D4.2 Data management, Intelligence and Security Layers (IT-2)

Document Identification			
Status	Final	Due Date	28/02/2025
Version	1.0	Submission Date	21/02/2025

Related WP	WP4	Document Reference	D4.2
Related Deliverable(s)	D4.1, D2.3, D2.4, D3.2	Dissemination Level (*)	PU - Public
Lead Participant	CeADAR	Lead Author	Andrés L. Suárez-Cetrulo (CeADAR)
Contributors	XLAB, BSC, ZSCALE, NKUA, UPC, NCSRD, ENG, TUBS	Reviewers	Marc Michalke (TUBS) Konstantinos Latanis (Suite5)

Keywords:
ICOS, Data Management Layer, Intelligence Layer, Security Layer

This document is issued within the frame and for the purpose of the ICOS project. This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No. 101070177. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

(*) Dissemination level: **(PU)** Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page). **(SEN)** Sensitive, limited under the conditions of the Grant Agreement. **(Classified EU-R)** EU RESTRICTED under the Commission Decision No2015/444. **(Classified EU-C)** EU CONFIDENTIAL under the Commission Decision No2015/444. **(Classified EU-S)** EU SECRET under the Commission Decision No2015/444.

Document Information

List of Contributors	
Name	Partner
Andrés L. Suárez-Cetrulo	CeADAR
Jaydeep Samanta	CeADAR
Sebastián Andrés Cajas Ordoñez	CeADAR
Romila Ghosh	CeADAR
Ricardo Simón Carbajo	CeADAR
Hrvoje Ratkajec	XLAB
Daniel Nikoloski	XLAB
Justin Činkelj	XLAB
Alex Barceló	BSC
Anastasios Giannopoulos	NKUA
Menelaos Zetas	NKUA
Anastasios Kaltakis	NKUA
Ivan Paez	ZSCALE
Gabriele Giammatteo	ENG
Maria Antonietta Di Girolamo	ENG
Vladan Velickovic	ENG
Marc Michalke	TUBS
Fin Gentzen	TUBS
Marla Grunewald	TUBS
Jordi García	UPC
Andreas Oikonomakis	NCSRD

Document History			
Version	Date	Change editors	Changes
0.1	21/11/2024	CeADAR	First draft version of ToC
0.2	18/12/2024	CeADAR. BSC, XLAB, ZSCALE	Updated ToC
0.3	21/01/2024	CeADAR, NKUA, XLAB, Zetta, ENG, NCSRD, ZSCALE, BSC	First round of contributions
0.4	29/01/2025	CeADAR	Review contributions and feedback to partners
0.5	03/02/2025	CeADAR, NKUA, XLAB, Zetta, ENG, NCSRD, TUBS, ZSCALE, BSC, UPC	Second round of contributions after first review
0.6	12/02/2025	CeADAR	Combined corrections from partners and version ready for internal review
0.7	18/02/2025	CeADAR	Integrated internal review feedback.
0.8	19/02/2025	CeADAR	Version ready for final quality check
1.0	21/02/2025	ATOS	Final version to be submitted

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Andrés L. Suárez-Cetrulo (CeADAR)	19/02/2025
Quality manager	Carmen San Román (ATOS)	21/02/2025
Project Coordinator	Francesco D'Andria (ATOS)	21/02/2025

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
AI	Artificial Intelligence
AIOps	AI Operations
API	Application Programming Interface
CPU	Central Processing Unit
DRL	Deep Reinforcement Learning
DDRL	Decentralised Deep Reinforcement Learning
Dx.y	Deliverable number y belonging to WP x
FL	Federated Learning
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IAM	Identity and Access Management
IT-1	First iteration
IT-2	Second iteration
JSON	JavaScript Object Notation
JWT	JSON Web Token
LSTM	Long Short-Term Memory
ML	Machine Learning
MLOps	Machine Learning Operations, used interchangeably with
MSE	Mean Squared Error
OIDC	OpenID Connect
PIP	Pypi Python library
REST	Representational state transfer
RNN	Recurrent Neural Network
TLS	Transport Layer Security
mTLS	mutual TLS
UI	User Interface
WP	Work Package
WPx	Work Package number x
YAML	yet another markup language

Executive Summary

This document is the implementation deliverable of the ICOS components Data Management, Intelligence Layer Module, Security Layer Module, and APIs and Interfaces in IT-2. The document, together with its annexes, presents the following aspects of the Data Management, Intelligence and Security Layers: a) their functionalities, b) prototype architecture and how they fit into the general ICOS architecture, c) their technical description and specifications, d) delivery (package, installation, download) and licensing information, and any limitations and future work to be done for the Final ICOS release.

In IT-2, the Data Management component continues to play a critical role in optimising data access and distribution across ICOS layers while maintaining infrastructure transparency. Building upon the foundation established in D4.1, this iteration enhances software architecture and integration with other ICOS components. The Data Management Layer addresses key data exchange needs, including transport, mutable data structures, and compute-intensive operations. Leveraging Eclipse Zenoh for communication and dataClay for distributed storage, it supports functionalities such as the ICOS data bus, training offloading, telemetry processing, and policy storage.

The Intelligence Layer in IT-2 includes the AI Coordination module, AI analytics, Trustworthy AI modules, Data Processing module, and AI Models and Data Repository. The AI coordination module has been extended to include a frontend to interact with the ICOS Meta-OS. This frontend, reachable from the ICOS CLI, aims to query telemetry to realise when new system utilisation or energy-related metrics need to be predicted and then requests training a model or regular predictions to the AI coordination backend presented in IT-1; furthermore, it allows ICOS users to request new metrics forecasts through an API (Export Metrics API). The AI coordination backend communicates with the Data Processing module, which is in charge of data access and manipulation, and the AI analytics module that builds and deploys AI models (support for multivariate modelling in IT-2 and model compression to allow AI workloads at the edge), the Trustworthy AI module, providing model explainability, monitoring, and privacy-aware model training (Federated Learning), and will connect to the AI Model and Data Repository, to allow reusing models from the online repository.

The Security Layer in IT-2 builds upon the foundation established in IT-1, incorporating advancements in identity management, anomaly detection, auditing, and trust. The Security Layer Coordination module facilitates seamless integration between security components and other ICOS modules. The Identity and Access Management (IAM) module ensures secure authentication and authorisation across the system. The Security Scan module actively monitors deployed ICOS infrastructure for vulnerabilities and compliance issues, while the Anomaly Detection module leverages AI to identify potential security threats. Additionally, the Audit module enhances system transparency, and Trust functionality guarantees the identity of ICOS components and enforces encrypted, verifiable communication between them.

All presented software will be used in the final iteration of ICOS (Complete ICOS version). Components under the future work section will still be reviewed and updated in the D5.3 – Complete ICOS version (M32) [8] as their integration is in progress at the time of writing this deliverable (M30).

1 Introduction

1.1 Purpose of the document

The purpose of this document is to report design and implementation efforts done in WP4 in tasks T4.1 (*Data Management*), T4.2 (*Intelligence Layer Module*), T4.3 (*Security Layer Module*) and T4.4 (*APIs and Interfaces*) regarding the development of ICOS components Data Management, Intelligence and Security Layers in the period M14-M30. It also delivers the second version (IT-2) of the Data Management, Intelligence Layer and Security Layer software using Deliverable 4.1 – "*Data management, Intelligence and Security Layers (IT-1)*" as input.

In this regard, the document plus its annexes presents the following aspects for each layer:

- the functionalities,
- the prototype architecture and how it fits into the general ICOS architecture,
- the technical description and specifications,
- delivery (package, installation, download) and licencing information, and
- current limitations and future work to be done for the next releases (*ICOS Final release*).

In this iteration, the work done in T4.4 is directly included in the functionalities of each ICOS component, presented in individual sections below.

1.2 Relation to other project work

The second implementation deliverable of WP4, "*D4.2 ICOS components Data Management, Intelligence and Security Layers*", the software and document relate to:

- Tasks T2.2 (*Compute continuum requirements definition*) and T2.3 (*AI, data management and trust/security requirements*) on the analysis of ICOS-related technologies, the definition of project use cases, and gathering of system requirements for the ICOS components.
- Task T2.4 (Architectural Design) and deliverable D2.4 – "*ICOS architectural design (IT-2)*" [6] that defines the updated ICOS system architecture, describing how its layers, including Data Management, Intelligence, and Security, function, what their properties are, and how they interact.
- Deliverable "*D2.3 - ICOS ecosystem: Technologies, requirements, and state of the art (IT-2)*" [1] summarising work conducted in the second iteration of ICOS in the task T2.1 – (*Ecosystem identification: Baseline technologies*) regarding use case descriptions, requirements gathering, and analysis of the state-of-the-art baseline technologies, which were also used in IT-1.

The work reported in the current document starts from the base set in the second integrated ICOS Beta release, delivered in M22, and covers works performed for posterior ICOS releases until the Final ICOS release (M32).

1.3 Structure of the document

This document follows the structure presented in D4.1. Thus, it is structured in seven major sections:

- **Section 1 – Introduction** covers the objectives of the document, introduces its content, structure, relationships with other project's documents and the glossary,
- **Section 2 – Data Management** provides the functional and technical description for all the components and new integrations reached in IT-2. This section also discusses constraints and future work for the final release of the Data Management Layer,
- **Section 3 – Intelligence Layer** provides a functional and technical description of all its modules, delivery (package, installation, download), and licensing information, as well as any limitations and future work to be done concerning integration activities for the final release of the Intelligence Layer.
- **Section 4 – Security Layer** provides the functional and technical description, delivery (package, installation, download), licensing information and as well as project constraints and future integration work for the Security Layer modules towards the final release,
- **Section 5 – Conclusion** summarises the content of the document and provides considerations on how the technical work will finalise in the project.
- **Section 6 – References:** lists references used.
- **Section 7 – Annexes:** provides various annexes with more detailed explanations of the main technical sections of the document.

1.4 Glossary adopted in this document

The following table provides definitions of ICOS concepts and artefacts (as identified in D2.3 [1] and D2.4 [6]) that are discussed in the following sections of this deliverable.

Table 1: ICOS concepts and artefacts

Cloud Continuum
An aggregation of heterogenous resources (CPU, memory, storage, networks, IoT devices, intelligence) managed seamlessly end-to-end that may span across different administrative/technology domains in multi-operator and multi-tenant settings.
ICOS Agent
The ICOS software component that needs to be installed on each node with actual underlying orchestrator (OCM or Nuvla) that manages the infrastructure where the jobs are executed. ICOS Agents are distributed throughout the continuum.
ICOS Controller
The ICOS Controller is responsible for managing the ICOS continuum, which consists of a set of agents based on proximity criteria, by providing control, intelligence decision-making and lifecycle management.
ICOS Instance
A subset of the CC participating in the execution of a multi-component Application of a certain topology, resource requirements and constraints.
ICOS Node
Any resource, physical or virtual, that is running either the ICOS software (can be either an ICOS Controller or ICOS Agent).

ICOS Shell
A client distribution that is used to access the ICOS. The ICOS Shell runs externally to the ICOS Instance.
ICOS User
The ICOS user is the actor that uses ICOS to deploy his/her applications over suitable infrastructure, exploiting the ICOS management and optimisation capabilities to deploy applications fuelling novel business opportunities.
ICOS Application
User application deployed on an ICOS instance.
ICOS testbed
Testing environment provided by NCSRD for ICOS.

2 Data Management

The Data Management component is responsible for managing and enabling access to data from the rest of ICOS layers, in a way that optimises the performance of the components and leverages the continuum characteristics of the architecture.

It is a transversal component in the ICOS architecture, aimed at supporting the various data needs in the rest of the layers, making the distribution and heterogeneity of the infrastructure transparent to them, and providing efficient data access at the same time.

In previous deliverable D4.1 [7] the foundation of the Data Management was established and the basic requirements for IT-1 were detailed. This deliverable, D4.2, aims to build upon that previous deliverable and provide further detail on the changes and improvements that have been included in the software architecture for IT-2. This document will also provide a more in-depth view on different integrations with other ICOS layers.

This section is structured as follows: the functional description of the Data Management is provided in subsection 2.1. Subsection 2.2 describes the technical details of the solution, and the different integrations present in IT-2. A brief overview and release notes on the new software releases for Zenoh and dataClay (the software blocks that shape the Data Management) can be found in 2.3. We finalise the Data Management section on this document discussing some limitations and future work in 2.4.

Table 4 in Annex I.A shows the different requirements of the ICOS project in which there is an involvement from the Data Management layer. Most of them were already presented in D4.1 and are related to the IT-1 efforts of the project, but the IT-2 brings attention to several new key requirements.

2.1 Functional description

At a very high level, the Data Management Layer will be addressing three distinct needs regarding data:

- Transport: meaning all the data messages and brokering, as well as handling data in motion.
- Mutable data structures: data that requires concurrent access and modifications.
- Compute: Very data-intensive computation that benefits from having data locality.

These basic needs, although overlap a little bit, categorise quite well the pillars that the Data Management is providing to the ICOS project.

The two software blocks that ICOS will use for fulfilling the Data Management Layer are the following:

- Zenoh[20] – a pub/sub/query protocol,
- dataClay[21] – a distributed active object store.

2.1.1 Fitting into the ICOS architecture

The general ICOS architecture for IT-2 was described in D2.4 and briefly depicted the different responsibilities and placement of the Data Management Layer software components.

We can find the following software components in the ICOS architecture:

- Data Bus for ICOS-Application communication: This communication backbone enables transversal communication on ICOS applications. This communication will happen while the application is agnostic to the exact topology and its changes. An example of the usage of this communication can be seen in the Topology Exporter functionality of ICOS. The data bus feature is achieved through the deployment of Zenoh components in the agents.
- Training offloading on the Intelligence Layer: A software component from the Data Management Layer is used by the Intelligence Layer in order to offload training operations. This allows an efficient and distributed usage of continuum resources (i.e. avoids roundtrip to far clouds when there are available computing resources closer). This is achieved through the deployment of dataClay services distributed in the continuum.

- OTLP Bridge on the Telemetry flow: To improve responsiveness and resource utilisation, a Data Management component is included among Telemetry components and is able to process data coming from Telemetry Agents. This software component, a dataClay service deployed on the agent next to the Telemetry Gateway, is able to take advantage of the logs that are being sent through standard protocols and provide them to other ICOS components that need that data. This can be done in a fast and efficient manner by taking advantage of data locality and minimising data copies.
- Policy Storage backend: The Dynamic Policy Manager can take advantage of dataClay features and use dataClay as the storage backend for its mutable persistent data. This simplifies the deployment of the Dynamic Policy Manager and provides a storage backend that is able to take advantage of the distributed nature of ICOS infrastructure.

The following section (2.2) contains more in-depth technical details on these different software components and their interaction.

2.2 Technical description

Previous deliverable D4.1 included extensive information on the software solutions used for implementing the Data Management layer. Most information in such document is still applicable at the time of writing D4.2. For some further details covering the releases and new features of these software, see Section 2.3.

The present deliverable focuses on IT-2 and thus the following subsections will elaborate on the technical aspects related to the new integrations involving Data Management.

2.2.1 Data bus

Eclipse Zenoh is a communication middleware designed to provide a set of unified abstractions to deal with data-in-motion, data-at-rest and computations at internet scale. Zenoh is composed of two main parts, firstly a decentralised and distributed networking protocol, and secondly, the Zenoh APIs to manage it.

Eclipse Zenoh APIs are available for the most popular programming languages (i.e. Rust, Python, C, C++, REST) the core of the networking routing and messaging mechanisms are written in Rust. The Zenoh v1.2.0 Rust API includes a publish, a subscribe and query examples, it can be found here: https://docs.rs/zenoh/latest/zenoh/.

To enable the direct communication between the ICOS controller and the ICOS agent nodes. Three zenoh-routers have been deployed in the stagging testbed [1]in the NCSR"D" infrastructure with the purpose to connect the ICOS controller with two cluster nodes, as illustrated in Figure 1.

The config file (i.e., `values.yaml`) included in the Zenoh Helm Chart describes the connecting end-points and the image/version of the feature that should be used, in this case it is `eclipse/zenoh tag:1.0.0`, this image is taken from the Docker Hub container repository. See the configuration in the source-code extract provided at Annex I.B.

[1] https://production.eng.it/gitlab/icos/data-management/zenoh-helm-chart

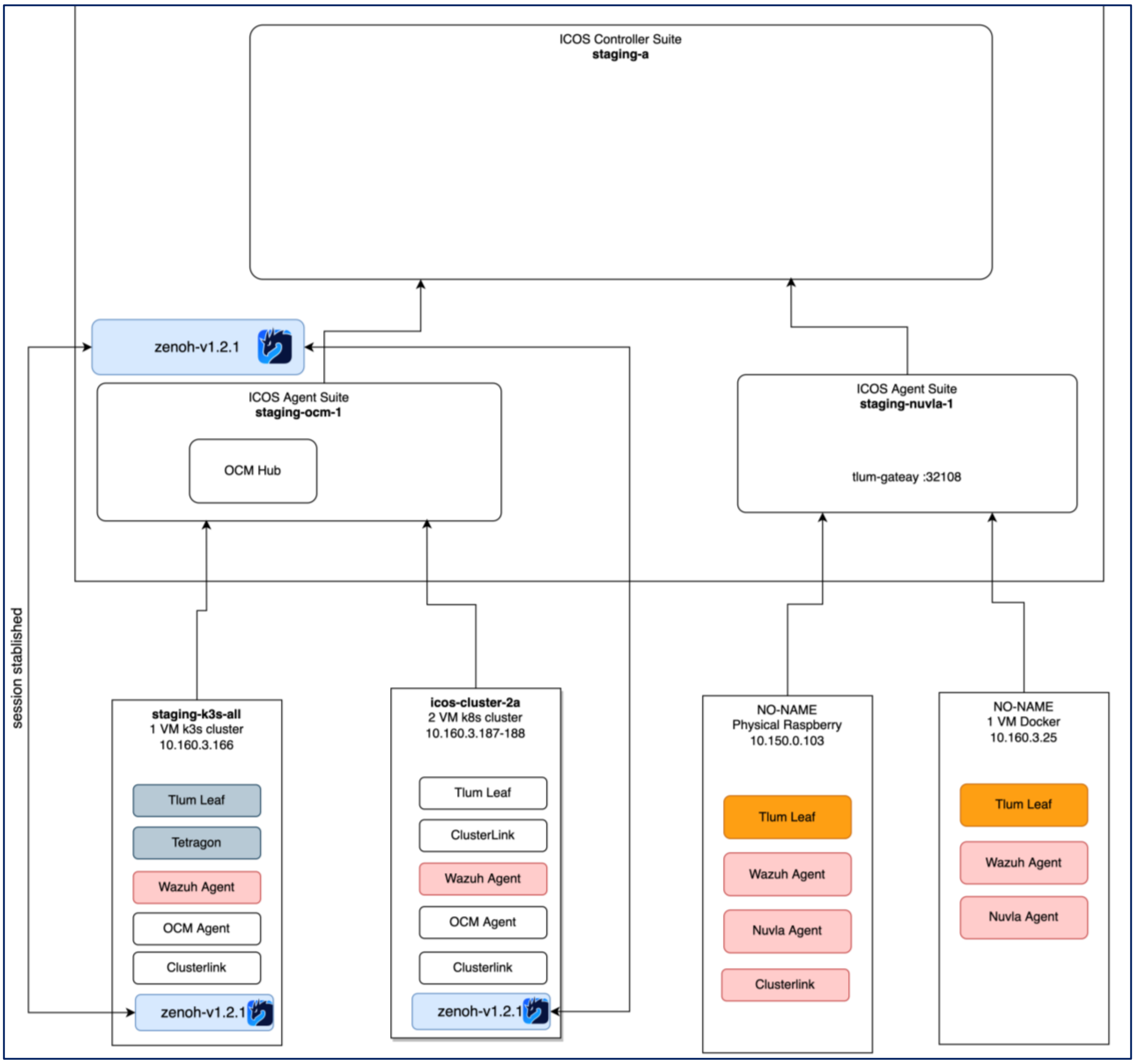


Figure 1. Eclipse Zenoh deployment in the stagging testbed

This Data bus is part of the Topology Exporter, and it is a key aspect to satisfy requirement CC_FR_03 (see Annex I.A).

2.2.2 Data management in the AI training workflow

The data management component plays a crucial role in a pivotal aspect of the ICOS Intelligence Layer, its AI training workflow, a task that requires both a high amount of data and a high number of computational resources. This process is further explained, from the point of view of the Intelligence Layer, in Section 3.5 (*Data processing*).

Figure 2 shows the sequence diagram of this training offloading procedure. In this figure we can see how the dataClay server is responsible for performing the model training operation, i.e. computation. The algorithms and data are provided by the Intelligence Layer, but dataClay offers a hardware and location agnostic interface for running this process.

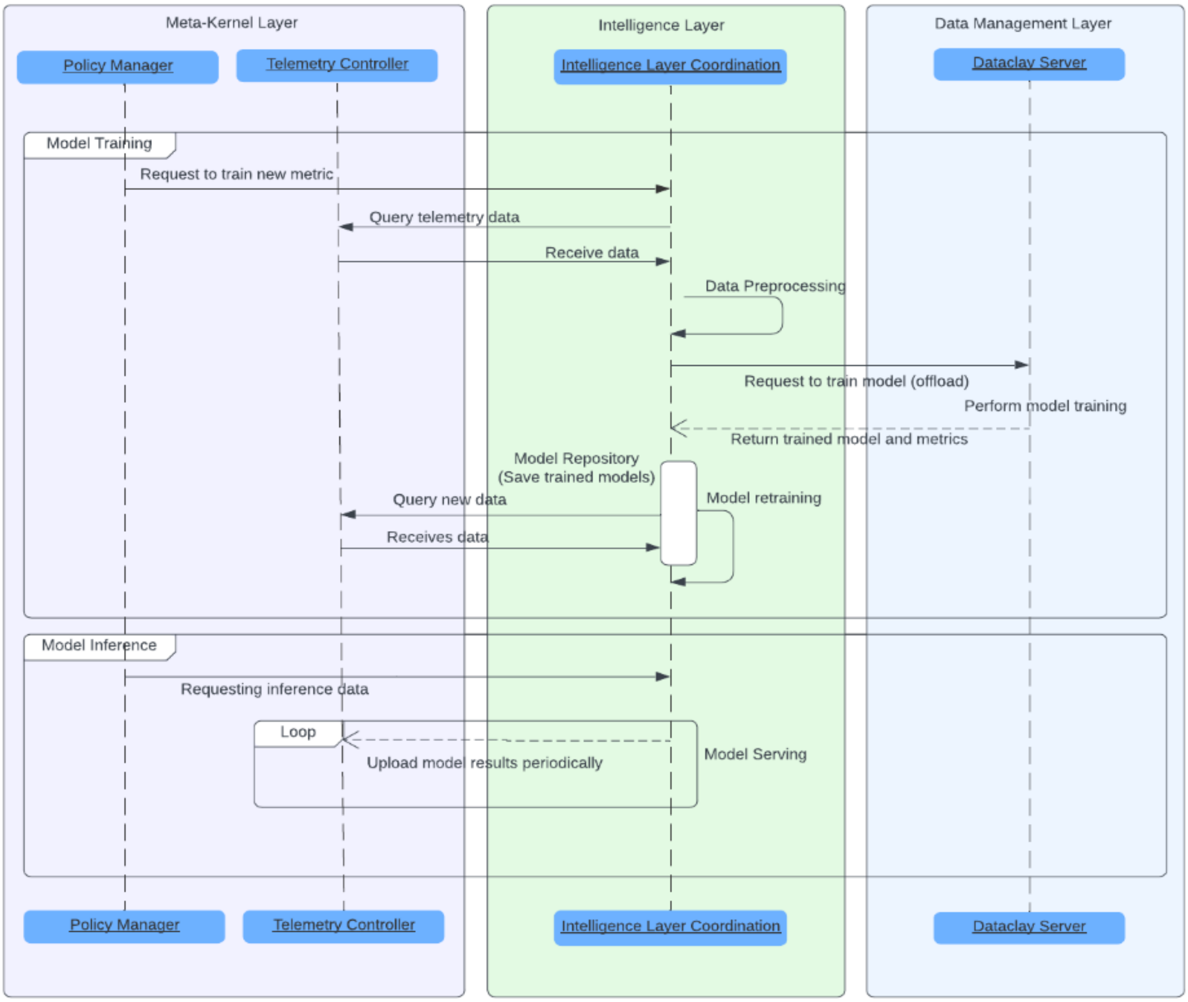


Figure 2. Sequence diagram for AI training offloading (from D2.4)

This design allows deployment of the dataClay backend in environments where there are enough computational resources; this is transparent for the Intelligence Layer, which can use dataClay through its object-oriented interface, without having to account for the distribution or location of data and computational resources.

This feature is responsible for satisfying requirements CM_FR_02 and CM_FR_15.

2.2.3 dataClay OTLP bridge

As already explained in a previous deliverable (D3.2 [4]), the Logging and Telemetry architecture contains a Telemetry Gateway in the ICOS Agent which aggregates logging from the different ICOS Nodes (Edge Node or Cluster). This is shown in Figure 3.

The communication between Telemetry Agent and Telemetry Gateway uses the OpenTelemetry protocols and standards; a new component within the Data Management, named dataClay OTLP Bridge, is included in IT-2 and deployed in the ICOS Agent. This component is able to aggregate live telemetry coming from the different Telemetry Agents and provide them for real-time processing of this data, transparent to the details of the topology, allowing fast and local access to this data.

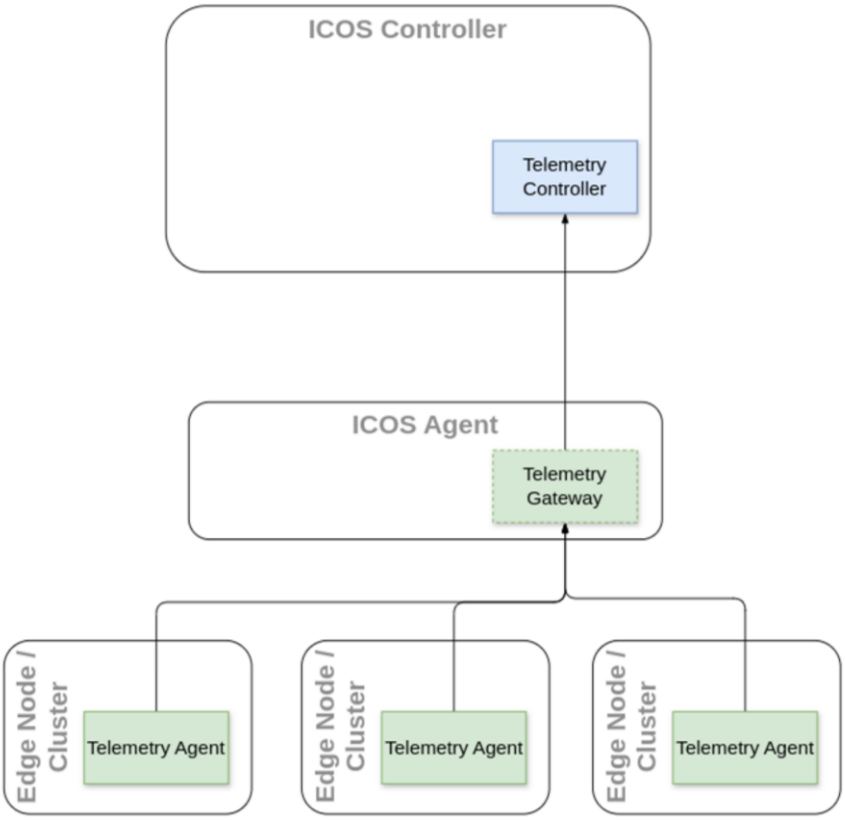


Figure 3. Distribution of the Telemetry components across the Continuum nodes (from D3.2)

This design takes advantage of the continuum and the resources available in the ICOS infrastructure. A current application of the dataClay OTLP Bridge is the privacy-aware model training process (Federated Learning) developed as part of the Intelligence Layer Trustworthy AI component. This is detailed in Subsection 3.3.2.4 (*Federated Learning*). Thanks to this component, the Federated Learning solution avoids a roundtrip to the Controller, increasing data locality and resulting in a more efficient resource utilisation.

2.2.4 Policy storage backend

In D3.2 [4], different integrations related to the Policy Manager were discussed, and one of those was the Policy Storage backend using dataClay.

This storage backend takes advantage of the object-oriented interface offered by dataClay and allows to programmatically use dataClay persistent and mutable objects from the Dynamic Policy Manager. The Dynamic Policy Manager offers a pluggable storage mechanism. In the previous release, the storage of the information policies used MongoDB as a non-relational database through an ORM Python library.

The current release of the Dynamic Policy Manager supports dataClay as the Policy Storage backend, using it through the native object-oriented interface of dataClay. This results in a versatile backend that is able to store data in the continuum and overall results in a tight integration of ICOS components.

2.3 Software releases

The following subsections present the latest releases for Zenoh and dataClay (the software used to fulfil the Data Management layer). These software releases have happened during the IT-2 development cycle. Delivery and usage of this Software remain as described in D4.1[7].

2.3.1 Eclipse Zenoh 1.2.0[2]

On October 21, 2024, Eclipse Zenoh v1.0.0 was released, and this marked a milestone in the development of the data management component as the open-source project moved out from incubation status to a full-project status, which implies a higher quality and stability in the source-code level, offering backward and forward compatibility for the coming versions for the Zenoh API.

Moving forward three months, six intermediate releases has been done, at the time of writing this deliverable, the latest release of Eclipse Zenoh is v1.2.0. This newest version brings new features and improvements while keeping focus on the values of the ICOS project. This update reflects priorities such as adaptability, security, and performance optimisation, ensuring an efficient and interconnected ecosystem for devices, applications, and data.

- **Openness and Interoperability:** Zenoh 1.2.0 embraces openness, enabling diverse technologies to collaborate. For example, the new querier API supports efficient and optimised data retrieval. Enhanced support for ROS2, a widely used framework in robotics, strengthens connections between Zenoh and other platforms, facilitating interoperability. There is an ongoing effort to integrate natively Zenoh, as Tier-1[3] communication middleware in the coming ROS2 release in May 2025; this is achieved in **rmw_zenoh**[4].
- **Adaptability and Scalability:** Zenoh 1.2.0 introduces features that adapt to varying technological needs. The stabilisation of liveliness API support ensures real-time monitoring of active participants in the network. Zenoh-Pico[5], which is made for small IoT devices, has also been updated to work with the latest version. With huge improvements in performance and extension in scope as it is now compatible with Raspberry Pi Pico series[6].
- **Security and Privacy:** The protocol enhancements address critical issues such as fragmentation and message integrity, ensuring secure and reliable data transmission. These updates safeguard interactions across the network, whether between IoT devices or in the cloud.
- **Performance Optimisation:** In addition to the querier, the new advanced publisher/subscriber mechanisms also improve data throughput and reliability. These optimisations enhance system performance while minimising resource consumption, key to efficient IoT-to-cloud integration.
- **Technology Agnosticism:** Zenoh 1.2.0 is committed to supporting diverse hardware and software platforms, including new features for QNX operating systems, highlights its technology-agnostic approach. This openness reduces dependence on specific vendors and encourages widespread adoption.
- **Future Market Enablement:** Zenoh continues to foster innovation by bridging gaps in IoT-to-cloud operations. The new tool ZettaC2 (for Zetta Control Center) is now available on all operating systems. It has been made for monitoring and managing Zenoh systems. By simplifying complex interactions, it paves the way for emerging edge markets and collaborative data-sharing environments. Zenoh 1.2.0 demonstrates how foundational values like openness, adaptability, security, and performance guide advancements in IoT-to-cloud ecosystems, delivering a system designed for the future.

[2] https://github.com/eclipse-zenoh/zenoh/releases/tag/1.2.0

[3] https://github.com/ros2/rmw_zenoh/issues/265

[4] https://github.com/ros2/rmw_zenoh

[5] https://github.com/eclipse-zenoh/zenoh-pico

[6] https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html

2.3.2 dataClay 4.1

In November 2024, dataClay v4.1 was released[7]. This new release includes several features and improvements that streamline deployment and development of Continuum applications and shape dataClay software beyond its HPC inception and more into a full-fledged continuum-ready storage system solution.

The following dataClay features are key for a successful Data Management within the ICOS project:

- **Async**: migrating to an async implementation (instead of threads) results in a lighter service, with higher concurrent capabilities and a more efficient use of processing resources. This is crucial for constrained devices, a common occurrence in computing continuum environments. This new release includes several fixes and improvements related to the internal async implementation.
- **Proxy**: a new microservice within the dataClay stack, which simplifies the deployment of dataClay services in restricted networks or NAT-ted environments. Given the distributed nature of the computing continuum, where full connectivity between devices is not guaranteed, this new microservice improves the flexibility of dataClay deployments. This new release fixes several bugs that challenged its usage within ICOS architecture.
- **JWT token support**: the previous proxy feature also includes support for processing and validating JWT tokens [34]; this mechanism is compatible with the ICOS security architecture and thus solves the challenge of both authentication and authorisation across distributed environments. This new release includes more examples and internal mechanisms to accommodate more kinds of tokens.

2.4 Limitations and future work

The deployment of Eclipse Zenoh v1.0.0 was performed manually, following the Open Call infrastructure On-Boarding v1.3 document [9] provided by the testbed coordinator together with the WP5 leader. At the time of writing this deliverable, the software engineer needs to request the authorised authentication to the staging test bed and get the right access level to the infrastructure. This will be streamline in the security characteristics on the ICOS platform.

The deployment of dataClay services also needs manual intervention given that there is no automatic discovery of storage/computational resources. Placement of dataClay services will impact the task offloading mechanisms used by the Intelligence Layer and thus the user performing the initial deployment process must take that into account and take the appropriate decisions.

Communication between Data Management components is partially automatic. Our goal is to provide the necessary automations in order to establish connectivity between the services needing it (e.g. the different Zenoh routers, or the dataClay backend services). At the time of submitting this document (M30), the workaround includes some manual steps as part of the deployment process.

[7] https://dataclay.readthedocs.io/en/4.1.0/releasenotes/4-x.html#november-2024

3 Intelligence Layer

This section presents the progress regarding development and implementation of the Intelligence Layer modules and components from IT-1 as presented in D4.1 [7] to IT-2 as presented in this deliverable.

In IT-2, multiple enhancements have been introduced in each module, for instance:

- AI Analytics:
  - AI multivariate and multioutput models can now be trained in the Intelligence Layer using multiple metrics, for instance, to predict CPU and memory utilisation at once.
  - AI model compression methods have been integrated within AI Analytics to improve AI model efficiency using *quantization* and *knowledge distillation* [16]. This can be enabled during model training.
  - The AIOps framework MLFlow has been included within the Intelligence Layer for experiment tracking and visualise model evaluation results.
    - Model meta-information has been integrated into the MLFlow registry for the same purpose.
- Trustworthy AI:
  - support for privacy-aware model training (federated learning) has been added, which also intends to reduce data movement.
  - model explainability has been integrated into MLFlow to enhance model transparency with SHAP [30], a framework initially presented in D4.1 [7]
  - model learning curves (training and validation) added in the MLFlow framework to understand AI model's suitability.
  - confidence scores and confidence intervals added into each model prediction at the Intelligence coordination module backend (Intelligence API presented in D4.1 [7] to aid in decision-making processes rating how confident the model is per prediction output.
- AI Coordination:
  - Frontend API created to interact with the MetaOS. This is reachable from the ICOS Shell using Keycloak [40] for Identity and Access Management; it queries Telemetry to gather ICOS node metrics and interacts with the backend API to request model training and inferencing.
- Data processing:
  - Enhancements integrating with dataClay [21] for data access and AI model training offloading.

This section also introduces the AI Models and Data Repository, an online repository that acts as the final Intelligence Layer module, to be delivered in D4.3 – "*ICOS Dataset and AI models marketplace*".

The section is structured as follows: we present the final (IT-2) design of the Intelligence Layer in Subsection 3.1, an in-depth update in IT-2 over the Intelligence coordination module in Subsection 3.2, the Trustworthy AI module in Subsection 3.3, the AI Analytics module is presented in Subsection 3.4, the Data Processing module in Subsection 3.5 and we introduce AI Model and Data Repository in Subsection 3.6.

Next, we document project constraints and Software limitations in Subsection 3.7. Finally, we document licensing and instructions to download and set up the Software in Annex II.I.

3.1 Fitting into the ICOS architecture

This subsection presents the final (IT-2) design of Intelligence Layer modules, components and functionalities.

The ICOS Intelligence Layer is located in the ICOS Controller as stated in D2.4 [6]. The Intelligence Layer contains the following modules:

1. The Intelligence Layer Coordination module, which is made of two components:

- **Intelligence coordination frontend**: this component has the Export metric API that communicates with the ICOS Shell, requests model training and inference to the *Intelligence coordination backend* covered below, reading data for inference and training, and storing model results into Telemetry. It also verifies the user's permissions and authentication against Keycloak.
- **Intelligence coordination backend**: this component receives requests from the frontend for training and inference, returning inference results; it stores models in its model registry, and communicates with all other Intelligence Layer modules and APIs (e.g. LOMOS API "2" described in Section 4.5).

