Table of Contents

LHCb Nightly Build System............................................................................................................................1
  The System................................................................................................................................................1
  Status of nightlies.................................................................................................................................1
  Configuration......................................................................................................................................1
  Regular builds.................................................................................................................................1
  Monitoring.........................................................................................................................................1
  Running from the nightlies.................................................................................................................1
  Nightly tests reference files..............................................................................................................2
    Platform specific references........................................................................................................3
  Nightly tests input data......................................................................................................................3
  Reproducing nightly tests locally.....................................................................................................3
  Documentation...............................................................................................................................4
LHCb Nightly Build System

The System

The LHCb Nightly Build System is a collection of tools and scripts that allow automation of build and test tasks for LHCb software.

A technical description of the system can be found in LHCbNightliesImplementation and hints on how to fix some problems in LHCbNightliesTroubleshooting.

Status of nightlies

The current status of the LHCb nightly builds is available at https://lhcb-nightlies.web.cern.ch/ . See also the status of the LCG nightlies.

RSS feeds reporting about error/warning details for slots, projects or specific platforms can be defined on the summary webpage:

- use "Custom RSS feed" button in the top of the summary webpage

Configuration

The Nightly Build System configuration is defined in terms of slots, projects and platforms.

A slot is a consistent set of inter-related software projects that should be built and tested together. The configuration of a slot includes, in addition to the list of configured projects, the list of platforms the slot should be built for and some extra metadata for the fine tuning of the build.

Software projects are defined in the slot configuration with enough detail to be able to check-out the code from the software repositories. The special version HEAD means master plus all merge requests targeting master that are labelled with the name of the slot (see LBCORE-1156), or with all-slots. Any other version maps directly to Git (as in git checkout version). Special configurations can be defined on an ad hoc basis (for example projects in 2016-patches check out 2016-patches plus all merge requests targeting 2016-patches branch that are labelled with all-slots label).

For the platforms we use the strings that identify the SupportedPlatforms.

Regular builds

Every night we start the build of several slots, whose configurations are stored on Gitlab (LHCbNightlyConf).

Monitoring

%TODO%

KIBana monitoring of the LHCb build machines can be found here

Running from the nightlies
The following instructions are for DaVinci, but they apply to any other project; instructions for Gauss are here. The first thing is to decide which slot. Usually one slot builds DaVinci on the latest LHCb, or the LHCb release candidate, and the other uses the head of Gaudi. Which one to pick is up to what you want to do. See https://lhcb-nightlies.web.cern.ch/ for the definitions. Then decide on the day. Make sure that the version you picked actually compiles. Now you have a slot, say lhcb-head and a day, say last night.

First do

```
lb-dev --nightly lhcb-head [ day ] DaVinci HEAD
```

where day is optional. The default is Today, or pick up a day like Mon, Tue... This builds you a directory ./DaVinciDev_HEAD/. In there check out what you need

```
cd DaVinciDev_HEAD
git lb-use DaVinci
git lb-checkout DaVinci/master Phys/DaVinci
```

and any other needed packages (see Git4LHCb on how to work with LHCb software under Git). Then do

```
./run gaudirun.py <options>
```

This will execute gaudirun.py in the environment of your local project. You can also use

```
./run bash --norc
```

or

```
./run tcsh -f
```

to start a new subshell in the modified environment.

See also GaudiCMakeConfiguration#Building_with_CMake.

You can also run from ganga %TODO% instructions to be updated, previous instructions for SetupProject no longer valid

**Nightly tests reference files**

The testing infrastructure is described in GaudiTestingInfrastructure wiki. The nightlies execute in all slots all the tests that have been defined for packages being build in that slot. The tests that fail because of a mismatch between the output and the reference file produce a special file with extension '.new'.

If you want to copy the nightly reference files to commit them as replacements without needing to re-run the tests yourself, you can use getNightlyRefs, a script that adds the .new files to some local checkout. For example, to update the Brunel references with those from today's lhcb-head nightly, for the currently defined CMTCONFIG platform:

```
lb-dev nightly lhcb-head Brunel HEAD
cd BrunelDev_HEAD
git lb-use Brunel
git lb-checkout Brunel/master Rec/Brunel
git lb-checkout Brunel/master BrunelSys
getNightlyRefs lhcb-head
```

Then replace the existing *.ref files with the uploaded *.ref.new, commit and push to a new branch to make a merge request. To do the replacement of multiple files in one go (recursively, ignoring symlink refs),

Running from the nightlies
you can use

```
find -name '*.new' -exec bash -c 'new={}; ref=${new%.new}; test -f "$ref" -a ! -L "$ref" & & mv -i

This recipe also works if you first download platform-specific refs for more than one platform.

You can simply upload all the new references for a given application/nightly slot/platform combination to the local directory, without need for a local checkout:

```
getNightlyRefs lhcb-sim09 Mon Gauss x86_64-slc6-gcc48-opt
# or when using LbEnv
CMTCONFIG=x86_64-centos7-gcc9-opt BINARY_TAG=$CMTCONFIG lbn-get-new-refs lhcb-master-mr 184 Brunel

Since LHCb!2260 (and following fixes), new references produced by validateWithRef are made as minimal as possible. This is done by only storing the preprocessed stdout plus all summary tables (counters, TTree and histograms). In this way irrelevant lines would not be part of the ref files, thereby reducing diffs and conflicts. Occasionally, this allows for rebasing reference updates on other reference updates with automatic conflict resolution:

```
git rebase -Xtheirs origin/master

Here -Xtheirs means that non conflicting changes are merged and all conflicts are resolved in favour of the branch being rebased. When using this recipe, inspect visually the diff to make sure the conflict resolution is sensible.

Platform specific references

To have different references depending on the platform you need to suffix the filename with the platform, e.g. hlt1_reco_baseline.ref.x86_64+avx2+fma-centos7-gcc8-opt. You can have a single reference match multiple platforms by omitting some of the tokens split by -. For instance, hlt1_reco_baseline.ref.x86_64 would match all but the avx2 platform and hlt1_reco_baseline.ref.x86_64+avx2+fma would match only the avx2 platform. The general logic is that the platform where you run is split into tokens with - as a separator and the same is done with the last part of the ref filename. The filename where the most tokens are a subset of the platform's tokens is the reference used for the validation, see here.

Nightly tests input data

When nightly tests require event data as input, they should used files stored in the CERN-SWTEST storage element, and described in the Test Files database. See the TestFileDB TWiki for details

Reproducing nightly tests locally

To re-run the nightly tests locally for a certain project (e.g. Moore), you should do the following steps (in bash). First, copy the full project from the nightlies:

```
cp -r $LHCBNIGHTLIES/lhcb-head/Today/MOORE/MOORE_HEAD .
cd MOORE_HEAD
```

Then, setup the environment and configure your local copy:

```
. /afs/cern.ch/lhcb/software/nightlies/lhcb-head/Today/setupSearchPath.sh
export USE_CMAKE=1
make configure
```

You can now run a single test with
make test ARGS="-R cosmics -V"

If you want to run the test manually (without make), obtain the command from make test ARGS="-R cosmics -V -N" by removing "--report" "ctest".

For more options for selecting which tests to run, see the CMake FAQ

### Documentation

- A New Nightly Build System for LHCb (*Marco Clemencic and Ben Couturier*)

---

This topic: LHCb > LHCbNightlies
Topic revision: r31 - 2020-04-21 - RosenMatev

---

Reproducing nightly tests locally