

HPC Software as a Service: A Flexible Approach to Data Logistics

François Bodin^{1,2}[0000–0002–0928–2564], Laurent Morin^{1,2}[0009–0001–4630–1234],
and Germaine Nyastikor^{1,2}[0009–0009–4444–2672]

¹ Université de Rennes

² Irisa, Campus de Beaulieu, 35042 Rennes, France <http://www.irisa.fr>
{yawa-germaine.nyastikor, laurent.morin, francois.bodin}@irisa.fr

1 Introduction

The use of High-Performance Computing (HPC) facilities is rapidly evolving. Initially designed to address capacity problems, these facilities are now being leveraged to process the vast amounts of data generated by scientific instruments (e.g., satellites, telescopes). While cloud facilities are designed to provide scalable processing capacity, the operational constraints of HPC centers prevent deploying applications in the same manner. Among them, cybersecurity concerns, tied to the HPC center's status (e.g., France's "zone à régime restrictif" or ZRR [12]), represent a significant hurdle.

The need to support cross-facility workflows [9], which involves heterogeneous tasks (IA, compute, HPDA) distributed across multiple infrastructures (the digital continuum: HPC centers, data centers, edge, and cloud facilities), is another critical aspect of this evolution. These workflows are relevant to the collaborative system of systems topic, as each system has its own operational *modus operandi* (e.g., access management, job and resource management).

In this paper, we propose an approach to deploy scientific applications as a service (SaaS) on HPC systems. This proposal aims to design a trade-off that can be deployed on the majority of systems. To do so, we rely on task runners, rather than on workflow managers. The runners must provide two capabilities: first the execution of jobs on the host system (application service); second a mechanisms for implementing data logistics.

A data logistic ecosystem refers to a distributed system in which nodes are first used to expose system's topology for efficient data transfers, second is based on decentralized service models with a local autonomy, and finally includes the support a federated security mechanisms [3]. This paper aims at providing some of its essential blocs.

It is essential to note that **regulatory considerations are significant roadblocks**, and desirable technical solutions are not viable if they cannot be deployed in accordance with regulations. We propose an approach compatible with current HPC center regulations.

The proposed architecture is designed to enable scientific application developers to deploy their code as a SaaS, making it available to the community.

This is exemplified by the DDF Pipeline [6,23], which processes Low-Frequency Array (LOFAR) radio telescope measurement Sets (soon needed by the Square Kilometre Array telescope SKA). This pipeline is developed by few developers, requires frequent updates, and may be used by a large astrophysics community.

The structure of this document is as follows. Section 2 presents related work, focusing on existing approaches for scientific workflows deployment and data logistics. Section 3 introduces the foundational concepts underpinning our approach, including systems-of-systems and identity federation. Section 4 details the Ephemeral Buffer mechanism for data logistics, with Section 4.1 outlining its high-level properties and Section 4.2 describing its use in cross-facility workflows. Section 5 discusses the integration of Application Services using our proposed architecture. Section 6 illustrates a practical use case through an experiment involving the DDF Pipeline and the SKA project. Finally, Section 7 concludes the paper by summarizing our contributions and highlighting ongoing evaluations related to cybersecurity constraints.

2 Related Works

Scientific workflows are characterized by the large volumes of data they produce and consume. Existing tools for managing or executing these workflows often target the scientific community as users, rather than as developers of these tools.

In the Workflow Community Summit 2024 report [24], the focus is on the future of workflows in a digital landscape that is increasingly driven by AI. This report provides a set of recommendations and research directions, including the synergy between AI and HPC, the rise of cross-facility workflows, and data-driven HPC workflow dynamics. These issues are directly addressed in this work.

To support these emerging needs, we adopt the concept of the digital continuum, which we define, in our context, as a collaborative system-of-systems (SoS). The systems under consideration operate under strict cybersecurity regulations [12]. We are not considering HPC cloud solutions such as those described in [17,7] nor solutions such as the EasyOP cloud service platform [22]. While many local solutions can be used on each systems, our European ecosystem makes it nearly impossible to obtain a common solution for a shared distributed storage, a common job manager, unified containers, etc.

