D5.6 Second HIDALGO Portal Release and System Operation Report

Document Identification			
Status	Final	**Due Date**	30/09/2020
Version	1.2	**Submission Date**	16/04/2021

Related WP	WP5	**Document Reference**	D5.6
Related Deliverable(s)	D5.1, D5.2, D5.3, D5.7	**Dissemination Level (*)**	PU
Lead Participant	ATOS	**Lead Author**	F. Javier Nieto (ATOS)
Contributors	USTUTT, PSNC, ECMWF, ICCS, KNOW	**Reviewers**	Tamás Tomaschek (MK)
			Marcin Plociennik (PSNC)

Keywords:
HPCaaS, HIDALGO, CoE, Portal, Web, Entrypoint, CI/CD infrastructure, SSO, workflows, training, data management, visualization, support tools, interactive notebooks

Document Information

List of Contributors	
Name	Partner
F. Javier Nieto	ATOS
Raúl Santos	ATOS
Dineshkumar Rajagopal	USTUTT
Anna Mack	USTUTT
Krzesimir Samborski	PSNC
Marcin Lawenda	PSNC
Claudio Iacopino	ECMWF
Milana Vuckovic	ECMWF
Konstantinos Nikas	ICCS
Manuela Rauch	KNOW

Document History			
Version	Date	Change editors	Changes
0.1	22/07/2020	F. Javier Nieto	TOC.
0.2	07/08/2020	F. Javier Nieto	Updated ToC, introduction
0.3	15/08/2020	K. Samborski, D. Rajagopal	Contributions to section 3, 5.1.1, 6, 11, 12, 13, 13.2
0.4	17/08/2020	C. Iacopino	Contributions to section 5.1
0.5	20/08/2020	A. Mack, M. Rauch	Contributions to section 8
0.6	10/09/2020	K. Samborski, D. Rajagopal, F. J. Nieto, K. Nikas, M. Vuckovic	Updates in section 3, contributions to section 4, 9, 10 and 12. Update of section 5
0.7	29/09/2020	M. Lawenda	Section 7
0.8	14/12/2020	F. J. Nieto, R. Santos	Updates in section 11
1.0	15/01/2021	F. J. Nieto	Several modifications (section 2, 5, 11)
1.1	16/03/2021	D. Rajagopal, K. Nikas	Address internal review comments (section 6 and 10)
1.2	06/04/2021	F. J. Nieto	Complete modifications for addressing internal review comments (section 2 to 11)

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	F. Javier Nieto (ATOS)	15/04/2021
Quality manager	Marcin Lawenda (PSNC)	15/04/2021
Project Coordinator	Francisco Javier Nieto (ATOS)	16/04/2021

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
API	Advanced Programming Interface
A&A	Authentication and Authorization
CAS	Central Authentication Service
CI/CD	Continuous Integration / Continuous Deployment
DC	Data Catalogue
DMS	Data Management System
DRF	Django-Rest-Framework
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
FAQ	Frequently Asked Questions
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HCMS	Hybrid Multi-Cloud Management System
IdAM	Intelligent Digital Asset Management
MooCs	Massive Open Online Courses
MSX	Project Milestone X
MVP	Minimum viable product
OIDC	OpenID Connect
Q&A	Questions and Answers
REST	Representational State Transfer
SAML	Security Assertion Markup Language
SCM	Source Control Management
SEO	Search Engine Optimization
SPA	Single-Page Applications
SSO	Single Sign On
VM	Virtual Machine
WCAG	Web Content Accessibility Guidelines
WP	Work Package

Table 1 List of acronyms

Executive Summary

This deliverable presents the second version of the HiDALGO Portal. First of all, it presents the main features and the current architecture, highlighting the changes with respect to the first version, which are focused on notebooks, a new frontend, the ticketing system, documentation and some additional features in tools like the data management and the orchestrator. Then, it presents the current implementation of the features. Such implementation is focused on describing what was implemented (including some details about the tools used), the APIs available (generally, REST APIs and online GUIs) and examples on how to use the functionalities available. In this concrete version of the Portal, the deliverable describes the solution for Continuous Integration and Continuous Deployment (based on Jenkins), highlighting the pipelines used and the monitoring system based on Zabbix. For the Single-Sign-On (based on Keycloak), the document describes the changes done in the configuration and the integration with the rest of components. For the workflows orchestration (based on Croupier), this document addresses the changes done to support Apache Mesos and Spark, as well as the new ECMWF Cloud. The training tool (based on Moodle) includes now more courses and we show some new examples. As the data management and catalogue (based on CKAN), we address the addition of the Choropleth Map extension and the data sharing features for CKAN, as well as the new Polytope tool for collecting data from ECMWF. In the case of visualization (based on Visualizer and COVISE), the document shows the new features, including new types of diagrams and 3D images that can be embedded in websites. This document also extends the information about the support tools (based on Zammad and Askbot). It also introduces the new interactive notebooks feature (based on Jupyter), so users can now develop some test code and run it easily. The last feature addressed is the new frontend, as the mean to centralize all the features and to provide some enhanced functionality for executing applications. Finally, the document describes the available infrastructure for integration and production deployments, as well as the new infrastructure available for training, listing the Virtual Machines available and the components deployed.

1 Introduction

1.1 Purpose of the document

This document aims at describing the implementation of the second release of the HiDALGO Portal (to be renamed as the Global Challenges Portal), which gives access to HiDALGO services in a simple way, as a one-stop-shop. Such solution consists of a set of tools covering several aspects useful for HiDALGO stakeholders, like training, execution of simulations, visualization of results, user support, data management and even code testing in a simple way.

The document goes through all the features implemented for the second version, including the implementation of the frontend (and backend) that puts all of them together, reducing the complexity to access the HiDALGO services.

Moreover, as a result of the review conducted in September 2020, we have included additional information about improving the user experience, by monitoring how they use the frontend and enabling the collection of feedback from their side. The adequate setup, deployment problems and errors solved in some of the components delayed the release of a fully functional version of the Portal release. Finally, we have also added additional information about tools to analyse the frontend usability in a more autonomous way, as well as a definition of the plans for involving external stakeholders in the testing and usage of the HiDALGO Portal.

1.2 Relation to other project work

This document is directly related to D5.2[1] and D5.3[2], since they describe the features and designs to be followed in the Portal implementation, according to the requirements defined in D6.1[3] and D6.4[4], as well as the first implementation done. It is closely related to D5.4[5], which provides many details about the tools used for supporting users. It is also related to the workflows defined in D6.2[6] and WP4 in general (to be supported by the Portal). It is the second release of the portal development in T5.3, that will be updated in D5.7.

1.3 Structure of the document

This document is structured in 11 major chapters:

Chapter 2 talks about the features that have been implemented in the context of the second release of the Portal, in line with the designs done previously (D5.2[1] and D5.3[2]).

Chapter 3 to **11** describe how the features were implemented, the supported functionality, the components used, the APIs available and how these features can be used in the context of HiDALGO.

Chapter 12 provides a description of the infrastructure in which the components have been deployed, both for integration and operation, including also information about the training infrastructure.

Chapter 13 just provides a summary and a set of conclusions obtained after the current implementation.

2 Features, Architecture and Roadmap

2.1 Current Features Available

The list of desired features and user stories were defined in the context of D5.2[1]. The features were categorized as application execution, application visualization, interactive code, data management, data visualization, user community support, user discovery, centralized user management and other/non-functional aspects.

The progress towards features implementation has been done in several aspects. First of all, the application execution part has been improved, facilitating the management of applications and the generation of forms for collecting the inputs, so the execution configuration will be easier. There are ongoing developments for including integration with the data catalogue in order to make even easier the selection of input files for the applications. This will be also linked with the visualization once the results are available, so it will be easier to visualize results fast, linking to Visualizer and COVISE.

The usage and orchestration of resources has also improved by enhancing the support to the existing infrastructures (i.e. the HPDA one), starting the access to new infrastructures (like the one from ECMWF) and even the support to data management.

In the case of data management, Polytope facilitates now the access to ECMWF data, so coupling will be easier and more effective. Also, there is more progress with respect to the features of the data catalogue (i.e. easier sharing of information). Efficient data movement is a topic that will be addressed in the future as well, since moving very large datasets through CKAN has shown not to be very efficient.

The area of visualization has been improved with the addition of several features to Visualizer, so more diagrams are available (and some 3D visualization work is ongoing, for richer diagrams that could be very useful for scenarios like COVID-19 simulations) and a new version of COVISE that generates HTML code for the 3D visualization of complex simulations, so now it is possible to do a much better integration with the Frontend.

The second version of the Portal has also included several tools for enabling users support. Now HiDALGO can offer a ticketing system, the Questions and Answers (Q&A) forum and a good documentation through the Wiki.

Finally, the online Notebooks are already available, so stakeholders can create their prototypes of code online and test it before launching very large executions and integrating chunks of code in complex software applications. This tool will be further developed in the future, since there is a lot of room for improvement with more libraries, tools, examples, etc.

2.2 Current Portal Architecture

Taking into account the requested features and the current implementation, we have defined the architecture that has been followed during the implementation. It is based on the previous one defined under D5.2[1], with a few modifications related to experience granted during the previous implementation and to the features that were expected for M24.

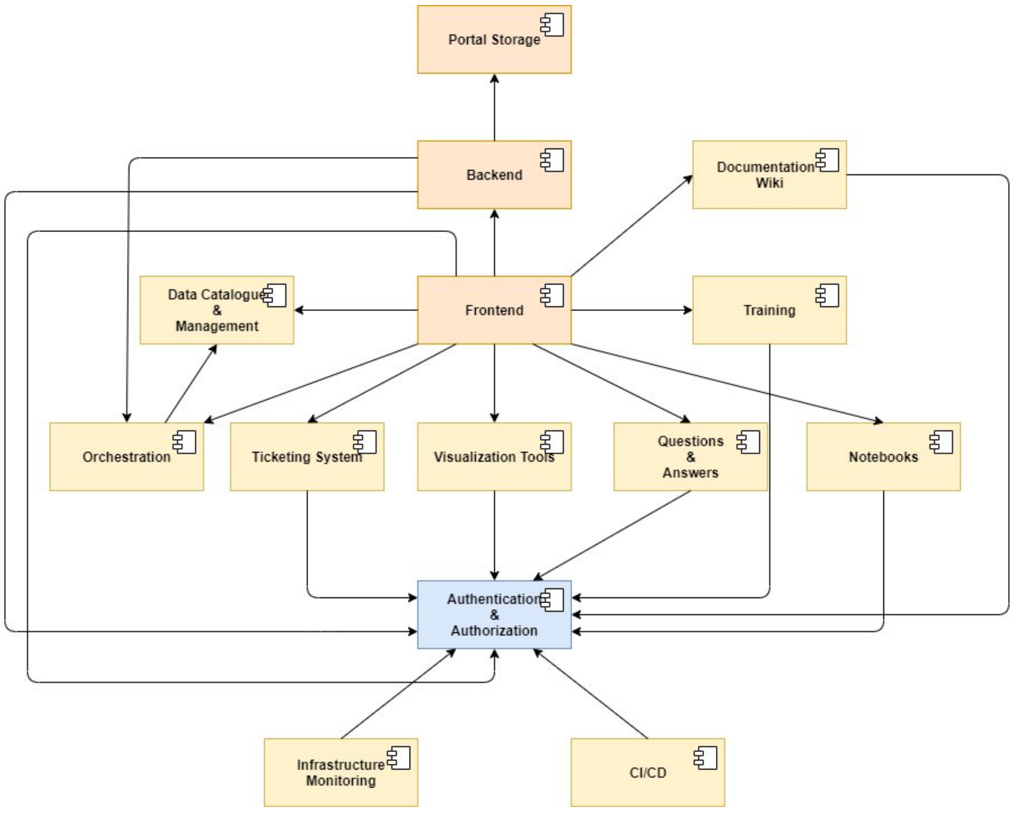


Figure 1: Current architecture of the Portal

In the end, the architecture is very similar. It has two main central points: the Frontend and the Authentication & Authorization. In that sense, now the architecture is a bit more specific with respect to differentiating the tools for supporting users and the documentation (the Ticketing System was added, as well as the Documentation Wiki). It also adds more links with the Authentication & Authorization (the other components need to check security tokens in order to enable an authorized access to the functionalities), since this is not centralized through the Frontend.

Finally, although the Frontend centralizes the access to the rest of functionalities, these are external tools, and those internal to the Frontend (applications management) are not included as part of the high-level picture. Instead, we highlight the presence of the backend and the Portal storage. The reason for the selection of this architecture is because the Frontend represents the one-stop-shop of HiDALGO, so all the components can be accessed (it embeds the GUI of other components or it links to them, in case it is not technically feasible).

The Backend takes care of the background connection with components like the Orchestrator, also guaranteeing that security is addressed, separating the business logic from the GUIs (Frontend). Although it is possible to access the Orchestrator through its GUI, such possibility will be disabled for most of the users (only administrators will be able to access to it). Finally, the Portal Storage takes care of information that the Backend needs to manage (such as the applications available, instances, etc.).

2.3 Portal Roadmap Implementation

Deliverable D5.2[1] was proposing a roadmap related to the potential user needs. For the first version, the focus was on enabling the execution of applications, as well as features related to users' management and data management (the catalogue, basically). In the case of the current version, it was important to progress more in previous features and to increase the support to applications execution, as well as data management and visualization.

Additionally, there was more progress with respect to user support and the usage of notebooks in order to facilitate code prototyping.

After the implementations, we have re-checked the implementation status with respect to the roadmap defined initially for MS4.

Main Component	Original Plan for MS4	Current Implementation
Single Sign On	Login once and access all the services	Partially Done
	All services connected to one account	Partially Done
	Users to sign up themselves	Done
	Manage users' roles and permissions	Done
	Group permissions and assign users to certain groups	Done
Portal Maintenance	Have a sandbox environment, in order to test changes quickly without risk	Done
	Test changes automatically, so new features do not break other parts of the code	Done
Application Execution	Execute a pilot and retrieve the results that are interesting to me	Done
	Abstract users from the complexity of the underlying infrastructure	Done
Application Status	Know the status of a running pilot, so users may	Done

Main Component	Original Plan for MS4	Current Implementation
Visualization	access partial results if they want to	
	Check the logs of a pilot execution in order to know what happened	Postponed
Explore and Manage Data	Allow users to explore the data available, so they can see data that can be used	Done
	Create a public dataset and share it with other users, if the user wants to	Done
	Create private datasets, so only concrete users (or groups of users) may know about it	Done
Internal Storage	Store small input datasets in the platform, so they can be used for running pilots	Done
External Storage	Use datasets stored outside HiDALGO in order to execute pilots	Done
Visualization	Visualize datasets, so it is possible to show demonstrations	Done
	Visualize datasets with temporal information, so it is possible to understand them	Done
Book	Have all documentation organized for users	Postponed
	Developers can treat documentation as code, so it is easier to keep it updated	Partially Done
	Make available information for running pilots and for using the provided UIs	Partially Done
Support	Enable the possibility to keep on discussions through email for general support information	Done
Non-Functional	Make the UI compliant with GDPR, so there will not be any legal issues	Partially Done

Table 2 Original features proposed for MS4 vs current implementation

We have also analysed the roadmap expected for MS5, related to M24, since the Portal was expected to release additional features. The analysis is in the following table.

Main Component	Original Plan for MS5	Current Implementation
Application Execution	Full abstraction of the technical complexity for users that want to run an application	Partially Done
	Possibility to easily add new applications and pilots to HiDALGO through the UI	Done

Main Component	Original Plan for MS5	Current Implementation
Application Status Visualization	Visualize workflow execution stages in real time	Ongoing
Explore and Manage Data	Allow users to access datasets information about format, quality and quantity	Done
Internal Storage	Store large input datasets in the platform, so they can be used for running pilots	Done
External Storage	Allow to store produced datasets outside HiDALGO	Done
Visualization	Visualize simulations results, so users can validate them	Done
	Visualize datasets with geospatial information, so it is possible to understand them	Done
	Visualize 3D information in such a way it can be understood by users	Done
Support	Enable the possibility to keep on discussions through an open discussion forum	Done
	Provide a FAQ, so repetitive questions can be easily answered for stakeholders	Postponed
Non-Functional	Provide a reliable UI that is accessible whenever it is necessary	Postponed
	Provide secure UI infrastructure, that avoids malicious attacks	Done

Table 3 Original features proposed for MS5 vs current implementation

Additionally, we have defined a process for opening the Portal to the public, so stakeholders will be able to use HiDALGO services through this release of the Portal. We plan to organize internal demonstrations, followed by a beta testing campaign (that will involve also external stakeholders). After such beta testing campaign, the Portal will be open for the public. The expected plans are:

- *April 2021:* Internal demos and testing with HiDALGO partners, collecting feedback directly and organizing several teleconferences;
- *May-July 2021:* Beta testing campaign organization and execution (we consider a collaboration with the ReachOut[1] project, since the tools provided are useful for collecting feedback);

[1] https://www.reachout-project.eu/view/Main/

- *September 2021:* The Portal will be open for all stakeholders and we will invite some of the members of the Associated Partners programme to use HiDALGO services.

As a result of the internal and beta testing campaigns, we will also test new look and feels for the Portal, in order to improve usability as much as possible.

3 CI/CD Infrastructure

Software development process follows the various steps from the requirement gathering to the deployment of the application on the infrastructure, and it can be optimized with the Continuous Integration (CI) and Continuous Deployment (CD) methodology to improve the quality of the product development. Jenkins[7], Git and Ansible are the well-known tools used for supporting CI/CD methodology, which is installed as mentioned in D5.3[2] for providing the experience of seamless integration and deployment within the project. Jenkins is successfully adopted within the portal development to automate the integration, deployment and testing of its services by using the Ansible scripts. Ansible is an automatic application deployment tool, which is used with Jenkins for automating the application deployment in the cloud infrastructures. Moodle[8], Cloudify[12] and Matchmaking services were already deployed using Ansible script and defined the Jenkins pipeline for the different steps like integration, testing and deployment. Jenkins and its current setup satisfied the project goals, so the changes are only limited with the secure usage of USTUTT Git repository and the provisioning of new Jenkins pipeline for the portal services (Askbot and Zammad) as detailed in the rest of this chapter.

3.1 Implemented Solution

3.1.1 CI/CD Infrastructure with Jenkins

The portal development in HiDALGO, as well as the overall chosen deployment strategy, follow well-known CI/CD patterns and best practices to automate the complete software development process. This process is detailed in the previous deliverable D5.3[2] by discussing the selected CI/CD tools, namely Jenkins for automatically building and testing software, and Ansible for automated deployment of the built artefacts. Jenkins acts as a middle-man between developers and infrastructures to automate the process of application development as defined in Jenkins pipelines.

Currently, a single Git deployment account is created for the HiDALGO project, which is restricted with read-only permissions, and is therefore only used to fetch the source code during the deployment phase. USTUTT Git is securely integrated with Jenkins by using Jenkins credential store, which is one improvement from the D5.3[2] in the CI/CD infrastructure as depicted in Figure 2, which reflects the current workflow.

The specific Jenkins workflow is detailed below; it ranges from building of a Jenkins pipeline to the deployment of applications in the corresponding Virtual Machine (VM). Each service is provided with two VMs for integration and deployment, which is detailed in Chapter 13.

1. The portal developer or administrator can submit a manual Web request to build a new Jenkins pipeline by using the Jenkins Web GUI. After successful submission and processing of the request, the pipeline is built locally in the Jenkins VM and provides access to the cloud VMs and the USTUTT Git repositories.
2. During the build of the Jenkins pipeline, the latest source code of the Ansible script is downloaded from the USTUTT Git repository for deploying the services in cloud VM.
3. The Jenkins pipeline is defined in Groovy[10] language, which automates the different stages of the software development pipeline. Currently, software deployment is automated with Ansible and the functional tests are conducted with basic shell scripts to ensure the correctness of installations. Bash scripts and Ansible scripts are run as a separate command within the Jenkins pipeline to perform the corresponding stages and provide log details of each stage in the Jenkins GUI.
4. Ansible connects with the respective cloud VMs remotely and fetches the latest source code of the corresponding service from USTUTT Git repository for deploying that service in the cloud VM.

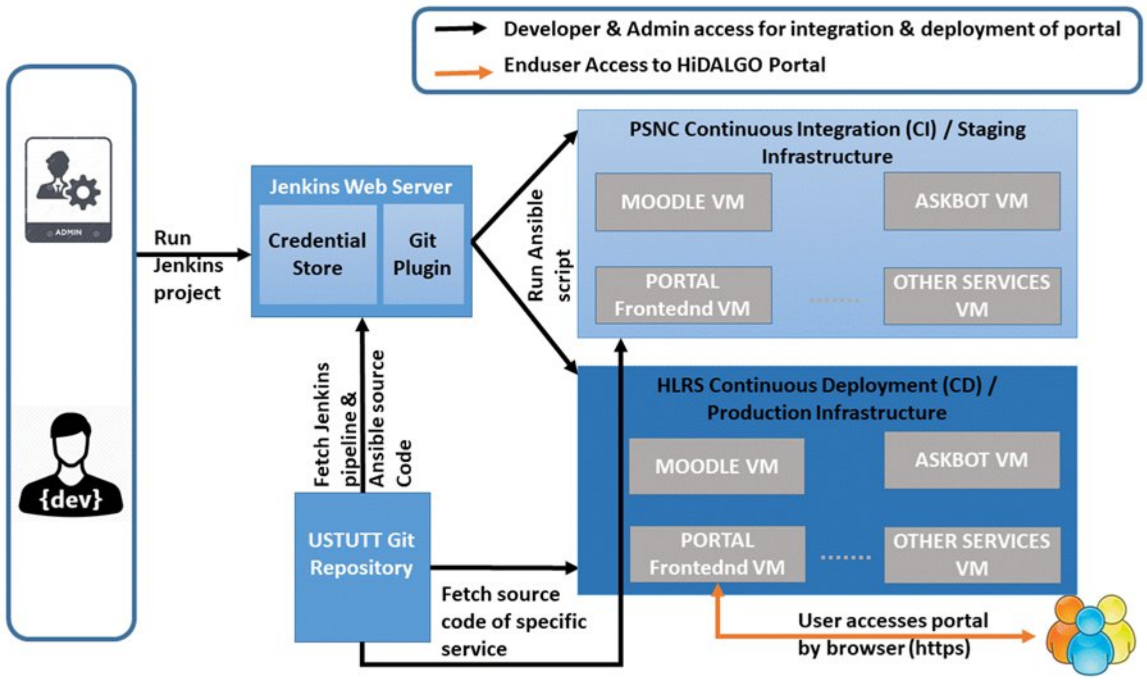
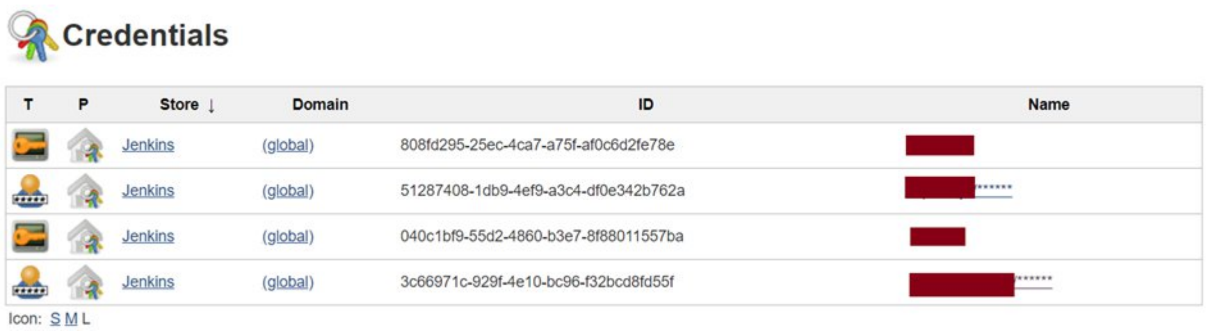

Developer & Admin access for integration & deployment of portal
Enduser Access to HiDALGO Portal
ADMIN
{dev}
Run Jenkins project
Jenkins Web Server
Credential Store
Git Plugin
Run Ansible script
PSNC Continuous Integration (CI) / Staging Infrastructure
MOODLE VM
ASKBOT VM
PORTAL Frontednd VM
OTHER SERVICES VM
HLRS Continuous Deployment (CD) / Production Infrastructure
MOODLE VM
ASKBOT VM
PORTAL Frontednd VM
OTHER SERVICES VM
Fetch Jenkins pipeline & Ansible source Code
USTUTT Git Repository
Fetch source code of specific service
User accesses portal by browser (https)

Figure 2: CI/CD Infrastructure with Jenkins and USTUTT Git repository.

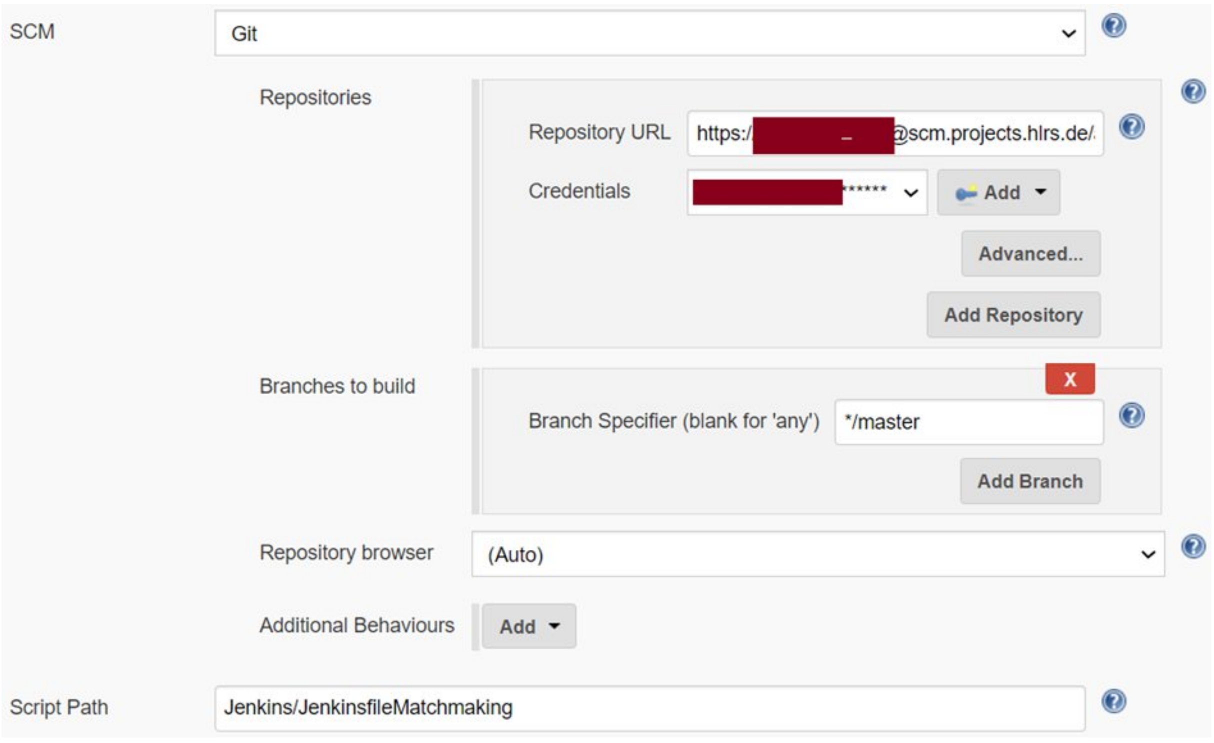
USTUTT Git relies on password-based authentication, so a Jenkins project usually includes a Git URL with the password for fetching the corresponding Git repository. This is improved right now by enabling the Git plugin and credentials store in the Jenkins configurations to fetch a Git repository without passing password values in a plain text. Jenkins' credentials store is designed to store the credentials of external applications securely so that a Jenkins project or Jenkins pipeline can refer to the credential store instead of providing plain credential in a hard-coded manner; the mechanism improves the overall security of operations. Jenkins credential store is managed by the administrator as shown in Figure 3 for storing the USTUTT Git credentials. The Jenkins pipeline is configured to use Jenkins credential store for accessing USTUTT Git SCM (Source Control Manager) as shown in Figure 4[2].


