This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101069732

aerOS

EUROPEAN IOT-EDGE-CLOUD

D2.7 – aerOS architecture definition (2)

Deliverable No.	**D2.7**	**Due Date**	**31-05-2024**
Type	**R – Document, Report**	**Dissemination Level**	**Public**
Version	**1.0**	**WP**	**WP2**
Description	aerOS final architecture design, technical components identification and description.		

Copyright

The aerOS consortium consists of the following 27 partners:

UNIVERSITAT POLITÈCNICA DE VALÈNCIA	ES
NATIONAL CENTER FOR SCIENTIFIC RESEARCH "DEMOKRITOS"	EL
ASOCIACION DE EMPRESAS TECNOLOGICAS INNOVALIA	ES
TTCONTROL GMBH	AT
TTTECH COMPUTERTECHNIK AG (*third linked party*)	AT
SIEMENS AKTIENGESELLSCHAFT	DE
FIWARE FOUNDATION EV	DE
TELEFONICA INVESTIGACION Y DESARROLLO SA	ES
COSMOTE KINITES TILEPIKOINONIES AE	EL
EIGHT BELLS LTD	CY
INQBIT INNOVATIONS SRL	RO
FOGUS INNOVATIONS & SERVICES P.C.	EL
L.M. ERICSSON LIMITED	IE
SYSTEMS RESEARCH INSTITUTE OF THE POLISH ACADEMY OF SCIENCES IBS PAN	PL
ICTFICIAL OY	FI
INFOLYSIS P.C.	EL
PRODEVELOP SL	ES
EUROGATE CONTAINER TERMINAL LIMASSOL LIMITED	CY
TECHNOLOGIKO PANEPISTIMIO KYPROU	CY
DS TECH SRL	IT
GRUPO S 21SEC GESTION SA	ES
JOHN DEERE GMBH & CO. KG*JD	DE
CLOUDFERRO S.A.	PL
ELECTRUM SP ZOO	PL
POLITECNICO DI MILANO	IT
MADE SCARL	IT
NAVARRA DE SERVICIOS Y TECNOLOGIAS SA	ES
SWITZERLAND INNOVATION PARK BIEL/BIENNE AG	CH

Disclaimer

Authors

Name	Partner	e-mail
Carlos E. Palau	P01 UPV	cpalau@dcom.upv.es
Ignacio Lacalle	P01 UPV	iglaub@upv.es
Rafael Vaño	P01 UPV	ravagar2@upv.es
Andreu Belsa	P01 UPV	anbelpel@upv.es
Salvador Cuñat	P01 UPV	salcuane@upv.es
Raúl San Julián	P01 UPV	rausanga@upv.es
Raul Reinosa	P01 UPV	rreisim@upv.es
Harilaos Koumaras	P02 NCSRD	koumaras@iit.demokritos.gr
Vasilis Pitsilis	P02 NCSRD	vpitsilis@iit.demokritos.gr
Thanos Papakyriakou	P02 NCSRD	thpap@iit.demokritos.gr
Spyros Georgoulas	P02 NCSRD	spygeorgoulas@iit.demokritos.gr
Andreas Sakellaropoulos	P02 NCSRD	asakellaropoulos@iit.demokritos.gr
Roger Briz	P03 INNOVALIA	rbriz@innovalia.org
Andreas Locatelli	P04 TTC	andreas.locatelli@ttcontrol.com
Jan Ruh	P04.1 TCAG	jan.ruh@tttech.com
Anna Ryabokon	P04.1 TCAG	anna.ryabokon@tttech.com
Philippe Buschmann	P05 Siemens	philippe.buschmann@siemens.com
José Fontalvo-Hernández	P05 Siemens	jose-eduardo.fontalvo-hernandez@siemens.com
Korbinian Pfab	P05 Siemens	korbinian.pfab@siemens.com
Renzo Bazan	P05 Siemens	renzo.bazan.ext@siemens.com
Amparo Sancho Arellano	P05 Siemens	amparo.sancho-arellano@siemens.com
Ken Zangelin	P06 FIWARE Foundation	ken.zangelin@fiware.org
Ignacio Dominguez Martinez-Casanueva	P07 TID	ignacio.dominguezmartinez@telefonica.com
Lucia Cabanillas Rodriguez	P07 TID	lucia.cabanillasrodriguez@telefonica.com
Fofy Setaki	P08 COSM	fsetaki@cosmote.gr
George Lyberopoulos	P08 COSM	glimperop@cosmote.gr
Ioannis Chouchoulis	P10 IQB	giannis.chouchoulis@inqbit.io
Ioannis Makropodis	P10 IQB	giannis.makropodis@inqbit.io

Vasiliki Maria Sampazioti	P10 IQB	vasiliki.maria.sampazioti@inqbit.io
Katerina Giannopoulou	P11 FOGUS	kgiannopoulou@fogus.gr
Joseph McNamara	P12 L.M. ERICSSON LIMITED	joseph.mcnamara@ericsson.com
Katarzyna Wasielewska-Michniewska	P13 IBSPAN	katarzyna.wasielewska@ibspan.waw.pl
Przemysław Hołda	P13 IBSPAN	przemyslaw.holda@ibspan.waw.pl
Wiesław Pawłowski	P13 IBSPAN	wieslaw.pawlowski@ibspan.waw.pl
Amine Taleb	P14 ICTFI	amine.taleb@ictficial.com
Tarik Zakaria Benmerar	P14 ICTFI	tarik.benmerar@ictficial.com
Tarik Taleb	P14 ICTFI	tarik.taleb@ictficial.com
Vaios Koumaras	P15 INF	vkoumaras@infolysis.gr
Nikolaos Gkatzios	P15 INF	ngkatzios@infolysis.gr
Eugenia Vergi	P15 INF	evergis@infolysis.gr
Eduardo Garro Crevillen	P16 PRO	egarro@prodevelop.es
Alvaro Martínez Romero	P16 PRO	amromero@prodevelop.es
Kyriacos Orphanides	P17 ECTL	kyriacos.orphanides@eurogate-limassol.com
Alessandro Cassera	P17 ECTL	alessandro.cassera@eurogate-limassol.com
Jon Egaña	P20 S21SEC	jegana@s21sec.com
Oscar Lopez	P20 S21SEC	olopez@s21sec.com
Alexander Wagner	P21 JD	wagneralexander2@johndeere.com
Artur Bargiel	P22 CF	abargiel@cloudferro.com
Danish Abbas Syed	P24 POLIMI	danishabbash.syed@polimi.it
Francesco Dellino	P25 MADE	francesco.dellino@made-cc.eu
Lucie Stutz	P27 SIPBB	lucie.stutz@sipbb.ch

History

Date	Version	Change
15 April 2024	0.0	Initial planning and timeline, WP2 general meeting announced
22 April 2024	0.1	Initial ToC, sections' structure, and assignments
24 April 2024	0.2	Final ToC, sections' structure, and assignments
15 May 2024	0.3	First round of contributions
22 May 2024	0.4	Second round of contributions

24 May 2024	0.5	Submitted for internal review
30 May 2024	0.6	Address internal reviewers' comments
31 May 2024	1.0	Deliverable submitted

Key Data

Keywords	IoT, aerOS, meta operating system, continuum, network & compute fabric, service fabric, data fabric, aerOS knowledge graph, aerOS distributed state repository, architecture, federation, orchestration, federated orchestration, aerOS Infrastructure Element, aerOS Domain
Lead Editor	P02 NCSRD, Vasilis Pitsilis
Internal Reviewer(s)	P25 MADE, Carlo Ongini P26 NASERTIC, Daniel Cobo Boregga

Executive Summary

This document, the second iteration of the aerOS architectural design, delivers the finalized reference architecture, detailing all functional and technical concepts and components. Building upon the groundwork established in the initial deliverable, this document not only presents enhancements and practical implementations of the aerOS framework but also guides further instantiation and deployment for all use cases in WP5 and through subsequent open calls. Additionally, it provides a refined prototype that serves as a robust reference for future implementations beyond the project's conclusion.

To develop the comprehensive aerOS architecture, we adopted and refined a robust methodology throughout the project's lifespan. This approach effectively processed incoming requirements, identified key stakeholders, mapped critical concerns and flows, defined architectural components and their interactions, and instructed deployments while incorporating feedback. These steps were fundamental to ensuring both the feasibility and efficiency of our architectural goals.

Reflecting on the rationale for an IoT-Edge-Cloud continuum, this deliverable moves beyond identifying deficiencies in the current landscape. As of today, IoT developers were limited in their ability to leverage distributed capabilities across the continuum and lacked a common execution environment supportive of IoT service deployment and reuse. Building upon aerOS reference architecture, the project has successfully demonstrated how IoT developers are enabled to leverage distributed capabilities across the continuum and benefit from a common execution environment. This shift from isolated resource usage to a unified compute and network fabric has been realized, providing a cohesive orchestration and execution environment—now evident in the implemented service fabric. The means to provide the most efficient and smart orchestration of underlying resources is based on an innovative data fabric implementation, which is thoroughly discussed along with computing and service fabric.

As a Meta-OS, aerOS manages and orchestrates underlying fabrics, presenting a seamless continuum of compute, network, and service resources. This advancement offers IoT developers a streamlined service deployment experience, successfully transitioning a theoretical concept into practical application.

The basic concepts and innovations of aerOS as a Meta-OS are presented. Federation of distributed resources is the basis for a smart orchestration that can span across several administrative domains. The technologies that make possible the federation of heterogeneous (hardware and software) and scattered resources are explained. An innovative orchestration architecture, which separates smart-enabled decision layer from enforcement layer is introduced. The combined activity of federation and orchestration across all domains ensures the most efficient usage of resources and the optimal placement of IoT applications, and this is documented, within this deliverable, along with the supporting innovative data fabric mechanisms.

The infrastructural components enabling the implementation of these concepts are detailed, along with the processes for integrating any compute or network resource as an aerOS element. The capabilities of aerOS domains and the essential services running within each domain are demonstrated, ensuring clarity on the operational aspects of aerOS.

This document aims to provide stakeholders with a clear understanding of the innovations introduced by aerOS that support IoT development across the cloud-edge continuum, the benefits of these enhancements, and the processes involved in transitioning legacy compute and network resources to aerOS elements. Furthermore, it outlines the procedures for deploying IoT services within the aerOS ecosystem, reflecting the practical achievements and readiness of aerOS for widespread adoption.

Table of contents

List of tables

List of figures

List of acronyms

Acronym	Explanation
AAA	Authentication, Authorization and Access
AGV	Automated Guided Vehicles
CB	Context Broker
CEI	CloudEdgeIoT
CMF	Container Management Framework
CNF	Cloud Native Function
CNCF	Cloud Native Computing Foundation
CNI	Container Network Interface
CR	Custom Resource
CRI	Container Runtime Interface
CRD	Custom Resource Definition
CSI	Container Storage Interface
CSR	Context Source Registration
DSNB	Distributed State Network of Brokers
DSR	Distributed State Repository
FaaS	Function-as-a-Service
FQDN	Full Qualified Domain Name
HLO	High-Level Orchestrator
IE	Infrastructure Element
IaaS	Infrastructure-as-a-Service
IoT	Internet of Things
JSON-LD	JavaScript Object Notation for Linked Data
K8s	Kubernetes
LCM	Life Cycle management
LLO	Low-Level Orchestrator
MANO	Management and Orchestration
MEC	Mobile Edge Computing
Meta-OS	Meta Operating System
MVP	Minimum Viable Product
NFV	Network Function Virtualization
NGSI-LD	NextGeneration Service Interface – Linked Data
OAuth	Open Authorization
OIDC	OpenID Connect

OPC UA	Open Platform Communications Unified Architecture
PaaS	Platform-as-a-Service
Protobuf	Protocol Buffers
RA	Reference Architecture
RDF	Resource Description Framework
REST	Representational State Transfer
RPi	Raspberry Pi
SDN	Software Defined Network
SPARQL	SPARQL Protocol and RDF Query Language
TOSCA	Topology and Orchestration Specification for Cloud Applications
URL	Uniform Resource Locator
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VPN	Virtual Private Network

1. About this document

The primary aim of this deliverable is to offer a comprehensive overview of the architecture that underpins aerOS as a Meta-OS instantiation, reflecting the culmination of developments and enhancements made since the initial version. This document delves deep into the system's structure, elucidating its design principles, components, and their interplay. It delineates the high-level vision and objectives, introduces fundamental architectural concepts, and details how various elements synergize to achieve the desired functionality and performance across the continuum.

Furthermore, the document outlines the technology stack, data flow, communication protocols, and key integration points, providing a definitive roadmap for both the development and deployment of the system. By capturing all essential architectural aspects, this deliverable aims to serve as a definitive reference for stakeholders, engineers, and decision-makers. It is designed to ensure a unified understanding of the system's design and to guide its successful implementation and ongoing evolution.

As the second and final document in this series, it integrates all advancements since the first deliverable, based on the deployment of Minimum Viable Product (MVP), integration of Pilots' architecture design, and precise component implementation details emerging from the activities of WP3 and WP4. These elements have been essential in refining and finalizing an efficient and accurate architecture for aerOS, tailored to meet both current and anticipated future needs.

1.1. Deliverable context

Item	Description
Objectives	**O1** (Design, implementation, and validation of aerOS for optimal orchestration): Design a Meta-OS approach, based on open sources components, for efficient resource provisioning and services orchestration on heterogeneous nodes across the IoT-edge-cloud continuum. **O2** (Intelligent realization of smart network functions for aerOS): Design networking integration and components development to support programmable functions, service mesh methods, and secure communication channels between distributed resources. Design for industry IoT communication technologies and protocols integration. **O3** (Definition and implementation of decentralized security, privacy, and trust): Design a holistic cross layer solution for cybersecurity and federated and distributed data governance. Design for dedicated components seamless integration, with aerOS services, for cybersecurity, privacy, and trust.
Work plan	D2.7 receives input from • T2.1 (state-of-the-art): Novel components and technologies research for further design choices. • T2.2 (use cases and requirements): Receives requirements to drive architecture building and components design. To be evaluated and fulfilled with the proposed architecture blueprint. D2.7 defines WP5 process as it guides: • T5.2 which undertakes the implementation in vertical industry pilots based on the produced architecture blueprint. D2.7 establishes a close collaboration with: • WP3, in the way to develop, deploy, and connect infrastructure components to support continuum implementation as a product of network and compute fabric and

	service fabric orchestration. • WP4, to employ data fabric features to optimize usage of data based on data autonomy, interoperability governance and provide data as a product to AI consumers.
Milestones	This deliverable accomplishes the realisation of *MS5 – Final architecture defined*, achieved in M21, and contributes to *MS6 – Final integrated software solution* that will be achieved in M24 and to the realisation of *MS7 – Final Software Components released*, that will be achieved in M30.
Deliverables	This deliverable is part of an iteration of living deliverables. The first version is in M12 and the second in M21. This deliverable receives inputs from "D2.6 aerOS architecture definition (1)", which is the initial version, and which indirectly integrates inputs from "D2.1 State-of-the-Art and market analysis report", "D2.2 Use cases manual, requirements, legal and regulatory analysis (1)". Additionally, it receives input from "D2.3 Use cases manual, requirements, legal and regulatory analysis (2)" and "D5.2 Integration, evaluation plan and KPIs definition (2)". It is expected to support final technical deliverables versions "D3.3 Final distributed compute infrastructure" and "D4.3 Software for delivering intelligence at the edge final release" and "D5.4 Use cases deployment and implementation (2)".

1.2. The rationale behind the structure

The content of the deliverable is organised in eight main sections, that can efficiently present T2.5 and explain the architectural structure of aerOS as a Meta-OS.

- Section 1. Provides the overview, context, and structure of this deliverable.
- Section 2. Provides the methodology and the standards consultation used to provide a solid and structured architectural design approach.
- Section 3. Provides insights on the feedback received since the initial version of the deliverable, the received validation of the design and the motivation behind any adaptations towards finalizing the architecture.
- Section 4. Provides the status, as of today, across the IoT edge to cloud systems and the emerging needs that guide the transition to a continuum across the path from edge IoT devices to cloud resources. It highlights the fragmented nature of computing and network resources, of services deployment and the data heterogeneity and the need to move towards unified fabrics to proceed from monolithic applications to intelligently distributed services across the continuum. aerOS approach, as a Meta-OS, to enable and orchestrate the continuum that integrates all the fabrics is presented in detail.
- Section 5. This section introduces the basic concepts and the vision that aerOS introduces as a Meta-OS across the continuum. The implementing building blocks that aerOS employs towards continuum establishment and operation. Architectural decisions and their significance and consequences are detailed. Technologies employed towards a federated orchestration of distributed resources are explained. The basic components serving as building blocks undertaking aerOS services deployment are exposed and aerOS services are described.
- Section 6. Building on the foundational concepts and components introduced earlier, this section offers a comprehensive exploration of the aerOS reference architecture through multiple viewpoints. It systematically delineates the roles and functionalities of each component, clarifies their interconnections, and maps their interactions. This structured analysis ensures each component's placement and interaction are optimally aligned with achieving the operational goals of aerOS. The section breaks down into high-level, functional, process, deployment, and business views, each providing targeted insights to inform and guide effective system implementation.
- Section 7. This section examines the practical applications and efficacy assessments of the architecture. It details the, MVP based, aerOS demonstrator as an applied IoT service orchestration over the continuum, describes the mapping of aerOS architecture concepts in each of the pilots which guides

components integrations, demonstrating practical applications and adaptability and finally discusses how aerOS aligns with and supports European strategic goals, ensuring compliance and relevance in broader initiatives.

- Section 8. Concludes the document by summarizing the key findings and achievements of the prototype aerOS Reference Architecture. This section reflects on the architecture achievements and challenges, outlining the significant impacts on the IoT-Edge-Cloud continuum. It also delineates the next steps for aerOS, proposing directions for future developments, enhancements, and broader implementation strategies to ensure continued relevance and optimization in real-world applications.
- Appendix A. Finally, this section contains a brief presentation of main, commonly, and frequently aerOS related, and for its purposes produced, terms usage.

1.3. Outcomes of the deliverable

The outcome of the deliverable is the final document of aerOS Meta-OS reference architecture.

aerOS manages to integrate diverse resources across the IoT-Edge-Cloud continuum under a common exposure facility. This integration provides a seamless environment where resources can be managed and utilized efficiently, irrespective of their physical or virtual nature.

aerOS provides a cohesive management and orchestration of these resources via the integration of them within peer entities, the aerOS domain, which provide all the set of core functionalities needed to securely expose their resources and support decision making for services placement within their jurisdiction or over any other domain across the continuum.

aerOS significantly enhances the flexibility and scalability of IoT applications. by federating all integrated domains exposes a unified runtime which supports reusable applications that are built once and can run on all underlying hosting nodes, despite any architectural variations.

1.4. Lessons learnt.

Building a Meta-OS for the continuum emerges as a complex and multifaceted endeavor, demanding a very careful approach to navigate through its intricacies. Recognizing this, our journey underscored the importance of breaking down complexities into manageable steps and proceeding methodically, analyzing, and refining each component along the way. From the outset, it became evident that a careful design approach is paramount, as the convergence of numerous technologies and interactions necessitates a solid foundation from day one. Without this foundational design, the risk of encountering obstacles is not unlikely, potentially leading to dead ends and setbacks.

It is of paramount importance to keep the commitment to guide efforts according to the perspective and expectations of industry IoT developers. It became clear that deviating from this focal point, however innovative the solution may seem, could result in solutions that fail to meet market needs. Thus, maintaining a continuous dialogue with stakeholders, validating ideas, and incorporating feedback throughout the development lifecycle proved indispensable.

Furthermore, our journey underscored the value of creating a dedicated playground for idea validation, where concepts could be tested, refined, and adapted in real-time. Leveraging Minimum Viable Products (MVPs) emerged as a powerful tool in this regard, offering a tangible platform for experimentation and iteration, enabling rapid validation and course correction.

Something equally important was the realization that the employment of existing standardized open source frameworks adds a great advantage as it enables the integration of already battle tested solutions which also offer the community support. For example the integration and extension of FIWARE Orion-LD CB provided a robust basis for building aerOS data fabric core storage engine and federation mechanisms.

Lastly, as pioneers in a field with limited prior knowledge, we recognized the importance of community validation. Engaging with the broader community, soliciting feedback, and integrating valuable insights emerged as essential components of our process. By leveraging the collective wisdom of the community, we were able to refine our ideas, address blind spots, and ensure alignment with industry standards and best

practices. Through these lessons learned, we forged a path forward that not only addresses the complexities of building a Meta-OS for the continuum but also fosters innovation, collaboration, and continuous improvement.

1.5. Version-specific notes

This document represents the second and final version in a series of two architecture deliverables. Taking over from the first version, which provided an initial framework, this document is designed to be a comprehensive and self-contained reference. It not only includes the changes and updates made since the first version but also incorporates the foundational content to offer a complete and coherent overview of the architecture.

The content of this deliverable is the result of the collaborative work of partners in most of the work packages of the project, as their experiences and input were instrumental in driving the architecture revisions. While most of this document has been prepared in the ground of Task 2.5 (which is the responsible one for this deliverable), input and feedback is coming from technical and integration work packages. WP3, WP4 and WP5 have tacitly but effectively collaborated for this version.

For sections that are more research and theory-oriented, this document largely replicates the content from the first version. These sections, written just a few months ago, remain relevant and accurate, and thus, do not require significant updates.

In contrast, sections that detail the actual design and implementation of the architecture have been significantly enhanced. This final version includes all the updates and improvements that have emerged from the technical components' implementation, continuous feedback loops, and integration of the Minimum Viable Product (MVP).

The sections that inherit from D2.6 (although substantially improved when relevant are): 4, 5.5, 5.6, 5.7 and 6.

The new sections, and those that have been largely modified are: 5.1. 5.2, 5.3, 6.6, 7, 7.1, 7.2 and 7.3.

The work done all this period, from first version to this final version has provided valuable insights which support clarification of components' responsibilities and interactions. Based on a more complete understanding of the Meta-OS and its expected operations over the continuum, we have made notable changes that result in a clearer presentation of the integrated aerOS fabrics, explained in detail in this document, and their functionalities. This enhanced clarity allows for an enhanced depiction of the various components and their roles within the aerOS architecture. Additionally, we have strengthened aspects related to the security policy framework to offer seamless and automatic federation of user information and integrate associations with all integrated entities, thereby enhancing the access management framework. These improvements collectively contribute to a more resilient and effective orchestration of resources across the continuum, from edge to cloud, thereby providing a flexible, trusted, and unified development and execution environment for IoT service developers across multiple industry verticals.

These updates reflect the latest developments and refinements in our architectural approach, ensuring that this document serves as a definitive guide for stakeholders. This final version is not just a summary of changes but a standalone document that integrates the foundational elements from the first version while providing comprehensive updates and enhancements. Reflecting on the latest advancements and feedback from ongoing development and MVP integration, it ensures that stakeholders have a complete understanding of both the theoretical underpinnings and practical Meta-OS implementation possibilities based on aerOS architecture.

2. Architecture definition methodology

An architecting process should raise early in the overall development process of a system definition. Such is the case of aerOS, that aims at delivering a Meta-Operating System (Meta-OS) for governing the IoT-edge-cloud continuum. The process must begin with the discussions about what is feasible, efficient, hard, costly, etc., most commonly in parallel with systems analysis and requirements definition activities, resulting in a set of requirements and finally in an architecture that meets those requirements. Identified stakeholders along with a set of technical/user/system requirements and architectural decisions should provide the necessary input for coming up with an architecture description which most probably in turn will lead to updated requirements as architecture evolves through a reviewing process where more architectural decisions are taken. A precise architecture description will provide detailed technical specifications and drive the development process towards realization of the system architecture.

To this end, ISO/IEC/IEEE 42010 standard "Systems and software engineering – Architecture Description" and first published back in 2011 [1], provides valuable guidelines by defining what should be considered when building an architecture description, while it does not mandate how to produce one. Without mandating any specific architecting process, it provides a conceptual model of architecture description and best practices for defining a hopefully highly efficient one. aerOS reference architecture definition is based on the methodology, principles and best practices included in the latest update of the standard, issued in 2022.

In the standard, the architecture of a system is defined as: "*fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution*". An Architecture Description (AD) is used to express an Architecture of a System. Stakeholders have interests in a System; those interests are called Concerns. A system's Purpose is one very common Concern. Concerns range over a wide spectrum of interests (including technical, personal, developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological, social influences). The terms "concern" and "requirement" are not synonymous. A concern is an area of interest. So, i.e., system reliability might be a concern/area of interest for some stakeholders.

Systems have Architectures. Every System inhabits its Environment. A System acts upon that Environment and vice versa. Architecture Descriptions are comprised of AD Elements. Correspondences are used to identify or express named relations within and between AD elements. Creating an Architecture involves making Architecture Decisions. In the process of how to best answer the questions listed as the stakeholders' concerns, Architecture Views provide what the answers are, while Architecture Viewpoints provide how they can be captured. An Architecture View is important to capture the rationale for the key decisions and to include them in the architecture description. Architecture Viewpoints, as an abstraction that yields a specification of the whole system related to a particular set of concerns, reflects the architecting purpose, typical stakeholders and their perspectives, identified concerns, defined aspects of the entity of interest, and particular AD Elements. A well-defined set of viewpoints, reviewed by stakeholders and developers, facilitates capturing architectural decisions. Although the difference between Views and Viewpoints is quite clear, we will use the terms interchangeably for the rest of the document, as this is a common practice for a more accessible content, while in fact we are closely following the approach of the methodology.

This document provides the aerOS reference architecture. The main difference between software architecture and reference architecture is that software architecture is a design solution for a specific software system; on the other hand, reference architecture offers a high-level design solution for a class of similar software systems belonging to a given domain. Thus, reference architecture is more abstract than software architecture and must be instantiated and configured to attend to the specificities of the software being built. Such instantiation will be exhibited in the various aerOS pilot use cases in the revisions of this document to follow.

Although an Architecture Framework maybe considered to offer a higher level of abstraction than a reference model; defined in the standard as the conventions, principles and practices for the description of architectures established within a specific domain of an application and community of Stakeholders; however, minimum requirements set for a framework are considered common for a reference model as well: (1) Information identifying the architecture framework, (2) The identification of one or more stakeholders, (3) The identification of one or more stakeholders' concerns, (4) One or more architecture viewpoints that frame those concerns, (5) Any correspondence rules.

aerOS architecting process is comprised of the following steps:

1. Identification of aerOS Stakeholders.
2. Recording of technical, user, and system Requirements.
3. Identification of system Purpose and Environment.
4. Identification of Stakeholders Concerns.
5. Identification of core Architectural decisions.
6. Architecture Views/Viewpoints development.
7. Re-evaluation of Viewpoints, cross-View consistency, Architectural decisions, and Requirements.

This document, being the second and final version aerOS architecture, embodies requirements (item 2 in the above mentioned list) of both deliverables "D2.2 Use cases manual, requirements, legal and regulatory analysis (1)" and "D2.3 Use cases manual, requirements, legal and regulatory analysis (2)" which provide the initial and the final version of requirements, use cases and scenarios definition as well as legal and regulatory analysis.

A selection of architectural viewpoints that address aerOS stakeholders' concerns, capturing their requirements to provide for a consistent aerOS reference architecture description, is documented in this deliverable and comprises of the following viewpoints:

1. **High-level view**. It describes interactions, relationships, and dependencies between the system and its environment.
2. **Functional view**. It describes the main functional elements of the architecture, interfaces, and interactions.
3. **Process view**. It deals with the dynamic aspects of a system, describes the system processes and their interactions, and focuses on the run time behaviour of the system.
4. **Data view**. It describes data models, data flows, and how this data is manipulated and stored.
5. **Deployment view**. It provides a consistent mapping across the existing and emerging technologies and the functional components specified in the Functional View.
6. **Business view**. It addresses the business processes, organizational structures, roles, responsibilities, and strategic objectives that the system supports.

In the following sections all the prerequisite information is provided to describe the aerOS environment, concepts, terminology, and architectural decisions, before presenting and documenting the various architecture viewpoints considered to provide for a precise aerOS reference architecture description.

3. aerOS initial Reference Architecture validation and review motivations

This section details the methodologies and approaches employed to validate the initial Reference Architecture of aerOS and to gather crucial feedback that informed the updates and revisions. Understanding these methods is key to appreciating the context and rationale behind the architecture concepts acceptance and any changes adopted.

The motivation behind the comprehensive review and validation process of the initial architecture stems from the need to ensure that the architecture remains robust, scalable, and adaptive to the rapidly evolving technological landscape and user requirements. As the foundation upon which all project functionalities are built, the initial architecture must not only meet current specifications but also anticipate future expansions and integrations. The review process was driven by the objective to validate the architecture against real-world scenarios through the deployment of a Minimum Viable Product (MVP), ensuring that it effectively supports all intended use cases while maintaining high standards of performance and security. Furthermore, the review sought to incorporate feedback from all project partners, including technical teams, and end-users and industry experts as final consumers, to refine the architecture's design and functionality. This proactive approach was designed to identify and mitigate potential deficiencies early in the development cycle, thereby enhancing the overall system resilience and user satisfaction.

The cornerstone of the validation process was the development of an MVP, designed to test the core functionalities of the Reference Architecture. The MVP provided a practical, real-world environment to evaluate the architecture's effectiveness and gather actionable feedback from system developers and integrators. This approach made it possible to both identify and address critical shortcomings and to refine the user experience based on direct interaction with the deployed system. The process of MVP deployment and architectural concepts realization and validation was also supported by weekly meetings, supervised and coordinated by the technical leader and with the participation of all technical partners, with the task to bring up and address any issues regarding the transition from design to implementation.

Alongside the MVP, detailed pilot users' feedback was received via WP5 tasks. Regular meetings concerning the transfer and implementation of architectural concepts to aerOS pilots were held. Each pilot actively engaged in dedicated meetings designed to facilitate the effective transfer of architectural concepts to its specific use cases. Complementing these focused discussions, regular meetings of WP5 provided a strategic overview of the ongoing integration strategies and mapping of components to pilot use cases. These meetings acted as a source of input regarding the efficiency and usability of aerOS architecture.

Meta-OS design, as an innovative field, does not demonstrate much prior examples to follow and thus aerOS participation in the European Cloud, Edge and IoT Continuum initiative has been instrumental in validating design and obtaining feedback on the strategic direction. Engaging in extended discussions within assembly of all sibling projects allowed aerOS to present its innovative concepts and, in turn, gather insights that are not readily available in existing literature or practice. This collaborative environment facilitated a rich exchange of ideas and experiences, offering a valuable external perspective. The feedback received through these interactions confirmed that aerOS architectural design path is aligned, or even complementary (see 7.3), with what other researchers and development teams are building and with emerging industry trends and emerging standards. This process introduced the possibility to refine design but also reinforced confidence in the project's direction.

The methodology employed to design the aerOS Reference Architecture, as explained in section 2, is based on the ISO/IEC/IEEE 42010 standard "Systems and software engineering – Architecture Description". No changes have been made to this approach and as a result, the architecture continues to be described in this document through its diverse viewpoints and by illustrating the interfaces of the various components.

Overall, the key ideas of architecture have not changed to any significant degree since "D2.6 aerOS architecture definition (1)". That is because aerOS had already foreseen to create a modular architecture, which allows the system to flexibly integrate and scale various services and components. This modularity is crucial in supporting a range of functionalities from the edge to the cloud, seamlessly adapting to diverse operational requirements

and environments. This foresight in design has ensured that the core principles remain relevant and robust against diverse deployment requirements, and they can support the integration and orchestration of heterogenous resources in a diverse and extended technological landscape, maintaining consistency in strategic direction and deployment capabilities. Finally, the key concepts of the project that enable the delivery of the project's objectives and targeted outcomes have not changed. The federation of aerOS domains, for example, not only supports a common execution environment across the edge-to-cloud continuum but also allows for administrative autonomy and nuanced control over resource exposure. The idea of an orchestration framework which introduces separation of concerns regarding, the AI supported, decision engine and the enforcement layer seamlessly aligns with the underlying federation of resources and their on-demand and semantically enriched, discovery and exploitation, across the continuum. Furthermore, integrating all aerOS domains as peers within the continuum, each equipped with the same set of core services has proven to be a solid approach confirming the service fabric's ability to orchestrate services securely and efficiently over a unified network and compute fabric. The aerOS continuum model, essentially a knowledge graph that correlates and semantically describes all integrated resources, has exceeded our initial expectations and has proven to be a crucial element in deploying robust services and fostering collaboration across all domains on diverse processing units.

A short reference of most noticeable updates would include:

- A clearer view of Network and Compute fabric and Service fabric: These updates detail their respective levels of concern, their interactions, and their support in establishing a continuum for resource abstraction and service orchestration.
- Robust integration of asynchronous communication within each domain: Standardization based on AsyncAPI extends the seamless integration of components, enhancing the overall system's efficiency and interoperability.
- Integration of LDAP in the Identity Management (IdM) system: This enhancement provides more efficient and granular control over resources, whether they are APIs or data, thereby strengthening access management and security.
- Updates in the deployment process, including the detailing of specific flows such as IoT monitoring and actuation service materialization, immutable exchange of key messages, tailored re-orchestration and others.
- Clear mapping to the tentative pre-standardisation activity driven by EUCEI, and how aerOS aligns with the building blocks and components, and goes beyond, technologically.

Instantiation of the reference architecture and mapping of all design concepts and component to pilots are presented in section 7.

