Social Mining & Big Data Analytics

SoBigData

RESEARCH INFRASTRUCTURE

Deliverable D9.4

SoBigData e- Infrastructure Common Facilities 1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871042

DOCUMENT INFORMATION

PROJECT	
PROJECT ACRONYM	SoBigData-PlusPlus
PROJECT TITLE	SoBigData++: European Integrated Infrastructure for Social Mining and Big Data Analytics
STARTING DATE	01/01/2020 (48 months)
ENDING DATE	31/12/2023
PROJECT WEBSITE	http://www.sobigdata.eu
TOPIC	INFRAIA-01-2018-2019 Integrating Activities for Advanced Communities
GRANT AGREEMENT N.	871042

DELIVERABLE INFORMATION	
WORK PACKAGE	WP9 - JRA2 - E-Infrastructure and Supercomputing Network
WORK PACKAGE LEADER	CNR
WORK PACKAGE PARTICIPANTS	BSC, EGI, Nubisware, OpenAIRE, USFD, UNIPI, FRH, UT, LUH, AALTO, ETH Zürich, TUDelft
DELIVERABLE NUMBER	D9.4
DELIVERABLE TITLE	SoBigData e- Infrastructure Common Facilities
AUTHOR(S)	Massimiliano Assante (CNR), Alessia Bardi (CNR), Enol Fernandez (EGI), Andrea Manzi (EGI), Pasquale Pagano (CNR)
CONTRIBUTOR(S)	
EDITOR(S)	Massimiliano Assante (CNR), Valerio Grossi (CNR), Beatrice Rapisarda (CNR)
REVIEWER(S)	Valerio Grossi (CNR)
CONTRACTUAL DELIVERY DATE	31/12/2020
ACTUAL DELIVERY DATE	14/01/2021
VERSION	1.0
TYPE	Report
DISSEMINATION LEVEL	Public
TOTAL N. PAGES	28
KEYWORDS	Data Analytics, Online Coding, Science Monitoring, Continuous Integration

EXECUTIVE SUMMARY

This deliverable reports the design principles and software architectures characterising the release and development of the SoBigData e-Infrastructure common facilities, namely the social mining computational engine, the online coding and workflow design frameworks, and the online science monitoring dashboard. This report is the first of two versions of the document, each of which describing the design associated with a specific version of the infrastructure to be made available at M12 (December 2020) and at the end of the third year (SoBigData e-Infrastructure common facilities 2, December 2022). Specifically, the deliverable focuses on the design principles and reference architectures included in the first release of the SoBigData e-Infrastructure common facilities at M12.

The deliverable consists of six sections. Section 1 briefly introduces the role of this deliverable for the development and delivery of the SoBigData e-Infrastructure common facilities. Section 2 describes the SoBigData e-infrastructure logical architecture contextualising the common facilities and how they relate with the rest. Section 3, section 4 and section 5 document the first release of the e-Infrastructure common facilities included in this report and available at M12, reporting the design principles and reference architectures of the released solutions. Specifically, section 3 describes the social mining computational engine, Section 4 presents the online coding and workflow design frameworks - which includes the RStudio and the Jupyter Notebooks via JupyterHub - and Section 5 reports the online science monitoring dashboard.

Finally, section 6 concludes the report illustrating the whole Release Management process and its components for continuous integration.

DISCLAIMER

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871042.

SoBigData++ strives to deliver a distributed, Pan-European, multi-disciplinary research infrastructure for big social data analytics, coupled with the consolidation of a cross-disciplinary European research community, aimed at using social mining and big data to understand the complexity of our contemporary, globally-interconnected society. SoBigData++ is set to advance on such ambitious tasks thanks to SoBigData, the predecessor project that started this construction in 2015. Becoming an advanced community, SoBigData++ will strengthen its tools and services to empower researchers and innovators through a platform for the design and execution of large-scale social mining experiments.

This document contains information on SoBigData++ core activities, findings and outcomes and it may also contain contributions from distinguished experts who contribute as SoBigData++ Board members. Any reference to content in this document should clearly indicate the authors, source, organisation and publication date.

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

GLOSSARY

EU	European Union
EC	European Commission
H2020	Horizon 2020 EU Framework Programme for Research and Innovation
VRE	Virtual Research Environment
OIDC	OpenID Connect
JWT	JSON Web Tokens
WP	Work Package

TABLE OF CONTENTS

1 Relevance to SoBigData++

1.1 Purpose of this document

This document reports the releases of the social mining computational engine, the online coding and workflow design frameworks, and the online science monitoring dashboard available in the e-infrastructure at M12. It collects the design principles and reference architectures of the released solutions.

1.2 Relevance to project objectives

One of the main goals of the SoBigData++ project is to support cross-disciplinary research and innovation on the multiple aspects of social complexity from combined data-driven and model-driven perspectives, possibly implementing Open Science practices by means of exploiting an integrated platform where executions can be repeated, compared, discussed and logged. The common facilities listed in this deliverable, that have been developed and made available in the e-infrastructure during the first year of the project, facilitate and support the above-mentioned activities.

1.3 Relation to other work packages

The e-infrastructure common facilities pave the way for the creation of a platform where interdisciplinary tools, methods, and services can be contributed by Work Package 8 and Work Package 10, towards a view where these tools, methods, and services can be shared according to tailored policies, and easily combined.

1.4 Structure of the document

The remainder of the document is as follows: Section 2 briefly introduces the e-infrastructure logical architecture as a whole, describing how the common facilities relate to the rest. Section 3, section 4 and section 5 reports on the e-infrastructure common facilities available to date. Specifically, section 3 describes the Social Mining Analytics Engine design principles and reference architecture. Section 4 illustrates the work performed during the period for what concerns the Online Coding and Workflow integrated, namely RStudio and JupyterHub. Section 5 reports on the Online Science Monitoring Dashboard design principles and architecture released at Month 12. Finally, Section 6 describes the Release Management procedure that has been put in place for the software continuous integration and delivery in the e-infrastructure.

2 The SoBigData e-infrastructure logical architecture

The SoBigData Infrastructure primary objective is not only to support social mining practitioners with seamless access to datasets and methods of interest, rather provide them with a platform for the design and execution of large-scale social mining experiments accessible seamlessly on computational resources from the project cloud.

SoBigData++ WP9 is called to support the development of the SoBigData e-infrastructure in close collaboration with other work packages that are respectively called (a) to operate the infrastructure to provide virtual access to the integrated resources (WP7), (b) integrate existing and newly collected datasets in the infrastructure (WP8), and (c) integrate existing tools and methods for mining social data in the infrastructure (WP10). Within this context, WP9 puts in place actions comprising: (i) studies and definition of best practices/policies for the harmonization of federated resources available at the local infrastructure sites; (ii) support for adaptation of existing resources to the identified best practices; and (iii) realization of VREs supporting scientists in benefitting from the integration of the federated resources and infrastructures.

The e-infrastructure common facilities provide the tools and the access to the computational resources, that are needed to enact such practices, by relying on the e-infrastructure service whose logical architecture is outlined in the following.

