Package & release management

why should you care and how can we distribute the workload?

•••

1st Real Time Analysis Workshop
M.Clemencic, B.Couturier

LHCb Software stack

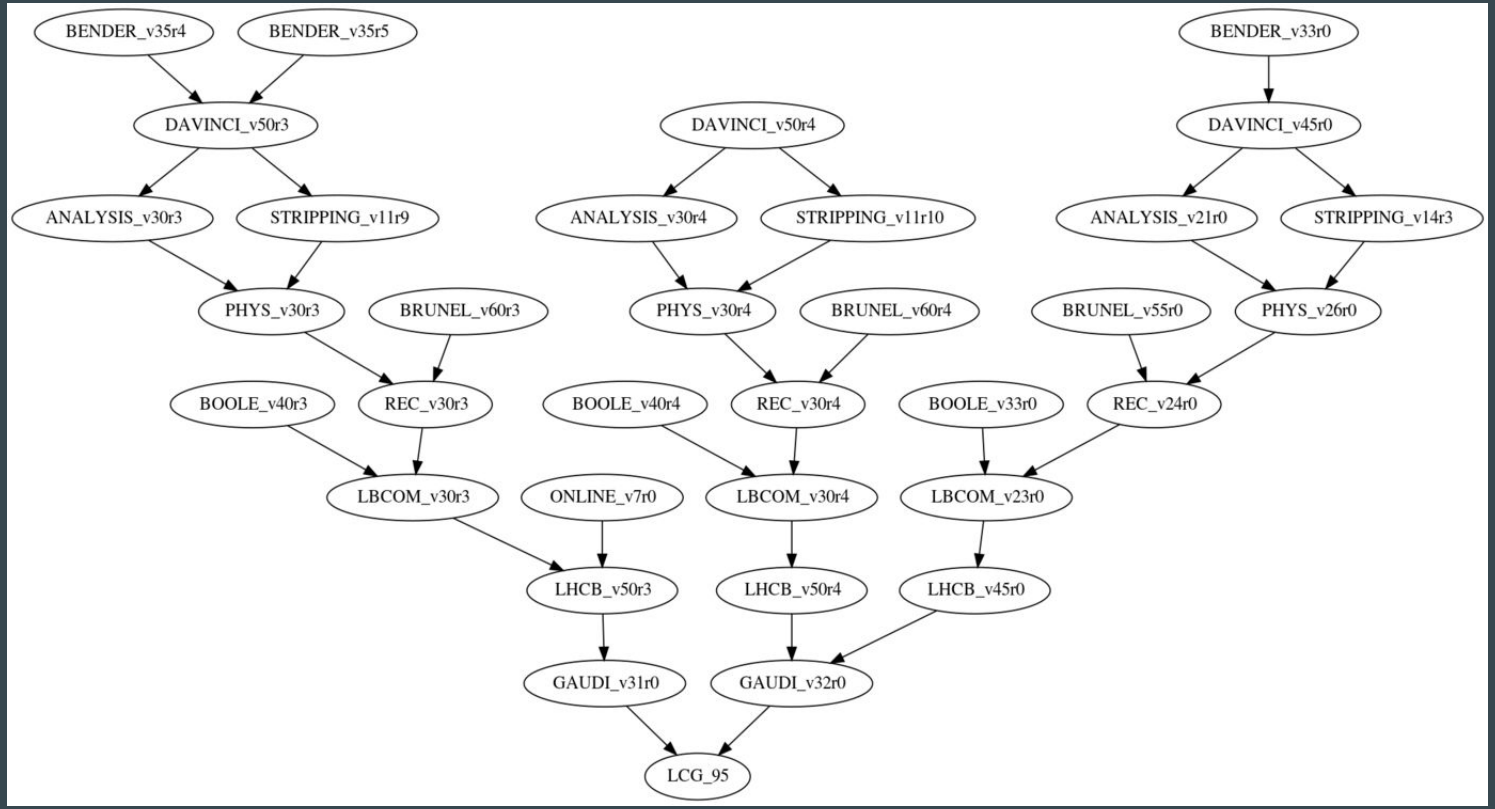
Requirements

1. **Reproducibility**
   We need to make sure we can re-run the code exactly as is. Forever (with forever TBD).
2. **Portability**
   We need to run on a variety of environments, on nodes we do not control (online farm, grid, lxplus) with operating systems that evolve slowly (slc5, slc6, centos7...)
3. **Performance**
   We want performance, and therefore the latest compilers and tools
4. **Quality**
   Of course we want as few bugs as possible...

