D2.4 Infrastructure Provisioning, Workflow Orchestration and Component Integration (M11)

HiDALGO2

CENTRE OF EXCELLENCE

Date: December 15, 2023

EuroHPC Joint Undertaking

Document Identification			
Status	Final	Due Date	30/11/2023
Version	1.6	Submission Date	15/12/2023

Related WP	WP2	Document Reference	D2.4
Related Deliverable(s)	D2.1, D2.7, D3.1. D4.1	Dissemination Level (*)	PU
Lead Participant	USTUTT	Lead Author	Haroon, Sameer (USTUTT)
Contributors	ATOS, PSNC, SZE	Reviewers	Konstantinos Nikas (ICCS)
			David Caballero (MTG)

Keywords:

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

Document Information

List of Contributors	
Name	**Partner**
Sameer Haroon	USTUTT
Davide Padeletti	USTUTT
Jesus Gorronogoitia	ATOS
Piotr Dzierzak	PSNC
Bartosz Bosak	PSNC
Piotr Kopta	PSNC
Tomasz Piontek	PSNC
Akos Kovacs	SZE

Document History			
Version	**Date**	**Change editors**	**Changes**
0.1	11/10/2023	USTUTT	Table of Contents
0.2	03/11/2023	ATOS (PSNC, SZE)	Version 0.1 of Section 3, Section 2, Description of Services, Structure for WFOs
0.3	09/11/2023	USTUTT	Version 0.1 of Section 1, 5, Introduction and initial CI/CD
0.4	15/11/2023	ATOS (SZE, PSNC)	Version 0.2 of Section 2, 3, Functional Specs of WFO
0.5	15/11/2023	USTUTT	Version 0.2 of Section 1, 5, Redraft after initial feedback
0.6	17/11/2023	SZE	Version 0.1 of Section 4 First draft of Dashboard requirements and plan
0.7	20/11/2023	SZE	Version 0.2 of Section 4 Redraft after feedback
0.8	21/11/2023	ATOS, PSNC, SZE	Version 0.3 of Section 2, 3, Development Roadmaps
0.9	22/11/2023	USTUTT, PSNC	Version 0.4 of Section 2 Integrating contributions and restructuring, text content rephrased after feedback
0.91	24/11/2023	USTUTT	Version 0.4 of Section 5 Diagrams for each tier remade and integrated
1.0	27/11/2023	USTUTT, ATOS, PSNC, SZE	All sections integrated in Version 1.0.
1.1	28/11/2023	USTUTT, ATOS	Texts revised and updated for multiple sections and references partially added and numbered in order, Annex's added
1.2	28/11/2023	USTUTT	Final update for internal review.
1.3	01/12/2023	USTUTT, ATOS	Redraft after feedback from internal reviewers
1.4	05/12/2023	ALL	Redraft after additional internal feedback

1.5	13/12/2023	USTUTT	Draft prepared for Quality Management
1.6	15/12/2023	USTUTT, PSNC	Quality control and final updates

Quality Control		
Role	**Who (Partner short name)**	**Approval Date**
Deliverable leader	Sameer Haroon (USTUTT)	15/12/2023
Quality manager (deputy)	Dennis Hoppe (USTUTT)	15/12/2023
Project Coordinator	Marcin Lawenda (PSNC)	15/12/2023

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
AAI	Authentication and authorisation infrastructure
API	Application Programming Interface
CI/CD	Continuous Integration / Continuous Delivery
CMS	Content Management System
CoE	Centre of Excellence
CPU	Central Processing Unit
Dev	Development
Devops	Development Operations
DMS	Data Management System
Dx.y	Deliverable number y belonging to WP x
Flops	Floating point operations
GB	Gigabyte
GSI	Grid Security Infrastructure
HPC	High Performance Computing
IDM	Identity Management
IP	Internet Protocol
JWT	JSON Web Token
N/A	Not applicable
Prod	Production
PFlops	PetaFlops
RAM	Random Access Memory
SSH	Secure Shell
TB	Terabyte
TFlops	TeraFlops
WFO	Workflow Orchestrator
WP	Work Package

Executive Summary

This deliverable provides an overview of the components that make up the core foundation of the HiDALGO2 ecosystem, and support the project's use cases in achieving their scientific, technical and environmental goals. It builds upon the work of previous deliverables in HiDALGO2, as well as of D2.1, "Requirements Analysis and Scenario Definition". It also collaborates with the development of D4.1, "Data Management and Coupling Technologies" and D3.1, "Scalability, Optimization and Co-Design Activities".

Here, HiDALGO2 outlines its technical infrastructure, consisting mainly of compute resources and web services, and layouts their distribution across the technical resources within the project, and the wider EuroHPC JU network. Individual components are broken down and described, and a map is developed of how they connect together to build up the high-level system context design. One of these components, the planned HiDALGO2 dashboard is given a look ahead, outlining its design guidelines, and laying the foundations for its development in D2.7, a future deliverable.

An in-depth description of the central components supporting the pilot use cases, the Workflow Orchestrators, MathSO and QCG, is given. A careful analysis of the general and specific requirements of the individual pilot simulations has been gathered from the partners, and a detailed development roadmap attempts to address these requirements, as well as introduce future features.

The deliverable rounds out the system design of the HiDALGO2 environment with a strategy and vision for integrating all of its constituent components. The design of three distinct layers of component integration, or CI/CD pipelines, are shown in architecture diagrams following the C4 diagramming system, providing a successively lower level of detail in the system design.

Finally, the roadmaps drawn out in the individual chapters give the direction the HiDALGO2 technical infrastructure and supporting development will follow, which will then be documented in future iterations of this deliverable, D2.6 and D2.7.

1. Introduction

1.1 Purpose of the document

The deliverable “D2.4 Infrastructure Provisioning, Workflow Orchestration and Component Integration (M11)” captures a detailed overview of the overall technical architecture of the HiDALGO2 project and provides a foundation with support for flexibility and growth of the project requirements and solutions, which will be continuously reviewed throughout the runtime of the project and documented in future iterations of this document (D2.5 in M24 and D2.6 in M35).

This deliverable includes a description of the initial infrastructure and services of HiDALGO2, including available hardware and software for pilots and web services, and reports on the development of the HiDALGO2 dashboard (Task T2.3), the Workflow Orchestrators (Task T2.4), and the Component Integration (Task T2.5) of HiDALGO2 software and services.

1.2 Relation to other project work

This document builds upon the motivation behind incorporating various services and strategies from deliverables in HiDALGO2, as well as the requirements collected in “D2.1 Requirements Analysis and Scenario Definition (M6)”. It also coordinates closely with “D4.1 Data Management and Coupling Technologies (M11)” and “D3.1 Scalability, Optimization and Co-Design Activities (M12)”. D2.4 also provides a platform and overall design for further detailed elaboration of the HiDALGO2 dashboard in the future deliverable “D2.7 HiDALGO2 Dashboard and Services (M14)”.

1.3 Structure of the document

Chapter 2 describes the hardware and software resources available to the project and the overall System Design.

Chapter 3 summarises the proposed features of the HiDALGO2 Dashboard and its planned development roadmap.

Chapter 4 introduces the two workflow orchestrators available to the project, including their architecture, design, and implementation.

Chapter 5 discusses HiDALGO2’s vision for achieving a consistent Component Integration, through Continuous Integration and Continuous Deployment (CI/CD) across its services and pilots.

Chapter 6 summarises and concludes this deliverable while giving an outlook about future objectives and challenges.

2. Infrastructure Provisioning

This chapter introduces HiDALGO2's methodology towards infrastructure provisioning. which comprises the underlying hardware (e.g., EuroHPC JU [2] supercomputers) and software (e.g., Jupyter Notebooks [3] that is required to execute HiDALGO2's pilots and Web services, which together make up the HiDALGO2 ecosystem.

This chapter is structured as follows: Section 2.1, Operational Environment, gathers the broad technical specifications of the current state of the ecosystem, Section 2.2 introduces a high-level system design of the environment, while Section 2.3, Development Roadmap, outlines the next objectives and necessary steps taken for the upcoming years.

2.1 Operational Environment

The main components comprising the HiDALGO2 technical environment are described here, with a detailed breakdown of their hardware and software specifications and how they fit together in the overall HiDALGO2 system architecture. Before delving into the environment, it is worthwhile to recap from experiences in HiDALGO2, and to give a motivation for why HiDALGO2 needs and maintains these services and systems.

2.1.1 Motivation

HiDALGO2 needs HPC and other compute infrastructure, to provide its pilots with a test bed to run, test, execute and benchmark their codes and newer features. This infrastructure is also helpful in testing out pilot integrations and workflows with other needed tools, like orchestrators and visualisers. It also provides ample opportunities to develop and test the CI/CD methodology of HiDALGO2, especially when it comes to transferring this methodology and experience to deploy pilot applications on EuroHPC systems.

The HiDALGO2 environment also includes a number of web services that support and complement the project's efforts. These are hosted on premise at USTUTT and PSNC, and range from data management to project management. Some services in the project, like the ticketing support system, and the user forums system, have their origins in HiDALGO2, and a detailed overview of the methodology incorporated for supporting its users is detailed in the deliverable D5.4 of HiDALGO2. Many of the HiDALGO2 services have distinct development and production environments, to make sure that installation and upgrade procedures are tested in a safe environment, before transferring over to the live system in use by users.

For completeness, a short recap of the fundamental requirements related to the provisioning of HiDALOG2's environment, as captured in the earlier deliverable D2.1, "Requirements Analysis and Scenario Definition", are provided and briefly addressed here.

REQ-HPII-001: Access to HPC infrastructure

- Dedicated Resources for Pilot Simulations as well as for Benchmarking are available internally in the project from PSNC, as well as externally through EuroHPC systems.

REQ-HPII-02: Platform Scalability

- Allow for scalability to the number of managed applications and the load/data that each one produces and consumes. Both PSNC and USTUTT commit to increasing resources, e.g. for Data Management.

REQ-HPII-003: Platform Robustness

- High levels of robustness against hardware and software failures, utilise redundancy, load-balancing, and checkpoints. This requirement is fulfilled for web services through the CI/CD system, as well as by using strategies like separating Development and Production environments.

Table 1 gives a listing of the services, and section 2.1.3 goes into more detail into the technical specifications of the services, as well as a description of the importance of the service to HiDALGO2, its developers and its users.

Table 1: Overview of Web Services

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

2.1.2 Supplementary infrastructure for computing

The following subsections detail the hardware and software resources allocated to the individual components making up the HiDALGO2 ecosystem. Using internal resources, like PSNC Eagle and the Prototype cluster described below, affords more freedom for the HiDALGO2 pilots to develop and test out their applications. It also gives the project the opportunity to test out its Workflow Orchestrators (detailed in Section 4), as well as its CI/CD pipelines.

PSNC Altair

The main HPC system at PSNC is the Huawei Cluster "Altair", which consists of more than 63,000 Intel Xeon Platinum 8268 processors, yielding a theoretical peak performance of about 5,9 PFlops. Each node is equipped with 192 or 386 GB RAM, and nodes are interconnected with InfiniBand EDR. There are 9 nodes outfitted with 8 GPU Nvidia V100 32GB cards, each offering 62,4 TFlops of peak performance.

PSNC Eagle

The HPC system Huawei Cluster "Eagle" consists of almost 33,000 CPUs Intel Xeon E5-2697 yielding a theoretical peak performance of about 1,4 PFlops. Each node is equipped either with 128 or 256 GB RAM memory and nodes are interconnected with InfiniBand FDR. There are 3 nodes outfitted with 2 GPU Nvidia V100 32GB cards each offering 15,6 TFlops of peak performance.

Additional Resources

- HPDA cluster designed for highly robust and efficient data analytics. It is composed of 24 nodes, equipped with Intel Xeon E5-2697 processors (345 cores in total) and 690 GB of memory RAM. There are main frameworks preinstalled and optimised for high performance data processing: Apache Spark and Hadoop.
- Cloud system – the system is composed of a significant number of compute nodes and managed through OpenStack. Both USTUTT and PSNC provide hardware resources for the allocation of the HiDALGO2 web services.
- Storage – Currently, the PSNC mass storage infrastructure consists of several types of systems, including mid-size disk arrays, high-performance disk arrays, software-defined disk servers, as well as specialised systems, including an efficient file server and SSD arrays. USTUTT will also be providing increasing amounts of mass storage to help with HiDALGO2 tasks like Uncertainty Quantification as the project progresses.

Prototype Cluster

Table 2: Prototype Cluster infrastructure specification

Name	Prototype Cluster		Prod[1]	PSNC	IP	62.3.170.126
Software	Slurm [4]		Dev[2]	N/A	IP	N/A
Hardware	CPU cores	8	RAM	16 GB	Hard Disk	40 GB +2 TB
Domain	http://prototype.hidalgo2.eu					
Description	The prototype HiDALGO2 cluster constitutes the training infrastructure for project participants. It is composed of one access node and four compute nodes (Figure 1). Slurm Workload Manager [4] software is installed on the nodes. Disk storage is shared across the access and compute nodes. It provides 2 TB of storage with the possibility of increasing capacity. Users can log in to the system using SSH keys. The access node communicates with compute nodes using private IP addresses. In the future it will be integrated and allowed access from within QCG and MathSO.					

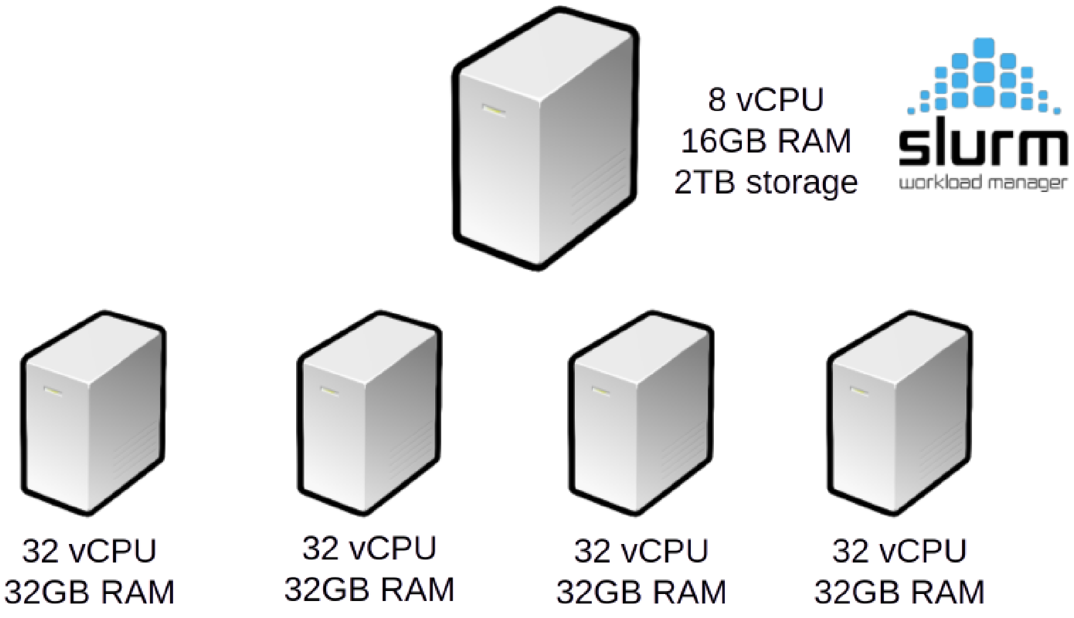


Figure 1: Prototype HiDALGO2 Cluster

[1] Prod = production environment
[2] Dev = development/staging environment

JupyterHub

Table 3: JupyterHub infrastructure specification

Name	JupyterHub		Prod	PSNC	IP	62.3.171.173
Software	Jupyter Notebook, JupyterHub, Slurm, Python3, TensorFlow		Dev	N/A	IP	N/A
Hardware	CPU cores	6	RAM	12 GB	Hard Disk	40 GB + 2 TB
Domain	https://jupyter.hidalgo2.eu					
Description	The HiDALGO2's JupyterHub service gives users access to computational environments and resources without burdening the users with installation and maintenance tasks. The infrastructure is composed of one head node and two compute nodes as shown in Figure 2. The JupyterHub software is integrated with the Slurm Workload Manager, which is installed on all cluster nodes. Disk storage is shared across the access node and compute nodes. It is 2 TB with the possibility of increasing the capacity. Users can log in to the service using HiDALGO2 IDM (Keycloak). New accounts are created after first log in to the JupyterHub.					