Credentials

T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	808fd295-25ec-4ca7-a75f-af0c6d2fe78e	[redacted]
		Jenkins	(global)	51287408-1db9-4ef9-a3c4-df0e342b762a	[redacted]******
		Jenkins	(global)	040c1bf9-55d2-4860-b3e7-8f88011557ba	[redacted]
		Jenkins	(global)	3c66971c-929f-4e10-bc96-f32bcd8fd55f	[redacted]******

Icon: S M L

Figure 3: Jenkins credential store to manage the HLRS credentials.


SCM: Git

Repositories
- Repository URL: https:/[redacted]@scm.projects.hlrs.de/
- Credentials: [redacted]******

Branches to build
- Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

Script Path: Jenkins/JenkinsfileMatchmaking

Figure 4: Jenkins project is configured to use Git credentials.

3.1.2 Components Monitoring with Zabbix

Zabbix is an enterprise-level software designed for real-time monitoring of millions of metrics collected from tens of thousands of servers, virtual machines and network devices. Although there are other tools available (like Prometheus, Nagios, etc...), Zabbix was selected because

[2] HiDALGO Jenkins weblink - https://hidalgo-jenkins.hlrs.de

it provides several required features by default, it is easy to configure and it performs as needed.

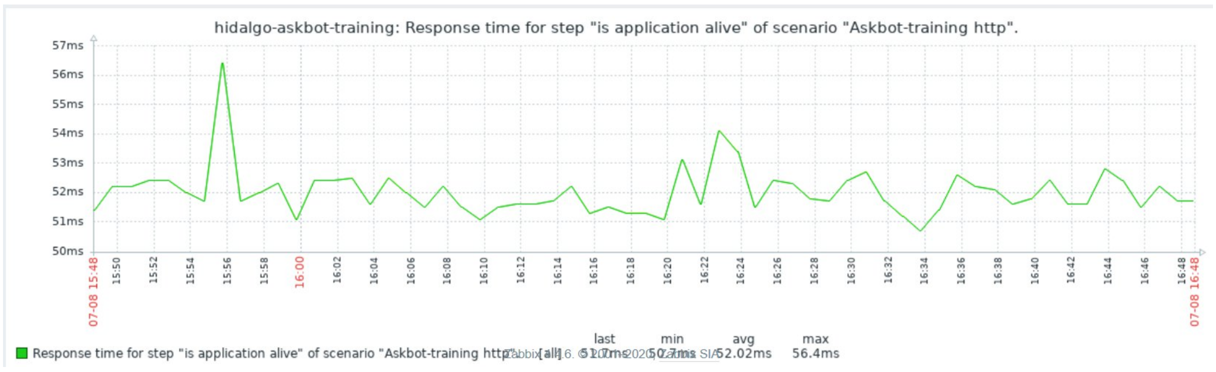


Figure 5: Zabbix monitoring Askbot response time.

Zabbix allows for monitoring any metric that can be obtained from the service, without interfering it. Moreover, it is possible to install Zabbix client on a service machine to gain access to internal variables, like CPU and RAM usage or IO delays.

Zabbix agents support both passive (polling) and active checks (trapping). Zabbix may perform checks based on an interval, however, it is also possible to schedule specific times for item polling. The following list of checks is supported by Zabbix agent out of the box:

- Network: packets/bytes transferred, errors/dropped packets, collisions
- CPU: load average, CPU idle/usage, CPU utilization data per individual process
- Memory: free/used memory, swap/pagefile utilization
- Disk: space free/used, read and write I/O
- Service: process status, process memory usage, service status, DNS resolution, TCP connectivity, TCP response time
- Files: file size/time, file exists, checksum, MD5 hash, RegExp search
- Logs: text log, Windows eventlog
- Other: system uptime, system time, users connected, performance counter (Windows)

The usage of Zabbix is focused on the monitoring of the Portal components. For each component deployed, we have been creating the corresponding hosts and metrics in Zabbix, in such a way it is possible to monitor their availability, response time, resources consumption, etc… This information may be also useful in order to scale (up or down) the VMs and containers resources accordingly. The services and items monitored by the time we write this report are listed in the following table.

Host	Monitored Items
Askbot (https://askbot.hidalgo-project.eu/)	Download speed Last error message Response code

Host	Monitored Items
CKAN (ribes-212.man.poznan.pl ribes-21.man.poznan.pl ribes-135.man.poznan.pl rhus-134.man.poznan.pl), Streaming (http://sophora-145.man.poznan.pl)	Database status Services status Webserver status CPU load Disk utilization Disk read/write rates Number of logged in users Number of running processes System boot time Network speed Network errors Available memory Checksum of /etc/passwd Zabbix agent availability

Table 4: List of courses, course instructors and the objective of the course is detailed here.

CKAN and Askbot are currently included in Zabbix to monitor its availability and status, which will be further extended in the project duration to include all other services and portal for supporting the automatic application monitoring. Moodle, Cloudify, Wiki.js, Zammad, Interactive notebook, Visualization tools and Portal will be monitored in a single place with Zabbix to ensure the correct operations of HiDALGO portal by the system administrator.

3.2 Usage and Examples

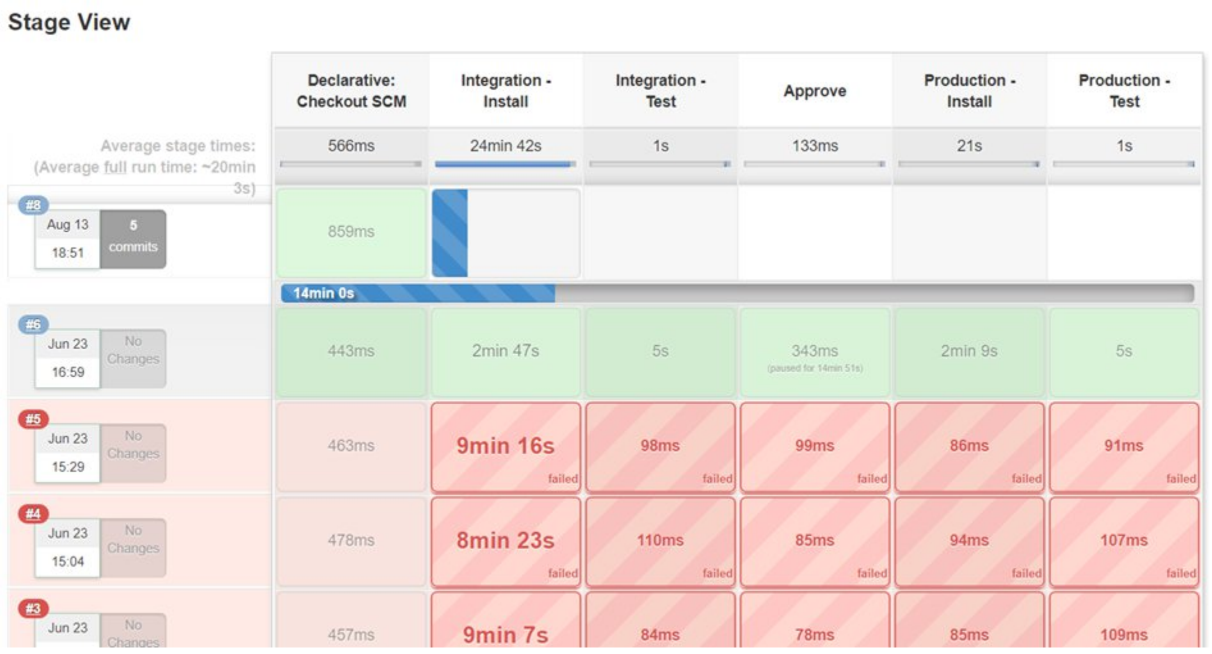
3.2.1 Development Pipelines

An automatic software development pipeline (or Jenkins pipeline) consists of multiple stages and each stage runs specific functionalities with multiple tools to automate the software development activities (i.e. compile, run automated tests, etc). Six stages are defined in the Jenkins pipeline, which is detailed below.

1. Checkout SCM - Clone the Jenkins pipeline and Ansible scripts to Jenkins workspace
2. Install in the integration infrastructure - Build or install in the integration environment
3. Integration test – Test the basic functionalities by using the ping and curl commands

4. Manual approval for deployment – Wait for manual approval from developer or admin
5. Install in the deployment infrastructure - Build or install in the deployment environment
6. Deployment test - Test the basic functionalities by using the ping and curl commands

Jenkins pipelines were provided for Moodle, Cloudify and Matchmaking services as detailed in D5.3[2], which is extended with Zammad and Askbot services. Zammad and Askbot are the newly introduced services for providing customer support ticket and a community forum, which both are automated through Jenkins pipelines; detailed in Appendix 1. The execution of the Zammad pipeline is shown in Figure 6[3], and it is similar to other pipelines. Ansible script is used for the installation of service in the integration and deployment VMs. The bash script is used for testing the installation with ping and curl commands to ensure the verification of installation.



Stage View

	Declarative: Checkout SCM	Integration - Install	Integration - Test	Approve	Production - Install	Production - Test
Average stage times: (Average full run time: ~20min 3s)	566ms	24min 42s	1s	133ms	21s	1s
#8 Aug 13 18:51 5 commits	859ms	14min 0s				
#6 Jun 23 16:59 No Changes	443ms	2min 47s	5s	343ms (paused for 14min 51s)	2min 9s	5s
#5 Jun 23 15:29 No Changes	463ms	9min 16s failed	98ms failed	99ms failed	86ms failed	91ms failed
#4 Jun 23 15:04 No Changes	478ms	8min 23s failed	110ms failed	85ms failed	94ms failed	107ms failed
#3 Jun 23 No Changes	457ms	9min 7s	84ms	78ms	85ms	109ms

Figure 6: Jenkins pipeline with six stages for Zammad.

3.2.2 Zabbix Monitoring

As Zabbix is set up, it is possible to check out the metrics already in place, but it is also possible to include new metrics and to monitor new components (seen as 'hosts' by Zabbix), as we add them to the HiDALGO Portal.

Adding a new host

[3] HiDALGO Jenkins installation - https://hidalgo-jenkins.hlrs.de/job/zammad/

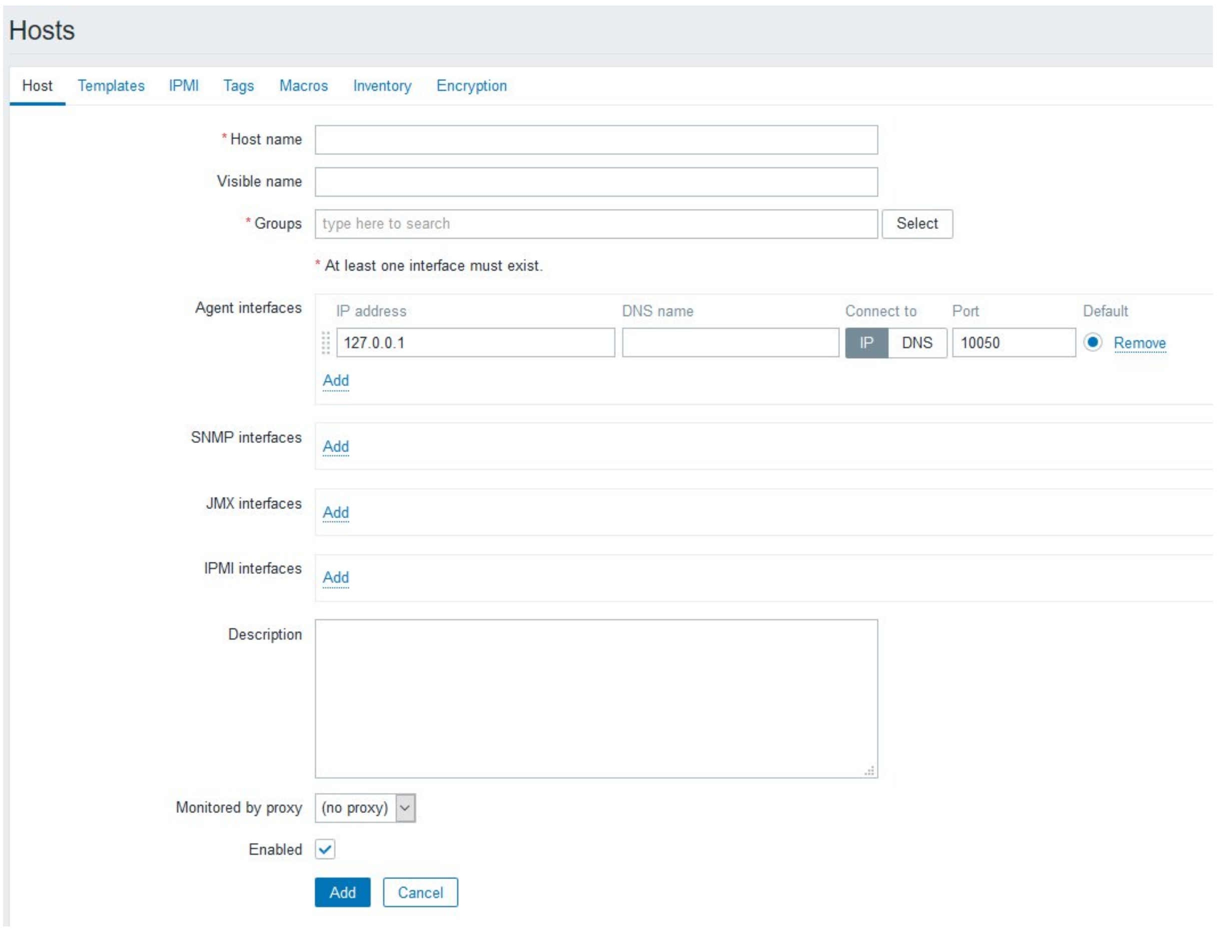
It is necessary to first point to the service that is going to be monitored. In 'Hosts' submenu under 'Configuration' tab, click 'Create host' button.

The required fields are:

• Host name – name displayed in Zabbix

• Groups – a group the server belongs to (as configured before)

• Interfaces - IP or DNS address with port



* Host name

Visible name

* Groups: type here to search [Select]

* At least one interface must exist.

Agent interfaces:

IP address	DNS name	Connect to	Port	Default
127.0.0.1		IP / DNS	10050	● Remove

SNMP interfaces: Add

JMX interfaces: Add

IPMI interfaces: Add

Description

Monitored by proxy: (no proxy)

Enabled: ✓

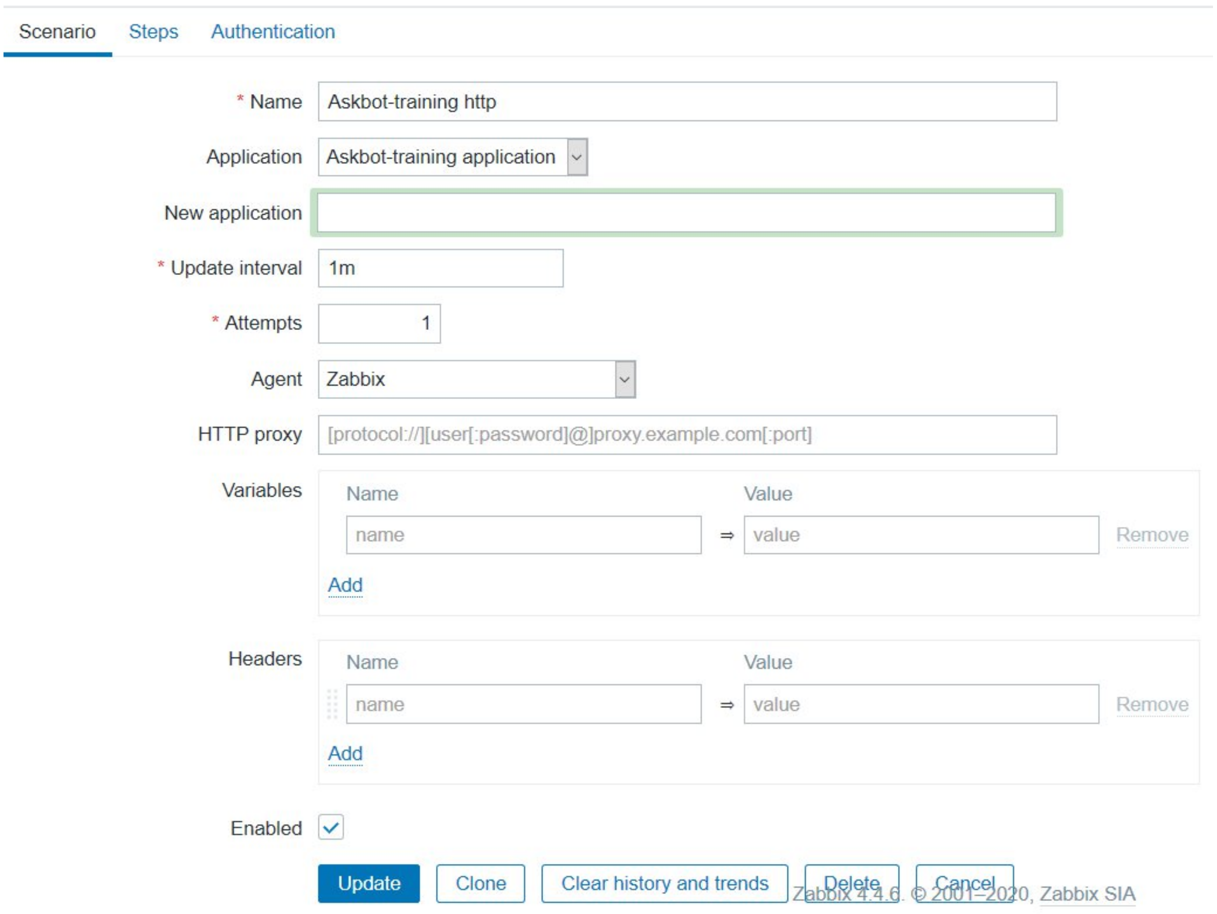
Figure 7: Adding a new host in Zabbix.

Adding a web scenario

Next important step is specifying what metrics should be collected, e.g. code returned from the service. On the Host properties screen, access the 'Web scenarios' tab and click 'Create web scenario' button.

In 'Steps' tab add new step and provide following info:

- Name – scenario identification.
- Application – previously defined application
- Update interval – time between checks (e.g. 1 minute)
- Agent – Zabbix
- Enabled – Yes



Scenario | Steps | Authentication

* Name	Askbot-training http
Application	Askbot-training application
New application	
* Update interval	1m
* Attempts	1
Agent	Zabbix
HTTP proxy	[protocol://][user[:password]@]proxy.example.com[:port]
Variables	Name: name ⇒ Value: value
Headers	Name: name ⇒ Value: value
Enabled	✓

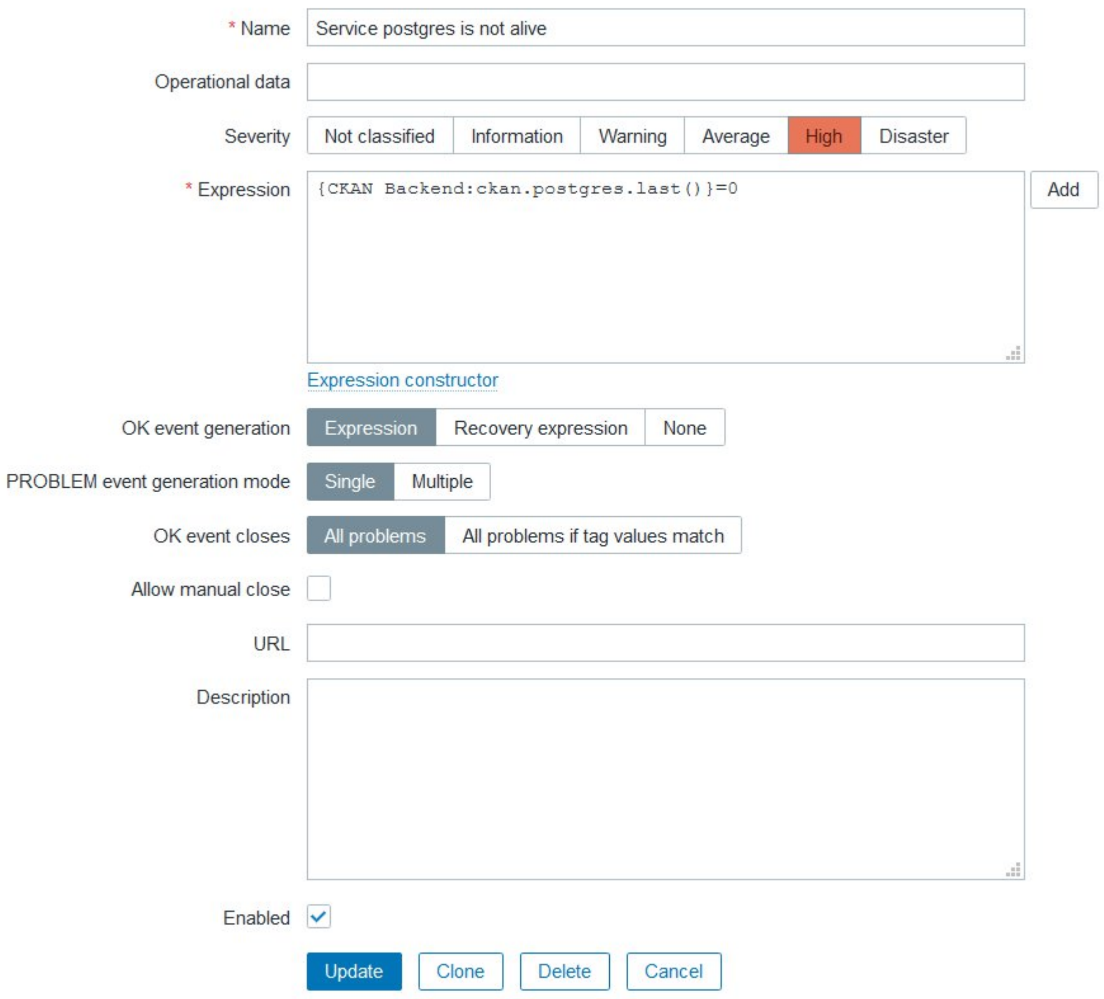
Figure 8: Adding a new web scenario in Zabbix.

Configuring Zabbix Agent

In order to monitor more inaccessible internal metric it is necessary to first install Zabbix Agent on the client machine:

```
sudo apt install zabbix-agent.
```

Now it is possible to add to the host new items that utilize Agent’s capabilities.



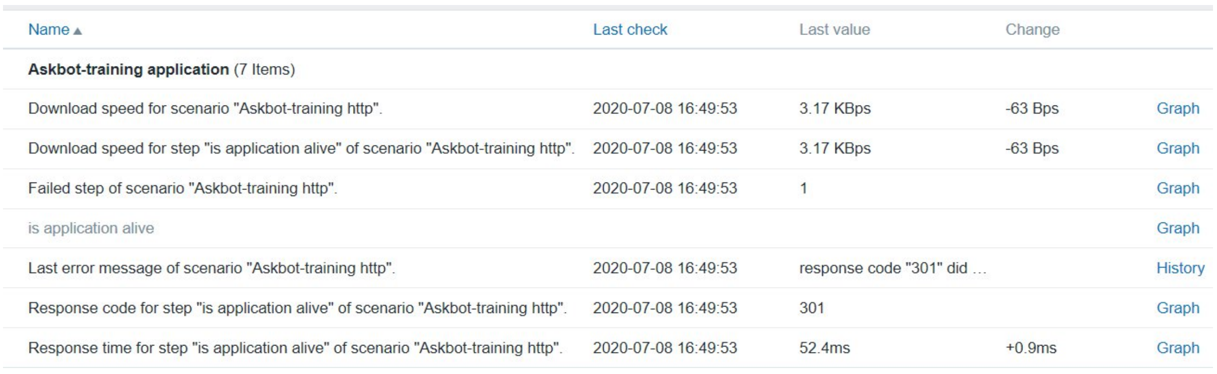
* Name	Service postgres is not alive
Operational data	
Severity	Not classified / Information / Warning / Average / [✓ High] / Disaster
* Expression	{CKAN Backend:ckan.postgres.last()}=0
	Expression constructor
OK event generation	[✓ Expression] / Recovery expression / None
PROBLEM event generation mode	[✓ Single] / Multiple
OK event closes	[✓ All problems] / All problems if tag values match
Allow manual close	☐
URL	
Description	
Enabled	✓

Figure 9: Example of database trigger.

Testing and viewing data

After the service has been set up, it is now be monitored according to the items/scenarios. It takes about 5 minutes to gather enough data for graphs to appear.

In order to view the data access the ‘Monitoring’ menu and click on the ‘Latest data’ option.



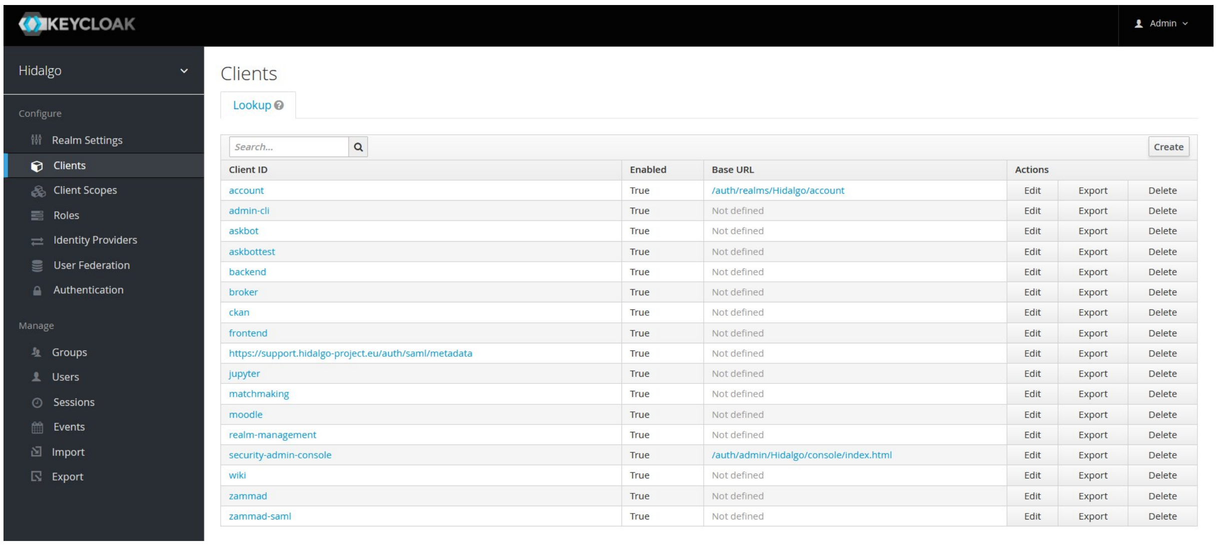
Name ▲	Last check	Last value	Change	
Askbot-training application (7 Items)				
Download speed for scenario "Askbot-training http".	2020-07-08 16:49:53	3.17 KBps	-63 Bps	Graph
Download speed for step "is application alive" of scenario "Askbot-training http".	2020-07-08 16:49:53	3.17 KBps	-63 Bps	Graph
Failed step of scenario "Askbot-training http".	2020-07-08 16:49:53	1		Graph
is application alive				Graph
Last error message of scenario "Askbot-training http".	2020-07-08 16:49:53	response code "301" did ...		History
Response code for step "is application alive" of scenario "Askbot-training http".	2020-07-08 16:49:53	301		Graph
Response time for step "is application alive" of scenario "Askbot-training http".	2020-07-08 16:49:53	52.4ms	+0.9ms	Graph

Figure 10: Accessing metrics in Zabbix.

