D4.2 Blue Cloud VRE Common Facilities (Release 1)

Work Package	WP4, Developing and operating the Blue Cloud VRE, its services and Virtual Labs
Lead Partner	CNR
Lead Authors (Org)	Massimiliano Assante (CNR), Leonardo Candela (CNR), Pasquale Pagano (CNR)
Contributing Author(s)	Andrea Dell'Amico (CNR), Gianpaolo Coro (CNR), Roberto Cirillo (CNR), Luca Frosini (CNR), Lucio Lelii (CNR), Marco Lettere (Nubisware), Francesco Mangiacrapa (CNR), Giancarlo Panichi (CNR), Fabio Sinibaldi (CNR)
Reviewers	Anton Ellenbroek (FAO), Francesca Spagnoli (TRUST-IT)
Due Date	31-10-2020, M13
Submission Date	06-11-2020
Version	1.0

Dissemination Level

X	PU: Public
	PP: Restricted to other programme participants (including the Commission)
	RE: Restricted to a group specified by the consortium (including the Commission)
	CO: Confidential, only for members of the consortium (including the Commission)

Blue-Clou Innovative services for Marine Research & the Blue Economy - has received funding from the European Union's Horizon p call BG-07-2019-2020, topic: [A] 2019 - Blue Cloud services, Grant Agreement n. 862409.

DISCLAIMER

"Blue-Cloud, Piloting Innovative services for Marine Research & the Blue Economy" has received funding from the European Union's Horizon programme call BG-07-2019-2020, topic: [A] 2019 - Blue Cloud services, Grant Agreement n.862409.

This document contains information on Blue-Cloud core activities. Any reference to content in this document should clearly indicate the authors, source, organisation, and publication date.

The document has been produced with the funding of the European Commission. The content of this publication is the sole responsibility of the Blue-Cloud Consortium, and it cannot be considered to reflect the views of the European Commission. The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content.

COPYRIGHT NOTICE

VERSIONING AND CONTRIBUTION HISTORY

Version	*Date*	*Authors*	*Notes*
0.1	01/10/2020	CNR	Definition of the deliverable Table of Contents and planning of contributions
0.5	19/10/2020	CNR	The first version of planned contributions integrated into the overall document.
0.8	02/11/2020	CNR	Complete version ready for review submitted.
0.9	02/11/2020	FAO	Reviewer commented version
1.0 RC	03/11/2020	CNR	Comments review and preparation of the release candidate

Contents

Table of Figures

Executive summary

The Blue-Cloud project plans to pilot a cyber platform bringing together and providing access to multidisciplinary data from observations and models, analytical tools, and computing facilities essential to support research to understand better and manage the many aspects of ocean sustainability.

To achieve this goal, Blue-Cloud is developing, deploying, and operating the Blue-Cloud platform whose architecture consists of two families of components: (a) the *Blue Cloud Data Discovery and Access* service component to serve federated discovery and access to 'blue data' infrastructures; and (b) the *Blue Cloud Virtual Research Environment (VRE)* component to provide a Blue Cloud VRE as a federation of computing platforms and analytical services.

This deliverable presents the Blue Cloud Virtual Research Environment constituents by focusing on both new services and revised existing services that have been developed in the reporting period to serve the needs of the Blue Cloud community. In particular, this deliverable describes a total of 11 services and components. These services and components contribute functionalities to the Blue Cloud VRE Enabling Framework (Identity and Access Management, VRE Management), Collaborative framework (Workspace and Social Networking), Analytics Framework (Software and Algorithm Importer, Smart Executor), Publishing Framework (Catalogue Service) and improved support for RStudio, JupyterHub, ShinyProxy, and Docker Applications. The services are described below by reporting their design principles, architectures, and main features.

The deliverable also describes the procedures and approaches governing services and components released by highlighting how Gitea (as Git hosting service), Jenkins (as automation server), and Maven (as project management and comprehension tool) are used to guarantee continuous integration processes.

Services and components discussed in this deliverable contribute to 11 gCube open-source software system releases (from gCube 4.16 up to gCube 4.25.1) and are in the pipeline for the next ones. They have been used to develop and operate the Virtual Laboratories of the Blue Cloud gateway https://blue-cloud.d4science.org and its underlying infrastructure. At the time of this deliverable (November 2020), the gateway hosts a total of 8 VREs and VLabs, including five specifically conceived to support the co-development of some of the Blue-Cloud demonstrators (namely, the Aquaculture Atlas Generation for Demonstrator #5, the Blue-Cloud Lab for several demonstrators, the GRSF pre for Demonstrator #4, the Marine Environmental Indicators for Demonstrator #3, the Zoo-Phytoplankton EOV for Demonstrator #1). This gateway and its tools serve more than 400 users that (since January 2020) performed a total of more than 5000 working sessions, more than 1700 accesses to the Workspace, and more than 750 analytics tasks. These exploitation and uptake indicators are likely to grow in the coming months thanks to data updates and continued use, further development of existing VLabs, and finally, the creation of new ones.

1. Introduction

The Blue-Cloud project is part of 'The Future of Seas and Oceans Flagship Initiative' and is undertaken by a consortium of 20 organisations. It aims:

- To build and demonstrate a Pilot Blue Cloud by combining distributed marine data resources, computing platforms, and analytical services;
- To develop services for supporting research to understand better & manage the many aspects of ocean sustainability;
- To develop and validate several demonstrators of relevance for marine societal challenges;
- To formulate a roadmap for the expansion and sustainability of the Blue Cloud infrastructure and services.

The project federates leading European marine data management infrastructures (SeaDataNet, EurOBIS, Euro-Argo, Argo GDAC, EMODnet, ELIXIR-ENA, EuroBioImaging, CMEMS, C3S, and ICOS-Marine), and horizontal e-infrastructures (EUDAT, DIAS, D4Science) to capitalise on what exists already and to develop and deploy the "Blue Cloud" framework. The federation will be realized at the levels of data resources, computing resources, and analytical service resources.

The federation requires the development of a system whose architecture was presented in ***Deliverable 2.6 Blue Cloud Architecture (Release 1)*** (Schaap et al. 2020). The document clarified that the technical framework consists of two major families of components:

- the ***Blue Cloud Data Discovery and Access*** component to serve federated discovery and access to blue data infrastructures;
- the ***Blue Cloud Virtual Research Environment (VRE)*** component to provide a Blue Cloud VRE as a federation of computing platforms and analytical services.

The Blue Cloud Virtual Research Environment components range from services to promote the collaboration among its users to services supporting the execution of analytics tasks embedded in a distributed computing infrastructure, to services enabling the co-creation of entire Virtual Laboratories. This component is built on the D4Science infrastructure and the gCube open source technology (Assante et al. 2019a, 2019b) and deployed via the Blue Cloud gateway (accessible at https://blue-cloud.d4science.org) to make the services and Virtual Laboratories available.

The deliverable complements and updates the description of the Blue Cloud Virtual Research Environment previously included in Deliverable D2.6 by focusing on both new services and revised versions of existing services that have been further developed in the reporting period to serve the needs of the Blue Cloud community. This document describes selected services by highlighting their design principles, architectures, and main features. These components contribute to 11 gCube releases, from gCube 4.16 (November 2019) to gCube 4.25.1 (October 2020). Some of them are in the pipeline producing the forthcoming gCube release.

The deliverable is organised as follows. Section 2 briefly recalls the Blue Cloud VRE Architecture. Section 3 describes the two services contributing to the Enabling Framework part, namely the Identity and Access Management solution and the VRE Management solution. Section 4 describes the two services contributing to the Collaborative Framework part, i.e., the Workspace and the Social Networking service. Section 5 documents the six services contributing to the Analytics Framework, namely the Software and Algorithm Importer, the Smart Executor, RStudio, JupyterHub,

ShinyProxy, and DataMiner for DockerApps. Section 6 describes the service contributing to the Publishing Framework, namely the Data Catalogue. Section 7 documents the process enabling the release of these components and their integration into the gCube technology releases and their deployment into the D4Science infrastructure. Finally, Section 8 concludes the report.

2. Blue Cloud VRE Architecture

The Blue Cloud overall architecture was discussed in depth in Deliverable 2.6. In particular, the document clarified that the technical framework of Blue-Cloud consists of two major families of components:

- the ***Blue Cloud Data Discovery and Access*** service component to serve federated discovery and access to blue data infrastructures;
- the ***Blue Cloud Virtual Research Environment (VRE)*** component to provide a Blue Cloud VRE as a federation of computing platforms and analytical services.

This deliverable focuses on the VRE part by highlighting the new components and the extensions and enhancements implemented to the software frameworks powering the Blue Cloud VRE.

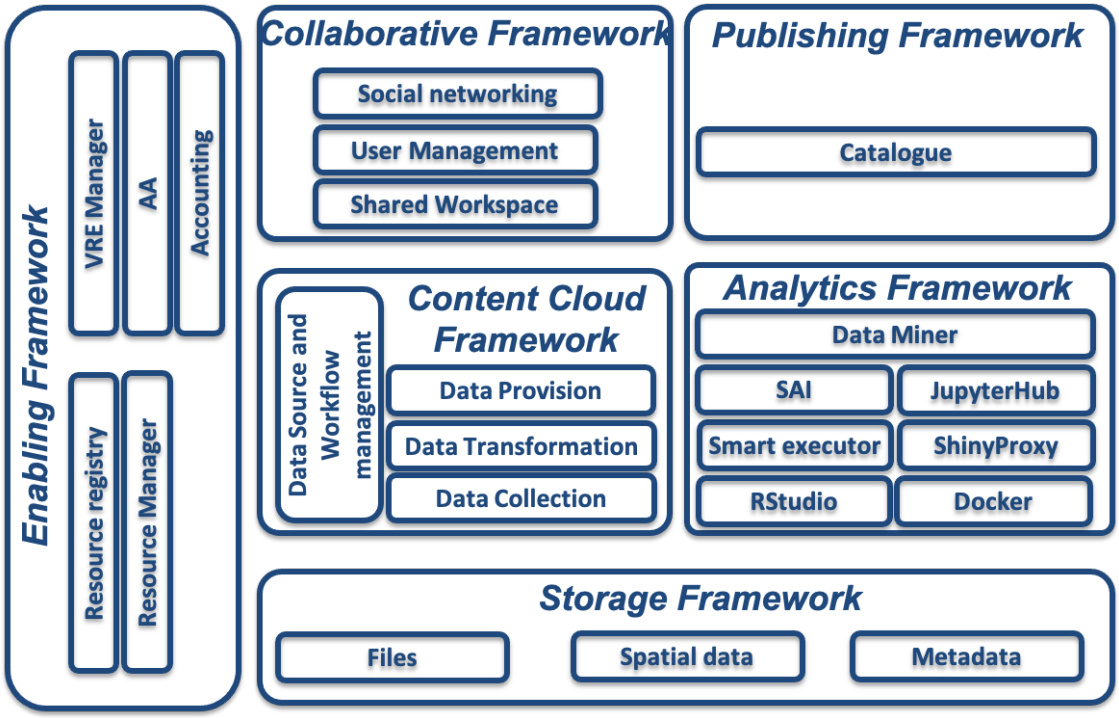

Figure 1. Blue Cloud VRE Architecture

Figure 1 depicts the revised version of the Blue Cloud VRE architecture resulting from the extension and enhancement of the architecture presented in D2.6 to serve the needs and requirements emerging in the Blue Cloud domain. The architecture consists of services and components organised in 6 frameworks:

- ***Enabling Framework***: to support the operation of all services, VREs and VLabs. In particular, it includes (a) a resource registry service, to which all the e-infrastructure resources (data sources, services, computational nodes, etc.) can be dynamically (de)registered and discovered by users and other services; (b) authentication and authorization services, as well as auditing services, capable of granting and tracking access and usage actions from users; (c) a VRE management approach for deploying specialized VREs/VLabs based on a selected subset of "applications". This document complements the description of these

components stemming from D2.6 by documenting the new solution for Identity and Access Management (cf. Sec. 3.1) and the VRE Management (cf. Sec. 3.2).

