D3.2 Demonstrator Implementation Guidelines

Work Package	WP3, Blue Cloud Pilot Demonstrators
Lead Partner	IFREMER
Lead Author (Org)	CNR
Contributing Author(s)	Pasquale Pagano (CNR), Massimiliano Assante (CNR), Leonardo Candela (CNR), Cécile Nys (IFREMER), Gilbert Maudire (IFREMER)
Reviewers	Sara Garavelli (TRUST IT) Anton Ellenbroek (FAO) Guy Cochrane (EMBL-EBI)
Due Date	31.01.2020, M4
Submission Date	05.05.2020
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	06.03.2020	CNR	First version
0.2	12.03.2020	Cécile Nys (IFREMER)	First revision
0.3	07.04.2020	IFREMER	Addition of text and revision
0.4	08.04.2020	Anton Ellenbroek (FAO)	Review
0.5	10.04.2020	Gilbert Maudire (IFREMER)	Inclusion of a chapter about the demonstrator roadmap
0.6	16.04.2020	IFREMER	Versions concatenation
0.7	30.04.2020	Cécile Nys, Gilbert Maudire & Dominique Briand (IFREMER)	Inclusion of illustrations and additions after WP3 tele-meeting
0.8	05.05.2020	Guy Cochrane (EMBL)	Review
0.9	05.05.2020	IFREMER	Corrections and update after review
1.0	05.05.2020	IFREMER	Final version

Contents

Table of illustrations

Glossary

Term	Definition
ABAC	Attribute-Based Access Control
API	Application Programming Interfaces
CaR	Container-as-Resource
CPU	Container-as-Resource
DM	DataMiner
D4Science	Digital for Science
GB	Gigabyte
gHNs	gCube Hosting Nodes
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IS	Information System
RAM	**Random-access memory**
SAI	Software Algorithms Importer
S2S	Service2Service
TLS	Transport Level Security
U2S	User2Service
Vlabs	Virtual Labs
VM	Virtual Machine
VRE	Virtual Research Environment
WebUI	Web User Interface
WP	Work Package
WPS	Web Processing Service
β	Beta

Executive summary

This deliverable, D3.2 “Demonstrator Implementation Guidelines” draws first a roadmap towards the hosting of the demonstrators’ technologies by the Blue-Cloud VRE, in order to set up the Blue-Cloud Virtual Laboratory.

In a second part, this document proposes information about the services and capacities provided by the Blue-Cloud VRE and gives guidelines to minimise the effort required to transform standalone software components into services proposed by the Blue-Cloud Virtual Laboratory. Finally, its aim is to illustrate how to maximise the exploitation of the services and capacities provided by the Blue-Cloud VRE. Consequently, this second part starts with a short introduction about D4Science VRE. Then, it reports on the different integration options that are available and their benefits. Each integration option contains pointers to existing D4Science manuals and guidelines which can be found on the web and which are regularly maintained.

However, the data access mechanisms are not part of the implementation guidelines. For the time being, the demonstrators must bring some of their own data or retrieved external data to try out the VRE and their workflows. The Blue-Cloud work package 2 (WP2) provides the Blue-Cloud Discovery and Access service to obtain access to large and multidisciplinary datasets. Deliverable D2.6 “Blue Cloud Architecture” will document this aspect as part of the description of the general architecture. It will describe how to establish connections between data and the VRE using the data broker that will facilitate and harmonize access to external sources. Moreover, a Data Taming service is planned, whereby the aim will be to interoperate data formats to enable easier input for VRE processes.

This Deliverable, D3.2 “Demonstrator Implementation Guidelines”, starts with a roadmap for the demonstrators followed up with a short introduction of D4Science VREs. It then reports on the various integration options and their benefits. Each integration option contains pointers to existing D4Science manuals and guidelines which can be found on the web and which are regularly maintained.

1 Roadmap for demonstrators

As defined in the project work plan, all demonstrators must adapt their technologies and tools with the Blue-Cloud VRE in order to make them as integrated as possible in the Blue-Cloud Virtual Library. However, some pre-processing computation, demanding huge computation capacity, will not have to be integrated in the VRE. In addition, access to data, especially for very large datasets which cannot be uploaded locally to the VRE, will be described by the WP2 in D2.6 “Blue Cloud Architecture”.

This integration is well advanced for some demonstrators, thanks to previous projects (e.g. Blue Bridge), whenever these VRE technologies are really new for other demonstrators’ developers.

In order to align the level of knowledge of all demonstrator developers, this document provides, in a second part, information about the provided environment and services.

However, it appears necessary to provide practical guidance during all the demonstrator integration process in the VRE, taking benefit of the experience of the demonstrators which have already partly completed this process.

The main objective of this roadmap is that, within the next 8 months, β-versions of the demonstrators will make use of the VRE components and can be activated via the Blue-Cloud Virtual Library. This roadmap will rely on the following incremental process:

- Work-package 3 meeting (Webinar format), on the 22nd of April 2020, to present the components the Blue-Cloud VRE to all demonstrators, presented in the second part of this document, and the previous experience of already integrated demonstrators (Demonstrator #4, Demonstrator #5);
- Homework within each demonstrator to elaborate a work plan for the integration: VRE components to be used, integration planning, potential issues, more necessary information (two months, before the next Blue-Cloud General Assembly). It will have to provide preliminary feedback to technical work-packages 2 and 4;
- Series of per-demonstrator meetings (at least two meetings) to provide additional information and to study and solve potential issues (4 months). In the meantime, homework by demonstrators to update the technologies and tools used by the β-versions of the demonstrators for integration in the Blue-Cloud VRE;
- Full work-package 3 meeting to summarize experiences from all demonstrators;
- Provision of the β-versions of the demonstrators (2 months).

2 Introduction to the Blue Cloud VRE

The Blue-Cloud VRE is powered by the D4Science infrastructure. It exploits cloud-based hardware resources (*hardware layer*) through the exploitation of a *service layer* organised in four software frameworks.

The hardware layer is organized as a dynamic pool of virtual machines, supporting computation and storage. It consists of an *OpenStack*[1] installation, supporting the deployment of services in the upper layer by provision of computational and storage resources. The services layer is organized into e-infrastructure middleware, storage, and end user services. This service layer is organised into four service frameworks, which can be summarized as follow:

- ***Enabling Framework*** that includes support services for the operation of all services and the VLabs. As such it includes:
    - a *resource registry* service, to which all e-infrastructure resources (data sources, services, computational nodes, etc.) can be dynamically (un)registered and discovered by user and other services;
    - *Authentication and Authorization* services;
    - *Auditing* Services, capable of both granting and tracking access and usage actions from users;
    - a *VRE manager*, capable of deploying in the collaborative framework VREs inclusive for a selected number of "applications", generally intended as sets of interacting services;
- ***Storage Framework*** which includes services for efficient, advanced, and on-demand management of digital data, encoded as either files in a distributed file system, collections of metadata records or time series in spatial databases. These services are used by all other services in the architecture, except for the enabling framework;
- ***Analytics Framework*** that includes the services required for running user provided methods and a plethora of pre-installed standard statistical methods, provided out of the box, to compute over given input data. Both scenarios use, in a transparent way, the underlying powerful computation Blue-Cloud services (e.g. parallel computing);
- ***Collaborative framework*** that supports all VLabs deployed by the scientists. It also provides, for each VLab, *social networking* services, *user management* services, *shared workspace* services, and *analytical laboratory services,* all accessible through a WebUI.

[1] OpenStack www.openstack.org

These frameworks provide different integration options, all aimed to support user communities to benefit from the power of the Blue-Cloud computing approach – scalability, availability, capacity flexibility, security, enhanced collaboration, failover management, distributed and replicated storage, etc. – without the need to know technical details, specific technologies, or specific working practices.

- The first integration option is related to the provisioning of software methods elaborated by the community to prepare and harmonize data, which require an execution within the infrastructure.
- The second integration option is related to the provisioning and integration of services and applications into the infrastructure. Those services and applications are independent from the infrastructure. However, thanks to their integration into the infrastructure they can improve the service and performance quality.
- The third integration option is related to the provisioning of containerised software that can be outsourced to the infrastructure to improve the application in terms of scalability while promoting collaboration.

In this document, three separate sections will be encountered:

- Section 3 describes the secure context where all the integrated software will be executed. It includes key information about the operation and the service to service connection.
- Section 4 describes the provisioning, management and integration of software methods, services and applications.
- Section 5 describes the provisioning, management and integration of containerised software.

3 The D4Science Security

D4Science.org provides access to a set of services hosted by different organisations in the EU. The connection between the sites is secured with Transport Level Security (TLS), which provides communication security over the computer network.

D4Science.org ensures privacy and data integrity between two communicating computer applications. In particular, any connection between a client (e.g., a web browser) and a D4Science.org server has the following properties:

- **Private (or secure) connection** through the adoption of symmetric cryptography, which encrypts the data transmitted. The keys for this symmetric encryption are generated uniquely for each connection and are based on a shared secret (negotiated at the start of the session). The server and client negotiate the details about which encryption algorithm and cryptographic keys to use before the first byte of data is transmitted. The negotiation of a shared secret is both secure, as the negotiated secret is unavailable to eavesdroppers and cannot be obtained (even by an attacker who places himself in the middle of the connection) and reliable, as no attacker can modify the communications during the negotiation without being detected;
- **Authentication** of communicating parties using public-key cryptography. This authentication can be made optional on the client's side; however, it is ensured on the server's side;
- **Integrity for the connection** is ensured because each message transmitted includes a message integrity check using a message authentication code to prevent undetected loss or alteration of the data during transmission;
- **Forward secrecy** ensures that a future disclosure of encryption keys cannot be used to decrypt any TLS communications recorded in the past.

