Working with Git

Observations from a Newbie

Bruce Cota

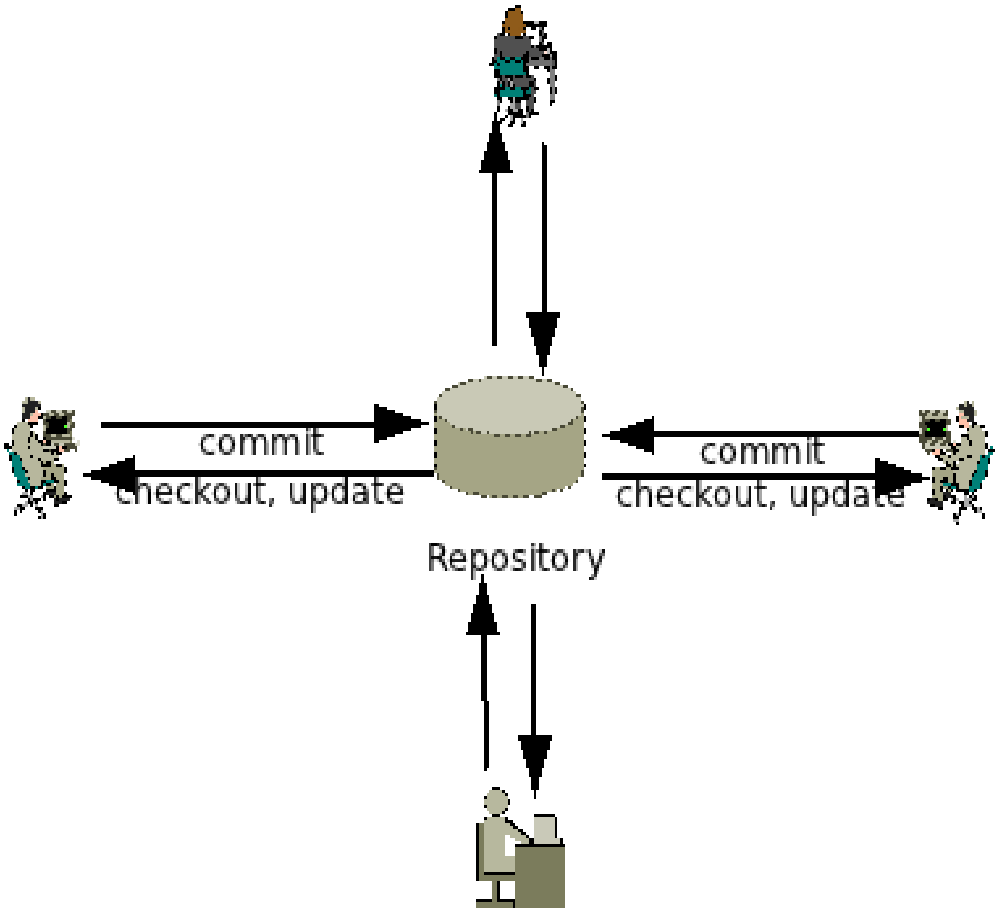
February 7, 2012

Goals

- Why Git?
- Typical Workflow with Git
- How to Visualize the Git Repository

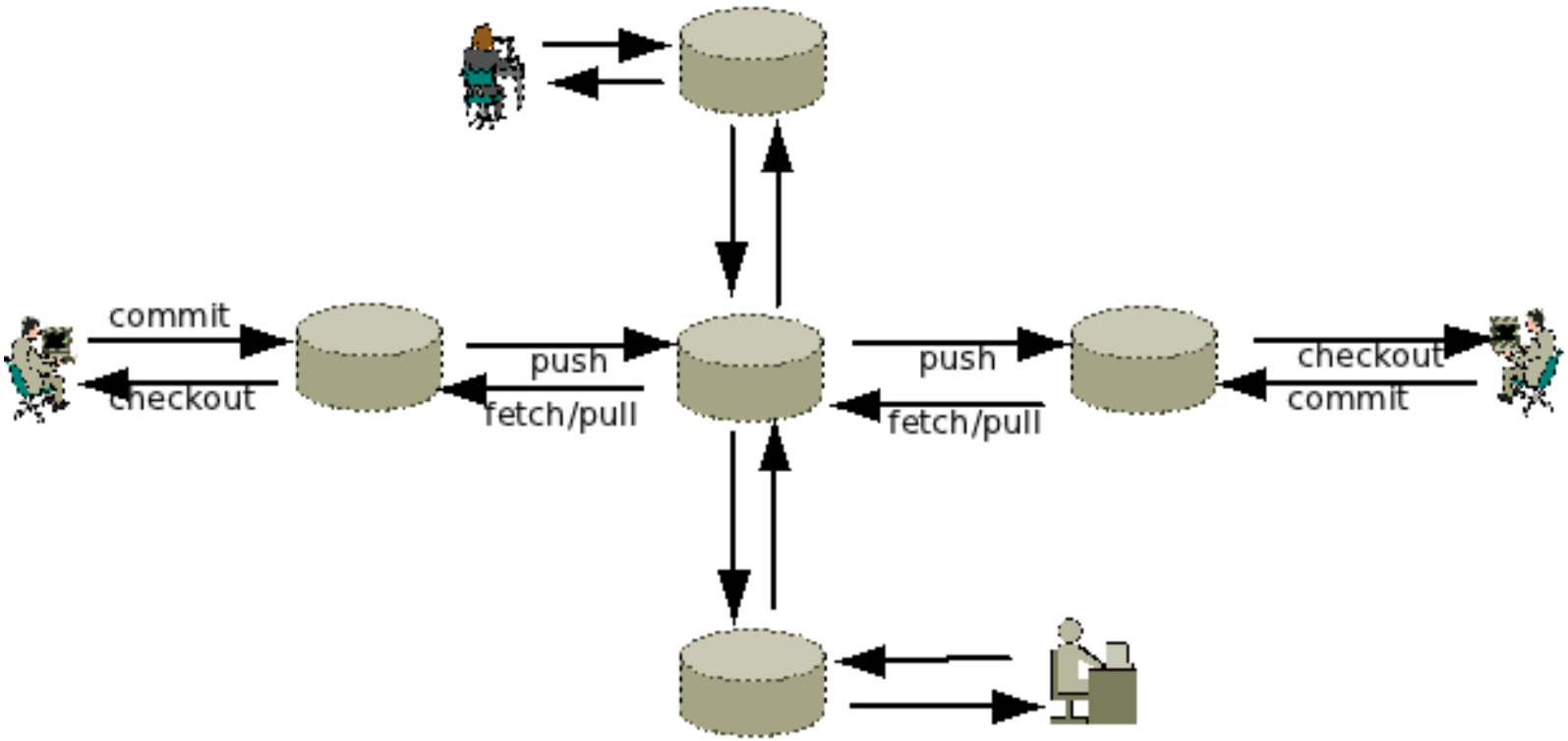