2. The AI analytics module, which handles model training, inferencing, and integrates model compression and analysis.
3. The Data Processing module, which provides data pipelines as introduced in D4.1 [7] and communicates with Data Management to offload model training (see Section 2.2.2) and access data available across the ICOS infrastructure.
4. The Trustworthy AI module, integrated within the Intelligence Layer to enhance confidence in the models by leveraging model monitoring and explainability. Part of this module is the privacy-aware training capability (federated learning), which integrates with the AI analytics module for model training and the Data Processing module to initiate the federated learning process across the ICOS infrastructure.

Outside of the ICOS controller, we find:

- the AI Models and Data Repository, which is an online repository of AI models trained in ICOS. It also contains datasets used to train such models. The *Intelligence coordination* backend will offer a client to communicate to this repository in D4.3 – "*ICOS Dataset and AI Models Marketplace*", allowing the Intelligence Layer to import pre-trained models.

All of these are illustrated in Figure 4.

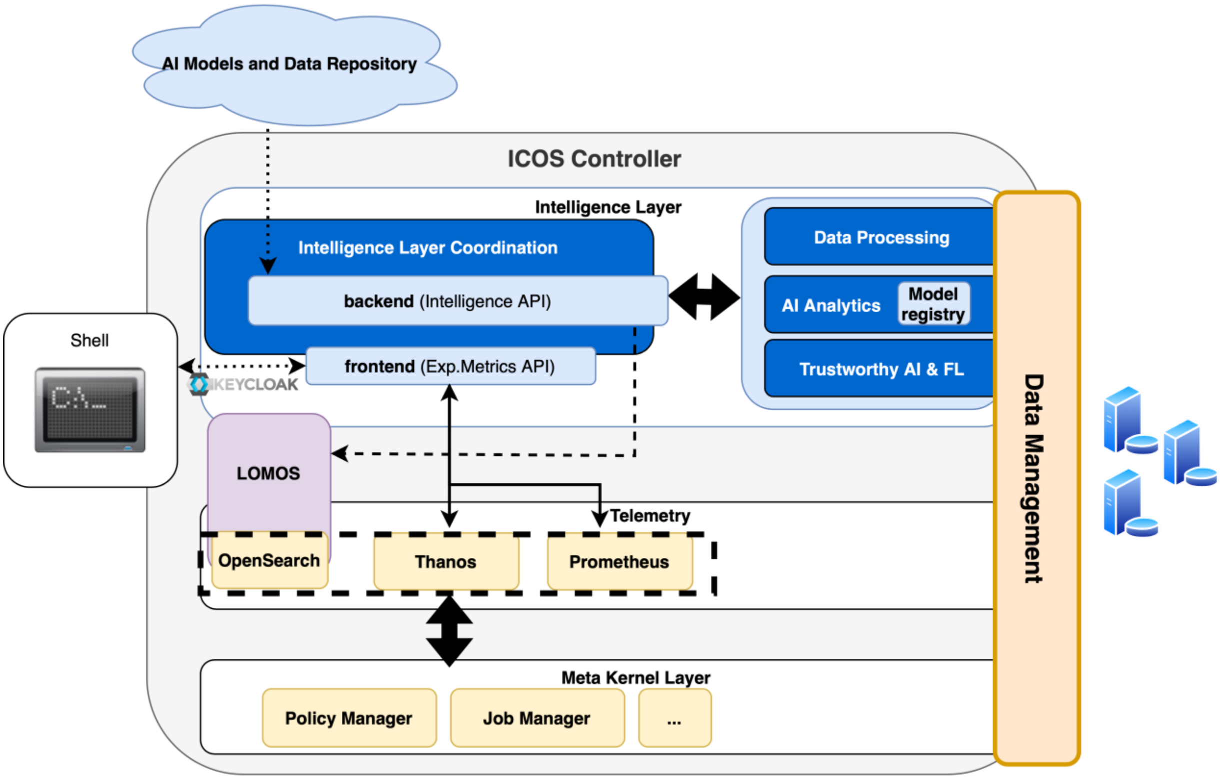


Figure 4. Intelligence Layer modules in the ICOS architecture – enhanced from D2.4

Additionally, this project will provide containers for *ICOS users* to run AI workloads leveraging the same library versions as Intelligence at the controller, empowering them to train models and contribute to the *AI Models and Data Repository*, enhancing this way progressively the list of models available online. Such containers receive the name of **AI support containers.** This also will enable use cases to easily integrate with containers running data management libraries like *dataClay*, which is usable for AI offloading across the ICOS infrastructure, as introduced in Section 2 of this document.

AI support containers, presented in Figure 5, were initially introduced in D2.4 to run AI in ICOS agents (see Figures 9 and 10 in Section 3.2.2 of D2.4 [6]). After task force discussions concerning the MetaOS architecture and overall security, MetaOS-integrated AI has been left solely for the ICOS controller. Use cases will still be able to use AI support containers managed by users in their own infrastructure (edge devices).

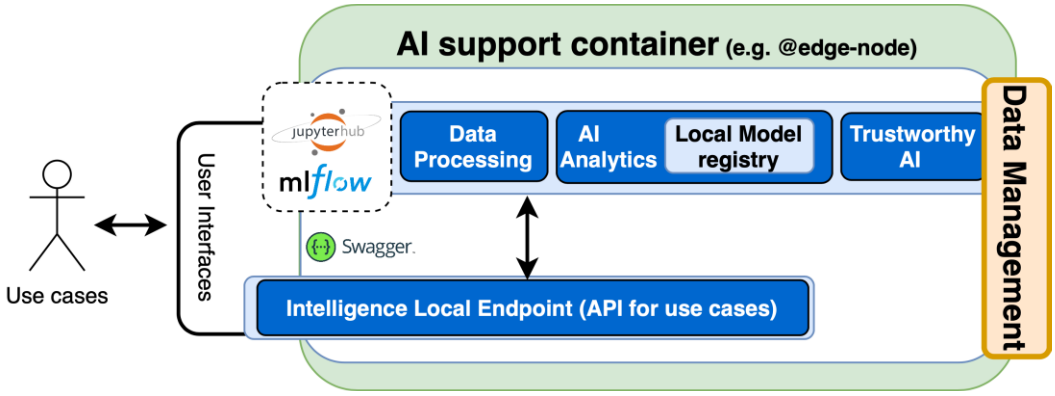


Figure 5. Modules and functionalities in AI support containers

Regarding AI training (and federated learning) offloading, the design stays as presented in Figure 2 of this deliverable and in D2.4. The main difference concerning learning tasks is that the ICOS agent no longer plays a role in the process; the learning process is offloaded instead to a node in the ICOS infrastructure running a dataClay container, as specified in Section 2, with the relevant AI libraries (see Figure 6).

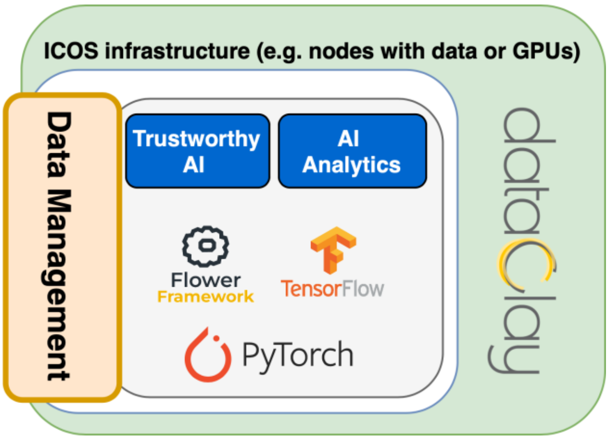


Figure 6. Modules and libraries in the dataClay containers for Intelligence offloading

As part of the ICOS Final release (D5.3 [8]), instructions on how to run **AI support** and **dataClay** containers to offload AI will be provided.

3.2 Intelligence Layer Coordination module

Initially introduced in D4.1, the Intelligence Layer coordination module allows AIOps across the cloud continuum. It provides mechanisms for sharing, updating, and deploying models. It interacts with a model registry that stores essential metadata such as model descriptions, algorithms, and performance metrics obtained during model evaluation.

- D4.1 established a framework for managing and training basic ML models. The initial focus was to provide an API and model registry able to serve models provided as well as offload training using the Data Management component.
- D4.2 enhances AI pipelines by adding a new frontend API that ensures seamless communication with the Meta Kernel Layer. Furthermore, the internal backend (Intelligence API) now integrates with new functionality from other Intelligence Layer modules (e.g., enhanced pipelines, model monitoring, compression, and explainability, new algorithms, and MLFlow integration).

3.2.1 Functional description

The main addition to this module in IT-2 is an Intelligence Layer coordination frontend API. This is visible in Figure 4 as the main access point from the ICOS Shell and bidirectional communication point with Telemetry.

This frontend API, due to its nature of reading and exporting AI metrics predictions from and to Telemetry, receives the name of **Export metrics API**. This component of the Intelligence Layer coordination module acts as a middle layer between WP3 components and the Intelligence Layer.

- The export metrics API, upon request to generate predictions, queries the *Intelligence Layer (backend) API* for the requested model predictions and uploads it to the telemetry periodically for a specified time interval. Similarly, when a request is made to the export metrics API to train a model for a specific node, for instance, metrics forecasting, the data is collected from Thanos and read by the Intelligence Layer API to train that specific model and saved in the model repository ready to be served for model inferencing.

IT-3 will see the deployment of the intelligence coordination module alongside other WP3 modules. This integrated system will facilitate automated model training, inference, retraining, and monitoring, ensuring continuous improvement of model performance in the production environment.

At the time of writing this deliverable, Intelligence meets most of the requirements defined by WP2. Basic functionality was met for all requirements between IT 1 and IT 2.2 except for Federated learning-related requirements (CM_FR_16, SST_FR_04), where integration work is being performed towards IT-3. Table 5 in Annex II.A reviews the degree of accomplishment of each of these in each iteration.

As described in D2.4 and in Section 3.1, the Intelligence Layer coordination module is located at the ICOS controller and communicates with the ICOS Shell, Meta-Kernel Layer and Telemetry. The architecture of this module and its interaction with other components is also described in the D4.1 which showcases the Data management, Security Layer and Distributed Meta Kernel layers respectively.

3.2.2 Technical description

The Intelligence Layer coordination module is built on Python 3.10 and is packaged as Docker container for deployment at the ICOS controller which is already described in D4.1 [7]

In IT-1, *Data processing* and *AI Analytics* were the main modules supported in Intelligence Layer coordination. In IT-2, support for new functionalities in these modules has been added as well as for the Trustworthy AI module (this module is described in detail later in Subsection 3.3).

The Intelligence Layer coordination module is mainly composed of two components:

- AI coordination backend (Intelligence API): This serves as the foundational backend component for the Intelligence coordination module. Its responsibilities encompass the entire spectrum of model operations, including training, data processing, model compression, retraining with drift detection capabilities, and inference request processing. The updated list of endpoints for this API backend can be found in Annex II.B.

- AI coordination frontend (Export metrics API): This acts as a gateway to the Intelligence API, facilitating model training and inference requests originating from various ICOS modules. Additionally, it plays a crucial role in data preparation. By leveraging telemetry data sourced from Grafana, Prometheus, or Thanos and managed by the Data Management component, the Export Metrics API enables the Intelligence API to effectively preprocess time-series data for training both univariate and multivariate models.

Since the *Intelligence coordination backend* was already described in the *Intelligence coordination* sections of deliverables D4.1 and D2.4, we devote the rest of this section to explaining the *AI coordination frontend*, components that interact with it, and the nature of these interactions.

We will start linking this to D3.3 – “*Meta-Kernel Layer Module Integrated (IT-2)*” [5], on the integration performed between the ICOS Shell and the Intelligence Layer. This piece of integration, which is achieved through the *Export metrics API*, is introduced in the next subsection.

3.2.2.1 User interaction

To allow for interaction with the system by the user, the Export Metrics API is integrated with parts of the ICOS Shell. Currently, the *ICOS CLI* allows for interaction with the functionality of the intelligence functions through two commands: "*train metrics*" and "*predict metrics*". Both commands result in POST requests being sent to first the ICOS shell backend and then the Export metrics API as described in the next subsection.

- Each of these calls finally targets the respective endpoint of the Export Metrics API, namely `/create_model_metric` and `/train_model_metric`.
- The requests furthermore have the authentication token of the user attached (see Section 4.4), allowing for evaluation by the intelligence components.

For both commands, a payload in JSON format is sent, which defines all the needed information and context for the respective functionality, which is further described in the next subsections. The training or prediction process result is then communicated back to the user through the CLI, displaying the respective output. The ICOS shell hereby serves as a stateless middleware to translate the queries and provide an easy user interface. Some of these functionalities are also planned to be implemented into the ICOS GUI to allow for even better accessibility of these functionalities.

3.2.2.2 AI coordination frontend – Export Metrics API

As effective monitoring of metrics is vital for maintaining system reliability, the ICOS Export Metrics API project ensures comprehensive metric tracking and analysis. Leveraging the `prometheus_client` library, the project ensures effective monitoring of system metrics and predictive insights through seamless integration with Prometheus [24][, Thanos, Grafana, and Intelligence coordination backend. All the components and libraries part of this process are covered in Annex II.D.

The export metrics API offers functionality for real-time metric tracking and predictive analysis, addressing the following four objectives:

- **1) Metric creation**: This feature supports the creation and management of a wide range of metric types, such as Counter, Gauge, Info, and Enum. These types cater to diverse monitoring needs. For instance, Counters are useful for tracking cumulative values like the number of requests processed, while Gauges allow tracking values that can fluctuate, such as memory, CPU and energy consumption usage. Additionally, Info metrics provide metadata-like information, and Enum metrics are particularly effective in representing finite states. The API also enables dynamic updates to these metrics, making it easier for system operators to adapt monitoring configurations to changing requirements.

  The Export Metrics API supports both **dynamic** and **static** *metric creation.*

- Dynamic metric creation occurs when a request is made via the `/create_model_metric` endpoint, generating metrics on demand based on telemetry data and machine learning models. This dynamic process is targeted to applications running in ICOS, whose metrics to be forecasted may differ from one to another.
- Static metric creation operates in the background, periodically querying the *ICOS Aggregator* data in Telemetry [4] to detect newly added nodes in the ICOS infrastructure. When a new node is identified, the API automatically provisions default metrics for CPU, Memory, and Energy consumption. Conversely, if a node is no longer present in the ICOS environment, the corresponding metrics are automatically unregistered, ensuring resource efficiency and accurate monitoring. This static process is targeted to core metrics of the ICOS Meta-OS, such as CPU utilisation and energy consumption. MetaOS processes, such as the matchmaking module, use future predicted values of these metrics to be AI-driven.

▸ **2) Predictive metrics**: Telemetry-based metric creation is a critical feature that combines real-time data from sources such as Grafana or Prometheus and Thanos with AI models hosted at the Intelligence Layer. This enables the creation of predictive metrics, where historical data trends and current states are analysed to predict system behaviour. For example, this could include forecasting workload surges (on CPU, memory, etc) or identifying potential anomalies before they impact operations. The integration with predictive models ensures that monitoring extends beyond reactive systems to become a proactive solution.

- Predictive metrics are created using the `/create_model_metric` endpoint, which simplifies the interaction between telemetry data and the Intelligence Layer (Figure 7).

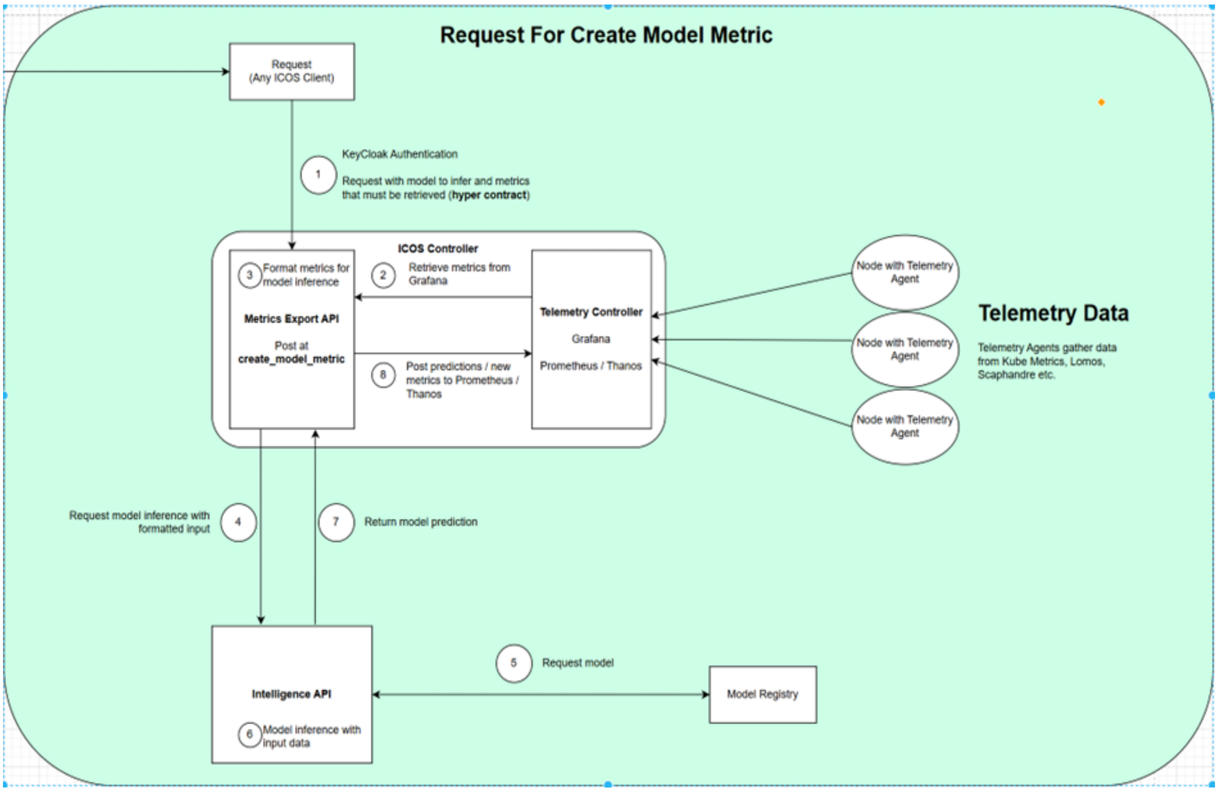

Request For Create Model Metric
Request (Any ICOS Client)
1
KeyCloak Authentication
Request with model to infer and metrics that must be retrieved (hyper contract)
ICOS Controller
3
Format metrics for model inference
Metrics Export API
Post at create_model_metric
2
Retrieve metrics from Grafana
8
Post predictions / new metrics to Prometheus / Thanos
Telemetry Controller
Grafana
Prometheus / Thanos
Node with Telemetry Agent
Node with Telemetry Agent
Node with Telemetry Agent
Telemetry Data
Telemetry Agents gather data from Kube Metrics, Lomos, Scaphandre etc.
4
Request model inference with formatted input
7
Return model prediction
Intelligence API
6
Model inference with input data
5
Request model
Model Registry

Figure 7. Metric creation via model inference

▸ **3) Model training**: the frontend API also facilitates model training workflows to the MetaOS, by preparing datasets using Telemetry data sourced from Grafana or Prometheus/Thanos.

- These datasets are then posted to dataClay, where they are used to train AI models via the Intelligence coordination backend.
- The trained models are subsequently then used again at *metric creation*, enabling the generation of new *predictive metrics*. This feature ensures that the system evolves alongside the monitored environment, continuously learning and improving its predictions.

- The training process is initiated via the `/train_model_metric` endpoint from the AI coordination backend, streamlining the workflow from data preparation to model integration (Figure 8).

- **4) Lifecycle management:** To ensure efficient resource utilisation, this frontend API includes functionalities for stopping and unregistering metrics that are no longer required.
  - The `/stop_model_metrics` endpoint can halt active metric generation, freeing up resources for other tasks.
  - Similarly, the `/unregister_metric` endpoint allows the removal of outdated or obsolete metrics, keeping the monitoring setup clean and efficient.

- This lifecycle management capability ensures that the monitoring system remains lean and adaptable over time.

A more in-depth view of the workflows followed by this component, including authentication workflows in Keycloak, integration with Telemetry and the Intelligence Layer, as well as API workflows, are covered in Annex II.C.

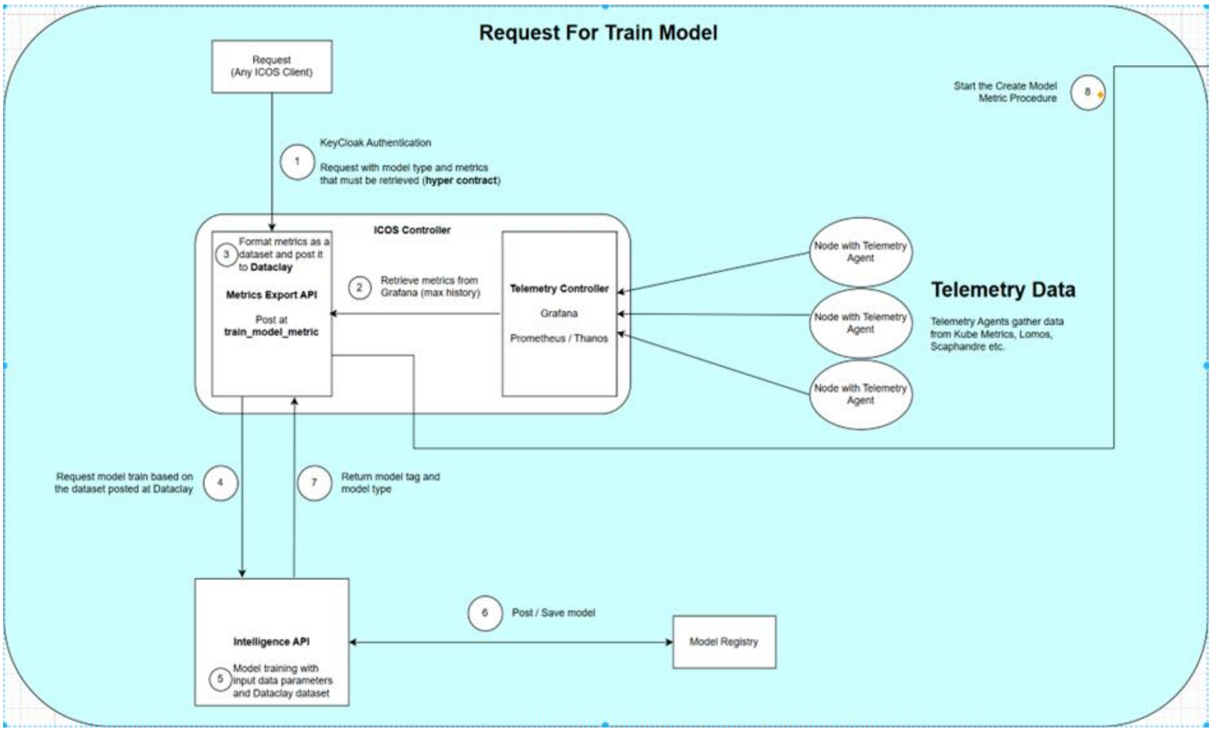

Request For Train Model
Request
(Any ICOS Client)
1
KeyCloak Authentication
Request with model type and metrics that must be retrieved (hyper contract)
ICOS Controller
3
Format metrics as a dataset and post it to Dataclay
Metrics Export API
Post at train_model_metric
2
Retrieve metrics from Grafana (max history)
Telemetry Controller
Grafana
Prometheus / Thanos
Node with Telemetry Agent
Node with Telemetry Agent
Node with Telemetry Agent
Telemetry Data
Telemetry Agents gather data from Kube Metrics, Lomos, Scaphandre etc.
8
Start the Create Model Metric Procedure
4
Request model train based on the dataset posted at Dataclay
7
Return model tag and model type
Intelligence API
5
Model training with input data parameters and Dataclay dataset
6
Post / Save model
Model Registry

Figure 8. Metric creation with model training

3.2.2.2.1 Technical specification

The ICOS Metrics Export API uses the Python *prometheus_client* library, which provides a powerful framework for metric generation and management. The API utilises RESTful endpoints, offering a modular and scalable architecture that supports diverse deployment options.

- The design of the Intelligence frontend is explained in more detail below. The architecture comprises two major workflows:
  - Metric creation via model **inference** (Figure 7): This focuses on using pre-trained models to generate new metrics based on telemetry data. It involves fetching data from Grafana or Prometheus/Thanos, formatting it for model inference, and posting predictions to Prometheus/Thanos for monitoring.
  - Metric creation with model **training** (Figure 8): This adds a model training step, where *Telemetry* data is used to train new models, which are then deployed to generate metrics. Together, these workflows provide a comprehensive solution for both reactive and predictive monitoring needs.
- These workflows ensure seamless integration of telemetry data, predictive modelling, and metric generation.

The Intelligence coordination frontend component empowers ICOS with robust metric monitoring, predictive analytics, and seamless integration into its operational ecosystem

- Deployment alongside ICOS Controller.
- Supports Docker and Helm for scalability.
- Requires environment variables for Grafana, Prometheus or Thanos and Intelligence Layer API integration.

- Key Features:
  - Swagger UI
  - Keycloak-based authentication for security.

3.3 Trustworthy AI module

The Trustworthy AI module in ICOS is dedicated to ensuring AI's ethical, responsible, and privacy-preserving use. Its mission is to uphold the principles of transparency and fairness, as outlined in D4.1, which emphasises the development of tools to create models that deliver reliable results, protect user data, and promote ethical standards.

The following subsections delve into the core aspects of this module, highlighting its key components:

- AI explainability through SHAP for interpretability and transparency,
- mechanisms to ensure model confidence and robustness,
- model and data monitoring with drift detection, and
- federated learning to safeguard privacy.

Together, these functionalities aid the Trustworthy AI module on aligning to ICOS's overarching goals promoting the responsible adoption of AI.

3.3.1 Functional description

As outlined in deliverable D4.1, the Trustworthy AI module is a key component of the Intelligence Layer Coordination API. See D4.1 [7] for a high-level functional description of this module.

3.3.2 Technical description

In IT-2, the intelligence Layer has introduced several features to improve AI trustworthiness. This section is devoted these new features, which are outlined below:

- *AI explainability* – This is achieved through the library SHAP (SHapley Additive exPlanations), which provides detailed insights into the factors influencing model decisions, enhancing transparency in AI-driven processes.
  - SHAP values serve as a feature of importance analysis [30] for the given models, offering meaningful insights into their outputs.
- *Model confidence and confidence intervals:* A per-sample model score and a matching confidence interval are provided by the configuration of the model confidence and confidence intervals.
  - We have used NumPy and calculated metrics such as the Mean Absolute Error (MAE) to assess the confidence and uncertainty associated with model predictions.
  - This setup guarantees a measurable indicator of the model's prediction accuracy, allowing for a more thorough comprehension of prediction dependability and promoting well-informed decision-making, particularly in crucial applications.
- *Drift detection and automated model re-training:* We have integrated NannyML into the Intelligence Layer, which enables real-time tracking of data distribution shifts and model performance estimation.
  - This can be used for univariate and multivariate drift detection, as well as a triggering system to identify when models require retraining based on poor performance; thus, improving the robustness of the models deployed in the registry if this is added to the model pipelines.
- *Federated Learning* model training, for privacy-preserving and enhancing security by not moving data out of the nodes.
  - This leverages the Flower framework that enables collaborative model training across multiple decentralised devices or organisations while preserving data privacy.
  - Once local models are aggregated into one, this aggregated (global) model becomes part of the AI coordination backend model registry, and it is made available to be used for inference (as any other model trained through the Intelligence Layer).

The next four subsections go more in-depth into each of them. Technical specifications of the Software developed in this module are provided in Annex II.H.

3.3.2.1 Explainable AI

To ensure transparency and reliability in ICOS, we will employ explainable AI techniques to interpret and monitor the models provided by the Intelligence Layer as described in D4.1. This toolset is aimed to allow data scientists or users using AI models, to identify and mitigate biases, enhancing the accuracy and fairness of AI systems.

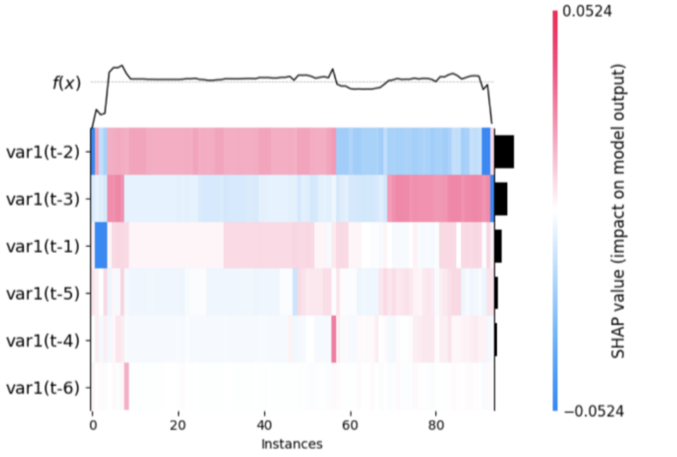


Figure 9. Summary bar plot highlighting the average importance of top features

This element of the Trustworthy AI module provides visual tools to help users understand how models arrive at their conclusions.

- By demystifying the decision-making process, we aim to increase trust and confidence in the models, enabling users to make informed decisions. To enhance model interpretability, we have integrated SHAP (SHapley Additive exPlanations) into the Intelligence Layer's model training pipeline.
- This tool generates visual dashboards that illustrate how the model arrives at its predictions, promoting transparency and trust. For this purpose, N samples from the inference data are used for SHAP analysis, allowing the user to customise this parameter. By default, this value is set to 100.

Examples are visualised in Figure 9 and Figure 10.

- Figure 9 shows a bar plot illustrating the average importance of the top features, with SHAP values ranging from negative (blue) to positive (red), highlighting the model’s output per feature vector. This shows that lags 1, 2, and 3 are the most significant, while other variables contribute less.
- Figure 10 consists of a waterfall plot depicting the stepwise contribution of *E[f(X)]* , where SHAP values are notably high for lags 3 and 2, followed by lag 5.

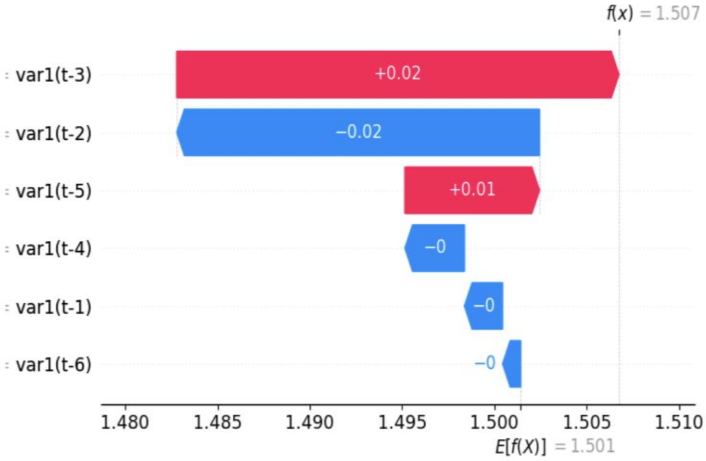


Figure 10. Waterfall plot showing the stepwise contribution of individual features to a specific prediction

MLFlow is used to store the SHAP explanations. For this purpose, SHAP explainer is initialised for the model’s predictions on a subset of the test data, which helps us interpret the model’s output through SHAP values. It then generates three types of SHAP plots - a waterfall plot, a bar plot, and a heatmap. The plots are created using the commands:

```
plot_func(shap_data, show=False)
mlflow.log_figure(figure=fig, artifact_file=file_name)
```

to log the figures for easy tracking and analysis. Just remember that the MLFlow tag ID should be used within the same session to keep all artifacts organised in one place.

3.3.2.2 Prediction confidence scores

In IT-2, the *AI coordination backend* API responses incorporate a *confidence score* and a *confidence interval* alongside the predicted value, providing a more comprehensive assessment of the prediction.

- The confidence score provides this information, indicating how closely the predicted values align with the true values by looking using in this case the expected Mean Absolute Error (MAE).
- The confidence interval, computed by specifying a certain range (CI – 95% in this development made in IT-2), creates a mask that determines whether each prediction lies within that range.
  - When there is a match within the CI, a flag is set to true; otherwise, it is set to false.

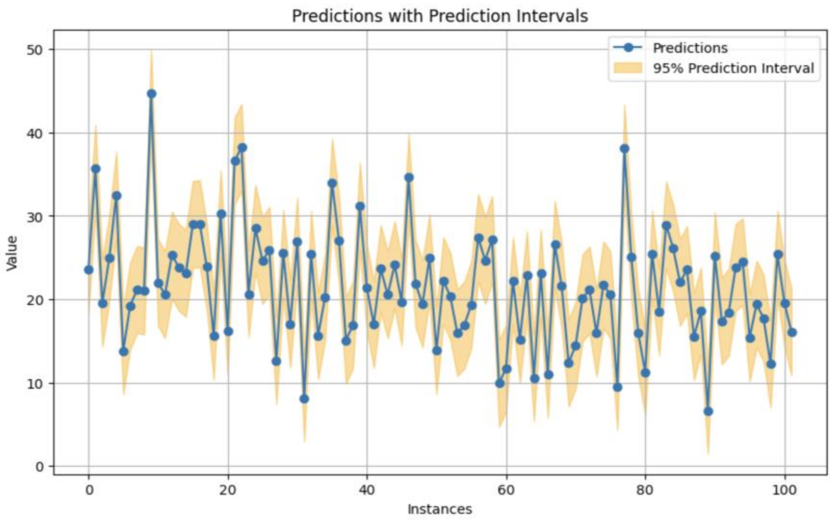
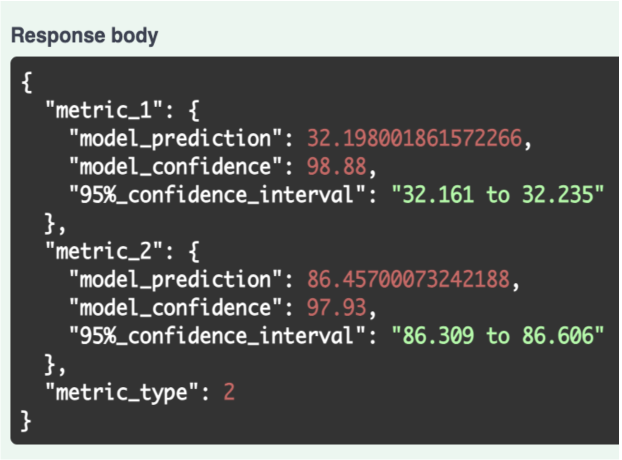


Figure 11. Confidence intervals in ICOS experiments over an XGBoost model

This enhancement is visually represented in Figure 11, where the shaded region indicates the area bounded by the upper and lower confidence intervals, and as API call a response body in Figure 12.



```
{
  "metric_1": {
    "model_prediction": 32.198001861572266,
    "model_confidence": 98.88,
    "95%_confidence_interval": "32.161 to 32.235"
  },
  "metric_2": {
    "model_prediction": 86.45700073242188,
    "model_confidence": 97.93,
    "95%_confidence_interval": "86.309 to 86.606"
  },
  "metric_type": 2
}
```

Figure 12. Body response with model confidence through the Intelligence API

3.3.2.3 Model monitoring

As AI models in production are susceptible to performance degradation over time, we have integrated a monitoring system to detect data shifts and model performance degradation, ensuring the trustworthiness of deployed models.

- This monitoring functionality in the Trustworthy AI module aids on identifying shifts, drifts, and impacting AI models or their data they receive for training and inference.
- For this, we have integrated NannyML for monitoring and performance estimation.
  - When underperformance is detected, the module initiates strategies such as retraining with new data or model replacement.
  - To facilitate this continuous monitoring process, the module collects, reports, and analyses incoming data, leveraging historical performance metrics stored in the Intelligence coordination backend model registry.

- This enables the module to assess deviations from expected performance and take appropriate actions.

