Version Control with Svn, Git and git-svn

Kate Hedstrom

ARSC, UAF

Version Control Software

- System for managing source files
  - For groups of people working on the same code
  - When you need to get back last week's version
- In the past, I have used RCS, CVS, and SVN, each better than the last
- Git is the newest widely used open source version control system

Getting Started With SVN

- Tell it where the archive is with a URL:

```
file:///local/path
        or
http://host/path
```

- For a new archive:

```
% svnadmin create /local/path
```

- From an existing archive:

```
% svn checkout URL
```

Local Files

- After a checkout, svn will keep a private copy of each file under the .svn directory
- You will also have the “sandbox” files, for you to use and edit
- “svn diff” will show differences between them
- Can only point to one repository

Main svn commands

- import - bring sources into a repository
- checkout - get sources out of the repository
- commit - check in changes
- update - get changes from repository
- status - find out which files would change on commit
- diff - find out what you changed
- help

Read-only svn Commands

- Checkout – get sources out of the repository
- update – get changes from repository
- status – find out which files you have changed
- diff – find out what you changed
- info – find URL of the repository
- log – see the history
- help

Revision Numbers

- Svn uses a database to store the files on the server
- The whole project has one revision number to describe that snapshot
- Can see the numbers with "svn log"
- Every commit creates a new revision number

Updates

- **An update when two people have changed things can involve:**
  - No conflict because changes are in different places
  - Conflict because two different changes to the same region in the code
- **If there is a conflict this must be resolved by human intervention**
- **One option is to revert (undo)**

Conflicts

- **If there is a conflict, svn will provide you with four files:**
  - The original file (filename.mine)
  - The older file from the trunk (filename.r41)
  - The newer file from the trunk (filename.r45)
  - A merge attempt (filename)
- **The merge failures will look something like:**

```
 Clean code before
<<<<<<< .mine
My code
=======
New code
>>>>>>> .r45
Clean code after
```

- **Once you've cleaned up the mess, tell svn you're ready:**

```
svn resolved filename
```

- **This will cause svn to delete the extra files and allow a commit to take place**
- **You can instead toss your changes with:**

```
svn revert filename
```

Learn more

- Version Control with Subversion, by Collins-Sussman et al., 2004, O' Reilly
- Online at http:svnbook.red-bean.com/
- svn help

Onward to Git

- **Git was designed by Linus Torvalds for managing the Linux kernel and therefore has these goals:**
  - Fast
  - Support many users
  - Distributed

Distributed?

- Every checkout gives you a copy of the whole repository
- Can compare branches, history while offline
- Can check in your changes to your local repository
- Sharing updates with others is optional

Why change from svn?

- **For our needs as ROMS users and developers, git solves some problems:**
  - Save our own changes
  - Apply patches from the repo one at a time – especially for those waiting months between updates
  - “git format-patch” and “git am” smoother than “diff” and “patch” in the face of conflicts

Getting Started With Git

- **Set up who you are:**

```
% git config ––global user.name “you”
% git config ––global user.email    \
  “you@home”
```

- **Get colorful (if you want):**

```
% git config ––global color.ui “auto”
```

- **Without “--global” applies to current directory only**

Start a New Repository

- **In the directory with your code:**
  - git init
  - git add .
  - git commit -m “initial message”
- **You now have a .git directory with a database of your files**
- **Revision numbers are SHA1 numbers, same for the same content**

From a Repository

- **From a git url:**
  - git clone <url>
- **Could be another local directory:**
  - git clone dir1 dir2
- **From an svn url:**
  - git svn clone <url>
- **Default is to suck down the entire history into the database**

Main git commands

- add – add sources to next commit
- commit – check in changes locally
- checkout – change branches
- push – send your changes to a remote site
- pull/fetch – get changes from remote site
- status – find out which files would change on commit
- diff – find out what's different between index and current sandbox
- help

Example

- **Change/add some local files**
  - git add newfile
  - git commit
- **"git add" adds files to the commit list for the next commit**
- **Can selectively add only some of your changes to make logical commits**
  - git commit -a   #commits all changes

Git example

```
% ls /my/src/cpp
cpp.h  cpp.c   Makefile  ...
% cd /my/src/cpp
% git init
# Tell git which files to track
% git add .
% git commit
[make some changes]
% git commit -a
```

Comments on Previous

- Svn requires you to set up branches, tags, trunk – no more
- Svn requires you to delete the starting directory and checkout fresh – no more
- Tracked files have to be explicitly added

What about Branches?

