Scaling Scientific Workflows in Europe: Architecture and Deployment of the Galaxy-Pulsar Computational Network

Author list with affiliation >> PLESE ADD YOURSELF AT THE END OF THE MANUSCRIPT <<

Target journals: NARGAB, …

Abstract

The exponential growth of data-intensive research in the Life Sciences and other scientific domains has led to an increasing demand for computational resources across Europe. While major EU initiatives have supported the development of large-scale infrastructures, researchers still face significant challenges in accessing and integrating heterogeneous computing environments. Fragmentation in both technical architectures and policy frameworks continues to hinder seamless interoperability and efficient resource sharing.

To address these challenges, we present the European Pulsar Network (EPN), a distributed computing architecture built upon the Galaxy workflow management system and the Pulsar job execution service. The network enables transparent job offloading from Galaxy servers to remote computing clusters, allowing resource scaling without requiring direct user intervention or awareness. This architecture supports a flexible and scalable approach to workload distribution, ensuring improved performance and resource availability across different infrastructures.

A key component of the network is the Open Infrastructure framework, which facilitates the streamlined deployment of both Galaxy servers and Pulsar endpoints by compute providers. This is achieved through automation tools such as Terraform and Ansible, eliminating the need for manual reconfiguration of existing systems. This approach promotes simplified integration and ease of adoption by new sites.

To date, the EPN comprises six national Galaxy endpoints, other than the central European instance, and thirteen Pulsar endpoints distributed across Europe. This collaborative effort is actively enhancing the scalability, resilience, and interoperability of the Galaxy ecosystem in support of FAIR and reproducible data analysis.

Graphical Abstract

Introduction

In recent years, the demand for computational resources in the Life Sciences has increased significantly (1, 2). This growth is primarily driven by advancements in high-throughput technologies such as next-generation sequencing, high-resolution imaging, and integrative omics analyses, which generate vast amounts of complex data requiring sophisticated computational processing (3). Similar computational demands are also increasingly common in

scientific domains beyond the Life Sciences domain, such as biodiversity and climate science, muon spectroscopy in materials science, and astrophysics.

In this context, the European Galaxy server (usegalaxy.eu) plays a central role by offering a public, user-friendly platform for data analysis. Widely adopted across the Life Sciences community and beyond, it allows researchers to process complex datasets without the need to manage a local infrastructure. As shown in Fig. 1, both the number of users and the number of submitted jobs on usegalaxy.eu have steadily increased over the years, reflecting the growing computational needs in research and the platform's expanding user base.

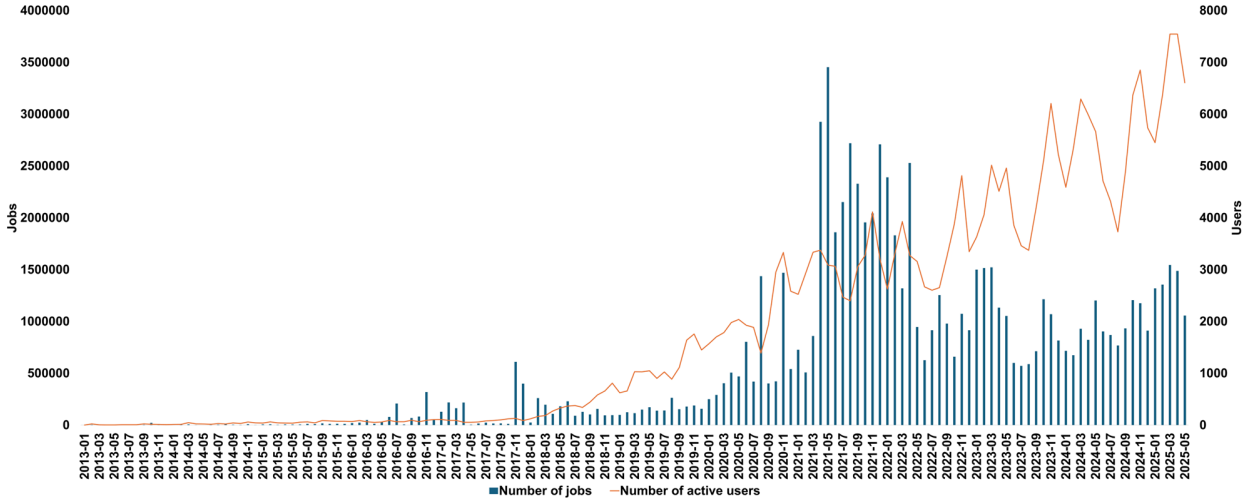

Fig. 1 - Number of users, defined as users submitting at least one job, and the number of jobs over the time on usegalaxy.eu.

To address these escalating computational needs, the European Union has made substantial investments through strategic funding initiatives, including Horizon Europe (https://horizoneurope.apre.it/), the European Strategy Forum on Research Infrastructures (https://www.esfri.eu), EuroHPC (https://www.eurohpc-ju.europa.eu), and the European Open Science Cloud (https://eosc.eu). These programmes have explicitly supported the development and deployment of large-scale computational research infrastructure, providing researchers with extensive computational infrastructure, fostering collaborative research, and enhancing scientific innovation across EU member states.

Nevertheless, significant challenges still hinder the effective integration and interoperability of these diverse computing infrastructures: the heterogeneous nature of hardware configurations, software environments, middleware stacks, and workflow management systems across European research institutions complicates the seamless utilization of distributed computational resources by researchers (4). Additionally, fragmentation in authentication and authorization frameworks, driven also by heterogeneous legal and institutional policies at the national or regional levels, further exacerbates these interoperability issues.

In this work, we describe the European Pulsar Network (EPN), designed to distribute computational analysis jobs across multiple data centers via national Galaxy (5) portals. The

Galaxy portals function as gateways to the computational resources of the EPN while acting as platforms for configuring, submitting, and managing data analysis tasks. Developed within the Horizon Europe EuroScienceGateway (6) project, the EPN aims to provide efficient and structured access to data, tools, and workflows, supported by a suitable IT infrastructure. To this end, we implemented a dedicated deployment and monitoring framework that ensures the infrastructure is scalable, easy to manage, and continuously monitored, fostering a sustainable and federated computing environment across Europe for research purposes. The network currently consists of thirteen computational endpoints supporting six National Galaxy instances across Europe in addition to the pre-existing European Galaxy instance. The network has been designed to allow for easy onboarding of new endpoints, sharing among Galaxy servers, or removal if no longer needed.

Methods

Galaxy

Galaxy functions as the access gateway to the EPN. It is a scientific workflow management system designed to simplify the execution of complex data analyses for researchers across diverse scientific domains. It provides an intuitive web interface through which users can build, customize, and share multi-step computational workflows without using the command line. Originally developed for bioinformatics, Galaxy now supports a broader range of tools and data types, and can scale from a small local server to large Cloud and HPC infrastructures. Moreover, its transparent execution model and workflow history features foster reproducibility, collaboration, and accessibility in data-driven research.

