PRIMAGE

Medical imaging
Artificial intelligence
Childhood cancer research

D2.3 – Deployment of intermediate version of hybrid open cloud and processing middleware

Project Full Title: *PRedictive In-silico Multiscale Analytics to support cancer personalized diaGnosis and prognosis, Empowered by imaging biomarkers*
Project acronym: *PRIMAGE*

Project type: Horizon 2020 | RIA (Topic SC1-DTH-07-2018)
Grant agreement no: 826494

Document information:

Deliverable no.	**D2.3**
Title	Deployment of Intermediate version of hybrid open cloud and processing middleware
Work Package	WP2
Dissemination level	PU
Nature	O
Responsible partner	UPV-I3M
Main Contact person	iblanque@dsic.upv.es dquilis@dsic.upv.es
DOI	10.5281/zenodo.7777471
Deliverable due submission date	31/08/2019
Deliverable actual submission date	02/09/2019

History of changes:

Change explanation	Pages affected	Change made by (Name & surname)	Date
Table of contents and complete initial version	All document	Ignacio Blanquer (UPV), J. Damian Segrelles (UPV)	29/7/2019
Description of the architecture	Sections 2-3	Ignacio Blanquer (UPV)	30/7/2019
Interactive Application	Section 4	Ignacio Blanquer (UPV)	31/7/2019
Introduction and Conclusions	Sections 1, 5	Ignacio Blanquer (UPV)	31/7/2019
A High-Throughput Compute Batch job Application	Section 4	J. Damian Segrelles (UPV)	12/8/2019
List of Abbreviations	1	J. Damian Segrelles (UPV)	19/8/2019
Improvements	All	J. Damian Segrelles (UPV)	19/8/2019
Proofreading	All	J. Damian Segrelles (UPV)	30/8/2019

Table of Contents

1. Introduction 5

1.1. Scope of the document 5

1.2. Target Audience 5

1.3. Structure of the document 5

2. Software Architecture Proposal 6

2.1. Security Components 6

2.2. Data Storage Components 8

2.3. Processing Components 9

3. Software Repositories, Documentation and Endpoints 13

3.1. Repositories for the project 13

3.2. Computing Back-end 13

3.3. Storage Back-end 13

4. Implementation of Sample applications 15

4.1. Application 1. A High-Throughput Compute Batch job 15

4.2. Application 2. An interactive application. 17

5. Conclusion 20

6. Acronyms and Abbreviations 21

Acknowledgement

This document is a deliverable of the PRIMAGE project. This project has received funding from the European Union´s Horizon2020 research and innovation programme under grant agreement no.: 826494.

1. Introduction

The objective of this deliverable is to show in practical terms the deployment of PRIMAGE (PRedictive In-silico Multiscale Analytics to support cancer personalized diaGnosis and prognosis) applications and services on top of a public cloud offering. The deliverable is of type "Other", which relates to a prototype. This report is an explanatory guide and a reference document for understanding and using the prototype.

1.1. Scope of the document

This document constitutes the third deliverable of WP2. It covers a first preliminary prototype that addresses a set of fundamental and mandatory requirements needed for the implementation of the applications in PRIMAGE. The document comprises the procedures that a user should follow to grant access to the resources, the URLs for the repositories of software, configuration descriptions and binaries, as well as the key services of this first version of the PRIMAGE platform.

1.2. Target Audience

The document is mainly intended for internal use, although it is publicly released. The main target of this document is the global team of technical experts of the PRIMAGE project that will access and use the resources. It also serves to the dissemination and exploitation partners to understand the relations with other projects in the landscape of EOSC (European Open Science Cloud).

1.3. Structure of the document

This document is organised in 6 sections, starting with this introduction. Section 2 summarizes the software architecture proposal, with special attention to the three layers (security, storage and processing), identifying the actual components to be used. Section 3 describes the repositories and URLs to be used in the frame of the project, and section 4 describes two canonical applications that can be used as examples. Finally, section 5 presents the conclusions and section 6 includes some references and acronyms.

2. Software Architecture Proposal

This section details the software architecture for the PRIMAGE platform. The details on the requirements, use cases and selection of technologies are given in deliverable D2.2. In this section, the components selected for each one of the services identified in the architecture will be described. First, an overview of the architecture is given in figure 1. The architecture identifies two types of resources and services. On one hand, cloud services deal with the resources related to the cloud offering. On the other hand, HPC (High Performance Computing) services interact with the supercomputer Prometheus. This deliverable will focus mainly on the cloud-based services. The Datacloud and processing architecture must provide resources to store data, host PRIMAGE services, run processing pipelines and guarantee the security of the system.

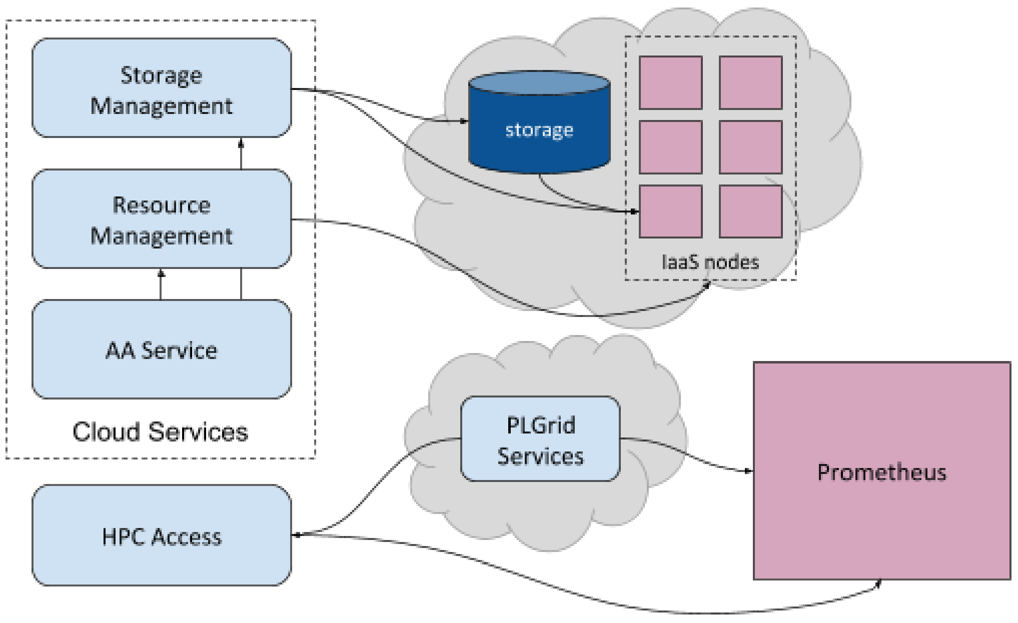


Figure 1. *Overview of the Cloud platform architecture.*

2.1. Security Components

The security components proposed at the security architecture (see figure 2) will guarantee that all users accessing the services are properly authenticated and authorised. For this purpose PRIMAGE selected the EGI-Checkin service (Authentication and Authorisation (AA) Service in figure 1), available in the EOSC marketplace (https://marketplace.eosc-portal.eu/services/egi-check-in), as it is widely supported by the EU Research Infrastructures and compatible with the main European academic authentication services. This first version of the platform will support the following requirements (see Deliverable D2.2 for more details):

- R3.1. Restrict access to applications.
- R3.2. Restrict access to job manager services.
- R3.3. Restrict access to platform services.
- R3.4. Restrict access to infrastructure services.
- R3.5. Restrict access to storages.

