Git at Scale

02/20/2018 • 4 minutes to read

In this article

By: Saeed Noursalehi

We recently announced that the entire Windows codebase has moved to a single Git repo that is hosted on Azure DevOps. This migration presented us with some very interesting scale challenges to solve, ranging from issues inherent in the Git protocol and object storage, to the performance of the git command line, to workflow challenges for a team of that size working in one repo. In this series of articles, we'll discuss each of these scale challenges in depth, and how we have solved them to enable a team of this size to work successfully in Git.

The scale journey has been a long one, and these articles will take you through that journey. When talking about "Git at scale", it's important to remember that there are a number of independent factors that affect Git scale (things like number of people, number of files, etc), and that there is a wide spectrum of repo and team sizes. To help us focus on some specific examples along that scale spectrum, we'll refer to the following classes of repos throughout these articles.

- Small repos, e.g. the GVFS repo
- Medium repos, e.g. the Azure DevOps repo
- Extra large repos, e.g. the Windows repo

These are all real repos that we use inside Microsoft, and they range from tiny to the biggest repo we have. As we go through various scale challenges and solutions, we'll refer back to these repos to make the problems more concrete. At the end of the series, we will come back to these repos and discuss how each of these teams have composed the various scale solutions to create the best possible experience for their teams.

These articles are intended for people who have a good understanding of Git, and are thinking about how to make Git scale to larger teams.We will not cover the basics of how Git works, but where necessary, we will cover the inner workings of Git that are relevant to the scale problem at hand.

Small repos

Even though GVFS is a product meant for the world's largest repos, GVFS itself has a rather ordinary codebase when it comes to its scale needs. GVFS stats:

- Repo size: 10MB packfile, 2MB working directory, 400 files
- Team: 10 people
- Branches: 300
- Build/Test: 30s build, 10s unit tests, 30m functional tests

This is a pretty typical repo when compared the to the thousands of repos within Microsoft and elsewhere, and this is pretty much the size of repo that Git excels at. There are no scale challenges to speak of here. Most of the challenges fall into the bucket of making sure code quality is kept at a high level, that there is a clear workflow around pull requests and release process, that sort of thing.

Medium repos

Azure DevOps and Team Foundation Server are both built out of one codebase. At Microsoft, we consider this a medium-sized repo, though compared to most other repos in the world, it's definitely on the larger end of the spectrum. We currently have on the order of 10 or so repos in Microsoft that are around this size. Azure DevOps stats:

- Repo size: 10GB packfile, 3GB working directory, 100K files
- Team: 400 people
- Branches: 20K
- Build/Test: 10m build, 5m unit tests, 5m basic functional tests, 5h full functional tests

For a repo of this size, we're now in a pretty good place, but we've had to solve many scale issues to get here. The most painful part of our path to get here was the rate at which 400+ people can push changes to master, making it difficult to ever win the race to push.

Extra large repos

The Windows codebase is in a class of its own. This code includes all of OneCore, Desktop, Server, Xbox, IOT, Mobile, and HoloLens. Along with the fact that this codebase has been around for decades and still supports legacy features all the way back to the beginning of Windows, you can imagine that the codebase is quite large. Windows stats:

- Repo size: 100GB packfile, 300GB working directory, 3.5M files
- Team: 4000 people
- Branches: Estimated to reach between 150K to 250K
- Build/Test: A few seconds to few minutes incremental build, 12h full build, a few minutes incremental tests, several hours for full validation

With a repo of this size, we've had to use every trick we can think of to make Git scale and to ensure that developers can be fully productive. We recently announced the Git Virtual File System as one of the pieces in that puzzle. In this series of articles, we'll examine that entire puzzle, and how all the pieces fit together.

In the next article in the series, we'll discuss the specific issues that Git encounters as a repo grows in size.

Git at Scale articles

- Technical Scale Challenges
- Limited Refs
- The Race to Push
- GVFS Design History
- GVFS Architecture

Other articles coming soon

- Windows repo case study

Saeed Noursalehi is a Principal Program Manager on the Azure DevOps team at Microsoft, and works on making Git scale for the largest teams in Microsoft