

D5.12 - THIRD PORTAL RELEASE

Grant Agreement	676547
Project Acronym	CoeGSS
Project Title	Centre of Excellence for Global Systems Science
Topic	EINFRA-5-2015
Project website	http://www.coegss-project.eu
Start Date of project	October 1st, 2015
Duration	36 months
Due date	April 30 th , 2017
Dissemination level	Public
Nature	Report
Version	1.0
Work Package	WP5
Leading Partner	ATOS (F. Javier Nieto)
Authors	Burak Karaboğa, Yossandra Sandoval, Sergiy Gogolenko
Internal Reviewers	Andreas Geiges, Paweł Wolniewicz
Keywords	Portal, Tools, CoE Services
Total number of pages:	20

Copyright (c) 2016 Members of the CoeGSS Project.



The CoeGSS (“Centre of Excellence for Global Systems Science”) project is funded by the European Union. For more information on the project please see the website [http:// http://coegss-project.eu/](http://coegss-project.eu/)

The information contained in this document represents the views of the CoeGSS as of the date they are published. The CoeGSS does not guarantee that any information contained herein is error-free, or up to date.

THE CoeGSS MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, BY PUBLISHING THIS DOCUMENT.

Version History

Version	Name	Partner	Date
From	Burak Karaboga	ATOS	06.04.2018
Initial Template	Burak Karaboga	ATOS	10.04.2018
First Draft (ver. 0.3)	Burak Karaboga, Yosandra Sandoval, Sergiy Gogolenko	ATOS, HLRS	26.04.2018
First Version (ver. 0.4)	Burak Karaboga	ATOS	27.04.2018
Reviewed Version	Burak Karaboga, Javier Fco. Nieto	ATOS	27.04.2018
Approved by	Coordinator		21.06.2018

Abstract

The third release of the CoeGSS Portal introduces major changes on top of the previous version as well as adding new features and components. Following the document structure of the previous version, this document provides detailed information about the current state of the portal; its components, their deployment and configuration details. For the sake of coherence and readability, this document aims to highlight only the new features and changes to the previous version of the CoeGSS portal while keeping a summary of the text from the preceding deliverable and referring to it where necessary.

Abstract.....	3
1. Introduction	4
2. Implemented Portal Architecture	5
3. Frontend	7
4. Authentication & Authorization	11
5. HPC Job Submission	13
6. Summary.....	17
References	18
List of tables	20
List of figures.....	20
List of Abbreviations.....	20

1. Introduction

This aim of this document is to describe the state of the third release of the CoeGSS Portal by focusing only to the changes and updates to the previous releases [1] [2] and referring to them where necessary.

The document describes the CoeGSS Portal implementation, detailing several aspects about the implementation with the objective of facilitating the understanding about the implementation and acting as a guideline for those who may want to deploy the Portal components.

In order to do so, Section 3, describes the implemented features and remembers the followed architecture, while Sections 4 to 6 provide information about the implementation, configuration and the testing details of the components that are enhanced or introduced in this version. Finally, Section 7 summarizes the document and provides some conclusions.

The work towards the third release has been focused on the implementation and integration of the HPC Job Submission component and replacing the single sign-on provider Shibboleth to FIWARE IDM [3]. This release also introduces some changes to the Matchmaking sub-component of the Frontend.

2. Implemented Portal Architecture

2.1 Implemented Features

For the third portal release is comprised of the following changes, new features and enhancements:

- The Single-Sign-On (SSO) component consisting of Shibboleth and LDAP has been replaced by FIWARE IDM due to several compatibility issues and the high complexity of its maintenance, configuration and compatibility issues with the other CoeGSS components.
- HPC Job Submission component has been implemented allowing job submission to HPC systems.
- Matchmaking component has been improved in order to make smarter matches and offer a better user experience.
- A new virtual machine has been introduced to the infrastructure hosting the new SSO provider.

2.2 Implemented High Level Architecture

The third release of the CoeGSS Portal introduces some changes to the architecture described in the second release of the CoeGSS Portal [2] (see: Figure 1) by replacing Shibboleth and LDAP with FIWARE IDM as the SSO provider and adding an additional component to take care of the HPC job submission.

Although introducing a brand new component and some changes to the existing components, the third release does not alter the architecture of the CoeGSS Portal in any major way. The updated architecture introduces a new component called HPC Job Submission and replaces the SSO component with FIWARE IDM.

The second release of the CoeGSS Portal had introduced Shibboleth 2.0 relying on LDAP as the authentication and authorization provider. Due to its maintenance, configuration and integration complexity as well as the compatibility issues between Shibboleth and the HPC Job Submission component, this version introduces FIWARE IDM as the new SSO provider. The motivation behind this replacement and the implementation details are documented thoroughly in section 5.

All the tools and code developed have been uploaded to the internal CoeGSS Git repository (hosted at HLRS under the R1 master branch) so they are available to the CoeGSS consortium. This repository is private but, once the Portal implementation is stable, its code will be made

public. For further details about the implementation of any of the mentioned components, please refer to the next sections.