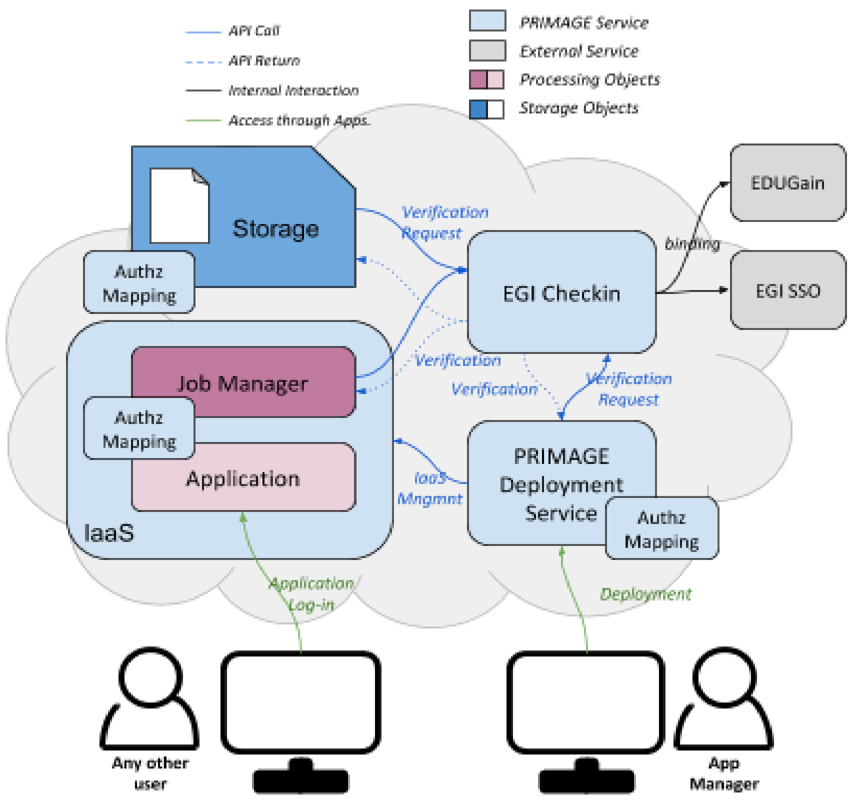
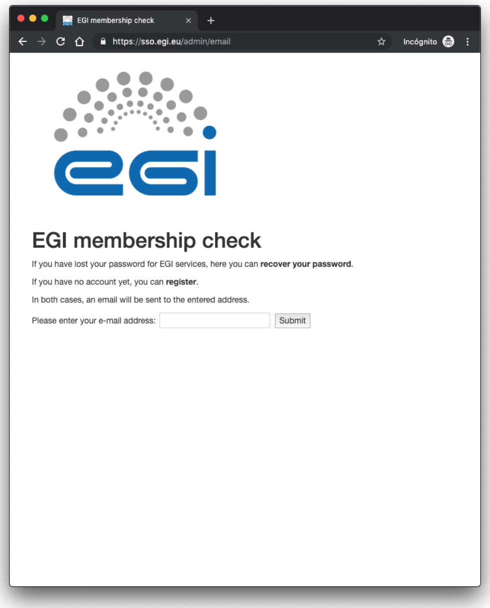


Figure 2. *Security architecture*.

In order to use the services two different options are possible::

- Create an EGI (European Grid Infrastructure) SSO (Single Sign On) account using the link https://sso.egi.eu/admin/email and follow the steps. After a short evaluation process, the account will be activated and the services will be accessible.
- Get authenticated throughout an existing IdP (Identity Provider) in https://aai.egi.eu.

Figure 3 shows two snapshots of the account creation.



https://sso.egi.eu/admin/email

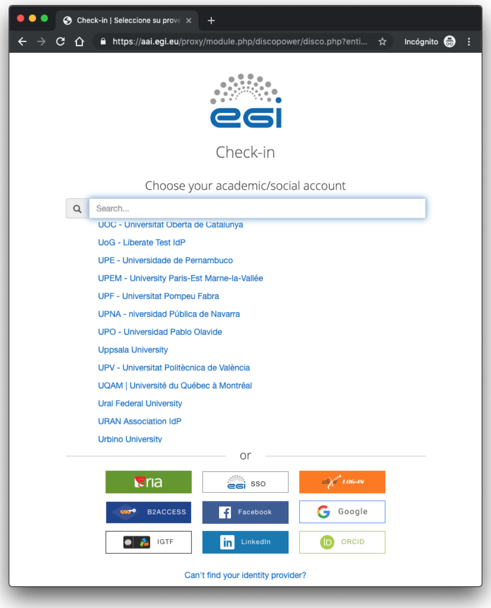
EGI membership check

If you have lost your password for EGI services, here you can **recover your password**.

If you have no account yet, you can **register**.

In both cases, an email will be sent to the entered address.

Please enter your e-mail address: Submit



https://aai.egi.eu/proxy/module.php/discopower/disco.php?enti...

Check-in

Choose your academic/social account

Search...

UOC - Universitat Oberta de Catalunya
UoG - Liberate Test IdP
UPE - Universidade de Pernambuco
UPEM - University Paris-Est Marne-la-Vallée
UPF - Universitat Pompeu Fabra
UPNA - niversidad Pública de Navarra
UPO - Universidad Pablo Olavide
Uppsala University
UPV - Universitat Politècnica de València
UQAM | Université du Québec à Montréal
Ural Federal University
URAN Association IdP
Urbino University

or

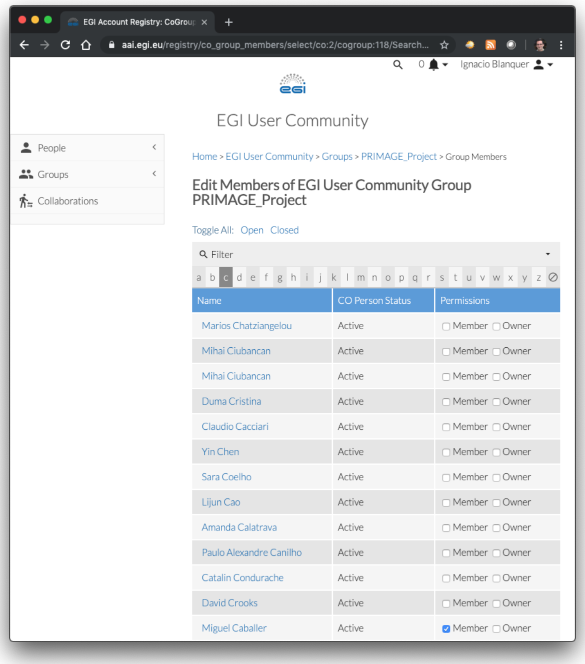
SSO · B2ACCESS · Facebook · Google · IGTF · LinkedIn · ORCID

Can't find your identity provider?

Figure 3. *Account creation snapshots*.

Once the user has been created, she will need to request the membership to the PRIMAGE_Project Check-in Group. This project is managed by UPV, and should be requested to dquilis@dsic.upv.es. In this request, the new user should provide the unique user name and the purpose of such access (e.g. the task where the user will develop her work).

The group admin will add the user to the group, so she can access the services directly. Figure 4 shows the process for adding a user to the group.


EGI Account Registry: CoGroup
aai.egi.eu/registry/co_group_members/select/co:2/cogroup:118/Search...
Ignacio Blanquer
People
Groups
Collaborations
Filter
a b c d e f g h i j k l m n o p q r s t u v w x y z

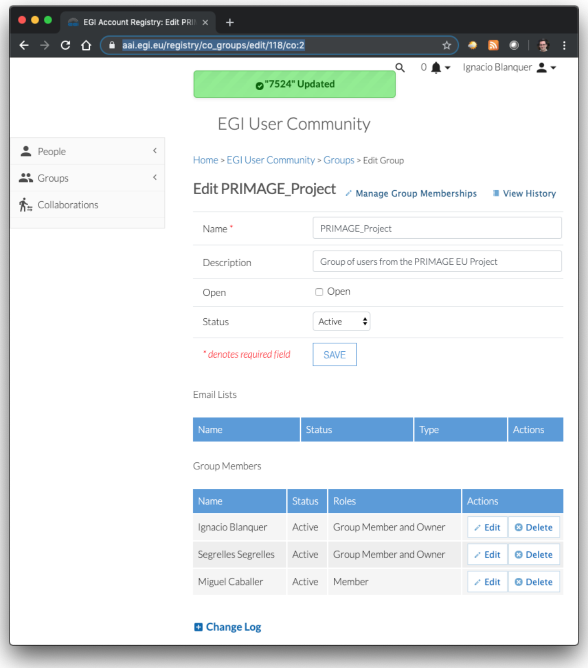
EGI User Community

Home > EGI User Community > Groups > PRIMAGE_Project > Group Members

Edit Members of EGI User Community Group PRIMAGE_Project

Toggle All: Open Closed

Name	CO Person Status	Permissions
Marios Chatziangelou	Active	☐ Member ☐ Owner
Mihai Ciubancan	Active	☐ Member ☐ Owner
Mihai Ciubancan	Active	☐ Member ☐ Owner
Duma Cristina	Active	☐ Member ☐ Owner
Claudio Cacciari	Active	☐ Member ☐ Owner
Yin Chen	Active	☐ Member ☐ Owner
Sara Coelho	Active	☐ Member ☐ Owner
Lijun Cao	Active	☐ Member ☐ Owner
Amanda Calatrava	Active	☐ Member ☐ Owner
Paulo Alexandre Canilho	Active	☐ Member ☐ Owner
Catalin Condurache	Active	☐ Member ☐ Owner
David Crooks	Active	☐ Member ☐ Owner
Miguel Caballer	Active	☑ Member ☐ Owner