4. IoT-edge-cloud continuum ecosystem rationale

The outstanding evolution in the last few years of technologies like Kubernetes (K8s), OpenStack, StarlingX and others (mainly related to cloud computing) -that allow a significatively advanced management over the previous machine virtualization techniques- has opened the possibility of optimizing usage and maximizing efficiency of computing resources. The latter, together with the advent of edge computing as a real alternative for IT ecosystems deployment (mainly due to their benefits in privacy and latency) and the increased miniaturization level and computing capacity of devices in the edge, is creating a solid scenario to build on for generating advanced combined approaches. Cloud-native technologies are also spreading their influence beyond classic, centralized systems and are proposing more lightweight alternatives, compliant with edge computing principles (such as CNCF, which stands for Cloud Native Computing Foundation).

The aforementioned combination of both approaches has nurtured the concept of **Cloud-Edge-IoT continuum.** The term, and the research around it, has been fostered specially in EU during the last few years, being mainly promoted by the EC via a joint coordination and support action named CloudEdgeIoT.eu. Pragmatically, the goal behind this initiative consists in the creation of a paradigm (CEI – CloudEdgeIoT) as a result of the convergence across the whole digital spectrum driven by the advancement in computing technologies. It

embraces the idea of approaching the inclusion of a wide variety of heterogeneous computing resources as a single manageable entity (spanning from IoT devices, edge nodes, private or public clouds, etc.).

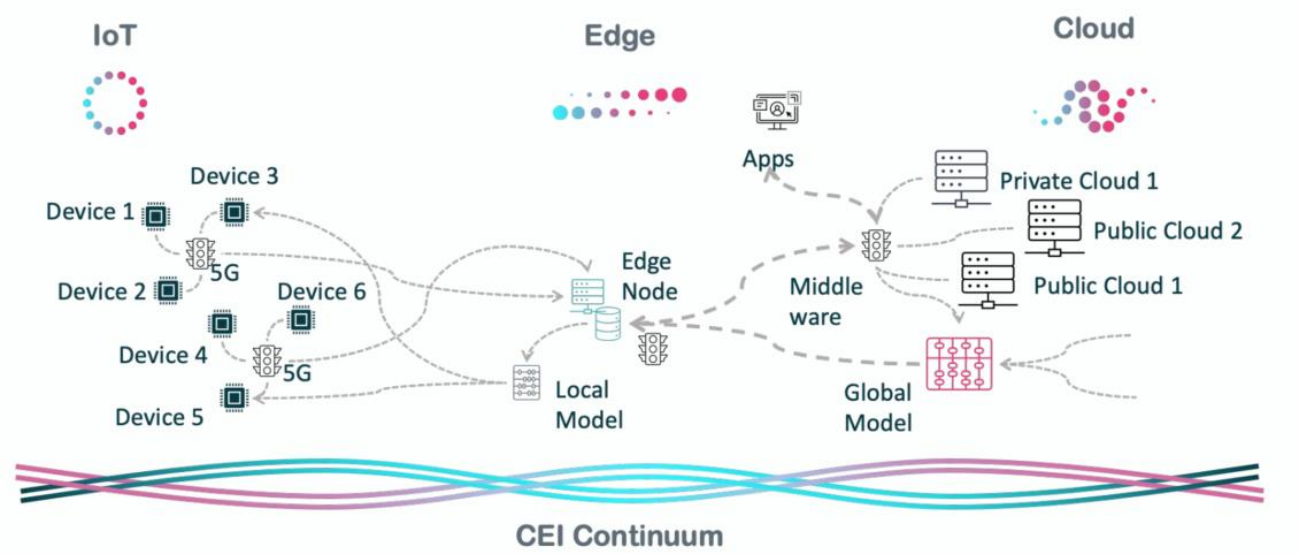
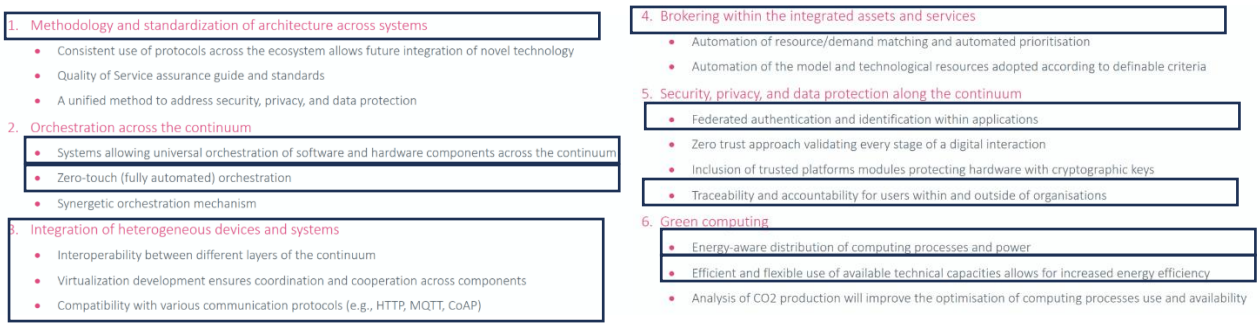


*Figure 1. Cloud-edge-IoT continuum perspective source: **EUCloudEdgeIoT***

Technologically, the continuum approach should achieve better management of the widespread and heterogeneous computing resources stretching along the line from small devices to large cloud datacentres. The continuum must enable and simplify the execution of workloads (applications, services, workflows…) leveraging aspects like network virtualization, energy management, performance, dynamic demand by on-going services, etc. The underlying classification of those elements (nodes) in the continuum, their abstraction, and a common way of accessing and managing them will allow to alleviate the fragmentation and the isolation of technologies for handling those resources, which is the situation nowadays.

In addition, the management of distributed resources in such a way will open up the capacity to support data-intensive applications (data consumers) that make use of data sources which might be located in any spot across the continuum. This will also enable advanced access policy management, ensuring data governance and allowing the emergence of concepts like data spaces or data fabric. Also, manipulating workloads and network in a Meta-Operating System (Meta-OS) for the continuum will enable the reduction of latency in certain distributed services, like real time verticals, multimedia streams, or bandwidth-constrained applications. Here, security and privacy also play a key role, as well as automated discovery and adjustment, enabling multi-tenant (multi-stakeholder) participation in a continuum. All the previous will require interdisciplinary approaches, like the one proposed in aerOS.

According to the most prominent initiative in the field, the compelling need of solutions for managing the computing continuum is expressed by the following list of requirements (that must be covered in years to come):



aerOS tackles the 6 aspects, focusing mainly on the signalled traits. It is within the objective of the project to deliver a Meta-OS, deployable in heterogeneous resources and across verticals, that will serve as the first (and main) solution for orchestrating the continuum. And that is exactly what this document delivers: the reference architecture of aerOS Meta-OS.

aerOS shares this quest with five other EU-funded projects (ICOS, FLUIDOS, NEPHELE, NEMO and NEBULOUS), that share their ideas and advances periodically, collaborating in the pursue of a reliable, unified, European vision of the continuum and its management. This unification starts with the definition of the terms that rule the functioning of it, and continues by defining the building blocks, required functionalities and prominent examples and success stories. This is being done by gathering representatives of EU Meta-OS projects to work collaboratively on the development of a common standardised taxonomy for all Meta-OS European projects. Besides, **the CloudEdgeIoT ecosystem** has developed a shared taxonomy in the field of the continuum, eliminating ambiguities and duplication of terminology and being as specific as possible.

Among the previous, aerOS is introducing their specific proposals into the new paradigm of continuum. A full list of terms is included in the Appendix A, however, the most representative terms (that will repeat over the document are):

- **Infrastructure Element:** the most granular entity of computing (able to execute workloads) in the continuum. It can be instantiated in many forms.
- **Domain**: grouping of Infrastructure Elements according to certain aspects defined by aerOS.
- **Data Fabric (thus, federation):** the conception of all data available in a continuum as a single box that can be queried and will forward the proper information. Mechanisms within are rather sophisticated.

4.1. IoT as enabler of edge and cloud computing

As the years go by, increasing data volumes keep forcing the leveraging of data processing capabilities into on-demand available computer system resources, known as “the cloud”. To achieve scalable and sustainable solutions, the distributed edge-cloud computing complexity must be managed via standards and deployment models. To face this challenge the Next Generation Internet of Things (IoT) has emerged to define the requirements and appropriate approaches to harness the power of IoT and edge-cloud devices. IoT elements consist of physical objects with sensors, computational power and connectivity capabilities that connect and exchange data over the Internet.

However, the challenge of latency arises, among other glaring issues. To start off, data need to travel back and forth between the edge devices and the cloud. As a result, traditional cloud computing models suffer from latency problems. IoT can alleviate this issue by bringing the computational power closer to the data sources; thus, reducing latency. However, certain factors such as network distance limitations may persist and cannot be eliminated by IoT alone. Another problem of increasing the load in the cloud is bandwidth consumption, since sending large amounts of data from the edge devices into the cloud can consume significant bandwidth, straining the network. IoT devices can ease this by employing edge-computing techniques, processing, and filtering data locally, sending only the relevant data to the cloud. This selective data transmission reduces bandwidth usage and optimizes the usage of the network resources. However, bandwidth limitations will persist and need to be addressed at the infrastructure level.

To exploit the advantages that IoT offers, an innovation shift is required towards an edge-cloud computing continuum, in which computing resources, as well as storage resources can be located everywhere in the network. With this, an expanded network compute fabric is created, spanning over both the devices and the cloud.

In aerOS, the role of IoT will be crucial, as the ever-increasing number of devices will be an intrinsic part of the continuum. Those will be considered as an active part of the complexity and heterogeneity of the continuum, providing relevant data to both manage the infrastructure and facilitate added-value services to stakeholders. Data served by IoT devices will be integrated in the global Meta-OS following specific mechanisms (i.e., Data Fabric – see 5.5.2), posting data in a way that will be accessible by any participant of the continuum. In addition, IoT devices will be associated to an element of the global ecosystem (even being an active part of it), whenever they have the required computing capacity. This design choice is aligned with the miniaturization trend.

4.2. Rationale towards an IoT-Edge-Cloud continuum

Each one of the computing layers that enable IoT applications, namely IoT Devices, Edge computing, and Cloud computing, exhibit distinct characteristics concerning data handling and sovereignty. IoT devices are responsible for data collection and interaction with the physical environment. They possess limited resources and focus on real-time data processing, often transmitting only essential information to conserve bandwidth. Edge computing, located between IoT devices and the cloud, aggregates, and pre-processes data locally, reducing latency and ensuring faster response times. The cloud, on the other hand, provides extensive computational power and storage capacity, enabling large-scale data analysis and long-term storage. Each layer plays a vital role in the overall architecture, addressing specific requirements and challenges.

Certain applications benefit from leveraging the vast computational resources of the cloud. These applications involve intensive data processing, complex analytics, or require access to sophisticated Machine Learning (ML) algorithms. By harnessing the power of the cloud, organizations can analyse massive volumes of data, derive meaningful insights, and execute resource-intensive tasks efficiently. On the other hand, some applications demand low latency and real-time responsiveness. These applications operate on the edge, closer to the data sources, to minimise delays and enable near-instantaneous decision-making. Edge computing ensures faster response times, reduced network congestion, and improved overall performance for latency-sensitive applications.

However, a growing number of new applications necessitate a dynamic approach that intelligently utilises the resources of all three computing layers (IoT, edge, and cloud) to achieve optimal performance while adhering to data sovereignty and privacy restrictions. These applications will leverage the strengths of each layer while optimizing resource utilisation. They will employ intelligent data routing and processing mechanisms to determine where and how to handle data most effectively. By intelligently distributing tasks across IoT devices, edge servers, and the cloud, these applications strike for a balance between real-time responsiveness, efficient data processing, and compliance with data regulations. This dynamic utilization of resources allows organizations to achieve high-performance outcomes, while respecting data sovereignty, privacy, and compliance requirements.

The dynamic capabilities of those new applications drive the innovative concept of IoT-Edge-Cloud Continuum; an architecture approach where the management of data, services, network resources, and computing capabilities is performed with an overarching and unifying view across the three computing layers.

4.2.1. From heterogeneous IoT data to a unified data fabric

The IoT landscape has experienced a proliferation of heterogeneous data sources, containing data in different encoding formats and structures, but also, exposing these data through different access protocols depending on the technology of each data source.

The composition of a continuum based on multiple technological domains ranging from IoT devices, Edge sites, to the Cloud, highly increases the complexity of the data management activities. Data sources are not only statically located in physical locations, such as IoT sensors, but they can also be spread across multiple physical and virtual locations across the different domains. This constitutes a highly changing environment, wherein new data sources become available, move to other domains, and even disappear (unexpectedly or orchestrated). Such an intricate and heterogenous data landscape presents several challenges in two main aspects: i) data consumption and ii) data governance.

Vertical services deployed on the continuum, such as ML/AI applications, that aim to find and analyse these data to realize their business demands, would have to deal with such a complex and heterogenous data landscape. Vertical services first would have to find and understand the data of interest for their use case. They would also need to implement specific mechanisms for ingesting, processing, and consuming the data of interest. Moreover, advanced use cases would require ML/AI applications to correlate and combine heterogenous data from multiple data sources. This whole workflow entails an extremely time-consuming effort that, in addition to having data engineering skills, requires a deep understanding of the available data.

Similarly, data governance processes must be able to cope with this diversity of data in a dynamic environment. The security and privacy teams must ensure that data are properly classified and protected, accessed only by

authorised consumers, and used for a specific reason. Keeping track of all these activities calls for a holistic view of the available data and how they are exchanged within the continuum.

The aerOS Meta-OS achieves to deliver a feeling of a continuum, thus, when it comes to data management, aerOS provides a uniform access to all the data, regardless of their location in the continuum, their original encoding format, or the underlying technology used by the "golden" data source. To this end, aerOS builds on a semantic layer over the continuum that abstracts data consumers from the technological complexities and facilitates the discovery, understanding, and access to all the data through a uniform interface. Precisely, this is the idea behind the data fabric paradigm; however, aerOS goes beyond that, by extending the data fabric throughout the IoT-Edge-Cloud continuum following federated and distributed architectures. The aerOS Data Fabric aims to become a one-stop-shop for data consumers to easily find, understand, and access any data available in the continuum; whereas the data governance team is provided with a complete view of how these data are being used within the continuum.

4.2.2. From a distributed cloud eco-system to a unified network and compute fabric

Over the course of time, the field of computation has witnessed the evolution of diverse paradigms, ranging from traditional parallel and grid computing to the advent of cloud computing. Cloud computing, encompassing service models like Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS), offers numerous advantages and capabilities. These include scalability, on-demand resource provisioning, a pay-as-you-go pricing model and streamlined provisioning of applications and services.

The emergence of 5G and beyond has brought forth a multitude of novel applications, such as massive Internet of Things, mobile video conferencing, connected vehicles, e-healthcare, online gaming, and virtual reality. As indicated by both industry and academic research initiatives, these applications require high data rates ranging from 1 to 100 Gbps, as well as low latency in the range of 0.1 to 1 ms for ultra-low latency applications. However, cloud computing alone falls short in meeting these evolving requirements due to several issues.

The primary challenge lies in the considerable distance between cloud resources and end devices, as the connection is established over the Internet, resulting in latency issues. Additionally, the processing capacity of cloud servers is insufficient to effectively cater to the emerging demands. For instance, the latest generation of general-purpose computing instances in Amazon EC2 cloud service possess processing capabilities on the order of 5 to 50 Gbps. Nevertheless, this level of processing power fails to adequately accommodate the vast number of applications and the traffic generated by IoT, where high data rates are necessary for many applications, such as data rates of multiple-Gbps for high-quality 360-degree video.

The concept of edge computing, encompassing paradigms such as cloudlet, mobile edge computing, and fog computing, was introduced as a solution to address the challenges associated with cloud computing. While edge computing improves latency and enhances processing capacity by providing resources at the network edge, it is not expected to sustain the continuous growth in traffic volume. Additionally, the latency achieved through edge computing falls short of the stringent requirements for ultra-low-latency applications, which demand round-trip latencies of less than 1 ms, possibly as low as 0.1 ms.

The concept of distributed cloud computing has recently emerged as a solution to enhance the performance of cloudlet, mobile edge computing, and fog computing paradigms. It achieves this by utilizing the computational and storage capabilities of nearby intelligent devices for offloading computations or caching data. However, there are significant challenges related to the limitations of computation and power, the mobility of neighbourhood devices, and particularly the security concerns associated with offloading computations to these devices. To address these challenges, there is a need for a more secure, power-efficient, and reliable computational infrastructure with a high processing capacity, which can complement the existing computational paradigms.

In this regard, the in-network computing paradigm, which is based on programmable data plane technology (an evolved concept of Software Defined Networking – SDN), can provide power-efficient network elements with high processing capabilities at the network's edge. By effectively utilising in-network computing, packets can be processed at line-rate, along the path, and before reaching the edge/cloud servers. This paradigm offers faster

processing capabilities at locations closer to end devices, surpassing the performance of edge or cloud servers employed in traditional edge and cloud computing paradigms.

Figure 2 provides a schematic representation of the different computing resources that may be utilised by aerOS whereby its Infrastructure Elements can be hosted. It illustrates the in-network computing fabric, which comprises network elements such as programmable switches, Field Programmable Gate Arrays (FPGAs), and smart Network Interface Cards (NICs) accelerators embedded within a host. These network elements can be strategically positioned between end-devices and servers offered by edge computing, including Multi-Access Edge Computing (MEC) servers, fog nodes, and cloudlets. Alternatively, they can also be deployed between end-devices and cloud datacenters, or even between edge servers and cloud servers. This architecture enables efficient utilization of computing capabilities across various computing resources. The different resource types, shown in Figure 2, may serve different purposes. Below a limited selection of examples is provided:

(i) Analytics: The available resources can be leveraged for conducting analytics tasks. This includes ML, data aggregation, heavy flow detection, query processing, control operations, and deep packet inspection. Such tasks can be performed either on the path in the case of In-Network Computing or at specific locations such as central cloud servers, edge cloud servers, or end devices.

(ii) Caching: The diverse resources can be utilised to establish a caching infrastructure on top of storage servers. This aims to reduce data access time and is relevant for key-value store applications as well as information-centric caching.

(iii) Security: The resources distributed across different locations, namely in-network computing and edge cloud, can be employed to fulfil specific or comprehensive functionalities necessary for detecting and mitigating attacks on the infrastructure or the provided services. This approach aims to minimise attack mitigation latency and operational costs associated with dedicated security servers.

(iv) Technology Specific Applications: Applications can either run exclusively on a single resource type, such as the central cloud, or specific components of the applications may be distributed across alternative resource types, such as the edge cloud or the end devices. End devices may also offload their computations to virtual resources running in the edge cloud, in order to overcome the limited available resources at the devices. These virtual resources at the edge cloud can expand and shrink dynamically and will have scalability with respect to the service requests. End devices can offload their computations to the edge cloud in their proximity, thereby overcoming the poorness of resource limitation in the device, as well as guaranteeing real-time interactive responses. In cases where the edge cloud or cloudlet is inaccessible in proximity, there remains the possibility of connecting to a remote central cloud. However, such a connection may result in a degradation of response time for obtaining the required service.

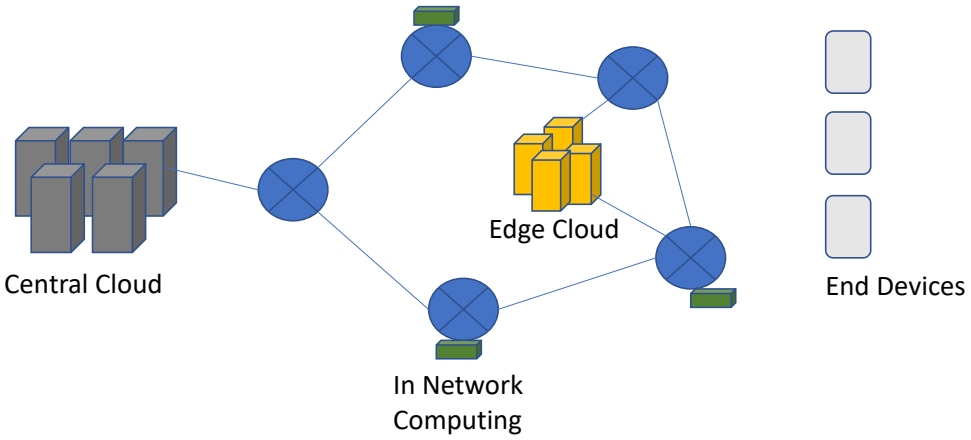


Figure 2. Schematic of aerOS Cloud Resource Types: Central Cloud, In-network Computing fabric, Edge Cloud and End Devices

In the presence of diverse cloud resources, including cloud, near and far edge cloud, and in-network computing, ensuring efficient orchestration of these resources is of utmost importance from various perspectives [2]. There exist different strategies for resource orchestration, spanning from initial resource selection to service and resource migration. A significant research direction lies in orchestrating a wide array of resources to meet application requirements, necessitating solutions for resource allocation and traffic steering to deploy desired applications effectively. While some studies have addressed resource allocation in hybrid environments consisting of network elements and cloud computing nodes [3], [4], with the aim of maximizing request admission or minimizing deployment costs, their scope remains limited. Further research is warranted to explore orchestration approaches that encompass additional objectives, such as maximizing throughput, minimizing bandwidth utilization, reducing power consumption, and optimizing quality of service metrics, including latency and reliability.

Effectively, the current cloud ecosystem is comprised of multiple cloud providers, each representing a separate cloud domain. These domains are characterised by their individual Virtual Infrastructure Managers (VIMs) or controllers, which govern the virtualised infrastructure. This decentralised architecture necessitates the management of applications and resources across multiple cloud domains, leading to challenges in scalability, complexity, and cost-effectiveness. With the advent of near and far edge computing, the diversity within the cloud ecosystem is further accentuated. Edge cloud providers bring computation closer to the point of data generation or consumption, enabling reduced latency, improved responsiveness, and enhanced user experience. However, integrating these edge cloud domains into the existing multi-cloud environment amplifies the complexity of managing applications across diverse infrastructures. Moreover, considering that such hybrid/multi-cloud environments currently rely on container management frameworks such as Kubernetes, which are not properly serve the requirements of such heterogenous and distributed IoT services and equipment, the need for developing a homogeneous overlay management system is becoming a necessity. To achieve specific objectives such as geographical distribution, low end-to-end latency, efficient bandwidth utilization, and other application-specific requirements, it becomes imperative to deploy applications across multiple cloud domains, but under a unified management framework, which may include catering to a dispersed user base, reducing latency for cloud-based services, accommodating bandwidth-intensive applications, and more.

As applications become hyper-distributed, they rely on computing resources (clouds, edge, data centres in general) belonging to different providers, connected via networks with varying bandwidth, latencies, and probability of connectivity loss, often beyond the control of the application owner. These computing infrastructures consequently operate as isolated aggregations with fragmented resources, making seamless provisioning of hyper-distributed applications challenging. Managing such deployments through individual interfaces to each cloud VIM becomes highly complex, has limited scalability and lacks cost-effectiveness.

The complexity of managing applications across multiple (cloud) domains is further exacerbated by the need for external networks and the maintenance of Quality of Service (QoS) requirements. Applications rely on communication among their components or microservices, necessitating network connectivity between different (cloud) domains. QoS parameters such as latency, bandwidth, jitter, and packet losses must be maintained to ensure optimal application performance. Managing this network-compute fabric in a unified and uniform manner poses challenges at several levels, as mentioned above. To address the complexities and limitations associated with managing multi-cloud environments, there is a growing need for a unified framework. Such a framework would provide a centralised management approach for the diverse cloud domains and the associated network infrastructure. The offer of a unified interface would streamline application deployment, resource allocation, and network connectivity across cloud domains. This unified framework aims to resolve the aforementioned challenges and maintain QoS requirements across the entire network-compute fabric.

4.2.3. From monolithic applications to intelligent distributed services

In a typical cloud-centric approach, applications and services usually have a monolithic, albeit highly scalable, architecture. It assumes the existence/availability of a specific set of cloud-based computing resources but may also include selected IoT/edge resources, playing mainly the role of data sources. A monolithic application is a single, closely connected software system that is designed and delivered as a single entity. The use of the cloud as today's ultra-powerful “mainframe” allows to take advantage of its incredible potential, but, at the same time,

it introduces obvious latency and privacy issues. This is exacerbated when applications become data-centric, as data typically does not come from the cloud and may be sensitive imposing additional restrictions on centralised processing.

The term "monolith" is frequently associated with something big and glacial, which is not far from the truth of a monolith architecture software design. A monolithic architecture is a single, vast computing network with a single code base that connects all business concerns. Making changes to this type of application necessitates accessing the code base and developing and delivering an updated version of the service-side interface. This makes updating difficult and time-consuming. A monolithic architecture has the following advantages:

- Simple deployment - A single executable file or directory simplifies deployment.
- Development - It is easier to build an application when it is developed using a single code base.
- Performance - In a centralised code base and repository, one API may frequently perform the same job as several APIs using microservices and communication between internal components does not cause additional cost overhead.
- Testing is simplified since a monolithic application is a single, centralized entity, allowing for faster end-to-end testing than a dispersed program.
- Simple debugging - With all code in one location, it's easy to track a request and identify a problem.

However, a monolithic application has the following drawbacks:

- Slower development pace - A huge, monolithic program complicates and slows development (specifically with time and growing complexity of the system).
- Scalability - Individual components cannot be scaled or scaling requires some essential modifications.
- Reliability - If any module fails, the availability of the entire application may suffer.
- Adoption of technology is hampered since any changes to the framework or language influence the entire program, making modifications costly and time-consuming.
- Lack of adaptability - A monolith is confined by the technology that it presently employs.
- Deployment - Making a little update to a monolithic program necessitates redeploying the entire monolith.
- Single-ownership assumption is made omitting the problem of potential restricted data sharing and processing.

Moreover, the monolith application cannot take advantage of the heterogenous infrastructure on top of which the solution could be deployed in a distributed way. The IoT-Edge-Cloud Continuum governed by aerOS introduces an agile architecture and mechanisms that allow for dynamic (re)allocation of resources (both computational and data-related), and efficient deployment/configuration of services. Thanks to the application of thoughtfully selected mechanisms and intelligent decision-making techniques aerOS will be able to intelligently manage the execution of application workloads and deployment of services across the continuum. In particular, it will support dynamic distribution and placement of services and user applications, offering user-configurable “policies”. Additionally, thanks to the concept of Data Fabric, distributed data sources and services based on them, will be able to flexibly manage data access policies and mechanisms as close to the data origin as possible, minimizing possible security and privacy risks.

A proper application type that can take the advantage of this distributed aerOS architecture is based on a microservices architecture software design. A microservices architecture, commonly known simply as microservices, is an architectural solution that is based on a collection of independently deployable services. These services have their own deployment requirements, business logic and database, and they serve a specialised purpose. Each service has its own lifecycle undergoing updates, testing, deployment, and scalability.

Microservices split important business issues into separate, independent code bases. Microservices do not diminish complexity, but they do make it visible and controllable by breaking activities down into tiny processes that work independently of one another while contributing to the overall total. Moreover, with the proper deployment, micro services can optimise the utilisation of the underlying infrastructure with respect to computational costs and energy consumption.

However, microservices come also with their challenges, which aerOS comes to address. When establishing a microservice application in the cloud, the scheduler must properly plan each service on dispersed compute clusters, which may have varying resource demands. Furthermore, network connection between various services must be managed carefully, as communication circumstances have a substantial impact on service quality (e.g., service response time). It is becoming increasingly vital to ensure the required performance of service-based applications, particularly the network performance between the relevant services. Therefore, the unified network-compute fabric that aerOS envisions becomes fundamental for the optimal microservice placement process, while the unified management the aerOS offers as a Meta-OS facilitates this orchestration process.

Considering additionally the distributed nature of the aerOS across the continuum, the orchestration of the microservices is also evolved. In a microservice architecture, orchestration and choreography are two techniques of interacting with other software components. In general, orchestration is used when there is a need for a centralized authority to control the interactions. Orchestration is based on the *orchestra* notion, in which numerous artists are masters of their instrument, which is a service in our microservices design. The conductor of an orchestrator directs everyone in relation to the nodes. Similarly, when a parent service administers the request, it forwards the request to the appropriate service and gathers the data. The final answer is created in the primary service known as orchestrator and delivered to the client application.

Microservices, unlike orchestration, function in parallel in choreography, which is used when there is a need for a more decentralized and autonomous interaction between services, which is the case for the aerOS Meta-OS. Substantial parts of the systems are built on an event-driven architecture, in which a service gets data from a message bus, performs business logic, and then submits data to another message bus.

There are several subjects in microservice choreography to which a service may subscribe and update another topic. The microservice's task is specified, and it simply checks whether the subject is empty or not. Unless the subject is empty, it continues to accomplish the work at hand. In a choreographed microservices architecture, adding and deleting services is significantly easier. All that is needed is to connect (or detach) the microservice from the proper channel in the event broker. The installation and removal of microservices does not compromise current logic with loose service coupling, resulting in reduced development turnover.

aerOS is proposing a hybrid approach in the management of service placement between the typical Orchestration and Choreography approach, which is based on the combination of two-level orchestrators and an intelligent load balancer that are coordinated in a decentralized and autonomous way, adopting the event-driven choreography principle.

4.2.4. From decentralized services to federated domains

In recent research trends, there has been a significant shift towards developing more resource-efficient and effectively managed distributed computing environments. Current research [5] suggests the transition from traditional centralized and semi-centralized models to advanced federated architectures. These federated systems represent a paradigm shift, offering a more egalitarian framework where all nodes within a network operate as peers, each with equal access to resources and decision-making capabilities. Such an approach not only enhances autonomy but also promotes a more granular level of control over distributed resources, fundamentally altering the ecosystem for orchestration and resource utilization.

Federation in distributed systems transcends basic decentralization by embedding peer-based operational equivalence across the network. This ensures that each node can retain its autonomy and does not have to depend on other nodes' capabilities to be fully functional and at the same time no single node bears excessive burden or possesses undue control, thereby eliminating operational bottlenecks and facilitating a more streamlined and resilient orchestration process. The move towards federated architectures is driven by the necessity for systems

to adapt to diverse and dynamic operational environments without compromising on autonomy, efficiency or scalability.

In contemporary distributed computing environments, the centralized orchestration model often results from a fragmented understanding of the state across the continuum. Each administrative space would possess only a partial view, limited to its local resources. This fragmentation compels a centralized approach where local state fragments are aggregated at a central location, typically where the orchestrator resides, enabling efficient service component placement decisions across the continuum.

However, this centralized model introduces inefficiencies and hierarchies in decision-making. Some alternate designs attempt to alleviate complete centralization by establishing a hierarchical structure in orchestration capabilities. In these designs, "semi-central" nodes possess an expanded, albeit incomplete, view of the continuum's state. These nodes can make limited decisions or escalate them to higher-level nodes that encompass a comprehensive view of the continuum, thereby enabling final orchestration decisions. Such hierarchical models inherently create disparities in node capabilities, leading to a more complex, energy-intensive, and slower orchestration process.

Against this paradigm, architectures which integrate federation, by treating each domain as an equal peer within the continuum, ensure that all domains have equal and direct access to comprehensive state information of all resources. This democratization of information underpins the ability to deploy a uniform set of orchestration services across all domains. Consequently, any domain can make informed placement decisions and manage resources anywhere across the continuum, from the far edge to the cloud. This peer-based architecture not only simplifies the orchestration process but also enhances the autonomy of each domain.

As the discussion often revolves around how to achieve optimal orchestration, ensure data consistency, and handle the complexities of cross-domain transactions and interactions in a secure and efficient manner, the integration of cutting-edge technologies such as artificial intelligence, distributed ledger technologies, and advanced cybersecurity measures strongly support the development of these federated systems. Artificial intelligence can further enhance decision-making with predictive analytics and automated management tasks, distributed ledger introduces immutable records and enhanced security protocols for transactions across the network, and cybersecurity advancements ensure robust protection mechanisms are in place to safeguard against evolving threats. Together, these technologies provide the essential tools needed to manage and secure distributed IoT-Edge-Cloud networks effectively and promote the adoption of federated architectures.

Thus, the shift towards federated models not only represents a significant innovation in the way distributed services are orchestrated but also sets a new standard for the future of networked systems. It shifts away from hierarchical and centralized models, promoting an equitable and efficient framework where each node or domain retains full autonomy and capabilities, mirroring a truly distributed network that leverages collective intelligence for operational excellence. This research trend points towards an ecosystem where distributed computing not only meets the demands of modern applications more efficiently but also does so in a manner that is inherently secure and scalable, driven by the latest advancements in technology.

5. The aerOS continuum

5.1. Meta-OS approach and aerOS vision

A Meta-OS aims at mimicking crucial services and functionalities of an operating system, operating within an environment that integrates numerous distributed input/output resources in an information-driven manner. The type and number of functionalities offered may categorize it as neither an operating system nor a framework. A good example of a Meta-OS approach is the popular Robot Operating System (ROS) [6] in the robotics domain. ROS, a Meta-OS for robots, is an open-source platform whose functions are equivalent to that of an operating system; functions include low-level device control, hardware abstraction, message passing between processes, implementation of commonly used functionality, and package management. The ROS Meta-OS also provides libraries and tools for obtaining, building, writing, and running code across multiple computers while accommodating different combinations of hardware implementations. Meta-OSs developed in various domains

enable different levels of coordination among heterogeneous devices (physical or virtual), operating systems, and services [7] [5].

aerOS builds on the Meta-OS approach to intelligently manage and integrate a distributed set of resources into a seamless continuum, supporting the orchestration of hyper-distributed applications across the IoT–Edge–Cloud spectrum. aerOS aims to implement a Meta-OS that unifies and orchestrates computing and network resources in the most efficient way, providing a common and unified execution environment for IoT service developers across a distributed computing environment. These resources are located in various geographical and administrative domains and have significant diversity in their capabilities and operating systems. End users, i.e., IoT developers from various industry verticals using aerOS, should experience seamless integration of these underlying resources and transparently benefit from the smart, aerOS-enforced orchestration and federation decisions in their effort to deploy services in closest possible alignment to their requirements leveraging the most appropriate and efficient resource capabilities exposed from the entire infrastructure, from edge to cloud.

