Teaching basic lab skills
for research computing

Git lesson using worksheets

I attended my first Software Carpentry workshop in 2015 as a helper and was mesmerized by how Ivan Gonzalez taught the Git lesson using an easel pad and different colored markers. He had separate boxes for the working directory, stage and history and went back and forth between the terminal and drawing on the pad to recap his last set of commands. Ever since, Git has been one of my favorite lessons to teach, because the material has depth and is challenging to explain, but it can be taught well with the help of Ivan's drawings and the well maintained lesson materials.

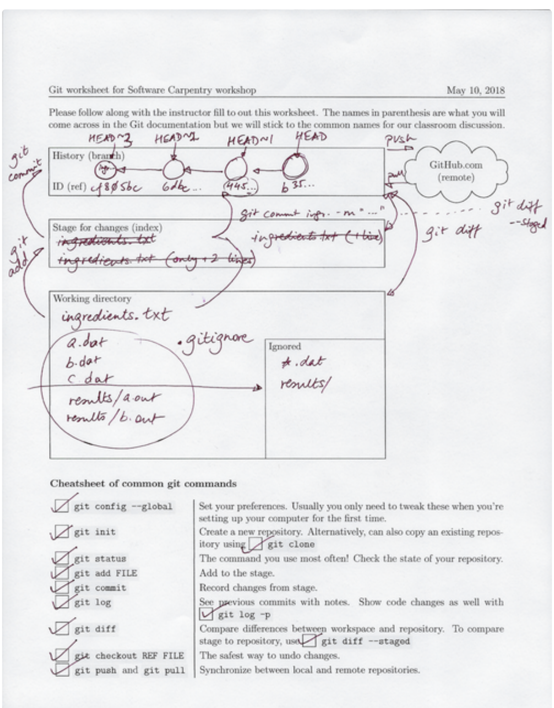
The Git drawings seemed to work well until last year when I taught in a room with afternoon sunlight and a difficult to see blackboard. I was scheduled to teach in the same room at a workshop earlier this month, and to remedy the hard-to-see blackboard, I used a document camera and drew on a worksheet instead.

Herein was a nice opportunity: why not have the learners also draw long? During my PhD studies, I've had two classes, biochemistry and phylogenetics, where the instructor had us draw along and I found I enjoyed drawing things in the classroom. How often does one get to draw as an adult? I was instantly transported back to kindergarden.

In addition to the Git drawing section at the top of the worksheet, I added a cheatsheet of the Git commands we would be covering that afternoon with little checkboxes next to the commands. That way, learners can get the visceral feeling of checking off a command when it is covered and understood (or have an opportunity to protest if it wasn't understood).

Drawing along can have educational value in addition to feeling good. In Chapters 4 and 5 of **"How to Teach Programming (and Other Things)" (http://third-bit.com/teaching/)** Greg Wilson discusses combining words with visuals (dual coding) and of the need to slowly present a diagram in pieces to later help trigger recall of what was said by pointing at the diagram.

Here is the cheetsheet **clean (https://github.com/omsai/git-swc-worksheet/releases/tag/v2018.05.10)** and filled-in:


Git worksheet for Software Carpentry workshop — May 10, 2018

Please follow along with the instructor fill to out this worksheet. The names in parenthesis are what you will come across in the Git documentation but we will stick to the common names for our classroom discussion.

Cheatsheet of common git commands

☑ `git config --global`	Set your preferences. Usually you only need to tweak these when you're setting up your computer for the first time.
☑ `git init`	Create a new repository. Alternatively, can also copy an existing repository using ☑ `git clone`
☑ `git status`	The command you use most often! Check the state of your repository.
☑ `git add FILE`	Add to the stage.
☑ `git commit`	Record changes from stage.
☑ `git log`	See previous commits with notes. Show code changes as well with ☑ `git log -p`
☑ `git diff`	Compare differences between workspace and repository. To compare stage to repository, use ☑ `git diff --staged`
☑ `git checkout REF FILE`	The safest way to undo changes.
☑ `git push` and `git pull`	Synchronize between local and remote repositories.

The reason to use arrows of the history items pointing back to their immediate previous items is to make the "detached HEAD" error state more clear. When we create the detached HEAD state in the classroom, we don't see recent commits because the history items are only aware of their previous commits; or at least that's a good enough mental model of Git's true behavior.

Learner **sticky note feedback (https://github.com/tem11010/2018-05-10-UConn/wiki/Stickies-feedback#day-1-afternoon-git)** from that section of the workshop is slightly clouded by some confusion during the collaboration section in the late afternoon, and because Git was taught without teaching the Shell first. I will use the worksheets again at a more traditional workshop in 2 months and hope you will also try using Git worksheets!

Pariksheet Nanda 2018-05-26 WORKSHOPS · TEACHING · GIT