EGI Account Registry: Edit PRI
aai.egi.eu/registry/co_groups/edit/118/co:2
Ignacio Blanquer
People
Groups
Collaborations

"7524" Updated

EGI User Community

Home > EGI User Community > Groups > Edit Group

Edit PRIMAGE_Project Manage Group Memberships View History

Name *	PRIMAGE_Project
Description	Group of users from the PRIMAGE EU Project
Open	☐ Open
Status	Active
* denotes required field	SAVE

Email Lists

Name	Status	Type	Actions

Group Members

Name	Status	Roles	Actions
Ignacio Blanquer	Active	Group Member and Owner	Edit Delete
Segrelles Segrelles	Active	Group Member and Owner	Edit Delete
Miguel Caballer	Active	Member	Edit Delete

Change Log

https://aai.egi.eu/registry/co_group_members/select/co:2/cogroup:118/Search.familyNameStart:c   https://aai.egi.eu/registry/co_groups/edit/118/co:2

Figure 4. *Administrative interface for adding the user Miguel Caballer to the PRIMAGE_Project group.*

2.2. Data Storage Components

According to the Requirements and the Use Cases identified in Deliverable D2.2, data storage in PRIMAGE must give access to data through standard protocols such as POSIX (Portable Operating System Interface). Depending on the application behaviour, data should be transferred in advance and copied locally or data could accessible remotely.

For this purpose PRIMAGE selects the use of ONEDATA (Global Data Access Solution for Science) (https://www.onedata.org/) as it is a technology integrated in EGI-Datahub, well known by the two partners leading WP2. It supports diverse back-end technologies (including S3) and provides POSIX mounted volumes. ONEDATA is secure and multitenant and it can be configured to provide near cache and the integration of local volumes.

Figure 5 shows the data storage architecture. This architecture considers the use cases to be supported in the first version of the platform and will support the following requirements (see Deliverable D2.2 for more details):

- R2.1. Create a Persistent storage.
- R2.2. Mount a Persistent storage on a running application.
- R2.3. Mount a Persistent storage on a local computer.

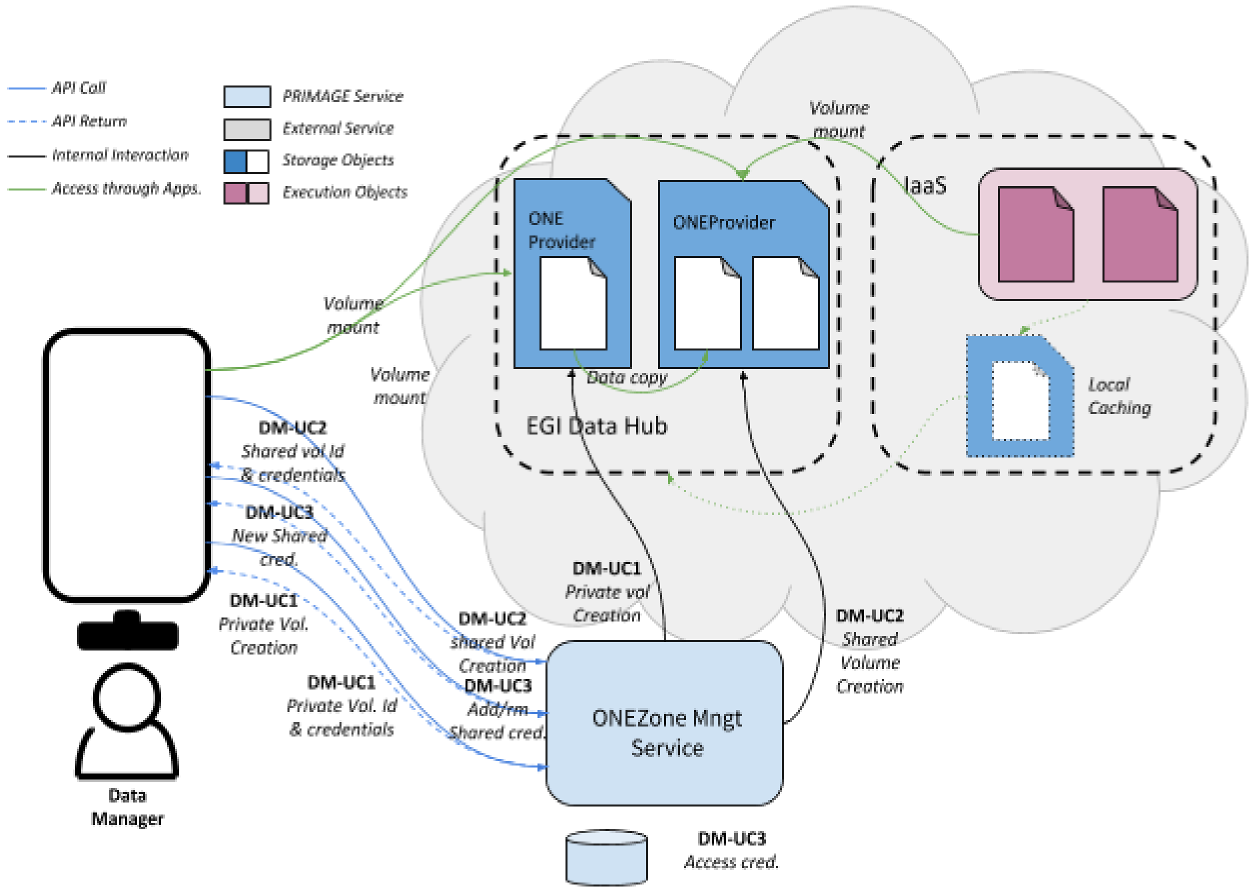


Figure 5. *Storage services Architecture.*

The storage objects are created by users who act with the role of data manager (who provides raw or curated data so that the services of the PRIMAGE platform can access them). It is important to indicate that an application administrator can act as a data administrator when requesting persistent storage for applications to install. Conceptually, we distinguish between persistent shared and private storage, although technically both are the same type of object and are provided by ONEDATA.

Once a persistent storage object is created on the platform, it must be accessible as a POSIX volume in applications deployed in cloud services. In addition, the data must be able to be downloaded to access resources that are not closely linked to data storage resources or due to network access limitations.

Data volumes must be shared, so that different applications from different users can access storage for inspection, download or even editing. Authentication will be provided by the central authentication service and authorization (see figure 2) will be managed at the storage resource level (for example, sharing access tokens).

2.3. Processing Components

Closely related to the storage, processing components will deal with the orchestration of resources and the execution of the different types of jobs, such as batch, HTC (High-Throughput Computing), parallel and interactive. First, in figure 6, it is defined the architecture for the deployment of applications and then, in figure 7, it is shown the architecture for the execution of jobs.