4 Single Sign On

4.1 Implemented Solution

As already explained in D5.3[2], Keycloak[11] was selected as the solution for Identity Management (IDM) and Single Sign On (SSO) solution. At this stage, two instances of Keycloak are available and connected to several components of the Portal. Keycloak supports several protocols in order to enable SSO and, in this case, we are using OAuth2 (see standards like OpenID[14] and SAML[13]) as the main way to do it. Central Authentication Service (CAS)[15] has been considered for components like Moodle, but for the moment, we are using the same solution for all the components, in order to ease maintenance.



Clients

Client ID	Enabled	Base URL	Actions		
account	True	/auth/realms/Hidalgo/account	Edit	Export	Delete
admin-cli	True	Not defined	Edit	Export	Delete
askbot	True	Not defined	Edit	Export	Delete
askbottest	True	Not defined	Edit	Export	Delete
backend	True	Not defined	Edit	Export	Delete
broker	True	Not defined	Edit	Export	Delete
ckan	True	Not defined	Edit	Export	Delete
frontend	True	Not defined	Edit	Export	Delete
https://support.hidalgo-project.eu/auth/saml/metadata	True	Not defined	Edit	Export	Delete
jupyter	True	Not defined	Edit	Export	Delete
matchmaking	True	Not defined	Edit	Export	Delete
moodle	True	Not defined	Edit	Export	Delete
realm-management	True	Not defined	Edit	Export	Delete
security-admin-console	True	/auth/admin/Hidalgo/console/index.html	Edit	Export	Delete
wiki	True	Not defined	Edit	Export	Delete
zammad	True	Not defined	Edit	Export	Delete
zammad-saml	True	Not defined	Edit	Export	Delete

Figure 11: Keycloak clients configuration.

One realm has been created and the list of clients is configured for such realm, according to the components to be integrated. Up to now, the components integrated with the SSO are: frontend and backend of the Portal, Moodle, CKAN, Jupyter notebooks and Zammad. The initial configuration is already available for more components of the Portal, although such integration is ongoing, since in some cases is more complicated to address, or it cannot be addressed directly through the GUI, requiring a backend that takes care of the interaction (as in the case of the Orchestrator).

As already explained in D5.3[2], each time a user needs to be authenticated, the components contact Keycloak. If the user already logged in and has been granted access to the component, Keycloak will provide the corresponding security token. Otherwise, Keycloak is the one identifying the user.

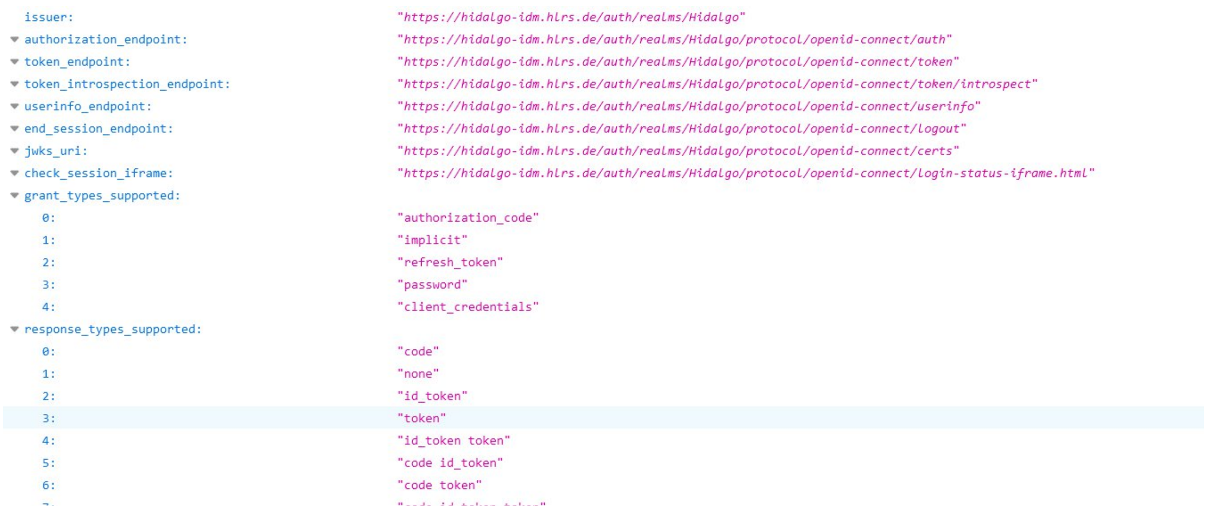
In order to facilitate the usage by stakeholders, the configuration is being changed in order to enable self-registration of the users. This requires configuring a SMTP server, since it is necessary to activate the email verification feature, for security reasons. Additionally, the team is looking at the way to deal with temporary accounts that would be provided to stakeholders, so they can be disabled when certain period expires. This feature will require additional functionality in the frontend side in order to check users' profiles.

The list of software components used is the following:

- Keycloak, version 12.04[4]
- Nginx Proxy, version 1.19.3[5]
- Letsencrypt Nginx Companion, version 2.1.0[6]
- Postgres Database, version 9.4.26[7]

4.2 Available APIs

As explained in D5.3, Keycloak follows several standards that have to do with IDM and SSO. The most important interface in use is the OpenID[14] endpoints, which enable SSO through OAuth2. Additionally, SAML2.0[13] is available, for identity provider metadata.



```
issuer:                            "https://hidalgo-idm.hlrs.de/auth/realms/Hidalgo"
▾ authorization_endpoint:          "https://hidalgo-idm.hlrs.de/auth/realms/Hidalgo/protocol/openid-connect/auth"
▾ token_endpoint:                  "https://hidalgo-idm.hlrs.de/auth/realms/Hidalgo/protocol/openid-connect/token"
▾ token_introspection_endpoint:    "https://hidalgo-idm.hlrs.de/auth/realms/Hidalgo/protocol/openid-connect/token/introspect"
▾ userinfo_endpoint:               "https://hidalgo-idm.hlrs.de/auth/realms/Hidalgo/protocol/openid-connect/userinfo"
▾ end_session_endpoint:            "https://hidalgo-idm.hlrs.de/auth/realms/Hidalgo/protocol/openid-connect/logout"
▾ jwks_uri:                        "https://hidalgo-idm.hlrs.de/auth/realms/Hidalgo/protocol/openid-connect/certs"
▾ check_session_iframe:            "https://hidalgo-idm.hlrs.de/auth/realms/Hidalgo/protocol/openid-connect/login-status-iframe.html"
▾ grant_types_supported:
    0:                             "authorization_code"
    1:                             "implicit"
    2:                             "refresh_token"
    3:                             "password"
    4:                             "client_credentials"
▾ response_types_supported:
    0:                             "code"
    1:                             "none"
    2:                             "id_token"
    3:                             "token"
    4:                             "id_token token"
    5:                             "code id_token"
    6:                             "code token"
```

Figure 12: Keycloak OpenID endpoints.

Keycloak also has a web console which allows to manage realms, clients, users, configuration, etc… This web interface also includes the forms for doing login and registration activities, which are exposed by Keycloak.

[4] https://hub.docker.com/r/jboss/keycloak
[5] https://hub.docker.com/r/jwilder/nginx-proxy
[6] https://hub.docker.com/r/jrcs/letsencrypt-nginx-proxy-companion
[7] https://hub.docker.com/_/postgres?tab=description&page=1&name=9.4

4.3 Usage and Examples

Deliverable D5.3[2] already described the main feature related to Keycloak and how to use them. As shown in Figure 11, the Keycloak dashboard is the place to configure the realm characteristics, as well as the clients, users and other aspects (groups, federations, etc.). In the case of HiDALGO, the configuration was focused on the clients part, since each component requires to configure a client.

Additionally, we recently changed the login configuration, so now users are allowed to self-register, as a way to facilitate the involvement of stakeholders. Therefore, now the login form has changed and, when selecting the 'Register' option, the following form has to be completed.

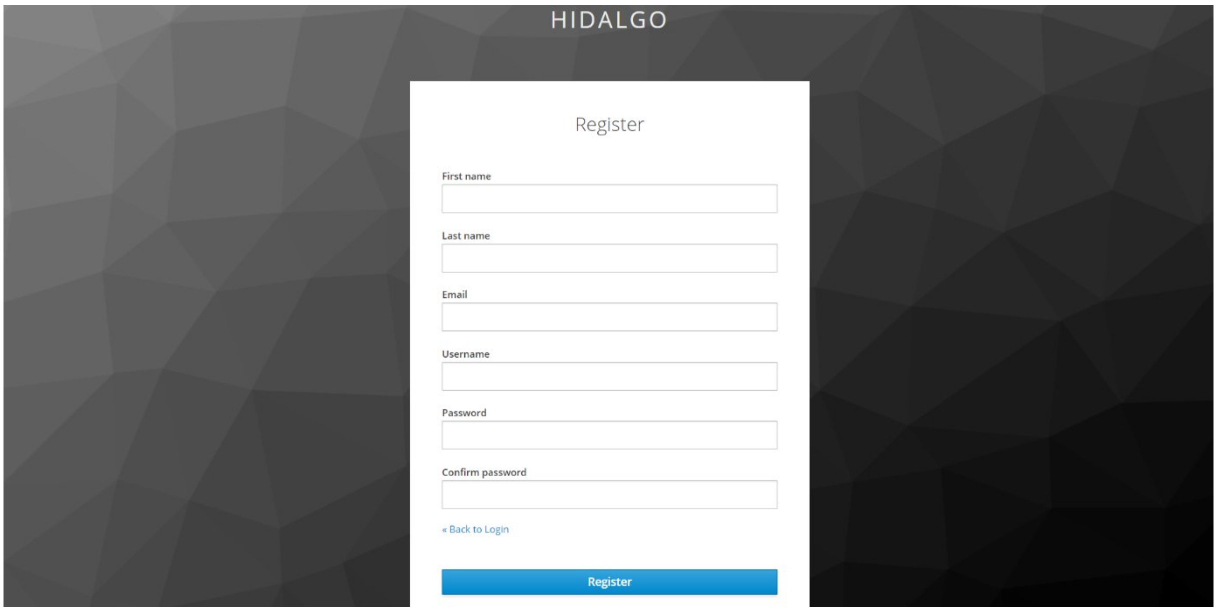


Figure 13: Keycloak registration form.

After the form completion, the user receives an email to activate the account and it will be possible to make login with the new user.

5 Workflows Orchestration

5.1 Implemented Solution

5.1.1 European Weather Cloud

As part of the coupling of ECWMF weather data for the UAP and human migration pilots, the HiDALGO orchestrator aims to access cloud resources hosted by the ECWMF European Weather Cloud (EWCloud) to perform data retrieval and post-processing.

This integration with EWCloud aims at achieving the following:

- Reducing the size of the data that will be transferred to the pilot applications
- Allowing familiarisation with ECMWF data and its processing
- Offering an extendible platform for future applications

The remainder of this section gives a general overview of EWCloud, its capabilities and a status of the integration.

European Weather Cloud

In December 2018, ECMWF and EUMETSAT joined forces to set up a federated Cloud Computing infrastructure focused on meteorological data. The vision is to establish a "European Weather Cloud" to serve the European Meteorological Infrastructure and its users. ECMWF is currently running a pilot phase of two years, started in January 2019. EWCloud revolves around the concept of "users-to-the data", by providing to users transparent access to services, infrastructure and data holdings based on agreed federation principles. Federation in the EWCloud is a loose coupling and is implemented via the following elements:

- Common web presence

  A single website marking the web presence of the "European Weather Cloud". The website purpose is for communication and public relations.

- Hybrid Cloud Management System (HCMS)

  The Hybrid Multi-Cloud Management System (HCMS) acts as an orchestration layer that runs on top of cloud infrastructures and abstracts away the heterogeneity of the underlying cloud technology (VMware, OpenStack or other) and geographical distribution of infrastructures from the end-users perspective. The HCMS allows the creation and management of processing environments (i.e. Virtual Machines, Kubernetes deployments) in any of the underlying infrastructures. A COTS technology called Morpheus is used as HCMS.

- Harmonised Data Access API

  ECMWF data holdings (MARS, C3S, CAMS, etc.) are accessible for downloading data, using the existing APIs, both via the Internet and via EWCloud. The preferred option is, however, to reduce as much as possible the transfer of data via the Internet and, instead, perform computation close to the data by relying on a higher local network speed. This is the main motivation for the integration of EWCloud with the Hidalgo orchestrator Cloudify Croupier.

EWCloud-Cloudify Croupier Integration

Different level of integrations can be conceived between Cloudify Croupier and EWCloud. In the current approach, ECMWF has setup a number of Virtual Machines (VMs) with the post-processing capabilities required by the pilot applications. The intention is to allow Cloudify to connect to these VMs via SSH and execute the algorithm required by the workflow. Preliminary tests have been successfully conducted but a full integration has not been achieved yet.

5.1.2 CKAN Client Support

As a way to ease data management tasks, the Croupier Orchestrator[18] now allows for the definition of operations for data movement and publication, embedded in the tasks definition. In previous versions of the orchestrator, it was necessary to create a new 'hpc.nodes.Job' task of type 'SHELL', which was running scripts accessing the CKAN Client. Although the scripts have been improved, now it is also possible to have a normal HPC task (a 'croupier.nodes.Job') which, at the end of the task definition, includes a new option called 'data_mover_options', that specifies the configuration of data movement.

The current version of the Orchestrator now supports data movement with GridFTP (as a result of the collaboration with the EUXDAT project), thanks to this 'data_mover_options' tag, and it is supporting more and more features of the data movement through CKAN, in line with the features provided by the CKAN Client. It is possible to use data publication in CKAN thanks to this new option, and more developments are done in the scripts, so it will be possible to move datasets with a concrete identifier or name.

The plan for the near future is to combine the usage of GridFTP and CKAN client depending on the circumstances (the Orchestrator will make a choice) and some monitoring information could be extracted from data movement tasks. Also, we will analyse how to support solutions like Polytope (see Section 7), directly from the Orchestrator.

5.1.3 Implementation of the Apache Spark Extension

Croupier is the plugin developed by ATOS for managing HPC applications in CoeGSS project, which is further enhanced together with USTUTT in HiDALGO for executing Spark applications in HPDA infrastructures. Croupier Apache Spark extension is based on the Apache Mesos REST API v1.0[19] and the application submission command (spark-submit) for managing the execution of Spark applications in USTUTT HPDA infrastructure. A potential target HPDA infrastructure has therefore to support both the Apache Mesos REST API version 1.0 and spark-submit command to interface croupier plugin with the infrastructure.

USTUTT offers access to a Cray Urika GX system within the project to execute Apache Spark applications (cf. Deliverable D5.1[20] on page 12). This system is used for testing the Croupier Apache Spark extension. As a first proof-of-concept, the well-known Apache Spark word count example was successfully deployed through the plugin onto USTUTT's infrastructure.

The Urban Air Pollution pilot intends to make significant use of Apache Spark in their workflow. Thus, this pilot is considered as the first real-world application to test and evaluate the developed Apache Spark extensions in croupier plugin. We will report on the findings in the next iteration of this deliverable.

The Croupier plugin manages the operations of job submission, monitoring and cancelling of jobs as mentioned below:

- There is no generic REST API for submitting jobs in Mesos scheduler, and it depends on the framework used by the application. Spark application is initially supported with the spark-submit command to submit jobs in batch mode, and the command manages job submission with Mesos scheduler. In the future, if there should be the requirement to support a wider range of HPDA frameworks (e.g. TensorFlow), then the platform has to provide the commands or procedures to submit jobs in batch mode.
- Apache Mesos provides a REST API to get the status of an application by using /framework API. The API is used for sending the application status details to Cloudify web GUI through the Croupier plugin.
- Apache Mesos provides a REST API to cancel the application during the application execution by using /teardown API. The API is used for cancelling the application when the user requests the cancellation from the Cloudify web GUI.

```
wordcount_job:                              # Spark Wordcount Applications
     type: croupier.nodes.Job
     properties:
         job_options:
             type: 'SPARK'
             pre:                           # Run commands before submitting applications
               - 'module load tools/proxy'
             application: {get_input: job_app_full_path}   # Full path of *.jar or *.py
             application_params:            # New features to provide application parameters
               - {get_input: app_ip_forest_depth}
               - {get_input: app_ip_forest_trees}
               - {get_input: app_ip_training_ratio}
```

```
              total_executor_cores: {get_input: job_executor_cores}# Nb of executor cores
              executor_memory: {get_input: job_executor_memory}    # Size of executor memory
              driver_cores: {get_input: job_driver_cores }         # Size of driver cores
              driver_memory: {get_input: job_driver_memory }       # Size of driver memory
              post:                   # Run commands after executing applications
                - 'module unload tools/proxy'
          deployment:
            bootstrap: 'scripts/download_input.sh' # Prologue scripts to download input files
            revert: 'scripts/upload_output.sh'     # Epilogue scripts to upload output files
            inputs:
              - { get_secret: hidalgo_ckan_key }        # CKAN Key
              - { get_input: upload_file_location }     # Upload file location
              - { get_input: download_file_location }   # Download file location
```
Table 5: Cloudify blueprint for defining the workflow of Apache Spark word count application.

Table 5 provides the generic Cloudify blueprint for defining the workflow of word count example as an Apache Spark application by using the following options (features are highlighted in bold with inline comments prefixed by #):

1. `application`: This option provides the full path to the Apache Spark application. Apache Spark applications are supported either as a compiled JAR or Python file.
2. `class_name`: By providing a JAR file as a full path, then the class name is required to be provided as well in order to invoke an application.
3. `application_params`: This optional parameter allows to define additional application parameters, which will be passed as arguments to the application.
4. `total_executor_cores`: The specified number of executor cores are used to reserve the required amount of computing resources for execution.
5. `executor_memory`: Spark application submitted with the size of executor memory specified here.
6. `driver_cores`: Spark application submitted with the number of driver cores specified here.
7. `driver_memory`: Spark application submitted with the size of driver memory specified here.

The Croupier Spark extension is available as a fork of the official Croupier plugin from ATOS Github repository, and it has been merged with the official ATOS repository after completing unit tests and code review. The documentation of Croupier will be updated accordingly in order to provide details to the usage of the Apache Spark extension so that pilots can update their blueprints according to their data analytics workflow needs.

5.2 Available APIs

The APIs of the Orchestrator component have not changed with respect to the ones presented in D5.3 (section 5.2). Since the API is the standard Cloudify interface, there are no changes at all. Also, the blueprint included as example (in Annex 3 of D5.3) remains as a valid example on how to run an example.

As a new development, the backend of the Portal is now able to connect to the Orchestrator through a specific client, so the frontend can be used now to run instances of workflows. Such specific interface is shown in section 11.

5.3 Usage and Examples

The Orchestrator can be used directly through its own web interface, which was already shown in the deliverable D5.3[2]. Table 5 shows an example of a blueprint for running an application in a platform with Apache Spark. This example can be executed through the command line interface, through the Orchestrator GUI or through the Frontend.

The new Frontend features and the implemented backend are described in Section 11 of this document. Such section also shows examples for running applications through the Orchestrator using the new Frontend.

6 Training

6.1 Implemented Solution

Training is one of the key services of the HiDALGO portal. It is detailed in Deliverable D5.3[2] together with the architecture and installation of the Moodle[8] application onto cloud VMs. The service is further enhanced to support the features as per the needs of the project, which is detailed below:

1. A self sign-in feature is enabled with email verification, so the user can register to Moodle and verify their registration by a verification-link to improve security in the user registration.
2. An official HiDALGO email (no-reply@hidalgo-project.eu) is set it up for sending email notifications through Moodle. That enables us to send HiDALGO portal users course notifications from an official email to ensure the authenticity of the communication.
3. A course instructor can upload the file with the size limit of up to 15MB, so large files can be shared in the course instead of default 2MB.

The following HiDALGO courses are created and maintained by the course instructors as mentioned in Table 6, and it is accessible by students after self-enrolling.

Course Name	Purpose & Description	Course Instructor
Cloudify & CKAN	Explains the Cloudify and CKAN tools. The main objective of the course is to define the workflow of GSS or HPC applications in Cloudify.	USTUTT and PSNC
HiDALGO and its Services	Introduce available HiDALGO services.	PSNC
Migration Pilot (Modelling and Tools)	Tools used for developing the Migration application with Python3.	BUL
HPC usage tutorial	Submit HPC applications using Slurm batch scheduler and MPI commands.	PSNC
Social Networks Pilot	Introduce the Social networks application, motivation and its workflow.	PLUS
Urban Air Pollution (UAPv1.0) QuickStart Tutorial for Beginner	Introduce the UAP application, motivation and its workflow.	SZE

Table 6: List of courses, course instructors and the objective of the course is detailed here.

6.2 Available APIs

The Moodle is a standard web application with a GUI interface, which is detailed in the previous Deliverable D5.3[2] to provide the base platform for offering HiDALGO online courses. The system is enhanced with the new functionalities to support self-sign-in and single-sign-on authentication as shown in Figure 14. OAuth2 protocol is configured in the system to support the self-sign-on authentication and the feasibility to support the integration with the portal. The system is also configured to support automatic E-mail notification, so the user can get the regular notification through E-mail as shown in Figure 15. The system supports automatic E-mail notification for the self-help functionalities activities such as the user activation and course registration to avoid any manual intervention.

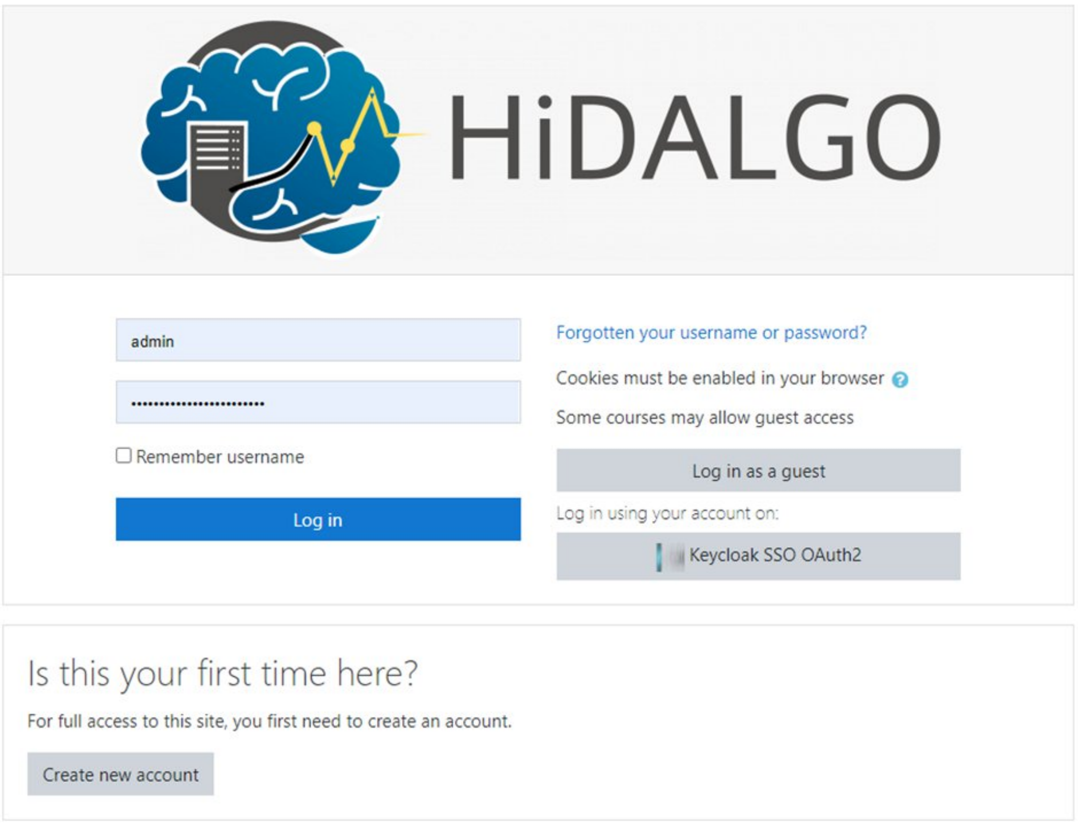
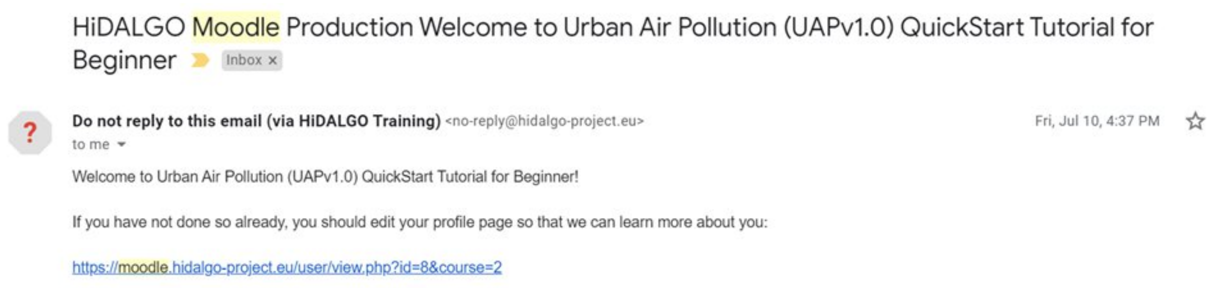

HiDALGO
admin
••••••••••••••••••••••••
Remember username
Log in
Forgotten your username or password?
Cookies must be enabled in your browser
Some courses may allow guest access
Log in as a guest
Log in using your account on:
Keycloak SSO OAuth2
Is this your first time here?
For full access to this site, you first need to create an account.
Create new account

Figure 14: Users can self-register by using the "Create new account" button and self-sign-in by using the "Keycloak SSO Oauth2" button.


HiDALGO Moodle Production Welcome to Urban Air Pollution (UAPv1.0) QuickStart Tutorial for Beginner

Inbox ×

Do not reply to this email (via HiDALGO Training) <no-reply@hidalgo-project.eu>

Fri, Jul 10, 4:37 PM

to me

Welcome to Urban Air Pollution (UAPv1.0) QuickStart Tutorial for Beginner!

