D2.5 Infrastructure Provisioning, Workflow Orchestration and Component Integration (M24)

HiDALGO2

CENTRE OF EXCELLENCE

Date: December 20, 2024

EuroHPC Joint Undertaking

Document Identification			
Status	Final	**Due Date**	31/12/2024
Version	1.0	**Submission Date**	20/12/2024

Related WP	WP2	**Document Reference**	D2.5
Related Deliverable(s)	D2.1, D2.4, D2.7, D3.1. D4.1	**Dissemination Level (*)**	PU
Lead Participant	ATOS	**Lead Author**	Jesús Gorroñogoitia (ATOS)
Contributors	USTUTT, PSNC, SZE	**Reviewers**	Michal Kulczewski - PSNC
			Ana Sánchez - MTG

Keywords:

Operational Environment, Infrastructure Provisioning, Dashboard Architecture, Component Integration, Hybrid Workflows, Development Roadmap

Document Information

List of Contributors	
Name	**Partner**
Youssef El Faqir El Rhazoui	ATOS
Jesus Gorroñogoitia	ATOS
Sameer Haroon	USTUTT
Piotr Dzierżak	PSNC
Marcin Lawenda	PSNC
Bartosz Bosak	PSNC
Michal Kulczewski	PSNC
László Környei	SZE
Luis Torres	MTG
Christophe Prud'homme	UNISTRA
Michael Zikeli	FAU

Document History			
Version	**Date**	**Change editors**	**Changes**
0.1	05/09/2024	ATOS	Table of Contents
0.15	10/09/2024	Marcin Lawenda (PSNC) Harald Koestler (FAU)	ToC, timeline and responsibilities approved
0.2	24/10/2024	ATOS	Initial contribution to section 4
0.3	05/11/2024	ATOS, USTUTT, PSNC, UNISTRA, FAU	Initial contributions to sections 2, 3, 4 and 5
0.4	26/11/2024	ATOS, USTUTT, PSNC, MTG	Final contributions to sections 2, 3, 4 and 5
0.5	02/12/2024	ATOS, USTUTT, PSNC, SZE	Final contributions to sections 1, 2, 3, 5 and 6
0.6	03/12/2024	ATOS	Peer-Review version
0.7	12/12/2024	ATOS	Peer-Review amendment. QM version
0.99	18/12/2024	Harald Koestler	Quality assurance check
1.0	20/12/2024	Marcin Lawenda	Final check and improvements

Quality Control		
Role	**Who (Partner short name)**	**Approval Date**
Deliverable leader	Jesús Gorroñogoitia (ATOS)	12/12/2024

Quality manager (deputy)	Harald Koestler (FAU)	18/12/2024
Project Coordinator	Marcin Lawenda (PSNC)	20/12/2024

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
AI	Artificial Intelligence
API	Application Programming Interface
CFDR	Computational Fluid Dynamics Render
CI/CD/CB	Continuous Integration / Continuous Delivery / Continuous Building
CoE	Centre of Excellence
CPU	Central Processing Unit
Dev	Development
Devops	Development Operations
DMS	Data Management System
DNS	Domain Name System
DVFS	Dynamic voltage and frequency scaling
Dx.y	Deliverable number y belonging to WP x
EuroHPC	The European High Performance Computing Joint Undertaking
EO	Energy Optimizer
Flops	Floating point operations
GB	Gigabyte
GUI	Graphical User Interface
GPU	Graphics Processing Unit
HDD	Hard Disk Drive
HDFS	Hadoop File System
HPC	High Performance Computing
HPDA	High Performance Data Analytics
IDM	Identity Management
I/O	Input/Output
IP	Internet Protocol
JEC	Job Energy Comparator

JEP	Job Energy Predictor
JER	Job Energy Recommender
JU	Joint Undertaking
MPI	Message Passing Interface
MTW	Material Transport in Water
N/A	Not applicable
NFS	Network File System
NVML	NVIDIA Management Library
PPE	Power and Performance Estimator
Prod	Production
RAM	Random Access Memory
RAPL	Running Average Power Limit
RDM	Research Data Management
RES	Renewable Energy Sources
SA	Sensitivity Analysis
SCP	Secure File Copy
SSL	Secured Socket Layer
SSH	Secure Shell
SSO	Single Sign On
TB	Terabyte
UQ	Uncertainty Quantification
VCPU	Virtual Central Processing Unit
VM	Virtual Machine
WebUI	Web User Interface
WFO	Workflow Orchestrator
WP	Work Package
2FA	Two Factor Authentication

Executive Summary

This deliverable, D2.5, is the second one of the series of documents (D2.4, D2.5, D2.6) on "Infrastructure Provisioning, Workflow Orchestration and, Component Integration" that reports on the updates, since the first document of the series, D2.4 [1] , on different topics reported therein:

- The holistic HiDALGO2 platform of computing and services integrated within a unique Dashboard (see D4.6 [2]): this platform makes up the core foundation of the HiDALGO2 ecosystem. It supports the project's pilots in achieving their scientific, technical, and environmental goals. It builds upon the work reported in previous deliverables for requirements elicitation, D2.1 [3] and D2.2 [4] , but also upon other deliverables published up to date, where the HiDALGO2 services and pilots' applications are described, namely D2.7 [5] , D3.1 [6] , D4.1 [7] , D4.6 [8] , D5.6 [9] . Updates since D2.4 on the allocated computing resources, including infrastructure for new services, and the delivered services have been reported.
- The development progress of the planned features for the HiDALGO2 Workflow Orchestrators, namely the MathSO and the QCG, which are the central components of the Dashboard supporting the delivery, execution, and monitory life cycle of pilot's simulations as hybrid workflows on EuroHPC sites. The current state of development of scheduled and new features, addressing the elicited requirements, are reported or re-scheduled in the roadmap until the end of the project.
- The new framework for job energy consumption monitoring and optimization, based on the Eviden SEMS EO tool, for enforcing power capping on compute nodes, whose need is motivated, and grounded on requirements elicited from the SEMS clients. Within HiDALGO2, extended features for an AI-powered power and performance estimator, a job energy comparator, a job energy predictor, and a job energy recommender are described and scheduled for development.
- The deliverable progresses on the strategy and vision for integrating all its constituent components. The design and implementation of the three distinct layers of component integration, or CI/CD pipelines, at the level of i) the HiDALGO2 services, ii) the delivery of the pilots' applications, and iii) the sharing of the HiDALGO2 services and pilot's application with other EuroHPC CoEs' users, have been improved since D2.4 and reported in this document.

Finally, the roadmaps drawn out in the individual chapters give the direction the HiDALGO2 technical infrastructure and supporting development will follow, which will be documented in the last iteration of this series of deliverables, in D2.7.

1. Introduction

1.1 Purpose of the document

The deliverable, D2.5, is the second one in the series of documents (D2.4, D2.5, D2.6) on “Infrastructure Provisioning, Workflow Orchestration, and Component Integration”. It provides updates, since D2.4 [1] , on:

- The HiDALGO2 services integrated within the Dashboard,
- The allocated infrastructure for computing and services,
- The allocation of Euro-HPC resources for pilots’ simulation,
- The features and their current status of implementation for the main HiDALGO2 Workflow Orchestrators, namely the MathSO and the QCG,
- The motivation, requirements, and features for the planned Energy Monitoring and Optimization framework for HPC,
- The updated CI/CD platform for the delivery of the HiDALGO2 services,
- The updated CI/CD platform for the delivery of the HiDALGO2 pilots’ simulations,
- The common, shareable CI/CD platform for the delivery of HiDALGOs’ services and pilot applications from all the CoEs of EuroHPC.

This deliverable keeps the same structure of D2.4, but only provides updates, since D2.4, for those sections whose content has evolved. It always refers readers to D2.4 when it is needed to complement or understand the updates of any section. In a few key sections, some D2.4 content has been borrowed and acknowledged, as being of great importance for reading to be included therein.

1.2 Relation to other project work

This document, as the second one of the series, provides the updates, on the topics of interest, since its last release, D2.4. It builds upon the motivation behind integrating the various HiDALGO2 services into a common Dashboard, as well as the available CI/CD pipelines for the delivery of those services and the pilots’ applications within a common and sharable CI/CD framework for all the CoEs of the EuroHPC. This motivation leverages on the requirements collected in D2.1 [3] and D2.2 [4] . It also coordinates closely with other deliverables published up to date, where the HiDALGO2 services and pilots’ applications are described, namely D2.7 [5] , D3.1 [6] , D4.1 [7] , D4.6 [8] , D5.6 [9] .

1.3 Structure of the document

Chapter 2 describes the updates on the hardware and software resources allocated to the project.

Chapter 3 describes the updates on the features and the current state of development of the HiDALGO2 Workflow Orchestrator Engines, their frontends integrated within the HiDALGO2 Dashboard, and the updates in their development roadmap.

Chapter 4 introduces a framework for energy monitoring and optimization for HPC, its motivating requirements, proposed features, and its planned development roadmap.

Chapter 5 describes the current implementation of the CI/CD system for delivering the HiDALGO2 services, pilot applications, and simulations, as well as the strategy for a public and shareable CI/CD framework for all EuroHPC-related CoEs.

Chapter 6 summarizes and concludes this deliverable while giving an outlook on future work.

2. Infrastructure Provisioning

HiDALGO2 provides a comprehensive ecosystem to support the development of its use cases, comprising of extensive hardware resources and software services. This ecosystem also directly supports the developers themselves, while simultaneously growing in the direction of supporting and serving HiDALGO2's future users.

The following sections provide updates, and point to the relevant sections in the previous iteration of the deliverable (D2.4), for further information.

2.1 Motivation and requirements

As the requirements and motivations for provisioning infrastructure for the HiDALGO2 project remain consistent and will continue to do for the remainder of the project and beyond, please refer to Deliverable D2.4, Section 2.1.1.

2.2 Design and Architecture

Following on from D2.4, it is provided herein a recap of the current list of web and software services allocated and utilised by the HiDALGO2 project, as well as an updated architecture of how these services build up and fit into the HiDALGO2 infrastructure design.

2.2.1 System Design

The overall design of the HiDALGO2 ecosystem continues to grow on the foundations established and documented through Deliverable D2.4, section 2.2. A reworked design diagram, with an alternative labelling scheme is provided in Figure 1, to provide an additional viewpoint. The individual components themselves are described in the following sections, or have already been elaborated upon in Deliverable D2.4, Section 2.1.3.

2.2.2 List of Services

Table 1 gives an updated listing of the current services offered in the HiDALGO2 environment. For newly added services. details about the purpose of the services, as well as details about their infrastructure are provided in the following sections. Most of the services, however, were already established by year 1 of HiDALGO2. For details on already existing services and infrastructure, please refer to Deliverable D2.4, section 2.1.3.

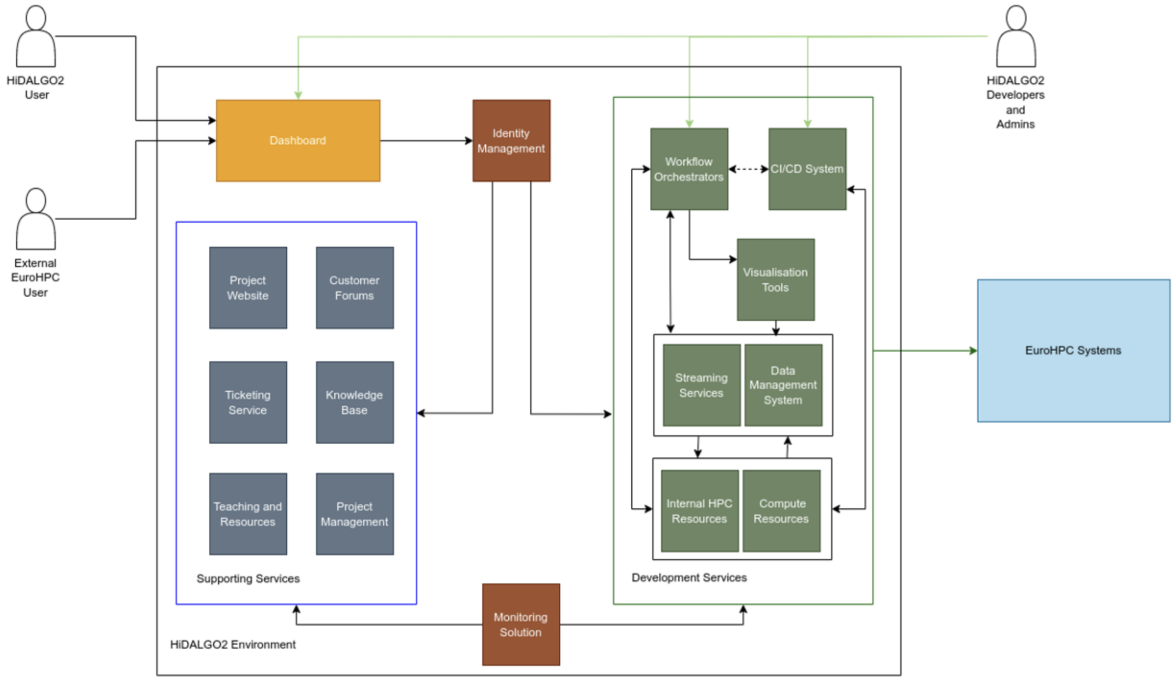


Figure 1. System Design Diagram for HiDALGO2

Table 1. Overview of Web Services

	Service	Description	Domain Link
1	Website	Project Website	https://hidalgo2.eu
2	MathSO Portal	Workflow Orchestrator	https://portal.hidalgo2.eu
3	Prototype	Compute Cluster	https://prototype.hidalgo2.eu
4	JupyterHub	Jupyter Notebooks	https://jupyter.hidalgo2.eu
5	IDM	Identity Management	https://idm.hidalgo2.eu
6	Askbot	User Forums	https://askbot.hidalgo2.eu
7	CKAN	Data Management System	https://ckan.hidalgo2.eu
8	Bitbucket	Git Repository	https://git.man.poznan.pl/stash/projects/HIDALGO2
9	Zammad	User Support	https://ticket.hidalgo2.eu
10	Wiki	Knowledge Management	https://wiki.hidalgo2.eu
11	Open Project	Project Management	https://project.hidalgo2.eu
12	Moodle	Learning Platform	https://moodle.hidalgo2.eu
13	SEMS	Energy Monitoring	
14	Dashboard	HiDALGO2 dashboard	https://dashboard.hidalgo2.eu

2.3 Current implementation

2.3.1 Supplementary infrastructure for computing

A new cluster infrastructure for SEMS Energy Monitoring and Optimization for HPC cluster has been created (see Table 2).

Table 2. Infrastructure for SEMS Monitoring and Optimization for HPC cluster

<table>
<tr><td>Name</td><td>SEMS Cluster</td><td>Prod</td><td>PSNC</td><td>IP</td><td>150.254.224.172</td></tr>
<tr><td>Software</td><td>Rocky Linux</td><td>Dev</td><td>N/A</td><td>IP</td><td>N/A</td></tr>
<tr><td>Domain</td><td colspan="5">N/A</td></tr>
<tr><td>Description</td><td colspan="5">The cluster contains one access node and six compute nodes.
<table>
<tr><th>[NAME]</th><th>[CPU]</th><th>[RAM]</th><th>[DISK]</th><th>[OS]</th></tr>
<tr><td>hidalgo2-sems-access</td><td>8</td><td>8GB</td><td>40GB</td><td>Rocky Linux 9</td></tr>
<tr><td>hidalgo2-sems-comp-1</td><td>8</td><td>8GB</td><td>40GB</td><td>Rocky Linux 9</td></tr>
<tr><td>hidalgo2-sems-comp-2</td><td>8</td><td>8GB</td><td>40GB</td><td>Rocky Linux 9</td></tr>
<tr><td>hidalgo2-sems-comp-3</td><td>8</td><td>8GB</td><td>40GB</td><td>Rocky Linux 9</td></tr>
<tr><td>hidalgo2-sems-comp-4</td><td>8</td><td>8GB</td><td>40GB</td><td>Rocky Linux 9</td></tr>
<tr><td>hidalgo2-sems-comp-5</td><td>8</td><td>8GB</td><td>40GB</td><td>Rocky Linux 9</td></tr>
<tr><td>hidalgo2-sems-comp-6</td><td>8</td><td>8GB</td><td>40GB</td><td>Rocky Linux 9</td></tr>
</table>
CephFS share – 1TB</td></tr>
</table>

In the PSNC cloud services, new storage shares based on CephFS were created. The new resources were mounted to three clusters: Jupyter, Prototype and SEMS. In the case of the first two clusters, the user data was moved from the NFS share to the new one based on CephFS. This results in better performance of the disk system.