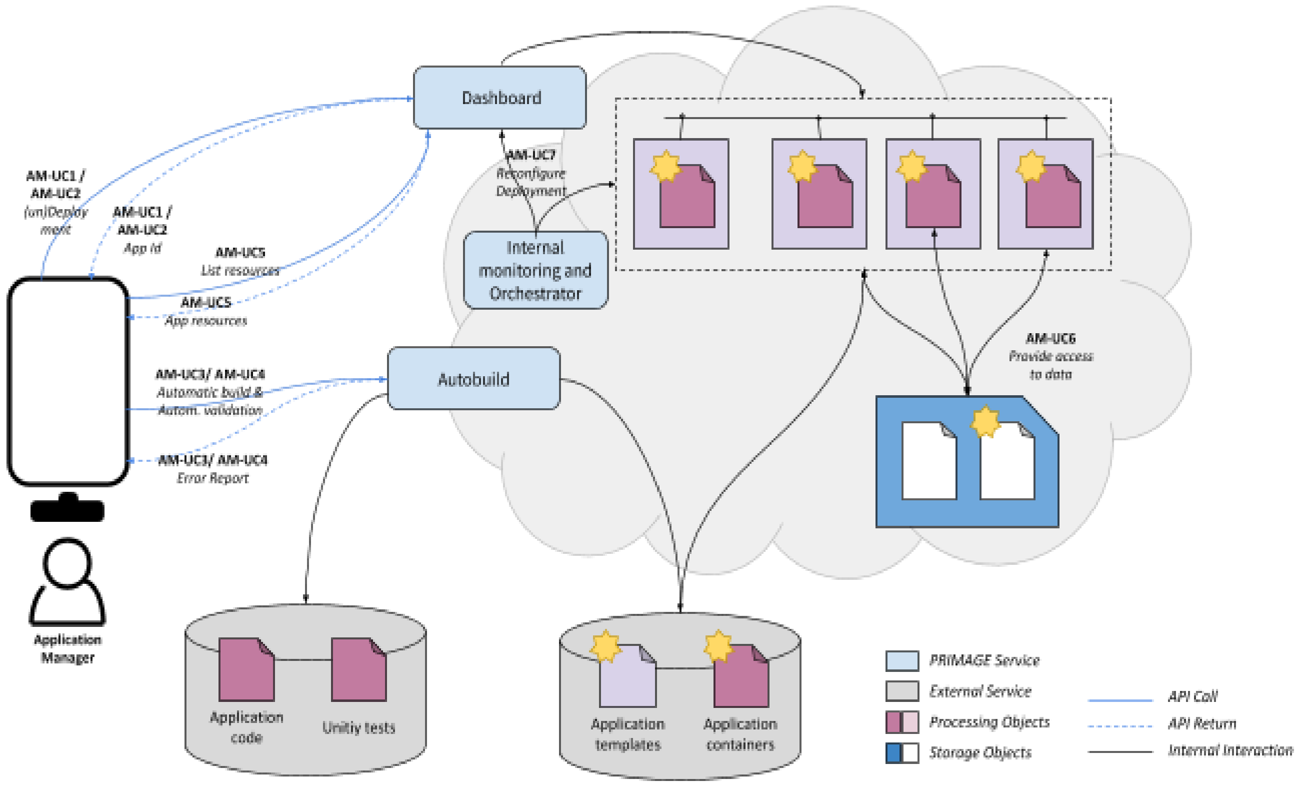


Figure 6. *Deployment of applications (services, frameworks) on the cloud resources.*

The requirements (see Deliverable D2.2 for more details) that will be addressed in this first version of the prototype will be:

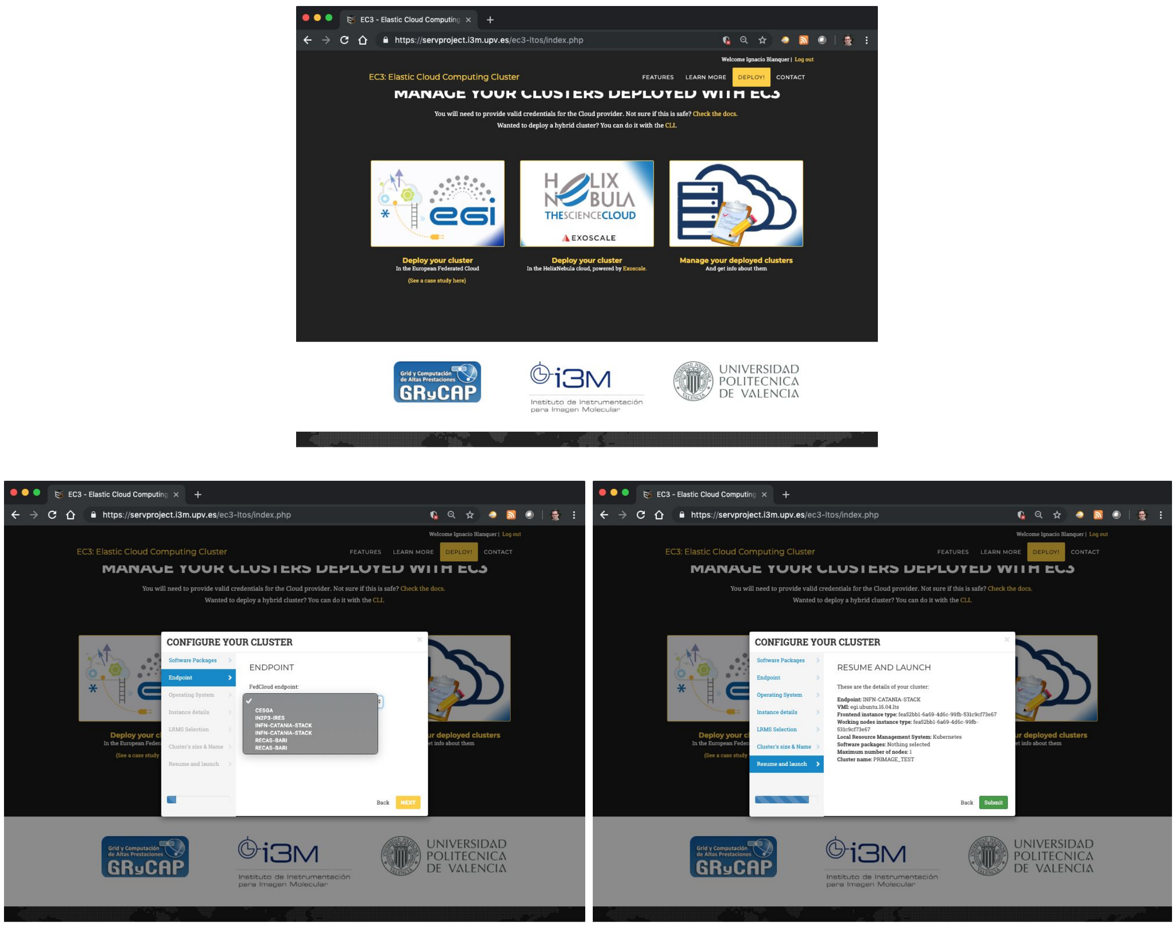
- R1.1. Application Deployment
- R1.2. Application Undeployment
- R1.3. List Applications
- R1.4. Inspect Application
- R1.5. Reconfigure Application
- R1.8. Automatic build of containers
- R1.9. Provide a private Registry
- R1.11. Submit a batch application
- R1.12. Submit an HTC application
- R1.13. Submit an interactive application
- R1.17. List applications submitted
- R1.18. Delete an application
- R1.19. Retrieve output of an application

Applications to be deployed in the cloud will be embedded as containers and application topologies are coded into templates. An application will comprise three parts:

- The application template, either coded into OASIS (Organization for the Advancement of Structured Information Standards) TOSCA (Topology and Orchestration Specification for Cloud Applications) or RADL (Resource and Application Description Language), which describes the topology of the application as a combination of nodes and configuration relations. Application templates will be mostly platform-agnostic and will be available in the code repositories (see next section).

- Configuration recipes with the necessary information to configure and install the software dependencies expressed in the application templates. Basically, recipes for the installation of Kubernetes[1] will be provided and stored in the Ansible Galaxy Repository[2].
- Container descriptions, with the application-specific software dependencies, which code the dependencies at service level.

For the deployment of applications, EC3 (Elastic Compute Cluster in the Cloud) LTOS (Long Tail Of Science) portal will be used, which deploys the Virtual Infrastructures in EGI Compute Cloud. Figure 7 shows several steps on the deployment of a K8s (Kubernetes) cluster.



You will need to provide valid credentials for the Cloud provider. Not sure if this is safe? Check the docs.
Wanted to deploy a hybrid cluster? You can do it with the CLI.

In the European Federated Cloud (See a case study here)
In the HelixNebula cloud, powered by Exoscale.
And get info about them

FedCloud endpoint:
CESGA
IN2P3-IRES
INFN-CATANIA-STACK
INFN-CATANIA-STACK
RECAS-BARI
RECAS-BARI

These are the details of your cluster:

Endpoint: INFN-CATANIA-STACK
VMI: egi.ubuntu.16.04.lts
Frontend instance type: fea52bb1-6a69-4d6c-99fb-531c9cf73e67
Working nodes instance type: fea52bb1-6a69-4d6c-99fb-531c9cf73e67
Local Resource Management System: Kubernetes
Software packages: Nothing selected
Maximum number of nodes: 1
Cluster name: PRIMAGE_TEST

Figure 7. *Deployment service through the EC3 Dashboard.*

The execution of jobs will be performed through Kubernetes. Kubernetes applications can be batch applications, HTC applications and interactive services. Therefore, it is a convenient backend for dealing with the requirements of the different types of applications in PRIMAGE. HTC batch applications can be coded as Kubernetes Jobs and interactive services as Statefulsets. K8s services can be created to provide external connectivity for interactive applications. Detailed descriptions are given in the last section of the document.