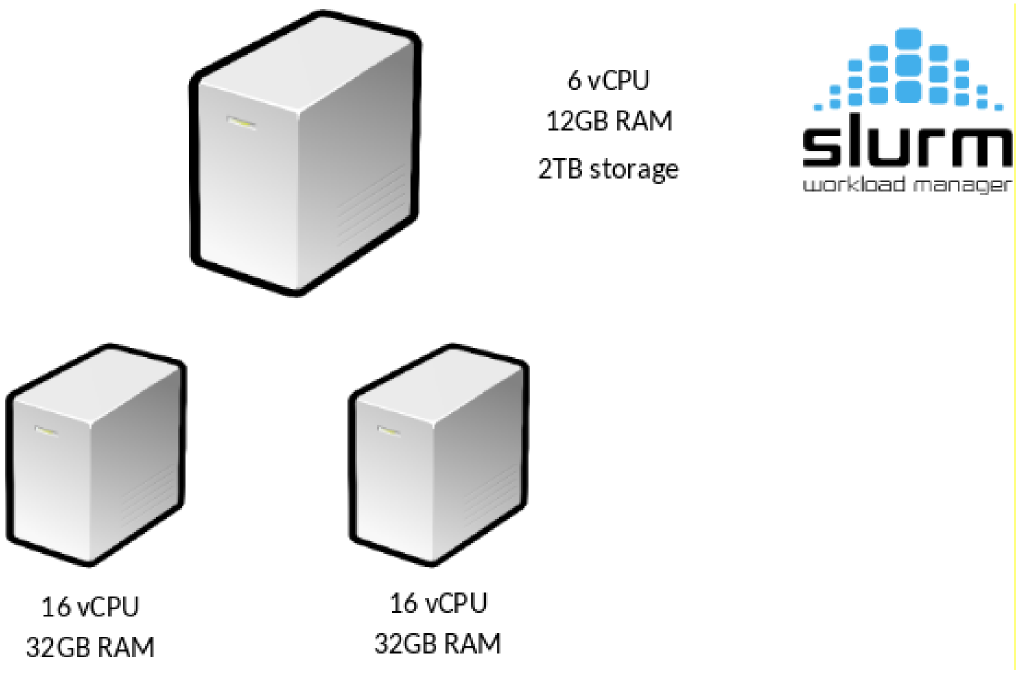


Figure 2: JupyterHub infrastructure

EuroHPC Sites

HiDALGO2's project members have applied for and gained access to several EuroHPC JU supercomputers, through various access schemes, such as the development and benchmarking schemes. These supercomputers include LUMI, Meluxina, Leonardo, Karolina, Discoverer and Vega. A detailed description and specification, along with references and other related links are provided by the EuroHPC JU on an overview webpage [5] . The current status of the actual system coverage of the pilot use cases is described in Section 2.3, and will be further elaborated in the deliverable D3.1

2.1.3 Infrastructure for Web Services

All following services, unless otherwise mentioned, are deployed on virtual machines that run the current version of Ubuntu [6] Linux (22.04) as their Operating System. Most services also have a development instance running to serve as a testbed for installing, upgrading and maintaining the service. Usually, the production and development environments are split up between the two infrastructure providers, PSNC and USTUTT. In case no development instance is needed, the term N/A (Not applicable) is used.

Table 4: Project Website infrastructure specification

Name	Website		Prod	PSNC	IP	62.3.170.227	
Software	Nginx [7] , PHP [8] , WordPress [9]		Dev	USTUTT	IP	To be allocated	
Hardware	CPU cores	4	RAM	8 GB	Hard Disk		40 GB
Domain	https://hidalgo2.eu						
Description	The main project website for HiDALGO2, where all updated information about the project can be found, including events, blog posts and deliverables and other documents. The website will also link to the HiDALGO2 customer services and the main Dashboard Portal.						

Table 5: CKAN infrastructure specification

Name	Data Management - CKAN		Prod	PSNC	IP	62.3.171.19	
Software	CKAN [5]		Dev	N/A	IP	N/A	
Hardware	CPU cores	8	RAM	16 GB	Hard Disk		2 TB
Domain	https://ckan.hidalgo2.eu						
Description	CKAN provides HiDALGO2 with a flexible Database for storage of various kinds of data, managing the input and output data from pilot simulations. CKAN						

	includes a stable metadata feature, to form catalogues of datasets, which allow for a simplified browsing and searching of the required data.

Table 6: Hadoop infrastructure specification

Name	HDFS Cluster	Prod	PSNC	IP	62.3.171.42 62.3.171.103
Software	Ambari, HDFS, NiFi [11]	Dev	N/R	IP	N/R
Domain	N/R				
Description	Hadoop provides HiDALGO2 with an additional Data Management Solution, focusing on storing and transferring massive amounts of data, and supporting the analysis of this large amount of data. The infrastructure contains one installation node (Ambari), two name nodes with public IP addresses, two data nodes and one node with Apache NiFi software.				

[NAME]	[CPU]	[RAM]	[DISK]	[IP PUB]	[DOMAIN]
hidalgo2-ambari	8	16GB	40GB + 500GB	62.3.170.106	prunus-106.man.poznan.pl
hidalgo2-nn-1	8	8GB	40GB + 500GB	62.3.171.42	sophora-42.man.poznan.pl
hidalgo2-nn-2	8	8GB	40GB + 500GB	62.3.171.103	sophora-103.man.poznan.pl
hidalgo2-dn-1	8	16GB	40GB + 10TB		
hidalgo2-dn-2	8	16GB	40GB + 10TB		
hidalgo2-nifi	16	32GB	40GB + 500GB	62.3.171.105	sophora-105.man.poznan.pl

Table 7: Data Streaming infrastructure specification

Name	Streaming		Prod	PSNC	IP	62.3.171.226
Software	Kafka [13] , ElasticSearch [12]		Dev	USTUTT	IP	To be allocated
Hardware	CPU cores	4	RAM	8 GB	Hard Disk	40 GB
Domain	https://stream.hidalgo2.eu					
Description	The Data Streaming system for HiDALGO2 provides fast, high performant data pipelines for streaming data, connecting data flows from input to simulations to visualisation to other outputs. They also allow for complex search queries across data sources.					

Table 8: Information Management infrastructure specification

Name	Wiki	Prod	USTUTT	IP	141.58.0.233
Software	Wiki-js [14]	Dev	PSNC	IP	To be allocated

Hardware	CPU cores	2	RAM	4 GB	Hard Disk	40 GB
Domain	https://wiki.hidalgo2.eu					
Description	The Wiki platform provides HiDALGO2 with both a public and private knowledge gathering system. This system helps record and document project specific processes, collaborate and share knowledge through documentation and tutorials to help developers and users utilise HiDALOG2 services.					

Table 9: Identity Management infrastructure specification

Name	IDM		Prod	PSNC	IP	62.3.171.16
Software	Keycloak [15]		Dev	USTUTT	IP	141.58.0.233
Hardware	CPU cores	4	RAM	8 GB	Hard Disk	40 GB
Domain	https://idm.hidalgo2.eu					
Description	The HiDALGO2 IDM enables Single Sign On for internal and external users, by ensuring that a single set of credentials can be used across the HiDALGO2 services, including the dashboard, the workflow orchestrators and all customer and supporting services. that support Keycloak credential management.					

Table 10: Forums and FAQs infrastructure specification

Name	Askbot		Prod	PSNC	IP	62.3.171.180
Software	Askbot [16]		Dev	USTUTT	IP	141.58.0.233
Hardware	CPU cores	4	RAM	8 GB	Hard Disk	40 GB
Domain	https://askbot.hidalgo2.eu					
Description	Askbot is a Forum Service to provide a convenient method for Users and Customers to view common queries. It allows experts to disseminate information and keep the project up to date as developments progress, keeping potential customers in the loop.					

Table 11: Monitoring infrastructure specification

Name	Monitor		Prod	PSNC	IP	62.3.170.54
Software	Zabbix [17]		Dev	USTUTT	IP	N/A
Hardware	CPU cores	4	RAM	8 GB	Hard Disk	40 GB
Domain	https://monitor.hidalgo2.eu					
Description	Zabbix is a framework to monitor status of all VMs and Services. In case of any problems, notifications are sent out to Admins as well as a maintenance email list.					

Table 12: Customer Support – Ticketing System infrastructure specification

Name	Ticket		Prod	USTUTT	IP	141.58.0.233
Software	Zammad [18]		Dev	PSNC	IP	62.3.170.21
Hardware	CPU cores	2	RAM	4 GB	Hard Disk	30 GB
Domain	https://ticket.hidalgo2.eu					
Description	The ticketing system, Zammad, will set up a formalised method for HiDALGO2 developers and admins to respond to any problems users and customers face while using the HiDALGO2 platform. For example, in case of any private queries related to account credentials on the Workflow Orchestrators, the users can submit a "ticket" with a description of their problem, and a HiDALGO2 representative would get back to the user with the ideal solution.					

Table 13: Educational Platform infrastructure specification

Name	Moodle		Prod	USTUTT	IP	141.58.0.233
Software	Moodle [19]		Dev	PSNC	IP	62.3.171.60
Hardware	CPU cores	2	RAM	4 GB	Hard Disk	30 GB
Domain	https://moodle.hidalgo2.eu					
Description	Moodle is a popular open-source learning management system (LMS) that provides a platform for creating online courses and managing virtual learning environments. This platform will allow HiDALGO2 experts to create and organize courses with various resources such as text, multimedia, quizzes, assignments, and more. Courses can be structured with different sections, and content can be organised in a hierarchical manner, and also be used to provide feedback to the training course participants.					

Table 14: Project Management infrastructure specification

Name	Open Project		Prod	USTUTT	IP	141.58.0.83
Software	Open Project [20]		Dev	PSNC	IP	To be allocated
Hardware	CPU cores	2	RAM	4 GB	Hard Disk	40 GB
Domain	https://project.hidalgo2.eu					
Description	HiDALGO2 uses OpenProject to carry out a broad synchronization of the planning and progress of the individual tasks of the project. These are organised in close alignment with the Grant Agreement, with tasks divided according to Work Packages and their constituent tasks.					

Table 15: Workflow Orchestrator: MathSO Portal infrastructure specification

Name	MathSO		Prod	PSNC	IP	62.3.170.136
Software	MathSO		Dev	USTUTT	IP	To be allocated
Hardware	CPU cores	8	RAM	16 GB	Hard Disk	40 GB
Domain	https://portal.hidalgo2.eu					
Description	SZE's MathSO platform's primary function is to deploy and monitor HPC workflows through a web portal without the need of HPC knowledge, or even CLI knowledge of the application. This provides a convenient method for users and developers of HiDALGO2 to execute their simulations on varying HPC systems.					

Table 16: Workflow Orchestrator: QCG Portal infrastructure specification

Name	QCG		Prod	PSNC	IP	Not public yet
Software	OS: OKD 3.x/OKD 4.x [42]		Dev	USTUTT	IP	To be allocated
Hardware	CPU cores	30	RAM	8 GB	Hard Disk	30 GB
Domain	https://qcg.hidalgo2.eu					
Description	PSNC's QCG Portal is also a Workflow Orchestrator like MathSO, with a highly customised GUI that provides a clear and easy way for users to run and monitor their applications on the HPC system of their choice. The Portal is linked to the HiDALGO2 ecosystem via the Keycloak SSO mechanism.					

Table 17: Dashboard infrastructure specification

Name	Dashboard		Prod	N/A	IP	N/A
Software	N/A		Dev	N/A	IP	N/A
Hardware	CPU cores	N/A	RAM	N/A	Hard Disk	N/A
Domain	N/A					
Description	The HiDALGO2 dashboard will provide a central point of access for all main HiDALOG2 web services as well as running and monitoring simulation applications through the Workflow Orchestrators. The resources will be allocated once development and testing of the portal is further underway.					

2.2 Design and Architecture

The following section arranges the previously described components of the HiDALGO2 infrastructure into a cohesive design and architecture.

In Figure 3's System Context level diagram [21] , we see how HiDALGO2 fits as a project into the wider scheme of EuroHPC. The vision provided by the EuroHPC JU and coordinated through CASTIEL2 [22] , is to leverage the computing power of all the main HPC sites of Europe. In return, HiDALGO2 will offer the use of its simulation pilot codes and exchange knowledge and training through its services.

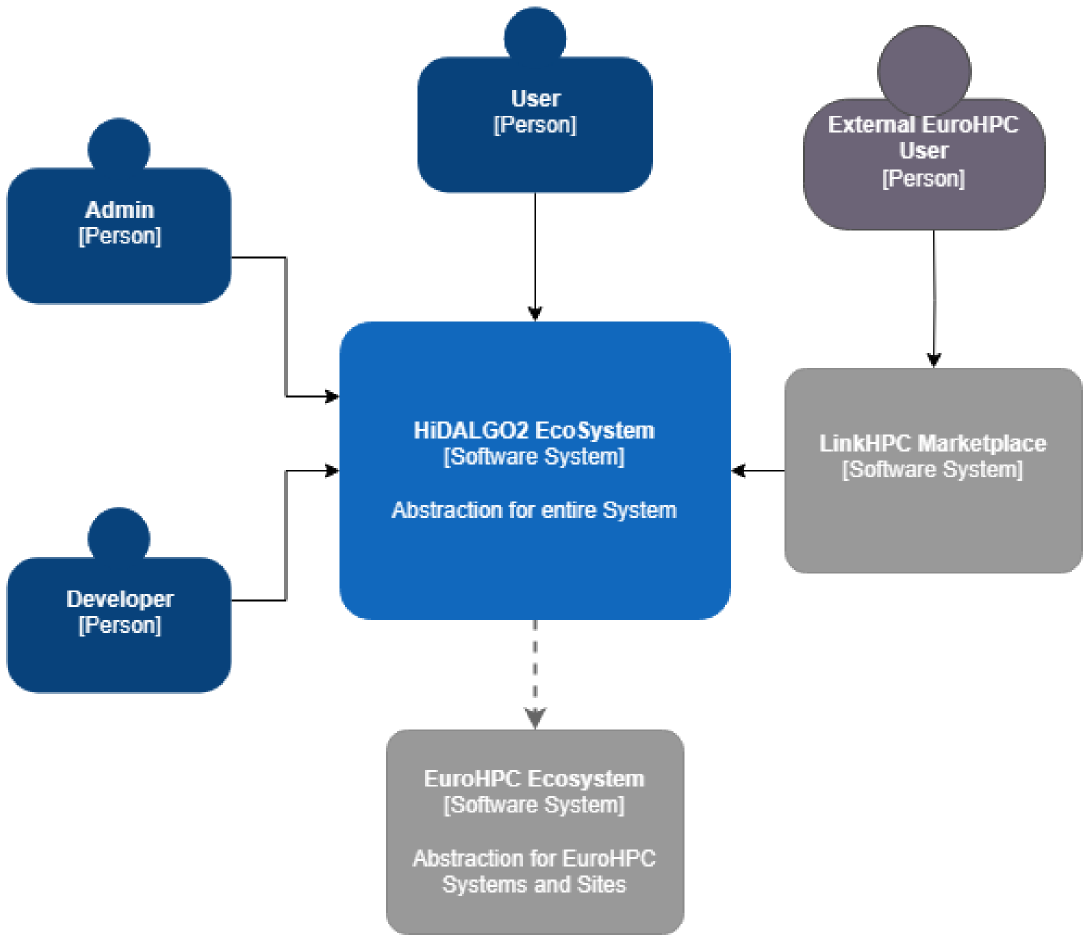


Figure 3: System Context Diagram for HiDALGO2

In Figure 4, as specified in the C4 modelling system [21] we go one level lower and observe the main building blocks of the HiDALGO2 ecosystem and how different users are expected to interact with them.

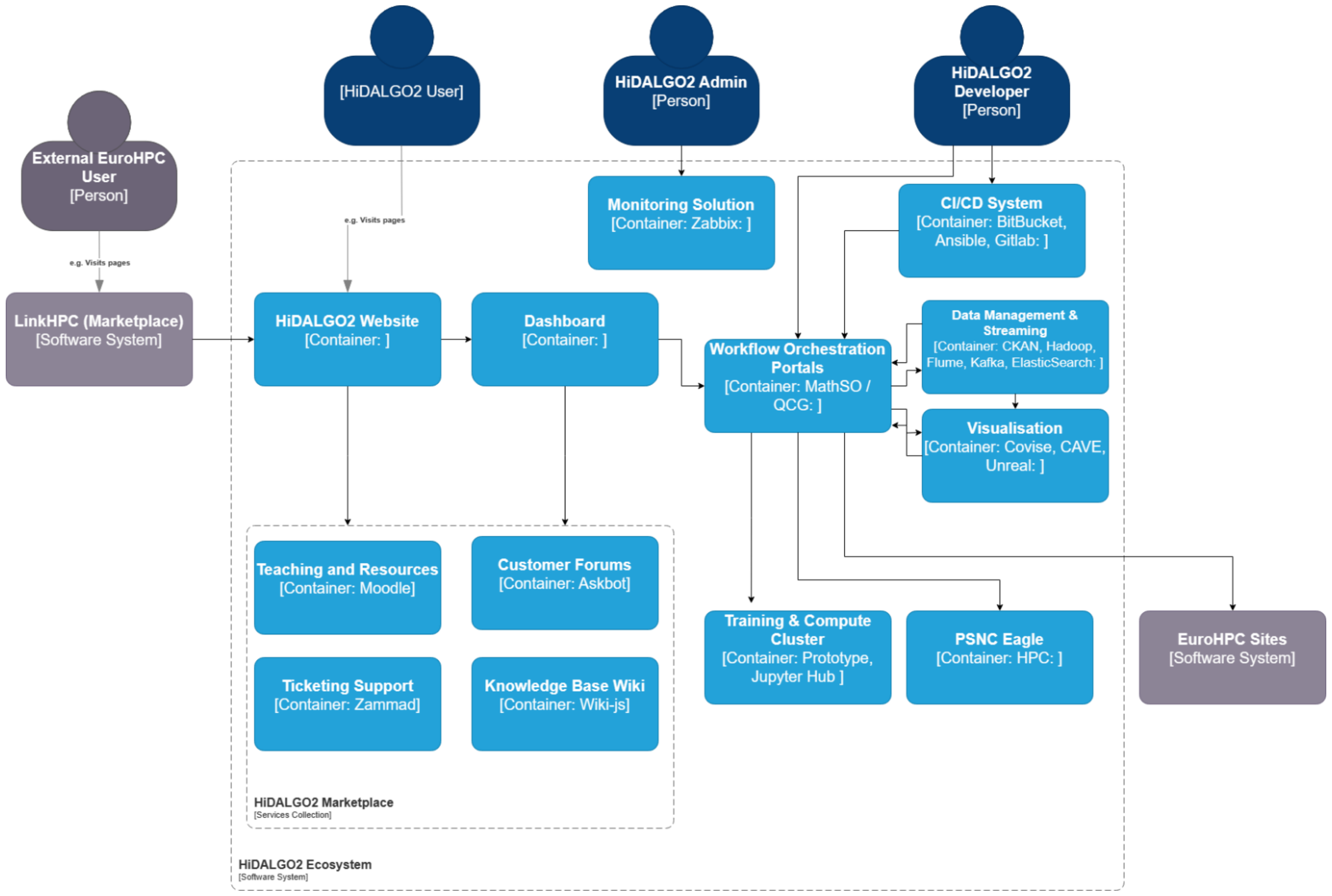

[HiDALGO2 User]
HiDALGO2 Admin [Person]
HiDALGO2 Developer [Person]
External EuroHPC User [Person]
e.g. Visits pages
e.g. Visits pages
LinkHPC (Marketplace) [Software System]
HiDALGO2 Website [Container:]
Dashboard [Container:]
Monitoring Solution [Container: Zabbix:]
CI/CD System [Container: BitBucket, Ansible, Gitlab:]
Workflow Orchestration Portals [Container: MathSO / QCG:]
Data Management & Streaming [Container: CKAN, Hadoop, Flume, Kafka, ElasticSearch:]
Visualisation [Container: Covise, CAVE, Unreal:]
Teaching and Resources [Container: Moodle]
Customer Forums [Container: Askbot]
Ticketing Support [Container: Zammad]
Knowledge Base Wiki [Container: Wiki-js]
HiDALGO2 Marketplace [Services Collection]
Training & Compute Cluster [Container: Prototype, Jupyter Hub]
PSNC Eagle [Container: HPC:]
EuroHPC Sites [Software System]
HiDALGO2 Ecosystem [Software System]

Figure 4: System Container Diagram for HiDALGO2

Most users will be redirected, either through the HiDALGO2 project website or from the LinkHPC marketplace (a planned library or catalogue of EuroHPC services) to the HiDALGO2 Dashboard. From the Dashboard, users can select the applications and simulation codes they are interested in, which will take them to the HiDALGO2 Workflow Orchestrators (WFOs), MathSO and QCG, where they can customise, run and monitor their applications.

