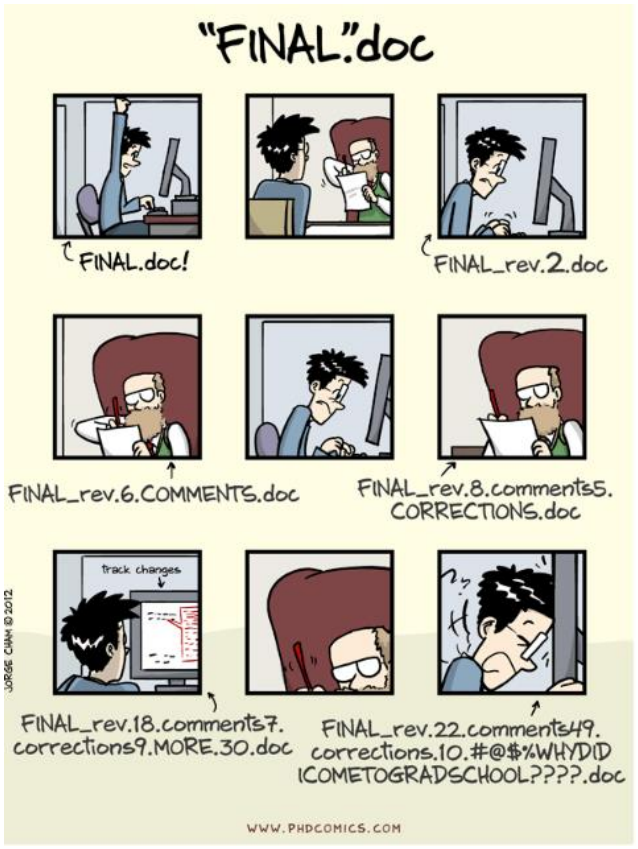
Introduction to Git and GitHub

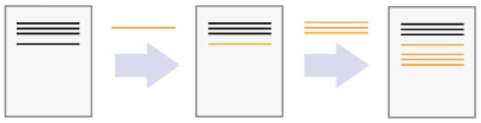
Things to do before class

1) Sign up for a GitHub account at: https://github.com/.
2) Download and install GitHub desktop for at https://desktop.github.com/ (git works on all platforms--Windows, Mac, Linux) to host code and interact with an online GitHub account.
3) Add git bash through a command line utility like Powershell. When installing, if given the option, add the git command line utilities to your environment variables. This will make sure that you can use git from the command line.
4) Make sure that the git commands are accessible from the git shell. Open the git shell, and type "git –version." This should provide some info on your version of Git Desktop. If you get a message like "git is not recognized as an internal command," it means that you need to add it to your environment variables. For windows 10, instructions are here: https://stackoverflow.com/questions/26620312/git-installing-git-in-path-with-github-client-for-windows.
5) We will be using the Software Carpentry Lesson. I will skip lots of it, but may want to follow along and follow up here: https://swcarpentry.github.io/git-novice/.
6) At the end, we will go through an exercise to help understand the process of software collaboration using this repository: https://github.com/djlampert/git_intro_exercise.
7) It's good to have a text editor. Notepad++ (https://notepad-plus-plus.org/downloads/) is intuitive for Windows, gedit is available on most platforms.

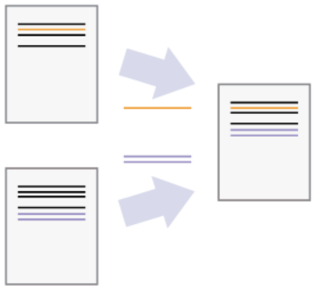
Reasons to use GitHub


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDID ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

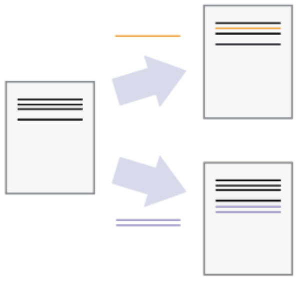
Keeping track of changes



Merging changes



Resolving conflicts



Key git commands
Use "git" followed by the git commands, vs other commands that are specific to the terminal/shell. Git is used to keep track of the history of changes in a repository, which is a folder filled with files and other content either on a local computer or online.

status	--	show what has been done since the last save point/commit
add	--	stage a file or group of files/directories to the queue to be recorded to history
commit	--	record changes to history, creating a "save point."
push	--	upload changes in the history to online GitHub repository
pull	--	pull changes from the online repository to the local
log	--	show summaries of changes throughout the history of the repository
diff	--	compare different points in history
clone	--	make a copy of an online GitHub repository locally
remote	--	change settings for online (remote) repos push/pull
fork	--	create a new "branch" to test out changes in parallel (for collaborating)

Working with the Git Shell
Open the git shell.

Open a windows or other file explorer program and navigate to the GitHub desktop home directory.

Describe how to view and move in and out of directories using ls/dir and cd.

Show how to open a text file from the git shell using notepad editor.

Use "git config --list" to view important settings.

Use git config to change the user name. Make sure students username and email are consistent with their online GitHub account they created before the session.

Creating a Git Repository
Mkdir planets.

Cd planets.