- ***Storage Framework***: includes services for efficient, advanced, and on-demand management of digital data represented by files in a distributed file system, collections of metadata records, and time series in spatially-enabled databases. It is used by most other services of the Blue Cloud VRE Architecture.
- ***Content Cloud Framework***: includes services required to collect, transform, harmonize, and provide, via a variety of APIs, all metadata records published by the D4Science community and those provided by the organizations integrated by the D4Science consortium. It is not planned to be exploited by Blue-Cloud and thus is not documented by this document.
- ***Collaborative framework***: supports all the VREs and VLabs deployed and provides social networking, user management, and shared workspace services, and Web-based User Interface access to the information cloud and to the analytics framework via analytics laboratory services. We describe here the Workspace (cf. Sec. 4.1) and the Social Networking components (cf. Sec. 4.2) of this framework.
- ***Analytics Framework***: includes the services required for executing analytics methods and processes provided by the scientists. It uses, in a transparent way, the power of the underlying distributed computing cloud and of a plethora of standard statistical methods provided out of the box to compute over input data. Compared to the services included in D2.6, this framework has been extended by a revised version of the Software and Algorithm Importer component (cf. Sec. 5.1), the Smart Executor (cf. Sec. 5.2), the RStudio-based solution (cf. Sec. 5.3), the JupyterHub-based solution (cf. Sec. 5.4), the ShinyProxy-based solution for ShinyApps (cf. Sec. 5.5), and the DataMiner-based solution for DockerApps (cf. Sec. 5.6).
- ***Publishing framework***: services enabling users to document and make “public” any artifact, i.e., made available online. It primarily consists of a catalogue service (cf. Sec. 6.1), where diverse typologies of objects (e.g., datasets, processes, services) can be described, catalogued, and made searchable and accessible using user/community defined metadata.

3. Enabling Framework Components

According to Section 2, the Blue Cloud VRE Enabling Framework is composed of five distinct macro components. In this chapter, we report on the ones that have been either re-implemented or have been re-designed to adapt to the evolving technologies. The VRE common facilities for the Enabling Framework part are represented by the following two macro components:

- Identity and Access Management (IAM);
- VRE Management.

3.1 Identity and Access Management (IAM)

In the previous architecture, the Identity and Access Management Service (IAM) was "encapsulated" into two different services, namely the Identity Management Service (IdM) and the Authentication and Authorization Service (AA). The former was available into the Gateway service (Liferay web portal technology) and offered features such as login via a federated Identity Provider (e.g., Google, LinkedIn, EOSC Portal) and via the oAuth2 standard. The latter was based on the gCube Authorization framework[1], a token-based authorization system in a gCube-based infrastructure, such as D4Science, compliant with the Attribute-based access control (ABAC)[2] that defines an access control paradigm, whereby access rights are granted to users through the use of policies which combine attributes.

The IdM and the AA services together provided the IAM solution of D4Science for more than eight years. However, we realised that the technology in which this solution was implemented could not fulfil the relevant requirements in terms of openness and international standards adoption of the research infrastructures and e-infrastructures Blue-Cloud builds upon. Therefore, we introduced an industry level Authorization Provider software, granting as much functionality as possible and respecting the philosophy of being open-source, extensible by nature, and compliant with open and international standards for fine-grained authorization workflows.

[1] https://wiki.gcube-system.org/gcube/Authorization_Framework
[2] https://en.wikipedia.org/wiki/Attribute-based_access_control

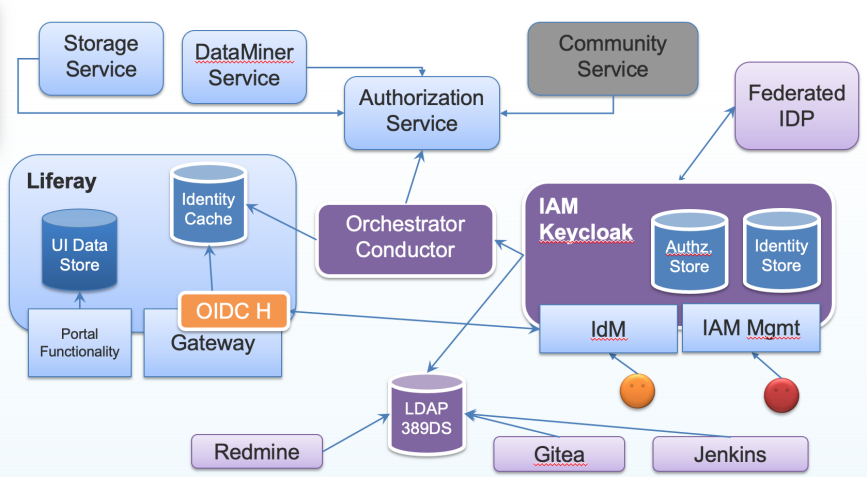

Figure 2. The Identity and Access Management Architecture

Figure 2 shows the architecture of the new IAM solution. It introduces a dedicated IAM Service that governs User Identity and Authorization management including the login process. The entire responsibility of governing Identity Management related workflows is offloaded to the above-mentioned industry quality IAM software component Keycloak[3], including the federation of the Identity Providers (e.g., Google, LinkedIn, EOSC Portal).

The Gateway service (the Liferay web portal technology) only keeps as much User information as necessary to fulfil its navigation and visualization requirements. It takes advantage of a User information cache, which is created or updated every time a user logs into the Gateway. This makes it much less dependent on technology, version, and custom code artefacts.

The Authentication and Authorization Service (Authorization Service in Figure 2) processes, which will remain supported for backward compatibility, are paired with a service provided by the IAM architecture that is interoperable by design, as it adheres to open standards such as OAuth2, User-Managed Access (UMA), and OpenID Connect (OIDC) protocols.

While oAuth2 protocol was already supported in the previous IAM solution, UMA and OIDC were not. Both protocols extend the oAuth2 protocol: UMA is more focused on the Authorization part while OIDC on the Authentication one. UMA is a lightweight access control protocol that defines a centralized workflow to allow an Entity to manage access to its resources. Specifically, it gives resource owners granular management of their protected resources by creating authorization policies on a centralized authorization server; this server then authorises who and what can get access to these resources and for how long. OpenID Connect (OIDC) is an identity layer over oAuth2 which uses JSON web tokens (JWT), allowing third-party applications to verify the identity of the User based on the authentication performed by an Authorization Server and to obtain basic user profile information. During the past years, OIDC has become the leading standard for single sign-on and identity provision on the Internet.

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

A central part of the new IAM architecture is represented by the Orchestrator service, based on the Conductor[4] technology. Conductor is a Workflow Orchestration engine that runs in the Cloud. Conductor Workflows are defined using a JSON based DSL and include a set of tasks that are executed as part of these workflows. These features made it the ideal candidate to manage the IAM one-to-many communication patterns among the services involved. As a matter of fact, the orchestrator service reduces the burden of services that act as Event sources, which are not called to orchestrate and know the details on their own. User related events generated from these services (including the IAM) are sent out to the Orchestrator, which takes care of notifying the "interested" services directly by applying the required JSON based workflow.

Lastly, the IAM service exports data onto a new LDAP (389DS based) server of the Blue-Cloud infrastructure, which is used by *(i)* the software code repository tool (based on Gitea), *(ii)* the software integration tool (based on Jenkins) and *(iii)* the issue tracker tool (based on Redmine), to authenticate project members and allow them access to these tools with their Blue-Cloud credentials.

3.2 VRE Management

VRE Management facilities comprise a set of services and applications offering functions for defining, creating, and deploying VLabs. These services support VLab Designers and VRE Managers through graphical user interfaces to instruct the Blue-Cloud infrastructure on the expected features of the desired VLab, as well as allowing to easily update the VLab once defined and operational.

The administration of these cooperation environments is a four-tasks activity envisaging:

- a **definition phase** in which a VLab Designer specifies the characteristics of a new VLab to serve an application scenario;
- an **approval phase** where the VRE Manager decides whether the specified VLabs can be accepted or rejected. For the accepted VLab, the VRE Manager also decides how this VLab has to be deployed, e.g., which hosting nodes will be exploited;
- a **verification phase** for the VRE Manager to validate a VLab resulting from the approval phase;
- a **management phase** in which the VRE Manager operates on a deployed VLab to customise specific aspects (e.g., the layout of user interfaces constituents a.k.a. portlets) and monitors the operational state of the VRE as a whole.

Further details and information can be found in the gCube Wiki page related to the service at the web address: https://wiki.gcube-system.org/gcube/VRE_Administration.
These services were included in gCube before the beginning of Blue-Cloud. Blue-Cloud developments contribute to their re-design to adapt them to evolving technologies.

[4] https://netflix.github.io/conductor/

3.2.1 VRE Enabling front-end applications

These include a set of interaction-oriented services and front-end applications providing functions to support the VRE Manager in the above-mentioned phases.

These front-end applications (portlets) were available before the Blue-Cloud project start, yet during the project, they have been re-designed to adapt to the evolving technologies. In particular, they are now responsive (capable of displaying on different devices such as smartphones, tablets, and desktop).

3.2.1.1 VRE Definition portlet

The VRE Definition portlet assists the *definition phase*. The procedure is performed by the VRE Designer and leads to the specification of a Virtual Lab Environment, i.e., the selection of the resources and the identification of other characteristics describing the requirements of a VLab devised to serve the needs of a specific demonstrator.

This procedure is supported by a dedicated portlet, the VRE Definition Portlet, that implements a wizard-based approach.

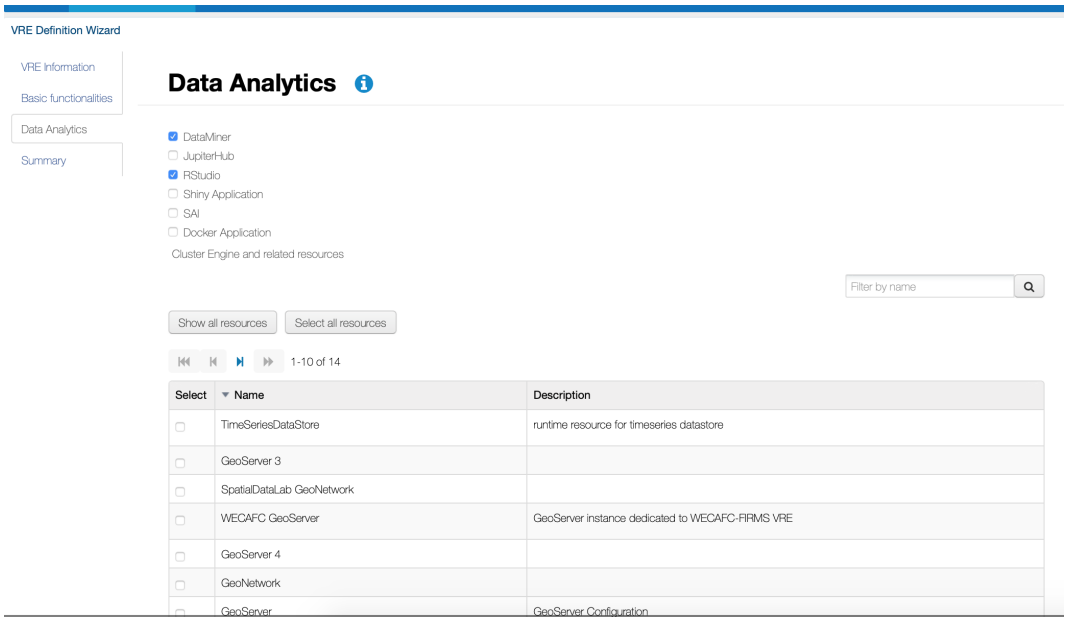

Figure 3. The VRE Definition portlet: the Data Analytics step

Figure 3 shows the selection of the Data Analytics resources. The user of the VRE Designer is provided with descriptive information to select the resources to add to the VLab, e.g., for Data Analytics, the DataMiner Service, the list of Server and Algorithms available for deployment.

