D7.3 Final version HPC integration Handbook

Lead partner:	EGI Foundation
Version:	1
Status:	Under EC review
Dissemination Level:	PUBLIC
Keywords:	HPC, EOSC Compute Platform
Document Link:	https://documents.egi.eu/document/3804

Deliverable Abstract

The HPC integration handbook describes how HPC systems can be incorporated into the EOSC Compute Platform delivered by the EGI-ACE project. This handbook is the result of the piloting activities with use cases and providers supporting workflows that run on combined Cloud, HTC and HPC resources. In this document we detail the integration mechanisms used in the pilots and provide information for providers on how to integrate HPC systems into the EOSC Compute Platform, and for users on how to use these HPC systems via EOSC for running container-based workloads.

EGI-ACE receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101017567.

go.egi.eu/egi-ace

EGI-ACE receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101017567.

DELIVERY SLIP

	Name	Partner/Activity
From:	Enol Fernández	EGI Foundation/WP7
Moderated by:	Sjomara Specht	EGI Foundation/WP1
Reviewed by:	Marco Rorro Sebastian Luna-Valero	EGI Foundation EGI Foundation
Approved by:	SDS	

DOCUMENT LOG

Issue	Date	Comment	Author
v.0.1	08/07/2022	Full draft - The initial version, M7.3 HPC integration handbook, was published in Zenodo.	F. Antonio (CMCC) H. Bayındır (TUBITAK) I. Díaz (CESGA) M. Dulea (IFIN-HH) C. Fernández (CESGA) E. Fernández (EGI Foundation) J. Gomes (LIP) A. Lahiff (UKAEA) D. Southwick (CERN) D. Spiga (INFN) T. Tanin, (IICT-BAS) G. Sipos (EGI Foundation)
v.0.2	28/07/2022	Incorporate comments from reviewers	E. Fernández, M. Rorro, S. Luna-Valero
v.0.3	28/07/2022	Send to SDS	
v.0.4	01/08/2022	Comments received and included	E. Fernández
v.1	01/08/2022	Final	

TERMINOLOGY

https://confluence.egi.eu/display/EGIG

Contents

Executive summary

The EOSC Compute Platform, delivered by the EGI-ACE project, is a system of federated compute and storage facilities, complemented by diverse access, data management and compute platform services. The EOSC Compute Platform is designed to support a wide range of scientific data processing and analysis use cases, including the hosting of scientific services and data spaces. The infrastructure layer of the EOSC Compute platform initially builds on compute cloud, container and High Throughput Compute facilities. During EGI-ACE this layer was extended with High Performance Computing (HPC) systems, and access to the federated HPC resources was be demonstrated by Open Science workflows that span across all EOSC Compute platform continuum.

The HPC integration work in EGI-ACE focused on the following areas:

1. EOSC-compliant federated access management on HPC systems via the EOSC Portal.
2. Availability and reliability monitoring of federated HPC sites.
3. Integrated usage accounting across HPC, cloud and HTC sites.
4. Access to distributed, federated data from HPC systems.
5. Portable container-based applications for cloud compute, HTC and HPC systems.

These areas were investigated by 4 HPC centres, CESGA (Spain), IICT-BAS (Bulgaria), LIP/INCD (Portugal), and TUBITAK (Turkey), all members also of the EuroCC project.

The findings will be configured on these sites and will be validated through 4 scientific workflows:

- Climate research use case from ENES (CMCC, Italy)
- High Energy Physics simulations for the High Luminosity run of the Large Hadron Collider (CERN) HEP
- Cross platform fusion workflows (UK Atomic Energy Authority) PROMINENCE
- Photon and neutron science use case from the ELI-NP Research Infrastructure (IFIN-HH, Romania)

Based on this work this document provides an architecture blueprint for HPC providers about how to provision HPC resources with federated access management, monitoring, accounting, data and application access mechanisms in EOSC.

Using this blueprint EGI-ACE will engage with external HPC providers to boost the uptake, and ultimately to increase the presence and accessibility of HPC systems via the EOSC Portal.

1 Introduction

The EOSC Compute Platform, delivered by the EGI-ACE project, is a system of federated hybrid compute and storage facilities, research data hosting, processing and analytics tools, and a set of complementary services for distributed data and compute access to support processing and analytics for distributed data and compute use cases. The Platform builds on the existing EGI infrastructure and seeks to expand it towards HPC systems to support heterogeneous computing workflows through a combined use of HTC, HPC and Cloud.

Under the guidance of four international use cases, the project delivers this architecture blueprint for the provisioning of HTC and HPC resources with integrated federated access management, monitoring, and accounting. The blueprint is focused on four areas: (1) access to HPC systems via policies, solutions and protocols supporting federated identities; (2) the external publication of HPC usage metrics; (3) the execution of portable container-based applications in IaaS, HTC and HPC systems, and (4) federated data access from HPC systems.

This handbook is delivered by the 'HPC integration' task members of EGI-ACE:

- 4 HPC providers belonging to National Competence Centres (NCCs) of the EuroCC project (H2020 project with 50% funding from EuroHPC JU, 50% national funding). NCCs are the central points of contact for HPC and related technologies in their country. EGI-ACE includes as providers: CESGA (ES), IICT-BAS (BG), LIP/INCD (PT), and TUBITAK (TR).
- 4 use case pilots with applications from different disciplines: climate research with a use case from ENES[1]; High Energy Physics with a use case looking into the computational needs of the future High Luminosity run of the Large Hadron Collider (HL-LHC)[2]; fusion[3] with a platform to run workloads across different computing infrastructures, and photon and neutron science with a use case from ELI-NP[4].
- 1 technology provider (INFN) with broad experience in delivering solutions for HTC, HPC and cloud computing, which contributed with support for the access to providers via middleware and the setup of JupyterHub for ENES pilot.

This handbook is organised as follows. First, we will describe the requirements from the use case pilots included in the HPC integration task (Section 1). Then, the document details the different options explored for providing access to the available HPC systems and to make them an integral component of the EOSC Compute Platform covering the access, security and operational integration (Section 2, 3, and 4). The following two sections cover how users can transfer data (Section 5) and execute their workflows as containers (Section 6) in the HPC systems. Finally, conclusions are given.

[1] https://is.enes.org/
[2] https://home.cern/science/accelerators/high-luminosity-lhc
[3] https://marketplace.eosc-portal.eu/services/prominence
[4] https://www.eli-np.ro/

2 Requirements from pilots

2.1 HEP

The High Luminosity run of the Large Hadron Collider (HL-LHC) in 2027 is expected to produce 1 Exabyte of physics data for processing, of which, the goal is to demonstrate processing of 1 Petabyte of physics data in 24 hours through an HPC site. In preparation of this goal, HEP pilot work has identified requirements in line with the thematic topics presented in this milestone document.

Due to the large volume of data that must be transferred to and from a participating HPC site, as well as potentially stored on shared file systems, it is critical that detailed information covering site connectivity and storage capabilities is published alongside applicable usage policies governing these services. Physical descriptions of the network and storage topologies alongside relevant benchmarks demonstrating expected throughput capabilities greatly aids data-driven workload users of HPC sites. Support of data transfer services (XrootD, globus, gftp, etc.), their protocols, and expected throughputs, allows better informed decisions on workload selection and scaling, and reduces potentially invasive benchmarking.

Currently, HEP workloads are served from CVMFS when available or containerized via Singularity images. The lack of a clear consensus on a preferred containerization service for HPC requires communication from site operators on what service(s) they choose to support, and any affecting permission policies enforced. If compute/worker nodes are restricted to LAN connectivity, this should be clearly communicated, as this entails intermediate storage on the shared storage system as an additional step to export/publish results. HEP workloads are diverse and will be heterogeneous in nature. AAI is detailed in the following sections.

2.2 PROMINENCE

The PROMINENCE[5] platform allows users to transparently run containerised workloads, including individual jobs as well as workflows, across any number of clouds simultaneously. Users are presented with a simple batch-system style interface available as either a CLI which can be run anywhere or a REST API. All infrastructure provisioning is handled automatically and is completely invisible to users.

Integrating PROMINENCE with existing HPC systems would allow users to access a wider variety of resources and enable users to run multi-node jobs requiring low-latency interconnects more readily, as such hardware is still rarely available in private research clouds. Requirements for HPC integration are:

- A minimum of CentOS 7 (or equivalent).
- Support for unprivileged containers, either Singularity or udocker.
- Ideally, outgoing access to the internet on port 443 (https) is required for access to external storage systems. If outgoing access is only available on login nodes this can be dealt with but access from the worker nodes is preferred.

[5] https://marketplace.eosc-portal.eu/services/prominence

- Access via ssh is fine but HTC-like access, e.g., ARC CE or HTCondor CE, could also be easily supported if available.
- The HPC site's security policies need to allow a single system to be able to submit jobs on behalf of multiple users. PROMINENCE users are of course not given direct access to the resources but are able to submit jobs to PROMINENCE using arbitrary container images running arbitrary commands.