The simulation codes will run according to the user's customisation and their choice of HPC system. The WFOs will provide an easily accessible GUI for the users to run and monitor their simulation applications, and the HiDALGO2 data management system (DMS/CMS) will transfer the results of the application run from HPC systems to HiDALGO2 servers.

The WFOs can then go through the data logs, as well as visualise the data results through a multitude of visualisation tools that HiDALGO2 provides, e.g. the COVISE application[23] .

The Dashboard also allows users the opportunity to browse through available services, e.g. Moodle for educational courses related to the simulation use cases and running them on HPC systems, customer services such as the Zammad ticketing service, or go through the public Wiki for information on the project.

Internally, the developers of the pilot simulations use cases will link their local development repositories to the central repository of HiDALGO2. From here, the HiDALGO2 CI/CD will build the pilot codes and deploy them to the HPC systems. Then they can run the simulations by hand, or also through the monitoring and visualisation tools through the HiDALGO2 WFOs.

A further objective of the project is to deploy and run the simulation use cases from the HiDALGO2 system to the EuroHPC sites, e.g. LUMI in Finland. This has already been achieved in a limited scope through SZE's UAP pilot and will be extended to all other HiDALGO2 pilot applications.

2.3 Development Roadmap

Key points in the future development of the HiDALGO2 technical infrastructure are detailed in the following subsections.

2.3.1 Further access to EuroHPC Sites

HiDALGO2 partners have until now acquired access to EuroHPC systems through the Benchmark & Development Access Calls in order to deploy, test and benchmark their pilots. Some partners, like UNISTRA, have access to the EuroHPC currently through ICCS access, to deploy and test their pilot code deployment. In the next periods of the project, partners will utilise the Regular and Extreme Scale Access Calls, and increase the coverage of the HiDALGO2 pilot use cases across the HPC systems and partitions. The deliverable D3.1, "Scalability, Optimisation and Co-Design Activities", will go into further detail on this topic.

Table 18: Current EuroHPC JU machine coverage matrix

	Urban Air Pollution (SZE)	Urban Building Modelling (UNISTRA)	Renewable Energy Sources (PSNC)	Wildfires (MTG)	Benchmarking (ICCS)
LUMI	Awarded	To be requested	Awarded	Awarded	Awarded
Vega	Awarded	To be requested	To be requested	Awarded	Awarded
Karolina	Awarded	To be requested	Requested / Waiting	Awarded	Awarded
Meluxina	Awarded	To be requested	Awarded	Awarded	Awarded
Discoverer	Awarded	To be requested	To be requested	Requested / Waiting	Awarded
Leonardo	To be requested	To be requested	To be requested	To be requested	Awarded
MareNostrum5	System not available yet	System not available yet	System not available yet	System not available yet	System not available yet
Deucalion	System not available yet	System not available yet	System not available yet	System not available yet	System not available yet

2.3.2 Extending Web Services

The currently available services that make up the HiDALGO2 ecosystem are to be extended with additional functionality. An overview of the latest status of the services is tracked in a live document, a screenshot of which is added as an appendix in Annex 1.

- All Web Services are to be integrated with the HiDALGO2 Keycloak IDM, as well as the Operational Portal services and the Workflow Orchestrators.
- 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.
- 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.
- 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.

3. Workflow Orchestration

The execution of applications in HPC clusters requires running complex pipelines that demand high expertise in multiple technology domains, including secure access to HPC frontends, deployment of application binaries and input datasets within those frontends, dispatching application jobs into the HPC scheduler, or uploading jobs' results for further analysis and visualisation.

Average HPC users (mostly scientists and engineers, like in the HiDALGO2 pilots) rarely have the expertise to cope with these execution pipelines. Workflow Orchestrators (WFOs) assist these users in the end-to-end automated execution of these pipelines, for applications that, like most of HiDALGO2 pilot's experiments, can be specified as a job-based workflow, where each job is submitted to a target HPC cluster for execution, before its input datasets and parameters have been procured. WFOs manage the execution of jobs within target HPCs and the collection of their outputs, either to procure inputs for next jobs in the execution workflow or to collect final results.

Due to this complexity of running applications on HPC systems, HiDALGO2 will provide two Workflow Orchestration (WFO) engines, namely MathSO and QCG. Offering multiple WFOs to end users increases what HiDALGO2 can offer for HPC application execution, as both WFOs provide unique and complementary features that can fit a wider range of adopters. This chapter will go into more depth into these features and how users can make the most of the offerings of the two WFO's.

Additionally, offering two WFOs extends the number of pilot use cases and EuroHPC systems that HiDALGO2 can target successfully. Each EuroHPC system has some differing processes and dependencies, and having two different teams developing two different WFO's enables a closer supporting of individual pilot use cases and EuroHPC systems. For example, in case QCG does not currently support a particular pilot requirement or target EuroHPC system, HiDALGO2's development attempts to coordinate in such a way that it is likely that MathSO will prioritise supporting this particular pilot or EuroHPC system. In this way, HiDALGO2 attempts to provide a more complete and in-depth pilot application support for end users, as well as wider EuroHPC coverage.

In the following subsections, we will provide the initial specification, both functional and technical, of both these WFOs. The functional specification is based on requirements elicited from pilots' stakeholders, which are described in Section 3.1. Based on these requirements, but also on the own feature roadmap planned by the WFOs owners, as a follow-up from HiDALGO2, the functional specification for each WFO is provided in Section 3.2. Section 3.3 provides a technical specification of both WFOs, framed within the overall HiDALGO2 architecture, introduced in Section 2.2. This section concludes with Section 3.4, which describes the WFOs' development plan for new features, within the HiDALGO2 lifetime.

3.1 Requirements

The list of elicited requirements is included in Annex 2. The following provides a summary of the most relevant requirements, grouped by similarities.

A first group of requirements (REQ_WFO_3, REQ_WFO_15, REQ_WFO_16) are related with the procurement of input datasets and parameters needed for workflow's job execution in target HPC centres. To support this data procurement, one requirement expresses the need to interoperate with the HiDALGO2 DMS (REQ_WFO_2). Other requirements indicate the need to specify the target HPC system for job execution, either manually (REQ_WFO_4), or automatically, based on some optimisation objectives (REQ_WFO_10). Related to HPC connectivity, other requirements express the need to manage the secure connections with the target HPC system and the safekeeping of users' credentials (REQ_WFO_1). Some requirements express the need to support specialised jobs, either based on containers (REQ_WFO_11) or ensembles (REQ_WFO_6). There are more requirements related to container-based jobs, such as the need for an image registry (REQ_WFO_14), or their interoperability and performance, scalability reliability and security in HPCs (REQ_WFO_17, REQ_WFO_18, REQ_WFO_19, REQ_WFO_20). The list includes other ones, related with the workflow management (REQ_WFO_12, REQ_WFO_13).

This list is not exhausted and will be evaluated and eventually refined and extended with new needs identified by pilot's stakeholders and the HiDALGO2 technical team (e.g., to fulfil the needs identified by other technical work packages) along the project's development phases.

The next section will consider these requirements to elaborate the features that the HiDALGO2 WFOs will offer for application execution in HPCs.

3.2 Functional Specification

HiDALGO2 project will provide two WFOs engines, namely MathSO and QCG. Both engines permit application (e.g., pilot simulations) owners to specify them as job-based workflows, and execute them in selected target HPC clusters. These engines offer complementary functionalities that are described in the following subsections.

3.2.1 MathSO

MathSO portal is a python-based web application which was developed to fulfil the requirements and functionalities by HiDALGO2.

The main function is to deploy and monitor HPC workflows through a web portal without requiring users' expertise on HPC systems, or even on the scheduler used to deliver their jobs.

Each workflow is defined as an application. The application defines:

- job requirements, such as the input files of the job,
- their repository location,
- the parameters of the job or the management of the job outputs to be uploaded into a results repository.

This application is described in a YAML format (defined by SZE), which supports multiple jobs in a single workflow and dependencies between them. The portal itself can handle many HPC clusters per user. The user can define HPC access by adding her credentials and define which HPC partition she wants to use. Each different partition configuration adds a new HPC entry to the list available to select where to run the application. These credentials are stored encrypted in the portal's database.

The portal can access the HiDALGO2 CKAN data repository, by integrating the API key authentication method. The user can upload/download datasets using only the portal.

MathSO portal uses any OAuth2-based IAM tool to authorise users, in particular Keycloak, an open-source implementation. This service is deployed within the portal by default. However, in HiDALGO2, its Keycloak instance will be used instead.

MathSO currently support the industry standard Slurm job manager to support predefined cloud infrastructure usage.

Job Management

The portal uses its orchestrator module to connect to the HPC infrastructure through an SSH connection. SSH is used as de facto standard in HPC systems and widely available across most of EuroHPC systems (some use a VPN connection to increase the security level). Through this connection the MathSO portal monitors the job status continuously. After the job finishes, the user can see and download all the logs created by the workflow, or can visualise the results through the web interface.

Input repository

The portal connects to the CKAN data repository, which is selected as one of the DMS repositories in the HiDALGO2 project, by using its API. It can show the available datasets, create a new one and fill it with data, or edit its preferences like tags. By adding some special input fields in the YAML application description, the desired dataset can be selected during the instance configuring process.

User integration

Using a third-party OAuth2 authentication method, MathSO portal can be used with a variety of IDMs. MathSO uses Keycloak as a deployed solution for IDM. We define 4 different user types:

- demo user - The demo user role cannot create any new application, cannot add an HPC system and a Data Repository configuration, and can only access certain applications, which were selected as available for demo during uploading.
- user - The normal user can define HPC system and Data Repository entries for its own profile, but can only use applications already uploaded to the Portal.
- developer - The developer can do all the activities of the normal user but can also upload/delete applications as well.
- administrator - Administrator role is mainly for maintaining other users.

The users can bind different HPC accounts to their MathSO account. The connection can be defined by credentials or various SSH key authentication methods like RSA or ed25519.

Visualisation

To be able to visualise workflow output data, SZE has developed a custom tool, which is fully integrated into the portal. It can visualise different meshes and geometry formats through the web interface. The visualisation tool will be further described through work in WP4.

Support all available EuroHPC Systems

MathSO is already implementing some new features required by some of the EuroHPC centres. For example, LUMI needs to add the LUMI account ID to the submitted job, and MathSO needs to define the precise partition on which we are going to run our application, because the "default" partition is not defined on LUMI system. Moreover, MathSO needs to add some special parameters like QoS and accounting information on some specific HPC systems. Thus, MathSO considers the differences in HPC Systems.

3.2.2 QCG Portal

QCG-Portal, or shortly QCG, aims to simplify the definition and execution of computing scenarios on remote computing resources. With an intuitive and highly adjustable interface, integrated data management and compliance with modern authentication and authorisation protocols, OAuth2.0 and OIDC in particular, it can efficiently support diverse application types ranging from basic serial applications, through parallel ones built on top of MPI, OpenMP or CUDA, and ending with complex ensemble executions and workflows. In this way QCG addresses the needs of a variety of target user communities and scenarios that require HPC power. Importantly, QCG has been already employed by diversified real-life research (e.g., EOCOE[3], DEEP[4], PRACE-LAB[5]) and commercial (VOXBOX[6]) projects.

QCG ensures efficient access to the computing resources managed by Slurm LRMS, thus it can interact with a majority of current EuroHPC supercomputers. Also, since the system architecture boasts a high-level of flexibility, the system's deployment can be relatively easily adapted to local policies and restrictions of particular resources. For the time being, QCG offers the following main functionalities:

Job submission / management / monitoring through a modern web portal

The main goal behind QCG is to bring an effective environment for the remote execution of complex computational experiments on top of HPC systems. Consequently, the full-featured and user-oriented job submission, management and monitoring are basic mechanisms that QCG, through a web portal, offers to its audience.

QCG can run jobs of different type and complexity, including serial jobs as well as parallel jobs based on MPI, OpenMP or CUDA technologies. Furthermore, QCG is able to run not only the applications that are pre-installed on a computing resource, but also those that are provided in the form of the Apptainer container images. The undoubted advantage of the execution of container images with QCG is that they can be easily built by users on their own computers and then seamlessly uploaded to the target computing resource through web interface. Likewise, regular applications, the container-based applications may benefit from the usage of parallelisation mechanisms. However, it should be kept in mind that the ensuring of the proper and scalable functioning of MPI codes requires consistency of the MPI versions used to build the image and to run it on the cluster[7].

Data Management

QCG uses an IBIS system developed by PSNC for in-house data management. Thanks to its synergistic integration with IBIS, QCG allows users to benefit from automatic data staging-in and staging-out to and from experiments. This is supported by an ergonomic selection process for input files and the ability to browse both the experiment's results directory after computations and the working directory during computations, all from a user-friendly explorer-like interface built into the portal. The QCG-IBIS integration also allows users to manage and store data in a dedicated cloud-accessible space or in a folder on the cluster.

[3] https://www.eocoe.eu
[4] https://deep-hybrid-datacloud.eu/
[5] https://prace-lab.pl/
[6] https://www.vox.pl/projekty-voxbox
[7] https://apptainer.org/docs/user/latest/mpi.html

To ensure seamless integration with the HiDALGO2 ecosystem, QCG will be extended to support the selected DMSs that will be used in the project. The integration will be implemented primarily at the workflow orchestration level, with some user interface integration as well.

CKAN DMS is the most likely candidate for initial integration, as it was already verified during the HiDALGO project and provides valuable features for HiDALGO2. Integration with CKAN is anticipated to be relatively smooth, as both QCG and CKAN are designed to run on the Internet and conform to Internet security standards, like OAuth2.0/OIDC protocols.

On the other hand, direct integration between QCG and Hadoop/HDFS is considered more problematic because Hadoop/HDFS is not compatible with OAuth2.0/OIDC, but rather uses IAM based on Kerberos. Therefore, integration between the web part of QCG and Hadoop/HDFS is unlikely to be implemented unless appropriate extensions to the core mechanisms of Hadoop/HDFS are discovered or developed from scratch and then tested. That being said, the workflow orchestration can be realised in a hybrid mode, where multi-domain executions (between EuroHPC sites) are realised with CKAN, while a single-domain executions relay on Hadoop/HDFS for the greater performance.

Workflow Orchestration

Currently, QCG supports execution of so-called in-allocation workflows, that consist of steps executed inside a single task in a queuing system. The in-allocation workflows are executed with the help of the QCG-PilotJob tool, being a lightweight implementation of the Pilot Job paradigm [24] . The QCG-PilotJob service can easily be run inside a running task on a cluster and plays the role of a user-personal task scheduler. Then, by tackling some tasks by its own, it can relieve the cluster's scheduler of their processing. It is therefore perfectly suited for running scenarios where many jobs need to be executed, such as diverse ensemble-oriented experiments or uncertainty quantification studies. QCG-PilotJob's also supports the orchestration of DAG workflows, thus it allows for execution of more sophisticated scenarios. QCG-Portal currently offers a basic template for executing QCG-PilotJob, but this will be extended in the future.

That said, in-allocation workflows are not sufficient for the needs of HiDALGO2 project. For the complex workflow orchestration, with sizeable tasks and heterogeneous resource requirements, the full-featured mechanism needs to be implemented at both user interface and service layers of QCG. To enable workflow orchestration and processing of subsequent steps of a workflow without user interaction, it is crucial to ensure the availability of user's credentials through the whole period of a workflow execution.

It is also important to follow a well-recognised workflow description format, preferably common for the HiDALGO2 project. According to the circumstances, it may be needed to enable, thus describe, hybrid execution of workflows, where some steps are local to a computational resource, and some of steps may need to be executed across many domains.

