EPJ Web of Conferences **295**, 08008 (2024) https://doi.org/10.1051/epjconf/202429508008
CHEP 2023

Facilitating the preservation of LHCb Analyses with APD

Chris Burr[1,*], *Ben* Couturier[1,**], and *Ryunosuke* O'Neil[2,***]

[1]CERN
[2]The University of Edinburgh

Abstract. High Energy Physics experiments at the Large Hadron Collider generate petabytes of data per year that go through multiple transformations before final analysis and paper publication. Recording the provenance of these data is therefore crucial to maintain the quality of the final results. While tools are in place within LHCb to keep this information for the common experiment-wide transforms, analysts have had to implement their own solutions for the steps dealing with ntuples. This gap between centralised and interactive processing can become problematic. In order to facilitate the task, ntuples extracted by LHCb analysts via so-called "Analysis Productions" are tracked in the experiment bookkeeping database and can be enriched with extra information about their meaning and intended use. This information can then be used to access ntuples more easily: a set of Python tools allow querying of ntuple file locations with associated metadata, and integrate their processing within analysis workflows. The tools are designed with the intention of ensuring analysis code continues to be functional into the future and are robust against evolutions in how data is accessed. This paper presents the integration of these new tools into the LHCb codebase and demonstrates how they will be used in LHCb data processing and analysis.

1 Introduction

The Large Hadron Collider experiments at CERN record petabytes of data from proton and ion collision events inside their detectors. The analysis of those data is a complex process, involving many different steps and a number of different pieces of software. In line with the organisation's values, the CERN Open Data Policy [1] mandates that the experiments make their data available, following the FAIR [2] principles. This implies that the data be Findable, Accessible, Interoperable, Reusable; the requirements for the data to be easily findable and reusable imply that their provenance must be kept, from the initial events to the final numbers and plots in published papers. They also imply that all tools, and the environment required to run, then be cautiously recorded within each step.

The event processing chain for the LHCb experiment is schematised in figure 1. Data recorded by the detector is reconstructed into particle tracks in real-time, and the events involving physical processes of interest are recorded. Later on, they are reconstructed in further detail, building up particle decay chains which are then indexed such that analysts can

*e-mail: chris.burr@cern.ch
**e-mail: ben.couturier@cern.ch
***e-mail: r.oneil@cern.ch

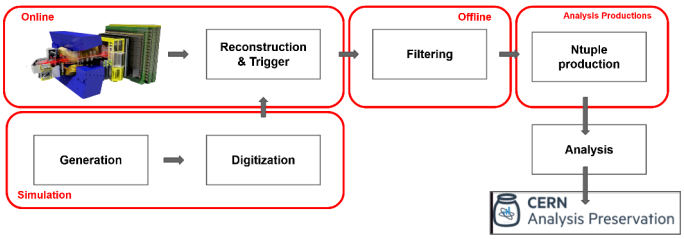

Figure 1: The LHCb data processing chain.

quickly find the particle decays they seek. These steps are centrally managed and carried out using versioned software from the LHCb software stack, and the files produced are registered in the LHCb book-keeping system [3]. LHCb analysts can use the *Analysis Productions* [4, 5] system, which records the configuration and scripts used while extracting *ntuples* from these files in the bookkeeping. *ntuples* are reduced files containing only the information needed for a particular analyses. At this stage, the centralised processing, managed and time-limited, is over. Analysis is done on the ntuples and analysis teams use various tools to perform measurements, a process that can last from several months to years. The analysis can subsequently be preserved in dedicated databases such as the CERN Analysis Portal [1]. During such a long process, it is difficult to guarantee that all data transformations are recorded faithfully, especially as this is not a linear process and many different processing methods may be tried and abandoned until the final goal is achieved. The goal of this work is to provide LHCb analysts with environments with all the tools needed for their tasks, as well as helpers that make it easy to identify the data to be used and facilitate access to it from analysis scripts and common workflows.

2 LHCb analysis environments

LHCb analysts should have access to the CERN/LHCb specific tools that are used in the centralised processing software, such as ROOT [6], but also standard tools from the Python ecosystem (e.g. numpy, pandas, scikit-learn) or dedicated to High Energy Physics (Scikit-HEP [7]). These can be easily used via environments created using the conda tool [2], comprised of packages of those tools from the Conda-forge repository [8]. This approach was presented at CHEP 2019 [3]. Such environments are made available to LHCb users using the CernVM file system [9]. Each environment is defined by a list of packages installed, and their versions, making it easy to recreate it if needed. For each analysis, we can therefore identify the environment needed for a processing step by a base operating system (or docker image) and the conda environment for the analysis packages. While the long term durability of the infrastructure involved is not guaranteed, enough information has however been recorded to reproduce the environment for preservation.