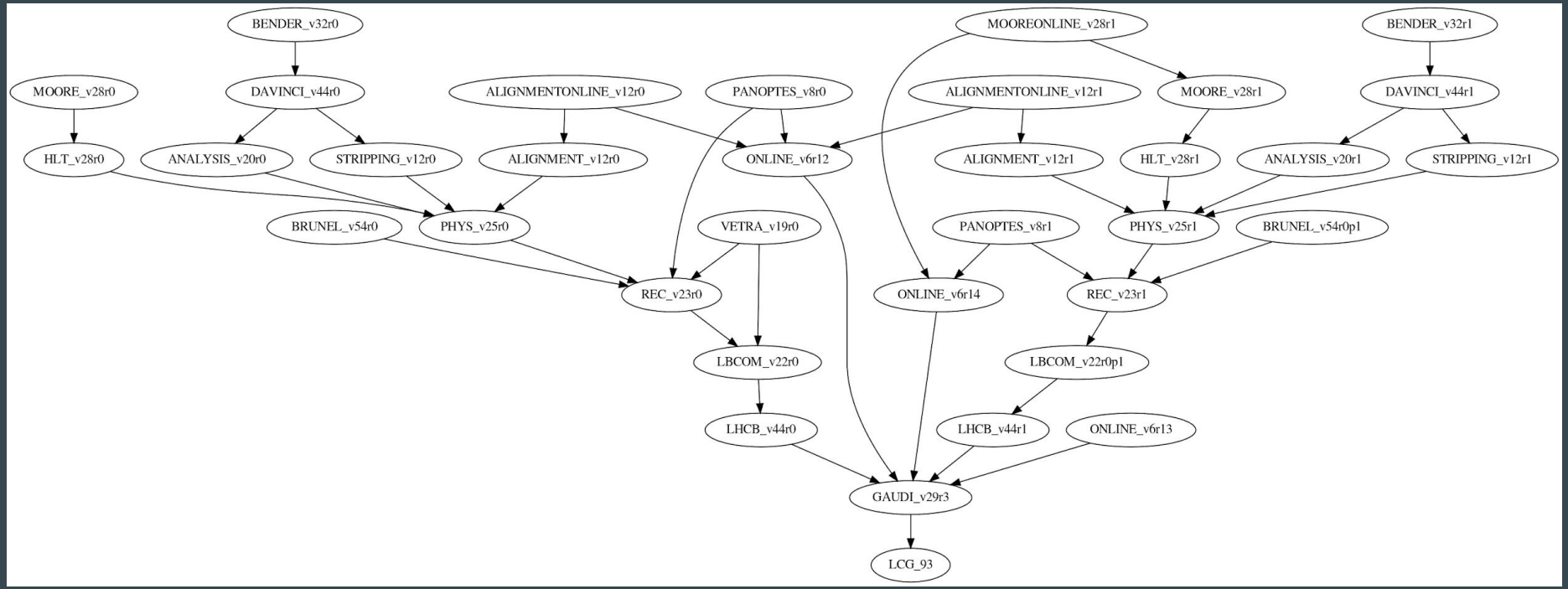
Some characteristics

- Several million lines of C++, quite a lot of python too
  - Using the Gaudi framework and ROOT
  - Requires careful testing (of functionality AND performance)
- Using external software from the LCG stack.
  Effectively an OS in an OS, which allows us to:
  - Use the latest compilers and C++ standard
  - Upgrade even core packages (e.g. binutils to allow producing AVX512 on centos7)
- Fine grained package and project structure to suit the developers needs and allow work on existing infrastructure (think lb-dev on lxplus)

