© 2018 IJSRSET | Volume 4 | Issue 8 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section : Engineering and Technology

Review on Version Control with Git

Pranjal Govekar [1], Dr. Shivani Budhkar [2]
[1]Student, MCA, P.E.S's Modern College of Engineering Pune, Maharashtra, India
[2]Associate Professor, MCA, P.E.S's Modern College of Engineering Pune, Maharashtra, India

ABSTRACT

Git is a way more popular than any other VCS (Version Control System). Git comes to existence in 2005 and since then it is getting acquired by more and more people and organization day by day. This paper take look in Git specification comparing with others VCS tools, also taking focus on Git popularity and reasons behind becoming popular VCS tool in market. Here, we present review on Version Control with Git that will help to know about success of Git.

Keywords: Git, VCS (Version Control System)

I. INTRODUCTION

Version Control also known as Source Control or Revision Control is a category of software that helps in managing the changes made in the collection of information, such information can be of any type like documents, computer program, where such tools help in tracking the changes by maintaining it into a special kind of database. It becomes more difficult and important when era of computing began. [14]

Nowadays need of VCS (Version Control System) is, to collaborate on large where many people are involved in keeping track of changes made by same or different in the same or different file. It is essential for every organization having multiple developers working on same project. [14]

As team, it is common to design, develop and deploy multiple versions of same software with different functionalities, where developer simultaneously work on updates and bug fixation hence, to maintain almost same copies of the software is possible with version control system.

A. Evolution of version control system

An easy way of VCS is by storing files in time-stamped directory but this method is error prone as it is easy to forget a working directory and accidentally write to wrong file. Therefore, developer developed local version control system where changes made to the files are stored into the simple database. But then the problem was, what if somebody wants to collaborate with other people working on same system. To overcome this problem CVCSs (Central Version Control systems) were developed. These systems basically have single server as central server which is responsible to hold all version files, from where number of clients can check the files. For many years this was standard for version control. These tools allow multiple developers for simultaneous modification. But again, there is the downside of this system. If central server goes down for an hour nobody can collaborate for that period, and if hard disk becomes corrupted then might have chance to lose everything if proper backup has not done. This is where DVCSs (Distributed Version Control Systems) comes in, where client copies full repository with entire history of the project, so no worries if any server dies and developers

collaborating with each other through that server then it can be restored by simply copying any of the client repository to the server, because every clone is exact full backup of the data. [10]

B. Git Overview

In April 2005, git comes to existence which is free and open source distributed version control system made by Linux creator Linus Torvalds to handle everything from small to very large projects with speed and efficiency.

As all known "Necessity is the mother of inventions" Similarly git invention is done to fulfil Linux Kernel project maintenance. From 1991, developers from different places have started simultaneously to collaborate on linux kernel project. Hence changes made to software where exchange through Patches and Archived files till 2002. In 2002, linux kernel project began to use proprietary DVCS software called BitKeeper for version controlling. Most of Linux Kernel community members were not happy with this move but project leader and core developer adopted BitKeeper. That time license for "community" version of BitKeeper to use by a developer at no cost for open source and free project.

Due to some disputes between community of Linux Kernel and BitMover, In April 2005, BitMover (Original author of BitKeeper) announced that it would stop providing free of charge version of BitKeeper to Linux Community user. That was time where Linux Kernel project need to be maintained by other VCS present in the market but Linus Torvald was not wanted any of them. This made Linux Development community and Linus Torvald to build their own tool based on the lesson they learned using BitKeeper. [2]

II. LITERATURE SURVEY

There are many Version Control software in market based on Client server model and Distributed model where some are Open Source and some are Proprietary, but git earned way more popularity than any other tools.

You can compare interest in Git by time with other VCS tools since git has been launched. The graph below is generated with Google Trends.