2.3 ELI-NP

The 10 PetaWatts High Power Laser System (HPLS) - commissioned in 2020, and the Variable Energy Gamma-ray (VEGA) System - under construction, hosted at the Extreme Light Infrastructure – Nuclear Physics (ELI-NP) facility, will support breakthrough experiments in laser and nuclear physics whose preparation, optimisation and validation need intensive HPC simulations. The largest consumers of HPC resources are the particle-in-cell (PIC) simulations that are essential for predicting the results of the experimental investigation on ion acceleration and QED effects, laser-to-gamma conversion, for the development of nuclear detector systems, etc.

ELI-NP pilot is intended to meet in general the HPC user requirements and in particular the necessity for providing them with reliable PIC computing resources. User access to such computing resources to run open-source codes such as EPOCH and PIConGPU can be offered on bare-metal clusters through the AAI described in this document, or on virtual clusters through the EGI Check-in service. Generally, the virtual HPC clusters are served through pre-defined, containerized VM images. Efficient workload managers like SLURM should be provided on virtual clusters too for efficient use of the resource by multiple users.

In order to ensure superior scalability, the bandwidth available for internal communication should be at least 50 Gb/s. On high-density clusters, Mellanox EDR should be considered. For fast external data transfers, it is recommended to have internet connectivity of min. 10 Gb/s.

The computational power and the storage capacity required depend strongly on the nature of the simulated experiment and of the spatio-temporal scale chosen for the simulation. As a typical example, the running of a 2-dimensional PIC code for investigating gamma-ray generation and pair production typically requires at least 600 CPU cores. To run a 3-dimensional code, more than 1000 cores are needed, and the generated data (including the restart files) are of the order of 1 TB.

The preparation by the provider of the resources for a new PIC computing project requires close communication with the user for serving appropriate computing environments and preliminary benchmarks of the chosen software on available resources.

2.4 ENES

In several domains, such as climate science, scientific advances now rely on technologies and software solutions from both the HPC and Big Data landscapes. However, being able to efficiently exploit HPC infrastructures for running scientific data analysis is not trivial. A unified model that also allows the deployment on HPC of the same services already exploited in the cloud can pave the way for a wider range of opportunities in the scientific community, further fuelling the adoption of the *HPC as a Service* (HPCaaS) paradigm. In this respect, software containers are good candidates for supporting the *portability* and *deployment* of data analytics frameworks over multiple platforms. Thanks to the development of container technologies to become more HPC-friendly, thus targeting unprivileged environments (e.g., udocker, Singularity, Sarus), scientists could exploit the benefits of this model also on HPC infrastructures.

The *ENES pilot* aims to address services and usage scenarios relevant to the European Network for Earth System modelling (ENES) community, which gathers the European modelling community and supports the dissemination of model results to the climate research and impact communities. The pilot operates on top of the ENES Climate Analytics Service (ECAS)[6], one of the *EOSC-Hub Thematic Services* as well as a *Compute Service* in the *IS-ENES3* project. In the European Open Science Cloud (EOSC) context, ECAS represents a central component of the *ENES Data Space*[7] set up in the *EGI-ACE* project with the aim to provide an open and cloud-enabled data science environment for climate scientists. In this environment, the *Ophidia HPDA framework*[8] represents a core computing engine of the ECAS service and it can greatly benefit from the exploitation of HPC resources for running parallel data analysis applications and workflows.

In such a context, the main goal of the ENES pilot is to explore solutions for the execution of container-based climate workloads, focusing on simplifying the portability of applications across the different computing services available to EOSC users. Specifically, in order to enable a transparent and portable deployment of ECAS on top of the HPC resources made available in the EGI infrastructure, the pilot targets the use of HPC-friendly unprivileged containerization solutions (i.e., *udocker*) for the ECAS core components. Among them, the Ophidia platform, a JupyterHub instance as an entry point to a data science environment and other climate community tools and Python Data Science modules are considered. This will allow climate scientists to easily execute docker-based data analytics and visualisation jobs on scientific datasets hosted for example on the *EGI DataHub*[9]. To this end, the proper orchestration of container-based jobs for climate applications via HPC batch scheduler (e.g., Slurm) will also play an important role in this process. All these requirements have been taken into consideration for the exploration of a set of potential climate-oriented usage scenarios over HPC resources. For the ENES pilot two main application scenarios have been selected that are described in sections below.

Both scenarios have been set up and tested on the resources provided by TUBITAK. Though some technical difficulties were encountered for the implementation of the pilot use cases, in terms of proper integration of the different components on the HPC unprivileged environment. In the long term, the goal is to try to reuse the experience from the pilot to

[6] https://marketplace.eosc-portal.eu/services/enes-climate-analytics-service
[7] https://enesdataspace.vm.fedcloud.eu
[8] https://ophidia.cmcc.it
[9] https://www.egi.eu/services/datahub/

integrate HPC-based services within the context of the ENES Data Space for supporting extreme-scale scenarios.

3 Access to HPC Systems

Traditionally, access to HPC systems is performed via login nodes where users connect to with SSH. From those nodes, users can interact with the batch system and submit jobs for their execution. The SSH credentials are typically locally managed usernames and password or SSH keys. For EGI-ACE, we tested and deployed new ways of accessing the providers that leverage federated authentication, so users do not need to manage a new set of credentials for each of the individual HPC systems. The following mechanisms were considered for the EGI-ACE pilots:

- SSH access with federated identities: The HPC system login node is configured to accept federated identities with new technologies like OpenID Connect or by synchronising SSH keys from a federated identity system like EGI-Check-in. In EGI-ACE we have piloted the access with SSH-OIDC[10] as described below. This access method minimises the requirements to the users, who just need to get access tokens from Check-in, and does not introduce disrupting changes to the providers. SSH access with SSH keys from a federated identity system like EGI Check-in is possible with the synchronisation of accounts with the federated LDAP registry that include among other user information, the public keys of the users. This is currently used in the C-SCALE project for the integration of HPC providers[11] and not tested in EGI-ACE.
- HPC as a Service: Users are presented with APIs to create virtual infrastructures with HPC capabilities, e.g. specialised hardware like GPU accelerators and low latency networks like InfiniBand, to run their workloads. Virtualisation may introduce some extra overhead, although this may be acceptable for a wide number of HPC needs and bare-metal clusters may be also supported in HPC as a Service offers. For EGI-ACE the HPC as a Service is provided using OpenStack APIs for the provisioning of the virtual infrastructure with automated deployment of clusters using EC3.
- Access via middleware and or gateways: In this case, users do not get direct access via SSH, but a set of middleware components will handle the user access and the interaction with the HPC system. For EGI-ACE, we considered the existing HTC middleware[12] for delivering access to the providers. Similarly, to using HTC middleware, in EGI-ACE we also consider using a web accessible portal (e.g. Jupyter) that handles the submission of jobs to the HPC cluster and users do not have direct access to the underlying system. Gateways themselves need privileges to submit jobs on behalf of the user. Using these middleware or gateways is not always possible at HPC systems due to policy restrictions.

---

[10] https://github.com/EOSC-synergy/ssh-oidc
[11]C-SCALE - D3.1 Initial Design of the Compute Federation https://zenodo.org/record/5084884
[12] https://docs.egi.eu/users/compute/high-throughput-compute/

3.1 SSH with federated identities

3.1.1 Provisioning and deprovisioning of accounts

Before a user can access an HPC cluster, a user account needs to be available on the system. Accounts can be either created 'on-the-fly' as the user first accesses the system or be provisioned via some offline mechanism. The provisioning and deprovisioning of accounts normally follow strict policies and procedures that are system-specific, thus providing common solutions may impose undesirable requirements on the individual centres and clash with local policies and setups. Even without a common approach for the provisioning of accounts, it can be technically supported with LDAP registries like the one provided by Check-in[13] or using pool accounts as in most grid systems. Besides the provisioning of the accounts, federated users need to be mapped to these local accounts, which is determined by the configuration of the access mechanism to the system.

3.1.2 OIDC token based ssh access

OIDC (OpenID Connect) is an identity layer, built on top of OAuth protocol. It allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User. This protocol reduces the friction experienced by the End-Users while allowing service providers to get the information they need relatively quickly and well-defined manner. EGI Check-in is built on top of the same technology.

With the increasing requirement of accessing HPC systems from cloud environments and using these systems for fast and distributed processing, the need for using existing authentication systems for accessing HPC clusters became a necessity.

One of the ways for achieving this interconnection is utilisation of a layer called SSH-OIDC. SSH-OIDC adds capabilities for authenticating a user over SSH using its OIDC token obtained from any OpenID Connect identity provider, including EGI Check-in.

