Git(hub) Cheatsheet

Words to the wise

- **These are `git` commands to be run in TERMINAL**: Make sure you are not running them from the R CONSOLE - they are NOT THE SAME!
- If you are working within RStudio Cloud, you will get this warning message when pushing: `error: cannot run rpostback-askpass: No such file or directory`. **Ignore it and do what you're doing!**
- Be careful with quotes. Programming needs plain quotes, *not curly quotes*. If you ever copy/paste via a Word Document or Google Doc, there is a good chance your quotes became curly. Your commands will NOT WORK in that case!
- When we type passwords in UNIX, no symbols appear - the password is STILL BEING TYPED, I promise.
- All `git` commands can be run from anywhere INSIDE the git repository - doesn't matter which specific directory (doesn't need to be "top level") as long as you are in one of them.
- When you make typos (like `git pusj` misspelling `git push`), `git` will try to suggest to you which command you actually want! It's very helpful.

Vocabulary

- A **repository** is a directory that `git` is managing
- Files that `git` knows about are **tracked**, i.e they are **under version control**
- File changed that will become part of the next commit are **staged**
- The **origin** refers to the remote (i.e., not on your computer - in this case it's `github.com`) repository where your local repository was originally cloned from.
- The word **master** refers to the primary repository branch. *Don't worry about it in this class*
  - When you see `origin/master` displayed, interpret this to mean: the repository as it looks on `github.com`

Initial setup

To avoid being constantly prompted, you need to do this ONE TIME ONLY on your computer (or, in a given RStudio Cloud project) in **terminal**.

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

For example, I would type:

```
git config --global user.email "spielman@rowan.edu"
git config --global user.name "Stephanie Spielman"
```

The *very basic*, non-branch-using `git` command reference

Command	Description
`git clone <URL>`	Clone the repository to a new computer. **This should only be run ONCE**.
`git add <filename>`	Stage a file with changes
`git rm <filename>`	Remove a file from being tracked by git entirely
`git mv <filename>`	Move a file and track this move with `git` in a shortcut version of `mv <filename> <newfilename>; git add <newfilename>` (Note: there is no analogous shortcut for `cp`)
`git commit -m "message"`	Commit files that have been added/removed with a message! **Don't forget to type the `-m "message"` - you will be stuck in VIM if you forget, and that will not be pleasant.**
`git push`	Send commit(s) to `github.com`
`git pull`	Obtain commits pushed to `github.com` onto the current machine. **Run this EVERY TIME you return to the project**
`git status`	Check the status of files
`git log`	See past commits and associated messages (press the `q` key to exit)
`git checkout <filename>`	Obtain the current `github.com` version of the file - useful when you screwed something up and want to start from scratch for this commit
`git diff <filename under VC>`	Compare current UNSTAGED file version to the tracked version of the file

The individual `git` workflow

> **At any and all times, running `git status` will show you what is staged, what is tracked but not staged, and what is not tracked at all.**

1. **Obtain** a local version (aka, on your computer) of a repository with `git clone <repository URL>`
2. **Add files for staging** with `git add <filename>`
   - Sometimes you will others use `git add .` to add all contents of current directory, or similar `git add *`. **This has the potential for MAJOR DANGER, so you should always specify files INDIVIDUALLY to add.** You may therefore type `git add` several times for each file to add - yes!!
   - Any time you make a change to a file, it must be re-added
   - You can also remove files from being tracked entirely by git with `git rm <filename>`
3. **Stage changes** (add/rm) with `git commit -m "An informative message that broadly says what the commit does"`
4. **Push** changes to the `github.com` repository versions with `git push`
   - Depending on your local setup, you may be prompted for your `github.com` username and password every time.
5. **Update** your local repository with the `github.com` version of the repository with `git pull`
   - *This is ALWAYS STEP 1 when returning to a project! YOU DON'T WANT CONFLICTS*

Miscellaneous Troubleshooting

Some messages you might see, what they mean, and how to deal with them:

When trying to remove a file with `git rm` (or directory with `git rm -r`):

```
error: the following file has local modifications:
    <name of file>
(use --cached to keep the file, or -f to force removal)
```

This means you want to remove the file, but because you changed the file from its last staged version, `git` is scared to do this! You have two choices:

1. Tell `git` no I really DO want to remove this with : `git rm -f <name of file>`
2. Revert the file and then remove it:

   ```
   git checkout <name of file>
   git rm <name of file>
   ```

When trying to add or remove a file:

```
fatal: pathspec <name of file> did not match any files
```

This means you are trying to add or remove a file that doesn't exist! Potentially, the file isn't there at all - (did you spell it wrong? are you in right directory? However, if the file *does* exist and you can't `git rm` it, that's because the file isn't actually under version control! `git` doesn't already know about the file so it doesn't want to remove it. Instead, you need to just remove the file regularly with plain `rm` instead of `git rm`.

When checking `status` or making a `commit`

```
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

This means what's on your computer exactly matches what is on github - there are no changes made and nothing to commit!

When trying to `commit`

```
error: switch `m' requires a value
```

You forgot the commit *message*! You can't just type `git commit -m` - you need `git commit -m "YOUR MESSAGE GOES HERE"`.