- **See the branches:**
  - git branch
- **Make a new branch:**
  - git branch <new>   # copy of current
- **Switch to that new branch:**
  - git checkout <new>
- **Both in one:**
  - git branch -b <new>

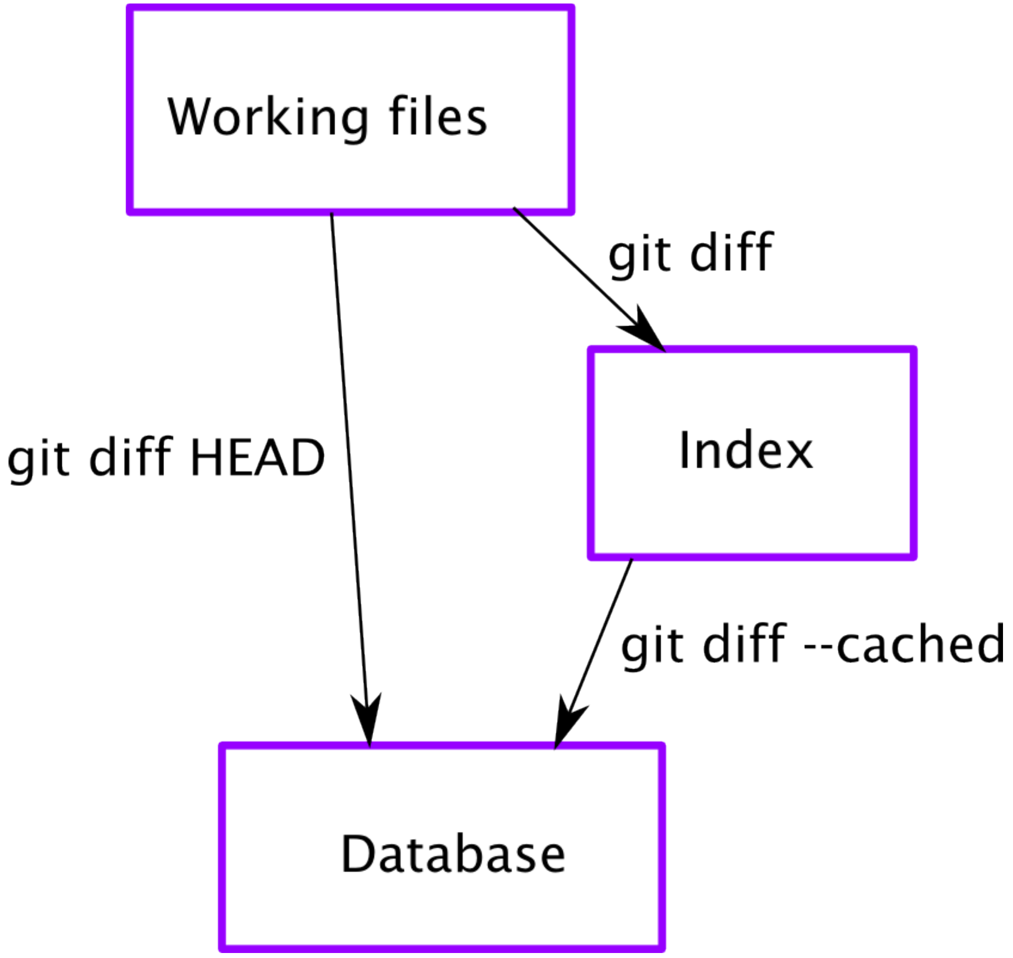
Seeing History

- git log
- gitk (gui)
- git diff HEAD^
- git log HEAD^^^ or HEAD~3
- git diff b324a87 (SHA1)
- git diff --cached (between index and HEAD)

Index?

- The index is a store of what would be checked in on "commit"
- Contains files that merged cleanly
- "git diff" shows difference between index and current sandbox
- "git diff HEAD" shows difference between last checked in and sandbox

Index as Staging Area



Coordination

```
# on midnight
% git clone <URL> roms
% cd roms
[make some changes]
% git commit -a
% git push origin master
```

```
# on cygnus
% git clone …
% cd roms
% git pull
% make
```

- **Coordinate code on multiple systems**
- **Coordinate between multiple programmers**
- **Can be single version or multiple branches**