The resources for the namenodes and datanodes in the Hadoop cluster were extended. For the datanodes, the RAM memory was increased from 8 to 16 GB. For the namenodes, the CPU cores were increased from 8 to 16 and the RAM from 16 to 32 GB.

Currently the resources for HiDALGO2 in PSNC infrastructure consumes:

- 30 virtual machines
- 342 CPU cores
- 500 GB RAM
- 28 TB storage for virtual machines
- 5 TB additional storage on CEPH

– 20 public IP addresses

2.3.2 Infrastructure for Web Services

In order to maintain the project's resources, operating systems and software installed on them are constantly updated. The most important updates include Keycloak [10] , Zabbix [11] and Wordpress [12] .

2.3.3 Infrastructure for Dashboard

The dashboard infrastructure requirements are described in D2.7 section 3.3. The minimum hardware requirements are imposed by the software listed in software requirements and the additional resources required by deployments within the dashboard. Current setup links all additional services outside the dashboard deployment, so basic services require minimal storage and computational resources. The current resources (see Table 3) supporting deployment on *dashboard.hidalgo2.eu* with a minimal 4 VCPU, 8GB RAM, 40GB HDD system are sufficient, even with several additional static pages.

Table 3. Virtual machine for HiDALGO2 dashboard

Name	Dashboard		Prod	PSNC	IP	62.3.171.60	
Software	Nginx, PHP		Dev	N/A	IP	N/A	
Hardware	CPU cores	4	RAM	8 GB	Hard Disk		40 GB
Domain	https:// sophora-60.man.poznan.pl dashboard.hidalgo2.eu api.dashboard.hidalgo2.eu						
Description	The main dashboard instance for HiDALGO2.						

Two DNS entries for the service were also created. Domain dashboard.hidalgo2.eu is dedicated for the main dashboard application and the api.dashboard.hidalgo2.eu for the backend server. An SSL cert for the new domains was also requested and obtained.

2.4 Development Roadmap

This section continues the development roadmap related to infrastructure and service provisioning in HiDALGO2.

2.4.1 Further access to EuroHPC Sites

All HiDALGO2 partners now have direct access to EuroHPC sites. Previously, this was done through shared access through multiple partners, or limited to certain access schemes. The new HiDALGO2 partner, FAU Erlangen, and their use case, MTW (Material Transport in Water) are also included in the updated Table 4 that describes the current status of access to the EuroHPC sites for the HiDALGO2 pilots.

Table 4. Access to EuroHPC sites

	Urban Air Pollution (SZE)	Urban Building Modelling (UNISTRA)	Renewable Energy Sources (PSNC)	Wildfires (MTG)	Bench-marking (ICCS)	Material Transport in Water (FAU)
LUMI	Awarded	Awarded	Awarded	Awarded	Awarded	Awarded
Vega	Awarded	Awarded	Awarded	Awarded	Awarded	Awarded
Karolina	Awarded	Awarded	Awarded	Awarded	Awarded	Awarded
Meluxina	Awarded	Awarded	Awarded	Awarded	Awarded	Awarded
Discoverer	Awarded	Awarded	Awarded	Awarded	Awarded	Awarded
Leonardo	To be requested	Awarded	Awarded	Awarded	Awarded	Awarded
MareNostrum5	Awarded	System not available yet	Awarded	Awarded	To be Requested	Awarded
Deucalion	Awarded	System not available yet	Awarded	To be Requested	To be Requested	Awarded

2.4.2 Extending Web Services

Hereafter, following up from the deliverable D2.4, it is addressed the development roadmap points related to HiDALGO2 web services.

- All Web Services are to be integrated with the HiDALGO2 Keycloak IDM, as well as the Operational Portal services and the Workflow Orchestrators.
  - This point was fulfilled. Updated information for the integration status of each individually provided service is recorded in Annex 1: Services Matrix, in Figure **Error! Reference source not found.**.
- Services are to be customised to allow for access to external and 3rd party users. For example, a separate realm in the Keycloak IDM shall be created to service external users while maintaining the security of the HiDALGO2 ecosystem.

    - As HiDALGO2 has not serviced external users yet, this point is still valid, and to be followed in the next phase.
- In order to ensure compatibility with the planned vision of the EuroHPC CI/CD Pilot Platform, it is planned to have a Gitlab mirror of the HiDALGO2 Bitbucket repository or change the Git repository solution altogether.
    - This point was full-filled, and more details are in Section 5, Component Integration, in both this deliverable and the revised Deliverable D2.4.
- Additional storage resources to be allocated for the Data Management System, from both USTUTT and PSNC side, to allow for expected increase in data storage needs as Tasks from Year 2 of the project start e.g. the Uncertainty Quantification runs.
    - This point has been partly full-filled. Some infrastructure has been extended with additional storage resources. However, it is expected that this will be a continuous requirement, with institutions providing more resources as the needs of the services and use cases grow.
- Additional resources needed to support AI and HPDA services.

This is a new point, and expected to be a continuous requirement, as pilot use cases advance their development in AI related use cases, so more resources will need to be allocated to support this development. (Compute, Acceleration, Storage and networking).

3. Workflow Orchestration

3.1 Requirements

The list of initially collected requirements for the workflow orchestration has been described in both D2.1 and D2.4. These requirements keep valid and play pivotal role also after the second round of the requirements collection, which has been summarised in D2.2. From the previous year, there are two new workflow orchestration specific requirements defined:

- REQ-POR-028, which applies for the integration with QCG for ensemble runs and uncertainty quantification and
- EQ-POR-029, which applies for simulation environment specification.

Further requirements, like the need for project-wide SSO mechanism, or powerful visualisation of generated data, although not directly bound to orchestration, influence on the orchestration realisation at least to some extent.

Please note, that due to the requirements alignment, the set of requirements presented in D2.4 with the infix WFO, has now migrated to the set of portal requirements (POR infix) in Annex II of D2.2. Additionally, to enhance readability, the requirements have been organized into several distinct groups. This deliberate modification allowed us to highlight specific workflow orchestration requirements in connection with the broader needs of the portal system as a whole. Let's explore the main groups of this list and their impact on the orchestration:

- orchestration - this group outlines all main requirements for the workflow orchestration in HiDALGO2; it includes the WFO requirements inherited from D2.4;
- workflows – concentrates on the standardisation of the workflows so they can run seamlessly on different underlying resources;
- monitoring – outlines expectations in regards of the monitoring of the pilot run, including health and progress of workflow execution, as well as resource utilisation;
- authentication and authorisation – include requirements related to workflow maintenance and execution security: the need for integration with HiDALGO2 Identity Management System, incorporating user roles and adhering to the selected way of credential management;
- interfacing data management – express the needs of pilots' workflows and workflow orchestrator users to access data stored in data management systems available to HiDALGO2 community, e.g., CKAN;
- visualisation – includes requirements for the visualisation of the data of completed or running workflow: running visualization should be possible from the GUI of the workflow orchestrator.

Both HiDALGO2 WFOs support the execution of hybrid workflows: those that combines both job-based simulation tasks with AI and HPDA analytic tasks.

3.2 MathSO

3.2.1 Functional Specification

Since the submission of D2.4, the following new functionalities have been implemented into the MathSO WFO. There are various features (see Figure 2) enhancing WebUI, workflow, data transfer and integration.

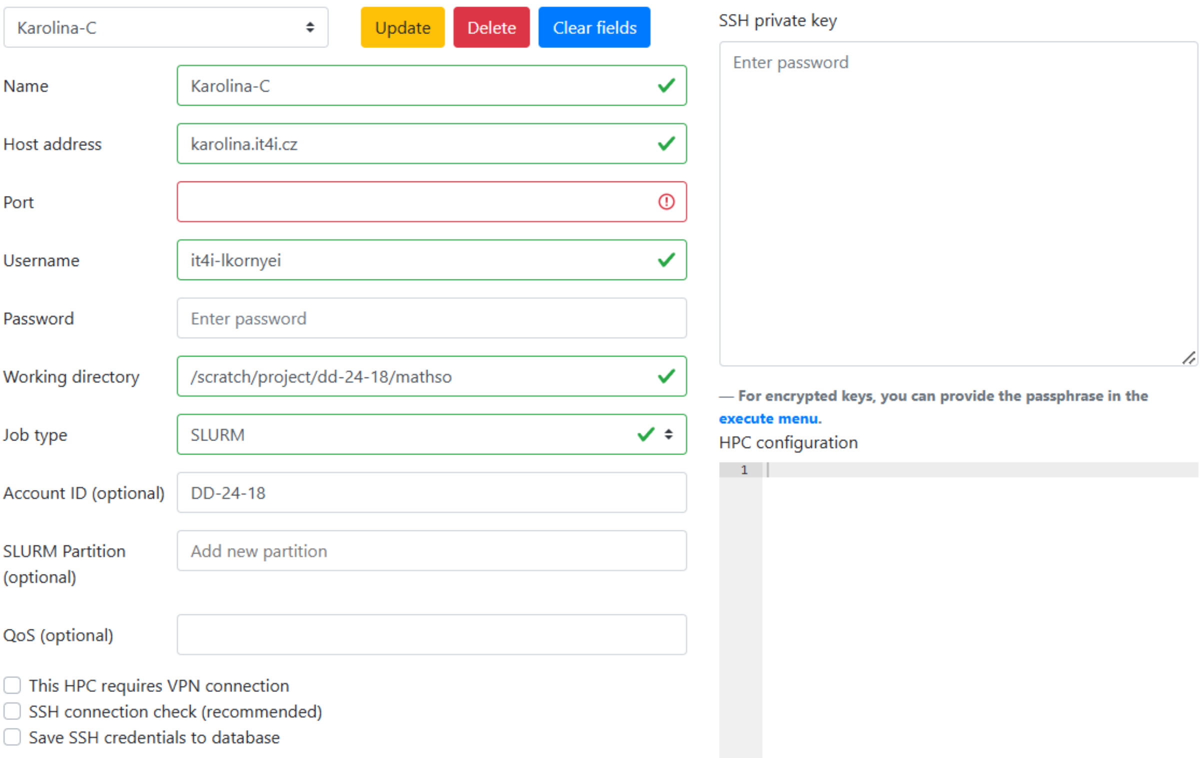


Figure 2. HPC Configuration in the MathSO portal, showcasing some of the new features detailed in this section[1]

Two-factor authorization

One-time passcodes and 2FA links are tested. When connecting, the WFO detects 2FA challenges. The user has the ability to read the output of the SSH prompt, copy the authenticator link and enter the 2FA response. This feature enhances connectivity to EuroHPC machines, like Vega.

[1] The “Port” value was intentionally deleted to indicate behaviour in case of empty compulsory field.

Encrypted or unsaved private keys

While the portal does store the private SSH key for connection, the user has the option not to save it. However, this way, on every job submission, the user has to provide the private key. The user also has the ability to encrypt the private key. Encryption passwords are never stored, and are always prompted for. Alas, the user has the ability to use the same password for all encrypted keys, so it needs to be entered only once.

Connection test and partition availability.

The user has the ability to test the configured HPC sites, if connection is available (see Figure 3). Also, the portal can query the availability of the configures partitions, so the user can decide beforehand, which resource to use. There is also a connection test for data repositories. These features facilitate pinpointing failures in the workflow system.

HPC based configuration.

Several new configuration settings are available, when setting up an HPC. It is possible to choose the deployment directory, allowing the user to set a preferred storage system, like project or scratch. It is also possible to set various environment variables specific to the HPC, including, for example, module names for singularity or MPI. The variables will be stored in the deployment directory in a file named hpc.conf, so applications will be able to take advantage of the different settings. It is our aim, to have as many configurations setting detected, as possible,

Workflow configuration

All workflow configuration is saved as parameter.conf, enabling a simpler development of blueprints.

Tracking compulsory fields

While filling out various settings in the MathSO WFO, cells will be colour coded. Red coloured cells are to be filled in. Green coloured cells are compulsory, and already filled in. The coding does not check for the validity of the input.

Various file I/O implementations

While several HPC machines may not support file transfer tools, like wget or curl, others might restrict internet access. The WFO currently support data transfer through the SSH connection using AsyncSSH. The SSH connection is maintained throughout the simulation run, so down- and upload of data is happening seamlessly. However, if connection is lost, the user may need to reauthenticate to proceed with execution or transfer output data. Additional transfer methods are planned, including simple methods with wget or curl, including "just link" or copy if transferring from or to a local repository.

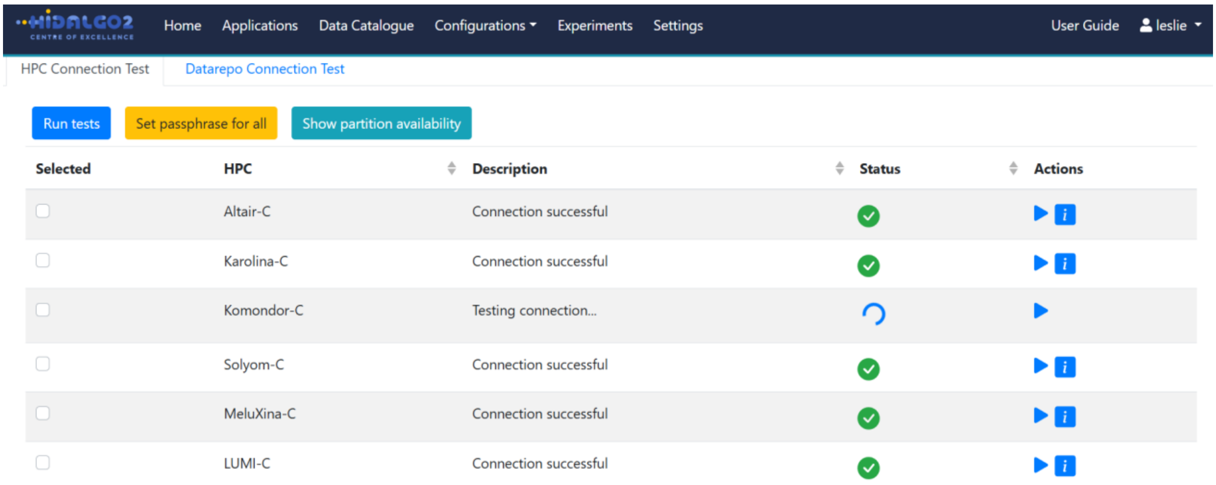


Figure 3. Testing connections to HPC sites

3.2.2 Design and Architecture

The current version of the MathSO portal does not include any design or major architectural changes. There is an initial implementation of an SSH-based data repository, which supports data storage on HPC clusters. This will facilitate deployment of application images and input and output data, as separate download will not be necessary, but is still in testing.

It also must be noted, that the Python SSH support module has been switched to AsyncSSH, after having had encountered connection problems, including to the LUMI EuroHPC system. The previous WFO architecture initiated a separate connection for every SSH command, which may have triggered the security protocols. The current implementation uses a persistent connection, which is kept alive during simulation, and for additional few minutes. This time frame can be configured by the user, and can be used to inspect simulation logs, for example. The timer resets with every interaction. At timeout, the connection is closed, and further interactions may prompt the re-entry of the security keys or passwords.

3.2.3 Current implementation

This section will discuss the status of the features described in the D2.4 development roadmap.

Parameter Sweep Function

A primary implementation of the feature is already in place. The current version uses the array job feature of the SLURM system to mass-deploy simulations, however it still needs testing and implementations outside the SLURM system.

Enhanced Data Repository Support

MathSO already supports CKAN, and has recently added SSH repositories. Support for more types will be implemented, as newer DMS systems will be available

Improved Data Management

As stated in the previous section, MathSO already supports data upload for results via AsyncSSH, and will support additional data transfer methods as well.