2.1 Logical architecture and common facilities

Figure 1 shows the e-infrastructure logical architecture, with the common facilities for the Science Monitoring and the Online coding and workflow design on top.

The e-infrastructure is built by exploiting the web-accessible virtual machines operated and provisioned by D4Science.org together with services for their management and administration (federated resources). This layer also includes the enabling services that are required to support the operation of all services and VREs. Specifically, it includes a resource registry service, to which all e-infrastructure resources (data sources, services, computational nodes, etc.) can be dynamically (de)registered and discovered by users and other services; an authentication and authorization service, as well as auditing services, capable of granting and tracking access and usage actions from users; a VRE management approach for deploying specialized VREs/VLabs based on a selected subset of applications.

On top of this layer, two engines, namely the Storage Engine and the Social Mining Analytics engine are built to deliver common facilities to the users. The former operates transparently to the users and virtualises the available storage technologies and their actual physical hardware. The latter, to date, is realized by exploiting and enhancing the gCube DataMiner engine operated by D4Science.org in order to federate and integrate existing software frameworks provided within SoBigData++ consortium members.

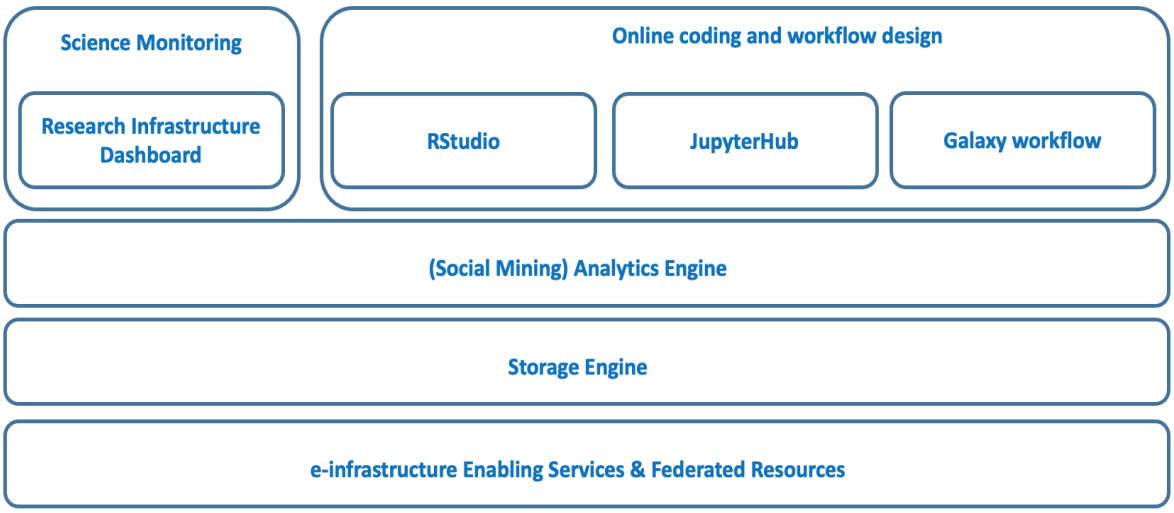

Figure 1: The e-infrastructure common facilities logical architecture

The online coding and workflow design framework and the science monitoring dashboard sit on top of these facilities and exploit them to access or store data (Storage Engine) and to execute methods over this data (Analytics Engine).

The online coding and workflow design frameworks enable users to create live documents with code, text and visualizations that capture the whole research process: developing, documenting, and executing code, as well as communicating the results. The SoBigData++ coding common facilities available and integrated into the e-infrastructure to date manifests in two of three technologies of the figure above, namely (i) RStudio, for performing online coding with R language, aiming at performing statistical analyses, and (ii) JupyterHub, to create and execute Jupyter notebooks that are capable to access seamlessly to computational environment and resources available on the e-infrastructure.

The online Science monitoring application manifests in the Research Infrastructure Dashboard (RID). RID monitors and quantifies the outputs of the e-infrastructure in the scholarly communication ecosystem. It also identifies every research product (publications, datasets, software, and other types) produced thanks to the SoBigData infrastructure that is available in the OpenAIRE Research Graph (https://graph.openaire.eu), which as of December 2020 includes more than 120 millions research products linked to funding projects, organizations, authors.

3 Social Mining Analytics Engine design principles and architecture

The Social Mining Analytics Engine (SMAE) includes a set of services and components for performing data processing and mining on information sets. Users' analytical methods can be developed by almost any programming language (R, Java, Python, Fortran, Octave, etc.) and then imported via the Software and Algorithms Importer and executed via the DataMiner System. The Smart Executor System completes SMAE by adding support for the execution of scheduled and repetitive tasks.

The Social Mining Analytics Engine provides:

- support for the execution of analytical methods on multi-core computational nodes;
- support for multi-tenancy and concurrent access;
- support for Auditing;
- support for the execution of analytical methods on sets of computing nodes;
- support for scheduled and repeated execution;
- support for reproducibility and repeatability of the computed results thanks to the provenance information automatically generated by the engine;
- support for the standard Web Processing Service (WPS) protocol.

3.1 Software and Algorithms Importer and DataMiner systems

The Software and Algorithms Importer (SAI) is an interface allowing the registered user to easily import analytical methods into DataMiner, which in turn, publishes these methods as-a-Service and manages their execution in a multi-tenant and concurrent computational environment. Additionally, it allows scientists to update their analytical methods with just a few interactions with the interface and without following long software re-deploying procedures each time. In summary, SAI produces processes that run on the computing computational environment and are accessible via the WPS standard.

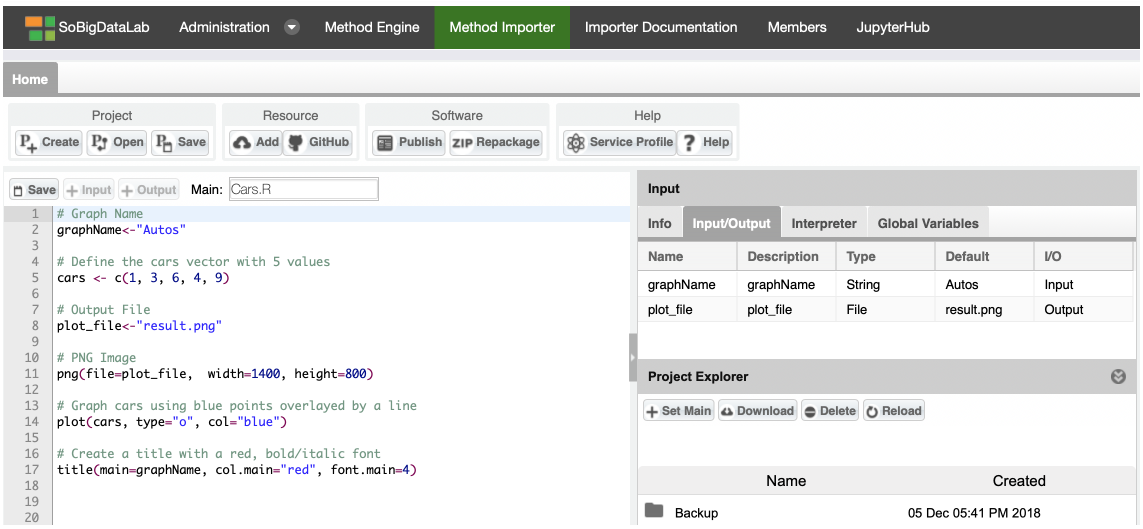

Figure 2: The Software Algorithms Importer (SAI) in the SoBigData Lab VRE

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

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

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

Once imported, an analytical method becomes exploitable through the DataMiner component.

