https://doi.org/10.1051/epjconf/202429504042

The ALICE Grid Workflow for LHC Run 3

Maxim Storetvedt[1], Latchezar Betev[1], Håvard Helstrup[2], Kristin Fanebust Hetland[2], and Bjarte Kileng[2]

[1]CERN, Geneva, Switzerland
e-mail: mstoretv@cern.ch

[2]Faculty of Engineering and Science, Western Norway University of Applied Sciences, Bergen, Norway

Abstract. In preparation for LHC Run 3 and 4 the ALICE Collaboration has moved to a new Grid middleware, JAliEn, and workflow management system. The migration was dictated by the substantially higher requirements on the Grid infrastructure in terms of payload complexity, increased number of jobs and managed data volume, all of which required a complete rewrite of the middleware using modern software languages and technologies. Through containerisation, self-contained binaries, managed by the JAliEn middleware, we provide a uniform execution environment across sites and various architectures, including accelerators. The model and implementation have proven their scalability and can be easily deployed across sites with minimal intervention.

This contribution outlines the architecture of the new Grid workflow as deployed in production and the workflow process. Specifically shown is how core components are moved and bootstrapped through CVMFS, enabling the middleware to run anywhere fully independent of the host system. Furthermore, we examine how new middleware releases, containers and their runtimes are centrally maintained and easily deployed across the Grid, also by the means of a common build system.

1 Introduction

The ALICE Collaboration has completed its migration to a new Grid workflow [1]. Centred around the new JAliEn (Java ALICE Environment) middleware, it comes with a reworked deployment stack and a complete rewrite of core components [2]. Through the use of modern features and software languages, this new workflow is aimed at better handling the substantially higher requirements of LHC Run 3 and Run 4, accommodating complex multicore payloads, an increased number of jobs and managed data volume – and enables a uniform execution environment for Grid jobs, across sites, configurations and architectures.

The above features have been achieved in part by incorporating containers, the existing CVMFS distributed file-system and self-contained binaries directly as part of the Grid middleware and its underlying infrastructure [3]. This has in turn created a deployment stack that is both independent of host configurations, and may also be quickly rolled out across Grid sites in a more automated manner through a centralised build system. With central recipes used for package deployments, bootstrap environments and payload environments, there is also less need for maintenance interventions by Grid site admins. Furthermore, it has facilitated the adoption of new features and hardware, such as better resource management and the inclusion of accelerators like GPUs [4].

Given the larger changes to the overall ALICE Grid, this contribution aims to provide an overview of key changes and architectural highlights, detailing how a new Grid workflow comes together for Run 3 and beyond. The paper is organised as follows: it starts with a general overview of central components of the workflow, focusing on key changes and features. It will thereafter proceed to examine how this is achieved across the ALICE Grid, before moving on to the main mechanisms of deployment, and lastly the maintenance process. A number of conclusions are provided at the end.

2 A new Grid Workflow

A Grid workflow entails the procedure of handling executable payloads, i.e. jobs. This includes not only steps directly involved in executing a binary, but also central processes such as data transfers, storage and package distribution. An overview of the new workflow as deployed within ALICE is depicted in Figure 1.