Use "git init" to start tracking history.

View hidden files/folders using "ls –a" or "dir –force."

Note "git init" creates a hidden directory that keeps track of changes to the repository.

Use "git status" to view changes to the repository.

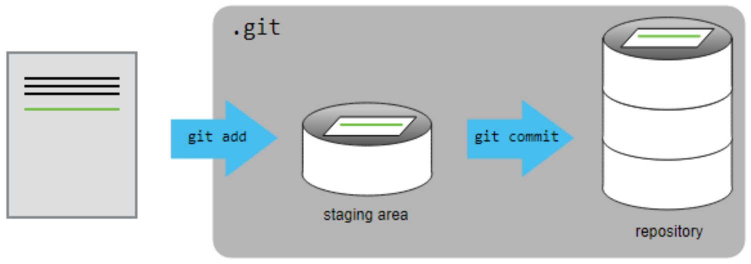
Create a file in the "planets" directory called "mars.txt" and the phrase "Cold and dry, but everything is my favorite color" into the file.

Use dir/ls and “git status” to see the file.

Use “git add mars.txt” to stage the file to be added to history.

Use “git status” to see that the file is ready to be committed to history.

Use “git commit” to record changes to history (create a “save point”). Use “-m” to add a message describing the changes.



Editing a repository
Check git status.

Use “git log” to see a summary of commitments. If the log is longer than a page, you can scroll through and end with “Q.”

Edit the “mars.txt” file to add “The two moons may be a problem.”

Check git status

Use “git diff” to see specifics on changes since the last commit (save point).

Use “git add --all” to add all changes (rather than one file at a time).

Commit changes with message “added concerns about moons.”

Check the log

Edit the “mars.txt” file to add “the low humidity is nice.”
Create a new file “venus.txt” and add the text “it’s hot and humid here”

Add just the mars changes, then show status.

Use “git reset” to unstage changes.

Show how multiple changes can be added and committed at once, then view the log again.

Use “git diff’ followed by “HEAD~1” and “HEAD~2” (the number counts backwards in the list of commits) or by using the actual log number of the commits (which can be grabbed using the mouse with right click) to see changes across a period of time.

Use “git checkout” followed by HEAD or the commit number to recover old versions of a file.

Working with Remote Repositories on GitHub

Git is a programming language to keep track of versions of files. GitHub is a website that hosts repositories of code.

Go to your GitHub home page, and click on “Repositories,” then click “Create Repository.”

Call the repository “planets.” The other settings should be fine. These steps are equivalent to making a new directory and initialize git, just on the web instead of on your computer.

Cope the repo http address by clicking the button on the right, then click “git remote add origin <http url>.” This will “connect” the local and remote (internet) versions of the “planets” code.

Type “git remote –v” to show the targets associated with git pulls and fetches.

Type “git push origin master” to “push” the changes from your local planets repo to the GitHub repo. “Master” is the main branch of the local repo on your computer, “origin” is the remote repo on GitHub. There can be issues with passwords at this point. If two-factor authentication is used, you have to add a “token” rather than the password (this recently happened to me).

Navigate through commits on GitHub to see how the interface works.

Create a readme file on the GitHub repo, and put in a short description of the planets software. Commit the changes on the remote repo.

Type “git pull origin master” in the command line, which will “pull” changes from the online repo to the local repo. You should be able to see the readme. View the log also.

Cloning a GitHub Repository from someone else

Get the students to navigate to one of your repos (e.g., my Streeter-Phelps Python model at https://github.com/djlampert/StreeterPhelps). Then have the students click “fork” to create their own version. This should create “<username>/SteeterPhelps” for each of their repos.

Navigate out of the planets repo, and then “clone” the repo locally by typing the following: “git clone https://github.com/djlampert/StreeterPhelps.git.” This should make a local copy that includes an html file that opens in a browser.

Have each student create a new file “color.txt” that contains the name of their favorite color.

Have each student push the changes to their version of the Repo. There is no need to set up origin master, etc., since this is done automatically when “cloning” a repo.

Collaborating with GitHub
Have the class navigate to the "git_intro_exercise" repo on your GitHub page, fork the repo to their GitHub profile, then clone it locally. The "forks" should show up on the host's account.

Have each person in the class pick a country and edit the file to provide info (make sure that no two students do the same country).

Have the students push their changes to their online intro exercise repo. Have them run "git remote –v" to verify the origin.

Have the students create a "pull request" on their GitHub repo, which will alert the instructor of updates that they should "pull" from each user.

After all the pull requests have been created, the instructor should "pull" all the changes on their GitHub repo.

The instructor should then create a pull request for the class. Students can accept the changes to the online repo, then they can pull the online changes locally. Another option is to have them import to their PC directly using "git remote add upstream https://github.com/djlampert/git_intro_exercise.git" to add the original repo as a source, then "git pull upstream master" to pull the other students changes locally.

Final thoughts
There are lots of other features to deal with managing conflicting changes, going backwards in time to trace issues, and interacting with security and log-in. These can be dealt with as they come up. This exercise should cover the essentials. GitHub has lots of primers and help tools to understand how to manage complex software updates.