Depending on the selection of a workflow format and the availability of the tools supporting it, QCG may need to be further extended with a graphical interface for workflow definition and the monitoring of its execution. If a decision is made to self-develop such a mechanism, it will likely be based on the Task Execution Monitoring platform described in below subsection.

Cyclic Tasks

Cyclic execution of tasks is the next planned feature for QCG. Cyclic execution is requested whenever periodical simulations are required by the scenario. An example application interested in cyclic tasks is the RES Pilot [25] , but we expect that other HiDALGO2 use-cases can be interested in this feature as well. In order to enable automatic submission of periodic tasks, it is necessary to enable access to valid user credentials even after the user logs off. In this respect, the core functionality required by this mechanism is like the core functionality required by the workflows.

Parameter Sweep

Parameter sweep tasks are a common technique to explore parameter space of a model. With QCG, our goal is to enable multidimensional parameter space exploration. Depending on requirements of HiDALGO2 Pilots, the Parameter Sweep functionality will be based on QCG-PilotJob, Slurm Job Arrays or selected Workflow Orchestration mechanism.

Security

Ensuring safe and easy access to HPC infrastructure has been a challenge for many years. For a long time, even the words "safe" and "easy" themselves were seen as contradictory when users had to deal with traditional GSI proxy certificates. Fortunately, thanks to modern authentication and authorisation mechanisms, access to computing resources is becoming increasingly more and more smooth. The popularisation of OAuth2/OIDC protocols has changed the way how web services cooperate and allowed users to benefit from the Single-Sign-On (SSO) mechanisms. QCG follows this idea for the authentication and authorisation of users and services on all levels of its architecture, starting from federated login, through the cooperation with supplementary services, like data management systems, and ending, whenever possible and permitted, with token-based authentication of users on a computing cluster. For all of these purposes, QCG uses the Keycloak IAM infrastructure and JWT tokens by default.

CLI and REST API

Graphical web portal is not the only possible way to run experiments on computing infrastructure offered by QCG. The more conservative approach is to use QCG command line client (CLI), which is also provided to users as an alternative to the main graphical interface. What should be noted, the CLI supports OAuth2 Device Authorisation Grant[8], thus it stays in harmony with the core security mechanisms and token-based authentication. For more demanding use-cases, there is also an option to directly use REST API exposed by QCG.

Integration with a cluster

QCG enables remote access to Slurm-managed clusters. The preferred integration scheme is to deploy a lightweight Python QCG-Agent service on the Slurm head node. The agent's main role is to efficiently broker between the remote QCG-Portal and the Slurm controller daemon. To do this, QCG-Agent must be able to connect to the remote QCG-Portal and execute commands on behalf of the particular user who submitted the job. The former requirement can be met by allowing agent connections to the remote service in the firewall, while the latter requires configuring `sudo` permissions for the agent.

One of the first goals on the HiDALGO2 QCG development roadmap is to enable uniform access from QCG-Portal to EuroHPC resources. Being aware that not all computing centres are willing to install custom software on their resources, we are developing an alternative QCG-Agent that will communicate with Slurm via SSH. At the cost of efficiency, it will provide more portability.

There are two main subtasks behind this goal:

a) **Development of an SSH agent**

   In order to enable access to EuroHPC resources, where deployment of an original QCG-Agent is not possible, we have started to develop an alternative QCG Agent, which is not deployed on a cluster, but it is deployed externally and connects to the cluster via SSH protocol. Currently, the proof-of-concept version of the agent is already developed and it will be fine-tuned in the next months.

b) **Management of user's credentials**

   Since the usage of SSH protocol for communication with the cluster requires user credentials to be provided in an SSH-compatible form, QCG needs to support these new types of credentials. Basically, there are two high-level choices related to the type of SSH credentials we may support:

   - a pair of traditional keys (private and public),
   - SSH certificates.

[8] https://datatracker.ietf.org/doc/html/rfc8628

> Undoubtedly the keys are still the most popular solution to access computing resources, but they are vulnerable to security risks. The main problem is that the private keys are valid for a long time and therefore can be a subject of interest for malicious individuals who can steal them. SSH certificates are more secure, but these are still rarely supported. Nevertheless, it should be noted that users of the Leonardo cluster are obligated to use SSH certificates and there are ongoing works to enable SSH certificates on LUMI. The final decision on which of two options should be implemented will be taken during the next few months of the project.

The next short-term goal is to enable access to many clusters from a single instance of QCG-Portal. At this stage the portal is bound to a single cluster only. Thus, if you need to support many clusters, you need many QCG-Portals. Since this limitation is highly inconvenient for multi-cluster environment, like the HiDALGO2 one, it will be resolved soon.

Customisation

One of the key priorities when designing the QCG system was to enable easy adaptation of the offered features toward actual needs of diversified use-cases and various groups of users. As a result, QCG provides two complementary customisation options that can significantly improve usability of the portal by target communities, namely:

- application templates,
- application-specific graphical interfaces.

Application templates are the basic mechanism included out-of-the-box with the portal. With the templates, expressed as JSON files, local QCG administrators can define both how the interface should look for a particular application and how the application should be executed on a cluster. The big advantage of this approach is the easiness of creating a new template, however the drawback is that the interface can be built only from predefined building block, which is not always sufficient. For more demanding scenarios, QCG supports embedding custom graphical interfaces into the portal, which can be integrated with the portal's main functionalities to improve input definition, application progress monitoring, and results presentation.

Remote desktop applications

Another interesting functionality supported by QCG is forwarding remote desktop applications running on a cluster to a user's laptop. This functionality has a couple of possible applications. First, it allows users to run computationally demanding visualisations on a cluster and benefit from the hardware acceleration available on cutting-edge GPU nodes. Second, it allows users to access software with visual interfaces that is licensed or difficult to install on their personal computers. A practical example of application of this mechanism is a JupyterLab environment that is available from QCG-Portal, while runs on a remote cluster.

Visualisation

During the HiDALGO2 project, QCG will be integrated with the Visualisation tools requested by the pilots. The selection of tools and the order of their integrations will depend on the expectations of HiDALGO2 Pilot owners as well as the functional circumstances. It is anticipated that the priority of integration will be given to the tools developed by HiDALGO2 partners, i.e. [26] . QCG will enable integration of both the tools with the server-side rendering as well as the tools with client-side rendering.

Task execution monitoring

In order to enable comprehensive monitoring of pilot executions, thus enhancing the basic mechanisms of tasks monitoring available already in the portal, QCG will be enhanced with a full-featured monitoring solution. It will have a fourfold purpose:

1. Tracking the progress of workflow execution
2. Tracking the progress of execution of ensembles or subtasks managed by toolkits like QCG-PilotJob or mUQSA;
3. Monitoring of resources consumed by a task, like CPU usage or memory usage;
4. Live monitoring of the user application execution, in particular tracking the progress and visual validation of produced results (with custom application monitoring templates).

The monitoring solution for QCG will provide support for various means of graphical representation of monitored data, including tables, charts and images, and thanks to the compatibility with template paradigm provided by QCG, it will be adaptable to certain needs of pilots or individual users.

mUQSA

mUQSA (Multipurpose Uncertainty Quantification and Sensitivity Analysis) is a comprehensive solution based on QCG software and the EasyVVUQ library for the automated UQ&SA of diverse models, including complex parallel applications. mUQSA in a form of custom application graphical interface built on top of QCG-Portal, provides an intuitive wizard that allows users to define UQ&SA studies according to popular techniques and then, after required computations, it allows to visualise the produced results in a form of an interactive web page. What is crucial, the platform takes full care of automated and efficient execution of the required computations on an HPC cluster.

In HiDALGO2, the mUQSA platform will be offered in addition to QCG in a SaaS model and will be available for all interested Pilots. At the moment it is already used by the RES application to some extent.

3.3 Design and Architecture

WFOs are included within the overall HiDALGO2 architecture introduced in section 0 and shown in Figure 4, as one of the components connected to the dashboard, which will be part of the main HiDALGO2 Website. Within this dashboard, users can get access to both MathSO and QCG portals. WFOs, will be integrated with other HiDALGO2 components, namely the DMS, the Training Cluster, and bound to both the PSNC HPC Cluster and other EuroHPC sites.

The delivery of the WFOs will be managed by the HiDALGO2 CI/CD pipelines.

The specific technical architecture of both WFOs is introduced in the following subsections.

3.3.1 MathSO

MathSO is developed mostly in Python and Javascript following modern web-based applications. As presented inFigure 5, the Portal comprises three major elements to ensure most of the functionality. The portal itself is the frontend, and there is also the backend where the business logic and workflows are developed. Last but not least, the third component is the orchestrator, which creates the connection between the portal and the HPC systems. The portal is also deployed with a fully functional Keycloak IDM, but it can be deployed without it, by using an already deployed OAuth2 capable IDM. Between the three major components and other connected components, like CKAN, the communication is mainly based on REST API. The Orchestrator establishes the connection between the portal and the HPC system. It uses an SSH to connect to the HPC system, through a library to query and control the Slurm job manager.

3.3.2 QCG Portal

QCG system, as presented in Figure 6, has a clear separation of the frontend and backend layers, where the frontend is responsible for the graphical part, and the backend, being a central component in the system, takes care of business logic. The communication in the system is realised with the well-defined API through the REST protocol.

The integration with computing resources is implemented through QCG-Agents, which are responsible for the communication between QCG backed and Slurm scheduler. There are two types of QCG-Agents: the on-cluster agent and SSH agent. Preferably, the former agent should be used, since, through the closer integration with the scheduler, it offers greater functionality and performance. Unfortunately, due to possible restrictions imposed by clusters' administrators, the use of this type of agent is not always possible, and therefore SSH agent needs to be used as a replacement.

The core QCG components are intrinsically supported by data management systems and security services. IBIS DMS is a default choice for data management in a typical deployment of QCG. The close integration of IBIS with both the frontend and backend layers of QCG, gives significant advantages, like easy selection of input data to experiments, automatic data staging, or built-in ability to preview data during calculations. Nevertheless, through the plug-in mechanisms, QCG can also be integrated with DMS.

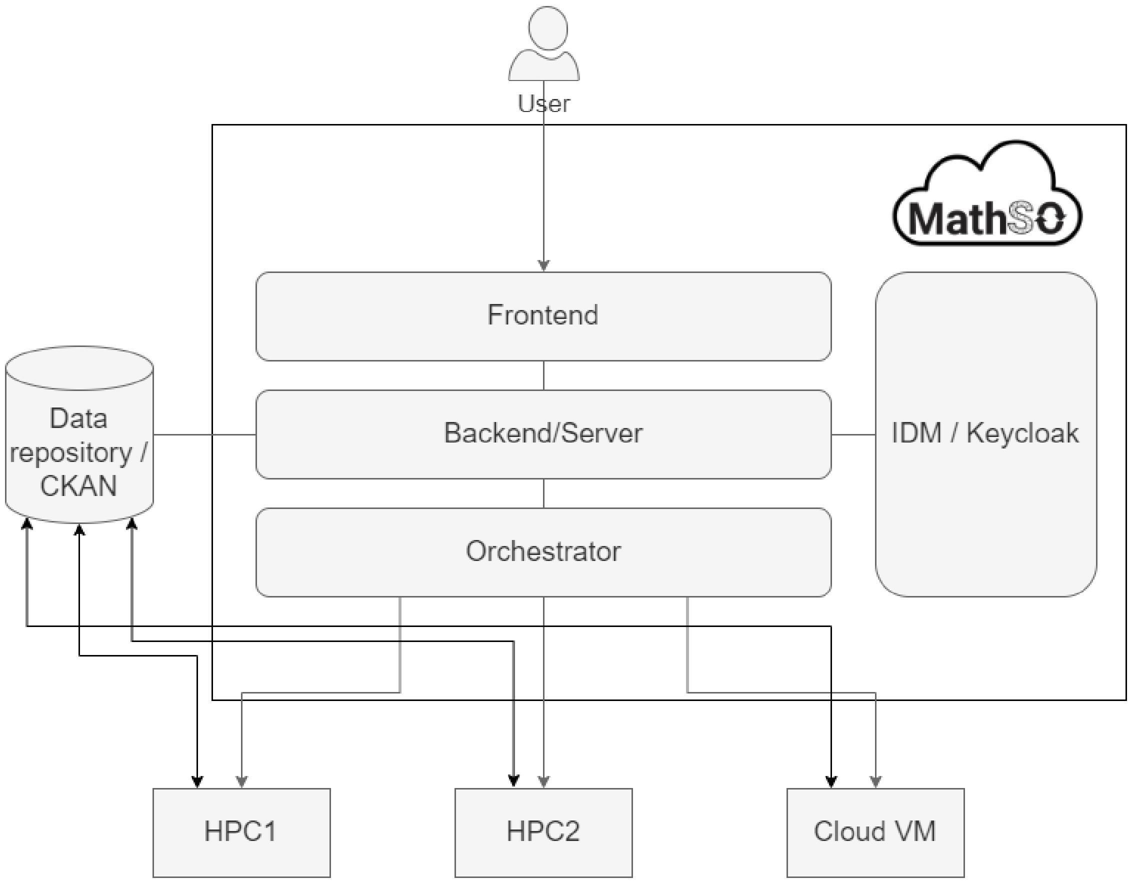


Figure 5: High-Level Design of MathSO portal

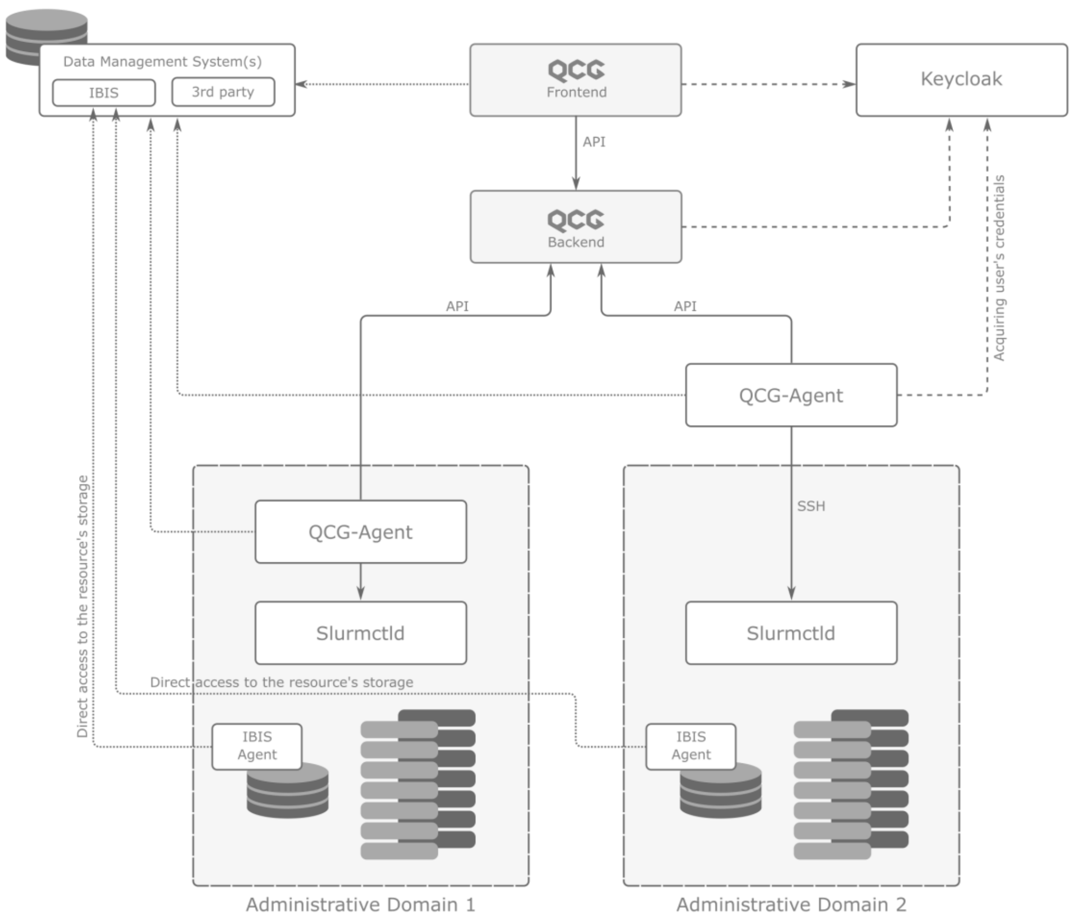


Figure 6: High-level architecture of QCG

In turn, the AAI ecosystem where QCG is deployed, should support OAuth2.0/OIDC protocols. The preferred and validated solution is Keycloak, which offers a comprehensive set of features, starting from an intuitive interface, through advanced options for management, authorisation, and federation of users, and ending with extensive configuration capabilities. In addition to the above requirements, the deployment based on the SSH QCG-Agent requires mechanisms to exchange tokens for SSH-compatible user credentials securely, like keys or SSH certificates. For this purpose, Vault or an alternative solution deployed on a particular EuroHPC resource will be used.

3.4 Development Roadmap

Some of the aforementioned features for HiDALGO2 WFOs will be implemented or extended within the project's lifetime. The following subsections list those features and planned development timelines for each engine. Annex 3 presents the development roadmaps in a Gantt chart.

3.4.1 MathSO

Parameter Sweep Function

MathSO adds this function to permit the user to define multiple jobs differing in only one parameter in one step. With this function, users are able to pick one of the parameters from a saved input set and add multiple values to it. Each value will generate a new workflow that will be deployed.

Timeline

- M18 - Feature Release

Add Data Repository Support

Although MathSO uses CKAN as main data repository, the HiDALGO2 DMS platform needs to be integrated to transfer data not only from CKAN but from Hadoop as well, if demanded by users.

