G4LBNE Migration from CVS to git and some other stuff

Brett Viren

Physics Department

BROOKHAVEN NATIONAL LABORATORY

G4LBNE phone meeting 2014/2/13

Outline

The Migration

Basic `git`

Build Issues

Summary

Overview of Migration

1. Backed up old CVS
2. Run `cvs2git` following this Fermilab Redmine page but had to run it on my workstation as it failed on the Fermilab server.
3. Create initial `git` repository for `cvs2git` data
4. Tar it up, copy it to server and unpack

Cruft

```
$ ls -l /cvs/projects/lbne-beamsim/
drwxrwsr-x. 3 p-lbne-beamsim redmine 6144 Feb  7 12:02 CVSROOT
drwxrwsr-x. 5 p-lbne-beamsim redmine 4096 Sep 13 10:04 g4lbne
drwxrwsr-x. 7 p-lbne-beamsim redmine 2048 Feb 11 10:24 g4lbne.git
drwxrwsr-x. 6 p-lbne-beamsim redmine 4096 Apr 23  2013 g4numi
drwxrwsr-x. 4 p-lbne-beamsim redmine 2048 Feb 11 09:30 lbne-beamsim

$ ls -l /cvs/projects/lbne-beamsim/lbne-beamsim/
drwxrwsr-x. 13 p-lbne-beamsim redmine 4096 Feb 11 09:30 g4lbne
drwxrwsr-x.  7 p-lbne-beamsim redmine 4096 Feb 11 09:30 g4numi
```

- Multiple copies of `g4lbne` and `g4numi` CVS directories +`CVSROOT`
- `g4lbne.git` is made from `lbne-beamsim/g4lbne/`

Any reason to keep the CVS cruft around?

Redmine Changes

Modified Redmine w.r.t. repositories

- "Deleted" CVS (actual repository still there)
- Added the new `git` as primary repository
- Added a secondar `worch` repository

Still browse at usual location:

```
https://cdcvs.fnal.gov/redmine/projects/lbne-beamsim/repository
```

The Migration

Basic git

Build Issues

Summary

The parts of git

CVS has just a working directory and a remote repository, git has these parts:

working directory holds the files you edit, compile, etc.

index a staging area maintained by git for preparing commits

repository the git file system holding commits, branches and tags

- a *local* repository is the one associated with your working directory
- a *remote* repository is some other one but otherwise is not special

- Everything, except the working directory, is under `.git/` which is found at top level in the working directory itself.
- One rarely needs to explicitly look in `.git/`
- Your local repository can serve as another's remote repository

git from nothing

Start a new git repository from nothing is easy:

```
$ mkdir somedir
$ cd somedir
$ git init
```

Cloning

To “clone” a git repository means to make a local repository based on some other existing one.

```
$ git clone <URL>
```

anonymous URL clone with no authentication but only one-way copy

```
http://cdcvs.fnal.gov/projects/lbne-beamsim/g4lbne.git
```

authenticated URL clone with Fermilab Kerberos and Services accounts mitigated by Redmine.

```
ssh://p-lbne-beamsim@cdcvs.fnal.gov/cvs/projects/lbne-beamsim/g4lbne.git
```

