GIT VERSION CONTROL SYSTEM

Shchirov P.D.

Belarusian State University of Informatics and Radioelectronics
Minsk, Republic of Belarus

Perepelitsa L.A. – Lecturer

This paper presents the basic ideas of Git version control system, reveals its role, fundamental advantages and impact on the Software Development Cycle.

The term Git refers to a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. This type of software is the most commonly used version control system. There are some reasons for that. Firstly, it can be successfully applied for tracking changes in any set of files. Secondly, Git allows and encourages to have multiple local branches that can be entirely independent of each other. The creation, merging, and deletion of those lines of development takes seconds. Thirdly, Git makes collaboration easier, as it helps coordinate work among developing source code programmers. It allows changes by multiple people to all be merged into one source.

The main goals of Git version control system are speed, data integrity, and support for distributed, non-linear workflows (branch system). Switching from a centralized version control system to Git version control system changes the whole process of software development. Mainly used for high-level agile software development Git has other systems (CVS, Subversion, Perforce, Bazaar, and so on) as the competitors. The main is Subversion.

Subversion (also known as SVN) is a free centralized management system. Comparing these two systems, it becomes evident that both have different pros and cons. Thus, Git in comparison to SVN is hard to learn and it does not have a friendly UI (user interface). In addition, Git’s speed decreases while it deals with a large number of files. SVN, on the contrary, is much easier to learn, and SVN successfully controls a large number of files. Nevertheless, despite its competitiveness, SVN is constantly losing its popularity. Why? Let's list and analyse the main advantages of Git, thanks to which it tends to replace SVN:

1) Git is undoubtedly rich in its branching capabilities. Branches allow developing features, fixing bugs, or safely experimenting with new ideas in a contained area of a repository. Compared to centralised version control systems, Git branches are very easy to handle. They provide an isolated environment for every change of a codebase, and ensures that the main branch always contains production-quality code. Using branch

system is not only more reliable than directly editing production code, but also provides organisational benefits by representing development work at the same granularity as the agile backlog [1].

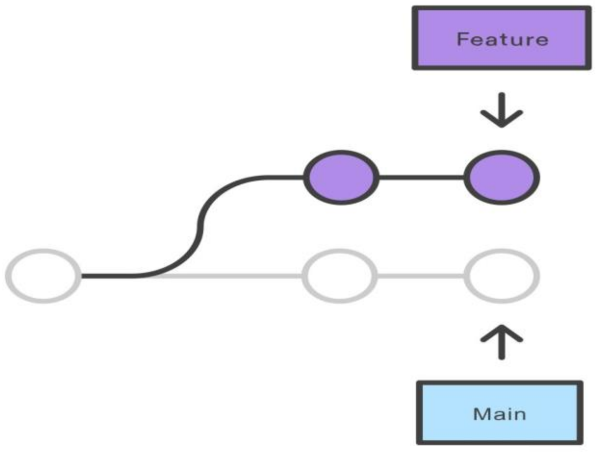

Figure 1 – Visualization of branch

2) Also, one of the main advantage of Git version is that it has a distributed model and all the users can have their copy of the code on their local repository. Instead of a working copy, each developer gets their local repository with a full history of commits (the "git commit" command is used to save changes to the local repository). Having a full local history makes Git fast since it means no need in a network connection to create commits and to inspect previous versions of a file. Distributed development also makes it easier to scale engineering team. If someone breaks the production branch in SVN (Subversion), other developers can’t check their changes until it is fixed. With Git, this kind of blocking does not exist. Similar to a branch system, distributed development creates a more reliable environment. Even if the developers obliterate the repository, it can be simply cloned [2].

