Table of Contents

Preparing the release

Updating/Tagging the sources

The GIT repository used for the sources is:

ssh://git@gitlab.cern.ch:7999/lhcb-core/LbScripts.git

To work on the clone, fork it in Gitlab and create Merge requests with new features. Once done, just tag the code as for any GIT project and push the tag to the main repo.

To generate the release notes: [... sync you branch with origin master here] cd LbScriptsSys/doc ./getreleasenotes.sh > release.notes git add release.notes git commit -m "Added release notes" git push origin master

Building

- Trigger by hand the release Jenkins job:
  - go to https://jenkins-lhcb-nightlies.web.cern.ch/job/nightly-builds/job/release/build
  - add LbScripts in the field `projects_list`
  - click on the `Build` button (leave all the others fields empty)

- Once the build is completed, check the checkout log
  - go to https://lhcb-nightlies.cern.ch/release/ and check the status of the **LbScripts** build requested
  - if there are problems, ask the project manager to fix them, and restart the build

Releasing the files

The files stored on the build machine need to be copied to the RPM repo.

Follow the recipe to deploy the RPMs to a repository, as mentioned in ProjectRelease

Creating the links RPMs

Then you build the RPMs like so:

To create the meta RPM to put that version as the prod one:

```
lbn-generate-lbscriptsspec ${MyVersion} --prod -o lb.spec -b $TMPDIR/toto && rpmbuild -bb lb.spec
```

Setting the new release as the "dev" version

To create the meta RPM to put that version as the dev one:

```
lbn-generate-lbscriptsspec ${MyVersion} --dev -o lb.spec -b $TMPDIR/toto && rpmbuild -bb lb.spec
```

Then copy the files to the RPM repository:

```
lb-release-rpm --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2018 $TMPDIR/toto/rpmbuild/
lb-release-rpm --copy --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2018 $TMPDIR/toto/rp
```

And install the RPM in the RPM database

```
$LHCBHOME/software/lbinstall/afslbinstall update LBSCRIPTSDEV
```

To install on CVMFS log-in as cvllhcb on cvmfs-lhcb and run:

```
cvmfs_transaction
cvmfslbinstall update LBSCRIPTSDEV
```

 IN CASE OF PROBLEMS, you can force the link by hand

```
cd $LHCBRELEASES/LBSCRIPTS
ln -nsf LBSCRIPTS_vXrYpZ dev
```

Switching the production release to a new version

To create the meta RPM to put that version as the dev one:

```
lbn-generate-lbscriptsspec ${MyVersion} --prod -o lb.spec -b $TMPDIR/toto && rpmbuild -bb lb.spec
```

Then copy the files to the RPM repository:

```
lb-release-rpm --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2018 $TMPDIR/toto/rpmbuild/
lb-release-rpm --copy --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2018 $TMPDIR/toto/rp
```

And install the RPM in the RPM database on AFS:

```
$LHCBHOME/software/lbinstall/afslbinstall update LBSCRIPT
```

To install on CVMFS log-in as cvllhcb on cvmfs-lhcb and run:

```
cvmfs_transaction
cvmfslbinstall update LBSCRIPTS
```

IN CASE OF PROBLEMS when putting a release in production, the link can be updated by doing:

```
cd $LHCBRELEASES/LBSCRIPTS
ln -nsf LBSCRIPTS_vXrYpZ prod
```

==========

For LBSCRIPTS in SVN - Valid until v8r4x - Obsolete afterwards

==========

Release tools

LbScripts can be released with the same tools as for all other projects, but some extra steps are needed for install project.

⚠ a cached version of the environment is generated, so it is necessary to make sure that you do not have LbScripts in your User_release_area when performing the release.

The LbConfiguration package generates the LbLogin script whichs contains the name of the released version.

⚠ Therefore, LbConfiguration must ALWAYS be released with the same version as LbScripts itself !

/ install_projects specific steps

When releasing LbLegacy, install_project.py must be updated at each new release of LbScripts in order to make sure that the auto-update mechanism of install_project and LbScripts works correctly. In this scripts the script_version variable must be set to the day of the release with the following pattern: YYMMDD, e.g.:

```
script_version = '120206'
```

It should increase monotonously, as install_project uses this variable to check wherther a new version is available.

The LbScripts version should be set to the version of LbScripts to be released as this is used to choose the version to e downloaded from the repository e.g

```
lbscripts_version = "v6r6p4"
```

Once all packages have been tagged individually, call:

```
tag_package -P LbScripts version
```

Building the release

```
cd $LHCBRELEASES
mkproject -p LbScripts -v vXrYpZ --not-parallel
```

⚠ Don't forget to edit/fix the environment cache files

Two cache files have to be edited, for both **sh** and **csh**.

