An EU-Canada joint infrastructure
for next-generation multi-Study Heart research

Deliverable D2.1:

Initial infrastructure framework and documentation

Reference	D2.1_euCanSHare_BSC_07112019
Lead Beneficiary	BSC
Author(s)	Josep Ll. Gelpí, Laia Codó, Alejandro Canosa
Dissemination level	Public
Type	Report
Official Delivery Date	30th November 2019
Date of validation by the WP Leader	28th November 2019
Date of validation by the Coordinator	29th November 2019
Signature of the Coordinator	[illegible]

euCanSHare is funded by the European Union's H2020 Framework under Grant Agreement 825903.

Version Log

Issue Date	Version	Involved	Comments
19th Nov 2019	0.2	BSC	Initial complete draft.
20th Nov 2019		UB	First review
27th Nov 2019		UB	Review by Project Manager and Project Coordinator
29th Nov 2019	1.0	BSC	Revised and corrected final version.

Executive Summary

EuCanSHare aims to establish a unified environment for cardiovascular data sharing and analysis. The euCanSHare data platform will provide support for i) a unified Data Catalogue combining in a single site all data managed by the project; ii) a platform to manage data access credentials; iii) a computational platform that combines controlled access to data, analysis and visualization tools.

This document describes the technological foundations of the euCanSHare data portal. It is a cloud-based environment providing the computational infrastructure for data management and analysis, an authentication and authorization system, and the necessary user interfaces. The document is organized as follows: Section 1 describes the motivation and overall strategy; section 2 summarizes the design of the infrastructure; software components are described in detail in section 3 and finally, section 4 outlines the development roadmap until the release of the first complete prototype (M24).

Table of Contents

1 Introduction

1.1 Motivation and strategy

euCanSHare aims to establish a unified environment for cardiovascular data sharing and analysis. euCanSHare will provide to the cardiac research community the first centralised, secure and easy-to-use platform to leverage European and Canadian cardiovascular research data and technologies, improve data discoverability, and lead to cutting-edge collaborative research in the domain of cardiovascular personalised medicine. The main issue and the ultimate motivation to build such infrastructure is the present **high fragmentation** of cardiovascular data. Phenotypic data is spread in a plethora of institutions, each of them with **particular data access policies**, and incompatible data formats. Some of the data providers have data access platforms but others **lack programmatic data access**. None of them provide an infrastructure for *in situ* data analysis. Genotypic or image data may be available, but it is not necessarily linked to general bioinformatics resources available in Europe, and in consequence global analysis done at these levels are not available. euCanSHare aims to solve some of these issues by generating a unified platform.

Through the integrated platform, researchers should be able to identify in a **common catalogue** studies and datasets provided by project's supported cohorts and associated repositories like the European Genome-Phenome Archive (EGA)[1], EuroBioImaging (EuBI)[2], or Biobanking and Biomolecular Resources Infrastructure (BBMRI-ERIC)[3]. In the developed scenario users will apply for data access through a specialized **Data Access Manager** and download data to a personal workspace for further analysis. Data providers, in turn, would be able to register their datasets in the catalogue and manage access rights to them. Developers would deploy analysis tools and get them executed in a controlled environment. The **computational infrastructure** should provide the means for: data management and data communication between providers and the central data infrastructure, user authentication and management of data access credentials, a virtual research environment providing private workspace and an execution platform for analysis tools, and data visualization. Following previous developments, euCanSHare's computational infrastructure will be cloud-based. This provides maximum flexibility in the combination of heterogeneous software deployments, while adding the possibility to deploy the full system in additional computational sites, for instance, to deploy analysis tools at data providers location to minimize data transfer needs. The components of the platforms will be orchestrated as virtual machines, working in a protected network. User authentication and authorization will be managed centrally to provide a single-sign-on facility for all components. Links to European-wide authentication systems like ELIXIR AAI[4] will be provided. A central metadata repository will be provided to power the data catalogue, and the appropriate data channels will be set to communicate with data providers in a transparent manner.

1.2 Background

EuCanSHare computational infrastructure is based on already existing components, that will be adapted to work in an integrated way and will be complemented with specific developments.

[1] European Genome-Phenome Archive (EGA) http://ega-archive.org
[2] EuroBioImaging (EuBI) https://www.eurobioimaging.eu/
[3] Biobanking and Biomolecular Resources Infrastructure (BBMRI-ERIC) http://www.bbmri-eric.eu/
[4] ELIXIR https://elixir-europe.org/services/compute/aai

The infrastructure layout has been designed as an **evolution of MuGVRE**[5], the cloud-based infrastructure built for the project MuG[6]. Its evolved infrastructure (**openVRE**[7]) will act as a base where the different components will be plugged-in as necessary. openVRE will provide data and tools management, data storage, and authentication and authorization services.

2 Computational infrastructure initial design

2.1 Conceptual design

EuCanSHare's central platform is conceived as a unique reference site for the different types of interested users. **Non-identified users** should have access to the necessary metadata for obtaining a broad overview of the available studies, irrespective of their final location. This should provide enough information to initiate the process of application for data access. In turn, the system should allow **data managers** to handle such application and grant access according to the appropriate policies. Once data access is granted, the system should provide a virtual environment that includes **analysis tools** and manages the **controlled access to the data.**

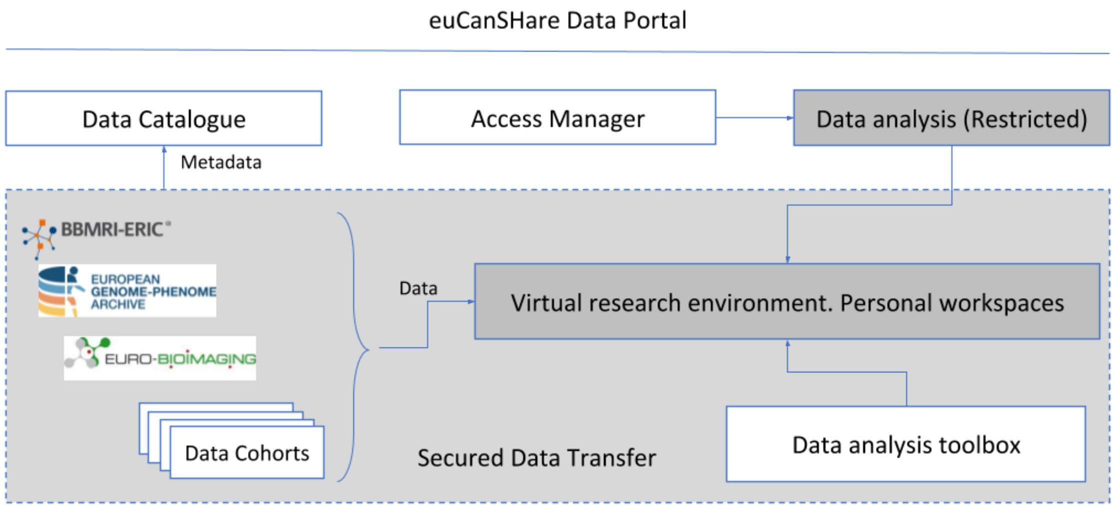


Figure 1. Conceptual design for the euCanSHare computational infrastructure.
Greyed boxes indicate regions of the infrastructure under controlled access and controlled data transmission.

To build the indicated functionalities the following components should be designed and implemented (Figure 1):

1. A **web-based front-end** providing a unified access point. The web-front end should link to platform's components, but also give access to the project information, conditions and terms of use, access policies, general documentation and the relevant contact points.
2. A **Data Catalogue** built on the metadata provided by project's data repositories. These include the participating data cohorts, the EGA and the EuBI archives, as well as, public data repositories that can be considered necessary. Basic access to the Data Catalogue would be public, but controlled access can also enabled to provide more detailed information. One of the aspects that should be included in the Catalogue is the documentation and procedures to apply for data access credentials.