This necessitates aerOS, as a Meta-OS, to implement, operate, and manage the continuum across all involved layers. To understand aerOS as a Meta-OS, it is necessary to explore the concept of the continuum as a derivative of layered fabrics supporting end-to-end and edge-to-cloud orchestration, lifecycle management, and federation of computing and networking nodes (physical or virtual), services, and data. Thus, aerOS can be perceived as the enabler and administrator of the continuum, acting as a key facilitator with the dual role of i) orchestrating diverse resources and ii) establishing a unified execution environment for IoT services deployment. This enabler effectively bridges the edge-to-cloud pathway, offering functionalities akin to an operating system that manages legacy hardware resources.

To ensure that network, compute, and data resources work together, enabling optimal performance, scalability, and reliability for hyper-distributed applications aerOS builds on the idea of "fabric" which abstractly represents a unified framework that seamlessly integrates and orchestrates these resources. Within this context, in the highly distributed computing environment encompassing a diverse range of physical and virtual computing and network devices, the establishment of the **"aerOS Network and Compute Fabric"** enables seamless connectivity, unified exposure, and orchestrated management of the underlying infrastructure. This fabric spans from the edge to the cloud and across multiple domains and stakeholders. Currently, many non-connected, distinct computing islands host applications with specific demands. Elements in these islands provide processing, storage, or networking capabilities only to a certain extent and often rely on big cloud providers to execute intensive computing tasks. The aerOS Network and Compute Fabric aims to expand these capabilities by providing federated access to unused and available resources from edge to cloud, establishing a "continuum" of resource exposure.

Built on the foundation of the "Network and Compute fabric", aerOS offers a comprehensive framework for building and managing microservices and container-based applications. It provides the infrastructure to deploy, manage, and scale services efficiently across various environments. Essential mechanisms, deployed as aerOS microservices, facilitate seamless and transparent resource sharing, orchestration, and federation, supporting core aerOS functionalities. This is the **"aerOS Service fabric"**. Service Fabric includes management and orchestration (MANO) capabilities for full lifecycle management (LCM), including development, deployment, and scaling of IoT applications and services. This abstraction empowers developers to create scalable, reliable, and highly available applications by simplifying access to underlying infrastructure and ensuring efficient placement, resiliency, and migration of services. It enhances overall performance by continuously selecting the most efficient placement of services on appropriate resources. aerOS Service Fabric consists of basic and auxiliary services running within each registered domain on top of their network and compute elements. This fabric exhibits key characteristics for effective service management. Orchestration decisions within the aerOS Service Fabric break down IoT service deployment requests into microservices, provisioning them based on resource availability and required capabilities. This provides comprehensive support for IoT developers to deploy applications across vertical domains, from the edge to the cloud (public or private). Features include lifecycle management, intelligent orchestration, resilience, scaling, discoverability, messaging, monitoring, and more.

The Network & Compute fabric as well as the Service fabric encompass extensive information about their capabilities, availability, and context-related information (e.g., location). Each computing resource and service produces substantial runtime data, providing insights into their current state and how their capabilities and

availability change over time. A comprehensive layer, the **"aerOS Data Fabric",** is implemented to facilitate the efficient management, integration, and access to all this data across distributed systems and diverse data sources. The Data Fabric provides a unified and consistent data layer. Methodologies, pipelines, and tools, constituting this layer, abstract the complexities of data acquisition, processing, semantic and syntactic homogenization, storage, governance, provision, enabling aerOS data producers and consumers across discrete administrative domains to transparently exchange data. At the heart of aerOS Data Fabric a knowledge graph is employed to store, represent, and correlate aerOS-integrated resources (hardware and services) and their properties. This choice allows resources, along with their state, to be depicted as entities with connected properties and attributes. Relationships deployed within the knowledge graph represent their physical, multilevel connections, with flexible relationship objects visualizing a potentially "mobile" world where the presence of computing resources and IoT services may change anytime, triggered by unforeseen events. This contextualized provision of aerOS information, representing and interrelating, devices, data, and resources is used as the substrate for cognitive operation related to both continuum orchestration and services peer support.

Thus, aerOS builds both on the concept of developing an unprecedented graph of interconnected or interrelated computing, networking, services, and IoT resources for the continuum and on the mechanism to distribute this information on demand and in real-time among all connected domains and elements. These capabilities enable access to information about all aerOS entities deployed, providing a full overview of the ecosystem at any time for any consumer, and the exploitation of this information across the ecosystem, from edge to cloud, to develop mechanisms that recursively adapt and reconfigure resource usage and orchestration to meet user or owner criteria.

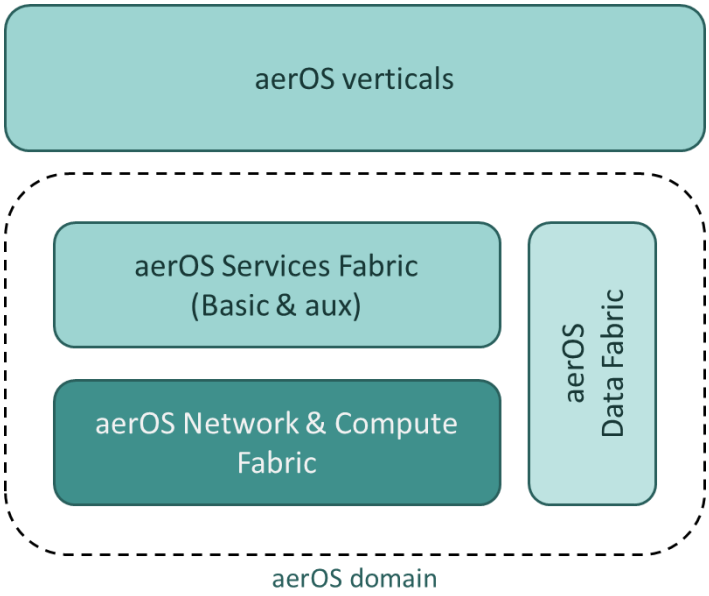


Figure 3. aerOS domain as part of the continuum

All the above highlight a system that can seamlessly integrate a wide spectrum of computing and network resources and services, spanning from the edge to the cloud. It represents a unified architecture where computing capabilities distributed across different administrative domains and physical locations, are federated, and orchestrated allowing thus for the most efficient and optimized placement of workloads. This seamless integration and orchestration of resources, services, and data across multiple domains, locations, and heterogeneous devices and operating systems, along with the data fabric capabilities to share and distribute data under interoperable mechanisms, establish a continuum. A continuum that spans distinct administrative domains, diverse device capabilities and architectures, and different connectivity networks, working seamlessly and leveraging all underlying resources for the benefit of IoT developers and users. A continuum that provides a common execution environment for this community.

As a result, **aerOS, as a Meta-OS**, intelligently establishes and manages a seamless continuum that integrates resources and services, generating valuable data continuously monitored, processed, and utilized by distributed AI services to enhance continuum functionality. Insights derived from AI-based data pipelines feed into the

orchestration processes, adapting and reconfiguring resources and services, enabling a Meta-OS self-sustaining loop. This process operates under a zero-touch management paradigm, where automation and intelligence drive the continuous evolution and optimization of the continuum ecosystem. Ultimately, aerOS provides a comprehensive framework to establish a seamless continuum across computing, service, and data layers. As a Meta-OS, aerOS supports the establishment, sustainability, and efficiency of this continuum, offering IoT service developers, in industry verticals, a unified execution environment built upon a transparent ecosystem of resources. This unified environment streamlines the development, deployment, and management of IoT services, fostering efficiency and innovation within the industry.

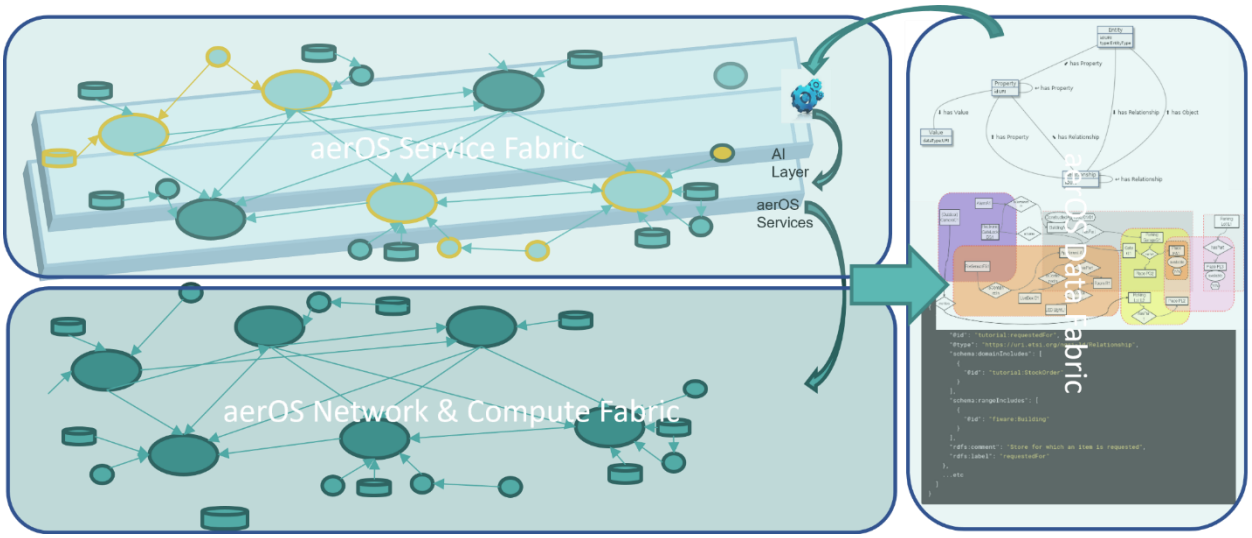


Figure 4. Compute, Service and Data Fabrics as aerOS continuum constituents.

5.2. aerOS building blocks

To develop aerOS, following a Meta-OS approach that encompasses all continuum aspects discussed earlier and addresses the project objectives, the architecture design incorporates key building blocks structured to realize its fundamental concepts. This section provides a comprehensive overview of the aerOS system, consolidating the main concepts and building blocks. Subsequent sections delve into precise descriptions and thorough analyses of the core elements, services offered, and functionalities exposed by the system.

The approach chosen to describe the overall capabilities and components of aerOS is incremental, reflecting the evolving demands of transitioning from a legacy IoT system's approach to a Meta-OS for the IoT-Cloud-Edge continuum. We gradually introduce the system's fundamental concepts, implementing components, and exposed capabilities, acknowledging the dynamic nature of this transformation.

The design and establishment of the aerOS continuum, which seamlessly integrates IoT devices and resources from the edge to the cloud, is built upon multiple **aerOS domains**. Each of these domains consists of a collection of **Infrastructure Elements** (IEs), with at least one IE per domain. Among all aerOS domains, the Entrypoint Domain is assigned a special role across the continuum, functioning as the primary access point for the system. While uniquely instantiated in one domain, it is capable of seamless migration to any other during runtime. The integration of these domains and their constituent IEs forms the fundamental building blocks of aerOS. This architecture leverages these building blocks to deploy a comprehensive suite of services, all the way from edge to cloud, thereby creating a federated environment exposed as a continuum. This structured approach ensures a scalable framework that embodies the capabilities expected from a Meta-OS. It supports the orchestration of diverse and dynamic IoT application requirements across the continuum, as well as the orchestration of the underlying resources providing the hosting environment for these applications.

To facilitate the seamless integration of resources and the deployment and management of services across all domains—from the far edge to the cloud—two key concepts play a prominent role: **federation and orchestration**. aerOS incorporates its own design, perspective, and implementation of these concepts as functional enablers for the realization of an efficiently governed continuum. Their interdependent and mutually supportive integration establishes an innovative federated orchestration process. Before detailing the tangible

building blocks that constitute the aerOS stack, it is essential to explore these concepts from an aerOS perspective and understand their integration within the aerOS architecture.

Federation plays a crucial role in leveraging the collective capabilities and resources across multiple aerOS domains within the IoT-Edge-Cloud environment. It is built as a set of services within each domain which collaboratively establish the backbone to support resource sharing across the continuum. This is primarily built on the capability to publish and share, under a common and interpretable schema, all information regarding capabilities and status of all constituting elements of each domain. This opens the possibility to components that need it, regardless of their hosting domain, to have on-demand access to this information. This transparent access empowers components to make informed decisions regarding resource engagement, even beyond the boundaries of their respective domains. By breaking down silos and enabling cross-domain communication, federation promotes efficient resource utilization and collaboration, ultimately enhancing the overall functionality and performance of the aerOS ecosystem.

aerOS federation is implemented based on a **"Federator"** enabler (component) included within each aerOS domain and a central registry informed of all integrated domains in the continuum. The Federator is implemented as a set of services and functionalities, facilitating the bidirectional exchange of information with other domains of the continuum. The central registry is located in the Entrypoint domain (see 5.3.2), and its role is to keep an inventory of all integrated domains and promote new domains registration and connection with those that are already part of the continuum. Once the registrations are set, all continuum information is disseminated across aerOS domains, through the federator functionalities inherent to each domain. This distributed approach mitigates the risk of single points of failure and minimizes runtime dependencies, ensuring the robustness and reliability of the federation framework. Further details regarding this design are elaborated in section 5.4.3.

Federation incorporates mechanisms for efficiently transporting information and facilitates resource sharing across all domains within the aerOS ecosystem. Its underlying framework leverages functionalities provided by the various aerOS fabrics discussed earlier. Microservices, as part of these fabrics, are utilized to retrieve data, expose information, and facilitate resource sharing.

While federation provides the essence of continuum, **Orchestration** is the complementary counterpart which enables aerOS to act as a Meta-OS over the continuum. It is vital for managing and coordinating the deployment and execution of containerized workloads across multiple domains spanning from the edge to the cloud. Building upon the foundation laid by federation, aerOS orchestration elevates standard orchestration principles, particularly those of CNCF, by integrating distributed AI-driven decision support systems, trust services, and knowledge graph-based aerOS state models. This integration empowers aerOS orchestration to optimally allocate workload placement based on a comprehensive understanding of resource availability and capabilities across the continuum. Leveraging the capabilities provided by federation, aerOS orchestration exploits cross-domain communication facilities to coordinate activities over resources situated in various administrative domains throughout the aerOS ecosystem. Furthermore, facilitated by the aerOS Data Fabric enabled as part of the federation process, orchestrator components gain a holistic view of resource availability and capabilities across the entire continuum in real-time. This extensive information equips orchestrators with a multitude of options for workload placement decisions.

In each aerOS domain, an orchestrator component plays a pivotal role in making informed decisions regarding workload placement. The architecture of the aerOS orchestrator comprises a dual-layered design. At the higher layer, the **High-Level Orchestrator** (HLO) receives deployment requests using a templated descriptive model. Supported by AI algorithms and trust management components, as well as the privileged access to real-time information across all domains provided by federation functionalities, the HLO makes well-informed decisions on workload placement. If local Infrastructure Elements (see 5.4.1) can accommodate the workload, the HLO provides a suitable Deployment Decision Blueprint to the **Low-Level Orchestrator** (LLO), which leverages local deployment facilities within the domain to execute the deployment. Conversely, if a remote Infrastructure Element is deemed more appropriate, the request is forwarded to the remote high-level orchestrator, to take on with the deployment using the respective LLO.

A key takeaway regarding aerOS orchestration is its inherently distributed nature, where each orchestrator operates within its domain's jurisdiction while also benefiting from real-time knowledge of all domains' statuses and capabilities thanks to aerOS federation and the Data Fabric support. This enables orchestrators to make

informed decisions and access remote orchestrators to deploy workloads based on specific demands and resource availability. Finally, the innovative architectural decision to implement orchestration in two layers—where the higher layer integrates pluggable decision support systems such as AI and trust components—ensures efficient and secure workload placement, driven by sophisticated decision-making processes.

The functionalities described above are realised through a cohesive set of interconnected functional components residing within the aerOS ecosystem. These components are not confined to specific identifiable elements, but are distributed across all aerOS domains, residing on some of the contained Infrastructure Elements. Collectively, these components provide the aerOS Federated Orchestration capabilities, as a framework, spanning from edge to cloud, which provides the basis for management and orchestration of all resources and deployed functionalities across the continuum. Although federation and orchestration are closely interrelated, for the purpose of this document, each concept is referred separately to provide a more detailed and granular description of their functionality. This approach allows to delve into specific aspects, while maintaining an understanding of their cohesive nature within the aerOS ecosystem.

Concluding **aerOS Federation and Orchestration** concepts, and before delving into the detailed description of the aerOS building blocks, which provide the foundational implementation infrastructure, it is essential to underscore the significance of the aerOS Data Fabric and the knowledge-graph based model representation and their importance supporting o process over a federated continuum. These concepts are pivotal for depicting the state of aerOS resources and their relationships across the continuum.

Each aerOS domain integrates an aerOS Data Fabric component, which empowers any interested consumer within the domain—whether it be the high-level orchestrator, the trust manager, or an AI decision support component—to request and receive data seamlessly from the continuum. This approach eliminates the need for these components to directly access other domains or data producers across the continuum. Instead, a local Data Fabric agent handles all the underlying complexities, ensuring efficient and secure data retrieval.

The technologies and protocols developed and integrated for the realization of the Data Fabric enable the creation of the aerOS distributed state repository. This repository facilitates various methods of data interaction: consumers can poll for data, receive notifications about subscribed changes, or be updated based on established registrations. These interactions can occur across different administrative domains, extending from the edge to the cloud.

A particularly interesting aspect of the aerOS Data Fabric is its ability to provide updates through multiple mechanisms while maintaining a unified and coherent view of the continuum. The choice of a knowledge-graph model plays a crucial role here, allowing the representation and querying of infrastructure within a context of dynamically related resources. This model enables resources to be characterized and connected by a set of properties that are indicative of their most important features. This characterization is critical in determining the eligibility of resources for task execution, making the knowledge-graph model an invaluable tool for efficient resource management and orchestration within the aerOS continuum.

In summary, the aerOS Data Fabric and the knowledge-graph based model representation are key enablers of the aerOS ecosystem, providing a robust framework for data management and resource orchestration. These components ensure that each domain can interact with the continuum in a seamless, efficient, and context-aware manner, significantly enhancing the overall functionality and performance of the aerOS Meta-OS.

aerOS stack and building blocks

The **Infrastructure Element** (IE) is the fundamental building block of the aerOS system, providing the essential computational unit required to host and support the deployment of workloads. It can be any physical or virtual entity capable of supporting containerized workloads, with aerOS offering dedicated Low-Level Orchestrators (LLOs) to manage this diversity. IEs are deployed within or connected to an aerOS domain, playing a crucial role in hosting, and executing containerized applications or services. As the minimal execution unit, IEs form the base of the aerOS stack by exposing the core capabilities necessary for workload execution. These are enhanced with a set of lightweight aerOS self-intelligent enablers, as detailed in section 5.5.5. Each IE is equipped to support containerized workloads, and it is agnostic of the purpose of these workloads. This ensures that IEs can fulfil various roles within a domain, executing workloads regardless of their specific purpose. These workloads can range from vertical-oriented IoT applications to components of the aerOS Data Fabric, parts of the orchestrator, or any other tasks. IEs contribute their capabilities to the network, with the compute fabric and

service fabric taking over to deploy and manage the most suitable and appropriate services. This structured approach allows IEs to provide their resources in a way that enables seamless integration and efficient workload management across the aerOS ecosystem. The flexibility and robustness of IEs ensure that they can adapt to various demands, supporting the dynamic and heterogeneous nature of the IoT-Edge-Cloud continuum.

An **aerOS Domain** constitutes a complete aerOS runtime environment, formed by at least one or more Infrastructure Elements (IEs). There are two prerequisites that must be met for a set of connected compute and network resources to qualify as an aerOS domain. The first prerequisite is that these compute resources are integrated as IEs, as described earlier. This integration ensures that they can support workload execution and provide a minimum set of aerOS integration capabilities, such as manageable network functionality. These IEs act as the fundamental units within the domain, enabling the deployment and execution of containerized applications and services. The second prerequisite is the deployment of a core set of basic aerOS services on top of these IEs. This means that **an aerOS domain can be defined as a group of IEs that share the same set and instance of basic aerOS services**, ensuring a cohesive and functional execution environment. The basic aerOS services running within each domain are distilled from the previous discussions and include:

- **Federation service**, provided by the federator component enables the bidirectional exchange of information with the other domains of the continuum. Exchange of information regarding the status of domain and IEs facilitates resource sharing, workload distribution, and interoperability. It is responsible for managing subscriptions, and registrations to other aerOS domains which are instrumented to discover resources across the continuum. It is built as a set of coworking microservices offered by aerOS fabrics. Underlying protocol used for these exchanges is NGSI-LD, discussed in section 5.4.3.1.
- **Orchestration service** which is crucial for managing the deployment of IoT services based on user-defined intentions. This service translates user requests into actual deployment instructions and execute them on Infrastructure Elements (IEs). This orchestration process operates at two levels: the high level, which receives constraints and queries the continuum and handles decision-making complexities with the support of AI-enriched decision support systems, and the low-level which has the actual access to underlying IEs and translates decision to implementation directives. Based on the federation services, orchestration service within each domain acknowledges the state of IEs across the continuum and can produce decisions including remote domains resources.
- **Data Fabric services**, enable seamless data integration and accessibility within the domain and across the continuum, supporting advanced data management and query capabilities. They manage the identification, collection, and integration of data in an interoperable manner. Data Fabric services facilitate seamless data access, for all aerOS consumers. This ensures that data is readily available and accessible across the entire aerOS ecosystem, supporting efficient and informed decision-making. Raw data concerning all aspects of aerOS, including domain status and IoT applications, are ingested and transformed into a common, interpretable format. This standardized information forms the substrate for the federation service, which is responsible for sharing and propagating this data across the entire continuum.
- **Cybersecurity services** which enforce authentication, authorization, and access control policies based on roles and identities. These services ensure secure access to all domain resources and validate requests in close coordination with the Entrypoint domain's policies and users' registries. By integrating robust security protocols, they maintain the integrity and confidentiality of data and services across the aerOS continuum.

Additionally, a range of aerOS services integrated within each domain provide an intelligence layer to the aerOS Meta-OS. These intelligent services enhance the system's ability to make informed decisions, optimize resource allocation, and improve overall operational efficiency.

- **AI decision support** services, which leverage data retrieved by the Data Fabric to provide critical input to the orchestrator. These services analyse and interpret the data to recommend optimal workload placements, whether locally within the current domain or across other domains, ensuring efficient and effective resource utilization.

- **Trust management services**, which ensure the integrity and reliability of interactions within the aerOS ecosystem. These services validate and monitor the trustworthiness of IEs within each domain and provide insights to the decision engine regarding the reliability of candidate hosting resources.
- **Embedded analytic tools**, which provide real-time data analysis and insights directly within the aerOS ecosystem. These services enable continuous monitoring, assessment, and optimization of system performance and IoT applications, empowering stakeholders with actionable intelligence to enhance decision-making and operational efficiency.

Finally, each aerOS domain, in addition to these core services, exposes a comprehensive and efficient Application Programming Interface (API) to facilitate communication with stakeholders, agents, and other domains.

Figure 5 provides a comprehensive diagram reflecting the aerOS architecture. This diagram illustrates the formation of the continuum through the seamless federation of aerOS domains. It highlights the operations within each domain and the unified layer provided by the aerOS Meta-OS, offering a cohesive environment for IoT developers.

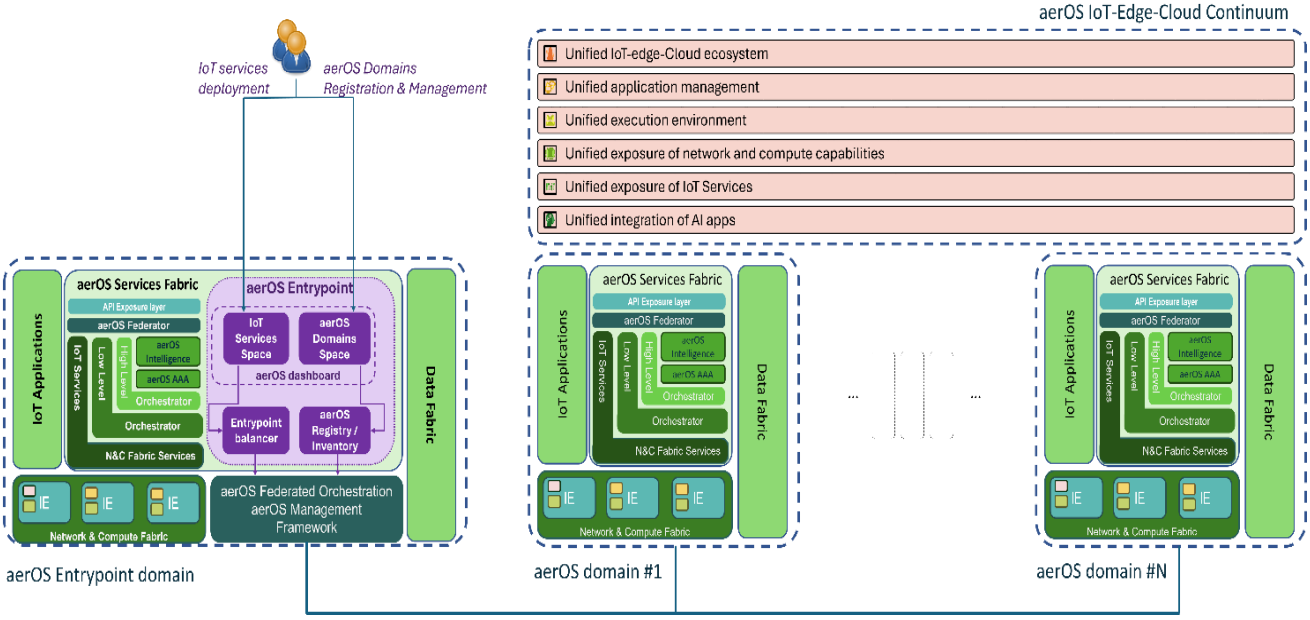


Figure 5. aerOS architecture.

"Entrypoint domain", reserves an extra mention, in this narrative. It is enriched with additional components related to management and AAA capabilities, designating this unique node as a key aerOS point of presence. This domain hosts the **aerOS Management Portal**, which serves as the primary user interface to the system, providing access to Meta-OS management functionalities. It integrates several crucial features: the aerOS access dashboard, a registry of users and policies, an inventory of registered domains, and a mechanism to support balanced distribution of user service deployment requests across the continuum. The dashboard is the sole, graphical, entrypoint to the aerOS ecosystem for all stakeholders, acting as a single window for managing the continuum, similar to a terminal or shell in a traditional OS. It maintains connectivity to the Identity and Authorization Manager and provides a user-friendly administrative interface for the User Registry, enabling the creation, editing, and deletion of users, as well as role management. Furthermore, the dashboard offers space offering functionalities related to the management and visual representation of the current state of aerOS domains and the continuum. The uniqueness of this "special" aerOS domain does not imply immutability. It can be exchanged or migrated to another aerOS domain at any time, either due to failure or by administrative choice, ensuring flexibility and resilience in the system's management infrastructure.

By design **aerOS is not a centralised solution** and opts for a fully federated and decentralized architecture. To achieve this, it leverages the **aerOS Management Framework**. This framework, comprising the Management Portal and a suite of mechanisms, facilitates the creation and upkeep of federation connections among the numerous aerOS domains constituting the continuum. Each time a new domain joins the continuum, the Management Framework orchestrates the necessary networking procedures to ensure its seamless discovery and connection with existing domains.

The subsequent sections delve into the intricacies of aerOS building blocks and components, describing their pivotal roles within the system. These components are carefully designed to orchestrate a federated continuum of resources spanning from the edge to the cloud. The overarching aim is to provide a flexible, trusted, and unified development and execution environment tailored for IoT services developers across diverse industry verticals.

5.3. Conformance of an aerOS continuum

As it has been explained, aerOS Meta-OS revolves around the concept of domains. These are virtual groupings that gather one or more computing nodes (Infrastructure Elements, see section 5.4.1).

A "domain" is a relevant figure in aerOS technology, outstanding as a crucial aspect whenever designing a continuum. It can be defined as "*a set of elements (that can run containerised workloads) that are judiciously grouped together based on certain criteria, and that allow the Meta-OS system administrator to organise the deployment and other technological decisions accordingly*". In addition, each domain must contain a unique instance of the full set of aerOS basic services.

5.3.1. Laying out the domains in a desired continuum:

Depending on various aspects, the decision on what (should) constitute a domain in a specific case will be different. Even though there are some common guidelines that have been envisaged by the aerOS developers, the final decision on how to group computing nodes to form a domain is completely open and up to the adopter's system administrator.

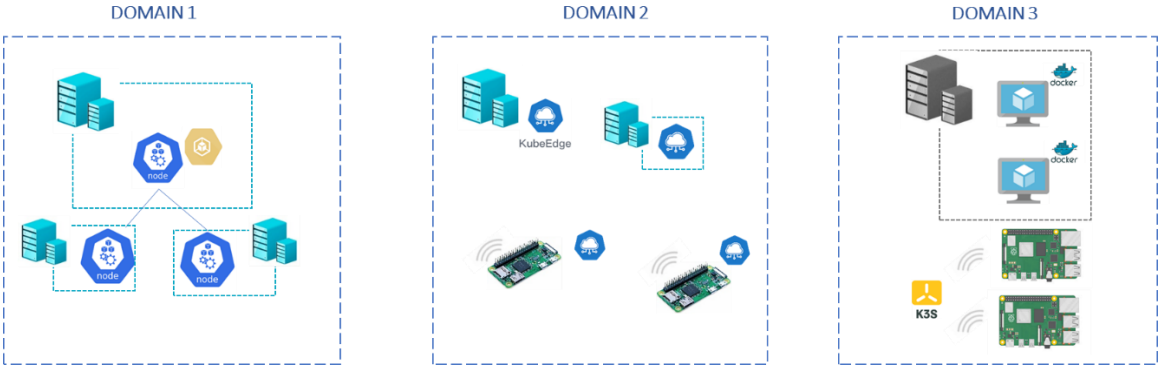


Figure 6. Example of domains topology design in an aerOS continuum

Pragmatically, the decision on how to design the domains in a use case serves to delimit and put boundaries to the scope of aerOS installation, and to understand where certain components should be placed.

As a reference, the decision might be guided by the following orientations:

- **Geographically**: it might be convenient to create a domain that includes all the computing resources in a specific geographical area. For instance, if a cloud provider owns servers (among other equipment) in different regions, one domain could be created per each region, facilitating the management and the incorporation/removal of those elements into/from a continuum. Another example would be within a campus.

- **Per administrative scope**: if a continuum includes computing elements that are owned by different companies, it might be interesting to create such boundaries via the creation of specific per-entity domains. There, the domain of company A would still be able to share workloads, service and data with company B, but still several managerial and operational aspects (such as which aerOS installations are needed) would remain independent.
- **Based on container management framework**: another valid criterion for designing the domains in a continuum is the container management framework underlying the computing nodes (IEs). For instance, if a continuum must include certain nodes that are part of a K8s cluster, others that are Virtual Machines (with capacity to execute containerised workloads), others that require KubeEdge cloud and edge counterparts, or any other possible cases, this could guide the decision as well. Here, it is relevant to point out that aerOS is capable of handling different container frameworks in the same domain (thanks to the versatility of the orchestration schema – see section 5.4.2), but, often, those may impose certain restrictions that may require specific treatment. In such cases, the management benefits from the separation in different domains.
- **Depending on the network**: a typical of the aforementioned restrictions is the network. A relevant pre-condition is that every domain is accessible (i.e., has its own IP or addressable namespace – more details in section 5.5.1). aerOS assumes that companies stick to their pre-existent network configuration and topologies. Therefore, a design decision may come from the limitations/options available in that regard.
- **Other**: although the previous have been envisaged as the most common decision criteria, it does not prevent system administrators to deepen their considerations:
    - Operational decision: for instance, in manufacturing or logistic environments, a domain could comprise all elements that stay in the same production line or process area (e.g., parking yard).
    - Per type of computing node: it may be relevant to separate domains based on the hardware or the type of nodes. In some cases, specific conditions by COTS or commercial equipment might be of application, driving the domain topology design.
    - Per tier in the continuum: elements pertaining to the cloud (e.g., in the case of using AWS, Azure or other cloud VPS), edge elements, or IoT tier could be also valid examples, as long as every domain will comply with the minimum requirements (see section 6.5).

Often, the design of domains in a continuum will be a combination of the previous, especially in large scale scenarios. Therefore, the step of defining and selecting domains is crucial for the rest of Meta-OS configuration and deployment.

Several examples on how (and why) certain domain topologies have been designed (in aerOS pilots) can be found in section 7.2.

5.3.2. Entrypoint domain selection

A crucial step in the conformance of an aerOS continuum is the selection of the entrypoint domain. Once the layout of the different domains has been established, the system owner of the adopter entity must decide which of those should act as the domain holding special characteristics.

aerOS has been designed to be as decentralized as possible, and indeed the workload and data orchestration, execution and management is done without prejudice of location or position of the domain in the continuum. However, due to the very nature of distributed systems, one of the domains must act as the entrypoint.

In aerOS, the **entrypoint domain** is the one containing the necessary singleton elements for the proper **management of the continuum**. Reasonably, there are some aspects that do not match the fully distributed paradigm, such as the aerOS portal, which is a web service that contains the UI through which the user handles different aspects of the continuum, its services and the data. Also, although security matters are highly decentralized (every domain handles the roles, permissions and access profiles to different aerOS' domains features and data), there is still the need of a federating entity gathering together common policies for a continuum.

