

Report on shared software stack prototype

MultiXscale Deliverable 1.1
Deliverable Type: Report
Delivered in December, 2023

MultiXscale
EuroHPC Centre of Excellence for
Multiscale Modelling



**Co-funded by
the European Union**



EuroHPC
Joint Undertaking

Acknowledgement

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) under grant agreement No 101093169.

Disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European High Performance Computing Joint Undertaking (JU). Neither the European Union nor the granting authority can be held responsible for them.

Project and Deliverable Information

Project Title Project Ref. Project Website EuroHPC Project Officer	MultiXscale: EuroHPC Centre of Excellence for Multiscale Modelling Grant Agreement 101093169 https://www.multixscale.eu Dr. Linda Gesenhues
Deliverable ID Deliverable Nature Dissemination Level Contractual Date of Delivery Actual Date of Delivery	D1.1 Report Public Project Month 12 (31 st December, 2023) 28 th December, 2023
Description of Deliverable	Intermediate report on development of a stable, shared software stack.

Document Control Information

Document	Title:	Report on shared software stack prototype
	ID:	D1.1
	Version:	As of December, 2023
	Status:	Accepted by Steering Committee
	Available at:	https://www.multixscale.eu/deliverables
Review	Document history:	Internal Project Management Link
	Review Status:	Reviewed
Authorship	Written by:	Pedro Santos Neves (RUG)
	Contributors:	Bob Dröge (RUG)
	Reviewed by:	Caspar van Leeuwen (SURF)
	Approved by:	Alan O'Cais (University of Barcelona)

Document Keywords

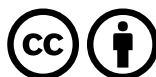
Keywords:	MultiXscale, HPC, software , CI/CD
-----------	------------------------------------

28th December, 2023

Disclaimer: This deliverable has been prepared by the responsible Work Package of the Project in accordance with the Consortium Agreement and the Grant Agreement. It solely reflects the opinion of the parties to such agreements on a collective basis in the context of the Project and to the extent foreseen in such agreements.

Copyright notices: This deliverable was co-ordinated by Pedro Santos Neves¹ (RUG) on behalf of the MultiXscale consortium with contributions from Bob Dröge (RUG). This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit:

<http://creativecommons.org/licenses/by/4.0>



¹p.m.santos.neves@rug.nl

Contents

Executive Summary 1

1 Introduction 2

1.1 Scope of the deliverable 2

1.2 Target audience 2

1.3 Report outline 2

2 Passive Continuous Deployment 3

2.1 Overview and setup of CernVM-FS network 3

2.2 Streaming scientific software to clients 3

2.3 Availability on (Euro)HPC systems 3

3 Supported software 5

3.1 Key MultiXscale Applications 5

3.2 Other applications 5

3.3 End-user experience 5

4 Supported client systems 7

4.1 Operating Systems 7

4.2 Hardware 7

4.2.1 CPUs 7

4.3 GPU support 8

4.4 Interconnects 8

5 Continuous Integration and Monitoring 9

5.1 Continuous Integration 9

5.2 Monitoring 9

6 Conclusion and outlook 10

Acknowledgements 11

References 11

List of Figures

1 Simplified schematic representation of a typical CernVM File System (CernVM-FS) setup. Icon source: <https://www.flaticon.com>. 3

List of Tables

1 Key MultiXscale Applications 5

Executive Summary

This report summarizes the progress and work carried out for the shared software stack prototype. This software stack is implemented through the European Environment for Scientific Software Installations (EESSI) [1], a project focusing on developing a shared scientific software stack, the design and capabilities of which will be elaborated on in this report.

We begin, in Section 2, by addressing Continuous Deployment (CD) and describing the development and the implementation of the file system, which is responsible for distributing scientific software. Several milestones in the development of the filesystem were achieved, beginning with the setup of a CernVM File System (CernVM-FS) software distribution network, hosted and operational at the University of Groningen, with funding from the MultiXscale CoE and additional server CernVM-FS instances which are hosted by Amazon Web Services (AWS) and Microsoft Azure. These additional server instances provide redundancy and reliability for the system. In addition, the shared software stack is now also included in EuroHPC systems Vega and Karolina and in a number of MultiXscale partner HPC systems. Furthermore, in collaboration with the developers from CERN, the deployment of the shared software stack is made easier by including it in the default CernVM-FS configuration. Lastly, to ensure a low barrier of entry and ease of use and maintenance by system administrators and end users, extensive documentation on accessing the shared software stack is provided in the [EESSI documentation portal](#)².

