How to Choose the Right Version Control Software

Version control protects source code and facilitates collaboration—a single source of truth for a project's code. But which version control software should you buy? Well, that depends. Keep reading to learn more about the selection process.

ASSEMBLA

What to Consider

At a high level, you'll need to think about your development process, the size and types of files your team works with, and organizational priorities (e.g., security vs. speed). Here are a few considerations that must be made when selecting a VCS tool.

Collaboration/ease of use

Collaboration is at the core of version control. Being able to facilitate collaboration for your team should be a key component when choosing a version control system. As we discuss the different types of version control, you'll see that collaboration can be facilitated in many ways.

Security

Some version control systems are better than others at inherent security features. A classic crux of this debate when it comes to version control is distributed versus centralized version control systems. We'll discuss this more in depth later on. For some teams, it is necessary to have access control down to the file level and not just the repository or space. The level of granularity with which you can control these factors can vary across different version control systems.

Security is one key consideration when choosing the best version control tool for your team.

File type and data size

Some version control platforms are better suited for handling large, binary files than others. If the types of projects that your work focuses on require a number of binary files (e.g., graphic assets and text files), it's necessary to make sure your version control system works as well as possible with those types of files.

Your team's expertise/bandwidth to handle setting up and maintaining

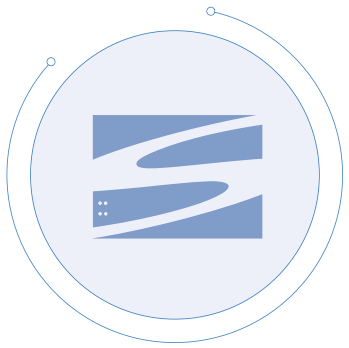
Regardless of the above factors, how a version control system molds to your team's expertise is an important question that will help you better calculate cost.

Types of Version Control Software

Before we discuss individual version control systems, it's important to discuss one of the primary categories used to describe them: distributed versus centralized systems. Neither of these categories is inherently better than the other, but depending on business needs, one may be preferable in any given organization.

Centralized Version Control

Centralized version control systems have one server containing all of the relevant data and files to a repository. For developers to work on part of a project in a centralized system, they must "check out" the file, kind of like a library book. Popular centralized version control systems include CVS, Perforce (both centralized and distributed), and SVN.


SVN is one example of centralized version control.

Distributed Version Control

Distributed version control systems work quite differently as all users check out a full history of the repository, as opposed to just one piece of it. If something happens to a digital book, no one else's experience is interrupted. That's how distributed version control systems work—every check out is a copy of the full version and history of the repository. Git and Mercurial are popular examples of distributed version control systems.

Each of these models has inherent strengths and weaknesses. Centralized repositories allow for tighter security and line of sight over data. Distributed systems allow for concurrent workflows that can speed up the overall development process.

SVN, Git, and Perforce

In this guide, we'll focus on three of the most popular VCS options: Git, SVN, and Perforce. Each of these options has its own inherent assets and can be best suited for different types of projects.

SVN

SVN is the veteran VCS option, introduced in 2000 by CollabNet. SVN became part of the Apache family in 2009. SVN is a centralized version control system.

SVN is a favorite of industrial companies and older enterprises who have a mix of old and new code. Over the last few months, the team at Assembla has made improvements to SVN to make it more modern and cloud-friendly.

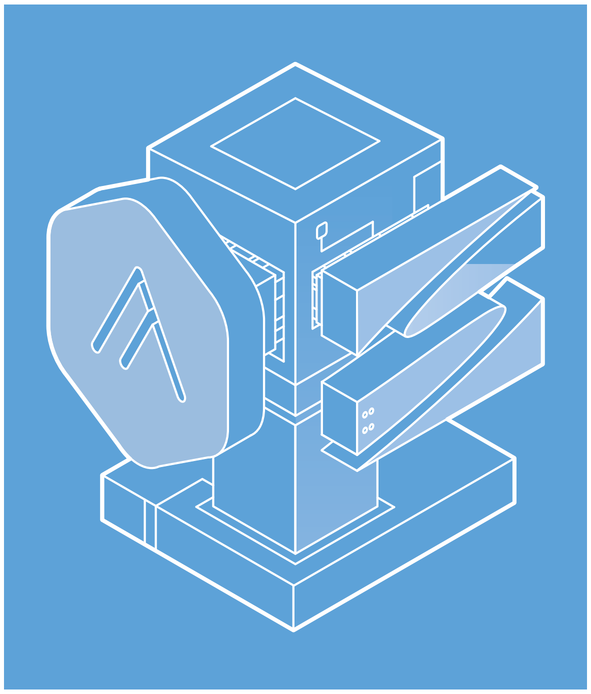
Although SVN's established nature can be its strength, some critique SVN for its older features. Last year, we invested in improvements to SVN. You can read more about the updates we've made here.

