Table of Contents

Software Release Guide **1**
- Scope 1
- Overview of the procedure 1
- For Release Managers - Preparation of the build 1
  - For Projects 1
  - For Data Packages 2
- For Release Shifters - Deployment Prerequisites 2
  - Special information 2
  - Prerequisites for new release shifters 2
- Deployment of Projects 2
  - Update of the RPM repository 2
  - Update of the Software configuration Database 3
  - Software Installation (CVMFS) 4
  - Update Software Project Web Page 5
- Deployment of Data Packages 6
  - Trigger by hand the lhcb-release Jenkins job (if needed) 6
  - Update of the RPM repository 6
  - Software Installation (CVMFS) 7
- Deployment of LbEnv 8
  - Install new versions 8
  - Update stable version 9

Special Instructions and troubleshooting **10**
- Installation on AFS 10
- Deployment from Old Tarfiles 10
- Removing Software (archival) 10
  - Projects 10
  - Data Packages 11
- Troubleshooting (For Releasers) 11
  - If there is a quota problem 11
  - In case of problems with incomplete installs 11
- Troubleshooting (For Librarians) 12
  - If the nightly builds do not work 12
  - If lb-release-rpm prints messages like "error: not an rpm package" 13
  - If everything has gone so wrong that you you want to sit in a corner and cry... or restart from scratch 13
- For the Librarian 14
  - Releasing a new version of LCG externals 14
    - Until Gaudi v23r2 14
    - As from Gaudi v23r3 14
    - As from LCG 86 15
    - As from LCG 87 16
    - Updating the Software Configuration DB for a new LCG or Gaudi 17
  - Releasing a new version of LCG Grid 17
  - Preparing the meta RPM with the software to be installed locally on the Online farm 18
  - Releases of the nightly builds on /cvmfs/lhcbdev.cern.ch 19
  - Preparing the RPM and releasing a new version of CMake 19
  - Installing new generators 19
  - CVMFS Appendix 20
    - IT CVMFS Information 20
    - LHCB Software on cvmfs-lhcb 20
      - Scripts 20
    - CRON 21

Software Release Guide

Scope

The following documentation applies to the release of LHCb Physics projects as well as data packages. Ganga is installed on a separate CVMFS volume, and there are specific instructions for LHCbDIrac which can be found here

⚠ These instructions do not apply to LbScripts. Use LbScriptsRelease instead.

Overview of the procedure

The software is built by the Jenkins continuous integration engine. The configuration of the projects, their dependencies and which stacks should be built is defined in the software configuration database, with the typical procedure including the following steps:

1. Release manager tags a project (or several) and triggers the build by importing the projects to the software configuration database
2. Jenkins builds the project(s) and prepares the RPM packages
3. Release manager checks the outcome of the build and either restarts the build or requests a release in JIRA
4. Release shifter copies the RPM packages to the official LHCb package repository (YUM format). At this point, anybody can perform local installations using `lbinstall`
5. Release shifter marks the RPM as "released" in the software configuration database, and specifies which platforms have effectively been released. This is necessary for the build system to know what is released and isn't when building stacks, but also for the automated release procedures on CVMFS to know which platforms should be installed
6. Release shifter performs the installation on CVMFS and updates the documentation

Instructions for all the steps are detailed below.

For Release Managers - Preparation of the build

For Projects

- prepare the project as usual (release notes, tags, etc.), following the instructions at PrepareProjectReleaseGit
- register the project/version to the software database

```
lb-sdb-import <Project> <version>
```

* Check that the import into the software configuration was correct by running:

```
lb-sdb-query listReleaseStacks
```

This should show the projects to release and the platforms for which they should be released. You can select which platforms to release for by using lb-sdb-addplatform --release (they should of course be a subset of the platforms offered by the projects this one depends on).

Within 10-15 min the build should start to appear in the nightlies page https://lhcb-nightlies.cern.ch/release. Note that, for a dependency tree of projects, `lb-sdb-import` can be called just once for the top level project as it will pull in all the dependencies (unless you need to provide the `--sourceurl` option, in which

case each project has to be added individually, in the dependency order).

- if there is a problem in the build, it can be re-started via the dedicated button after you must log in (the build will not restart by itself after a retag)
- once satisfied by the build, take note of the build id (you can use the direct link icon) and make the request via LHCb Deployment Request form

For Data Packages

- Make the request via LHCb Deployment Request form. The tool `lb-sdb-import` is only for projects, so it cannot be used for data packages.

For Release Shifters - Deployment Prerequisites

See the LHCb shift database to find who is this week's shifter.

Special information

Starting from 13 October 2016, you should set your environment to DEV as : ". /cvmfs/lhcb.cern.ch/lib/LbLoginDev.sh" *Starting from 28 March 2019, you don't need to set your environment to DEV.

Prerequisites for new release shifters

If you are a new release shifter, please make sure you have the prerequisites that are needed at some point of the release procedure.

- You should be added as a "developer" in the LHCb Deployment Jira project (https://its.cern.ch/jira/browse/LHCBDEP/), to receive and track release requests
- You must be added to the lhcb-cvmfs-librarians E-group (for access to the IT managed CVMFS installation machine and for communication with other release shifters)
- You must have a valid grid certificate (For release of DecFiles package)
- You need to be registered as a software deployment shifter in the shifts database
- You need to be registered to use Jenkins, to be able to trigger release builds from the web interface (for data packages)
- Check that the CMTPATH environment variable is not defined in the shell which you are using

As shifter, you can self-assign shifts through the shift database shift database. Usually this is done by block-booking weeks for individual shifters.

Deployment of Projects

Update of the RPM repository

The goal of this step is to make sure that:

1. The RPMs have been copied to the YUM repository (currently in EOS)
2. The YUM metadata has been updated (the lb-release-rpm internally uses the "createrepo" yum command).