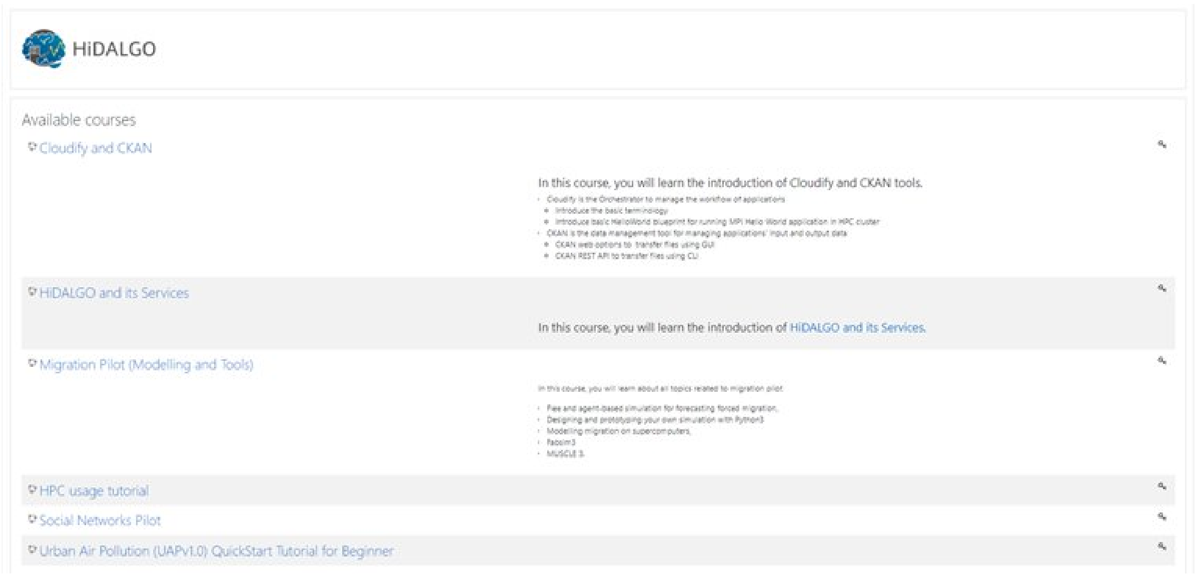
If you have not done so already, you should edit your profile page so that we can learn more about you:

https://moodle.hidalgo-project.eu/user/view.php?id=8&course=2

Figure 15: Moodle email notification for course registration.

6.3 Usage and Examples

The service is mainly hosted for sharing the training materials with the end-users, so the different courses are created for disseminating the course materials within HiDALGO community. Figure 16 provides the list of available course in the moodle for self-study and it is currently restricted with the self-enrolment key for the course registration as shown in Figure 19. Courses are organized with different topics to share the slides, documents and ZIP files in a central location as shown in Figure 18.


HiDALGO

Available courses

Cloudify and CKAN

In this course, you will learn the introduction of Cloudify and CKAN tools.

- Cloudify is the Orchestrator to manage the workflow of applications
  - Introduce the basic terminology
  - Introduce basic HelloWorld blueprint for running MPI Hello World application in HPC cluster
- CKAN is the data management tool for managing applications' input and output data
  - CKAN web options to transfer files using GUI
  - CKAN REST API to transfer files using CLI

HiDALGO and its Services

In this course, you will learn the introduction of HiDALGO and its Services.

Migration Pilot (Modelling and Tools)

In this course, you will learn about all topics related to migration pilot.

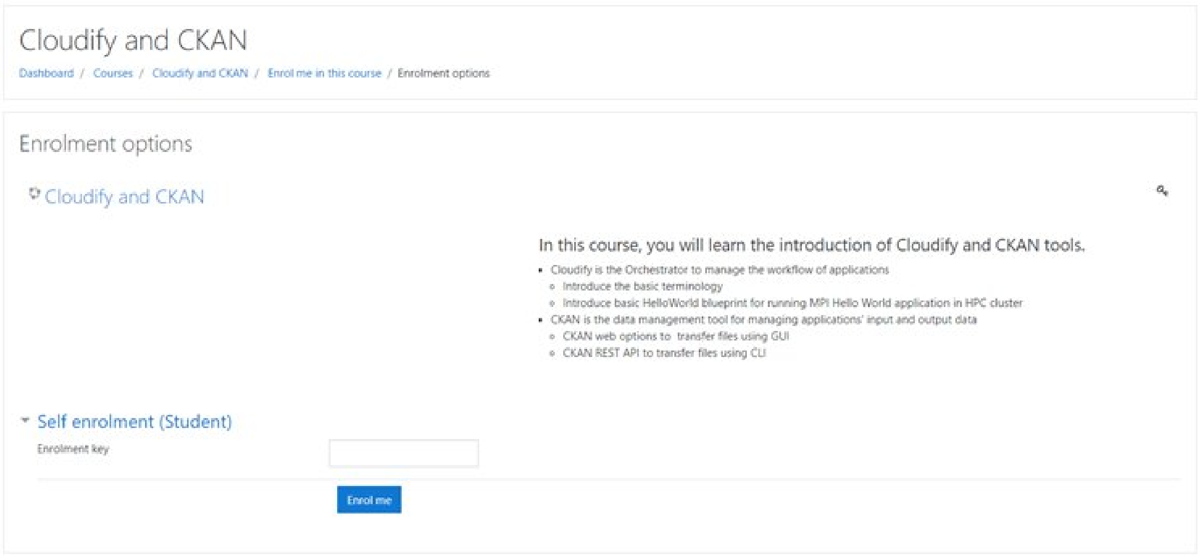
- Flee and agent-based simulation for forecasting forced migration,
- Designing and prototyping your own simulation with Python3
- Modelling migration on supercomputers,
- Fabsim3
- MUSCLE 3.

HPC usage tutorial

Social Networks Pilot

Urban Air Pollution (UAPv1.0) QuickStart Tutorial for Beginner

Figure 16: Home page details the list of available courses and its descriptions.


Cloudify and CKAN

Dashboard / Courses / Cloudify and CKAN / Enrol me in this course / Enrolment options

Enrolment options

Cloudify and CKAN

In this course, you will learn the introduction of Cloudify and CKAN tools.

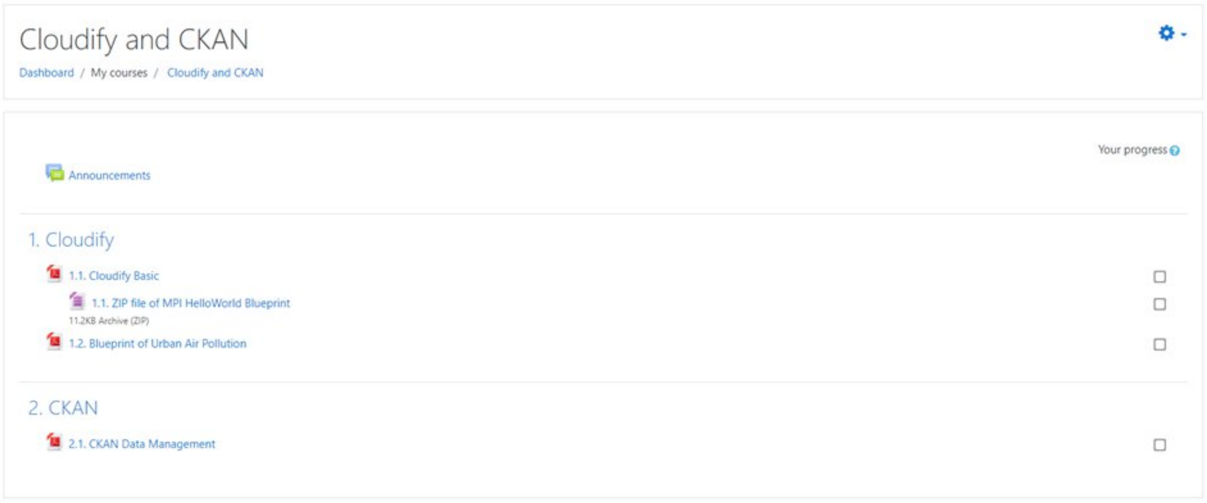
- Cloudify is the Orchestrator to manage the workflow of applications
  - Introduce the basic terminology
  - Introduce basic HelloWorld blueprint for running MPI Hello World application in HPC cluster
- CKAN is the data management tool for managing applications' input and output data
  - CKAN web options to transfer files using GUI
  - CKAN REST API to transfer files using CLI

Self enrolment (Student)

Enrolment key

Enrol me

Figure 17: Students can self-enrol the course by using the enrolment key.


Cloudify and CKAN

Dashboard / My courses / Cloudify and CKAN

Your progress

Announcements

1. Cloudify

1.1. Cloudify Basic

1.1. ZIP file of MPI HelloWorld Blueprint

11.2KB Archive (ZIP)

1.2. Blueprint of Urban Air Pollution

2. CKAN

2.1. CKAN Data Management

Figure 18: Content of the Cloudify and CKAN course.

7 Data Management and Catalogue

7.1 Implemented Solution

7.1.1 Polytope

We have implemented a polytope web API interface and polytope downloader tool that facilitate the collection of data from ECMWF, served through the Climate Data Store (CDS). The web interface is a standalone application that allows users to:

- Create a polytope data request
- Browse the requests done
- Download output data from the polytope servers

To get started you must have a valid API key from ECMWF. In the web interface you should log in with your email and API key. Once this is configured, it is possible to use the forms in order to specify the data to download and to check the historical data about the requests done so far. It is important to clarify that it is a kind of intermediate system so, first, users do the request (to the MARS system) and, then, they have to access the lists of requests and they will be able to download the data once the request is finished.

Additionally, the users can also make use of the polytope downloader, which is a Python script for download data, but through a command line. The polytope-downloader input parameters are the following:

- --auth - auth data (email:key)
- --req - request id
- --path – output file path

7.1.2 Choropleth Map for CKAN

We have installed and configured the CKAN[9] extension named ckanext-mapviews. This extension adds regular and choropleth maps to CKAN, using the new Resource View being developed on CKAN's master branch (currently unreleased). The current version of CKAN deployed in HiDALGO is v2.9.2.

To start creating choropleth maps, you need two things: the data you want to plot, and a GeoJSON defining the geographical regions you'd like to plot it. The data itself needs to be in a resource inside the DataStore, and the map needs to be in the same domain as CKAN itself (to avoid same-origin policy issues). The easiest way to do so is to upload the GeoJSON as another resource.

Each GeoJSON feature needs a property related to a column in the data. It can be an id, name, or anythings that uniquely identifies that feature, so we know where to plot the data.

7.1.3 Datasets Sharing

Organizations are the primary way to control who can see, create and update datasets in CKAN. Each dataset can belong to a single organization, and each organization controls access to its datasets.

Datasets can be marked as public or private. Public datasets are visible to everyone. Private datasets can only be seen by logged-in users who are members of the dataset's organization. Private datasets are not shown in dataset searches unless the logged in user (or the user identified via an API key) has permission to access them.

After some time of using the CKAN platform, we noticed that we needed the functionality of sharing datasets between users from different organizations.

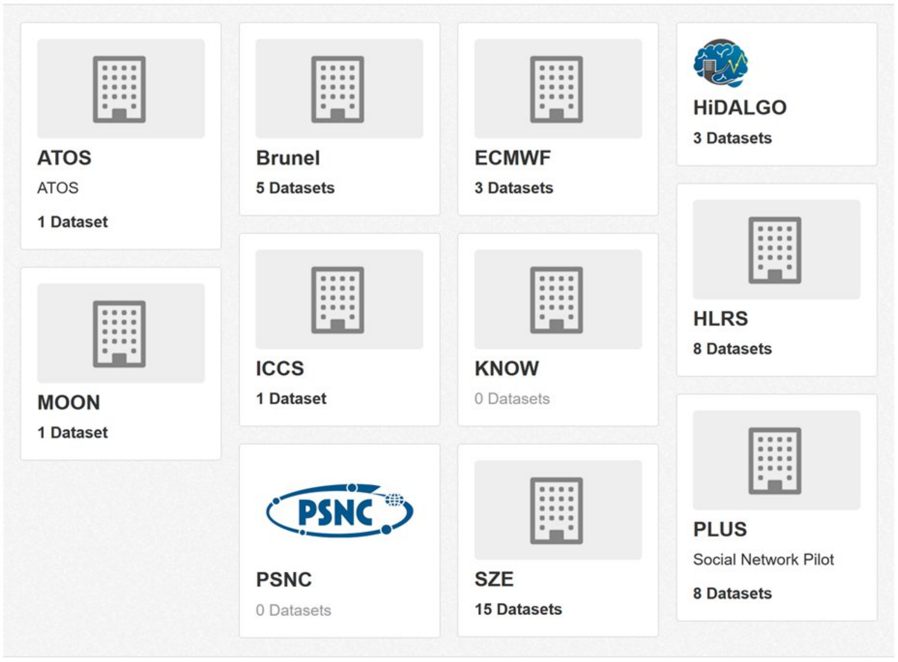
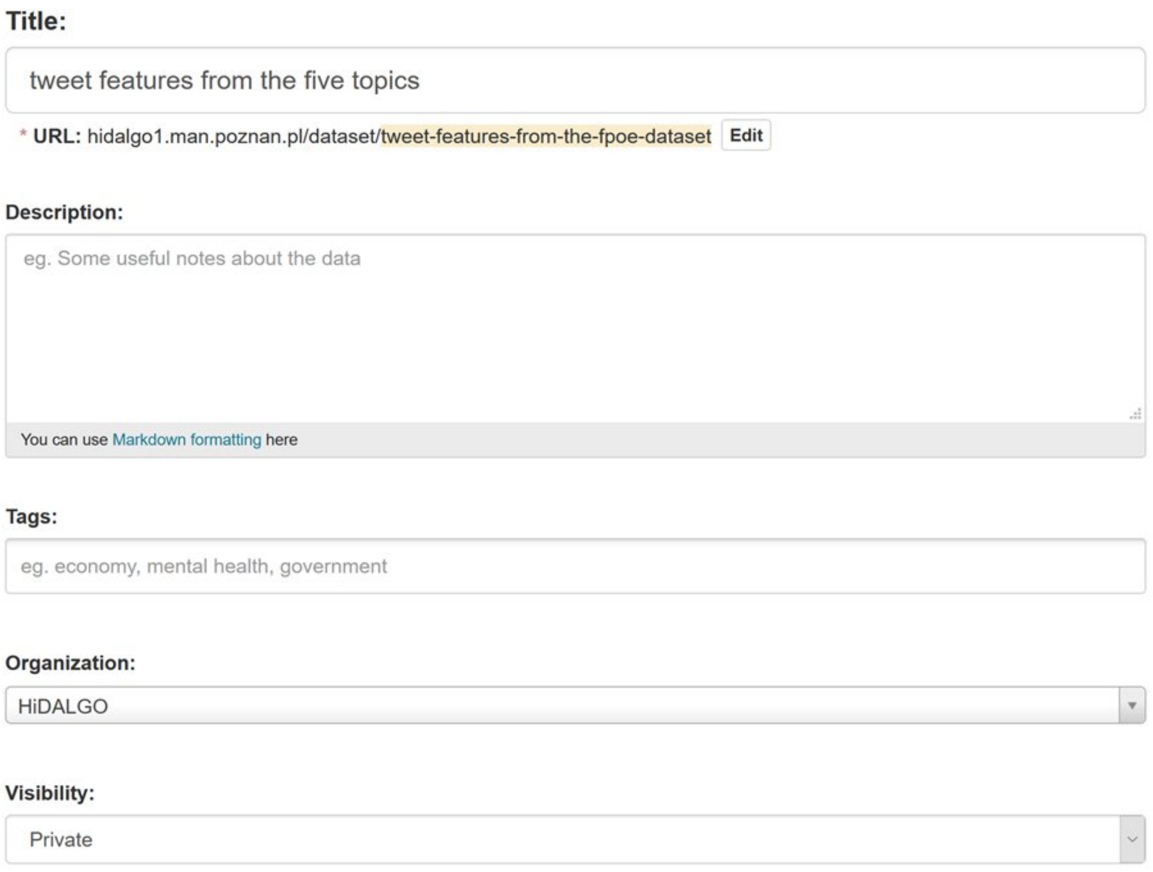


Figure 19: Organization list in the CKAN.

We have created a dedicated organization named HiDALGO. Each user of the HiDALGO project belongs to his basic organization (e.g. PSNC, SZE, etc.) and additionally to the HiDALGO organization. As a result, each private dataset assigned to the HiDALGO organization is visible to all users of the HiDALGO project.


Title:

tweet features from the five topics

* **URL:** hidalgo1.man.poznan.pl/dataset/tweet-features-from-the-fpoe-dataset Edit

Description:

eg. Some useful notes about the data

You can use Markdown formatting here

Tags:

eg. economy, mental health, government

Organization:

HiDALGO

Visibility:

Private

Figure 20: Dataset form related to the HiDALGO organization.

7.2 Available APIs

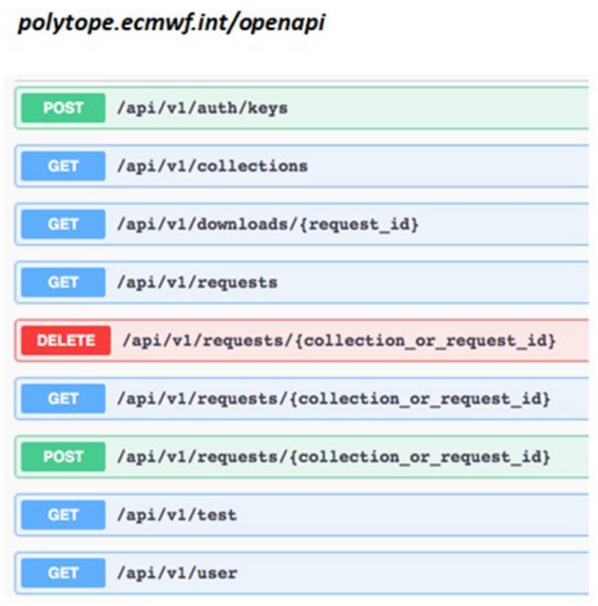
Every day, ECMWF produces ~120TiB of raw weather data, represented as a six-dimensional collections. The raw data is also stored in the world's largest meteorological archive (MARS), currently holding over 300 PiB of primary data - which is also served around the world on demand.

As explained before, ECMWF has developed the Polytope, an open-source service which allows users to request arbitrary n-dimensional stencils ("polytopes") of data from highly-structured n-dimensional datasets. The data extraction is performed server-side (collocated with the data), allowing for large data reduction prior to transmission and less complexity for the user.

The polytope API is located at http://polytope.ecmwf.int/openapi/ and allows you to:

- Authorize
- Get collections
- Download data
- Get specific request or list of requests on collection

- Delete request
- Test if the server is alive
- Get user information



polytope.ecmwf.int/openapi

POST /api/v1/auth/keys
GET /api/v1/collections
GET /api/v1/downloads/{request_id}
GET /api/v1/requests
DELETE /api/v1/requests/{collection_or_request_id}
GET /api/v1/requests/{collection_or_request_id}
POST /api/v1/requests/{collection_or_request_id}
GET /api/v1/test
GET /api/v1/user

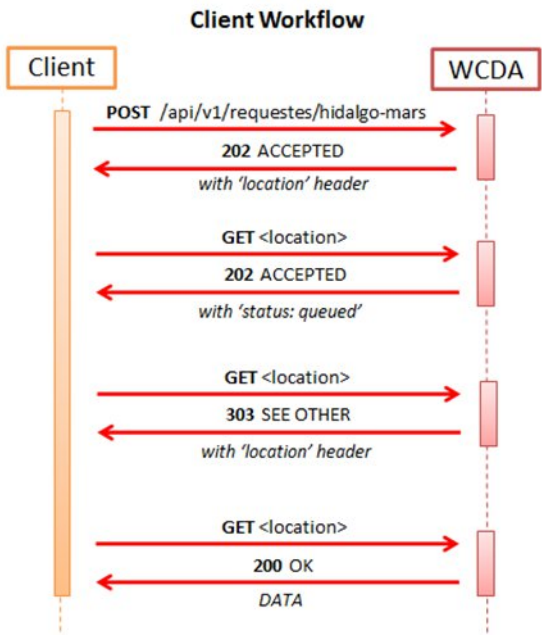


Figure 21: Dataset form related to the HiDALGO organization.

Additionally, a web interface has been created, in order to facilitate the way to access to the features of Polytope. The next section shows such GUI and how it should be used.

7.3 Usage and Examples

7.3.1 Polytope

To get started with the online client, you must have a valid API key from ECMWF. In the web interface you should log in with your email and API key.

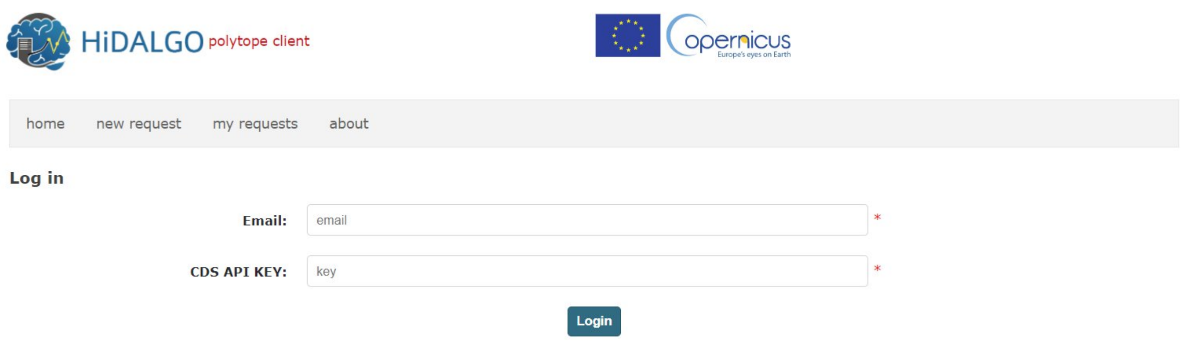
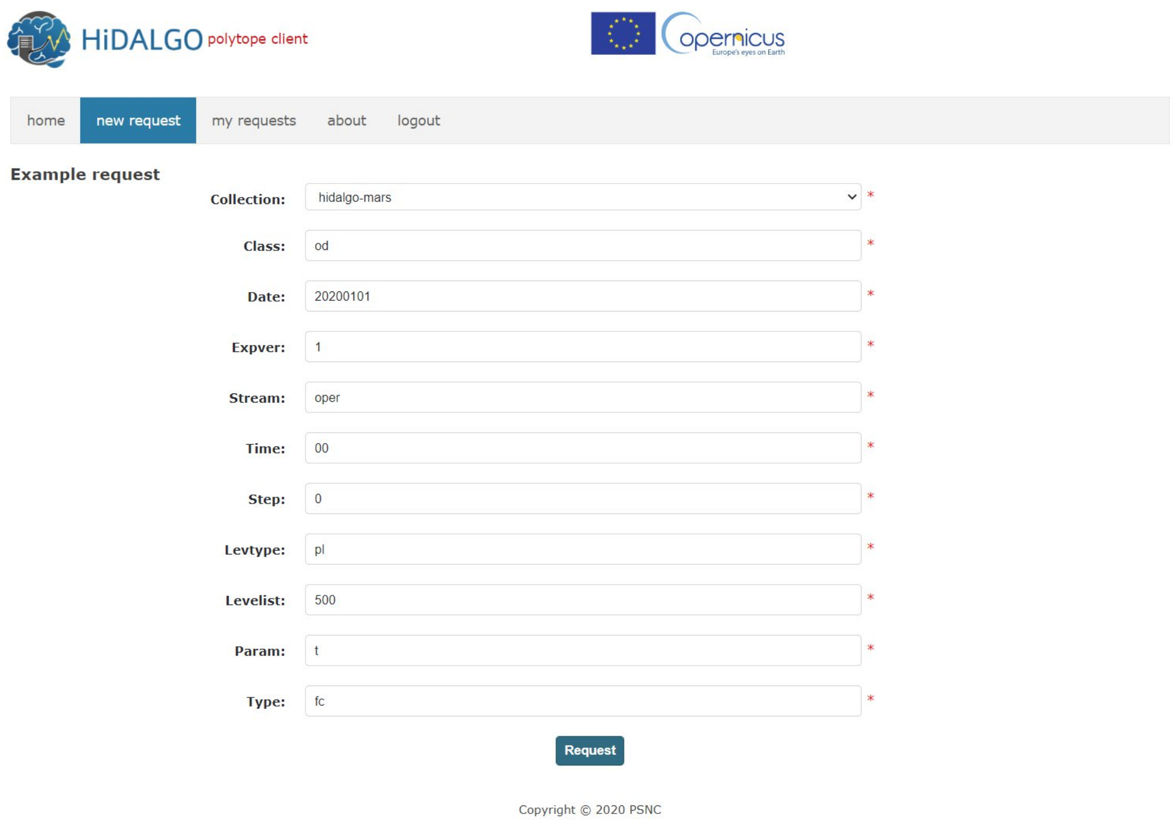


Figure 22: The polytope web API interface – log in form.

In the “new request” form user should provide several parameters related with ECMWF API. There is a dedicated collection named “hidalgo-mars” to the HiDALGO users.

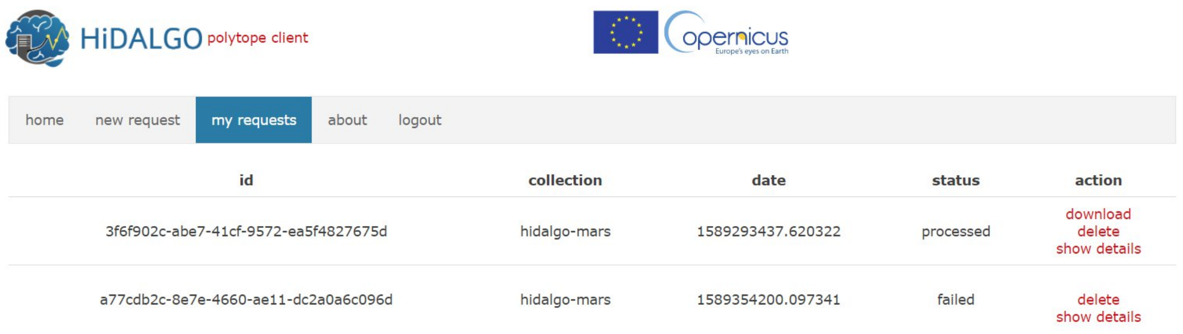


Example request

Collection:	hidalgo-mars *
Class:	od *
Date:	20200101 *
Expver:	1 *
Stream:	oper *
Time:	00 *
Step:	0 *
Levtype:	pl *
Levelist:	500 *
Param:	t *
Type:	fc *

Copyright © 2020 PSNC

Figure 23: The polytope web API interface – new polytope request.

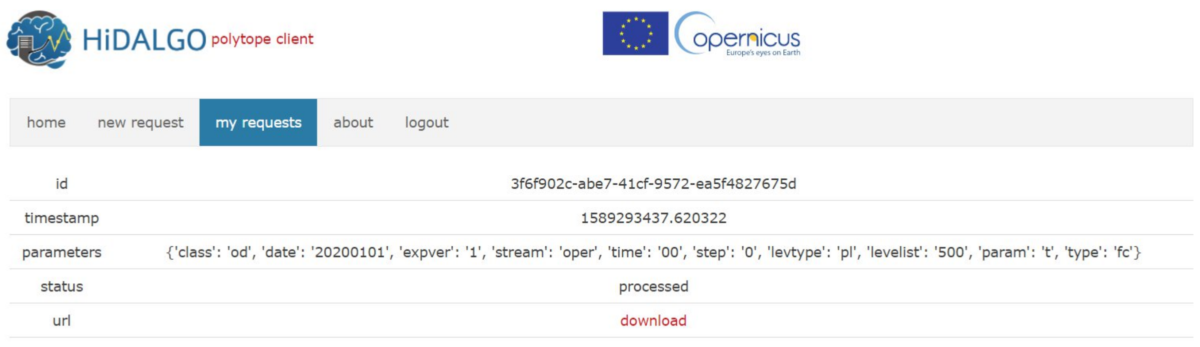


id	collection	date	status	action
3f6f902c-abe7-41cf-9572-ea5f4827675d	hidalgo-mars	1589293437.620322	processed	download delete show details
a77cdb2c-8e7e-4660-ae11-dc2a0a6c096d	hidalgo-mars	1589354200.097341	failed	delete show details

Figure 24: The polytope web API interface – request list.

