

TFS – Team Foundation Server

Contents

[1. TFS (Team Foundation Server) 3](#_Toc138083857)

[ What is TFS? 3](#_Toc138083858)

[ There are 4 features of TFS: 3](#_Toc138083859)

[ TFS Physical Architecture: 4](#_Toc138083860)

[ TFS Technical Architecture 5](#_Toc138083861)

[ Why Version Control? 6](#_Toc138083862)

[ TFVC (Team Foundation Version Control) 6](#_Toc138083863)

[ How to configure TFVC visual Studio 6](#_Toc138083864)

[ TFVC Code Review in Visual Studio 7](#_Toc138083865)

[ Use Local or Server Workspace 7](#_Toc138083866)

[o Use Local Workspace 7](#_Toc138083867)

[o Server Workspace 8](#_Toc138083868)

[ Check-in Check-out 8](#_Toc138083869)

[o Check-out 8](#_Toc138083870)

[o Make Modification 9](#_Toc138083871)

[o Check-in 9](#_Toc138083872)

[ Branching 9](#_Toc138083873)

[1. Main Only 10](#_Toc138083874)

[2. Development Isolation 10](#_Toc138083875)

[3. Feature Isolation 11](#_Toc138083876)

[ Changeset Tab 11](#_Toc138083877)

[ Multiple Checkout 11](#_Toc138083878)

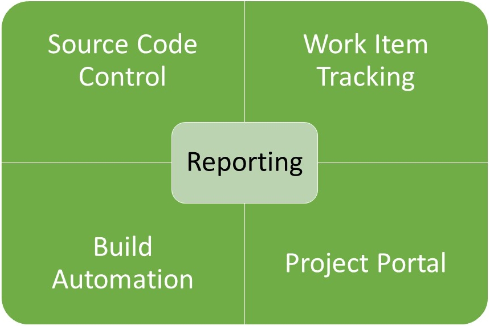
[ Reference 13](#_Toc138083879)

# TFS (Team Foundation Server)

## What is TFS?

* + Team Foundation Server is a product from Microsoft. It is an application lifecycle management platform for .NET Developers.
  + It is a combination of reporting tool and version control tool.

## There are 4 features of TFS:



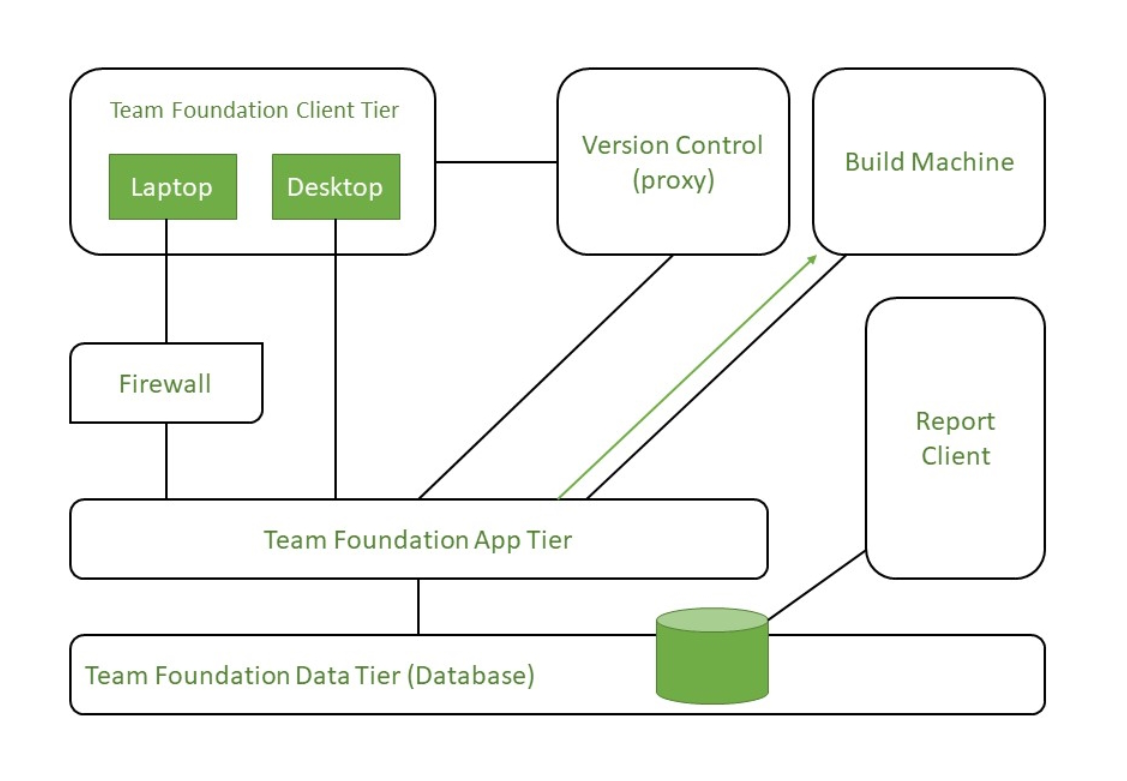
1. **Source Code Control**: TFS is version control system. It lets the team manage all source files or artifacts for a project and lets the developer store data in a centralized repository. These files can be checked-out for working and can be stored as a local copy. Files can be checked in and checked out on a team project too. While one checks out the one file then that file is blocked for every other user.
2. **Work Item Tracking:** A work item is a fundamental piece of work on a team project. For example: Testing working items, Bug working items. This helps to track the status of the assigned work. For example: Who is assigned with the work? What is the work status? etc... This helps to track the status of the assigned work. For example: Who is assigned with the work? What is the work status? Etc.
3. **Build Automation**: TFS enables the team to create and manage product builds regularly. For example, a team may perform daily builds and upload them to a shared server. The build output can be a .exe file.
4. **Project Portal:** Per project there is one share-point where the developer stores documents related to the project and finds reports on it.

