Table of Contents

Preparing a software project for release

This wiki explains the procedure to be used by software release managers to prepare a project for release. **It assumes the project is in gitlab**. Software release procedures are discussed in the lhcb-release-managers E-group, all software release managers should be members of this group

For reference the old procedure for projects hosted in SVN is documented here

Prerequisites

Before making a release, you should ensure that:

- You have merged all merge requests that you would like to include in the release.
  - You need to have "Master" privileges in gitlab for the project you want to release
  - Go to the gitlab home page, click on "Merge Requests" and check that there are no open Merge Requests for the branch that you want to release (or, if there are any, that they are requests that you explicitly want to exclude)
- Your project has been successfully built and tested in the nightlies
  - Note that, depending on the slot configuration, the nightly may or may not pick up pending merge requests.

Preparing a release of the master or someOther branch

These are the most common use cases: you wish to tag and release the complete contents of the current head of the **master** branch or of **someOther** branch of a gitlab project. In the following example we prepare the release of version **vXrY** of the project **MyProject**

Open a window on lxplus or other machine with access to gitlab.cern.ch, go to a clean directory and execute the following commands:

```
git clone ssh://git@gitlab.cern.ch:7999/lhcb/MyProject.git
cd MyProject
```

You are now at the head of your local copy the **master** branch.

- If you want to release **someOther** branch, move to it **NOW** with the command

```
git checkout someOther
```

You are now at the head of the branch (**master** or **someOther**) that you want to release. Create the branch (**vXrY-release**) in which you will modify the files needed for release and move to it:

```
git checkout -b vXrY-release
```

Update the project version and dependencies

- In the file `CMakeLists.txt`, update the project version number to `vXrY` and update the project dependencies (i.e. change the version of used projects if they have changed)
- For projects that depend on Gaudi versions prior to v28r0, do the corresponding thing also for the CMT files:
  - In the file `MyProjectSys/cmt/requirements`, update the project version number to `vXrY`
  - In the file `cmt/project.cmt`, update the project dependencies (i.e. change the version of used projects if they have changed)

Update the release notes

- It is suggested to use the merge request descriptions of all the merge requests accepted since the last (e.g. vXr(Y-1)) release. Use the following command (see also the section below if you are releasing an application or many projects):

```
lb-gen-release-notes vXr(Y-1) vXrY
```

This creates the file `ReleaseNotes/vXrY.md` containing the merge request identifier, title, author and description of all MRs merged between the releases vXr(Y-1) and vXrY. The generated file is formatted in Gitlab markdown, according to the file `ReleaseNotes/release_notes_template.md`. If this file does not exist, the command uses a default one, but it is recommended that you provide one for your project (just copy an existing one from another project and adapt it to your project. Note that you can use labels on Gitlab merge requests to sort the MRs into categories, by adding code snippets like

```
### New features
{{ section(['new feature']) }}
### Bug fixes
{{ section(['bug fix']) }}
```

(in the above example it is assumed you have defined Gitlab labels `new feature` and `bug fix` in your Gitlab project).

You can then edit the generated file as desired. Once done, you are ready to commit the changed files to gitlab

- First do a `git status` to double check that you are on the `vXrY-release` branch, and that you see as modified the files that you have touched in the previous steps
- execute the following commands

```
git add CMakeLists.txt ReleaseNotes/vXrY.md
## in case of projects based on Gaudi < v28r0, also:  git add cmt/project.cmt MyProjectSys/cmt/re
git commit -m "Documentation and dependencies for vXrY release"
git push origin vXrY-release
```

Now go to gitlab

- Make a merge request of the **vXrY-release** branch to the branch that you wish to release (either **master** or **someOther**)
- Merge the request
- Go to the Repository/tags menu and create a new tag, called `vXrY` on the branch that you wish to release (either **master** or **someOther**)
- In the release notes of the tag, you can copy/paste the content of the file `ReleaseNotes/vXrY.md` (hence the interest to use gitlab markdown!)
- And you are done!

