Continuous Integration with TeamCity

On this page:

What is Continuous Integration?

Continuous Integration is a software development practice in which developers commit code changes into a shared repository several times a day. Each commit is followed by an automated build to ensure that new changes integrate well into the existing code base and to detect problems early. To learn more about continuous integration basics, please refer to Martin Fowler's article.

What is TeamCity?

JetBrains TeamCity is a user-friendly continuous integration (CI) server for developers and build engineers free of charge with the Professional Server License and easy to set up!

What can you do with TeamCity?

- Run parallel builds simultaneously on different platforms and environments
- Optimize the code integration cycle and be sure you never get broken code in the repository
- Review on-the-fly test results reporting with intelligent tests re-ordering
- Run code coverage and duplicates finder for Java and .NET
- Customize statistics on build duration, success rate, code quality, and custom metrics
- and much more.

To learn more about major TeamCity features, refer to the official JetBrains site.
The complete list of supported platforms and environments can be found here.

Basic TeamCity concepts

This section explains the main concepts. The complete list can be found here.

The TeamCity build system comprises the server and Build Agents.

Concept	Description
Build Agent	A piece of software that actually executes a build process. It is installed and configured separately from the TeamCity server, i.e. the agent can be installed on a separate machine (physical or virtual, and it can run the same operating system (OS) as the server or a different OS. Build Agents in TeamCity can have different platforms, operating systems, and pre-configured environments that you may want to test your software on. Different types of tests can be run under different platforms simultaneously so the developers get faster feedback and more reliable testing results. ⚠ It is possible for the server and an agent to coexist on the same computer, but for production purposes we recommend installing them on different machines for a number of reasons, the server performance being the most important.
TeamCity Server	The server itself does not run either builds or tests: the server's job is to monitor all the connected build agents, distribute queued builds to the agents based on compatibility requirements, and report the results. All information on the build results (build history and all the build-associated data except for artifacts and build logs), VCS changes, agents, build queue, user accounts and user permissions, etc. are stored in a database. ⚠ It is possible for the server and an agent to coexist on the same computer, but for production purposes we recommend installing them on different machines for a number of reasons, the server performance being the most important.

Project	A TeamCity Project corresponds to a software project, or a specific version/release of a software project. A project is a collection of build configurations.
Build Configuration	A combination of settings defining a build procedure. The settings include VCS Roots, Build Steps, Build Triggers described below.
VCS Root	A collection of version control settings (paths to sources, username, password, checkout mode and other settings) that defines how TeamCity communicates with a version control (SCM) system to monitor changes and get sources for a build.
Build Step	A task to be executed. Each build step is represented by a build runner providing integration with a specific build tool (like Ant, Gradle, MSBuild, etc), a testing framework (e.g. NUnit), or a code analysis engine. Thus, in a single build you can have several steps and sequentially invoke test tools, code coverage, and, for instance, compile your project.
Build Trigger	A rule which initiates a new build on certain events. For example, a VCS trigger will automatically start a new build each time TeamCity detects a change in the configured VCS roots.
Change	Any modification of the source code which you introduce. If a change has been committed to the version control system, but not yet included in a build, it is considered pending for a certain build configuration.
Build	Refers to both: the actual process of creating an application version and the version itself. After the build process is triggered, it is put into the build queue and is started when there are agents available to run it. After the build is finished, the build agent sends Build Artifacts to the server.
Build Queue	A list of builds that were triggered and are waiting to be started. TeamCity will distribute them to compatible build agents as soon as the agents become idle. A queued build is assigned to an agent at the moment when it is started on the agent; no pre-assignment is made while the build is waiting in the build queue.
Build Artifacts	Files produced by a build, for example, installers, WAR files, reports, log files, etc, when they become available for download.

Basic CI Workflow in TeamCity

To understand the data flow between the server and the agents, what is passed to the agents, how and when TeamCity gets the results, let's take a look at a simple build lifecycle.

a. The TeamCity server detects a change in your VCS Root and stores it in the database.
b. The build trigger sees the change in the database and adds a build to the queue.
c. The server finds an idle compatible build agent and assigns the queued build to this agent.
d. The agent executes the Build Steps. While the build steps are being executed, the build agent reports the build progress to the TeamCity server sending all the log messages, test reports, code coverage results, etc. to the server on the fly, so you can monitor the build process in real time.
e. After finishing the build, the build agent sends Build Artifacts to the server.

Now you can proceed with TeamCity installation.