This means: * Removing the COMPILER_PATH env variable * Adjusting the PATH and LD_LIBRARY_PATH

A quickcheck of the consistency of the two files can be done using:

```
EnvCompare.py `envcache v7r5 sh` `envcache v7r6 sh`
EnvCompare.py `envcache v7r5 csh` `envcache v7r6 csh`
```

The release also needs to be locked:

```
mkproject -p LbScripts -v vXrYpZ -a K
```

Creating the RPMs

You need get a copy of the NightlyTools repository:

```
mkdir -p $TMPDIR/build
cd  $TMPDIR/build
git clone https://gitlab.cern.ch/lhcb-core/LbNightlyTools.git
cd LbNightlyTools
. setup.sh
```

Then you build the RPM like so:

```
lbn-generate-lbscriptsspec v8r1p1  -o lb.spec -b $TMPDIR/toto && rpmbuild -bb lb.spec
```

To create the meta RPM to put that version as the prod one:

```
lbn-generate-lbscriptsspec v8r1p1 --prod -o lb.spec -b $TMPDIR/toto && rpmbuild -bb lb.spec
```

To create the meta RPM to put that version as the dev one:

```
lbn-generate-lbscriptsspec v8r1p1 --dev -o lb.spec -b $TMPDIR/toto && rpmbuild -bb lb.spec
```

Then copy the files to the RPM repository:

```
lb-release-rpm --copy --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2018 $TMPDIR/toto/rp
```

And install the RPM in the RPM database (just the metadata) e.g.

```
$LHCBHOME/software/rpmrel/afslbpkr rpm -- -ivh --justdb $LHCBTAR/rpm/lhcb/LBSCRIPTS_v8r4-1.0.0-1.
$LHCBHOME/software/rpmrel/afslbpkr rpm -- -Uvh --justdb $LHCBTAR/rpm/lhcb/LBSCRIPTSDEV-8.4.0-1.no
```

Setting the new release as the "dev" version

Once the release of LbScripts vXrYpZ has been deployed to the the release area, it is necessary to change the link $LHCBRELEASES/LBSCRIPTS/dev:

```
cd $LHCBRELEASES/LBSCRIPTS
rm dev && ln -s LBSCRIPTS_vXrYpZ dev
```

or

```
cd $LHCBRELEASES/LBSCRIPTS
ln -nsf LBSCRIPTS_vXrYpZ dev
```

Switching the production release to a new version

A "prod" symbolic link exists on AFS that points to the production version of LbScripts.

 When putting a release in production, the link has to be updated by doing:

```
cd $LHCBRELEASES/LBSCRIPTS
rm prod && ln -s LBSCRIPTS_vXrYpZ prod
```

The links should be updated on CVMFS and on local install by install project itself.

Upon going to production, you need to create the prod meta RPM and install the packages with the -justdb option in lppkr e.g.

```
cd /afs/cern.ch/lhcb/software/rpmrel
./afslbpkr rpm -- -ivh --justdb tmp/LBSCRIPTS-8.3.1-1.rpm
./afslbpkr rpm -- -Uvh --justdb tmp/LBSCRIPTS-8.3.1-1.rpm
```

Test uses cases

This section provides some (incomplete) information about the scenarios that should be tested when releasing a new version of LbScripts.

Main types of installations: * Single repository: Used for normal installations on a mahine without AFS or CVMFS * Chained repositories: Used for grid jobs which use software from a shared area, but also install software in their local disk space

Main use case: * Install some new software * Add versions/new projects to an existing area * Check whether a project is installed * Check whether the SQLDDDB data package is installed (this is a special case)

Prerequisites

Make sure that the account used for testing does not have the LHCb environment already set up ! The variables CMTSITE and CMTPROJECTPATH especially ! For accounts linked to group z5, touch the file ".nogrouplogin" to avoid invoking the group scripts automatically !

A version of LbScripts is needed. For the production version:

```
wget http://lhcbproject.web.cern.ch/lhcbproject/dist/install_project.py
```

For the dev version:

```
wget http://lhcbproject.web.cern.ch/lhcbproject/dist/devel/install_project.py
```

If a version of install_project is already there, it is also possible to use the --dev-install option to switch to the "dev" version of install project.