## TFS Physical Architecture:



* So, there are three parts of the Architecture Team Foundation Server, Team Foundation Build Server, Team Foundation Server Proxy.
  + **Team Foundation Server**: It can be called the heart of the deployment. It exposes web services that are consumed by different types of clients. It stores entire information in database.
  + **Team Foundation Server Build**: It is useful for the building of projects on TFS. It must be installed separately and configured with TFS.
  + **Team Foundation Server Proxy**: This is an optional component. It is used in a distributed team (onsite developers). For better performance of the remote developers, this team foundation server proxy must be installed near them.

## TFS Technical Architecture



* **Team Foundation Data Tier**: It can be called the backbone of TFS where all the source control-related items and reports are stored.
* **Team Foundation App Tier:** The data tier is abstracted by the App Tier called the TFS. Requests to this tier come via the Team Foundation Client Tier. The requests can come from Team Explorer, Excel, or a project. The Client connects to TFS over HTTP or HTTPS. The app tier gets the data from the database. The app tier and the data tier can be on different machines.
* **Build Machine:** It performs build services to generate the executable file or the deployable unit.
* **Version Control Proxy:** It is required for remote developers. The check-out is done through this proxy. It caches the checked-out item and is used only for check-out. Check-in happens only on the TFS.
* **Report Client:** It uses the SSRS of the database to generate reports.
* **Agile Model**: TFS supports Agile methodology such as Scrum and Kanban. It provides features like backlog management, sprint planning, task boards, and burndown charts to help teams manage their projects effectively.
* TFS has **report and analysis** capability to give insight into the project, process, team performance and quality metrics. It offers predefined reports and allows you to create custom reports.
* As we discussed using TFS we must set up TFS Server or we can use it online version which is called AzureDevOps Server which is cloud based. To use their services, we need to install Visual studio or Azure DevOps Server. After, that we can connect from this tool and use TFS server.
* Once, we connected we can create project, manage source code, and track work item, setup build and release pipeline
* Microsoft rebranded the Team foundation server as Azure DevOps Server in **September 2018**.
* Azure DevOps Server retained the core capabilities but with the additional enhancement and integration with Azure Services.

## Why Version Control?

* Whether your software is large or small version control system is consisting of software which is helps us to trace the consistent changes in our project code.
* Azure DevOps Services provide two version control:
  + Git (Distributed version Control)
  + TFVC (Centralized Version Control)
* Without version control, our code has multiple copies of code on our computer system.

## TFVC (Team Foundation Version Control)

* + Team Foundation version control is a centralized version control System.
  + Typically, team member has only one version of each file on their dev machine.
  + Historical Data Only maintained on server. Branches are path-based and created on servers.
  + It has two workflow models:
    - Server Workspace
    - Local Workspace

## How to configure TFVC visual Studio

1. In visual studio we have option called Team Explorer which we use (short cut: ctrl+\, crl+m)
2. After that click on connect it’s giving the list of projects. If not then go to manage connection and configure the visual studio for TFVC.
3. Then configure the workspace. Then confirm your path and hit Map&Get to map the workspace to your project.
4. Once, you configure the version control after that to need to add your solution to the mapped path and then add the solution to the source control through the menu.
5. After that you need to check–in the solution for that right click on the solution and select the check-in option. Now you can see that solution on source control page.
6. While check-in file in old file once compares both the files old and new file and resolve the conflicts and then check-in file so, everyone get updated file.

