Modern Version Control with Git

Aaron Perley (aperley@andrew.cmu.edu)
Ilan Biala (ibiala@andrew.cmu.edu)
https://www.andrew.cmu.edu/course/98-174/
Why should you take this course?

“Version control software is an essential part of the every-day of the modern software team's professional practices.”
-- Atlassian Git Tutorial

9. Know Git well.
I realize that Git is more prevalent in some development communities over others, but Git is more than just a VCS (version control system). Because of its efficiencies in branching, it enables a very effective new flow that can be leveraged by both individuals and teams.
Why should you take this course?

From a 2013 Fox News report:
Git ≠ Github
What this course isn’t

• For seasoned Linuxbeards
What this course isn’t

• A crashcourse on git commands
What this course is about

• Forming a **mental model** for how git interacts with versions of files
• Understanding how to use git (and a bit of Github) in a collaborative setting
Course Website

https://www.andrew.cmu.edu/course/98-174/
Syllabus

• No prerequisites

• 3 Free Elective credits

• No official textbook, but we recommend Pro Git by Scott Chacon (free, online)

• No office hours unless specifically requested
  • Email Aaron and Ilan if you have questions

• Slides and lecture notes posted online
More Syllabus

• 2 unexcused absences allowed
  • 3+ unexcused absences and we have to give you a No Pass
  • Email us if you’re going to miss for a legitimate reason
  • More than 15 minutes late = unexcused

• Discussion of in-class and homework assignments is encouraged, but write up your own answers

• No late work
  • Email us if you think you need an extension
Grade Breakdown

Pass/No Credit. Need to get 70% out of:

• 20% Weekly Lecture Attendance
• 30% Submitted work (often in-class)
• 20% Midterm (Date TBA)
• 30% Final (Date TBA)
Waitlist

- If you are on the waitlist, please keep coming to class.
- There is a 99.99999% chance you will be able to get in off the waitlist
What is Version Control?
Goals of Version Control

• Be able to search through revision history and retrieve previous versions of any file in a project
• Be able to share changes with collaborators on a project
• Be able to confidently make large changes to existing files
Named Folders
Approach

• Easy
• Familiar
• ...

• Can be hard to track
• Memory-intensive
• Can be slow
• Hard to share
• No record of authorship
Centralized Version Control Systems

• A central repository determines the order of versions of the project

• Collaborators “push” changes to the files to the repository

• Any new changes must be compatible with the most recent version of the repository. If it isn’t, somebody must “merge” it in.

• Examples: SVN, CVS, Perforce
Distributed Version Control Systems (DVCS)

- No central repository, each developer has their own copy
- Developers work on their own copy of the repository locally and sync changes with others
- Examples: Git, Mercurial
Git

• Created in 2005 by Linus Torvalds to maintain the Linux kernel. Oh, and he created that too.
• Distributed VCS

https://www.git-scm.com/
Installing Git

https://www.andrew.cmu.edu/course/98-174/lecturenotes/installing_git.html
Git Init

Initializes a new git repository in an existing folder

• The folder is now called a git repository
• Changes to any files in the folder (and its subfolders) can be tracked by git
• Git stores its metadata in a hidden .git folder in the repository root

$ mkdir myrepo
$ cd myrepo
$ git init
Git Clone

• Download an existing repository (and all of its history!)

$ git clone https://github.com/autolab/Autolab.git
$ cd Autolab
Git Log

List the history of a repository

$ git log

Press ‘q’ to exit, use arrow keys (or j,k) to scroll
What is fad72e4?

• Commits are uniquely represented by SHA-1 hashes
• The first 6-7 characters of a hash are usually enough to identify it uniquely from all the other commits in the repository
• This is called the short hash
Okay, so what is a commit?

1. A **snapshot** of all the files in a project at a particular time.
2. A **checkpoint** in your project you can come back to or refer to.

Anything else?

3. The **changes** a commit makes over the previous commit
Homework

• One hour or less a week
• Released on Thursdays after class, due next Thursday at beginning of class
• Posted on the course website
• Email us if you have questions, I’ll be happy to help you out!

Submit on Autolab:
https://autolab.andrew.cmu.edu/courses/98174-s18/