Introduction

Completed

- 1 minute

This module introduces Azure Repos and GitHub and explores ways to migrate from TFVC to Git and work with GitHub Codespaces for development.

Learning objectives

After completing this module, students and professionals can:

- Describe Azure Repos and GitHub.
- Migrate from TFVC to Git.
- Work with GitHub Codespaces.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with Git and version control principles is helpful but is not necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Introduction to Azure Repos

Completed

- 2 minutes

Azure Repos is a set of version control tools that you can use to manage your code.

Using version control is a good idea whether your software project is large or small.

Azure Repos provides two types of version control:

- Git: distributed version control
- Team Foundation Version Control (TFVC): centralized version control

What do I get with Azure Repos?

- Use free private Git repositories, pull requests, and code search: Get unlimited private Git repository hosting and support for TFVC that scales from a hobby project to the world's largest repository.
- Support for any Git client: Securely connect with and push code into your Git repository from any IDE, editor, or Git client.
- Web hooks and API integration: Add validations and extensions from the marketplace or build your own-using web hooks and REST APIs.
- Semantic code search: Quickly find what you are looking for with a code-aware search that understands classes and variables.
- Collab to build better code: Do more effective Git code reviews with threaded discussion and continuous integration for each change. Use forks to promote collaboration with inner source workflows.
- Automation with built-in CI/CD: Set up continuous integration/continuous delivery (CI/CD) to automatically trigger builds, tests, and deployments. Including every completed pull request using Azure Pipelines or your tools.
- Protection of your code quality with branch policies: Keep code quality high by requiring code reviewer sign-out, successful builds, and passing tests before merging pull requests. Customize your branch policies to maintain your team's high standards.

- Usage of your favorite tools: Use Git and TFVC repositories on Azure Repos with your favorite editor and IDE.

For further reference on using git in Azure Repos, refer to Microsoft Docs.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Introduction to GitHub

Completed

- 3 minutes

GitHub is the largest open-source community in the world. Microsoft owns GitHub. GitHub is a development platform inspired by the way you work.

You can host and review code, manage projects, and build software alongside 40 million developers from open source to business.

GitHub is a Git repository hosting service that adds many of its features.

While Git is a command-line tool, GitHub provides a Web-based graphical interface.

It also provides access control and several collaboration features, such as wikis and essential task management tools for every project.

So what are the main benefits of using GitHub? Nearly every open-source project uses GitHub to manage its project.

Using GitHub is free if your project is open source and includes a wiki and issue tracker, making it easy to have more in-depth documentation and get feedback about your project.

What are some of the features offered by GitHub?

- Automate from code to cloud: Cycle your production code faster and simplify your workflow with GitHub Packages and built-in CI/CD using GitHub Actions.
    - Automate your workflows: Build, test, deploy, and run CI/CD how you want in the same place you manage code. Trigger Actions from any GitHub event to any available API. Build your Actions in the language of your choice, or choose from thousands of workflows and Actions created by the community.
    - Packages at home with their code: Use Actions to automatically publish new package versions to GitHub Packages. Install packages and images hosted on GitHub Packages or your preferred packages registry in your CI/CD workflows. It is always free for open source, and data transfer within Actions is unlimited for everyone.
- Securing software together: GitHub plays a role in securing the world's code—developers, maintainers, researchers, and security teams. On GitHub, development teams everywhere can work together to secure the world's software supply chain, from fork to finish.
    - Get alerts about vulnerabilities in your code: GitHub continuously scans security advisories for popular languages. Also, it sends security alerts to maintainers of affected repositories with details so they can remediate risks.
    - Automatically update vulnerabilities: GitHub monitors your project dependencies and automatically opens pull requests to update dependencies to the minimum version that resolves known vulnerabilities.
    - Stay on top of CVEs: Stay updated with the latest Common Vulnerabilities and Exposures (CVEs), and learn how they affect you with the GitHub Advisory Database.
    - Find vulnerabilities that other tools miss: CodeQL is the industry's leading semantic code analysis engine. GitHub's revolutionary

approach treats code as data to identify security vulnerabilities faster.
    - Eliminate variants: Never make the same mistake twice. Proactive vulnerability scanning prevents vulnerabilities from ever reaching production.
    - Keep your tokens safe: Accidentally commit a token to a public repository? GitHub got you. With support from 20 service providers, GitHub takes steps to keep you safe.

- Seamless code review: Code review is the surest path to better code and is fundamental to how GitHub works. Built-in review tools make code review an essential part of your team's process.

    - Propose changes: Better code starts with a Pull Request, a living conversation about changes where you can talk through ideas, assign tasks, discuss details, and conduct reviews.
    - Request reviews: If you are on the other side of a review, you can request reviews from your peers to get the detailed feedback you need.
    - See the difference: Reviews happen faster when you know exactly what changes. Diffs compare versions of your source code, highlighting the new, edited, or deleted parts.
    - Comment in context: Discussions happen in comment threads within your code—bundle comments into one review or reply to someone else who is in line to start a conversation.
    - Give clear feedback: Your teammates should not have to think too hard about what a thumbs-up emoji means. Specify whether your comments are required changes or just a few suggestions.
    - Protect branches: Only merge the highest-quality code. You can configure repositories to require status checks, reducing human error and administrative overhead.

