D2.6 Infrastructure Provisioning, Workflow Orchestration and Component Integration

Date: November 29, 2025

Document Identification			
Status	Review	Due Date	30/11/2025
Version	1.0	Submission Date	29/11/2025

Related WP	WP3, WP4, WP5	Document Reference	D2.6
Related Deliverable(s)	D2.1, D2.2, D2.4, D2.5, D2.7, D2.8, D3.2, D4.2	Dissemination Level (*)	PU
Lead Participant	USTUTT	Lead Author	Sameer Haroon (USTUTT)
Contributors	PSNC, SZE, ATOS, UNISTRA, MTG, FAU	Reviewers	Jesús Gorroñogoitia (ATOS)
			Harald Köstler (FAU)

Keywords:
operational environment, infrastructure provisioning, component integration, deployment, hybrid workflows, orchestration, development roadmap, energy management

Document Information

List of Contributors	
Name	Partner
Sameer Haroon	USTUTT
Michael Zikeli	FAU
Christophe Prud'homme	UNISTRA
Bartosz Bosak	PSNC
Tomasz Piontek	PSNC
Piotr Kopta	PSNC
Michał Kulczewski	PSNC
Wojciech Szeliga	PSNC
Youssef El Faqir El Rhazoui	ATOS
Angela Rivera	MTG
Luis Torres	MTG
László Környei	SZE
Aron Nemeth	SZE

Document History			
Version	Date	Change editors	Changes
0.1	08/10/2025	Sameer Haroon (USTUTT)	First draft with TOC and points
0.2	16/10/2025	Marcin Lawenda (PSNC), Rahil Doshi (FAU)	ToC, timeline and responsibilities approved
0.21	27/10/2025	Christophe Prud'homme (UNISTRA)	Section 5.3.2: UB CI/CD and Ktirio GUI.
0.22	28/10/2025	Michael Zikeli (FAU)	Section 5.3.5: MTW CI/CD
0.3	29/10/2025	Sameer Haroon (USTUTT)	Section 1, Section 2.2
0.31	30/10/2025	Bartosz Bosak (PSNC)	Section 3.2: QCG Portal
0.32	02/11/2025	Youssef Rhazoui (ATOS)	Section 4: Energy Management
0.33	05/11/2025	Michael Zikeli (FAU)	5.3.5: MTW CI/CD (update)

0.40	08/11/2025	Sameer Haroon (USTUTT)	Executive Summary, Metadata
0.5	10/11/2025	Sameer Haroon (USTUTT)	Section 2.2, 2.3, 2.4
0.51	11/11/2025	Sameer Haroon (USTUTT)	Section 5.1
0.6	13/11/2025	Angela Rivera (MTG) Luis Torress (MTG)	Section 5.5
0.61	13/11/2025	Michał Kulczewski (PSNC)	Section 5.4
0.7	18/11/2025	Michael Zikeli (FAU)	Section 5.6 update
0.71	19/11/2025	László Környei (SZE)	Section 3.2, 5.2
0.8	20/11/2025	Sameer Haroon (USTUTT)	Version ready for internal review
0.81	22/11/2025	Sameer Haroon (USTUTT)	Feedback from reviewers, shortening content for Section2, moving to Annex.
0.82	24/11/2025	Youssef Rhazoui (ATOS)	Update after feedback, moving to Annex
0.83	25/11/2025	Bartosz Bosak (PSNC)	Update after feedback, moving to Annex
0.84	26/11/2025	László Környei (SZE)	Update after feedback, moving to Annex
0.9	27/11/2025	Sameer Haroon (USTUTT)	Version ready for quality check
0.95	28/11/2025	Rahil Doshi	Quality assurance check
1.0	29/11/2025	Marcin Lawenda	Final check and improvements

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable Leader	Sameer Haroon (USTUTT)	27/11/2025
Quality Manager	Rahil Doshi (FAU)	28/11/2025
Project Coordinator	Marcin Lawenda (PSNC)	29/11/2025

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
EC	European Commission
Dx.y	Deliverable number y belonging to WP x
WP	Work Package
UAP	Urban Air Pollution use case
UB	Urban Buildings use case
RES	Renewable Energy Sources use case
WF	Wildfires use case
MTW	Material Transport in Water use case
AAI	Authentication and Authorization Infrastructure; federated identity/authorization across services.
ABAC	Attribute-Based Access Control; permissions decided by user/group/attribute claims.
API	Application Programming Interface; programmatic access to services.
BaaS	Benchmarking as a Service; automated benchmarks with published results.
CKAN	Comprehensive Knowledge Archive Network; open-data portal for datasets/results.
CLI	Command-Line Interface; text-based tools to prepare/launch tasks.
CPUh/GPUh	CPU-hours / GPU-hours; accounting units for compute consumption.
CVE	Common Vulnerabilities and Exposures; public identifiers for security issues.
CycloneDX	OWASP SBOM specification describing software components/dependencies.
EuroHPC	European high-performance computing initiative operating flagship systems.
FMU	Functional Mock-up Unit; packaged co-simulation component (FMI).
GHCR	GitHub Container Registry; OCI image and artifact registry.
GHPages	GitHub Pages; static website hosting for benchmark/report sites.
GUI	Graphical User Interface; here the Ktirio web front-end.

HPC	High-Performance Computing; large-scale cluster/supercomputer workloads.
IAM	Identity and Access Management
KPI	Key Performance Indicator; metric to track progress/success.
kub-cd	GitHub repository orchestrating deployments and HPC job submissions.
MathSO	Orchestration service used to submit jobs from the platform to HPC systems.
OCI	Open Container Initiative; standards for containers and registries.
OIDC	OpenID Connect; identity protocol for tokens/SSO over OAuth2.
PR	Pull Request; code review unit that also drives preview deployments (/pr-XYZ).
purl	Package URL; canonical identifier scheme for software packages.
QCG	QosCosGrid; HPC workflow/scheduling tools — noted earlier but not used here.
REST	Representational State Transfer; HTTP-based API style for orchestration triggers.
SBOM	Software Bill of Materials; machine-readable inventory of build components.
SIF	Singularity Image Format; Apptainer's HPC container image format.
SLO	Service Level Objective; target level for reliability/performance metrics.
SLURM	Simple Linux Utility for Resource Management; common HPC job scheduler.
SLSA	Supply-chain Levels for Software Artifacts; provenance/security framework.
SPDX	Software Package Data Exchange; SBOM/metadata standard from the Linux Foundation.
SSO	Single Sign-On; one login granting access across multiple services.
SWID	Software Identification Tags; ISO standard for software tagging.
VCS	Version Control System; systems like Git tracking code history.
WASM	WebAssembly; portable binary format for running code in the browser.
YAML	"YAML Ain't Markup Language"; human-readable config/CI format.
GPU	Graphical Processing Unit
DAG	Directed Acyclic Graph

Executive Summary

Deliverable D2.6 is the final report in a series of deliverables (D2.4, D2.5, D2.6) on HiDALGO2's "Infrastructure Provisioning, Workflow Orchestration and Component Integration". It describes the technical components and the infrastructure that make up HiDALGO2's operational environment, providing a concise summary of the results achieved in the previous deliverables of the series, with focus kept on updates since the last reporting period, and on looking towards maintaining and improving the technical environment going into the final year of the HiDALGO2 project.

HiDALGO2's simulation and product development depend on different forms of compute infrastructure. Firstly, for faster development and prototyping, HiDALGO2 developers have access to in-house HPC, AI and HPDA resources. Secondly, HiDALGO2 has secured access to all available EuroHPC machines, for large scale runs of our simulation codes, and in the future, for scaling up AI and HPDA runs. HiDALGO2's technical environment also consists of a considerable cloud resource allocation, to provision virtual machines to host the supporting and developmental services that HiDALGO2 development needs, as well as providing services for future external users and stakeholders. These services, and the resources behind them, as well as those around compute and HPDA, are reviewed shortly in chapter 2, while readers are referred to D2.4 and D2.5 for detailed descriptions.

Some of the main services provided through HiDALGO2, for both its developers and for external users, are the hybrid workflow orchestrators, namely, the MathSO Portal and the QCG Portal. These WFOs give the ability to deploy and run HiDALGO2 simulation codes on in-house as well as EuroHPC machines, along with features such as data orchestration and integration with visualisation tools. They are reviewed in depth in chapter 3, where an overview of the tools, as well as updates around their development is given, and detailed development roadmaps are provided.

HiDALGO2 has also taken on the development of a suite of energy management tools for HPC applications. These are to be released directly on Jupiter, an exascale EuroHPC machine, and should provide benefits not only to HiDALGO2 pilot codes, but also the wider European HPC CoE network. Details and updates for its developments are given in chapter 4.

The last chapter is dedicated to the topic of component integration. D2.6 focuses on both the CI (continuous integration) and CD (continuous deployment) of the main products of HiDALGO2, the pilot simulation codes and their library frameworks, and how these codes and libraries can be efficiently and securely deployed to EuroHPC machines, through coordination and cooperation with CASTIEL2, for use by HiDALGO2 developers and future users, as well as any interested EuroHPC users.

1 Introduction

1.1 Purpose of the document

Deliverable D2.6 is the final iteration of a series of deliverables (D2.4, D2.5, D2.6) on the topic “Infrastructure Provisioning, Workflow Orchestration, and Component Integration”. It provides updates since the last report in the series, D2.5[1] , but also, as it is the last one on this topic, it provides a concise wrap up of, or references to, the material already covered in D2.4[2] and D2.5[1] , so that the reader can treat D2.6 as an up to date report on the topics:

- Technical Infrastructure for HiDALGO2
- Services integrated into the HiDALGO2 dashboard
- EuroHPC resource access for HiDALGO2 pilots
- HiDALGO2’s Workflow Orchestrators (WFOs), MathSO Portal and QCG Portal.
- Energy Monitoring and Optimization framework from ATOS/Eviden
- Component Integration and Continuous Integration / Deployment in HiDALGO2

The deliverable also allots a subsection to each chapter to give a roadmap for the above topics, as the HiDALGO2 project moves into its final year (Year 4, 2026).

1.2 Relation to other project work

This document builds upon the work of the previous deliverables in the same series, D2.4 [2] and D2.5 [1] . It also is closely related to D2.7 [5] and D2.8 [9] , the HiDALGO2 Dashboard deliverables, as well as D2.1 [3] , and D2.2, the HiDALGO2 Requirements Gathering deliverables. Aside from this, the document refers to the Pilot Code use cases, which are detailed in D5.7 [6] , and on the Data Management technical frameworks in D4.3, and deliverable D4.4 [8] .

1.3 Structure of the document

This document is structured in **6** major chapters:

Chapter 2 presents the high-level design of HiDALGO2’s environment, along with details on the technical infrastructure supporting this environment and its components.

Chapter 3 goes into detail about HiDALGO2’s two Workflow Orchestrators (WFOs), the MathSO portal, and the QCG portal, both of which integrate into the HiDALGO2 dashboard.

Chapter 4 describes the development of the Energy Management suite of tools from ATOS and how it will benefit HiDALGO2 pilot use cases.

Chapter 5 explains HiDALGO2's strategy for CI/CD, and provides a description of how it has been implemented across services and pilot codes.

Chapter 6 rounds up the deliverable with a conclusion and a roadmap for the last upcoming year of HiDALGO2.

The document ends with relevant **references** and **annexes**. Keeping in mind the recommended length for HiDALGO2 deliverables (main content section of 50 pages), the chapters described above focus more on updates, design, and roadmaps, while details on results and infrastructure have been moved to the Annex section (numbered as **Chapter 7**) of this deliverable.

2 Infrastructure Provisioning and Services

HiDALGO2's operational environment has grown over the course of the project, and has reached a level of stability that is expected to support the project for the remainder of its lifetime.

2.1 Motivation and Requirements

A concise recall of the main motivation behind a technical infrastructure in terms of requirements is reiterated here. For detailed overall requirements, please see related deliverables D2.4[2] , D2.1[3] and D2.2.

- **REQ-HPII-001:** Access to HPC infrastructure

  The core component of HiDALGO2, the pilot simulation codes, requires compute resources for development and scalable runs. This requirement is met through in-house compute resources and test beds, which include clusters and prototyping systems, as well as through connecting to EuroHPC systems for more powerful compute requirements.

- **REQ-HPII-02:** Platform Scalability

  HiDALGO2 requires substantial development services, such as code repositories and data storage systems, as well as supporting services, such as a reliable ticketing and knowledge sharing system. These number and resource demand of these services have increased over time, and are provisioned through in-house cloud resources.

- **REQ-HPII-003:** Platform Robustness

  Redundancy in cloud resources is introduced through having development as well as production virtual machines to host and deploy services. For compute, pilot code developers have access to multiple EuroHPC machines, as well as multiple options for local compute and development resources.

The big picture view and objective shaping HiDALGO2's technical environment is about helping HiDALGO2's developers succeed, through easy access to technical resources and supporting services, in their goal of developing their pilot use cases, and to help external users in accessing and using them.

2.2 Design and Architecture

The overall design of HiDALGO2's environment has been refined and extended over the course of the project. Cloud infrastructure from the partners USTUTT and PSNC make the platform on which most of the HiDALGO2 web services are hosted and maintained. A list of the services themselves, and more details about the infrastructure behind them is given in section 2.3. Aside from the cloud infrastructure, HiDALGO2 also requires compute infrastructure for various different use cases. This is met on different scales, from in-house small compute clusters hosting Jupyter notebooks[19] for experimentation, to securing access to EuroHPC supercomputers with extreme scale access. Further details are also given in section 2.3.

Figure 1 shows how individual components of the environment come under the HiDALGO2 system, and how they interact with internal and external users and developers, as well as their connections to external systems and services. The overall design and architecture envision a flow through the environment, where users land at the HiDALGO2 dashboard[1], which is the landing page from where all other services can be reached in an integrated fashion, through a central Identity Management service which verifies the specific user's access rights. More information about the HiDALGO2 dashboard is covered in D2.8, and will be further expanded upon in the upcoming deliverable D2.9.

The user can access one or more of HiDALGO2's supporting services. For example, if the user wants to learn more about a pilot use case or a related topic or training, the HiDALGO2 dashboard will lead them to the Moodle instance. If they instead want to ask a general question, or raise an issue, they can go the Askbot service, or the Zammad ticketing service. The general HiDALGO2 website is also accessible from the Dashboard, where the user can learn more about the project in general, or view the latest news and events.

If the user is already interested in a specific use case, they can click on one of them e.g. the icon displaying the use case of Urban Air Pollution. That would take the user to the MathSO portal, another component of the HiDALGO2 system, where one can configure the simulation parameters, deploy and run it, and get back results. In this way, a complete use case run workflow is envisioned from the entry of the user to HiDALGO2's operational environment.

[1] https://dashboard.hidalgo2.eu

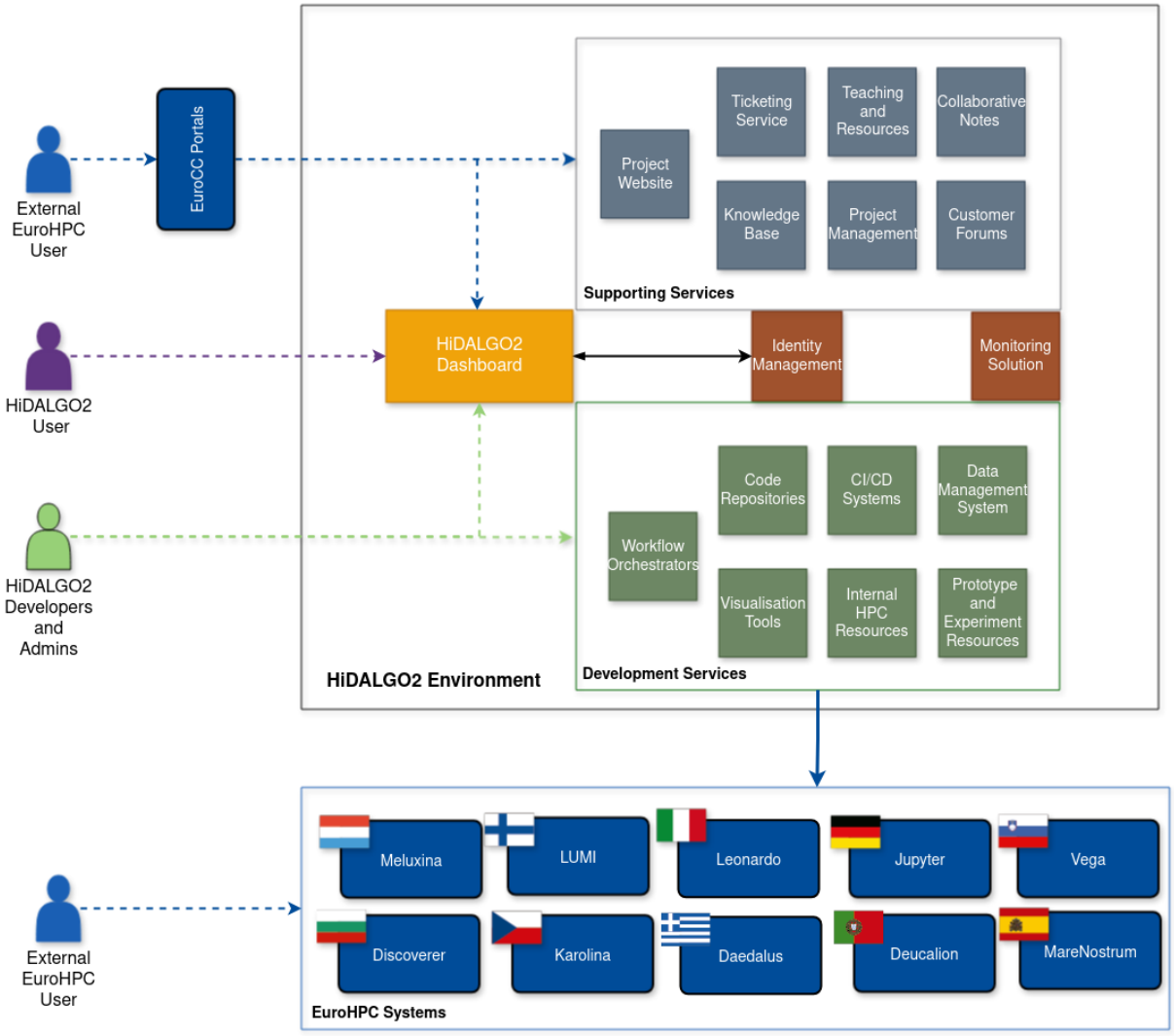


Figure 1: HiDALGO2 Environment Design

2.3 Implementation

As shown above in Figure 1, HiDALGO2's environment is split between Supporting and Development services. In Annex I: Infrastructure and Services of this document, Table 5 and

Table 6 give a simplified and updated list of the individual applications making up these two types of services and, if applicable, their web domains. We also provide a more comprehensive overview of the technical infrastructure behind these services in the same Annex.

This year, HiDALGO2 moved its principal code management repository from Bitbucket to GitLab (Table 1). GitLab is a major open-source code management software, with

multiple features that better fit to HiDALGO2.GitLab provides more tools for statistics, and integrates well with both internal Gitlab repositories of different HiDALGO2 pilot codes, as well as external Gitlab instances such as CASTEIL2's. This necessitated a migration of all internal code repositories from the older Bitbucket instance to the new GitLab instance, and was successfully carried out over. This also included updating the mirroring of pilot code repositories to CASTIEL2's GitLab.

Table 1: GitLab infrastructure specification

Name	Code Repository		Prod	PSNC	IP	N/A	
Software	GitLab		Dev	N/A	IP	N/A	
Hardware	CPU cores	8	RAM	16 GB	Hard Disk	60 GB	
Domain	https://code.hidalgo2.eu						
Description	GitLab is a major open-source code management software, and replaces the previous code management						

A new service, called the Ktiro GUI application, is now provided through HiDALGO2 as well. This is a simulation configuration GUI, developed by UNISTRA, which prepares an Urban Building simulation run for a user. Details to this service are provided in Table 2, and elaborated upon in section 5.3: CI/CD for Urban Buildings.

Table 2: Ktirio GUI infrastructure specification

Name	Ktirio GUI		Prod	USTUTT	IP	141.58.0.233	
Software	Custom developed		Dev	UNISTRA	IP	N/A	
Hardware	CPU cores	4	RAM	8 GB	Hard Disk	60 GB	
Domain	https://ktirio.hidalgo2.eu						
Description	Ktirio GUI is simulation configuration GUI, developed by UNISTRA, which prepares an Urban Building simulation run. Users can access this through the Urban Buildings icon on the HiDALGO2 dashboard.						

In addition, another important service added was a collaborative note taking and storing application, as shown in Table 3. This software, called HedgeDoc[2], provides a way

[2] https://hedgedoc.org/

for HiDALGO2 members to easily set up collaborative meeting notes and ideas for group discussions very quickly, and stores them for later recall, as well as for exporting directly in markdown or to other formats.

Table 3: HedgeDoc infrastructure specifications

Name	HedgeDoc		Prod	PSNC	IP	N/A
Software	HedgeDoc		Dev	USTUTT	IP	N/A
Hardware	CPU cores	2	RAM	4 GB	Hard Disk	40 GB
Domain	https://hdoc.hidalgo2.eu					
Description	HedgeDoc is an open-source, web-based, self-hosted collaborative markdown editor, for easy collaboration on notes, graphs and presentations in real-time.					

If the reader is interested in specifications for the rest of the HiDALGO2 services, similar specification tables are provided for all developmental and supporting services in previous versions of this deliverable, in section 2 of D2.4 and D2.5. To avoid redundancy, they are not copied over into D2.6. As mentioned before, a concise, matrix overview is provided instead in Annex I: Infrastructure and Services of this document.

In addition to compute resources within HiDALGO2's environment, Table 7, also moved to Annex I: Infrastructure and Services, gives an overview of access to EuroHPC resources. It is expected that access awards run out and are renewed at different points over the final year of HiDALGO2's project. In order to reach the KPIs that the project targets, keeping the access to EuroHPC resources is critical for the success of HiDALGO2.

2.4 Development Roadmap

HiDALGO2's operational environment has reached a level where it has achieved most of its previous development goals. However, as the final of the year of the project is crucial for the continuous development of pilot code simulations, as well as their delivery to end users, the project is prepared to scale and grow with any further resources and services required. Some expected changes or growths that are likely to require expanding HiDALGO2's environment are:

- **In-house GPU resources**
  - Especially to help with early prototyping and experimental runs with AI surrogate modelling, it will be helpful for HiDALGO2 developers to have easier access to GPU resources. Ideally, we would like to boost our JupyterHub and Prototyping Clusters with GPU resources. Later, the pilots and developers should use EuroHPC AI calls and AI Factory calls, to get access to resources on scale.