3.2.1.2 VRE Manager portlet

The VRE manager portlet (vre-deploy) assists with the *approval and verification phase,* the action after the definition phase, aiming at actually deploying the VLab. To perform this action, the VRE Manager should use the VRE Manager portlet.

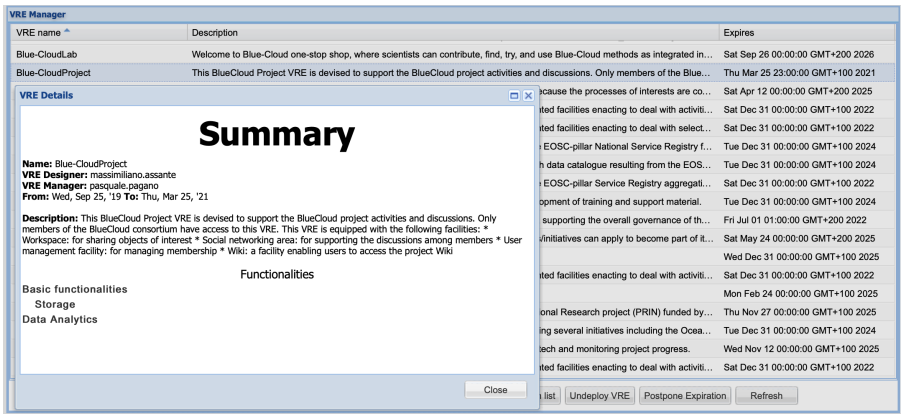

Figure 4. The VRE Manager Portlet: a screenshot

VLabs to be approved are in the "Pending" status. Using the "Action" menu, the VRE Manager can analyse the VRE definition ("View Definition"), edit a VRE definition made by a VRE Designer ("Edit"), start the approval phase ("Approve") or withdraw the VLab designed by a VRE Designer ("Withdraw").

4. Collaborative Framework Components

The Blue Cloud VRE Collaborative Framework includes a set of services and components, enabling their users to collaborate on an activity or a project by sharing material and communicating in smart and flexible ways. In this section, two constituents of this framework are detailed:

- The Workspace component, to organise and share digital materials using an interface resembling a standard file system with items organised in folders.
- The Social Networking, to have discussions and information exchanges using social networking approaches and practices (e.g., post, hashtags, mentions, likes).

4.1 Workspace

The workspace service is a key component of every D4Science-based VRE and VLab. It resembles a typical file system with items organised in folders. Internally, it supports an open-ended set of items that (i) contain rich and extensible metadata and (ii) rely on an array of storage solutions. The Workspace is fully integrated with the rest of the services of a VRE/VLab (Figure 5) to facilitate access to and store new content. It is integrated with most of the analytics components, i.e., Data Miner, the Software and Algorithm Importer, RStudio, and JupyterHub (cf. Sec. 5). It is also linked with the publishing framework to allow the contents of the Workspace to be published via the Data Catalogue (cf. Sec. 6). Moreover, the Workspace allows for sharing its sections, thus facilitating collaborative practices (Assante et al. 2019b).

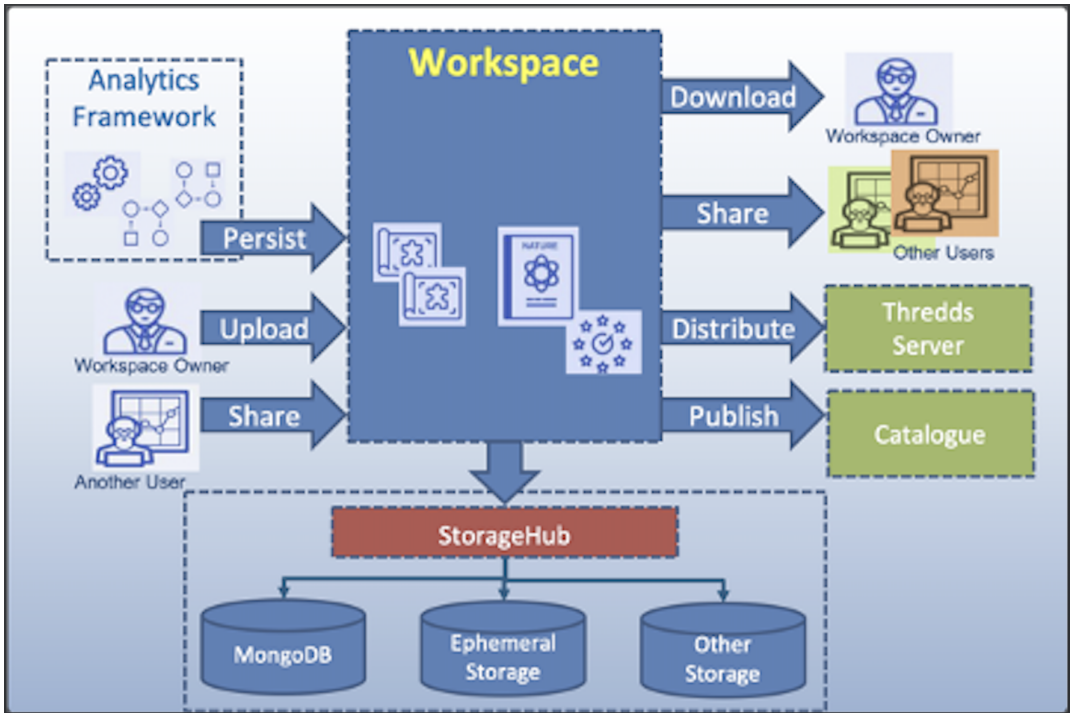

Figure 5. Workspace interactions diagram

The graphical user interface of the Workspace is reported in Figure 6.

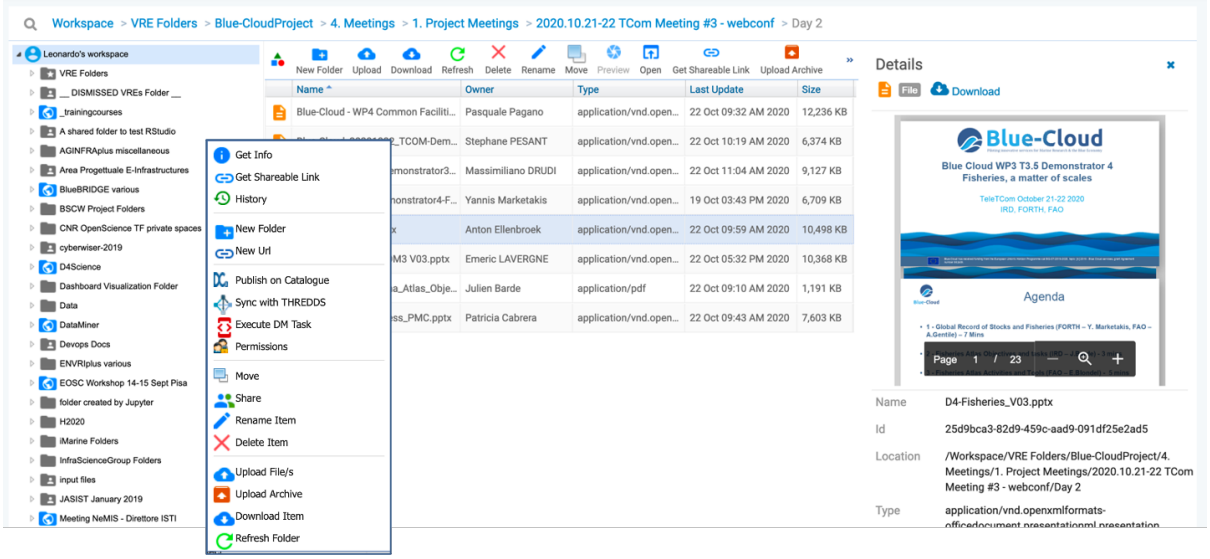

Figure 6. The Workspace graphical user interface

The workspace content is organised in folders. Two types of a folder are supported:

- VRE folder: paired with a VLab and available to all VLab members;
- plain folder created by a user and either accessible only to her/him or shared with other users.

For every workspace element, either a folder or an item, it is possible to see some metadata (e.g., a name, a description, creation, and update dates) and possibly a preview of its content. Moreover, it is possible to create links (as URI's) pointing to it. For folders, it is possible to create links as well enabling those who receive it to access the folder in "guest mode", and no login will be requested.

The contextual menu offers shortcuts allowing the users to perform some actions on it, e.g., to publish the item into the catalogue, to execute a data miner process passing the item as input.

4.2 Social Networking

The social networking component provides its users with facilities for communicating and cooperating by exploiting social networking practices (e.g., rich posts, likes, hashtags, and mentions).

Figure 7 shows the software architecture of the social networking collaborative platform, which relies on the Social Networking Engine, an Apache Cassandra[5] database for storing related data, and on Elasticsearch[6] for the retrieval of data. The Engine exposes its facilities through an HTTP REST Interface and comprises two services: (i) the Social Networking Service to efficiently store and access social networking data (Posts, Comments, Notifications, etc.) in the underlying Cassandra Cluster, and (ii) the Social Networking Indexer Service building on Elasticsearch indices to perform search operations over the social networking data.

[5] https://cassandra.apache.org
[6] https://www.elastic.co/elasticsearch

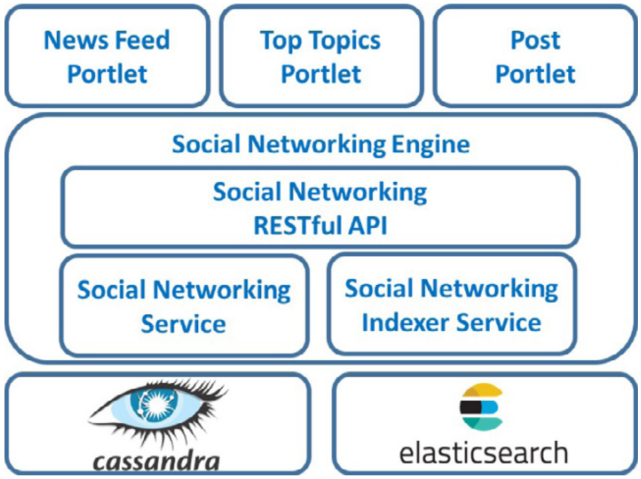

Figure 7. The architecture of the social networking collaborative platform

4.2.1 Service

The service relies on a core Java library called gCube Social Networking Library (SNL). SNL offers methods for posts and comments creation, retrieval and removal, and for managing notifications. The Social Networking Service exposes a subset of the functionalities over an SSL protocol in a standard, reliable and secure way.

