Using Jenkins for C++ Projects

(Please do not publish this page, I'll update it regularly until it reaches a certains quality to be publically available)

Using Jenkins in your C++ Project

Jenkins can be used for non-java projects without any knowledge of the Java languages and tools, especially C++.

Jobs can do the following tasks without any problem:
- Checkout
- Build
- Execute automated Tests
- Publish Tests Results
- Concatenate Tests Results
- Analyze the source
- Build documentation

Checkout and Build

The most part of C++ projects uses fairly different approach than java. You could use maven for your C++ project, however you have access to various builder inside jenkins, through its rich set of plugins: bash, python, cmake.

I would recommend to stick with bash (on Linux) or the Batch (on windows), eventually used with the Conditional Executor plugin, since most of the time, each platform has their own specificity (how deependencies are handled, ...). I usually work with a set of in-VCS build script (bash) that will execute all the commands needed to build the sources.

For simple projet, using the cmake executor can allow multiple platform support.

Jenkins can watch over CVS or SVN or GIT without any problem. Please considere dropping CVS since Jenkins need to look at every file individually, and this consume lot of resources. SVN is perfectly integrated inside Jenkins. Simply forget CVS.

Once built, publish the exe/dll using the Publish Artifact.

Job Splitting

I usually split the build job in two jobs:

1. Prepare job : it performs a complete wipe out of the workspace. It checkout all the source and perform a build from scratch. Executed daily.
2. Build job : incremental build. Executed everytime a modification is done in the source (so potentially several times a day)

Of course, the Prepare job triggers the Build job. The trigger plugin can be used to trigger the “Prepare” job in case of a failing “Build”. Both jobs need to share the same workspace or the same slave (node), or you have to deal with the publish/copy workspace cleverly. I prefere to stick these two jobs on the same node and share the same workspace.

Job Matrix

I use Matrix job to trigger the build on different platform, on different configuration (debug, release). So at the end, you have all binaries always available.

Tests Execution

TBD

Test Concatenation and Publication

TBD

Source Analysis

TBD