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

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

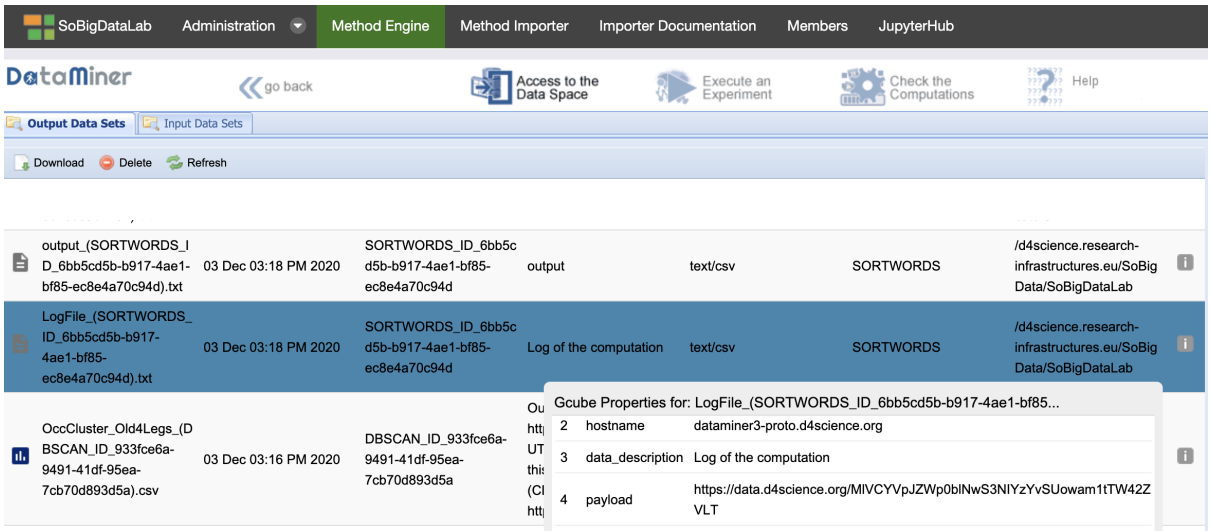

Figure 3: The DataMiner System in the SoBigData Lab VRE

The Execution Space presents two panels:

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

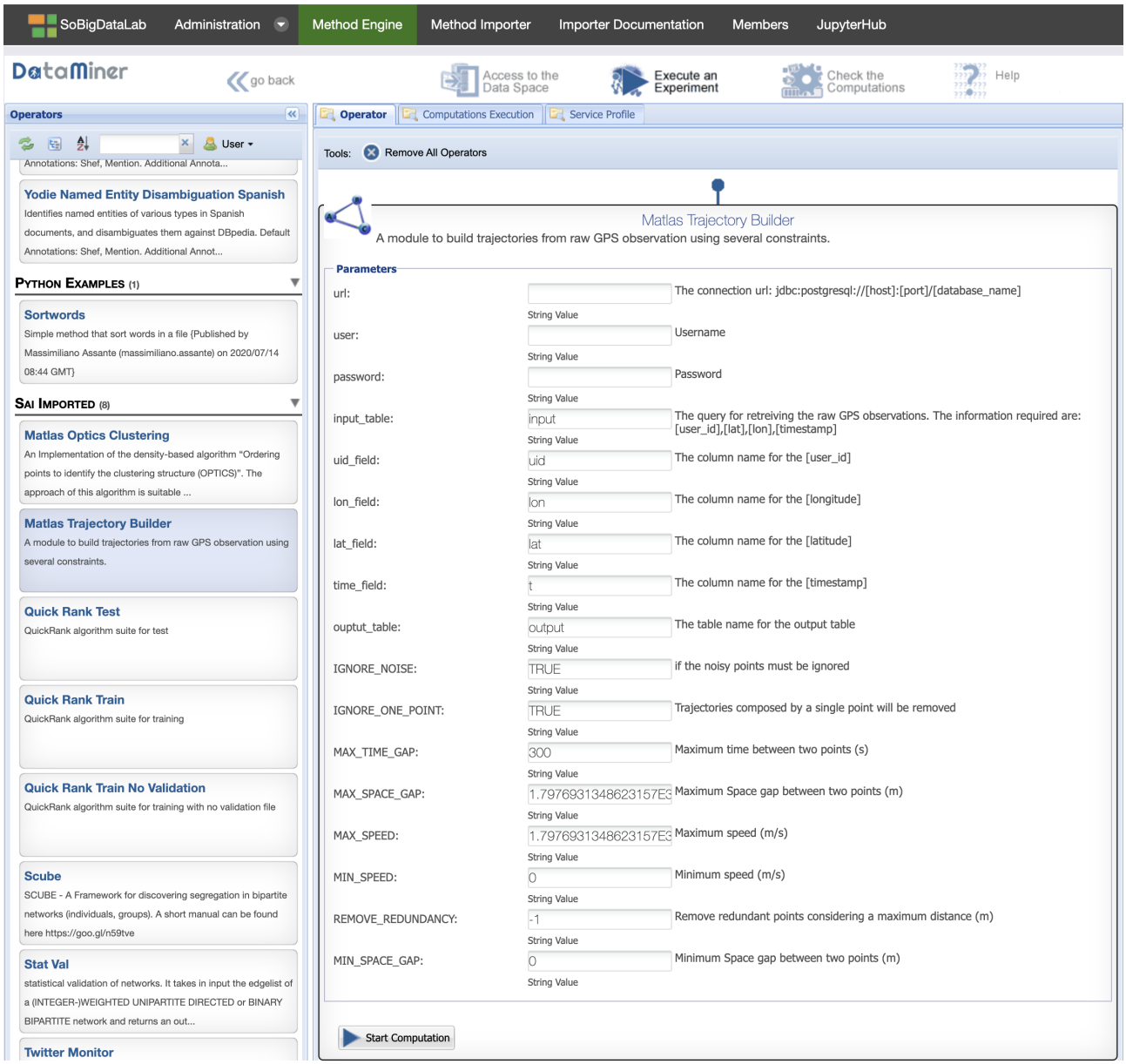

Figure 4: The M-Atlas Method interface in the DataMiner System

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

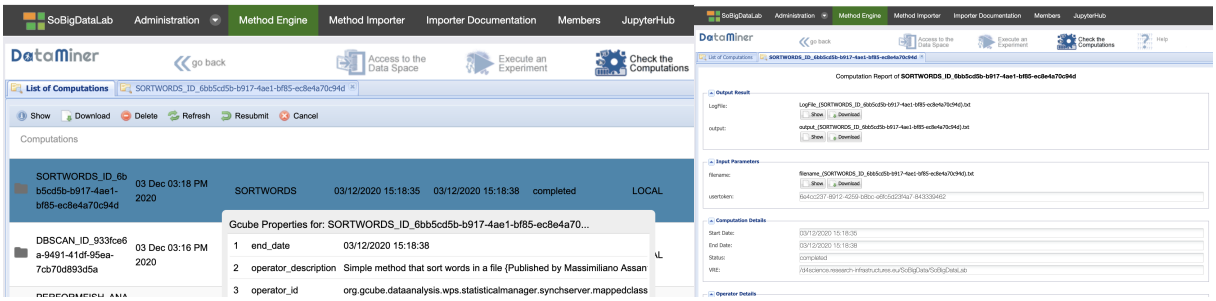

Figure 5: The DataMiner System computations space

3.1.1 Design principles and implementation

DataMiner (DM) is a service based on the 52North WPS implementation. The DM is developed in Java as a web service running on an Apache Tomcat instance, endowed with gCube libraries.

The complete DM architecture is made up of two sets of servers (clusters) that operate in a VRE (Figure 6): the master and the worker clusters. In a typical deployment scenario, those clusters are made up each of 16 servers (eg, Ubuntu 18.04 LTS x86 64 with 16 virtual CPUs, 32GB of random-access memory, 100GB of ephemeral storage) managed by a load balancer - HaProxy - that distributes the requests uniformly to these servers. Each machine is endowed with a DM service that communicates with the D4Science Resource Registry to notify its presence and capabilities. The balancer is indexed on the Resource Registry and is the main access point to interact with the DMs. The worker cluster serves cloud computations

The master and the worker clusters are dynamically provisioned by D4Science through an orchestration engine endowed with management software that uses Ansible scripts to configure multi-tier applications in a reliable and consistent manner.

When a WPS request comes to the master cluster balancer, it is distributed to one of the cluster services (master DM). The DMs host processes provided by several developers. In particular, two kinds of algorithms are hosted: *local* and *cloud* algorithms.

Local algorithms are directly executed on the master DMs and possibly use parallel processing on several cores and a large amount of memory. Instead, cloud algorithms use distributed computing with a MapReduce approach and rely on the DMs in the worker cluster (cloud nodes).

With respect to the standard 52 north implementation, DM adds a number of features enabling collaboration among users and repeatability and reusability of the results. This is achieved by exploiting the common services available in the infrastructure and in particular:

- Even if the computations can start with a file provided either as an HTTP link or embedded in a WPS execution request, the DataMiner accepts inputs from workspace folders shared among several users. This enables a group of people performing challenging experiments;
- the outputs of the computations are written onto the distributed cloud storage and are immediately returned to a client at the end of the computation. Afterwards, an independent thread also makes accessible this information via the workspace. Indeed, after every successfully completed

computation, a workspace folder is created that contains the input, the output, the parameters of the computation, and a provenance document summarising 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. This is the main way by which DataMiner fosters collaborative experimentation.

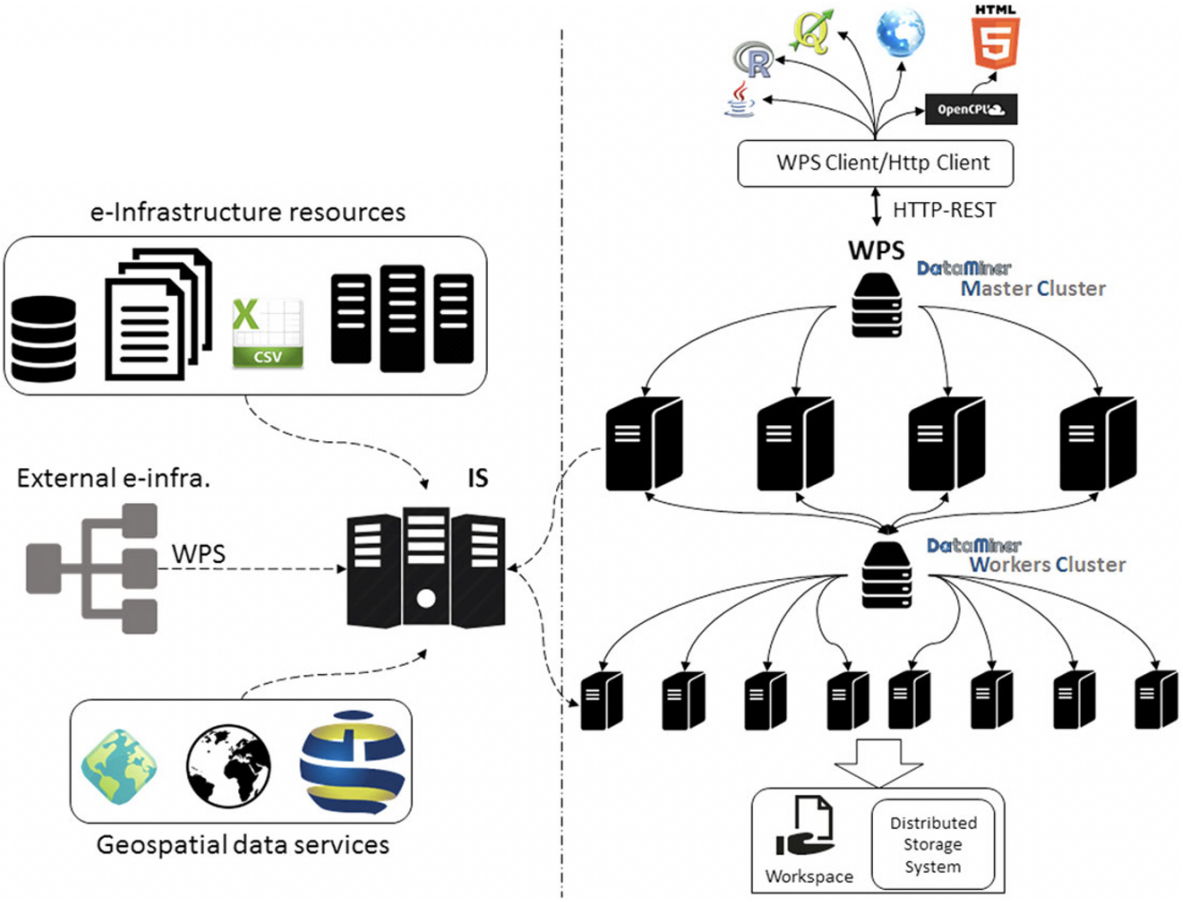

Figure 6: The DataMiner System architecture

The Software and Algorithms Importer and the Data Miner engine are available in the following git repository:

- Software and Algorithms Importer (enabling the import of the analytical method): https://code-repo.d4science.org/gCubeSystem/statistical-algorithms-importer
- Data Miner (engine): https://code-repo.d4science.org/gCubeSystem/dataminer
- Data Miner Client (client for Java applications): https://code-repo.d4science.org/gCubeSystem/data-miner-manager-cl
- Data Miner Pool Manager (enabling the deployment of analytical methods): https://code-repo.d4science.org/gCubeSystem/dataminer-pool-manager