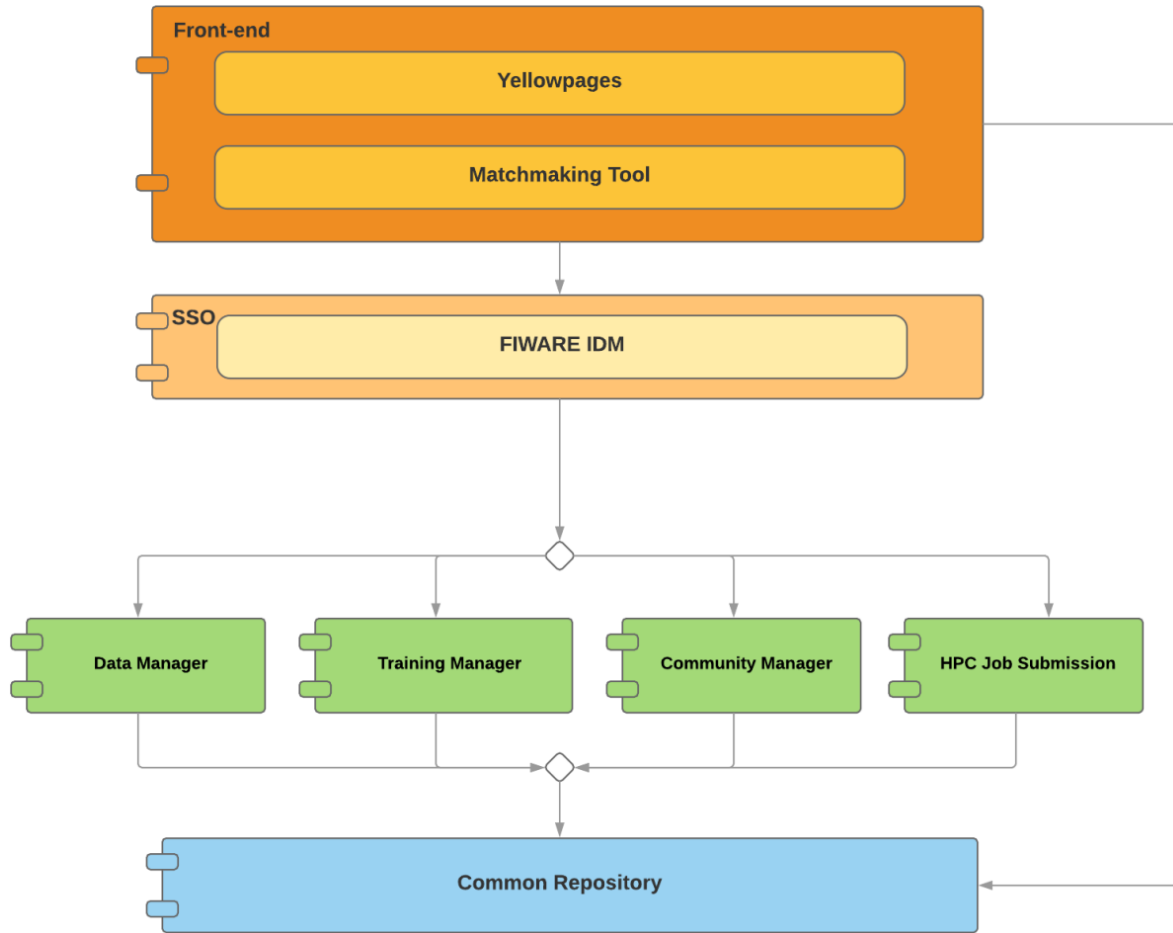


Figure 1. CoeGSS Portal High Level Architecture

3. Frontend

3.1 Component Description

This section describes the Frontend component, which provides a single point of access for all other components that are implemented in the context of CoeGSS.

The current release of the portal introduces changes to the configuration and deployment of the component as well as some improvements to the Matchmaking sub-component.

3.2 Sub-Components

3.2.1 Matchmaking Tool

The current release introduces a new version of the Matchmaking Tool which has an enhanced matchmaking algorithm and updated functionality compared to the previous release [2]. In order to achieve a more understandable description of the tool, the user interaction is described in detail.

The match algorithm implemented for the third release of the portal allows to understand the user (user_a) as well other users (user_b) to build matches around the topics that they may like in common. The tool asks questions to each side of users (the logged in user and any others user registered on the portal) related with the CoeGSS project.

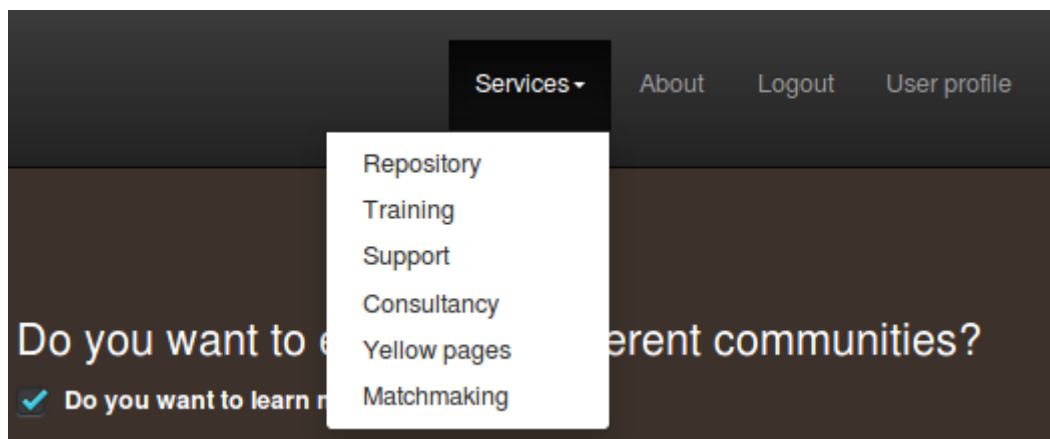


Figure 2. Menu option of Matchmaking Tool

After entering username and password via login page the user is allowed to access the option Matchmaking on the Services menu as presented in Figure 2. When the user chooses the option Matchmaking, the web interface of the tool is shown, see Figure 3. There, the tool will ask random questions about the user that will produce a match percentage with others user, according to the answers provided. Those matches will be presented on the right side of the interface.