SSH-OIDC is a relatively simple to install and administer system which requires installation on both the client and server system. It creates a side-channel to transfer OIDC related validation traffic and other side-tasks to be handled during login. This side channel can be encrypted with TLS for enabling around-the-world, in-the-open secure deployment.

In general, SSH-OIDC has these outstanding features:

- Allows any user to login via any OpenID Connect provider.
    - The HPC provider can still restrict access based on local policies, membership in a Virtual Organization, or both.
- Creates required users and groups automatically, assigns persistent usernames.
    - Groups users from the same roles under the same groups, allowing cooperation.
    - Allows username creation schemes to be configured.

[13]LDAP support in Check-in: https://docs.egi.eu/users/aai/check-in/vos/#ldap

- Installation of the components does not interfere with normal user authentication capabilities on either end.

The server side of SSH-OIDC contains two components. A service called motley-cue and an authentication plug-in called pam-ssh-oidc. The former module handles user mapping and related tasks, while the latter plugin allows tokens to be passed instead of passwords and be verified for allowing the user in.

Similarly, the user side of the installation contains two components. A command line tool mccli for interfacing with motley-cue service on the server side and an oidc-agent tool[14] for obtaining the tokens and managing the accounts (hence tokens) on the client side, including encryption for increased security. After completing the installation, a user might get its token from an OpenID Connect provider via the oidc-agent tool and connect to a supporting server via mccli tool.

Due to evolving nature of the software, an installation guide is not included in this handbook, however, the following links will provide all the necessary information regarding its installation and use:

- SSH-OIDC official document repository (GitHub).
- Client quick installation guide (kit.edu).

Since the software is distributed as RPM and DEB packages, installation, and maintenance of the utilities via standard system management commands is possible, and straightforward.

3.1.3 ENES SSH-OIDC Scenario

The ENES pilot consists of two main application scenarios. The first one consists of the use of community-based tools for the execution of climate analysis in parallel on large datasets requiring the use of HPC resources. This scenario targets a set of users with at least some basic experience with the HPC environment. From a technology perspective, it aims at understanding how federated authentication solutions and data analytics services from the cloud ecosystem can be also exploited on HPC for climate science. As shown in Figure 1, the final implementation makes use of: (i) *SSH-OIDC* as AAI solution (relying on the EGI Check-in service) for ssh-based authentication and login (see the “*OIDC token based ssh access*” paragraph under the “*Access - Authentication and Authorization*” section), (ii) *udocker* for packing the whole set of community-based data analytics tools (i.e., Ophidia) and supporting parallel execution via the *slurm* resource manager provided by the HPC system (see the related section under “*Application support*”) and (iii) services to enable a federated data access (i.e., the *EGI DataHub* service based on the *Onedata* technology, see the “*DataHub*” paragraph under the “*Data Transfers*” section) to a collection of datasets.

[14] https://github.com/indigo-dc/oidc-agent

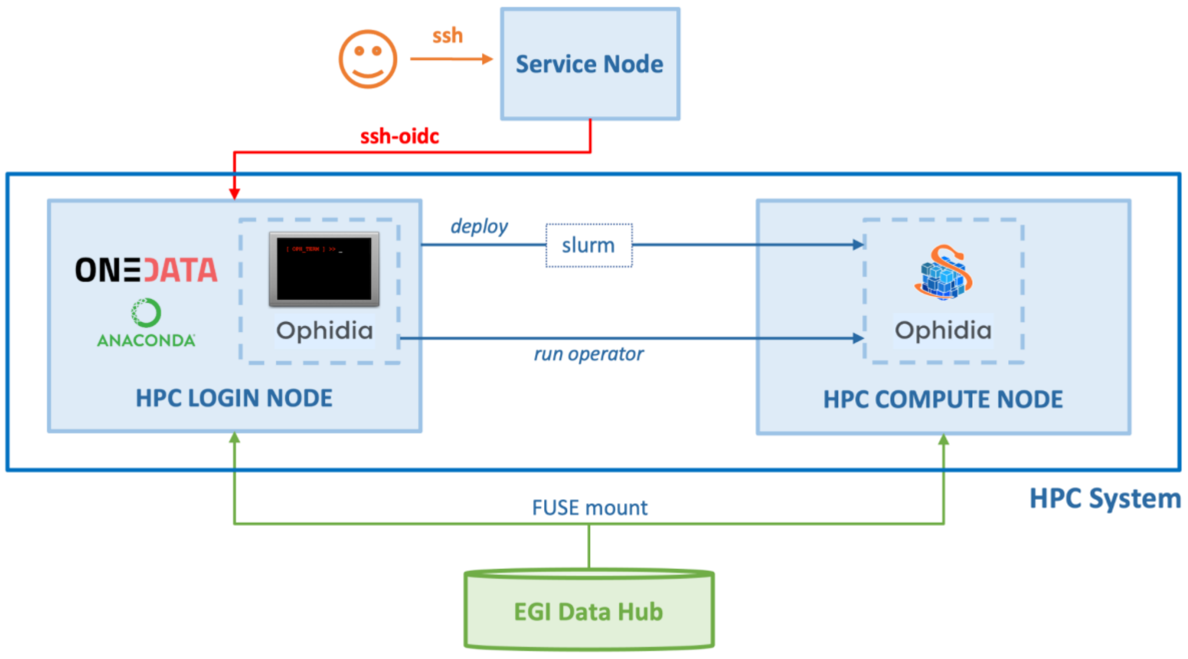


Figure 1: ENES pilot - First scenario

3.2 HPC as a Service

EGI operates a federated IaaS cloud that allows users to create virtual infrastructures for running their workloads. Although these rely on hypervisors to create Virtual Machines, specialised hardware like GPU accelerators and low latency networks like InfiniBand can be configured as PCI Passthrough devices without any significant performance overhead thus enabling the execution of certain HPC workloads.

Compared to HPC on bare-metal clusters, the HPCaaS solution offers well-known cost-efficiency, flexibility and resource scaling advantages, users can benefit from the OS and flavours of choice for implementing the infrastructure that best suits their applications, while admins don’t need to install MPI software on hosts and can easily migrate VMs between hosts in case of scheduled interruptions. Moreover, virtualization allows the construction of various testing environments for optimising the resource configurations of parallel applications.

Such advantages make HPCaaS widely adopted by commercial cloud providers, despite some drawbacks, such as the consumption of a small fraction of resources by hypervisors and the inherent influence of the overhead due to virtualization on the applications’ runtime and scalability.

Because the construction of virtual clusters by administrators of HPC clusters is not often a possibility, it is necessary to provide HPC users with automatic means of generating MPI clusters that offer familiar compute environments and resource management systems.

EC3 (Elastic Cloud Compute Cluster)[15] provides a tool to create elastic virtual clusters on an Infrastructure as a Service (IaaS), including support for SLURM or similar job management systems found in HPC systems. This approach allows for a very flexible and customizable environment for executing the user applications with complete control on the operating system and applications available and the associated hardware to each of the virtual nodes, while keeping similar interfaces as those available in the HPC centres.

HPCaaS is particularly useful for testing and optimising HPC applications in various environments because the migration of the applications between the bare-metal and the cloud clusters is straightforward as long as these two share the same Linux flavour and MPI version.

When providing HPC by using IaaS clusters it is essential to ensure appropriate performance conditions of the infrastructure are met. Even if virtual clusters with any OpenStack flavour can be deployed, better HPC performance of an application is achieved with fewer VMs per node, best would be achieved with only one VM per node using all the resources available. As in the case of bare-metal HPC clusters, resource fragmentation leads to poorer performance. The performance of any virtual cluster strongly depends on both host infrastructure and virtualization method. For this reason, it is recommended to build HPCaaS using an efficient hypervisor and an optimised HPC-ready OpenStack cluster (preferably homogenous and with low latency interconnect) coupled with a parallel file system (for applications which require that).

3.2.1 HPCaaS for ELI-NP

Motivated by the need for providing a versatile computing environment for HPC simulations at ELI-NP, an exploratory study on possible HPCaaS solutions implementable in the EGI infrastructure has been conducted.

The study was focused on the implementation of virtual clusters on OpenStack with MPI capabilities and low latency interconnects, and the investigation of the scaling performances of the EPOCH code[16] when running on this infrastructure.

Virtual clusters were deployed by using the InfiniBand (IB) Single Root I/O Virtualization (SR-IOV) interfaces and creating Virtual Functions on the compute nodes, for which a script was developed. Ubuntu-based flavour VMs were configured using Mellanox OFED (OpenFabrics Enterprise Distribution) with OpenMPI. These were utilised for conducting benchmarks on CLOUDIFIN OpenStack site, with KVM hypervisor, and the results were compared to those of similar tests performed with EPOCH on the bare-metal host cluster and FinisTerrae III supercomputer at CESGA.