[1] https://github.com/indigo-dc/tosca-types/blob/master/examples/kubernetes_cluster.yaml
[2] https://galaxy.ansible.com/indigo-dc/kubernetes and https://github.com/indigo-dc/ansible-role-kubernetes

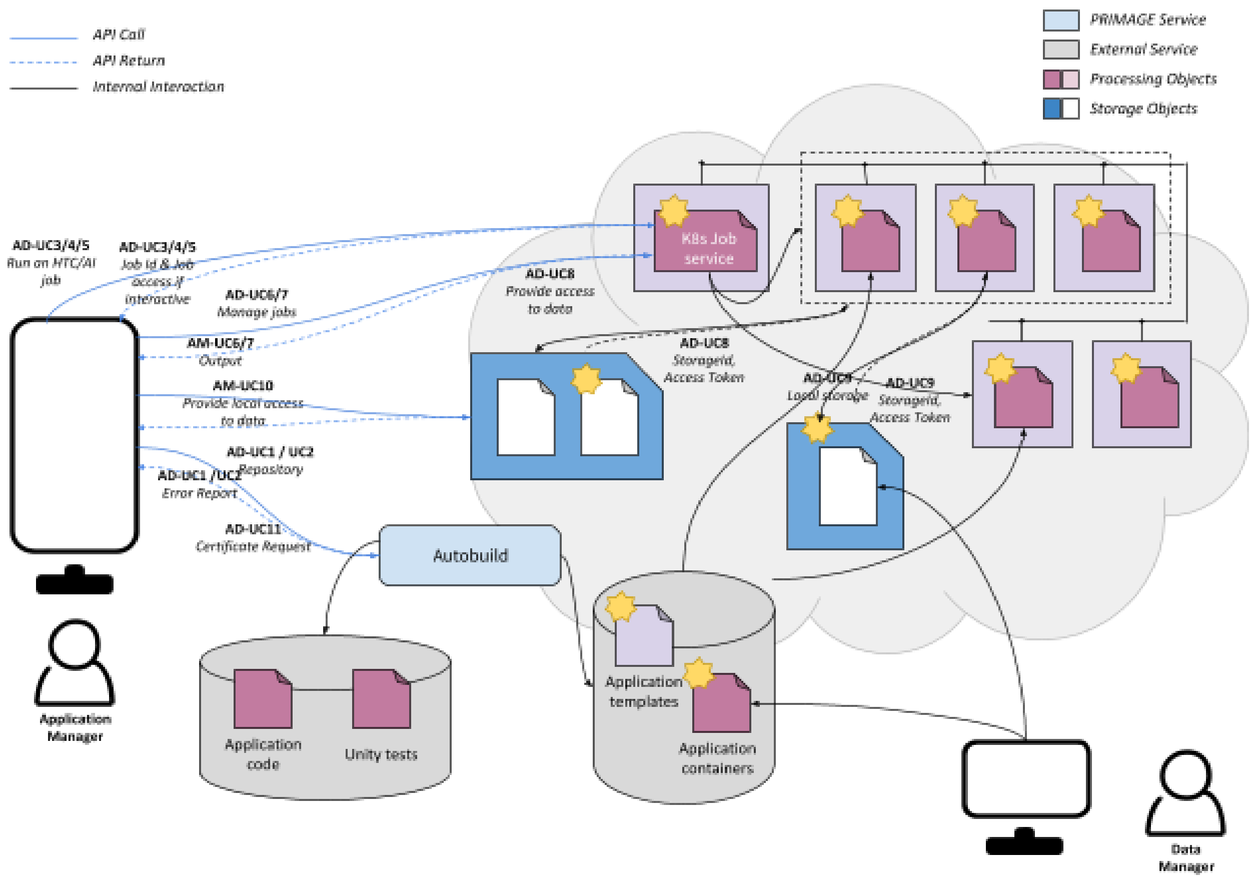


Figure 8. *Architecture for the execution of jobs on the PRIMAGE platform.*

3. Software Repositories, Documentation and Endpoints

This section includes the references to the software repositories, documentation repositories and service endpoints that will be used in the project. This should be carefully read by the developers of new applications.

3.1. Repositories for the project

The list of repositories of the project are the following:

- Source code repositories (https://gitlab.com/primageproject)
  - ec3client (https://gitlab.com/primageproject/ec3client). Customized tool to deploy elastic kubernetes clusters in the cloud backends of PRIMAGE.
  - Containers (https://gitlab.com/primageproject/containers). Container specifications for the dependencies of the applications to be run in the infrastructure.
  - Example of an image processing application to demonstrate the functionality of the platform (https://plg-cyfronet-01.datahub.egi.eu/#/onedata/data/a3f6c83ea357807e5ed5e3e8821ca048/Z3VpZCNzcGFjZV9hM2Y2YzgzZWEzNTc4MDdlNWVkNWUzZTg4MjFjYTA0OCNhM2Y2YzgzZWEzNTc4MDdlNWVkNWUzZTg4MjFjYTA0OA). Code for the application for the segmentation of Neuroblastoma tumours, see the details for accessing the files in section 3.3.
- Binary repositories (https://hub.docker.com/u/primage). Organization for the Docker images used for the execution of the containers.
- Documentation (https://gitlab.com/primageproject/documentation). A repository where documentation on how to use the platform is being collected.

3.2. Computing Back-end

The default computing back-end will be the EGI Cloud Compute. EGI Cloud Compute is an EOSC service[3] that comprises a wide set of federated IaaS (Infrastructure as a Service) resources. In the period June 2018-July 2019, EGI Cloud Compute served more than 26 million CPU hours.

EGI Cloud compute can be easily accessed through the EGI applications on demand service (https://marketplace.egi.eu/42-applications-on-demand-beta). How to access one data files are described in 3.3. EGI Applications on demand is an exploratory service for new communities that want to start using EGI Cloud Compute. This way, a user community can tune up and mature their services before scaling up to a production platform. The access to the resources is granted for free if registered in the EGI AAI (Authentication and Authorisation Infrastructure) system.

As the cloud orchestrator services are platform-agnostic, the applications can be deployed elsewhere requiring minimal changes. The final production back-end of the platform will be evaluated in the sustainability plan.

3.3. Storage Back-end

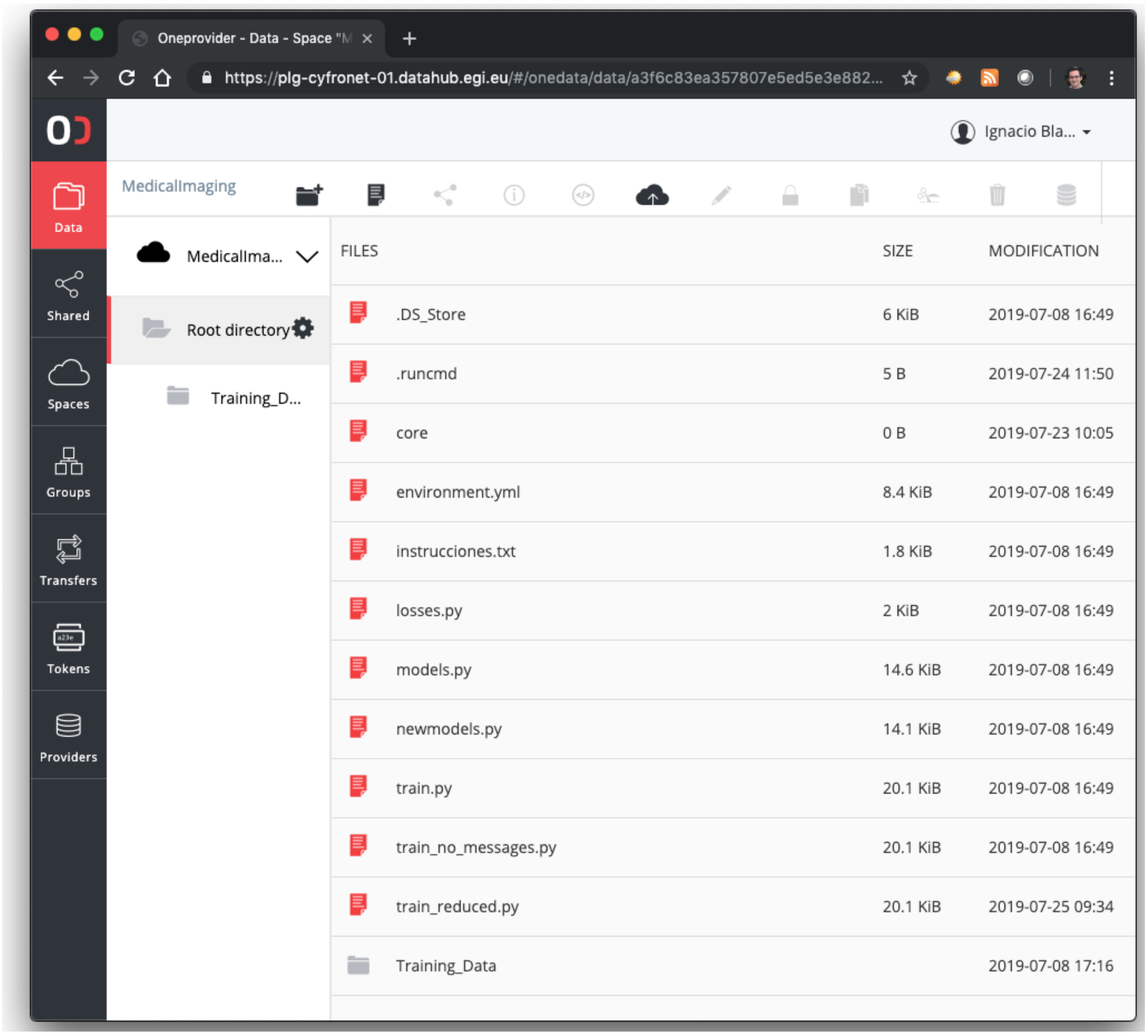
For the storage backend, the project selects using EGI Data Hub, an EOSC service listed in the EOSC marketplace, despite the deployment system is platform agnostic and it could be reproduced in other IaaS platform. EGI Data Hub is available in https://datahub.egi.eu/.

