Table of Contents

Build and test CORAL and COOL using lcgcmake

Useful info about lcgcmake on the gitlab project page.

```
cd /home/avalassi/lcgcmake
rm -rf lcgcmake

git clone -b master ssh://git@gitlab.cern.ch:7999/sft/lcgcmake.git

wget -O lcgcmake/cmake/toolchain/heptools-85.cmake https://gitlab.cern.ch/sft/lcgcmake/raw/LCG_85
\mv lcgcmake/cmake/toolchain/heptools-85.cmake lcgcmake/cmake/toolchain/heptools-85.cmake.ORIGINA

cat lcgcmake/cmake/toolchain/heptools-85.cmake.ORIGINAL | sed 's|3_1_5|HEAD|' > lcgcmake/cmake/to

env -i USER=$USER HOME=$HOME TERM=$TERM SHELL=/bin/bash CORAL_AUTH_PATH=$CORAL_AUTH_PATH CORAL_DB

cd /home/avalassi/lcgcmake
rm -rf lcgcmake-install
rm -rf lcgcmake-build
mkdir lcgcmake-build
rm -rf lcgcmake-logs
mkdir lcgcmake-logs

source /cvmfs/sft.cern.ch/lcg/contrib/gcc/4.9/x86_64-slc6-gcc49-opt/setup.sh
export CC=`which gcc`
export CXX=`which g++`
export FC=`which gfortran`

export PATH=/cvmfs/sft.cern.ch/lcg/contrib/CMake/3.2.3/Linux-x86_64/bin:$PATH

export CCACHE_DIR=`pwd`/lcgcmake-ccache

cd lcgcmake-build
ln -sf ../lcgcmake-logs logs
date
( time cmake -DLCG_INSTALL_PREFIX=/cvmfs/sft.cern.ch/lcg/releases -DLCG_VERSION=85 -DCMAKE_INSTAL
date
( time make -j8 CORAL ) > logs/mylog-02a-makeCORAL.txt 2>&1
\cp -dpr projects/CORAL-HEAD/src/CORAL-HEAD-stamp/CORAL-HEAD-build.log logs/CORAL-HEAD-build-02a.
date
( time make -j8 COOL ) > logs/mylog-02b-makeCOOL.txt 2>&1
\cp -dpr projects/COOL-HEAD/src/COOL-HEAD-stamp/COOL-HEAD-build.log logs/COOL-HEAD-build-02b.log
date
```

Debug the CORAL and COOL lcgcmake builds on the nightly build nodes

This section gives quick recipes to build and test CORAL and COOL using lcgcmake, or bypassing lcgcmake, on the nightly build nodes.

The procedures described in this section can be useful when debugging the behaviour of lcgcmake in the nightly builds. This is necessary only very rarely.

In the following, it is assumed that a nightly build/test has already been executed and needs to be debugged by selectively rebuilding CORAL/COOL and relaunching their tests. To start a new nightly build from scratch, a new experimental/dev2/dev3/dev4 build should be launched instead from phsft-jenkins.

Build and test CORAL and COOL using lcgcmake on the nightly build nodes

This section describes how to rebuild CORAL/COOL using lcgcmake.

First of all, you should identify the relevant (Linux or Mac) build node and build directory from the cdash.cern.ch dashboard. You should then:

- log in the relevant build node as user `sftnight` using `ssh` and change directory to the relevant top-level lcgcmake directory
- source the setup script in the top-level lcgcmake directory
- if needed, clean the existing build products and remove the existing install directory for CORAL/COOL
- rebuild CORAL/COOL
- launch the CORAL/COOL tests (if needed, you can list all existing tests using `make test ARGS=-N`, which simply executes `ctest -N`)

For instance, to rebuild CORAL and relaunch CORAL tests in dev4 on lcgapp-slc6-x86-64-17:

```
ssh sftnight@lcgapp-slc6-x86-64-17.cern.ch
bash
cd /mnt/build/jenkins/workspace/lcg_ext_dev4/BUILDTYPE/Release/COMPILER/gcc49/LABEL/slc6/
. setup.sh
cd build/
make clean-CORAL
rm -rf ../install/CORAL/3_1-patches/x86_64-slc6-gcc49-opt
make CORAL
make test ARGS="-R coral-test"
```

The results of the CORAL test are in `build/tests/coral-tests.log`.