It was found that the mean run time for the virtualized system is longer than in its bare-metal version by less than 4% in the case of a single VM (with 40 cores) and by less than 10% in the case of interconnected VMs. Also, when more than one VMs are considered (that is

[15] J. of Computer and System Science 79 (2013) 1341-1351

[16] Extendable PIC (Particle in Cell) Open Collaboration, T D Arber et al 2015 Plasma Phys. Control. Fusion 57 113001

when the data is not written locally), data transfers through NFS protocol hinders scaling when the number of VMs increases.

Performance improvements could be obtained by using a Type 1 hypervisor, such as VMware, for decreasing packet latency, or/and a parallel file system that scales well (e.g., Lustre, which significantly improved the results of bare-metal tests on FinisTerrae III), instead of NFS. Alternative solutions may consider the implementation of containerization over an HPC infrastructure, or the use of the OpenStack Ironic service to initialise hosts starting from a VM image which contains the OS and, e.g., udocker.

For the automation of the deployment of virtual clusters an instance of EC3 was installed on a VM as a new extension service of CLOUDIFIN. Because currently, EC3 works with Ethernet network devices only, in order to implement InfiniBand support on the cluster to be generated, two strategies are possible: 1) using the Ethernet support on IB devices for getting network bandwidth higher than Ethernet; 2) Programming a separate tool for attaching IB interfaces to VMs and using Ansible rules in EC3 for cluster reconfiguration.

In implementing the first method, the IB network devices (ConnectX-4 or newer) were configured as Ethernet port type on hosts. Successful node-to-node tests were performed between the OpenStack controller and one compute node running several VMs, the speed being downgraded from FDR (56 Gbps) to QDR (40 Gbps). For multiple compute nodes an IB switch that can use the Ethernet Mode is required. The drawbacks are that this feature can be provided on the previous generation of IB switches only under a paid license, and the initial IB speed is downgraded. Moreover, this would be a temporary solution, as NVIDIA currently offers two separate production lines of switches, for Ethernet and IB, respectively.

For implementing the second solution, a tool has been developed in Python that uses the libvirt module. The tool, which works independently of EC3, is attaching the Virtual Functions of the IB network device to any new virtual machine. After this is done, the virtual cluster should be reconfigured using Ansible rules.

3.3 Access via middleware / gateways

The goal of this integration option is to allow the exploitation of processing resources available in an existing HPC cluster by community specific workload management systems or gateways external to the HPC itself. This means that we expect that externally submitted jobs need to run on HPC resources. A typical workflow is the one where experiment specific pilot jobs reach the HPC resource and call back the Experiments Workload Management Systems and receive payloads, which are executed inside a runtime environment.

In this scenario the HTCondorCE can represent an HPC edge node with access to the external IP ranges that can be carefully defined upfront. The edge node has the role then to submit to the internal batch system. The latter is a key aspect and here HTCondor represents a suitable solution because it natively supports the compatibility with server batch middleware among which SLURM which is popular between HPC centres.

Since one of the objectives of the work is to allow the exploitation of processing resources available in HPC clusters, e.g., by single-node (multicore) or single-core HTC jobs for data processing, it is natural to rely on the job router daemon capability natively built-in by HTCondor which is the key feature which allows translating HTC submitted jobs into the internal batch. This provides the ability to transform vanilla jobs to the “slurm” batch type, thus allowing HTCondor to interface with Slurm and therefore supporting a mixing of HPC and HTC resources. Another key feature of the job routing daemon is to allow both automatic transformations as well as custom policies. Last but not least, this grants a high level of flexibility since, for example, one could use any python based script for the implementation of a custom policy.

Another key aspect is that using such an HTCondor based approach opens the possibility even to the federation of distributed providers (either HPC or not) building a pool of heterogeneous resources.

Another approach, still HTCondor based, has been developed and tested in order to provide a simplified and lightweight strategy. It can be considered an extension of the above scenario which aims to further reduce the requirements for the site. The simplified integration strategy is based on the idea that HPC resource provisioning can be decoupled from the presence of HTCondor edge service. In other words, one can implement a pluggable lightweight service that takes care to submit slurm jobs properly configured to start an HTCondor startd daemon. The latter can be configured to connect back to any external pool federating then the provided resource. In this case routing and matching, rules will be managed at the level of the main HTCondor pool where the resources are supposed to connect back. Daemon services authenticate via HTCondor Tokens which allow the main pool the full control to possibly revoke credentials.

In the end the minimal requirement for HPC centres can be offering the possibility to deploy an edge service. In the very minimal scenario, this can be any login node (preferably a dedicated node) where a script/cron job can run “forever”.

An additional plus would be:

- Outbound connectivity from the nodes. While solutions to overcome lack of outbound connectivity have been studied, the impact on performance would be not negligible.
- CVMFS for making software easily available at the HPC centre.
- singularity/udocker for the execution of software packaged as containers without extra privileges.

3.3.1 ENES JupyterHub Scenario

The second scenario of the ENES pilot targets the use of a Jupyter-based Python environment on top of a HPC infrastructure for the execution of complex climate analysis in the form of Notebooks. With respect to the first one, this scenario supports less experienced users which do not need direct access to HPC facilities but only to their computing power for the execution of notebooks. Moreover, this scenario aims to demonstrate how cloud-based services for data science can be moved transparently to HPC systems.

In this case, a udocker container will host several components such as the Jupyter Notebook server, the Ophidia centralised components and the Python modules for results post-processing and visualisation; additionally, multiple udocker worker containers will be deployed on multiple cluster nodes to enable larger scale parallel data analysis. A single JupyterHub instance is deployed on a service node integrating the EGI Check-in service to handle authentication and authorization aspects and provide users with a uniform, secure and easy way to access the environment. The service node can directly access the HPC cluster in order to orchestrate the deployment of the containers on the HPC compute nodes via the slurm batch system. In this regard, a customised version[17] of the *batchspawner* for Jupyterhub[18] has been considered in order to use ssh to submit a slurm job on the HPC system and then use a SSH tunnel to make JupyterHub able to contact the job running inside the HPC network. More specifically, a service account has been set up to spawn on the HPC nodes the udocker containers as jobs on behalf of each authenticated user and then tunnel the interface back to the JupyterHub instance. In this way, once a user is authenticated, the *batchspawner* plugin will connect to the cluster login node and submit a Jupyter Notebook job and the Ophidia server-side components as well. As soon as the job starts the execution, it sets up a SSH tunnel with the front-end host so that JupyterHub can proxy the Notebook interface to the user. This setup allows users to simply access a HPC system via a browser and exploit all their Python environment and data.

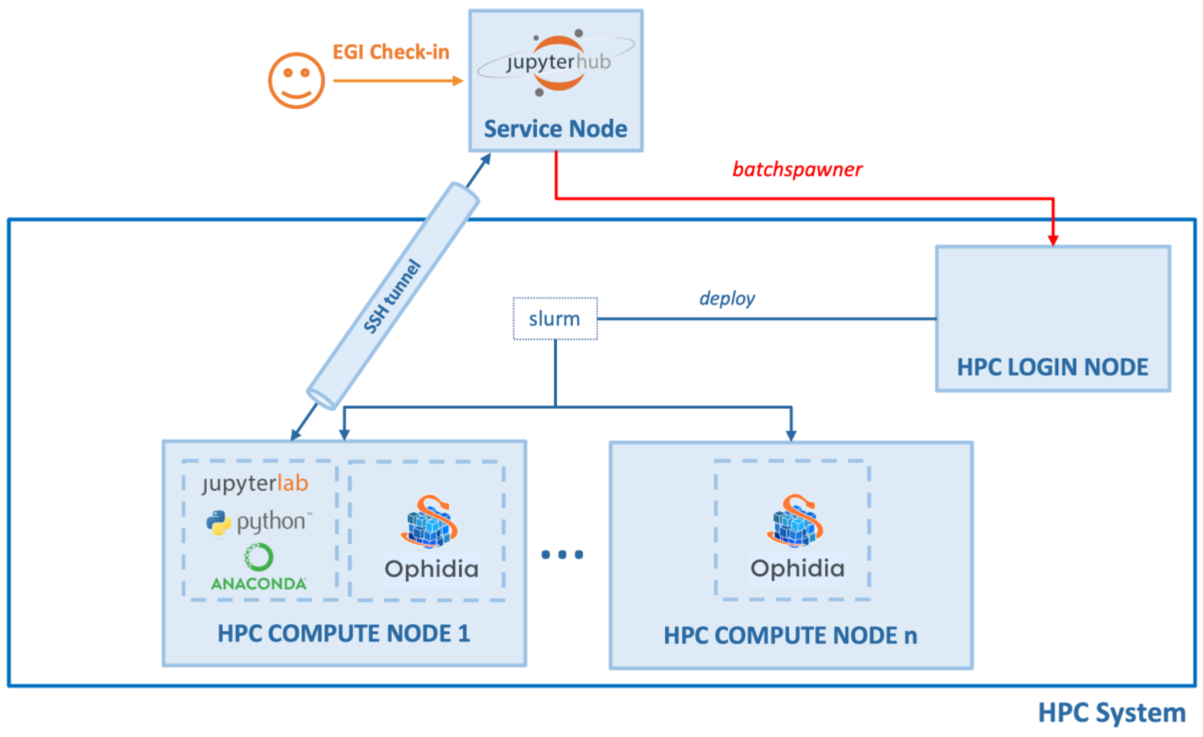


Figure 2: ENES pilot - Second scenario

3.4 Other integration mechanisms

The EGI-ACE pilots covered the integration mechanisms described above, but there are other possibilities that may be explored in future work:

- HPC as a Service on a Bare-metal. Virtualisation brings certain levels of overhead which are very critical for some workloads thus they are not suitable for running on

[17] https://github.com/DODAS-TS/remote-slurm-spawner/blob/master/remote_slurm_spawner/remote_slurm_spawner.py#L631
[18] https://github.com/jupyterhub/batchspawner

virtual machines as currently done for the HPC as a Service pilot described above. Instead, the hosts for deploying the HPC workloads can be created dynamically with a bare-metal management tool like OpenStack Ironic[19]

- Delivery of a containerized HPC infrastructure as a Service. This flavour is aimed at HPC providers and users who want to deploy an HPC cluster from EGI-ACE validated templates preconfigured with SSH access via OIDC or HTC middleware, and all the services and tools facilitating the integration with EGI-ACE. The implementation shall be driven by Ansible and result in an HPC-ready cluster on top of LXC containers, enforcing isolation from the underlying system without compromising the performance. Having a containerized HPC cluster on top of a physical HPC cluster has some advantages – (a) the function of the physical HPC cluster remains as it is, (b) the containerized HPC Cluster can utilize the entire resources provided by the physical cluster or their subset, (c) the containerized HCP cluster can be easily upgraded, (d) it is easy to bring the physical cluster in its initial state if desired. The deployment of a containerized HPC system includes two stages – (1) deployment of LXC containers to host the HPC cluster and (2) installation of HPC components. The first step usually will be performed by the HPC provider, considering their security concerns. The next step should be essentially the same for all providers in EGI-ACE. A proof-of-concept will be created for capturing an existing physical HPC cluster as LXC containers template (performing physical to container conversion), allowing to backup or replicate the cluster, or to repurpose the hardware while continuing to use the original HPC system.

[19] https://www.openstack.org/use-cases/bare-metal/

4 HPC security guidelines

The security in the HPC ecosystem should encompass three main categories: data confidentiality, system integrity and system availability. Security requirements should be integrated already in the HPC system planning/design, while it will be more difficult to integrate them later. For the network design, at least the user and management network should be separated, network topology should be documented well, and it facilitates debugging in the case of security and other problems. For provisioning and configuration of the nodes use configuration management tools as they provide consistency and automation of the service configuration, traceability of configuration changes, efficient change management, control over running processes and permissions over the files, configuration backup and documentation. Configuration of the nodes should include OS hardening and disabling unnecessary services. Use central access and identity management software to manage and control users' access to services and use a central remote logging service to store logs. For services that require authentication, implement Multi-Factor Authentication, if possible, monitor network activity and data access, enable auditing and check the integrity of the trusted computing base (configuration files, binaries, kernel modules and other critical files). Define vulnerability handling and incident response procedures.
Further reading:

- AARC Project: https://aarc-project.eu/policies/policy-development-kit/
- WISE: https://wise-community.org/published_documents/
- Service Management System processes: https://documents.egi.eu/public/ShowDocument?docid=3807
- EGI-CSIRT: https://csirt.egi.eu

5 Operational integration

Having access using federated authentication and authorisation is the first step to integrating HPC centres in the EOSC Compute Platform. Operational integration enhances the federation from the operational perspective: they provide insights on the capacity consumption by users, simplifies user interaction via a helpdesk, or provide service availability and reliability metrics.

5.1 Accounting

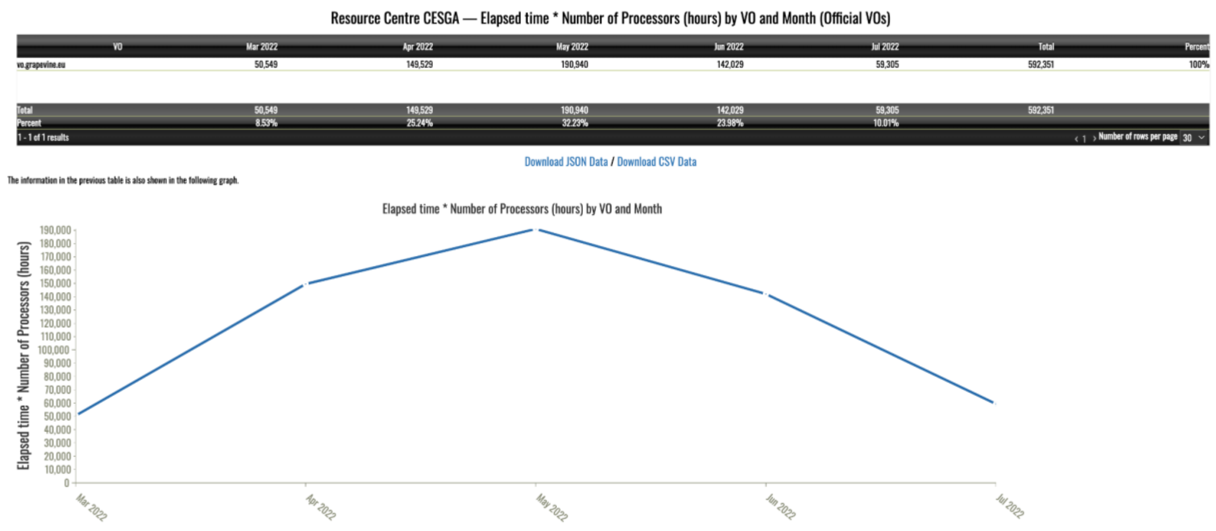
The Accounting service, provided by APEL[20], parses the logs from many computation platforms such as SLURM, OpenStack, OpenNebula and others, sends them to APEL, where they will be summarised, and transformed and sent to the Accounting Portal for display. In the case of some large organisations, such as WLCG or OSG, those can do the summarization and send their records already processed.

ARGO Messaging System (AMS)[21] is used both for publication of non-summarized or summarised data, and for sending the final data to the Portal. As part of the task, we evaluated the scenario of using the APEL software to publish an existing SLURM-based workload, in this case the parsing is done directly by the apel-parsers component instead of second-party parsers like cASO, which in many cases were developed to address the fragmentation existing in data representation between the existing Cloud provider management software.

We installed apel-parser successfully and configured it to point to our SLURM-powered computation node. The publication is done at fixed intervals, and in case of errors or gaps in the publication there are special commands that allow selective republication of the accounting data, although these must be used in tandem with APEL, so that old data is replaced seamlessly with the corrected/more complete data.

In our case, the publication was successful, and in brief, the existing accounting solutions seem to work correctly for our use case in a SLURM computing node.

[20] https://docs.egi.eu/internal/accounting/
[21] https://docs.egi.eu/internal/messaging/


Resource Centre CESGA — Elapsed time * Number of Processors (hours) by VO and Month (Official VOs)

VO	Mar 2022	Apr 2022	May 2022	Jun 2022	Jul 2022	Total	Percent
vo.grapevine.eu	50,549	149,529	190,940	142,029	59,305	592,351	100%
Total	50,549	149,529	190,940	142,029	59,305	592,351	
Percent	8.53%	25.24%	32.23%	23.98%	10.01%		

1 - 1 of 1 results ‹ 1 › Number of rows per page 30

Download JSON Data / Download CSV Data

The information in the previous table is also shown in the following graph.