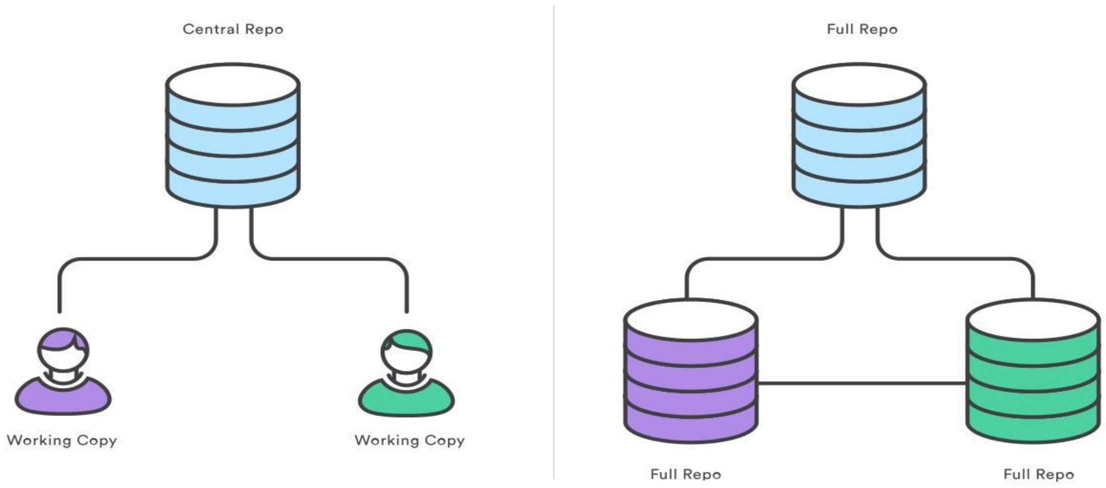

Figure 2 – Visualisation of repositories of centralised and

3) Multiple source code management tools enhance the core Git functionality with pull requests. A pull request is a way for the developers to merge any branch into their repository. This software makes it easier for leaders of the project not only to keep track of changes, but also to let developers initiate discussions around their work before integrating it with the rest of the codebase. When developers get stuck with a hard problem, they can open a pull request to ask for help from the rest of the team. Alternatively, junior developers can be confident that they are not destroying the entire project by treating pull requests as a formal code review.

4) Git copes successfully with continuous integration and delivery environment. Its functionalities allow developers to run scripts when certain technical movements occur inside a repository, letting developers automate deployment to the content core. Any code can be built and deployed from specific branches in different servers. Git can be configured to deploy the most recent commit from the develop branch to a test server whenever a pull request is initialised. Combining this kind of build automation with a peer review leads to the highest possible code confidence and security as it moves from development to the production stage.

5) Git can not be overestimated for product management. The possibility of more frequent releases means more rapid customer feedback and faster updates as a reaction to that feedback. The feature branch

workflow also provides flexibility when priorities change. For instance, there's no problem, being halfway through a release cycle, to postpone one feature instead of another time-critical one. That initial feature can be around in its branch until the software engineer has time to turn to it. Branch facilitates an agile workflow where developers are encouraged to create and share smaller changes more frequently. In turn, changes can get pushed down. The "git push" command is used to push the local repository content to a remote repository. In such a way the deployment pipeline becomes faster than the monolithic releases that are common with centralized version control systems. As the result, the faster release cycle is performed. This functionality makes it easier to manage innovation projects, beta tests, and rapid prototypes, being independent codebases.

6) It is worth mentioning that by choosing Git as a developing version control system, the company attracts progressive software designers seeking deep and extensive knowledge of programming and latest technologies, focusing on constant development.

In conclusion, Git is all about efficiency. It eliminates the software development process from wasting time while passing commits over a network connection and integrating changes in a centralised version control system. Moreover, it gives the possibility to minimise the amount of required man-hours and recruiting junior software developers for a safe programming process. All these factors affect the Git system efficiency. Git version control system gives the possibility for software programmers to substitute unnecessary sets of activities, react to customer complaints immediately and accelerate the process of software development. This system provides a career growth opportunities and extensive implementation. Being agile Git version control system produces a great impact on the Software Development Cycle.

References:

1. Git Magic — [Electronic resource]. — Access mode: https://www.csc.kth.se/utbildning/kth/kurser/DD2385/material/gitmag — Date of access: 21.03.2022.
2. Pro Git — [Electronic resource]. — Access mode: https://git-scm.com/book/ru/v2 — Date of access: 15.12.2021.
3. A Detailed Introduction to Git — [Electronic resource]. — Access mode: https://tproger.ru/translations/beginner-git-cheatshet — Date of access 10.02.2022.
4. Git VS SVN — [Electronic resource]. — Access mode: https://www.perforce.com/blog/vcs/git-vs-svn-what-difference. — Date of access: 18.03.2022.