Case study of Rhett Sutphin

Deployment

Hudson lives in its own Tomcat 6 instance, fronted with apache 1.3 via mod_jk. It runs on a single Xserve under OS X Server 10.4. The Xserve has two 1GHz G4s and 2GB of RAM — a bit slow but it works fine. All builds run directly on this machine — no slaves.

Jobs

I use Hudson in support of the software development activities of the Robert H. Lurie Comprehensive Cancer Center at Northwestern University. The cancer center develops a mixture of custom internal and open source applications and libraries, using java and ruby. Hudson is used a bit differently for each.

Java

Our actively-developed java applications are built with maven 2. Hudson's native m2 support, though still in alpha at this writing, has matured rapidly and makes setting up these projects as easy as pie. Hudson is configured to poll the subversion repository every 3 minutes and build (generally "clean deploy") whenever there's a change. When a build is successful, Hudson (via maven's deploy phase) publishes a snapshot to our local m2 repository.

Hudson is also used to build and deploy snapshots of Bering, a tool for database refactoring which we've open-sourced.

Another problem solved with Hudson is a perennial maven-project one: how to refer to dependencies that aren't published in a public repo. We needed to push them into our own repository, but didn't want to give SSH access to the web server to every developer on the each project. The solution is a section in the projects' subversion repositories where developers can check in outside libraries. The metadata for the libraries is stored in a simple TSV file. Whenever there's a commit in this area, Hudson runs a ruby script which examines the libraries and the metadata and uses `mvn deploy:deploy-file` to redeploy any changed snapshots or new releases.

Ruby

The cancer center is using Ruby on Rails for all new internal application development. Hudson's flexible shell script support makes it easy to run an application's full test suite using a separate rails environment. Just like the java builds, hudson is configured to poll the subversion repository every 3 minutes and build whenever there's a change.

The CI::Reporter gem generates junit-like XML reports for the test suite. Hudson aggregates these, giving a history of tests and results, just like for java projects. (The only thing I haven't figured out is how to get the "unstable" vs. "failed" distinction to work for rake-built projects.)

Notification

Hudson setup was mostly very easy. One issue was that there is presently no way to globally map version control usernames to e-mail addresses except by using a default address suffix. Our open source projects are collaborations among developers from several institutions, so this wouldn't work for me. (Actually, even Northwestern's internal global username doesn't map cleanly onto a single e-mail suffix.)

You can set e-mail addresses for each individual user on his or her "people" page, but that page doesn't exist until the developer has committed something. Fortunately, Hudson's configuration is cleanly filesystem-and-text-based, so it was easy to write a script to generate user directories for all the users.