In short, the **entrypoint holds the relevance of the direct interaction with the continuum IT professionals**, acting as the landing area and exposing via a UI the main management capabilities of the continuum.

Noteworthy, all underlying domains will perform all operations of the continuum acting as equal peers (including the entrypoint).

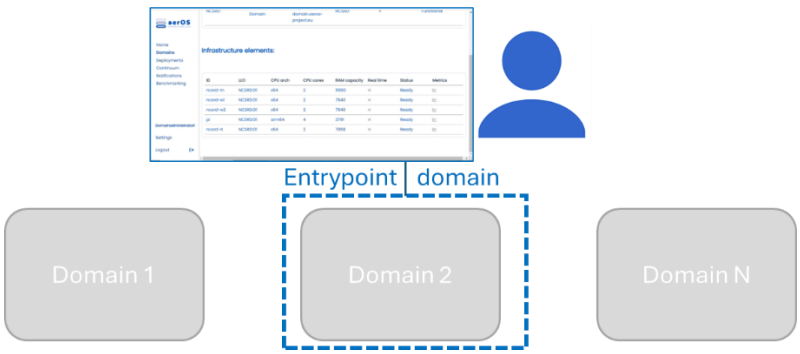


Figure 7. Concept of entrypoint domain in an aerOS continuum

It is important to highlight here that only small set of components will live in an entrypoint domain compared to all the rest. Everything related to data acquisition, data transformation, data exposition, federation, nodes' exposure for potential microservice allocation, monitoring, and smart capabilities remains equal to all nodes. Therefore, characteristics such as data access, trust or interdomain communication remail unaltered regardless the entrypoint domain selection.

Without digging in detail of the components (see 5.5 and 5.6 for that purpose), here is the list of the elements of the Meta-OS of aerOS that will need to mandatorily live in the entrypoint domain:

- **aerOS Management Portal**: very relevant element as it acts as the "gate" to the continuum data and management, including the commissioning of workloads orchestration.
- **aerOS Identity Manager**: relying on mature open-source solutions, it becomes necessary to share common information about profiles from all the domains in the continuum.
- **Active Directory Database** to feed the two previous elements and to act as rooting element for users, profiles and global information about the users of aerOS.

Remarkably, one of the paramount requirements of aerOS is flexibility, therefore **there is no mandate on which domain should act as the entrypoint**. Resting on various aspects, the decision on which domain in a continuum should act as entrypoint may be different. Even though there are some common guidelines that have been envisaged by the aerOS developers, the **final decision** on where to place the entrypoint domains (i.e., where to place the components of the list above) is completely open and **up to the adopter's system administrator**. Since any domain in the continuum can act as entrypoint as per user's choice, impact in terms of KPIs' fulfilment can vary. Also, there will exist variability in the impact of the decision depending on the number of actionable aerOS nodes or legacy nodes. From another perspective, data transactions efficiency may come into play when intertwined with network considerations. Also, when the design of a continuum is oriented to data-intensive applications, their structure, format, semantics, etc. are important elements to consider for the decision. Also, errors in transactions could be related to the necessary inter-domain communication related to the entrypoint.

The following table offers some hints to serve as guidance for selecting an entrypoint in a continuum.

Table 1. aerOS Terminology table

Angle taken from adopter	Guidelines and recommendation	Impact in KPIs
User Performance: it is desired to maximize QoS and QoE for the user managing the continuum.	Here, the recommendation is to settle the entrypoint domain that hat has better inbound and outbound access from the internet, assuming that the user will be connecting from a remote machine (browser's host). Therefore, the QoE will be increased. However, it is worth mentioning that for different	QoE ↑ QoS-UI ↑ More agile update of security users and policies ↑

It must be considered that a user will access via the public internet to the management portal.	operations (e.g., data delays) will depend on other factors as well.	
Overall system performance: two aspects outstand: (1) the more capacity in the entrypoint, the better manageability, (2) aerOS is a distributed system relying on federation of domains.	The guidelines here will be to place the entrypoint domain where most abundant computing resources rely. Usually, this leads to select the entrypoint in the domain that is in (or closer to) the cloud, as it will entail: (i) better capacity to install larger software, (ii) more availability, (iii) better network connection, allowing for the federation to experiment smaller delays.	Response time in orchestration ↓ NFV and SDN potential ↑ Cloud-native compatibility ↑
Data angle: aerOS allows to inspect existent data via the portal, using EAT, Grafana and other tools. Also, there will be cases with heavy batch data loads, that clearly benefit from entrypoint closeness.	If: (i) the expected use cases are data intensive, at rest and at move, such as managing batch and stream, pub/sub schemas or direct query access and, (ii) the origin of data is usually known, the recommendation is to place the entrypoint at the same domain where data will be coming from.	Consistency ↑ Data losses ↓ Simultaneous data pipelines ↑ Data transfer rate ↑
Exploitation angle: in a case where a company wishes to employ aerOS to ensure compatibility with their already existent tools, it is important to place the entrypoint correctly.	The recommendation is to spot the entrypoint domain where enterprise applications are located. This roots on the potential sharing of permissions, cybersecurity policies, roles-based access, etc. Also, placing the manageability web dashboard close to the already functional elements will reduce the risk of losses, as same backup/maintenance mechanisms will be easier to replicate.	Unsuccessful correct authentications ↓ Efficiency of API Gateways ↑ Trust level ↑
Developers' usability: the installation of aerOS and further services requires access to certain repositories, and may imply modifications to low-level capacities in the IEs forming the continuum.	Here, the recommendation is to decide the entrypoint where: (i) the easier management access to IEs (via SSH, K8s management tools, …) is, (ii) the more permissions are provided to developers – e.g., access to system kernel or network interfaces, installation options, (iii) better network connectivity to access package repositories.	Diversity of IEs supported ↑ IoT scenarios covered ↑ Self-features in nodes ↑
Multi-stakeholder presence and preference: a continuum can (and most likely will) contain computing nodes and data coming from different owner (stakeholders) that wish to share their resources for a common purpose. Such scenarios entail some specific reflections.	In general, in those cases it is recommended to place the entrypoint domain in one domain that is owned by the entity with the most prominent role (coordinator of a consortium, main representative of a cluster, etc.). Once this is decided, the previous considerations would apply. However, it is accepted in aerOS (in those cases) to propose the existence of multiple entrypoint portals, each of them acting over the "continuum" of a single stakeholder, and performing the federation over mutual resources and data.	Number of stakeholders deploying aerOS ↑ Data exploitation ↑ Federation potential ↑
Overall analysis: A relevant consideration that applies to all reflection is that, in general, a wise selection for the entrypoint is the one that has more computation, storage and network capacity. Usually, those criteria imply that cloud domains (high availability, reliability, bandwidth) are often preferred. Also, if the adopter entity knows in advance where the services will be deployed most of the time, or if the data location is acquainted for, the selection of the entrypoint domain will be more straightforward.		

However, due to the intelligent design of aerOS decentralization capabilities, there are elements that SHOULD NOT be a factor for the decision. This is because most of aerOS basic and auxiliary services are conceived to be present in all domains. Some of those factors could be:

(i) Type of nodes included in the domain: as they will appear to the orchestration schema equal to the rest (defined by their own characteristics, without preferences).

(ii) Consistency of formats in the data, or the convergence of data sets into a semantically expressed ontology. Tools such as semantic annotator and semantic translator may live in any domain (entrypoint or not).

(iii) Data Fabric features. Every domain will have its own Data Fabric, that will include data federation, data security, data protection, etc.

Trust in message exchange – IOTA Tangle will be present in the continuum materialised in a *hornet* node per Infrastructure Element, therefore regardless the domain type (entrypoint or not), the desired messages will be immutable and traceable.

In the next figure, a simpler example can be found. Departing from the demonstrator showcased in the mid-term review (April 10th, 2024), an uncomplicated rationale can be explained.

There, three domains came into play. One was focused on IoT sensing and actuation while the other two provided useful computing equipment. One of the latter offered nodes tiered on edge and far-edge spots of the continuum, sticking to unreliable network (4G SIM card underlying a low-power wireless connection), while the other rested at a cloud datacenter. Based on reflections #1 and #2 in the table above, and considering that exploitation and data diversity were not a determinant factor, the selection of the entrypoint fell to the cloud domain. Further information about the particularities of the demonstrator can be found in Section 7.1.

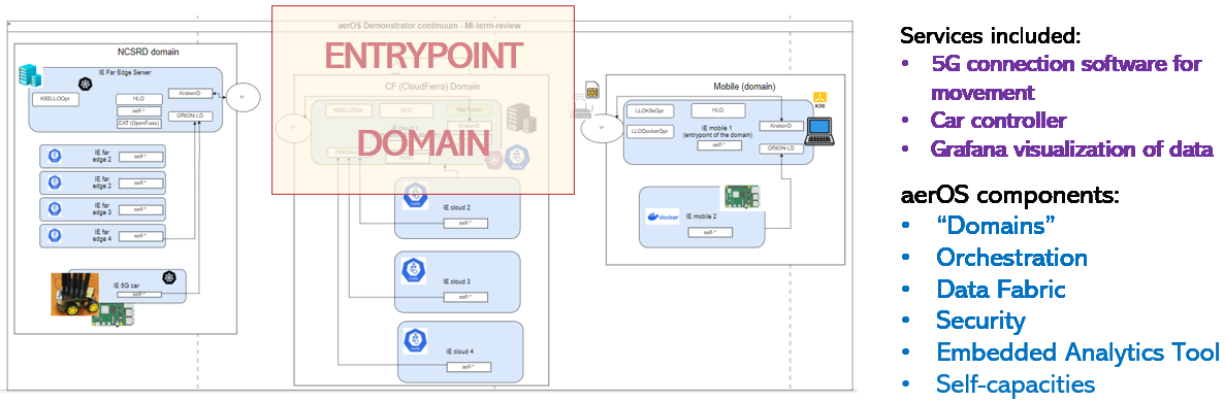


Figure 8. Simple example of entrypoint domain selection rationale

5.3.3. Next steps after continuum conformance

At this stage, system owners will be ready to tackle the deployment of aerOS. The philosophy of the installation and usage strongly relies on those domains (see conceptual installation procedure in deliverable D5.2 "Integration, evaluation plan and KPIs definition (2)"). Depending on whether a node belongs to one domain or to another, and also the role that it holds within it, will contain certain basic and auxiliary services or other.

Remarkably, once domains have been selected, and the components of aerOS architecture have been properly placed, those must be combined together to form the actual continuum. The mechanism through which such combination (i.e., **federation**) takes place in aerOS is thoroughly described in section 5.4.3.

5.4. aerOS stack and runtime

To develop aerOS, following a Meta-OS approach that encompasses all continuum aspects discussed earlier, the system is structured around key fundamental concepts. Section 5.3 has offered a comprehensive overview of the aerOS system, consolidating such main concepts and building blocks.

The aerOS runtime involves any parts of the Meta-OS of aerOS that make the global functioning of the resource and service fabric to function. In a way, the aerOS runtime covers those essential aspects of the Meta-OS around which the rest of the components (basic and auxiliary services) orbit. The conception of this runtime is a novelty introduced by the aerOS project can be conceived as the first step to consider an IoT-Edge-Cloud IT ecosystem "aerOS-compliant" or "powered by aerOS".

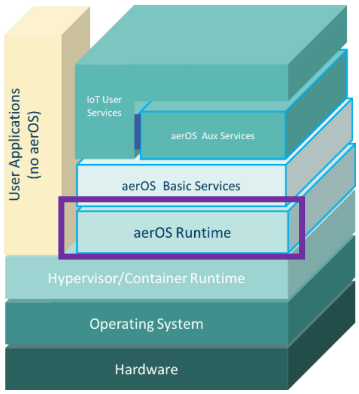


Figure 9. aerOS runtime component as part of aerOS stack

The same way the Linux-based computers work over a set of processes interacting with a set of instructions of the processor (the *kernel*) to allow the management of the filesystem, the network, interfaces, etc. the **aerOS runtime** allows to handle the underlying complexity derived from widespread, heterogeneous resources into a practical, agreed, standardised, canonical set of methods and tools that permit the interaction (southbound) and the creation of services on top of it (northbound).

In practical terms, it can be said that the installation of an aerOS runtime in the computing elements of an IoT-Edge-Cloud deployment becomes the first step for continuously using the continuum, thus becoming the essence of the Meta-OS. Therefore, a distributed system that builds on top the presented runtime is what can be called "aerOS-based".

The goals of the aerOS runtime are the following:

- To abstract the underlying heterogeneous resources so that they are seen (and managed) uniformly (taking advantage of the concept of IE).
- To manage diverse operating systems (e.g., Ubuntu, custom Yocto-based OS), container runtimes (e.g., Docker, containerd) or any deployment management layer above (e.g., Kubernetes), so that the effective execution of workloads is achieved as expected.
- To be able to orchestrate such workload execution based on requirements (intentions as *blueprints*) expressed by users (deployers) but also considering the global evolution of the continuum and advancing eventualities thanks to ML models.
- To introduce that smartness all around the continuum, ensuring an actual decentralisation in the orchestration, avoiding single points of failure, and truly empowering the edge areas in a distributed ecosystem.
- To make the resources (although quite different and geographically dispersed) discoverable from any point of the continuum; thus, allowing a quick and efficient re-distribution of the workflows in a system.
- To make sure that certain actions related to the continuum (e.g., reorganization) are traceable and immutable.

The following sub-sections dig deeper in the particular mechanisms that conform the aerOS runtime.

5.4.1. aerOS Infrastructure Element

As mentioned, the essential functioning of the Meta-OS of aerOS is based on the establishment of computing nodes (infrastructure Elements) and how they group together (in domains). In aerOS, an IE is the most granular entity able to be controlled and managed by the Meta-OS, conceived as the most atomic element for computing, network and data orchestration in the continuum.

The whole organisation of the services in aerOS assumes that there are entities (forming part of the continuum) that can be relied to offload the execution of workloads upon. Those entities (IEs) are heterogeneous, as is the continuum, but have several aspects in common. The features that must characterise an IE of aerOS are:

- Capacity of running containerised workloads.
- Existence of an underlying OS where upper software can be installed.
- Availability of computing, storage, and networking capabilities.
- Availability of network interfaces that can be controlled.
- Capacity of hosting an API exposing their monitoring and services.
- Capacity of OS accessibility and manageability at administrator level.

In an IoT-Edge-Cloud deployment, every computing-capable piece of equipment (virtualised or not) meeting the previous characteristics would qualify as a potential IE of aerOS. This way, this concept can embrace a wide variety of potential computation targets. Figure 8 depicts only a few examples of potential IEs in aerOS. Some of those examples are: i) nodes in a K8s cluster (*master* and *workers* alike), ii) an edge-computing single-board computer (SBC) with a preinstalled OS (e.g., Raspberry Pi), iii) nodes in a KubeEdge deployment (*cloud* and *edge* alike), and iv) virtual machines. Other examples might be envisioned.

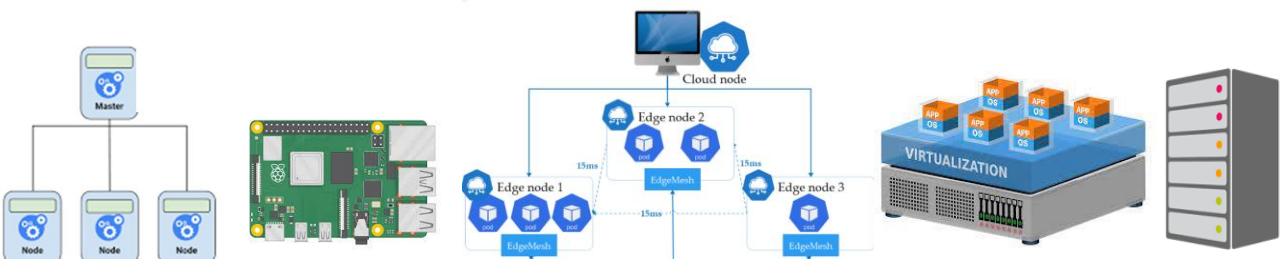


Figure 10. Possible Infrastructure Elements in a continuum

The previous examples already help to realise that the concept of IE has been established to nominate resources that can live across the whole continuum, starting from resource-constrained devices up to traditional (and new) edge computing equipment and local data centres or large clouds. This way, the first step towards a uniform management of the underlying complexity of the continuum is laid out.

However, all IEs will have their peculiarities. In order to consider those in the Meta-OS functioning, aerOS is developing a novel, semantic description for expressing the different capacities and characteristics of a single IE in an aerOS continuum (such semantic data model is elaborated in section 5.4.3.3). This includes total resources, ownership, list of peripherals available, computing capacity, container management framework, and OS, among many others. Within these definitions, enough descriptive fields have been established to indicate aspects about where in the continuum the specific IE is located. As said, IEs may exist in the whole arch of the topology of the continuum. All previous information can be used to populate the "tag" about which IE flavour it is. This way, IEs are more easily identifiable depending on the spot in the continuum that they are located in, opening up advanced orchestration capacities. For instance, it might be established that the "far-edge IEs" cannot accept heavy workloads such as full-fledged, large databases.

A relevant, specific case related to IEs in the architecture of aerOS are the **IoT devices**. As indicated in Section 4.1, IoT devices are a crucial part of "continuum" deployments, as they can be considered both productive elements (providing context, monitoring, exploitable data or actuation capacities) and also potential receptor elements that could be orchestrated globally in the Meta-OS. In such case, the characterisation of IoT devices within the continuum topology/architecture is very simple. In the case that an IoT device entails enough "smart" capabilities to meet the requirements of the list above (has an OS, can run containerised workloads, can expose an API and can be accessed in terms of storage and network), **it must become actual part of the continuum as an IE**. In the case that any of the previous were not met, the IoT device would need to be attached to a proper IE. This is the case of usual IoT devices such as certain sensors. Here, the expected set up would be for them to directly attach (connect) to their nearest IE (for instance, an edge element with I/O capacity to connect sensors via some communication protocol).

In addition, although the IE is the most granular entity in aerOS, there is no need of directly mapping one single "computing entity" to one aerOS IE (1:1 relation). It is possible that several computing entities are joined and seen (from the continuum perspective) as a single IE. These specific configurations are a user's choice (system administrator); thus, enhancing the trait of flexibility of the aerOS architecture.

As outlined in section 5.3, on top of IEs, another level of organisation emerges. In aerOS, the grouping of various IEs form the **domains**. These are characterised as a set of one or more IEs sharing a common instance of the basic services of aerOS among them.

The two previous descriptions are rather useful in terms of deployment organisation, as it has been designed (by architecture) that IEs and domains must stick to certain functioning roles (directly mapping to components installation):

- **IEs:** must incorporate HW/SW monitoring capacity, continuously providing to the whole Meta-OS the information about its status. Also, they must take an active role in terms of smart self-capacities (see 5.5.5), including the registration of judiciously selected events that must be traceable.
- **Domains:** must incorporate exposure capacities, being the essential assembly unit. It also must contain the intelligence to govern services within its boundaries (IEs that are part of it), and must deal with users, cybersecurity, data handling and serverless usability.

5.4.2. aerOS decentralised orchestration

At the very core of aerOS proposition is the capability of accomplishing smart, automatic, decentralised decision making in terms of service orchestration across the continuum. In the moment when all heterogeneous computing resources in such continuum are abstracted and accessible (as IEs) and the status of those resources is known across the ecosystem (see Section 5.4.3), the Meta-OS is ready to unleash its orchestration power. The other two sub-sections (Section 5.4.1, Section 5.4.3) expose how to deal with the former (IE, Distributed State Repository -DSR). Here, the approach behind aerOS orchestration is portrayed.

As per the original conception of aerOS, an orchestration process has been envisioned to oversee arranging, managing, and coordinating services, provisioned as part of applications, with a comprehensive management of both IT and logical network resources. Allocation and orchestration of logical resources executing a service chain requires solving constraint-based double optimisation problems, with data about application requirements and infrastructure as input. The discussed module includes the federated orchestration capacity, provided that a service cannot be executed in a local domain, of spanning deployments' requests along the continuum, having an overall view of it, and offloading to other domains.

After the analysis of the state-of-the-art and the evolution of the technical design of aerOS architecture (depicted in this document), which has been successfully tested in the first implementation approach (the MVP, see section 7.1.1), it has been decided that the decentralised decision-making must be carried out by **a clear two-level structured orchestrator**. In addition, the federation part of the deal is taken over by another, separate element (aerOS Federator – see Section 5.5.8).

According to this structure, the decentralised decision-making, materialised in services orchestration decisions, is divided into a **HLO** and a **LLO** (as mentioned in Section 5.2). The goal of these modules is to permeate the intelligence and flexibility of aerOS in the allocation of computing spots for the containerised workloads that are handled by the continuum. There, by making use of advanced AI algorithms that optimise parameters such as latency, it can be smartly decided which resources within the continuum to employ. Also, to ensure accuracy on decision-making, the principle of locality must be overcome, reaching the horizon of continuum visibility. Here comes into the scene the federated infrastructure of a distributed network of brokers, that allows observation of the current state of the whole continuum. The component in charge of connecting the previous (the different domains) is the **aerOS Federator**, and the conjunction of all the elements (HLO, LLO, aerOS Federator) is the **aerOS Federated Orchestrator.**

In this section, the focus is put on how the two-level orchestration will be implemented in aerOS. **This point is where a major part of aerOS novelty** lies. Figure 9 expresses the basic functional principle of it.

The aerOS decentralised orchestration decision-making process begins with the expression of an end user intention to deploy a vertical service (a non-aerOS, Basic or Auxiliary, service) in the previously established

IoT-Edge-Cloud computing continuum: a set of IEs running aerOS Runtime and Basic services grouped in domains. Because of the heterogeneity nature of the continuum, it is not possible to provide users complete freedom to express their potential deployment requirements. Thus, this deployment intention is expressed through the selection and specification of a finite set of predefined requirements, which have been properly defined after a thorough process which involved staff with previous experience in the subject (technical task leaders and use case scenarios leaders). After the definition of the deployment intention in the Management Portal by filling a form (New service deployment), this component converts the request into an *Intention Blueprint* (in TOSCA[1] format), which needs to be forwarded to the aerOS Federated Orchestrator module. There, the combination of HLO and LLO (supported by overall continuum perspective brought by the aerOS Federator) achieve the successful execution of the vertical service in an IE (or set of IEs) of the continuum.

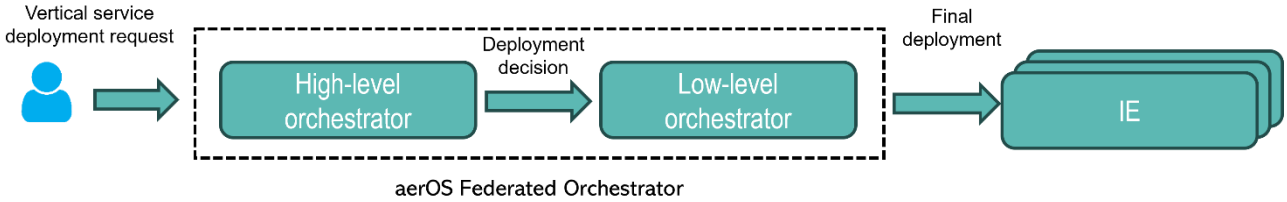


Figure 11. aerOS two-level structured orchestration for decentralised decision-making

It is worthwhile to describe the orchestration module in more detail. As mentioned, the decentralised decision-making orchestration module or the aerOS orchestration basic service is composed of two different modules:

- **High-Level Orchestrator (HLO):** this component follows a general approach for all the continuum, being independent from the IE types which compose a domain. Every domain has exactly one instance of HLO. This part of the orchestrator is in charge of taking the final deployment decision, thus deciding where each component of the required service will be deployed (specific IEs from specific domains). However, the HLO does not actually deploy the services, but sends its decision in the form of a °*Implementation Blueprint* to the specific LLOs, depending on the selected IEs.
- **Low-Level Orchestrator (LLO):** this component follows a specific/custom approach depending on the underlying container management framework of the IEs of a domain. In that sense, the *Implementation Blueprints* coming from an HLO will be interpreted differently by each LLO in order to provide the expected (deployment) functionality. In aerOS, the LLOs are based on the K8s Operator Pattern, which provide a framework to automatically control the lifecycle management of resources. This does not mean that LLOs only support the management of K8s workloads, but that actually leverage this widely used and tested resource management framework to finally reflect the desired state of services (to be deployed in a certain IE), with a common way to express the Implementation Blueprints (custom K8s Custom Resources) that is independent on the container management framework used in the selected IE, so that agnostic for the HLO. Therefore, a custom K8s Operator has been developed to cover the most common and state-of-the-art container management frameworks: Kubernetes, Docker and standalone containerd.

The first step of the process in Figure 10 (vertical service deployment request with specific requirements) is conducted in the aerOS Management Portal (described in depth in the section 5.5.8), which provides a user-friendly interface to facilitate the deployment request with determined requirements. The portal is only deployed in a single domain, named as *Entrypoint domain*. This Entrypoint domain does not act as a superior layer entity in the continuum. On the contrary, any domain in the continuum can act as an entrypoint, as per user's choice (see mitigation/migration concept in Section 5.3).

[1] https://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html

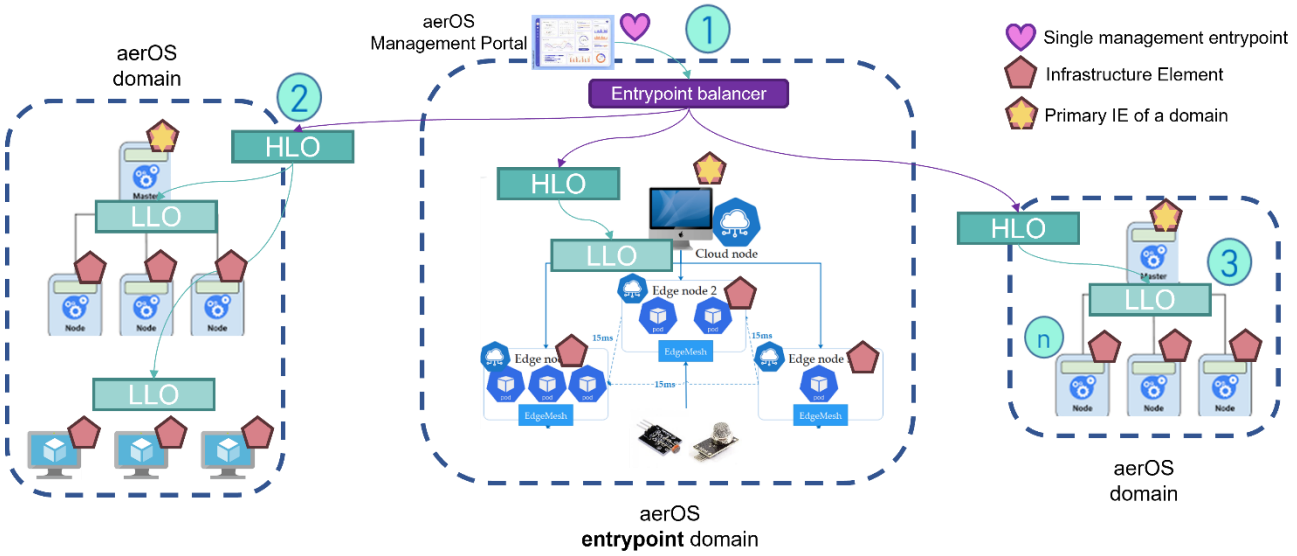


Figure 12. Entrypoint domains in decentralised decision-making of aerOS

1 After the definition of the deployment intention in the Management Portal, this component converts the request into an *Intention Blueprint*, which is further forwarded to the HLO. As explained above, each aerOS domain has a single instance of the HLO, so it is needed to decide to which domain will be forwarded the orchestration request to. Selecting a single domain to act as an "orchestration Entrypoint" would add a new centralization point to the architecture (thus, potential single point of failure), which collides with the decentralized nature of aerOS decision-making. Therefore, given how the network's load balancing has been approached in the state of the art [8], aerOS has foreseen the development of a specific component here, called *Entrypoint balancer* – which is based on the state-of-the-art least connection load balancing algorithms – that aims to relax the centralised needs of a unique, 1:1, direct relation Portal->HLO, towards a 1:N, fairly distributed approach. In fact, with this addition, the existence of an Entrypoint balancer emphasises the decentralised nature of aerOS orchestration. This way, regardless of the "Entrypoint domain" choice, the inception of orchestration (the HLO that will initiate the process) is balanced across domains. This conceptual approach is represented in Figure 10, in which the central domain has been picked for the sake of visual clarity.

2 The aerOS Management Portal interacts with the appropriate HLO (as selected by the Entrypoint balancer). The HLO receives the *Intention Blueprint* and executes the necessary logic to conduct the allocation decision, which is run for each service component that compose the service. This logic includes a pre-filtering engine (to dismiss the IEs that do not meet the specific requirements) and the usage of judiciously selected frugal ML algorithms[2] that optimise the allocation based on the current and forthcoming state of the continuum. For the latter to happen, the HLO benefits from the federation of domains exerted by aerOS Federator. In that regard, the HLO is aware of the domain to which belongs the selected IE to deploy the service component, so if the selected IE belongs to another domain, the deployment request is sent to the HLO of the selected domain. Finally, an *Implementation Blueprint* is generated and fed to the proper LLO(s).

Reflecting on the two previous points, it is worth to highlight the different between HLOs and the *Entrypoint balancer*. Directly requesting the HLO of the Entrypoint domain to be the unique entry gate for deciding the allocation would create unnecessary unbalance in the continuum. Simply put, if the HLO of a hypothetic domain (A) is always charged with the first decision (allocation within the domain A, or offloading to other domain), it might be overloaded, as it will be forced to perform certain calculations every time that a workload must be run in the continuum. However, creating a *load balancer* **before** directly requesting to the HLO of domain A, guarantees that other HLOs in the continuum are requested first, thus distributing better the "needs of the first processing of the decision". In other words, the main differences between both are:

[2] The algorithms to be used, their design, training, inference, etc. will be a matter of focus during the next months of the project, as an intersection of tasks T3.3 and T4.3.

- In every moment, there is only one *Entrypoint balancer* existing in the continuum (located exactly where the management portal is, and attached to it), whereas there are always as many HLOs as domains exist.
- The HLO interprets the workload that must be executed, the *Intention Blueprint*, the state of the continuum, etc. and takes an allocation decision. This requires (likely, heavy) processing to be performed by the IE where the HLO lives. On the contrary, the *Entrypoint balancer* doesn't analyse the information of the workload to be executed. It just applies a (rather simple) algorithm to forward the first request to one HLO in the continuum (not based on the operation information of the particular case, but on balancing rules). Thus, not heavy processing is required.

Thus, in essence, the existence of an Entrypoint balancer does not overlap the role of the HLO. In contrast, they are mutually complementary to deliver a more efficient, decentralised continuum.

3 Afterwards, LLO(s) (several may exist in a domain, one for each container management framework that run the IEs) receives the *Implementation Blueprint* (it follows the same format instead of the LLO type) and interprets it in a way that actual workload deployment can take place on its underlying, controlled IEs. Therefore, custom developments have been made to achieve the LLOs expected functionalities, which are described insightfully in the deliverable "D3.2 Intermediate distributed compute infrastructure implementation" and in the future "D3.3 Final distributed compute infrastructure specification and implementation".

To summarize, this process has been depicted as a sequence diagram that has been included in section 6.3.

n As a live system, aerOS allows to review those decisions in real-time. In addition, apart from the user, aerOS provides flexibility to the continuum itself (via its IEs) to exert **reallocation requests**. These reallocation mechanisms are triggered both manually by the end users to react to some notifications or performance information provided by the aerOS ecosystem, or automatically by the aerOS basic services (self-capabilities, orchestrator itself) to avoid potential service execution failures. Finally, a HLO receives a reallocation request, then makes a new allocation decision, and finally sends instructions to the corresponding/selected LLO(s) in order to perform the needed actions in the proper IEs: remove service workloads from the old IEs and run them in the new selected ones. Same as the service orchestration process, this process has been depicted as a sequence diagram that has been included in section 6.3.

Deployment approaches: Once the orchestration process has been described from an architectural point of view, it is interesting to move to a more practical approach to envision possible deployment. Three general and simplified deployment scenarios have been identified:

1. **Fully automatic deployment:** users do not select a specific domain as an entrypoint to deploy a service because they want to let the aerOS Meta-OS decide the best deployment location. This is the optimal (and most novel) functioning of aerOS, that offloads all the decision work to the smart, automated Meta-OS intelligence. In this case, the singleton Management Portal triggers the "Entrypoint balancer", which will forward the request to one available HLO (based in the configurable balancing rules). In this scenario the need for the Entrypoint balancer is emphasized, because without this component the same domain will always be used as the entrypoint for the orchestration request, risking unnecessary overloads. By following the aerOS approach, a new centralization point is avoided.
2. **Semi-automatic deployment:** domain administrators or users with a great knowledge of the resources that underline each domain may want to take part in the decision process to better control the final deployment location of the required services. At the end, human intervention may improve any decision taken by state-of-the-art AI models, and in addition, these human decisions can even improve those AI models. Therefore, in this deployment scenario, the user (through the proper form of the portal) can specify only a subset of requirements, or give overall, vague instructions about the domains/IEs for prioritization or selection purposes. The portal forwards the deployment request to the HLO of the chosen domain, or to one domain from a set of selected domains, so here the "Entrypoint balancer" can also fit. Then, the orchestration process is performed as per the first scenario.
3. **Manual deployment:** users select a specific domain, or even a specific IE or set of IEs from a domain to deploy the requested service. In this scenario, the *Intention Blueprint* is directly forwarded to the specific HLO, so the Entrypoint balancer can be bypassed.