- Data Miner Manager User interface (the main user interface component): https://code-repo.d4science.org/gCubeSystem/data-miner-manager
- Data Miner WIdget (a component of the main user interface): https://code-repo.d4science.org/gCubeSystem/data-miner-manager-widget
- Data Miner Executor (a packed user interface enabling just the execution of the methods): https://code-repo.d4science.org/gCubeSystem/data-miner-executor

3.2 Smart Executor

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

It is typically used to periodically invoke a computational method imported into the DataMiner. A common case is the monthly aggregation of raw data that is gathered daily. In this example, the data aggregation is implemented via a computational method imported in DataMiner; the monthly execution of it is instead realized by a task of the Smart Executor service. The combination of the two services will allow either a single user or all the virtual laboratory users to access the collection of aggregated data through the Workspace. It enables useful patterns to manage repetitive tasks that can be scheduled with a known frequency.

3.2.1 Design principles and implementation

The SmartExecutor service allows to execute *Tasks* and monitor their execution status. Each instance of the SmartExecutor service can run the Tasks related to the plugins available on such an instance. Each instance of the SmartExecutor service publishes descriptive information about the co-deployed plugins in the Resource Registry.

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

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

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

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

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

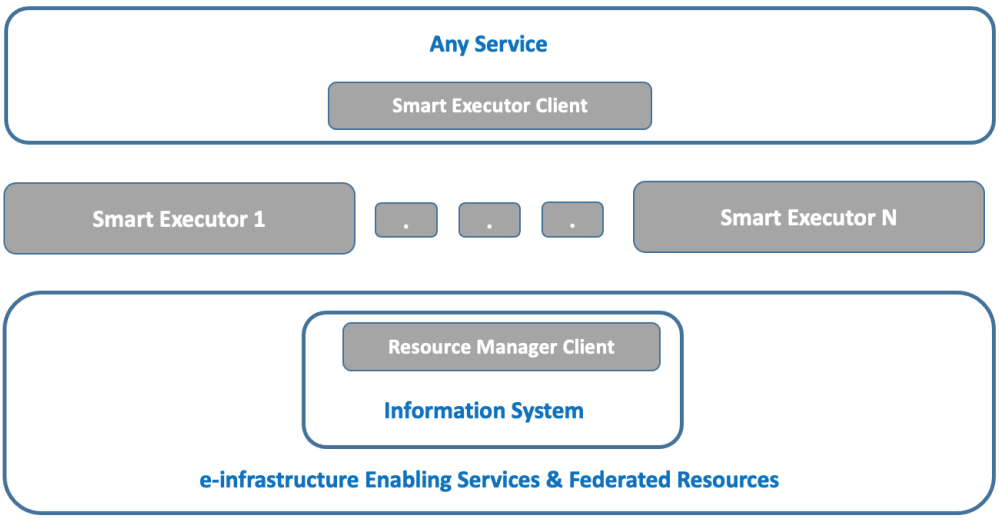

Figure 7: The Smart Executor System architecture

The Smart Executor components are available in the following git repository:

- Smart Executor service: https://code-repo.d4science.org/gCubeSystem/smart-executor
- Smart Executor Java client: https://code-repo.d4science.org/gCubeSystem/smart-executor-client

4 Online Coding and Workflow design principles and architecture

The online coding and workflow system enables users to create live documents with code, text and visualizations that capture the whole research process: developing, documenting, and executing code, as well as communicating the results. The SoBigDataPlusPlus coding common facilities available and integrated into the e-infrastructure at M12 are represented by two applications, namely (i) RStudio, for performing online coding with R language to perform statistical analyses, and (ii) JupyterHub, that allows executing Jupyter notebooks providing users with access to computational environments and resources of the e-infrastructure.

In the following, these two available facilities are presented, together with their design and logical architecture solution.

4.1 RStudio

The RStudio allows performing online statistical analyses with R. SoBigData makes the RStudio Application accessible on the e-infrastructure ensuring its operation and orchestration in a cluster. The cluster is composed of multiple hosts, each of which is assigned in exclusive mode to a user for an entire online session. At the end of the session, all the content stored in that host may be removed by the e-Infrastructure enabling services and a new host assigned to the user. The user can persist the R-session into the private Workspace (exploiting the Cloud storage via the Storage Service) that is accessible through the RStudio Application.

