EPJ Web of Conferences **214**, 06034 (2019) https://doi.org/10.1051/epjconf/201921406034
CHEP 2018

REANA: A System for Reusable Research Data Analyses

Tibor Šimko[1,*], *Lukas* Heinrich[2], *Harri* Hirvonsalo[3], *Dinos* Kousidis[1], and *Diego* Rodríguez[1]

[1]CERN, Geneva, Switzerland
[2]New York University, New York, NY, USA
[3]CSC, Espoo, Finland

Abstract. The revalidation, reinterpretation and reuse of research data analyses requires having access to the original computing environment, the experimental datasets, the analysis software, and the computational workflow steps which were used by researchers to produce the original scientific results in the first place.
REANA (Reusable Analyses) is a nascent platform enabling researchers to structure their research data analyses in view of enabling future reuse. The analysis is described by means of a YAML file that captures sufficient information about the analysis assets, parameters and processes. The REANA platform consists of a set of micro-services allowing to launch and monitor container-based computational workflow jobs on the cloud. The REANA user interface and the command-line client enables researchers to easily rerun analysis workflows with new input parameters. The REANA platform aims at supporting several container technologies (Docker), workflow engines (CWL, Yadage), shared storage systems (Ceph, EOS) and compute cloud infrastructures (Kubernetes/OpenStack, HTCondor) used by the community.
REANA was developed with the particle physics use case in mind and profits from synergies with general reusable research data analysis patterns in other scientific disciplines, such as bioinformatics and life sciences.

1 Introduction

The reproducibility of scientific results is crucial for advancing science and testing new theories and hypotheses. Yet research is rarely reproducible. In a recent poll conducted by Nature, more than half of 1500 scientists working in various scientific disciplines replied that they could not reproduce their own or another group's results[1]. Reproducibility is usually ensured by documenting all the processes how the research is conducted. In the computational data analysis domain, this means to capture the full information about input data and parameters, the analysis software, together with the operating system runtime environment[2] and the detailed computational steps and recipes how the researcher performed the analysis and produced the original results.

The experimental data in particle physics is expensive to take. The testing of new theories sometimes demands to "resurrect" data from many dozens of years ago[3]. It is therefore necessary to preserve well-structured information about data, software, compute environments, and the associated analysis pipelines in order to facilitate future data reuse[4, 5].

*e-mail: tibor.simko@cern.ch

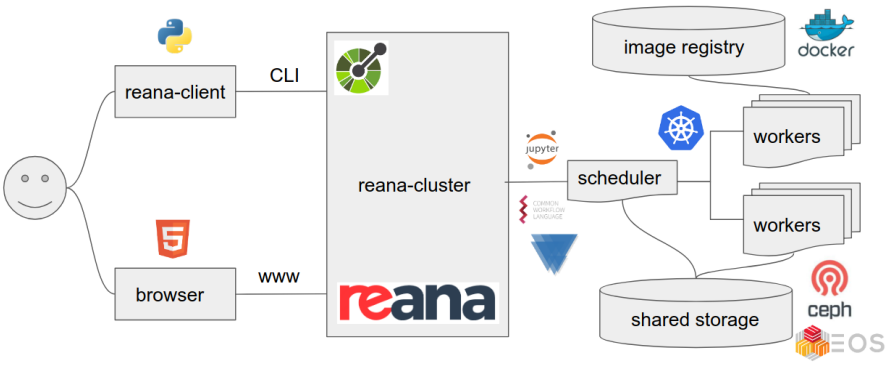

Figure 1. An overview of the REANA platform. User can interact with the platform via a command line and Web clients. The platform dispatches users' requests to workflow engines and runs necessary tasks on the containerised cloud. The workflow tasks share the same workspace throughout the run.

In this paper we describe the nascent REANA platform[6] that aims at facilitating reproducible science practices by leveraging on industry-standard container technologies useful for preserving and reinstantiating runtime environments. REANA allows users to structure their analyses and run them on remote computational clouds. The overall platform is described in Section 2. The user interaction with the platform is discussed in Section 3. The architecture of the REANA system is presented in Section 4. Some illustrative examples from particle physics analyses are shown in Section 5.

2 REANA platform

REANA is a reusable and reproducible research data analysis platform born to facilitate code and data reuse. The platform generalises computational practices used in the particle physics community and aims to facilitate the adoption of declarative workflow systems to run data analysis processes on remote compute clouds.