A review of the previous roadmap points from D2.5 is as follows:

- All Web Services are to be integrated with the HiDALGO2 Keycloak IDM, as well as the Operational Portal services and the Workflow Orchestrators.
  - This has been achieved.
- 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.
  - This is still to be done. In general, the strategy for external users is to be finalised.
- 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 is done, both, in the case of changing HiDALGO2's git repository from Bitbucket to GitLab, as well as maintaining a mirror of pilot codes with CASTIEL2's GitLab repository. More details in Section 5.
- 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.
  - Additional storage nodes had been allocated. More storage requirements are expected in the final year, with the focus on meeting ensemble scenarios and uncertainty quantification KPIs.

3 Workflow Orchestration

HiDALGO2 offers two Workflow Orchestrators, developed independently by two of the partners, MathSO by SZE, and QCG by PSNC. Both offer different strengths and conveniences when it comes to deploying and executing applications. QCG initially developed from the point of view of supporting the Renewable Energy Sources (RES) pilot code more closely, while MathSO developed with Urban Air Pollution (UAP) code as the priority to support. Now, both WFOs are working together with the rest of the pilot codes to support them, and the experience gained is expected to have the two WFOs ready for wider community use by the end of HiDALGO2, to support and ease the execution of many different kind of HPC pilot code applications.

3.1 MathSO Portal

The MathSO portal is a web-based front end for running and managing high-performance computing (HPC) simulations, developed around the MathSO research group at Széchenyi István University. It serves as a central hub where users can configure, launch, and monitor computational jobs, primarily large-scale CFD and urban air-pollution simulations, on various EuroHPC machines, and supercomputers, like Altair or Komondor using containerized workflows. In the HiDALGO2 project context, the MathSO portal acts as one of the main user-facing components for infrastructure provisioning and services - giving scientists and engineers a unified environment to submit experiments, connect to HPC resources, and explore results in an interactive way.

3.1.1 Requirements and Specifications

The MathSO portal aims to provide a unified web-based environment for orchestrating and deploying complex simulations and workflows on diverse HPC systems within the HiDALGO2 ecosystem. It will offer a marketplace-like interface for applications, blueprint-based software environment definitions, and robust job submission to multiple EuroHPC centres, supporting workload managers such as SLURM[32] and PBS/Pro[34] . The portal must integrate with a Data Management System using OAuth2/OIDC and REST APIs to automate input/output data flows between HiDALGO2 storage, external data providers (e.g. Copernicus), and target HPC clusters. Users should be able to select or automatically obtain an optimal HPC system per job and browse a persistent archive of completed workflows with metadata, logs, and data locations.

Beyond core job management, MathSO must support advanced workflow and container-based capabilities, including container integration (e.g. Apptainer), reusable and

parameterizable workflows, and a combined registry for workflows and container images, with appropriate resource allocation and networking support (MPI, GPU, etc.). It should enable ensemble execution, coupled multi-simulation scenarios across clusters, and in-portal post-processing through visualization and data analytics tools. Non-functional requirements include profiling and performance monitoring, scalability and reliability of containerized workflows, and adherence to security best practices. Together, these features position MathSO as the central workflow orchestration and user-facing component for deploying, managing, and analysing HiDALGO2 simulations across diverse pilots and HPC infrastructures.

The aim of MathSO portal is to support the deployment of HPC applications to HPC systems (See Figure 2 for a list). It also supports integration features to tightly integrate HiDALGO services. The following systems are currently accessible from the portal:

- **EuroHPC Systems:**
  Lumi, MeluXina, Vega, Karolina, Discoverer and Deucalion
- **Other HPC Systems:**
  Altair (PSNC Cluster), Komondor (NCC Hungary), Solyom (SZE)
- **HiDALGO2 Services:**
  HiDALGO2-Training cluster, HiDALGO2-JupyterHub

Support for the remaining EuroHPC system (MareNostrum 5 and Leonardo) is underway. Besides deployment, SFTP connection for data transfer or direct execution on the head node is also supported, so also resources on the JupyterHub can be reached. Further details about requirements are submitted in the deliverable D2.2.

3.1.2 Design and Architecture

In the last reporting period, the MathSO orchestrator has been redesigned with the following two concepts in mind. First, it must aesthetically fit the design of the HiDALGO2 dashboard. Second, it must be able to link with the dashboard for displaying appropriate applications within a pilot. In this section, we go over the new design and architecture of the new MathSO orchestrator. Below, we are focusing on the design and architecture of the main features of the orchestrator. Please, keep in mind that the length of this document does not allow for showcasing all details of the implementation.

3.1.2.1 Landing page

The landing page of the MathSO orchestrator resembles the outline of the Dashboard, detailed in D2.8 [9] . The landing page, as shown on Figure 3, consists of the following areas:

- **Top ribbon** contains links to various modules of the frontend.
- **Login/logout area** is on the top right for user login, logout and manual.

- **System status** informs the user about deployment status and summary.
- **Recent deployments** show the last five deployments, including quick access to information, deletion or post processing.
- **Pinned applications** list applications, that has been pinned by the user.
- **Pinned instances** list configured applications, called instances, that has been pinned by the user.
- **Pinned HPC systems** list configured HPC systems, including at-a-glance partition usage that has been pinned by the user.

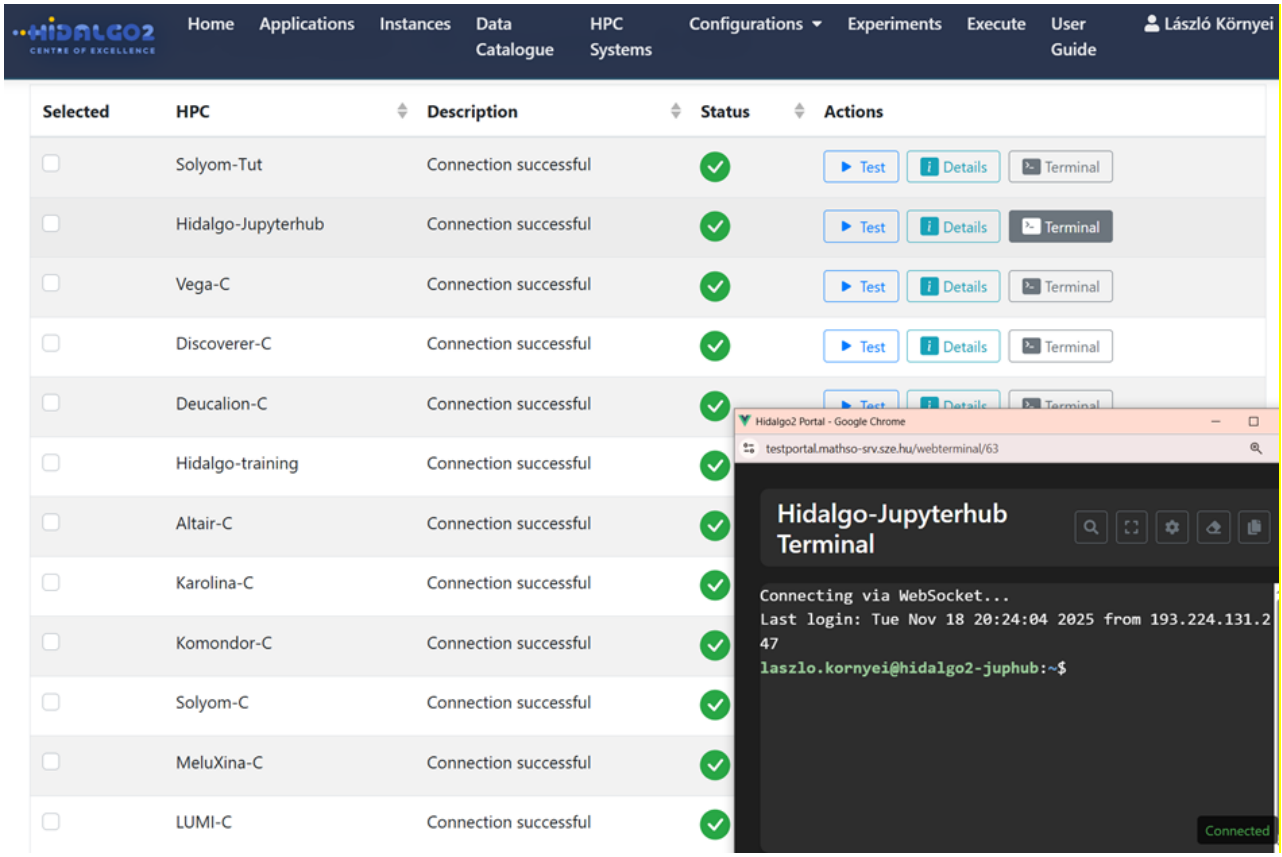


Figure 2: Connection testing for connected systems. Terminal access of connected systems is available with a single push of a button.

3.1.2.2 Applications, Instances, Data catalogues, HPC Systems

As shown in Figure 4, the layout design of applications, instances, data catalogues and HPC systems follow the same schema, which is the tile form, that also appears on the HiDALGO2 Dashboard. The page consists of two areas:

- The **Tile Area** contains tiles of the specific kind. All tiles can be configured with a picture, so that they are visually easier to find. Tiles also contain action buttons, that execute functionalities in context of the specific tile. Additionally, all tiles may contain tags for easier search.
- The **Filter Bar** contains a search box, and a tag selection field to facilitate finding the appropriate items.

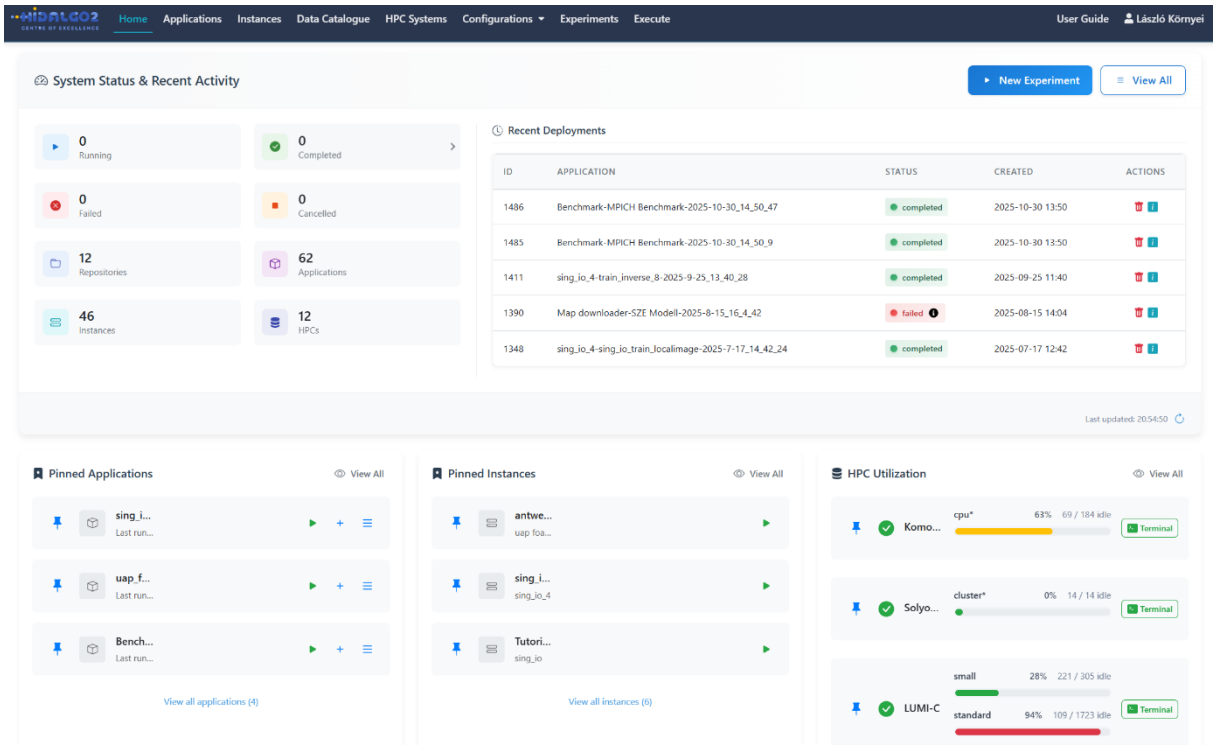


Figure 3: MathSO Workflow Orchestrator landing page layout. Designed in a way that all current information is visible at a glance.

Items in this list can have 3 states with regards to visibility:

- **Pinned items** appear at the beginning, and on the landing page.
- **Regular items** appear afterwards, and not on the landing page.
- **Archived items** are hidden, and only visible if the “Show Archived” switch is toggled.

3.1.2.3 Blueprint editor

The blueprint editor, as shown in Figure 5, is available for developer users only. It enables creating new applications and editing current ones. The basic five sections of the editor are the following:

- **Basic information**, including name, description and working directory prefix
- **Blueprint inputs**, for parameters, files and containers (containing different code versions or accelerator libraries). Parameters are to be edited by the application user and configured into instances. Hidden parameters can be used to configure the application by the developer without recompiling the container.

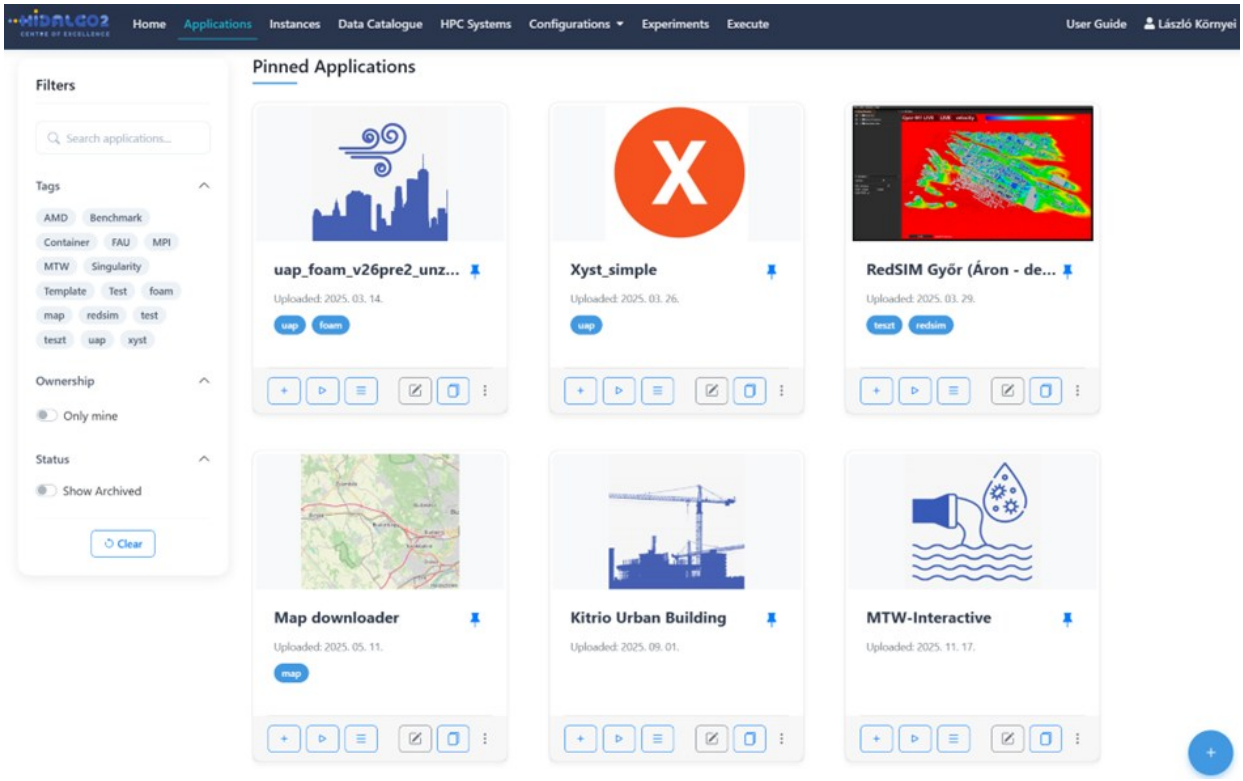


Figure 4: MathSO Portal Application Manager. The pinned applications are showcased including the official codes and applications, that have already been integrated: Xyst, RedSIM and UAP-FOAM (UAP), Ktirio Urban Building and Material Transport in Water.

- **Node templates** are the actual commands that are executed on the HPC system with e.g. SRUN or SBATCH on a SLURM[32] system, or on the head/login node. Templates are structured in a directed acyclic graph (DAG) for dependency, and are executed accordingly.
- **Expected results** will mark output types for simulation post processing. 2D results are processed to the image viewer system, while 3D results are shown in the appropriate visualization tool, like CFDR.

At the end of the editor, the developer can preview the generated YAML blueprint file, which can be downloaded, copied, and eventually reuploaded after offline edit.

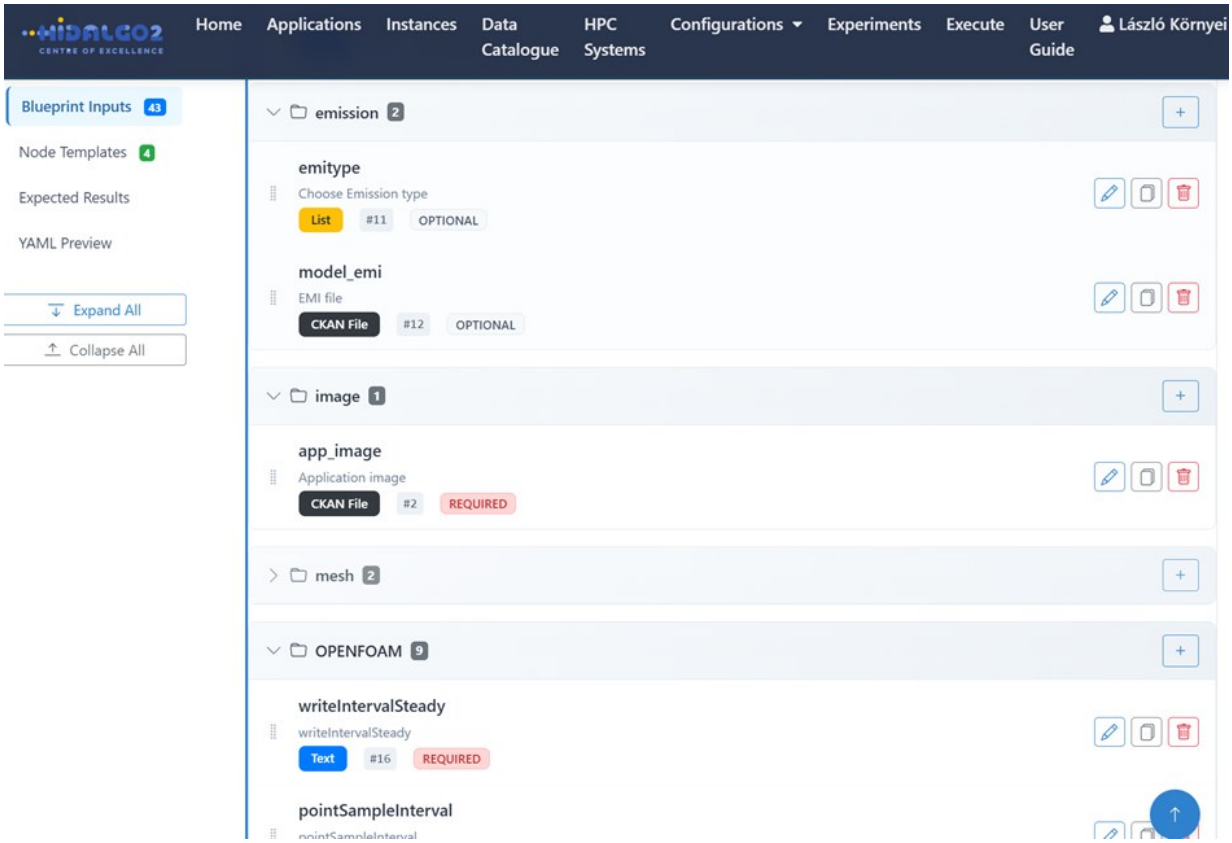


Figure 5: Blueprint editor showing the list of blueprint inputs for the edited application. Beside type, group and optionality, more advanced features can be chosen when editing.

3.1.2.4 Instance editor

Regular users of the MathSO portal configure applications with the instance editor (depicted in Figure 6). Blueprint inputs are configured by the user to set up the application for deployment to the HPC system. Instance editor design uses a split setup with group listing on the left pane, and edit fields on the right.

3.1.3 Implementation and Development Roadmap

In keeping with the style of D2.6, for the sake of clarity and page length, the section on implementation details and the development roadmap are placed in Annex II: MathSO Portal Implementation Status and Roadmap.

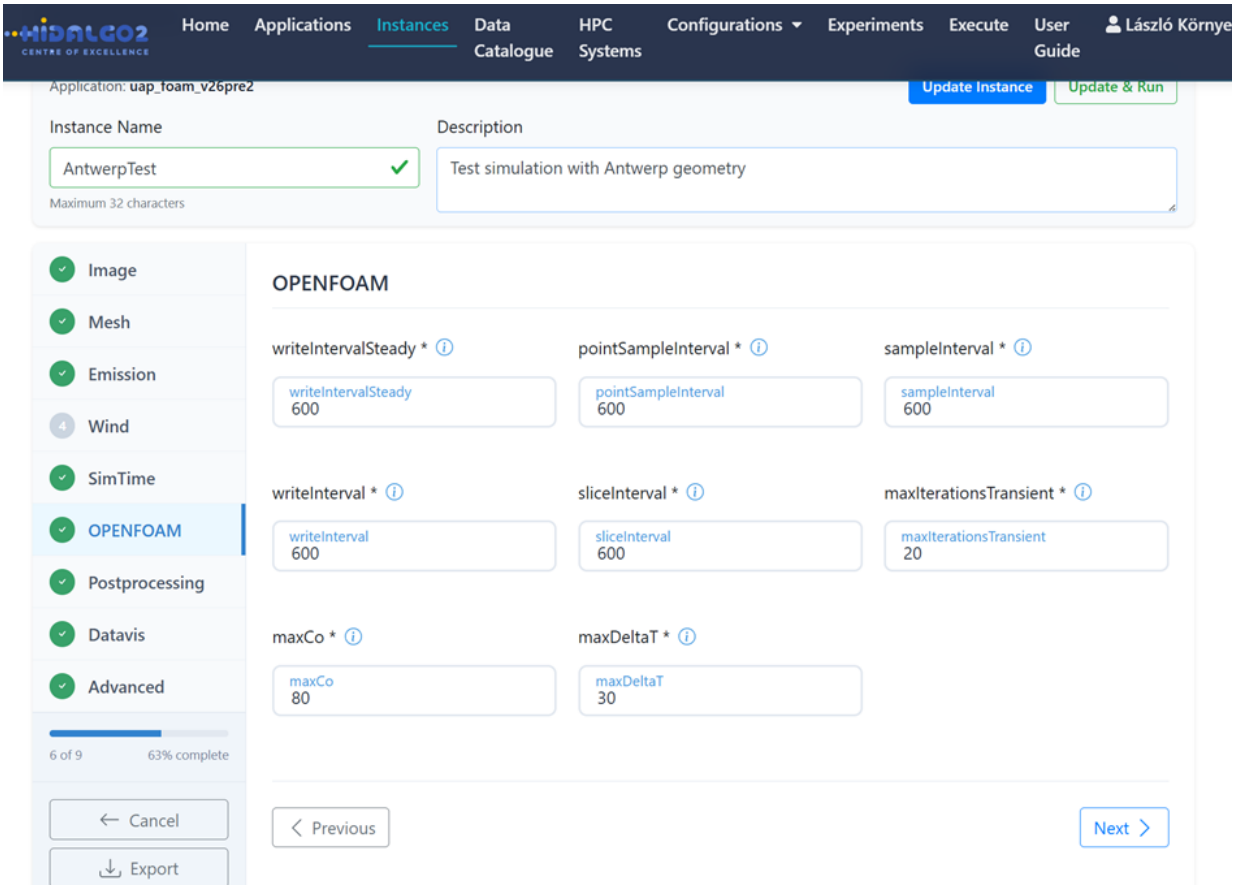


Figure 6: MathSO instance editor configuring UAP-FOAM.

