Git Version control system And its Impact

Shchirov P. D., student gr. 153501

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

*Research Adviser: Ludmila A. Perepelitsa, Lecturer*

This thesis reveals Git's fundamental advantages over SVN and Git's impact on the Software Development Cycle as a whole.

In this thesis, we’ll discuss how Git benefits each aspect of software development. Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. The main goals of this software are speed, data integrity, and support for distributed, non-linear workflows (branch system). Switching from a centralized version control system to Git changes the way any development team or developer by its own creates software. By the end of this article, it should be clear that Git is for high-level agile software development. Subversion is Git's main competitor.

Subversion (also known as "SVN") is a free centralized management system. If to compare Git and SVN, they all have different pros and cons. For example, Git has distributed model (further in the thesis) and in Git all the users have their copy of the code on their local repository. Despite all the advantages of the Git system, it is hard to learn it and Git doesn’t have a good UI (user interface) as compared to SVN. In addition, Git becomes slow 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. Despite its competitiveness, SVN is losing its popularity. Why? Let's list and analyze all the main advantages of Git, thanks to which it replaces SVN:

1) Undoubtedly, the main advantage of Git is its branching capabilities (branches allow developers to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository. Developers always create a branch from an existing branch). Compared to centralized version control systems, git branches are very easy to handle. Feature branches provide an isolated environment for every change to your codebase. When a developer wants to start working on something, no matter how big or small, they create a new branch. This ensures that the main branch always contains production-quality code. Using branches system is not only more reliable than directly editing production code, but it also provides organizational benefits. They let developers represent development work at the same granularity as the agile backlog.