Actions:

- Check on the release build summaries page if the build is OK: https://lhcb-nightlies.cern.ch/release
- If there are problems, ask the project manager to fix them, and restart the build
- Update the RPMs repository (**to be done from any host**)
  - connect to lxplus.cern.ch

    ```
    ssh lxplus7.cern.ch
    bash
    kinit ${USER}@CERN.CH
    ```
  - publish the produced RPMs (all of them, unless explicitly requested otherwise, see partial install section)
    - Define the build ID to extract RPMS from

      ```
      build_id=1234
      ```
    - Try to copy the RPMs to the repository

      ```
      lb-release-rpm --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2019 /e
      ```
    - If everything is fine, do the real copy and update the repository info

      ```
      lb-release-rpm --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2019 --
      ```
  - Check it's all there

    ```
    ls -ltr /eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2019/ | tail -10
    ```

    In case of problems see here

BE CAREFUL: Since 20141010, the release procedure automatically bumps up the release number for packages for which there is already a version in the RPM repo. If your slot contains more projects that the ones requested for release, you need to **ONLY copy the needed ones** to the RPM repo. In that case the **--rpm-regex** option of lb-release-rpm should be used.

N.B. If you want to release all rpms EXCEPT some matching a specific string (e.g. gcc62) you can use:

```
lb-release-rpm -c --rpm-regex '^(?!.*(gcc62)).*'  --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-
```

Update of the Software configuration Database

The goal of this step is to make sure that:

1. The project is referenced properly in the Software configuration DB
2. The list of platforms released for the project have been updated

Actions:

- Verify that the software configuration DB has been updated:

  ```
  lb-sdb-query d <Project> <version>
  ```

  This should show projects in the slot or already released. If some are missing re-run:

  ```
  lb-sdb-import <Project> <version>
  ```

- Update the software database.
  First try in dry-run mode:

```
lb-deployment-updatesdb /eos/project/l/lhcbwebsites/www/lhcb-nightlies-artifacts/release/l
```

If more platforms/projects are being updated than necessary you can limit using the options of the lb-deployment-updatesdb command.If all ok, just run:

```
lb-deployment-updatesdb --update /eos/project/l/lhcbwebsites/www/lhcb-nightlies-artifacts/
```

In case of SSO, or certificate problems:

```
ssh -fN  -L12345:localhost:7474 lbariadne
export SDBURL=http://localhost:12345/db/data/
lb-deployment-updatesdb [...]
```

You can ignore the warnings such as: WARNING: Error encountered while acquiring CERN SSO cookie for Ariadne. Attempting to connect with no SSO.

- Check whether the platforms for **all projects** in the stack being deployed were updated correctly, so for each project do:

```
lb-sdb-query listPlatforms <Project> <version>
```

which should show the complete list of platforms for which your project was released. If it's not the case add the missing ones with

```
lb-sdb-add-platforms <Project> <version> <platform1> [<platform2>...]
```

- Check that the projects are correctly flagged as released: call

```
lb-sdb-query listReleases
```

and check that they are not in the list, otherwise call

```
lb-sdb-release -r <Project> <version>
```

to manually flag them as released.

Software Installation (CVMFS)

Actions:

- Connect to "cvlhcb" account on host "cvmfs-lhcb"

```
ssh yourusername@cvmfs-lhcb
```

- Check that nobody else is connected to install software

```
who -q ; pgrep -fl -u cvlhcb
```

if someone else is connected and if there are processes running as cvlhcb, perhaps try sending them an email to see what they are doing before proceeding

- `sudo` into the all powerful account

```
sudo -i -u cvlhcb
```

- Start a transaction on the server to be able to write to the disk

```
cvmfs_transaction
```

- if the command returns that a transaction was already created, somebody else is installing at the same time. Check with the **who** command before going any further.
- Note:
  1. **cvmfs_transaction** is a wrapper around the CVMFS command that creates the transaction on the repository lhcb.cern.ch, and logs the command and result. It is best to use it instead of using the direct command **cvmfs_server transaction lhcb.cern.ch**
  2. even if you decide to use **cvmfs_server** instead of the wrapper, it is best to specify the repository in the command as this server could potentially host several repositories in the future.

- Install the project (all registered platforms)

```
install_software.sh <Project> <version>
```

  - Note:
    1. in case of problems or missing dependencies you can alter the options to lbinstall by setting the OPTS environment variable e.g

```
OPTS=--no-strict install_software.sh <Project> <version>
```

    2. If the binary RPMs do not get deployed (packages with x86_64[...]), check wether the SoftwareConfigurationDB was updated correctly as per the following instructions. The DB is the source of information for the list of platforms to deploy.

- Check that it all went fine with

```
ls -ltr /cvmfs/lhcb.cern.ch/lib/lhcb/<PROJECT>/<PROJECT>_<version>/InstallArea/
```

  the platforms you have installed should all be there.
- Update the CVMFS catalogs

```
cvmfs_publish
```

  - Note:
    1. **cvmfs_publish** is a wrapper around the CVMFS command that publishes the transaction on the repository lhcb.cern.ch, and logs the command and result. It is best to use it instead of using the direct command **cvmfs_server publish lhcb.cern.ch**
    2. Even if you decide to use **cvmfs_server** instead of the wrapper, it is best to specify the repository in the command as this server could potentially host several repositories
- Remember to log out

Update Software Project Web Page

Note: this is not about Doxygen documentation (automatically generated, with some delay, after the installation)

Actions:

- Wait for the project to be visible in CVMFS:

```
ls -ltr /cvmfs/lhcb.cern.ch/lib/lhcb/<PROJECT>/<PROJECT>_<version>/InstallArea/
```

- Prepare the web pages links

/eos/project/l/lhcbwebsites/www/projects/scripts/addrel.py

Deployment of Data Packages

WARNING The procedure for the release of data packages via RPMs is still under heavy development and the automation is not yet in place.

