

Requirements v2 and v3 for the LArSoft CI system

Eric Church, Ruth Pordes, Erica Snider

4-Feb-2015

Revision 2

Abstract

The LArSoft v1 CI system is deployed on a Jenkins server at Fermilab. With each trigger, a workflow runs successfully and presents valuable information as to the state of the most recent build of the LArSoft code. There are, nevertheless, features still missing or which are desired for a fuller more robust system. We express below changes as needed or desired for each of version 2 and 3 of the LArSoft CI system.

This document expands on the v0-v1 requirements found at <https://cdcvs.fnal.gov/redmine/documents/775>. We leave introduction to the nomenclature to that document and further documentation of the current implementation to <https://cdcvs.fnal.gov/redmine/projects/lar-ci/wiki>.

1. State of v1

The LArSoft CI system largely implemented by Marc Mengel is up and running at http://dbweb4.fnal.gov:8080/LarCI/app/view_builds/index. As of today it builds on two architectures: SLF5 and SLF6.

2. v2 features: Required

- Documentation on the wiki of unit and CI tests currently running.
Currently the tests are pretty self-describing, but it would not be out of order to have a few sentences of documentation for each unit test or CI suite that runs.
- Need alerting upon failure
Right now the CI tests fail silently. It is suggested that a failure report should be emailed to the MicroBooNE analysis tools conveners and designated LBNE-35ton developers. This list should be expanded to include the appropriate LAr1ND and Lariat offline people, and other individuals as specified by any of the partner experiments, when their unit and CI tests are running.
See related optional requirements in the next section.
- Some aesthetic GUI features should be added to the reporting website

1. It seems to be true that when many triggers launch at once the platforms don't all report consistently and the columns of the reporting page don't align.
 2. Generally, one could lay out the page more nicely, with better separation among the current builds and older ones. The green dots indicating success of stages of the workflow should be larger. Text reporting failure/success should be highly visible, clear and succinct.
 3. One would like actual data points on the Google time-series strip-charts
 4. One would like to be able to click on particular builds (data points) on any of the time-series strip-charts and be taken to the log for that build.
- Someone must be responsible for watching the top-level reporting page and watching for failures.

In addition to the automated alarming coming from the Jenkins system or our workflow script, it is inevitable that things may look weird in an unanticipated way on the top website and a developer will need to drill down to find out what's wrong, and fix it or email the responsible party to fix the problem.
 - It shall be possible to configure a test such that it will execute regardless of the success or failure of previous tests in the test workflow to which it belongs.
 - The reporting system shall list all tests that were skipped due to the failure of a previous test in a given test workflow, and provide simple and obvious visual cues at all relevant levels that indicate that the test was skipped as opposed to having passed, or failed.
 - There shall be a mechanism that will use the CI system to generate a new set of reference data without creating test failures, which could terminate test workflows.

One anticipated problem is that the new data products to appear in larsoft v04_00_00 around, say, 9-Feb-2015 will break all the tests which compare current build to previous build or which try to read up an old file and process it. Many CI tests will need to be revisited at that time.

3. v2 features: Desired

- Alerting on failure, optional requirement

In addition to alerting a fixed list of people for each failure as specified above, it shall be possible to generate a target the list of code authors in the triggering commit, or a set of people defined by the particular repositories or packages that were changed by the triggering commit. The two options here are severable.
- We need the Jenkins workflow to run on OSX-Mavericks and Yosemite slave nodes. We can probably drop the SLF5 build in the near future. (We think LBNE still relies upon it at some low level now, and so can't drop it yet.)

- The development model on the workflow side has not been exercised by people beyond the core CI team. The core team has access to the lar-ci git repo, and so it is easy to change the workflow.cfg file and push to change which code gets built, e.g., upon receipt of a trigger. This is not do-able for people outside the core team. A new mechanism for changing this list should be investigated, without having to push to the lar-ci repo. Even for those who have access to the git repo, it is clumsy that one must force the tagging of this repo with the tag which is in current use by Jenkins. We have at least once inadvertently placed that tag in the wrong place via forcing and thus picked up the entirely wrong code for the workflow.
- Currently we allow two simultaneous triggers per build, as configured by the Jenkins managers. This allows two builds per platform as triggered by two nearly simultaneous (greater than the current 5m sleep time) triggers. This should probably be expanded to, say, 4 or 5 allowed simultaneous builds, as check-ins to the larsoft repos become more frequent. The web page needs first to be demonstrated to be able to handle this — see above point (1) under GUI changes.

4. v3 features: Desired

- On the reporting side, only Marc has made changes of consequence. One reason is that though the core CI team has access to the lar-ci-reporting git repo, the procedure for making changes there and testing before pushing is not straightforward. For one thing, one must be on a machine with django 1.4, not later. This is because django 1.5 and beyond deprecates and removes the execute_manager, which is currently used heavily in the django interface to the reporting db. In principle, upon running django 1.4 one may then get a local apache instance up and test the reporting db and make changes to the website <http://localhost:8000/>. Marc has provided these instructions, though it doesn't quite yet work for me. There is a connection to rexdb that seems to not succeed. It would be nice if a LARSoft developer could more easily exercise the code in the reporting and website lar-ci-reporting repo.

- CMS-like relMon monitoring

The only quantitative data comparisons we have now in the CI tests are Kolmogorov-Smirnov tests between specific histograms: the hits in 3 separate planes. The tests are run between the latest build's histograms and those from a build from a previous tagged release. N pairs of histograms can be specified; we're using 3 for this test. All histograms are shown on the resulting CI page along with the KS score. When N grows to be large one can imagine that it is not feasible to show all plots, but to aggregate the results in some way. One can also imagine that rather than running K-S tests, chi2 tests can be run, etc.

CMS's relMon <https://twiki.cern.ch/twiki/bin/view/CMSPublic/RelMon> describes a fuller suite of tests that is run on CMS releases. Example output sits at, say, http://cms-service-reldqm.web.cern.ch/cms-service-reldqm/ReleaseMonitoring/CMSSW_6_2_0_SLHC7VSCMSSW_6_2_0_SLHC1/FullSimReport_UPG2017/, which is a nice bar/pie chart summary of consistency of histograms aggregated within subsystems of the detector. Green in each pie chart shows what fraction of many pairwise chi2 p values come in above some low threshold like 1E-05.

So, for our CI system this may amount to an organization of our existing K-S testing when applied to many more histograms than currently used in CI tests.

...more desired (?) v3 changes...

- The reporting DB holds information that we think is satisfactory. Nevertheless, it's been suggested that rather than grep'ing out the needed quantities from the logs of the various stages of the workflow as we do now, that instead the full logfile be held as a blob in the DB.
- Security: do we need to authenticate users ?
- Back-up: we should back up the DB, say, weekly.