[3] https://marketplace.eosc-portal.eu/services/egi-cloud-compute

EGI DataHub[4], is a service designed to make data discoverable and available in an easy way across all EGI federated resources. It allows users to make their data available using different levels of access: from completely unrestricted open access to open data to authenticated access to closed data sets. This is possible as a result of the seamless integration with the EGI AAI service, also integrated in the PRIMAGE project.

The EGI DataHub is built on top of the EGI Open Data Platform using Onedata technology to connect a wide range of existing storage services, regardless of their underlying technology (e.g. Lustre, Amazon S3, Ceph, NFS, or dCache). More information on EGI Datahub can be found on *Matthew Viljoen, Łukasz Dutka, Bartosz Kryza, Yin Chen, "Towards European Open Science Commons: The EGI Open Data Platform and the EGI DataHub", Procedia Computer Science, Volume 97, 2016, Pages 148-152, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2016.08.294*[5].

The project has requested a shared space named "MedicalImaging",stored in the Onedata Provider plg-cyfronet-01.datahub.egi.eu. Access to the storage can be obtained by creating issues in the OneClient repository. Figure 9 shows a snapshot of such shared space.



FILES	SIZE	MODIFICATION
.DS_Store	6 KiB	2019-07-08 16:49
.runcmd	5 B	2019-07-24 11:50
core	0 B	2019-07-23 10:05
environment.yml	8.4 KiB	2019-07-08 16:49
instrucciones.txt	1.8 KiB	2019-07-08 16:49
losses.py	2 KiB	2019-07-08 16:49
models.py	14.6 KiB	2019-07-08 16:49
newmodels.py	14.1 KiB	2019-07-08 16:49
train.py	20.1 KiB	2019-07-08 16:49
train_no_messages.py	20.1 KiB	2019-07-08 16:49
train_reduced.py	20.1 KiB	2019-07-25 09:34
Training_Data		2019-07-08 17:16

Figure 9. *Snapshot of the EGI Datahub shared space.*

[4] https://www.egi.eu/about/newsletters/introducing-the-egi-datahub-prototype/
[5] http://www.sciencedirect.com/science/article/pii/S187705091632110X

4. Implementation of Sample applications

This section describes how the canonical applications described in deliverable D2.2 can be implemented.

4.1. Application 1. A High-Throughput Compute Batch job

High-Throughput Compute Batch job is a canonical example that can be used to build other complex applications, such as processes that are part of muti-scale models or quantitative imaging biomarkers workflows (e.g. segmentation processes).

Figure 10 shows the architecture of High-Throughput Compute Batch job. In the figure, an Application Developer (AD) has created an application for Image Segmentation. The application is coded into Python3 and requires specific software libraries. The code is available in a Github repository which is linked to the autobuild and testing system. The application is structured into three folders in the repo: Application Code, Application Dependencies and Unity tests. The AD commits a new change on the repository (1), which triggers the building and testing (2) of an application image, which is uploaded (3) into a private registry. Then, the application developer mounts a previously created persistent storage (4) on the local compute to explore the files(s) to process. Then, the AD submits a job (5) using a specific command from PRIMAGE platform and specifies the files to process, as well as the storage id. This result on a set of jobs submitted to a queue system (e.g. an elastic Kubernetes) on the PRIMAGE platform (6). The jobs mount the storage (7), pull the application container from the registry (8), run the job and produce the output on the same storage, which can be seen by the AD from his/her computer (9).

For this canonical example, the application will consist of a queue system for submitting jobs on a Kubernetes cluster. The application will launch independent jobs in which all of them will share and have accessible the same remote file system. In this case, the shared repository will contain the required executable, the input files to process (e.g. medical images) and also will save the results when the jobs end. The remote file system also can be accessible from the local computer. This way, the user can upload data in a shared repository to prepare the process, submit the jobs, and finally view the results from the console.

The steps for the deployment are:

1. Create a volume on a shared repository (see figure 9). For this purpose an EGIDatahub valid account, a provider (plg-cyfronet-01.datahub.egi.eu in our case), a data space (MedicalImaging in our case) and the access token to the volume will be needed.
2. Mount the shared volume from the local computer. For this purpose the modified Docker image of ONEClient in the PRIMAGE Project (`primage/oneclient:latest`) can be used:

   ```
   docker run -dit --privileged -e MOUNT_POINT=$MOUNT_POINT -e
   ONECLIENT_ACCESS_TOKEN=$ACCESS_TOKEN -e ONECLIENT_PROVIDER_HOST=$PROVIDER_HOST --name
   primage_oneclient primage/oneclient:latest bash
   ```

   Where `ONECLIENT_ACCESS_TOKEN` contains the access token, `ONECLIENT_PROVIDER_HOST` the provider and `MOUNT_POINT` the absolute path where the volume will be mounted in the container.

   Also, the web interface (see figure 9) can be used through a web browser and the EGIDatahub valid account.

3. Create a specialized Docker image if needed, to execute the jobs. In this example, the PRIMAGE customized oneclient image was extended with the support of required scripts to execute a segmentation process (`primage/segmentation_htc_job:latest`). The code of the Dockerfile and the

associated scripts is available in the public repository https://gitlab.com/primageproject/containers/tree/master/Segmentation_HTC_Job/Docker.

4. Deploy a self-managed Kubernetes cluster in EGI LTOS Portal (see figure 7). For this purpose,the same EGI Checkin credentials can be used. It can be deployed through the LTOS Portal at https://servproject.i3m.upv.es/ec3-ltos/index.php.
5. Access through ssh (Secure Shell) to the K8s cluster following the instructions given from the Dashboard.

```
Cluster name: PRIMAGE_PILOT
Frontend IP: XXX.XXX.XXX.XXX
Username: cloudadm
Secret key: XXX
```

6. Launch the HTC batch job by using the YAML file (batch_python.yaml) in the same GitLab repository https://gitlab.com/primageproject/containers/tree/master/Segmentation_HTC_Job/Docker.

```
sudo kubectl apply -f batch_python.yaml
```

7. Once the process has finished, the cluster can be deleted safely. Bear in mind that only the files under the mounted directory will be preserved.
8. Such files are still accessible from your local computer:

```
docker exec -it primage_oneclient ls -s /mnt/oneclient/MedicalImaging
```

Also, it is possible to access that files using the web interface.

