D2.6 Blue Cloud Architecture (Release 1)

Work Package	WP2, developing the Blue Cloud discovery and access service and overall Blue Cloud architecture
Lead Partner	MARIS
Lead Author (Org)	MARIS
Contributing Author(s)	Dick M.A. Schaap (MARIS), Peter Thijsse (MARIS), Pasquale Pagano (CNR-ISTI), Massimiliano Assante (CNR-ISTI), Enrico Boldrini (CNR-IIA), Merret Buurman (DKRZ), Mattia d'Antonio (CINECA), Chris Ariyo (CSC), Gilbert Maudire (IFREMER), Cecile Nys (IFREMER)
Reviewers	Sara Garavelli (TRUST IT), Blue-Cloud Technical Committee (TCOM)
Due Date	30.06.2020, M9 [Extension agreed with EC to 30 June 2020]
Submission Date	27.07.2020
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-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.

DISCLAIMER

COPYRIGHT NOTICE

VERSIONING AND CONTRIBUTION HISTORY

Version	*Date*	*Authors*	*Notes*
0.1	26.06.2020	MARIS	First version
0.2	16.07.2020	MARIS	Final version after TCOM review
0.3	23.07.2020	MARIS	Final version after comments round
1.0	27.07.2020	TRUST-IT	Deliverable submission

Contents

Executive summary

This report D2.6 describes the Blue Cloud architecture as it is known at Month 9. It is the first release of the architecture as it is expected that the actual development and implementation of the Blue Cloud system will provide further challenges and elaborated details. For that purpose, two more releases of the architecture document are planned, namely report D2.7 in Month 15 and report D2.8 in Month 27.

The technical framework of the pilot Blue-Cloud will feature:

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

In this report, the architecture and functionalities of each of these components are described in detail as well as the roles of partners that will develop and host modules.

The Blue Cloud Data Discovery and Access service architecture is based upon a combination of the GeoDab metadata broker service of CNR-IIA, and the SeaDataNet CDI service modules as developed by MARIS, IFREMER, and EUDAT in the framework of the EU SeaDataCloud project. For the Blue-Cloud Discovery and Access service and its modules, additional developments will be needed such as adapting and upgrading of existing services, adding new services, testing modules, integrating modules, and testing the integrated service, in order to achieve the planned functionality.

The Blue Cloud VRE will be based upon the existing **D4Science e-infrastructure** as developed and managed by CNR-ISTI. This e-infrastructure already hosts multiple Virtual Labs and offers a variety of services, which can be adopted for the Blue Cloud. The D4Science e-infrastructure also has proven solutions for connecting to external computing platforms and means for orchestrating distributed services, which will be instrumental for smart connections to the other e-infrastructures in the Blue-Cloud system.

The Blue Cloud demonstrators will be developed as Virtual Labs embedded in the D4Science VRE e-infrastructure and supported by data input from the Blue Cloud Data Discovery and Access service and other data resources, and possible additional computing services. The demonstrators are being worked out in a cooperation between WP3 and WP4, analysing their scientific workflows and technical set-up, and considering the present D4Science VRE infrastructure and services that will provide the basis platform. However, at this time of development, there is no insight yet whether the demonstrators will require upgrading of existing functionality or development of additional services. Therefore, the described architecture of the VRE concerns the current D4Science e-infrastructure.

In addition, consideration is given to integration aspects, such as two-way linking between the 2 Blue Cloud components, and expanding the VRE with additional platforms for computing and

algorithms, and where needed for specific demonstrators, direct access to data infrastructures. Moreover, aspects of authentication and monitoring are considered on full Blue Cloud scale.

The initial Blue Cloud architecture as described in this report, is designed to be scalable and sustainable for near-future expansions, such as connecting additional blue data infrastructures, implementing more and advanced blue analytical services, configuring more dedicated Virtual Labs, and targeting more (groups of) users.

1 Introduction

The technical framework of the pilot Blue-Cloud will feature:

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

A major challenge is to establish federations at the levels of data resources, computing resources and analytical service resources. The federation of data resources by means of the Blue Cloud Data Discovery and Access service should facilitate that multi-disciplinary datasets from existing blue data infrastructures can be shared with external users as well as can provide a major resource of data input for the Blue Cloud VRE. In addition, VRE users should be able to ingest data sets from other external sources, including their own data sources, using internal VRE functionality for data ingestion and for setting up and running direct access to selected remote blue data infrastructures. Overall, data input could be in-situ data, earth observation data, and model outputs. The federation by means of the Blue Cloud Virtual Research Environment (VRE), should facilitate that computing and analytical services can be shared and combined for specific applications. Analytical services can be various algorithms and generic services, for instance for sub setting, pre-processing, publishing, and viewing data and data products.

In the project, the Blue Cloud Data Discovery and Access service will be configured and deployed for a selected number of blue data infrastructures, represented in the Blue Cloud consortium. And the Blue Cloud VRE will be dimensioned to drive five high-level demonstrators whereby for each demonstrator a so-called Virtual Lab will be configured as a combination of analytical and generic services, selected from the overall service portfolio. There is already a large portfolio of existing services managed by the Blue Cloud founders. These will be used for developing and configuring the VRE and the Virtual Labs, whereby upgrading of selected existing services and developing additional services might take place for providing the planned Demonstrators their full requested functionality.

An important prerequisite for the Blue Cloud architecture is that it must be designed to be scalable and sustainable for near-future expansions, such as connecting additional blue data infrastructures, implementing more and advanced blue analytical services, configuring more dedicated Virtual Labs, and targeting more (groups of) users.

A start with detailing the Blue Cloud architecture was made at the 1st Blue Cloud TCOM meeting in Amsterdam – The Netherlands, 22 – 23 January 2020, where the overall concept of the Blue Cloud system with its two main components, the development of five demonstrators as Virtual Labs, the use of a brokerage service, and the EUDAT and D4Science e-infrastructures were presented and discussed. Following the TCOM, the concept of the Discovery and Access service was further elaborated and documented in D2.1 - Blue Data Infrastructures – Services Description Report, and followed up as part of the ongoing analyses for D2.2 - Blue Data Infrastructures – Services Analysis Report, which will be released at end of M10.

More input for the design of the Blue Cloud system architecture was gathered from the Blue Cloud demonstrator teams, asking for their perspective concerning the input, output, and workflow processes that they envision for demonstrator Virtual Labs. These requirements were documented in D3.1 - Demonstrator general technical requirements.

The Blue Cloud VRE will be primarily based upon the existing D4Science e-infrastructure, which already hosts multiple Virtual Labs and offers a variety of services, which can be adopted for the Blue Cloud, while new services might be specified and developed for full support of the Blue Cloud demonstrators. Further insights into the D4Science e-infrastructure, in particular from the perspective of how to configure Virtual Labs for deployment of the Blue Cloud demonstrators, were given as part of D3.2 – Demonstrator Implementation Guidelines.

Based upon these deliverables and discussions at several meetings since the 1st TCOM, a common and more detailed understanding about the Blue Cloud architecture was achieved which resulted in the first draft of the report D2.6 – Blue Cloud architecture (1st release). The D2.6 draft was presented and reviewed at the 2nd TCOM, which took place 29 – 30 June 2020, and this brought forward a number of suggestions, which were worked out in an edited version, which was circulated one more time to core members of the TWG for final acceptation. As it is expected that the actual development and implementation of the Blue Cloud system will provide further challenges and elaborated details, in near future two more releases of the architecture document are planned, namely report D2.7 in Month 15 and report D2.8 in Month 27.

2 Architecture of Blue Cloud Discovery and Access service

2.1 Overall concept

The **Blue Cloud Data Discovery and Access service** will be one of the two main components of the Blue-Cloud technical framework, next to the **Blue Cloud Virtual Research Environment (VRE)**. The Blue Cloud service will facilitate discovery and retrieval of in-situ data, earth observation data, data products, and model output, for external users in stand-alone mode, and for users of the VRE through connectivity. These data sets are managed in blue data infrastructures that will be connected to the Blue Cloud service to serve federated discovery and access.

The pilot Blue-Cloud project aims at federating initially in total 10 blue data infrastructures. Each of these existing infrastructures have been described in deliverable D2.1 - Blue Data Infrastructures – Services Description Report, in particular with a focus on their current data discovery and access mechanisms.

The overall concept is that the Blue-Cloud Data Discovery and Access service will harvest metadata from the blue data infrastructures by means of protocols such as CSW or OAI-PMH. As part of WP2 currently analyses are ongoing for preparing a mapping per catalogue service of each blue data infrastructure to extract and harmonise individual metadata outputs to a common metadata model (ISO19115 – 19139), starting at syntactic level. Thereby also a suitable aggregation is being explored and agreed with each blue data infrastructure in order to support a common Blue-Cloud catalogue with collections of a common level. The analyses are making good progress and will be documented in deliverable D2.2 - Blue Data Infrastructures – Services Analysis Report which is expected by end M9, while interim results will be discussed at the second TCOM meeting, planned end M8.

In the Blue Cloud project use will be made of the GEODAB metadata brokerage service software kit as developed and managed by CNR-IIA. The mappings are made against the common GEODAB metadata model, and the GEODAB service will be set up by CNR-IIA to generate, maintain, and provide the common Blue-Cloud catalogue in a dynamical way with the latest entries as derived from the blue data infrastructures.

For the data access part of the Blue-Cloud data discovery and access service, a data brokerage service will be developed, integrating the Blue Cloud metadata catalogue (see above), including a shopping mechanism and interfaces, both for human users and machines, to support the actual discovery and retrieval functions. This part will make use of the experience and software services that MARIS, IFREMER, and EUDAT have developed and are managing for the SeaDataNet CDI service. For the Blue Cloud selected services will be adopted and/or adapted.

As part of the activities for Deliverable D2.2 - Blue Data Infrastructures – Services Analysis Report, analyses are currently ongoing for each of the blue data infrastructures, focusing on their existing data delivery mechanisms and their fitness-for-purpose. As documented in D2.1, a number of blue data infrastructures are deployed as fully open data repositories with direct download links (data in different formats and standards), while another number are configured with a shopping mechanism, featuring user login. Use will be made of existing web services (API's), but where needed, new or adapted API's will be defined and agreed together with the technical representatives of the blue data infrastructures. These should then be configured by each to be fit for interacting with the central part of the data brokerage service. The API's should deal with the particulars of the local set-ups and also, they should arrange that data requests can be handled and responded at the agreed collection level (see above). Interim results for the choice of API's will be discussed at the second TCOM meeting, planned end M8, while the finally agreed API's will be documented in deliverable D2.2 - Blue Data Infrastructures – Services Analysis Report, expected by end M9.

The Blue Cloud data discovery and access service will provide a common interface, both by web services as by GUI, for discovery and retrieval of data collections from the federated blue data infrastructures. The GUI will include facilities for mapping and viewing the locations of data sets, as this will be part of the query dialogue. Moreover, conceptually it is planned to set up the query mechanism as a two-step approach, whereby the first step will focus on identifying interesting data collections and products, while the second step will focus on drilling down and subsetting within the identified collections and products in order to get more specific data. For the second step, geographic and temporal criteria will be instrumental, next to additional criteria which could be specific per blue data infrastructure. Finally, users should be able to download and store the retrieved data collections on their own machines or in a data pool as part of the Blue Cloud VRE.

The two-step approach for data discovery and access is necessary to deal with most of the blue data infrastructures, in particular in cases with observation (raw) data which often can be very large collections with numerous data sets. The second step is then necessary to select a specific geographical area or a specific time slot or a specific variable from that large collection. There are also cases, when one step can be sufficient, such as in case of specific data products, that a user wants to download as a complete file. The geographic query options should be supported by a mapping interface, which can be compiled from OGC WMS services to be provided by the blue data infrastructures.

Implementing this approach will largely depend on the interfaces of blue data infrastructures, that should be supportive. This is part of the ongoing analyses for D2.2. and as a result, it might be that some blue data infrastructures will have to update or expand their existing services for metadata and data access.

The resulting Blue Cloud Data Discovery and Access service will facilitate users:

- to search and discover interesting data sets
- to complete and submit a shopping basket with interesting data sets
- to stay informed about the progress of the shopping requests

- to download the data sets once ready for downloading
- to ingest data sets into the VRE data pool for use in VRE applications.

It will facilitate managers of blue data infrastructures:
- to stay informed about the shopping requests and associated users for their repository
- to prepare periodic management reports

This way, the Blue Cloud Data Discovery and Access service will provide a delayed mode service to oversee and to select interesting data sets from the connected blue data infrastructures, followed by downloading and using of the selected data sets by external and VRE users. Next to the offer provided by the blue data infrastructures, the Blue Cloud Data Discovery and Access service will also index and make available selected data products, resulting from the Blue Cloud demonstrator Virtual Labs, to support a wider distribution and publishing.

The development of the Blue Cloud Data Discovery and Access service is undertaken as part of WP2 and its operational release is planned for end M17 (early March 2021).

2.2 Involved blue data infrastructures

The following blue data infrastructures are pillars under the initial Blue Cloud data discovery and access service:
- SeaDataNet (physics, bathymetry, chemistry, geology, geophysics, and biology)
- EMODnet Bathymetry (bathymetry)
- EMODnet Chemistry (chemistry)
- EurOBIS – EMODnet Biology (marine biodiversity)
- Euro-Argo and Argo GDAC (ocean physics and marine biogeochemistry)
- ELIXIR-ENA (biogenomics)
- EuroBioImaging (microscopy)
- EcoTaxa (bio images)
- WekEO (CMEMS ocean analysis and forecasting and C3S climate analysis and forecasting)
- ICOS-Marine (carbon).

These blue data infrastructures are mostly complementary to each other, dealing with other data originators and/or different stages in the processing chains from data acquisition to data products to knowledge. Each of them has been described in deliverable D2.1 - Blue Data Infrastructures – Services Description Report.

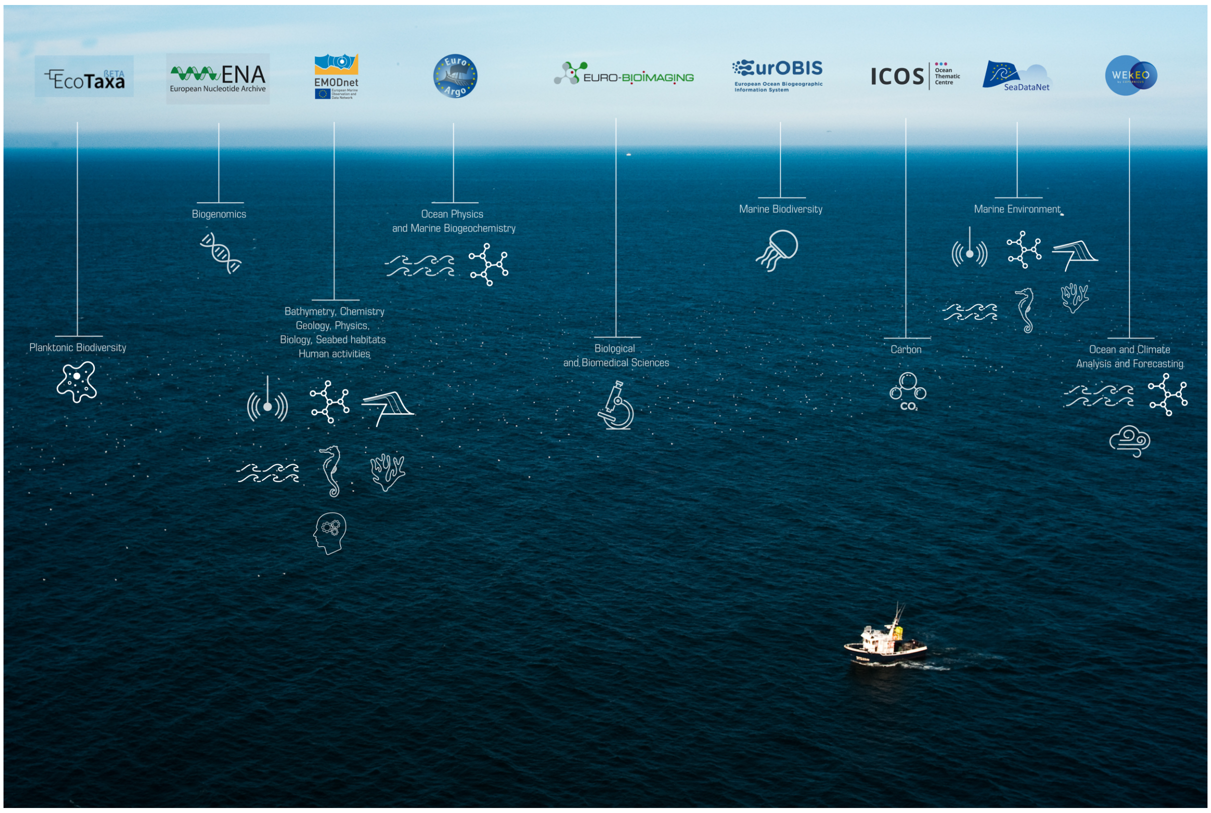


Figure 2.1: Blue data infrastructures as included in initial Blue Cloud discovery and access service

2.3 Architecture and modules

The Blue Cloud Data Discovery and Access service will consist of a number of modules (services) as indicated in the figure 2.2:

- **Blue Cloud metadata brokerage**, operated by CNR-IIA, dynamically interacting with each of the blue data infrastructures to retrieve, extract and harmonise metadata entries for each blue data infrastructure into a common Blue Cloud metadata catalogue;
- **Blue Cloud data sources**, comprising blue data infrastructures, that are gathering and managing catalogues and data collections from multiple data and data product originators; the Blue-Cloud VRE is also considered as Blue Cloud data source, concerning publishable data products as resulting from the demonstrator Virtual Labs.

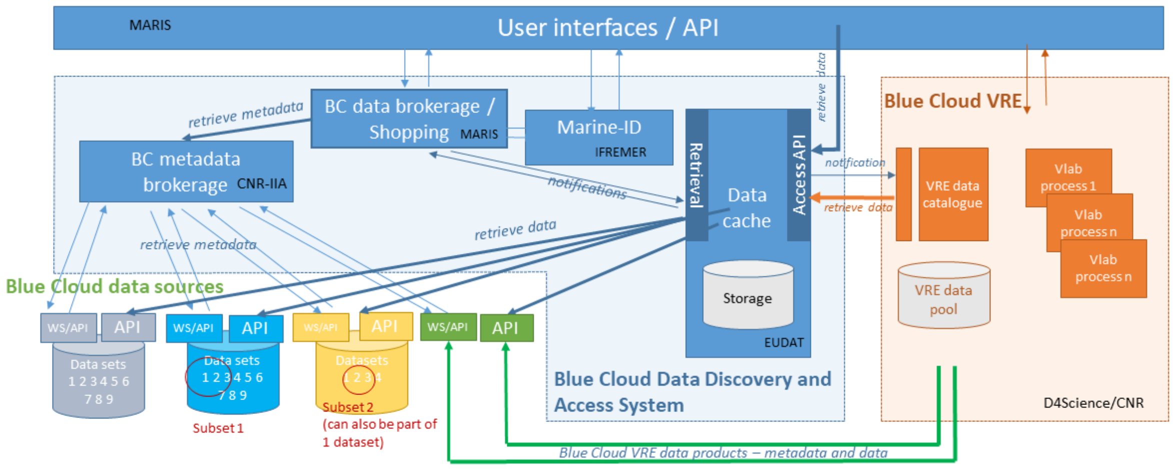


Figure 2.2: Architecture of the Blue Cloud discovery and access service

- **Marine-ID service,** operated by IFREMER, for registration and authentication of users to the Blue Cloud Data Discovery and Access service. Users only have to register once to receive their login details;
- **Data cache**, operated by EUDAT, for temporary storage of data packages, consisting of data sets, retrieved from the Blue Cloud data sources, plus associated metadata, as retrieved from the Blue Cloud data brokerage, and following the instructions as received from the Blue Cloud data brokerage. External users can download these data sets, after receiving information from the Blue Cloud data brokerage, while the VRE can also be triggered to retrieve data packages for ingestion into the catalogue and data pool of the Blue Cloud VRE;
- **Blue Cloud Data brokerage service,** operated by MARIS. This service performs the master role in the Blue Cloud Data Discovery and Access service, interacting with the other modules. Regularly, it retrieves the latest Blue Cloud metadata catalogue from the Blue Cloud metadata brokerage, and ingests this into the discovery interface, whereby users can query the catalogue. The common metadata catalogue should include sufficient metadata for each blue data source to allow a two-step query approach, going from identifying interesting data collections and products to subsetting within the identified collections and products in order to get more specific data. Part of the metadata to support the query concept should be OGC WMS – WFS services, supporting detailed mapping and viewing of data locations. The Blue Cloud Data brokerage service also contains a shopping mechanism with basket and ledger, by which users (external users and VRE) and blue data infrastructures can be informed about shopping transactions and their status in time. It interacts with the Blue Cloud Data Cache to give it precise instructions about retrieving data sets from the blue data infrastructures and to insert these for temporary storage, and to bundle these with associated metadata as retrieved from the Blue Cloud Data brokerage service. It interacts with the Marine-ID service as users need to login to submit shopping baskets and to have access to the transaction ledger. It interacts with registered users and VRE to inform and instruct them about data

packages that are ready for downloading by users or retrieval for ingestion by the VRE. Finally, it also interacts with the Blue Cloud Data Cache to receive information about the actual downloading by users and retrieval for ingestion by the VRE in order to update the ledger;

- **User interfaces and API's,** operated by MARIS, to interact with users for discovery and shopping transactions, and to provide access to the transaction ledger for users and blue data infrastructures.

In this figure, the Blue Cloud VRE is given with simplified graphics, only to underpin the exchange with the Blue Cloud Data Discovery and Access service. Much more detail about the Blue Cloud VRE will be given in the next chapter.
The Blue Cloud Data Discovery and Access service will focus on making data collections discoverable and retrievable, supporting subsetting, albeit at a metadata level. In a later stage, there is ambition to explore also inclusion of subsetting functionality at data level to support more smart retrieval of subsets from data collections.

2.4 More details on interaction with temporary Data Cache at EUDAT

Several of the modules in the Blue Cloud Data Discovery and Access service will be based upon the experience and services that MARIS, IFREMER and EUDAT have developed and are operating for the SeaDataNet CDI service as a result of the EU SeaDataCloud project. Several services will be adapted or upgraded to be fit for the Blue Cloud situation.

Just like in the SeaDataCloud project, EUDAT will provide the data cache where data from various sources can be brought together and aggregated for download by the user or for transfer to the VRE data pool, where it can be used by the VRE users. The data cache component is situated between the Shopping Service / Data Brokerage Mechanism, developed and operated by MARIS, the various data sources, and the VRE data pool. It consists of three components: The actual storage, the accessing part, and the API (see also figure 2.2):

- **Retrieval component:** The Data Brokerage will trigger the data cache and provide information about how and where to access the data that is to be aggregated. The EUDAT Retrieval component uses this information, along with credentials kept in a secure store, to retrieve the data from the diverse data sources;
- **Storage component:** The data will be stored on EUDAT's storage facilities during the data access process. The exact EUDAT service to be used is to be determined based on technical requirements (e.g. possible persistence in case of later similar downloads or later access by the user via different means, need for identification, etc.). Annex 1 gives an overview of the EUDAT service stack which comprises various B2XXX services. In the storage case, EUDATs B2SAFE or B2STAGE might be used. In case none of B2SAFE's or B2STAGE's key functionalities are required, a technically less complex solution such as plain iRODS installation, a dedicated file system storage, or an object storage system, can be envisaged;
- **EUDAT data access API:** This is the part where the aggregated data can be accessed from other components of the Blue-Cloud system. The data will be accessed by two different paths: Direct download by users, or transfer to the VRE data pool:

- **Direct download by users:** For users who wish to download their data for further use on their laptops, desktop computers or in their home institutions, a download link will be provided. A solution similar to the one used in SeaDataCloud might be implemented or reused. This implicates that a user communicates the desire to download to the GUI at the Shopping located at MARIS, which transmits it to the EUDAT API, which then returns a download link, which the Data Brokerage service returns to the user who clicks it and receives the data. This way, the user downloads the data directly from EUDAT, thus an additional transfer to the Data Brokerage premises is unnecessary. All the communication leading up to it happens between the user and the Data Brokerage service.
- **Transfer to the VRE data pool:** Users might want to transfer their data to the VRE data pool for further processing on the powerful D4Science VRE premises. This will take place in a very similar way as the download by users described above, except that the download will be carried out by the VRE. The experiences in SeaDataCloud have shown that pulling/downloading data is much more efficient and less error-prone than pushing/uploading the data, so the workflow will be:
  a. The EUDAT API receives a request for transferring a batch of data to the VRE data pool (request from data broker to EUDAT API).
  b. The EUDAT component notifies the VRE that a data batch is available for pull (asynchronous request from EUDAT to VRE API, including download link and user name).
  c. The VRE downloads the data using the specified link (request from VRE to EUDAT API).
  d. Then, the data can be deleted or kept for further use (e.g. later downloads or transfers). This will depend on policies as well as on technical considerations and is not decided yet.