Data packages are different from standard projects:

- They do not need to be compiled **for several platforms** (make is run nonetheless to perform basic actions)
- They are part of the DBASE or PARAM projects
- They are not registered in the software configuration Database and the builds may not always be triggered by the release manager.

The release steps are therefore different from normal projects.

Trigger by hand the lhcb-release Jenkins job (if needed)

- go to https://jenkins-lhcb-nightlies.web.cern.ch/job/nightly-builds/job/release/build
- add the list of packages (with their versions) in the field `packages_list` (e.g. `RawEventFormat v1r1 Gen/DecFiles vXrY`), the name of the container can be specified with something like `"container":"name"` (e.g. `PARAM:TMVAWeights v1r0`). The container can be omitted for `DBASE`.
- click on the `Build` button (leave all the others fields empty)
- watch it being built at https://jenkins-lhcb-nightlies.web.cern.ch/job/nightly-builds/job/checkout/

- Once the build is completed, check the checkout log
    - go to https://lhcb-nightlies.cern.ch/release/
    - click on the project name (DBASE or PARAM) in the build summary, and look for the package build log (after the line "building data packages in...")

Update of the RPM repository

The goal of this step is to make sure that:

1. The RPMs have been copied to the YUM repository (currently in EOS)
2. The YUM metadata has been updated (the lb-release-rpm internally uses the "createrepo" yum command).

Actions:

- If you didn't trigger the build, check that everything is OK (see trigger data package build section)
- If there are problems, ask the project manager to fix them, and restart the build
- Update the RPMs repository (*to be done from any host *)
    - connect to lxplus.cern.ch

```
ssh lxplus7.cern.ch
bash
kinit ${USER}@CERN.CH
```

    - publish the produced RPMs (all of them, unless explicitly requested otherwise, see partial install section)
        - Define the build ID to extract RPMS from

```
build_id=1234
```

◊ Try to copy the RPMs to the repository

```
lb-release-rpm --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2019 /e
```

· For old stacks only released as tar files:

```
lb-release-oldtar /eos/project/l/lhcbwebsites/www/lhcb-nightlies-artif
```

◊ If everything is fine, do the real copy and update the repository info

```
lb-release-rpm --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2019 --
```

· For old stacks only released as tar files:

```
lb-release-oldtar --copy /eos/project/l/lhcbwebsites/www/lhcb-nightlie
```

♦ Check it's all there

```
ls -ltr /eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2019/ | tail -10
```

In case of problems see here

BE CAREFUL: Since 20141010, the release procedure automatically bumps up the release number for packages for which there is already a version in the RPM repo. If your slot contains more projects that the ones requested for release, you need to **ONLY copy the needed ones** to the RPM repo. In that case the **--rpm-regex** option of lb-release-rpm should be used.

N.B. If you want to release all rpms EXCEPT some matching a specific string (e.g. gcc62) you can use:

```
lb-release-rpm -c --rpm-regex '^(?!.*(gcc62)).*'  --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-
```

Software Installation (CVMFS)

Actions:

- Connect to "cvlhcb" account on host "cvmfs-lhcb"

```
ssh yourusername@cvmfs-lhcb
```

- Check that nobody else is connected to install software

```
who -q ; pgrep -fl -u cvlhcb
```

if someone else is connected and if there are processes running as cvlhcb, perhaps try sending them an email to see what they are doing before proceeding

- `sudo` into the all powerful account

```
sudo -i -u cvlhcb
```

- Start a transaction on the server to be able to write to the disk

```
cvmfs_transaction
```

♦ if the command returns that a transaction was already created, somebody else is installing at the same time. Check with the **who** command before going any further.

♦ Note:

1. **cvmfs_transaction** is a wrapper around the CVMFS command that creates the transaction on the repository lhcb.cern.ch, and logs the command and result. It is best

to use it instead of using the direct command **cvmfs_server transaction lhcb.cern.ch**
2. even if you decide to use **cvmfs_server** instead of the wrapper, it is best to specify the repository in the command as this server could potentially host several repositories in the future.

- Install the data package

```
cvmfslbinstall install <PROJECT>_<Hat>_<Package>_<version>
```
- Check that it all went fine with

```
ls -ltr /cvmfs/lhcb.cern.ch/lib/lhcb/<PROJECT>/<Hat>/<Package>/<version>/
```
- Update the CVMFS catalogs

```
cvmfs_publish
```

♦ Note:
1. **cvmfs_publish** is a wrapper around the CVMFS command that publishes the transaction on the repository lhcb.cern.ch, and logs the command and result. It is best to use it instead of using the direct command **cvmfs_server publish lhcb.cern.ch**
2. Even if you decide to use **cvmfs_server** instead of the wrapper, it is best to specify the repository in the command as this server could potentially host several repositories

- Remember to log out

Deployment of LbEnv

Install new versions

LbEnv is the set of LHCb environment scripts and tools used for development. It will be released to CVMFS by a CRON running on cvmfs-lhcb.cern.ch, but in the meantime some tools have been prepared for this task (namely **update_lbenv.sh**).