HPC Account Monitoring

As stated in the previous section, MathSO already supports collecting information from the HPC centres, including availability of predefined partitions, used/idle nodes and their overall usage. Additional information queries about the deployed jobs will be implemented on M36.

3.2.4 Development Roadmap

This section provides an update of the implementation roadmap of the MathSO orchestrator.

Parameter Sweep Functions

The feature needs more time for implementation and testing, delaying the release to M30.

Additional Data Repository Support

The MathSO team is continuously working on the integration of new DMS platforms. An abstraction layer may be necessary, to facilitate integration for function, like dataset download, upload, query, listing and filtering. Implementation and release will depend on the availability of the new DMS platforms.

HPC Account Monitoring

While some features are already implemented, job information query feature is still planned for release by M36.

For the User Notification System, and the Intelligent JOB Submission, there is no update in release plan for M36 and M42, respectively.

3.3 QCG Portal

3.3.1 Functional Specification

Functionality of QCG workflow orchestrator has been extended in several directions and provides a few new capabilities in comparison to the version described one year ago in D2.4. The list below summarises the main achievements in regards of new functionality:

Data Management Systems – CKAN integration

As the central DMS in HiDALGO2, CKAN has been prioritized for integration with QCG to facilitate seamless project-wide data management (see Figure 4). Current mechanism is based on direct calls from QCG-Agent to CKAN API to download or upload data. The work has started to employ a recently developed in WP4 dedicated HiDALGO2 Data Transfer Tool that provides an abstraction layer over different DMSes used in the project. The tool uses Apache NIFI's transfer groups mechanism to boost the performance of the transfers.

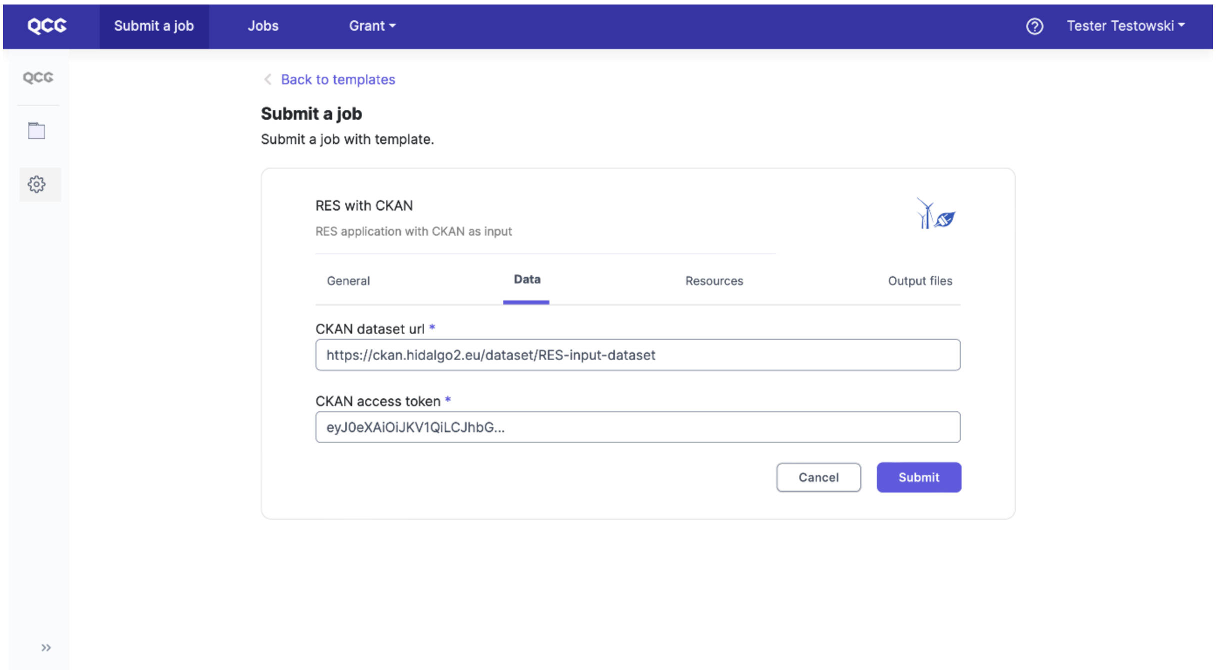


Figure 4. Selecting a CKAN dataset for a job submitted from QCG-Portal

Workflows with Apache Airflow

In order to add full-featured support for workflows in QCG the decision has been made to integrate the system with the widely used and highly flexible Apache Airflow platform. The preliminary integration has already materialised. At this moment it allows to start and supervise the Apache Airflow workflows from QCG as well as launch individual steps of the workflow processed with Airflow through QCG-Backend (see Figure 5 and Figure 6).

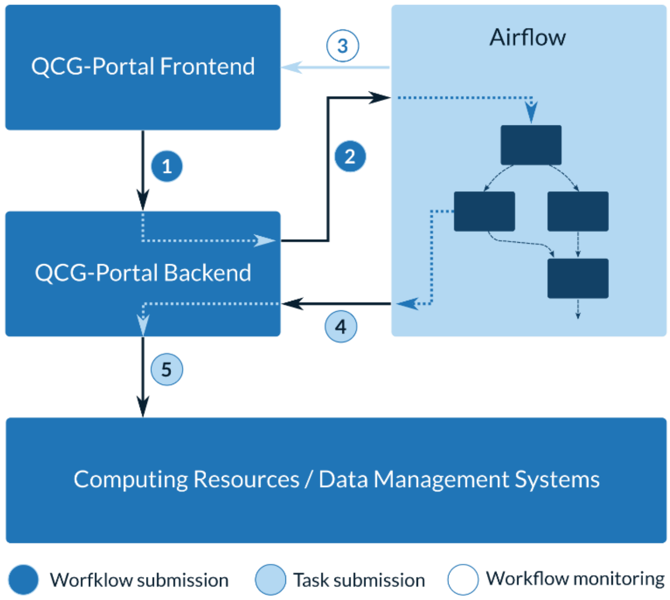


Figure 5. High-level scheme for QCG and Airflow integration

At present, the main development challenges are in the security layer. Firstly, to ensure the required isolation in the execution of users' workflows, Airflow must be configured to work with Kubernetes, which has not been done yet. Secondly, there is currently no mechanism in place to securely handle long-running workflows. Future efforts will therefore prioritize deploying Airflow with Kubernetes and developing a robust credential management strategy to enable seamless execution of workflows while upholding a high level of security. Additionally, adapting the QCG frontend to accommodate the specific requirements of workflow tasks will also be a key objective.

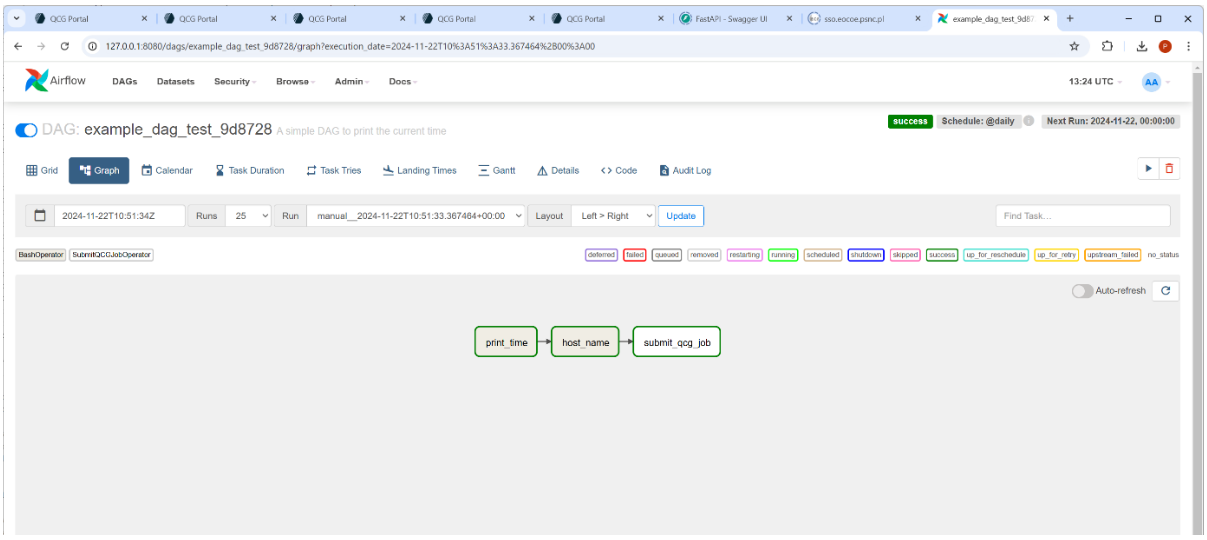


Figure 6. Airflow GUI presenting the execution of a workflow launched from QCG and involving submission of tasks to QCG-Backend via the newly created QCGOperator

Integration with a cluster through off-site SSH-based QCG-Agent

The need for the development of QCG-Agent for the integration of the system with EuroHPC sites has been described in D2.4. From the time of publishing that deliverable, the first-version of the agent has been released and integrated with the HiDALGO2 infrastructure. It allows for the integration with practically all Slurm-managed resources, including the majority of EuroHPC sites. The current solution needs to be updated once the preferred representation and source of user credentials will be established for the project – at this moment only user keys stored in QCG-Backend are supported.

Monitoring of tasks with help of Grafana

Newly created monitoring mechanism of jobs launched with QCG workflow orchestrator, allows to monitor various aspects of experiment execution based on custom monitoring templates. As presented in Figure 7 it employs Grafana for the visualisation of monitored metrics and Redis as a monitoring data exchange intermediary. An example dashboard for monitoring of mUQSA platform is presented in Figure 8.

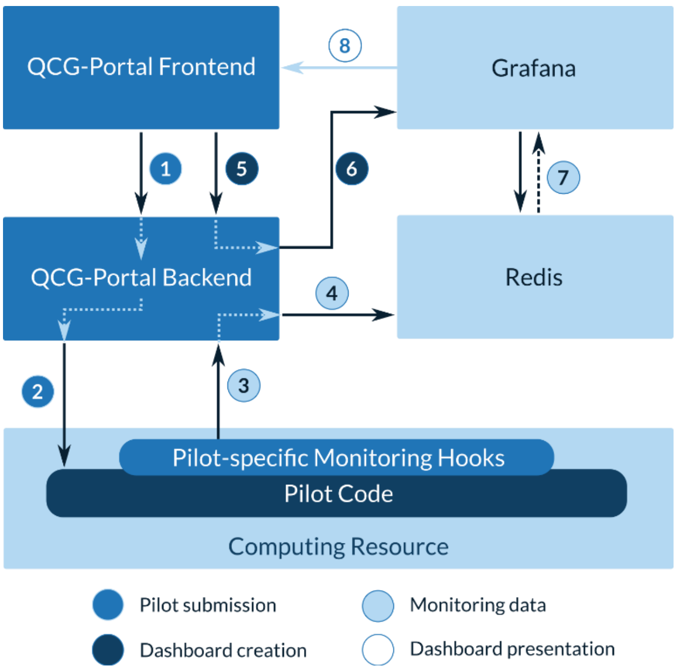


Figure 7. High-level architecture of QCG monitoring subsystem

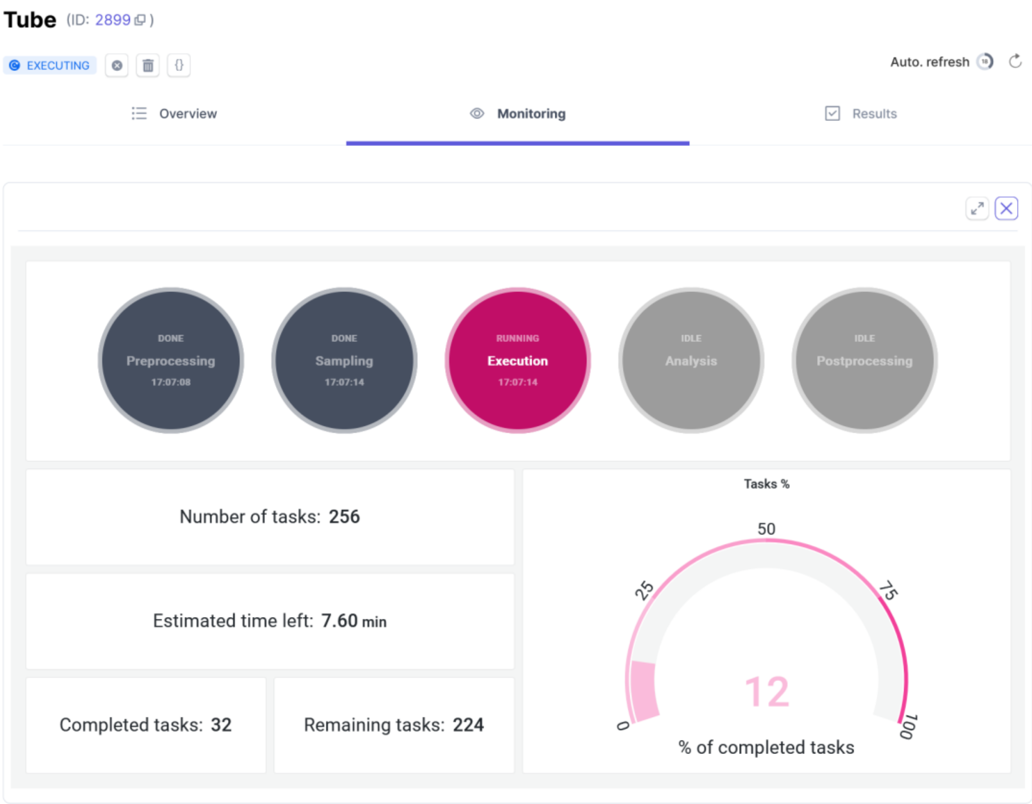


Figure 8. Tracking progress of mUQSA with the new monitoring mechanism in QCG

mUQSA release

To large extent the first edition of the mUQSA platform has been developed before the month 12 of the project, what was reported in D2.4. However, at that stage, the platform had several minor drawbacks, and more importantly, it was not offered to all HiDALGO2 users. The actual release of the platform for HiDALGO2, with the fine-tuned interface (see Figure 9), came in June 2024, when it was deployed on the Altair cluster and made available to the community. It is worth noting that mUQSA has been offered to participants of the HiDALGO2-CIRCE-SEAVEA Hackathon that took place from 5th to 7th of June 2024. The positive reception of the tool by the SEAVEA project representatives resulted in its inclusion in SEAVEA toolkit [13] .

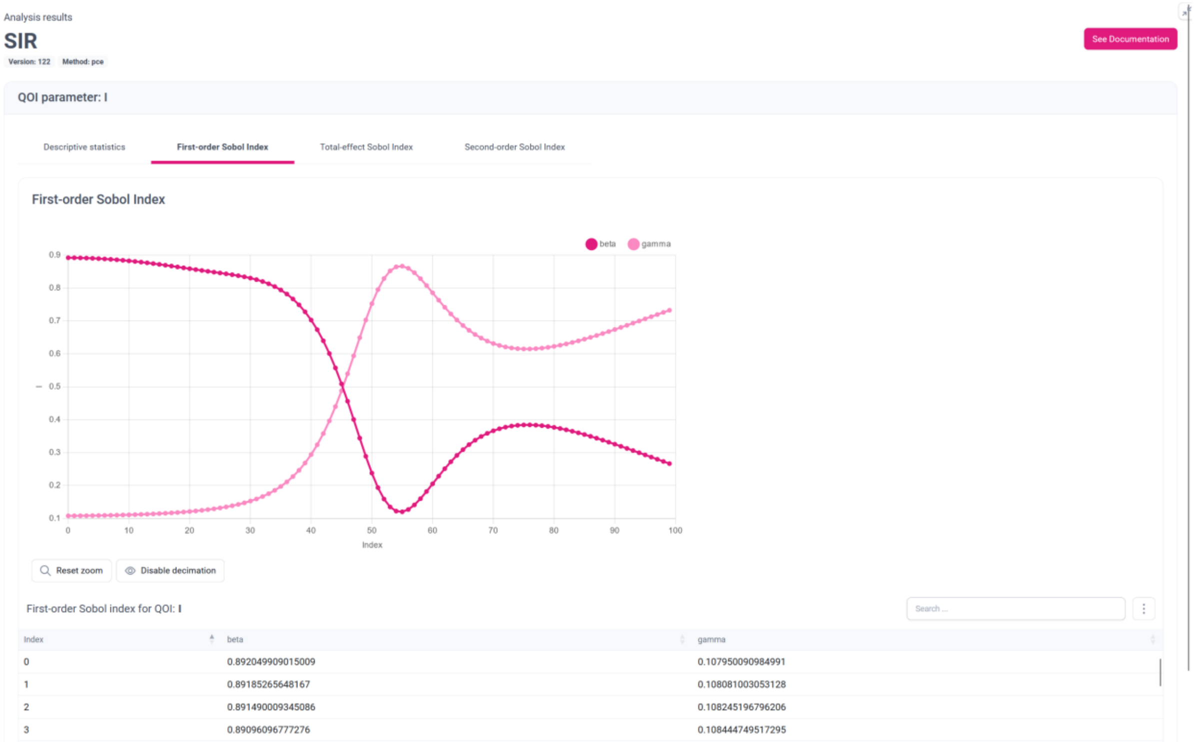


Figure 9. Interactive web page generated by mUQSA for the UQ&SA results analysis

3.3.2 Design and Architecture

Since the previous deliverable D2.4, where the basics of QCG architecture has been described, the core layer of the architecture has not changed. However, a few auxiliary components have been incorporated to fulfil the needs of HiDALGO2. To highlight the modifications in regards to the original architecture introduced in D2.4, Figure 10 presents the up-to-date architecture scheme, with the new components marked with the blue colour.