Timeline

- M24 - Feature Release

Improve Data Management

For now, MathSO portal only supports selecting and sending the data which must be downloaded to the target HPC centre as input data for a job. However, within HiDALGO2 MathSO will add data management support between the target HPC and the selected data repository to not just download but re-upload the results as well.

Timeline

- M30 - Feature Release

HPC Account Monitoring

MathSO already queries job's information from the HPC centres. MathSO can also collect information about the HPC centre itself. MathSO will be extended to monitor the availability of the HPC partitions, the used/idle nodes and their overall usage.

Timeline

- M36 - Feature Release

User Notification System

MathSO have only implemented an admin notification system via mail, to report when there is some failure about the job execution. We would like to extend it to users, so if a job is finished or failed not only the developer/admin but the user also will be notified about it.

Timeline

- M36 - Feature Release

Intelligent JOB Submission

For now, only users are granted with the right to specify which HPC centre will be used for running an application. In the future we want to implement an intelligent system that can decide (based on user inputs) which HPC centre fits best the requirements based on the remaining core hours of the available HPC accounts, the desired end time, and the number of idle nodes in target partition.

Timeline

- M42 - Feature Release

3.4.2 QCG Portal

Enabling Access to EuroHPC Computing Resources from QCG-Portal

The pivotal aim of this task is to enable access to different EuroHPC sites from QCG-Portal. The task will be realised iteratively and will be dependent on 1) development of SSH QCG-Agent, and 2) selection of mechanism(s) for the management of user credentials (keys / SSH certificates)

Timeline

- M12 - Proof-of-concept
- M21 - 1st Release (depends on the selection of particular technologies)

Access to many Computing Clusters from a single instance of QCG-Portal

At this stage QCG-Portal is bound to a single cluster only. If you need to support many clusters, you need many QCG-Portals. Our goal is to enable access to many clusters from a single instance of QCG-Portal.

Timeline

- M15 - Proof-of-concept
- M21 - 1st Release

Workflows

At the moment QCG supports execution of in-allocation workflows only, which is far too less for the needs of the HiDALGO2 project. Thus, the full-featured support for workflows will be implemented at both user interface and service layers of QCG.

When implementing this support, it is important to follow a well-recognised workflow description format, preferably common for the HiDALGO2 project. Depending on particular circumstances, it may be needed to enable, thus describe, hybrid execution of workflows, where some steps are local to a computational resource, and some of steps may need to be executed across many domains.

The final goal is to provide users with an interface that will allow both the definition of a workflow as well as the monitoring of its execution.

Timeline

- M15 - Proof-of-concept
- M21 - 1st Release
- M36 - 2nd Release

Cyclic Tasks

Cyclic execution of tasks, requested explicitly by the RES pilot so far, but possibly useful also for other pilots whenever periodical simulations are required, is a next extension planned for QCG.

Timeline

- M24 - Proof-of-concept
- M30 - 1st Release

Parameter Sweep

The multidimensional parameter sweep functionality will be offered from QCG to support diverse parameter explorations scenarios.

Timeline

- M18 - Proof-of-concept

HiDALGO2 Data Management System

The integration of QCG-Portal with CKAN and possibly Hadoop/HDFS DMS is going to be implemented in addition to existing and core DMS supported by QCG, i.e., IBIS. Support for new systems is mandatory for effective execution of HiDALGO2 workflows.

Timeline

- M15 - Proof-of-concept of CKAN integration
- M24 - 1st Release of CKAN integration
- M15 - Proof-of-concept of Hadoop/HDFS integration (conditional)
- M24 - 1st Release of Hadoop/HDFS integration (conditional)

Visualisation

Support for the visualisation tools on a level of Workflow Orchestrator GUI is a common feature requested by Pilots. However, since the Pilots may be interested in different tools and at the different stages of the project realisation, the support will be added dynamically depending on Pilots requirements. For example, one requirement from the Wildfires pilot is that the visualiser should provide support as well to export the interchange file formats which are injected in the different visualisation solutions, such as FGA, VDB and SHP formats, as well as generic raw ASCII and Binary files.

Timeline

- M18 or M21 - Proof-of-concept (with the first tool)
- M27 - 1st Release (one or more tools)
- M36 - 2st Release (one or more tools)

Tasks Execution Monitoring

In order to enable sophisticated monitoring of HiDALGO2 experiments executed with QCG, an advanced QCG-Monitoring solution will be provided in addition to the basic mechanisms of monitoring already available in QCG-Portal.

Timeline

- M15 - Proof-of-concept
- M21 - 1st Release
- M30 - 2nd Release

mUQSA

The mUQSA platform will be offered as an extension to QCG Portal to support automated Uncertainty Quantification and Sensitivity Analysis.

Timeline

- M12 - Proof-of-concept
- M18 - 1st Release
- M33 - 2nd Release

4. Dashboard Architecture

Creating a modern web dashboard architecture involves a multitude of considerations, technologies, and design principles. The complete design of the HiDALGO2 dashboard will be carried out in an upcoming deliverable D2.7, "HiDALGO Dashboard and Services". In this deliverable, the scope is limited to a brief overview of considerations that the Dashboard must comply with in order to integrate the HiDALOG2 ecosystem, as well as explore key aspects such as user experience design, data management, scalability, security, and technology stacks.

4.1 Vision and Requirements

The HiDALOG2 Dashboard is envisioned to be a central point, or the operational portal, for all stakeholders to access the HiDALGO2 ecosystem. Users will be guided to choose their services of choice, from running visualisation tools to submitting help tickets. All services will be conveniently available from a central overview screen.

Drawing from previously collected requirements in deliverable D2.1, "Requirements Analysis and Scenario Definition", a summary of the key requirements pertaining to the planned Dashboard Portal implementation are summarised here:

- **REQ-POR-001 – 006:** Select an HPC application from so called marketplace or pilot selection. After the configuration of the pilot, the user needs to be able to select an HPC system and run the selected pilot using unified input and output selection. The portal needs to be able to select input files from CKAN datastore and move results back to it.
- **REQ-POR-007 – 008:** Standardise workflow between the two available Workflow Orchestration portal MathSO and QCG.
- **REQ-POR-009 – 012:** Set up a workflow for different pilots on the WFOs,
- **REQ-POR-014 – 015:** Real-time infrastructure monitoring of the user defined HPC infrastructure.
- **REQ-POR-016 – 017:** Authentication and Authorisation throughout the whole HiDLAGO2 infrastructure.
- **REQ-POR-019 – 020:** Data management and data movement between HiDALGO2 ecosystem and EUROHPC machines. It is planned to integrate the Data Management System developed in WP4 into the Operational Portal.
- **REQ-POR-020 – 026**: Connect and integrate all web services to the portal e.g. Ticketing System, Training System etc**.**
- **REQ-POR-027 – 034:** Web based visualisation tool requirements and features.

These visualisation requirements are for the web visualiser developed by SZE which will be integrated with the Dashboard. Under the web-renderer codename SZE developed a Web Assembly based visualiser that can continuously render data on web interface from different data sources mostly using the HTTP protocol. This visualiser will be further described through WP4 of HiDALGO2.

4.2 Design Guidelines

As mentioned before, the detailed design and architecture of the HiDALGO2 dashboard will be planned out in the upcoming deliverable D2.7, "HIDALGO2 Dashboard and Services". A brief overview of the design guidelines that will dictate the actual architecture is given here.

Application Architecture

The Dashboard Portal will follow principals of modern software design, especially following recommendations of how-to best interface and integrate with the disparate internal and external web services in the HiDALGO2 environment, as developed in the field of Enterprise Application architecture [27] .

User Experience Design

The design of a modern dashboard should prioritise usability. This includes a clean layout, understandable navigation, and interactive elements like drop-down menus, sliders, and buttons. The goal is to make it easy for users to find the information they need without extensive training or documentation. Dashboards must be accessible on a variety of devices, including desktops, tablets, and smartphones. A responsive design ensures that the dashboard automatically adjusts to different screen sizes and orientations, providing a seamless experience across devices. Users should be able to customise their dashboard view to suit their needs. This can include the ability to add, remove, or rearrange widgets and the capability to set personal preferences for data display, such as specific metrics or time ranges.

Data Management

Modern dashboards often need to pull data from various sources, such as databases, APIs, and third-party services. Efficient data integration is crucial for ensuring that the dashboard displays up-to-date and accurate information.

Choosing the right database technology is crucial. Options include relational databases like PostgreSQL [28] or MySQL [29] and NoSQL databases like MongoDB [30] or Cassandra [31] , depending on the nature and structure of the data.

Scalability and Performance

As the number of users and the volume of data grow, the dashboard must be able to scale accordingly. Techniques like load balancing, caching, and asynchronous data loading can help manage increased demand. frontend and backend optimisations are crucial for performance. This includes minimising the size of assets like CSS, JavaScript, and images, optimising database queries, and using efficient algorithms for data processing.

Security

Secure login mechanisms and role-based access control are essential for protecting sensitive data. Users should only be able to access data and functionalities that are relevant to their role within the organisation. Data transmitted between the client and server should be encrypted to prevent interception. Additionally, sensitive data stored in databases should be encrypted at rest. Regularly auditing the dashboard for security vulnerabilities is vital. This includes keeping all software up to date and using tools to scan for weaknesses.

Technology Stack

Common technologies for building the frontend of a web dashboard include HTML, CSS, JavaScript, together known as HTLM5 [32] and frameworks like React [34] , Angular [33] , or Vue.js [35] . These technologies enable the creation of a dynamic and interactive user interface.

The backend, responsible for data processing and handling API requests, can be built using languages like Python, Node.js, Java, or Go, and frameworks like Django, Express, or Spring Boot.

4.3 Development Roadmap

The initial release of the Operational Portal is planned for M18. The detailed breakdown of the development roadmap will be described in D2.7, “HiDALGO2 Dashboard and Services”, along with all necessary tools and services that will be integrated into the portal.

Based on user feedback, the second release of the HiDALGO2 Dashboard is planned for M24, after which, a 6-month release cycle is planned until the end of the project, to be able to fulfil all growing requirements on an iterative and sustainable agile basis.

5. Component Integration

HiDALGO2's vision for realising a reliable integration and deployment of its services is to differentiate and group the project's needs into three tiers. These three tiers reflect the highly customised toolchains and workflows essential to ensure consistent and continuous component integration across several project layers, from deploying the project's website to running simulations on HPC infrastructure Even though there are specific details to take into consideration for each of the three layers, all are developed through an agreement of fundamental system design and architecture principals [41] .

5.1 CI / CD for Project-Level Web Services

5.1.1 Requirements

The web services making up the HiDALGO2 project are diverse, ranging from Identity Management to educational courses and customer support services. However, there are key requirements that the CI/CD system must ensure across all services.

- **REQ-CICD-001:** The Infrastructure behind the Web Service must be described and specified.
- **REQ-CICD-002:** All Infrastructure dedicated to a Service must be allocated through a CI/CD automation pipeline.
- **REQ-CICD-003:** Services themselves must be built and deployed through a CI/CD automation pipeline.
- **REQ-CICD-004:** Dependencies and additional sub-systems for running the above Services must be handled by the CI/CD automation pipeline.
- **REQ-CICD-005:** Separate Development and Production Instances must be provided and handled by the CI/CD pipeline.
- **REQ-CICD-006:** The setting up and deployment of Services and Infrastructure must be reliable and idempotent (replicable).

5.1.2 Design and Architecture

The internal CI/CD design for the Web Services Infrastructure must cover the requirements listed above in Section 5.1.1. To help fulfil these requirements, we have chosen Ansible [36] . This Python-based automation platform employs "playbooks", which are collections of well-organised and modular scripts written in YAML syntax to consistently deploy all web services.

We chose Ansible, over alternatives like Puppet [38] and Chef [39] , for several reasons. Chief amongst these reasons was the fact that Ansible does not introduce its own Domain Specific

Language. Instead, it uses YAML syntax to add on to the capabilities of BASH/Shell scripting [37] . Hence, all Ansible commands can be reduced to shell scripts in their basic form.

Other reasons to choose Ansible included the availability and stability of core modules of the Ansible Galaxy hub, such as roles and playbooks uploaded for free by the community, from where the HiDALGO2 project can leverage pre-existing configurations for common dependencies of its services.

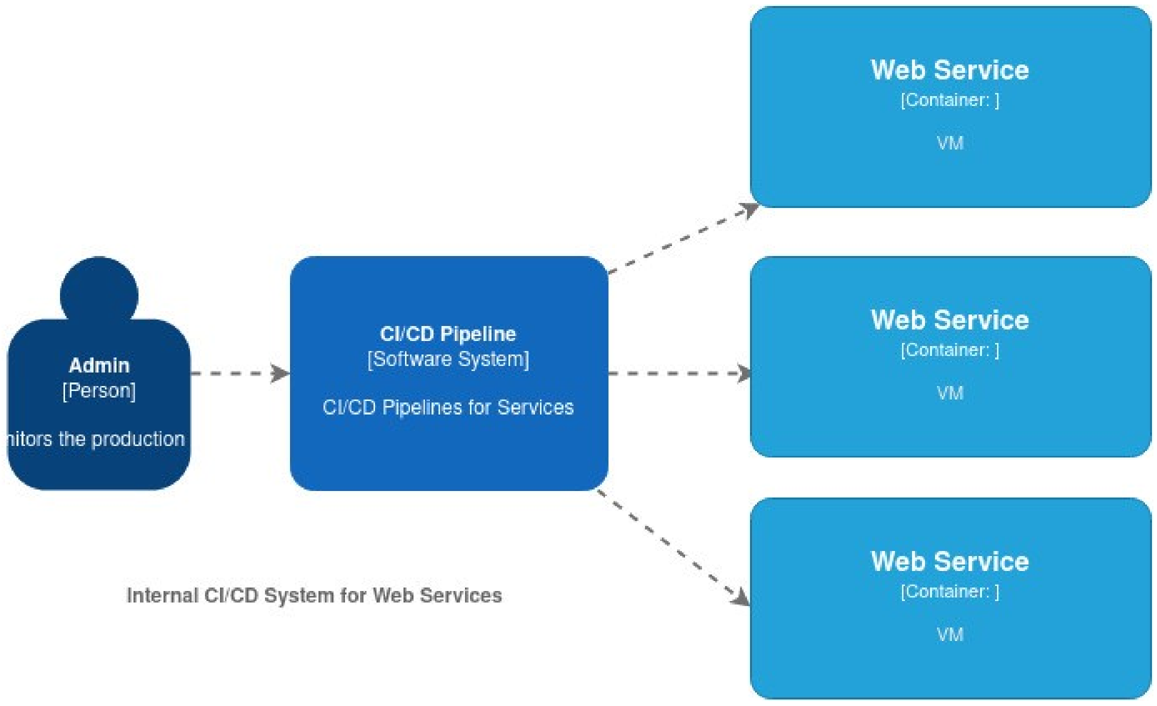


Figure 7: Overview Internal CI/CD for Web Services.

Figure 7 illustrates the main idea of the CI/CD system when it comes to Services. The pipelines are responsible for the setup and running of each individual Service in its assigned and deployed Virtual Machine, or other underlying infrastructure.

Figure 8 presents further details about the CI/CD system. Firstly, this comprises running scripts which allocate and deploy the infrastructure for the individual services, i.e. the Virtual machines described in Section 2.1, or other potential Resource Units (RUs), such as Kubernetes clusters in the future, and then secondly, a collection of scripts and Ansible playbooks for installing and setting up the main software and dependencies required by the Service.

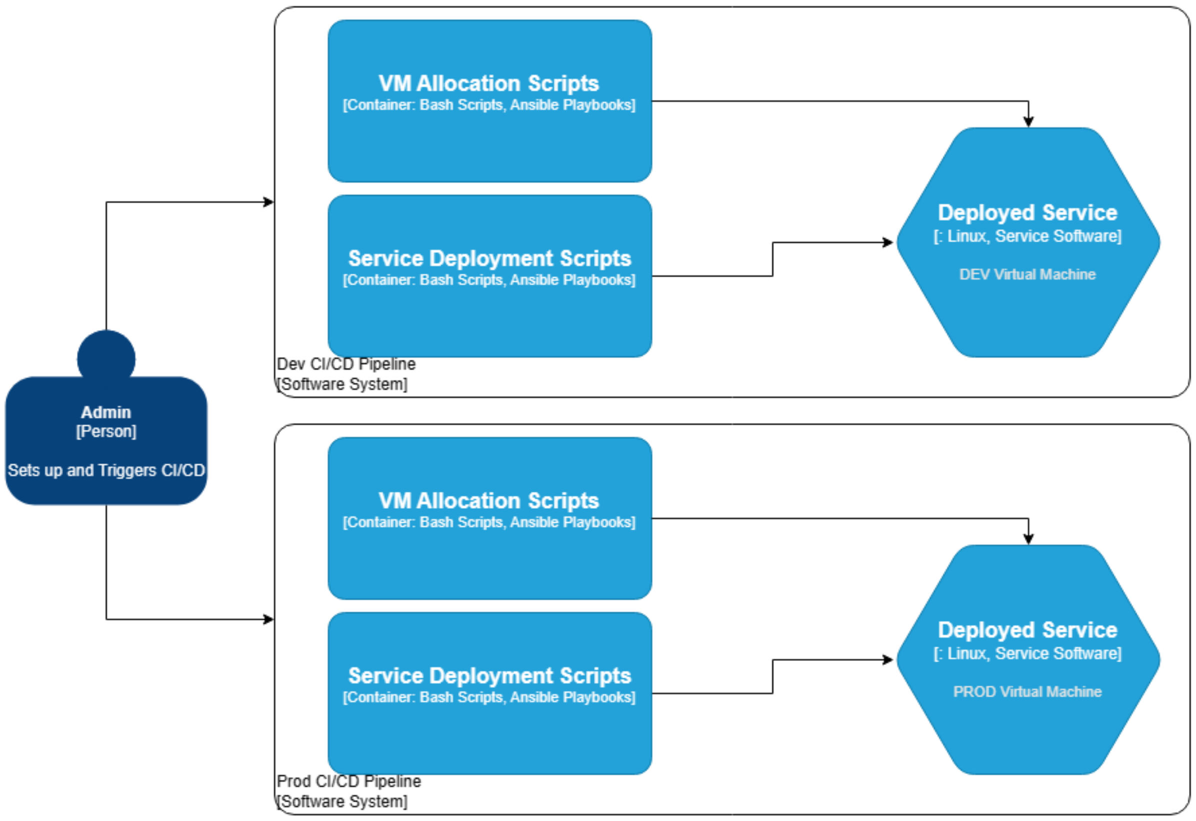

VM Allocation Scripts
[Container: Bash Scripts, Ansible Playbooks]
Service Deployment Scripts
[Container: Bash Scripts, Ansible Playbooks]
Deployed Service
[: Linux, Service Software]
DEV Virtual Machine
Dev CI/CD Pipeline
[Software System]
Admin
[Person]
Sets up and Triggers CI/CD
VM Allocation Scripts
[Container: Bash Scripts, Ansible Playbooks]
Service Deployment Scripts
[Container: Bash Scripts, Ansible Playbooks]
Deployed Service
[: Linux, Service Software]
PROD Virtual Machine
Prod CI/CD Pipeline
[Software System]

Figure 8: CI/CD Component Diagram

The web service deployment must maintain the base OS libraries to their latest and appropriate version, fetch and pull the proper codebase of the web service itself, build the service, and then launch it. To do so, several complimentary programs and services must be set up, e.g., databases, firewalls, programming language libraries, and programs like web proxies.