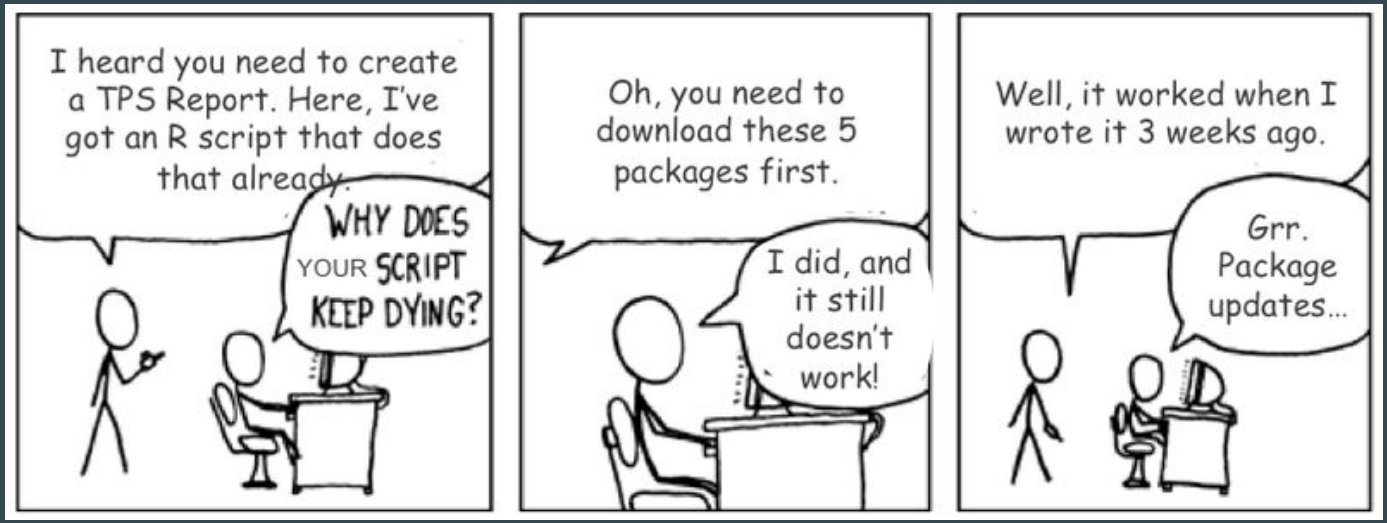
The LCG 95 stack



The LCG 93 stack



What do we risk ?