3 Data provenance

Within LHCb, analysts typically extract lists of file names from the book-keeping system, and/or directly copy data to local or user storage. The former is error-prone and not robust

[1] https://analysispreservation.cern.ch
[2] https://docs.anaconda.com/
[3] https://cds.cern.ch/record/2699537

as operational decisions related to the mass storage systems may cause files to be moved, thus changing some or all of the file names and invalidating such lists. The second method involves potentially unnecessary duplication of data, and file copies in user-managed storage may be later moved or deleted e.g. when users leave. Both approaches do not record the provenance of the files.

This paper proposes a way for LHCb analysts to enrich their datasets with tag metadata i.e. key/value pairs, using the Analysis Productions system. Data produced using this system is registered in the book-keeping system and can be selected using tags added by analysts, giving meaning to the samples and preserving information about data provenance in the analysis code.

Some of this metadata can be derived directly from the Analysis Productions jobs, as they are key parameters to any file in the LHCb book-keeping, e.g. whether we have data or a Monte Carlo simulation, the data type, which corresponds to the data taking year, the type of decay in case of simulated data. For each analysis, it is possible to add one or several tags to indicate what the data is used for in the analysis (e.g. evaluating errors, double checking resolution). These tags can be set in the LHCb Analysis Productions web application as shown in Figure 2.

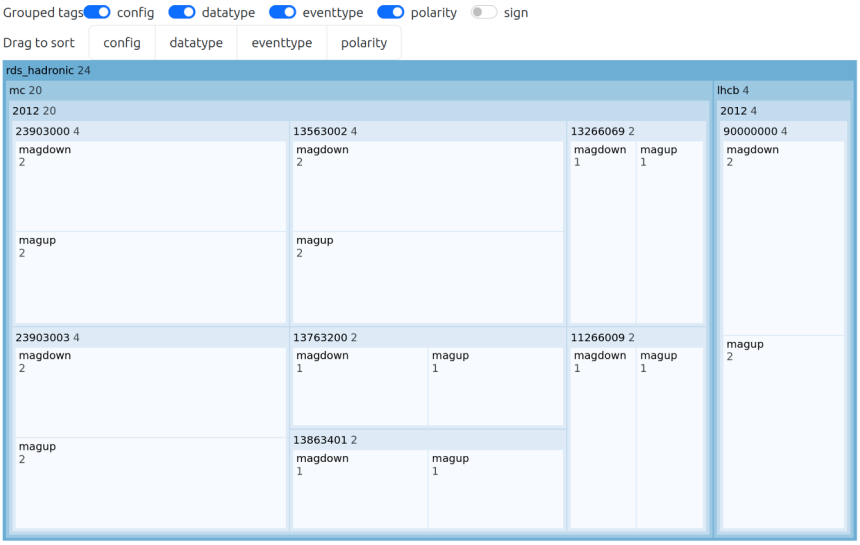

Figure 2: Tree view visualisation of the samples produced for an analysis, grouped by configuration (i.e. real data or Monte Carlo simulation), data taking year, event type and magnet polarity.

Files generated by Analysis Productions can therefore be identified by working group and analysis name, as well as several tag/value pairs. This also makes data sharing between analyses easier: provided conventions are established for the meaning of the various tags, it becomes possible to mine the data previously derived from LHCb output.

The Analysis Productions web application also allows the referencing of files from other Analysis Productions, thus making sure the use of those files is properly recorded, which is helpful for data management purposes.

4 The Analysis Productions Data Python package

To be useful in the context of data analysis, a programming interface has to be defined to query the files and specify which ones to process. As well as a graphical user interface, the Analysis Productions application proposes a REST [10] (Representational State Transfer) interface that can be queried from any programming language. This type of interface has the advantage that, for long term preservation, it is possible to replace the server by a static website serving the data for the LHCb analyses. The Python language [11] is very popular amongst LHCb analysts, and it is therefore a good candidate to provide a client interface to the system: this is the goal of the Analysis Productions Data (APD) Python package. APD provides a simple interface to the Analysis Productions server: it is a Python package published to standard repositories (the Python Package Index pypi.org) and conda-forge, and it is installed in the LHCb analysis environment available to users. It has minimal dependencies and can be installed on any machine, independently of the other LHCb software applications. Using APD from Python scripts requires minimal effort from the analysts, who do not have to deal with a list of files any more, as in Listing 1.

```
from apd import get_analysis_data
dataset = get_analysis_data("sl", "rds+hadronic")
files = dataset(config="lhcb", datatype="2012", polarity="magdown",
↪  eventtype="90000000", sign="rs")
```

Listing 1: Retrieving a list of files for the "SL/RDs_hadronic" analysis for data for polarity *magdown* and the sample with *right-sign* decay.

