Case study of Sven Reimers

Machines

We deploy Hudson on 4 servers — each one of them are Windows 2003 server on Xeon box and 2GB of memory. Hudson is deployed on Tomcat, which sits behind Apache for HTTP compression and user authentication based on LDAP. The source code is hosted on Subversion.

Currently we are not using master/slave feature, but we are investigating that.

Initial set up

Despite the large number of jobs, setting things up worked like a breeze. A colleague of mine created a script to create the job configuration xml files and some files that contained the parts we could put into the view configuration.xml. All this could easily be done, because we could pull the information from our existing well-organized 'build system' that stores such information.

What we run

Each server handles different kind of jobs (so we know which server to look at, given a job.) One Machine does MDA based codegeneration (>400 jobs, Java), one is for manually written Java Code (>200 jobs), one is designated for building native artifacts (MS VisualStudio .NET based), last machine builds Innosetupbased installers. Ant is the default, using a centralized in-house 'build system' containing a lot of infrastructure hacks (somehow comparable to Maven - as far as processing is concerned, e.g. fetching of binaries, javadoc, crossreferenced sources, FindBugs,...)

Many builds do a basic testing as a part of the build, but there are a number of separate "test jobs" that perform large lengthy tests (including database) — we do love the JUnit integration display from Hudson.

There are dependencies between the jobs. When a job runs, it fetches its dependencies from a central binary repository (you see why I said it resembles Maven) and when it's done, it "puts back" the artifacts to this repository. Because the number of jobs are so large, when an upstream job is built, we can't afford to build all its downstream jobs that rely on this new artifact. So we are investigating a better way to chain builds, so that a change will cascade but within a reasonable resource consumption.

Integrating it into our workflow

Eventually, all such dependencies converge into the final artifact, which is the installers. And when it's time for a release, we just pick them up and ship them.