To realize this workflow, a component will have to be developed and added by CNR-ISTI to the VRE that can handle requests to pull data. As both CNR-ISTI and EUDAT's participating data centres DKRZ, CINECA and CSC are part of the European research network GÉANT, this transfer will benefit from the very fast network backbone between major European research centres.

Authentication and authorization aspects:

As all the communication with the individual users (except for the actual data download) occurs between the user and the Data Brokerage service, the requests to the EUDAT API will not need to authenticate the user on whose behalf a request happens. Since the Data Brokerage service must authenticate the users anyway, using the Marine-ID service, repeating this step on the backend is not necessary. Similarly, the authorization happens at the Data Brokerage service, as it knows best which users are allowed to access which data. The EUDAT API authenticates the Data Brokerage service (to make sure that only legitimate requests are made to the data backend) and assumes that the data broker is authorized to access any part of the data. This reduces complexity, avoids redundancy and thus inconsistency in the permissions management.

The user will be informed by e-mail by the Data Brokerage service that his/her request has been processed and that the data package is ready for downloading. The user then has to login with Marine-ID to his/her account at the Data Brokerage service to find the shopping request and the download option. At that moment, the Data Brokerage service will communicate with the EUDAT API to retrieve a temporary link (say valid for 6 hours) which will allow the user to do the actual download from the Data Cache at EUDAT. The download option for a shopping request will stay active in the user account at the Data Brokerage service for a longer period (say 15 days), but when using each time a new temporary download link will be requested by the Data Brokerage service from the EUDAT API and provided to the user. This set-up already functions very well in the new SeaDataNet CDI service.

2.5 Distribution of Blue Cloud VRE data products through Blue Cloud Data Discovery and Access service

As indicated in D1.2 - Data Management Plan –, the Blue Cloud project aims to deploy the FAIR principles for data management in Blue Cloud, making data findable, accessible, interoperable and re-usable (FAIR). This not only applies to data input, but also to the data products that researchers using and running the five Virtual Labs at the Blue-Cloud VRE will generate and would like to disseminate and publish.

The D4Science infrastructure (see also chapter 3) as basis for the Blue Cloud VRE already has a utility for building and maintaining provenance information about the processes as applied for generating specific data products in the Virtual Labs. Functionality includes storing information on data input, use and settings of algorithms, and data output into standard PROV records (the PROV standard[1] defines a data model, serializations, and definitions to support the interchange of provenance information on the Web). These provenance records support to document data products and provide very useful information about the data products for any potential use and re-use performed both by end-users and machine algorithms. They are also very useful for researchers that want to analyse and re-run the analytical processes, possibly changing input or settings for comparisons. Moreover, the D4Science infrastructure already has a catalogue service for all data sets and data products as managed for the Virtual Labs. This concerns two catalogues:

- a **CSW-compliant catalogue,** based on GeoNetwork technology, enabling users to browse and search for geospatial items by relying on the accompanying metadata;
- an **SDMX-compliant catalogue,** based on Fusion Registry technology – for searching statistical data by relying on their structural metadata;

which are joint in one **overall catalogue,** based on CKAN technology, which enables users to perform faceted search on the entire set of resources managed by the Virtual Labs. These existing D4Science services will provide the basis and will be adapted to serve as the so-called Blue Cloud VRE data catalogue and VRE data pool.

[1] PROV standard: https://www.w3.org/TR/prov-overview/

The VRE data catalogue and data pool will give users overview and access to all data sets as included in the VRE as input, but will contain also all data products generated in the Virtual Labs. It should be made possible that VRE researchers can label a subset of their data products for wider publication, which will allow to compile automatically a dedicated service to give metadata about and access to public VRE data products. This VRE products catalogue and products pool will be treated like one of the Blue Cloud data sources and connected to the Blue Cloud Data Discovery and Access service in a comparable way as the blue data infrastructures. The development and configuration of the VRE data catalogue and pool, including the public data product services, will take place in WP4. This will include developing of web services and/or API's for sharing product metadata and giving access to the data products.

The overarching VRE data catalogue and data pool are realised through a family of technologies leveraging cloud-based technologies to deliver high scalability, security, reliability, and availability. To any object stored in it, a unique and persistent identifier is generated and a unique and persistent web locator (PURL) is generated. Versioning is automatically enabled to guarantee persistence also when new versions are incrementally generated and stored.

2.6 Direct Blue Cloud VRE connectivity to selected blue data sources

As indicated earlier, the Blue Cloud Data Discovery and Access service will be a delayed mode service as it will work with retrieving selected data sets from blue data sources, and thereafter providing these for downloading by external users as well as for retrieval and ingestion into the VRE data catalogue and data pool services. An important function is to provide external and VRE users a common interface for discovery and access to a range of well-established and maintained blue data sources. The delayed mode will not hamper most of the VRE demonstrators, as several have a preparatory phase in which they gather and organise their data input, followed by processing and analytical steps. However, in case of demonstrator 3 – environmental indicators – there is a much more operational need for fast and regular updated access to specific data sets, in particular from WEkEO (bundling CMEMS, C3S, and Sentinel data and products) and Argo GDAC.

For these kind of operational purposes, it should also be made possible in the Blue Cloud architecture to establish and manage direct connections from the Virtual Labs to blue data infrastructures. These connections might be preconfigured with settings, underpinning the data requirements of the demonstrator. Also, technologies should be explored for optimising the transfer, e.g. by streaming, parallelisation of data transfer from source to VRE, pre-processing at the source for reducing transfer volumes, and deploying applications as Docker containers at the source platform for running applications close to the data, remotely steered from the VRE.

The following Chapter 3 will go deeper in the current architecture and services of the D4Science infrastructure that will provide the basis for the Blue Cloud VRE. While Chapter 4 will highlight a number of integration issues which are relevant in the case of direct connectivity as discussed in this paragraph.

3 Architecture of Blue Cloud Virtual Research Environment (VRE)

The Blue Cloud Virtual Research Environment (VRE) will facilitate collaborative research using a variety of data sets and analytical tools, complemented by generic services such as sub-setting, pre-processing, harmonizing, publishing and visualization. Within the Blue Cloud project, a number of Demonstrators will be developed. Each Demonstrator will enact a family of analytical workflows (or pipelines) which consist of a series of applications and make use of selected datasets as input. Virtual Labs will be configured within the Blue Cloud VRE, each implementing a Demonstrator. For each Virtual Lab group accounts of researchers should be configured. The multi-disciplinary datasets can be retrieved from the blue data infrastructures by means of the Blue Cloud data discovery and access service, or can be retrieved and ingested by users from other data portals and (own) resources.

The Blue Cloud VRE will be based upon the existing **D4Science e-infrastructure** as developed and managed by CNR-ISTI. This e-infrastructure already hosts multiple Virtual Labs and offers a variety of services, which can be adopted for the Blue Cloud. The D4Science e-infrastructure also has proven solutions for connecting to external computing platforms and means for orchestrating distributed services, which will be instrumental for smart connections to the other e-infrastructures in the Blue-Cloud system.

The Blue Cloud demonstrators will be developed as Virtual Labs embedded in the D4Science VRE e-infrastructure and supported by data input from the Blue Cloud Data Discovery and Access service and other data resources, and possible additional computing services. The demonstrators are being worked out in WP3 for their scientific workflows and technical set-up, thereby considering the present D4Science VRE infrastructure and services as basis. For this process, there is close cooperation between WP3 and WP4, as the demonstrators might require adaption of existing services and development of additional services. This has resulted so far in Deliverable D3.1 - Demonstrator general technical requirements which gives information from the perspective of the demonstrator scientific teams concerning the input, output, and workflow processes that they envision for their Virtual Labs. A second Deliverable D3.2 – Demonstrator Implementation Guidelines was prepared by CNR-ISTI as manager of the D4Science e-infrastructure and as WP4 leader, providing guidance on how to use the D4Science infrastructure and its VRE for configuring Virtual Labs.

The further detailing of each of the demonstrators and analysing the impact of the confrontation between demonstrator requirements and present VRE provisions are currently ongoing and have not yet resulted in firm conclusions. Therefore, there is no insight yet whether the demonstrators might require upgrading of existing functionality or development of additional services. For that reason, the functionalities and architecture of the current D4Science e-infrastructure will be described in detail.

3.1 D4Science e-infrastructure – general overview

The D4Science architecture consists of a hardware layer and a service layer. The hardware layer is organized as a dynamic pool of virtual machines, supporting computation and storage, while the services layer is organized into e-infrastructure middleware, storage, and end user services. The hardware layer consists of an OpenStack installation, supporting the deployment of services in the upper layer by provision of computational and storage resources. The service layer, illustrated in **Figure 3.1**, consists of five service frameworks, which can be summarized as follows:

- ***Enabling Framework***: the enabling framework includes services required to support the operation of all services and the VREs supported by such services. As such it includes: a *resource registry* service, to which all e-infrastructure resources (data sources, services, computational nodes, etc.) can be dynamically (de)registered and discovered by user and other services; *Authentication and Authorization* services, as well as *Auditing* Services, capable of both granting and tracking access and usage actions from users; and a VRE manager, capable of deploying in the collaborative framework VREs inclusive of a selected number of "applications", generally intended as sets of interacting services;
- ***Storage Framework***: the storage framework includes services for efficient, advanced, and on-demand management of digital data, encoded as: files in a distributed file system, collection of metadata records, and time series in spatial databases; such services are used by all other services in the architecture, exception made for the enabling framework;
- ***Analytics Framework***: the analytics framework includes the services required for running methods provided by scientists taking advantage, in transparent way, of the power of the underlying computation cloud (e.g. parallel computation) and of a plethora of standard statistics methods, provided out of the box to compute over given input data;
- ***Collaborative framework***: the collaborative framework supports all VREs deployed by the scientists and for each of them provides *social networking* services, *user management* services, *shared workspace* services, and WebUI access to the information cloud and to the analytics framework, via *analytics laboratory services.*

The *Content Cloud Framework* instead will not be exploited by Blue-Cloud and it will not be reported in this document.

- *Content Cloud Framework*: the content cloud framework includes all services required to collect, transform, harmonize, and provide via APIs of different kinds all metadata records of interest to the D4Science community and provided by data sources managed by the organizations in the D4Science consortium. The data collection and provision activity are ruled by workflows, configured by data curators (e.g. transformation rules) and orchestrated by a local enabling layer, in order to keep the content cloud up to date with respect to the content available in the aggregated data sources;

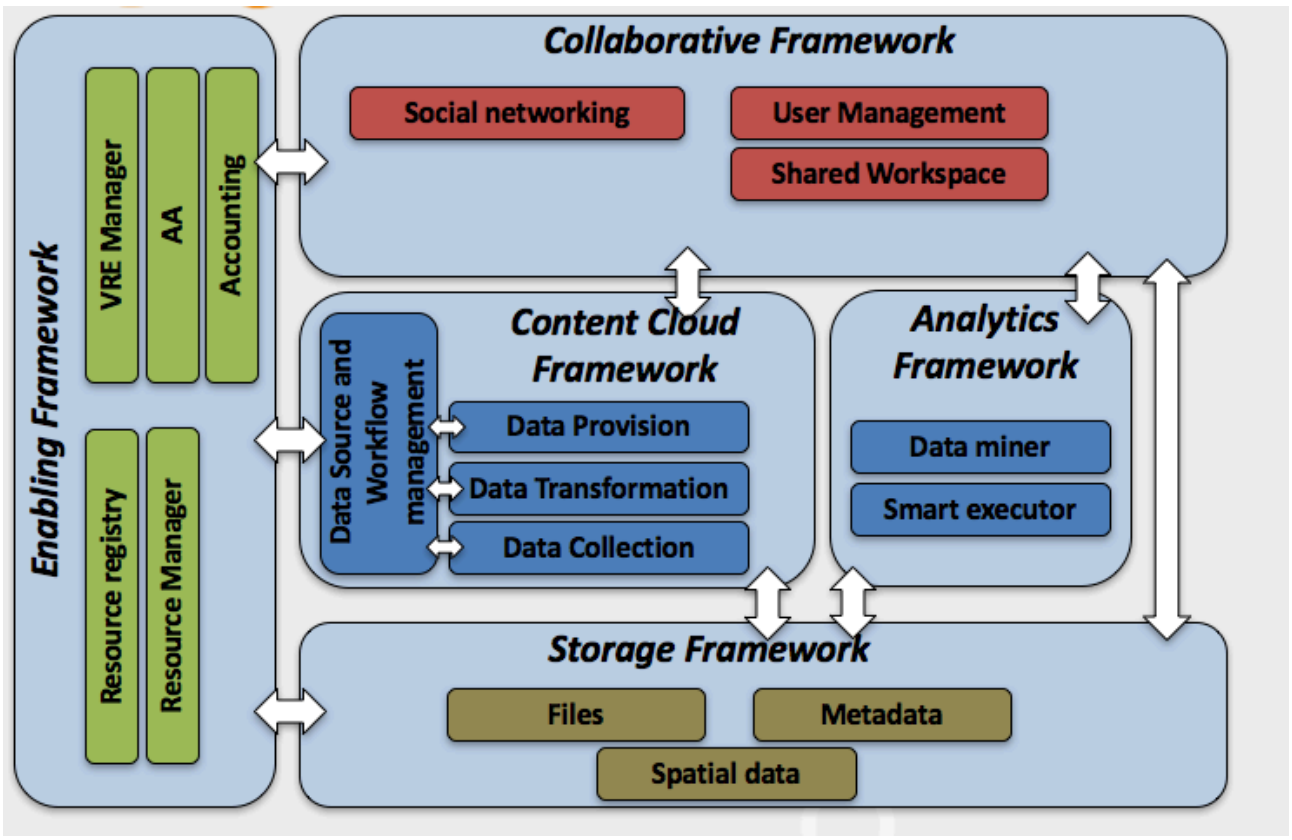


Figure 3.1: High-level architecture of the D4Science infrastructure

3.2 D4Science Hardware Layer

3.2.1 Enabling Technology

The following well-known technologies have been selected to manage the D4Science hardware resources:

a. Ceph, http://www.ceph.com, has been selected as block storage since it is Amazon S3 compatible and OpenStack Swift compatible, it is completely distributed, and it may even use disposable server hardware;

b. Openstack, http://www.openstack.org, has been selected as cloud-computing software platform. It uses ceph as storage;

The ***Ceph Storage*** offers object, block, and file storage under a unified system. It has been designed to provide excellent performance, reliability and scalability. It supports rapid provisioning of massively scalable cloud storage and enables computation intensive workloads. It provides access to the storage via application written in Java, Python, Ruby, C, etc. It scales to Petabytes and it offers linear scaling with linear performance increase.

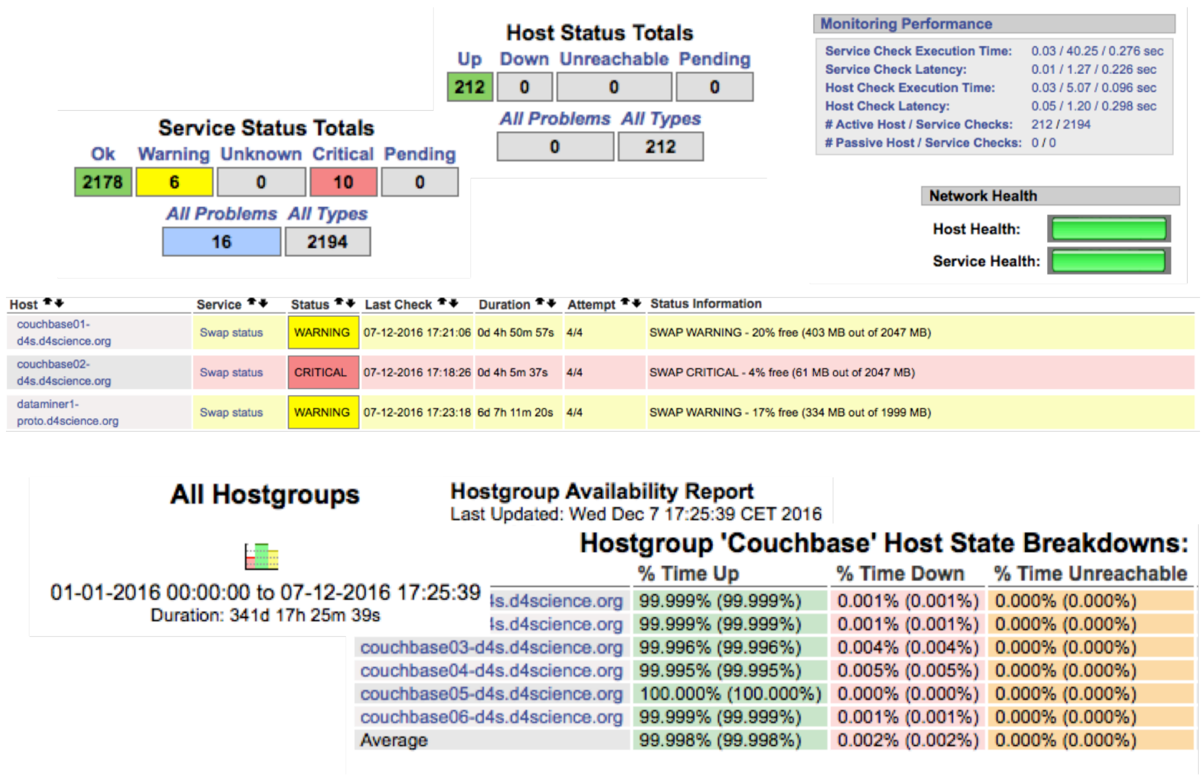
The Openstack, open source cloud computing platform, provides Infrastructure-as-a-Service (IaaS). OpenStack lets the D4Science Enabling Framework deploys virtual machines and other instances that handle different tasks on the fly. It makes horizontal scaling affordable, which means that services that benefit from running concurrently can easily serve more or fewer tasks – issued either by users or by other services - on the fly by just spinning up more service instances.

3.2.2 Supporting Technology

3.2.2.1 Monitoring and Alerting System

The D4Science Infrastructure is currently counting 212 servers. This does mean neither that all of them are exploited at the same time nor that all of them have to be active concurrently to deliver specific service capabilities. Servers are allocated dynamically in accordance with the Cloud-computing approach and are activated/deactivated in response to load, failures, changes in policies and deployment strategies. This complexity requires proper monitoring infrastructure to check the servers and the services running on the servers and to proper alert when failures are identified. The D4Science exploits two well-known technologies to perform this task: Nagios and Prometheus.

Nagios is an enterprise-class monitoring and alerting solution that provides extended insight of the infrastructure enabling quickly identification and resolution of problems before they may affect critical business processes. It provides monitoring of all mission-critical infrastructure components including applications, services, operating systems, network protocols, systems metrics, and network infrastructure. Nagios provides a central view of operations, network, and business processes running on the infrastructure. Powerful dashboards provide at-a-glance access to powerful monitoring information and third-party data. Views provide users with quick access to the information they find most useful. Spot problems easily with advanced data visualization reports. Moreover, alerts are sent to infrastructure managers and the D4Science quality assurance task force via email or mobile text messages, providing them with outage details so they can start resolving issues immediately. Finally, multiple APIs provide for simple integration with in-house and third-party applications. In particular, for well-known technologies exploited in the D4Science e-infrastructure, e.g. MongoDB, Cassandra, Couchbase, PostgreSQL, etc, existing add-ons have been installed to extend monitoring and native alerting functionality; for technologies developed both by D4SCIENCE and by the exploited framework, i.e. gCube and D-Net, specific add-ons have been designed, implemented, and installed to extend monitoring and native alerting functionality in order to have a fully-complete and always up-to-date image of the status of the D4Science e-infrastructure. Overall, 2194 service checks have been added and continuously executed to the monitoring and alerting infrastructure.



Host Status Totals

Up	Down	Unreachable	Pending
212	0	0	0

All Problems	All Types
0	212

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
2178	6	0	10	0

All Problems	All Types
16	2194

Monitoring Performance

Service Check Execution Time:	0.03 / 40.25 / 0.276 sec
Service Check Latency:	0.01 / 1.27 / 0.226 sec
Host Check Execution Time:	0.03 / 5.07 / 0.096 sec
Host Check Latency:	0.05 / 1.20 / 0.298 sec
# Active Host / Service Checks:	212 / 2194
# Passive Host / Service Checks:	0 / 0

Network Health

Host Health:	🟩
Service Health:	🟩

Host	Service	Status	Last Check	Duration	Attempt	Status Information
couchbase01-d4s.d4science.org	Swap status	WARNING	07-12-2016 17:21:06	0d 4h 50m 57s	4/4	SWAP WARNING - 20% free (403 MB out of 2047 MB)
couchbase02-d4s.d4science.org	Swap status	CRITICAL	07-12-2016 17:18:26	0d 4h 5m 37s	4/4	SWAP CRITICAL - 4% free (61 MB out of 2047 MB)
dataminer1-proto.d4science.org	Swap status	WARNING	07-12-2016 17:23:18	6d 7h 11m 20s	4/4	SWAP WARNING - 17% free (334 MB out of 1999 MB)

All Hostgroups

01-01-2016 00:00:00 to 07-12-2016 17:25:39
Duration: 341d 17h 25m 39s

Hostgroup Availability Report
Last Updated: Wed Dec 7 17:25:39 CET 2016

Hostgroup 'Couchbase' Host State Breakdowns:

	% Time Up	% Time Down	% Time Unreachable
s.d4science.org	99.999% (99.999%)	0.001% (0.001%)	0.000% (0.000%)
s.d4science.org	99.999% (99.999%)	0.001% (0.001%)	0.000% (0.000%)
couchbase03-d4s.d4science.org	99.996% (99.996%)	0.004% (0.004%)	0.000% (0.000%)
couchbase04-d4s.d4science.org	99.995% (99.995%)	0.005% (0.005%)	0.000% (0.000%)
couchbase05-d4s.d4science.org	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)
couchbase06-d4s.d4science.org	99.999% (99.999%)	0.001% (0.001%)	0.000% (0.000%)
Average	99.998% (99.998%)	0.002% (0.002%)	0.000% (0.000%)

Figure 3.2: Nagios Status Report and Availability Report for the Auditing Cluster

Prometheus is an open-source system monitoring and alerting toolkit originally built at SoundCloud[2]. It is a standalone open source project and maintained independently of any company. Prometheus's main features are:

- a multi-dimensional data model with time series data identified by metric name and key/value pairs;
- time series collection happens via a pull model over HTTP;
- PromQL, a flexible query language to leverage this dimensionality;
- no reliance on distributed storage; single server nodes are autonomous;
- pushing time series is supported via an intermediary gateway;
- targets are discovered via service discovery or static configuration;
- multiple modes of graphing and dashboarding support.

The multiple modes of graphing and dashboarding support feature has been exploited by adopting Grafana[3], which allowed to query, visualise, alert on and understand Prometheus data on metrics.

Grafana allowed to virtually define dashboards as set of servers that collectively perform a specific task. In the D4Science we defined a catch-all dashboard to include all servers and then specific virtual clusters to monitor the performance and the exploitation of physical resources for the key enabling software frameworks exploited in the infrastructure and reported in Section 2.2 and subsequent sections.