[5] L. Codó et al., "MuGVRE. A virtual research environment for 3D/4D genomics," bioRxiv, p. 602474, Apr. 2019 doi: https://doi.org/10.1101/602474. Available at: http://vre.multiscalegenomics.eu/home/
[6] Multiscale Complex Genomics project (MuG), http://www.multiscalegenomics.eu
[7] openVRE, https://github.com/inab/openVRE

3. An **Access Manager** designed to provide support for managing access credentials. The access manager should provide a working environment for i) researchers applying for data access, ii) data managers responsible for evaluating such applications. For each role, the manager will provide a tailored interface. Researchers should have access to their applications, follow their state, communicate with the Data Access Committee, and eventually obtain the access credentials. Data managers will manage the received applications, and eventually grant or revoke credentials as desired. When available, the Access manager should provide the infrastructure for automatic granting procedures, and interface with the Smart Contracts prototype (T3.6).
4. A **Virtual Research Environment (VRE)** acting as a central workspace for authenticated users. For each user's role, a specific workspace will be provided. The workspace will combine analysis tools, data visualization, management tools, and access to the granted data. The environment should be linked to the appropriate computational infrastructure with the corresponding capabilities to perform the desired analysis. **Programmatic Access** to the infrastructure will allow to generate large scale analysis and eventually interface the infrastructure with external workflow managers.
5. A **Data management infrastructure** providing transparent communication channels between the central catalogue, the VRE, cohort data providers and data repositories.

2.2 Infrastructure technical design

Given the increasing multidisciplinary nature of collaborations, complexity and heterogeneity of scientific data, and high-end computation needs, it is becoming common practice to develop virtualized infrastructures with **interoperable modules** to provide an adaptable, reusable and scalable system. Accordingly, the initial EuCanSHare platform is built upon existing research infrastructures and IT solutions combined to fulfil the following requirements:

Table 1. EuCanSHare platform requirements and adopted solutions.

Design Requirements	Technical Solution
Intuitive and interactive web-based catalogue that simplifies and facilitates the discovery, exploration, sharing and analysis of data for researchers, regardless of their programmatic skills.	EuCanSHare catalogue will be initially based on the **OBIBA software stack**, designed for a complete support to cohort data management. It will be enriched/completed with the metadata obtained from EGA and EuBI and public repositories.
Flexible and modular design to guarantee an easy incorporation of new services and data sources.	All components will be assembled/developed as independent modules, implemented in **virtualized systems**, using the appropriated data communication channels
Integration of advanced authentication and authorization services able to centrally manage fine-grain data access	The central authentication services will be based on KeyClock, providing support for **OpenID connect** identity providers. Obiba'

control on federated EuCanSHare resources.	Agate server will be configured as OpenID client. All data communications and API accesses will be controlled using the OAuth2 protocol. Authorization services will be initially kept at their original sites.
Portable and versatile cloud-based computational infrastructure that permit to conveniently integrate heterogeneous software components while providing scalable compute resources on-demand.	The platform will be built on top of OpenVRE system, relying on **OCCI compliant cloud** cloud managers like OpenNebula or OpenStack.
Software scheduler(s), able to manage analysis workflows, and computational resources in a transparent and adaptable manner. This will be an elastic infrastructure with automatic adaptation to user loads.	Two initial software schedulers, available at openVRE will be used. **Open Grid Engine** will be used for applications requiring stable computational needs, while **PyCOMPS/PMES** will be used when computational needs may depend on the specific analyses.
Data storage solutions able to grow on demand, with a fast data mobilization infrastructure, and providing the necessary data synchronization capabilities among data providers and analysis.	Lower level storage system will be based on NoSQL **MongoDB** database manager. It provides the necessary characteristics of stability, ability of growth, and horizontal scalability. Cohort data will be managed by Obiba's Opal server, also over MongoDB

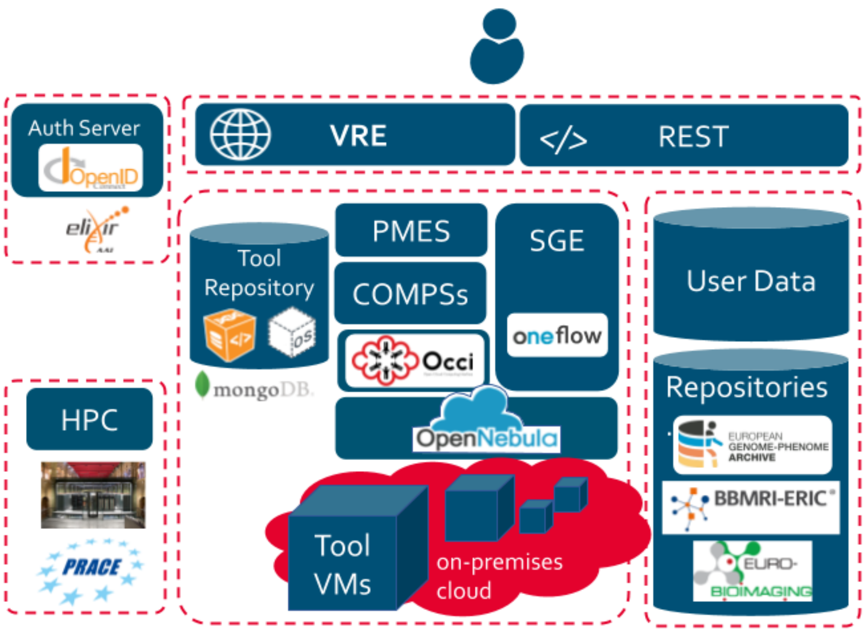


Figure 2. EuCAnSHare central platform infrastructure.

3 Software components

The following section describes individually the software components used in the initial installation and their specific function.

3.1 Cloud infrastructure

The infrastructure is built on top of the ELIXIR-ES cloud infrastructure, an **on-premises private cloud** located at the Barcelona Supercomputing Center (BSC) and hosting other relevant research infrastructures like MuGVRE or EGA. Based on Starlife[8] HPC facilities, the cloud stack is managed by OpenNebula[9] a well-known cloud middleware that enables an efficient administration of the underlying cloud resources by multiple tenants. Following the classical cluster-like architecture, OpenNebula is composed by a front-end with different interfaces (such as REST, XML-RPC and Web-based) controlling multiple remote hosts where a hypervisor (e.g. Xen[10]; KVM[11]; or VMWare[12]) virtualize the actual computational resources (e.g. cores, RAM memory, network, etc.). The euCanSHare prototype is based on **KVM-enabled virtual machines**. Moreover, OpenNebula offers good connectivity with other infrastructure providers, like Amazon EC[13], which could be used to outburst resources if the local infrastructure is not enough, or with a number of open cloud standards, like the **Open Cloud Computing Interface**[14](OCCI), that permits the remote management of virtual machines (VMs) and other cloud resources by compliant cloud orchestration services.

Table 2: ELIXIR-ES computational resources

	Cores	RAM memory	Storage	Network
ELIXIR-ES cloud	14 x 40 cores	14 x 160GB	Shared GPFS of 1TB (NFS access)	10Gb Ethernet
EuCanSHare "tenancy"	36 cores	120GB	500 GB	local cloud VLAN 1 public address

euCanSHare has a set of **dedicated resources in the ELIXIR-ES cloud** (see table 2). Through OpenNebula, the resources are managed to maintain a series of euCanSHare VMs allocating web servers, database systems and other housekeeping servers. Cloud administrators instantiated these VMs at boot time and maintain them permanently on the system. Additionally, the platform can include auto-managed VMs, as computational services implemented as openVRE tools are able to be fully automatically controlled making use of the OCCI server (see below Programming Model Enacting Service).

[8] BSC Starlife, https://www.bsc.es/es/marenostrum/star-life
[9] OpenNebula, http://www.opennebula.org
[10] Xen, http://www.xen.org/
[11] KVM, http://www.linux-kvm.org/page/Main_Page
[12] VMWare, http://www.vmware.com/
[13] Amazon EC2, http://aws.amazon.com/en/ec2/
[14] OCCI, http://occi-wg.org

3.2 EuCanSHare catalogue. OBIBA software stack

The euCanSHare data catalogue will combine cohorts data, genomic data hosted at the EGA[1] and image data provided by EuBI[2]). Support for cohorts' data management within the catalogue is being based in the **Obiba software stack**[15,16]. We have deployed servers for Agate (authentication and authorization server)[17], Opal (data server - Figure 4)[18], and Mica (Cohorts server - Figure 5)[19] and a provisional front end based on the Mica Drupal client (Figure 6)[20]. Table 3 lists them.

Table 3: EuCanSHare data services at BSC.