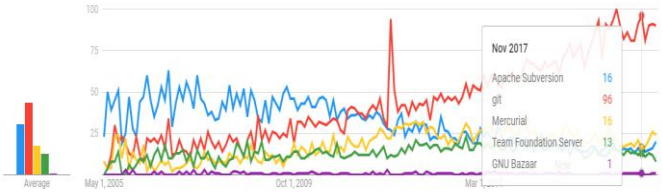

Figure 1: Google trends for VCS

A. Comparison of version control software

a. General Information [13]

Software (Maintainer)	Repository Model	Platform Supported	License	Cost
CVS (The CVS Team)	Client – server	Unix-like, Windows, OS X	GNU GPL	Free
Git (Junio Hamano)	Distributed	POSIX, Windows, OS X	GNU GPL	Free
Mercurial (Matt Mackall)	Distributed	Unix-like, Windows, OS X	GNU GPL	Free
Subversion (Apache Software Foundation)	Client – server	Unix-like, Windows, OS X	Apache	Free

Team Foundation Server (Microsoft)	Client – server, Distributed	Windows, cross-platform via Visual Studio Team Services	Proprietary	Free for up to 5 users in the Visual Studio Team Services or for open source projects through codeplex.com ; else non-free, licensed through MSDN subscription or direct buy.

b. Technical Information

Programming language: The coding language in which the application is being developed

Scope of change: Describes whether changes are recorded for individual files or for entire directory trees.

Network protocols: lists the protocols used for synchronization of changes. [13]

Software	Programming Language	Scope of Change	Network protocols
CVS	C	File	pserver, ssh
Git	C, shell scripts, Perl	Tree	custom (git), custom over ssh, HTTP/HTTPS, rsync, email, bundles
Mercurial	Python, C	Tree	custom over ssh, HTTP, email bundles (with standard plugin)
Subversion	C	Tree	custom (svn), custom over ssh, HTTP and SSL (using WebDAV)
Team Foundation Server	C++ and C#	File and Tree	SOAP over HTTP or HTTPS, Ssh

B. **Version Control System Review**

a. **Concurrent Version System (CVS)**

Very first popular central version control system for collaborative work is Concurrent Versions System (CVS). It has been in the market since 80s.

CVS uses client server architecture, where server is responsible to hold current project and its history. Which allows client connected to server to "Check out" complete copy of project, work on this copy and later "Check in" their changes.

- ✓ Moving or renaming files does not include a version update
- ✓ Security risks from symbolic links to files
- ✓ No atomic operation support, leading to source corruption
- ✓ Branch operations are expensive as it is not designed for long-term branching [16]

b. **Apache Subversion (SVN)**

SVN is abbreviation of Apache Subversion which is originally developed by CollabNet to provide alternative for CVS with some improvements.

Many developers switch to SVN as it is like improved version of CVS

- ✓ Newer system based on CVS
- ✓ Includes atomic operations
- ✓ Insufficient repository management commands
- ✓ Slower comparative speed [16]

c. **Git**

Git is completely different from CVS and SVN. The main purpose of git is to make faster distributed revision control where it was primarily developed for Linux.

Git also comes with wide variety of tools which helps user to navigate through history of system. As Git allows cloning entire repository it is possible to work without internet connection.

- ✓ Full history tree available offline
- ✓ Distributed, peer-to-peer model
- ✓ Not optimal for single developers [16]

d. Mercurial

Mercurial is software belong to same time when Git was released. Mercurial was designed and developed with same motive as Git was, that is to maintain Linux kernel project. But git was selected for maintaining Linux kernel project.

- ✓ Easier to learn than Git
- ✓ No merging of two parents [16]

