TeamCity

The Ball OS project case study

KALLE LAUNIALA
kalle.launiala@protonit.net

ProtonIT Oy is a micro-size company focused on development of software and information management solutions and tools. The CEO Kalle Launiala has a long history of creating software developer productivity tools ever since the beginning of .NET: starting from database O/R mappers, through leading a full team of developers with T4 + XML based self-automation, and finally ending with a full-stack cloud platform called "The Ball" that brings together the whole history.

The Ball Project

The Ball platform is a full stack .NET-based platform running on Microsoft Azure PaaS services. It originated from the developer self-automation innovation called Abstraction Design Methodology (ADM) in 2010, which led to the general availability launch on Microsoft's Techdays in Finland in March 2013. Due to the nature of developer self-automation, the platform and its core code have been freely available as open source (under MIT-alike free to use & relicense) since 2010.
The platform's main focus was to lower the cost and technical expertise barrier for businesses entering digitalization, with the Web Developer being the highest technical level required to deploy solutions on the platform. The other angle for using the platform were senior developers & software architects, who could take the platform and build it onwards either on their own systems or via co-operation as a community.

Challenges

The platform consists of four logical parts:

- Infrastructure (Azure Cloud): WebRole and WorkerRole "accelerators" that allow flexible deployment of actual projects
- ASP.NET WebApp for WebUI & Device Interface connections; deploys on top of the accelerator
- WorkerRole project; deploys on top of the accelerator
- HTML/Javascript WebUI pages/templates

The whole deployment was Visual Studio deploy & console tooling assisted early on. However, the guidance on the deployment procedures became too difficult to follow. We wanted the developers using the platform to focus on developing, not stumble on the initial deployment. So, we started looking.

Options considered (guidance vs. full solution)

Having experience in automating builds on TFS in closed environments, we screened various options when choosing a deployment and build platform.
While there are easy-to-use online build systems, one of the key aspects of being able to deploy a platform means that developers need to be able to deal with the whole build system. So we needed to deliver an "out-of-the-box" solution with the possibility for the recipients to maintain the system themselves effectively.

Why TeamCity?

This is where TeamCity came in. The initial tests for building & running the tools on TeamCity resulted in an immediate success.

Furthermore, the very mature and detailed build statistics (especially during the build) as well as the free pricing model for open source projects made TeamCity an easy choice to make.

Experience and results of using TeamCity

TeamCity delivers simplicity and flexibility, especially when it comes to build templates that are very powerful and easy to use.

The out-of-the-box support for Visual Studio solutions, msbuild scripts and even .cmd/.bat file-based tooling provided by TeamCity turned out to be surprisingly good. The build progress recognizes different steps and accurately estimates the time to complete the build.

While we were configuring our builds in the late 2014, TeamCity 9.0 was released, which introduced repository-storable build configurations. That is really a killer feature for developer software stack providers, who have to document & guide relatively complex build scenarios for end developers. For us TeamCity 9.0 supporting Git for storing project settings made it possible to pre-create a full “four project + templates” skeleton to be distributed.

Our setup was very straightforward on Azure-hosted Windows VM. We are using now D1 type of instance, running the build agent on a non-persistent SSD. TeamCity recovers well between stop/start of the VM when the build agent drive needs to be refreshed from repository.

As a result, the deployment documentation was reduced to simply configuring deployment parameters (host names and security keys for various services) and straight on to being able to build from The Ball public repository.