The REANA platform overview is presented in Figure 1. It consists of a **client** used by the researchers and a **cluster** that runs the platform itself. The users can interact with the system by means of a command-line client or a web user interface. Typical usage scenarios will be described in Section 3. The REANA cluster is composed of a set of micro-services that permit to instantiate, launch and monitor computational workflows based on client requests. The architecture of the platform will be described in Section 4. The computational workflows run on supported backends (Kubernetes on OpenStack) and use container technologies (Docker) to instantiate necessary environments. A shared storage (Ceph) is used for the various workflow steps to exchange temporary results.

One of the particularities of the REANA platform is the abstraction of user practices. Different research teams use different tools and facilitating reproducibility at large means supporting various computational workflow practices used by the community. REANA supports several such scientific pipeline or workflow systems. For simple user needs, the Serial workflow engine implements the sequence workflow pattern [7] and provides an easy entry point to the REANA platform by allowing to run simple "shell script" use cases where commands are run sequentially and each step produces outputs for the next step. For more realistic data analysis needs, it is necessary to support parallel execution commands. REANA

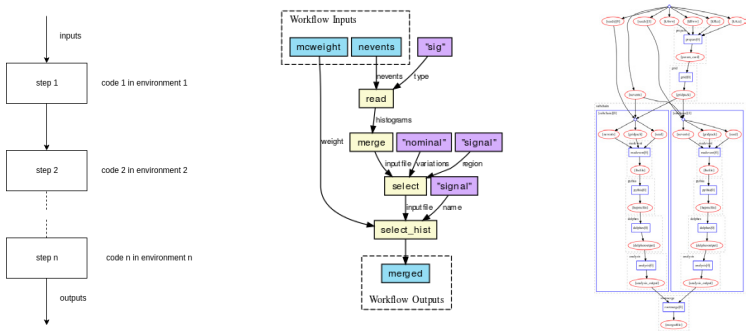

Figure 2. Examples of a serial workflow (left) and more complex Directed Acyclic Graph (DAG) workflows using CWL (center) and Yadage (right) workflow specifications. The Serial workflow is using the sequence workflow pattern where each computational step can run in a different containerised environment and the outputs of previous steps are passed along as inputs to later steps. The CWL and Yadage workflows are an example of DAG computations where the steps can be run in parallel in a "map-reduce" fashion. The full computational graphs in particle physics analyses can consist of several thousands nodes.

supports two of such systems; the Common Workflow Language standard [8] that emerged notably in the bioinformatics and the life science scientific domains, and Yadage workflow system [9] that was born in the particle physics community itself. Both systems allow to express the computational workflow in the form of Directed Acyclic Graph (DAG) that the task scheduler then executes on supported backends. The Serial, CWL and Yadage based analysis pipelines are illustrated in Figure 2.

In addition to supporting multiple workflow systems, REANA aims at supporting various computational backends used in the community. Currently the main execution platform is the containerised compute cloud running Kubernetes jobs using Docker container technology. Support for more compute backends, such as HTCondor and other High Performance Computing and High Throughput Computing platforms, is under preparation.

Due to its focus on supporting diverse reproducibility practices in diverse heterogeneous research teams, the REANA platform aims at becoming a multi-user, multi-workflow, multi-backend platform that accompanies users in their daily practices, all the while leveraging and facilitating the use of modern container technologies. Making it easy for researchers to write reproducible analysis pipelines helps to prepare the data analysis preservation and to facilitate its future reuse.

3 REANA client

The goal of a reproducible research data analysis is, in essence, to provide structured "runnable recipes" addressing (1) what the input is (data files, parameters, live database calls); (2) what software was used to analyse the data (custom code, frameworks, notebooks); (3) which computing environments were used to run the analysis software (operating systems, software packages and libraries, CPU and memory resources); (4) which computational steps were taken to run the analysis (simple shell commands, structured computational workflows, local or remote task execution). This allows to instantiate the analysis on the compute clouds and run the analysis to obtain its (5) output results.

The researchers express the analysis structure by means of a `reana.yaml` specification file describing the inputs, the workflow steps and the produced outputs, see Figure 3. The

```
version: 0.4.0
 inputs:
   files:
     - code/mycode.py
     - data/mydata.csv
   parameters:
     myparameter: myvalue
 workflow:
   type: cwl
   file: workflow/myworkflow.cwl
 outputs:
   files:
     - results/myplot.png
```

Figure 3. An example of the `reana.yaml` specification file describing a data analysis. Note the usage of runtime code, data and parameters, the definition of computational workflow, and the expected outputs. This specification file (and its linked resources) sufficiently captures analysis steps so that they can be executed on the remote REANA cloud platform.

```
$ export REANA_SERVER_URL=http://example.org # connect to REANA cloud
$ export REANA_ACCESS_TOKEN=XXXXXXX
$ reana-client create -n myanalysis          # create new analysis
$ export REANA_WORKON=myanalysis
$ reana-client upload ./code ./data          # upload code and data
$ reana-client start                         # start workflow run
$ reana-client status                        # check its progress
$ reana-client download                      # download final plots
```