First check in DRYRUN mode whether the tool is going to update correctly:

```
CVMFS-Shared ~ > DRYRUN=1 update_lbenv.sh
2019-03-07T16:39:30+0100 Will install:
lbenv-kit.stable.x86_64-centos7.347-4a87636109.tar.bz2
lbenv-kit.stable.x86_64-slc6.347-4a87636109.tar.bz2
lbenv-kit.unstable.x86_64-centos7.347-68420333c0.tar.bz2
lbenv-kit.unstable.x86_64-slc6.347-68420333c0.tar.bz2
2019-03-07T16:39:30+0100 Unpacking lbenv-kit.stable.x86_64-centos7.347-4a87636109.tar.bz2 ...
2019-03-07T16:39:30+0100 DRYRUN mode. Would add link /cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/testin
2019-03-07T16:39:30+0100 Unpacking lbenv-kit.stable.x86_64-slc6.347-4a87636109.tar.bz2 ...
2019-03-07T16:39:30+0100 DRYRUN mode. Would add link /cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/testin
2019-03-07T16:39:30+0100 Unpacking lbenv-kit.unstable.x86_64-centos7.347-68420333c0.tar.bz2 ...
2019-03-07T16:39:30+0100 DRYRUN mode. Would add link /cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/unstab
2019-03-07T16:39:30+0100 Unpacking lbenv-kit.unstable.x86_64-slc6.347-68420333c0.tar.bz2 ...
2019-03-07T16:39:30+0100 DRYRUN mode. Would add link /cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/unstab
```

If this seems satisfactory:

```
CVMFS-Shared ~ > update_lbenv.sh
Starting TRANSACTION
Created TRANSACTION
2019-03-07T16:39:37+0100 Will install:
lbenv-kit.stable.x86_64-centos7.347-4a87636109.tar.bz2
lbenv-kit.stable.x86_64-slc6.347-4a87636109.tar.bz2
lbenv-kit.unstable.x86_64-centos7.347-68420333c0.tar.bz2
```

```
lbenv-kit.unstable.x86_64-slc6.347-68420333c0.tar.bz2
2019-03-07T16:39:37+0100 Unpacking lbenv-kit.stable.x86_64-centos7.347-4a87636109.tar.bz2 ...
2019-03-07T16:39:37+0100 Downloading https://lhcb-rpm.web.cern.ch/lhcb-rpm/lbenv-kits/lhcb.cern.c
tar: Removing leading `/' from member names
2019-03-07T16:39:43+0100 Adding link /cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/testing/x86_64-centos7
`/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/testing/x86_64-centos7' -> `../347/stable/x86_64-centos7'
2019-03-07T16:39:43+0100 Unpacking lbenv-kit.stable.x86_64-slc6.347-4a87636109.tar.bz2 ...
2019-03-07T16:39:43+0100 Downloading https://lhcb-rpm.web.cern.ch/lhcb-rpm/lbenv-kits/lhcb.cern.c
tar: Removing leading `/' from member names
2019-03-07T16:39:47+0100 Adding link /cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/testing/x86_64-slc6 ->
`/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/testing/x86_64-slc6' -> `../347/stable/x86_64-slc6'
2019-03-07T16:39:47+0100 Unpacking lbenv-kit.unstable.x86_64-centos7.347-68420333c0.tar.bz2 ...
2019-03-07T16:39:47+0100 Downloading https://lhcb-rpm.web.cern.ch/lhcb-rpm/lbenv-kits/lhcb.cern.c
tar: Removing leading `/' from member names
2019-03-07T16:39:51+0100 Adding link /cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/unstable/x86_64-centos
`/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/unstable/x86_64-centos7' -> `../347/unstable/x86_64-centos
2019-03-07T16:39:51+0100 Unpacking lbenv-kit.unstable.x86_64-slc6.347-68420333c0.tar.bz2 ...
2019-03-07T16:39:51+0100 Downloading https://lhcb-rpm.web.cern.ch/lhcb-rpm/lbenv-kits/lhcb.cern.c
tar: Removing leading `/' from member names
2019-03-07T16:39:57+0100 Adding link /cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/unstable/x86_64-slc6 -
`/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/unstable/x86_64-slc6' -> `../347/unstable/x86_64-slc6'
`.installed_kits.txt.2019-03-07T16:39:37+0100' -> `/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/.install
removed `.installed_kits.txt.2019-03-07T16:39:37+0100'
`/cvmfs/lhcb.cern.ch/lib/var/lib/LbEnv/.installed_kits.txt' -> `.installed_kits.txt.2019-03-07T16
Starting PUBLISH ====================================
gio  7 mar 2019, 16.39.57, CET
Using auto tag 'generic-2019-03-07T15:39:57Z'
Processing changes...
...........
```

Update *stable* version

When a new *testing* version of LbEnv is ready for production, the release shifter is asked to update the prod version, in which case he/she has to run from cvmfs-lhcb.cern.ch:

```
update_lbenv_stable.sh
```

which starts a transaction update the prod version and publish the transaction if successful.

WARNING: in case of failure the transaction is kept open for the shifter to fix the problem or abort it.

Special Instructions and troubleshooting

Installation on AFS

Since 2018-01-22, installation on AFS is not needed, but the old instructions are still available at ProjectReleaseOld.

Note that only librarians will be able to change the content of AFS, so contact <lhcb-software-librarians@cern.ch>.

Deployment from Old Tarfiles

Very old versions of software projects cannot be deployed with RPMs, in which case it's possible to refer to the instructions at ProjectReleaseOld.

Removing Software (archival)

Projects

- Archive the project documentation (step to be defined, for the time being try `python $LHCBDOC/scripts/gc_archive.py <project> <version>`, noting that `<project>` must be in all lowercase). This must be done **before** removing the software!

