ICOS

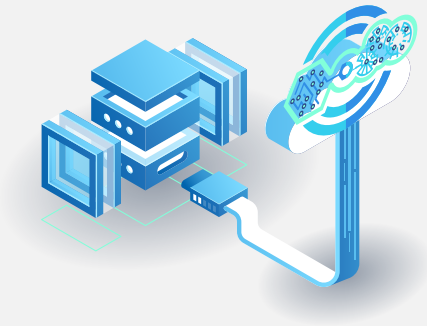
Contents

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

What is ICOS?

The next wave of transformation in computing systems architecture is driven by the rapid development of computing and sensing device technologies and the ever-growing demand for data-intensive applications in the edge and cloud. This leads to the paradigm shift of computing being centered around dynamic, intelligent and yet seamless interconnection of IoT, edge and cloud resources in one computing system, to form a continuum. Various instantiations of such a continuum, referred also as cloud continuum, IoT continuum, edge-to-cloud or fog-to- cloud, are expected to provide the means for real-time or offline data processing, both at the edge and cloud.

ICOS represents an **IoT2Cloud meta-Operating System** that aims to manage the continuum by providing a framework that will be extensible, open, secure, adaptable, AI-powered as well as well highly performant and technology agnostic. ICOS aims at achieving a balanced combination of both cloud and edge computing capabilities, according to the specific requirements of the considered workloads, to facilitate easy and transparent use of the available resources, by providing efficiency, interoperability, robustness, and also ensuring security.



A MetaOS for the Continuum

A typical cloud continuum scenario includes a certain number of high performance cloud computing facilities, either public or private, and a number of devices at the edge, with various levels of computing capacity. Cloud computing facilities provide unlimited and ubiquitous computing and storage capacity, but their remoteness becomes a challenge, since user data must be transported to the cloud and, therefore, applications expose high latency inherent to distance, while users lose their privacy. Alternatively, devices at the edge, such as local data centers, servers, or even nearby computing devices such as smartphones, in-car computers, or embedded computing facilities around the city, provide limited computing capacity, but their neighboring location reduces network data transfer and latency, and it preserves data privacy. A typical cloud continuum scenario is illustrated in Fig. 1.

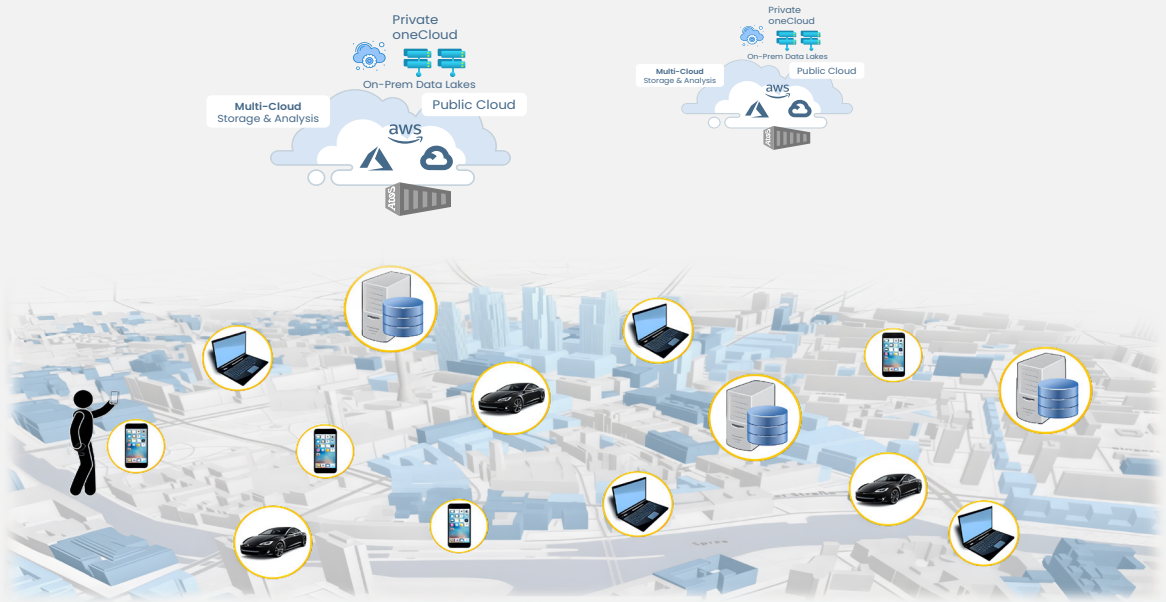


Figure 1. Typical continuum scenario.

A balanced combination of the capabilities of both technologies, cloud and edge computing, according to the specific requirements of the workloads considered, is essential for an efficient and effective management of the continuum. But this is not an easy task: several critical and complex challenges must be addressed. For instance, and just to name a few:

- The continuum is made up of a large number of heterogeneous computing devices and systems, some of which are dynamically joining and leaving the system. This requires a **sophisticated resource management** mechanism**,** maintaining up-to-date information about the resource catalog, their technological characteristics, their interconnection topology, and their real-time availability.

- Containerization is the trending technology that allows an application to run on different nodes with different architectures and operating systems. However, there are different platforms for managing containers at runtime, with Kubernetes and Docker being the most popular, but not the only existing technologies. **Transparent application execution across the continuum** requires each application to be executable on any node, regardless of the containerization technology on that node.

- This problem becomes more challenging when dealing with **multicomponent applications,** where different components of the same application might eventually be allocated on different nodes with different technologies. Each component should have the option to be offloaded to any node, and all components should be able to communicate and coordinate transparently, regardless of the local host technology.