Together with the answers to the questions asked by the tool (Your Answer), the user has to provide the ideal answers they would prefer from other users (Their answer). The user has to include a level of importance for each type of answer. The importance levels are: Mandatory, Very Important, Somewhat Important, Not Important.

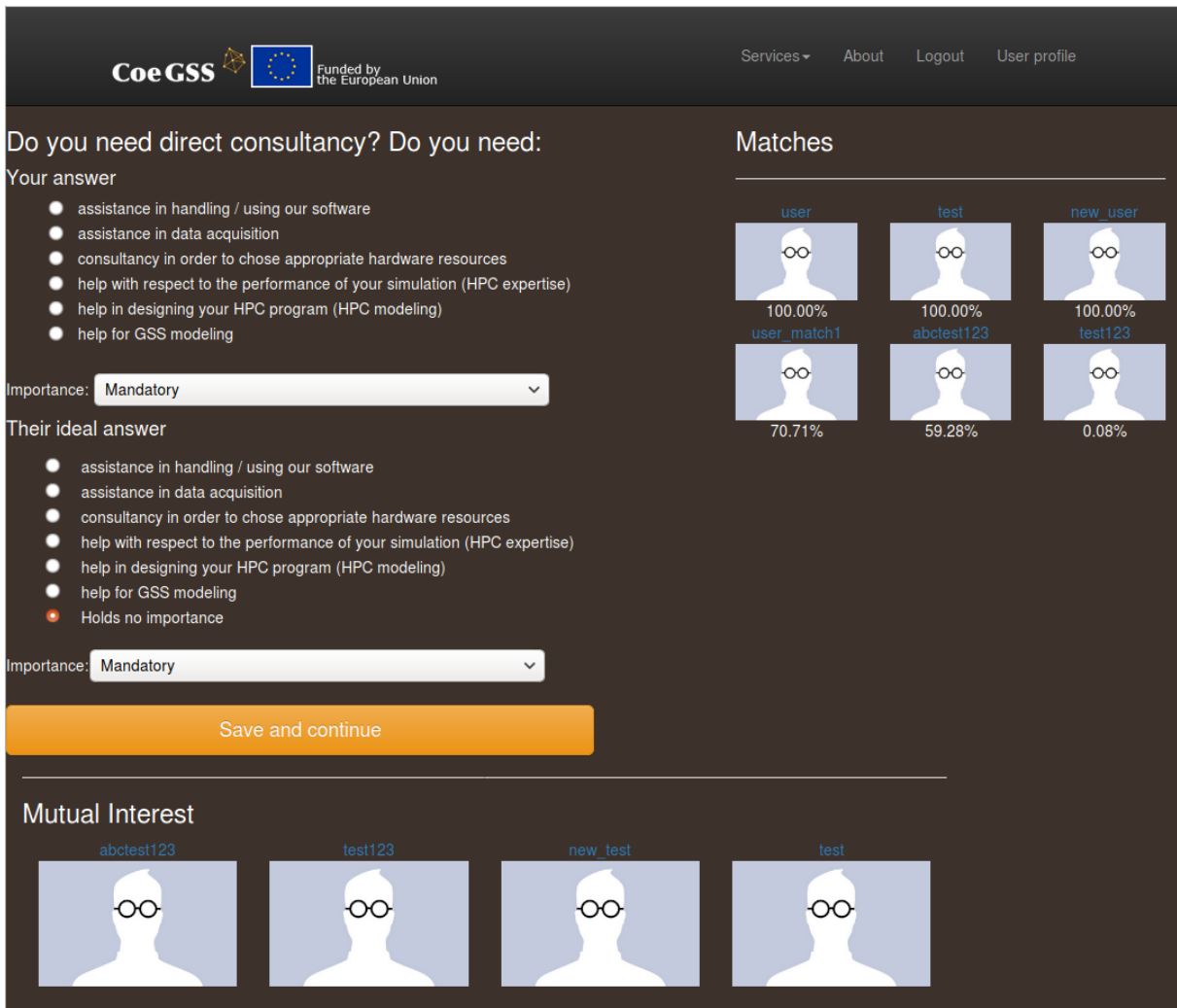


Figure 3. Matchmaking user interface

If the user clicks on one of the users listed on the matches section, a new interface will be presented with the username and match percentage as shown on Figure 4. There he/she can mark "like" for this particular user. Then the user gets more information about the profile for which he/she marked "like", see Figure 5. If the marked user also likes the logged user, both will be connected and listed on the mutual interest section (bottom site - Figure 3).