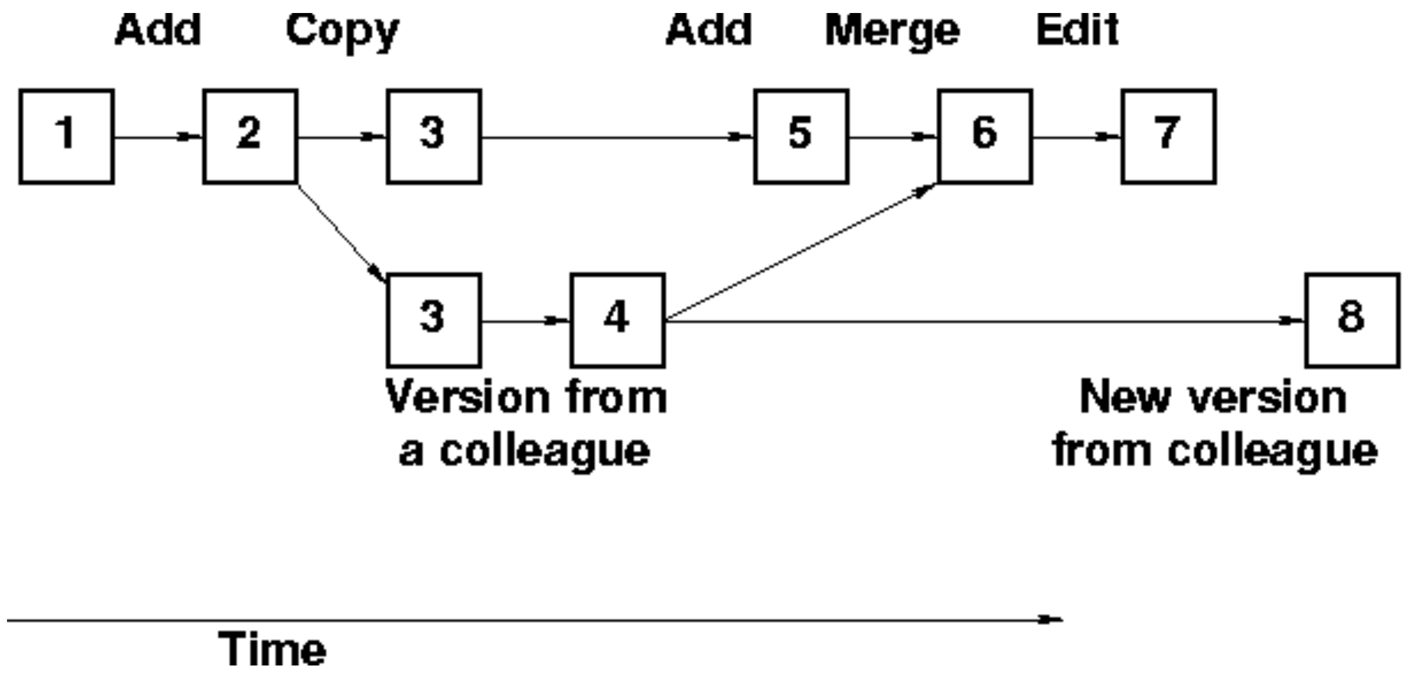
Other git commands

- delete – no longer need a file
- move – rename a file or move to new location
- merge – merge changes from another branch
- cherry-pick – pick one update from some other branch
- remote – register a remote repo
- rebase – reorder the history in your local repo (scary stuff!)
- Tag – nicer name for a specific SHA1

Revision Numbers

- git uses a database to store the files locally
- The branch has one revision number to describe that snapshot – it’s a SHA1 with 40 characters
- Can see the numbers with “git log”
- Every file and every tree of files has a unique SHA1 number

Branches



- Branch can differ by a few files or every ROMS file
- Branch creation is lightweight
- Rebase can be used to put change 5 after 6

Updates

- **An update when two people have changed things can involve:**
  - No conflict because changes are in different places
  - Conflict because two different changes to the same region in the code
- **If there is a conflict this must be resolved by human intervention**
- **One option is to reset (undo)**

Conflicts

- **If there is a conflict, git will let you know**
- **The merge failures will look something like:**

```
Clean code before
<<<<<<< HEAD:<file>
My code
=======
New code
>>>>>>> branch:<file>
Clean code after
```

More Conflicts

- Once you've cleaned up the mess, tell git you're ready:

```
git add filename
```

- This will cause git to place the new version into the index
- You can instead toss your changes with:

```
git checkout HEAD filename
```

- Once all the files are clear (check with "git status") commit the index to the repo:

```
git commit
```

Git Svn Handshaking

- **Not quite as robust as git alone**
- **Based on Perl scripts in svn distribution (not always installed)**

```
git svn clone <url>
git svn clone -s -r 1043 <url>
git svn rebase          # fetch from upstream
git svn dcommit         # commit to upstream
git svn log
```

Git Drawbacks?

- Best with one project per repository (roms, plotting, matlab tools all separate entities)
- Yet another tool to learn
- Git-svn doesn't handle svn Externals
- ROMS expects valid svn entries in Id tags
- More rope to hang yourself…

My Insane Repo Collection

- Bare repository on cygnus (Linux workstation)
- Cloned to each supercomputer via ssh
- Cloned to Enrique's system via ssh
- git-svn only working on Mac laptop
- Mac has my git-svn directory, plus clone of cygnus repo, also NCAR CCSM-ROMS and Hernan’s trunk, both via git-svn

My Branches

- Copy of the svn code
- Copy of the same code in the bare cygnus repo
- Copy of the fish branch
- Any other thing I'm working on temporarily, like CICE coupling

Learn more

- **Version Control with Git, by Jon Loeliger, 2009, O' Reilly**
- **Online at http://git-scm.com/documentation - there are even videos**
- **git help**
- **If you like these ideas, but prefer a Python tool, check out Mercurial at: http://mercurial.selenic.com/**