Subversion and Git

Problems with “Sneakernet”

- Hard to keep everyone in sync
- Hard to keep a log of what/why changes were made
- Impossible to roll back to a previous version

Solution: Revision Control

- Two paradigms: Centralized and Decentralized
- Basic idea: track changes to a code project over time, and be able to share those changes

Centralized Revision Control

- Basic Idea: code exists on a server, you can get some revision of it, work on it, and push your updates back to the server.
- Advantages
  - Very easy to understand
  - Very standard
- Implementations: CVS, Subversion, etc

Terminology*

- Checkout: get a fresh copy of code from a server
- Update: get the most recent copy of code from the server
- Commit: send a set of changes to the server
- Commit Message: a sentence or two describing what's in your commit
- Diff: a standard file format (.diff, .patch) for storing changes between two files (or sets of files)

*Subversion terminology!

Subversion

- Checkout
  - svn co svn+ssh://username@host/path/to/repo [./localname]
  - Other protocols are svn://, http://, and https://
- Update
  - svn up
  - Run in your local copy (or any subdirectory of your local copy)
  - You can use svn up -rREV to “update” to a past revision REV

*You can use “checkout” instead of “co” and “update” instead of “up”... but why would you?

Subversion

- Add
  - svn add
  - “Adding” files lets subversion keep track of them
- Delete
  - svn rm / svn del
  - Deletes a file from being tracked under subversion
  - ALSO NUKES YOUR LOCAL COPY
- Status
  - svn status
  - Shows you what files have been changed, added, or deleted

Subversion

- Commit
  - svn ci
  - This will prompt you to enter a commit message
  - You can also specify svn ci -m “message” on the command line
- History
  - svn log
  - Show the commit history for a project

Subversion

- Diff
  - svn diff
  - By default, the differences between your local copy and the most recent revision
  - Useful for seeing what changes you've made
- Revert
  - svn revert .
  - Roll back your changes
  - BE CAREFUL, YOU CAN'T [easily] UNDO THIS

Subversion Conventions

- Three directories in a repository
  - /trunk
    - “Bleeding edge”: it probably works
    - Not suitable for release
  - /tag
    - Copies of code marked as “stable” at some point in time
  - /branch
    - Copies of code that have been taken in a slightly different direction
- You probably only care about /trunk at this point

Why Subversion Sucks

- Network reliant
  - You need a server to do anything
- Centralized
  - If the server goes down, what do you do?
- Big
  - You only store one revision of the code on your computer at a time

Decentralized (Distributed) Revision Control

- Basic Idea: track changes (and commits) separately from sharing those changes
- Committing changes and pushing them to the server are different operations
- Commit lots of times (microcommitting), Push once (“atomic commit” paradigm)

Git

- Developed by Linus Torvalds
  - You may know him better for “Linux”
  - Written to track the Linux Kernel Sources

Git Terminology

- Clone: get a fresh copy from the server
- Pull: get changes from the server
- Commit: make a record of your changes
- Push: save that record of changes to the server
- Branch: make a “branch” in the history of a file: change a file in an alternate source tree
- Merge: combine a branch back in to the tree
- Checkout: like subversion's revert

Git

- Clone
  - git clone https://user@url/path/to/repo
- Pull
  - git pull
- Commit
  - git commit
  - You must specify which files you want to commit (-a for all changed files)
  - You can specify the commit message with -m (like svn)

Git

- Push
  - git push
- Branch
  - Make a new branch: git branch <branchname>
  - Switch to a branch: git checkout <branchname>
  - Commit like normal in the new branch
  - Default branch is called “master”
  - Delete a branch with git branch -d <branchname> (after merging in your changes)
  - “Hard delete” a branch with git branch -D <branchname> (dangerous!)

Git

- Merge
  - git merge <branchname>
  - Merges a branch called <branchname> into the current one
  - Git will try to auto-merge changes, and will tell you when it can't
- Checkout
  - Change to a fresh copy of a branch
  - “fresh copy” can mean a new branch or a clean copy of an old one

Git

- Logging
  - git log
  - Show the history of your tree
  - There are some neat options for this
- Status/Info
  - git status
    - Like svn status
  - git info
    - Gives you information about the repository

Remember

- Ask for help
- RTFM
- Google is your friend