TeamCity in Clueda

JetBRAINS

TeamCity

Clueda

Clueda is a cognitive computing solutions provider. We extract knowledge and market moving events from infinite amounts of unstructured data in real-time. The Best in Big Data 2013 award-winning technology is based on over 100 man-years of research in systems biology and cognitive computing. The successful implementation of our software helps traders work more efficiently and increase trading profits on a daily basis. Founded in 2012 as spin-off from the elite research institution Helmholtz Zentrum Munich, Clueda currently has 25 employees.

The Pain

Within two years, Clueda has experienced significant growth, both in the size of our engineering team and complexity of our products. While the previously used development infrastructure worked well to build a product that our customers praised, we were looking for tools options that could sustain our growth and maintain our products at a high quality.

In the original small-team workflow, developers built projects on their own machine and manually deployed artifacts to an Artifactory repository. This is simple and straightforward. However, with the number of repositories reaching over 250 at a team size of about 20 developers, keeping track of the status and dependencies of the myriad of experimental and productive projects in development at any point in time became a prime issue.

Best practices suggest a continuous integration workflow: Instead of manually deploying builds to an artifact storage or even testing or productive system, we wanted to automate as many of such error-prone tasks as possible. Ideally, a testing system will always be updated with the most current state of the code in source control. But let's be humble: an obvious starting point is centralizing compilation, testing and packaging of software artifacts.

Solution Space

There are many tools designed to help in the task of automated building and deploying, the most prominent ones being Jenkins, TeamCity and Bamboo. But before looking at the contenders, we need to identify the key points to be addressed.

Most of our development is JVM-centric, with Scala as the primary programming language. We use git for source control, sbt and Maven as build tools. A good solution would make it easy to run both and report results in a uniform and easily comprehended fashion.

Configuring build processes is not a one-off task. Requirements evolve, new steps are added, configuration changes. With dozens of projects at any one time, it is absolutely necessary that a solution be able to centrally manage build configurations and apply them to multiple projects at once, while still allowing individual projects to adapt and override these settings.

A preliminary survey showed that only the three "big guns" would fit our overall requirement for flexibility and compatibility with all our projects.

Looking at them in detail, we found:

- **Bamboo** was a pleasurable experience in regards to usability, but it lacked the key features we needed in the first place.
- **Jenkins** certainly has the largest ecosystem and rudimentarily supports sbt as a build tool. There is a Jobs DSL plugin available that may support scripting more complex tasks and some level of abstraction on the build process.
- **TeamCity** is the only tool we found that has first-class support for build templates, which were straightforward to setup.

Decisions

TeamCity quickly turned out to be the option that promised to save the most time. During evaluation, we had direct interaction with TeamCity developers, who were very responsive to our questions. This made the final decision very easy.

Relief

After 2 months of using TeamCity productively, we have automated many parts of building and deployment of all active projects. Code quality checks, such as Scalastyle and a self-built configuration verification, were easy to introduce into all configurations and have greatly simplified maintaining and improving our coding standards.

Whenever a developer pushes changes to their git repository, TeamCity pulls these changes and triggers the build process. After a successful build, which includes compiling, tests and code quality checks, development artifacts are deployed directly to Artifactory. Release builds are triggered manually and run through the sbt-release plugin, which ensures correct versioning and disables use of snapshot artifacts in a release, while TeamCity keeps track of the changes since the last release version. Nightly test runs ensure that our artifacts are always in a compatible state.

Outlook

Because it is simple to improve the process, we are not content with the status quo. There's plenty of ideas on how to improve processes. Here is some of our ideas.

Documentation

TeamCity already enables a direct view on the documentation generated for each build. We will consolidate generated docs and automatically deploy them to a searchable website.

Dependency Management

Dependency Hell is cause for a large share of the swear words uttered in software development offices. Centrally detecting such errors should lead to a more tranquil environment.

Deployment

Continuous integration is seen by some as an old hat. Continuous deployment involves bringing the code from source control to production fully automated. TeamCity does not offer specific features for this, but it allows arbitrary deploy processes to be integrated into the environment, thus simplifying tracking the success of such operations.

Visualization

Visualization can improve team engagement. There are a variety of plugins available for this task. Unfortunately, none of them really fit what we had in mind. Luckily, TeamCity offers an extensive REST API to access the build and server state, enabling us to make our own.