From the “my requests” list user have ability to:

- Download the output data from the polytope servers
- Browse request details
- Delete request



id	3f6f902c-abe7-41cf-9572-ea5f4827675d
timestamp	1589293437.620322
parameters	{'class': 'od', 'date': '20200101', 'expver': '1', 'stream': 'oper', 'time': '00', 'step': '0', 'levtype': 'pl', 'levelist': '500', 'param': 't', 'type': 'fc'}
status	processed
url	download

Figure 25: The polytope web API interface – request details.

As for the Polytope downloader, an example (and its results) would be the following:

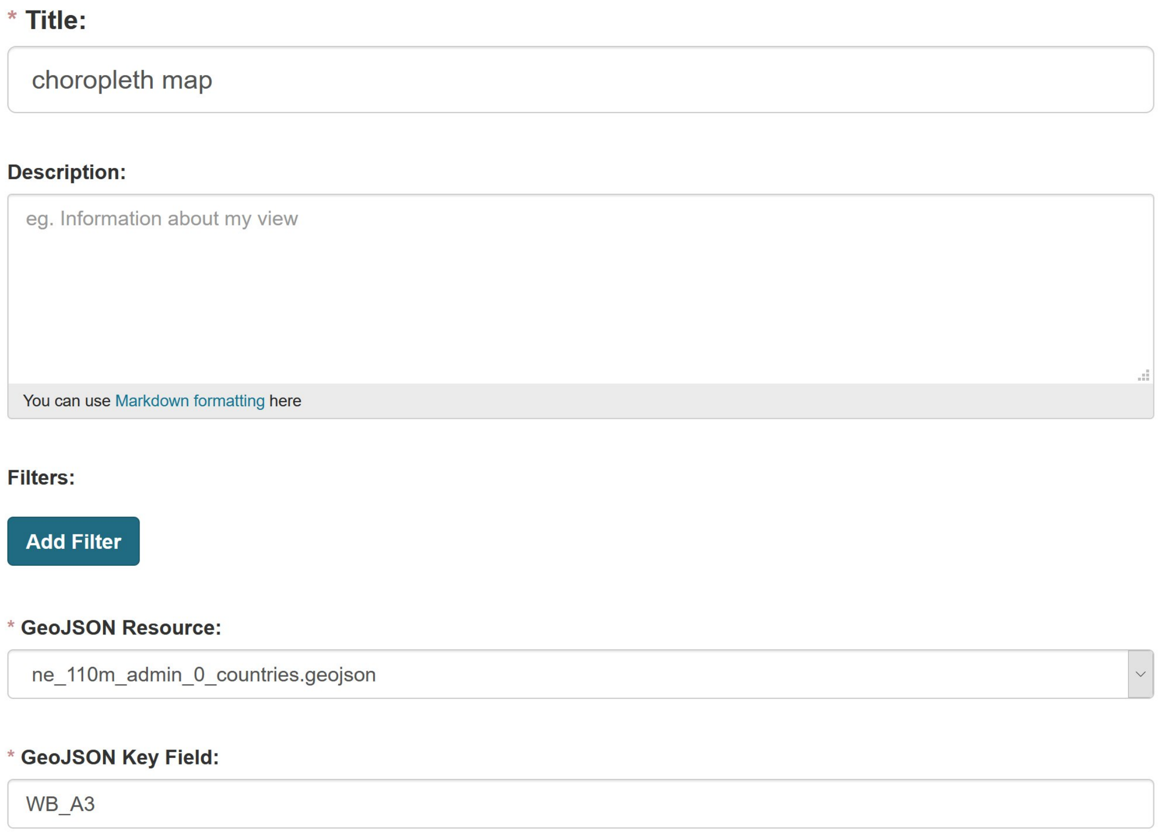
```
polytope-downloader.py --auth "email:key" --req "reqid"
--path “download/file.grib”
>Directory download already exists
>Found request: reqid; Status: processing
>Found request: reqid; Status: processed
>Data saved to download/file.grib
```

Finally, in the case of the REST API, it is also possible to run HTTP calls to the API. As an example, this is a request to execute the method to get collections:

```
curl -X GET
"http://polytope.ecmwf.int/api/v1/collections" -H
"accept: application/json" -H  "Authorization:
email:apiKey"
Response status: 200
Response body:
{
  "message": [
    "debug",
    "dummy",
    "ecmwf-mars",
    "fdb-test",
    "hidalgo-mars",
    "lexis-mars",
    "mars-test",
    "webmars-test"
  ]
}
```

7.3.2 Choropleth Map

Go to the data file's manage resource page and create a new Choropleth Map view. You'll see a form with a few fields. Enter a title, leave the description empty (if you want). Now we need to add the GeoJSON. Select in the GeoJSON Resource field the resource you created.



* Title:

choropleth map

Description:

eg. Information about my view

You can use Markdown formatting here

Filters:

* GeoJSON Resource:

ne_110m_admin_0_countries.geojson

* GeoJSON Key Field:

WB_A3

Figure 26: Choropleth Map form view.

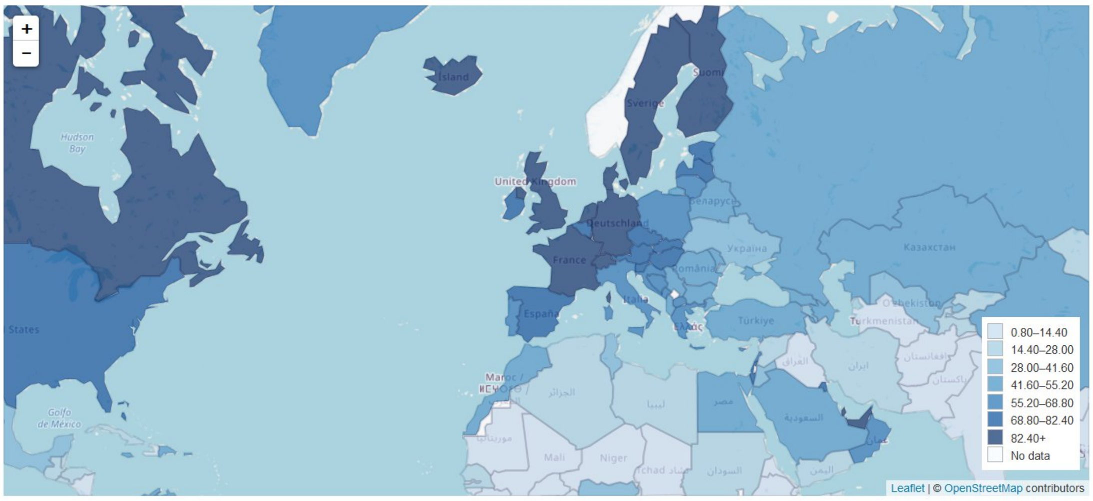


Figure 27: Example of choropleth map – Internet users per 100 people.

8 Visualization

8.1 Implemented Solution

In HiDALGO two visualization tools with different functionalities are provided and are accessible within the HiDALGO portal. In order to visualise data interactively in a dashboard the visualization tool Visualizer, developed by Know-Center, is used. Three-dimensional simulation data is visualized in the HLRS software COVISE. Both tools keep under development under WP3 and they are described separately in the following sections.

8.1.1 Visualizer

Visualizer is a web-based visualization tool enabling users to investigate large tabular data sets. It allows users to easily configure new dashboards with a few clicks. Dashboards can be shared with other users or deployed to any website using I-Frames. Using Visualizer, users have full control over data security and privacy, since all interaction remain on the client, nothing is shared with the server unless dashboard sharing and collaborative data analysis is enabled. The dashboard supports users in selecting suitable visualizations depending on the selected data fields using a rule-based recommender.

Initially, users need to select the data set they want to investigate. This is either done by selecting local files, defining an URL to a remote data set or by pushing interesting data sets to the dashboard if it is integrated within another website.

After selecting one or multiple data sets, users can perform simple data cleaning and transformation operations. In addition, the dashboard supports automatic data type detection.

As mentioned above, the integration with the Portal is done by including the Visualizer GUI in an I-Frame of the frontend, which enables a link to the tool in a menu at the left side.

More details about the tool implementation can be found in the WP3 deliverables, since this tool is developed in the context of such WP.

8.1.1 COVISE

COVISE stands for COllaborative VIsualization and Simulation Environment. It is an extendible distributed software environment to integrate simulations, post-processing and visualization functionalities in a seamless manner. COVISE Rendering modules support virtual environments ranging from workbenches over powerwalls, curved screens up to full domes or CAVEs. The users can thus analyse their datasets intuitively in a fully immersive environment through state-of-the-art visualization techniques including Volume rendering and fast sphere rendering. COVISE is an open source software. The source code (we use version v2021.1) is available on GitHub (https://github.com/hlrs-vis/covise) or as build for all supported Operating Systems (on https://fs.hlrs.de/projects/covise/support/download/).

More details about the tool implementation can be found in the WP3 deliverables, since this tool is developed in the context of such WP.

8.2 Available APIs

8.2.1 Visualizer

Visualizer is exposed through a web GUI. The actual dashboard has three main areas, the data selection area, the visualization selection area and the actual dashboard showing multiple coordinated views.

Each visualization is configured within its own I-Frame enabling users to create their own visualizations and using their libraries independently from all other visualizations.

Brushing within one visualization updates all other visualizations depending on the configuration for incoming events. Visualizations can either filter or highlight the selected data or remain unchanged depending on the configuration.

We can see a sample dashboard with four different visualizations in Figure 28. On the left side data is divided in to categorical and numerical fields. Depending on which data fields are selected, different visualizations are enabled within the VisPicker on the right side.

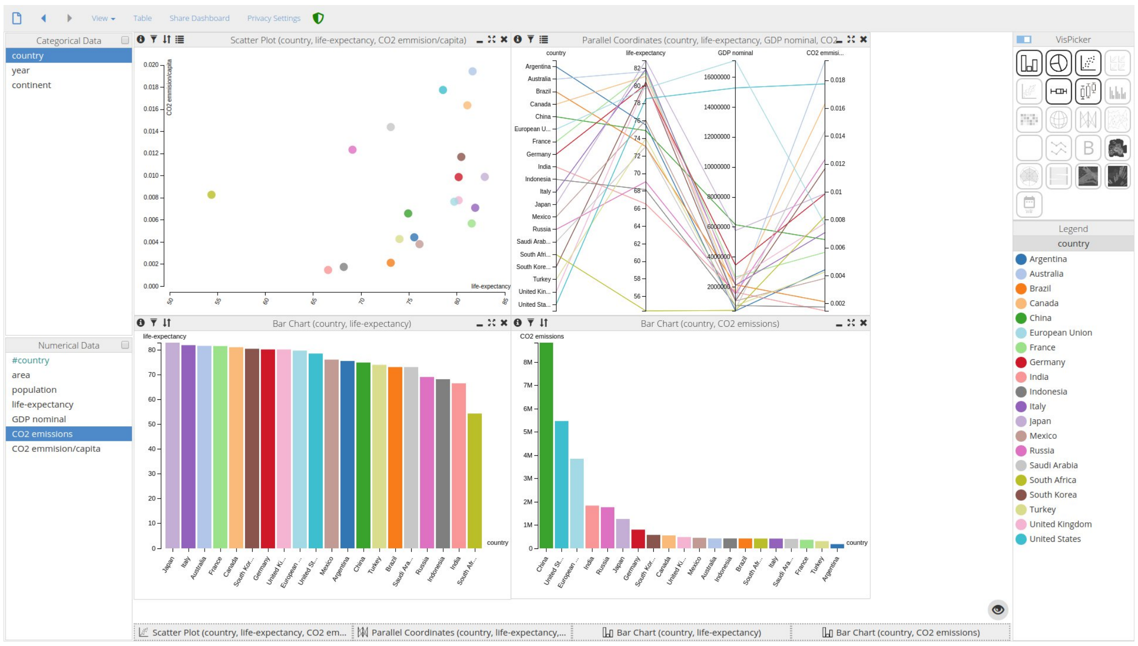


Figure 28: Visualizer showing multiple coordinated visualizations

While configuring a dashboard, the URL is constantly updating containing all dashboard information.

Opening this URL in the browser opens the corresponding dashboard within the browser. If a local data set is used, it needs to be selected beforehand. Therefore, this can be used by the frontend in order to directly list the graphs that could be visualized as an outcome of concrete simulations. Using this URL, dashboards can be easily deployed within any other website.

The following example shows how Visualizer can be integrated to any website using I-Frames:

```
<html>
<head>
  <style>
    iframe {
      width: 800px;
      height: 800px;
      border: 1px solid black;
    }
  </style>
</head>

<body>
  <iframe src="URL" frameborder="0" id="visualizer" name="visualizer"></iframe>
  <script>
    var data = "";
    window.addEventListener("message", function (event) {
      if (event.data === "visualizer-ready") {
        show();
      }
    });
    function show() {
      window.frames.visualizer.postMessage(data, "*");
    }
  </script>
</body>
</html>
```

Here the URL can be set either directly in the I-Frame

```
<iframe src="http://vismobile.know-center.tugraz.at/#..."  frameborder="0" id="visualizer" name="visualizer"></iframe>
```

or by selecting the I-Frame and changing the source

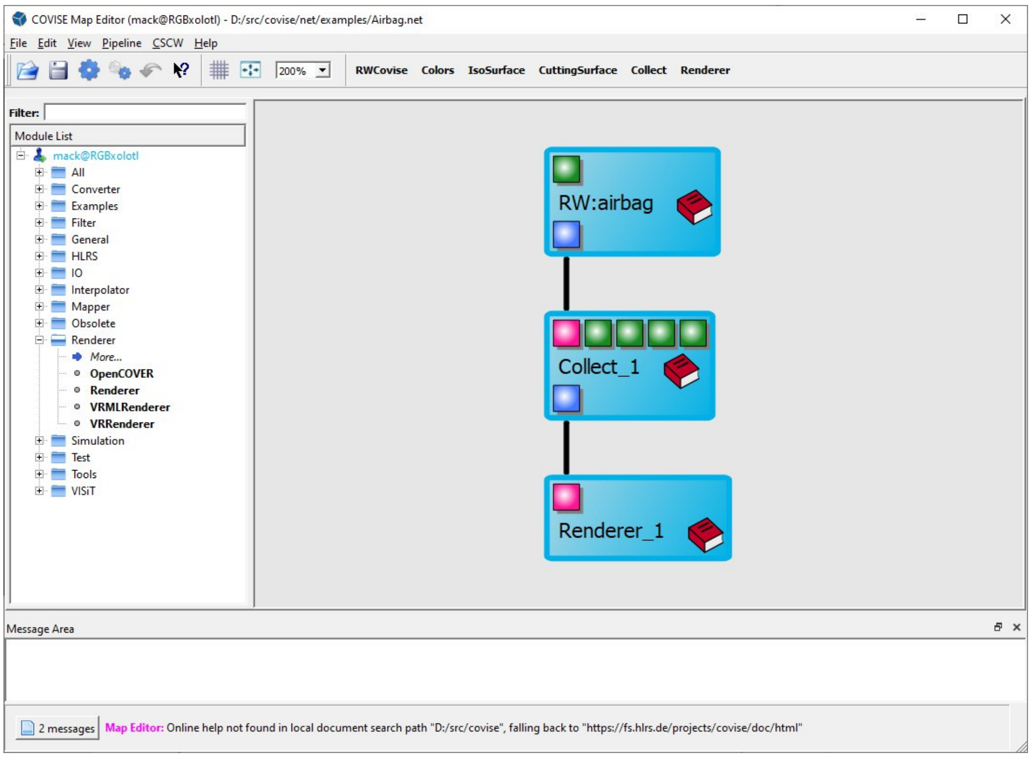
```
document.getElementById("visualizer").src ="http://vismobile.know-center.tugraz.at/#...";
```

The data itself is pushed to the I-Frame using

```
 window.frames.visualizer.postMessage(data, "*");
```

8.2.1 COVISE

The main way to expose COVISE is through a desktop application that acts as client to load and visualize the data locally. We show the so called MapEditor in Figure 29, which is the main part of the user interface. In COVISE an application is divided into several processing steps, which are represented by COVISE modules. This usage is recommended for developers to visualize simulation outputs on a personal computer or with a virtual environment system.



Map Editor: Online help not found in local document search path "D:/src/covise", falling back to "https://fs.hlrs.de/projects/covise/doc/html"

Figure 29: COVISE GUI

In order to provide a simple and user-friendly application of COVISE an extension is being developed. The data processing is executed in a virtual machine and generates a single html file as output, which can be opened easily in a browser or integrated in a website. This workflow allows the user to enjoy an interactive, three-dimensional visualization without installing additional software. The execution of COVISE is done automatically and needs no adjustments.

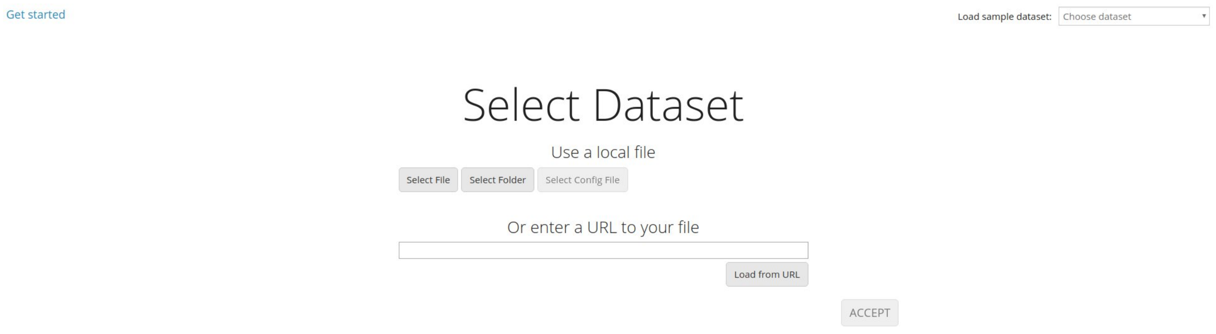
This is the way in which COVISE will be integrated in the Portal, since it is possible to generate the corresponding HTML for visualization, after simulations have been executed.

8.3 Usage and Examples

8.3.1 Visualizer

Visualizer has been introduced and benchmarked in Deliverable 3.2. This subsection provides an overview on how to use the tool.

Figure 30 shows the data selection view of Visualizer enabling users to either select local files or specify the URL to a remote data set.



Get started

Load sample dataset: Choose dataset

Select Dataset

Use a local file

Select File | Select Folder | Select Config File

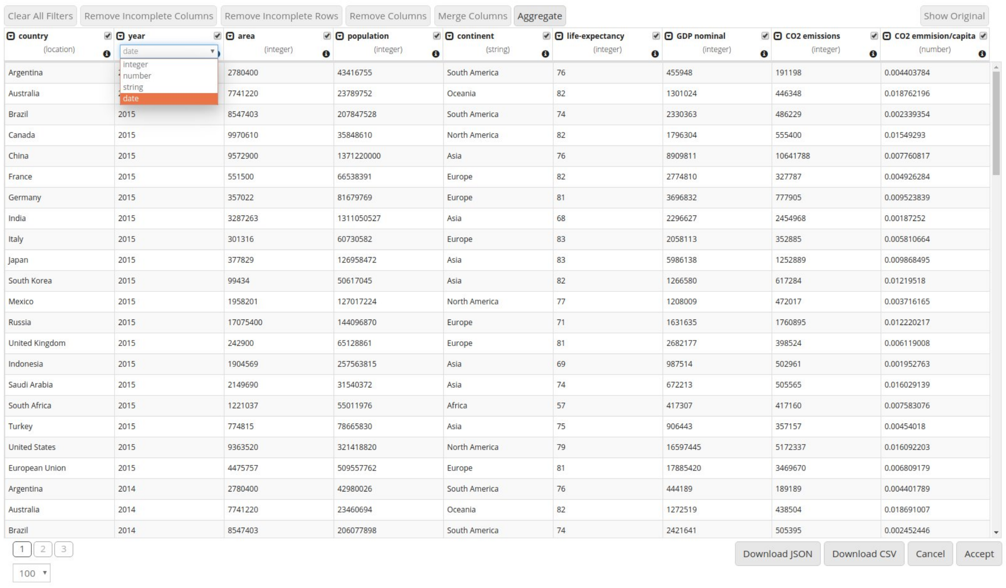
Or enter a URL to your file

Load from URL

ACCEPT

Figure 30: Data selection in Visualizer

After selecting a data set, the table view enables users to investigate their data. Visualizer performs automatic data type detection; however, users can change them on demand as shown in Figure 31. In addition, simple data cleaning and transformation operations can be performed in the table view.



Clear All Filters | Remove Incomplete Columns | Remove Incomplete Rows | Remove Columns | Merge Columns | Aggregate | Show Original

country	year	area	population	continent	life-expectancy	GDP nominal	CO2 emissions	CO2 emmision/capita
(location)	date (integer / number / string / date)	(integer)	(integer)	(string)	(integer)	(integer)	(integer)	(number)
Argentina	[illegible]	2780400	43416755	South America	76	455948	191198	0.004403784
Australia	[illegible]	7741220	23789752	Oceania	82	1301024	446348	0.018762196
Brazil	2015	8547403	207847528	South America	74	2330363	486229	0.002339354
Canada	2015	9970610	35848610	North America	82	1796304	555400	0.01549293
China	2015	9572900	1371220000	Asia	76	8909811	10641788	0.007760817
France	2015	551500	66538391	Europe	82	2774810	327787	0.004926284
Germany	2015	357022	81679769	Europe	81	3696832	777905	0.009523839
India	2015	3287263	1311050527	Asia	68	2296627	2454968	0.00187252
Italy	2015	301316	60730582	Europe	83	2058113	352885	0.005810664
Japan	2015	377829	126958472	Asia	83	5986138	1252889	0.009868495
South Korea	2015	99434	50617045	Asia	82	1266580	617284	0.01219518
Mexico	2015	1958201	127017224	North America	77	1208009	472017	0.003716165
Russia	2015	17075400	144096870	Europe	71	1631635	1760895	0.012220217
United Kingdom	2015	242900	65128861	Europe	81	2682177	398524	0.006119008
Indonesia	2015	1904569	257563815	Asia	69	987514	502961	0.001952763
Saudi Arabia	2015	2149690	31540372	Asia	74	672213	505565	0.016029139
South Africa	2015	1221037	55011976	Africa	57	417307	417160	0.007583076
Turkey	2015	774815	78665830	Asia	75	906443	357157	0.00454018
United States	2015	9363520	321418820	North America	79	16597445	5172337	0.016092203
European Union	2015	4475757	509557762	Europe	81	17885420	3469670	0.006809179
Argentina	2014	2780400	42980026	South America	76	444189	189189	0.004401789
Australia	2014	7741220	23460694	Oceania	82	1272519	438504	0.018691007
Brazil	2014	8547403	206077898	South America	74	2421641	505395	0.002452446

1 2 3

100

Download JSON | Download CSV | Cancel | Accept

Figure 31: Visualizer table view

After confirming the selected data sets, users can proceed to the actual dashboard view and create a dashboard depending on their demands. A sample dashboard is shown in Figure 28.

After creating the dashboard, users can perform interactions within the dashboard. Interactions within one of the visualizations highlight the corresponding fields in all other visualizations, as shown in Figure 32.

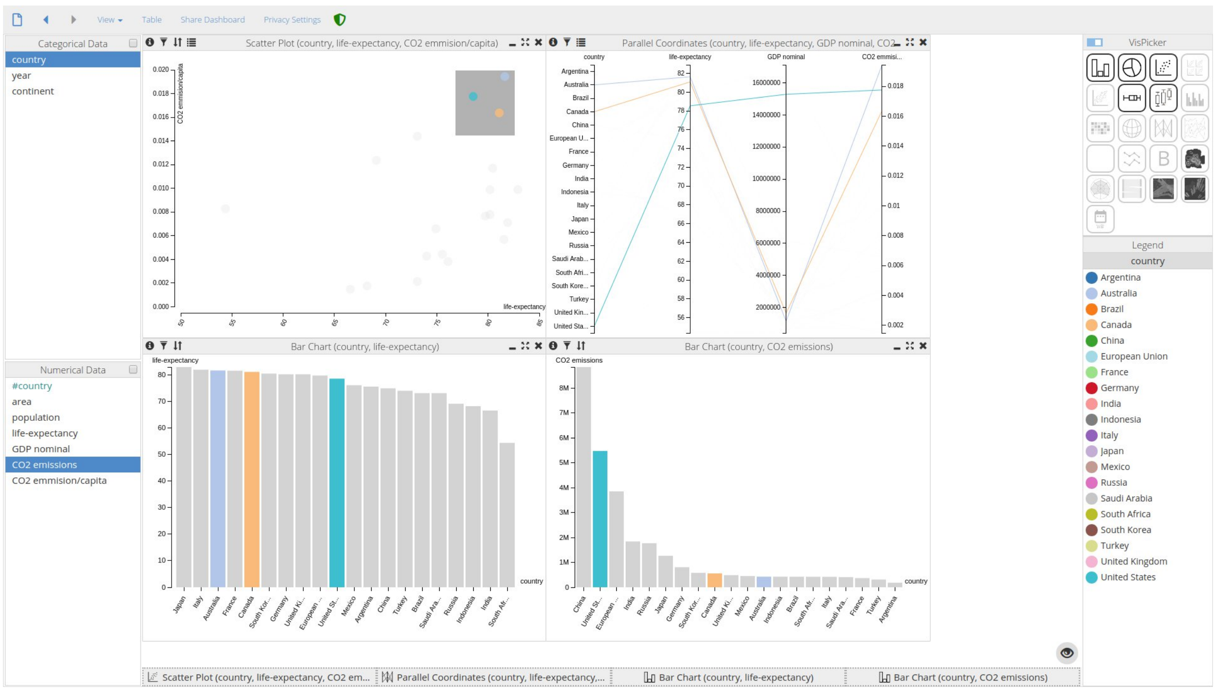


Figure 32: Coordinated multiple views in Visualizer

8.3.1 COVISE

This section covers the usage of the web interface, generated by COVISE for enabling remote visualization. The user interface in the web application is kept simple. The menu allows to switch view modes and to scroll through data for several time steps, as shown in Figure 33. This demonstration is used to show the simulation output of the use case Urban Air Pollution. In this example the city centre of Stuttgart is visualized. The different components like buildings, streets or park areas can be hidden by toggling buttons in the menu.

