Main Page

From Git SCM Wiki

Git Wiki Homepage

Welcome to the Git wiki web site. This wiki is a community effort to provide an accurate source of information for all things related to Git. Make sure to visit the Git website (http://git-scm.com) (http://git-scm.com) for current updates and information. The Git web site source (http://github.com/schacon/gitscm) is also managed using Git. The maintainers of the Git website would love patches for improving the website. It's a great way to try out Git. If you come across anything in this wiki or the Git homepage that needs to be fixed; help yourself and fix it. The more people who help work on this wiki the better this site will serve Git users. Think of this site as a Wikipedia for Git.

Starting points

- Installation
- Git Tools – *Note: Have you written one? Add it here and announce it on the list!*
- Git Documentation
- Git Community Support
- Git Rev News (https://git.github.io/rev_news/)
- Projects using Git
- FAQ
- Documentation/SubmittingPatches (http://git.kernel.org/?p=git/git.git;a=blob;f=Documentation/SubmittingPatches;hb=refs/heads/master)
- Git Hosting
- Links to Git-related websites
- Git Compared
- Summer of Code 2011 projects
- Summer of Code 2012 ideas (https://github.com/peff/git/wiki/SoC-2012-Ideas)
- Git on Windows

News

- October 21, 2016: Git User's Survey 2016 has been closed. GitSurvey2016.
- October 12, 2016: Git User's Survey 2016 has been opened.

Older news

Good starting places for developing Git would be

- Janitor
- ToDo
- Wishlist

Using this Wiki

- UsingWiki

Disambiguation

Git is not to be confused with at least these three unrelated projects which also provide commands named git. These should be uninstalled (or renamed) to use this git.

- gnuit (GNU Interactive Tools) (http://hulubei.net/tudor/git/) is a set of file browsing and viewing tools (a text-based file manager).
- Git (Guitar/Instrument Tuner) (http://sourceforge.net/projects/git) is a defunct portable guitar tuner.
- Grit (GBA Raster Image Transmogrifier) (http://www.coranac.com/projects/grit/) is a tool to convert images to the Game Boy Advance hardware. It used to be called "git", but have since changed name.

Installation

From Git SCM Wiki

From sources

If you already have Git installed (many new Linux distributions come with Git), you can get the latest development version via Git itself by using the command:

```
git clone git://git.kernel.org/pub/scm/git/git.git
```

- If you have problems connecting (Git uses port 9418), you can try to access the repository over the HTTP protocol (this method is considerably slower but works even behind firewalls and such):

```
git clone http://www.kernel.org/pub/scm/git/git.git
```

The *main* place to get Git is at http://www.kernel.org/pub/software/scm/git/. You can also use one of many kernel.org mirrors (http://www.kernel.org/mirrors/) .

The latest install directions are included with Git under INSTALL and the most current version online is here (http://git.kernel.org/?p=git/git.git;a=blob_plain;f=INSTALL;hb=master) .

Windows Systems

The most convenient choice is Git for Windows (https://git-for-windows.github.io/) .

Others

- EclipseIDE-based GIT (http://www.eclipse.org/egit/) client, based on a pure Java implementation (http://www.eclipse.org/jgit) of GIT's internals.
- Windows Explorer extension (Git-Cheetah)
- Git Extensions for Windows and Visual Studio (http://sourceforge.net/projects/gitextensions/)
- Tortoise Git (http://code.google.com/p/tortoisegit/) is installed as an extension to the Windows Explorer

OS X

Git for the OS X is installed by using the git-osx-installer (http://code.google.com/p/git-osx-installer/downloads/list) but if you haven't got Leopard you're screwed because there aren't any install scripts and the source won't compile.

Alternatively, you can install Git via MacPorts (http://www.macports.org/) or Fink (http://www.finkproject.org/) . Both work with Tiger, but may pull in a couple of dependencies. With MacPorts, be sure to check out possible extra features with `port variants git-core`; for example, to install Git with bash completion use `sudo port install git-core +bash_completion`.

Another option for Git installation for users running OS X 10.5 or higher on an Intel machine is the Homebrew (http://mxcl.github.com/homebrew) package manager. Installation is as simple as `brew install git`.

Ubuntu / Debian

Install precompiled:

```
sudo apt update
sudo apt install git
```

- How To Install Git on Debian 10 (https://www.digitalocean.com/community/tutorials/how-to-install-git-on-debian-10)

Getting and installing documentation

You can find all the plain text documentation in the Git source tree's Documentation/ (http://www.kernel.org/git/?p=git/git.git;a=tree;f=Documentation) directory.

In order to build the HTML version of the documentation you need to have AsciiDoc version 7.0 or greater installed. Man pages also require that xmlto is installed.

To build and install documentation from the Git source code simply run:

```
$ make install-doc
```

If you want to avoid having to install the documentation tools, autogenerated documentation is available in separate tarballs from:

- http://www.kernel.org/pub/software/scm/git/

beginning with the 1.4.0 version, as:

- http://www.kernel.org/pub/software/scm/git/git-htmldocs-1.4.0.tar.gz
- http://www.kernel.org/pub/software/scm/git/git-manpages-1.4.0.tar.gz

Also, many distributions bundle pregenerated documentation with either the git-core package or in a separate git-doc package.

If you already track development in the public Git project repository, you may also choose to fetch the autogenerated HTML and man pages from the `html` and `man` branches. Read INSTALL (http://www.kernel.org/git/?p=git/git.git;a=blob;hb=HEAD;f=INSTALL) for more detailed documentation (at the end of the file). Once you have fetched the branches, you can use:

```
$ git tar-tree man > git-man.tar
$ git tar-tree html > git-html.tar
```

to easily extract the documentation.

For people on OS X (and on distributions that do not manage XML catalog files automatically), write-up by Steven Grimm may be helpful: http://article.gmane.org/gmane.comp.version-control.git/35565

Interfaces, frontends, and tools

From Git SCM Wiki

Contents

        - 2.4.3.15 git-svnsync
    - 2.4.4 Mercurial
        - 2.4.4.1 hg-to-git
        - 2.4.4.2 hg-fast-export
        - 2.4.4.3 git-hg
        - 2.4.4.4 git-remote-hg
        - 2.4.4.5 git-cinnabar
    - 2.4.5 Darcs
        - 2.4.5.1 darcs-to-git
        - 2.4.5.2 Darcs-Git
        - 2.4.5.3 darcs-fast-export
        - 2.4.5.4 darcs2git
    - 2.4.6 Bazaar
        - 2.4.6.1 Git-bzr
        - 2.4.6.2 bzr-fast-export
        - 2.4.6.3 BzrToGit
        - 2.4.6.4 git-remote-bzr
    - 2.4.7 Perforce
        - 2.4.7.1 git-p4 (built in)
        - 2.4.7.2 git-p4import
        - 2.4.7.3 git-p4raw
    - 2.4.8 Monotone
        - 2.4.8.1 mtn2git
    - 2.4.9 ClearCase
        - 2.4.9.1 git-cc
        - 2.4.9.2 CC2GIT ClearCase GIT Bridge (Clearvision)
        - 2.4.9.3 git-ucmimport (IBM Rational ClearCase)
    - 2.4.10 Others
        - 2.4.10.1 quilt2git / git2quilt
        - 2.4.10.2 SCCS import
        - 2.4.10.3 rcs-fast-export
        - 2.4.10.4 rcs-fast-import
        - 2.4.10.5 rpm2git
        - 2.4.10.6 vss2git (Visual Source Safe to Git)
        - 2.4.10.7 vss2git (Visual SourceSafe to GIT or SVN)
        - 2.4.10.8 tfs2git (Team Foundation Server to Git)
        - 2.4.10.9 git-tfs (Team Foundation Server bridge)
        - 2.4.10.10 SRC
        - 2.4.10.11 Git-Mediawiki
        - 2.4.10.12 ldap-git-backup
        - 2.4.10.13 BitKeeper
- 2.5 Hooks
    - 2.5.1 Better Commit Policy
    - 2.5.2 git-notify
    - 2.5.3 git-commit-notifier
    - 2.5.4 git-multimail
    - 2.5.5 archive-tag
    - 2.5.6 git-blacklist
    - 2.5.7 Hooks for integration with IBM Rational Team Concert
    - 2.5.8 CIAbot
    - 2.5.9 irker
    - 2.5.10 Gitspread
- 2.6 Wikis, blogs, etc.
    - 2.6.1 ikiwiki
    - 2.6.2 Gollum
    - 2.6.3 wikiri
    - 2.6.4 git-wiki
    - 2.6.5 WiGit
    - 2.6.6 Nuki

Frontends and Interfaces

Version Control Interface layers

git-wizard

git wizard (https://github.com/makelinux/git-wizard) - handy wrapper script

git-m

git-m (https://github.com/makelinux/gitm/) is multiple git replication and management utility. It allows in one line command replicate tree of git repositories to another host.

Zit

Zit (http://git.oblomov.eu/zit) by Giuseppe Bilotta is the Git-based single file content tracker; it uses Git to independently track single files within a directory; sort of like what RCS does, but with the power, flexibility, elegance and ease of use of Git. Still in alpha stage.

Gitless

Gitless (http://gitless.com/) is an experiment to see what happens if you put a simple veneer on Git, and change its underlying concepts.

Historical

- Easy Git (eg) (http://www.gnome.org/~newren/eg/) Easy Git is a wrapper for Git that is designed to make Git easy to learn and use. Easy Git aims to be easily learnable, interchangeable with Git, fully capable, and compatible with Git. Users can move back and forth between using Easy Git and Git itself.
- vng (http://repo.or.cz/w/vng.git) is a Darcs-like Git porcelain by Thomas Zander.
- (Archived since 2018.) fgit (https://github.com/l0b0/fgit) (Folder Git) runs a Git command in several repositories. Use to garbage collect, check status, pull or anything else on multiple repositories in a single command.
- yap (Yet Another Git Porcelain) (http://repo.or.cz/w/yap.git) Yap is an alternative porcelain for Git that is designed to have a friendlier, more orthogonal interface. It is also easily extensible with Python plugins.

Interfaces to other programming languages

C

- libgit2 (https://libgit2.github.com/) is a project to replace the core parts of Git with an implementation reusable for language bindings and higher-level applications.
- libgit-thin (http://repo.or.cz/w/git/libgit-gsoc.git) is an older attempt in the same vein which started as a Google Summer of Code 2007 project (http://code.google.com/p/google-summer-of-code-2007-git/) . Read this announcement (http://article.gmane.org/gmane.comp.version-control.git/53433) for more details. It also comes with Python bindings called PyGit, which demonstrate the current capabilities.

Perl

- Git::CPAN::Patch (http://search.cpan.org/~yanick/Git-CPAN-Patch) provides a suite of Git commands aimed at making trivially easy the process of grabbing any distribution off CPAN, stuffing it in a local Git repository and, once gleeful hacking has been perpetrated, sending back patches to its maintainer.
- Git::PurePerl (http://p3rl.org/Git::PurePerl) is pure Perl interface to Git repositories. It was mostly based on Grit (http://grit.rubyforge.org/) (described below).
- Git::Wrapper (http://search.cpan.org/dist/Git-Wrapper) is a Perl interface to the Git executables. It provides an API that uses Perl data structures for argument passing, instead of CLI-style --options as Git.pm does.

Python

- Dulwich (http://samba.org/~jelmer/dulwich/) is a pure-Python read-write implementation of the Git file formats and protocols. It is named after the village in which Mr. and Mrs. Git live in the Monty Python sketch. A Git repository can be found at git://git.samba.org/jelmer/dulwich.git and its gitweb here (http://git.samba.org/?p=jelmer/dulwich.git;a=summary) .

- Pygit2 (http://github.com/libgit2/pygit2) is a set of Python bindings to libgit2 (see above).
- GitPython (https://github.com/gitpython-developers/GitPython) is a Python module that provides object model access to your Git repositories by calling the Git executables and parsing output. Once you have created a repository object, you can traverse it to find parent commit(s), trees, blobs, etc.. Port of the grit (http://grit.rubyforge.org/) library in Ruby created by Tom Preston-Werner and Chris Wanstrath.
- vcs (http://bitbucket.org/marcinkuzminski/vcs/) is Various Control System abstraction layer written in Python. Uses mix of already mentioned Dulwich and Git binary for the Git backend. Provides simple yet fully featured interface and is under heavy development stage.

Ruby

- Grit (http://grit.rubyforge.org/) is a Ruby library for extracting information from a Git repository in an object oriented manner. This includes a partial native Ruby implementation (http://github.com/blog/107-supercharged-ruby-git) , which is used for a number of the operations.
- GitStore (http://github.com/georgi/git_store) implements a versioned data store based on the revision management system Git. You can store object hierarchies as nested hashes, which will be mapped on the directory structure of a Git repository. Basically GitStore checks out the repository into a in-memory representation, which can be modified and finally committed. GitStore reads and writes the Git repository natively in Ruby.
- GitRb (http://github.com/minad/gitrb) is another partial native implementation of Git in Ruby similar to grit.
- v (http://github.com/boof/v) is another Ruby library providing a Git adapter that makes working with Git repositories easy. Git operations are mapped almost 1:1 to ruby methods and executed in a separate thread returning futures when they're queued. It supports core Git objects and convenience objects (indexes, branches, ...) which will provide a convenient API. It is still under development, means except for few examples there is no real documentation and most of the Git operations still need to be implemented (first CommandLine wrapper (porcelain and plumbing), then pure Ruby (plumbing and then porcelain)).
- Ruby/Git (http://jointheconversation.org/rubygit/) (deprecated) is a Ruby library that can be used to create, read and manipulate Git repositories. Unmaintained, abandoned in favor of mentioned above Grit (http://grit.rubyforge.org/) .

Objective-C

- ObjectiveGit (http://github.com/schacon/objective-git) is an Objective-C partial implementation of Git and a corresponding object level library that can be used to create, read and manipulate Git repositories. It also contains Git server implementations (receive-pack and upload-pack).

Java

- JGit (http://www.eclipse.org/jgit/) is a BSD licensed, pure Java implementation of the Git file format, core algorithms, and network transport protocols. It can be found embedded inside of many Git related applications, including EGit (http://www.eclipse.org/egit/) , Gerrit Code Review (http://code.google.com/p/gerrit) , NBGit (http://nbgit.org/) , and JetBrains TeamCity (http://www.jetbrains.com/teamcity/) .
- JavaGit (http://javagit.sourceforge.net) is an API providing access to Git repositories for Java applications. JavaGit is engineered to provide the developer with access to the raw Git commands through a command API as well as an object API designed to represent the .git repository, the working tree and other, familiar Git concepts. JavaGit uses the Git binaries installed on the host machine to provide Git functionality and has been designed to easily accommodate additional methods of access to Git repositories. JavaGit is released as open source software under the GNU LGPL license.
  Currently in early alpha version

See also: NYU Open Source Programming Class Releases JavaGit API 0.1.0 Alpha (http://thread.gmane.org/gmane.comp.version-control.git/91381) thread on the Git mailing list.

PHP

- glip (http://fimml.at/glip) , the *Git library in PHP*, provides read and write access to Git repositories without using exec() or system() calls. Therefore it works on many shared web hosting accounts.

.NET and Mono

- dotGit (http://github.com/pheew/dotgit/tree/master) (no activity since mid-2009) is an experimental implementation for .NET written in C#. It is still in it's early stages but can read objects from packed and loose storage. So all the read-only operations are supported. A diff engine is in the works.
- Git# (http://www.eqqon.com/index.php/GitSharp) (no activity since late 2009) for .NET and Mono is written in C#. GitSharp provides a user friendly well documented API (http://henon.github.com/GitSharp/) . It is aimed to be fully compatible to the original Git and can be either used as stand alone command line application or as library for gui frontends, IDE plugins, etc. The status of the project is beta.
- NGit (http://github.com/slluis/ngit) Automated jgit port to C#. The MonoDevelop team really wants to provide integrated support for Git in the IDE. GitSharp was a little bit outdated so it was a challenge to bring it up-to-date to match JGit. So they finally decided to use Sharpen, our open source Java to C# translator, to automatically convert the JGit code base to C# (details) (http://developer.db4o.com/Blogs/Projects/tabid/168/entryid/965/Default.aspx)

Haskell

- hlibgit2 (http://github.com/jwiegley/gitlib/) is a complete, FFI level wrapper around libgit2.
- gitlib (http://github.com/jwiegley/gitlib/) is an abstraction of the Git "model", allowing the use of many different backends, including forking command-line Git and hlibgit2.
- hs-libgit (http://github.com/vincenthz/hs-libgit/) provides a lowlevel interface in haskell using Git core through fork/exec. The goal is to make use of libgit2 as soon as it's usable.
- hit (http://github.com/vincenthz/hit) is a read-write haskell reimplementation of Git's storage format. Completely inter-operable with the official Git tools. It doesn't use Git's CLI and provides some basic API to manipulate a Git store (reading from individual or packed objects, and writing).

JavaScript

- git.js (http://github.com/danlucraft/git.js) is a Git implementation written in pure JavaScript. Git.js is MIT licensed. It features both a command-line Git client for node.js (http://nodejs.org/) and an in-browser repository API for accessing repos available via HTTP.

Patch-management Interface layers

StGIT (Stacked Git)

- StGIT (homepage (http://procode.org/stgit/) , tutorial (http://procode.org/stgit/doc/tutorial.html) , gitweb (http://repo.or.cz/w/stgit.git) , GitHub (https://github.com/ctmarinas/stgit)) provides a Quilt-like patch management functionality (i.e. pushing/popping patches to/from a stack) in the Git environment. You can easily manage your patches in the scope of Git until they get merged upstream.
  See also: Quilt project page (http://savannah.nongnu.org/projects/quilt/) .

Guilt (formerly Git Queues (gq))

- guilt (download (http://guilt.31bits.net/) , howto (http://repo.or.cz/guilt.git/blob/HEAD:/Documentation/HOWTO) , manpage (http://guilt.31bits.net/man/guilt.html) , gitweb (http://repo.or.cz/w/guilt.git) , GitHub

(https://github.com/jeffpc/guilt)) by Josef "Jeff" Sipek is a series of Bash scripts which add a Mercurial queues (http://www.selenic.com/mercurial/wiki/index.cgi/MqExtension) -like functionality and interface to Git. The one distinguishing feature from other Quilt-like porcelains, is the format of the patches directory. All the information is stored as plain text - a series file and the patches (one per file). This easily lends itself to versioning the patches using any number of of SCMs.

cj-git-patchtool

- cj-git-patchtool (GitHub (https://github.com/pflanze/cj-git-patchtool)) by Christian Jaeger is a tool to edit the history of a Git repository. It does this by turning the history in question into a set of Git patch files and a file containing the list of the patches.

