Building a LArSoft patch release

Lynn Garren

2019 LArSoft Workshop

June 24, 2019

Introduction

- Experiments often need special patch releases for production.
- Experiments are responsible for building these releases themselves.
- LArSoft provides tools, instructions, and consultation.
- MicroBooNE is already doing this.

Warnings

- Anyone making a larsoft patch release will have the same permissions as the larsoft release manager
  - We expect that the designated release manager for an experiment will be the person making the larsoft patch release, but this is not a requirement.
- It is very important to use the provided tools and follow the approved procedures.
  - Do not be tempted by shortcuts.

Overview of steps required to make a patch release

- make a redmine larsoft release request
- clone the larsoft suite and experiment code
- build and test
  - make changes
  - build and test
  - repeat until at minimum the unit test are successful
- trigger a CI test with your changes
- update versions and tag
- build on Jenkins
- install on cvmfs
- make the appropriate larsoft cross package tag
- make and upload release notes

Making a release

- Instructions are on the larsoft wiki
  - https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/How_to_tag_and_build_a_LArSoft_patch_release
- We use the larsoft redmine request to keep everyone informed
- Work from a designated patch branch provided by the LArSoft release manager
- Experiment code will also need a patch branch.
  - When appropriate, the experiment and larsoft patch branches should have the same name.
- You will need to make sure everything builds properly for both c2 and e17.
  - Please test with the prof builds.
  - Debug builds can hide problems.

Step zero - create the redmine issue

- MicroBooNE is already doing this
  - https://cdcvs.fnal.gov/redmine/issues/22768
- If the request is for a brand new patch release, the larsoft release manager will need to make branch tags

Step one - create a working directory

- source /cvmfs/larsoft.opensciencegrid.org/products/setup
- setup mrb
- setup larreltools
- export MRB_PROJECT=<your experiment>
- Work on a scratch disk, not in your home directory
  - You will need working space
- startPatchRel `pwd` <experiment> <new patch tag> <existing patch branch>
  - argoneut, dune, icarus, lariat, sbnd, boone
  - For this exercise, the new patch tag is v08_22_00_01
  - For this exercise, the existing branch is v08_22_00_br
  - Note that experiment code without a branch will be on develop

Step two - create experiment branches and build

- source v08_22_00_01/e17p/local*/setup
- cd $MRB_SOURCE
- cd <expt code>
- git checkout v08_22_00 -b v08_22_00_br
- git checkout feature/team_for_v08_22_00_01
  - If there is no branch, you will need to edit ups/product_deps
  - Change larsoft and other versions
  - add py3 qualifier set to the matrix
- git checkout v08_22_00_br
- git merge feature/team_for_v08_22_00_01

Step three - build and test

- cd $MRB_BUILDDIR
- mrbsetenv
- mrb t -jN

- edit source code and rebuild as necessary until the unit tests are successful

- commit and push changes
- run the experiment CI tests with these branches
  - trigger —build-delay 0 —revisions “sbnd*@v08_22_00_br lar*@v08_22_00_br” —workflow sbndcodestandalone_wf
  - You can use feature branches and specify packages separately

Step four - update the package versions

- updatePatchVersion (do not run yet)
- First time:
  - v08_09_01 will become v08_09_01_01
- Second time:
  - v08_09_01_01 will become v08_09_01_02
- Did the package need a new release?
  - start from the bottom of the build tree
  - cd larcoreobj
  - git diff LARSOFT_SUITE_v08_22_00_01
  - If there are no changes, use mrb uv to restore previous version
- For this exercise, only the experiment code will get a new version

Step four continued

- cd $MRB_SOURCE
- mkdir ../notag
- mv lar* ../notag
  - lariat has to do this in pieces…
- mrb uc
- NOW run updatePatchVersion
- cd into each remaining package and make sure the version update is needed
- For new larsoft patch releases:
  - update larsoft/releaseDB/CMakeLists.txt
  - update larsoftobj/bundle/CMakeLists.txt
  - update larsoft product versions

Step five - final build

- Make a final build with the new release versions
- cd $MRB_BUILDDIR
- mrb z
- mrbsetenv
- mrb t -jN
- cd larsoft/releaseDB
- copyToSciSoft lar*
  - check them before you do this
  - will NOT overwrite any existing file on SciSoft

Step 6 - tag

- First make sure all changes except ups/product_deps are committed
  - The tool will commit the changes in ups/product_deps
- tagPatchRel <existing branch> <new master tag>
  - tagPatchRel v08_22_00_br v08_22_00_01
- dogit status
  - should be on the master branch
  - should be up to date with remote origin
- Notice that these tags are all on the patch branch.
  - patch releases are not merged with master or the head of develop

step 7 - build

- LArSoft releases are built on Jenkins using buildFW, not mrb.
  - Sometimes we find problems that do not show up in a mrb build
- To avoid conflicts with normal release builds, use special patch build jobs
  - larpatch-slf
  - larpatch-mac
  - https://buildmaster.fnal.gov/buildmaster/view/LArSoft/
- Since this release is for your experiment, building for macOS is optional.
- If a problem shows up in the build stage, you may need to move your tag. Hopefully the CI test will have caught anything, though.

step 9 - download and upload

- login to scisoftportal
- mkdir tmp
- cd tmp
  - this just makes cleanup easier
- copyFromJenkins -N -q s84-e17 -q s84-c2 larpatch-slf
- copyToSciSoft *
- cleanup:
  - rm *.bz2
  - rm *.txt

step ten - install on cvmfs

- ssh cvmfslarsoft@oasiscfs.fnal.gov
- cat README
- cvmfs_server transaction larsoft.opensciencegrid.org
- ./scripts/installBundleSLF.sh larsoft v08_22_00_01 s84-e17
  - The script will install the e17 and c2 debug and prof builds.
  - This is a convenience wrapper around pullProducts
- If there are both macOS and SLF builds, use installBundle.sh
- IMPORTANT: make sure /cvmfs/larsoft.openscience.grid/products/.working is empty
  - If it is not empty, remove anything in that directory
  - do not remove the .working directory itself
- You can abort if necessary (see the README)

step 10 continued

- PUBLISH
- cvmfs_server publish larsoft.opensciencegrid.org
- DO not leave a transaction open
  - It will lock cvmfs so no one else can work

Step 11 - cross package tag

- Go back to your working directory
- make sure that larreltools is setup
- cp-lar-tag <larsoft release> <larsoftobj release>
  - cp-lar-tag v08_22_00_01 v08_15_02_01
- These changes are made directly in redmine

Step 12 - make the release notes

- makePatchRelNotes <working directory> <larsoft tag> <previous larsoft tag>
  - makePatchRelNotes v08_22_00_01 v08_22_00
  - makePatchRelNotest v08_22_00_02 v08_22_00_01
- cd <larsoft tag>
- cat ReleaseNotes
- cut and paste the first line of the ReleaseNotes file
  - this line is inserted in the larsoft release list
  - https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/LArSoft_release_list
- Click on the red “Release Notes” link on the far right
- cut and paste the rest of the file into the new wiki page
  - be sure to replace the existing default content
  - edit the top portion to add relevant information

Summary

- There is a lot here
- Yes, it's scary
- You have the same permissions as a larsoft release manager
  - BE CAREFUL
- Permissions must be requested
- We provide tools and procedures to help avoid mistakes
  - Always check the instructions
  - They might have changed
  - Helps make sure you didn't forget a step
  - https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/How_to_tag_and_build_a_LArSoft_patch_release
- The SciSoft team is here to help you master this.