- On cvmfs-lhcb, start a transaction in the usual way, then list the RPMs you want to remove using the **cvmfslbinstall** command:

```
cvmfslbinstall list URANIA_v6r2p1
URANIA_v6r2p1                            1.0.0   1    local
URANIA_v6r2p1_index                      1.0.0   1    local
URANIA_v6r2p1_x86_64_centos7_gcc62_dbg   1.0.0   1    local
URANIA_v6r2p1_x86_64_centos7_gcc62_do0   1.0.0   1    local
URANIA_v6r2p1_x86_64_centos7_gcc62_opt   1.0.0   1    local
URANIA_v6r2p1_x86_64_slc6_gcc49_dbg      1.0.0   1    local
URANIA_v6r2p1_x86_64_slc6_gcc49_do0      1.0.0   1    local
URANIA_v6r2p1_x86_64_slc6_gcc49_opt      1.0.0   1    local
URANIA_v6r2p1_x86_64_slc6_gcc62_dbg      1.0.0   1    local
URANIA_v6r2p1_x86_64_slc6_gcc62_do0      1.0.0   1    local
URANIA_v6r2p1_x86_64_slc6_gcc62_opt      1.0.0   1    local
```

Then you can remove the packages using **cvmfslbinstall remove**, passing it the list of files to be removed (it will order them correctly and check dependencies). Please note that lbinstall does **NOT remove directories if they contain files that do not belong to the RPM packages** (e.g. generated files). Therefore it is necessary to check the project top directory if still existing and check what is left and remove manually if necessary.

N.B. after removing the packages, the files created by hand that do not belong to any package are NOT removed. You may need to check the directory for those.

N.B. Be careful about the packages returned, **cvmfslbinstall list URANIA_v6r2** returns the packages for v6r2 and v6r2p1...

Remember to run `cvmfs_publish` at the end, as usual, to close the transaction.

Data Packages

- got to the CVMFS publishing machine (cvmfs-lhcb) and uninstall

```
cvmfs_transaction
cvmfslbinstall remove <PROJECT>_<Hat>_<Package>_<version><Package> <version>
cvmfs_publish
```

Troubleshooting (For Releasers)

If there is a quota problem

When copying tars. First check what's wrong

```
fs lq $LHCBTAR/*
```

Then increase quota with

```
afs_admin sq $LHCBTAR/<PROJECT> <quota>
```

In a project:

```
fs lq /afs/cern.ch/lhcb/software/releases/<PROJECT>/*
```

Then increase quota with

```
afs_admin sq /afs/cern.ch/lhcb/software/releases/<PROJECT>/<PROJECT>_<version>/*
```

In rpms... on 2/12/2016 Ben did (see JIRA):

```
for r in $LHCBTAR/rpm/lhcb/<PROJECT>_<version>_*rpm; do
          for p in `rpm -qp --requires $r | grep LCG`;
               do ./afslbinstall --disable-yum-check --just-db install  $p;
           done;
    done
```

In case of problems with incomplete installs

Sometimes the automatic installation may fail, for example because not all platforms were correctly built in the release build, or because some of the needed external dependencies are missing. Depending on the reason for the incomplete install, you may want to ignore the missing dependencies, and install anyway, or just install those platforms for which the dependencies are OK.

- First find the available packages:

```
/afs/cern.ch/lhcb/software/lbinstall/afslbinstall  query "${MyProject^^}_${MyVersion}"
```

- There are two useful options of `afslbinstall` to help with partial deployment of the packages returned by the above query:

```
/afs/cern.ch/lhcb/software/lbinstall/afslbinstall install --nodeps <package_name>
```

installs just the package but none of its dependencies. Be careful with this because you have to deploy by hand, one package at a time, the whole dependency stack for the deployment to be useful

```
/afs/cern.ch/lhcb/software/lbinstall/afslbinstall install --no-strict <package_name>
```

installs the complete set of dependencies, but skipping any missing dependencies

For completeness, `afslbinstall` called without options installs the complete set of dependencies but stops with an error if a dependency is missing. It is what is called internally by `lb-deployment-afs-install`

```
/afs/cern.ch/lhcb/software/lbinstall/afslbinstall install <package_name>
```

- Hint: all the above can be condensed in a single command to install all platforms ignoring missing dependencies:

```
for r in $(/afs/cern.ch/lhcb/software/lbinstall/afslbinstall query "${MyProject^^}_${MyVersion}")
/afs/cern.ch/lhcb/software/lbinstall/afslbinstall install --no-strict $r
done
```

- Or, if you want to exclude a platform (e.g. because it has missing dependencies):

```
for r in $(/afs/cern.ch/lhcb/software/lbinstall/afslbinstall query "${MyProject^^}_${MyVersion}"
/afs/cern.ch/lhcb/software/lbinstall/afslbinstall install $r
done
```

- Don't forget to do the same on cvmfs, where you replace `/afs/cern.ch/lhcb/software/lbinstall/afslbinstall` with `cvmfslbinstall`

Troubleshooting (For Librarians)

If the nightly builds do not work

The nightly build system is nothing more than wrappers around simple tools that can be called by any user, so it is possible to prepare the RPMs for the release by hand.

- Go to an Centos7 machine with AFS (e.g. lxplus or a build machine), to a temporary working directory (e.g. /build/$USER)
- Prepare the configuration file to drive the build

  ```
  lbn-gen-release-config MyProject vXrY > lhcb-release.json
  ```

  or

  ```
  lbn-gen-release-config --cmt  --pack 'Hat/Package vXrY' > lhcb-release.json
  ```

  for data packages

- Checkout the sources and prepare the shared RPM

  ```
  lbn-checkout --verbose lhcb-release.json
  lbn-rpm --shared --builddir tmp/checkout --verbose lhcb-release.json
  ```

- For each required platform build, test and prepare the RPM (not needed for data packages)

  ```
  export CMTCONFIG=...
  lbn-build --verbose --clean --with-tests -j 4 lhcb-release.json
  lbn-rpm --verbose lhcb-release.json
  ```

- Check if the builds/tests are ok (not needed for data packages)
  - start the Python mini web server

    ```
    cd artifacts && python -m SimpleHTTPServer
    ```
  - with a web browser connect to the port 8000 of the build machine used (e.g. http://lxbuild00.cern.ch:8000/) and navigate to the
    - build log: summaries.<CMTCONFIG>/<Project>/build_log.html
    - test reports: summaries.<CMTCONFIG>/<Project>/html

- Publish the RPMs (from lxplus.cern.ch)

  ```
  cp -v -n artifacts/MYPROJECT*.rpm $LHCBTAR/rpm/lhcb
  ```

  then, from lxplus.cern.ch

  ```
  createrepo --workers=20 --update $LHCBTAR/rpm/lhcb
  ```

- If lxplus.cern.ch works, continue with the procedure to install to AFS, otherwise you should install on AFS from the generated tar files, then continue with the normal procedure

If lb-release-rpm prints messages like "error: not an rpm package"

It might mean that the RPM file is probably corrupted.