Apologies in advance for any inaccuracies.
Don't ask me too many questions :)

Development Team Working with Subversion



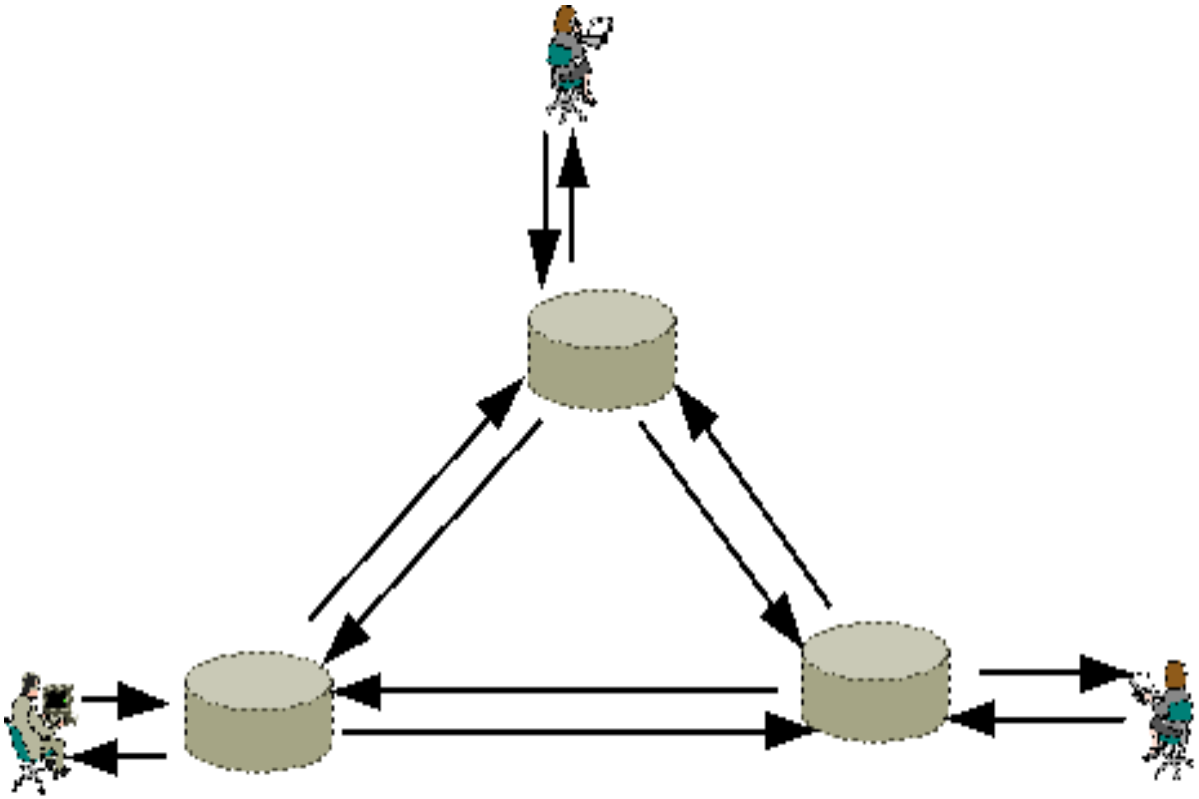
Developers sharing a subversion repository.