3.2 QCG Portal

The development of the QCG-Portal Workflow Orchestrator over the past year has focused on enhancing flexibility of the solution and introducing new functionalities designed to streamline its use for the target stakeholders of the HiDALGO2 project. One of the key development activities was the implementation of a redesigned templating mechanism, which importantly enhances the customisability and functionality of the orchestrator. This development was not planned in the roadmap; however, it was driven by the realistic expectations of Pilot stakeholders, particularly RES, which have been addressed. While it led to some delays in the execution of the initial development plans, the remaining roadmap items can be readily realised as needed, depending on the specific requirements of the Pilots.

3.2.1 Requirements and Specification

The list of requirements collected in the previous reporting periods for specifying the functionality of QCG-Portal remains valid and has served as a reference for the development carried out during the last year. However, several discussions with the Pilot owners, aimed at identifying detailed expectations regarding workflow orchestration in their specific scenarios, led to a decision to modify the application template mechanisms offered by QCG and to redesign the GUI towards more Pilot-oriented design, tailored to the specific identified requirements. This activity was, a significant, additional effort, but it was essential to reach the ultimate goal, that is to provide a solution that will satisfy end users in terms of intuitiveness, usability, and aesthetics. The original version of the templates showed weaknesses in these areas, as the built-in customization options were limited to selected views, thus they did not allow to create a comprehensive and use case-oriented interface. Moreover, some more advanced mechanisms, such as workflow execution or progress monitoring, had to be always outsourced to external services, such as Airflow[3] and Grafana[4]. While these tools provide powerful support for complex workflows and executions, their use in moderately complex workflows can add an unnecessary overhead, potentially causing inconsistencies and reducing the interface's responsiveness and perceived lightness.

Therefore, the priority was to enhance the intuitiveness and overall user experience (UX) of the orchestrator. There were proposed several new mechanisms that extended or modified key functional areas of the orchestrator, making them more flexible and easier to adapt to individual needs of the project's Pilots. Consequently, by leveraging

[3] https://airflow.apache.org/
[4] https://grafana.com/

the newly introduced capabilities, HiDALGO2 Pilot owners can now build more consistent and refined interfaces, allowing end users to execute their scenarios seamlessly and in line with current user interface design standards.

The following list outlines the key new features introduced into the system over the last year.

3.2.1.1 Python-based Application Templates

The legacy templating mechanism has been grounded in the Jinja2[5] framework used with JSON descriptions. The templates allowed to define the basic layout of the submission form, as well as to define, in quite a complicated way, commands to run a job executed on a cluster. This mechanism was effective as long as the execution scenario was fairly typical. Complex use cases, like those in the HiDALGO2 pilots, required dedicated solutions and complicated custom developments. The new templating mechanism, based on the Python language, provides much more flexibility, and is more affordable to system administrators and application maintainers. With the new templates both the definition of the execution procedure as well as graphical layouts for each phase of the processing, can be easily and highly customised.

3.2.1.2 Task abstraction

Task abstraction is a newly introduced mechanism that generalises the concept of a task in QCG-Portal, making it no longer limited to jobs executed on HPC clusters via schedulers. The previous implementation treated a task strictly as an HPC job, which was neither universal nor sufficient to address the needs of heterogeneous workflows, as those in HiDALGO2, where not all jobs are executed on HPC computational nodes, but may also be executed in Cloud or PaaS environments. Thus, in the new implementation, a task can still represent a parallel job running on computational nodes, but it may also be a lightweight command, such as downloading input data on a cluster access node, a specific job in a remote cloud, or even an entire workflow integrating multiple task types.

3.2.1.3 Platforms

The concept of Platforms is closely tied to the new task model, as each individual task must be executed within a selected Platform. At this stage, two platforms are available: *Slurmctld*, which enables the execution of traditional HPC jobs on computational

[5] https://jinja.palletsprojects.com/

nodes, and *Slurm UI*, which allows selected commands to be run on the cluster's access node. Subsequent platform types, including the *PaaS* platform, will be introduced iteratively in accordance with the needs of Pilots.

3.2.1.4 Built-in workflow orchestration

One of the key benefits of the Python-based definition of application templates and task abstractions is the introduction of a new mechanism for executing workflows directly within QCG-Portal. This functionality does not fully replace the support for Airflow workflows described in D2.5, as it provides only basic orchestration capabilities. However, it offers an easy way to configure and execute workflows of low to moderate complexity. Managing workflows directly at the QCG-Portal level significantly streamlines the solution and improves usability through a more integrated and straightforward interface. Based on the conducted analysis, this mechanism is sufficient to support the HiDALGO2 Pilot scenarios and has already been successfully applied in the RES Pilot.

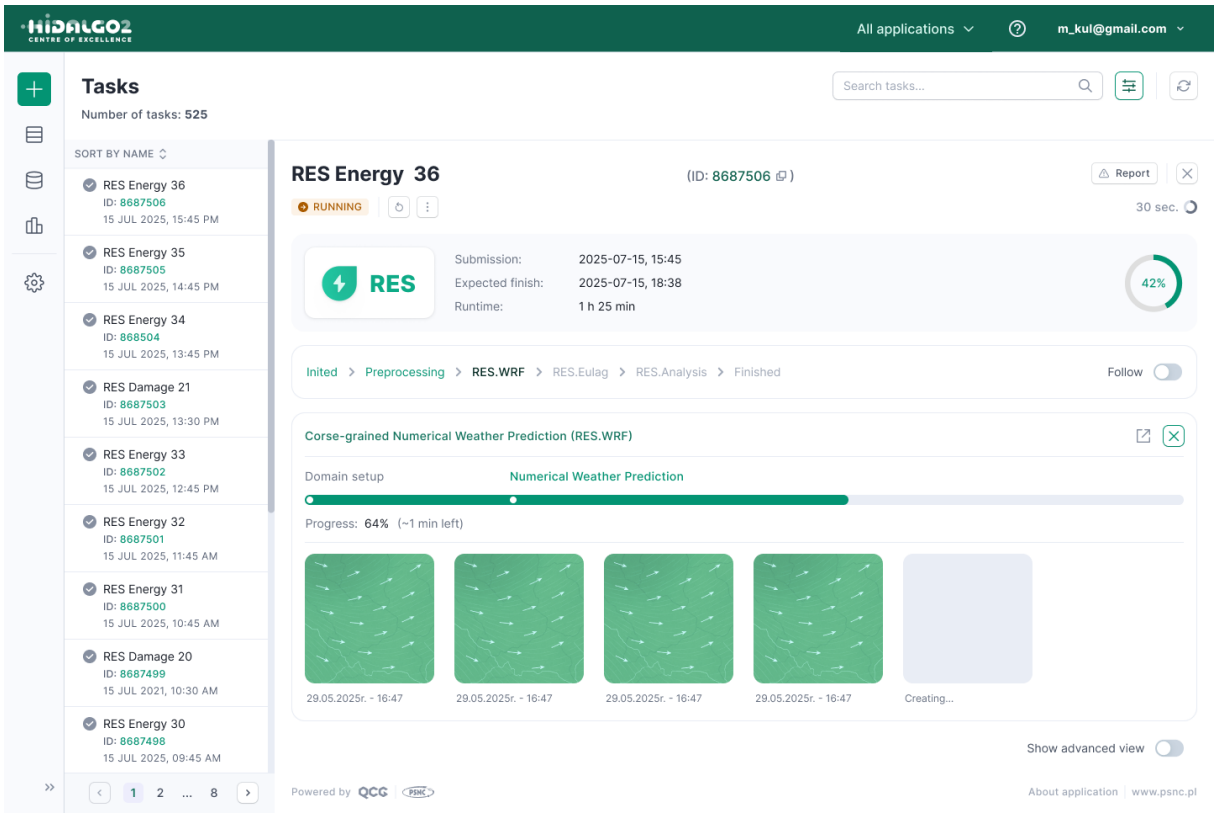


Figure 7: Monitoring of RES workflow execution in a customised view in QCG-Portal, using new templating mechanisms

3.2.1.5 New interface with built-in execution monitoring

The discussed changes in the back-end services were primarily motivated by the need to enable seamless development of graphical interfaces tailored to the requirements of individual applications, with the Pilot scenarios being the first beneficiaries. The previous version of the QCG-Portal interface had significant limitations in both the simplicity and the scope of customisation, which was restricted mainly to the job submission form unless fully custom application GUIs were developed with substantial programming effort. Moreover, since some parts of the GUI originated from external code bases (such as custom GUIs or Grafana), it was extremely difficult to create use case–specific interfaces with a consistent look and behaviour, which is essential for HiDALGO2 use cases.

The newly implemented graphical interface, built using the *shadcn library*[6], reflects the architectural and templating changes introduced in the system. It provides a modern look and feel, along with harmonised customisation capabilities for all stages of workflow execution in HiDALGO2, namely:

- preparation of input data,
- monitoring of execution progress, with the ability to track both overall workflow status and individual steps independently,
- presentation of results, including basic visualisation mechanisms.

Figure 7 above showcases the main view of the redesigned interface, presenting a list of executed workflows. Figure 8, in turn, illustrate the views generated using the new templating mechanism for the RES Pilot, showing workflow execution progress monitoring and result presentation, respectively.

3.2.1.6 Application Interface Builder

The development of graphical layouts of templates in Python, without the possibility to visually see the design, would be poorly ergonomic. To streamline this process, a dedicated graphical Application Interface Builder for the definition of custom template views has been introduced (see Figure 9). This early implementation of the tool allows users to interactively design layouts, which can then be exported and integrated with the Python code of a template, significantly simplifying the overall template creation process. At this stage, the builder's capabilities are limited to creating template layouts from a set of predefined graphical components designed with a special focus on the RES Pilot requirements. This set will be iteratively extended to address the needs of

[6] https://ui.shadcn.com/

other HiDALGO2 use cases. Furthermore, the builder is ultimately intended to support the creation of workflows and individual execution procedures as well.

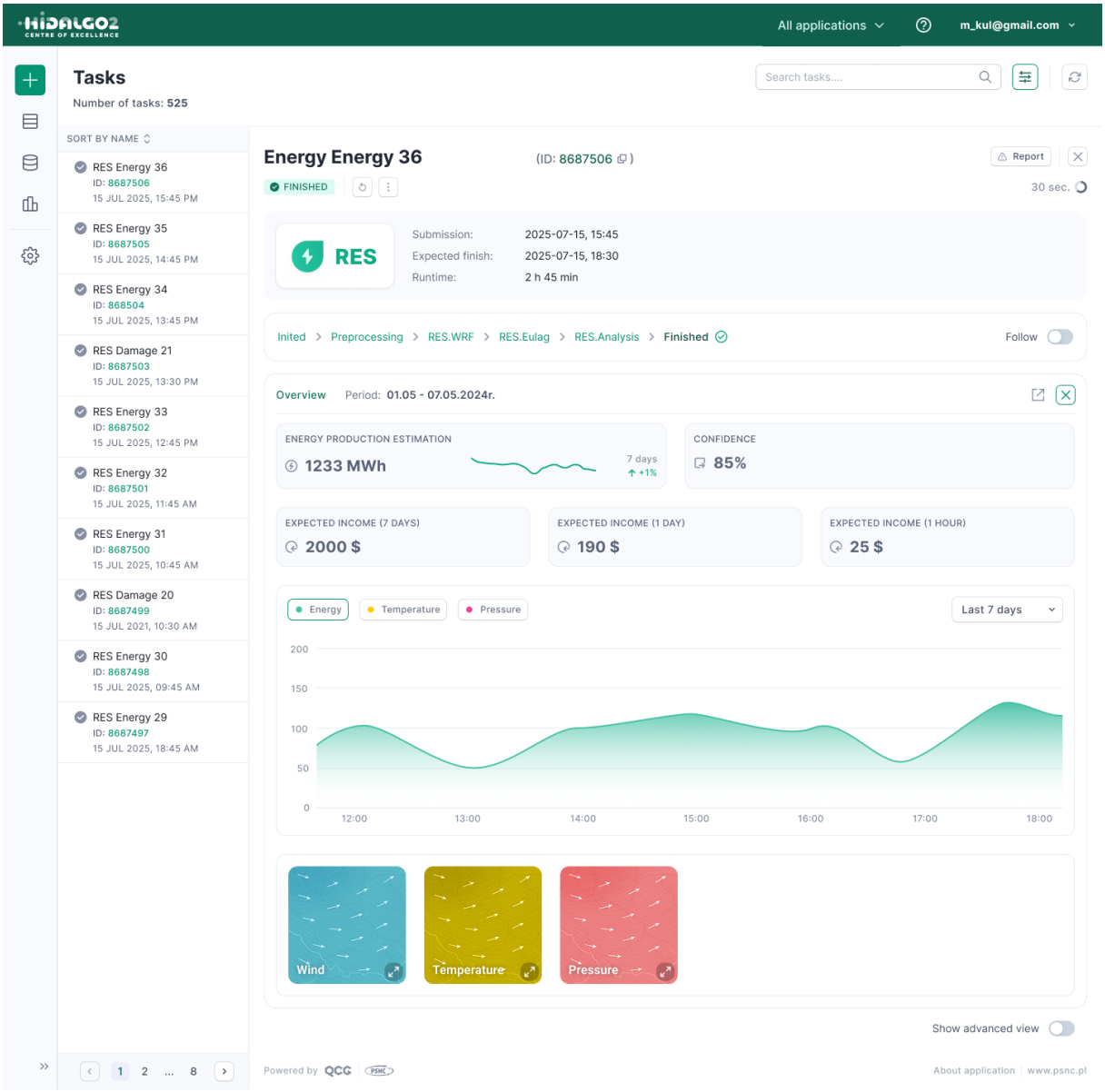


Figure 8: Presentation of RES Pilot results in a customised way using new templating mechanism

3.2.1.7 mUQSA

Over the past year, the mUQSA platform has been extended with several new features addressing the expectations and feedback reported by end users. In particular, the updates included extensions to Quasi Monte Carlo method and enhanced visualisation of statistics for scalar QoIs and Sobol's Indices requested by RES Pilot. In addition, in

response to the feedback collected after the HiDALGO2–CIRCE–SEAVEA VVUQ Hackathon, a new tabular mode for specifying input parameters has been introduced.

For the full reference of identified requirements see D2.1 [3] and D2.2 [4]

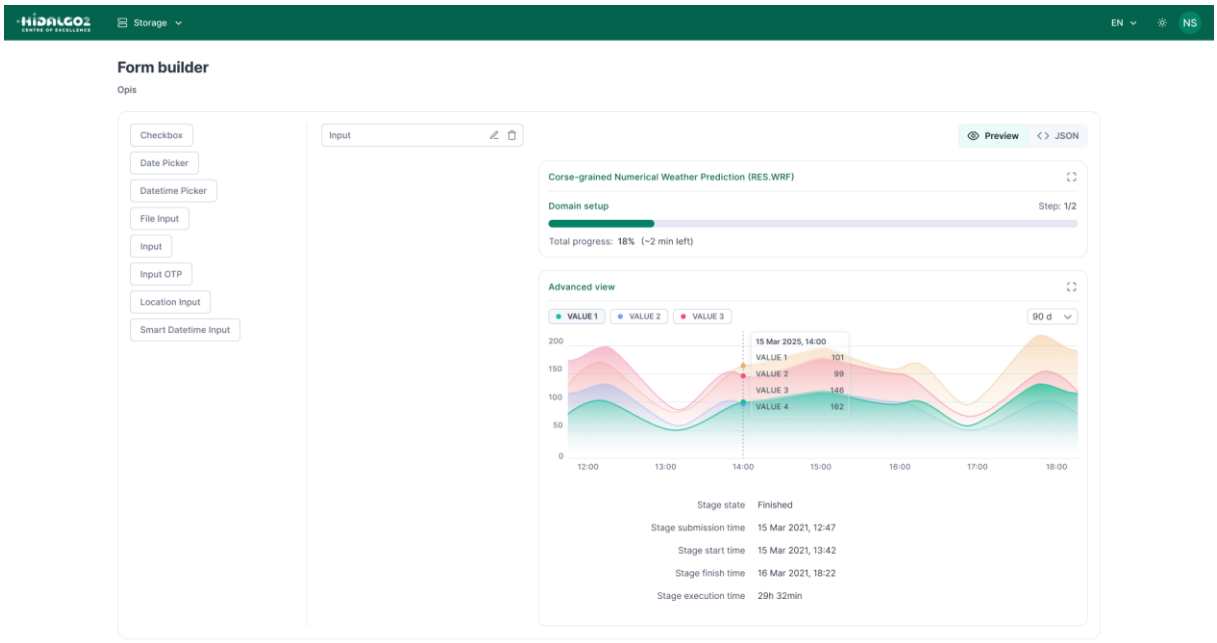


Figure 9: The interface of the Application Template Builder. The builder allows to interactively define layouts from the provided set of components, such us text boxes, tables or charts. This set can be extended with new components depending on needs

3.2.2 Design and Architecture

Since the last reporting period, the architecture of the QCG Workflow Orchestrator has been slightly reshaped to address the recognised HiDALGO2 Pilot requirements. The modifications reflect changes resulting from the introduction of a new templating mechanism and are primarily related to the system backend, where several new components have been introduced. The schematic diagram of the new architecture focusing on this part of the system is presented in Figure 10. There are a couple of new key concepts incorporated into the architecture, namely:

- **Application Templates** – which play a role of mediator between the graphical user interface and the underlying infrastructure. The templates are associated with individual applications, which allows to adjust the interface to specific expectations and deliver the requested level of User experience (UX).

- **Platforms** – represent different types of resources that QCG-Portal is able to connect to. Two main platforms are Slurmctld[32] , which allow to interact with the Slurm queuing system, e.g. to submit and monitor a job, and Slurm UI platform, which allow to run specific commands on the cluster's frontend node, e.g. copy files. The platforms can be employed in different workflow configurations for the realisation of specific application goals.
- **Agents** - handle low-level communication with system resources through defined interaction protocols. Currently, the framework supports an SSH-based agent that executes specific commands (defined by the Platforms) on remote systems, in particular EuroHPC sites available to HiDALGO2.
- **Accounts** – store user account credentials required to log-in to specific resources. The configuration of accounts may not be needed for some resources, e.g. when they support SSH Certificates.

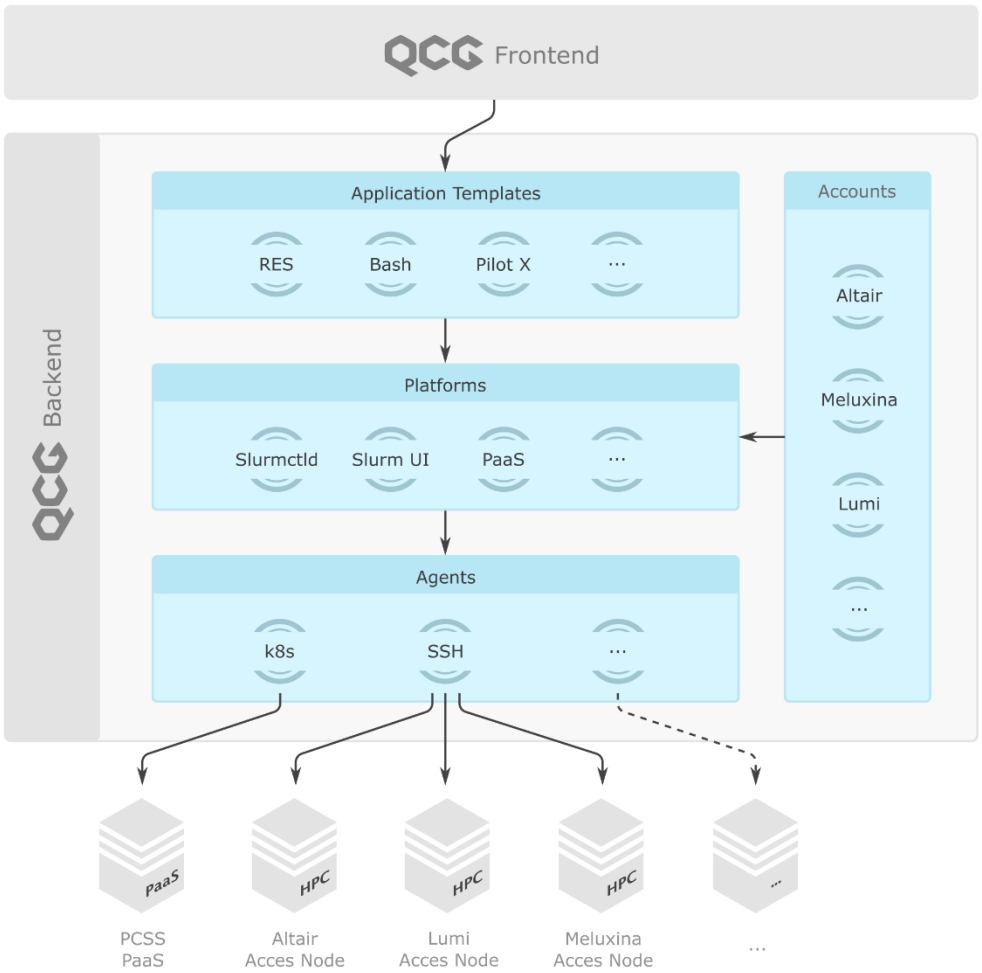


Figure 10: Architecture of the QCG Backend services

The redesigned QCG-Portal architecture enables lightweight workflow creation directly within application templates, eliminating dependence on external workflow engines for moderately complex tasks. This improvement also ensures a more uniform and intuitive user interface, with built-in mechanisms for monitoring of the progress of execution and visualisation of results.

