Future Generation Computer Systems 185 (2026) 108672

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

AI4EOSC: A federated cloud platform for Artificial Intelligence in scientific research ☆

Ignacio Heredia [a,1], Álvaro López García [a,*,1], Fernando Aguilar Gómez [a], Diego Aguirre [b], Caterina Alarcón Marín [b], Khadijeh Alibabaei [g], Lisana Berberi [g], Miguel Caballer [b], Amanda Calatrava [b], Pedro Castro [f], Alessandro Costantini [e], Mario David [h], Jaime Díez [f], Stefan Dlugolinsky [c], Giacinto Donvito [e], Leonhard Duda [g], Borja Esteban Sanchis [g], Saúl Fernandez Tobías [a], Andrés Heredia Canales [a], Valentin Kozlov [g], Sergio Langarita [b], João Machado [j], Daniel San Martín [f], Germán Moltó [b], Giang Nguyen [d], Marta Obregón Ruiz [a], Marcin Płóciennik [i], Susana Rebolledo Ruiz [a], Vicente Rodriguez [b], Judith Sáinz-Pardo Díaz [a], Martin Šeleng [c], Viet Tran [c]

[a] Instituto de Física de Cantabria (IFCA), CSIC-UC, Avda. los Castros s/n, Santander, 39006, Cantabria, Spain
[b] Instituto de Instrumentación para Imagen Molecular (I3M), Centro Mixto CSIC - Universitat Politècnica de València (UPV), Camino de Vera s/n, Valencia, 46022, Valencia, Spain
[c] Institute of Informatics, Slovak Academy of Sciences (IISAS), Dúbravská cesta 9, Bratislava, 84507, Slovakia
[d] Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 2, Bratislava, 84216, Slovakia
[e] Istituto Nazionale di Fisica Nucleare (INFN), Via Enrico Fermi 40, Frascati, 00044, Roma, Italy
[f] Predictia Intelligent Data Solutions, Fernando de los Ríos 48, Santander, 39006, Cantabria, Spain
[g] Karlsruher Institut für Technologie, Kaiserstraße 12, Karlsruhe, 76131, Germany
[h] Laboratório de Instrumentação e Física Experimental de Partículas, Av. Prof. Gama Pinto 2, Lisboa, 1649-003, Portugal
[i] Poznańskie Centrum Superkomputerowo Sieciowe, Jana Pawła II 10, Poznan, 61-139, Poland
[j] Centro Nacional de Computação Avançada (CNCA), Avenida do Brasil, 101, Lisboa, 1700-066, Portugal

ARTICLE INFO

Keywords:
Cloud platform
Artificial Intelligence
Machine Learning
European Open Science Cloud
MLOps
Federated Learning
Provenance tracking
W3C PROV
Open science infrastructure

ABSTRACT

The rapid growth of Artificial Intelligence and Machine Learning in scientific research has highlighted a gap between industry-standard machine learning operations (MLOps) tools and platforms and the unique requirements of modern and Open Science, particularly regarding the FAIR (Findable, Accessible, Interoperable, and Reusable) principles. This paper presents AI4EOSC, a federated, open-source platform designed to operationalize the full AI/ML life-cycle within the European Open Science Cloud (EOSC) ecosystem. Our methodology tackles the fragmentation of distributed research infrastructures by integrating a modular and distributed architecture comprising an AI development platform, a serverless AI-as-a-Service layer, and a federated orchestration model that is able to integrate heterogeneous computing and storage resources from distributed e-infrastructures. AI4EOSC also introduces a "FAIR-by-design" approach that enforces metadata standardization (via MLDCAT-AP) and W3C PROV-compliant provenance tracking through a platform-integrated CI/CD pipeline. The added value of AI4EOSC is demonstrated through the delivery of a diverse set of community installations, which show consistent and seamless deployment across heterogeneous cloud providers. These installations are validated by a set of scientific cases, showing how our work reduces the manual burden on researchers while ensuring high levels of reproducibility and interoperability and providing a unified environment for the development, training, and production of AI/ML models in the EOSC.

☆ This article is part of a Special issue entitled: 'MLOps' published in Future Generation Computer Systems.

* Corresponding author.
E-mail addresses: aloga@ifca.unican.es, aloga@ifca.es (Á. López García).

[1] First and second authors contributed equally to this work. Other authors are listed in alphabetical order.

https://doi.org/10.1016/j.future.2026.108672
Received 16 March 2026; Received in revised form 5 June 2026; Accepted 11 June 2026
Available online 26 June 2026

1. Introduction

The adoption of Artificial Intelligence (AI) and Machine Learning (ML) in scientific research has moved beyond early experimentation. The impact of these techniques over the last few years has revolutionized the field in a wide range of applications, for example, by making it possible to tackle new scientific endeavors [1–3], by automating manual and tedious tasks [4] or by speeding up some time-consuming and effort-intensive tasks [5]. These techniques are a reality and are now part of the basic set of tools used by a wide range of scientists and researchers [6]. However, this rapid adoption has exposed a critical gap: the infrastructure and practices needed to develop, share, and deploy AI/ML models in a reproducible, transparent, and interoperable manner are still missing or remain fragmented within the scientific community.

This gap can be identified along two different axes. On the one hand, there is a large life-cycle fragmentation: the tools available for researchers working on AI/ML (integrated development environments, experiment tracking, model repositories, compute infrastructures, etc.) are normally independent components that work in isolation, each addressing a different AI/ML life-cycle phase. Assembling all these components to support researchers in the development of their pipelines poses a significant technological burden on individual scientists. On the other hand, from a scientific standpoint, these tools have been developed without considering the specific needs of scientific research, and in particular, there is a clear absence of *FAIR-by-design* enforcement. The FAIR principles (Findable, Accessible, Interoperable, Reusable) [7] have become a key pillar for open science practices and are the norm in scientific research.[2] However, in spite of this fact, their application to AI/ML assets remains largely manual, and hence, their adoption is uneven.

We argue that these two axes cannot be treated as independent problems that can be addressed separately. General data platforms address data-level FAIRness but they are not well suited to the dynamic nature of model development, in which provenance, metadata, and deployment artifacts evolve continuously. FAIR compliance of AI/ML assets cannot be easily achieved by simply adding metadata on top of a fragmented pipeline, and structural (i.e., FAIR-by-design) enforcement is needed. Metadata generation and its interoperability are only easily achievable if a single platform controls how assets are generated, built, versioned, and published; provenance records are only meaningful if they can capture the whole chain from data ingestion through training to deployment; interoperability cannot be achieved if there are vendor lock-ins or other trade-offs. This creates significant architectural dependence, meaning that end-to-end integration is a necessary requirement to achieve FAIR-by-design enforcement. To the best of our knowledge, there is no open-source platform that addresses both axes together, particularly within the federated, multi-institutional context of European research infrastructures.

Machine Learning Operations (MLOps) is the closest analog in the industry. MLOps has emerged as a set of practices and tools designed to facilitate the management of the entire AI/ML life-cycle [8], analogous to how DevOps bridges the gap between software development and IT operations. Although industrial MLOps platforms have been successfully adopted, they are designed for cluster-locked and vendor-controlled environments and lack native federation across heterogeneous and distributed e-infrastructures. Moreover, its adoption in the scientific and research contexts is not prominent, and the systems supporting it prioritize operational performance over the scientific transparency, open standards, and FAIR compliance that research demands.

[2] The FAIR principles aim to provide guiding principles and standards designed to optimize the reuse of digital assets.

This gap is particularly critical in the context of the European Open Science Cloud (EOSC) ecosystem. The EOSC is an initiative of the European Commission aimed at creating a "web of FAIR data and services", as a transformative process towards the promotion of open science practices among researchers, and also provides a comprehensive environment for the sharing, processing, and analysis of research data [9]. In the context of AI/ML applications and the EOSC ecosystem, the heterogeneity of computing providers and the diversity of scientific communities make it more complex to reach this ambitious vision, and tackling it from a simple infrastructure standpoint is not enough, requiring platforms that can operationalize open science principles across the entire AI/ML life-cycle.

In this context, we present the AI4EOSC platform, a federated open-source platform that simultaneously addresses both axes of the gap. The central architectural contribution of this work is the integration of the full AI/ML life-cycle (from interactive development environments and federated training, through serverless deployment and drift monitoring, to W3C PROV-compliant provenance tracking and semantic interoperability via MLDCAT-AP) into a unified system where FAIR principles are enforced at the infrastructure level, automatically and transparently, without placing an additional burden on researchers. AI4EOSC is tightly integrated within the EOSC ecosystem, spans the cloud-to-edge continuum, integrates natively with the EOSC and other research infrastructures, and has been deployed and validated in production by three independent scientific communities.

The remainder of this manuscript is structured as follows: Section 2 introduces the background and related work. In Sections 3 and 4, we discuss the general architecture of the AI4EOSC platform and the design considerations that have underpinned its development. In Section 5, we provide performance metrics that validate the robustness of the platform. In Section 6, we explain how different scientists can leverage the platform across the full AI/ML life-cycle. Finally, Section 7 presents the main conclusions and future work.

2. Related work and motivation

Although there are many private and proprietary platforms to deploy AI/ML workloads, few have focused on addressing the specific open science and reproducibility requirements that are key to scientific research. The current landscape is characterized by framework fragmentation, where existing solutions address specific parts of the AI/ML life-cycle, often in isolation. We argue that there is a clear lack of integrated, open-science-focused solutions that span the entire life-cycle from development to deployment.

2.1. ML life-cycle platforms and industrial MLOps

Standard industrial or enterprise frameworks such as Kubeflow [10] or Polyaxon [11] provide robust automation but remain largely "cluster-locked" to specific Kubernetes environments. Federation of these clusters is not possible without the use of additional tools or specialized control planes. This restricts their utility for executing workloads across the geographically distributed research e-infrastructures typically used in scientific collaborations. Furthermore, while some existing tools offer experiment tracking (either open-source like MLflow [12] or LabML [13], or proprietary ones like Weights and Biases [14], Valohai [15] or Comet [16]), their focus remains on industrial workflows with limited support for open science and automated Findable, Accessible, Interoperable, and Reusable (FAIR) principles.

Moreover, end-to-end support for the whole AI/ML life-cycle process exists as commercial and closed-source products (e.g., H2O.ai [17]) and restricts model deployment or data storage to their specific cloud solutions (such as Google Vertex AI [18], Amazon Sage Maker [19] or Azure ML Studio [20]), thus defeating the open principles on which science is built. Similarly, model-serving solutions such as Seldon Core [21] prioritize operational performance over scientific transparency and deep tracking required for research trustworthiness.

2.2. Model repositories and scientific catalogs

General-purpose repositories such as Hugging Face [22], OpenML [23] (an open platform to share datasets and AI/ML models), and specialized scientific catalogs such as the BioImage Model Zoo [24] or Kipoi [25] have been demonstrated to be essential for sharing assets with a broader community. However, model metadata is often unstructured and limited, and repositories are often detached from the training process, making it impossible for users to assess the full lineage or provenance of a model. AI4EOSC overcomes this by integrating the repository directly with a platform-enforced CI/CD provenance pipeline, ensuring that users can fully assess the provenance of any given asset.

2.2.1. Provenance in ML pipelines

The AI4EOSC provenance system shares its core objective with other PROV-based ML pipeline initiatives, most notably yProv [26], which also treats provenance as a first-class concern and aims to capture the end-to-end relationships between data, models, and processing steps. However, these two approaches reflect a fundamental design trade-off. yProv achieves fine-grained lineage tracking by integrating tightly with the execution environment and instrumenting pipeline components to emit provenance information at runtime. This enables the detailed capture of model evolution, parameter tuning, and intermediate data transformations, making it well suited to controlled settings where the full pipeline is under unified management.