Section 3, titled *Supported Software*, focuses on the available scientific applications and their accessibility for users. The key applications for MultiXscale (A Load-balancing Library (ALL), Extensible Simulation Package for Research on Soft Matter Systems (ESPResSo), Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS), Open Field Operation And Manipulation (OpenFOAM), Widely Applicable Lattice Boltzmann solver from ERLAngen (waLBerla)) are now included in the shared software stack. Additionally, software relevant for other disciplines was also included in order to widen the audience of potential users, which we believe is key for the project to be sustainable. The radio astronomy community has shown interest in EESSI through the Square Kilometre Array (SKA) project and is currently assessing the potential value of EESSI for their domain. In total, 139 unique applications are available, together with 690 unique extensions (for Perl, Python and R), and a total of 1709 builds.

We give an overview of the *Supported client systems* in Section 4, that is, how we ensure the shared software stack works on a wide range of Linux distributions. This section also describes how the software installation used by the end-user is optimized for the specific hardware in their system, and which CPU architectures are supported. Finally, it describes our current level of GPU and interconnect support.

Lastly, we discuss *Monitoring and Continuous Integration (CI)* in Section 5 and provide an overview of how we aim to ensure the quality of the shared software stack. Monitoring will be done for both the infrastructure as well as for the applications that make up the shared software stack. Continuous Integration (CI) workflows, likewise, are employed at two levels, the first when the infrastructure of the shared software stack is changed, and the second when new software is shipped with the shared software stack. In addition, we have a [GitHub Action](#) that facilitates the use of the shared software stack in CI of other software, including the key applications in MultiXscale.

²<https://www.eessi.io/docs/>

1 Introduction

1.1 Scope of the deliverable

This deliverable aims to provide notes on the progress of the development of the shared software stack prototype and its transition to a production-quality Technology Readiness Level (TRL). In addition, it lists currently implemented features and software and gives an overview of current development and future plans.

1.2 Target audience

This report is aimed at High Performance Computing (HPC) system managers and administrators, scientific software developers, junior and expert end-users who wish to use, deploy or extend the shared software stack.

1.3 Report outline

In Section 2 we outline the progress, design, and implementation details of the deployment of the shared software stack through what we call *passive CD*, where the deployment of software at hosting sites is done automatically, without any intervention of local site administrators.

In Section 3 we discuss the available software, both generally and in key applications specific to MultiXscale. We also describe how an end user can easily start using the shared software stacks via a familiar interface.

Section 4 details the design and implementation of the measures which allow the shared software stack to be utilised across a very broad spectrum of Linux distributions and the progress made in supporting additional technologies such as Message Passing Interface (MPI), and preliminary NVIDIA CUDA GPU acceleration support. Additionally, we describe in Subsection 4.2.1 which CPU types the software is available and optimised for (at the time of writing, December 2023).

In Section 5 we describe how monitoring of the performance and health of the system is achieved currently, and what work will be done in the future to extend the collected metrics and their visualisation. Furthermore, we lay out the implementation of CI at the multiple levels of the shared software stack, and the potential these workflows have on enhancing the capabilities of the shared software stack.

Lastly, in Section 6 we outline future developments and work planned for the shared software stack and we summarise the progress achieved so far.

2 Passive Continuous Deployment

The shared software stack is being made available using CernVM-FS, a filesystem developed by CERN specifically for distributing software installations. CernVM-FS is highly optimized for this particular use case, and offers various levels of caching and redundancy to further improve the performance and increase resilience against failure. Client packages are available for, and can be easily installed on, many Linux distributions, which align well with the goal of EESSI to make the shared software stack available to virtually any Linux user.

In addition, this filesystem provides the foundation for the *passive continuous deployment* that we aim to do with the software stack: new software can be continuously added to the shared software stack on a centrally located server, and after several minutes this new software will then automatically become visible and usable on all client systems. No operation is required from the clients themselves.

2.1 Overview and setup of CernVM-FS network

Based on the experience within the EESSI project, a new network of CernVM-FS servers has been set up to provide a production-ready infrastructure for this shared software stack. MultiXscale funding has been used to purchase a physical server located at the University of Groningen. This server hosts the CernVM-FS stratum 0, which holds the definitive copy of the software stack (see Fig. 1, dark green circle in the center). The stratum 0 server is tightly secured with strict firewall rules and Yubikeys.