- All your code and documentation in one place: Hundreds of millions of private, public, and open-source repositories are hosted on GitHub. Every repository has tools to help you host, version, and release code and documentation.

    - Code where you collaborate: Repositories keep code in one place and help your teams collaborate with the tools they love, even if you work with large files using Git LFS. You can create or import as many projects as possible with unlimited private repositories for individuals and groups.
    - Documentation alongside your code: Host your documentation directly from your repositories with GitHub Pages. Use Jekyll as a static site generator and publish your Pages from the /docs folder on your main branch.

- Manage your ideas: Coordinate early, stay aligned, and get more done with GitHub's project management tools.

    - See your project's large picture: See everything happening in your project and choose where to focus your team's efforts with Projects and task boards that live right where they belong: close to your code.
    - Track and assign tasks: Issues help you identify, assign, and keep track of tasks within your team. You can open an Issue to track a bug, discuss an idea with an @mention , or start distributing work.

- The human side of software: Building software is more about managing teams and communities than coding. Whether on a group of two or 2000, GitHub has the support your people need.

    - Manage and grow teams: Help people organize with GitHub teams, level up to access administrative roles, and fine-tune your permissions with nested teams.
    - Keep conversations: Moderation tools, like issue and pull request locking, help your team stay focused on code. And if you maintain an open-source project, user blocking reduces noise and ensures productive conversations.
    - Set community guidelines: Set roles and expectations without starting from scratch. Customize standard codes of conduct to create the perfect one for your project. Then choose a pre-written license right from your repository.

GitHub offers excellent learning resources for its platform. You can find everything from git introduction training to deep dive on publishing static pages to GitHub and how to do DevOps on GitHub right here .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Migrate from TFVC to Git

Completed

- 2 minutes

Migrating the tip

Most teams wish they could reorganize their source control structure.

Typically, the structure the team is using today was set up by a well-meaning developer a decade ago, but it is not optimal.

Migrating to Git could be an excellent opportunity to restructure your repo.

In this case, it probably does not make sense to migrate history anyway since you are going to restructure the code (or break the code into multiple repos).

The process is simple:

- Create an empty Git repo (or multiple empty repos).
- Get-latest from TFS.
- Copy/reorganize the code into the empty Git repos.
- Commit and push, and you are there!

If you have shared code, you need to create builds of the shared code to publish to a package feed. And then consume those packages in downstream

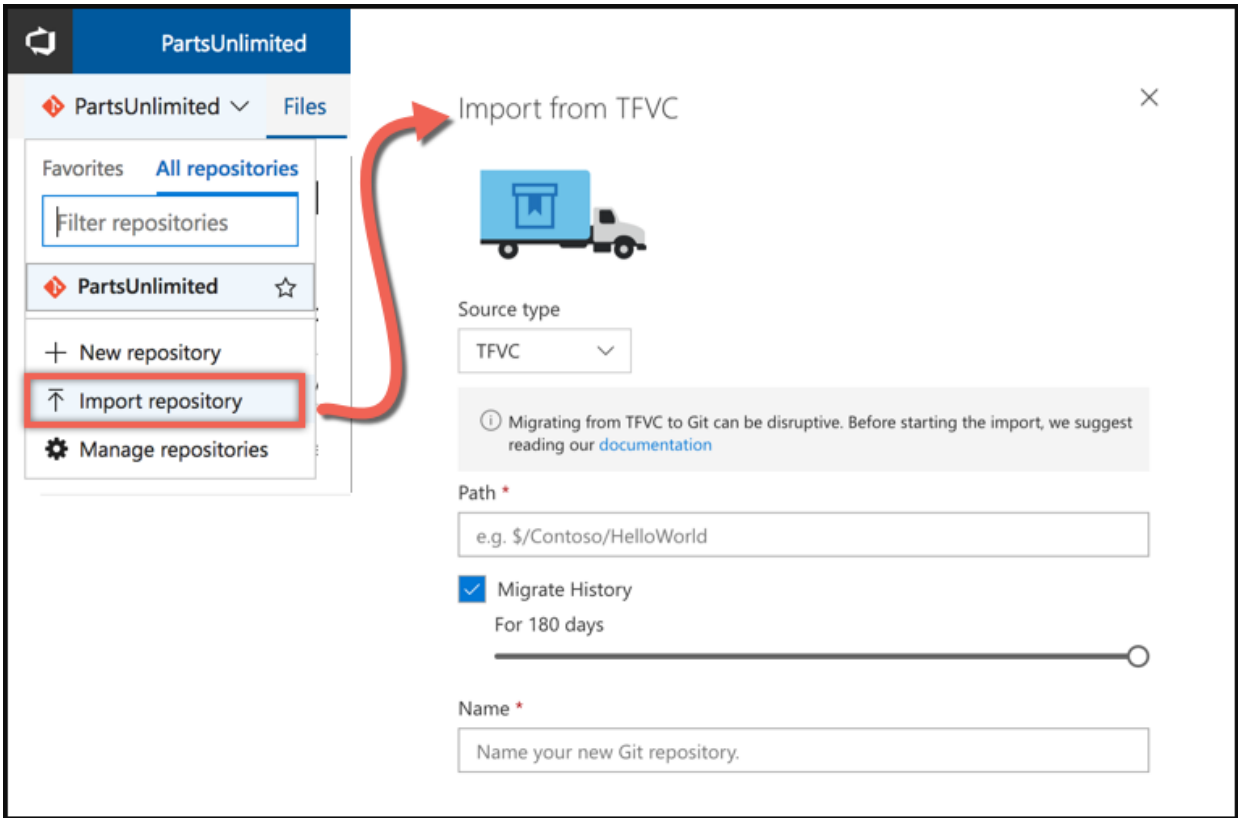
applications, but the Git part is straightforward.