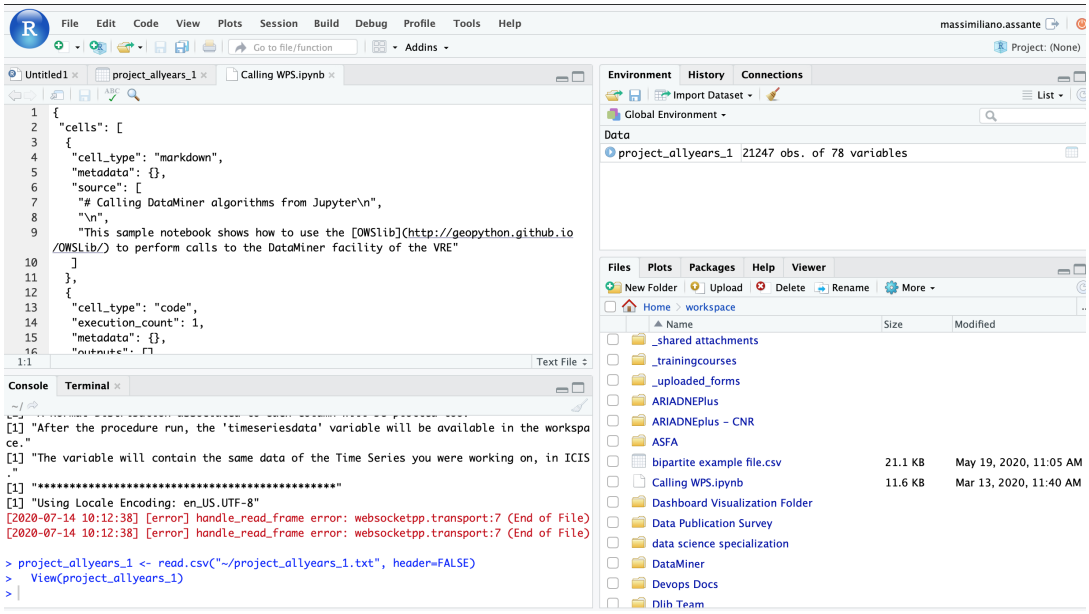

Figure 8: VRE The RStudio Interface with private Workspace mounted transparently

4.1.1 Design principles and implementation

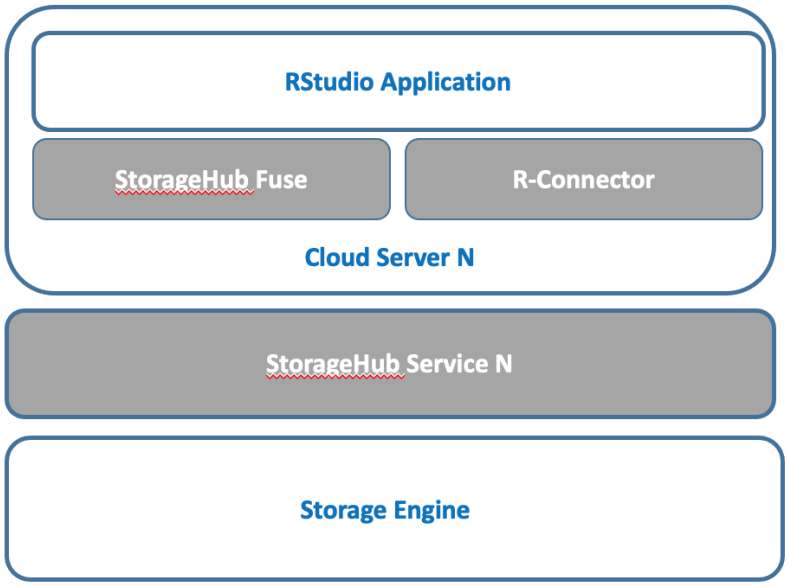

Figure 9: The RStudio solution reference architecture

Figure 9 shows the Design and implementation reference architecture of the RStudio coding solution facility available in the e-infrastructure. The RStudio applications are deployed on a cluster of cloud servers and are shared equally among the users. The user starting new RStudio sessions (from the e-infrastructure Virtual Lab) are assigned automatically to one of the instances.

As above mentioned, users can persist the R-sessions into the private Workspace that is accessible through the RStudio Application. This feature is enabled by the "*StorageHub Fuse"* component, a custom component developed during the period which exploits the Linux Filesystem in Userspace (FUSE)[1] software interface to realise a bridge between the Linux filesystem and the SoBigData workspace interface, namely the *StorageHub service*. The StorageHub Service is replicable and a proxy (HA-Proxy) on top is used for proxying requests to the deployed instances of it. StorageHub relies on 2 different storage technologies to store the metadata of the items being stored, namely Apache Jackrabbit as metadata repository and PostgreSQL as Apache Jackrabbit Back-end Database. One other distinguishing feature of this service is that the actual payload of the items can be stored on a number of in house and commercial storage technologies, for instance in a MongoDB Cluster, but also on other types, including Cloud Storages solutions (e.g., Amazon S3) that are provided by the e-infrastructure Storage Engine. Finally, the *"R-Connector"* component, a custom component developed during the period, connects the User SoBigData Identity and the R Application User identity transparently by relying on the OpenID Connect (OIDC) standard. OIDC is an identity layer over oAuth2 which uses JSON web tokens (JWT), allowing

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

third-party applications to verify the identity of the User and to obtain basic user profile information. During the past years, OIDC has become the leading standard for single sign-on and identity provision on the Internet.

The *StorageHub Fuse Integration* library, the *R-Connector,* and the *StorageHub service* source code are available in the following git repository:

- StorageHub Fuse Integration: https://code-repo.d4science.org/gCubeSystem/sh-fuse-integration
- R-Connector: https://code-repo.d4science.org/gCubeSystem/RConnector
- StorageHub Service:https://code-repo.d4science.org/gCubeSystem/storagehub

4.2 JupyterHub

Notebooks are becoming the de-facto standards in order to provide an easy online coding system combined with interactive computing. In the context of SoBigData++, the deployment and operation of a Notebooks service integrated with the SoBigData e-infrastructure has to satisfy the following requirements:

- The service should be integrated and accessible via a number of SoBigData++ Virtual Research Environments (VREs);
- The service should be integrated with the SoBigData Authentication/Authorization system;
- The access to the SoBigData Workspace should be implemented, in order to provide access to VRE Workspaces users files within the notebooks environment;
- The users should be able to access different flavours of notebooks both in terms of software pre-installed and resource needed;
- In addition to the SoBigData Workspace, users need persistent storage to save a working copy of their notebooks across sessions;
- The users should be able to publish the notebook to the SoBigData catalogue.

Considering that the SoBigData e-infrastructure is powered by D4Science and the positive experience of integration with D4Science performed during the AGINFRA+ project, the JupyterHub deployment over Kubernetes cluster has been selected in order to offer a notebook solution with seamless access from the SoBigData++ VREs graphical environment.

4.2.1 Design principles and implementation

Figure 10 shows the reference architecture of the JupyterHub solution made available on the SoBigData e-infrastructure. The Kubernetes deployment has been selected to offer a way to automatically provision notebooks servers as containers, with the ability to select the image flavours to run and the resource limits (in terms of CPU and RAM). Kubernetes offers an easy way to scale the deployment, by adding new workers to the existing installation, thus allowing extending the capacity of the service if needed.

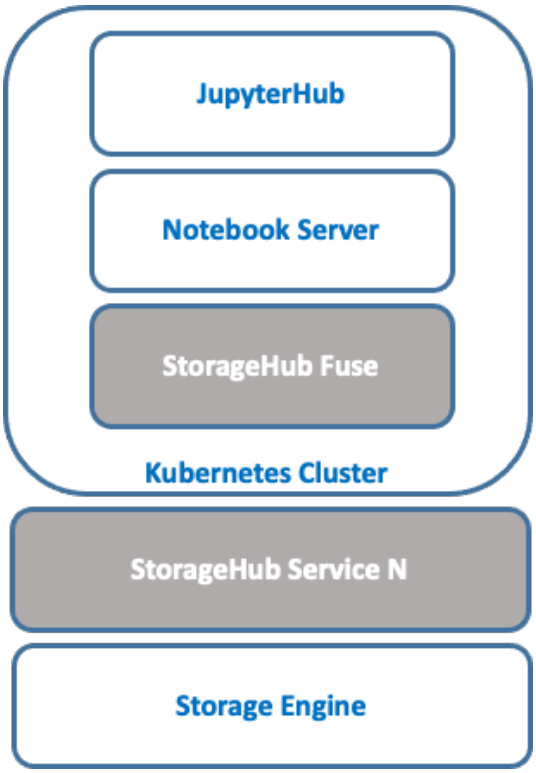

Figure 10: The JupyterHub solution reference architecture

For the purpose of the JupyterHub installation over the Kubernetes cluster, the project Zero to JupyterHub[2] has been selected as it offers many customizable hooks that are needed for the integration into the SoBigData environment. First of all, the possibility to extend the authentication system for the JupyterHub in order to use the SoBigData Authn/Authz framework[3]. A new JupyterHub Authenticator class has been implemented for the purpose of taking a user personal token from a VRE and using it to get user account details stored in JupyterHub DB. JupyterHub extensions specific for SoBigData are available[4], together with other customizations developed for EGI needs. Further integration activities were required to expose the JupyterHub GUI within the VRE portal, this was needed in order to include the GUI as an iframe.