D4Science.org provides access to services and data via Virtual Environments, called Virtual Laboratories in Blue-Cloud. Each VLab enables services and data exploitation to the users authorized to access the VRE.

D4Science.org is empowered by a token-based authorization system compliant with Attribute-Based Access Control (ABAC) and supports several patterns for the integration of tools, services, applications and software that are presented in the following sections.

Alternative authentication models that could be used are illustrated in following figures (Figure 1 & Figure 2) and the detailed in the following paragraphs (See 3.1 The Authorization Model & 3.2 OAuth2.0).

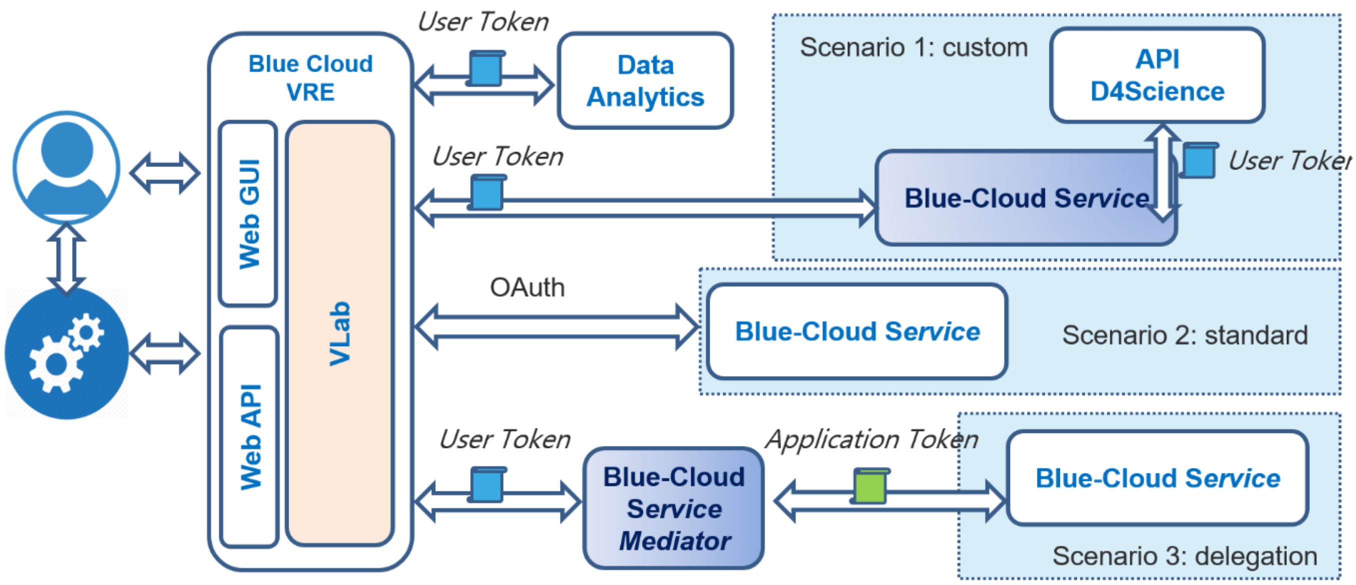
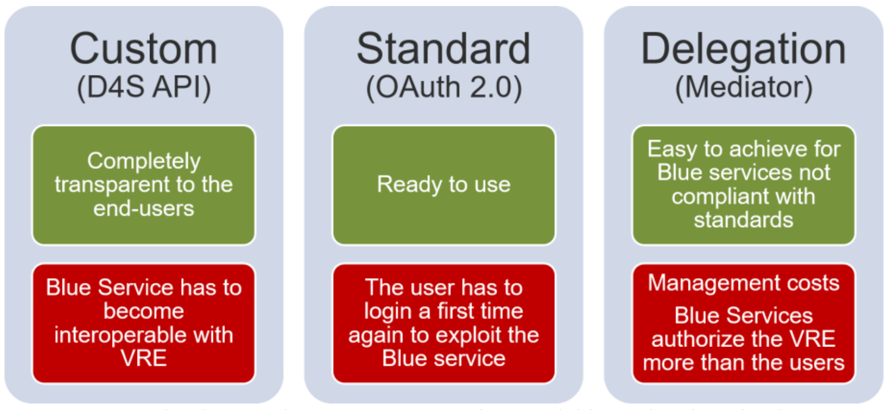


Figure 1. Security context of the VLabs on the Blue-Cloud VRE : Multiple scenarios and different solutions



Custom (D4S API)	Standard (OAuth 2.0)	Delegation (Mediator)
Completely transparent to the end-users	Ready to use	Easy to achieve for Blue services not compliant with standards
Blue Service has to become interoperable with VRE	The user has to login a first time again to exploit the Blue service	Management costs Blue Services authorize the VRE more than the users

Figure 2. The three authentication approaches available on the Blue-Cloud VRE.

3.1 The Authorization Model

The D4Science Authorization framework is a token-based authorization system. This framework is compliant with the *Attribute-Based Access Control (ABAC)*[2] that defines an access control paradigm whereby access rights are granted to users through the use of policies which combine attributes. ABAC defines access control based on attributes describing:

- 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);
- the environmental or contextual information (either the VRE or the VO where the operation is executed).

ABAC is a logical access control model that is distinguished by its access control to objects by evaluating rules against the attributes of the entities (requesting entity or target resource). ABAC relies on the evaluation of attributes of the requesting entity, attributes of the targeted resource,

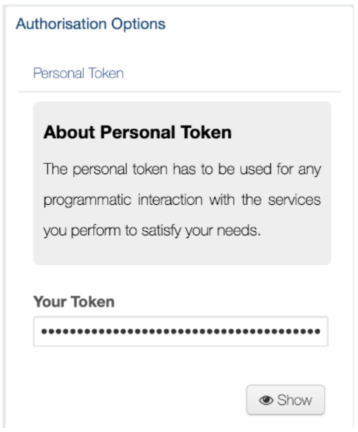
[2] https://en.wikipedia.org/wiki/Attribute-based_access_control

environment conditions, and a formal relationship or access control rule defining the allowable operations for entity-resource attribute and environment condition combinations.

The token is a string generated on request by the Authorization service for identification purposes and associated with every entity belonging to the D4Science 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 any service in 3 ways:

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

A personal token can be retrieved using the token widget deployed on every Virtual Laboratory of the Blue-Cloud VRE gateway.


Authorisation Options

Personal Token

About Personal Token

The personal token has to be used for any programmatic interaction with the services you perform to satisfy your needs.

Your Token

••••••••••••••••••••••••••••••••••••••

The gCube Authorization Framework controls access to applications to allow or prevent the clients to perform various operations in the application. This is controlled by the Authorization Service embedded in the SmartGears (See 6.1 Servlet-based container: SmartGears) framework with the help of authorization policies. The purpose of authorization policies is to control clients' accesses. The authorization policies determine at runtime whether or not a particular action is denied. You can define authorization policies that satisfy the authorization requirements using the policy language.

All the policies created in the system 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 a *user* or a *role* the access to a specific *service* or *class of services*. The S2S policies are used to deny a *service* or a *class of services* the access to specific service or class of services. To facilitate the possibility to allow access only to few clients an 'except restriction' is defined in the policies.

Three types of token are supported:

- User Token: the user token has to be used by any user for any programmatic interaction with the services.
- Qualified Token: it is a token associated with a mnemonic label. All the operations performed with this token are accounted for by the user. The mnemonic label will help the user in identifying better the different exploitation patterns of the infrastructure services.
- Application Token: It is a token associated with an application identifier. All the operations performed with this token are accounted to the specified application and not to a user. It is released by the D4Science Infrastructure Manager by any application requesting it. 2.2 Integration Patterns

3.2 OAuth2.0

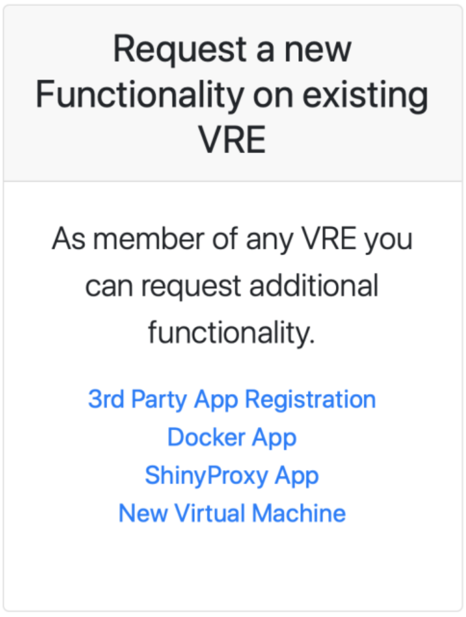
By means of the OAuth 2.0 protocol (authorised) third party applications can operate on a user's behalf over the D4Science infrastructure (while protecting the member's credentials). For more information about the OAuth authorization framework please visit the official *OAuth site*[3]. For technical details also see the *OAuth 2.0 RFC*[4].

