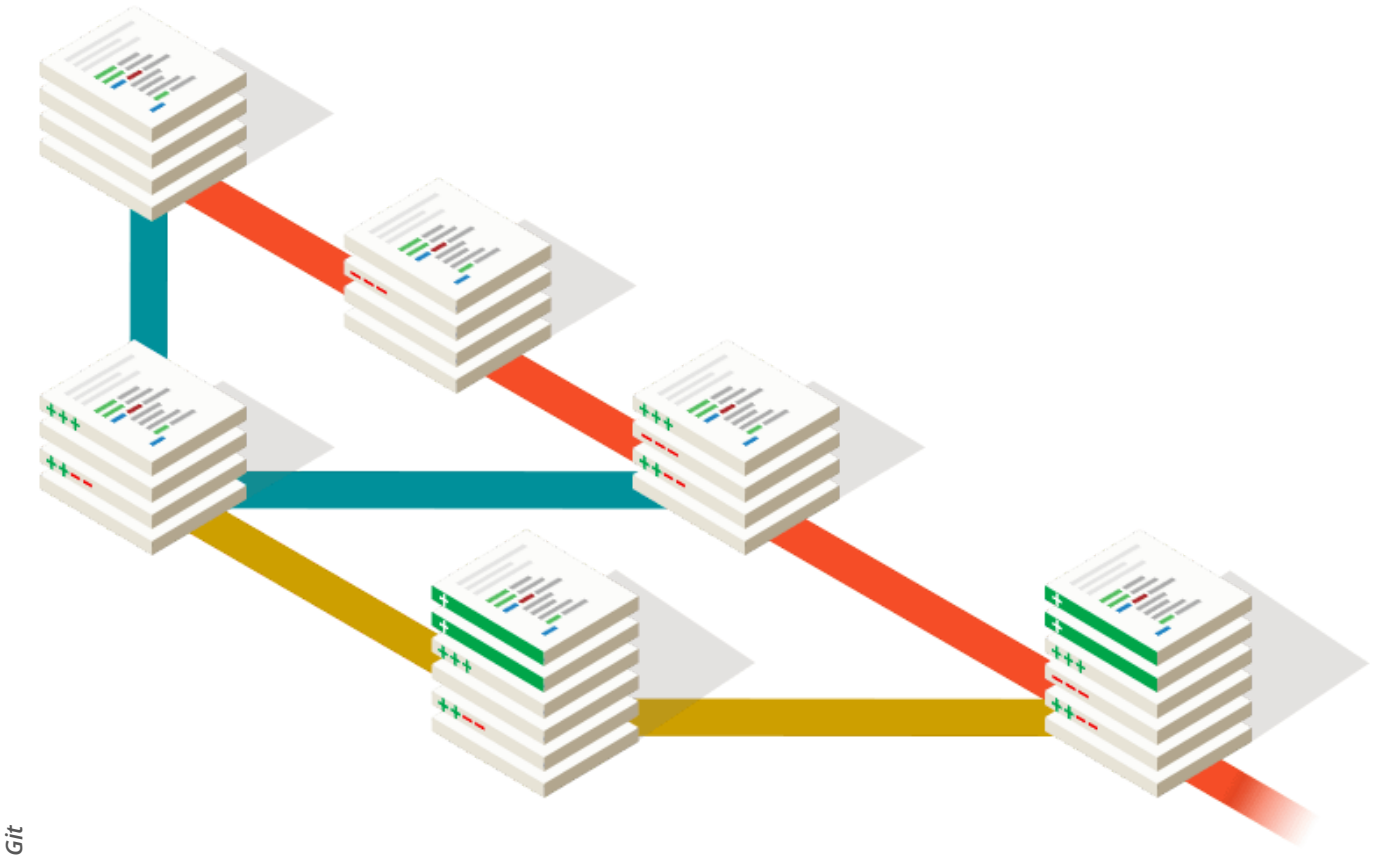
BIZ & IT —

Microsoft hosts the Windows source in a monstrous 300GB Git repository

Virtualized file system approach makes Git work better for huge repositories.