Galaxy also provides a Python-based library, BioBlend (7), that enables automated interaction with Galaxy servers via their REST API. It streamlines tasks such as launching workflows, managing datasets and histories, and tracking tool executions. Further extending its capabilities, Galaxy supports connectivity with a wide range of storage backends, ranging from institutional to public and commercial services, allowing seamless data access and management. Additionally, Galaxy can integrate with platforms like Zenodo to export, publish, and archive datasets, workflows, and histories, thus supporting the principles of FAIR (Findable, Accessible, Interoperable, and Reusable) science.

To ensure responsiveness and scalability, especially in environments serving many concurrent users, Galaxy can be integrated with external components that optimize different layers of the system architecture. NGINX, a high-performance web server and reverse proxy, is used to efficiently handle incoming requests, serve static content, and balance traffic across multiple Galaxy processes. PostgreSQL is used as the primary relational database management system to store and manage all core metadata related to users, histories, datasets, workflows, tool executions, and system configurations, enabling efficient querying and indexing of large volumes of records generated through user activity. Finally Celery, an open-source distributed task queue, is used to manage background and asynchronous tasks. Celery allows Galaxy to offload non-blocking operations, such as processing data uploads, recalculating disk usage, managing metadata, purging datasets, generating workflow reports, and preparing history

exports, ensuring that these resource-intensive activities do not interfere with user interactions or tool execution.

Pulsar

Pulsar (8) is a remote job execution system developed within the Galaxy Project to enable distributed and flexible computation across heterogeneous environments. Its purpose is to decouple the execution of Galaxy jobs from the main server by allowing them to be offloaded to remote computing facilities (Fig. 2).

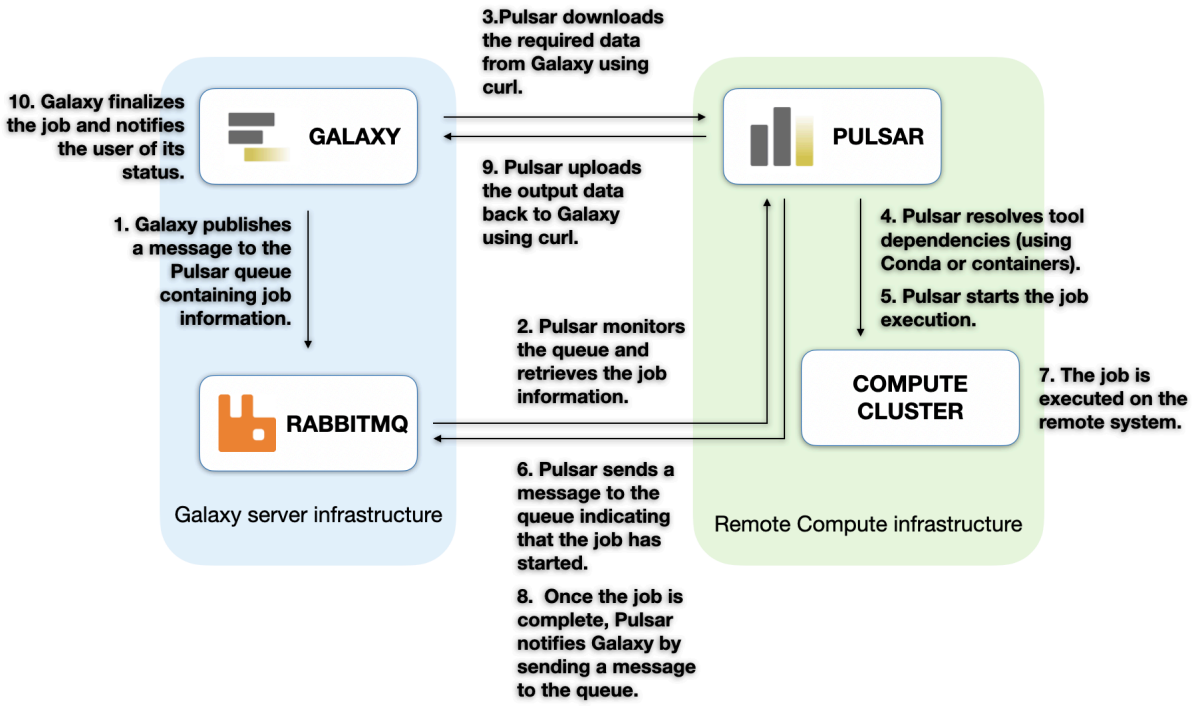

Fig. 2 - . Workflow of remote job execution using Pulsar and Galaxy. This diagram illustrates the asynchronous communication between Galaxy and a remote Pulsar server using RabbitMQ as a message broker. Galaxy publishes job information to a dedicated message queue (1), which is monitored by Pulsar (2). Pulsar retrieves the job payload, downloads input data from Galaxy via curl (3), resolves dependencies (4), and starts the job (5) on the connected compute cluster (6,7). Upon job completion, Pulsar sends the status update to Galaxy via the queue (8), and uploads output data back using curl (9). Galaxy then finalizes the job and informs the user (10).

Pulsar operates as a lightweight service receiving job execution requests from a Galaxy instance. It exposes a RESTful API through which Galaxy submits job descriptions, input metadata, and execution parameters. Alternatively, to avoid direct network exposure for RESTful communication, Pulsar operates using the AMQP (Advanced Message Queuing Protocol) message queue protocol. With this approach, Galaxy and Pulsar exchange job-related messages asynchronously via the RabbitMQ message broker: Galaxy serializes job information into messages and publishes them to a designated, authenticated, queue. Each queue is associated with a specific Pulsar endpoint, and only that endpoint is authorized to connect and

consume messages from it. Remote Pulsar endpoints, acting as consumers, subscribe to the queue and asynchronously retrieve the job payloads and execute them independently. Once a job is received, Pulsar handles input staging, either by accessing a filesystem shared between the submitting Galaxy server and the remote cluster or by downloading data from the submitting Galaxy server with curl (https://curl.se/). Then, Pulsar dispatches the job to the batch system it is connected to, such as HTCondor or SLURM. Tools dependencies must be available on the remote cluster compute nodes: as for Galaxy, Pulsar can resolve tool dependencies through multiple mechanisms, including Conda packages (https://anaconda.org), Docker (https://www.docker.com/) and Singularity (https://apptainer.org/) Containers, allowing it to adapt to the specific characteristics and constraints of the underlying infrastructure. Upon job completion, output files are collected and sent back to the Galaxy server, with curl again, where they are made available for further analysis or visualization in the user's history. Execution logs, environment metadata, and job state information are also collected and sent back to the Galaxy instance, so that the provenance of each output is trackable.

Galaxy-Pulsar integration

From the Galaxy point of view, Pulsar is a backend runner, integrated through a job configuration layer that defines execution destinations. Administrators can assign specific tools or workflows to run through Pulsar based on criteria such as job size, tool requirements, user group, or current system load. This dynamic routing mechanism ensures optimal resource usage and facilitates fine-grained control over job distribution across multiple remote environments. From the end user's perspective, the interaction remains entirely transparent: job submission and monitoring continue through the familiar Galaxy web interface, while the execution may occur on a completely different physical infrastructure. Alternatively, Galaxy also allows users to manually select the execution endpoint for their jobs through the user preferences interface.

CERN VM FileSystem

Whether an analysis is executed on the EPN directly within Galaxy or remotely via Pulsar, data analysis tools can require both reference data and a reliable mechanism for resolving software dependencies. To prevent unnecessary duplication and ensure consistency across sites, container images and reference data are distributed using a CernVM File System (CVMFS) (9): a read-only file system designed to deliver software and data to distributed computing environments. The CVMFS volume is shared across all the EPN gateways and endpoints.

HTCondor

HTCondor (https://htcondor.org/) is a widely adopted open-source batch system designed for managing and executing large volumes of compute jobs across heterogeneous resources. It is particularly effective in shared, multi-user environments where fine-grained control over job prioritization, fair-share scheduling, and resource allocation is required. In the context of distributed infrastructures, HTCondor provides mechanisms for job queuing, dynamic resource discovery, and fault-tolerant execution.

The Virtual Galaxy Cloud Nodes images

The Virtual Galaxy Compute Node (VGCN) (https://github.com/usegalaxy-eu/vgcn) is a Rocky Linux 9 (https://rockylinux.org/) pre-built image for creating Virtual Machines (VM) on cloud-based infrastructures, hosting essential services and tools required for Galaxy job execution, including container runtimes, monitoring agents, and remote job runners. VGCN images are intended to be cloud-init configurable, allowing site-specific customization and integration into a variety of deployment models, from single-node setups to distributed clusters.

Infrastructure and software deployment automation

Terraform (https://developer.hashicorp.com/terraform) enables the declarative provisioning of virtual IT infrastructure across Cloud platforms such as OpenStack, including commercial providers such as Oracle Cloud and AWS. Complementing this, Ansible automates the configuration and setup of software environments on the IT infrastructure, streamlining the deployment of complex services and enforcing uniformity across nodes. Together, they enable the scalable and reproducible setup of analysis platforms and associated resources.

Infrastructure management

Continuous Integration and Continuous Delivery (CI/CD) systems are used by the EPN to automate maintenance tasks, apply software updates, and perform routine testing across all the components of a deployed infrastructure. Jenkins (https://www.jenkins.io/), a CI/CD application, is used to orchestrate automated data analysis pipelines on the EPN to validate updates, trigger redeployments, and integrate modifications into live environments with minimal downtime. Since configuration files are stored on GitHub, any modification is enacted on the EPN by Ansible in response to pull requests.

TESP-API: A Lightweight TES Execution Backend

The GA4GH Task Execution Service (TES) API (10) defines a standard interface for submitting and managing batch computing tasks across heterogeneous execution environments.

The TESP API (https://github.com/CESNET/tesp-api) provides a minimal implementation of the GA4GH TES, intended for simple deployment and integration with container-based execution environments. It can run either in standalone mode—with an embedded Pulsar service launched via `docker compose --profile pulsar up`— or in a mode using an external Pulsar instance, where `docker compose up` starts only the API and database components. This approach allows flexible setups for testing, development, or production use.

The service receives TES tasks and converts them into local `singularity exec` or `docker run` commands. All data movement is handled by the worker node itself, avoiding intermediate transfers through the API or Pulsar layers. As a result, input and output files specified in the TES task are transferred directly between client-side storage and the compute node, following a simplified `storage – worker – storage` pattern.

In Galaxy, jobs submitted to Pulsar using *PulsarTesJobRunner* do not declare explicit inputs or outputs in the TES request. Instead, data transfer is delegated to executor components, with the first and last executors handling input staging and output collection.

Standardized job submission and workflow portability

WfExS-backend (Workflow Execution Services backend workflow engine orchestrator) (11), whose developments started within the EOSC-Life project, provides a solution for orchestrating scientific workflow execution integrating the GA4GH TES specs.

WfExS takes advantage of the fact that many workflow engines, such as Nextflow (12) and CWLtool (https://www.commonwl.org), use Docker containers to encapsulate the tools executed at each workflow step. These engines typically rely on a predictable structure of input/output directories and files within the container environment. To bridge the gap between containerized execution and standardized APIs, WfExS introduces a command-line translation layer, called TESSAP, which both mimics the behavior of the Docker CLI (for the subset of commands typically used by workflow engines) and translates these commands into GA4GH TES API calls. This allows workflow engines to operate as if they were interacting with Docker directly, while actual task execution is redirected to a remote TES-compliant service, such as Pulsar nodes running TESP.

Results

The Pulsar Network architecture

Figure 3 illustrates the EPN architecture. Each Pulsar endpoint and each Galaxy server communicate via a dedicated message queue brokered by RabbitMQ. Once a user submits a job and this is earmarked for execution in a particular Pulsar endpoint, Galaxy dispatches the job request to the queue dedicated to the destination endpoint, which continuously monitors it. Upon receiving the job, Pulsar transfers the required input data from the requesting Galaxy instance, executes the computational task, either locally or submits it to a connected cluster, uses the shared CVMFS volumes for accessing reference datasets and software dependencies, and finally uploads the resulting output and its associated metadata and logs back to the originating Galaxy server. To be noted, each single Pulsar endpoint can be configured to serve multiple Galaxy instances.

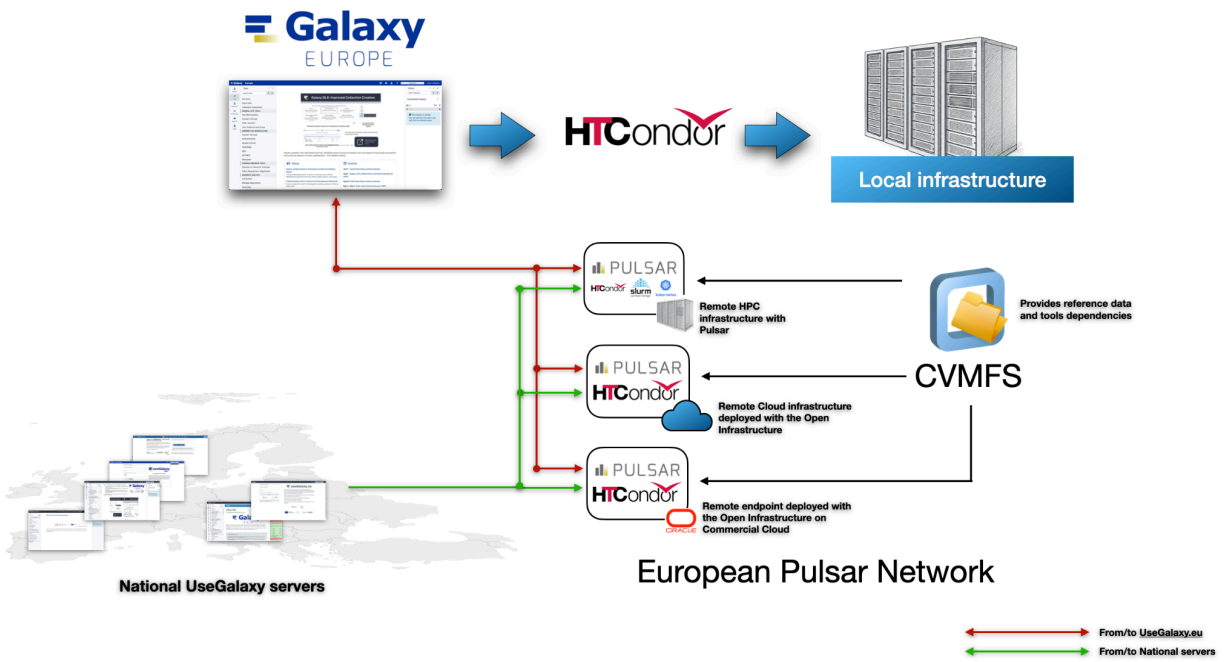

Fig. 3 - The European Pulsar Network architecture. By default, the European and national Galaxy servers submit jobs to local IT resources, e.g., usegalaxy.eu relies on an HTCondor-based compute cluster. The Pulsar endpoints of the EPN can be used as an alternative to relieve pressure from the local infrastructures. The Pulsar endpoints are configured to retrieve jobs metadata from a dedicated message queue of the RabbitMQ broker, and to dispatch them to the compute cluster they are connected to. Reference data and tool dependencies are provided through shared CVMFS volumes

The Open Infrastructure - Pulsar Endpoints

The Open Infrastructure (OI) framework provides a ready-to-use VGCN cloud image, containing the software components required to create a fully operational Pulsar node or Galaxy server.

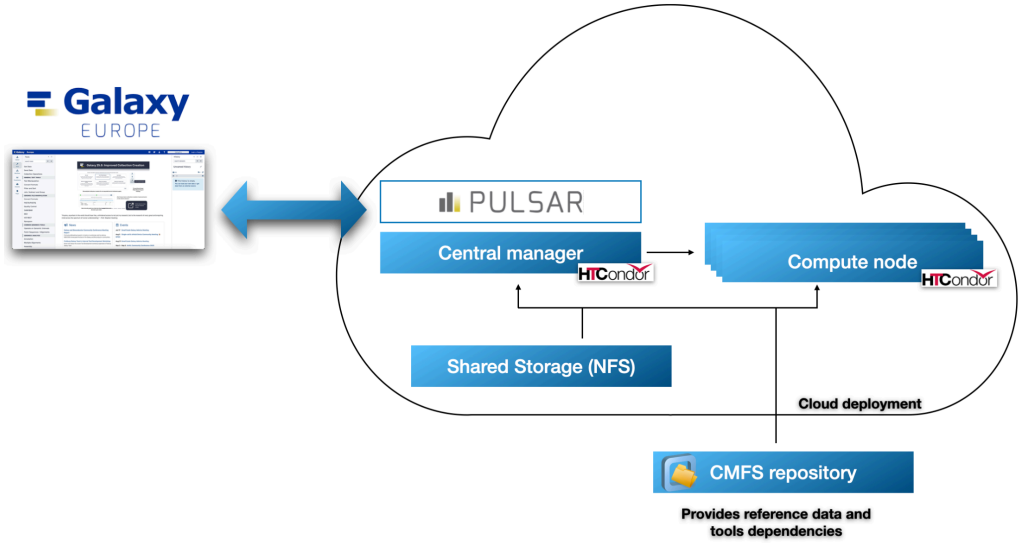

Fig. 4 - A Pulsar endpoint default configuration deployed with the OI and connected to UseGalaxy.eu. Each endpoint consists of a virtual machine running both the HTCondor Central Manager and the Pulsar service, a configurable number of compute nodes, and an NFS server that supplies shared storage to all the virtual machines within the endpoint. Finally, the CVMFS volume is connected to each compute node to provide reference data and software dependencies.

The automated deployment (Fig. 4) consists of an HTCondor setup, including a Central Manager virtual machine that hosts both the HTCondor daemons and the Pulsar application, one or more compute nodes, and an NFS node providing shared storage across all the components of the endpoint. Core configuration parameters, such as the number of compute nodes and storage volumes, are managed via a Terraform variables file, allowing seamless extension of the endpoint's computational capacity. Once the RabbitMQ queue to be monitored is specified, a new endpoint can be instantiated with a single command.Terraform deploys the necessary virtual machines and sets up the networking and the shared NFS storage across them, while Ansible configures HTCondor on each node and sets up the Pulsar endpoint along with the services required for its operation (.

During the configuration process, the OI allows for one or more queues to be associated with the computational endpoint. Ansible configures a dedicated Pulsar daemon for each Galaxy instance that will be served by the endpoint, ensuring that each one listens to its assigned queue and submits jobs to the HTCondor cluster in the backend. The OI framework also includes an Ansible playbook to dynamically add or remove the Galaxy instances authorized to utilise the Pulsar endpoint.

The entire deployment procedure has been thoroughly documented in the Pulsar Network documentation (https://pulsar-network.readthedocs.io).The deployment strategy and its implementation have been successfully tested also on the commercial platform Oracle Cloud (Fig. 5), and EOSC EU Nodes (https://galaxyproject.org/news/2025-01-29-esg-eosc/), demonstrating the generalisability of the framework and enabling administrators, in principle, to take advantage of available credits from commercial providers.

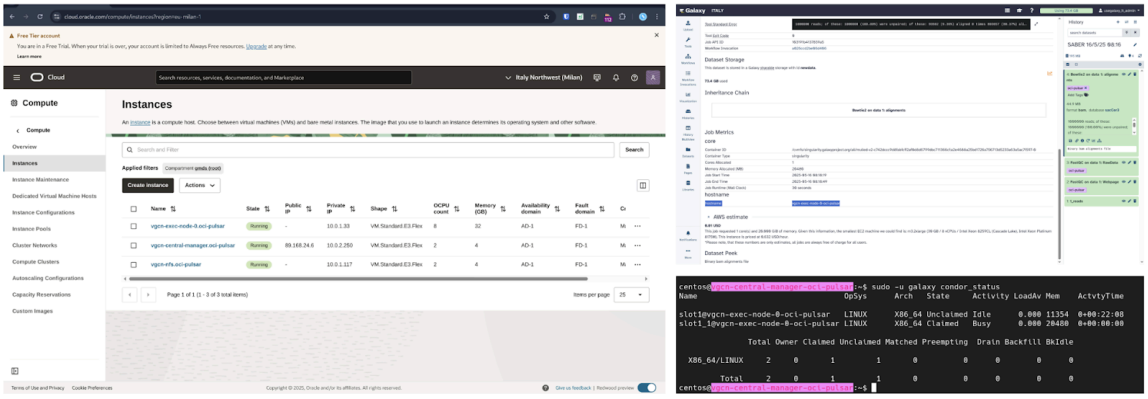

Fig. 5 - Pulsar endpoint deployment on the Oracle Cloud Infrastructure (OCI). The OCI dashboard detailing the VMs created through Terraform (left). A Galaxy job submitted through usegalaxy.it on the OCI endpoint (top right). The status of the HTCondor cluster deployed using OCI resources (bottom right).

The European Pulsar Network and UseGalaxy.* public servers

Currently, the EPN includes thirteen Pulsar endpoints across ten countries (Fig. 6) supporting six national UseGalaxy instances in addition to the European one. Additionally, the HCMR institute has deployed an additional Pulsar endpoint as part of the *FairEase* project (https://fairease.eu/), which, although independent from the EuroScienceGateway project, has adopted the Pulsar Network infrastructure for its distributed job execution needs.

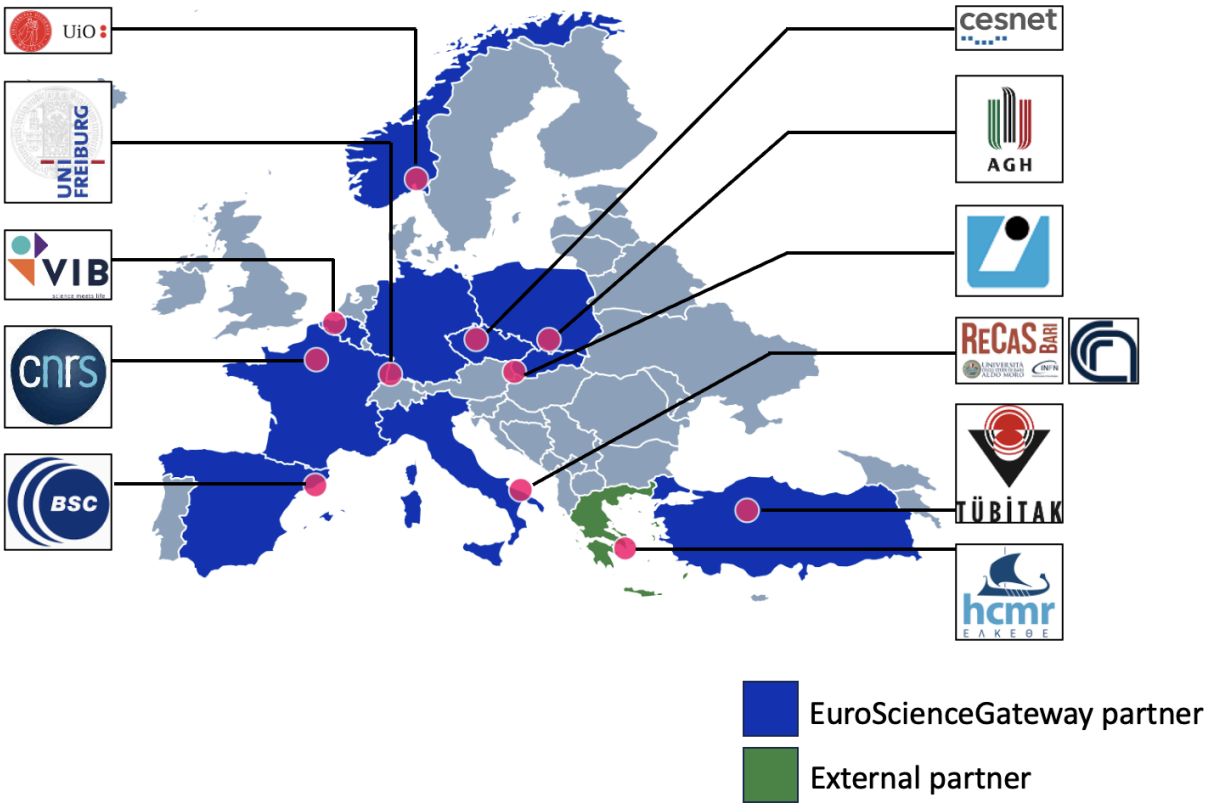

Fig. 6 - Map of the Pulsar endpoints from EuroScienceGateway partners.

SABER, the EPN monitoring system

Given the scale of the infrastructure, regular and thorough testing and monitoring are essential to detect any underperforming endpoint, identify issues, and promptly restore full operability. To this end, we developed SABER - Systematic API Benchmark for pulsar Endpoint Reliability, (https://github.com/usegalaxy-it/saber), a Python-based application designed to iteratively test all the Pulsar endpoints connected to a Galaxy server. Rather than merely checking their online status, SABER submits to Galaxy a small batch of actual jobs, targeting Pulsar endpoints for execution, allowing the components of the endpoint to be thoroughly validated. SABER can be configured in a fine-grained way through a YAML configuration file. It supports testing multiple Pulsar endpoints assigned to different Galaxy instances, also defining input data and workflows

independently for each Galaxy server, specifying the time to wait for each job to complete, choosing whether to clean histories for debugging, and optionally including the local compute endpoint in the tests alongside the remote ones.

A dedicated Jenkins pipeline is then used to automatically and periodically run SABER, achieving continuous monitoring and testing of the whole network (Fig. 7). SABER is designed to be instance-agnostic and can be used to test Pulsar endpoints connected to any Galaxy instance. A SABER job is executed daily for the EPN, and the results are made available as HTML reports at monitor.usegalaxy.it, with additional markdown versions also accessible on Github (https://github.com). Moreover, the Galaxy histories created during the test cycle can be easily explored for debug purposes, with jobs tagged and grouped by the Pulsar endpoint used for their execution.

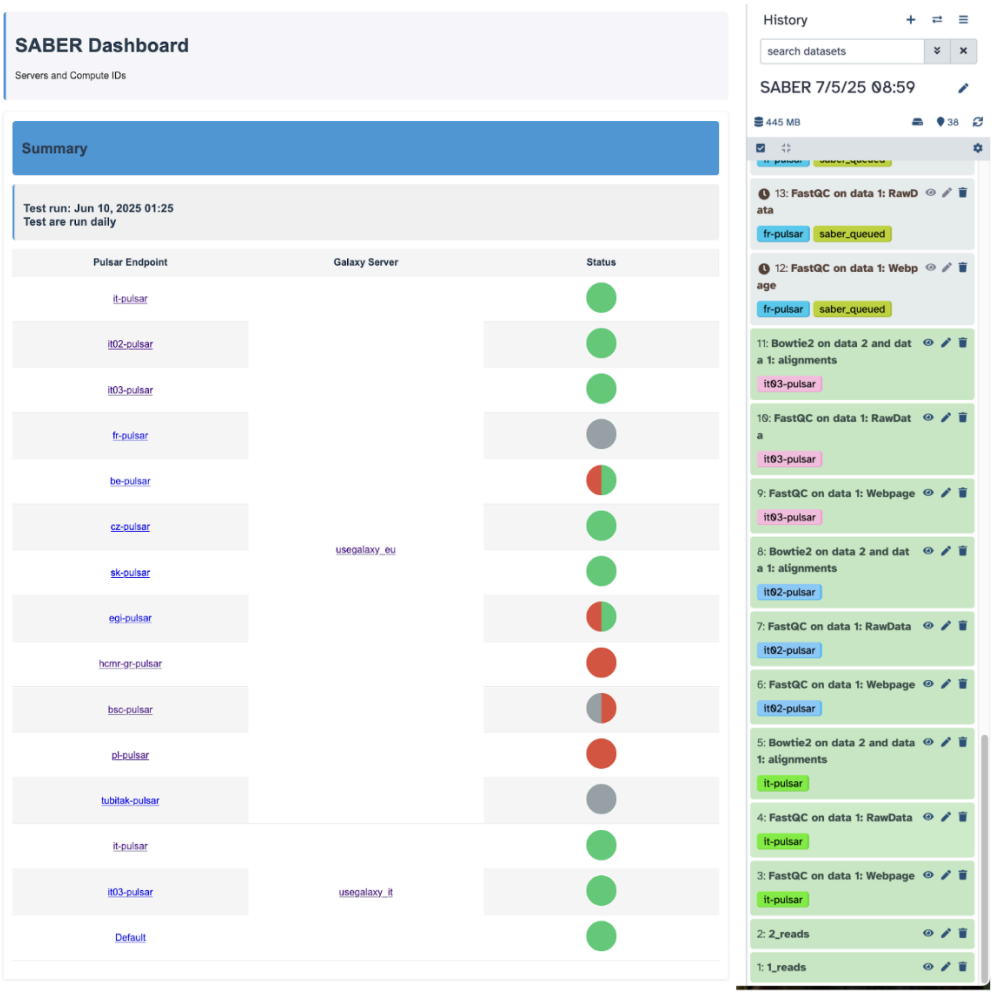

Fig. 7 - The SABER Dashboard displaying the results of the tests performed on the Pulsar endpoints (left), alongside the corresponding Galaxy job history for the tested workflows (right).

The Open Infrastructure - UseGalaxy servers

The Open Infrastructure framework enables the deployment, configuration and maintenance of new production-grade Galaxy servers, ready for submitting jobs to Pulsar endpoints

Again, Terraform is used to provision the virtual infrastructure as described in Table 1 and Fig. 8, while Ansible handles the configuration of the software components through a set of playbooks.

VM	Description	Public IP needed	Provisioned by OI
Galaxy	Hosts the Galaxy application and the Nginx web server, with a public IP address to allow user access via the web interface.	yes	yes
RabbitMQ	Hosts the RabbitMQ message broker used to connect Galaxy to Pulsar endpoints and to Celery. Exposes a public IP to allow access from remote services deployed on separate infrastructures.	yes	yes
Celery	Executes asynchronous background tasks triggered by Galaxy, such as job preparation, tool dependency resolution, and metadata setting. Works in coordination with RabbitMQ to handle distributed task queues efficiently.	no	yes
PostgreSQL DB	Acts as the primary relational database backend required for Galaxy's operation.	no	yes
PostgreSQL DB replica	Maintains a real-time copy of the primary PostgreSQL database to ensure high availability and data redundancy	no	yes
PostgreSQL DB backup	Performs periodic snapshots of the PostgreSQL database to allow recovery in case of data corruption or accidental deletion.	no	yes
NFS server	Provides shared file storage for Galaxy and associated services.	no	yes

HTCondor CM	Coordinates and manages the scheduling and execution of jobs within the HTCondor pool.	no	yes
HTCondor compute nodes	Execute the computational jobs scheduled by HTCondor.	no	no
Control VM	Used to deploy and manage the entire infrastructure by orchestrating Terraform and Ansible.	yes	no

Table 1 - Overview of the virtual machines composing the UseGalaxy infrastructure, detailing their primary functions, public IP accessibility, and whether they can be deployed using the OI automation tools.

The virtual machine hosting the Galaxy server acts as a reverse proxy and handles HTTPS termination. This frontend also supports the open TUS resumable upload protocol through a TUSD server, enabling resumable, chunked uploads to better support users with large datasets. As the access point for end users, this VM is assigned a public IP address.

A separate virtual machine running PostgreSQL, handles the database used by Galaxy to store metadata, user information, histories, and workflow definitions. The database is configured with replication and backup mechanisms on separated VMs, to ensure data durability and support disaster recovery strategies.

The Galaxy components and compute nodes have access to the shared volume provided by a storage VM through NFS. This component stores the user-uploaded data, intermediate job files, and analysis results. Compute tasks are devolved to a VM running the HTCondor Central Manager, which coordinates job scheduling and resource allocation across the local Condor pool. This manager is responsible for assigning jobs to available compute slots and monitoring the state of the cluster.

The communication between Galaxy and remote Pulsar endpoints is managed via a RabbitMQ broker. This component handles message queues used by the Pulsar endpoints, which receive and execute jobs on external or federated infrastructures. The RabbitMQ server is also exposed through a public IP address with secure HTTPS connections and mandatory authentication for each message queue, to ensure that all communications are protected and authorized.

Finally, a Celery task manager is deployed to handle Galaxy's asynchronous background tasks: it interacts with both the Galaxy and RabbitMQ servers to ensure responsive and scalable task execution.

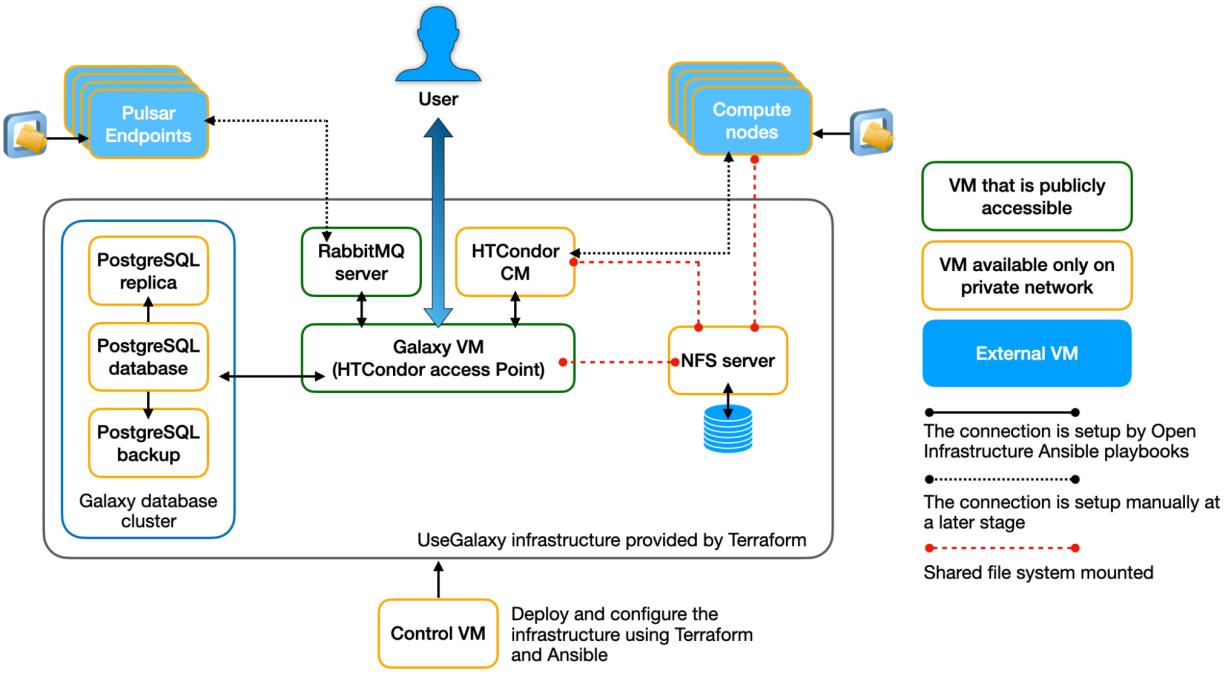

Fig. 8 - The UseGalaxy architecture based on the OI framework and deployed on cloud resources. A Control VM orchestrates the setup and configuration of the entire infrastructure. Initially, Terraform provides dedicated VMs for each of the core components: Galaxy, the database cluster, the HTCondor Central Manager, and the NFS server. Then, Ansible configures the software stack for each service. Finally, a Jenkins-based continuous integration/continuous delivery (CI/CD) system supports the infrastructure management tasks. The Ansible playbooks and roles used for deploying the virtual infrastructure are reused to automate routine operations, including software updates, tool installations, and configuration maintenance.