AWS and Azure clouds are being used to run stratum 1 servers (in Fig. 1 the light green circles over the circumference), which mirror the stratum 0. These are the servers that end-user clients connect to to make the shared software stack available on their system. An optional, but highly advisable final intermediate layer between the mirror servers and the HPC cluster, made up of so called squid proxy servers provides lower latency and faster access to the hosted software (Fig. 1, represented by the smaller light blue circles).

Clients will automatically connect to the closest mirror server. If one server would become unavailable, they automatically switch over to the another option. In the near future, more of these mirror servers can and will be added, in order to further increase the performance and redundancy.

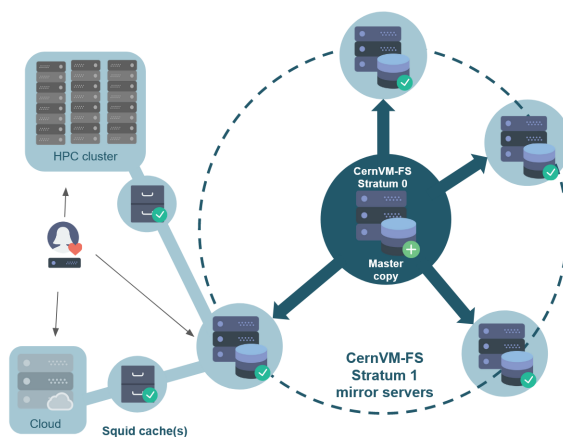


Figure 1: Simplified schematic representation of a typical CernVM-FS setup. Icon source: <https://www.flaticon.com>.

2.2 Streaming scientific software to clients

Users of the shared software stack mount the filesystem on their Linux machine, which only involves installing a CernVM-FS client package and an accompanying configuration package. This should make the new shared software stack available on their system, accessible at `/cvmfs/software.eessi.io`. Note that this does **not** require downloading the full software stack to their system. Instead, the CernVM-FS client offers a view of the actual filesystem, and the scientific applications themselves are being "streamed" to the client on-the-fly whenever they are being used. Critically, only the files that are actually being *accessed* are downloaded by the client. This greatly reduces the amount of data that has to be downloaded as compared to, for instance, containers.

2.3 Availability on (Euro)HPC systems

Through collaboration with the developers of CernVM-FS, the EESSI configuration has recently been added to the default CernVM-FS client configuration. This means that the shared software stack provided by EESSI automatically

becomes available to nearly any system that already has the CernVM-FS client installed. More and more HPC clusters are already providing this client, and this includes several European High Performance Computing Joint Undertaking (EuroHPC) systems. For instance, the EESSI shared software stack is now already available on EuroHPC systems Vega in Slovenia and Karolina in Czechia, and there is ongoing contact with other sites to discuss additional deployments via the collaboration activities of the CASTIEL2 project. Additionally, several MultiXscale partners, such as SURE, the University of Groningen and Ghent University, among others, have made the stack available to the users of their system(s).

In order to provide users of the shared software stack with instructions on getting access to and using the stack, extensive documentation is available at the [EESSI documentation portal](https://www.eessi.io/docs/)³. The landing page of the documentation provides quick links based on the type of user:

- For end users, which focuses on initializing and using the shared software stack.
- For system administrators, which focuses on making the shared software stack available on their systems.
- For contributors, which explains how someone can add software to the shared software stack.

³<https://www.eessi.io/docs/>

3 Supported software

Scientific software has become increasingly important and diversified in a large number of fields, with some applications becoming widely adopted standards and many others in a flux of development and adoption. Given the environment within which it is developed, a scientific software stack will typically be characterized by (mostly) open source applications for which performance and optimisation are very important. Initial development efforts of EESSI focus on this use case, while simultaneously providing advice on how the EESSI infrastructure can be leveraged and adapted for alternative use cases (such as industrial use cases and licensed software)

Software is currently built and optimised for eight target CPU types (see Section 4.2.1), with plans to extend the target list further, and using two recent compiler toolchain versions.

3.1 Key MultiXscale Applications

In addition to the broad scope of the shared software stack, Key MultiXscale Applications (hereafter: key applications) are available, supporting the work being carried out in Work Packages 2 through 4 in making these applications exascale-ready and producing cutting-edge research.