PETER BRIGHT - 2/7/2017, 12:20 AM



Git

Enlarge

Git, the open source distributed version control system created by Linus Torvalds to handle Linux's decentralized development model, is being used for a rather surprising project: Windows.

Traditionally, Microsoft's software has used a version control system called Source Depot. This is proprietary and internal to Microsoft; it's believed to be a customized version of the commercial Perforce version control system, tailored for Microsoft's larger-than-average size. Over the years, Redmond has also developed its own version control products. Long ago, the company had a thing called SourceSafe, which was reputationally the moral equivalent to tossing all your precious source code in a trash can and then setting it on fire thanks to the system's propensity to corrupt its database. In the modern era, the Team Foundation Server (TFS) application lifecycle management (ALM) system offered Team Foundation Version Control (TFVC), a much more robust, scalable version control system built around a centralized model.

Much of the company uses TFS not just for version control but also for bug tracking, testing, automated building, and project management. But large legacy products, in particular Windows and Office, stuck with Source Depot rather than adopting TFVC. The basic usage model and theory of operation between Source Depot and TFVC are pretty similar, as both use a centralized client-server model.

Since 2013, Microsoft has been integrating Git into TFS, and today TFS and Visual Studio offer full support for centralized version control using TFVC and distributed version control using Git. With this

first-party support for the system, Git adoption has spread within the company, most visibly in open source projects such as ChakraCore, the JavaScript engine used in the Edge browser, but also in closed source products—including, as it turns out, Windows itself.

We've written about OneCore, Microsoft's restructuring of Windows and unification of the operating system across phones, tablets, Xbox, PCs, servers, HoloLens, and beyond. Before OneCore, Microsoft had multiple incompatible forks of Windows, each with their own development streams, causing substantial duplication of effort. With OneCore, the common parts were brought together, and the unique customizations—things like Xbox's dashboard, HoloLens's 3D interface—cleanly isolated and layered on top.

Just as Windows' development had become complex and fragmented, so too did the company's internal systems for things like source control, issue tracking, testing, building, code analysis, and all the other tasks that fall under the application lifecycle management umbrella. And just as Windows' development was unified as OneCore, the company has embarked on an effort to unify its ALM and develop what it calls One Engineering System (1ES).

The cornerstone of 1ES is TFS, but for 1ES, the company wanted to do more than just standardize on TFS; it wanted to switch to a single version control system. TFVC, Source Depot, and Git were the obvious contenders, though other options such as Mercurial were also considered. In the end, the company standardized on Git.

However, this decision came with some complexity. The Windows codebase, for example, is large, with decades of history. It has millions of files, taking hundreds of gigabytes of storage. In a centralized version control system, this isn't too big an issue; only the central server needs to store all of this data, with each developer only needing to store the latest source code on their local systems. But decentralized systems don't work this way; by default, making a local working copy of a remote repository in Git requires replicating *everything*, including the decades of history. This is key to its decentralized nature—every repository contains all the history of all the files, making them all equal peers. For Windows, this meant that every developer would need to fetch millions of files and hundreds of gigabytes. The initial clone of the repository took hours, and even simple tasks such as checking to see if all files are up to date took many minutes.

Accordingly, Microsoft has been working to enhance Git to improve the way it handles vast repositories. Central to this effort is a new project released (in part) as open source Git Virtual File system (GVFS). The premise of GVFS is straightforward enough: rather than fetching all the data at once, only a bare skeleton of the repository needs to be populated up front. The virtualized file system subsequently retrieves additional data on a demand-driven, as-needed basis. Building one particular Windows component, for example, will cause GVFS to fetch the files that make up that component, along with anything that the component depends on, but it will stop short of fetching all the many hundreds of gigabytes the repository contains.

This work requires changes to Git itself, which Microsoft is working to contribute back to the Git project. This work is naturally open source. So too is a large portion of GVFS itself. But a key portion is not; while the code for fetching files and interacting with a remote Git repository is all open, the actual *file system* bit that runs in kernel mode is not.