TopGit

- TopGit (GitHub (https://github.com/greenrd/topgit)) is a patch management interface done as a set of shell scripts, which can be used to manage a set of third-party patches on top of another project. It is a very thin layer on top of Git (allowing you to use the index); it attempts to keep all history of your changes until final cleanup; and it allows you to specify patch dependencies instead of linearizing patches in a patch series (patch queue).
  See also: TopGit - A different patch queue manager (http://permalink.gmane.org/gmane.comp.version-control.git/91197) announcement on the Git mailing list.

Patchy Git (pg), deprecated

- pg (homepage (http://www.spearce.org/category/projects/scm/pg/)) by Shawn Pearce aims to help the user manage a set of patches on top of the current branch. pg is somewhat like Quilt or StGIT, but it does have a slightly different feature set. *Note that pg is no longer being actively developed.*

Graphical Interfaces

Graphical Interfaces - Proprietary

GitKraken

- GitKraken (https://www.gitkraken.com/) is a popular Git GUI client for Windows, Mac and Linux. It's free for non-commercial use. It's a great tool for Git beginners and advanced users to increase efficiency through the intuitive interface, seamless integrations and a faster, more fluid workflow.

gitSafe

- gitSafe (homepage (http://www.kodespace.com/gitSafe/)) by cmroanirgo is a SourceSafe-like GUI for Git using MFC. *NOTE:* This software is Closed Source.

SmartGit

- SmartGit (http://www.syntevo.com/smartgit) by Syntevo GmbH is a graphical Git client which runs on all major platforms (e.g. Linux, Mac OS X, Microsoft Windows). It requires (SUN-) Java 1.5 or newer as well as a Git installation. Closed source, free of charge for non-commercial usage.

SourceTree

- SourceTree (http://www.sourcetreeapp.com) is a fast and friendly GUI for Git and Mercurial for Mac OS X 10.6+ and Windows 7+ (closed source, commercial). A trial version is available.

Agit (Android)

- Agit (http://market.android.com/details?id=com.madgag.agit) is Git client for Android with features like pull-to-refresh fetching, periodic sync, and gently animated diff transitions. It's the first ever Git client targeted at mobile phone, and currently is a 'read-only' client, supporting clone & fetch, but not commit and push.

Working Copy (iOS)

- Working Copy (https://workingcopyapp.com/) is a full-featured Git client for iOS. It supports cloning, fetching, committing, pushing and merging. Using iOS app extensions it provides read/write access to repositories from other applications.

Tower

- Tower (https://www.git-tower.com) by Fournova is a graphical Git client for Mac OS X 10.8+ and Windows 7+ (closed source, commercial). A trial version is available.

GitVine

- [1] (https://insanesharpness.gitlab.io/GitVine) by Insane Sharpness, is a Graphical Git Client for Windows10, Linux & macOS. Inspired by Clearcase Version Tree. Completely Free. Supports both Viewing and Modifications. Also has an integrated terminal for full Git control.

Graphical Interfaces - FLOSS

Git Watcher

- Git Watcher [2] (https://github.com/demian85/git-watcher) is a multi-platform desktop app that shows real-time diff file information for working directory and index and allows you to commit/push changes. It also organizes submodules in tabs and has many other options and tools. Currently in development!

Cocoon

- Cocoon [3] (https://projects.kde.org/projects/playground/sdk/cocoon) is a KDE 4 playground project to create a GUI for Git.

gitk (distributed with Git)

- gitk (https://git-scm.com/docs/gitk) is a simple Tcl/Tk GUI for browsing history of Git repositories easily, distributed with Git.

git-gui (distributed with Git)

- git-gui (https://git-scm.com/docs/git-gui) by Shawn Pearce is a tool for creating commits and managing branches. It was inspired by and initially based on gitool. Written in Tcl/Tk. Stable versions are shipped with Core Git since version 1.5.

tig (Text-mode Interface for Git)

- tig (homepage (http://jonas.nitro.dk/tig/)) by Jonas Fonseca is a simple Git repository browser written using ncurses. Basically, it just acts as a front-end for `git-log` and `git-show`/`git-diff`. Additionally, you can also use it as a pager for Git commands.

QGit

- QGit (homepage (http://libre.tibirna.org/projects/qgit/wiki/QGit)) is a Qt GUI for browsing the history of Git repositories. It can also act as a commit tool, a file annotation and modification history browser, and as

a graphical interface to StGIT. The sources for the latest, 2.5 version are available at http://repo.or.cz/w/qgit4/redivivus.git. This version compiles with Qt4 and is directly portable to Linux, Windows and MacOS.

CodeReview

- CodeReview (homepage (https://github.com/FabriceSalvaire/CodeReview)) is a Qt GUI written in Python for browsing the history of Git repositories and show a side-by-side diff of the changes similar to kdiff (http://kdiff3.sourceforge.net) . It is a clone of the QBzr (http://wiki.bazaar.canonical.com/QBzr) diff tool. It is thus a local code review tool and not a complete GUI for git. It should work on Linux, Windows and MacOS.

Giggle

- Giggle (homepage (http://live.gnome.org/giggle)) is a graphical frontend for browsing history of Git repositories (think of gitk on GTK+). Result of a Hackathon 2007, later developed by Imendio's Carlos Garnacho and Sven Herzberg and then by Mathias Hasselmann. Currently maintained by the GNOME community. Available from http://git.gnome.org/browse/giggle/

gitview (in `contrib/`)

- GitView is a GTK based repository browser for Git written in Python by Aneesh Kumar K.V. It can be found in `contrib/` directory of Git source tree.

git-forest

- *git-forest* (git-forest (http://dev.medozas.de/gitweb.cgi?p=hxtools;a=blob_plain;f=sdevel/git-forest) file and manpage (http://dev.medozas.de/gitweb.cgi?p=hxtools;a=blob_plain;f=doc/git-forest.1)) is a text-based tree visualizer, using Unicode characters for tree graphics. Part of its key features are: display of tags, heads and other refs (like gitk), constantly vertical branch graphics (allows to follow a branch with the mouse easily), display of octopus merges. It can also display history in reverse.

A screenshot may be found at git-forest.png (http://jengelh.medozas.de/images/git-forest.png) . Written in Perl by Jan Engelhardt.
See also: text-based tree visualizer (http://permalink.gmane.org/gmane.comp.version-control.git/76404) announcement on the Git mailing list.

GitForce

- **GitForce** (homepage (http://gdevic.github.com/GitForce) , GitHub (https://github.com/gdevic/GitForce)) by Goran Devic is a Git frontend GUI for both Windows and Linux. It is in active development.

Qct

- Qct (http://qct.sourceforge.net/) (Mercurial repository (http://bitbucket.org/sborho/qct/overview/)) by Steve Borho is Qt/PyQt based GUI commit tool, meant to be SCM and platform agnostic. Supports Mercurial (`hg`), Bazaar (`bzr`), Cogito (but not Git directly atm), Subversion (`svn`), Monotone (`mnt`) and CVS. The README (http://hg.borho.org/qct/rev/5c5bdab35c08) mentions gct (also on this page), as an alternative supporting plain Git. License: uknown/ no license.

git-cola

- git-cola (http://cola.tuxfamily.org/) (GitHub (http://github.com/davvid/git-cola/tree/master) , homepage (http://cola.tuxfamily.org/) , gitweb (http://repo.or.cz/w/git-cola.git)) by David Aguilar is an advanced Git commit tool, similar to git-gui, written in PyQt4. git-cola features a graphical 2D history viewer, easy, interactive partial-diff staging, inotify support, and more. You can get tarballs at http://cola.tuxfamily.org/.

Native packages exist for Debian, Fedora, and Arch. Mac OSX and Windows builds are available on the website.

GitX

- GitX (homepage (http://gitx.laullon.com/) , GitHub (https://github.com/laullon/gitx) , download (https://github.com/downloads/laullon/gitx/GitX-L_v0.8.4.zip)) by Pieter de Bie, Germán Laullón and others is a gitk clone aiming to provide a more native interface to MacOS X users with OS X-only features. It has a native interface and tries to integrate with the operating system as good as possible. Examples of this are drag and drop support and QuickLook support.
  See also: GitX v0.1: Gitk clone for OS X (http://permalink.gmane.org/gmane.comp.version-control.git/93980%7C%7CAnnounce) post on the Git mailing list.

Git Extensions

- *Git Extensions* (GitHub (http://github.com/gitextensions/gitextensions/)) is a toolkit for Windows users. The main feature is a shell extension that enables a context menu in Windows Exlorer to use most Git functions. The toolkit also contains a standalone GUI and a Visual Studio 2005/2008 plugin. The toolkit is mainly written in C#, the shell extensions code is written in C++. The toolkit is still under development, there is a beta version downloadable here: http://code.google.com/p/gitextensions/
  See also: Announcement: Git Extensions stable (windows shell extensions) (http://permalink.gmane.org/gmane.comp.version-control.git/103371) on the Git mailing list.

TortoiseGit

- TortoiseGit (http://code.google.com/p/tortoisegit/) (gitweb (http://repo.or.cz/w/TortoiseGit.git)) by Li Frank is a port of TortoiseSVN (http://tortoisesvn.tigris.org/) to Git. It is Microsoft Windows Explorer extension, written in C++. As of March 2017, TortoiseGit is at version 2.4.0.0 and implements a full set of git tasks such as create, browse and clone repositories, pull, commit, push, show log, diff two versions, create branches and tags, create patches etc.

Dreamweaver GIT (GITWeaver Dreamweaver Tortoise Extension)

- GITWeaver (http://github.com/ChrisMcKee/gitweaver) by Chris McKee is an Adobe Dreamweaver (MX->CS5) extension that adds TortoiseGIT functionality to Adobe Dreamweaver. Windows Compatible. GPL/MIT license.

git-cheetah

- git-cheetah (http://git.wiki.kernel.org/index.php/MSysGit:GitCheetah) is a cross platform filemanager plugin. It was started as a Windows explorer extension so the standard gui tools (namely Git Gui, Gitk History, Git Gui Blame) of Git would be available from the filemanager. It has been ported to Mac OS X (Finder before 10.6) and Gnome Nautilus. Written completely in C, it has a simple infrastructure ready to add more plugin implementations. You can find the repository at http://repo.or.cz/w/git-cheetah.git.

gitg

- gitg (gitg (http://git.gnome.org/cgit/gitg/)) by Jesse van den Kieboom is a clone of GitX for gtk+/GNOME. As such it tries to follow the implementation of GitX closely, while providing tight integration into the GNOME desktop.

git-age

- git-age (http://wiki.github.com/krig/git-age) is a `git blame` visualizer, written using PyGTK. Shows the file with information on author, commit etc on each line. Also colors the background of the line darker for older commits and lighter for newer commits. Useful for quickly seeing what parts of a file have changed recently. Will also attempt to retrieve Gravatars for all authors in the file asynchronously.

StupidGit

- StupidGit (http://wiki.github.com/gyim/stupidgit) by Ákos Gyimesi is a cross-platform Git GUI written in wxPython. It tries to be easy to use for beginners and has advanced support for submodules. It is quite early in development, but already implements many features including merge, cherry-pick and easy switching between versions.

PyjamasGitWeb

- PyjamasGitWeb (git clone gitolite@pyjs.org:pyjamasgitweb) is two projects in one: a JSONRPC web service which uses python-git to serve Git repository information, and a matching front-end as a pyjamas application. Pyjamas (http://pyjs.org) can run the front-end either compiled to javascript to run in a web browser or as a Desktop application (pure Python). Cacheing of answers to queries is performed (in the browser or in the desktop app) so that any file or Git link previously clicked on will come up instantly. Demo is at http://pyjs.org/pygit

Summary (feature matrix)

	gitk	git-gui	
written in	Tcl	Tcl	C
UI toolkit	Tk	Tk	c
Open Source	yes	yes	y
last activity	2008	2008	2
tree view	X	ext	X
history viewer	X		X
history search	X		X
(un)stage files		X	X
partial staging		X	
undo diff fragments			
checkout	X	X	X
cherry-pick	X		X
committing		X	X
remote features		X	
stash			
format patch	X		
interfaces w/ external diff viewers			
diff +/- highlighting	X	X	X
compare commits	X		X
merge features		X	
inotify (http://en.wikipedia.org/wiki/Inotify) support			
interactive rebase			
apply patches via drag'n'drop			
i18n	de, es, it, sv	de, fr, hu, it, ja, ru, sv, zh_cn	
Screenshot	X (http://lwn.net/Articles/140350/)	X (http://www.spearce.org/2007/01/git-gui-screenshots.html)	X (v

Deprecated and stalled projects

- (h)gct (gitweb (http://repo.or.cz/w/hgct.git)) by Fredrik Kuivinen is a GUI enabled commit tool. It has support for both Git and Mercurial. In Debian in commit-tool (http://packages.debian.org/search?searchon=names&keywords=commit-tool) package. Written in PyQt. Does not support Git 1.6 and higher.
- gitool (announcement (http://thread.gmane.org/gmane.comp.version-control.git/26415) , download (http://ozlabs.org/~paulus/gitool) , gitweb (http://www.kernel.org/git/?p=gitk/gitk.git;a=shortlog;h=new)) by Paul Mackerras is a tool for creating commits. Superseded by git-gui (see above). Last development activity in 2007. Written in Tcl/Tk.
- GitNub (homepage (http://github.com/Caged/gitnub/wikis) , GitHub (http://github.com/Caged/gitnub/tree/master)) by Justin Palmer is an Mac OSX Leopard client for Git written in Ruby and Objective-C.
- KGit (http://kgit.sourceforge.net/) by Abhijit Bhopatkar is intended to be a small but functional frontend to the popular source control program, Git. It is similar to gitk with two significant differences. First of all, it is written in Qt using lop and the KDE libraries. Secondly, it is not only a repository viewer, unlike gitk. It is now defunct and deprecated by the author. Has no future.
- pmpu (http://pmpu.sharesource.org/) (Push Me Pull You) by Mark Williamson is a graphical interface for a distributed version control system. Currently it contains proof-of-concept supports the Mercurial, Git and bzr systems. PMPU can make use of external history views and commit tools. Preview release. As of November '09, no developer activity for more than 1 year.
- Katana ([4] (http://dekorte.com/projects/shareware/Katana/)) a comprehensive Git UI that focuses on making common operations easy. Visual diffs supported.
- RepoWatch (homepage (http://www.doomstick.com/projects/repowatch/)) is a graphical client for Mac OS X that provides very quick access to the basics of Git or Mercurial with minimal effort: just press Command-Option-Enter.
- *GitJungle* ([5] (http://www.plasticscm.com/labs/gitjungle.aspx)) is a graphical branch explorer displaying branch history horizontally instead of vertically as most Git graphical tools do. It shows merge links, commits, tags and branches and displays diffs when clicking on a specific commit. It runs on Windows, MacOS X and Linux. It is free and requires Mono to run.
- teamGit (http://www.devslashzero.com/teamgit) by Abhijit Bhopatkar is a successor of kgit. It is intended to be a complete development workflow management app with Git as a base tool. As of oct 2008 it can be used as a pretty good commit tool. Please see feature table below.
- pyrite (http://pyrite.sophiasuchtig.com/) by Govind Salinas is a Git GUI and front end written in Python.
- Qt Creator (http://www.qtsoftware.com/developer/qt-creator) is a lightweight, cross-platform integrated development environment (IDE) for developing with Qt that provides a built in GUI for working with Git repositories.
- RabbitVCS (http://www.rabbitvcs.org/) is a set of graphical tools written to provide simple and straightforward access to the version control systems you use. Currently, it is integrated into the Nautilus file manager and supports Git (0.14 Beta 1 (http://blog.rabbitvcs.org/archives/247)) and Subversion, but the goal is to incorporate other version control systems as well as other file managers. Runs on Linux.
- Gource (http://code.google.com/p/gource/) is a software version control visualization tool. Software projects are displayed by Gource as an animated (http://code.google.com/p/gource/wiki/Videos) tree with the root directory of the project at its centre. Directories appear as branches with files as leaves. Developers can be seen working on the tree at the times they contributed to the project.

Web Interfaces - need to move these

Sprout (formerly GitMac) (Mac OS X)

- Sprout (homepage (http://sproutmacapp.com)) Sprout is an App for Mac OS X that focuses on being a fast, user-friendly Git client. For developers who are new to Git, transitioning from Subversion, or just want a nice Mac UI alternative. Focuses on browsing and managing repositories and branches, and making common operations easy.

Web Interfaces

GitLab

- GitLab (https://about.gitlab.com) is a fast, secure and stable solution based on Rails & Gitolite. Free and open-source. Distributed under the MIT License.

gitweb (distributed with Git)

- gitweb provides full-fledged web interface for Git repositories. It is written in Perl and was maintained by Kay Sievers; from Git version 1.4.0 it is distributed with Git. Used at kernel.org (http://www.kernel.org/git/) and repo.or.cz (http://repo.or.cz) . The XMMS2 project (http://wiki.xmms2.xmms.se/index.php/Main_Page) maintains their own version of gitweb which has some additional features. The latest snapshot of this effort can be downloaded from http://git.xmms.se/?p=gitweb-xmms2.git;a=summary. John 'Warthog9' has added in caching to the gitweb, and split it into many modules. See it at work at http://www.kernel.org/git/?p=git/warthog9/gitweb.git;a=summary. Changes are published at `git://git.kernel.org/pub/scm/git/warthog9/gitweb.git`. See also: Gitweb - caching (http://permalink.gmane.org/gmane.comp.version-control.git/35692) message at the Git mailing list.

GitList

- GitList (https://github.com/klaussilveira/gitlist) is an elegant and modern web interface for interacting with multiple Git repositories. It allows you to browse repositories using your favorite browser, viewing files under different revisions, commit history, diffs. It also generates RSS feeds for each repository, allowing you to stay up-to-date with the latest changes anytime, anywhere. GitList was written in PHP, on top of the Silex microframework and powered by the Twig template engine. This means that GitList is easy to install and easy to customize. Also, the GitList gorgeous interface was made possible due to Bootstrap.