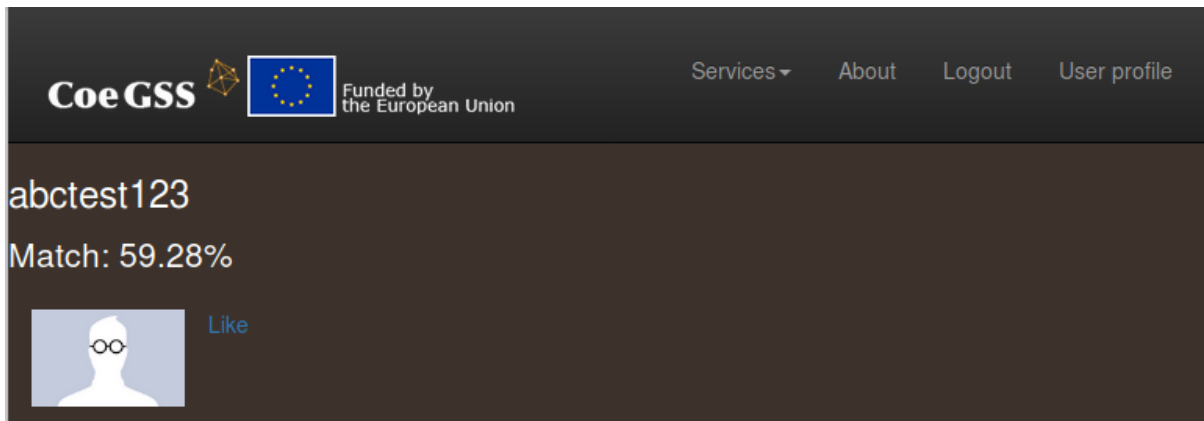


Figure 4. Match user - like option

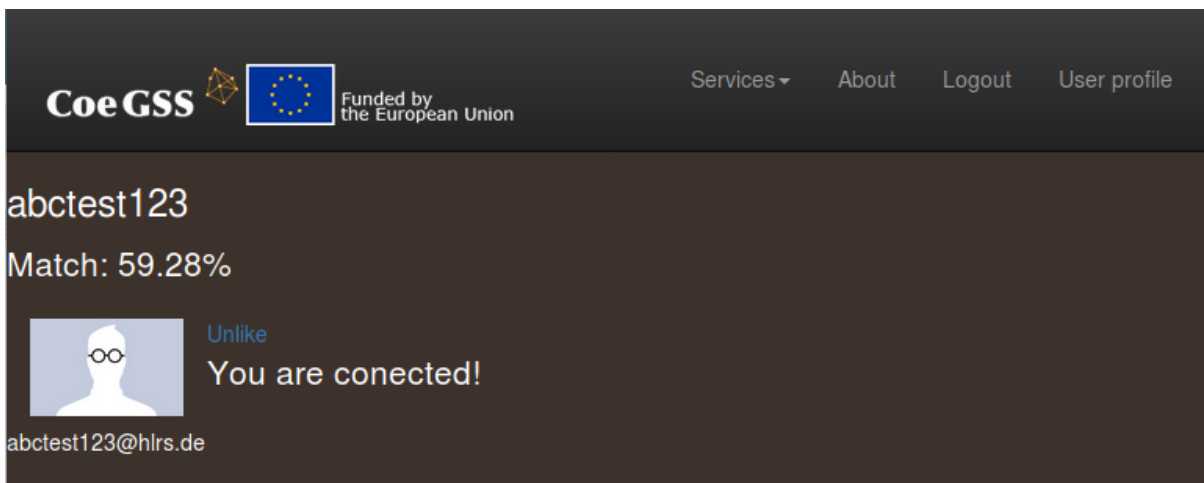


Figure 5. Match user, more detail information - unlike option

3.3 Component Configuration

The Frontend component's main configuration point is the *settings.py* file, which can be found under the relative path *coegss_portal/coegss_portal/settings.py*. Through this file, the path for the static and template files can be modified, authentication backends as well as FIWARE IDM configuration can be managed, Database connection can be configured and other Django apps can be added to the web application.

The current version of the CoeGSS portal introduces a new configuration file, *settings.ini* which can be found under the relative path *coegss_portal/coegss_portal/settings.ini*. The sensitive information in *settings.py* file is replaced with references to the variables defined in the *settings.ini* file which add another layer of security to the Frontend application.

The current settings for the configuration points listed above in *settings.py* are listed as follows:

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

```
INSTALLED_APPS = [
    'sso.apps.SsoConfig',
    'registration',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'frontend',
    'matchmaking',
    'tags_input',
    'social_django',
]
```

```
AUTHENTICATION_BACKENDS = [
    'sso.backends.keyrock.KeyrockOAuth2',
    'django.contrib.auth.backends.ModelBackend',
]
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'coegss_portal',
        'USER': 'portal',
        'PASSWORD': 'DB_PASS',
        'HOST': 'localhost',
    }
}
```

```
FIWARE_IDM_ENDPOINT = config('FIWARE_IDM_ENDPOINT')
SOCIAL_AUTH_FIWARE_KEY = config('SOCIAL_AUTH_FIWARE_KEY')
SOCIAL_AUTH_FIWARE_SECRET = config('SOCIAL_AUTH_FIWARE_SECRET')
```

3.4 Component Deployment

The current version of the Frontend component is deployed in a virtual machine running Ubuntu 14.04 and is hosted at the High Performance Computing Centre Stuttgart. The service can be accessed via [https:// portal.coegss.hls.de](https://portal.coegss.hls.de)

3.5 Component Testing

As the previous version [2], the current version of the component has been tested manually. Creation of automated testing mechanisms mentioned in the previous version is still in progress and is planned to be completed within the project lifespan.

4. Authentication & Authorization

Within this section of the deliverable, the CoeGSS authentication and authorization mechanism, which is now handled by a deployment of FIWARE IDM, is described. This SSO mechanism represents a key component of the entire CoeGSS Portal architecture since it is used for user management including authentication and authorization, providing a single sign-on for all CoeGSS services except the Training Manager.

4.1 Component Description

In the previous release, this component was composed of a Shibboleth 2.0 deployment which relied on an existing LDAP deployment for authentication. Although this solution proved useful for providing a single sign-on to access the services provided by the CoeGSS portal, its configuration, maintenance and integration complexity was very high, which made it very costly to introduce new components to the architecture. Besides these, we had several compatibility issues between Shibboleth and the HPC Job Submission component which was introduced in this release.

The current release of the CoeGSS portal solves these issues by replacing Shibboleth and LDAP deployments by an SSO solution from FIWARE [4]. FIWARE provides cloud hosting services based on OpenStack technology and a set of components offering a number of added-value functions “as a service”, also known as the Generic Enablers [5]. The Generic Enabler providing the SSO functionality is called FIWARE IDM which proved to be featuring an easier configuration and a smoother integration process.

Basically, FIWARE IDM consists of two components: Keyrock, an extension of Openstack Keystone, and Openstack Horizon.