Other Workflow management systems integration

Discussion

Typically, public Galaxy servers utilize local computational resources, leveraging either HPC or cloud infrastructure, depending on availability. The European Pulsar Network represents a scalable framework for distributed job execution that extends the computing capabilities of national and pan-European Galaxy instances. By decoupling job execution from the local IT resources available to a Galaxy instance by leveraging remote compute endpoints, the architecture facilitates the integration of heterogeneous infrastructures, including institutional HTC/ HPC systems and Cloud environments, while maintaining compatibility with Galaxy's execution model and user experience.

Currently, the European Pulsar Network supports six national Galaxy instances (France, the Czech Republic, Belgium, Italy, Spain, and Norway) in addition to the central European Galaxy server. From an organizational perspective, the decision to rely on national Galaxy instances

instead of a single centralized portal reflects several practical and strategic benefits. First, data governance requirements, particularly in the biomedical domain, often mandate that sensitive datasets remain within national boundaries to comply with GDPR and institutional policies. Second, proximity between data and compute infrastructure enhances performance and reduces network costs, especially for large-scale omics datasets. National instances also support autonomy in scaling, funding, and user support, enabling tailored environments aligned with local priorities. Yet, thanks to shared protocols and tools like Pulsar, these instances retain full interoperability within the federated ecosystem.

Once deployed, the Pulsar endpoints are not limited to a single Galaxy instance: the architecture is designed to support shared usage across multiple Galaxy servers, promoting a more efficient use of resources and fostering collaboration at both national and European levels. Actually, this approach offers strategic advantages: it allows a single infrastructural investment to serve both national and European Galaxy instances, reducing unnecessary duplications and simplifying maintenance operations. The Pulsar Network's distributed architecture also brings built-in resilience: if a specific endpoint becomes unavailable, jobs can be rerouted to alternate nodes, minimizing downtime and preventing single points of failure, thus ensuring high reliability and availability for users. Monitoring and testing play a crucial role in maintaining operational integrity: SABER can be used to routinely test endpoint responsiveness and health, ensuring that failures are quickly identified and mitigated, making the Pulsar network dependable even as it grows in scale and complexity.

The first core principle behind the design of the network has been to provide transparency for users and providers: researchers submit, monitor, and access the results of their analyses on Galaxy as they did before, without being exposed to the complexity of the underlying infrastructure. On the other hand, advanced users can optionally select a specific execution endpoint through a simple and intuitive interface if they so wish.

On the provider side, ease of deployment and maintainability are equally critical. Indeed, on the infrastructure side, enabling resource providers to contribute with minimal effort was the second guiding principle. This led to the development of the OI framework, which standardizes and automatizes the resources provisioning and configuration operations. With this approach, a new node can be instantiated and connected to the EPN with minimal manual intervention, encouraging broader participation from institutional and national partners. Furthermore, the EPN adopts a pull-based, event-driven architecture: instead of requiring inbound connectivity, each Pulsar instance establishes an outbound connection to a central broker to receive job instructions. This architecture eliminates the need to open ports on the compute nodes, making the system much more compatible with closed or secured networking environments. As a result, Galaxy can offload tasks to infrastructures previously unreachable due to network constraints.

Finally, the adoption of the open-source model underpins the network's sustainability and trustworthiness. Tools like Galaxy, Pulsar, Terraform, Ansible, and Jenkins provide transparency, auditability, and long-term viability. Deployment automation is fully codified and version-controlled, with Ansible ensuring idempotent configuration and Jenkins automating continuous integration and delivery. As a result, the entire system remains consistent, up to

date, and fully manageable through version-controlled code. Moreover, routine operations, such as tool updates, workflow deployment, and dependency management, are reproducible and traceable, enabling stable evolution of the infrastructure, enforcing uniformity across nodes and enabling scalable and reproducible setup of analysis platforms and associated resources.

Conclusions and outlook

Although significant progress has been made in the development and funding of Cloud Computing and HPC/HTC infrastructures across Europe, effectively harnessing these resources remains a challenge in many research contexts. Researchers are often required to interact with a variety of computing environments, each with its own access methods, interfaces, and policies. This diversity, while reflecting the richness of the European landscape, can pose barriers to seamless adoption, especially for users without specific knowledge of the underlying infrastructures.

The EPN addresses these issues by providing a robust, scalable, and user-friendly solution for distributed job execution within the Galaxy ecosystem. By enabling transparent integration of European computing resources through a unified and modular architecture, it strikes a balance between usability for end users, who interact via the familiar Galaxy interface, and flexibility for infrastructure providers, who can deploy new Galaxy servers or Pulsar endpoints without reconfiguring existing systems. This approach not only simplifies access for researchers but also facilitates the sharing of computational resources at the European scale using mature and automated deployment technologies. For example the OI framework has been successfully used to deploy the *usegalaxy.it* Galaxy instance on the Italian CINECA and ReCaS-Bari Cloud infrastructures (https://galaxyproject.org/news/2024-11-23-usegalaxy-it-starting/), demonstrating its viability and the effectiveness of this approach.

A further refinement of the framework is Pulsar integration beyond Galaxy, enabling its use as a backend for other workflow management systems such as CWL-based engines or Nextflow. This evolution would further broaden the accessibility of the infrastructure, opening it up to new user communities and enhancing its relevance across a wider range of data-intensive scientific domains.

References

1. Stephens,Z.D., Lee,S.Y., Faghri,F., Campbell,R.H., Zhai,C., Efron,M.J., Iyer,R., Schatz,M.C., Sinha,S. and Robinson,G.E. (2015) Big Data: Astronomical or Genomical? *PLOS Biology*, **13**, e1002195.

2. Grossman,R.L. (2019) Data Lakes, Clouds, and Commons: A Review of Platforms for Analyzing and Sharing Genomic Data. *Trends Genet*, **35**, 223–234.

3. Sternberg,M.J.E. and Yosef,N. (2018) Computation Resources for Molecular Biology: Special Issue 2018. *Journal of Molecular Biology*, **430**, 2181–2183.