P3X Enhanced GitList

- P3X Enhanced GitList (https://github.com/patrikx3/gitlist) is a fork of the klaussilveira Gitlist. What is different about is, that it requires/uses the latest PHP version, works with sub-modules. Update all dependencies with monthly release. With big GIT repos/commits, it works with 64Mb memory (some Twig templates are removed and moved to the client and web workers - eg. huge diffs). You will love it to work it on OpenWrt. Provides multiple themes with dark mode - 11 light and 5 dark. Code editor with syntax highlighting, editable files. All changes in the original fork are synced with the enhanced version. 100% responsive with Bootstrap 3. Latest Fontawesome for icons. The markdown engine uses Emojis with Twitter's Emojis. Besides, the commits and logs are parsed as Markdown and Emojis. The live is at https://gitlist.patrikx3.com/.

klaus

- klaus (https://github.com/jonashaag/klaus) (Demo (http://klausdemo.lophus.org)) is a standalone, zero-configuration-required Git viewer written in Python (WSGI) that ships with a minimalist but elegant Web interface inspired by GitHub. klaus is developed by Jonas Haag.

WebGit .NET

- WebGit .NET (https://github.com/otac0n/WebGitNet) IIS Hosting for Git "Smart HTTP" using ASP.NET MVC. It is written in C# and is hosted on Github.com. Aimed at internal hosting of Git repositories in a

windows environment, it supports syntax highlighting via SHJS and post-create hooks.

Gitorious (discontinued)

- Gitorious (https://en.wikipedia.org/wiki/Gitorious) provides free open source infrastructure for open source projects that use Git. Open Source and Ruby on Rails based. Acquired by #Gitlab in 2015.

InDefero

- InDefero (http://www.indefero.net) is a clone of GoogleCode with Git, Mercurial and Subversion browser, wiki, download area, issue tracking and code review supporting multiple private/public projects. Using a shared user account with SSH keys you can control the read/write access rights to your repositories from within the web interface. GNU GPL/Clean MVC PHP5 code, very fast and easy to extend. Download (http://projects.ceondo.com/p/indefero/downloads/) , view source (http://projects.ceondo.com/p/indefero/source/tree/master/) . Free hosting also available.

Wit

- Wit (http://github.com/dchokola/Wit) is a Ruby/eRuby web interface to Git that provides much easier setup, configuration, maintainability, and customizability than gitweb, all in fewer lines of code and with a prettier interface. It is (barely) maintained by Daniel Chokola.

wit (defunct?)

- *wit* (download (http://www.absolutegiganten.org/wit/)) is a Python implementation maintained by Christian Meder. Uses PATH_INFO URLs extensively. See it in work here (http://www.grmso.net:8090/) .

gitarella

- gitarella (http://flameeyes.is-a-geek.org/projects#gitarella) (freshmeat project page (http://freshmeat.net/projects/gitarella/)) is a Ruby-based Git web frontend, created and maintained by Flameeyes. It is inspired by and following the style of gitweb. It supports CGI and FastCGI interfaces. See it in work on gitarella repository (http://git.flameeyes.is-a-geek.org/gitarella/) .

ginatra

- *ginatra* (GitHub (https://github.com/NARKOZ/ginatra) , demo (http://narkoz.github.io/ginatra/demo)) is web interface written in Ruby using Sinatra (http://www.sinatrarb.com) web framework.

git-php

- git-php (http://code.google.com/p/git-php/) is a PHP Git web frontend created by Zack Bartel. The goal of git-php is a robust PHP web interface to Git repositories. It is meant to be easily customizable through styles and the ability to be embedded into any PHP page. For example, creating a Git repository viewer plug-in for your favorite PHP based CMS should be easy with git-php. It seems to be inspired by gitweb.
- http://people.proekspert.ee/peeter/git/git.php is a PHP Git web frontend created by Peeter Vois. This is continued work of Zack's version of git.php. git-php does now have syntax highlighting of code, graph view of the repository including branch and tag, diff can be browsed against several parent versions, tree browser does have some nice icons, simple search for branch heads and tags, any version of tree can be downloaded in tar.gz or zip format and the filename will include hash of the tree or tag if attached, files can be downloaded independently in binary form. Git php is providing interface for sending bundles to the owner of the site. This is useful for those who would like to stay anonymous or do not like to send bundles via e-mail. The bundle is also tested against the repository before acceptance.
- http://github.com/josegonzalez/git-php is a PHP Git web frontend created by Jose Diaz-Gonzalez. Contains updates to Peeter Vois and Zack Bartel's work that adds a GitHub-like stylesheet, as well as fixes for the latest updates to Git syntax. It is also an attempt to ameliorate the differences between gitweb and all other gitweb derivatives with git-php.

GitPHP

- GitPHP (http://www.xiphux.com/programming/php/gitphp/) (demo (http://gitphp.xiphux.com/)) is a PHP Git web interface with gitweb-like look (the xmms2 fork of Git), created by Christopher Han. It makes use of Smarty (http://smarty.php.net/) templates, so it can be easily modified and customized. It offers syntax highlighting via GeSHi (http://qbnz.com/highlighter/) PHP class, has support for snapshots and projects categories; it can be run with msysGit.

viewgit

- viewgit (http://viewgit.sourceforge.net/) is another PHP web interface for Git, written from scratch. It aims to be easy to set up and upgrade, light on dependencies, and comfortable to use. It provides all basic features such as commitdiffs, RSS feeds of changes, downloading of trees, checkouts, syntax highlighting using GeSHi, etc.

cgit

- cgit (http://hjemli.net/git/cgit/) is a fast and lightweight webinterface written in C. It uses libgit.a to avoid forking of Git commands and an on-disk cache to avoid duplication of expensive repository operations. The repository is hosted on hjemli.net (http://hjemli.net/git/) and can be cloned from git://hjemli.net/pub/git/cgit. Users of cgit include freedesktop.org (http://cgit.freedesktop.org) , compiz-fusion.org (http://cgit.compiz-fusion.org/) , dyne.org (http://git.dyne.org) , repo.or.cz (http://repo.or.cz/c/) , savannah.gnu.org (http://git.savannah.gnu.org/cgit/) and gnome.org (http://git.gnome.org/cgit/) .

gogs

- gogs (http://gogs.io/) (Github (http://github.com/gogits/gogs)) is a "painless self-hosted Git service". The goal of this project is to make the easiest, fastest, and most painless way of setting up a self-hosted Git service. Runs on Linux, Mac OS X, Windows and ARM using Go (http://golang.org/) .

Gitalist

- Gitalist (http://www.gitalist.com/) (gitweb (http://git.shadowcat.co.uk/gitweb/gitweb.cgi?p=catagits/Gitalist.git;a=summary) , GitHub (http://github.com/broquaint/Gitalist) , demo (http://example.gitalist.com/)) is a web frontend for Git repositories written in Perl and backed by Catalyst (http://www.catalystframework.org/) web framework. It used gitweb.cgi as template, and attempts to be URI compatible with gitweb.
  See also: Zac Stevens' talk at London Perl Workshop 2009 (http://conferences.yapceurope.org/lpw2009/talk/2446) .

GitStat

- *GitStat* (SourceForge project (http://sourceforge.net/projects/gitstat)) is a GPL'd, web-based Git statistics/monitoring system. It retrieves a specified Git tree, analyzes changesets, and shows graphical information like the number of changesets per day, the number of people who submitted changesets for a specific version (tag) etc. Users may subscribe to gitstat so that they automatically receive an email notification if any change is applied to a specified directory. See also gitstat 0.1: kernel development statistics / monitoring system (http://permalink.gmane.org/gmane.comp.version-control.git/56161) announcement on Git mailing list. You are welcome to try gitstat at http://tree.celinuxforum.org/gitstat

GitStats

- *GitStats* - with a 's', not the same as GitStat - (SourceForge project (http://sourceforge.net/projects/gitstats/)) is a statistics generator for Git repositories. It examines the repository and produces some interesting statistics from the history. Currently it outputs only HTML. As opposed to GitStat, it generates the HTML statically, and is therefore suitable to generate pages to upload to a dumb HTTP server. Some examples (http://gitstats.sourceforge.net/examples/) are available.

Git Treemap

- Git Treemap (https://github.com/joe42/git_treemap/blob/master/README.rst) generates an interactive treemap representation of a repository or a subdirectory thereof. Git Treemap offers three distinct views, which either provide an overview of each file's size, recent commits, or recent commits introduced by any specific author. The webpage is generated statically, much like with GitStats.

git-browser

- All Links Dead 2010/11/05. git-browser (http://straytree.com/) by Arteem Khodush is an experimental gitk-like web interface. It visualizes commit history graph and shows commit diffs. User interface is done in JavaScript, and is rather heavy on the *client* side. view online (http://straytree.com/git-browser/by-commit.html?r=git) download (http://straytree.com/git-browser.tar.bz2) gitweb mirror (http://repo.or.cz/gitweb.cgi/git-browser.git) .

pitweb

- pitweb (http://pitweb.danfis.cz) by Daniel Fiser is a web interface for Git repositories written in Python and licensed under LGPL.

PyjamasGitWeb

- PyjamasGitWeb (git clone gitolite@pyjs.org:pyjamasgitweb) is two projects in one: a JSONRPC web service which uses python-git to serve Git repository information, and a matching front-end as a pyjamas application. Pyjamas (http://pyjs.org) can run the front-end either compiled to javascript to run in a web browser or as a Desktop application (pure Python). Cacheing of answers to queries is performed (in the browser or in the desktop app) so that any file or Git link previously clicked on will come up instantly. Demo is at http://pyjs.org/pygit written in Python and licensed under LGPL.

Git Enablement Server (G.E.S.)

- Git Enablement Server (http://github.com/dvdotsenko/ges) is a combination of Git SmartHTTP server (git_http_backend replacement) and a web interface for consuming and browsing filesystem trees sprinkled with Git repositories.
- The front-end is entirely JavaScript (Single page, Sammy.js-based) that talks to server (Python, WSGI-based git_http_backend replacement + JSONRPC server supporting calls from Web client).
- The server is targeting Personal, Small Team use cases. The goal of the project is to make an absurdly simple-to-install-and-use *private* collaboration server.
- Install is just "git clone URI", "git submodule init", "git submodule update --recursive". In it's simplest form, the server is set-up and started just by double-clicking the executable. As of Nov 25, 2010, is tested to work equally well on Linux and Windows. To run, requires only two things: Python and command-line Git executable.

git2html

- git2html (http://hssl.cs.jhu.edu/~neal/git2html) generates a set of **static HTML** pages for exploring a Git repository. Because the pages are static, **no CGI script is required** to explore the repository. This makes git2html more secure and robust than other web viewers. Due to the decision to use static HTML, git2html is also less flexible, for instance, it does not support showing a diff between arbitrary revisions.

git webcommit

- git webcommit (https://github.com/lennie/git-webcommit/) A web-based checkout status viewer and commit tool. Useful for example when people want to keep track of changes when collaborating on a website that is tracked in Git, especially if some participants are not command-line savvy. Written in PHP and has very few requirements. Uses the git command-line tool for staging changes to the index and writing commits to the repository.

Grack

- Grack (http://github.com/schacon/grack) aims to replace the builtin git-http-backend CGI handler distributed with C Git with a Rack application. This reason for doing this is to allow far more webservers to be able to handle Git smart http requests.

Git-Webby (deprecated)

- Git-Webby (http://github.com/codigorama/git-webby/) was inspired in the Grack (http://github.com/schacon/grack) Smart-HTTP server handler (written by Scott Chacon (http://github.com/schacon)) but developed using Sinatra (http://www.sinatrarb.com) and aims replace the original git-http-backend including new features.

pgkiss

- (pgkiss (https://gitlab.com/gima/pgkiss)) is a very simple PHP wrapper for "git http-backend"-CGI script. It enables hosting of Git repositories on a web server with just PHP and Git available. Currently relies on web-server provided authentication if (or when) such is needed.

ungit

- ungit (https://github.com/FredrikNoren/ungit) is a web-based git client that aims to bring user friendliness to git without sacrificing its versatility. It displays an interactive graph of the repository in the browser, and provides easy ways to to create commits, rebase branches, stage changes by hunk, etc. The whole interface builds upon a visualization of the commit graph where nodes can be manipulated directly, e.g. to rebase a branch by simple drag-and-drop.

GitSSH2

- GitSSH2 (http://www.gitssh2.com) is a git web interface client built in Symfony's PHP framwork. It can connect to remote servers over ssh and run git commands such as commit, branch, push, pull, etc. Integrates with Github and Gitlabs issue tracker. Simple interfaces for users not to familiar with git, with user access roles to limit what different users can do per project.

Versioning for Magento 1

- Versioning (https://www.luigifab.fr/magento/versioning) (github (https://github.com/luigifab/versioning)) is a free and open-source module for Magento 1 translated in 7 languages. It allow admin to update his website with his GIT repository from Magento backend (git reset --hard). It also display upgrade history, local repository status (git status), local repository differences (git diff) and revisions differences (git diff from..to).

Access control / Project hosting

gitosis (unmaintained ?)

- *gitosis* (gitweb (http://eagain.net/gitweb/?p=gitosis.git) , README (http://eagain.net/gitweb/?p=gitosis.git;a=blob;f=README.rst;hb=HEAD) by Tommi 'Tv' Virtanen is a tool to manage Git repositories, provide access to them over SSH, with tight access control and not needing shell accounts. Described in Tv's cobweb: Snakepit and gitosis, things I've been working on (http://eagain.net/blog/2007/10/12/snakepit-and-gitosis.html) on author's blog and Hosting Git repositories, The Easy (and Secure) Way (http://scie.nti.st/2007/11/14/hosting-git-repositories-the-easy-and-secure-way) post on scie.nti.st (http://scie.nti.st) , Garry Dolley blog. *(Python, requires setuptools)*

SCuMD

- *SCuMD* (gitweb (http://github.com/gaffo/scumd) , by Mike 'gaffo' Gaffney is a tool provide access to Git repositories over SSH, with pluggable access controls and without the need for shell accounts or the normal ssh daemon exposed. *(Java, require java 1.6)*

ssh_acl

- ssh_acl (http://www.inf.ufpr.br/ribas/ssh_acl.html) by Bruno Ribas is a set of Bash scripts which can be used to manage Git repositories access.

gitolite

- *gitolite* (http://github.com/sitaramc/gitolite), by Sitaram Chamarty, (sitaramc at gmail), is inspired by gitosis, plus an urgent need to manage per-branch permissions. It is written entirely in Perl, and designed to be usable on any Unix machine that managed to install Git and Perl. It does not require root access to install or use.
- Other advantages: an "easy install" script + copious documentation to make things as painless as possible, especially with the critical ssh pieces; simpler, yet more powerful config file syntax; ability to split the config into parts and delegate authority to different people; better logging helps figure out exactly what was allowed/denied and why; "personal" branch namespace for developers. Also there's a version that allows "deny" in the access control list, making it truly powerful for pretty much all sorts of access restriction needs

Girocco

- *girocco* (http://repo.or.cz/w/girocco.git) by Petr Baudis is a Git project hosting framework, powering e.g. the repo.or.cz hosting site. It supports project forking, both mirror and push project modes, is highly configurable and flexible, but contains just the essential features. It is suitable both for public hosting and intranet Git hosting sites for developers' cooperation within closed shops.

SCM-Manager

- *SCM-Manager* (http://www.scm-manager.org) by Sebastian Sdorra is a java webapplication to create and manage Git, Mercurial and Subversion repositories. SCM-Manager allows easy access to the repositories over the HTTP or HTTPS protocol. It has centralized user, group and permission management and is completely configurable from within the web interface.
- By using SCM-Managers plugin API it is possible to automate various tasks. There is already a couple of plugins available, such as:
  - JIRA (enables you to close issues out of your IDE)
  - notify (the plugin sends automated messages to a list of subscribed addresses whenever a repo has changes)
  - LDAP

SCM-Manager Universe

- *SCM-Manager Universe* ([6] (https://www.scm-manager.com/scm-manager-universe)) by TRIOLOGY is an Ubuntu Linux based virtual machine, which comes with a bundle of ready to use open source software components for software development, including
  - SCM-Manager
  - Jenkins
  - Sonatype Nexus
  - SonarQube
  - Bugzilla

- The appliance is free and open source software and distributed under BSD-License.
- It has a comprehensive documentation, including tutorials, a sample project and tool descriptions.
- All of the software development tools authenticate over Single Sign-On against a central LDAP server with reasonable default users and groups configuration.
- The appliance has also a simple backup mechanism, which allows regular backups of all relevant files.

- TRIOLOGY is offereing a free update service and commercial support.

Gitblit

- *Gitblit (http://gitblit.com/)* (GitHub (http://github.com/gitblit/gitblit) , Google code (http://code.google.com/p/gitblit/)) is an open-source, pure Java stack for managing, viewing, and serving Git repositories (based on JGit). It's designed primarily as a tool for small workgroups who want to host centralized repositories. Gitblit is available in two variations:
  - *Gitblit GO* - a complete & integrated pure Java stack
  - *Gitblit WAR* - a traditional WAR distribution

(Java, requires Java 6 Runtime Environment (JRE or JDK) or up)

GitBucket

- GitBucket ([7] (https://github.com/takezoe/gitbucket)) is an easily installable Github clone powered by Scala with user management (LDAP).

Filesystem interfaces

git fs

- git fs (http://github.com/g2p/git-fs) A fuse filesystem for browsing Git commits and past worktrees.

figfs

- figfs (http://www.qotw.net/~reillyeon/projects/figfs/) A fuse interface to Git, with editable workspace support written in OCaml; related university site (http://www.seas.upenn.edu/~cse400/CSE400_2008_2009/websites/grant/project.html) ; inactive since 2009

GitFS (from mitch at sfgoth.com (Mitchell Blank Jr))

- GitFS (http://www.sfgoth.com/~mitch/linux/gitfs/) (via archive.org (https://web.archive.org/web/20100218153353/http://www.sfgoth.com/~mitch/linux/gitfs)) is a FUSE-based filesystem for working with source trees stored in Git repositories. The eventual goal is to provide a convenient way to work with lots of branches and patches. Currently (pre-release version 0.03) only very basic functionality is implemented - read-only access to the existing tags and objects. (dead since 2006?)