Software	Purpose	Technologies
ALL	Load balancing	C++ header only library
ESPResso	Molecular simulations	MPI, CUDA
LAMMPS	Molecular simulations	MPI, CUDA, Kokkos
OpenFOAM	Computational fluid dynamics	MPI
waLBerla	Computational fluid dynamics	MPI, CUDA, HIP

Table 1: Key MultiXscale Applications

3.2 Other applications

In addition to software directly required by the MultiXscale project, EESSI welcomes software contributions from other communities. The adoption of the shared software stack by the scientific software community is considered to be a very important aspect of the project which will contribute to its long term sustainability, which is reflected in the design and planning of not only the shared software stack, but also in the support portal (see D5.2 report of MultiXscale) and community contribution policy and GitHub App facilitating software contributions (see D5.1 report of MultiXscale).

This approach is proving effective in engaging contributors and end-users. As an example of the early adoption of the shared software stack by researchers at large, the radio astronomy community has been the first to request a scientific software suite tailored to the SKA radio telescope. Included software ranges from compiler toolchains, which allow users and HPC managers to compile their own custom applications, popular statistical and numerical programming environments such as R. It also includes general-purpose programming languages such as Python, TensorFlow (an AI and machine-learning platform), domain specific applications like GROMACS for molecular dynamics research. This amounts to a total of 1709 builds, 139 software packages, and 690 extensions at the present, with the numbers expected to increase as more applications are requested and newer versions of existing software are released.

3.3 End-user experience

Care was taken to make the initialisation and setup of the shared software stack as easy as possible for end-users, with auto-detection of the optimized binaries that should be served (for more details, see Section 4.2.1). Given that the distribution is guaranteed via CernVM-FS, this initialisation script can be easily sourced with:

```
source /cvmfs/software.eessi.io/versions/2023.06/init/bash
```

Applications are installed, managed and made available to HPC users through a module environment. Software installation and management is achieved via EasyBuild, a standardised software installation and management framework for HPC systems. New software can be contributed by the community, making use of the GitHub App implemented in Work Package (WP) 5 and described in the associated Deliverable D5.1. The software is then made available to users via Lmod, a module system that facilitates and abstracts environment management from users with a command as simple as:

```
module load Python/3.11.3-GCCcore-12.3.0
```

This ensures that the necessary compatible dependencies are loaded when the desired software – in this example, Python v3.11.3 – are loaded.

Environment modules, and Lmod in particular, are therefore the end-user interface of EESSI. Many existing users of HPC systems are likely to be quite familiar with environment modules giving their ubiquity on HPC systems.

4 Supported client systems

A key goal is to make the shared software stack work on a wide range of systems, irrespective of hardware and Linux distribution. The following sections explain how that is achieved.

One of the main goals of the shared software stack is to make sure it works well on as many Linux compute resources as possible, and it should not matter if this is a workstation, virtual machine, HPC cluster, or cloud infrastructure. However, we also need to make the actual scientific applications work on all those different systems. This section discusses how this has been achieved and what kind of hardware is and will be supported.

4.1 Operating Systems

For our software stack to work on an Operating System (OS), we require two things:

1. our distribution mechanism (CernVM-FS) needs to be supported on that OS.
2. our software stack needs to *function* correctly under that OS.

The first requirement means we need an OS that is supported by CernVM-FS. As discussed in Section 2, using the CernVM-FS ensures that the shared software stack can be mounted on the many different Linux distributions that support it, as discussed in Section 2.

The second requirement means we need some form of isolation from the host OS. Usually, software binaries depend on certain system libraries, and they will no longer work if you try to use them on another system where these libraries do not exist or are incompatible. For instance, in terms of Linux distributions, software that was compiled for a Debian distribution may no longer work if you try to use it on a RedHat distribution: operating system packages providing the required system libraries may be missing or have a different version. This is why we build against our own intermediate OS, this is conceptually similar to how a container provides an OS for isolation from the host but with the critical difference that our approach avoids the need for a runtime environment. The intermediate OS must be compatible with the architecture of the system it is running on. To ensure availability across a large number of systems, we offer support for both x86_64 and aarch64 architectures, and will support riscv64 in the future. Lastly, and since CernVM-FS distributes all its files under the prefix /cvmfs/<repo_name>/, we need an OS that can install under a custom prefix. [Gentoo Prefix](#) is a Linux based OS that offers this essential feature, and is thus our Linux distribution of choice. Through this system, which we will actively maintain, we ensure the shared software stack is widely portable.

