Submitted by: B S Suhas

Registration no: 15IT110

Versioning Tools

GIT

Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files. As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Its current maintainer since 2005 is Junio Hamano.

As with most other distributed version control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version tracking abilities, independent of network access or a central server.

Git is free software distributed under the terms of the GNU General Public License version 2.

Advantages of Git

- Distributed model: This means your work is your own. You can let others see only what is necessary. Not everything has to be public. There are other advantages to the distributed model, such as the speed (since almost everything is local) and possibility of working offline
- Branching and merging are easy: They are also cheap (fast and consume very little space) so that one can branch whenever he wants. This means he can sandbox features and ideas till they are ready for the mainstream.
- Workflow is flexible: Compared to Centralized VCS, git has the qualities that allow to choose your own workflow. It can be as simple as a centralised workflow to as hierarchical as the dictator-lieutenant workflow.
- Data integrity is assured: Because git uses SHA1 trees, data corruption due to external reasons can be easily detected.
- Fast: Git is very fast, even when compared to other DVCS, for local as well as network operations
- Staging area: Make sure your commits have logically grouped changes and not everything else you are working on.

- Free: Git being open source, is free to use.

A few commands of git are:

git init

This command turns a directory into an empty Git repository. This is the first step in creating a repository. After running git init, adding and committing files/directories is possible.

git add

Adds files in the to the staging area for Git. Before a file is available to commit to a repository, the file needs to be added to the Git index (staging area). There are a few different ways to use git add, by adding entire directories, specific files, or all unstaged files.

git commit

Record the changes made to the files to a local repository. For easy reference, each commit has a unique ID.

It's best practice to include a message with each commit explaining the changes made in a commit. Adding a commit message helps to find a particular change or understanding the changes.

git status

This command returns the current state of the repository.

git status will return the current working branch. If a file is in the staging area, but not committed, it shows with *git status*. Or, if there are no changes it'll return *nothing to commit, working directory clean.*

git branch

To determine what branch the local repository is on, add a new branch, or delete a branch.

git merge

Integrate branches together. *git merge* combines the changes from one branch to another branch. For example, merge the changes made in a staging branch into the stable branch.

git pull

To get the latest version of a repository run *git pull*. This pulls the changes from the remote repository to the local computer.

Mercurial

Mercurial is a free, distributed source control management tool. It offers one the power to efficiently handle projects of any size while using an intuitive interface. It is easy to use and hard to break, making it ideal for anyone working with versioned files. Mercurial is truly distributed, giving each developer a local copy of the entire development history. This way it works independent of network access or a central server. Committing, branching and merging are fast and cheap. Mercurial's implementation and data structures are designed to be fast. One can generate diffs between revisions, or jump back in time within seconds. Therefore Mercurial is perfectly suitable for large projects. Most of Mercurial is written in Python, with a small part in portable C for performance reasons. Hence Mercurial is platform independent. The functionality of Mercurial can be increased with extensions. Extensions are written in Python and can change the workings of the basic commands, add new commands and access all the core functions of Mercurial. Mercurial supports a set of consistent commands which most users feel comfortable with. Mercurial is free software licensed under the terms of the **GNU General Public License Version 2** or any later version.

Disadvantages of Mercurial

- Combining different features and functionality poses a problem when using different extensions.
- One can undo only the last commit using the rollback command. Additional extensions need to be used for more rollbacks.
- There are no partial checkouts in Mercurial, which is a big limitation for large projects.

SVN

Apache Subversion which is often abbreviated as SVN, is a software versioning and revision control system distributed under an open source license. Subversion was created by CollabNet Inc. in 2000, but now it is developed as a project of the Apache Software Foundation, and as such is part of a rich community of developers and users. It has most of CVS' features. Subversion versions directories as first-class objects, just like files. Copying, Renaming and deleting are versioned operations. Subversion allows arbitrary metadata to be attached to any file or directory. These properties are key-value pairs, and are versioned just like the objects they are attached to. Subversion also provides a way to attach arbitrary key-value properties to a revision. These properties are not versioned, since they attach metadata to the version-space itself, but they can be changed at any time. SVN supports only atomic commits. Hence no part of a commit takes effect until the entire commit has succeeded. Branching and tagging are constant time operations in SVN. Subversion supports locking files so that users can be warned when multiple people try to edit the same file. Subversion notices when a file is executable, and if that file is placed into version control, its executability will be preserved when it it checked out to other locations and thus preserves executable flag. All output of the Subversion command-line client is carefully designed to be both human readable and automatically parseable. The Subversion APIs come with bindings for many programming languages, such as Python, Perl, Java, and Ruby. The Subversion command-line client (svn) offers various ways to resolve conflicting changes, include interactive resolution prompting.

Disadvantages of SVN

- Branches in SVN are cumbersome and hence are used sparingly.
- SVN has a centralized architecture. Thus, if the central repository is lost due to system failure it must be restored from backup and changes since that last backup are likely to be lost. Depending on the backup policies in place this could be several human-weeks worth of work.
- Access control is needed because of its centralized architecture.
- SVN is slow because it may involve network latency.
- The working directory sizes are larger in SVN. One of the reasons is that an SVN working directory always contains two copies of each file: one for the user to actually work with and another hidden in .svn/ to aid operations such as status, diff and commit.