5.4.3. aerOS distributed state repository

5.4.3.1. NGSI-LD mechanisms for aerOS

One of the keys behind the efficient, smart orchestration of aerOS is the capacity of accessing the state of the continuum in a decentralised way. This means that, regardless the spot in the ecosystem, any HLO can observe the available resources across the continuum to take the best allocation decisions. To achieve this, far from relying on a central repository, aerOS has envisioned a novel, ambitious paradigm for sharing the state of the continuum. Therefore, a well-defined and mature specification for modelling and exchanging contextual information must be selected, so NGSI-LD perfectly fits in the aerOS scope. NGSI-LD[3] is an information model and API for publishing, querying, and subscribing to context information, standardized by ETSI ISG CIM and based on JSON-LD.

In NGSI-LD, the world is seen as a set of entities. Precisely, an NGSI-LD entity has:

- **an entity identifier** that uniquely identifies the entity,
- **an entity type** that can be seen as a description of what attributes one can typically expect to be present, i.e., the data model, and
- **attributes** which are where the real data of an entity are stored. Attributes include system timestamps (creation, modification and deletion), values and sub-attributes, which are basically pieces of metadata that describe the attribute such as the unit.

Thus, almost anything can be modelled as an entity, for example, a room, a building, a temperature sensor, or a person. This is all up to the data model used. In the case of aerOS, a custom ontology has been developed (thoroughly described in section 5.4.3.3), in which **IEs** and **ServiceComponents** are the key entities to handle in the context of orchestration.

This contextual information is managed by Context Brokers (CB), which stores the most recent value of the attributes of NGSI-LD entities. These values can be obtained through a direct HTTP request to a REST API or via a publish/subscribe mechanism provided by the CB. Despite the existence of several implementations of NGSI-LD Context Brokers, the most logical choice for aerOS is FIWARE Orion-LD[4] for two main reasons: (i) it is developed in C, which is translated into a higher execution performance because this language is compiled rather than interpreted; and (ii) FIWARE Foundation is an active partner of the aerOS project, so the broker can straightforwardly be enhanced and fine-tuned to meet the needs of the project.

The context of each CB has a local scope as it is attached to the broker. Nevertheless, NGSI-LD provides mechanisms for distributing this local context information among different Context Brokers based on the creation of Context Source Registrations (CSR, from now on shorted to just “registrations”). A registration is a mechanism to inform an NGSI-LD CB about where to find more (non-local) entities. Upon queries, not only is an entity looked up in the local store of a broker, but the registrations are also consulted and for all matching registrations (e.g. a query to retrieve all entities of a certain type and the CB has a registration indicating that entities of that type can be found in another broker), a distributed query is sent to the broker behind the registration, and its information (its entities) is appended to the final response, so the process is completely transparent for the end user.

5.4.3.2. aerOS Federation enablers

In the above subsection, it has been stated that aerOS has envisioned a novel, ambitious paradigm for federated sharing of the state of the continuum to get rid of centralization. Concretely, the concept of a “distributed state repository” has been embraced. In the context of the aerOS architecture, a distributed state repository is a decentralised storage system responsible for maintaining the state information among different elements or components present in the continuum, fragmentarily corresponding to local domains.

[3] https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.08.01_60/gs_cim009v010801p.pdf
[4] https://github.com/FIWARE/context.Orion-LD

In practical terms, the distributed state of aerOS is handled by one instance of Orion-LD in each aerOS domain, building a Distributed State Network of Brokers (DSNB), with the contextual information (as NGSI-LD entities) federated among them through the automatic establishment of the necessary "inclusive" registrations by the aerOS Federator module included in the aerOS management framework (see section 5.5.8). To achieve it, the API of the CB must be reachable by the CBs of the other domains, so it must be publicly exposed through the API Gateway of each domain or via custom networking solutions defined in the aerOS network fabric.

NGSI-LD is quite versatile and offers several ways to configure the registrations for the distributed state, but in aerOS it has been decided that each broker in the DSNB only is responsible of managing the state of its own domain (pertinent fragmentation in local domains to achieve global federation), therefore avoiding replication of data. Thus, in aerOS all the NGSI-LD entities live in a single CB, but users or services in aerOS can query any of the brokers of the DSNB and obtain the same data, despite being actually stored in a single domain broker, provided that the registrations are set up correctly. For this layout to work, each CB needs to have (at least) one registration for each and every other broker in the DSNB (see Figure 11). However, NGSI-LD also provides mechanisms to replicate entities by creating specific "inclusive" registrations, which can be leveraged to replicate key entities such domains so that enhance the Meta-OS with resiliency and backup mechanisms.

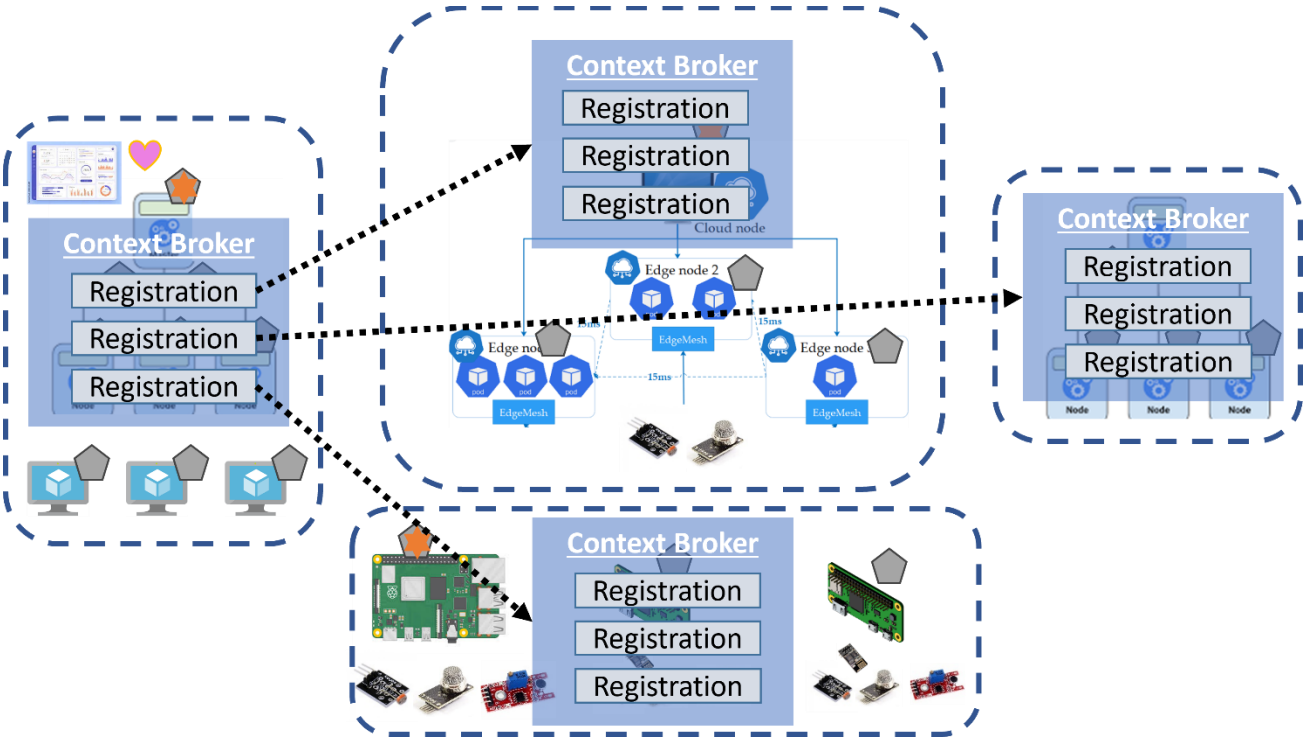


Figure 13. Example of a Distributed State Network of Brokers

5.4.3.3. aerOS Continuum Ontology

The inherent complexity of the IoT-Edge-Cloud continuum managed by the aerOS Meta-OS needs to be modelled into a data ontology as easily as possible, being understandable by humans and efficient for machine communications. In addition, there is a clear lack of existing ontologies for the computing continuum, and the minimal initiatives that have been found did not fit into the continuum conceived in aerOS. Therefore, an ontology for the IoT-Edge-Cloud continuum has been created from scratch for aerOS, inspired by some existing ontologies (e.g., FOAF[5]) and standardization initiatives such as OASIS TOSCA[6]. This ontology, as shown in

[5] http://xmlns.com/foaf/spec/

[6] https://groups.oasis-open.org/communities/tc-community-home2?CommunityKey=f9412cf3-297d-4642-8598-018dc7d3f409

Figure 12. aerOS continuum ontology is intended to encapsulate the essential concepts, relationships, and properties relevant to data management, processing, and orchestration within this distributed computing architecture, so that it has been designed having into consideration two essential pillars in the aerOS architecture: (i) Domain federation and continuum management; and (ii) Decentralized orchestration. Before moving to a more detailed description of this ontology, it must be stated that, as it was explained in "D4.2 Software for delivering intelligence at the edge intermediate release", (i) the Linked Open Terms (LOT) methodology has been followed to develop it; and (ii) this is a live ontology that will be enhanced as the project progresses, so some parts of the continuum may not be covered yet or may even be expanded.

When it comes to continuum management, entities that belong to the aerOS AAA (Authentication, Authorization and Access) block aim to represent the human side of the continuum, which is the relationship between them and the services and resources that conform the continuum through the definition of users, roles, and organizations. To create these classes, the *FOAF* ontology has been used, so aerOS users are assigned with a unique identifier, and also present additional information such as their given and last name following the *FOAF Person* concept. This aerOS user is member of an organization, which has a set of predefined roles that map to the permissions to perform certain actions in the aerOS Meta-OS. Therefore, each user is member of an organization and has assigned a role within this organization, which is the owner of the aerOS continuum in use.

The physical computing resources (converted into Infrastructure Elements after the installation of aerOS, see section 5.3) are the minimal computing unit of aerOS, so they must be represented to show the current state of the continuum by taking advantage of the defined monitoring processes in aerOS. Infrastructure Element emerges then as the central piece of the ontology. For instance, Self-awareness module (see 5.5.5) updates the attributes of IE entities to enable its performance monitoring.

Nevertheless, isolated IEs entities are not enough to depict the aerOS continuum, so conceptual layers must be added on top of the IE entity. The domain entity can be seen as the source of truth for the aerOS domain federation, because it groups a set of IEs and Low-Level Orchestrators (LLO), presents a single public URL, a *Boolean* attribute to indicate if the domain is the entrypoint of the continuum and a custom status as the IEs (preliminar, functional or removed). Moreover, each IE is linked to a single LLO that is mapped to a certain container management framework: Docker, Kubernetes, *containerd* and possibly others as this is expandable.

At this point, the ontology covers the aerOS network and compute fabric, but the other essential pilar is still pending: the decentralized service orchestration. According to the aerOS stack and Service Fabric (see section 5.4 and 5.5.3), aerOS follows a micro-services approach translated into containerized workloads as the minimal execution module. Thus, the entity named ServiceComponent aims at describing these containerized workloads, by including the necessary information to finally run them in the IEs (container image, environment variables, network connections…). In that regard, ServiceComponents indeed are the core entities of the aerOS orchestration as the whole orchestration process is performed independently for each ServiceComponent. This implies that the specification of requirements to make the allocation decision must be included in the ontology: IE requirements to let the HLO perform a pre-filtering of candidate IEs and SLAs to feed the allocation AI algorithm.

Like the relationship between Domain and IE entities, the Service entity has been thought as a conceptual layer in top of ServiceComponents to apply a logical grouping among them, which facilitates their management by the end users. Moreover, a lifecycle management model, managed by the HLO and underlying LLOs, has been designed for the orchestrated IoT services in the continuum, which is applied to the *status* attribute of the ServiceComponent entity instead of the Service because of its logical nature. This way, for instance, when services are removed from the continuum, their components (containers) are removed from the IEs, but their associated ServiceComponent entities remain stored in the distributed state repository with a *Finished* status. This approach allows to control the full lifecycle of service deployments with a transparent monitoring solution, or even redeploy them if necessary or requested by end users due to the persistence of that information.

Finally, as this continuum ontology is expected to be one of the main outcomes of aerOS, it will be properly and publicly published according to the standards of the LOT methodology, thus available in well-known public ontology repositories such as FIWARE Smart Data Models, which is fully aligned with the NGSI-LD standard.

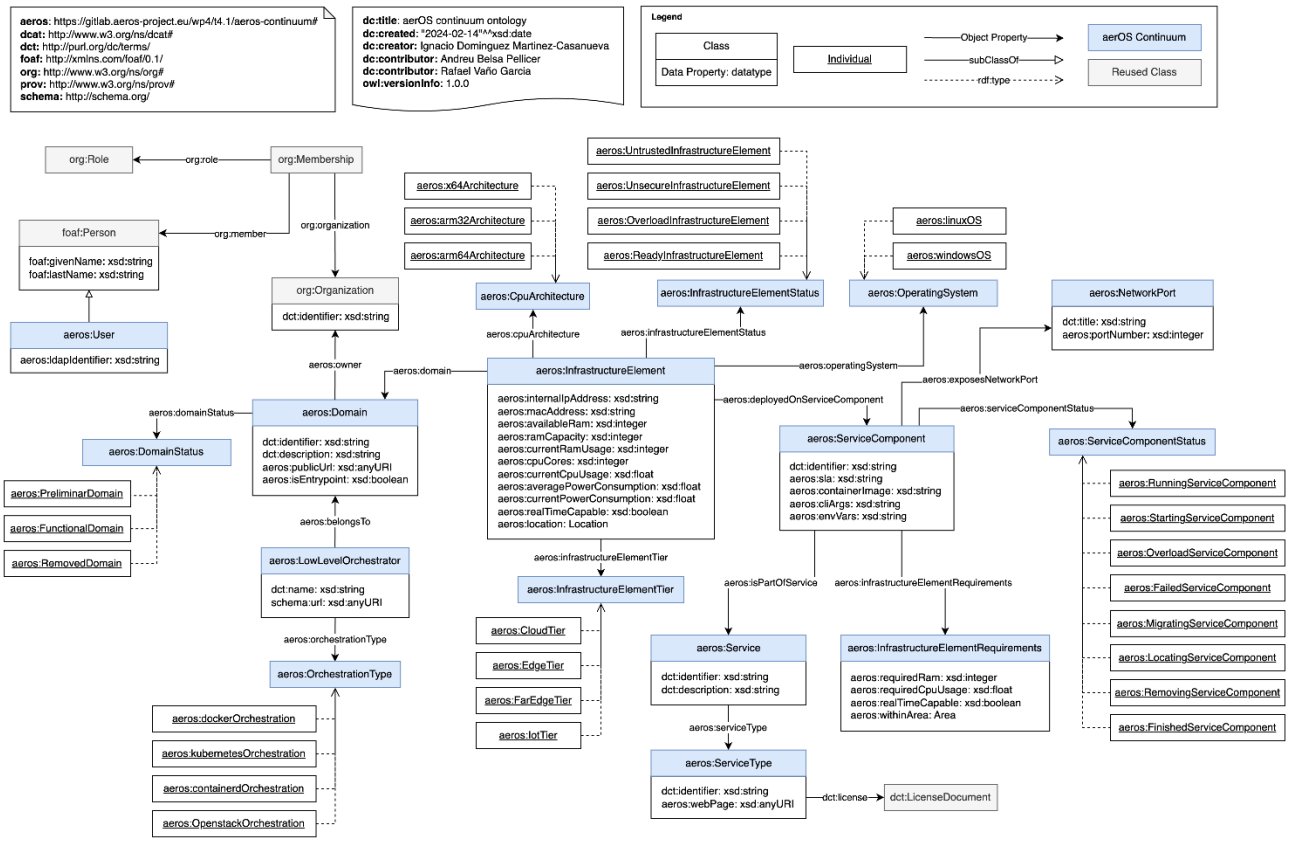


Figure 14. aerOS continuum ontology

5.5. aerOS basic services

5.5.1. Network and compute fabric

aerOS initial concern as a Meta-OS is to establish an infrastructure continuum over a set of diverse communication and computational resources. These resources are not only located in different administrative domains and geographical locations, but they also consist of different architectures, whether physical or virtual. The aerOS **Network and Compute fabric**, a main component of its architectural design, establishes the underlying framework that abstracts this variety of heterogeneous computing resources and exposes them as a homogenized IoT-Edge-Cloud continuum environment. To this end, aerOS has developed a suite of dedicated services, among its core offerings, which create a unified layer for managing connectivity, communication, and computational resources, ensuring efficient integration and secure network interactions.

This design integrates and exposes heterogeneous resources' capabilities under a unified management and orchestration framework, providing a common interface for allocation, runtime access and monitoring. It has the dual responsibility of both providing a common access interface, which abstracts heterogeneities, and provisioning for their network connectivity. This layer provides the basis for a seamless operation of applications and services, as explained next in the service fabric section.

Network and Compute fabric harnesses the aerOS domain as its fundamental building block. Each domain, with the support of aerOS network and compute fabric services, acts as an independent administrative unit with the capacity to oversee its own compute, storage, and network resources. The unified control plane, provided by aerOS Network and Compute fabric, plays a pivotal role in streamlining the management and orchestration of these resources across the continuum. It offers advanced functionalities such as dynamic resource allocation, intelligent load balancing, and automated scaling, ensuring optimal resource utilization and performance. Through seamless integration with the federated domains, facilitated by standardized protocols and interfaces,

the unified control plane enables cohesive governance and coordination of resources. This integration exposes a consolidated pool of resources, poised to serve as the foundational underpinning for hosting IoT services and applications, offering enhanced scalability, reliability, and security for the entire ecosystem.

The network fabric serves as the communication backbone, facilitating fast and dependable packet transmission between integrated components across the aerOS ecosystem. Meanwhile, the compute fabric encompasses a diverse array of computing resources, referred to as Infrastructure Elements (IEs) within aerOS. These resources are responsible for delivering the requisite processing power to execute the requests of IoT developers, utilizing containerized workloads for efficient deployment. By leveraging this compute fabric, aerOS ensures that a pool of resources is readily available to fulfill user queries, seamlessly providing the most optimal candidates based on the specific requirements of each use case. This abstraction shields users from the complexities associated with discovering and managing heterogeneous architectures, offering a streamlined and user-friendly experience within the aerOS environment.

While the initial architecture release set the design principles for the compute and network fabric, the implementation of the MVP provided valuable input that enabled a more elaborate and detailed definition which is reported in this section. As initially planned the key features of aerOS network and compute fabric are:

- **Scalability**: the ability to expand seamlessly, without sacrificing performance, as additional infrastructure is introduced to the aerOS ecosystem.
- **Reliability**: ensure robust and continuous operation, even when there are component failures, by supporting redundancy and failover mechanisms.
- **Flexibility**: The fabric should support various types of workloads and applications, allowing for dynamic allocation and reallocation of resources as needed.
- **High Performance**: tacitly provide for the fulfillment of each use case the most efficient resources while ensuring the most efficient communication among dispersed computing nodes.
- **Transparency**: abstract access variety and offer homogenized interface for resource management and orchestration

The Network and Compute fabric within aerOS mutually benefit from their interaction with the Service fabric. While the Network and Compute fabric serves as the execution substrate for services, the Service fabric facilitates the provisioning and management of compute resources through APIs. This symbiotic relationship enables dynamic scaling and orchestration in response to fluctuating workload demands. By abstracting access to computing, networking, and storage capabilities, aerOS runtime and services provide a unified interface for users. Additionally, APIs expose and federate these resources coherently across all aerOS domains, ensuring seamless integration from edge to cloud. This cohesive approach optimizes resource utilization and enhances overall system efficiency within the aerOS ecosystem.

As aerOS evolves instrumenting container orchestration, it leverages on containerized environment interfaces as its foundational framework. Essential functionalities such as persistent storage for data availability and redundancy, networking between Infrastructure Elements (IEs) with advanced features like overlay networks and network segmentation, and management of network resources are all standardized through interfaces like CNI or CSI. To expand its capabilities, aerOS incorporates extensions to these interfaces using operator technologies.

Within each aerOS domain and on top of each IE, certain services are responsible for managing the domain's compute, storage, and network resources. Concurrently, other services facilitate the federation of these domains' resources. This federation enables the sharing of information regarding integrated capabilities and availabilities, offering an on-demand response to existing resource availabilities. Moreover, it establishes a standardized access channel for providing resources to host services within each domain, regardless of its location across the continuum.

aerOS network fabric encompasses the full range of capabilities aerOS provides for establishing connectivity among IEs within each aerOS domain and across remote domains, located anywhere from edge to cloud. Within an aerOS domain, this connectivity is primarily based on the Container Network Interface (CNI) and the programmability potential it offers. Virtual networks are deployed as overlays on top of existing connectivity and offer the possibility to programmatically define their behaviour. The integration with CNI allows for

dynamic network configuration, enabling customizable networking solutions to meet specific use cases, and ensuring flexible and scalable network management. This high level of programmability supports advanced networking features such as custom routing, traffic policies, and network isolation, thereby enhancing the overall efficiency and performance of containerized applications. Often -whenever possible-, network fabric of aerOS makes use of advanced utilities available in the used frameworks, e.g.,within the Kubernetes clusters. While all components, their interactions and actual technologies are detailed in WP3 deliverables ("D3.1 Initial distributed compute infrastructure specification and implementation" and "D3.2 Intermediate distributed compute infrastructure implementation") the architectural design principles are listed in this section.

When it comes to connectivity across remote domains, the aerOS network fabric relies on additional network functions to establish secure paths over public networks. These are implemented as cloud native functions (CNFs) which are responsible for tasks such as routing, switching, firewall, load balancing and tunnelling, all built on top of containerized environment. Beyond secure paths, there is also provision for dedicated and completely isolated paths which can ensure both privacy and low latency. This comes as part of the network fabric involving cloud native implementation of VPNs.

Security within this service and network fabric is paramount and TLS is employed to secure domain access ensuring data integrity and privacy and Role-Based Access Control (RBAC) is utilized to enforce fine-grained access control policies, allowing only authorized entities to access or manipulate resources. Furthermore, Network Policies are implemented to control traffic flow among both IEs and services enhancing security within each domain.

As aerOS evolves in the cloud native domain, regarding the networking area this is reflected in the shift from traditional Virtual Network Functions (VNFs) to cloud-native implementations. While traditional VNFs which were typically deployed as monolithic virtual machines, often faced challenges in terms of scalability, flexibility, and resource efficiency, aerOS builds networking capabilities on cloud-native network functions, designed to run in containerized environments and taking advantage of the microservices architecture. Cloud native network services are deployed and used in service chaining or mesh topologies. As an example, we can reference that accessing an aerOS domain involves a channel built with a chain of programmable network functions like ingress proxy, TLS certification enforcing, load balancer, API gateway. All these are implemented as microservices, and their behaviour can always be adapted based on exposed programmability. In cases when services require direct access to another service located in a remote domain, a network service mesh practice injects all required network functions. Generally, that paradigm shift towards cloud native networking practices not only optimizes resource utilization but also integrates seamlessly with the programmable infrastructure, providing a robust and adaptable network fabric.

Concluding, aerOS network and compute fabric design transforms the network path from the edge to the cloud from just a connectivity medium to a unified computing platform with integrated connectivity capabilities, able to support the implementation and deployment of end-to-end services which can drive innovation and support diverse, heterogeneous verticals by providing a lower entry barrier for distributed applications, since edge and far-edge devices will be part of this unified platform. The result is to uncomplicate the connectivity in such environments, that often require for the IoT developer having a previous deep knowledge on the network and connection details of every part of the architecture.

5.5.2. Data Fabric

The creation of an IoT-Edge-Cloud continuum brings a highly distributed and dynamic data landscape. With the goal of providing a holistic view of all the data available in the IoT-Edge-Cloud continuum, whilst enabling data governance mechanisms that help ensure a responsible use of data, aerOS aligns with the two recent data management approaches that focus on decentralizing the management of data, namely, data mesh and data fabric [9].

To cope with the complex data landscape of the continuum, aerOS shifts the management of data close to their sources, i.e., to the data providing domains. In this sense, aerOS embraces the **data as a product** thinking and the **domain-oriented data ownership** principle, as proposed by the data mesh paradigm. Owners of data providing domains are responsible for turning their raw datasets into high-quality data products that data consuming domains can easily discover, understand, trust, and access. Nevertheless, building data products requires data engineering skills, as well as following standard data models and interfaces, which enable

interoperability with data consuming domains. To help data owners in the creation of data products, aerOS proposes a **self-service data infrastructure** that follows the architecture defined by the **Data Fabric paradigm**.

The Data Dabric paradigm introduces a metadata-driven architecture that automates the integration of data from heterogenous sources and enables uniform access to the data through a standard interface. Aligning the Data Dabric architecture with the data mesh principles of data as a product and domain-oriented data ownership, results into the following novelties: i) the Data Dabric, as the self-service data infrastructure, transforms the raw dataset of the data providing domain into a data product; and ii) the resulting data product can be accessed and shared with data consuming domains through the standard interface of the Data Fabric.

Knowledge graphs, as previously described in [9], represent a promising technology for the realisation of the Data Dabric architecture. Knowledge graphs enable contextualized understanding of data by explicitly representing in a graph structure the facts (i.e., the data) connected with the knowledge we have about them (i.e., the concepts). These explicit representations of knowledge, as concepts that relate to other concepts, in a formal way that can be understood by machines are known as ontologies, and these play a crucial role in the creation of the semantic layer.

Building on the principles of the knowledge graph, aerOS proposes a definition of the data product as the transformation of a raw dataset into a semantically annotated data integrated in the knowledge graph, as depicted in Figure 15. This mapping between the conceptual level (represented by ontologies) and the physical level (represented by raw datasets such as data streams) is known as semantic lifting. In this semantic lifting process, the concepts extracted from the physical datasets are captured in the knowledge graph and linked with concepts from other physical datasets. As a result, each data product represents a subgraph of the whole knowledge graph created by the aerOS Data Fabric.

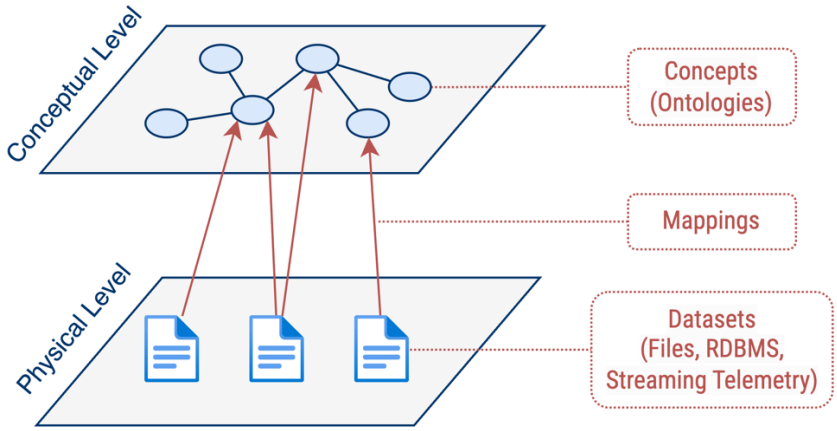


Figure 15: Semantic lifting based on mappings between the conceptual and physical levels.

To implement the knowledge graph, aerOS has adopted the ETSI NGSI-LD standard [10]. NGSI-LD (see 5.4.3.1) defines an information model derived from the property graph model, which additionally can reference ontologies. Thus, the NGSI-LD standard enables building property graphs where data are semantically annotated. In addition, the Representational state transfer (REST) API defined by NGSI-LD facilitates the management and interactions with the graph. The compact and natural information model of NGSI-LD, along with a friendly REST API, makes NGSI-LD a promising graph standard for implementing knowledge graphs, compared to other traditional graph standards such as the Resource Description Framework (RDF), which is deemed more verbose and entails a steep learning curve. In terms of the NGSI-LD standard, the NGSI-LD Context Broker is pinpointed as the component that stores the knowledge graph. The NGSI-LD Context Broker, by means of the NGSI-LD API, allows data consumers to interact, query, and even subscribe to notifications in the stored knowledge graph.

The aerOS Data Fabric, as the self-service data infrastructure as per the data mesh paradigm, provides a generic framework called Data Product Pipeline for data owners to build and onboard their own data products into the knowledge graph. The aerOS Data Fabric, by means of the Data Product Manager, exposes an interface towards

data owners to onboard new data products and orchestrate the pipeline that turns raw datasets into data products. These components are reflected in the high-level architecture of the aerOS Data Fabric shown in Figure 16.

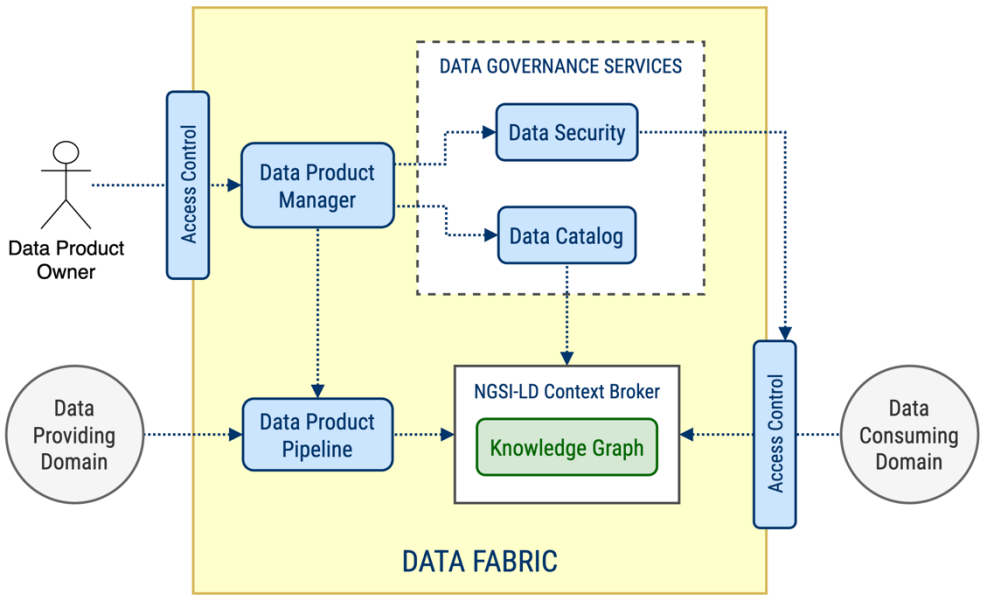


Figure 16. High-level architecture of the aerOS Data Fabric.

But, as illustrated in the high-level architecture, in addition to creating and sharing data products, the aerOS Data Fabric also includes data governance mechanisms. aerOS must govern a distributed mesh of data products scattered throughout the continuum; thus, mechanisms for cataloguing and controlling the consumption of data products must be put in place.

In this regard, the aerOS Data Fabric also relies on the knowledge graph to integrate the metadata that supports the data governance services. For example, when it comes to cataloguing data, the knowledge graph would contain metadata that describe the owner of a given data product, the domain that the data product belongs to, and even the concepts related to the data product.

Similarly, for securing access to data, sensitive classification of data or access control policies based on the domain that provides a data product, could also be captured as metadata in the knowledge graph.

5.5.3. Service fabric

While the aerOS Network and Compute fabric provides a seamless and unified layer that removes concerns regarding the underlying hardware differences and the geographical or administrative distribution of resources, thereby taking care of the underlying "execution environment," the aerOS Service fabric complements this by establishing a common "service runtime environment." This environment offers a robust framework for managing and orchestrating IoT applications as microservices in a standardized and unified manner.

The aerOS Service fabric is built as a set of dedicated microservices capable of managing IoT service deployment, orchestration, and lifecycle management (LCM). The primary concern of this fabric is to relieve IoT developers of runtime concerns and decisions regarding their applications' orchestration and to automate lifecycle management and integration with existing consuming and producing counterparts. It comprises a set of aerOS basic services deployed within each domain, with some operating directly on top of each IE and others having a single running instance within the domain. Collectively, these services orchestrate microservices, as containerized-based applications, on top of the available IEs. Additionally, to achieve the most efficient results, the aerOS Service fabric integrates advanced AI techniques to optimize resource usage, enhance service performance, and proactively mitigate inferred failures.

The aerOS Service fabric consists of a set of cloud-native services with clearly defined boundaries and responsibilities, enabling specialized functionalities and independent deployment. The designed service fabric includes robust lifecycle management tools and APIs, facilitating dynamic scaling based on workload demands to ensure optimal workload placement and resource utilization. While the orchestration of IoT applications is

the primary concern of the Service fabric, it also encompasses security, monitoring, observability, scaling, and resiliency as integrated functionalities.

Built on top of container management framework provisions, the entire aerOS framework is constructed as containerized microservices. The aerOS Service fabric extends capabilities and orchestrates IoT services across various environments that support containerized workload execution, ultimately managing distributed IoT applications and services seamlessly across diverse aerOS domains.

With the goal to enforce an orchestration process beyond each domain's administrative boundaries, which should promote workloads' most efficient placement based on the overview of the status of all IEs and domains, aerOS service fabric has adopted a federated orchestration process design which is realised based on a set of basic services deployed within each domain. This is provided by the combined actions of orchestration services and the federator component. These services provide to the aerOS service fabric the capability to address constraint-based optimisation, placement related problems, considering both application requirements and resources availability, and achieve thus the best placement of IoT vertical applications all the way from the edge where IoT devices are integrated to the cloud where more powerful computing resources are available. Service fabric on one hand receives users' deployments requests, as submitted in the form of templated submissions compiled in the Entrypoint dashboard, the **Intention Blueprint**, and on the other hand has an overall knowledge of the status and capabilities of all IEs and domains across the continuum based on the services provided by the federator, as implemented within aerOS Data Fabric.