Development Team Working with Git



Each developer has her own repository!

Peer to Peer Version Control



Maybe your used to seeing a diagram like this?
Is this really the norm?

Why Use Private Repositories?

Version control is useful even for a team of one.

- incrementally logging your own work.
- rolling back to previous versions.
- branches for experimenting.

It is difficult to version control your own work without a private repository

Coarse Grained Version Control vs Fine Grained Version Control

- **Coarse Grained** Auditing and merging of units of work that are complete enough to share with the rest of the team.
- **Fine Grained** Auditing and merging of units of work that are untested or "half baked", and would disrupt the work of the rest of the team.

- Subversion
  - **Coarse Grained** commits and updates
  - **Fine Grained** not really
- Git
  - **Coarse Grained** push, fetch/pull
  - **Fine Grained** commit

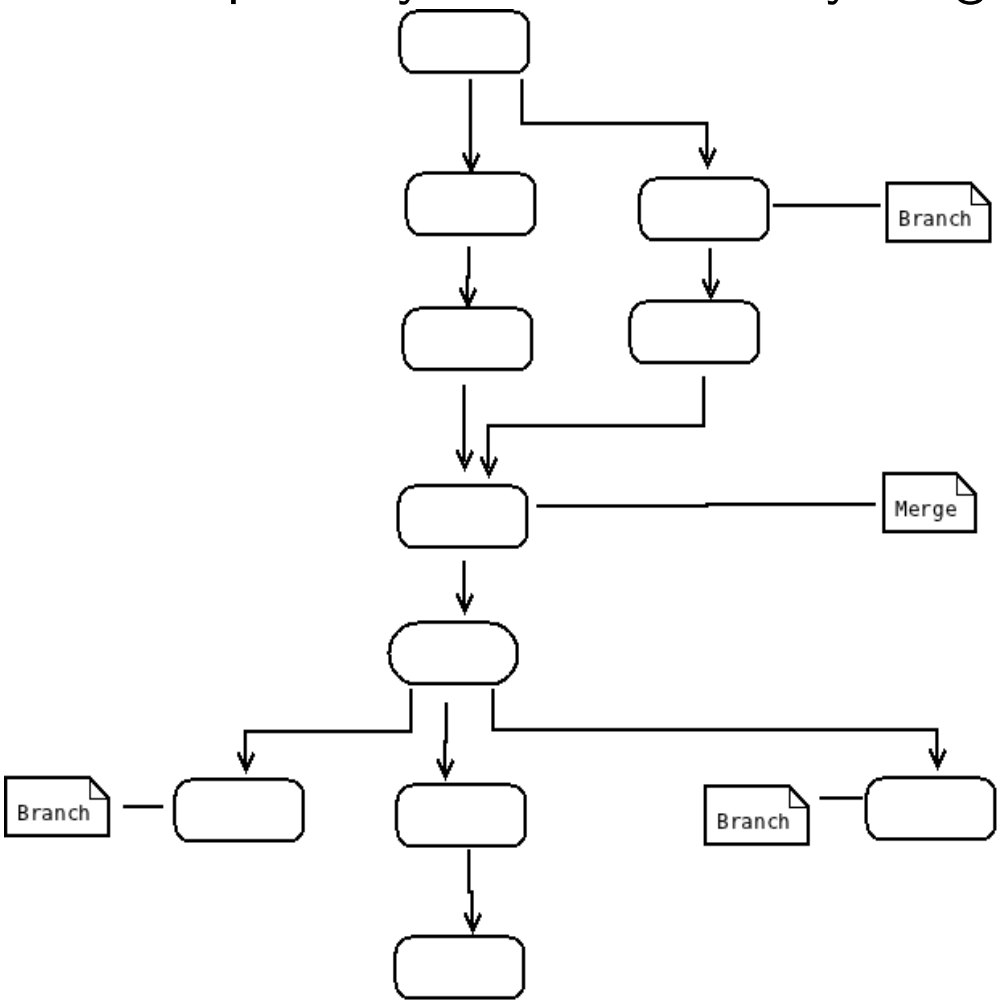
Brief History

- SCCS and RCS version controlled individual files.
- CVS version controlled every file in a directory (using RCS).
- Subversion bypassed RCS and felt like a (mostly) improved CVS.

Personally, I continue to think of a Subversion repository as a directory of version controlled files.

Git Repository

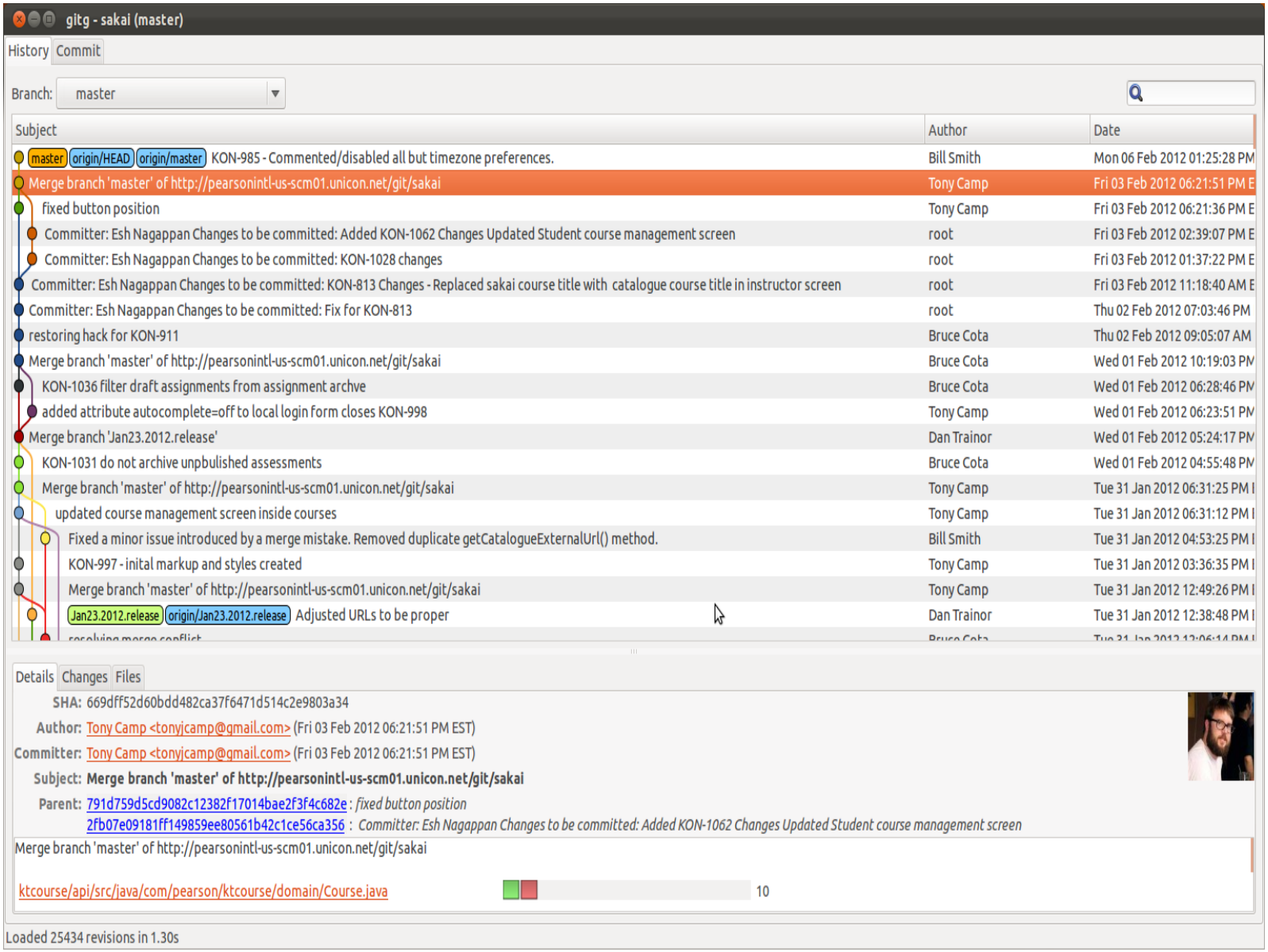
A Git Repository is a directed acyclic graph of commits.