According to the IEEE, a system is "a collection of components organized to accomplish a specific function or set of functions." In [15], Maier proposed a definition of collaborative systems of systems, and described principles intended to guide system designers in architecting complex systems. To distinguish a simple system from a collaborative SoS, Maier identifies two key properties:

Operational Independence of the Components: Each component must be capable of operating independently to fulfill its own goals, even while contributing to the overarching system purpose.

Managerial Independence of the Components: Each component must be individually managed, retaining autonomy even as it participates in the larger system's objectives.

According to Maier, communication between components is a critical element in SoS. Since each component operates and is managed independently, the communication protocols and structures between components become essential to achieving the overall system objectives, which in our case is to run cross-facilities workflows.

The SaaS architecture [1,2] proposed in this paper has been designed based on two guiding principles.

The first principle is described in the paper "End-To-End Arguments in System Design" [21], published in 1984. This work addresses the question of where certain functionalities should be implemented within a system to achieve efficient usage. It argues that some functions are best implemented at the application level rather than within the lower layers of the system.

The second principle is presented in [4]. In this paper, Micah Beck proposes a conceptual model for designing systems that support a diversity of applications and implementations. Building upon the end-to-end argument [21], this work argues that to support different applications and allow multiple independent implementations, the spanning layer must remain simple and flexible.

3 Assumptions and Foundational Concepts

The proposed approach aims to find a trade-off between Managed Service Providers (MSPs) such as FlatSim [10] and full SaaS solutions as defined in [2]. We consider users as scientists willing to run a cross-facility workflow, and maintainers as scientists providing access and typically developing applications needed by the workflow. We base our approach on the following assumptions:

Global Identity: Users and maintainers are authenticated within a federation recognized across all infrastructures required to run their workflows.

Maintainer Resource Access Rights: Maintainers have sufficient rights to execute their applications on at least one of the HPC systems.

Accounting and accountability: Maintainers are responsible for tracking and preventing abuse of resource usage. Although this rule may seem arbitrary, it is mandated by HPC resource usage regulations that all jobs shall be executed under the responsibility of the maintainer, who remains accountable for any misuse, cybersecurity breaches, and other incidents.

Job Pull Mode: Users are not allowed in any circumstance to directly initiate a job execution on target systems. As a consequence, only a tool controlled by the application maintainer can pull a job and decide to execute it, thus maintaining the chain of responsibility.

Data Transfers: Data transfers shall require a limited and controlled access to the storage systems of the target machines.

Micro-service architecture: *Application Services* as well as their associated tools shall be designed as simple, modular, and composable components. This approach, promoted in legacy Unix systems [19] and used in recent web-oriented architectures, favor robustness and auditability of critical systems. It shall be leveraged by the usage of standard like REST APIs.

3.1 High level example of a cross-facility workflow

Consider the example illustrated in Figure 1. A user, U1, wishes to execute a workflow spanning infrastructures S1, S2, and S3. S3 is a data center where U1 has access to the necessary input datasets and the rights to publish resulting data products. S1 and S2 are independently managed HPC centers (e.g., IDRIS and TGCC in France).

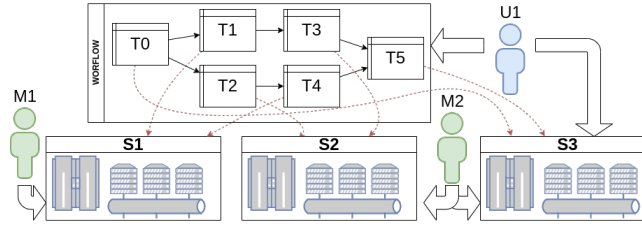


Fig. 1: Workflow execution example using three infrastructures.

Maintainer M1 has published an application capable of running tasks T1 and T4, while maintainer M2 has deployed the applications needed for tasks T2 and T3. Both M1 and M2 have made their applications available as services without requiring direct user accounts on S1 or S2. The workflow manager used is chosen by U1 (like NiFi[25], Galaxy[26], or any compatible technology).

To execute the workflow, U1 must be allowed to transfer the relevant input data to S1 and S2. Moreover, S1 and S2 must be able to automatically transfer data between themselves and with S3 as needed. This reflects a typical situation in scientific domains, where data is accessed from a data center (e.g., Data Terra), processed on an HPC system, and then sent to post-processing infrastructures.