## TFVC Code Review in Visual Studio

* + Before, we check in code we can ask someone else to review our code. We can do request from visual studio.
  + For the code Review Request:
    - go to the My Work page in Team Explorer
    - Select Request Review
    - After that on New Code Review page we can select the reviewer’s name and add subject for the code review and then submit the request.
    - That’s all we need to do suspend the code so, it’s saved one server and we can do work on different task. while you wait to hear back from the code review.

## Use Local or Server Workspace

* In TFVC, when we create or edit workspace in visual studio, we can specify whether it’s location is **server** or **local**.
* A **local workspace** caches the unmodified version of each of your files to enable you to edit, compare, and do other things without being connected to the server.
* We can change the location from server to local and local to server whenever we need.

### Use Local Workspace

* + - "**Local workspace**" refers to a feature that allows developers to work with version-controlled files on their local machines.
    - In Local workspace, the source control files are stored on the developer machine which is managed locally.
    - That means developer not require a network connection for such operation checking out, view and editing.
    - Local workspace providing several advantages over the server workspace:
      1. They provide faster and more responsive operation because it performed most of the operation locally.
      2. They also enable better support for file operations outside of Visual Studio, as the files are accessible through regular file system operations.
    - **Note**:
      1. If we are working with Azure DevOps and newer version of Microsoft version control system then, the concept of local workspace is not applied.

### Server Workspace

* + - server workspace when the specific conditions aren't met for using a local workspace, or if you use the **Enable get latest on check-out** option.

#### Work on server workspace while disconnected

* + - You can still work in a server workspace if you're temporarily disconnected from the TFVC server by taking the solution offline.
    - Later when the connection is available, you can take the solution online to return to the connected behavior.

## Check-in Check-out

* In Team Foundation Version Control (TFVC), the terms "check in" and "check out" are used to describe the process of managing changes to version-controlled files.

### Check-out

* + - When you "check out" a file, it means you are requesting exclusive write access to that file. Checking out prevents others from making changes to the file until you check it back in.
    - By checking out a file, you are essentially reserving it for yourself, indicating that you intend to make modifications to it.
    - Once you have checked out a file, you can edit it locally on your machine, apply changes, and save the modifications.
    - For Check-Out follow below steps:
      1. Open the TFVC-enabled IDE (e.g., Visual Studio).
      2. Connect to your TFVC server and open the project/solution containing the file you want to check out.
      3. In the Solution Explorer or Source Control Explorer, locate the file you wish to modify.
      4. Right-click on the file and select the "Check Out" option. This action requests exclusive write access to the file, marking it as checked out by you.
      5. The file will be downloaded to your local workspace, allowing you to make changes to it.

### Make Modification

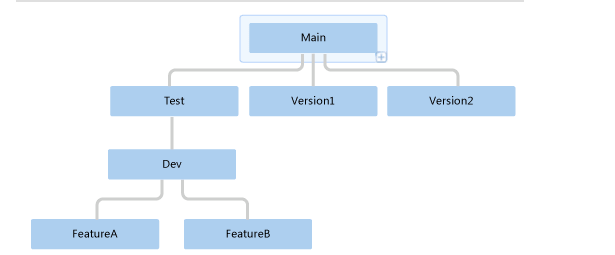
1. Open the checked-out file in your preferred editor within the IDE.
2. Apply the necessary changes to the file.
3. Save the modifications locally.

### Check-in

1. Once you have completed your changes, right-click on the file in the Solution Explorer or Source Control Explorer.
2. Choose the "Check In" or "Pending Changes" option.
3. A dialog or window will appear, presenting a list of pending changes you have made.
4. Review the list and provide a meaningful comment or description for the changes you made.
5. Optionally, associate the changes with a work item or bug, if applicable.
6. Click the "Check In" button to upload your changes to the TFVC server.

## Branching

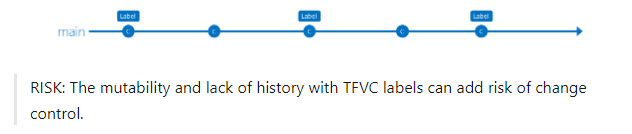
* You can use branches to accomplish the following goals:
  + Manage concurrent work by multiple teams on the same codebase.
  + Isolate risks that are introduced by different sets of changes to the codebase.
  + Take snapshots and then support subsequent isolated changes (for example, to create a release branch).



* Branching is an important and powerful technique for creating a parallel set of versions of your files.
* Unlike Git branches, which are repository scoped, TFVC branches are path scoped and not as lightweight.