3.2.3 Implementation and Roadmap

To improve the readability of the overall deliverable, this section on current implementation and the developmental roadmap, along with an updated Gannt chart, for the QCG portal have been moved to Annex III: QCG Portal Implementation Status and Roadmap.

4 Energy Monitoring and Optimization for HPC

This section will focus on the current state of the tools we proposed in the previous deliverable. If the reader wants a broader view of this development, please refer to deliverable 2.5[1] , section 4.
We also encourage you to read Annex IV: Energy Management Suite to review the current implementation and results of the developed modules. Due to the large size of this deliverable, we have moved the implementation and results of the developed modules to Annex IV: Energy Management Suite, instead of this chapter here.

4.1 Requirements and Specification

All details of the requirements and functional specifications remain unchanged in this deliverable, so please refer to deliverable 2.5[1] for more details.

4.2 Design and Architecture

We are still maintaining our plan to develop energy tools aimed at enhancing energy management. To achieve this, we proposed the creation of four tools that will be integrated into the Eviden Argos Suite: The Power and Performance Estimator (PPE), the Job Energy and Performance Comparator (JEPC), the Job Energy Predictor (JEP), and the Job Energy Recommender (JER).

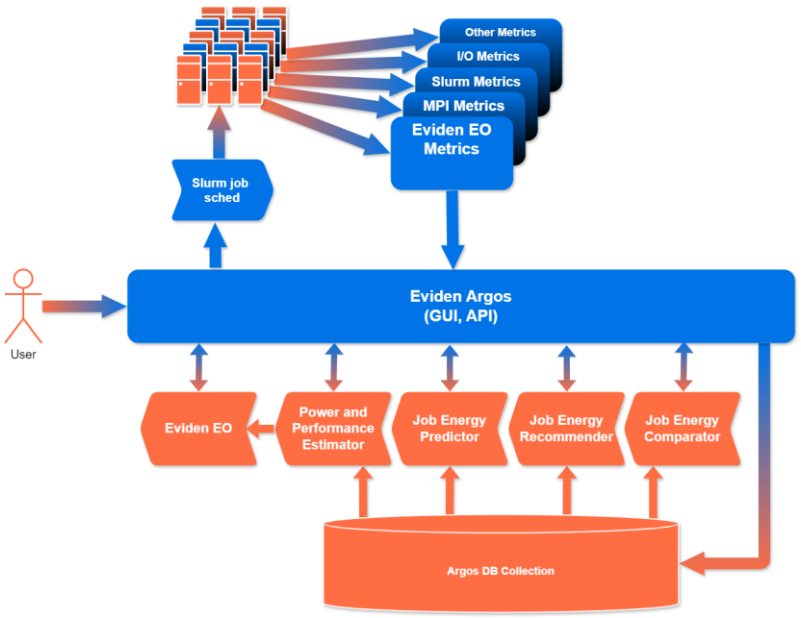


Figure 11: Architecture of the Argos Energy Monitoring and Optimization framework.

Figure 11 (extracted from the previous deliverable 2.5[1]) illustrates how these four modules are integrated into the Eviden Argos Suite. They are powered by the Argos DB and provide their functionality through an API that will connect to the Argos GUI.

4.2.1.1 Power and Performance Estimator

The PPE tool was designed to characterize and predict the behaviour of the target job, and its implementation was based on the results of Antici et al[28] . Given the outcomes of that work, we decided to develop a custom implementation of it.

Figure 12 depicts the architecture of our custom implementation of the PPE tool, which includes three workflows. The Ground Truth Tagger is responsible for retrieving data from ArgosDB and characterizing whether the selected job is memory-bound or compute-bound. This is what we consider the job behaviour.

The inference workflow is responsible for making predictions about the behaviour of the incoming job, such as determining whether it is memory-bound or compute-bound. The training workflow is executed by an internal process to ensure that the model remains up to date. It is daily trained using data from the last 30 days to maintain its relevance and accuracy.

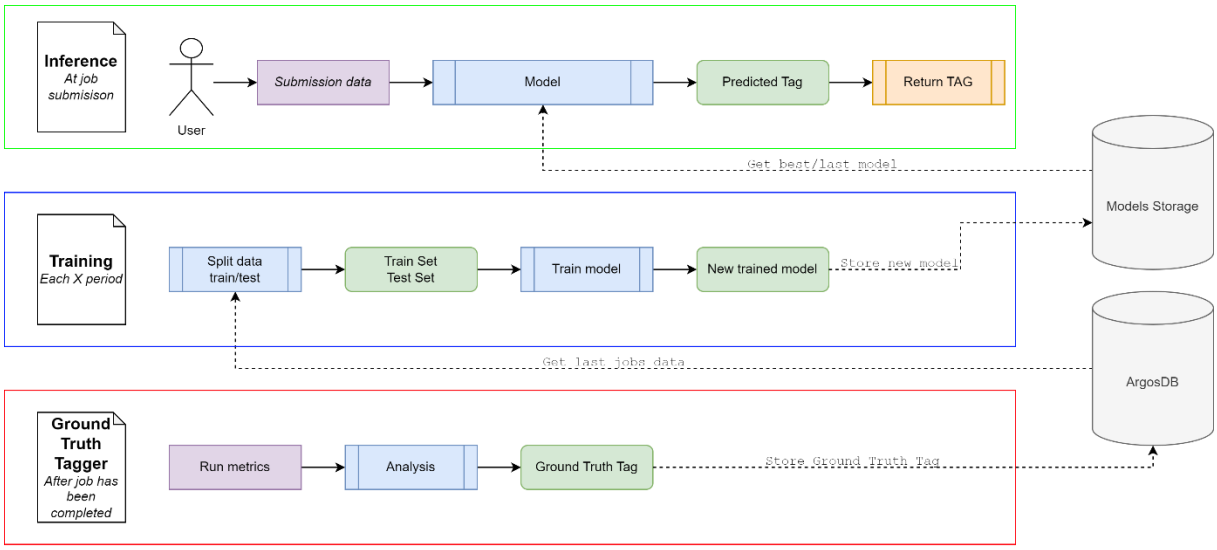


Figure 12: PPE architecture

4.2.1.2 Job Energy and Performance Comparator

The JEPC tool was originally designed to compare jobs across different runs in terms of energy and performance, a functionality referred to as Comparator. Additionally, JEPC incorporates another functionality called Scalability, which is based on Amdahl's law[29] and aims to demonstrate how jobs can scale by adding more resources.

Figure 13 illustrates the use case diagram for the Comparator functionality. In the Argos GUI, users can view, select, and assign app IDs to a set of jobs. After selecting the jobs or app IDs for comparison, the system calculates a weighted metric to evaluate the set based on energy and performance. Then, the system presents the user with up to five of the most interesting jobs, highlighting the best trade-offs between energy and performance.

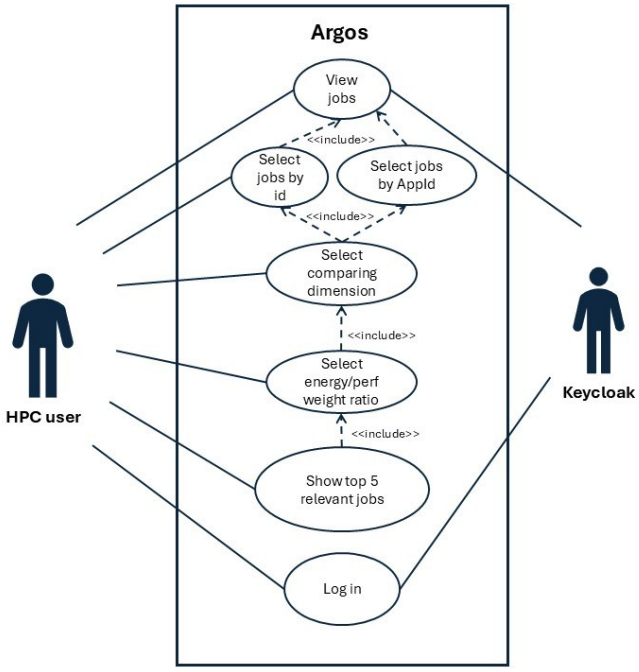


Figure 13: Comparator use case diagram

Figure 14 illustrates the interface of the Comparator tool, where users have the capability to adjust the energy/performance widget to prioritize the trade-off between energy and performance. If a user wants to focus solely on performance, they can slide the

widget all the way to the left. The ranking table displays various icons to highlight specific jobs: for example, a thumbs-up icon signifies the job with the best energy/performance ratio, while a thumb-down icon indicates the job with the least favourable ratio.

Figure 15 pertains to the Scalability tool, with the image presented being a draft of its final appearance. This tool requires two jobs assigned to the same app ID[7] to estimate the parallel part of the program. Once this estimation is made, the Scalability tool can suggest the number of compute units (e.g., nodes, cores) that the user should use to achieve a certain percentage of the maximum theoretical speed, as defined by Amdahl's law[29] .

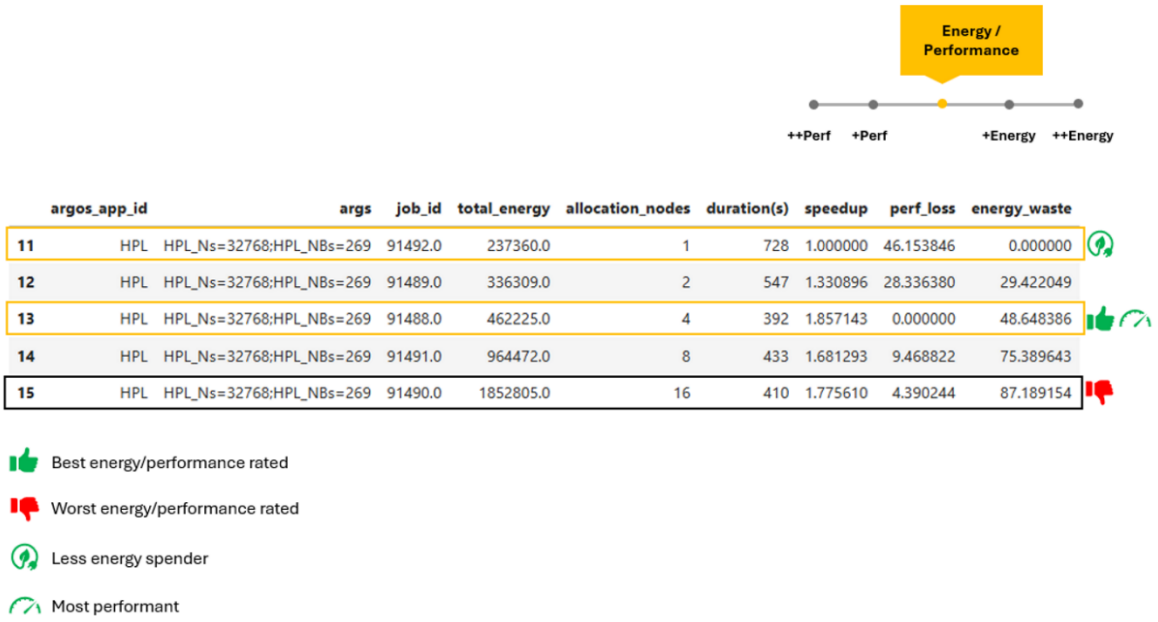


Figure 14: Comparator mock-up

4.2.1.3 Job Energy Predictor

At the time of writing this document, we have just begun developing this module, and there is not enough progress to present any significant work at this stage.

[7] An app ID is a tag set by the user to identify different job runs as the same application.

4.2.1.4 Job Energy Recommender

In the status of the roadmap, we did not start this tool yet.

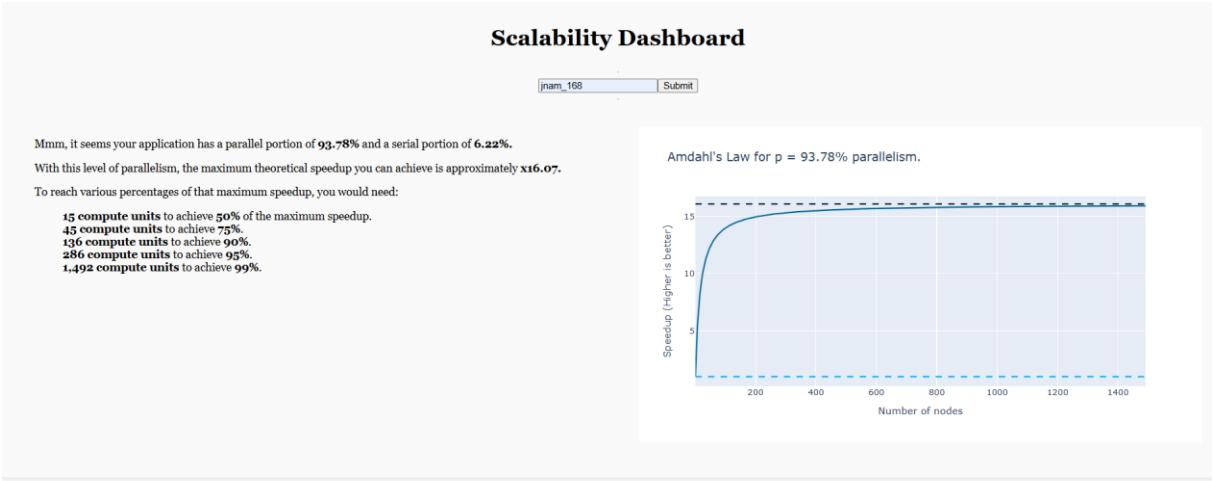


Figure 15: Scalability mock-up

4.3 Implementation and Development Roadmap

To maintain the page count and readability of this deliverable, Implementation details and results have been moved to section to Annex IV: Energy Management Suite, as well as the development roadmap. Figure 36 in Annex IV: Energy Management Suite illustrates the execution of the plan in relation to what was initially planned (M34).

5 Component Integration and Deployment

5.1 Overview

5.1.1 HiDALGO2's strategy for CI/CD

HiDALGO2's strategy for CI/CD has been described in detail in the previous deliverables of this series, in D2.4[2] and D2.5[1] . Basically, the strategy differentiates between CI/CD for services and CI/CD for pilots, as well as a special highlight on how the CI/CD for pilots' interfaces with CI/CD for EuroHPC systems.

As there are no significant changes or updates in the requirements and design of the **Services CI/CD System** since D2.5, this deliverable will focus on the core part of HiDALGO2, the pilot code applications and updates in the development of their CI/CD pipelines, as well as how they interface with EuroHPC systems. However, a note is made later in section 5.7, on a planned attempt for next year, for migrating the currently split, local Services CI/CD systems at USTUTT and PSNC premises, to a common CI/CD system on the recently set up HiDALGO2 GitLab instance, instead.

For clarity's sake, the division used in D2.5 between System 2, or **Pilots CI/CD system**, and System 3, the **EuroHPC CI/CD system** is unified. The pipelines consist of CI, or continuous integration, where the code is built and tested on local development platforms, and CD, or continuous deployment, where code is delivered to EuroHPC systems, where it can then be separately run and executed. In some cases, there is also an extra step, CB, or continuous benchmarking, where automated benchmarking or test execution of the code is carried out on the target platforms.

5.1.2 Public facing EuroHPC CI/CD

Even though D2.6 describes a unified CI/CD system for the pilot codes, a special note is still made regarding the connection to current, and any future, public facing CI/CD platforms from the EuroHPC community, such as the efforts by CASTIEL2. Here, the flexible strategy adopted by HiDALGO2 and detailed in D2.5 has paid off. At the time of the writing of D2.5, it was not yet sure which direction CASTIEL2 will take in choosing their preferred solution for a common, public facing CI/CD platform.

HiDALGO2 pilots have an independent, local CI pipeline, where the product of that CI build is then interfaced through CD pipelines supported by CASTIEL2. This includes both of the two solutions now suggested by CASTIEL2, the EuroHPC GitLab platform, with deployment of codes through GitLab runners[21] , and the EESSI platform[15] , which deploys Easy Build[17] recipes through the Cern VMFS[18] system. Details for these systems, and HiDALGO2's use of them are covered in D2.5, and will be explained per pilot case bases in the following sections. An overview of how the systems interface is given in Figure 16.

Furthermore, HiDALGO2 pilots have explored even more ways of deploying to EuroHPC systems, to future proof their delivery and deployment. This includes deployment through HiDALGO2's workflow orchestrator, MathSO, as well as external methods such as GitHub Actions. These methods are also explained in the next sections, and ensure that HiDALGO2 pilots can switch and sustain different forms of CI/CD pipelines, also after CASTIEL2 reaches the end of its project term.

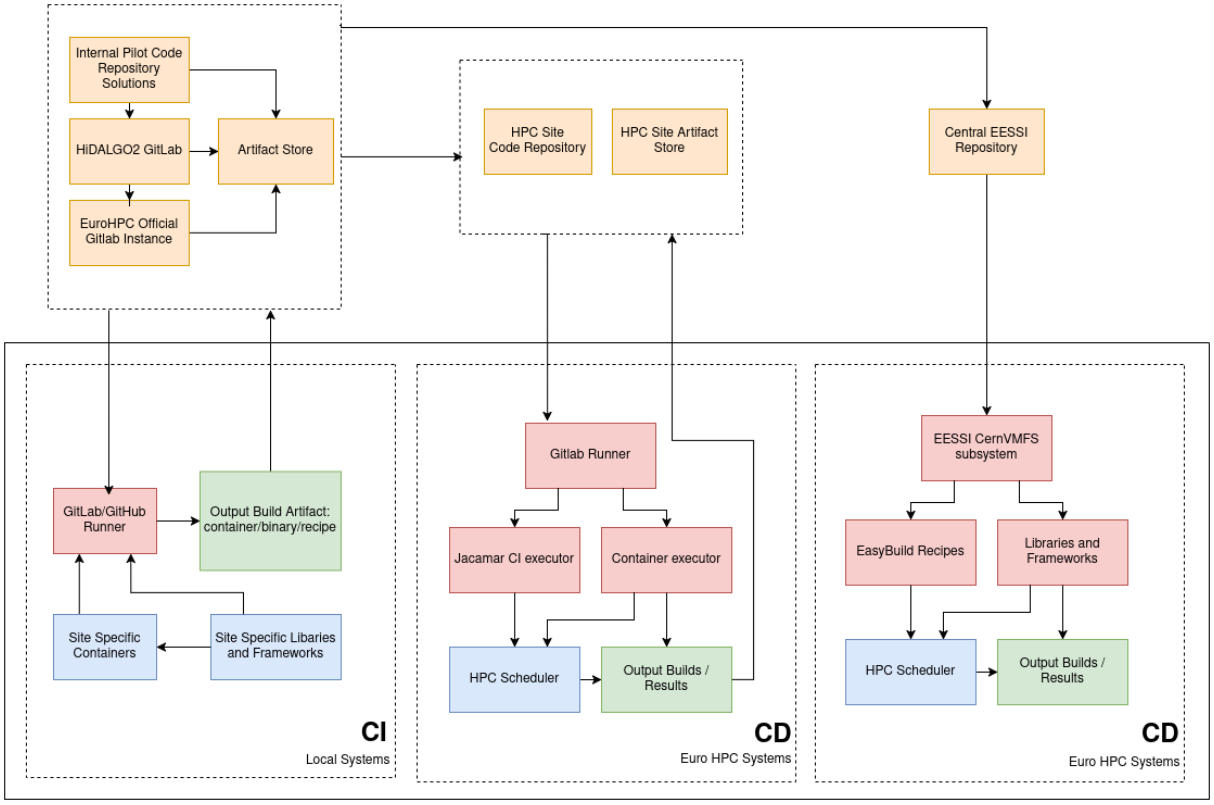


Figure 16: Interfacing systems from Pilots CI/CD to EuroHPC CI/CD

5.2 CI/CD for Urban Air Pollution

The UAP pilot leverages the workflow system of the MathSO workflow orchestrator to realize CI/CD functionality within its workflow. The container-based application execution itself serves for deployment purposes. Using the execution system of the orchestrator, container creation itself can be also done within the portal, satisfying the need for integration. Integration with DMS systems like CKAN and local SSH repositories makes the transfer of the container images more efficient and seamless, enabling the copy containers directly to the HPC systems before execution.

While there are some minor feature updates with regards to the CI/CD system itself, there has been no major update since D2.5. The roadmap will adhere to current needs and challenges. The SZE team is also considering integrating the deployment method supported by EESSI. While parts of the workflow that are supported by EESSI modules could be substituted by these, non-free part would still use standard deployment by containers.

5.2.1 Design

As already detailed in D2.5, the MathSO CI/CD system is designed with a dual actor mindset, as depicted on the figure below. A developer will develop code, create application container and test it deploying the image to the developer (HPC) systems, while a user will run the application with deploying code to the user HPC system. The central focus of the system is twofold. First, to use the HPC resources for container creation, improving creation time. Second, to enable offsite container creation for proprietary and sensitive code base separately and focus on code execution only, so user HPC system does not have access to application source code.

5.2.2 Implementation

The current implementation of the MathSO CI/CD pipeline is extensively described in the resubmitted version of the deliverable D2.4. There are a couple of important additional features within the MathSO portal, that is supporting seamless deployment to the HPC systems.