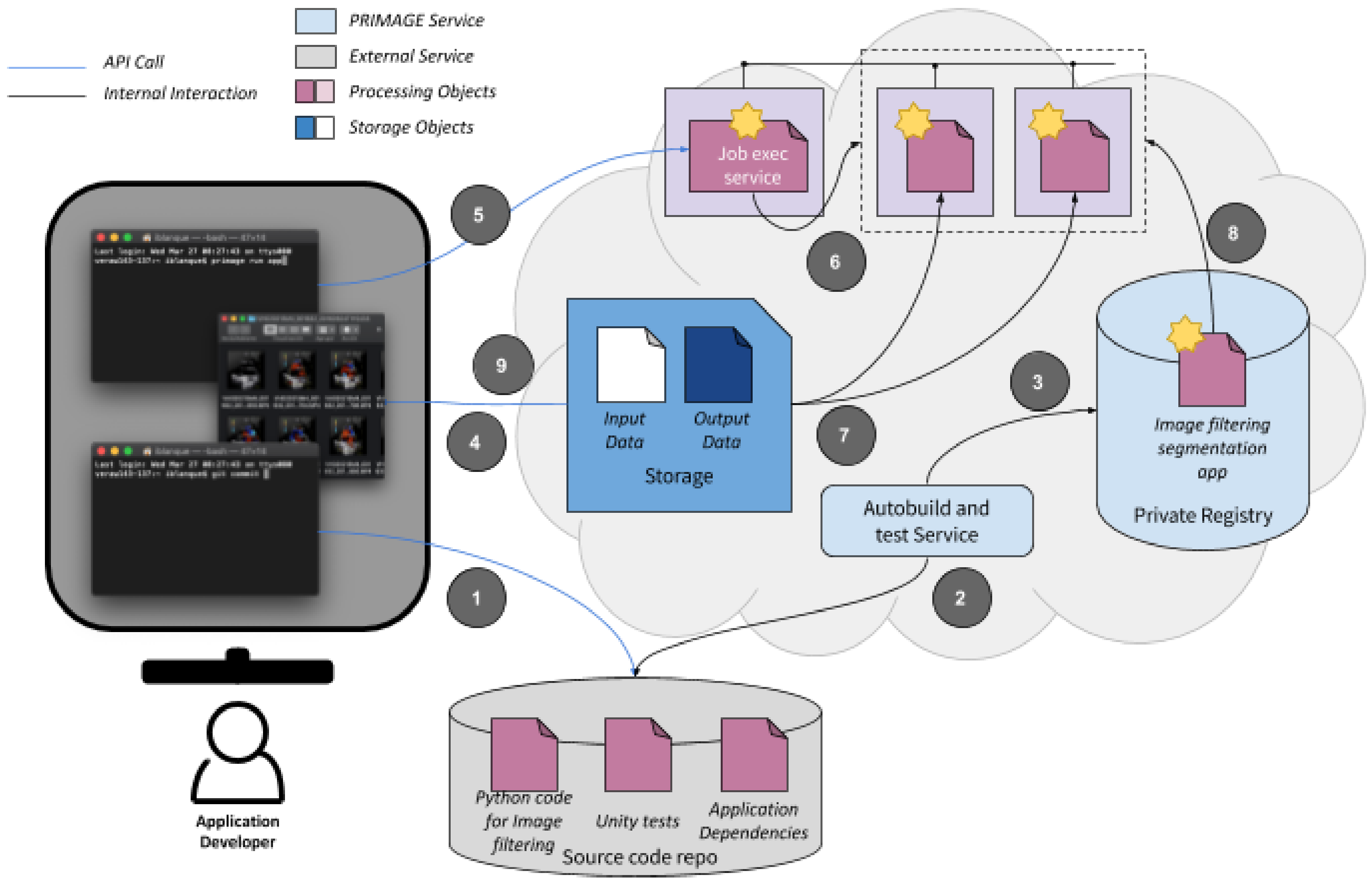


Figure 10. *Canonical application for the HTC Batch job architecture*.

4.2. Application 2. An interactive application.

Interactive applications are other canonical example that can be used to build more complex applications, such as the PRIMAGE Clinical Decision Support System (CDSS) Dashboard.

Figure 12 shows the architecture of High-Throughput Compute Batch job. In that figure it can be seen an Application Developer (AD) that has created a Jupyter application notebook for Image filtering. The application requires Jupyter and it also produces some graphics. The code is available in a GitLab repository which is linked to the autobuild and testing system. The application is structured into three folders in the repo: Application Code, Application Dependencies and Unity tests. The AD commits a new change on the repository (1), which triggers the building and testing (2) of an application image, which is uploaded (3) into a private registry. Then, the application developer runs the application through a PRIMAGE command (4) that runs the application as a service (5) mounting a previously created persistent storage (7) on the application and pulls the previously defined container (7). The user receives the URL (Uniform Resource Locator) to connect, and accesses the application through his/her computer (8).

For canonical example, the application will consist on a Jupyter Notebook deployed on a Kubernetes cluster, which mounts a remote filesystem, which is also accessible from the local computer. This way, the user can upload data, process it remotely and view the results from the Jupyter notebook.

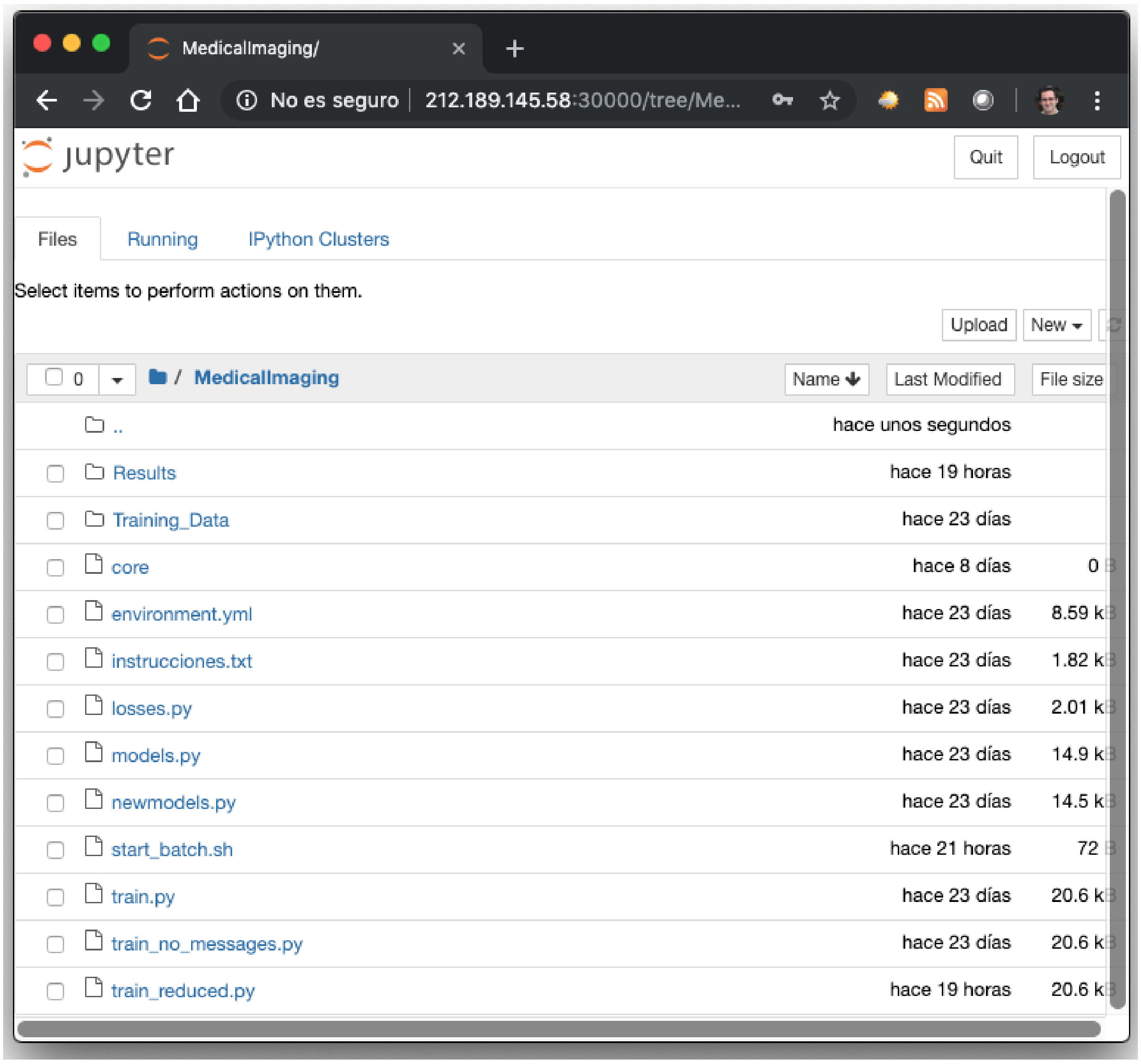
The steps for the deployment are:

1. Create a volume on a shared repository. For this purpose, an EGIDatahub valid account, a provider (plg-cyfronet-01.datahub.egi.eu in our case), a data space (MedicalImaging in our case) and the access token to the volume will be needed.
2. Mount the shared volume from the local computer. For this purpose the modified Docker image of ONEClient in the PRIMAGE Project (`primage/oneclient:latest`) can be used:

   ```
   docker run -dit --privileged -e MOUNT_POINT=$MOUNT_POINT -e
   ONECLIENT_ACCESS_TOKEN=$ACCESS_TOKEN -e ONECLIENT_PROVIDER_HOST=$PROVIDER_HOST --name
   primage_oneclient primage/oneclient:latest bash
   ```

   Where `ONECLIENT_ACCESS_TOKEN` contains the access token, `ONECLIENT_PROVIDER_HOST` the provider and `MOUNT_POINT` the absolute path where the volume will be mounted in the container.