[2] http://soundcloud.com/
[3] https://grafana.com/grafana

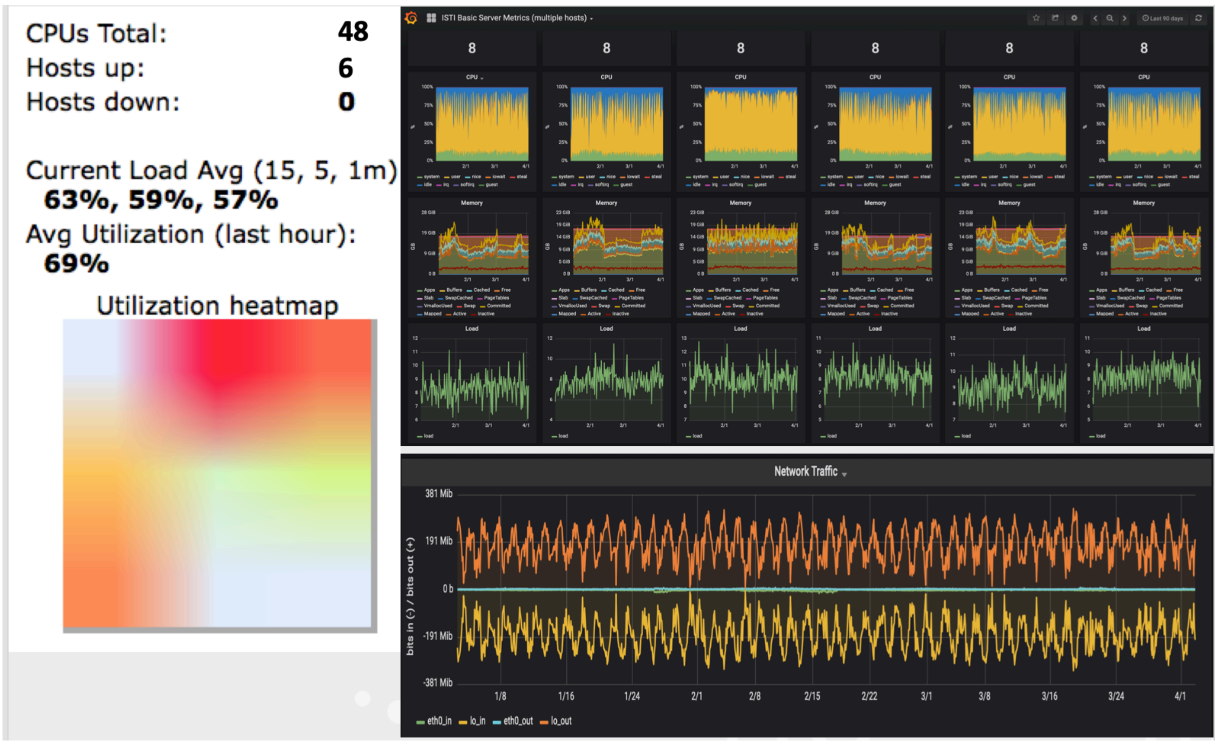


Figure 3.3: Prometheus Aggregated View via Grafana for the Auditing Cluster

3.2.3 Provisioning System

The D4Science is currently counting 212 servers and one of its core ambitions in designing it was the reduction of the deployment, operation, and maintenance costs. To achieve this ambition a key aspect was to automatize the configuration and management of servers, combining multi-servers software deployment, supporting ad hoc task execution, and configuration management.

Ansible is a free-software platform allowing to configure servers according to *idempotence*. Idempotency is basically based on the description of what state is required on a server and Ansible figures out how to get to that state. This approach is opposite to other approaches that require to specify what to run on a server and how to run it. This allows to reduce drastically the costs of operations since it becomes possible to run Ansible plays over and over and it does the right thing according to the status of the server instead of repeating commands and configurations. Ansible is really useful for repeatedly setting up servers in the Cloud which need to be set up the same way.

In order to exploit Ansible in the D4Science e-infrastructure, it was needed to define a number of resources and configuration scripts that than are exploited by Ansible to perform the activities

- inventories - list of servers to configure and maintain
- playbooks - collection of plays, or simply a collection of roles for a 1-play playbook
- plays - a collection of roles
- roles - generally, one service (like postgres or nginx)
- tasks - a command that Ansible runs via its modules, like a task for installing a package via apt-get

- handlers - like tasks that get called when other tasks request them via notifications. Typically used to restart services.
- host vars - variables that apply to one collection of hosts
- modules - provided by Ansible to do things like configure MySQL (mysql module), install via apt-get (apt module), copy over files (file module), add users (user module).

Overall, to manage the D4Science 197 roles have been defined.

3.3 D4Science Enabling Framework

3.3.1 Overview

The Enabling Framework is realized by a combination of services and libraries powered by the gCube System open-source project. Those services promote the optimal exploitation of the resources available in the D4Science Cloud Infrastructure and the integration of technology external to it. They insulate as much as possible the management of the infrastructure from the data and the data management services that are hosted in or accessible through the infrastructure itself.

The motto at the heart of the management facilities is *less dependencies for more management* meaning that the requirements posed to resources (even independent resources) to be managed are minimal, close to zero in some cases. All the implemented solutions are prioritized in order to pursue this goal.

Towards new directions of openness and interoperability called by our growing community, management facilities move along:

- adoption of standards
- support for new software platforms by implementing a zero-dependency approach to software management.

The Enabling framework is composed by three main systems: Resource Management System, Information System, and Security System. These are complex ICT systems that exploit tailored persistence technologies managed via web services.

The Resource Management System supports the creation of a Virtual Research Environment and its exploitation via the registration, management, and utilization of the resources assigned to it.

The Information System supports the registration, discovery, and access of the resources profile.

The Security System ensures the correct exploitation, auditing, and Auditing of the resources under the policies defined at registration time and customized at VRE definition time. It is orthogonal to all services operating in the infrastructure and its components are deployed on all computing nodes.

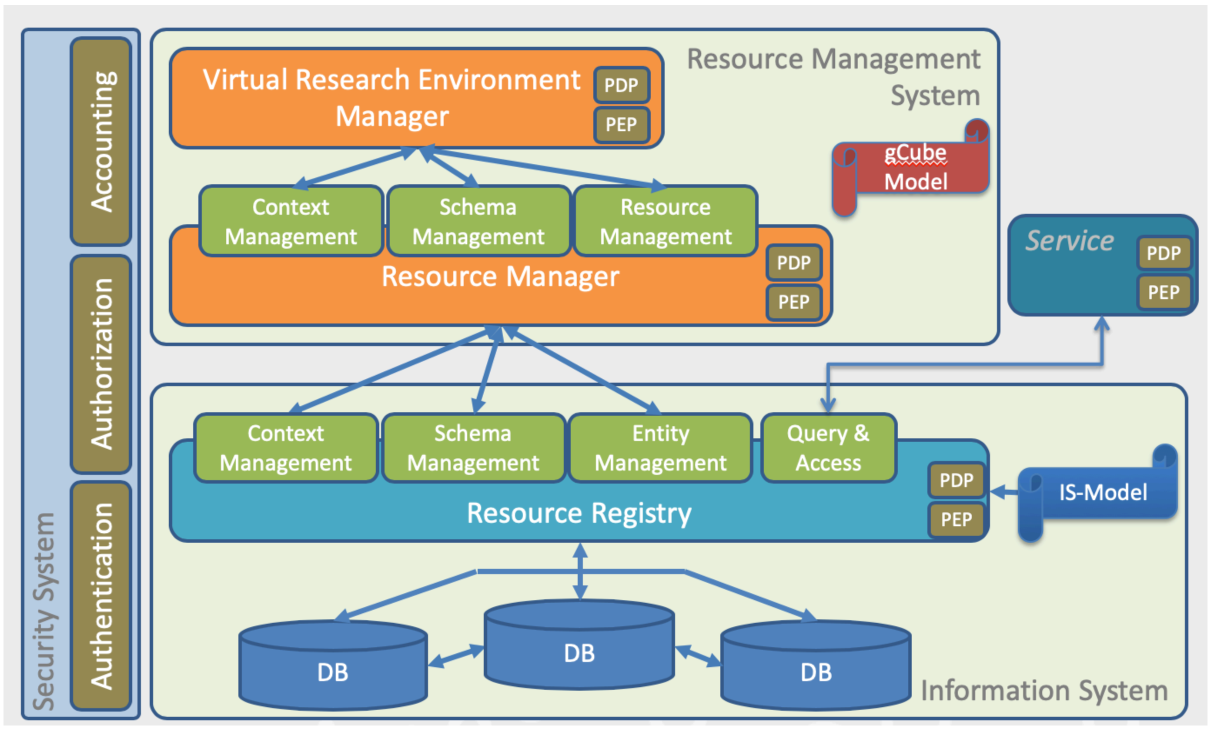


Figure 3.4: Enabling Framework Architecture

3.3.2 Key Features

Extensible notion of resource	a resource model which is open to modular extensions at runtime by arbitrary third parties
Transparent software resource management	nearly zero-dependency requested to managed resources for being part of the infrastructure
Environment propagation	operational information among services are transparently propagated over a range of protocols (SOAP, HTTP/S, and more)
Dynamic Deployment and Optimal Resource (re)Allocation	remote deployment and (re-)configuration of resources across the infrastructure
Resource lifetime management	complete running of the entire lifetime of resources ranging from creation and publication to discovery, access and consumption
Self-elastic management	dynamic resource provisioning to meet peaks and lows in demand
Interoperability, openness and integration at software level	third-parties software can be added to the infrastructure at runtime
Support to standards	crucial functionalities are accessible via recognized standards in order to enhance interoperability

3.3.3 Subsystems

3.3.3.1 Resource Registry

The gCube Resource Registry is the core subsystem connecting producers and consumers of resources. It acts as a registry of the infrastructure by offering global and partial views of its resources and their current status and notification instruments.

The approach provided by the Resource Registry is of great support for the dynamic allocation of resources and the interoperability solutions offered by the Resource Manager system. The feedback obtained during the first reporting period has been used to improve the quality of the design, the APIs of both services and clients, the design of the Graphical User Interfaces (GUIs), and the REST APIs (to strictly adhere to REST principles). Furthermore, the client's APIs has been simplified and enriched: two new Java clients have been released: Resource Registry Context Client and Resource Registry Schema Client which now makes a total of four Java Clients:

- Resource Registry Context Client;
- Resource Registry Schema Client;
- Resource Registry Publisher;
- Resource Registry Client.

Key Features

Resource Publication, Access and Discovery	The Resource Registry is functionally complete offering Java and WEB APIs to register new resources, to discover, and access them.
Consistency with the new Resource Model	The Resource Registry grants publication and access to resources compliant with the Resource Model
Production level QoS - Responsiveness	Each query served in milliseconds; thousands of queries served each hour
Production level QoS - Scalability	Infrastructures with more than 100K of resources successfully powered
Production level QoS - Permanent and Uninterrupted Functioning	The Resource Registry instances have been continuously up for more than one year without human intervention
Flexible deployment scenarios	The Resource Registry components can be deployed in several ways, to best fit the needs of the infrastructure or a specific community

Architecture

The design of the Resource Registry supports distribution and replication wherever it is possible while abstracting clients from the deployment scenario. It exploits HAProxy for proxing requests to the deployed instances of the Resource Registry web service. **HAProxy** is a free, very fast and reliable solution offering high availability and load balancing for very high traffic web applications. Over the years it has become the de-facto standard open-source load balancer and it is now shipped with most mainstream Linux distributions. For these reasons, it is deployed by default in the D4Science Cloud Infrastructure.

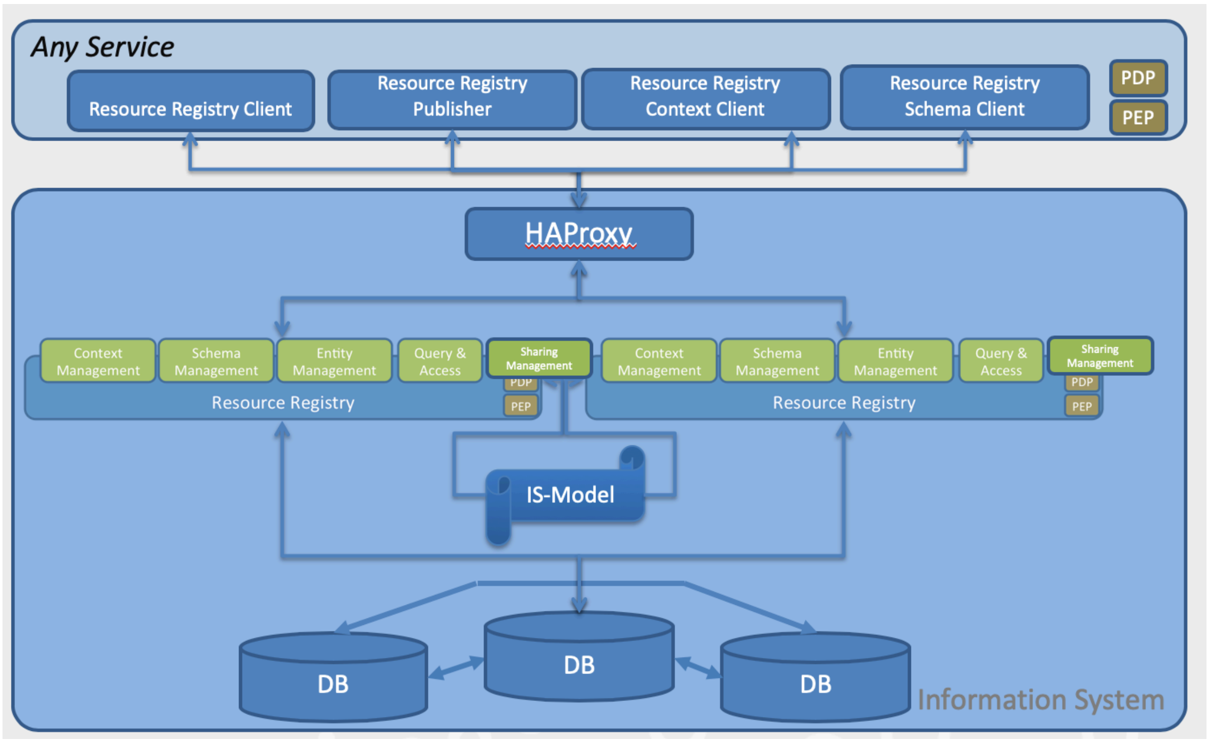


Figure 3.5: Resource Registry Architecture

The Resource Registry web service has now 5 port-types, each responsible for:

- **Context Management**: manage hierarchical contexts;
- **Types Management**: manages the definition of entities and relations types and their schema. This choice allows for easy extension and support modification to the resource model. This is the key factor for the sustainability of the service and infrastructure that have to last for several years
- **Instances Management**: manage instances of registered Entity and Relation type;
- **Sharing Management**: manages instances sharing across different contexts;
- **Query & Access**: query instances and get the schema definition of registered types.

Every port-type is exposed with a REST APIs. Query & Access is the only port-type that can be used by every client and it is documented at: https://dev.d4science.org/swagger/registry/. The other four port-types require a specific authorization that are assigned only to the Resource Manager.

The Resource Registry web service is stateless making possible to replicate it horizontally.

3.3.3.2 Resource Manager

The Resource Manager is responsible for providing Resources compliant with the gCube-Model. In fact, this service is the only one entitled to perform operations on the Resource Registry. It does so by exposing three port types:

1. Context Management enables Resource Registry context management by checking if the requester has the proper role/rights to do the requested action.
2. Schema Management enables schema management on Resource Registry by checking if the requester has the proper role/rights to do it;
3. Resource Management: enables to manage Resource instances by checking if:
    - the requester has the proper role/rights to do the requested action;
    - the action can be performed looking at the policies attached to the entities and relation instances;
    - the action involves other entities or relations.

When all these checks are performed, and if and only if the action is feasible, the Resource Manager translates the incoming request in one or more outgoing requests to the Resource Registry service.

Key Features

Resource Publication, Access and Discovery	The Resource Manager offers Java and WEB APIs to register new new resource types and instances.
Consistency with the gCube Model	The Resource Registry grants publication and access to resources compliant with the gCube Model at Resource level.

Architecture

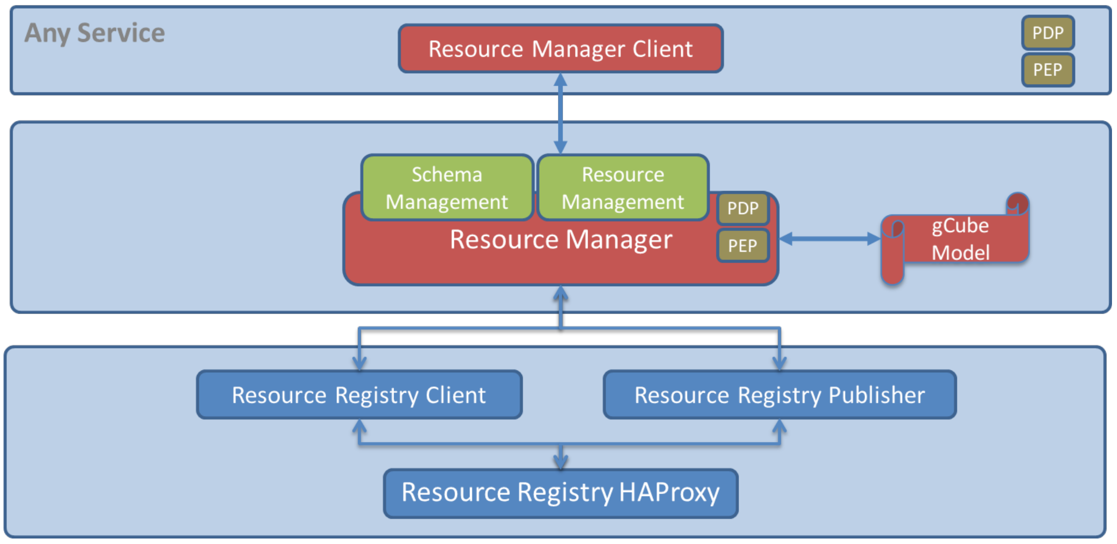


Figure 3.6: Resource Manager Architecture

As depicted in Figure 3.6: Resource Manager Architecture, the Resource Manager uses the Resource Registry Client to query the Resource registry and get the actual knowledge of the infrastructure.

When the Resource Manager receives a request, once performed the proper checks, it uses the Resource Registry publisher to make it effective.

Both Resource Registry Client and Publisher interact with one of the instances of Resource Registry through HA-Proxy.

3.3.3.3 Virtual Research Environment Manager

VRE Manager is responsible for providing context guarantees based on the gCube-Model.

The VRE Manager operates on the D4Science Cloud Infrastructure by using components of:

- the enabling technologies such as Resource Manager;
- supporting technologies such as Provisioning System.

The VRE Manager contacts the Resource Registry to get a current view of the infrastructure; uses the provisioning system to deploy/undeploy services and data; asks the Resource Manager to update the infrastructure state consistently.

Key Features

Context Management	The VRE Manager offers Java and WEB APIs to create, edit, and delete security context, i.e. Virtual Research Environment.
Consistency with the gCube Model	The VRE Manager grants publication and access to resources compliant with the gCube Model at context level.

Architecture

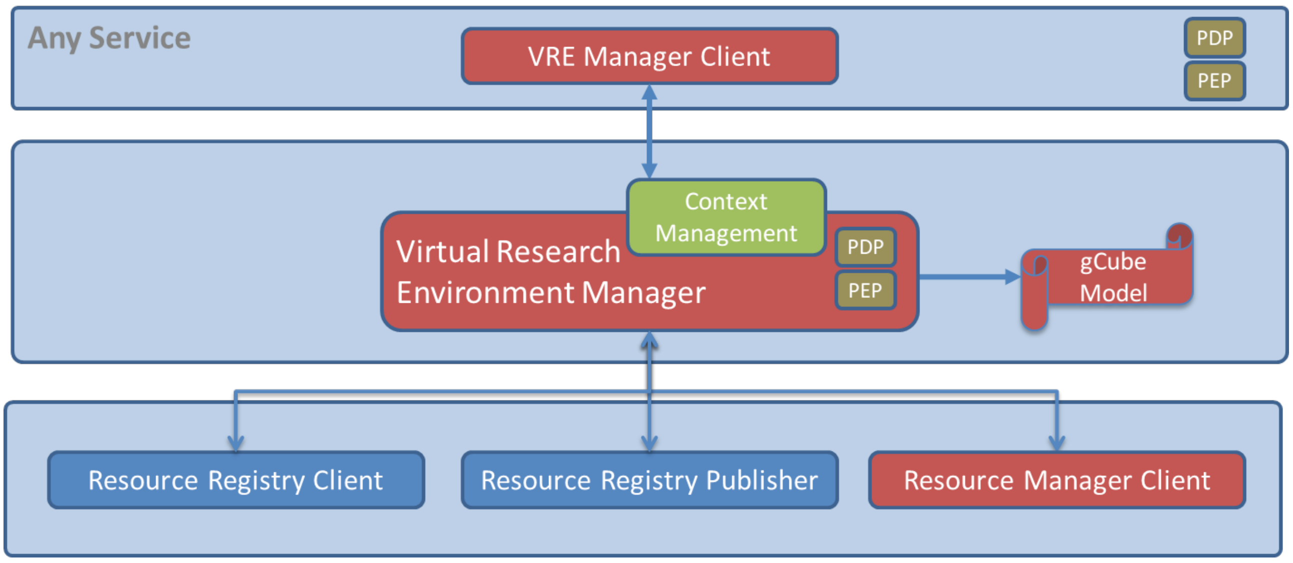


Figure 3.7: VRE Manager Architecture

As shown in Figure 3.7: VRE Manager Architecture, the VRE Manager uses the Resource Registry Client to query the Resource registry and get the actual knowledge of the infrastructure resources. By exploiting this information, the VRE Manager provides the support for the creation of VRE. It creates a new security context and registers it in the Resource Registry. Then it creates a secure symmetric key to enable encrypted conversion in the newly created security context. Finally, it interacts with the Resource Manager to allocate infrastructure resources to the newly created security context.

During the VRE lifetime, when the VRE Manager receives requests for VRE modifications, once performed the proper checks, it interacts with the Resource Manager to either edit, modify, or delete a Virtual Research Environment.

3.3.3.4 Authentication and Authorization

The goal of the Policy-oriented Security Facilities is to protect D4Science Cloud Infrastructure resources from unauthorized accesses.

Service Oriented Authorization and Authentication is a security framework providing ''security services'' as web services, according to ''Security as a Service'' ('''SecaaS''') research topic. It is based on standard protocols and technologies, providing:

- an open and extensible architecture
- interoperability with external infrastructures and domains, obtaining, if required, also so-called ''Identity Federation''
- total isolation from the enabling framework and technologies: zero dependencies in both the directions

The Policy-oriented Security Facilities are powered by the gCube Authorization framework. The **gCube Authorization framework** is a token-based authorization system. The token is a string generated on request by the Authorization service for identification purposes and associated with every entity interacting with the infrastructure (users or services).

The token is passed in every call and is automatically propagated in the lower layers.

The token can be passed to a service in 3 ways:

- using the HTTP-header: adding the value ("gcube-token","{your-token}") to the header parameters
- using the query-string: adding gcube-token={your-token} to the existing query-string
- logging via the default authentication widget showed by the browser using your username as username and your token as password.

The personal token can be retrieved using the token widget deployed on every environment of the portal.

This framework is compliant with the Attribute-based access control (ABAC) that defines an access control paradigm whereby access rights are granted to users through the use of policies which combine attributes together.

ABAC defines access control based on attributes that describe:

- the requesting entity (either the user or the service),
- the targeted resource (either the service or the resource),

- the desired action (read, write, delete, execute),
- and environmental or contextual information (either the VRE or the VO where the operation is executed).

ABAC is a logical access control model that is distinguishable because it controls access to objects by evaluating rules against the attributes of the entities (requesting entity or target resource) actions and the environment relevant to a request. ABAC relies upon the evaluation of attributes of the requesting entity, attributes of the targeted resource, environment conditions, and a formal relationship or access control rule defining the allowable operations for entity-resource attribute and environment condition combinations.

The Authorization framework is compliant with the XACML reference architecture. XACML is the OASIS standard for fine-grained authorization management based on the concept of Attribute-based access control (ABAC), where access control decisions are made based on attributes associated with relevant entities while operating in a given operational context, a natural evolution from Role Based Access Control (RBAC).