The Intention Blueprint is received in the exposed HLO REST API exposed by each aerOS domain. From there on, aerOS service fabric initiates the process of service placement including resources orchestration. As a first step the received Intention Blueprint, a custom TOSCA file, is parsed and all service components, requirements and constraints are registered in the aerOS continuum as part of the aerOS knowledge graph. The second step includes the selection of a set of candidate IEs, these that match service requirements, by querying the continuum via data fabric provided capabilities. Subsequently the set of candidates are forwarded to the Allocator which integrates predictive and optimisation algorithms to make the most efficient selection for the actual IE which should host the request. At this point it is important to say that all communication among these components is standardized both in terms of communication channels using AsyncAPI specifications and in terms of payloads expected in each stage. Just for informative reasons we refer that Protocol Buffers (*protobuf*) payloads are used to exchange information mapping to aerOS continuum entities (IEs, aerOS domains, service components, ownerships, etc). The flexibility provided by this design enables each domain administrator to integrate their own decision engine by replacing the Allocator with a custom one as long as it respects the communication specifications (all fully documented). The Implementation Blueprint which is the output of the Allocator is forwarded to the deployment engine component. This HLO component is responsible to understand if this deployment refers to a local IE or a remote one, located to another aerOS domain. Based on this decision the final step is to call another HLO exposed API, the HLO Allocation Endpoint, of the domain which hosts the selected IE. This is the exit point towards the actual enforcement layer and all information regarding both deployed service component and selected IE are available. An **Implementation Blueprint** is sent to the LLO which relies on cloud-native technique of custom operators for abstracting the access to integrated resources.

Thus, aerOS Service fabric, as a core functionality, integrates a doubled-layered orchestration engine. A concept which enforces separation of concerns regarding a) the decision as HLO acts as an AI-powered Decision-Making Engine as it interfaces with AI/ML services, and b) the enforcement as LLOs have the actual knowledge of how to proceed to workloads placement, on the underlying IEs. This schema provides the required flexibility and extendibility to integrate more resources over which aerOS can orchestrate services. It only requires the development of an additional LLO (K8s operator).

Beyond making transparent to the user the whole process of services placement aerOS service fabric provides a framework fully capable of overseeing the runtime status of deployed services and responding proactively or on event triggers as required to enforce decisions that ensure security, scaling and resiliency of hosted services. Most of the services orchestration functionality are detailed in next sections. Though it is crucial to mention that once services are deployed, they are not left "helpless", aerOS Service fabric provides the services to either sense abnormalities and request service migration or even recognize "suspicious" behaviours and request services isolation. A full suite ready to raise functions on response to events is hosted and serverless capabilities support programable responses to registered events.

As privacy and security is of paramount importance, corresponding services are closely integrated within Service fabric, validating usage and access to resources and data. While all the cybersecurity services which minimize concerns regarding a secure and safe environment are presented in section 5.5.4, it would be worthy to mention that having designed for an end-to-end security integration aerOS Service Fabric strongly connects with DevPrivSecOps pipelines. The development and integration of a component automates aerOS service updates after being validated in GitLab pipelines and security preserving tool chains.

As a final note, aerOS has standardized all components and functionalities of its Service Fabric. The exposed APIs and payloads are fully documented based on these standards using OpenAPI. Internal communication among Service Fabric components is also standardized. Both payloads and communication channels, whether REST APIs or broker topics, adhere to OpenAPI or AsyncAPI standards. This thorough standardization and the accompanying documentation provide stakeholders who adopt aerOS over their infrastructure with the flexibility to extend or even replace components, with the only requirement being to maintain interface contracts.

In conclusion, the Service Fabric, operating on top of the Compute and Network Fabric, plays a critical role in enhancing and efficiently utilizing the underlying resources. It manages the deployment, scaling, and orchestration of services across the continuum, ensuring that application components are optimally placed to meet performance requirements, Service Level Agreements (SLAs), and energy efficiency criteria. This synergy between the Compute and Network Fabric and the Service Fabric ensures a cohesive and efficient infrastructure that supports the dynamic needs of IoT applications.

5.5.4. aerOS cyber security components

The aerOS cybersecurity architecture is a multi-layered and integrated security framework designed to ensure a robust and trustworthy environment. It represents a sophisticated cybersecurity framework designed to protect its infrastructure and data. By using a separation of concerns approach, aerOS offers a modular cybersecurity architecture that combines elements to validate access to distributed resources, register appropriate security levels, and detect threats at runtime. This integrated approach ensures that each component not only supports the overall security objectives but also enhances the system's ability to manage and mitigate potential threats effectively.

A key component of the aerOS cyber security system is the aerOS Identity Management (IdM), whose ability is to register and evaluate policies for resource and data access. It utilizes Keycloak[7] IdM, which provides comprehensive functions to strengthen its cybersecurity by managing the authentication and authorization of aerOS clients. Focusing on authentication, this aerOS IdM design employs advanced mechanisms to defend aerOS from unauthorized access. The aerOS Management Portal, serving as the primary interface for entities interacting with the framework, incorporates a module linked to Keycloak for authentication and authorization. This module uses OpenID Connect8, a protocol based on the OAuth 2.0 framework, to offer secure token-based authentication and Single Sign-On (SSO) capabilities. This enables entities to access the aerOS portal using credentials obtained from their respective organizations. Elements deployed within the aerOS continuum requiring access to protected endpoints must first obtain an ID token from aerOS IdM, which is then used to make API requests. aerOS leverages Role-Based Access Control (RBAC) to grant users access based on their assigned roles. The integration of aerOS IdM with OpenLDAP has been implemented to enhance the adoption of aerOS IAM by stakeholders, facilitating the automatic federation of user information from the LDAP directory. This eliminates the need for manual transfer of user data to aerOS IdM, streamlining user management and group associations. By leveraging both the Identity Management system for authentication and authorization and OpenLDAP for user management, a robust and flexible access management infrastructure has been developed. The integration of LDAP with the aerOS IdM element further improves aerOS Role-Based Access Control (RBAC) policies, providing additional layers of access control based on user attributes and enabling more precise and dynamic security policies. aerOS implements a system of precise control and management over resources, which is seen in the establishment of different roles. Each role is associated with specific access rights within the aerOS services environment and linked to a corresponding group in

[7] https://www.keycloak.org/
[8] https://openid.net/developers/how-connect-works/

OpenLDAP. These groups are efficiently synchronized with aerOS IdM through LDAP federation, ensuring consistent and secure access management across the platform.

A Secure API Gateway is located within each domain by the modular design of the aerOS cybersecurity framework, and it serves as a checkpoint for validating and enforcing policies registered in the IdM components. The implementation of this API Gateway is based on KrakenD, which acts as a central hub, enforcing security measures and access controls on all APIs. The integration of the aerOS API Gateway component into the aerOS architecture plays an important role in enhancing the overall security framework of the system. Initially, aerOS is described as a Meta-OS comprised of multiple APIs, which inherently lack built-in security mechanisms to counter threats such as unauthorized access. Recognizing this vulnerability, the incorporation of the aerOS secure API Gateway is strategic, addressing the critical need for robust security within the aerOS. This aerOS APIs guard enhances security by utilizing the user roles and groups established in the Identity Management (IdM) system. This integration between the aerOS API Gateway component and IdM streamlines access control by granting users permission to specific APIs and functionalities based on their assigned roles. Beyond access control, aerOS API Gateway fosters a unified entry point for all aerOS components, allowing them to interact seamlessly with various APIs. This eliminates the need to manage multiple access points, simplifying communication and data flow within the system. Furthermore, the API Gateway offers granular control over incoming and outgoing API traffic with features such as message caching and packet modification in the aerOS API Gateway itself. This empowers administrators to tailor data based on specific needs, perform additional internal checks, and enrich functionalities through scripting support. In essence, the aerOS API Gateway component complements the existing security features within the project's architecture, particularly the IdM system. It strengthens the overall security posture by adding a critical layer of defence specifically focused on API security, which is vital for protecting the integrity of the Data Fabric.

aerOS cyber security framework ensures security at all levels, including the integration of resources at the lowest level. This is designed to integrate security capabilities into each individual Infrastructure Element (IE). Self-security in aerOS is robustly supported by an integrated tool, Suricata[9], an advanced open-source tool for network analysis and threat detection at the node level. This component, comprising a Log Monitoring module and an ETL (Extraction, Transformation, Load) processing module, plays a crucial role in real-time threat detection and response within the aerOS infrastructure. Suricata continuously monitors network traffic generated by the network cards of IEs in real-time. This enables the self-security module to identify malicious activity and potential vulnerabilities or attacks. Upon detecting any malicious activity or vulnerabilities, the information is immediately relayed to the self-diagnose component. This ensures timely interventions, bolstering the security of the entire aerOS ecosystem. By offering real-time network traffic analysis and threat detection, it strengthens the overall security posture of IEs and domains within the aerOS ecosystem. The inclusion of this self-security module complements the existing security layers within the aerOS architecture. It bolsters security by providing real-time threat detection, improving vulnerability management on IEs, and enhancing intrusion prevention through early warnings.

Beyond resources control and policies enforcement, aerOS takes proactive measures to improve the security foundation and foster trusted communications and interactions among connected IEs. Therefore, an ongoing process of estimating and validating the trustworthiness of integrated pieces is carried out. The Trust Management component continuously assesses the trustworthiness of individual Infrastructure Elements (IEs) and entire aerOS domains. This information is crucial for informed decision-making within the aerOS ecosystem. The Trust Management component works by collecting various attributes from each Infrastructure element (IE). These attributes include security events, health scores, service activity, communication patterns, update status, reputation, and even system information like CPU and RAM usage. Each attribute is assigned a weight based on its significance to security. Security events, for example, are considered more critical than the number of services running on an IE. By analysing these weighted attributes, the Trust Management component calculates a trust score for each IE. This score reflects the overall reliability and security of the IE within the aerOS environment. In terms of alignment with the overall aerOS security architecture, the Trust Management component enhances the security strategy by providing dynamic, real-time assessments of trust. This is crucial in a complex environment like aerOS, where numerous IEs operate and interact. By enabling a clear, continuously updated view of each IE's trust level, aerOS can ensure that all operational decisions, such as

[9] https://suricata.io/

service allocation and system management by the High-Level Orchestrator (HLO), are made based on robust and up-to-date trust assessments. This alignment with the broader security strategy ensures that aerOS can maintain high levels of security and operational efficiency across its continuum.

Overall, the aerOS security architecture exemplifies a comprehensive and effective approach to cybersecurity, characterized by its multi-layered and integrated security components. From identity and access management with aerOS IdM to real-time threat detection with Suricata, and from the cohesive integration of the aerOS API gateway to the dynamic assessments by the Trust Management component, each element of the architecture is purposefully designed to address specific security needs while contributing to the overall security posture of the system.

5.5.5. aerOS self-* and monitoring

Explanation of the service:

This basic service is materialised in a suite of automated self-* features (microservices) that an IE is continuously applying to itself. In a widely varied environment where a large number of IEs co-live, each of those should have a set of capabilities to modify their behavior / status in such a continuum. This set of self-* capabilities is divided in two, those that are strictly necessary (core) and those whose installation is optional depending on the circumstances (non-core). Those capabilities, in combination with the global orchestration and data management, allows the IEs to still be considered empowered entities, playing a crucial role in a large environment, thus reinforcing the decentralisation principle of aerOS.

Functionally, this service supports the monitoring of inner parameters (some that are exposed to the whole continuum and some that are not) through the self-awareness and self-realtimeness modules, the logic of understanding whenever an action must be brought upwards (for instance, rejecting service/workload assignments or triggering specific orchestration requests) or if a specifically labelled application is complying with the agreed Service Level Agreement (SLA at node level) using the self-orchestrator, self-diagnose and self-optimisation and adaptation modules.

This service also includes the management of the IoT devices that can be attached to the IEs, offloading the need of central management of configuration, control, or healing, by using the self-configuration and self-healing modules. It also involves certain cybersecurity traits, to relax the demands of centralised security/privacy control, through the self-security module. Besides, it implements active attempting to recovery after abnormal activities, dependability, and long-term, up-to-date synchronisation with its custom configuration and horizontally scaling of resources, mainly by applying the self-scaling (and other) modules. All the functionalities of this basic service share the automation degree and the dynamicity and capacity of parameterisation by the user of the continuum. This is partly possible using the self-API module.

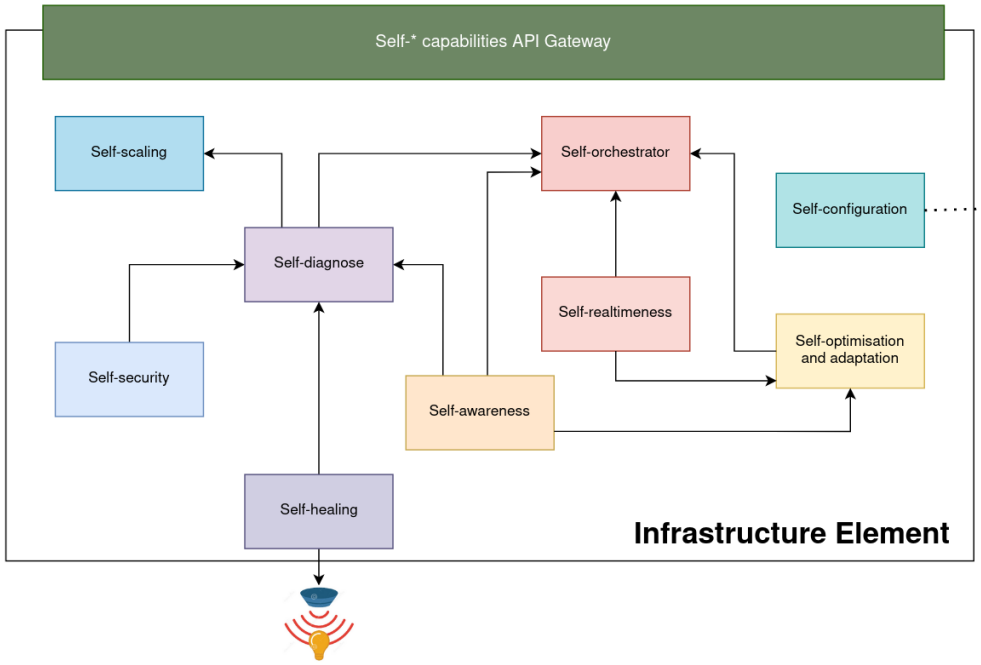


Figure 17. Self- capabilities relationships*

Necessity of the service in aerOS:

In a continuum, many situations might happen that would require advanced logic in the most granular places to take place. Actions like down network, sudden disconnection from the continuum, a peak of demand in the running services, increased energy consumption, change of energy feeding type, modification of configuration (e.g., bit framerate) will likely occur at some spots of the continuum along the time. Relying on a central entity to continuously gather all those data from all the IEs in a continuum would mean an unbearable network overload and would clearly jeopardise the autonomy and the decentralisation capacity of the whole ecosystem. Therefore, there is the need of creating mechanisms within every IE that will constantly monitor and act upon those events, proactively inspecting and identifying their occurrence, and introducing a certain degree of intelligence at edge/device level.

Materialisation in the architecture:

These basic services run as a suite of microservices, each of them tackling a specific functionality, that is installed in the IEs (depending on their flavour or role in the continuum). These microservices are lightweight, as they are expected to run in heterogeneous IEs, that might live at resource-constrained equipment (e.g., close to the edge or far-edge of the continuum). More details can be found in the deliverable "D3.2 Intermediate distributed compute infrastructure implementation" and in the future "D3.3 Final distributed compute infrastructure specification and implementation".

5.5.6. aerOS decentralised AI

In aerOS, AI is considered from an internal and external perspective. External AI fulfils the requirements coming from user applications, e.g., those developed within pilot applications. Examples of external AI can be frugal federated learning on local data or deployment of a prediction model (e.g., air quality prediction model). In particular, external AI is a specific task that may have a corresponding workflow and can be commissioned to be executed on the aerOS infrastructure. This functionality is supported by auxiliary AI services described in Section 5.6.1.

Internal AI supports aerOS internal continuum management by providing intelligent decision-making that spans across different base services and respective components. Examples of components using internal AI are: HLO for service allocation and self-* components for self-scaling and self-adaptation/optimization. As part of these services, AI-related functionalities are deployed in different locations in the continuum. AI-related

functionalities can be “built-in” in specific services being part of an internal component (e.g., in most cases already trained model is used for predictions), or they can be used as separately deployed services (e.g., when required, use communication API to request prediction from a model available elsewhere and “wrapped” as a service). In the former case, these are the design considerations for specific base services being developed in aerOS which can also include specific needs related to frugality or explainability depending on the nature of the service and foreseen potential placement in the continuum. In the latter case and in the case of external AI, aerOS aux AI services can be used to provide execution environment for AI-related functionalities (e.g., deploying trained model for inferencing, training a model in a distributed way) and support for additional features such as explainability/interpretability or frugality of the solution. Aux AI services are not obligatory in the aerOS deployment but can be included depending on the characteristics of the use case.

5.5.7. aerOS common API

The primary objective of a common aerOS API is to establish a communication environment that fosters seamless interaction among services spanning the entire compute continuum, encompassing the edge to the cloud. These services include both the system's internal set of basic and auxiliary services, and IoT services deployed by vertical stakeholders. While this section specifically focuses on the APIs employed internally within the system to enable and ensure communication within and across system domains, a common approach unifies how vertical IoT services communicate and exchange data. The rationale behind the design of these APIs is to abstract the complexity of interacting with individual services, which often implement diverse APIs and produce incompatible data sets. Services exposure is based on aerOS domain level aggregation and abstraction. This means that each aerOS domain exposes a common set of endpoints, communicating using the same data structures and request-response patterns, regardless of the underlying implementation within each domain and its IEs, and all aerOS domains encompass the same technique to provide a single point of control and access to the exposed API.

The above discussion highlights the two main concerns that were addressed when designing aerOS common API. First, how all underlying services should be abstracted and aggregated to a minimum but efficient API which can take advantage of all underlying services. Second, how to provide a single point of access which could additionally enforce security policies, handle rate limiting, and of course efficiently route requests to the appropriate domain services. The answer to the first question was guided by the overall architectural decisions and the answer to the second emerged through a thorough state of the art analysis regarding APIs centralized access control architectures.

To illustrate the previous, first, the reasoning behind the APIs to be exposed is discussed and then the common exposure implementation decision is presented.

As previously mentioned, within each aerOS domain, three primary building blocks act as functional components, enabling the implementation of the Compute and Network Fabric, the Service Fabric, and the Data Fabric at the domain level. These components collectively support the seamless integration of the domain into the broader aerOS continuum. This integration encompasses robust authentication and authorisation, federation of aerOS domains and IEs status across the continuum, and orchestration requests submission which enables coordination and management of the various components and services within the continuum. Thus, the following three service groups are primarily identified for API domain exposure.

- Authentication and Authorization Service which is responsible for handling user authentication and authorisation to access domain services. It ensures that only authenticated and authorized users can interact with the system's functionalities. The common API abstracts the underlying implementation details of different authentication mechanisms, allowing developers to interact with the service using a uniform and consistent API contract. Users and applications can securely obtain access tokens or authentication tokens to authenticate themselves and gain access to the relevant services.
- Data Fabric and Mesh Services, based on NGSI-LD, implementing federation across aerOS continuum. The data fabric and mesh services are implemented using the NGSI-LD standard. The common API abstracts the complexities of the underlying implementations, providing a unified way for clients to query, update, and manage data using the NGSI-LD data model. Services and applications can seamlessly interact with the data fabric and mesh services, irrespective of their specific NGSI-LD implementations, ensuring consistent data exchange. aerOS takes advantage of integrated data mesh

and fabric technologies and provides a "Data Access Layer" which acts as an intermediary between the services that produce data and the applications or clients that consume these data. This is of paramount importance both for aerOS domains exchanging status information and enabling thus a coherent view of all continuum state, and for vertical IoT services requesting data produced and exposed in other domains.

- Orchestration Services to facilitate the dynamic allocation and scaling of resources, ensuring optimal performance and resource utilization. The common API acts as an intermediary for clients to interact with the orchestration services, abstracting the intricacies of underlying orchestrators. This can be implemented with message queues and event streaming to facilitate asynchronous communication and data exchange between domains. Messages or events can be published and consumed across domains, enabling decoupled communication and cross-domain deployment requests.

To realise the above concept, the common API can be exposed through a centralised API Gateway, deployed within each domain. The API Gateway will act as a single-entry point for other domains, clients, and applications to access the underlying services.

Beyond forwarding requests to the appropriate exposed components, API Gateway seamlessly integrates authentication and authorization requests, verifying user credentials and granting access tokens for secure service access. It routes requests to the appropriate data fabric and mesh services, orchestrating data retrieval and up-dates as per the NGSI-LD standard. Additionally, the API Gateway can interact with message broker to coordinate orchestration tasks and distribute workloads efficiently. By adopting this common API approach with a centralized API Gateway, the ecosystem achieves a cohesive and standardized communication environment. The abstraction of underlying implementations simplifies development efforts, enhances system scalability, and enables seamless integration of new services into the ecosystem. This results in a unified and user-friendly experience for all stakeholders, fostering an agile and dynamic IT system architecture.

Conclusively, by leveraging an API gateway, aerOS domains provide a robust, centralized, and standardized interface for interacting with the exposed services. The gateway's techniques, such as centralized access control, API composition, request/response transformation, and load balancing, contribute to enhanced security, performance, and scalability. Additionally, features like caching, rate limiting, and logging further improve system efficiency and user experience. Overall, the API gateway plays a vital role in providing a unified, efficient, and secure access layer to the services, promoting a seamless and cohesive ecosystem.

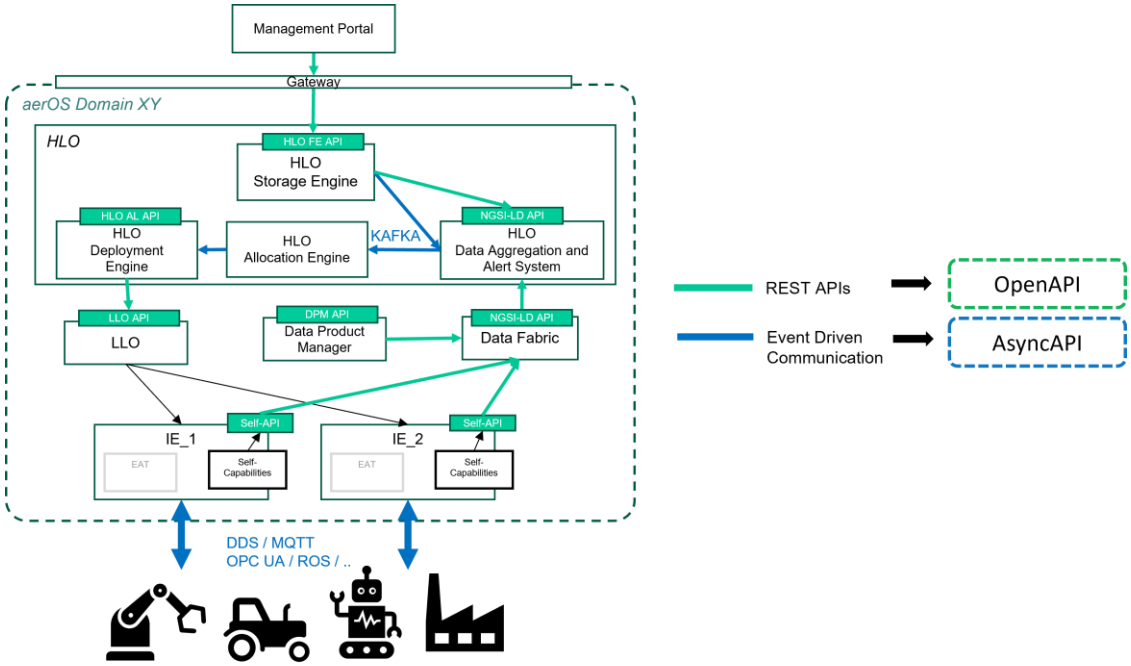


Figure 18. aerOS APIs: REST APIs and event driven communication

In the context of aerOS, REST-based APIs (such as HLO-FE, LLO, and self-* capabilities) are defined using the OpenAPI standard specification. However, as technology continues to evolve, there is a growing need for standardized specifications of asynchronous interfaces—a capability that OpenAPI does not inherently provide. To address this limitation, the AsyncAPI initiative has emerged, aiming to establish an industrial standard for specifying asynchronous interfaces.

Similar to OpenAPI, the use of AsyncAPI would benefit both core and auxiliary services in aerOS by providing a framework to describe interfaces for protocols such as Kafka and MQTT. For instance, the High-Level Orchestrator (HLO) employs Redpanda to facilitate event-driven communication. Additionally, auxiliary services and pilot projects within this framework can utilize AsyncAPI to standardize the sharing of IoT data structures across the data fabric, thus integrating various industrial interfaces, including ROS2, DDS, OPC UA, and MQTT. OpenAPI would remain being the preferred choice for REST-based interfaces due to its widespread adoption. By using both OpenAPI and AsyncAPI, aerOS ensures that all API information is documented consistently. Additionally, both standards allow for code generation, simplifying the creation of servers or clients.

Additionally, the integration of low-code tools as auxiliary services within the aerOS project enhances flexibility to trigger actions over the APIs. Behaviour trees, functioning as graphical low-code interfaces, enable users to define triggers and adjust parameters interactively. These behaviour trees do not directly orchestrate services within the aerOS domains; that role is specifically reserved for the High-Level Orchestrator (HLO) and Low-Level Orchestrator (LLO). Instead, behaviour trees trigger functionalities within already running applications, activating specific service functionalities without initiating or terminating the services themselves. This user-friendly approach allows for easy modification of operational logic.

5.5.8. aerOS management framework

The Meta-OS developed in aerOS must be managed by end users through the use of a common and recognisable interface, as in a traditional operating system, in which users can use a terminal or a command shell for this purpose. For instance, these tools allow users to install, uninstall and run programs or manage the set of users with the proper rights and permissions to interact with the operating system. In addition, OS have evolved to provide these users with a simpler and cleaner way of interacting with the system: a visual interface (e.g., a desktop) on top of the internal processes, so that they can manage the operating system using user-friendly "frontends", while the actual processes are still being executed in the background, which means that they are hidden to the users.

In aerOS, the intention is to follow the same approach that is fully adopted in traditional operating systems, so it have been decided to create one component to act as a single window for end users to manage the Meta-OS: the **aerOS Management Portal**. This means that the portal is deployed in a unique domain, named as **"Entrypoint domain"**. However, the portal provides migration capabilities to be moved to another domain or IE due to a user requirement or an unexpected failure. Thus, this allows to avoid the loss of the unique management entrypoint, as is the case with high availability systems. As a global reflection, this approach is completely aligned with the principles of aerOS of **flexibility and decentralisation** ("single entrypoint" does not mean a centralized computing approach). This has been expressed more succinctly in Section 5.4.2.

This user-friendly dashboard, developed as a modern Single-page web application (SPA), acts as a frontend for performing operations regarding the Meta-OS, which are finally performed by the aerOS Basic Services in the background. For example, an administrator can confirm the addition of a new domain or of a new IE to the continuum using the portal, but the inclusion of this domain is managed by the aerOS Federator component. The interaction between the User Interface (UI) of the Management Portal and those basic services is undertaken by the backend of the aerOS Management Portal. Furthermore, this dashboard displays useful information gathered by these services to inform users of the status of the continuum (topology graph of the added domains, deployed services, state of domains and their IEs, etc) in real time. Finally, it is important to highlight that this portal does not add new capabilities to the Meta-OS, but leverages the capabilities offered by aerOS to respond to user requests from a functional point of view. It, however, serves as the single window access for interacting with the continuum.

This set of capabilities or actions which can be performed by users have been properly defined based on previously specified requirements by technical developers and potential end users of aerOS (e.g., use cases leaders). In that regard, a **benchmarking tool** has been identified as necessary to help this kind of end users to

improve their knowledge about the current performance of the aerOS Meta-OS. This tool is fed with data from the continuum to provide two main functionalities that complements the Management portal:

1) Benchmarking and comparison of IE/domains performance against known standards and methodologies, such as TPCx-IoT and RFC 2544.
2) Internal Technical KPIs dashboard, which provides a snapshot of these KPIs defined for aerOS technical components in deliverable D5.2 and which are directly measurable from the aerOS stack.

When it comes to Meta-OS end users' management, aerOS envisions the definition of a set of roles and permissions to be applied to them. Using a more practical example, the content that is displayed in the dashboard changes based on the role of the logged user, e.g., an administrator can only deploy certain services in a set of domains on which he has rights and permissions, as an example. In addition, a common list of user roles has been implemented for all the domains that compose the continuum, as described in the aerOS cybersecurity components of the architecture (Section 5.5.4).

Previously introduced in Section 5.4.2 the **Entrypoint balancer** is an important part of the aerOS management framework (stuck to the Management portal) to avoid the addition of an additional centralization point, so that it emphasises the decentralised nature of aerOS. After conducting a thorough review of the cutting-edge network load balancers, it has been determined that the most suitable LB algorithm in the case of the entrypoint balancer is the Least Connection type (or one of its modifications). This algorithm includes a straightforwardly updatable weighting function that is fed with metrics from the data fabric, such as the number of orchestration requests managed by each domain's HLO or IE capabilities. Moreover, more detailed technical information can be consulted in "D4.2 Software for delivering intelligence at the edge intermediate release" and future D4.3.

However, the aerOS Management Framework reaches beyond the capabilities offered only by the Management Portal. It also includes other management tools that oversee the creation and maintenance of the federation mechanisms between the multiple aerOS domains that build the continuum. Here comes into the scene the **aerOS Federator**, specifically the block related to the registration and discovery of these domains (Domain Registry and Discovery). The actions within the scope of this block are not performed directly by end users such as the ones of the Portal, but are smart, automatically conducted to react to some user actions, such as the creation of new domains or the modification of the IEs that belong to an existing domain. **aerOS is not a centralized solution**, so by taking advantage of the mechanisms described in Section 5.4.3 (aerOS distributed state repository), this block is on charge of establishing the appropriate mechanisms to achieve a fully federated and decentralized architecture among the aerOS domains. The set of distributed NGSI-LD Context Brokers, conforming the Distributed State Network of Brokers (DSNB), plays a major role in this architectural block.

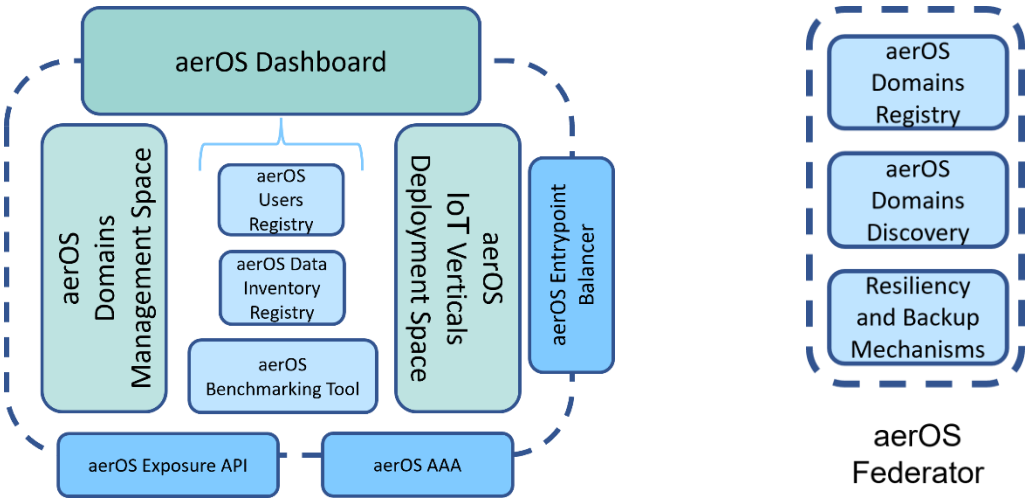


Figure 19. aerOS Management Framework (left: aerOS Management Portal, right: aerOS Federator)

5.6. aerOS auxiliary services

In aerOS, the auxiliary services compose the last category of the services considered in the architecture. They exist to provide complementary functionality across the continuum, without having an explicit core functionality (such as cybersecurity, data management, etc.). These services can be considered as commodities that aerOS will research and deliver to provide flexibility and innovation across the whole ecosystem. These auxiliary services are: (i) Auxiliary AI (AI workflows in the continuum and use cases deployments) and (ii) Embedded Analytics.

5.6.1. aerOS auxiliary AI

The proliferation of IoT devices and the rising popularity of IoT-Edge-Cloud infrastructure deployments enables a new approach to prepare, use, and maintain AI solutions within different use cases coming from various domains. Specifically, model training can be done in a decentralized fashion without moving the data to a central location (e.g., cloud) and exploiting their locality and processing "closer to their origin". This approach, called Federated Learning (FL), is attractive as it allows to reduce the computational load of a single infrastructure element and scale the system dynamically. Moreover, it helps to mitigate some privacy issues that may arise from data owners. Furthermore, often there is a need to deploy AI-related services to perform inference closer to the edge. In this case, the model can be wrapped as a service exposing API and deployed in the continuum. This may require application of additional mechanisms to support frugality, i.e., techniques that allow the use of AI models in resource-restricted conditions (limited processing power or memory, low network bandwidth). Here, model reduction with quantization and pruning can be applied and/or a "light" service deployment. Finally, to provide accountable and trustworthy AI-driven solutions explainability/interpretability of AI models should be possible by providing a dedicated service of type function as a service. Note that explainability should be considered only when required (e.g. for critical functions) because it adds additional computational costs and may influence the performance of the overall solution.

The auxiliary AI to be deployed in aerOS can be divided in two main blocks, which are described in the following subsections.