Figure 3: Accounting for HPC usage at CESGA[22]

5.2 Monitoring

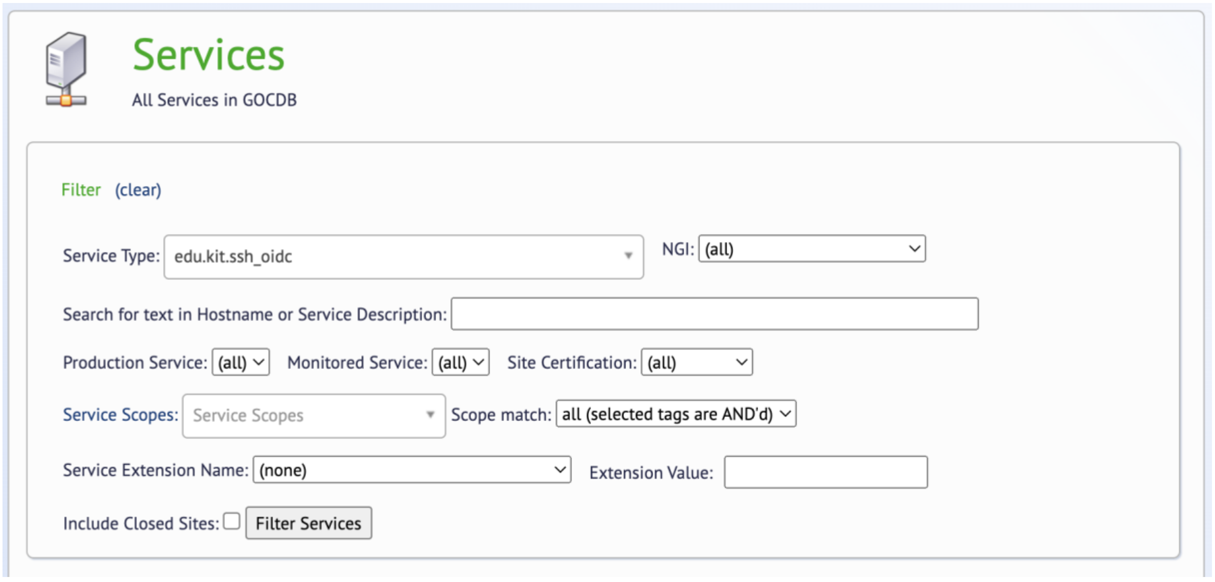
Monitoring is key to gain insights into the status of an infrastructure. HPC providers in EGI-ACE already count with detailed monitoring to ensure the regular operation of the systems and detection of internal issues, thus this kind of monitoring will not be covered by the pilots. Instead, we will focus on ensuring the access interfaces of the EOSC Compute Platform are operational and available. ARGO[23] - the monitoring solution of EGI - can be extended with new probes and as part of the pilots we will implement those missing. For accessing the HPC providers using HTC middleware there are already an extensive set of probes available. Similarly, for HPC as a Service, which can be monitored using the existing EGI Cloud related monitoring probes. In the case of ssh-oidc based access, the probes can leverage the existing monitoring credentials in ARGO to obtain Check-in tokens that can be used to access the system.

The EGI Configuration Database (GOCDB)[24] is a central registry that records topology information about all sites participating in the EGI infrastructure. As the HPC centres are already part of the infrastructure, there exists an entry in the GOCDB that lists all the relevant endpoints made available to users. New HPC providers entering the infrastructure need to follow EGI's PROC09[25] (Resource Centre Registration and Certification) that covers all the steps required for registering and certifying new Resource Centres (sites) in the EGI infrastructure. Certified Resource Centres make resources available to international user communities and guarantee a minimum quality of service of the resources (currently expressed in terms of monthly availability and reliability as obtained by the monitoring).

---

[22] https://accounting.egi.eu/egi/elap_processors/SITE/DATE/2021/7/2022/7/custom-vo.grapevine.eu/onlyinfrajobs/
[23] https://docs.egi.eu/internal/monitoring/
[24] https://docs.egi.eu/internal/configuration-database/
[25]https://confluence.egi.eu/display/EGIPP/PROC09+Resource+Centre+Registration+and+Certification

GOCDB collects information about all the service endpoints available at each site. These endpoints are used by ARGO to automatically monitor and calculate the availability and reliability metrics depending on their service type (service types are pieces of software while service endpoints are a particular instance of that software running in a certain context). A new service type for SSH-OIDC is now available and the probes' implementation is in progress for enabling complete monitoring of SSH-OIDC endpoints.



Services

All Services in GOCDB

Filter (clear)

Service Type: edu.kit.ssh_oidc NGI: (all)

Search for text in Hostname or Service Description:

Production Service: (all) Monitored Service: (all) Site Certification: (all)

Service Scopes: Service Scopes Scope match: all (selected tags are AND'd)

Service Extension Name: (none) Extension Value:

Include Closed Sites: ☐ Filter Services

Figure 4: New service type available in GOCDB

The idea behind the SSH-OIDC probes being developed is to check the functionality of the various aspects of the connection endpoint: connectivity, endpoint health, and ensuring the correct assignment of user/groups to federated users. Since SSH-OIDC is a brand-new endpoint with novel features, making sure that features are working as expected becomes important.

When a user logs-in via SSH-OIDC, all Check-in metadata is also carried with the user, allowing a user to use its previously defined groups and memberships inside the service the user connected to. Considering ARGO is another user logging into a server to check its health, these checks become practical. Just by connecting to the endpoint with SSH-OIDC and comparing the memberships obtained with a known good list, a probe can verify end to end functionality of an SSH-OIDC endpoint.

5.3 Helpdesk

The EGI Helpdesk[26] is the entry point and ticketing system/request tracker for issues concerning EGI services. New service providers can integrate into the Helpdesk by creating a dedicated support topic listed on the Helpdesk user interface (for users to ask questions or raise issues directly to the provider). Resource centres registered in the Configuration

[26] https://docs.egi.eu/internal/helpdesk/

Database will be automatically available in the Helpdesk for routing tickets as needed so no extra integration step is needed to be part of the Helpdesk.

6 Data Transfers

This section documents the different data transfers methods tested for the pilots of the HPC integration.

While many HPC centres offer File Transfer Services (FTS) such as UFTP or GridFTP for staging large dataset in and out of the centre, these FTS appliances depend on properly configured WAN connections, NAT rulesets, traffic shapers, and fast storage nodes, especially for connections over 40Gb/s. In order to validate the performance of these appliances, GÉANT operates a dedicated network testbed service (GTS) which permits network researchers to test end-to-end network performance utilising the GÉANT core services at 10-100Gb/s and above (as national links are upgraded). The GTS allows access to both virtual environments and bare-metal Data Transfer Nodes (DTN[27]) where users can employ an array of provided network testing applications, or even configure their own services, such as XrootD used in big-data science.

6.1 DataHub

EGI DataHub[28] is a high-performance data management solution that offers unified data access across globally distributed environments and multiple types of underlying storage. Users can transparently access, store, process and publish data backed by storage providers worldwide. Replica management functionality enables data to be replicated across providers on-demand or use Quality of Service rules to manage file replica distribution and redundancy.

DataHub is based on Onedata technology[29] and consists of the following main concepts:

- Spaces. All data in Onedata is organised into Spaces which are volumes containing an arbitrary directory and file hierarchy, supported by one or more storage providers.
- Zones. Zones are created by deploying a Onezone service. Onezone takes care of user authentication and authorization. EGI DataHub is an example of a zone, and is integrated with EGI Check-in.
- Providers. Each Zone comprises a network of providers who make storage resources available to users. The Oneprovider service is typically deployed at each site near the storage resources and enables storage to be registered to Onezone. Storage systems such as Ceph as well as those supporting S3, Swift or POSIX are supported.

From the users perspective, Onedata provides a set of interfaces for easy access and management of data distributed among the storage providers resources. Specifically, data can be accessed in four ways:

- Web interface, a web-based, graphical user interface to manage spaces, control access rights and manage the user account.
- Command line interface. The oneclient command-line tool gives users the ability to mount spaces and provides POSIX-like access to data. Oneclient is based on Fuse

[27] https://wiki.geant.org/display/NETDEV/DTN
[28] https://docs.egi.eu/users/data/management/datahub/
[29] https://onedata.org

and since version 18.02.2 and 19.02.0-rc1 can be installed using Anaconda from the official Onedata conda repository. This allows users working on a HPC system to mount Onedata virtual filesystems in their own home directories and access and process data directly from the console. The only requirement to be satisfied to make the Oneclient work is to have fuse/fuse3 installed on the system and be able to access the fusermount tool.

- RESTful APIs. Onedata exposes both CDMI and its own API which provides the ability to manage spaces.
- Since version 18.02.2, it is also possible to access data managed by Onedata directly from Python, using the OnedataFS Python library. In this way, Onedata can be easily integrated with Jupyter Notebooks via the OnedataFS Python library and the OnedataFS-Jupyter plugin, thus allowing users to store the notebooks directly in Onedata data spaces and access Onedata spaces from within the notebooks.

EGI DataHub supports multiple access policies including unauthenticated open access and access restricted to members of a VO.

6.2 Parallel rsync (prsync)

Parallel rsync is a part of the common parallel ssh (pssh) suite available on many distributions. Unlike other technologies within this section, its goal is not to copy between two sites as fast as possible, but to copy a file to multiple sites as fast as possible. The application of this technology could be an edge case, for instance to support extrinsic uncertainty quantification across multiple HPC systems. However, it is not foreseen to be a primary transport protocol.

This should also not be confused with the use of the GNU parallel command which can be used with rsync. However, when attempting to do this, no significant performance gain is observed over sequential single runs of rsync for large files.

6.3 rclone

Rclone is an open source and MIT licensed command line application to manage cloud storage. The application supports over 40 cloud storage providers (including but not limited to Amazon S3, Ceph, HDFS, Dropbox) and the standard protocols like HTTP, FTP and SFTP. This flexibility allows Rclone to interface with almost any enterprise and consumer cloud storage system and transfer large amounts of data with ease.