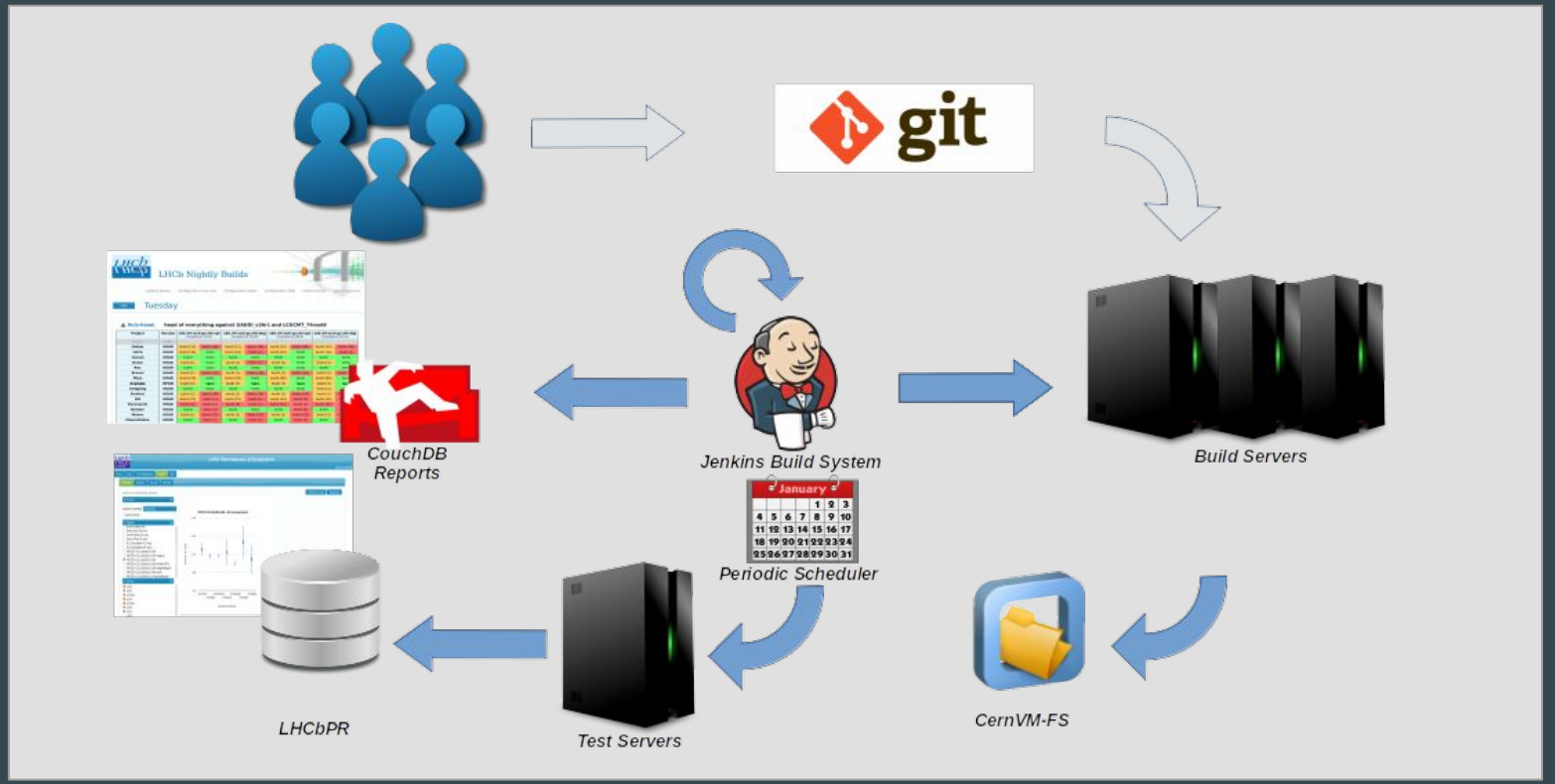


http://www.xkcd.com

How can we manage this ?

(or at least how we are trying to...)

The LHCb build and Release system



Source code management

Git version control + workflows for collaboration

- Allows to tune the workflows to our needs
- Can have fine grained permissions
- Main repository for source code preservation in the long term

Configuration and builds

Migrated to CMake a few years back

- Moved away from HEP specific “cmt” tool
- Aligning with industry standards as much as possible
- *LHCb CMake configuration files are being refactored to simplify them and to follow standard & modern CMake syntax* (c.f. Work by T.Cluzel, S.Ponce and M.Clemencic)

External software

We are using the LCG releases as produced by CERN EP-SFT, who is integrating the tool, providing builds and checking the errors. They are currently using their own tool (LCGCMake, https://gitlab.cern.ch/sft/lcgcmake)

- HSF Packaging group is reviewing alternatives
  (e.g. spack https://spack.io/, a tool popular in HPC)
- Nix (https://nixos.org/nix/) is also a contender

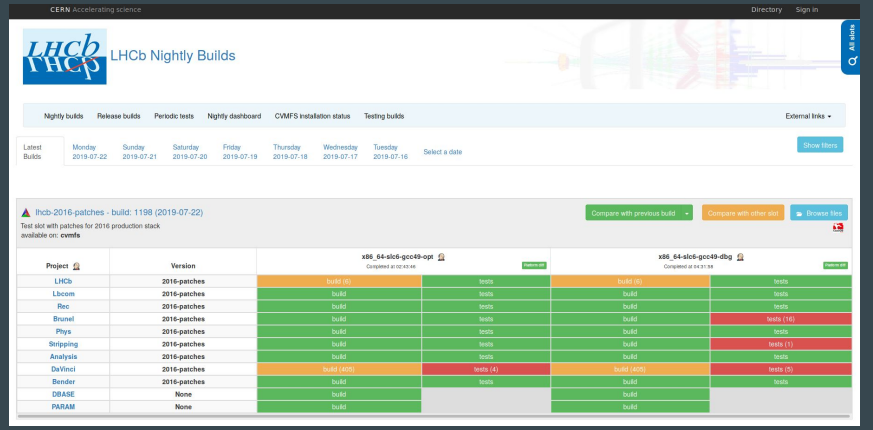
Now releasing all tools and environments scripts as standard python packages
(with our own pypi server).