APIs have not been written for other languages, such as C++ for ROOT macros, but it is easy to invoke the Python API and save the list of files to CSV or JSON format for consumption from other languages. In order to avoid processing samples that are derived from the same input data with different configurations, which would be problematic, some safeguards have been put in place: APD checks that this is not the case and throws an exception if it is. Listing 2 illustrates this case.

```
dataset = get_analysis_data("sl", "rds_hadronic")
dataset(config="lhcb", datatype="2012", polarity="magdown", eventtype="90000000")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/cvmfs/lhcbdev.cern.ch/conda/envs/default/2023-04-26_20-20/"
  "linux-64/lib/python3.10/site-packages/apd/analysis_data.py", line 391, in __call__
    raise ValueError("Error loading data: " + error_txt)
ValueError: Error loading data: 1 problem(s) found
{'config': 'lhcb', 'polarity': 'magdown', 'eventtype': '90000000', 'datatype':
↪  '2012'}: 2 samples for the same configuration found, this is ambiguous:
    {'config': 'lhcb', 'polarity': 'magdown', 'eventtype': '90000000', 'datatype':
     ↪  '2012', 'sign': 'rs', 'version': 'v0r0p4882558', 'name':
     ↪  '2012_magdown_data_bsdstaunu', 'state': 'ready'}
    {'config': 'lhcb', 'polarity': 'magdown', 'eventtype': '90000000', 'datatype':
     ↪  '2012', 'sign': 'ws', 'version': 'v0r0p4882558', 'name':
     ↪  '2012_magdown_data_bsdstaunu_ws', 'state': 'ready'}
```

Listing 2: Retrieving data with a list of tags selecting multiple datasets.

In conclusion, the APD Python package is an abstraction layer that allows analysis code to query the data files to be processed by a series of meaningful tags, avoiding hardcoded file

lists. This in itself however does not completely allow tracking the provenance of the results: the integration with a workflow tool can help further in that direction.

5 Snakemake interface

We chose to provide an interface to the Snakemake workflow engine [12], as it is adapted to data analysis and already popular among LHCb physicists as it is taught during the LHCb Starterkit [13], the software introduction course aimed at LHCb newcomers. Furthermore, this is a Python based tool and therefore allows using the code described previously. We propose an integration that eases the use of ntuples created by LHCb Analysis Productions within the workflows, by specifying the metadata tags identifying the dataset. The tags can be matched to Snakemake wildcards, therefore allowing for very flexible workflows. Furthermore, APD returns Snakemake remote objects for the XRootD protocol [4], which allows Snakemake to check the modification dates of remote files (this is the criteria to decide which workflow artefacts are outdated and should be recreated).

```
rule create_histo:
   input:
       data=lambda w: dataset(config=w.config, datatype=w.datatype,
                              eventtype=w.eventtype, polarity=w.polarity)
   output: f"bmass_{{config}}_{{datatype}}_{{eventtype}}_{{polarity}}.root"
   run:
       import ROOT
       inputfiles =  [ f for f in input ]
       f = ROOT.TFile.Open(output[0], "RECREATE")
       rdf = ROOT.RDataFrame("SignalTuple/DecayTree", input)
       h = rdf.Histo1D(("BM_Hist","BM_Hist", 200, 0., 25e3), "B_M")
       h.Write()
       f.Close()
```

Listing 3: Snakemake rule that creates a histogram of a single variable (`B_M`).

Listing 3 shows a Snakemake rule that creates a histogram of a single variable(`B_M`) in the TTree "SignalTuple/DecayTree" for any combination of config (i.e. Monte Carlo simulation or real data), datatype, event type (for simulation), polarity. Workflow tools inherently track the dependency graph between rules, and commonly allow displaying them; Figure 3 illustrates the dependency graph for a workflow that applies the rule presented in Listing 3 to two event types and to the two LHCb magnet polarities and then combines the results.

6 Continuous integration and testing

Using Snakemake during an analysis allows dependencies between analysis artefacts to be tracked using rule definitions, but the accuracy of those workflows can only be ensured if they are tested on a regular basis. Published LHCb analyses are required to provide a GitLab [5] repository on the CERN GitLab instance, with relevant scripts for preservation.