- Data can be produced, consumed, and stored, at any node in the continuum. Furthermore, data will be live, so they could be transferred, transformed and/or replicated along the continuum, as long as privacy constraints are guaranteed. **Effective and transparent data management** mechanisms must be provided.

- In such a complex and heterogeneous scenario, deciding the optimal workload placement (scheduling) along the continuum is a major challenge. Solutions need to balance the advantages of the cloud (more capabilities) and the edge (closer data), consider the interrelationship between the different application components, and consider the flow of data between the distributed nodes during runtime. Furthermore, solutions should be dynamic (rescheduling) in order to react in case some execution issue limits the expected performance.

- In the continuum there are a large number of devices generating huge amounts of data. In this context, a large number of decisions must be constantly taken. This is a perfect scenario where intelligence might be generated and consequently used to support the **multiple decision-making processes along the continuum.** A seamless integration of an intelligence layer in the cloud continuum is of utmost importance.

- And finally, by conceiving an open, dynamic, mobile, and highly alive environment like the continuum, the door is opened to a huge span of unforeseen security risks. **A thorough security architecture design** and seamless integration with the continuum will be critical for the secure success of a metaOS in the continuum.

ICOS has been designed to facilitate the user to interact with a single system and manage their applications, regardless of where the individual workloads are being run. The user merely states what properties the components of each application need, e.g., a certain amount of compute resources, a minimum security level, or a connection to specific IoT devices, while ICOS takes care of the placement of each component as well as their interconnection even across cluster boundaries if necessary. Placing these components not only requires knowledge of the node's connected devices, their compute resource utilization, and other potentially dynamic properties but also demands the ability to foresee the changes in these dimensions after the components have been deployed. This requires ICOS to employ sophisticated matchmaking and prediction capabilities to anticipate the foreseeable future whenever components have to be placed on compute nodes. In addition, advanced features for sophisticated decision-making like user-defined policies, forecast metrics, and ML model creation are needed for the MetaOS to be able to fulfill the demanding task of application placement across such a vast number and range of hardware.

The ICOS Architecture

The ICOS MetaOS for the continuum has been designed with two main roles: Controllers and Agents. **ICOS Controllers** are at the core of system management, and have two main responsibilities: managing the resources along the continuum and providing an efficient and effective execution environment. **ICOS Agents** represent an ICOS driver that runs along the continuum and becomes the single point of management on the computing devices (nodes) attached to them.

An ICOS system requires at least one Controller to be responsible for managing the continuum and taking the runtime decisions. However, ICOS has been designed as a multi-controller system. This means that more than one Controller could cooperate in the management of ICOS. Agents are the ICOS point of management on the computing nodes (either in the cloud or at the edge).

One or more nodes are attached to an Agent, and one or more Agents are connected to a Controller. Fig. 2 shows an illustration of a typical ICOS instance. In this figure, a set of more or less powerful computing devices (nodes) can be seen at the edge, which are attached to different Agents (one Agent can be managing several nodes). In addition, one cluster in the cloud is also attached to an Agent. Finally, all agents are connected to a controller, who will be responsible for managing the continuum.

Next, the **ICOS Controller** and **ICOS Agent** are described.