Key Features

Security as a Service	Authentication and Authorization provided by web services called by resource management modules
Flexible authentication model	The user is not requested to have personal digital certificates
Attribute-based Access Control	A generic way to manage access: access control decisions are based on one or more *attributes*
Support to different categories of attributes	User related attributes (e.g. roles) and environment related attributes (e.g. context)
Modularity	Composed by different modules: each module has a well-defined scope and provides well-defined services
Support to standards	All the operations delivered by the facilities are built atop of recognized standards
High performance	The design and architectural choices have been made paying great attention to performances
Resource Usage Tracking	Administrators and users can monitor applications resources usage

Architecture

The XACML standard proposes a reference architecture with commonly accepted names for the various entities involved in the architecture. The nomenclature is not new (SAML uses similar names to describe entities in its ecosystem), nor is the architecture complicated, allowing for easier common base of understanding of the standard. Five modules compose it:

1. Policy Administration Point (PAP) - Point which manages access authorization policies

2. Policy Decision Point (PDP) - Point which evaluates access requests against authorization policies before issuing access decisions
3. Policy Enforcement Point (PEP) - Point which intercepts user's access request to a resource, makes a decision request to the PDP to obtain the access decision (i.e. access to the resource is approved or rejected), and acts on the received decision
4. Policy Information Point (PIP) - The system entity that acts as a source of attribute values (i.e. a resource, subject, environment)
5. Policy Retrieval Point (PRP) - Point where the XACML access authorization policies are stored, typically a database or the filesystem.

The 5 modules' capabilities are implemented by gCube as follow.

- Policy Administration Point (PAP) is implemented by the gCube Authorization Service
- Policy Decision Point (PDP) is implemented by a PDP library distributed with gCube SmartGears
- Policy Enforcement Point (PEP) is implemented by a PEP library distributed with gCube SmartGears
- Policy Information Point (PIP) is implemented by the gCube Resource Registry (Information System)
- Policy Retrieval Point (PRP) is implemented by a database controlled exclusively by the gCube Authorization Service

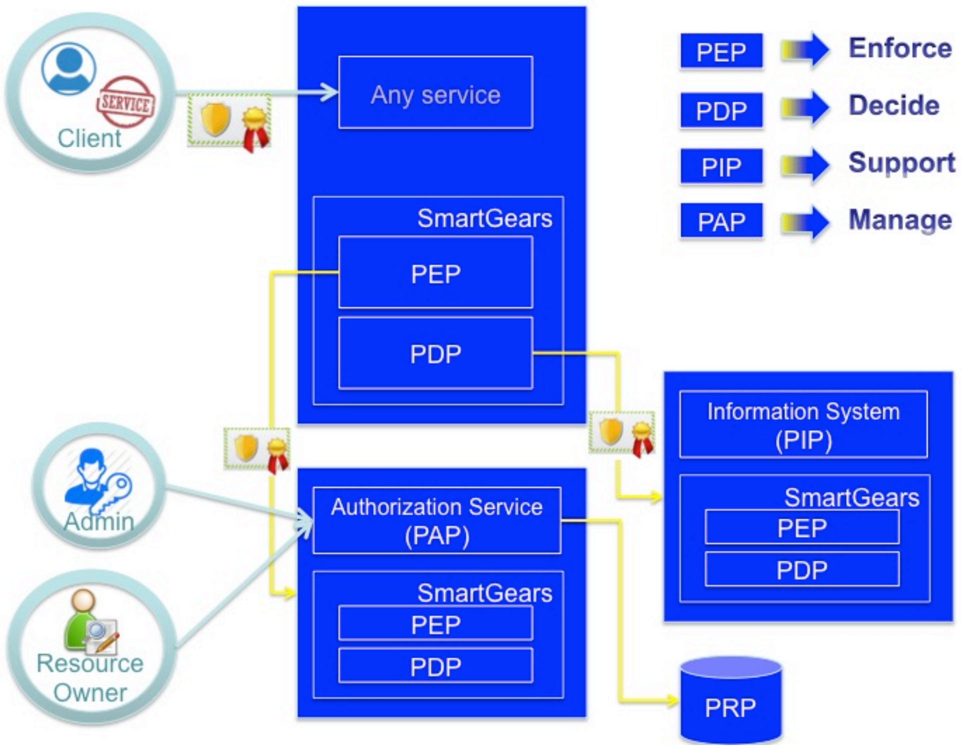


Figure 3.8: Authorization Architecture

The gCube Authorization Framework controls access to applications to allow or prevent the clients to perform various operations in the applications. This is controlled by the Authorization Service with the help of authorization policies. The purpose of authorization policies is to control clients' access. The authorization policies determine at runtime whether or not a particular action is denied.

All the policies are used to DENY to a client an operation in a specific context.

Two types of policy are supported:

- User2Service (U2S)
- Service2Service (S2S)

The U2S policies are used to deny to a user or a role the access to specific service or class of services. The S2S policies are used to deny to a service or a class of services the access to specific service or class of services. To make easier the possibility to allow access only to few clients except restriction clause can be added to the policies. For every policy, a specific ACTION, i.e. Access, Write, Delete, and Execute, can be specified (if supported from the service) otherwise all the ACTION will be denied.

The resource owner uses the policy-authoring tool (GUI) (part of the PAP) to write policies governing access and exploitation of his/her own resources.

The policy administrator then uses the PAP GUI to administer the policies. Please note that policies are not distributed to PDPs upon their creation but at first request referring access/exploitation of a given resource. PDPs use a cache with TTL to avoid the exchange of too many requests.

The PEP intercepts the business level request to access the resource decorated with a token. It resolves the token by sending a request to the PAP and gets back information about the validity of the token to operate in the specific operational context. If the access is denied (invalid token) a Deny Response is immediately issued. If the access is permitted the request to the PAP allows to populate the PDP cache with the appropriate policies. Then it produces a request out of it and sends it to the PDP for actual decision-making.

The PDP, on receiving the request, looks up the policies deployed on it and figures out the ones which are pertinent to the specific request. It may, if necessary, query the PIP for additional attributes that are needed to evaluate the policies. By exploiting the attributes contained in the request, the attributes obtained from the PIP and attributes that are generic to the operational context, the PDP decides whether the request can be allowed (Permit response), denied (Deny response), is not applicable since none of the policies deployed on it can be used to evaluate the request (NotApplicable response) or there was some issue with evaluating the response against the policy, for example due to lack of sufficient attributes available to the PDP (Indeterminate response).

The response is then sent by the PDP to the PEP. The PEP parses the response from the PDP and handles each of the four possible response cases. If either a Permit or a NotApplicable response is getting back then the business request is passed to the service, otherwise a Deny response is issued.

Highlight 1: flow of control governing the authorization flow.

3.3.3.5 Auditing

Auditing is defined as the recording, summarizing, and classifying of service invocations and other events, e.g. storage of data, systematically, in a simpler sense, is the procedure of communicating and translating raw data from the infrastructure operation to its managers and stakeholders.

Key Features

open and extensible Auditing model	the underlying Auditing model is flexible to adapt to diverse provider needs
highly modular and extensible architecture	the entire subsystem comprises a large number of components clearly separating the functional constituents
multiple options for storage	the subsystem can rely on an array of diverse solutions for actually storing records

Architecture

The gCube Auditing architecture is logically divided in four different layers:

- Consumer Layer
- Enabling layer
- Backend layer
- Storage layer

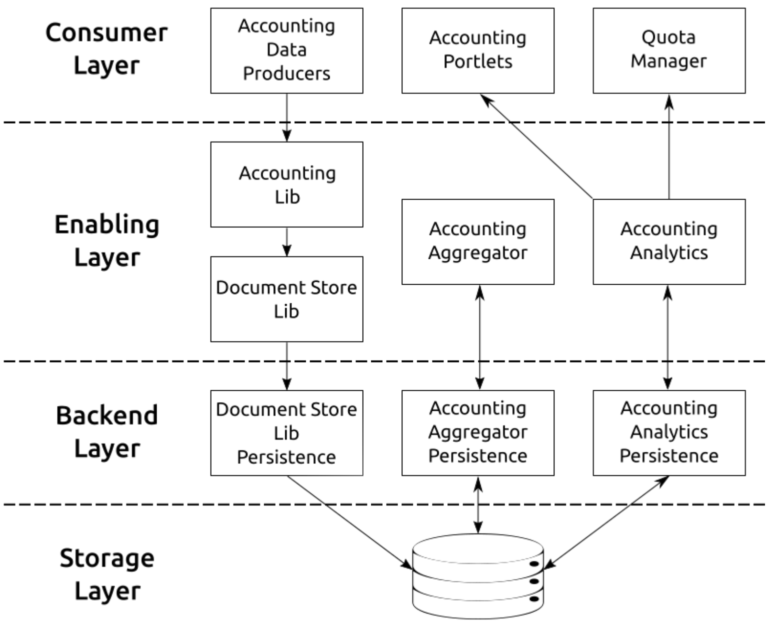


Figure 3.9: Auditing Architecture

All component respects a set of common rules adopted to ensure high-availability, fully-distributed operations, low-operation costs

Each enabling layer has its own correspondent back-end implementation;

Each back-end implementation is dynamically discovered at run-time. This allows to decouple the deployment of a different back-end from the development of the enabling layer. In other word each component on the enabling layer must not have any dependency over a certain back-end implementation.

Auditing Enabling Layer

The **Accounting Lib** collects, harmonizes and stores Auditing data. It is mainly based and developed exploiting the facilities provided by the Document Store Lib.

The **Accounting Analytics** exposes a common access point interface to query the collected Auditing data.

The **Accounting Aggregator** aggregates the collected Auditing data according to dynamically defined policies. The D4Science Auditing policies have been defined to incrementally aggregate past Auditing data without loss of information

Auditing Storage Layer

This layer is not developed by gCube. Rather it relies on technologies guaranteeing HA (High Availability).

In the current settings, it is implemented by relying on **CouchBase**. Other supported backend technologies are CouchDB and MongoDB.

Auditing Backend Layer

Each component in this layer has been explicitly developed over a certain storage technology. They rely on the Resource Registry to discover the information needed to connect to the underling storage. In other words, any component does not have hard-coded connection information or a local configuration file. This approach allows to retrieve the storage connection information by specifying the underlying storage technology and the enabling component to use.

The first filter allows switching to different storage backend at runtime and it supports the co-existence of different storage backend – particularly useful to migrate from one storage type to another without any downtime.

The second filter allows keeping separated the connection information for each component. This allows supporting tailored access policies for each component, e.g. write-only for Auditing-lib connection and read-only for Auditing-analytics.

The **document-store-lib-BACKEND** supports the connection to and the storage of Auditing data to the technology selected as persistence layer. It has been implemented to support the three underlying technologies: document-store-lib-couchdb, document-store-lib-couchbase, document-store-lib-mongodb;

The **Accounting-analytics-persistence-BACKEND** supports the connection to and the discovery and access of Auditing data to the technology selected as persistence layer. It has been implemented to support the three underlying technologies: Auditing-analytics-persistence-couchdb, Auditing-analytics-persistence-couchbase;

The **Accounting-aggregator-persistence-BACKEND** supports the connection to and the aggregation of Auditing data to the technology selected as persistence layer. It has been implemented to support the three underlying technologies: Auditing-aggregator-persistence-couchdb, Auditing-aggregator-persistence-couchbase;

Auditing Consumer Layer

Each component in this layer allows either producing or consuming Auditing information. It does not include only a graphical interface designed for managers, i.e. **Accounting Portlet**. Rather it includes all the components that collect Auditing data as the Quota Manager, currently in development stage.

3.4 D4Science Storage Framework

3.4.1 Overview

The Storage framework is realized by a combination of services and libraries powered by the gCube System open-source project. It is composed by two main systems: File-Based System and Spatial Data Repository System. These acts as main driver for clients that interface the storage resources managed by the system or accessible through facilities available within the system.

The File-Based System supports functions for **standards-based** and **structured access** and **storage of files** of arbitrary size.

The Spatial Data Repository System is composed by a number of different spatial data repositories for storing **spatial data** in different (standard) formats (e.g. NetCDF, vector data, raster data etc.)

3.4.2 Key Features

Standards compliancy	Support for standard communication protocols / interfaces and data / metadata formats.
Economy of scale	Services constituting one aggregative infrastructure may be hosted over servers maintained at different sites
Failover Management	Automatically transfers control to a duplicate computational node when faults or failures are detected
Support of geospatial dataset lifecycle	Support for generation, revision, publishing, access, visualization and sharing of geospatial data.
Support for analysis and processing	Support for high performance operations over datasets
Geo-referencing datasets	Provide analysis tools to create standard spatial representation of datasets

3.4.3 Subsystems

3.4.3.1 File-Based Store System

The File-Based Store system includes services providing clients functions for standards-based and structured access and storage of files of arbitrary size. This is a fundamental requirement for a wide range of system processes, including indexing, transfer, transformation, and presentation. Equally, it is a main driver for clients that interface the resources managed by the D4Science infrastructure or accessible through facilities available within the same infrastructure.

The File-Based System is composed by a service abstracting over the physical storage and capable of mounting several different store implementations, (by default clients can make use of the MongoDB store) presenting a unified interface to the clients and allowing them to download, upload, remove, add and list files or unstructured byte-streams (binary objects). The binary objects must have owners and owners may define access rights to files, allowing private, public, or shared (group-based) access.

All the operations of this service are provided through a standards-based, POSIX-like API which supports the organization and operations normally associated with local file systems whilst offering scalable and fault-tolerant remote storage

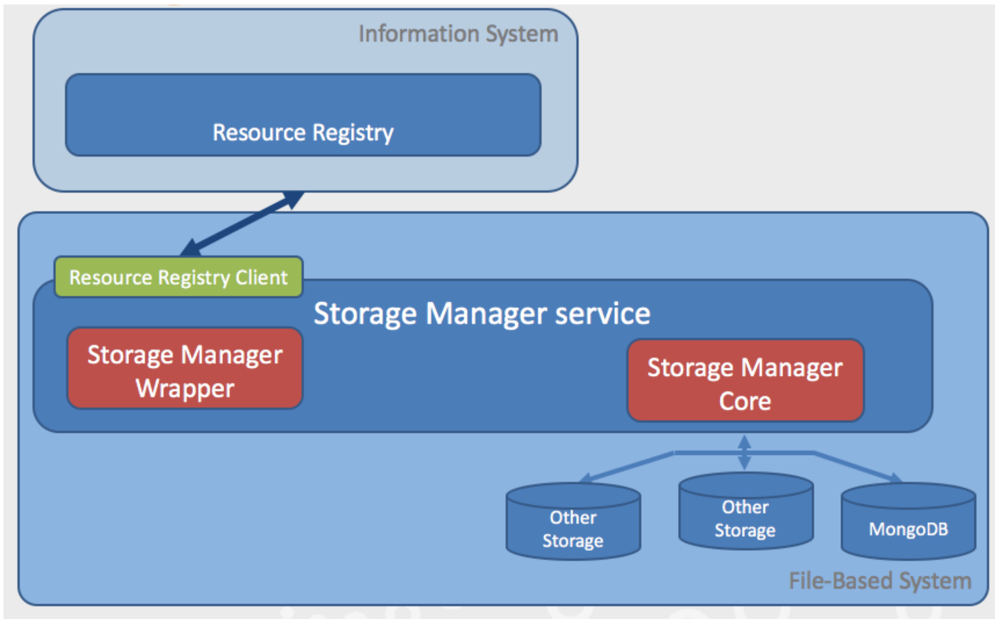


Figure 3.10: File-Based System Architecture

As shown in Figure 3.10: File-Based System Architecture the core of the Storage Manager service is a software component named Storage Manager Core that offers APIs allowing to abstracts over the physical storage. The Storage Manager Wrapper instead is a software component used to discover back-end information from the Resource Registry service of the D4Science Infrastructure. The separation between these two components is necessary to allow the usage of the service in different contexts other than the D4Science Infrastructure.

3.4.3.2 Spatial Data Repositories

The Spatial Data Repositories include services, technology, policies and practices designed in order to provide the following features:

- **Data Discovery**: The ability to browse, query and access metadata files about accessible geospatial datasets. This feature is obtained exploiting **GeoNetwork** webservice, the Open Source catalogue for geospatial metadata compliant with standards mandated by Open Geospatial Consortium (OGC).
- **External Repository Federation**: Transparently extend the Data Discovery capabilities by including output from registered external catalogues and repositories in order to give users global result from a single point.
- **Data Access & Storage:** Provide users and applications to access/store geospatial data in standard formats. Due to the heterogeneity of spatial data representation and formats, the following technologies have been adopted:
    - **PostGIS**: Geospatial extension of **PostgreSQL** relational DBMS;
    - **GeoServer**: Open Source application for management and dissemination of geospatial data through standards mandated by OGC;
    - **Thredds Data Server**: Unidata's Thematic Real-time Environmental Distributed Data Services (THREDDS) is a web server that provides data access for scientific geospatial dataset formats (i.e. NetCDF).
- **Data Processing**: The Data Processing framework includes services designed to perform analysis and transformations over geospatial datasets. The adoption of **52°North Web Processing Service** (WPS) grants users a standardized way to interact with Data Processing facilities.
- **Data Visualization**: Web application, named GeoExplorer, designed to render views of overlapped geospatial datasets on a specific Earth projection, with the ability to query / inspect and export selected data and rendered images.

The set of spatial data repositories and the comprehensive set of integrated technologies for their management, discovery, and exploitation is fully integrated with both infrastructure's enabling technology and layers in order to exploits provisioning, monitoring, Auditing, authentication and storage facilities of the infrastructure.

Key Features

Support of geospatial dataset lifecycle	Support for generation, revision, publishing, access, visualization and sharing of geospatial data.
Support for analysis and processing	Support for high performance operations over geospatial datasets
Dataset enrichment and harmonization	Provide tools to harmonize and add information on existing data
Georeferencing datasets	Provide analysis tools to create standard spatial representation of datasets

Standards compliancy	Support for standard communication protocols / interfaces and data / metadata formats.
External repository federation	Gather all available information in one single point.
Policies adoption assurance over third-party technologies	Configuration/orchestration of third party technologies in order to ensure access policy compliancy.
Horizontal scalability	Ability to expand / contract the SDI resource pool in order to accommodate heavier or lighter loads.

Architecture

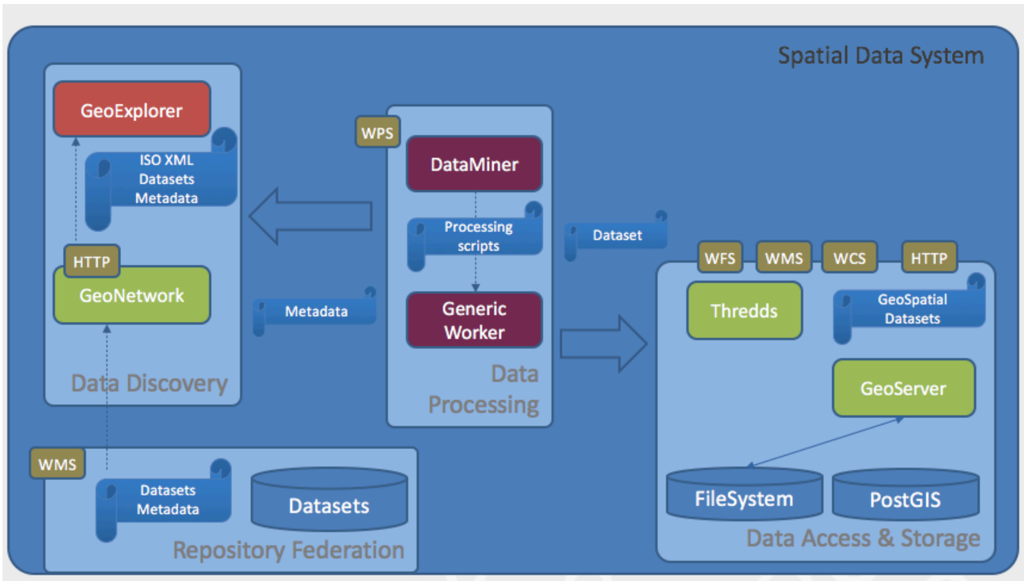


Figure 3.11: Spatial Data Repositories Architecture

The set of Spatial Data Repositories technologies selected and integrated are not only fully compliant with the Open Geospatial Consortium (OGC) standards, i.e. Web Map Service (WMS), Web Coverage Service (WCS), and Web Feature Service (WFS). Rather specific mediators and validators have been designed and implemented to respect the INSPIRE Directive, the EU initiative geared to help to make spatial or geographical information more accessible and interoperable for a wide range of purposes supporting sustainable development.

The Data Discovery components allows the discovery at runtime all available datasets independently of their locations and it is indifferent to the technology used to persist them. It also indifferent to the fact that the data are maintained by a repository managed by the D4Science Cloud Infrastructure or by an independent provider. The same applies also to the Data Processing components that first discovery the datasets to process and then are able to process them independently of the technology used to persist them and the provider entitled to manage them.

3.5 D4Science Analytics Framework

3.5.1 Overview

The Analytics Framework includes a set of features, services and methods for performing data processing and mining on information sets. These features face several aspects of data processing ranging from modeling to clustering, from identification of anomalies to detection of hidden series. This set of services and libraries is used by the e-infrastructure to manage data mining problems even from a computational complexity point of view. Algorithms are executed in parallel and possibly distributed fashion, using the same e-infrastructure nodes as working nodes. Furthermore, Services performing Data Mining operations are deployed according to a distributed architecture, in order to balance the load of those procedures requiring local resources.

3.5.2 Key Features

Parallel Processing	Support for the execution of algorithms on multi-cores computational nodes
Distributed Processing	Transparent distribution of the execution on sets of computational nodes
Failover Management	Automatically transfers control to a duplicate computational node when faults or failures are detected
State-of-the-art data mining algorithms	General purpose algorithms (e.g. Clustering, Principal Component Analysis, Artificial Neural Networks, Maximum Entropy, etc.) supplied as-a-service
Data trends generation and analysis	Identification of trends; inspection of time series to automatically identify anomalies; basic signal processing techniques to explore periodicities in trends

3.5.3 Subsystems

3.5.3.1 Data Miner System

The Data Miner System's goal is to offer a unique access for performing data mining or statistical operations on heterogeneous data. These data can reside on the client side in the form of CSV files or they can be remotely hosted, as SDMX documents or, furthermore, they can be stored in a database.

The Data Miner System is composed by a service, namely Data Miner service, able to take such inputs and execute the operation requested by a client interface, by invoking the most suited computational infrastructure, choosing among a set of available possibilities: executions can run on multi-core machines, or on different computational infrastructures, like Windows Azure, CompSs and other options.

Algorithms are implemented as plug-ins which makes the injection mechanism of new functionalities easy to deploy.

Key Features

Openness	Interaction with external software supporting Standard protocols
Parallel Processing	Support for the execution of algorithms on multi-cores computational nodes
Distributed Processing	Transparent distribution of the execution on sets of computational nodes
State-of-the-art data mining algorithms	General purpose algorithms (e.g. Clustering, Principal Component Analysis, Artificial Neural Networks, Maximum Entropy, etc.) supplied as-a-service

Architecture

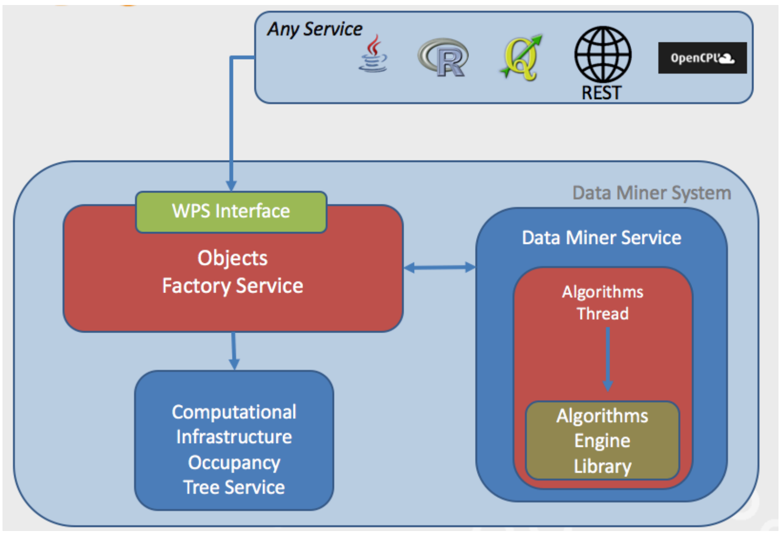


Figure 3.12: Data Miner System Architecture

According to Figure 3.12: Data Miner System Architecture, the Data Miner System comprises the following components:

- **Computational Infrastructure Occupancy Tree service**: a service which monitors the occupancy of the resources to choose among when launching an algorithm;
- **Data Miner Service**: a service executing all the computations asked by a single user\service. It is composed by two components:
    1. **Algorithms Thread**: an internal process which puts in connection the algorithm to execute with the most unloaded infrastructure resource which is able to execute it. Infrastructures are weighted even according to the computational speed; the internal logic will choose the fastest available;
    2. **Algorithms Engine Library**: a container for several data mining algorithms as well as evaluation procedures for the quality assessment of the modeling procedures. Algorithms follow a plug-in implementation and deploy;

- **Object Factory Service**: a service acting as a broker for Data Miner Service and a link between the users' computations and the Occupancy Tree service;

It is worth to notice that thanks to the support of HTTP-REST and WPS protocols the Data Miner System is capable of interacting with different external software supporting such standard (e.g. QGIS, OpenCPU) and different programming languages, in particular Javascript, R, Java.