The GitLab Continuous Integration system (GitLab CI) is therefore a convenient way to run the workflows at every new change in the code committed to the repository. Fully specified and versioned conda environments, available globally via CVMFS, can be used to

[4] https://xrootd.slac.stanford.edu/
[5] https://about.gitlab.com/

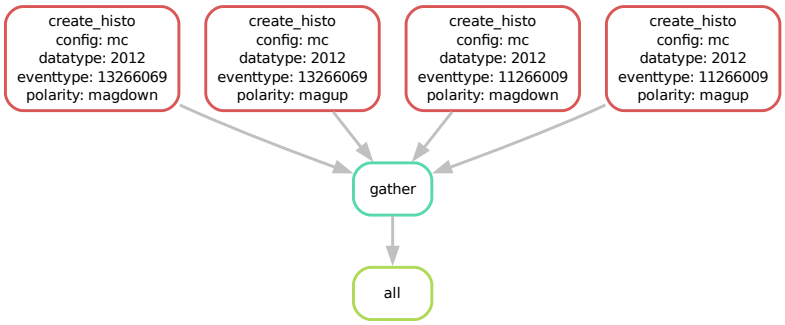

Figure 3: Dependency graph to apply the `create_histo` (see Listing 3) rule to two event types and two LHCb magnet polarities.

run analysis scripts in a reproducible way. They feature ROOT, common Python packages (including matplotlib, numpy, scipy) as well as Scikit-HEP [7].

One last component is needed: a storage system to host data files. LHCb production files are hosted on the Worldwide LHC Computing Grid (WLCG [14]), and accessed via the XRootD protocol. Analysis Productions files are kept at the CERN site, on the EOS [15] storage instance. However, access to storage, both to read or write data, requires proper authentication and authorisation.

7 Authentication and Authorisation

Access to Analysis Productions information is restricted to the members of the LHCb collaboration. Users accessing Analysis Productions must authenticate using the CERN Single Sign On (SSO) system, access being granted to all members of the LHCb collaboration as defined by the *lhcb-general* CERN e-group.

Authentication of users from Python scripts or command line tools is done using OpenID Connect [16] to authenticate and authorise users against CERN SSO. Invoking the `apd-login` command creates a request and blocks until the user has authenticated with CERN SSO and authorised the login. Once this is done, `apd-login` is provided with a short-lived JSON Web Token [17] (JWT) that grants access to the Analysis Productions data. This is cached on the local file system for further use.

Several authentication methods can be used when accessing data stored on the EOS system at CERN via the XRootD protocol: in the case of interactive access, users can either have CERN Kerberos tickets, or X509 grid proxies (also used to submit jobs on the WLCG). Any process with access to such credentials may carry out any of the actions permitted to the user. It is possible to delegate Kerberos credentials to a continuous integration system (CI), but this is not advised as this delegates all of the user's rights to the CI. It is safer to create dedicated accounts with limited rights and accesses. This is not a very practical way to delegate credentials to a continuous integration system, as all those extra accounts have to be managed, and a better solution had to be found.

Each GitLab CI job is given a JWT by the GitLab server; it is available in the environment under the name `CI_JOB_JWT_V2`. This is a way for the CI job to claim that it was created by GitLab, following the standard defined in Ref. [17]. The job can pass it on to any external service that can therefore check that the token was indeed generated by GitLab. Such tokens are time limited, to the maximum duration of the CI job.

We use such tokens to identify CI jobs, and generate for them EOS tokens for a set of directories defined in a configuration database for the project. EOS tokens are a finer grained way to give access to files or directory sub-trees in the EOS filesystem: a token can give access to a portion of the EOS filesystem, in read-only or read-write mode. The drawback is that they need to be appended to the file URL and are authenticated on the server side. Furthermore, depending on the path one wants to read or write, the appropriate one must be chosen from all the tokens passed to the job.

The APD package provides a way for the CI job to get the EOS tokens: when in a GitLab environment (i.e. when the variable `CI_JOB_JWT_V2` is defined), the `apd-login` command forwards it to the APD service and receives EOS tokens defined for the project. It then provides two methods: the first one, `apd.auth(url)` appends a read-only EOS token to the file URL if an appropriate token is found. In interactive use with Kerberos tickets, for example, this function is a simple pass-through. The second one, `apd.authw(url)` appends a read-write token if available instead. The Analysis Productions web application allows configuring the list of EOS tokens per GitLab repository that will be granted to each CI job. Furthermore, the creation of EOS tokens is recorded in logs, and the lifetime of the tokens is short, limited to the lifetime of the CI jobs themselves.

Using APD requires users to modify their code. However, once this is done, their code can get access to data files in both interactive and GitLab CI use cases.