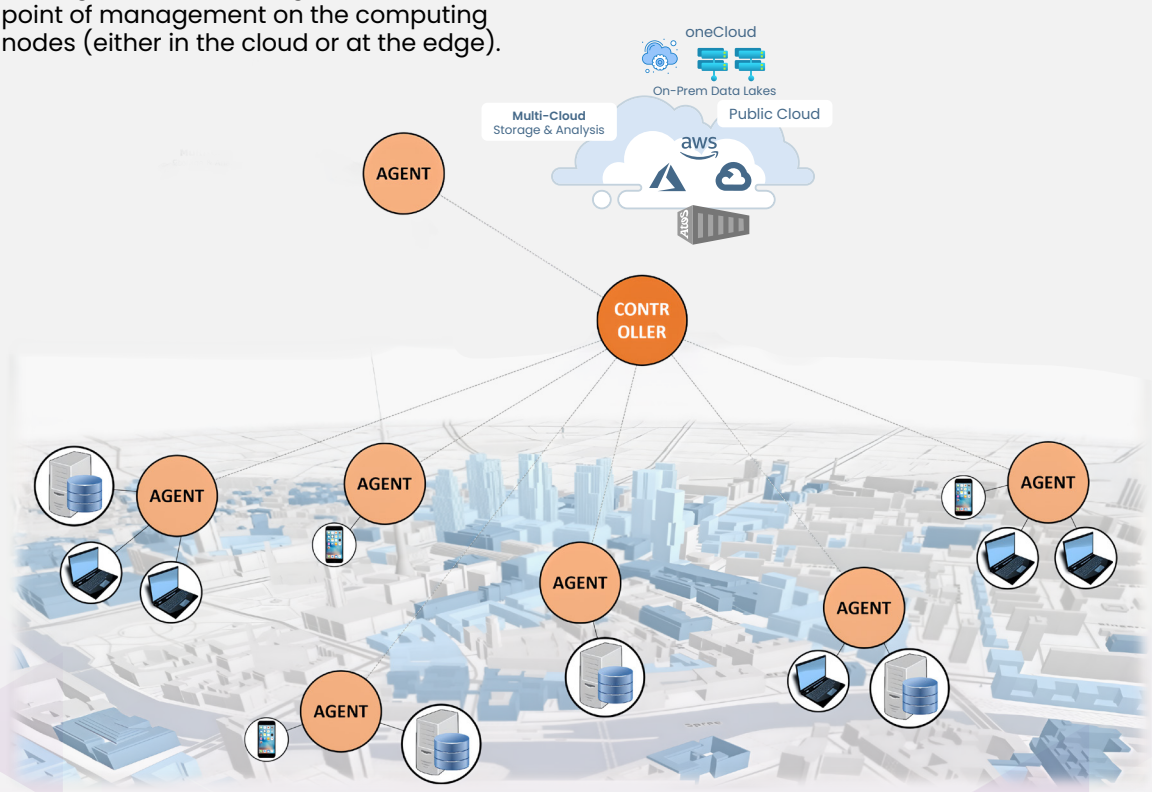


Figure 2. Sample ICOS scenario

The ICOS Controller

The ICOS Controller is designed with a three-layer architecture. As illustrated in Fig. 3, the **Meta-Kernel Layer** handles all tasks related to continuum management, as well as runtime decision-making and management. It is also responsible for collecting and storing telemetry data from the infrastructure via the Agents.

The **Intelligence Layer** processes this telemetry data, providing the Meta-Kernel Layer with the intelligence needed for both continuum and runtime management decisions. It also offers predictive monitoring to anticipate various runtime events, such as resource utilization, network load, and potential security risks. Additionally, this layer manages the training and retraining processes through federated learning

to ensure the continuous update of the intelligence model.

Lastly, the **Security Layer** focuses on identity and access management, maintaining trusted and encrypted communication channels, and detecting security-related risks, including vulnerabilities, threats, anomalies, and events such as audits.

The Meta-Kernel Layer

The **ICOS Meta-Kernel Layer** is responsible for continuum and runtime management, as well as telemetry collection and management. Fig. 4 illustrates the architecture of the ICOS Controller Meta-Kernel Layer, which is organized into three main blocks: **Continuum Management, Runtime Management,** and the **Telemetry Controller.**

The **Continuum Management** block has been designed through two components:

- **Resources Onboarding** component is a functional element responsible for facilitating the onboarding process of ICOS Agents and edge devices. The onboarding workflow of an ICOS Agent is illustrated in the diagram below.

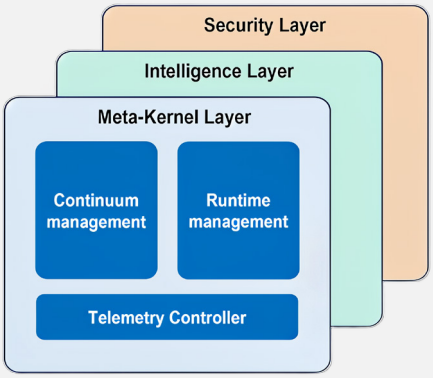


Figure 3. Three-layer architecture for the ICOS Controller

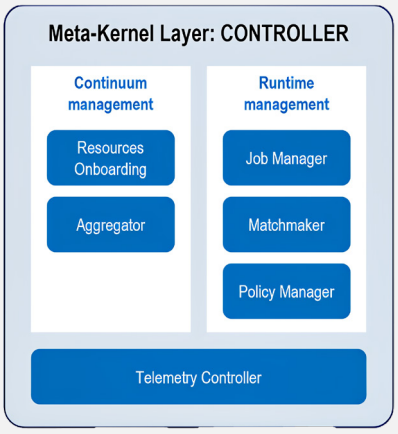


Figure 4. The ICOS Controller Meta-Kernel Layer

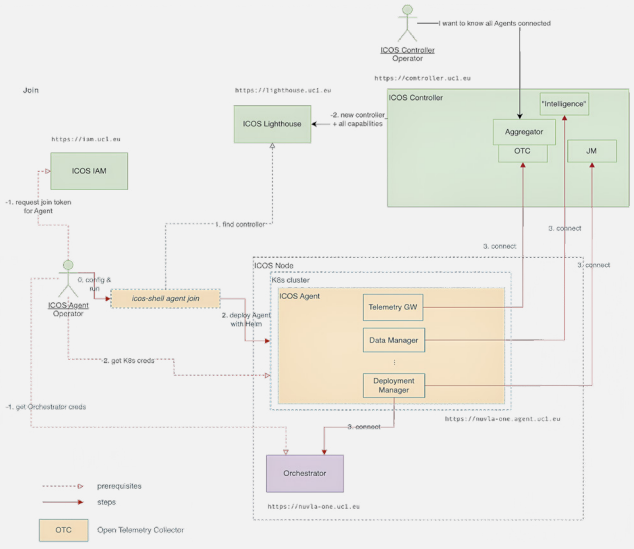


Figure 5. ICOS Agent onboarding.

The main steps of the onboarding process are as follows: the ICOS Agent Operator prepares a set of credentials to be used by the ICOS Agent for secure communication with the ICOS Controller and a specific instance of the orchestrator. The operator then executes the icos-ops agent join command using the provided configuration. The resulting Helm chart configuration parameters must subsequently be used to deploy the ICOS Agent.

The onboarding procedure for an edge device largely depends on the type of Cloud-Edge Orchestrator employed. Within the ICOS project, both Open Cluster Management (OCM) and Nuvla have been integrated as orchestrators. Each orchestrator offers a distinct onboarding mechanism for edge and cloud resources typically devices equipped with a Container Orchestration Engine (COE), such as Docker or Kubernetes, or stand-alone COE installations.