3.2 Application Services based on Ephemeral Buffers

In our approach, the cross-facility execution of a scientific application is defined at high level and is considered in a federation of resources bonded by a global identity infrastructure. We define two new abstractions, the first is based on the concept of *Ephemeral Buffers* for operating temporary data transfers across infrastructures, and the second on *Application services* for the deployment of micro-service based workload within infrastructures.

For both, we minimize the deployment requirements, only the one needed for interoperability while ensuring the regulation, cybersecurity, and accountability. In particular, *Ephemeral Buffers* provides the mechanism to exchange data for anyone in the federation, but in a traceable way and for a restricted time span. It is combined with the *Application Services* interface providing the ability to declare micro-services targeting compute-demanding or data-intensive tasks on one side, and on the other side eventually run the tasks submitted in a voluntary way for anyone in the federation.

3.3 Trust and accountability

In our context, the introduction of a global identity infrastructure is a major requirement in the creation of a trusted federation. This component can rely on well established standard like SAML2 or OpenID Connect[5,11,20], heavily used nowadays to secure identification of most Internet services. Not described in the current proposal, these standards can also be used to ensure the integrity of cybersecurity sensible data with a cipher-based signature mechanism. Based on this common base, we propose to build a chain of trust attached to both the *Ephemeral Buffers* and *Application Services* mechanisms. The high availability of these services is critical to the infrastructure.

The confidentiality of data and application systems are out of scope of this proposal. The data protections can be ensured when needed by standard cryptographic protocols with a dedicated key management systems. The confidentiality of applications need to be handled at the micro-service level.

On the other hand, the metadata (i.e. connections between systems and users) are supposed to be public to the federation, and their integrity ensured by a signature system at the federation level. The capability of the solution to offer an end-to-end traceability over the execution of workflows defines the **trust zone** of the system. Metadata availability must be provided at runtime for every identified user in the federation (but limited to identifiers, owner, and location). Long term metadata availability is the charge of infrastructure providers at the level required by their own regulation.

4 Ephemeral Buffers for Data Logistics

We propose an approach to operate data logistics around the concept of *Ephemeral Buffers* (EB) [16] offering a streamlined mechanism for data transfer. This approach enables a task to write data into a buffer of a predefined maximum size and for a predefined time on a remote system. Tasks on the receiving system are then responsible for transferring the data from the buffer into the appropriate workspace.

The rationale behind this approach is to simplify authentication processes and avoid complex cybersecurity challenges. By limiting direct access to remote storage and decoupling authentication from individual workflows, *Ephemeral Buffers* provide a secure yet flexible solution for managing data logistics.

4.1 Ephemeral Buffers: High-Level Principles

Ephemeral Buffers (EB) are characterized by the following key properties:

- **System-Dependent Maximum Size:** Each buffer has a predefined maximum size determined by the system configuration, which cannot be modified by users. If the data to be transferred exceeds this size, additional buffers shall be allocated.

- **Time-Limited Availability:** Each buffer has a system-defined lifespan. Once this lifetime expires, the buffer is automatically destroyed without prior notification, ensuring efficient resource management and minimizing the risk of stale or unused data persisting in the system.
- **Finite Availability:** The total number of buffers, and the overall buffer size are limited and system-dependent, preventing over-allocation and promoting efficient resource utilization.
- **Unique Identifier (UID):** Each buffer is assigned a unique identifier, enabling precise tracking and management across the system.
- **Access Control Metadata:** Ensures secure access to buffers. Possible implementations include tokens from an Authentication and Authorization Infrastructure (AAI) or a POSIX Access Control List (ACL) supported by an external mechanism.
- **Annotation Tags:** A size-limited list of arbitrary UTF-8 strings, writable only, and initialized at buffer creation time.

4.2 Data Logistics using Ephemeral Buffers

An example use case can be found in the context of the DDF Pipeline for the SKA project. This pipeline processes radio telescope measurement sets (TMS) as input and produces images as output. The computation requires high-performance resources [6].

Figure 2 illustrates a potential workflow utilizing the DDF Pipeline. In this scenario, the originating infrastructure transfers the TMS data to the HPC system using Ephemeral Buffers (EBs). When a new buffer is created, the corresponding DDF Pipeline task on the HPC system is notified. It retrieves the TMS data from the buffer and initiates a compute job to perform the image computation. Once the job completes, the resulting images are transferred to the publishing task using EBs on the destination system.

