IENT301 ASSIGNMENT

Version Control Systems

225025426 Liyema Booi

225097524 Zanele Mndaweni

225008580 Makhapongo Baloi

MAY 24, 2024

NELSON MANDELA UNIVERSITY
[Company address]

Integrated Enterprise Systems- Version Control Systems

How Does Git work?

Git is distributed version control system used to track code changes and allows users to manage their project using simple commands.

To start using git in a specific project, you have to create a new Git repository in your project folder. This repository can then be stored locally or on a website such as GitHub. Commits are basically save points in the project's history. They represent changes made to the project at a particular point in time. Each commit is identified by a unique SHA-1 hash, allowing Git to track and retrieve changes efficiently. When using git, it is important to be aware about the three spaces inside Git, Working directory, staging directory, committed. The working directory represents files in your project folder, files can be edited but changes made in this stage are not tracked and not ready to be committed. The staging directory can be seen as a rough draft for files ready to be committed that were added from the working directory, in this stage you can still modify files and add them again or remove files. The committed stage is the final stage where committed files are kept, snapshots from the staging directory are stored in the commit history.

Benefits of using Git

- Open Source and free

  Git is available freely on the internet and can be used to manage big projects without having to spend any money. Git is also open source allowing users access to it source code and they can modify it to meet their needs

- Branching and merging

  These two features set Git apart from other source code management systems. With Git, users can have multiple local branches that can be completely independent of each other. These branches are lightweight and easy to merge.

- Fast and small

  There is no need to connect to a central server, users can perform all operations on their locally stored copy of the directory, making performing these operations fast.

- Security

  Git makes use of SHA1 algorithm to secure and ensure the integrity of the data saved within the Git repository.

Drawbacks of using Git

- Complexity for New Users

  Git's vast feature set and flexibility can be quite overwhelming for beginners. The learning curve can be steep, especially for those who are not accustomed to using command-line interfaces.

- Storage Inefficiency for Large Binary Files

  Git is not optimized for handling large binary files, such as media files or compiled binaries. Such files can cause repositories to bloat and become difficult to maintain.

- Working in the wrong branch

  With the benefit of using branches comes the possibility of making the mistake of working in the wrong branch having forgotten to perform the checkout command (used to switch between branches) or working without realizing the repository had been left in a branch rather then the master.

How Does Subversion (SVN) work?

Subversion (SVN) is a centralised version control system that's makes use of central database, designed to manage changes to files and directories over time. Subversion has three main core functions in the way it works, the repository and working copy, updating and committing and versioning. The repository is the central database which is typically a file server that acts as the master copy for all versions of the files and directories in a project. The working copy is the local copy of the repository stored on a user's local machine where they are able to make modifications to files and edit code. The update function in Subversion aligns the working copy with the latest changes from the repository. The commit function sends the changes that are in the working copy on your local machine to the repository essentially creating a new version. The way that this is possible is through the versioning function in Subversion that tracks every change made to files and folders, this allows users to go back to any previous versions, make comparisons and to view the modification history. Users use URLs to identify different file and directory versions.

Benefits of using Subversion

- Easier approach to getting things done

  The path to creating branches and merging is a relatively short and easy to follow. This makes subversion easier to learn and require less learning to get started which may make it more appealing to non-technical contributors.

- Allows for the storage of large binary files

  For teams that require storage for binary files that will change frequently, this can be done without much worry about the exponential storage increases with every change made. For teams that require this, this feature can be quite helpful for some workflows and version control use cases.

Drawbacks of using Subversion

- Limited performance and scalability issues

  SVN is a central single-server system which makes scalability a problem and significantly lowers performance. This is especially noticeable when working with large repositories and file sizes. There isn't really an exact benchmark data where limits of Subversion begin to show, this makes the planning and scaling very difficult.

- Minimal Merging Capabilities

  Subversion fails to handle cases where two or more developers are working on the same code base that need to be merged. Developers often waste lots of time having to resolve conflicts manually.

- Complex branching model

  Branches allow users to work on multiple versions of code at the same time. However, in SVN, since branches are identified by naming conventions, if changes are made need to be made across numerous branches, this can prove to be quite complex and because of this, errors can occur.

Differences/reasons to choose Subversion or Git

Reasons to choose Subversion over Git include:

- Subversion forces users to collaborate because unlike with Git which is distributed, subversion is centralised meaning whenever a commit is made, it is made to a central server. This forces collaborating developers to consider each other's edits sooner rather then later which can save the hassle of having to deal with hard-to-resolve issues down the line. SVN allows developers to check on the progress of all other developers branches and warn them of potential conflicts ahead of time. With Git on the other hand, users won't know about these conflicts until the developer decides to make their changes public.
- With Git its quite easy to wind up working in the wrong branch. With Subversion on the other hand, all files are laid out in front of you so it makes it very easy to see from the path which branch you are working in and makes it so that this mistake isn't made so easily
- It is easier to check out subdirectory with SVN then it is to do with Git which can be a challenging task

Reasons to choose Git over Subversion include:

- Git allows user to work on their local machine and offline while SVN users are required to be connected to the repository server which essentially destroys the possibility of working offline.
- Git lowers the risk of completely losing the main repository as it uses a distributed model of version control. This risk is reduced as collaborating developers have to clone the main repository. While with SVN, the centralized model creates a potential single point of failure should anything happen to the main repository.
- In terms of merging and conflict resolution, Git by far is better, as it was designed for an open source setting where lots of contributors may be working on different parts of the

code base. Git has built up a robust system for resolving merge conflicts, making the process smoother and more manageable which allows for this type of collaboration.

Demonstration of Using Git

Demonstration of Using Subversion

Short Reflection

References

AVIAD. (2012, February 22). *How-To Geek*. Retrieved from Version Tracking With Subversion (SVN) For Beginners: https://www.howtogeek.com/66731/version-tracking-with-subversion-svn-for-beginners/

Blog, P. (2018, August 16). *Programmer's blog*. Retrieved from Advantages of SVN over Git: https://blog.programster.org/advantages-of-svn-over-git#:~:text=Advantages%20of%20SVN%20over%20Git%201%20Forced%20Collaboration,5%20Easier%20Merging%20...%206%20User%20Permissions%20

Git. (n.d.). *Git*. Retrieved from About: https://git-scm.com/about/distributed

Marijan, B. (2021, September 16). *PhoenixNAP*. Retrieved from How Does Git Work?: https://phoenixnap.com/kb/how-git-works

Martin, D. (2021, July 3). *Bodhizazen*. Retrieved from Git: Advantages and Disadvantages: https://bodhizazen.net/git-advantages-and-disadvantages/

Sharma, D. K. (2023, August 12). *K21 Academy*. Retrieved from Git | Version Control System | Git Workflow | Advantages: https://k21academy.com/devops-foundation/git-overview-workflow-advantages/

Stickman, N. (2022, May 13). *Akamai*. Retrieved from SVN vs Git: Which Version Control System Should You Use?: https://www.linode.com/docs/guides/svn-vs-git/

SVNbook. (n.d.). *SVN Book*. Retrieved from Version Control the Subversion Way: https://svnbook.red-bean.com/en/1.6/svn.basic.in-action.html

Tortoise SVN. (n.d.). *Tortoise SVN*. Retrieved from Basic Concepts: https://tortoisesvn.net/docs/release/TortoiseSVN_en/tsvn-qs-basics.html