Preparing a stack release

Recent `LbEnv` versions add the `--stack` option to `lb-gen-release-notes`, which allows for passing `JSON` description of the stack being released. An example file is as follows

```
{
    "LCG": ["96b with ROOT 6.18.04", "96b with ROOT 6.18.04", []],
    "Gaudi": ["v33r0", "v33r0", ["LCG"]],
    "LHCb": ["v50r6", "v51r0", ["Gaudi"]],
    "Lbcom": ["v30r6", "v31r0", ["LHCb"]],
    "Rec": ["v30r6", "v31r0", ["Lbcom"]],
    "Brunel": ["v60r6", "v61r0", ["Rec"]],
```

```
    "Phys": ["v30r6", "v31r0", ["Rec"]],
    "Moore": ["v50r0", "v51r0", ["Phys"]],
    "Analysis": ["v30r6", "v31r0", ["Phys"]],
    "Stripping": ["v15r1", "v15r2", ["Phys"]],
    "DaVinci": ["v50r6", "v51r0", ["Analysis", "Stripping"]]
}
```

This replaces the explicit versions passed on the command line and helps with automatically populating the preamble with dependencies. In addition, it allows collecting "highlight changes" from the dependent projects (useful for releasing application projects such as DaVinci).

To use this mode, clone the relevant projects under a directory (e.g. `~/stack`) and then put a JSON file in it (e.g. `~/stack/release_versions.json`), where the format is `["last released ver", "ver to release", ["dep1", "dep2"]]`. Then just run `lb-gen-release-notes -s ../release_versions.json` from each project directory.

Hints

- The command `lb-gen-release-notes` is only available in the new `LbEnv` environment. See LbEnv and LHCbGroupLogin or instructions on how to set it up
- In order to extract the release notes from Gitlab, you need to create a Gitlab token: go to https://gitlab.cern.ch/profile/personal_access_tokens and create a personal token with API scope (needs to be done only once). Make a note of the generated token (e.g. `dsfadgsdfhfj`) and then add it to your shell once per session (`export GITLAB_TOKEN="dsfadgsdfhfj"`)
- It's a good idea to check the validity of the markdown of the `vXrY.md` file before you commit it - paste its contents into a comments window in your Gitlab project and check that the format is what you expect and that all hyperlinks are working correctly

Preparing a release of the master or someOther branch, including some packages from other projects

From time to time it may be necessary to temporarily include into the release of a project some patches (most often bug fixes) from packages residing in other projects, without waiting for a new release of the entire software stack to pick up the fixes.

In the following example we prepare the release of version **vXrY** of the project **MyProject**, and add a package `OtherPackage` taking the version that was released with version `vNrM` of the `OtherProject` in which it resides

- Follow the instructions above to clone a copy of the project and to create a branch (**vXrY-release**) in which to prepare the files needed for release
- In this branch, checkout the additional packages to be temporarily added to the release:

```
git lb-use OtherProject
git lb-checkout OtherProject/vNrM OtherPackage
```

If more patches are required in the release, you can repeat `git lb-use` and/or `lb-checkout` for other projects and/or packages as much as needed

And then follow the same instructions as above to:

- Update the project version and dependencies
- Create the release notes
- Commit the changed files to gitlab

Now go to gitlab. **DO NOT make a merge request!** Simply tag the branch, using the same instructions as above

Post-processing

In all cases some post-processing is needed to get the tag released and to update the documentation.

- Trigger the release and request deployment - see here.
- Document the release: the release procedure creates automatically a directory in AFS where one can add documentation about the release in addition to the automatically generated doxygen documentation. The new release appears in a table at the URL http://cern.ch/lhcbdoc/myproject/releases (see for example the Brunel table). One can add comments to the table by adding a file called `description.html` to the ES directory $LHCBDOC/myproject/releases/vXrY
- Run `addrel.py project version branch` to add it to a branch in announcements.

-- MarcoCattaneo - 2019-02-13

This topic: LHCb > PrepareProjectReleaseGit
Topic revision: r14 - 2020-08-18 - PatrickSKoppenburg