9. Create an specialized Docker image if needed. In this case, the PRIMAGE customized oneclient image was extended with the support of ipyhton, ipyhton notebooks and Jupyter (`primage/jupyter:latest`). The code of the Dockerfile and the associated scripts is available in the public repository https://gitlab.com/primageproject/containers/tree/master/Segmentation/Docker.
10. Deploy a self-managed Kubernetes cluster in EGI LTOS Portal (see figure 7). For this purpose, the same EGI Checkin credentials can be used. It can be deployed through the LTOS Portal at https://servproject.i3m.upv.es/ec3-ltos/index.php.
11. Access the K8s cluster following the instructions given from the Dashboard, and deploy the Jupyter Notebook application by using the YAML file in the same GitLab repository https://gitlab.com/primageproject/containers/tree/master/Segmentation/Docker.
12. Access the Jupyter notebook front end and process the files (see figure 11).



MedicalImaging/

No es seguro | 212.189.145.58:30000/tree/Me...

jupyter Quit Logout

Files Running IPython Clusters

Select items to perform actions on them.

Upload New

0 / MedicalImaging

Name	Last Modified	File size
..	hace unos segundos	
Results	hace 19 horas	
Training_Data	hace 23 días	
core	hace 8 días	0 B
environment.yml	hace 23 días	8.59 kB
instrucciones.txt	hace 23 días	1.82 kB
losses.py	hace 23 días	2.01 kB
models.py	hace 23 días	14.9 kB
newmodels.py	hace 23 días	14.5 kB
start_batch.sh	hace 21 horas	72 B
train.py	hace 23 días	20.6 kB
train_no_messages.py	hace 23 días	20.6 kB
train_reduced.py	hace 19 horas	20.6 kB

Figure 11. *Jupyter front end.*

13. Once the process has finished, the cluster can be deleted safely. Bear in mind that only the files under the mounted directory will be preserved.
14. Such files are still accessible from your local computer (as well as on the web site):

```
docker exec -it primage_oneclient ls -s /mnt/oneclient/MedicalImaging
```

Also, it is possible to access that files using the web interface.

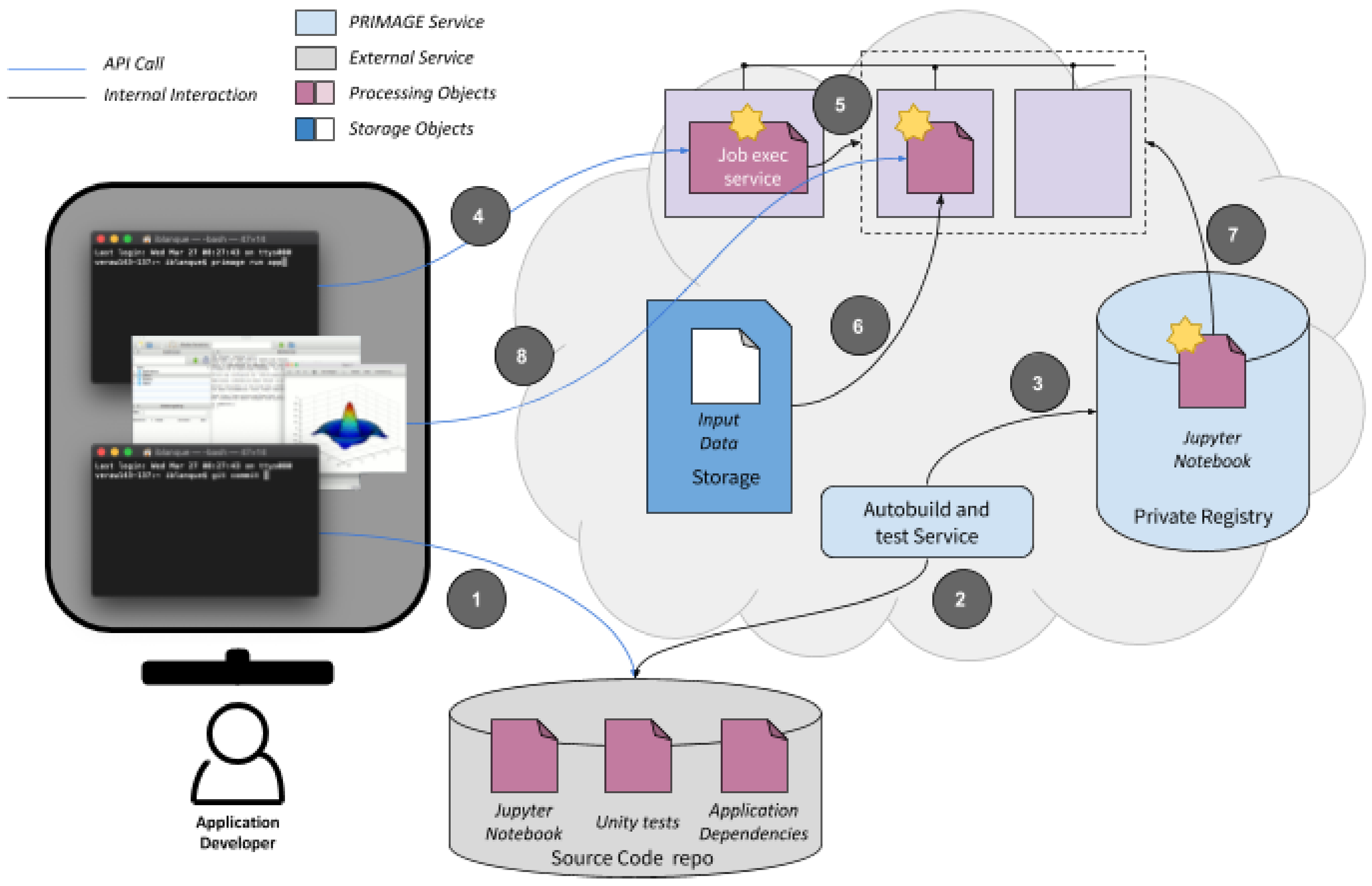


Figure 12. *Canonical application for the interactive application architecture.*

5. Conclusion

This document describes in practical terms the first version of the cloud platform defined for the project. This platform fulfils a first set of 21 requirements and it is deployed on a European funded Research Infrastructure from the EOSC initiative. The document contains URLs, procedures, command examples, snapshots and examples to develop applications using two canonical examples. The document will be extended with the support of the rest of requirements, as well as with additional services and compatibility, as the project evolves.

The use of EOSC services, such as EGI check-in, EGI Datahub and EGI Cloud Compute will be relevant for the sustainability of the project and will help on building the EOSC concept. As data and services are being developed, the project will consider releasing the access to such services and data, according to the privacy regulations imposed by the nature of the data.

6. Acronyms and Abbreviations

ACRONYM	DEFINITION
AA	Authentication and Authorisation
AAI	Authentication and Authorisation Infrastructure
AD	Application Developer
CDSS	Clinical Decision Support Systems
EC3	Elastic Compute Cluster in the Cloud
EDUGAIN	EDUcation Global Authentication INfrastructure
EGI	European Grid Infrastructure
EOSC	European Open Science Cloud
HPC	High Performance Computing
HTC	High-Throughput Computing
IaaS	Infrastructure as a Service
IdP	Identity Provider
K8s	Kubernetes
LTOS	Long Tail Of Science
OASIS	Organization for the Advancement of Structured Information Standards
ONEDATA	Global Data Access Solution for Science
POSIX	Portable Operating System Interface
PRIMAGE	PRedictive In-silico Multiscale Analytics to support cancer personalized diaGnosis and prognosis
RADL	Resource and Application Description Language
SSH	Secure Shell
SSO	Single Sign On
TOSCA	Topology and Orchestration Specification for Cloud Applications
URL	Uniform Resource Locator
YAML	Ain't Markup Language