The UIDs of the EBs are either communicated by the workflow orchestrator during task instantiation, or produced as task results. This enables seamless data transfers and coordination between tasks across infrastructures.

An important property of this use case is that the HPC center must send the resulting images to an external system responsible for cataloguing and visualization. Using EBs, the task running on the HPC system can directly transfer the data to its destination. Crucially, the task executed under the responsibility of the DDF Pipeline maintainer does not require any credentials for the destination system.

4.3 Ephemeral Buffers Usage and deployment

More generally, data logistics are typically planned during workflow orchestration across the various systems. However, Ephemeral Buffers can be created dynamically at any stage of the workflow execution. It can be optimized by the infrastructure provider or adapted by the application itself.

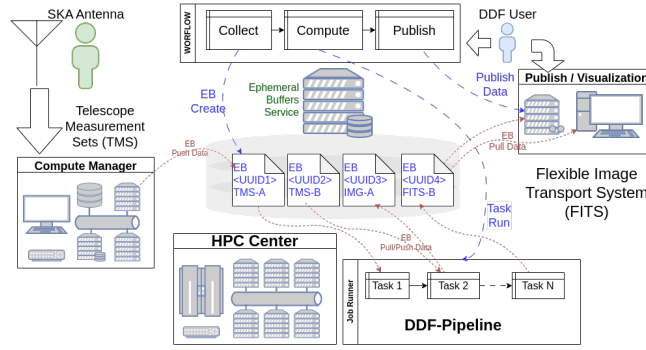


Fig. 2: Use of Ephemeral Buffer in the case of DDF Pipeline.

In the case of an infrastructure owner, new Ephemeral Buffers creations or allocations can be redistributed to the appropriate storage locations depending on the bandwidth, the storage occupancy, or the need for a cost reduction.

At the application level, if the data to be transferred exceeds the capacity of a single buffer, the user may need to chain multiple buffers and implement a write-read-delete pipeline to manage the transfer efficiently. In a large extend, many producer-consumer algorithms can be implemented using Ephemeral Buffers for complex and data-dependent workflows.

The management of the strict time limitation of Ephemeral Buffers can be seen as a challenge. At the workflow level, EBs shall be managed as temporary storage, and a dedicated resiliency mechanism need to be implemented beside the natural execution flow. However these mechanisms are usually present in HPC applications with long running workflow.

At the federation level, the primary objective of an Ephemeral Buffer network service is to provide a trusted layer between the various data providers and compute resource providers. Buffers can be allocated in a traceable way for all members of the federation. But installed very closed to computing resources, it obviously operates as an effective cache mechanism for an efficient workflow deployment.

5 Application Services For Microservice Deployment

We propose to extend our approach with the concept of *Application Services*. An *Application Service* is a simplified mechanism for publishing micro-services and executing tasks with these micro-services on a remote system. This approach can be seen as a simple abstraction of an Managed Service Provider (MSP) using the Software as a Service (SaaS) paradigm for scientific applications.

Like the Ephemeral Buffers, the concept relies on the deployment of a web service as a trusted layer between the scientific application developers, the computing service maintainers, and the scientists users. Three types of elements

compose the web-service: *microservices*, *runtimes*, and *jobs*. An additional *policy* element complete the system for the specification of access control rules.

The high-level approach can be illustrated with the following scenario. A developer would like to offer to the community in the federation a compute-intensive software. First he declares a *micro-service* in the web *Application Service* describing its inputs, outputs, a bootstrapping code, and names the type of its micro-service. He then contacts an application maintainer, someone with enough computing quota in a computer center, and asks him to share his resources to the community. If they agree, the application maintainer declares a *runtime* willing to execute tasks (for instance using a workflow format [8]) from the micro-services (identified by the micro-service type) and installs the associated executing daemon inside the compute center. At this stage the *micro-service* is visible and operational to scientific users in the federation. They can post *jobs* to a given *micro-service* in the web *Application Service*. It can then be distributed to the appropriate *runtime*. It will remain there until the *runtime*'s executing daemon decides to fetch the *jobs* and execute the task in the compute center.