To follow the principles of DevOps and robust CI/CD, we have set up a multi-stage area, with two VMs/RUs dedicated to each web service, namely, Development and Production. This division in infrastructure ensures that a live system in production is not affected adversely, e.g., when an update in the main software of the service or one of its dependencies can cause problems. Instead, these changes will be tested first on the Development (Dev) instance of the service and only migrated over to Production (Prod) when the update on the Dev proved safe. An update or configuration change is considered safe, when there is a high degree of confidence, either through detailed manual verification or, where possible, through automated testing, that the update will not cause any adverse effects or outages to the Service.

5.1.3 Development Roadmap

Currently, both USTUTT and PSNC host the Infrastructure code in local repositories, and follow their own CI/CD pipelines for achieving the integration and deployment of services. The following steps will bring the HiDALGO2 Infrastructure CI/CD to the required standard:

- Pushing Code for Services by PSNC and USTUTT to shared common code repository.
- Have dedicated repositories for each service, as well repositories to provision Infrastructure resources for each service.
- Have dedicated branches for each Service, to control integration and deployment on Development and Production instances.
- Integrate a code quality component to the CI/CD pipeline, such as the linter (the code checking tool) proofs the scripts for any errors in the code syntax.
- Integrate automated testing of deployed services. This could be achieved e.g., by using the Ansible Molecule framework [40] and composing testcases for each service.
- Use Gitlab Runners in combination with Ansible and other repository scripts to carry out the Integration tests, as well as to deploy the development and production VMs.

5.2 CI / CD for Project-Level Simulation Code Deployments

5.2.1 Requirements

Although each pilot is unique, and requires different stages of building the final executable, either in the form of a binary package or more substantial container images, all share the following core requirements:

- **REQ-CICD-007:** The CI/CD system should ensure that pilots have up to date code in the project's on-premise Git repository.
- **REQ-CICD-008:** All pilot codes must be correctly versioned, to have correctly versioned builds.
- **REQ-CICD-009:** The CI/CD system should take into consideration the particular nature of each pilot's build process and store the final built image or executable on the repository, or link to it.
- **REQ-CICD-010:** The CI/CD system should have access to the HPC system, so that registered Gitlab Runners tied to particular pilot repositories can pull the built image and deploy it on the HPC system.

5.2.2 Design and Architecture

The diagram in Figure 9 shows how the CI/CD system of HiDALGO2 interacts with the workflow orchestrators to trigger an execution of the pilot after deploying and installing them on the HPC system. The main building blocks making up the CI/CD system, as shown in Figure 10, are the Bitbucket repository, where pilot use cases will push versioned tags of their codes as they are developed, and a Gitlab mirror, set up to be able to use Gitlab runners and maintain synergy with future integrations with the external EuroHPC CI/CD Pilot Platform and EuroHPC systems.

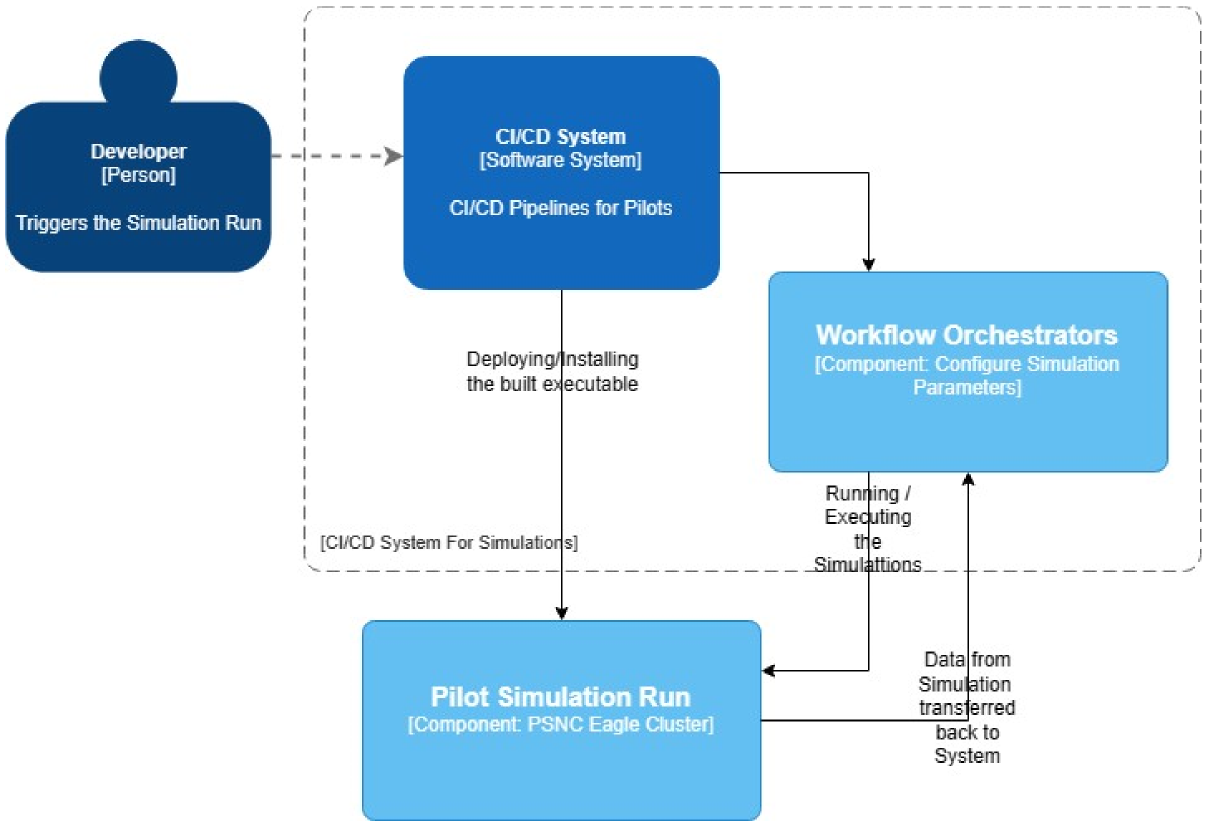

Developer
[Person]
Triggers the Simulation Run
CI/CD System
[Software System]
CI/CD Pipelines for Pilots
Workflow Orchestrators
[Component: Configure Simulation Parameters]
Deploying/Installing the built executable
[CI/CD System For Simulations]
Running / Executing the Simulattions
Pilot Simulation Run
[Component: PSNC Eagle Cluster]
Data from Simulation transferred back to System

Figure 9: System Context Diagram for Internal CI/CD for Pilots

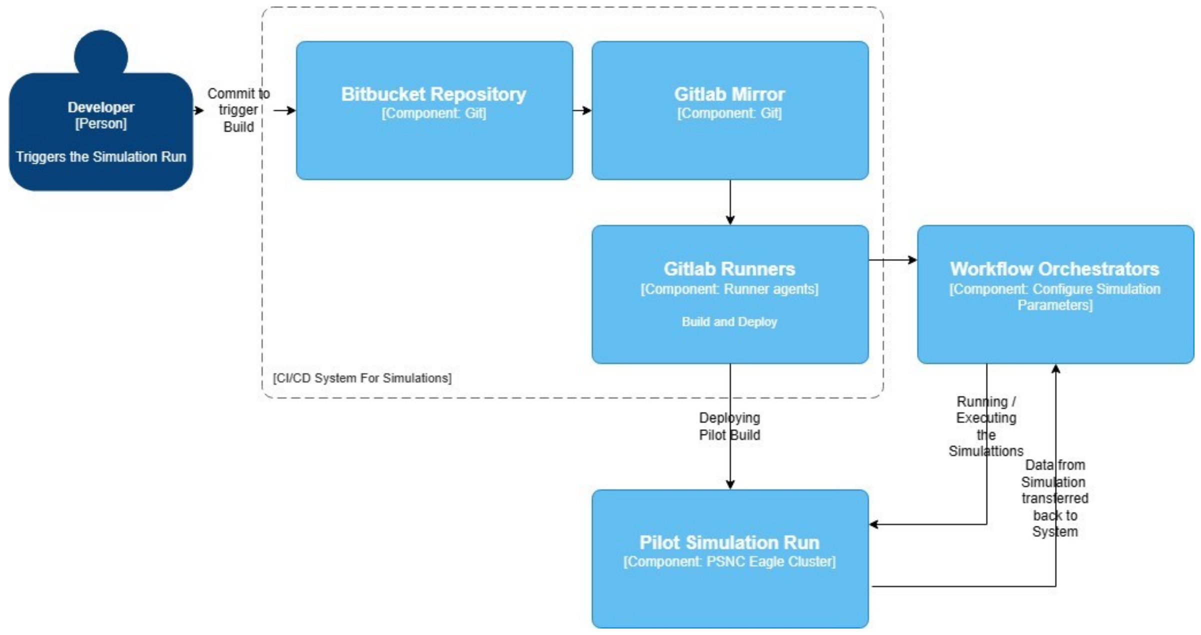

Developer
[Person]
Triggers the Simulation Run
Commit to trigger Build
Bitbucket Repository
[Component: Git]
Gitlab Mirror
[Component: Git]
Gitlab Runners
[Component: Runner agents]
Build and Deploy
Workflow Orchestrators
[Component: Configure Simulation Parameters]
[CI/CD System For Simulations]
Deploying Pilot Build
Running / Executing the Simulattions
Data from Simulation transferred back to System
Pilot Simulation Run
[Component: PSNC Eagle Cluster]

Figure 10: Container Diagram for Internal CI/CD for Pilots

This push will cause a similar change in the Gitlab mirror of the Bitbucket repository. After registering this trigger, Gitlab runners linked to the particular repository of the pilot use case will initiate the build and deploy pipeline. The build component of this pipeline is individual to each pilot use case. Figure 11 shows the particular build pipeline of the Urban Building pilot. The container is built according to the local CI from UNISTRA, after which a final container image is pushed to the Container Registry. From there, the registered Gitlab runner will fetch the container, connect to the HPC system, in this case, the PSNC Eagle, and deploy the container. Optionally, the runner will also trigger a run of the simulation through the REST API provided by the HiDALGO2 workflow orchestrators, either MathSO or QCG.

5.2.3 Development Roadmap

Currently, the state of the CI/CD for pilot simulations is limited to storing pilot codes to the Bitbucket instance for HiDALGO2. Next steps to realise the full potential of the CI/CD system are:

- Setting up of Gitlab Mirror (Month 12)
- Setting up Runners on HPC Systems (Month 13)
- Testing all pilots with deployments (Month 13)
- Testing all pilot executions through Gitlab Runners and Workflow Orchestrators. (Month 14)

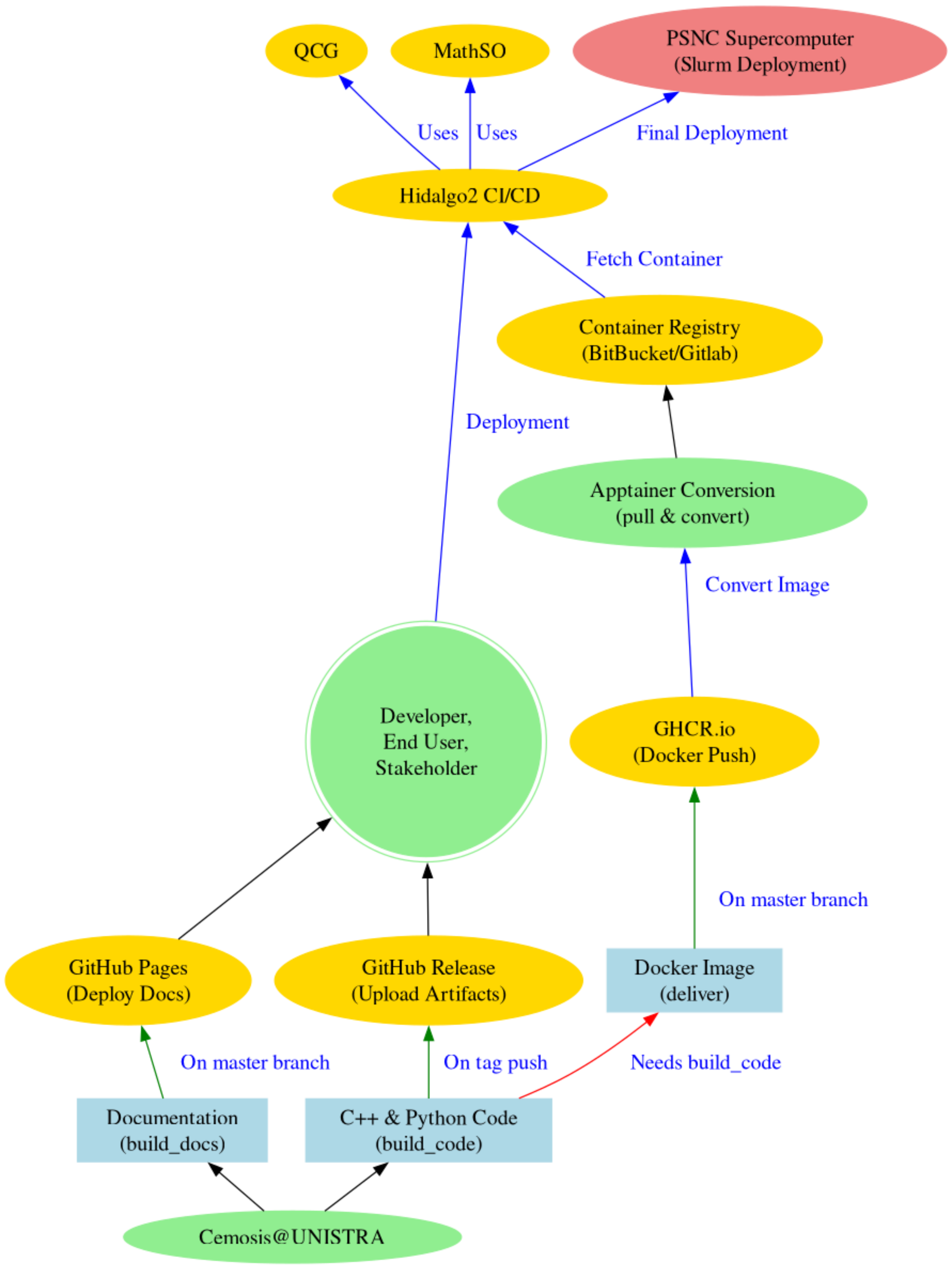

QCG
MathSO
PSNC Supercomputer (Slurm Deployment)
Uses
Uses
Final Deployment
Hidalgo2 CI/CD
Fetch Container
Container Registry (BitBucket/Gitlab)
Deployment
Apptainer Conversion (pull & convert)
Convert Image
Developer, End User, Stakeholder
GHCR.io (Docker Push)
On master branch
GitHub Pages (Deploy Docs)
GitHub Release (Upload Artifacts)
Docker Image (deliver)
On master branch
On tag push
Needs build_code
Documentation (build_docs)
C++ & Python Code (build_code)
Cemosis@UNISTRA

Figure 11: Component Diagram for Urban Buildings Pilot Deployment

5.3 CI/CD for HiDALGO2 to EuroHPC Level

5.3.1 Requirements

The requirements for the external CI/CD system build on the requirements and achieved results of the internal CI/CD system for pilots.

- **REQ-CICD-011:** All pilot codes must have correctly versioned tags pushed to HiDALGO2 repository.
- **REQ-CICD-012:** Mirror set up so all changes in pilot codes in HiDALGO2 repository are reflected in the Central Repository provided by the EuroHPC CI/CD Pilot Platform.
- **REQ-CICD-013:** HiDALGO2 CI must build appropriate packages/artifacts from these code repositories.
- **REQ-CICD-014:** Gitlab Runners from EuroHPC sites must be able to fetch correctly built packages from the Central Repository, to be able to deploy them on the particular EuroHPC machine.