The power of the Rclone doesn't only come from its wide connectivity options. Rclone can perform a variety of tasks over these connections. It can copy, move, sync and verify files in either direction, and can perform these tasks with multiple connections and threads. Moreover, if the connection is disrupted during any large job, Rclone can resume from the last good file transferred instead of starting over. These features allow Rclone to be used in both manual and automated scenarios with confidence.

Rclone requires Oauth authentication from most popular providers, yet its integrated configuration manager allows these remotes to be added with minimal effort. Rclone supports an arbitrary number of remote connections, allowing many accounts to be connected to a single Rclone installation. Hence a person or a large research group can connect all data resources to a single Rclone instance and manage all of them with ease.

The application can be found in package repositories of most Linux distributions. Also, it can be downloaded in a variety of ways. The homepage of Rclone is https://rclone.org and it can be directly downloaded from https://rclone.org/downloads. A list of supported providers and per provider capabilities can be found at https://rclone.org/overview/.

6.4 Aspera

Aspera is a software specialised in the movement of big data files over long distances. Belongs to IBM and is based on FASP (Fast Adaptive and Secure Protocol) technology. Similar to the connectionless UDP protocol, FASP does not expect any feedback on every packet sent. Only the packets marked as really lost must be requested again by the recipient. As a result, it does not suffer as much loss of throughput as TCP does on networks with high latency or high packet loss.

Aspera requires a software license. In Spain, research institutions connected to the academic network NREN RedIris can take advantage of the institution license to use the software to move data. Currently, RedIris has a license to use up to 10Gbit/s bandwidth (Aspera is licensed in various ways, one of them is based on the bandwidth usage). RedIris additionally provides a safe storage system to move data to and from remote locations than once there can be moved faster using the high speed RedIris academic network.

6.5 HSCP and UDR

HSCP[30] and UDR[31] (as well as similar protocols such as RB-UDP[32] and Tsunami[33]) are based on reliable UDP (UDT[34]) which in principle should markedly improve network performance, particularly over very long baselines or on noisy connections subject to significant dropouts. UDP requires significantly less handshaking but on its own is impacted by lossy connections; UDT overcomes this by providing a reliable mechanism for a server to request only those packets which have not been received. In addition, the technology allows for a high level of parallelisation.

UDR is a wrapper around rsync that enables the underlying rsync to use UDT, in a similar way to the more complete and feature-rich Aspera mentioned above. While it is currently not supported, and indeed the author recommends investigating the use of congestion control algorithms to improve performance, there have been some recent attempts to revive the use of this protocol. For example, as detailed in Chase Wright's blog[35], one user found that for Wide Area transfers of large files over dual 1Gbps connections, standard rsync achieved average bandwidths of 100Mbps, while using UDR this increased to 250-300 Mbps. However, this improvement will typically only be visible on long WAN transfers of large files.

[30] https://sourceforge.net/projects/hscp/
[31] https://github.com/martinetd/UDR
[32] https://www.evl.uic.edu/cavern/RBUDP/Reliable%20Blast%20UDP.html
[33] http://tsunami-udp.sourceforge.net/
[34] https://udt.sourceforge.io/
[35] https://chasewright.com/remote-rsync-over-high-speed-but-latent-wan-udr-udt/

Movement of many smaller files will require a different approach; or that the small source files be batched into larger files using some suitable tool. UDR is open source released under the Apache 2.0 license.

HSCP was developed by the National Institutes of Natural Sciences, Okazaki Research Facilities and like UDR does not seem to be supported (although this is not explicitly stated on the home page). Unlike UDR, it is not a wrapper around an existing tool but has been developed from scratch, with a higher level of security. It also provides a secure, authenticated, reliable control channel for command flow but an insecure data channel for bulk transfer as shown in figure 5. However, evidence of its performance in real situations is scarce.

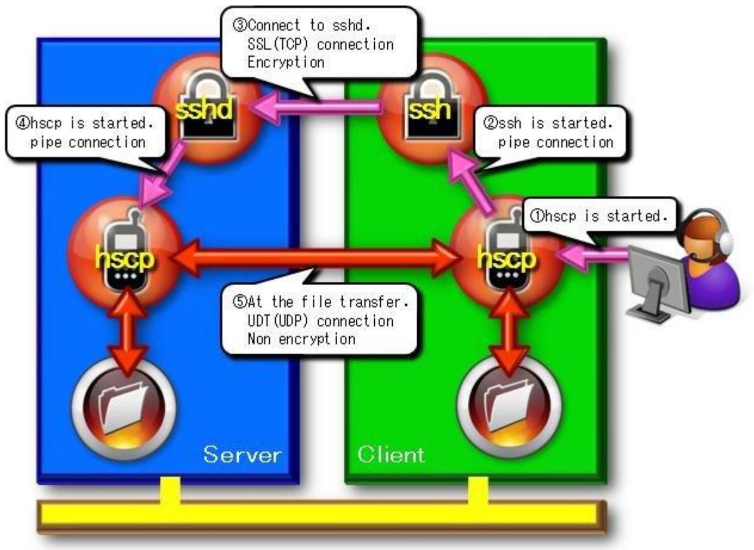


Figure 5: Secure and Insecure Channels in HSCP

Note that while UDT offers significant benefits for many use cases, and there was a rash of development into protocols based on this in the early to mid 2010's, realisation of that benefit in real world situations was difficult, and, combined with general improvements in networking hardware and software, has led to the continued use of TCP based protocols.

7 Application support

All pilots included in EGI-ACE plan to execute their workloads using containers. The preferred tool for supporting these containers is udocker.

7.1 udocker

udocker is a user-oriented tool to execute containers in user space without requiring root privileges. udocker enables basic download and execution of containers by non-privileged users in Linux systems. It can be used to access and execute docker containers in batch systems and interactive clusters that are managed by other entities such as grid infrastructures, HPC clusters or other externally managed batch or interactive systems. udocker is a wrapper around several tools and technologies to pull container images and execute them with minimal functionality. Since root privileges are not involved, most operations that require privileges will not work under udocker. This limitation does not affect user applications. udocker itself is written in Python and has a minimal set of dependencies so that it can be executed in a wide range of Linux systems.

Since udocker does not require any type of privileges nor the deployment of additional software by system administrators, it can be easily deployed by the end user in HPC clusters. Users can download udocker themselves and install the tool in their home directory.

Advantages of udocker:

- Can be deployed by the end-user
- Does not require privileges for installation
- Does not require privileges for execution
- Does not require compilation, just download udocker
- Encapsulates several tools and execution methods
- Includes the required tools already statically compiled to work across systems
- Provides a docker like command line interface
- Supports a subset of docker commands: search, pull, import, export, load, save, login, logout, create and run
- Allows loading of docker and OCI containers
- Supports NVIDIA GPGPU applications
- Can execute in systems and environments where Linux namespaces support is unavailable
- Runs both on new and older Linux distributions including CentOS 6, CentOS 7, CentOS 8, Ubuntu 14, Ubuntu 16, Ubuntu 18, Ubuntu 20, Ubuntu 21, Alpine, Fedora, etc

The basic flow for udocker usage is:

1. The user downloads udocker to its home directory and executes it
2. Upon the first execution udocker will download additional tools
3. Container images can be fetched from Docker Hub with `udocker pull`
4. Containers can be created from the images with `udocker create`
5. Containers can then be executed with `udocker run`

In addition and similar to other container tools:

A. Containers can be loaded from file with udocker load -i
B. Tarballs can be imported with udocker import

Figure 6 provides an outline of using udocker to execute containers. Containers can be downloaded from a repository with pull, alternatively they can be loaded or imported from files. The container layers and metadata corresponding to the images are stored in the user home directory under $HOME/.udocker/layers. The content of these layers can then be extracted (flattening) to create a single file system tree for execution that is also placed under $HOME/.udocker/containers. Code within the extracted directory tree can then be executed using one of the supported execution modes.