AI4EOSC takes the opposite approach: provenance is collected non-intrusively from metadata already exposed by existing platform components without requiring any modification to their internal behavior. While this constrains the level of detail to what those components natively provide, it offers significantly greater flexibility and ease of adoption in heterogeneous real-world deployment environments where pipeline components are diverse and independently operated. Furthermore, the AI4EOSC approach explicitly targets semantic interoperability by producing a PROV-O compliant JSON-LD output and supporting alignment with domain-specific ontologies, enabling provenance information to be reused and queried across system boundaries rather than remaining confined to the originating pipeline.

2.3. Evolution from the DEEP platform

The DEEP-Hybrid-DataCloud (DEEP) [27] platform pioneered a distributed architecture for transparent access to European e-infrastructure with a focus on scientific applications. Although DEEP successfully provided standardized APIs (DEEPaaS) and eased access to accelerators (GPUs), it lacked the deep semantic interoperability and automated metadata generation now required within the EOSC federation and ecosystem. AI4EOSC represents a significant evolution, introducing a serverless AI-as-a-Service (AIaaS) layer, federated learning (FL) capabilities, and a comprehensive FAIR-by-design framework that was previously absent.

2.4. Open science and FAIR principles

The *FAIR principles* have become a central pillar of modern open science practices. In the context of AI/ML research, these aspects should not be considered just as a mere set of best practices, but as an important vehicle towards AI/ML model transparency, reproducibility, and trustworthiness. While general data platforms such as Zenodo or Dataverse address data-related FAIRness (such as data provenance or licensing) [28], they are often unable to respond to the dynamic nature of AI/ML model development and deployment, hence limiting the adoption of FAIR principles beyond data. Researchers who want to ensure the FAIRness of their complete AI/ML solutions must bundle together a set of tools, standards, and practices, at the cost of reproducibility, efficiency of their research, or compliance with open science principles. AI4EOSC addresses this by operationalizing these principles across the entire AI/ML life-cycle through automated, "FAIR-by-design" mechanisms that are easily extensible to accommodate new or unforeseen requirements.

2.5. AI4EOSC motivation

In this fragmented context, AI4EOSC addresses the need for an approachable, open-source solution that streamlines model creation for both technical and non-technical scientists. The platform differentiates itself through several core architectural innovations (as described in Section 3):

- **Transparent Inter-Cluster Federation:** Our federated workload management system is able to scale across many physically distributed datacenters and e-infrastructures, with a variety of deployment and execution methods, via a single and unified control plane that avoids the infrastructure silos of standard MLOps stacks.
- **Cloud-to-Edge Continuum:** AI4EOSC mobilizes resources across a continuum, enabling deployment on heterogeneous hardware including low-capacity edge devices and cloud computing providers.
- **Hardened Federated Learning support:** The platform extends standard Federated Learning (FL) with token-based authentication, novel aggregation methods, divergence monitoring, and server-side differential privacy and metric privacy notions to defend against client inference attacks.
- **Integrated Provenance Tracking:** The platform automatically transforms training fragments into W3C PROV-compliant RDF graphs, providing lineage traceability that is currently detached or proprietary in industry platforms.
- **Semantic Interoperability:** AI4EOSC leverages MLDCAT-AP in order to deliver interoperability at the semantic level with other platforms and data spaces, enabling seamless interactions with other marketplaces.
- **Integrated Drift Monitoring:** Through the *Frouros* library and the *DriftWatch* system, researchers can detect and visualize concept and data drift at inference time.
- **Seamless EOSC Integration:** AI4EOSC is natively integrated with the EOSC, enabling researchers to leverage EOSC's infrastructure for data sharing (e.g., EOSC EU Node resources), collaboration, or long-term preservation. This integration ensures that AI/ML research is not only FAIR but also interoperable with the broader European research ecosystem.

To synthesize these differences and explicitly contrast AI4EOSC with the state-of-the-art, Table 1 provides a feature-wise comparison across key technical and scientific dimensions.

3. The AI4EOSC platform architecture

In the following section, we describe the architecture of AI4EOSC using a simplified view of the C4 Model [29] to offer a clear and structured visualization of its components and interactions. This simplified representation aims to enhance article readability. For a detailed and comprehensive view of the AI4EOSC architecture, please refer to the complete C4 implementation available in [30].

Our architecture follows a modular design and is organized into four distinct systems: AI4EOSC Development Platform, AI4EOSC AI/ML as a Service, AI4EOSC LLM Services, and AI4EOSC Platform Orchestration. These components are interconnected in order to deliver specific functionality, while remaining integrated with the rest of the systems. This modularity is illustrated in Fig. 1. The legend for all C4 diagrams is shown in Fig. 2.

Table 1
Feature-wise comparison of AI4EOSC against representative industrial MLOps stacks, scientific tools, and the predecessor DEEP platform.

Feature	Kubeflow [10]	Polyaxon [11]	Seldon [21]	SageMaker [19]	OpenML [23]	Hugging Face [22]	DEEP [27]	AI4EOSC (This work)
Scientific Focus	Low	Low	Low	Low	High	Medium	High	**High**
Openness	Full	Open-core	Open-core	Closed	Full	Partial	Full	**Full**
FAIR-by-Design	No	No	No	No	Partial	No	Low	**Native**
Provenance	Limited	Limited	Limited	Proprietary	Limited	Limited	Low	**W3C PROV**
LLM Integration	Partial	Partial	Partial	High	No	High	No	**Native (RAG)**
Learning Schemes	Dist.	Dist.	N/A	Dist.	N/A	N/A	Dist.	**Fed. & Dist.**
Drift Detection	External	Limited	Yes	Yes	No	No	No	**Native**
Infrastructure	Kubernetes	Kubernetes	K8s/Cloud	Prop. Cloud	N/A	N/A	e-Infras	**e-Infras/Edge**
Federation	Cluster-lock	Cluster-lock	Cluster-lock	Vendor-lock	N/A	N/A	Partial	**Inter-cluster**
Delivery Model	Software	Both	Both	Platform	Both	Platform	Both	**Both**
EOSC integration	No	No	No	No	No	No	Partial	**Native**

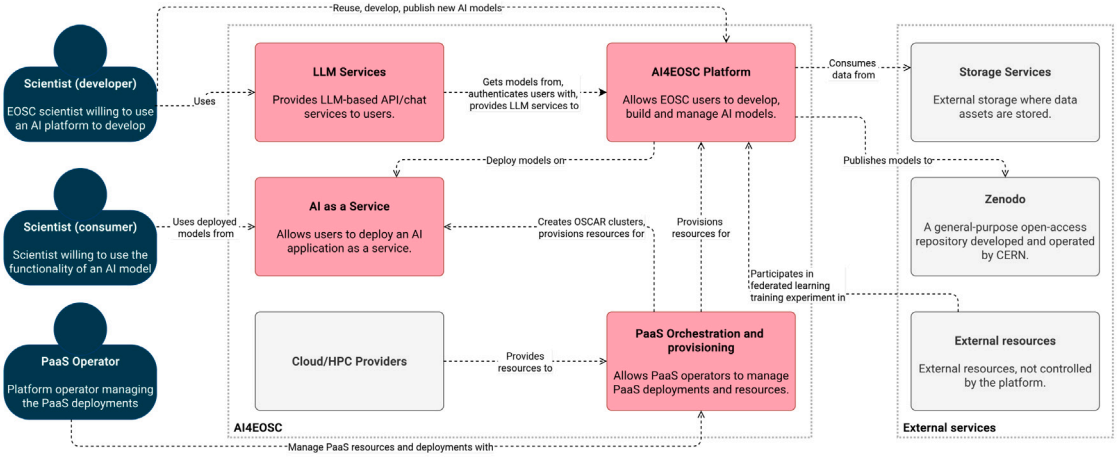

Fig. 1. Simplified view of the AI4EOSC architecture system landscape view (C4 notation).

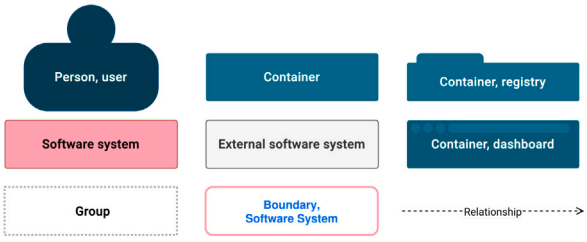

Fig. 2. Legend for all C4 diagrams used in this manuscript.

3.1. AI4EOSC development platform system

This system, depicted in Fig. 3, includes all the services needed to deliver a comprehensive platform and toolbox to enable researchers to build, develop, train, and share AI/ML models following the FAIR principles. It also includes some core, common, and shared services that constitute the main AI4EOSC control plane, which serve as auxiliary services to both this system and others. The most relevant components are explained in the next subsections.

3.1.1. Workload management system

The AI4EOSC WMS handles two primary types of execution: user-oriented workloads and system-level management tasks. User workloads refer to both AI modules Section 3.1.3 and other specialized tools Section 3.1.4 that the platform offers to enrich the user experience. Both can be executed in two distinct modes. On the one hand, in *standard* mode, resources are persistent, providing researchers with continuous access via Interactive Development Environments (IDEs) or APIs. On the other hand, the *batch* mode utilizes a transient allocation strategy, where resources are provisioned strictly for the duration of a training process to maximize global cluster efficiency and resource sharing.

In addition, to enhance user experience and provide structural functionality to the platform, AI4EOSC augments Nomad jobs with transparent sidecar tasks. These components manage additional functions that are not directly exposed to users, such as automating connectivity to external storage systems, performing dataset synchronization from remote repositories at runtime, and triggering user notifications. Moreover, system-level jobs, including Traefik reverse proxies, the Platform API, and the administrative dashboard Section 3.1.2, are also part of these workloads to maintain platform stability and service accessibility without direct user intervention.

3.1.2. Platform API and dashboard

The Platform API (PAPI) is the component that delivers the platform functionality via a well-defined set of endpoints, following a REST architecture. It was developed following the OpenAPI specifications, providing a formal description and documentation of the PAPI functionality.

Built on top of the API, the AI4EOSC Dashboard is the main user entry point to the AI4EOSC Platform. Both PAPI and the dashboard are built with extensibility in mind in order to be able to easily integrate new functionality without interfering with the current one.

The Dashboard (Fig. 4) allows users to easily explore and view the associated metadata for the different catalogs available in the system: AI modules Section 3.1.3, tools Section 3.1.4, LLM services Section 3.3, and other external AI catalogs Section 4.2.2. It also provides the ability to launch development environments or codespaces in

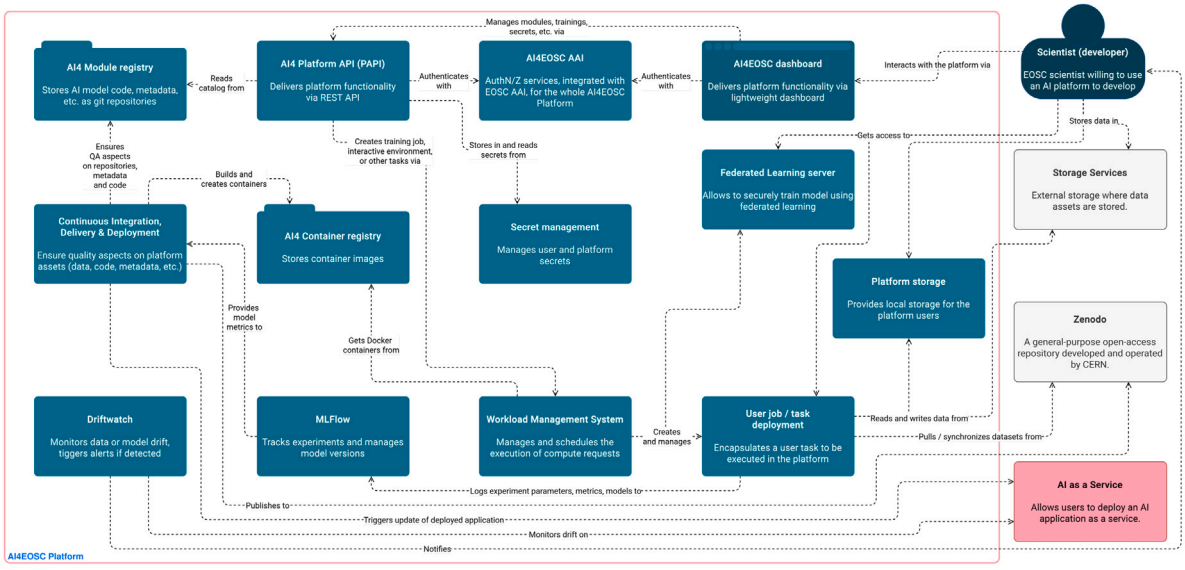

Fig. 3. Simplified AI4EOSC Development platform architecture: container view (C4 notation).