Containers are also changing the deployment landscape,
maybe they will allow to simplify our toolset ?

Continuous Integration

Continuously building the software is mandatory to ensure quality

- LHCb “Nightly builds”
  - Currently using the Jenkins build management tool
  - Using the notion of “nightly slots” to manage the complexity
    Set of project dependencies and build options
  - Unit-tests are run straight after the build (but shouldn’t be too long)
- Builds performed on VMs provided by CERN IT



https://lhcb-nightlies.cern.ch/nightly/summary/

Integration and Regression tests a.k.a. LHCbPR

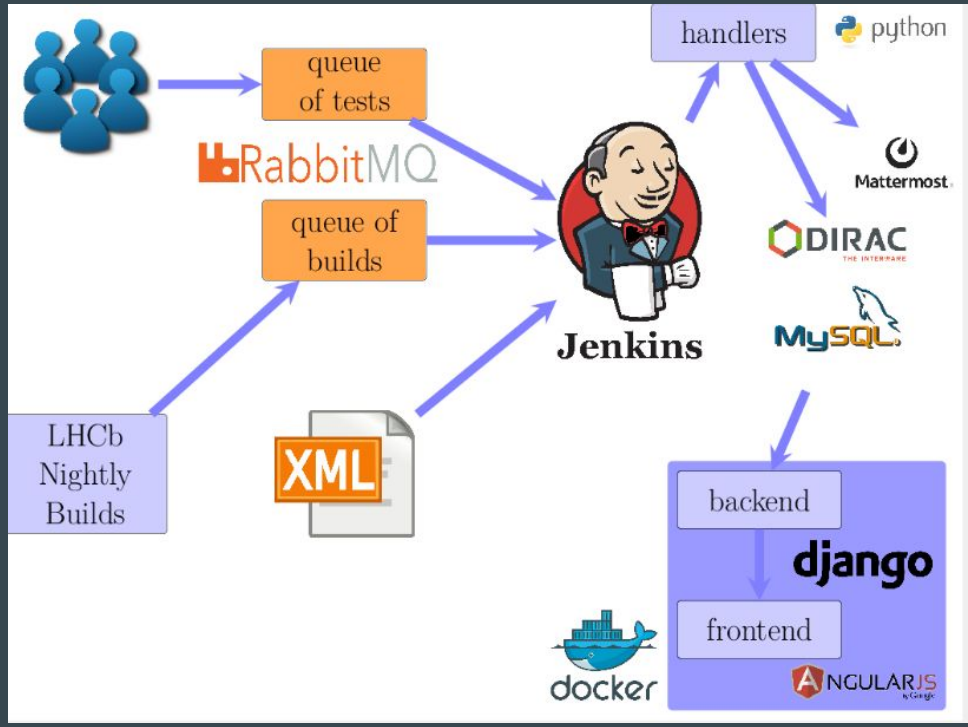
Running Integration and Regression tests

- Allows running longer test
- Not necessarily for every build
- Allows to use dedicated resources for the performance test, c.f. HLT1 perf tests

C.f. Maciej's report at the LHCb week:

https://indico.cern.ch/event/825670/contributions/3454204/attachments/1863511/3063580/LHCbPR_status_LHCbWeek_17Jun2019.pdf

Trigger and Simulation groups invested a lot of effort (c.f. Rob's work on the scripts)



Continuous Integration future

The current testing scheme is clearly not enough...

- Need to test every new Merge Request
  - Rob and Rosen already worked on this
  - Nightlies refactoring needed to make this practical
    (i.e. introduce checkout per project instead of per slot, rebuild only when needed)
    - *Work already started on this task*
      (c.f. Marco's talk at the LHCb Week:
      https://indico.cern.ch/event/825670/contributions/3454193/attachments/1863608/3063592/Comp_RTA_92_lhcb_week_Core_infrastructure.pdf*)*
    - Common effort with M.Clemencic, S.Chitic, R.Matev, R.Currie, M.Szymanski
- Also planning to use repository managers from the industry
  (such as Sonatype Nexus instead of EOS)

Continuous Integration future

- Why not use **gitlab-ci** ?
  - Wouldn't be able to manage our complex dependent builds, which is why we are continuing with Jenkins for the Physics software
  - Still recommended when possible (e.g. python packages)

- Looking into automating deployments
  maybe put in place Continuous Delivery (and Continuous Deployments one day?).