3.5.3.2 Smart Executor System

The SmartExecutor service allows to execute "gCube Tasks" and monitor their execution status. Each instance of the SmartExecutor service can run the "gCube Tasks" related to the plugins available on such an instance.

Each instance of the SmartExecutor service publishes descriptive information about the co-deployed plugins.

Key Features

Repetitive Tasks	Task can be scheduled to be periodically repeated.
Tasks take over	Task can be taken in charge from new instances in case of instance failure or instance overload.

Architecture

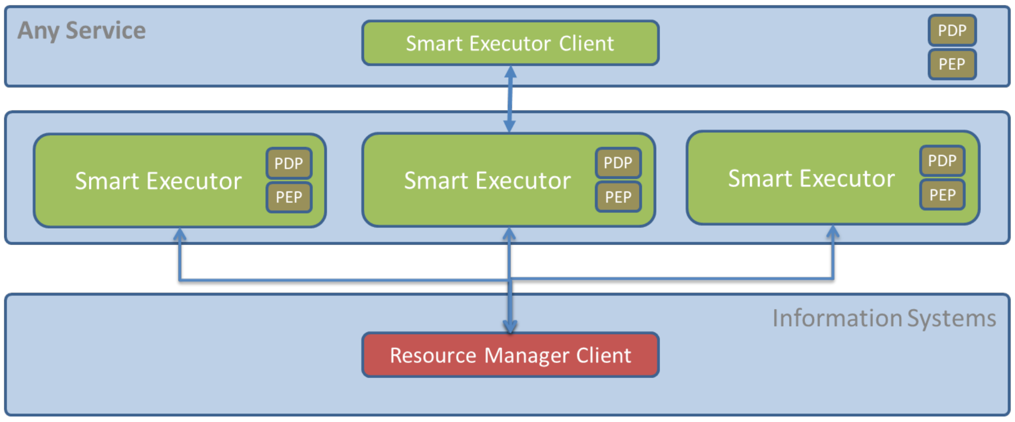


Figure 3.13: Smart Executor System Architecture

Clients may interact with the SmartExecutor service through a library (SmartExecutor Client) of high-level facilities to simplify the discovery of available plugins in those instances. Each client can request to execute a "gCube Tasks" or getting information about the state of their execution.

The SmartExecutor service allows tasks execution through the use of co-deployed plugins. The service allows to pass inputs parameter to the plugin requested to run.

The execution is invoked every time it matches the scheduling parameters. The way to schedule the plugin execution is indicated by scheduling parameter. There are two different way to schedule an execution:

- **run and die**: the plugin is launched just for one time and after this execution it won't be repeated;
- **scheduled**: the plugin repeats its execution over time according to a delay interval or to a “cron” expression.

SmartExecutor instances could take care of a scheduled run when the node where it was previously allocated crashes or is overloaded. To achieve this goal a scheduled task description is registered in the Information System through the Resource Manager.

3.6 D4Science Collaborative Framework

3.6.1 Overview

The Collaborative Framework is realized through a combination of software components (services and libraries) powered by the gCube System. Three main subsystems characterise the Collaborative Framework:

- Social Networking System
- Shared workspace System
- User Management System

These systems provide consumers with a homogenous abstraction layer over different external technologies enabling to operation of the framework. The external technologies involved comprises, Apache Cassandra, Apache Jackrabbit, Elastic Search, MongoDB, and Liferay Portal. In particular, the Social Networking System exploits an Apache Cassandra cluster and an Elastic Search cluster, the Shared Workspace System exploits an Apache Jackrabbit repository (metadata) and a MongoDB cluster (payload) for its backend, the User Management System exploits Liferay Portal for its backend and to allow users to login for personalized services or views.

3.6.2 Key Features

Collaboration	You can share posts and have multiple discussions on the VRE homepage, adding comments and files in line with the discussion.
Folder Sharing	Folder sharing enables the reuse of content and the ease of creating multiple VREs for different audiences with shared content.
Custom Notifications	Important and personalized alerts appear in each user’s notification area, and custom applications can add their own notifications.
Responsive Design support	Web Applications are based on Twitter Bootstrap, making it possible to create responsive pages that look great regardless of device.
Economy of scale	Services constituting one aggregative infrastructure may be hosted over servers maintained at different sites

3.6.3 Subsystems

3.6.3.1 Social Networking System

Social Networking System comprising services conceptually close to the common ones promoted by social networks – e.g., posting news, commenting on posted news, likes, private messages and notifications; It is composed by 2 main services, the Social Networking Service and the Social Indexer Service.

The Social Networking Service logic relies on the Social Networking Model, this Model is used also for the efficient storage of the Social Networking Data (Posts, Comments, Notifications etc.) in the underlying Apache Cassandra Cluster. This Cluster is queried by means of a Java client.

Architecture

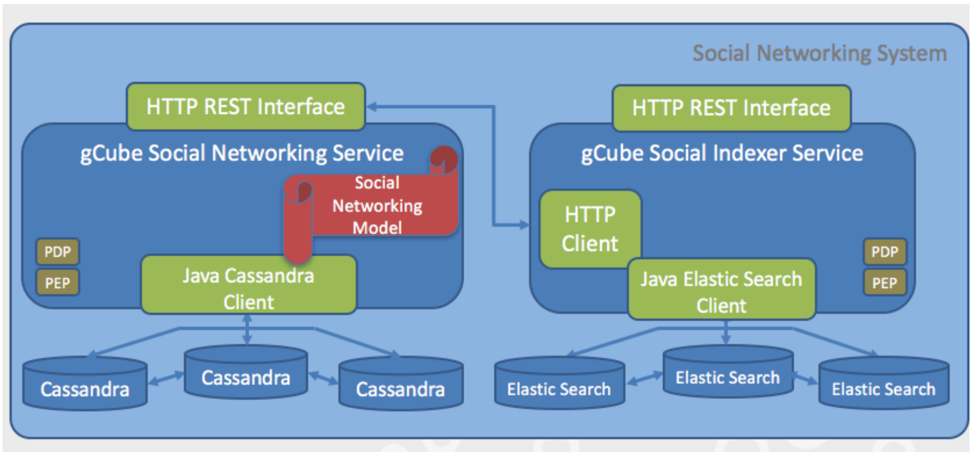


Figure 3.14 Social Networking System Architecture

The Social Networking Service exposes an HTTP REST Interface for the external, and non, services of the infrastructure. The Social Indexer Service uses such interface for the retrieval of the Social Networking Data to index by means of an Elastic Search Cluster. The Social Indexer Service exposes an HTTP REST Interface for the external, and non, services of the infrastructure needing to perform search operations over the Social Networking Data.

Both Services rely on the Policy Decision Point (PDP) and the Policy Enforcement Point (PEP) to intercepts user's access request and evaluate these requests against authorization policies of the Authorization System of the Infrastructure.

3.6.3.2 Shared Workspace System

The Shared Workspace System provides a remote (Cloud) folder-based file system, supporting sharing of folders and different item types (ranging from binary files to information objects representing, for instance, tabular data, workflows, distribution maps, statistical algorithms).

Architecture

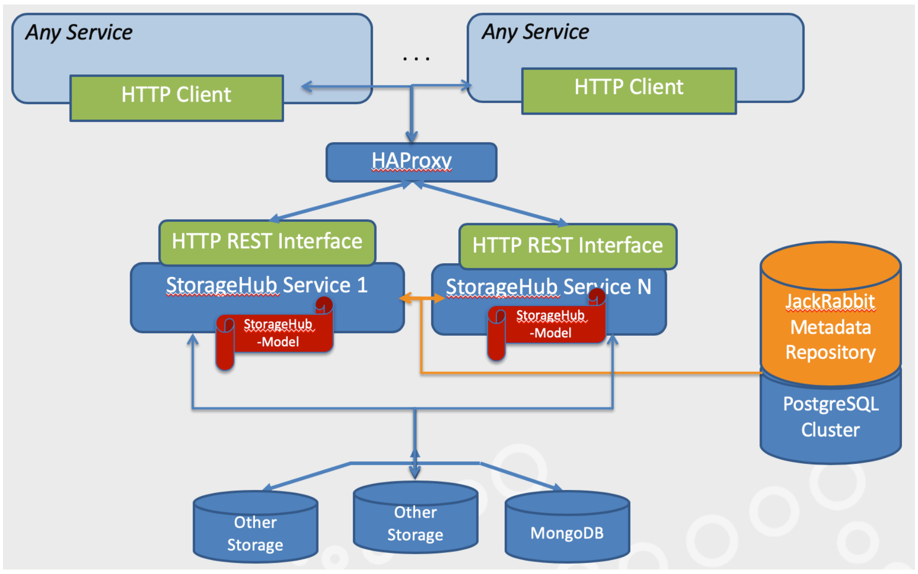


Figure 3.15: Shared Workspace System Architecture

With respect to the last reporting period the Shared Workspace System was heavily redesigned with the aim of increasing its scalability and overall performance. It consists of one gCube service, named StorageHub Service, relying on 2 different storage technologies to store the metadata of the items being stored, namely Apache Jackrabbit as metadata repository and PostgreSQL as Apache Jackrabbit Bach-end Database. The StorageHub Service is replicable and a HAProxy on top is used for proxying requests to the deployed instances of it. One other distinguish feature of the StorageHub Service is that the actual payload of the items can be stored on a number of in house and commercial storage technologies, for instance in a MongoDB Cluster, but also on other types, including Cloud Storages solutions (e.g. Amazon S3).

The StorageHub Service identifies a core set of capabilities to work on JackRabbit content. Together with its model, named StorageHub model, exposes content in the content repository as HTTP resources, fostering a RESTful style of application architecture. Home RESTFUL interface processes HTTP requests coming from clients. The following operations are supported:

- retrieve content;
- create content;
- modify existing content;
- remove existing content;
- move existing content to a new location;
- copy existing content to a new location;

3.6.3.3 User Management System

Users are the fundamental entity managed by this System. As a matter of fact, User is an entity that can sign into the portal and do something. Users are assigned a Role and a Role is what defines the user’s privileges. The User Management System provides functionality to manage personal profiles and users in the D4Science Virtual Research Environments, supporting user groups (for the purpose of group specific privileges) and roles for application specific needs related to the user’s role within a given VRE.

3.7 D4Science Third-Party Frameworks

3.7.1 NLPHub

NLPHub is a distributed system that orchestrates and combines several state-of-the-art text mining services that recognize spatiotemporal events, keywords, and a large set of named entities. NLPHub integrates: (i) the Stanford CoreNLP software for English, German, French, and Spanish; (ii) the Italian NLP Tool (Tint) for Italian; (iii) Apache OpenNLP toolkit that includes methods for language detection, tokenization, part-of-speech tagging, morphological parsing, and named entity recognition; (iv) ItaliaNLP that includes methods for part-of-speech tagging, tokenization, morphological parsing, lemmatization, named entity recognition, clustering, words similarity assessment, and sentiment analysis; (v) NewsReader event recognizer; (vi) Keywords NEW that produces tag clouds of words and nouns. For all of them the access is offered as-a-service.

An orchestrator algorithm (AMERGE), implemented as a DataMiner process, receives a user-provided input text, along with the indication of the text language (optionally), and a set of annotations to extract (selected among those supported for that language).

NLPHub is exploitable with anonymous access only via its graphical user interface. By exploiting the user interface, a user may upload a text file and obtain an annotated text. Programmatic exploitation of the NLPHub is possible only by joining a VRE offering it as a service.

3.7.2 Share Latex

ShareLaTeX is an online LaTeX editor that allows real-time collaboration and online compiling of projects to PDF format. In comparison to other LaTeX editors, ShareLaTeX is a server-based application, which is accessed through a web browser. On July 20, 2017, ShareLaTeX was acquired by Overleaf. Overleaf plans to continue ShareLaTeX under the brand Overleaf v2 which was in beta testing up until the 4th of September 2018. The provision of the ShareLaTeX service is therefore limited to the availability of the open source license and it will be guaranteed by D4Science.org until Overleaf will not change it.

3.7.3 RShiny Application

The Shiny application is delivered by the user to the D4Science.org as a complete Shiny Application on the Docker Registry[4].

The user is responsible to build the standard Docker image as described in the ShinyProxy documentation by writing the docker file, building the docker image and providing the configuration for ShinyProxy.

[4] https://hub.docker.com

D4Science.org makes accessible the Shiny Application on the D4Science infrastructure by ensuring its operation and orchestration in a cluster. The cluster is composed by multiple Docker hosts which run in parallel mode and act as managers and workers.

3.7.4 RStudio Application

The RStudio allows to perform online statistical analyses with the R software for statistical computing. The user is responsible to implement proper R scripts as described in the RStudio documentation.

The D4Science Infrastructure makes accessible the RStudio Application on the D4Science infrastructure by ensuring its operation and orchestration in a cluster. The cluster is composed by multiple hosts, each of which is assigned in exclusive mode to a user for an entire online session. At the end of the session, all the content stored in that host may be removed by D4Science Infrastructure. All the data, scripts, and other resources that the user needs to persist have to be stored into the Workspace that is accessible through the RStudio Application.

3.8 D4Science Software Methods Provision and Integration

The data processing platform (named DataMiner) is an open-source computational system based on the gCube system. This platform is fully integrated with the D4Science e-Infrastructure, and has been conceived to meet the Open Science paradigm requirements, thus to promote collaborative experimentation and open publication of scientific findings, while tackling Big Data challenges. DataMiner is able to interoperate with the services of the D4Science e-Infrastructure, and uses the Web Processing Service (WPS) standard to publish the hosted processes. Further, it saves the computational provenance of an executed experiment using the Prov-O standard. DataMiner implements a Cloud computing Map-Reduce approach and is able to process Big Data and save outputs onto a collaborative experimentation space. This space allows users to share computational information with other colleagues. This service was conceived to execute processes provided by communities of practice in several domains, leveraging integration effort at the same time. The DataMiner deployment is fully automatic through ANSIBLE scripts and is spread across different machines providers, including the Italian Garr network.

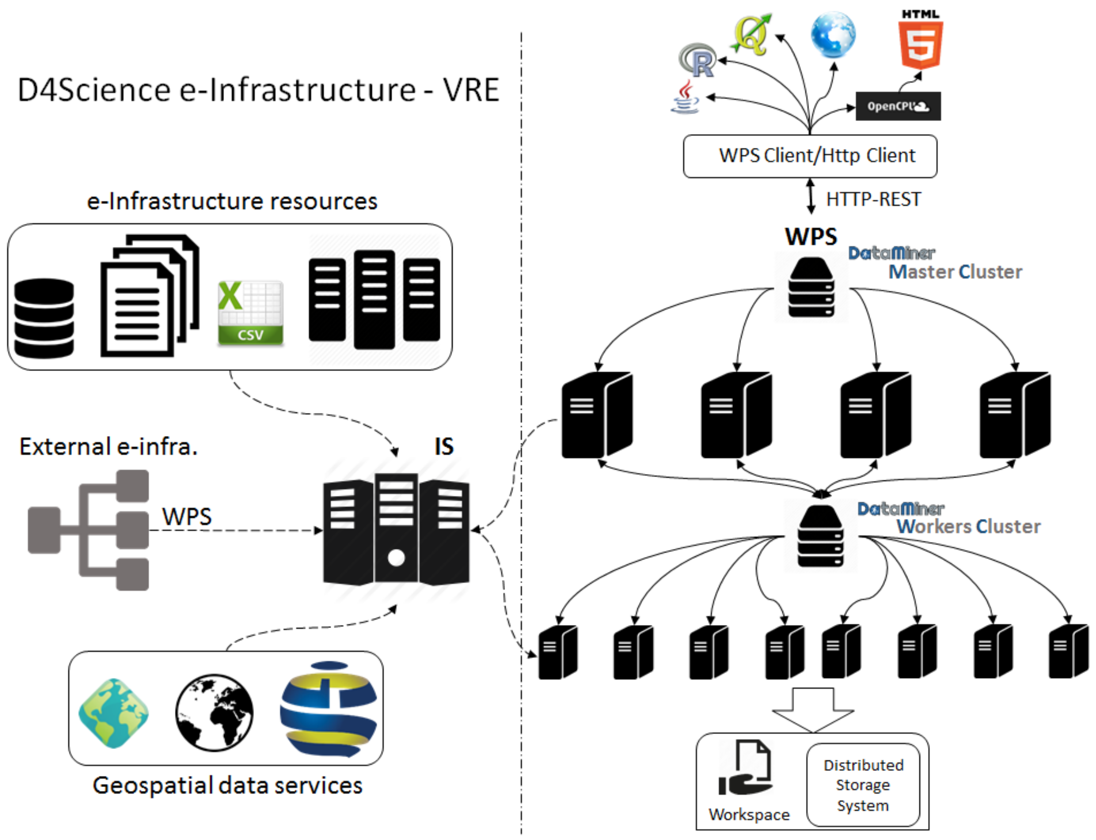


Figure 3.1: Architecture of the DataMiner data processing system

The DataMiner (DM) [2] architecture is made up of two sets of machines (clusters) that operate in a Virtual Research Environment [3]: Master and Worker cluster. In a typical deployment scenario, the Master cluster is made up of a number of powerful machines (e.g. Ubuntu 18 x86 64 with 16 virtual CPUs, 32 GB of random-access memory, 100 GB of disk) managed by a load balancer that distributes the requests uniformly to the machines. Each machine is endowed with a DM service that communicates with the D4Science Information System (IS), i.e. the central registry of the e-Infrastructure resources, to notify its presence and capabilities. The balancer is indexed on the IS and is the main access point to interact with the DMs. The machines of the Worker cluster have the same computational power and serve Map-Reduce computations. DM is based on the 52North WPS implementation, but extends it to meet D4Science e-Infrastructure requirements. It is developed with Java and the Web service runs on an Apache Tomcat instance endowed with gCube system libraries. Further, it offers a development framework to integrate new algorithms and interact with the e-infrastructure.

Using the WPS standard in a Cloud computing system allows a number of thin clients to use the processes. The DataMiner services use the security services of the D4Science e-Infrastructure and require a user token to be provided for each operation. This token is passed via basic HTTPS-access authentication, which is supported by most WPS and HTTP(S) clients. The token identifies both a user and a Virtual Research Environment and this information is used by DM to query the IS about the capabilities to be offered in that VRE, i.e. the processes the user will be able to invoke with that authorization.

The DataMiner computations can take inputs from the D4Science Workspace. Inputs can also come from Workspace folders shared among several users. This fosters collaborative experimentation already at the input selection phase. Inputs can also come from external repositories, because a file can be provided either as a HTTP link or embedded in a WPS execution request. The outputs of the computations are written onto the D4Science Distributed Storage System and are immediately returned to a client at the end of the computation. Afterwards, an independent thread also writes this information on the Workspace. Indeed, after a completed computation, a Workspace folder is created which contains the input, the output, the parameters of the computation, and a provenance document summarizing this information. This folder can be shared with other people and used to execute the process again. Thus, the complete information about the execution can be shared and reused.

DataMiner can also import processes from other WPS services. If a WPS service is indexed on the IS for a certain VRE, its processes descriptions are automatically harvested, imported, and published among the DM capabilities for that VRE. During a computation, DM acts as a bridge towards the external WPS systems. Nevertheless, DM adds provenance management, authorization, and collaborative experimentation to the remote services.

3.8.1 Functional specifications

DataMiner satisfies functional specifications related to the processing of a large variety of data types (including geospatial data) in the wide context of Big Data processing and Open Science. Indeed, several computational systems exist, also used by e-Infrastructures, that typically parallelise the computation on several available cores/processors or machines of the e-Infrastructure. Nevertheless, DataMiner also satisfies new requirements requested by new Science paradigms, which include:

- Publishing local-machine processes, provided by a community of practice (e.g. scripts, compiled programs etc.), as-a-Service;
- Managing several programming languages;
- Interoperate with other services of an e-Infrastructure, possibly through a standard representation of the processes and of their parameters;
- Saving the “provenance” of an executed experiment, i.e. the set of input/output data, parameters, and metadata that would allow to reproduce and repeat the experiment;
- Supporting data and parameters sharing through collaborative experimental spaces;
- Being economically sustainable, e.g. easy to install and deploy on several partners machines;
- Supporting security and Auditing facilities;
- Managing and analysing Big Data;
- Designing and executing Workflows that combine different processes published as services.

DataMiner was conceived to satisfy the reported requirements. One major advantage is that all the DM services publish their capabilities using a standard, which enhances the interoperability with other external services and software, with respect to using custom clients. Further, the DM clusters are managed by fast load balancers that are able to dynamically add machines and to ignore them when offline. Since the Worker nodes are exact replicas of the Master nodes, the Worker cluster can be used directly from clients and fosters alternative usages of the Cloud computing system. For example, external users of D4Science (authorised with proper tokens) may also implement their own Cloud computations by invoking the Worker cluster in custom workflows. Another DM feature is that it can interact with data preparation and harmonisation services. This speeds up the typical time consuming phase of data preparation for an experiment. Further, providing a shared experimentation area allows reusing the results of processes and also fosters multidisciplinary experiments. Users could also be services or external machines (e.g. sensors) that produce experimental data at different frequencies and time scales, while other processes analyse these data and take decisions. The same facilities are automatically offered to desktop software supporting WPS. Further, generating and storing provenance information improves the possibility to repeat and reproduce an experiment executed by other scientists. Finally, since processes and service installation is fully automatic through ANSIBLE scripts, it is easy to deploy DataMiner on a number of machines providers. The hosted processes currently hosted by DataMiner are written with the R, Java, Fortran, Linux-compiled, .Net, Octave, KNIME, and Python programming languages and have been provided by developers with heterogeneous expertise (e.g. biologists, mathematicians, agronomists, physicists, data analysts etc.).

