

Introduction to Version Control Systems

Hui Chen ^a

^aCUNY Brooklyn College, Brooklyn, NY, USA

February 9, 2023

Outline

- 1 Need for Version Control Systems
- 2 Git Basics
- 3 Merge Conflicts
- 4 Cool Features
- 5 VCS Hosting
- 6 In-Class Exercise on VCS
- 7 Summary and Questions

Outline

- 1 Need for Version Control Systems
- 2 Git Basics
- 3 Merge Conflicts
- 4 Cool Features
- 5 VCS Hosting
- 6 In-Class Exercise on VCS
- 7 Summary and Questions

Readings and References for the Lecture

- ▶ Free git book: <https://git-scm.com/book/en/v2>
- ▶ Chapter 11.1 and 11.5 of the textbook
 - ▶ 11.1 Software Configuration Management
 - ▶ 11.5 Tools for Configuration Management

SE is a team effort

- ▶ Increase of functionality/quality/complexity → cannot achieve software break through alone
- ▶ Successful software development career → programming skill *and* team skills (i.e., plays well with others and can help make team win)

“There are no winners on a losing team, and no losers on a winning team.” – Frederick Brooks Jr.

Examples: Increase of Team Size

- ▶ How many programmers developed Space Invaders for Arcade Console in 1981?
- ▶ How many programmers developers Super Mario Bros. NES (Nintendo) in 1985
- ▶ How about Resident Evil 6 for PC, PS3, Xbox 360 600 2013
- ▶ How about ...

Version Control Systems

A necessary tool of the trade

- ▶ Main idea – Foster team collaboration by providing a “centralized” location to store project files
- ▶ What are they?
 - ▶ Version (snapshot) code, docs, config files and so on at key points in time
 - ▶ Complete copy of every versioned file per snapshot

There are several related terms, and some of them are often considered synonyms:

- ▶ Source Code Management System (SCMS)
- ▶ Version Control System (VCS)

Why

Why do it?

- ▶ Roll back if introduce bugs
- ▶ Separate deployed from development version of code
- ▶ Keep separate branches of development
- ▶ Documented history of who did what and when
- ▶ Track what changed between revisions of a project
- ▶ ...

Evolution of 50 years

- ▶ SCCS & RCS (1970s)
- ▶ RCS (1982)
- ▶ CVS (1986)
- ▶ Subversion (2001)
- ▶ Git and Mercurial (2005)

Centralized vs. Distributed

- ▶ Older VCS (e.g. CVS and SVN) imposed a centralized system, i.e. one computer system (server) holds the repo, others are copies
- ▶ Newer VCS (e.g. git and mercurial) do not require a central server - they are decentralized or distributed (although having a central server is often convenient)

We will focus on git, a distributed VCS – perhaps, the most popular VCS today

Outline

- 1 Need for Version Control Systems
- 2 Git Basics**
- 3 Merge Conflicts
- 4 Cool Features
- 5 VCS Hosting
- 6 In-Class Exercise on VCS
- 7 Summary and Questions

Where to get git?

Check out <https://git-scm.com/downloads>

- ▶ Linux (Ubuntu or Debian): `sudo apt-get install git`
- ▶ OS X: `brew install git` in the nutshell
- ▶ Windows: download and run the installation package

Although lots of GUI tools are available, *the command line is the best way to use VCSs.*

Learn to use command line!

Basic git operations

1. `git clone`
2. `git add`
3. `git commit`
4. `git push`
5. `git pull`
6. `git rm`
7. `git status`

Example Git Workflow

A good read is at

<https://blog.o Steele.com/2008/05/my-git-workflow/>

Questions

On VCS basics?

Outline

- 1 Need for Version Control Systems
- 2 Git Basics
- 3 Merge Conflicts**
- 4 Cool Features
- 5 VCS Hosting
- 6 In-Class Exercise on VCS
- 7 Summary and Questions

A little history about versioning models

- ▶ Older VCS like CVS use the lock-modify-unlock model
- ▶ Newer VCS like Subversion and Git uses the copy-modify-merge model
 - ▶ Two developers can modify the same file “simultaneously”, which may lead to a *merge conflict*.