FUSE for Windows on the horizon?

Currently, that file system driver is available as a preview with a restrictive license. Microsoft says that the driver isn't yet ready for prime time—you should only test GVFS in a virtual machine or similar discardable environment. The version available now is just a preview. But the driver itself may turn out to be useful for more than GVFS, and in so doing could fill a longstanding gap in Windows' functionality.

Developing file system drivers is pretty complex on any platform—if a file system driver crashes, you have the double inconvenience of crashing the machine with a blue screen or kernel panic *and* the specter of data loss due to screwing something up with how data is read from or written to the disk—but Windows makes it particularly awkward. That's because Windows has no first party, supported equivalent to FUSE ("file system in userspace"), a framework for developing file systems *without* having to write kernel code.

FUSE is available on macOS, Linux, FreeBSD, Android, and more. It can be used to develop full file systems that store data on disks, but just as often, it's used for "virtual file systems" of the very kind that Microsoft has created with GVFS. With GVFS, files are stored locally on a regular NTFS disk or remotely on a Git server. GVFS doesn't manage the actual on-disk layout of how that data is stored; it just provides a sort of intercept layer. If a program tries to open a file that hasn't yet been cached locally, GVFS will fetch it from the remote Git repository and store it locally on NTFS before allowing the open operation to proceed.

There are many FUSE file systems that work the same kind of way, transparently fetching files from, for example, cloud storage or remote systems connected by ssh, as well as copying them back to the remote system whenever the local file is modified.

Lacking FUSE, Windows has no good way of developing this same kind of virtual file system. This is unfortunate. Windows 8.1 included a neat way of using OneDrive: all your cloud files "appeared" local, but the data would only actually be fetched when you attempted to open the file. However, Windows 8.1 didn't use a file system driver for this integration with OneDrive. While attempts to open cloud files from within Explorer (and within certain applications) were properly intercepted, causing only a slight delay while the file was downloaded before it could be opened, Windows 8.1 *didn't* intercept attempts to open files made from the command-line or through low-level Win32 APIs. This made the OneDrive integration rather uneven: in some places, it worked as it should, transparently fetching and saving files as you worked with them, but in others it just produced error messages. As a result, Microsoft removed the feature in Windows 10.

(In contrast, Dropbox's new Project Infinite capability, which has recently become available to business users, *does* use a file system driver and so should offer much greater compatibility.)

Microsoft describes the GVFS driver as the "moral equivalent of the FUSE driver in Linux." If it truly is the moral equivalent of FUSE, it suggests that Windows will at last get the same kind of extensibility and scope for user mode file systems that Unix users have enjoyed for many years. It might even provide the basis for a better reimplementation of the OneDrive cloud storage feature that was taken away.

Microsoft isn't alone in facing scaling limits from existing version control systems; a few years ago, Facebook switched from a combination of Git and Subversion to Mercurial. Facebook felt that neither Git nor Subversion offered the scalability that it needed; it considered modifying Git, but that it would be easier to extend and improve Mercurial instead.

READER COMMENTS 186

SHARE THIS STORY

← PREVIOUS STORY NEXT STORY →

Related Stories

Microsoft and GitHub team up to take Git virtual file system to macOS, Linux

Windows switch to Git almost complete: 8,500 commits and 1,760 builds each day

Microsoft embraces git with new TFS support, Visual Studio integration

Microsoft snaps up GitHub for $7.5 billion

Today on Ars

Apple threatens to boot Epic—including Unreal Engine—off Mac and iOS

Cities sue Netflix, Hulu, Disney+, claim they owe cable “franchise fees”

Secret Service buys location data that would otherwise need a warrant

You can now play an ultra-rare *Quake* arcade cabinet at home

Third-party Mac repair shops will gain access to Apple tools, parts

The first road tests of the Volkswagen ID.3 electric car are showing up

This 9,000-year-old skeleton is the oldest cremation in the Near East

Review: Smartly satirical *Teenage Bounty Hunters* is a perfect weekend binge