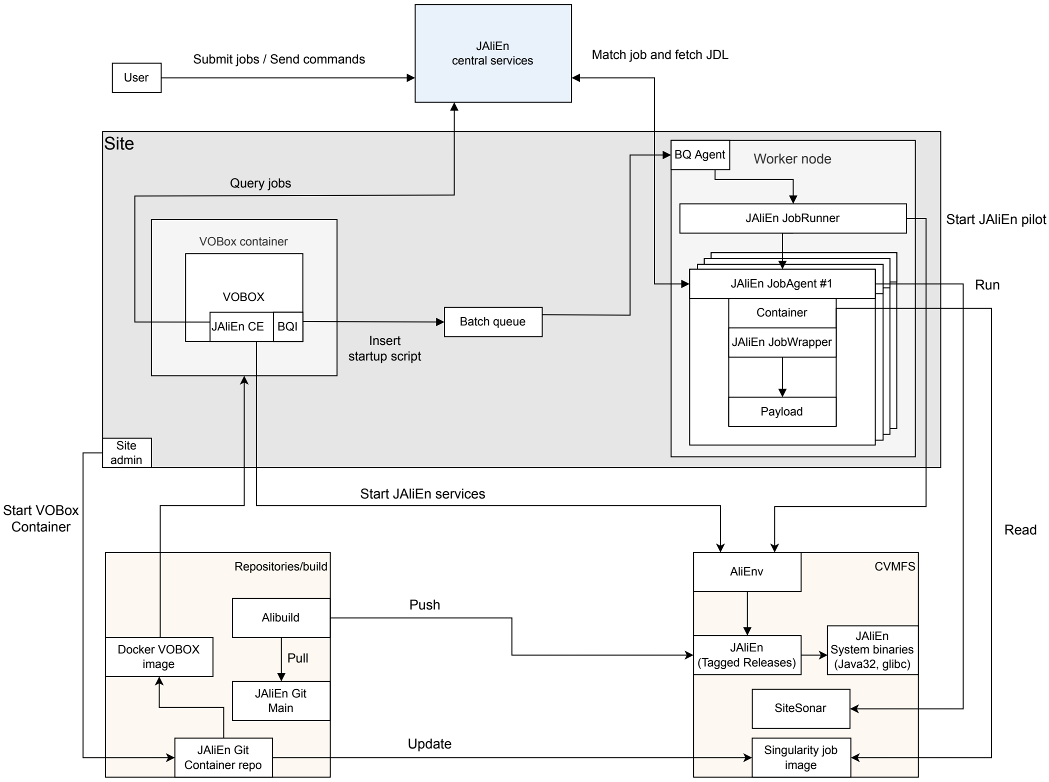

Figure 1. Simplified overview of the new ALICE Grid workflow. Core components are either containerised, or deployed and bootstrapped directly through CVMFS – tied to central recipes and builds.

A key characteristic of the new workflow is the containerisation of Grid site components and services – for both the front-end site services and the corresponding job pilots on each worker node (WN). These containers provide a common environment across sites, each tied to a central recipe that is automatically built and distributed through CVMFS, and separate from that of the executing host. This separation extends to the core components necessary for providing the common container, where Java runtime, libraries and configurations are likewise directly taken and bootstrapped from CVMFS. Update handling and versioning are now automated, and can be both managed and triggered centrally (Section 4).

2.1 ALICE VO-Boxes

ALICE maintains the VO-Box model for site services, i.e. requiring a dedicated front-end node to serve as an entrypoint for the site and its computing resources. The VO-Box also serves as the host for the JAliEn site component, "JAliEn CE", which is used to interface with the local computing resources.

To simplify the deployment process, the VO-Box now is available as a preconfigured container, needing only valid credentials to be supplied [5]. All of its binaries, however, are taken directly from CVMFS, allowing them to be both usable by non-container VO-Boxes, as well as allowing updates to be decoupled from the base container image.

Having all binaries taken from CVMFS applies not only to VO-Boxes, but to all JAliEn site executables – including those used on the WNs. Upon detecting a queued task compatible with its site, the JAliEn CE creates a startup script used to launch a new job pilot, and submits that script to the computing resources of the site. All executable pointers within this script are directed towards binaries in CVMFS, meaning there are prerequisites neither for the environment nor for the packages provided by each WN. As a consequence, the only requirements for achieving a working ALICE Grid site are to deploy the ALICE VO-Box, possibly as a container, and provide it with a valid host certificate and computing resources to submit to. The startup script takes care of the rest.

2.2 Job Pilots and WNs

Environment, paths and credentials used to launch job pilots are provided by startup scripts, in part achieved through calls to *alienv* (see Section 4), which keeps track of versions, binaries and required libraries. These scripts are in turn used to launch and initialise job pilots directly from CVMFS, corresponding to the current version of JAliEn configured for the site.

Each job pilot is composed of three separate components, subdividing roles and responsibilities:

- The **JobRunner** – Resource handler [6].
- The **JobAgent** – Job matcher and monitoring handler [7].
- The **JobWrapper** – Payload executor [7].

The hierarchy between these components can be seen in Figure 2.

Once a startup script has prepared a working environment, it will proceed to launch a *JobRunner*, which tracks and coordinates the system resources provided by the job slot. If there are sufficient available resources, the JobRunner will proceed to launch a *JobAgent*, which will match and pull a compatible queued task from the central cervices. A job may request multiple cores per slot, which will in turn be given to the payload. While not all the cores and other resources assigned to a slot are taken, the JobRunner may proceed to launch additional JobAgents to fully utilise the given slot. Once a job is matched, a JobAgent will proceed to set up monitoring for the new job, and thereafter start a *JobWrapper*. This component runs with separate credentials in its own Java Virtual Machine (JVM), independent of the agent/runner, and is responsible for fetching required files and executing the payload,

