Interoperability

SVN

Clone one part (branch, folder, tree) of a Subversion repository git svn clone URL

Clone an entire SVN repo with tags, trunks, and branches git svn clone —stdlayout URL git svn clone —stdlayout http://svn.apache.org/repos/asf/commons/proper/logging/ git svn clone —stdlayout http://ambientideas.unfuddle.com/svn/ambientideas_sample11-svnscm/

or just the trunk git svn clone http://ambientideas.unfuddle.com/svn/ambientideas_sample11-svnscm/trunk

Get new commits from Subversion git svn rebase

Push Git commits back to Subversion git svn dcommit

ClearCase

- Conversion Tools
- CMBridge
  - Commercial tool
  - Round-trip conversion
  - SVN and Git support
- A workflow that uses ClearCase for checkouts and Git for desktop version control
  - Merely a manual way of working with the two systems
  - Puts strain on the developer to maintain state in Git and push back to CC as needed
  - Can require locking
- Another workflow that slightly improves on the previous one
  - Still no real automation
  - Burdens the developer
  - But doesn't require the approval, purchase or download of any bridging tools
- git-cc scripts
  - Provides `gitcc` script.

  - Offers convenience "bridge" functions to put changes back into CC, treating it as the source-of-record.
  - Similar to the git-svn bridge that is part of the official Git distribution

- git-clearcase bridge
  - Readme
  - Perl based
  - Only works with one view
  - No history import
  - `gitclearcase initcc` to start
  - `gitclearcase pullcc` to push changes back in (checkout, merge, checkin to CC)
  - If conflicts, resolve, then `gitclearcase commitcc`

- Native Git conversion
  - Write your own scripts
  - git fast-import

Fast-Import

Git `fast-import` is a human-readable transport format for recreating repositories. Ben Lynn has written a quick description of fast-import that is helpful to read.

The standard use of fast-import is a one time import of source code history from another version control system. Using sed, awk, grep and similar tools, developers can craft an output to almost any version control system that matches this fast-import file format.

Example export:

```
blob
mark :1
data 13
Testing
JUNK

blob
mark :2
data 14
Testing2
```

```
JUNK

commit refs/heads/master
mark :3
author Matthew McCullough <matthewm@ambientideas.com> 1286663588 -0600
committer Matthew McCullough <matthewm@ambientideas.com> 1286663588 -0600
data 11
More stuff
M 100644 :1 test1.txt
M 100644 :2 test2.txt

blob
mark :4
data 22
Testing
JUNK
MOREJUNK

blob
mark :5
data 23
Testing2
JUNK
MOREJUNK

commit refs/heads/master
mark :6
author Matthew McCullough <matthewm@ambientideas.com> 1286664669 -0600
committer Matthew McCullough <matthewm@ambientideas.com> 1286664669 -0600
data 15
Another commit
from :3
M 100644 :4 test1.txt
M 100644 :5 test2.txt
```

Create a Git repository from this formatted input file:

```
mkdir importedproject
cd importedproject
git init
git fast-import --date-format=rfc2822 < /myimportfile
```

and then switch to the master branch after the import with:

```
git checkout master .
```

There is an equivalent `git fast-export` command that outputs an already-Git repository to this plaintext format. It can be used to study the format and learn how to craft your own import files.

```
git fast-export HEAD^^..HEAD
```

You'll likely want to redirect this output to a file.