GitFS (Python)

- GitFS (https://github.com/rossbiro/GitFS) declared pre-alpha

Inferno filesystem translator

- gitfs (inferno) (http://github.com/manzur/gitfs) A Git filesystem translator for inferno, a portable plan9.

git-fuse-perl

- git-fuse-perl (https://github.com/mfontani/git-fuse-perl) by Marco Fontani. A simplistic attempt to provide a FUSE (http://fuse.sourceforge.net/) read-write interface to a Git repository. Requires perl-fuse (http://search.cpan.org/dist/Fuse/) .

gitfuse

- [8] (https://github.com/davesque/gitfuse) , Python

git-fs

- git-fs (https://github.com/patrickhaller/git-fs) , written in C, Patrick Haller

gitfs

- gitfs (https://github.com/wereHamster/gitfs) , blogpost (https://blog.caurea.org/2009/07/22/having-fun-with-gitfs.html) , C

Tools

Editors and IDE integration

Emacs integration

- Preliminary Emacs mode for Git from Alexandre Julliard (in `contrib/emacs` directory) consist of `git.el` which is a project tree browser similar to pcl-cvs, and `vc-git.el` which is a VC backend (see also vc-git-hacking at gnuvola (http://www.gnuvola.org/wip/)).
- Magit (http://magit.github.com/magit/) is an alternative Git mode for Emacs. Unlike the standard Emacs vc mode (git.el) it can take full advantage of Git-specific features. Supports TopGit and StGit (see above).
- *See also:* EmacsWiki:Git (http://www.emacswiki.org/emacs/Git) and EmacsWiki:Category:VersionControl (http://www.emacswiki.org/emacs/CategoryVersionControl) pages on EmacsWiki (http://www.emacswiki.org)

Eclipse plugin (EGit)

- EGit (http://www.eclipse.org/egit/) a plugin for Eclipse (http://www.eclipse.org/) IDE.
- EGit for Mylyn (http://www.javaforge.com/project/EGIT) integrates Git, Eclipse (http://www.eclipse.org/) and Mylyn (http://www.eclipse.org/mylyn/) .

JetBrains IDEs: IntelliJ IDEA 9.0 / PyCharm 1.0 / RubyMine / etc.

- As of October 2010, many (all?) JetBrains (http://www.jetbrains.com/) IDE products apparently have built-in support for GIT without any plugin downloads required. Product versions with GIT support include:
- IntelliJ IDEA 9.0+ (http://www.jetbrains.com/idea/) - for Java
- RubyMine 2.0+ (http://www.jetbrains.com/ruby/) - for Ruby
- PyCharm 1.0+ (http://www.jetbrains.com/pycharm/) - for Python, in Beta (as of October 2010)
- WebStorm 1.0+ (http://www.jetbrains.com/webstorm/) - for Web Dev
- PHPStorm 1.0+ (http://www.jetbrains.com/phpstorm/) - for PHP

Git4Idea

- The Git4Idea plugin provides basic Git integration support for the IntelliJ/JetBrains IDEA (v6 & above) development environment. The Git4Idea plugin can be downloaded from within IDEA itself using the IDE Settings->Plugins config panel. The plugin home page (http://plugins.intellij.net/plugin?id=3033) contains the latest release information and source code is also available here (http://github.com/markscott/git4idea/tree/master) . Git4Idea uses the Git command line tool to work; Cygwin 'git' on Windows is recommended.

KDevelop

- Development of a Git plugin for kdevelop has begun. Basic support implemented (http://drwxr-xr-x.blogspot.com/2008/06/now-kdevelop-has-basic-git-support.html)

NetBeans plugin

- NetBeans comes with Git support out of the box. Also see: NetBeansPlugin

Padre

- Padre-Plugin-Git (http://search.cpan.org/~KAARE/Padre-Plugin-Git/) provides basic Git integration for Padre, a text editor being developed as an IDE for Perl.

PIDA

- PIDA (http://pida.co.uk/) Integrated development environment supporting GIT, Subversion, Darcs, Mercurial, Monotone, Bazaar-NG (*bzr*). It allows embedding Vim or emacs. *(Python, GTK)*

Sublime

- GitSavvy (https://github.com/divmain/GitSavvy) Full git and GitHub integration with Sublime Text 3. *(Ruby)*

TextMate

- Git-tmBundle (https://github.com/textmate/git.tmbundle/) Provides a variety of tools for interacting with Git repositories, such as push, pull, tag, fetch, browse annotations, merge, switching branches, stashing, committing, etc. *(Ruby)*

Vim

- VCS Command plugin (http://code.google.com/p/vcscommand/) is a Vim plugin for various source code control systems and includes a Git component. It can be used to view the differences side by side, commit, blame, etc.
- fugitive.vim (https://github.com/tpope/vim-fugitive) is a Vim plugin with lots of features, for example: View any blob, tree, commit, or tag in the repository with :Gedit ... Edit a file in the index and write to it to stage the changes. Use :Gdiff to bring up the staged version of the file side by side with the working tree version and use Vim's diff handling capabilities to stage a subset of the file's changes. ... :Gblame brings up an interactive vertical split with Git blame output. i :Gmove does a <code>git mv</code> on a file and simultaneously renames the buffer. :Gremove does a <code>git rm</code> on a file and simultaneously deletes the buffer. Use :Ggrep to search the work tree (or any arbitrary commit) with <code>git grep</code>, ... :Glog loads all previous revisions of a file into the quickfix list so you can iterate over them and watch the file evolve! Use :Gbrowse to open the current file on GitHub... and much more.
- gitv (https://github.com/gregsexton/gitv) is a 'gitk clone' plugin for the text editor Vim. The goal is to give you a similar set of functionality as a repository viewer. Using this plugin you can view a repository's history including branching and merging, you can see which commits refs point to. You can quickly and easily view what changed to which files and when. You can perform arbitrary diffs (using Vim's excellent built in diff functionality) and you can easily check out whole commits and branches or just individual files if need be.
- Other Vim tips (http://vim.wikia.com/wiki/Using_Git_from_Vim)

Visual Studio

- Git Source Control Provider (http://gitscc.codeplex.com/) provides source control integration in Visual Studio 2008/2010 solution explorer through the standard interfaces.
- Git Extensions (http://code.google.com/p/gitextensions/) . See Graphical Interfaces above or website for details.

Merge tools

imerge

- git-imerge (https://github.com/mhagger/git-imerge) is an open-source tool that helps you perform difficult Git merges and rebases by allowing conflicts to be resolved incrementally. Simplified usage instructions are available in the tldr-pages project (https://github.com/tldr-pages/tldr/blob/master/pages/common/git-imerge.md#git-imerge) .

Diffuse

- Diffuse (http://diffuse.sourceforge.net) is a graphical tool for merging and comparing text files. Diffuse is able to compare an arbitrary number of files side-by-side and gives users the ability to manually adjust line-matching and directly edit files. Diffuse can also retrieve revisions of files from Bazaar, CVS, Darcs, Git, Mercurial, Monotone, Subversion, and SVK repositories for comparison and merging.

dirdiff

- dirdiff (http://www.liacs.nl/~sverdool/gitweb.cgi?p=dirdiff.git) (gitweb) is a graphical tool to display the differences (a la diff) between files in directories. Given two or more directory trees, dirdiff will display the differences between them in various glorious colors. It provides merging and the creation of patches. Link is to version modified by Sven Verdoolaege, which adds some more Git support, and which is based on dirdiff-2.1.tar.gz (http://samba.org/ftp/paulus/dirdiff-2.1.tar.gz) by Paul Mackerras. Not actively developed. In Tcl/Tk. (To be not confused with dirdiff (http://sourceforge.net/projects/dirdiff/) in Perl).

Meld

- Meld (http://meld.sourceforge.net) is a visual diff and merge tool. You can compare two or three files and edit them in place (diffs update dynamically). You can compare two or three folders and launch file comparisons. You can browse and view a working copy from popular version control systems such as Git, CVS, Subversion, Bazaar-ng and Mercurial, etc...

Kdiff3

- Kdiff3 (http://kdiff3.sourceforge.net/) is a graphical tool for merging and comparing text files. Kdiff3 is able to compare two or three files side-by-side and gives users the the ability to specify directories of files. Works on KDE/Linux, any UNIX or Linux that supports Qt and MS Windows (there is also a version for OS/X)

Code Compare

- Code Compare (http://www.devart.com/codecompare/) is a free tool designed to compare and merge different files and folders. Code Compare integrates with all popular source control systems: TFS, SVN, Git, Mercurial, and Perforce. The integration with Visual Studio helps to make all of the development and merging operations within one environment.

Per-file merge drivers

git-merge-changelog

- The estimable Bruno Haible wrote a merge driver for ChangeLog files. The driver currently lives in the gnulib sources (http://git.sv.gnu.org/gitweb/?p=gnulib.git;a=blob;f=lib/git-merge-changelog.c) .

Interaction with other Revision Control Systems

You may find the DVCS Migration Guide (http://www.catb.org/~esr/reposurgeon/dvcs-migration-guide.html) a useful resource if you are attempting to migrate a project history to Git.

Multiple Systems

Built-in import (built in)

- Core Git contains import commands for importing from other systems: `git-archimport` for GNU Arch (http://www.gnu.org/software/gnu-arch/) (`tla`), `git-cvsimport` for CVS (http://www.nongnu.org/cvs/) (needs `cvsps`: CVSps patches (http://ydirson.free.fr/en/software/scm/cvsps.html)), `git-svn` for Subversion (http://subversion.tigris.org/) (`svn`) (using `SVN::Perl` module), `git-quiltimport` for Quilt (https://savannah.nongnu.org/projects/quilt/) , and `git-p4` for Perforce (http://www.perforce.com/) (`p4`). For Quilt see also quilt2git / git2guilt below. For Perforce import there is also a script using alternative import method, which works without changing local state: git-p4-import (http://permalink.gmane.org/gmane.comp.version-control.git/25352) . There also exists a (dumb and slow) unpublished BitKeeper (`bk`) importer.

reposurgeon

- reposurgeon (http://catb.org/esr/reposurgeon/) is a repository-editing tool supporting multiple VCSes including CVS, Subversion, Git, Mercurial, Bazaar, darcs, bk, and SRC. It can import to Git from any of the VCSes it supports, and export from Git to all supported VCSes. It actually speaks Git import streams natively and relies on fast-import/fast-export tools to talk to most other VCSes; the exception is Subversion dump streams, for which it also has native read (but not write) support.

Tailor

- Tailor (http://wiki.darcs.net/DarcsWiki/Tailor) is an any-to-any version control system converter, with support for most free SCMs. It was written by Lele and it is implemented in Python. Note that you need VersionOne (http://wiki.darcs.net/DarcsWiki/Tailor/VersionOne) . Note also that as of 2012-12-09, it only has rudimentary support for branches, and mostly supports linear histories. Development effectively ceased years ago, making it unlikely these deficiencies will be repaired.

CVS

git-cvsexportcommit (built in)

- `git-cvsexportcommit` exports a commit from Git to a CVS checkout, making it easier to merge patches from a Git repository into a CVS repository. Supports file additions, removals, and commits that affect binary files.

git-cvsserver (built in)

- `git-cvsserver` is a CVS emulation layer for Git. It is highly functional; the mapping is bidirectional, so people who like CVS, can do both checkout and commit using CVS, and it shows up in Git. However, not all methods are implemented, and for those methods that are implemented, not all switches are implemented. Testing has been done using both the CLI CVS client, and the Eclipse CVS plugin. Most functionality works fine with both of these clients.

cvsps

- Warning: this code has been end-of-lifed by its maintainer in favor of cvs-fast-export. Several attempts over the space of a year to repair its deficient branch analysis and tag assignment have failed. Do not use it unless you are converting a strictly linear repository and cannot get rsync/ssh read access to the repo masters. If you must use it, be prepared to inspect and manually correct the history using reposurgeon.

- The 3.0 version of cvsps (http://catb.org/~esr/cvsps) has gained a --fast-export option that emits a Git fast-import stream. You can feed this directly to Git fast-import. CVS tags become Git lightweight tags. Branching is detected and handled. Merges are not, but you can easily fix those up with reposurgeon (which uses cvs-fast-export as its CVS-reading front end).

cvs-fast-export

- cvs-fast-export (http://www.catb.org/~esr/cvs-fast-export) (originally by Keith Packard as parsecvs) is a CVS file parser and Git import tool. It directly reads RCS ,v files and generates a Git-style fast-import-stream. reposurgeon uses this to read CVS repositories. Branching and tags are fully supported. It can process above 64K CVS commits per minute on common hardware. An auxiliary tool, cvssync, mirrors CVS repositories so cvs-fast-export can see the masters locally. A wrapper, cvsconvert, both drives the conversion and checks the history against CVS to verify correctness.

cvs2git

- cvs2svn (http://cvs2svn.tigris.org/) has a cvs2git (http://cvs2svn.tigris.org/cvs2git.html) mode for converting directly from CVS to Git. cvs2svn/cvs2git is very robust and gives high-quality conversions with many features and customization options (http://cvs2svn.tigris.org/features.html) . cvs2svn/cvs2git only supports one-time conversions. cvs2git's output can be read by Shawn Pearce's git-fast-import (http://www.kernel.org/pub/software/scm/git/docs/git-fast-import.html) , which has been bundled with core Git since version 1.5.
  Note that cvs2svn/cvs2git itself can't work with remote repositories. If you can't get a copy of the CVS repository directly, you might be able to recreate it indirectly via information read over the CVS protocol (http://cvs2svn.tigris.org/faq.html#repoaccess) using a tool like CVSsuck (http://cvs.m17n.org/~akr/cvssuck/) .

fromcvs/togit

- fromcvs (http://gitorious.org/fromcvs) by Simon 'corecode' Schubert is a conversion tool ment convert from CVS to multiple SCMs. At the moment there is a Git and a hg output and a SQLite output for quick CVS changeset navigation. For the CVS-to-Git conversion, git-fast-import by Shawn Pearce is needed (see above). fromcvs handles branches (though no tags yet), including native support for vendor branches. Incremental operation is supported. Conversion speed should be well above 10 changesets per second on common hardware. fromcvs/togit can take advantage of two CPUs.

gc-utils

- gc-utils (http://gcutils.sourceforge.net/) (freshmeat page (http://freshmeat.net/projects/gc-utils/) , gitweb (http://git.experimentalworks.net/?p=gcutils.git) , announcement (http://thread.gmane.org/gmane.comp.version-control.git/78540)) is a small set of scripts wrapping git-cvsexportcommit and git-cvsimport, allowing to import and update CVS repositories into Git very easy and let you export patches back into a CVS working copy. Written in Bash.

Bigitr

- Bigitr (https://github.com/mikjo/bigitr) Bidirectional, asymmetric synchronization between Git and CVS, across multiple branches. Synchronizes full states, not patches. Uses reserved branches in Git to represent state of CVS. Intended to allow development to move to Git in cases where processes have been implemented around CVS. Can push Git to CVS, pull from CVS into Git, or both. Does not use git-cvsimport or git-cvsexportcommit. Can be used with a repository initially created using git-cvsimport to preserve earlier history. It is configuration-driven, and can run arbitrary programs as hooks at many points in the process. Has built-in error reporting through email. Logs all Git and CVS operations, including output and error, and logs all diffs. Honors local CVS and Git configuration.

CvsntGitImporter

- CvsGitImporter (https://github.com/CamTechConsultants/CvsntGitImporter) is a front-end to fast-import for CVSNT. It uses the features of CVSNT to construct accurate commits and merges between branches, as well as importing tags and branches. It is able to reorder and split commits where necessary to cope with cases where tags have been moved around (within reason). Written in C#.

git-cvs

- git-cvs (https://github.com/osamuaoki/git-cvs) bidirectional operations between a single CVS tree and git

Subversion

When choosing git migration tools you need to clearly grasp the difference between an *gateway* (supporting live operation on a Subversion repository through git) and an *importer* (designed to move an entire Subversion history to git). The programs in this section were usually designed for one of these purposes and may have serious hidden flaws if used for the other.

Importers and gateways are listed first, then exporters, then auxiliary tools. See git-svn on how to use Git as an Subversion client. Here is a feature matrix of the production-quality importers:

	reposurgeon	git-svn	svn2git	SubGit	agito
role?	importer	gateway	importer	gateway	importer
handles branching?	yes	yes	yes	yes	yes
makes annotated tags?	yes	no	yes	yes	yes
splits mixed-branch commits?	yes	no	no	yes	yes
makes .gitignore files?	yes	no	no	yes	yes
documentation	excellent	good	good	excellent	passable
written in	Go	Perl	Ruby	Java	Python
last activity?	2020	2012	2012	2013	2012
maintainer	Eric S. Raymond	Eric Wong	James Coglin, Kevin Menard	TMate Software	Simon Howard

reposurgeon

- reposurgeon (http://www.catb.org/esr/reposurgeon/) is especially useful for Subversion-to-git and CVS-to-git migrations. It reads Subversion dump files directly and both reads and writes Git fast-import streams. It also reads CVS repositories using cvs-fast-export as a front end. Unlike many other conversion tools, reposurgeon fully supports multibranch repositories, automatically converts svn:ignore properties into .gitignore files, and can even handle the pathological mixed-branch commits sometimes found in Subversion repositories. The surgical commands assist in cleaning up conversion artifacts; they are especially useful for large, old repositories that have been through previous conversions with tools like cvs2svn.

  If you are doing a full import rather than gatewaying, reposurgeon is probably what you want. It has been tested against a lot of large, old, nasty repositories and is thus known to be robust in the presence of repository malformations (a property regularly checked by a test suite that is a rogue's gallery of Subversion botches).

git-svn (built in)

- git-svn (Git repository (http://git.bogomips.org/git-svn.git)), written by Eric Wong, is a conduit for bidirectional operation between a Subversion repository and Git. It supports Subversion branches and tags, importing multiple Subversion repositories into a single Git repository, and incrementally updating the Git repository with changes from the Subversion parent. It is designed for developers who wish to contribute

to projects that use Subversion, but would rather be using Git. Written in Perl using the SVN::Core Perl library.