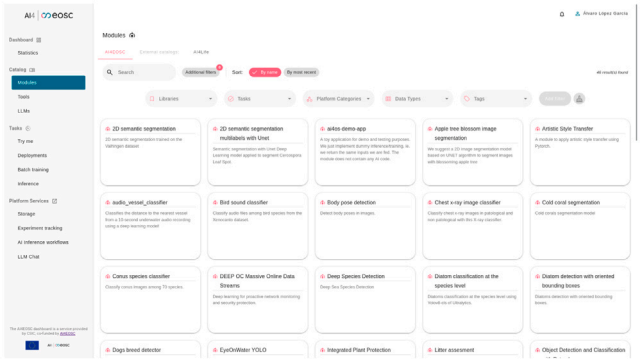

Fig. 4. A snapshot of the AI4EOSC dashboard (available at https://dashboard.cloud.ai4eosc.eu) providing an overview of the existing AI modules in the platform.

the platform, batch executions, easy management of existing deployments, platform-wide secret management (e.g., API keys, storage integration), and resource consumption reporting, among other user-facing functionalities.

3.1.3. AI modules

The AI modules are the main platform assets with which users interact, as they encapsulate the AI/ML models that the platform is able to manage. They are both produced and consumed by users, although the AI4EOSC development team also maintains reference implementations of relevant AI models. They are created and stored as git repositories (currently hosted on GitHub, but any other software forge can be used), allowing users to easily manage and upload new updates to the modules, as well as making it possible to enforce quality assurance mechanisms and MLOps practices Section 3.1.5 from the platform point of view. To support the development of new AI modules that comply with the AI4EOSC platform and best practices (e.g., metadata standards Section 4.1.1), a set of predefined software templates is provided to users.

3.1.4. Tools

Tools are a collection of additional assets that the AI4EOSC Platform provides to users to support them during the entire ML development life-cycle. These tools profit from the platform's extensibility Section 4.2 as a means to enhance and augment the current functionalities.

Unlike AI modules Section 3.1.3, tools are developed and curated by the developers of the AI4EOSC Platform to ensure that the integration is implemented correctly. Once integrated, tools are easily self-deployable by users in their personal workspaces, similar to AI modules.

For technical reasons, some of these tools are not suitable for per-user deployment and are therefore provided as platform-wide services, such as the experiment tracking service or drift monitoring Section 3.1.5; however, the integration mechanism remains the same.

In the following, we explain the main set of tools offered by the platform.

Development environment — codespaces. The AI4EOSC Platform allows users to deploy a ready-to-use development environment, referred to as Codespaces, to create their new AI modules Section 3.1.3. The environment can be used out-of-the-box with multiple versions of the main Deep Learning frameworks such as TensorFlow [31] and PyTorch [32]. Additional frameworks are also supported by starting from the provided plain Ubuntu or NVIDIA CUDA flavors or by installing a custom software stack.

When configuring a deployment, users can select their preferred IDE for code development. The platform currently supports both Visual Studio Code (VS Code) bundled with many useful extensions and installed using Code Server and JupyterLab. All IDE endpoints are password- or token-protected to prevent malicious actors from accessing the user data and code.

Federated learning servers. The AI4EOSC Platform enables the execution of federated learning servers [33], allowing AI/ML models to be trained collaboratively across multiple institutions without sharing private data. The platform currently supports the two most widely used and robust frameworks, namely, Flower [34] and NVFLARE [35]. Both FL servers can be launched interactively from the dashboard, and FL clients can be deployed within the AI4EOSC Platform or connected from outside, thus easily accommodating the individual privacy needs of the different data owners. This flexible and distributed approach allows leveraging a wide range of resources in the cloud continuum that could not support a full ML training cycle by themselves due to their low capacity (e.g., edge devices).

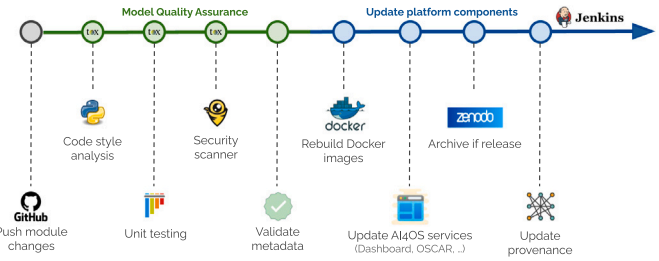

Fig. 5. CI/CD pipeline for AI modules.

Beyond the standard framework capabilities, AI4EOSC has extended the Flower framework with a set of platform-level enhancements. First, we implemented token-based client–server authentication [36], managed directly from the dashboard via the platform secret management system. This allows operators to create and revoke client tokens at any time, ensuring that only authorized clients can participate in the training. Second, the server incorporates a mechanism to monitor the divergence between the model weights of different clients, allowing the detection of clients that deviate significantly from the rest and may introduce poisoning behavior with the option to revoke their access tokens immediately. Third, server-side differential privacy can be applied during aggregation using fixed-clipping Gaussian mechanisms, and the platform further incorporates the notion of server-side metric differential privacy [37], which scales the noise multiplier dynamically based on a distance metric computed from local model parameters, thereby protecting against client-inference attacks. Finally, AI4EOSC introduces *FedAvgOpt* [38], a novel aggregation method that optimizes the aggregated weights to be as close as possible to the individual client weights using the Nelder–Mead method, improving convergence without compromising privacy. The supported aggregation strategies (i.e., *FedAvg*, *FedAvgM*, *FedMedian*, *FedProx*, *FedOpt*, *FedAdam*, *FedYogi*, and *FedAvgOpt*) can be selected directly from the dashboard.

The security and privacy implications of this learning approach, including its role in defending against poisoning and inference attacks, are discussed in Section 4.4.3.

3.1.5. MLOps practices

Extending the DevOps approach to the ML realm, AI4EOSC delivers a series of tools to allow users to address the key challenges that specifically arise during the ML cycle.

Continuous integration and delivery of models. A key component of the AI4EOSC platform that underpins the MLOps model management is the CI/CD system comprising both platform-enforced and user-defined pipelines executed via Jenkins. This system ensures that any AI module integrated into the marketplace satisfies the same consistent requirements, for example, by ensuring that published models include the required metadata or by triggering updates on existing services based on a given module.

To achieve this, AI4EOSC maintains a platform-wide pipeline definition enforced in all new and existing modules. This pipeline is executed only after the user-defined jobs are executed, and it is optimized to execute different tests according to the changes that trigger them, such as metadata updates only or modifications to the source code.

The platform-level CI/CD pipeline is shown in Fig. 5. As shown, there is a set of steps that comprise software and model quality assurance (e.g., code style, unit testing, security scanning, metadata validation), and a second set of steps that are focused on the delivery of the model to the corresponding components, including keeping track of the steps in the provenance system. From a platform perspective, this ensures the following key aspects:

- Metadata validation: We ensure that the user-defined metadata Section 4.1.1 is compliant with the metadata schema that the platform relies on. This is a critical step, considering that the PAPI, Dashboard, and interoperability with other metadata schemes require that accurate and consistent metadata is in place.
- Docker image generation and publication: The platform relies on Docker containers to ensure that the modules are self-contained and executable in a wide variety of systems. Therefore, we trigger the build of new images and their publication in the platform Harbor container registry[3] and Docker Hub.[4] To avoid unnecessary builds, the rebuild step is triggered only when non-metadata changes are detected and can be disabled in specific git branches.
- Zenodo release integration: In order to ensure that the code remains accessible, and following the FAIR principles also for software and model assets, we ensure that the Zenodo-GitHub integration is in place. Otherwise, we create a deposit and directly upload any pending releases.
- Notification to other platform services: As we will explain in Section 3.2, the platform provides an AI as a Service component that allows delivering the AI/ML model functionality following a serverless model. In order to notify the services that a new version of the model is available, we notify the services via defined webhooks, providing an updated metadata version, so that they can react to these changes and update the running modules if they are configured to do so. Moreover, we also notify the PAPI to trigger an update of the cached model metadata.
- Notification of provenance updates: The AI4EOSC platform keeps track of the model provenance Section 4.1.2. Therefore, whenever a model is updated, we notify the provenance system, including the CI/CD-gathered metadata (e.g., tests executed), ensuring that the provenance graph always reflects the latest changes.

Experiment tracking. Users deploying AI modules Section 3.1.3 or Development Environments Section 3.1.4 can log their training metrics into a dedicated MLflow [12] tracking server instance that we have deployed, featuring a custom Authentication GUI to be able to integrate it with our authentication system. MLflow credentials are saved in the platform secret manager and automatically injected into the WMS deployments Section 3.1.1 created from the dashboard Section 3.1.2.

By logging multiple training runs and saving the corresponding model weights with proper versioning, the reproducibility of the models can be enhanced. This information is later included in the provenance chain, as explained in Section 4.1.2.

Drift monitoring. AI4EOSC has developed a drift monitoring system, DriftWatch.[5] This component allows monitoring of AI modules Section 3.1.3 in production, warning the model developer when performance starts to degrade and model retraining is required. In order to detect if a drift is happening, a detector is needed. DriftWatch is agnostic to the underlying drift detection framework chosen by developers, such as Alibi-detect [39]. However, AI4EOSC also developed Frouros [40], a Python library for drift detection in machine learning systems that supports a large combination of classical and more recent algorithms for both concept and data drift detection. When an AI module is developed, users can define reference clean/faulty data and train a Frouros detector on them (users handling image data would need to first train an autoencoder to reduce the dimensionality of their data). Then, at inference time, when the models are deployed in the production AIaaS Section 3.2, the inference data is passed through the Frouros detector, and the resulting p-values for detection are sent to the DriftWatch server [41]. From an intuitive user interface, users can log in and visualize whether drift has occurred, eventually inspecting the data that caused the drift.