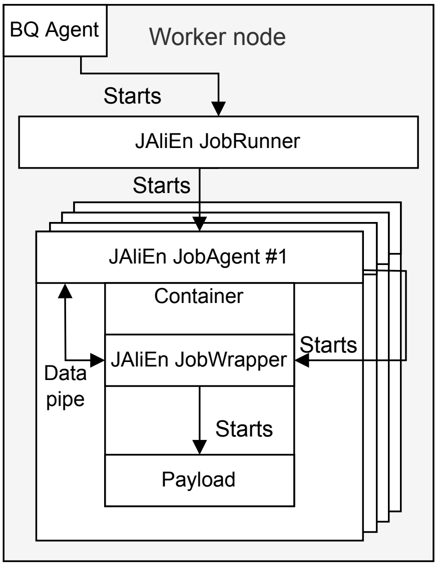

Figure 2. Through the submitted startup script, a batch queue (BQ) agent on a WN will start a Job Runner, which in turn may launch multiple JobAgents depending on the resources available. Each JobAgent provides a containerised environment to a JobWrapper, which handles the payload execution, while maintaining monitoring and a pipe for data transfer.

before ultimately uploading the results. Being an independent process in its own JVM, the JobWrapper is automatically containerised by the JobAgent, providing both isolation and a common execution environment for the job payload across sites.

2.3 Jobs

By default, all ALICE Grid jobs are wrapped in a common container by the JobAgent. As for JAliEn, binaries for the container runtime may be taken taken directly from CVMFS, albeit local runtimes are also supported. Upon matching a job, the JobAgent performs a compatibility check for potential means of providing a payload container, and will examine both CVMFS and host binaries.

In addition to multicore, ALICE jobs also support the use of accelerators, viz. GPUs. For this purpose, the JobAgent performs an additional check on its selected container, examining if host libraries required by a GPU are compatible with it. If there are no conflicts, appropriate mounts and flags are forwarded to the runtime providing the container. At the time of writing, this is only supported for Nvidia and AMD GPUs on the Apptainer[1] runtime.

The default image for job containers is provided as an extracted directory tree in CVMFS, based on CentOS 7[2]. It is tied to a recipe available on Git, where additional packages may be added through pull requests[3]. The specific container image may also be set per *site* if needed, which will apply to all jobs executed on that particular site. Two additional images are provided for this purpose:

[1] https://apptainer.org/

[2] /cvmfs/alice.cern.ch/containers/fs/singularity/centos-latest

[3] https://gitlab.cern.ch/jalien/jalien-jobcontainer

- AlmaLinux 8.7 – Used for newer payloads (no ROOT5 support) and jobs using a GPU (required by drivers).
- AlmaLinux 9.1 – Testing container (currently not used in production).

The container image can neither be set nor customised by users, nor through the jobs submitted by them.

3 Compatibility

The discussed Grid workflow comes with a new set of requirements, in particular for sites and their hosts and WNs – in regard to the OS, kernel, configurations and available packages. In order to work around incompatibilities and to better understand what configurations were used across the Grid, a separate project was launched to map the configurations used by sites and their WNs, which eventually became SiteSonar [8].

While the information provided by the new monitoring framework allowed for working around common issues, e.g. enabling unprivileged bind-mounts within job containers, it also made it evident that most WNs would nevertheless require larger changes or interventions from their respective site admins. Consequently, a working solution was made to move as many components as possible to CVMFS, including core libraries such as *glibc*. These were modified, when needed, to accept paths as organised within the ALICE repository in CVMFS – in turn creating an environment similar to that of a container, without the process wrapping, providing everything needed to initialise a job pilot. Once a pilot has started, a full container may be launched as before. This allows all site components to run on hosts across the Grid, largely independent of their configuration – provided they run a recent[4] Linux kernel.

4 Distribution and maintenance

JAliEn releases are handled through *Alienv*, a *modulecmd*[5] wrapper that tracks dependency trees and CVMFS paths as required by each version of an executable. It is tightly interconnected with *Alibuild*[6], an automated build system used to both build and publish new releases – when a new JAliEn release is tagged in the source repository, it will be picked up, built and pushed to CVMFS by Alibuild. A corresponding Alienv tag for the release is also added, allowing that specific version to be quickly selected and loaded from CVMFS.

4.1 Site updates

Alienv is used for starting various components across the Grid, setting paths and other environment variables according to corresponding locations in CVMFS. By default, Alienv will select the latest tag available at startup time. Newer versions released after startup will be considered only when a restart is done.

The above process also applies to the JAliEn CE component, which may quickly be set to a specific version, or updated to the latest, at startup or through a restart. This will also affect the WNs for that particular site, as the Alienv call with its version tag is propagated to the startup scripts. Consequently, solely updating the JAliEn CE will also update all site WNs, as the startup script will now point to a job pilot corresponding to the JAliEn version used on the VO-Box. As pilots become gradually replaced, the full site is eventually switched with no further action needed.