- The **Aggregator (AGR)** component provides a unified view of the infrastructure's static and dynamic characteristics across the ecosystem architecture. It collects and exposes resource information such as performance, availability, connected peripheral hardware, system metadata

and deployed applications. It also obtains forecasted metrics (CPU usage mainly) to provide enhanced view of the current and future resources utilization.

This information is structured and made accessible via a web API, which is primarily consumed by the Match Maker service to support intelligent resource allocation. It is implemented in Go and interacts directly with Thanos (the centralized telemetry hub) to retrieve real-time metrics from the ICOS ecosystem. The component's functionality and output have been validated through integration with the ICOS Testbed, ensuring alignment with the actual deployed infrastructure and seamless integration with the matchmaking service.

The **Runtime Management** block has been implemented through three components:

- The **Job Manager (JM)** is the runtime core component in charge of managing the job lifecycle. It receives requests for application execution and manages all steps to the application deployment, including finding the appropriate nodes to offload the execution and interfacing with the application deployment component. The JM keeps track of the application execution through the entire lifecycle and maintains a database with all jobs state information.

The JM is also responsible to organize any eventual task rescheduling in case of underperformance or policies violation. When the Policy Manager (described later) detects any runtime anomaly, it enforces policies by informing the JM to execute the user-defined remediation actions, such as application rescheduling or scaling up and down.

- The **Match Maker (MM)** is the component responsible to find the most appropriate nodes along the continuum to execute an application. It is triggered by the JM upon an execution request and returns the optimal <app-nodes> mapping to the same JM. In order to retrieve the available resources and topology, the MM requests a snapshot of the continuum to the AGR.

An application execution request is configured through an **App Descriptor** manifest, a user defined yaml file that describes the components of the application. Components are workload units that can be executed in any node, and must be containerized to facilitate offloading to any node in the continuum. Virtualization technologies currently considered in ICOS are kubernetes and docker compose.

App descriptors contain the following information:

  - Component requirements, including hardware features (RAM, CPU architecture, GPU, ...) and I/O requirements (data volumes, IoT devices, ...). Components could be forced to be allocated into a specific node through a node-label constraint.
  - Component policies allow to specify objective metrics used for optimal mapping, which can be based on performance, power consumption, node security, or a combination of these. Policies can also be dynamic and flexible user defined rules that influence application deployment and their behavior.
  - Component and inter-component dependencies description, to provide knowledge of the communication flow between components and help configuring the intercluster application execution through ClusterLink.

The MM can also handle application redeployment. When the Policy Manager (next component) detects underperformance or policies violation, the anomaly can be remediated by migrating the application into a more suitable node.

- The **Policy Manager** (also known as the **Dynamic Policies Manager, DPM**) is a key ICOS component that ensures the system meets desired efficiency, performance, and security thresholds. It works throughout the entire application lifecycle—from deployment to decommissioning—enforcing policies consistency at each stage. The key features include:
  - Flexible Policy Model: Allows for the definition of various policies related to performance, security, and efficiency for both applications and infrastructures.
  - Continuous Monitoring: The system is constantly monitored through the ICOS telemetry framework.
  - Non-conformance Detection: Identifies deviations from set standards and triggers notifications for corrective actions.
  - Policy Enforcement: Enables automatic corrective measures to restore or optimize performance.

A key strength of the DPM is its tight integration with the ICOS telemetry framework. Policies are translated into queries for metrics within the telemetry system, allowing the DPM to continuously evaluate policy enforcement with up-to-date data and quickly react to violations.

And finally, the **Telemetry Controller (TC)**, implements the cloud-native monitoring and observability strategies for ICOS. It features a highly distributed architecture, which scales in accordance with the distribution of ICOS resources. As depicted in Fig. 6, the module consists of three distinct sub-modules, each

playing a specific role and requiring deployment in different parts of the system:

- **Telemetry Controller:** Deployed in the ICOS Controller, this sub-module receives and aggregates data (metrics and logs) from all the nodes registered with that controller. It is responsible for the long-term storage of these metrics and provides tools for analyzing the collected data.
- **Telemetry Agent.** It is deployed in the computational resources (e.g., edge devices, clusters) available to ICOS. This component is responsible for the collection of metrics and logs.
- **Telemetry Gateway.** It is deployed in the ICOS Agent and is responsible for collecting all metrics and logs for that agent.

The Telemetry Agent and the Telemetry Gateway are described in more detail in section 3.2 as part of the ICOS Agent description.

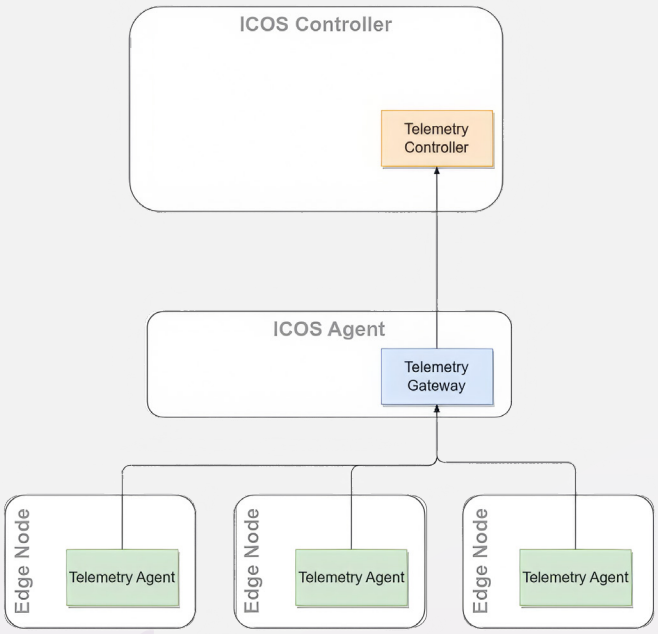


Figure 6. Telemetry Controller architecture.