4.2 Hardware

4.2.1 CPUs

As mentioned in the previous subsection, the current prototype of the shared software stack supports both x86_64 and aarch64 CPUs. However, for optimal performance, it is important that binaries are built for a specific microarchitecture (e.g. AMD Zen 3, Intel Skylake, etc.). This is why the scientific software installations are done multiple times, once for each specific microarchitecture. Currently, the prototype of the shared software stack provides optimized installations for the following microarchitectures:

- x86_64
 - generic
 - Intel Haswell
 - Intel Skylake (with AVX512)
 - AMD Zen 2
 - AMD Zen 3
- aarch64
 - generic
 - Neoverse N1
 - Neoverse V1

By using a smart auto-detection mechanism, users of the software stack will automatically get access to binaries that are optimized for their system's CPU microarchitecture. If an optimized binary is not available for a given microarchitecture, e.g. because it is too new, auto-detection falls back to the newest architecture that *is* supported. As a last

resort, if a microarchitecture is completely unrecognized then the auto-detection mechanism will fall back to generically optimized binaries.

4.3 GPU support

The prototype of the shared software stack currently contains initial support for NVIDIA GPUs. While support for NVIDIA GPUs is in a more mature stage, in future work we intend to assess the necessary steps to support AMD GPUs. There are three key challenges when it comes to NVIDIA GPU support:

1. The Compute Unified Device Architecture (CUDA) Software Development Kit (SDK) does not allow redistribution of the full SDK, only runtime components can be redistributed.
2. The drivers on the host system need to be found by the software in the shared software stack.
3. Newer versions of the CUDA SDK only work when sufficiently recent drivers are present on the host. The compatibility range can be increased (within limits) by installing the [CUDA Compatibility Libraries](#).

To overcome the first challenge, we install the full CUDA SDK in our build environment. This allows us to build CUDA enabled software (e.g. ESPResSo, GROningen MACHine for Chemical Simulation (GROMACS) and TensorFlow). However, we then ship only the part that the CUDA SDK license allows to be redistributed in the shared software stack. This is enough to *run* CUDA enabled software (such as ESPResSo, GROMACS and TensorFlow), but not to build new CUDA-enabled software. To enable end users to build CUDA-enabled software, system administrators of the host site (i.e. the site offering the shared software stack on their system) can install a full CUDA SDK copy in a special directory (which we refer to as the `host_injections` directory), that then seamlessly integrates with the shared software stack (appearing to the end user as if the full copy was shipped as part of the stack).

To resolve the second challenge, the visibility of the GPU drivers, the host site needs to run a small script, which also creates links in a specific location under the `host_injections` directory that point to the GPU drivers on the host system. Software installed in the shared software stack will look for the drivers in the defined location under `host_injections` directory, follow the links, and correctly pick up the ones from the host.

Finally, the third challenge is mitigated by checking the compatibility and printing a clear warning about whether the host drivers can support the required CUDA version. If the end-user loads a module that requires CUDA as a dependency, the version of CUDA required is checked against the version compatible with the host driver. If that combination is incompatible, instructions are printed to contact the sysadmin and request either a driver update, or to consider installing compatibility libraries.

Since enabling GPU support requires some steps from system administrators from host site, these will be clearly described in the [GPU documentation](#).

4.4 Interconnects

The MPI installations that are included in the shared software stack have been compiled in such a way that they should already support many different types of interconnects, e.g. Infiniband, Omnipath, and EFA. Site-specific tuning of MPI can typically be done by the use of environment variables for the specific MPI implementation. However, some host sites may still prefer their own MPI installations to be used, e.g. because they have a highly optimized MPI library from the vendor of their HPC system.

To make this possible, we inject a small entry (known as an `RPATH`) in the header of any executable compiled with MPI support in the shared software stack. This causes the executable to *first* look in (another) specific location in the `host_injections` directory for an MPI library, before it falls back to using the MPI shipped with the shared software stack. Thus, if a host site installs (or links) their own MPI installation in the `host_injections` directory, that gets used preferentially.

Note that a requirement is that the MPI installation injected in this way is so-called *ABI compatible* [2] with the MPI library used in the shared software stack.

