MIS 407

w01c03

Prof. Smith

Recap

- Last class we reviewed terminal/command line interfaces, and the 8+1 basic commands (I added **cd** to the list)
- You've submitted your first ICA (marks have been posted).
- You should have read section 1.1 through 1.3 of the ProGit online book
  - https://git-scm.com/book/en/v2

For Today…

Date	Class	Module	Topic
9-Jan-17	w01c01	**Programming Environment**	Course Intro
11-Jan-17	w01c02		Cmd Line and Text Editor (Atom)
13-Jan-17	w01c03		Git & Github
16-Jan-17	w02c04	MLK Day	
18-Jan-17	w02c05		Atom Usage, and Addons
20-Jan-17	w02c06		**Test 1**

Goals:

- Introduce Git and and GitHub
- Conduct our second in-class assignment ICA02

Intro to Git

- Git is a VCS (version control system).
- Most large, multi-member, group projects will use some form of VCS
- Git is one of the most popular.

History of Git

- Developed by Linus Torvalds (Linux Fame)
  - He wasn't happy with existing VCS's
  - They had a number of problems, and were generally slow and cumbersome.
  - He hated existing VCS so much that he didn't even use them in his Linux Kernel project
    - He used tar-balls and patches – a rather "manual" method.

Why Git

- Our projects really won't be that large, so why use Git?
  - Group work requires complex coordination of updates and code, Git helps make this "safer"
  - The power of "versioning" and "branching"
  - MOST importantly, it's a key means through which to participate in a programming community.

See 7:13 onward of … https://www.youtube.com/watch?v=Vo9KPk-gqKk

Intro Concepts

- A “repo” is a repository of your code who’s state is managed by Git.
- Files are tracked, untracked, or ignored.
- There are multiple Git clients out there, we will use the official release from

  www.git-scm.com.

NOTE: We will only be using the command land version. Knowing the command line version will allow you to more fully understand and control Git.

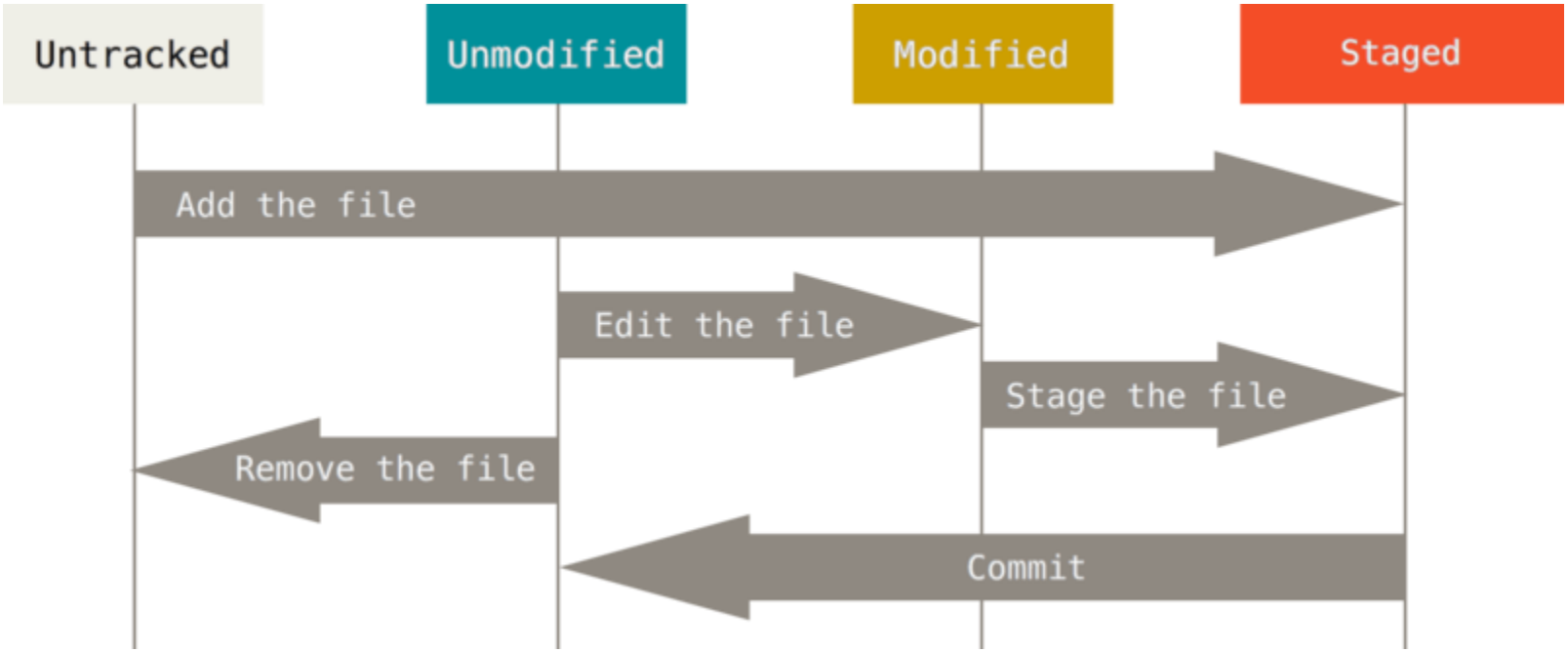
General Process

- Initialize a repo (later, you'll usually clone one from GitHub)
- Add and commit files to the repos.
- Modify files in the repos, stage them, and them commit them back into the repo.

The four "states" of a change to files in your repo

- 1. Untracked
  - A file that has never been committed to the repo and is not staged.
- 2. Unmodified
  - A file that has been previously committed to the repos (thus, it is tracked) but no changes to the file have been made since the last commit.
- 3. Modified
  - A file that is being tracked, and changes have been made to the file since the last commit.
- 4. Staged
  - A tracked file that has bee readied to be committed to the repos. You add any modified files to a staging area. The staging area is "built up" and organized until you are ready to commit the changes to the repo.
- Committed
  - This is a state of a file, but is worth nothing here.
  - Once you "commit" the changes that are staged, the file moves to the Unmodified state.

The four possible “states” of a of a file in your repo



Let's try this…

- Got to our GitHub site MIS407-17s
  - https://github.com/MIS407-17s
- Work through the following:
  - 1.git_basic_local_workflow.md

- By next class…
  - First: Review sections 2.1 through 2.5 of ProGit, Chapter 2 “Git Basics”
    - https://git-scm.com/book/en/v2

- During Next class…
  - Continue with working with/learning Git
  - Review how to submit an assignment using Git/GitHub.
  - Quick intro to Atom
  - Review for first test (challenge).
    - Hint: It will be a Git/GitHub challenge where you need to accomplish a task using techniques covered and submit a copy of your Bash session (as a text file).