

Welcome Message

Thank you for your interest in reviewing TeamCity 3.0, an innovative continuous integration server and distributed build management tool that automates routine processes and streamlines the development process while assisting teams implement the industry's best practices and improve communication.

This guide is not an in-depth roadmap to TeamCity's functionality, but rather a quick overview of the program that highlights key beneficial features so you will be able to write an informed review in a minimal amount of time.

Overview

This section provides an "in-a-nutshell" description of TeamCity 3.0 with short descriptions of each of the main feature areas.

Company Mission

JetBrains' flagship product, IntelliJ IDEA, has been widely praised by both industry pundits and developers alike for the productivity gains it helps individual software developers achieve through its "intelligent assistant" approach which automates and/or eliminates many peripheral, redundant, time-consuming tasks that are necessary to handle, but that distract from the central objective of creating code. JetBrains' products are produced by and for professional developers with this goal in mind.

Introduction

TeamCity's fundamental premise is to identify necessary but redundant or time-consuming things in the team development process that detract from overall team productivity. TeamCity provides intelligent, automated solutions to reduce or eliminate human error, time-eating analyses, miscommunication, and other problems that can bog down teams and contribute to missed targets and poor morale.

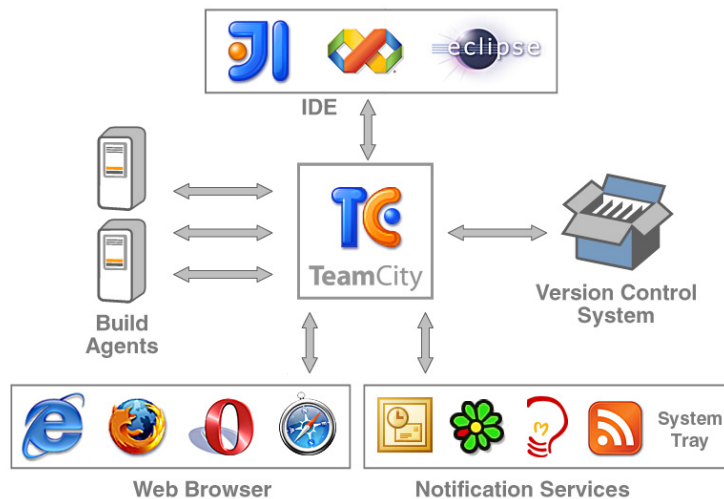
The major challenging areas that TeamCity addresses are:

- Continuous integration and testing
- Software build automation and management
- Team communication relating to both of the above

Along with these issues, TeamCity provides integrated and automated solutions for code inspection and code coverage analysis that can serve to reduce the chances of problems occurring in the areas mentioned above. In addition, there are features for integrating and extending TeamCity in different environments. Following sections will provide an overview of all these feature areas.

Speaking of environments, TeamCity is independent of any IDE, and the server component can run on Windows, UNIX/Linux or Mac. A web browser is the only client software required, although there are integration plugins for IntelliJ IDEA, Eclipse and Microsoft Visual Studio that enable the IDE to serve as a TeamCity client providing the most-needed features for developers, plus some extended features (more about these later). For Windows users, there is also a separately-installable component that displays TeamCity notifications in the Windows system tray.

How TeamCity Pulls All the Pieces Together



The basic unit of work of TeamCity is a software *build*. To build your software using TeamCity, you need to have at least one computer (referred to as a “Build Agent”) with a set of predefined parameters (system and environment variables, JDK version, .NET Framework version, etc).

Each software build has its own *Build Configuration* – a set of actions and parameters that serve to control all aspects of the build. The Build Configuration specifies things like the version control system settings, the build runner (Ant for example) the build schedule (daily, weekly, specific time), and properties and environment variables needed to run the build.

To be able to run a build, a Build Agent must meet the Build Configuration’s requirements. If there is a *free* Build Agent matching the Build Configuration requirements at the time the build is triggered, the build is started. Otherwise, the build enters the *build queue* and is processed when a compatible build agent becomes available.