Note that the procedure above, involving `make clean-CORAL` and `make CORAL`, erases and downloads the CORAL tarfile from scratch, hence:

- if you want to edit files locally, you must use a less invasive CORAL cleanup procedure (just remove selected files in the `build/projects/CORAL-3_1-patches/src/CORAL-3_1-patches-stamp` directory instead)
- if you accept a rebuild from scratch but wish to modify source code, you must trigger an "lcg-prepare-builds" build on phsft-jenkins

Build and test CORAL and COOL bypassing lcgcmake on the nightly build nodes

This section describes how to rebuild CORAL/COOL bypassing lcgcmake.

Incremental rebuild (using make outside lcgcmake)

This should only be used for incremental rebuilds, not for full rebuilds, as this procedure relies on lcgcmake having already executed the cmake step

First of all, you should identify the relevant (Linux or Mac) build node and build directory from the cdash.cern.ch dashboard. You should then:

- log in the relevant build node as user `sftnight` using `ssh` and change directory to the relevant CORAL/COOL source directory
- set a few environment variables and rebuild using "make" (which will normally call ninja internally)

For instance, to rebuild CORAL in dev4 on macitois18 (note that the original copy of the lcgcmake build may be kept in this example):

```
ssh sftnight@macitois18.cern.ch
bash
cd /ec/build/workspace/lcg_ext_dev4/BUILDTYPE/Release/COMPILER/native/LABEL/mac1011/build/project
###mv build.x86_64-mac1011-clang73-opt build.x86_64-mac1011-clang73-opt.original
###cp -pPR build.x86_64-mac1011-clang73-opt.original build.x86_64-mac1011-clang73-opt
export BINARY_TAG=x86_64-mac1011-clang73-opt
export PATH=$(cd `find ../../../../../../install/ninja -name bin`; pwd):$PATH
export VERBOSE=1
make
```

Full rebuild (using make outside lcgcmake)

This should be used for full rebuilds: calling make will internally execute cmake first.

First of all, you should identify the relevant (Linux or Mac) build node and build directory from the cdash.cern.ch dashboard. You should then:

- log in the relevant build node as user `sftnight` using `ssh` and change directory to the relevant top-level lcgcmake directory
- source the setup script in the top-level lcgcmake directory
- change directory to the relevant CORAL/COOL source directory
- set a few environment variables and rebuild using "make" (which will normally call ninja internally)

For instance, to rebuild CORAL in dev4 on macitois18 (note that the original copy of the lcgcmake build may be kept in this example):

```
ssh sftnight@macitois18.cern.ch
bash
cd /ec/build/workspace/lcg_ext_dev4/BUILDTYPE/Release/COMPILER/native/LABEL/mac1011
. setup.sh
cd /ec/build/workspace/lcg_ext_dev4/BUILDTYPE/Release/COMPILER/native/LABEL/mac1011/build/project
###mv build.x86_64-mac1011-clang73-opt build.x86_64-mac1011-clang73-opt.original
export BINARY_TAG=x86_64-mac1011-clang73-opt
export PATH=$(cd `find ../../../../../../install/ninja -name bin`; pwd):$PATH
export VERBOSE=1
export PATH=$(cd `find ../../../../../../install/cmake -name bin`; pwd):$PATH
unset CORALCOOL_CMAKE_LCGSUMMARY
```

```
export LCG_releases_base=/ec/build/workspace/lcg_ext_dev4/BUILDTYPE/Release/COMPILER/native/LABEL
export CC=/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/
export CXX=/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin
export CMAKE_BUILD_TYPE=Release
for pkg in ccache Boost CppUnit expat Frontier_Client mysql oracle Python QMtest sqlite XercesC g
  do pkg1=$LCG_releases_base/$pkg; if [ -d $pkg1 ]; then export CMAKE_PREFIX_PATH=`find $pkg1 -mi
make
```

-- AndreaValassi - 2016-07-19

This topic: Persistency > PersistencyLCGCMake
Topic revision: r5 - 2016-09-01 - AndreaValassi