Benefits of SVN

Security of a centralized repository

One of the biggest strengths of SVN is that it's a centralized repository. Depending on an organization's workflow, this can be ideal or hold the team back. A centralized repository keeps the history of changes on a central server from which everyone requests the latest version of the work and pushes the latest changes to. This means that everyone sharing the server also shares everyone's work.

This model also allows for more access control because the server is central and admins can restrict users to certain files, something that is virtually impossible in a distributed system. Depending on the goals of your project, this can be a benefit.



Assembla has invested in improvements to SVN so that you can more easily take advantage of its many benefits for your team.

File locking and support for large, binary files

One of SVN's primary advantages is the ability to lock files. A locked file means that users cannot make a commit to the file until it is unlocked. This feature is useful for security and access purposes in an organization where management needs to keep a close eye on what's being committed.

File locking is also very useful when it comes to working with binary files (e.g., word documents and graphics). Binary files cannot be merged in Git. But with file locking in SVN, there's a suitable workaround: lock-modify-unlock. For this reason, SVN is often preferred by designers or companies that rely heavily on graphic assets (gaming, visual effects, etc.)

Permissions and managerial control

SVN supports more granular repository permissions than some other VCS options. It's important to note that there can be variations in functionality depending on your version control provider/host. With SVN, administrators can grant developers access to certain parts of a repository. Combined with file-locking capabilities, administrators can keep a tight rein on access to certain parts of a repo or file

Less to keep track of

Centralized version control solutions require you to effectively keep track of only two data repositories: the SVN server and files in your local working copy. On the other hand with Git, you would have to worry about infinitely more repositories as all users are effectively working on a full copy. This can be problematic in terms of security of those repositories.

Git

Another type of version control is Git. Git has been around since 2005 and has become a very popular solution for developers. Unlike Subversion, Git is a distributed version control system and is free/open source.

When it was originally created, some of the goals for Git included speed, simple design, and a fully distributed model. [Git SCM]

Git's distributed nature makes it a great fit for teams with distributed workforces.

Benefits of Git

Distributed version control

Just as many of the benefits of SVN stem from its centralized nature, the same is true for Git and its features that are related to its distributed nature. Git is ideal for agile teams with a distributed workforce. Because developers get a full history of their local repository, the workflow is ultimately faster.

Branching

One of the biggest advantages of Git is the ability to branch. This is part of the benefit of being distributed. A branch allows users to work on some lines of code without interfering with the production code.

Speed

Depending on the makeup and priorities of your team, having a distributed version control system can increase speed and efficiency because every developer has access to a local repository. Granted, this must be balanced out by your security and access needs.

Perforce

A favorite of gaming and VR/AR studios, Perforce is often considered industry standard. Perforce has many strengths as a version control system.

Perforce is the industry standard version control tool for gaming studios.

Benefits of Perforce

Supports both centralized and distributed workflows

Because both systems have inherent strengths, this level of flexibility is something other version control systems simply don’t offer.

Performance and scalability

Speed is a major strength of Perforce. Perforce repositories hold millions of files and many terabytes of data. Their case studies indicate that “Perforce is 5 to 10 times faster than SVN when syncing large numbers of files.” It’s no question that Perforce is very powerful, especially for industries that depend on large files.

Industry standard for gaming studios

Like SVN, Perforce handles binary files well. This is part of the reason it is so popular with gaming studios. Additionally, Perforce integrates directly with the two premier gaming engines: Unity and Unreal Engine.

Need to talk through your options?

Reach out to Assembla at sales@assembla.com or by phone at (800)405-4408.

TRY IT FREE