Table of Contents

Development Use Cases.....1
- Motivation.....1
- HLT Use Cases.....1
  - Person X wants to develop a new HLT line.....1
  - New Loki functionality needed for a new HLT Line.....1
  - Test of the newly developed reconstruction code feects on HLT line under development.....2
  - Chosing which reconstruction changes are included in the HLT.....2
  - Preparation of HLT Settings.....3
- Framework and application.....3
  - Adding a single line fix to a production application.....3
  - Adding a single line fix to several production applications.....3
  - Distributed management of software releases.....4
- Analysis Uses Cases.....4
  - All LHCb developers are using old Moore, DaVinci, and Stripping versions in parallel with their development of the new code.....4
  - Data Analyst modifies a package and ships it to the Grid with Ganga.....5
- Reference data.....5
  - Update of Data Packages.....5
- Common Scripts.....5
  - Nightly tools.....5

Development Use Cases

Motivation

One of the conclusion of the 6th LHCb Computing Workshop is that we could benefit from modern development techniques and tools that enable, for example, teams to easily perform code reviews. Git in association with Gitlab/Github are potential candidates but imply a change that is not limited to the tooling used, but also affects the workflow used to develop and release the software.

The goal of this page is to ease the transition by listing the known use cases for development in order to make sure they are all covered.

HLT Use Cases

Person X wants to develop a new HLT line

Contact: Vava

Description: Developer needs to add HLT line for further analysis. The has to be reviewed to make sure it follows proper HLT conventions and does not slow down the whole trigger.

Frequency: High

Priority: High

Workflow: Developer creates a fork of the HLT repo, in that fork they create a new 'topic' branch of the relevant repository, develops his/her changes in this branch, and then submits a merge request from their fork to the main repository. (note: this assumes all development is contained to a single repository). The curator(s) of the 'official' repository decides whether or not, possibly after feedback from a code review, to accept the merge request.

New Loki functionality needed for a new HLT Line

Contact: Vava

Description: New Loki functionality needed for a new HLT Line (from Vanya)

Frequency:

Priority:

Workflow:

Vanya forks the repository which will host the augmented code, starts a new topic branch in this fork, implements the requested functionality, and finally creates a merge request from his forked repository to the main repository. Two different cases can happen: 1) the development is fully contained within the 'HLT' project repository, in which case no further action is required. 1) the changes pertain to other projects. This needs to be further worked out, but something on top of git subtree is presumably appropriate -- see eg. https://developer.atlassian.com/blog/2015/05/the-power-of-git-subtree/ or http://tdd.github.io/git-stree/ -- one can import 'part' of another git repo inside the current repo.

Discussion: I think the general recommendation is to avoid using subtree, submodule and other sub* tools with git (http://blogs.atlassian.com/2014/04/git-project-dependencies/). The real solution is to have a package manager that deals with dependencies between projects/repositories. What we need is an `environment` which contains one or more repositories. In this particular case you would clone the two repositories containing the changes you are interested in testing together (in this case the forks of the HLT and LoKi repositories which contain the proposed changes) and build them. Together they form an `environment`, in which you run what ever things you need to run. --Tim

I match your atlassian blog against submodules, and raise you a blog on why subtree is superior: http://blogs.atlassian.com/2013/05/alternatives-to-git-submodule-git-subtree/ On another note: unless we go for a 'one repo per package' model (which I don't like!), a package manager won't solve the problem of adding a single package (which is a small fraction of a repo) to another repo -- but that is the problem that subtree solves, see the item below on "Distributed management of software releases" which demonstrates how to 'split off' a directory from one repo, and subsequently 'add' it to another repo, while keeping the history. The claim is that one can also push back to the original parent repo, but I haven't tried that yet... -- Gerhard

Test of the newly developed reconstruction code feects on HLT line under development

Contact: Vava

Description: During HLT line development Alpha, Beta, Gamma have been working on new reconstruction options and once implemented, X needs to integrate these in order to properly test the rates etc of their lines

Frequency:

Priority:

Workflow:

persons Alpha, Beta and Gamma start topic branches for each of their developments. Once implemented, they each request a merge request. Curator X of the 'official' repository reviews these requests, based on eg. feedback from reviewers, and decides which one to merge first, lets take Alpha for this. At this point, it is possible that the changes from Beta and Gamma no longer cleanly apply, and X informs Beta and Gamma of this (actually, Beta and Gamma can see this from their gitlab merge request status pages), and asks them to resolve the conflicts, and then processes their requests.

Chosing which reconstruction changes are included in the HLT

Contact: Vava

Description:We eventually decide to merge only the reconstruction changes from Gamma and Beta, while Alpha needs to wait another round

Frequency:

Priority:

Workflow:

X only merges the requests from Gamma and Beta, and decides not to merge Alpha's request....

Discussion: is this meant to chronologically follow the previous scenario or is an alternative to it? --Tim

Preparation of HLT Settings

Contact: Vava

Description: At the end of the HLT line development process, HLT responsible Z needs to, with the WG liaisons, prepare the relevant HLT Settings files on top of all this and release the correct mixture of what we are happy with for use in the pit

Frequency:

Priority:

Workflow:

Assuming X merged the requests from Gamma and Beta, there is now a branch with their changes included. Z creates a new branch on top of this branch, and adds the required further changes, which results in yet another merge request. This merge request is then accepted and the basis for a release. (note: this assumes all changes are within a single repo -- the multiple repo version needs further thought, but again, probably would be done on top of git subtree infrastructure.)