The Social Networking Service and the Social Networking Library have been re-designed to adapt to evolving technologies. Detailed information about the service and the library can be found at the URLs below:

- https://wiki.gcube-system.org/gcube/Social_Networking_Library;
- https://wiki.gcube-system.org/gcube/Social_Networking_Service.

4.2.2 Indexer

The Social Networking Indexer Service (also Social Networking Data Discovery service) offers full-text search capabilities over the social networking data in the Blue-Cloud. The Social Networking Indexer Service has been re-designed as well like the Library, to adapt to evolving technologies.

The full-text search is enabled by Elasticsearch, a distributed, RESTful search and analytics engine capable of addressing a growing number of use cases. It is based on the Apache Lucene software library. It runs over one or more cluster nodes and is reachable over the http(s) protocol. ElasticSearch allows the organisation of documents in one or more indices/types according to their schema, which can be defined in JSON format.

The main goal of the Social Networking Indexer Service is to let users quickly search over Blue Cloud's (potentially huge) amount of data, taking into account their profile (e.g., a user is allowed to search only the data of the VLabs in which she is registered). In order to do that, a Java client library, the social-data-indexing-common, receives the query submitted from the users and returns the list of posts belonging to the user's VLabs, if any, sorted according to a score; and a Java gCube

Smart Executor Plugin, which triggers the indexing of data on the service when necessary, social-data-indexer-se-plugin.

Detailed information about the service and the library can be found at the URLs below:

- https://gcube.wiki.gcube-system.org/gcube/Social_Networking_Data_Discovery

4.2.3 User Interfaces

Figure 8 shows the user interface of the social networking area. Starting from the top right, we can see the “Share Updates” Portlet (Post Portlet in the Architecture Figure) allowing members to post. Just below it, the News Feed portlet collects the posts and shows them in reverse chronological order allowing members to comment on them. On the top left, the users can find some statistics on the usage of the platform, the Workspace shared folder content of the VLab, and the Top trending topics, showing them using the prefixed hashtag. This is indeed the area VLab users rely on to communicate with their VLab co-workers and be informed on others' achievements, discussions, and opinions. It resembles a social networking environment with posts, tags, mentions, comments, and reactions, yet its integration with the rest makes it a powerful and flexible communication channel for researchers.

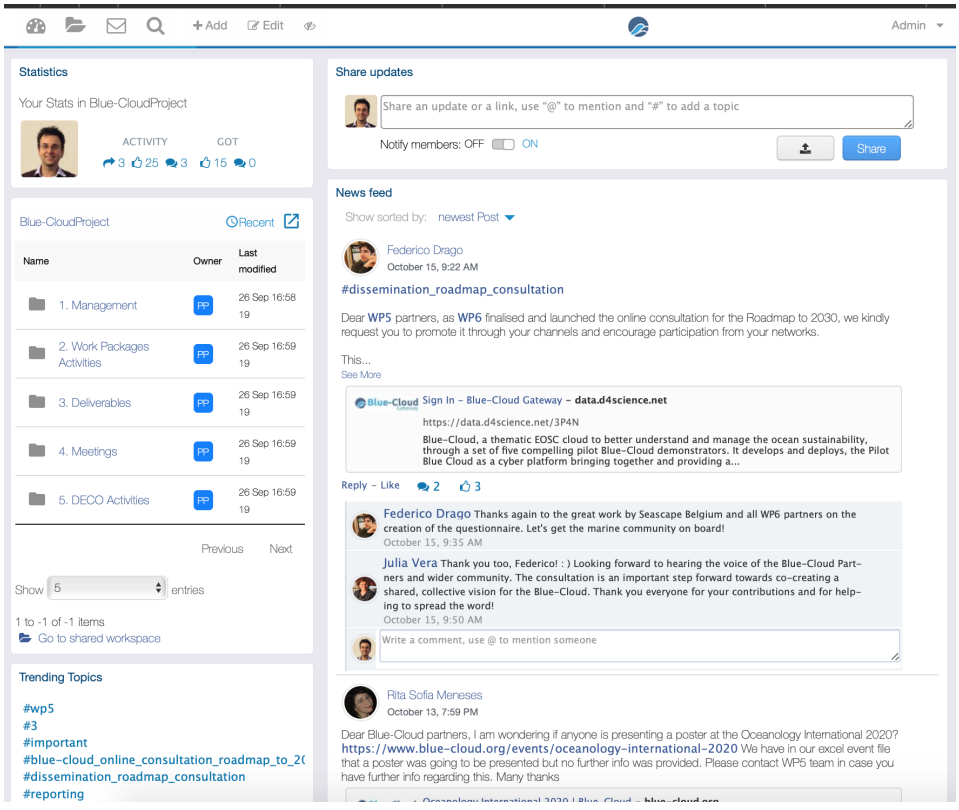

Figure 8. The social networking user interface: a screenshot

5. Analytics Framework Components

The Analytics Framework includes a set of services and components for performing data processing and mining on information sets.

As presented in D2.6, the Blue Cloud VRE Analytics Framework was designed to include the DataMiner System, the Software and Algorithms Importer, and the Smart Executor System. This set of services is complemented by integrating RStudio and by supporting the integration of RShiny applications. To meet the Blue community requirements gathered through the Blue-Cloud demonstrators, the Blue-Cloud VRE extended this set of systems, components, and tools by adding (a) support for dynamic and interactive notebooks via JupyterHub, and (b) the support for community-specific applications delivered as a Docker container.

These enhanced capabilities of the Analytics Framework allow users to perform science by selecting the most appropriate and familiar tool to implement the analytical methods and data processing. Users' analytical methods can be developed by almost any programming language (R, Java, Python, Fortran, Octave, etc.). Moreover, users can select from several integration patterns of the Blue Cloud VRE.

The Blue-Cloud Lab is set-up as a typical VLab that provides access to the overall components of the Analytics Framework in a user-friendly and easy-to-use way.

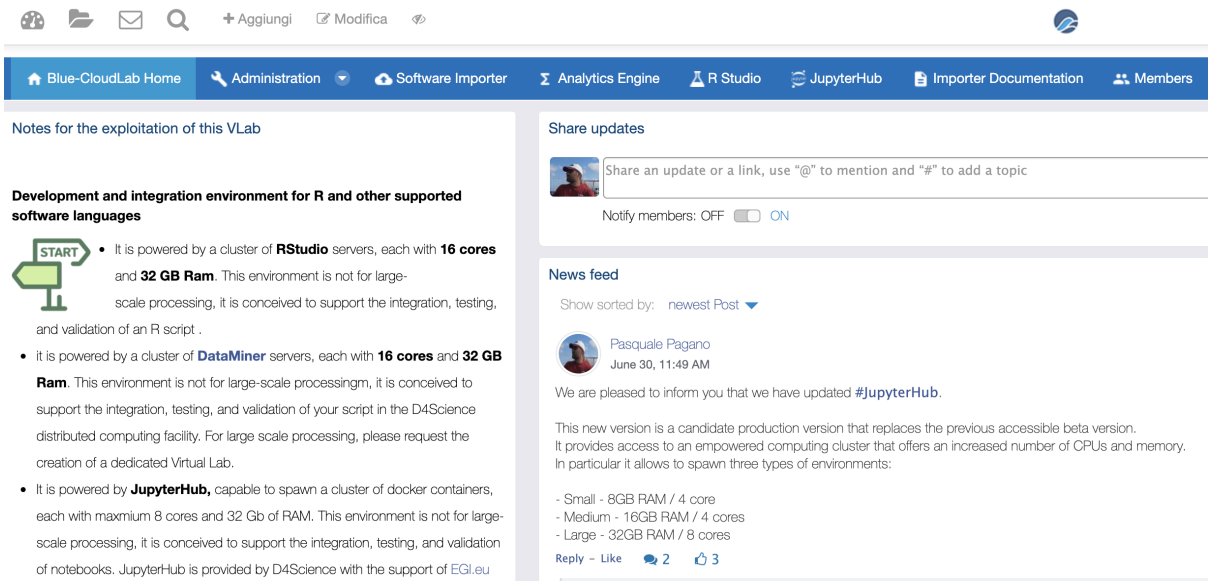

Figure 9. Virtual Laboratory typical graphical user interface

DataMiner, RStudio, and JupyterHub are all integrated with the Blue Cloud storage allowing to download, upload, remove, add and list files, define access rights to files, and enabling private, public, or shared (group-based) access.

For entire applications, Docker containers (e.g., with an RShiny application) can be deployed in different ways:

- If a public container is already available in Docker Hub or any other public container registry, it is sufficient to report the coordinates of that container;
- If a public container is not yet available:
    - a build of a public image can be requested. It must be accessible from the D4Science Jenkins instance so the process can be automated. The resulting container image will be uploaded into Docker Hub and deployed into the cluster;
    - a build of a private image can also be requested. Also, in this case, it must be accessible from the D4Science Jenkins instance so that the process can be automated. The resulting container image will be uploaded into the D4Science's private registry and deployed into the cluster.

The Blue Cloud VRE Analytics Framework provides:

- support for the execution of analytical methods on multi-core computational nodes (DataMiner, Smart Executor, RStudio, JupyterHub, and Docker);
- support for multi-tenancy and concurrent access (DataMiner, Smart Executor, RStudio, JupyterHub, and Docker);
- support for Auditing (DataMiner)
- automatic distribution of the execution of analytical methods on sets of computing nodes (DataMiner, Docker Swarm);
- automatically transfers control to a duplicate computational node when faults or failures are detected (DataMiner, Docker Swarm);
- support for scheduled and repeated execution (Smart Executor);
- automatic generation of provenance information to enable reproducibility and repeatability of the computed results (DataMiner);
- support for the standard Web Processing Service (WPS) protocol (DataMiner).

5.1 Software Importer and DataMiner

The Software and Algorithms Importer (SAI) is an interface allowing any user to easily and quickly import scripts into DataMiner, which in turn, publishes these scripts as-a-Service and manages multi-tenancy and concurrency. Additionally, it allows scientists to update their scripts without following long software re-deploying procedures each time. In summary, SAI produces processes that run on the Blue-Cloud VRE Cloud computing platform and are accessible via the WPS standard.

In order to import a script, three main steps are required:

1. indicate Input, Output, and types of the main script orchestrating the process;
2. create the Software: this operation packages the script and prepares it for execution on the computing platform. It has to be used each time either a change to the interface (I/O) or the required dependencies are needed;
3. publish the Software: this operation enables the execution of the script on the computing platform in the context of the virtual laboratory where it has been imported.

Additionally, the Repackage function can be used to upgrade a published script that has been evolved without changing neither the I/O nor its dependencies.

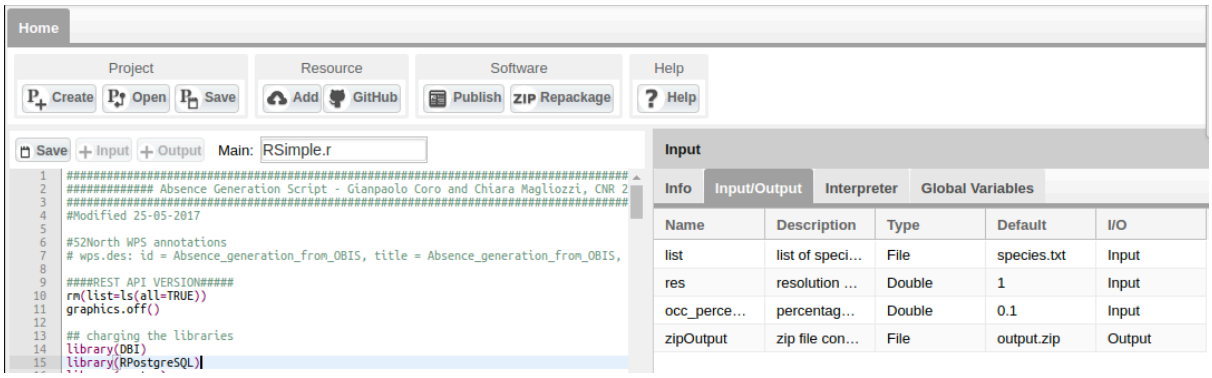

Figure 10. Blue-Cloud VRE Software and Algorithms Importer Interface

Once imported, scripts become exploitable through the DataMiner component.