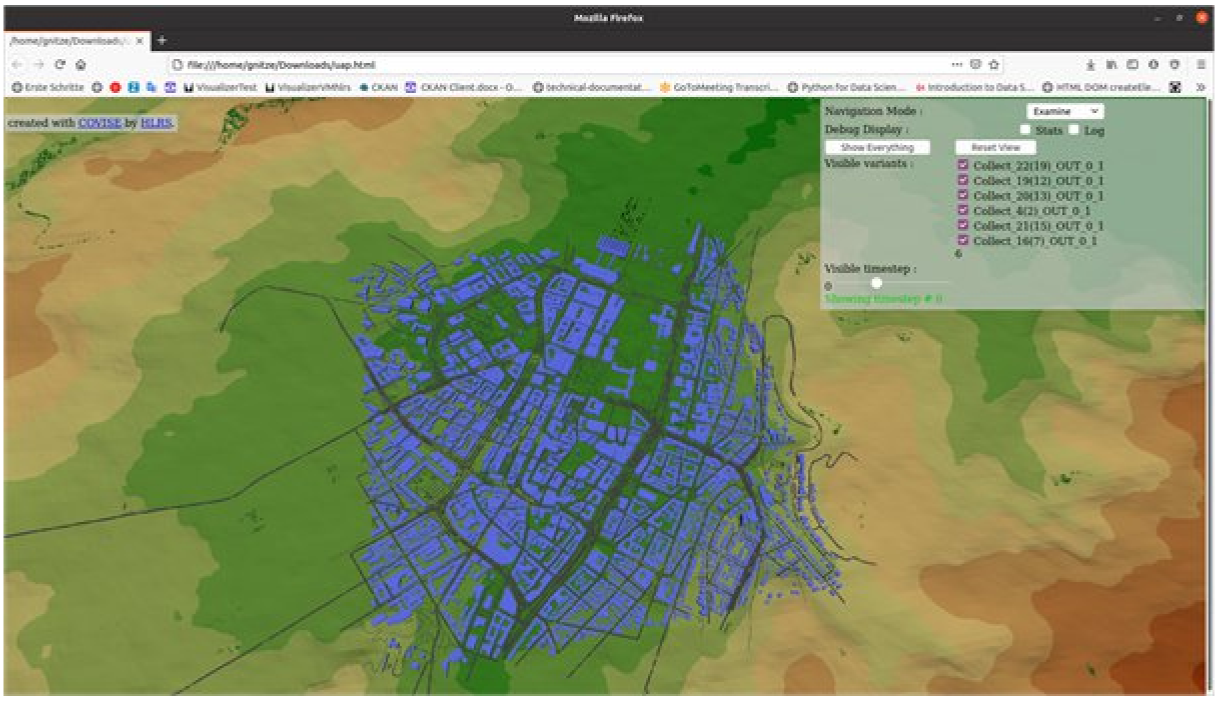


Figure 33: COVISE web application

In the simulation, the airflow and spread of oxides of nitrogen (NOx) in urban areas are observed. These results are visualized in the web application. The concentration of NOx is illustrated by a so-called cutting surface pointing out areas with high rates of NOx, see Figure 34.

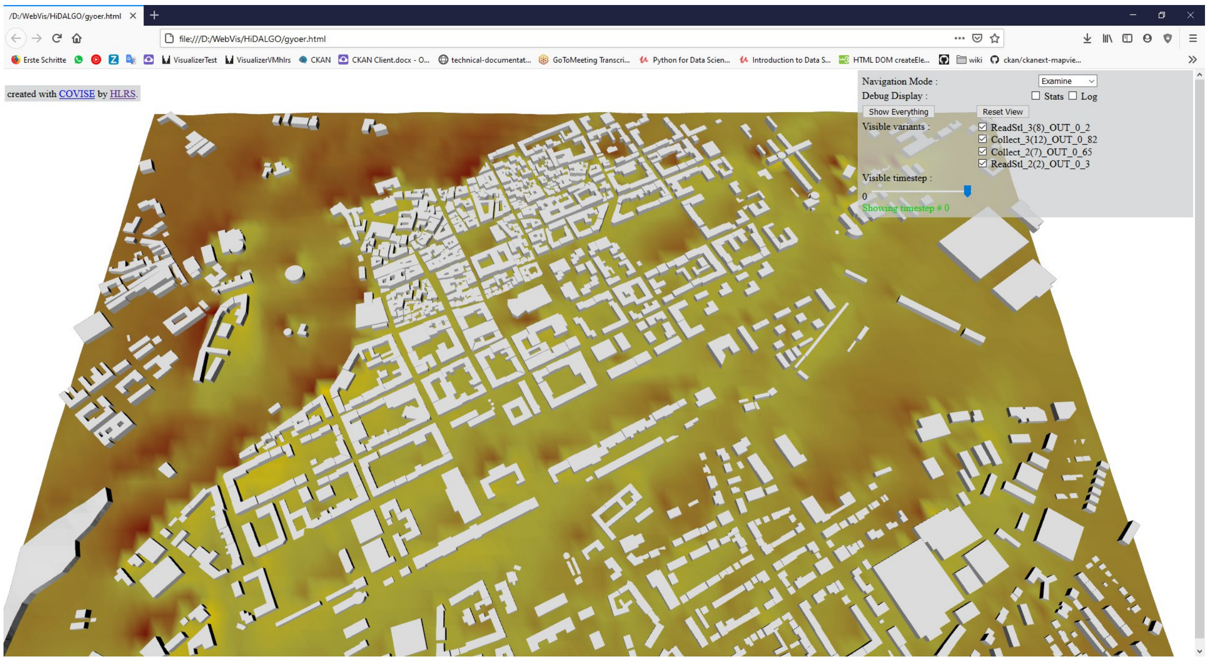


Figure 34: COVISE web application showing nitrogen oxide concentration in urban areas

The airflow is visualized by wind-lines along the propagation direction, see Figure 35.

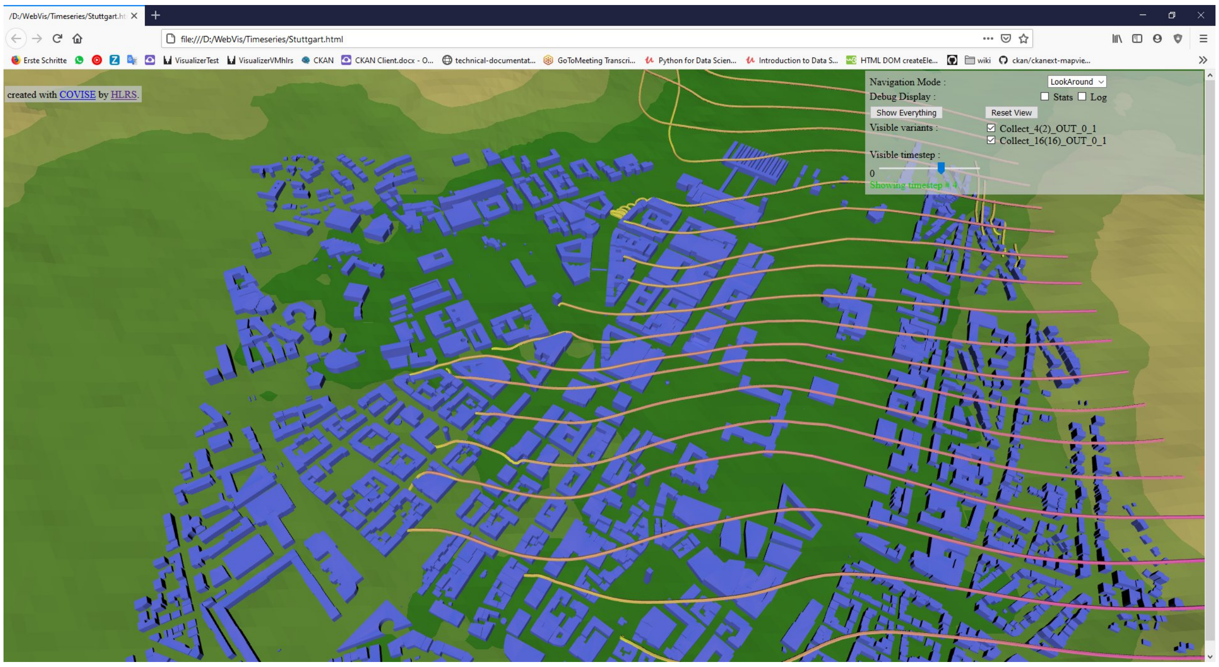


Figure 35: COVISE web application showing airflow in urban areas

9 Support Tools

9.1 Implemented Solution

9.1.1 WiKi

Support tools are detailed in Deliverable D5.4[5] with Zammad and Askbot. Zammad is the support ticketing tool, which is used by customers for raising customer requests by email or Zammad web GUI. Askbot is a community forum, which is used by the community to discuss problems and solutions in a common platform. Zammad and Askbot are available for public access, so we started provisioning customer supports to address the customer queries as defined in D5.4[5]. In this deliverable, Wiki.js[21] is introduced to HiDALGO users for sharing the documentation of HiDALGO toolbox and other non-confidential information in a collaborative editing platform.

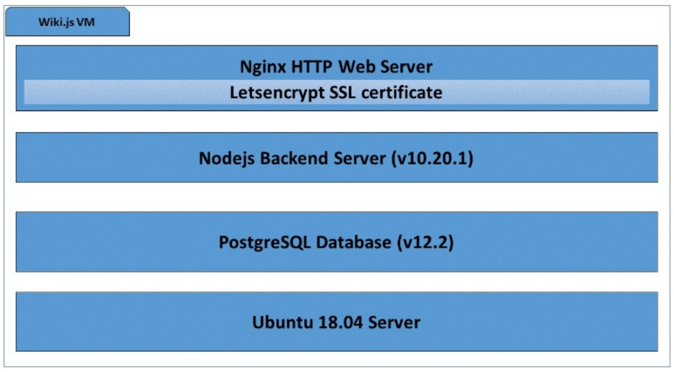

Wiki.js VM
Nginx HTTP Web Server
Letsencrypt SSL certificate
Nodejs Backend Server (v10.20.1)
PostgreSQL Database (v12.2)
Ubuntu 18.04 Server

Figure 36: Wiki.js Software Stack.

Wiki is an internal supporting tool used in HiDALGO to share information within the consortium. It is further planned to integrate the Wiki with the portal to make produced, non-confidential information also available to HiDALGO community. Wiki is the tool developed for supporting collaborative editing, so it would be considered as an identical tool for building a sustainable HiDALGO community by allowing the users to exchange their information in a common platform.

Wiki.js[21] is a well-established open source Wiki solution, and thus it was selected among other competitors due to its simplicity, markdown support and Keycloak OpenID authentication. Specifically, Wiki.js is the only collaborative editing tool that supports Keycloak OpenID for authentication, to the best of our knowledge. As a consequence, it will be straightforward to integrate it with the HiDALGO portal. Wiki.js and Wiki are the terms used interchangeably, and both refer to the installation of HiDALGO Wiki[8].

[8] HiDALGO Wiki weblink - https://hidalgo-wiki.hlrs.de/.

Wiki.js, specifically version 2.2.51, and its installation architecture details are depicted in Figure 36. The editing tool is installed, for now, manually in the production VM to allow internal access within the consortium. Wiki.js supports local authentication, so the user credentials are stored safely inside a PostgreSQL database. Further, Wiki.js manages the authorization by managing groups and page rules to restrict the functionalities and page access at the fine-granularity level. As of now, three groups are created having different access permissions:

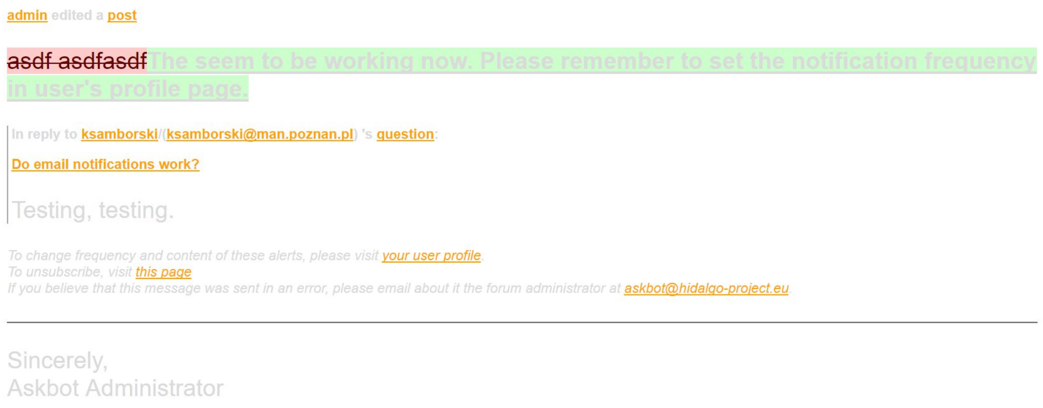
- The “Admin” group is created for administrating the users, wiki pages and changing configurations.
- The “Write” group is allowed to create and edit both public and internal wiki pages.
- The “Guest” group is allowed to create and edit only public wiki pages.

Page rules provide permission at the page level, meaning that this feature allows to restrict the permission at a fine-granular level. Currently, Wiki is setup with the global page-rules for allowing access only to the consortium members assigned to the “Write” group. Wiki.js supports page creation by markdown scripts, which is very common for the documentation and simple to learn by few commands. Wiki.js installation would be automated with Ansible scripts for supporting CI/CD operations in the portal development (cf. Section 3). Moreover, Wiki.js authentication and authorization will be configured to support Keycloak SSO for potential integration with the portal (cf. Section 4). Wiki.js is configured to support self sign-in, but the features of automatic user verification and password reset are not functioning properly in the current version. The improperly functioning features are planned to fix in the upcoming Wiki.js version, so the Wiki.js installation will be updated accordingly to fix those bugs. Public wiki pages will be created to share the documentations of HiDALGO toolbox and common information to ensure public accessibility.

9.1.1 Askbot

In a span of last couple of months several improvements have been implemented into Askbot. Among these are the introduction of email notifications and backup solution. Askbot can now notify users about events like new replies, votes, answers or private messages via emails (instantly or periodically). Notification filters and frequency can be set individually for each user.

By default, instant notifications are triggered by answering questions posted or subscribed by user. Periodic emails contain aggregated notifications for the events that happened during that period.


admin edited a post

~~asdf asdfasdf~~The seem to be working now. Please remember to set the notification frequency in user's profile page.

In reply to ksamborski/(ksamborski@man.poznan.pl) 's question:

Do email notifications work?

Testing, testing.

To change frequency and content of these alerts, please visit your user profile
To unsubscribe, visit this page
If you believe that this message was sent in an error, please email about it the forum administrator at askbot@hidalgo-project.eu

Sincerely,
Askbot Administrator

Figure 37 Example of notification email

Another aspect in which there was some progress is related to the backup procedure. Since Askbot and its database is run in Docker container, external volumes of data were specified. Those volumes contain application database and setting and are accessible from the host OS, outside of the container.

This solution allows copying internal application data to an external location (network file system) and then restore it in case of failure or rollback.

Data restoration is performed as a final part of Ansible script. This ensures that every new installation can immediately provide a recent snapshot of Askbot knowledge base.

Backing up the data is triggered periodically by Jenkins task.

9.2 Available APIs and Usage

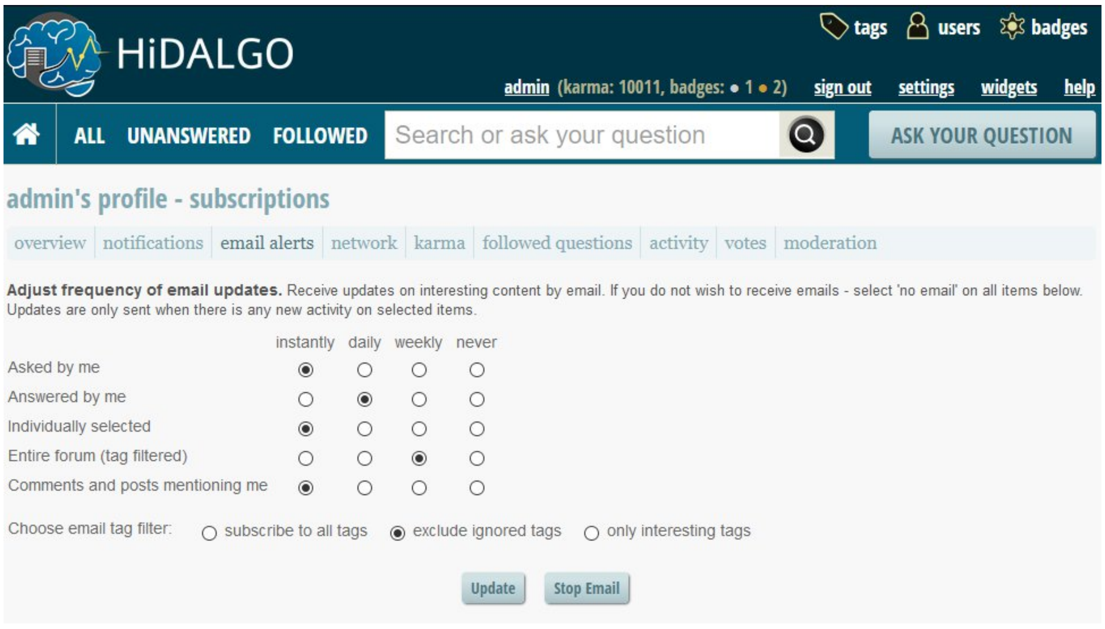
9.2.1 WiKi

Wiki.js is a standalone application with its own GUI, so it is installed as an individual component with the separate domain name (https://hidalgo-wiki.hlrs.de/). Wiki.js provides graphical GUI for registering and log-in in the home page as shown in Figure 39. Users are currently activated manually by admin, due to the problem with automatic user verification by email.

9.2.2 Askbot

Askbot main interfaces and features have been widely explained in the deliverable D5.4[5] and, therefore, they are not replicated here. Since there is a new feature available for enabling email notifications, this section aims at showing how to activate it.

Document name:	D5.6 Second HIDALGO Portal Release and System Operation Report					Page:	58 of 83
Reference:	D5.6	Dissemination:	PU	Version:	1.2	Status:	Final


HiDALGO
tags users badges
admin (karma: 10011, badges: ● 1 ● 2) sign out settings widgets help
ALL UNANSWERED FOLLOWED
Search or ask your question
ASK YOUR QUESTION

admin's profile - subscriptions

overview notifications email alerts network karma followed questions activity votes moderation

Adjust frequency of email updates. Receive updates on interesting content by email. If you do not wish to receive emails - select 'no email' on all items below. Updates are only sent when there is any new activity on selected items.

	instantly	daily	weekly	never
Asked by me	●	○	○	○
Answered by me	○	●	○	○
Individually selected	●	○	○	○
Entire forum (tag filtered)	○	○	●	○
Comments and posts mentioning me	●	○	○	○

Choose email tag filter: subscribe to all tags [✓ exclude ignored tags] only interesting tags

Figure 38 Example email notification settings

Users just need to enter in their profile and select the 'email alerts' tab, where they will be shown the kind of notifications they can receive and how they can be configured (in terms of periodicity). Once the user does the selection, clicking in 'Update' will save the selection and notifications will be activated. All of them can be disabled by clicking in 'Stop Email'.

9.3 Usage and Examples

Askbot and Zammad are tools that were already described in detail in the deliverable D5.4[5]. Therefore, this section is focused on the usage of the Wiki solution. First of all, users have to sign-in, as shown in the figure below.

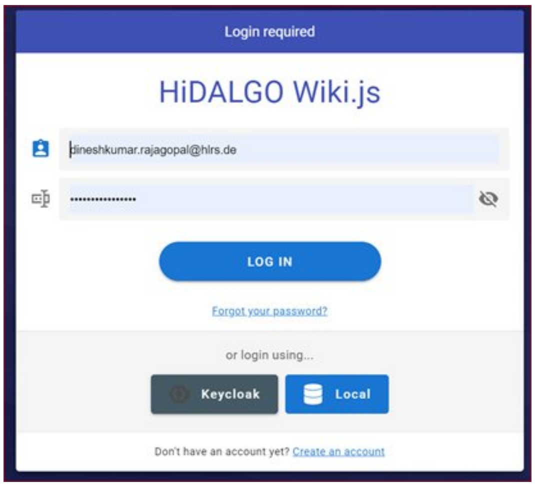
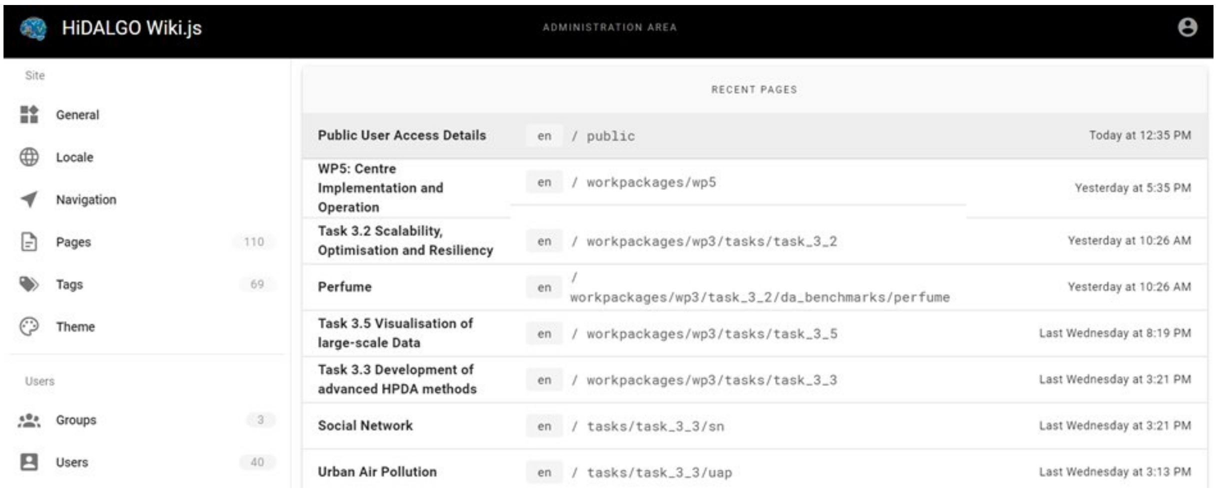


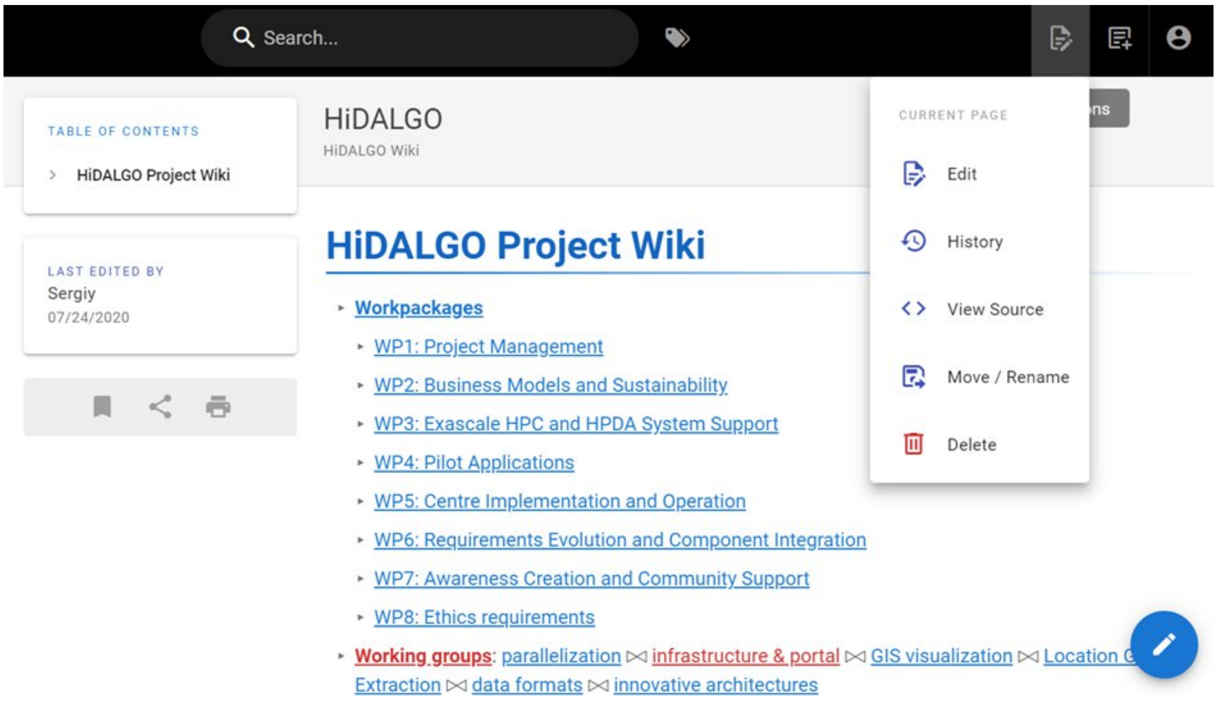
Figure 39: Wiki sign-in and user registration page.

The Admin users can manage the users, groups and page rules as shown in Figure 40[9]. The Write users have complete control to create, edit and delete all the wiki pages in the system, which is shown in Figure 41 and Figure 42.



Public User Access Details	en	/ public	Today at 12:35 PM
WP5: Centre Implementation and Operation	en	/ workpackages/wp5	Yesterday at 5:35 PM
Task 3.2 Scalability, Optimisation and Resiliency	en	/ workpackages/wp3/tasks/task_3_2	Yesterday at 10:26 AM
Perfume	en	/ workpackages/wp3/task_3_2/da_benchmarks/perfume	Yesterday at 10:26 AM
Task 3.5 Visualisation of large-scale Data	en	/ workpackages/wp3/tasks/task_3_5	Last Wednesday at 8:19 PM
Task 3.3 Development of advanced HPDA methods	en	/ workpackages/wp3/tasks/task_3_3	Last Wednesday at 3:21 PM
Social Network	en	/ tasks/task_3_3/sn	Last Wednesday at 3:21 PM
Urban Air Pollution	en	/ tasks/task_3_3/uap	Last Wednesday at 3:13 PM

Figure 40: Admin UI for managing Users, Groups and Page rules.



HiDALGO Project Wiki

- Workpackages
  - WP1: Project Management
  - WP2: Business Models and Sustainability
  - WP3: Exascale HPC and HPDA System Support
  - WP4: Pilot Applications
  - WP5: Centre Implementation and Operation
  - WP6: Requirements Evolution and Component Integration
  - WP7: Awareness Creation and Community Support
  - WP8: Ethics requirements
- Working groups: parallelization ⋈ infrastructure & portal ⋈ GIS visualization ⋈ Location G Extraction ⋈ data formats ⋈ innovative architectures

Figure 41: User can edit an existing page.

[9] HiDALGO Wiki weblink - http://hidalgo-wiki.hlrs.de/

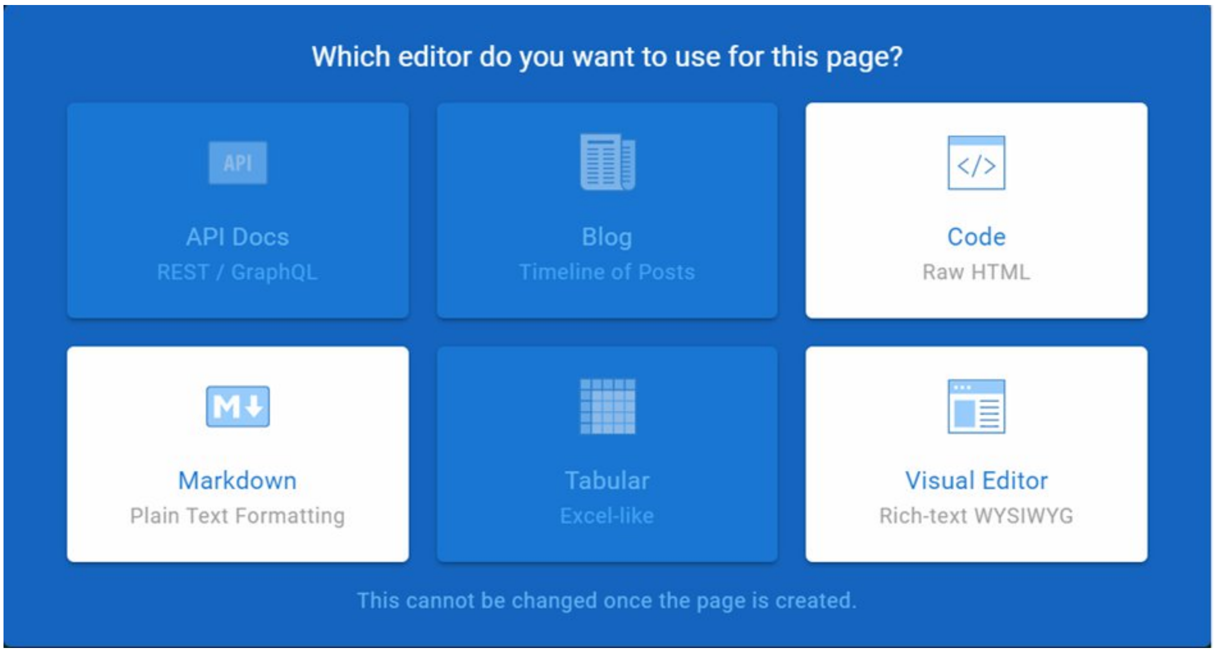


Figure 42: Users can create a new page with different editors. Markdown is used as the default editor.