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

To request the authorisation of a third party application it is sufficient for a member of an existing Blue-Cloud Virtual Laboratory to open a support request by accessing the D4Science support web site at https://support.d4science.org.

In the *Request a new Functionality on existing VRE*, the user has to select *3rd Party App Registration*[5]. This link opens a new interface where further information has to be provided:

- Subject: the name of the third party application;
- Description: any information about the third party application that may be useful;
- Authorized Redirect URLs: at least one URL that must be absolute, and without arguments;
- Logo URL (optional): absolute URL of the application logo.


Request a new Functionality on existing VRE

As member of any VRE you can request additional functionality.

3rd Party App Registration
Docker App
ShinyProxy App
New Virtual Machine

The request is managed by the Blue-Cloud VRE support team and processed in the shortest time possible. If all the requested information is properly specified, the request is managed in five working days.

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 .

[3] https://oauth.net/2/
[4] https://tools.ietf.org/html/rfc6749%7C
[5] https://support.d4science.org/projects/d4science-support/issues/new?issue%5Btracker_id%5D=28

4 Provisioning, management and integration

4.1 Software Methods Provision and Integration

The data processing platform (named DataMiner) is an open-source computational system built on the gCube system. The platform is fully integrated with the D4Science e-Infrastructure (Figure 3), and meets the Open Science paradigm requirements, in order 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 to use 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. DataMiner 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.

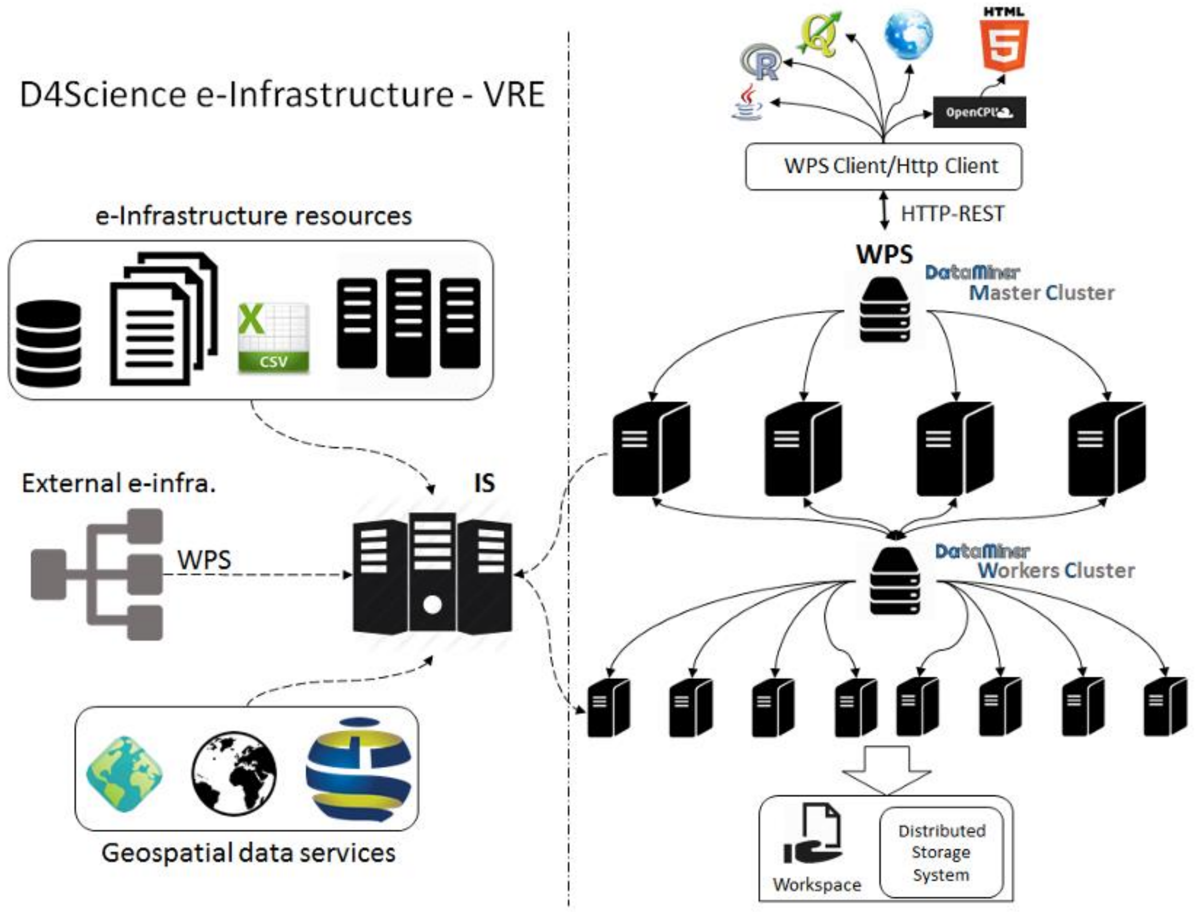


Figure 3. DataMiner (data processing platform), open-source computational system built on the gCube system. DataMiner is fully integrated with the D4Science e-Infrastructure.

The DataMiner (DM) [References 2] architecture consists of two sets of machines (clusters) that operate in a Virtual Research Environment [References 3]: the Master and the 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 equipped 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.

4.1.1 Functional specifications

DataMiner (DM) meets functional specifications related to the processing of a large variety of data types (including geospatial data) in the wider context of Big Data processing and Open Science. Other computational systems used by e-Infrastructures also parallelise the computation on several available cores/processors or machines. But DataMiner satisfies 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 accounting facilities;
- Managing and analysing Data;
- Designing and executing Workflows that combine different processes published as services.

One major advantage is that all DM services publish their capabilities using a standard, which enhances the interoperability with other external services and software, compared to using custom clients.

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 implement their own Cloud computations by invoking the Worker cluster in custom workflows.

DM services can interact with data preparation and harmonisation services. This speeds up the typical time-consuming phase of data preparation. Furthermore, a shared experimentation area allows for the reuse of results of processes and also fosters multidisciplinary experiments.

DM users can 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 make decisions.

The WPS standard behind DM can also be used by external (including desktop) software that support the WPS standard. Further, generating and storing provenance information, based on the PROV-O standard, improves the possibility to repeat and reproduce any DM task, such as experiments 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.).

4.1.2 Interface of the DataMiner system

DataMiner offers a Web GUI to the users of a VRE (Figure 4). On the left-hand panel (Figure 4 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

"Describe Process" operation to get the descriptions of the inputs and outputs. When a user selects a process, in the right-hand 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 (Figure 4 b), where a summary sheet of the provenance of the experiment can be obtained ("Show" button, Figure 4 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 (Figure 4 d), with provenance information attached that refers to the computation that used the dataset.