[3] https://registry.cloud.ai4eosc.eu/
[4] https://hub.docker.com/u/ai4oshub
[5] https://drift-watch.cloud.ai4eosc.eu/

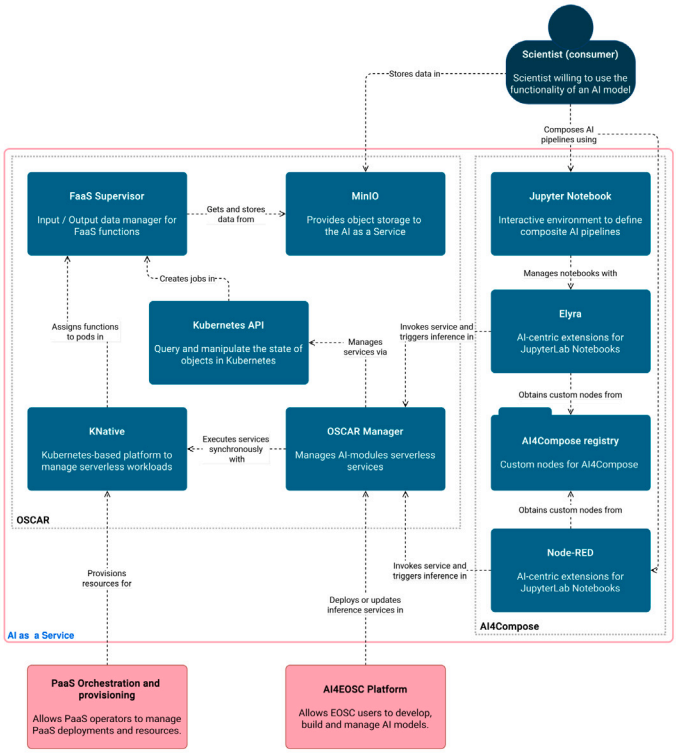

Fig. 6. Simplified AI4EOSC AI/ML as a Service architecture: container view (C4 notation).

3.2. AI/ML as a service system

The AIaaS, shown in Fig. 6, allows users to deploy AI modules Section 3.1.3 in a serverless inference platform based on OSCAR [42].

The OSCAR cluster allows autoscaling of resources on demand, automatically deploying multiple instances of the model to run predictions as needed. The platform allows both synchronous and asynchronous calls to the endpoints via file uploads to a MinIO object store. The AI4EOSC Dashboard Section 3.1.2 allows users to automatically create and manage serverless endpoints for any AI module in the catalog. Thanks to the DEEPaaS API [43] which interfaces the module, users are able to send inference requests to any AI module in a standardized way.

Leveraging on those serverless endpoints, the AI4EOSC Platform allows the creation of AI pipelines where users can compose multiple inference requests to different AI models, as well as data preprocessing and postprocessing steps.

Pipelines can currently be created with two different visual composition tools: Node-Red, whose instances are managed via the OSCAR cluster, and Elyra, integrated with Jupyter-based environments such as the AI4EOSC Development Environment Section 3.1.4 or the EGI Notebooks. Users can choose between these two tools to compose their pipelines based on their working environment requirements and prior experience.

3.3. Large language models services system

This system, described in Fig. 7, is designed to provide a federated inference system focused on the needs of Large Language Models (LLMs). The system leverages vLLM [44] and LiteLLM in order to provide an easy way to include additional LLM providers that can be geographically distributed. To achieve this, each resource provider deploys its own LLMs with vLLM and aggregates them with a local LiteLLM instance. Then, the platform's LiteLLM instance aggregates the

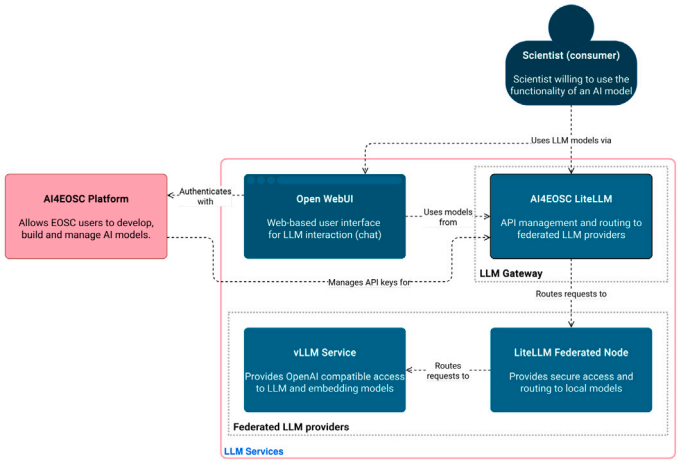

Fig. 7. Simplified AI4EOSC Large Language Model Services architecture: container view (C4 notation).

different local LiteLLMs, effectively serving all provider LLMs under a single endpoint.

Moreover, the AI4EOSC Dashboard allows users to easily manage API keys to access the service. This system can be directly used via its OpenAI-compatible API endpoints, in order to be integrated with any user tool, and via the platform's conversational engine based on OpenWebUI. Moreover, this system is key to providing AI4EOSC platform users with a specific support agent that can support users with direct answers from the documentation knowledge base.

3.4. Platform orchestration system

This system, depicted in Fig. 8, assists resource providers and platform operators by automatically provisioning the resources needed to deploy the rest of the platform services. In the AI4EOSC platform, we use the INDIGO-PaaS Orchestration system [45,46] to handle the deployment of platform components across the different IaaS platforms made available by the federated cloud providers.

The system is composed of different microservices.

- The Infrastructure Manager [47,48], which is responsible for the submission of the deployment to the proper Cloud provider. The provider is selected according to the AI-Ranker [49], which generates a ranked list of providers by combining the prediction of deployment success/failure with the estimated deployment creation time.
- The Federation-registry [50], which is the configuration management database for dynamic cloud federation. Built using FastAPI and neo4j, the federation registry can efficiently manage federation-related information and SLA agreements.
- The INDIGO-PaaS Orchestrator Dashboard, which provides a user-friendly interface for managing and monitoring the TOSCA-based deployments, built on Flask and MySQL.

With the developed TOSCA [51] templates, the PaaS Orchestration layer is able to automate the deployment of the platform components [52], including the Workload Management System Section 3.1.1 and the AI as a Service platform Section 3.2.

These highly reproducible recipes make the AI4EOSC platform a suitable solution that can be easily adopted by external communities wishing to profit from the technologies developed here. Moreover, the adoption of well-known technology standards allows cloud-enabled resource providers to easily join the federation and contribute to the existing AI4EOSC platform.

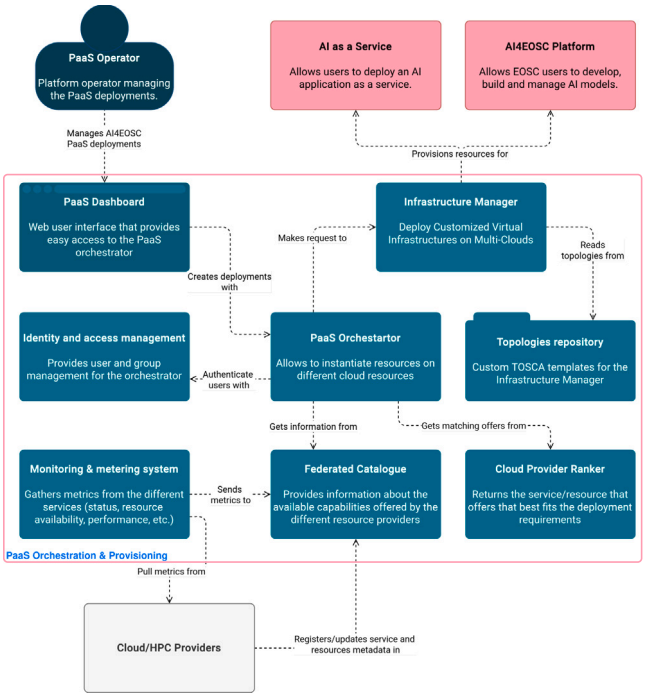

Fig. 8. Simplified AI4EOSC platform orchestration architecture: container view (C4 notation).

4. Platform design considerations

4.1. Platform FAIR design

As mentioned when presenting the motivation, the FAIR principles are central to the design of the AI4EOSC platform. Ensuring that AI modules and their associated assets are transparently discoverable, well-described, and reusable by the broader scientific community is a core requirement for a research-oriented platform. AI4EOSC addresses this through standardized metadata schemas, comprehensive end-to-end provenance tracking, and commitment to openness. This approach ensures that every asset produced by the platform can be understood, reproduced, and built upon by others.

4.1.1. AI4EOSC metadata

In the current data-driven ecosystem, rich metadata are of paramount importance to comprehensively describe a given asset in a homogeneous way. Well-defined metadata also allow interoperability, both at the platform and AI/ML model levels. To ensure that all the AI modules developed in the platform provide consistent metadata, the AI4EOSC Platform enforces that each AI module Section 3.1.3 or tool Section 3.1.4 registered in the AI4EOSC catalog must include a valid metadata file following a defined schema implemented via JSON Schema.

The AI4EOSC metadata schema includes both user-defined fields (some of them optional) as well as fields automatically filled from external sources such as license and modification dates from GitHub. User-defined fields include title, summary, description, DOI, external links (e.g., to dataset or model weights), and tags such as the software libraries used or the data types used as input.

Once integrated, the Platform API retrieves the module's metadata and serializes it into different formats (JSON-LD, RDF Turtle), being able to transform it into different application profiles, such as the MLDCAT Application Profile. This allows for greater interoperability of our models with external catalogs, allowing semantic interoperability with formats that are compatible with MLDCAT-AP.

4.1.2. Provenance

Provenance is essential for science, and a FAIR-by-design system should consider asset provenance as one of its pillars. AI4EOSC provides an integrated and extensive provenance system that allows the transparent traceability of any AI module Section 3.1.3, thus improving its reproducibility.

The system is designed to operate in a non-intrusive manner, leveraging metadata already generated by existing architectural components, such as MLflow and Jenkins, without requiring modifications to these systems. Once changes to a module have been committed to the Git repository, the CI/CD pipeline triggers the collection of provenance metadata from multiple sources: module metadata Section 4.1.1 and training resources Section 3.1.1 from the Platform API Section 3.1.2, experiment tracking metrics from MLflow , and build information from the CI/CD pipeline itself Section 3.1.5.

These heterogeneous, non-semantic metadata fragments are stored in their native JSON representation in a PostgreSQL database. This design choice is deliberate: rather than imposing a rigid schema at ingestion time, the system preserves the original structure of each source, enabling flexible querying and transformation. Semantic interoperability is then achieved by applying RDF Mapping Language (RML) rules via CARML over the stored JSON fragments. These declarative mappings define how disparate metadata elements correspond to a unified semantic model, thereby enabling the integration and linkage of otherwise disconnected provenance information in a coherent end-to-end provenance graph.

The resulting provenance graph is serialized as an RDF document encoded in JSON-LD, ensuring compliance with the PROV-O ontology for standardized provenance modeling. The system also supports alignment with domain-specific ontologies, allowing for contextual enrichment and adaptability across different AI application domains. Having the provenance data in RDF further allows conversion to external formats, such as FAIR4ML [53] or any additional mapping, and it can be extended with virtually any information that the platform or the user is able to provide.

A comparative analysis of this design with other PROV-based ML pipeline approaches is presented in Section 2.

In order to present the provenance information to users, an interactive web-based visualization tool has been developed and integrated into the AI4EOSC dashboard, providing dynamic, user-friendly interfaces for navigating complex provenance graphs. Additionally, users can query the provenance graph in natural language via integration with a chatbot assistant (Section 3.3), thereby enhancing transparency, traceability, and reproducibility within AI deployment workflows.

Fig. 9 shows a concrete provenance record for the exemplary `ai4os-demo-app` module. When a developer commits new code, the Jenkins CI/CD agent detects the change (`jenkins-polling`), triggering Build 101. This activity, which is associated with Jenkins acting on behalf of the AI4EOSC platform, validates metadata, builds a Docker image, and notifies the provenance system. Simultaneously, a training run (UUID `5e96fe94`) is executed on Nomad resources with metrics logged to MLflow, producing a versioned build artifact. All these relationships are captured as W3C PROV predicates (e.g., `wasGeneratedBy`, `wasAssociatedWith`, etc.), forming a complete, queryable lineage graph for the module.

4.1.3. Openness

AI4EOSC integrates open-source principles at its core: all the platform software is publicly available,[6] which, combined with the provided TOSCA deployment recipes used in the PaaS Section 3.4, enables communities to redeploy and build on top of our platform.