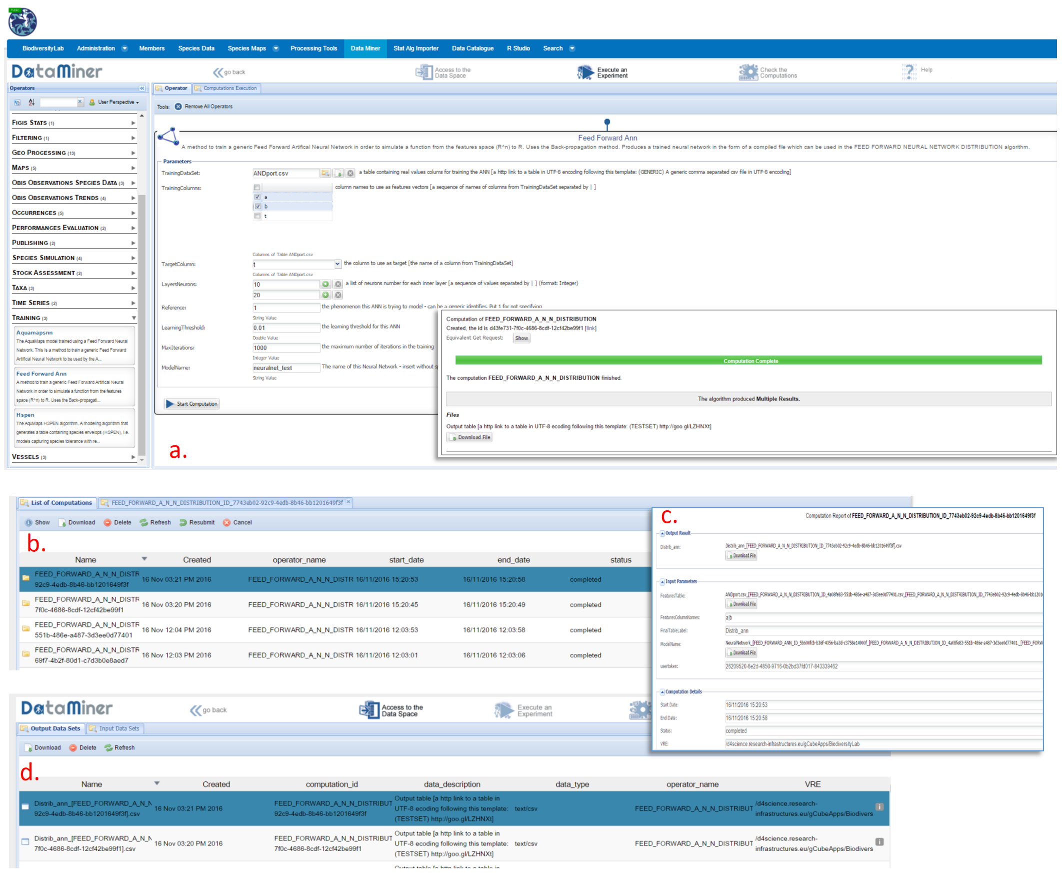
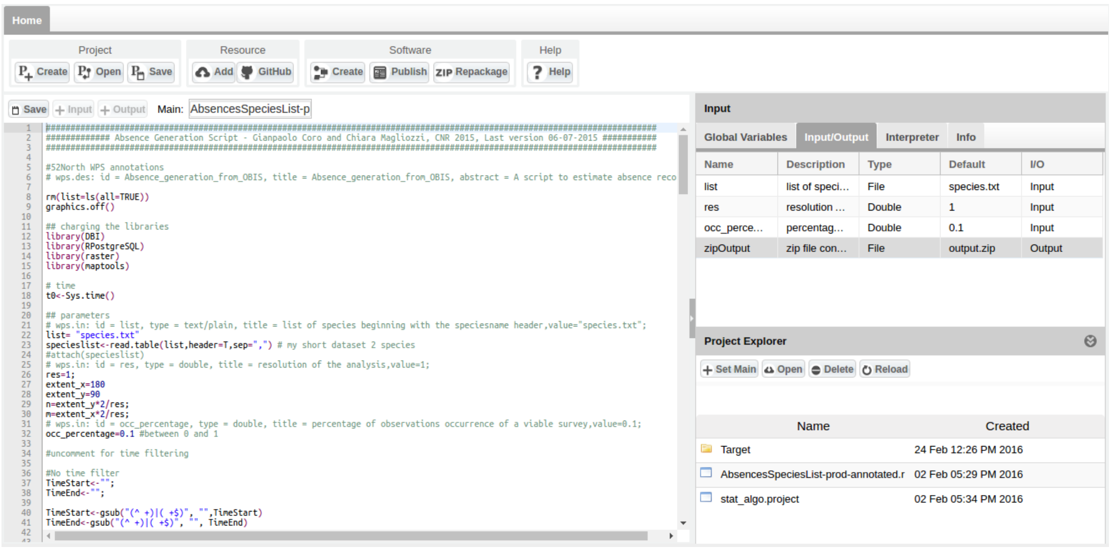


Figure 3.17: Interface of the gCube DataMiner system

DataMiner offers a Web GUI to the users of a VRE. On the left panel (a), the GUI presents the list of capabilities available in the VRE, which are semantically categorised (the category is indicated by the process provider). For each capability, the interface calls the WPS *DescribeProcess* operation to get the descriptions of the inputs and outputs. When a user selects a process, in the right panel the GUI on-the-fly generates different fields corresponding to the inputs. Input data can be selected from the Workspace (the button associated to the input opens the Workspace selection interface). The "Start Computation" button sends the request to the DM Master cluster, which is managed as explained in the previous section. The usage and the complexity of the Cloud computations are completely hidden to the user, but the type of the computation is reported as a metadata in the provenance file. In the end, a view of the Workspace folders produced by the computations is given in the "Check the Computations" area (b), where a summary sheet of the provenance of the experiment can be obtained ("Show" button, c). From the same panel, the computation can be also re-submitted. In this case, the Web interface reads the Prov-O XML information associated to a computation and rebuilds a computation request with the same parameters. The computation folders may also include computations executed and shared by other users. Finally, the "Access to the Data Space" button allows the user to obtain a list of the overall input and output datasets involved in the executed computations (d), with provenance information attached that refers to the computation that used the dataset.

3.8.2 Software and Algorithms Integration



```
#######################################################################################
############ Absence Generation Script - Gianpaolo Coro and Chiara Magliozzi, CNR 2015, Last version 06-07-2015 ###########
#######################################################################################

#52North WPS annotations
# wps.des: id = Absence_generation_from_OBIS, title = Absence_generation_from_OBIS, abstract = A script to estimate absence reco

rm(list=ls(all=TRUE))
graphics.off()

## charging the libraries
library(DBI)
library(RPostgreSQL)
library(raster)
library(maptools)

# time
t0<-Sys.time()

## parameters
# wps.in: id = list, type = text/plain, title = list of species beginning with the speciesname header,value="species.txt";
list= "species.txt"
specieslist<-read.table(list,header=T,sep=",") # my short dataset 2 species
#attach(specieslist)
# wps.in: id = res, type = double, title = resolution of the analysis,value=1;
res=1;
extent_x=180
extent_y=90
n=extent_y*2/res;
m=extent_x*2/res;
# wps.in: id = occ_percentage, type = double, title = percentage of observations occurrence of a viable survey,value=0.1;
occ_percentage=0.1 #between 0 and 1

#uncomment for time filtering

#No time filter
TimeStart<-"";
TimeEnd<-"";

TimeStart<-gsub("(^ +)|( +$)", "",TimeStart)
TimeEnd<-gsub("(^ +)|( +$)", "", TimeEnd)
```

Name	Description	Type	Default	I/O
list	list of speci…	File	species.txt	Input
res	resolution …	Double	1	Input
occ_perce…	percentag…	Double	0.1	Input
zipOutput	zip file con…	File	output.zip	Output

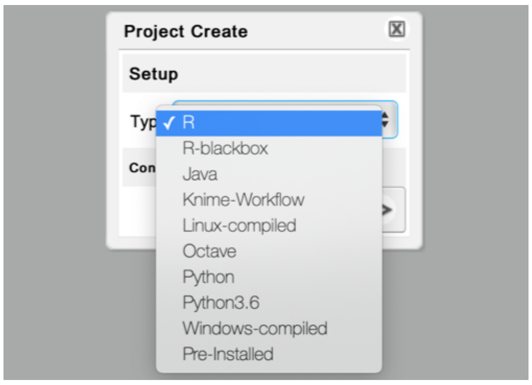
Name	Created
Target	24 Feb 12:26 PM 2016
AbsencesSpeciesList-prod-annotated.r	02 Feb 05:29 PM 2016
stat_algo.project	02 Feb 05:34 PM 2016

Figure 3.18: Interface to import an R process on DataMiner

Prototype scripting is the basis of most models in environmental sciences. Scientists making prototype scripts (e.g. using R and Matlab) often need to share results and make their models used also by other scientists on new data. To this aim, one way is to publish scripts as-a-Service, possibly under a recognized standard (e.g. WPS). The Statistical Algorithms Importer (SAI[5]) is an interface

[5] Described in Coro, G., Panichi, G., & Pagano, P. (2016). A Web application to publish R scripts as-a-Service on a Cloud computing platform. Bollettino di Geofisica Teorica ed Applicata, 57, 51-53. With a user guide available at https://wiki.gcube-system.org/gcube/Statistical_Algorithms_Importer

that allows scientists to easily and quickly import scripts onto DataMiner. DataMiner 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 DataMiner Cloud computing platform and are accessible via the WPS standard.



The SAI interface for R scripts resembles the R Studio environment, a popular IDE for R scripts, in order to make it friendly to script providers, whereas the interface for software written in other programming languages does not allow to edit the main script. However, SAI provides support for scripts implemented in several languages as shown in the following picture.

The *Project* button allows creating, opening and saving a working session. A user uploads a set of files and data on the workspace area (lower-right panel). Upload can be done by dragging and dropping local desktop files. As next step, the user indicates the “main script”, i.e. the script that will be executed on DataMiner and that will use the other scripts and files.

For R scripts integration, after selecting the main script, the left-side editor panel visualises it with R syntax highlighting and allows modifying it.

Afterwards, the user indicates the input and output of the script by highlighting variable definitions in the script and pressing the *+Input* (or *+Output*) button. In the case of other programming languages than R, the Input and Output variables should be manually specified directly in the Input / Output panel.

For R scripts, SAI also supports WPS4R annotations inside the script to automatically generate inputs and outputs. Other tabs in this interface area allow setting global variables and adding metadata to the process. In particular, the *Interpreter* tab allows indicating the R interpreter version and the packages required by the script and the *Info* tab allows indicating the name of the algorithm and its description. In the *Info* tab, the user can also specify the algorithm’s name and category.

Once the metadata and the variables information has been fulfilled, the user can create one DataMiner as-a-Service version of the script by pressing the *Create* button in the Software panel. The term “software”, in this case indicates a Java program that implements an as-a-Service version of the user-provided scripts. The Java software contains instructions to automatically download the scripts and the other required resources on the server that will execute it, configure the environment, execute the main script and return the result to the user. The computations are orchestrated by the DataMiner computing platform that ensures the program has one instance for each request and user. The servers will manage concurrent requests by several users and execute code in a closed sandbox folder, to avoid damage caused by malicious code.

Based on the SAI Input / Output definitions written in the generated Java program, DataMiner automatically creates a Web GUI. By pressing the *Publish* button, the application notifies DataMiner that a new process should be deployed. DataMiner will not own the source code, which is downloaded on-the-fly by the computing machines and deleted after the execution.

This approach meets the policy requirements of those users who do not want to share their code. The *Repackage* button re-creates the software so that the computational platform will be using the new version of the script. The repackaging function allows a user to modify the script and to immediately have the new code running on the computing system. This approach separates the script updating and deployment phases, making the script producer completely independent on e-Infrastructure deployment and maintenance issues. However, deployment is necessary again whenever Input / Output or algorithm's metadata are changed.

To summarise, the SAI Web application relies on the D4Science e-Infrastructure and enables a software (R, Java, Fortran, Linux-compiled, .Net, Octave, KNIME, Python), provided by a community of practice working in a VRE, with as-a-Service features. SAI reduces integration time with respect to direct Java code writing. Additionally, it adds (i) multi-tenancy and concurrent access, (ii) scope and access management through Virtual Research Environments, (iii) output storage on a distributed, high-availability file system, (iv) graphical user interface, (v) WPS interface, (vi) data sharing and publication of results, (vii) provenance management and (viii) Auditing facilities.

3.9 D4Science Tools Provision and Integration

The tools provision and integration of software deployed as service follows:

3.9.1 SmartGears

SmartGears is a set of Java libraries that turn Servlet-based containers and applications into gCube resources, transparently.

In this section, we introduce SmartGears[6] and explain how it is an improvement over existing gCube solutions. The discussion is relevant to node and infrastructure managers, who perform and maintain SmartGears installations, and to developers, who package or write software for a gCube infrastructure.

A piece of software is an infrastructure resource (the so-called Software as Resource, SaR) if we can manage it in the D4Science infrastructure. This means that we can do a number of things with the software, including:

[6] More details about SmartGears can be found in the set of dedicated wiki pages at the address: https://wiki.gcube-system.org/gcube/SmartGears.

- discover where it is deployed in the infrastructure, so as to use it without hard coded knowledge of its location. For this, we need to describe each and every software deployment, and publish these descriptions, or profiles, in the infrastructure;
- monitor and change the status of its deployments, so as to take actions when they are not in an operational status (e.g. redeploy the software, or at least prevent discovery and usage of the deployments). For this, we need to track their current status, report it in the profiles we publish, and republish the profiles when the status changes;
- dedicate its deployments to certain groups of users, in the sense that only users in those groups can use them. We can change the sharing policies of individual deployments at any time, i.e. share them across more or less groups. We can also grant different privileges to different types of users within given groups.

Publication, *discovery*, *lifecycle management*, *controlled sharing* are the pillars of the resource management. Yet relying on humans to compile deployment profiles, publish them in the infrastructure, keep track and change the status of deployments, or enforce sharing policies is all but practical. In some cases, it is downright impossible. We need instead automated solutions that live alongside each and every deployment and help us turn it into a resource we can manage. SmartGears is one such solution.

We focus on software that can be used over the network, such as distributed applications and network services. Software deployments then correspond to software endpoints.

Typically, software endpoints run within containers and, in D4Science, containers can be resources in their own right, the so-called gCube Hosting Nodes (gHNs).

Managing gHNs is a way to manage multiple endpoints simultaneously (e.g. to deactivate a gHN means to deactivate a set of endpoints at once). Equally, it is a way to manage underlying hardware resources (e.g. dedicate a gHN to selected groups of users).

This is a notion of "Container-as-Resource" (CaR), and it raises the same requirements as SaR, including publication and discovery, lifecycle management, and controlled sharing. SmartGears helps us meet these requirements too, i.e. turns containers as well as the endpoints therein into gCube resources.

SmartGears is not a development framework. Rather, SmartGears is invisible to the software, not part of its stack at all. As a result, any software can run in the infrastructure: SaR becomes a nature that software acquires at runtime.

Indeed, SmartGears has few requirements of the software. All we ask of software is that it is based on the Servlet specifications, which define the hooks that we need to track its lifecycle and its use. The software is thus a Web Application and may more specifically be a Soap service, a Rest service, or a generic Web Application. It may adopt different standards and technologies (e.g. JAX-RPC, JAX-WS, JAX-RS, but also Dependency Injection technologies, persistence technologies, etc.). And of course, it may run in any container that is Servlet-compliant (Web Containers, Application Servers).

Finally, the evolution of SmartGears is inconsequential for the software: most of the APIs of SmartGears remain private to SmartGears.

Containers and applications need a minimal set of requirements before SmartGears can turn them into gCube resources:

- containers must comply with version 3 of the Servlet specifications;
- applications must include at least one gcube-app.xml configuration file alongside their deployment descriptor (i.e. under /WEB-INF);

In addition:

- node managers must define a GHN_HOME environment variable that resolves to a location where SmartGears can find a container.xml configuration file.

Starting from version 3, the Servlet specifications allow SmartGears to intercept relevant events in the lifecycle of individual applications whilst being shared across all applications, in line with the deployment scheme of SmartGears. In particular, the specifications introduce a ServletContextInitializer interface that SmartGears implements to be notified of application startup. The specifications also allow programmatic registration of filters and servlets, which SmartGears uses to transparently manage applications without the need of additional configuration in their web.xml descriptor.

Configuration is thus limited to WEB-INF/gcube-app.xml and $GHN_HOME/container.xml, which provide the configuration of, respectively, the application and the container as gCube resources. Details about their contents are available in the gCube Wiki Appendices.

Smartgears is distributed as a tarball that contains the libraries, scripts, and configuration files required to install Smartgears in a given container, and to maintain the installation over time. Instructions on how to download, install and maintain Smartgears are available in the SmartGears Web Hosting Node (wHN) Installation.

3.9.2 OAuth2.0

By means of the OAuth 2.0 protocol (authorised) third party applications can operate on a user's behalf over the D4Science infrastructure (while protecting the member's credentials). For more information about the OAuth authorization framework please visit the official OAuth site[7]. For technical details also see the OAuth 2.0 RFC[8]. In the following, the steps needed to authorize third party applications to operate on a user's behalf and the D4Science infrastructure are explained.

This exploitation case makes it possible to integrate in the infrastructure services, tools, and applications that are not deployed on SmartGears powered containers.

More details about how the OAuth 2.0 service work can be exploited in D4Science can be found in the set of dedicated wiki pages at the address:

https://wiki.gcube-system.org/gcube/OAuth2.0

[7] https://oauth.net/2/
[8] https://tools.ietf.org/html/rfc6749%7C

3.10 D4Science Exploitation Models

D4Science offers access to services delivered and managed via tailored Virtual Research Environment. Each VRE can have different access policies that can be selected at any time during the operation of the VRE:

- *private access*: the VRE is private and a user can access it only by invitation issued by the VRE Manager. Upon acceptance of the user of the invitation, the user becomes member of the VRE with a user role;
- *restricted access*: a user registered to the gateway and with a valid identity can request access to the VRE. The VRE Manager can approve or reject any user request. In case of approval, the user becomes member of the VRE with a user role;
- *public access*: a user registered to the gateway and with a valid identity can become member of the VRE by simply accessing it. The VRE Manager is just notified.

Furthermore, four different roles are supported by default and additional ones can be defined as per request of the Customer:

- *VRE Manager*: this kind of user manages user registration/deregistration to the VRE and assign/remove roles by accessing the Administration dashboard;
- *User*: any user with this role can access the VRE data and share private data with other selected members of the VRE;
- *Processor* (Optional): any user with this role can register its own process and execute a process by accessing the Method Engine service;
- *Editor*: the users with this role can publish data to all members of the VRE by exploiting the Catalogue service.

Tailored exploitation models have been designed for a single user, a group of users, and for community of users. The main characteristics of each of them are reported in the following table.

As-a-User

Description	You need a workspace where you can store, access, and optionally share files and datasets. You wish to follow the activities performed by the other users of a large community and to join one of the existing applications offered by the existing VREs.
Reserved Resources	Zero. Storage and computational jobs queues are shared with other users.
Resources Highlights	Your data is kept private until you share it. However, the computational resources are shared with other users and assigned to you only when needed.
How	Register on one of the available Gateways and join one of the existing public Virtual Research Environment.

Type of engagement	You are a single user requiring immediate access to data and tailored applications.
Cost	Free to use according to the Terms of Use.

As-a-Group

Description	You need to create a dedicated environment with specific applications and manage the users by authorizing them to join. Only authorized users will have access to the data used and generated by the newly created applications. You may exploit any of the capabilities and you may add your specific applications and data.
Reserved Resources	From Zero to the resources assigned to manage your private applications.
Resources Highlights	Your data is kept private until you share it with other user of the newly created application. However, the computational resources are shared with other users and assigned to you only when needed.
How	Register on one of the available Gateways, join the gCubeApps community and ask the gCubeApps VO Manager to become a VRE Designer. Then, design your specific new VRE that will be subject to approval by the VO Manager. On average, a new application is created in less than 24 hours.
Type of engagement	You manage a small set (up to hundreds) of users focusing on a specific scientific topic requiring a common infrastructure to store, maintain, and process data.
Cost	Free to use according to the Terms of Use and without installing any custom and private service.

As-a-Community

Description	You need to manage a community for a medium/long period and you want to offer members an integrated storage and computational platform for the execution of daily tasks. You may exploit any of the available capabilities and you may add your specific applications and data. Data and applications are kept private to your community and confidentiality and security are guaranteed by encryption.
Reserved Resources	A minimum equivalent Amazon m3.xlarge – 15 GiB of memory, 13 EC2 Compute Units (8 virtual cores with 3.25 EC2 Compute Units each), EBS storage only, 64-bit platform - is assigned to the community. According to the applications selected in the Application Bundles that have to be offered to the community, the minimum configuration can grow up to 2 m3.2xlarge – 30 GiB of memory, 26 EC2 Compute Units (16 virtual cores with 3.25 EC2 Compute Units each), EBS storage only, 64-bit platform – resources that are reserved to the community. Additional resources can be assigned on-demand to scale up distributed computations and up to the

	negotiated quota as established in the negotiated phase between the Community Manager and the iMarine Infrastructure Manager.
Resources Highlights	Your data is kept private until you or any other authorized user shares it with either other users of the newly created applications or with the wider community. Only the members of the community use the computational resources assigned to it and the data is moved and stored according to an encryption key that is specific to the community.
How	Contact D4Science through the contact-point and report the description of the community, the average number of users, the scientific applications you wish to host in D4Science, and any additional information useful for a proper analysis of your request.
Type of engagement	You manage a medium/large set of users focusing on many scientific topics ranging from the analysis of statistical and biodiversity data to the management of geo-referenced data. The community needs a common infrastructure to store, maintain, and process data through the creation of focused applications (VREs), each of which is focusing on a specific scientific goal.
Cost	Negotiated. It depends on the QoS and the number of users of the community.

3.11 D4Science APIs: Useful Links

- Developers web site: to get information about a set of commonly used APIs:
  - https://dev.d4science.org/,
- Profile & Social Networking API: to get profile and user information or boost your content's reach by making it easy for people to share it on Virtual Research Environments (VREs)
  - https://dev.d4science.org/swagger/social-networking/
- Workspace (Storage Hub) API: to learn how to browse, upload and download user's workspace files and folders:
  - https://gcube.wiki.gcube-system.org/gcube/StorageHub_REST_API
- Metadata Catalouge (gCat) API: to learn how to publish and search collections of metadata for items including data, services, and related information objects:
  - https://wiki.gcube-system.org/gcube/GCat_Service
- Information System API: to learn how to interact with resources hosted in the Infrastructure and its Virtual Research Environments (VREs):
  - https://dev.d4science.org/swagger/registry/
- New Methods/Algorithms for DataMiner: to learn how to implement custom Methods and Algorithms for DataMiner:
  - https://wiki.gcube-system.org/gcube/Category:Statistical_Algorithms_Importer
- Supported languages for new Methods/Algorithms for DataMiner:
  - https://wiki.gcube-system.org/gcube/Statistical_Algorithms_Importer:_Create_Project

- DataMiner online interfaces
    - Web: https://wiki.gcube-system.org/gcube/DataMiner_Manager)
    - Web Processing service: https://www.opengeospatial.org/standards/wps
- DataMiner overview:
    - https://wiki.gcube-system.org/gcube/Data_Mining_Facilities.
- authorisation framework.
    - https://dev.d4science.org/authorization/
    - https://wiki.gcube-system.org/gcube/Authorization_Framework.
- Spatial Data Infrastructure capabilities
    - https://gcube.wiki.gcube-system.org/gcube/SDI-Service
    - https://gcube.wiki.gcube-system.org/gcube/Spatial_Data_Storage_and_Publishing
    - https://gcube.wiki.gcube-system.org/gcube/Spatial_Data_Discovery_and_Access.

To communicate with D4Science and to get additional information about Docker, ShinyApps, and third-parties services

- https://support.d4science.org/

3.12 D4Science References

[1] 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**, Future Generation Computer System (2019)
DOI: 10.1016/j.future.2019.05.063

[2] L. Candela, D. Castelli, P. Pagano (2013) **Virtual Research Environments: An Overview and a Research Agenda**. Data Science Journal, Vol. 12

[3] Assante, M., Candela, L., Castelli, D., Cirillo, R., Coro, G., Frosini, L., Lelii, L., Mangiacrapa, F., Marioli, V., Pagano, P., Panichi, G., Perciante, C., Sinibaldi, F. **The gCube System: Delivering Virtual Research Environments as-a-Service**. Future Generation Computer Systems (Vol. 95)
DOI: 10.1016/j.future.2018.10.035

[4] *Coro, G., Panichi, G., Scarponi, P., & Pagano, P.* (2017). **Cloud computing in a distributed e-infrastructure using the web processing service standard**. Concurrency and Computation: Practice and Experience, 29(18), e4219

4 Integration aspects

In this chapter a number of integration aspects are considered, which are relevant for the interactions between the 2 main Blue Cloud components and for further federation of computing resources and algorithms. The Blue Cloud VRE will have its basis at the D4Science infrastructure, as operated by CNR-ISTI. But it should be possible to connect to other services than data at blue infrastructures and to e-infrastructures as part of the VRE and its Virtual Labs and to perform analyses in workflows which are divided over multiple platforms, with the Virtual Lab at D4Science in the director role.

4.1 Link from Data Discovery and Access service to VRE Data Pool.

As described in Chapter 2 there is a data exchange planned between the Blue Cloud Data Discovery and Access service and the Blue Cloud Virtual Research Environment (VRE). In paragraph 2.4 this bridge is already detailed from the perspective of the Blue Cloud Data Cache that is to be developed and operated by EUDAT.