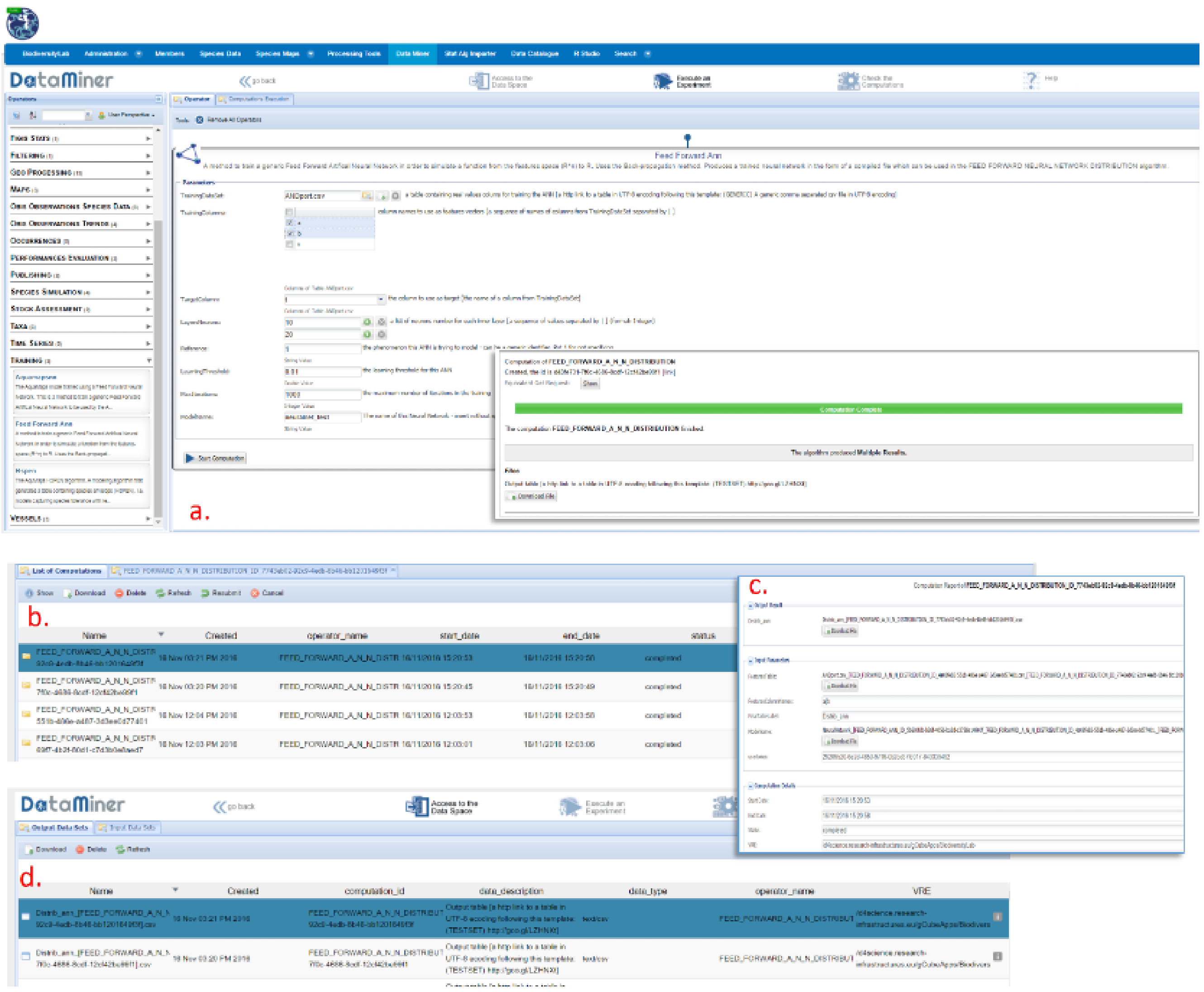
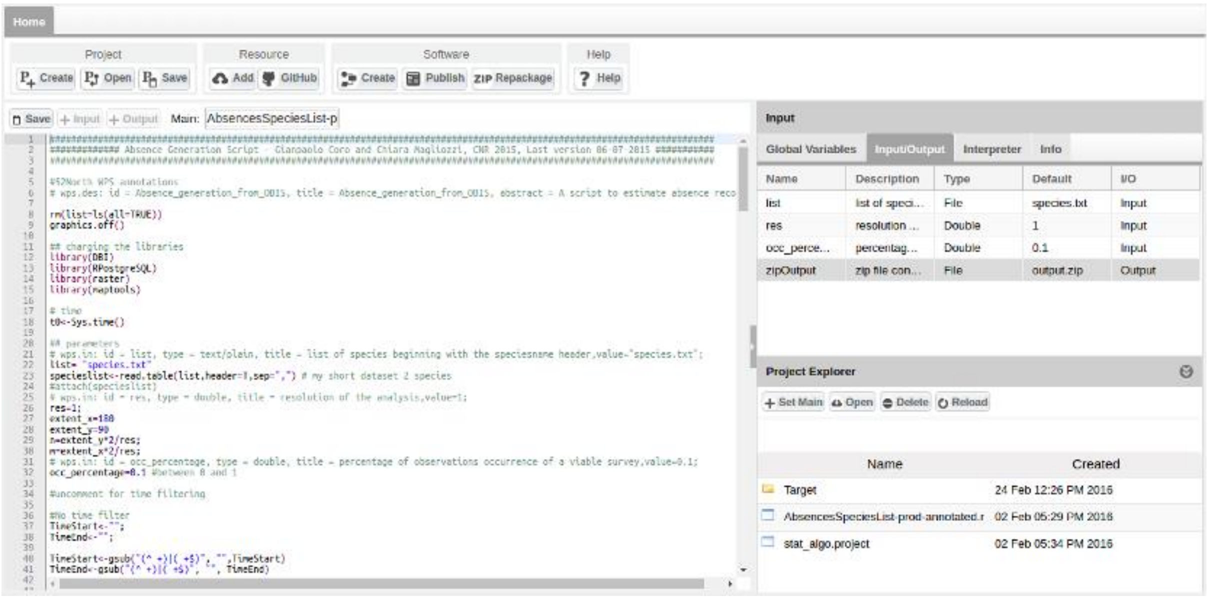


Figure 4. DataMiner Administrator and User interfaces.

4.1.3 Software Algorithms Integration

Prototype scripting is the basis of most models in environmental sciences. Scientists developing prototype scripts (e.g. using R and Matlab) often need to share results and make their models available to others for review or for exploitation with new data. To this aim, DM allows to publish scripts as-a-Service, preferably under a recognized standard (e.g. WPS). The Software Algorithms Importer, SAI [References 5], is an interface that allows to import scripts into DataMiner (DM). DataMiner in turn publishes these scripts as a service and manages multi-tenancy and concurrency. Additionally, it allows users to update their scripts without time-consuming software re-deploying procedures.

In summary, SAI produces processes that run on the DataMiner Cloud computing platform and are accessible via the WPS standard.



Main: AbsencesSpeciesList-p

```
##########################################################################################
############# 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)
```

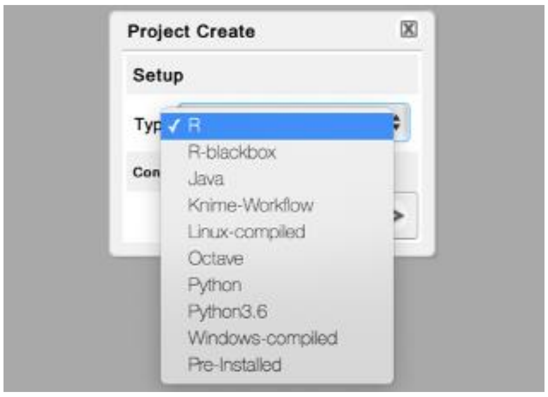
Name	Description	Type	Default	I/O
list	list of speci...	File	species.txt	Input
res	resolution ...	Double	1	Input
occ_perce...	percentag...	Double	0.1	Input
zipOutput	zip file con...	File	output.zip	Output

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 5. Interface to import a R process on DataMiner.

The SAI interface for R scripts resembles the R Studio environment (Figure 5), a popular IDE for R scripts, in order to make it accessible to script providers. 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 users to indicate the version of the R interpreter and the packages required by the script and the "Info tab" allows users to indicate 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) accounting facilities.

4.2 Services and Applications Provision and Integration

An application or service is defined as a stand-alone system running on a remote server and offering one or more functionality either via web User Interfaces or via Application Programming Interfaces (APIs).

A service or an application, hereafter *service*, can be integrated in the Blue-Cloud VRE via different patterns, each pattern is characterized by an effort required for the implementation and a resulting benefit. Integration is not achieved by only hosting the service in the infrastructure, there are mutual benefits for both the *service* and the infrastructure in performing this integration: the *service* may exploit the infrastructure frameworks and capabilities, such as the Analytics and Storage frameworks and the hardware and data resources associated to them. The infrastructure, in turn, would gain in terms of broadening its offering by making the integrated *service* part of it.

4.2.1 Gold level integration pattern

A gold level of integration is achieved by making the *service* available on an authorised SmartGears node of the infrastructure. A SmartGears node (See Appendix 1) is a customised Java 8 Servlet container (Apache Tomcat 8), this implies that in this integration pattern the *service* has to be able to run on a Java Virtual Machine.

The benefits of this integration pattern are that the *service* becomes fully interoperable with the infrastructure and can delegate the following functions:

- Authorisation and Authentication;
- Users and Roles Management;
- Auditing and Tracing;
- Monitoring.

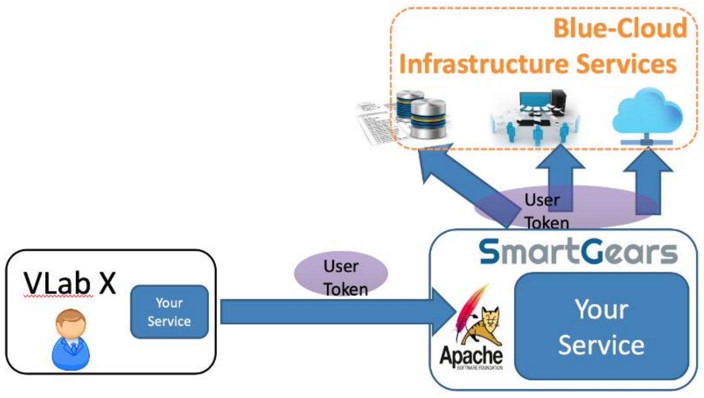


Figure 6. Gold level integration pattern.

Figure 6 depicts the higher level of integration, where the infrastructure provides the *service* with authentication, authorization, monitoring and auditing. The User Token is passed transparently among service calls and the calls arriving to the *service* are already authorized.

4.2.2 Silver level integration pattern

A silver level of integration is achieved by passing the User Token to the *service*, which in this case runs over its own technology server (inside or outside the infrastructure premises), by means of a HTTP GET request over the HTTPS (HyperText Transfer Protocol Secure) protocol.

The User Token is not passed transparently among service calls, the *service* reads the User Token from the HTTP GET request and uses it to perform the infrastructure service calls needed for its functions. The very first request would be to validate and resolve (obtain user identifier and VLab identifier) the token.