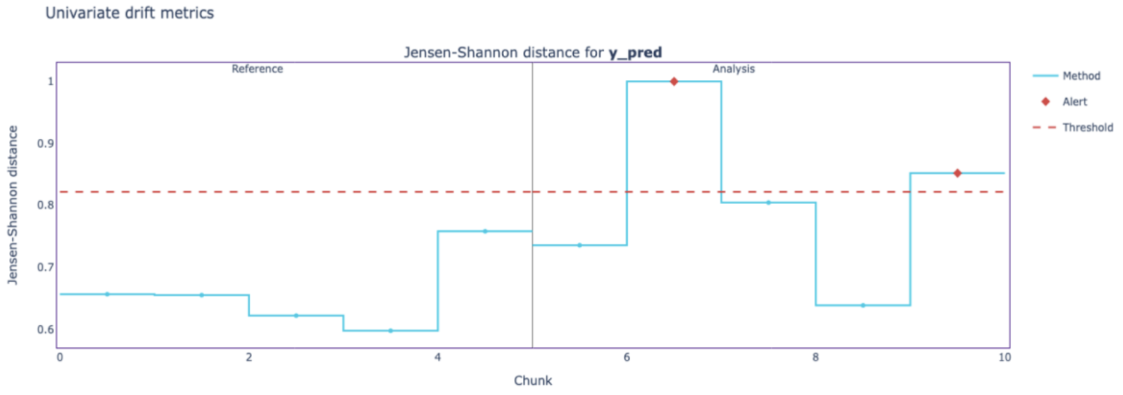


Figure 13. Historical data (reference) vs data received while the model is in the registry (analysis)

Figure 13 indicates potential shifts marked as red alerts over the performance of an XGBoost model.

- This is achieved by storing a NannyML object ain the model registry along with trained models.
  - This object analyses training data predictions by dividing them into time-ordered chunks based on a specified window size for later comparing the baseline distribution (reference data) with data received for at inference time (analysis data).
  - Drift detection is activated through the `data_drifts` Intelligence API (*backend*) call, utilising statistical measures (Jensen-Shannon divergence [19] in this case) to identify shifts in predictions.

3.3.2.4 Federated learning

Federated Learning was initially introduced in the Trustworthy AI module in D4.1 for privacy-aware model training.

- The core part of this component is ICOS Federated Learning (**ICOS-FL**), a customisable and pip-installable function built during IT-2 on top of the Flower FL framework. ICOS-FL initiates the FL learning process upon receiving a request from the ICOS applications.
  - ICOS-FL focuses on distributed model training and aggregation.
  - The current prototype trains an LSTM model to predict key metrics such as CPU usage, memory usage, and power consumption. This designed to be easily generalised, allowing for the integration of diverse computational metrics across different environments to facilitate flexible Federated Learning training cycles. A usage demonstration of this can be found in Annex II.E.3.
  - Data is retrieved from the ICOS monitoring pipeline, which relies on a stack consisting of Scaphandre (for energy usage data), Prometheus (for metric collection), OpenTelemetry, and the dataClay OTLP (OpenTelemetry Protocol) bridge.

ICOS-FL enables multiple nodes (called **clients**) to collaboratively train a shared model under the coordination of a central server (the **aggregator**).

- Each client trains the model on local data and returns updates (gradients or model weights) to the server.
- The server aggregates these updates to produce a global model.

This approach enhances privacy because raw data never leaves the clients while achieving an overall model that benefits from the combined knowledge of all participants.

Key functional aspects include:

- 1) Federated Learning Loop: This process repeats for a specified number of rounds (or until a certain convergence criterion is met).
  - The server (ICOS-FL server) coordinates the training rounds by sending an initial model to the clients. This task is planned to be initiated from ICOS Intelligence in the Controller suite. The ICOS-FL server is initiated through the dataClay OTLP bridge, thus, all steps below and model aggregation activities will be offloaded as AI training tasks.
    - This is integration piece to aggregate models through dataClay is expected to be finished by the ICOS final release.
  - Each client trains locally using its assigned data (metrics from dataClay).
  - The server aggregates the model updates to form a new global model by following a specified aggregation policy—FedAvg in the prototype, or alternatively FedProx, FedSGD, and others.
- 2) LSTM Model for Time Series Prediction:
  - The platform currently supports an LSTM model that predicts CPU usage, memory usage, and power consumption.
  - Continuous retraining is supported, allowing the model to be periodically retrained as new data arrive.
- 3) Data Ingestion:
  - Real-time metrics are collected by Prometheus (which fetches them from Scaphandre) and exposed through the OpenTelemetry collector.
  - The dataClay OTLP bridge transforms those metrics into dataClay objects, effectively serialising them into DataFrames for the training process.
- 4) Model Export and Deployment:
  - The resulting model is exported in BentoML [25]format, making it portable for inference through the Intelligence API (*backend*) or further use in downstream applications.

The ICOS-FL server is planned to be triggered under the control/coordination of a Controller through the OTLP dataClay bridge.

- The intelligence backend will initiate federated learning training tasks outside the Controller (e.g. aggregation), determining how many nodes should participate in the training process.
- The ICOS-FL client takes advantage of data available in the local dataClay instance (through the OTLP dataClay bridge, see Sections 2.2.2 and 2.2.3).
  - Local training is then offloaded to this dataClay instance guaranteeing efficient training without unnecessary data transfers.
  - Once the training job completes, the final global model is stored and made available to the rest of the ICOS environment.

The next subsections delve into the technical details of how ICOS-FL and its supporting services (Prometheus, OpenTelemetry, the dataClay OTLP bridge, etc.) work together to provide a federated learning solution for system metrics.

3.3.2.4.1 Prototype architecture

Figure 14 illustrates the prototype architecture.

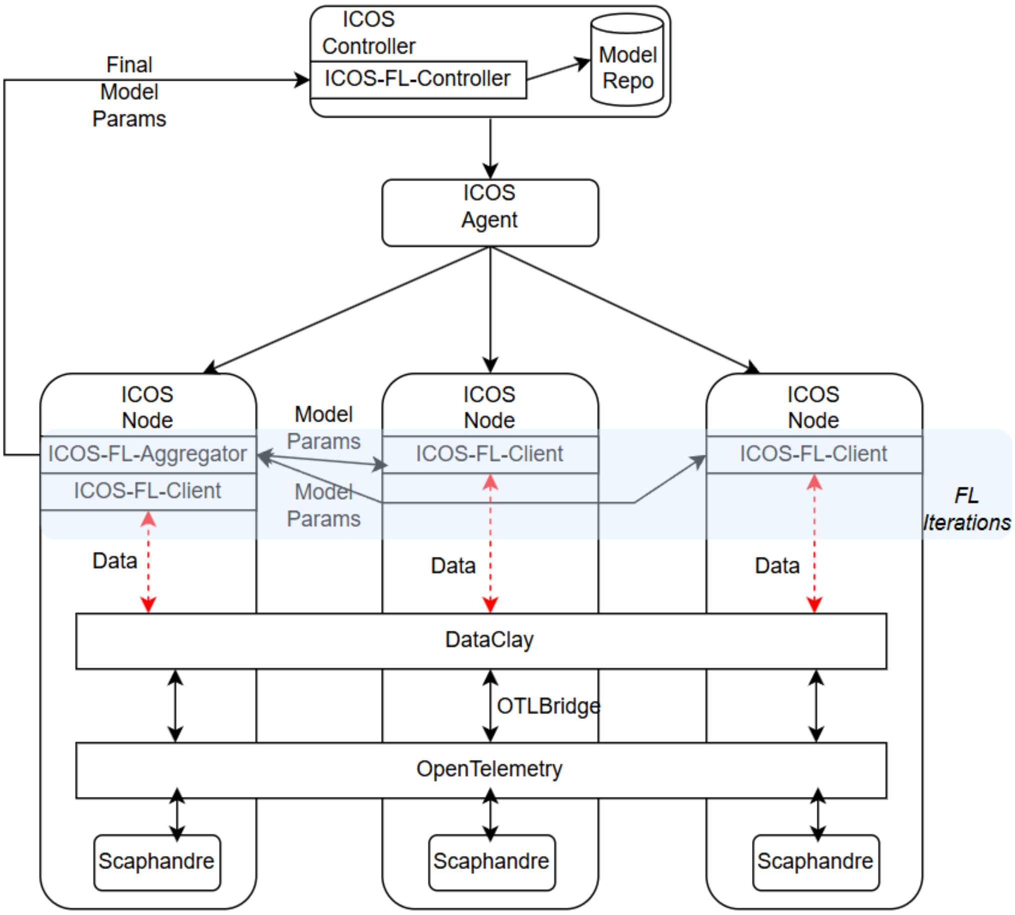


Figure 14. FL prototype architecture

It contains the following elements:

- **Scaphandre**: Runs as a container to collect power consumption metrics.
- **Prometheus**: Gathers metrics (including those from Scaphandre) and provides an endpoint for scraping these metrics.
- **OpenTelemetry Collector**: Receives metrics from Prometheus (and potentially other sources), packaging them in OpenTelemetry format.
- **dataClay OTLP Bridge**: Connects the OpenTelemetry stream to dataClay. Metrics are batched or streamed, then serialised into dataClay objects (typically transformed into a pandas DataFrame).
- **ICOS Federated Learning (ICOS-FL)**: Composed of two main pip-installable packages:
  - `icos_fl_server`: Coordinates the FL process, aggregates client updates, and stores/export the final model.
  - `icos_fl_client`: Performs local training rounds and sends model updates back to the server.

Annexes II.E.1 and II.E.2 provide, respectively, sequence diagrams for this process and a more detailed description of each of the FL process components from Figure 14.

3.4 AI Analytics

The AI Analytics framework was introduced in D4.1 as a toolkit for building and deploying machine learning models. The first version of the model, as presented in D4.1, aggregates Intelligent energy-aware task offloading, load time series forecasting, forecasting upcoming load anomalies, and CPU utilisation forecasting to improve workload distribution. This module is designed to be efficient and scalable, making it ideal for use in large-scale projects.

Building upon the existing framework, IT-2 introduces several key enhancements:

- AI models training enhancements
  - Improved energy forecasting and support for multivariate and multioutput metrics forecasting.
- MLOps capabilities extended for enhanced model tracking and management.
- Model compression techniques implemented to minimise model size and reduce computation.

3.4.1 AI Models

This subsection presents models that come pre-trained as part of the Intelligence Layer model registry. Most of them are also trainable through the AI Analytics module and using the Intelligence layer coordination APIs. These models focus on the following prediction tasks:

3.4.1.1 Intelligence for resource-allocation

D4.1– "*Data management, Intelligence and Security Layers (IT-1)* introduced basic functionality for the three types of models below.

- Anomaly detection: This element of AI Analytics is focused on classifying anomalies in the load-balancing change patterns. This remains the same as in D4.1.
- Load balancing and energy forecasting: This model is intended to predict future load balancing considering energy consumption. In IT-2 there have been changes and formal experiments.
  - Their numerical results and testing using the **Decentralised Computation Offloading scheme (DECOFFEE) for Energy Efficiency** across ICOS Layers [11] are presented in Annex II.F.
  - Its training process is intended to be performed through the federated learning process presented in Section 3.3.2.4 to reduce data movement.
- Metrics forecasting: In IT-1, a training and inference pipeline was presented to predict **CPU and RAM utilisation** on 5-minute intervals using univariate models. This was presented as the first integration between the Intelligence Layer modules and data management.
  - In IT-2 there have been changes to improve AI models and pipelines provided as part of the component. These are covered in Annex II.G.
  - The version of this component now employs PyTorch LSTMs to forecast CPU and memory usage, supporting both univariate and multivariate (simultaneous CPU/memory) approaches.

3.4.1.2 Intelligence for security

At the time of writing this report (M30), experiments are going on for a fourth and last model to be trainable through the AI Analytics module using Tetragon [36] logs to identify security issues.

- This piece of work, which integrates Security and Intelligence Layers is covered in Section 4.8 and will be reviewed in the ICOS Final release (D5.3) [8].

As introduced at the start of Section 3, this is not the only piece of integration between Security and Intelligence.

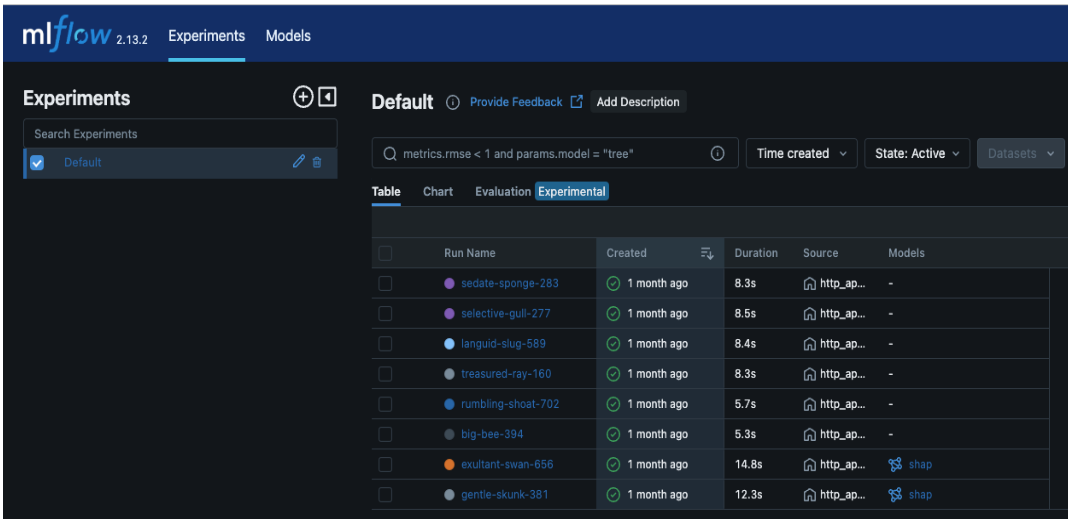
- Deliverable D2.4 already covered the integration between LOMOS for anomaly detection to be used through the Intelligence Layer Coordination backend API.
  - LOMOS is covered further in Section 4.5.

3.4.2 AIOps system

IT-1 provided AIOps by providing the Intelligence backend API for AI as a service, allowing model training and inference on demand and connected to the Intelligence Layer model registry allowing continual learning through model reuse [18] and fine-tuning.

In IT-2, Intelligence Layer leverages MLFlow [22], an open-source platform, to facilitate the development and deployment of new AI models.

- This provides capabilities for tracking projects, models, and experimental parameters, enabling users to conduct various experiments and effectively monitor their outcomes through detailed logs. See its user interface in Figure 15.



Run Name	Created	Duration	Source	Models
sedate-sponge-283	1 month ago	8.3s	http_ap...	-
selective-gull-277	1 month ago	8.5s	http_ap...	-
languid-slug-589	1 month ago	8.4s	http_ap...	-
treasured-ray-160	1 month ago	8.3s	http_ap...	-
rumbling-shoat-702	1 month ago	5.7s	http_ap...	-
big-bee-394	1 month ago	5.3s	http_ap...	-
exultant-swan-656	1 month ago	14.8s	http_ap...	shap
gentle-skunk-381	1 month ago	12.3s	http_ap...	shap

Figure 15. MLFlow UI showing experiments of each model training request

3.4.2.1 Functional description

IT-1 integrated BentoML [25] as an initial piece of AIOps for the Intelligence Layer with the capability to track model metadata, performance metrics, and data split information during each training iteration.

In IT-2, this framework has been enhanced by

- incorporating model explainability techniques (see Trustworthy AI module) and
- by capturing learning curves for neural network models (e.g., PyTorch) throughout the training and evaluation phases.

These additions provide a more comprehensive and visually informative representation of model performance.

3.4.2.2 Technical description

MLFlow can be divided into multiple components. The two most relevant to this integration piece are:

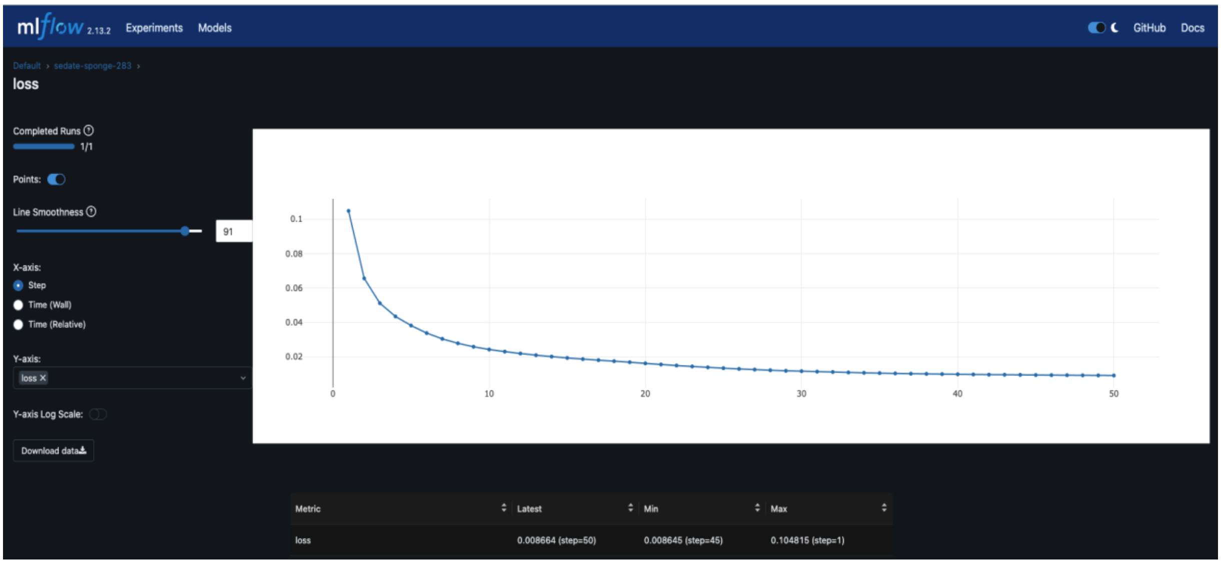
- 1) MLruns: These are directories generated upon MLFlow initialisation at the commencement of model training.
  - An mlrun houses essential metadata pertaining to each training experiment, including model metrics, descriptions, and dataset information.
- 2) MLFlow UI: User interface that integrates data visualisation to facilitate the exploration of experiments related to each model training run.
  - This tool provides a visual representation of the experiments, along with the associated model parameters captured during the training process.

During each model training iteration, the `mlflow.start_run()` function is invoked from the Intelligence backend to initiate an MLFlow experiment and create an `'mlruns'` folder to log experimental data. Subsequently, metadata information is logged such as

- performance metrics (`mlflow.log_metrics(performance_metrics)`),
- hyperparameters (`mlflow.log_params(datasplit_information)`),
- and model tags (`mlflow.set_tag("bentoml.model", bento_model.tag)`).

For DL frameworks like PyTorch [28] and TensorFlow [29], the Intelligence backend in IT-2 can log loss curves for each epoch, enabling a detailed analysis of model performance as illustrated in Figure 16.

Furthermore, IT-2 incorporates a storage optimisation mechanism by limiting the number of experiments tracked for each run. This limit is configurable by the administrator within the Intelligence Layer container configuration file, with a default value of *10*. To prevent excessive storage consumption, the system automatically deletes older experiments when the configured limit is reached.



loss

Completed Runs 1/1

Points: ●

Line Smoothness 91

X-axis:
● Step
○ Time (Wall)
○ Time (Relative)

Y-axis: loss ×

Y-axis Log Scale: ○

Metric	Latest	Min	Max
loss	0.008664 (step=50)	0.008645 (step=45)	0.104815 (step=1)

Figure 16. Loss curves representing model’s training loss reducing with each epoch

3.4.3 Model compression

The Intelligence Layer in IT-2 introduces model compression as a set of techniques designed to reduce computational overhead while maintaining predictive effectiveness.

3.4.3.1 Functional description

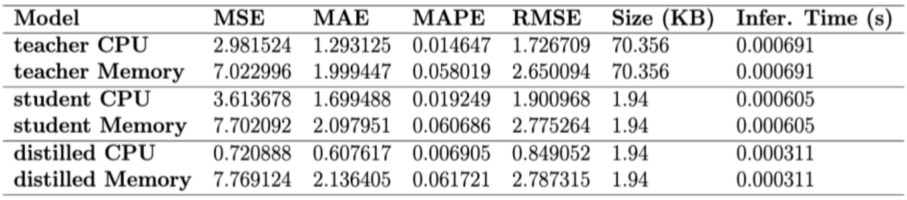
- The aim here is to enhance the operational efficiency of AI models in the continuum by:
  - reducing model size (MB),
  - optimising memory consumption (MB) and
  - accelerating inference times.
- The methods implemented address scalability challenges in resource-constrained edge and specific use cases where low inferencing times are required. This addresses the ICOS requirement CM_FR_14 – model optimisation techniques. See this requirement in Annex II.A.

3.4.3.2 Technical description

Two core techniques for model compression have been implemented and inbuilt into the metrics forecasting model training pipeline in the Intelligence Layer coordination backend API:

- 1) Knowledge distillation: This compression mechanism performs a knowledge transfer to a smaller architecture.
  - Distilled model trained using an RNN from an initial LSTM model (baseline).
  - Model size reduction of ~ 70x while obtaining comparable performance metrics.

- These results can be observed in Figure 17. Similarly, Figure 18 depicts the teacher, student and distilled model performance over time.
- The distilled model generalised better to CPU and Memory time series in some experiments.



Model	MSE	MAE	MAPE	RMSE	Size (KB)	Infer. Time (s)
teacher CPU	2.981524	1.293125	0.014647	1.726709	70.356	0.000691
teacher Memory	7.022996	1.999447	0.058019	2.650094	70.356	0.000691
student CPU	3.613678	1.699488	0.019249	1.900968	1.94	0.000605
student Memory	7.702092	2.097951	0.060686	2.775264	1.94	0.000605
distilled CPU	0.720888	0.607617	0.006905	0.849052	1.94	0.000311
distilled Memory	7.769124	2.136405	0.061721	2.787315	1.94	0.000311

Figure 17. Distilled model results vs teacher and student performance

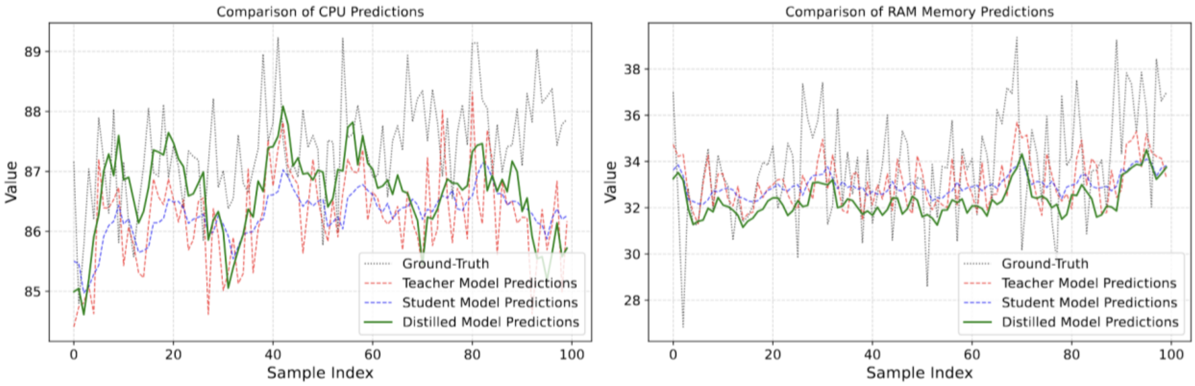
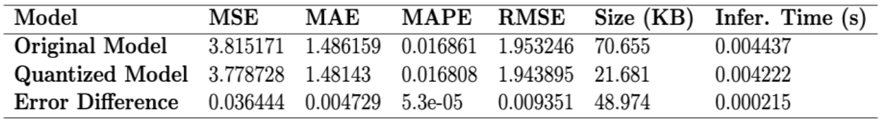


Figure 18. CPU (left) and RAM (right) preds for teacher, student, and distilled models

- 2) <u>Dynamic quantisation</u>: This model compression method is based on reducing the precision of several model parameters.
  - Weights and activations of specific model layers are converted from floating points 32 (FP32) to integer 8 (INT8).
  - This procedure has been shown to improve the model size by 3x, while also reducing the inference time This is shown in Figure 19 and Figure 20. We can see that our experiments reduced the CPU usage from 70.356 KB to just 21.68 KB.



Model	MSE	MAE	MAPE	RMSE	Size (KB)	Infer. Time (s)
Original Model	3.815171	1.486159	0.016861	1.953246	70.655	0.004437
Quantized Model	3.778728	1.48143	0.016808	1.943895	21.681	0.004222
Error Difference	0.036444	0.004729	5.3e-05	0.009351	48.974	0.000215

Figure 19. Quantisation results in experiments for CPU metrics forecasting

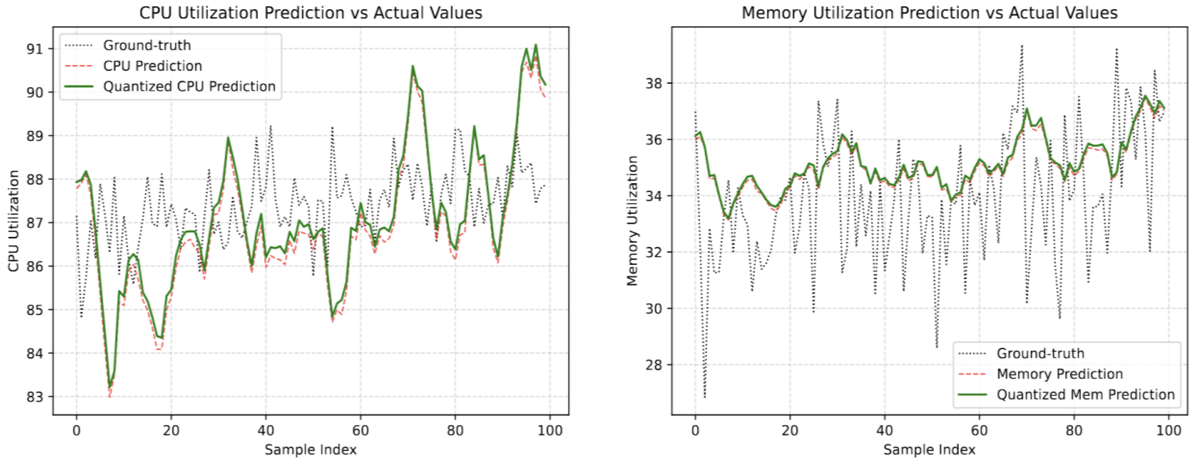


Figure 20. CPU (left) and RAM (right) preds for the quantised vs non-quant. models and ground truth

Model pruning has been discarded as a compression method from IT-2 after initial experiments not contributing to improving model efficiency while maintaining their accuracy. Technical specifications for model compression are covered in Annex II.H.2.

3.5 Data Processing

The Intelligence Layer encompasses a module dedicated to data preparation, transformation, and read/write operations.

- This module offers libraries and frameworks for transforming raw data extracted from the data management component (dataClay [21]) into a suitable format for model training.
- It also facilitates distributed workload by offloading computational tasks such as intensive model training to other ICOS nodes when a request is made to the Intelligence Layer.

In IT-1, basic data processing was implemented using raw dataClay data (see deliverable D4.1) along with task offloading. IT-2 builds upon this foundation by enhancing the data-cleaning process and introducing support for multivariate models.

3.5.1 Functional description

This module serves for three purposes:

- Data access through data management: Extracting raw training dataset from dataClay by connecting to its client and reading the dataset as an alias that is used as unique identifier in the dataClay server.
- Data transformation and pipelining: Once the data is accessed, the data processing module performs further operations over such dataset to clean it and transform it for modelling.
- Model training offloading: since the dataClay client is part of the Data Processing module, AI offloading is actioned by the Intelligence backend interacting with this module and AI Analytics.

3.5.1.1 Fitting into the ICOS architecture

The data processing module is part of the Intelligence coordination API, which is deployed at the controller; a more detailed description of the architecture is already provided in deliverable D4.1 [7]

3.5.2 Technical description

This module as presented in IT-1 performs the following actions.

- 1) the raw dataset obtained from the data management component undergoes preprocessing tailored to the specific AI model requirements:
  - univariate models demand tabular data where feature vectors should be structured as a set of single value lags or timesteps, while
  - multivariate models require feature vectors including values for multiple metrics.
- 2) Subsequently, the pre-processed dataset undergoes further processing steps as outlined in the sequence diagram within Section 3.1.2.2 of deliverable D4.1 [7].
- 3) Finally, upon request, the model training task can be offloaded to another ICOS node using dataClay, as detailed in the sequence diagram within Section 4.12 of deliverable D2.4 [6].

IT-2 introduces several enhancements, including the integration of **pipelines** and code optimisation over the Intelligence AI training offloading presented in D2.4. This optimises the source code, particularly for scenarios where local model training is preferred, with the aim to improve efficiency and reduce resource consumption by avoiding the overhead associated with offloading operations when not required.

3.5.2.1 Prototype architecture

The interaction of the data processing module with the Intelligence Layer coordination API and the data management component is illustrated in Figure 4, Section 3.1.

- A detailed architectural description was presented in the definition of the Intelligence coordination frontend in Section 3.2.2.2; this included an overview of the data flow, illustrating how telemetry data is retrieved from its source, subsequently transferred to dataClay, and ultimately utilised by the Intelligence Layer coordination backend for further processing and analysis.

3.5.2.2 Description of components

The data processing module comprises a suite of components, including data ingestion, transformation, splitting, model training (with offloading capabilities), evaluation, and deployment. These components have already been described in deliverable D4.1.

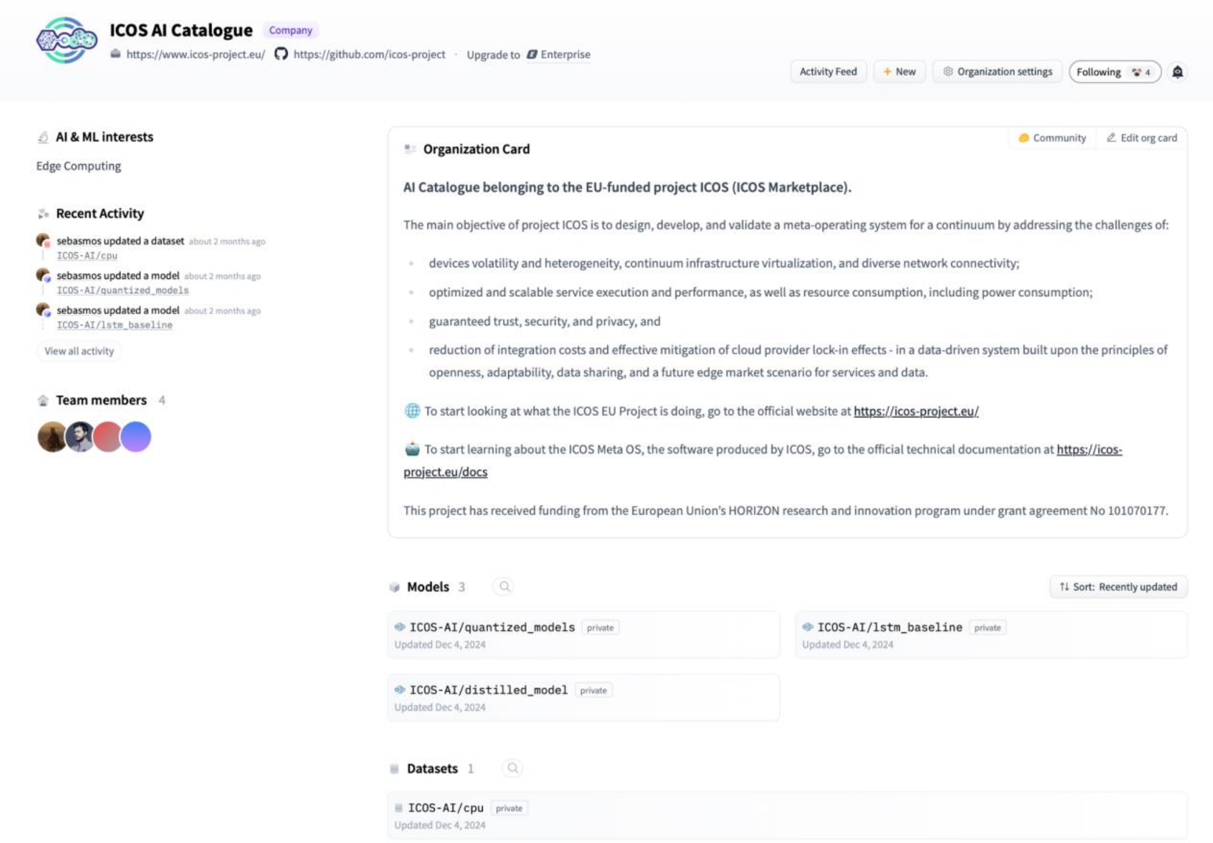
Given that these components are already embedded within the AI pipelines used in AI Analytics, a separate "*Technical specification*"" section is not required.

3.6 AI Models and Data Repository

The fifth module of the Intelligence Layer is the "AI Models and Data Repository". This project is openly accessible in Hugging Face (https://huggingface.co/ICOS-AI) and will be delivered as part of D4.3 – "ICOS Dataset and AI models marketplace". This section introduces the repository and its components.

3.6.1 Functional description

A preliminary view by M30 of ICOS is shown in Figure 21.



sebasmos updated a dataset about 2 months ago
ICOS-AI/cpu

sebasmos updated a model about 2 months ago
ICOS-AI/quantized_models

sebasmos updated a model about 2 months ago
ICOS-AI/lstm_baseline

AI Catalogue belonging to the EU-funded project ICOS (ICOS Marketplace).

The main objective of project ICOS is to design, develop, and validate a meta-operating system for a continuum by addressing the challenges of:

- devices volatility and heterogeneity, continuum infrastructure virtualization, and diverse network connectivity;
- optimized and scalable service execution and performance, as well as resource consumption, including power consumption;
- guaranteed trust, security, and privacy, and
- reduction of integration costs and effective mitigation of cloud provider lock-in effects - in a data-driven system built upon the principles of openness, adaptability, data sharing, and a future edge market scenario for services and data.

🌐 To start looking at what the ICOS EU Project is doing, go to the official website at https://icos-project.eu/

To start learning about the ICOS Meta OS, the software produced by ICOS, go to the official technical documentation at https://icos-project.eu/docs

This project has received funding from the European Union's HORIZON research and innovation program under grant agreement No 101070177.

ICOS-AI/quantized_models private
Updated Dec 4, 2024

ICOS-AI/lstm_baseline private
Updated Dec 4, 2024

ICOS-AI/distilled_model private
Updated Dec 4, 2024

ICOS-AI/cpu private
Updated Dec 4, 2024

Figure 21. Preliminary view of the online ICOS AI Models and Data Repository

Figure 21 shows example repositories containing datasets and models used by ICOS Intelligence. This module, not covered in D4.1, is described for the first time in the following subsections. Extra technical documentation will be given in the form of an appendix to D4.3- "*ICOS Dataset and AI models marketplace*".

3.6.1.1 Fitting into the ICOS architecture

The AI models and Dataset Repository, formerly known as "*AI Models Marketplace*", in the project's grant agreement [1], has been previously described as "A collection of pre-trained analytics and ML models to be reused, updated, and refined to foster the application of new AI techniques in the ICOS meta-OS". While the described functionality is provided at the Controller level by the Intelligence model registry, the online ICOS AI marketplace is online and reachable not only through the Intelligence Layer but also through a web interface.

As covered in "*Section 3.3.2.5 AI Models Repository*" in D2.4 – "*ICOS Architectural Design (IT-2)*", this *cloud-based repository* differs from *model registries* in the controller.

- **Model registries** have a version of the current models running on the device.
- The **AI models' repository** aims to contain versions of prior successful models for meta-OS-related and user layers.

See both represented in Figure 22. Model registries appear inside the controller, while the AI repository is conceptually outside ICOS.

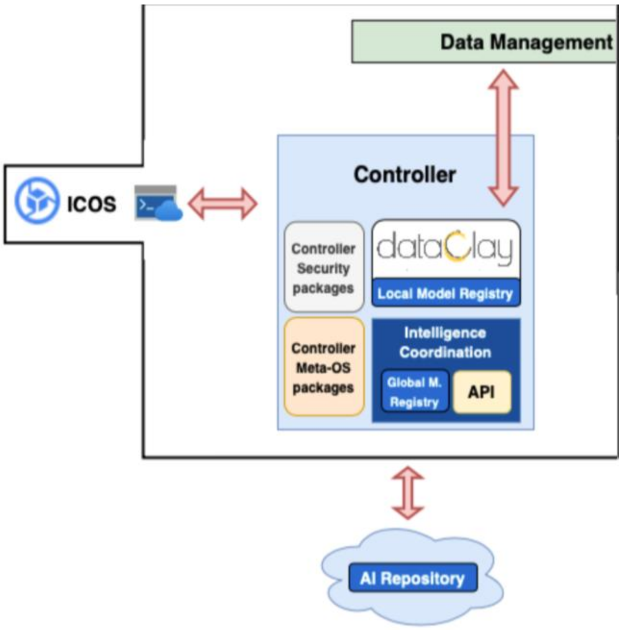


Figure 22. Fragment of Figure 7 in D4.1. AI repository as online catalogue external to ICOS elements

3.6.2 Technical description

The AI models repository module, called AI marketplace or ICOS marketplace in the project proposal, is made of:

1. **model and data repository** available online and
2. a **client** in the Intelligence coordination module (developed using Python 3.10) to allow pulling and pushing models and datasets between ICOS intelligence and the repository.