Figure 4. The typical `reana-client` interaction scenario of a user running the example analysis presented in Figure 3. The corresponding sequence diagram is shown in Figure 5.

researchers then install a **reana-client** command-line utility that can connect to a remote REANA cloud deployment. The client offers a set of commands that allow to upload and run the analysis. Starting from the given `reana.yaml` specification file, the users can `create` a workspace, `upload` input data and runtime code, `start` the workflow, monitor its progress `status`, list their running `workflows`, and finally, `download` the output results. Figure 4 shows a typical user interaction scenario with the platform using the command-line client. Figure 5 illustrates this interaction via a generic sequence diagram.

4 REANA cluster

The user requests sent by the REANA clients enter the REANA cluster for execution. This section describes what is happening inside the platform.

The REANA cluster deployment is composed of a set of micro-services that to manage and monitor container-based computational workflows. The component architecture is presented on Figure 6. The main components and their roles are:

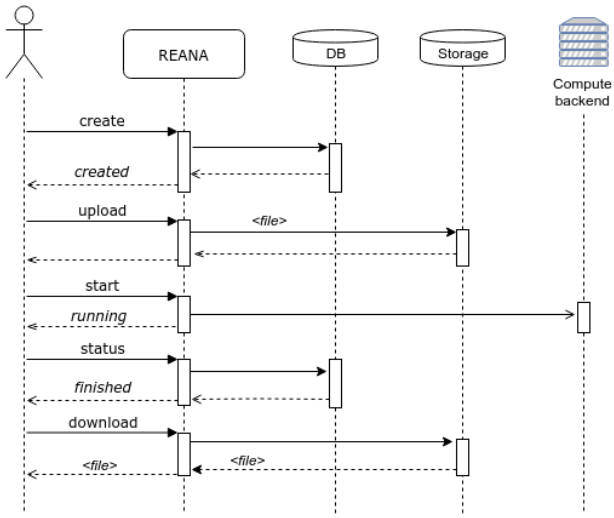

Figure 5. The sequence diagram expressing typical interaction between the REANA client and the REANA cluster.

- `REANA-Server` implements the public REANA API which offers all needed functionality for users to reproduce their analyses. Its main responsibility is authentication and authorization of user requests, once this is performed, it will pass the message (workflows) to the next component, the REANA-Workflow-Controller;
- `REANA-Workflow-Controller` takes care of instantiating and managing computational workflows, offering means to control workflow workspaces, i.e. populating it with the needed files, and offering actions such as starting, stopping and deleting workflows;
- `REANA-DB` provides database models and utilities;
- `REANA-Workflow-Engine` components take care of executing computational workflows; three engines are available: simple Serial sequential workflow engine, full Common Workflow Language (CWL) and Yadage workflow engines;
- `REANA-Job-Controller` launches and manages jobs on compute platforms. Its main responsibility is to offer a common API to submit and manage jobs on multiple backends.

The components are developed in separate source code repositories and they co-exist in a micro-service architecture. The common utilities and schemas are abstracted in the `REANA-Commons` package. The robustness of the platform is achieved by means of unit and integration tests for which `pytest-REANA` package provides fixtures and test utilities.

The components are developed mostly in the Python programming language and the Flask web development framework. The REST API protocols are described using the OpenAPI standard. The Bravado library is used for inter-component communication.

The REANA platform and its components can be easily deployed on Kubernetes clusters by means of a provided **reana-cluster** utility script, see Figure 7.

5 Examples

The developed REANA platform and its workflow approach was validated on a set of realistic particle physics analyses form the four LHC experiments. The examples include ALICE