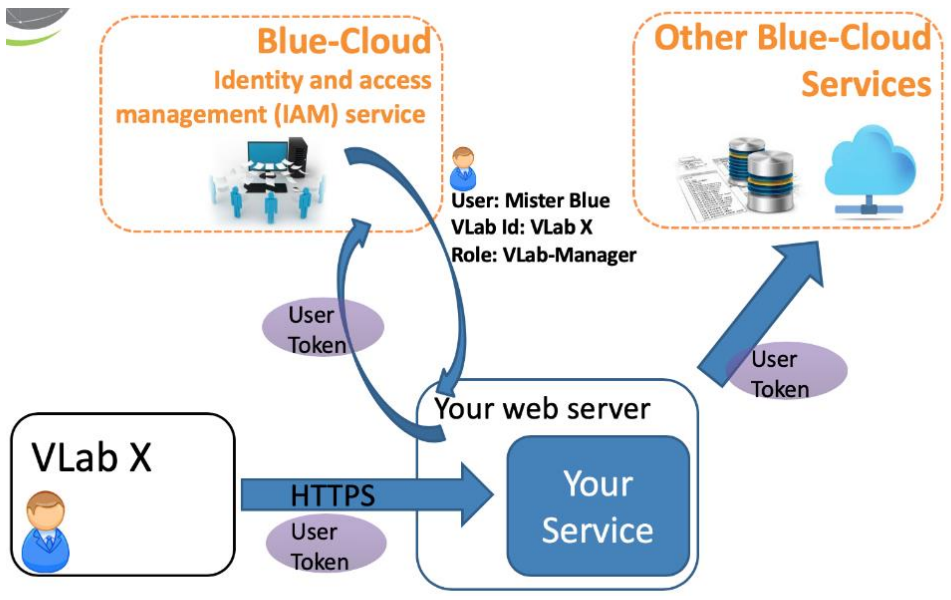


Figure 7. Silver level integration pattern.

The benefits of this integration pattern are that the *service* becomes interoperable with the infrastructure and may delegate the following functions:

- Authorisation and Authentication;
- Users and Roles Management;
- Monitoring (only if the *service* is hosted within the infrastructure premises).

Figure 7 depicts the silver level of integration, where the infrastructure provides the *service* with the User Token via a HTTP GET request. The User Token can be used to obtain user identifier (Mister Blue in figure), the roles (VLab-Manager in figure) of the user and the VLab identifier (VLab X in figure) and successively perform other service to service calls required for the *service* functions.

4.2.3 Bronze level integration pattern

A bronze level of integration is a lightweight integration with few benefits and can be used only in particular cases where the *service* needs to contact some Blue-Cloud service but cannot perform operations on behalf of the users. The figure below shows an example of a *service* needing authorization to access the Storage service of the Blue-Cloud VRE for some of its functions.

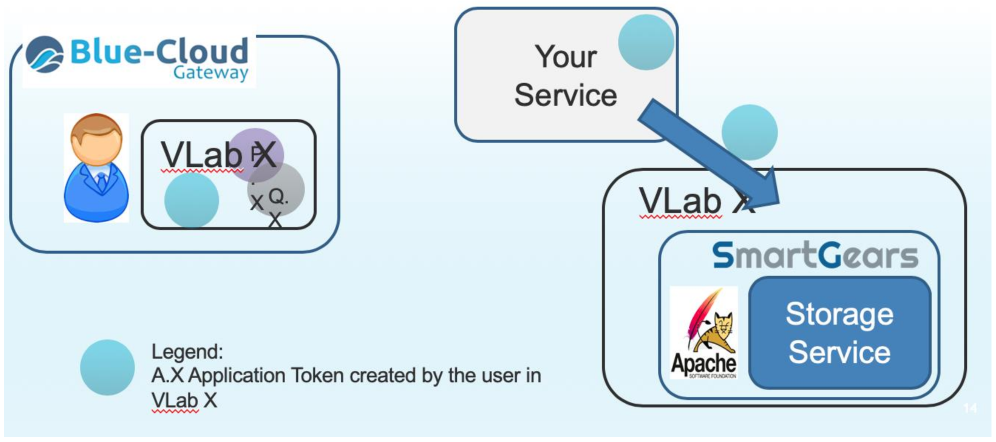


Figure 8. Bronze level integration pattern.

Figure 8 depicts the bronze level of integration, where the infrastructure provides the *service* with an Application token that can be stored in the service to perform service calls required for the *service* functions.

4.3 Interactive Exploratory Computing

The Blue-Cloud VRE offers access to two environments providing a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management: RStudio and JupyterLab.

4.3.1 RStudio

RStudio is the premier integrated development environment for R. It provides syntax highlighting, code completion, and smart indentation allowing to execute R code directly from the source editor.

RStudio provided by the Blue-Cloud VRE is integrated within the Workspace making it possible to read, store and update any content the user has previously stored in the Workspace. The connection to the Workspace allows also to save computations executed in RStudio in the workspace and to share them with other users or preserve it across RStudio sessions.

RStudio is available to all members of the Blue-CloudLab environment by a clicking on the RStudio 'button' in the blue-button bar (Figure 9).

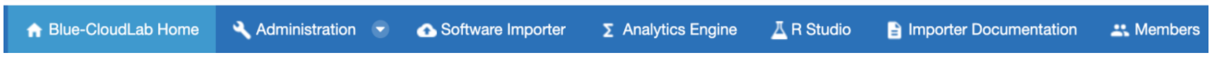
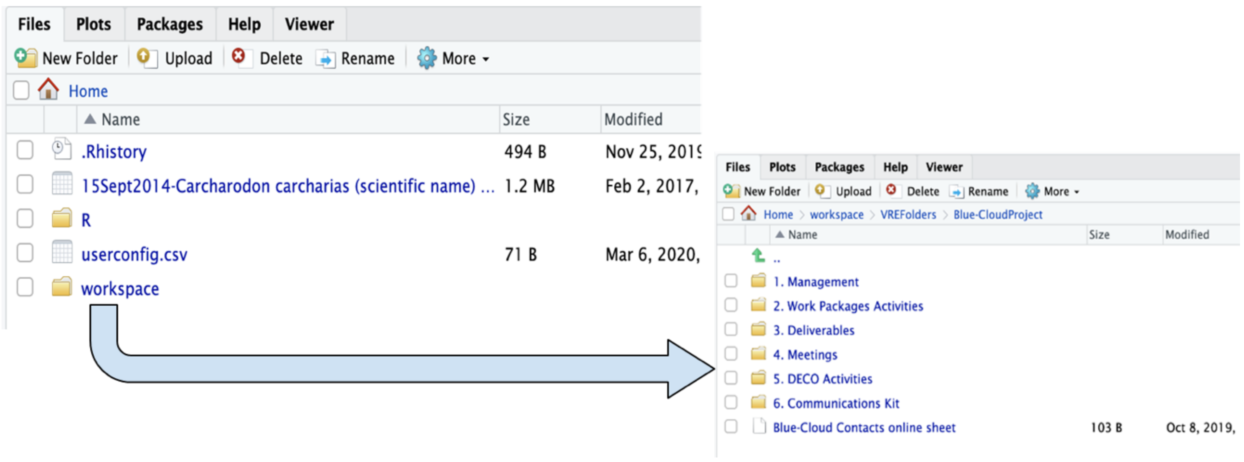


Figure 9. Blue-button (menu) bar of a VRE.

One RStudio instance is assigned from the RStudio cluster of the Virtual Laboratory. This means that the same user may be assigned different instances across user sessions. It is therefore fundamental to save any file generated in the RStudio session in the personal Workspace.

The Workspace appears as a folder in the RStudio interface (Figure 10) and can be navigated as any other local folder.



Home

Name	Size	Modified
.Rhistory	494 B	Nov 25, 201
15Sept2014-Carcharodon carcharias (scientific name) ...	1.2 MB	Feb 2, 2017,
R		
userconfig.csv	71 B	Mar 6, 2020,
workspace		

Home > workspace > VREFolders > Blue-CloudProject

Name	Size	Modified
..		
1. Management		
2. Work Packages Activities		
3. Deliverables		
4. Meetings		
5. DECO Activities		
6. Communications Kit		
Blue-Cloud Contacts online sheet	103 B	Oct 8, 2019,

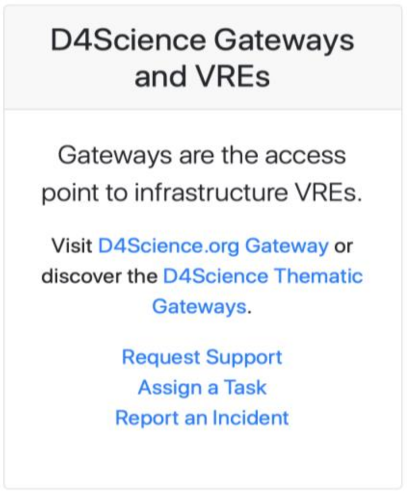
Figure 10. RStudio interface with the Workspace folder.

The RStudio environment can be requested and activated in any Virtual Laboratory.

To request the provision of RStudio it is sufficient to be a member of an existing Blue-Cloud Virtual Laboratory and open a support request by accessing the D4Science support web site at https://support.d4science.org.

In the D4Science *Gateways and VRES*, the user has to select *Request Support*. This link opens a new interface (Figure 11) where a number of information have to be provided:

- Subject: the name of the demonstrator;
- Description: any information about the R environment the user needs to know. This description must include the list of R packages that should be made available in any RStudio instance of the RStudio cluster. This is particularly relevant for the user since it will avoid the setup of the RStudio instance at any session.