Both elements are described in more detail in the next subsections.

3.6.2.1 Prototype architecture

This component is made of two different elements: i) **repositories** and ii) **Intelligence Layer's client**. These are presented in the next two subsections.

3.6.2.1.1 Model and data repository

This is a repository containing:

1. models developed using ICOS Intelligence and AI Support functions that were successfully deployed and validated, and thus shared in the repository, plus
2. sample datasets used to train these models as well as
3. data gathered using ICOS.

Example models and data are:

- Metrics forecasting models: CPU, Energy and security related models part of the registry
- Use cases related models: predictive maintenance model and energy management model
- Telemetry metrics data: sample data captured in the ICOS testbeds used to train metric forecasting models.
- Open energy data: energy data captured using Scaphandre [31] and utilised for energy forecasting models (e.g. DECOFFEE) in ICOS. See D4.1 Data management, Intelligence and Security Layers IT-1.

Each model and dataset will have their own repository. Note that models and datasets have different sections in the ICOS "*AI Models and Data Repository*"; see Figure 21. Such repositories will use **model cards** and **dataset cards** respectively to list relevant information of models and data.

- **Model cards**[8] give details regarding trained AI models stored in the repository. They cover details about the model's building process, underlying assumptions made during its creation, potential variations in its performance as well as its expected results [13]. Metadata referring to model information such as license, language, library, datasets, metrics, and eval_results can also be provided as a YAML file.
- **Data cards**[9] are the common format used in Hugging Face for dataset documentation. They work as model cards above, helping the community to understand the contents of the dataset and give a context on how this should be used. Dataset metadata can also be added in the form of a YAML file, allowing users to search for them.

Information of models and data will be represented using open model standards and data model schemas to optimise their search and reusability. The set of model and dataset fields to cover in their cards will be extracted from relevant literature reviews that analyse 32,111 model cards [14] and 7,433 datasets respectively [15]. For data cards, common metadata standards and common fields contained by semantic web ontologies (e.g., information contained by the RDF vocabulary DCAT[10]) will also be considered. The complete list of model and data information contained in the "*AI Models and Data Repository*" will be delivered in the appendix to D4.3 – "*ICOS Dataset and AI models marketplace*".

3.6.2.1.2 AI repository client

The AI Models and Data Repository is made available to the AI Coordination Layer through Intelligence API functions that allow pulling models from the repository into the model registry and pulling data into the data processing module as well as the inverse process pushing models and data into it.

This client allows seamless interactions between the online ICOS AI "*Model and Data Repository*" and the ICOS Intelligence Layer. It is actioned by the AI coordination module both for model and data pull and push actions. Detailed documentation as well as configuration steps for the integration between

[8] Model cards: https://huggingface.co/docs/hub/en/model-cards

[9] Data cards: https://huggingface.co/docs/hub/en/datasets-cards

[10] Data Catalog Vocabulary (DCAT) - Version 3: https://www.w3.org/TR/vocab-dcat-3/

ICOS Intelligence and the AI Models and Data Repository will be delivered in an Annex document part of D4.3 – "*ICOS Dataset and AI Models Marketplace*".

3.6.2.2 Description of components

The Intelligence Layer uses both push and pull mechanisms to synchronise the ICOS AI and Data Models repository and the Intelligence Layer. This synchronisation is a choice made by the ICOS Controller administrator or relevant technical users able to make decisions concerning the models running in ICOS. While push and pull mechanisms will be covered in further detail in D4.3 together with a "Technical specification" section, they are initially introduced as follows.

Push mechanisms

- Models: models should be first stored in the Intelligence model registry. The model's name and version may be specified in an API call in the form, defining model card parameters as well as model metadata.
- Datasets: once the data processing module loads a dataset, its dataset alias can be specified to the Intelligence API, which is part of AI coordination, to be pushed into the AI Models and Data Repository with data-card-related information and metadata associated. As for models, developing an API call will be considered for this operation. The user configured in the repository client should have special permissions to push into the AI Models and Data repository.

Pull mechanisms

This is expected to follow the inverse mechanism described above. Thus:

- Models will be loaded into the Intelligence model registry. For this, BentoML and Hugging Face integration[11] to load models will be explored.
- Datasets will be downloaded and assigned an alias to be used by Intelligence.

The Intelligence Layer will be able to pull datasets to train models with data used by the ICOS community. Members of the ICOS-AI organisation in Hugging Face will also be able to push and share new datasets and models with the Edge-to-cloud Continuum community. The process to submit models and datasets to the repository will be described as part of D4.3 – "*ICOS Dataset and AI Models Marketplace*".

[11] BentoML-HF 1: https://docs.bentoml.com/en/latest/build-with-bentoml/model-loading-and-management.html

3.7 Limitations and future work

A manner to review the work completed by M30 and its limitations is by looking at the successful completion of the system requirements (see Annex II.A). Additionally, D2.2 [2] defined a set of System Use Cases (SUC) for Intelligence. They are listed below.

- SUC_AI_1 – Manage Model
  - This was initially achieved in IT-1 through the Intelligence coordination backend, its API and its model registry.
- SUC_AI_2 – Train Model
  - Achieved in IT-1 to train univariate models for utilisation metrics.
  - This has been improved in IT-2 for multivariate modelling.
- SUC_AI_3 – Clean and Pre-Process Data
  - This is part of the data and model pipelines developed through IT-1 and IT-2, built in the Intelligence Layer.
- SUC_AI_4 – Validate Model
  - The AIOps framework integration (MLFlow) into the Intelligence Layer performed in IT-2 allows application developers to analyse model results and experiment using the Intelligence Layer environment, for a later push to the model registry at the Controller.
- SUC_AI_5 – Deploy Model
  - Developers or admins with access to the controller can deploy models in the model registry provided in IT-1.
  - In IT-2, model training functionality has been added to the ICOS Shell interacting with the Intelligence frontend. Models trained through this process are deployed into the Intelligence model registry through an automated process (see Section 3.2).
- SUC_AI_6 – Inference
  - IT-1 provided model inference functionality as part of the Intelligence Layer backend.
  - IT-2 provides ICOS Shell access to model inference using the Intelligence frontend (Intelligence Layer Coordination module).

A limitation in the work performed by M30 is that the intelligence coordination frontend does not expose all Intelligence functionalities listed for the backend (see Section 3.2). This will be further reviewed in the ICOS Final release (D5.3) [8]. Extra functionalities in the frontend may be left for future updates during exploitation.

- AI model-related SUCs (SUC_AI_7-10) – Classify Nodes, Detect and Classify Anomalies, Detect and Predict policies and requirements violations
  - AI models in the Intelligence Layer predict anomalies and future metric values from IT-1.
  - During IT-2, a workflow was co-designed between Intelligence Layer developing partners and the MetaOS module leaders to receive and self-identify new system metrics to be predicted and enable MetaOS modules to read such predictions from Telemetry. Thus, MetaOS modules (e.g., aggregator, and policy manager) read these forecasts and fetch them to their decision-making numerical and classification models for matchmaking, finding anomalies, and violations.
  - The AI Analytics module provides functionality to deploy and use models for tasks such as: energy consumption forecasting, load balancing, anomaly detection and utilisation metrics like CPU and RAM.
    - Pre-trained models for energy consumption, anomaly detection, metrics forecasting and load balancing are packaged in the Intelligence Layer, to tackle the cold-start problem [17] right after the ICOS setup in a given infrastructure.
    - The training functionality provided is metric-agnostic. Therefore, the Intelligence Layer can be personalised depending on the infrastructure needs. Infrastructure providers and ICOS users can

picture this model training and update functionality as Intelligence regular updates to keep it up to date with ongoing changes in the underlying infrastructure and workloads.

In regard to energy forecasting, this is achieved through the DECOFFEE framework, which has been designed to be trained through a federated learning process to avoid data movement.

- The DECOFFEE framework, although developed and tested as presented in Section 3.4.1.1 and Annex II.F, it cannot be fully integrated in the ICOS testing infrastructure due to the lack of heavy task traffic flows. This is attributed to the fact that DECOFFEE shows significant advantages in task execution latency and load balancing under the arrival of multiple tasks in the ICOS agents/nodes. Thus, although DECOFFEE is delivered and is operable in the frame of ICOS, we highlight the limitations of its full integration within ICOS Testbeds.
- In the ICOS final release (D5.3) [8], the intelligence API will incorporate an option for FL training inside the training call which will delegate the triggering of the FL-server to dataClay. This triggering option will be the first step to initialise the FL training process. Regarding FL aggregation, due to late meetings of an AI-MetaOS taskforce setup in December 2024, it was agreed -at the architectural level- with other ICOS partners to offload model aggregation through dataClay as any other AI training workloads. This piece of work aims to offload model aggregation conceptually as any other model training task using the Data Management component.

Finally, at the deployment side, the Intelligence Layer container, while boasting extensive support for various libraries, faces challenges during deployment at the controller stage that require some adjustments.

- One of these issues is a potential duplication of libraries between the Intelligence Layer containers, that live at the ICOS Controller, and dataClay containers that are deployed in other nodes of the ICOS infrastructure. This duplication of libraries has a high storage footprint (~ GBs).

This issue can be alleviated by devising a mechanism to store large libraries such as PyTorch or TensorFlow only at the dataClay containers so that during training offloading all workloads are diverted to the dataClay container and the intelligence container only receives the results or output of the training. However, this would make the Intelligence Layer dependant on dataClay containers, which may present other design conflicts due to the Data Management limitations presented in Section 2.4. While in M30 of the project, this seems out of scope in ICOS, any update on this integration task would be reviewed in D5.3 – "*Third ICOS Release: Complete ICOS version*" [8].

4 Security Layer

This section presents the progress regarding development and implementation of Security Layer modules and components from IT-1 as presented in D4.1 to IT-2 as presented in this deliverable. It is structured as follows: we present the final (IT-2) design of Security Layer in Subsection 4.1, progress regarding the Security Layer Coordination API in Subsection 4.2, the Security Scan in Subsection 4.3, the Identity and Access Management module in Subsection 4.4 and the new development of IT-2 versions of Anomaly detection module in Subsection 4.5, the Audit module in Subsection 4.6 and the Trust functionality in Subsection 4.7.

Limitations and future work are documented in Subsection 4.8. Finally, we document licensing and instructions to download and set up the Software in Annex III.C.

4.1 Fitting into the ICOS architecture

This subsection presents the final (IT-2) design of Security Layer modules, components and functionalities (see Figure 23):

1. The Security Layer Coordination module is located in the ICOS Controller and communicates with the Security Scan and the Telemetry Controller from the Meta-kernel layer.
2. The Security Scan is composed of server and agents' components. The agents are located on the ICOS nodes and communicate information about the detected anomalies to the server, located in the ICOS Controller. The server is integrated with the Security Coordination module API that exposes scan results in the form of security metrics to the Telemetry Controller.
3. The IAM is part of the ICOS Core suite and is deployed on a separate cluster. It provides API to ICOS Shell and ICOS services in ICOS Controller, Agent and Edge for authentication and authorisation of ICOS users and ICOS services.
4. The Anomaly detection module is part of the Security layer but since it is also an AI tool, it was a design decision to integrate it with the Intelligence Coordination Module API. The latter acts as an access point to the Intelligence Layer, verifying users' and services' permissions to access this and other modules in the Layer. The "engine" of the Anomaly detection module, LOMOS, is deployed outside of ICOS in an isolated environment where it has direct access to hardware needed to run the training (GPUs). LOMOS has a secure connection through VPN to the Telemetry Controller to query necessary data (logs) for training and inference. As LOMOS is in an isolated environment, we have developed a LOMOS API-2 (REST API) that enables to query LOMOS for the most useful results (e.g., top 10 anomalies, anomalies in a certain period). LOMOS API-2 is located at the ICOS Controller.
5. The Audit component is packaged as a unified helm chart that wraps each described component below. For the first layer of auditing Trivy [35] is deployed on the master of each edge infrastructure to analyse any application manifest and containers which are about to be deployed. Tetragon [36] runs on each participating ICOS node as an Audit agent and all event logs are collected centrally to the Tetragon master node (Audit server), also located on the ICOS node, before they are forwarded to the Telemetry Controller in the ICOS Controller using Prometheus exporter[12].
6. Trust is a system-wide functionality for guaranteeing that ICOS modules and components in the ICOS Controller and ICOS agent are what they claim they are and, in parallel, their communication is encrypted (mTLS). Also, ICOS users access the ICOS system through trusted and secure communication.

[12] https://github.com/prometheus/node_exporter

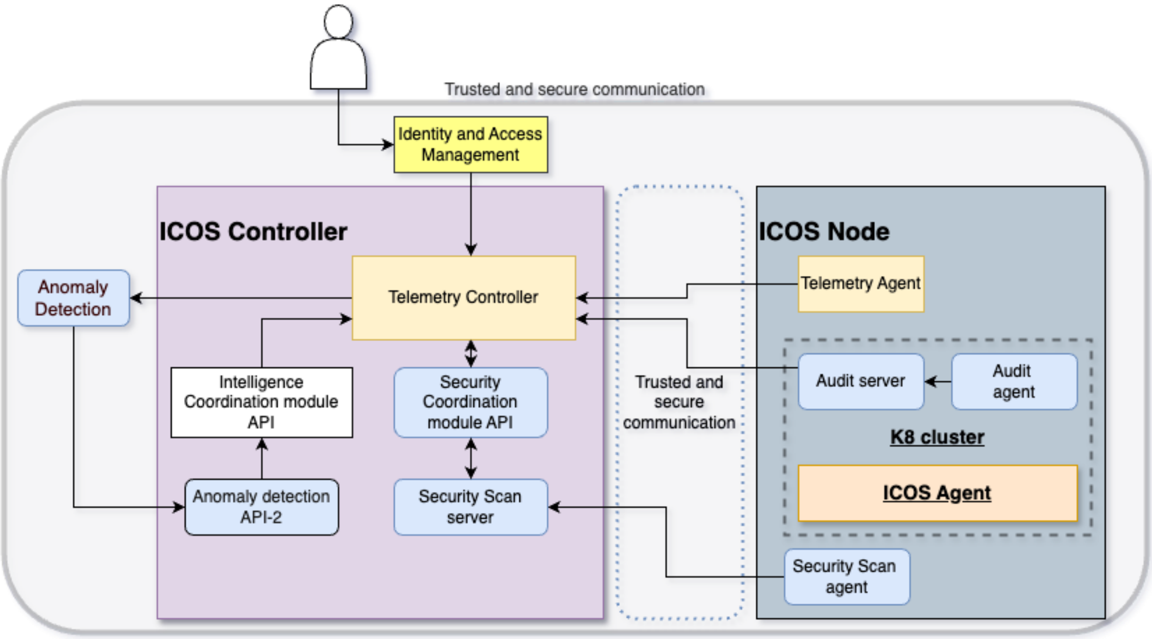


Figure 23. Security layer modules and functionalities in the ICOS architecture

4.2 Security layer coordination module

This section provides the updated functional and technical description of the Security Layer Coordination Module API from D4.1 [7].

4.2.1 Functional description

The updated version (IT-2) retains the functionality of acting as the reverse proxy for the inbound traffic, primarily to forward requests from the Telemetry for the security scans performed by the Security Scan module.

The functionality of forward proxy is deprecated. In the development since IT-1, the only Security Layer module that now communicates through the Security Layer Coordination Module API is the Security Scan. In this case the Security Coordination API acts as a reverse proxy. The Audit module communicates directly with the Telemetry and the Anomaly detection module communicates with the Intelligence Coordination module API. Other modules in the Security Layer were removed in the final ICOS architecture and their functionalities have been implemented as part of the Meta-kernel layer.

By acting as the reverse proxy for the Security Scan, the Security Layer Coordination module IT-2 contributes to meeting the following ICOS functional requirements from the D2.3 (see Table 12 in Annex III.A).

4.2.2 Technical description

The Security Layer Coordination module consists of multiple backend services joined together in a single Docker container. The IT-2 implements the reverse proxy functionality and aggregation of Security Scan results into security metrics consumed by Prometheus [24]. It is implemented using Python and FastAPI framework following OpenAPI specifications.

Time scheduling and CRON jobs were not implemented as there is no ICOS service that would need them.

The module is integrated with the Identity and Access management for checking the identity and permissions of users (e.g. other ICOS modules) trying to access the Security Scan module.

The module is distributed as a part of the ICOS Controller suite.

The API documentation in the form of Swagger UI is available here: http://10.160.3.20:31328/docs

Description of components:

- Reverse-Proxy API (from IT-1): Serves as a gateway for the Security Layer's incoming traffic/request.
- Aggregator of scan results (in IT-2): This component aggregates results from different security scans and provides aggregated results in the form of security metrics to Telemetry.
- Credentials & JWT management: A component that oversees handling API/service credentials and issuing JWT tokens for communication with modules inside the Security Layer (Security Scan) with the aim to achieve Zero-trust architecture.

4.3 Security Scan

This section provides the updated functional and technical description of the Security Scan module from D4.1 [7].

4.3.1 Functional description

In the updated, IT-2, version the Security Scan module is delegated to actively check for security issues on the ICOS nodes and report these issues to the Telemetry component and to the ICOS user as well as to check the security of deployed ICOS modules and components.

As in IT-1, the Security Scan module integrates the Wazuh security platform [41] to detect vulnerabilities (CVE), misconfigurations (CIS benchmark) and malware on the ICOS nodes. For checking the ICOS modules and components, in IT-2 we use Trivy [35] - a container image scanner that scans images for vulnerabilities (CVE), IaC issues and misconfigurations and SBOM (host OS packages and software dependencies). It is integrated in the ICOS CI/CD as a security check all ICOS services must pass before their code is published on the ICOS developer repository[13].

With the Wazuh and Trivy tools, the Security Scan module contributes to meeting the following ICOS functional requirements from the D2.3 [1] (see Table 12 in Annex III.A).

4.3.2 Technical description

There is no major change regarding Wazuh setup from IT-1. The only difference in IT-2 is that the Wazuh server is now containerised and deployed as part of the ICOS Controller suite.

The Wazuh agents still need to be manually installed on the ICOS node (VM or barebone). The documentation is provided here:

- https://www.icos-project.eu/docs/Administration/Computational%20Resources/wazuh_agent/

Trivy is implemented as a job in ICOS CI/CD.

Regarding more detailed description of components, prototype architecture and technical specifications, there is no change since D4.1 [7].

[13] https://production.eng.it/gitlab

4.4 Identity and Access Management

As provided in the deliverable D4.1 [7] the Identity and Access Management (IAM) module is a key component of the ICOS architecture, responsible for ensuring that authorised individuals have appropriate access to system resources. In the context of ICOS, these individuals are either users who interact with the ICOS MetaOS (including Application Integrators and Infrastructure Providers, as defined in D2.2 [2]) or other services (part of ICOS or external). The resources to be protected are the ICOS services, which include components and modules that provide system functionalities, such as the Job Manager, Telemetry, and Intelligence Layer.

4.4.1 Functional description

The IAM module is part of the Security Layer providing its functionalities to ICOS services at all levels of the continuum (Controller, Agent and Edge). It has three keys' functions (as provided in the deliverable D4.1 [7]):

- Management of Identities permissions.
- Authentication of users.
- Authorisation of requests.

In the IT-2, the implementation focused on:

- Implementing the authorisation mechanism and providing guidelines for developers to integrate it into their ICOS services.
- Automating the deployment and management of OAuth2 clients, permissions, and users in a declarative manner (including a script for creating users and clients).

With the authentication and authorisation functionalities, IAM contributes to meeting the following ICOS functional requirements from the D2.3 [1] (see Table 13 in Annex III.A).

4.4.2 Technical description

The IAM system uses Keycloak, an open-source IAM software[14]. This solution supports OIDC [32] and OAuth 2.0 protocols [33] and is released under the Apache 2.0 license. The ICOS IAM API is also based on the Keycloak API.

In IT-2, a script has been developed to automate the creation and management of Keycloak environments. This script simplifies provisioning by reading configurations from two YAML files (Figure 51 and Figure 52), enabling seamless setup and updates of Keycloak realms, users, roles, controllers, and agents.

The entire process is implemented in Python (v. 3.12.6). It starts with a YAML input file and outputs a YAML file at the end. The Keycloak Admin API is used for server interaction, while a YAML parser handles the configuration.

Here are more detailed steps, as outlined in the workflow shown in Figure 24:

1. Reads configuration from both static (`static_values.yaml`) and dynamic (`dynamic_values.yaml`) files.
2. Connects to Keycloak and performs the following operations:
   a. Creates realms, users, roles, controllers, and agents.
   b. Updates Keycloak to reflect changes from `dynamic_values.yaml`.
   c. Removes deleted configurations from Keycloak.
3. Generates updated YAML files after processing.

[14] https://www.keycloak.org/

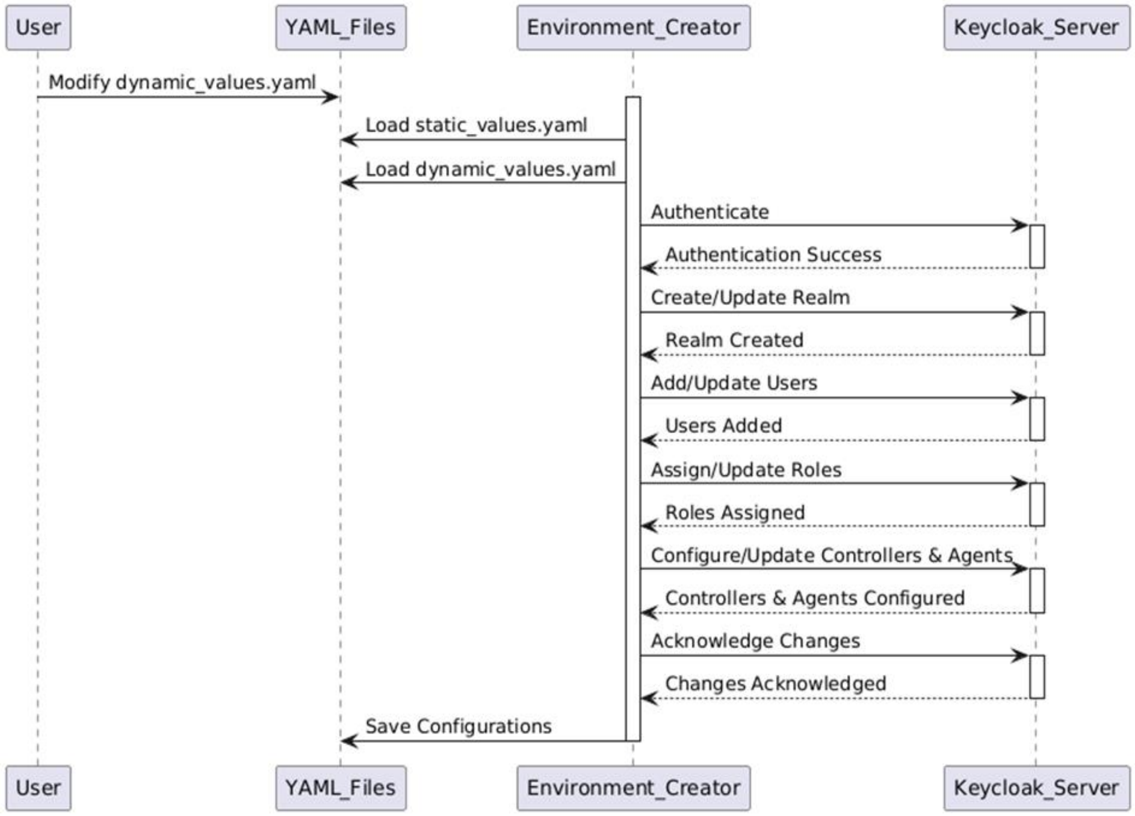


Figure 24. Sequence diagram for the Keycloak automation script

Annex III.B shows examples of configuration files:

- *dynamic_values.yaml* (Figure 51): Contains modifiable configurations like server details, realm information, and user data.
- *static_values.yaml* (Figure 52): Contains stable configurations like realm roles, controller clients, and core clients.

4.5 Anomaly detection

This section provides the functional and technical description of the Anomaly detection in ICOS.

4.5.1 Functional description

The Anomaly detection module is implemented using an AI-based log monitoring solution - LOMOS.

Traditional log monitoring solutions are limited to rule-based (manual) analysis of time series data. In contrast, LOMOS makes use of state-of-the-art Natural Language Processing (NLP) architectures to model log streams and capture their normal operating conditions. Concretely, it uses deep learning methods, such as Mask Language Modelling, common in self-supervised NLP, and Hypersphere Volume Minimisation, mapping normal samples to their representations. With these, LOMOS calculates anomaly scores for sequences of log templates, using NLP models for the IDs linked to these templates. Without manual pre-processing of raw logs from unstructured data, LOMOS aims to identify patterns in logs and identify anomalous behaviour.

Thus, LOMOS monitoring system does not depend on any manually defined rules or human intervention but relies on that behavioural model to automatically detect deviations that would represent any kind of abnormal situations, including potential security threats.

The anomaly detection module contributes to meeting the following ICOS functional requirements from the D2.3 [1] (see Table 14 in Annex III.A).

4.5.2 Technical description

The core of the system is LOMOS, which comprises the LOMOS Controller, LOMOS Parsing, and LOMOS Anomaly Detection modules developed in Python. See Figure 25.

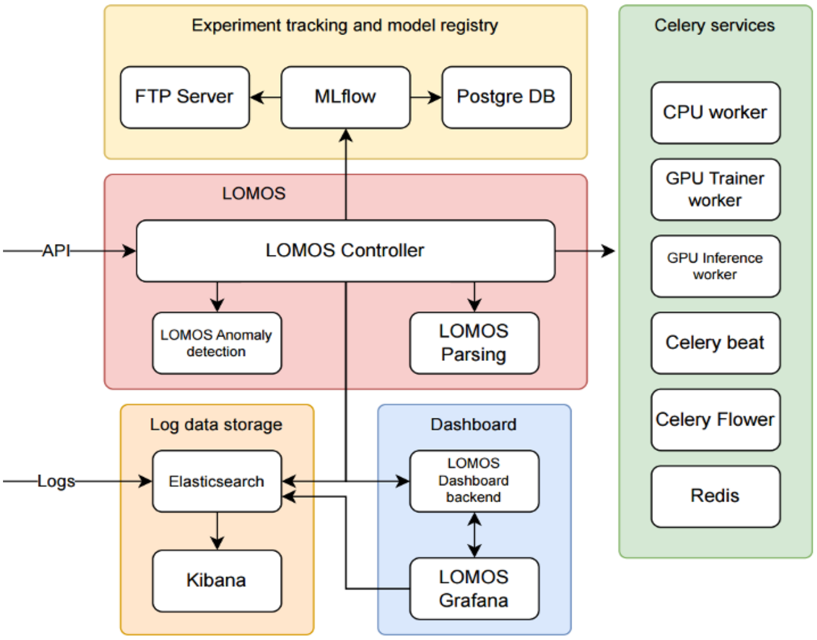


Figure 25. LOMOS architecture

The architecture of LOMOS can be described as follows:

- The **LOMOS Controller** is the central component that coordinates the system's operations. It can initiate log parsing and anomaly detection tasks on Celery CPU and GPU workers. The LOMOS Controller provides a REST API through which it receives requests for training or inference tasks. It is also used to monitor and manage periodic tasks.
- The **Celery services** are another crucial component of the system. Celery is employed to distribute learning and inference tasks among CPU and GPU workers. It encompasses:
  - a web dashboard named Celery Flower,
  - a scheduler called Celery Beat,
  - and Redis, which facilitates communication between the components. Both Celery and Redis are free and open source.
- The third component is focused on **experiment tracking and model registry**, comprising three free and open source services:
  - Postgres DB for storing all tabular data,
  - a File Transfer Protocol (FTP) server for storing artifacts, and MLFlow as a dashboard and
  - Representational State Transfer (REST) API access point.

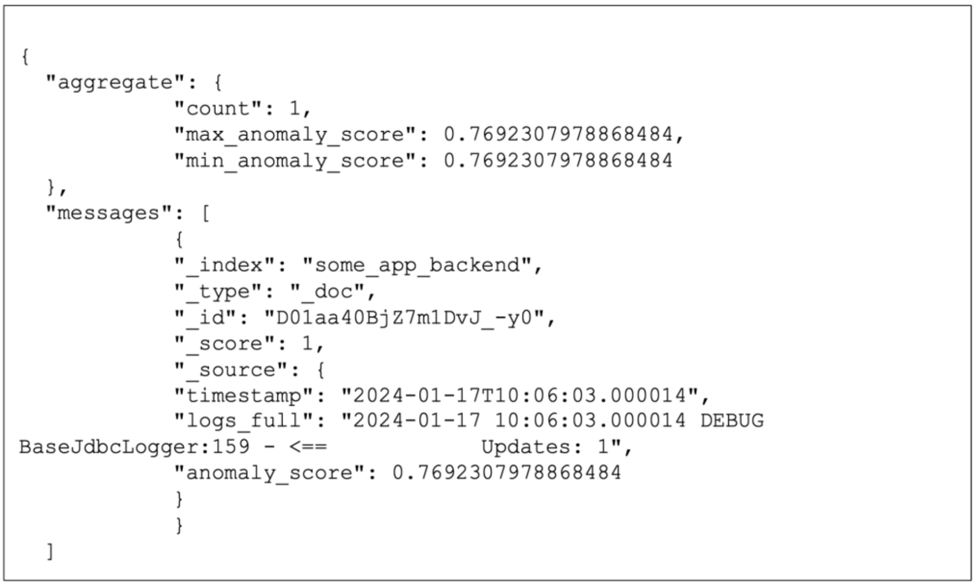
On top of this, MLFlow [22] provides an official Python API, allowing developers to easily integrate it into systems. It is used to log training parameters and metrics such as loss and other evaluation metrics.

Artifacts like weights of the trained model or tabular and visual outputs can also be stored. All values and artifacts can be retrieved using the experiment's unique identifier.

Logs are stored in the Log data storage section, within Elasticsearch indices.

- Kibana [42] is deployed for managing and exploring Elasticsearch.
- Elasticsearch [43] is well-suited for handling log data, as it can store documents in JavaScript Object Notation (JSON) format. Elasticsearch supports automatic index rollover based on time or index size.

The data can be moved through different phases and later stored in a format optimised for low storage requirements, rather than for query speed. Old data can also be automatically deleted to make space for new logs. There is an official Python client for Elasticsearch. The original Elastic is no longer free for commercial use, so we use the alternative OpenSearch, which is licensed under Apache 2.0 and is therefore free and open source.



```
{
  "aggregate": {
            "count": 1,
            "max_anomaly_score": 0.7692307978868484,
            "min_anomaly_score": 0.7692307978868484
  },
  "messages": [
            {
            "_index": "some_app_backend",
            "_type": "_doc",
            "_id": "D01aa40BjZ7m1DvJ_-y0",
            "_score": 1,
            "_source": {
            "timestamp": "2024-01-17T10:06:03.000014",
            "logs_full": "2024-01-17 10:06:03.000014 DEBUG
BaseJdbcLogger:159 - <==           Updates: 1",
            "anomaly_score": 0.7692307978868484
            }
            }
  ]
```

Figure 26. Sample LOMOS anomaly score

The final part is Dashboard. The actual dashboard is developed in Grafana [23], which is a free and opensource analytics and interactive visualisation web application. By default, it does not support input forms. During ICOS, the LOMOS Dashboard backend was extended it and developed in Python. Now LOMOS supports configuration forms in the dashboard. This enables users to configure and run the training and inference processes through a web dashboard. The LOMOS dashboard backend passes the requests to the LOMOS controller which executes the jobs.

- The LOMOS API-2 – it is named “2” because LOMOS Controller already has an API for receiving requests for training and inference – is also a REST API, written in Python, and it is containerised and integrated into the ICOS Controller Suite. It enables communication between LOMOS, located outside ICOS, and ICOS.
- LOMOS API-2 exposes the anomaly scores to the Intelligence Coordination API to be queried by Meta-kernel services (e.g. Telemetry). Higher score means a bigger presence of the anomalous events in the logs. The anomaly score also includes the timestamp and a sample log line with the anomaly.

Figure 26 showed an example of the anomaly score in JSON format.

4.6 Audit

This section provides the functional and technical description of the Audit functionality in ICOS.

4.6.1 Functional description

The Audit module in the scope of ICOS aims to provide a robust security auditing solution that integrates Trivy for security scanning, Tetragon for runtime behaviour monitoring, and finally a Prometheus exporter for auditing metrics collection.

In containerised environments, a significant portion of vulnerabilities arises from misconfigurations, particularly when containers are granted more privileges than necessary. In some cases, the application containers may have extended privileges that are essential for their execution. These overly permissive configurations can expose both the containerised application and the underlying host system to serious security risks.

Trivy provides a first layer of auditing on a given application by analysing application's manifest and its container images. This analysis covers security aspects that relate on what privileges are granted by the current configuration and how much they comply with best security practices for containerised environments by providing a severity score [LOW, MEDIUM, HIGH].

Tetragon is able to perform real time monitoring on events emitted by the kernel using eBPF (Extended Berkeley Packet Filter) [37] and produce audit logs on those events. Since kernel events are emitted in high volumes, Tetragon is using Trivy's analysis as a baseline in order to produce real time audit logs that focus on the security aspects identified by Trivy.

Prometheus exporter is forwarding the produced audit logs to the Telemetry Controller to be used by the Policy manager to monitor compliance of the running application and to notify Job manager if the policy violation occurred and remediation action is needed (see Figure 27).

With Trivy performing security scans and Tetragon producing audit logs at runtime, Audit module contributes to meeting the following ICOS functional requirements from the D2.3 (see Table 15 in Annex III.A).

4.6.2 Technical description

A containerised application can have multiple containers, and each container can have its own configuration coming from a Kubernetes manifest and in addition each container could run on different Kubernetes nodes. The deployment of an application as a whole on different edge infrastructure instances poses a challenge on how to collect and associate auditing logs with each respective application since each container will produce its own auditing metrics. Another challenge is the large volume of the events emitted by the kernel and the distinction on which contain meaningful security insights. In order to address those challenges the audit component combines both Trivy and Tetragon capabilities as described below.

Tetragon overcomes the challenge of associating which event is emitted by which application since it supports Kubernetes awareness on workload identities such as pods, namespaces and containers. On the other hand, it is also an observability tool that produces a high volume of events and a filtering technique on those events is needed in order to reduce the overall overhead of data collection of irrelevant events.

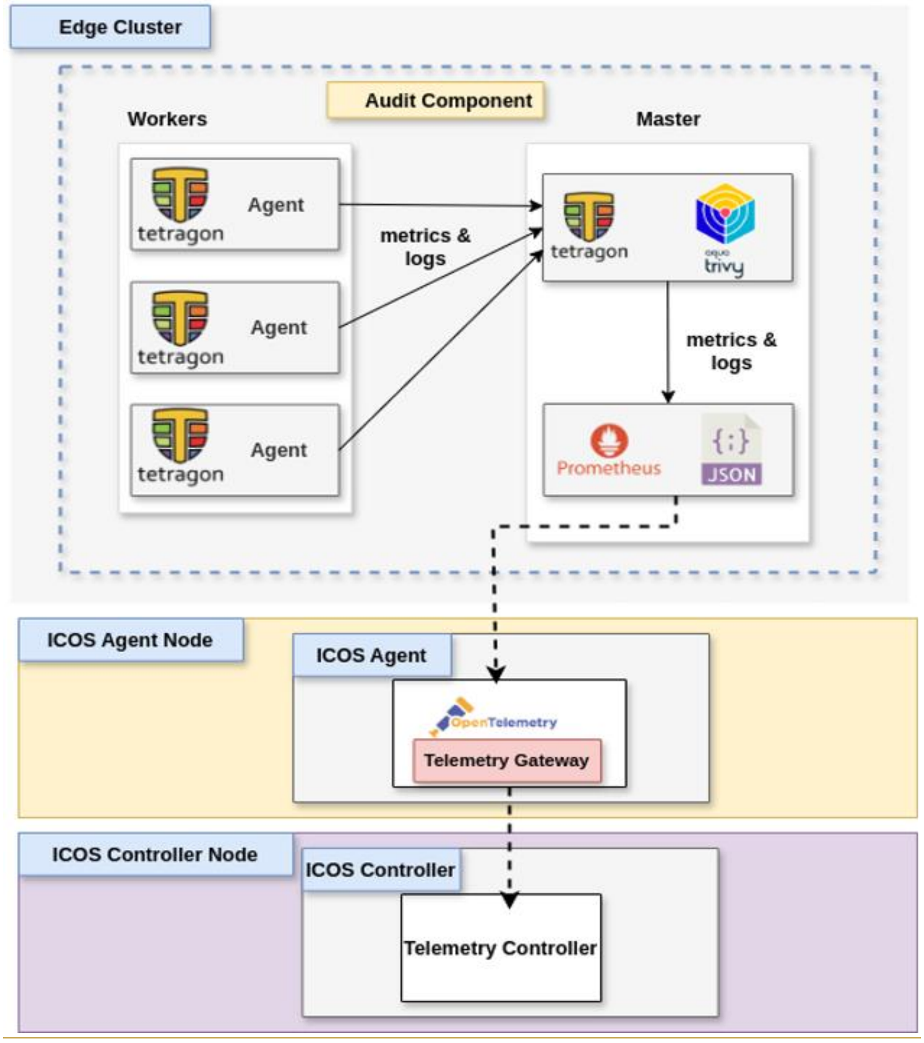


Figure 27. Audit module architecture

This filtering mechanism is achieved by applying pre-defined monitoring policies using Tetragon's CRDs (Custom Resource Definitions) which target the following aspects of a containerised application:

- **Filesystem**: access and modifications on critical files (for instance files in /etc and /tmp folders should be monitored)
- **Privileges escalation**: processes on runtime, elevate their privileges in order to perform their tasks.
- **Linux capability usage**: Linux capabilities are a set of fine-grained access control mechanisms that divide the traditional root privileges into distinct units, allowing processes to execute specific privileged operations without requiring full root access.
- **Networking**: Connections outside the cluster should be monitored.
- **Binary execution**: Malicious actors often use core binaries that are shipped by default in most Linux distributions such as git, wget, netcat etc. The execution of those binaries should be monitored or blocked during execution.