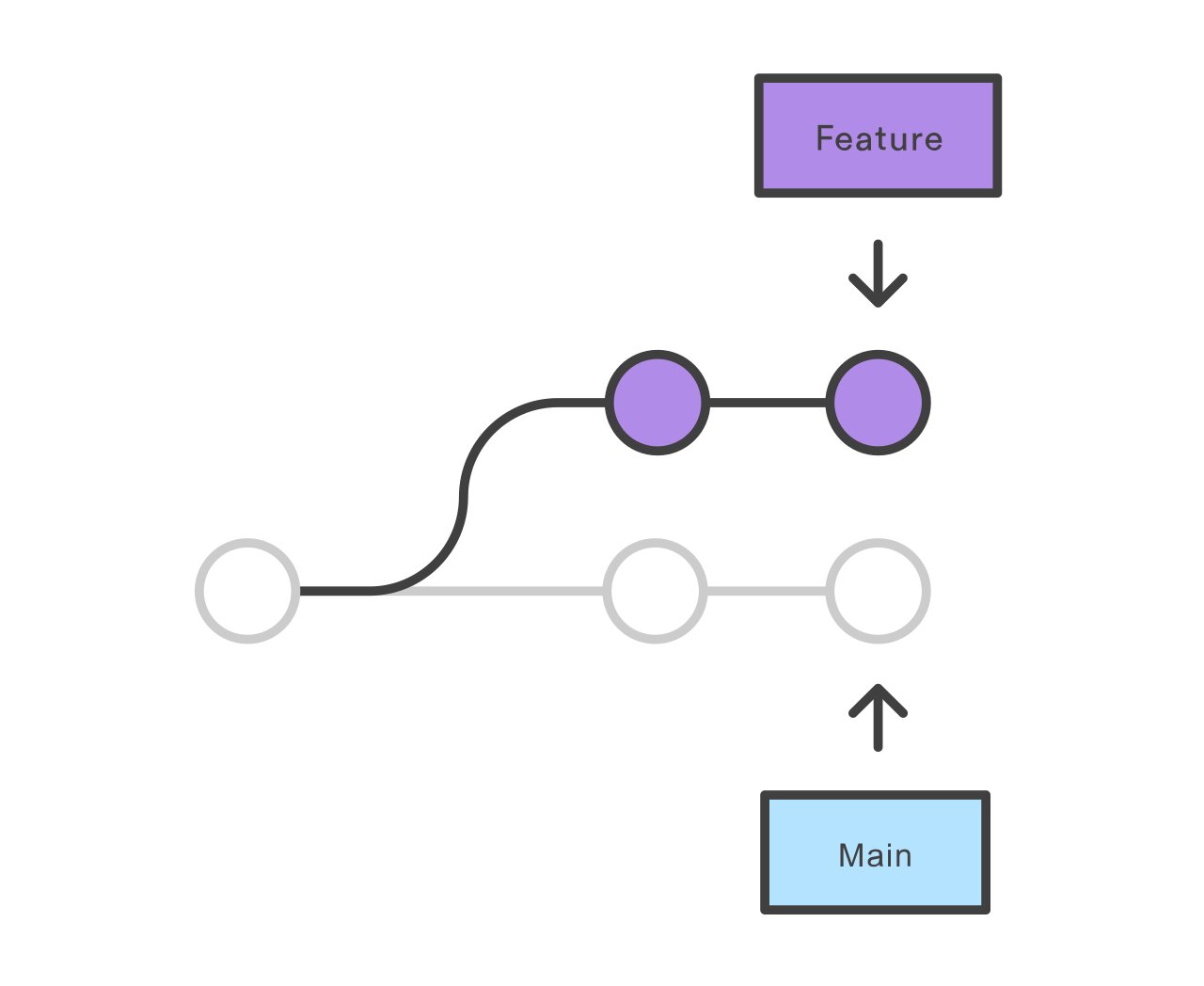


Figure 1 – Visualization of branch system of Git

The main branch (before that master branch) is created by “git init” command and in most cases used as the branch into which other changes are merged. The name master was widely accepted until the end of 2020, the Software Freedom Conservancy and the Git project recognized its shortcomings and began looking for an alternative in June 2020. Almost all projects have followed suit: since the end of 2020, new GitHub repositories have been named main branch by default, and since mid-2021, new GitLab repositories have done the same.

2) Git is a distributed version control system. Instead of a working copy, each developer gets their loсal repository with a full history of commits (the "git commit" command is used to save your changes to the local repository). Having a full local history makes Git fast since it means you don’t need a network connection to create commits and to inspect previous versions of a file. Distributed development also makes it easier to scale your engineering team. If someone breaks the production branch in SVN(Subversion), other developers can’t check in their changes until it’s fixed. With Git, this kind of blocking doesn’t exist. Everybody can continue going about their development in their local repositories. Similar to a branch system, distributed development creates a more reliable environment. Even if the developers obliterates the repository, they cansimply clone someone else’s and start a new one.