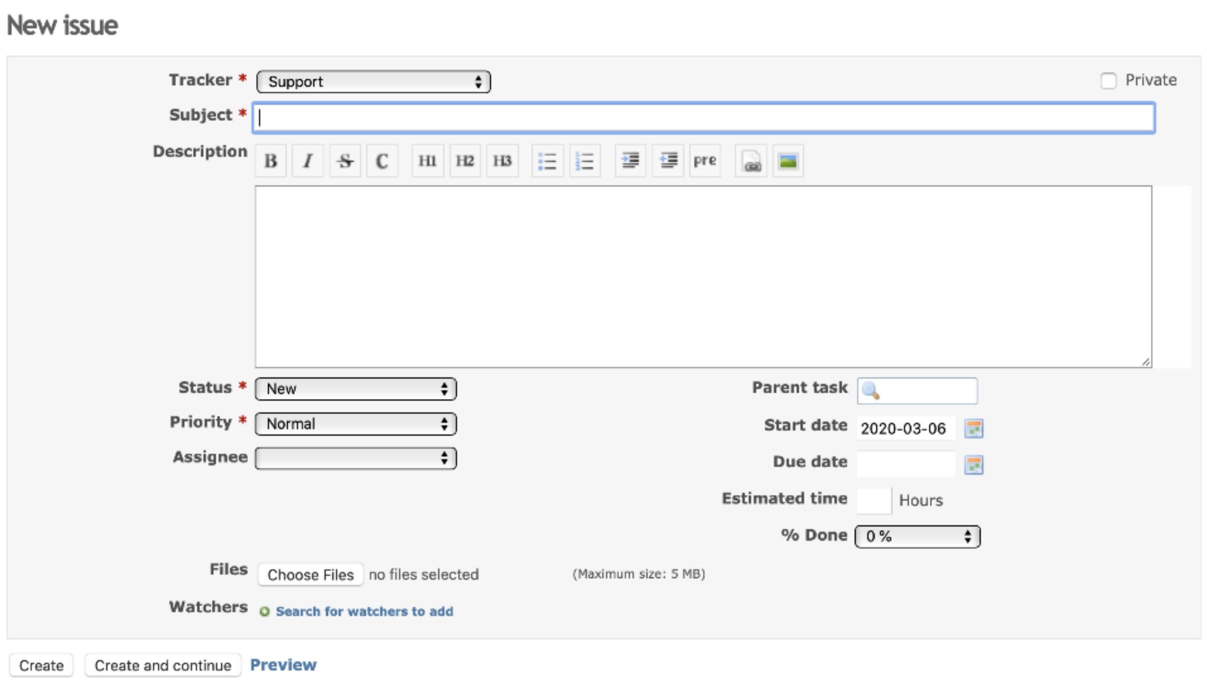
D4Science Gateways and VREs

Gateways are the access point to infrastructure VREs.

Visit D4Science.org Gateway or discover the D4Science Thematic Gateways.

Request Support
Assign a Task
Report an Incident

The request is managed by the Blue-Cloud VRE support team and processed in the shortest time possible. If all the requested information is properly specified, the request is managed in two working days.



New issue

Tracker *	Support
Private	☐
Subject *	
Description	
Status *	New
Priority *	Normal
Assignee	
Parent task	
Start date	2020-03-06
Due date	
Estimated time	Hours
% Done	0 %
Files	Choose Files no files selected (Maximum size: 5 MB)
Watchers	Search for watchers to add

Figure 11. Request Support interface on the D4Science Gateways and VREs.

4.3.2 JupyterLab

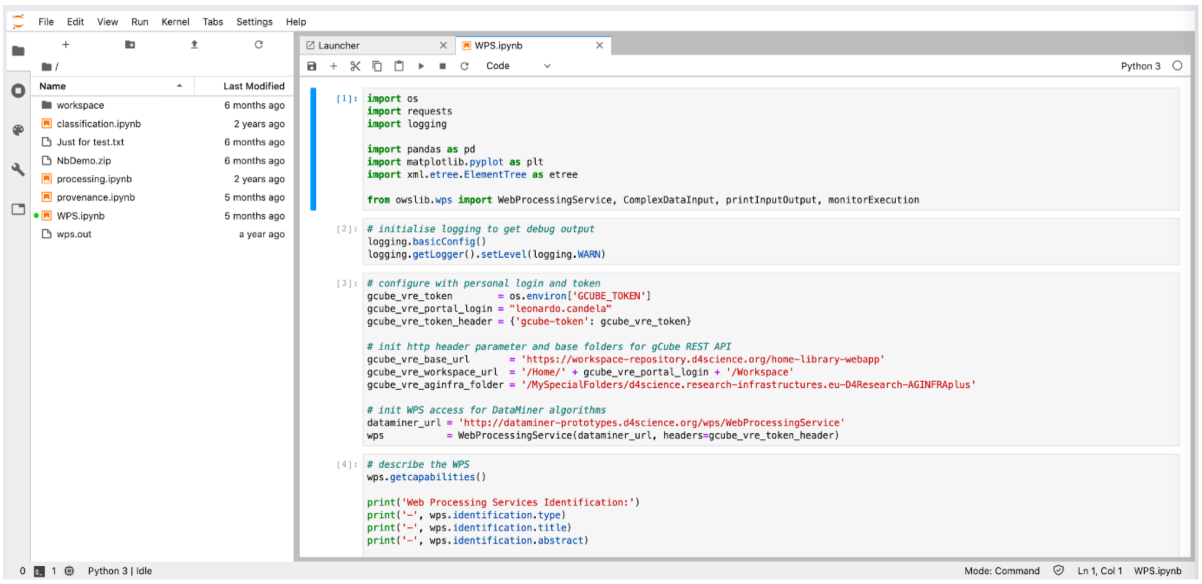
Jupyter notebooks are useful for documenting and recording analytical processes. Notebooks are documents that combine live runnable code with narrative text (Markdown), equations (LaTeX), images, interactive visualizations and other rich output. The Blue-Cloud VRE notebook platform is served by *JupyterLab*[6].

The JupyterLab provided by Blue-CLoud VRE is integrated with:

- the Workspace (appearing in the JupyterLab UI) thus making it possible to seamlessly read, store and update any content the user has previously stored in the Workspace directly (including content stored into VRE folders). This integration allows to store the notebooks produced by JupyterLab in the Workspace and from there they can be shared and published;
- the DataMiner thus making it possible to seamlessly invoke any DataMiner algorithm through an extended version of the OWSLib library that takes care of interfacing with DataMiner via the WPS protocol and the user authorization token (See 3.1 The Authorization Model);

JupyterLab (Figure 12) is available to all members of any VRE where it has been deployed by clicking on the button of the VRE menu.

[6] https://jupyter.org/



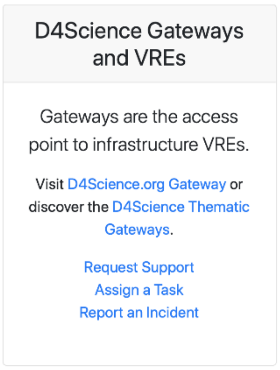
Name	Last Modified
workspace	6 months ago
classification.ipynb	2 years ago
Just for test.txt	6 months ago
NbDemo.zip	6 months ago
processing.ipynb	2 years ago
provenance.ipynb	5 months ago
WPS.ipynb	5 months ago
wps.out	a year ago

```
[1]: import os
     import requests
     import logging

     import pandas as pd
     import matplotlib.pyplot as plt
     import xml.etree.ElementTree as etree

     from owslib.wps import WebProcessingService, ComplexDataInput, printInputOutput, monitorExecution

[2]: # initialise logging to get debug output
     logging.basicConfig()
     logging.getLogger().setLevel(logging.WARN)

[3]: # configure with personal login and token
     gcube_vre_token        = os.environ['GCUBE_TOKEN']
     gcube_vre_portal_login = "leonardo.candela"
     gcube_vre_token_header = {'gcube-token': gcube_vre_token}

     # init http header parameter and base folders for gCube REST API
     gcube_vre_base_url       = 'https://workspace-repository.d4science.org/home-library-webapp'
     gcube_vre_workspace_url  = '/Home/' + gcube_vre_portal_login + '/Workspace'
     gcube_vre_aginfra_folder = '/MySpecialFolders/d4science.research-infrastructures.eu-D4Research-AGINFRAplus'

     # init WPS access for DataMiner algorithms
     dataminer_url = 'http://dataminer-prototypes.d4science.org/wps/WebProcessingService'
     wps           = WebProcessingService(dataminer_url, headers=gcube_vre_token_header)

[4]: # describe the WPS
     wps.getcapabilities()

     print('Web Processing Services Identification:')
     print('-', wps.identification.type)
     print('-', wps.identification.title)
     print('-', wps.identification.abstract)
```

Figure 12. JupyterLab in Blue-Cloud VREs.

JupyterLab can be requested and activated in any Virtual Laboratory.

To request the provision of JupyterLab it is sufficient to be a member of an existing Blue-Cloud Virtual Laboratory and open a support request by accessing the D4Science support web site at https://support.d4science.org.



D4Science Gateways and VREs

Gateways are the access point to infrastructure VREs.

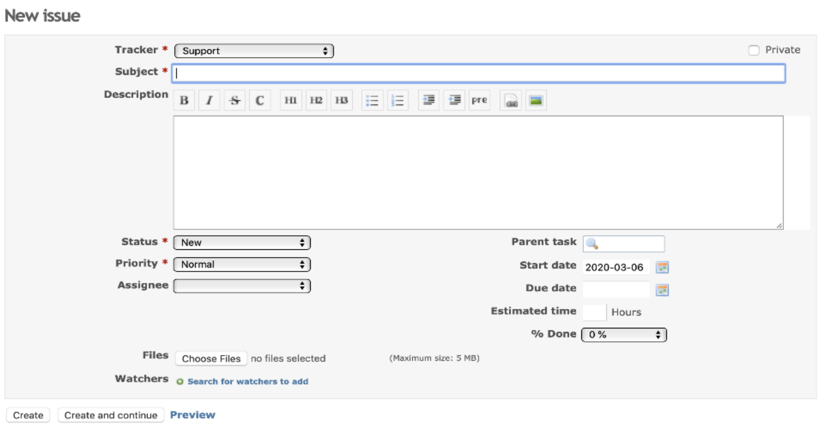
Visit D4Science.org Gateway or discover the D4Science Thematic Gateways.