The Audit component automatically installs all Tetragon dependencies alongside the pre-defined observability policies. The pre-defined observability policies are mapped with the first layer auditing analysis by Trivy which target the same security context. A set of Trivy audit rules and how they map on Tetragon observability policies is presented in Table 2.

Table 2. Sample of Tetragon observation policies

Requirement ID	Requirement Name	Description
KSV007	/etc/hosts aliases unmanaged	Filesystem
KSV009	network namespace permits processes in the pod to communicate with processes bound to the host's loopback adapter	Networking
KSV012	run as a non-root user to ensure least privileges	Privilege Escalation
KSV014	Root file system is not read-only	Filesystem, Privilege Escalation
KSV020	run with user ID > 10000 to avoid conflicts with the host's user table	Privilege Escalation, Linux Capability Usage
KSV021	run with group ID > 10000 to avoid conflicts with the host's user table	Privilege Escalation, Linux Capability Usage
KSV023	Host path is mounted	Filesystem, Privilege Escalation
KSV117	The ports which are lower than 1024 receive and transmit various sensitive and privileged data	Networking

The binary execution policy produces audit logs based on the image scanning report which reports which Linux binaries are potential tools for a malicious action and should be blocked in order to reduce the attack surface.

4.7 Trust

This section provides the functional and technical description of the Trust functionality in ICOS.

4.7.1 Functional description

As presented in D2.4 [2], Trust is implemented as an architecture-wide functionality in ICOS, present at three levels:

- In identification and encrypted communication between ICOS components located inside the Controller and ICOS components located inside the Agent using CNI (Container Network Interface) with VPN encryption technologies (e.g., Wireguard).
- In identification and encrypted communication between ICOS components in the ICOS Controller and in the ICOS Agent using certificates and (m)TLS.
- In identification and encrypted communication between ICOS users and ICOS components using VPN technologies (e.g., Wireguard).

For trust and encrypted communication between ICOS components located in ICOS Controller and ICOS agent, we use self-signed certificate authority (CA). We decided against using public and widely used CA like Let's encrypt [38] as it would be necessary to expose ICOS components to the internet and rely on the external service. By using self-signed CA, we have complete control and management over the trust chain as well as ability to isolate ICOS components.

For self-signed CA, we use step-cat[15] that allows for issuing X.509 certificates for use with TLS and mutual TLS (mTLS) and plays well with Kubernetes as the main containerisation technology in ICOS.

For encrypted communication between ICOS components in the ICOS Controller and in the ICOS Agent we use Cilium, an open source, cloud native solution for securing network connectivity. It enables the encryption of all node-to-node traffic with just one switch, no application changes or additional proxies, features automatic key rotation with overlapping keys, efficient data path encryption through in-kernel IPsec or WireGuard, and can encrypt all traffic, including non-standard traffic like UDP [44]. For the ICOS Controller and ICOS Agent, Cilium is configured as the Container Network Interface (CNI) in Kubernetes cluster and uses Wireguard as an encryption protocol [39].

Wireguard is also used as the VPN technology for secure and encrypted access for ICOS users to the ICOS system.

The implementation of Trust functionality contributes to meeting the following ICOS functional requirements from the D2.3 [3] (see Table 16 in Annex III.A).

4.7.2 Technical description

The **step-ca** is distributed as a part of the ICOS Core suite. It is composed of the following components:

1. **Step Certificates**: An online certificate authority and related tools for secure automated certificate management, so it is possible to use TLS everywhere.
2. **Autocert**: A Kubernetes add-on that automatically injects TLS/HTTPS certificates into containers.
3. **Step Issuer**: A certificate issuer for cert-manager using Step Certificates.

The architecture of step-ca in ICOS is presented in Figure 28. Certificate authority is a service for managing certificates and is located in the Kubernetes cluster. It communicates with the Kubernetes-compatible Cert manager[16] that is responsible for distributing certificates to ICOS modules and components in the Kubernetes cluster using Nginx Ingress Controller.

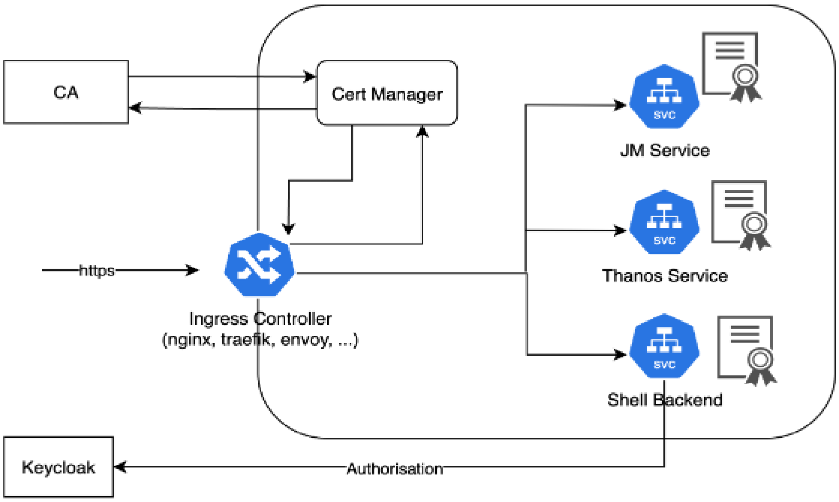


Figure 28. Self-signed CA architecture

Cilium and Wireguard

Cilium CNI operates by deploying agents on each Kubernetes node that program eBPF to handle pod networking. The agents create and manage virtual network interfaces for pods and handle routing

[15] https://github.com/smallstep/certificates
[16] https://github.com/cert-manager/cert-manager

between nodes. In the figure provided below (see Figure 29) the Cilium CNI networking interfaces are depicted as a single interface (cni iface) in order to emphasise the traffic relay to the Wireguard tunnel where the encryption takes place.

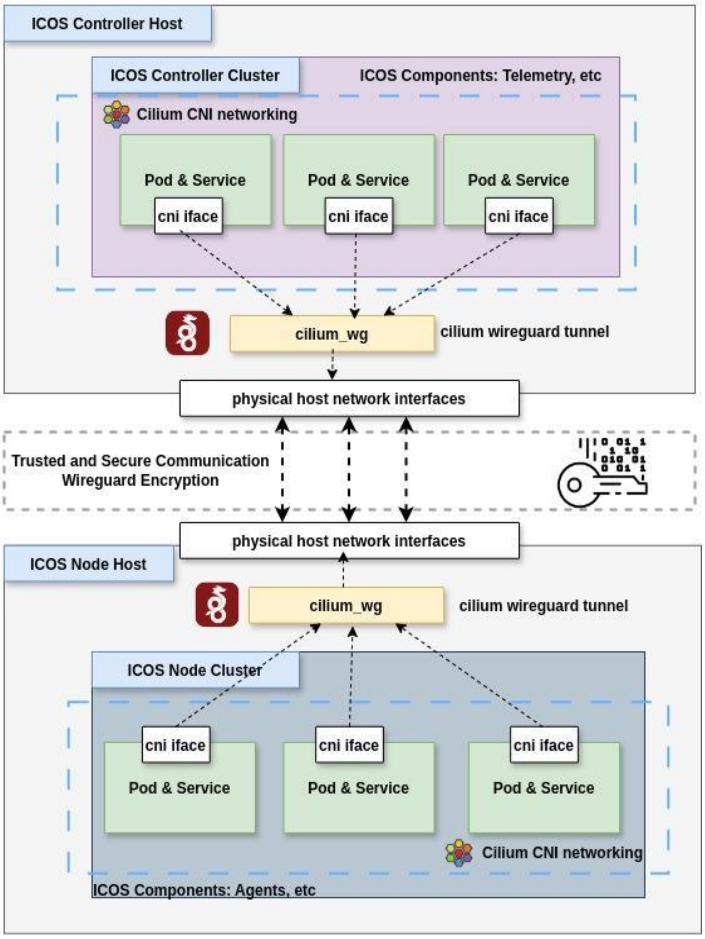


Figure 29. Cilium-Wireguard Encrypted Networking

Cni iface: A set of interfaces that Cillium creates and manages as it operates as a cni:

- cilium_net_bridge: Main interface programmed with *eBPF* for pod networking which handles pod-to-pod traffic within the node.
- cilum_vethX: For each pod, cilium creates this interface in order to pair it with its dedicated container interface (usually *eth0*). With this pairing the connection with *cilium_net_bridge* interface is achieved.

Upon installation with Wireguard encryption enabled, Cilium creates a WireGuard interface (cilium_wg0) on each node. As pods communicate across nodes, their traffic is encapsulated and encrypted through these WireGuard tunnels. Each node maintains encrypted tunnels to other nodes where pod communication is required.

The overall traffic exchange and encryption flow is:

1. Cilium agent detects cross-node traffic
2. Traffic is redirected to the local WireGuard interface
3. WireGuard encrypts the packets using public-key cryptography
4. Encrypted packets are sent to the destination node
5. Destination node's WireGuard interface decrypts the traffic
6. Cilium routes decrypted packets to the target pod

4.8 Limitations and future work

In the development of the Security Layer, we encountered certain limitations that impacted the design of the Layer's modules and components:

1. **Security scan:**
   a) To provide full scanning functionality, Wazuh agents need to be manually installed on infrastructure where ICOS is running, and this has to be done in the ICOS on-boarding phase. Detailed documentation for this step can be found in the next link: https://www.icos-project.eu/docs/Administration/ICOS%20Edge/wazuh_agent/
   b) The Wazuh server, however, is containerised and deployed as part of the ICOS Controller Suite.
2. **Anomaly detection**: LOMOS is deployed outside of ICOS due to hardware requirements (GPU) and an additional API was developed (LOMOS API-2) for communication with ICOS (Intelligence AI Coordination backend).

In terms of future work, there is still ongoing work about providing **intelligence functionality to the Security Layer**, with a focus on robust security models based on Tetragon logs for detection of security issues (anomalies) in ICOS and application containers as presented in D2.4, section 4.11.3.

- The first step involves leveraging unsupervised learning techniques to label Tetragon logs. This approach identifies patterns, clusters, and potential anomalies in the logs without prior labelled data, providing a foundational understanding of typical and atypical log behaviours.
- Tetragon's core functionality is monitoring and tracking process execution lifecycles through eBPF, which means the log patterns are relatively static and structured. This static nature of kernel logs adds additional challenges of distinguishing between benign and malicious patterns. In order to create a concrete solution that leverages Tetragon logs for anomaly detection, a comprehensive methodology must address several critical aspects:
  - **The Static Nature of Process Execution:** Process execution patterns captured by Tetragon follow highly predictable sequences that contain most variances from legitimate actions by the system. The static aspect of process execution log data makes the task of detecting anomalies significantly difficult, as the execution traces of system calls and interactions between processes often follow well-defined flows, and as a result, more subtle malicious activity can sometimes be hidden. As normal operations are so predictable, it can be challenging to identify signs of true anomalous behavior.
  - **Monitoring Policy Design:** The development of effective Tetragon monitoring policies requires the selection and monitoring of specific system calls, while ensuring that the policies capture relevant process interactions without overwhelming system resources.
  - **Generation of Anomalous Patterns:** The creation of effective training datasets for Tetragon-based anomaly detection requires deliberate engineering of malicious patterns, where anomalous actions are specifically designed to trigger the pre-defined monitoring policies. This approach involves simulating attack scenarios or anomalous patterns in order to generate a range of training examples, while observing how those patterns reflect on the generated logs.
- Once the logs were successfully labelled, a supervised ML model, such as LogBERT, log-inspector, or other anomaly detection frameworks, can be trained to predict and identify anomalies based on the labelled data. These models specialise in analysing log data, detecting unusual behaviours, and classifying events that could indicate potential security threats. Finally, the system would use the predictions from the trained model to generate alerts, providing early warnings about vulnerabilities or suspicious activity, thus enabling proactive system defence.

This integration and automatic flow of relevant Tetragon logs into Telemetry or data management components is currently ongoing, and an unsuccessful completion could jeopardise the impact of any AI models deployed. The final status of this piece of work will be reviewed and updated accordingly as part of D5.3 – "*Third ICOS Release: Complete ICOS version*" *[8]*.

5 Conclusions

This document has covered the functional and technical aspects of the second version (IT-2) of Data Management, Intelligence Layer and Security Layer software. This document has used D4.1 as the input and first version of the Data Management, Intelligence Layer and Security Layer software (IT-2) and the second version of the architecture (D2.4) and provided updates where applicable.

- The **Data Management** component architecture was initially described in D4.1 as a set of software blocks, that allows ICOS components to access data regardless of its location in the continuum, aiming at optimising data access by executing part of the code within the data store and therefore minimising data movements within and across nodes. Building upon this foundation, IT-2 enhances software architecture and integration with other ICOS components. The Data Management component fulfils critical data requirements, encompassing transport, mutable data structures, and computationally intensive operations. Employing Zenoh for communication and dataClay for distributed storage facilitates functionalities such as the ICOS data bus, training offloading, telemetry processing, and policy storage.
- The **Intelligence Layer** in IT-2 includes the four next modules: AI coordination, data processing, Trustworthy AI, and AI Analytics. It also introduces the AI Model and Data repository, which will be delivered in D4.3 (M36). Built from the base provided in D4.1, the Intelligence Layer has been extended as follows. First, as part of technical discussions held in task forces aiming to provide an update on ICOS Intelligence towards D2.4, the AI coordination Layer has been extended to include a frontend to interact with the meta-OS. This new frontend component queries telemetry data to determine when predictions are required for system usage or energy-related measurements. It then sends requests to the AI coordination backend (detailed in IT-1) to train a new model or generate predictions. The Intelligence Layer modules have been significantly enhanced, including (i) support for multivariate model training, enabling the concurrent forecasting of numerous system metrics like CPU and memory usage; (ii) integration of MLOps tools, streamlining performance analysis via training and validation curves, and improving model interpretability for greater transparency; (iii) For AI trustworthiness, confidence levels and ranges have been incorporated to indicate prediction accuracy. Furthermore, model monitoring improves robustness, and federated learning allows for privacy-preserving model development while minimising data transfers; (iv) for efficient AI at the edge, model size reduction techniques, such as quantisation and knowledge distillation, have been employed and added to the backend of ICOS Intelligence.
- The **Security Layer** in IT-2 leverages the base from IT-1 (D4.1) and enhances identity management, anomaly detection, auditing, and trust mechanisms. The Security Layer Coordination module enables seamless integration between security components and other ICOS modules. The Identity and Access Management (IAM) module provides secure authentication and authorisation across ICOS. The Security Scan module actively monitors deployed ICOS infrastructure for vulnerabilities and compliance deviations, while the Anomaly Detection module employs AI to identify potential security threats. Furthermore, the Audit module improves system transparency, and the Trust functionality assures the identity of ICOS components and enforces encrypted, verifiable communication among them. All these modules are currently part of ICOS and will finalise their integration in the ICOS Final release (task T5.3 and deliverable D5.3 [8]).

Table 3 presents which parts of Data Management, Intelligence Layer and Security Layer IT-2 can be considered as ICOS assets as an update over D4.1.

ICOS modules and components in this document "Data Management, Intelligence and Security Layers (IT-2)" finalise their development planned period in M30. Integration work will still be performed among them towards the ICOS final release. For this reason, this document will also be used as input to D5.3 – "*Third ICOS Release: Complete ICOS version*" [8], that will expand on integration pieces involving components covered in this deliverable.

The online component which will continue its development process in this work package will be the "AI Model and Data Repository", presented in Section 3.6, which is planned to be delivered in D4.3 – "*ICOS Dataset and AI Models Marketplace*" (M36).

Table 3. Data Management, Intelligence and Security Layers IT-2 parts as ICOS assets

ICOS Modules	ICOS ASSETS
Data Management	1. Data Bus which allows horizontal communication across computing continuum thanks to the Zenoh routers. It is used by the Topology Exporter but its features can be leveraged by other applications. 2. dataClay backend enables advanced features such as the task offloading (used by the Intelligence Layer for training workloads) or the Policy storage backend. It is a data management solution distributed across the ICOS infrastructure. 3. dataClay OTLP Bridge leverages the dataClay backends for reading processing and storing real-time data metrics close to its source and using them from federated learning workloads. ▸ The Data Management assets that were already presented in D4.1: Metadata Service, Storage backends, the REST API, and a storage manager plugin.
Intelligence Layer	1. Intelligence Layer Coordination frontend (Export Metrics API): This element enables real-time and predictive monitoring by collecting telemetry data from ICOS nodes and integrating it with AI models in the Intelligence Layer. It supports dynamic and static metric creation, allowing both on-demand telemetry-based metric generation and automated discovery of new nodes for resource monitoring (CPU, Memory, and Energy consumption). The API also provides lifecycle management for metrics, ensuring efficient resource allocation and accurate tracking across the ICOS infrastructure**.** 2. Intelligence Layer Coordination backend (Intelligence Layer Coordination API in D4.1 [7]): This API supports async calls to model building and predictions, has its own model registry, and allows the reuse of previously trained models in an ICOS controller. This has been enhanced to communicate with extra APIs such as Export Metrics API and LOMOS API 2, and updated modules as covered in this deliverable. 3. Trustworthy AI module: This module contains methods for model monitoring, explainability and confidence scores that have been inbuilt into the AI Analytics module predictions. 4. Model compression: Several mechanisms, as described in this deliverable, have been inbuilt in the AI Analytics module to improve AI model efficiency and reduce its footprint at the edge. 5. ICOS-FL Module (Server and Clients) enables decentralised training operations among different ICOS nodes, while also allowing for knowledge sharing between local learning models. This module offers an extra capability for ICOS administrators to initiate and execute FL training rounds without sharing the local data of ICOS nodes. Once FL training is done, the aggregated model (across ICOS nodes) is stored in the ICOS AI Models Repository for future usage. 6. AI Models and Data Repository: This online repository, which integrates with the Intelligence Layer, is one of the ICOS KERs. Planned for D4.3 – "ICOS

ICOS Modules	ICOS ASSETS
	Dataset and AI Models Marketplace," it intends to allow ICOS to leverage an online catalogue of AI models and datasets. ▸ The Intelligence Layers models that were already presented in D4.1 for intelligent energy-aware task offloading, load time series forecasting, forecasting upcoming load anomalies, and CPU utilisation forecasting to improve workload distribution model.
Security Layer	1. Anomaly detection module analyses ICOS services or user applications logs to detect any kind of abnormal situation that could be a potential security threat. 2. Audit module provides real-time behaviour monitoring of Kubernetes containers to identify events that pose a security threat (breach security compliance policy). 3. Trust provides identification and secure (encrypted communication) between ICOS components in the ICOS Controller and in the ICOS Agent, between ICOS Controller and the ICOS Agent and between ICOS users and ICOS system. ▸ The Security Layers assets that were already presented in D4.1: Security Layer Coordination module API, Identity and Access Management module, Security Scan module.

6 References

[1] *Towards a functional continuum operating system (ICOS),* Grant Agreement N°101092562, Horizon EU.

[2] ICOS. *D2.2 - ICOS ecosystem: Technologies, requirements, and state of the art (IT12)*. Francesco D'Andria, 2023.

[3] ICOS. *D2.3 - ICOS ecosystem: Technologies, requirements, and state of the art (IT-2)*. Georgios Xylouris. 2024

[4] ICOS. *D3.2 - Meta-Kernel Layer Module Developed (IT-2)*. Francesc Lordan. 2024

[5] ICOS. *D3.3 - Meta-Kernel Layer Module Integrated (IT-2).* Kalman Meth. 2025

[6] ICOS. *D2.4 - ICOS architectural design (IT-2).* Jordi García. 2024

[7] ICOS. *D4.1 - Data management, Intelligence and Security Layers (IT-1).* Hrvoje Ratkajec. 2023

[8] ICOS. *D5.3 - "Third ICOS Release: Complete ICOS version*". 2025

[9] *Open Call infrastructure On-Boarding v1.3 document,* NCSRD (internal), 2024

[10] Giannopoulos, A., Spantideas, S., Kapsalis, N., Karkazis, P., & Trakadas, P. (2021). Deep reinforcement learning for energy-efficient multi-channel transmissions in 5G cognitive hetnets: Centralized, decentralized and transfer learning based solutions. *IEEE Access, 9, 129358-129374.*

[11] Giannopoulos, A., Paralikas, I., Spantideas, S., & Trakadas, P. (2024). HOODIE: Hybrid Computation Offloading via Distributed Deep Reinforcement Learning in Delay-Aware Cloud-Edge Continuum. IEEE Open Journal of the Communications Society.

[12] Goodwin, P. and Lawton, R. (1999). On the asymmetry of the symmetric MAPE. *International journal of forecasting, 15(4), 405-408.*

[13] Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., ... & Gebru, T. (2019, January). Model cards for model reporting. In Proceedings of the conference on fairness, accountability, and transparency (pp. 220-229).

[14] Liang, W., Rajani, N., Yang, X., Ozoani, E., Wu, E., Chen, Y., ... & Zou, J. (2024). What's documented in AI? Systematic Analysis of 32K AI Model Cards. arXiv preprint arXiv:2402.05160.

[15] Yang, X., Liang, W., & Zou, J. (2024). Navigating Dataset Documentations in AI: A Large-Scale Analysis of Dataset Cards on Hugging Face. arXiv preprint arXiv:2401.13822.

[16] Polino, A., Pascanu, R., & Alistarh, D. (2018). Model compression via distillation and quantization. arXiv preprint arXiv:1802.05668.

[17] Chen, L., Bai, Y., Huang, S., Lu, Y., Wen, B., Yuille, A. L., & Zhou, Z. (2022). Making your first choice: To address cold start problem in vision active learning.

[18] Suárez-Cetrulo, A. L., Quintana, D., & Cervantes, A. (2023). A survey on machine learning for recurring concept drifting data streams. Expert Systems with Applications, 213, 118934.

[19] Zheng, X., Li, P., Hu, X., & Yu, K. (2021). Semi-supervised classification on data streams with recurring concept drift and concept evolution. Knowledge-Based Systems, 215, 106749.

[20] Zenoh, https://zenoh.io/, retrieved 2023-09-20.

[21] dataClay, https://dataclay.bsc.es/, retrieved 2024-12-20.

[22] MLFlow, https://mlflow.org/, retrieved 2024-12-20.

[23] Grafana, https://grafana.com/, retrieved 2024-12-20.

[24] Prometheus, https://prometheus.io/, retrieved 2024-12-20.

[25] BentoML, https://docs.bentoml.org/en/latest/index.html, retrieved 2024-12-20.

[26] Pandas, https://pandas.pydata.org/, retrieved 2024-12-20.

[27] Scikit-learn, https://scikit-learn.org/stable/index.html, retrieved 2024-12-20.

[28] PyTorch, https://pytorch.org/, retrieved 2024-12-20.

[29] TensorFlow, https://www.tensorflow.org/, retrieved 2024-12-20.

[30] SHAP, https://github.com/shap/shap, retrieved 2024-12-20.

[31] Scaphandre, https://github.com/hubblo-org/scaphandre, retrieved 2024-12-20.

[32] OpenID Connect protocol, https://openid.net/, retrieved 2024-12-12.

[33] OAuth 2.0 protocol, https://oauth.net/2/, retrieved 2024-12-12.

[34] JSON Web Token, https://jwt.io/, retrieved 2024-12-12.

[35] Trivy, https://github.com/aquasecurity/trivy, retrieved 2024-12-12.

[36] Tetragon, https://tetragon.io/, retrieved 2024-12-12.

[37] eBPF, https://ebpf.io/, retrieved 2024-12-12.

[38] Lets encrypt, https://letsencrypt.org/, retrieved 2024-12-12.

[39] Wireguard, https://www.wireguard.com/, retrieved 2024-12-12.

[40] Keycloak, https://www.keycloak.org/, retrieved 2024-12-12.

[41] Wazuh, https://wazuh.com/, retrieved 2024-12-12.

[42] Kibana, https://www.elastic.co/kibana/, retrieved 2024-12-25.

[43] Elastic Search, https://www.elastic.co/, retrieved 2024-12-25.

[44] Cilium, https://cilium.io/, retrieved 2024-12-12.

Annexes

This section contains annexes to this deliverable.

Annex I: Data management

I.A Requirements

Table 4. Requirements starting to be met with the IT-2 of the Data Management Component [1]

Requirement ID	Requirement Name	Description
CC_FR_03	Topology awareness	ICOS should be able to monitor and maintain the topology of the created Cloud Continuum.
CM_FR_02	Workload offloading	ICOS MUST be able to distribute the workload of the application components offloading part of their computation onto other nodes of the infrastructure, and coordinate the offloaded components
CM_FR_10 [D4.1]	Data management	ICOS must be able to maintain the data sources topology as well as data source types (metadata) for proper application data assignment.
CM_FR_11 [D4.1]	Data access	ICOS must be able to provide different data access methods, including selective access as well as streaming access, according to flexible high level data access application program interfaces
CM_FR_15	ML Scalability and maintenance	ICOS SHOULD perform model training detached from ICOS nodes. By training the models on a separate platform with more computing power, it is possible to produce models with higher accuracy and to reduce the time required to train them. Once the models are trained, they can be deployed on ICOS nodes to make predictions and carry out specific tasks.
DRT_FR_01 [D4.1]	Data distribution across the continuum	ICOS SHOULD take advantage of all available devices to place data. Flexibility to adapt to different configurations (cloud-only, edge-only, cloud-edge)
DRT_FR_02 [D4.1]	Transparent data access	ICOS must be able to provide location and format transparent data access methods through flexible high level data access application program interfaces (API)
DRT_FR_03 [D4.1]	Minimisation of data transfers	ICOS MUST avoid unnecessary data movements to increase performance, reduce network congestion, and favour privacy by exploiting near-data processing
DRT_FR_04 [D4.1]	Support for distributed/parallel execution	ICOS SHOULD provide the integration of data management with the execution runtime to support efficient scheduling and execution of the required tasks.

Requirement ID	Requirement Name	Description
DRT_FR_05 [D4.1]	Failure recovery mechanism/management	ICOS MUST provide the capability to restart the failing transfers of the data in case of the failures (e.g., losing the connectivity)
DRT_FR_06 [D4.1]	Secure data exchange	ICOS SHOULD support data preserving communication techniques between modules (interfaces) for secure data exchange.
DRT_FR_07 [D4.1]	Data recompilation	Control Nodes must be able to recollect data in a scheduled or periodic way from the nodes it orchestrates

I.B Zenoh Helm chart snippet

```
config:
  zenohJson: |-
    {
        "adminspace": {
            "enabled": true,
            "permissions": {
                "read": true,
                "write": true
            }
        },
        "connect": {
            "endpoints": [
                "tcp/10.160.3.20:7447"
            ]
        },
        "listen": {
            "endpoints": [
                "tcp/0.0.0.0:7447"
            ]
        },
        "metadata": {
            "name": "give-me-a-name",
        },
        "mode": "router",
        "plugins": {
        },
        "scouting": {
            "gossip": {
                "autoconnect": {
                    "peer": [],
                    "router": []
                },
                "enabled": true,
                "multihop": true
            },
            "multicast": {
                "enabled": true
            }
        }
    }
local:
  ports:
  - name: zenoh
    port: 7447
    protocol: TCP
    targetPort: 7447
  replicas: 1
  type: LoadBalancer
  zenohLocal:
    args:
    - -c
    - /etc/zenoh.json
    env:
      rustLog: z=debug
    image:
      repository: eclipse/zenoh
      tag: 1.0.0
```

Figure 30. Zenoh configuration Helm chart

Annex II: Intelligence Layer

II.A Requirements

Table 5. Requirements starting to be met with the IT-2 of the Intelligence Layer [1]

Requirement ID	Requirement Name	Description
CM_FR_08	Predictive Monitoring/SLOs violations	ICOS SHOULD predict potential future SLOs violations analysing monitoring metrics with ML techniques, and prepare to avoid/absorb such risk
CM_FR_14	Model Optimisation Techniques	ICOS SHALL provide different methodologies for the pruning calculation of different deep neural network backbones that provides efficient sparse connectivity for modeling high-performance models on edge nodes. Distillation techniques will be used to shorten training times using the high capability of large models, the ICOS system will offer various techniques for distilling various deep neural network backbones.
CM_FR_15 [D4.1]	ML Scalability and maintenance	ICOS SHOULD perform model training detached from ICOS nodes. By training the models on a separate platform with more computing power, it is possible to produce models with higher accuracy and to reduce the time required to train them. Once the models are trained, they can be deployed on ICOS nodes to make predictions and carry out specific tasks.
CM_FR_16	Federated Learning benchmarking	ICOS MUST be able to train models over open-source federated learning frameworks. Models trained on ICOS nodes (the edge or on the cloud) will be able to share knowledge when applicable to learn collaboratively.
CM_FR_17 [D4.1]	AI for resource management	ICOS MUST classify infrastructure that facilitates the resource management on the network, and implements various resource management algorithms for the continuum, as well as clustering the various resources in the ecosystem with the goal of optimising those available.
CM_FR_18 [D4.1]	MLOps frameworks	ICOS MUST allow the storage, retrieval and modification AI models available to be used by clients and AI-as-a-Service (AIaaS) providers from ICOS

Requirement ID	Requirement Name	Description
CM_FR_19	AI Marketplace	ICOS SHOULD use data science and MLOps frameworks to ensure reproducibility and proper documentation of each model built or experiment run.
CM_FR_20 [D4.1]	Continuous learning	The ICOS SHOULD support algorithms for continuous learning, adapting to drifts and shifts and avoiding catastrophic forgetting.
OP_FR_03 [D4.1]	Resources classification	ICOS MUST be able to catalogue the reliability, trustworthiness, confidence, of system resources, in order to take into consideration during the resources allocation process
OP_FR_06	Data Processing and ML modeling	ICOS MUST support dashboard/interface for data processing and ML modeling purposes. A concrete demonstration of the model training and inference performance will be provided via a dedicated GUI. Data processing, ML and FL platforms will be based on open-source tools to offer scalability and transparency of the deployed models.
SST_FR_04	Secure & robust model training	ICOS SHOULD provide the ability to train models in a federated learning fashion to ensure privacy and trust. Privacy can be guaranteed by training data on the edge. Robustness is assured by assessing the data against anomaly detection.
SST_FR_05	Providing trustable models	ICOS SHALL use models ethically unbiased. By using these models, ICOS strives to preserve trust in the outputs of the ICOS system and provide a fair and equitable experience for all.
SST_FR_09 [D4.1]	Anomaly detection	ICOS MUST provide a mechanism for the detection of anomalies (e.g., any kind of abnormal situations, including potential security threats, that is recorded in application, system, or network logs) in the applications/services on the cloud/network/edge provider

II.B Intelligence coordination – backend endpoints

The models deployed in the AI Analytics module entail using Swagger API or curl through the AI coordination backend service, as well as training and testing and testing pipelines may be fully parameterised using this API, which is a crucial component of the design. See the endpoints in Table 6.

Table 6. AI Analytics API endpoints

HTTP request (ICOS Shell)	description	ICOS component	component name	component port	component endpoint
POST /train	Trains a model as per parameters	Analytics module	API to Train a model	3000	POST `/core/analytics_train/`
POST /predict	Generates model predictions	Analytics module	API to get model predictions	3000	POST `/core/analytics_predict/`
POST /detect_drift	Detects drifts by monitoring the data	Analytics module	detect drifts	3000	POST `/core/analytics_drift_detection/`
POST /get_anomalies	Gets anomalies from LOMOS api	Analytics module	Gets anomalies – linked to LOMOS (see Section 4.5)	3000	POST `/core/analytics__get_anomalies/`
POST /launch_mlflow_ui	Start mlflow ui service	Analytics module	Start mlflow ui	5000	POST `/core/analytics_launch_mlflow_ui/`

The API endpoint '`get_anomalies`' obtains anomaly scores, presented as part of the Security Layer in this document, but reachable from the Intelligence API due to its AI function. The integration between Intelligence and Security Layers for this has been already documented in D2.4 [6], Sections 3.2.2 – "*Intelligence Layer*" and 4.11.1 – "*Anomaly detection*"

The AI support container will have all the API endpoints described in Table 6 except for an additional Jupyter lab API service to launch a Jupyter lab notebook session. See this below.

Table 7. AI Support container API endpoints

HTTP request (ICOS Shell)	description	parameter	ICOS component	responsible	component name	component port	component endpoint
POST `/jupyterlab_service`	Starts a Jupyter lab service	App descriptor	Analytics module	CeADAR	API to start jupyter lab	8080	POST `/core/analytics_jupyterlab_service`

Additionally, a JupyterHub service is supported in the AI Support container (for edge devices), offering user login capabilities. However, it requires administrator access to create and manage user accounts within the container before notebooks can be shared. The steps to do this are described below.

Configuring JupyterHub

- 1) Enter the Docker Container:
  - Use the following command to access the ICOS Intelligence Coordination API Docker container:

```
docker exec -u root -it icos_intelligence_docker /bin/bash
```

- 2) Create a User Account (Inside Docker):
  - Once inside the container, create a user account:

```
passwd UC1        # Replace UC1 with your desired username
```

- 3) Run JupyterHub:
  - Execute the following command within the Docker container, using your JupyterHub configuration file:

```
jupyterhub -f /path/to/jupyterhub_config.py  # Replace with the actual path
```

- 4) Access JupyterHub (Browser):
  - Open your web browser and navigate to the appropriate address to access your JupyterHub session.
    - Enter the credentials you created in step 2.

II.C Intelligence coordination – frontend workflows

The export metrics API seamlessly integrates with the ICOS architecture, forming a crucial bridge between ICOS components and the telemetry data sources, Intelligence Layer functionalities, and monitoring platforms like Prometheus and Thanos. Below is a detailed description of how this integration is achieved:

- Authentication workflow with Keycloak: All requests to the export metrics API and the Intelligence API are secured using Keycloak [40], ensuring that only authorised clients can access the system. Keycloak provides role-based access control, making it easier to manage user permissions across various components. This ensures that sensitive data, such as telemetry metrics and machine learning predictions, is securely handled.

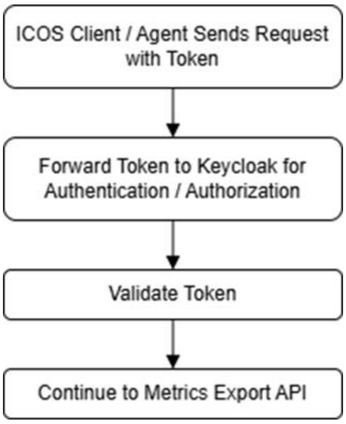

ICOS Client / Agent Sends Request with Token
Forward Token to Keycloak for Authentication / Authorization
Validate Token
Continue to Metrics Export API

Figure 31. Authentication workflow with Keycloak

- Telemetry integration: Telemetry agents like Kube Metrics, ICOS Telemetry Agent, LOMOS, Scaphandre and more, collect real-time data from nodes and systems. This data is streamed to Prometheus, Thanos and Grafana, which act as the primary telemetry controllers in the ICOS architecture. These platforms enable historical data queries and facilitate the creation of metrics for monitoring and predictive purposes.

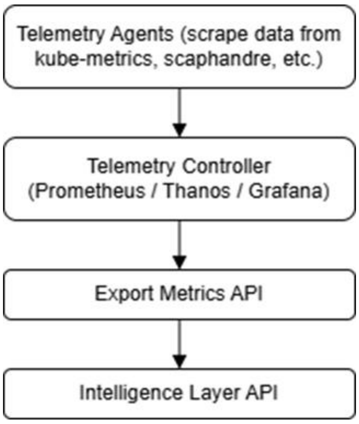

Telemetry Agents (scrape data from kube-metrics, scaphandre, etc.)
Telemetry Controller (Prometheus / Thanos / Grafana)
Export Metrics API
Intelligence Layer API

Figure 32. Telemetry integration workflow

- API workflow: the *Metrics Export API* handles the flow of data from telemetry sources to predictive models and back to the monitoring systems. It interacts with Grafana, Prometheus, and Thanos to retrieve telemetry data and uses the Intelligence API for machine learning-based model training and inference.