### Main Only

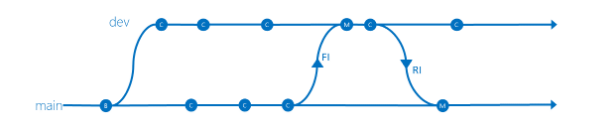
* The Main Only strategy can be folder-based or with the main folder converted to a Branch, to enable additional visibility features.



* You commit your changes to the main branch and optionally indicate development and release milestones with labels.

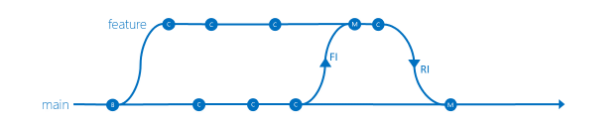
### Development Isolation

* When you need to maintain and protect a stable main branch, you can branch one or more dev branches from main.
* It enables isolation and concurrent development.



* work can be isolated in development branches by feature, organization, or temporary collaboration.
* It's possible that there are changes in the main branch.
* Always forward integrate (FI) main to the dev branch and resolve merge conflicts. Then reverse integrate (RI) changes back to main.
* Maintain the same quality bar across branches. Build and run build verification tests (BVTs) on dev the same way you are doing on main.
  + **NOTE**: With this strategy, teams are likely to keep the **dev** branch around forever, potentially building a large merge ticket history.

### Feature Isolation

* Feature isolation is a special derivation of the development isolation, allowing you to branch one or more feature branches from main, as shown, or from your dev branches.
* When you need to work on a particular feature, it might be a good idea to create a feature branch.
* Keep the lifetime of feature work and the associated feature branch short-lived.
* Forward integrate (FI) changes from the parent branch frequently. Reverse integrates (RI) back to the parent only when some agreed team criteria, for example definition of done, is met
* Rollback of features on main can be costly and may reset testing.

## Changeset Tab

* + The **Changesets** tab displays detailed data about every change that has been made to an item.
  + All changesets that have modified the item are listed in a table that includes the following columns:
    - **Changeset**: The ID of the changeset.
    - **User**: The name of the user who checked in the changeset.
    - **Date**: The date and time the changeset was checked in.
    - **Comment**: The comment entered by the user who checked in the changeset.

## Multiple Checkout

* In Team Foundation Version Control (TFVC), multiple checkouts, also known as concurrent checkouts, allow multiple developers to check out and edit the same file simultaneously.
* This feature is useful when collaboration and parallel development are required.
* To use this feature follow below steps for configure setting in visual studio:

1. In Visual Studio Team Explorer, connect to the project for which you want to configure check-out settings.
2. Under Project, select Settings.
3. On the Settings page, in the Team Project section, select Source Control.
4. In the Source Control Settings dialog box, on the Check-out Settings tab, select or clear the Enable multiple check-out checkbox.
5. Select or clear the Enable get latest on check-out checkbox.
6. Select OK.

## Reference

* + [How to Setup](https://learn.microsoft.com/en-us/azure/devops/repos/tfvc/set-up-team-foundation-version-control-your-dev-machine?view=azure-devops)
  + [What is TFS](https://www.perforce.com/blog/vcs/what-team-foundation-server#how)
  + [TFS Blog From GeeksForGeeks](https://www.geeksforgeeks.org/what-is-team-foundation-server/)
  + [Summary Chat](https://chat.openai.com/share/b7e5b461-605d-4597-abb8-7a2e04427453)
  + [Team Foundation Version Control](https://learn.microsoft.com/en-us/azure/devops/repos/tfvc/what-is-tfvc?view=azure-devops)
  + [Setup and Use TFVC on visual studio](https://learn.microsoft.com/en-us/azure/devops/repos/tfvc/share-your-code-in-tfvc-vs?view=azure-devops)
  + [Merge, Resolve Conflict. Changeset](https://learn.microsoft.com/en-us/azure/devops/repos/tfvc/get-history-item?view=azure-devops)
  + [Code Review](https://learn.microsoft.com/en-us/azure/devops/repos/tfvc/get-code-reviewed-vs?view=azure-devops)
  + [Check-in Check-out](https://learn.microsoft.com/en-us/azure/devops/repos/tfvc/share-your-code-in-tfvc-vs?view=azure-devops)
  + [Branching](https://learn.microsoft.com/en-us/azure/devops/repos/tfvc/branch-folders-files?view=azure-devops)
  + [Multiple Checkout](https://learn.microsoft.com/en-us/azure/devops/repos/tfvc/configure-check-out-settings?view=azure-devops)