In this paragraph the bridge is considered from the perspective of Blue Cloud VRE that is operated by CNR-ISTI.

The VRE Data Pool will be based on the D4Science Shared Workspace, hereafter shortly named Workspace. The workspace is at the core of the D4Science infrastructure since, abstracting over several technologies, it offers a single space where to manage data and products either uploaded, shared, or generated in the VRE.

The Shared Workspace System provides an easy-to-use interface and APIs on top of a distributed, fault-tolerant, replicated, and secure cloud storage.

The Workspace provides a cloud storage file sharing service, supporting:

- Sharing of folders and files of different item types (ranging from binary files to information objects representing, for instance, tabular data, distribution maps, algorithms/methods);
- Publication of the items contained in a directory on the Catalogue;
- Distribution of the items contained in a directory via the THREDDS Server as well as their continuous and automatic synchronization with a THREDDS collection;
- Persistence of any computation results/output. The results are automatically saved and accompanied by provenance information as well as by the input files used to launch the computation.

The VRE Data pool will be manifested as a directory in the Workspace and the user will be able to use it as any other directory. Thus, she will be able to share content saved in that directory, to publish data in the Catalogue, to start computation using those data, etc.

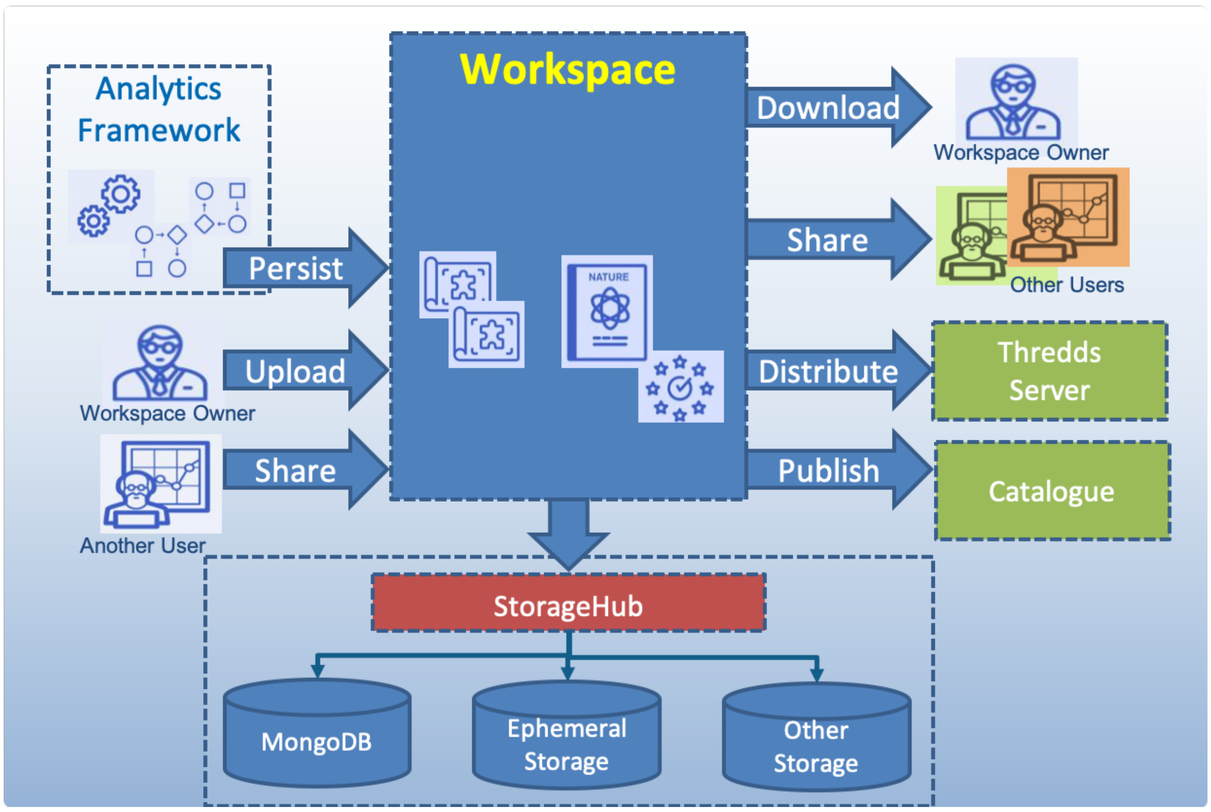


Figure 4.1: The Workspace

It is worth reporting that the Workspace uses versioning and versions automatically the stored items so that it is possible to refer and access any version of the item at any time using its persistent, unique, and web-accessible identifier (Persistent URL). This feature is exploited by the Analytics Framework to support reproducibility. The Analytics Platform generates a detailed provenance record for every analytics task executed by the platform, by reporting the Persistent URL to the version of the data specified as input parameters, the parameters of the execution, and any other metadata that would allow a user to reproduce and repeat the analytical task. This information is stored into the workspace and documented by a PROV-O-based accompanying record together with the data generated by the task.

The Workspace is accessible via the graphical user interface and via the StorageHub APIs (https://gcube.wiki.gcube-system.org/gcube/StorageHub_REST_API).

To exploit those APIs, a valid authorization token has to be used. In short, the APIs provide the following REST operations:

- **Retrieve WS**: to retrieve the user Workspace;
- **Folder Listing**: to list the content of a folder;
- **Retrieve VRE Folder**: to retrieve the VREFolder related to the token;
- **Find**: to find a file or a folder by name or pattern;
- **Delete**: to remove a file or a folder (including subfolders);
- **Download**: to download a file or a folder in ZIP format;
- **Get Public Link**: to get a public link of a file;

- **Create Folder**: to create a folder in the given parent folder;
- **Unzip**: to upload a zip file in a specific folder;
- **Upload file**: to upload a file in a folder.
- **Versions**: to get a specific version of a file.

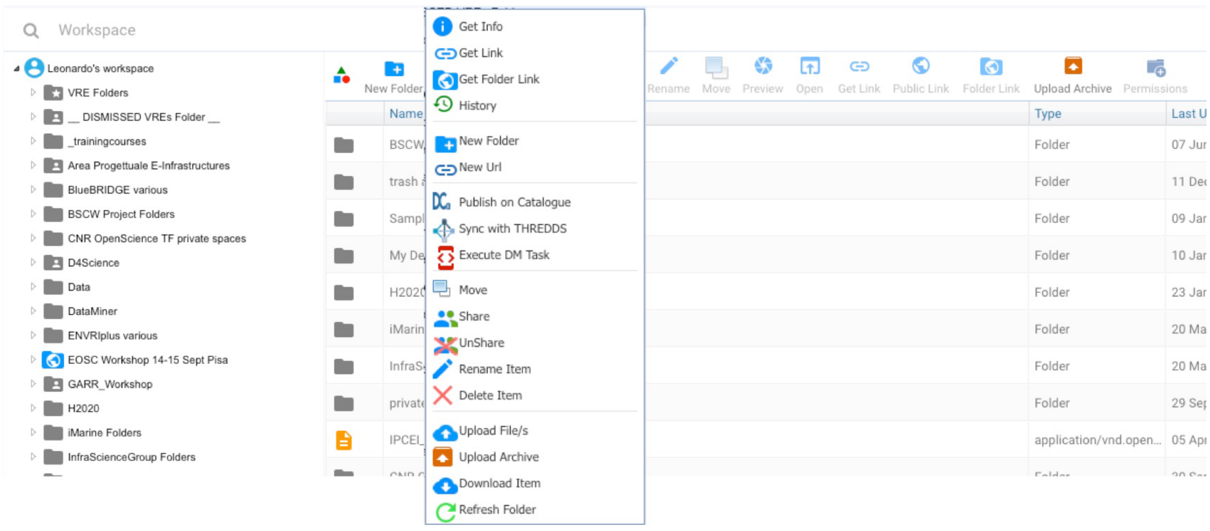


Figure 4.2: The Workspace Graphical User Interface

4.2 Authentication interoperability between systems

The D4Science token-based Authorization System is compliant with the Attribute-based access control (ABAC), which defines an access control paradigm whereby access rights are granted to users through the use of policies which combine attributes together. The D4Science token is a string generated on request by the Authorization service for identification purposes and associated with every entity belonging to D4Science (users or services). The token is passed in every call and is automatically propagated in the lower layers. It can be passed to a D4Science service in 3 ways:

- using the HTTP-header: adding the value ("gcube-token","{your-token}") to the header parameters;
- using the query-string: adding “gcube-token={your-token}” to the existing query-string;
- logging via the default WEB authentication shown by the browser using your username as username and your token as password.

A Blue Data Service is either a pre-existing service or a new one deployed and operated by an infrastructure different from D4Science that wishes to become interoperable with the Blue-Cloud VRE. Hence, any Blue Data Service needs to use a D4Science token to be able to identify the user identity, authorize the user request, and call any other D4Science service if needed.

There exist 2 types of D4Science tokens: User and Application token. The former is used by “humans” using the VRE whereas the latter is used by services/applications.

There are 3 possible approaches a Blue Data Service can adopt, they are depicted in the 3 scenarios present in Figure 4.3, each of these has pros and cons.

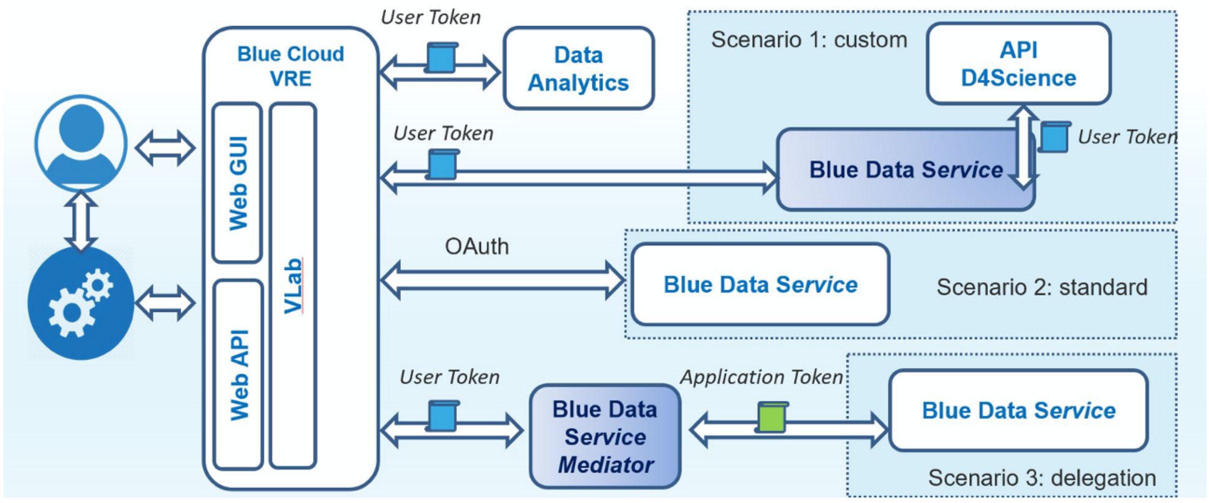


Figure 4.3: the supported Scenarios enabling interoperability across services

Pros and cons of each modality are:

1. **The custom modality:** requires that any Blue Data Service expects to receive a User Token as input parameter, this parameter can have any name (d4s-token in the example) and is added in the HTTP Query String whenever the Blue Data Service is called via HTTPS: eg https://mybluedataservice.eu/endpoint?d4s-token=**12345-abcd**
   In turn, the Blue Data Service uses this token to authenticate and resolve the user and the role associated by making a HTTP REST call to a given D4Science API[9] using the d4s-token received in the previous step.
   For instance, suppose that the token **12345-abcd** would belong to the user "Andrea Rossi" in the VLab named Blue-CloudLab, then the HTTPS REST call:

   https://api.d4science.org/rest/2/people/profile?gcube-token=**12345-abcd**

   Would return the following JSON:

```
{
 "success": true,
 "message": null,
 "result": {
  "roles": [VRE-Manager],
  "context": "/d4science.research-infrastructures.eu/D4OS/Blue-CloudLab",
  "avatar": "https://services.d4science.org/image?img_id=5567",
  "fullname": "Andrea Rossi",
  "username": "andrea.rossi"
```

[9] https://api.d4science.org/rest/2/people/profile

```
 }
}
```

PROS: Completely transparent to the end-users.
CONS: Blue Data Service needs to be slightly modified to become interoperable and this is not always possible.

See also API available at: https://dev.d4science.org

2. **The standard modality:** requires that any Blue Data Service is compliant with The OAuth 2.0[10] and OpenID Connect. OAuth 2.0 is designed only for authorization, for granting access to data and features from one application to another. OpenID Connect[11] (OIDC) is a thin layer that sits on top of OAuth 2.0 that adds login and profile information about the person who is logged in. To this end each Blue Data Service is provided with a Client ID and a Client Secret, that only the Blue Data Service and the D4Science Authorization Server know. The user logs in on the Blue Data Service via her D4Science credentials, in turn, the Blue Data Service uses the access token returned by the OAuth/OpenID Connect flow to interact with any other D4Science service on behalf of the user.

   PROS: Blue Data Service may already be compliant with such standard and required no modification except for the configuration parameters.
   CONS: The user has to login a first time to exploit the Blue service;

3. **Delegation (Mediator) modality:** requires that any Blue Data Service is "proxied" by a specific mediator service (that may need to be implemented in some cases) capable of authorising the user request and mediate them by exploiting an application token, which in turn is used by the Blue Data Service to authorise the incoming request and may be used to call some, but not every, D4Science service. This modality is not recommended and could be used for specific (corner) cases.

   PROS: Easy to achieve for Blue data services not compliant with standards, few modifications required.
   CONS: Blue-Cloud Management costs, Blue Data Services authorise the VRE and not the users.

[10] https://www.oauth.com/

[11] https://openid.net/connect/

4.3 Blue Cloud System monitoring

Monitoring of the availability and performance of the Blue Cloud system and its components is needed to get insight in its performance.

In paragraph 3.2.2 it is already explained how Nagios is used in the D4Science infrastructure for monitoring availability (uptime) of the VRE and its services. It is reported that for technologies developed both by D4Science and by the exploited framework, i.e. gCube and D-Net, specific add-ons have been designed, implemented, and installed to extend monitoring and native alerting functionality in order to have a fully-complete and always up-to-date image of the status of the D4Science e-infrastructure.

In the Blue Cloud Architecture this set of add-ons will be extended to add any Blue Cloud service that will be exploited in the Blue Cloud VRE. To do so, it will be requested to any of these Blue Cloud services a web service endpoint that can be invoked to monitor the service availability and possibly another web service endpoint that can be invoked to monitor the service reliability.

In Chapter 3 about D4Science, in several paragraphs also auditing software components were described. This set of components allows to record, summarize and classify service invocations and other events, e.g. storage of data, access to the Workspace, usage of the Catalogue, etc.

The following set of indicators are currently collected and made available. This is done by means of two dashboards accessible for the Blue Cloud VRE managers: the Accounting Dashboard and the DataStudio Dashboard.

Accounting Dashboard

The **Accounting Dashboard** allows analysing the exploitation of the different services offered by the Blue Cloud VRE. This analysis can be performed either at Blue Cloud VRE level or at the level of each hosted Virtual Laboratory. Several measures and indicators can be monitored in this way across time. The data are aggregated by month. There are five measures: Accesses, Catalogue, Method Invocation, Social Interactions, and Users. Each measure then offers several indicators that are accessible both at Blue Cloud VRE level and Virtual Laboratory level.

The Accesses measure includes the following indicators:

- Access to any Virtual Laboratory;
- Access to the Workspace;
- Access to the Message;
- Access to Notification settings;
- Access to the User Profile;
- Access to the Catalogue.

The Catalogue measure includes the following indicators:

- Download of a manifestation associated with an item published in the catalogue;
- Access to the item metadata;
- Queries performed in the catalogue.

The Method Invocation measure includes just one indicator:

- Method invocation, i.e. number of jobs executed in the Blue-Cloud VRE.

The Social Interactions measure includes the following indicators:

- Posts
- Replies
- Likes

The User measure include just one indicator

- Users, i.e. number of users registered/unregistered to/from the Blue-Cloud VRE.

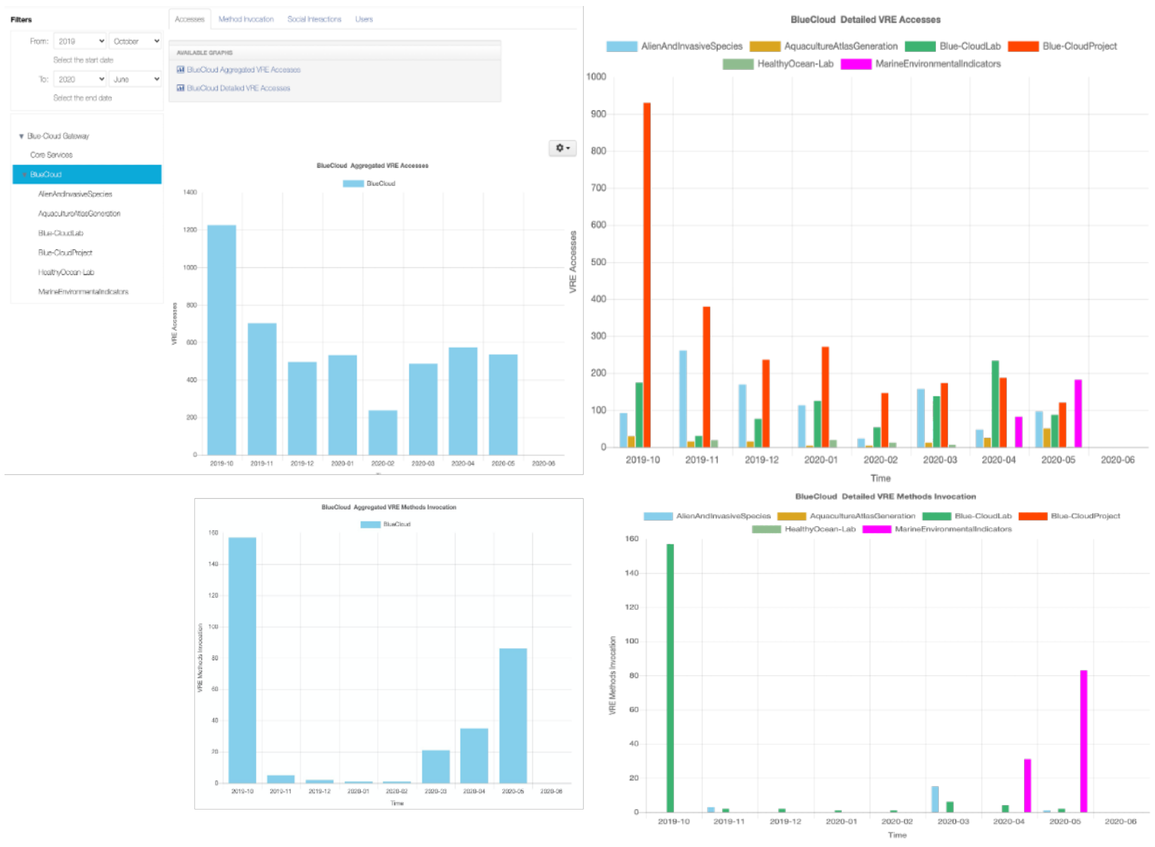


Figure 4.4: Accounting dashboard

The Accounting Dashboard can be extended with additional accounting data plugins. Each plugin has to define the measure and the indicators and it has to report data monthly aggregated. This mechanism will be exploited to add specific measures for the Blue Cloud services.

DataStudio Dashboard

The **DataStudio Dashboard** is realized exploiting Google Analytics and Google Data Studio. It can be used to analyse the exploitation of the Blue-Cloud VRE in terms of web accesses, geographically distribution of the accesses, web sessions and their average duration. Any Blue Cloud service may contribute to the DataStudio Dashboard by specifying the Google Accounting ID defined for Blue Cloud VRE.

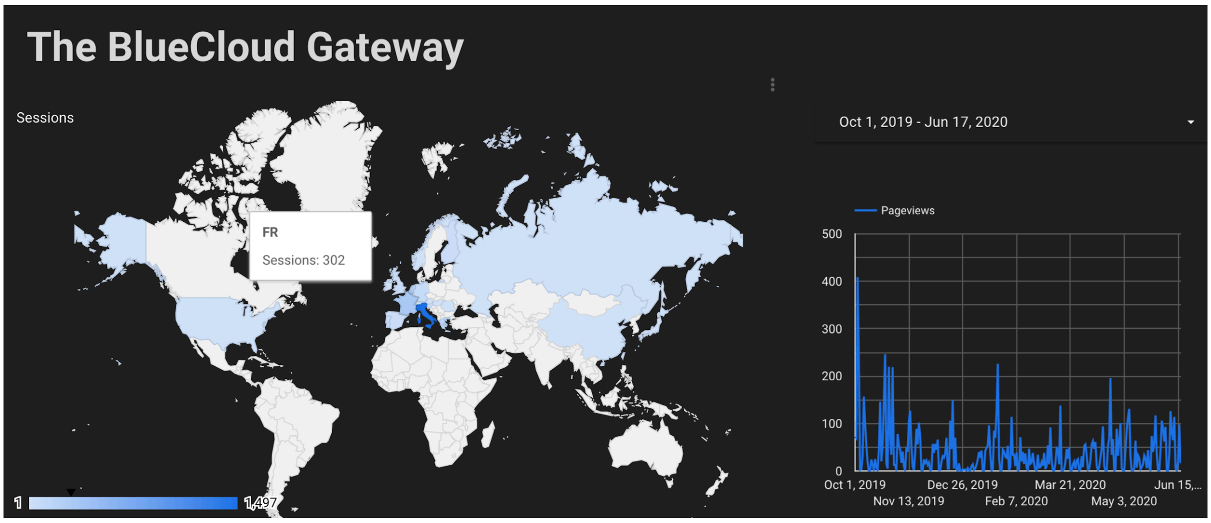


Figure 4.5: DataStudio dashboard

Annex 1 EUDAT Collaborative Data Infastructure

EUDAT has developed a service stack that forms the Collaborative Data Infrastructure (CDI). The services are as follows:

- B2SAFE, Replicate Research Data Safely.
- B2STAGE, Get Data to Computation.
- B2FIND, Find Research Data.
- B2SHARE, Store, Share and Publish Research Data.
- B2DROP, Sync and Exchange Research Data.

In addition, a set of EUDAT core operational services, essential for the management of the CDI have been defined as follows:

- B2ACCESS (identity and authorisation), easy-to-use and secure Authentication and Authorisation platform.
- B2HANDLE (Persistent IDentification (PID) management), a service to register persistent identifiers called Handles to data objects and retrieve data objects via these identifiers, serving a purpose similar to DOIs for papers.
- B2HOST, a Service Hosting Framework that allows communities to deploy and operate their own applications and data-oriented services on machines next to the data storage.

