**Microsoft Azure** (formerly **Windows Azure** [/ˈæʒər/](https://en.wikipedia.org/wiki/Help:IPA/English)) is a [cloud computing](https://en.wikipedia.org/wiki/Cloud_computing) service created by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) for building, testing, deploying, and managing applications and services through Microsoft-managed [data centers](https://en.wikipedia.org/wiki/Data_center). It provides [software as a service (SaaS)](https://en.wikipedia.org/wiki/Software_as_a_service), [platform as a service (PaaS)](https://en.wikipedia.org/wiki/Platform_as_a_service) and [infrastructure as a service (IaaS)](https://en.wikipedia.org/wiki/Infrastructure_as_a_service)and supports many different [programming languages](https://en.wikipedia.org/wiki/Programming_language), tools and frameworks, including both Microsoft-specific and third-party software and systems.

Design

Microsoft Azure uses a specialized operating system, called Microsoft Azure, to run its "fabric layer": a cluster hosted at Microsoft's data centers that manages computing and storage resources of the computers and provisions the resources (or a subset of them) to applications running on top of Microsoft Azure. Microsoft Azure has been described as a "cloud layer" on top of a number of Windows Server systems, which use Windows Server 2008 and a customized version of [Hyper-V](https://en.wikipedia.org/wiki/Hyper-V), known as the Microsoft Azure Hypervisor to provide virtualization of services.[[32]](https://en.wikipedia.org/wiki/Microsoft_Azure#cite_note-32)

Scaling and reliability are controlled by the Microsoft Azure Fabric Controller, which ensures the services and environment do not fail if one or more of the servers fails within the Microsoft [data center](https://en.wikipedia.org/wiki/Data_center), and which also provides the management of the user's Web application such as memory allocation and load balancing.[[34]](https://en.wikipedia.org/wiki/Microsoft_Azure#cite_note-34)

Azure provides an [API](https://en.wikipedia.org/wiki/Application_programming_interface) built on [REST](https://en.wikipedia.org/wiki/Representational_state_transfer), [HTTP](https://en.wikipedia.org/wiki/HTTP), and [XML](https://en.wikipedia.org/wiki/XML) that allows a developer to interact with the services provided by Microsoft Azure. Microsoft also provides a client-side managed class library that encapsulates the functions of interacting with the services. It also integrates with [Microsoft Visual Studio](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio), [Git](https://en.wikipedia.org/wiki/Git_(software)), and [Eclipse](https://en.wikipedia.org/wiki/Eclipse_(software)).[[35]](https://en.wikipedia.org/wiki/Microsoft_Azure#cite_note-35)[[36]](https://en.wikipedia.org/wiki/Microsoft_Azure#cite_note-36)[[37]](https://en.wikipedia.org/wiki/Microsoft_Azure#cite_note-37)

In addition to interacting with services via API, users can manage Azure services using the Web-based Azure Portal, which reached General Availability in December 2015. The portal allows users to browse active resources, modify settings, launch new resources, and view basic monitoring data from active virtual machines and services. More advanced Azure management services are available.

**Azure Pipelines** helps you implement a build, test, and deployment pipeline for any app. configure and manage continuous integration and continuous delivery (CI/CD) for the app and platform of your choice.

**Team Foundation Server** (commonly abbreviated to **TFS**) is a [Microsoft](https://en.wikipedia.org/wiki/Microsoft) product that provides [source code management](https://en.wikipedia.org/wiki/Revision_control) (either with Team Foundation Version Control or [Git](https://en.wikipedia.org/wiki/Git_(software))), reporting, [requirements management](https://en.wikipedia.org/wiki/Requirements_management), [project management](https://en.wikipedia.org/wiki/Project_management) (for both [agile software development](https://en.wikipedia.org/wiki/Agile_software_development) and [waterfall teams](https://en.wikipedia.org/wiki/Waterfall_model)), automated builds, lab management, [testing](https://en.wikipedia.org/wiki/Software_testing) and [release management](https://en.wikipedia.org/wiki/Release_management) capabilities. It covers the entire [application lifecycle](https://en.wikipedia.org/wiki/Application_Lifecycle_Management), and enables [DevOps](https://en.wikipedia.org/wiki/DevOps) capabilities.[[1]](https://en.wikipedia.org/wiki/Team_Foundation_Server#cite_note-1) TFS can be used as a back-end to numerous [integrated development environments](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDEs) but is tailored for [Microsoft Visual Studio](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio) and [Eclipse](https://en.wikipedia.org/wiki/Eclipse_(software)) on all platforms.[[2]](https://en.wikipedia.org/wiki/Team_Foundation_Server#cite_note-2)

## Architecture[[edit](https://en.wikipedia.org/w/index.php?title=Team_Foundation_Server&action=edit&section=2)]

### Server architecture**[**[**edit**](https://en.wikipedia.org/w/index.php?title=Team_Foundation_Server&action=edit&section=3)**]**

Team Foundation Server is built on [multi-tier](https://en.wikipedia.org/wiki/Multitier_architecture), scalable architecture. The primary structure consists of an application tier responsible for processing logic and maintaining the web application portal (referred to as Team Web Access or TWA). TFS is built using [Windows Communication Foundation](https://en.wikipedia.org/wiki/Windows_Communication_Foundation) web services. These may be consumed by any client, although the client object model is recommended. The data tier and application tier can exist on the same machine.