Framework and application

Adding a single line fix to a production application

Contact: Marco

Description: As a result of production validation tests (or crashes in production), a bug is spotted in a released application and a maintainer is asked to produce a patch. In order to test the patch, the maintainer needs to check out a minimal compilation unit (in CMT: package), compile the fix and run tests against the production application. The user does not have access to large local storage and CPU resources, so compiling the whole application from sources is not an option. The patch then has to be deployed as a new version of the production application

Frequency: Several times per year

Priority:

Workflow:

Adding a single line fix to several production applications

Contact: Marco

Description: As a result of production validation tests (or crashes in production), a bug is spotted in a released application. A maintainer develops a patch. Since the same code is used by several production applications (e.g. multiple versions of Brunel and/or Moore), all these applications have to be patched in a consistent way

[This is the main justification for the granular tagging of packages in the CMT world]

Frequency: Several times per year

Priority:

Workflow:

Distributed management of software releases

Contact: Marco

Description: The LHCb software is managed as several independent applications, each with its release manager, which share code at different levels. Currently the software is packaged in projects which roughly correspond to the boundaries of what different applications need to share. Although all applications depend on the same software stack, different applications have different release cycles, so the lower levels of the stack can be (and are) released earlier than the higher levels. Typically LHCB, LBCOM, REC are released together (with BRUNEL), followed by PHYS which is then picked up by multiple releases of HLT+MOORE and ANALYSIS+STRIPPING+DAVINCI.

Later, DaVinci might need a patch to code residing in REC, but that should not be picked up by the corresponding versions of Brunel and Moore.

Frequency: Several times per year

Priority:

Workflow:

Proof-of-principle (to be further streamlined and properly 'packaged') of how to import a single 'package', in this case a specified subdirectory of one project 'inside' another project:

```
git clone ssh://git@gitlab.cern.ch:7999/LHCb-SVN-mirrors/Brunel.git  # check out the 'target'
git clone ssh://git@gitlab.cern.ch:7999/LHCb-SVN-mirrors/Rec.git  # check out the 'source' rep
cd Rec  # go into the source repo
git subtree split --prefix=Tf/PatAlgorithms -b PatAlgorithms-patch   # and create a branch in
cd ../Brunel # now go to the target repo
git remote add Rec ../Rec  # add a pointer to the Rec repository which contains the special Tf
git checkout -b brunel-with-patalgorithms-patch # create a branch for this version of Brunel w
git subtree add --prefix=Tf/PatAlgorithms Rec PatAlgorithms-patch  # add the PatAlgorithms-pat
```

Todo: use a specified tag of Rec (i.e. a tag which specifies Tf/PatAlgorithms), push the 'split' Rec repo back up to gitlab, and use it instead of a local repo when adding the subtree to the local Brunel repo. Push the Brunel repo back to gitlab, and tag it, and check that one can checkout the tagged version of Brunel, and that it includes Tf/PatAlgorithms. Try to modify Tf/PatAlgorithms inside Brunel, and push it back to both Brunel on gitlab, as well as Rec on gitlab... (and replace 'Brunel' with 'DaVinci' to really match the challenge)

Analysis Uses Cases

All LHCb developers are using old Moore, DaVinci, and Stripping versions in parallel with their development of the new code.

Contact: Vava

Description: This is needed to reprocess old data in one way or another, and need all this to happily coexist in a reasonably sized cmtuser directory.

Frequency:

Priority:

Workflow: Old code remains 'as is' in old repositories, so no action is needed....

Data Analyst modifies a package and ships it to the Grid with Ganga

Contact: Roel

Description: It is not uncommon for Analysts to have to fix/improve DecayTreeTuple Tools. They check out a few packages which they modify and ship to the grid with their jobs using Ganga. Would it be possible to do this with a whole project ?

Frequency: High

Priority: High (Can't customize DecayTreeTuple otherwise)

Workflow: See the "Distributed management of software releases" item as necessary prerequisite. Once that works, the next step is to integrate it into Ganga...

Reference data

Update of Data Packages

Contact: Ben

Description: Data packages contain reference data deployed by the software, currently grouped in the DBASE and PARAM projects but they are deployed individually on a regular basis. At the moment we have no replacement for the SVN workflow, as grouping them in GIT may not be a good idea (as we can identify managers per package, not per project).

Frequency: High

Priority: High

Common Scripts

Nightly tools

Contact: Ben/Marco

Description: The scripts to contain the nightlies allow to build a whole LHCb Software stack.

Frequency: Medium

Priority: Low

Workflow: Already managed in gilab, managed by Marco & Ben

Contact: Ben

Description: LHCb environment scripts

Frequency: Medium

Priority: Low

Workflow: Still in SVN, managed with CMT. Replacement already prototyped in Gilab (see LbEnv repo)

-- BenjaminCouturier - 2015-11-23

This topic: LHCb > LbSoftUseCases
Topic revision: r11 - 2015-11-28 - GerhardRaven