The TeamCity server interacts with:

- The VCS (Version Control System) used to manage your project's source file revisions
- The Build Agents on which the builds run
- A web browser in which you can monitor builds (among other things)
- An IDE where you modify your source code

Main Feature Areas

Let's proceed now to a quick look at each of TeamCity 3.0’s main feature areas.

Continuous Integration and Build Monitoring

Most development organizations acknowledge the value of having continuous and automated test/build operations many times each day. There are many tools that automate continuous integration, but TeamCity is constantly innovating new techniques to perfect the process and help teams painlessly implement the industry’s best practices.

One of TeamCity's innovations is its unique *Pre-tested Commit* feature that allows teams to always keep the code in their version control repository error-free and running. Typically developers submit their changes to version control *before* verifying that it works. TeamCity revolutionizes the Edit-Commit-Verify process, replacing it with Edit-Verify-Commit. When running a *Pre-tested Commit* changes are tested remotely, if they pass the tests the changes are *automatically* committed to version control, if they fail nothing is committed and the developer is notified and can start fixing the code.

With TeamCity, team members can *remotely* test and integrate their code changes many times per day. Once configured (configuration is intuitive and straightforward), TeamCity takes care of complex test running, error handling and notifications, and integration chores. All a developer needs to do is submit changes to version control and continue working. Tests run remotely so the developer's computer isn't bogged down. If tests fail, the developer is notified immediately and can take corrective action.

Developers can monitor the status of their own changes on the *My Changes* page. The page is simple, intuitive and is a one-stop shop for useful information like a list of the developers' modifications, check-in comments, and build results (shown in real-time if a build is in progress). Viewing file diffs, test results, the build log, a list of failed tests, and a list of previously failed tests that were recently fixed and more detailed build information are all just a click away.

TeamCity makes it easy to monitor builds, by providing real-time build results which include a color coded scheme to let you know if the build failed any tests, and a progress bar with an estimated time until completion. When the build is running developers can stop the build and view a thread dump in a popup window. TeamCity also assists with troubleshooting by notifying you of builds that are hanging.

Version Control Integration

Many features like viewing file diffs and tracking changes wouldn't be possible without tight integration with version control. TeamCity integrates with Borland StarTeam, CVS, IBM Rational ClearCase (Base and UCM), Microsoft Visual SafeSource, Perforce, Subversion, and Team Foundation Server to automatically handling check-in of submitted code and updating sources for building.

Common problem: As a rule, continuous integration teams want to trigger a new builds as soon a new code is checked in, but sometimes such frequent triggering this isn't advantageous and can needlessly waste resources creating bottlenecks.

TeamCity's solution: TeamCity has a number of flexible features to help teams efficiently manage version control. Check out rules can be configured to prevent unwanted triggering when changes are submitted to a certain parts of the repository (e.g. sources/docs), certain types of files are checked in (e.g. *.html, *.doc), or when a particular user commits changes.

TeamCity's flexible checkout rules can also help improve efficiency by excluding unneeded sources or map paths to different directories on the Build Agent during the check out process.

TeamCity can also optionally add a label into Version Control for the sources used for a particular build. This is useful when the sources of a particular build need to be collected to remake the build.

The program also streamlines project setup by "sharing" VCS sources and settings (called VCS roots) with different build configurations and projects. Just select a checkbox to share the VCS root and it will be available in a drop-down menu in all your projects. TeamCity monitors changes made to these sources, and if the changes might disturb one of your other projects, TeamCity will prompt you to make a quick copy of your sources.

Software Build Automation and Build Management (Build Grid)

TeamCity uses a pool of computers (called “Build Agents”) to simultaneously create multiple builds of your projects, using different build configurations. This approach to the intelligent distribution of builds to multiple machines is a technology we call the Build Grid. All of the Build Agents on the grid are managed from the TeamCity server, which provides real-time build results. Build Agents can use different operating systems and platforms, have some predefined parameters that make it compatible for creating certain types of builds and testing certain changes.