JupyterHub is also providing customizable hooks to implement a selection of Notebooks images to run and related flavours. A default image has been configured initially derived from the Datascience notebooks distribution[5], with further libraries installed by default. The image is automatically built and published to DockerHub so it can be easily made available to the Kubernetes cluster. A default profiles list has also been included, so users can select the flavours of the notebooks to run as depicted in Figure 11.

[2] https://zero-to-jupyterhub.readthedocs.io/en/latest/
[3] https://dev.d4science.org/authorization
[4] https://github.com/EGI-Foundation/egi-notebooks-hub
[5] https://github.com/EGI-Foundation/egi-notebooks-images/tree/master/single-user-d4science

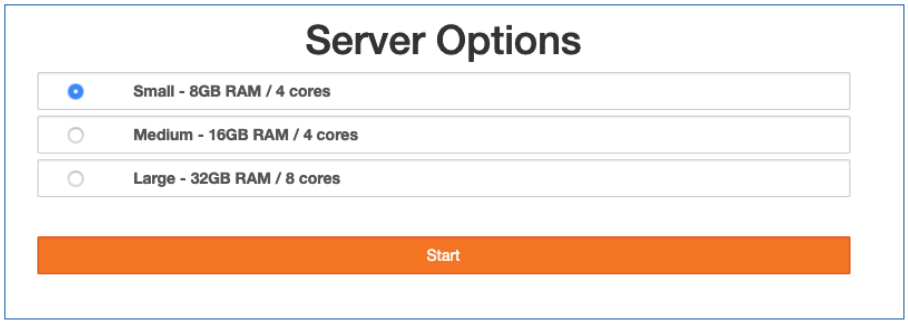

Figure 11. Notebooks server options

Further integration steps have been performed in order to implement access to the D4Science workspace. The access is implemented via the StorageHub Fuse library, which allows mounting the user workspace folders in the notebook environment. The library integration[6] allows the workspace to be mounted in a sidecar container of the user's notebook pod and made available under the user's home.

Finally, the Kubernetes cluster has been configured with an NFS server which is needed to host the users' volumes in order to offer persistent working folders.

More details of the integration activities performed are shown in Figure 12.

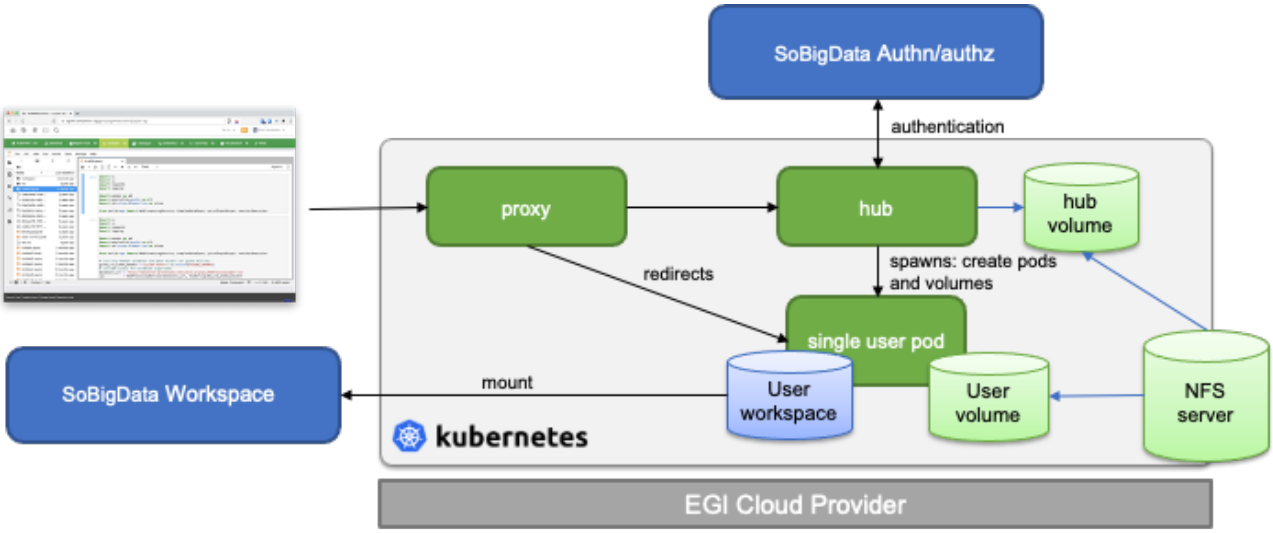

Figure 12: The JupyterHub deployment for the SoBigData e-infrastructure

The first implementation of the service has been made available to users of the SoBigDataLab VRE in November 2020 and members of WP8 have been able to do initial tests. WP8 has also requested the integration of new libraries as base notebooks image and more will follow in the context of WP7.

[6] https://github.com/EGI-Foundation/egi-notebooks-images/tree/master/d4science-storage

The currently deployed cluster has enough capacity to offer concurrent access to 80 users each with notebooks of 8 GB RAM (the RAM available on the cluster is the limiting factor). The initial uptake of the service will be monitored and the resources available on the cluster extended accordingly.

In order to improve the operation of the cluster, automatic backup of the NFS server volume will be implemented together with improved cluster monitoring. The plan is to provide a dedicated Graphana dashboard to aggregate the data coming from the probes installed on the cluster nodes and to configure alarms for administrators.

From the functional point of view, one of the requirements of the project is to let the user publish the notebooks to the SoBigData catalogue. Sharing of notebooks is already available through the Workspace and also publishing into the catalogue can be performed via the Workspace. However, a future extension to publish notebooks directly from the Jupyter environment will be analyzed and possibly implemented to further simplify and promote the publication of notebooks.

The current deployment of the service does not support multi-site clusters; hence the Kubernetes resources are provided by one D4science.org site (GARR Napoli). Investigations will be performed in order to support the deployment over multiple sites, which will also improve the availability of the service in case of upgrades and site incidents.

5 Online Science Monitoring Dashboard: design principles and architecture

The Online Science monitoring dashboard monitors and quantifies the outputs of the SoBigData infrastructure in the scholarly communication ecosystem. It identifies every research product (publications, datasets, software, and other types) produced thanks to the infrastructure that is available in the OpenAIRE Research Graph (https://graph.openaire.eu), which as of December 2020 includes more than 120M research products linked to funding projects, organizations, authors. Once identified, the products are analysed to produce statistics and charts with indicators about different aspects of Open Science and research impact (e.g. Open Access, links between research products, data re-use). The dashboard also acts as a single entry point for users to discover, search, browse, and get access to research products related to the infrastructure and hosted in several scholarly communication sources (e.g. repositories, journals, archives).

The identification of SoBigData products is performed by analysing the OpenAIRE Research Graph looking for links to the two funding projects of the infrastructure (SoBigData and SoBigData++), which can be found in the metadata records describing the research products or in the acknowledgement statements of the full-texts of publications. For the identification of funding information in the full-texts, a full-text mining algorithm is run over all the ~10 million Open Access full-texts aggregated in the OpenAIRE infrastructure. A dedicated full-text mining algorithm will also be implemented to identify publications that mention the SoBigData infrastructure or re-use its outputs.