The Snakemake interface also appends tokens to input files from EOS files at the same time as it returns Snakemake XRootD wrappers for remote files. This allows Snakemake rules to transparently read files from CVMFS. When writing back to EOS, it is necessary to invoke the `apd.snakemake.remote()` method, specifying the write mode.

8 Further work

The system in place allows creating workflows that derive data from ntuples created by the Analysis Productions framework. The derived data can be stored in EOS, with naming conventions allowing to find out which data it contains. This is not however very satisfactory and it would be useful to be able to register this data in the LHCb bookkeeping and add metadata as can be done with Analysis Productions output. This would allow for easier sharing between analyses, instead of re-running common workflows. The idea is therefore to introduce *Derived Analysis Productions*, on a similar principle as the Analysis Productions except that the workflow would be run by GitLab CI instead of being driven by the LHCb grid software.

The current Analysis Productions system and related APD Python package rely on the fact that all data is stored on EOS at CERN. This simplifies access as the site is inherently known, and relies on the EOS tokens to grant access to file and directories from the continuous integration system. This constraint could actually be reviewed, as efforts are underway for the WLCG to use tokens instead of the current X509 infrastructure, and maybe use pre-signed URLs for data access, as can be done with the Amazon S3 [6] interface. APD would have to be updated to support pre-signed URLs, but the same interface could be kept. This would open the door to the development of analysis scripts that could process data stored on public clouds for example.

9 Conclusion

Analysis Productions and the APD Python package help LHCb analysts accessing their data while keeping track of the provenance of the derived artefacts: Analysis Productions data can

[6] https://docs.aws.amazon.com/s3

be found using metadata declared at production time, and integration with the Snakemake workflow tools allows tracking dependencies between derived artefacts. Integration with the GitLab CI continuous integration system improves reproducibility further. This does not solve all problems related to the preservation of analyses, but facilitates the use of good practices while keeping a low barrier of entry. The Analysis Productions framework and tools have been received enthusiastically by analysts and are the standard system used for creating ntuples in Run 3. Given the clear advantages the APD package provides to both analysis workflow development and data provenance tracking, it is experiencing rapid uptake as we explore the LHCb Run 3 dataset.

References

[1] CERN, Tech. rep., CERN, Geneva (2020), https://cds.cern.ch/record/2745133
[2] M. Wilkinson et al., *The FAIR Guiding Principles for scientific data management and stewardship* (2016), https://doi.org/10.1038/sdata.2016.18
[3] Z. Mathe, *Feicim: A browser and analysis tool for distributed data in particle physics* (2012), http://cds.cern.ch/record/1491175
[4] C. Burr, *Analysis Productions: A declarative approach to ntupling* (CHEP 2023), https://cds.cern.ch/record/2860345
[5] N. Skidmore, E. Rodrigues, P. Koppenburg, PoS **EPS-HEP2021**, 792 (2022)
[6] R. Brun, F. Rademakers, Nucl. Instrum. Meth. A **389**, 81 (1997)
[7] E. Rodrigues et al., EPJ Web Conf. **245**, 06028 (2020), 2007.03577
[8] conda-forge community, *The conda-forge Project* (2015), https://doi.org/10.5281/zenodo.4774216
[9] J. Blomer et al., J. Phys. Conf. Ser. **331**, 042003 (2011)
[10] R.T. Fielding, Doctoral dissertation, University of California, Irvine (2000), http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
[11] G. Van Rossum, F.L. Drake, *Python 3 Reference Manual* (CreateSpace, Scotts Valley, CA, 2009), ISBN 1441412697
[12] F. Mölder et al., *Sustainable data analysis with snakemake [version 2; peer review: 2 approved]* (2021), https://doi.org/10.12688/f1000research.29032.2
[13] A. Puig (LHCb Starterkit Team), J. Phys. Conf. Ser. **898**, 082054 (2017)
[14] I. Bird et al., *Update of the Computing Models of the WLCG and the LHC Experiments* (2014), https://cds.cern.ch/record/1695401/files/LCG-TDR-002.pdf
[15] A. Peters, E. Sindrilaru, G. Adde, J. Phys. Conf. Ser. **664**, 042042 (2015)
[16] D. Fett, R. Kuesters, G. Schmitz, *The Web SSO Standard OpenID Connect: In-Depth Formal Security Analysis and Security Guidelines* (2017), https://arxiv.org/abs/1704.08539
[17] M. Jones, J. Bradley, N. Sakimura, *RFC 7519: JSON Web Token (JWT)* (2015), https://datatracker.ietf.org/doc/html/rfc7519