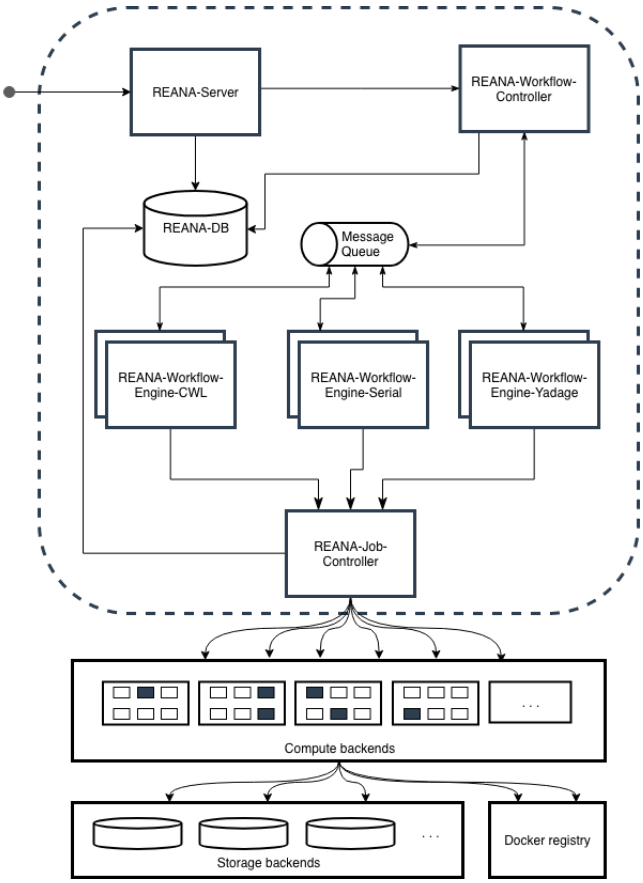

Figure 6. REANA cluster architecture and its components. The message passed to the cluster is received by API server. The diagram shows the interplay between REANA internal components and the role of the central database and message queue systems. The REANA cluster dispatches jobs on containerised clouds. Note the support for several workflow systems (CWL, Serial, Yadage). The workflow jobs use shared storage and pull necessary environment images from Docker registries.

```
$ minikube start --kubernetes-version="v1.11.2"
$ pip install reana-cluster
$ reana-cluster init
```

Figure 7. The deployment of a local REANA cluster is facilitated by the **reana-cluster** script.

analysis train test run and validation, ATLAS BSM search and recast analysis, CMS Higgs-to-four-lepton simplified analysis working on top of open data [5], and LHCb rare charm decay search analysis.

Figure 8 shows the ATLAS recast analysis example[10] using the official ATLAS Analysis Software Group stack[11]. The compute environment is captured by a Docker image encompassing all the required software packages at their correct versions. The analysis takes as input data the DxAOD ROOT file, a dataset ID, and a cross section value. The analysis

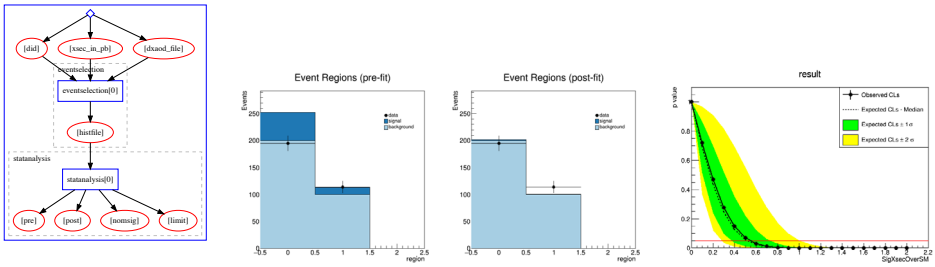

Figure 8. Example of an ATLAS recast analysis running the event selection and statistical analysis steps. This example workflow uses official ATLAS Software Group containerised environments. We see the example prefit, postfit and limit plot. The example can be found at [10].

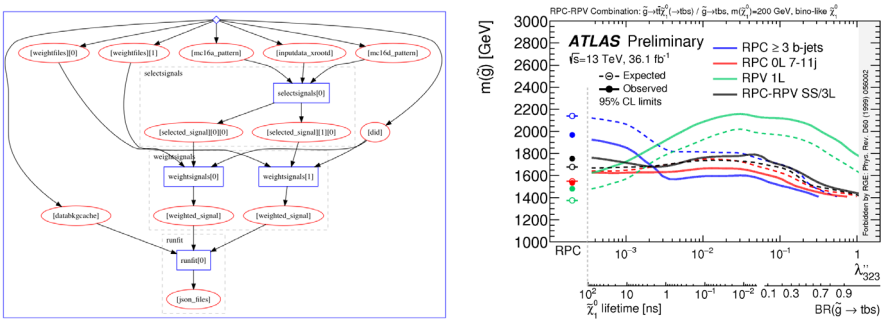

Figure 9. Left: A reinterpretation workflow for an analysis optimized for final states originating from the production of strongly interacting supersymmetric particles[12]. For reinterpretation, two separate Monte Carlo samples corresponding to different run-conditions of the LHC are processed and combined to yield an overall p-value signifying the viability of the signal in light of the observed data. Right: An earlier iteration of the same analyses was used to produce published reinterpretations[13].

consists of two computational steps, the event selection step and the statistical analysis which implements the limit setting for outputs produced by the event selection.