Most of these components are depicted in Figure 1 and described in more detail below

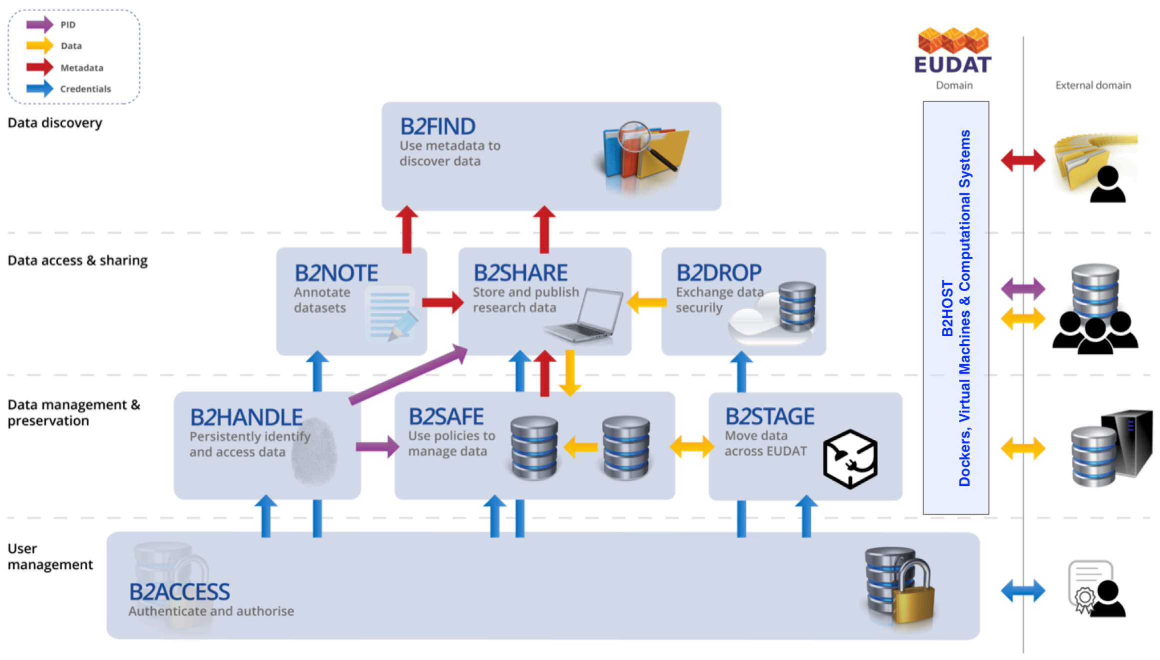

PID
Data
Metadata
Credentials
Data discovery
Data access & sharing
Data management & preservation
User management
EUDAT
Domain
External domain
B2FIND
Use metadata to discover data
B2NOTE
Annotate datasets
B2SHARE
Store and publish research data
B2DROP
Exchange data securly
B2HANDLE
Persistently identify and access data
B2SAFE
Use policies to manage data
B2STAGE
Move data across EUDAT
B2ACCESS
Authenticate and authorise
B2HOST
Dockers, Virtual Machines & Computational Systems

Figure 1: EUDAT Collaborative Data Infrastructure Services Suite & Architecture

B2ACCESS

B2ACCESS is an easy-to-use and secure Authentication and Authorization platform developed by EUDAT. B2ACCESS is versatile and can be integrated with a wide range of services. When integrated with a given service, the user may log in by using different authentication methods like, Home organisation identity provider, Google account, EUDAT ID.

EUDAT IDs are created by the B2ACCESS upon registration. Therefore, B2ACCESS is an Identity Provider for the users that do not have neither a Google account nor a Home Organisational Identity.

B2ACCESS offers also the tool for the managements of the EUDAT IDs and contain the following features:

- Support for personal certificates to support eduGAIN, Social Identities (Facebook, Google, Microsoft, Gitlab), ORCID and local accounts
- IdP support for SAML, OpenID, OAuth2, X.509
- SP support for SAML, OIDC, OAuth2, X.509
- Supports several methods of authentication via the users' primary identity providers (OpenID, SAML, OAuth2, X.509)
- Integrated services: B2SHARE, B2SAFE, B2STAGE, B2DROP, B2NOTE, GEF, SPMT, DPMT, Confluence wiki, Gitlab
- Allows group-, community- and service managers to specify authorisation decisions.

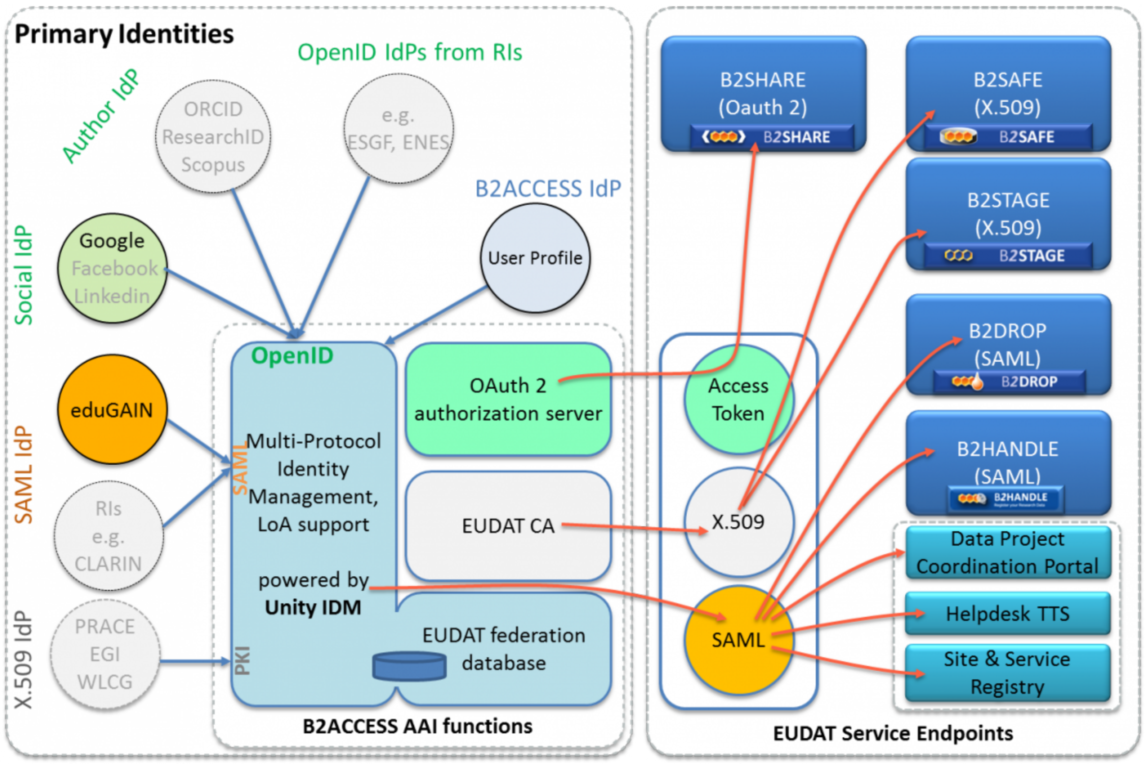


Figure 2: B2ACCESS Components & Architecture

B2SAFE

B2SAFE is a robust, safe and highly available service that allows both community and departmental repositories to implement in a trustworthy manner sets of data management policies across multiple administrative domains.

B2SAFE is a solution intended for:

- Provide an abstraction layer to virtualize large-scale data resources
- Guard against data loss in long-term archiving and preservation
- Optimize access for users from different regions
- Bring data closer to powerful computers for compute-intensive analysis

B2SAFE features:

- Based on the execution of auditable data policy rules and the use of persistent identifiers (PIDs)
- Respects the rights of the data owners to define the access rights for their data and to decide how and when it is made publicly referenceable
- Able to aggregate data from different disciplines into a storage system of trustworthy and capable data service providers
- Support for repository packages (e.g. DSPACE, FEDORA) and a lightweight HTTP-based solution
- Support metadata
- Optimize and extend policies to support data curation and provenance
- Support authorization on basis of community access rules
- Integration with other EUDAT services

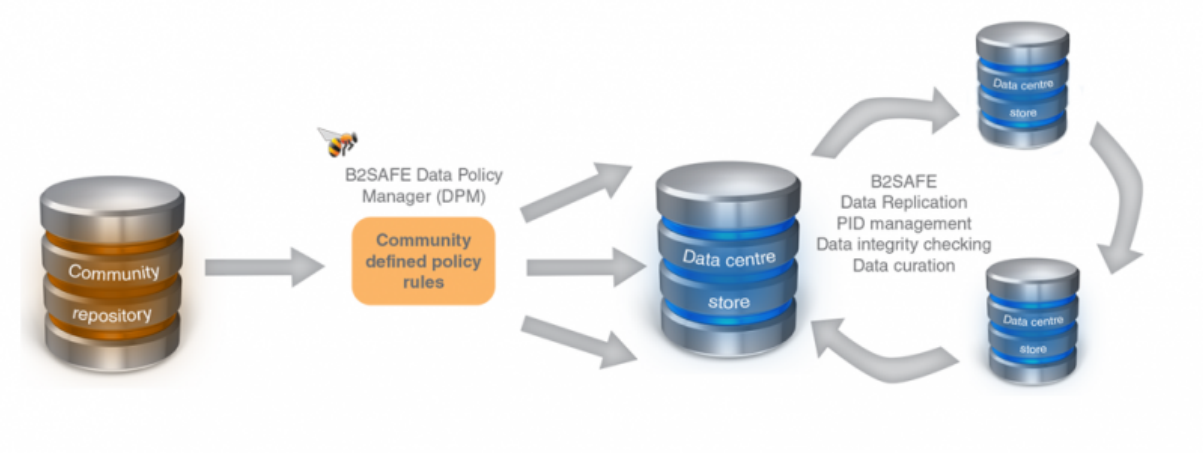


Figure 3: B2SAFE Process & Architecture

B2STAGE

B2STAGE is a reliable, efficient, light-weight and easy-to-use service to transfer research data sets between EUDAT storage resources and high-performance computing (HPC) workspaces.

The service allows users to:

- Transfer large data collections from EUDAT storages to external HPC facilities for processing
- In conjunction with B2SAFE, replicate community data sets, ingesting them onto EUDAT storage resources for long-term preservation
- Ingest computation results into the EUDAT infrastructure
- Access data through a RESTful HTTP interface

B2STAGE features:

- An extension of the B2SAFE and B2FIND services, which allow users to store, preserve and find data
- Data-staging script facilitates staging, ingestion and retrieval of persistent identifier (PID) information of transferred data
- users negotiate access to remote HPC services in parallel
- Collaboration with other infrastructures, such as the European Grid Infrastructure (EGI) and Partnership for Advanced Computing in Europe (PRACE)
- Support GridFTP protocol
- Furthermore, B2STAGE implements a set of Restful HTTP-APIs by adopting an extensible framework used as a basis for the development of community-tailored HTTP-APIs

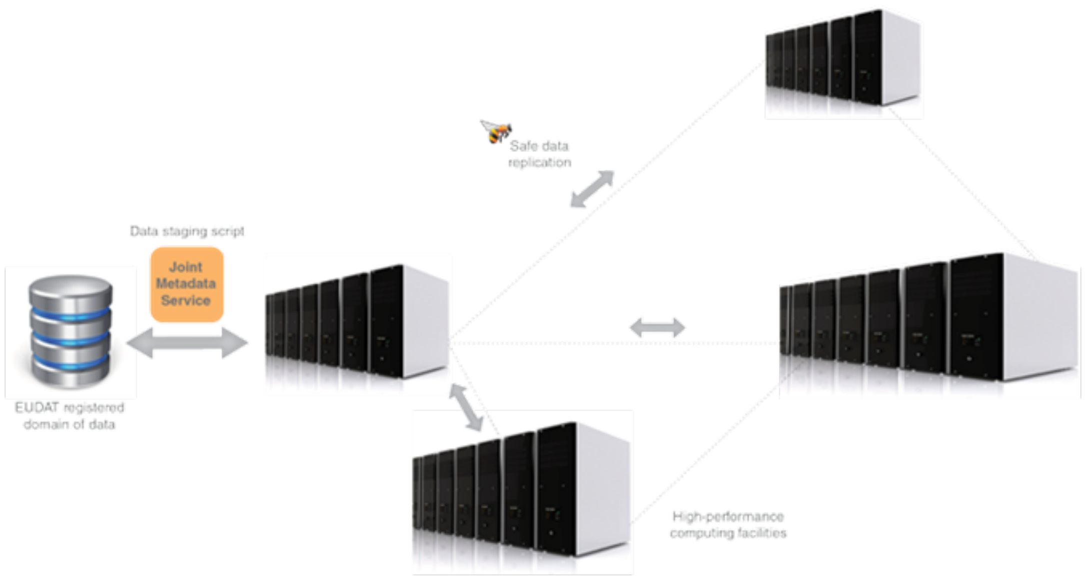

Safe data replication
Data staging script
Joint Metadata Service
EUDAT registered domain of data
High-performance computing facilities

Figure 4: B2STAGE Process

B2HANDLE

B2HANDLE enables EUDAT services and user communities to assign Persistent Identifiers (PIDs) to different kinds of managed objects stored in the EUDAT CDI.

PIDs are used in EUDAT to reliably identify and cite data objects over a long period of time and it is thus a vital part of long-term data management.

The underlying technology of B2HANDLE is based on the Handle System, which is a reliable, redundant and scalable system built on top of an open architecture. B2HANDLE is mostly transparent to the end-users, especially shielding them from the complexity of infrastructure details.

B2HANDLE is a distributed service, with the organisations hosting the service mirroring each other's Persistent Identifiers, thus ensuring the sustainability and reliability of PIDs in the EUDAT domain.

B2HANDLE Features include:

- Follows policies to register data and make it long term referable and citable
- Reliability through mutual PID mirroring
- Provides abstraction layer between a globally unique persistent identifier and physical location of data objects
- Machine readable via HTTP RESTful API

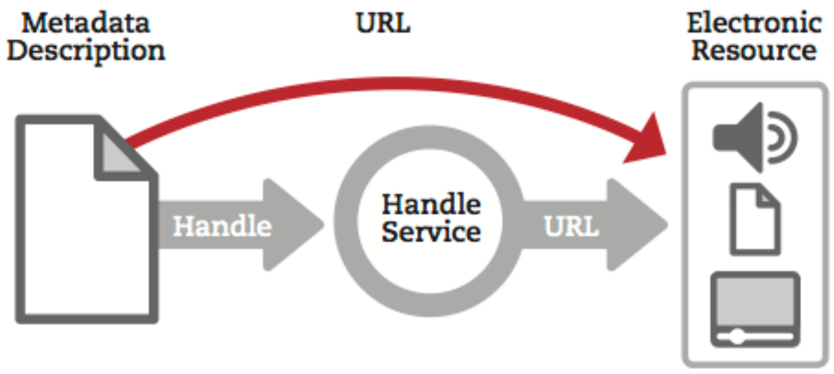


Figure 5: B2HANDLE System

B2SHARE

B2SHARE is a user-friendly, reliable and trustworthy way for researchers, scientific communities and citizen scientists to store, publish and share small-scale research data from diverse contexts.

A formidable solution to:

- Store registered research data (incl. software) and add domain meta data
- Preserve (small-scale) research data for long-term
- Share: allows data, results or ideas to be shared worldwide

B2SHARE features

- Integrated with the EUDAT collaborative data infrastructure
- Free upload and registration of stable research data
- Data assigned a permanent identifier, which can be retraced to the data owner
- Data owner defines access policy
- Community-specific metadata extensions and user interfaces
- Openly accessible and harvestable metadata
- Representational state transfer application programming interface (REST API) for integration with community sites
- Data integrity ensured by checksum during data ingest
- Professionally managed storage service – no need to worry about hardware or network
- Monitoring of availability and use

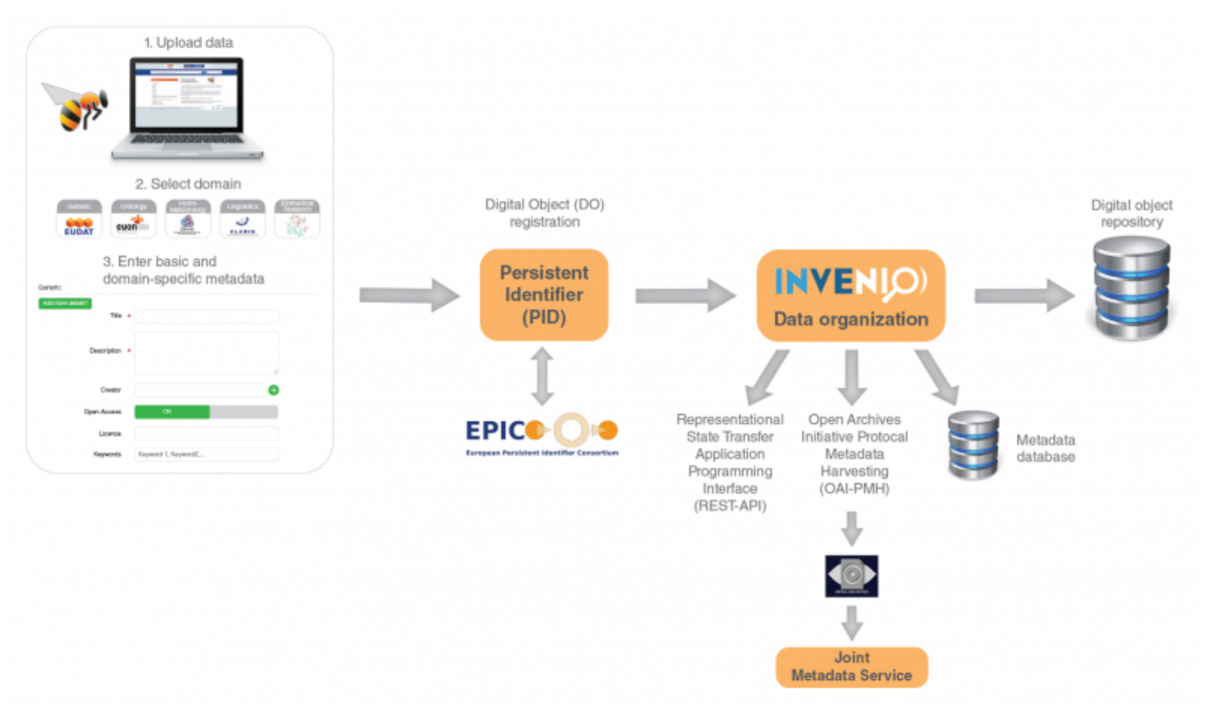

1. Upload data
2. Select domain
3. Enter basic and domain-specific metadata
Digital Object (DO) registration
Persistent Identifier (PID)
EPIC
INVENIO
Data organization
Digital object repository
Representational State Transfer Application Programming Interface (REST-API)
Open Archives Initiative Protocal Metadata Harvesting (OAI-PMH)
Metadata database
Joint Metadata Service

Figure 6: B2SHARE Architecture

B2NOTE

B2Note allows easily creating, searching and managing annotations. An annotation is a keyword or commentary attached to a data object (data collection, file) that explains or classifies it. B2NOTE is a standalone service for annotating data content hosted within the EUDAT CDI.

There exist three types of annotations in B2Note

- The semantic tag, a keyword from an ontology (a semantic tag coming from identified ontology repositories - currently only Bioportal
- The free-text keyword, to be created and used when a specific semantic term is not found
- The comment, a more comprehensive annotation

B2NOTE Features;

- Creation RDF triples
- Harvests information from ontology repositories
- Supports semi-automatic annotation using text mining
- Supports manual data annotation
- Easy to use user interface
- Integrated in B2Share: access files with B2Share, then annotate them with B2Note.
- Annotations are created and stored in a machine-readable format using the W3C Web Annotation model
- Annotations are searchable (in opposite to hand-written annotations).

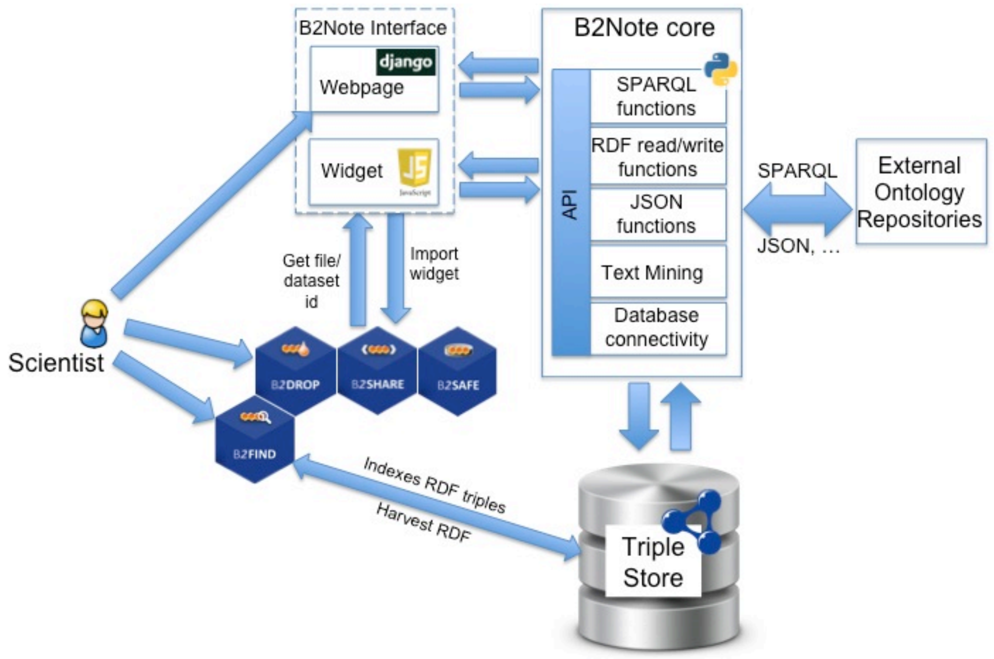


Figure 7: B2NOTE Components & Architecture

B2DROP

B2DROP is a secure and trusted data exchange service for researchers and scientists to keep their research data synchronized and up-to-date and to exchange with other researchers.

The service is simple to use and open to all researchers, scientists, communities alike to synchronise and exchange data with one or multiple users. B2DROP will be fully integrated with the B2 suite of services to allow user-friendly data sharing.

An ideal solution to:

- store and exchange data with colleagues and team members,
- synchronise multiple versions of data,
- ensure automatic desktop synchronisation of large files.

B2DROP Features

- Cloud Storage Federation, collaboration with GEANT in OpenCloudMesh
- B2DROP as a workspace area to computing facilities
- Integration with EUDAT CDI (e.g. B2SHARE)

Users can:

- Define with whom to exchange data, for how long and how
- Are offered up to 20GB of storage space for research data
- Access and manage permissions to files from any device and any location.

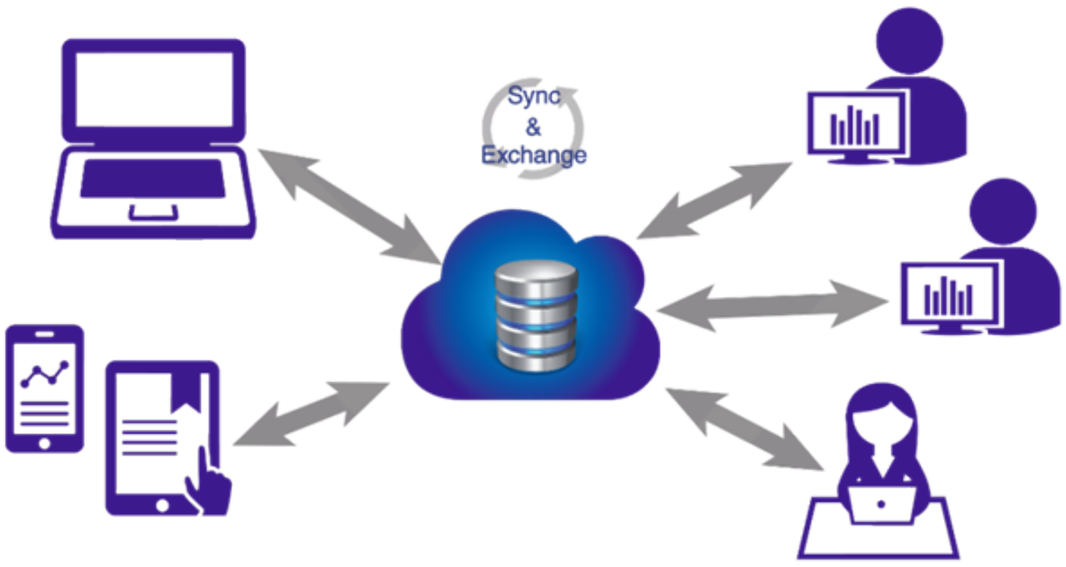


Figure 8: B2DROP

B2HOST

The EUDAT Collaborative Data Infrastructure offers a Service Hosting Framework called B2HOST that allows communities to deploy and operate their own applications and data-oriented services on machines next to the data storage location. Reasons for such services next to the data can be:

- Volume of the data is too large to be transferred efficiently on demand to third party data processing and analysis facilities
- Licensing restrictions that prevent even the smallest volume of data from being copied to a third party which provides the compute facilities.

In both cases, the use of B2HOST allows for the data to remain local, with a (community-specific) service interfacing between the data and external clients.

Resource providers within EUDAT offer service hosting capabilities in tandem with their storage service. These provide access to resources such as bare-metal or virtualized machines with basic execution system platforms (operating systems with a selection of software, tools and libraries). The only allowable use of these resources is for EUDAT communities to deploy and operate data-oriented services hosted at specific data centres. Community service managers can request the appropriate resources through B2HOST. Community service managers can also offer resources and join B2HOST.