/courses/5724/modules/items/366435>



<https://byu.instructure.com/courses/5724/modules/items/362834>



<https://byu.instructure.com/courses/5724/modules/items/366436>


<https://byu.instructure.com/courses/5724/modules/items/362835>




<https://byu.instructure.com/courses/5724/modules/items/362827>



<https://byu.instructure.com/courses/5724/modules/items/362932>

✓  (<https://byu.instructure.com/courses/5724/modules/items/362933>)

✓  (<https://byu.instructure.com/courses/5724/modules/items/362935>)

 (<https://byu.instructure.com/courses/5724/modules/items/412348>)

✓  (<https://byu.instructure.com/courses/5724/modules/items/366437>)

✓  (<https://byu.instructure.com/courses/5724/modules/items/366438>)

✓  (<https://byu.instructure.com/courses/5724/modules/items/415270>)