# GIT Summary

**Git** is a [free](http://en.wikipedia.org/wiki/Free_software) [distributed revision control](http://en.wikipedia.org/wiki/Distributed_revision_control), or software [source code](http://en.wikipedia.org/wiki/Source_code) management project with an emphasis on being fast. Every Git working directory is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server.

**GitHub** is a [web-based hosting service](http://en.wikipedia.org/wiki/Shared_web_hosting_service) for projects that use the [Git](http://en.wikipedia.org/wiki/Git_%28software%29) revision control system. It is written in [Ruby on Rails](http://en.wikipedia.org/wiki/Ruby_on_Rails) and offers both commercial plans and free accounts for open source projects.The site provides social networking functionality like feeds, followers and the network graph to display how developers work on their versions of a repository.

Features

* In terms of speed, GIT clearly beats Subversion, Perforce, darcs, [BitKeeper](http://git.or.cz/gitwiki/BitKeeper), [ClearCase](http://git.or.cz/gitwiki/ClearCase) and CVS
* Fully distributed – each developer a local copy of the entire development history, and changes are copied from one such repository to another. Changes are imported as additional development branches, and can be merged in the same way as a locally developed branch.
* Non-linear development - Git supports rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history. Branches merging is common and mostly automated in GIT.
* Simple internal file formats allow for easy manual repair of corrupted repositories.
* Superior compression (fsfs) vs. CVS and Subversion
* Repositories can be published via [HTTP](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol), [FTP](http://en.wikipedia.org/wiki/File_Transfer_Protocol), [rsync](http://en.wikipedia.org/wiki/Rsync), emails holding patches, or a Git protocol over either a plain socket or [ssh](http://en.wikipedia.org/wiki/Secure_Shell). Git also has a CVS server emulation, which enables the use of existing CVS clients and IDE plugins to access Git repositories.
* Subversion and svk repositories can be used directly with git-svn.
* Cryptographic authentication of history. The Git history is stored in such a way that the name of a particular revision (a "commit" in Git terms) depends upon the complete development history leading up to that commit. Once it is published, it is not possible to change the old versions without it being noticed.
* Git has a well-defined model of an incomplete merge, and multiple algorithms for completing it

# Issues

* It is slightly more expensive to examine the change history of a single file than the whole project. To obtain a history of changes affecting a given file, Git must walk the global history and then determine whether each change modified that file. This method of examining history does, however, let Git produce with equal efficiency a single history showing the changes to an arbitrary set of files. For example, a subdirectory of the source tree plus an associated global header file is a very common case.
* Git detects renames while browsing the history of snapshots rather than recording it when making the snapshot.Given a file in revision N, a file of the same name in revision N−1 is its default ancestor. However, when there is no like-named file in revision N−1, Git searches for a file that existed only in revision N−1 and is very similar to the new file. However, it does require more [CPU](http://en.wikipedia.org/wiki/Central_processing_unit)-intensive work every time history is reviewed, and a number of options to adjust the heuristics.
* Manual garbage collection via git-gc --prune can be slow and there is no automated process
* Periodic explicit object packing. Git stores each newly created object as a separate file. Although individually compressed, this takes a great deal of space and is inefficient. This is solved by the use of "packs" that store a large number of objects in a single file (or network byte stream), delta-compressed among themselves. Packs are compressed using the [heuristic](http://en.wikipedia.org/wiki/Heuristic_%28computer_science%29) that files with the same name are probably similar, but do not depend on it for correctness. Newly created objects (newly added history) are still stored singly, and periodic repacking is required to maintain space efficiency. Git does periodic repacking automatically but manual repacking is also possible with the git gc command.

# SVN Import Options for GIT

GitHub can directly import SVN projects. All you’ll need is the repository URL. Importing a subversion repository into git, while not overly difficult, can definitely be streamlined, especially when your goal is to host it on GitHub.

Also, [svn2git](http://github.com/jcoglan/svn2git/tree/master) is designed to provide a complete svn import. Unlike git-svn, it will create proper git tags from your svn “tags”

[git-svn](http://www.kernel.org/pub/software/scm/git/docs/git-svn.html) can be used to import as well. Note that there may be issues if you have branches or tags. If you only have a trunk, like many svn users, this method should work for you without issue.

# Sources

<http://git.or.cz/gitwiki/GitSvnComparsion>

<http://en.wikipedia.org/wiki/Git_(software)>

<http://github.com/blog/156-subversion-importing>