- Keyrock is responsible for handling authentication and authorization requests and providing all aspects of SSO mechanism in general.
- Horizon is responsible for providing a dashboard for the portal administration to manage trusted applications in the system as well as login, registration and application authorization forms to the end user.

The new authentication flow based on the FIWARE IDM deployment is as follows:

1. User access a restricted resource and is redirected to FIWARE IDM
 - a. If the user has not logged in yet, they are asked for their login credentials
 - b. After a successful login, FIWARE IDM creates a session, creates an authorization token for the user and redirects them back to CoeGSS portal
2. CoeGSS Portal validates the token, creates a session for the user and sends them to the requested resource.

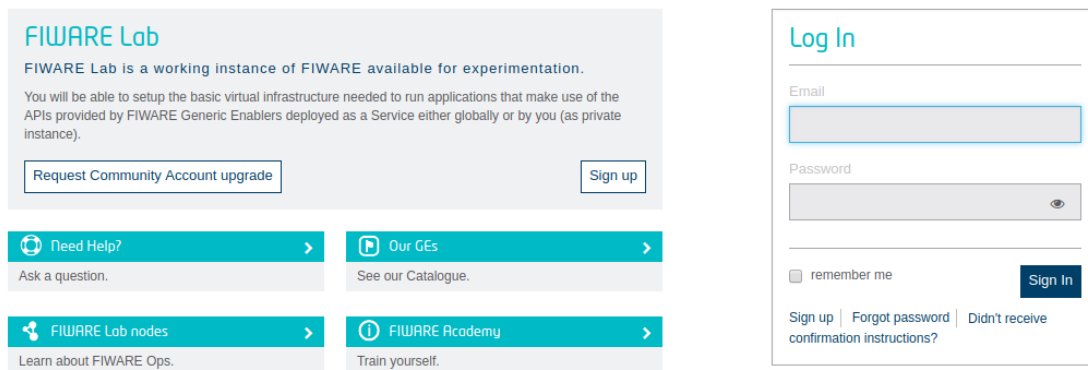


Figure 6. FIWARE IDM login page

4.2 Component Configuration

The component itself is being used as it is without changing any of its configuration details although it requires some configuration on the CoeGSS Portal side to work properly. Although the configuration is hosted by the Portal application, it is detailed here since it is directly related with FIWARE IDM.

The configuration values related to FIWARE IDM can be found in the *settings.ini* file of the Frontend component which is described in detail in Section 4. These values are listed below:

```
FIWARE_IDM_ENDPOINT = config('FIWARE_IDM_ENDPOINT')
SOCIAL_AUTH_FIWARE_KEY = config('SOCIAL_AUTH_FIWARE_KEY')
SOCIAL_AUTH_FIWARE_SECRET = config('SOCIAL_AUTH_FIWARE_SECRET')
```

4.3 Component Deployment

FIWARE IDM is hosted by its own virtual machine at the High Performance Computing Centre Stuttgart. The virtual machine hosting FIWARE IDM is based on a standard Ubuntu 16.04 installation which is extended to provide the ability to store Secure Shell (SSH) public keys. The component is publicly reachable via URL <http://idm.coegss.hlr.de>.

5. HPC Job Submission

This section describes the HPC Job Submission component, which provides the necessary functionality allowing the users to submit jobs to HPC systems through the CoeGSS portal.

5.1 Component Description

The third release of the CoeGSS Portal introduces the HPC Job Submission as a new component which is responsible for providing the functionality and the user interface to allow job submissions to the HPC systems.

The component is mainly based on a deployment of Cloudify, an open source cloud orchestrator that allows execution of workflows and operations [6] [7]. Through plugins, Cloudify can be easily extended to support other tools and infrastructures, as well as to implement new workflows that represent different behaviours to perform over the application [7].

HPC Job Submission component uses an extension implemented in the context of the MSO4SC [8] project enabling HPC job submissions. Such extension extends the TOSCA DSL and allows sending jobs to SLURM and retrieving some monitoring metrics. CoeGSS enhanced this plugin in order to make it compatible with TORQUE as well. Thanks to this solution, it is possible to implement the CoeGSS simulation workflows in an easy way using TOSCA, a standardized language also supported by graphical tools. All the complexity of interacting with the workload managers is hidden, so stakeholders only need to focus on preparing their TOSCA files.

5.1.1 General Architecture

HPC plugin enables Cloudify to manage HPC resources in one or more infrastructures. The core of the plugin is written in OASIS TOSCA 1.2 and Python 2.7. Python is used to implement behavioural part, whereas TOSCA files specify descriptive part [9]

plugin.yaml is the main TOSCA file in the plugin that defines types of nodes and relationships between them. It introduces two types of nodes – *hpc.nodes.Compute* derived from *cloudify.nodes.Compute* and *hpc.nodes.job* derived from *cloudify.nodes.Root*. The former addresses description of target HPC platforms and includes specifications for lifecycle (*cloudify.interfaces.lifecycle*) and external job monitor (*cloudify.interfaces.monitoring*) interfaces, while the latter is aimed to define jobs for execution on the HPC resources. Similarly, *plugin.yaml* defines two types of relationships – *job_contained_in_hpc* derived from *cloudify.relationships.contained_in* and *job_depends_on* derived from *cloudify.relationships.depends_on*. The first one serves for assigning jobs to target HPC systems, the second one allows specifying inter-dependencies between jobs.

