Migrate from TFVC to Git

04/03/2017 • 5 minutes to read

By: Edward Thomson

Azure DevOps provides a simple migration tool to migrate from Team Foundation Version Control to Git. If you're not using TFVC, you can migrate from another system manually.

Before you try to migrate your source code from a centralized version control system to Git, be sure that you familiarize yourself with the differences between centralized and distributed version control systems, and plan your team's migration. After you've prepared, you can begin the migration.

Requirements

In order to make migrations simple, there are a number of requirements on the TFVC Import tool:

1. Only a single branch is migrated. When planning your migration you should choose a new branching strategy for Git; migrating only the main branch supports a topic-branch based workflow like GitFlow or GitHub Flow.
2. A "tip migration", importing only the latest version of the source code, is suggested. You can opt to migrate some history, up to 180 days, so that your team doesn't need to refer back to TFVC as often, but this is discouraged unless your history is very simple.
3. You do not have binary assets like images, scientific data sets, or game models in your repository. These assets should use the Git LFS (Large File Support) extension, which the import tool does not configure.
4. The imported repository cannot exceed 1GB in size.

If you do not meet these requirements, you can use the Git-TFS tool to perform the migration, or perform a manual migration.

In general, the process to migrate from TFVC is very easy:

Steps to migrate

1. Check out the latest version of your branch from TFVC onto your local disk.
2. Remove binaries and build tools from your repository and set up a package management system like NuGet.
3. Convert version control system-specific directives that you need to retain in Git. For example, convert `.tfignore` files to `.gitignore`, and convert `.tpattributes` files to `.gitattributes`.
4. Check in the final changes to Team Foundation Version Control and perform the migration to Git.

Steps 1-3 are optional; if you do not have any binaries in your repository and you do not need to set up a `.gitignore` or a `.gitattributes`, then you can skip straight to performing the migration.

Check out the latest version

Create a new TFS workspace, and map a working folder for the server directory that you're going to migrate to Git. You do not need to do a full working folder mapping; you only need to map folders that contain binaries that you're going to remove from the repository and folders that contain version control system-specific configuration files like `.tfignore`.

Once you have set up your mappings, get the folder locally:

prettyprint

```
tf get /version:T /recursive
```

Remove binaries and build tools

Due to the way Git stores the history of changed files, providing a copy of every file in history to every developer, checking in binary files directly to the repository will cause it to grow quickly and cause performance issues.

For build tools and dependencies like libraries, adopt a packaging solution with versioning support, such as NuGet. Many open source tools and libraries will already be available on the NuGet Gallery, but for proprietary dependencies, you will need to create your own NuGet packages.

Once you have moved your dependencies into NuGet, make sure that they will not be included in your Git repository by adding them to your `.gitignore` file.

Convert version control-specific configuration

Team Foundation Version Control provides a `.tfignore` file that will ensure that certain files are not added to your TFVC repository. This can be used for automatically generated files like build output, so that it is not accidentally checked in.

If you rely on this behavior, convert your `.tfignore` file to a `.gitignore` file.

Cross-platform TFVC clients also provide support for a `.tpattributes` file that controls how files are placed on the local disk or checked in to the repository. If you use a `.tpattributes` file, convert it to a `.gitattributes` file.

Check in to TFVC and perform the migration

Check in any changes you made that remove binaries, migrate to package management, or convert version control-specific configuration. Once this final change is made in TFVC, you can perform the import.

Follow the Import repositories documentation to actually perform the input.

Advanced migrations

The Git-TFS tool is a two-way bridge between Team Foundation Version Control and Git, and can be used to perform a migration. Git-TFS is appropriate if you want to attempt a migration with full history, more than the 180 days that the Import tool supports, or if you want to attempt a migration that includes multiple branches and merge relationships.

Before you attempt a migration with Git-TFS, you should be aware that there are fundamental differences between the way TFVC and Git store history:

- Git stores history as a snapshot of the repository in time, while TFVC records the discrete operations that occurred on a file. Change types in TFVC like rename, undelete and rollback cannot be expressed in Git; instead of seeing that file `A` was renamed to file `B`, you may only see that file `A` was deleted and file

`B` was added in the same commit.

- Git does not have a direct analog of a TFVC label: labels can contain any number of files at any specific version and can reflect files at different versions. Although conceptually similar, Git tags point to a snapshot of the whole repository at a point in time. If you rely on TFVC labels to know what was delivered, Git tags may not be provide this information.
- Merges in TFVC occur at the file level, not at the entire repository. You can merge only a subset of changed files from one branch to another, then merge the remaining changed files in a subsequent changeset. In Git, a merge affects the entire repository, and you cannot see both sets of individual changes as a merge.

Because of these differences, we generally advise users to do a tip migration and keep their TFVC repository online but read-only to view history.

If you want to attempt an advanced migration with Git-TFS, that project provides documentation on how to do a migration from TFVC to Git, cloning a single branch with history or cloning all branches with merge history.

Update your workflow

Moving from a centralized version control system to Git is more than just migrating code. Your team needs training to understand how Git is different from your existing version control system and how these differences affect day-to-day work. Learn more.

Get started with unlimited free private Git repos in Azure Repos.

Edward is a Program Manager for Azure DevOps, focusing on Git and Version Control. He is the author of “Git for Visual Studio” training from O’Reilly Media and a contributor to “Professional Team Foundation Server 2013”.