Electronic Communications of the EASST
Volume 85 Year 2025

deRSE25 - Selected Contributions of the 5th Conference for Research Software Engineering in Germany

Edited by: René Caspart, Florian Goth, Oliver Karras, Jan Linxweiler, Florian Thiery, Joachim Wuttke

Software Infrastructure for the fully containerized Cluster at GSI/FAIR

Dmytro Kresan, Mohammad Al-Turany, Denis Bertini, Matteo Dessalvi, Raffaele Grosso, Dennis Klein, Matthias Kretz, Victor Penso, Christian Tacke

DOI: 10.14279/eceasst.v85.2704

Electronic Communications of the EASST (https://eceasst.org).
Published by **Berlin Universities Publishing**
(https://www.berlin-universities-publishing.de/)

The Software Infrastructure Development for the Fully Containerized Cluster at GSI/FAIR

Dmytro Kresan, Mohammad Al-Turany, Denis Bertini, Matteo Dessalvi, Raffaele Grosso, Dennis Klein, Matthias Kretz, Victor Penso and Christian Tacke

GSI Helmholtz Centre for Heavy Ion Research GmbH, Darmstadt, Germany
https://www.gsi.de/
d.kresan@gsi.de

Abstract: The scientific research at the Facility for Antiproton and Ion Research (FAIR) spans a wide range of fields including nuclear physics, atomic physics, and heavy ion physics. Workflows for simulations and data analysis in FAIR experiments range from High Throughput Computing (HTC) to Message Passing Interface (MPI) calculations and traditional batch processing. Operating a shared computing cluster that is scalable enough to meet the diverse needs of such a heterogeneous user community presents a significant challenge.
We briefly summarize the principles of a fully containerized approach used in the GSI/FAIR computing cluster, which has been successfully operating for five years. This approach involves additional software infrastructure to support a reproducible containerized environment and enable reliable testing. This paper presents the fundamental building blocks of this software infrastructure in detail.

Keywords: computing cluster, Linux containers, continuous integration

1 Introduction

The usage of Linux containers on High Performance Computing (HPC) infrastructures of the scientific community is becoming more popular. This technology, complemented by additional user-specific software, can significantly improve the usability of an HPC cluster. It also eases maintenance by enabling customized, lightweight runtime environments that are decoupled from the underlying minimal operating system. By isolating dependencies and software versions within containers, researchers can avoid system-wide conflicts and ensure consistent performance across all working nodes of a computing cluster. Container technologies like Apptainer or Podman support HPC-specific optimizations, allowing close-to-native performance while portability across clusters gets drastically increased.
Experiments at the Facility for Antiproton and Ion Research (FAIR) in Darmstadt, Germany, focus on fundamental research in the areas of atomic, nuclear and astrophysics, and heavy-ion collisions. The detector setups vary in size and complexity, along with the amount of data to be analyzed. Depending on their size, experiments are organized and conducted by different groups of scientists, ranging from large international collaborations to small local teams of a few physicists.
The computational requirements of such users differ greatly. It is therefore very challenging to

set up and operate a single shared cluster that offers sufficient scalability and flexibility for such a heterogeneous community. A fully containerized approach, explained in Section 2, was developed and applied to achieve this goal. In this paper we will focus on the Virtual Application Environment (VAE), a term we have defined specifically for the GSI / FAIR computing cluster which describes standardized solution for workflow with containers. It is centrally maintained and covers approximately 80 % of use cases at GSI / FAIR. The remaining cases are handled by permitting users to run their own containers. In the next section, we describe the essential building blocks of the software that support the containerized environment.

2 Software Infrastructure for VAE

By fully containerized, we refer to an HPC operation model in which each user application is executed inside a container. In contrast, the host system contains only a minimal set of packages—on our GSI cluster this includes hardware drivers, Slurm [YJG03], and Apptainer. This minimalist setup simplifies cluster maintenance while giving users the flexibility to install any required software within their dedicated container images. For users with limited HPC experience, preparing and building container images can be challenging. To improve usability and facilitate access to the required software environment, we provide a standardized container image together with a dedicated software stack distributed via CVMFS (CERN Virtual Machine File System) [BBC11]. This approach ensures consistency, portability, and minimal local configuration across different computing sites. Dedicated submit nodes are configured to automatically initiate a login session within the containerized environment upon user access. This seamless integration removes the need for manual container management, enabling users to immediately work within a pre-configured and optimized setup tailored to their workflows. The environment provides access to a comprehensive HPC and High Energy Physics (HEP) software stack. Moreover, jobs submitted from within such container are, by default, executed in the same environment: each job runs in a new container built from the same image and inherits the environment variables from the submitting shell. This setup represents the concept of the VAE. Apptainer was chosen as the container engine for the entire cluster. Designed specifically for HPC and scientific computing in multi-user environments, it provides native MPI support. Moreover, Apptainer containers run with minimal privileges, thereby enhancing security in HPC systems.

We define four key pillars of the software infrastructure required to operate a fully containerized cluster: automatic login into a containerized environment when accessing the cluster; launching containers on worker nodes upon job submission; building and testing container images; and building and deploying a software stack that supports user-specific software. The following subsections provide further details on each of these pillars.