It offers a Web GUI organised in three main areas: Data Space, Execution Space, Computations Space.

The Data Space allows for accessing, reusing, and downloading the set of input and output datasets used and generated in one or more execution. Data Space items are enriched with business metadata reporting the computational method used for their generation, its execution environment (including the input parameters), the virtual laboratory where it was generated, and the date of generation.

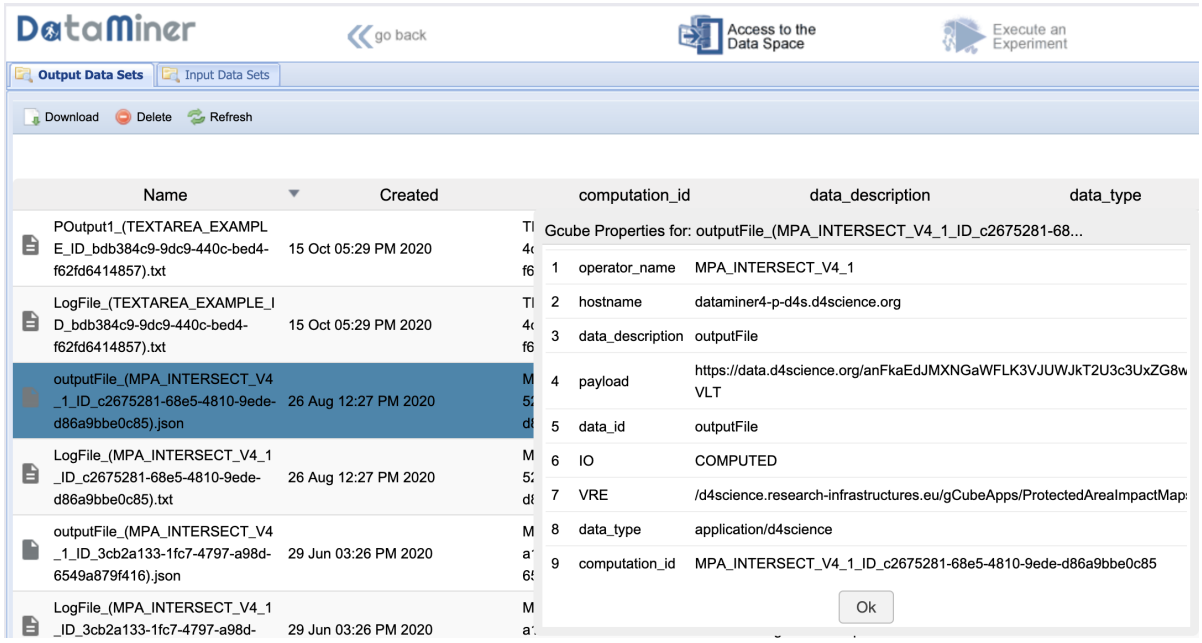

Figure 11. Blue-Cloud VRE Analytics Data Space Interface

The Execution Space presents two panels:

- on the left panel, the GUI presents the list of computational methods available in the virtual laboratory, which are semantically categorised (the category is indicated through SAI). For each method, the interface calls the WPS Describe Process operation to get the descriptions of the inputs and outputs;

- on the right panel, the GUI shows a form enabling to specify the input parameters of the selected computational method. Input data can be selected from the Workspace clearly.

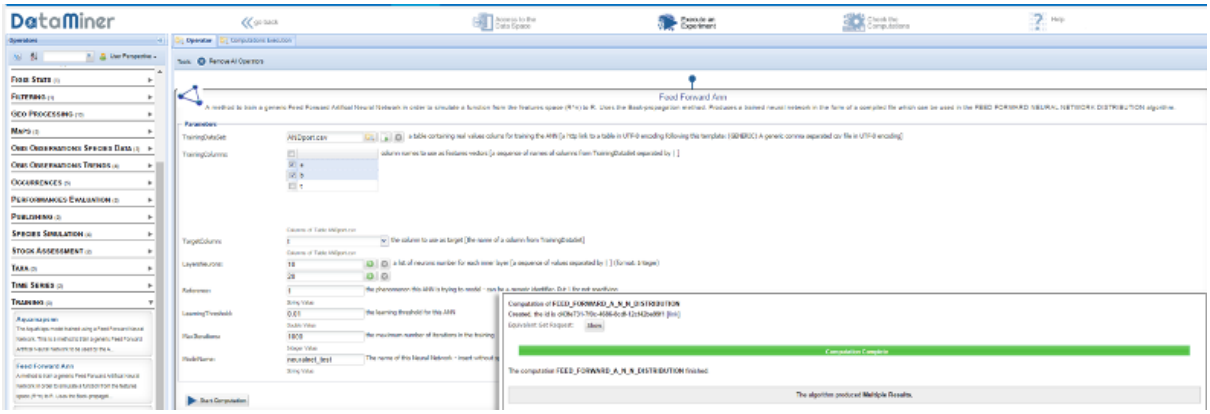

Figure 12. Blue-Cloud VRE Execution Space Interface

The Computations Space represents an important added-value since it reports a summary sheet of the provenance of the execution, either performed by the user or shared with him. From this same space, the computation can also be re-submitted. In this case, the Prov-O XML information associated with the computation is used to rebuild the computation request with the same parameters, and the execution can be re-submitted.

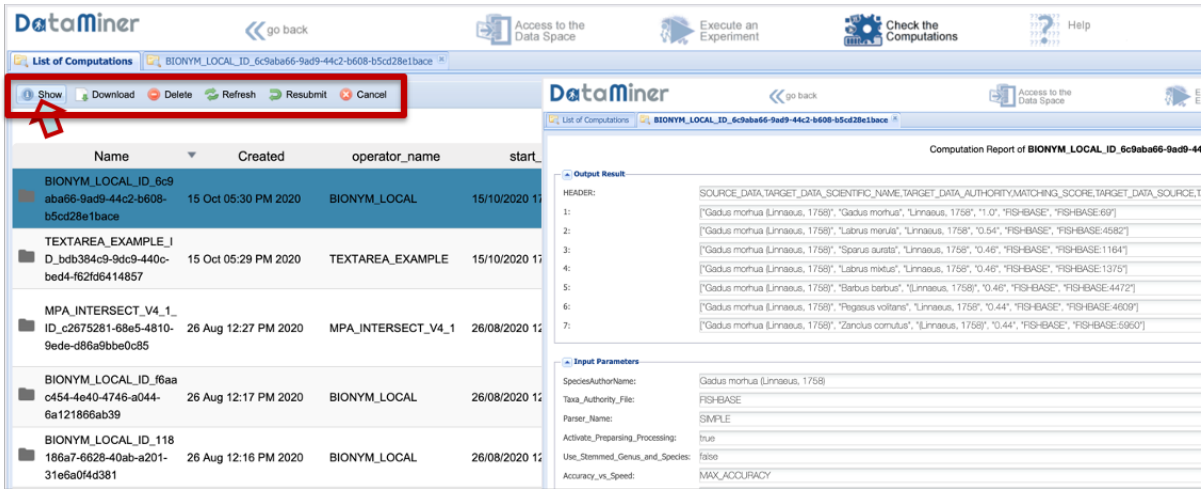

Figure 13. Blue-Cloud VRE Computation Space Interface

5.2 Smart Executor

The SmartExecutor service allows the users to execute tasks and monitor their status. Any task can be either scheduled, or repeated periodically, or activated upon request. A task has to be implemented as a plugin of the service, while its usage and exploitation can be performed using the SmartExecutor REST API.

It is typically used to periodically invoke a computational method imported into the DataMiner. A common case is the monthly aggregation of raw data that is gathered daily. In this example, the data aggregation is implemented via a computational method imported in DataMiner; the monthly execution of it is instead realized by a task of the Smart Executor service. The combination of the

two services will allow either a single user or all the virtual laboratory users to access the collection of aggregated data through the Workspace. It is a useful pattern to manage confidential data.

5.3 RStudio

The RStudio allows its users to perform online statistical analyses with R.

The Blue-Cloud VRE makes the RStudio Application accessible on the D4Science infrastructure ensuring its operation and orchestration in a cluster. The cluster is composed of multiple hosts, each assigned exclusively to a user for an entire online session. At the end of the session, all the content stored in that host will be removed by the D4Science Infrastructure. All data, scripts, and other resources are thus secured and not made accessible to others. The user can persist the R-session into the private Workspace that is accessible through the RStudio Application.

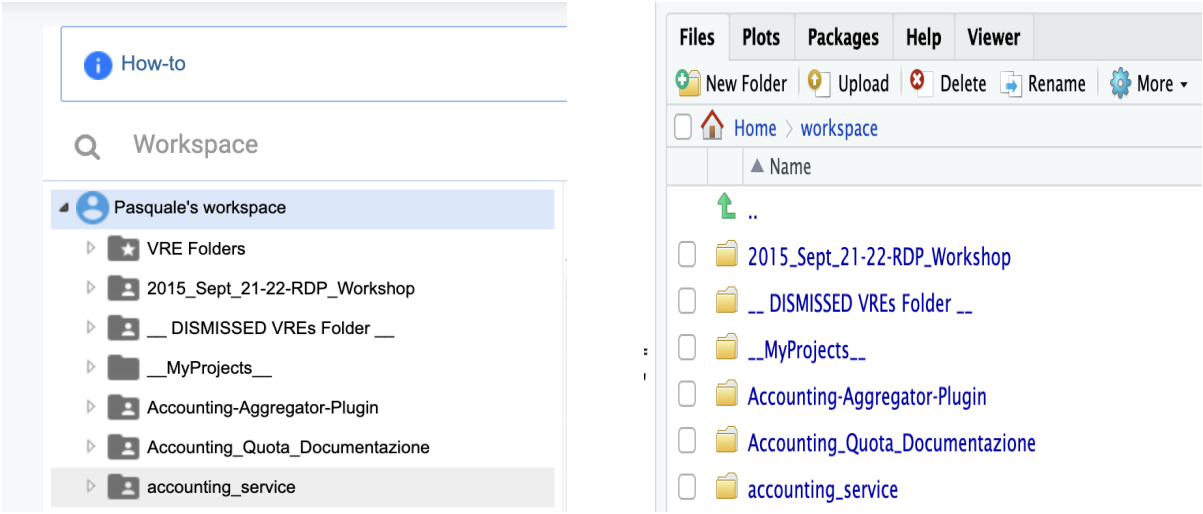

Figure 14. Blue-Cloud VRE Workspace and RStudio Workspace

5.4 JupyterHub

JupyterHub is a web-based interactive development environment 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 Blue-Cloud VRE makes the JupyterHub accessible on the D4Science infrastructure ensuring its operation and orchestration in a cluster, which is made by multiple hosts assigned exclusively to a user for an entire online session. Currently, the cluster supports 80 users (at 8 GB RAM; Cores per user: 4/8; Memory per user (GB): 8/16/32; Storage per user (GB): 10GB; Allocation type: pledged).