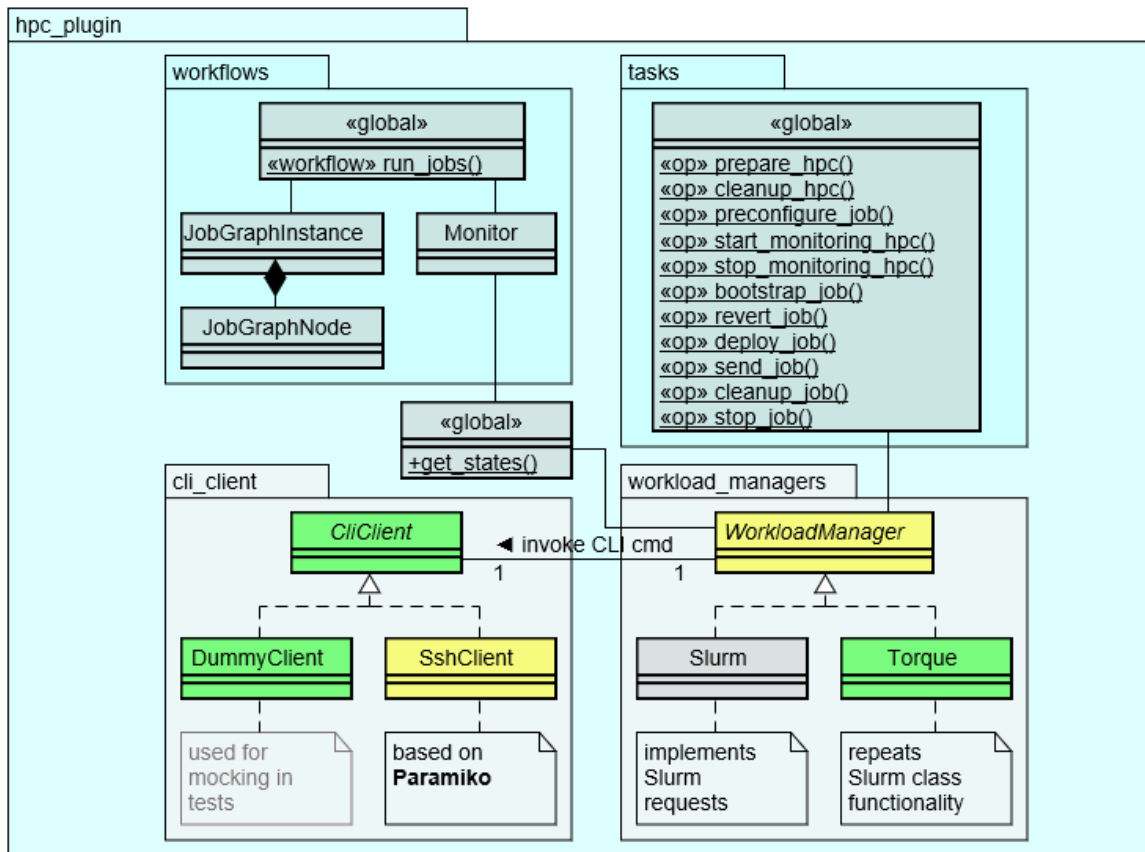


Figure 7. Class diagram of the Python codes from HPC plugin for Cloudify

Figure 6 illustrates class diagram of the Python codes from HPC plugin for Cloudify. The main *hpc_plugin* module consists of four submodules. Submodule *workload_manager* contains classes for different resource managers that provide a single interface for interacting with workload managers installed on HPC systems. This interface comprises methods for submitting jobs (*submit_job*), stopping jobs (*stop_job*), monitoring job states (*get_states*), and removing auxiliary files produced by jobs (*clean_job_aux_files*). In current version, submodule *workload_manager* incorporates *WorkloadManager* interface implementations for Slurm and Moab/TORQUE. In both cases, these implementations use the command line interface (CLI) of the corresponding resource manager. In order to invoke CLI commands, the plugin contains submodule *cli_client* which aggregates classes responsible for execution of CLI commands in different ways. In particular, class *SshClient* establish SSH connection to the remote HPC system in order to launch CLI commands remotely, class *DummyClient* simply ignores input commands which is useful for mocking in unit tests. Besides *workload_manager* and *cli_client*, HPC plugin contains submodules *workflow*, which accounts for Cloudify workflows, and *operations*, which collects implementations for Cloudify operations (overall 15 basic operations). Both submodules interact with resource managers on target HPC platforms via

classes from submodule *workload_manager*. The workflow requires such interaction for monitoring purposes, whereas operations need it for managing jobs on HPC systems.

5.1.2 Moab/TORQUE Extension

The core of HPC plugin was designed and implemented in the frame of MSO4SC project [8]. Basic MSO4SC version of the plugin supported only systems with Slurm resource managers. On the one hand, all major HPC platforms deployed in PSNC use Slurm as a default workload manager. On the other hand, TORQUE is used as a resource manager on HPC platforms deployed in HLRS. In order to bridge this gap, we extended HPC plugin with Moab/TORQUE support. It required from us significant changes in the submodules *workload_manager* and *cli_client*. These changes are reflected on Figure 7 where we marked our new classes with green and classes substantially modified by us with yellow.

The major improvement is related to introducing the class *Torque* which implements interface *WorkloadManager* by means of calling Moab/TORQUE CLI commands [10]. Job cancellation is fulfilled via *qdel* calls while job submission is performed via *qsub -V* calls. Query of job states is done with two TORQUE calls. At first, we determine job ids by their names with *qselect -N*. Next, if the list of job ids is not empty, we obtain detailed monitoring information about the jobs by calling *qstat -f*. Further, this information is parsed to identify job statuses. The decision about job status is taken based on the values of fields '*job_state*' and '*exit_status*'. Note that examination of '*job_state*' is insufficient since TORQUE states do not convey information whether job failed or finished successfully. The latter information can be obtained only from the field '*exit_status*'.