Service	Description	Site	Access
EuCanSHare platform	Main platform portal	https://eucanshare.bsc.es	public
Virtual Research Environment	Analysis framework	https://vre.eucanshare.bsc.es	Restricted
Mica front-end	Cohort browser	https://studies.eucanshare.bsc.es	Public
Mica server	Cohort studies server	https://mica.eucanshare.bsc.es	Restricted
Opal server	Cohort data repository	https://opal.eucanshare.bsc.es	Restricted

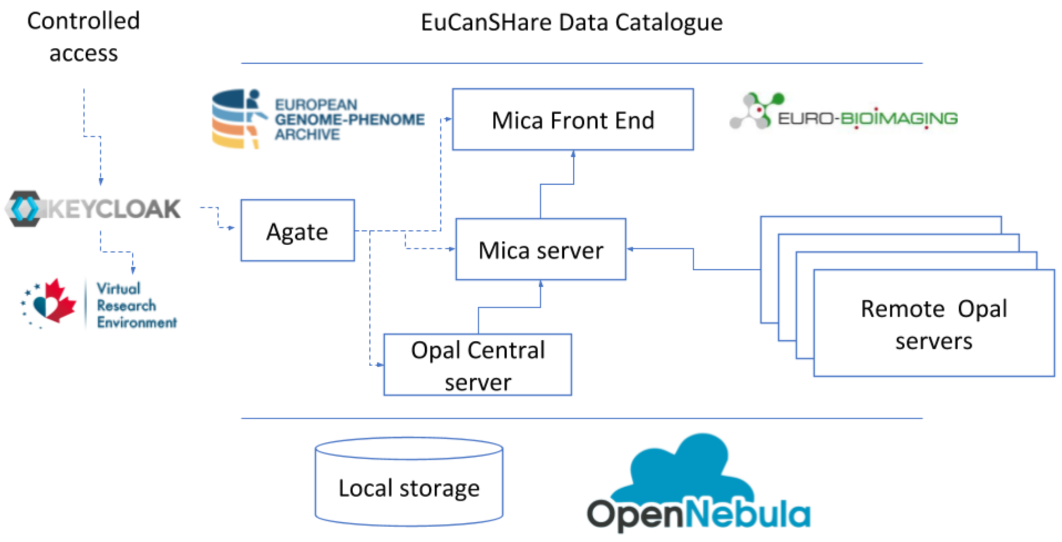
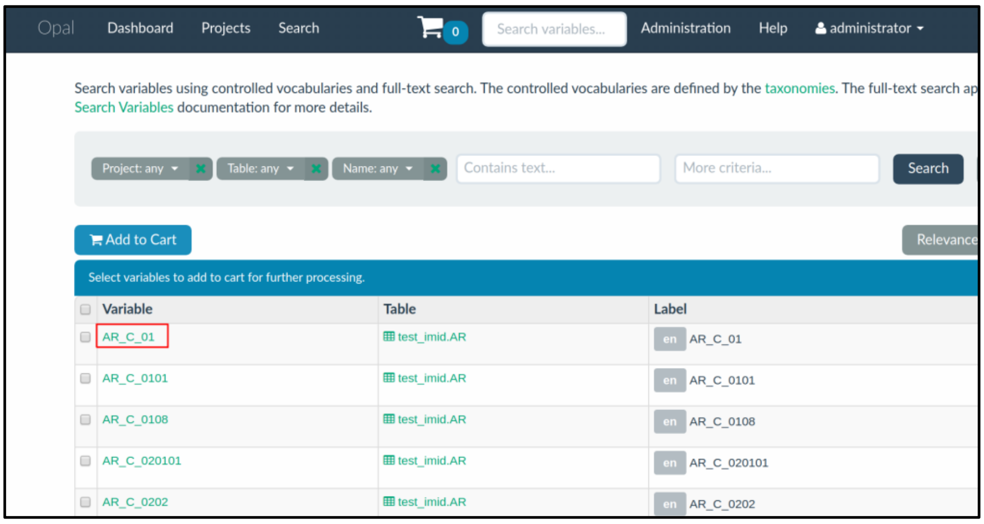


Figure 3. Integration of OBIBA software stack in EuCanSHare computational infrastructure.

[15] http://obiba.org
[16] Doiron, D., Marcon, Y., Fortier, I., Burton, P. and Ferretti, V., 2017. Software Application Profile: Opal and Mica: open-source software solutions for epidemiological data management, harmonisation and dissemination. International journal of epidemiology.
[17] https://agate.eucanshare.bsc.es
[18] https://opal.eucanshare.bsc.es
[19] https://mica.eucanshare.bsc.es
[20] http://studies.eucanshare.bsc.es

Figure 3 describes how OBIBA software stack is integrated into the euCanSHare computational infrastructure. The Agate server is configured as an OpenId[21] Client of the euCanSHare central authentication server (based on the KeyCloak) providing a single-sign-on system for the project and being able to accept identity providers like the forthcoming European Life Sciences ID systems based on ELIXIR AAI[4]. The Opal server at BSC will host the euCanSHare public datasets, while controlled access datasets will be declared at the central Mica server but will be kept in distributed Opal servers (data already available from MORGAM, and SHIP cohorts).

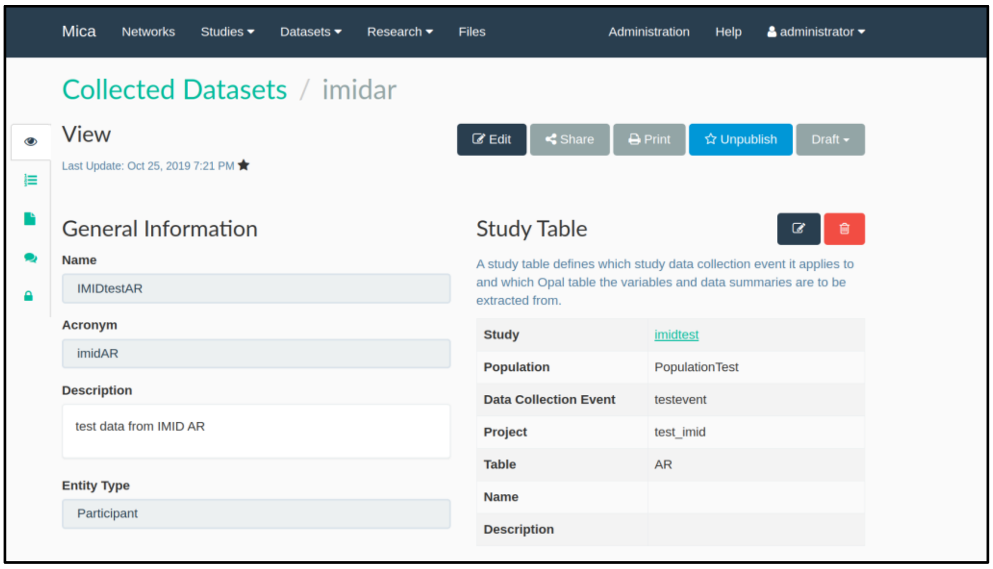


Search variables using controlled vocabularies and full-text search. The controlled vocabularies are defined by the taxonomies. The full-text search ap
Search Variables documentation for more details.

Select variables to add to cart for further processing.

Variable	Table	Label
AR_C_01	test_imid.AR	en AR_C_01
AR_C_0101	test_imid.AR	en AR_C_0101
AR_C_0108	test_imid.AR	en AR_C_0108
AR_C_020101	test_imid.AR	en AR_C_020101
AR_C_0202	test_imid.AR	en AR_C_0202

Figure 4. Variables related to a test dataset stored in the Opal server.



Collected Datasets / imidar

View

Last Update: Oct 25, 2019 7:21 PM

General Information

Name

IMIDtestAR

Acronym

imidAR

Description

test data from IMID AR

Entity Type

Participant

Study Table

A study table defines which study data collection event it applies to and which Opal table the variables and data summaries are to be extracted from.

Study	imidtest
Population	PopulationTest
Data Collection Event	testevent
Project	test_imid
Table	AR
Name	
Description	

Figure 5. Mica server displaying datasets information coming from the Opal server.

[21] https://openid.net

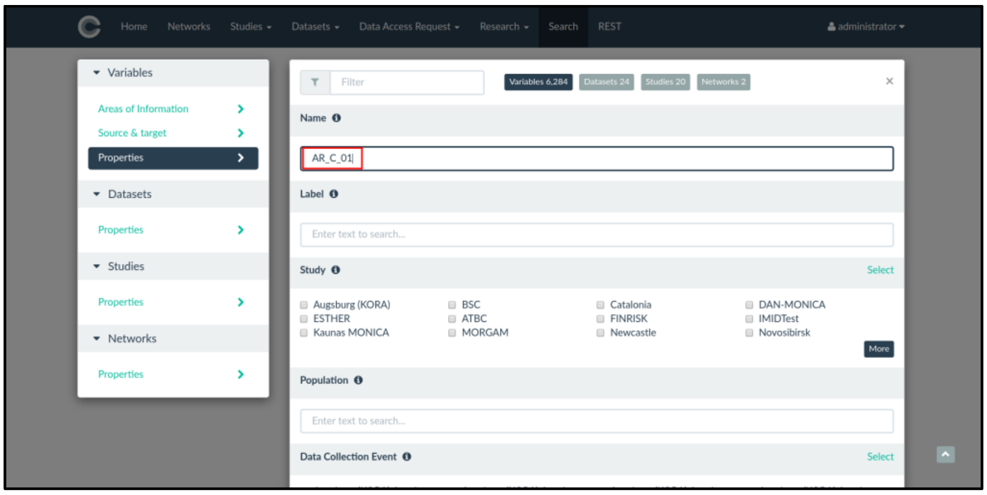


Figure 6. Mica client searching feature that allows the filtering of Mica server data.