It is worth noting that two brand new components have been integrated with QCG, namely **Grafana** for monitoring and **Apache Airflow** for executing workflows. In addition, **CKAN** has appeared in the Data Management Systems box as already supported. What's more, **QCG-Agent** has been marked in blue, as it was not previously implemented.

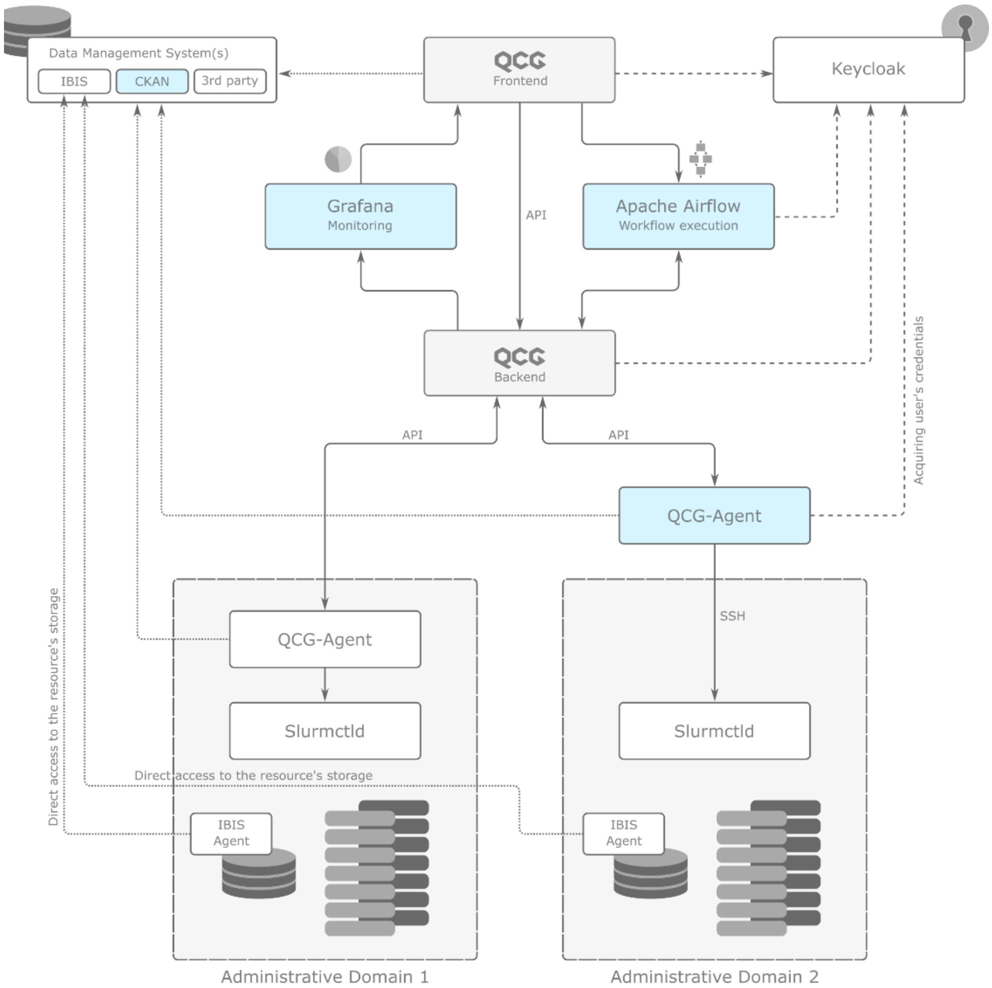


Figure 10. QCG workflow orchestration architecture with new components marked in blue

3.3.3 Current implementation

3.3.3.1 Enabling access to EuroHPC computing resources from QCG-Portal

Realisation

M12 – Proof-of-concept: **realised**

M21 – 1st release: **realised**

Current implementation details

- The off-site SSH-based QCG-Agent has been implemented in order to enable access to EuroHPC resources over the SSH connection.
- The agent requires SSH private keys of users to connect to resources, which are kept internally in QCG.
- The agent has been already integrated with the Altair and Lumi resources.

- It is relatively straightforward to configure the agent to use other Slurm-based Euro HPC resources.
- The QCG-Portal instance using the new QCG-Agent has been deployed on https://qcg.hidalgo2.eu and offered to the HiDALGO2 community.
- Repositories with the code (restricted access):
  - gitlab.pcss.pl/qcg/qcg-ncomp.git,
  - gitlab.pcss.pl/qcg/qcg.hidalgo2.eu.git (deployment scripts)

Divergences from the plan

No significant deviations have been identified, but two key security enhancements are planned for the near future. First, the QCG-Portal needs to be integrated with a dedicated, validated storage solution to securely manage users' private keys for the sites that require traditional SSH credentials. Second, support for SSH certificates should be implemented and made default for all sites that already enabled this modern form of authorisation.

3.3.3.2 Access to many computing resources from a single instance of QCG-Portal

Realisation

M18 – Proof-of-concept: **realised**

M24 – 1st release: **planned**

Current implementation details:

- Implemented support for many Slurms with different configurations,
- The functionality tested with Lumi and Altair,
- The instance deployed on https://qcg.hidalgo2.eu.

Divergences from the plan

No divergences

3.3.3.3 Workflows

Realisation

M18 – Proof-of-concept: **realised**

Current implementation details

- QCG-Backend is able to submit jobs to Airflow (version 2.4.2).
- New QCG Operator for Airflow that allows to submit jobs using QCG API.
- Tested on a local Airflow Executor, plans to move it to the Kubernetes Executor.
- Running in the test instance of QCG-Portal.

Divergences from the plan

No divergences

3.3.3.4 Cyclic tasks

Realisation

M24 – Proof-of-concept: **realised**

Current implementation details

- Technically this functionality is already implemented through the integration with the Apache Airflow and the @schedule mechanism it provides.

Divergences from the plan

The possibility of practical usage of this mechanism is frozen until the issue of the safe access to user credentials without the user interaction is resolved (obviously the problem applies also to the execution of any long-lasting Airflow workflow).

3.3.3.5 New Data Management Systems

Realisation

M15 – Proof-of-concept of CKAN integration: **realised**

M24 – 1st release of CKAN integration: **realised**

M15 – Proof-of-concept of Hadoop/HDFS integration (conditional): **not-realised, planned**

M24 – 1st release of Hadoop/HDFS integration (conditional): **planned**

Current implementation details

- Created hooks for downloading input data from CKAN datasets and for uploading output data to CKAN.
- Accessing CKAN with API Key.
- Automatic compressing and tagging data uploaded to CKAN.
- Integrated with the HiDALGO2 instance of QCG-Portal at https://qcg.hidalgo2.eu/ and tested with HiDALGO2 CKAN at https://ckan.hidalgo2.eu/.

Divergences from the plan

The implementation of support for Hadoop/HDFS has been postponed until the HiDALGO2 Data Transfer Tool is integrated with QCG. Additionally, the existing CKAN integration is expected to be redeveloped using the HiDALGO2 Data Transfer Tool to enhance its capabilities.

3.3.3.6 Visualisation

Realisation

M18 or M21 - Proof-of-concept (with the first tool): **not-realised, planned**

Current implementation details

- Initiated collaboration with SZE aimed to integrate CFDR visualiser into QCG leveraging the built-in support for forwarding desktop applications.

Divergences from the plan

Underlying mechanism for the integration with the visualisation tools is in place. The work has been started to integrate and configure CFDR visualisation tool for the RES pilot, but it is not yet completed. The proof-of-concept solution is expected to be developed within approximately one month after the integration details are settled between SZE and PSNC, likely before M24. The integration of other visualization tools will be carried out based on the pilot's requirements.

3.3.3.7 Tasks execution monitoring

Realisation

M15 - Proof-of-concept: **realised**

M21 - 1st Release: **realised**

Current implementation details

- Developed extendable QCG monitoring subsystem based on Grafana (version 9.3.15) and Redis (version 7.0.11).
- Implemented API for sending monitoring data from running applications.
- Added support for defining monitoring dashboards in QCG templates.
- Created generic scripts for monitoring resources consumption (CPU and memory) and QCG-PilotJob workflow executions.
- Deployed on the Altair cluster and made available to PSNC-registered users through the main PSNC's QCG portal: https://qcg.pcss.plcloud.pl/.

Divergences from the plan

No divergences

3.3.3.8 mUQSA

Realisation

M12 - Proof-of-concept: **realised**

M18 - 1st Release: **realised**

Current implementation details

- mUQSA User Interface developed on top of QCG-Portal as Custom Application GUI.
- EasyVVUQ and QCG-PilotJob used to perform UQ/SA calculations on HPC resources.
- Several UQ/SA algorithms supported, including Monte Carlo, Polynomial Chaos Expansion and Stochastic Collocation.
- Implemented dedicated monitoring facilities for mUQSA (dashboard, application monitoring hooks).
- Deployed on Altair cluster and made available to PSNC-registered users through the main PSNC's QCG portal: https://qcg.pcss.plcloud.pl/.

Divergences from the plan

No divergences

3.3.4 Development Roadmap

Although there were no significant modifications in the initially proposed plan, there are a few divergences that need to be reported. The explanation of these changes is summarised below, while the updated version of the roadmap can be found in Appendix 2.

3.3.4.1 Enabling access to EuroHPC resources

Due to the complex and not yet unified access to EuroHPC resources, the process of integration of QCG with individual resources will likely be longer than expected. The particular problems relate mostly to the divergences in authentication/authorisation mechanism (e.g. Leonardo users SSH Certificates, while other sites use SSH keys). Depending on the progress of unification of the access, HiDALGO2 will adapt to the established mechanism or implement its own solution. The planned extension of the task is estimated for 6 months, so the task will end at M36 of the project.

3.3.4.2 HiDALGO2 Data Management Systems

The development of the generic HiDALGO2 Data Transfer Tool, while highly beneficial, has impacted the original schedule for integrating DMSes with the QCG system. As a result, although the initial release of the integration has been completed, an additional release is planned for M30 of the project. By that time, QCG will be fully integrated with the HiDALGO2 Data Transfer Tool, enabling seamless support for all DMSes managed by this mechanism.

3.3.4.3 Visualisation

The planned for M21 proof-of-concept release of the working integration of QCG with visualization tool has not been achieved so far, but it is expected to be ready for the M24, where the initial integration with the first tool, namely RedSim, for the needs of the first Pilot, namely RES, is expected. The moderate delay allowed us to align the activities in these tasks with the development schedules of individual Pilots and it does not negatively affect the project implementation.

3.3.4.4 mUQSA

Positive feedback from the project partners and participants of several events, encouraged us to put additional efforts to development of mUQSA in the past year. The achieved progress allow us to quite safely speed-up the release of the next version of the platform and schedule it on M27.

4. Energy Monitoring and Optimization for HPC

Energy consumption has become as important as performance in the HPC area. Historically, the primary focus in HPC design has been on maximizing computational power—how fast and efficiently systems can perform tasks (e.g., simulations, solvers, forecasting, etc.). However, with the rise in energy costs and growing environmental concerns, reducing energy consumption without sacrificing performance has become a priority.

Nowadays, tracking the energy usage and adjusting the power consumption in HPC systems can maintain high performance while minimizing their environmental footprint and energy costs. The HPC market not only focuses on performance (see the TOP500 [14]) but also on efficiency (see the Green500 [15]).

Energy and Power Definitions

To fully appreciate the importance of energy efficiency, it helps to understand the distinction between energy and power. Energy refers to the total amount of work done. In computing, this is the total energy consumed by a system while it runs a particular task or job. Power, on the other hand, is the rate at which that energy is consumed.

In HPC systems, energy consumption is cumulative, depending on how long the system is operational, while power represents the instantaneous consumption of resources. Both metrics are vital for understanding the operational efficiency of an HPC system. Optimizing an HPC system means balancing these two: minimizing the energy use without compromising the power needed to get the job done efficiently.

Cost Efficiency and Environmental Impact

HPC systems, like the ones used in research labs or by big tech companies, are expensive to operate—both in terms of hardware and energy use. A supercomputer like the Leonardo at Cineca in Italy (top 9 of TOP500), which can perform up to 241 petaflops, requires about 7.5 megawatts of power to run at full capacity. To put that into perspective, that would be enough energy to power a small town.

In addition to the financial burden, there's the environmental impact. The more energy these systems consume, the higher their carbon footprint. As global demand for computing power rises, HPC centres are under pressure to operate more efficiently. Techniques like dynamic voltage and frequency scaling (DVFS) allow the system to lower power use when full performance is not required, reducing waste [16] .

Performance and System Lifespan

Energy optimization is closely linked to the performance and longevity of HPC systems. Power management techniques, such as DVFS, can dynamically adjust the power usage of the system components based on the current workload. While these techniques save energy, they also influence system performance. Reducing power may lead to lower computational throughput or longer execution times for certain tasks. However, it can also reduce the heat generated, which prolongs the lifespan of system components and decreases the need for extensive cooling systems. This balance between energy savings and performance is crucial for the effective operation of HPC systems.

Tools for Energy Monitoring and Optimization

There are multiple tools for energy monitoring and power tuning. Most of these tools give detailed insights into where power is being consumed, helping the system administrator to make real-time or programmed adjustments. Those include Intel RAPL, NVIDIA NVML, LIKWID or Eviden EO. In the following you can have a brief description of each:

- Running Average Power Limit (RAPL [17]): Intel's RAPL is a well-established tool that offers hardware-level control over power consumption in modern Intel processors. For example, RAPL can impose power limits on processors during periods of low demand, reducing energy consumption without shutting down the system or degrading performance too severely.
- NVIDIA Management Library (NVML [18]): On the GPU side, NVML provides extensive monitoring and control capabilities. NVML can report power usage, temperature, and fan speeds, allowing fine-tuning of GPU operation to optimize energy efficiency for specific workloads like deep learning or large-scale simulation.
- LIKWID [19] : It is a lightweight toolset designed for Linux systems that provides both performance and energy monitoring features. Its simple interface allows users to monitor CPU and memory performance, apply DVFS, and tune cache and memory settings, all in pursuit of higher energy efficiency.
- Energy Optimizer (EO [20]): Eviden EO is a tool designed for enforcing power capping on compute nodes. The tool includes:
    - The ability to define collections of power capping rules that can be toggled on or off as needed.
    - The capability to allocate a power budget to a group of components, which is then automatically converted into individual capping rules, ensuring a fair distribution of power (based on a percentage of nominal power consumption).
    - For select equipment, it offers experimental determination of the minimum effective power cap—the most stringent cap that still allows the component to function. This is critical, as overly aggressive capping can render components unresponsive.

