Documentation On Various Version Control Tools

IT350: Software Engineering
Assignment 3

Course Mentor : Biju R Mohan
Course Instructor : Ms. Raksha R Nadigar

Submitted by:

Abhishek S
15IT202 3rd year IT
23rd January, 2018
National Institute Of Technology Karnataka

GIT

Git is one of the most popular version control and source code management system. Designed by the creator of Linux Kernel, this is the most loved tool compared to its competitors, simply because it is easy to understand and the development of the tool is an ongoing process. Many system admins and open-source developers use Git to power their repositories.
It is mainly written in C, Bash Scripting and Perl. It has a **Distributed** repository model. With a concurrency model of “Merge” , Git is supported over various platforms such as POSIX, Windows and OS X and is free to use. It has a storage method known as “Snapshot”. All changes that will be recorded will happen for entire directory trees.
Git mainly uses custom (git), HTTP/HTTPS, rsync, email and bundle protocol for synchronization of changes.

The main features of Git are , it supports :

- Merge file renames
- File renames
- Symbolic links
- Merge tracking
- EOL conversions
- Tags
- International support
- Unicode filename support

Basic Git commands

- git init: turns a directory into an empty Git repo.
- git add: adds files into the staging area of Git.
- git commit: record changes made to the files to a local directory.
- git status: return the current state of the repo.
- git branch: to determine which branch the local repo is on.
- git checkout: to switch branches.

Mercurial

Developed initially for larger development program, Mercurial is a version control system similar to Git. A very fast an efficient application used by Web Developers and designers.
Aside from being very scalable and incredibly fast, its is far simpler compared to Git. There aren't many functions to learn and are less complicated , it is the reason why most system admins prefer Mercurial over Git. It also comes with a web-interface and excessive documentation which makes learning it easier.The development process of the tool is active and the people help keeping it up to date. It has a distributed repository model. Its a merge concurrency mode, supports platforms such as Unix-like OS, Windows, OS X and is free.
Mercurial is written in Python and C making it cross-platform enabled. It uses "Changeset" storage method and the scope of change is tree based. It uses network protocols such as custom over ssh, HTTP, email bundles for data synchronization. The command line interface is one of the best and is very intuitive to operate. It's easy to use and extensions can be written easily.

Mercurial has support for

- Atomic commits
- File renames
- Merge file renames
- Symbolic links
- Signed revision
- Merge tracking
- EOL conversion
- International support
- Unicode filename support

Basic Mercurial commands

- hg init: create a repo.
- hg clone URI: clone a repo to your local machine
- hg status: status of a repo.
- hg heads: displays repository versions.
- hg update -C: disregards any uncommitted changes.
- hg log: shows the repo history.

Why choose Mercurial over Git and SVN?

- It has an easier learning curve and the commands are similar to other VCS.
- Hg also hides many advanced features making the work of a new comer easy.
- If you want to automate tasks then the Hg's python interface is better suited than Git's C.
- Performance can be improved by extensions.
- SVN is centralized and Hg is distributed. If in SVN, the central repo is lost, there is no backup. But in Hg the repo can be recovered from the most recently synced clone.

SVN

SVN, also known as Subversion is the version control system that has the widest adoption. Most form of open-source will use SVN because many other large products such as Ruby, Python, Apache and many more use it too. It is so popular that many clients for SVN are available. If you are on windows, **Tortoise SVN** might be ideal. Maintained by **Apache Software Foundation**, that developers are always working on keeping the tool up to date. It uses a client server repository model, "**Merge lock**" concurrency model, works on Windows, OS X and Unix-like OS and is free of cost.
Written in C, it uses "**Change set**" storage model and "**tree**" model as a scope of change. It uses protocols such as custom(svn), custom over ssh, HTTP and SSL(using WebDAV)

Some features of SVN

- It is centralised
- Supports atomic commits
- File renames
- Merge file renames
- Symbolic links
- Merge tracking
- EOL conversions
- International support
- Unicode filename support

Basic SVN commands

- SVN checkout: create a working copy.
- SVN commit: save changes to the repository.
- SVN list: list directory entries.
- SVN add: add new files to a SVN repo.
- SVN delete: remove file from a repo.
- SVN diff: displays difference between your working copy and the copy in the SVN repo

Why SVN over Git?

- Better IDE integration and GUI. one can see built-in SVN-clients in most IDE.
- It is centralized.
- Full revision history: it maintains versioning for directories, renames and file metadata.
- With SVN you don't need to get a complete repository cloned. You can get any subtree of your repository.
- Built-in authorization and authentication mechanisms
- Locking support: SVN supports "**Merge and Lock**".

Verdict

Git is comes as an excellent overall package for an efficient VCS. All further revisions to the SE project will be made through Git.

Reasons for choosing Git

1. It has a distributed repository system.
2. Many source code management tools such as Bitbucket enhance core Git functionality with pull requests. A pull request is a way to ask another developer to merge one of your branches into their repository.
3. Git can be used for Marketing, product management and for designers. Feature branches lend themselves to rapid prototyping.
4. A larger community base.
5. Security: it is designed specifically to maintain the integrity of source code.
6. Flexibility: A key design objective of Git is the kind of flexibility it offers to support several kinds of nonlinear development workflows and its efficiency in handling both small scale and large scale projects as well as protocols.

Terminologies used

1. Snapshot: indicates that a committed file(s) is stored in its entirety—usually compressed.
2. Changeset: A *changeset*, in this context, indicates that a committed file(s) is stored in the form of a difference between either the previous version or the next.
3. Scope of change: Describes whether changes are recorded for individual *files* or for entire directory *trees*.
4. Concurrency model: describes how changes to the working copy are managed to prevent simultaneous edits from causing nonsensical data in the repository. In a *lock* model, changes are disallowed until the user requests and receives an exclusive lock on the file from the master repository. In a *merge* model, users may freely edit files, but are informed of possible conflicts upon checking their changes into the repository.
5. Atomic commits: refers to a guarantee that all changes are made, or that no change at all will be made.
6. File renames: describes whether a system allows files to be renamed while retaining their version history.

7. Merge file renames: describes whether a system can merge changes made to a file on one branch into the same file that has been renamed on another branch (or vice versa). If the same file has been renamed on both branches then there is a rename conflict that the user must resolve.
8. Symbolic links: describes whether a system allows revision control of symbolic links as with regular files.
9. Merge tracking: describes whether a system remembers what changes have been merged between which branches and only merges the changes that are missing when merging one branch into another.
10. Endofline(EOL) conversions: describes whether a system can adapt the end of line characters for text files such that they match the end of line style for the operating system under which it is used.
11. International support: indicates if the software has support for multiple language environments and operating system
12. Unicode filename support: indicates if the software has support for interoperations under file systems using different character encodings.