3.3 openVRE: the computational infrastructure

openVRE[7] is a **Platform as a Service** (PaaS) composer that collects job deployment requests from a web exposed Virtual Research Environment (VRE) and coordinates the resource and **service deployment** over dynamic PMES virtual appliances or directly on a queueing batch system like OGS. openVRE aims to be a white canvas with a set of operational services and protocols to handle the computational and data resources on an underlying OCCI-compliant cloud provider. As a result, a tailored computational infrastructure is rapidly assembled, enabling to build, run, and operate applications in cloud-based infrastructures. After configuring openVRE for a particular project, the researcher accesses to a tailored VRE with a set of ready-to-use services (datasets, analysis tools and visualizations) fully adapted to their needs. Examples of openVRE implementations are MuGVRE[5], and OpenEBench[22].

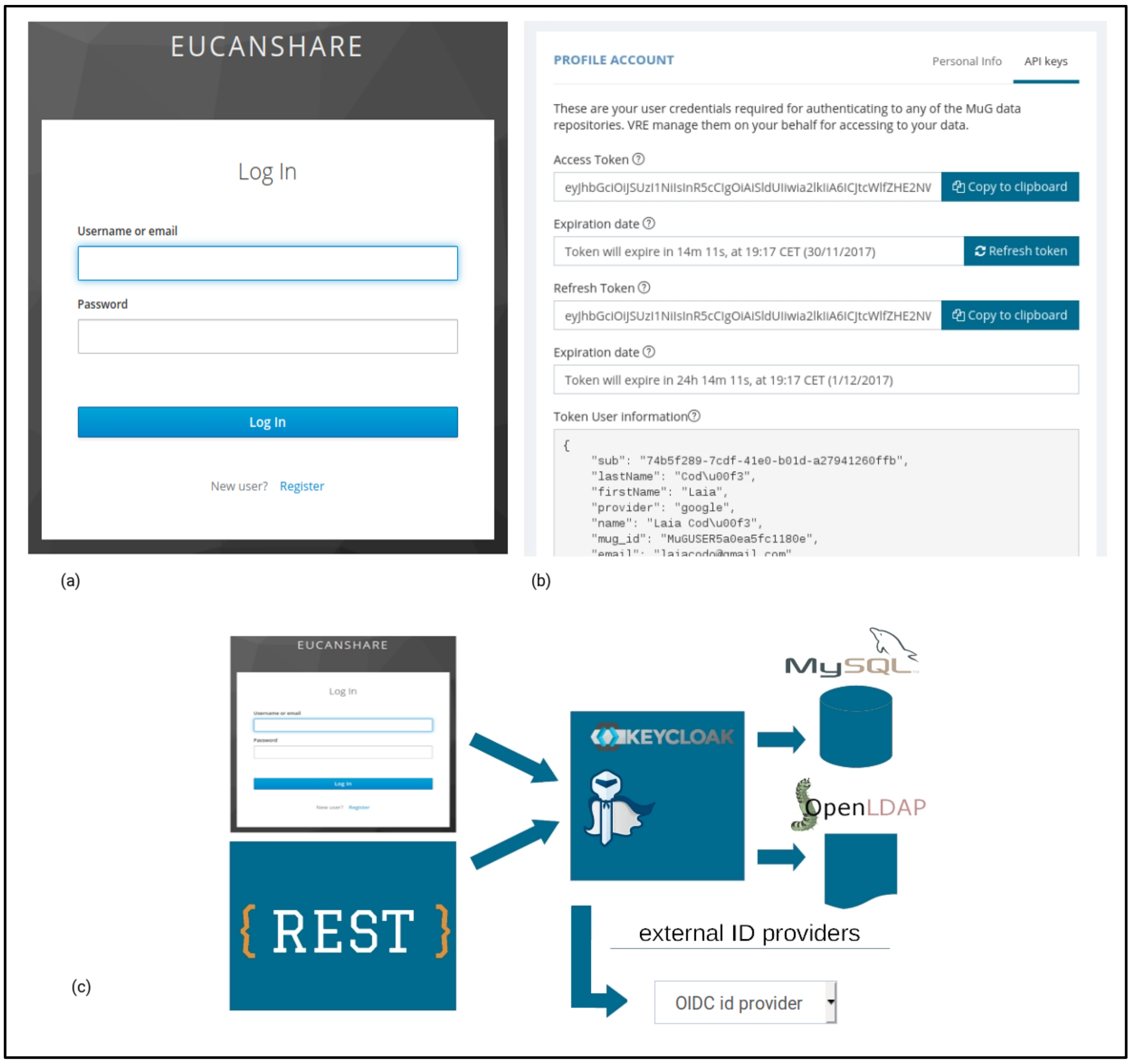
3.3.1 Authentication

The euCanSHare infrastructure should assure a complete data privacy with respect to users' data and activities. To this end, access to the catalogue, workspace and tools, either interactively or through REST APIs is made using an encrypted channel (https, ssh), and users are authenticated on every access.

The euCanSHare site uses **Keycloak**[23] as Identity and Access Management solution. Keycloak implements OpenID Connect 1.0 (OIDC), which supports the OAuth2 **code authentication flow** for Web access (based on username/password), and a **token-based authentication** for the remaining services. EucanSHare openVRE displays the access tokens in use and allows to refresh them (see Figure 7.b) so that the user is able to authorize himself to the publicly available services via REST.

To ease user registration, additional external OIDC identity providers (idPs) might be accepted, like Google, ORCID or ELIXIR AAI[4]. Once the authentication through these providers has taken place, an openVRE internal user record is created, with all security considerations in place independently of the idP used.

[22] https://openebench.bsc.es/vre
[23] https://www.keycloak.org/



PROFILE ACCOUNT

These are your user credentials required for authenticating to any of the MuG data repositories. VRE manage them on your behalf for accessing to your data.

Access Token: eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJtcWlfZHE2NV

Expiration date: Token will expire in 14m 11s, at 19:17 CET (30/11/2017)

Refresh Token: eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJtcWlfZHE2NV

Expiration date: Token will expire in 24h 14m 11s, at 19:17 CET (1/12/2017)

Token User information

```
{
    "sub": "74b5f289-7cdf-41e0-b01d-a27941260ffb",
    "lastName": "Cod\u00f3",
    "firstName": "Laia",
    "provider": "google",
    "name": "Laia Cod\u00f3",
    "mug_id": "MuGUSER5a0ea5fc1180e",
    "email": "laiacodo@gmail.com"
```

Figure 7. (a) EuCanSHare Login page. (b) User profile details including authorization tokens. (c) Schema of the centralized authorization service based on Keycloak.