- SAFETY WARNING: While git-svn has been commonly used for one-time full conversions of Subversion histories to Git and could have been listed as both gateway and importer, it is a poor tool for importing. It doesn't make annotated tag objects from (unmodified) Subversion tag copies, nor does it handle mixed-branch commits properly. Worse, it sometimes silently produces conversions with correct head content but incorrect or damaged older revisions. Its is especially likely to do this if the repository was at one time in CVS and converted with cvs2svn, but can have other causes as well. *See also:* git-svn-bugfix script (http://permalink.gmane.org/gmane.comp.version-control.git/102500%7C%7CANNOUNCE) In general git-svn is liable to stumble badly on my kind of malformation in the SVN history, producing an amplification of it in the git translation. Use a better-quality importer like reposurgeon or agito instead.

svn2git (Ruby)

- svn2git (GitHub (http://github.com/nirvdrum/svn2git/)), forked by Kevin Menard from James Coglan's project, is a Ruby tool for importing existing SVN projects into Git and GitHub. It uses git-svn; the major added feature is that it makes annotated Git tags from Subversion tag copies (however, it leaves the empty tag-copy commits in place). Be aware that as a wrapper for git-svn it inherits git-svn's tendency to stumble over repository malformations.

SubGit

- SubGit (http://subgit.com/) is a tool for establishing bi-directional Git-SVN mirror of Subversion repository, developed by TMate Software. SubGit works on the server side as a set of hooks installed into Git and/or SVN repository. Every incoming modification sent to SubGit-enabled repository triggers these hooks, as result SubGit immediately converts incoming changes to Git or SVN correspondingly. SubGit is not based on git-svn, it uses proprietary translation engine written in Java. Being SVN to Git importer, SubGit converts arbitrary SVN branches and tags into Git branches and tags preserving all the history including meaningful merge-tracking data; it also converts svn:ignore properties to .gitignore files, svn:eol-style/svn:mime-type properties to .gitattributes files. SubGit can also be used as a Git to SVN exporter.
  See also: SubGit vs. git-svn comparison (http://subgit.com/documentation/gitsvn.html)

agito

- agito (GitHub (https://github.com/fraggle/agito)) is a SVN-to-Git importer written by Simon Howard that behaves better than git-svn in handling tags on deleted and recreated branches. It replicates Subversion's default set of ignore properties with a .gitignore generated into the project root directory, but does not translate ignore properties explicitly set on individual Subversion directories.

git-svnconvert

- git-svnconvert, written by Rutger Nijlunsing in Ruby, appeared only on the Git mailing list in post git-svnconvert: YASI (Yet Another SVN importer) (http://marc.theaimsgroup.com/?l=git&m=114460050718696%7C%7CANNOUNCE) .

svn-all-fast-export

- svn-all-fast-export (Gitorious (http://repo.or.cz/w/svn-all-fast-export.git)) by Thiago Macieira, is a tool that exports branches from Subversion onto multiple Git repositories via git-fast-import. Written in C++, it uses the Qt Core library. It came into being to serve KDE's needs for a conversion to Git. It is reportedly faster than git-svn as of 2010-03-30: 8 minutes instead of 110 minutes on a multi-branch 15,5K revisions repository.

git-rails-plugins

- GIT sub-project porcelain that provides an alternative to git-svn dcommit when managing rails vendor plugins which are held in a central SVN repository. Currently git-svn is not able to dcommit into a Subversion repository when a Git repository contains submodules. git-rails-plugins (http://github.com/nazar/git-rails-plugins/tree/master) is an alternative (http://panthersoftware.com/articles/view/4/git-svn-dcommit-workaround-for-git-submodules) implementation to sub-modules.

git2svn (Perl/Bash)

- git2svn (http://voltar.org/git2svn/) , written by Paul Miller, is a Git repository exporter designed to allow a user to import numerous Git repositories into a trac (http://trac.edgewall.org/) wiki. Trac only supports SVN (natively) and only supports a one-repository configuration. Although there are Git plugins for Trac, they are cumbersome and (because of trac's design) only support one repository at a time. git2svn solves that by importing as many Git repositories as needed into directories of an SVN repository. It does this using native SVN and Git commands with Perl as glue.

git2svn (fast-export)

- git2svn (gitweb (http://repo.or.cz/w/git2svn.git)), written by Love Hrnquist strand, is a tool to convert the "git fast-export" dump into a SVN dump and load it into a new fresh SVN repository. git2svn also supports incremental updates. *NOTE: git2svn assumes its the only process that writes into the SVN repository*. Written mainly in Perl.
  See: git2svn 0.1 (http://permalink.gmane.org/gmane.comp.version-control.git/79680%7C%7CANNOUNCE) post on the Git mailing list.

git2svn.pl (yet another one)

- git2svn.pl (gitweb (http://repo.or.cz/w/nagiosplugins.git?a=blob;f=tools/git2svn.pl)), by Thomas Guyot-Sionnest, has been written to keep Nagios-plugins SVN tree updated from Git. It uses "git svn set-tree" to update the old subversion repository. Forced updated are possible (with manual intervention) as set-tree just apply a given snapshot with the last commit message. Under normal operation all commits are applied (not just the latest) and a "From:" line is inserted in the message to keep tracks of who committed what from SVN commit logs.

svnExport.pl (one-way, no rebase)

- svnExport.pl (GitHub (https://github.com/martinlong1978/Git-Svn-push)), by Martin Long, has been written to allow a one way sync to a subversion repository, without the need to rebase or write commit details back to the GIT repository. This makes is suitable for syncing a central or shared GIT repository to a SVN repository for backup, or to satisfy corporate/project requirements. The removal of the need to rebase prevents issues with downstream repositories. Works on multiple repositories, creating new branches in SVN automatically. For non-linear histories without a branch in tack, only the first-parent chain is committed.

svneverever

- svneverever (http://git.goodpoint.de/?p=svneverever.git;a=summary) , written by Sebastian Pipping in Python, is a tool that helps understanding history of a subversion repository *before* migrating it to Git. It runs through all history collecting additions of directories. In the end it presents a tree of *all directories ever having existed* in the repository. This information is a good basis to writing rules files for svn-all-fast-export/svn2git. A more detailed introduction to svneverever can be found in the author's blog post Before svn2git you want to run: svneverever (http://blog.hartwork.org/?p=763) .

git-rails-plugins

- GIT sub-project porcelain that provides an alternative to git-svn dcommit when managing rails vendor plugins which are held in a central SVN repository. Currently git-svn is not able to dcommit into a

Subversion repository when a Git repository contains submodules. git-rails-plugins (http://github.com/nazar/git-rails-plugins/tree/master) is an alternative (http://panthersoftware.com/articles/view/4/git-svn-dcommit-workaround-for-git-submodules) implementation to sub-modules.

git-svnsync

- git-svnsync (http://git-svnsync.gforge.inria.fr/) is a hook-based bi-directional server-side synchronisation tool between a git and a subversion repository. It is designed to allow a smooth transition of projects from a subversion repository to a git repository. Git-svnsync guarantees that any branch update ('svn commit' or 'git push') is applied atomically in both repositories, thus providing a seamless experience to the developers.
  - Unlike git-svn, it runs on the server and does not rebase commits. The developers can use the git repository normally (like any regular git repository), especially 'git merge' and 'git pull' are supported.
  - Compared to SubGit, it has a minimalist approach. The translation engine does not attempt to recreate the detailed non-linear history from git. Instead it creates only one svn commit for each git push. Consequently the accurate history is available only in the git repository and merge tracking is not interoperable (this means that merging must be made with git exclusively).

Mercurial

Another list of Mercurial related tools is maintained here (https://github.com/felipec/git/wiki/Comparison-of-git-remote-hg-alternatives) .

hg-to-git

- hg-to-git is Mercurial-to-Git converter, written by Stelian Pop, appeared only on the Git mailing list in post Mercurial to Git converter. (http://permalink.gmane.org/gmane.comp.version-control.git/36601) It supports incremental conversion, supports (multiple) hg branches and converts hg tags. Written in combination of shell script and Python.

hg-fast-export

- hg-fast-export is another converter using `git-fast-import` as backend and is written in Python using the Mercurial Python classes directly. It uses a single pass, supports tags, Mercurial named branches, incremental imports and merges. It is quite fast since no subshells and/or pipe communication is used except for the feed to `git-fast-import`. The source is available as part of the fast-export repository at repo.or.cz (http://repo.or.cz/w/fast-export.git) .

git-hg

- Utility for checking out and tracking a Mercurial repository: https://github.com/offbytwo/git-hg

git-remote-hg

- git-remote-hg (https://github.com/felipec/git-remote-hg) Git remote helper that allows continous bidirectional communication between a local git repository and remote mercurial repos in all the usual git tools

git-cinnabar

- git-cinnabar (https://github.com/glandium/git-cinnabar) git remote helper to interact with mercurial repositories. Contrary to other such helpers*, it doesn't use a local mercurial clone under the hood.

Darcs

darcs-to-git

- Having had problems with darcs2git and tailor, Steve Purcell (http://www.sanityinc.com/) wrote another simple darcs import tool called darcs-to-git, which can incrementally import a single darcs source repository: article (http://www.sanityinc.com/articles/converting-darcs-repositories-to-git) / darcs-to-git gitweb (http://git.sanityinc.com/?p=darcs-to-git.git)

Darcs-Git

- Darcs-Git (http://wiki.darcs.net/DarcsGit) was an effort to make Darcs work with Git repositories, in particular the Linux kernel repository. It was part Darcs-Unstable branch, available from http://abridgegame.org/repos/darcs-unstable .

darcs-fast-export

- A generic fast-export script for darcs by Miklos Vajna, works with Git fast-import, Bazaar fast-import and Mercurial fastimport. Vajna's original has disappeared, but a GitHub clone maintained by Brian Warner can be found here (https://github.com/warner/darcs-fast-export) .

darcs2git

- *darcs2git* (gitweb (http://repo.or.cz/w/darcs2git.git)) by Han-Wen Nienhuys (his homepage (http://www.xs4all.nl/~hanwen/)) is a Darcs to Git converter utilizing `git-fast-import`. It tries to map Darcs conflict resolutions onto Git branch merges. Written in Python.
  See: darcs2git.py - convert darcs repository using gfi (http://marc.info/?l=git&m=117123822725808&w=2) at the Git mailing list

Bazaar

Git-bzr

- Git-bzr allows bidirectional synchronization between Bazaar and Git repositories, much like git-svn. However, git-bzr is able to keep all merge history etc. Ruby original can be found at GitHub (http://github.com/pieter/git-bzr/) (port to shell script: mcepl (http://github.com/mcepl/git-bzr/) , kfish (http://github.com/kfish/git-bzr)).

bzr-fast-export

- Bazaar frontend for git-fast-import. Announcement here (https://lists.ubuntu.com/archives/bazaar/2008q1/038391.html) , lives in the `exporters` subdirectory of the bzr-fastimport project (https://launchpad.net/bzr-fastimport) .

BzrToGit

- A shell script that migrates a Bazaar repository to Git, written by Henrik Nilsson and can be found here (http://sortedbits.ath.cx/scripts/bzrtogit.sh) .

git-remote-bzr

- git-remote-bzr (https://github.com/felipec/git-remote-bzr) Git remote helper that allows continous bidirectional communication between a local git repository and remote Breezy/Bazaar repos in all the usual git tools

Perforce

git-p4 (built in)

- Git includes a built-in mechanism to pull code from Perforce and to submit back from Git to p4.
- git-p4 manual page (http://git-scm.com/docs/git-p4) .
- git-p4 example usage.
- git-p4 install and developer information.

git-p4import

- The script `gitp4-import.py` in `contrib/p4import` may not work anymore and is left for reference only.

git-p4raw

- The project `git-p4raw` can import an entire p4 depot faithfully, preserving branch relationships. It uses the p4d database files directly. See the GitHub repository (https://github.com/samv/git-p4raw) .

Monotone

mtn2git

- A Ruby script that imports Monotone repositories with different modes of operation; checkout and fast-import. Depending on the method is the completeness and reliability: checkout method is almost complete. It also allows further updates. Find it at GitHub (https://github.com/felipec/mtn2git) .

ClearCase

git-cc

- git-cc (https://github.com/charleso/git-cc) is a simple bridge between Clearcase and Git. It supports both base ClearCase and UCM. In case of base ClearCase, per-file histories are merged based on commit messages and timestamps. Incremental bi-directional synchronization is supported.

CC2GIT ClearCase GIT Bridge (Clearvision)

- Complete Integration between IBM Rational ClearCase and Git from Clearvision. The CC2Git Bridge allows developers to quickly and easily transport code between the two tools, enabling one continuous integrated SCCM environment with the reduced costs associated with open source software. Looking to Migrate; the CC2Git bridge allows companies to perform a controlled migration within longer timescales, learning and resolving dependencies as they arise, using live data but without affecting project teams. Read More on our website: http://www.clearvision-cm.com (http://www.clearvision-cm.com/clearvision-products/clearcase-and-git-bridge-cc2git/ash_flypage.tpl.html)

git-ucmimport (IBM Rational ClearCase)

- git-ucmimport, also written by Rutger Nijlunsing, is a converter from IBM Rational ClearCase UCM to Git. The current version can be found on http://eludias.dyndns.org/git-ucmimport.rb. The announcement was in http://www.gelato.unsw.edu.au/archives/git/0607/23790.html. In Ruby.

Others

quilt2git / git2quilt

- These utilities convert patch series in a quilt repository and commit series in Git back and forth. You can download files here (http://home-tj.org/files/misc) ; they can be found on HomeTJ webpage (http://home-tj.org/wiki/index.php/Misc) .

SCCS import

- There exists proposed `conftib/fast-import/git-sccsimport.py` in fast importer for SCCS files (http://marc.info/?i=c5df85930801200312o7cd5d307v1a39fb35179249a9@mail.gmail.com) post by James Youngman on the Git mailing list, currently maintained at github.com/robohack/git-sccsimport (https://github.com/robohack/git-sccsimport) ; and sccs2git-gfi (http://search.cpan.org/src/HMBRAND/VCS-SCCS-0.11/examples/sccs2git-gfi) importer by Sam Vilain in VCS::SCCS CPAN module.

rcs-fast-export

- rcs-fast-export (gitweb (http://git.oblomov.eu/rcs-fast-export)) is a Ruby script that can be used to export RCS histories in a form suitable to be piped to fast-import tools (only tested with git's). It parses RCS files directly, without requiring any external tool or library. It exports symbols as lightweight tags, and makes up a decently sensible name for branches, based on their initial revision number. It supports the typical `--authors-file` option to convert from usernames to full author identities, and it has an option to (lightweight) tag each RCS revision with its revision number.

rcs-fast-import

- rcs-fast-import (http://www.catb.org/esr/rcs-fast-import) (freshmeat (http://freshmeat.net/projects/rcs-fast-import)) by Eric S. Raymond is a tool that makes a valiant effort to stuff metadata from modern distributed version control systems back into RCS. It may be useful for extracting the revision histories of individual files from a project history. *(Python)*.

rpm2git

- rpm2git (http://gitorious.org/projects/rpm2git/pages/Home) (gitorious (http://gitorious.org/projects/rpm2git/repos/mainline) , announcement (http://www.gnome.org/~federico/news-2008-07.html#rpm2git)) by Federico Mena-Quintero is a tool to help import patches from SRPM (source rpm) into appropriate place in a Git repository *(in Python)*. *See also:* How to use rpm2git (http://blip.tv/file/1155469) screencast.

vss2git (Visual Source Safe to Git)

- vss2git (http://code.google.com/p/vss2git/) is a tool written in C# to convert a Visual Source Safe repository into a Git one. It preserves deleted files and renames, and attempts to construct meaningful changesets based on chronologically grouping individual project/file revisions.

vss2git (Visual SourceSafe to GIT or SVN)

- vss2git / vss2svn (http://github.com/Gediminas/vss2git/) - tool (written in C++) converts Visual SourceSafe database to GIT / SVN repository. Does not use direct analysis of VSS database - simple cycle "ss get" "git/svn commit" is used. Deleted files are skipped during conversion process. File revisions are grouped by user but split by date. Only cpp & h files are converted (hardcoded but easy to edit). (6GB database 10 years development took ~4 hours to convert)

tfs2git (Team Foundation Server to Git)

- tfs2git (http://github.com/WilbertOnGithub/TFS2GIT) - PowerShell script that converts Team Foundation Server repositories to a Git repository. Replays the history from the Team Foundation Server and adds it to a Git repository.

git-tfs (Team Foundation Server bridge)

- git-tfs (https://github.com/spraints/git-tfs/) is a two-way bridge between TFS and Git, similar to git-svn. You need .NET 4 and either the 2008 or 2010 version of Team Explorer installed.

SRC

- [http://catb.org/~esr/src SRC] is a special-purpose VCS for single-file, single-developer projects by the author of reposurgeon and rcs-fast-import. It uses RCS as its storage engine but provides a Subversion-like CLI with integer monotonic-increasing revision numbers, lockless operation, embedded help, and many other convenience features. SRC includes both fast-export and fast-import commands for interchange with Git. Import is done through rcs-fast-import and has significant technical limitations.

Git-Mediawiki

- Git-Mediawiki (https://github.com/Git-Mediawiki/Git-Mediawiki) Git remote helper that allows continous bidirectional communication between a local git repository and remote Mediawiki website in all the usual git tools

ldap-git-backup

- ldap-git-backup (https://github.com/elmar/ldap-git-backup) back up your LDAP data in a Git repository

BitKeeper

- The open-source release of BitKeeper has "bk fast export" and "bk fast-import" subcommands. As of March 2017, the 7.3.1ce version, fast export (to git) works well but fast-import (from git) rather poorly.