As developers integrate their work results into the Version Control System, the TeamCity server automatically distributes the updated sources to Build Agents that are capable of running the build, so there is no need to install version control software on all of your build agents. Build Agents also automatically update themselves, thus further saving the system administrator’s time.

With TeamCity you can trigger builds whenever changes are checked into the source control system (as mentioned above), manually (via the web interface or your IDE), periodically (to create daily or nightly builds), or when a build that is used by a different build configuration completes successfully. You can improve efficiency by using a combination of these conditions and exceptions, plus, specify useful options like starting a new build automatically when the last build failed, or prevent triggering using VCS checkout rules (mentioned in detail above).

If all of the compatible Build Agent’s are busy, the build enters the Build Queue and will be processed as soon as resources are available. The Build Queue can all easily be monitored in the web interface, where builds can be manually moved up or down in the queue.

Common problem: When there's the problem of failed builds and their aftermath. It's necessary to figure out what code broke the build, and who submitted problem code and when – usually a time consuming activity in all but the smallest teams. Once identified, the responsible developers have to be contacted, and they must fix bad code before the next build can happen. TeamCity all but eliminates this kind of morass.

TeamCity’s solution: When any submitted code breaks a build, the developer(s) who submitted it are immediately and automatically notified that their code may have broken the build. Team leaders and developers alike monitor build status in the web interface. For any failed build, they can check (in seconds, not hours) what changes were made by whom, both directly in the failed build, and since the last successful build.

Team Communication

Common problem: You know the scenario: the build is broken, but it takes a while for people to realize it, and then nobody is quite sure whose code broke it, but everybody assumes someone else is on top of the situation and that a fix will happen. Valuable time is lost sorting it all out.

No more of that with TeamCity.

TeamCity’s solution: First of all, the customizable array of notifications (with delivery options that include e-mail, Jabber/XMPP, Atom/RSS feeds, the Windows system tray, and IDE messages) ensures that anyone who submitted code that could have broken the build is notified the moment something fails. The failure shows up immediately in the web interface so anyone on the team can see the status (and keep an eye on any or all builds).

Then there's a very handy feature called *Take responsibility*. If nobody has taken on responsibility for a fix, this fact is evident for all to see at a glance in the Web interface (and a couple of other possible ways). The pain of the hapless developer whose code broke the build is assuaged with a handy link that enables him/her to not only take on the fix, but to effortlessly inform the entire team that someone is working on it, and also let them know when the fix is complete.

TeamCity's *external status widget* also makes it easy to extend the latest build information beyond the development team (like to company management and customers) by adding a small HTML fragment to any webpage.

Administration

Common problem: Managing your automated build process can be a daunting task involving editing settings in different programs, various scripts and configurations.

TeamCity's solution: TeamCity's feature-rich web interface enables efficient administration and automation of common user tasks. Related tasks are intuitively compartmentalized and organized simplifying the management of projects, build configurations, users, and build agents. Everything is straight forward and there are plenty of tooltips that provide additional information and guidance.

Setting up and configuring projects and build configurations are easy, just following a seven-step Q&A about your configuration and you're ready to go. You can also copy/clone your projects and build configurations.

Once installed, Build Agents are managed in the web interface. The *Agents* tab provides information on the Build Agents current status, availability, and properties, as well as the means to authorized/unauthorized them so they added to/removed from the build grid. Here agents can also be assigned a run configuration policy, which limits the number of build configurations the Agent can run, so they will be available for high priority build configurations.

The flexible build history clean-up policy helps keep Build Agents from getting weighed down with old build data (build logs, artifacts, etc.) by systematically automating the process of removing old unneeded data, while preserving required results.

User accounts are also created and managed in the web interface. The System Administrator assigns project-level roles and permissions to all of the users. Users configure their own VCS credentials and various notification settings, so they get just the information they need and don't suffer from information overload.