MYSITEROOT and CMTCONFIG must be defined, e.g.

```
export MYSITEROOT=/opt/install/lib
export CMTCONFIG=x86_64-slc5-gcc43-opt
```

N.B. In order to test LbScripts before the release of a new version, it is necessary to invoke LbLogin with the following options:

```
LbLogin --user-area-scripts --scripts-version=""
```

Single area

* Install a version of Gaudi and check logs and presence of files

```
python install_project.py -b Gaudi v23r4
```

* Use install project to check that the software is installed

```
python install_project.py -b --check Gaudi v23r14
```

* Check group permissions on repositories (XXX check what they should be)

* Check that --check on SQLDDDB is always false

```
python install_project.py --check SQLDDDB
```

* install Brunel

```
python install_project.py -b Brunel v44r0
```

* Setup the Brunel environment and run the SAM test

```
. ./LbLogin.sh
SetupProject Brunel
cd $BRUNELSYSROOT/cmt
cmt TestPackage sam
```

* Try SetupProject with --use

```
python install_project.py AppConfig v3r147
SetupProject --debug --use="AppConfig v3r147"  Brunel v44r0
```

Chained area

In this configuration, the MYSITEROOT contains several repositories (2 normally) separated by a colon like a PATH.

```
export MYSITEROOT=/opt/install/joblocal:/opt/install/shared
```

In this configuration, for the tests to be more realistics, it is necessary to make sure that the test cannot modify files in the shared area (e.g. chaining with cvmfs with give that effect). e.g. using two different accounts on lxplus (with /tmp/lben/siteroot containing the previous installation).

```
setenv MYSITEROOT '/tmp/benlhcb/LocalArea:/tmp/lben/siteroot'
cd /tmp/benlhcb/LocalArea
python ./install_project.py AppConfig v3r146
source ./LbLogin.csh
```

Check that AppConfig v3r146 is in /tmp/benlhcb/LocalArea/lhcb/EXTRAPACKAGES/

```
SetupProject --debug --use="AppConfig v3r146"  Brunel v44r0
```

Run Brunel with a few events

```
gaudirun.py $APPCONFIGOPTS/Brunel/DataType-2012.py $BRUNELROOT/options/COLLISION12-Beam4000GeV-Ve
```

The same tests as for a single area should be re-run.

InstallProjectWithRPM

Troubleshooting and special installations

Install on AFS

This has to be done from an SLC6 node.

- install on AFS
  - First, set some environment variables that define the project name and version to be installed. Used in the following commands (so you can simply cut and paste them directly). Works for bash shells, so recommend switching to this if it isn't your default.

    ```
    MyProject=<ProjectName>
    MyVersion=vXrY
    ```

    To check the following commands will work for you, now run

    ```
    echo ${MyProject^^}
    ```

    which should return the project name, in upper case.
  - prepare the AFS volume

    ```
    lb-project-manage-volume -c $MyProject $MyVersion
    ```
  - install the new project

    ```
    /afs/cern.ch/lhcb/software/lbinstall/afslbinstall install ${MyProject^^}_${MyVersio
    ```

or

```
/afs/cern.ch/lhcb/software/rpmrel/rpmextractor.py -v ${LHCBTAR}/rpm/lhcb/${MyProject^^}_${MyVersi
```

- - Rebuild the project to get the proper hardoded paths in LbLogin.sh and such files.

    ```
    cd /afs/cern.ch/lhcb/software/releases/LBSCRIPTS/LBSCRIPTS_${MyVersion}/LbScriptsSys
    cmt br cmt make
    ```

- Add CVMFS to the CMAKE_PREFIX_PATH

Add the following LbLogin.sh

```
export CMAKE_PREFIX_PATH=${CMAKE_PREFIX_PATH}:/cvmfs/lhcb.cern.ch/lib/lhcb:/cvmfs/lhcb.cern.ch/li
```

And to LbLogin.csh:

```
setenv CMAKE_PREFIX_PATH ${CMAKE_PREFIX_PATH}:/cvmfs/lhcb.cern.ch/lib/lhcb:/cvmfs/lhcb.cern.ch/li
```

- Don't forget to edit/fix the environment cache files

Two cache files have to be edited, for both **sh** and **csh**.

This means: * Removing the COMPILER_PATH env variable * Adjusting the PATH and LD_LIBRARY_PATH

A quickcheck of the consistency of the two files can be done using:

```
EnvCompare.py `envcache v7r5 sh` `envcache v7r6 sh`
```

```
EnvCompare.py `envcache v7r5 csh` `envcache v7r6 csh`
```

- release and lock the AFS volume

```
lb-project-manage-volume -r $MyProject $MyVersion
lb-project-manage-volume -l $MyProject $MyVersion
```

- prepare the web pages links

```
$LHCBDOC/scripts/addrel.py $MyProject $MyVersion
```

-- BenjaminCouturier - 13-Feb-2012

This topic: LHCb > LbScriptsRelease
Topic revision: r32 - 2018-01-26 - BenjaminCouturier