First, a two-panel file transfer web user interface, as shown in Figure 18, was developed, that supports transfer between different data repositories, including CKAN-SFTP, CKAN-CKAN and SFTP-SFTP connections (As this work is tightly coupled to the work done in T4.1, more details will be reported in D4.2)

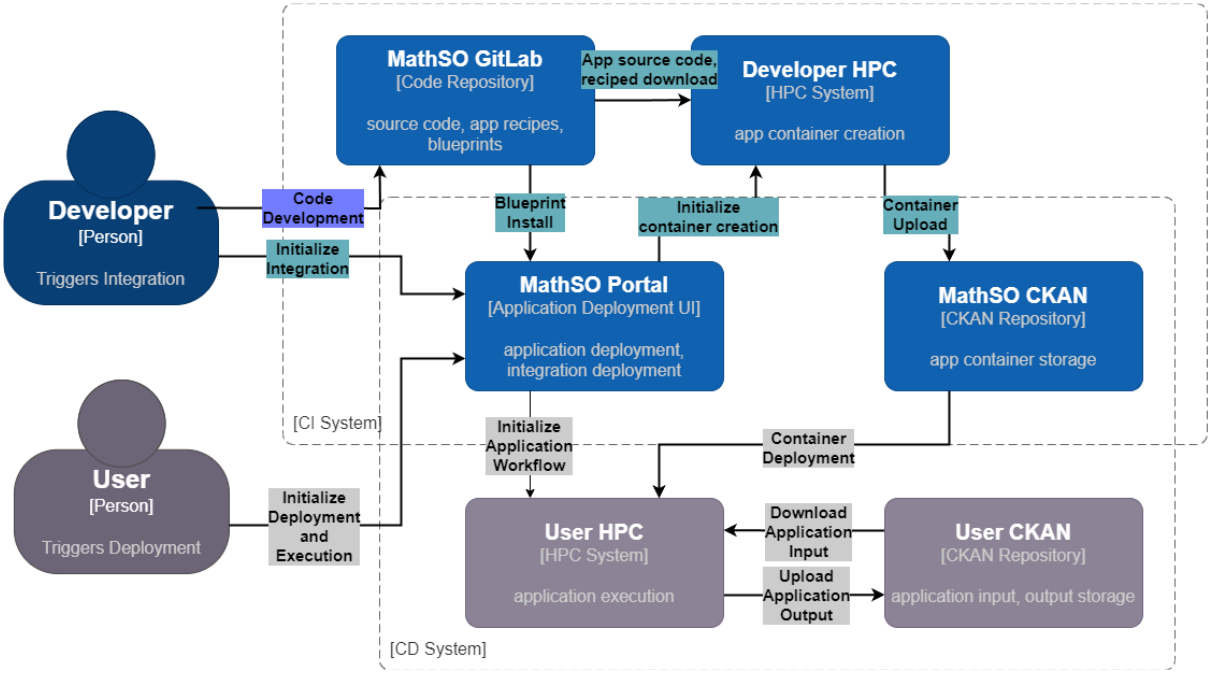


Figure 17: Design of the MathSO CI/CD System

This way application images can be deployed separately from application executions, so transferring the image will not delay actual execution later. This file transfer feature is not limited to application images, so persistent input data can be preloaded, or simulation results can be downloaded or transferred to other systems, like the HiDALGO2 JupyterHub for post processing.

Second, while deploying the application, a local application image can also be selected for running. This way, retransfer or copying can be avoided, even symbolic link or a direct reference can be used for execution, saving disk space and transfer time.

Lastly, documentation of the MathSO system includes CI/CD a tutorial, so external users can also use it efficiently.

5.2.3 Development Roadmap

The development of the MathSO CI/CD system is planned along the following roadmap, which has been updated to reflect the current requirements and time frame of the HiDALGO2 project.

- Improve documentation, including the testing of a separate tutorial section within the MathSO portal tutorial (ongoing, M48).
- Investigating the support for the EESSI system within the MathSO portal. While the developer can access EESSI directly by loading the specific modules, there

is currently no direct support for finding the environment or loading modules (M48).

- Improve GitLab integration for streamlined CI and blueprint installation. While the developer can directly access GitLab features within the portal, the is currently no dedicated user interface for this feature (M48).

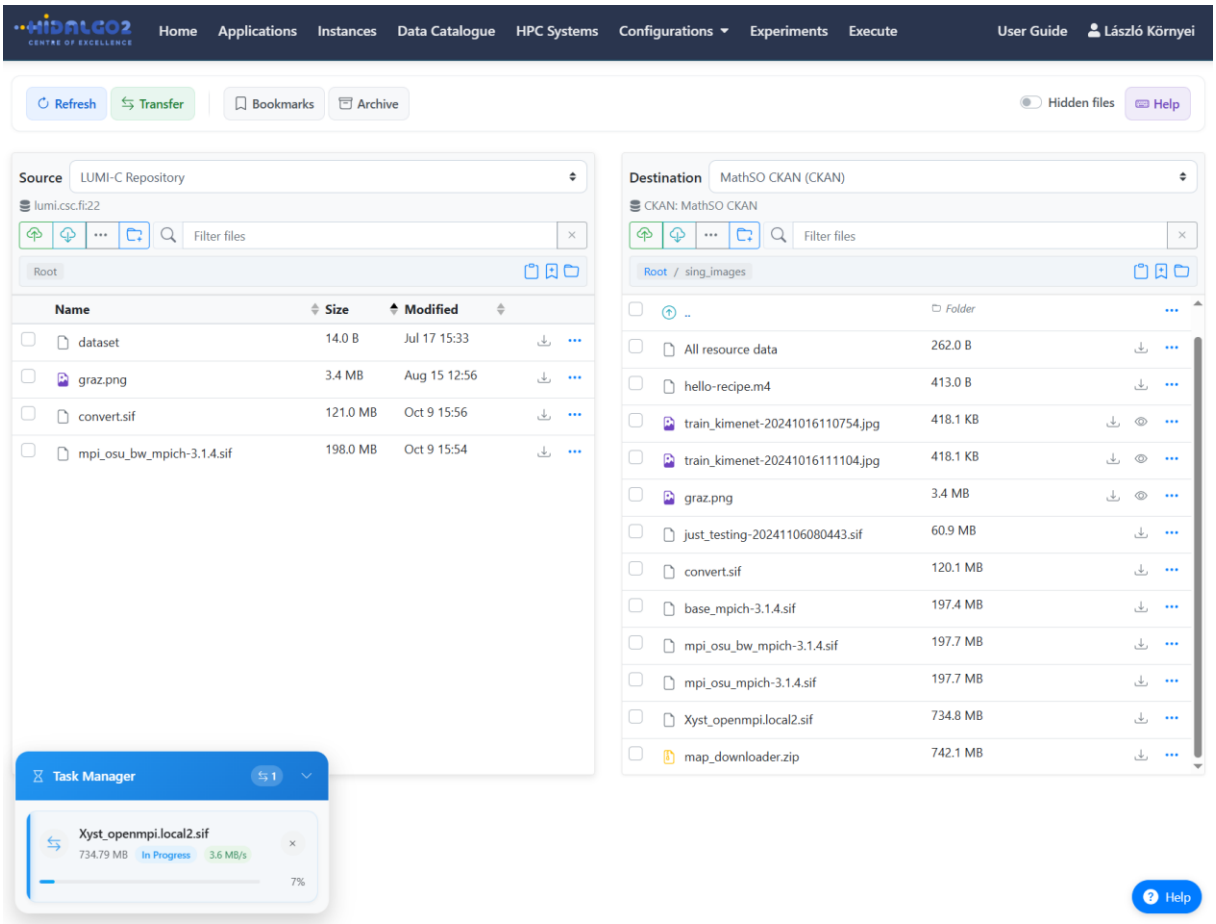


Figure 18: Two-pane data transfer interface within the MathSO portal for application image transfer. The transfer of Xyst is shown onto the LUMI HPC System.

5.3 CI/CD for Urban Buildings

The partner **UNISTRA** operates a CI/CD process that turns the **Feel++/Ktirio** stack into a practical HPC service. Code changes trigger builds, tests, packaging and releases, and authenticated events forward runs to EuroHPC through MathSO and the kub-cd and bench.ktirio.fr repositories. Ktirio GUI is automatically deployed to ktirio.hidalgo.eu with three routes that expose the latest release, the main-branch build and per-pull-request previews. The platform publishes meshes, fields, logs and reports to HiDALGO2's supported data portals: CKAN, Girder and Zenodo and exposes dashboards that summarize performance and results on cases.ktirio.fr. The current implementation delivers stable containers, automated deployments and multi-site runs, and it is extending test depth and provenance. The roadmap focuses on security hardening, self-hosted runners, WebAssembly packaging and performance accounting in order to guarantee that urban-scale simulations are reproducible, portable and auditable across EuroHPC sites.

5.3.1 Design

The partner UNISTRA operates a continuous integration and delivery strategy to run the Feel++/Ktirio stack as High-Performance Computing Software-as-a-Service. The strategy covers the complete path from source code to execution on EuroHPC systems and other registered machines, including containerized builds, automated testing, packaging, release, deployment, remote job orchestration, and publication of data and reports to FAIR-compliant repositories. The guiding principles are reproducibility through pinned toolchains and container images, traceability through systematic capture of parameters, datasets and run metadata, and portability through the Docker-to-Apptainer conversion used on heterogeneous clusters.

The overall architecture is organized around a small number of triggers and services. Pull requests, release tags and authenticated REST events initiate pipelines on GitHub Actions or GitLab CI. Build runners produce container images and software artifacts. The GUI forwards simulation runs to EuroHPC queues through three complementary components: MathSO and the GitHub repositories "kub-cd", "bench.ktirio.fr", which acts as a unified deployment orchestrator for EuroHPC and any other machine that has been registered. This approach does not depend on Kubernetes; where needed, services are composed with Docker Compose and converted to Apptainer for execution on HPC systems. Artifacts produced by the pipelines include Docker and Apptainer images, Python wheels, functional mock-up units, three-dimensional meshes and fields, logs and human-readable reports. The platform publishes these artifacts to CKAN, Girder and Zenodo, which ensures durable access and citation.

The user-facing service is Ktirio GUI, which is automatically deployed at ktirio.hidalgo.eu. The deployment exposes three routes that map directly to the software life-cycle. The /stable route serves the most recent release, the /testing route serves the continuous integration build from the main branch, and the /pr-XYZ route exposes an ephemeral preview of pull request number XYZ. These routes are created and removed automatically, which allows external users and reviewers to evaluate new features without waiting for a formal release and to compare behaviour across versions.

Data and control follow a simple processing chain. Input tools (the Ktirio GUI, CLI tools and benchmarking utilities) prepare scenarios, geospatial layers and weather data. The orchestration layer (MathSO, QCG and kub-cd) receives these inputs and submits jobs to EuroHPC queues, typically backed by SLURM. Upon completion, the runs publish their meshes, fields, logs and reports to the data platforms, and the output tools (the Ktirio GUI, ParaView, and the Benchmarking-as-a-Service sites) render the results to end-users.

The following matrix (Table 4) summarizes the main pipelines; a narrative description follows to clarify the current maturity. The matrix is informative and does not replace the textual description that follows.

Table 4: Main Ktirio pipelines matrix summary

Component / Stage	Build	Test	Package	Release	Deploy	HPC-Run	Docs
Feel++ (core libraries)	Done	Done	Done	Done	n/a	n/a	Done
Ktirio Urban Building	Done	In progress	Done	Done	Done	Done	In p.
Ktirio GUI (Desktop/WASM)	Done	In progress	Done	In prog	Done	n/a	In p.
Benchmarking (BaaS)	Done	Done	Done	Done	Done	Done	Done
Docker→Apptainer images	Done	Done	Done	Done	Done	n/a	n/a

In practical terms, the Feel++ core libraries and the Benchmarking-as-a-Service site operate end-to-end pipelines from build to publication, and they are considered stable.

The Ktirio Urban Building engine is built and packaged systematically, and it is already exercised through production runs; the extension of test coverage and documentation is being carried out. The Ktirio GUI is built and packaged on every change and is deployed automatically to ktirio.hidalgo.eu, while its dedicated test and documentation pipelines are being reinforced. Multi-architecture Docker images are converted to Apptainer for execution on EuroHPC systems. The list of targeted compute sites includes LUMI, MeluXina, Karolina, Vega, Discoverer. Leonardo, MareNostrum and Deucalion deployments are undergoing.

Quality and security are enforced by policy rather than by convention. Protected branches and mandatory reviews guarantee that only validated changes reach the release channels. Secrets are handled either through OpenID Connect or through encrypted organization secrets when OIDC is not available, and the repositories apply container linters and source code formatters (clang-format and clang-tidy for C++ and ruff for Python). Image scanning is in place. Reproducibility is supported by pinned base images on **ghcr.io** (the GitHub container registry) and deterministic builds and by explicit dataset versioning on CKAN, Girder and Zenodo.

5.3.2 Implementation

The implementation status is presented from the perspective of the user pathway. The automatic publication of **Ktirio GUI** to ktirio.hidalgo.eu is operational. The service exposes the /stable route for releases, the /testing route for the main branch, and on demand the /pr-XYZ route for each pull request, with the corresponding endpoints created and retired automatically. This mechanism enables non-regression checks and user acceptance testing without friction and provides a permanent “try-it-now” entry point for stakeholders.

The orchestration of deployments and runs is centralized in the public GitHub repository **kub-cd**[8]. This repository provides a single control plane for heterogeneous environments: it is used by Ktirio GUI to submit production runs to EuroHPC systems and to any other registered machine, and it applies the same conventions for image selection, parameters, credentials and provenance across targets. The orchestration layer interoperates with MathSO when appropriate and interfaces with site schedulers such as SLURM for execution. The approach deliberately avoids Kubernetes; service composition is handled with Docker Compose on the submission side, while Apptainer images are executed on HPC systems to ensure portability and policy compliance.

[8] https://github.com/feelpp/kub-cd

The platform also operates **Benchmarking as a Service** through the **feelpp-benchmarking[20]** stack. Benchmark campaigns are triggered automatically from bench.ktirio.fr, and the corresponding repository collects and publishes the resulting reports via GitHub Pages. This integration turns each benchmark run into a traceable artifact: the container image, the code revision, the parameters and the performance summaries are persisted and can be reviewed from the public site. In parallel, the Ktirio pipeline produces meshes and fields, logs and human-readable reports that are published to Girder and CKAN (the project data portal), ensuring durable and citable access to inputs and results.

The **data-to-compute** loop is already exercised end-to-end. Once Ktirio GUI has prepared a case and the associated assets are published in CKAN, MathSO can submit jobs to EuroHPC systems or to any machine where accounts are available, using the same conventions and credentials model as kub-cd. In this configuration, the platform effectively offers HPC as a Service—through Ktirio GUI for production scenarios—and Benchmarking as a Service—through feelpp-benchmarking and bench.ktirio.fr—which together provide a robust and scalable environment for large-scale simulations and systematic performance testing.

Observability covers both pipeline events and HPC executions. The CI systems report build duration and success rates; the orchestrators capture queue latency, job start time and time-to-solution; and the published reports include, when available, estimates of CPU-hours consumed.

Figure 19 gives a detailed overview of the implementation.

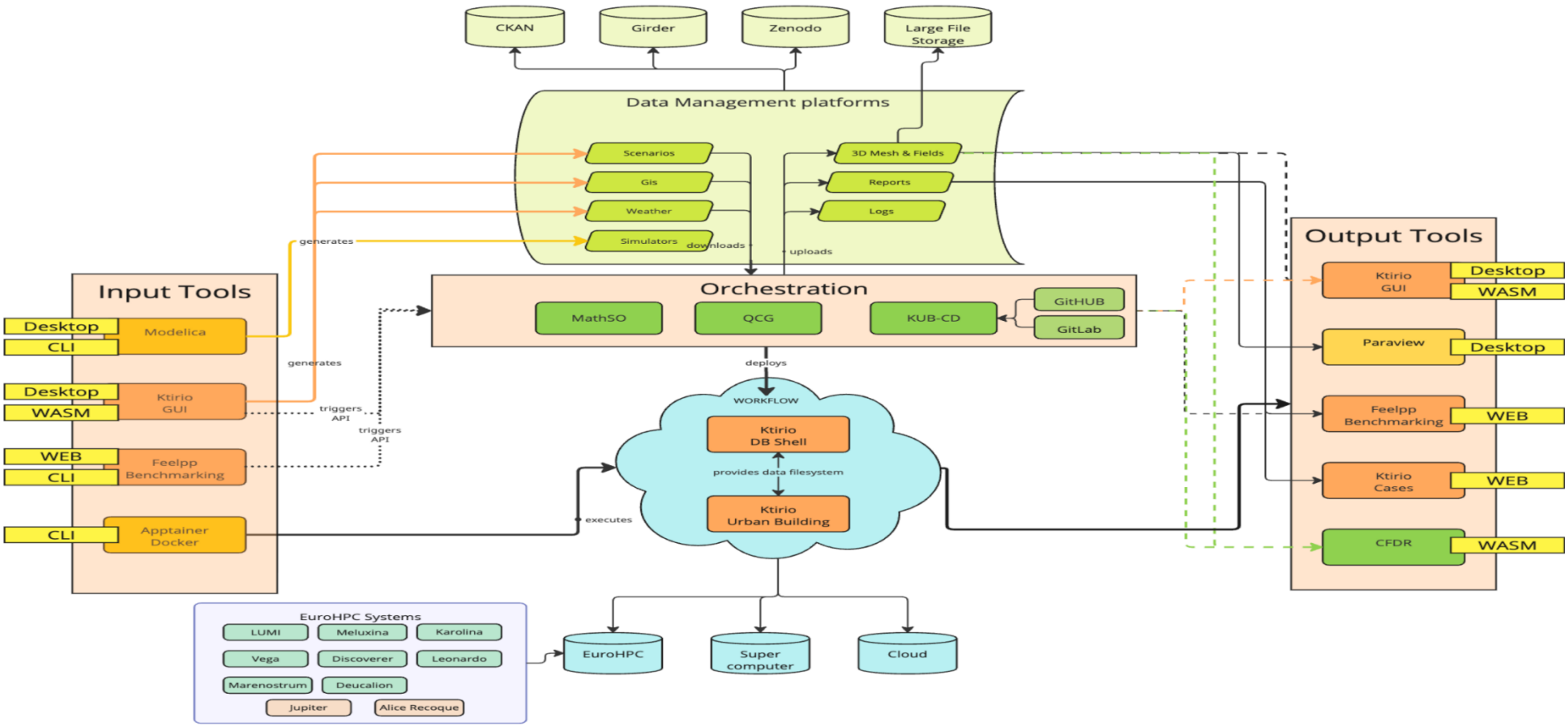


Figure 19: Urban Buildings CI/CD Framework

5.3.3 Development Roadmap

The development roadmap consolidates the items that remain to be delivered before the end of the project and clarifies milestones, expected deliverables, indicators, and risks. The sequence below expresses some next steps for the roadmap in terms of project months (M36, M40).

Milestones and timeline

By **M36**, regular regression tests will be instituted as a standing activity so that validated results remain reproducible as the software and the execution environments evolve. These tests will be scheduled and executed automatically and will cover representative production scenarios, datasets, and architecture variants.

By **M36**, identity and access management based on Single Sign-On (SSO) will be enabled in the GUI. This capability will be provided through Keycloak, either via a dedicated instance operated by the partner UNISTRA or by relying on the Keycloak instance made available within the HiDALGO2 consortium. The objective is to remove predefined accounts and to rely on authenticated, auditable user identities for front-end access and orchestration triggers.

By **M40**, image scanning with systematic Software Bill of Materials (SBOM) generation and attestation will be integrated into all release jobs. Promotion to the stable channel will be gated by the successful completion of these checks and of the integration tests on a reference cluster.

By **M40**, dashboards will be updated to expose cost and performance indicators for benchmarking, with an intermediate update targeted by M36. The M40 update also covers the refresh of all deployments and maintenance playbooks in order to align versions, credentials, and retention policies across sites.

By the end of the project (**M48**), the platform will support automatic deployment of new workflows that are derived from user-owned resources: users will provide their own repositories and credentials, and the platform will deploy these workflows on supercomputers under the user's allocations. Once the GUI is open to external users, production runs will not use the partner UNISTRA's accounts. Instead, the environment will be cloned or duplicated on demand to support the user's access to EuroHPC resources, subject to site AAI and acceptable-use policies.

Expected deliverables.

At project close, the platform is expected to provide: a hardened security baseline with SBOM generation and signed container images; dedicated EuroHPC runners for early validation of changes; improved Apptainer and WebAssembly packaging; automated provenance reporting for every run; SSO-based access via Keycloak; an extended

catalogue of Benchmarking-as-a-Service scenarios; and, critically, user-scoped workflow deployment whereby user repositories and credentials are used to drive jobs under user allocations on EuroHPC systems.

Risks and mitigations

The risk register focuses on operational constraints and heterogeneity. Access to EuroHPC systems and queue pressure will be mitigated by distributing tests across sites and by using off-peak windows. Quotas on compute time and storage will be handled through quota-aware scheduling and through selective retention or compaction of artifacts. Secrets and credentials will be protected by OpenID Connect where possible and by a rotation policy with least-privilege tokens otherwise. Divergence across MPI stacks and compilers will be addressed by pinning toolchains inside container images and by isolating site-specific modules in launch scripts. Architectural diversity in CPUs will be managed through a matrix of builds and per-site image variants with runtime checks; where relevant, additional variants will be prepared for accelerator nodes. IAM integration will be staged to ensure that user-owned credentials and repositories are supported without exposing the partner UNISTRA's resources.

Aggregation and reporting

The benchmarking layer will aggregate indicators over time, and the same infrastructure will schedule the regular regression tests described above. In combination with automated deployment and unified orchestration via kub-cd, this reporting layer will ensure that the platform remains auditable, portable, and ready for external evaluation at all times.

5.4 CI/CD for Renewable Energy Sources

The Renewable Energy Sources is using a CI/CD framework based on Gitlab and Gitlab runner for efficient development and deployment. Code changes trigger builds, tests, and deployment on local HPC cluster. RES can be executed via QCG portal on a HPC machine, results are then stored in CKAN and visualized inside the portal. The roadmap focuses on delivering recipes for building, deploying, and benchmarking RES containers across EuroHPC-JU sites.

5.4.1 Design

The design and components of RES CI/CD are as following:

- **Workflow Orchestration (WFO)**: QCG Portal
- **Code Repository**: GitLab (PSNC-hosted), HiDALGO2 repository
- **CI/CD System**:
  - **CI**: Multistage CI pipeline executed on PSNC local GitLab Runners.
  - **CD**: Recipes to build and deploy on local HPC cluster. Recipes to build the container with a target specific build and run a script collection (planned). Delivery as a docker to EuroHPC-JU machines (planned).
- **Data Management**:
  - **Simulation Data and results**: HiDALGO2 CKAN.
  - **Build recipes:** GitLab (PSNC-hosted), HiDALGO2 repository (planned)
  - **Source Code & Performance Data**: HiDALGO2 GitLab.
- **Visualization Tools**: Proprietary and CFDR (planned).

The ultimate goal is to make the RES pilot easily deployable and executable with containerized version of the code.

5.4.2 Implementation

The development and deployment of RES is based on Gitlab and Gitlab runner framework running locally at PSNC infrastructure. The workflow consists of compilation, testing, and validation the changes in the code in local HPC environment. The details of implementation are given in Deliverable D2.5. To facilitate using RES pilot, it has been integrated with the QCG Portal, which provides details such as job orchestration, post-processing, and results visualization. Refer to Section 3.2.2 for details.