The dashboard will also be connected to a Zenodo community dedicated to SoBigData, where researchers will transparently deposit when deciding to publish the research objects they have produced using the tools and services of the infrastructure. Users are also able to manually add research products to the dashboard, in case it has not been automatically identified, by using the Link functionality of the OpenAIRE EXPLORE portal (https://explore.openaire.eu) or the dashboard itself.

5.1 Design principles and implementation

The Online science monitoring dashboard is built with the OpenAIRE Research Community Dashboard [1] (https://connect.openaire.eu), a framework for the set-up and deployment of highly configurable portals for research communities and infrastructures. The capacity of the dashboards to support several configuration scenarios is the leading principle that guided the design in order to be able to address different requirements of different stakeholders (discipline-specific research communities, mature and young research infrastructures). The configuration options can be defined using a Manager Dashboard, that allows selected users - called *managers* or *curators* to:

- Configure the algorithms that OpenAIRE will use to identify the relevant products by specifying the funded projects, the data sources where the products and their descriptive metadata records are available (including specific Zenodo communities). When the dashboard serves a disciplinary research community, managers can also specify a list of keywords and subject terms: all the research products whose metadata matches such terms will be included in the dashboard. This latter option

is not available for dashboard serving research infrastructures. On the other hand, managers of a research infrastructure can use the Manager dashboard to define, fine-tune, and test the full-text mining rules.

- Manage end-users' claims: users of the dashboard can use the Link functionality of the OpenAIRE EXPLORE portal and the dashboard to assert that a product is relevant for the infrastructure. The manager can confirm or reject those assertions.
- Configure statistics and charts: OpenAIRE makes available a number of statistics (e.g., percentage of Open Access publications and data, number of publications linked to data and to software, growth of Open Access publishing through the years). The managers can decide which are public and which are private, for internal monitoring.
- Configure the look and feel of the dashboard, by choosing colours, fonts, and logos to reflect the identity of the infrastructure it serves.
- Personalise the content of the web pages, by enabling/disabling menu items and modifying the content of the dashboard pages, for example, to include useful instructions about best practices on Open Science to be followed by the researchers of the infrastructure.

The specified configurations are used by the other components in order to provide a unique and tailored experience. In particular, the configuration of the algorithms is used by the Oozie jobs that analyse the OpenAIRE Research Graph and enriches it by tagging the products that are relevant to the infrastructure. The products and their tags are then (i) indexed on a Solr cluster to support search and browse functionality on top of which the API (https://develop.openaire.eu) and the dashboard web GUI are built upon; (ii) analysed with the Statistics Tool based on Apache Impala for the calculation of statistics and the production of charts that are shown by the web GUI.

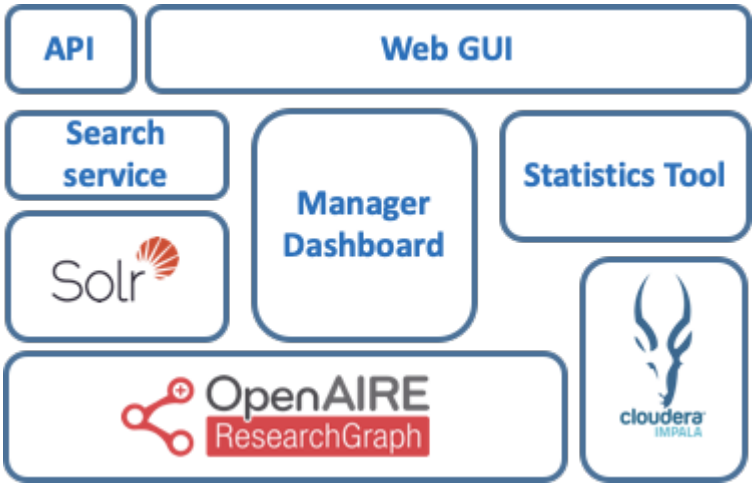

Figure 13: Architecture of the Online Science Monitoring Dashboard

6 Release Management: software continuous integration

The majority of the components described in this report are contributed by the gCube software system and other open-source systems as RStudio, JupyterHub, and many others.

gCube [2] is an open-source software system designed and implemented to enable the building and operation of a Service-Oriented Infrastructure supporting the definition of Virtual Research Environments (VREs). It is exploited by the D4Science infrastructure and in turn by SoBigData, its components are widely used in several application frameworks.

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

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

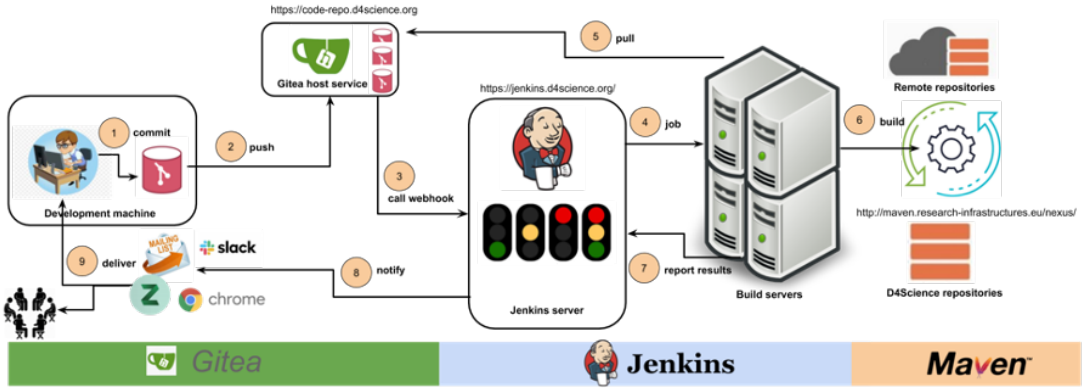

Figure 14: Continuous integration workflow

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

The workflow involves interaction with the Git repositories (managed by the D4Science Gitea instance https://code-repo.d4science.org/) and uses the Maven project management and comprehension tool.

6.1 Version Control System

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

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

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

6.2 Build Tool

A build tool is a tool that automates everything related to building a software project.

The advantage of automating the build process is to minimize the risk of humans making errors while building the software manually. Additionally, an automated build tool is typically faster than a human performing the same steps manually. gCube adopts Maven as its build tool.

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

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

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

6.3 Continuous Integration

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

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

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

- a Jenkins project builds always the master branch;
- if a Jenkins project is configured to build a second branch, this branch must generate a software artifact with a different version than master branch;

- the development of new features and bug fixing are done in dedicated branches (task branching);
- merges into master branch are performed when the feature/fix is stable;
- commits are not always pushed to the remote repository, but only when they add a significant, self-contained and consistent piece of working code;
- the master branch MUST be always in a releasable state

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

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

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

REFERENCES

[1] Baglioni M. et al. (2019) ***The OpenAIRE Research Community Dashboard: On Blending Scientific Workflows and Scientific Publishing***. In: Doucet A., Isaac A., Golub K., Aalberg T., Jatowt A. (eds) Digital Libraries for Open Knowledge. TPDL 2019. Lecture Notes in Computer Science, vol 11799. Springer, Cham. https://doi.org/10.1007/978-3-030-30760-8_5

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