PROCEEDINGS OF SCIENCE

Data and Software Preservation through Containerisation in KM3NeT

Tamás Gál for the KM3NeT Collaboration*
Erlangen Centre for Astroparticle Physics, Erlangen (Germany)
E-mail: tgal@km3net.de

In this contribution we discuss of how we implemented procedures for maintaining and developing software and preserving data processing pipelines in KM3NeT using containerisation technologies and continuous integration.

The New Era of Multi-Messenger Astrophysics - Asterics2019
25 - 29 March, 2019
Groningen, The Netherlands

*Speaker.

PoS(Asterics2019)027

1. Introduction

One crucial aspect of scientific research is interoperability and reproducibility. Methods, procedures, software and even hardware can be documented easily but a big challenge is to preserve software and data in a way that it remains re-usable on current and next generation computing systems. The KM3NeT collaboration [1] has defined and developed workflows based on state of the art containerisation tools and techniques to achieve these goals while minimising the compatibility requirements. The web-based open source Git-repository manager *GitLab* [2] provides all the ingredients for future-proof software development including essential features like version management, issue tracking, discussion platforms and continuous integration. The continuous integration is based on Docker [3], which performs operating-system-level virtualisation – also known as containerisation – and is used to declare the environment needed to run a specific software. Each software project is developed, compiled and tested in an isolated and independent container and later also deployed to a target production system as a single bundled image including all requirements. These so-called Docker images serve as a starting point for Singularity [4] images, which is another containerisation solution specifically designed to run on High Performance Computing (HPC) clusters and thus is well suited for fully reproducible analysis chains running on large and sometimes heterogeneous computer clusters and grids. Singularity images are already successfully used in production pipelines of the ANTARES experiment [5] and have proven to be a convenient solution with minimal setup, maintenance and compatibility requirements. In order to coordinate and describe such pipelines, a scheme using the Common Workflow Language has also been defined. The strategy behind these methods and their general applicability are presented in this contribution.

2. Containerisation

Containerisation is the encapsulation of a system environment which runs directly on the kernel of the host system and has limited access to its resources. Depending on the implementation, the environment is either almost fully isolated or integrates itself to the host environment. It was primarily designed for micro-service virtualisation and allows to run multiple instances of containers on a single machine without the performance drawbacks of virtual machines. Isolation, flexible resource allocations and prioritisation are the key features of enterprise solution containerisation systems where lots of idling containers are common. For science on the other hand the requirements are quite different.

2.1 Requirements for Science

- Maximum performance needed.
- 100% reproducibility.
- Ability to interact with containers with standard user privileges.
- One container to utilise all available host sources.
- No need for resource isolation.

2.2 Docker

Docker is a containerisation software designed for micro services, development and continuous integration. It encapsulates the software in images, which consist of reusable "layers". This layered design reduces the amount of data to be stored by sharing them between images, i.e. several images can for example be based on a specific base operating system which only needs to be stored once. Additional applications, libraries or custom configurations are then added as layers. Official public repositories like Docker Hub or privately hosted Docker registries allow easy sharing of images and layers. The downside of Docker containers however is that they may not be fully reproducible due to complex layer dependencies which may change over time. The main benefit is very high level of encapsulation – very similar to Virtual Machines but with native performance. Docker containers however are designed to run isolated from the host environment and are not meant and allowed to run on HPC (e.g. privileged permissions are required).

2.3 Singularity

Singularity is similar to Docker, but created for and by HPC engineers, scientists and Linux developers. It does not require a root-owned base container daemon and runs under the initial user's privileges. Additionally it prevents privilege escalation via the kernel features. `/home`, `/dev`, `/sys` and `/proc` are mounted from the host machine and a direct usage of host resources (like network, file systems, devices etc.) is possible. These features make it safe for use in HPC environments. It also supports Message Passing Interface (MPI) and can natively use GPUs by default. Singularity is compatible to other container solutions – especially Docker, which is often used in development pipelines. The containers are single-file based images and can be shared by simply copying them to the target system, which only needs Singularity installed to run it. The containerisation architecture of Singularity does not include as many isolation tweaks and thus it's a bit more performant than Docker, which was designed around micro-service process isolation.