5.6.1.1. Control of AI workflows in the continuum

aerOS aux AI services cover different functionalities required to execute AI tasks using aerOS infrastructure. AI tasks cover two main scenarios: federated learning and distributed inference. AI tasks can be decomposed into sub-tasks that may have dedicated requirements and their execution can span over several IEs (functionality divided between services deployed over different IEs). In the simplest case AI task may have only one sub-task, i.e., workflow consisting of one step, e.g. deployment of a service offering predictions done by ML model. Note that AI task is a specialization of a general task that can be executed using aerOS infrastructure.

Within aerOS dedicated services are prepared to: monitor and orchestrate specific task execution (AI Task [n] Controller) and execute an AI sub-task (AI Local Executor). They are deployed as auxiliary services on the aerOS infrastructure using aerOS service deployment and orchestration mechanisms. AI Task [n] Controller is responsible for task "n" and is collaborating with AI Task Executor services that are executing parts of this tasks on different IEs. In federated learning, at the end of the process, AI Task [n] Controller will have a new shared model trained in a federated way within AI Local Executor services.

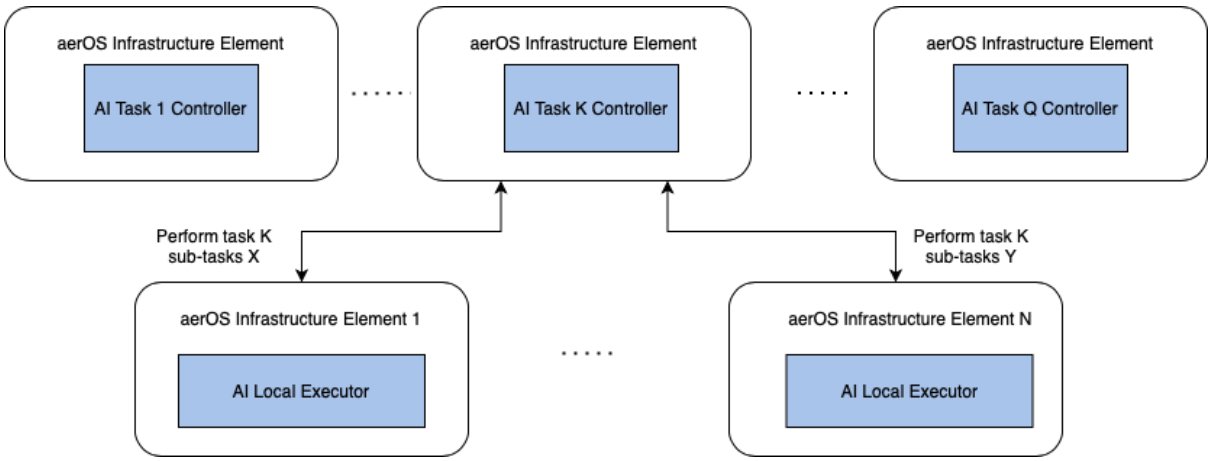


Figure 20. AI workflow in the continuum

Frugal solutions allow deployment and execution in resource-restricted environments. Frugal applications can be deemed as the ones most suitable to be deployed close to the edge, instead of a centralized deployment. Consequently, frugality can be treated as an "add-on" to decentralized AI.

Generally, frugality requirements are very use case specific and different techniques can be used to meet objectives of considered scenarios. Frugality support studied in aerOS covers availability of "minimized" service deployment (that allow to run services on a resource-restricted devices) and mechanisms for model reduction such as quantization and pruning (that allow to reduce the model disk size and improve inference speed). Pruning in neural networks is a technique used to reduce the size of a model by removing parts of the network that contribute little to its output. The main goal of pruning is to improve the efficiency of neural networks without significantly sacrificing accuracy. This technique can be essential for deploying models on devices with limited computational resources, such as smartphones or embedded systems. Pruning reduces the number of parameters in the model, which decreases its storage requirements. Next, with fewer calculations, the pruned network can offer faster inference times, making it more suitable for real-time applications. Smaller models require fewer computational resources, which can lead to lower power consumption. Quantization is a technique used to reduce the precision of the numbers representing the weights and activations within a model. Quantization works by mapping a large range of values to a smaller one, often through rounding operations. This process is vital for deploying deep learning models on resource-constrained devices such as mobile phones, embedded systems, and IoT devices, as it can significantly reduce the model's memory footprint and speed up inference while maintaining acceptable levels of accuracy. The main challenge in quantization is maintaining the model's accuracy with reduced numerical precision, which requires careful selection of the quantization scheme and possibly adjustments to the model or training procedure.

5.6.1.2. Explainable AI

AI is a powerful technology that can perform complex tasks, such as image recognition, natural language processing, and decision making, that normally require human intelligence. However, many AI systems are not transparent or interpretable, meaning that their internal logic and reasoning are hidden or difficult to understand by humans. This poses a challenge for trust, accountability, and ethics in AI applications, especially when they affect human lives, rights, or well-being.

Explainable AI (XAI) is a concept in which the results of an AI solution can be understood by humans. It can be used to describe an AI model, its expected impact, potential biases, and to help characterise model accuracy, fairness, transparency, and outcomes in AI-powered decision. XAI is crucial for an organisation in building trust and confidence when putting AI models into production.

There are different types and levels of explainability in AI, depending on the audience, the context, and the purpose of the explanation. For example, a technical explanation may be suitable for developers or regulators who need to verify the correctness or compliance of an AI system, while a layman explanation may be appropriate for end-users or customers who need to understand the rationale or implications of an AI recommendation or prediction.

According to NIST, there are four principles for explainable AI systems:

- Explanation: AI systems should deliver accompanying evidence or reasons for all outputs.
- Meaningful: AI systems should provide explanations that are understandable to individual users.
- Explanation accuracy: AI systems should provide explanations that correctly reflect the system's process for generating the output.
- Knowledge limits: AI systems should acknowledge the limits of their knowledge and indicate when they reach sufficient confidence in their output.

To achieve these principles, various methods and techniques have been proposed and developed in the field of XAI. These include:

- Transparent algorithms: These are algorithms that are inherently interpretable or comprehensible by design, such as decision trees, rule-based systems, or linear models.
- Post-hoc explanations: These are explanations that are generated after the model has been trained or deployed, such as feature importance scores, local approximations, or counterfactual examples.
- Interactive explanations: These are explanations that are elicited through user feedback or queries, such as natural language dialogues, visualizations, or interactive interfaces.

In aerOS, AI is used internally to support intelligent decision making when managing the continuum, and externally to enable executing of arbitrary AI tasks using aerOS infrastructure. In both cases, the need may arise to explain and/or interpret predictions made by ML models. The objective for explainability support in aerOS is to prepare mechanisms to optionally "plug-in" such functionality in the AI task execution. To this aim, a service will be prepared - AI Explainability Service - for handling predefined cases like the interpretability of HLO allocator decisions. However, it will also provide methods that can be used for a more comprehensive number of use cases. Various SotA methods for explainability in AI have been analyzed and the most promising ones are based on calculating Shapley values to provide users with easy-to-understand explanations that are mathematically provable. Note that frugality mechanisms and explainability/interpretability have influence on AI model accuracy (and other metrics) so their inclusion in AI task workflow shall be optional.

Although explainability is not directly related to the architecture, it is likely that successful validations will lead to a set of operational and technological recommendations.

5.6.2. Embedded analytics

As described in "D2.6 aerOS architecture definition (1)", The aerOS Embedded Analytics Tool (EAT) can be compartmentalised into three roles; these are the analytics framework, function authoring and visualisation. This section will reiterate these primary roles to provide a final architectural view of EAT with additional detailing of the aerOS Function Template structure. As a result, a holistic EAT architecture is presented.

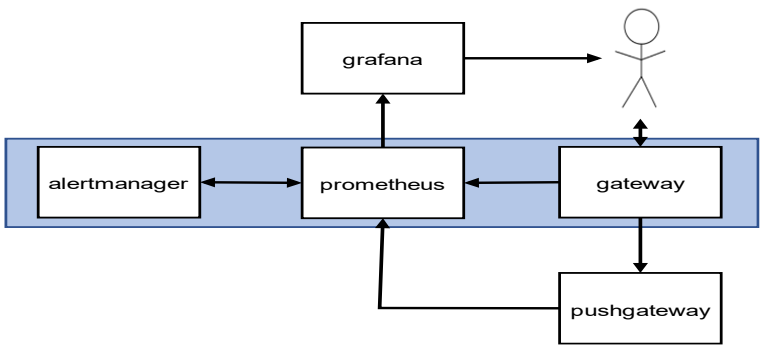


Figure 21. Embedded Analytics Tool Architecture

EAT (illustrated in Figure 21) provides a framework for the design, implementation, and deployment of specialised functions. These may be straightforward policy-based functions for validation use cases or intelligence-based models for smart decision making. The framework supports multiple dashboards for operations (*gateway*) and visualisation (Grafana). The *pushgateway* allows in-function metrics to be exposed to Prometheus monitoring which is then visualised through Grafana to the user. The *alertmanager* component monitors Prometheus metrics related to the health of the gateway, alerting the user if required. All these components are hosted on the aerOS Gitlab and are installed as nodes in a Kubernetes cluster through *Helm charts*. To access these features of EAT, functions must be created using the aerOS template.

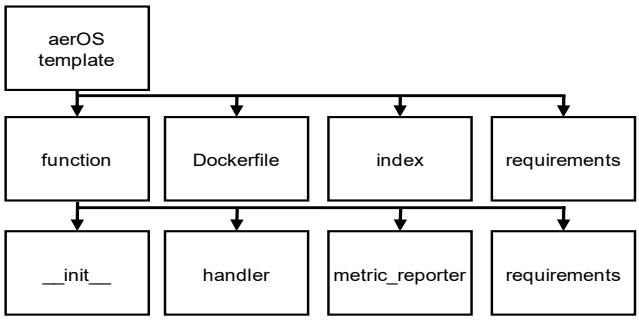


Figure 22. aerOS Template for authorised function on EAT

The aerOS template is presented in Figure 22 and details the two layers of the function template. The first layer contains information specific to how the function interacts with EAT. This includes the image build file (*Dockerfile*), what libraries Docker needs to compile the function image (requirements) and the wrapper used to interface with the function (index). The second layer contains information specific to the function operation. The *__init__* component contains operations to carry out when the function is deployed, this includes building the Grafana dashboard where in-function metrics will be visualised. The handler component contains operations to carry out when the function is triggered, such as a policy or intelligence-based decision making. The *metric_reporter* component contains operations for exposing in-function metrics to Prometheus. These metrics can then be visualized through the dashboard instantiated when the function was deployed. The requirements component contains required libraries for handler executions. The aerOS template must be used to create EAT functions, this process is handled through the *faas-cli* application.

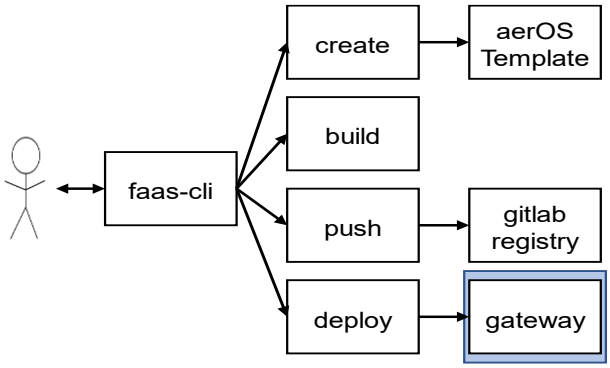


Figure 23. Function Authoring

The steps involved from the creation to the deployment of EAT functions is presented in Figure 23. The *faas-cli* application allows users to create EAT functions using the aerOS template. The application also enables the build, push, and deployment of functions to EAT. Build packages the EAT function as a Docker image and stores the image locally. Push allows the user to specify a container registry to store the image. Deploy takes an image, either locally or from a container registry and onboards the image as a node in the Kubernetes cluster. This allows users to dynamically deploy and update functions as versioning can be controlled at deployment.

Interfaces between EAT and other aerOS components can be viewed on two levels. The first level is the interface provided by EAT for the function i.e. REST, this interface is triggered through a HTTP request which provides a body that is passed into the function handler. The second level is interfaces established inside the function, such as queries to the Data Fabric (Section 5.5.2) or pushing metric to Prometheus. Both interfaces are available to users however the execution of a function will always require a HTTP request to trigger and a response to signify the execution has concluded.

The installation of EAT comes with three prepackaged functions, these functions provide generalised stratified sampling, anomaly detection and data drift detection based on Data Fabric models. These functions may require minor edits on an ad hoc basis, depending on the complexity of data models being used.

5.7. User services and global pilot services

From user services and pilots' point of view, multiple features and functionalities that enable IoT devices to communicate with cloud services and other IoT devices over a global network are required. Pilot services need to perform reliably under the constraints of limited memory and processing power of IoT devices which will be located at the pilot's premises:

- **P1- Industry-Data driven cognitive production lines:** Automated workstations, Automated Guided Vehicles (AGVs) for transportation within industrial facilities, sensors for ambient temperature and humidity, optical sensors, and computation servers.
- **P2- Facilities/Energy:** storage drives, power supplies, radiators, interfaces and controllers.
- **P3- Agriculture:** High performance computing platforms, ECUs to provide connectivity, and vehicle connectors.
- **P4- Transportation and logistics:** IPTV cameras for video streams
- **P5- Smart buildings:** temperature, humidity, and air quality sensors.

Taking into consideration the aforementioned devices and pilot needs, in the following lines global pilot and user requisites have led to consider the following required services defined for aerOS:

- **Portability service**

  Portability services are the features and functionalities that allow aerOS to adapt to different hardware and software platforms, and to interoperate with other systems and devices in a global network. The rationale behind this service is the variety of hardware and IoT devices that exist among the five pilots. This service may include the following functionalities:

  - *Abstraction:* aerOS provides an abstraction layer that hides the specific details of the underlying hardware and software, and that provides a standard interface for applications and services.
  - *Adaptability:* aerOS is able to adapt to the changing conditions of the environment, such as the variability of energy demand, resource availability, quality of service, security and privacy.
  - *Integration:* aerOS facilitates the integration of different systems and devices, both within and outside the specific sector of each pilot, through common protocols and formats of communication and data.
  - *Reusability:* aerOS enables the reuse of existing components and services, as well as the development of new modular and customizable components and services.

- **Scalability service**

  aerOS provides the ability to increase or decrease its size, capacities, performance and functionality according to the needs and demands of the users and applications run by the pilots. In this regard, it would include the following functionalities from the pilots' point of view:

  - *Elastic*: aerOS is able to scale elastically, that is, automatically adjust the resources allocated to each node or device according to the conditions of the pilot´s environment, such as energy demand,

resources demand or workload. This allows to improve energy efficiency, quality of service and resilience of the system.

- *Adaptable:* aerOS is able to scale adaptably, that is, modify its configuration to incorporate new technologies, standards and requirements. This allows to innovate and evolve with the market and the expectations of the users.
- *Horizontal scale:* aerOS is able to scale horizontally, that is, add or remove IEs or devices to the domains and networks without affecting the operation of the system. This allows to leverage distributed computing capacity and cloud storage.
- *Vertical scale:* aerOS is able to scale vertically, that is, increase or decrease the resources allocated to each node or device, such as memory, processor or bandwidth. This allows to optimize the use of resources and adapt to the variations of the workload.

- **Modularity**

aerOS design departs from a series of basic services, together with other functionalities (auxiliary services) that can be included as add-ons if so required by the application in the pilots. It may include the following functionalities:

- *Customisation:* aerOS enables the customisation of the system by allowing the users to select and combine different modules of services and functions according to their preferences and needs.
- *Variety*: aerOS enables the variety of the system by allowing the providers to offer different modules of products and services that can be configured in multiple ways, to serve heterogeneous customer demand.
- *Reconfiguration*: aerOS enables the reconfiguration of the system by allowing the users and providers to change or update the modules of products and services over time according to changing conditions or requirements.
- *Standardization*: aerOS enables the standardisation of the system by allowing the providers to use common interfaces, protocols, and formats for the modules of products and services that can facilitate integration, interoperability and compatibility.

- **Connectivity services**

This service allows to support different connectivity protocols, such as 5G, Ethernet, Wi-Fi, BLE, IEEE 802.15.4, among others. This would include the following functionalities:

- *Integration:* aerOS enables the integration of data and information from different sources and smart devices located at pilots´ premises.
- *Accessibility:* aerOS enables the accessibility of data and information to different pilot users.
- *Communication:* aerOS enables the communication of data and information between different devices, systems, and pilots' actors.
- *Interoperability*: aerOS enables the interoperability of data and information across different platforms, standards, and formats.

- **Security services**

This service will allow aerOS to include add-ons that bring security to the device by way of RBAC access, SSL support, and components and drivers for encryption. It would include the following functionalities:

- *Update*: aerOS enables the update of devices and systems in the product-service system. This can fix vulnerabilities and bugs and improve performance and functionality.
- *Authentication:* aerOS enables the authentication, authorization and access control via the cybersecurity components described in Section57. This can prevent unauthorized access and ensure data integrity to the pilots' assets.

- *Encryption:* aerOS enables the encryption of data and information in transit and at rest. This can protect data confidentiality and privacy coming from pilots' premises.
- Firewall: aerOS enables the firewall of devices and systems in the product-service system by using rules, policies or filters. This can block malicious traffic and prevent cyberattacks, such as denial-of-service (DoS) or ransomware that can affect pilots' facilities operations.

6. aerOS Reference Architecture

This section presents the revised aerOS architecture, evolved based on the experience and validation input from the MVP deployment. The deployment of aerOS components and their integration and interaction within the MVP, while not altering the initial Reference Architecture (RA), has influenced the final different viewpoints under which it is inspected and presented. Although the changes since the initial submission are incremental, this document provides the comprehensive, updated RA, ensuring it is self-contained and serves as the single point of reference for the aerOS architecture. Having introduced the key concepts and components of the aerOS architecture in section 5 and adhering to the design guidelines outlined in section 2, now the aerOS RA is presented through a set of viewpoints as identified within the described methodology. Firstly, it is provided an overview of the entire architecture, from a high-level viewpoint, including a descriptive guide for an IoT service deployment instantiation, and delve into different instantiations of aerOS Entities in subsection 6.1. Following that, the functional viewpoint (in subsection 6.2) is introduced, which describes the role of the main components of the aerOS systems. Moving on, subsection 6.3 focuses on the process view of the RA, outlining how components communicate to accomplish fundamental functionalities. Moreover, some activity and sequence diagrams that detail the runtime behavior of aerOS are provided, demonstrating interactions between different components among aerOS components and how aerOS interfaces with external actors and components. Subsequently, subsection 6.4 explores the data viewpoint, and how aerOS IEs, aerOS domains and IoT devices are included as new data sources and data consumers and dynamically become part of the distributed knowledge graph. The role of Context Brokers as interconnected Context Registries (DSNB) enabling the federation across aerOS domains which is fundamental in aerOS ecosystem is described. Next, subsection 6.5 delves into the deployment viewpoint of the RA, which covers runtime operations. It presents the software component topology on the physical layer and the interconnections between these components during aerOS domains and the vertical IoT services deployment. Finally, in subsection 6.6, it is discussed the business viewpoint of the architecture, which serves as a guide for developing application components and supporting the decision-making process of stakeholders involved.

6.1. High-level view

aerOS proposes an overarching approach that unifies access to, and usage of, network and computing resources from edge to cloud, offering a transparent infrastructure layer. On top of this hosting underlay, aerOS design offers the substrate which can seamlessly enable the deployment of IoT applications without any adaptation regardless of the selected hosting node. Based on these inherent functionalities, aerOS establishes a common execution environment which removes any need for adaptations to various runtimes or architectures and thus exposes underlying resources as a continuum. Even more, and going one step further, aerOS acting as expected from a Meta-OS, provides the management and orchestrating overlay which overviews, secures and automates services orchestration and resources management over the established continuum. As an outcome IoT developers and resource owners can trust aerOS Meta-OS to host their efforts and resources and can further federate, share, and (re)use physical or virtual resources, data, and application components. All of this is achieved with the embedding of advanced AI techniques for best results while maintaining full ownership, governance, and security.

Thus, aerOS addresses the issue which emerges from the fact that numerous isolated processing units and private computing islands with restricted resources, lack the services needed to implement and deploy comprehensive IoT solutions. This is particularly relevant in both industrial and personalized contexts. Many computing units and private networks currently function merely as data concentrators, forwarding large volumes of data to centralized commercial cloud infrastructures operated by a limited number of service providers. This isolation prevents the effective utilization of existing computing power at the edge, which is crucial for many IoT

solutions requiring more computational resources. This deprives vertical IoT stakeholders from having full control of their services and governance of their data. Additionally, although a variety of already developed services exist for most of industry verticals they cannot be reused and each time another organisation wants to solve similar problems they have to re-develop a solution from scratch, or to fully adapt their existing operating runtime environment, as there is no “lingua franca” for IoT service developers to provide a common underlying layer where existing solutions can run and address similar cases with similar requirements.

The proposed approach outlines a secure federation of individual computing resources, enhanced by privacy and security enforcement mechanisms and technologies, to create a network and computing continuum. This federation forms an environment comprising a combined pool of resources capable of transparently hosting parts or the entirety of intended IoT tasks as close as possible to data sources, generally as close as possible to their declared requirements, while maintaining complete control over data availability and governance. Industry verticals or individual users can contribute "off the shelf" resources, even with restricted capabilities, as long as they support virtualized containerization environments. These resources can be integrated, through a well-documented and straightforward procedure, as aerOS IEs or domains, becoming part of a larger common execution ecosystem where services can leverage existing resources or services across the aerOS continuum. The fulfilment of the following core objectives is targeted by the proposed solution:

- To allow isolated resources to be exposed and orchestrated as a continuum within which IoT service developers, from different verticals, may, transparently, deploy their applications, accompanied with a set of requirements, without having to manage all the underlying complexity regarding compute, network, and operating resources.
- To provide a unified, cloud-native execution environment built on microservices architecture which will extensively utilise container-oriented virtualisation runtimes, taking advantage thus of cloud-native benefits, including simplified orchestration, enhanced portability, reliable reproducibility, and seamless scaling of applications.
- To take advantage of the most suitable resources, and introduce monitoring, predictive and orchestration mechanisms to enforce adherence to user SLAs.
- To provide information and IoT service sharing, that benefits all users, while introducing secure control and privacy governance mechanisms.

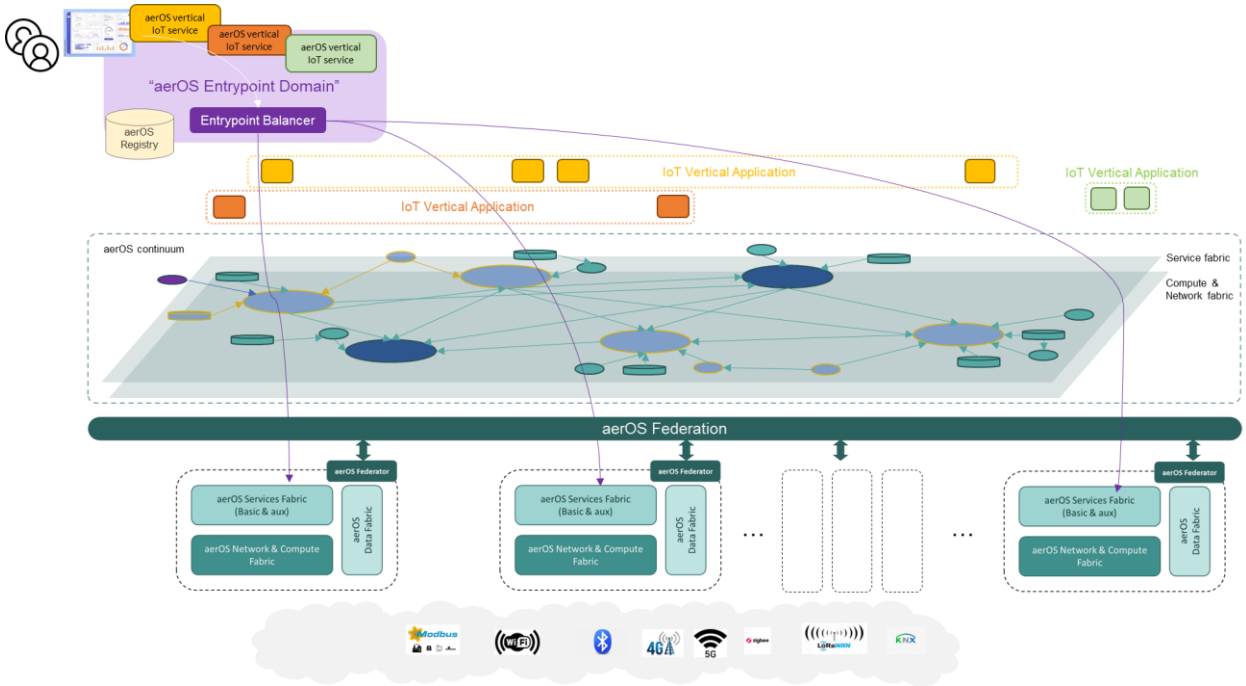


Figure 24. aerOS high-level View

The suggested architecture introduces a guided procedure for the deployment of an aerOS domain on top of available computing and network resources. These resources (known as IEs) may vary in terms of capabilities, architecture, units' number, and can either be "bare metal" resources or, and possibly mixed with, virtual machines. The deployment of aerOS basic, and selected auxiliary, services on top of these resources provides a seamless integration with the rest of the aerOS ecosystem and a common execution environment. Figure 22 introduces a high-level view of the aerOS ecosystem where, by rendering of isolated resources into aerOS domains and making thus use of aerOS services, a continuum of IEs is exposed as a common federated infrastructure ready to transparently host IoT services according to users' deployment requests and under the orchestration of aerOS Meta-OS. Users' IoT services deployments requests are submitted via the "aerOS Entrypoint" which provides a guided, and based on a multitude of templates, IoT application deployment process.

There are cases when the requested SLA of the IoT service dictates that it should be broken into more than one component. For instance, in an AI workflow, parts of the workflow requiring direct access to data might need to be placed in an edge aerOS domain, while other parts requiring more intensive processing could be deployed in a public or private cloud domain that supports specific AI processing capabilities. aerOS, by meeting the objectives set above, considers all candidates across the continuum and transparently deploys the components on the chosen domains without the user needing to adapt any part of the application or worry about data ownership and privacy. The submitted application, leveraging smart orchestration, can be deployed within a user-selected domain or be set for the most efficient placement based on requirements for computing resources or data consumption. The aerOS orchestrator manages this by utilizing all available federated information across the continuum, doing so in a way completely transparent to the users. As a result, a pool of resources hosts the entire application under an efficient placement strategy, allowing verticals to receive the requested services without needing to understand the intrinsic arrangements.

Thus, from a high-level view it is obvious that aerOS stakeholders are supported to easily render their resources as aerOS enabled IEs and register them as a domain within aerOS. Subsequently, the process of IoT service deployment across the continuum, part of which is their registered domain, is transparent to them. aerOS AI enabled federated orchestration takes care of hiding all the details related to the most efficient placement and the common aerOS underlying runtime will enable their placement to any node of a plethora of underlying hosts. aerOS federated orchestration functionality, and its success, is based on the provided capability to have a real time perception of resources provisioning and availability across the whole continuum from edge to cloud. Similarly, user IoT services can consume data produced in other domains in the continuum without the explicit knowledge of where they come from, and how to parse and interpret these data. Both these features are based on data interoperability. Data interoperability is part of the Data Fabric features that handle data as a product and enable metrics exchange and capabilities exposure among infrastructure elements and domains. In the same way, data interoperability is enabled within industry verticals' applications produced data. In the figure above it is obvious that data fabric is constituted from aerOS components deployed within each aerOS domain. Data interoperability, a provision of aerOS data fabric, is important within aerOS architecture. Existing smart data ontologies will enforce this interoperability for industry verticals and an aerOS knowledge graph which supports a management information model, developed and still extended with aerOS project, will offer this homogenized status exchange among all infrastructure elements across the continuum.

Thus, from a high-level perspective, aerOS stakeholders can easily render their resources as aerOS-enabled IEs and register them as a domain within aerOS. Subsequently, the process of IoT service deployment across the continuum, including their registered domain, becomes transparent to them. The AI-enabled federated orchestration within aerOS manages all the details related to the most efficient placement, utilizing a common aerOS underlying runtime to enable the deployment of services to any node within the diverse pool of underlying hosts. The success of aerOS federated orchestration relies on its ability to maintain a real-time perception of resource provisioning and availability across the entire continuum, from edge to cloud. Similarly, user IoT services can consume data produced in other domains within the continuum without needing explicit knowledge of their origin or how to parse and interpret them. This is made possible through data interoperability, a key feature of the Data Fabric that handles data as a product and facilitates metrics exchange and capabilities exposure among infrastructure elements and domains. Data interoperability is crucial within aerOS architecture. It is supported by existing smart models that enforce interoperability for industry verticals and an aerOS knowledge graph that maintains a management information model. This model, developed and continuously

extended within the aerOS project, ensures homogenized status exchange among all infrastructure elements across the continuum. As illustrated in the figure above, the Data Fabric comprises aerOS components deployed within each aerOS domain reinforcing its significance within the aerOS ecosystem.

The whole process, of sharing and using resources in aerOS federated environment, is framed with the appropriate security mechanisms, which provide both data protection and services access control on the hosting domains. Data governance mechanisms enable the control over which data are allowed to be shared with others. Domains' security mechanisms decide whether to provide the domains' resources for external domains requests for services deployment. Even more, trust scores, attributed to each aerOS domains and IE, are taken into consideration, within the orchestration process when choosing the most appropriate and secure domains to deploy part of the tasks.

6.2. Functional view

To implement the high-level vision and achieve its objectives, aerOS relies on facilities provided by seamlessly interacting modules. These modules work together with various stakeholders and integrate with a diverse array of existing devices, platforms, systems, and data sources. The main functional blocks of the aerOS ecosystem, which are crucial for registering resources and deploying services within the aerOS continuum, are detailed in this functional view of the reference architecture. We describe these functional components and highlight their importance in the realisation of the vision set in the high-level view of aerOS.

aerOS IE maps to the high-level concept of rendering existing processing units, either physical or virtual, as aerOS-enabled IEs. IE abstracts all architectural variety of underlying substrate and provides an execution layer ready to host IoT services under a strictly secured and monitored way. Basic functionality of the IE is to report its status continuously and periodically regarding all available and disposable resources and hosted services status. This information becomes part of the continuum knowledge graph and is federated, and other parts of the continuum may request access to available resources, to host services based on its capabilities. At the end is the minimum unit within aerOS providing processing power and computational capabilities. In the case these capabilities are match for the requirements of a service deployment request and under the condition it has the availability it can serve requests originating from anywhere across the continuum.

aerOS domain maps to the high-level concept of exposing available resources into the continuum. It acts as a wrapper around available computing and network resources, rendered to IEs, which integrates all the functionalities required to transparently enrol them at the disposal of the Meta-OS. It provides all the necessary abstraction overlays to securely integrate a set of IEs as part of the continuum. Consisting of one or more IEs, aerOS domain eventually serves as a fundamental unit within the ecosystem, acting as the primary interface for industry verticals to connect with the aerOS continuum. It offers a unified execution environment equipped with aerOS basic services, including federation, orchestration, and management capabilities. Resource owners register their available computing and networking resources as aerOS domains and can subsequently run IoT applications on top of them taking advantage of provided orchestration and lifecycle management facilities. Nevertheless, their applications are not restricted to be uniquely executed within this domain, because depending on their requirements, they can be partially or fully hosted on other domains across the continuum. Conversely, when resources within a registered domain are underutilized, they are made available to the Meta-OS to be used according to overall continuum demands. Each aerOS domain is integrated with the aerOS Entrypoint domain to incorporate security services, share resource capabilities via federators, and receive information about available resources across the continuum. This integration ensures a seamless and efficient operation within the aerOS ecosystem.

aerOS Entrypoint domain maps to the high-level view concept of providing a guided support for all users to interact with the continuum. It gives users the feeling of "sitting" and operating over a continuum stripped of all underlying distracting technical details. Users are registered with a certain rolein the aerOS continuum in the Entrypoint Domain. Subsequently services deployment requests and resources monitoring are graphically supported via the management dashboard. Based on that roles, users are enabled with different level of permissions regarding access and actions requests. While resembling other aerOS domains in structure, the Entrypoint domain distinguishes itself with additional capabilities specifically tailored for ecosystem management. Serving

as a central access point, with integrated graphical interface support, for operators and developers alike, it offers a suite of tools and interfaces for tasks such as registration in the continuum, service deployment, and resource management. By doing so, the Entrypoint domain ensures the seamless operation of the aerOS ecosystem. Migration of management services, if needed, is seamless ensuring that such transitions does not disrupt the ecosystem's functionality. Moreover, the Entrypoint domain takes charge of registering domains for decentralized state information propagation, ensuring that crucial data flows efficiently from the edge to the cloud. Integral to its operation is the integration of AAA control point. This integration provides a centralized source of truth for security and privacy control within the aerOS ecosystem, ensuring that stringent security measures are upheld across all interactions and operations.

