Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

ISTI Technical Reports

D4Science interoperability framework: developer guidance for service integration

Massimiliano Assante, ISTI-CNR, Pisa, Italy
Andrea Dell'Amico, ISTI-CNR, Pisa, Italy
Elisa Molinaro, ISTI-CNR, Pisa, Italy
Alfredo Oliviero, ISTI-CNR, Pisa, Italy

ISTI-TR-2025/015

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

D4Science interoperability framework: developer guidance for service integration

Massimiliano Assante, Andrea Dell'Amico, Elisa Molinaro, Alfredo Oliviero

ISTI-TR-2025/015

Abstract

The D4Science Interoperability Framework provides the architectural backbone for integrating and operating heterogeneous services within the D4Science ecosystem. D4Science is a distributed digital infrastructure supporting international research communities through the operation of Virtual Research Environments (VREs) that enable collaborative access to data, analytical tools, and computing resources in line with Open Science and FAIR principles. The framework integrates core components such as the VRE Gateway, Virtual Labs, Identity and Access Management (IAM), and the Analytics Engine to deliver secure, scalable, and reproducible workflows. This document offers detailed guidance for service developers, covering containerisation, orchestration with Docker Swarm, authentication with OpenID Connect and OAuth2, and integrated monitoring and auditing tools. By combining federated access, container-based deployment, and robust governance mechanisms, this solution enables developers and researchers to create reusable, interoperable, and compliant services that advance collaborative and transparent scientific research.

Keywords: D4Science, Interoperability, VRE, Open Science, FAIR Principles, Containerisation, IAM, Cloud Computing.

Citation

Assante M., Dell'Amico A., Molinaro E., Oliviero A. *D4Science interoperability framework: developer guidance for service integration*. Technical Reports 2025/015. DOI: 10.32079/ISTI-TR-2025/015.

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Area della Ricerca CNR di Pisa
Via G. Moruzzi 1
56124 Pisa Italy
http://www.isti.cnr.it

Technical Report, No. ISTI-TR-2025/015, October 23, 2025
Consiglio Nazionale delle Ricerche - Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"

D4Science Interoperability Framework: Developer Guidance for Service Integration

Massimiliano Assante, Andrea Dell'Amico, Elisa Molinaro*, Alfredo Oliviero

Abstract
The D4Science Interoperability Framework provides the architectural backbone for integrating and operating heterogeneous services within the D4Science ecosystem. D4Science is a distributed digital infrastructure supporting international research communities through the operation of Virtual Research Environments (VREs) that enable collaborative access to data, analytical tools, and computing resources in line with Open Science and FAIR principles. The framework integrates core components such as the VRE Gateway, Virtual Labs, Identity and Access Management (IAM), and the Analytics Engine to deliver secure, scalable, and reproducible workflows. This document offers detailed guidance for service developers, covering containerisation, orchestration with Docker Swarm, authentication with OpenID Connect and OAuth2, and integrated monitoring and auditing tools. By combining federated access, container based deployment, and robust governance mechanisms, this solution enables developers and researchers to create reusable, interoperable, and compliant services that advance collaborative and transparent scientific research.

Keywords
D4Science, Interoperability, VRE, Open Science, FAIR Principles, Containerisation, IAM, Cloud Computing

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", Consiglio Nazionale delle Ricerche, Via G. Moruzzi 1, 56124, Pisa, Italy
***Corresponding author**: elisa.molinaro@isti.cnr.it

Contents

1. Introduction

The continuous growth of data intensive research and the increasing need for collaboration across scientific disciplines make it essential to provide researchers with integrated environments that combine data, computing, and analytical tools in a secure and efficient way. In this context, D4Science [1] delivers a federated ecosystem of Virtual Research Environments (VREs) that support collaborative scientific work and open access to research resources.

D4Science offers a comprehensive set of services that enable the sharing, management, and analysis of data while ensuring compliance with FAIR [2] (Findable, Accessible, Interoperable, Reusable) and Open Science principles. Through its modular and scalable architecture, it allows research communities to create customised VREs (also referred as Virtual Labs) that integrate datasets, applications, computational resources, and publishing facilities, all accessible through a single web based gateway.

The D4Science Interoperability Framework provides the technological foundation for this ecosystem and ensures that services, data, and tools can operate seamlessly together, regardless of their underlying technologies. Built around key components such as the VRE Gateway, the Identity and Access Management (IAM) system, and the Cloud Computing Platform (CCP), the framework enables secure access, container based deployment, and reproducible workflows.

This document focuses on developer guidance for service

integration within D4Science. It describes practical workflows for onboarding services, containerisation and orchestration practices, authentication and authorisation with OpenID Connect and OAuth2, data access through shared workspaces, and the use of monitoring, auditing, and provenance facilities. Architectural aspects are discussed only to provide context for implementation choices, while comprehensive system design and deployment details remain outside the scope of this document.

2. Interoperability Framework Components

The D4Science Interoperability Framework builds upon a secure, scalable, and modular architecture designed to integrate distributed services and resources within a unified environment. At its core, the framework is supported by the Identity and Access Management (IAM) system, which governs authentication, authorisation, and federated access across all D4Science components.

Once authenticated through IAM, users can seamlessly access the VRE Gateway, Virtual Labs, and all the services within them, including the Analytics Engine for the execution of computational methods and workflows. Each component contributes a distinct capability to the overall ecosystem: the VRE Gateway provides the main access point and collaborative interface; Virtual Labs offer configurable and domain specific workspaces; and the Analytics Engine ensures scalability, reproducibility, and adherence to FAIR principles through containerised and traceable computation.

To support maintainability and automation, all components are managed through continuous integration and deployment (CI/CD) pipelines, via Jenkins [1], ensuring consistent service updates and high operational reliability across environments.

The sections that follow describe these core components in logical order:

- *Identity and Access Management (IAM):* establishes secure and federated access through standard authentication protocols and integration with EGI Check in.
- *VRE Gateway:* provides a centralised access point for D4Science services and collaborative tools.
- *Virtual Labs (VLabs):* offer tailored environments that integrate data, applications, and computational services into unified workspaces.
- *Analytics Engine:* enables scalable, reproducible, and provenance aware execution of computational workflows and analytical processes.

Together, these components deliver a cohesive and interoperable infrastructure where users can securely access, execute, and share data, methods, and applications across diverse research domains.

[1] https://www.jenkins.io

2.1 Identity and Access Management

Identity and Access Management (IAM) is a fundamental component of the framework, ensuring secure, transparent, and user friendly authentication and authorisation across all integrated services. IAM underpins the entire interoperability model, as it allows users from different organisations and domains to access shared services and data resources through a single, trusted mechanism.

The D4Science IAM adopts industry standard protocols such as OAuth2 and OpenID Connect (OIDC) to provide Single Sign On (SSO) across multiple services and Virtual Research Environments (VREs). By relying on token based authentication, users can move between different D4Science services without repeated logins, maintaining a seamless and consistent experience. The IAM system also supports the exchange of verified identity attributes and group memberships, allowing services to make fine grained authorisation decisions based on roles, permissions, and community specific policies. At the heart of the IAM architecture is an open source identity management platform based on Keycloak[2], which provides a complete implementation of OIDC and OAuth2. Through this service, D4Science acts as an identity broker between its internal users and a variety of external identity providers. These include institutional identity federations, the EGI Check in service, and social providers such as Google, LinkedIn, or GitHub. This approach allows users to log in using their preferred credentials while maintaining a secure and traceable identity representation within the D4Science domain.

The IAM system issues short lived JSON Web Tokens (JWT) that encapsulate information about the authenticated user, including username, roles, and groups. These tokens are digitally signed and validated by all authorised backend services, ensuring that authentication data remain verifiable and tamper proof throughout the session lifecycle. Access tokens can be refreshed automatically without re-authentication, improving usability while maintaining strong security controls.

As illustrated in Figure 1, when users access a service, the system redirects them to a secure IAM login page. After successful authentication, the IAM issues an authorisation code that is exchanged for an access token. This token is then used by the service to verify identity and determine the corresponding access rights. This mechanism follows the OAuth2 authorisation *code grant* flow, which ensures that credentials are never exposed to client applications or transmitted insecurely.

The IAM also supports delegated authorisation, allowing a service to act on behalf of a user when interacting with other components. For instance, a workflow execution launched from a Virtual Lab may access data from the Shared Workspace or invoke analytical methods in the Analytics Engine using delegated tokens. This delegation ensures that all actions remain traceable to the initiating user, reinforcing accountability and reproducibility.

To simplify access management, users are organised into

[2] https://www.keycloak.org

Figure 1. Illustration of the IAM login and authorisation flow within the D4Science ecosystem.

groups and roles that reflect their participation in specific VREs or projects. Role based access control (RBAC) mechanisms allow VRE managers to assign permissions such as data upload, method execution, or administrative rights. These configurations are synchronised dynamically across D4Science services, ensuring that access rules remain consistent and automatically enforced.

Integration with EGI Check-in extends the reach of the IAM service to users belonging to national or thematic research infrastructures across Europe. This integration facilitates cross institutional collaboration and enables interoperability with other EOSC aligned systems, ensuring that D4Science remains part of the broader European Open Science ecosystem.

From a service developer perspective, IAM provides an interface for integrating new services within the D4Science federation. Developers can register their applications as trusted clients, obtain credentials, and implement token verification through standard libraries compatible with OAuth2 and OIDC. This uniform approach simplifies onboarding and guarantees compliance with the security policies of the platform.

The deployment of the IAM system followed a structured and modular process. It included the configuration of the D4Science VRE infrastructure, integration of containerised services such as JupyterHub and ShinyProxy, and implementation of continuous integration and deployment (CI/CD) pipelines for controlled updates. The result is a secure, flexible, and scalable identity layer that guarantees both user trust and operational efficiency.

2.2 VRE Gateway

All functionalities of the D4Science Service Interoperability Framework are accessible programmatically and via Web User Interfaces. For the latter they are accessible through the VRE Gateway, a user friendly web portal that serves as the main access point to the D4Science ecosystem. The Gateway provides end users with a coherent and integrated entry to Virtual Research Environments (VREs), also referred to as Virtual Labs (VLabs), enabling them to collaborate, share data, and execute computational workflows within a unified environment. D4Science hosts VRE Gateways for different scientific domains [3, 4, 5]. An always up-to-date list of D4Science supported VRE Gateway is available at *https://www.d4science.org/gateways*.

The VRE Gateway plays a central role in the architecture of the framework, acting as both the user interface layer and the orchestration point for services and resources distributed across the infrastructure. It integrates authentication and authorisation provided by the Identity and Access Management (IAM) system and connects seamlessly with back end services such as the File Sync and Share (former Workspace), the Analytics Engine, and collaborative communication tools. This integration allows users to move effortlessly from one service to another, maintaining full access control and traceability.

Beyond providing access, the Gateway embodies D4Science's principles of openness, usability, and interoperability. It is designed to support both scientific communities and service developers by providing intuitive interaction mechanisms and consistent access to data, tools, and computational resources. Its modular architecture allows new functionalities to be added or customised to suit the needs of different communities and projects.

The VRE Gateway offers the following key functionalities:

- *Gateway Home:* a central hub where users can collaborate with team members by posting messages, sharing updates, and commenting on posts. The home area acts as the community space for announcements, discussion threads, and cross project communication.
- *Virtual Labs (VLabs):* web based, on demand environments accessible from within the Gateway that provide tailored services, datasets, and shared tools to support domain specific research activities and demonstrators.
- *File Sync and Share:* a collaborative area for storing and sharing files such as project deliverables, presentations, and datasets. It ensures secure storage, versioning, and accessibility for data driven collaboration and is directly integrated with analytical tools such as JupyterLab, RStudio, and the Analytics Engine.
- *User Management:* authorised users (VLab Managers) can manage VLab membership by approving new users, assigning or withdrawing roles, removing users, and sending communications to members. Role based permissions are automatically synchronised with the IAM, ensuring secure and consistent access control.
- *Members Area:* a directory that enables VLab members to view and connect with others in the VRE, fostering collaboration and facilitating networking across projects and disciplines.

The VRE Gateway simplifies user interaction with the D4Science infrastructure, consolidating a complex array of services into an intuitive and consistent web experience. It supports multilingual interfaces, responsive design, and accessibility standards, ensuring that users can efficiently work from different devices and contexts.

The Gateway is centred around the Home page (Dashboard), accessible after login. The Dashboard provides an integrated overview of the entire D4Science services available on it, allowing users to monitor activities, navigate across VREs/VLabs, and access shared data and resources in real time. It brings together project management, collaboration, and computation under a single interface, supporting both everyday research tasks and large scale analytical workflows. Figure 2 and Figure 3 show respectively the current SoBigData [6] VRE Gateway and the Blue-Cloud2026 Gateway Homes, which serve as the main landing pages providing access to collaborative spaces and shared functionalities available to users.

Specifically, the Dashboard Page includes:

- *Access to VLabs:* direct links to Virtual Labs and their descriptions, allowing users to enter domain specific workspaces preconfigured with the necessary tools, datasets, and applications.
- *User File Sync and Share:* a cloud based file system that supports folder management and sharing of various data types such as binary files, tabular data, workflows, and computational results. Each user can manage personal, shared, or VRE wide folders:
  - Private Folders: accessible only by the owner;
  - Shared Folders: accessible to selected users or groups;
  - VRE Folders: available to all members of a given VRE/VLab.

  The File Sync and Share service (Workspace) is fully integrated with the D4Science APIs, ensuring seamless interaction between data storage, computational tools, and external repositories. Stored data can be directly processed through the Analytics Engine, enabling a frictionless workflow from data upload to analysis and visualisation.
- *Private Messages:* an integrated email like service that allows users to communicate efficiently. It supports the exchange of large or complex objects directly from the File Sync and Share area without redundant data transfer.
- *User Profile and Account:* a personal area for managing profile information, institutional affiliations, and notification preferences. Users can customise their workspace and control visibility settings, ensuring flexibility in multi community environments.
- *Notifications and Activity Stream:* a unified alert system that provides real time updates about new uploads, messages, or workflow results. These notifications are context aware and help users maintain awareness of project progress and team activity.

Through its comprehensive and extensible design, the VRE Gateway represents the core interaction layer between users and the D4Science infrastructure. It harmonises communication, computation, and data access, providing a unified interface that promotes openness, reproducibility, and effective collaboration across scientific communities.

2.3 Virtual Labs

Virtual Labs (VLabs), accessible directly from the D4Science VRE Gateway, represent one of the most distinctive components of the D4Science ecosystem. They offer web based, on demand environments that integrate external systems such as data repositories, storage services, and computational infrastructures into cohesive online workspaces. Each VLab provides a unified and controlled setting where data, methods, and users interact according to shared access rules, fostering reproducibility, collaboration, and cross disciplinary innovation.

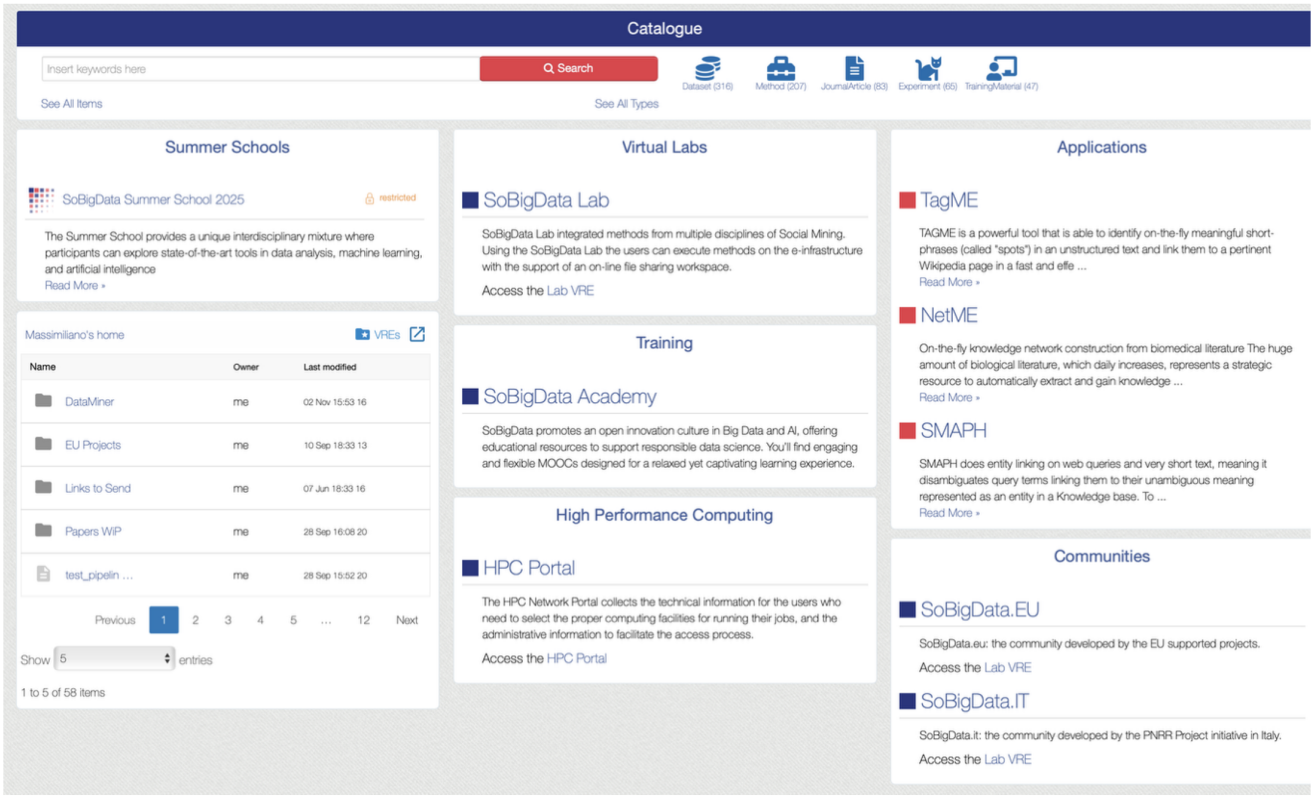

Figure 2. The SoBigData VRE Gateway Home.

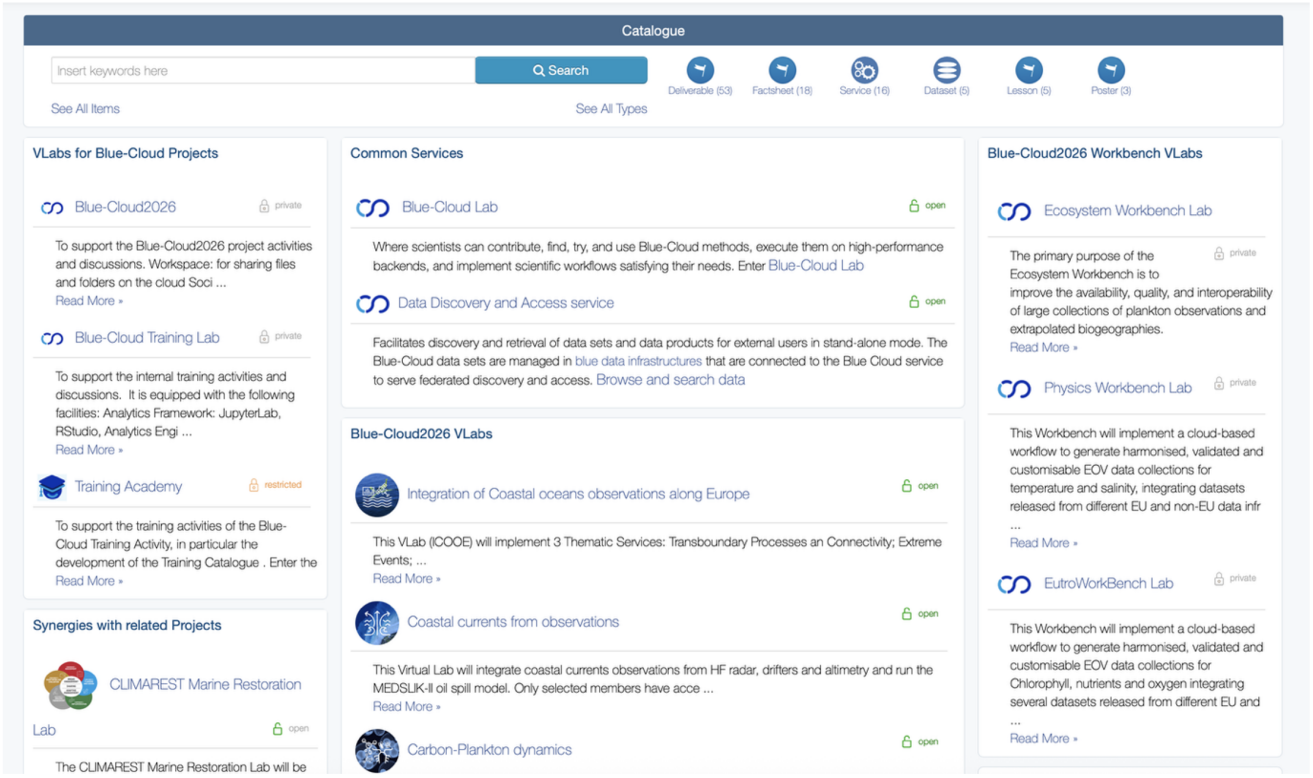

Figure 3. The Blue-Cloud2026 VRE Gateway Home.

A VLab is tailored to a specific scientific community or project. It brings together heterogeneous resources, datasets, tools, workflows, and computing services, and exposes them through a harmonised web interface. The underlying D4Science architecture abstracts the complexity of resource management and federation, allowing end users and developers to work within an integrated, secure, and reproducible digital ecosystem.

VLabs unify distributed resources and expose them as a common, shared environment, enabling users to:

- *Tailored Services:* each VLab is configured to meet the specific needs of its user community, supporting domain specific workflows for data analysis, computation, modelling, or knowledge sharing.
- *Integration of Resources:* external systems such as data repositories, cloud storages, and high performance computing infrastructures are seamlessly integrated and made accessible as part of the same workspace.
- *Collaboration and Sharing:* VLabs act as collaborative environments where users can share datasets, tools, and results, supporting joint activities, co creation of knowledge, and open science practices.
- *Transparency and Provenance:* every operation within a VLab, data upload, workflow execution, or file sharing, is automatically tracked, ensuring full traceability and compliance with FAIR and reproducibility principles.

By integrating VLabs into the D4Science framework and making them accessible through the VRE Gateway, users can focus on their scientific and development tasks without being burdened by infrastructure complexity. This approach promotes efficiency, reduces duplication of effort, and ensures that each community operates within a structured and interoperable environment.

Each VLab operates as an autonomous but interoperable node within the broader D4Science federation. It benefits from shared platform services such as the Identity and Access Management (IAM), the File Sync and Share (former Workspace) system, and the Analytics Engine. This ensures that authentication, authorisation, data storage, and computational execution are handled consistently across all VLabs. The modularity of this approach allows new VLabs to be created, replicated, or customised with minimal configuration effort.

Figure 4 shows an exemplar VLab available within the D4Science environment. This VLab serves as a fully functional demonstration of the D4Science Interoperability Framework capabilities, offering users a unified and collaborative workspace that supports both research and service development activities.

The VLab Home provides an intuitive and user friendly interface designed to foster collaboration and enhance productivity. Key features of the VLab include:

- *File Sync and Share (former Workspace):* a dedicated cloud storage area accessible to all VLab members, enabling them to upload, share, and organise files and datasets relevant to their collaborative projects. The File Sync and Share service ensures that all resources are easily accessible, version controlled, and centrally managed.
- *Posts Feed:* a dynamic space for communication among VLab members, allowing users to post messages, comment on updates, and interact with contributions from their peers. The feed supports lightweight community coordination and informal project communication.
- *VLab Overview:* a concise abstract displayed on the home page, summarising the purpose, objectives, and available functionalities of the VLab, guiding both new and existing users in navigating the workspace.
- *Navigation Bar and Tools:* a clear menu structure providing access to the main services and applications available to members, including:
  - JupyterLab: an interactive environment for developing and executing notebooks, supporting data analysis, visualisation, and reproducible computation;
  - RStudio: an integrated development environment for R, supporting statistical analysis and the creation of shareable research outputs;
  - Analytics Engine: a platform for executing advanced computational workflows and large scale data analysis. It supports containerised execution environments, ensuring scalability, portability, and reproducibility;
  - Members Directory: a searchable list of all VLab members, facilitating networking, collaboration, and transparency;
  - How To Documentation: a structured repository of user guides, tutorials, and developer notes that assist members in exploiting the VLab functionalities efficiently.

Each VLab includes a well defined governance model that identifies specific roles and responsibilities. VLab Managers can approve membership requests, assign or revoke roles, configure access rights, and manage communication among participants. These actions are synchronised with the IAM, ensuring that all operations respect the security and policy framework defined at platform level. Users with appropriate permissions can also integrate additional data sources or deploy domain specific applications, extending the capabilities of their VLab without compromising interoperability.

From a developer perspective, VLabs act as the main integration layer for new services within the D4Science ecosystem. Developers can register new tools, expose APIs, and

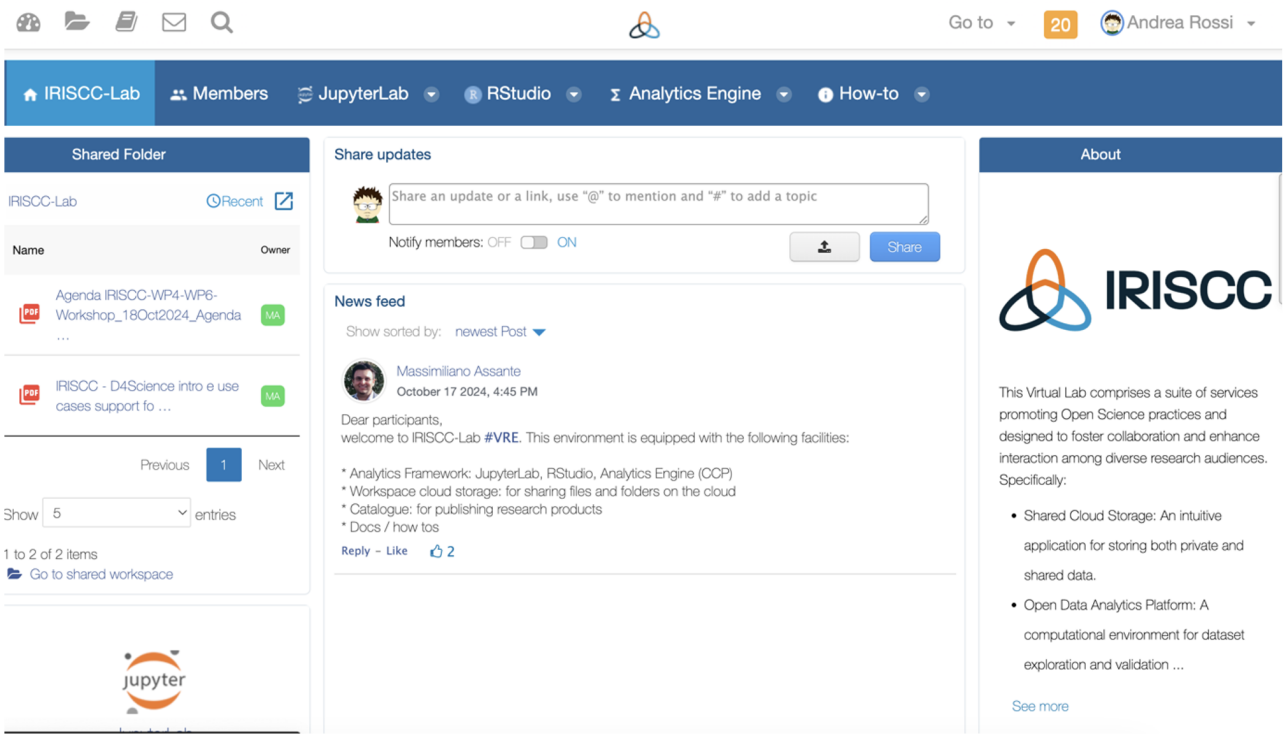

Figure 4. An exemplar Virtual Lab as a functional demonstration.

connect containerised applications through the VLab architecture, leveraging the shared infrastructure and federation mechanisms already in place. This model lowers the entry barrier for service integration and ensures compliance with the framework's security, reproducibility, and FAIR guidelines.

2.4 Analytics Engine

The Analytics Engine, powered by the Cloud Computing Platform (CCP) [7], is a core component of the D4Science framework, enabling scalable, reproducible, and traceable execution of computational methods and analytical workflows. Built on a microservice oriented architecture, the Analytics Engine supports flexibility, interoperability, and performance, allowing researchers and developers to deploy, execute, and manage algorithms seamlessly while adhering to FAIR principles.

The system provides an execution environment where computational methods can be defined, shared, and reused across Virtual Labs (VLabs). It integrates storage, authentication, and provenance tracking through a unified interface and an automated backend. This design abstracts the technical complexity of distributed computing, letting users focus on scientific research and method development rather than infrastructure management.

Key features of the Analytics Engine include the *methods importer tool*, which simplifies the integration of new computational methods by automatically generating containerised descriptors, and the *execution lifecycle tracker*, which records every stage of execution, from submission to output generation, to ensure transparency and traceability. Researchers can monitor ongoing workflows through a real time *execution monitor*, which provides live feedback and performance metrics, while the platform archives all configurations and inputs to support reproducibility and long term verification.

The flexible and inherently distributed nature of the CCP execution infrastructures allows users to design and run methods that align computational workloads with the most suitable resources, fostering optimisation based on data locality, computational demand, and infrastructure capabilities. By accommodating a wide spectrum of execution contexts, from High Performance Computing (HPC) systems to experimental environments, the CCP enables distributed and scalable collaboration, ensuring reproducibility and interoperability across diverse research domains.

D4Science offers a native execution infrastructure based on Docker Swarm and enriched by a dedicated image registry implemented through Harbor. When designing methods for these container based environments, scientists face virtually no limitations regarding programming languages, environments, versions, or dependencies. By encapsulating computational methods and their dependencies into isolated containers, the CCP ensures reproducibility, portability, and scalability. This approach allows researchers to deploy methods without compatibility issues, maintaining consistent execution conditions regardless of underlying hardware or software variations. In addition, the use of containers significantly reduces the overhead typical of traditional virtualisation technologies, improving the efficiency of complex analytical workflows.

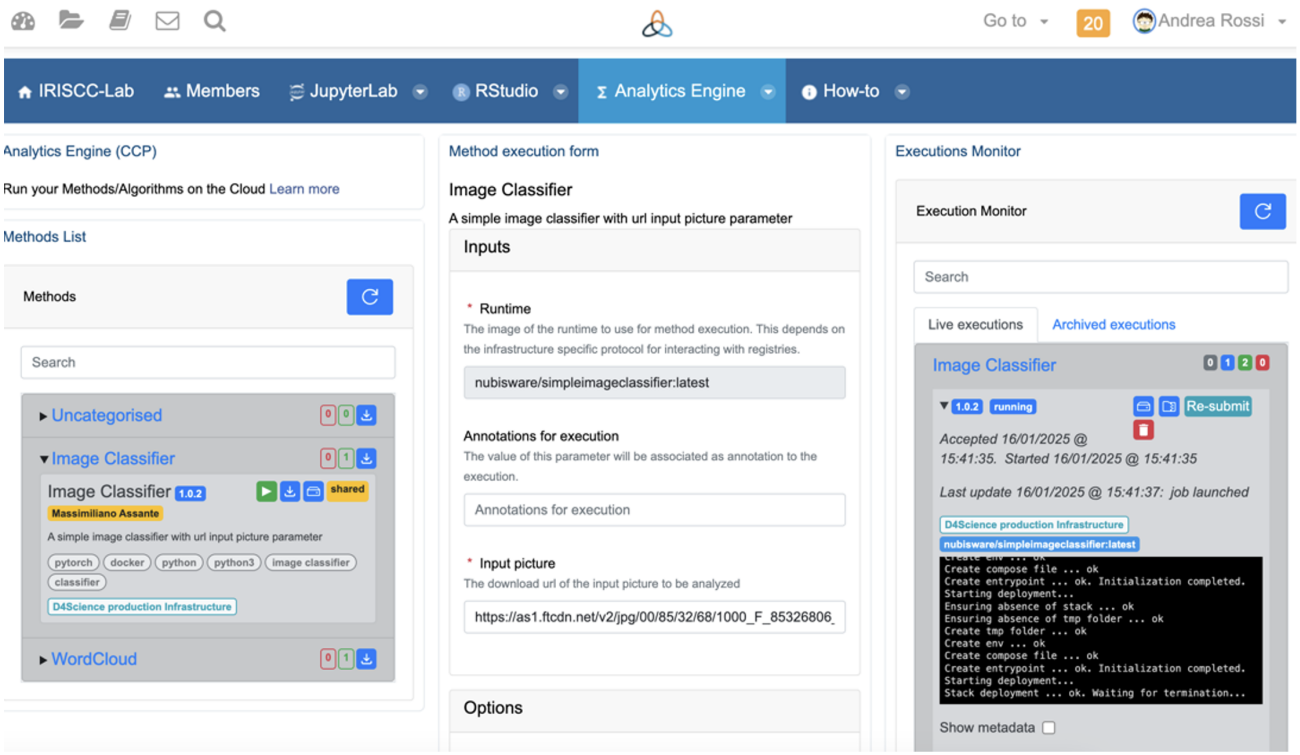

Figure 5. Example of a Cloud Computing Platform method execution.

By integrating containerisation, flexible infrastructure management, and robust automation tools, the CCP acts as a key enabler of large scale distributed computing. Its architecture promotes innovation, scalability, and reproducibility, supporting the Open Science vision that underpins the D4Science ecosystem.

Within this architecture, users can take advantage of interactive and programmatic environments that are natively supported by D4Science, including:

- *JupyterLab:* a browser based interactive development environment where users can combine live code, visualisations, and documentation. Integrated with the File Sync and Share (former Workspace) system, JupyterLab allows direct access to datasets for analysis, modelling, and visualisation, while executing computations on remote D4Science resources.
- *RStudio:* a professional development environment for R that supports statistical analysis, data modelling, and reproducible reporting. RStudio sessions within D4Science run in isolated containers managed by the Analytics Engine, ensuring consistent dependencies, security, and performance while providing seamless access to shared data in File Sync and Share.
- *RShiny Applications:* interactive web applications developed in R, Python or Julia and deployable directly within D4Science VLabs. Developers can containerise RShiny apps, making them accessible through the VRE Gateway as dynamic, web based tools. These applications can retrieve data from File Sync and Share, trigger computational workflows via the Analytics Engine APIs, and present interactive visual analytics to the research community.

This tight integration between analytical tools and the D4Science infrastructure enables complete, end to end workflows, from data discovery and preparation to computation, visualisation, and dissemination, within the same federated environment. The Analytics Engine provides both graphical interfaces and RESTful APIs, enabling users and external systems to submit jobs, retrieve outputs, and access provenance information programmatically. This interoperability layer facilitates automation and cross platform analytical pipelines, extending D4Science's role as an enabling infrastructure for large scale, collaborative science.

2.4.1 Provenance Management

Provenance management within the Analytics Engine ensures complete traceability and reproducibility of every analytical process. Each execution is automatically logged, recording metadata such as the user identity, input datasets, parameters, software versions, runtime configuration, and output artefacts. This metadata allows researchers to reproduce experiments, verify results, and understand how each dataset or analytical output was derived.

To structure and standardise provenance information, the Analytics Engine adopts the PROV Ontology (PROV-O), an

OWL2 based formalisation of the W3C PROV Data Model. This ontology supports a precise representation of entities, activities, and agents involved in computational workflows, ensuring alignment with international standards for scientific data management and exchange.

Aligned with FAIR principles, the Analytics Engine automatically generates and stores a PROV-O compliant XML file alongside each execution within the user's File Sync and Share area. These records include detailed descriptions of input and output datasets, parameters, and configurations, allowing users to replicate computational tasks with full reproducibility.

In addition to its technical features, the CCP strongly aligns with the principles of Open Science, ensuring that all scientific outputs are transparent, repeatable, and reusable. Every execution is accompanied by detailed provenance documentation that captures the origin, transformations, and outcomes of data. This not only supports reproducibility but also establishes a verifiable lineage for scientific results, reinforcing trust and attribution in collaborative research. The integration of dynamic resource allocation further allows users to scale computational resources based on demand, making the platform suitable for projects of any size or complexity.

In summary, the Analytics Engine provides the technological foundation for executing, tracking, and reproducing computational research within D4Science. By combining container based execution, flexible infrastructure management, and provenance aware analytics, also with environments such as JupyterLab, RStudio, and RShiny, it transforms the VRE into a fully integrated analytical ecosystem, bridging Open Science principles with operational efficiency and scientific excellence.

3. Interoperability Framework Guidance

The D4Science Service Interoperability Framework offers developers a flexible, standards based environment for integrating new tools and deploying custom services within VREs/VLabs. Developers can leverage the framework to build applications, methods for the analytical engines, or data services that exploit the platform's existing capabilities for authentication, storage, computation, and collaboration.

The framework supports a wide range of integration scenarios, enabling developers to select the most appropriate approach according to the nature of the service, the programming environment, and the expected usage patterns. Three main categories of service deployment are available:

- *JupyterHub:* this service enables the deployment of dedicated and customised notebook servers for various environments, including JupyterLab, RStudio Server, and Pluto.jl. JupyterHub allows each user or group to operate in isolated execution spaces with preconfigured libraries and dependencies. Developers can define custom Docker images to extend notebook environments with additional tools, ensuring that data analysis and modelling workflows run consistently across all sessions. Integration with the File Sync and Share (former Workspace) system allows users to access their datasets directly from within notebooks and to store outputs and derived artefacts automatically.

- *ShinyProxy:* this service facilitates the deployment of interactive web applications, such as RShiny apps or Dash applications developed in R or Python. ShinyProxy provides a lightweight gateway that launches each application instance inside its own container, offering full isolation, scalability, and secure user management through the D4Science Identity and Access Management (IAM) system. Developers can publish domain specific visualisation dashboards, simulation tools, or analytical interfaces that run on demand and are accessible directly from the VRE Gateway. The integration with D4Science authentication ensures that applications inherit the same access policies and user roles defined at platform level.

- *Custom Containerised Services:* beyond JupyterHub and ShinyProxy, developers can deploy fully customised services using containerisation. By packaging applications and their dependencies into Docker containers, developers can integrate virtually any type of service into the D4Science infrastructure. These services can then be orchestrated through Docker Swarm, the native D4Science container management layer. Docker Swarm has been chosen for its simplicity, ease of configuration, and reduced operational overhead, providing a more lightweight and developer friendly approach compared to Kubernetes. It supports automatic load balancing, fault tolerance, and horizontal scaling, enabling rapid deployment and efficient lifecycle management of containerised services.

Each of these approaches leverages the same foundational components of the D4Science ecosystem, containerisation, orchestration, standardised Identity and Access Management (IAM), and continuous integration and deployment (CI/CD) pipelines. These technologies collectively streamline development and maintenance by automating service provisioning, updates, and monitoring, thus significantly reducing operational overhead.

The integration workflow for developers typically involves the following stages:

1. Designing the application or analytical service using container compatible technologies.

2. Building a Docker image that encapsulates all dependencies, runtime libraries, and configuration parameters.

3. Publishing the image to the D4Science Harbor registry, where automated vulnerability scanning and version tagging ensure security and traceability.

4. Registering the service within the VRE or VLab via the Activity Tracker to enable proper orchestration and IAM authorisation.

5. Validating the deployment through the Analytics Engine or the VRE Gateway interface.

Once deployed, these services can seamlessly interact with the D4Science ecosystem by accessing shared data through the File Sync and Share component, using tokens issued by the IAM for secure API calls, and integrating with the Analytics Engine for large scale computational execution. This level of interoperability ensures that services developed within D4Science can collaborate with other components while maintaining compliance with FAIR and Open Science principles.
By adopting this modular and container based approach, D4Science empowers developers to focus on innovation rather than infrastructure management. The framework simplifies deployment, enhances reproducibility, and guarantees a uniform user experience across all VREs, making it an ideal environment for building scalable and sustainable scientific services.

3.1 JupyterHub: Notebook Servers for JupyterLab, RStudio, and Pluto.jl

JupyterHub within D4Science provides a flexible and scalable framework for hosting interactive computational environments that meet the diverse needs of researchers, analysts, and developers. Through its integration with the D4Science Virtual Research Environment infrastructure, JupyterHub enables users to create and manage isolated notebook servers that are automatically connected to the platform's authentication, storage, and computational resources. The service is based on the open source *JupyterHub* project (`https://jupyter.org/hub`) and extends beyond JupyterLab to include other interactive environments such as RStudio Server and Pluto.jl[3], making it an essential tool for a wide range of data driven and computational workflows.
When users access JupyterHub from within a Virtual Lab, they are authenticated through the Identity and Access Management system and automatically assigned a dedicated server instance. This instance operates in an isolated containerised environment with customised configurations, ensuring both security and reproducibility. Each notebook server can be preconfigured with specific libraries, kernels, or runtime dependencies corresponding to the scientific domain of the Virtual Lab. This approach guarantees a consistent computational setup for all users, eliminating conflicts and dependency mismatches that often arise in shared infrastructures.
JupyterHub seamlessly integrates with the File Sync and Share component, providing direct access to user files, datasets, and project outputs. Users can open, edit, and save notebooks directly from their shared directories, maintaining a consistent and traceable research workflow. Files produced within notebooks are synchronised automatically, ensuring that results and scripts are preserved and accessible to collaborators through the same shared environment.
A distinctive feature of the D4Science JupyterHub deployment is its ability to support multiple interactive environments under a unified management interface. This includes JupyterLab, RStudio Server, and Pluto.jl, each suited to different analytical and development needs.

- *JupyterLab:* an advanced, browser based interface for working with Jupyter notebooks, code, and data. It supports multiple programming languages, interactive widgets, and real time visualisation tools, making it ideal for Python based workflows, exploratory analysis, and machine learning experiments. The environment also supports collaborative editing, allowing multiple users to work simultaneously on the same notebook within a shared Virtual Lab context.

- *RStudio Server:* a professional development environment for R users, fully integrated into D4Science through JupyterHub. These servers are preloaded with domain specific R packages and can be extended to include custom libraries or dependencies as required by individual Virtual Labs. Users can perform statistical modelling, data visualisation, and reporting directly within the VRE. Since all data and outputs are stored in the File Sync and Share system, analytical workflows remain consistent and reproducible across sessions.

- *Pluto.jl:* an interactive notebook system for the Julia programming language, offers reactive notebooks that update results immediately when code or data are modified. It is particularly suited for educational and exploratory computational work, where responsiveness and clarity of results are essential. Within D4Science, Pluto.jl runs as a managed JupyterHub service, benefiting from the same authentication, data access, and storage features available to all other environments.

JupyterHub also supports dynamic resource allocation and monitoring, enabling the D4Science infrastructure to automatically assign computational resources based on workload and user demand. This ensures that notebook sessions are responsive and scalable, adapting to the available cluster capacity. Through integration with the Analytics Engine, notebooks can trigger the execution of large scale computations, delegating intensive processing to the underlying Cloud Computing Platform while maintaining an interactive workflow for the user.

3.1.1 Custom JupyterLab Notebook Images

It is worth to note that developers can extend JupyterHub by designing new container images that provide specific software stacks for their communities. Each image can include domain oriented libraries (e.g. PyTorch), analytical frameworks, or specialised scientific tools. Images are stored and versioned

[3] https://plutojl.org

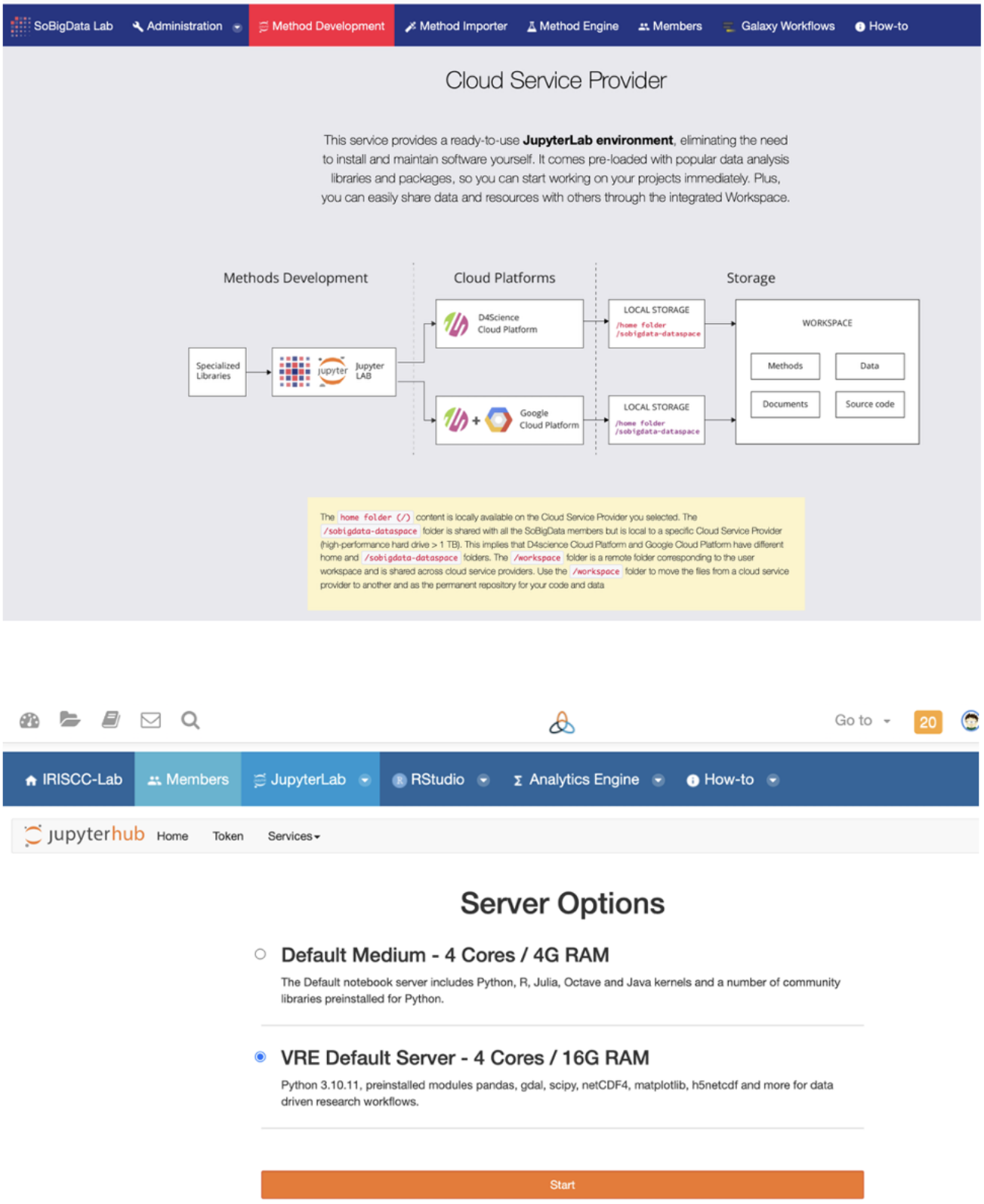

Figure 6. Example of JupyterLab interface showing two available notebook server configurations.

in the D4Science Harbor registry [4], ensuring secure management, traceability, and reproducibility. Once published, these

[4] https://harbor.d4science.org

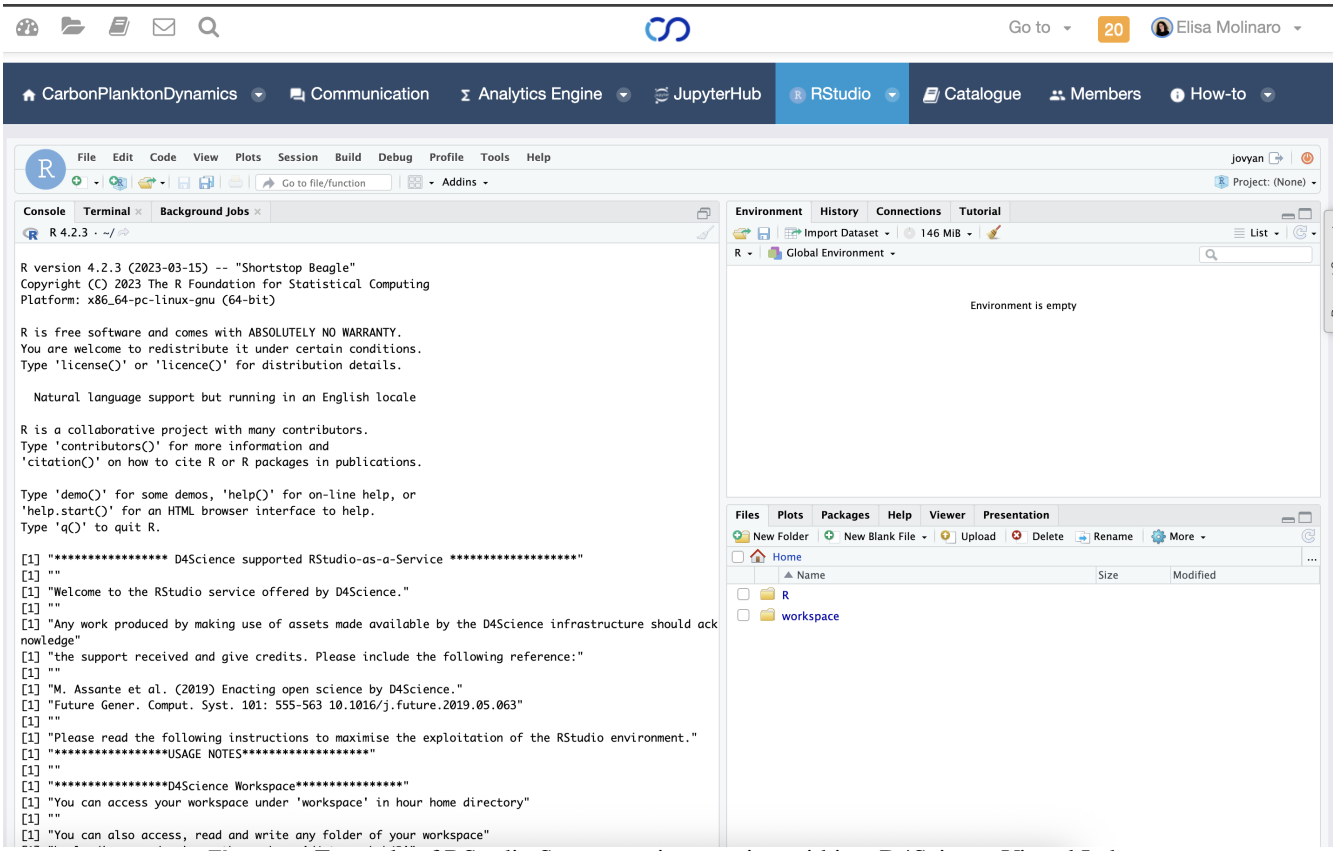

Figure 7. Example of RStudio Server session running within a D4Science Virtual Lab.

environments can be made available to all members of a Virtual Lab or restricted to specific projects through the Identity and Access Management authorisation model.

This modular and extensible design makes JupyterHub a cornerstone of interactive computing in D4Science. By supporting reproducible notebook environments for Python, R, and Julia users, it unifies the practices of data analysis, modelling, and experimentation under a common infrastructure.

3.2 ShinyProxy: Interactive Applications for R and Python

ShinyProxy within D4Science provides a robust and flexible service for deploying interactive web applications that support advanced data exploration, monitoring, and visualisation. This component enables developers to create and publish interactive tools directly within Virtual Labs, offering researchers and decision makers dynamic and accessible ways to work with complex datasets. The service is based on the open source *ShinyProxy* technology (`https://www.shinyproxy.io`), which extends the popular RShiny framework to enterprise and research scale environments through containerisation and orchestration. In addition to RShiny applications, D4Science also supports the deployment of interactive Python applications developed with the *Dash* framework (`https://dash.plotly.com`), ensuring that both R and Python communities can benefit from the same infrastructure.

When an application is launched through ShinyProxy, it is executed inside an isolated container with all required dependencies, libraries, and data sources preconfigured. Each instance runs independently from others, ensuring full isolation and resource control while maintaining consistent performance. Users authenticate through the D4Science Identity and Access Management system, which enforces secure authorisation and role based access. This integration allows ShinyProxy to inherit the same federated authentication model and security standards used across the entire D4Science platform, eliminating the need for developers to handle authentication or data access logic in their applications.

The ShinyProxy service is particularly valuable for building and sharing interactive dashboards, monitoring tools, and simulation platforms that visualise scientific or operational data in real time. For example, RShiny applications can display analytical results, trends, and indicators through dynamic charts or maps, while Dash applications can implement complex interactive visualisations for exploratory analysis, machine learning results, or climate simulation outputs. Both frameworks support flexible layouts, parameter selection widgets, and interactive filtering, allowing users to engage directly with data and results without programming knowledge.

Because the D4Science infrastructure manages the foundational services such as container orchestration, access con-

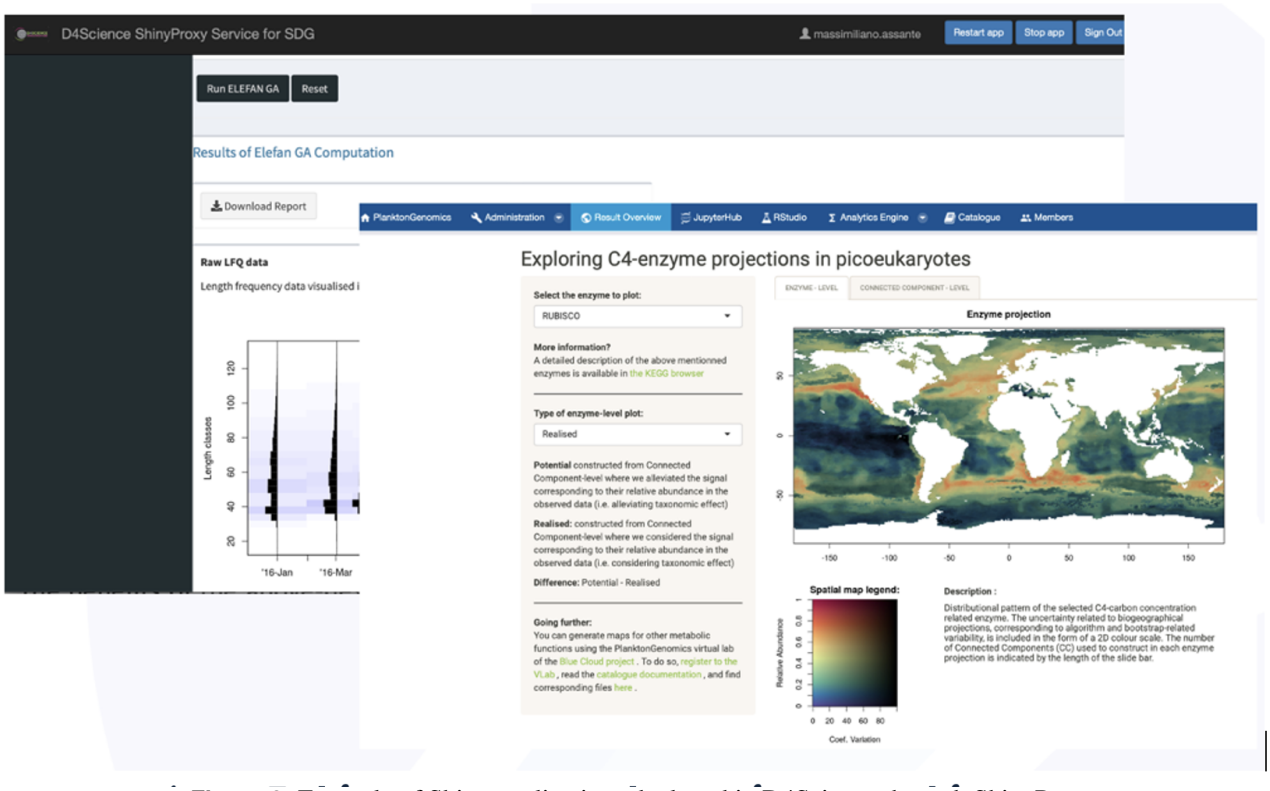

Figure 8. Example of Shiny applications deployed in D4Science through ShinyProxy.

trol, and monitoring, developers using ShinyProxy can focus exclusively on the scientific and functional aspects of their applications. This delegation of system level responsibilities significantly reduces development overhead, shortens the deployment cycle, and ensures compliance with FAIR and Open Science principles. Applications deployed in this way benefit from consistent user management, secure communications, and automatic scalability based on usage.

ShinyProxy is fully integrated with the D4Science container orchestration layer, which relies on Docker Swarm for managing application instances. This integration allows ShinyProxy to dynamically scale the number of running containers depending on concurrent user requests. When the demand increases, new instances are created automatically; when it decreases, idle instances are gracefully terminated, optimising resource utilisation while maintaining responsiveness. Such elasticity makes it possible to support computationally demanding visualisations or simulations without compromising performance for other users.

Each deployed application can access the D4Science File Sync and Share system to read and write datasets, and can interact programmatically with other services, including the Analytics Engine, through RESTful APIs. This interoperability enables developers to design complete workflows that combine data access, computation, and interactive visualisation within a single environment. For example, an RShiny or Dash application may invoke a remote method on the Analytics Engine, retrieve the results, and immediately render them as a dynamic visualisation for user exploration.

Figure 8 illustrates examples of Shiny applications deployed through ShinyProxy within D4Science. These applications demonstrate how scientific teams can transform complex data analysis workflows into intuitive and interactive interfaces, fostering collaboration and transparency in research.

The main capabilities of ShinyProxy in D4Science can be summarised as follows:

- *Support for Multiple Frameworks:* ShinyProxy natively supports both RShiny and Dash applications, enabling developers to select their preferred language and framework while relying on the same secure infrastructure.
- *Customisation and Flexibility:* Developers can design highly specialised applications by preloading containers with the required datasets, software dependencies, and configuration files. Each application can be tailored to the domain or project needs of a Virtual Lab.
- *Secure Deployment:* All applications are executed within containerised environments integrated with the Identity and Access Management system, ensuring isolation, access control, and consistent compliance with platform security standards.
- *Scalability and Monitoring:* The infrastructure automatically scales active instances based on user demand,

maintaining responsiveness during peak activity. Administrators can monitor performance and usage through integrated dashboards and logs.

Through these features, ShinyProxy extends the analytical and collaborative capabilities of D4Science, bridging computation and communication. It allows developers to convert analytical results into interactive, shareable tools that empower scientists, policy makers, and stakeholders to explore and interpret data intuitively. By integrating R and Python visualisation technologies within a secure, scalable, and reproducible infrastructure, D4Science enhances the accessibility and impact of Open Science.

3.3 Custom Containerised Services

The D4Science Virtual Research Environment allows developers to deploy fully customised containerised services, providing maximum flexibility for integrating specialised tools, applications, and workflows. By adopting containerisation, developers can encapsulate an application together with all its dependencies, configuration files, and runtime environment into a self contained software unit. This ensures consistent behaviour across heterogeneous infrastructures, an essential requirement in research computing where software stacks are frequently complex and variable.

Containerisation in D4Science enables service developers to create portable, reproducible, and secure applications that can be deployed and scaled automatically within the Virtual Research Environment. Each service runs in an isolated environment, preserving system stability while allowing fine grained access control through the Identity and Access Management (IAM) system. This design reduces operational complexity and promotes FAIR and Open Science principles by guaranteeing that services remain reproducible and accessible over time.

3.3.1 Container Images, Registry, and Engine

The integration of containerised applications relies on three core elements: container images, container engines, and container registries. As shown in Figure 9, container images encapsulate all the required binaries, libraries, and configuration files necessary for the service to execute reliably across different environments. This encapsulation enhances reproducibility and avoids version conflicts. D4Science employs Docker[5] as its primary container engine, providing a mature, stable, and well documented ecosystem that supports both command line and programmatic workflows through APIs. Developers can also perform local testing using tools such as Docker Compose to simulate service behaviour before deployment.

Container images are stored and managed within secure registries. D4Science supports both public registries, such as Docker Hub (`https://hub.docker.com`), and its dedicated Harbor[6] based registry available at `https://harbor.d4science.org`. The Harbor registry adds advanced features including vulnerability scanning, version tagging, and role based access control. This setup allows developers to manage image lifecycles efficiently while ensuring compliance with security requirements. Private repositories can also be created for project specific or proprietary applications, guaranteeing that sensitive images are stored securely. Together, these technologies form a robust and scalable foundation for container management within D4Science.

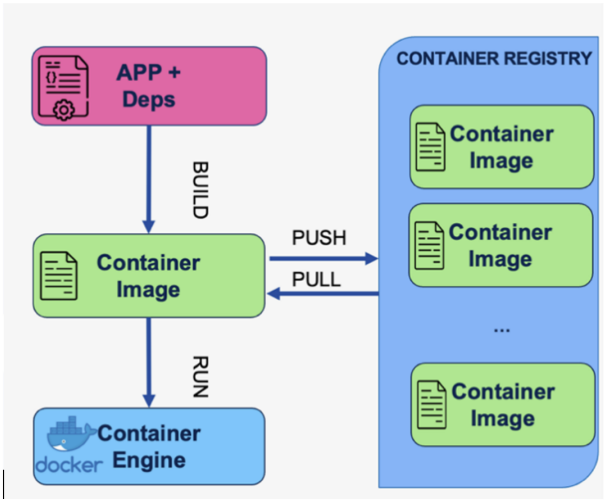

Figure 9. Schematic representation of container images, registry, and engine integration within the D4Science environment.

3.3.2 Container Orchestration with Docker Swarm

To operate containerised services at scale, D4Science uses orchestration systems such as Docker Swarm and, where required, Kubernetes. Docker Swarm has been adopted as the main orchestration layer for service developers due to its simplicity, lightweight footprint, and seamless integration with the existing D4Science infrastructure. It automates essential tasks including service deployment, scaling, load balancing, and health monitoring, ensuring reliability and high availability within the VRE.

Running services within the Swarm cluster follows a structured and well defined workflow. This guarantees consistency and stability across environments while simplifying maintenance and updates. When building container images, following best practices improves both security and performance. Developers are encouraged to:

- avoid unnecessary privileges and use trusted base images;
- exclude sensitive information such as credentials or tokens;
- minimise image size to reduce transfer and startup times;

[5] https://www.docker.com
[6] https://goharbor.io

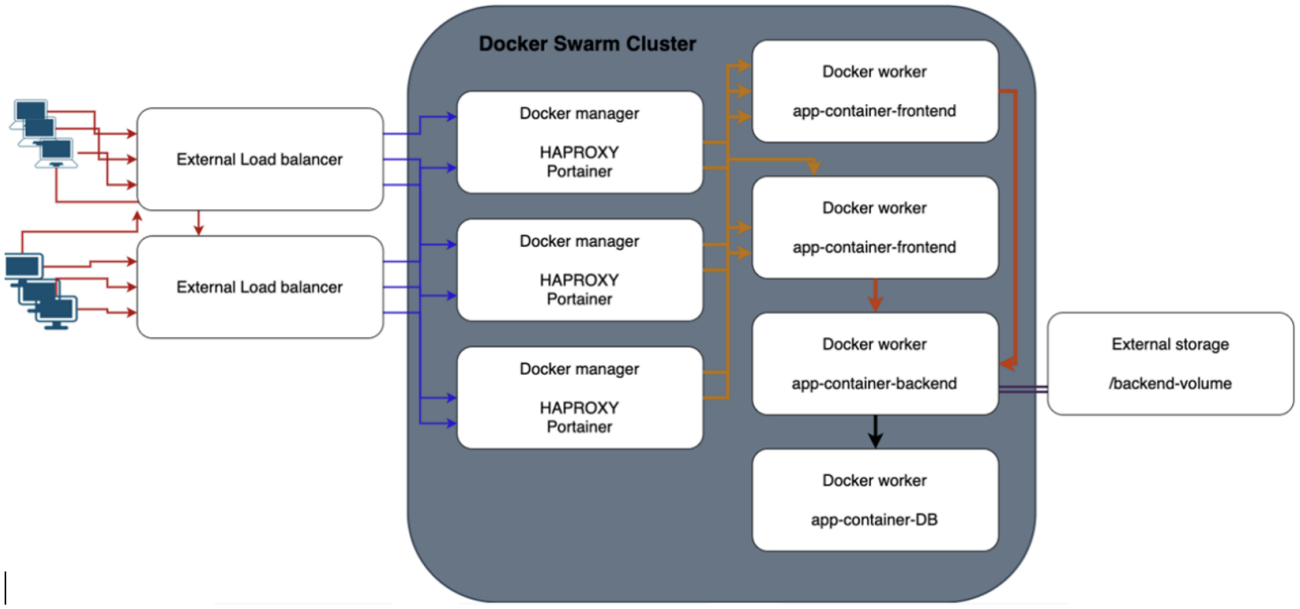

Figure 10. Logical view of the container running environment and data flow in the D4Science VRE.

- consolidate instructions in the Dockerfile to limit the number of layers;
- perform vulnerability scans using tools like Docker Scan or Clair[7];
- define health checks to ensure that services remain operational.

Once built, images are published to a registry. Public registries are suitable for open source applications, whereas the Harbor registry is recommended for private or collaborative projects requiring controlled access and auditing. Harbor performs automated vulnerability checks and supports detailed role management, allowing administrators to define contributor permissions per project. Version tagging provides clear traceability of image evolution, enabling reproducibility and simplifying rollback if necessary.

3.3.3 Container Runtime Environment in the VRE

The runtime environment for containers in D4Science is designed to guarantee secure, scalable, and efficient operations. As illustrated in Figure 10, the system includes multiple components such as external clients, load balancers, internal Docker services, and storage volumes that interact through defined and protected channels.

Clients interact with the platform through external load balancers that forward incoming requests to internal HAProxy instances operating within the Swarm cluster. HAProxy terminates secure HTTPS connections and routes traffic to the appropriate containers based on service configurations. Internal container communication occurs within dedicated virtual networks that isolate services from external access, ensuring security and integrity. Containers that expose public endpoints are explicitly linked to the HAProxy network to manage accessibility safely.

For persistent data storage, containers are connected to external storage volumes managed by the D4Science infrastructure. This design decouples data from containers, preserving user information, results, and configurations even after container updates or redeployments. Such separation is fundamental to maintaining continuity in long running scientific processes and ensures alignment with FAIR data principles.

3.3.4 Security, Authentication, and Single Sign On

All containerised services within D4Science are secured by the Identity and Access Management (IAM) system, which implements OpenID Connect, OAuth 2.0, and SAML protocols to provide unified and federated authentication. The IAM solution is based on *Keycloak*, a mature open source identity platform widely used in research and enterprise environments. Through identity brokering, users can log in with institutional credentials or social accounts while benefiting from the same security model as other D4Science services. Integration with the EGI Check-in service further extends federation across the European research ecosystem.

From a technical perspective, authentication follows the authorisation code grant flow defined by OAuth 2.0. Users who access front facing services are redirected to the IAM login page. If they are not already authenticated through another D4Science application, they enter their credentials and are then redirected back to the requested service. At that point, the service receives a JSON Web Token (JWT) containing user identity and role information. This token allows backend services to validate user permissions and enforce fine grained access control without handling credentials directly.

Service developers integrating their applications into the VRE

[7] https://clairproject.org

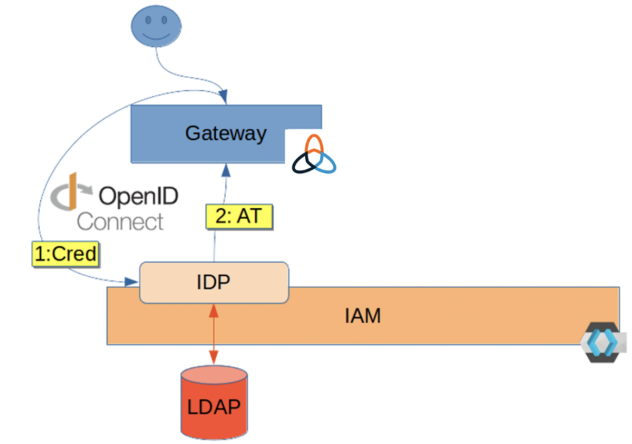

Figure 11. Logical representation of the authentication workflow for containerised services in D4Science.

must ensure compatibility with these standards. Libraries for managing OAuth 2.0 and OpenID Connect tokens are available for all major programming languages, such as node-openid-client for Node.js or Spring Security OAuth 2 for Java. Once authorised by D4Science administrators, developers can retrieve the OpenID configuration metadata from the platform's well known URI endpoint and configure their applications to validate tokens issued by the IAM.

By enforcing secure authentication, enabling controlled access to services, and providing full compatibility with federated identity providers, the D4Science IAM system strengthens trust and collaboration across institutions. Combined with containerisation, orchestration, and registry management, this ensures that every service within the Virtual Research Environment operates in a reliable, scalable, and transparent manner fully compliant with FAIR and Open Science principles.

3.4 Auditing and Accounting

Auditing and accounting are essential components of the D4Science Virtual Research Environment, enabling transparency, compliance, and responsible resource management. By systematically collecting and analysing anonymised records of user actions, requests, and resource utilisation, the platform provides administrators and service developers with valuable insights into how the infrastructure and deployed services are being used. This process supports continuous improvement, optimises system performance, and ensures that all activities comply with organisational and legal requirements.

Several technologies can be adopted to facilitate auditing and accounting in container based environments. One widely adopted approach in D4Science is the use of an *NGINX based Smart Proxy*, although other reverse proxy solutions such as HAProxy, Traefik, or Envoy can be used for similar purposes. The key requirement is to intercept and log all incoming requests that traverse the Virtual Research Environment.

As shown in Figure 12, the NGINX Smart Proxy is strategically positioned at the edge of the platform, intercepting

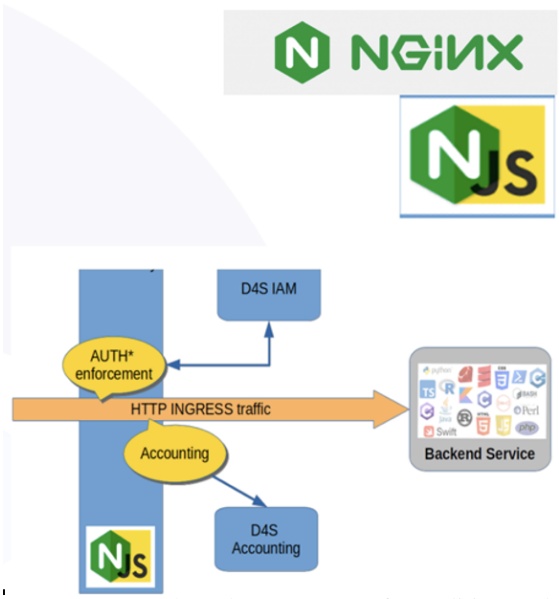

Figure 12. NGINX based Smart Proxy for auditing and accounting within the D4Science VRE.

all traffic before it reaches containerised services. When a request is received, the proxy extracts relevant metadata such as the user identifier, session token, request path, timestamp, and payload size. This information is securely forwarded to an auditing service or database, which stores and aggregates the data for compliance checks and reporting. By consolidating authentication and auditing at the proxy layer, the system guarantees that all containerised services adhere to the same security and traceability standards.

In combination with IAM tokens, the Smart Proxy validates session credentials and enforces access policies before forwarding requests to backend services. This centralised mechanism not only ensures a consistent security policy across multiple applications but also simplifies development for service providers. Custom applications deployed within D4Science do not need to implement or maintain their own authentication or authorisation logic, as these responsibilities are handled automatically by the Smart Proxy. Developers can thus focus entirely on the scientific and functional aspects of their applications, while still benefiting from full security, auditing, and accounting capabilities provided by the underlying infrastructure.

This approach enhances reliability, reduces configuration complexity, and guarantees that all user interactions are logged and monitored consistently across services. In this way, the Smart Proxy serves as a powerful intermediary between users and applications, reinforcing trust, transparency, and compliance within the D4Science ecosystem.

3.5 Monitoring and Metrics

Monitoring and metrics collection constitute a fundamental element of the D4Science infrastructure, ensuring transparency, performance, and reliability across all integrated services. Ob-

servability within the Virtual Research Environment (VRE) allows both developers and operators to understand how services behave in real conditions, anticipate failures, and optimise system utilisation. Through a combination of automated data collection, visual analytics, and intelligent alerting, the D4Science monitoring stack guarantees that every service remains responsive, reproducible, and aligned with FAIR and Open Science principles.

The core of this monitoring system is built on *Prometheus* (`https://prometheus.io`), a mature open source technology designed for multi dimensional time series data storage and retrieval. Prometheus periodically scrapes metrics from configured targets across the D4Science ecosystem, including containerised services, orchestration nodes, and network interfaces. Each metric is collected with high temporal resolution and stored as a labelled time series, which can then be queried using the Prometheus Query Language (PromQL). This model enables detailed analysis of trends over time, correlations between services, and detection of performance anomalies. Examples of collected metrics include request latency, throughput, resource consumption, and application specific indicators such as the number of running containers or data transfers performed.

Prometheus supports the definition of complex alerting rules that continuously evaluate collected metrics against predefined thresholds. When conditions are met, the system automatically triggers alerts via the integrated Alert Manager, which can dispatch notifications through email, chat systems, or external monitoring tools. This mechanism ensures that potential issues, such as service degradation, resource saturation, or unresponsive endpoints, are immediately brought to the attention of system administrators. Alerting thresholds are tuned to balance sensitivity and stability, thereby preventing both unnoticed failures and excessive notifications.

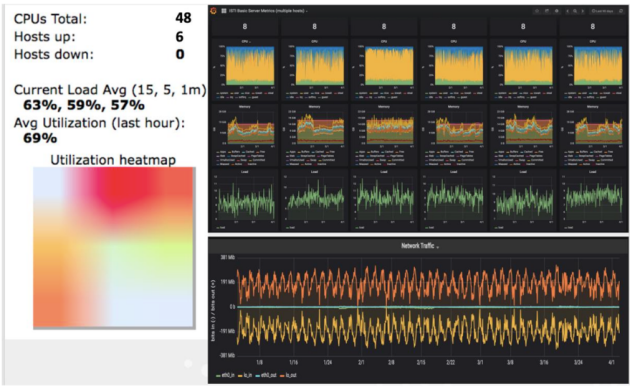

Figure 13. Prometheus metrics aggregated and visualised through Grafana dashboards for the auditing cluster.

Complementing Prometheus, D4Science employs *Grafana* (`https://grafana.com`) as its primary interface for data visualisation and dashboard management. Grafana offers a flexible and interactive web interface that allows users to visualise large volumes of metrics as intuitive charts, graphs, and tables. It enables the creation of both high level overviews and detailed diagnostic panels, providing a unified view of platform behaviour. Developers can create custom dashboards for specific services or workflows, while system administrators can maintain aggregated dashboards that summarise the health of entire clusters.

Grafana's collaboration features allow teams to share dashboards in real time, jointly analyse anomalies, and coordinate responses to incidents. The system supports advanced querying and filtering, enabling dynamic exploration of data without requiring external tools. Each dashboard can include annotations and threshold markers to identify events of interest, such as deployments or maintenance windows. Integration with the D4Science Identity and Access Management (IAM) system ensures secure authentication and fine grained access control, so that each user sees only the information relevant to their project or operational role. This tight coupling between Grafana and IAM reinforces D4Science's commitment to privacy, accountability, and transparency.

In addition to Prometheus and Grafana, D4Science employs *Uptime Kuma* (`https://uptime.d4science.org/`) as part of its continuous alerting and service availability monitoring framework. Uptime Kuma is an open source tool designed to continuously verify the availability, latency, and response quality of all critical components of the D4Science infrastructure. It operates by executing periodic health checks against registered service endpoints, simulating user access to ensure that interfaces and APIs remain responsive. Each check is associated with configurable thresholds for response time and error codes, allowing for early detection of performance degradation or downtime. The D4Science Uptime Kuma dashboard provides a real time overview of all monitored endpoints, visually distinguishing healthy and problematic services. Administrators can instantly detect disruptions, while historical data enable long term trend analysis and reliability assessment.

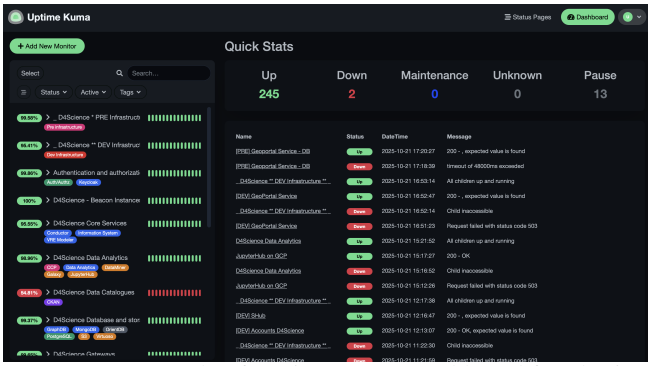

Figure 14. Example of Uptime Kuma Dashboard for alerting system (Admin view)

When anomalies are detected, Uptime Kuma automatically triggers notifications through configured communication channels such as email, or Slack. These alerts complement the Prometheus and Alert Manager pipelines, providing an ad-

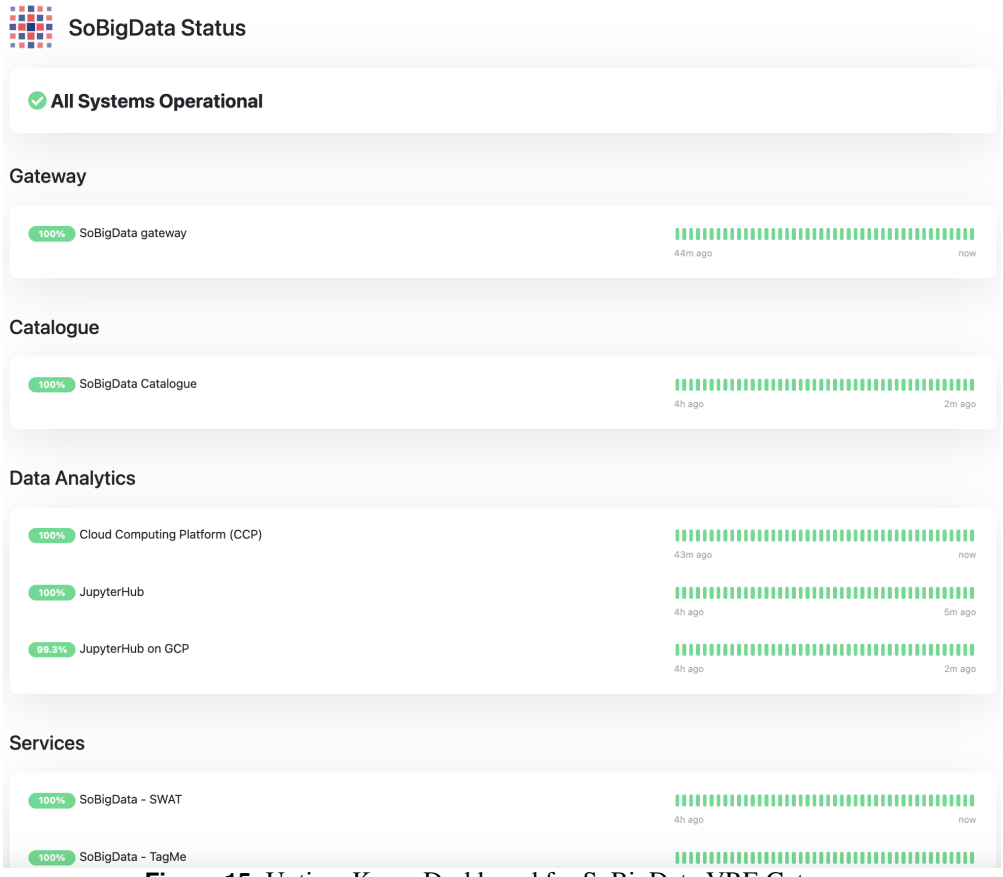

Figure 15. Uptime Kuma Dashboard for SoBigData VRE Gateway

ditional external validation layer that focuses specifically on service availability from the user perspective. This redundancy ensures that critical issues are detected both from within the infrastructure and from external monitoring points, significantly improving fault detection coverage. By correlating data from Uptime Kuma and Prometheus, operators can differentiate between infrastructure related problems and service specific degradations, facilitating more targeted and efficient interventions.

The combination of Prometheus, Grafana, and Uptime Kuma forms a cohesive and resilient monitoring and observability ecosystem within D4Science. Prometheus captures high fidelity metrics describing internal system behaviour, Grafana translates those data into actionable insights through real time visualisations, and Uptime Kuma continuously validates service reachability and user facing performance. Together, they enable predictive maintenance, performance optimisation, and long term quality assurance.

Figure 13 shows an example of an aggregated Grafana dashboard for the auditing cluster, where Prometheus metrics are displayed to provide an integrated view of performance indicators and resource utilisation. Dashboards like these empower developers, operators, and project coordinators to verify compliance with service level objectives and to sustain reproducibility and reliability across D4Science Virtual Research Environments.

Through the integration of auditing, accounting, monitoring, and alerting subsystems, D4Science guarantees that all services deployed within its infrastructure remain transparent, traceable, and accountable. These capabilities collectively ensure operational excellence, scientific reproducibility, and full alignment with FAIR principles and Open Science best practices.

3.6 Service Developer Support

The D4Science Interoperability Framework provides a comprehensive support system to assist service developers throughout all phases of development, deployment, and maintenance. Developers may encounter technical challenges, require new computational or storage resources, or need to report incidents related to their Virtual Research Environments (VREs). To streamline these processes, D4Science offers a dedicated support platform known as the *VRE Activity Tracker*, accessible at `https://support.d4science.org/`. This platform centralises the submission, monitoring, and management of all requests, providing an organised channel of communication between developers and the D4Science operations team.

The Activity Tracker is powered by the open source *Redmine* technology (`https://www.redmine.org`), a solid and widely used project management system that supports

detailed issue tracking, role based access control, and collaborative workflows. Within D4Science, this system has been adapted to align with the infrastructure's federated authentication model and to integrate directly with the Identity and Access Management (IAM) system. Each request submitted through the Activity Tracker is automatically linked to the authenticated user identity, ensuring that all actions are traceable and correctly attributed. This connection guarantees secure access while maintaining a complete and verifiable record of all support activities.

The support process is organised around tickets, allowing for systematic management of all issues and requests. Each ticket represents a specific task that can be categorised, prioritised, and assigned to the relevant expert or operational group. This structured workflow ensures that requests are addressed efficiently and transparently, while also avoiding duplication or oversight. Through the Activity Tracker, developers can monitor the status of their requests in real time, exchange messages with the support team, and review the entire history of interactions, attachments, and resolutions.

When creating a new ticket, service developers are invited to select the most appropriate request type from the available categories:

- *Incident:* to report unexpected service interruptions, software malfunctions, or infrastructure related errors that compromise normal operations.
- *Support:* to request general guidance, troubleshooting, or clarifications that are not connected to critical failures.
- *Docker Image:* to manage the deployment, registration, or update of Docker images that support custom container based services within D4Science.
- *Database:* to request the creation of database instances, the expansion of existing ones, or changes in configuration to support service requirements.

In addition to selecting a request category, developers can define the priority level of the ticket as *Low*, *Normal*, *High*, or *Urgent*. This prioritisation assists the support team in scheduling interventions and ensures that time critical issues receive prompt attention. Each ticket follows a predefined sequence of states including *New*, *In Progress*, *Feedback*, *Resolved*, and *Closed*, thus ensuring consistent monitoring from creation to completion. Automatic notifications are generated to inform both the requester and the support team of any update, guaranteeing that all parties remain aligned throughout the process.

The Activity Tracker allows users to attach configuration files, screenshots, and logs to provide all information required for diagnosis and resolution. This feature supports efficient collaboration and facilitates collective learning, since recurrent problems and their solutions can be documented and referenced by the entire community. Over time, this process builds a shared repository of technical knowledge that improves self sufficiency and accelerates problem solving. Furthermore, aggregated statistics and reports generated by the Tracker enable D4Science management to analyse trends, assess support efficiency, and identify areas for improvement within the infrastructure.

By adopting this centralised and collaborative support system, D4Science guarantees that service developers receive consistent assistance, clear communication, and transparent handling of all technical matters. The VRE Activity Tracker functions not only as a help desk but also as a knowledge platform that strengthens coordination, supports the continuous evolution of services, and contributes to the overall sustainability of the D4Science ecosystem.

3.7 REST APIs

The D4Science ecosystem exposes a comprehensive suite of REST APIs available at `https://dev.d4science.org`, designed to ensure seamless integration, interoperability, and efficient development of services within its Virtual Research Environments (VREs). These APIs empower developers to interact programmatically with core D4Science components, enabling the creation of new tools, data workflows, and analytical services that are fully interoperable with the existing infrastructure. By following standard REST design patterns, each API supports structured requests and responses encoded in JSON, ensuring transparency, reproducibility, and long term maintainability.

The D4Science APIs cover all functional domains of the infrastructure, including identity management, file synchronisation and sharing, metadata cataloguing, analytics execution, and geospatial data access. All endpoints use secure HTTPS communication and OAuth2 compliant authentication tokens obtained through the Identity and Access Management (IAM) system. This guarantees that only authorised users and registered applications can access or modify resources within the infrastructure. Detailed API documentation and examples are available through the developer portal at `https://dev.d4science.org`. Practical examples of API usage, including authenticated token retrieval and StorageHub queries, are provided in such API documentation.

Main functional domains of the D4Science REST APIs:

- *Identity and Access Management (IAM):* provides authentication and authorisation endpoints supporting token exchange, session validation, and user role inspection through standard protocols such as OAuth2 and OpenID Connect.
- *File Sync and Share (StorageHub):* enables users and applications to manage files, datasets, and folders stored in their collaborative workspace. Typical operations include uploading, renaming, sharing, and retrieving files using endpoints such as `/storagehub/items`.

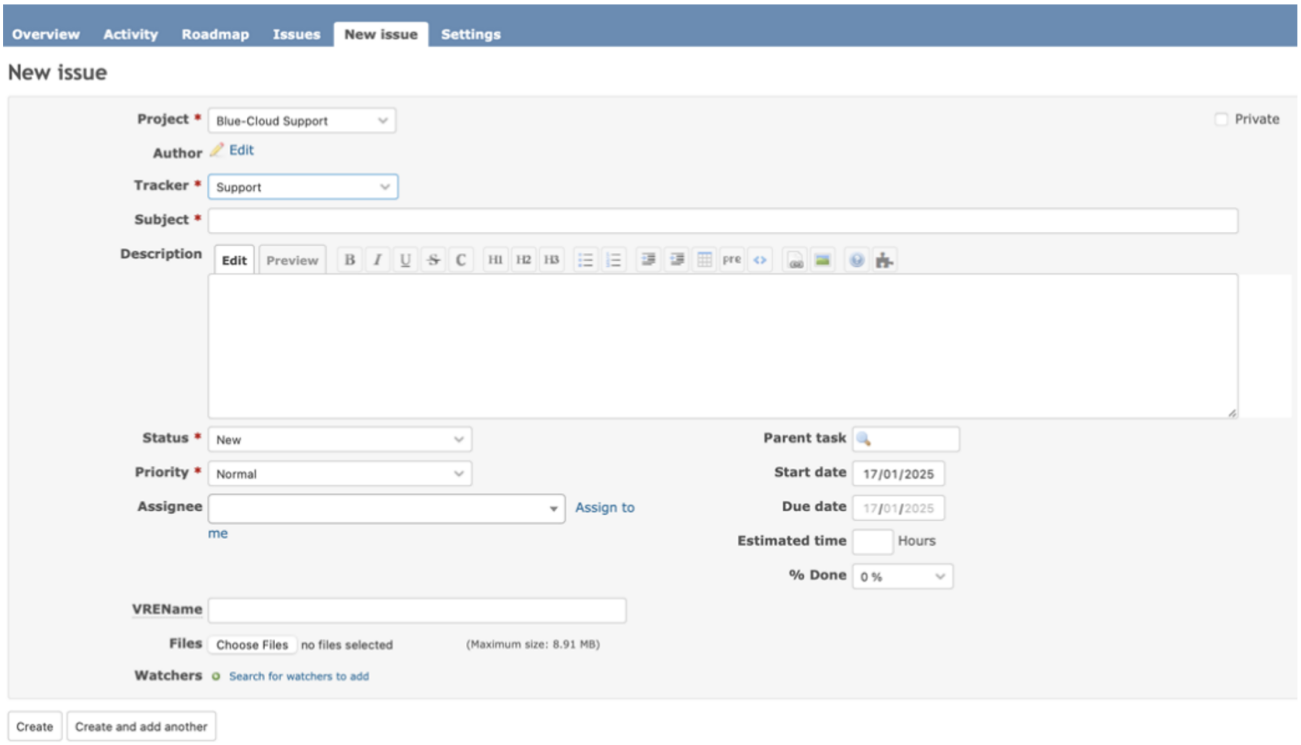

Figure 16. The Activity Tracker input form for creating a new service request.

- *Catalogue API:* offers programmatic registration, search, and update of metadata records associated with datasets, services, or workflows. It is aligned with FAIR principles and supports persistent identifiers, schema validation, and linked metadata models.

- *Analytics Engine API:* allows developers to submit and monitor computational jobs executed within the Analytics Engine environment. Methods, executions, and results can be programmatically managed through endpoints such as `/analytics/methods`

- *Geospatial API:* supports the publication, discovery, and access of spatial datasets and services. It includes endpoints for layer management, geographic metadata retrieval, and spatial search queries.

- *Social and Profile API:* provides access to user information, roles, and membership management, as well as social collaboration functions for posting messages, commenting, and content contribution within the VRE community.

Each endpoint follows consistent conventions using standard HTTP methods (`GET`, `POST`, `PUT`, `DELETE`) and communicates structured JSON responses. Returned metadata include identifiers, creation timestamps, and links to related resources, ensuring that each operation remains transparent and traceable. In case of errors, standard HTTP status codes are returned together with diagnostic messages describing the reason of failure, enabling straightforward debugging and robust client implementations.

The D4Science REST APIs are natively integrated with the auditing and monitoring subsystems of the platform. All API calls are securely logged and associated with the user's identity through the IAM layer, ensuring accountability and traceability of every action. Metrics such as usage frequency, latency, and service load are continuously monitored to guarantee performance and compliance with FAIR and Open Science guidelines.

By combining a consistent interface design, strong authentication, and comprehensive documentation, the D4Science APIs provide a reliable foundation for interoperability and cross-project integration. They enable research communities and developers to extend the capabilities of the infrastructure, build new applications, and automate workflows while maintaining security, transparency, and reproducibility across all operations. This approach reinforces the mission of D4Science to foster open, collaborative, and data-intensive science by offering programmable access to every component of its ecosystem.

4. Conclusions and Future Work

The D4Science Interoperability Framework described in this report represents a mature, production ready environment for data-driven research that supports the seamless integration of data, tools, and computational services within collaborative research settings. Its architecture demonstrates how a unified ecosystem can facilitate the principles of Open Science by en-

abling secure access, reproducibility, and transparency across distributed infrastructures. Through its core components, the Identity and Access Management system, the VRE Gateway, the Virtual Labs, and the Analytics Engine, D4Science provides researchers and developers with a coherent environment in which to design, deploy, and execute scientific workflows efficiently.

The framework's modular design allows for continuous evolution. Each service, from the File Sync and Share platform to the REST APIs, contributes to a federated infrastructure that scales with the needs of the research communities it serves. By promoting interoperability and standard compliance, the framework ensures that data and services can be reused and extended beyond individual projects, maximising the return on investment of public Research Infrastructures (RIs). Its alignment with FAIR principles further guarantees that every digital object, computation, and dataset remains findable, accessible, interoperable, and reusable throughout its life cycle.

Future work will focus on extending the interoperability layer to support a wider range of external platforms and data repositories, including advanced connections with high performance computing resources and cloud orchestration services . Particular attention will be dedicated to enhancing provenance management, possibly cross RIs, improving scalability, and strengthening automation in the continuous integration and deployment processes. Another priority is to enrich the developer experience by expanding the REST API catalogue, providing new software development kits, and delivering updated guidance documentation and training materials.

At the community level, the next phase will consolidate collaboration with European and international initiatives such as EOSC, and other domain specific infrastructures. The aim is to foster an ecosystem of interoperable digital services that transcends disciplinary boundaries and empowers researchers to work collaboratively across institutions and scientific domains. By continuously improving usability, reliability, and openness, D4Science reaffirms its role as a digital infrastructure supporting data driven science, transparent knowledge production, and collective innovation.

Acknowledgements

The authors whish to thank the continuous support of the D4Science team and all members of the collaborating research communities for its active engagement, continuous innovation, and commitment to the principles of open and reproducible science. Special thanks are extended to the technologists and system administrators who maintain the operational services, ensuring reliability and availability for all users.

The work described in this document has benefited from discussions and technical exchanges within projects adopting the D4Science infrastructure, including initiatives such as BlueCloud 2026, IRISCC, SoBigData RI, FAIR-EASE, and other European and international collaborations. Their feedback and real-world use cases have been essential in shaping the evolution of the platform and the development of this guidance documentation.

Funding

This work is partially supported by the projects: "FOSSR – Fostering Open Science in Social Science Research" (IR0000008), funded by the European Union, NextGenerationEU; "SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" (IR0000013), funded by the European Union, NextGenerationEU.

Competing interests

The authors have no competing interests to declare that are relevant to the content of this technical report.

References

[1] Massimiliano Assante, Leonardo Candela, Donatella Castelli, Roberto Cirillo, Gianpaolo Coro, Andrea Dell'Amico, Luca Frosini, Lucio Lelii, Marco Lettere, Francesco Mangiacrapa, Pasquale Pagano, Giancarlo Panichi, Tommaso Piccioli, and Fabio Sinibaldi. Virtual research environments co-creation: The D4Science experience. *Concurrency and Computation: Practice and Experience*, 35(18):e6925, 2023.

[2] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1):160018, March 2016.

[3] M Assante, A Boizet, L Candela, D Castelli, R Cirillo, G Coro, E Fernández, M Filter, L Frosini, G Kakaletris, et al. Realising a science gateway for the agri-food: the aginfra plus experience. 2021.

[4] Dick Schaap, Massimiliano Assante, Pasquale Pagano, and Leonardo Candela. Blue-cloud: Exploring and demonstrating the potential of open science for ocean sustainability. In *2022 IEEE International Workshop on Metrology for the Sea; Learning to Measure Sea Health Parameters (MetroSea)*, pages 198–202. IEEE, 2022.

[5] Gayane Aghakhanyan, Andrea Barucci, Maria Antonietta Pascali, Massimiliano Assante, Giulio Bagnacci, Elena Bertelli, Francesca Pia Caputo, Maria Emanuela Cuibari, Emanuele Carlini, Roberto Carpi, et al. Navigator: A regional multimodal imaging biobank initiative powered by ai tools for precision medicine in oncology. *European Journal of Radiology*, page 112327, 2025.

[6] Fosca Giannotti, Roberto Trasarti, Kalina Bontcheva, and Valerio Grossi. Sobigdata: Social mining & big data ecosystem. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, pages 437–438, 2018.

[7] Massimiliano Assante, Marco Lettere, Alfredo Oliviero, and Pasquale Pagano. Empowering collaborative and reproducible large-scale data analytics with d4science. *ERCIM News*, 2025(140), 2025.