In a similar spirit, every AI module built with the platform has a public code repository attached to it, hosted under a common GitHub

[6] https://github.com/ai4eosc

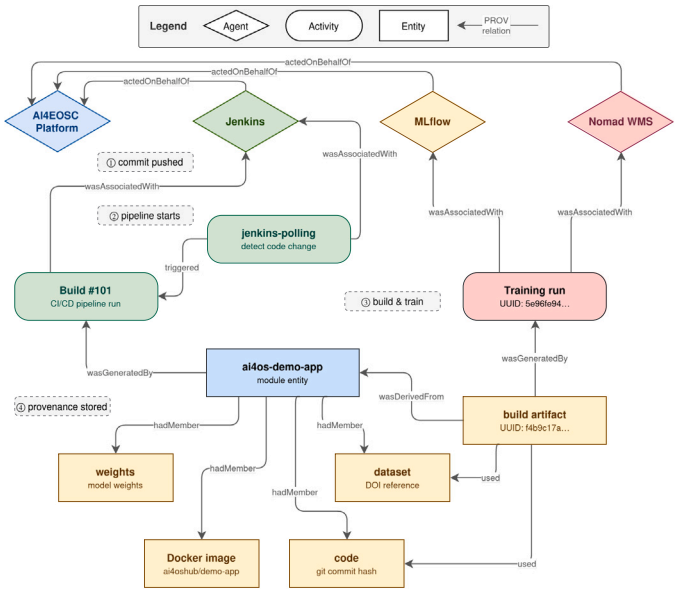

Fig. 9. Example of a provenance graph built for a demo application.

organization.[7] The code, as well as any other downloadable asset (model weights, training dataset, metadata, provenance file, etc.), is clearly linked from the AI module page in the AI4EOSC Dashboard Section 3.1.2. To avoid vendor lock-in, AI4EOSC not only allows but *facilitates* the redeployment of all AI modules in external clouds Section 4.2.1.

4.2. Platform extensibility and interoperability

One of the design decisions of AI4EOSC is that any scientific platform must be able to adapt to the needs of its user community, and not vice versa. In order to be able to comply with this design motto, our systems are built with extensibility in mind, meaning that we are able to integrate, to the extent technically possible, with external tools easily. In this section, we present the three AI4EOSC extensibility and interoperability axes.

4.2.1. External deployment platforms

All executable assets in the platform are based on Docker containers, which are being continuously released via reproducible builds into a public Docker Hub organization Section 3.1.5. This implies that the AI modules Section 3.1.3 are easily deployable by external cloud providers and in any other infrastructure where Docker is available. From the Dashboard Section 3.1.2, we give users direct access to deployment in any external cloud provider via the Infrastructure Manager, which is part of our orchestration system Section 3.4. We have also developed connectors that can be deployed on the EOSC EU Node, allowing users to leverage its resources.

4.2.2. External AI catalogs

Due to our AI module design Section 3.1.3, which hides the module's internal workings behind a unified API [43], we can easily build connectors with external AI module catalogs. These connectors allow the deployment of these external modules in our platform. To demonstrate this, we built a connector to the BioImage Model Zoo catalog [24] that leverages their standardized metadata to map their functionality into our platform and makes their AI modules directly available in our dashboard Section 3.1.2. This feature allows one-click deployments of those external modules, allowing BioImage Model Zoo users to continue working with their tools but making their outcomes available in the AI4EOSC marketplace.

4.2.3. External datasets

AI/ML models rely heavily on data for development and training. Therefore, it is important to provide users with the ability to directly download data. Additionally, in a metadata-rich platform such as AI4EOSC, users can specify which dataset has been used to train a given model by referencing their DOI or URL. This capability is particularly valuable because it makes it easy to reproduce the results, ensuring transparency and reliability in the research process. Our dashboard enables users to designate the data they wish to synchronize at launch time, streamlining the data management process.

We implement this functionality transparently via WMS sidecar tasks Section 3.1.1 and by extending the DataHugger tool [54] with additional integrations, currently including Zenodo, Data Europa, Dryad, or SeaNoe.

4.2.4. External storage providers

In addition to the platform storage, we can connect user deployments with any RCLONE-compatible external storage, including Nextcloud, S3, Owncloud, Dropbox, and Google Drive. In order to do so, we allow users to easily link the platform with selected storage systems (e.g., NextCloud) or manually provide their user credentials for other storage providers.

4.3. Security and access control

Taking into account the multi-user, multi-institution, and multi-country nature of the AI4EOSC platform, security and data protection have been carefully considered in the design and implementation. In this regard, the platform adopts a layered security model that addresses authentication, authorization, secret management, tenant isolation, and data protection.

4.3.1. Authentication and authorization

The platform enforces authentication and authorization through a dedicated Keycloak instance using OpenID Connect (OIDC) as the protocol. Thus, all AI4EOSC services use this endpoint as the authoritative source of authentication, acting as a single trusted broker with external federated identity providers, such as EGI Check-In or MyAccessID (eduGAIN) services. This design, aligned with the AARC Blueprint Architecture [55], reduces the trust boundary to a single managed service, lowering the complexity of dealing with identity providers at the individual service level, and thus decreasing the exposure level.

Authorization follows a role-based access control (RBAC) derived both from the internal group mapping in the system and from the inherited entitlements from trusted identity providers. Platform services enforce different tiered access levels (from short-lived and constrained demo access to complete resource access).

4.3.2. Secret management

In addition to user authentication, the platform manages a set of other secrets that can grant access to different components of the platform. These include storage tokens used to sync with external providers or tokens to join a federated learning scenario, among others. In order to manage them securely, AI4EOSC enforces the usage of Vault, so that secrets are never exposed as plaintext to the platform infrastructure, as they are injected via the WMS Section 3.1.1.

AI4EOSC does not enforce credential rotation, but users can define the secret lifetime when creating them, and secrets can be revoked at any time via the platform dashboard or the API. This allows, for instance, invalidating the token of a malicious client participating in a federated learning scenario or revoking a compromised credential. This authentication is also intended to prevent poisoning attacks by unauthorized clients attempting to connect to the training, as will be further explored in Section 4.4.3.

[7] https://github.com/ai4os-hub

4.3.3. Tenant isolation

AI4EOSC provides two levels of isolation. At the organizational level, Nomad namespaces are leveraged to partition resources across dedicated platform instances, for example, for different research groups, institutions, or projects such as iMagine or KMD4EOSC Section 6.2. This ensures that the resources, catalogs, and configurations of different communities are fully segregated. Moreover, at the workload level, user deployments are isolated at the internal network level, ensuring that deployments from different users do not interfere with or observe each other. Network-level access between deployments is restricted by default, so that inter-deployment communication is only possible through the explicitly exposed public endpoints, protected by the token-based access control described previously.

4.4. Privacy considerations

For a scientific platform like AI4EOSC, privacy is a paramount concern, as researchers frequently handle sensitive, proprietary, or geographically restricted datasets (e.g. medical datasets). Ensuring robust privacy mechanisms not only guarantees compliance with European data protection regulations such as the GDPR but also establishes the necessary trust among institutions to enable collaborative AI workflows without compromising data confidentiality.

4.4.1. Data retention

According to the GDPR, the platform was designed following the data minimization principle. Uploaded data is only stored in the user's personal space and is not retained after the user deletes it. When a user deployment is terminated, all locally cached data in the compute nodes (including datasets synchronized from external data providers) are completely removed. Users are required to adhere to the AI4EOSC privacy policy upon registration, and data-processing agreements are in place with all federated infrastructure providers.

4.4.2. Logging and traceability

All user-facing actions are always routed through the Platform API (PAPI) (Section 3.1.2), which provides a centralized and structured audit log of all platform activities. This includes job submissions, deployment life-cycle events, secret creation and revocation, and catalog interactions, enabling operators to trace the origin of any platform action.

4.4.3. Federated learning

Federated learning is a privacy-preserving design approach because it does not require participating clients to share their data with each other or with a central server. However, additional security and privacy risks remain, including poisoning and client-side inference attacks, which must be addressed at the platform level.

As described in Section 3.1.4, AI4EOSC extends Flower with token-based authentication and client weight divergence monitoring to defend against poisoning from both unauthorized and malicious authenticated clients. Server-side differential privacy and metric privacy [37] further protect against inference attacks in the aggregated model. Together, these mechanisms form a layered defense that operates transparently within the platform without requiring additional configurations from researchers. More specific measures remain complex to implement generically at the platform level and must be addressed on a case-by-case basis.

4.5. Reliability, fault tolerance, and elasticity

The distributed and federated nature of the AI4EOSC platform requires a robust and reliable resource management system. Our design prioritizes system graceful degradations to maintain service availability despite the heterogeneous reliability of the underlying highly distributed provider nodes.

4.5.1. Fault tolerance and self-healing

To address the potential failure of nodes in the WMS, we leverage the native self-healing capabilities of the underlying orchestrator (i.e., HashiCorp Nomad, as described in Section 3.1.1). The system maintains a constant state and heartbeat monitor of all federated nodes and supports graceful degradation due to the Nomad server redundancy. Additionally, Nomad incorporates consensus protocols to elect new cluster leaders in the event of a failure or network partition.

4.5.2. Resource elasticity and backpressure

The platform differentiates between the elasticity requirements of the training and inference workloads as follows:

- **AIaaS Autoscaling:** For inference services, the platform implements reactive horizontal autoscaling. Based on real-time metrics (e.g., request latency or CPU/GPU utilization), the AIaaS layer can dynamically provision additional container instances to handle traffic spikes, ensuring low-latency responses for end-user applications.
- **WMS Queueing and Determinism:** For training workloads, we adopt a deterministic resource allocation strategy. Given the high cost and limited availability of GPU resources across scientific resource providers, we do not implement automated hardware provisioning (cluster autoscaling). Instead, we manage high demand through a queuing system. This prevents the over-saturation of the system as new jobs are held in a pending state until sufficient resources are released by the completed tasks.

5. Performance evaluation

5.1. Scalability

To assess the scalability of the platform, in Fig. 10 we show the time it takes to search and retrieve the full information of a single job as a function of the total number of jobs deployed in the AI4EOSC cluster federation. In blue, we show the scenario when all the jobs are running in the same datacenter, while in red, we show the case where the jobs are equally distributed across all the datacenters of the federation. As can be seen, search times increase sub-linearly with the number of total jobs, while job retrieval times stay constant, both in the range of a few tens of milliseconds. More importantly, increasing the number of datacenters in the federation only increases the search times by a small fixed cost. This, together with the sublinear increase in search times, shows the robust scalability capacity of the AI4EOSC federation, both in the number of jobs managed and in the number of integrated infrastructure providers.

5.2. Training efficiency

To test the possible overhead of training in an AI4EOSC environment, we trained the same Computer Vision model (from a real-life AI4EOSC use case published in [56]) in three different scenarios: (1) directly in a Virtual Machine (VM); (2) inside a Docker container deployed in a Virtual Machine; and (3) in an AI4EOSC Development Environment Section 3.1.4, which is, in the end, a Docker container running in a VM with Nomad/Consul orchestration Section 3.1.1 and accessed via VS code Section 3.1.4.

In each scenario, the model is trained for 25 epochs on the same resources (one NVIDIA T4 16 GB GPU, 20 CPU cores, 86 GB RAM, 40 GB disk) with the same environments (Python 3.13, TensorFlow 2.20, CUDA 12.4, cuDNN version 9.21.01). The dataset is approximately 1 GB in size.

As shown in Table 2, the dockerization on top of the VM does not introduce significant overhead. Running in an AI4EOSC environment results in a minor time increase (less than 1%). This very small difference, caused by the Nomad/Consul orchestration overhead, demonstrates that the AI4EOSC platform can transparently serve federated resources at scale to users without significant performance loss.