The TC serves as the primary data source for other ICOS components within the Controller, providing insights into the behavior of ICOS infrastructure and applications. Specifically:

- The Aggregator queries the TC to build a system topology, which includes the characteristics and relationships of available nodes. This topology is fundamental for the matchmaking and orchestration mechanisms within ICOS.
- The Policy Manager queries the TC to assess the enforcement of active policies.
- The ICOS Shell can submit queries to the TC on behalf of the user to generate charts and dashboards for visualization.

The Intelligence Layer

The ICOS Intelligence Layer provides advanced AI-driven capabilities across the cloud continuum. It consists of five main modules: (i) Intelligence Layer Coordination, (ii) AI Analytics, (iii) Data Processing, (iv) Trustworthy AI, and (v) the AI Models and Data Repository. Each module ensures the seamless integration of data pipelines, model training, and inference, ultimately enabling proactive and optimized resource usage within ICOS.

- The **Intelligence Layer Coordination** orchestrates the end-to-end AI lifecycle. It exposes two core components: (i) a frontend (Export Metrics API), which receives requests from the ICOS Shell to generate forecasts and train new models; and (ii) a backend (Intelligence API), which manages the model registry, handles versioning, and coordinates training jobs and inference tasks. This coordination ensures that model updates, metadata, and telemetry are consistently integrated with other ICOS components.
- The **AI Analytics** module is responsible for model training, inference, and performance optimization. It supports both univariate and multivariate time-series forecasting, enabling simultaneous predictions of metrics like CPU and memory usage. The module also features model compression techniques - quantization and knowledge distillation - to reduce computational overhead, while MLFlow integration facilitates experiment tracking and advanced model performance analysis.
- The **Data Processing** module offers streamlined data pipelines to access distributed datasets and offload large-scale training across the ICOS infrastructure. It interacts with Data Management to seamlessly integrate stored and real-time data, ensuring that the Intelligence Layer can efficiently handle diverse data sources, including those needed for federated learning scenarios.
- The **Trustworthy AI** module enhances transparency, reliability, and privacy. It integrates SHAP-based explainability to illustrate how models reach their predictions, supports federated learning for privacy-aware training, and logs confidence intervals in model outputs. By continuously monitoring model behavior and performance, it ensures that decision-making in the Intelligence Layer remains robust and trustworthy.
- The **AI Models and Data Repository**, hosted outside the ICOS Controller, serves as the primary store of pre-trained models and associated datasets. It supports model sharing, discovery, and versioning, facilitating collaboration among developers and enabling the Intelligence Layer Coordination module to import and export models for broader use within the ICOS ecosystem.

The Security Layer

The **ICOS Security Layer** is responsible for guaranteeing the security of ICOS users, resources, and applications at all times. It includes modules for authentication and authorization operations in the system, assessing the security of resources

and applications and suggesting remediation or mitigation actions, proactive discovery of anomalous behaviours and security-sensitive events, and verification of the compliance of resources and applications. Trust (identity validation) and Privacy (anonymization and encryption) are included as architecture-wide functionalities.

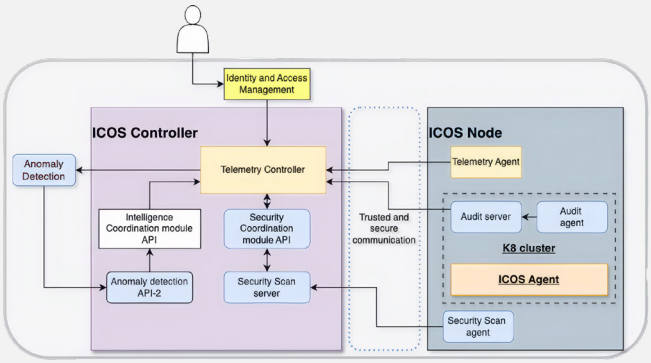


Figure 7. Security layer components and functionalities in ICOS

- The **Security Layer Coordination** module API acts as a reverse proxy service to forward requests from the Telemetry for the security scans performed by the Security Scan module sends Security Scan results as security metrics back to the Telemetry Controller.
- The **Security Scan** module is delegated to actively check for security issues on the ICOS nodes and report these issues to the Telemetry Controller and to the ICOS user, as well as to check the security of deployed ICOS modules and components. It integrates the Wazuh security platform to detect vulnerabilities (CVE), misconfigurations (CIS benchmark) and malware on the ICOS nodes. For checking the ICOS modules and components, it uses Trivy, a container image scanner that scans images for vulnerabilities (CVE), IaC issues and misconfigurations and SBOM (host OS packages and software dependencies). Trivy 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.
- **Identity and Access Management** 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) or other services (part of ICOS or external). The resources to be protected are the ICOS services. It has three keys' functions: i) Management of Identities permissions, ii) Authentication of users, iii) Authorisation of requests. It has been implemented using Keycloak, an open-source IAM software that supports OIDC4 and OAuth 2.0 protocols.
- **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 deep learning methods, such as Mask Language Modelling, common in self-supervised NLP, and Hypersphere Volume Minimisation, to model log streams and capture their normal operating conditions. Without manual pre-processing of raw logs from unstructured data, LOMOS is able to identify patterns in logs and identify anomalous behaviour, including potential security threats.
- 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.