- As part of its functionality and part of the static metric creation method, the Export Metrics API implements an automated background process that interacts with the ICOS Aggregator at regular intervals (suggested interval time is 10 * telemetry scrape interval time). This mechanism ensures continuous discovery of infrastructure changes by dynamically detecting new or removed nodes. If a new node is found, the API proactively provisions standard telemetry metrics for CPU, Memory, and Energy consumption. If a node is removed, the system automatically de-registers its associated metrics, preventing redundant data accumulation.
  - The workflow includes retrieving telemetry metrics, formatting them for inference or training, and posting the results as new metrics to Prometheus or Thanos.
  - Figure 34 and Figure 35 show screenshots of the Swagger UI of the AI coordination frontend (export metrics API) and the AI coordination backend (Intelligence API), respectively, as a demonstration of how these interactions occur at the API level.

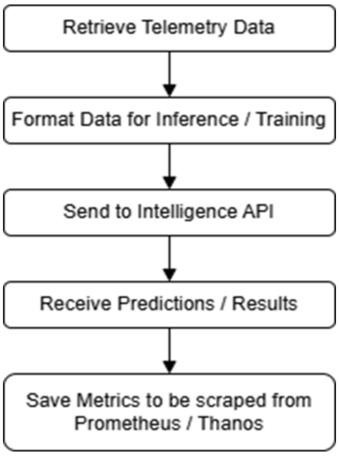
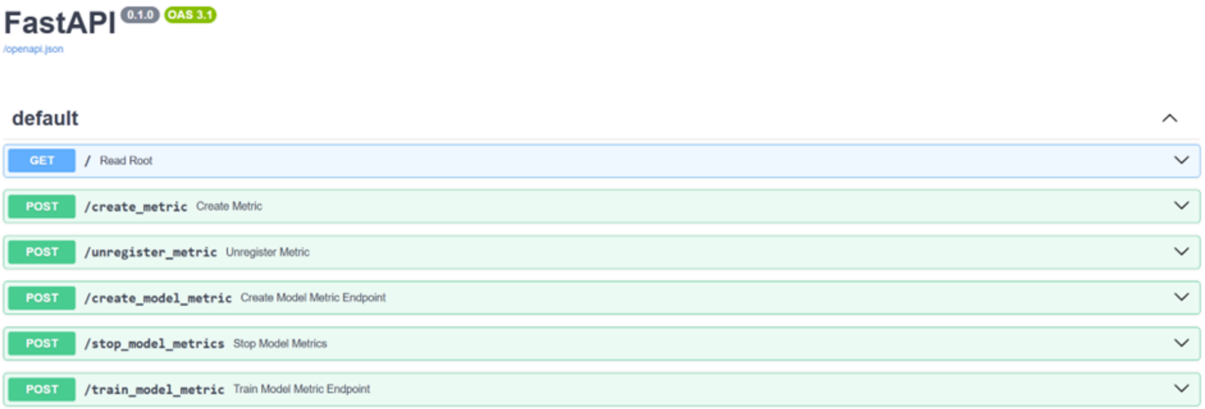

Retrieve Telemetry Data
Format Data for Inference / Training
Send to Intelligence API
Receive Predictions / Results
Save Metrics to be scraped from Prometheus / Thanos

Figure 33. Export Metrics API Workflow

- Metrics and predictions in Prometheus/Thanos: After the AI models generate predictions, the results are saved from Prometheus client library and be scraped from /metrics endpoint as new metrics from Prometheus and Thanos. This ensures that both real-time and predictive insights are available for monitoring, enabling operators to respond proactively to changes in the system.
- Model registry integration: Model training and inference activities are performed by querying the Intelligence Layer AI Coordination API developed as part of D4.1 – “Data management, Intelligence and Security Layers (IT-1)”. Thus, the functionality presented in previous deliverables is kept. New models trained are registered into the model registry part of the Intelligence Layer, ensuring that they can be reused for future inference tasks. This reduces redundancy and enhances the efficiency of the system. The model registry is also framework-agnostic, thus ensuring compatibility between trained models within the ICOS Intelligence Layer and allowing system admins to import their own models to be queried by Intelligence frontend and backend APIs (Export Metrics and Intelligence APIs respectively).
- Role of dataClay in the architecture: dataClay serves as a key component in the training workflow. When telemetry data is used to train machine learning models, it is first prepared and stored in dataClay as structured datasets. These datasets are then accessed by the Intelligence API for model training. dataClay plays an integral role in ensuring that data is organised and readily available for processing.



FastAPI 0.1.0 OAS 3.1

/openapi.json

default

GET / Read Root

POST /create_metric Create Metric

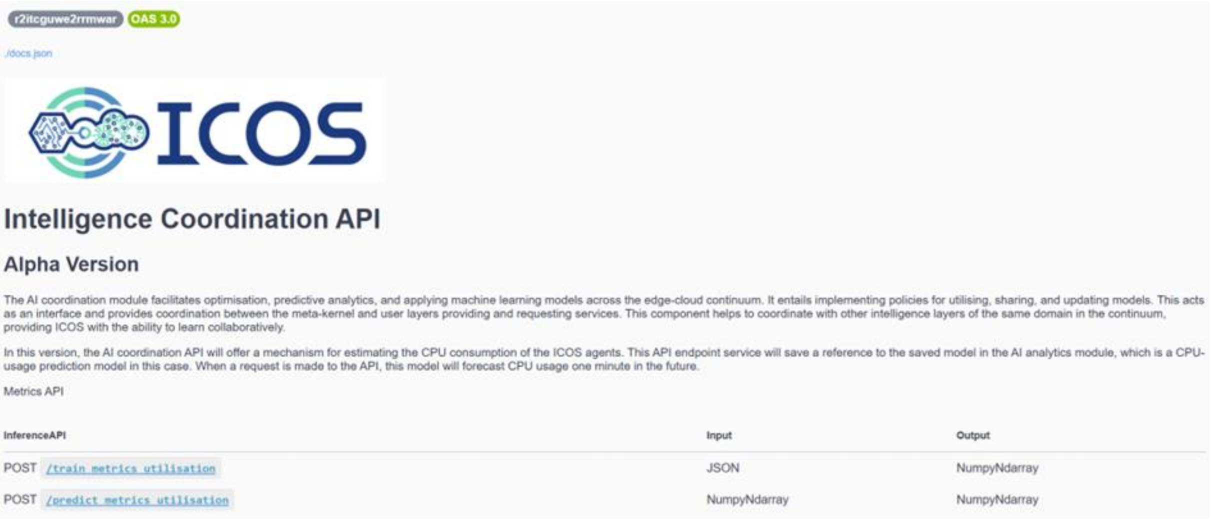
POST /unregister_metric Unregister Metric

POST /create_model_metric Create Model Metric Endpoint

POST /stop_model_metrics Stop Model Metrics

POST /train_model_metric Train Model Metric Endpoint

Figure 34. Export Metrics API Swagger documentation



r2itcguwe2rrmwar OAS 3.0

./docs.json

Intelligence Coordination API

Alpha Version

The AI coordination module facilitates optimisation, predictive analytics, and applying machine learning models across the edge-cloud continuum. It entails implementing policies for utilising, sharing, and updating models. This acts as an interface and provides coordination between the meta-kernel and user layers providing and requesting services. This component helps to coordinate with other intelligence layers of the same domain in the continuum, providing ICOS with the ability to learn collaboratively.

In this version, the AI coordination API will offer a mechanism for estimating the CPU consumption of the ICOS agents. This API endpoint service will save a reference to the saved model in the AI analytics module, which is a CPU-usage prediction model in this case. When a request is made to the API, this model will forecast CPU usage one minute in the future.

Metrics API

InferenceAPI	Input	Output
POST /train_metrics_utilisation	JSON	NumpyNdarray
POST /predict_metrics_utilisation	NumpyNdarray	NumpyNdarray

Figure 35. Intelligence Coordination API Swagger documentation

II.D Intelligence coordination –AI coordination frontend components and integration

The AI Coordination frontend (Export Metrics API) comprises several interconnected components that facilitate metric creation, predictive analytics, and model training. Each component plays a vital role in achieving the system's monitoring objectives.

- AI coordination frontend (Export Metrics API): This component is part of the AI Coordination module and integrates Intelligence's backend (Intelligence API) with MetaOS components such as Telemetry and the Aggregator. Figure 36 depicts an example predictive metric diagram, demonstrating how telemetry data is transformed into actionable metrics using machine learning models. In this diagram the prediction of CPU consumption of an ICOS node is predicted. Green and yellow lines represent predicted and actual CPU consumption respectively.
  - It manages communications between the ICOS MetaOS and Intelligence for metric creation, updates, and deletions.
  - It provides RESTful endpoints for tasks such as generating new metrics, stopping active metrics, and unregistering obsolete metrics.
  - It supports telemetry-based predictive metric creation and model training workflows by integrating with ICOS AI Coordination backend (Intelligence API in D4.1; the Intelligence API is ultimately responsible for training AI models, managing them through a model registry and using them for inference (e.g. predict metrics requested by AI Coordination frontend – "Export Metrics API").
- Telemetry data sources: The system integrates with data sources such as Grafana, Prometheus, and Thanos, which act as repositories of telemetry data.
  - Telemetry agents like Kube Metrics, LOMOS, and Scaphandre continuously stream system data to these platforms. Figure 37 and Figure 38 show diagrams of Grafana and Thanos, illustrating their roles in storing and visualising telemetry data.
- AI coordination backend – Intelligence API: This component performs model training and inference tasks. It is tightly integrated with the Metrics Export API to support workflows such as predictive metric creation and model updates.
  - The Swagger UI for the Intelligence API has been already shown in Section 3.2, highlighting its capabilities for training machine learning models and handling inference requests.
- AI coordination backend – Model registry: The AI coordination backends' model registry, already described in D4.1, stores trained models to be reused. This component enables efficient management of machine learning models, allowing them to be seamlessly integrated into monitoring workflows.
- Keycloak: Keycloak is used to secure access to AI coordination (Metrics Export API and Intelligence API). It provides robust authentication and authorisation mechanisms, ensuring that only authorised users can interact with the system.
- dataClay for data management: dataClay is utilised to prepare and store datasets for model training. It acts as the intermediary storage system, where telemetry data is formatted and organised before being used by the Intelligence API for training machine learning models.
  - Data residing in dataClay can be used through the storage system active features, allowing task offloading of learning processes –i.e., executing these processes within the dataClay backends, where data is.

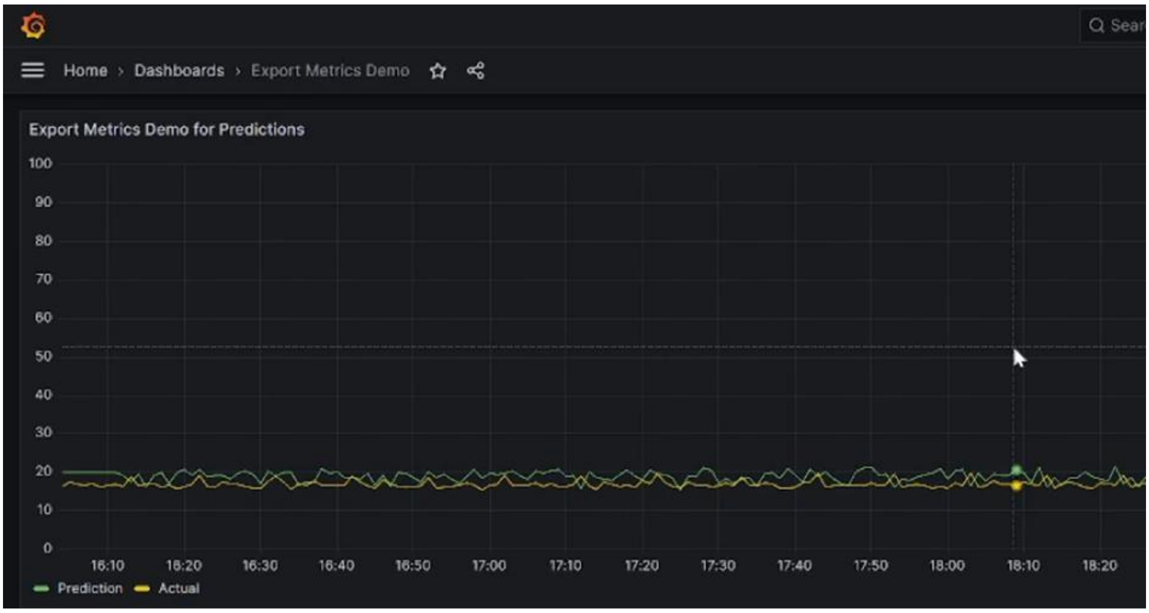
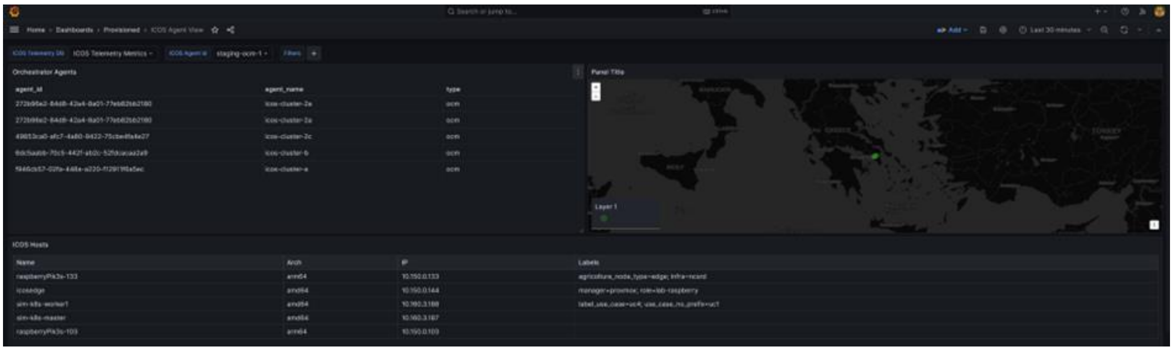


Figure 36. Prediction of CPU consumption (percentage)



Orchestrator Agents

agent_id	agent_name	type
272b9ba2-84d8-42a4-8a01-77eb82bb2180	icos-cluster-2a	ocm
272b9ba2-84d8-42a4-8a01-77eb82bb2180	icos-cluster-2a	ocm
49853ca0-afc7-4a80-8422-75cbe8fa4e27	icos-cluster-2c	ocm
6dc5aab6-70c5-442f-ab2c-52fdcacaa2a9	icos-cluster-b	ocm
5946cb57-02fa-448a-a220-f12911f8a5ec	icos-cluster-a	ocm

ICOS Hosts

Name	Arch	IP	Labels
raspberryPik3s-133	arm64	10.150.0.133	agriculture_node_type=edge; infra=ncsrd
icosedge	amd64	10.150.0.144	manager=proxmox; role=lab-raspberry
sim-k8s-worker1	amd64	10.160.3.188	label_use_case=uc4; use_case_ns_prefix=uc1
sim-k8s-master	amd64	10.160.3.187	
raspberryPik3s-103	arm64	10.150.0.103	

Figure 37. ICOS Grafana telemetry dashboard

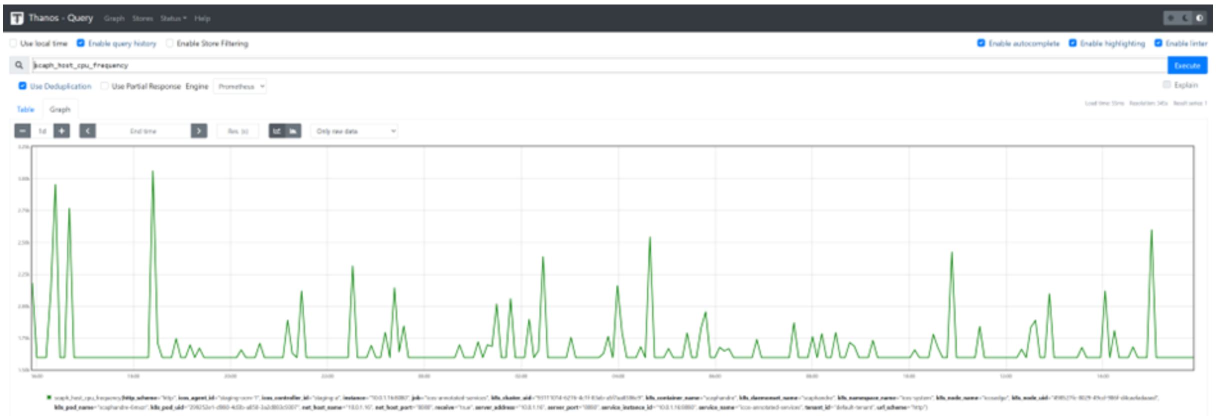


Figure 38. Thanos query for CPU frequency at an ICOS agent

II.E Trustworthy AI – Federated Learning Sequence Diagram & Usage Demonstration

This annex first provides sequence diagrams for the Federated Learning training process.

Then, it provides a practical walkthrough on how to set up the ICOS-FL environment, launch the necessary containers (including Scaphandre, Prometheus, and the dataClay OTLP bridge), and run the ICOS-FL server and clients for federated training.

II.E.1 Sequence Diagram

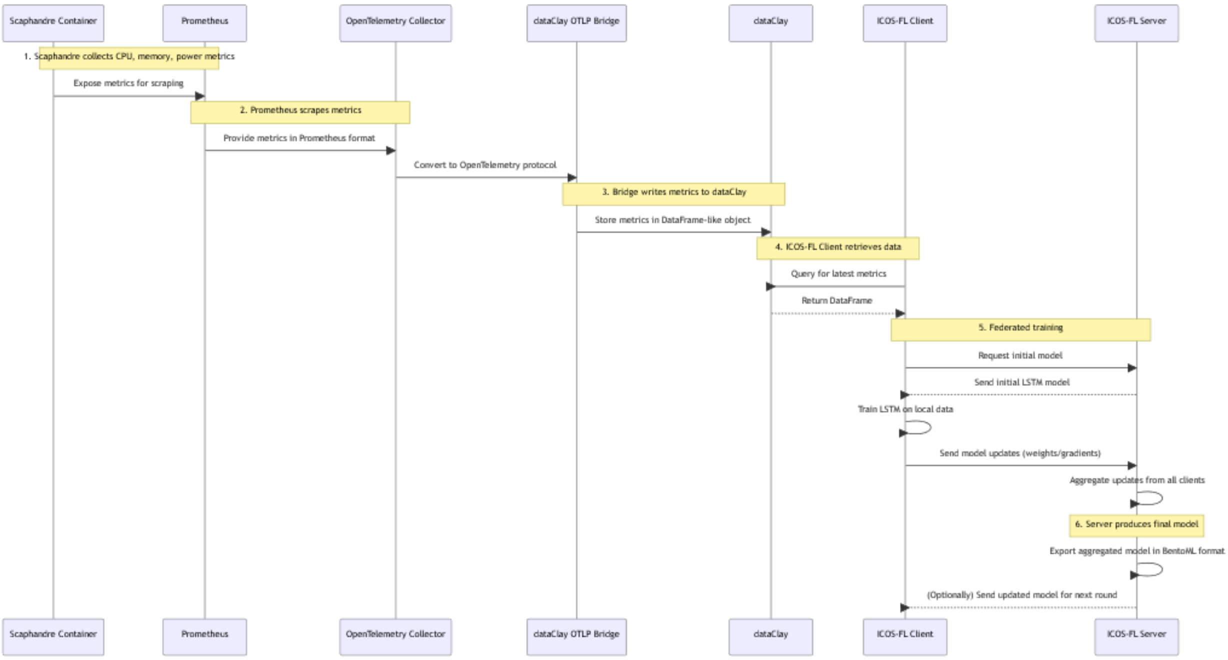

Scaphandre Container
Prometheus
OpenTelemetry Collector
dataClay OTLP Bridge
dataClay
ICOS-FL Client
ICOS-FL Server
1. Scaphandre collects CPU, memory, power metrics
Expose metrics for scraping
2. Prometheus scrapes metrics
Provide metrics in Prometheus format
Convert to OpenTelemetry protocol
3. Bridge writes metrics to dataClay
Store metrics in DataFrame-like object
4. ICOS-FL Client retrieves data
Query for latest metrics
Return DataFrame
5. Federated training
Request initial model
Send initial LSTM model
Train LSTM on local data
Send model updates (weights/gradients)
Aggregate updates from all clients
6. Server produces final model
Export aggregated model in BentoML format
(Optionally) Send updated model for next round
Scaphandre Container
Prometheus
OpenTelemetry Collector
dataClay OTLP Bridge
dataClay
ICOS-FL Client
ICOS-FL Server

Figure 39. Interactions between ICOS elements and actions during FL training processes

Below is a high-level overview of the pipeline:

- 1) **Scaphandre** -> **Prometheus**: Scaphandre exposes energy-related metrics (CPU usage, memory usage, power consumption). Prometheus scrapes these metrics at specified intervals.
- 2) **Prometheus** -> **OpenTelemetry Collector**: The collector ingests these metrics and transforms them to the OpenTelemetry protocol.
- 3) **OpenTelemetry Collector** -> **dataClay OTLP Bridge**: The OTLP Bridge ingests the metrics from the collector and stores them in dataClay objects.
- 4) **ICOS-FL** -> **dataClay**: The ICOS-FL clients retrieve the data from dataClay (already in a DataFrame form) to conduct local training.
- 5) **ICOS-FL Server**: Oversees the training rounds, aggregates updates, and exports the final model in BentoML format.

II.E.2 Components of the FL process

- 1) ICOS-FL Server **(**icos_fl_server**)**
  - **Controller -** in IT-2, this element initialises the federated learning process, specifying the model type (e.g., LSTM), model artifacts and any related hyperparameters. The first step here is to trigger a dataClay container through the dataClay OTLP bridge.
    - **Aggregator** - at dataClay node. It receives the FL workload from the controller. It waits for *clients* in other ICOS nodes to connect (also through the OTLP bridge), dispatches the initial model, and aggregates the updates from each client.
    - **Aggregator->Controller -** Once the final (globally aggregated) model is available in the FL server, this is transferred back to the model registry in the Intelligence Layer at the ICOS Controller.
- 2) ICOS-FL Client (icos_fl_client)
  - Runs on each participating ICOS node.
  - Retrieves metrics from dataClay (as a DataFrame) for local training.
  - Conducts the training on the local dataset for the current round.
  - Sends model weight updates (or gradient updates) back to the server.
  - Supports continuous retraining, periodically re-fetching data and updating the model.
- 3) dataClay OTLP Bridge
  - Intercepts data from the OpenTelemetry collector and writes it into dataClay.
  - Ensures real-time or batched data ingestion, making metrics available for FL clients.
  - In practice, it includes a Docker Compose stack with an OpenTelemetry container, the bridge container, and a dataClay server container.
- 4) Prometheus and Scaphandre
  - **Scaphandre**: Container that monitors power usage (CPU frequencies, etc.) for energy-efficient operation metrics.
  - **Prometheus**: Scrapes metrics from Scaphandre and other sources, offering them to the OpenTelemetry collector.
- 5) Docker Compose Deployment
  - A typical deployment scenario includes Docker Compose services for Scaphandre, Prometheus, the dataClay OTLP Bridge, and optionally any pipeline for the OTLP collector.
  - The environment is configurable so that you can enable or disable dummy collectors, adapt ports, or store data in alternative volumes or remote dataClay services.

II.E.3 Usage demonstration

Figure references in this subsection show screenshots demonstrating each stage.

- 1) Install Python (version 3.10 or higher): Make sure Python 3.10+ is installed on your system. You can verify this by running:

```
python –version
```

- 2) Create a Virtual Environment: Use your preferred environment manager (e.g., conda, venv, pyenv) to create and activate a virtual environment. For example, using conda:

```
conda create -n icos-fl python=3.12
conda activate icos-fl
```

- 3) Clone the Repositories: Clone both the ICOS-FL repository and the OTLP-dataClay bridge repository:

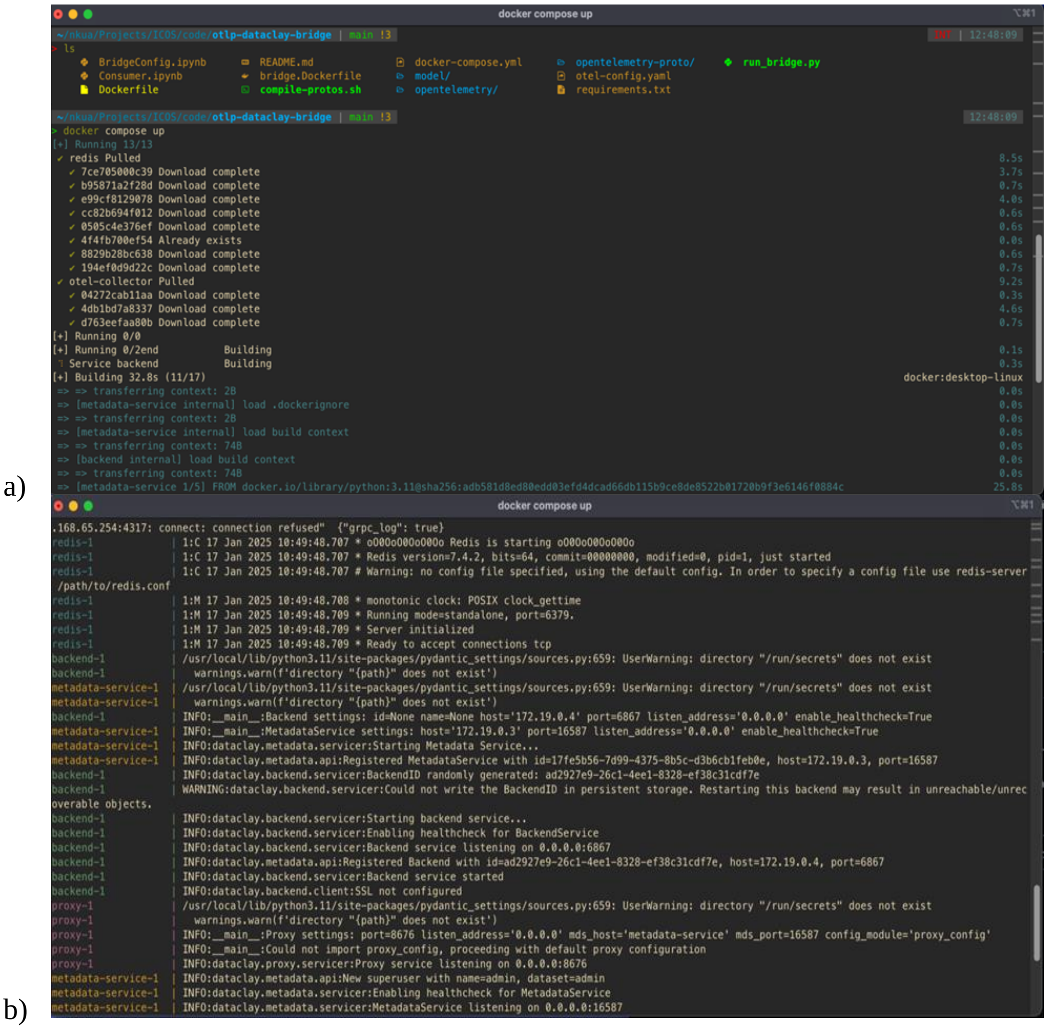
```
git clone https://github.com/anaskalt/icos-fl.git
git clone https://production.eng.it/gitlab/icos/data-management/otlp-dataclay-bridge.git
```

- 4) Copy Configuration Files: From the ICOS-FL repository, copy the new OpenTelemetry configuration and the docker-compose.yml file (which includes Scaphandre and Prometheus) into your working directory. This ensures that when you run docker compose up, all required services will be spun up together.

- 5) Launch the Docker Services: Navigate to the folder containing the new docker-compose.yml and run:

```
docker compose up
```

This starts the containers for Scaphandre, Prometheus, OpenTelemetry, dataClay, and the OTLP bridge.



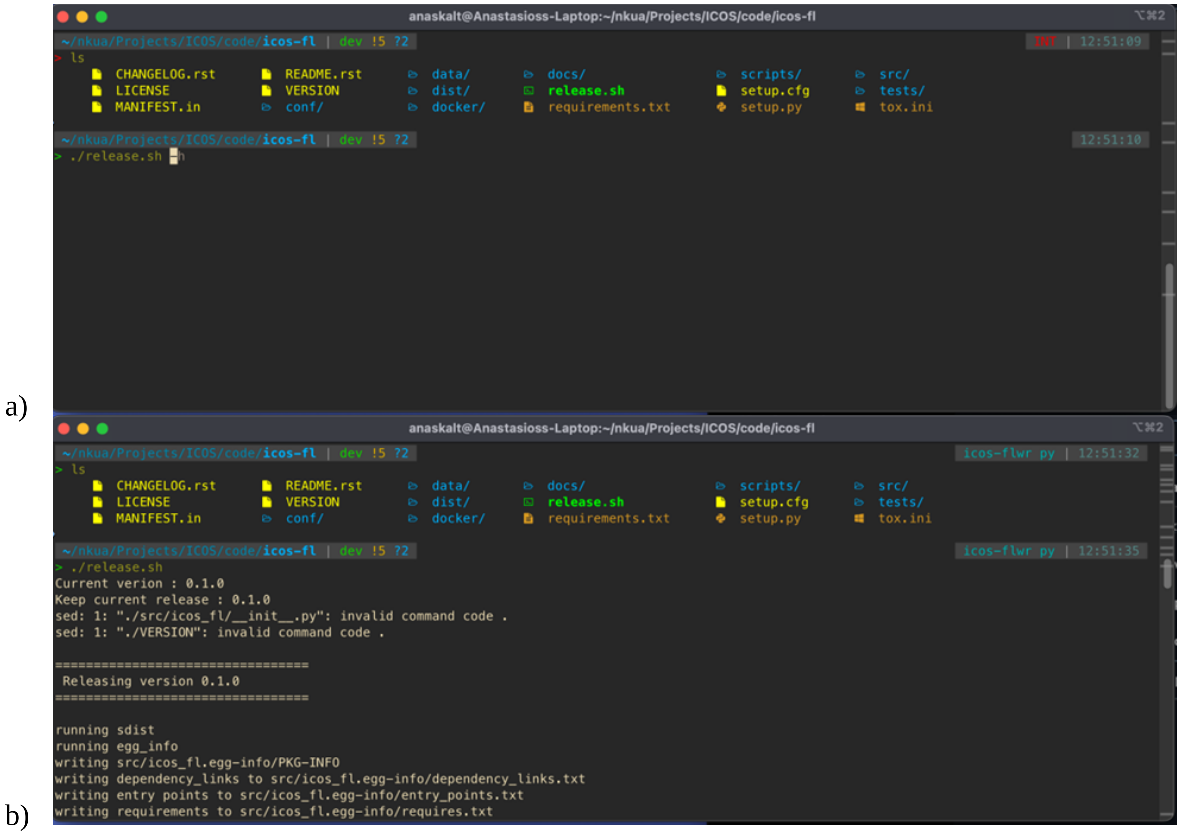
```
~/nkua/Projects/ICOS/code/otlp-dataclay-bridge | main !3                                              INT | 12:48:09
> ls
  BridgeConfig.ipynb      README.md             docker-compose.yml     opentelemetry-proto/    run_bridge.py
  Consumer.ipynb          bridge.Dockerfile     model/                 otel-config.yaml
  Dockerfile              compile-protos.sh     opentelemetry/         requirements.txt

~/nkua/Projects/ICOS/code/otlp-dataclay-bridge | main !3                                                    12:48:09
> docker compose up
[+] Running 13/13
 ✓ redis Pulled                                                                                                 8.5s
   ✓ 7ce705000c39 Download complete                                                                             3.7s
   ✓ b95871a2f28d Download complete                                                                             0.7s
   ✓ e99cf8129078 Download complete                                                                             4.0s
   ✓ cc82b694f012 Download complete                                                                             0.6s
   ✓ 0505c4e376ef Download complete                                                                             0.6s
   ✓ 4f4fb700ef54 Already exists                                                                                0.0s
   ✓ 8829b28bc638 Download complete                                                                             0.6s
   ✓ 194ef0d9d22c Download complete                                                                             0.7s
 ✓ otel-collector Pulled                                                                                        9.2s
   ✓ 04272cab11aa Download complete                                                                             0.3s
   ✓ 4db1bd7a8337 Download complete                                                                             4.6s
   ✓ d763eefaa80b Download complete                                                                             0.7s
[+] Running 0/0
[+] Running 0/2end           Building                                                                           0.1s
 ⠙ Service backend           Building                                                                           0.3s
[+] Building 32.8s (11/17)                                                                       docker:desktop-linux
 => => transferring context: 2B                                                                                 0.0s
 => [metadata-service internal] load .dockerignore                                                              0.0s
 => => transferring context: 2B                                                                                 0.0s
 => [metadata-service internal] load build context                                                              0.0s
 => => transferring context: 74B                                                                                0.0s
 => [backend internal] load build context                                                                       0.0s
 => => transferring context: 74B                                                                                0.0s
 => [metadata-service 1/5] FROM docker.io/library/python:3.11@sha256:adb581d8ed80edd03efd4dcad66db115b9ce8de8522b01720b9f3e6146f0884c   25.8s
```

```
.168.65.254:4317: connect: connection refused"  {"grpc_log": true}
redis-1             | 1:C 17 Jan 2025 10:49:48.707 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
redis-1             | 1:C 17 Jan 2025 10:49:48.707 * Redis version=7.4.2, bits=64, commit=00000000, modified=0, pid=1, just started
redis-1             | 1:C 17 Jan 2025 10:49:48.707 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server
 /path/to/redis.conf
redis-1             | 1:M 17 Jan 2025 10:49:48.708 * monotonic clock: POSIX clock_gettime
redis-1             | 1:M 17 Jan 2025 10:49:48.709 * Running mode=standalone, port=6379.
redis-1             | 1:M 17 Jan 2025 10:49:48.709 * Server initialized
redis-1             | 1:M 17 Jan 2025 10:49:48.709 * Ready to accept connections tcp
backend-1           | /usr/local/lib/python3.11/site-packages/pydantic_settings/sources.py:659: UserWarning: directory "/run/secrets" does not exist
backend-1           |   warnings.warn(f'directory "{path}" does not exist')
metadata-service-1  | /usr/local/lib/python3.11/site-packages/pydantic_settings/sources.py:659: UserWarning: directory "/run/secrets" does not exist
metadata-service-1  |   warnings.warn(f'directory "{path}" does not exist')
backend-1           | INFO:__main__:Backend settings: id=None name=None host='172.19.0.4' port=6867 listen_address='0.0.0.0' enable_healthcheck=True
metadata-service-1  | INFO:__main__:MetadataService settings: host='172.19.0.3' port=16587 listen_address='0.0.0.0' enable_healthcheck=True
metadata-service-1  | INFO:dataclay.metadata.servicer:Starting Metadata Service...
metadata-service-1  | INFO:dataclay.metadata.api:Registered MetadataService with id=17fe5b56-7d99-4375-8b5c-d3b6cb1feb0e, host=172.19.0.3, port=16587
backend-1           | INFO:dataclay.backend.servicer:BackendID randomly generated: ad2927e9-26c1-4ee1-8328-ef38c31cdf7e
backend-1           | WARNING:dataclay.backend.servicer:Could not write the BackendID in persistent storage. Restarting this backend may result in unreachable/unrec
overable objects.
backend-1           | INFO:dataclay.backend.servicer:Starting backend service...
backend-1           | INFO:dataclay.backend.servicer:Enabling healthcheck for BackendService
backend-1           | INFO:dataclay.backend.servicer:Backend service listening on 0.0.0.0:6867
backend-1           | INFO:dataclay.metadata.api:Registered Backend with id=ad2927e9-26c1-4ee1-8328-ef38c31cdf7e, host=172.19.0.4, port=6867
backend-1           | INFO:dataclay.backend.servicer:Backend service started
backend-1           | INFO:dataclay.backend.client:SSL not configured
proxy-1             | /usr/local/lib/python3.11/site-packages/pydantic_settings/sources.py:659: UserWarning: directory "/run/secrets" does not exist
proxy-1             |   warnings.warn(f'directory "{path}" does not exist')
proxy-1             | INFO:__main__:Proxy settings: port=8676 listen_address='0.0.0.0' mds_host='metadata-service' mds_port=16587 config_module='proxy_config'
proxy-1             | INFO:__main__:Could not import proxy_config, proceeding with default proxy configuration
proxy-1             | INFO:dataclay.proxy.servicer:Proxy service listening on 0.0.0.0:8676
metadata-service-1  | INFO:dataclay.metadata.api:New superuser with name=admin, dataset=admin
metadata-service-1  | INFO:dataclay.metadata.servicer:Enabling healthcheck for MetadataService
metadata-service-1  | INFO:dataclay.metadata.servicer:MetadataService listening on 0.0.0.0:16587
```