10Interactive Notebooks

10.1 Implemented Solution

As described in D5.2[1], the HiDALGO portal will allow users to write and execute code for testing, prototyping and visualizing their data. To provide these capabilities, HiDALGO has selected to integrate **Jupyter Notebooks** to its portal.

The Jupyter Notebook is an open source web application maintained by **Project Jupyter**[10] that allows users to create and share documents that contain live code, equations, visualizations and text. These notebook documents are both human-readable documents containing the analysis descriptions and results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis. More specifically:

- Jupyter supports over 40 programming languages, including Python, R, Julia and Scala.
- Users can develop code that produce rich, interactive output, such as HTML, images, video, LATEX and custom MIME types.
- Jupyter can be used for data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, etc. It can leverage Big Data tools, such as Apache Spark, and explore the data with tools such as pandas, TensorFlow and scikit-learn.

ICCS performed a thorough analysis of the different available options for integrating Jupyter Notebooks to the HiDALGO portal. The analysis concluded that in order for the service to be able to scale to high number of users, the best solution was the installation of **JupyterHub**[11], a multi-user version of the notebook designed for companies, classrooms and research labs. Its most prominent features that led to its adoption by the project are the following:

- **Customizability**: JupyterHub can be used to serve a variety of environments, as it supports numerous kernels, including Jupyter Notebook, Jupyter Lab, RStudio and others. Therefore, if at a later point HiDALGO chooses to offer as a service an alternative to Jupyter Notebook, it will be provided by JupyterHub as well.
- **Flexibility**: JupyterHub can be configured with authentication and supports a number of authentication protocols, including OAuth, which is supported by the Single Sign On service of the HiDALGO portal.
- **Scalability**: JupyterHub can scale to tens of thousands of users leveraging modern-day container technology.

[10] http://jupyter.org/
[11] https://jupyter.org/hub

- **Portability**: JupyterHub is open source and can be run on a variety of infrastructures.

To provide the desired scalability and allow numerous users of the HiDALGO portal, it was decided to install JupyterHub on top of a **Kubernetes** cluster. The cluster currently comprises three VMs, the specifications of which are provided in Table 7, but can be easily extended if needed. Node_1 acts both as the kubernetes master, where all the core services run, and as a kubernetes worker node that can spawn application containers; Node_2 and Node_3 act only as worker nodes.

	Node_1 Master	Node_2	Node_3
Number of CPUs	4	2	2
CPU Frequency	2.6 GHz	2.6 GHz	2.6 GHz
RAM	12 GB	8 GB	8 GB
OS	Ubuntu 18.04	Ubuntu 18.04	Ubuntu 18.04
Linux kernel	4.15.0	4.15.0	4.15.0

Table 7 Specifications of the three VMs that host our jupyterhub installation

The kubernetes setup process was facilitated by leveraging the ansible scripts provided by **kubespray**[12], an open source project that offers scripts to create kubernetes clusters as well as to modify them by adding or removing master and/or worker nodes. Then, JupyterHub was installed over the kubernetes cluster using helm, the kubernetes package manager.

As a result, each user executes its own instance of Jupyter Notebook, inside a pod, which is kubernetes' group of docker containers. The data of each user is stored in a **hostpath**, a type of volume supported by kubernetes that mounts a directory from the host node's filesystem into the pod. To enable notebooks to access the same user's data independently of the node where the user happened to log on, the hostpath was setup as an NFS shared mount point common for all the nodes of our cluster.

Component	Version	Repository	License
Docker	19.03.9	https://github.com/docker/docker-ce	Apache License 2.0
Kubernetes Client	1.18.2	https://github.com/kubernetes/kubernetes.git	Apache License 2.0
Kubernetes Server	1.18.2	https://github.com/kubernetes/kubernetes.git	Apache License 2.0
Helm Client	2.17.0	https://github.com/helm/helm	Apache License 2.0
Helm Server	2.17.0	https://github.com/helm/helm	Apache License 2.0

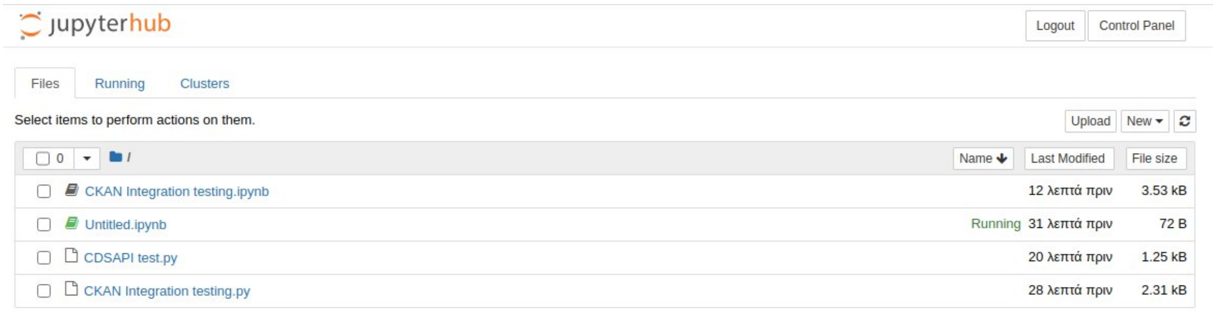
[12] https://github.com/kubernetes-sigs/kubespray

Component	Version	Repository	License
Jupyterhub	0.9.0	https://github.com/jupyterhub/jupyterhub	BSD License

Table 8 Details of all the components that make up the jupyterhub installation

10.2 Available APIs

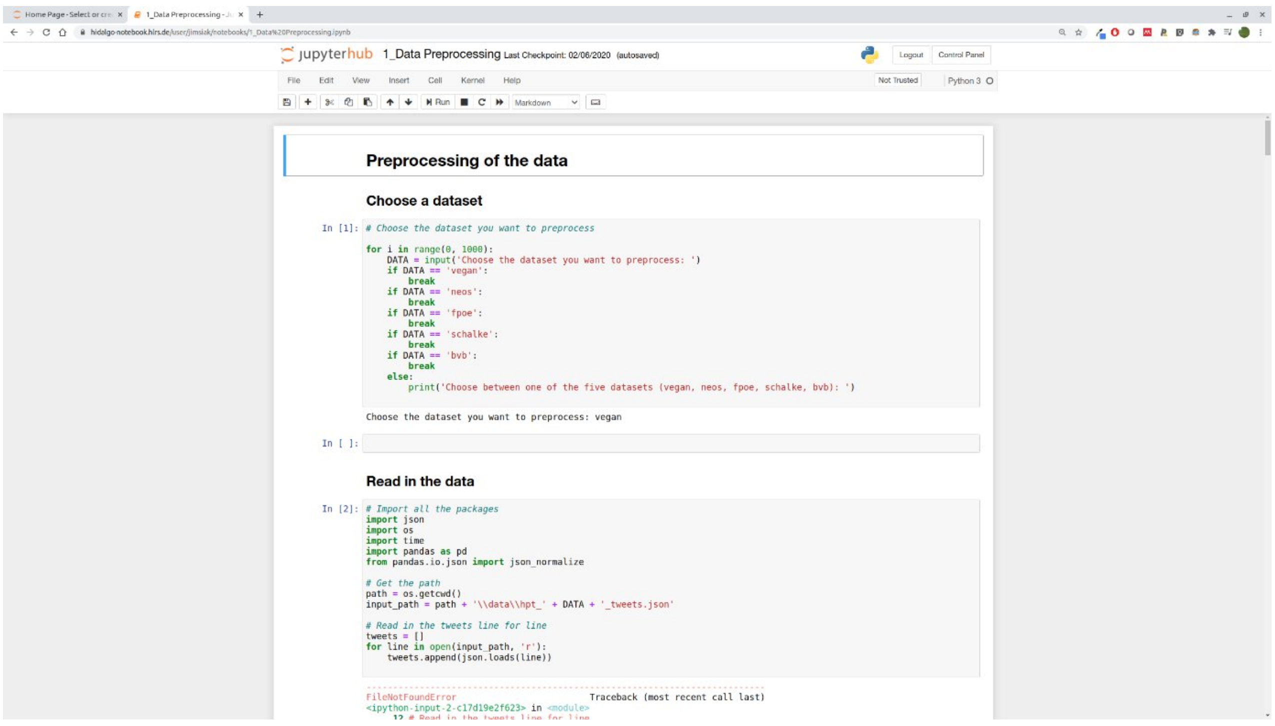
HiDALGO's JupyterHub portal can be currently accessed via https://notebook.hidalgo-project.eu/hub/login. As shown in the following picture (Figure 43), once the user logs in, she is presented with a list of previously saved notebooks and she is able to create a new notebook, upload a notebook stored in her local machine or upload any necessary data.



Select items to perform actions on them.

Name		Last Modified	File size
CKAN Integration testing.ipynb		12 λεπτά πριν	3.53 kB
Untitled.ipynb	Running	31 λεπτά πριν	72 B
CDSAPI test.py		20 λεπτά πριν	1.25 kB
CKAN Integration testing.py		28 λεπτά πριν	2.31 kB

Figure 43 HiDALGO's JupyterHub GUI


Home Page - Select or cre
1_Data Preprocessing - Ju
hidalgo-notebook.hlrs.de/user/jimsiak/notebooks/1_Data%20Preprocessing.ipynb
jupyterhub
1_Data Preprocessing Last Checkpoint: 02/06/2020 (autosaved)
Logout
Control Panel
File
Edit
View
Insert
Cell
Kernel
Help
Not Trusted
Python 3
Run
Markdown

Preprocessing of the data

Choose a dataset

```
In [1]: # Choose the dataset you want to preprocess

        for i in range(0, 1000):
            DATA = input('Choose the dataset you want to preprocess: ')
            if DATA == 'vegan':
                break
            if DATA == 'neos':
                break
            if DATA == 'fpoe':
                break
            if DATA == 'schalke':
                break
            if DATA == 'bvb':
                break
            else:
                print('Choose between one of the five datasets (vegan, neos, fpoe, schalke, bvb): ')

        Choose the dataset you want to preprocess: vegan

In [ ]:
```

Read in the data

```
In [2]: # Import all the packages
        import json
        import os
        import time
        import pandas as pd
        from pandas.io.json import json_normalize

        # Get the path
        path = os.getcwd()
        input_path = path + '\\data\\hpt_' + DATA + '_tweets.json'

        # Read in the tweets line for line
        tweets = []
        for line in open(input_path, 'r'):
            tweets.append(json.loads(line))

        ---------------------------------------------------------------------------
        FileNotFoundError                         Traceback (most recent call last)
        <ipython-input-2-c17d19e2f623> in <module>
```

Figure 44 GUI for viewing, modifying and executing a notebook

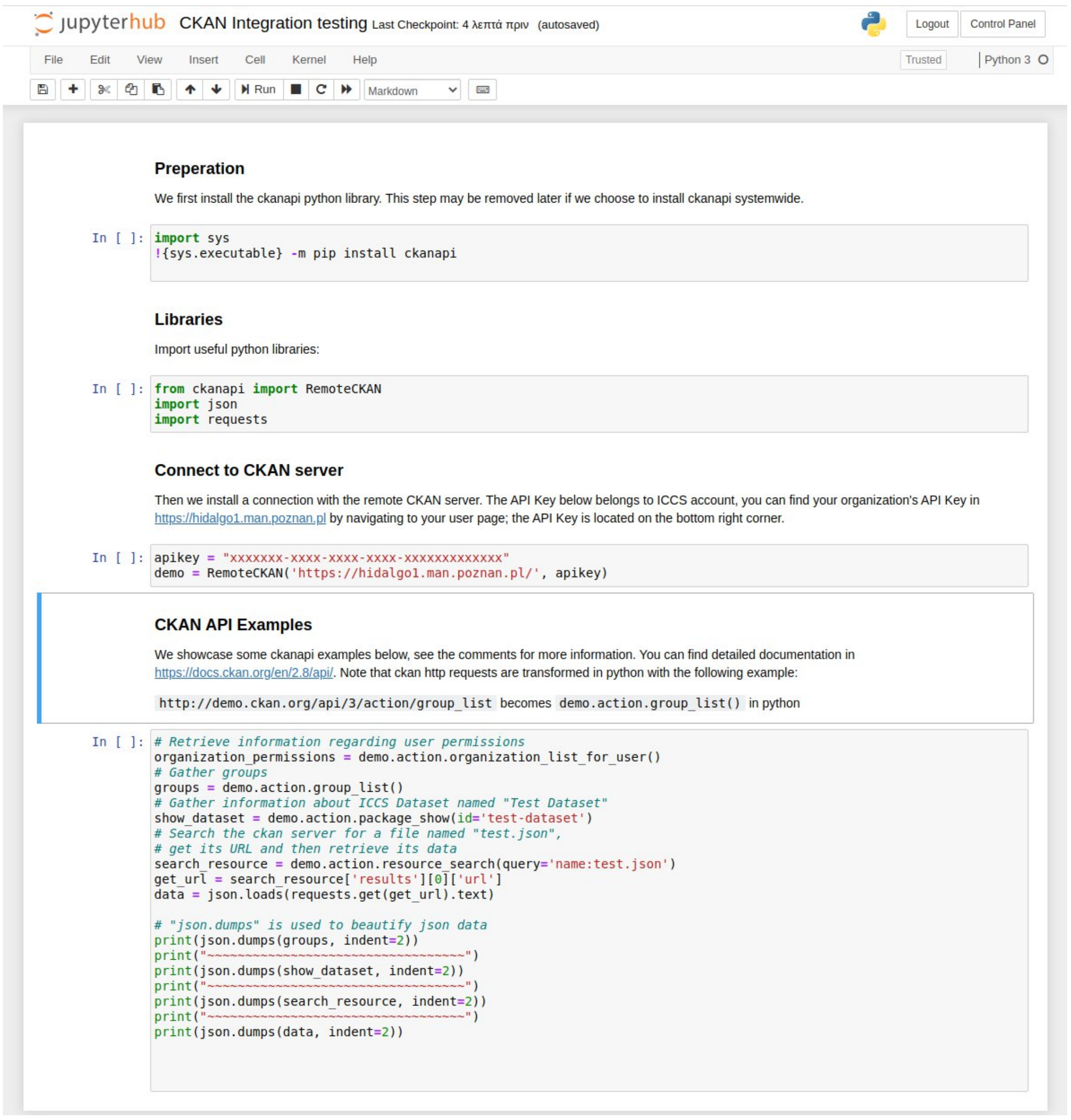
When the user selects a notebook to open, a new tab opens in her browser showing the contents of the notebook. As shown in the figure that follows (Figure 44), the user has then option to modify, execute, interrupt, etc. the kernel.

JupyterHub has been integrated with the HiDALGO CKAN repository to allow the portal users to access and manipulate their data stored in the repository. To enable this we leverage **ckanapi**[13], a python module that can be used in a Python 2 or Python 3 application in order to utilise the **CKAN Action API**[14].

10.3 Usage and Examples

The next figure (Figure 45) provides an example that shows how a user can access data stored in her folder in the HiDALGO CKAN repository.

[13] https://github.com/ckan/ckanapi
[14] https://docs.ckan.org/en/latest/api/index.html#action-api-reference



jupyterhub CKAN Integration testing Last Checkpoint: 4 λεπτά πριν (autosaved)

Logout Control Panel

File Edit View Insert Cell Kernel Help Trusted Python 3

Preperation

We first install the ckanapi python library. This step may be removed later if we choose to install ckanapi systemwide.

```
In [ ]: import sys
        !{sys.executable} -m pip install ckanapi
```

Libraries

Import useful python libraries:

```
In [ ]: from ckanapi import RemoteCKAN
        import json
        import requests
```

Connect to CKAN server

Then we install a connection with the remote CKAN server. The API Key below belongs to ICCS account, you can find your organization's API Key in https://hidalgo1.man.poznan.pl by navigating to your user page; the API Key is located on the bottom right corner.

```
In [ ]: apikey = "xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx"
        demo = RemoteCKAN('https://hidalgo1.man.poznan.pl/', apikey)
```

CKAN API Examples

We showcase some ckanapi examples below, see the comments for more information. You can find detailed documentation in https://docs.ckan.org/en/2.8/api/. Note that ckan http requests are transformed in python with the following example:

`http://demo.ckan.org/api/3/action/group_list` becomes `demo.action.group_list()` in python

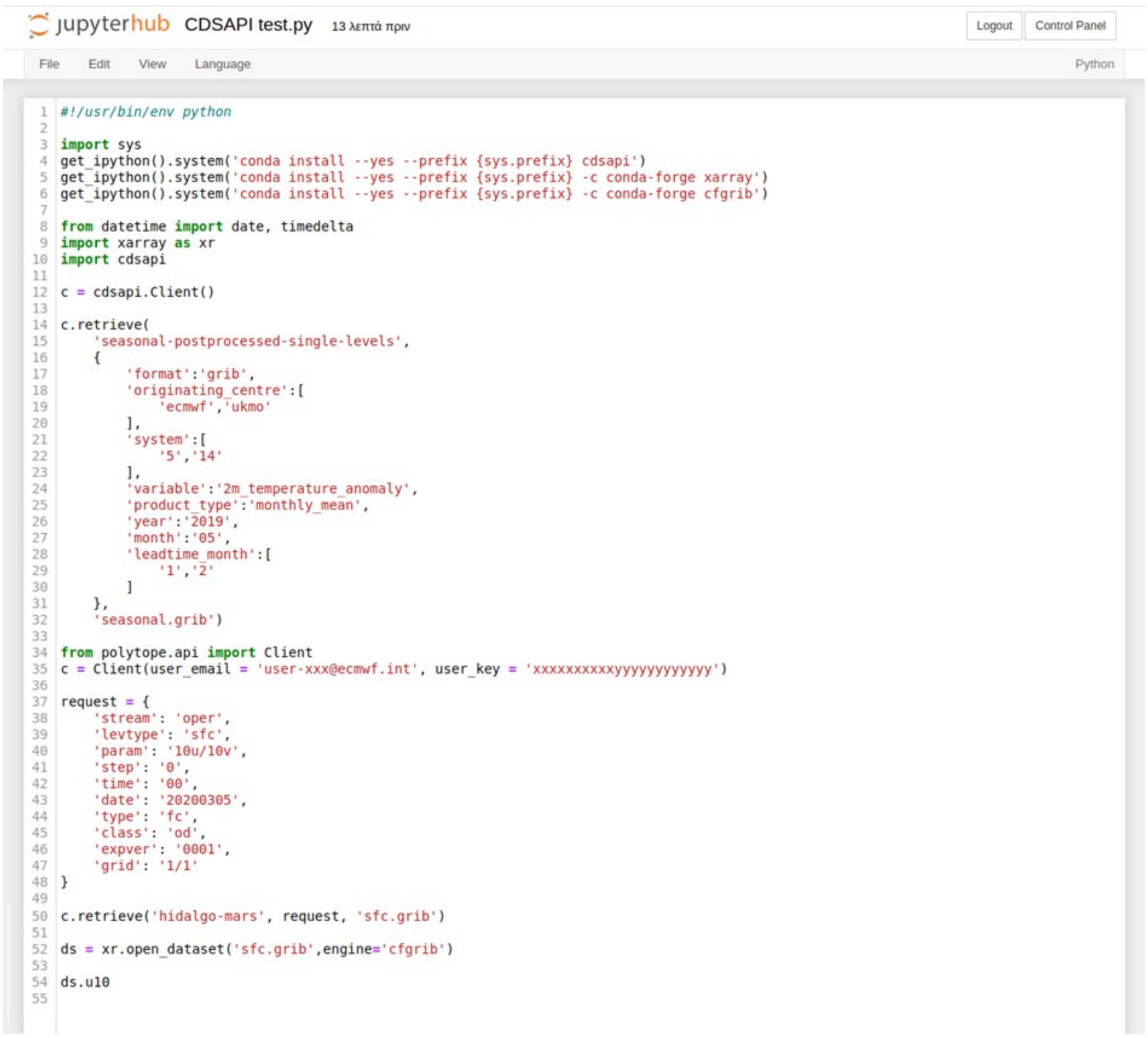
```
In [ ]: # Retrieve information regarding user permissions
        organization_permissions = demo.action.organization_list_for_user()
        # Gather groups
        groups = demo.action.group_list()
        # Gather information about ICCS Dataset named "Test Dataset"
        show_dataset = demo.action.package_show(id='test-dataset')
        # Search the ckan server for a file named "test.json",
        # get its URL and then retrieve its data
        search_resource = demo.action.resource_search(query='name:test.json')
        get_url = search_resource['results'][0]['url']
        data = json.loads(requests.get(get_url).text)

        # "json.dumps" is used to beautify json data
        print(json.dumps(groups, indent=2))
        print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
        print(json.dumps(show_dataset, indent=2))
        print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
        print(json.dumps(search_resource, indent=2))
        print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
        print(json.dumps(data, indent=2))
```

Figure 45 Example of accessing HiDALGO CKAN repository through HiDALGO's JupyterHub

Finally, the next figure (Figure 46) presents an example notebook for retrieving and working with weather and climate data stored in the Climate Data Store[15] and ECMWF MARS Archive[16]. After installing the libraries for retrieving (Climate Data Store API (CDS API)[17] and Weather and Climate Data API (WCDA)) and processing the data (xarray[18] and cfgrib[19]) using conda, the user can manipulate the data to prepare them for the use in the applications.

[15] https://cds.climate.copernicus.eu/
[16] https://confluence.ecmwf.int/display/UDOC/MARS+user+documentation
[17] https://cds.climate.copernicus.eu/api-how-to
[18] http://xarray.pydata.org/en/stable/
[19] https://github.com/ecmwf/cfgrib



```python
#!/usr/bin/env python

import sys
get_ipython().system('conda install --yes --prefix {sys.prefix} cdsapi')
get_ipython().system('conda install --yes --prefix {sys.prefix} -c conda-forge xarray')
get_ipython().system('conda install --yes --prefix {sys.prefix} -c conda-forge cfgrib')

from datetime import date, timedelta
import xarray as xr
import cdsapi

c = cdsapi.Client()

c.retrieve(
    'seasonal-postprocessed-single-levels',
    {
        'format':'grib',
        'originating_centre':[
            'ecmwf','ukmo'
        ],
        'system':[
            '5','14'
        ],
        'variable':'2m_temperature_anomaly',
        'product_type':'monthly_mean',
        'year':'2019',
        'month':'05',
        'leadtime_month':[
            '1','2'
        ]
    },
    'seasonal.grib')

from polytope.api import Client
c = Client(user_email = 'user-xxx@ecmwf.int', user_key = 'xxxxxxxxxxyyyyyyyyyyyy')

request = {
    'stream': 'oper',
    'levtype': 'sfc',
    'param': '10u/10v',
    'step': '0',
    'time': '00',
    'date': '20200305',
    'type': 'fc',
    'class': 'od',
    'expver': '0001',
    'grid': '1/1'
}

c.retrieve('hidalgo-mars', request, 'sfc.grib')

ds = xr.open_dataset('sfc.grib',engine='cfgrib')

ds.u10
```

Figure 46 Using JupyterHub to retrieve and process weather and climate data

11 Frontend and Applications GUI

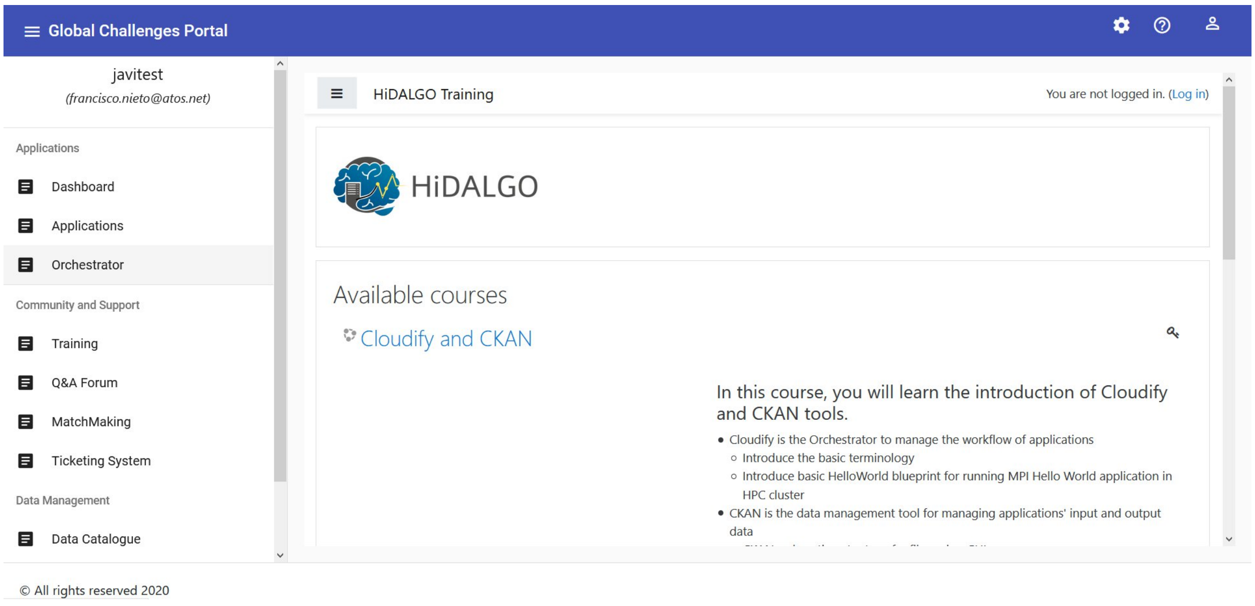
11.1 Implemented Solution

The Frontend part has suffered several modifications. First of all, it has improved the way in which applications can be installed and executed. On the other hand, we have enabled the integration of the different components of the Portal in a one-stop-shop, where users can access all the features easily.

11.1.1 Integration of the Portal Components

In the first version of the Portal, we had several services available, but they were deployed in a quite isolated way, so it required to any user to know the concrete URLs to access them, with no connection between them.

In this version, the effort has been focused on the integration of these separated components, so now it is possible to access only the frontend and get access to the rest of the available elements, such as the training (Moodle), Q&A (Askbot), etc.



In this course, you will learn the introduction of Cloudify and CKAN tools.

- Cloudify is the Orchestrator to manage the workflow of applications
  - Introduce the basic terminology
  - Introduce basic HelloWorld blueprint for running MPI Hello World application in HPC cluster
- CKAN is the data management tool for managing applications' input and output data

Figure 47 HiDALGO Portal with the Training tool

There are two different ways in which such integration has been done:

- Usage of iFrames
- Open new tabs

Due to the complexity of the applications to be integrated, using Javascript was not the solution in order to integrate the GUIs of the other components. The Frontend is implemented in Angular and, therefore, the way to do it was to include each component as a 'module' in Angular. The list of components supported in v2 is:

- Training (Moodle)
- Q&A (Askbot)
- Ticketing System (Zammad)
- Orchestration (Croupier)
- Data Catalogue (CKAN)
- Visualizer

Only in the case of Croupier and Zammad, since their GUIs do not allow iFrames (due to the X-Frame option embedded in their code), the solution was to include some page with information about the tool and a button to open the tool in a new tab.

