**SVN vs GIT Comparison Documentation Release 0.0.1**

**25 Mar 2018**

Contents

[**SVN vs GIT – Technical Aspects** 3](#_Toc509745782)

[**Licensing** 3](#_Toc509745783)

[**Clients** 3](#_Toc509745784)

[**Type** 3](#_Toc509745785)

[**Tree Model** 3](#_Toc509745786)

[**Branching** 3](#_Toc509745787)

[**Merge Process** 4](#_Toc509745788)

[**Changes Storage** 4](#_Toc509745789)

[**Commit Structure** 4](#_Toc509745790)

[**Network Protocol** 4](#_Toc509745791)

[**Source Code Size Maximum** 4](#_Toc509745792)

[**Maintainer** 4](#_Toc509745793)

[**User Interface** 4](#_Toc509745794)

[**SVN vs GIT Practical Aspects** 5](#_Toc509745795)

[**Centralized Vs Distributed** 5](#_Toc509745796)

[**Centralized** 5](#_Toc509745797)

[**Distributed** 5](#_Toc509745798)

[**Space Consumption** 6](#_Toc509745799)

[**Repository Structure** 6](#_Toc509745800)

[**Branches** 6](#_Toc509745801)

[**Tag** 7](#_Toc509745802)

[**Security** 7](#_Toc509745803)

[**Commits** 7](#_Toc509745804)

[**Workflow** 8](#_Toc509745805)

[**References** 8](#_Toc509745806)

# **SVN vs GIT – Technical Aspects**

## **Licensing**

* Both are opensource

## **Clients**

* SVN – Has the command line interface and some GUI clients like Tortoise SVN and Visual SVN.
* GIT - Also has the command line interface and lots of GUI clients like Git Extensions, Source Tree, GitHub, Bit bucket, Git Kraken and Tortoise Git.

## **Type**

* SVN - CVCS (Centralized Version Control System) - There is just one central repository where all operations are performed and from where every user gets the files. This allows a top-down access control and change locking features.
* GIT - DCVCS (Decentralized Version Control System) - Every user gets a full copy of the repository on its machine therefore, almost every operation is performed locally. There can be a central remote repository for collaboration to and from where the users will push and pull modifications. It also allows multiple remote repositories.

## **Tree Model**

* SVN - Linear Tree (aka Stream) - SVN uses a linear tree as it’s data structure which means that every single commit will be represented as a node on that tree (or line). Every merge from other branches results on a single new commit on the tree like any other modification.
* GIT – DAG (Directed acyclic graph)- The GIT tree has a Graph where every commit (graph node) has reference to, at least, one parent commit. This is especially helpful when you have branches and merges happening on your repository. this structure allows the user to find where each modification happened, and in which branch it occurred.

## **Branching**

* SVN - Branches on SVN are treated as normal folders on the repository. As you have only the central repository every branch is automatically remote and public (every user with access to the repository will have access to the branch). Branches on SVN are basically copies of the affected folders and this copy can be made locally and then committed to the central repo or remotely.
* GIT - Branches are pointers to specific commits on the repository. They are created locally and can be pushed/pulled to/from a remote repo or not.

## **Merge Process**

* SVN
* GIT

## **Changes Storage**

* SVN- Deltas.
* GIT -Files Snapshots.

## **Commit Structure**

* SVN
* GIT

## **Network Protocol**

* SVN- custom (SVN), custom over SSH, [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) and SSL.
* GIT-custom (git), custom over SSH, [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)/[HTTPS](https://en.wikipedia.org/wiki/HTTP_Secure).

## **Source Code Size Maximum**

* SVN - 41mb.
* GIT – 23mb.

## **Maintainer**

* SVN- Apache Software Foundation.
* GIT- Junio Hamano.

## **User Interface**

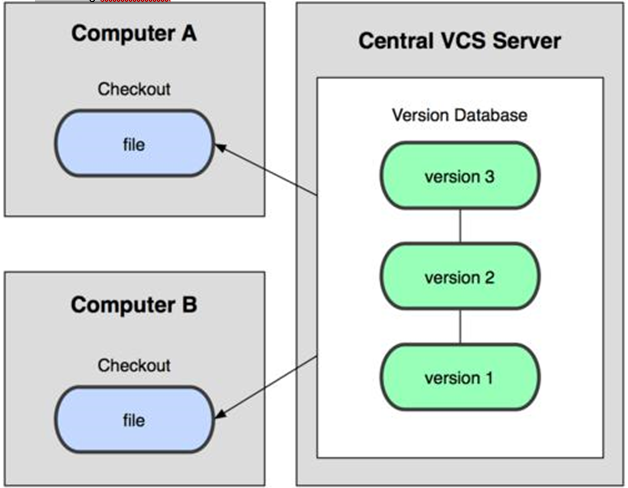
* Subversion's UI is more mature than Git's
* Git use command line

