

Git Me Up

(Linus says “You are ugly and stupid!” if you don’t use Git)

Abstract

Version Control Systems (VCS) are at the core of any professional developer’s ‘must have toolkit’! Until recently most VCS’s were centralised solutions (such as CVS and SVN) and hindered the modern “commit often” philosophy as well as preventing off-line commits and truly collaborative work. Solutions such as GIT, which is a distributed VCS, support professional developers using modern agile/pragmatic development techniques.

Author: Nigel Rausch

Nigel Rausch is active in the Brisbane Software community and heavily involved with the organisation of Rails and Software community events. He has 30 years of programming experience and has successfully started, grown and sold a large corporate ISP. In recent years, Nigel has developed primarily for the web, first in PHP, before moving on to become a strong local advocate for Ruby on Rails. He is an agile convert with a passion for quality development practices.

Introduction

History of Version Control System

Version Control Systems are not a new idea and have been used in the software development industry for over 35 years. VCS allows an individual or team of developers to manage working on the same code base.

Why Use Revision Control

VCS is like the undo button in your editor on steroids. It allows the developer to review a project at any point they or another team member has committed. It also allows a team to work on the same code base, even editing the same files.

Another important reason for using VCS, is to allow team members to work on new features within a new branch without affecting the “production” branch.

Examples of Well Known VCS

- Centralised
 - CVS - Concurrent Versions Systems
 - SVN - Subversion
- Distributed
 - Bazaar
 - Git
 - Mercurial

SVN is still one of the most widely used VCS’s. It has many desktop clients and is implemented by default into most IDE’s.

Git Creator

Git was designed and written by Linus Torvalds as a replacement for a BitKeeper which was used in version control of the Linux Kernel. The project is now managed by Junio Hamano.

Who Uses Git?

Git is being used in many projects such as the **Linux Kernel**. The following are some of the high profile projects using Git:

- Amarok
- Android
- Arch Linux
- Aquamacs Emacs
- BlueZ
- Btrfs
- Clojure
- Digg
- DragonFly BSD
- Elinks
- Fedora
- FFmpeg
- **GNOME**
- GPM
- GStreamer
- Gthumb
- **GTK**
- Hurd
- **Linux kernel**
- Linux Mint
- LMMS Music Production Software
- Merb
- One Laptop Per Child
- OpenFOAM
- Parrot virtual machine
- Perl
- **Perl 6**
- Qt
- Rsync
- **Ruby on Rails**
- **Samba**
- SproutCore
- Sugar
- SWI Prolog
- VLC
- Wine
- X.org Server
- x264
- YUI

Performance of Git

Git has the ability to perform many merges very rapidly. Linus gives examples of the VCS used prior to Git for the Linux Kernel. The merge of each single patch took three seconds. This is compared to Git which will merge six patches per second.

How To Use Git

Git has approximately one hundred and fifty commands which are all available from the command line. Forty or so of these are primary “Porcelain” commands and only about eleven commands are required for everyday usage. They are

- git
 - init
 - add
 - commit
 - checkout
 - log
 - branch
 - merge
 - clone
 - pull
 - fetch
 - push

Many IDE’s and desktop GUI’s will handle all these commands for you although it is sometimes useful to use these commands directly to accomplish specific tasks.

git init

This initialises a repository in the current working directory. No files are committed to the repository - only a new .git subdirectory is created containing a base configuration file and git repository related directories.

git clone

Use this command to clone an existing repository into a new directory.

git add

Adds files to the commit index to be tracked ready for a later commit.

git commit

Commits tracked files to the local repository with a commit message. This does not commit to external repositories (see git push).

git checkout

This command changes a current working tree to another branch/revision.

git log

Show commit log history. This command has many options as well as displaying file differences.

git branch

List branches or create a new branch. The developer can create a new branch even when they have uncommitted changes.

git merge

Allows merging of branches or commit points into the current working tree. Normally git automatically merges changes when pulled from external repository.

git fetch

This command fetches the latest changes and saves into another branch/origin. It does merge them into the working tree.

git pull

When run from the local repository, this command fetches and merges the changes from an external repository. This is equivalent to running the individual fetch and merge commands.

git push

This command pushes committed changes to a remote repository. Git supports multiple external repositories.

Graphical User Interfaces

Generally it is easier to use a GUI based git management tool. The Linux installation of git includes a basic GUI interface via the “git gui” command.

Many IDE's and Editors now have plugins or native support, such as:-

- IntelliJ
- Rubymine
- Textmate

Conclusion

Git and other distributed VCS's are still relatively new. The concepts and operation of distributed VCS's are very different to VCS's that people are currently using. The initial high learning curve to convert to Git will be rapidly offset by its power and performance.

Resouces

<http://git-scm.com/>

<http://gitbranch.com>

<http://github.com>