[4] Linux 3.10+, with RedHat patches.
[5] https://modules.readthedocs.io/
[6] https://alisw.github.io/alibuild/

4.2 Maintainability

As seen in the previous section, simply restarting a JAliEn component allows it to be switched to the latest version as provided in CVMFS. While this process can be initiated by a local site admin, it can also be triggered centrally – allowing JAliEn updates to be handled automatically, independently of the site. As with JAliEn, the process of updating and maintaining packages, and other parts of the environment, have likewise shifted to be more centrally managed. Both the job environment and (optionally) the VO-Box are now provided by containers, tied to central recipes and automatically distributed.

The default behaviour is to direct a newly started service to the latest production release, though sites may optionally be subscribed to different release "channels" as needed:

- *New* – The latest published release in CVMFS
- *Pro* – The latest release considered stable for general (production) use
- *Custom* – A specific release/version may also be set

New releases are published regularly (typically every ~2 weeks), with the symbolic link corresponding to each release channel updated accordingly. The frequency of new releases, in addition to the inclusion of new frameworks and extensions, such as SiteSonar, is a good indicator that the JAliEn framework is itself very maintainable – keeping up with the needs of the ALICE Collaboration.

5 Conclusions

ALICE has moved to a new Grid middleware and workflow system based around JAliEn, aimed at overcoming the computing challenges of LHC Run 3 and beyond. It benefits from the emergence of new tools and developments that have come to light since the release of the original ALICE middleware, providing for new features and conveniences.

Paired with a streamlined codebase, and tied to a convenient build and publishing mechanism, the above has enabled the system to be quickly rolled out across the ALICE Grid, fully replacing legacy implementations. It is now used across 78 distinct VO-Boxes, deployed at 54 computing sites and managing over 4000 WNs – providing ~200,000 CPU cores. At the time of writing, this has in turn enabled over 115M jobs to be successfully executed since the beginning of Feb. 2023, when the last legacy site was transitioned, with over 580,000 jobs completed each day on average.

The vast majority of all jobs now run in a common, containerised environment across Grid sites, ensuring compatibility and ease of setup. Thanks to tighter integration with CVMFS, more independence is obtained from each underlying host system, allowing for more automated maintenance and updates. This has also enabled jobs to quickly benefit from new developments, such as multicore slots and GPU accelerators. All this has resulted in a modernised Grid workflow, capable of serving the ALICE Collaboration in the years to come.

References

[1] ALICE Collaboration, *The ALICE Experiment at the CERN LHC*, Journal of Instrumentation Vol. 3, Institute of Physics Publishing / SISSA, 2008. DOI: 10.1088/1748-0221/3/08/S08002

[2] A.G Grigoras, C. Grigoras, M.M Pedreira, P. Saiz, S. Schreiner, *JAliEn A new interface between the AliEn jobs and the central services*, Journal of Physics: Conference Series 523. DOI:10.1088/1742-6596/523/1/012010

[3] Jakob Blomer, Predrag Buncic and René Meusel, *The CernVM File System – A virtual appliance for LHC applications*, CERN 2013. [Online]: https://jblomer.web.cern.ch/cvmfstech-2.1-0.pdf

[4] David Rohr, *Usage of GPUs in ALICE Online and Offline processing during LHC Run 3*, EPJ Web Conf. 251 04026 (2021). DOI: 10.1051/epjconf/202125104026

[5] Maxim Storetvedt, Maarten Litmaath, Latchezar Betev, Håvard Helstrup, Kristin Fanebust Hetland, Bjarte Kileng, *Grid services in a box: container management in ALICE*, EPJ Web Conf. 214 07018 (2019). DOI: 10.1051/epjconf/201921407018

[6] Sergiu Weisz and Marta Bertran Ferrer, *Adding multi-core support to the ALICE Grid Middleware*, J. Phys.: Conf. Ser. 2438 012009 (2023). DOI: 10.1088/1742-6596/2438/1/012009

[7] Maxim Storetvedt, Latchezar Betev, Håvard Helstrup, Kristin Fanebust Hetland, Bjarte Kileng, *Running ALICE Grid Jobs in Containers – A new approach to job execution for the next generation ALICE Grid framework*, EPJ Web Conf. 245 07052 (2020). DOI: 10.1051/epjconf/202024507052

[8] K. Wijethunga, M. Storetvedt, C. Grigoras, L. Betev, M. Litmaath, G. Amarasinghe, I. Perera, *Site Sonar – A Flexible and Extensible Infrastructure Monitoring Tool for ALICE Computing Grid*, these proceedings (2024).