2.1 Login into Container

To access cluster resources, users log in to a dedicated set of nodes known as submit nodes. These nodes serve as the command-line interface to the cluster's resource management system, Slurm, and provide access to all storage systems connected to the compute nodes. User logins to the submit nodes are managed by the OpenSSH sshd daemon [The24], which is configured to

launch a Linux container for each session. This creates a VAE for every user login. As a result, logging into a container is seamless for users and is an integrated feature of the submit node infrastructure. Below are the example commands, showing how to access different environments on the cluster:

```
# Access to the host environment on native hardware
ssh virgo.hpc.gsi.de

# Access to the VAE version 25
ssh vae25.hpc.gsi.de

# Access to custom container image
export APPTAINER_CONTAINER=/absolute/path/container.sif
ssh -o SendEnv=APPTAINER_CONTAINER virgo.hpc.gsi.de
```

From a technical standpoint, this functionality is implemented by assigning a custom shell script [Pen25] to the sshd daemon, which is executed upon user login. Once access is granted, the script launches an interactive shell inside a VAE instantiated via the Apptainer container runtime. Users can optionally define an environment variable to specify the desired version of the VAE container, or administrators may configure a text-based menu to allow users to select from a set of available environments at login. The specific container hosting a user session is then registered with the workload management system, ensuring that any applications submitted to the cluster execute within the same containerized environment.
The script executed by the SSH daemon is called in place of the default shell by means of the SSH *ForceCommand* option, allowing administrators to define how a container is launched on behalf of a user. A crucial aspect of this configuration is the bind-mounting of all necessary resources into the container, including access to Slurm services, which enables users to interact with the resource management system. The script also manages input environment variables that users provide during remote login. Users requiring customization can build their own containers and utilize the aforementioned facilities to ensure that logging into the submit nodes launches their private container. This process is as simple as copying a container to the cluster storage and setting the appropriate path via an environment variable before login, as shown in the last command line example above.
Extending the script's functionality is straightforward; for instance, administrators can modify it to support additional container runtimes. This would enable integration with alternative methods for launching containers on Slurm, such as the OCI-based container support developed by SchedMD [Sch]. However, it is crucial to thoroughly test any modifications made to the script executed by the SSH daemon, as a faulty implementation could potentially prevent even root access to the machine.

2.2 Plugin for Seamless Job Submission

Jobs are submitted to the cluster via Slurm. To ensure that jobs dispatched to worker nodes are transparently executed within a VAE, we developed a Slurm plugin called slurm-singularity-exec [KPK+24]. On the submit node, this plugin supports optional command-line arguments

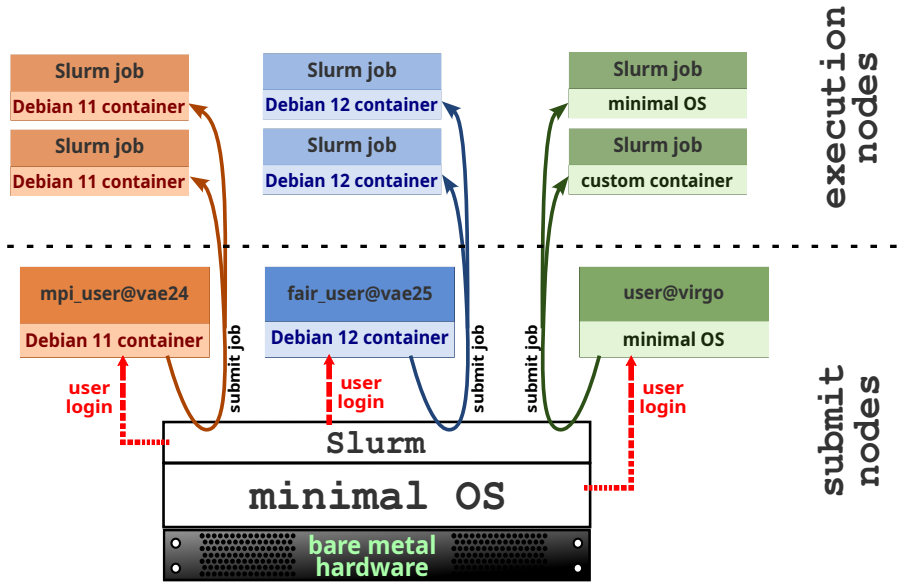

Figure 1: Schematic representation of the job submission process performed by several users each logged in in a different container or in the base operating system.

such as `--singularity-container` (to specify an alternative container image) and `--singularity-bind` (to define bind mounts). These options are propagated to the worker nodes. By default, users do not need to take any additional action, the environment automatically ensures that their job runs in the same container environment which they are currently using. This feature is schematically illustrated in Figure 1.
The user experience remains seamless: standard Slurm tools such as `srun`, `sbatch`, and `salloc` function as expected by the users, while the plugin transparently ensures correct container execution. This lowers the barrier to use container-based HPC environments and has proven to be both robust and stable.