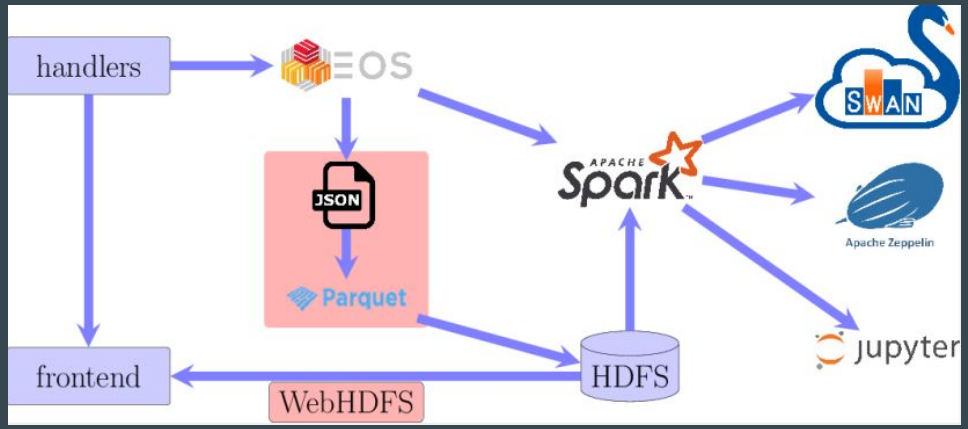
LHCbPR developments

LHCbPR is being improved to allow for easier exploitation of the results

- Using hadoop for storage to improve scalability
- And allowing to process the data using notebooks for greater flexibility

Still some needs to be addressed (e.g. improve visibility of the test results).

Mattermost integration was really effective on that topic



Integration of the User Environment

- Taylored to work with existing resources:
  - Tools like lb-dev were developed to compile one package out of a big project
  - Compiling the whole LHCb on lxplus is not practical
- Marco developed the “hackathon setup” to compile a whole stack
  - Can be practical to use on a desktop/laptop, in conjunction with containers
- Rosen investigating new ways to develop the stack: https://indico.cern.ch/event/825670/contributions/3454203/attachments/1863744/3064161/2019-06-17_Summary_of_QA.pdf

As it is user facing, this part needs to be clear and well documented

Deployment

Using the CernVM File System

- Global read-only file system
- Efficient as file deduplication is part of the design
- Installed on all grid nodes, online farm and on lxplus
- Used for releases as well as nightlies

Local installations are nonetheless possible using ***lbinstall***
(this part is highly dependent on the tools provided by the LCG stack)

LHCb tools and scripts

Still using a monolithic set of python tools in production (LBSCRIPTS).

Migrating to **LbEnv.** Why ?

LbEnv is a set of standard pypi packages which we release in a virtualenv on CVMFS

LbEnv developed to support Python 2.7 and 3 from the start...

- Standard pypi packages make the integration of externals tools and packages easier
- And of course allow to collaborate more easily within the experiment

c.f. Marco's slides at the LHCb week:
https://indico.cern.ch/event/825670/contributions/3454193/attachments/1863608/3063592/Comp_RTA_92_lhcb_week_Core_infrastructure.pdf

Conclusion

- Active developments at all levels
  - Some common effort between core soft and RTA
  - Slowly changing design to allow collaboration on all the tools (e.g. use of pypi packages, with automated testing)
- Common themes
  - Avoid re-developing tools already available in the industry
  - Simplify and focus on LHCb specific use cases
  - QA is critical (but very hard to get right!)

Functionality bound to the limits imposed by the chosen tools and infrastructure

We also need to keep up with the evolution of tools and services that we use

Next steps

- Clearly a lot of work in the pipeline
- Some major refactoring already ongoing for identified issues:
  - CMake config refactoring
  - Nightlies scripts refactoring, already in common between Core Software & RTA WP5 (Rosen, Rob, Marco, Stefan & Maciej)
- Missing features?
  - WP5 is the right place to have the big picture on the tools needed
    - And to prioritize the requests
  - Tools and environment are being refactored to allow for easier integration of new contributions