blue[1]

GitNotes

Charlie Jung

May 22, 2018

*Kalid Azad from A Visual Guide to Version Control

Version Control:
A good **VCS** has these elements:
Backup and Restore: Files are saved as they are edited
Need a file from May 27, 2018? No problem!
Synchronization: People can share files and stay up-to-date with the latest version
Short-Term Undo: Throw away your changes?
You can go back immediately to the last *stable* version
Long-Term Undo: Go back to a project a year ago to fix a bug
Track Changes: As files are updated: you can leave messages ("commit messages") explaining why the change happened. (in VCS; not the file)
Track Ownership: VCS tags every change to a user
Sandboxing: Making a big change? Make temp changes in an isolated area before testing them out and checking in your changes.
Branching and Merging: You can work on a new branch merge later
Language and Technical Jargon:
Basic Actions:
Add: Put a file in the repository for the first time.
Begin Tracking with Version Control)
Revision: What version a file is? (v1, v2, etc...)
Head: Latest Revision in a repository
Check Out: Download a file from a repository
Check In: Upload a file to a repository.
File gets a new revision and people can check the latest Revision out
Check In Message: A short message describing what has changed
Changelog/History: A list of changes made to a file since it was created
Update/Sync: Synchronize your **local files** with the latest from the repository.
This lets you grab all the latest revision of all the files
Revert: Throw away your local changes and reload
the latest version from your repository
Advanced Actions:
Branch: Create a separate copy of a file/folder for private use (bug fixing, testing)
Branch is both a verb ("Branch the code") and a noun ("Which branch is it in?)
Diff/Change/Delta: Finding the difference between two files
Merge or Patch: Apply changes from one file to another: Bring it up to date
Conflict: When pending changes to a file contradict each other (both changes cannot be applied)
Resolve: Fixing the changes that contradict each other and checking in the correct version
Locking: Taking control of a file so nobody can edit it unless you unlock it
Breaking the Lock: Forcibly unlocking a file so you can edit it
Check out for edit: Checking out an "editable" version of a file
Check Ins:

Simple Scenario:
Checking in a file (list.txt) and modifying it over time.
Basic Check In:

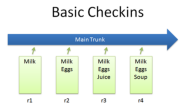

Figure 1: Basic Check In Diagram

Subversion Example:
svn add list.txt
(modify the file)...
svn ci list.txt -m "Changed"
Checkout and Edit:

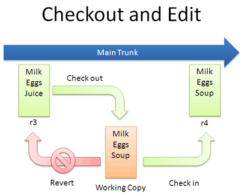

Figure 2: Basic Check Out and Edit Diagram

Subversion Commands:
svn co list.txt (get latest version)
... edit file ...
svn revert list.txt (throw away)
svn co -r2 list.txt (check out a particular revision)
Diffs:

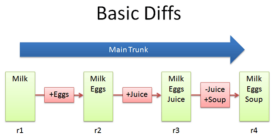

Figure 3: Basic Differences Diagram

Most VCS's store: src file and diffs
Subversion:
svn diff -r3:4 list.txt
Branching:

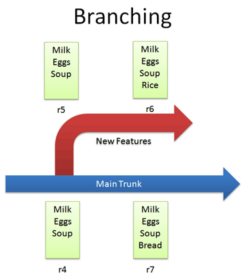

Figure 4: Basic Branching Diagram

Branches let us copy code into a separate folder
In Subversion, you create a branch by copying a directory to another
svn copy http://path/to/trunk http://path/to/branch
Merging:
Let's say we wanted to get the "rice" feature from our experimental branch into the main line. How would we achieve this?
Diff r6 and r7 and apply that to the main line.
Nope. We want to apply changes that happened in the specific (our case, experimental) branch.
We diff r5 & r6 and apply it to the main trunk
Diagram:

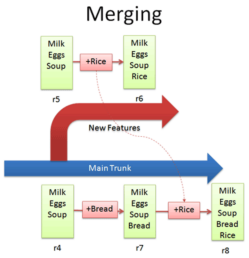

Figure 5: Basic Merging Diagram

In Subversion: Merging is very close to diffing. Inside the main trunk:
Run in SVN:
svn merge -r5:6 http://path/to/branch
(Note: Subversion does not currently track what merges have been applied: If you are not careful, your changes could
be applied twice.)
Current Advice: Keep a changelog message reminding you that you have already merged r5 to r6 in main
Conflicts:

Many times, the VCS can automatically merge changes to different parts of a file.
Conflicts can arise when changes don't sit well:
Joe wants to remove eggs and replace it with cheese (-eggs, +cheese)
Sue wants to replace eggs with a hot dog (-eggs, +hotdogs)
Conflicts

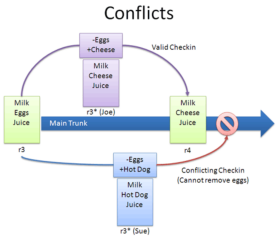

Figure 6: Basic Conflicts Diagram

The quicker check-in wins out:
If VCS reports a conflict:

- Re-apply your changes
- Override old changes with your own changes

Tagging:

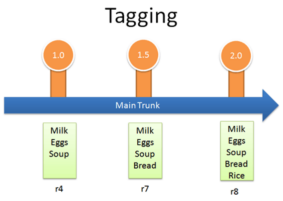

Figure 7: Basic Tagging Diagram

(in trunk)
tags are named branches (non-editable)
svn copy to/revision to/tag
Real Life Example:
MainLine → *StableWindows*
Each part (UI, Networking) has a unique branch
Reverse Integration: From branch → *trunk*
Forward-Integrate: Receive Latest Patches from main into their own branch ("pull")
Git Commands:

git clone
git config
git add
git status
git commit
git push
git pull
git branch
git checkout
git merge
Branches are just pointers to commits