While many users overlook the fact that a container encapsulates the application environment, some leverage this capability to manage multiple versions of their application runtime stack. Testing and development environments can be achieved simply by modifying the command-line options provided to Slurm. When combined with a container registry and a continuous integration system, the Slurm cluster at GSI can be utilized to fully automate the testing of containerized environments.

2.3 Build and Test of Containers

The security and patch updates of the operating system and packages are also of high importance in a containerized environment. Therefore, regular builds must be performed. Builds of images for the VAE are implemented with dependency tracking based on Makefiles. The process is managed via a GitLab workflow, which is triggered by a commit to a dedicated branch of the Git repository. We use a two-branch deployment scheme: the main branch for the production system

Figure 2: Data throughput distribution across the Object Storage Targets (OSTs) of the Lustre test system. Such a uniform distribution of I/O throughput over the storage elements indirectly indicates that no processes timed out and that the test completed successfully.

and a secondary branch for development deployments.

After a successful container image build, two stages of testing are performed. In the first stage, job submission and MPI initialization are tested. To perform this, we compile a test program consisting only of starting of MPI and submit the resulting binary as a job on the cluster. The test is successful if during compilation, job submission and binary execution no errors where thrown. The second stage involves a parallel I/O benchmark based on OpenMPI [GFB+04]. Approximately 100 MPI processes are launched on the cluster, performing asynchronous writes to the distributed storage while measuring the time required for read and write operations. The test is considered successful if none of the MPI ranks time out during the I/O process. An example of the I/O throughput distribution across Lustre [Bra19] test system Object Storage Targets (OST) is shown in Figure 2.

If both testing stages are successful, the container image is labeled with a timestamp and deployed to a dedicated CVMFS repository. Once the publishing process is complete, the VAE is enabled across the entire cluster. As part of the CVMFS publishing process, obsolete container images older than 14 days are removed.

2.4 Manage Software Stack

The wide spectrum of research topics across various fields of physics at GSI and FAIR necessitates the installation of a diverse range of software tools on the computing cluster. We use the Spack [GLC+15] package manager to build, install, and maintain the software stack associated with a VAE. Spack was originally created for multi-user scientific environments rather than single-user, single-system development. It supports the installation of multiple versions of a package, each with different build variants. By design, Spack is compiler- and MPI-agnostic and can be easily deployed on a shared file system. Supported by its community-driven, HPC-focused package recipes, it was selected as the most suitable package manager for the cluster at GSI/FAIR.
The build process is performed on a dedicated server within a containerized environment and is subsequently distributed to the worker nodes of the cluster via CVMFS. This setup supports different Linux distributions and CPU architectures as build targets.

The scripts that define the build setup and environment are stored in Git repositories, ensuring that the software installation is fully reproducible at any point in time.

3 Summary

We have developed all the essential building blocks of the software infrastructure required for operating the computing cluster within a containerized environment: automatic login into container space on a submit node; Slurm plugin for seamless execution of a computing job in container environment; CI workflows for building, testing and deployment of container images; and Spack-based tools for maintaining the software stack. The concept of VAE, specially created for the GSI/FAIR cluster, has been proven as user-friendly and effective ready-to-be-used solution for better user experience during learning stage. The achieved scalability and reproducibility have ensured stable operation for over five years, meeting the computing needs of experiments and research groups at GSI, FAIR, and partner institutions.
The clear separation between the host system and user application layers enables seamless updates to the operating system and to the resource management tools without affecting the user environment or the reproducibility of computational results.

Bibliography

[BBC11] J. Blomer, P. Buncic, P. Charpentier. Distributing LHC application software via the CernVM file system. *J. Phys. Conf. Ser.* 331:042003, 2011. doi:10.1088/1742-6596/331/4/042003

[Bra19] P. Braam. The Lustre Storage Architecture. *CoRR* abs/1903.01955, 2019. doi:10.48550/arXiv.1903.01955

[GFB+04] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *European Parallel Virtual Ma-*

chine/Message Passing Interface Users' Group Meeting. Pp. 97–104. 2004.
doi:10.1007/978-3-540-30218-6_19

[GLC+15] T. Gamblin, M. P. LeGendre, M. R. Collette, G. L. Lee, A. Moody, B. R. de Supinski, S. F. Keller. The Spack Package Manager: Bringing Order to HPC Software Chaos. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. Pp. 1–12. 2015.
doi:10.1145/2807591.2807623

[KPK+24] M. Kretz, V. Penso, D. Klein, D. Kresan, M. Mayer. GSI-HPC/slurm-singularity-exec: v3.1.0. May 2024.
doi:10.5281/zenodo.11189831

[Pen25] V. Penso. OpenSSH Container Login. Aug. 2025.
doi:10.5281/zenodo.16752892

[Sch] SchedMD. Container Guide. https://slurm.schedmd.com/containers.html. Accessed: 2025-08-06.

[The24] The OpenSSH Project. OpenSSH Portable Release. https://www.openssh.com/, 2024. Version 9.7p1.

[YJG03] A. B. Yoo, M. Jette, M. Grondona. Slurm: Simple Linux Utility for Resource Management. In *Job Scheduling Strategies for Parallel Processing*. Pp. 44–60. Springer, 2003.
doi:10.1007/10968987_3