3. Software Development Workflow

Through developments within the ASTERICS project, we have set up a system of container-based software development, deployment and preservation - including data processing and data analysis. This is based on a self-hosted GitLab instance where everyone has full access to file bugs, feature requests and to discuss or follow other project relevant topics. Milestones and KanBan-like boards – as shown in Figure 1 – help to communicate the development status of each project and to attract collaborators. Continuous integration pipelines like in Figure 2 are used to test and deploy our software and the corresponding documentation. The same technology is used to build and publish container images after each software release.

3.1 Integration and Deployment Pipelines

The standards for developing, maintaining and archiving official KM3NeT software have been defined. Every changeset occured during the software development triggers a continuous integration pipeline, as demonstrated in Figure 3. The pipeline is executed in isolated Docker containers and ensures an independent and reproducible environment. A failure at any stage will be reported

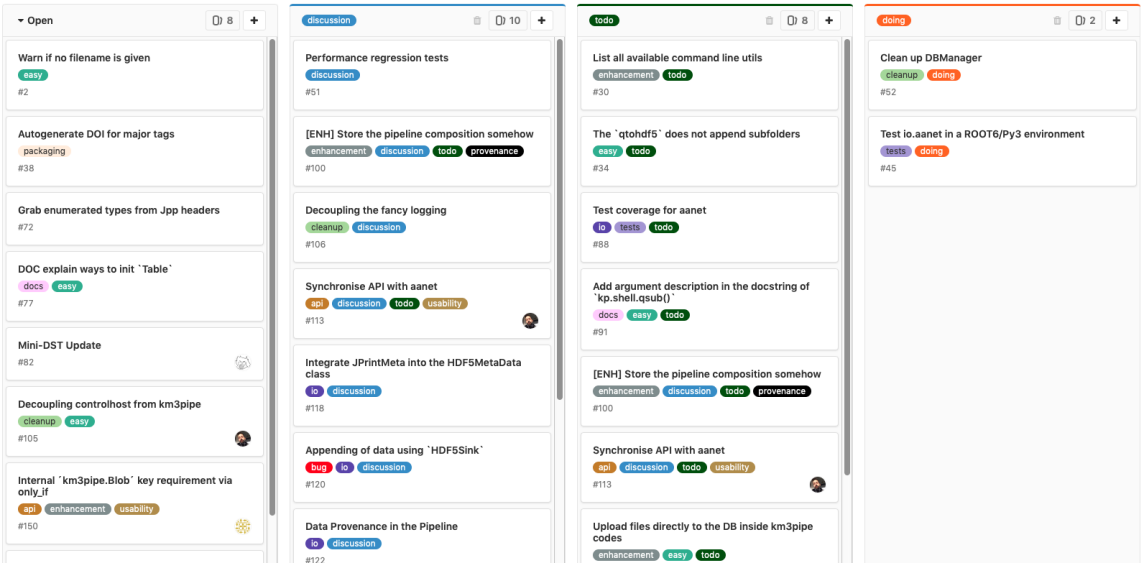

Figure 1: KanBan board of a KM3NeT software project.

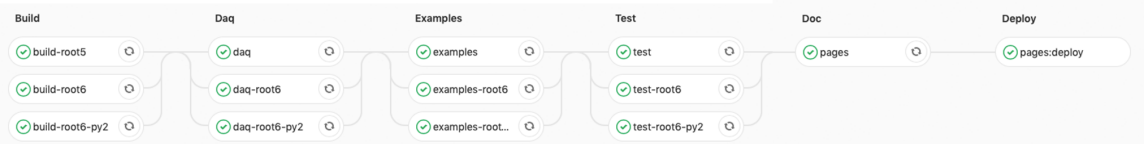

Figure 2: A typical continuous integration pipeline of a KM3NeT software project.

to the project developers and maintainers who participated in the changeset. Additional stages are executed when releasing a software version to create dedicated Docker and Singularity images to be used for further developments and analysis chains respectively The software used to orchestrate the version control, related discussions, running of pipelines and releasing images for production is provided by *GitLab*

3.2 Distribution of Software

Docker images of commonly used software are regularly pushed to a self-hosted docker registry to be used in production or reused in other development pipelines which depend on them. Singularity images are published through an SFTP server.