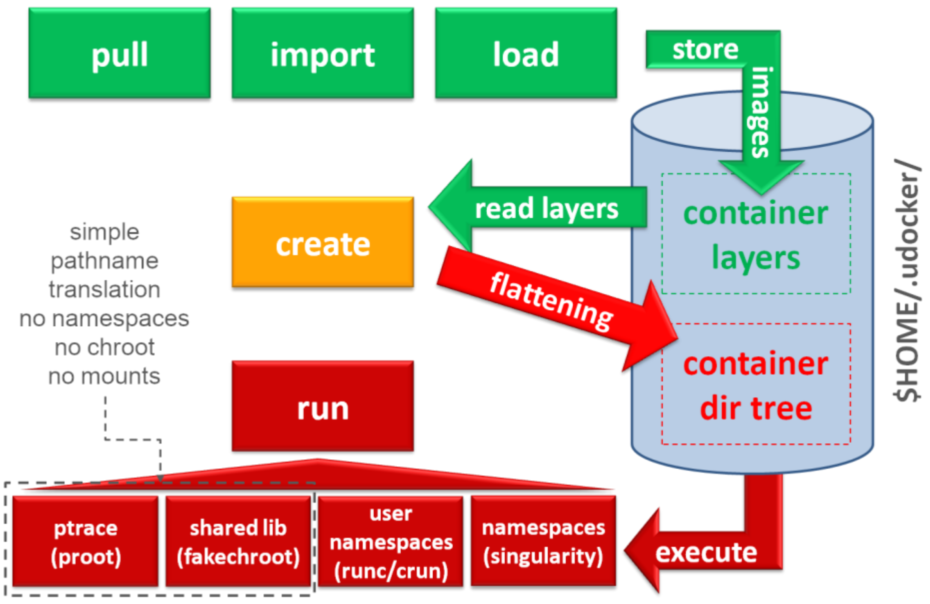


Figure 6: outline of containers execution with udocker

Installation can be performed using several methods. The installation from a release tarball is the recommended approach. Installation instructions for sites without outbound connectivity and additional information for shared installations are also available from the installation manual. Detailed installation instructions are available at: https://indigo-dc.github.io/udocker/installation_manual.html

The user manual including command references and usage recommendations and considerations is available at: https://indigo-dc.github.io/udocker/user_manual.html. Information on the use of udocker with GPUs and MPI is also available in the user manual.

udocker “executes” the containers by simply providing a chroot like environment to the extracted container. udocker is meant to integrate several technologies and approaches hence providing an integrated environment that offers several execution options. For further information on selecting execution modes see the user manual section on udocker setup.

Table 1: Supported execution modes

Execution mode	Tools	Description
P1	PRoot	Accelerated mode using seccomp
P2	PRoot	Same as P1 without seccomp accelerated mode
F1	Fakechroot	Execution through loader invocation
F2	Fakechroot	Same as F1 with modified loader to prevent loading from host
F3	Fakechroot	Execution with fixed ELF headers for libraries and executables
F4	Fakechroot	Same as F3 plus dynamical fixing of ELF headers
R1	runc or crun	Rootless user mode namespaces
R2	runc or crun	Same as R1 plus P1 for software installation
R3	runc or crun	Same as R1 plus P2 for software installation
S1	Singularity	Uses singularity if available in the host

The Pn modes are the most generic and are used by default. The Fn modes are the fastest but require shared libraries matching the container libc, these libraries are provided with udocker for the most popular distributions. Both the Pn and Fn modes do not require the use of Linux namespaces. These modes intercept calls to the system and perform pathname translation to mimic a chroot environment.

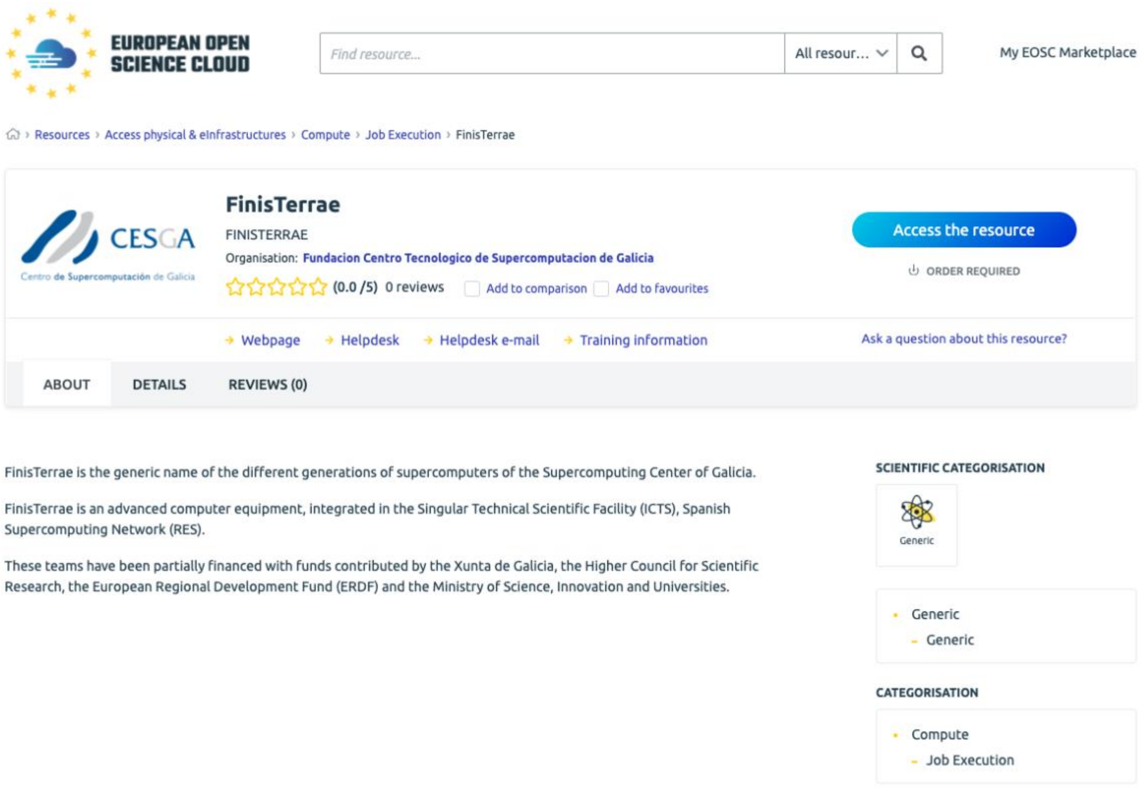
Most of the required tools are provided with udocker already compiled and ready for use. However, the Sn mode requires the installation of the corresponding tool in the host system. The Rn modes require "user namespaces" support enabled in the Linux kernel.

Relevant links:

- Source code repository: https://github.com/indigo-dc/udocker
- Releases: https://github.com/indigo-dc/udocker/releases
- Complete documentation: https://indigo-dc.github.io/udocker

8 EOSC Marketplace integration

The HPC providers participating in EGI-ACE should be accessible through the EOSC Marketplace. Individual providers may be registered independently in the marketplace as shown in figure 7 for CESGA:



FinisTerrae

FINISTERRAE

Organisation: Fundacion Centro Tecnologico de Supercomputacion de Galicia

(0.0 /5) 0 reviews

FinisTerrae is the generic name of the different generations of supercomputers of the Supercomputing Center of Galicia.

FinisTerrae is an advanced computer equipment, integrated in the Singular Technical Scientific Facility (ICTS), Spanish Supercomputing Network (RES).

These teams have been partially financed with funds contributed by the Xunta de Galicia, the Higher Council for Scientific Research, the European Regional Development Fund (ERDF) and the Ministry of Science, Innovation and Universities.

- Generic
  - Generic

- Compute
  - Job Execution

Figure 7: FinisTerrae - CESGA HPC - registered in EOSC Marketplace

An "EGI Accelerated Cloud" entry will be created in the EOSC Marketplace for offering a common entrypoint for the different HPC providers in EGI-ACE. This service will offer cloud resources in combination with HPC capabilities, to cover workflows, where collocated cloud and HPC resources are used to support a data space, e.g., long-running services, are hosted as cloud VM, and jobs with high computational needs are spawned to the collocated HPC as needed. In this configuration, the access control to the HPC system can be better controlled and be limited to the cloud resources associated with it, thus avoiding policy restrictions that do not allow the wide opening of HPC resources to the internet. The service will be initially supported by four providers: LIP/INCD (PT), CESGA (ES), TUBITAK (TR), IICT-BAS (BG), and will leverage the engagement process and resource allocation processes defined for EGI-ACE and described in D2.5 (EOSC Compute Platform Handbook)[36]. We intend to engage with additional HPC providers from the EGI network and beyond and sign them up as suppliers behind the 'EGI Accelerated Cloud' service entry.

[36] https://documents.egi.eu/document/3813

9 Conclusions

Thanks to the four EGI-ACE HPC pilots, the EOSC Compute Platform will be expanded to cover HPC providers to support computing workflows that make combined use of HTC, HPC and Cloud. In the first half of the piloting activity, the focus was on providing access to the HPC systems using federated authentication and providing the use cases with the tools to run their containerised workload across different providers. In this document we report the various integration options and the main areas of work the pilots of the project have developed until M18 of the project (June 2022). With the piloting activities now completed, EGI will create a new entry in the EOSC marketplace named “EGI Accelerated Cloud” that will allow users to request access to HPC providers as part of the EOSC Compute Platform. During the remaining period of EGI-ACE our focus will shift to populate the “EGI Accelerated Cloud” service with multiple suppliers from and beyond the EGI Federation and serving new use cases require HPC services in the EOSC Compute Platform.