5.4.3 Development Roadmap

The next step is to apply CI/CD workflow to EuroHPC sites. The current deployment is based on compiling the source code, and fetching from the Gitlab repository on the

target machine. Supporting many different HPC sites with many different versions of programming environment is challenging, therefore providing a containerized version of the RES application is required. Once ready, the CI/CD will account EuroHPC-JU sites. For each scenario and target machine, a recipe to build containerized version of RES will be provided, along with tests and benchmarks.

Additional challenge is to obey the regime of domain decomposition during the compilation. Therefore, the RES runner is equipped with tool to preconfigure domain decomposition and required hardware components based on the provided details of the problem, such as mesh size, its resolution or timestamp. Next goal is to add this pre-configuration as an additional step during building of a container.

Roadmap in details:

CI/CD Enhancement

- Develop RES as a container
  - Provide a container version of RES. **[M42]**
  - Create recipes for deployment and benchmarking **[M44]**
  - Provide domain pre configuration as a step during container build **[M46]**

QCG-Portal Integration

- Add support for CFDR visualization **[M42]**
- Extend support to additional EuroHPC clusters **[M46]**

Testing

- Validate pre-configuration mechanism based on provided problem/mesh **[M46]**
- Validate deployment and runtime on different HPC machines. **[M47]**

5.5 CI/CD for Wildfires

5.5.1 Design

For the CI stage, the recommendations from HiDALGO2's technical work package, WP2, have been adopted using GitLab to ensure traceability of the ongoing development.

In terms of the deployment process, different strategies have been selected depending on the model. For the WRF-SFIRE model the approach has been to use a solution based on EasyBuild[17] and EESSI[15] . Following the recommendations from the EPICURE team, custom EasyBuild recipes have been used for each of the HPCs and a joint solution using EESSI is being analysed.

For another model in the Wildfires use case, the FSE model, an alternate approach has been chosen. The model is first dockerised and then wrapped in an Apptainer container with a custom library available at Meluxina (next steps will explore other HPCs).

5.5.2 Implementation

The implementation has gone through several stages according to the state and needs of the project. As described in D2.5 [1] , the first stage was exclusively reliant on GitLab[21] taking full advantage of versions, branches and tags.

By using GitLab, some mechanisms were available in terms 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 installation packages to production and the subsequent release of the software in production.

During the next stage of development, a new approach based on EasyBuild[18] was taken. On the recommendation of the Discoverer HPC technical team, support from the EPICURE project was requested to analyse a deployment method that ensures replicability and stability. The support was granted and both teams managed to develop EasyBuild recipes for each of the HPCs for which the EPICURE team offered assistance and the HiDALGO2 team had resources: Discoverer, Meluxina, LUMI, Karolina and Vega.

Along with the EasyBuild[17] recipes, it has been analysed and tested the viability of using EESSI[18] for CD tasks. As of now, the installation of the applications by adapting the available EasyBuild recipes has been successfully tested on Karolina and to some degree on MareNostrum5 and Deucalion. The next steps in regards of the use of EESSI will try to test the possibility of submitting the recipe to EESSI central repository so they would be available on any HPC in which EESSI is installed and also, to assess the possibility of it being used within a container in the HiDALGO2 dashboard.

As explained above, Docker-Apptainer has been used to run the FSE model in Meluxina.

5.5.3 Development Roadmap

CI/CD Enhancement:

- Consolidate the deployment of WRF-SFIRE with EasyBuild. Evaluate use of standalone EasyBuild recipes vs EESSI. **[M42]**
- Collaborate with EESSI to include WRF-SFIRE in its central repository. **[M42]**
- Consolidate the deployment of FSE using Apptainer in other HPCs. **[M40]**

MathSO Integration:

- Test initial integration in one HPC. **[M36]**
- Extend support to other HPCs after initial test. **[M42]**
- Create tutorials and documentation for end users. **[M44]**

5.6 CI/CD for Material Transport in Water

The **Material Transport in Water (MTW)** pilot is developed by FAU to model complex three-way coupled flow problems using high-performance computing (HPC) resources. MTW builds upon the waLBerla multi-physics simulation framework and leverages its CI/CD strategies to ensure maintainable, reproducible, and scalable software development and deployment across diverse HPC infrastructures.

5.6.1 Design

The MTW pilot is implemented as a fork of the main waLBerla[13] repository. The waLBerla framework provides the parallel data structures and simulation routines used as the backend for MTW. The numerical model is described using FAU's code generation framework for Lattice Boltzmann simulations, lbmpy. Detailed model descriptions and pilot implementation aspects are documented in Deliverables D5.4, D5.6, and D5.7.

Integration with HiDALGO2 development services follows the general environment design illustrated in Figure 21. The relevant service components include:

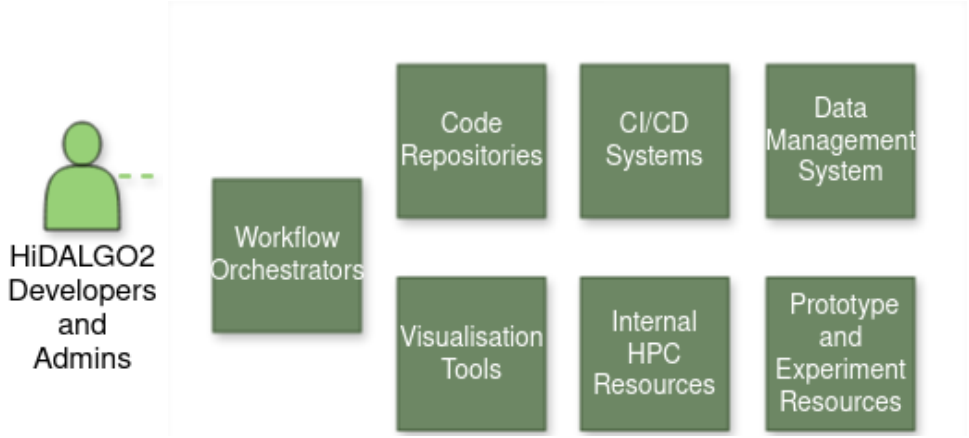


Figure 20: Cutout of Error! Reference source not found. **highlighting Development Services**

- **Workflow Orchestration (WFO)**: MathSO
- **Code Repository**: GitLab (FAU-hosted)
- **CI/CD System**:
  - **CI**: Multistage CI pipeline executed on FAU local GitLab Runners.
  - **Deployment**: Open-Source Codebase on HiDALGO2 GitLab Server with target specific build- and run script collection.
  - **CD**: Spack and EasyBuild package and deployment to EESSI, E4S, etc. in progress
- **Data Management**:

  - **Binaries or Simulation Data**: HiDALGO2 CKAN.
  - **Source Code & Performance Data**: HiDALGO2 GitLab.
- **Visualization Tools**: ParaView and CFDR (planned).

This design enables a seamless connection between code development, version control, automated builds, and user-facing deployment. Goal is to ensure that the MTW pilot is easily deployable and executable for both experienced HPC developers and new users through the HiDALGO2 platform.

5.6.2 Implementation

The MTW development process is based on multistage CI pipelines integrated within waLBerla and GitLab. Each code update triggers automated builds using multiple compilers (GNU, LLVM, Intel) and runs regression tests (red path of Figure 22). These are executed within Docker containers via CMake and CTest. The runtime environments for these containers are managed with Spack build caches, stored in a FAU local GitLab OCI registry. Stable development versions are merged to the project default branch where it is automatically mirrored to the HiDALGO2 repository. Merge requests to the default branch trigger a more sophisticated CI pipeline that includes tests for older compiler versions, various variations of build parameters as well as a FAU locally hosted Continuous Benchmarking (CB) (green path of Figure 22). Detailed information about this CB pipeline can be found in the article by Alt et al.[12] .

To ease the execution on various compute clusters and improve reproducibility, FAU developed a **research** Data Management (**RDM**) infrastructure which was introduced in in Deliverable D2.5[1] . This system abstracts the build and run process through systematically organized shell scripts, allowing users to execute simulations directly on HPC clusters. (Blue path of Figure 22.) Aided by this abstraction, the MTW pilot has been integrated into HiDALGO2's MathSO Portal (described in more detail in section 3.1), enabling web-based orchestration and monitoring of simulation executions for users new to HPC systems. (Yellow path of Figure 22.)

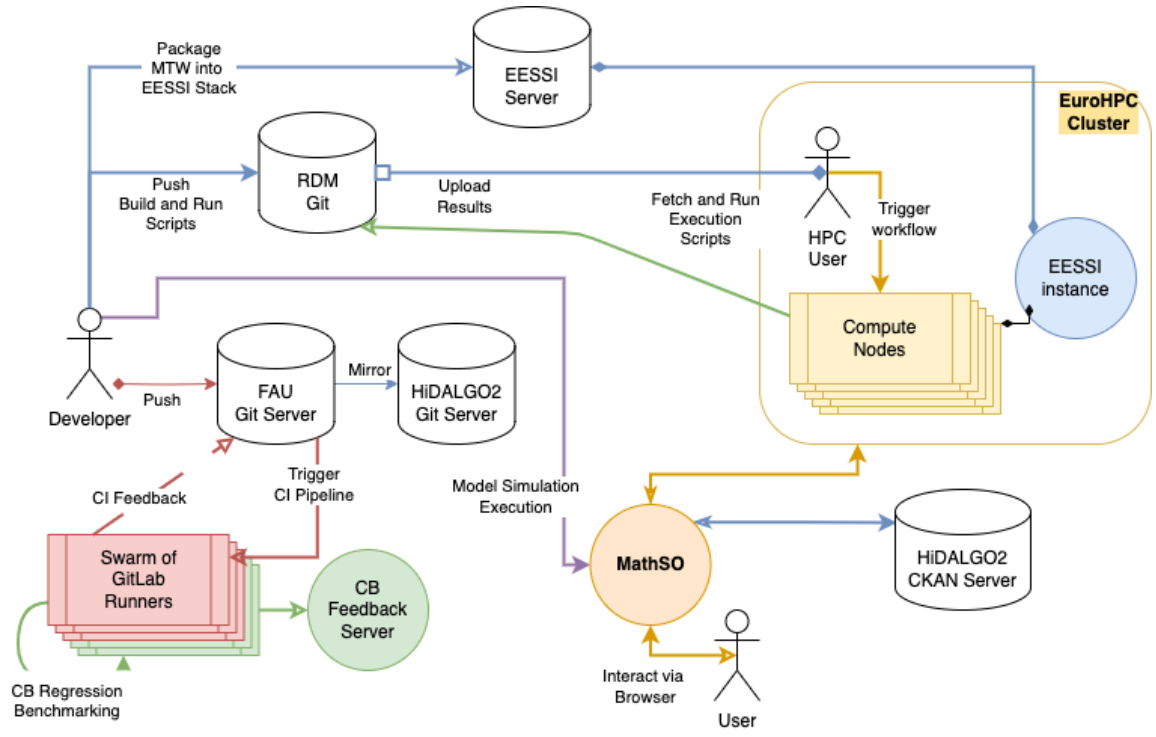


Figure 21: MTW CI/CD/CB/WFO Architecture

The workflow orchestration (WFO) for the MTW pilot can be accessed from the HiDALGO2 dashboard[9], as shown in Figure 23. MathSO provides a graphical front end for managing and executing HPC simulations. **Applications** in MathSO are abstractions of simulation workflows, designed by the pilot developers to provide a guided and reproducible execution of the underlying simulation code. To use MTW, a user first creates an **Instance** of the respective **Application** (Figure 24).

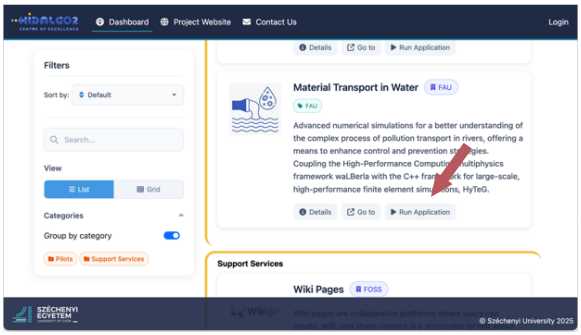


Figure 22: MTW WFO link on HiDALGO2 Dashboard

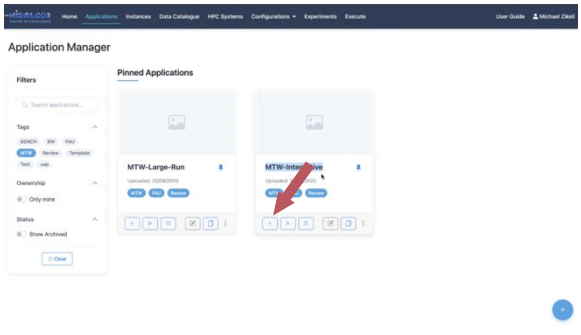


Figure 23: MathSO Application Page

[9] https://dashboard.hidalgo2.eu/

Each **Instance** (Figure 25) represents a reusable simulation configuration that can be executed on any computing cluster registered within MathSO. Once configured, the instance can be deployed (Figure 26) to a target system, where the job is submitted and executed. The results of past executions are accessible through the **Experiments** (Figure 27) page, where users can monitor the status of running **Experiments** and review metadata, execution logs, and output data of past **Experiments**, as detailed in Figure 28. In the upcoming project period, this functionality will be extended to include a CFDR (D4.6, D4.7) visualization of simulation results directly from the **Experiments** page.

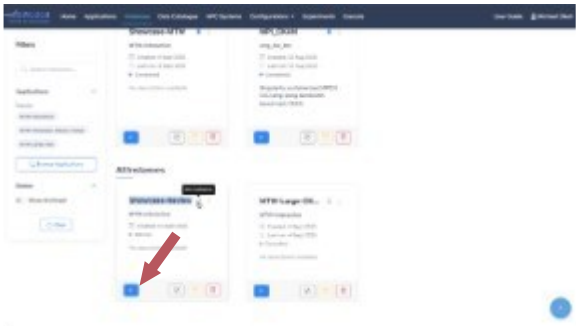


Figure 24: MathSO Instances page

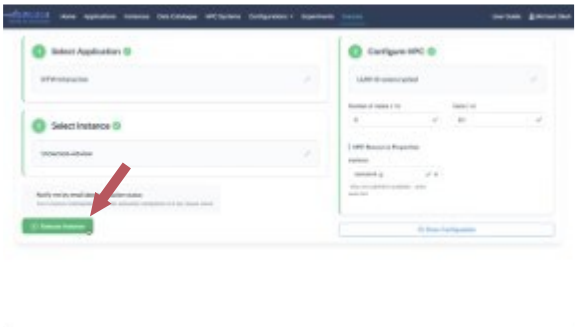


Figure 25: MathSO Execute page

A **video tutorial** demonstrating the orchestration of MTW runs using MathSO is also available on YouTube[10]. This might be more comfortable for the reader, as the images in this section needed to be kept to a thumbnail size.

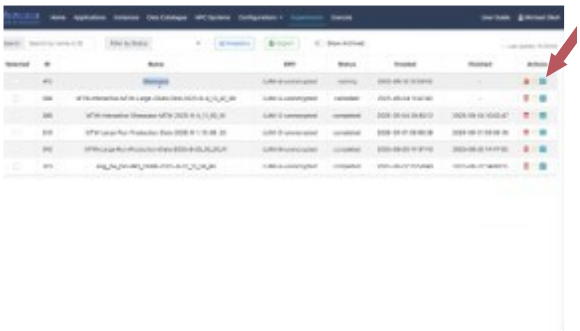


Figure 26: MathSO Experiments page

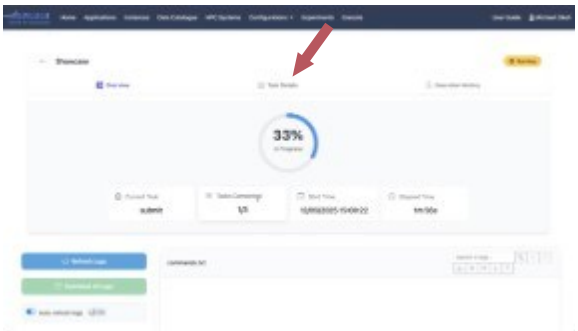


Figure 27: Experiment Status window

[10] https://youtu.be/rowCuXBwzb8

The orchestration process for MTW follows three sequential stages: *Pre-processing*, *Execution*, and *Post-processing* (Figure 29). During the *Pre-processing* stage, MathSO retrieves the required MTW binary from the selected target system, prepares the runtime environment, and adjusts the simulation parameters that were changed by users during the **Instance** configuration. In the *Execution* stage, MathSO automatically generates and submits the corresponding job script to the chosen HPC cluster. Finally, the *post-processing* stage involves querying the metadata associated with the submitted job, collecting and compressing the simulation output, and uploading the resulting data packages to the HiDALGO2 CKAN repository.

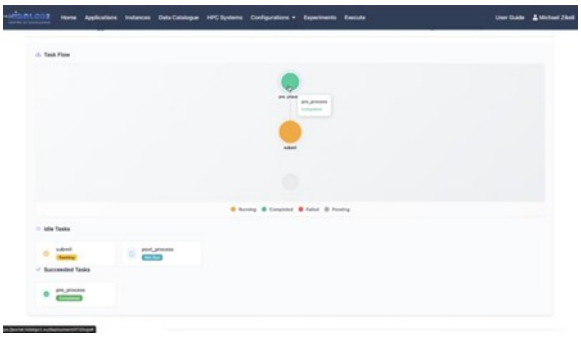


Figure 28: Experiment execution graph

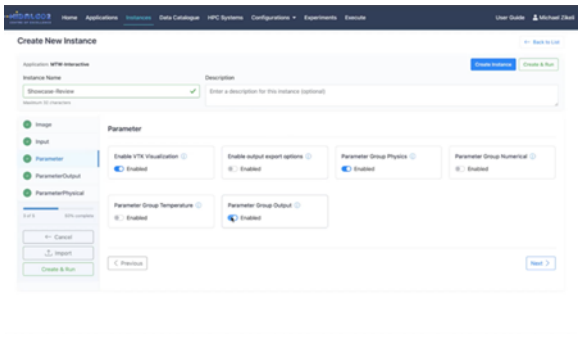


Figure 29: Instance Configuration

This level of automation and user convenience is achieved by modelling and maintaining the MTW pilot as a MathSO **Application**. The modelled abstraction of MTW includes simulation-specific parameter files that are stored and shared via CKAN. These files are fully operational yet remain editable through dynamic field objects in MathSO, allowing users to adjust parameters without directly modifying the file contents (Figure 30: Instance Configuration). The execution routine corresponds to the run script from FAU's RDM[1] infrastructure, adapted to the structure and syntax of MathSO. Post-processing functionalities are also defined within the model and can be activated or deactivated by the user as needed. Currently, integrating an application into MathSO requires providing the corresponding binary for each supported target system (such as LUMI, MareNostrum, or Vega) to ensure compatibility across different HPC environments.

HPC clusters heavily differ in hardware and software environments, which makes the reuse of binaries impossible. The process of manually creating and providing binaries to MathSO is of course a bottleneck in the deployment process. To address this issue, we currently aim to package MTW as a deployable software module, using HPC pack-

age managers such as Spack[16] and EasyBuild[17] . Packaging MTW facilitates deployment of waLBerla via the European Environment for Scientific Software Installations (EESSI), which offers pre-built scientific software stacks across EuroHPC systems. We collaborate with the EESSI project to include MTW in its repository[11], which will be updated to a production-ready release.

5.6.2.1 Development Roadmap

CI/CD Enhancement:

- Develop MTW as a packaged software module.
  - Use of Spack vs EasyBuild. **[M42]**
  - Create installation recipes. **[M44]**
- Collaborate with EESSI/ E4S[12] for pre-built distribution across EuroHPC clusters. **[potentially HiDALGO3, if funded]**

MathSO Integration:

- Extend support to additional EuroHPC clusters (beyond LUMI). **[M46]**
- Add browser-based visualization and post-processing. **[M39]**
- Create tutorials and documentation for end users. **[M48]**

User Experience and Testing:

- Validate deployment and runtime on diverse architectures (MareNostrum, Vega, etc.). **[M47]**
- Collect user feedback to simplify configuration interfaces. **[M48]**
- Optimize parameter handling and data output routines. **[M48]**

[11] EESSI package of MTW available under /cvmfs/dev.eessi.io/mtw/
[12] https://github.com/xsdk-project

5.7 Overall Development Roadmap for HiDALGO2 CI/CD

HiDALGO2 will continue its efforts in strengthening its component integration across the different software types, with effort given on maintaining flexibility in the choice of intermediate tools chosen to help deploy our CI/CD pipelines. This includes integration and deployment pipelines for both services and pilot codes.

For the **Services CI/CD system**, we will start efforts to migrate existing scripts, recipes, Ansible playbooks, and development deployment pipelines to the recently set up HiDALGO2 GitLab instance. This will replace earlier efforts that retained automation scripts and playbooks, local to the partner hosting the services. So PSNC hosted services had their scripts, and secrets such as administrator credentials, securely in local infrastructure, and the same case was with USTUTT hosted services. In the next year, the plan is to investigate options (a commercial Vault application or GitLab's internal secret management) to host these deployment scripts and automation pipelines in the common HiDALGO2 GitLab, but in such a way that secrets and credentials are stored and retrieved by these automation pipelines in a secure manner, without compromising any of PSNC or USTUTT's internal security.

If it turns out, that this has more harm than benefit, the efforts will be dropped, or reduced, in such a way, that only publicly safe automation scripts and playbooks are hosted on the HiDALGO2 GitLab, instead of full CI/CD pipelines with credential management and automated deployment of these services to our VMs. In that case, future staff members will still be able to use these scripts for faster and more reliable deployment, but credentials and other sensitive steps will still be done manually.

A reminder again, that in this deliverable, we have treated HiDALGO2's **Pilots CI/CD System** as part of, and connecting to **CI/CD for EuroHPC Systems**. We have worked with CASTEIL2 to deliver and deploy our Pilot Codes to a number of EuroHPC machines. In the final year of HiDALGO2, it is foreseen with confidence that all HiDALGO2 pilot codes will have an automated integration, build, delivery and deployment to any desired EuroHPC system. We will continue to work with future pan-European efforts such as CASTIEL3 or any dedicated EuroHPC CI/CD call, in case there is a need to modify or adapt our current CI/CD methodologies to interface with requirements or tools/solutions from the EuroHPC side.

6 Conclusion

This deliverable reports on the advancements in HiDALGO2's technical operating environment and tool development since the last deliverable of this series, D2.5, and as the project nears the end of its penultimate year, it also looks ahead and plans for further growth and stability, so that we can support our developers and future users in HiDALGO2's final year, and beyond.