4. Analysis Chains

Modern cloud (grid) computing consists of heterogeneous operating systems which are constantly maintained and updated, which raises a big challenge to organise and install the required software and its dependencies on them. Singularity images provide a way to distribute an all-in-one package which stays persistent and reproducible regardless of the changes of the host system. This technology is already successfully utilised in ANTARES analysis pipelines. For KM3NeT a similar approach has already been implemented and tested, including the full integration of the

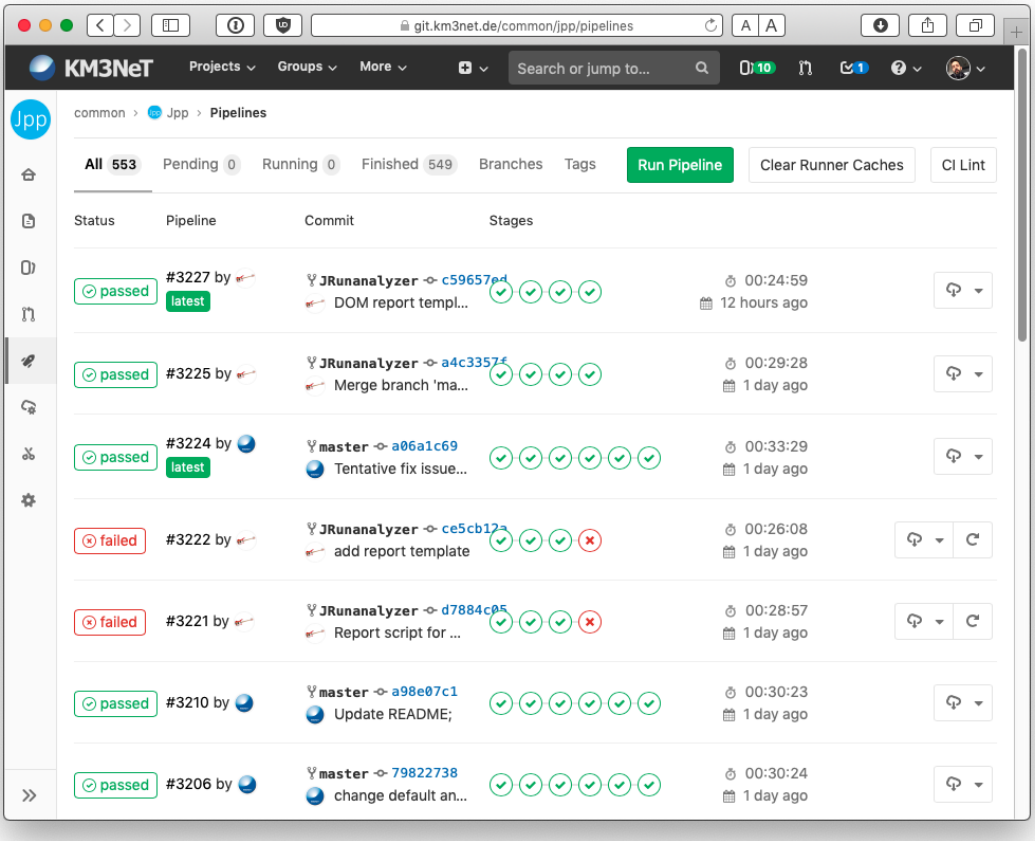

Figure 3: Pipelines triggered on each update of the software project.

software development pipeline. To describe the analysis chains and make them easily portable, the Common Workflow Language [6] (CWL) is currently being evaluated, which is a language to write analysis workflows and tools in a portable and scalable way accross many software and hardware environments like workstations, clusters, cloud or HPC environments. CWL combined with Singularity is a powerful tool to minimise efforts in maintenance, compatibility and reproducibility in heterogeneous computational environments.

References

[1] S Adrián-Martínez et al. Letter of intent for KM3net 2.0. *Journal of Physics G: Nuclear and Particle Physics*, 43(8):084001, jun 2016.

[2] Gitlab. `https://www.gitlab.com`. Accessed: 2019-06-15.

[3] Docker. `https://www.docker.com`. Accessed: 2019-06-15.

[4] Singularity. `https://www.sylabs.io`. Accessed: 2019-06-15.

[5] M. Ageron et al. ANTARES: the first undersea neutrino telescope. *Nucl. Instrum. Meth.*, A656:11–38, 2011.

[6] Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, Matt Scales, Stian Soiland-Reyes, and Luka Stojanovic. Common Workflow Language, v1.0. 7 2016.