4.1 Requirements

Some requirements aim to extend the functionalities of the EO tool, while others focus on developing models for reporting, comparing, visualizing, or improving the energy efficiency of HPC systems:

- **REQ-EO-001:** Energy optimization must not significantly limit system performance.
- **REQ-EO-002: E**nergy optimization must be aware of the target application.
- **REQ-EO-003:** Classify the behaviour of application to better understand when it is worth to limit power.
- **REQ-EO-004:** Define the power cap based on the application's behaviour (e.g., computational or memory limitations).
- **REQ-EO-005:** Build a model to predict the target application's behaviour.
- **REQ-EO-006:** Report energy consumption across different application runs.
- **REQ-EO-007:** Report energy consumption across different user applications within the same system.
- **REQ-EO-008:** Build a model to predict the energy consumption of user jobs.
- **REQ-EO-009:** Create a recommendation system to limit job energy consumption.

4.2 Functional Specification

This contribution to HiDALGO2 project focus on the development of energy solutions that increases the functionality of the Eviden Argos Suite. It splits these energy solutions into four main tools: the Power and Performance Estimator (PPE), the Job Energy Comparator (JEC), the Job Energy Predictor (JEP), and the Job Energy Recommender (JER).

Power and Performance Estimator

The PPE module is planned to be integrated into the Eviden EO tool. The Eviden EO tool can cap system power depending on the nature of the target application, whether it is memory or compute bound. This is because when the system is wasting time on non-CPU instructions (e.g., I/O operations or memory stalls), the CPU can be downclocked to reduce both performance and power consumption. Generally, reducing system's power will increase the final energy consumption of the system, this happens because energy is calculated from system's power but also from task duration. Then, usually downgrading power increases the task time spent. However, in certain situations, the overall energy consumption is reduced, when reducing the power (e.g., memory-bound applications).

Currently, the Eviden EO tool allows the user to define the ongoing application bound (Compute, Memory, or Mix bound). The PPE module will automatically provide the application bound as a tag to the Eviden EO tool. PPE will integrate AI models for bound prediction, and these models will be fed with a data pipeline ingested with data from different tools. The data is planned to be collected from ahead-run resources and post-run resources. Ahead-run data will include Slurm configuration or data generated from previous job runs. Post-run data includes I/O, performance counters, MPI, or system metrics.

The PPE module will learn from post-run metrics, but ahead-run data will be the input for the tag prediction.

Job Energy Comparator

The JEC will be an independent tool for comparing and visualizing Eviden EO traces. The Eviden EO tool collects energy traces from applications, which are later saved and describe the application's energy footprint. The JEC component will be responsible for taking these energy traces and comparing and visualizing them. To achieve this, the key is to define how similar two energy footprints are by using clustering or time-series techniques. Subsequently, the component will visualize this comparison among traces. The objective of JEC is to help understand how different application behaviours affect energy consumption.

Job Energy Predictor

The JEP is intended to be an energy predictor for system jobs. The main idea of the JEP module is to implement an AI model to predict job energy consumption. Similar to the PPE, data produced before and after job execution will be also used, as well as the output from the Atos EO tool, to validate the energy prediction.

Since, at runtime, only Slurm parameters can be used, alternative ways to gather data from the ongoing job to improve model predictions will be explored. This may include using a knowledge database to find similar jobs.

Job Energy Recommender

The JER module will help users configure various parameters when launching a job to improve energy consumption without compromising final performance. The recommendations will specifically focus on setting Slurm parameters such as the number of nodes, time limit, and amount of memory, enabling the automatic energy optimization across the system without compromising performance.

4.3 Design and Architecture

Figure 11 shows the architecture for the energy modules described in the previous section. The flow starts when the user access to the Eviden Argos Suite via GUI and API. Then, Argos launches the user job in the cluster, typically using the Slurm job scheduler, which is responsible for distributing the workload across the nodes of the cluster.

Once the job starts running in the cluster, the job will be profiled by collecting various metrics (e.g., energy, MPI, I/O, etc.). After the job finishes its execution, all the collected metrics will be processed and integrated into a common format (Argos DB collection).

Subsequently, the JEC, JEP, JER, and PPE modules will use that data to train their models, provide reports or interact with the user (e.g., the recommender).

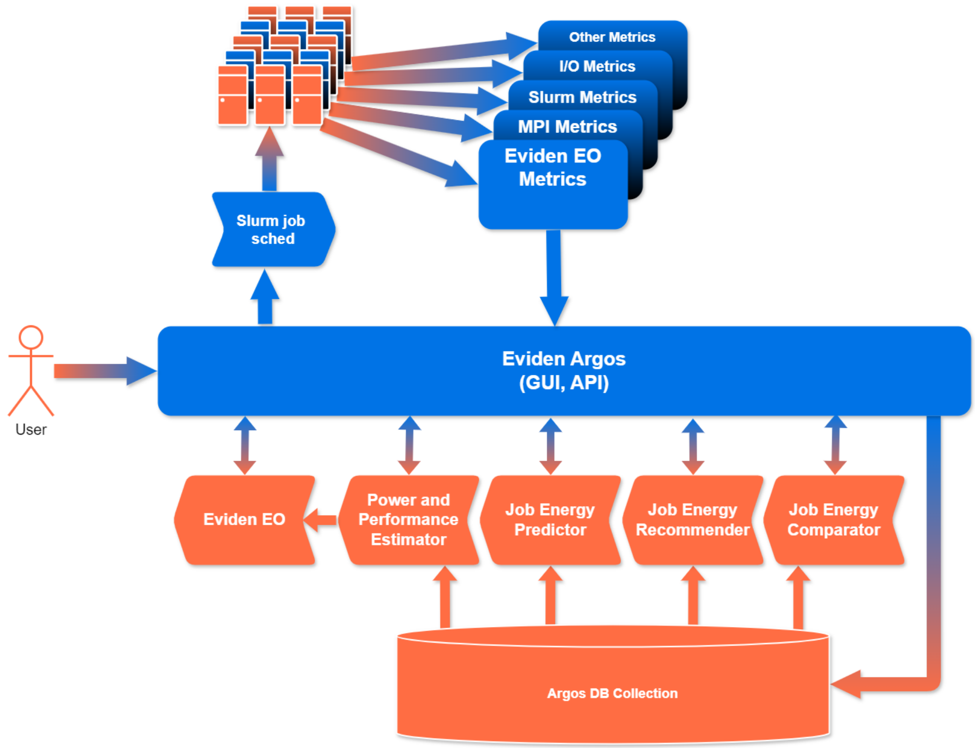


Figure 11. Architecture of the Argos Energy Monitoring and Optimization framework

4.4 Current Implementation

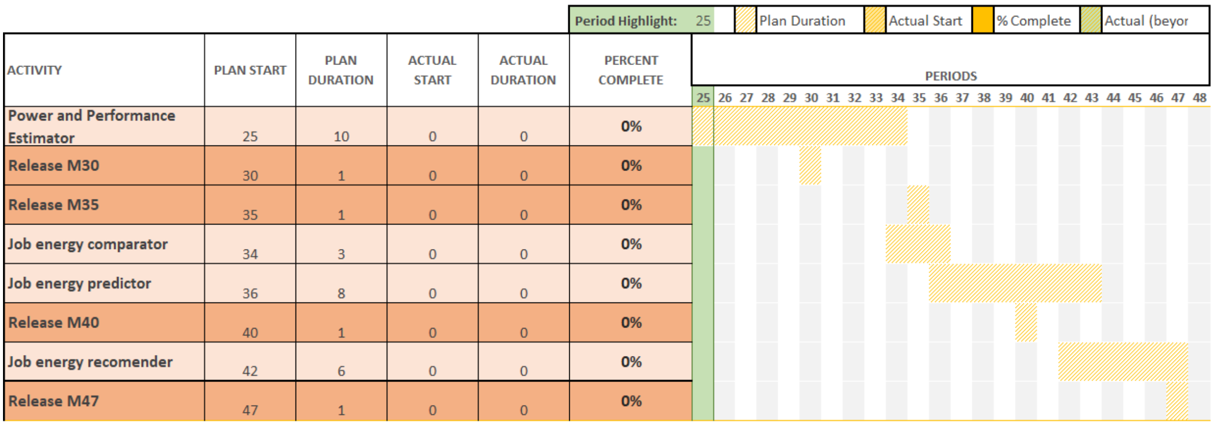
The development of the extensions to the Argos Monitoring and Optimization framework will start in M25. Therefore, this document accounts for its planning in the next section.

4.5 Development Roadmap

In relation to the development roadmap, Figure 12 shows the development plan. Since this task starts in month M25 and ends in month M48, there are 23 months to complete it, which is reflected in the Gantt diagram depicted in the figure.

There are four main tasks in the roadmap: PPE with a ten-month duration, JEC extending over three months, JEP accounting for eight months, and JER with a duration of six months.

Regarding milestones, four releases have been established: Release M30, Release M35, Release M40, and Release M47. Each release corresponds to the named month. Release M30 will include the first version of the PPE module. Release M35 will include the final version of the PPE module. Release M40 will feature the JEC component and the first version of the JEP module. Finally, Release M47 will include the second version of the JEP module and the JER components.



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Power and Performance Estimator	25	10	0	0	0%
Release M30	30	1	0	0	0%
Release M35	35	1	0	0	0%
Job energy comparator	34	3	0	0	0%
Job energy predictor	36	8	0	0	0%
Release M40	40	1	0	0	0%
Job energy recomender	42	6	0	0	0%
Release M47	47	1	0	0	0%

Figure 12. Roadmap for the Energy Monitoring and Optimization framework

5. Component Integration and Deployment

HiDALGO2 updated and rewrote its strategy for Component Integration and Deployment in the revision of Deliverable D2.4. However, as this section is a key part of the project goals, at some points, D2.4 is not referred back to, but repeated with minor updates, information carrying forward from the revised edition of D2.4.

5.1 HiDALGO2's Strategy for CI/CD

HiDALGO2's strategy for realising a reliable integration and deployment of its components is to differentiate between three broad CI/CD systems, with three distinct objectives:

- **System 1**: "Services CI/CD System", for building and deploying project level software and web services, for internal purposes.
- **System 2**: "Pilots CI/CD System", the internal CI focused system, specially crafted by each use case for itself, to automate and test the build quality of its codes and their released. These Pilot CI systems are extended to also incorporate CD, where use cases are deployed to select HPC infrastructure, including EuroHPC infrastructure. This is done on HiDALGO2's own initiative, without burdening or depending on official EuroHPC CD system.
- **System 3**: "EuroHPC CI/CD System" to bridge the gap between internal and external (public-facing) deployments, with a special focus on automatic deployments to EuroHPC JU Systems, by interfacing with the common integration and deployment platform envisioned by the EuroHPC JU, as part of the CASTIEL2 CI/CD activity.

These different systems reflect the highly customised toolchains and workflows essential to ensure consistent and continuous component integration across several project layers and use cases. These can range from very specific HiDALGO2 internal use cases, such as the reliable deployment of web services, such as the project website or Moodle, the teaching platform, to the compiling, building and deploying of simulation codes on HPC infrastructure.

5.2 Services CI / CD System

This section introduces the CI/CD system for HiDALGO2 services delivery.

5.2.1 Requirements

Please refer to Deliverable D2.4, Section 5.1.1 for requirements related to the Services CI/CD System.

5.2.2 Design and Architecture

For details on the overall design and architecture, including the decisions behind the software toolchain that make up the core components of the Services CI/CD system, please refer to the Deliverable D2.4 Section 5.1.2. However, a streamlined recap of the architecture is provided through the next section, with the design of each part of the pipeline shown in Figure 13 described in more detail along with the current implementation.

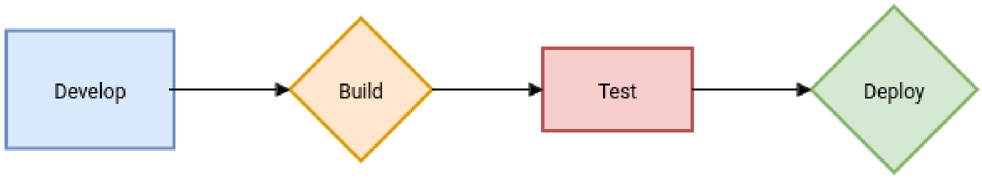


Figure 13. Services CI/CD Overview Pipeline

5.2.3 Current implementation

The design around the Services CI/CD system is meant to ensure a consistent, idempotent and reliable method for deploying HiDALGO2 web services. This is done to particularly ensure, that whenever there is a problem or fault arising, either due to an update, or due to a bug in the running of the web service, it can be restored it to a stable state through automated means. The level of automation for deploying web services is, due to the nature of the problem, different than for actually developing a web service. It also depends from service to service, and the use case. HiDALGO2 currently employs a flexible system, where the services that can benefit from it are deployed through automated means, while services that require a great deal of manual maintenance or configuration, are allowed to be deployed and maintained classically.

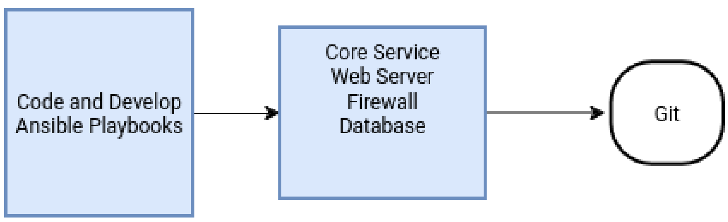


Figure 14. Services CI/CD Pipeline Development stage

As shown in Figure 14, the **development** stage of the automation for HiDALGO2's web services consist of developing Ansible playbooks that ensure a reliable and scriptable deployment for the core web service, as well as all the supporting services needed in order to successfully host the core web service. For example, for

HiDALGO2's learning and teaching platform, the software service, "Moodle", is deployed, but Moodle also needs a Web Server to host it, a database to store the data from the courses and users reliably, as well as a firewall to protect it. The Ansible playbooks and roles that automate the deployment of these components together to deploy and server Moodle are committed as they are developed to a Git solution hosted at the partner institutions of HiDALGO2. In this case, they are stored in USTUTT's *Forgejo* [21] repositories. There will be a point in the roadmap, where all such repositories will be collected in the official *Bitbucket* repository of the project.

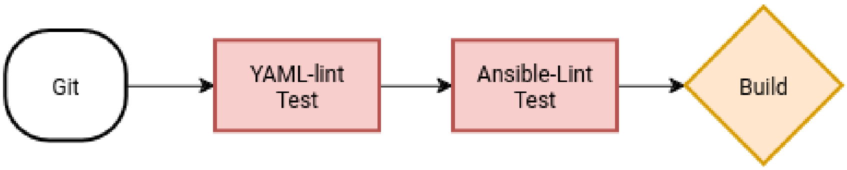


Figure 15. Services CI/CD Pipeline Build Stage

The **Build** stage of the Ansible playbooks consist of a number of different "static" testing stages (see Figure 16). These static tests can be thought equivalent to "unit" testing in traditional code development terms. The traditional form of unit testing is not required, as each Ansible playbook code statement is a kind of verification or unit test in itself. (For example, a typical statement in an Ansible playbook is, "confirm that this service is running").

The official tool, "YAML-lint" is applied on the playbook code, to see if they pass the required YAML specifications. Then, the specialised "Ansible-Lint" tool runs its tests, to see if all official Ansible best practices are followed by the playbook codes.

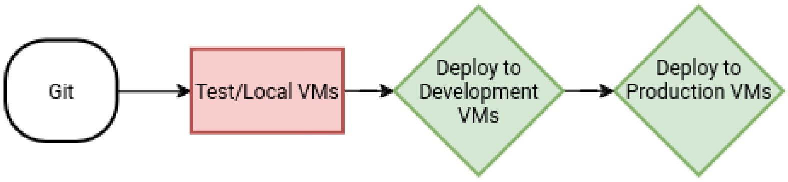


Figure 16. Services CI/CD Pipeline Testing and Deployment Stage

The **testing** stage of the Services CI/CD pipeline is separate from the static code related tests described above. They are meant to test the functionality of the deployed services as a whole, and can be thought of as "integration" tests. When developing the playbook, the playbooks are tested out on local virtual machines, and when they are ready for more deployable testing, they are first deployed to "development" VMs. This is especially useful in terms of updates or bug fixes to services: the original service can continue functioning and serving its users from the "production" VM, while the code upgrades or fixes are first tested out safely on the development VMs. These development VMs replicate the environment of the production VMs as closely as possible, so that all changes behave the same across the development and production infrastructure.