3.3.2 Personal workspace

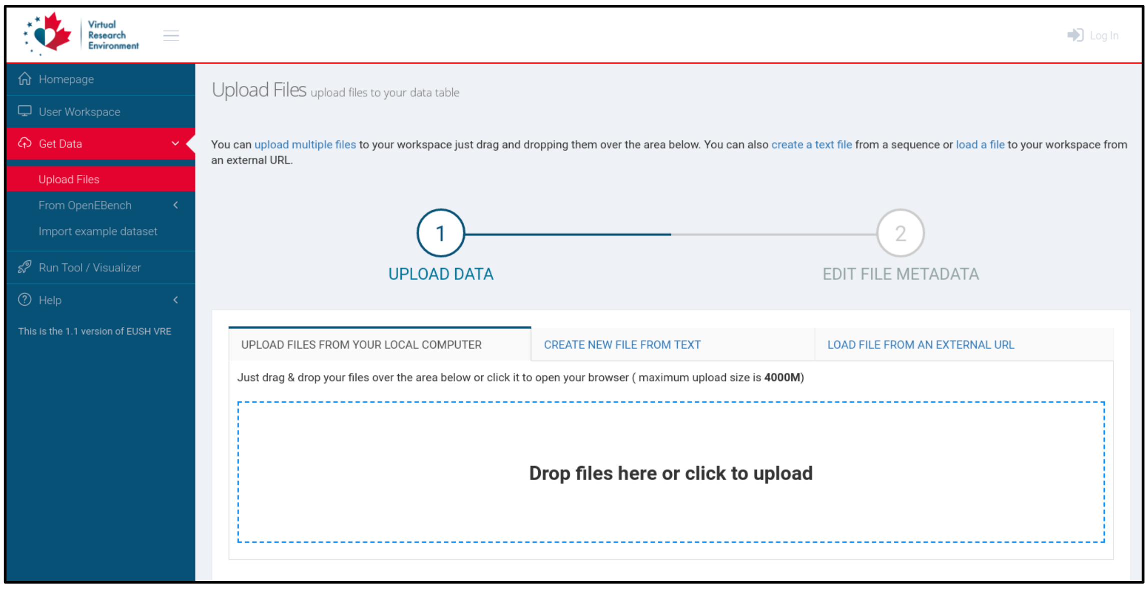
The euCanSHare personal workspace, provided by openVRE, is the central environment for the activity of authenticated users. The contents and layout of the workspace is adapted to the user's assigned roles. It is a filesystem-based layout where uploaded data and analysis results are available. The workspace gives also **access to analysis and tools**, selected according to data types and file types (formats), recovering results as soon as they are available.

Getting data

Users can populate the workspace in several ways (Figure 8):

- **Direct upload**: Files from user's local computer can be uploaded directly in the workspace through an HTTPS protocol. The amount of data that can be uploaded in this way is limited due to the technical limitations of the protocol.
- **Create files**: A text editor is available to create simple plain text files. This is intended for data or metadata of reduced format that can be simply typed in.
- **Upload from External URL**: given a URL (FTP/HTTP), the platform downloads the data into the workspace. This is the recommended procedure to include bulky data, as the procedure is performed in the background and no limit in size applies, being only limited by the user's quota available in their workspace. This option is also recommended for obtaining data from public repositories.

- **Import from Repository**: internally operating as "Upload from External URL", the platform might integrate any REST-based data repository allowing the user to browse the content and import it into the workspace with one click. This feature will be extended to **include the euCanSHare Catalogue**.


Virtual Research Environment
Log In
Homepage
User Workspace
Get Data
Upload Files
From OpenEBench
Import example dataset
Run Tool / Visualizer
Help
This is the 1.1 version of EUSH VRE
Upload Files upload files to your data table
1
UPLOAD DATA
2
EDIT FILE METADATA
UPLOAD FILES FROM YOUR LOCAL COMPUTER
CREATE NEW FILE FROM TEXT
LOAD FILE FROM AN EXTERNAL URL
Drop files here or click to upload

You can upload multiple files to your workspace just drag and dropping them over the area below. You can also create a text file from a sequence or load a file to your workspace from an external URL.

Just drag & drop your files over the area below or click it to open your browser (maximum upload size is **4000M**)

Figure 8. Options to upload data into the workspace.

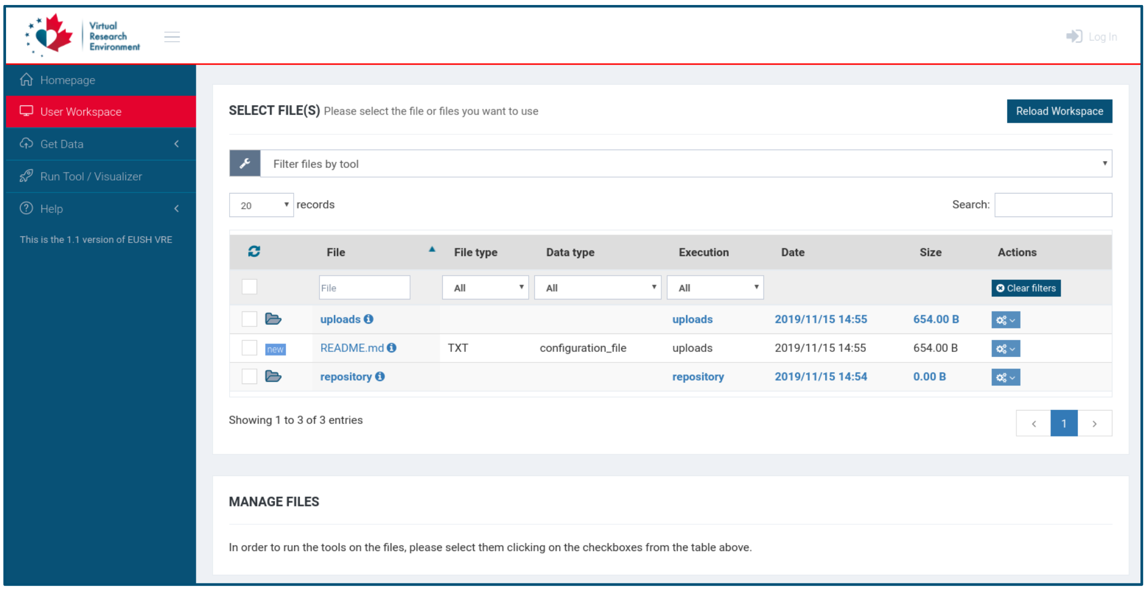
Data files should be "validated" after upload. Validation includes a number of internal checks on formats, but also requires the user to fill in a series of metadata items. These include descriptor fields like data types (e.g. 'DNA sequence') and formats (e.g. 'FASTA') selected from a predefined list. Data types and formats enable the system to select the appropriate set of tools and visualizers usable with the uploaded files. Metadata for files obtained from installed tools are automatically obtained from the tools metadata manifest.

The workspace

VRE workspace (Figure 9) is organized with a **file system layout** with an intuitive look-and-feel. There are two types of data objects: files and folders grouping files. The *Uploads* folder include all data uploaded by the user in either manner (direct, edit, or URL). Data from repositories that is grouped under *Repository* folder. The remaining folders correspond to "run" folders containing the result files of executed tools. A new folder is generated for any new process started in the VRE. Files can be filtered by any name, format, data type, or project. Also, a tools-based filter allows to select only valid data input for a given tool.

Three interactive toolkits are associated with workspace's data and allow to **select the desired analysis tool**, visualize result files, etc. Toolkits' content is adapted to each file/folder thanks to the available metadata. The toolkits contain the following options:

- File toolkit: Download data or folder, edit metadata, delete, pack and compress.
- Visualization toolkit: List of visualizers accepting the selected file/s (based on file the format).
- Tools toolkit: List of tools accepting the selected file/s as inputs (based on data type and format metadata).


Virtual Research Environment

Log In

Homepage
User Workspace
Get Data
Run Tool / Visualizer
Help
This is the 1.1 version of EUSH VRE

SELECT FILE(S) Please select the file or files you want to use

Reload Workspace

Filter files by tool

20 records

Search:

		File	File type	Data type	Execution	Date	Size	Actions
		File	All	All	All			Clear filters
		uploads			uploads	2019/11/15 14:55	654.00 B	
	new	README.md	TXT	configuration_file	uploads	2019/11/15 14:55	654.00 B	
		repository			repository	2019/11/15 14:54	0.00 B	