  Trivy provides a first layer of auditing on a given application by analysing an application 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.

  Tetragon is able to perform real time monitoring on events emitted by the kernel using eBPF (Extended Berkeley Packet Filter) 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.
- **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 Cilium as 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 (step-ca as self-signed Certificate Authority) and (m)TLS.
  - In identification and encrypted communication between ICOS users and ICOS components using VPN technologies (e.g., Wireguard).

The ICOS Agent

The ICOS Agents implements an ICOS driver that runs along the continuum and becomes the single point of management on the computing devices (nodes) attached to them. Fig. 8 illustrates the architecture of the ICOS Agent.

The **Deployment Manager (DM)** is a component responsible for executing user-defined application deployment tasks on target Container Orchestration Engine (COE) clusters. It serves as a bridge between the ICOS Job Manager and the underlying orchestrators that manage collections of edge and cloud resources.

The ICOS project integrates two orchestrators —Open Cluster Management (OCM) and Nuvla— each requiring a dedicated implementation of the Deployment Manager. As a result, two distinct versions of the DM have been developed to interface with their respective orchestrators. The example below on Figure 8 illustrates the DM's operation with the Nuvla orchestrator. Deployment-related tasks (such as create, get, or terminate) are initiated by the user and processed by the Job Manager, which transforms them into executable jobs. The DM is then responsible for forwarding these tasks to the orchestrator, monitoring their execution, and reporting the status and results back to the Job Manager.

The **App Setup Manager (ASM)** is a component that enables applications to become aware of changes in their deployment. Thus, applications can react to them and their containers can adapt their configuration accordingly. To facilitate this, the ASM uses a messaging bus to broadcast information about all containers deployed for an application, including their locations.

Applications that wish to leverage this feature of the ICOS meta-OS can subscribe to specific topics on the bus. Each component's topology is associated with a distinct topic, enabling fine-grained control over the information an application receives. For example, containers supporting a component can be notified only about changes in other containers belonging to the same component. This is exactly the case for applications building on the distributed and parallel execution (D&PE) component.

This mechanism is particularly useful for applications building on the Distributed and Parallel Execution (D&PE) component. To distribute computational workloads across multiple replicas, the D&PE runtime in each container dynamically reconfigures itself, adapting to the current set of available

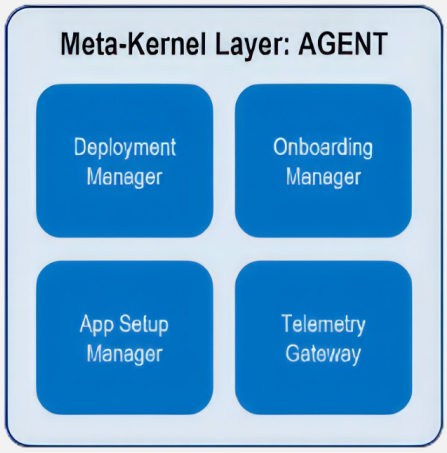


Figure 8. Architecture of the ICOS Agent

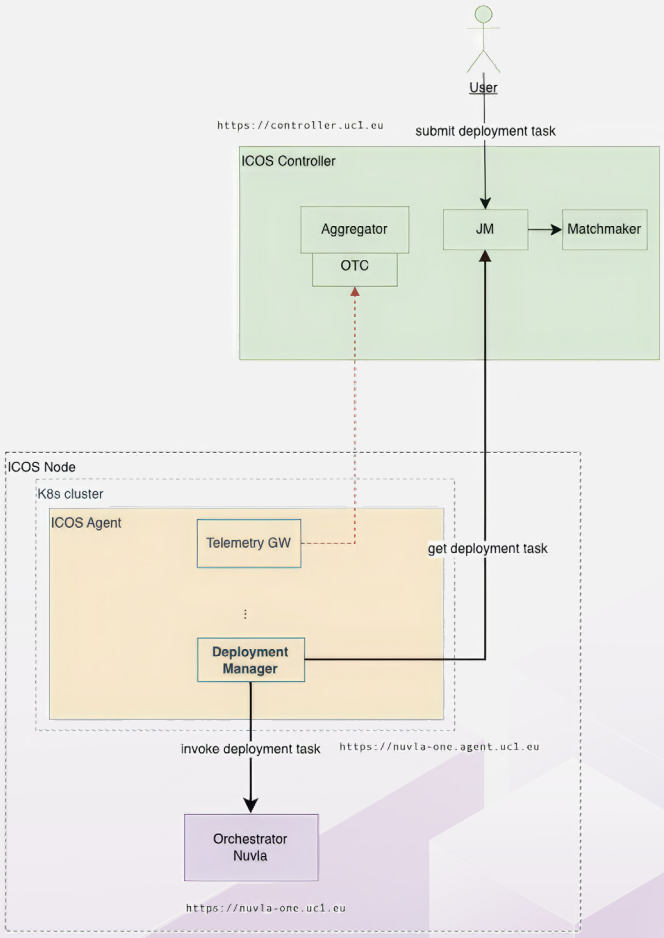


Figure 9. Deployment Manager acting as a connector between the ICOS Controller (Job Manager) and the cloud-edge orchestrator (example: Nuvla)

resources where it can offload parts of the workload effectively.