This could happen during the copy, in which case it is enough to remove the corrupted files from $LHCBTAR/rpm/lhcb and call again `lb-release-rpm`.

If it still does not work, check that there are no permanent problems in $LHCBTAR/rpm/lhcb (e.q. quota) and that the files are copied correctly with something like (bash)

```
for f in /eos/project/l/lhcbwebsites/www/lhcb-nightlies-artifacts/release/lhcb-release/${build_id
  cmp $f $LHCBTAR/rpm/lhcb/$(basename $f)
done
```

If everything looks correct, try to rebuild the project via the rebuild button in the dashboard.

If everything has gone so wrong that you you want to sit in a corner and cry... or restart from scratch

THIS SHOULD NOT BE DONE, but I document it in any case.

To undo a release, so that it can be replayed from the beginning you should (not that it is a somehow reverse order to what done in the normal procedure):

- uninstall from CVMFS... you shouldn't have gone that far... TODO
- revert the software database (should not be needed) TODO
- remove the old-style tarballs and html files

  ```
  rm -i $LHCBTAR/html/${MyProject^^}_${MyProject^^}_${MyVersion}*
  rm -i $LHCBTAR/${MyProject^^}/${MyProject^^}_${MyProject^^}_${MyVersion}*
  ```
- remove web page links TODO
- uninstall the RPMs
  - check that you know what you want to remove

    ```
    /afs/cern.ch/lhcb/software/lbinstall/lbinstall list ${MyProject^^}_${MyVersion}
    ```

- actually remove them

```
/afs/cern.ch/lhcb/software/lbinstall/lbinstall remove $(/afs/cern.ch/lhcb/software/r
```

- remove the AFS volume in the release area

```
afs_admin delete /afs/cern.ch/lhcb/software/releases/${MyProject^^}/${MyProject^^}_${MyVer
```

- remove the RPMs from the repository

```
rm -i $LHCBTAR/rpm/lhcb/${MyProject^^}_${MyVersion}*.rpm
createrepo --update --workers=20 $LHCBTAR/rpm/lhcb
```

For the Librarian

Releasing a new version of LCG externals

Until Gaudi v23r2

Until Gaudi v23r2, the dependencies to be included in LCGCMT were included in the Gaudi requirements files. The procedure to build the tarball was therefore the following:

For Gaudi, one more setup needs to be done: the generation of the tarball of its dependencies. This has to be done for each and every optimized >binary< (x86-64-slc6-gcc46-opt, x86-64-slc5-gcc46-opt, x86-64-slc5-gcc43-opt, i686-slc5-gcc43-opt)

```
cd $Gaudi_release_area
mkLCGCMTtar -n GAUDI_<version> -b <binary>
```

This script produces a log file in the local directory from where it has been run. Check it to see if everything was OK.

As from Gaudi v23r3...

The generation of the tarball of its dependencies. This has to be done for each and every optimized and debug >binary< (x86-64-slc6-gcc48-opt, x86-64-slc6-gcc48-dbg, x86-64-slc6-gcc49-opt, x86-64-slc6-gcc49-dbg).

The project LHCbExternals contains the list of dependencies to be included in LCGCMT. It is therefore necessary to:

1. Make sure that the dependency list is up to date
2. Tag a new version of LHCbExternals, with a version matching LCGCMT e.g. LCGCMT 63 -> LHCbExternals v63r0 LCGCMT 63a -> LHCbExternals v63r1 etc...

Release LHCbExternals to AFS:

```
cd /afs/cern.ch/lhcb/software/releases/
mkproject -p LHCbExternals -v vXrY -a ngc
mkproject -p LHCbExternals -v vXrY -a K
```

Now prepare the tar ball of LCGCMT, which has to be done differently depending on the version of LCGCMT:

* Up to LCGCMT 66:

```
cd /afs/cern.ch/lhcb/software/releases/
mkLCGCMTtar -n LHCBEXTERNALS_<version> -b <binary>
```

If everything has gone so wrong that you you want to sit in a corner andcry... or restart from scratch

This script produces a log file in the local directory from where it has been run. Check it to see if everything was OK.

* As from LCGCMT 68:

```
mkLCGCMTtarFromRPM LHCbExternals v69r0p1 $CMTCONFIG
```

As from LCG 86

We do not extract dependencies any more, the LHCBEXTERNALS project is abandoned.

To Generate the install project tar files LCGCMT for x86_64-slc6-gcc49-opt, just run (LbScript v8r7 at least is needed)

```
makeTarFromRPM LCGCMT v86r0 x86_64-slc6-gcc49-opt ./LHCBEXTERNALS_86.json
```

Where LHCBEXTERNALS_86.json is a file of the form:

```
{
    "heptools": {
    "version": 86,
    "packages":[
        "AIDA",
        "Boost",
        "eigen",
   "CLHEP",
   "COOL",
   "CORAL",
   "CppUnit",
   "Frontier_Client",
   "GSL",
   "HepMC",
   "HepPDT",
   "Python",
   "QMtest",
   "Qt",
   "RELAX",
   "ROOT",
   "XercesC",
   "fastjet",
   "fftw",
   "graphviz",
   "libunwind",
   "neurobayes_expert",
   "oracle",
   "pyanalysis",
   "pygraphics",
   "pytools",
   "sqlite",
   "tcmalloc",
   "vdt",
   "xqilla",
   "xrootd",
   "tbb",
   "rangev3",
   "cppgsl",
   "ipython"]

    }
}
```

As from LCG 87

We now just need to pre-install the build dependencies so that the nightly builds can continue working. The RPM needed at install time are pulled in as needed.