To support scalability, the application tier can be load balanced and the data tier can be clustered. If using [Microsoft SQL Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server) 2012 or later, AlwaysOn SQL Server Failover Clusters and Availability Groups are supported which allows for geographic replication of data.[[4]](https://en.wikipedia.org/wiki/Team_Foundation_Server#cite_note-4) The primary container is the project collection. A project collection is a database that contains a group of Team Projects. The Project Collection is another scalability mechanism, in that each collection can be placed on different SQL Servers or SQL Server instances. 'Oe' configuration database per TFS instance stores project collection metadata. Data from the project collection databases is aggregated into the warehouse database, which denormalizes the data in preparation for loading into an Analysis Services cube. The warehouse and the cube allow complex trend reporting and data analysis.

TFS can integrate with an existing [SharePoint](https://en.wikipedia.org/wiki/SharePoint) farm. SQL Server Reporting Services are supported for more advanced reporting against the data warehouse or the Analysis Services data cube. These installations can be on the same system or on different systems. Build servers, lab management servers, release management servers and proxy servers (to reduce some of the load on the application tier), test machines and load test machines can also be added to the infrastructure.[[5]](https://en.wikipedia.org/wiki/Team_Foundation_Server#cite_note-5) To support teams requiring enterprise project scheduling, TFS also integrates with [Microsoft Project Server](https://en.wikipedia.org/wiki/Microsoft_Project_Server), which allows enterprise level portfolio management, resource management and project tracking.

## Source control[[edit](https://en.wikipedia.org/w/index.php?title=Team_Foundation_Server&action=edit&section=7)]

Team Foundation Server supports two different types of [source control](https://en.wikipedia.org/wiki/Source_control) - its original source control engine called Team Foundation Version Control (TFVC) and with the release of TFS 2013, it supports [Git](https://en.wikipedia.org/wiki/Team_Foundation_Server#Git) as a core source control repository.

### Team Foundation Version Control

TFVC is a centralized version control system allowing teams to store any type of artifact within its repository. TFVC supports two different types of workspaces when working with client tools - Server Workspaces and Local Workspaces.[[12]](https://en.wikipedia.org/wiki/Team_Foundation_Server#cite_note-12) Server workspaces allow developers to lock files for check-out and provide notification to other developers that files are being edited. A frequent complaint for this model is that files on the development machine are marked as read-only. It also requires developers to "go offline" when the server can't be contacted. Local workspaces were designed to avoid these problems. In a local workspace scenario files are not read-only and they do not have to be checked out before working on them. As long as the files are on the developer's local machine, it doesn't matter if the server is connected or not. Conflicts are dealt with at [check-in](https://en.wikipedia.org/wiki/Revision_control) time.

To improve performance for remote clients, TFS includes the ability to install [Proxy Servers](https://en.wikipedia.org/wiki/Proxy_Server).[[13]](https://en.wikipedia.org/wiki/Team_Foundation_Server#cite_note-13) Proxy servers allow source control contents to be cached at a site closer to the developers to avoid long network trips and the associated latency. Check-ins are still performed directly against the TFS application tier so the Proxy Server is most beneficial in read scenarios.

As part of the source control engine, TFS supports a number of features to help developers ensure the code that is checked in follows configurable rules. This rule engine is called a Check-in Policy. There are several out of the box policies such as the Changeset Comments Policy which will not allow a check-in unless the developer enters a check-in comment. These policies are extensible and can be used to examine all aspects of the code being checked in, the comments and the related work items. TFS also supports a Code Analysis feature that when used independently is known as [FxCop](https://en.wikipedia.org/wiki/FxCop). The inclusion in TFS means that the analysis can run against code checked into the server and during automated builds.

### Git

With the release of TFS 2013, Microsoft added native support for [Git](https://en.wikipedia.org/wiki/Git_(software)). This is not a Microsoft specific implementation but a standard implementation based on the libgit2[[14]](https://en.wikipedia.org/wiki/Team_Foundation_Server#cite_note-14) library. This is the same library that powers the popular [GitHub](https://en.wikipedia.org/wiki/GitHub) and the code is freely available from GitHub. Because Microsoft took the approach of using a standard library, any Git client can now be used natively with TFS (in other words, developers can use their favorite tools and never install the standard TFS clients). This allows tools on any platform and any IDE that support Git to connect to TFS. For example, both [Xcode](https://en.wikipedia.org/wiki/Xcode) and [Android Studio](https://en.wikipedia.org/wiki/Android_Studio) support Git plug-ins. In addition, if developers do not want to use Microsoft's Team Explorer Everywhere plug-in for [Eclipse](https://en.wikipedia.org/wiki/Eclipse_(software)), they can choose to use eGit[[15]](https://en.wikipedia.org/wiki/Team_Foundation_Server#cite_note-15) to connect to TFS.

Using Git does not preclude the benefit of using TFS work item or build system. When checking code in with Git, referencing the work item ID in the check-in comment will associate the check-in with the given work item. Likewise, Team Build will also build Git projects.

One of the major reasons to use TFS as a Git repository is that it is backed by SQL Server and is afforded the same protection as Team Foundation Version Control. This gives developers some choices when choosing the type of project and work style that works best for them.