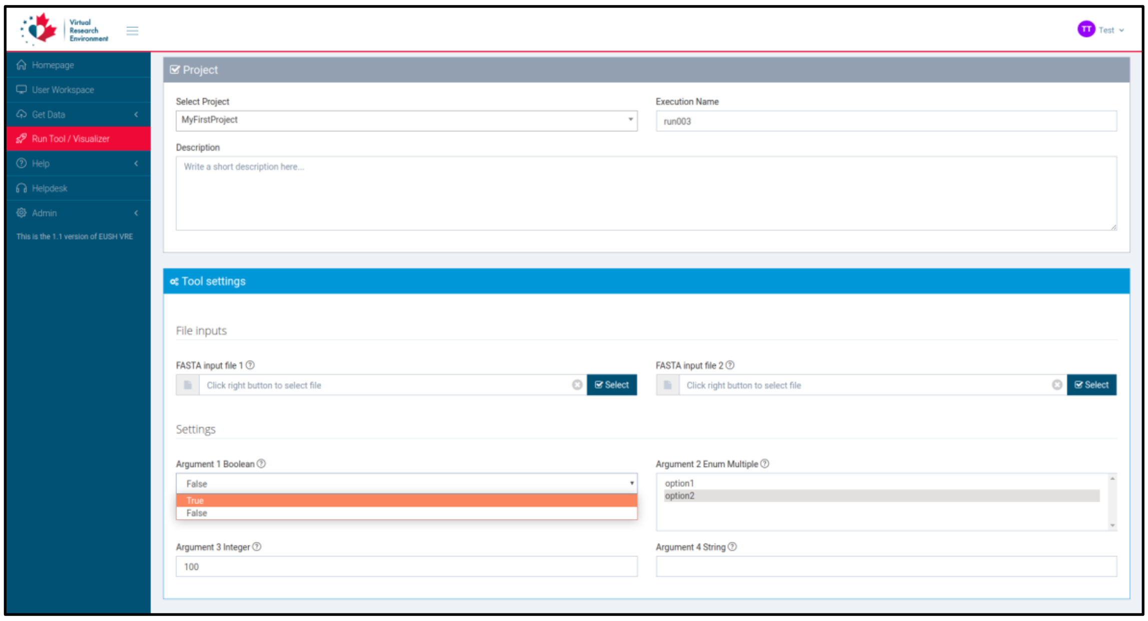
Showing 1 to 3 of 3 entries

MANAGE FILES

In order to run the tools on the files, please select them clicking on the checkboxes from the table above.

Figure 9. euCanSHare personal workspace built on openVRE.

The selection of a specific tool triggers the **analysis configuration screen** (Figure 10) where user can assign the selected data files to the appropriate input parameters and arguments. After configuration, the **job is sent** to the VRE backend who orchestrates the cloud services in a transparent way. Progress of the execution can be followed in the main workspace.


Virtual Research Environment

Test

Homepage
User Workspace
Get Data
Run Tool / Visualizer
Help
Helpdesk
Admin
This is the 1.1 version of EUSH VRE

Project

Select Project: MyFirstProject

Execution Name: run003

Description: Write a short description here...

Tool settings

File inputs

FASTA input file 1: Click right button to select file — Select

FASTA input file 2: Click right button to select file — Select

Settings

Argument 1 Boolean: False (True, False)

Argument 2 Enum Multiple: option1, option2

Argument 3 Integer: 100

Argument 4 String:

Figure 10. Configuration screen for a sample tool.

openVRE provides constant tracking of the state of the operations performed and the available space in the VRE.

3.3.3 Job process management

The openVRE backend is able to process VRE's job petitions using (i) an auto-scalable queueing system based on OGS/Oneflow, or (ii) an elastic cluster deployment service based on pyCOMPs/PMES. Open Grid Scheduler (OGS) successfully manages web application backends where the major requirement is to be able to deal with variable levels of user demand.

Workflows that show a more complex structure where, for instance, computational resources should be adjusted at run time, are recommended to be configured using COMPSs/PMES.

OGS/Oneflow: Auto-scalable queuing system

Open Grid Scheduler (OGS)[24] (former Sun Grid Engine SGE) is designed to manage distributed software executions in heterogeneous computational environments. OGS is used normally in cluster-based infrastructures as a general process scheduler. Capabilities of OGS include, among others, resource management, remote execution, parallel execution management, interactive processes, monitoring and accounting, and integration with Amazon EC2 or Hadoop. To better exploit OGS features into a cloud-based environment like openVRE (Figure 11), an OpenNebula self-provisioning tool called oneFlow[25]is added to the equation. oneFlow is able to automatically trigger the deployment/undeployment process of VMs in front of monitorized parameters, like the CPU workload of these VMs, the I/O stress or a certain network, etc.

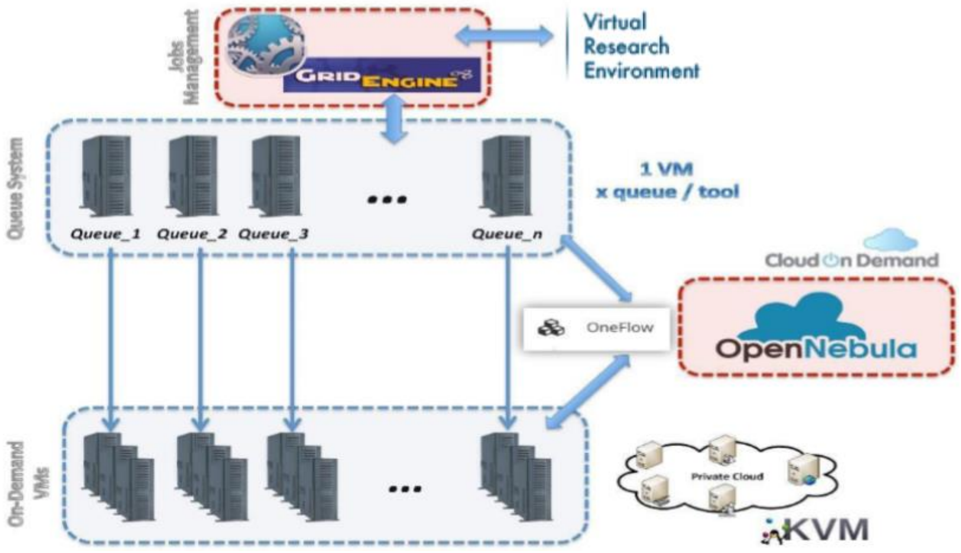


Figure 11. Schema of the implementation on OneFlow

Analysis **tools are implemented as VMs** under the control of OGS. Each VM packs an application with the corresponding OGS queue configuration. In the case of increased demand on a certain application (i.e. certain VM), oneFlow instructs OpenNebula to replicate such VM, which is translated into an increased number of hosts available in the queue system for such an application. Once the workload for such group of VMs decreases, OneFlow undeploys them one by one, always keeping at least one instance ready to accept new job petitions. In this way, allocated resources are dynamically and transparently adjusted on-demand.

[24] OGS, Open Grid Scheduler (former SGE, Sun Grid Engine) https://sourceforge.net/projects/gridscheduler

[25] OpenNebula, http://docs.opennebula.org/5.8/integration/system_interfaces/appflow_api.html

COMPSs programming model

COMP Superscalar (COMPSs)[26] is a programming model designed to ease the development of **applications for distributed infrastructures** (e.g. Clusters, Grids, and Clouds). To this end, COMPSs features a runtime system able to discover applications' parallelism of at execution time and dynamically distribute the tasks. openVRE implements COMPSs python binding (pyCOMPSs) as advanced software scheduler.

COMPSs hides parallelization complexities to developers so that programmers do not need to deal with the duties of parallelization, such as thread creation and synchronization, data distribution, messaging or fault tolerance. Instead, COMPSs is based on user's sequential programming, which makes it appealing to users that either lack parallel programming expertise or are looking for better programmability. User's Java applications are directly supported by COMPSs runtime, while other languages like Python, are supported through the appropriate bindings (pyCOMPSs). Although programming is sequential, execution is parallel, since at runtime COMPSs builds a workflow composed by the tasks of the application, which is connected through edges that denote data dependencies between them, and determined by annotations specifying the needs of each task. From this workflow, the COMPSs runtime is able to execute different application tasks at a time within a master-workers architecture (Figure 12.a). In this scenario, the user submits its application to the master node, which orchestrates the parallelization and launches the tasks in the available resources, distributing the input data and collecting the results.

PMES (discussed next section) and COMPSs form a good tandem for enabling virtual elastic compute clusters (Figure 12.b). At execution time, when COMPSs tasks demand extra resources, the master schedule extra jobs in a queue system, or provisions extra VMs on the cloud, elastically demanding the resources according to the specific needs of the execution. PMES server provisions these VMs, which become COMPSs workers and remotely start user's execution.