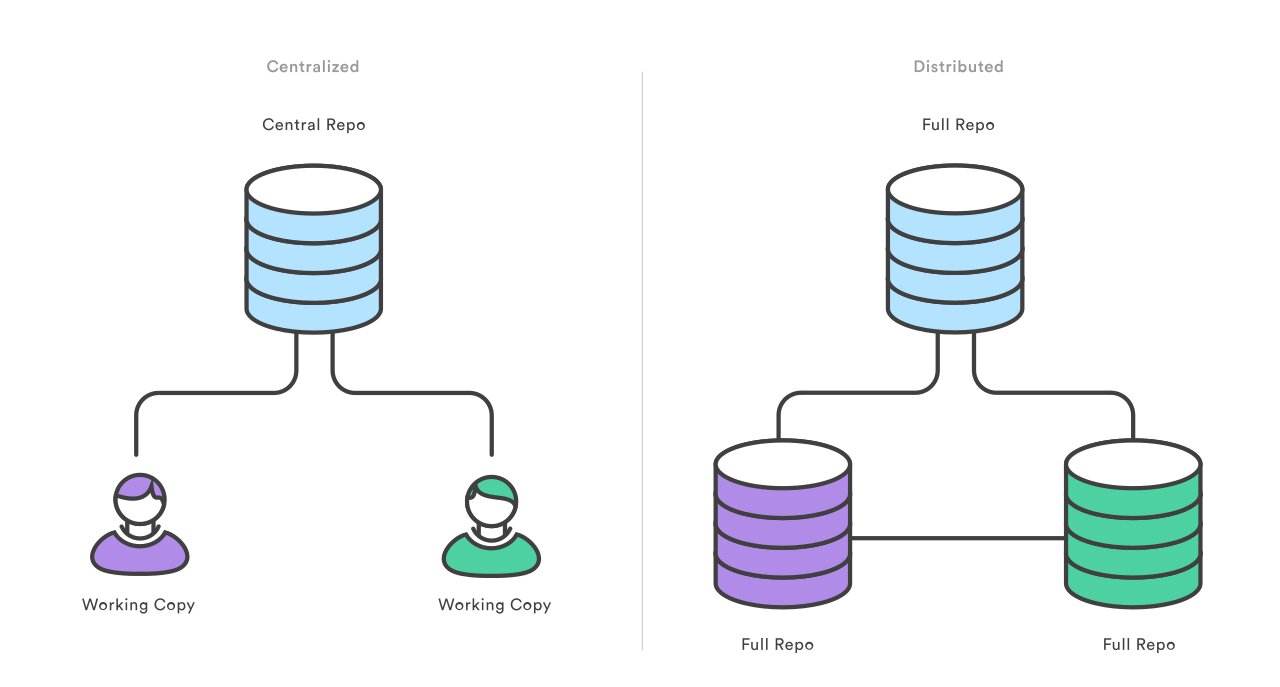


Figure 2 – Visualization of repositories of centralized and distributed systems.

3) Many 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 lets 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) The ultimate result of branch system, distributed development, pull requests is a faster release cycle. This result being a capability of release cycle 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) the deployment pipeline faster than the monolithic releases that are common with centralized version control systems.

5) As you might expect, Git works very well with continuous integration and delivery environment. Git functionalities allow developers to run scripts when certain technical movements occur inside of a repository, which lets 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 initialized. Combining this kind of build automation with a peer review leads to the highest possible code confidence as it moves from development to the production stage, and it secures the movement from the development to the production stage.

Finally, the benefits of Git for product management can’t be overestimated. More frequent releases mean more rapid customer feedback and faster updates in reaction to that feedback. For example, instead of waiting for the next release 6 weeks from now, an organization can push a solution out to customers as quickly as developers can write the code. The feature branch workflow also provides flexibility when priorities change. For instance, being halfway through a release cycle and wishing to postpone one feature instead of another time-critical one, it’s no problem. That initial feature can be around in its branch until the software engineer has time to come back to it. This functionality makes it easier to manage innovation projects, beta tests, and rapid prototypes being independent codebases. Software development workflow determines the employees of the project. It always helps to hire engineers that are familiar with technologies and workflows, but using Git also provides other advantages. Employees are drawn to companies that provide career growth opportunities and understanding how to leverage Git in both large and small organizations is a boon to any programmer. By choosing Git as a developing version control system, the company is deciding to attract forward-looking developers.

In conclusion, Git is all about efficiency. It eliminates everything from wasting time while passing commits over a network connection and integrating changes in a centralized version control system. Moreover, it gives the possibility to minimize the amount of man-hour required for software development and recruiting of junior developers because of a safe programming environment. All of these factors affect the result of the developing process. Efficiencies of the Git system can be extended over other software developers. Usage of Git gives the possibility for software developers to substitute unnecessary sets of development activities and accelerate the process of software development. They let designers test new interfaces on the actual product with little overhead. They let developers react to customer complaints immediately. Being agile is all about finding out what works as quickly as possible, magnifying efforts that are successful, and eliminating ones that aren’t. Git serves to improve the entire software development cycle by ensuring that all developers do their job more efficiently.

**References:**

1. Git [Electronic resource]. — Mode of access: https://en.wikipedia.org/wiki/Git. — Date of access: 20.03.2022.

2. Git Magic [Electronic resource]. — Mode of access: https://www.csc.kth.se/utbildning/kth/kurser/DD2385/material/gitmagic.pdf. — Date of access: 21.03.2022.

3. Pro Git [Electronic resource]. — Mode of access: https://git-scm.com/book/ru/v2. — Date of access: 15.12.2021.

4. A Detailed Introduction to Git [Electronic resource]. — Mode of access: https://tproger.ru/translations/beginner-git-cheatsheet/. — Date of access 10.02.2022.

5. Git VS SVN [Electronic resource]. — Mode of access: https://www.perforce.com/blog/vcs/git-vs-svn-what-difference. — Date of access: 18.03.2022.