JupyterHub uses ephemeral storage. The content locally stored is removed at the end of each session. All the data, scripts, and other resources that the user needs to persist must be stored in the Workspace that is accessible through JupyterHub.

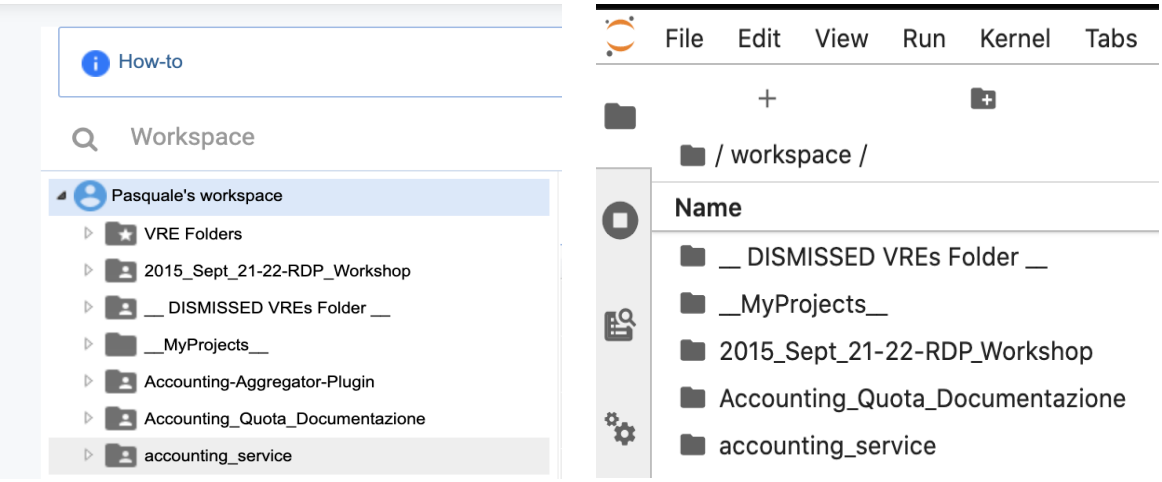

Figure 15. Blue-Cloud VRE Workspace and JupyterHub Workspace

5.5 ShinyProxy and Docker

ShinyProxy allows deploying Shiny apps in an enterprise context with no limitations on their usage. Shiny is an R package that makes it easy to build interactive web apps straight from R. A Shiny app can be embedded in R Markdown documents or used to build dashboards by combining the computational power of R with the interactivity of the modern web.

When deploying a Shiny app with ShinyProxy, the application is simply bundled as an R package and installed into a Docker image. Every time a user runs an application, a container spins up and serves the application. This has numerous advantages:

- fully isolated environment per session;
- plug and play different docker images (even with different R versions or different Shiny versions);
- control on memory and CPU usage via the Docker API;
- monitoring and debugging using standard Docker tooling.

The Blue-Cloud VRE uses Docker Swarm to run multiple Docker containers across a cluster of virtual machines. Docker Swarm defines a manager container that runs on a virtual machine to deploy containers to the various agents and report on their status and deployment information for the cluster. The manager is the primary interface in Docker. Agents are "docker machines" running on virtual machines that register themselves with the manager and run Docker containers. When the client sends a request to the manager to start a container, the manager finds an available agent to run it. It uses a least-utilized algorithm, ensuring the agent running the least number of containers will take charge of the newly requested container.

Routing external traffic into the cluster, load balancing across replicas, and DNS service discovery are a few capabilities that require an additional layer. The Blue-Cloud VRE exploits the HAProxy load balancer to accomplish those capabilities.

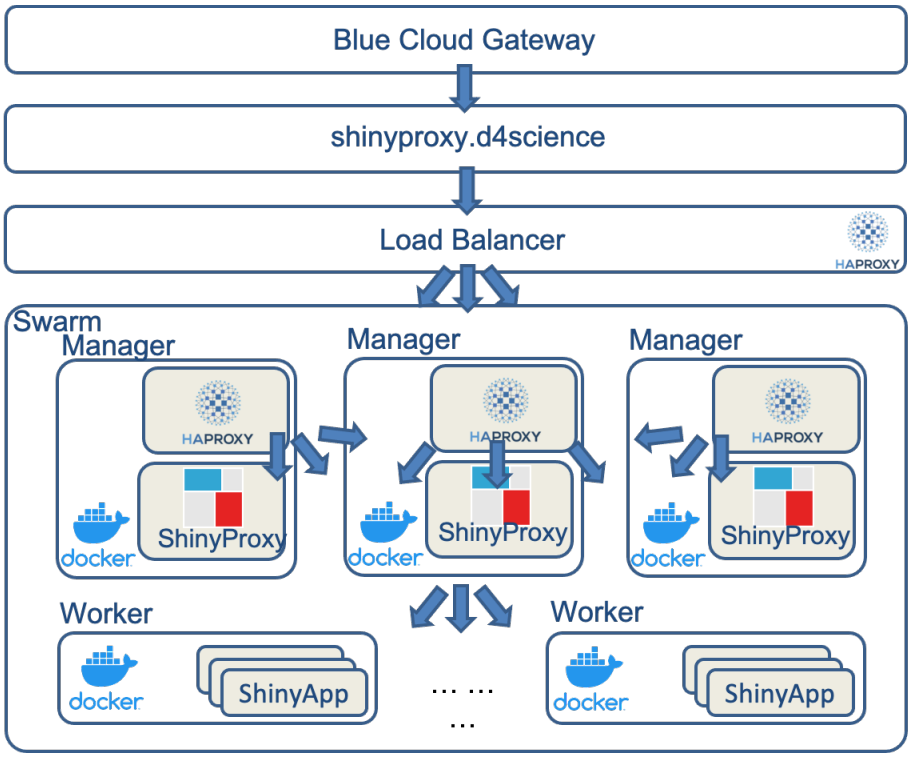

Figure 16. Blue-Cloud VRE cluster supporting Shiny and any other Docker app

There are three ways to exploit HAProxy:

- one HAProxy container: Swarm's ingress routing mesh forward clients' requests to it;
- one HAProxy container: HAProxy receives clients' requests directly, without using the ingress routing mesh;
- create a replica of HAProxy on each node: each will receive clients' requests directly.

The Blue-Cloud VRE adopted the third approach guaranteeing the capacity to handle more requests as there are more HAProxy running instances. In order to make this deployment efficient, the VRE uses an external L4 load balancer, still benefitting from an HAProxy in TCP mode, in front of the Swarm cluster. This allows the system to spread the load across the different HAProxy containers.

This architecture is used to deploy also any other application delivered as a Docker container.

5.6 Docker and DataMiner

A Docker image represents an easy-way to deliver software in packages called *containers*, which are isolated from each another and bundle their own software, libraries, and configuration files; thus, they may contribute to simplifying the configuration and operation of the Blue-Cloud VRE infrastructure.

The Blue-Cloud VRE offers the Docker Image Executor algorithm, which allows its users to retrieve and run an image (from a public Docker repository, e.g., DockerHub) in a Swarm cluster while guaranteeing replicability, reusability, sharing, and accounting of the Docker image. Each execution will be accounted for and presented in the Blue-Cloud VRE Computation Space Interface. A screenshot of the Docker Image Executor method is presented in Figure 17.

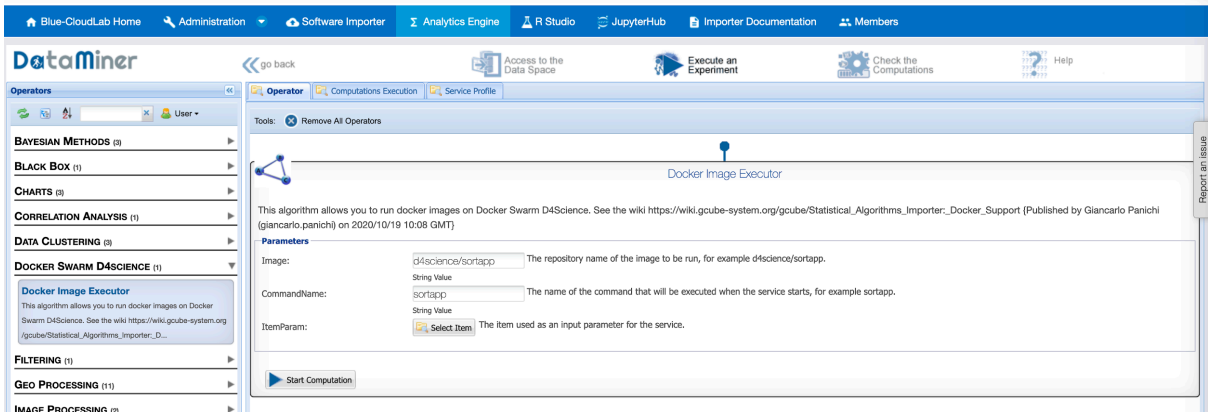

Figure 17. The DataMiner Docker Image Executor Algorithm

6. Publishing Framework Components

The Publishing Framework includes a set of services and components enabling their users to document and make “public” (made available online) any artifact.

Its primary component is the VRE Data Catalogue that, according to D2.6, is expected to be populated with contents gathered through the Blue-Cloud discovery and access service (see Figure 18). However, the data catalogue is a service on its own that is a key component of D4Science-based VREs and VLabs for enacting open science practices and FAIR data management (Assante et al. 2019a,2019b).

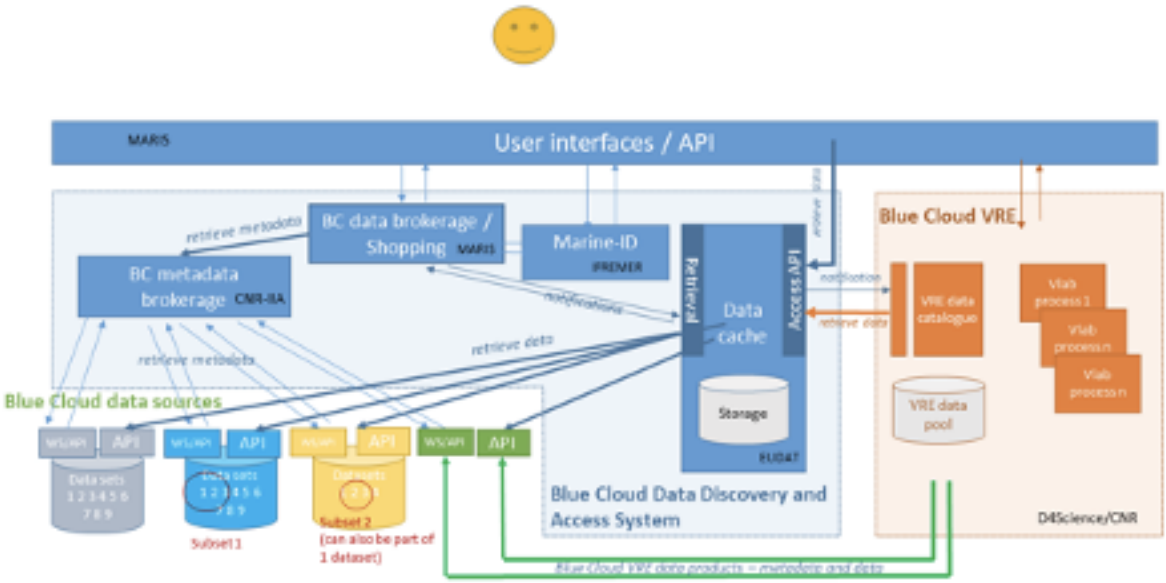

Figure 18. VRE Data Catalogue and Blue Cloud discovery and access service

6.1 The VRE Data Catalogue Service

The VRE Data Catalogue service is a catalogue service built on open-source technology for data catalogues (CKAN ckan.org), but extended to (a) be integrated with D4Science services and (b) support a rich, community-defined, and extensible set of catalogue item typologies.