If needed, the scientific developer or the application maintainer can independently add a *policy* descriptor fixing the limits in access or usage of his service. For instance, the developer can restrict the access of some features to a subset of the federation. The application maintainer might also want to introduce on his side some fair usage rules like a limited number of jobs per user and per unit of time.

5.1 Application Services: High-Level Principles

Application Services principles reuse the generic properties already described for the Ephemeral Buffers: the Unique Identifier (UID), the Access Control Metadata, and the Annotation Tags. In complement, they are characterized by the following key properties:

- **Application types:** Application types create the link between a micro-service and a runtime able to execute its jobs. It is declared as a mime type name. While declarative only, conventions could be defined to simplify the execution of standardized micro-services codes on normalized runtimes.
- **Policy properties:** A policy is defined by a scope and a list of rules in the form `<subject>:<action>:[<condition>]` that is for now implementation dependent.
- **Runtime properties:** A Runtime is defined by a unique identifier (UUID), a name, an application type, and a policy UUID.
- **Micro-Service properties:** A Micro-service is defined by a unique identifier (UUID), a signature, a code bootstrap, either an application type or by an explicit runtime UID, and a policy UUID.
- **Micro-service Signature** is composed of the list of input and output names passed by values, and the list of input and output names passed by Ephemeral Buffer UUID.

- **A Job is defined** by a unique identifier (UUID), a micro-service UUID, and a set of input/output values or Ephemeral Buffer UUIDs corresponding to the micro-service signature.

5.2 Application Services Usage and deployment

The deployment and the usage of Application services have to be considered in the global context of a cross-facility execution of a workflow. The overall design is described in Figure 3.

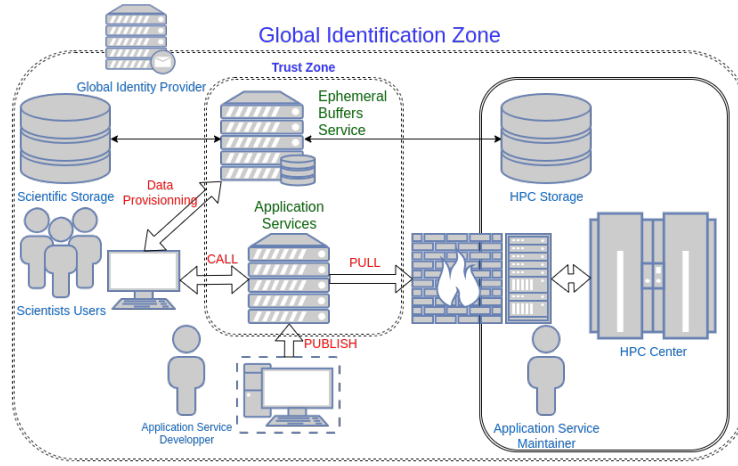


Fig. 3: Application Services operating as a trust zone.

All services deployed within a federation is controlled by a Global Identity provider. Operations issued in any of the services require a prior authentication process providing an appropriate access token. The traceability of all following actions is then guaranteed.

In the federation, both Ephemeral Buffers and Application Services form a trust zone, i.e. a zone where the authenticity of all operations have been verified and recorded. Sensible code and data shall be securely signed whenever possible. The only potentially resource demanding application is the Ephemeral Buffer Service that has to offer significant storage at a reasonable performance. Both services shall offer a strong horizontal scalability, and support high availability deployments.

Based on this architecture, an application service developer, is able to *publish* its software elements as micro-services. At this stage, it only serves as a software publishing mechanism. The scale and granularity of microservices shall be able to vary significantly, from the simple call of a little context free computation, to the complex coordination of multiple highly compute intensive scientific applications.

In the federation, Scientist users can already *provision*, simplify, or optimize data transfers between infrastructures only with the Ephemeral Buffer Services. It is typically useful between the scientific databases distributed worldwide and the HPC center on which they have computing quotas. But they can also *call* micro-services published in the federation, and create jobs ready to be scheduled by an HPC center willing to provide resources for this kind of task.

Part of the federation, application maintainers in HPC center are encouraged to provide resources to the community, declare runtimes to the application services, and pull jobs for their execution. The first and obvious reason for this is that an application developer can also be an application maintainer. In order to integrate its work in a larger collaboration, improve his algorithms, or promote the usage of his scientific work (in particular when seeking certification for published papers), he can share the resources allocated for his project on the HPC center. Another reason could also be to help in the deployment of standardized computations or in the execution of complex workflow.