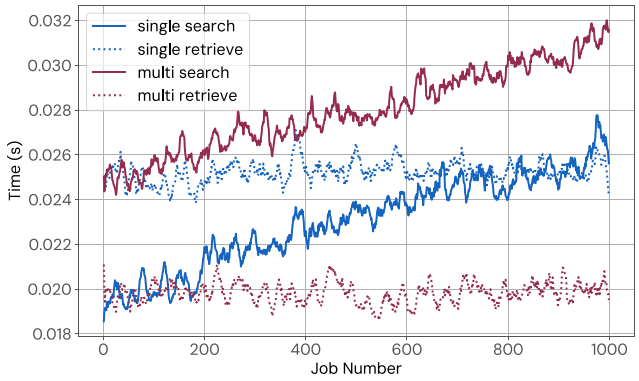

Fig. 10. Search and retrieval times for a single WMS deployment, as function of the total jobs in the cluster. In blue, when launching all jobs in the same datacenter. In red, the load is distributed equally among all the datacenters. For clarity, a rolling mean with window size 10 has been applied to smoothen the curves.

Table 2
Training time (in seconds) for a real-life AI4EOSC use case trained for 25 epochs, averaged over 5 training runs.

VM	VM + Docker	AI4EOSC
112.4 ± 1.7	112.6 ± 1.2	113.5 ± 1.2

Table 3
Training time (in seconds) for a federated learning setting with 5 rounds, averaged over 5 training runs with the standard deviation.

Server \ *Clients*	AI4EOSC	VMs
AI4EOSC	229.18 ± 5.90	233.60 ± 14.61
VM	222.61 ± 14.16	226.96 ± 11.49

5.3. Federated learning efficiency

Concerning the Flower FL server and the efficiency of the implementation in AI4EOSC, we conducted four experiments with the same use case, considering the four clients presented in [56]. The experiments are as follows: (1) both the FL server and the four clients are deployed in AI4EOSC; (2) the FL server is deployed in AI4EOSC and the four clients run in four VMs; (3) both the FL server and the four clients run in five different VMs; and (4) the FL server is deployed in a VM and the clients run in four separate AI4EOSC deployments. Concerning the training, each client trained the model for one epoch, and the FL process was conducted during five rounds. The results obtained in the four setups (mean and standard deviation after repeating the experiments five times) are listed in Table 3.

Regarding the client-side resources used in these scenarios, in AI4EOSC, we deployed four instances, each configured with 4 vCPUs, 10 GB of RAM, and 20 GB of disk space. These specifications were chosen to ensure comparability with the selected flavor in the four VMs, matching the number of vCPUs and disk capacity, and providing a similar amount of RAM (10.7 GB). In all cases, the same version of the FL server and the dependencies for running the clients were used (including Flower extensions).

As shown in Table 3, running the FL server in the AI4EOSC environment does not introduce a significant time overhead. Specifically, the difference compared to the best-case scenario (server on a VM and clients on AI4EOSC, with an average of 222.61 s) is less than 3% when the server runs on AI4EOSC with clients also on AI4EOSC, and does not exceed 5% in the worst-case scenario (server on AI4EOSC and clients on a VM). Furthermore, when the server runs on AI4EOSC, better results are obtained by running the clients in that environment, representing an improvement of nearly 2%. However, this difference remains minimal in absolute terms. Furthermore, the best average performance was achieved when the server ran on a VM and the clients ran on AI4EOSC (222.61 s). Nevertheless, considering the standard deviations, this result is very similar to the scenario in which both the server and clients run on VMs within the same cloud infrastructure (226.96 s), with a difference of less than 2%.

Finally, these results indicate that using the AI4EOSC platform in this FL setting does not introduce significant computational overhead, with training times remaining comparable across configurations.

5.4. Cold-start

Cold-start latency is observed mainly in the first invocations after service deployment or scale-up, when the platform must complete additional preparation steps such as image handling, container startup, and initialization before the request can be served. Cold start is particularly important for the AIaaS serverless component (based on OSCAR), as it handles short-lived inference calls (spawn container, make an inference, kill the container). While execution times are application and container image-dependent, for a reference implementation, synchronous parallel OSCAR invocations are complete with sub-second latency, indicating that the platform can sustain interactive request–response workloads when the service is already warm and resources are available. This was achieved in a 3-node OSCAR cluster (24 cores and 54 GBs of RAM available), showing an average first synchronous invocation time of 1521 ms and warm synchronous invocations of 992 ms.

Cold-start latency is not critical in the WMS, because the WMS is meant to execute longer jobs and tasks, such as AI training jobs, and thus user experience is not meaningfully impacted by warm/cold start delays (typically in the few seconds if the image is pre-cached in the node, to tens of seconds if the full image has to be pulled from the AI4EOSC container registry).

6. Use cases and usage scenarios

6.1. End-to-end user journey: precipitation nowcasting with federated learning

To better illustrate how a researcher can interact with the AI4EOSC platform across the full AI/ML life-cycle, in this section, we describe a concrete use case focused on radar-based precipitation nowcasting [56]. This use case aims to represent a broad class of scientific scenarios in which training data are distributed across institutions and cannot be centralized due to technical, legal, or administrative constraints.

Problem and data. The goal of the selected case is to predict precipitation, expressed as Vertically Integrated Liquid (VIL), in a five-minute horizon, using sequences of radar images captured by a meteorological radar located in the borderland between the Czech Republic and Slovakia. The dataset covers four months of radar observations (between April and July 2016), with images captured every five minutes and stored as HDF5 files. To simulate a realistic federated scenario, the images were divided into four quadrants, each treated as a separate client with a distinct and heterogeneous data distribution. The goal of this case is to validate the feasibility of using federated learning for such predictions, where data is not directly accessible.

Data were stored in the storage system of the platform, which can be securely accessed from user training jobs. In a realistic scenario, users would have been able to link their external storage system with their accounts, so that their datasets were not centrally stored but rather accessed via their own storage systems only during training. As outlined in Section 4.4, data are not retained after the training process is finished.

Development. The researchers deployed a Codespace via the AI4EOSC Dashboard, selecting a TensorFlow environment running on an NVIDIA Tesla T4 GPU (16 GB RAM, 8 CPU cores, 10 GB disk). The development environment provided secure and authenticated access to version-controlled repositories and the MLflow experiment-tracking service, allowing the team to log and compare training runs across different model configurations and hyperparameter settings.

Federated training. Once a suitable convolutional neural network architecture was identified and validated, the team configured a Flower federated learning server via the dashboard, deploying four client instances within the platform itself, one corresponding to each zone. Token-based authentication was used to ensure that only authorized clients participated in the training. The FL process ran for ten rounds of ten epochs each. Following the federated phase, a novel personalized FL approach (*adapFL*) was applied, further fine-tuning the aggregated global model locally on each zone's data for a set of ten additional epochs. This adaptive step allows each client to recover the zone-specific characteristics lost during federated aggregation.

Deployment and results. In order to quickly validate the results, the generated models were deployed as permanent WMS endpoints for prototyping and evaluation, and as serverless AIaaS endpoints via OSCAR for on-demand inference. The platform CI/CD pipeline automatically built and published the corresponding Docker images, validated the module metadata, and triggered provenance graph updates, ensuring full traceability of the training process without any additional intervention from researchers. The *adapFL* approach yielded improvements in terms of both metrics evaluated (MSE and MAE) over individual training and vanilla FL in all four zones, demonstrating that privacy-preserving collaborative training on heterogeneously distributed data can yield better predictive models than local training alone.

6.2. Platform adoption

Besides building a generic and horizontal platform and fostering the adoption of the technology developed within multidisciplinary domains, we have closely worked with several initiatives. The aim is for these communities to collaborate on the co-design of the platform to build a customized version that adapts to the needs of the user community. The interactions with those communities have led to a virtuous circle, where the platform has guided them on their adoption of AI, and they have in turn provided feedback for user-driven platform development.

6.2.1. iMagine

The iMagine[8] initiative is a European Union-funded research project that focuses on developing and operating an AI-enabled imaging platform for aquatic science. It aims to give researchers involved in the study of oceans, seas, coastal, and inland waters access to rich image datasets, advanced AI-powered analysis tools, and best-practice guidance for scientific image processing. The iMagine platform was built on top of the software stack developed within the AI4EOSC project.

Throughout the iMagine project (formally concluded in August 2025), the Competence Center team provided support to the users and collaborated closely with the developers of the AI4EOSC project, providing feedback to enhance the platform. This partnership enabled hands-on experience across the full AI workflow, from image collection and dataset preparation to model training and deployment for aquatic science applications, ultimately resulting in the publication [57]. It is important to note that the project has developed several use cases built on top of the services provided within the platform, including those presented in [58] (focusing on satellite-derived bathymetry mapping) and [59] (focusing on distance classification of marine vessels), among others.

6.2.2. KMD4EOSC

KMD4EOSC[9] is a Polish national project for data storage and sharing, and efficient processing of large volumes of data in HPC, BigData, and AI Models. The project is co-funded by the European Funds for a Modern Economy Program and the European Union. This project is one of the flagship projects of the Polish Research Roadmap.

The KMD4EOSC project designs, builds, and implements infrastructure, services, and applications for storing and collecting scientific and economic data, as well as for processing and analyzing them and making them available to the scientific community and the economy in an efficient manner and in accordance with good practices, requirements, and regulations. AI4EOSC has been deployed within the project to serve as a platform for providing the whole life-cycle of AI/ML models, integrated with repositories, datasets, and lower infrastructure and data management services.

6.2.3. AI4Life

AI4Life,[10] a Horizon Europe-funded project to empower life science researchers to harness the full potential of AI/ML methods for bioimage analysis, began collaborating with AI4EOSC in order to leverage the AI4EOSC software stack and platform. The AI4EOSC and AI4Life teams worked on the integration of both platforms as a means to implement and pilot a model for interoperability across research infrastructures aligned with FAIR principles. This integration enables efficient sharing and use of AI/ML models and computing resources across domains.

Central to this effort is deploying models from the BioImage Model Zoo in the AI4Life cloud marketplace. Compatibility was ensured through defined execution environments such as PyTorch, standardized metadata rich input and output formats, and consistent prediction pipelines, allowing deployment via Dockerfile and DEEPaaS-compatible APIs.

7. Conclusion and future work

The growing adoption of AI and ML in scientific research demands platforms that go beyond industrial MLOps frameworks to address the specific needs of open science: reproducibility, FAIR compliance, provenance tracking, and seamless integration with federated research infrastructures to enable easy access to the technologies by researchers. This paper presented AI4EOSC, a federated, open-source platform that operationalizes these principles across the full AI/ML life-cycle within the European Open Science Cloud ecosystem.

The main architectural contribution of AI4EOSC is not the provision of any single or standalone tool, but rather the integration of development, training, deployment, and provenance into a unified, FAIR-by-design system that enforces open science compliance at the infrastructure level, alleviating the burden from individual researchers. This integration is not merely a technical assembly, but a methodological response to the lack of infrastructure-level FAIR enforcement in distributed research environments. AI4EOSC's main innovations include a lightweight inter-cluster federation based on Consul and Nomad that avoids the infrastructure fragmentation and silos of Kubernetes-centric stacks, an automated W3C PROV-compliant provenance pipeline that captures the full model lineage with limited user intervention, and extended federated learning support with token-based authentication, novel aggregation strategies, and server-side differential privacy. The AI4EOSC value proposition has been validated through three real-world community adoptions: iMagine, KMD4EOSC, and AI4Life. This demonstrates consistent functionality and deployment across heterogeneous resource providers and diverse scientific domains.