The architecture of the service is depicted in Figure 19.

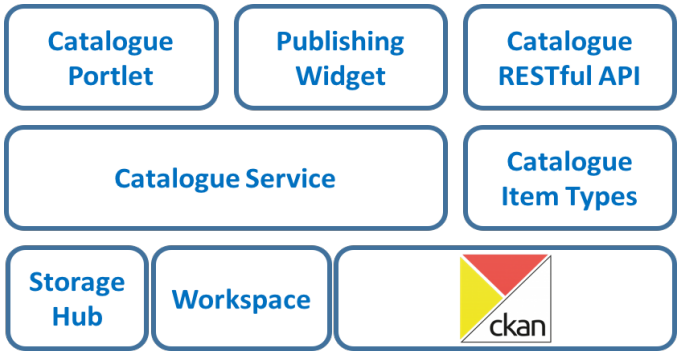

Figure 19. Catalogue Service Architecture

The Catalogue Service is the core component enabling to implement the business logic of the overall service. The catalogue service interacts with a component supporting the creation of catalogue item typologies, i.e., specifications characterising items in terms of attributes, controlled vocabulary, etc. The data model supported by the catalogue is shown in Figure 20.

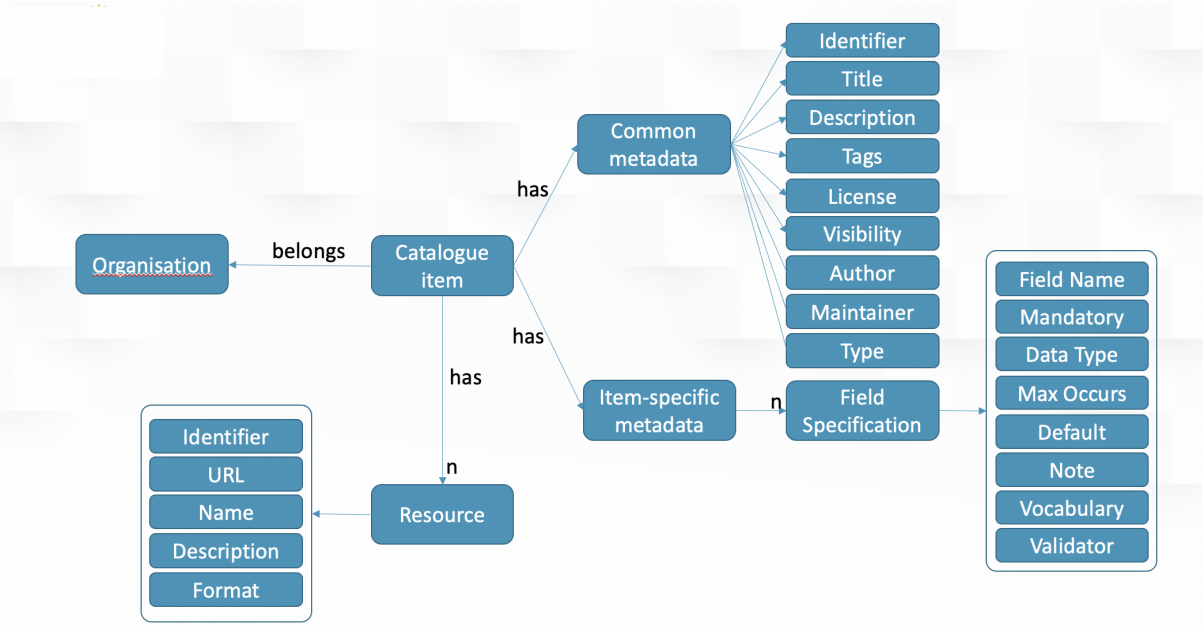

Figure 20. Catalogue Service data model

According to this data model:

- every catalogue item belongs to one (and only) ***organisation*** representing the context/authority responsible for the publishing of the item. Organisations are usually paired with VLabs for the items stemming from them or with other existing contexts where the items pre-exist VLabs;
- every catalogue item is characterised by a set of ***common metadata,*** including a unique identifier, a title, a description, a list of tags (e.g., keywords, subjects), a licence, visibility (whether the item is publicly available or visible only to the members of a VLab), an author, a maintainer, and a type;
- every catalogue item may have a list of ***item-specific metadata*** depending on the type. In practice, this is a list of fields, each having a name, a mandatory directive (whether the field is mandatory or optional), a type (e.g., string, number, spatial extent), a max occur directive (whether the field can be instantiated one time only or many times), a default value, a descriptive note helping to understand the intended meaning of the field, a controlled vocabulary (if any) of allowed values to use to compile the field, and a validator (if any) to check the inserted value adherence to specific validation rules;
- every catalogue item can be equipped with a set of ***resources***, i.e., catalogue item constituents representing the real payload of the item or part of it. Resources are characterised by their identifier, a name, a description, a format, and, most importantly, the URL pointing to the content.

There are two main routes to populate the catalogue with contents: by harvesting from existing data sources and by explicitly publishing via the Web GUI or the Web API (a.k.a. the gCat REST Service). Figure 21 clarifies the approach. The catalogue builds upon standards including DCAT, CSW, OAI-PMH as well as CKAN standard APIs.

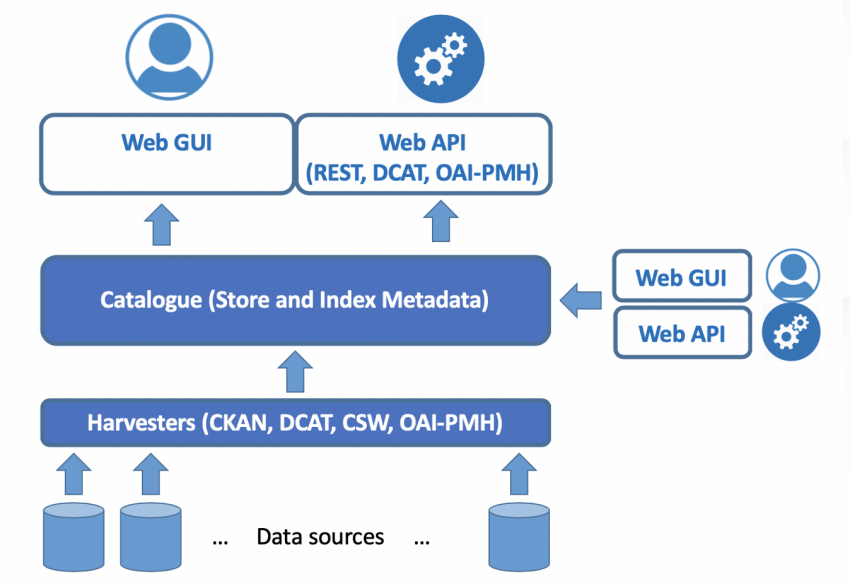

Figure 21. Catalogue Service: Feeding and Consumption options

For what regards the consumption options, the catalogue offers both (i) programmatic access for "machines" to get contents via a proprietary REST API (a.k.a. the gCat REST Service) and standards like DCAT and OAI-PMH, and (ii) human-oriented access via a dedicated Web-based GUI to search the catalogue contents via queries, to browse them via a set of options, and to access and visualize every item.

7. Release Management: software continuous integration and delivery

The components discussed so far are part of the gCube software system and other open-source systems as RStudio, JupyterHub, ShinyProxy, Docker, HAProxy, and many others.

gCube (Assante et al. 2019a)(www.gcube-system.org) is an open-source software system designed and implemented to enable the building and operation of a Service-Oriented Infrastructure supporting the definition of Virtual Research Environments (VREs). It is exploited by the D4Science infrastructure, and its components are widely used in several application frameworks.

It covers a rich array of "mediators" for the integration and exploitation of services and facilities offered by other tools and systems like the ones listed above. This system has been implemented with the support of other initiatives funded by the European Commission (see https://www.gcube-system.org/about for the complete list of projects).

The enabling technologies selected to properly support the continuous integration process in gCube are Gitea (https://gitea.io/ as Git hosting service), Jenkins (https://www.jenkins.io/ as automation server) and Maven (https://maven.apache.org/ as project management and comprehension tool).

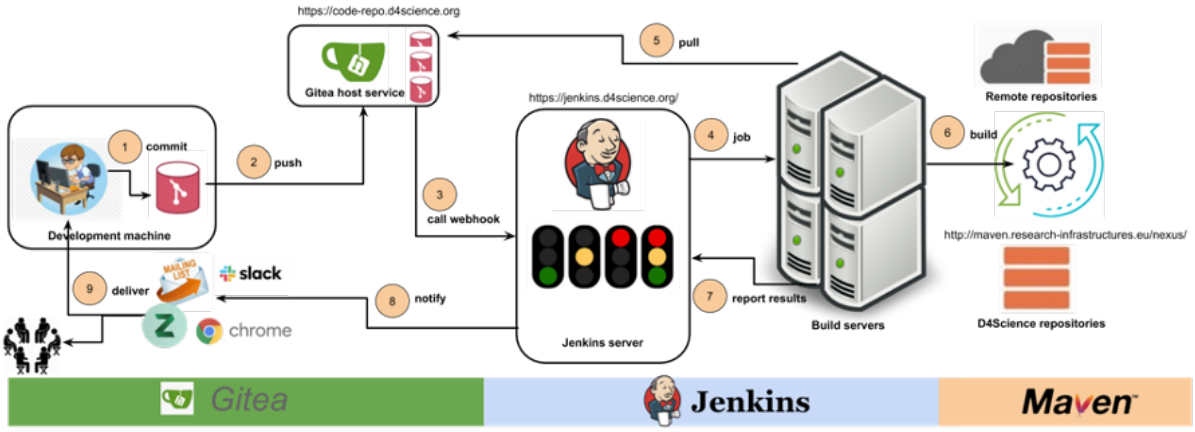

Figure 22. gCube continuous integration and delivery workflow

Their proper configuration and interactions are the foundation for automating the non-human part of the software development process, with continuous integration and delivery.

The workflow involves interaction with the Git repositories (managed by the D4Science Gitea instance https://code-repo.d4science.org/) and benefits from the Maven project management and comprehension tool. Maven is based on the concept of a Project Object Model (POM). Maven can manage a project's build, reporting, and documentation from a central piece of information. Every gCube component is described by a POM. Thanks to POM component definition, Maven, among other things, allows to:

- easy the build process;
- provide a uniform build system;
- generate quality information;

- provide guidelines for best practices development;
- migrate new features transparently.

7.1 Version Control System

A Version Control System is a software keeping track of every modification to the files belonging to a repository. VCSs are especially used to keep track of software source code. The gCube software system adopts Git as its code management system.

A complete code hosting solution for Git, based on Gitea (https://gitea.io/), has been deployed on the D4Science infrastructure (https://code-repo.d4science.org/) to properly manage the Git repositories and its settings, and to finally enable the creation of new components.

gCube is currently organized in 274 repositories in Gitea managed by 16 developers: https://code-repo.d4science.org/gCubeSystem

7.2 Build Tool

A build tool automates everything related to building a software project. The advantage of making the build process automatic is to minimize the risk of human errors. Additionally, an automated build tool is typically faster than a human performing the same steps manually. gCube adopts Maven as its build tool.