Git Concepts

A commit is basically a snapshot of the working directory.
A commit can have multiple children if you branch from it.
A commit with multiple parents is created by a merge.
A branch is a swimlane of commits.

gitg


gitg - sakai (master)
History
Commit
Branch:
master
Subject
Author
Date
master
origin/HEAD
origin/master
KON-985 - Commented/disabled all but timezone preferences.
Bill Smith
Mon 06 Feb 2012 01:25:28 PM
Merge branch 'master' of http://pearsonintl-us-scm01.unicon.net/git/sakai
Tony Camp
Fri 03 Feb 2012 06:21:51 PM E
fixed button position
Tony Camp
Fri 03 Feb 2012 06:21:36 PM E
Committer: Esh Nagappan Changes to be committed: Added KON-1062 Changes Updated Student course management screen
root
Fri 03 Feb 2012 02:39:07 PM E
Committer: Esh Nagappan Changes to be committed: KON-1028 changes
root
Fri 03 Feb 2012 01:37:22 PM E
Committer: Esh Nagappan Changes to be committed: KON-813 Changes - Replaced sakai course title with catalogue course title in instructor screen
root
Fri 03 Feb 2012 11:18:40 AM E
Committer: Esh Nagappan Changes to be committed: Fix for KON-813
root
Thu 02 Feb 2012 07:03:46 PM
restoring hack for KON-911
Bruce Cota
Thu 02 Feb 2012 09:05:07 AM
Merge branch 'master' of http://pearsonintl-us-scm01.unicon.net/git/sakai
Bruce Cota
Wed 01 Feb 2012 10:19:03 PM
KON-1036 filter draft assignments from assignment archve
Bruce Cota
Wed 01 Feb 2012 06:28:46 PM
added attribute autocomplete=off to local login form closes KON-998
Tony Camp
Wed 01 Feb 2012 06:23:51 PM
Merge branch 'Jan23.2012.release'
Dan Trainor
Wed 01 Feb 2012 05:24:17 PM
KON-1031 do not archive unpbulished assessments
Bruce Cota
Wed 01 Feb 2012 04:55:48 PM
Merge branch 'master' of http://pearsonintl-us-scm01.unicon.net/git/sakai
Tony Camp
Tue 31 Jan 2012 06:31:25 PM I
updated course management screen inside courses
Tony Camp
Tue 31 Jan 2012 06:31:12 PM I
Fixed a minor issue introduced by a merge mistake. Removed duplicate getCatalogueExternalUrl() method.
Bill Smith
Tue 31 Jan 2012 04:53:25 PM I
KON-997 - inital markup and styles created
Tony Camp
Tue 31 Jan 2012 03:36:35 PM I
Merge branch 'master' of http://pearsonintl-us-scm01.unicon.net/git/sakai
Tony Camp
Tue 31 Jan 2012 12:49:26 PM I
Jan23.2012.release
origin/Jan23.2012.release
Adjusted URLs to be proper
Dan Trainor
Tue 31 Jan 2012 12:38:48 PM I
Details
Changes
Files
SHA: 669dff52d60bdd482ca37f6471d514c2e9803a34
Author: Tony Camp <tonyjcamp@gmail.com> (Fri 03 Feb 2012 06:21:51 PM EST)
Committer: Tony Camp <tonyjcamp@gmail.com> (Fri 03 Feb 2012 06:21:51 PM EST)
Subject: Merge branch 'master' of http://pearsonintl-us-scm01.unicon.net/git/sakai
Parent: 791d759d5cd9082c12382f17014bae2f3f4c682e : fixed button position
2fb07e09181ff149859ee80561b42c1ce56ca356 : Committer: Esh Nagappan Changes to be committed: Added KON-1062 Changes Updated Student course management screen
Merge branch 'master' of http://pearsonintl-us-scm01.unicon.net/git/sakai
ktcourse/api/src/java/com/pearson/ktcourse/domain/Course.java
10
Loaded 25434 revisions in 1.30s

