98-174 S17
Modern Version Control with Git

Ilan Biala (ibiala@andrew.cmu.edu)
Aaron Perley (aperley@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 $\neq$ 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
### Last Semester's Schedule

[https://www.andrew.cmu.edu/course/98-174/f16/](https://www.andrew.cmu.edu/course/98-174/f16/)

<table>
<thead>
<tr>
<th>Date</th>
<th>Lecture</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/29</td>
<td>Git Basics, Version Control, and Git</td>
<td>Slides</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9/5</td>
<td>No Class (Labor Day)</td>
<td>Slides</td>
</tr>
<tr>
<td>9/12</td>
<td>Simple Git Commands</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9/19</td>
<td>Optional: Git Config and .gitignore</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td>9/26</td>
<td>More on Git Commit</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10/3</td>
<td>Branching and Merging</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td>10/10</td>
<td>Linux Tools on Git</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td>10/17</td>
<td>Miscellaneous Git Commands</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td>10/24</td>
<td>No Class</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td>10/31</td>
<td>Remotes</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td>11/7</td>
<td>No Class</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td>11/14</td>
<td>Miscellaneous Git Commands</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td>11/21</td>
<td>Rebase</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td>11/28</td>
<td>Github</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
<tr>
<td>12/5</td>
<td>Final</td>
<td>Slides, Git Book on Git Basics</td>
</tr>
</tbody>
</table>

Good luck on your final exams!
Course Website

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

Pass/No Credit, like every StuCo. To pass, get 70% out of:
• 20% Weekly Lecture Attendance
  (Tuesdays 6:30PM – 7:20PM, Baker Hall 140F)
• 30% Submitted work (often in-class)
• 20% Midterm (Date TBA)
• 30% Final (Date TBA)
More Course Details

• Prerequisite: Basic Unix Survival
• 3 Free Elective credits
• No official textbook, but we recommend Pro Git by Scott Chacon (free, online)
• No office hours unless specifically requested
  • Email Ilan and Aaron if you have questions
• Slides and lecture notes posted online
Course Policy

• By StuCo Policy, students with more than 2 unexcused absences must be given a No Pass in the course. Thus, email us if you’re going to miss class for a legitimate reason, and you might get an excused absence.

• More than 15 minutes late = unexcused absence. Don’t be late.

• Academic integrity applies. Don’t cheat.

• No late homework.
Waitlist

• If you are on the waitlist, please keep coming to class.
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

https://www.atlassian.com/git/tutorials/what-is-version-control
Named Folders

Approach

- Easy
- Familiar
- ...

- Can be hard to track
- Memory-intensive
- Can be slow
- Hard to share
- No record of authorship
Local Database of Versions Approach

- Provides an abstraction over finding the right versions of files and replacing them in the project
- Can’t share with collaborators
Centralized Version Control Systems

• A central, blessed repository determines the **order** of commits ("versions" of the project)
• Collaborators “push” changes (commits) to this repository.
• Any new commits must be compatible with the most recent commit. If it isn’t, somebody must “merge” it in.

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

- No central repository
- Every repository has every commit
- 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