Single branch import

If you are on TFVC and in Azure DevOps, you have the option of a simple single-branch import. Click on the Import repository from the Azure Repos top-level drop-down menu to open the dialog. Then enter the path to the branch you are migrating to (yes, you can only choose one branch). Select if you want history or not (up to 180 days). Add in a name for the repo, and the import will be triggered.

Import repository

Import repository also allows you to import a git repository. It is beneficial to move your git repositories from GitHub or any other public or private hosting spaces into Azure Repos.



There are some limitations here (that apply only when migrating source type TFVC): a single branch and only 180 days of history.

However, if you only care about one branch and are already in Azure DevOps, it is an effortless but effective way to migrate.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Use GIT-TFS

Completed

- 3 minutes

What if you need to migrate more than a single branch and keep branch relationships? Or are you going to drag all the history with you?

In that case, you're going to have to use **GIT-TFS** . It's an open-source project built to synchronize Git and TFVC repositories.

But you can do a once-off migration using the Git TFS clone.

GIT-TFS has the advantage that it can migrate multiple branches and preserve the relationships to merge branches in Git after you migrate.

Be warned that doing this conversion can take a while - especially for large or long-history repositories.

You can quickly dry-run the migration locally, iron out any issues, and then do it for real. There are lots of flexibilities with this tool.

If you are on Subversion, you can use **GIT-SVN** to import your Subversion repo similarly to **GIT-TFS** .

Migrating from TFVC to Git using GIT-TFS

If Chocolatey is already installed on your computer, run `choco install gittfs`

Add the GIT-TFS folder path to your PATH. You could also set it temporary (the time of your current terminal session) using: `set PATH=%PATH%;%cd%\GitTfs\bin\Debug`

You need .NET 4.5.2 and maybe the 2012 or 2013 version of Team Explorer (or Visual Studio). It depends on the version of Azure DevOps you want to target.

Clone the whole repository (wait for a while.) :

```
git tfs clone http://tfs:8080/tfs/DefaultCollection $/some_project
```

Advanced use cases of cloning the TFVC repository into Git are documented here .

```
cd some_project
git log
```

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Develop online with GitHub Codespaces

Completed

- 2 minutes

GitHub Codespaces addresses several issues from which developers regularly suffer.

First, many developers are working with old hardware and software systems, which are not refreshed.

Second, developers are often tied to individual development systems. Moving from location to location or system to system is inconvenient or slow to configure.

A problem for the developers' organizations is the proliferation of intellectual property across all these machines.

What is GitHub Codespaces?

Codespaces is a cloud-based development environment that GitHub hosts. It is essentially an online implementation of Visual Studio Code.

Codespaces allows developers to work entirely in the cloud.

Codespaces even will enable developers to contribute from tablets and Chromebooks.

Because it is based on Visual Studio Code, the development environment is still rich with:

- Syntax highlighting.
- Autocomplete.
- Integrated debugging.

- Direct Git integration.

Developers can create a codespace (or multiple codespaces) for a repository. Each codespace is associated with a specific branch of a repository.

Using GitHub Codespaces

You can do all your work in codespaces within a browser.

For an even more responsive experience, you can connect to a codespace from a local copy of Visual Studio Code.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 4 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices isn't a supported Azure Repos version control system?

○

Git.

○

Team Foundation Version Control (TFVC).

○

Source Safe.

2.

Which of the following choices is the exact number of days of history you can import Team Foundation Version Control (TFVC) to Git using the "Import repository" feature in Azure DevOps?

○

90 days.

○

180 days.

○

365 days.

3.

Which of the following choices describe GitHub Codespaces?

○

It's a platform for hosting and managing packages, including containers and other dependencies.

○

It's an AI pair programmer that helps you write code faster and with less work.

It's an online implementation of Visual Studio Code.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module introduced Azure Repos and GitHub and explored ways to migrate from TFVC to Git and work with GitHub Codespaces for development.

You learned how to describe the benefits and usage of:

- Describe Azure Repos and GitHub.
- Migrate from TFVC to Git.
- Work with GitHub Codespaces.

Learn more

- Integration of Azure Repos and Git Repositories .
- Integration of Azure Boards and GitHub .
- Import repositories from TFVC to Git - Azure Repos | Microsoft Docs .
- GitHub Codespaces .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .