Project Title	Blue-Cloud 2026: A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters
Project Acronym	Blue-Cloud 2026
Project Number	101094227
Type of project	RIA – Research and Innovation Action
Topics	HORIZON-INFRA-2022-EOSC-01
Starting date of Project	01 January 2023
Duration of the project	42 months
Website	www.blue-cloud.org

D5.1 – Blue-Cloud VRE Common Services 1st Release

Work Package	WP5 | Blue-Cloud VRE platform evolution and integration with EOSC resources and services
Task	N.A.
Lead author	Massimiliano Assante (CNR-ISTI)
Contributors	Leonardo Candela (CNR-ISTI), Roberto Cirillo (CNR-ISTI), Andrea Dell'Amico (CNR-ISTI), Enol Fernandez (EGI), Luca Frosini (CNR-ISTI), Lucio Lelii (CNR-ISTI), Marco Lettere (Nubisware), Francesco Mangiacrapa (CNR-ISTI), Pasquale Pagano (CNR-ISTI), Giancarlo Panichi (CNR-ISTI), Tommaso Piccioli (CNR-ISTI)
Peer reviewers	Patricia Cabrera (VLIZ), Sara Pittonet (Trust-IT)
Version	V1.0
Due Date	31/12/2023
Submission Date	30/12/2023

Dissemination Level

X	PU: Public
	CO: Confidential, only for members of the consortium (including the Commission)
	EU-RES. Classified Information: RESTREINT UE (Commission Decision 2005/444/EC)
	EU-CON. Classified Information: CONFIDENTIEL UE (Commission Decision 2005/444/EC)
	EU-SEC. Classified Information: SECRET UE (Commission Decision 2005/444/EC)

Funded by the European Union

Version History

Revision	Date	Editors	Comments
0.1	12/11/2023	Massimiliano Assante (CNR-ISTI)	Table of Content
0.2	17/11/2023	Massimiliano Assante (CNR-ISTI)	Added Introduction section
0.3	24/11/2023	Massimiliano Assante (CNR-ISTI)	Added VRE Logical arch. Section
0.4	27/11/2023	All authors	Added Analytics Computing Framework Section
0.5	05/12/2023	All authors	Added Enabling Framework Section Added Storage Framework Section
0.6	15/12/2023	All authors	Added Catalogue Framework Section
0.7	18/12/2023	Massimiliano Assante (CNR-ISTI)	Added summary, revised metadata and table of contents
0.8	22/12/2023	Patricia Cabrera (VLIZ)	Internal Review
0.9	28/12/2023	Massimiliano Assante (CNR-ISTI)	Addressed comments from internal review
1.0	29/12/2023	Sara Pittonet (Trust-IT), Pasquale Pagano (CNR-ISTI)	Final version

Glossary of terms

Item	Description
API	Application Programming Interface
CMS	Content Management System
CVS	Code Versioning System
DD&AS	Data Discovery & Access Service
DNS	Domain Name System
D4Science	Data Infrastructure promoting Open Science (managed by CNR)
EOSC	European Open Science Cloud
EOVs	Essential Ocean Variables
GUI	Graphical User Interface
JSON	Javascript Object Notation
IDE	Integrated Development Environment
OGC	Open Geospatial Consortium
OIDC	OpenID Connect
RIA	Rich Internet Application
UI	User Interface
UMA	User-Managed Access
VLab	Virtual Laboratory
VRE	Virtual Research Environment

Keywords

EOSC; Virtual Labs; Big Data; Virtual Research Environment; e-infrastructures

Disclaimer

The Blue-Cloud 2026 project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101094227. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

EXECUTIVE SUMMARY

This deliverable documents the design principles and software architecture characterising the release and development of the Blue-Cloud Virtual Research Environment (VRE) common services, namely the analytics computing framework, the catalogue framework, the storage framework and the enabling framework components. This report is the first of two versions, each one describing the design associated with a specific version of the VRE. This deliverable focuses on the design principles and software architecture included in the first release of ththis one as released at M12 (December 2023), while a second release is due at the end of the third year of the project and will be reported in D5.4 Blue-Cloud VRE Common Services 2nd Release (M36), due in December 2025.

The deliverable consists of six sections.

- Section 1 briefly introduces the role of this deliverable in the development and delivery of the Blue-Cloud VRE common services.
- Section 2 describes the Blue-Cloud VRE logical architecture of the common services and how they relate to the other services available in the VRE.
- Section 3, 4, 5 and 6 document the first release of the Blue-Cloud VRE common services available at M12, reporting the design principles and reference software architecture of the released solutions. Specifically, Section 3 describes the analytics computing framework which includes the Analytics Engine, the RStudio and the Jupyter Notebooks via JupyterHub. Section 4 presents the VRE Catalogue framework and its components, and section 5 reports on the Storage framework.
- Finally, section 6 concludes the report by illustrating the services composing the Enabling framework, which is used as a common ground for all the above-mentioned frameworks.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. Introduction

The Horizon Europe Blue-Cloud initiative started its journey in 2019 with the aim to create a European Open Science Cloud for marine data by federating data and e-infrastructures and making data products and technologies available as open science resources for the broad marine research community. Since 2023, the Blue-Cloud 2026 follow-up project aims at a further evolution of this pilot ecosystem into a Federated European Ecosystem to deliver FAIR & Open data and analytical services, instrumental for deepening research of oceans, EU seas, coastal & inland waters. Building upon the pilot Blue-Cloud project, the current Blue-Cloud technical framework is extensible and open by design, constantly evolving according to the needs of the community, facilitating collaborative research and the uptake of Open Science principles, through a distinguished set of services. . Among those, the following are considered Blue-Cloud core services:

- The **Data Discovery and Access Service** (DD&AS) is an overarching service to facilitate federated queries and retrieval, and smart sharing of multi-disciplinary datasets with human and machine users.
- The Blue-Cloud **Virtual Research Environment (VRE)** orchestrates the computing and analytical services in specific integrated and managed applications exploiting federated Blue-Cloud data resources as well as external data resources**.**
- Five **dedicated Virtual Laboratories** (VLabs) co-created with top-level marine researchers to showcase the power of the Blue-Cloud Open Science platform via real life scientific cases [5].

The Blue-Cloud VRE Common Services described in the following chapters encompass a wide range of functionalities, facilitating collaboration, data analytics, result publication, and seamless integration with external systems, and therefore play a pivotal role in promoting Open Science practices within VLabs, enabling researchers to leverage the benefits of advanced e-infrastructures. . They also provide interfacing with the Data Discovery & Access Service (DD&AS) and the interfacing with external services such as the WEkEO DIAS of Copernicus Marine, enabling researchers to leverage external resources and data sources, expanding the capabilities and richness of their research endeavours. These functionalities contribute to the comprehensive and interconnected nature of the Blue-Cloud ecosystem.

It is important to note that these VRE services are built on the D4Science [2, 4] infrastructure and leverage the gCube [1] open-source technology. This infrastructure provides a reliable and scalable foundation for the VRE services, ensuring their stability and performance. The exploitation of the gCube open-source technology further enhances the flexibility and extensibility of the services, allowing for continuous innovation and improvement.

The VRE services are deployed in the Blue Cloud VRE Gateway, accessible to any researcher upon authentication at https://blue-cloud.d4science.org. This gateway serves as a centralised hub, providing researchers with easy and convenient access to the services and VLabs.

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

2. VRE logical architecture and common services

Figure 1 illustrates the Blue-Cloud VRE Common Services logical architecture, with the Blue-Cloud Virtual Laboratories that can be built on top of it. The VRE is built by exploiting the Virtual Machines operated and provisioned by D4Science, together with services for their management and administration (VRE federated resources). Above this layer, a number of services form the enabling framework, essential for supporting the operation of all services and VREs. Specifically, this framework includes:

1. **Information System service**: This service allows for the dynamical (de)registration and discovery of all VRE resources (data sources, services, computational nodes, etc.), by users and other services;
2. **Identity and Access Management (IAM) service:** Responsible for handling authentication and authorization services, as well as auditing services. It is capable of granting and tracking access and usage actions from users; and
3. **VRE Management System service:** This service facilitates the deployment of specialised VLabs based on a selected subset of applications.

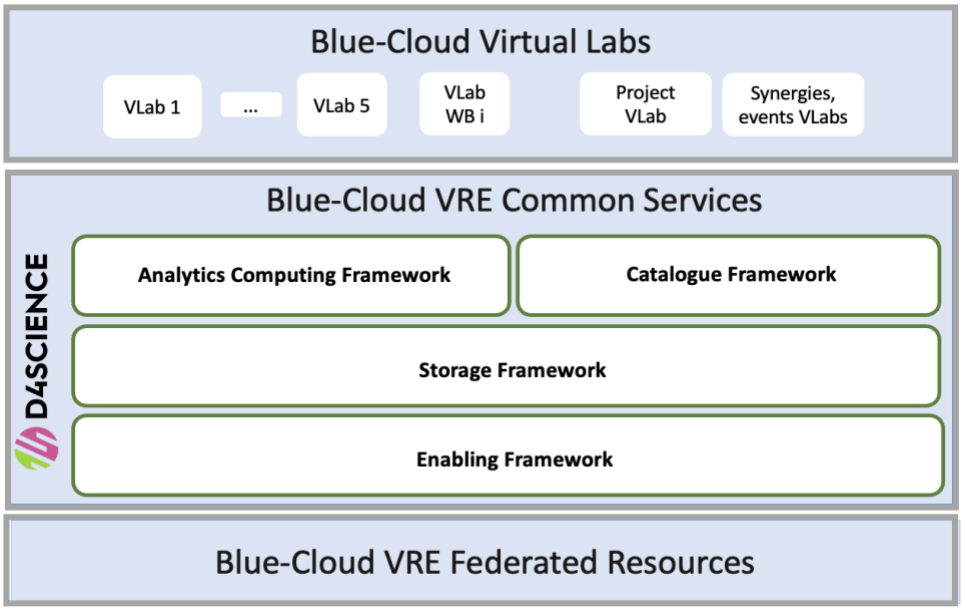


Figure 1 The Blue-Cloud VRE common services logical architecture

On top of this layer, three frameworks, namely the Storage Framework, the Catalogue Framework and the Analytics Computing Framework are built to provide common facilities to users.

The **Analytics Computing Framework**, to date, offers set of data analytical services that can be requested and customised in the VLabs:

- **Analytics engine:** This service features the following two engines.

    - **DataMiner**: the engine that was used in Blue-Cloud and will be maintained in Blue-Cloud 2026. It comprises a rich array of ready to use analytical methods ranging from data clustering methods to geospatial data analytics, occurrence data management, and species distribution maps generation.
    - **Cloud Computing Platform (CCP)**: a newer engine sharing the same design principles and main features as the DataMiner. CCP is being evolved in Blue-Cloud 2026, to enhance its functionality by leveraging on the most recent advancements in IT and software engineering.
- **RStudio:** This service allows users to perform online statistical analyses. It offers a predefined list of R packages, and each VLab can define its RStudio server configurations. The configurations include a Standard (4 cores/8GB RAM) and a Large (8 cores/32GB RAM) one.
- **JupyterLab**: a web-based interactive development environment designed for Jupyter notebooks, code, and data. It allows users to configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning.

The **Catalogue Framework** primary component is the VRE Data Catalogue. This service is built on open-source technology for data catalogues, CKAN (ckan.org). Through this Catalogue users can search and browse data, products, and resources (posters, deliverables, etc) of interest from the Blue-Cloud community.

The **Storage Framework** operates transparently to users and virtualises the available storage technologies and their actual physical hardware.

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

3. Analytics Computing Framework

The Analytics Computing Framework includes a set of services and components for performing data processing and mining on information sets. Section 3.1 is about the two available engines for importing and execution of analytical methods. The ***Cloud Computing Platform*** **(CCP)** is the newer engine that will evolve in Blue-Cloud 2026, and the ***DataMiner***, the engine that was used in the previous Blue-Cloud project and it will be maintained but not evolved in Blue-Cloud 2026.

Section 3.2 is about **JupyterHub,** used as the enabling platform for the **RStudio** application, which allows users to perform online statistical analyses and the **JupyterLab** application**,** the web-based interactive development environment for Jupyter notebooks. Section 3.3 is about the support offered by the VRE for the deployment of **ShinyApps**, via ShinyProxy and any other **Containerised (Docker[1]) applications**.

3.1. Analytics Engines

3.1.1. Cloud Computing Platform

Over the last decade, several fields of Information and Communication Technology (ICT) have experienced a major evolution and numerous advancements. One of the most noteworthy is the widespread adoption of Microservice development patterns. This resulted in substantial improvements in terms of interoperability and composability of software components even when these components are distributed across a network.

REST APIs have become a fundamental technology to build Web Services, while Web components and Microfrontends aim to extend the advantages of service orientation directly into the Graphical User Interfaces (GUIs). The emergence of new programming languages with smaller footprint, improved domain specificity and simplified syntax, and their adoption has been boosted thanks to the increased freedom that aforementioned improvements allow.

As a result, the adoption of microservice and cloud-native approaches has led to the emergence of new requirements and novel patterns have emerged at the operational level as well. The so-called concept of “containerisation”, serving as the lightweight alternative to virtualisation of computing resources, has become the de facto standard for packaging and provisioning software. Tools and languages have been created for the automation of provisioning, microservice orchestration, inspection and monitoring of software and infrastructure components.

CCP Design principles and implementation

The vast landscape of new opportunities described above, in addition to the greatly increased requirements and expectations, represented the major drivers for the design and development of the

[1] https://www.docker.com/

CCP. This platform represents the result of a comprehensive re-evaluation of the existing DataMiner solution. After a critical overview of the current solution DataMiner and its Software Importer (SAI) (cf. Section 3.2), a use case driven analysis of the requirements for CCP has been performed.

The following use cases have been identified:

- **Use case 0 - Workspace abstractions**: CCP should be able to access file resources stored on different distributed workspace abstractions.
- **Use case 1 - Access Data as a Service**: CCP should be able to access data exposed as services.
- **Use case 2 - Low-level polyglot**: CCP should natively support as many programming and scripting languages as possible.
- **Use case 3 - High-level polyglot**: CCP should support different execution environments for scripts and applications.
- **Use case 4 - Support complex tools**: CCP should support complex tools that could have been used in pre-existing workflows.
- **Use case 5 - Support coarse grained parallelism**: CCP should support the possibility of subdividing scripts into parallel tasks.
- **Use case 6 - Support fine grained parallelism**: CCP should support the exploitation of parallel execution environments.
- **Use case 7 - Link to structured code repositories**: CCP should enable access to code stored on different code and artefact repositories.
- **Use case 8 - Develop with non-headless workflows**: CCP should support environments with their own visual or interactive frontend.
- **Use case 9 - Discovery, consolidation and community driven growth**: It should be possible to discover consolidated data sources, methods and algorithms, both interactively and programmatically.

CCP is a sophisticated system of distributed components encompassing different technological and operational domains. This system includes a plethora of resources (hardware, virtual machines, databases and datastores) and processes such as orchestration logic, application and integration software developed on purpose or made available by the open-source community. At the bottom of these stacks lie the physical resources which can be anything from full-blown datacentres to personal laptops. Each physical host can provide its own set of **services** including storage facilities, code repositories, databases and more. A special physical host is represented by the D4Science platform where the CCP core runs together with other core essential services. Most importantly, a physical host can provide a **computing infrastructure** which represents the set of computing resources used to run **methods** in **executions** performed on compatible **runtimes**.

A method is a lightweight data structure that describes an algorithm, procedure or function. This descriptor provides information about input parameters, expected output results, authorship and versioning. Most importantly, it defines the compatibility with one runtime and the affinity to one or more computing infrastructures. Method descriptors are all kept inside a dedicated **method repository** hosted by the D4Science datacentre.

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

In order to be executable, a method has to be associated with a runtime which can be seen as a (Docker) container. This container is able to provide everything needed by the method including its operating system and application dependencies. Runtimes are complex and "large" artefacts that are kept in specialised **runtime repositories**. One such repository is hosted by the D4Science datacentre but others can be provided among the services of a datacentre. Additionally, access to public runtime repositories offered by third parties is also feasible.
The **execution** of a method is a complex process that involves the following macro phases:

- delivering a method descriptor and a compatible runtime to one of the affine infrastructures;
- bootstrapping the runtime and deploying the method;
- running the method by conveying the user provided inputs;
- monitoring the execution by forwarding real-time logs to the requesting user;
- uploading the generated; and
- cleaning up by destroying the dedicated runtime.

All the information recorded during the execution of a method is stored in a dedicated **execution repository**. The execution repository serves as a temporary storage that helps users track their executions. Upon completion, an execution can be archived, restored or resubmitted.
These operations are coordinated by an internal piping and wiring network. In order to participate as a computing infrastructure, a datacentre has to provide a **controller** component that represents the infrastructure side termination of the wiring and piping. Users can interact with the CCP and its features in a way that hides away most of this complexity. This can be done either programmatically or by accessing simple widgets embedded in web pages or portals or more complex applications. This architectural vision translates to the logical architecture depicted in Figure 2.

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

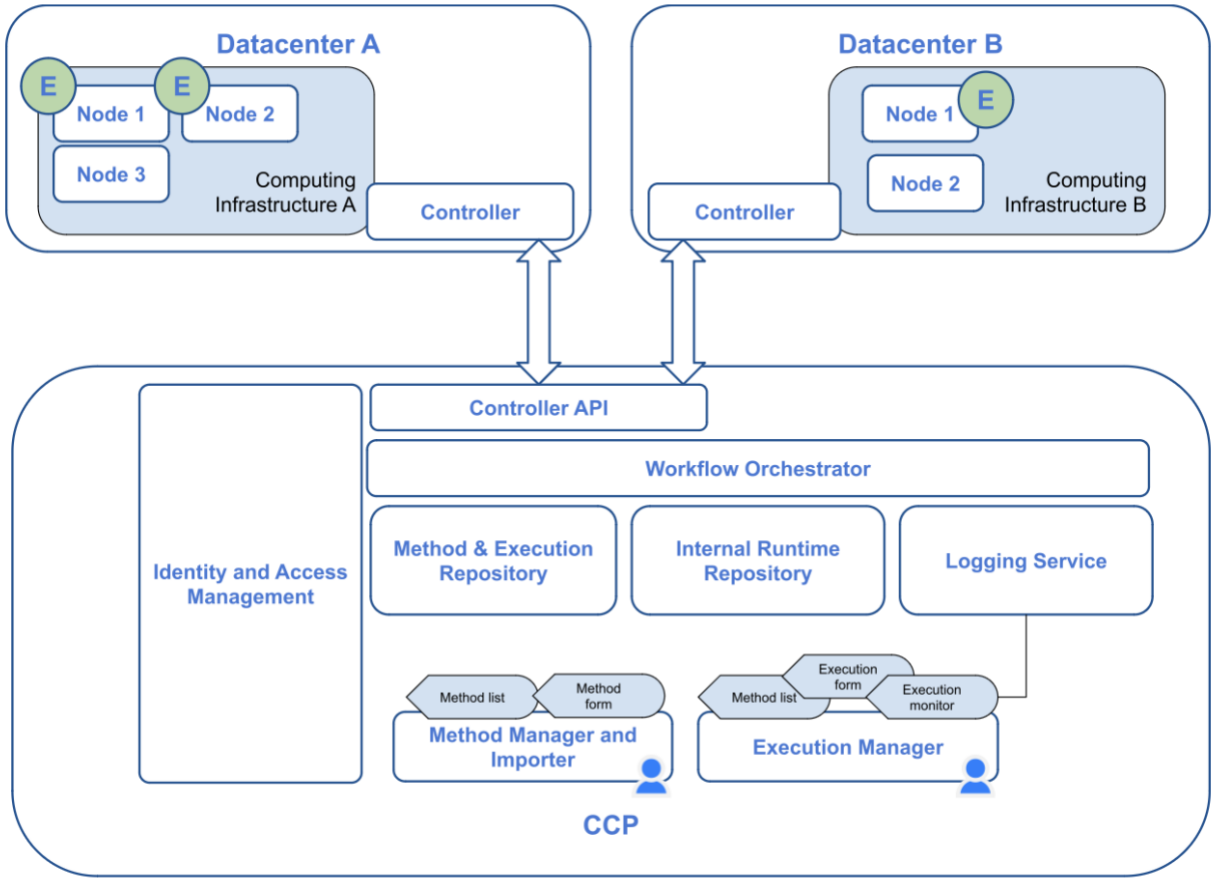


Figure 2 Logical architecture design

The logical architecture in Figure 2 illustrates the natively distributed nature of CCP. Starting from the top, **infrastructures** are computing resources that will run CCP executions (shown as green Es). These infrastructures can be hosted in datacentres. The term datacentre actually encompasses everything ranging from simple laptops up to clusters of **nodes**. *Nodes* refer to machines such as servers or PCs delegated to run the actual execution of methods.

A computing infrastructure can be connected to the CCP core by a **Controller** component. Controllers are processes that communicate via a dedicated API with the CCP in order to poll for *tasks* to perform on the **Infrastructure** they control. Tasks may include deploying new runtimes, provisioning and execution of methods, reporting on the overall status of the Infrastructure. In order to keep the current state for CCP, a repository of methods and executions is implemented. For methods, the descriptor data structures are kept in the proper repository.

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

For each execution, the repository stores a lot of information that allows for an exact reproduction of the execution:

- the method descriptor;
- the descriptor of the infrastructure that executed the method;
- the execution request with all input parameters;
- the standard output and error channels;
- all the data that is produced and uploaded by the method; and
- metadata identifying the user that requested the execution.

A repository for runtimes is provided in order to provide trusted and validated runtimes, with the flexibility to utilise external repositories or those provided by the datacentre hosting infrastructures. All the components adhere to well-defined and where possible, standard-compliant REST APIs. This allows users to programmatically interact with CCP without having to know the complex architecture.

A set of widgets (as shown in Figure 4), such as “Method List”, “Method Form”, “Execution Form”, and “Execution Monitor” are delivered in the form of *web components*. These components can be embedded into web pages, portals or other web based complex applications such as Jupyterhub. Two core applications are provided in D4Science portals involved in Blue-Cloud by combining the available widgets: the “CCP Method Manager and Importer” and “CCP Method Execution Manager”. Given that many of the operations involved are lengthy and asynchronous, CCP includes a *Logging Service* that is used to send back real-time notifications to user components about the state of a particular process. These notifications include advancement of Executions, advertisement of Infrastructures status updates, and error conditions.

All complex processes involved in CCP are implemented as workflows inside a *Workflow Orchestrator.* This Orchestrator not only provides a high level of flexibility and customisation, but also allows for a centralised endpoint to monitor progress and check for errors that may occur. At the foundation of all interactions among external actors, such as users and Controllers, a strong authentication and authorisation mechanism is enforced by *Identity and Access Management* (cf. Section 6.1). This mechanism not only enables security requirements, rather it also implements ownership attribution and auditing.

Figure 3 depicts a detailed technical view of the CCP instance that is currently available in the Blue-Cloud VRE.

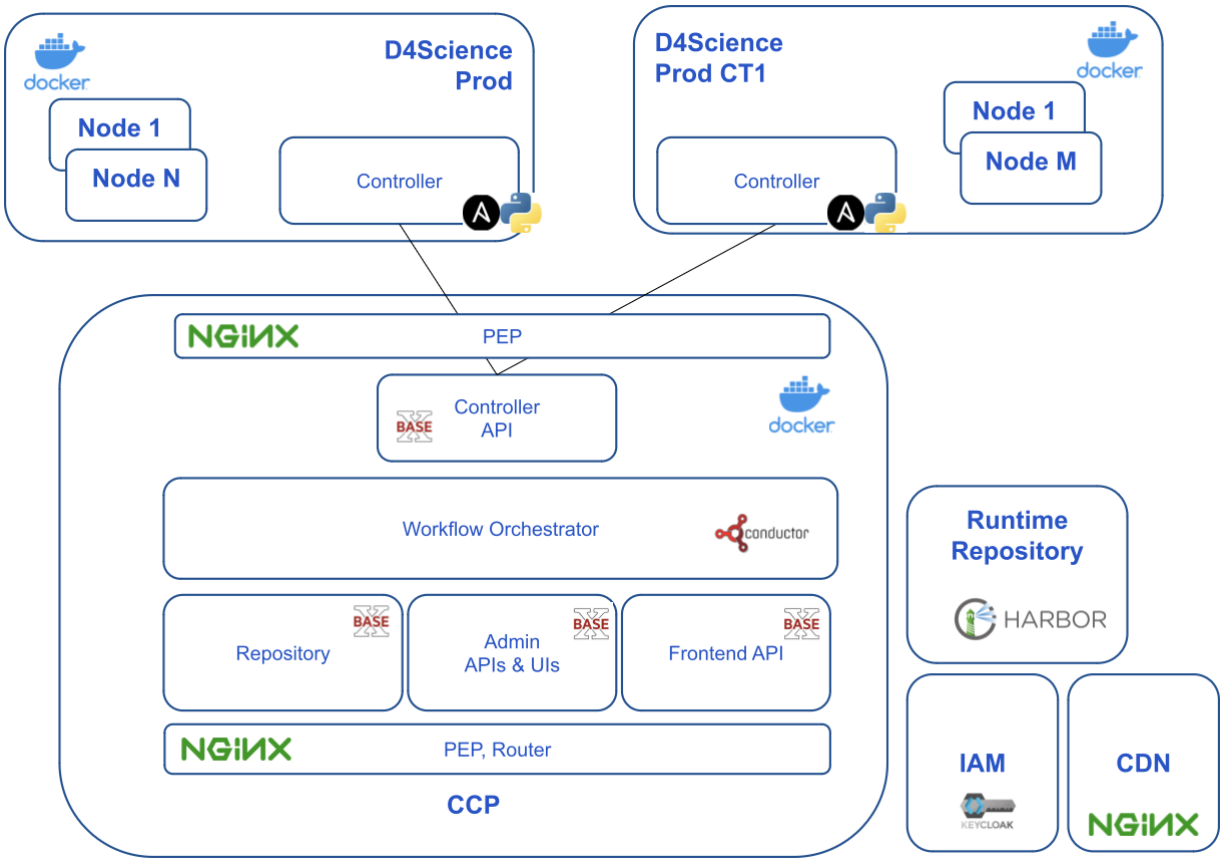


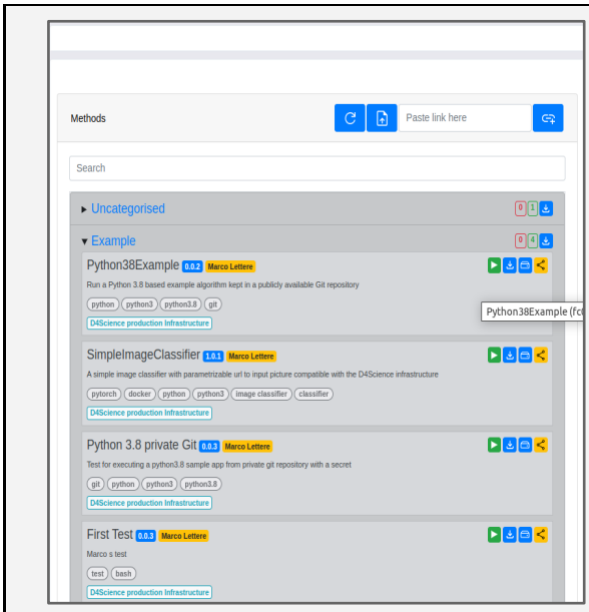
Figure 3 Detailed CCP architecture in Blue-Cloud VRE

CCP is packaged as a Docker stack that is deployed in the D4Science data centre at the CNR premises in Pisa. Access to the internal components either through APIs or Widgets is controlled by a Policy Enforcement Point (PEP) that authenticates and authorises every operation. Security enforcement is obtained in cooperation with the D4Science Identity and Access Management (IAM) service built with Keycloak (cf. Section 6.1).
Harbor is used to implement the runtime repository, or container registry. Currently Docker is supported in production as the type of runtime. Controllers are deployed outside the CCP stack "in front of" the Computing infrastructure they control. Communication towards CCP is performed via authenticated HTTPs calls, in order to grant access from any point of the Web. In the current deployment there are two infrastructures available one in a datacentre in Pisa (CNR) the second one in Catania (GARR-CT). Two main applications have been provided for Blue-Cloud users (Method importer and Method runner) as a combination of the developed UI Components.

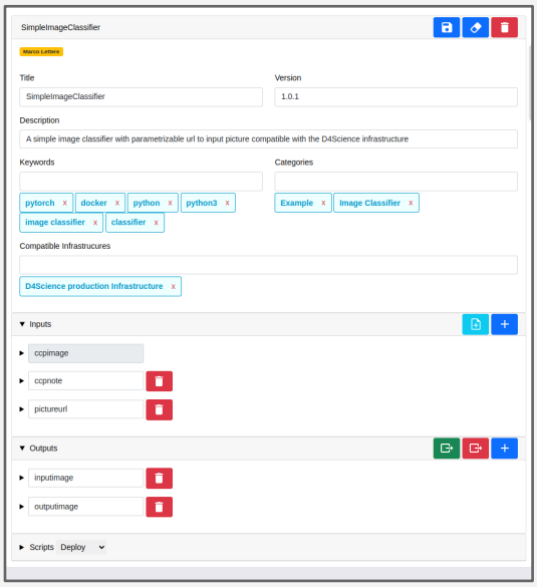
Figure 4 shows the Web User Interfaces of the CCP available in the Blue-Cloud VRE. Specifically:

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

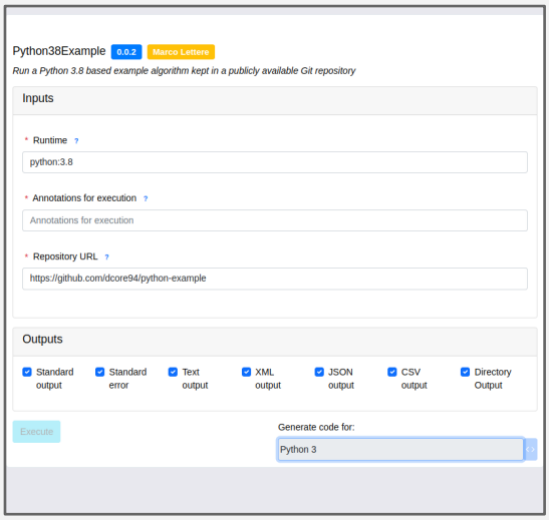
- Method List example: to list of the available methods. Searchable by name, description and keywords. Item is draggable to an Execution Form;
- A Method Form example: to specify a Method according to the OGC Process specification data model. Provide authorship and version information, naming, keywords, inputs, outputs and provisioning information;
- Execution Form example: to request the execution of a method by providing inputs. Allow for code generation in order to request an execution programmatically in languages such as Python, JPython for Jupyter, R, Julia; and
- Execution Monitor example: for the execution tracking with real time streaming of standard output and error logs produced by the method execution. Extra functionality like execution archiving, restoring or replay, code generation for programmatic resubmission of executions in common languages such as Python, JPython for Jupyter, R, Julia.



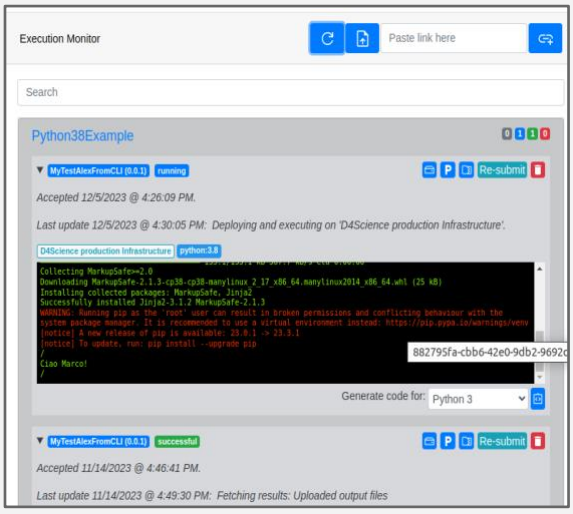
Methods List



Method Form



Execution Form



Execution Monitor

Figure 4 Some examples of CCP UI Components

At the time of writing, many features that are now available in the current version of CCP have been implemented after getting valuable suggestions from the Project WP3 and WP4 members. For instance, the feature of having standard output and error channels of the method execution reflected to the Execution Monitor (See Figure 4 bottom right).

Additionally, during the reporting period *(a)* the encoding of execution scripts has been made safer, the *(b)* possibility to use other runtime repositories has been finalised, and *(c)* the widget realising the method execution form has been improved by introducing the possibility to define a cardinality for input parameters and the possibility to upload small files as input values. Finally, the following enhancements have been added:

- Method lifecycle tracking, in order to keep a history of the important events related to a Method (Creation, sharing, updating);
- Clone functionality for a Method has been enhanced with the "derive facet" which provides the possibility to preserve a link to the cloned source Method;
- New mechanism of notifications to inform Infrastructure Managers of the creation of a new Method and VRE/VLab Managers when a Method is shared in their context.

3.1.2. Data Miner

The DataMiner (DM) [7] engine includes a set of services and components for performing data processing and mining on information sets. Analytical methods employed by users can be developed in almost any programming language (R, Java, Python, Fortran, Octave, etc.) and then imported via the Software and Algorithms Importer (SAI) and executed via the DataMiner engine. Once imported through SAI, DataMiner offers these scripts as a Service, handling aspects like multi-tenancy and simultaneous usage. This system also simplifies script updates for scientists, eliminating the need for extensive re-deployment processes. Essentially, SAI facilitates the creation of processes that operate on the Blue-Cloud VRE Cloud computing platform, accessible through the WPS standard.

To import a script, one must follow three steps in SAI:

1. Define the input, output, and types for the primary script that coordinates the process.
2. Construct the Software: This step involves packaging the script and readying it for use on the computing platform. It's necessary whenever changes are made to the script's interface (inputs/outputs) or its dependencies.
3. Deploy the Software: This final step allows the script to run on the computing platform within the virtual laboratory where it's been uploaded.

Moreover, there's a Repackage function, useful for updating an already published script that has undergone modifications without altering its inputs/outputs or dependencies.

Once imported, scripts become exploitable through the DataMiner component GUI which presents two panels (Figure 5):

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

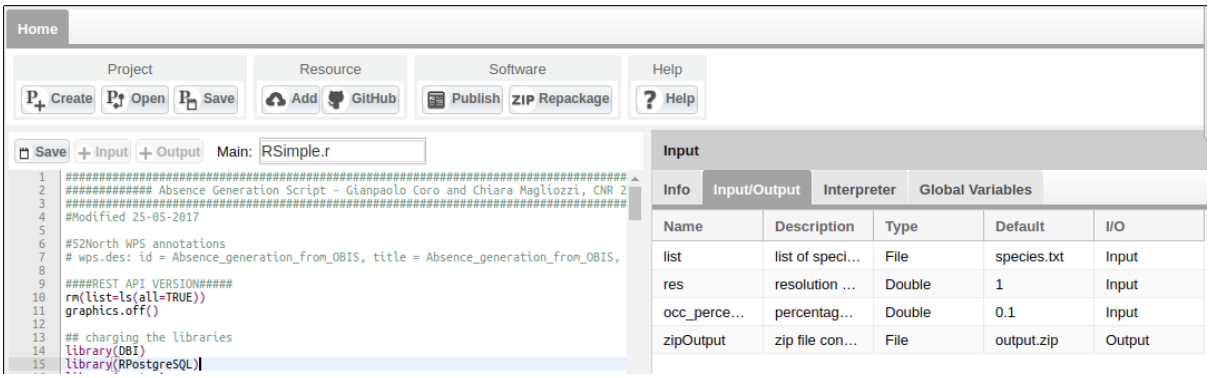


Figure 5 Blue-Cloud VRE Software and Algorithms Importer Interface

- On the left panel, the GUI presents the list of computational methods available in the VLab, which are semantically categorised (the category is indicated through SAI). For each method, the interface calls the WPS DescribeProcess operation to get the descriptions of the inputs and outputs.
- On the right panel, the GUI presents a form allowing to specify the input parameters of the selected computational method. Input data can be selected from the Workspace clearly.

DM is a Java-based Web Service (built on the 52°North WPS framework) operating on Apache Tomcat, enhanced with gCube libraries. The DM architecture consists of two server clusters within a Virtual VRE: the Master Cluster and the Workers Cluster, as shown in Figure 7. Typically, each cluster comprises 16 servers managed by a HA-Proxy load balancer, which evenly distributes requests. Every server runs a DM service, which informs the Resource Registry of its presence and capabilities. The Resource Registry indexes the load balancer, which serves as the primary interface for different DMs. The Workers Cluster is dedicated to cloud computations.

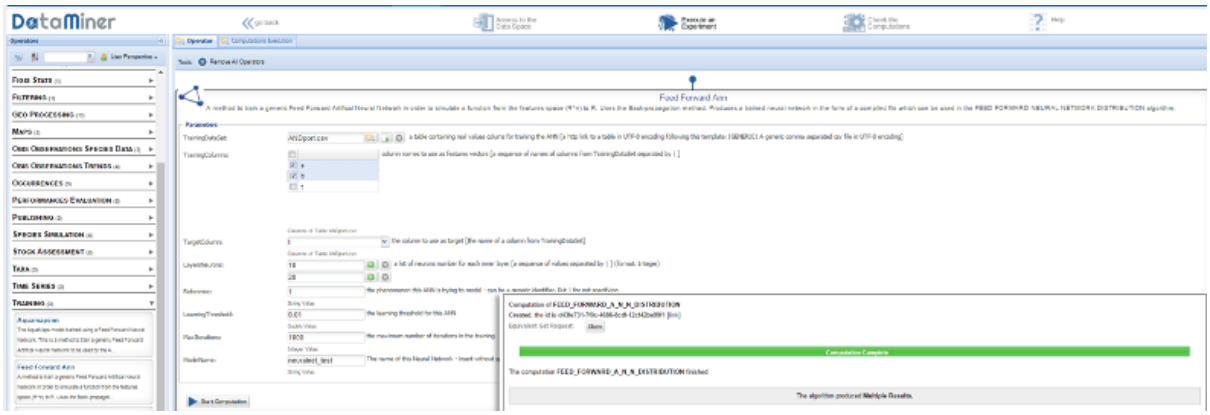


Figure 6 Blue-Cloud VRE Execution Space Interface

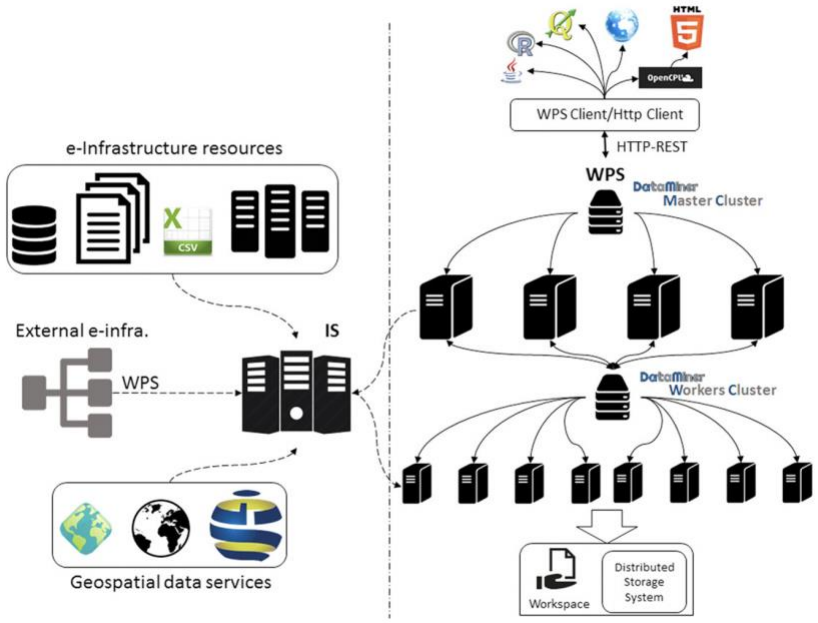


Figure 7 The DataMiner System architecture

The Master and Workers clusters are dynamically set up by D4Science, using an orchestration engine with Ansible scripts for consistent, reliable configuration of multi-tier applications.

Upon receiving a WPS request, the Master Cluster's balancer allocates it to one of its services (master DM). These DMs host various processes from multiple developers, including both local and cloud algorithms. Local algorithms execute directly on the master DMs, making use of parallel processing and significant memory. Cloud algorithms, on the other hand, operate on distributed computing principles like MapReduce and depend on the Workers Cluster's DMs (cloud nodes).

DM enhances the standard 52°North WPS implementation with features that promote user collaboration and result repeatability and reusability. This is achieved through the integration of common services in the infrastructure. Notably, DataMiner can process inputs from workspace folders shared among users, facilitating collaborative complex experiments. Inputs can also be received via HTTP links or embedded in WPS execution requests. The computation outputs are first stored in distributed cloud storage and then returned to the client. Subsequently, a separate thread makes these results accessible in the user's workspace. After each computation, a workspace folder is created containing inputs, outputs, parameters, and a provenance document summarising this data. This folder, shareable with others, allows for the re-execution and sharing of the entire execution process, thereby encouraging collaborative experimentation.

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

3.1.3. Analytics Engines Comparison

Table 1 compares the two different systems: Data Miner and CCP, focusing on various features relevant to their functionalities, across various technical aspects.

For virtualisation, Data Miner utilises “classic” virtual machines whereas CCP employs containers with isolated runtime. In terms of execution infrastructures, Data Miner's infrastructure is chosen by VLab management per VLab, while CCP offers multiple infrastructures selectable by users. For standard protocols, Data Miner uses OGC WPS (XML), contrasting with CCP's OGC API Processes (JSON).

The Method integration in Data Miner is handled through the SAI with GitHub support, while CCP integrates via Method Importer and supports any Code Versioning System (CVS). Data Miner excels in parallel executions, including MapReduce, whereas CCP's parallel execution support is basic but expected to improve. Regarding the availability of methods, Data Miner has many ready-to-use options, in contrast to CCP's fewer out-of-the-box methods. Both systems integrate with JupyterLab and RStudio, but CCP's integration is notably easier, also due to a code generation tool. Finally, the front-end technology of Data Miner is based on Rich Internet Applications (RIA), whereas CCP uses a web component-based approach, exportable in multiple Content Management Systems (CMS).

Table 1 Synoptic table comparing the DataMiner and CCP engines features

Features	Data Miner	CCP
Virtualisation	Virtual Machines	Containers, isolated runtime
Execution Infrastructures	Per VLab, selectable by VLab mgmt.	Multiple, selectable by users
Standard protocol	OGC WPS (XML)	OGC API Processes (JSON)
Method Integrations tool	Software Importer (SAI), Github support	Method Importer / Your IDE, any CVS support
Parallel Executions	Yes, also Map Reduce	Basic, more support will be added
out-of-the-box methods availability	Many	Few
JupyterLab and Rstudio integrations	Yes	Yes, Easier (Code generation tool)
Front-end technology	RIA	Web Component based, exportable in multiple CMS

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union’s Horizon Europe programme under Grant Agreement no. 101094227

The comparison between Data Miner and CCP reveals distinct strengths and preferences in their approach to analytics. Data Miner, with its use of classic virtual machines, excels in parallel executions and offers a wide range of ready-to-use methods, making it robust and immediately functional. Its integration through SAI and GitHub support indicates a focus on established technologies although with a learning curve that may be steeper.

On the other hand, CCP's use of containers and a web component-based front-end technology demonstrates a modern, flexible approach, allowing for a customisable user experience and easier integration with popular platforms like JupyterLab and RStudio.

Moreover, CCP's support for multiple infrastructures and a broader range of CVS, along with its potential for improvement in parallel execution capabilities, shows a system designed for adaptability and future growth. While each system has its unique advantages, the choice between Data Miner and CCP would largely depend on the specific needs of the scientific community, whether it be immediate, out-of-the-box functionality or a more customizable, modern approach.

Funded by the European Union

Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

3.2. JupyterHub

JupyterHub is an open-source web application that allows multiple users to access and use Jupyter notebooks within a shared computing environment. It is particularly valuable in educational and research contexts, as it simplifies the deployment and management of Jupyter notebooks for multiple users. Notebooks are increasingly recognised as the standard tool for offering an easy-to-use online coding system coupled with interactive computing. In the Blue-Cloud VRE, the establishment and management of a Notebook service that integrates with the Blue-Cloud Virtual Labs must meet several key criteria:

- The service must be seamlessly integrated and accessible through various Blue-Cloud Virtual Labs;
- It should be harmonised with the Blue-Cloud VRE Identity and Access Management Service (cf. Section 6.1);
- Access to the Blue-Cloud Storage services (cf. Section 5) is essential, enabling users to interact with files in the VLabs directly from the Notebook environment.
- Users should have the option to select from diverse Notebook versions, differing in pre-installed software and hardware specifications.
- There should be a facility for users to publish their notebooks within the Blue-Cloud VRE catalogue (cf. Section 4).

JupyterHub has been installed on the Blue-Cloud VRE Kubernetes[2] cluster. This setup provides a robust Notebook service offering uninterrupted access from the graphical interface of the Blue-Cloud VLabs.

The reference architecture for the JupyterHub solution within the Blue-Cloud VRE is illustrated in Figure 8. The Kubernetes solution automates the provisioning of Notebook servers. These servers, functioning as containers, provide users with options to choose specific image flavours and set limitations on CPU and RAM usage for each instance. The flexibility of the Kubernetes environment allows for scalable deployment, enabling the addition of new machines (worker nodes) to the existing cluster to enhance service capacity as needed.

[2] https://kubernetes.io/

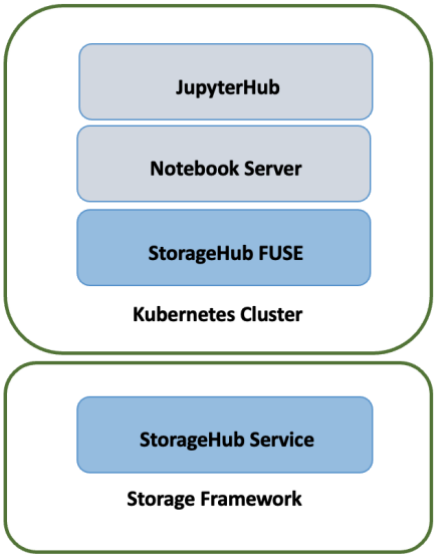


Figure 8 Jupyter Hub solution reference architecture

The project Zero to JupyterHub[3] is pivotal in deploying JupyterHub on Kubernetes. It offers a range of customizable add-ons crucial for integration within the Blue-Cloud framework. One notable feature is the extension of JupyterHub authentication system to incorporate the Blue-Cloud Identity and Access Management Service (cf. section 6.1). A specialised JupyterHub Authenticator class has been developed for this purpose. This class retrieves a user's personal OpenID Connect token from a VLab and employs it to access user account details in the JupyterHub database.

JupyterHub allows users to run different server instances (which might differ in hardware specifications or pre-installed libraries) according to the user's needs. As depicted in Figure 9 and 10, within the Blue-Cloud VRE JupyterHub is used to spawn both JupyterLab Notebooks and RStudio applications on the selected dedicated VRE Cloud servers. Independently from the fact that they are JupyterLab Notebooks or RStudio, Blue-Cloud offers different instances and the possibility to customise them based on the VLab requirements. These instances are similar in capacity and have ready-to-use libraries that are useful to conduct the activities of Blue-Cloud VLabs. Images are automatically built and published to a container registry (e.g., DockerHub) to make them available for the Kubernetes cluster. The system is highly flexible, so new servers can be easily allocated as needed in Blue-Cloud, in order to meet the demands and needs of the users.

[3] https://zero-to-jupyterhub.readthedocs.io/en/latest/

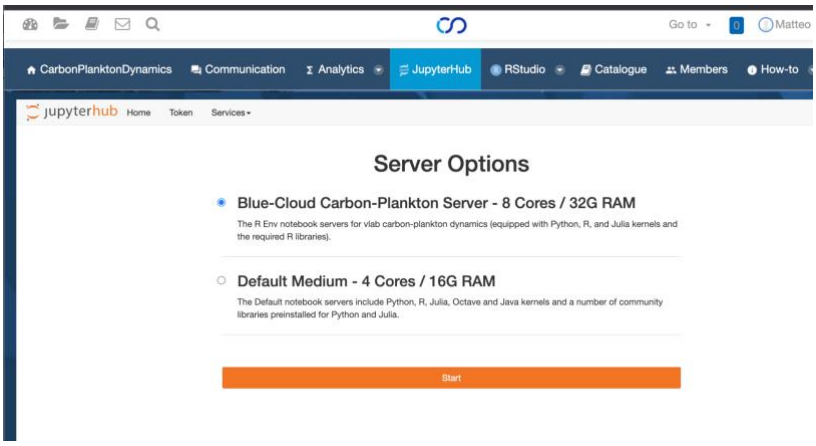


Figure 9 Blue-Cloud JupyterLab dedicated server for VLab Carbon Plankton Dynamics

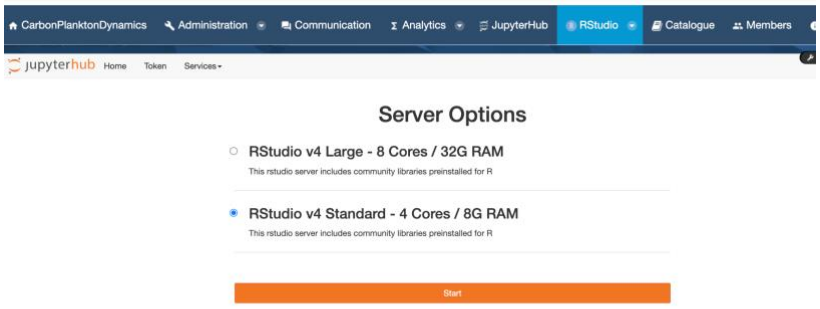


Figure 10 Blue-Cloud RStudio dedicated server for VLab Carbon Plankton Dynamics

Additional integration steps were undertaken to enable access to the Blue-Cloud VRE workspace. This access is established using the StorageHub Fuse library, enabling the mounting of user workspace folders within the Notebook environment. This feature is enabled by the **StorageHub Fuse** component, a custom component developed to work with the Linux Filesystem in Userspace (FUSE)[4] software interface. This connects the Linux filesystem and the Blue-Cloud VRE workspace backend service, i.e., the StorageHub service (cf. section 5.1). The library integration ensures that the workspace can be mounted in a sidecar container (a specific container that runs with the main container to provide extra functionality) alongside the user's primary Notebook container, and it is accessible within the user's home directory.

[4] https://en.wikipedia.org/wiki/Filesystem_in_Userspace

Furthermore, the Kubernetes cluster was configured with an NFS server to accommodate user volumes. This configuration serves the purpose of providing persistent working directories for users and supporting a persistent shared Dataspace (c.f. Section 5.2) for each VLab.

3.2.1. JupyterLab

JupyterLab represents the latest web-based interactive development environment catering to notebooks, code, and data. Its adaptable interface empowers users to customise and organise their workflows in fields such as data science, scientific computing, and machine learning. JupyterLab is seamlessly integrated, ensuring its smooth operation and management within a clustered environment running within the above described JupyterHub service on Kubernetes as depicted in Figure 8.

Since JupyterHub uses ephemeral storage, all the content stored in that host may be removed at the end of the session. All the data, scripts and other resources that the user needs to persist have to be stored into the Workspace or Dataspace (c.f. Section 5) that are accessible through JupyterLab as shown in Figure 11.

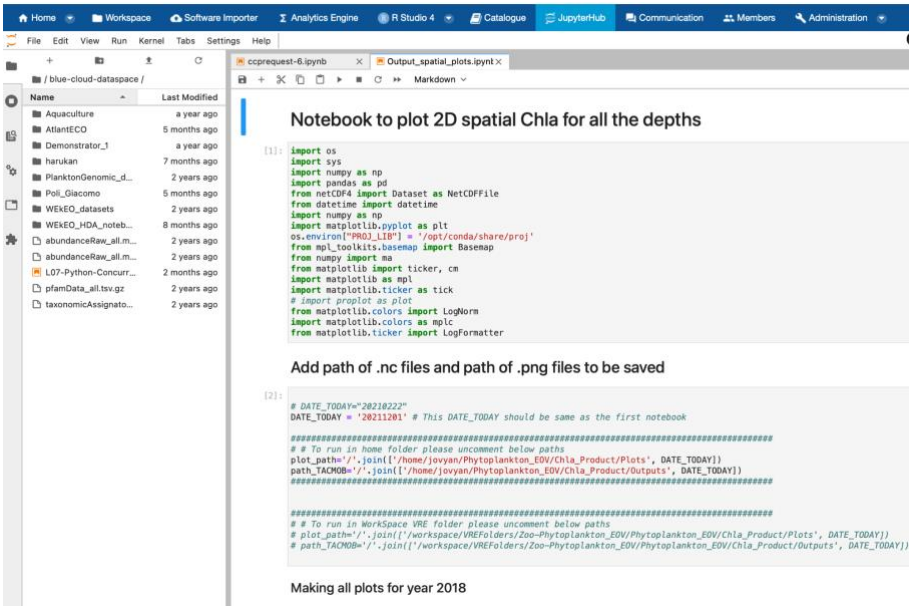


Figure 11 A Notebook in JupyterLab with access to the external volumes Workspace and Dataspace

3.2.3. RStudio

RStudio enables the execution of real-time statistical analyses using R, a programming language designed for statistical computing and graphics. RStudio is seamlessly integrated, ensuring its smooth operation and management within a clustered environment running within the above described JupyterHub service on Kubernetes as depicted in Figure 8.

All the data, scripts and other resources that the user needs to persist have to be stored into the Workspace or Dataspace (c.f. Section 5) that are accessible through RStudio as shown in Figure 12.

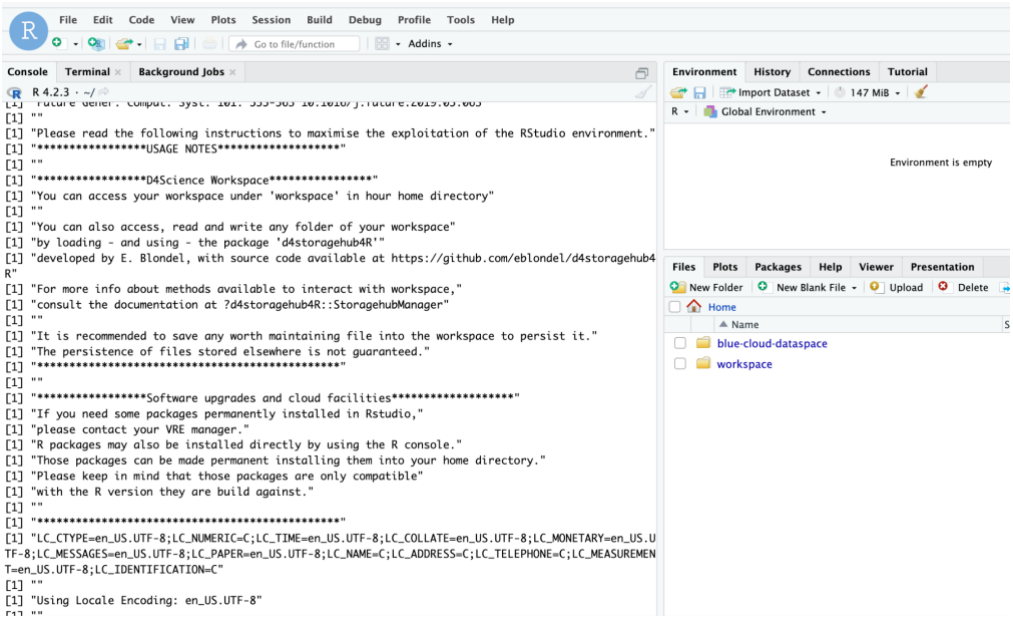


Figure 12 RStudio instance with access to the external volumes Workspace and Dataspace

3.3. ShinyProxy and Containerised Apps

ShinyProxy offers a robust platform for deploying Shiny applications in enterprise environments, eliminating limitations on concurrent app usage. Shiny, a comprehensive R package, simplifies the creation of interactive web applications directly from R, lately Shiny for Python[5] language was also added. These applications can be seamlessly integrated into R Markdown documents or utilised to construct dynamic dashboards, merging R's computational capabilities with the web's interactivity.

Employing ShinyProxy for deploying a Shiny app involves encapsulating the application within an R package, followed by its integration into a Docker image. This methodology brings several significant benefits:

[5] https://shiny.posit.co/py/

- Isolation and Security: Each user session is managed within a completely isolated environment, bolstering security and stability;
- Flexible Environment Management: Supports various Docker images, accommodating different R and Shiny versions;
- Resource Optimization: Memory and CPU usage can be finely tuned via the Docker API.
- Streamlined Monitoring and Debugging: utilises standard Docker tools for effective system monitoring and troubleshooting.

Blue-Cloud VRE and Containers Orchestration

In the Blue-Cloud VRE, container orchestration is a critical component, utilising both Docker Swarm and Kubernetes:

- Docker Swarm: Manages multiple Docker containers across a virtual machine cluster. Key components include a manager container orchestrating the environment and agent containers ('docker machines') that execute Docker containers. The manager employs a least-utilised algorithm for efficient container allocation.
- Kubernetes: As an additional orchestration tool, Kubernetes offers enhanced scalability and management features. It provides robust mechanisms for automated deployment, scaling, and operation of application containers across clusters of hosts, further augmenting the capabilities of Docker Swarm within the Blue-Cloud VRE.

Overall, these technologies, while catering to ShinyProxy, also effectively support the deployment of other Docker-containerised applications, showcasing their adaptability and scalability in the Blue-Cloud VRE platform.

Figure 13 provides a detailed depiction of a Docker Swarm instance actively managing multiple Docker containers distributed across a cluster of virtual machines. This setup is crucial for ensuring efficient resource allocation and high availability in distributed computing environments. Docker Swarm employs a sophisticated management system, central to which is the manager container. This manager container operates on a designated virtual machine, tasked with the comprehensive management of the Docker environment. It is responsible for deploying containers across various agents within the cluster and continuously monitoring and reporting the status and deployment details of each container, thereby maintaining a cohesive and well-organised cluster.

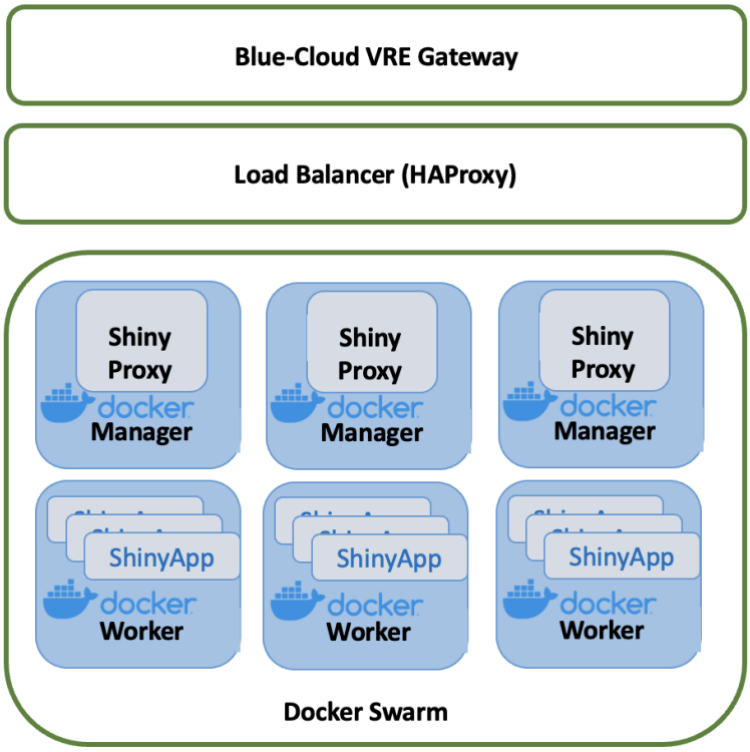


Figure 13 Blue-Cloud VRE cluster supporting Shiny and any other Docker app

The manager container is not merely a supervisory entity, it represents the primary interface for Docker operations, orchestrating the entire container lifecycle within the Swarm. Agents, functioning as 'docker machines', are critical components of this architecture. These agents, hosted on individual virtual machines, register themselves with the manager and are responsible for running the Docker containers. This registration is pivotal for the distributed nature of Docker Swarm, enabling a seamless, coordinated operation across the cluster.

When a client initiates a request to start a container, the manager's role becomes even more significant. It employs a sophisticated algorithm to locate an appropriate agent with available capacity. This algorithm, known as the least-utilised algorithm, ensures equitable distribution of workload by selecting the agent currently running the fewest number of containers to execute the newly requested container. This strategy not only optimises resource utilisation, rather it also improves the overall efficiency of the system.

Additionally, Docker Swarm's architecture requires additional capabilities for robust operation, especially in complex network environments. These include routing external traffic into the cluster, effectively load balancing across multiple container replicas, and facilitating DNS service discovery. These functionalities are vital for ensuring that Docker containers are accessible and operate efficiently under varying load conditions. To address these needs, the Blue-Cloud VRE integrates the HAProxy load balancer. HAProxy is renowned for its high performance and reliability, making it an ideal choice for managing traffic flow,

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

ensuring even distribution of requests across the containers, and providing essential DNS resolution services within the Docker Swarm cluster.

4. Catalogue Framework

The Catalogue Framework is a system realising a flexible and extensible catalogue that has been developed using the open-source data catalogue technology CKAN (ckan.org). It has been enhanced to (a) integrate seamlessly with D4Science services and (b) accommodate a diverse, community-driven, and expandable range of catalogue item types.

Figure 14 depicts the reference architecture of such a framework where (a) the *Catalogue Service* is responsible for the storage and indexing of the metadata of the items published and for supporting the search and browse operation on the catalogue content; (b) the *Catalogue Item Types* is responsible for the definition of item types; (c) the *Catalogue RESTful API* (aka gCat[6]) is a RESTful application for systematically interacting with the Catalogue Service by taking into account the Catalogue Item Types; (d) the *Catalogue Portlet* component implements the main GUI of the entire framework and allows to browse through the catalogue contents; and, (e) the *Publishing Widget* realising a wizard-like GUI component guiding the users to publish items in accordance with the defined item types. Concerning the payloads of the catalogue items, these are either referred by URLs or stored in the Storage Framework, in particular in a specific area of the Workspace (cf. Sec. 5).

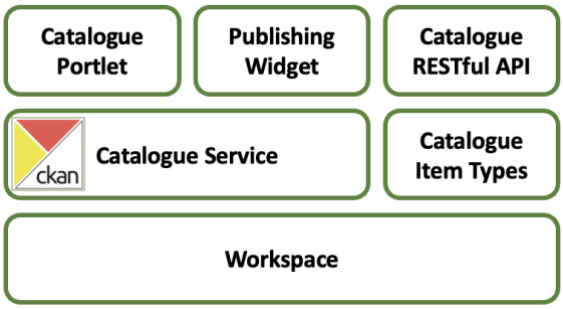


Figure 14 Blue-Cloud VRE Catalogue solution reference architecture

Figure 15 depicts the data model supported by the Catalogue Framework. In particular, it highlights that, every catalogue item:

- is characterised by ***common metadata*** (independently of the typology of the item) including (i) a unique identifier, (ii) a title, (iii) a description, (iv) a number of tags, (v) a licence, (vi) a visibility flag (to indicate whether the item is public, i.e. visible to everyone, or private, i.e. the item is visible to VRE members only), (vii) an author, (viii) a maintainer, and (ix) a typology. A typology adds additional metadata (see item-specific metadata);
- Is characterised by ***item-specific metadata***. Item-specific metadata depends on an ***item profile*** specifying a number of *field specifications* each characterising a metadata field by defining: (i) its name, (ii) the mandatory flag (whether the field is mandatory or optional), (iii) the type of field

[6] gCat documentation https://api.d4science.org/catalogue/docs/index.html

values (i.e. String, Text, Boolean, Number, Geometry in GeoJSon, Time, Time interval, List of Times), (iv) the maximum number of occurrences (i.e. whether the field is repeatable or not), (v) any default value to be proposed, (vi) any accompanying note to facilitate the data entry, (vii) any controlled vocabulary to facilitate the selection of suitable values, (viii) any validator to check the inserted value correctness;Is referring to zero or more ***resources***, i.e., objects representing identifiable payloads. Every resource has (i) an Identifier, (ii) the URL where the payload is stored, (iii) a name, (iv) a description, and (v) a format (e.g., CSV, XML).

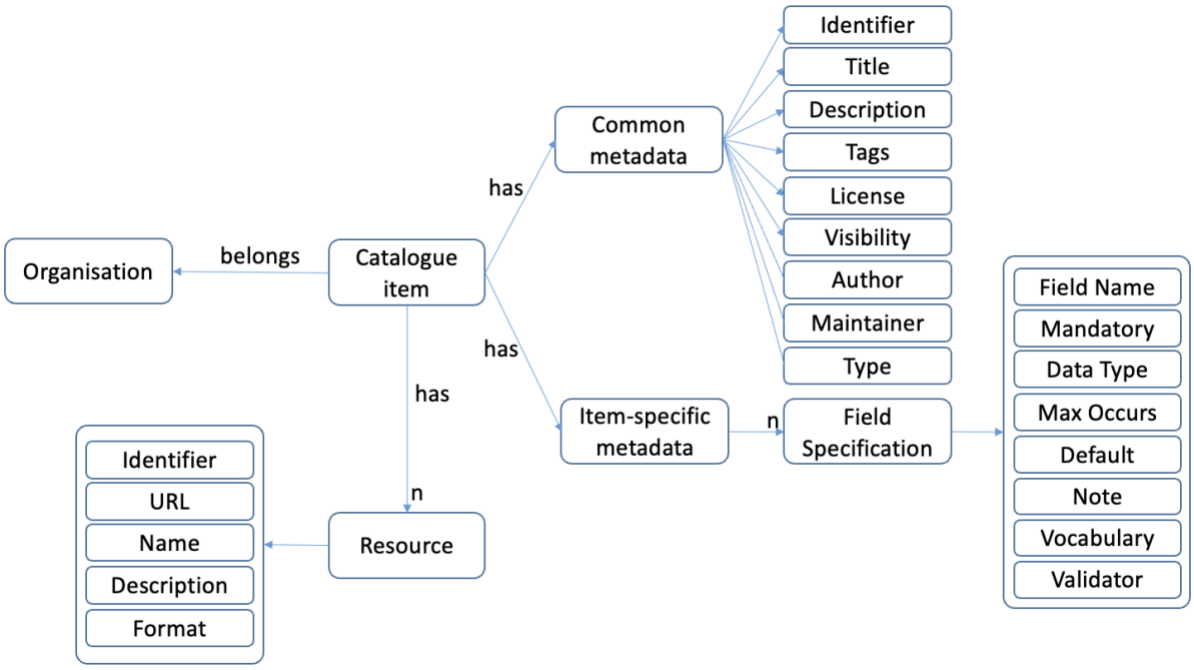


Figure 15 Blue-Cloud VRE Catalogue framework data mode

Figure 16 reports an (eXtensible Markup Language (XML) snippet of an item profile with a metadata field. In practice, an item profile is an XML document containing the name assigned to the specific typology and a list of metadata field specifications that besides characterising the information previously described for item-specific metadata may also contain two directives concerning tagging and group management. The tagging directive instructs the framework to also generate a tag in the common metadata depending on the specific value assigned to the field. The group management directive instructs the framework to assign the catalogue item to a group depending on the value assigned to the specific field.

```
<?xml version="1.0" encoding="UTF-8"?>
<metadataformat type="YOUR TYPE HERE">
    <metadatafield categoryref="category_id_#">
        <fieldId>ID of Metadata Field that identifies the field name in the Document (stored in the Service)</fieldId>
        <fieldName>Name of Metadata Field</fieldName>
        <mandatory>true|false</mandatory>
        <dataType>String|Time|Time_Interval|Times_ListOf|Text|Boolean|Number|GeoJSON</dataType>
        <maxOccurs>N|*</maxOccurs>
        <defaultValue>default value</defaultValue>
        <note>[the note is shown as a suggestion in the insert/update metadata form of Catalogue Publisher Widget]
              </note>
        <vocabulary isMultiSelection="true|false">
            <vocabularyField>field1</vocabularyField>
            <vocabularyField>field2</vocabularyField>
            <vocabularyField>field3</vocabularyField>
        </vocabulary>
        <validator>
            <regularExpression>a regular expression for validating values</regularExpression>
        </validator>
        <tagging create="true|false" separator="char_to_separate">onFieldName|onValue|onFieldName_onValue|onValue_onFieldName</tagging>
        <grouping create="true|false">onFieldName|onValue|onFieldName_onValue|onValue_onFieldName</grouping>
    </metadatafield>
</metadataformat>
```

Figure 16 Blue-Cloud VRE Catalogue framework: Metadata field XML Snippet

Concerning gCat, it offers CRUD (Create, Read, Update, Delete) operation for the following collections:

- Item Collection, i.e., the set of items managed by the catalogue;
- Resource Collection, i.e., the set of resources of a catalogue item;
- Profile Collection, i.e., the set of profiles characterising the catalogue;
- Namespace Collection, i.e., the set of defined namespaces[7];
- License Collection, i.e., the set of licences that can be associated to a catalogue item;
- Trash Collection, i.e., the set of items deleted that have not been removed yet;

In addition to that, administrators of the catalogue can operate on the following collections: Group Collection; Organization Collection; User Collection; Configuration Collection.

Figure 17 depicts two screenshots of the catalogue portlet. This portlet allows its users to browse through the catalogue contents, search for specific items, and access every catalogue item of interest with its related resources. Moreover, the portlet offers a user-friendly interface for allowing authorised users to manage contents, namely creating new items (via the Publishing Widget), altering existing items and removing items.

[7] A namespace is exploited to augment the semantic of a given field, fields sharing the same namespace are managed as a group of connected fields.

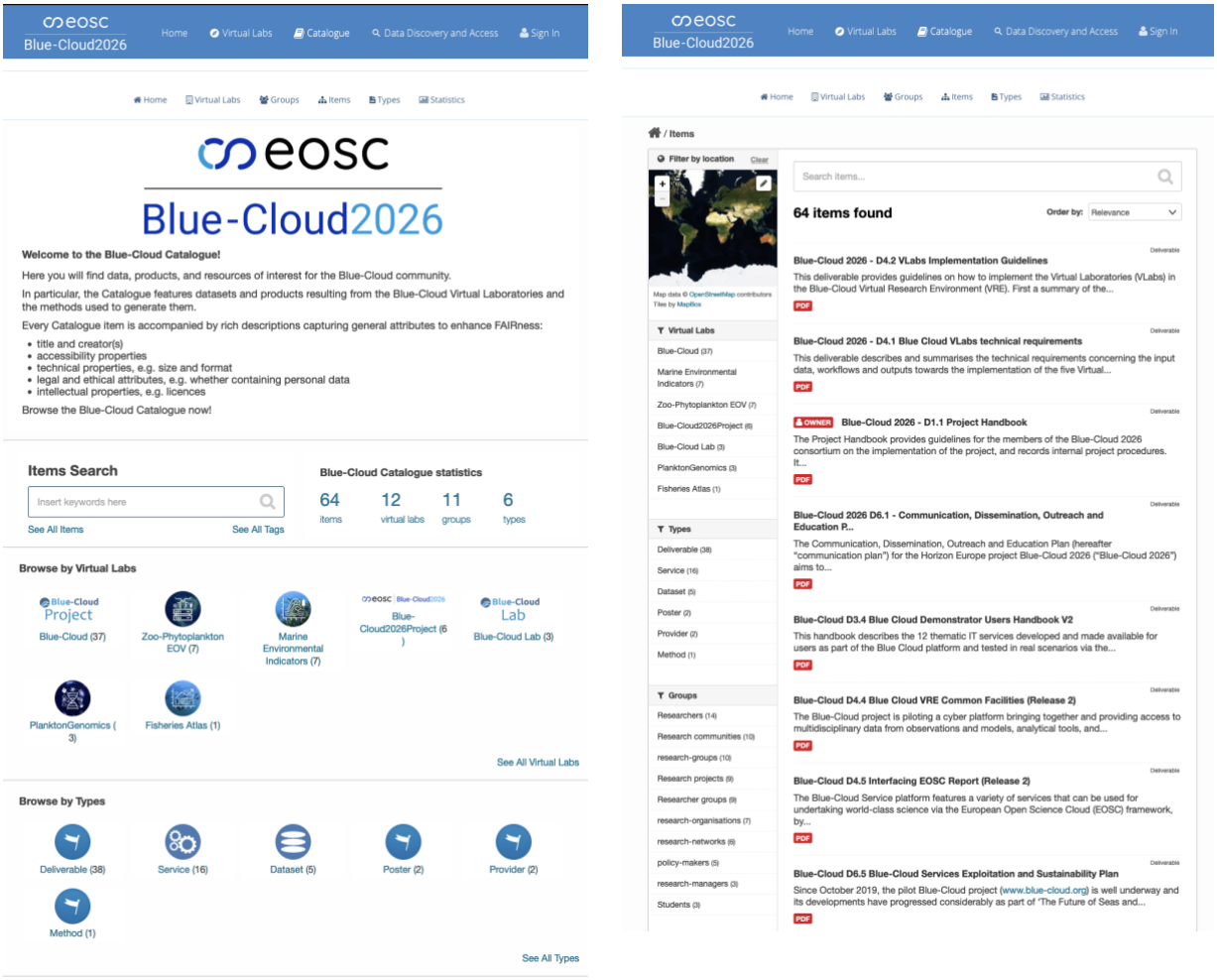


Figure 17 Blue-Cloud VRE Catalogue portlet

5. Storage Framework

The Storage Framework is composed of several storage technologies operated to deliver two key services: the Workspace and the Dataspace. Both of them work like a typical file system with folders that contain files and other digital items (e.g. hard and soft links), and both of them exploit tailored storage devices. Despite the similarities, the Workspace and the Dataspace are designed to own different features and deliver different capabilities that are synthetically reported in the following sections.

5.1. Workspace

The Workspace provides large, secure and reliable access to a storage volume that contains different spaces:

- VRE space: a shared storage area that is linked to any VLab and accessible by all VLab members;
- Shared space: a storage area that is created by a user and shared with other users;
- Private space: a storage area that is exclusive to any user. It hosts the DataMiner space, where the provenance information of each job run by the DataMiner engine is automatically stored; the VRE Data pool, where the datasets accessed through the Data Discovery & Access service are persisted; the CCP space, where the provenance information of each job run by the Cloud Computing Platform engine is stored upon request of the user.
- Volatile space: a temporary and fast space, where to store transient files that are automatically removed after a configurable period of time, set to one week for the Blue-Cloud community.

The access to shared areas is regulated by selecting one of the following access policies:

- Write Own: users can only update/delete their own files;
- Write Any: any user can update/delete any file;
- Read Only: users can read any file but cannot update/delete.

Regardless of the access policies that control the access of all other users, the VLab Manager has the authority to administer the VRE space that is linked to the VLab under their management and the owner of a shared folder can administer the folder that they choose to share.

Figure 18 depicts the Workspace solution reference architecture. From the top, the workspace service acts as the interface layer, comprising the Workspace Portlet, Widget, and RESTful API. These components serve as the user's entry point to the VRE, facilitating interaction through Web Graphical User Interfaces and programmatic access via an API. The Storage Hub serves as an intermediary abstracting the complexity of data storage and retrieval operations. This layer ensures a seamless integration of the data persistence mechanisms, which are represented at the bottom tier of the architecture. The data persistence layer is indeed split into two distinct storage systems: S3 Storage for data, persisted in an object storage for scalable and secure data storage solutions; and Graph DB for metadata, enabling efficient metadata storage, querying, and management.

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

Each workspace element, whether a folder or a file, is equipped with metadata, including a name, a description, creation and update dates, the history of actions performed by any user on that element, and the list of versions for the same file. This set of metadata can be easily extended using the APIs of the Workspace service. Additionally, every file may support a content preview automatically generated according to its content type, and it is also possible to make links (as URIs) to it. For folders, these links can enable others to access the folder in "guest mode", without needing to log in. The contextual menu offers shortcuts for some actions on it, such as publishing the item into the catalogue, or initialising a data miner process with the item as input.

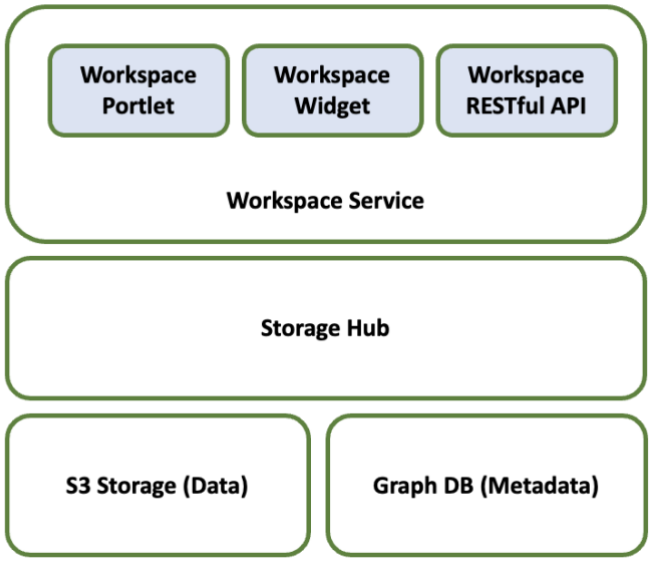


Figure 18 Blue-Cloud VRE Workspace solution reference architecture

The workspace supports in an efficient way encryption, compression, replication, versioning, and persistent unique identifiers as follows:

- **Encryption**: it uses a strong encryption algorithm to protect the data from unauthorised access. The encryption keys are managed by D4Science and depending on the level of control and security required may change for each VRE or community. Encryption is applied to data when they are persisted in the cloud and automatically decrypted when accessed via the Workspace interface or APIs;
- **Compression**: it reduces the size of the data by applying a compression algorithm, such as gzip, to save storage space and bandwidth. Compression is performed by D4Science to data when they are persisted in the cloud and automatically decompressed when accessed via the Workspace interface or APIs;
- **Replication**: it ensures the availability and durability of the data by replicating it across multiple servers and locations. The replication can be synchronous or asynchronous, depending on the consistency and latency requirements. For the Blue-Cloud VRE, synchronous replication has been

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

selected. With synchronous replication data is written to both primary and secondary storage devices at the same time. This ensures that the data is always consistent and up-to-date on all sides. However, this also demands more network capacity and performance and can create latency and complexity. It has been selected to offer to the application's high availability and fast failover. Replication also enables disaster recovery and backup scenarios.

Versioning: it stores multiple versions of a file in the same storage location. This feature lets the user save, access, and restore any version of a file that is stored in the service. With versioning, any user can undo accidental deletion or modification of the data. For example, if the user deletes a file, the Workspace does not erase it completely but moves it to the recycle bin where it can be easily restored. If the user overwrites a file, the Workspace makes a new version and keeps the old one assigning to each version a persistent unique identifier. Each version can be either accessed via the persistent unique identifier or restored as the default version. Versioning enables high availability, durability, and consistency for the data. However, it also consumes more storage space and network bandwidth, as each version of an object is a full copy and not just the difference from the previous version.

Persistent Unique Identifier: it assigns a unique, web-friendly, and persistent identifier to any file and folder stored by it. This identifier does not depend on the location or the content of the object but on the metadata that is written on the Workspace. It can be used to access the object using two types of persistent URLs (PURL) associated with it by the Workspace. The first type of PURL is a link to either the file or the folder that validates the access policies associated with it by verifying the user's identity and access privileges. The second type of PURL is a link to either the file or the folder that enables anonymous access to it.

The Volatile space is configured to lack replication and versioning and to deliver low latency and faster read and write speed. The persistent unique identifiers are UUIDs that are used to access the files via the Workspace APIs.

5.2. Dataspace

The Dataspace provides large and secure access to a storage volume, allowing users to access and share files over a network, as if they were stored locally. It can be configured with different access control lists (ACL) to enable either read-only mode or read-write mode according to the user role in the VRE.

Unlike the workspace, the Dataspace is designed to be stateless. This means that the Dataspace does not store any information about the user's activities and therefore the history of operations performed on the file is not supported. This enables efficient and fast recovery from crashes.

The Dataspace lacks features such as encryption, compression, replication, versioning and persistent unique identifiers. However, it compensates for these limitations by providing large volumes (in the order of TBs) with significantly faster read and write speed, lower latency, and lower congestion compared to the Workspace.

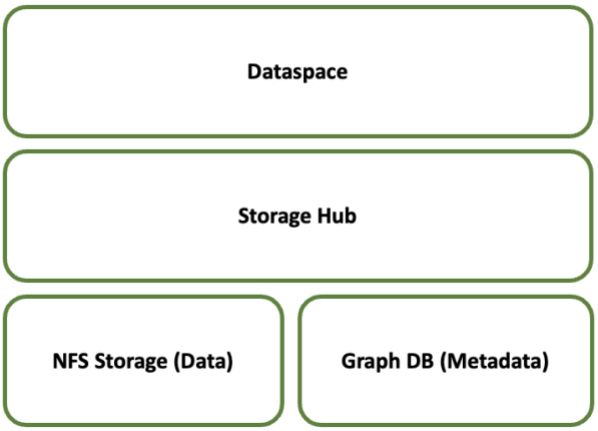


Figure 19 Blue-Cloud VRE Dataspace solution reference architecture

Figure 20 highlights the main pros and cons of the Workspace and Dataspace services.

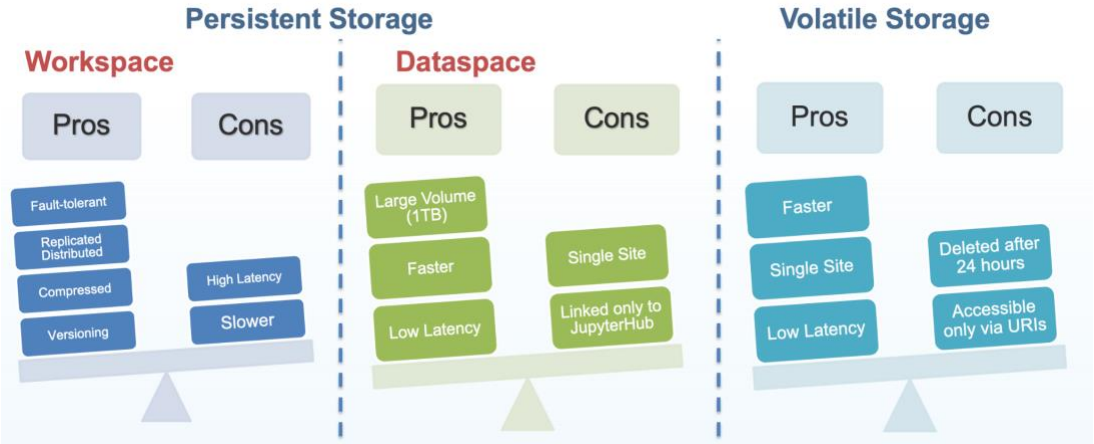


Figure 20 Blue-Cloud VRE Workspace vs. Dataspace; Persistent vs. Volatile storage

6. Enabling Framework

6.1. Identity and Access Management

The IAM (Identity and Access Management) system, developed during the previous Blue-Cloud Project, features a specialised IAM Service that oversees User Identity and Authorisation, including the login process. Keycloak[8], a component recognised for its industry-standard quality in IAM software, efficiently manages all aspects of Identity Management workflows. This encompasses seamless integration with various Identity Providers such as Google, LinkedIn, and the EOSC Portal.

As shown in Figure 21**,** the VRE Gateway service, utilising Liferay web portal technology, is designed to store only the essential user information necessary for its navigational and visualisation functions. It operates on a user information cache that is updated with each user login, thereby minimising its dependence on specific technologies, versions, and bespoke code artefacts.

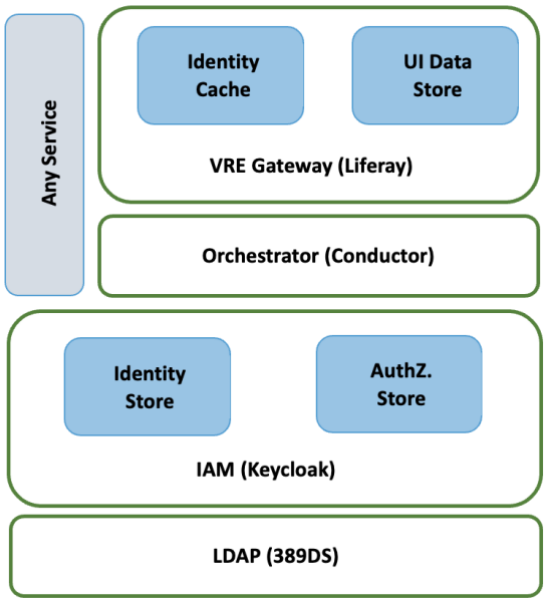


Figure 21 IAM Solution Reference Architecture

A key component of the updated IAM architecture is the Orchestrator service, which facilitates the creation of workflows that an engine subsequently executes. The primary objective of these orchestrator workflows is to reduce the load on services acting as event sources. These services are no longer required to orchestrate or be aware of complex details independently. User-related events originating from these services, including the IAM, are dispatched to the Orchestrator. The Orchestrator then assumes the responsibility of informing the relevant services directly by implementing the necessary workflow. Finally, the IAM service transfers data to a LDAP server (based on 389DS) within the Blue-Cloud VRE. This server

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

is utilised by several key tools: (a) the software code repository tool (Gitea-based), (b) the software integration tool (Jenkins-based), and (c) the issue tracker tool (Redmine-based). These tools employ the server for authenticating project members, enabling them to access these resources using their Blue-Cloud credentials.

This IAM solution is notable for its interoperability, adhering to open standards like OAuth2, User-Managed Access (UMA), and OpenID Connect (OIDC) protocols. The incorporation of UMA and OIDC, building upon the OAuth2[9] protocol, represents a significant enhancement. UMA focuses primarily on Authorisation, providing a streamlined protocol for centralised access control. It enables resource owners to establish detailed authorisation policies on a centralised server, which then controls access to these resources. OpenID Connect (OIDC) adds an identity layer to OAuth2, utilising JSON web tokens (JWT). This enables third-party applications to verify the user's identity through an Authorisation Server and to access basic user profile information. The effectiveness and efficiency of OIDC in identity verification, as evidenced during the Blue-Cloud Project, have been well-demonstrated and highly valued in various applications. Within the first 12 months of the project, this system has been updated to align with evolving technologies.

6.2. Information System

The purpose of the Information System (IS) is to provide access to well-organised information. In the Blue-Cloud VRE, such information is related to the set of resources (data, services, and tools) describing the VRE. The Blue-Cloud VRE manages a combination of these resources, offering capabilities and capacities for data-driven research via the Virtual Laboratories (VLabs).

To effectively monitor the allocation of these (federated) resources to VLabs, the VRE relies on the IS, which provides capabilities for publishing, discovering, and accessing these resources. Some of these resources are VRE core resources, while others are leveraged from external systems. Hence, the IS has to deal with:

- different set of managed resources: each system composing the federation defines the resources it wishes to contribute to the VRE;
- different models for describing resource characteristics: each system has its own model to define a resource. This mainly means that the same resource type may be described differently by the different organisations contributing to the VRE;
- different workflows governing registration and update of resources: each system may update any resource description at different time-frames and according to different policies.

In the VRE, the subset of resources exposed to a VLab are usually different from the subset exposed to another VLab. The subset of resources of a VLab can intersect (and usually does) the subset of resources of another VLab. A change that occurs to one or more resources exposed to different VLabs must be reflected consistently across all VLabs.

[9] https://oauth.net/2/

In the IS, the resources are defined using a model known as the IS Model, designed at ISTI - CNR [6] to support infrastructure federations such as Blue-Cloud VRE properly. The IS Model employs a graph model framework, which is advantageous in scenarios where the interconnectivity and topological information of data are as significant as the data itself.

Graph models are particularly apt in areas that require detailed mapping of data relationships and structures. In the IS Model, this approach facilitates a dynamic and intricate description of resources through the assembly of various components, or 'bricks', known as 'Facets'. These facets enable a flexible and customisable method for defining resources, meeting the unique requirements of the e-infrastructure.

Figure 22 depicts the IS and VRE Management Systems architecture, central to which is the Resource Registry. This registry plays a central function in the system. Firstly, it empowers the VRE Management System (as detailed in Section 6.3) to establish the necessary context or 'room' for the integration and management of VLab resources. This setting of context is essential for resource incorporation within the Vlab. Secondly, the VRE Management System utilises the registry to define the specific VRE Model – the Blue-Cloud Model. This involves identifying the resources and their facets and mapping the relationships between these resources, ensuring a cohesive and interconnected framework within the VRE. Lastly, the Resource Registry facilitates any Blue-Cloud service to efficiently register, discover, and access resource definitions within the Vlab's operational context. This feature is crucial for the seamless integration and operation of services within the Blue-Cloud ecosystem.

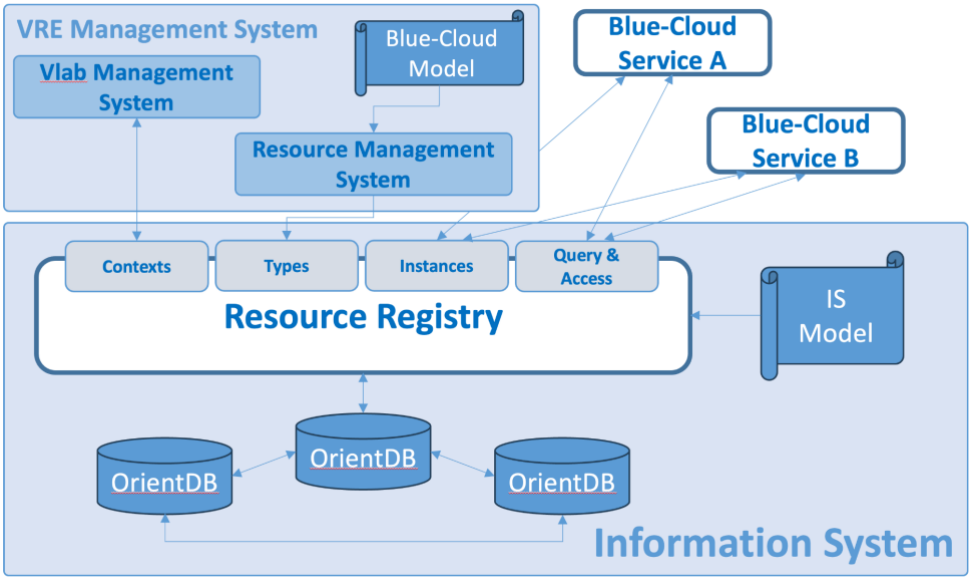


Figure 22 Information System and VRE Management System Reference Architecture

The IS Model is a cornerstone of the Blue-Cloud VRE, offering a sophisticated and adaptable approach to resource management and integration, crucial for the effective functioning of the VRE environment at large.

Funded by the European Union
Blue-Cloud 2026 – A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters has received funding from the European Union's Horizon Europe programme under Grant Agreement no. 101094227

6.3. VRE Management System

The VRE Management System encompasses a range of services and applications that provide functionalities for the design, creation, and deployment of VLabs. These services facilitate VLab Designers and VRE Managers in using graphical user interfaces to communicate the required features of a desired VLab to the Blue-Cloud infrastructure. They also allow for easy updates to the VLab once it is defined and operational.

Managing these collaborative environments involves a four-stage process:

1. A definition stage where a VLab Designer outlines the features of a new VLab to support a specific application scenario;
2. An approval stage where the VRE Manager decides on the feasibility of the proposed VLabs, determining whether they should be accepted or rejected. For accepted VLabs, the VRE Manager also decides on deployment details, such as the hosting nodes to be used;
3. A verification stage where the VRE Manager validates the VLab as per the specifications approved earlier;
4. A management stage where the VRE Manager customises certain elements of a deployed VLab (such as the layout of the user interface components, known as portlets) and monitors the overall operational status of the VRE.

For more comprehensive information, visit the gCube Wiki page dedicated to this service at https://wiki.gcube-system.org/gcube/VRE_Administration.

This system, which pre-dates the Blue-Cloud initiative, has been redesigned over time to align with evolving technologies.

References

[1] M. Assante, L. Candela, D. Castelli, R. Cirillo, G. Coro, L. Frosini, L. Lelii, F. Mangiacrapa, V. Marioli, P. Pagano, G. Panichi, C. Perciante, F. Sinibaldi ***The gCube system: Delivering Virtual Research Environments as-a-Service***. (2019) Future Gener. Comput. Syst. 95: 445-453 10.1016/j.future.2018.10.035

[2] M. Assante, L. Candela, D. Castelli, R. Cirillo, G. Coro, L. Frosini, L. Lelii, F. Mangiacrapa, P. Pagano, G. Panichi, F. Sinibaldi ***Enacting open science by D4Science***. (2019) Future Gener. Comput. Syst. 101: 555-563 10.1016/j.future.2019.05.063

[3] M. Assante, L. Candela, P. Pagano. (2021). **Blue-Cloud D4.4 Blue Cloud VRE Common Facilities (Release 2)**. Zenodo. 10.5281/zenodo.10070443

[4] L. Candela, D. Castelli and P. Pagano, ***The D4Science Experience on Virtual Research Environment Development***. Computing in Science & Engineering, vol. 25, no. 2, pp. 12-19, March-April 2023, doi: 10.1109/MCSE.2023.3290433

[5] M. Assante, L. Candela, D. Castelli, R. Cirillo, G. Coro, A. Dell'Amico, L. Frosini, L. Lelii, F. Mangiacrapa, P. Pagano, G. Panichi, F. Sinibaldi (2023) ***Virtual research environments co-creation: The D4Science experience.*** Concurrency Computat Pract Exper. 2023; 35(18):e6925. *doi:10.1002/cpe.6925*

[6] Frosini, L. (2019). **Transactional REST Information System for Federated Research Infrastructures enabling Virtual Research Environments**. Doctoral Thesis. May 2019, doi: 10.5281/zenodo.7464023

[7] G. Coro, G. Panichi, P. Scarponi, P. Pagano, P. (2017) **Cloud computing in a distributed e-infrastructure using the web processing service standard.** Concurrency Computat: Pract Exper. 2017; 29:e4219. doi: *10.1002/cpe.4219*