# Table of Contents

LHCb Nightly Build System

- The System: 1
- Status of nightlies: 1
- Configuration: 1
- Regular builds: 1
- Monitoring: 1
- Running from the nightlies: 2
- Nightly tests reference files: 2
  - Platform specific references: 3
- Nightly tests input data: 3
- Reproducing nightly tests locally: 3
- Documentation: 3
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 http://lhcb-nightlies.cern.ch. See also the status of the LCG nightlies. The older version of the status page can be reached at https://lhcb-nightlies-old.cern.ch/

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 non-WIP merge requests targeting master. 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 non-WIP merge requests targeting 2016-patches branch). It is also possible to target a WIP merge request to a specific nightly slot by giving the slot as a label - see LBCORE-1156

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). Some of the slots are still using the old XML format described in LHCbNightliesOldConfiguration#Configuration, these are slowly being migrated to a new python description that was first discussed in LBCORE-110. An editor exists for the XML configuration file, it can be accessed from the "Configuration Editor" button of the Nightly Builds status page

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.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
``` 

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.

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:

```bash
getNightlyRefs lhcb-sim09 Mon Gauss x86_64-slc6-gcc48-opt
```

### 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:

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

Then, setup the environment and configure your local copy:

```bash
. /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

```bash
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)
  LHCb-INT-2013-006, Poster-013-328
- Nightly build and test system supports LHC experiments (Stefan Roiser, Ana Gaspar, Yves Perrin, Victor Diez and Karol Kruzelecki) CERN Computer Newsletter April-June 2009
- The nightly build and test system for LCG AA and LHCb software (Karol Kruzelecki, Stefan Roiser and Hubert Degaudenzi) Computing in High Energy and Nuclear Physics (CHEP 2009) Prague.