Hooks

Better Commit Policy

- Better Commit Policy (https://marketplace.atlassian.com/apps/1213415/better-commit-policy-for-jira-git-svn?hosting=server&tab=overview&utm_source=git-wiki&utm_medium=git-wiki-jcp-listing&utm_campaign=gh) is a pre-receive hook that verifies commits against a custom set of configurable rules. The rules can be set up in an app inside Atlassian Jira, so those can be based on Jira specific information (Jira issue key, committer present in a Jira group, committer has a Jira account, the committer is assigned to the issue mentioned in the commit message, etc) as well as on many other aspects (https://www.midori-global.com/products/better-commit-policy-for-jira/server/documentation/?utm_source=git-wiki&utm_medium=git-wiki-jcp-listing&utm_campaign=gh#conditions) (like tag, branches, commit length, expected string in commit, only certain file types committed, changes are on the right file path, etc). Hook scripts of the policy can be installed on the Server to verify commits at push time, but also on the developer's machine, to check commits right before commit time, making it sure non-compliant changes don't reach the server.

git-notify

- git-notify (http://source.winehq.org/git/tools.git/?a=blob;f=git-notify;hb=HEAD) is the post-receive hook used by the wine project. It sends one email per commit. It also supports sending cia notifications.

git-commit-notifier

- git-commit-notifier (https://github.com/bitboxer/git-commit-notifier) is the post-receive hook used by lot of projects. It sends HTML email commit messages splitting commits that were pushed in one step. Changes are highlighted per word.

git-multimail

- git-multimail (https://github.com/mhagger/git-multimail) is a post-receive hook for sending email notifications of pushes. It is a plug-in replacement for the post-receive-email script available from Git's contrib directory, except with bug fixes, more configuration options, and many new features, including a one-email-per-commit mode that threads commit emails by branch. It is easily extensible in Python.

archive-tag

- archive-tag (https://redmine.koumbit.net/projects/git-hooks/repository/entry/archive-tag?branch=master&rev=HEAD) is a post-receive/update hook developed by Koumbit.org (http://www.koumbit.org/) and formerly used by the Aegir project (http://groups.drupal.org/aegir) to ease maintenance of tarballs associated with a Git repository. Archives (and optionally checksums) are created when tags are pushed to a repository. This is part of Koumbit's git-hooks (https://redmine.koumbit.net/projects/git-hooks) project.

git-blacklist

- git-blacklist (http://github.com/cxreg/git-blacklist) is an update hook which allows a multi-user centralized repository to block known-bad commits and refs from being pushed.

Hooks for integration with IBM Rational Team Concert

- The following article (http://jazz.net/library/article/194) shows several ways other SCM systems can be used in conjunction with Rational Team Concert 2.x. The main focus of this article is how the integration of Git and IBM Rational Team Concert work items can be achieved using Git hooks.

CIAbot

Note: The CIA service died in September 2012 and is unlikely to be resurrected. See irker, below, for its replacement.

CIA (http://cia.navi.cx/) is a system for tracking open-source project commits in real-time. It needs a script installed as a post-commit or update hook to send Git log summary messages to the CIA server,

- The Version 3 ciabot scripts, ciabot.py (http://www.catb.org/~esr/ciabot/ciabot.py) and ciabot.sh (http://www.catb.org/~esr/ciabot/ciabot.sh) , have been rewritten to work well with Git version 1.6.0 and later. Use ciabot.py (http://www.catb.org/~esr/ciabot/ciabot.py) unless your hosting site forbids Python hooks; the shell version is more likely to break due to not being able to call out to utilities it needs.
- These scripts evolved by translation from ciabot.pl (http://www.kernel.org/git/?p=cogito/cogito.git;a=blob;hb=HEAD;f=contrib/ciabot.pl) by Petr Baudis in the `contrib/` directory of Cogito. The ancestral Perl is somewhat fragile and may fail with newer Git versions, although Koumbit.org (http://koumbit.org/) has a fixed version (https://redmine.koumbit.net/projects/git-hooks/repository/entry/ciabot.pl?branch=master&rev=HEAD) in their Git hooks project (https://redmine.koumbit.net/projects/git-hooks) . This is the "version 2" script, still written in Perl and still kicking.

The Version 3 scripts are part of Git contrib/ as of 2010-04-02.

The advantage of the version 3 scripts are that they are a full rewrite in Python, but they force you to include a URL in the IRC output, optionnally wrapped with tinyurl, which some people may not want.

The advantage of the version 2 script is that it does not talk to tinyurl (but they don't support gitweb links)

irker

irker (http://catb.org/~esr/irker) is a lightweight, decentralized replacement for the CIA service. It consists of two parts: a repository hook script, and a small service daemon that relays notifications to IRC servers. Unlike CIA, it doesn't have a single point of failure at a central server; rather, forge sites and major development hubs are expected to run their own irker instances which are privately accessible to local repository hooks. Git is well supported.

Gitspread

Gitspread (https://gitlab.com/sunny256/gitspread) solves the problem of pushing to several remotes or mirrors over a slow connection, for example a mobile connection from a laptop. By using a hook/daemon setup on a server with a fast connection, commits and branches are automatically spread to many mirrors by pushing only once to that repository.

Source code mirrors: GitHub (https://github.com/sunny256/gitspread) , Bitbucket (https://bitbucket.org/sunny256/gitspread) , repo.or.cz (http://repo.or.cz/w/gitspread.git)

Wikis, blogs, etc.

ikiwiki

- Ikiwiki (http://ikiwiki.info/) is a *wiki compiler*, written in Perl. It converts wiki pages into HTML pages suitable for publishing on a website. Ikiwiki stores pages and history in a revision control system (http://ikiwiki.info/rcs/) such as Subversion (http://ikiwiki.info/rcs/svn/) or Git (http://ikiwiki.info/rcs/git/) . There are many other features (http://ikiwiki.info/features/) , including support for blogging (http://ikiwiki.info/ikiwiki/blog/) , as well as a large array of plugins (http://ikiwiki.info/plugins/) (for example recentchanges (http://ikiwiki.info/plugins/recentchanges/) plugin).

Gollum

- Gollum (https://github.com/github/gollum) is a simple, Git-powered wiki with a sweet API and local frontend. It is the wiki engine of GitHub (https://github.com/) .

wikiri

- wikiri (http://blitiri.com.ar/p/misc.html) is a simple, single-file wiki written in Python. It uses reStructuredText for markup, and has optional Git support for history tracking. You can browse the repository (http://blitiri.com.ar/git/?p=wikiri) or just clone it from `git://blitiri.com.ar/wikiri`.

git-wiki

- git-wiki (http://atonie.org/2008/02/git-wiki) by Simon Rozet (http://purl.org/net/sr/) a wiki that relies on Git to keep pages\u2019 history and Sinatra (http://sinatra.rubyforge.org/) (Ruby web framework) to serve them. git-wiki is available on GitHub (http://github.com/sr/git-wiki) under WTFPL license terms (http://sam.zoy.org/wtfpl/) . See also git-wiki forks (http://github.com/sr/git-wiki/network) , especially al3x (http://github.com/al3x/git-wiki) 's and minad (https://github.com/minad/olelo) 's one. Demo installation for minad's wiki under http://git.awiki.org.

WiGit

- WiGit (http://el-tramo.be/software/wigit) by Remko Tronçon (http://el-tramo.be/) is a simple themable PHP wiki with Git as a backend, Textile (http://textile.thresholdstate.com/) PHP class for marking up text. It has history support, basic user support (from HTTP authentication), and pretty URLs. See WiGit: A Simple Git-based Wiki (http://el-tramo.be/blog/wigit-intro) blog post for introduction.

Nuki

- *Nuki* (GitHub (http://github.com/patrickt/nuki/tree/master)) by Patrick Thomson is a Wiki engine written in Nu (http://programming.nu/) and powered by Nunja (GitHub (http://github.com/timburks/nunja/tree/master)), a cross-platform web server that's scripted with Nu, that uses Git for version control, and Markdown (via NuMarkdown (http://programming.nu/posts/2007/10/10/markdown-in-nu)) for markup.

git-blog

- git-blog (GitHub (http://github.com/elliottcable/git-blog/tree/master)) is a minimalist blogware that uses Git, using Ruby to 'deploy' Markdown posts to XHTML when git-pushed to your server.

Tekuti

- Tekuti (http://wingolog.org/software/tekuti/) is weblog software written in Scheme, using Git as its persistent store. It uses WikiPedia:SXML for markup, and requires Guile (http://www.gnu.org/software/guile/) or other Scheme implementation, and mod_lisp (http://www.fractalconcept.com/asp/mod_lisp) (a bit of misnomer: it is similar to WikiPedia:FastCGI or SCGI that it creates long-lived CGI process, but with simpler protocol). Tekuti includes a script to import posts from Wordpress into a suitably-laid-out directory.
  Announcement: I'm telling you (http://wingolog.org/archives/2008/02/29/im-telling-you) on Andy Wingo wingolog.

Chuyen

- Chuyen (http://code.istique.net/?p=chuyen.git;a=summary) is a weblog software written in Python, using the django web framework and Git as its data storage backend through pygit. It uses ReStructuredText as its markup language.

eWiki

- eWiki (http://github.com/patrikf/ewiki/tree/master) is a Wiki with a Git backend, written in pure PHP (no ugly system() or exec() calls).

Pystl

- Pystl (http://blog.codezen.org/entries/tag/pystl) (gitweb (http://codezen.org/cgi-bin/gitweb.cgi?p=pystl.git)) by Jack Miller is very simple, small blog engine in Python, using Git for version control, Markdown for markup and Tenjin for templating. It doesn't have web interface, so it also have no comments.

gitit

- gitit (http://github.com/jgm/gitit/tree/master) , a Wiki written in Haskell, uses Git for storage, HAppS for a web server, and pandoc for markdown processing.

Shinmun

- Shinmun (http://www.matthias-georgi.de/shinmun) (GitHub (http://github.com/georgi/shinmun)) by Matthias Georgi is a small Git-based blog engine written in Ruby. Write posts in your favorite editor, `git-push` it and serve your blog straight from a repository. Supports Markdown, Textile or HTML files.

OddmuseGit

- OddmuseGit (http://www.foo.be/cgi-bin/wiki.pl/OddmuseGit) , two Bash scripts to interface between Git and the Oddmuse (http://www.oddmuse.org/) wiki. Handy for merging multiple wikis or having distributed Oddmuse wikis without changing the existing backend.

Levitation (export Wikipedia page history into a Git repository)

- Levitation (http://github.com/scy/levitation) is a project to convert Wikipedia database dumps into Git repositories. It has been successfully tested with a small Wiki (bar.wikipedia.org) having 12,200 articles and 104,000 revisions. Seems to have stalled.

Blawd

- Blawd (http://github.com/perigrin/blawd) is a blogging engine written in Perl for generating simple blogs from markdown files stored in Git repositories. It is successfully used in a couple different blogs in the Perl community. It is based upon Moose and Git::PurePerl.

Jekyll

- Jekyll (http://jekyllrb.com/) is the engine behind Github Pages (http://pages.github.com/) , and one of its deployment methods (http://jekyllrb.com/docs/deployment-methods/) is through git hooks.

Sputnik

- Sputnik (http://sputnik.freewisdom.org/en/) is an "extensible meta-wiki" that can use Git as a storage engine (http://sputnik.freewisdom.org/en/Git_Storage)

Bug/issue trackers, etc.

Review Assistant

- Review Assistant (http://www.devart.com/review-assistant/) is a code review plug-in for Visual Studio. It allows to create review requests and respond to them without leaving Visual Studio. Review Assistant supports TFS, Subversion, Git, Mercurial, and Perforce. The tool includes support for formal code review and allows users to add comments to a piece of code or to the entire review level. It allows teams to discuss code without scheduled meetings.

Critic

- Critic (https://github.com/jensl/critic) is web based code review tool based on Git. Reviews can be automatically created and updated by pushing changes to Git repository maintained by Critic. Reviewers can be assigned automatically based on users' preferences. Written in Python; stable and mature.

ditz

- Ditz (http://ditz.rubyforge.org) is a simple, light-weight distributed issue tracker designed to work with distributed version control systems like darcs and Git. Ditz maintains an issue database file on disk, written in a line-based and human-editable format. This file is kept under version control, alongside project code. Changes in issue state is handled by version control like code change: included as part of a commit, merged with changes from other developers, conflict-resolved in the standard manner, etc.

ticgit

- TicGit (http://wiki.github.com/schacon/ticgit) is a distributed bug tracker written in Ruby. It stores bug information in a separate branch, allowing multiple developers to control the bug tracking system by merging their ticgit branch.

git-issues

- git-issues (http://github.com/jwiegley/git-issues) is a distributed issue tracking system based on Git repositories, written as standalone Python script. One can check this script in along with project in order to ensure that all contributors are able to view the bug database using the same version of the script that was used to create them. Originally a rewrite of **ticgit** in Python. (*Python*)

cil

- CIL (http://www.kapiti.geek.nz/software/cil.html) - Command-line Issue List (written in Perl), allows fast creation of a distributed issue tracker which fits in well with any VCS.
  See also: Blog posts tagged 'cil' (http://www.kapiti.geek.nz/random/label:cil.html) on kapiti.geek.nz.

milli

- milli (http://www.steve.org.uk/Software/milli/) is a simple bug tracking system, written in Perl, which allows you to store reported bugs within your projects. To allow easy sharing of bugs the system will create a `.milli/` directory beneath your project, containing a single text file for each bug. The intention is that you'll add the `.milli` directory to your revision control system, so that anybody who can currently checkout, view, or commit updates may also do the same with bug reports. (*Perl*)

Bugs Everywhere

- Bugs Everywhere (http://bugseverywhere.org/) is a distributed bugtracker, designed to complement distributed revision control systems. It uses distributed revision control as a backend for bug state. Supported version control backends: Arch, Bazaar, Darcs, **Git**, Mercurial, Monotone. (*Python*)

SD (Simple Defects)

- SD (Simple Defects) (http://syncwith.us/sd/) is a peer-to-peer bug tracker that's built for sharing and use both online and offline. With SD, you can sync your bugs back and forth between other instances of SD, and even between SD and other bug trackers that SD supports (RT, Hiveminder, Trac, GitHub Issues, Google Code, Redmine). **Git** and Darcs VCS integration. (*Perl*)

git-cl

- *git-cl* (cgit (http://neugierig.org/software/git/?r=git-cl) , GitHub (http://github.com/martine/git-cl) , README (http://neugierig.org/software/git/index.cgi?url=git-cl/tree/README)) by Evan Martin is a tool for integrating code reviews on Rietveld (http://codereview.appspot.com). *(Python)*

Gerrit Code Review

- Gerrit (http://code.google.com/p/gerrit/) (demo (http://review.source.android.com/)) is a web based code review system, forked off the open sourced Rietveld (http://code.google.com/p/rietveld/) code review system. Gerrit runs in any standard Java servlet container. *See also:* Gerrit and Repo, the Android Source Management Tools (http://newblogtopic.blogspot.com/2008/11/gerrit-and-repo-android-source_04.html) blog post by Jeff Bailey

codeBeamer Collaborative ALM Solution

- codeBeamer (http://www.intland.com/products/cb/overview.html) - (free trial at JavaForge (http://www.javaforge.com)) Award winning collaborative ALM solution with wiki, document management, highly customizable issue tracking, continuous integration, version control with Git (and Mercurial, Subversion, etc.), source code analysis, forums and more.

BugTracker.NET

- An ASP.NET bug tracking system (http://ifdefined.com/bugtrackernet.html) with Git integration.

Jira Git plugin

- Git Plugin for JIRA (http://confluence.atlassian.com/display/JIRAEXT/Jira+Git+Plugin) A plugin for JIRA (http://www.atlassian.com/software/jira/) linking to Git repos using the Java implementation of Git.

git-bugzilla

- *git-bugzilla* (gitweb (http://code.istique.net/?p=git-bugzilla.git) , manpage (http://code.istique.net/?p=git-bugzilla.git;a=blob;f=docs/git-send-bugzilla.txt)) by Steve Frcinaux, is a tool to post (attach) patches to Bugzilla (http://www.bugzilla.org/) , modelled mostly after `git-format-patch`/`git-send-email`. It is

written in Perl and requires WWW::Mechanize (http://search.cpan.org/~petdance/WWW-Mechanize-1.50/lib/WWW/Mechanize.pm)

git-bz

- *git-bz* (script (http://git.fishsoup.net/cgit/git-bz/plain/git-bz) , cgit (http://git.fishsoup.net/cgit/git-bz/)) is a command-line tool for using Git with Bugzilla. You can file commits (in Git-format-patch format) as new bugs, attach them to existing bugs, or apply patches attached to bugs to your working tree. It has some neat features like rewriting the commit to include the URL to the newly filed bug before attaching it to the bug. git-bz is a self-contained Python script that acts as a Git subcommand and has no external dependencies other than Git.

scm-bug

- *scm-bug* (http://www.mkgnu.net/scmbug) is a generic glue between VCS systems (Git, SVN and CVS in 0-26-21) and bug-tracking software (Bugzilla, mantis, request tracker and test director in 0-26-21). It provides features such as notifying the bugtracker when a related commit is done, optional workflow-enforcing checks, version description document generation. Written in Perl.

IBM Rational Team Concert

- The following article (http://jazz.net/library/article/194) shows several ways other SCM systems can be used in conjunction with Rational Team Concert 2.x. The main focus of this article is how the integration of Git and IBM Rational Team Concert work items can be achieved using Git hooks.

