Using Git and GitHub – Maddie's top commands

MMH

November 7, 2016

Introduction

git = local
GitHub = git with friends
Git repositories should be thought of as "projects"

This compilation of git commands isn't comprehensive, but it covers the commands that I use on a regular basis. I should preface all of this by mentioning that *I use the command line*, but you don't have to. If you prefer GUI work, you can download the GitHub GUI at https://desktop.github.com/. It's useful for both git and GitHub. And I know next to nothing about how to use it.

Some other AMAZING resources for the MOD group:
https://github.com/modscripps
https://github.com/OceanMixingGroup
Guys, we can't let Jonathan Nash be way better at version control than we are...

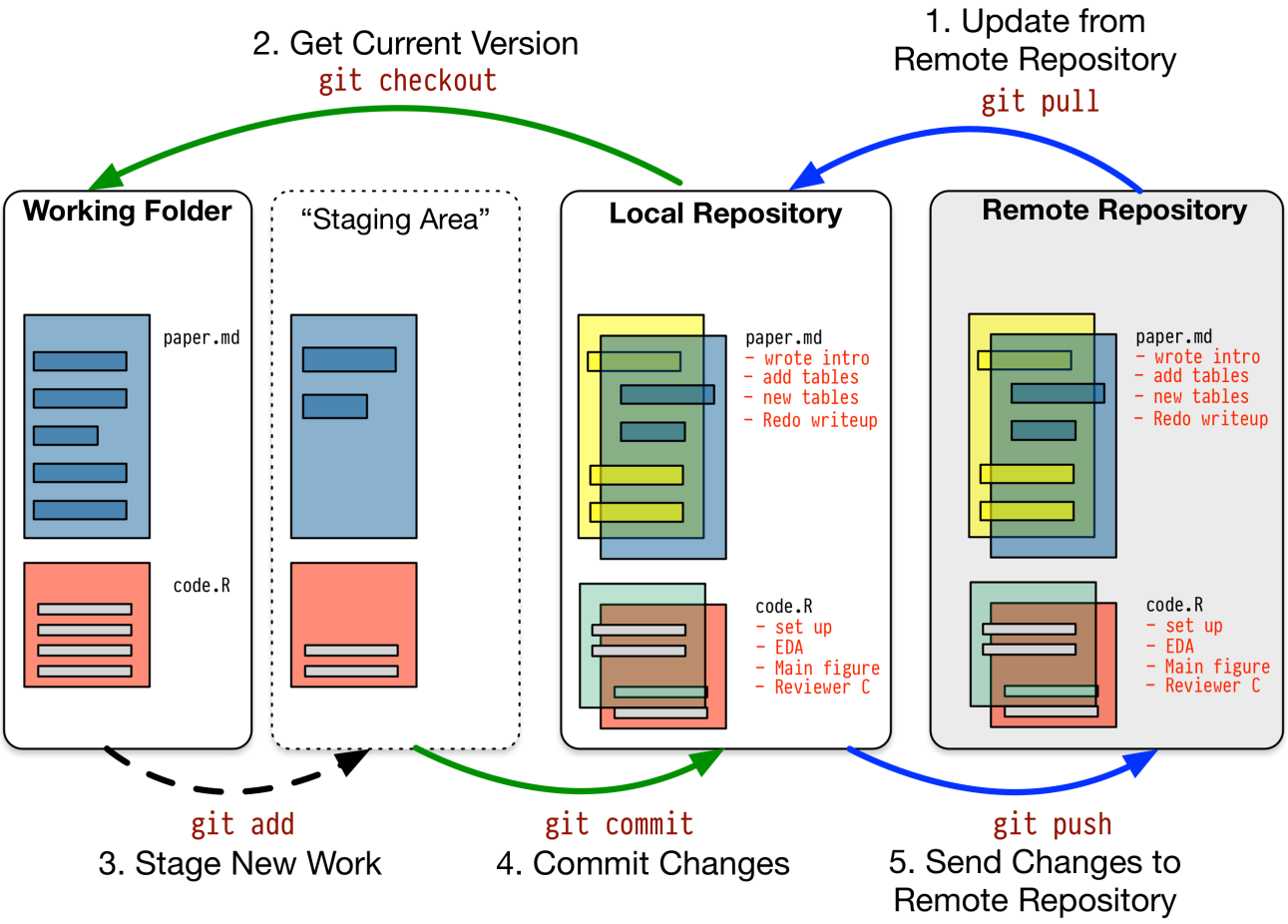

Figure 1: Okay, so this is a nice figure to contextualize what's going on with git.

Local git commands

I. Making Changes

A. **git init**

i. creates a git repository
mkdir yourdirectoryname
cd yourdirectoryname
vi README.md
git init

a. ".md" stands for markdown

b. This is for best practices purposes

B. **git add**

i. adds files within your working directory into the staging area
git add -A

a. adds ALL files

ii. if the repository contains a file called .gitignore, files with designated file extensions or within specified subdirectories will not be added
vi .gitignore
git add .gitignore

a. Examples of .gitignore files can be found at https://github.com/github/gitignore

b. It has to be added to the repo in order to work!!

C. **git commit**

i. takes what is in staging directory and puts into git working directory
git commit -m "YOUR MESSAGE HERE"

a. commits changes in staging area

b. '-m' adds a message so that you can comment on your changes

II. Visualizing Progress

A. **git status**

i. shows the status of your woking directory and staging area relative to the current git repo

B. **git diff**

i. sees all content and understands in terms of additions and removals

ii. more complicated than git status; gives more detail on the file changes specified in git status

III. Making BIG changes – Using Branches
for more info see: https://www.atlassian.com/git/tutorials/using-branches

A. **git branch**

i. creates a new line of development

ii. Use this when you are about to make major changes, or changes that might break things
git branch <branch-name>
   a. git branches must have one word names