The metadata is contained in files such as:

LHCBEXTERNALS_88.json:

```
{
    "heptools": {
    "version": 88,
    "packages":[
        "AIDA",
        "Boost",
        "eigen",
   "CLHEP",
   "COOL",
   "CORAL",
   "CppUnit",
   "Frontier_Client",
   "GSL",
   "HepMC",
   "HepPDT",
   "Python",
   "QMtest",
   "Qt",
   "RELAX",
   "ROOT",
   "XercesC",
   "fastjet",
   "fftw",
   "graphviz",
   "libunwind",
   "neurobayes_expert",
   "oracle",
   "pyanalysis",
   "pygraphics",
   "pytools",
   "sqlite",
   "tcmalloc",
   "vdt",
   "xqilla",
   "xrootd",
   "tbb",
   "rangev3",
   "cppgsl",
   "ipython"]

    }
}
```

The metadata files are kept and versioned in https://gitlab.cern.ch/lhcb-core/rpm-recipes (LHCBEXTERNALS sub directory).

Follow the instructions in the README.md to get the list of rpms to install, and to install them on CVMFS and AFS.

Then update the software configuration DB, e.g.

```
lb-sdb-addplatform LCG 91 x86_64-slc6-gcc62-opt
lb-sdb-addplatform LCG 91 x86_64-slc6-gcc62-dbg
lb-sdb-addplatform LCG 91 x86_64-centos7-gcc62-dbg
```

```
lb-sdb-addplatform LCG 91 x86_64-centos7-gcc62-opt
lb-sdb-addplatform LCG 91 x86_64-centos7-gcc7-opt
lb-sdb-addplatform LCG 91 x86_64-centos7-gcc7-dbg
```

Updating the Software Configuration DB for a new LCG or Gaudi

First import Gaudi without asking for a release:

```
lb-sdb-import --norelease Gaudi v28r3
```

If we are dealing with a new LCG, add the corresponding platforms:

```
lb-sdb-addplatform LCG 89 x86_64-slc6-gcc62-opt
lb-sdb-addplatform LCG 89 x86_64-slc6-gcc62-dbg
lb-sdb-addplatform LCG 89 x86_64-centos7-gcc62-dbg
lb-sdb-addplatform LCG 89 x86_64-centos7-gcc62-opt
lb-sdb-addplatform LCG 89 x86_64-centos7-gcc7-opt
lb-sdb-addplatform LCG 89 x86_64-centos7-gcc7-dbg
```

If Gaudi features more platforms than LCG, then specify the list of platforms to release:

```
lb-sdb-addplatform --release Gaudi v28r3 x86_64-slc6-gcc62-opt
[...]
```

Then schedule Gaudi for release:

```
lb-sdb-release Gaudi v28r3
```

Then check if everything is ok:

```
$ lb-sdb-query show Gaudi v28r3
Node 382601 Properties
------------------------------
project   : GAUDI
version   : v28r3

Node 382601 relationships
------------------------------
2683451:REQUIRES(O)     -> (ID:382602, project:LCG, version:89)
2683462:REQUESTED_PLATFORM(O) -> (ID:382306, platform:x86_64-centos7-gcc62-dbg)
2683463:REQUESTED_PLATFORM(O) -> (ID:382305, platform:x86_64-centos7-gcc62-opt)
2683464:REQUESTED_PLATFORM(O) -> (ID:382607, platform:x86_64-centos7-gcc7-dbg)
2683466:REQUESTED_PLATFORM(O) -> (ID:382608, platform:x86_64-centos7-gcc7-opt)
2683468:REQUESTED_PLATFORM(O) -> (ID:382303, platform:x86_64-slc6-gcc62-dbg)
2683469:REQUESTED_PLATFORM(O) -> (ID:382302, platform:x86_64-slc6-gcc62-opt)
2683470:REQUESTED_PLATFORM(O) -> (ID:382307, platform:x86_64-centos7-gcc62-do0)
2683471:REQUESTED_PLATFORM(O) -> (ID:382609, platform:x86_64-centos7-gcc7-do0)
2683473:REQUESTED_PLATFORM(O) -> (ID:382304, platform:x86_64-slc6-gcc62-do0)
2683449:PROJECT(I)      <- (ID:122379, project:GAUDI, sourceuri:gitlab-cern:gaudi/Gaudi)
2683474:RELEASEREQ(I)   <- (ID:122366, project:NONE, type:RELEASE, version:NONE)
```

Releasing a new version of LCG Grid

After LHCbDirac v8r0 (before then mkLCGCMTTar was necessary)

```
mkLCGCMTtarFromRPM  LHCbDirac vXrY $CMTCONFIG --nonatives
```

If some RPMs are missing from the middleware, it is possible to create them and to add them to the LHCb externals repository in $LHCBTAR/rpm/lcg The code necessary is in the LCGRPM repo. It can be used to create the spec.

```
git clone https://gitlab.cern.ch/lhcb-core/LCGRPM.git
cd LCGRPM/package
./createExternalRPMSpec.py  -s Grid -o ext.spec voms 2.0.12-3 x86_64-slc6-gcc48-opt
rpmbuild -bb ext.spec
```

After that step, the file should be copied to the RPM repo and the RPM repo should be reindexed:

```
cp /tmp/rpmbuild/RPMS/noarch/voms_2.0.12_3_x86_64_slc6_gcc48_opt-1.0.0-1.noarch.rpm $LHCBTAR/rpm/
 createrepo --workers=20 --update $LHCBTAR/rpm/lcg
```