Code Quality

TeamCity has numerous code quality features to ensure the code base is constantly being thoroughly tested. Tests are performed on the server-side and any discovered problems can be viewed in the web interface, where in one click the questionable code can be launched in the developer's IDE. (This capability is provided with a special integration plugin discussed a bit later). TeamCity automates searching for code duplicates in both Java and .NET code. Code inspections include checks for unused and unreachable code, declaration redundancies, and general inspection of implementation code and can be run on both IntelliJ IDEA projects and Maven2 pom.xml files.

TeamCity can optionally run code coverage analysis for Ant and/or IntelliJ IDEA projects to determine the extent to which project source code is covered by unit tests. Just mark a checkbox when setting up a build configuration, and the build results page in the web interface will include an overview of coverage statistics and a link to a detailed report.

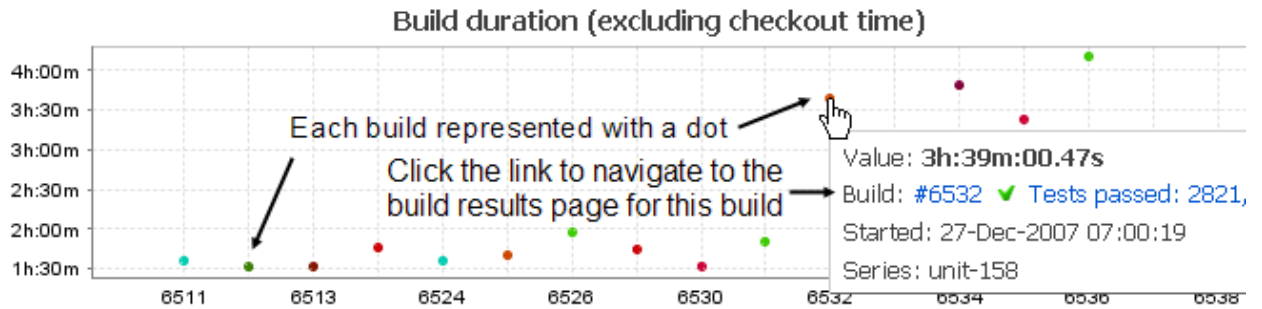
Statistics

Common problem: Bottlenecks often arise during the build process. They may be caused by something as simple as lots of builds stacking up in the Build Queue or something more complex like a change that was introduced that somehow greatly reduced build performance.

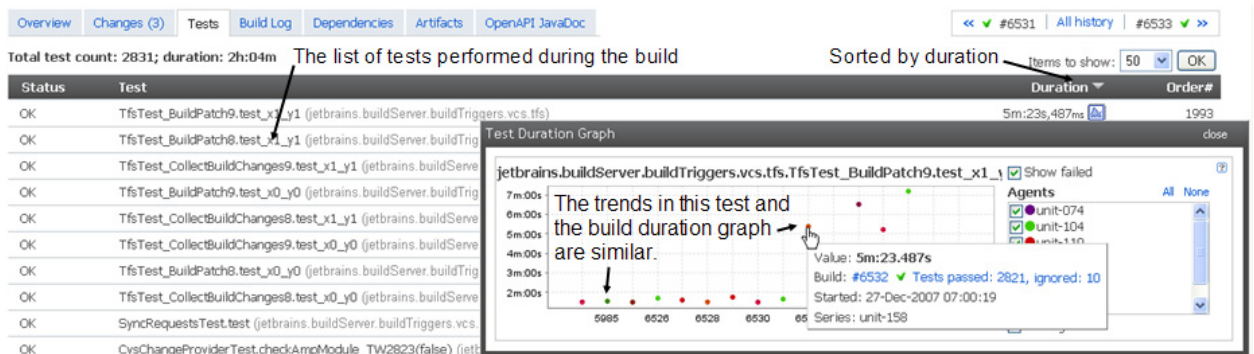
TeamCity Solution: As they say a picture is worth a thousand words. TeamCity provides numerous visual build metrics to track things like the build duration, successful build rate, time spent in the build queue, test count, the duration of individual tests, the number of duplicates, the number of errors and warnings found during inspection, and artifact size over time. All of these metrics provide valuable information that helps teams monitor and improve the efficiency of the build process.

