TeamCity in Devbridge

JetBRAINS

TeamCity

DEVBRIDGE® GROUP

Devbridge Group is an international software design and development company that builds comprehensive, custom solutions for enterprise mobile and web. Devbridge combines engineering expertise with elegant design aesthetic to deliver exceptional results for leaders in the manufacturing, healthcare, financial services and retail industries.

Devbridge Group produces more than 100,000 engineering hours annually, building custom cloud-based and mobile solutions. Long-term partnerships with Devbridge are built on trust and transparency. The company follows a streamlined agile process, is committed to Responsive Design, and always pursues excellence.

Pre-TeamCity

A few years ago, most of our projects were just plain .NET web applications, and using one CI server (CruiseControl.Net) was enough. But as the company grew, so did our client base, and we realized not everyone wanted their projects to be written in C#. We started offering solutions created with PHP, NodeJS, Objective-C, Java and other languages. We also began using other tools for CI. For instance, we used Jenkins, and it ended up causing some problems.

The first problem was that some teams used CC.NET, while others used Jenkins. Sharing reusable build steps was problematic, because they were either "shell scripts" or CI application dependent configurations, not to mention we were dealing with different OS installations (Windows, Linux, OS X) on each of our CI machines.

The second problem was the usage balance between our CI machines. Some machines were used for running development and testing builds, others were used for deployment. For us, development builds occur frequently, so the machine responsible for this was always at max capacity while the production build machine wasn't used nearly as much.

Deciding

After a frustrating time, we decided that we needed a single CI environment where we could reuse our existing CI machines. This would allow for a painless transition from other CI software.

We took Jenkins, Go CD, Atlassian Bamboo and JetBrains TeamCity and installed them on our test machine for comparison. Next, we created build configurations for already existing projects and checked for the following:

- **Simplicity of creating a build**

  To move everyone from their current CI software to the new one, we wanted it to have a simple and clear way of creating builds so that the learning curve would be as low as possible.

- **Build reusability (templates, etc.)**

  A lot of our projects use mostly the same build steps, so we need a way for creating templates and other ways of reusing those configurations.

- **Scalability**

  We wanted our CI machines to be easily managed, added and balanced for our builds. The ability to intelligently pick the right machine for the build was necessary.

- **Active Directory support and role management**

  At Devbridge Group, we use Active Directory to manage user accounts, groups and permissions. The CI software that we chose had to support authorization and, if possible, groups.

- **Ease of configuration and the overall look and feel**

  Using an ugly piece of software should be a thing of the past. We wanted the software to be easy to understand, aesthetically pleasing and responsive.

- **REST API**

  We needed the software to have a RESTful API so that we could connect it to our other monitoring software, re-run builds and so on.

Choosing TeamCity over others

The reason we picked TeamCity was because it had the best results for our necessary requirements.

- **Simplicity of creating a build**

  Creating builds in TeamCity was a breeze. TeamCity came bundled with most of the tools (runners) we needed for C# and JAVA projects. To create the build steps, we just needed to select a runner in correct order with one or two additional parameters. For PHP we just downloaded “meta runners” from the official JetBrains GitHub repository and created build steps. Additionally, we were pleased that it’s possible to create artifacts from a build and then use it as a Report page. For example, code coverage.

- **Build reusability**

  TeamCity has templates that make it possible to copy build configurations from one project to another. It’s possible to create beautiful, reusable templates and add configuration parameters so that when you create a new build from the template, you can specify certain data: build.xml location, GIT tag prefix and so on.

  Another great feature is the meta-runner. It allows the user to create smaller templates for a specific sequence of build steps. For example, you can create a Heroku deployment build step sequence, then extract them all into a single runner, and use it in other templates or even share it with the world (like JetBrains did with the meta-runner power-pack).

- **Scalability**

  TeamCity has agents that can be installed on different machines and will then connect to the central CI server software. The central software manages which agent will be used for each build. Additionally, you can define requirements for the agents to use. You can achieve this by using the parameters provided by each agent (you can also add your own).

  If you have more than one agent assigned, then your builds will be intelligently divided among those agents and the agents that don't meet your requirements will be automatically skipped.

- **Active Directory support and role management**

  This was the first thing we actually checked for. We were pleased to find that TeamCity supports both AD user and group synchronization. This made our lives easier because we could assign specific roles for specific groups.

  The overall user and role management is some of the best we've seen in CI.

- **Ease of configuration and the overall look and feel**

  TeamCity has pretty simple configurations. Most of them, apart from LDAP, were done in a good-looking admin panel on the main CI server, eliminating the need to dig through a pile of XMLs or run through unclear lists of data.

- **REST API**

  TeamCity comes with a REST API that is well documented and has a lot of action calls that can be used in your own software: triggering/stopping particular builds, getting build test information and event triggering a whole CI server and backup.

Half a Year with TeamCity

Since we started using TeamCity, creating and managing builds became significantly easier. We already have templates and meta-runners for creating new projects, and we are no longer divided by using separate software to achieve the same results. Additionally, we've eliminated the problem of having some CI machines idle while others are maxed out.

Overall, we think that TeamCity is a powerhouse that has strong features for JAVA and .NET projects. It also allows simple creation of additional runners and templates for any language and platform you might choose.