Figure 40. Screenshots of the console output and the running containers during the FL process

- 6) Build the ICOS-FL Package: Switch to a new terminal (while the Docker containers continue running) and navigate to the ICOS-FL repository. Build the package by running:

```
./release.sh
```



a)

```
anaskalt@Anastasioss-Laptop:~/nkua/Projects/ICOS/code/icos-fl                                        ⌥⌘2
~/nkua/Projects/ICOS/code/icos-fl | dev !5 ?2                                          INT | 12:51:09
> ls
    CHANGELOG.rst     README.rst     data/     docs/                scripts/     src/
    LICENSE           VERSION        dist/     release.sh           setup.cfg    tests/
    MANIFEST.in       conf/          docker/   requirements.txt     setup.py     tox.ini

~/nkua/Projects/ICOS/code/icos-fl | dev !5 ?2                                                12:51:10
> ./release.sh
```

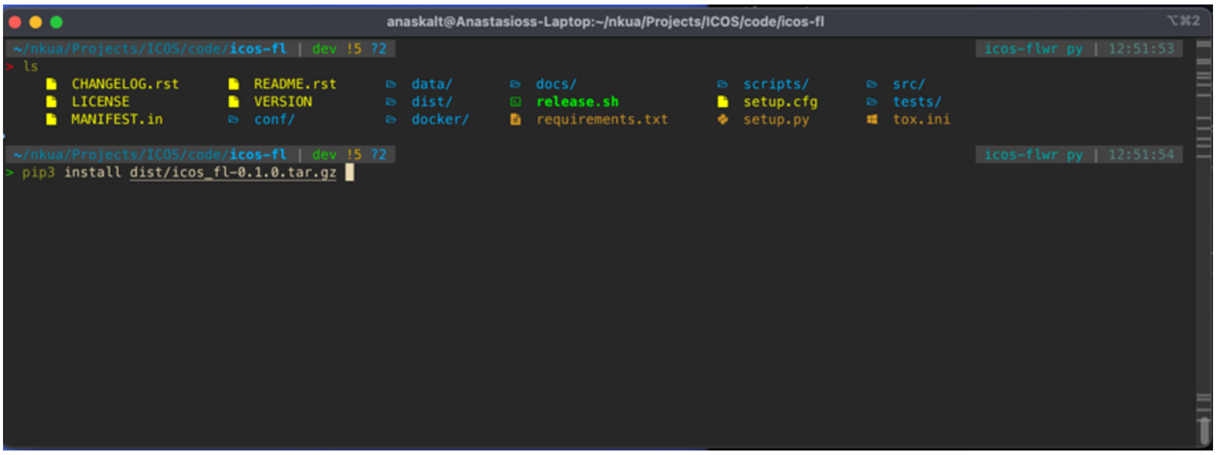
b)

```
anaskalt@Anastasioss-Laptop:~/nkua/Projects/ICOS/code/icos-fl                                        ⌥⌘2
~/nkua/Projects/ICOS/code/icos-fl | dev !5 ?2                                  icos-flwr py | 12:51:32
> ls
    CHANGELOG.rst     README.rst     data/     docs/                scripts/     src/
    LICENSE           VERSION        dist/     release.sh           setup.cfg    tests/
    MANIFEST.in       conf/          docker/   requirements.txt     setup.py     tox.ini

~/nkua/Projects/ICOS/code/icos-fl | dev !5 ?2                                  icos-flwr py | 12:51:35
> ./release.sh
Current verion : 0.1.0
Keep current release : 0.1.0
sed: 1: "./src/icos_fl/__init__.py": invalid command code .
sed: 1: "./VERSION": invalid command code .

================================
 Releasing version 0.1.0
================================

running sdist
running egg_info
writing src/icos_fl.egg-info/PKG-INFO
writing dependency_links to src/icos_fl.egg-info/dependency_links.txt
writing entry points to src/icos_fl.egg-info/entry_points.txt
writing requirements to src/icos_fl.egg-info/requires.txt
```

Figure 41. Screenshots of the FL related build process steps and output

- 7) Install the ICOS-FL Package: Once the build is complete, install it into your virtual environment:

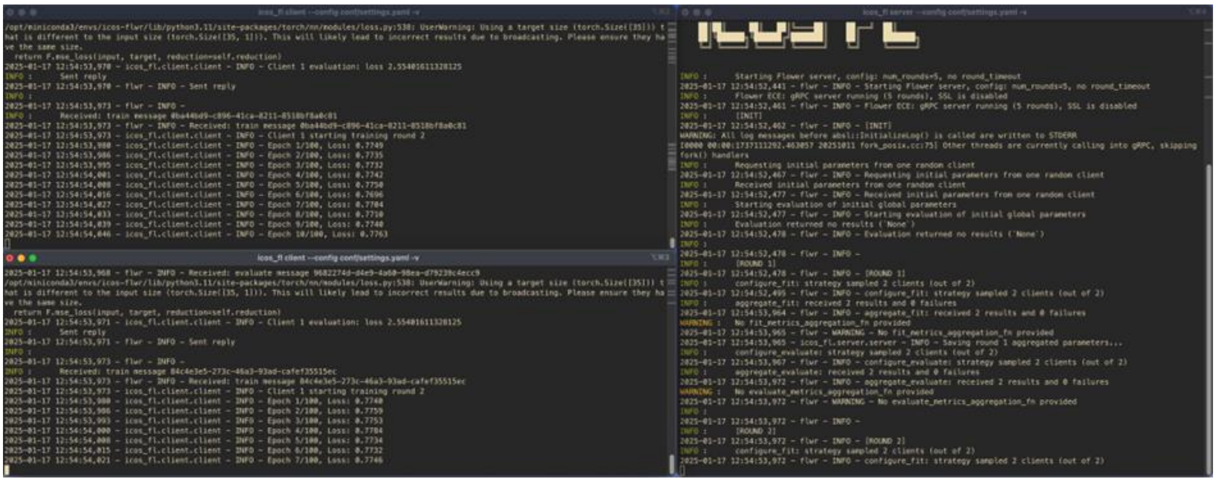
```
pip install dist/icos_fl-<version>.tar.gz
```



```
anaskalt@Anastasioss-Laptop:~/nkua/Projects/ICOS/code/icos-fl                                        ⌥⌘2
~/nkua/Projects/ICOS/code/icos-fl | dev !5 ?2                                  icos-flwr py | 12:51:53
> ls
    CHANGELOG.rst     README.rst     data/     docs/                scripts/     src/
    LICENSE           VERSION        dist/     release.sh           setup.cfg    tests/
    MANIFEST.in       conf/          docker/   requirements.txt     setup.py     tox.ini

~/nkua/Projects/ICOS/code/icos-fl | dev !5 ?2                                  icos-flwr py | 12:51:54
> pip3 install dist/icos_fl-0.1.0.tar.gz
```

Figure 42. Screenshot of the successful installation of the FL env

- 8) <u>Run the Server and Clients</u>: For the example demonstration, open three additional terminals:
    - Terminal 1 – t1: `icos_fl server --config conf/settings.yaml -v`
    - Terminal 2 – t2: `icos_fl client --config conf/settings.yaml -v`
    - Terminal 3 – t3: `icos_fl client --config conf/settings.yaml -v`


icos_fl client --config conf/settings.yaml -v
icos_fl client --config conf/settings.yaml -v
icos_fl server --config conf/settings.yaml -v

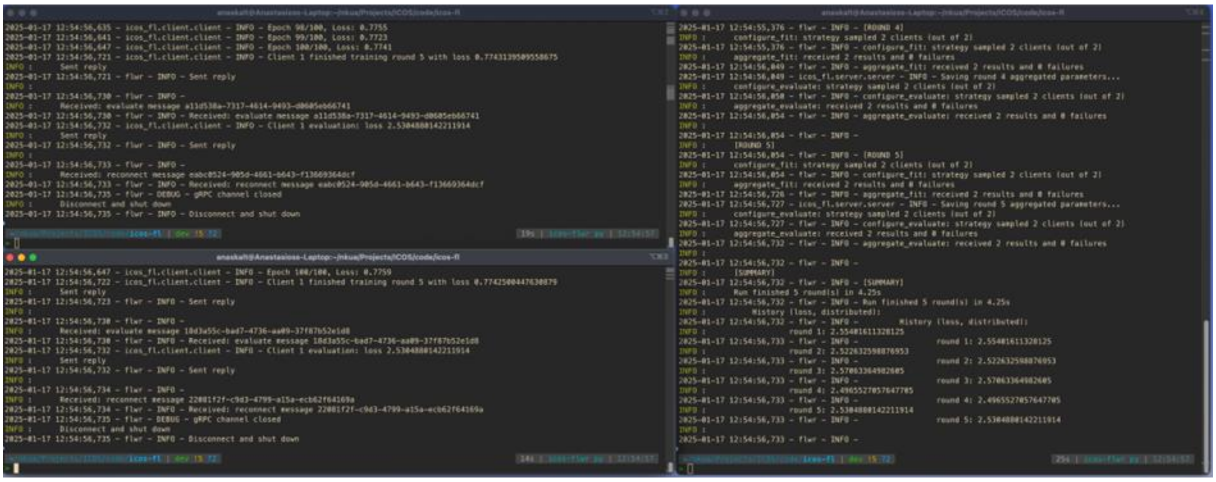
```
/opt/miniconda3/envs/icos-flwr/lib/python3.11/site-packages/torch/nn/modules/loss.py:538: UserWarning: Using a target size (torch.Size([35])) t
hat is different to the input size (torch.Size([35, 1])). This will likely lead to incorrect results due to broadcasting. Please ensure they ha
ve the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
2025-01-17 12:54:53,970 - icos_fl.client.client - INFO - Client 1 evaluation: loss 2.55401611328125
INFO :      Sent reply
2025-01-17 12:54:53,970 - flwr - INFO - Sent reply
INFO :
2025-01-17 12:54:53,973 - flwr - INFO -
INFO :      Received: train message 0ba44bd9-c896-41ca-8211-8518bf8a0c81
2025-01-17 12:54:53,973 - flwr - INFO - Received: train message 0ba44bd9-c896-41ca-8211-8518bf8a0c81
2025-01-17 12:54:53,973 - icos_fl.client.client - INFO - Client 1 starting training round 2
2025-01-17 12:54:53,980 - icos_fl.client.client - INFO - Epoch 1/100, Loss: 0.7749
2025-01-17 12:54:53,986 - icos_fl.client.client - INFO - Epoch 2/100, Loss: 0.7735
2025-01-17 12:54:53,995 - icos_fl.client.client - INFO - Epoch 3/100, Loss: 0.7732
2025-01-17 12:54:54,001 - icos_fl.client.client - INFO - Epoch 4/100, Loss: 0.7742
2025-01-17 12:54:54,008 - icos_fl.client.client - INFO - Epoch 5/100, Loss: 0.7750
2025-01-17 12:54:54,016 - icos_fl.client.client - INFO - Epoch 6/100, Loss: 0.7696
2025-01-17 12:54:54,027 - icos_fl.client.client - INFO - Epoch 7/100, Loss: 0.7704
2025-01-17 12:54:54,033 - icos_fl.client.client - INFO - Epoch 8/100, Loss: 0.7710
2025-01-17 12:54:54,039 - icos_fl.client.client - INFO - Epoch 9/100, Loss: 0.7740
2025-01-17 12:54:54,046 - icos_fl.client.client - INFO - Epoch 10/100, Loss: 0.7763
```

```
2025-01-17 12:54:53,968 - flwr - INFO - Received: evaluate message 9682274d-d4e9-4a60-98ea-d79239c4ecc9
/opt/miniconda3/envs/icos-flwr/lib/python3.11/site-packages/torch/nn/modules/loss.py:538: UserWarning: Using a target size (torch.Size([35])) t
hat is different to the input size (torch.Size([35, 1])). This will likely lead to incorrect results due to broadcasting. Please ensure they ha
ve the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
2025-01-17 12:54:53,971 - icos_fl.client.client - INFO - Client 1 evaluation: loss 2.55401611328125
INFO :      Sent reply
2025-01-17 12:54:53,971 - flwr - INFO - Sent reply
INFO :
2025-01-17 12:54:53,973 - flwr - INFO -
INFO :      Received: train message 84c4e3e5-273c-46a3-93ad-cafef35515ec
2025-01-17 12:54:53,973 - flwr - INFO - Received: train message 84c4e3e5-273c-46a3-93ad-cafef35515ec
2025-01-17 12:54:53,973 - icos_fl.client.client - INFO - Client 1 starting training round 2
2025-01-17 12:54:53,980 - icos_fl.client.client - INFO - Epoch 1/100, Loss: 0.7740
2025-01-17 12:54:53,986 - icos_fl.client.client - INFO - Epoch 2/100, Loss: 0.7759
2025-01-17 12:54:53,993 - icos_fl.client.client - INFO - Epoch 3/100, Loss: 0.7753
2025-01-17 12:54:54,000 - icos_fl.client.client - INFO - Epoch 4/100, Loss: 0.7784
2025-01-17 12:54:54,008 - icos_fl.client.client - INFO - Epoch 5/100, Loss: 0.7734
2025-01-17 12:54:54,015 - icos_fl.client.client - INFO - Epoch 6/100, Loss: 0.7732
2025-01-17 12:54:54,021 - icos_fl.client.client - INFO - Epoch 7/100, Loss: 0.7746
```

```
INFO :      Starting Flower server, config: num_rounds=5, no round_timeout
2025-01-17 12:54:52,441 - flwr - INFO - Starting Flower server, config: num_rounds=5, no round_timeout
INFO :      Flower ECE: gRPC server running (5 rounds), SSL is disabled
2025-01-17 12:54:52,461 - flwr - INFO - Flower ECE: gRPC server running (5 rounds), SSL is disabled
INFO :      [INIT]
2025-01-17 12:54:52,462 - flwr - INFO - [INIT]
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1737111292.463057 20251011 fork_posix.cc:75] Other threads are currently calling into gRPC, skipping
fork() handlers
INFO :      Requesting initial parameters from one random client
2025-01-17 12:54:52,467 - flwr - INFO - Requesting initial parameters from one random client
INFO :      Received initial parameters from one random client
2025-01-17 12:54:52,477 - flwr - INFO - Received initial parameters from one random client
INFO :      Starting evaluation of initial global parameters
2025-01-17 12:54:52,477 - flwr - INFO - Starting evaluation of initial global parameters
INFO :      Evaluation returned no results (`None`)
2025-01-17 12:54:52,478 - flwr - INFO - Evaluation returned no results (`None`)
INFO :
2025-01-17 12:54:52,478 - flwr - INFO -
INFO :      [ROUND 1]
2025-01-17 12:54:52,478 - flwr - INFO - [ROUND 1]
INFO :      configure_fit: strategy sampled 2 clients (out of 2)
2025-01-17 12:54:52,495 - flwr - INFO - configure_fit: strategy sampled 2 clients (out of 2)
INFO :      aggregate_fit: received 2 results and 0 failures
2025-01-17 12:54:53,964 - flwr - INFO - aggregate_fit: received 2 results and 0 failures
WARNING :   No fit_metrics_aggregation_fn provided
2025-01-17 12:54:53,965 - flwr - WARNING - No fit_metrics_aggregation_fn provided
2025-01-17 12:54:53,965 - icos_fl.server.server - INFO - Saving round 1 aggregated parameters...
INFO :      configure_evaluate: strategy sampled 2 clients (out of 2)
2025-01-17 12:54:53,967 - flwr - INFO - configure_evaluate: strategy sampled 2 clients (out of 2)
INFO :      aggregate_evaluate: received 2 results and 0 failures
2025-01-17 12:54:53,972 - flwr - INFO - aggregate_evaluate: received 2 results and 0 failures
WARNING :   No evaluate_metrics_aggregation_fn provided
2025-01-17 12:54:53,972 - flwr - WARNING - No evaluate_metrics_aggregation_fn provided
INFO :
2025-01-17 12:54:53,972 - flwr - INFO -
INFO :      [ROUND 2]
2025-01-17 12:54:53,972 - flwr - INFO - [ROUND 2]
INFO :      configure_fit: strategy sampled 2 clients (out of 2)
2025-01-17 12:54:53,972 - flwr - INFO - configure_fit: strategy sampled 2 clients (out of 2)
```

Figure 43. Console output from two clients (left – t2/t3), and server aggregating the updates (right – t1).

- 9) <u>Completion of Federated Training</u>: After the designated number of federated rounds (configured in `settings.yaml`), the training and aggregation will conclude. The final aggregated model is stored locally (e.g., in BentoML format to be compatible with the *Intelligence coordination backend*).


anaskalt@Anastasioss-Laptop:~/nkua/Projects/ICOS/code/icos-fl
anaskalt@Anastasioss-Laptop:~/nkua/Projects/ICOS/code/icos-fl
anaskalt@Anastasioss-Laptop:~/nkua/Projects/ICOS/code/icos-fl

```
2025-01-17 12:54:56,635 - icos_fl.client.client - INFO - Epoch 98/100, Loss: 0.7755
2025-01-17 12:54:56,641 - icos_fl.client.client - INFO - Epoch 99/100, Loss: 0.7723
2025-01-17 12:54:56,647 - icos_fl.client.client - INFO - Epoch 100/100, Loss: 0.7741
2025-01-17 12:54:56,721 - icos_fl.client.client - INFO - Client 1 finished training round 5 with loss 0.7743139509558675
INFO :      Sent reply
2025-01-17 12:54:56,721 - flwr - INFO - Sent reply
INFO :
2025-01-17 12:54:56,730 - flwr - INFO -
INFO :      Received: evaluate message a11d538a-7317-4614-9493-d0605eb66741
2025-01-17 12:54:56,730 - flwr - INFO - Received: evaluate message a11d538a-7317-4614-9493-d0605eb66741
2025-01-17 12:54:56,732 - icos_fl.client.client - INFO - Client 1 evaluation: loss 2.5304880142211914
INFO :      Sent reply
2025-01-17 12:54:56,732 - flwr - INFO - Sent reply
INFO :
2025-01-17 12:54:56,733 - flwr - INFO -
INFO :      Received: reconnect message eabc0524-905d-4661-b643-f13669364dcf
2025-01-17 12:54:56,733 - flwr - INFO - Received: reconnect message eabc0524-905d-4661-b643-f13669364dcf
2025-01-17 12:54:56,735 - flwr - DEBUG - gRPC channel closed
INFO :      Disconnect and shut down
2025-01-17 12:54:56,735 - flwr - INFO - Disconnect and shut down

                                 icos-fl | dev !5 ?2                                           19s | icos-flwr py | 12:54:57
```

```
2025-01-17 12:54:56,647 - icos_fl.client.client - INFO - Epoch 100/100, Loss: 0.7759
2025-01-17 12:54:56,722 - icos_fl.client.client - INFO - Client 1 finished training round 5 with loss 0.7742500447630079
INFO :      Sent reply
2025-01-17 12:54:56,723 - flwr - INFO - Sent reply
INFO :
2025-01-17 12:54:56,730 - flwr - INFO -
INFO :      Received: evaluate message 18d3a55c-bad7-4736-aa09-37f87b52e1d8
2025-01-17 12:54:56,730 - flwr - INFO - Received: evaluate message 18d3a55c-bad7-4736-aa09-37f87b52e1d8
2025-01-17 12:54:56,732 - icos_fl.client.client - INFO - Client 1 evaluation: loss 2.5304880142211914
INFO :      Sent reply
2025-01-17 12:54:56,732 - flwr - INFO - Sent reply
INFO :
2025-01-17 12:54:56,734 - flwr - INFO -
INFO :      Received: reconnect message 22081f2f-c9d3-4799-a15a-ecb62f64169a
2025-01-17 12:54:56,734 - flwr - INFO - Received: reconnect message 22081f2f-c9d3-4799-a15a-ecb62f64169a
2025-01-17 12:54:56,735 - flwr - DEBUG - gRPC channel closed
INFO :      Disconnect and shut down
2025-01-17 12:54:56,735 - flwr - INFO - Disconnect and shut down

                                 icos-fl | dev !5 ?2                                           14s | icos-flwr py | 12:54:57
```

```
2025-01-17 12:54:55,376 - flwr - INFO - [ROUND 4]
INFO :      configure_fit: strategy sampled 2 clients (out of 2)
2025-01-17 12:54:55,376 - flwr - INFO - configure_fit: strategy sampled 2 clients (out of 2)
INFO :      aggregate_fit: received 2 results and 0 failures
2025-01-17 12:54:56,049 - flwr - INFO - aggregate_fit: received 2 results and 0 failures
2025-01-17 12:54:56,049 - icos_fl.server.server - INFO - Saving round 4 aggregated parameters...
INFO :      configure_evaluate: strategy sampled 2 clients (out of 2)
2025-01-17 12:54:56,050 - flwr - INFO - configure_evaluate: strategy sampled 2 clients (out of 2)
INFO :      aggregate_evaluate: received 2 results and 0 failures
2025-01-17 12:54:56,054 - flwr - INFO - aggregate_evaluate: received 2 results and 0 failures
INFO :
2025-01-17 12:54:56,054 - flwr - INFO -
INFO :      [ROUND 5]
2025-01-17 12:54:56,054 - flwr - INFO - [ROUND 5]
INFO :      configure_fit: strategy sampled 2 clients (out of 2)
2025-01-17 12:54:56,054 - flwr - INFO - configure_fit: strategy sampled 2 clients (out of 2)
INFO :      aggregate_fit: received 2 results and 0 failures
2025-01-17 12:54:56,726 - flwr - INFO - aggregate_fit: received 2 results and 0 failures
2025-01-17 12:54:56,727 - icos_fl.server.server - INFO - Saving round 5 aggregated parameters...
INFO :      configure_evaluate: strategy sampled 2 clients (out of 2)
2025-01-17 12:54:56,727 - flwr - INFO - configure_evaluate: strategy sampled 2 clients (out of 2)
INFO :      aggregate_evaluate: received 2 results and 0 failures
2025-01-17 12:54:56,732 - flwr - INFO - aggregate_evaluate: received 2 results and 0 failures
INFO :
2025-01-17 12:54:56,732 - flwr - INFO -
INFO :      [SUMMARY]
2025-01-17 12:54:56,732 - flwr - INFO - [SUMMARY]
INFO :      Run finished 5 round(s) in 4.25s
2025-01-17 12:54:56,732 - flwr - INFO - Run finished 5 round(s) in 4.25s
INFO :          History (loss, distributed):
2025-01-17 12:54:56,732 - flwr - INFO -         History (loss, distributed):
INFO :                  round 1: 2.55401611328125
2025-01-17 12:54:56,733 - flwr - INFO -                 round 1: 2.55401611328125
INFO :                  round 2: 2.522632598876953
2025-01-17 12:54:56,733 - flwr - INFO -                 round 2: 2.522632598876953
INFO :                  round 3: 2.57063364982605
2025-01-17 12:54:56,733 - flwr - INFO -                 round 3: 2.57063364982605
INFO :                  round 4: 2.4965527057647705
2025-01-17 12:54:56,733 - flwr - INFO -                 round 4: 2.4965527057647705
INFO :                  round 5: 2.5304880142211914
2025-01-17 12:54:56,733 - flwr - INFO -                 round 5: 2.5304880142211914
INFO :
2025-01-17 12:54:56,733 - flwr - INFO -

                                 icos-fl | dev !5 ?2                                           25s | icos-flwr py | 12:54:57
```

Figure 44. Logs indicating the completion of the FL training process – same terminals as in Figure 43

With these steps, the ICOS-FL environment has been successfully launched, as well as its the data ingestion pipeline (Scaphandre → Prometheus → OpenTelemetry → dataClay), and executed a simple Federated Learning workflow.

II.F AI Analytics – Load balancing model and experiments

Following the framework presented in Section 3.2.1 of D4.1, this document outlines the simulation results for the proposed task offloading scheme. Specifically, assuming the same system model and neural network structures as presented in Figures 10-12 of D4.1, this version describes the changes and numerical results performed during IT-2 for implementing and testing the Decentralised Computation Offloading scheme (DECOFFEE) for Energy Efficiency across ICOS cloud/edge layers [11].

The goal of this implementation was to extend the AI Analytics functionalities with a hybrid (i.e. both horizontal and vertical offloading) computation offloading scheme via Distributed Deep Reinforcement Learning (DDRL) [10]. In brief, DECOFFEE algorithm trains multiple autonomous DRL agents/models (one per ICOS Agent or Node), aiming to ensure that, given computational task arrivals (i.e. user applications), each agent selects where to run the task (i.e. locally, or to another ICOS edge node, or to ICOS cloud layer) so as to minimise task computation delay (and energy consumption) and task drop rate, considering deadlines across tasks.

II.F.1 Functional description

DECOFFEE is a framework that allows agents to decide dynamically between local processing, horizontal offloading (to other edge agents), and vertical offloading (to the cloud). It ensures fair CPU sharing for offloaded tasks, balancing workload distribution across the continuum to maximise system performance and task completion rates. Functionally, DECOFFEE instantiates a DRL model at each ICOS node for purposes of decision-making on incoming computational tasks.

The DRL functional architecture is shown in Figure 45, where each edge node agent receives as input the accumulated information of local task features received and the load of all the other available ICOS (edge/cloud) nodes via the ICOS Telemetry Controller. By inferencing the DRL model, each agent finally decides where to offload the task.

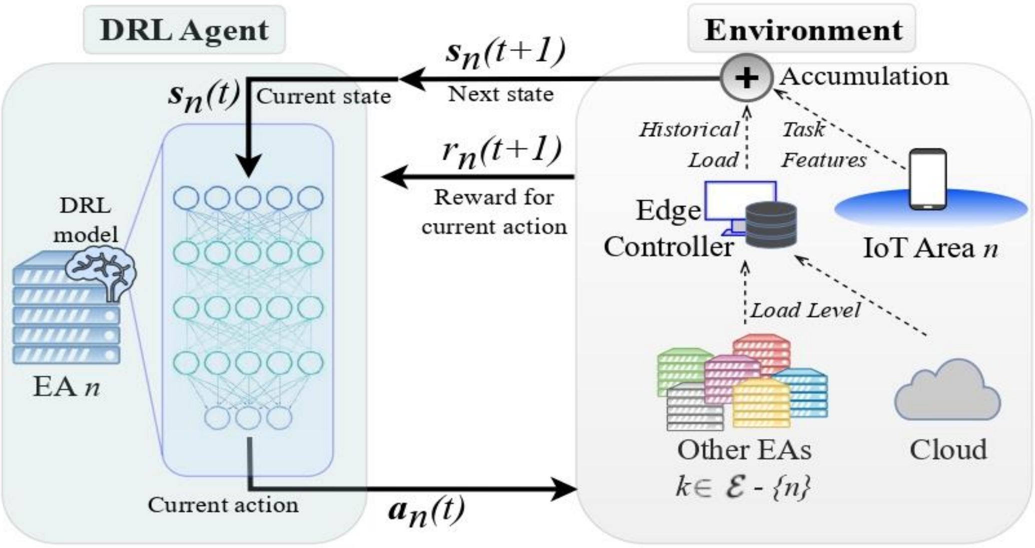


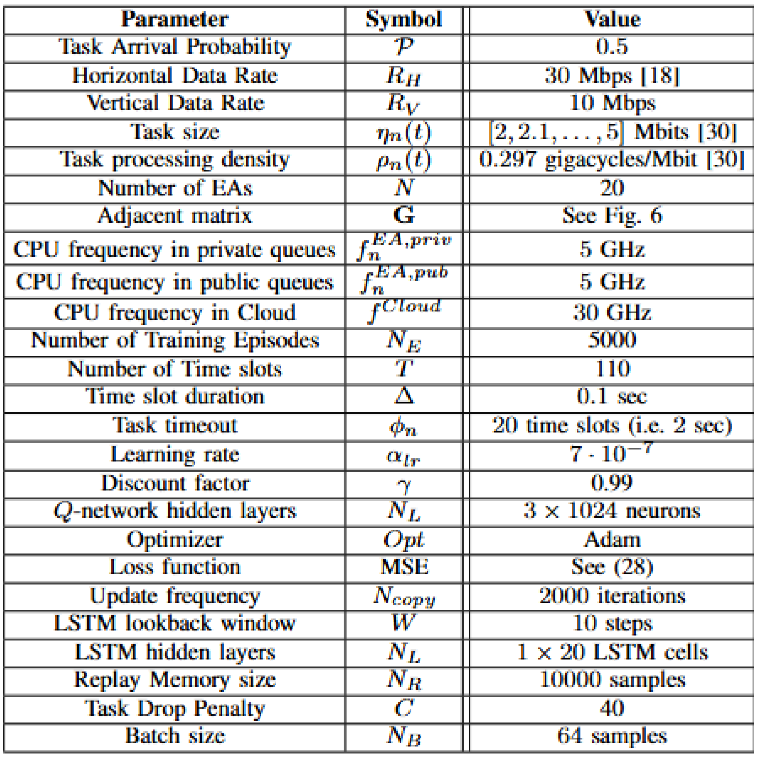
Figure 45. The DRL interaction loop between a DECOFFE-based ICOS node and its env for task offloading

II.F.2 Technical description

Since DECOFFEE requires high task traffic (i.e. multiple user application requests) to reveal performance benefits, **the training and testing phases of DECOFFEE were performed via simulation analysis.**

Implementation libraries: All AI/ML models were implemented in Python 3.x, using PyTorch library (version 1.9.0), and the CUDA (version 10.2).

Optimal training convergence: The impact of the critical learning parameters on the training performance were firstly assessed. The parameters of the simulation environment were set as tabulated in Figure 46 [11].



Parameter	Symbol	Value
Task Arrival Probability	$\mathcal{P}$	0.5
Horizontal Data Rate	R_H	30 Mbps [18]
Vertical Data Rate	R_V	10 Mbps
Task size	$\eta_n(t)$	$[2, 2.1, \ldots, 5]$ Mbits [30]
Task processing density	$\rho_n(t)$	0.297 gigacycles/Mbit [30]
Number of EAs	N	20
Adjacent matrix	$\mathbf{G}$	See Fig. 6
CPU frequency in private queues	$f_n^{EA,priv}$	5 GHz
CPU frequency in public queues	$f_n^{EA,pub}$	5 GHz
CPU frequency in Cloud	f^{Cloud}	30 GHz
Number of Training Episodes	N_E	5000
Number of Time slots	T	110
Time slot duration	Δ	0.1 sec
Task timeout	ϕ_n	20 time slots (i.e. 2 sec)
Learning rate	α_{lr}	$7 \cdot 10^{-7}$
Discount factor	γ	0.99
Q-network hidden layers	N_L	3×1024 neurons
Optimizer	Opt	Adam
Loss function	MSE	See (28)
Update frequency	N_{copy}	2000 iterations
LSTM lookback window	W	10 steps
LSTM hidden layers	N_L	1×20 LSTM cells
Replay Memory size	N_R	10000 samples
Task Drop Penalty	C	40
Batch size	N_B	64 samples

Figure 46. Simulation parameters to train DECOFFEE DRL agents

Figure 47 initially shows the impact of the learning rate on the training convergence, which was tested across a range of values. In general, the choice of learning rate plays a crucial role in balancing the speed of convergence with the stability of the learning process, since it has a direct influence on how intensely the neural model weights are adjusted during the back-propagation steps. The specific tuning of the learning rate was set under moderate task arrival traffic across all DECOFFEE agents (P=0.5; task arrival probability).

As presented in Figure 47a, where the learning curve of the DECOFFEE scheme across different learning rates is depicted. As such, a joint minimisation of the task completion delay, energy consumption, and task drop rate is guaranteed. The performance metric of HOODIE training is the cumulative reward that was collected in a series of 5000 episodes and averaged across all the distributed DECOFFEE agents [11]. Note that, since the task delay is considered a negative metric, the ideal value is zero, which explains why the reward curves are negative and increasing.

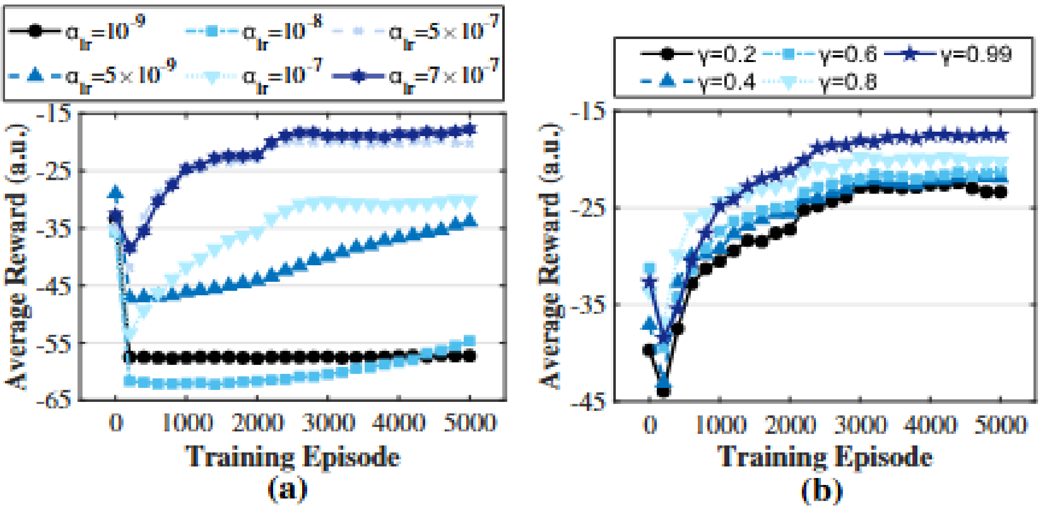


Figure 47. Experiment tracking and learning curves over training episodes in DECOFFEE experiments

In Figure 47b, the impact of the discount factor is also demonstrated, where different reward curves for varying values are shown. Stabilising the value of the discount factor may have generally serious impact on the DRL policy convergence, since it directly regulates balance between immediate and future rewards, the stability and speed of learning. Figure 47b shows that the optimal performance was observed for the value of 0.99, which means that DECOFFEE agents exhibit more importance on future rewards, making them more focused on long-term outcomes.

Comparative numerical analysis: Considering the best configuration of DECOFFEE scheme, a comparative analysis was then performed, comparing the DECOFFEE against six baseline offloading methods (Figure 48):

- <u>Baseline 1 - Random Offloader (RO):</u> Agents randomly choose between local execution, vertical offloading, or horizontal offloading, with a random node selected as the destination.
- <u>Baseline 2 - Full Local Computing (FLC):</u> All tasks are processed locally by the agent.
- <u>Baseline 3 - Vertical Offloader (VO):</u> All tasks are offloaded to the Cloud node for computation.
- <u>Baseline 4 - Horizontal Offloader (HO):</u> Tasks are offloaded to another edge node, with destinations selected randomly.
- <u>Baseline 5 - Balanced Cyclic Offloader (BCO):</u> Actions are selected cyclically (e.g., local, Cloud node, edge node 1, edge node 2, etc.).
- <u>Baseline 6 - Minimum Latency Estimation Offloader (MLEO):</u> Agents estimate the queue delay for all placement options (local, other edge nodes, or Cloud) and select the one with the lowest estimated delay.

The comparison focuses on two performance metrics: average computation delay (which is proportional to the energy consumption) and drop ratio, calculated over 200 validation episodes. For delay analysis, tasks were allowed up to 10 seconds to stay in the system, highlighting the minimum delay for completed tasks (Figure 48a-c). For drop ratio (Figure 48d-f), a stricter 2-second timeout was applied, measuring the proportion of dropped tasks to total arrivals. Delay values are represented as negative to reflect their adverse impact on task completion.