# **SVN vs GIT Practical Aspects**

## **Centralized Vs Distributed**

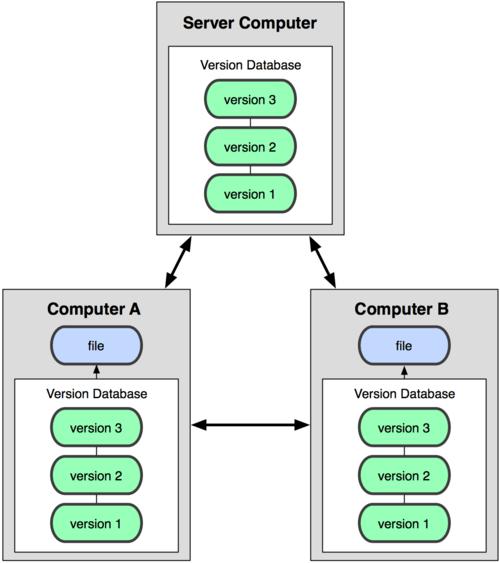
### **Centralized**

* The repository is located on a remote server where all operations are performed.
* When the user checks out a directory from the repository it gets only the files, with minimal versioning information



### **Distributed**

* Every developer got a full copy of the repository stored locally where most of the operations are performed.
* The local repository can point to one or more remote repositories where the work can be shared with other developers.



## **Space Consumption**

* Knowing that when one developer clone a GIT repository it actually gets a full copy of the repository, a concern may arise in terms os local disk space consumption. Well, while you may have more data stored in your disk all the GIT database is compressed and highly optimized Which means that the local repository will occupy less space in disk than one may think.
* When you checkout a folder from a subversion repository you end up with 2 uncompressed versions of the folder locally, your working copy and the snapshot of the last revision, the one you checked out, of the folder. SVN does that to allow the developer to make diffs between the working copy and the last revision locally without needing to go to the server.
* We can compare the two options to see which has the biggest size, that can be done importing an existing SVN repository to a GIT repository using the GIT-SVN tool and then compares sizes of the folders

## **Repository Structure**

* On subversion you have a tree model with just one branch where the revisions are stored sequentially.
* GIT uses a Graph structure where each commit is a node that knows every of its parents. The commit is the central piece on the git repository

## **Branches**

* For SVN branches are just another directory on the repository with its own history information. You can merge revisions between branches, but they are still treated as separated entities. Because of the centralized nature of SVN all branches are remote and public. The administrator can restrict the access to the branch to a certain group of developers, but still, the branch is located on the server.
* On git a branch is basically a named pointer for a specific commit on the tree, because of that nature branches are very cheap for the system. Branches can be only local or can be published to a remote repository for collaboration with other developers. Every new copy of the repository, cloned from a remote repository, is treated as a new local branch of the main repository, that means that every developer repository is an isolated entity and the work on that local repository does not interfere with other developers work until it is pushed to the shared repository and pulled by the others.

## **Tag**

* Tags and branches are very similar in both systems. For SVN a TAG is just another folder in the main repository, that shouldn’t receive any update and remain static, that can be achieved using access restrictions on the new tag.
* On git a tag is a static pointer to a specific commit. Unlike the branch pointer, the tag pointer is static, which means that it cannot be moved to another commit.

## **Security**

* SVN offers a more granular security control, you can have different access rights for each folder on your repository, remember that tags and branches are also folders, which if perfect for a top-down management.
* **Use centralized authentication via AD(**Active Directory**) or LDAP(**Lightweight Directory Access Protocol**)**
* **Build your SVN server as a virtual machine (VM)**
* **Use SVN replication to provide high availability and fault tolerance**
* GIT security is repository-wise, you can allow or revoke access of a user to the whole repository only. You can apply read-only access to certain branches also, but for the whole repository, never to a single folder or file.
* Git uses a common cryptographic hash function called secure hash function (SHA1), to name and identify objects within its database. Use SSH and HTTPS

## **Commits**

* Every commit on SVN creates a new revision to the remote repository, that means that to be able to commit some changes to the code the developer need to have access to the main repository.
* On git the commit is happening locally, so the developer does not need to have access to the remote. The work can be committed locally and then when the developer gains access to the remote repository, it can push all the commits at once.

## **Workflow**

SVN:

* The trunk directory represents the latest stable release of a project.
* Active feature work is developed within subdirectories under branches.
* When a feature is finished, the feature directory is merged into trunk and removed.

GIT:

* A Git repository stores the full history of all of its branches and tags within the *.git*
* The latest stable release is contained within the master
* Active feature work is developed in separate branches.
* When a feature is finished, the feature branch is merged into master and deleted.

## **References**

* Progit book – Scott Chacon and Ben Straub
* Version control with subversion – Ben Collins-Sussman, Brian W.Fitzpatrik and C.Michel Pilato
* Stack Overflow – Why git is better than SVN? How git branch works?