Backups, metadata, and large files

bup

- *bup* (GitHub (http://github.com/apenwarr/bup)) by Avery Pennarun is software that can efficiently backup large amounts of data (hundreds of GB) across large numbers of files (millions), using a Git-formatted repository as storage. It also efficiently stores huge individual files, such as virtual machine disks, and can backup directly to a remote machine rather than using space on the local machine. It is being used by major companies such like WP hacked Help, they utilize it effectively to take database backups of wordpress malware redirect (https://secure.wphackedhelp.com/blog/wordpress-malware-redirect-hack-cleanup/) infected websites which require cleanup. A FUSE module is included so you can mount your backup repository as a filesystem and restore files that way.
  See also: Bup: it backs things up (http://apenwarr.ca/log/?m=201001#04) blog post (and comments (http://news.ycombinator.com/item?id=1029990)).

git-annex

- git-annex (http://git-annex.branchable.com) allows Git to track files whose content should be tracked out-of-band some other way. This is useful when dealing with files larger than Git can currently easily handle, whether due to limitations in memory, checksumming time, or disk space. Annexed files can co-exist in the same Git repository with regularly versioned files, which is convenient for maintaining documents, Makefiles, etc that are associated with annexed files but that benefit from full revision control.

chronoversion

- Chronoversion is a simple Python script which attempts to use the Git content tracker to provide chronological snapshots similar to Plan 9's *venti* filesystem. The two differences are that, being based on Git, it isn't a filesystem in the sense of needing kernel support to gain access to the data (so it can be used on systems where you can only install userspace programs) and it allows programmatic decisions about what to snapshot, rather than recording everything into the snapshot. By David Tweed. It can be downloaded (at least currently) from chronoversion.tgz (http://www.personal.rdg.ac.uk/~sis05dst/chronoversion.tgz) .

metastore

- metastore (https://github.com/przemoc/metastore) originally written by David Härdeman is a tool which allows the metadata to be stored in a separate file, which can be stored along with the rest of the data in the repository (or separately). This is also useful for tripwire type checks and for other types of storage which drops some of the metadata (tar comes to mind).

gitperms

- *gitperms* (GitHub (https://github.com/harleypig/gitperms)) is a metastore clone written by Harley J Pig *(Perl)*.

etckeeper

- etckeeper (http://kitenet.net/~joey/code/etckeeper/) (gitweb (http://git.kitenet.net/?p=etckeeper) , announcement (http://kitenet.net/~joey/blog/entry/announcing_etckeeper/)) by Joey Hess is a collection of tools to let /etc be stored in a Git repository. It hooks into apt to automatically commit changes made to /etc during package upgrades. It used to (http://git.kitenet.net/?p=etckeeper.git;a=commit;h=a9ce9965c06571a57522106691dac2f9892125ba) use metastore (http://david.hardeman.nu/software.php) but now uses a script to track file metadata that Git does not normally support, but that is important for /etc, such as the permissions of /etc/shadow. It's quite modular and configurable, while also being simple to use if you understand the basics of working with Git.
  See also: IsiSetup (http://www.isisetup.ch/)

git-cache-meta

- git-cache-meta by jidanni is minimalistic approach to saving file ownership and permissions. It was submitted to `contrib/metadata` and can be found in git-cache-meta -- file owner and permissions caching, minimalist approach (http://permalink.gmane.org/gmane.comp.version-control.git/105133%7C%7CPATCH) post at Git mailing list.

Flashbake

- Flashbake (http://bitbucketlabs.net/flashbake/) is something between continuous, automated backup system and version control for writers. It is written in Python and uses Git for storage.
  See also: Flashbake: Free version-control for writers using Git (http://www.boingboing.net/2009/02/13/flashbake-free-versi.html) by Cory Doctorow on BoingBoing.

SparkleShare

- SparkleShare (http://sparkleshare.org/) is a collaboration and sharing tool. It aims to be simple to use. Written in Mono/C#.

Persy

- Persy (http://persy.digitalkultur.net/) is a file synchronization and backup tool. Very focused on Ubuntu users.

Subprojects or sets of repositories

These tools allow you to manage a "superproject" that consists of a set of repositories or incorporates other repositories ("subprojects") as subtrees. As applicable, they may allow you to make changes to the subprojects that are specific to the superproject, merge those changes with updates to the upstream subprojects, and send changes back to the upstream subprojects. Each tool has a brief analysis of the fundamental design choices and their implications relevant to maintaining a superproject. Someone who cares about the differences relevant to sending changes back upstream is invited to write that part!

Submodules (built in)

- `git submodule`, etc.
- The superproject tree stores the ID of the corresponding subproject commit, and the subproject repository location is stored in `.gitmodules` in the superproject. The git commands automate much of the work of maintaining a subproject working tree at the configured path in the superproject working tree, but as of 2016-10-27, many operations are still more of a hassle than working with a single project (summary (http://marc.info/?l=git&m=147753316624103&w=2)). If you want to make project-specific modifications to a subproject, you have to host your own fork repo of the subproject and make a commit in the subproject, followed by a commit to update the pointer in the superproject. Since the subproject commit ID is just data, revert, merge, and rebase on the superproject have the effects one would expect on the choice of subproject commit ID.

repo

- repo (http://android.git.kernel.org/?p=tools/repo.git) is a project to use Git to build OS distributions; similar to Git submodules, it can track specified branches from Git projects.
  See also: Gerrit and Repo, the Android Source Management Tools (http://newblogtopic.blogspot.com/2008/11/gerrit-and-repo-android-source_04.html) blog post by Jeff Bailey
- It looks like there's no support for tracking combinations of commit IDs that are intended to be used together, though in some scenarios it may be possible to guess based on commit times. Please update this if you know better.

metagit

- metagit (https://github.com/stettberger/metagit) is a tool for defining sets of Git (or whatever scm) repositories and perform actions like clone/pull/push on them. It is possible to list repositories on a remote site, e.g. find all repositories in a directory via ssh. The original announcement is here (http://article.gmane.org/gmane.comp.version-control.git/158452)
- As with repo, it looks like there's no support for tracking combinations of commit IDs that are intended to be used together, though in some scenarios it may be possible to guess based on commit times. Please update this if you know better.

gitslave

- gitslave (http://gitslave.sourceforge.net) by Seth Robertson creates a group of related repositories all of which are concurrently developed on and on which all Git operations should operate, so when you branch, each repository in the project is branched in turn. Similarly when you commit, push, pull, merge, tag, checkout, status, log, etc; each Git command will run on the superproject and all slave repositories in turn.
- As with repo, it looks like there's no support for tracking combinations of commit IDs that are intended to be used together, though in some scenarios it may be possible to guess based on commit times. Please update this if you know better.

git-subtree (contrib)

- git-subtree (https://github.com/git/git/tree/master/contrib/subtree) , originally by Avery Pennarun, is an experimental alternative to the git-submodule command, making it easier to use workflow which uses "subtree" merge strategy. The series introducing git-subtree was posted twice on the Git mailing list: here (http://thread.gmane.org/gmane.comp.version-control.git/117612) and here (http://thread.gmane.org/gmane.comp.version-control.git/118635) .
  See also: A new alternative to Git submodules: git subtree (http://apenwarr.ca/log/?m=200904#30) blog post (and comments (http://news.ycombinator.com/item?id=604889)). Some Git distributions install it by default.
- Subproject content is included in the superproject tree, so developers who don't interact with upstream don't have to do anything special. Supports both "import history" and "squash" modes. All metadata is in commit messages. In squash mode, revert, merge, and rebase on the superproject have unexpected effects on the implicit subproject-commit pointer.

braid

- Braid (http://cristibalan.github.io/braid/) is by Cristi Balan, in Ruby. It was inspired by Piston (http://piston.rubyforge.org/index.html) (a similar tool for SVN).
- Subproject content is included in the superproject tree, so developers who don't interact with upstream don't have to do anything special. Supports both "import history" and "squash" modes. Metadata is in `.braids` in the superproject. Squash mode does not create any extra commits, so revert, merge, and rebase on the superproject work the same as with a submodule.

git-subrepo

- git-subrepo (https://github.com/ingydotnet/git-subrepo) : TODO: figure out the important differences and explain them

git-subhistory

- git-subhistory (https://github.com/laughinghan/git-subhistory) : TODO: figure out the important differences and explain them

Other tools

git gitlab-init

- git gitlab-init (https://github.com/qguv/git-gitlab-init) by Quint Guvernator initializes a repository locally and remotely on Gitlab. It streamlines the process of making new repositories by removing a step that can't normally be done from the command-line.

gitdiffbinstat

- gitdiffbinstat (https://github.com/matthiaskrgr/gitdiffbinstat) by Matthias Krüger is a Bash script to compare branches/commits/tags etc by actual file size. The script does also detect if a file has just been renamed. It will output a summary a bit similar to the way `git diff --shortstat` does it with line changes.

git-ftp

- *git-ftp* ([9] (http://github.com/ezyang/git-ftp)) by Edward Z. Yang is a simple script written in Python for uploading files in a Git repository via FTP, only transferring new files and removing old files.

GitFTP-Deploy

- GitFTP-Deploy (https://gitftp-deploy.com) is a graphical client for uploading files from Git repository via FTP or SFTP, only transferring new files. macOS X 10.10+ (closed source, commercial). A trial version is available.

git-svn-replay

- http://search.cpan.org/perldoc?git-svn-replay by Paul Miller is a Perl script that replays Git commits into an SVN repository for the purpose of showing changes in packages like Trac, which only understand SVN natively, without the need for special plugins. It is probably not useful for anything besides listing changes for packages like Trac. GSR differs from git-svn, which is used for bi-directional commits. GSR works hard to preserve commit times and re-formats commit messages in a way that's nicer for Trac to display.

git2rss

- git2rss (http://bent.latency.net/git2rss) by Bennett Todd is a Perl script to generate an RSS summary of a Git repository. Currently to be used rather as example. Similar to darcs2rss

(http://darcs.simplicidade.org/repos/darcs2rss/darcs2rss) .

https://github.com/kimmobrunfeldt/git-hours

git2cl

- git2cl (http://josefsson.org/git2cl/) (gitweb (http://repo.or.cz/w/git2cl.git)) by Simon Josefsson is a Perl script to convert Git logs to GNU ChangeLog (http://www.gnu.org/prep/standards/html_node/Change-Logs.html) format. The tool expects output from `git --pretty --numstat --summary` on standard input. Inspired by cvs2cl (http://www.red-bean.com/cvs2cl/) tool. *References:* ANNOUNCE: git2cl (http://permalink.gmane.org/gmane.comp.version-control.git/41170) message on the Git mailing list.
- See also alternate solution: New script: git-changelog.perl - revised (http://marc.info/?l=git&m=119403382117250&w=2%7C%7CPATCH) by Ronald Landheer-Cieslak, with slightly different output, less dependencies, and being standalone command and not a filter.

git status-report

- `git status-report` (announcement (http://weblog.masukomi.org/2010/07/16/git-status-report) , GitHub (http://github.com/masukomi/git_accessories/blob/master/git-status-report)) by Kate Rhodes (*masukomi*) is a script (in Ruby) that outputs a bullet list of all your commits for the past *n* days, or the commits of one of your fellow contributors from any number of branches and repos. Useful to put together a status report for boss.

git-completion.bash (in `contrib/completion`)

- Bash tab-completion shipped with core Git. Inspired by and superceeds the tab-completion provided by gitcompletion (see below). Completion support is provided for almost all commands, including completion of local and remote reference names and long command line options.

gitcompletion and generate-completions

- gitcompletion (http://www.hawaga.org.uk/ben/tech/gitcompletion/) (Git repository (http://www.hawaga.org.uk/gitcompletion.git/)) by Ben Clifford is Bash tab-completion for Git, gitk, Cogito (*cg*) and StGIT (*stg*). Cogito has the cogito bits of the above already included. There is also `generate-completions.sh` by Johannes Schindelin, available on the Git mailing list archive generated completions (http://marc.theaimsgroup.com/?l=git&m=114515313911717%7C%7CFYI) . Yet another StGIT completions by Paolo Giarrusso, aka Blaisorblade are available at Re: latest stg/git commandline completions code (http://marc.theaimsgroup.com/?l=git&m=113025114026388) .

Gitdm

- *Gitdm* (the "git data miner") (gitweb (http://repo.or.cz/w/git-dm.git) , mirror) is the tool that Greg KH and Jonathan Corbet have used to crank out statistics on where kernel patches come from at LWM (configuration files (http://lwn.net/images/gitdm/gitdm-config-2.6.26.tar.bz2) , tar.bz2). Public repository for gitdm can be found at: git://git.lwn.net/gitdm.git

git-hours

- *git-hours* (https://github.com/kimmobrunfeldt/git-hours) by Kimmo Brunfeldt is a Node package that calculate hours spent on Git repository.

git-what-branch

- git-what-branch (http://github.com/SethRobertson/git-what-branch) by Seth Robertson allows you to perform forensics investigation to discover the merge-path a particular commit took in getting to any named branch or user specified object.

log remapper

- remapper (http://www.linux.org.uk/pub/people/viro/remapper/) (gitweb (http://www.kernel.org/git/?p=linux/kernel/git/viro/remapper.git) , snapshots (http://www.codemonkey.org.uk/projects/git-snapshots/remapper/)) by Al Viro, announced in Re: |RFC| Add "rcs format diff" support (http://marc.theaimsgroup.com/?l=git&m=114756555630327) , is a tool to (re)map position (file name and line number) in one revision to corresponding position (file name and line number) in other revision. E.g. user sends report "linux-2.6.16 with configuration XYZZY fails to compile with: `arch/wii/kernel/l33t.c:42:parse error`" and with the remap tool you can map the error report to the current commit's state, when that bit of code got moved around to `drivers/input/wii/l33tcontrol.c:31337`. It also already very useful for log comparison, with the noise due to line number changes excluded. Note that it's not just for build logs; the thing is useful for sparse logs, grep -n output, etc., etc.
  See also: Kernel space: Buried in warnings (http://www.linuxworld.com/news/2006/110706-kernel-warnings.html) article at LinuxWorld.com (http://www.linuxworld.com/) .

ArcheoloGIT

- In planning stage: an ArcheoloGIT tool to help reconstructing history from individual pieces, by Yann Dirson. Written in Perl. See Yann's Software page (http://ydirson.free.fr/en/software/) . *(stalled)*
  Repository: http://ydirson.free.fr/soft/git/argit.git/

gitcharts

- (*gitcharts* (http://github.com/dabroz/gitcharts)) by Tomasz Dąbrowski (dabroz) is a small C# app that generates charts based on lines of code (LOC) statistics in a given Git repository over time, and supports multiple projects in one repository.

pepper

- pepper (http://scm-pepper.sourceforge.net) is a scriptable repository statistics tool with support for Git. It ships with many common reports, including LOC and activity plots generated using the built-in Gnuplot API.

git-split

- git-split (http://people.freedesktop.org/~jamey/git-split) by Jamey Sharp and Josh Triplett is a tool to split the history of a Git repository by subdirectories and ranges, for example to break single monolithic repository imported from foreign SCM into smaller modules, easier to manage. Written in Python.
- The core command git filter-branch (http://www.kernel.org/pub/software/scm/git/docs/git-filter-branch.html) can be used to the same effect - examples here: http://log.emmanuelebassi.net/archives/2007/09/when-the-levee-breaks/ .

reposurgeon

- *reposurgeon* ([10] (http://www.catb.org/esr/reposurgeon/)) is an editing tool for repository histories that will work with any version-control system supporting import and export of Git fast-import streams. It supportes editing old comments and metadata, removing commits, fixing timezones, coalescing single-file commits, and other tricky operations. It can be very helpful in cleaning up cruft from repository conversions. As of 2.0 it has the ability to read Subversion dump files directly, and is the best option for Git conversions of Subversion repositories with messy histories.

git_fast_filter and git-rewrite-commits

- *git_fast_filter* (gitorious (http://gitorious.org/projects/git_fast_filter)) by Elijah Newren assists with quickly rewriting the history of a repository by making it easy to write scripts whose purpose is to serve as safe filters between fast-export (http://www.kernel.org/pub/software/scm/git/docs/git-fast-export.html) and fast-import (http://www.kernel.org/pub/software/scm/git/docs/git-fast-import.html) . git_fast_filter comes

with example programs,and a basic test-suite.
See announcement on the Git mailing list (http://permalink.gmane.org/gmane.comp.version-control.git/116028) and the following thread.

- *git-rewrite-commits* by Sven Verdoolaege is an alternate interface to git-filter-branch (http://www.kernel.org/pub/software/scm/git/docs/git-filter-branch.html) -like functionality (more similar to cg-admin-rewritehist from Cogito). It is available from the Git mailing list archives fro PATCH 0/6| Add git-rewrite-commits v2 (http://thread.gmane.org/gmane.comp.version-control.git/52323%7C) thread.

Timetrack

- git timetrack (http://wadobo.com/trac/dtt) is a new Git command that allows management of project time dedication by attaching a time-spent note per commit with minimal extra effort for the developers, providing then advanced querying of the dedicated time.

gitco

- gitco (http://blog.josefsson.org/?p=8) by Simon Josefsson is a tiny shell script to remove unversioned files from Git directories, similar to cvsutils (http://www.red-bean.com/cvsutils/) 'cvsco' or subversion's svn-clean. It basically calls the standard git clean (http://www.kernel.org/pub/software/scm/git/docs/git-clean.html) and git reset --hard (http://www.kernel.org/pub/software/scm/git/docs/git-reset.html) commands.

setuptools_git

- setuptools_git (http://ygingras.net/b/tag/setuptools_git) (formerly gitlsfiles) is Git plugin for Setuptools (http://peak.telecommunity.com/DevCenter/setuptools) , by Yannick Gingras (Python). Setuptools is a collection of enhancements to the Python distutils that allow you to more easily build and distribute Python packages, especially ones that have dependencies on other packages.

git-now

- git-now (https://github.com/toshi-kawanishi/git-now) is a new Git command. It create temporary commits. The commits' messages include time-stamp and diff.

git-buildpackage

- git-buildpackage (http://packages.debian.org/git-buildpackage) is a tool for managing the Debian packaging of an application in Git. Documentation (http://honk.sigxcpu.org/projects/git-buildpackage/manual-html/gbp.html) . There are other solutions and suggestions for Debian packaging and Git, for example see GitSrc (http://wiki.debian.org/GitSrc) .

gear

- Gear (http://en.altlinux.org/Gear) is a tool for building RPM packages from Git repositories.

0release

- 0release (http://0install.net/0release.html) can be used to make new releases of your software. It handles details such as setting the version number and release date, tagging the release in GIT, exporting, running any unit-tests, uploading the archive, pushing the tag, and updating your Zero Install feed. The workflow is: run the script (creates a tarball), do any manual release testing, then either publish everything or fail the release (reverting to your repository's previous state ready to make changes and try again). Only two user interactions are required: starting the script and confirming the generated release.

./release.sh

- ./release.sh (http://osndok.com/git-release/) is a simple mechanism to create and tag release branches on a remote (centralized/blessed) Git repository. Alpha-quality. Does not have any "server-side" hooks to actually do something with the releases or method to simplify tracking new release branches. (Do not confuse this with the "build recipes" in msysGit.)

