git_cheat_sheet

Cody Flagg

February 24, 2016

Vocabulary

- *Branch* : **everyone** is on a branch; if someone says otherwise then clearly they're an ignorant liar. Local branches can be re-merged with the "trunk" of the repository (called the "remote master"), or they can go off and develop work independently of the master for an indefinite amount of time (forever, even); these are usually called "development branches" in in Stack Exchange parlance.
  - https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell
- *Remote Repository* ("Remote") : the shared repository in the "cloud" i.e. on the GitHub web server
- *Local Repo/Branch* : your copy of the Remote Repository at the time you *last* cloned/pulled
- *Stage* : a git concept; essentially an intermediate purgatory stage before you have decided to "commit" your changes; you stage things with `git add`
- *Commit* : a git concept and command, `git add` that *saves your work locally*; committed files/changes must be *pushed* before they can be shared with others

Navigate Git Bash

- anything in angle brackets denote a in an example, not what you should literally type in the Git Bash window
- `cd` : changes the directory
- `cd C:/Users/MikeP/Documents/GitHub/` : sets the directory to your GitHub folder (if it exists)
- `cd devTOS` : will move to the "devTOS" folder if it exists within your current working directory
- `cd ..` : moves you "up" one directory
- Pressing tab after typing `cd` will be your best friend:
  - If you type `cd` and press tab, all files in the directory will appear
  - if you type a partial folder/file name `cd dev` + **TAB** will "tab-complete" to `cd devTOS`
- `ls` : list files in directory
- `rm <file name>`

Setting up a Repository

Git Clone

- You need to "clone", or copy, a repository ("repo") before you can use, modify, or push the files in it:

1. Navigate to the directory folder you want the repo to go into
2. execute the command: `git clone https://github.com/<repository_user_name>/<repository_name>.git`

e.g.

```
git clone https://github.com/NEONInc/devTOS.git
```

After you have a repository set-up: essential operations

- Execute the Add, Commit, Push, and Pull commands within a specific Git repo i.e. *devTos*, not within the root Git directory
- *ALWAYS FOLLOW THIS WORKFLOW* : add > commit > pull > push. That's it.
- You will avoid 99% of problems if you always add and commit the changes on *your* computer first, then pull updates from the remote repo, THEN push your changes.

Git Status

- Check which files have been added, modified, deleted, staged, and/or commited with `git status`
- This is a good way to check whether you have failed to commit files before you `git pull` or `git push`

Git Add

- You can add individual files to the staging area via `git add <filename>` e.g. `git add newSpreadsheet.xls`
- But it's easiest to just add all files, especially when there are many e.g. `git add --all`
  - You will find many different ways to execute the same arguments in Git, much like R e.g. `git add -A` does the same thing as `--all`

Git Commit

- All commits require a message, even if it's blank. It can seem annoying at first, but a brief description is helpful for tracking previous changes in the Git History log.
- Commit the files via `git commit -a -m "<message>"` e.g. `git commit -a -m "added column to newSpreadsheet"`
- `-a` means "commit all"
- `-m` means "I'm also including a message"

Git Pull

- It's a good practice to *always* pull from the Remote before you push, else you will quickly run into commit issues
- "Pull" merges changes and files from the "remote master" with your local branch via `git pull`

Git Push

- Push, or merge, YOUR changes to the remote via `git push`
- Your modifications and new files will now be shared with everyone else once *they* pull