4. Orazio,S.D., Eva,S., Jean-Karim,H., Mark,V.D.S., Wierenga,K., Paolo,M., Damian,T., Norberto,M.J., Álvaro,L.G., Wim,H., *et al.* (2023) A landscape overview of the EOSC Interoperability Framework - Capabilities and Gaps Zenodo.

5. The Galaxy Community (2024) The Galaxy platform for accessible, reproducible, and collaborative data analyses: 2024 update. *Nucleic Acids Research*, **52**, W83–W94.

6. leveraging the European compute infrastructures for data-intensive research guided by FAIR principles | EuroScienceGateway | Projekt | Fact Sheet | HORIZON *CORDIS* | *European Commission*.

7. Sloggett,C., Goonasekera,N. and Afgan,E. (2013) BioBlend: automating pipeline analyses within Galaxy and CloudMan. *Bioinformatics*, **29**, 1685–1686.

8. Afgan,E., Baker,D., Batut,B., van den Beek,M., Bouvier,D., Čech,M., Chilton,J., Clements,D., Coraor,N., Grüning,B.A., *et al.* (2018) The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Research*, **46**, W537–W544.

9. Buncic,P., Sanchez,C.A., Blomer,J., Franco,L., Harutyunian,A., Mato,P. and Yao,Y. (2010) CernVM – a virtual software appliance for LHC applications. *J. Phys.: Conf. Ser.*, **219**, 042003.

10. Kanitz,A., McLoughlin,M.H., Beckman,L., Malladi,V.S. and Ellrott,K. (2024) The GA4GH Task Execution Application Programming Interface: Enabling Easy Multicloud Task Execution. *Computing in Science & Engineering*, **26**, 30–39.

11. Fernández,J.M., Rodríguez-Navas,L., Muñoz-Cívico,A., Iborra,P. and Lea,D. (2025) WfExS-backend. 10.5281/zenodo.15462518.

Author List with affiliation

Example: Marco-Antonio Tangaro, Institute of Biomembranes, Bioenergetics and Molecular Biotechnologies, National Research Council, Via Amendola 122/O, Bari (Italy), e-mail: marcoantonio.tangaro@cnr.it

Stefano Nicotri, INFN - Istituto Nazionale di Fisica Nucleare - Sezione di Bari, via Amendola 173, 70126, Bari, (Italy), e-mail: nicotri@infn.it

Björn Grüning, University of Freiburg, Friedrichstraße 39, 79098 Freiburg, Germany, e-mail: gruening@informatik.uni-freiburg.de orcid: 0000-0002-3079-6586

Sanjay Kumar Srikakulam, University of Freiburg, Friedrichstraße 39, 79098 Freiburg, Germany, e-mail: srikakus@informatik.uni-freiburg.de orcid: 0000-0002-1752-5060

Armin Dadras, University of Freiburg, Friedrichstraße 39, 79098 Freiburg, Germany, e-mail: dadras@informatik.uni-freiburg.de orcid: 0000-0001-7649-2388

Mira Kuntz, University of Freiburg, Friedrichstraße 39, 79098 Freiburg, Germany, e-mail: kuntzm@informatik.uni-freiburg.de orcid: 0000-0003-4302-5091

Oana Kaiser, University of Freiburg, Friedrichstraße 39, 79098 Freiburg, Germany, e-mail: marchis@informatik.uni-freiburg.de

Anthony Bretaudeau, GenOuest, IRISA, Campus de Beaulieu, F-35000 Rennes, France. email: anthony.bretaudeau@inrae.fr

Paul De Geest, VIB, Data Core, Ghent, Belgium, email: paul.degeest@psb.vib-ugent.be

Sebastian Luna-Valero, EGI Foundation, Amsterdam, Netherlands email: sebastian.luna.valero@egi.eu

María Chavero-Díez, 1. Barcelona Supercomputing Center, Plaça d'Eusebi Güell, 1-3, 08034 Barcelona (Spain). 2. Biochemistry and Molecular Biomedicine Department, University of Barcelona, Av. Diagonal 643, 08028 Barcelona (Spain). email: maria.chavero@bsc.es orcid: 0000-0002-2298-1634

José Mª Fernández, Barcelona Supercomputing Center ,Plaça d'Eusebi Güell, 1-3, 08034 Barcelona, Spain, email: jose.m.fernandez@bsc.es orcid: 0000-0002-4806-5140

Salvador Capella-Gutiérrez, Barcelona Supercomputing Center ,Plaça d'Eusebi Güell, 1-3, 08034 Barcelona, Spain, email: salvador.capella@bsc.es orcid: 0000-0002-0309-604X

Josep Ll. Gelpí, 1. Barcelona Supercomputing Center, Plaça d'Eusebi Güell, 1-3, 08034 Barcelona (Spain). 2. Biochemistry and Molecular Biomedicine Department, University of Barcelona, Av. Diagonal 643, 08028 Barcelona (Spain). email: gelpi@ub.edu orcid: 0000-0002-0566-7723

Ján Astaloš, Institute of Informatics, Slovak Academy of Sciences, Dúbravská cesta 9, Bratislava, Slovakia, e-mail: jan.astalos@savba.sk orcid: 0000-0003-4502-4463

Boris Jurič, CESNET z.s.p.o, Generála Píky 430/26, 160 00 Praha 6, Czech republic, e-mail: 499542@muni.cz

Miroslav Ruda, CESNET z.s.p.o, Generála Píky 430/26, 160 00 Praha 6, Czech republic, e-mail: ruda@ics.muni.cz, orcid: 0000-0001-9123-0634

Łukasz Opioła, Academic Computer Centre CYFRONET of the AGH University of Krakow, Nawojki 11 st., 30-950 Kraków, P.O. Box 6, Poland . email: lopiola@agh.edu.pl, orcid: 0000-0003-1997-932X

Silvia Gioiosa, HPC High Performance Computing Department, CINECA, Casalecchio di Reno, Italy. email: s.gioiosa@cineca.it

SUPPLEMENTARY MATERIAL

As part of the EuroScienceGateway project, several European institutions have deployed Galaxy servers and/or Pulsar endpoints to provide accessible and scalable computing resources for the life sciences community. Table 2 summarizes the distribution of these services across participating countries, listing the public Galaxy portals (usegalaxy.*) and the associated Pulsar endpoints registered.

Country	Institution	Galaxy	Pulsar
Germany		usegalaxy.eu	DE01 and development endpoints
France		usegalaxy.fr	FR01
Czech Republic		usegalaxy.cz	CZ01
Spain		usegalaxy.es	ES01
Belgium		usegalaxy.be	BE01
Italy		usegalaxy.it	IT01 (ReCas-Bari), IT02 (CINECA), IT03 (GARR), IT04 (UniMi)

EGI		-	EGI01 (Deployed by EGI on INFN Cloud infrastructure in Italy)
Slovakia		-	SK01
Turkey		-	TUBITAK01
Poland		-	CFY01
Norway		usegalaxy.no	-
Greece		-	HCMR01

Table 2 - List of UseGalaxy servers and Pulsar endpoints.

Some countries, such as Italy, maintain multiple Pulsar nodes to leverage different national infrastructures, while others contribute through either Galaxy interfaces, Pulsar nodes, or both.