Intro to Git

Rahul Shah

Compiled on Sunday 24th June, 2018 at 21:22

1 Introduction

Some of you coming from FLL might remember how annoying it was when the laptop with all the code died and the only backup was on a flash drive at someone else's house. Since the code we write is significantly more complex and we are working together with more people, it can difficult to keep all of our code organized and up to date.

Git is a type of *source control* that we utilize to collaborate and organize our code. Git is a tool that allows us to make sure that all of our code is up to date and provides us with a way for multiple people to work on the robot code at the same time. That being said, git can be difficult to get used to and that is why you are here.

2 GitHub

GitHub is the website that we use to store all of our code, both new and old. If you do not already have an account, please create one now at: github.com

GitHub is basically Google Drive except it is much easier to use and allows collaboration. On the github website, one can view all versions of a project. For example: if you feel like you want to change the code a bit, but are not sure that it will work, you'll definitely want to leave the option to go back to the previous working code if this new thing fails. GitHub allows us to do that in a user-friendly manner.

3 Git Commands

```
git clone
git status
git add
git commit
git push
git pull
git remote
git checkout
git init
```

3.1 git clone

git clone allows you to download a github repository. The syntax is as follows:

```
git clone https://username@github.com/owner-username/repo-name
```

For example, if i was to clone the 2018 repository, since my username is *coder3462*, I would type

```
    git clone https://coder3462@github.com/Team1923/Power_Up_2018
```

into my terminal (or git bash on windows).

3.2 git status

Displays files or folders that have differences between the file on your system and the current file on committed to master on github, and paths in the working tree that are not tracked by Git (and are not ignored by the .gitignore).

3.2.1 git add

Adds files so that they are ready to be committed. While individual files are added like this:

```
git add Filename.java
```

It can be tedious to do that for all files, especially if you have made a lot of changes to a lot of files. A helpful shortcut is:

```
git add .
```

The . is a wildcard which adds all files to git.

3.2.2 git commit

Stores the current contents of the filesystem in a new commit along with a log message from the user describing the changes.

Syntax:

```
git commit -m 'fixed floating point error in ClassName.java'
```

3.2.3 git push

Updates the remote repository on github.com Syntax:

```
git push
```

If you have changed the remote then the syntax is:

```
git push origin master
```

3.2.4 git pull

If someone else has pushed, then to get their changes, you must run

```
git pull
```

which will add their changes to your filesystem

3.2.5 Merge Conflicts

Let's say that you were editing

```
MyClass.java
```

Now let's say that your friend James edited

```
MyClass.java
```

and pushes. When you try to push, you will be alerted of merge conflicts. After running

```
git pull
```

you will see the following text ¡¡¡ *HEAD*. The problem is that git does not know which version of the file should be committed. What do you think should be done?

3.2.6 git remote

cd to a git repository (go to the folder of a cloned git repo), then run the following command in git bash or terminal:

```
  git remote -v
```

3.2.7 git checkout

```
  git checkout from-branch-name path/to/the/file/you/want
```

During build season, we have 2 branches:

1. master - all of the code on master works completely
2. beta - not all of the code on beta works completely

4 Review

By this point, you should be able to...

- clone a github repo
- make some changes and push them from the command line

Please attempt to do all of these things, especially the last bullet, to ensure that you understand the basics of using git.

5 Required Coding Style

Your code should be indented with 2 spaces. PR's (Pull Requests) that contains code that is indented with tabs or 4 spaces will not be accepted.