Why You Want Version Control (git in particular)

Steve Johnson and Tim Henderson

4 Feb 09

Summary

- What is version control?
- Common actions
- Current popular version control systems
- Git!
- How to use git

What Is Version Control?

- “Save Game” for your text (and binary) files
  - Code
  - Research papers
  - TPS reports
- Collaboration tool
  - Easy to share
  - Merging
- Tracks changes
  - See who broke it, how, and when

Personal Version Control Actions

- Add files
  - What to save
- Commit
  - “Save game”
- Revert
  - “Reload game”

Collaborative Version Control Actions

- Push/pull
  - Give changes to others, get changes from others
- Branch
  - Preserve existing copy, work on experimental copy
- Merge
  - Put changes from experimental copy back into main

Common VCS

- CVS
  - STAY AWAY
- Subversion (svn)
  - Centralized
  - “CVS Done Right”
  - Until recently, predominant VCS for large open-source projects
- Bazaar (bzr)
  - Centralized and/or distributed
  - Branch from Subversion
  - Python!
  - Ubuntu

Common VCS, cont.

- Git
  - Distributed
  - Developed for use with the Linux kernel
  - By Linus et al
- Mercurial (hg)
  - Distributed
  - Developed for use with the Linux kernel
  - Essentially similar to git, but less tied to UNIX
  - Python!

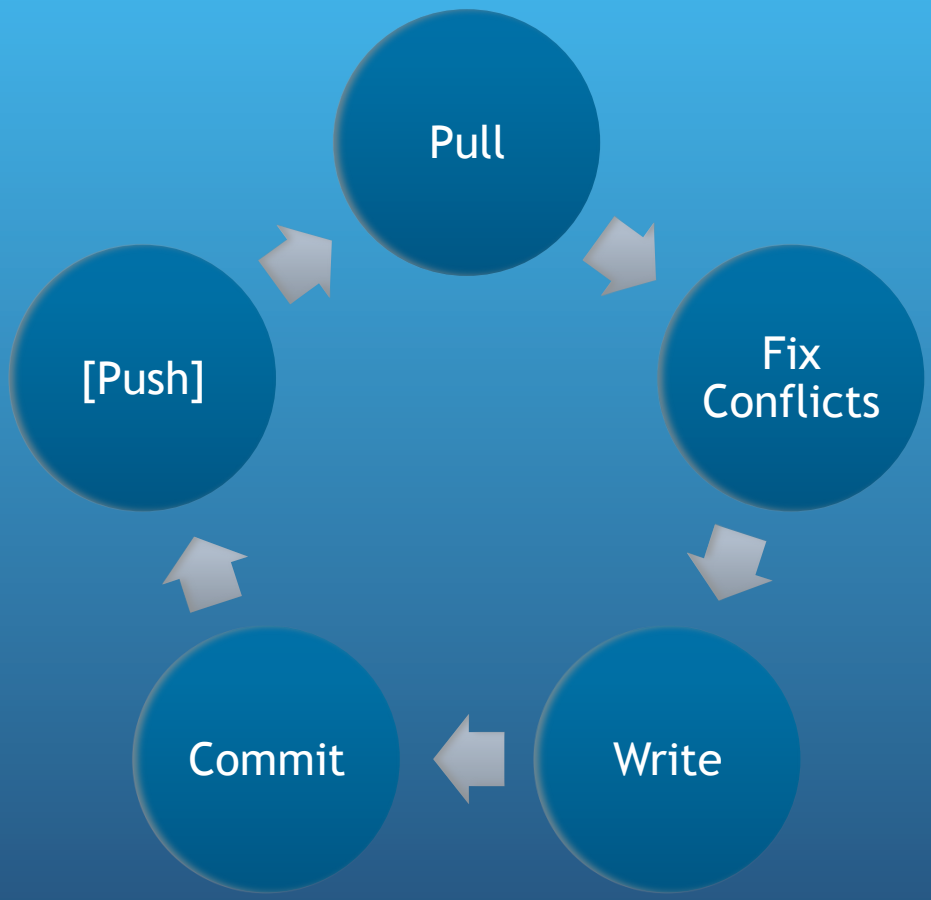
Git

- Distributed
  - Every working copy *is* a repository
  - All commits are local
- Clean
  - Creates a single .git directory
- Fast and small (Mozilla repo: 450 MB)

Git use cases for students

- Track local projects
- Collaborate on class projects
- Contribute to Hacker Society projects via Github
- Marketable skillz

Typical Workflow



Demo

- Init, add
- Commit
- Branch, merge
- Conflicts
- Collaboration

Github!

- MySpace for Code
  - But with excellent design
- Primary reason why git is better than Mercurial
- Unlimited free public repositories
  - Private repositories for a fee
- Excellent integration with EVERYTHING
- Fork any public project
  - Changes/branches/merges tracked
- Everything has pretty charts