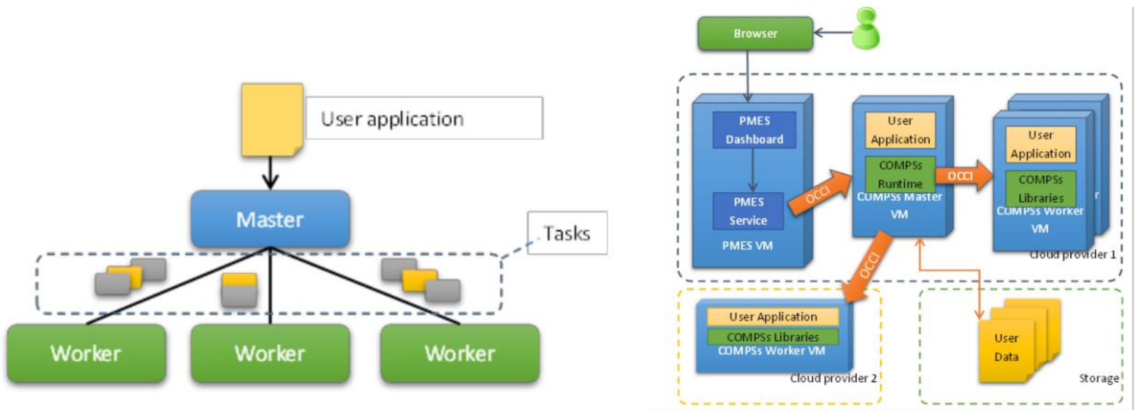


Figure 12: (a) Architecture of task distribution in COMPs (b) Virtual clusters' deployment using COMPS and PMES.

PMES: Programming Model Enacting Service

The Programming Model Enacting Service (PMES)[27] allows users to submit job **executions to remote clouds**. PMES receives user's job (e.g. application's name, list of inputs) and it is able to

[26] COMP Superscalar, an interoperable programming framework, SoftwareX, Volumes 3–4, December 2015, Pages 32–36, Badia, R. M., J. Conejero, C. Diaz, J. Ejarque, D. Lezzi, F. Lordan, C. Ramon-Cortes, and R. Sirvent, DOI: 10.1016/j.softx.2015.10.004

[27] F. Lordan et al., "ServiceSs: An Interoperable Programming Framework for the Cloud," J. Grid Comput., vol. 12, no. 1, pp. 67–91, Mar. 2014

(i) remotely instantiate and contextualize the image packing the application, (ii) stage-in the input data there, (iii) run the application in the virtual environment, and (iv) stage-out output data. Application's execution might either be a standalone executable, or a COMPSs enabled workflow, in which case PMES starts the COMPSs' runtime within the virtual machine, that becomes the master of the COMPSs' virtual cluster. Since PMES is also deployed within a virtual machine, as well as the COMPSs runtime deployed for the application execution, both of them interact with the Cloud provider (e.g. OpenNebula) through the use of an OCCI connector.

The use of COMPSs as workflow manager is considered within PMES in order to improve the performance on workflow executions. These applications are executed by launching the COMPSs runtime in a virtual machine; whereas to execute standalone applications, the PMES is also able to run the required command.

3.3.4 Data repository

openVRE user's data is divided in two types of repositories. **Metadata is held in a MongoDB**[28] database hosted at the Starlife data storage system following a data model mainly based on a collection of data types, files types among other file related attributes like the path where the file can be found in the file system.

The MongoDB server not only holds the metadata referring to user's files, but also the necessary data to correctly define tools and visualizers and how they interact with user's files. MongoDB also keeps track of user management, job execution, and other VRE functionalities like help, sample data collections, etc.

Data itself is stored in a standard filesystem in its original format. The filesystem is shared with the virtualized environments via the network file system protocol. The filesystem layout is organized per user so that the privacy of data is maintained. In fact, process managers specifically mount to the deployed VM only the data belonging to the user executing the application.

Catalogue data and metadata are managed by the OBIBA software stack (Mica and Opal servers), and stored in the MongoDB database at Starlife, for data locally stored at BSC and opal servers at THL (MORGAM cohorts), and UMG (SHIP cohorts).

3.4 Support for the integration of new tools

3.4.1 Protocol for the preparation and inclusion of new tools

Tools to be executed at openVRE should be deployed in a Virtual Machine. Tools itself should not be modified or adapted to the infrastructure. Instead, a **tool's integration protocol** (*Figure 13*) has been implemented in order to better support the process of wrapping, submission and annotation of user's pipelines or applications into openVRE. The whole process can be divided in two main steps, a wrapping process for preparing the application to be invoked from openVRE, and the actual integration of such code into the infrastructure.

[28] http://mongodb.com

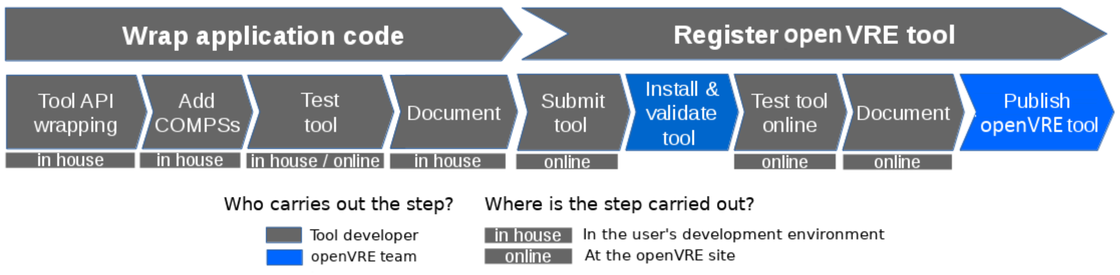


Figure 13: openVRE protocol for integrating new tools.

Application wrapping is based on the Tool API, a Python library developed in the MuG project that provides a common access interface for openVRE tools. It is formalized as a tool skeleton on top of which user adapts the application. The user should clone the Tool API repository in their own development environment and fill in the skeleton according to its pipeline's requirements. If parallelisation is to be enabled, pyCOMPSs decorators should also be added here. Once the tool has been set up, we suggest a functional testing by which an openVRE execution is emulated at user's own installation by creating a set of test job configuration files (job configuration and input's metadata JSON files - see annex euCanSHare VRE documentation). The process can be performed online through the developer's workspace. When the tool locally passes the functional tests, the code is ready to be documented and made available on an accessible software repository, as openVRE coding guidelines suggest.

From the developer's perspective, integrating a new tool into the openVRE infrastructure essentially implies its **definition and documentation**. Through a developer's workspace, the user provides the code location, tool's descriptive metadata (i.e. title, keywords, etc), deployment details (i.e. CPUs, memory, tool main script path, etc), logo images, etc. to eventually "Submit" the tool, at this point the ticketing system opens a communication channel with the user, and euCanSHare's support team is made aware of the Tool proposal. The submission is evaluated and validated. Then, a virtual machine instance with the tool's code is deployed in the cloud. After provisioning, the new tool is activated under the testing mode, and tool-related web pages are generated automatically based on the developer's tool definition. Finally, the tool is debugged, refined and tested on openVRE, example data sets are made available from the 'Get Sample Data' menu, and the tool help pages are prepared through the openVRE online markdown editor.

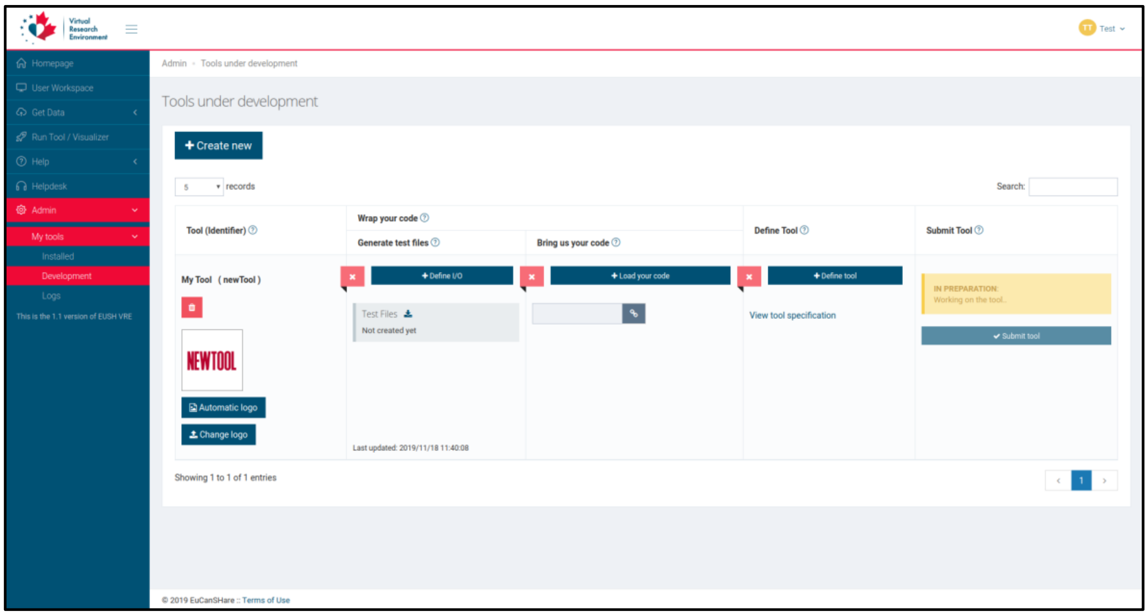
The whole integration protocol is fully described and documented step by step, including training material (see annex euCanSHare VRE Documentation).