Figure 9 shows the ATLAS analysis reinterpretation workflow for a search for supersymmetry[12, 13].

These and other similar analysis examples are available on the REANA project web site[6], validating our approach and offering a reasonable integration test suite base for the REANA platform development and evolution.

6 Conclusions

REANA is a nascent reusable and reproducible research data analysis platform that allows researchers to run structured analysis pipelines on remote compute clouds. REANA supports diverse research communities in their diverse computational workflow practices by means of a multi-user, multi-workflow, multi-compute backend support. The architecture of the REANA platform and the applicability of the container-based approach for typical particle physics data analyses has been validated by several examples from ALICE, ATLAS, CMS and LHCb collaborations. REANA aims at facilitating future reuse of particle physics data

by observing and evolving the current analysis practices existing in the HEP community and by leveraging the progress made in related computational science disciplines (astronomy, life sciences) and with the general container technologies in the wider IT industry at large.

References

[1] M. Baker, "1,500 scientists lift the lid on reproducibility", Nature **533**, 452—454 (2016), doi:10.1038/533452a.

[2] E. H. B. M. Gronenschild, P. Habets, H. I. L. Jacobs, R. Mengelers, N. Rozendaal, J. van Os, M. Marcelis, "The Effects of FreeSurfer Version, Workstation Type, and Macintosh Operating System Version on Anatomical Volume and Cortical Thickness Measurements", PLOS One (2012), doi:10.1371/journal.pone.0038234.

[3] Z. Akopoff *et al*, "Status Report of the DPHEP Study Group: Towards a Global Effort for Sustainable Data Preservation in High Energy Physics" (2012), arXiv:1205.4667.

[4] M. D. Hildreth, A. Boehnlein, K. Cranmer, S. Dallmeier-Tiessen, R. Gardner, T. Hacker, L. Heinrich, I. Jimenez, M. Kane, D. S. Katz, T. Malik, C. Maltzahn, M. Neubauer, S. Neubert, J. Pivarski, E. Sexton-Kennedy, J. Shiers, T. Šimko, S. Smith, D. South, A. Verbytskyi, G. Watts, J. Wozniak, "HEP Software Foundation Community White Paper Working Group – Data and Software Preservation to Enable Reuse" (2018), arXiv:1810.01191.

[5] X. Chen, S. Dallmeier-Tiessen, R. Dasler, S. Feger, P. Fokianos, J. B. Gonzalez, H. Hirvonsalo, D. Kousidis, A. Lavasa, S. Mele, D. Rodriguez Rodriguez, T. Šimko, T. Smith, A. Trisovic, A. Trzcinska, I. Tsanaktsidis, M. Zimmermann, K. Cranmer, L. Heinrich, G. Watts, M. Hildreth, L. Lloret Iglesias, K. Lassila-Perini, S. Neubert, "Open is not enough", Nature Physics (2018), doi:10.1038/s41567-018-0342-2.

[6] REANA reusable and reproducible analysis platform, http://www.reana.io/.

[7] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, A. P. Barros, "Workflow Patterns", Distributed and Parallel Databases **14**, 5–51 (2003), doi:10.1023/A:1022883727209.

[8] P. Amstutz, M. R. Crusoe, N. Tijanić (editors), B. Chapman, J. Chilton, M. Heuer, A. Kartashov, D. Leehr, H. Ménager, M. Nedeljkovich, M. Scales, S. Soiland-Reyes, L. Stojanovic, "Common Workflow Language, v1.0" Specification, Common Workflow Language working group (2016), doi:10.6084/m9.figshare.3115156.v2.

[9] K. Cranmer, L. Heinrich, "Yadage and Packtivity — analysis preservation using parametrized workflows" (2017), arXiv:1706.01878.

[10] REANA example, https://github.com/reanahub/reana-demo-atlas-recast.

[11] R. Seuster, M. Elsing, G. A. Stewart and V. Tsulaia on behalf of the ATLAS Collaboration, "Status and Future Evolution of the ATLAS Offline Software", J. Phys. Conf. Ser. **664** (2015) 072044. doi:10.1088/1742-6596/664/7/072044.

[12] ATLAS collaboration, "Search for supersymmetry in final states with missing transverse momentum and multiple b-jets in proton-proton collisions at $\sqrt{s}$ = 13 TeV with the ATLAS detector" (2018), ATLAS-CONF-2018-041, https://cds.cern.ch/record/2632347.

[13] ATLAS collaboration, "Reinterpretation of searches for supersymmetry in models with variable R-parity-violating coupling strength and long-lived R-hadrons" (2018), ATLAS-CONF-2018-003, http://cdsweb.cern.ch/record/2308391.