6 Implementation and First Experiments

A first version of Ephemeral Buffers and Application Services concepts have been fully implemented and deployed as web services with a standard REST OpenAPI [16]. A local Identity and Access Manager based on Keycloak [13] is used to emulate the identity manager of the federation. Both implementation have been designed in a way to ensure an efficient horizontal scalability, and an high availability.

The Ephemeral Buffer service has been deployed on Internet for workflow developers on NiFi, and is currently used to operate data transfers between asynchronous services. The implementation of storage operations is based on backends optimized with low-level native operations (currently an in-memory backend, and a low-latency memory-mapped file storage backend).

Two runtimes execution daemons have been implemented, a first one operating Bash shell scripts, and a second one operating simple workflow descriptors in CWL [8]. These execution daemon examples show that the implementation is small and straightforward in many cases (100 to 200 lines of codes including error management). For each runtime, a simple proof-of-concept application is able to operate the full declaration-allocation-execution chain, including the management of intermediate Ephemeral Buffers.

Using the Bash runtime, a proof-of-concept has been validated for the execution of the two major components of the DDF Pipeline workflow (the killMS and DDFacet applications), a complete run including data transfers with Ephemeral Buffers, is scheduled on the IDRIS High-Performance Computing center.

7 Conclusion

In this paper, we presented a solution for deploying scientific applications across the digital Continuum to support cross-facility workflows, particularly in HPC

centers. We addressed how to build task runners compatible with a workflow language [14] and introduced the concept of Ephemeral Buffers to manage data logistics between tasks.

Our approach seeks to define an architecture that facilitates the use of scientific applications developed by others, without requiring every user to hold accounts on all infrastructures within their digital Continuum. Another key objective is to comply with the strict operational regulations of HPC centers, which are major barriers to adopting solutions commonly used in the Cloud community.

Given that we operate within a collaborative system-of-systems context, we minimized the need for strong inter-infrastructures collaboration. The only required common agreement is the use of a federated identity system recognized by all stakeholders; in the French context, this is the RENATER federation [18].

The proposed solution remains agnostic to the workflow manager (e.g., NiFi, Galaxy, ...).

While the design was co-developed with the IDRIS HPC center, the proposed ideas are still undergoing evaluation from a cybersecurity standpoint.

References

1. Aleem, S., Ahmed, F., Batool, R., Khattak, A.: Empirical investigation of key factors for saas architecture. *IEEE Transactions on Cloud Computing* **9**(3), 1037–1049 (2021). <https://doi.org/10.1109/TCC.2019.2906299>
2. Amazon Web Services: Saas architecture fundamentals (2022), <https://docs.aws.amazon.com/whitepapers/latest/saas-architecture-fundamentals/saas-architecture-fundamentals.html>
3. Asch, M., Bodin, F., Beck, M., Moore, T., Taufer, M., Vilotte, J.: Cybercosm: New foundations for a converged science data ecosystem. *CoRR* **abs/2105.10680** (2021), <https://arxiv.org/abs/2105.10680>
4. Beck, M.: On the hourglass model **62**, 48–57 (2019). <https://doi.org/10.1145/3274770>
5. Cantor, S., Kemp, J., Philpott, R., Maler, E.: Assertions and protocols for the oasis security assertion markup language (saml) v2.0. Technical Report saml-core-2.0-os, OASIS (2005), <http://docs.oasis-open.org/security/saml/v2.0/>
6. Certenais, M., Bodin, F., Morin, L.: A description of the ddf pipeline in the exa-atow context. Deliverable D7.3, Exa-AToW Project (October 2024), <https://exatow.fenix.rudi-univ-rennes1.fr/media/download/907f9c48-0016-4abe-a073-826a98f1db10>, accessed April 2025
7. Church, P.C., Goscinski, A.M., Lefèvre, C.M.: Exposing hpc and sequential applications as services through the development and deployment of a saas cloud. *Future Gener. Comput. Syst.* **43–44**, 24–37 (2015), <https://api.semanticscholar.org/CorpusID:10811865>
8. Crusoe, M.R., Abeln, S., the CWL Community, e.a.: Methods included: Standardizing computational reuse and portability with the common workflow language. *Communications of the ACM* **65**(6), 54–63 (June 2022). <https://doi.org/10.1145/3486897>
9. Ferreira Da Silva, R., Badia, R.M., Bard, D., Foster, I.T., Jha, S., Suter, F.: Frontiers in scientific workflows: Pervasive integration with high-performance computing. *Computer* **57**(8), 36–44 (Aug 2024). <https://doi.org/10.1109/MC.2024.3401542>