N.B. /tmp/rpmbuild is a default of createExternalRPMSpec, and the -b option can be used to put the files to another directory e.g.

```
./createExternalRPMSpec.py -b /tmp/lben/toto -s Grid -o ext.spec voms 2.0.12-3 x86_64-slc6-gcc48-
```

Creates the RPM:
/tmp/lben/toto/rpmbuild/RPMS/noarch/voms_2.0.12_3_x86_64_slc6_gcc48_opt-1.0.0-1.noarch.rpm

Preparing the meta RPM with the software to be installed locally on the Online farm

First get a local version of LbNightlyTools:

```
mkdir -p /build/$USER
cd /build/$USER
git clone http://git.cern.ch/pub/LbNightlyTools
cd LbNightlyTools
. setup.sh
```

Now prepare the Meta RPM spec. There are two ways:

- By specifying explicitly the packages to be included:

```
lbn-generate-metaspec -o ofm.spec OnlineFarmMeta 1.1.1 MOOREONLINE_v23r6_x86_64_slc6_gcc48_
```

(You need to pass all necessary applications RPMson the command lines, dependencies are dealt with automatically) And build it:

```
rpmbuild -bb ofm.spec
```

- Using the ONLINE tag in the software configuration DB:

First add the project/version/combination to the soft configuration DB and check that the update worked:

```
lb-sdb-tag -p x86_64-slc6-gcc48-opt MooreOnline v23r7p15 ONLINE
lb-sdb-query listTag ONLINE
```

Removal can be done with

```
lb-sdb-tag -r -p x86_64-slc6-gcc48-opt MooreOnline v23r7pq5 ONLINE
lb-sdb-query listTag ONLINE
```

Then produce the RPM :

```
. lb-sdb-env.sh
lbn-generate-metaspec -o ofm.spec OnlineFarmMeta 1.1.1 -t ONLINE
```

(You need to pass all necessary applications RPMson the command lines, dependencies are dealt with automatically) Build the RPM:

```
rpmbuild -bb ofm.spec
```

Now copying it to the RPM repository:

```
lb-release-rpm --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2019 /tmp/rpmbuild/RPMS/noa
lb-release-rpm --rpm-dir=/eos/project/l/lhcbwebsites/www/lhcb-rpm/lhcb2019 --copy /tmp/rpmbuild/R
```

Now install on plus:

```
cd /group/hltsw
./onlinelbpkr install OnlineFarmMeta
```

Releases of the nightly builds on /cvmfs/lhcbdev.cern.ch

The releases should be completely automated, but here are the details needed to pcheck the system, perform manual installations/add slots etc...

To connect:

```
ssh cvmfs-lhcbdev.cern.ch
sudo -i -u cvlhcbdev
```

Then you can start a transaction and copy files to /cvmfs/lhcbdev.cern.ch

Preparing the RPM and releasing a new version of CMake

LHCb uses the Linux_x86_64 binaries from Kitware:

https://cmake.org/download/

The tools to repqckage the RPM can be found in:

https://gitlab.cern.ch/lhcb-core/rpm-recipes

In the cmake folder.

The RPM prepared just needs to be copied to the yum repository and the metadata rebuilt.

Installing new generators

The simulation team tests generators on /cvmfs/lhcbdev.cern.ch. Once validated they need to be installed on the production cvmfs.

The list will be expressed as a YAML file such as:

```
packages:
- name: 'lhapdf'
  LCG: [ '88']
  versions: [ '6.1.6.cxxstd' ]
- name: 'pythia8'
  LCG: [ '88' ]
  versions: [ '230' ]
- name: 'rivet'
```

```
  LCG: [ '88' ]
  versions: [ '2.5.2' ]
- name: 'thepeg'
  LCG: [ '88' ]
  versions: [ '2.0.3' ]
```

To perform the installation, it is first necessary to derive the list of RPM packages to install. This can be done in the following manner:

```
virtualenv yamlToPackages
. ./yamlToPackages/bin/activate
pip install --extra-index-url https://lbmultipython03.cern.ch/simple/ --trusted-host lbmultipytho

lbGetPackagesFromYAML <YAML FILE>
```

This will print out a list of packages. It is then necessary to login to cvmfs-lhcb and install them using the cvmfslbinstall command, e.g.

```
cvmfs_transaction
cvmfslbinstall install <packages>
cvmfs_publish
```

CVMFS Appendix

IT CVMFS Information

All information from IT Department can be found at https://twiki.cern.ch/twiki/bin/view/CvmFS/Installers

LHCB Software on cvmfs-lhcb

Scripts

All updates to software deployed on cvmfs should be done under account "cvlhcb". "${HOME}/bin" contains the necessary scripts to add/remove software:

- **cvmfs_transaction**: To start a transaction to deploy software on the server
- **install_software.sh**: To call install_project to add new projects/versions
- **remove_software.sh**: To Remove software versions
- **cvmfs_publish**: Synchronize the CVMFS file system

Some other scripts are also available (same name as above but prefixed with std_ perform the same action without logging)

The output of those command is automatically logged to "${HOME}/logs/install.log

Sometimes a package is not known to the `install_software.sh` script. In that case

```
cvmfslbinstall install DBASE_HAT_PKG_vXrY
```

may help.

CRON

The local CRON is running on this host to automatically update SQLDDDB every hour. The logs for this cron is in: ${HOME}/logs/update_sqldddb.log

A backup of the CRONTAB is available in ${HOME}/conf.crontab.backup

-- MarcoClemencic - 30 Jun 2014

-- MarcoCattaneo - 3 Jun 2017

-- MarcoClemencic - 2018-01-22

This topic: LHCb > ProjectRelease
Topic revision: r197 - 2019-04-17 - BenjaminCouturier