As anticipated in the Meta-Kernel Layer section, at the ICOS Agent level, telemetry data is collected and processed by two sub-modules: the Telemetry Agent and the Telemetry Gateway.

- The **Telemetry Agent**, deployed on computing resources like edge devices or virtual machines, collects metrics and logs directly from the source. It monitors system performance (e.g. CPU, memory, network), energy efficiency (e.g., power consumption of nodes and/or individual processes), and security status (e.g., security assessment results, audit logs, and other security-related information) using custom modules and third-party plugins. It's highly configurable to suit diverse environments, including low-resource ICOS devices. The **Telemetry Agent** does not store data but forwards it to the **Telemetry Gateway.**

- The **Telemetry Gateway (TG)** is deployed on the ICOS Agent and has been introduced in the ICOS Architecture primarily to enable the collection of telemetry data in the ICOS Agent, address networking issues (e.g., edge devices being unable to reach the Telemetry Controller directly due to firewalls or NAT), and aggregate and optimize the processing of telemetry data (e.g., discarding unnecessary metrics and reducing the amount of data sent to the controller).

Data Management

Data Management is a challenging aspect of the ICOS project due to the distributed nature of the computing continuum. Being able to address the data requirements of ICOS components and ensuring a smooth operation across the continuum lifecycle are key aspects of the stability and scalability of ICOS.

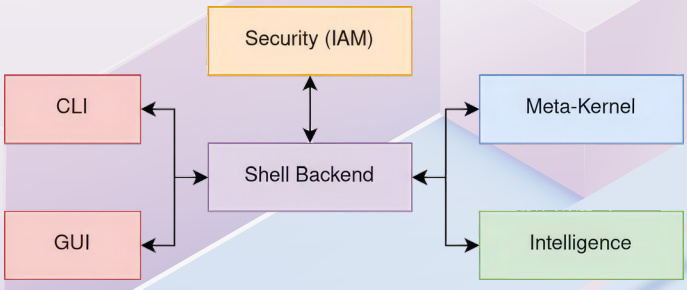
There are several architectural decisions on the Data Management that have been key for the successful implementation of the ICOS project:

- A horizontal **bus** across the computing devices. This bus is key to offer an up-to-date view of the topology of applications. User applications are able to query and leverage the distributed nature from their application deployments (e.g. for coordinating parallel and distributed workloads).

- An **distributed active storage system** to be used by the Intelligence layer. This storage system's goal is to solve all the requirements related to data and computation from the intelligence workloads —in the context of a distributed edge-cloud environment. The use of an **active** system allows to perform task offloading, making sure that available resources are used in an efficient manner. Resources will vary between computing devices; the task offloading mechanism addresses this heterogeneity and results in faster and more efficient intelligence workloads. The distributed nature of the storage is key for aggregating results being generated in a distributed environment, and that is also leveraged during the federated learning procedures. Data is being generated continuously (metrics, telemetry, etc.). First, data enters the distributed storage and is held distributed across the continuum. After that, federated learning takes advantage of the data distribution and the system performs a federated learning process.

Using ICOS

The user interface - **the ICOS shell** - consists of two separate frontends; a **command line interface (CLI)** as well as a **graphical user interface (GUI)**. In its current version, the CLI implements all functionalities supported by the backend, while the GUI is actively being worked on. The ICOS shell backend serves as the entry point to the ICOS system and is the user's portal to interact with ICOS controllers. It accepts authenticated requests against its RESTful API and forwards them to the respective component in the intelligence layer, meta-kernel layer or security layer. Since the backend is stateless, all states, for example authentication tokens, are managed by the CLI or GUI.



CLI

The ICOS CLI currently comes in several flavors:

- linux-amd64: a version compatible with most desktop PCs and laptops.
- linux-arm64: a version for mobile and other lightweight devices.
- darwin-arm64: a version for M1/2/3/4 Mac(book)s.
- darwin-amd64: a version for older (pre-M) Mac(book)s.

We provide the newest versions on our Github release page where they can be downloaded. The matching binary can then be executed on any supported machine in conjunction with a config file that specifies at least the lighthouse of an ICOS installation and a valid username and password combination.

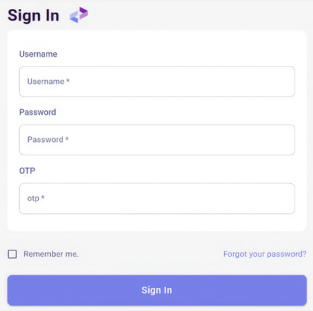
Upon first login, the list of controllers is retrieved from the lighthouse and provided to the user who can then select the controller they want to use. After this selection process, the setting is stored and used for all further commands until it is manually removed from the file. The CLI furthermore supports the use of OTP tokens for multi-factor authentication. The validity duration of an authenticated session depends on the specific keycloak settings and once the token expires, the user will have to log in again.

Once the login process is complete, other commands can be issued, for example to create, list or delete deployments, to show all devices connected to the system, or to interact with AI models to have the system predict future values of metrics. The CLI furthermore offers a --help parameter that provides explanations for the different options. The available commands allow the user to:

- Log in to and out off the system.
- Create, list, delete and update deployments together with their policies.
- Create, list and delete metrics models.
- List all resources connected to ICOS.
- List and add controllers to the lighthouse (intended for debugging only).