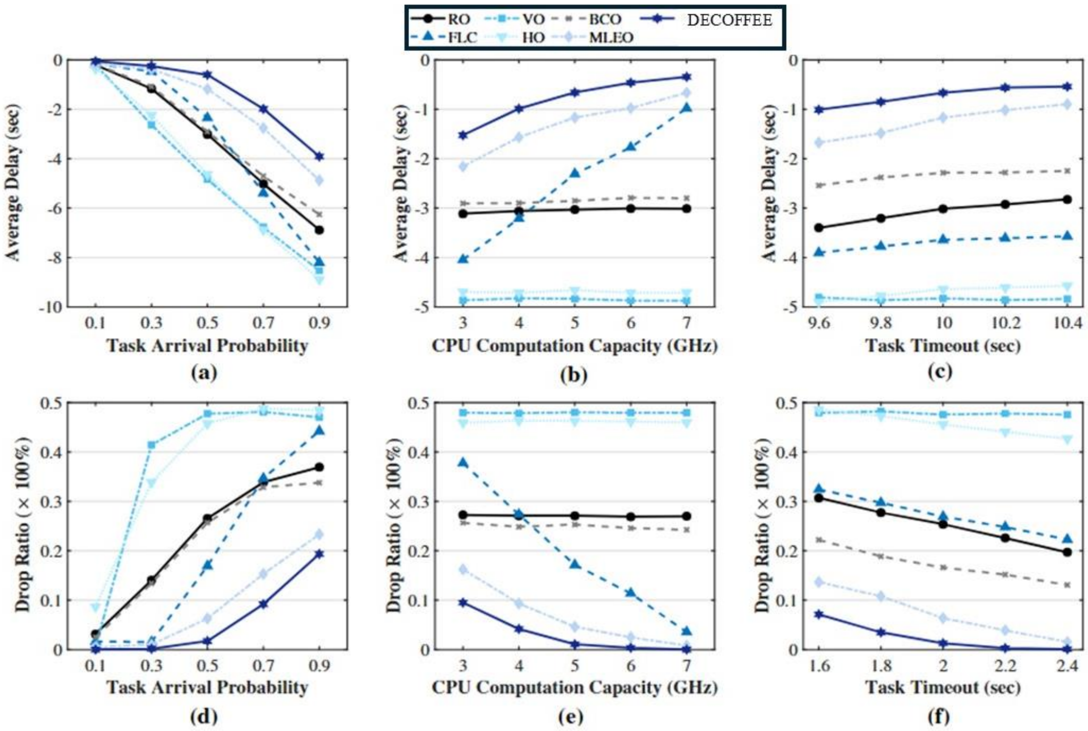


Figure 48. Performance comparison of DECOFFEE against six offloading schemes under varying conditions

(a) Average delay vs. task arrival probability; (b) Average delay vs. CPU computation capacity; (c) Average delay vs. task timeout; (d) Drop ratio vs. task arrival probability; (e) Drop ratio vs. CPU computation capacity; (f) Drop ratio vs. task timeout. Average delay is negative by convention

- The Impact of Task Traffic Intensity: Figure 48a shows the average delay as a function of task arrival probability. As the probability increases, delays decrease across all schemes due to higher system loads, with DECOFFEE consistently achieving the lowest delays. HO and VO perform well under low task loads but exhibit significant delays at higher probabilities. MLEO is competitive at moderate loads but falls behind HOODIE as probabilities approach 0.9. Figure 48d presents the drop ratio, which rises with task arrival probability. DECOFFEE maintains the lowest drop ratios, demonstrating its effectiveness under heavy loads, while FLC and HO show poor performance at high probabilities. MLEO performs adequately at moderate loads but struggles in extreme conditions.
- The Impact of Computational Capacity: Figure 48b highlights the effect of local CPU capacity (3–7 GHz) on average delay. Increasing CPU capacity reduces delays across most schemes, with DECOFFEE consistently achieving the lowest delays, particularly at higher capacities. MLEO also performs well but less effectively than DECOFFEE at lower capacities, where offloading is critical. Figure 48e shows drop ratios decrease with increasing CPU capacity. DECOFFEE achieves the most significant reduction, especially at lower capacities, while FLC benefits from local computation but remains less effective under heavy loads.
- The Impact of Task Timeout: Figure 48c shows that increasing task timeouts (9.6–10.4 seconds) improves delays across all schemes. DECOFFEE achieves the lowest delays by optimising scheduling dynamically, while FLC remains consistent due to exclusive local computation. MLEO performs slightly worse than DECOFFEE, particularly under tighter deadlines. Figure 48f shows that extending timeouts (1.6–2.4 seconds) reduces drop ratios, with HOODIE maintaining the lowest rates, demonstrating its robustness under strict deadlines.

II.G AI Analytics – Metrics forecasting and internal workflows in IT-2

Building on D4.1's ARIMA-based CPU utilisation forecasting evaluated via MSE/MAE/MAPE, RMSE [12], and having portability metrics such as the inferencing time, and RAM. IT-2 enhances the Intelligence Layer coordination backend API with multivariate prediction.

- The AI analytics module now employs PyTorch LSTMs to forecast CPU and memory usage, supporting both univariate and multivariate (simultaneous CPU/memory) approaches.

II.G.1 Functional description

In this version, the Intelligence Layer Coordination API extends its capabilities by introducing multivariate forecasting for CPU and memory utilisation, alongside the existing univariate approach. The API now integrates a refined mechanism to reference the updated LSTM-based models (trained on PyTorch) stored in the AI analytics module.

These models predict both CPU and memory usage one step ahead using historical sequences, enabling simultaneous multivariate prediction. This enhancement strengthens the ICOS resource allocation and management techniques, allowing dynamic scaling decisions based on CPU/memory trends, rather than isolated metrics.

II.G.2 Technical description

The training dataset was sourced from the ICOS Telemetry component via a Grafana dashboard, capturing CPU and memory utilisation metrics across ICOS nodes over a minimum of two weeks, recorded at 5-minute intervals in CSV format. Evaluation using Mean Absolute Error (MAE) demonstrated that LSTM-based models outperformed other architectures (e.g., ARIMA, XGBoost) for both univariate and multivariate forecasting tasks.

II.G.2.1Model Training Workflow

- 1) Data Retrieval & Preprocessing**:** The Export Metrics API retrieves raw telemetry data, which undergoes preprocessing (cleaning, outlier removal, data normalisation, and feature selection).
  - A configurable parameter defines the prediction type: univariate or multivariate. The data is then reshaped into a time-series structure compatible with the selected model.
- 2) Baseline Training Configuration**:** The default LSTM model employs a one-step-ahead forecasting approach with an 80/20 train-test split.
  - The Swagger API allows users to trigger training for univariate (ARIMA/XGBoost) or multivariate (LSTM) models, leveraging time-series data from Grafana.
- 3) Model Deployment & Storage**:** Pre-processed data is forwarded to the Intelligence Layer API (*backend*), which trains the model and saves it to the AI repository with performance metrics and a unique identifier.
  - Model parameters and metadata are logged in an MLOps tool for traceability and future analysis.
  - The Intelligence backend API subsequently serves forecasts using the trained model, accessible via its assigned tag.

II.G.2.2 Prototype architecture

Data preprocessing pipelines, part of the Data Processing component, ensure consistent input quality from distributed nodes, enabling the training of models to boost prediction accuracy. Agents or nodes interact via an Export Metrics API, which initiates workflows where intelligence modules preprocess data and dynamically route it to either univariate models or multivariate models. Both workflows culminate in models being added to a centralised repository, accessed through a unified prediction API. This API endpoint standardises interactions while serving predictions from any model, whether trained for a single node or across multiple nodes.

This whole process enhances the system by integrating metrics documentation directly into Swagger, improving API transparency and usability, as shown in Figure 49.

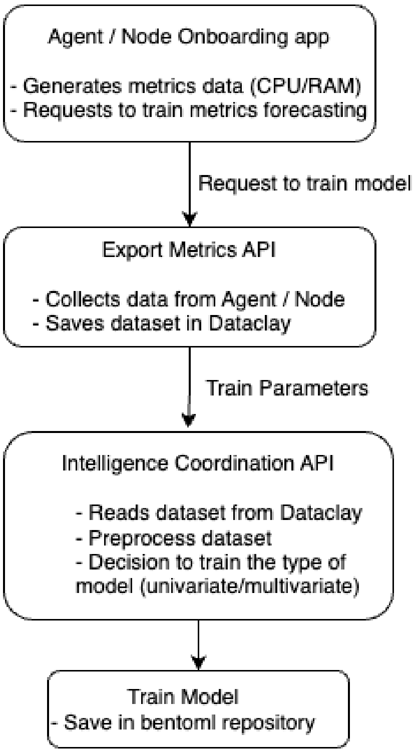

Agent / Node Onboarding app
- Generates metrics data (CPU/RAM)
- Requests to train metrics forecasting
Request to train model
Export Metrics API
- Collects data from Agent / Node
- Saves dataset in Dataclay
Train Parameters
Intelligence Coordination API
- Reads dataset from Dataclay
- Preprocess dataset
- Decision to train the type of model (univariate/multivariate)
Train Model
- Save in bentoml repository

Figure 49. Workflow of Metrics Collection, Forecasting, and Model Training in an Onboarding App

A similar procedure is performed for prediction generation, as shown in Figure 50; when a newly onboarded agent or node requires metrics predictions, it initiates a request through the Export Metrics API, which collects and provides access to the relevant performance metrics, such as CPU and memory usage.

The collected data is then processed and forwarded to the Intelligence Coordination backend API, which leverages trained models to generate accurate and actionable predictions. This streamlined interaction enables agents or nodes to make informed decisions based on real-time and forecasted system performance.

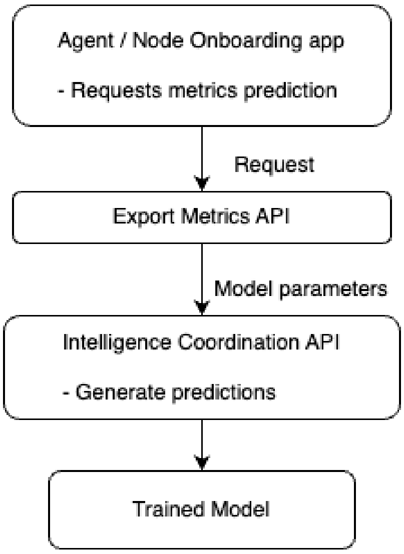

Agent / Node Onboarding app
- Requests metrics prediction
Request
Export Metrics API
Model parameters
Intelligence Coordination API
- Generate predictions
Trained Model

Figure 50. Predictive Intelligence Flow for Metrics Forecasting

II.G.2.3 Description of components

As detailed in D4.1, the LSTM model for data training is developed within the PyTorch framework. This model is subsequently deployed as a service utilising BentoML. The API facilitates a two-step process:

- Model training: Users can train the model using their provided dataset. Upon successful training, the model is securely stored within BentoML's model repository.
- Inference: The trained model is then exposed through a predict API. To generate predictions, the API accepts scaled values of past CPU and Memory utilisation as input. The output comprises a single predicted value for both future CPU and Memory utilisation.

II.H Technical specifications

This annex covers technical specifications of Intelligence Layer modules and functionalities.

II.H.1 Trustworthy AI module

Explainable AI

Description: This can be viewed using MLFlow UI by invoking the `start_mlflow_ui()` service endpoint within the Intelligence Coordination API. This initiates an MLFlow UI instance, typically running on port 5000, accessible via http://127.0.0.1:5000.

In the launched UI, locate and open the experiment containing model explanations. From there, click to view the SHAP plots.

Dependencies: MLFlow (2.13.2), Python (3.10), BentoML (1.2.16)

Deployment: Packaged inside Intelligence Layer Docker container.

Prediction confidence scores

The technical stack is composed of the following:

Programming Language: Python 3.10

Framework: Scikit-learn (for model performance metrics) [27]

Deployment: The results are available while generating model predictions.

Model monitoring

This drift detection functionality is part of the Swagger API and can be activated as needed, providing flexibility in monitoring and analysing data drift over time. As defined in Section 3.3.2.3, the API call is defined as follows:

```
POST/detect_drift
```

Description*:* Identifies and returns a list of detected drifts in the provided data. The `detect_drift` API call activates drift detection, employing statistical measures such as Jensen-Shannon Divergence to identify shifts in predictions. The resulting drift metrics highlight how predictions evolve over time, enabling users to detect critical issues such as performance degradation, concept drift, and input distribution shifts.

Values:

- **True:** Activates drift detection
- **False:** Skips drift detection.

Required: Yes

Input: A Boolean (True or False) to indicate if drift detection should be applied into the present data.

Output: A structured response containing detected drift events.

Federated Learning

The technical stack is composed of the following:

Programming Language: Python 3.10

Framework: Flower FL for orchestrating the Federated Learning process. PyTorch (for LSTM model implementation)

Model Architecture: LSTM for time-series prediction of CPU usage, memory usage, and power consumption.

Metrics Ingestion: Scaphandre → Prometheus → OpenTelemetry → dataClay pipeline.

Model Export: Final model stored in BentoML format.

Deployment: Docker Compose for local or distributed setups (multiple containers for Scaphandre, Prometheus, dataClay OTLP Bridge, etc.).

Continuous Retraining: Supported. The training loop can be triggered at intervals depending on new data availability or scheduled tasks.

II.H AI Analytics module

AIOps system – MLFlow

Description: The MLFlow UI can be launched by invoking the `start_mlflow_ui()` service endpoint within the Intelligence Coordination API. This initiates an MLFlow UI instance, typically running on port 5000, accessible via http://localhost:5000

Dependencies: MLFlow(2.13.2), Python(3.10), BentoML(1.2.16).

Deployment: Packaged inside Intelligence coordination Docker.

Model Compression

The quantisation and distillation features are configurable via a POST API endpoint `/core/analytics_train` exposed through the AI coordination backend service (Intelligence API). Users can submit training requests using either Swagger API documentation. The parameters are defined in the request body under `pytorch_model_parameters`, which includes the following configurable fields:

```
{
"pytorch_model_parameters":
        { "hidden_size": 64,
        "num_epochs": 50,ta
         "quantize": true,
        "distill": false }
}
```

There are 3 possible scenarios for implementing quantisation:

- Quantisation (`quantize: true`)**:** Dynamic quantisation from PyTorch's native quantisation module[17], which is applied post-training after the model is fully trained in FP32.
- Distillation (`distill: true`)**:** Trains two models concurrently: a teacher (larger) and a student (smaller) by default. Uses a customisable distillation loss function.
- Combined Workflow (`quantize: true distill: true`)**:** When both flags are enabled, the pipeline first trains a distilled student model for higher accuracy. Applies dynamic quantisation after distillation, producing a lightweight INT8 version of the tuned distilled model.

It is important to note that quantisation and distillation are mutually independent but can be combined for a compressed yet accurate model. The final model artifact (INT8 or FP32) is stored in the Intelligence Layer model registry for deployment.

[17] https://pytorch.org/docs/stable/quantization.html#torch.quantization.quantize_dynamic

II.I Delivery and usage

Deliverable D4.2 primarily focuses on package upgrades, notably for BentoML and MLFlow, to align with the latest versions used in IT-1. Additionally, it introduces new packages to enhance explainability and model monitoring capabilities. Importantly, the core API usage remains consistent with no significant changes from the previous deliverable.

II.I.1 Package information

Table 8 represents the dependencies of the Intelligence Layer container for a successful deployment and service.

Table 8. Intelligence Layer dependencies

Package name	Version	Package info
bentoml	1.2.16	https://github.com/bentoml/BentoML
dataclay	icos-ai-dataclay	https://github.com/bsc-dom/icos-ai-dataclay
mlflow	2.13.2	https://mlflow.org/docs/2.13.2/index.html
numpy	>=1.15.0,<1.25.2	https://numpy.org/doc/stable/
pandas	>=1.4.3,<2.0.0	https://pandas.pydata.org/docs/
seaborn	>=0.11.2,<=0.12.2	https://github.com/mwaskom/seaborn
matplotlib	3.7.1	https://matplotlib.org/3.7.1/index.html
scikit-learn	1.2.2	https://scikit-learn.org/stable/whats_new/v1.2.html
notebook	6.5.1	https://docs.jupyter.org/en/latest/
xgboost	1.7.6	https://xgboost.readthedocs.io/en/stable/
tqdm	4.65.0	https://tqdm.github.io/
pydantic	1.10.10	https://docs.pydantic.dev/latest/
scipy	1.10.1	https://docs.scipy.org/doc/scipy/
river	0.14.0	https://riverml.xyz/0.14.0/
tensorflow	2.10.0	https://github.com/tensorflow/tensorflow/releases/tag/v2.10.0
tflite	2.10.0	https://github.com/tensorflow/tflite-support/releases
tensorflow_hub	0.12.0	https://github.com/tensorflow/hub/tree/r0.12
tensorflow_datasets	4.7.0	https://github.com/tensorflow/datasets/tree/v4.7.0

Package name	Version	Package info
tesnsorflow-model-oprtimizations	0.7.3	https://github.com/tensorflow/model-optimization/tree/v0.7.3
keras-turner	1.3.5	https://github.com/keras-team/keras-tuner/tree/v1.3.5
statsmodels	0.14.0	https://github.com/statsmodels/statsmodels/tree/v0.14.0
fastapi	0.93.0	https://github.com/fastapi/fastapi/tree/0.93.0
category_encoders	2.6.3	https://github.com/scikit-learn-contrib/category_encoders/tree/2.6.3
shap	0.46.0	https://github.com/shap/shap/tree/v0.46.0
nannyml	0.9.0	https://github.com/NannyML/nannyml/tree/v0.9.0
torch	2.0.0	https://github.com/pytorch/pytorch/tree/v2.0.0

Additionally, the AI coordination frontend (Export metrics API) package dependencies are presented in Table 9.

Table 9. Export Metrics dependencies

Package name	Version	Package info
aiorwlock	1.4.0	https://pypi.org/project/aiorwlock/
annotated-types	0.6.0	https://pypi.org/project/annotated-types/
anyio	4.3.0	https://pypi.org/project/anyio/
bcrypt	4.2.0	https://pypi.org/project/bcrypt/
cached-property	1.5.2	https://pypi.org/project/cached-property/
certifi	2024.7.4	https://pypi.org/project/certifi/
cffi	1.17.1	https://pypi.org/project/cffi/
charset-normalizer	3.3.2	https://pypi.org/project/charset-normalizer/
click	8.1.7	https://pypi.org/project/click/
colorama	0.4.6	https://pypi.org/project/colorama/
contourpy	1.3.0	https://pypi.org/project/contourpy/
cryptography	44.0.0	https://pypi.org/project/cryptography/
cycler	0.12.1	https://pypi.org/project/cycler/
dataclay	4.0.0	https://github.com/bsc-dom/icos-ai-dataclay

Package name	Version	Package info
Deprecated	1.2.14	https://pypi.org/project/Deprecated/
deprecation	2.1.0	https://pypi.org/project/deprecation/
ecdsa	0.19.0	https://pypi.org/project/ecdsa/
fastapi	0.110.0	https://github.com/tiangolo/fastapi
fonttools	4.54.1	https://pypi.org/project/fonttools/
grpcio	1.66.2	https://pypi.org/project/grpcio/
grpcio-health-checking	1.66.2	https://pypi.org/project/grpcio-health-checking/
gunicorn	21.2.0	https://pypi.org/project/gunicorn/
h11	0.14.0	https://pypi.org/project/h11/
hiredis	3.0.0	https://pypi.org/project/hiredis/
httpcore	1.0.7	https://pypi.org/project/httpcore/
httpx	0.26.0	https://pypi.org/project/httpx/
idna	3.6	https://pypi.org/project/idna/
importlib_metadata	8.4.0	https://pypi.org/project/importlib-metadata/
jwcrypto	1.5.6	https://pypi.org/project/jwcrypto/
kiwisolver	1.4.7	https://pypi.org/project/kiwisolver/
matplotlib	3.9.2	https://matplotlib.org/
numpy	2.1.1	https://numpy.org/
opentelemetry-api	1.27.0	https://pypi.org/project/opentelemetry-api/
packaging	24.0	https://pypi.org/project/packaging/
pandas	2.2.3	https://pandas.pydata.org/
pillow	10.4.0	https://pypi.org/project/pillow/
prometheus_client	0.20.0	https://github.com/prometheus/client_python
protobuf	5.28.2	https://pypi.org/project/protobuf/
psutil	6.0.0	https://pypi.org/project/psutil/
pyasn1	0.6.1	https://pypi.org/project/pyasn1/
pycparser	2.22	https://pypi.org/project/pycparser/
pydantic	2.9.2	https://docs.pydantic.dev/
pydantic-settings	2.5.2	https://pypi.org/project/pydantic-settings/

Package name	Version	Package info
pydantic_core	2.23.4	https://pypi.org/project/pydantic-core/
PyJWT	2.9.0	https://pypi.org/project/PyJWT/
pyparsing	3.1.4	https://pypi.org/project/pyparsing/
python-dateutil	2.9.0.post0	https://pypi.org/project/python-dateutil/
python-dotenv	1.0.1	https://pypi.org/project/python-dotenv/
python-jose	3.3.0	https://pypi.org/project/python-jose/
python-keycloak	3.9.1	https://pypi.org/project/python-keycloak/
pytz	2024.2	https://pypi.org/project/pytz/
PyYAML	6.0.2	https://pypi.org/project/PyYAML/
redis	5.1.0	https://pypi.org/project/redis/
requests	2.32.3	https://pypi.org/project/requests/
requests-toolbelt	1.0.0	https://pypi.org/project/requests-toolbelt/
rsa	4.9	https://pypi.org/project/rsa/
six	1.16.0	https://pypi.org/project/six/
sniffio	1.3.1	https://pypi.org/project/sniffio/
starlette	0.36.3	https://pypi.org/project/starlette/
typing_extensions	4.10.0	https://pypi.org/project/typing-extensions/
tzdata	2024.2	https://pypi.org/project/tzdata/
urllib3	2.2.2	https://pypi.org/project/urllib3/
uvicorn	0.28.0	https://pypi.org/project/uvicorn/
wrapt	1.16.0	https://pypi.org/project/wrapt/
zipp	3.20.2	https://pypi.org/project/zipp/

Finally, Table 10 represents FL dependencies for the Federated Learning service before its integration with intelligence in IT-3. In the event of changing, these requirements will be updated in its pertinent GitHub repository (ICOS-FL: https://github.com/anaskalt/icos-fl) and in D5.3 [8].

Table 10. ICOS FL dependencies

Package name	Version	Package info
icos_fl	0.1.0	https://github.com/anaskalt/icos-fl
flwr	1.4.0	https://pypi.org/project/flwr/
torch	2.0.1	https://pypi.org/project/torch/
torchvision	0.15.2	https://pypi.org/project/torchvision/
numpy	1.24.4	https://pypi.org/project/numpy/
pandas	2.0.3	https://pypi.org/project/pandas/
scikit-learn	1.3.0	https://pypi.org/project/scikit-learn/
matplotlib	3.7.1	https://pypi.org/project/matplotlib/
pyyaml	6.0.0	https://pypi.org/project/pyyaml/
psutil	5.9.5	https://pypi.org/project/psutil/
scipy	1.10.1	https://pypi.org/project/scipy/

II.I.2 Installation instructions

The installation instructions remain largely consistent with those described in deliverable D4.1. However, the command to launch the Docker container has been modified to include additional configurations. These configurations, specified via environment variables, allow for the setting of resource limits and API timeouts for the containerised application.

As an example, the command to set the API timeout and allocate CPU and GPU resources is provided below.

```
docker run --network host -it --rm -p 3000:3000 -p 5000:5000 -e
BENTOML_CONFIG_OPTIONS='api_server.traffic.timeout=600 runners.resources.cpu=0.5
runners.resources."nvidia.com/gpu"=0' analytics:btdpshel3oztswar serve
```

The above command would enable the ICOS Intelligence Layer Coordination API server to be launched at http://0.0.0.0:3000, which could be accessed using any browser.

II.I.3 Licensing information

Intelligence backend

This remains the same as described in deliverable D4.1.

Regarding the Intelligence Layer in the Controller suite, this section remains the same as described in deliverable D4.1.

Intelligence frontend and federated learning functionality

Regarding the AI coordination frontend (Export Metrics API), as well as the ICOS Federated Learning (FL), these are licensed under the Apache License 2.0. This allows for free usage, modification, and distribution, provided that proper attribution is maintained and modifications are clearly stated.

Copyright © 2022-2024 National and Kapodistrian University of Athens.

II.I.4 Download

Table 11 lists the main Software packages covering the Intelligence Layer.

Table 11. Intelligence Software packages

Software	Instructions
AI coordination backend & Intelligence Layer (Controller suite)	This remains the same as described in deliverable D4.1.
AI coordination frontend (Export metrics API)	Available in ICOS GitHub: https://github.com/icos-project/Metrics-Export
ICOS Federated Learning (FL)	Code for the FL client/server initially available at: https://github.com/anaskalt/icos-fl

Annex III: Security Layer

III.A Requirements

Table 12. ICOS requirements met with the IT-2 of the Sec. Layer Coordination and Sec. Scan modules

Requirement ID	Requirement Name	Description
SST_FR_06	Secure infrastructure and code	ICOS SHOULD assess security of infrastructure (e.g., running Critical Security Controls - CIS benchmarks) and system and application code (e.g. running vulnerability scanning of docker images)
SST_FR_11	Compliance detection	ICOS MUST provide a mechanism for the detection of compliance problems regarding controls of specific standards and/or specific policies and rules.
SST_FR_12	Compliance enforcement	ICOS MUST provide a system to trigger infrastructure changes to ensure standard and/or policy compliance

Table 13. ICOS requirements met with the IT-2 of the IAM module

Requirement ID	Requirement Name	Description
SST_FR_07	SecureAPI	ICOS MUST provide API supporting AuthT/AuthZ and Audit capabilities
SST_FR_08	SecureLIB	ICOS SHOULD provide libraries supporting Authentication /Authorisation to 3rd parties
OP_FR_09	Granularity of the access	ICOS SHOULD provide the possibility to define different roles in accessing to ICOS

Table 14. ICOS requirements met with the IT-2 of the Anomaly detection module

Requirement ID	Requirement Name	Description
SST_FR_09	Anomaly detection	ICOS MUST provide a mechanism for the detection of anomalies (e.g., any kind of abnormal situations, including potential security threats, that is recorded in application, system, or network logs) in the applications/services on the cloud/network/edge provider
SST_FR_10	Anomaly mitigation and recovery	ICOS MUST be able categorise anomalies and recommend specific mitigation actions or recovery process

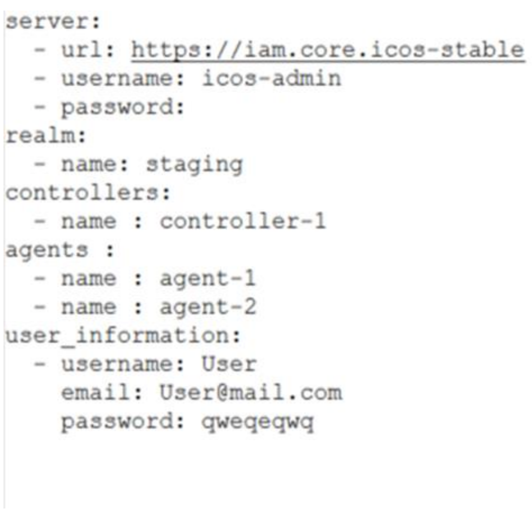
Table 15. ICOS requirements met with the IT-2 of the Audit module

Requirement ID	Requirement Name	Description
SST_FR_06	Secure infrastructure and code	ICOS SHOULD assess security of infrastructure (e.g., running Critical Security Controls - CIS benchmarks) and system and application code (e.g. running vulnerability scanning of docker images)
SST_FR_07	Secure API	ICOS MUST provide API supporting AuthT/AuthZ and Audit capabilities
SST_FR_11	Compliance detection	ICOS MUST provide a mechanism for the detection of compliance problems regarding controls of specific standards and/or specific policies and rules.
SST_FR_12	Compliance enforcement	ICOS MUST provide a system to trigger infrastructure changes to ensure standard and/or policy compliance

Table 16. ICOS requirements met with the IT-2 of the Trust functionality

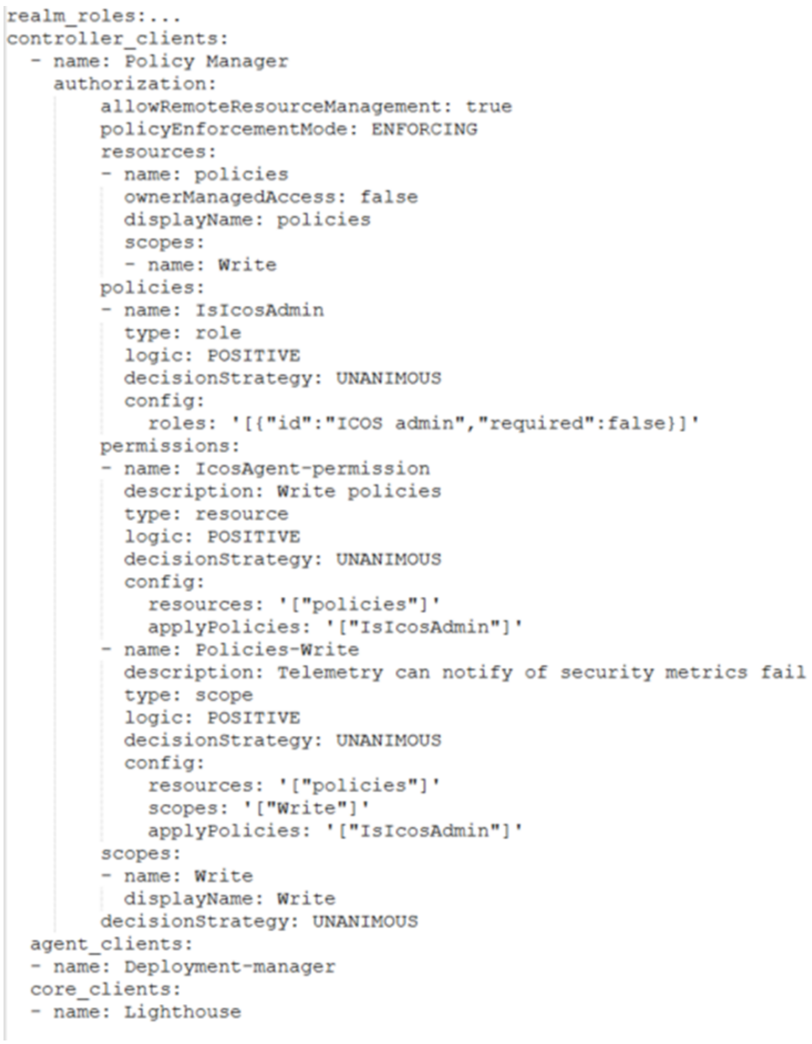
Requirement ID	Requirement Name	Description
SST_FR_01	Trust nodes	ICOS SHOULD be able to provide mechanisms for establishing trust procedures for the onboarding of ICOS nodes and the deployment of applications.
SST_FR_02	Secure comms	ICOS SHOULD be able to exploit available encryption and isolation mechanisms to provide sufficient levels of security at the cloud continuum level

III.B Example of IAM module configuration files



```
server:
  - url: https://iam.core.icos-stable
  - username: icos-admin
  - password:
realm:
  - name: staging
controllers:
  - name : controller-1
agents :
  - name : agent-1
  - name : agent-2
user_information:
  - username: User
    email: User@mail.com
    password: qweqeqwq
```

Figure 51. Example of dynamic_values.yaml



```
realm_roles:...
controller_clients:
  - name: Policy Manager
    authorization:
        allowRemoteResourceManagement: true
        policyEnforcementMode: ENFORCING
        resources:
        - name: policies
          ownerManagedAccess: false
          displayName: policies
          scopes:
          - name: Write
        policies:
        - name: IsIcosAdmin
          type: role
          logic: POSITIVE
          decisionStrategy: UNANIMOUS
          config:
            roles: '[{"id":"ICOS admin","required":false}]'
        permissions:
        - name: IcosAgent-permission
          description: Write policies
          type: resource
          logic: POSITIVE
          decisionStrategy: UNANIMOUS
          config:
            resources: '["policies"]'
            applyPolicies: '["IsIcosAdmin"]'
        - name: Policies-Write
          description: Telemetry can notify of security metrics fail
          type: scope
          logic: POSITIVE
          decisionStrategy: UNANIMOUS
          config:
            resources: '["policies"]'
            scopes: '["Write"]'
            applyPolicies: '["IsIcosAdmin"]'
        scopes:
        - name: Write
          displayName: Write
        decisionStrategy: UNANIMOUS
  agent_clients:
  - name: Deployment-manager
  core_clients:
  - name: Lighthouse
```

Figure 52. Example of static_values.yaml

III.C Delivery and usage

The following tables present package information on the Security Layer modules, with instructions how to download and set them up, as well as relevant information regarding licensing.

- Table 17 presents relevant information for the Software packages concerning Security in IT-2.
- Table 18 presents relevant installations instructions per Security Layer module in IT-2.
- Table 19 lists licensing information relevant to each of the Security Layer modules in IT-2.
- Table 20 explains download instructions for each of the Security Layer modules in IT-2.

Table 17. Package information per module

Requirement ID	Requirement Name
Security Layer Coordination	No change since D4.1.
Security Scan	No change since D4.1.
Identity and Access Management	The Identity and Access Management module is packaged and distributed using Docker containers. In addition, it is integrated as a Helm chart in the ICOS Core Suite to provide an easily deployable and configurable ready-to-use service integrated with the other services in the ICOS Continuum
Anomaly detection	Both the anomaly detection module as well as the anomaly detection API-2 are packaged as Docker containers and can be installed and used as such
Audit	The audit component is packaged as a unified helm chart which automatically installs Trivy, Tetragon and Prometheus exporter with all their dependencies
Trust	- Step-ca is packaged as a Docker container, and it works natively with Kubernetes. - Cilium is a container network interface which is developed for Kubernetes. - Wireguard is present as a kernel module on recent Linux distributions.

Table 18. Installation instructions per module

Requirement ID	Requirement Name
Security Layer Coordination	No change since D4.1.
Security Scan module	Wazuh server is installed in Kubernetes using the official Wazuh Kubernetes manifest, that can be found here: https://github.com/wazuh/wazuh-kubernetes. This allows to install it in the cloud and local environments.

Requirement ID	Requirement Name
Identity and Access Management	No change since D4.1.

Anomaly detection	The anomaly detection API2 can be build and run as a Docker container with the following commands:
	`LOMOS_API2_VERSION=0.0.1`
	`docker build --build-arg "LOMOS_API2_VERSION=$LOMOS_API2_VERSION" -t lomos-api2:$LOMOS_API2_VERSION .`
	`docker run -it --name lomos-api2 --rm -p25001:25001 --env-file=.env lomos-api2:$LOMOS_API2_VERSION`
	`curl "http://172.17.0.2:25001/api/top_anomaly?min_anomaly_score=0.7&from_timestamp=2024-01-08T00:00:01.200000Z&to_timestamp=2024-01-18T00:00:01.200000Z"`

Audit	The Audit component can be installed using the dedicated helm chart though the project's repository: `$helm install tetragon-auditing/icos-auditing`

Trust	Step-ca components are installed from the official helm repository.
	`$ helm install step-certificates smallstep/step-certificates`
	`$ helm install autocert smallstep/autocert`
	`$ helm install step-issuer smallstep/step-issuer`
	Install Cilium with Wireguard encryption using Cilium CLI by following:
	- https://docs.cilium.io/en/stable/gettingstarted/k8s-install-default/
	Enable Wireguard encryption:
	`$ cilium install --chart-directory ./install/kubernetes/cilium    --set encryption.enabled=true    --set encryption.type=wireguard.`

Table 19. Licensing per module and internal element in the Security Layer

Requirement ID	Requirement Name
Security Layer Coordination	No change since D4.1.
Security Scan	- The Wazuh is an open source, licensed with a modified GNU GPLv2 licence. - Trivy is an open-source software under the Apache 2.0 licence
Identity and Access Management	No change since D4.1.
Anomaly detection	- The Anomaly detection module is a proprietary software of XLAB. - Anomaly detection API-2 is an open-source software under the Apache 2.0 licence.
Audit	- Tetragon is open-source software under Apache 2.0 licence.

Requirement ID	Requirement Name
	▸ Trivy is open-source software under Apache 2.0 licence.
Trust	▸ Step-ca is an open-source software under the Apache 2.0 licence. ▸ Cilium is an open-source project under the Apache 2.0 licence. ▸ Wireguard is licensed under GPLv2, as it's part of the Linux kernel.

Table 20. Download instructions per module and internal element in the Security Layer

Requirement ID	Requirement Name
Security Layer Coordination	The source code of the Security Layer Coordination API can be downloaded or pulled from the project`s GitHub repository: https://github.com/icos-project/Security-coordination-module
Security Scan	▸ The source code of Wazuh can be downloaded or pulled from the official Wazuh GitHub repository: - https://github.com/wazuh/wazuh ▸ The source code of Trivy can be downloaded or pulled from the official Trivy GitHub repository: - https://github.com/aquasecurity/trivy
Identity and Access Management	▸ At the time of writing the deliverable, the source code of the Identity and Access Management module can be downloaded or pulled from the project's development Gitlab repository: - https://production.eng.it/gitlab/icos/security/iam ▸ The repository can be opened on invitation. To request access to this repository, please contact: - Mr. Gabriele Giammatteo (gabriele.giammatteo@eng.it)
Anomaly detection	The source code of the Anomaly detection API-2 can be downloaded or pulled from the project`s GitHub repository: https://github.com/icos-project/lomos-api2
Audit	The source code of Audit module can be downloaded or pulled from the project's GitHub repository: https://github.com/icos-project/tetragon-auditing
Trust	▸ The source code of step-ca can be downloaded or pulled from the official step-ca GitHub repository: - https://github.com/smallstep/certificates ▸ The source code of Cilium can be downloaded or pulled from the official Github repository: - https://github.com/cilium/cilium