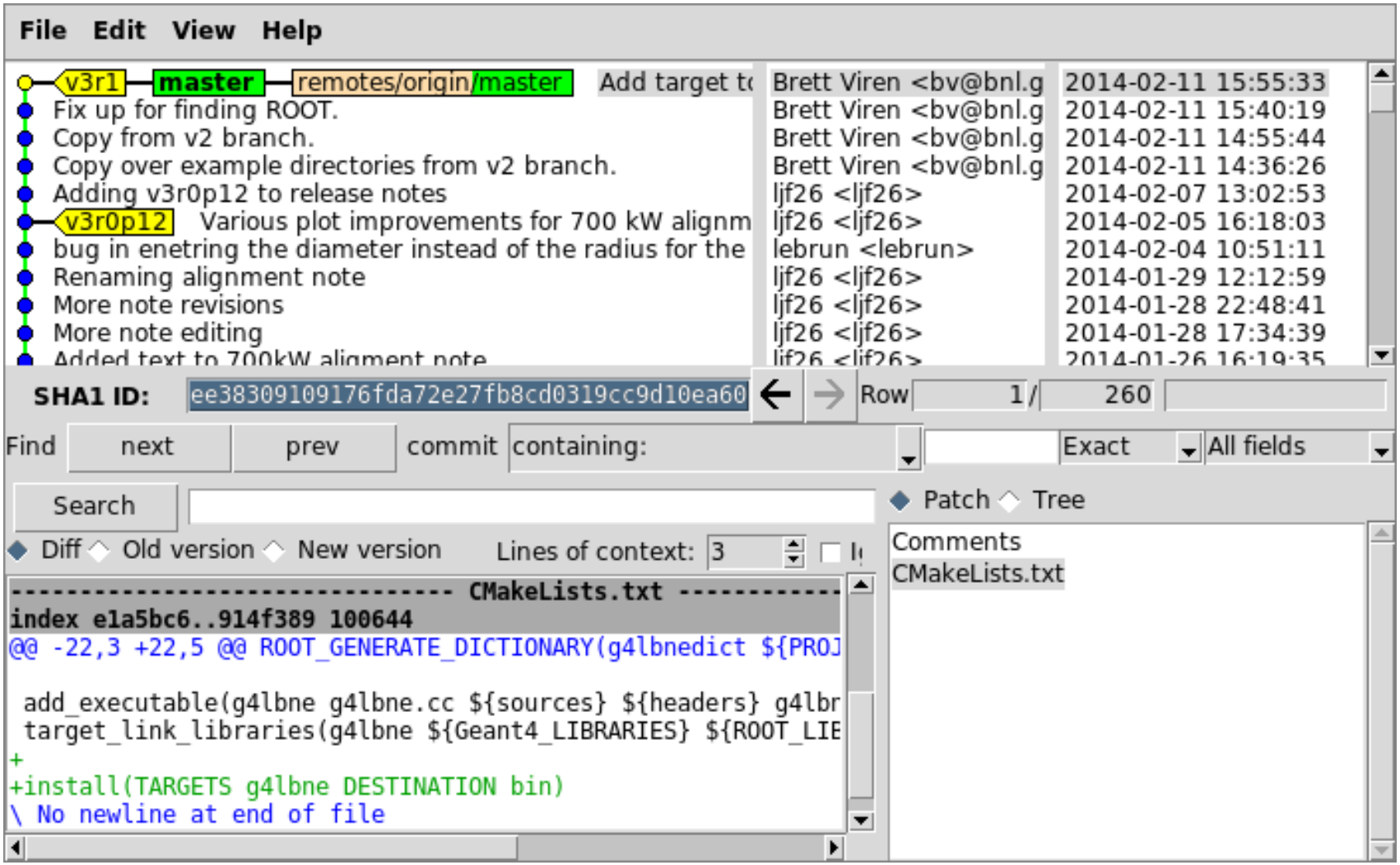
My First Clone

```
$ git clone http://cdcvs.fnal.gov/projects/lbne-beamsim/g4lbne.git
$ cd g4lbne
$ ls -l
total 92
-rw-r--r-- 1 bv bv   676 Feb 12 21:10 CMakeLists.txt
drwxr-xr-x 4 bv bv  4096 Feb 12 21:10 example
-rw-r--r-- 1 bv bv 12097 Feb 12 21:10 g4lbne.cc
...
$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/muonFlux
  remotes/origin/v2
```

origin default name the remote git repository

master conventional name for main branch

Visualizing the repository - gitk



Visualizing the repository - gitk

Use "`gitk -all`" for:

- Seeing recent commits
- Viewing their diffs
- Searching for a particular change
- Understanding which branches and tags are where
- Getting a specific commit hash (SHA1 ID)
- Instead of `-all` may focus on a specific directory or file.

Info from the command line

`git status` state of working directory, lists any working files added/modified/deleted

`git log [<file>]` see recent commit messages, potentially limited to one file

`git diff [<commit>]` see diff between working directory and local repository or some specific commit (various ways to "spell" commit)

`git branch -a` list local and remote branches

`git tag` list what tags have been placed

`git remote -v` list what remote repositories that your local repository knows

Updating your local repository

From time to time you will want to update your local repository with any changes that have been pushed to the remote Redmine `git` repository by others.

`git fetch` this will download all commits from the remote repository but not merge into local branches or working directory.

`git pull` this is a `fetch` plus a `merge` of the remote branch into your local branch and working directory.

- A "`git pull`" is most similar to a "`cvs update`".
- You must have a "clean" `pull` before you may upload any of your own commis
- During the merge of a `pull` a conflict may arise (more later)

Adding new files to git

Before a file can be tracked by git it needs to be added:

```
$ emacs somefile.cc
$ git add somefile.cc
```

This does not yet commit the file (technically it adds it to the git "index"). Rather, it makes git aware of it so that it may be committed in the future.