When values in the time spent in build queue appear high, the administrator can improve performance by reducing the number of builds that are triggered, configure certain Build Agents to build only the most important builds, or considering licensing additional Build Agents.

The administrator might also notice a sudden jump in, say, a build duration value that remains high in subsequent builds (see the build duration graph below).



The administrator can investigate the cause of this by clicking on the first high value to navigate to the build results page of that build. From there the administrator can examine the different individual tests, conveniently sorting them by duration, and look for similar trends in tests that seem to be taking a long time view the tests tab and quickly sort the individual tests by duration, and look for similar trends (see the tests tab of the build results page with a test duration graph of a test below).



The administrator can now review the files diffs of this build and begin to investigate which change(s) caused the decrease in build performance.

Extension and Add-ons

TeamCity is extensible via a Java API (documentation is provided in the download archive). TeamCity is based on open protocols, supports industry-standard VCSs, bug tracking tools, and thus, guarantees no vendor lock-in. Anyone can easily develop and deploy custom server-side plugins and build agent plugins for TeamCity:

Add-ons can be downloaded from the *My Settings & Tools* page of the TeamCity web interface.

Windows Tray Notifier

This separately-installed extension enables Windows users to receive TeamCity notifications as pop-up messages from the Windows system tray which can be reviewed any time via the tray icon (which also provides access to build status information).

You can define notification rules in your personal settings that specify what kind(s) of notifications you want to receive and how they should be delivered.

TeamCity Plugins for IntelliJ IDEA, Eclipse and Visual Studio

IDE integration plugins are a must-have for any developer, because it improves efficiency by eliminating the need to switch to a web browser. The most essential functionality for developers – initiating builds into the queue, viewing current builds and build/test results, for example – is accessible inside the IDE, including notifications and build results. The plugin adds a *TeamCity* item to the IDE's main menu as well as TeamCity icons and toolbars.

Using your IDE as a TeamCity client is of course efficient, but the plugin also provides some highly useful extended functionality that can help each developer minimize the possibility of submitting code that causes problems in the integration process.

Personal Builds

Developers using the TeamCity plugin can create a *personal build* which is run remotely on TeamCity, and is just the same as a "regular" build (including automatic integration of all the latest sources in version control) except that the changed code is not committed to version control.

If a personal build fails, the "real" build is not affected because the code was not committed to version control, so there is no adverse impact on the team. The developer receives immediate notification and can then work on a fix, easily navigating to problem code right in the editor. Once the developer knows the code will build, he/she can commit it to version control and let the regular build process take its course, confident that there will not be any problems.

This feature, called *Remote Run*, significantly reduces the chances of any developer submitting build-breaking code.

Pre-tested Commits

It happens quite often: some member of the team submits changes that break the build, and goes home before it is discovered. This is a critical problem especially for geographically dispersed teams.

With TeamCity the software building process never stops. Developers can perform a "Pre-tested Commit" of the modified code right from their IDE. The developer's code modifications get integrated into the latest available sources of the project code in the VCS, and the tests are run remotely. If the tests pass, the modified code is *automatically* submitted to the VCS. And if the tests fail, the developer receives notifications, can fix the discovered problems in the IDE, but the *project code base stays clean*

Quick navigation between TeamCity's web interface and the developer's IDE.

When problems are discovered in the code, like when it fails tests, the build results in web interface will list the failed test(s), some detailed information, and provide a link labeled *open in IDE*. Clicking on this link will open the source code in your IDE, and navigate directly to the problem in the code. Likewise, developers can make a one-click jump from their IDE to the build results page, by using clicking the view build results button on the toolbar.

Note: JetBrains is constantly expanding the functionality of existing TeamCity plugins and extensions.

This concludes the overview. We understand that really get a feel for how the program works you really need to see it in action and for that reason we invite you visit our demonstration server. Login and other information will be provided to you separately.

Thank you for your interest and please don't hesitate to contact our marketing director, Ann Oreshnikova, <Ann.Oreshnikova@jetbrains.com> if you have any questions.