In order to facilitate unit testing, we introduced the class *DummyClient* and the interface *ICliClient*. Interface *ICliClient* decouples executor of CLI commands from the workload manager. Class *DummyClient* is specifically designed for testing and serves as mock object for CLI executors. Apart from it, we extended the functionality of the class *SshClient* with the ability to launch commands over SSH in a login shell. This change allows running commands remotely within user's environment [11].

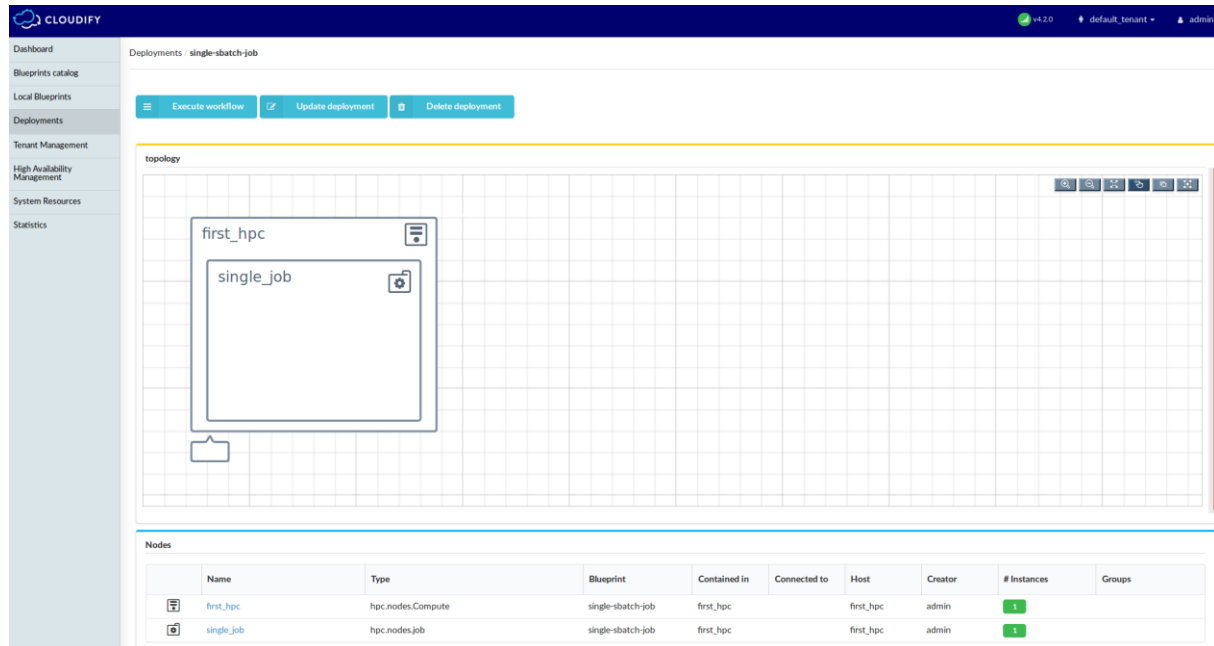
5.1.3 Usage Via Cloudfy's CLI

Process of running HPC jobs with Cloudfy HPC plugin comprises several steps. At first, the user must prepare blueprint. Each blueprint contains a mandatory TOSCA file which describes application topology. This file conveys information about the application's components, their interrelationships, as well as instructions for their installation, configuration, monitoring, and clean up. In addition to the TOSCA file, the blueprint may include auxiliary scripts (e.g., for bootstrapping and reverting the application), as well as additional data files. Next, the

blueprint must be uploaded and deployed on the Cloudify Manager through the CoeGSS portal. Both steps can be performed in Cloudify's CLI as follows:

```
cfy blueprints upload -b blueprint-id blueprint.yaml
```

```
cfy deployments create -b deployment-id -i ../local-blueprint-inputs.yaml -  
-skip-plugins-validation blueprint-id
```



Name	Type	Blueprint	Contained in	Connected to	Host	Creator	# Instances	Groups
first_hpc	hpc.nodes.Compute	single-batch-job	first_hpc		first_hpc	admin	1	
single_job	hpc.nodes.job	single-batch-job	first_hpc		first_hpc	admin	1	

Figure 8. Viewing blueprint deployment topology in Cloudify Manager Web GUI

HPC plugin does not demand manual installation of the plugin on the Cloudify Manager. It is automatically downloaded and installed from github repository during deployment of the blueprints. Note also that the deployment step allows transferring additional inputs to the blueprint. In the current version of the HPC plugin, this option is applied to notify the blueprint about credentials of available HPC systems and about external monitor if the latter is used for tracking job states.

Once the blueprint is deployed, the user can bootstrap and run jobs on the target HPC systems. It can be done via Cloudify's CLI:

```
cfy executions start -d deployment-id install
```

```
cfy executions start -d deployment-id run_jobs
```

As soon as the application is unnecessary, it can be uninstalled and blueprint can be removed from Cloudify Manager:

```
cfy executions start -d deployment-id uninstall
```


cfy deployments delete deployment-id
 cfy blueprints delete blueprint-id

Even though HPC plugin is a fully functional application, there are several directions for its improvement. Some of them are: support safer ways for storing credentials in blueprints via secret storage, implement wider range of job submission options, use template engines (e.g., Jinja2) for generating job scripts.