The final stage of the pipeline, the actual **deployment** of the production version of the web service, can be carried out through the manual confirmation and execution of the Ansible playbook by the designated system administrators for HiDALGO2, or through automated means triggered by a git push through the configured CI/CD platform. Currently, due to security concerns (such as unauthorised remote triggering of deployments by less privileged or malicious users), the preferred method for the deployment to production VMs is through a manual confirmation and triggering of the deployment by the HIDALGO2 system administrators. However, an automated deployment is being tested out on development VMs through *Forgejo* [21] Runners, that are part of the *Forgejo Git* offering. They are configured, once the build and development tests have passed, to check out and deploy the code on the production servers. HiDALGO2 is also investigating alternative options such as *Woodpecker CI*[22] and *Semaphore UI[23]* , along with Redhat's official offering, *AWX*[24] , to improve the efficiency and security in the automation and deployment of the project's web services.

5.2.4 Development Roadmap

This section provides an update on the implementation roadmap of the CI/CD system for HiDALGO2 service delivery. Some of the points previously slated to be included in 2025 are already incorporated in some services pipelines. However, some other points are to be delayed, especially regarding the scope and extent of coverage of services.

- Pushing Code for Services by PSNC and USTUTT to shared common code repository, taking care of sensitive information such as passwords and secrets, as well as institute specific security regulations. (Planned, release R12.2025).
- Have dedicated repositories for each service, as well repositories to provision Infrastructure resources for each service. (In Progress, release R12.2025.)
- Have dedicated branches or folders/directories for each Service, to control integration and deployment on Development and Production instances. (In Progress, Updated to R12.2025, to align with previous point.)
- Integrate a code quality component to the CI/CD pipeline (of suitable services), such as the linter (the code checking tool) proofs the scripts for any errors in the code syntax. (Updated to In Progress, R03.2025).
    - A developmental pipeline incorporating *ansible-Lint* is shown in Section 5.2.3.

- Integrate automated testing of (suitable) deployed services. This could be achieved e.g., by using the *Ansible Molecule* framework [25] and composing testcases for each (suitable) service. (Planned, R06.2025)
- Use (platform dependant) Runners, or alternatives, in combination with Ansible and other repository scripts to carry out the Integration tests, as well as to deploy development VMs. (In progress, R6.2025)
    - A developmental pipeline using the updated *Forgejo runners* is shown in Section 5.2.3. Alternative solutions such as *Woodpecker CI* and *AWX* will be considered over the next year.

5.3 Pilots CI / CD System

This section introduces the CI/CD system for building and delivering the HiDALGO2 simulation codes, that are of course, the heart of the HiDALGO2 project. Each pilot has a different purpose, a different developmental history, a list of dependencies and library frameworks, and therefore, a unique process for building the simulation codes.

5.3.1 Strategy

The overall HiDALGO2 strategy for Pilots CI/CD remains the same as in Deliverable D2.4, Section 5.2.2. However, as development has progressed, it has. been learnt that the uniqueness of each pilot means that the initial, or most closely related CI (continuous integration) stage of development is crucial for the end delivery and deployment of the codes on HPC systems. The resultant builds from these base-level CI systems are then taken and passed to higher, project level CI/CD systems, such as the project git repositories. From here, they would be mirrored or transferred to external CI/CD systems, such as CASTIEL2/EuroHPC level repositories or container registries.

Hence, in the next sections, it is taken a closer look at this fundamental CI system, with regards to each unique pilot code. Some of the CI systems of the pilots already extend to full CI/CD systems, deploying on EuroHPC machines, even without linking to external CI/CD systems. This does not mean that they are, or will not be, compatible with future CASTIEL2/EuroHPC systems, but means that they are also able to deploy to EuroHPC machines without depending on such agreements or externally provided systems.

5.3.2 Requirements

Please refer to Deliverable D2.4, Section 5.2.1, for the formal initially gathered requirements for the CI/CD system when it comes to pilot codes delivery.

5.3.3 Urban Air Project

5.3.3.1 Current implementation

The current implementation of the MathSO CI/CD pipeline is extensively described in the resubmitted version of the deliverable, D2.4. There are two important development points, that have been addressed since.

First, SCP based in-connection file transfer has been implemented using the AsyncSSH library to address incoming and outgoing connection limitations to and from the HPC. This way transferring singularity recipes and images are done through the MathSO WFO.

Second, a simple singularity application is created to demonstrate the CI/CD functionality of the WFO. Furthermore, documentation and additional tutorial-like are in work.

5.3.3.2 Development Roadmap

The development of the MathSO CI/CD system is planned along the following roadmap:

- Documentation of the current implementation, including the current UAP integration
  Timeline – M27
- Integrating and testing the system on the HiDALGO2 training cluster, to facilitate trainings – M30
- Investigation of possibilities for integrating EuroHPC supported CI/CD systems, including triggering of image creation and supporting image storage and deployment.
  Timeline – M30
- Improved GitLab integration for streamlined CI and blueprint installation
  Timeline – M36
- For pilots deployed from the MathSO portal, integrate to the MathSO CI/CD system, if the pilot requests.
  Timeline – M36
- Test and debug CI/CD system based on pilots' reports.
  Timeline – M48

5.3.4 Urban Building Model

The Urban Building pilot utilizes the Feel++ framework, supported by a robust CI/CD framework that facilitates efficient development and deployment.

5.3.4.1 Current implementation

The development and deployment of KUB builds on top of the Feel++ CI/CD framework. It employs GitHub Actions and Docker: GitHub Actions automate real-time workflows to compile, test, and validate code changes, facilitating rapid development cycles and ensuring code quality. On the other hand, Docker provides a containerized environment that encapsulates Feel++ and its dependencies, ensuring consistent operations across diverse computing environments. These Docker images, customized for various system requirements, are maintained on the GitHub Container Registry (ghcr.io) to accommodate multiple deployment scenarios.

The CI/CD workflow, see Figure 17, is crucial for efficiently integrating and deploying updates across all projects that utilize the Feel framework. The workflow leverages various main ingredients of GitHub Actions features.

- **Pull Requests and Merges** Triggering CI to verify that new code integrations meet all tests and standards.
- **Graphical User Interface (GUI)** thanks to workflow_dispatch events: Enabling developers to manually trigger pipelines through a GUI, which facilitates rapid deployment or testing.
- **Scheduled Runs** Conducting regular updates and maintenance checks to ensure continuous system integrity and responsiveness.

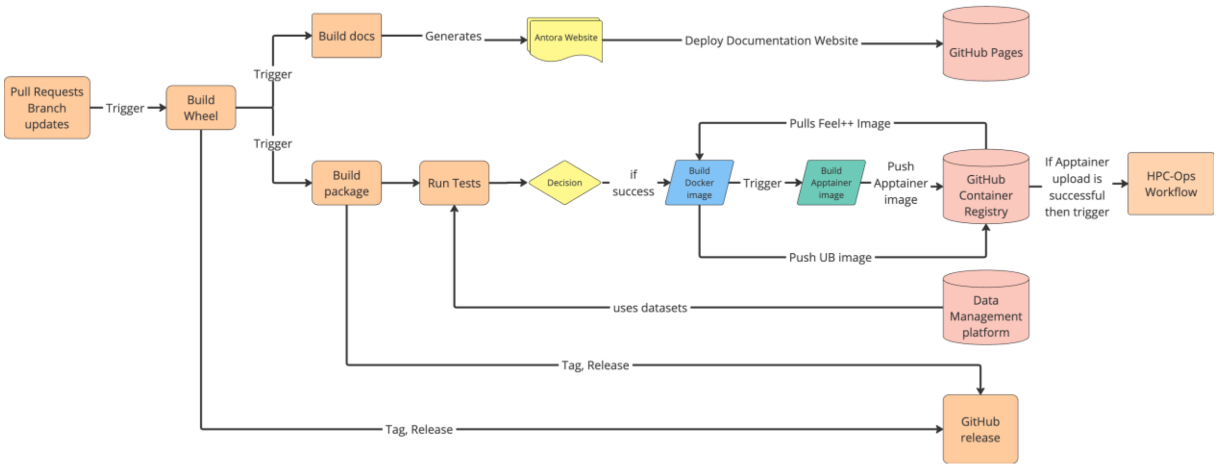


Figure 17. CI/CD workflow for Feel++

The next step is the workflow for high-performance computing applications incorporates specialized HPCOps (HPC DevOps) practices that ensure the software performs consistently across various HPC systems.

The tools and strategies for HPCOps are *(i)* **Reframe-HPC**: Utilized to define and manage systematic benchmarks that are reproducible across different HPC environments, facilitating the testing of performance and scalability; *(ii)* **SLURM**: Employs its REST API if available, *e.g.* on MeLuXiNa, otherwise scripted SLURM usage for CI/CD for scheduling and managing jobs on integrated HPC systems, allowing programmable job submission and monitoring directly from CI workflows; and *(iii)* **Apptainer**: Ensures that Docker containers can be deployed securely and efficiently in HPC settings, supporting portability and consistency.

Integration with EuroHPC JU supercomputers such as LUMI, Karolina, Meluxina, Discoverer, Vega, and Leonardo enhances the capability to perform large-scale simulations and check the parallel properties and correctness of our pilot. The operations include automatic testing that triggers larger-scale tests on designated HPC nodes once new changes are integrated and verified by standard CI/CD pipelines.

Regarding monitoring and reporting, performance results from these operations are automatically captured and uploaded to the data storage system, such as the performance reports.

Upon each upload event, the CI/CD pipeline of the respective repositories is triggered to fetch and parse the reports. These reports are then aggregated over time to track the progress of the pilot and ensure no regressions occur in both application's execution and quality of the results. Finally, figures are dynamically generated and published on the corresponding websites.

Figure 18 illustrates the continuous benchmarking workflow (HPCOps) that was set up for Ktirio Urban Building and the post-processing was done with reports.

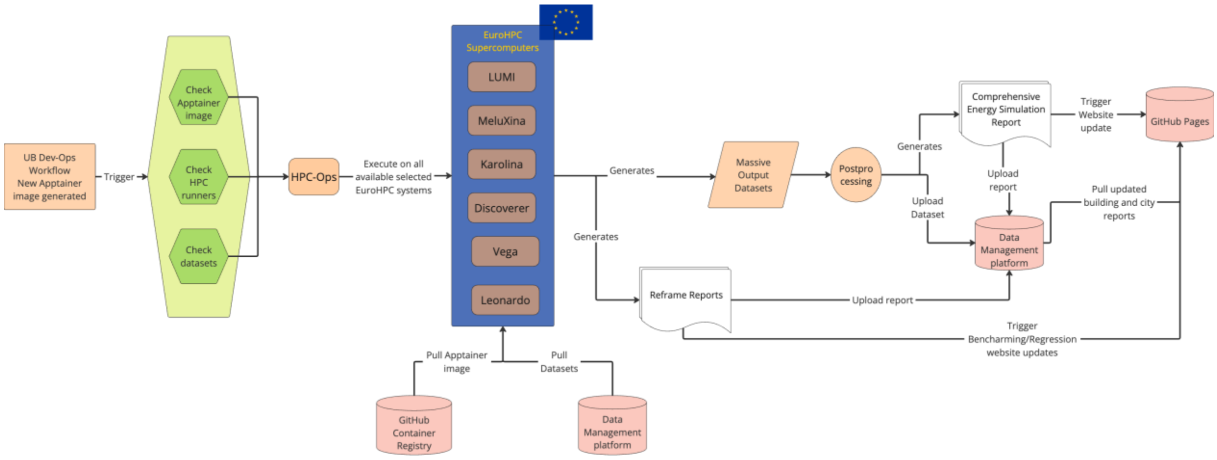


Figure 18. Continuous benchmarking workflow for Ktirio Urban Building

5.3.4.2 Development Roadmap

Our initial choice to build on top of containers our application has been very rewarding, as efforts are focused is on improving the application and not on spending too much time on porting to EuroHPC infrastructures. However, it was needed to tackle several challenges among which GPU porting and specific interconnect configuration such as slingshot.

Our next step to improve our framework is to switch to Spack packaging system.

Our backbone framework Feel++ has been successfully ported to Spack including with AMD GPU configurations. The package and the configuration can be easily fine-tuned thanks to Spack.

A full Spack configuration for Ktirio Urban Building should be ready during the first quarter of 2025, probably before. Moreover, thanks to Spack, containerization is readily available.

The plan is thus to have fine-tuned Apptainer images built from Spack for EuroHPC systems including GU architectures and that the CI/CD/CB workflows run with them.

Finally, about the same period, the design, implementation and deployment of the benchmarking analysis platform should be finished as well as the Ktirio Urban Building reporting site which are updated through our CI/CD/CB workflows.

5.3.5 Renewable Energy Sources

The Renewable Energy Sources is using CI/CD framework based on Gitlab and Gitlab runner for efficient development and deployment.

5.3.5.1 Current implementation

The development and deployment of RES is based on Gitlab and Gitlab runner framework. It automates the workflow to compile, test, and validate code changes in HPC environment. The CI/CD systems is working for Altair HPC system at the moment. The high-level overview of CI/CD workflow is depicted in Figure 19.

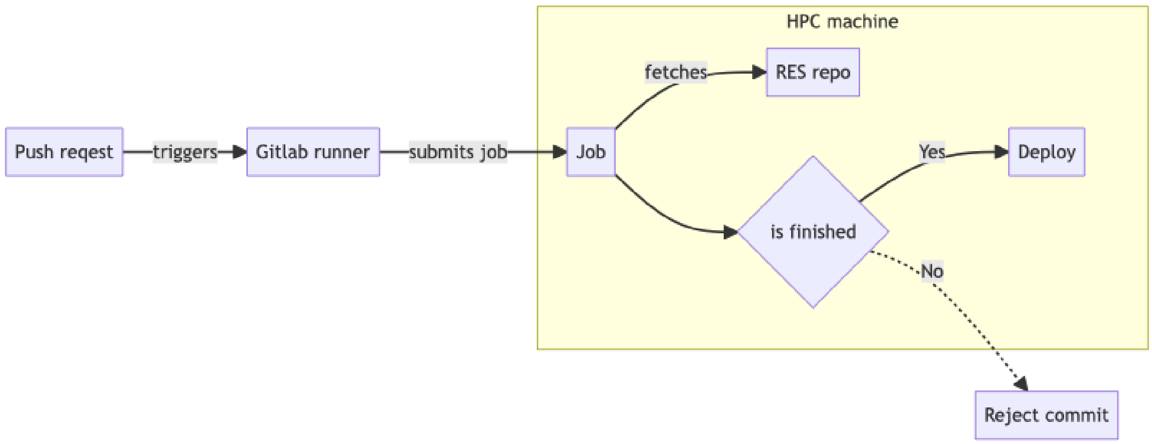


Figure 19. RES CI/CD workflow

Push request of new functionality of change in code triggers Gitlab runner. It is then performing following steps:

- setup_environment – creates unique directory – commit's hash – where repos are pulled

- schedule_task – submits new HPC job to test code changes
- check_status – check for HCP job status
- collect_artifacts – collects artifacts concerning job execution
- check_state – checks for last knows status change

When the job finishes with success, it is deployed on HPC machine to be used by the users. On failure, the Gitlab runner pipeline is rejected. Work is in progress to reject the commit on production environment in case of an execution failure.

5.3.5.2 Development Roadmap

The next step is to apply CI/CD workflow to EuroHPC sites. The current deployment is based on compiling the source code fetched from Gitlab repository. Supporting many different HPC sites with many different versions of programming environment is challenging, therefore providing containerised version of RES application is in progress. There are still many challenges related to complex workflow or RES application, though. To address them, Apptainer is dropped in favour of docker. The RES pilot will be built in layers, where layer 1 will hold generic MPI and Python environments, layer 2 is to hold 3rd party libraries used by RES, and layer 3 will contain RES modules.