The Frontend includes a button in the top left that shows a menu, giving access to the features, organized according to the type of functionality they provide (Applications, Community and Support, Data Management and Visualization).

Although this is a valid solution for integrating the tools, sometimes the look and feel is not the best one. Therefore, the next version will re-model the frontend, in such a way that the access will keep simple while the Frontend improves the aspect of the whole Portal.

Additionally, the support to the Jupyter Notebooks is ongoing, and more tools will be added, such as the Polytope interface (as one of the data management tools) and the web version of COVISE (as one of the visualization tools).

11.1.2 Execution of Workflows

In order to ease the way in which users can execute their simulations, the Frontend includes a more elaborated interface for managing the applications and their execution. The figure below shows how different parts of the Portal interact in order to make this to happen.

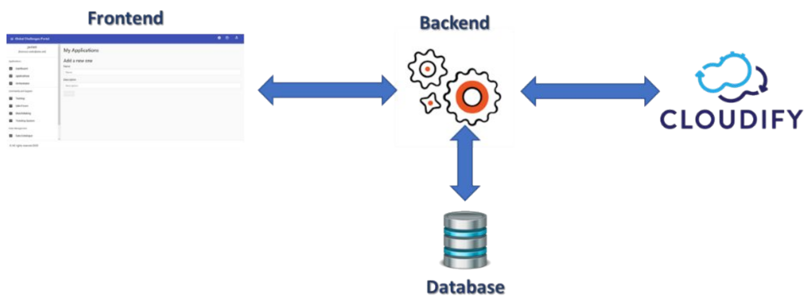


Figure 48 Implemented structure for execution workflows

Currently, the Frontend provides two options, one for adding new applications and another one to execute an existing application. In the first case, the user just needs to provide an application name, a description and a zip file with the blueprint that represents the application. Then, the Frontend communicates to the Backend about the new application and the Backend performs two actions: i) it stores the provided information in the local database and ii) invokes the Croupier REST API in order to install the new application.

As for the execution activity, the Frontend retrieves the information from the blueprint, identifying the input parameters required, creating a form with the list of parameters. Then, a user will be able to fill in these parameters and 'Run' the application.

The Backend receives the list of parameters, generates a YAML file with the inputs and sends this input file to the Orchestrator, as a first step to create an instance of the application (through the REST API). Then, invokes again the REST API, but this time for executing the instance it just created. As soon as some information is generated by the Orchestrator, the Backend provides this information to the Frontend, that will show it.

It is important to mention that all these operations are supervised by Keycloak, since both the Frontend and Backend check the security tokens against Keycloak.

While the Frontend is a web application implemented in Angular, the Backend is a Django application implemented in Python3, including multiple libraries, like cloudify-common, djangorestframework, mozilla-django-oidc, pyOpenSSL and PyYAML. An additional work has been done in order to deploy the Backend using uWSGI as web server and to dockerize the whole solution, although, since it has shown to be more complex than expected, we may decide to change to Gunicorn in the next release.

11.1.3 User Matchmaking

Matchmaking functionality and API are not changed from D5.3[2]. Matchmaking is implemented by using python2.7 and it is no longer supported, so it is planned to port to python3.6. Matchmaking algorithm is currently based on the Geometric Mean algorithm to calculate the user's relation, and it is planned to enhance with the machine learning algorithms (clustering) to group users based on the users' profile and preference information.

11.1.4 Usability, Users' Feedback and Monitoring

11.1.4.1 Testing Usability

There are some tools and services that can support developers for understanding how the users make use of the web applications deployed, generating heat maps (indicating the main areas where users move the mouse and click), generating useful questionnaires, measuring certain aspects related to performance (i.e. loading times), etc. Since these tools and services may provide valuable information to the consortium, we have analysed those that are available, in order to understand how they can be applied in the context of the project. The main issue detected, in general, is that the vast majority of tools and services are not for free (requiring some monthly fee), although most of them provide limited periods of free trials (usually, between 14 and 30 days).

Some of the widely used tools are Qualaroo[20], Crazyegg[21] and Chalkmark[22], which are not for free. While the first one is quite focused on collecting feedback from users (i.e. showing up some simple questions when users do certain actions), the other two keep track of the users' activity and where they click. Qualaroo can collect feedback with small text boxes (i.e. to collect a quick answer when a user cancels an action like a purchase) and other elements like simple satisfaction questions to select how many stars to give. Then that information is used together with IBM Watson to perform sentiment analysis depending on the answers received.

On the other hand, Crazyegg and Chalkmark are similar tools that monitor users' activities, generating snapshots, heatmaps, click density grids and recordings of individual sessions that can be analysed in order to see if users are following the expected path to complete certain tasks or if they are getting stuck in some part of the interface. This information can be used to adapt the interfaces, making easier for users to complete the expected tasks.

There are also other tools that can be used for free (at least, for a good set of basic features). From the analytical perspective, WebPageTest[23] allows to carry out some tests to the website, focused on its loading time and performance, so it is possible to figure out how to improve it (i.e. we can simulate the access to the website from different locations). Google Lighthouse[24] has some similarities, carrying out audits about the website performance and even providing information about SEO-related aspects. Additionally, mouseflow[25] is a tool like Crazyegg and Chalkmark, that generates heatmaps and scrolling statistics, so we can analyse users' behaviour (it is for free for one site, 500 recordings per month).

Another interesting tool is TAW[26], that analyses the accessibility of a website. It takes into account WCAG 2.0[22] in order to report to what extent people with different capabilities can access to the website content as expected. It is able to detect some problems automatically and it points out some other issues to be checked manually by the developer.

In the case of HiDALGO, we see interesting to use, at least, one tool for analysing the performance and another tool for generating heatmaps, since they are complementary.

[20] https://qualaroo.com/
[21] https://www.crazyegg.com/
[22] https://www.optimalworkshop.com/chalkmark/
[23] https://www.webpagetest.org/
[24] https://developers.google.com/web/tools/lighthouse
[25] https://mouseflow.com/
[26] https://www.tawdis.net

11.1.4.2 Collecting Users' Feedback and Monitoring their Actions

Since the consortium received some comments about having a more user-centric Portal, that will also retrieve information about its usage and will collect users' feedback. After analysing several options, we decided to implement a system that will monitor the users' actions anonymously and that will also allow users to provide some simple feedback about their satisfaction.

Solutions like Google Analytics can be integrated, but they require the users to accept certain cookies that they might not want to accept and, since it requires some effort to integrate anyway, we decided to include some code in the frontend that will interact with the Backend in order to store usage information in the database. Each time a user starts a session, a hash code will be generated for the session, keeping track of the main actions done (clicks done by the user and paths followed to reach some functionality). We will include the possibility to disable this option for operational environments, since we may expect a small drop in performance.

Thanks to such information, it will be possible to retrieve data and group it, in such a way it will be possible for us to understand the actions that users carry out most of the times, how they execute certain actions and if there are features that are not used in general. This information will be used to analyse the usability (we can see if they are efficient doing certain actions and how much time it takes to them), to reorganize the Portal (it might be necessary to give more visibility to certain features) and even to re-factor some code if it may require higher scalability (because it is widely used).

Finally, as mentioned before, we have planned to include a simple questionnaire, so they can provide some feedback. We will ask them to indicate the services/features that they use to make use of, as well as their level of satisfaction with each feature and with the Portal in general.

11.2 Available APIs

We can consider that the backend has two interfaces. First of all, the backend managing the interaction with other components and keeping the database information provides a REST API that is accessed by the frontend, in order to perform the complex operations. Such API has been created with the Django-Rest-Framework (DRF) and, although there are several modifications in the code that implements the services, we have maintained the same API we defined in D5.3[2], with the exception of minor additions. Therefore, section 5.2 of D5.3[2] contains the description of the API served in this version. It is important to bear in mind that now the REST API is fully integrated with Keycloak and it can be only invoked using a security token. The new API only contains two methods related to the reset of the backend.

REST API endpoints	API Description
`apps/reset`	Used to reset the applications stored in the database (GET method).
`instances/reset`	Used to remove the instances created for all the applications used from the Frontend (GET method).

Table 9 New Methods of the Backend REST API

On the other hand, there have been modifications in terms of the web-based GUI. Although we still maintain the Cloudify Web Console (as shown in D5.3[2]), there is a new version of the frontend that allows for a different way to manage the applications. Moreover, it puts together the rest of components under the same interface, as a way to provide a one-stop-shop.

The GUI consist of a top part with the HiDALGO logo and information from the connected user. Then, there is a button at the top left that shows the menu with all the options that can be selected. Once the user clicks, the corresponding component is loaded.

In those cases in which it is not possible to embed the component interface, a new tab is created. The next subsection shows how to use some of the new features of the frontend.

11.3 Usage and Examples

11.3.1 Using the Backend API

Since the main methods of the REST API have not changed, it is possible to invoke them as before. In order to be compliant with the security requirements, it is necessary to get the security token from the Keycloak first:

```
RESULT=`curl -k --data
"grant_type=password&client_id=curl&client_secret=xxback
endkeyxx&username=xxusernamexx&password=xxuserpasswordxx
" https://prunus-
212.man.poznan.pl/auth/realms/Hidalgo/protocol/openid-
connect/token`
TOKEN=`echo $RESULT | sed
's/.*access_token":"\([^"]*\).*/\1/'`
```

Then, it is possible to send requests in order to list the applications and instances available in Croupier:

```
curl -H "Authorization: Bearer $TOKEN"
http://x.x.x.x:9000/apps/
curl -H "Authorization: Bearer $TOKEN"
http://x.x.x.x:9000/instances/
```

It is also possible to add new applications and even to create new instances of an existing application, like in the following example:

```
curl --data-urlencode app=FACS-blueprint --data-
urlencode name=xxappnamexx --data-urlencode
inputs_file@my-blueprint-inputs.yaml -H "Authorization:
Bearer $TOKEN" -X POST http://x.x.x.x:9000/instances/
```

11.3.2 The Frontend Web GUI

The frontend has now several features available just doing a few clicks. In the case of the applications management, it is possible to access to the features of adding a new application and executing an existing one by using the menu and selecting the options under 'Applications'. In the case of adding a new application, for instance, we will be requested to provide the name of the application and a description. Then, we will see the option to upload the zip file with the blueprint.

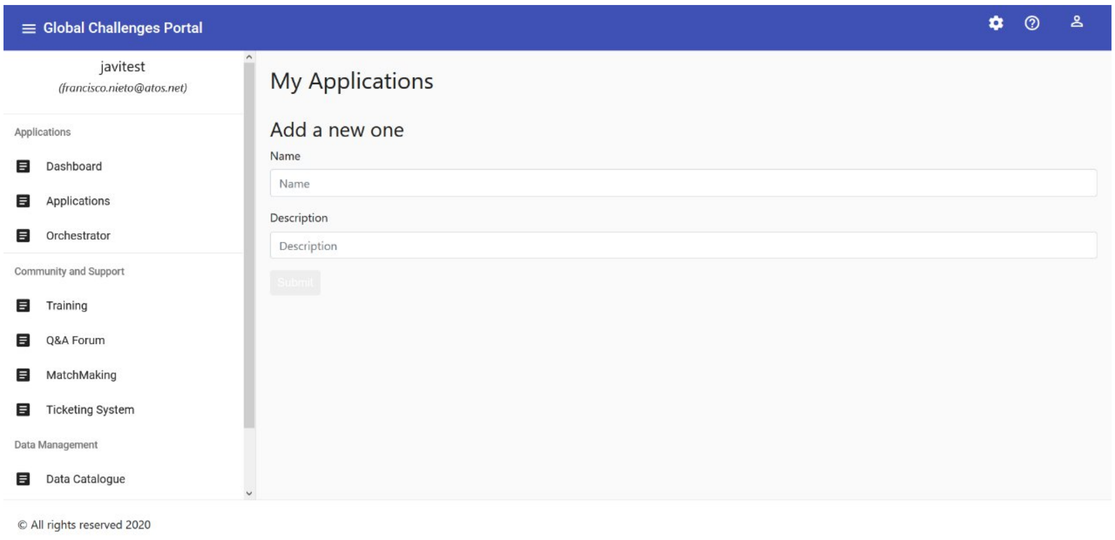
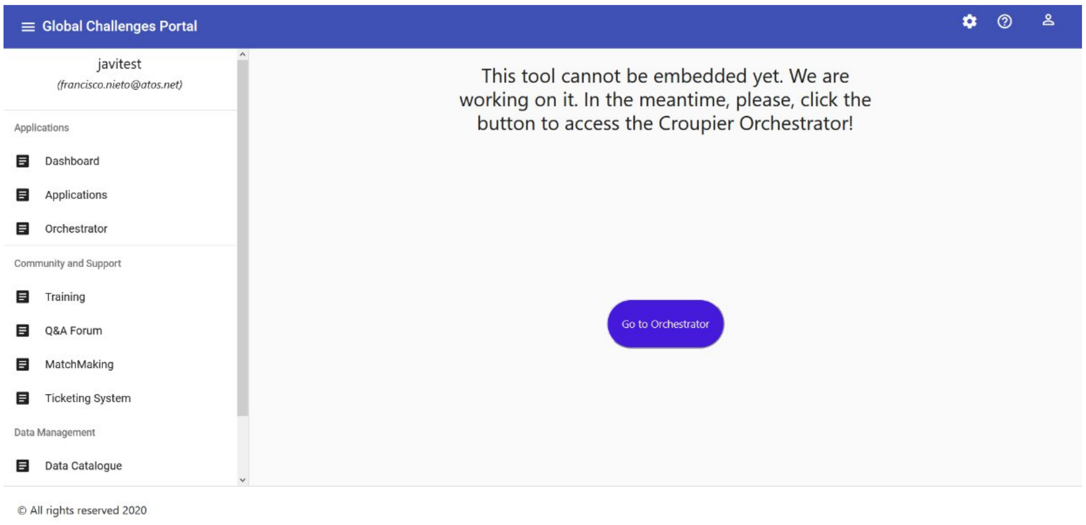


Figure 49 Adding an application from the Frontend

Other options, like the direct access to the Croupier GUI are done under 'Applications', selecting 'Orchestrator', but in such case, we are requested to click a button to open the dashboard in a separated tab.



This tool cannot be embedded yet. We are working on it. In the meantime, please, click the button to access the Croupier Orchestrator!

Figure 50 Accessing the Orchestrator GUI from the Frontend

In the case of the rest of features, most of them are embedded as iFrames, so it is necessary only to click on the corresponding option and it will be shown in the main part of the screen.

Since it is not easy to do a perfect integration, clicking on the top left button (next to the 'Global Challenges Portal' name), it is possible to hide the menu, so applications are shown in full screen (like the next figure shows with CKAN).

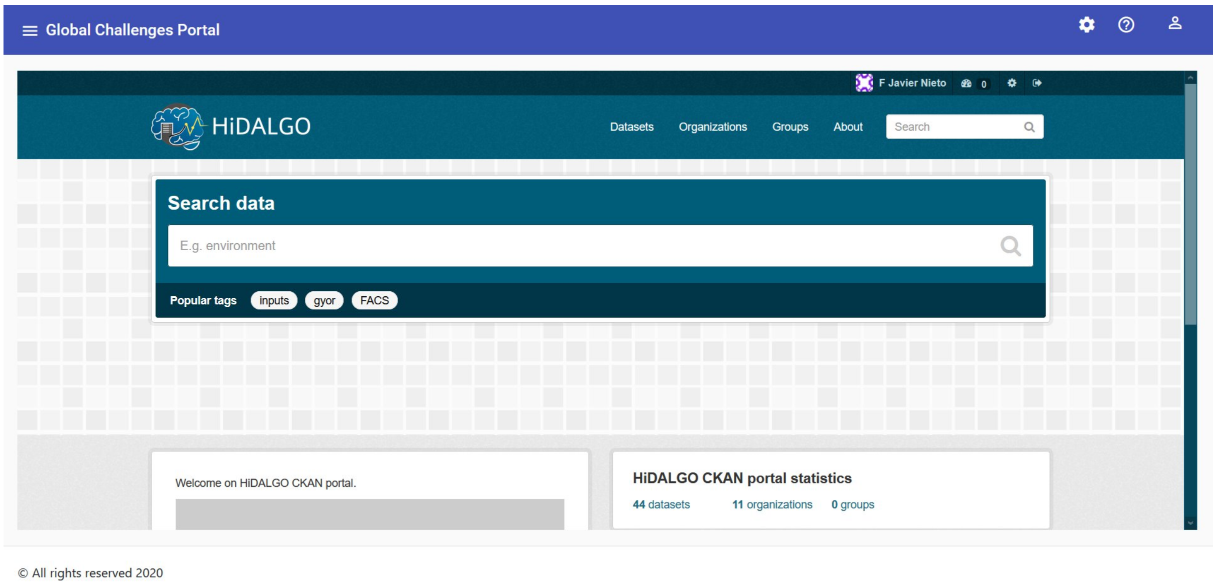


Figure 51 Access to CKAN from the Frontend hiding the menu

12 Available Infrastructures

12.1 Integration Infrastructure

For development and testing purposes of the services in HiDALGO project the Integration Infrastructure has been set up. The machines are set up and managed via Openstack environment.

Several services running on Integration Infrastructure have been described in D5.3[2]. Since then several new machines have been set up, as seen in Table below. It must be taken into account that the domain name is not the full URL to access the services.

Service	CPU Cores	RAM	VM Public IP	Domain Name
Streaming	4	8GB	62.3.171.145	sophora-145.man.poznan.pl
Visualization	4	8GB	62.3.170.209	prunus-212.man.poznan.pl
Support ticket	4	8GB	62.3.171.210	sophora-210.man.poznan.pl
Askbot	4	8GB	62.3.171.76	sophora-76.man.poznan.pl
Cloudify	4	8GB	62.3.171.103	hidalgo-cfy.hlrs.de
Matchmaking	4	8GB	62.3.171.89	sophora-89.man.poznan.pl
Zabbix	4	8GB	62.3.171.109	sophora-109.man.poznan.pl
Moodle	4	8GB	62.3.171.102	sophora-102.man.poznan.pl
Jenkins	4	8GB	62.3.171.42	sophora-42.man.poznan.pl
FrontEnd	4	8GB	62.3.171.105	sophora-105.man.poznan.pl
IDM	4	8GB	62.3.170.212	prunus-212.man.poznan.pl
Wiki	2	4GB	62.3.171.187	sophora-187.man.poznan.pl
Coegss-spark	32	32GB	150.254.165.237	ribes-237.man.poznan.pl
Notebook	6	12G	62.3.171.147	sophora-147.man.poznan.pl

Table 10 List of VMs in the Integration Infrastructure

12.2 Deployment Infrastructure

The deployment infrastructure is available in the same way as the integration one. In this case, the VMs are hosted at HLRS, also following a Cloud solution. In this case, the available VMs are listed in the following table.

Service	VM Public IP	Domain Name

Moodle	141.58.0.224	https://moodle.hidalgo-project.eu/
Cloudify	141.58.0.227	https://cloudify.hidalgo-project.eu/
Zammad	141.58.0.231	https://support.hidalgo-project.eu/
Matchmaking	No public IP	
Jenkins	141.58.0.225	https://hidalgo-jenkins.hlrs.de/
Wiki	141.58.0.223	https://wiki.hidalgo-project.eu/
CKAN	141.58.0.226	https://ckan.hidalgo-project.eu
Askbot	141.58.0.228	https://askbot.hidalgo-project.eu
Zabbix Monitoring	141.58.0.222	hidalgo-monitor.hlrs.de
Keycloak IDM	141.58.0.229	https://hidalgo-idm.hlrs.de/auth/
Portal Frontend & Backend	141.58.0.230	portal.hidalgo-project.eu
Notebook	141.58.0.232	notebook.hidalgo-project.eu
Notebook compute node-1	No public IP	
Notebook compute node-2	No public IP	
Visualizer	141.58.0.233	https://visualization.hidalgo-project.eu/
COVISE	141.58.0.233	https://visualization.hidalgo-project.eu/

Table 11 List of VMs in the Deployment Infrastructure

12.3 Training Infrastructure

Training infrastructure is a set of nodes on Eagle that aim to bring real-time HPC on a smaller scale for testing and demonstration purposes.

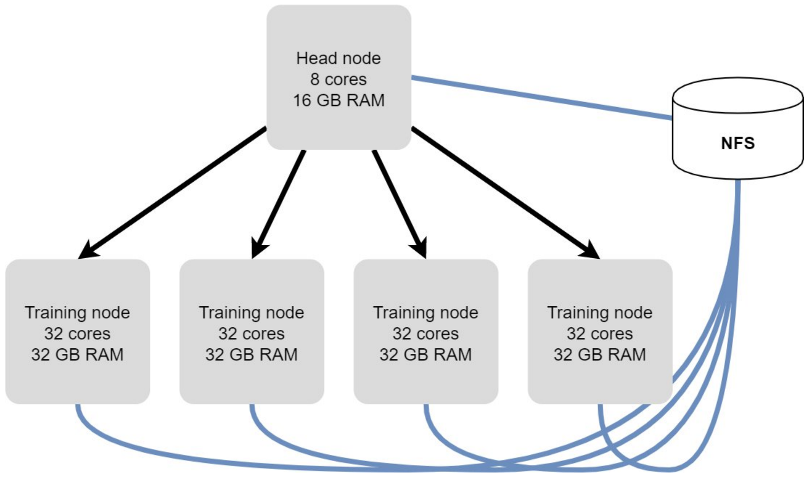


Figure 52 Architecture of Training Infrastructure nodes

It consists of 4 training nodes, where computation takes place and less powerful head node for commissioning work. All the nodes are connected to a network storage. They are equipped with SLURM workload manager and all the necessary libraries for pilot development. For security purposes the network is isolated apart from head node, and private addresses are used.

It can be accessed via SSH protocol, but it is also integrated with Cloudify service, which allows any user to perform work on it.

13Conclusions

This document describes the progress done in the second release of the HiDALGO Portal, including a description on how the feature was implemented, the APIs available and how it can be used. The progress done with respect to the previous version is important, since we added more features in several areas: applications execution, data management, users support, visualization, orchestration, users management and CI/CD. We added new tools for users support (now, besides training, we have a ticketing system and the Askbot, together with a Wiki with documentation) and we also included the Notebooks. We have now more tools for visualization, with more features and easily integrable with the Portal (Visualizer and COVISE). It is also interesting to highlight the availability of a new infrastructure for training.

All the implemented features facilitate the user experience with HiDALGO and open the opportunity to provide more and better services. The fact that the Frontend puts them together is good in order to have a one-stop-shop, even if we have a very complex system, made of many different technologies and languages.

Comparing the list of features to the original roadmap, we can say that most of the expected features are available, although there is still room for improvement. For instance, the look and feel of the Frontend needs to be improved and, therefore, several modifications are being implemented. It is also to provide a more user-centric approach, so we have defined some ways to collect information about the Portal usage and users' opinions, so we will be able to improve the Portal in the last release.

References

[1] HiDALGO, D5.2 – Portal Architecture and Roadmap. Carnero, Javier et al. 2019.

[2] HiDALGO, D5.3 – First HIDALGO Portal Release and System Operation Report. Nieto, F. Javier et al. 2019.

[3] HiDALGO, D6.1 – Requirements Process and Results Definition. Maritsch, Martin et al. 2019.

[4] HiDALGO, D6.4 – Initial Report on Requirements, Components and Workflow Integration. Tsoumakos, Dimitrios et al. 2019.

[5] HiDALGO, D5.4 – HiDALGO Support Concept. Rajagopal, Dineshkumar et al. 2020.

[6] HiDALGO, D6.2 – Workflow and Services Definition. Maritsch, Martin. 2019.

[7] Jenkins. URL: https://jenkins.io/. Last visited in December 2019.

[8] Moodle Training Service. *URL: https://www.moodle.org*. Last visited in December 2019.

[9] CKAN: https://ckan.org/, last visited in December 2019

[10]Apache Groovy Language, https://groovy-lang.org/

[11]Keycloak Server Administration Guide. URL: https://www.keycloak.org/docs/latest/server_admin/, last visited in December 2020

[12] About Cloudify. URL: https://docs.cloudify.co/5.0.0/about/, last visited December 2020

[13]OASIS. Security Assertion Markup Language (SAML) 2.0 Technical Overview. 22nd July 2004. http://xml.coverpages.org/SAML-TechOverviewV20-Draft7874.pdf. Last visited in December 2020.

[14] OpenID. OpenID Connect Core 1.0. 8th November 2014. https://openid.net/specs/openid-connect-core-1_0.html. Last visited December 2020.

[15]CAS Protocol 3.0 Specification, Apereo and Yale University. 13th January 2015. https://apereo.github.io/cas/5.1.x/protocol/CAS-Protocol-Specification.html. Last visited January 2021

[16] OASIS. TOSCA Simple Profile in YAML Version1.3. 18th September 2019. https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/cs01/TOSCA-Simple-Profile-YAML-v1.3-cs01.pdf. Last visited December 2020.

[17] Topology and Orchestration Specification for Cloud Applications Version 1.0. 25 November 2013. Oasis Standard. URL: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html, last visited December 2020

[18] Croupier GitLab Repository. URL: https://github.com/ari-apc-lab/croupier

[19]Apache Mesos REST API v1.0 - http://mesos.apache.org/documentation/latest/endpoints/

[20]HiDALGO, D5.1 – HiDALGO System Environment. Abhishek et al. 2019.

[21]Wiki.js official documentation, https://docs.requarks.io/

[22]Web Content Accessibility Guidelines (WCAG) 2.0, W3C Recommendation 11 December 2008. URL: https://www.w3.org/TR/WCAG20/, last visited April 2021

Annex 1: Jenkins Pipeline Definition for Zammad

```
#!/usr/bin/groovy

pipeline {
    agent {label 'master'}

    options {
        disableConcurrentBuilds()
    }

    environment {
        PYTHONPATH = "${WORKSPACE}/"
    }

    stages {
        stage("Integration - Install") {
                steps { integration() }
        }
        stage("Integration - Test") {
                steps { runUAT("sophora-210.man.poznan.pl", "https") }
        }
        stage("Approve for Production") {
                steps { approve() }
        }
        stage("Deployment - Install") {
                steps { deploy() }
        }
        stage("Deployment - Test") {
                steps { runUAT("support.hidalgo-project.eu", "https") }
        }
    }
}

def integration() {
        sh "ansible-playbook -i
Jenkins/Inventory/zammad_integration.INI ./zammad-integration.yml --vault-
password-file=~/HiDALGO/VaultPassword/zammad_vault.txt"
}

def deploy() {
        sh "ansible-playbook -i Jenkins/Inventory/zammad.INI ./zammad-
deployment.yml --vault-password-
file=~/HiDALGO/VaultPassword/zammad_vault.txt"
}

def approve() {
        try {
                timeout(time:1, unit:'DAYS') {
                        input('Do you want to deploy to live?')
                }
        } catch(err) {
                def user = err.getCauses()[0].getUser()
                if('SYSTEM' == user.toString()) { // SYSTEM means timeout.
                        didTimeout = true
                } else {
```

```
                                userInput = false
                                echo "Aborted by: [${user}]"
                      }
            }

}

def runUAT(hostname, protocol) {
        sh "Tests/ping_cloudify.sh ${hostname} ${protocol}"
}
```