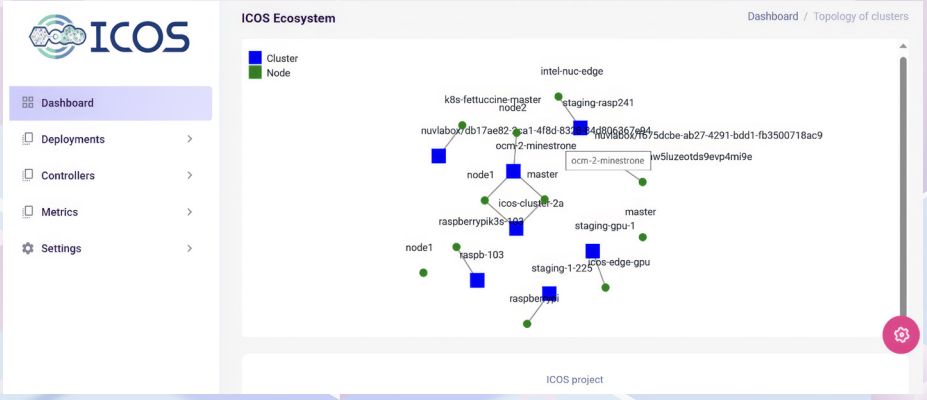
GUI

The Graphical User Interface (GUI) component of the project serves as a web-based platform designed to provide users with a simple and intuitive way to interact with the system. Developed using React.js, the interface ensures responsiveness, scalability, and maintainability. It communicates with the backend through RESTful APIs, enabling seamless access to system features and real-time data interactions. The primary goal of this module is to enhance user experience by making the system accessible, efficient, and user-friendly across various devices and platforms.



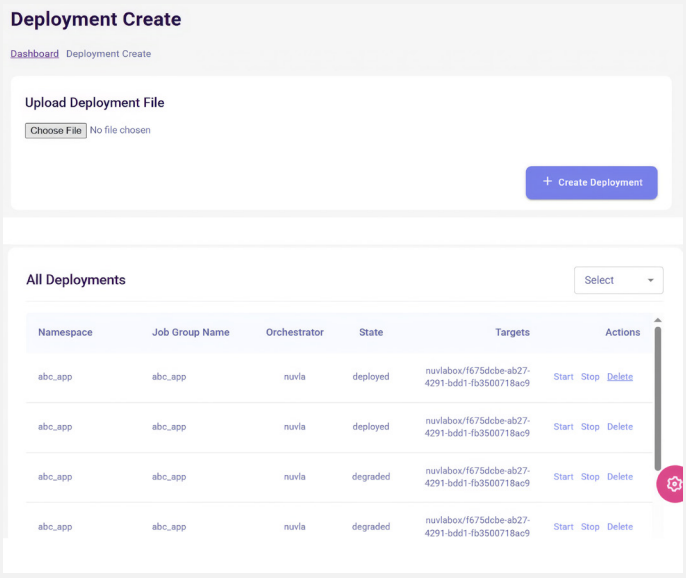
Users can easily log in to the GUI using their credentials along with a one-time password (OTP), ensuring both convenience and secure access to the system.

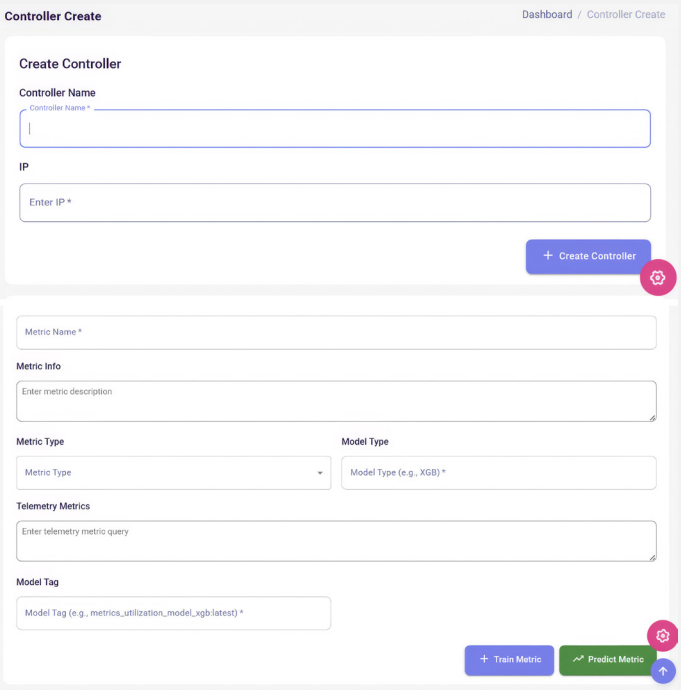
A visual representation of the current ICOS ecosystem is displayed as a scrollable, zoomable graph on the index page.



The GUI allows users to create a deployment by uploading a .yaml file with a predefined structure. Existing deployments are listed in the GUI and can be started, stopped, or updated through the interface.

Existing deployments are listed in the GUI and can be started, stopped, or updated through the interface. Additionally, it is possible to add a controller to the Lighthouse from the GUI.





ICOS

Atos

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

CeADAR
Ireland's Centre for Applied AI

zetta scale

PSNC

DEMOKRITOS

sixsq

HELLENIC REPUBLIC
National and Kapodistrian University of Athens
EST. 1837

FGC
Ferrocarrils de la Generalitat de Catalunya

ENGINEERING
THE DIGITAL TRANSFORMATION COMPANY

Suite5
We Deliver Intelligence

STELLANTIS

CRF

Technische Universität Braunschweig

UPC

sse Airtricity

Łukasiewicz
Centrum

XLAB

icos-project.eu
icos_project
icos_project
@icos_project

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