[8] https://www.imagine-ai.eu/

[9] https://kmd4eosc.pl/

[10] https://ai4life.eurobioimaging.eu/

Our evaluation demonstrated the efficiency of the AI4EOSC. However, a broader comparative benchmark against industrial MLOps stacks in a multi-tenant, federated, multi-country context is a significant challenge, as most industry-standard tools are designed for single-cluster environments. This situation makes direct performance comparisons difficult without substantial custom modifications to these framework codebases. Future work can focus on developing cross-platform benchmarks for federated scientific workloads.

Nevertheless, several limitations need to be acknowledged. Firstly, the current provenance model, while comprehensive for single-model workflows, requires further work to handle multi-model inference pipelines and federated learning scenarios where data remains siloed across institutions. Secondly, although the platform currently supports semantic interoperability through MLDCAT-AP serialization, full machine-actionability of AI/ML assets across heterogeneous repositories and data spaces remains an open and evolving challenge: deeper integration with emerging standards and external catalogs is needed to allow assets to be not only findable and accessible, but autonomously actionable by external systems. Finally, the operational overhead of maintaining a geographically distributed federation (including failure handling, performance variability across heterogeneous providers, and long-term governance) represents a practical challenge that will grow as the federation expands. This fact is a common issue of any geographically distributed and complex infrastructure, and AI4EOSC alleviates this issue by leveraging a single and unified control plane.

Future work in the near term will focus on deepening semantic interoperability and machine actionability by extending the current support for MLDCAT-AP, and on accommodating emerging standards, such as Croissant-ML [60]. Moreover, we plan to implement the OAI-PMH protocol to enable automated metadata harvesting across external marketplaces and repositories. We will also extend the MLflow experiment tracking component to serve as a full model registry, enriching the stored models with semantic provenance information, as we are already doing with the modules published through the dashboard. In the medium term, we plan to address the provenance gaps identified above, particularly for federated and multi-model scenarios, and to strengthen the platform's energy and sustainability awareness by integrating fine-grained carbon and water footprint measurements via Wattnet [61], implementing platform level optimizations, and informing users about the impact of their workloads. In the longer term, we aim to grow the current federation by lowering the barrier for new resource providers to join and to conduct systematic user studies to measure the actual impact of the platform on researcher productivity and reproducibility outcomes.

CRediT authorship contribution statement

Ignacio Heredia: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Álvaro López García:** Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Fernando Aguilar Gómez:** Writing – review & editing, Validation, Software, Investigation, Conceptualization. **Diego Aguirre:** Software. **Caterina Alarcón Marín:** Software. **Khadijeh Alibabaei:** Writing – review & editing, Validation, Software, Investigation. **Lisana Berberi:** Writing – review & editing, Validation, Software, Investigation. **Miguel Caballer:** Software. **Amanda Calatrava:** Writing – review & editing, Software, Conceptualization. **Pedro Castro:** Software. **Alessandro Costantini:** Writing – review & editing, Validation, Software, Resources, Methodology. **Mario David:** Writing – review & editing, Validation, Resources. **Jaime Díez:** Software. **Stefan Dlugolinsky:** Software. **Giacinto Donvito:** Writing – review & editing, Validation, Software, Resources, Methodology. **Leonhard Duda:** Software. **Borja Esteban Sanchis:** Software. **Saúl Fernandez Tobías:** Software. **Andrés Heredia Canales:** Visualization, Software. **Valentin Kozlov:** Writing – review & editing, Validation, Software, Investigation, Conceptualization. **Sergio Langarita:** Software. **João Machado:** Writing – review & editing, Resources. **Daniel San Martín:** Software. **Germán Moltó:** Writing – review & editing, Supervision, Software, Investigation. **Giang Nguyen:** Writing – review & editing, Writing – original draft, Investigation, Formal analysis. **Marta Obregón Ruiz:** Writing – original draft, Visualization, Software. **Marcin Płóciennik:** Writing – review & editing, Supervision, Resources. **Susana Rebolledo Ruiz:** Software. **Vicente Rodriguez:** Software. **Judith Sáinz-Pardo Díaz:** Writing – review & editing, Software, Methodology, Investigation, Formal analysis. **Martin Šeleng:** Software, Resources. **Viet Tran:** Supervision, Software, Resources, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Given their role as journal Editor, Álvaro López García had no involvement in the peer-review of this article and has no access to information regarding its peer-review. Full responsibility for the editorial process for this article was delegated to another journal editor. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors acknowledge Marcos Lloret Iglesias, the AI4EOSC project manager, for his constant support and help towards bringing the project to fruition. The authors acknowledge funding and support from the AI4EOSC project "Artificial Intelligence for the European Open Science Cloud", that has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement number 101058593. ALG also acknowledges the support from the *Consejería de Educación, Formación Profesional y Universidades* of the *Gobierno de Cantabria* via the "*Actividad estructural para el desarrollo de la investigación del Instituto de Física de Cantabria*" project.

Data availability

No data was used for the research described in the article.

References

[1] Y. Chervonyi, T.H. Trinh, M. Olšák, X. Yang, H.H. Nguyen, M. Menegali, J. Jung, J. Kim, V. Verma, Q.V. Le, T. Luong, Gold-medalist performance in solving olympiad geometry with AlphaGeometry2, J. Mach. Learn. Res. 26 (241) (2025) 1–39, URL: http://jmlr.org/papers/v26/25-1654.html.

[2] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al., Highly accurate protein structure prediction with AlphaFold, Nature 596 (7873) (2021) 583–589, http://dx.doi.org/10.1038/s41586-021-03819-2.

[3] L. Espeholt, S. Agrawal, C. Sønderby, M. Kumar, J. Heek, C. Bromberg, C. Gazen, R. Carver, M. Andrychowicz, J. Hickey, A. Bell, N. Kalchbrenner, Deep learning for twelve hour precipitation forecasts, Nat. Commun. 13 (1) (2022) 5145, http://dx.doi.org/10.1038/s41467-022-32483-x.

[4] H.O. Khogali, S. Mekid, The blended future of automation and AI: Examining some long-term societal and ethical impact features, Technol. Soc. 73 (2023) 102232, http://dx.doi.org/10.1016/j.techsoc.2023.102232.

[5] A. Sergeyuk, Y. Golubev, T. Bryksin, I. Ahmed, Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward, Inf. Softw. Technol. 178 (2025) 107610, http://dx.doi.org/10.1016/j.infsof.2024.107610.

[6] Q. Hao, F. Xu, Y. Li, J. Evans, Artificial intelligence tools expand scientists' impact but contract science's focus, Nature 649 (8099) (2026) 1237–1243, http://dx.doi.org/10.1038/s41586-025-09922-y.

[7] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L.B. da Silva Santos, P.E. Bourne, et al., The FAIR guiding principles for scientific data management and stewardship, Sci. Data 3 (1) (2016) 1–9, http://dx.doi.org/10.1038/sdata.2016.18.

[8] L. Berberi, V. Kozlov, G. Nguyen, J. Sáinz-Pardo Díaz, A. Calatrava, G. Moltó, V. Tran, Á. López García, Machine learning operations landscape: platforms and tools, Artif. Intell. Rev. 58 (6) (2025) 167, http://dx.doi.org/10.1007/s10462-025-11164-3.

[9] Directorate-General for Research and Innovation (European Commission), EOSC Executive Board, Strategic research and innovation agenda (SRIA) of the European open science cloud (EOSC), Publications Office of the European Union, 2022, http://dx.doi.org/10.2777/935288.

[10] Kubeflow: Machine learning toolkit for kubernetes, 2018, URL: https://www.kubeflow.org/.

[11] Polyaxon, Polyaxon, 2025, URL: https://polyaxon.com/.

[12] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S.A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, et al., Accelerating the machine learning lifecycle with mlflow, IEEE Data Eng. Bull. 41 (4) (2018) 39–45, URL: https://www.usenix.org/conference/opml19/presentation/tut5.

[13] V. Jayasiri, N. Wijerathne, A. Narasinghe, L. Nishshanke, Labml.AI: A library to organize machine learning experiments, 2020, URL: https://labml.ai/.

[14] CoreWeave, Weights and biases, 2020, URL: https://www.wandb.com/.

[15] Valohai, 2025, URL: https://valohai.com/.

[16] Comet, 2025, URL: https://www.comet.com/.

[17] H2O.AI, 2025, URL: https://h2o.ai/.

[18] Alphabet, Google vertex AI, 2025, URL: https://cloud.google.com/vertex-ai.

[19] Amazon, Amazon sagemaker, 2025, URL: https://aws.amazon.com/sagemaker/.

[20] Microsoft, Azure AI machine learning studio, 2025, URL: https://ml.azure.com/.

[21] Seldon Technologies Ltd., Seldon core: Open source platform for deploying machine learning models on kubernetes, 2015, URL: https://www.seldon.io/tech/seldon-core.

[22] Hugging face – the AI community building the future., 2026, URL: https://huggingface.co/.

[23] B. Bischl, G. Casalicchio, T. Das, M. Feurer, S. Fischer, P. Gijsbers, S. Mukherjee, A.C. Müller, L. Németh, L. Oala, L. Purucker, S. Ravi, J.N.v. Rijn, P. Singh, J. Vanschoren, J.v.d. Velde, M. Wever, OpenML: Insights from 10 years and more than a thousand papers, Patterns 6 (7) (2025) 101317, http://dx.doi.org/10.1016/j.patter.2025.101317.

[24] W. Ouyang, F. Beuttenmueller, E. Gómez-de Mariscal, C. Pape, T. Burke, C. Garcia-López-de Haro, C. Russell, L. Moya-Sans, C. De-La-Torre-Gutiérrez, D. Schmidt, et al., Bioimage model zoo: A community-driven resource for accessible deep learning in bioimage analysis, 2022, http://dx.doi.org/10.1101/2022.06.07.495102, BioRxiv.

[25] Ž. Avsec, R. Kreuzhuber, J. Israeli, N. Xu, J. Cheng, A. Shrikumar, A. Banerjee, D.S. Kim, T. Beier, L. Urban, A. Kundaje, O. Stegle, J. Gagneur, The kipoi repository accelerates community exchange and reuse of predictive models for genomics, Nature Biotechnol. 37 (6) (2019) 592–600, http://dx.doi.org/10.1038/s41587-019-0140-0.

[26] G. Padovani, V. Anantharaj, S. Fiore, Provenance tracking in large-scale machine learning systems, in: Workshop Proceedings of the 54th International Conference on Parallel Processing, in: ICPP Workshops '25, Association for Computing Machinery, New York, NY, USA, 2025, pp. 167–174, http://dx.doi.org/10.1145/3750720.3757295.

[27] A. Lopez Garcia, J.M. De Lucas, M. Antonacci, W. Zu Castell, M. David, M. Hardt, L. Lloret Iglesias, G. Molto, M. Plociennik, V. Tran, A.S. Alic, M. Caballer, I.C. Plasencia, A. Costantini, S. Dlugolinsky, D.C. Duma, G. Donvito, J. Gomes, I. Heredia Cacha, K. Ito, V.Y. Kozlov, G. Nguyen, P. Orviz Fernandez, Z. Sustr, P. Wolniewicz, A cloud-based framework for machine learning workloads and applications, IEEE Access 8 (2020) 18681–18692, http://dx.doi.org/10.1109/ACCESS.2020.2964386.

[28] ESFRI-EOSC Task Force, FAIR data productivity and advanced digitalization of research: An opinion paper by the ESFRI-EOSC task force and steering board expert group (e03756), 2024, http://dx.doi.org/10.5281/zenodo.10980285.

[29] S. Brown, The C4 Model, Oreilly, 2026, URL: https://www.oreilly.com/library/view/the-c4-model/9798341660113/.

[30] A. Lopez Garcia, L. Berberi, S. Langarita, M. Antonacci, A. Calatrava, G. Moltó, I. Heredia, A.H. Canales, AI4eosc/ai4-architecture: 2.0.0, 2024, http://dx.doi.org/10.5281/zenodo.13343448.

