What is Team Foundation Server?

**Team Foundation Server** (often abbreviated to **TFS**) is a Microsoft product that provides tools and technologies designed to help **teams** collaborate and orchestrate their efforts to finish projects or create a product. It enables DevOps capabilities covering the entire application lifecycle.

Since it was launched in 2005, **Team Foundation** Server (**TFS**) has evolved significantly until Microsoft launched Azure DevOps in 2018. As a part of the broader shift to cloud services, the company **renamed** the cloud-based Visual Studio Team Services (VSTS) to Azure DevOps Services and **TFS** as Azure DevOps Server.

## **What Is Source Control?**

Source control refers to tracking and managing changes to code. This ensures that developers are always working on the right version of source code.

### **Source Control vs. Version Control**

These two terms are used interchangeably. However, source control is specific to source code. [Version control](https://www.perforce.com/blog/vcs/what-is-version-control" \t "https://www.perforce.com/blog/vcs/_blank) also covers large binary files and digital assets.

## **What Is Source Control Management?**

Source control management (SCM) refers to tools that help you keep track of your code with a complete history of changes.

### **Source Control vs. Source Code Management (SCM)**

These two terms are also used interchangeably. However, source control is specific to tracking changes in source code. And source code management (SCM) tracks changes to a source code repository.

SCM also maintains a history of changes. This is used to resolve conflicts when merging updates from multiple developers.

### **Free Source Control**

Looking to get the best tool free? Now you can. Try [Helix Core from Perforce](https://www.perforce.com/products/helix-core) free for up to 5 users.

**[GET FREE TOOLS](https://www.perforce.com/products/helix-core/free-version-control)**

## **Why Source Control Is Important**

Source control is important for maintaining a single source of truth for development teams. Plus, using it helps facilitate collaboration and accelerates release velocity.

That’s because it allows multiple developers to work on the same codebase. They can commit and merge code without conflicts. And they can even make edits to shared code, without unknowingly overwriting each other’s work.

**Build Agent**

A build agent does the processor-intensive work of your build, which includes getting files from version control, provisioning the workspace, compiling the code, running tests, and merging files back into version control. Each build agent is dedicated to a single build controller.

Build Agent controller

which pools and manages the services of one or more build agents. It distributes processor-intensive work: for example, compiling code or running tests, etc., to the build agents. Each build controller is dedicated to a single team project collection.

## Definition

CI and CD stand for continuous integration and continuous delivery/continuous deployment. In very simple terms, CI is a modern software development practice in which incremental code changes are made frequently and reliably. Automated build-and-test steps triggered by CI ensure that code changes being merged into the repository are reliable. The code is then delivered quickly and seamlessly as a part of the CD process. In the software world, the CI/CD pipeline refers to the automation that enables incremental code changes from developers’ desktops to be delivered quickly and reliably to production.

## Why is CI/CD important?

CI/CD allows organizations to ship software quickly and efficiently. CI/CD facilitates an effective process for getting products to market faster than ever before, continuously delivering code into production, and ensuring an ongoing flow of new features and bug fixes via the most efficient delivery method.

## What is the difference between CI and CD?

Continuous integration (CI) is practice that involves developers making small changes and checks to their code. Due to the scale of requirements and the number of steps involved, this process is automated to ensure that teams can build, test, and package their applications in a reliable and repeatable way. [CI](https://www.synopsys.com/glossary/what-is-continuous-integration.html) helps streamline code changes, thereby increasing time for developers to make changes and contribute to improved software.

[Continuous delivery](https://www.synopsys.com/glossary/what-is-continuous-delivery.html" \t "https://www.synopsys.com/glossary/_self) (CD) is the automated delivery of completed code to environments like testing and development. CD provides an automated and consistent way for code to be delivered to these environments.

[Continuous deployment](https://www.synopsys.com/glossary/what-is-continuous-development.html" \t "https://www.synopsys.com/glossary/_self) is the next step of continuous delivery. Every change that passes the automated tests is automatically placed in production, resulting in many production deployments.

Continuous deployment should be the goal of most companies that are not constrained by regulatory or other requirements.

In short, CI is a set of practices performed *as developers are writing* code, and CD is a set of practices performed *after* the code is completed.

## How does CI/CD relate to DevOps?

DevOps is a set of practices and tools designed to increase an organization’s ability to deliver applications and services faster than traditional software development processes. The increased speed of DevOps helps an organization serve its customers more successfully and be more competitive in the market. In a DevOps environment, successful organizations “bake security in” to all phases of the development life cycle, a practice called *[DevSecOps](https://www.synopsys.com/software-integrity/solutions/devsecops.html" \t "https://www.synopsys.com/glossary/_self)*.

The key practice of DevSecOps is integrating security into all DevOps workflows. By conducting security activities early and consistently throughout the software development life cycle ([SDLC](https://www.synopsys.com/glossary/what-is-sdlc.html" \t "https://www.synopsys.com/glossary/_self)), organizations can ensure that they catch vulnerabilities as early as possible, and are better able to make informed decisions about risk and mitigation. In more traditional security practices, security is not addressed until the production stage, which is no longer compatible with the faster and more agile DevOps approach. Today, security tools must fit seamlessly into the developer workflow and the CI/CD pipeline in order to keep pace with DevOps and not slow development velocity.

The CI/CD pipeline is part of the broader DevOps/DevSecOps framework. In order to successfully implement and run a CI/CD pipeline, organizations need tools to prevent points of friction that slow down integration and delivery. Teams require an integrated toolchain of technologies to facilitate collaborative and unimpeded development efforts.

窗体顶端