Weld Lab git/GitHub Primer

Peter Dotti

1 Introduction

This document is intended to establish basic git and GitHub practices in the Weld Lab and to be a simplified introduction to using git and GitHub. The hope is that implementing these practices will enable the following:

1. Editing code without fear of disrupting a working version

2. Easily update versions of code once a testing version of the code is established

3. Inherent record keeping and version recording as automatically carried out by the git software

4. Backup versions (current and historical) of code as saved on both GitHub and the Citadel server

5. A unified procedure so that accessing and using other lab member code is easy

While there is desktop software for GitHub (and, presumably, git more generally), I found it not to be sufficiently intuitive, robust, or easy to justify using it over command line inputs, so this document will focus on use of command line git.

Hopefully it will be easy and painless to do these things. Please email / slack message me (Peter) if you have suggestions to improve this document in the future. Additionally, I hope to use git on this document itself for easy editing.

2 An Important Reference

If you find that this primer is lacking some piece of information that you would like, I strongly recommend using the online "git Book" from https://git-scm.com/book/en/v2. I found it to be very well written and well organized, although it will certainly have more details than you need (although, you will likely not need most details past the fourth chapter, unless you wish to do more advanced things).

3 Using git on your Computer

Git revolves around the use of "repositories." A repository is just some directory (folder) on your computer with a hidden git file in it that designates it as a repository and allows you to set and keep track of files.

You are able to change the files in the repository however you'd like, using whatever code editor or program you'd like, but you will regularly want to make a "commit" with git so that the version

of the repository is saved. Additionally, you will want to occasional use the "`push`" command to use the current version of the repository on your computer to update some remotely saved version of the repository (i.e., you will use `push` to update a stored version of the code on GitHub and the Citadel)

From here on, I will assume that the reader has some command line tool with git installed on it (instructions on how to do that here: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git). I will also assume that the reader knows how to change directories and get directory path names in the command line.

3.1   Making and Using a git Repository on your Computer

Once you've navigated to the directory that you want to initialize as a repository, you run the command

```
git init
```

At this point, you can fill the directory as usual (assuming it didn't already have code in it). If you want something to test basic git functionality, I would recommend saving a text file in the directory and editing it.

At this point, you can use the command

```
git status
```

And it will show something like

```
On branch master

No commits yet

Untracked files:

  <A list of the files in the directory>
```

The general meaning of this is that git knows that you have files in the directory that you might want to use as a repository, but they are currently "untracked" and git is waiting for you to indicate which files you want it to keep track of edits for (and moreover, which files it should copy over to an external server such as GitHub or the Citadel when you ask it to do so).

To indicate which of the files in the list you want it to track, use the command:

```
git add <filename>
```

for each of the files you'd like. If you want it to track all of the files, you can conveniently use

```
git add --all
```

4 Using git with Remote Repositories (e.g. GitHub and the Citadel)

```
Test
```