[31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., TensorFlow: a system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation, (OSDI 16), 2016, pp. 265–283, http://dx.doi.org/10.48550/arXiv.1605.08695.

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, 2019, http://dx.doi.org/10.48550/arXiv.1912.01703, arXiv:1912.01703 [Cs, Stat], arXiv:1912.01703.

[33] G. Nguyen, J. Sáinz-Pardo Díaz, A. Calatrava, L. Berberi, O. Lytvyn, V. Kozlov, V. Tran, G. Moltó, Á. López García, Landscape of machine learning evolution: privacy-preserving federated learning frameworks and tools, Artif. Intell. Rev. 58 (2) (2024) 51, http://dx.doi.org/10.1007/s10462-024-11036-2.

[34] D.J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K.H. Li, T. Parcollet, P.P.B. de Gusmão, et al., Flower: A friendly federated learning research framework, 2020, http://dx.doi.org/10.48550/arXiv.2007.14390, arXiv preprint arXiv:2007.14390.

[35] H.R. Roth, Y. Cheng, Y. Wen, I. Yang, Z. Xu, Y.-T. Hsieh, K. Kersten, A. Harouni, C. Zhao, K. Lu, et al., NVIDIA FLARE: Federated learning from simulation to real-world, 2022, http://dx.doi.org/10.48550/arXiv.2210.13291, arXiv preprint arXiv:2210.13291.

[36] J. Sáinz-Pardo Díaz, A. Heredia Canales, I. Heredia Cachá, V. Tran, G. Nguyen, K. Alibabaei, M. Obregón Ruiz, S. Rebolledo Ruiz, A. López García, Making federated learning accessible to scientists: The AI4EOSC approach, in: Proceedings of the 2024 ACM Workshop on Information Hiding and Multimedia Security, in: IH&MMSec '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 253–264, http://dx.doi.org/10.1145/3658664.3659642.

[37] J.S.-P. Díaz, A. Athanasiou, K. Jung, C. Palamidessi, Á.L. García, Metric-privacy-inspired noise calibration in federated learning: Improving convergence and preventing client inference attacks, Knowl.-Based Syst. 343 (2026) 115993, http://dx.doi.org/10.1016/j.knosys.2026.115993.

[38] J.S.-P. Díaz, Á.L. García, Enhancing the convergence of federated learning aggregation strategies with limited data, in: 2025 3rd International Conference on Federated Learning Technologies and Applications, FLTA, IEEE, Dubrovnik, Croatia, 2025, pp. 9–16, http://dx.doi.org/10.1109/FLTA67013.2025.11336682.

[39] A. Van Looveren, J. Klaise, G. Vacanti, O. Cobb, A. Scillitoe, R. Samoilescu, A. Athorne, Alibi detect: Algorithms for outlier, adversarial and drift detection, 2025, URL: https://github.com/SeldonIO/alibi-detect.

[40] J. Céspedes Sisniega, Á. López García, Frouros: An open-source Python library for drift detection in machine learning systems, SoftwareX 26 (2024) 101733, http://dx.doi.org/10.1016/j.softx.2024.101733.

[41] J. Céspedes Sisniega, V. Rodríguez, G. Moltó, Á. López García, Efficient and scalable covariate drift detection in machine learning systems with serverless computing, Future Gener. Comput. Syst. 161 (2024) 174–188, http://dx.doi.org/10.1016/j.future.2024.07.010, URL: https://www.sciencedirect.com/science/article/pii/S0167739X24003716.

[42] A. Pérez, S. Risco, D.M. Naranjo, M. Caballer, G. Moltó, On-premises serverless computing for event-driven data processing applications, in: 2019 IEEE 12th International Conference on Cloud Computing, CLOUD, IEEE, (ISSN: 2159-6190) 2019, pp. 414–421, http://dx.doi.org/10.1109/CLOUD.2019.00073, URL: https://ieeexplore.ieee.org/document/8814513/.

[43] Á. López García, DEEPaaS API: A REST API for machine learning and deep learning models, J. Open Source Softw. 4 (42) (2019) 1517, http://dx.doi.org/10.21105/joss.01517.

[44] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C.H. Yu, J. Gonzalez, H. Zhang, I. Stoica, Efficient memory management for large language model serving with PagedAttention, in: Proceedings of the 29th Symposium on Operating Systems Principles, ACM, Koblenz Germany, 2023, pp. 611–626, http://dx.doi.org/10.1145/3600006.3613165.

[45] A. Costantini, D.C. Duma, B. Martelli, M. Antonacci, M. Galletti, S.R. Tisbeni, P. Bellavista, G. Di Modica, D. Nehls, J.-C. Ahouangonou, C. Delamarre, D. Cesini, A cloud-edge orchestration platform for the innovative industrial scenarios of the IoTwins project, in: O. Gervasi, B. Murgante, S. Misra, C. Garau, I. Blečić, D. Taniar, B.O. Apduhan, A.M.A. Rocha, E. Tarantino, C.M. Torre (Eds.), in: Computational Science and Its Applications – ICCSA 2021, vol. 12950, Springer International Publishing, Cham, 2021, pp. 533–543, http://dx.doi.org/10.1007/978-3-030-86960-1_37.

[46] D. Salomoni, I. Campos, L. Gaido, J.M. De Lucas, P. Solagna, J. Gomes, L. Matyska, P. Fuhrman, M. Hardt, G. Donvito, L. Dutka, M. Plociennik, R. Barbera, I. Blanquer, A. Ceccanti, E. Cetinic, M. David, C. Duma, A. López-García, G. Moltó, P. Orviz, Z. Sustr, M. Viljoen, F. Aguilar, L. Alves, M. Antonacci, L.A. Antonelli, S. Bagnasco, A.M.J.J. Bonvin, R. Bruno, Y. Chen, A. Costa, D. Davidovic, B. Ertl, M. Fargetta, S. Fiore, S. Gallozzi, Z. Kurkcuoglu, L. Lloret, J. Martins, A. Nuzzo, P. Nassisi, C. Palazzo, J. Pina, E. Sciacca, D. Spiga, M. Tangaro, M. Urbaniak, S. Vallero, B. Wegh, V. Zaccolo, F. Zambelli, T. Zok, INDIGO-DataCloud: A platform to facilitate seamless access to E-infrastructures, J. Grid Comput. 16 (3) (2018) 381–408, http://dx.doi.org/10.1007/s10723-018-9453-3.

[47] M. Caballer, I. Blanquer, G. Moltó, C. De Alfonso, Dynamic management of virtual infrastructures, J. Grid Comput. 13 (1) (2015) 53–70, http://dx.doi.org/10.1007/s10723-014-9296-5.

[48] M. Caballer, G. Moltó, A. Calatrava, I. Blanquer, Infrastructure manager: A TOSCA-based orchestrator for the computing continuum, J. Grid Comput. 21 (3) (2023) 51, http://dx.doi.org/10.1007/s10723-023-09686-7.

[49] L. Giommi, G. Savarese, G. Vino, D. Ranieri, A. Costantini, G. Donvito, Improving the cloud provider ranking in the INDIGO paas orchestration system using AI techniques, in: O. Gervasi, et al. (Eds.), Computational Science and Its Applications – ICCSA 2025 Workshops, ICCSA 2025, in: Lecture Notes in Computer Science, vol. 15886, Springer, 2025, http://dx.doi.org/10.1007/978-3-031-97576-9_3.

[50] G. Savarese, M. Antonacci, L. Giommi, Federation-registry: The renovated configuration management database for dynamic cloud federation, PoS ISGC2024 (2024) http://dx.doi.org/10.22323/1.458.0021.

[51] D. Palma, M. Rutkowski, T. Spatzier, TOSCA simple profile in YAML version 1.0, 2015, OASIS Committee Specification Draft 04 / Public Review Draft 01, OASIS, URL: http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csprd01/TOSCA-Simple-Profile-YAML-v1.0-csprd01.html, Version Number: 1.0.

[52] M. Caballer, S. Zala, Á.L. García, G. Moltó, P.O. Fernández, M. Velten, Orchestrating complex application architectures in heterogeneous clouds, J. Grid Comput. 16 (1) (2018) 3–18, http://dx.doi.org/10.1007/s10723-017-9418-y.

[53] L.J. Castro, D. Garijo, D. Rebholz-Schuhmann, D. Solanki, J.T. Ciuciu-Kiss, D. Katz, L. Eklund, G. Bharathy, R.D.A.F.T. Force, FAIR4ml-schema, 2024, http://dx.doi.org/10.5281/zenodo.14002310, Version Number: 0.1.0.

[54] J.d. Bruin, Datahugger - One downloader for many scientific repositories, 2024, http://dx.doi.org/10.5281/zenodo.8370403.

[55] AARC Community members, AARC blueprint architecture 2019 (AARC-G045), 2019, http://dx.doi.org/10.5281/zenodo.3672785.

[56] J. Sáinz-Pardo Díaz, M. Castrillo, J. Bartok, I.H. Cachá, I.M. Ondík, I. Martynovskyi, K. Alibabaei, L. Berberi, V. Kozlov, Á. López García, Personalized federated learning for improving radar based precipitation nowcasting on heterogeneous areas, Earth Sci. Inform. 17 (6) (2024) 5561–5584, http://dx.doi.org/10.1007/s12145-024-01438-9.

[57] E. Azmi, K. Alibabaei, V. Kozlov, T. Krijger, G. Accarino, S.-D. Ayata, A. Calatrava, M.M. De Carlo, W. Decrop, D. Elia, S.L. Fiore, M. Francescangeli, J.-O. Irisson, R. Lagaisse, M. Laviale, A. Lebeaud, C. Leluschko, E. Martínez, G. Moltó, I. Ruiz Atake, A.A. Sepp Neves, D. Smyth, J. Soriano-González, M.A. Tayyab, V. Tosello, Á. López García, D. Schaap, G. Sipos, Best practices for AI-based image analysis applications in aquatic sciences: The iMagine case study, Ecol. Informatics 91 (2025) 103306, http://dx.doi.org/10.1016/j.ecoinf.2025.103306.

[58] D. García-Díaz, S.P. Viaña-Borja, M. Roca, G. Navarro, I. Caballero, Blending physical and artificial intelligence models to improve satellite-derived bathymetry mapping, Ecol. Inform. 90 (2025) 103328, http://dx.doi.org/10.1016/j.ecoinf.2025.103328.

[59] W. Decrop, K. Deneudt, C. Parcerisas, E. Schall, E. Debusschere, Transfer learning for distance classification of marine vessels using underwater sound, IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens. 18 (2025) 19710–19726, http://dx.doi.org/10.1109/JSTARS.2025.3593779.

[60] M. Akhtar, O. Benjelloun, C. Conforti, P. Gijsbers, J. Giner-Miguelez, N. Jain, M. Kuchnik, Q. Lhoest, P. Marcenac, M. Maskey, P. Mattson, L. Oala, P. Ruyssen, R. Shinde, E. Simperl, G. Thomas, S. Tykhonov, J. Vanschoren, J. Van Der Velde, S. Vogler, C.-J. Wu, Croissant: A metadata format for ML-ready datasets, in: Proceedings of the Eighth Workshop on Data Management for End-To-End Machine Learning, ACM, Santiago AA Chile, 2024, pp. 1–6, http://dx.doi.org/10.1145/3650203.3663326.

[61] M. Castrillo, J. Iglesias Blanco, J.S.-P. Díaz, Á. López García, Wattnet: matching electricity consumption with low-carbon, low-water footprint energy supply, 2026, http://dx.doi.org/10.5281/zenodo.17404776, URL: https://arxiv.org/abs/2601.11623.