Building a gCube software component typically implies performing the following activities:

- verifying the quality of the source;
- generating documentation from the source code;
- compiling source code;
- packaging compiled code into JAR, WAR, or ZIP files;
- installing the packaged code on a server, in the dedicated software repository.

Such activities are automated by plugging them inside the Maven build process.

Maven requires a dependency repository to work properly. The D4Science infrastructure makes use of an instance the Nexus technology (https://www.sonatype.com/nexus-repository-oss) - a free artefact repository with universal support for many formats, which is compatible with Maven.

The D4Science Nexus instance (http://nexus.d4science.org/nexus/content/repositories) is configured to host the following repositories:

- gcube-snapshots: for artifacts under development;
- gcube-staging: for artifacts to be deployed on the D4Science production infrastructure;
- gcube-releases: for artifacts deployed on the D4Science production infrastructure.

7.3 Continuous Integration

Continuous integration includes all the steps to create, remove, and modify and monitor a software component, which must be integrated and deployed in the D4Science infrastructure.

Jenkins enables to create a workflow to integrate a component, which is often referred to as a pipeline. A Jenkins instance has been deployed in the D4science infrastructure, and it is available at https://jenkins.d4science.org/.

One of the biggest risks of a non-properly configured Continuous Integration pipeline is to trigger too many builds on Jenkins and transform the pipeline into something we can call Continuous Building. Multiple builds of the same project simultaneously on the same slave worker can interfere with each other, and inconsistent situations are very common. For instance, this is the case when each commit is immediate to the master branch or when the Git repository is wrongly used as a backup system. To manage this issue, gCube implements the following rules and practices for any Jenkins project:

- a Jenkins project always builds the master branch;
- if a Jenkins project is configured to build a second branch, this branch must generate a software artifact with a different version than master branch;
- the development of new features and bug fixing are done in dedicated branches (task branching);
- merges into master branch are performed when the feature/fix is stable;
- commits are not always pushed to the remote repository, but only when they add a significant, self-contained, and consistent piece of working code;
- the master branch MUST always be in a releasable state

By adopting this set of rules and practices, builds are triggered only when a stable feature is merged into the master, while commits in the other branches do not involve Jenkins. If two branches are built at the same time in a Jenkins project (which should be temporary), their different versions guarantee that there are no conflicts in the published artifacts.

The history of builds executed on Jenkins can be accessed at:

https://jenkins.d4science.org/view/all/builds

7.4 Continuous Delivery

Continuous Delivery enables producing software faster rather than in big release cycles over weeks or months so that a stream of potential production-grade versions of gCube artifacts is available at any time and ready to be released.

Delivery builds publish artefacts as software versions. Typically, projects have many integration builds for verifications, validation, etc. but artifacts are published only regularly or manually during a formal release cycle.

gCube applies the following principles to Continuous Delivery:

- a regular build is not different from a release build. What makes a release special is how we see it;
- any human intervention should be required. All decisions can be calculated automatically;
- if you have multiple branches for some reasons, have a dedicated build job for each one of them in order to see and manage the current branch status easily;

- branch builds must enforce building from the top, never to be parameterized for building custom changesets;
- avoid to release branches;
- avoid having continuous delivery before doing code reviews enforced by the building system;
- block artifact deployments except for the build user;
- make it possible to start everything completely from scratch;
- do not have any snapshot dependency during releases;
- do not use version ranges in dependencies because it prevents reproducible builds;
- keep most logic inside Maven and Git to keep it reusable everywhere, without the need for a build server.

gCube uses the declarative pipelines of Jenkins to define the build flow. The pipelines allow us to use basic but common building blocks to define the whole build/release process.

Jenkins Pipeline (https://jenkins.io/doc/book/pipeline/) is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins. A pipeline has an extensible automation server for creating simple or complex delivery pipelines "as code" via pipeline DSL (Domain-specific Language).

In gCube, we use a Pipeline to trigger the builds of jobs, forming a gCube Release. The pipeline project is available at https://jenkins.d4science.org/job/gCube-Release-Jobs/. The Pipeline project can be launched in 4 different ways (Type parameter):

- SNAPSHOT-DRY-RUN (default): build snapshot artifacts, install the artifacts in a local repo, do not deploy;
- SNAPSHOT: build snapshot artifacts, install the artifacts in a local repo, deploy the artifacts to the gcube-snapshots Maven Repository;
- RELEASE-DRY-RUN: build release artifacts, install the artifacts in a local repo, do not deploy
- RELEASE: build snapshot artifacts, install the artifacts in a local repo, deploy the artifacts to the gcube-releases Maven Repository

7.5 Software Release

The gCube developers must follow all the procedures and rules adopted by the gCube system. They must guarantee that the master branch is releasable at any time.

When the Release Manager declares a gCube release open, and until it is declared closed, the artefact version in the POM on master must not have the -SNAPSHOT suffix.

When the Release Manager declares the gCube release close, and if the release includes a new version of a component, the developer must tag the master branch.

The Release Manager can test the full pipeline execution with the DRY-RUN builds. Once all the projects in the build set work, the SNAPSHOT or RELEASE build can be launched to effectively deploy the artefacts on the remote Maven Repository.

The following documentation is available for each release:

- released on: the date in which the release was made publicly available;
- Build Configuration: the configuration used to build the release;
- Build Report: this report lists all the released components with their version, location, and commit ID on the source control system;
- Tag Report: this report is the fingerprint of the release with the commit IDs and tags
- Release Notes: the aggregated release notes reporting all the major new features and fixed issues for each released component.

The releases, including the enhancements and extensions reported in this deliverable, can be easily accessed at https://code-repo.d4science.org/gCubeCI/gCubeReleases.

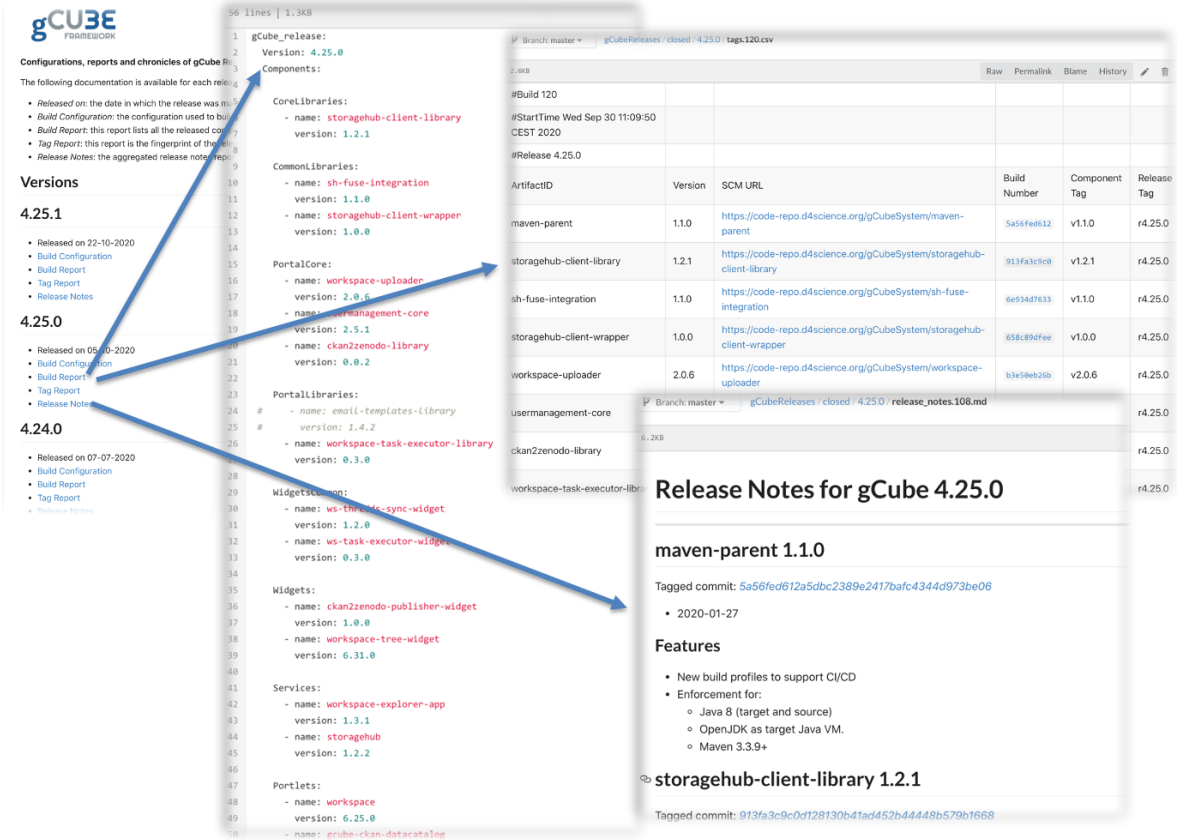

Figure 23. gCube Release documentation

8. Conclusion

The deliverable documents the revised and extended VRE Architecture of Blue-Cloud and its constituents. In particular, it described the revised version of the overall architecture (cf. Sec 2) and detailed (a) the two services contributing to the Enabling Framework part, namely the Identity and Access Management solution and the VRE Management solution; (b) the two services contributing to the Collaborative Framework part, i.e., the Workspace and the Social Networking service; (c) the six services contributing to the Analytics Framework, namely the Software and Algorithm Importer, the Smart Executor, the RStudio-based solution, the JupyterHub-based solution, the ShinyProxy-based solution for ShinyApps, and the DataMiner-based solution for DockerApps; (d) the service contributing to the Publishing Framework, i.e., the Data Catalogue. The process and technologies enabling the continuous release of these components and their integration into the gCube technology are also documented.

The 11 components presented in the deliverable have been included and released by the following 11 gCube open-source software releases: 4.16 (Nov. 2019), 4.17 (Dec. 2019), 4.18 (Dec. 2019), 4.19 (Feb. 2020), 4.20 (Feb. 2020), 4.21 (Mar. 2020), 4.22 (May 2020), 4.23 (Jun. 2020), 4.24 (Jul. 2020), 4.25 (Oct. 2020), 4.25.1 (Oct. 2020). Moreover, they are in the pipelines producing the forthcoming releases. They are exploited to update, develop, and operate the Blue Cloud gateway (https://blue-cloud.d4science.org/home) and the underlying infrastructure and services. At the time of this deliverable (November 2020), the gateway hosts a total of 8 VREs and VLabs, including five specifically conceived to support the co-development of some of the Blue-Cloud demonstrators (namely, the Aquaculture Atlas Generation for Demonstrator #5, the Blue-Cloud Lab for several demonstrators, the GRSF pre for Demonstrator #4, the Marine Environmental Indicators for Demonstrator #3, the Zoo-Phytoplankton EOV for Demonstrator #1). This gateway and its tools are serving more than 400 users that (since January 2020) performed a total of more than 5000 working sessions, more than 1700 accesses to the Workspace, more than 750 analytics tasks. These exploitation and uptake indicators are destined to grow in the coming months thanks to data updates and continued use, and the further development of existing VLabs, and finally, the creation of new ones.

References

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 (2019a) ***The gCube system: Delivering Virtual Research Environments as-a-Service***. Future Gener. Comput. Syst. 95: 445-453 10.1016/j.future.2018.10.035

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

D. M.A. Schaap, P. Thijsse, P. Pagano, M. Assante, E. Boldrini, M. Buurman, M. D'Antonio, C. Ariyo, G. Maudire, C. Nys (2020) ***Blue Cloud Architecture (Release 1)***. D2.6 Blue-Cloud Deliverable, July 2020