Events/Logs












Timestamp	Blueprint	Deployment	Workflow	Operation	Node Name	Node Id	Message
06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install				'install' workflow execution succeeded
06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.lifecyle.start	single_job	single_job_fogtu4	Task succeeded 'hpc_plugin.tasks.bootstrap_job'
 06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.lifecyle.start	single_job	single_job_fogtu4	.job bootstrapped
 06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install		single_job	single_job_fogtu4	Starting node
06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.lifecyle.start	single_job	single_job_fogtu4	Task started 'hpc_plugin.tasks.bootstrap_job'
06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.lifecyle.start	single_job	single_job_fogtu4	Sending task 'hpc_plugin.tasks.bootstrap_job'
 06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.lifecyle.start	single_job	single_job_fogtu4	Bootstrapping job..
 06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install		single_job	single_job_fogtu4	Configuring node
06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.relationship_lifecycle.preconfigure			Task succeeded 'hpc_plugin.tasks.preconfigure_job'
 06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.relationship_lifecycle.preconfigure			Preconfiguring HPC job..
 06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install		single_job	single_job_fogtu4	Creating node
06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.relationship_lifecycle.preconfigure			Task started 'hpc_plugin.tasks.preconfigure_job'
06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.relationship_lifecycle.preconfigure			Sending task 'hpc_plugin.tasks.preconfigure_job'
06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.monitoring.start	first_hpc	first_hpc_1ekgh	Task succeeded 'hpc_plugin.tasks.start_monitoring_hpc'
06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.monitoring.start	first_hpc	first_hpc_1ekgh	Sending task 'hpc_plugin.tasks.start_monitoring_hpc'
06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.monitoring.start	first_hpc	first_hpc_1ekgh	Task started 'hpc_plugin.tasks.start_monitoring_hpc'
 06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.lifecyle.start	first_hpc	first_hpc_1ekgh	.HPC ready on \$HOME/single_sbatch_20180306_145350
06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.lifecyle.start	first_hpc	first_hpc_1ekgh	Task succeeded 'hpc_plugin.tasks.prepare_hpc'
 06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.lifecyle.start	first_hpc	first_hpc_1ekgh	- manager: TORQUE
 06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.lifecyle.start	first_hpc	first_hpc_1ekgh	- remote host: hpcgogol@cl3fr2.hww.de, with remote login..
 06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install	cloudify.interfaces.lifecyle.start	first_hpc	first_hpc_1ekgh	Connecting to login node:
 06-03-2018 15:53	single-sbatch-job	single-sbatch-job	install		first_hpc	first_hpc_1ekgh	Starting node

Figure 9. Logs for install execution launched on HPC platform with TORQUE in Cloudify Manager Web GUI

5.2 Component Deployment

The current version of the component is deployed in a virtual machine running Ubuntu 14.04 and is hosted at the High Performance Computing Centre Stuttgart.

6. Summary

This document has described the third release of the CoeGSS Portal, providing information about the new and updated components deployed, their configurations and the testing performed, highlighting the changes and referring to the previous version of the deliverable where necessary. Therefore, not all functionality is described here as they were already documented in [2].

Although the aimed core functionality of the CoeGSS portal has been achieved with this version of the release, some minor improvements over the existing components such as visual customization of the FIWARE IDM tool and a better user experience would be beneficial.

References

- [1] C. Consortium, *First Portal Release*, 2016.
- [2] C. Consortium, *Second Portal Release*, 2017.
- [3] «FIWARE Catalogue: Identity Management,» [En línea]. Available: <https://catalogue.fiware.org/enablers/identity-management-keyrock>.
- [4] «FIWARE,» [En línea]. Available: <https://www.fiware.org/>. [Último acceso: 2018].
- [5] «What is FIWARE,» 2018. [En línea]. Available: <http://luisenlabs.com/what-is-fiware-and-how-your-startup-can-take-advantage-of-its-open-apis-platform-and-e-150k-free-grant/>.
- [6] "Cloudify," [Online]. Available: <https://en.wikipedia.org/wiki/Cloudify>.
- [7] "MSO4SC: D3.1 Detailed Specifications for the Infrastructure, Cloud Management and MSO Portal,," [Online]. Available: <http://book.mso4sc.cemosis.fr/deliverables/0.1/d3.1/README/>.
- [8] S. a. O. f. S. C. w. S. C. MSO4SC project: Mathematical Modelling. [Online]. Available: <http://mso4sc.eu>.
- [9] OASIS TOSCA Simple Profile in YAML Version 1.2: Committee Specification, 354 p., 2017.

[10] "Adaptive Computing Enterprises, Inc. A.1. Commands overview. In TORQUE 4.0.2 Administrator Guide, pp.133-206, 2012," [Online]. Available: <http://docs.adaptivecomputing.com/torque/4-0-2/torqueAdminGuide-4.0.2.pdf>.

[11] "Run remote SSH command with Full Login Shell," [Online]. Available: <https://superuser.com/questions/306530/run-remote-ssh-command-with-full-login-shell>.

List of tables

No table of figures entries found.

List of figures

Figure 1. CoeGSS Portal High Level Architecture	6
Figure 2. Menu option of Matchmaking Tool	7
Figure 3. Matchmaking user interface	8
Figure 4. Match user - like option	9
Figure 5. Match user, more detail information - unlike option	9
Figure 6. FIWARE IDM login page.....	12
Figure 7. Class diagram of the Python codes from HPC plugin for Cloudify	14
Figure 8. Viewing blueprint deployment topology in Cloudify Manager Web GUI	16
Figure 9. Logs for install execution launched on HPC platform with TORQUE in Cloudify Manager Web GUI.....	17

List of Abbreviations

DoW	Description of Work
EC	European Commission
CoeGSS	Centre of Excellence for Global System Science
GUI	Graphical User Interface
HPC	High Performance Computing
HPC-aaS	High Performance Computing as a Service
HPDA	High Performance Data Analysis
HTTP	Hypertext Transfer Protocol
IdP	Identity Provider
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
Q&A	Questions and Answers
SEO	Search Engine Optimization
SP	Service Provider
SSO	Single Sign-On
URL	Uniform Resource Locator
VM	Virtual Machine
WP	Work Package