10. ForMaTer: Flatsim product manual (2020), <https://formater.pages.in2p3.fr/flatsim/>
11. Hardt, D.: The OAuth 2.0 Authorization Framework. RFC 6749 (Oct 2012). <https://doi.org/10.17487/RFC6749>
12. Harry, G., Detournay, M., Bodin, F.: State of the art on cybersecurity practices. Deliverable D2.1, Exa-AToW Project (May 2024), <https://exatow.fenix.rudi-univ-rennes1.fr/media/download/f70142bf-f69e-4682-8f79-bd538b876d5b>, accessed April 2025
13. Hat, R.: Keycloak. <https://www.keycloak.org/> (2025), version 24.0.0
14. Herrera, C., Deutsch, T., Bodin, F., Martineau, O., Richet, G.: Report on workflows description languages. Deliverable D3.1, Exa-AToW Project (January 2025), <https://exatow.fenix.rudi-univ-rennes1.fr/media/download/bf02c9d8-093f-49f9-b792-168fc3c9559a>, accessed April 2025
15. Maier, M.W.: Architecting principles for systems-of-systems. *Systems Engineering* **1**, 267–284 (1998). [https://doi.org/10.1002/\(SICI\)1520-6858\(1998\)1:4<267::AID-SYS3>3.0.CO;2-D](https://doi.org/10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D)
16. Morin, L., Nyatsikor, F.: Ephemeral buffers, a streamlined mechanism for cross-facilities data transfer, <https://ebuffer.aqmo.org/docs/ui>
17. Netto, M.A.S., Calheiros, R.N., Rodrigues, E.R., Cunha, R.L.F., Buyya, R.: Hpc cloud for scientific and business applications: Taxonomy, vision, and research challenges. *ACM Comput. Surv.* **51**(1) (Jan 2018). <https://doi.org/10.1145/3150224>
18. Renater: La fédération Éducation-recherche, <https://services.renater.fr/federation/index>
19. Ritchie, O.M., Thompson, K.: The unix time-sharing system. *The Bell System Technical Journal* **57**(6), 1905–1929 (1978). <https://doi.org/10.1002/j.1538-7305.1978.tb02136.x>
20. Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C.: Openid connect core 1.0 (Nov 2014), https://openid.net/specs/openid-connect-core-1_0.html
21. Saltzer, J.H. and Reed, D., D.D., C.: End-to-end arguments in system design. *ACM Transactions on Computer Systems* **2**, 277–288 (1984). <https://doi.org/10.1145/357401.357402>
22. Sha, C., Zhang, J., An, L., Zhang, Y., Wang, Z., Bat, N., Verlič, M., Ji, Q.: Facilitating hpc operation and administration via cloud (04 2019). <https://doi.org/10.14529/js190105>
23. Shimwell, T.W., Tasse, C., Hardcastle, A.P., Williams M. J., e.a.: The LOFAR Two-metre Sky Survey. II. First data release. *Astronomy & Astrophysics*, Volume 622, id.A1, 21 pp. **622**, A1 (Feb 2019). <https://doi.org/10.1051/0004-6361/201833559>
24. Ferreira da Silva, R., Bard, D., Chard, K., de Witt, S., Foster, I., al.: Workflows Community Summit 2024: Future Trends and Challenges in Scientific Workflows. Tech. rep., Oak Ridge National Laboratory, USA (Oct 2024). <https://doi.org/10.5281/zenodo.13844758>, <https://hal.science/hal-04902062>
25. The Apache Software Foundation: Apache nifi. <https://nifi.apache.org/> (2025)
26. The Galaxy Community : The galaxy platform for accessible, reproducible, and collaborative data analyses: 2024 update. *Nucleic Acids Research* **52**(W1), W83–W94 (05 2024). <https://doi.org/10.1093/nar/gkae410>