C. The top 6 easiest to use Version Control System – G2 Crowd

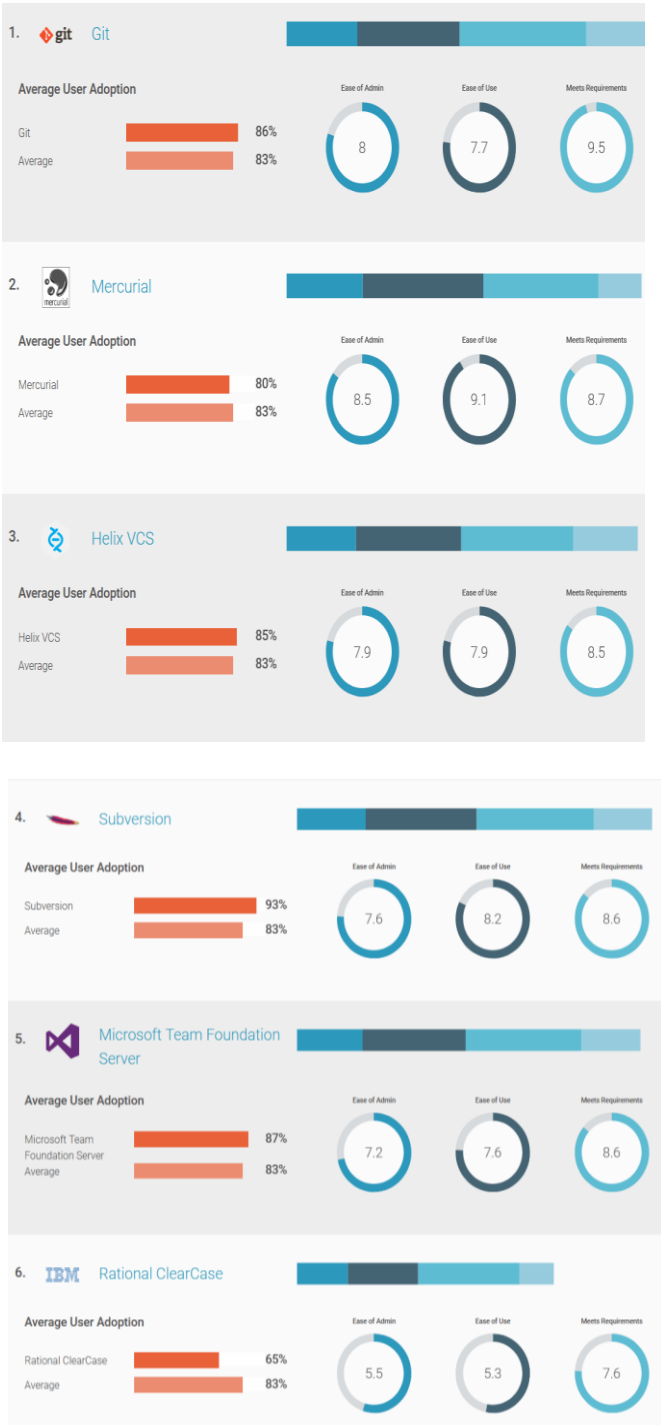

Figure 2: Best Version Control System of 2018 [4]

D. Benchmarks

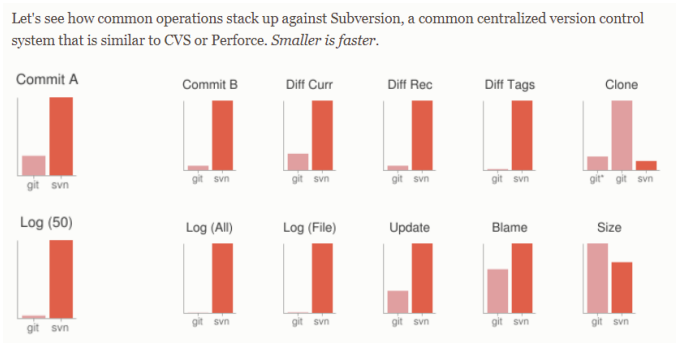

Figure 3: Benchmarks [17]

III. BLOCK DIAGRAM/ ARCHITECTURE AND APPLICATION DESCRIPTION

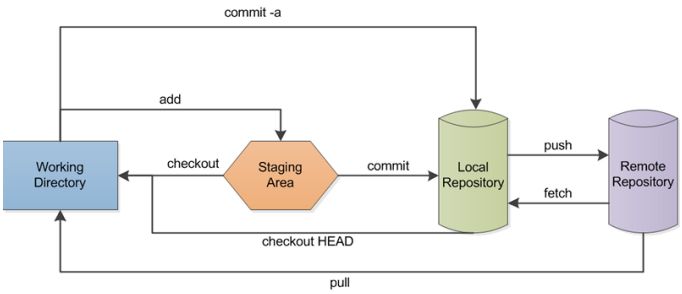

Figure 4: Git Workflow [15]

The above diagram shows how Git tool works to handle collaborative work between multiple developers. Each of them having entire copy of repository as local repository. Also, there is no need of remote repository if you don't want work collaboratively. So private repository are full-featured Git project maintaining all history of versions. [15]

As we can see in above diagram, Git allows us to do changes in working directory where we can store those changes in the staged area for now and commit it to your repo. When you want to share these changes with other then you can push those changes to remote repository.