5.3.6 WildFires

5.3.6.1 Current implementation

MeteoGrid uses its main internal development workflow in HiDALGO2 based on GitLab.

Versions in git behave like forks in time. Each instant in time is a composite of the previous time plus any changes that have occurred.

These ‘timelines’ mark the ones that are put into production in each installation, closed versions, etc.

Git makes it very easy to incorporate changes from one branch to another, either one at a time or in their entirety. The most commonly used mechanism for advancing branches marked as stable (those in production) is as follows.

- Each new feature is developed starting from the stable branch to which it will be applied.
- It is possible to launch several new features simultaneously, by several different people.
- When the feature is finalised, it is incorporated into the stable branches where it is needed.

- Normally, all new features are provided with tests that make sure the feature is not lost.

By using Gitlab, some help will be available in the form of a user interface and automatic processes:

- Branch view, along with the result of your tests.
- Request flow for adding changes to stable branches.

And, if properly configured:

- Execution of builds, testing and packaging for production release.
- Sending of installation packages to production and subsequent release of the software in production.

5.3.6.2 Development Roadmap

On the recommendation of the Discoverer technical team, the support of the EPICURE project [26] has been requested to analyse a deployment method for replication. It has been granted with support services and 6 months support has been requesting to establish the deployment methodology.

5.3.7 Material Transport in Water

5.3.7.1 Current implementation

As part of the waLBerla framework, the MTW-pilot makes use of waLBerla's established and fully automated CI/CD/CB pipeline. This pipeline is currently hosted solely on local resources from FAU and investigates efficient functionality in terms of node level performance. A visual representation of its work flow is given on Figure 21. To extend convenient deployment on EuroHPC clusters and scaling performance information for various hardware architectures, FAU concentrates to gather deployment and performance information for the various EuroHPC JU clusters with their Git based "Research Data Management" (RDM) Database, which is depicted in Figure 20. This collection forms the foundation for CD and CB of waLBerla and the MTW pilot in context of HiDALGO's CI/CD system, as it will streamline the tasks of building and running the application, as well as gathering, post-processing, and storing produced benchmark or simulation data. The integration of the RDM with EuroHPC resources into waLBerla's CI/CD/CB work flow is depicted on Figure 22.

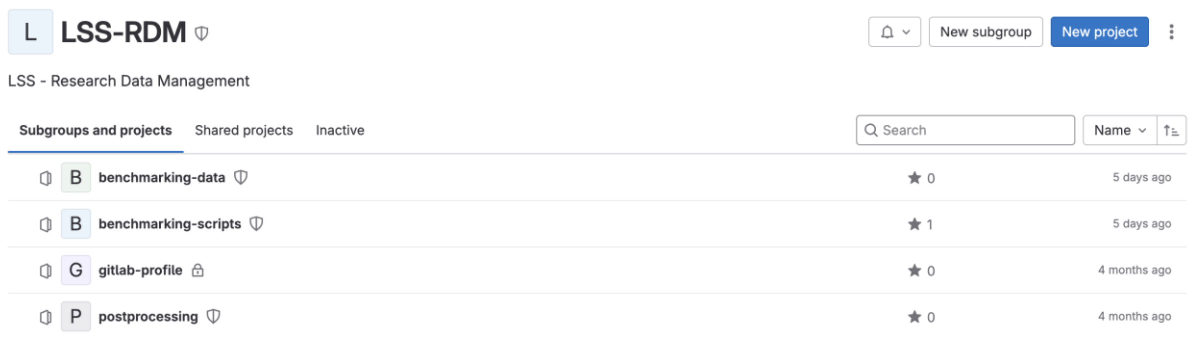


Figure 20. Screenshot from RDM Git Group from FAU to ease deployment and benchmarking on EuroHPC resources

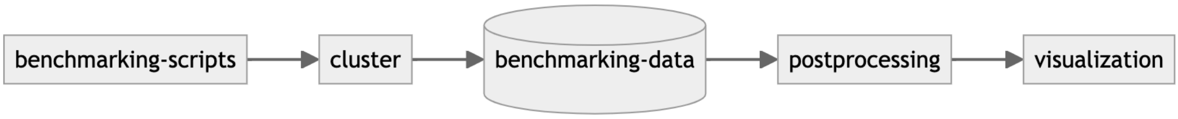


Figure 21. Intended automatic workflow of the RDM pipeline

The final paragraph provides an overview of FAU's CI/CD/CB capabilities. The following section will focus more independently on the MTW pilot's CI/CD system.

The current CI/CD pipeline for the MTW pilot is built on GitLab's CI functionality, with locally hosted runners managing automated builds, tests, and benchmarks. The pipeline is operational on FAU's local computing resources and is designed to ensure that each code change undergoes thorough testing and benchmarking across multiple compilers and hardware configurations. The ultimate goal is to guarantee robustness, compatibility, and performance, although certain aspects, particularly the extension to EuroHPC systems, are still in development and are also used in the CEEC CoE [27] .

CI Infrastructure

The pipeline employs the GitLab CI to automatically build and test the software whenever changes are pushed to the repository. Using a variety of compilers—such as GCC, Clang, Intel, Intel oneAPI, and AppleClang—the software is validated across different environments, ensuring portability and addressing potential compiler-specific issues. The tests range from unit tests to performance benchmarks, identifying any issues early in the development cycle.

GPU-specific tests are also included to verify the correctness and efficiency of parallel processing. These tests ensure that GPU-accelerated components of the software run as expected and help detect potential bottlenecks related to memory management or load balancing.

Continuous Deployment (CD) and Script Collection

A key development in our CD implementation is the creation of a comprehensive set of installation and execution scripts. These scripts form the foundation for automating the deployment and benchmarking of the MTW pilot on local and, in the future, EuroHPC systems. The scripts facilitate consistent installation and configuration processes across different clusters, streamlining the deployment of the MTW pilot and enabling continuous benchmarking (CB) on diverse architectures.

The pending goal is to extend this system to EuroHPC machines. Once fully implemented, it will ensure a fully integrated CI/CD/CB pipeline for the MTW pilot.

Continuous Benchmarking (CB) and Post-Processing

Continuous benchmarking (CB) is an integral part of the pipeline, which monitors the software's performance across various hardware architectures. Once the CI tests pass, performance benchmarks are automatically run to evaluate execution times and memory usage on local single node resources hosted at FAU. Although energy tracking has not yet been integrated, plans for incorporating such metrics are being considered for future enhancements.

Currently, the CB infrastructure is fully functional on FAU's local resources. The goal is to extend this system to EuroHPC clusters to test the MTW pilot's performance at production scales. However, this effort is ongoing as installation scripts for EuroHPC systems are still under development.

In addition to benchmarking, post-processing and visualization play a critical role in the feedback loop for continuous optimization. The MTW pilot leverages Grafana, an open-source analytics and monitoring platform, to visualize the collected performance data. Grafana's customizable dashboards allow the development team to track key performance metrics, identify bottlenecks, and make informed decisions about optimization. This visualization tool enhances the transparency of the benchmarking process and ensures that performance trends are easily accessible to developers.

Further details on waLBerla's continuous benchmarking infrastructure can be found in the work of Alt et al. [28] .

Pending Developments

The integration concept of the CI/CD/CB pipeline into EuroHPC systems is depicted on the right side of Figure 22. It is in progress, with the main challenge being the adaptation of the infrastructure to these new environments. Once complete, this will allow thorough testing and benchmarking of the MTW pilot on larger-scale resources. The delay in EuroHPC deployment is due to the late onboarding of FAU into the project.

FAU's current CD/CB can only be considered as semi-automated, as the execution requires authorized and manual user access to the target machine, to launch a pilot execution or benchmark. The planned integration of the MTW pilot into one of HiDALGO2's WFO tools (chapter **Error! Bookmark not defined.**), should allow for a user-friendly execution of benchmarks and the pilot itself, though.

The team has not adopted containerization at this stage. While containerization can improve deployment consistency, it also introduces complexities, particularly in high-performance computing (HPC) environments where optimal performance often requires fine-tuned configurations directly on the hardware. Containers may limit access to low-level hardware optimizations, which are crucial for maximizing the performance of the MTW pilot on specialized HPC systems. For this reason, and because it was not initially part of the design, the decision has been made to prioritize script-based deployment using FAU's RDM over containerized solutions.

Furthermore, to improve the consistency of HiDALGO's computing performance results, it is necessary to streamline representation and coordination of benchmarks among the five pilot applications. To realize this task, ReFrame [29] will be used as a post-processing tool within the MTW's CD/CB pipeline.

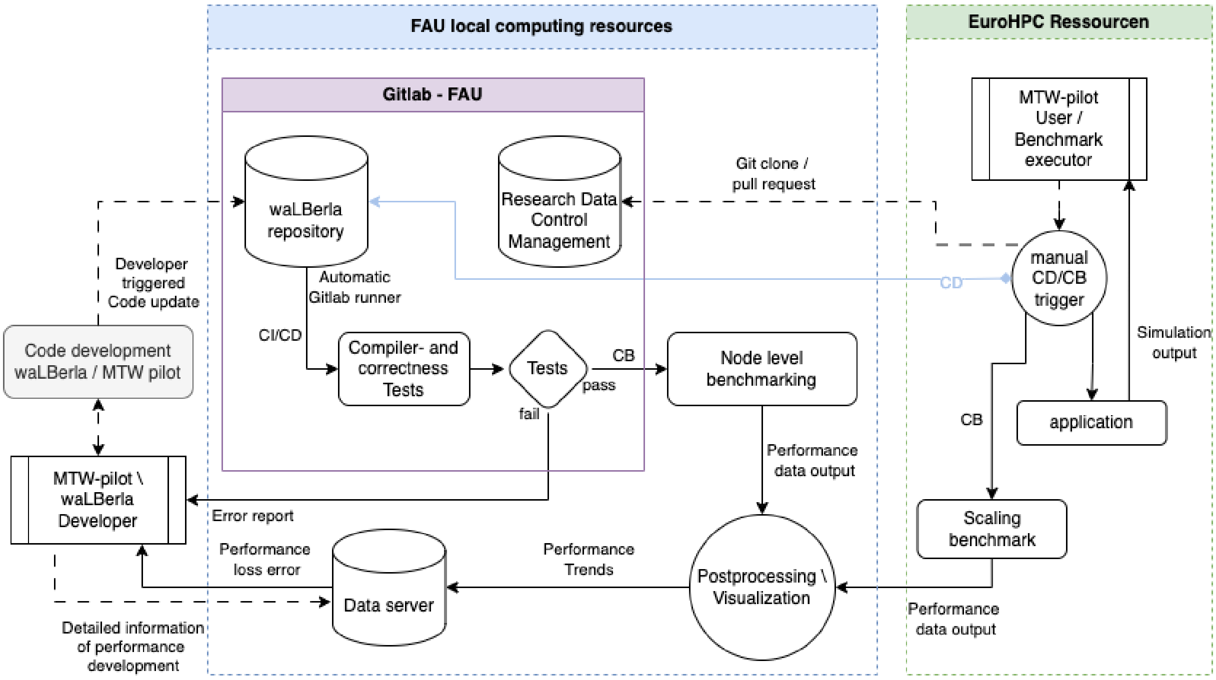


Figure 22. walBerla's CI/CD/CB local pipeline, and prototype to realize convenient deployment and benchmarking for the EuroHPC clusters[2]

[2] Continuous arrows represent automated processes, while dashed ones require user interaction. The curved rectangles represent compute intensive applications, while circles represent script execution. The cylinders represent servers and the framed rectangles a user.

In summary, the current CI/CD pipeline is well-established on local FAU systems, but significant work remains in extending it to EuroHPC clusters, particularly in integrating the MTW pilot into this broader infrastructure. The collection of scripts for deployment and benchmarking with the RDM data, serves as a foundation for future CD services, with ongoing efforts to ensure the software's scalability, reliability, and performance across all supported environments.

5.3.7.2 Development Roadmap

- Extended deployment and benchmarking coverage for EuroHPC resources (First established version) [M37]
- Decision whether to use QSC or MathSO [M30]
- Integration of MTW pilot into WFO tool [M46]
- Integration of ReFrame as post-processing stage within MTW's CD/CB pipeline [M26]

A conceptual representation of what FAU's CI/CD/CB pipeline will look like at the end of HiDALGO2 is depicted in Figure 23.

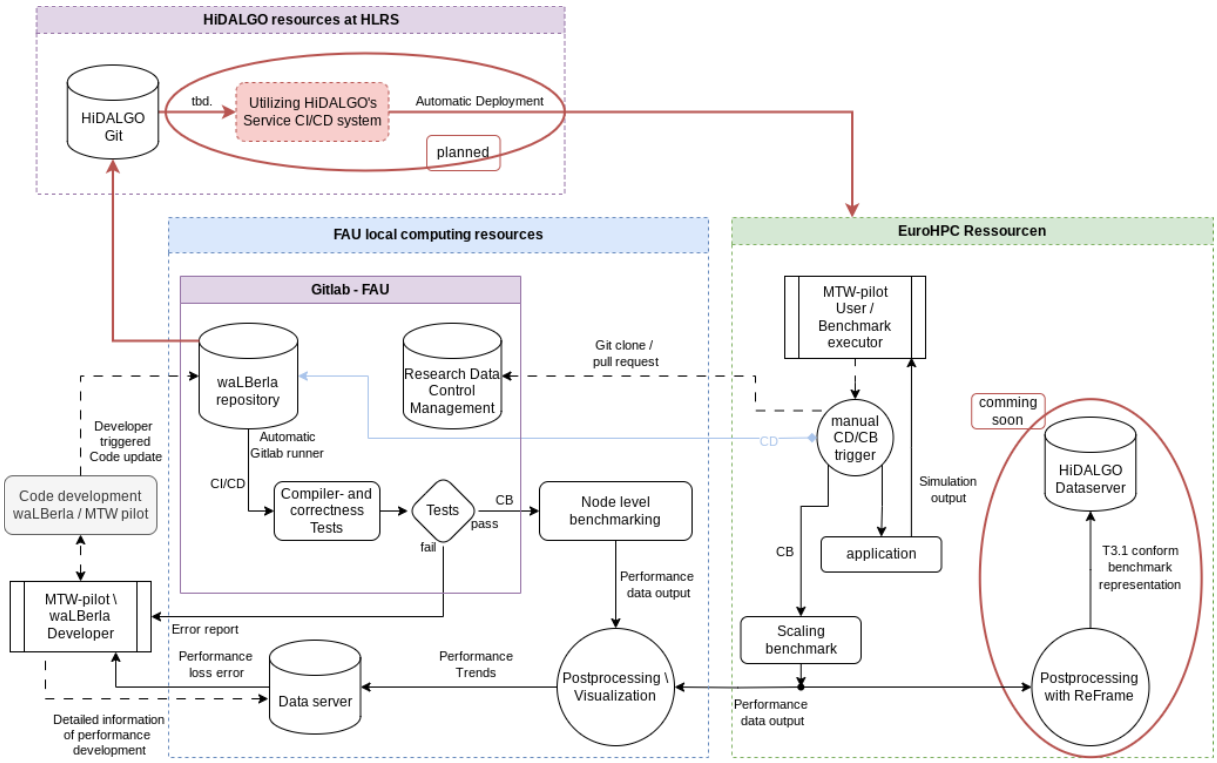


Figure 23. Planned extension of waLBerla's CI/CD/CB pipeline from Figure 22, after full realization of MTW pilot's roadmap

5.4 EuroHPC CI/CD

This section deals with the key topic of HiDALGO2's public facing CI/CD system.

5.4.1 Strategy

HiDALGO2, in order to be as adaptable as possible, decided to build upon a provided public facing CI/CD system through multiple, modular phases and components. The complete system, in fact, is envisioned as a bridge between HiDALGO2's internal Pilot CI/CD systems, to the solution agreed upon and decided through CASTIEL2 for the entire EuroHPC network, and all the participating institutions.

A key part of this strategy takes care of the fact, that discussions through CASTIEL2 are currently on going. At the time of the previous iteration of this deliverable, (Deliverable D2.4), it appeared likely that the consortium would agree to use a public facing GitLab instance, where all CoE codes would be collected, and deployed to from there to EuroHPC systems. Further details and diagrams for this system are available in Deliverable D2.4, Section 5.3.2.