Typical Subversion Workflow

1. repeat until ready to share
   1. edit some files
   2. update from repo and resolve conflicts
2. commit changes

Typical Git Workflow

1. create a new branch
2. repeat until ready to share
    1. edit some files
    2. commit changes
    3. occasionally pull "master branch" and merge them into the new branch
3. merge new branch into master
4. commit changes to master
5. push changes to shared repository

That looks like a lot more work. What do I get for it?
Well, it's a *little* more work – at least it is more git commands.
Some things you get for it

1. A history of your work
2. A convenient way of recovering changes you made between commits ("git diff")
3. Easy way of switching between the "master branch" and private branches.
4. Multiple branches for switching between multiple changes

Demo

Demo

Traps for SVN Users

1. Confusing “git commit” with “svn commit” (which is easy, since they are virtually identical).
2. Replacing “svn checkout with “git clone”.
3. Cutting and pasting changes across branches instead of using “git merge” or “git diff”.
4. Unnecessary fear of messing up your repository.

“svn commit” vs “git commit”

- git commits are not visible to the rest of the team until pushed.
- Knee jerk reaction may be to use “git commit; git push” whenever you would have used “svn commit”.
- Don’t do that. You should be doing many commits between pushes. That is the *whole point* of using git.
- In fact, you should often be committing to a branch.

“git commit” is practically identical to “svn commit”, but with respect to a private repository.

Don't use "git clone" in place of "git checkout"

- Every git repository has exactly one working directory. (Well, "bare repositories" have no working directory.)
- "git clone" initially feels like "svn checkout" because both create your working directory. But "git clone" creates a *repository*.
- SVN: If you need to work on two branches, two svn checkouts to create two different working directories.
- GIT: You *can* use "git clone" to create two different working directories, but then you have two repositories, and you can't merge or compare without pushing and pulling.
- GIT: You *probably* should work on both branches from one repository, and switch between using "git checkout".
- What if you have changes you really don't want to commit? Try "git stash".

Make maximum use of git merge and git diff

- If working on two branches, try to arrange to merge one branch into the other, instead of cutting and pasting (which might make it harder for other merges later).
- if merging doesn't work, generating a patch is usually easy:
  git diff 1e4a72...42e7b0f^ 1e4a72...42e7b0f > /tmp/patch
  git diff 1e4a72...42e7b0f^^^ 1e4a72...42e7b0f > /tmp/patch
  git diff 1e4a72...42e7b0f 1e4a72...42e7b0f^> /tmp/patch

Don't be afraid to wreck your repository

- It can be nerve wracking to change a shared subversion repository.
- Git is a more complicated than subversion, and seems scarier.
- But *your* git repository is just a directory on *your* computer. So just zip it up and experiment!
- The time to be nervous is when you push ...

A Cornucopia of Git Commands

- 146 files in /usr/lib/git-core (ubuntu). Most have many options
- Seems to be a solution to every problem if you look hard enough.
- Personal recent favorites
  - git log –since 'one day ago'; git log –author Cota
  - git stash; git stash list; git stash pop
  - git svn clone
  - git diff for generating patchs
  - git pull -s ours
    git pull -s recursive -X theirs

A Couple of Usage Patterns

- Sharing a branch with another developer (without going peer to peer)
- Maintaining a release branch.

Sharing Branches without Going Peer to Peer

It's pretty easy ...

- git push origin *your-branch*
- git pull origin *your-branch*

Managing a Release Branch

One possibility

1. Branch:
   git checkout -b ReleaseBranch
   git push origin ReleaseBranch;
2. Whenever work is done on the ReleaseBranch, immediately merge it into the master branch
   git push origin ReleaseBranch
   git checkout master
   git merge ReleaseBranch
   git push

Managing a Release Branch (cont.)

3 If a change that should *not* be merged into master, generate a reverse patch and apply it to master.
git push origin ReleaseBranch
git diff HEAD HEAD^ > /tmp/patch
git checkout master
git merge RelaseBranch
patch -p1 < /tmp/patch
git add –all
git commit -a
git push
Now you can keep merging future changes.

Conclusion

- If you're still using subversion, then start using Git.
- If the rest of your team is stuck on Subversion, then just use git-svn.
- Questions? (I probably won't have answers)
- Comments?
- Corrections?