IV. COMPANIES & PROJECTS USING GIT

As per Git website following are the companies and projects using Git. [18]

1. Google
2. Facebook
3. Microsoft
4. Twitter

5. LinkedIn
6. Netflix
7. Perl
8. PostgreSQL
9. Android
10. Linux
11. Ruby on Rails
12. Qt
13. Gnome
14. Eclipse
15. KDE
16. X

V. ADVANTAGES

1. Location: No central repository. All working copies is a clone of the repository itself. Not dependent on external server for work. No need to install or maintain a server.[1]
2. Development: Branches are easy to make. Commit work to local repository and push changes to others when you feel ready. Every checkout is a copy of the repository. Possibility to clean up your local commits if a mistake is made. It can work in small steps. No need to commit everything at once. [1]
3. Time Saving: Git, however, is lightning fast. [6]
4. Work Offline: With Git, we can work offline as we can have clone of remote repository which means we can do all operation while disconnected from internet. [6]
5. Undo Mistakes: Git allows restoring last commit and also allows restoring deleted commit. [6]
6. Make Useful Commits: With its unique "staging area" concept you can determine exactly which changes shall be included in your next commits, even down to single lines. [6]

VI. DISADVANTAGE

1. If your project contains many large, binary files that cannot be easily compressed the space needed to store all versions of these files can accumulate quickly.
2. If your project has a very long history (50,000 changesets or more), downloading the entire history can take an impractical amount of time and disk space. [8]

VII. CONCLUSION

In recent years, the Git got way more popularity than any other VCS, though they have that much of capability, like Mercurial got released in same time with similar functionality to Git, move over Mercurial is considered to be easier to use then also it is unknown for many people. There are many reasons behind this but main reason could be the services like github repository for Git tool. And comparing other VCS with Git we can found that they don't meet the business requirement as much of Git does.

VIII. REFERENCES

[1]. Carl Fredrik Malmsten. "Evolution of Version Control Systems". Report No. 2010:017. ISSN 1651-4769. P 7.

[2]. Scott Chacon and Ben Straub. "A Short History of Git". Pro Git (Second Edition). P 13 . Apress. 2014. ISBN 978-1-4842-0076-6.

[3]. Reshma Ahmed. "What Is Git ? Explore A Distributed Version Control Tool". https://www.edureka.co/blog/what-is-git/ . Accessed March 6, 2018.

[4]. "The Top 6 Version Control Systems". https://www.g2crowd.com/categories/version-control-systems#highest_rated. Accessed March 9, 2018.

[5]. "Using GIT". https://www.slideshare.net/wocommunity/using-git-13552975. Accessed April 1, 2018.

[6]. " Reasons for Switching to Git". https://www.git-tower.com/blog/8-reasons-for-switching-to-git/. Accessed April 1, 2018.

[7]. "Git". https://en.wikipedia.org/wiki/Git. Accessed April 1, 2018.

[8]. "What is version control: centralized vs. DVCS". https://www.atlassian.com/blog/software-teams/version-control-centralized-dvcs.Accessed April 1, 2018.

[9]. "Git for Version Control". https://courses.cs.washington.edu/courses/cse403/13au/lectures/git.ppt.pdf. Accessed April 1, 2018.

[10]. "A History of Version Control". http://ericsink.com/vcbe/html/history_of_version_control.html. Accessed April 1, 2018.

[11]. "6 Version Control Systems Reviewed". https://www.smashingmagazine.com/2008/09/the-top-7-open-source-version-control-systems/. Accessed April 1, 2018.

[12]. "The Top 6 Easiest to Use". https://www.g2crowd.com/categories/version-control-systems#easiest_to_use. Accessed April 1, 2018.

[13]. "Comparison of version control software". https://en.wikipedia.org/wiki/Comparison_of_version_control_software#History_and_adoption.Accessed April 1, 2018.

[14]. "Version control". https://en.wikipedia.org/wiki/Version_control. Accessed April 1, 2018.

[15]. "HISTORY OF GIT". https://hackaday.com/2017/05/11/history-of-git/. Accessed April 1, 2018.

[16]. "2018 Version Control Software Comparison: SVN, Git, Mercurial". https://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/. Accessed April 1, 2018.

[17]. "fast-version-control". https://git-scm.com/about/small-and-fast. Accessed April 1, 2018.

[18]. "distributed-is-the-new-centralized". https://git-scm.com/. Accessed April 1, 2018