5.3.2 Design and Architecture

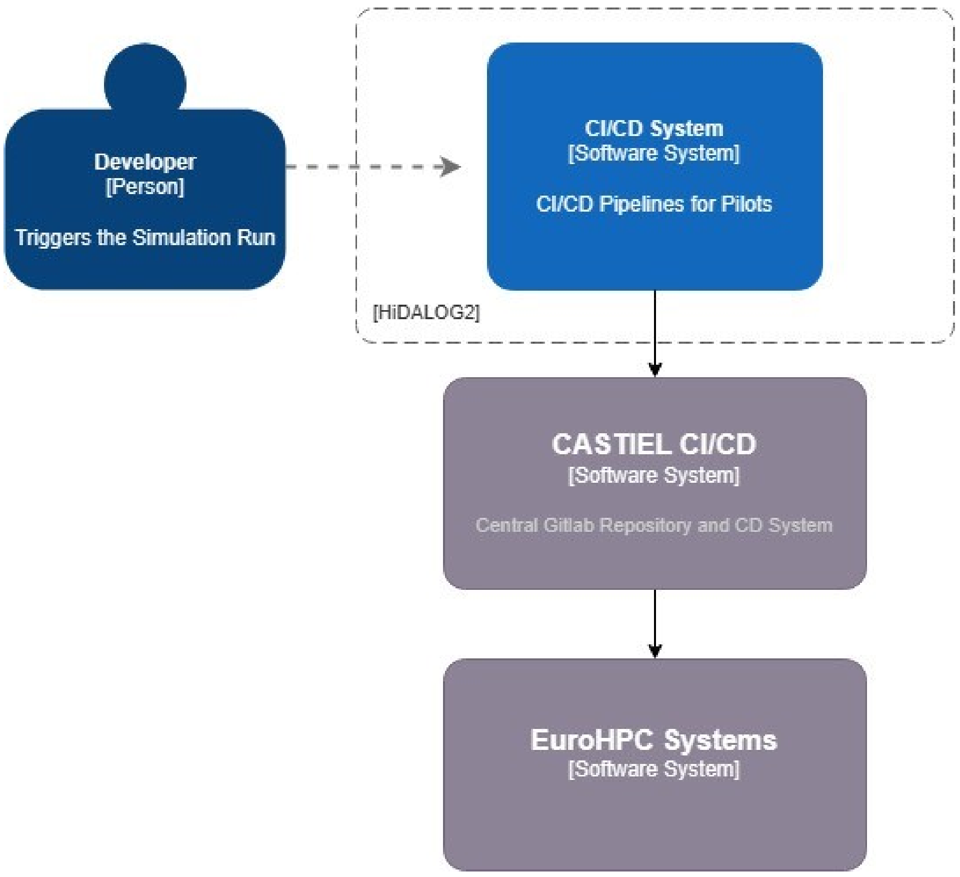


Figure 12: System Context Diagram for External CI/CD for Pilots

The external CI/CD system takes the CI component from the internal CI/CD System for HiDALGO2 pilots and fuses it with the EuroHPC CI/CD Pilot Platform. This is achieved with the help of the internal project Gitlab mirror, which enables a smooth mirroring of the HiDALGO2 pilot codes to the central EuroHPC Gitlab instance. From the HiDALGO2 side, a developer executes the trigger for the CI/CD system to run, resulting in an integrated final built package or image which is placed in a location that the EuroHPC Gitlab runners can access, pull, and finally deploy and install on the desired EuroHPC system. Figure 13 shows this process in a Level 2 / Container Diagram.

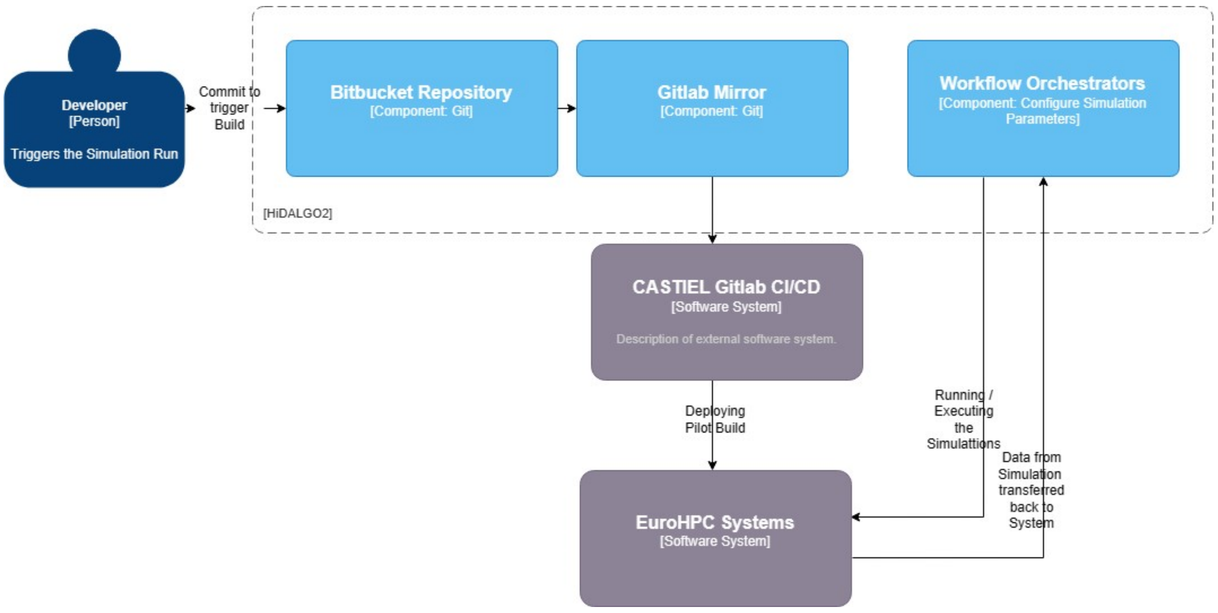


Figure 13: Component Diagram for External CI/CD System

An extension of the complete built pipeline for the Urban Building pilot is shown in Figure 14, with the “Final Deployment” steps to the EuroHPC systems carried out by the EuroHPC CI/CD Pilot Platform.

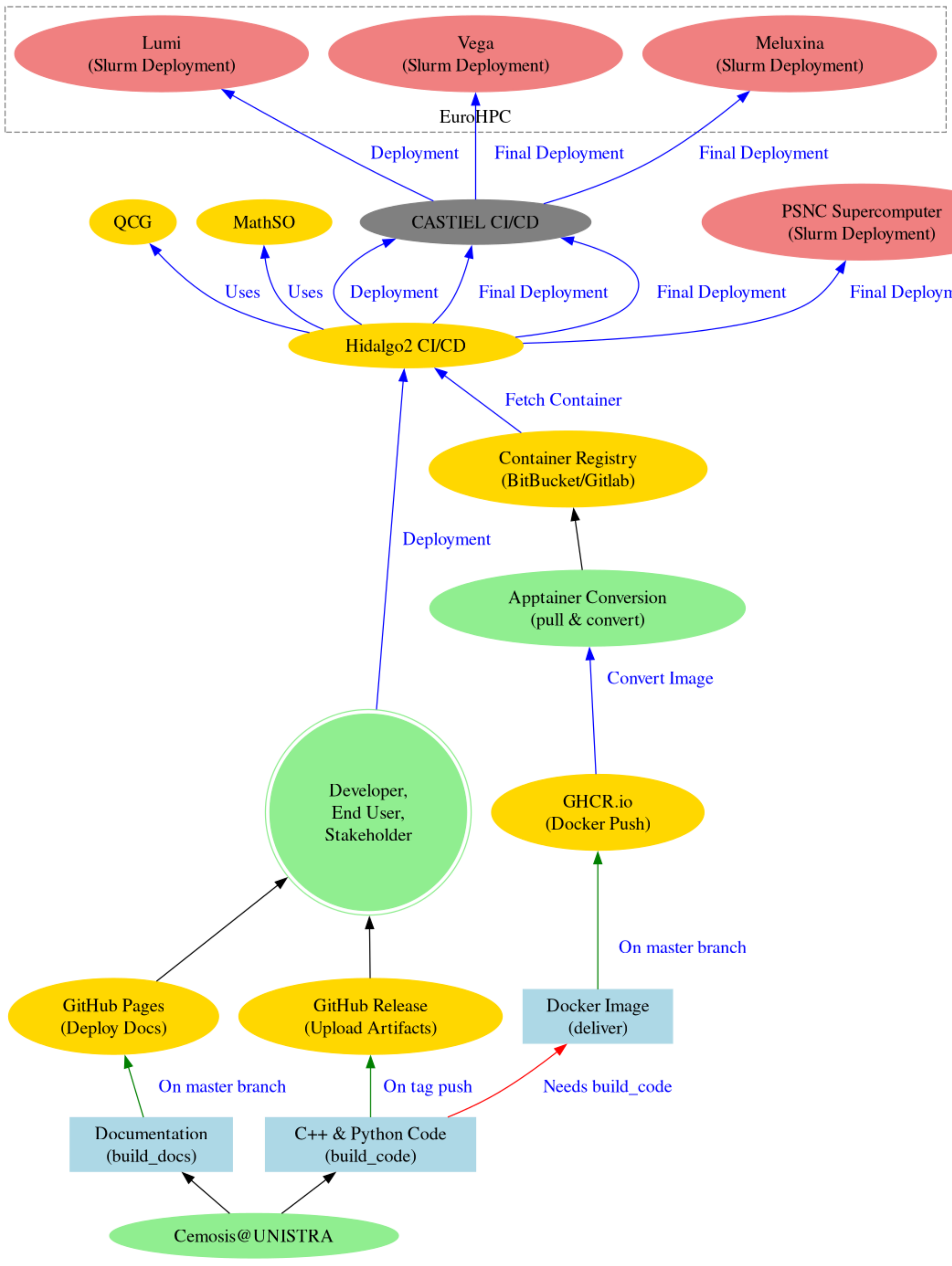


Figure 14: Component Diagram for Urban Building Pilot External Deployment

5.3.3 Development Roadmap

The next steps, after establishing and testing the internal CI/CD system for HiDALGO2, is establishing the connection between the HiDALGO2 CI/CD and the EuroHPC CI/CD Pilot Platform:

- Set up Gitlab Mirror instance in HiDALGO2 to mirror repositories in the central Gitlab instance, with correct access permissions (Month 12).
- Ensure that the final built executables/images/containers are in the correct place in the central Gitlab instance, with accessible links to any external container registries (Month 13).
- Work with Gitlab Runners implemented on the EuroHPC sites to correctly access, pull and deploy pilot executables/images (Month 14).

6. Conclusions

This deliverable addresses the challenges of establishing a sound foundation for the technical infrastructure of the HiDALOG2 project by building up a coherent system design that describes the core components of the HiDALOG2 ecosystem, and lays the groundwork for continuing the development and improvement of the pilot use cases.

The infrastructure is strengthened by setting up and maintaining various web services that provide support to developers and internal project members, as well as lay a platform for future external users and customers. This includes a brief look ahead at the envisioned HiDALOG2 dashboard, and an in-depth look at the advancement and integration of HiDALGO2's workflow orchestrators, MathSO and QCG. Requirements for the workflow orchestrators have been carefully collected from pilot partners, to ensure that the development of the workflow orchestrators aligns as closely as possible to the simulation use cases and project goals.

HiDALGO2 tackles a key challenge and goal set by the EuroHPC JU through this deliverable by developing a clear path to ease the deployment of its simulation use cases onto EuroHPC infrastructure. All partners collaborate closely to identify major requirements, strategies, technologies and workflows to set up efficient and reliable CI/CD pipelines, resulting in three tiers of component integration, building up from CI/CD for pilot codes and web services, testing out code deployment on internal project infrastructure, and leading up to a seamless connection between the HIDALGO2 CoE and deployment on the wider EuroHPC network of systems.

HiDALGO2 will continue working on achieving its technical infrastructure and component integration goals by following the roadmaps developed through this deliverable, and report on the advancement of its efforts in the follow up deliverables, D2.5 (M24) and D2.6 (M35).

References

[1] (2023), *Homepage of HiDALGO2 Project*, https://www.hidalgo2.eu, [retrieved: 2023-11-27].

[2] (2023), *About the EuroHPC JU*, https://eurohpc-ju.europa.eu/about_en, [retrieved: 2023-11-27].

[3] (2023), *Homepage of Jupyter Notebooks Project*, https://jupyter.org/, [retrieved: 2023-11-27]

[4] (2023), *Overview of the Slurm Workload Manager*, https://slurm.schedmd.com/overview.html, [retrieved: 2023-11-27].

[5] (2023), *Description of EuroHPC Machines*, https://eurohpc-ju.europa.eu/supercomputers/our-supercomputers_en, [retrieved: 2023-11-27]

[6] (2023), *Ubuntu Linux Homepage*, https://ubuntu.org/, [retrieved: 2023-11-27]

[7] (2023), *Nginx Homepage*, https://nginx.com/, [retrieved: 2023-11-27]

[8] (2023), *PHP Project Homepage*, https://php.net/, [retrieved: 2023-11-27]

[9] (2023), *Wordpress Organisation*, https://wordpress.org/, [retrieved: 2023-11-27]

[10] (2023), *CKAN Project*, https://ckan.org/, [retrieved: 2023-11-27]

[11] (2023), *Apache Hadoop Project Homepage*, https://hadoop.apache.org/, [retrieved: 2023-11-27]

[12] (2023), *Elasticsearch*, https://www.elastic.co/elasticsearch/, [retrieved: 2023-11-27]

[13] (2023), *Apache Kafka Project*, https://kafka.apache.org, [retrieved: 2023-11-27]

[14] (2023), *Wiki-js Homepage*, https://js.wiki/, [retrieved: 2023-11-27]

[15] (2023), *Keycloak Project*, https://www.keycloak.org/, [retrieved: 2023-11-27]

[16] (2023), *Askbot Project*, https://askbot.com/, [retrieved: 2023-11-27]

[17] (2023), *Zabbix Homepage*, https://www.zabbix.com/, [retrieved: 2023-11-27]

[18] (2023), *Zammad*, https://zammad.com/, [retrieved: 2023-11-27]

[19] (2023), *Moodle Homepage*, https://moodle.com/, [retrieved: 2023-11-27]

[20] (2023), *OpenProject Homepage*, https://openproject.org/, [retrieved: 2023-11-27]

[21] (2023), C4 Modelling System, https://c4model.info/, [retrieved 11.27.2023]

[22] (2023), CASTIEL2 and EuroCC, https://www.eurocc-access.eu/about-us/the-projects/, [retrieved 11.27.2023]

[23] (2023), *COVISE Homepage*, https://www.hlrs.de/solutions/types-of-computing/visualisation/covise, [retrieved: 2023-11-27]

[24] (2008) I Sfiligoi 2008 *J. Phys.: Conf. Ser.* **119** 062044

[25] (2023) HiDALGO2 report. D5.3 Research Advancements for the Pilots (M10). 2023

[26] (2024, to appear), HiDALGO2 report. D4.6 Visualisations for Global Challenges (M17).

[27] (Fowler M. (2002), Patterns of Enterprise Application Architecture, Addison-Wesley Professional, Berkeley

[28] (2023), PostgreSQL Homepage, https://www.postgresql.org/, [retrieved 11.27.2023]

[29] (2023), MySQL Homepage, https://mysql.com, [retrieved 11.27.2023]

[30] (2023), MongoDB Homepage, https://mongodb.com/, [retrieved 11.27.2023]

[31] (2023), Apache Cassandra Project, https://cassandra.apache.org/, [retrieved 11.27.2023]

[32] (2023), HTML5, https://developer.mozilla.org/en-US/docs/Glossary/HTML5 [retrieved 11.27.2023]

[33] (2023), Angular, https://angular.io/, [retrieved 11.27.2023]

[34] (2023), React, https://react.dev/, [retrieved 11.27.2023]

[35] (2023), Vue, https://vuejs.org/, [retrieved 11.27.2023]

[36] (2023), Ansible, https://ansible.com/, [retrieved 11.27.2023]

[37] (2023), Bash, https://www.gnu.org/software/bash/, [retrieved 11.27.2023]

[38] (2023), Puppet, https://puppet.com/, [retrieved 11.27.2023]

[39] (2023), Chef, https://chef.io/, [retrieved 11.27.2023]