The current support for interconnects already enables their use. Future work will focus on assessing their scalability in site-specific contexts and ensuring that, in relevant applications, their use is beneficial in large compute node contexts.

5 Continuous Integration and Monitoring

In order to ensure the availability, stability, and performance of the infrastructure and contents of the shared software stack, several components are (planned to be) in place for monitoring and continuous integration.

5.1 Continuous Integration

Continuous Integration (CI) is implemented at many levels to make sure that any proposed changes in the components of the software stack are being checked before they get merged into the software stack and made accessible to end users. CI pipelines are also extensively used when new pull requests are opened by contributors or EESSI developers to add a new scientific application to the shared software stack.

Examples of features implemented through CI workflows:

- At the infrastructure level, new releases are built, tested and distributed through CI workflows.
- Software licenses of contributed applications are checked to confirm they match a valid and accepted Software Package Data Exchange (SPDX) license.
- On new application contributions, count and identify which missing dependencies would need to be installed.
- Ensure that every newly contributed application has been built for all the supported CPU microarchitectures.
- Run the test suite, for which application specific tests can be created.

Because the shared software stack can be mounted and used on any Linux system, it can easily be used in CI workflows (e.g. GitHub Actions or GitLab Runners) as well. Besides all the capabilities that this provides to EESSI developers, it can be used by anyone else as well. For instance, scientific software developers can also use the shared software stack in their software development life cycle by making use of all the scientific libraries and applications provided by the software stack, allowing them to run various tests against different versions of dependencies of their software or using different compiler toolchains.

To make this process even easier, EESSI provides an [EESSI Github Action](#) that will make the EESSI software stack available in a GitHub Workflow. With just a single line of code this can be added to any existing GitHub Workflow, immediately resulting in a complete and reproducible build environment for the project. Though GitLab does not offer a similar kind of Marketplace, we will soon make documentation available that explains how developers can achieve the same thing in a GitLab CI/CD pipeline.

5.2 Monitoring

Monitoring is and will be done at several levels: first, the health of the infrastructure itself is being monitored to make sure that the servers hosting the shared software stack are available and in sync. Monitoring of disk space and the information gathered via the tools available in AWS is already ongoing. In addition, a [end-user facing status page](#)⁴ allows users to quickly get information on the availability of the system infrastructure and which sends alerts to the EESSI administrators in case issues are found. We intend to extend this with checks on the performance of the servers by doing regular bandwidth checks to and between the different servers.

The contents of the shared software stack will also be monitored properly by leveraging the test suite described in Deliverable D1.2. This will allow us to collect performance metrics of the scientific applications included in the software stack for a range of HPC sites (and in particular targeting EuroHPC sites), and these will be used to provide a status page of the stack on these sites. The page will also include an overview of all available applications.

⁴<https://status.eessi-infra.org>

6 Conclusion and outlook

In this report we outlined the progress that has been made so far in working towards a production-ready version of a scientific software stack that can be used on virtually any Linux system.

New infrastructure has been set up for this new stack, and the required configuration files for clients are now part of a common configuration package provided by **CernVM-FS**, which makes it extremely easy for potential users to start using the stack on their systems. As soon as users install the **CernVM-FS** package on their system using a package manager they will be given immediate get access to a large number of scientific applications, not only including the Key MultiXscale Applications and their dependencies, but also lots of other popular software.

With the stack containing its own operating system base layer and providing optimized binaries for a whole range of different CPUs, we already support lots of different types of client systems. Besides providing stable support for x86_64 CPUs, we also support popular ARM CPUs that are, e.g., available in the commercial Azure and AWS clouds, as well as in Deucalion, a recently inaugurated ARM based **EuroHPC** system. Even NVIDIA GPUs and different interconnects can already be used in conjunction with the software that is part of the stack.

Initial monitoring and continuous integration has been developed and set up to ensure the stability, reliability, and performance of the stack itself and its software installations. The **CI/CD** capabilities of our software stack also provide users and developers with unique possibilities to integrate the stack into their development workflows, unleashing lots of possibilities for testing in reproducible environments.