HiDALGO2 steadily increased the provisioning of storage and compute to support development activities in other work packages, for the purposes of simulation data transfer, processing, prototypal AI experimentation, ensemble scenarios and the initial phases of uncertainty quantification. As these activities gather steam next year, the technical infrastructure will increase as required. In particular, in-house GPU resources would greatly help the speed of development of AI surrogate models, and efforts to acquire these are under investigation.
Further cloud resources were also provisioned, to extend support to the HiDALGO2 dashboard, and for pilot specific services, like Urban Buildings Ktirio GUI application. The project has a whole moved its central code repository from Bitbucket to GitLab, ensuring better compatibility with external EuroHPC and CASTIEL2 code repositories.

Both of HiDALGO2's workflow orchestrators sped up on their development, with many new features introduced, along with major framework updates, security and stability updates, and most importantly for HiDALGO2, a concentrated effort to integrate the Pilot codes with the orchestrators, deploying and executing simulations on EuroHPC systems, including a workflow resulting directly from the HiDALGO2 dashboard. This has been detailed in section 3 and section 5 (especially 5.1 and 5.6) of this deliverable, and will be further elaborated upon in the upcoming deliverable D2.9 (specifically on the HiDALGO2 dashboard).
The SEMS energy management suite, introduced in the second half of the project, also saw substantial development in this reporting period. Starting with the opening of the exascale EuroHPC system, Jupyter, to the EuroHPC community, two of the major components of the SEMS suite will be installed and available for use by HiDALGO2 pilots, as well as other interested simulation codes, to monitor and make their runs more efficient.

As the project progresses, HiDALGO2 pilot codes have increased the extent of their CI/CD coverage to a wider reach of EuroHPC systems, supporting both of the official CASTIEL2 preferred methods of deployment; firstly container-based runner deployments, through tools such as GitHub Actions and GitLab CI[21] , and secondly, through EasyBuild[17] recipes distributed through the EESSI-CernVMFS[18] system. In the next year, all of the HiDALGO2 pilots are expected to support one, or both, methods to deployments, and by summing up together, of supporting all architectures and partitions available across EuroHPC systems.

References

[1] HiDALGO2, "D2.5, Infrastructure Provisioning, Workflow Orchestration and Component Integration (M24)", HiDALGO2 report, 2024. DOI:10.13140/RG.2.2.10604.68489

[2] HiDALGO2, "D2.4, Infrastructure Provisioning, Workflow Orchestration and Component Integration - revised", HiDALGO2 report, 2024. DOI:10.13140/RG.2.2.36417.1648

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

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

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

[6] HiDALGO2, "D5.7, Implementation Report on Pilot Applications (M30)", HiDALGO2 report, 2025. DOI:10.13140/RG.2.2.21832.02567

[7] HiDALGO2, "D4.3, Advances in HPDA and AI for Global Challenges (M16)", HiDALGO2 report, 2024. DOI:10.13140/RG.2.2.10065.95848

[8] HiDALGO2, "D4.4, Advances in HPDA and AI for Global Challenges(M34)", HIDALGO2 report, 2025. DOI:10.13140/RG.2.2.17442.90568

[9] HiDALGO2, "D2.8, Dashboard and Services (M28)", HiDALGO2 report, 2025. DOI:10.13140/RG.2.2.36072.17929

[10] HiDALGO2, "D5.4, Research Advancements for the Pilots", HiDALGO2 report, 2024. DOI:10.13140/RG.2.2.23080.79368

[11] HiDALGO2, "D5.6, Implementation Report on Pilot Applications", HiDALGO2 report, Jul. 2024. DOI:10.13140/RG.2.2.12011.55849.

[12] Alt, C. et al. (2024) 'A continuous benchmarking infrastructure for high-performance computing applications', International Journal of Parallel, Emergent and Distributed Systems, 39(4), pp. 501–523. DOI:10.1080/17445760.2024.2360190.

[13] Chair for System Simulation, "waLBerla (widely applicable Lattice Boltzmann from Erlangen)". Zenodo, Aug. 07, 2025. DOI:10.5281/zenodo.16760078.

[14] Chair for System Simulation (Friedrich-Alexander-Universität Erlangen-Nürnberg), "lbmpy". Zenodo, Oct. 23, 2025. DOI: 10.5281/zenodo.17522190.

[15] Dröge et al. (2022) 'EESSI: A cross-platform ready-to-use optimised scientific software stack", Software: Practice and Experience, 53(1), pp. 176-210. DOI: 10.1002/spe.3075

[16] Gamblin, T. *et al.* (2015) *The Spack package manager: bringing order to HPC software chaos*. *Proc. SC '15*, ACM. DOI:10.1145/2807591.2807623.

[17] Hoste, K., Timmerman, J. & Georges, A. (2012) EasyBuild: building software with ease. Proc. SC'12 Companion, IEEE/ACM. DOI:10.1109/SC.Companion.2012.81.
[18] EESSI-CernVM fs system, https://www.eessi.io/, [retrieved 10.11.2025]
[19] Jupyter Notebooks, https://jupyter.org/, [retrieved 16.11.2025]
[20] Feelpp Benchmarking Suite, https://github.com/feelpp/benchmarking, [retrieved 16.11.2025]
[21] GitLab CI, https://docs.gitlab.com/development/cicd/, [retrieved 16.11.2025]
[22] GitHub Actions, https://github.com/features/actions, [retrieved 16.11.2025]
[23] CMake, https://cmake.org/, [retrieved 16.11.2025]
[24] CTest, https://cmake.org/cmake/help/latest/manual/ctest.1.html, [retrieved 16.11.2025]
[25] Docker, https://www.docker.com/, [retrieved 16.11.2025]
[26] LLVM, https://llvm.org/, [retrieved 16.11.2025]
[27] GCC, https://gcc.gnu.org/, [retrieved 16.11.2025]
[28] Francesco Antici, Andrea Bartolini, Zeynep Kiziltan, Ozalp Babaoglu, and Yuetsu Kodama. 2024. MCBound: An Online Framework to Characterize and Classify Memory/Compute-bound HPC Jobs. In Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '24). IEEE Press, Article 56, 1–15. DOI:10.1109/SC41406.2024.00062
[29] S. Cho and R. Melhem, "Corollaries to Amdahl's Law for Energy," in IEEE Computer Architecture Letters, vol. 7, no. 1, pp. 25-28, Jan. 2008, DOI:10.1109/L-CA.2007.18.
[30] WRF-SFIRE, https://github.com/openwfm/WRF-SFIRE, [retrieved 16.11.2025]
[31] EPICURE, https://epicure-hpc.eu/, [retrieved 16.11.2025]
[32] SLURM, https://slurm.schedmd.com, [retrieved 16.11.2025]
[33] OpenFOAM, https://www.openfoam.com/, [retrieved 16.11.2025]
[34] PBS/Pro, https://altair.de/resource/altair-pbs-professional-for-hpe, [retrieved 20.11.2025]

Annexes

Annex I: Infrastructure and Services

Table 5: Supporting Services

	Service	Description	Domain Link
1	Website	Project Website	https://hidalgo2.eu
2	Dashboard	Workflow Orchestrator	https://dashboard.hidalgo2.eu
3	Keycloak	Identity Management	https://idm.hidalgo2.eu
4	Zabbix	User Forums	https://monitoring.hidalgo2.eu
5	AskBot	User Forums	https://askbot.hidalgo2.eu
6	Zammad	User Support	https://ticket.hidalgo2.eu
7	Wiki	Knowledge Management	https://wiki.hidalgo2.eu
8	Open Project	Project Management	https://project.hidalgo2.eu
9	Moodle	Learning Platform	https://moodle.hidalgo2.eu
10	HedgeDoc	Collaborative Notes	https://hdoc.hidalgo2.eu

Table 6: Development and Compute Services

	Service	Description	Domain Link
1	Altair HPC	In-house HPC System	https://pcss.plcloud.pl
2	Eagle HPC	In-house HPC System	https://pcss.plcloud.pl
3	Prototype	Compute Cluster	https://prototype.hidalgo2.eu
4	JupyterHub	Jupyter Notebooks	https://jupyter.hidalgo2.eu
5	Keycloak	Identity Management	https://idm.hidalgo2.eu
6	CKAN	Data Catalogue Storage	https://ckan.hidalgo2.eu
7	GitLab	Git Repository	https://git.hidalgo2.eu
8	Hadoop HDFS	Big Data Storage System	https://stream.hidalgo2.eu
9	Ktirio GUI	Simulation Configurator	https://ktirio.hidalgo2.eu
10	MathSO Portal	Workflow Orchestrator	https://portal.hidalgo2.eu
11	QCG Portal	Workflow Orchestrator	https://qcg.hidalgo2.eu
12	SEMS Suite	Energy Management	N/A
13	NIFI	Data Transfer System	N/A

Table 7: EuroHPC System Access

	Urban Air Pollution (SZE)	Urban Building Modelling (UNIS-TRA)	Renewa-ble Energy Sources (PSNC)	Wildfires (MTG)	Material Transport in Water (FAU)	Bench-marking (ICCS)
LUMI-C	Awarded	Awarded	In Process	Awarded	Awarded	Awarded
LUMI-G	Awarded	Awarded	N/A	N/A	Awarded	Awarded
Vega CPU	Awarded	Awarded	Awarded	Awarded	In Process	Awarded
Vega GPU	Awarded	Awarded	N/A	Awarded	N/A	Awarded
Karolina -CPU	Awarded	Awarded	Awarded	Awarded	N/A	Awarded
Karolina - GPU	Awarded	No Access	No Access	No Access	No Access	Awarded
Meluxina CPU	Awarded	Awarded	In Process	Awarded	Awarded	Awarded
Meluxina GPU	Awarded	Awarded	N/A	Awarded	Awarded	Awarded
Discoverer CPU	Awarded	Awarded	Awarded	Awarded	Awarded	Awarded
Leonardo DCGP	Awarded	Awarded	In Process	Awarded	N/A	In Process
Leonardo Booster	Awarded	Awarded	N/A	N/A	Awarded	Awarded
MareNostrum5 - GPP	Awarded	N/A	Awarded	N/A	Awarded	Awarded
MareNostrum5 - ACC	Awarded	N/A	N/A	Awarded	Awarded	Awarded
Deucalion x86	Awarded	N/A	Awarded	No Access	N/A	Awarded
Deucalion Arm	Awarded	Awarded	N/A	No Access	Awarded	Awarded
Deucalion GPU	Awarded	N/A	N/A	No Access	N/A	Awarded
Jupyter	Planned	Planned	Planned	Planned	Planned	Planned

Table 8: Service Matrix I

[INSTITUTE]	[NAME]	[DEV/PROD]	[CPU]	[RAM]	[DISK]	[OS]	[IP]	[DOMAIN]	[Status]	Integration	Integration	[ADMIN]	[PURPOSE]
PSNC	CKAN	PROD	8	16GB	40GB +2TB	Ubuntu 22.04	62.3.171.19	ckan.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	Data platform
PSNC	Streaming	PROD	4	8GB	40GB + 2TB	Ubuntu 22.04	62.3.171.226	stream.hidalgo2.eu	Running	In Progress	Running	Piotr Dzierżak	Streaming services
HLRS	Streaming	DEV							Dev - Ready	Dev - Ready	Dev - Ready		
HLRS	Wiki	PROD	2	4GB	20GB + 20 GB	Ubuntu 22.04	141.58.0.233	wiki.hidalgo2.eu	Running	Running	Running	Sameer Haroon	Wiki (Wiki-js)
PSNC	Wiki	DEV							Dev - Ready	Dev - Ready	Dev - Ready		
PSNC	IDM	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.171.16	idm.hidalgo2.eu	Running	Not require	Running	Piotr Dzierżak	IDM / Keycloak
HLRS	IDM	DEV	2	4GB	20+20 GB	Ubuntu 22.04	141.58.0.233		Running	Not require	Dev - Ready		
PSNC	Askbot	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.171.180	askbot.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	Q&Aoriented Internet forums
HLRS	Askbot	DEV					141.58.0.233		Dev - Ready	Dev - Ready	Dev - Ready		
PSNC	Monitor	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.170.54	monitor.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	Zabbix monitoring
HLRS	Monitor	DEV					141.58.0.233		Dev - Ready	Dev - Ready	Dev - Ready		
PSNC	Prototype0	PROD	8	16GB	40GB +2TB	Ubuntu 22.04	62.3.170.126	prototype.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	Training access node
PSNC	Prototype1	PROD	32	32GB	40GB	Ubuntu 22.04			Running	Running	Running	Piotr Dzierżak	Training node
PSNC	Prototype2	PROD	32	32GB	40GB	Ubuntu 22.04			Running	Running	Running	Piotr Dzierżak	Training node
PSNC	Prototype3	PROD	32	32GB	40GB	Ubuntu 22.04			Running	Running	Running	Piotr Dzierżak	Training node
PSNC	Prototype4	PROD	32	32GB	40GB	Ubuntu 22.04			Running	Running	Running	Piotr Dzierżak	Training node
PSNC	Support ticket	DEV											
HLRS	Support ticket	PROD	2	4GB	15+15GB	Ubuntu 22.04	141.58.0.233	ticket.hidalgo2.eu	Running	Running	Running	Sameer Haroon	Ticketing Service
PSNC	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	Moodle	PROD	2	4GB			141.58.0.233	moodle.hidalgo2.eu	Running	Running	Running	Sameer, Maksym	Moodle, Learning platform
PSNC	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	JupyterLab compute-1	PROD	16	32GB	40GB	Ubuntu 22.04			Running	Not require	Running	Piotr Dzierżak	JupyterLab compute node
PSNC	JupyterLab compute-2	PROD	16	32GB	40GB	Ubuntu 22.04			Running	Not require	Running	Piotr Dzierżak	JupyterHub compute node
PSNC	Website	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.170.227	www.hidalgo2.eu	Running	Not require	Running	John Kitsos	Website
PSNC	MathSO Portal	PROD	8	16GB	40GB	Ubuntu 22.04	62.3.170.136	portal.hidalgo2.eu	Running	Running	Running	Akos Kovacs	Portal (MathSO)
PSNC	QCG Portal	PROD	30	30GB	8GB	OKD 3.x/OKD 4.x		qcg.hidalgo2.eu	In Progress	In Progress	In Progress	Bartosz Bosak	Portal (QCG)
SZE	MathSO Portal	DEV	4	4GB	40GB	Ubuntu 22.04		testportal.mathso-srv.sze.hu	Dev - Ready	Dev - Ready	Dev - Ready		
PSNC	Dashboard	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.170.210	prunus-210.man.poznan.pl				Piotr Dzierżak	HiDALGO2 Dashboard
SZE	Dashboard	DEV							Dev - Ready	Dev - Ready	Dev - Ready		
HLRS	Open Project	PROD	2	4GB	20 + 20 GB	Ubuntu 22.04	141.58.0.83	project.hidalgo2.eu	Running	Not possibl	Running	Sameer Haroon	Project Management Software
HLRS	Ktirio GUI	PROD	4	8GB	60GB	Ubuntu 22.04	141.58.0.233	ktirio.hidalgo2.eu	Running	In Progress	In Progress		
PSNC	GitLab	PROD	8	16GB	60GB	Ubuntu 22.04		git.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	Code Repository
PSNC	Hedgedoc	PROD	2	4GB	40GB	Ubuntu 22.04		hdoc.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	Collaborative Notes

Table 9: Service Matrix II: Data

[NAME]	[CPU]	[RAM]	[DISK]	[OS]	[IP PRIV]	[IP PUB]	[DOMAIN]	[OS STATUS]	[SOFT STATUS]	Integration with Zabbix	[ADMINISTRATOR]	[PURPOSE]
hidalgo2-ambari	8	16GB	40GB + 500GB	Ubuntu 16.04	192.168.150.12	62.3.170.106	prunus-106.man.poznan.pl	Running		Running	Yosu	HUE: HDFS Browser
hidalgo2-nn-1	8	16GB	40GB + 500GB	Ubuntu 16.04	192.168.150.95	62.3.171.42	sophora-42.man.poznan.pl	Running		Running	Yosu	HDFS/YARN NameNode/ Resource Manager
hidalgo2-nn-2	8	8GB	40GB + 500GB	Ubuntu 16.04	192.168.150.166	62.3.171.103	sophora-103.man.poznan.pl	Running		Running	Yosu	HDFS Secondary NameNode
hidalgo2-dn-1	16	32GB	40GB + 10TB	Ubuntu 16.04	192.168.150.57	62.3.170.209	prunus-209.man.poznan.pl	Running		Running	Yosu	HDFS/YARN DataNode/ NodeManager 1
hidalgo2-dn-2	16	32GB	40GB + 10TB	Ubuntu 16.04	192.168.150.127	62.3.171.109	sophora-109.man.poznan.pl	Running		Running	Yosu	HDFS/YARN DataNode/ NodeManager 2
hidalgo2-dn-3	16	32GB	40GB + 10TB	Ubuntu 16.04	192.168.150.5	62.3.170.210	prunus-210.man.poznan.pl	Running		Running	Yosu	HDFS/YARN DataNode/ NodeManager 3
hidalgo2-dn-4	16	32GB	40GB + 10TB	Ubuntu 16.04	192.168.150.87	150.254.224.167	onoclea-167.man.poznan.pl	Running		Running	Yosu	HDFS/YARN DataNode/ NodeManager 4
...												
hidalgo2-nifi	16	32GB	40GB + 500GB	Ubuntu 16.04	192.168.150.93	62.3.171.105	sophora-105.man.poznan.pl	Running		Running	Yosu	NIFI Master server
hidalgo2-nifi-2	16	32GB	40GB + 500GB	Ubuntu 16.04	192.168.150.100	150.254.224.172	onoclea-172.man.poznan.pl	Running		Running	Yosu	NIFI Slave server

Annex II: MathSO Portal Implementation Status and Roadmap

Pilot integration

The MathSO portal supports several applications, especially pilot applications from consortium partners. While applications from the Urban Air Project pilot have been most mature, applications from Material Transport in Water and Urban Buildings have also been showcased. Please note that some applications are directly accessible through the Dashboard (like Ktirio), and do not access HPC systems through the portal. Some of the most important applications are as follows:

- **Material Transport in Water**
  The pilot application has been integrated using a precompiled binary that can be deployed the same way as a singularity container. Some predefined parameter files have been created, which can be optionally edited within the instance configurator, making this application highly versatile. See section 5.6.2 for further details.
- **Ktirio Urban Building**
  The first integration of the Urban Building pilot application uses a pre-generated city bundle from the ones uploaded to the HiDALGO2 CKAN. The bundle includes GIS metadata, LoD-0 meshes, weather information, building models and descriptions, and simulation configuration files.
- **RedSim Air comfort**
  The CPU or GPU solver can be configured to calculate and assess the air comfort parameters for cities based on weather conditions, like the wind direction.
- **Map downloader**
  The map area can be marked with points of interests, as shown in Figure 30. This input can be used to download a building geometry from OpenStreetMap for 3D urban modelling and CFD mesh generation.
- **UAP-Snappy**
  The OpenFOAM+snappyHexMesh based workflow creates a 3D CFD mesh from geometries downloaded by the map downloader.
- **UAP-FOAM**
  The OpenFOAM based implementation of the SZE Urban Air Pollution model supports several features within the portal, including coupling ECMWF weather data and pollution data from traffic simulation, point and plane sampling. Output can be visualized with images for 2D plane cuts, or 3D representations in CFDR.
- **Xyst**
  The basic integration of the highly scalable solver application uses an input mesh, that can be selected from a data repository, and a configuration file, which

can be edited within an edit window.

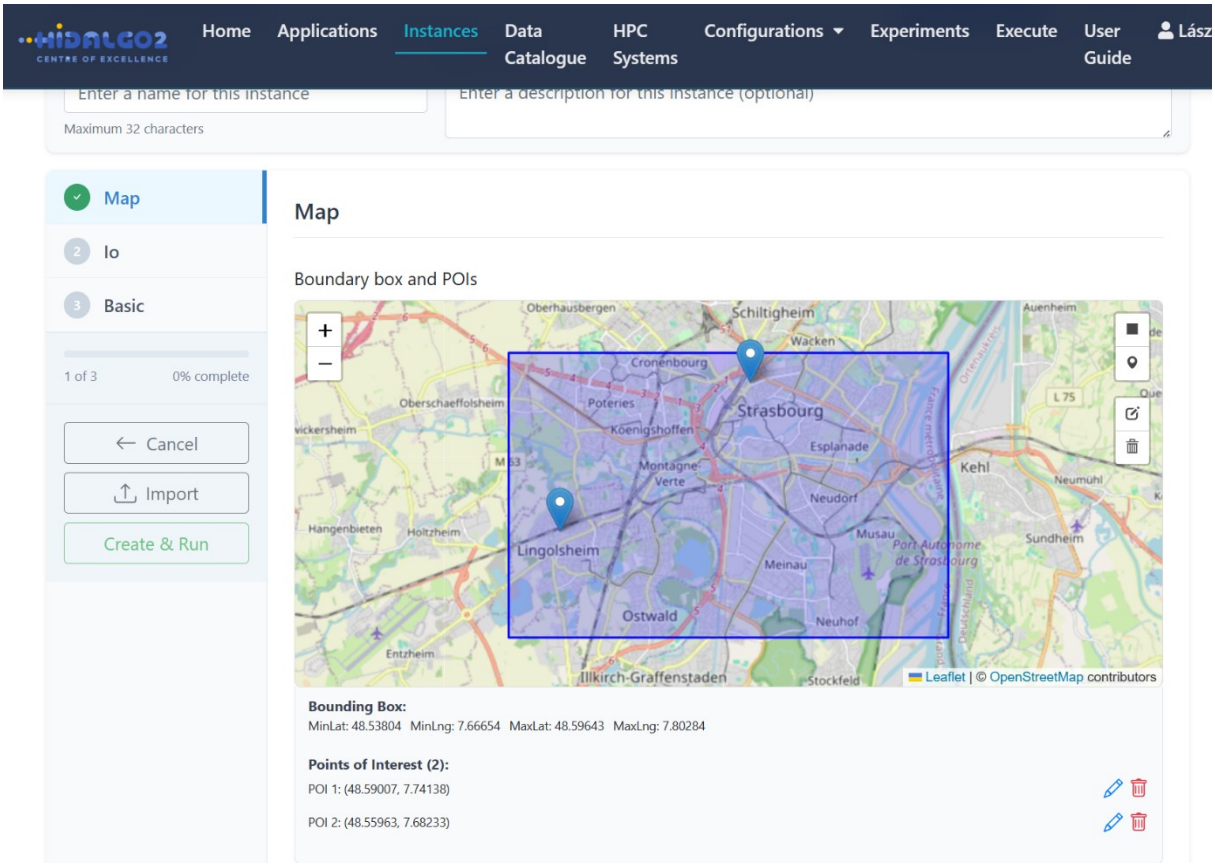


Figure 30: Map Downloader application for OpenStreetMap geometry download. Aside from simulation area, points of interests can be marked for sampling or point pollution source.