You may add already added files w/out problem.

Commit your own changes

A commit **always** is a local operation.

```
$ emacs somefile.cc
$ git add somefile.cc
$ git commit -m "A succinct and useful commit message"
# or
$ git commit -m "A succinct and useful commit message" somefile.cc
# or
$ git commit -a -m "A succinct and useful commit message"
```

- The "`add + commit`" will only commit those files added
- The "`commit somefile.cc`" commits only that file
- The "`commit -a`" commits all files modified (and which have been previously added)

Sharing your commits with others

There are many paradigms to share git commits. In our central-repository convention you “push” them to Redmine’s git repository.

```
$ git push origin master
# or just
$ git push
```

- First form explicitly says which remote to use (“origin”) and which branch (“master”)
- If the clone was made as described before, “`origin master`” are the default and can be omitted.
- You must use an authenticated clone to push.
  - If you started with an anonymous clone you can add an authenticated remote.

Adding more remotes

Say you started with an anonymous clone, then you made some changes and did some (local) commits and want to share them.

```
$ git remote add redmine \
  ssh://p-lbne-beamsim@cdcvs.fnal.gov/cvs/projects/lbne-beamsim/g4lbne.gi
$ git pull redmine master
$ git push redmine master
```

Pick any name you like instead of "redmine".

Conflicts

What are conflicts:

- `git` does a good job of merging commits to the same file made by multiple people.
- but, if they overlap the same part of code `git` can not figure out who's commit should "win"
  - this produces a "conflict"
- a conflict is reported during the merge part of a `git pull`

Resolving conflicts:

- Edit the conflicting file(s) and find regions delimited by "`<<< / === / >>>`"
- Pick your version, the remote version or some mix of the two
- add and commit the result.

Note: this is very similar to how conflicts were dealt with in CVS.

Where to go for more

There are boat loads of documentation on git!

Pro Git	http://git-scm.com/book/
online help	All commands are self documenting

```
$ git help
$ git help commit
```

Since git is so distributed we do not have to get hung up on where remote repositories live or even how many any particular project has. Some fantastic free hosting services:

FNAL Redmine	https://cdcvs.fnal.gov/redmine/
GitHub	https://github.com/
Gitorious	https://gitorious.org/
Gitolite	http://gitolite.com/ (DIY)

Some Nifty Extras

shell completion `git` comes with some setup for <TAB> completion. Use it, it makes working with git commands must easier. `https://github.com/git/git/tree/master/contrib/completion`

shell prompt you can also modify your prompt to indicate some simple `git` state such as active branch. For bash I add "`$(__git_ps1 "(%s)")`"

git flow a clone/branch/push/pull/merge methodology to well organize development. It can be enacted by plain git but there are additional `git` commands to assist. These can be included in figure g4lbne installations.

The Migration

Basic `git`

Build Issues

Summary

Note: these are largely independent from the migration to `git`.

CMake-based build of g4lbne

- CMake makes Makefiles and handles finding dependencies and platform vagaries.
- ROOT, Geant4, Art and others use it to build
- I added a CMake-based build to g4lbne v2 and am moving it to v3 (master)
  - Some issues still need to work out

Build outline:

```
$ git clone <URL>
$ mkdir build
$ cd build
$ cmake -DCMAKE_INSTALL_PREFIX=../install ../g4lbne
$ make && make install
```

A worch-based build for g4lbne

- Worch is a meta-build system that LBNE is using to build all our software.
- Initially targeting Larsoft+art (+ the gazillion externals)
- I’ve started work on a worch-based build for g4lbne
  - Just ironing out the CMake g4lbne build

Some benefits:

- integration with the rest of LBNE software
- versioning of the entire suite (g4lbne+Geant4+ROOT)
- automated installation on many platforms
- management of user environment

User Environment

Initially based on Environment Modules (EM) (`http://modules.sf.net`)

- EM is common, Free Software, used at other HEP institutions
- Worch has built-in support
- Light weight, flexible, allows layered releases
- Supports patterns for development environment

May eventually move to UPS

- Basis for Larsoft setup, so one less thing to learn
- Encumbers the installation process, but this is improving
- Provides development environment via Fermilab's "`mrb`" script

Stay tuned for developments on this front.

The Migration

Basic `git`

Build Issues

Summary

Summary

- Migration of G4LBNE from CVS is done and ready to use.
- Intro to git given here but lots of help on git available.
- Progress on installation and integration with the rest of LBNE software.
- Some work needed to decide on user and developer environment management.