In the upcoming months, given that initial support for ARM and NVIDIA GPUs has been achieved, we intend to acquire experience with these systems by exploring and technical details and possibility of including additional software that makes use of their capabilities. Also, we intend to study and identify the necessary steps to support further technologies, such as RISC-V CPU and AMD GPU support. In terms of software, we will be expanding the amount of scientific software packages, and updating versions for already included packages. Most of this work can be automated and user-triggered due to the work done in Work Package 5. The monitoring of the shared software stack will be extended by setting up detailed monitoring pages that will collect information from the test suite developed in **WP 1**. Finally, documentation will continuously be worked on as features in the shared software stack mature and/or new features are added and improved.

Acknowledgements

We are grateful to Amazon Web Services (AWS) and Microsoft Azure for generously sponsoring the EESSI project with cloud credits, feedback, and guidance. We gratefully acknowledge the support provided by the CernVM-FS development team in setup and configuration of the shared software stack's CernVM-FS repository, and for including it in the default CernVM-FS configuration for easy distribution. We thank the administrators of Vega at the Institute of Information Science, Slovenia (www.izum.si) and Karolina at the IT4Innovations National Supercomputing Center at the Technical University of Ostrava, Czechia (www.it4i.cz) for early deployment of the shared software stack.

References

Acronyms used

AWS	Amazon Web Services
CD	Continuous Deployment
CI	Continuous Integration
CernVM-FS	CernVM File System
EESSI	European Environment for Scientific Software Installations
HPC	High Performance Computing
WP	Work Package
MPI	Message Passing Interface
OS	Operating System
SDK	Software Development Kit
SKA	Square Kilometre Array
SPDX	Software Package Data Exchange
TRL	Technology Readiness Level
HIP	Heterogeneous-compute Interface for Portability
EuroHPC	European High Performance Computing Joint Undertaking

Software mentioned

ALL	A Load-balancing Library
ESPresSo	Extensible Simulation Package for Research on Soft Matter Systems
CUDA	Compute Unified Device Architecture
GROMACS	GRoningen MACHine for Chemical Simulation
LAMMPS	Large-scale Atomic/Molecular Massively Parallel Simulator
OpenFOAM	Open Field Operation And Manipulation
waLBerla	Widely Applicable Lattice Boltzmann solver from ERLAngen

URLs referenced

Page ii

<https://www.multixscale.eu> ... <https://www.multixscale.eu>
<https://www.multixscale.eu/deliverables> ... <https://www.multixscale.eu/deliverables>
 Internal Project Management Link ... <https://github.com/multixscale/planning/issues/101>
p.m.santos.neves@rug.nl ... <mailto:p.m.santos.neves@rug.nl>
<http://creativecommons.org/licenses/by/4.0> ... <http://creativecommons.org/licenses/by/4.0>

Page iii

<https://www.flaticon.com> ... <https://www.flaticon.com/authors/smashicons>

Page 1

EESSI documentation portal ... <https://www.eessi.io/docs/>
 GitHub Action ... <https://docs.github.com/en/actions>

Page 3

Yubikeys ... <https://en.wikipedia.org/wiki/YubiKey>
<https://www.flaticon.com> ... <https://www.flaticon.com/authors/smashicons>

Page 4

EESSI documentation portal ... <https://www.eessi.io/docs/>

Page 5

R ... <https://cran.r-project.org/>

TensorFlow ... <https://www.tensorflow.org/>
GROMACS ... <https://www.gromacs.org/>
EasyBuild ... <https://easybuild.io/>
Lmod ... <https://lmod.readthedocs.io/>

Page 7

Gentoo Prefix ... <https://wiki.gentoo.org/wiki/Project:Prefix>

Page 8

CUDA Compatibility Libraries ... <https://docs.nvidia.com/deploy/cuda-compatibility/index.html>
GPU documentation ... <https://www.eessi.io/docs/gpu/>

Page 9

EESSI Github Action ... <https://github.com/marketplace/actions/eessi>
end-user facing status page ... <https://status.eessi-infra.org>

Page 11

www.izum.si ... <https://www.izum.si/en/home/>
www.it4i.cz ... <https://www.it4i.cz/en/infrastructure/karolina>

Citations

- [1] B. Dröge, V. Holanda Rusu, K. Hoste, C. van Leeuwen, A. O’Cais, and T. Röblitz, “EESSI: A cross-platform ready-to-use optimised scientific software stack,” *Software: Practice and Experience*, vol. 53, no. 1, pp. 176–210, 2023. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3075>
- [2] “Application binary interface,” https://en.wikipedia.org/wiki/Application_binary_interface, accessed: 2023-11-30.