Jenkins use in the Release Manager

Originally, all of the builds ran using the LSF batch queuing system to run the individual steps of the build. However, as the SLAC computing center has upgraded the operating systems on their primary batch queue machines and upgraded the LSF software itself, more and more of the operating systems we use are not supported by the LSF system. Because of this, we have begun the process of moving all of the builds away from LSF and onto the Jenkins system. As mentioned on the Current Build Infrastructure page, Jenkins is used to run the builds for all operating systems except for RHEL5-64bit and RHEL6-64 bit as those two operating systems are still supported by LSF.

Even though the actual build steps are no longer preformed by calls to the LSF system, on the other operating systems, they still use the LSF system to trigger the start of the build on the Jenkins system. The use of LSF to trigger builds is baked deep into the current design of the Release Manager and would require major changes to remove this dependency. However, this is not an issue as the functionality needed to trigger the Jenkins build is simple and can be run on any of the batch nodes.

Jenkins Setup

In order to use Jenkins to control the build, the following has to be set up.

Jenkins Queue on Target Build Machines

The first step it to get the Jenkins client installed on all of the target build machines. After that a queue in the Jenkins system has to be set up that can be called and will trigger the run of a Jenkins script on the target machines. To date, this part of the process has been completely handled by Tony Johnson. The current Jenkins queues that are set up, the OSes they correspond to, and the build machine targets are:

- Fermi RM MacOSX Darwin - OS X Mountain Lion (ppa-pc90719)
- Fermi RM MacOSX - OS X Snow Leopard (bldmac01, bldmac02)
- Fermi RM RHEL5-32 - RedHat Enterprise 5 - 32bit (bldlnx06, bldlnx11, bldlnx12)
- Fermi RM Windows - Windows (glast-win04)

```
The build is invoked by executing the following curl command:

curl 'http://srs.slac.stanford.edu/hudson/job/<queue name>/buildWithParameters?token=GammaRay&buildId=<buildId>'
```

where <queue name> is the name of one of the Jenkins queues (with spaces escaped to %20) and <buildId> is the **buildId** from the ***build*** database table for the package you want to run a build on. Running this command will trigger the Jenkins build process on the relevant queue. This build process runs the following five programs in order:

- *checkoutBuild*
- *compileBuild*
- *createReleaseBuild*
- *testBuild*
- *finishBuild*

Note: If the system is completely switched to Jenkins, one of the build will need to insert the createDoxygen program into the build steps, probably after the compileBuild step. Right now this step is performed on the RHEL5-64 bit build process running under LSF.

Database Configuration Changes

The next step is to update the configuration for the build in the database. In practice this is quite simple as it only involves changing one database configuration parameter in the ***settings*** table. The configuration entry that needs to be changed in the **workflowStart** entry. The **workflowStart** entry tells the RM which of the scripts in the ***workflowScripts*** database table to use to start off the build process. For the LSF based builds this is set to the value 'Checkout' indicating that the RM should launch the *checkoutBuild* program to begin. For builds using the Jenkins system this database entry should be set to a value of "Launch" indicating that it should run the launchJenkins.pl script (see below).

launchJenkins.pl Script

The launch Jenkins script is in the grits-perl/infraBin/ReleaseManager directory in CVS and is installed at grits-perl/bin on the glastrm account on fermilnx-v03. It is a simple Perl script that launches the Jenkins build on the appropriate queue for the requested build. Like the other RM programs, it takes as input a single parameter, buildId, and is invoked in the following manner:

```
launchJenkins.pl --buildId <buildId>
```

where <buildId> is the **buildId** entry from the ***build*** table for the package to be built. From the buildId, the script determines which OS the build is for by looking up the **osId** from the ***build*** table. Once the OS has been determined the appropriate curl command is executed to launch the build process on the correct build queue. If the **osId** for the build is not supported, the script simply exits and prints a message to that effect. If successful, the script also sends an email indicating that it has successfully been called and giving the parameters of the build in question. This was implemented mainly for debugging purposes and the e-mail is being sent to Tom Stephens at the moment.

Jenkins RM Workflow

So the full process for the RM to launch a build in Jenkins is as follows:

1. User creates a tag (either a HEAD or Release tag on the entire package or a sub-package tag triggering the RM to create a LATEST tag)
2. Tag is detected by the RM
3. RM starts the launchJenkins.pl script as a LSF batch job on the express queue.
4. launchJenkins.pl determines the OS for the build, makes a call to the appropriate Jenkins queue, then exits

5. The Jenkins queue runs each of the build steps in order on the selected build machine.