Ryppl

- Ryppl (http://ryppl.org) is a a cross-platform package management system designed to accomodate both end-users and developers. Unlike a traditional package manager, which delivers binaries and/or a source snapshot, when ryppl downloads a package, it gives you a clone of a Git repository, with that package's entire development history. It also pulls down the correct versions of any repositories on which the requested repository depends. If you're an ordinary end-user, the fact that they are Git repositories may be invisible to you, but if you're a developer, it means you're already prepared to work on the package, keep track of your changes, and submit them to the official maintainer(s).
- Ryppl includes facilities for building, testing, and installing packages on the local machine. However, it also has integrated support for remote testing. That is, you can arrange that tests be run on build slaves located “out there” on the internet. This allows developers to discover portability issues without having direct access to every build platform.

fedora-packager

- fedora-packager (https://fedorahosted.org/fedora-packager) is a collection of tools for Fedora (http://www.fedoraroject.org) developers and includes fedpkg which is specifically for managing Fedora packages within Fedora's Git based package repository. Fedpkg thinly wraps common Git commands such as clone, commit, pull/push, etc...

gitbuilder

- gitbuilder (GitHub (http://github.com/apenwarr/gitbuilder/) , demo (http://versabanq.com/demo/gitbuild/)) by Avery Pennarun is a set of relatively simple scripts for automatically building Git-hosted project, optionally running unit tests, and reporting pass/fail results. In case of failures, it automatically uses git-bisect(1) (http://www.kernel.org/pub/software/scm/git/docs/git-bisect.html) to try to track down the first commit that started failing. It's also smart about branches; it knows how to build each commit only once, no matter how many branches include it, thus greatly simplifying future bisections.
  See also: Announce: gitbuilder, an autobuilder for Git-based projects (http://permalink.gmane.org/gmane.comp.version-control.git/94449) post on the Git mailing list.

bbchop

- *bbchop* (GitHub (http://github.com/Ealdwulf/bbchop)) by Ealdwulf Wuffinga is a tool for bisecting on an intermittent bug, using Bayesian decision theory.

Parabuild

- Parabuild (http://www.viewtier.com/products/parabuild/index.htm) is a Continuous Integration and Release Management system with native support for Git (http://www.viewtier.com/products/parabuild/git_continuous_integration_release_management.htm) .

Nico Schottelius scripts

- Some of Nico Schottelius (unix|linux) scripts (http://unix.schottelius.org/scripts/) deal with Git: `db-dump+git.sh`, `update-gitweb.sh`, `update-git+cogito.sh`.

William Morgan Git tools

- William's miscellaneous Git tools (http://git-wt-commit.rubyforge.org/) include `git-wtf`, a tool for working with feature branches (and its precursor `git-show-merges`), `git-publish-branch` to ease the task of

"publishing" a branch (mostly superseded by `git push -u <origin> <branch>`), and `git-rank-contributors` which ranks all the contributors by the size of their diffs (as opposed to `git shortlog --numbered --summary` which counts number of commits). By William Morgan and Jeff Balogh *(Ruby)*

git-central

- *git-central* (GitHub (http://github.com/stephenh/git-central)) by Stephen Haberman is a small collections of hooks, scripts, and practices (documentation) for use with a centralized (non-distributed/corporate) Git repository. Those include: svn-like revision numbers, combined diff-enabled commit emails, Hudson hooks, Trac hooks, branch locking, etc.

git-diffall

- *git-diffall* (GitHub (http://github.com/thenigan/git-diffall)) by Tim Henigan is a script which performs a directory diff on a complete Git repository using an external diff utility specified by the user. This script is compatible with all the forms used to specify a range of revisions to a Git diff.

git-blameall

- *git-blameall* (website (http://1dan.org/git-blameall)) Shows every line that was ever in the file, along with information about when it was added or deleted.

git-edit-index

- `git-edit-index` (http://kgb2.thruhere.net/git/?p=bertw/homerc.git;a=blob_plain;f=bin/git-edit-index) by Bert Wesarg is a tool which lets you edit files directly from the index, without changing the working copy. The temporary file sits next to the original file, and the index will be updated with the new content (including file mode).

git-build

- `git-build` (http://kgb2.thruhere.net/git/?p=bertw/homerc.git;a=blob_plain;f=bin/git-build) by Bert Wesarg is a tool which records the output from an arbitrary command, but also the state of the working directory (including untracked but not ignored files) the start directory used and the exit status of the command by creating a commit of these and storing it in a special ref `refs/builds/default`. The commit message is the command. The author date is set to the start date and the committer date to that of the end date of the command. The reflog is used to automatically discard old records. My main intention is to use this for builds, such as `./configure && make && make check`. Thats why this is called `git-build`, and also to have a command history by using the reflog. Dumping the environment at command start is on my TODO. Any other hints what should be recored is welcomed.

git-mklinks

- `git-mklinks` (http://kgb2.thruhere.net/git/?p=bertw/homerc.git;a=blob_plain;f=bin/git-mklinks) by Bert Wesarg is a tool which creates hard links for all files with the same content and file mode in the work tree.

git-link

- `git-link` (https://github.com/gvalkov/git-link) constructs repository browser (ie. gitweb) urls of git objects.

git latexdiff

- git latexdiff (https://www.gitorious.org/git-latexdiff) is a simple command that runs latexdiff (http://www.ctan.org/pkg/latexdiff) to generate a visual diff of two versions of a LaTeX file (either two commits, or one commit against the work tree).

git-private-push

- `git-private-push` (https://github.com/csonto/git-private) by Marian Csontos (talk) allows creating a hidden branch in remote repository. Use `git private-push` to backup your work in progress branches, to keep history when frequently rebasing a branch without cluttering everyone's view, or to let others know the branch may get rebased at any time.
  WARNING: Hidden branches offers absolutely no security or privacy. They are just neither displayed nor fetched by default.

gitwin

- `gitwin` (https://itefix.net/gitwin) by Tevfik Karagulle is a commercial packaging of git, OpenSSH, Nginx and many other related tools to make it a ready-to-use solution as a secure git repository on Windows. It supports SSH and HTTP(S) transports as well as gitweb with Highlighter and Gravatar support.

cwgit

- `cwgit` (https://itefix.net/cwgit) by Tevfik Karagulle is a minimal ssh-enabled git client for Windows systems. It is a packaging of git, OpenSSH client, Cygwin and many other related tools to provide a simple and ready-to-use git solution. Simplified BSD license.

See also

- InterfacesFrontendsAndToolsWishlist
- Plumbings
- Aliases - for list of handy tools implemented as Git aliases.

GitDocumentation

From Git SCM Wiki

Git Documentation

- Git glossary (GitWiki version, HTML version (http://www.kernel.org/pub/software/scm/git/docs/gitglossary.html) , source via gitweb (http://www.kernel.org/git/?p=git/git.git;a=blob;hb=HEAD;f=Documentation/glossary-content.txt))
- Git manual page (http://www.kernel.org/pub/software/scm/git/docs) - Online version bundled with Git. (The information displayed when using the command *man git* or *git help*)
- Git User's Manual (http://www.kernel.org/pub/software/scm/git/docs/user-manual.html) - Online version bundled with Git
- A tutorial introduction to git (http://www.kernel.org/pub/software/scm/git/docs/gittutorial.html) and A tutorial introduction to git: part two (http://www.kernel.org/pub/software/scm/git/docs/gittutorial-2.html) - Online version bundled with Git
- Everyday Git, with 20 commands or so (http://www.kernel.org/pub/software/scm/git/docs/giteveryday.html) - Online version bundled with Git
- Git for CVS users (http://www.kernel.org/pub/software/scm/git/docs/gitcvs-migration.html) - Online version bundled with Git. To migrate, consider Interfaces, frontends, and tools#CVS, and especially reposurgeon.
- Git for SVN users
- Git as Subverison Client: git-svn
- Many Howto (http://kernel.org/pub/software/scm/git/docs/howto/) documents come with git as well
- Documentation/technical (http://repo.or.cz/w/git.git?a=tree;f=Documentation/technical;hb=HEAD) for all the technical information relating to Git
- TroubleShooting

Documentation on this Wiki

All documentation residing on this wiki is put in the

- Git glossary (GitWiki version, HTML version (http://www.kernel.org/pub/software/scm/git/docs/gitglossary.html) , source via gitweb (http://www.kernel.org/git/?p=git/git.git;a=blob;hb=HEAD;f=Documentation/glossary.txt))
- Git QuickStart
- GitCheatSheet
- GitTheory
- An overview of the RevisionSpecification used by the `git log` command and several repository visualizers.
- GraftPoint; when and how to use them
- Introduction to git's IndexFile and WhatIsTheIndex
- BranchesInGit is an explanation about how Git understand branches

- Using common CommitMessageConventions
- SubprojectSupport in Git
- Ideas and thoughts on PatchManagement
- HowToWriteTests describes how to write test cases for git
- Using Aliases
- GitTips and GitFaq
- GitWorkflows
- Doing a graft at GraftPoint
- IndexCommandQuickref
- The RevisionSpecification
- Equivalent of *svn switch* for git-svn: GitSvnSwitch - when an SVN repos moves
- A guide to migrating your organisation from SVN to git: SvnMigration
- Installing and developing GitP4
- How to use Textconv feature

User contributed Documentation

- Pro Git Book (http://git-scm.com/book/en/v2) 2nd edition by Scott Chacon and the Git Community, under Creative Commons BY-NC-SA 3.0 (http://creativecommons.org/licenses/by-nc-sa/3.0/us/) license.
  - http://github.com/progit/progit2
- Ry's Git Tutorial (http://rypress.com/tutorials/git/index.html) - A comprehensive introduction to Git with lots of hands-on examples.
- gitready.com (http://gitready.com/) daily tips for the noob to the guru
- Git Reference (http://gitref.org/) , a quick reference for learning and remembering the most important and commonly used Git commands.
- git clone is not resumable - a practical workaround (http://xanjax.org/#gitnote.xml) for unreliable connections / large clones
- Carl Worth's A tour of git: the basics (http://cworth.org/hgbook-git/tour/) Mirror(PDF) (http://git-users.googlegroups.com/web/gitbasic.pdf?gda=TjULfjwAAACRcUWZoNoqHn0MmsB4ogNRkdJPXci4pHF-t7wRb-FXv_7_kSc1FlgquWbpkoDUr3jz_JaIg30gP77VRAtzw-G6) **Recommended**
- John Wiegley's tutorial describing Git From Bottom Up (http://www.newartisans.com/blog_files/git.from.bottom.up.php) Link to PDF, Mirror(PDF) (http://git-users.googlegroups.com/web/gitfrombottomup.pdf?gda=52xMLEQAAACpB0I6kTeoe_gDZwr_Wu3rSzh-BYU3WCJcU16DCLQHrmG1qiJ7UbTIup-M2XPURDRYda-7QfdsxTTX4bqgEJoy1bwkoqS54nAZJJjHLLKkeg) **Recommended**
- My Git workflow (http://osteele.com/archives/2008/05/my-git-workflow) and Commit Policies (http://osteele.com/archives/2008/05/commit-policies) by Oliver Steele
- Git magic (http://www-cs-students.stanford.edu/~blynn/gitmagic/) . A nice use-case driven tutorial. Chinese translation:Git ?? (http://docs.google.com/View?id=dfwthj68_675gz3bw8kj) .
- Shipping quality code with git (http://www.redhatmagazine.com/2008/05/02/shipping-quality-code-with-git/) at Redhat Magazine
- GitCasts (http://www.gitcasts.com/)
- Git for computer scientists (http://eagain.net/articles/git-for-computer-scientists/)
- Git in a Nutshell (http://jonas.iki.fi/git_guides/HTML/git_guide) by Jonas Jusélius is very well done Mirror(HTML) (http://www.chem.helsinki.fi/~jonas/git_guides/)
- An introduction to git-svn (http://utsl.gen.nz/talks/git-svn/intro.html)

- How to use Git and Subversion together (http://www.tfnico.com/presentations/git-and-subversion) , screencasts and blog-posts by Thomas Ferris Nicolaisen
- The Git Papers (http://www.jdl.com/papers) by Jon Loeliger
- Git wikibook: Source Control Management With Git (http://en.wikibooks.org/wiki/Source_Control_Management_With_Git)
- The Kernel Hacker's Git Tutorial (http://linux.yyz.us/git-howto.html) by Jeff Garzik
- Documents and Notes (http://git.rsbx.net/) by Raymond S Brand -Out of date? (edit if not)
- Packaging Software using Git (http://www.golden-gryphon.com/software/misc/packaging.html) is an analysis of schemes people use for packaging.

  See also vcs-pkg (http://vcs-pkg.org/) project, aim of which is to investigate the use of version control for distro package maintenance.

- Git scripts for Xcode users (http://www.jinx.de/teclog/Entries/2009/1/15_Git_and_DMG_build_scripts_for_Xcode.html) Build and distribution scripts for Xcode can now also found on github (http://github.com/jollyjinx/jnxfree) .
- Slides from Git as cvs replacement talk (http://www.breitbandig.de/git/git-as-cvs-en.odp) (for cvs refugees) and Advanced Git talk (http://www.breitbandig.de/git/advanced-git-en.odp) by Thomas Pasch
- Visual git tutorial (http://www.ralfebert.de/blog/tools/visual_git_tutorial_1/) - The visual git tutorial explains how to locally track project files. It shows how to add and commit changes, how to browse the history, revert changes and how to work with tags and branches.
- Mark Lodato made a visual git guide (http://marklodato.github.com/visual-git-guide/) - with source code repository (http://github.com/marklodato/visual-git-guide/) .
- Collaborative git workflow (http://www.eqqon.com/index.php/Collaborative_Github_Workflow) explains how to collaborate with git on an open source project that is (not necessarily) hosted on GitHub.
- HowTo: Installation of Apache WebDAV for git over http (http://wiki.impressive-media.de/doc/git-%C3%BCber-apache-und-webdav-git-%C3%BCber-http)
- HowTo: Installation of git 1.6.5 under Debian Lenny and Ubuntu Jaunty (http://wiki.impressive-media.de/doc/git-165-unter-debian-lenny-bzw-ubuntu-jaunty)
- GIT Tutorial: Maintain your project easily (How to: for beginners) (http://techblog.aasisvinayak.com/git-tutorial-maintain-your-project-easily/)
- HowTo: Installation and tutorial of Braid (submodule alternative) (http://wiki.impressive-media.de/doc/braid-f%C3%BCr-git-installieren-debian-lenny)
- Screencast: Git in Action - Introduction to distributed version control with git (http://www.ralfebert.de/blog/tools/git_screencast/)
- GitGuys Git Tutorials (http://www.gitguys.com/topics)
- 9 101 video tutorials (about 40 mins) by Jeremy Skinner with full quality download available. (http://www.ava.co.uk/GIT)
- Dr. Dobb's Journal - Git Tutorial: Branches and Workflow (http://www.drdobbs.com/architecture-and-design/git-tutorial-branches-and-workflow/240160315) - Includes a Git tutorial with very small steps and a lot of screen shots, by Scott Danzig

Publications

Here are some published resources about Git.

- Pro Git (http://www.apress.com/us/book/9781484200773) by Scott Chacon and Ben Straub (Apress)
- Git Essentials (https://www.packtpub.com/application-development/git-essentials) by Ferdinando Santacroce (Packt Publishing)
- Mastering Git (https://www.packtpub.com/application-development/mastering-git) by Jakub Narębski (Packt Publishing)
- Git for Teams (http://shop.oreilly.com/product/0636920034520.do) by Emma Jane Hogbin Westby (O'Reilly)
- Version Control with Git (http://shop.oreilly.com/product/0636920022862.do) by Jon Loeliger and Matthew McCullough (O'Reilly)
- Professional Git (http://www.wiley.com/WileyCDA/WileyTitle/productCd-111928497X.html) by Brent Laster (Wiley)
- Git in Practice (https://www.manning.com/books/git-in-practice) by Mike McQuaid (Manning Publications)
- Learn Git in a Month of Lunches (https://www.manning.com/books/learn-git-in-a-month-of-lunches) by Rick Umali (Manning Publications)
- Pragmatic Version Control Using Git (https://pragprog.com/book/tsgit/pragmatic-version-control-using-git) by Travis Swicegood (The Pragmatic Bookshelf)
- Pragmatic Guide to Git (https://pragprog.com/book/pg_git/pragmatic-guide-to-git) by Travis Swicegood (The Pragmatic Bookshelf)

Man pages

Each git command is documented in its own man page. If the man pages are installed on your local system running:

```
$ man git-log
```

will give you the man page for the git log command.

Alternatively you can use (especially on systems where there is no man command):

```
$ git help log
```

if both git and the git man pages are installed on your system.

git(7) (http://www.kernel.org/pub/software/scm/git/docs/git.html) serves as an overview of git commands and man pages.

Online versions of the Git man pages corresponding to the latest development version can be found at:

- http://www.kernel.org/pub/software/scm/git/docs/

There also exists a handful of technical documents that describes lowlevel details on how different parts of git operates. These are mostly useful for developers. You can find them in the Documentation/technical (http://repo.or.cz/w/git.git?a=tree;f=Documentation/technical;hb=HEAD) directory in the git source tree.

Git's Documentation To-Do list

If you wish that something were documented better (or at all), or would like to contribute to Git's documentation, you should check the Documentation_To-Do_List to see whether your wish has already been mentioned, or to find something to write about.

Non-english Language Documentation

Chinese

- http://www.bitsun.com/documents/GitUserManualChinese.html translation of official Git user's manual
- http://www.bitsun.com/documents/gittutorcn.htm translation of official tutorials
- http://www.linuxsir.org/bbs/showthread.php?t=281294 or http://www.newsmth.net/bbscon.php?bid=392&id=422519
- http://my.opera.com/dahema/blog/show.dml/171011

French

- Progit (http://git-scm.com/book/fr)
- Manuel git de Beedbox (http://wiki.beedbox.org/fr/documentation/git)

German

- German translation of the Git man pages (http://repo.or.cz/w/gitman-de.git) - currently work in progress.
- Several tutorials and installation howtos for: *DAV, Braid, Gitorious, Gitosis, Submodul, Debian* (http://wiki.impressive-media.de/searchplus/results/taxonomy:418)

Spanish

- http://gitorious.org/projects/curso-git-osluca - some notes that I wrote for a short 1-week course on Git

Japanese

- http://www8.atwiki.jp/git_jp/ - early attempts to translate the manuals. Has translation of the user-manual (completed).

Catalan

- http://ca.wikibooks.org/wiki/Tutorial_de_git - A basic tutorial in Catalan
- http://gabarro.org/wiki/git_mini-howto - A nice mini-howto of distributed workflows
- http://comesfa.poble.cat/Manual:GIT - A complete manual

Korean

- http://namhyung.springnote.com/pages/3132772 - A Korean translation of Git user's manual

(Documentation (http://git.or.cz/#documentation))