Request Support
Assign a Task
Report an Incident

In the D4Science *Gateways and VRES*, the user has to select *Request Support*. This link opens a new interface (Figure 13) where a number of information have to be provided:

- Subject: the name of the demonstrator;
- Description: any information about the JupyterLab environment the user needs to know. This description must include the list of packages and kernels the JupyterLab instance that are useful for the VRE to be provided with.

The Blue-Cloud VRE support manages and processes, in the shortest time possible, the requests made by the users If all the requested information is properly specified, the request is managed in two working days.



New issue

Tracker *	Support		Private
Subject *			
Description			
Status *	New	Parent task	
Priority *	Normal	Start date	2020-03-06
Assignee		Due date	
		Estimated time	Hours
		% Done	0 %
Files	Choose Files no files selected (Maximum size: 5 MB)		
Watchers	Search for watchers to add		

Figure 13. Request Support interface on the D4Science Gateways and VREs.

5 Containerised Software Provision and Integration

As reported in [References 6], containerization is an alternative virtualization. It encapsulates or packages software code and all its dependencies so that it can run on the Blue-Cloud infrastructure.

Containerization bundles the application code together with the related configuration files, libraries, and dependencies required for it to run. This single package of software or "container" is abstracted away from the host operating system, and stands alone and is portable. Containers are inherently smaller in capacity than a Virtual Machine (VM) and require less start-up time, allowing far more containers to run on the same compute capacity as a single VM.

Containerization allows applications to run anywhere. Containerized applications are isolated in that they do not bundle in a copy of the operating system.

This approach is particularly suitable for the Blue-Cloud VRE since each containerized application is isolated and operates independently of others that are provided by different groups working in different domains. The failure of one container does not affect the continued operation of any other containers and this allows to deliver production-quality operations even in presence of development activities performed by other users. Technical issues within one container never force any downtime in other containers.

Moreover, the isolation of applications as containers prevents the injection of malicious code aimed to attack other containers or the host system and this reduces the costs of operation and making it possible for the Blue-Cloud VRE to automate the installation, scaling, and management of containerized workloads and services.

5.1 Docker Applications

A *Docker Swarm*[7] cluster is available to deploy and run Docker containers. Only Docker containers are supported at this time and they can be deployed in different ways:

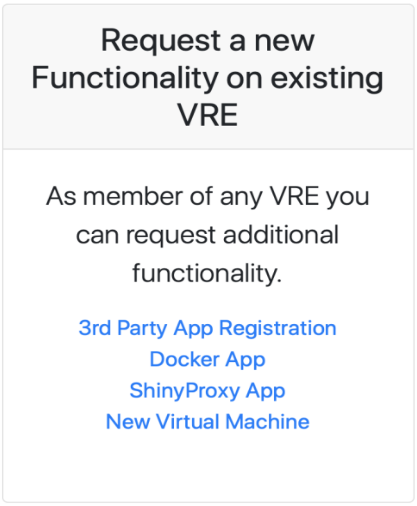
- A public container already available in *Docker Hub*[8] or any other public container registry.
- A build of a *public image* can be requested, which must be accessible from the D4Science *Jenkins*[9] instance so that the process can be automated. The result container image will be uploaded into Docker Hub and deployed into the cluster.
- A build of a *private image* can be requested, which must be accessible from the D4Science Jenkins instance so that the process can be automated. The result container image will be uploaded into the D4Science's private Registry and deployed into the cluster.

To request the deployment and provision of a Docker container is sufficient to be a member of an existing Blue-Cloud Virtual Laboratory and open a *Support Request* by accessing the D4Science support web site at https://support.d4science.org.

[7] https://docs.docker.com/engine/swarm/
[8] https://hub.docker.com/
[9] https://jenkins.d4science.org/

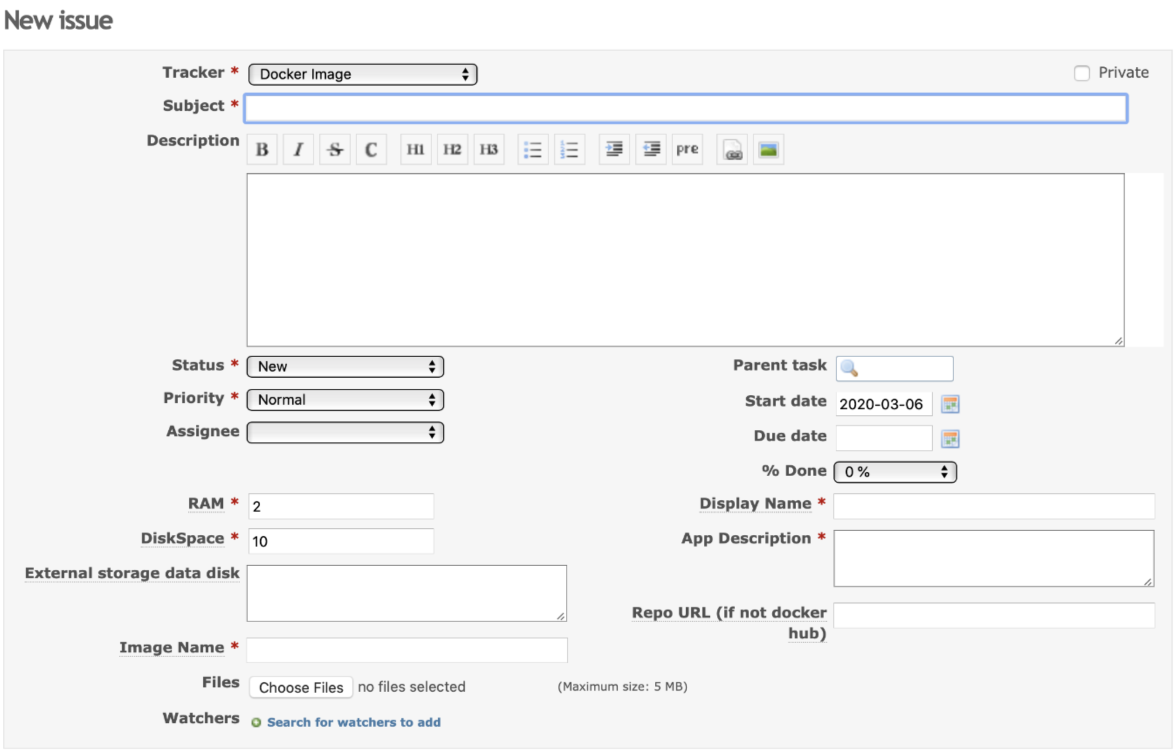
In the *Request a new Functionality on existing VRE*, the user has to select *Docker App*. This link opens a new interface (Figure 14) where a number of information have to be provided:

- Subject: the name of the demonstrator;
- Description: any information about the container that may be useful to proper provision it;
- RAM, DIskSpace: memory and disk resources required by the container;
- Image Name: the name of the image on either Docker HUB or another repository;
- Display Name: the name to assign in the VLAB to start the containerised application;
- App Description: any information about the application that may be useful to proper provision it;
- External storage data disk: volumes that store permanent data (if any).


Request a new Functionality on existing VRE

As member of any VRE you can request additional functionality.

3rd Party App Registration
Docker App
ShinyProxy App
New Virtual Machine


New issue

Tracker *	Docker Image
Private	☐
Subject *	
Description	
Status *	New
Priority *	Normal
Assignee	
Parent task	
Start date	2020-03-06
Due date	
% Done	0 %
RAM *	2
DiskSpace *	10
External storage data disk	
Display Name *	
App Description *	
Repo URL (if not docker hub)	
Image Name *	
Files	Choose Files no files selected (Maximum size: 5 MB)
Watchers	Search for watchers to add

Figure 14. "Request a new Functionality on existing VRE" support interface for requesting to add a "Docker App"

The request is managed by the Blue-Cloud VRE support team and processed in the shortest time possible. If all the requested information is properly specified, the request is managed in two working days.

5.2 Shiny Applications

Shiny is an R package that makes it easy to build interactive web apps straight from R. However, when multiple instances/users attempt to start a Shiny App at the same time, only a single R session is initiated on the serving machine. This is problematic and cannot be adopted as a solution by the Blue-Cloud VRE if one user starts a process for example that takes several seconds to complete, all other users will need to wait until that process has completed before any other tasks can be processed. For this reason, Blue-Cloud requires that the Shiny App is containerised. One of the benefits of deploying a containerised Shiny App is that each new instance will run in its own R session.