Merge Conflicts

Merge Conflicts

- ▶ When two developers edit the same file the developer that tries to commit the file last will have to combine his changes with that of the prior developer → merge conflicts
- ▶ Automatically combines conflicted files when the changes aren't overlapping
- ▶ True conflicts must be resolved by hand

Branches

- ▶ Branches create a separate thread of commits
- ▶ Named, so can switch between them and “master” or “main”
- ▶ Allows someone to implement a new feature separate from the main development trunk
- ▶ Merge to main branch at a later point, (but may not be possible)
 - ▶ See also git branch and git merge

Merging two branches can be difficult, in generally, not recommended unless you have a clear and strong argument to support it.

.gitignore

The .gitignore file specifies which files or file extensions to ignore

- ▶ You probably want to ignore generated (e.g. .class) file, or there will be unnecessary conflicts
- ▶ Use in project root directory for the entire project

Outline

- 1 Need for Version Control Systems
- 2 Git Basics
- 3 Merge Conflicts
- 4 Cool Features**
- 5 VCS Hosting
- 6 In-Class Exercise on VCS
- 7 Summary and Questions

Cool Features. Pull Request

- ▶ A way to manage contributions to software projects where contributors have no privilege to write to the repository.
- ▶ Do you want it in your organization and development process?
- ▶ Commonly used for open-source software projects.
- ▶ Main idea.
 - ▶ The project cannot trust you to make incremental commits to the repo, so instead:
 - ▶ Branch the main repo
 - ▶ Implement a feature
 - ▶ Send a pull request to project admin
 - ▶ Reviews code associated with pull request and if
 - ▶ acceptable then merges it to the central repository

Cool Features. `git bisect`

- ▶ Searches for a revision where a bug was first introduced by checking out a version of the code and asking whether the bug is there
- ▶ Can be scripted for complete automation
- ▶ Uses binary search, instead of sequential increasing efficiency

Cool idea. `git blame`

A versatile troubleshooting utility

- ▶ displays author metadata attached to specific committed lines in a file.
- ▶ is used to examine specific points of a file's history and get context as to who the last author was that modified the line.
- ▶ is also used to explore the history of specific code and answer questions about what, how, and why the code was added to a repository.

Outline

- 1 Need for Version Control Systems
- 2 Git Basics
- 3 Merge Conflicts
- 4 Cool Features
- 5 VCS Hosting**
- 6 In-Class Exercise on VCS
- 7 Summary and Questions

Cloud-based VCS hosting

- ▶ Github
- ▶ Bitbucket
- ▶ Gitlab

Github

We use Github. Note the following

- ▶ Github provides free accounts. Sufficient for this class.
- ▶ Github requires a token or a pair of public/private keys to push the code to the hosted repository.
 - ▶ For token access, follow [Creating a personal access token](#)
 - ▶ For public/private key access, follow [Adding a new SSH key to your GitHub account](#)
 - ▶ Recommend public/private key access

Outline

- 1 Need for Version Control Systems
- 2 Git Basics
- 3 Merge Conflicts
- 4 Cool Features
- 5 VCS Hosting
- 6 In-Class Exercise on VCS**
- 7 Summary and Questions

In-Class Exercise

Check out the assignment on Blackboard

Outline

- 1 Need for Version Control Systems
- 2 Git Basics
- 3 Merge Conflicts
- 4 Cool Features
- 5 VCS Hosting
- 6 In-Class Exercise on VCS
- 7 Summary and Questions**

Summary

Questions?

- ▶ Team Project
- ▶ VCS
- ▶ git
- ▶ Common commands
- ▶ Recommendation: Start with a `.gitignore` file
- ▶ Recommendation: use the command line
 - ▶ You are required to learn it
- ▶ Recommendation: don't use branches (the project is small, and learn that in the future)
- ▶ Careful with `git push -force` and similar commands
 - ▶ Check out <https://ohshitgit.com/>