



November 14, 2018

## Git vs. SVN – What Is The Difference?

GIT AT SCALE

Git and Subversion are two different types of version control systems.

### Git vs. SVN: Pros and Cons

Git is a distributed version control system. SVN is a centralized version control system. That leads to several key differences. If you're considering switching from SVN to Git, you'll want to take these into account.

#### 1. Server Architecture

Git software is installed on a workstation and acts as a client and a server. With SVN, there is a separate server and client. With Git, every developer has a local copy of the full version history of the project on their individual machine. With SVN, only the files a developer is working on are kept on the local machine, and the developer must be online, working with the server.

SVN users check out files and commit changes back to the server. Git changes happen locally. The advantage is that the developer doesn't have to be connected all the time. Once all the files are downloaded to the developer's workstation, local operations are faster.

In the past, Git developers each having a copy of the full version history meant they could easily share changes before pushing them to a central repository. Now all the sharing is done in central repositories, like a GitHub. And, in today's world, enterprises have projects that span multiple repositories that include large binary files. As projects grow, [storing locally is not really realistic or desirable](#).

#### SVN Wins for Storing Binary Files

When it comes to Git vs. SVN performance, the client-server model of SVN outperforms with larger files and codebases.

Storing large binary files in Git can slow down the very advantages they claim to have over SVN. Developers spend time waiting to check out the full repository onto their computer. Every time a large file is changed and committed, Git repositories grow exponentially.

Of course, there are workarounds for storing your binaries in Git, such as [Git LFS](#). But still, every developer action leads to a mountain of change history data. This is going to slow down performance.

In SVN, only the working tree and the latest changes are checked out onto local machines. Check outs take less time in SVN when there are a lot of changes to binary files.

#### 2. Branching

One of the most common complaints about SVN is its tedious branching and complicated merging model. It can be time consuming. SVN branches are created as directories inside a repository. This directory structure is the core pain point with [SVN branching](#). When the branch is ready, you commit back to the trunk.

Of course, you're not the only one merging changes. Your version of the trunk might not reflect developers' branches. This means conflicts, missing files, and jumbled changes riddle your branch.

#### Git Wins For Its Branching Model

Developers like Git because of its effective branching model. In Git, branches are only references to a certain commit, making them lightweight yet powerful. Git allows you to create, delete, and change a branch anytime without affecting the commits. If you need to test out a new feature or you find a bug, you can make a branch, make the changes, push the commit to the central repo, and then delete the branch.

#### 3. Access Controls

Access control is another key in the Git vs. SVN debate.

#### Toss Up — Git or SVN Based on Need

Both systems take different approaches when it comes to permissions and access. By default, Git assumes that all the contributors have the same permissions. On the other hand, SVN has a more granular access control system. SVN allows you to set permissions for individual files and directories. This can be useful in large organizations where different teams have different levels of access to the codebase.

SVN allows you to specify read and write access controls per file level and per directory level.

#### 4. Security

With SVN, the repository's change history is pretty consistent. To make any change to the repository's history, you need access to the central server.

Git's distributed nature allows anyone to change any part of their local repository's history. Although pushing a changed history is heavily discouraged, it can happen. This causes problems if other developers are relying on particular changes.

In Git, the complete history of the repository is "backed up" each time a developer clones it to their computer. This natural backup mechanism is useless if neglected.

#### Toss Up — SVN or Git, As Long as You Back Up

Making regular backups is highly encouraged in both Git and SVN. You do not want to be on the receiving end of a server crash without a recent copy of your shared server.

#### 5. Git SVN Storage Requirements

As the arguments for Git or SVN rage on, you may notice that when it comes to storage — there is no difference. Surprisingly, the disk space usage is equal for both Git and SVN repositories. This is true even though SVN tracks changes on a file level and Git tracks changes in the repository level.

#### SVN Wins (Only If You Have Large Binary Files)

Then again, storing large binary files in SVN would be much smaller than their Git equivalent.

#### 6. Usability

Usability can be a deciding factor in the Git vs. SVN debate.

#### SVN Wins (Easier to Learn)

SVN often considered easier to learn. This is especially true for non-technical users. They are able to catch on to common operations quickly.

Although both use the command line as the primary user interface, syntax in Git can overwhelm beginners. SVN is more readily used by non-programmers who want to version non-code assets. Learn more about [SVN commands](#) and see how they stack up against other version control systems.

### Git or SVN — What Should You Choose?

While SVN has some key strengths over Git, its popularity is waning. And many teams are looking to make a switch from SVN. You could migrate from SVN to Git (and use a Git SVN pull command to pull in commits).

There are a couple of reasons for making the switch from SVN. It isn't a great tool for automation and DevOps. And that it no longer has a vibrant community supporting it.

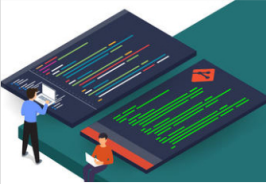
Git seems like a no-brainer when looking for a more modern and supported system to replace SVN. Especially since it is also open source: you won't have to budget for something that you aren't paying for today.

However, if you are working with large files, have large global teams, security concerns, or other "at scale" challenges, Git may create more problems than it solves.

You can get the best of both worlds with Perforce [Git tools](#) (Helix4Git and Helix TeamHub). Contact us to learn more.

[CONTACT US](#)

### Recommended Posts



October 17, 2019  
**How to Improve Your Git Code Review Workflow**  
GIT AT SCALE, CODING BEST PRACTICES



October 7, 2019  
**How to Use Git Shallow Clone to Improve Performance**  
GIT AT SCALE, VERSION CONTROL



September 25, 2019  
**How Secure Is Git?**  
GIT AT SCALE, SECURITY & COMPLIANCE

#### PRODUCTS >

VERSION CONTROL SYSTEM  
Helix Core  
Helix Core Apps  
Helix Plugins

#### SOLUTIONS >

BY NEED  
Version Control  
Repository Management  
Developer Collaboration

#### RESOURCES >

Papers & Videos  
Events & Webinars  
Recorded Webinars  
Blog

#### ABOUT >

Our Team  
Our Culture  
Careers  
Press

ENTERPRISE AGILE PLANNING

Hansoft

DEV COLLABORATION

Helix Swarm

Helix4Git

DEVELOPMENT LIFECYCLE MANAGEMENT

Helix ALM Suite:

- Helix RM

- Helix IM

- Helix TCM

Surround SCM

STATIC ANALYSIS

Helix QAC

Klocwork

...

Application Lifecycle Management

Agile Project Management

Backlog Management

Project Portfolio Management

Audit & Compliance

Git At Scale

DevOps

Performance & Scalability

IP Protection

Static Analysis

BY INDUSTRY

Game Development

Life Sciences

Software

Automotive

Embedded Systems

Government

Finance

Energy & Utilities

Aerospace & Defense

Subscribe

SUPPORT >

Documentation

Request Support

Video Tutorials

Perforce Knowledgebase

Training

Consulting

Support Plans

Maintenance Support

EOL Schedule

Release Notes

Download

Open Case

CUSTOMERS >

Case Studies

Contact

PARTNERS >

Integrations

Resellers

Partner Portal

QUICK LINKS >

Try For Free

Helix Core Demo

Helix ALM Demo

Hansoft Demo

Subscription Center

Customer Support Login

Licensing Requests

Educational Licenses

How To Buy

PERFORCE

Copyright © 2019 Perforce Software, Inc. All rights reserved. | Sitemap | Terms of Use | Privacy Policy