To implement this approach, ShinyProxy is used. ShinyProxy uses time-tested and mature enterprise Java technology bundled nicely as a *Spring boot*[10] web application. When deploying a Shiny application with ShinyProxy, the application is simply bundled as an R package and installed into a Docker image. Every time a user runs an application, a container spins up and serves the application. This has numerous advantages:

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

A *Shiny (proxy) app*[11] can be deployed in different ways:

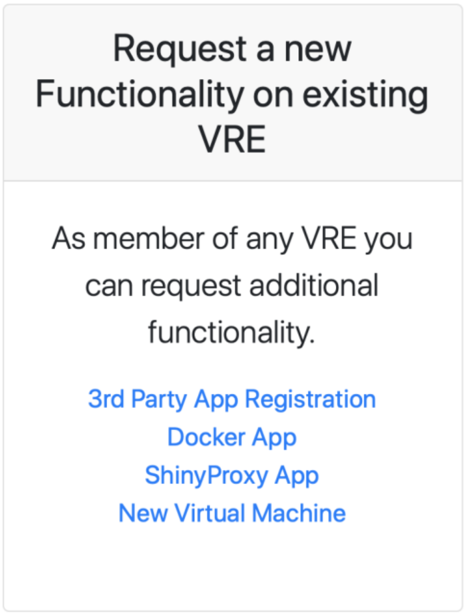
- As a public app available in *Docker Hub*[12] or any other public container registry. In this case, the image name and the run command are the only requirements.
- A build of a public image can be requested, this must be accessible from our *Jenkins*[13] instance so that the process can be automated. The resulting container image will be uploaded into *Docker Hub*[14].
- A build of a private image can be requested, this must be accessible from our *Jenkins*[15] instance so that the process can be automated. The result container image will be uploaded into the D4Science's private registry.

To request the deployment and provision of a Shiny App is sufficient to be a member of an existing Blue-Cloud Virtual Laboratory and open a support request by accessing the D4Science support web site at https://support.d4science.org.

[10] https://spring.io/projects/spring-boot
[11] https://www.shinyproxy.io/
[12] https://hub.docker.com/
[13] https://jenkins.d4science.org/
[14] https://hub.docker.com/
[15] https://jenkins.d4science.org/

In the *Request a new Functionality on existing VRE*, the user has to select *ShinyProxy App*. This link opens a new interface (Figure 15) where a number of information have to be provided:

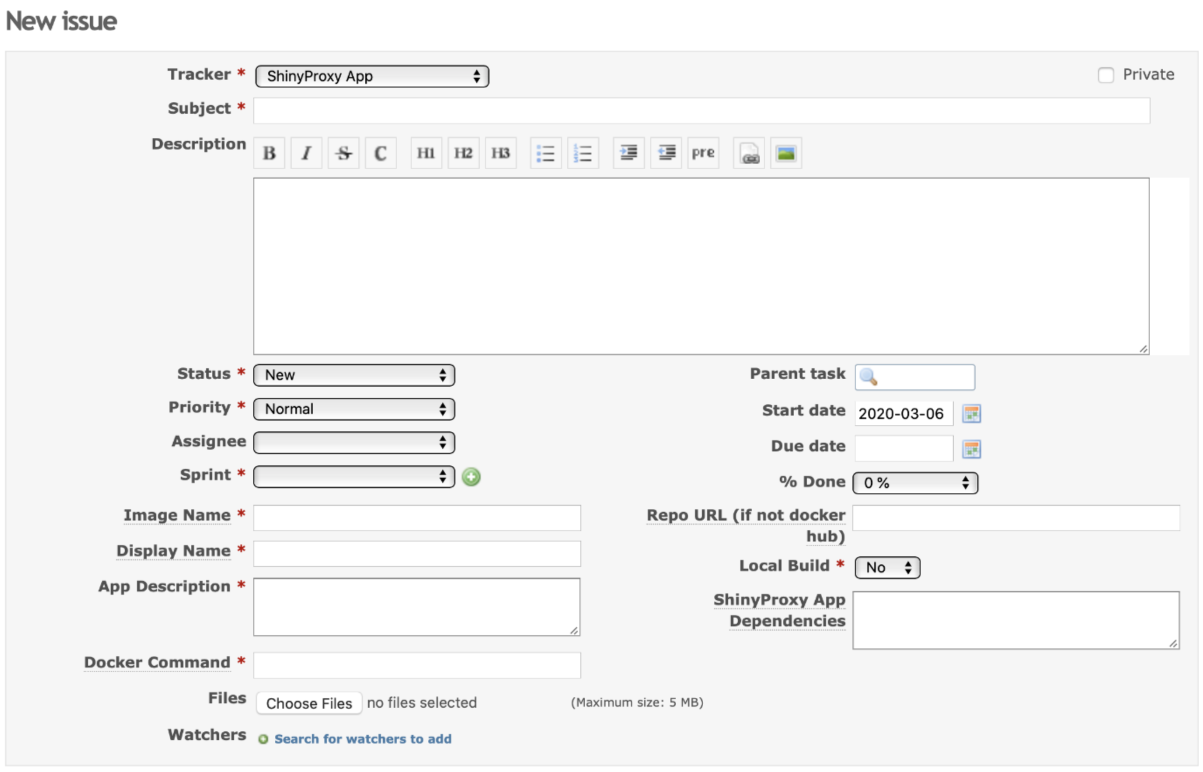
- Subject: the name of the demonstrator;
- Description: any information about the ShinyProxy App that may be useful to proper provision it;
- Image Name: the name of the image on either Docker HUB or another repository;
- Repo URL: container repository, if not docker hub. Or a code repository if a local build has been requested;
- Display Name: the name to assign in the VLAB to start the containerised application;
- App Description: any information about the application that may be useful to proper provision it;
- Docker Command: the docker command required to start-up the ShinyProxy App;
- Local Build: Set to Yes if you are requesting a public or private build. Specify in the Description field if it must be private.



Request a new Functionality on existing VRE

As member of any VRE you can request additional functionality.

3rd Party App Registration
Docker App
ShinyProxy App
New Virtual Machine



New issue

Tracker *	ShinyProxy App		Private
Subject *			
Description			
Status *	New	Parent task	
Priority *	Normal	Start date	2020-03-06
Assignee		Due date	
Sprint *		% Done	0 %
Image Name *		Repo URL (if not docker hub)	
Display Name *		Local Build *	No
App Description *		ShinyProxy App Dependencies	
Docker Command *			
Files	Choose Files no files selected (Maximum size: 5 MB)		
Watchers	Search for watchers to add		

Figure 15. "Request a new Functionality on existing VRE" support interface for requesting to add a "ShinyProxy App"

6 Appendix

6.1 Servlet-based container: SmartGears

SmartGears is a set of Java libraries that turn Servlet-based containers and applications into D4Science resources, transparently.

In this section, we introduce SmartGears and explain how it is an improvement over existing 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 D4Science 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:

- 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 D4Science 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 for 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 D4Science 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 life cycle 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 D4Science resources. Details about their contents are available in the *gCube Wiki Appendices*[16].

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*[17].

[16] https://wiki.gcube-system.org/gcube/SmartGears#Appendices
[17] https://wiki.gcube-system.org/gcube/SmartGears_Web_Hosting_Node_(wHN)_Installation

6.2 Documentation

- Developers website: to get information about a set of commonly used APIs: Dev Web Site (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) Profile and Social Networking RESTful Service (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: StorageHub REST API (https://gcube.wiki.gcube-system.org/gcube/StorageHub_REST_API);
- Metadata Catalogue (gCat) API: to learn how to publish and search collections of metadata for items including data, services, and related information objects: gCAT REST API (https://wiki.gcube-system.org/gcube/GCat_Service);
- Information System API: to learn how to interact with resources hosted in the Infrastructure and its Vlab : Information System REST API (https://dev.d4science.org/swagger/registry/);
- Authorisation framework :
  - Authorization (https://dev.d4science.org/authorization/),
  - Authorization Framework (https://wiki.gcube-system.org/gcube/Authorization_Framework).
- New Methods/Algorithms for DataMiner: to learn how to implement custom Methods and Algorithms for DataMiner: Software Algorithm Importer (https://wiki.gcube-system.org/gcube/Category:Statistical_Algorithms_Importer)
- Supported languages for new Methods/Algorithms for DataMiner: Create a new project with SAI (https://wiki.gcube-system.org/gcube/Statistical_Algorithms_Importer:_Create_Project)
- DataMiner online interfaces
  - Web: DataMiner Manager (https://wiki.gcube-system.org/gcube/DataMiner_Manager),
  - Web Processing service: Web Processing Service | OGC (https://www.opengeospatial.org/standards/wps).
- DataMiner overview: Data Mining Facilities (https://wiki.gcube-system.org/gcube/Data_Mining_Facilities);
- Spatial Data Infrastructure capabilities
  - SDI-Service (https://gcube.wiki.gcube-system.org/gcube/SDI-Service),
  - Spatial Data Storage and Publishing (https://gcube.wiki.gcube-system.org/gcube/Spatial_Data_Storage_and_Publishing),
  - Spatial Data Discovery and Access (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 : D4Science Support (https://support.d4science.org/)

7 References

1. *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.* (2019). **The gCube System: Delivering Virtual Research Environments as-a-Service**. Future Generation Computer Systems, Vol. 95

2. *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

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

4. *M. Assante, L. Candela, D. Castelli, R. Cirillo, G. Coro, L. Frosini, L. Lelii, F. Mangiacrapa, P. Pagano, G. Panichi, F. Sinibaldi*. (2019). **Enacting open science by D4Science**, Future Generation Computer System

5. *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.*

6. *IBM Cloud Education,* **Containerization (May 2019)**, consulted February 2020 https://www.ibm.com/cloud/learn/containerization