   b. main branch is always called "master"

B. **git checkout**

   i. use to begin operating in a project branch
   git checkout <branch-name>
      a. must move into the branch before committing devilish changes!

C. **git merge**

   i. merges changes from a branch into the directory you are currently working in
   ii. to merge your <branch-name> changes back into the main branch:
   git checkout master
   git merge <branch-name>

IV. Fixing things
for more info see: https://www.atlassian.com/git/tutorials/undoing-changes

A. **git log**

   i. Displays the commit history
   ii. lists codes or "commits"

B. **git checkout**

   i. use to check out an old version of the file
   git checkout <head> <filename>

C. **git reset**

   i. Undo local changes
   git reset
      a. Resets the staging area to match the most recent commit, but leaves the working directory unchanged
      git reset –hard
      b. Resets staging area AND working directory

GitHub commands

For more info see: https://www.atlassian.com/git/tutorials/syncing

I. Set up remote

A. **git remote**

i. list and add connections to remote repositories (like GitHub repos!)
git remote add <name> <remote-url>

a. assign a remote repository to a git workspace on your local machine

b. <name> is most often "origin"

c. now, any <remote-url> that follows can be replaced with the chosen <name>

B. **git clone**

i. make a copy of someone else's repository to your own computer
git clone <remote-url>

C. **git fork**

i. different from cloning–this continuously tracks with the master;

ii. use this if you plan to make a contribution back to the master with a useful change

II. Interact with remote

A. **git push**

i. pushes changes up to a remote repository
git push <remote-url> master

B. **git pull**

i. pull changes from a remote repository into the current repo
git pull <remote-url>

ii. If you are the owner of a shared repository on github, you will use this command to pull changes made by other users into the master branch

III. Fix things

A. **git revert**

i. Undoes, but still tracks!, an entire commit

ii. Because it still tracks, it is safer to use in a shared repository than *git reset*
git revert
OR *git revert <commit>*

B. **git rebase**

i. use when a development branch gets way ahead of the master to reflect the development changes instead of the master history with user changes overlaid