Feature Implementation

The following main features have been implemented into the MathSO portal in the last report period.

- Landing page, as described and shown in the design section.
- Application editor, with additional user management system for appropriate access.
- Overall portal redesign, as shown in the design section.
- Object manager (application, instance, repository, HPC) with appropriate functions for each type of object and user's role:
    - Application:
        - User: Create instance, View instances, Run application, Pin
        - Developer: Add, Edit, Duplicate, Archive, Delete

  - Instance: Add, Run, Edit, Archive, Delete, Pin (for both user and developer)
  - Repositories: Add, View, Edit, Delete, Transfer, Pin
  - HPC Systems: Add, View, Edit, Delete, System Information, Partition Usage Information, Connection testing, 2-factor-authentication, Add SSH repository, Terminal access
- Additional features for ease of use:
  - SSH connection keep-alive management to avoid rapid reconnection.
  - SSH key management, including key generation of various type.
  - API access for portal access from the command line or client tool.
  - Log repository for accessing simulation logs when not connected to the HPC system
  - 2-panel data management (Total Commander-like). See section on CI/CD.
- Simulation status report
  - Overview of currently running or completed tasks.
  - Details on task execution per node template.
  - Summary of job resource usage.
  - Execution history with time stamps.

The SZE team has also worked on documentation, training and automated testing and benchmarking of the portal system. The final versions will be available in the Moodle.

Development Roadmap

M36 Release:

- Direct interactive visualization and control with **Digital Twin Support** including external data retrieval and assimilation.

M42 Release:

- **Parameter Sweep Function,** as described in D2.5. Work in progress.
- While some applications already support CFDR, **Generalized Visualisation Support** is work in progress.
- The **User Notification System,** as described in D2.5. Work in progress.
- Support for **all EuroHPC systems**. Work in progress.
- **Wildfire Pilot Integration.** Work in progress.

M48 Release:

- **Improved DMS support** including Hadoop and HDFS access for HPDA workflows. Work in progress.
- **Intelligent Job Submission,** as described in D2.5. Work not yet started.

Annex III: QCG Portal Implementation Status and Roadmap

The new templating mechanism and architectural changes described in the previous section were essential for the realisation of HiDALGO2 goals, and significantly enhanced practical usability and versatility of QCG-Portal. However, they required major updates to the implementation of some modules of the system. In practice, the updates focused on significant improvements of existing functionalities rather than the introduction of completely new ones, allowing the overall feature set to stay aligned with the roadmap.

So far, bilateral collaboration with WP5 has focused on RES, making it the first Pilot to benefit from the development of a comprehensive platform based on the new templating mechanism. Nevertheless, this mechanism is highly generic, and the introduced functionality can be adapted and applied to other Pilots during the remaining phase of the project, within scope and in accordance with their specific requirements.

Feature Implementation

Access to EuroHPC computing infrastructure

Realisation

M21 – 1st release: realised

Current implementation details

- SSH-based QCG agent enables non-intrusive connection to EuroHPC machines
- User credentials are kept in the QCG-Portal backend
- Realised integration with Altair and Lumi

Roadmap

The SSH-based QCG agent detailed in the deliverable D2.5 enables non-intrusive integration of QCG-Portal with the remote computing resources exclusively over SSH protocol. This universal mechanism allows for seamless integration with all EuroHPC resources. However, one of its current limitations lays in the security mechanism, which is based on user credentials required to connect the orchestrator to resources being kept in the QCG-Backend. This simple mechanism ultimately may need to be replaced by more comprehensive solution, in particular based on SSH Certificates, as described in D2.5, once this mechanism became widely adopted by resource providers.

Divergences from the plan

Through the generic resource-integration mechanism and in line with the preferences of the RES pilot, access has been provided to selected resources, including Altair and LUMI, while other resources have not yet been integrated, but will be added in the near future.

Workflows and cyclic tasks

Realisation (workflows):

M27– 1st release (Airflow): realised

M30 – PoC release (built-in workflow engine): realised

M42 – 1st release (built-in wf engine): planned

Realisation (cyclic tasks):

M24 – PoC release (Airflow): realised

M30 – PoC release (built-in workflow engine): realised

M42 – 1st release (built-in workflow engine): planned

Current implementation details

- Basic support for workflow execution through AirFlow
- Built-in Python-based mechanism for definition and execution of workflows, with basic support for execution of loops and execution of cyclic tasks.

Roadmap

To meet the specific expectations of the Pilot owners, the originally proposed mechanism for workflow execution via the Airflow service has been extended with a built-in workflow orchestration system within QCG-Portal. This new mechanism enables the flexible construction of workflows directly in Python, including support for user-defined loops and cyclic tasks. In the remaining part of the project, the development will concentrate on improvements to this built-in mechanism unless specific needs for the advanced features provided by the Airflow-based orchestration are reported by the Pilots.

Divergences from the plan

Due to the shift in focus motivated by the expectation of Pilots that led to the temporary suspension of the Airflow-based orchestration development, the originally planned release of this orchestration at M30 was not completed. Instead, two new releases of the built-in workflow orchestration mechanism were scheduled: a pre-release delivered at M33 and the first official release planned for M36. Since support for cyclic tasks is now provided through this new orchestration mechanism, the corresponding releases for

cyclic-task functionality have been aligned with the workflow orchestration release schedule.

Visualisation and Monitoring

Realisation (Visualisation):

M24 – PoC release (CFDR): not realised

M36 – 1st release (CFDR): planned

M30 – PoC release (built-in visualization mechanisms): realised

M42 – 1st release (built-in visualization mechanisms + CFDR): planned

Realisation (Monitoring):

M30 – 1st release (Grafana): realised

M30 – PoC release (built-in monitoring mechanisms): realised

M42 – 1st release (built-in monitoring mechanisms): planned

Current implementation details

- Support for monitoring of execution of workflows and individual tasks through the integration with Grafana framework.
- Built-in monitoring capabilities that allow to display monitored metrics in a consistent way directly in the QCG-Portal GUI.
- Built-in mechanisms for presentation of results, including basic visualisation, directly in the QCG-Portal GUI.
- Provided a new Application Interface Builder tool to streamline the process of definition of the layouts for custom monitoring and the results presentation views.
- Developed custom monitoring and results presentation views for RES Pilot.

Roadmap

At this stage of the project, the monitoring and visualisation layers are implemented through the newly introduced templating mechanism, which provides the foundational support for consistent output presentation. The project's next development steps are expected to concentrate on strengthening the support for visualisation and monitoring through the recently introduced templating mechanism. This mechanism enables enhanced monitoring views and uniform behaviour across interfaces, as demonstrated in the RES Pilot. The accompanying Application Interface Builder allows users to interactively design custom views based on a core set of predefined components. In the remaining phase of the project, both the mechanism and the component set are expected to be further expanded in line with Pilot requirements

Divergences from the plan

Taking into account the current development status of Pilots and the relatively low urgency of integrating the CFDR visualiser into the QCG-Portal, the planned integration has been postponed to the next project year. Instead, efforts have focused on enabling basic visualisation capabilities directly within the templates, including support for interactive charts, images, and maps.

HiDALGO2 Data Management Systems

Realisation:

M24 –1st release (CKAN): realised

M30 –1st release (Hadoop/HDFS): not realised

M42 – 1st release (Hadoop/HDFS): planned (conditional)

Current implementation details

- Support for downloading and uploading data from/to CKAN from workflow steps.
- Accessing CKAN with API Key.
- Automatic compressing and tagging data uploaded to CKAN.

Roadmap

At this stage, the orchestrator includes mechanisms for downloading and uploading data to CKAN. This functionality has been fully sufficient for the current phase of the project, meeting the present needs of the RES Pilot. The integration of the HiDALGO2 Data Transfer Tool has been postponed to the final year of the project.

Divergences from the plan

The integration with the HiDALGO2 Data Transfer Tool has not been finalised. From the perspective of the Pilots using QCG-Portal, the integration with the HiDALGO2 Data Transfer Tool was not a priority at this stage due to other ongoing key developments.

mUQSA

Realisation:

M27 2nd release: realised

M33 3rd release: realised

Current implementation details

The list of new features added to mUQSA includes:

- the addition of visualisation capabilities for UQ statistics of scalar QoIs,
- enhancements to Quasi–Monte Carlo sampling (including the selection of sampling rules, support for fixed seeds, and visualisation of confidence intervals for bootstrap analysis),
- presentation of higher-order Sobol' indices,
- introduction of a tabular mode for parameter specification, with the ability to import and export data from and to Excel, as well as to copy and paste values from the clipboard directly within the interface.

Roadmap

The development of mUQSA has progressed in accordance with the established roadmap. Several new functionalities, including those requested by Pilot owners, have been incorporated, enhancing the practical usability of the solution. One of the capabilities considered for further development is the ability to restart an executed campaign from the point at which it was stopped; however, its implementation will depend on project priorities and available resources.

Divergences from the plan

The increased interest in usage of mUQSA and collected feedback from HiDALGO2 partners and participants of HiDALGO2-CIRCE-SEAVEA VVUQ Hackathon, motivated us to carry out an additional, initially unplanned release at M27.

.

QCG Portal Overall Development Roadmap Chart

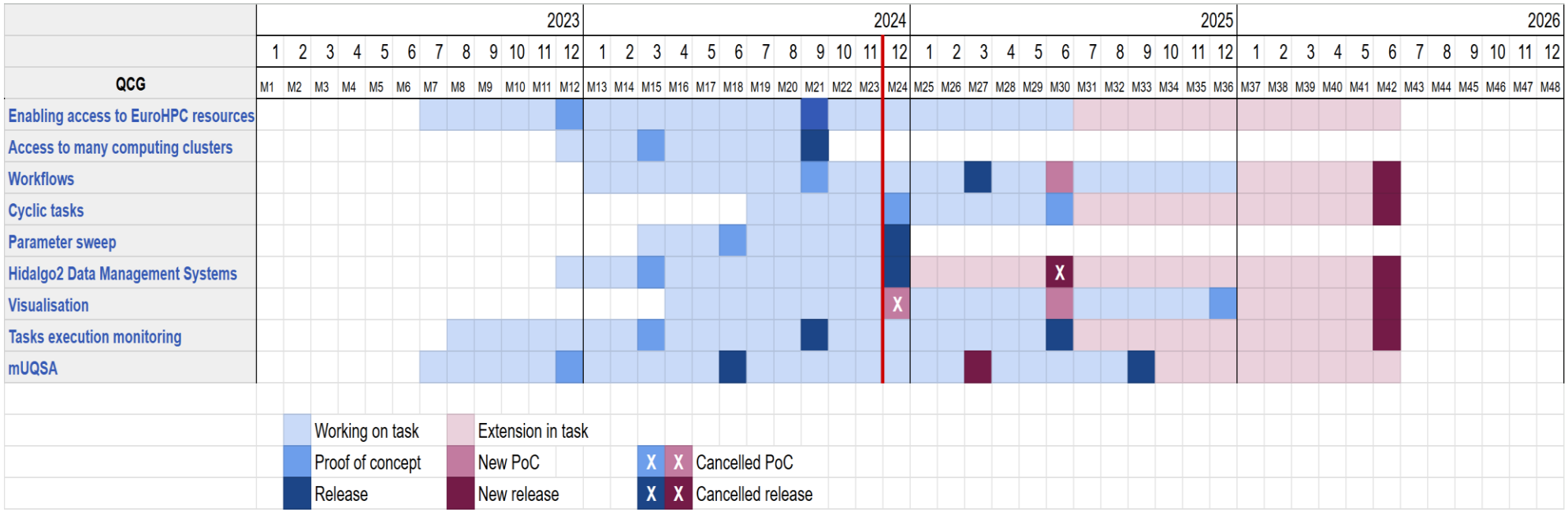


Figure 31: QCG Development Roadmap Gantt Chart

Annex IV: Energy Management Suite

Implementation details and results

Power and Performance Estimator

The implementation of the PPE tool has benefited from the results published in [28] . Although the development is mostly complete, some metrics required for its functionality are not yet available in the Eviden Argos Suite. This has resulted in a delay for its integration, which is now expected to be completed in the first half of 2026.

Figure 32: PPE with Random Forest model results over time

Figure 32 shows the results of implementing the work described in [28] . The implementation was carried out using two models: Random Forest (RF) and K-nearest Neighbors (KNN). The results presented here are based on our own validation. We are specifically showing the training results of the RF model, which was trained with data from the last 30 days and re-trained daily. As you can see, the average F1 macro score was 0.89.

The assessment of the KNN model, under the same conditions as the RF model, yielded an average F1 score of 0.85.

In **Error! Reference source not found.**, we present the training and inference times of the KNN model, with times displayed in different bars corresponding to the training durations used (15, 30, 45, and 60 days), referred to as Alpha. As shown, the worst-case scenario (60 days) averages 0.16 seconds.

In contrast, the RF model, being more computationally intensive, averaged 180 seconds for training and 0.03 seconds for inference in the worst-case scenario. All the presented results were run in a CPU AMD EPYC 7282 with 16 cores.

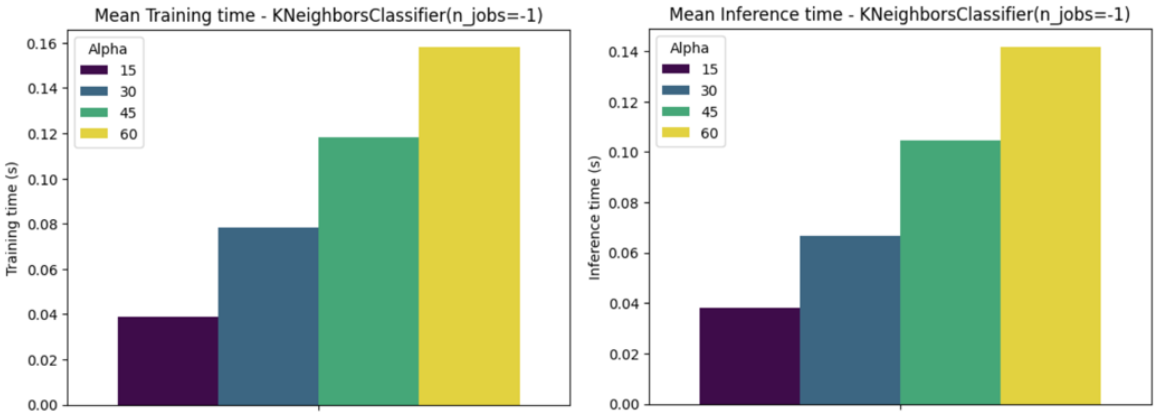


Figure 33: PPE training and inference times with KNN model

Job Energy and Performance Comparator

The development of JEPC was completed as anticipated and successfully integrated into Argos in version 2.1 as an API.

The main endpoints we developed for JEPC are the *ranking* and the *scalability*:

argos/analytics/v1/jepc/ranking/{cluster_name}/{jobid}/{metric}/{ep_tradeoff}

Where:

- cluster_name is the cluster where the job ran.
- jobid is the id of the job set to be analysed.
- metric is an extra argument to bring more comparison over jobs (e.g. number of nodes)
- ep_tradeof shows available values you can use when calling the ranking endpoint. When ranking, this value states how much the algorithm should consider energy and performance, e.g.:
  - "++energy" → 100% energy, 0% performance
  - "+energy" → 75% energy, 25% performance
  - "energy_performance" → 50% energy, 50% performance

- "+performance" → 25% energy, 75% performance
- "++performance" → 0% energy, 100% performance

The bigger the *ep_ratio* the better energy and performance the job was compared to the rest. A typical output is shown in Figure 34:


```
[
    {
      "id": 17832,
      "app_id": "jnam_29",
      "start_time": "2021-04-06T07:06:01",
      "end_time": "2021-04-06T07:09:14",
      "energy": 15039.380175,
      "nodes": 2560,
      "duration": 193.0,
      "performance_loss": 0.0,
      "energy_waste": 39.28,
      "ep_ratio": 1.21
    },
    {
      "id": 102413,
      "app_id": "jnam_29",
      "start_time": "2021-04-27T10:33:24",
      "end_time": "2021-04-27T10:37:09",
      "energy": 17629.151085,
      "nodes": 2560,
      "duration": 225.0,
      "performance_loss": 14.22,
      "energy_waste": 48.2,
      "ep_ratio": 1.04
    },
    {
      "id": 102434,
      "app_id": "jnam_29",
      "start_time": "2021-04-27T10:45:52",
      "end_time": "2021-04-27T10:49:47",
      "energy": 18437.736053,
      "nodes": 2560,
      "duration": 235.0,
      "performance_loss": 17.87,
      "energy_waste": 50.47,
      "ep_ratio": 0.99
    }
]
```


Figure 34: First Job Output of Energy and Performance Comparator

In the results of that set, the job with ID 17832 demonstrated the best trade-off between energy and performance (50-50). The *performance_loss* column indicates that this job was the fastest, with a duration that resulted in a performance loss of 0.0%. However, in terms of energy efficiency, it was not the best, as it had an energy waste of 39.28% due to utilizing more nodes for execution. Nonetheless, the overall trade-off suggests that the energy waste was justified to achieve better performance.

argos/analytics/v1/jepc/scalability/{cluster_name}/{jobid}

Where:

- cluster_name is the cluster where the job ran.
- jobid is the id of the job set to be analysed.

The output should look as shown in Figure 35.

```
{
    "parallel_part": 77.9346193913304,
    "max_speedup": 4.53198618113614,
    "speedups": {
      "50": 4,
      "75": 11,
      "90": 32,
      "95": 67,
      "99": 350
    }
}
```


Figure 35: Second Jo b Output of Energy and Performance Comparator

The output indicates that the estimated parallel region of the job set is 77%, with a maximum achievable speedup of 4.5x. Additionally, it specifies the number of nodes required to achieve the corresponding percentage of the maximum speed up. For instance, to attain 50% of the maximum speedup (2.25x), the user should utilize a minimum of 4 nodes. However, attaining any close to the maximum theorical speedup (e.g. 99%) the user would require 350 nodes, which greatly increases the resources and energy required.

Job Energy Predictor

The job energy predictor is currently in the early stages of development, with the focus on testing various models and approaches to enhance the final results.

At this point, we are developing two models: one aimed at predicting job duration and the other focused on job power. Once both models are completed, we plan to combine them to achieve accurate job energy predictions.

Job Energy Recommender

As planned, JER tool is not starting until **M42**.

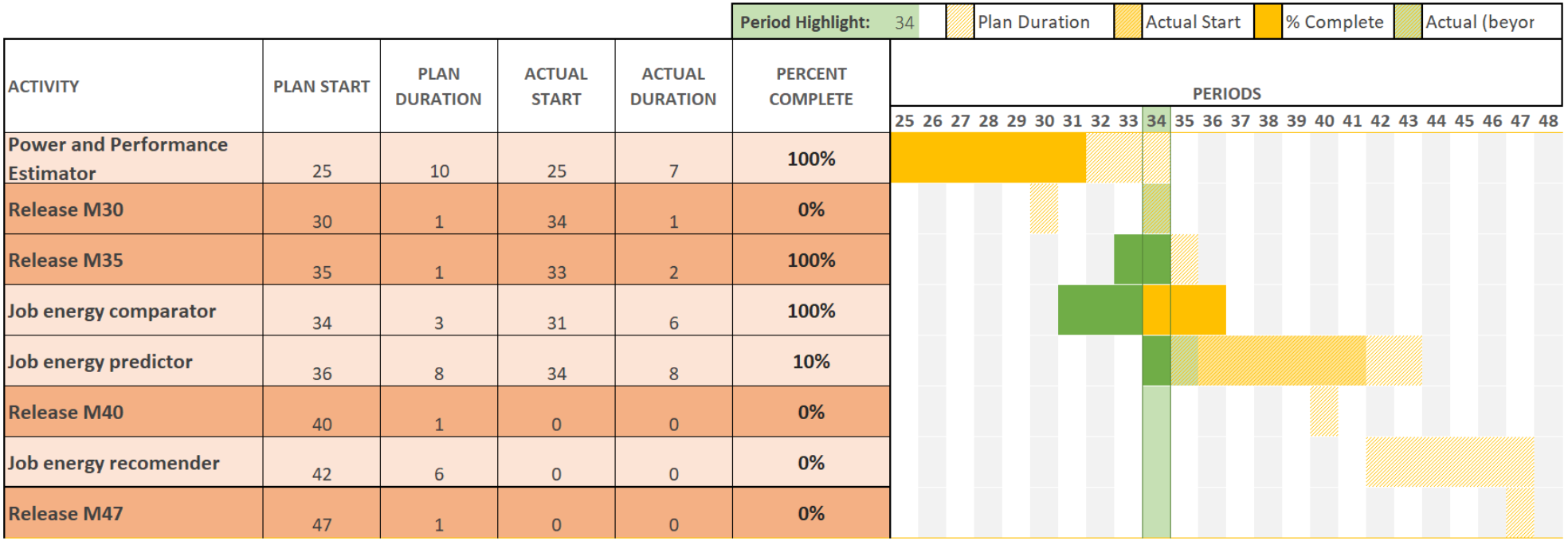
Development Roadmap and Gantt Chart

The **PPE** development was anticipated to take ten months, but it was completed in just seven months. However, the integration of the **PPE** has been put on hold due to the unavailability of required metrics in the Argos software. This has caused a delay in the planned release from **M30** until these metrics can be obtained.

In addition, we advanced the release **M35** to **M31** to facilitate the integration of **JEPC** into **Argos**. The development of **JEPC** was also expedited by three months, ultimately taking a total of six months to complete. This was three months longer than initially expected, primarily due to the stringent requirements for integration with Argos.

Lastly, the **Job Energy Predictor** commenced two months ahead of schedule and is projected to take eight months to complete, as originally planned.

Figure 36: Roadmap of energy monitoring and Optimization frameworkFigure 36 shows the detailed planning and development roadmap in the form of a Gantt chart



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Power and Performance Estimator	25	10	25	7	100%
Release M30	30	1	34	1	0%
Release M35	35	1	33	2	100%
Job energy comparator	34	3	31	6	100%
Job energy predictor	36	8	34	8	10%
Release M40	40	1	0	0	0%
Job energy recomender	42	6	0	0	0%
Release M47	47	1	0	0	0%

Figure 36: Roadmap of energy monitoring and Optimization framework