3.4.2 Developer's specific workspace

Apart from the regular account, a user can be granted with the "tool developer" role, by which developers are entitled to create and manage their own tool instances. Regular users can upgrade their account by editing their profile settings, which produces a petition to be processed by the openVRE support team. For debugging purposes, tool developers have extended access to the metadata of workspace files and jobs, being able to access job configuration JSON files, expected job output files, file metadata documents, etc. Moreover, they are granted access to the developer's workspace.

The developer's workspace is split in two sections, one meant to create and help developing new openVRE tools, and a second one dedicated to managing already integrated tools. In the first (Figure 14), tool developers freely initiate a new tool entry, represented as a new line in the central table. Each column represents the integration protocol steps described above for which openVRE either offers support or gathers information from the tool developer. These steps are: (1) the generation of test files for the in-house testing, a downloadable TAR file with the set of

JSON configuration files and a bash script with the very command openVRE will use to invoke the tool. To this end, the tool developer needs to provide a definition of the input files, arguments and expected output files of its tool. (2) The URL where the tested tool code is to be found. (3) The JSON validation of the tool definition fields required for the registration (deployment details, tool descriptions, etc). (4) The storage or automatic generation of tool logo images. Finally, the last column represents the submission status that can take values like “in preparation”, “submitted”, “to be reviewed”, “rejected” or “accepted”.



Admin • Tools under development

Tools under development

5 records Search:

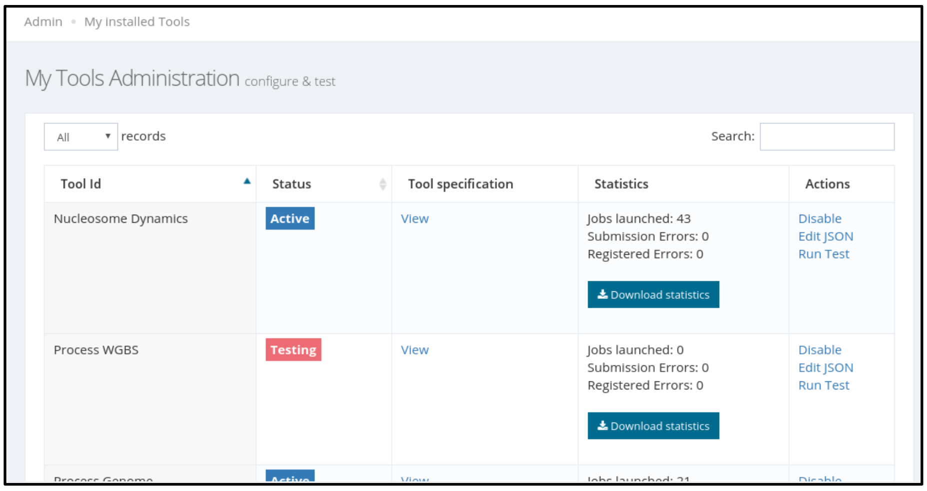
Tool (Identifier)	Wrap your code: Generate test files	Wrap your code: Bring us your code	Define Tool	Submit Tool
My Tool (newTool)	Test Files Not created yet Last updated: 2019/11/18 11:40:08		View tool specification	IN PREPARATION: Working on the tool..

Showing 1 to 1 of 1 entries

Figure 14: Tool developer’s workspace for integrating new tools.

A second developer’s Workspace is used once the tool is eventually accepted. It consists of a panel listing all the tools belonging to that user (Figure 15). The workspace displays the definitive tool definition as stored in the openVRE database, allows to download the tool usage statistics, and administer the tool status on the infrastructure:

- **Active**: the tool is eligible to be run for all users.
- **Inactive**: the tool is not eligible to be run
- **Testing**: the tool is eligible only by the tool developer owing the tool.



Admin • My installed Tools

My Tools Administration configure & test

All records Search:

Tool Id	Status	Tool specification	Statistics	Actions
Nucleosome Dynamics	Active	View	Jobs launched: 43 Submission Errors: 0 Registered Errors: 0	Disable Edit JSON Run Test
Process WGBS	Testing	View	Jobs launched: 0 Submission Errors: 0 Registered Errors: 0	Disable Edit JSON Run Test
Process Genome	Active	View	Jobs launched: 21	Disable

Figure 15: Tool developer’s workspace for managing integrated tools.
Extracted from MuGVRE for illustration purposes.

4 Present status and Development roadmap

At the present stage, the openVRE framework has been installed at StarLife cloud (https://www.bsc.es/marenostrum/star-life), hosting a series of Virtual Machines providing the basic functionality for the euCanSHare platform. Table 4 reports a summary of the VMs and active functionalities.

Table 4. Present status of euCanSHare data portal and components.

Virtual Machine	Function provided	Installed Software	Status
euCanSHare infrastructure			
Login	Internal login node. Main data portal web site	Apache 2 PHP 7.2 MariaDB 10.1 Wordpress 5.33	Data portal front-end prototype available at https://eucanshare.bsc.es
OBIBA	Agate, Mica, Opal, servers, catalogue front end	Agate 1.5.2, Mica Server 2-3.7 Opal 2.15.1, Drupal 7.67	Servers available at https://agate.eucanshare.bsc.es https://mica.eucanshare.bsc.es https://opal.eucanshare.bsc.es https://studies.eucanshare.bsc.es
euCanSHare-VRE	OpenVRE adapted to euCanSHare	openVRE stack PHP 7.2 MongoDriver 1.5	Available at https://vre.eucanshare.bsc.es
Starfile infrastructure			
KeyCloak	Authentication server	KeyCloak v4.8	Available at https://inb.bsc.es/auth
MongoDB	MongoDB manager	MongoDB 4.2	Available for internal use

4.1 Development roadmap

Once the basic framework has been established, we will continue on building the necessary components to complete a fully functional euCanSHare data platform. Specific technical milestones already scheduled are:

1. **Implementation of openVRE REST API (Q1 2020).** Will provide a fully programmatic access to the platform via REST, including data upload/download into the workspace, and analysis schedule. It will follow GA4GH specification for Task execution (TES), and Data management (DRS). The availability of such interface will allow to integrate openVRE hosted operations in external workflow managers and applications.
2. **Implementation of User roles: Data manager, Data Access managers (Q3 2020).** Full integration of authorization roles across the platform. Will allow to deploy specific workspaces according to user privileges.

3. **Implementation of Data Portal Front-end (Q3 2020).** A complete front-end with access to all portal components, including authentication and authorization.
4. **Linking personal workspace with OBIBA's internal workspaces (Q4 2020).** Full integration at the data level among openVRE and OBIBA workspaces, allowing to handle opal, and mica data from EuCanSHare VRE.
5. **Link to EGA and EuBI metadata and data (Q4 2020).** Availability of metadata as part of the euCanSHare catalogue and full controlled access to data from euCanSHare VRE.
6. **Implementation of analysis tools (extended till the end of project).** Complete the offer of analysis tools available at euCanSHare VRE.

5 Annexes

5.1 euCanSHare VRE Documentation

http://eucanshare.bsc.es/dataportal/?page_id=697

5.2 Documents, Software and data models

JSON schema and example of tool definition configuration file , compulsory for registering a now tool in VRE

1. tool definition JSON - schema https://github.com/Multiscale-Genomics/VRE_tool_jsons/blob/dev/tool_specification/tool_schema.json
2. tool definition JSON - example https://github.com/Multiscale-Genomics/VRE_tool_jsons/blob/dev/tool_specification/examples/pydockdna.json

JSON examples for the configuration files sent between VRE and tool VMs during the tool life cycle execution

3. input metadata JSON - example https://github.com/Multiscale-Genomics/VRE_tool_jsons/blob/dev/tool_execution/sample_project/myPydockProject/.input_metadata.json
4. configuration tool JSON - example https://github.com/Multiscale-Genomics/VRE_tool_jsons/blob/dev/tool_execution/sample_project/myPydockProject/.config.json
5. submit file - examples https://github.com/Multiscale-Genomics/VRE_tool_jsons/blob/dev/tool_execution/sample_project/myPydockProject/.submit
6. Output metadata JSON - example https://github.com/Multiscale-Genomics/VRE_tool_jsons/blob/dev/tool_execution/sample_project/myPydockProject_out/.results.json