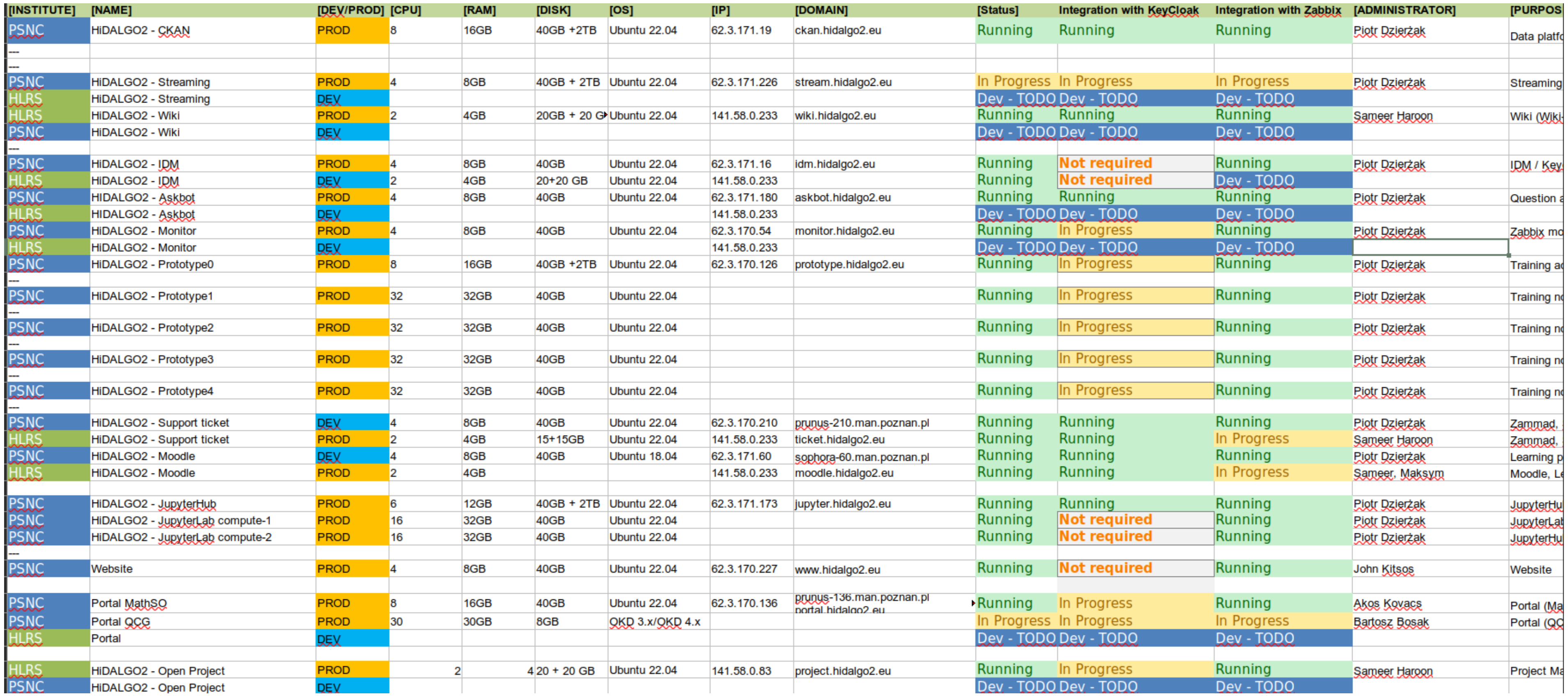
[40] (2023), Ansible Molecule Framework, https://github.com/ansible/molecule, [retrieved 11.27.2023]

[41] Fowler M. (2019), Software Architecture Guide, https://martinfowler.com/architecture/, [retrieved 11.27.2023]

[42] (2023), Fedora Linux OKD environment, https://docs.okd.io/3.11/install/prerequisites.html, [retrieved 12.10.2023]

Annexes

Annex 1: Services Matrix



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

PSNC	HiDALGO2 - Streaming	PROD	4	8GB	40GB + 2TB	Ubuntu 22.04	62.3.171.226	stream.hidalgo2.eu	In Progress	In Progress	In Progress	Piotr Dzierżak	Streaming
HLRS	HiDALGO2 - Streaming	DEV							Dev - TODO	Dev - TODO	Dev - TODO		
HLRS	HiDALGO2 - Wiki	PROD	2	4GB	20GB + 20 G	Ubuntu 22.04	141.58.0.233	wiki.hidalgo2.eu	Running	Running	Running	Sameer Haroon	Wiki (Wiki-
PSNC	HiDALGO2 - Wiki	DEV							Dev - TODO	Dev - TODO	Dev - TODO		

PSNC	HiDALGO2 - IDM	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.171.16	idm.hidalgo2.eu	Running	Not required	Running	Piotr Dzierżak	IDM / Key
HLRS	HiDALGO2 - IDM	DEV	2	4GB	20+20 GB	Ubuntu 22.04	141.58.0.233		Running	Not required	Dev - TODO		
PSNC	HIDALGO2 - Askbot	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.171.180	askbot.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	Question a
HLRS	HIDALGO2 - Askbot	DEV					141.58.0.233		Dev - TODO	Dev - TODO	Dev - TODO		
PSNC	HiDALGO2 - Monitor	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.170.54	monitor.hidalgo2.eu	Running	In Progress	Running	Piotr Dzierżak	Zabbix mo
HLRS	HiDALGO2 - Monitor	DEV					141.58.0.233		Dev - TODO	Dev - TODO	Dev - TODO		
PSNC	HiDALGO2 - Prototype0	PROD	8	16GB	40GB +2TB	Ubuntu 22.04	62.3.170.126	prototype.hidalgo2.eu	Running	In Progress	Running	Piotr Dzierżak	Training ad
---													
PSNC	HiDALGO2 - Prototype1	PROD	32	32GB	40GB	Ubuntu 22.04			Running	In Progress	Running	Piotr Dzierżak	Training no
---													
PSNC	HiDALGO2 - Prototype2	PROD	32	32GB	40GB	Ubuntu 22.04			Running	In Progress	Running	Piotr Dzierżak	Training no
---													
PSNC	HiDALGO2 - Prototype3	PROD	32	32GB	40GB	Ubuntu 22.04			Running	In Progress	Running	Piotr Dzierżak	Training no
---													
PSNC	HiDALGO2 - Prototype4	PROD	32	32GB	40GB	Ubuntu 22.04			Running	In Progress	Running	Piotr Dzierżak	Training no
---													
PSNC	HiDALGO2 - Support ticket	DEV	4	8GB	40GB	Ubuntu 22.04	62.3.170.210	prunus-210.man.poznan.pl	Running	Running	Running	Piotr Dzierżak	Zammad,
HLRS	HiDALGO2 - Support ticket	PROD	2	4GB	15+15GB	Ubuntu 22.04	141.58.0.233	ticket.hidalgo2.eu	Running	Running	In Progress	Sameer Haroon	Zammad,
PSNC	HiDALGO2 - Moodle	DEV	4	8GB	40GB	Ubuntu 18.04	62.3.171.60	sophora-60.man.poznan.pl	Running	Running	Running	Piotr Dzierżak	Learning p
HLRS	HiDALGO2 - Moodle	PROD	2	4GB			141.58.0.233	moodle.hidalgo2.eu	Running	Running	In Progress	Sameer, Maksym	Moodle, Le
PSNC	HiDALGO2 - JupyterHub	PROD	6	12GB	40GB + 2TB	Ubuntu 22.04	62.3.171.173	jupyter.hidalgo2.eu	Running	Running	Running	Piotr Dzierżak	JupyterHu
PSNC	HiDALGO2 - JupyterLab compute-1	PROD	16	32GB	40GB	Ubuntu 22.04			Running	Not required	Running	Piotr Dzierżak	JupyterLal
PSNC	HiDALGO2 - JupyterLab compute-2	PROD	16	32GB	40GB	Ubuntu 22.04			Running	Not required	Running	Piotr Dzierżak	JupyterHu
---													
PSNC	Website	PROD	4	8GB	40GB	Ubuntu 22.04	62.3.170.227	www.hidalgo2.eu	Running	Not required	Running	John Kitsos	Website
PSNC	Portal MathSO	PROD	8	16GB	40GB	Ubuntu 22.04	62.3.170.136	prunus-136.man.poznan.pl portal.hidalgo2.eu	Running	In Progress	Running	Akos Kovacs	Portal (Ma
PSNC	Portal QCG	PROD	30	30GB	8GB	QKD 3.x/QKD 4.x			In Progress	In Progress	In Progress	Bartosz Bosak	Portal (QC
HLRS	Portal	DEV							Dev - TODO	Dev - TODO	Dev - TODO		
HLRS	HiDALGO2 - Open Project	PROD	2	4	20 + 20 GB	Ubuntu 22.04	141.58.0.83	project.hidalgo2.eu	Running	In Progress	Running	Sameer Haroon	Project Ma
PSNC	HiDALGO2 - Open Project	DEV							Dev - TODO	Dev - TODO	Dev - TODO		

Figure 15: Services Matrix

Annex 2: Requirements for Workflow Orchestration

Table 19: Requirements for Workflow Orchestration

ID#	Title	Description	Type	Priority	Risk	Required By	Responsible	Dependencies/ Relationships	Pilots
REQ_WFO_1	Secured DMS with OAuth2 token authentication	The OAuth2/OIDC enables SSO mechanism, which is required by the HiDALGO2 environment.	F	High	High	PSNC	ATOS		ALL
REQ_WFO_2	DMS REST API for data transfer	WFOs should be able to request, through a REST API, to the HiDALGO2 DMS to get/put datasets from/to the HiDALGO2 storage into/from the target HPC system, before and after the job execution (i.e., job inputs and outputs). The goal is to enable staging-in of input data and staging-out output data directly from a web portal of a workflow manager.	F	High	High	PSNC	ATOS	REQ_WFO_1	ALL
REQ_WFO_3	Transfer external datasets (from Web providers) as inputs of job execution in target HPC clusters	The execution of pilot's simulations as jobs in target HPC cluster requires of input data sets that should be transferred from Cloud providers (e.g. Copernicus) to the HPC clusters, before job execution	F	High	Low	MTG	PSNC/SZE		Wildfires
REQ_WFO_4	Select target HPC cluster for simulation execution	Simulation's owner can select the target HPC system where the simulation (i.e. workflow job) will be submitted to	F	Medium	Medium	MTG	PSNC/SZE		Wildfires
REQ_WFO_5	Visualisation of simulation output	Upon the finalisation of the submitted simulation in a selected target HPC cluster, the simulation output should be collected into the selected HiDALGO2 storage. Data visualisation should be accessible from the WFO frontend.	F	High	Medium	MTG	PSNC/SZE	REQ_WFO_2, REQ_WFO_3	Wildfires

REQ_WFO_6	Ensemble definition and execution	Definition and execution of simulation ensembles	F	Medium	Medium	MTG	PSNC/SZE		Wildfires
REQ_WFO_7	Data analysis for simulation results	From WFO frontend, data analytics should be available for the analysis of simulation results, supporting the selection of specific results for evaluation.	F	High	Medium	MTG	PSNC/SZE	REQ_WFO_5	Wildfires
REQ_WFO_8	Support for coupling scenarios	WFOs should support the execution of coupling scenarios that involves the sequential execution of multiple simulations from diverse application pilots. Therefore, it is required that outputs of one simulation are made available as inputs of the next one within the same or another HPC cluster were following simulations are submitted to.	F	High	Medium	MTG	PSNC/SZE		Wildfires
REQ_WFO_9	Archive jobs	The information about terminated workflows should be available in a separate page for historic workflow information (e.g. job id, data location, logs)	F	Medium	Low	SZE	PSNC/SZE		Urban Air
REQ_WFO_10	Automatic HPC selection	Based on User accounts, the portal needs to be able to select the optimal HPC cluster for the job (GPU intense, CPU intense, MEM intense)	F	Medium	Low	SZE	PSNC/SZE	REQ_WFO_4	Urban Air
REQ_WFO_11	Container Integration	The system should integrate with container technologies like Apptainer for packaging and deploying applications in HPC environments.	F	High	Medium	UNISTRA	PSNC/SZE	REQ_WFO_23	Urban Buildings
REQ_WFO_12	Restart from Step	The ability to restart a workflow from a specific step or checkpoint, which is now also applicable to containerised workflows.	F	Medium	Low	UNISTRA	PSNC/SZE		Urban Buildings
REQ_WFO_13	Workflow Reusability	Support for defining reusable workflows, including those that are containerised, which can be called from other workflows.	F	Medium	Low	UNISTRA	PSNC/SZE	REQ_WFO_14	Urban Buildings

REQ_WFO_14	Workflow and Container Registry	A registry system for managing both workflows and container images, facilitating version control and reuse.	F	High	Low	UNISTRA	PSNC/SZE		Urban Buildings
REQ_WFO_15	Parameter Passing for Containers	Enabling the passing of parameters into workflows that initiate containerised tasks.	F	High	High	UNISTRA	PSNC/SZE	REQ_WFO_16	Urban Buildings
REQ_WFO_16	Resource Allocation for Containers	Managing resources for containerised tasks, especially when interfacing with HPC systems.	F	Medium	Medium	UNISTRA	PSNC/SZE	REQ_WFO_15	Urban Buildings
REQ_WFO_17	HPC and Container Networking	Ensuring containerised workflows can communicate effectively within HPC environments (MPI, GPU/Nvidia, GPU/ AMD…,)	F	High	Medium	UNISTRA	PSNC/SZE		Urban Buildings
REQ_WFO_18	Performance with Containers	The system should perform well when orchestrating containerised workflows, particularly in HPC contexts.	NF	Medium	Medium	UNISTRA	PSNC/SZE	REQ_WFO_17, REQ_WFO_22	Urban Buildings
REQ_WFO_19	Scalability of Containerised Workflows	Ability to scale containerised workflows efficiently across HPC resources.	NF	Medium	Medium	UNISTRA	PSNC/SZE		Urban Buildings
REQ_WFO_20	Reliability of Container Orchestration	Ensure container orchestration is reliable, maintaining workflow integrity upon restarts and when scaling.	NF	Medium	Medium	UNISTRA	PSNC/SZE	REQ_WFO_12, REQ_WFO_19	Urban Buildings
REQ_WFO_21	Security in Container Management	Adhering to security protocols for managing containers, crucial in HPC environments dealing with sensitive data.	NF	Low	Medium	UNISTRA	PSNC/SZE	REQ_WFO_1	Urban Buildings

Annex 3: Roadmap for Workflow Orchestration

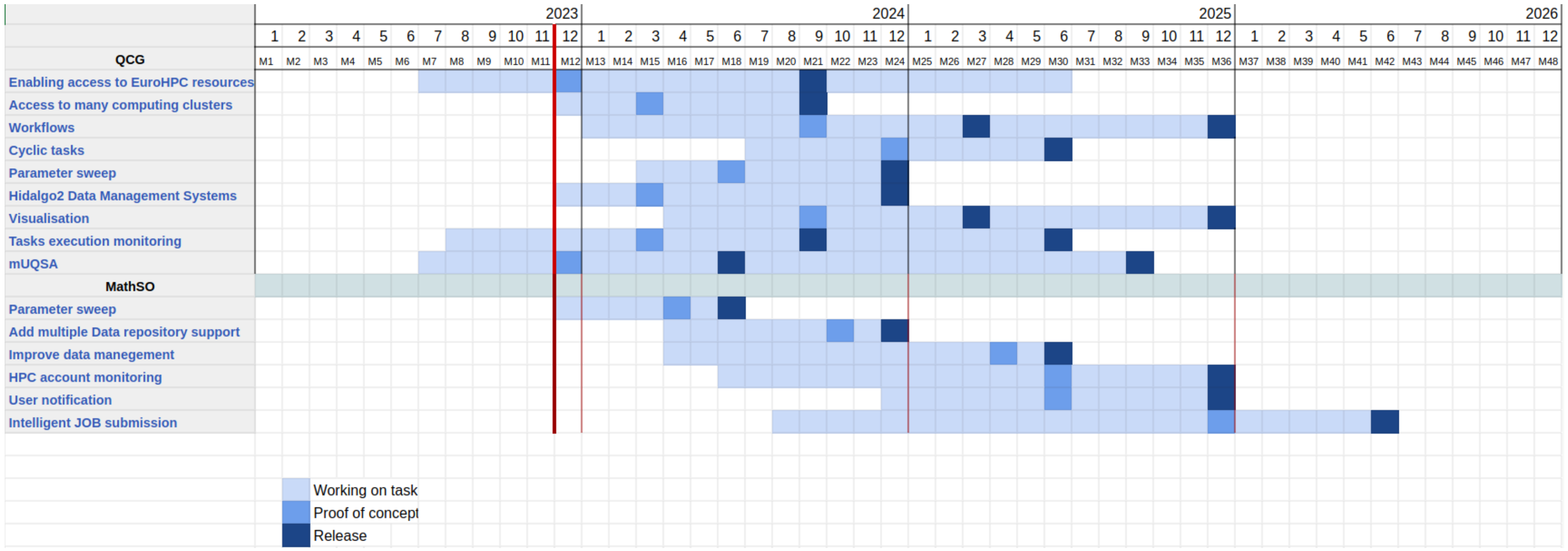


Figure 16: Gantt chart to show the development roadmap for the Workflow Orchestrators

Annex 4: Requirements for CI/CD

Table 20: Overview of CI/CD Requirements

Requirement Code	Description
REQ-CICD-001	The Infrastructure behind the Web Service must be described and specified.
REQ-CICD-002	All Infrastructure dedicated to a Service must be allocated through a CI/CD automation pipeline.
REQ-CICD-003	Services must be built and deployed through a CI/CD automation pipeline.
REQ-CICD-004	The CI/CD automation pipeline must handle dependencies and additional sub-systems for running the above Services.
REQ-CICD-005	Separate Development and Production Instances must be provided and handled by the CI/CD pipeline.
REQ-CICD-006	The setting up and deployment of Services and Infrastructure must be reliable and idempotent (replicable).
REQ-CICD-007	The CI/CD system should ensure that pilots have up-to-date code in the project’s on-premise Git repository.
REQ-CICD-008	All pilot codes must be correctly versioned to have correctly versioned builds.
REQ-CICD-009	The CI/CD system should consider the particular nature of each pilot’s build process and store the final built image or executable on the repository, or link to it.
REQ-CICD-010	The CI/CD system should have access to the HPC system, so that registered Gitlab Runners tied to particular pilot repositories can pull the built image and deploy it on the HPC system.
REQ-CICD-011	All pilot codes must have correctly versioned tags pushed to HiDALGO2 repository.
REQ-CICD-012	Mirror set up so all changes in pilot codes in HiDALGO2 repository are reflected in the Central Repository of the EuroHPC CI/CD Pilot Platform.
REQ-CICD-013	HiDALGO2 CI must build appropriate packages/artefacts from these code repositories.
REQ-CICD-014	Gitlab Runners from EuroHPC sites must be able to fetch correctly built packages from the Central Repository to deploy them on the particular EuroHPC machine.