It is important to note, that the Pilots CI/CD systems of HiDALGO2 are adaptable enough to interface with alternative CASTIEL2 solutions. For example, one of the solutions being considered is the EESSI-CernVM fs system [30] . If so, the HiDALGO2 Pilots CI/CD system would provide EasyBuild recipes, or it's compiled and built container images, and push them to the EESSI repositories, from which they would be streamed over and deployed to EuroHPC machines.

Thus, HiDALGO2, with its modular and flexible approach, is ready to work and interface with any solution ultimately decided as the best across EuroHPC systems through the CASTIEL2 initiative.

5.4.2 Requirements

Please refer to Deliverable D2.4, Section 5.4.1 to review HiDALOG2 specific gathered requirements for a public-facing CI/CD system.

5.4.3 Prototype GitLab Implementation

Please refer to Deliverable D2.4, Section 5.3.3, for a proposed prototype EuroHPC CI/CD system implementation, where the chosen solution is a centrally deployed GitLab instance for collecting CoE codes.

5.4.4 Development Roadmap

Please refer to the revised version of Deliverable D2.4, Section 5.3.4 for an updated roadmap, with regards to the GitLab solution for a EuroHPC CI/CD system. In case, CASTIEIL2 and the EuroHPC partners decide for a different solution, HiDALGO2 will adapt its development roadmap, and provide a report on its progress in the next iteration of this deliverable, D2.6.

6. Conclusions

This document addresses the challenge of building a solid foundation for the HiDALOG2 project infrastructure by creating a holistic platform encompassing the core services of the HiDALOG2 ecosystem and preparing the ground for further development and refinement of the pilot project use cases.

Since D2.4, the computing infrastructure has been strengthened by allocating resources for the Dashboard, and the Energy Monitoring and Optimization framework, and the infrastructure of HPDA has been extended. Moreover, EuroHPC resources for all pilots have been renewed.

The current status of the development of the features of the HiDALGO2 WFOs, namely MathSO and QCG, has been reported. Some new features have been incorporated since D2.4, incorporating changes in the architecture of the orchestrators. In general, the development is progressing as planned with few, minor deviations from the plan, which have been updated.

A new platform for job energy consumption monitoring and optimization, based on the Eviden SEMS framework, has been introduced: motivated, upon identified requirements, and functionally described. Proposed new AI-powered features and extensions to the platform, namely i) a power and performance estimator, ii) a job energy comparator, iii) a job energy predictor, and iv) a job energy recommender have been reported together with a development plan.

HiDALGO2 is continuously tackling the key challenge and goal set by the EuroHPC JU through this deliverable by developing a clear path to ease the deployment of its pilots' simulations onto EuroHPC infrastructure. All partners collaborate closely to identify major requirements, strategies, technologies and workflows to set up efficient and reliable CI/CD pipelines, resulting in three tiers of component integration, building up from CI/CD for HiDALGO2 services, testing out pilot code deployment on internal and external HPC infrastructure, and leading up to a seamless connection between the HIDALGO2 CoE and EuroHPC's common integration and deployment platform, for automatic deployment to the broader network of EuroHPC JU supercomputers. Updates on these three tiers have been reported, focusing on the current state of development and the roadmap for the next period until the end of the project.

HiDALGO2 will continue working on updating its computing and service infrastructure and the Dashboard component integration and CI/CD delivery (for services and pilots' application) goals by following the roadmap updated through this deliverable, and report on this progress in the last deliverable of the series, D2.6 (M35).

References

[1] D2.4, Infrastructure Provisioning, Workflow Orchestration and Component Integration (M11). HiDALGO2 report, 2024. DOI 10.13140/RG.2.2.29935.61605

[2] D4.6, Visualisations for Global Challenges (M17). HiDALGO2 report, 2024. DOI 10.13140/RG.2.2.30758.95045/1

[3] D2.1, Requirements Analysis and Scenario Definition (M6). HiDALGO2 report, 2023.

[4] D2.2, Requirements Analysis and Scenario Definition (M18). HiDALGO2 report, 2024.

[5] D2.7, HiDALGO Dashboard and Services (M14). HiDALGO2 report, 2024. DOI 10.13140/RG.2.2.14618.66248

[6] D3.1, Scalability, Optimization and Co-Design Activities (M12), HiDALGO2 report, 2023. DOI 10.13140/RG.2.2.16877.15849

[7] D4.1, Data Management and Coupling Technologies (M11), HiDALGO2 report, 2023. DOI 10.13140/RG.2.2.22745.90728

[8] D4.6, Visualisations for Global Challenges (M17). HiDALGO2 report, 2024. DOI 10.13140/RG.2.2.30758.95045/1

[9] D5.6, Implementation Report on Pilots' Applications (M15). HiDALGO2 report, 2024. DOI 10.13140/RG.2.2.12011.55849

[10] HiDALGO2 Keycloak , https://idm.hidalgo2.eu, [retrieved 12.31.2024]

[11] HiDALGO2 Zabbix, https://monitor.hidalgo2.eu, [retrieved 12.31.2024]

[12] HiDALGO2 Wordpress, https://hidalgo2.eu, [retrieved 12.31.2024]

[13] SEAVEA Toolkit, https://www.seavea-project.org/latest-release/, [retrieved 12.31.2024]

[14] TOP500, https://top500.org/, [retrieved 12.31.2024]

[15] GREEN500, https://top500.org/lists/green500/, [retrieved 12.31.2024]

[16] Le Sueur, Etienne, and Gernot Heiser. "Dynamic voltage and frequency scaling: The laws of diminishing returns." Proceedings of the 2010 international conference on Power aware computing and systems. 2010. (2023), https://dl.acm.org/doi/10.5555/1924920.1924921

[17] Running Average Power Limit. **Error! Hyperlink reference not valid.** [retrieved 12.31.2024]

[18] NVIDIA Management Library. https://developer.nvidia.com/management-library-nvml. [retrieved 12.31.2024]

[19] LIKWID. https://github.com/RRZE-HPC/likwid/wiki/likwid-powermeter. [retrieved 12.31.2024]

[20] Eviden EO. https://prace-ri.eu/wp-content/uploads/PRACE-at-SC17-Ludovic-Sauge.pdf [retrieved 12.31.2024]

[21] Forgejo homepage, https://forgejo.org/, [retrieved 01.12.2024].

[22] Woodpecker CI homepage, https://woodpecker-ci.org/, [retrieved 01.12.2024]

[23] Semaphore UI homepage, https://semaphoreui.com/, [retrieved 01.12.2024]

[24] Redhat AWX homepage, https://www.ansible.com/awx/, [retrieved 01.12.2024]

[25] Ansible Molecule Framework, https://github.com/ansible/molecule, [retrieved 12.31.2024].

[26] EPICURE project, https://epicure-hpc.eu/, [retrieved 12.31.2024]

[27] D2.1 – CI and performance tracking infrastructure. CEEC project report. 2023

[28] Alt, C., Lanser, M., Plewinski, J., Janki, A., Klawonn, A., Köstler, H., … Rüde, U. (2024). A continuous benchmarking infrastructure for high-performance computing applications. International Journal of Parallel, Emergent and Distributed Systems, 39(4), 501–523. DOI 10.1080/17445760.2024.2360190

[29] Vasileios Karakasis et al. “Enabling Continuous Testing of HPC Systems Using ReFrame”. In: Tools and Techniques for High Performance Computing. HUST – Annual Workshop on HPC User Support Tools (Denver, Colorado, USA, Nov. 17–18, 2019). Ed. by Guido Juckeland and Sunita Chandrasekaran. Vol. 1190. Communications in Computer and Information Science. Cham, Switzerland: Springer International Publishing, Mar. 2020, pp. 49–68. ISBN: 978-3-030-44728-1. DOI 10.1007/978-3-030-44728-1_3.

[30] EESSI-CernVM fs system, https://www.eessi.io/, [retrieved 01.12.2024]

Annexes

Annex 1: Services Matrix

[INSTITUTE]	[NAME]	[DEV/PROD]	[CPU]	[RAM]	[DISK]	[OS]	[IP]	[DOMAIN]	[Status]	Integration with KeyCloak	Integration with Zabbix	[ADMINISTRATOR]	[PURPOSE]
PSNC	HiDALGO2 - CKAN	PROD	8	16GB	40GB +2TB	Ubuntu 22.04	62.3.171.19	ckan.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	Data platform
---													

PSNC	HiDALGO2 - Streaming	PROD	4	8GB	40GB + 2TB	Ubuntu 22.04	62.3.171.226	stream.hidalgo2.eu	Running	In Progress	Running	Piotr Dzierżak	Streaming services, Elastic Search (KAFKA as alternative)
HLRS	HiDALGO2 - Streaming	DEV							Dev - Ready	Dev - Ready	Dev - Ready		
HLRS	HiDALGO2 - Wiki	PROD	2	4GB	20GB + 20 G	Ubuntu 22.04	141.58.0.233	wiki.hidalgo2.eu	Running	Running	Running	Sameer Haroon	Wiki (Wiki-js)
PSNC	HiDALGO2 - Wiki	DEV							Dev - TODO	Dev - TODO	Dev - TODO		

PSNC	HiDALGO2 - IDM	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.171.16	idm.hidalgo2.eu	Running	Not required	Running	Piotr Dzierżak	IDM / Keycloak
HLRS	HiDALGO2 - IDM	DEV	2	4GB	20+20 GB	Ubuntu 22.04	141.58.0.233		Running	Not required	Dev - Ready		
PSNC	HIDALGO2 - Askbot	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.171.180	askbot.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	Question and answer oriented Internet forums
HLRS	HIDALGO2 - Askbot	DEV					141.58.0.233		Dev - Ready	Dev - Ready	Dev - Ready		
PSNC	HiDALGO2 - Monitor	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.170.54	monitor.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	Zabbix monitoring, integration with Keycloak postponed
HLRS	HiDALGO2 - Monitor	DEV					141.58.0.233		Dev - Ready	Dev - Ready	Dev - Ready		
PSNC	HiDALGO2 - Prototype0	PROD	8	16GB	40GB +2TB	Ubuntu 22.04	62.3.170.126	prototype.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	Training access node, integration with Keycloak postponed
---													
PSNC	HiDALGO2 - Prototype1	PROD	32	32GB	40GB	Ubuntu 22.04			Running	Running	Running	Piotr Dzierżak	Training node, integration with Keycloak postponed
---													
PSNC	HiDALGO2 - Prototype2	PROD	32	32GB	40GB	Ubuntu 22.04			Running	Running	Running	Piotr Dzierżak	Training node, integration with Keycloak postponed
---													
PSNC	HiDALGO2 - Prototype3	PROD	32	32GB	40GB	Ubuntu 22.04			Running	Running	Running	Piotr Dzierżak	Training node, integration with Keycloak postponed
---													
PSNC	HiDALGO2 - Prototype4	PROD	32	32GB	40GB	Ubuntu 22.04			Running	Running	Running	Piotr Dzierżak	Training node, integration with Keycloak postponed
---													
PSNC	HiDALGO2 - Support ticket	DEV											
HLRS	HiDALGO2 - Support ticket	PROD	2	4GB	15+15GB	Ubuntu 22.04	141.58.0.233	ticket.hidalgo2.eu	Running	Running	Running	Sameer Haroon	Zammad, SAML protocol
PSNC	HiDALGO2 - Moodle	DEV	4	8GB	40GB	Ubuntu 18.04	62.3.171.60	sophora-60.man.poznan.pl	Running	Running	Running	Piotr Dzierżak	Learning platform
HLRS	HiDALGO2 - Moodle	PROD	2	4GB			141.58.0.233	moodle.hidalgo2.eu	Running	Running	Running	Sameer, Maksym	Moodle, Learning platform
PSNC	HiDALGO2 - JupyterHub	PROD	6	12GB	40GB + 2TB	Ubuntu 22.04	62.3.171.173	jupyter.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	JupyterHub access node
PSNC	HiDALGO2 - JupyterLab compute-1	PROD	16	32GB	40GB	Ubuntu 22.04			Running	Not required	Running	Piotr Dzierżak	JupyterLab compute node, Keras, Tensorflow
PSNC	HiDALGO2 - JupyterLab compute-2	PROD	16	32GB	40GB	Ubuntu 22.04			Running	Not required	Running	Piotr Dzierżak	JupyterHub compute node, Keras, Tensorflow
---													
PSNC	Website	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.170.227	www.hidalgo2.eu	Running	Not required	Running	John Kitsos	Website
PSNC	Portal MathSO	PROD	8	16GB	40GB	Ubuntu 22.04	62.3.170.136	prunus-136.man.poznan.pl portal.hidalgo2.eu	Running	Running	Running	Akos Kovacs	Portal (MathSO)
PSNC	Portal QCG	PROD	30	30GB	8GB	OKD 3.x/OKD 4.x		qcg.hidalgo2.eu	In Progress	In Progress	In Progress	Bartosz Bosak	Portal (QCG) - available at the end of Nov
HLRS	Portal	DEV							Dev - Ready	Dev - Ready	Dev - Ready		
PSNC	Portal - dashboard	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.170.210	prunus-210.man.poznan.pl				Piotr Dzierżak	
HLRS	HiDALGO2 - Open Project	PROD	2	4	20 + 20 GB	Ubuntu 22.04	141.58.0.83	project.hidalgo2.eu	Running	Not possible (paid license r	Running	Sameer Haroon	Project Management Software (Open Project)
PSNC	HiDALGO2 - Open Project	DEV							Not required	Not required	Not required		

Figure 24. Services Matrix

Annex 2: Roadmap for Workflow Orchestration

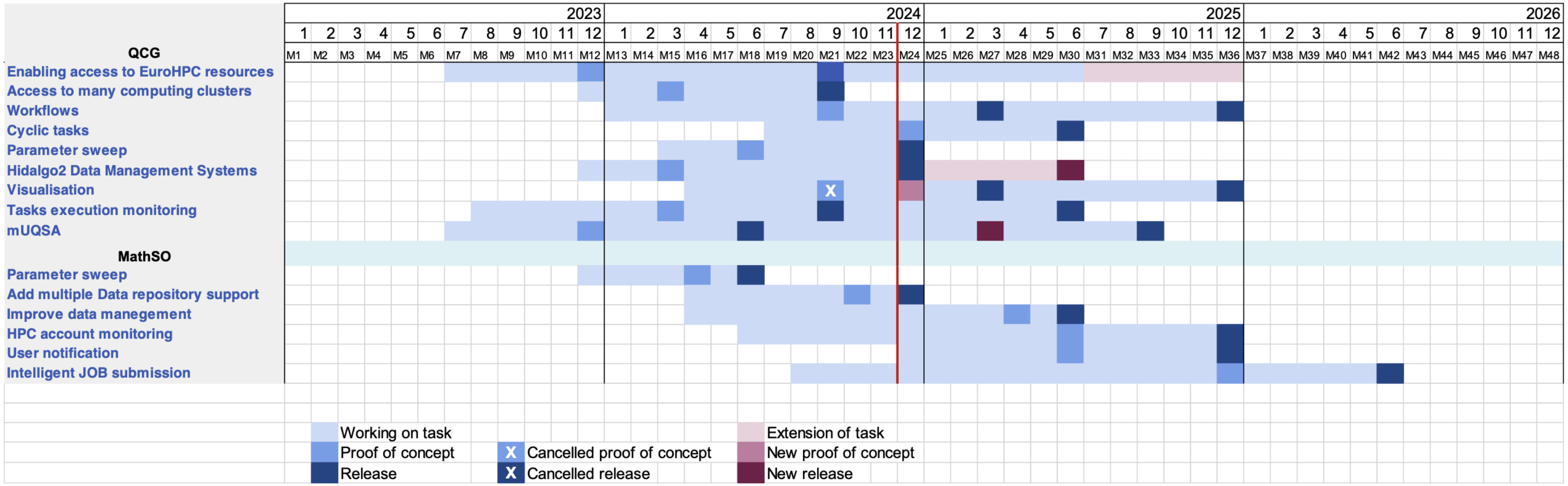


Figure 25. Gantt chart to show the development roadmap for the Workflow Orchestrators