As the narrative scales towards the high-level vision of aerOS, it becomes clear how resources become IEs, how domains are built on top of IEs, and how the Entrypoint domain supports access over these domains as a continuum. However, it still remains to align with the high-level vision where resources behave and expose themselves within the continuum, providing a unified execution environment for IoT developers. This is where the **aerOS Federated Orchestration** (see 5.4.2) and **aerOS Management Framework** (see 5.5.8) concepts and constructions come into play. They represent the structured interaction of all aerOS services to build the aerOS fabrics (as presented in Section 6). These frameworks enable the integration of diverse and scattered resources into a homogenized hosting infrastructure, over which services are transparently deployed and orchestrated by the aerOS Meta-OS. The aerOS Federated Orchestration provides each aerOS domain with the comprehensive view of all resources and capabilities dispersed from the edge to the cloud and based on this holistic perspective runs the orchestration process allocating the most efficient resources, across all the continuum, for each service deployment request. Integrating AI algorithms in the decisions engine, this orchestration over the federated resources across the continuum ensures optimal resource utilization and service performance across the federated environment. Meanwhile, the aerOS Management Framework focuses on the registration and discovery of core entities (such as aerOS Users and Domains) to form a federated environment. This framework is partially located in the Entrypoint domain and within each aerOS domain's federator component.

In Figure 23 the sequence of actions and involved concepts are illustrated. This diagram mentions the stakeholder’s interaction with the system and highlights the entities that have immediate interaction with the ecosystem, i.e., aerOS domains and aerOS management portal, abstracting at the same time all the layers and mechanisms that make sharing, federation, orchestration, and deployment possible within it. Users access aerOS provisions using the aerOS management portal. System Administrators register newly created domains to which they can have management access and can register to the aerOS federation process. IoT Service Developers can take advantage of a template driven process to deploy their applications, submitting desired characteristics which will be translated to orchestration-based placements for their services.

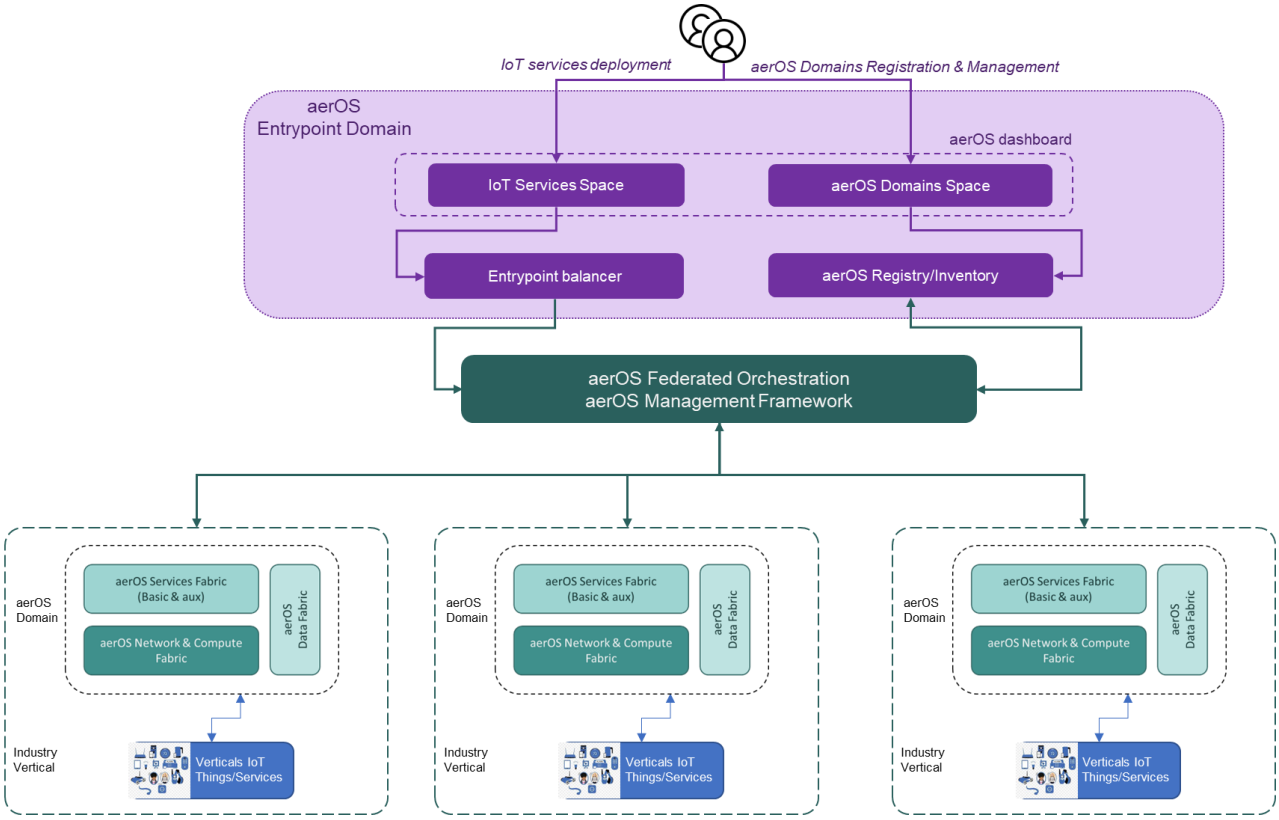


Figure 25. aerOS entities and actors overview

One path, "*aerOS Domains Registration & Management*", denotes the activities flow for aerOS domain operators and how domains are registered in the aerOS federation, and the other path "*IoT services deployment*" is relevant to IoT Service Developers and the flow following a service deployment request.

A more focused view on the functional blocks which implement these aerOS features in a domain level is depicted in Figure 24. As already discussed, each aerOS domain integrates three main building blocks acting as the functional components that make possible the implementation of the Compute and Network fabric, the Service Fabric, and the Data Fabric in a domain level and which finally support the domain integration to the aerOS continuum. Each aerOS domain provides an API exposure layer, technically implemented by an API gateway component, which is the single point of access to all underlying services. API exposure layer before "routing" external requests to responsible components transparently enforces security and privacy control as implemented within each domain from the AAA component.

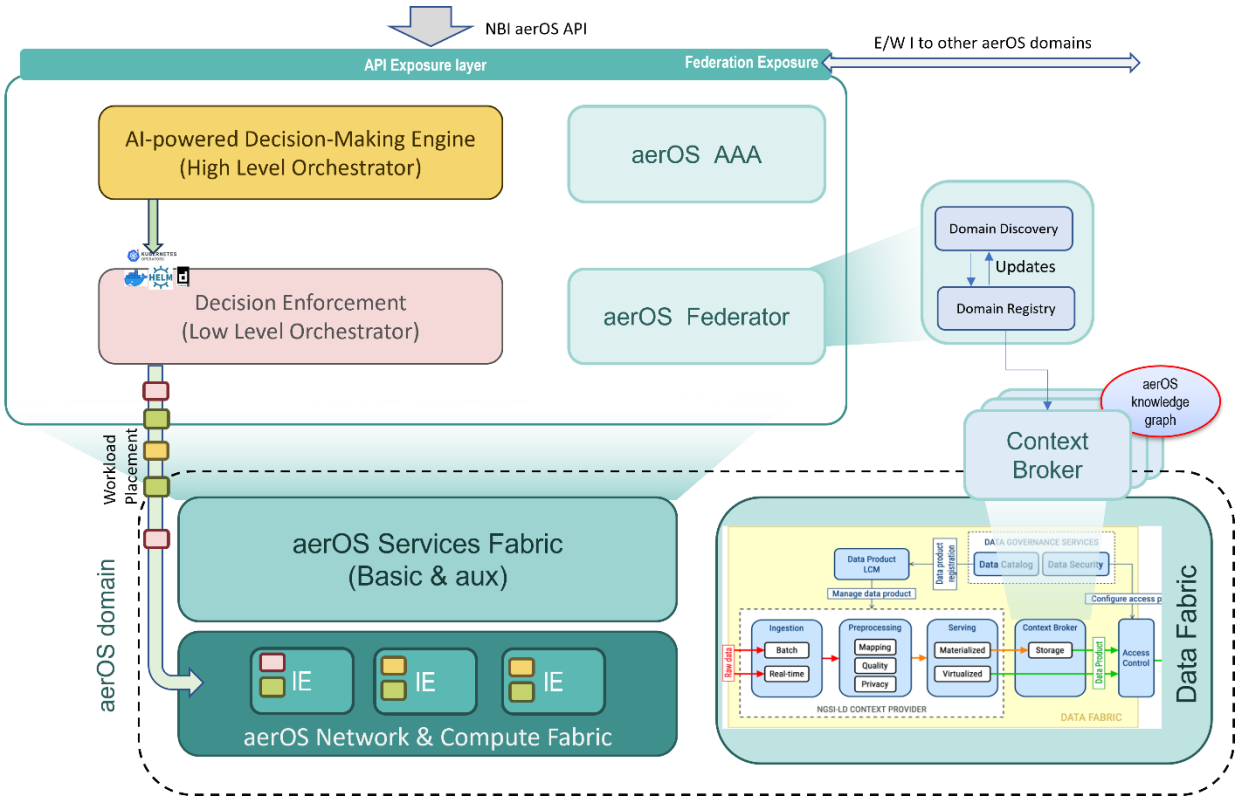


Figure 26. aerOS domain functional blocks

Figure 24 makes a clear positioning and interaction of aerOS domain functional blocks. At the bottom layer compute and network component provides the underlay for aerOS services to run. These services are aerOS management and orchestration services regarding both aerOS federated environment execution and verticals enforced IoT services. Data Fabric component spans across both of them and extracts, transforms and exposes data that subsequently feed aerOS management and orchestration services.

Data Fabric integrates data related to computing capabilities and deployed services in the domain. Initially it is responsible, building on Context Broker component capabilities, to establish context exchange with other domains and thus enable aerOS federation. It encompasses a domain registry and a domain discovery service which in fact implement aerOS federation services. These, register and keep receiving notifications from other aerOS domains regarding changes and updates done there and, the other way round, notify registered domains as to what is changing locally regarding resources' availability and services deployed modifications. All this communication is going through security and privacy component and exposed by domain API gateway. Additionally, Data Fabric's context broker component internally provides information of domain components' status and of other domains availability too, to components that need to support orchestration decisions. This means that exposed information flows directly towards High-Level Orchestrator's AI and trust management components. Thus, domain HLO can make the most efficient decisions, considering all aerOS continuum status, and either provide forward decisions to lower-level orchestrator or otherwise forward request to other aerOS domains HLO. LLO is, further, able to access underlying domains' compute and network resources and commission the actual placements. Thus, a closed loop that extends domain local restrictions is formulated. This closed loop entails a knowledge-based AI support which leads IoT services orchestration decisions across a pool of resources federated as a common execution environment.

6.3. Process view

The process view provides an overview of how the most relevant aerOS processes interact, communicate, and collaborate to achieve the desired functionality. Specifically, this view offers insights into the runtime behavior of aerOS, demonstrating how its processes seamlessly work together to accomplish their intended objectives.

Additionally, this view showcases the tasks and processes within the system, highlighting the interfaces with external elements and the interactions between different components. Moreover, it emphasizes the exchange of messages among these processes, facilitating effective communication and coordination.

The Process View of aerOS architecture has been drilled down into 8 processes, that respond to the most prominent and frequent activities/exchanges occurring over the Meta-OS:

- Installation and aggregation of computing resources to the continuum.
- Optimal deployment of a service by leveraging the aerOS continuum orchestration processes.
- Detailed interaction among the aerOS orchestration components.
- Service re-orchestration triggered by the self-orchestrator module.
- Secure access to data within the continuum through the Management Portal, regardless of its location.
- IoT Service (sensor monitoring and actuation) materialization.
- Trustworthy exchange of immutable and irreputable messages across the continuum.
- Decentralized AI task coordination and execution

All these processes provide a final and complete overview aimed at facilitating the understanding of the aerOS Reference Architecture.

Installation and aggregation of computing resources to the continuum

Figure 25 depicts the sequence diagram illustrating the installation and aggregation of computing resources to the continuum. This process begins with the SysAdmin, who possesses the necessary permissions to manage the computing resources available, initiating the aerOS installation on each computing resource.

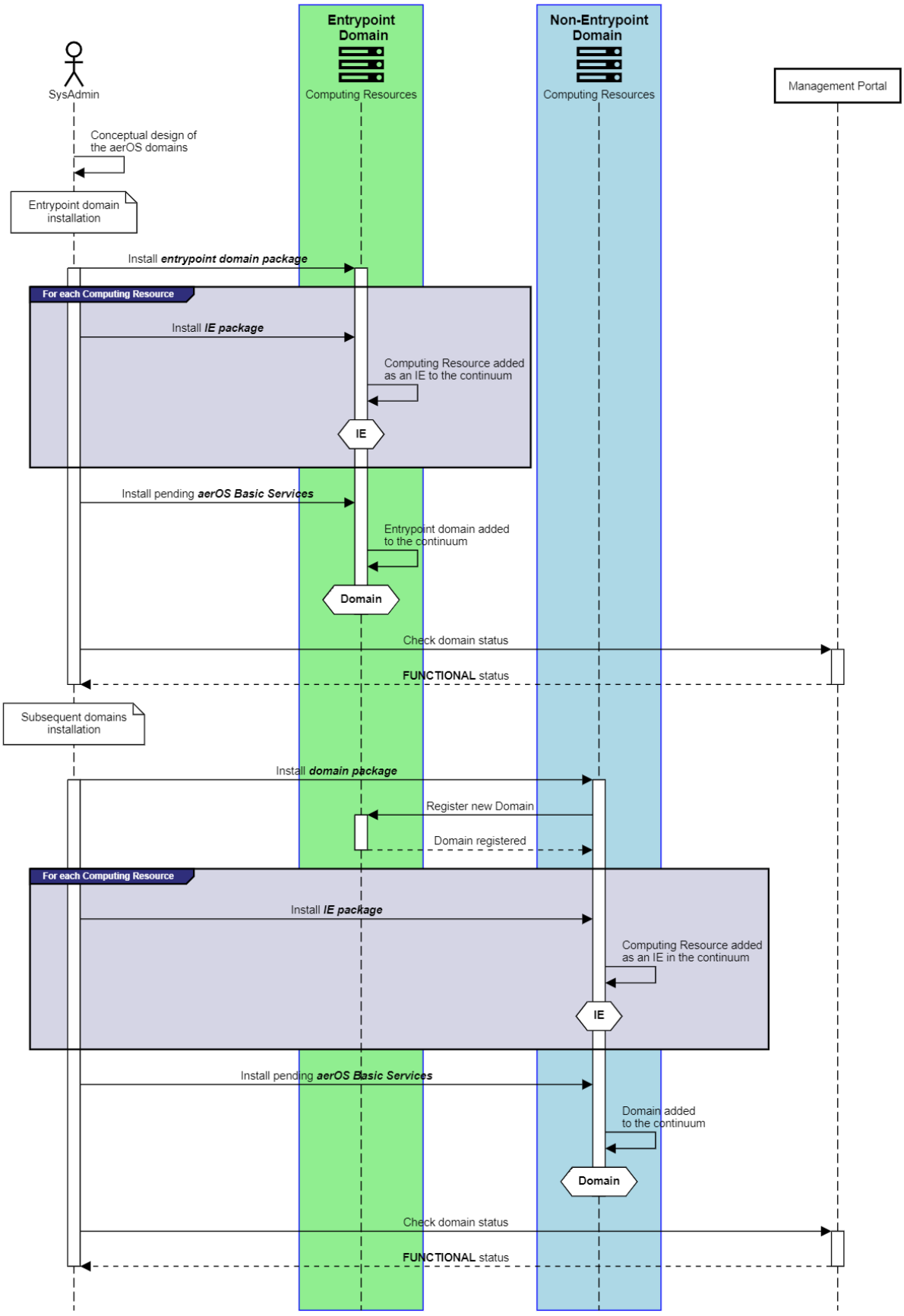


Figure 27. Installation and aggregation of computing resources to the continuum.

Optimal deployment of a service by leveraging the aerOS continuum orchestration process.

The next main process that has been detailed, represented as a sequence diagram Figure 26, illustrates the process of optimally deploying a service by leveraging the aerOS continuum orchestration processes. The process begins when a user, acting as IoT service deployer, declares their *Intention Blueprint* to deploy a service through the aerOS Management Portal. Then, the Management Portal interacts with the aerOS Orchestration to effectively coordinate the deployment process. The aerOS Orchestration determines the optimal location to initiate the service deployment within the continuum (HLO). Once an appropriate IE of the continuum is selected, the service is deployed (LLO). Upon successful deployment, the IoT service deployer receives a notification confirming the service's deployment.

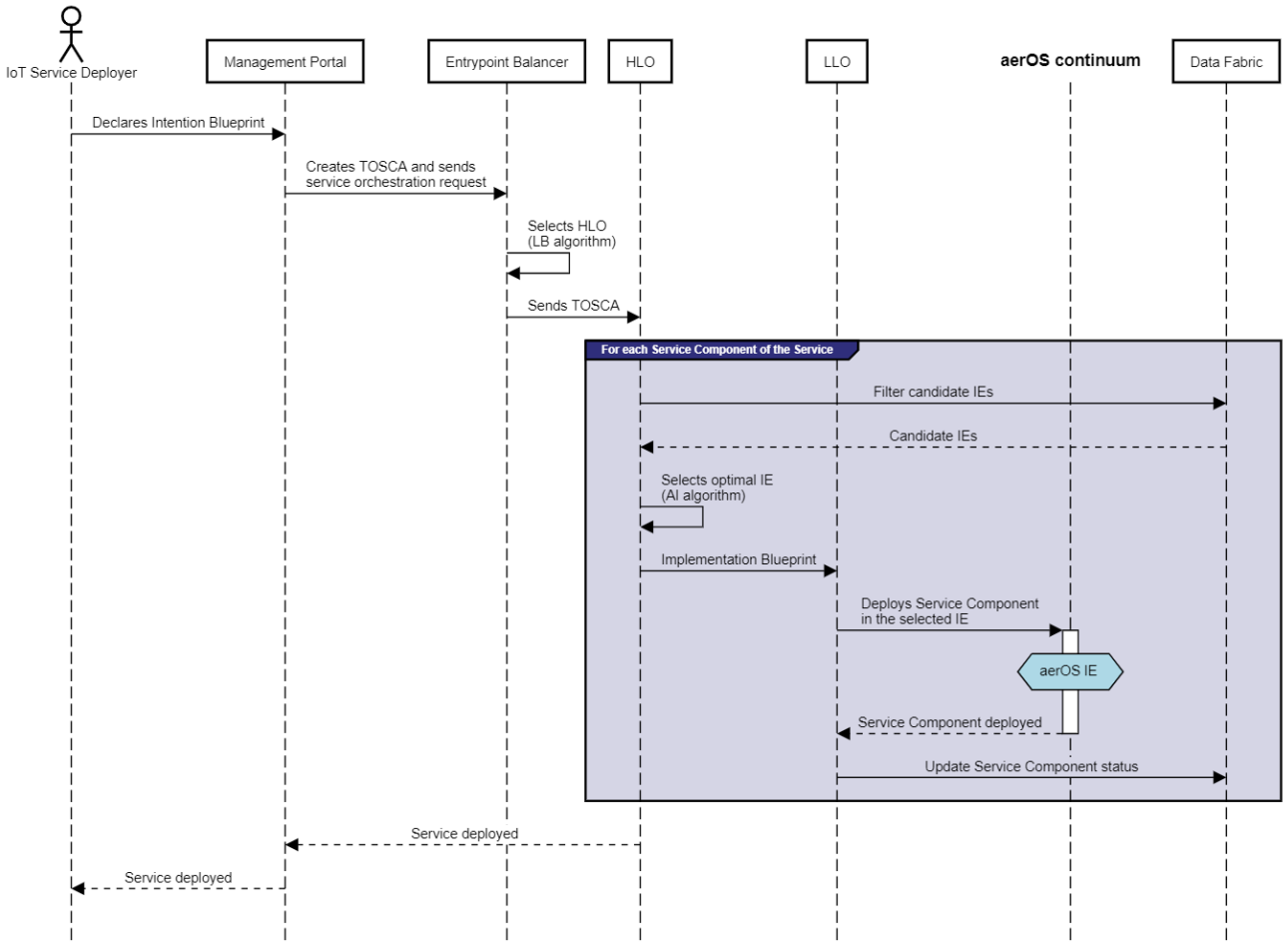


Figure 28. Optimally deploy a service by leveraging the aerOS continuum orchestration processes.

Detailed interaction among the aerOS orchestration components.

This process, which can be seen as an extension of the former process, aims to provide a more detailed view of the aerOS service orchestration process by focusing on the specific components involved in the aerOS orchestration, particularly the HLO. Therefore, interactions between the internal components of the HLO (Storage Engine, Data Aggregation, …), along with some key internal processes that occur in these components, are presented.

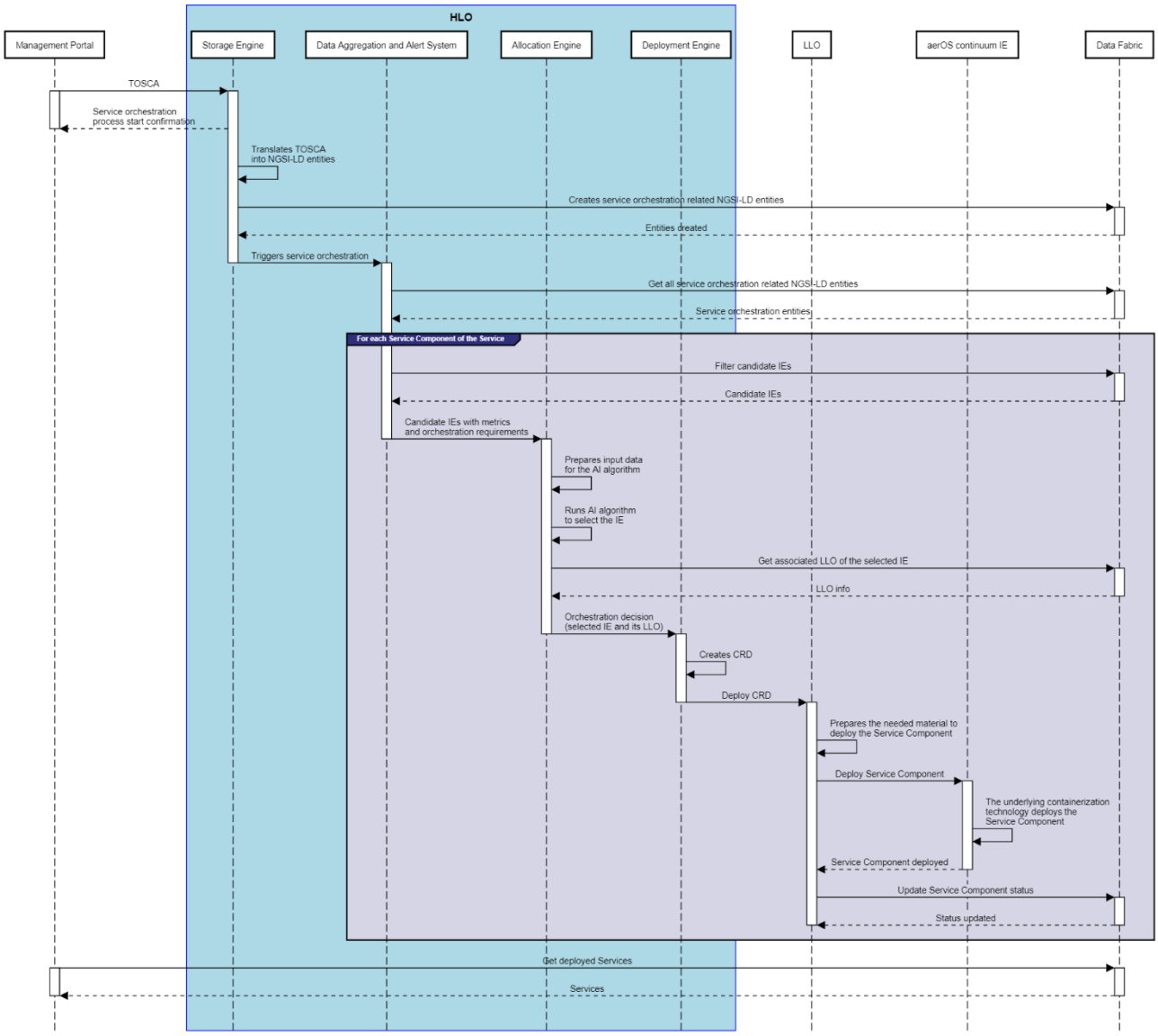


Figure 29. Detailed interaction among aerOS orchestration components.

Service re-orchestration triggered by the self-orchestrator module.

The last two diagrams were related to the aerOS orchestration process. Nevertheless, it is important to remind that not only is the orchestration in charge of the deployment of services, but also of service reallocation. In this regard, this diagram illustrates the process of re-orchestrating a service within an aerOS node. The process starts when the system administrator adds a rule in the node's Self-orchestrator module through its API. Every second, the node's self-awareness module updates the IE status data, sending a copy to the self-orchestrator module. When the self-orchestrator receives the information update, it converts it into facts to then run an internal rules engine, which compares the received facts with the previously entered rules by the administrator. If any rule matches any fact, the self-orchestrator module sends a reorchestration request to the HLO.

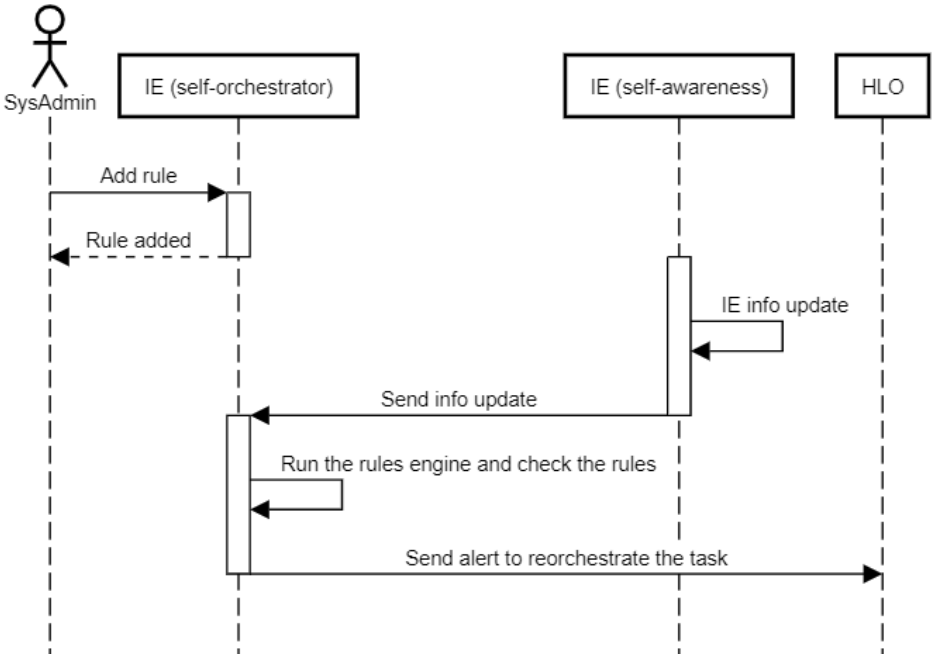


Figure 30. Service reorchestration triggered by the self-orchestrator module

Secure access to data within the continuum through the Management Portal, regardless its location.

Regarding the fifth main process mentioned for the Process View, it describes a sequence diagram that exemplifies how to access data available within the continuum through the Management Portal (e.g. the UI that displays the IEs of a domain with their current status), regardless of its location. This process begins with the login process in the Management Portal, which redirects the user to the Keycloak login page to let him introduce his credentials. These credentials are verified by the IAM, and if are correct, the user is authenticated, the portal stores the access token, and finally he is redirected to the welcome page of the portal. Then, the user accesses to a UI that needs to obtain data from the data fabric to build the page, so a request is sent to KrakenD with the previously obtained access token, which is checked by Keycloak to verify the user's permissions and, if authorised, KrakenD grants access to the requested information. Finally, the requested data is sent to the portal, which displays the requested page to the user.

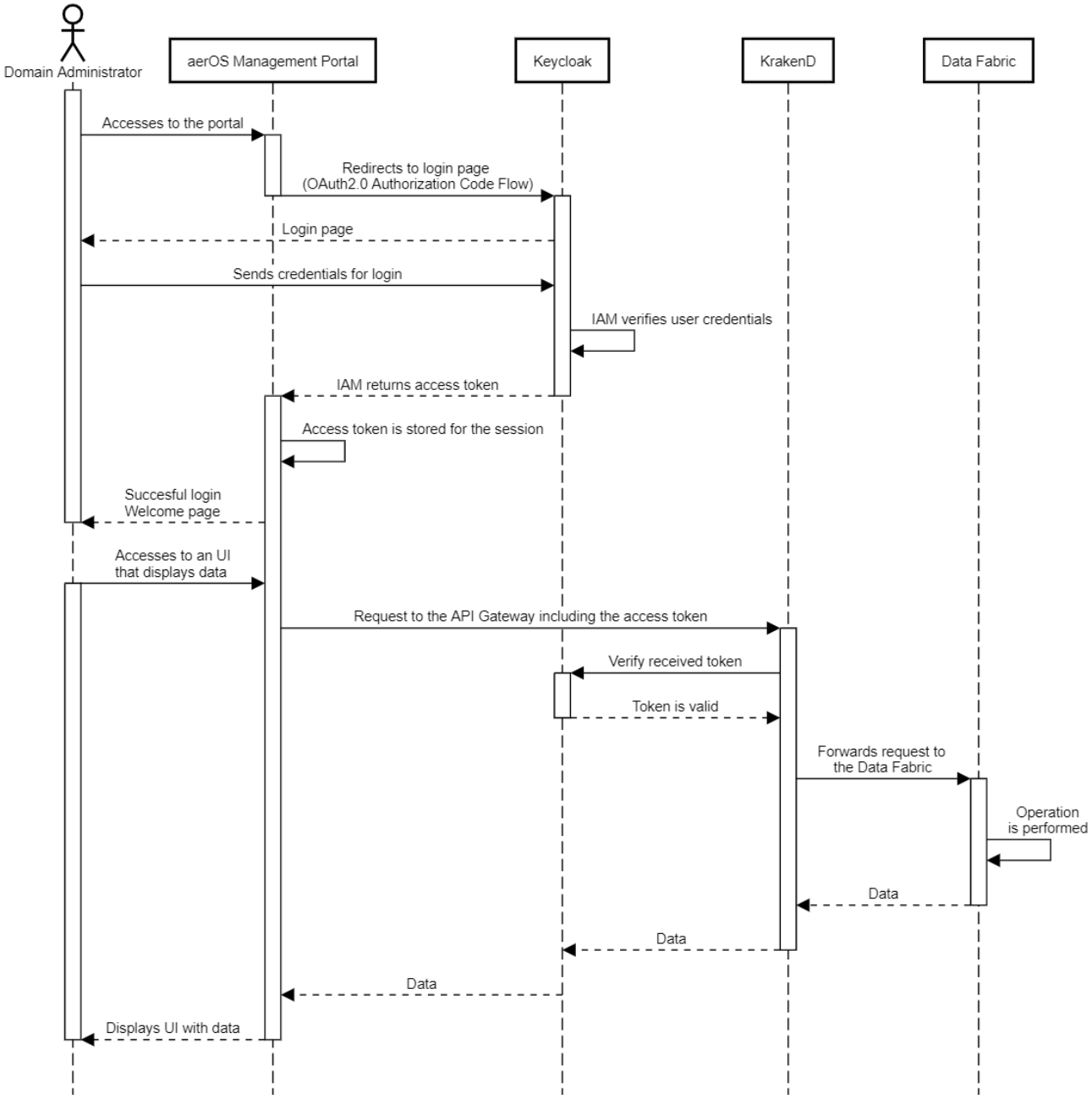


Figure 31. Secure access to data within the continuum through the Management Portal

IoT Service (sensor monitoring and actuation) materialization.

In the next process, an IoT service that includes monitoring and actuation wishes to be deployed via aerOS. This is a flow that was already demonstrated in the mid-term review in Brussels in April 2024 (see Section 0). Here, assumed situation (pre-requisites) are that: (i) aerOS has been installed successfully in (at least) two domains and those are functional, (ii) an IoT sensor exists, and is directly connected to a functional IE (see section 5.4.1 – IoT sensors), (iii) the portal is functional, and a user can execute the orchestration procedure (see above). In the process, a developer delivers an IoT Service code, which has been built considering the selection and usage of a Smart Data Model fitting the IoT data source. Once this component is properly packaged and made available to the user in the portal (DevPrivSecOps methodology), the user -via the portal- manipulates the form to commission the deployment. Here, the manual mode is selected, due to the necessity of direct connection IoT Service <-> Data element (sensor). Once the service is deployed (as it contains the proper software), the data updates are made via context update in the Context Broker (Orion-LD) that lives in the IE acting as head of

domain in the same domain of the selected IE (the one connected to the sensor). This process keeps alive as long as the IoT Service stands. As the intention of the whole scenario is to perform certain actuation whenever necessary, an Actuation Service is developed, packaged, and made available to the user in the portal. Then, the user -via the portal- decides to deploy the service. This time, as it is not compulsory for the service to live in a specific IE, the semi-automatic (or automatic) orchestration mode is selected. At this point, the Actuation service ends up running in an IE that can either belong to the same domain as the former, or not. For the sake of functionality illustration, the diagram in Figure 30. IoT Service (monitoring and actuation) deployment and functioning assumes that it runs in a different domain. This service is subscribed to the corresponding entities in its local Context Broker, which sits on Head of Domain 2 (this is a relevant point). For this to properly function, the Data Fabric mechanisms should have previously guaranteed that the federation (in subscription to specific entities) has already been achieved. Whenever the actuation target/threshold is reached, the logic of the service will perform certain operations, and will initiate the actuation via updating a selected field in the corresponding entity (details will depend on the specific service and data model design). Once this occurs, the remote Head of Domain 1 (which is subscribed to changes thanks to the aerOS Federation) is notified and proceeds to forward the notification to the exposed endpoint in the IE connected to the sensor. Here, the logic embedded in the IoT Service will interpret the actuation and will directly act upon the sensor, completing the lifecycle of the scenario.

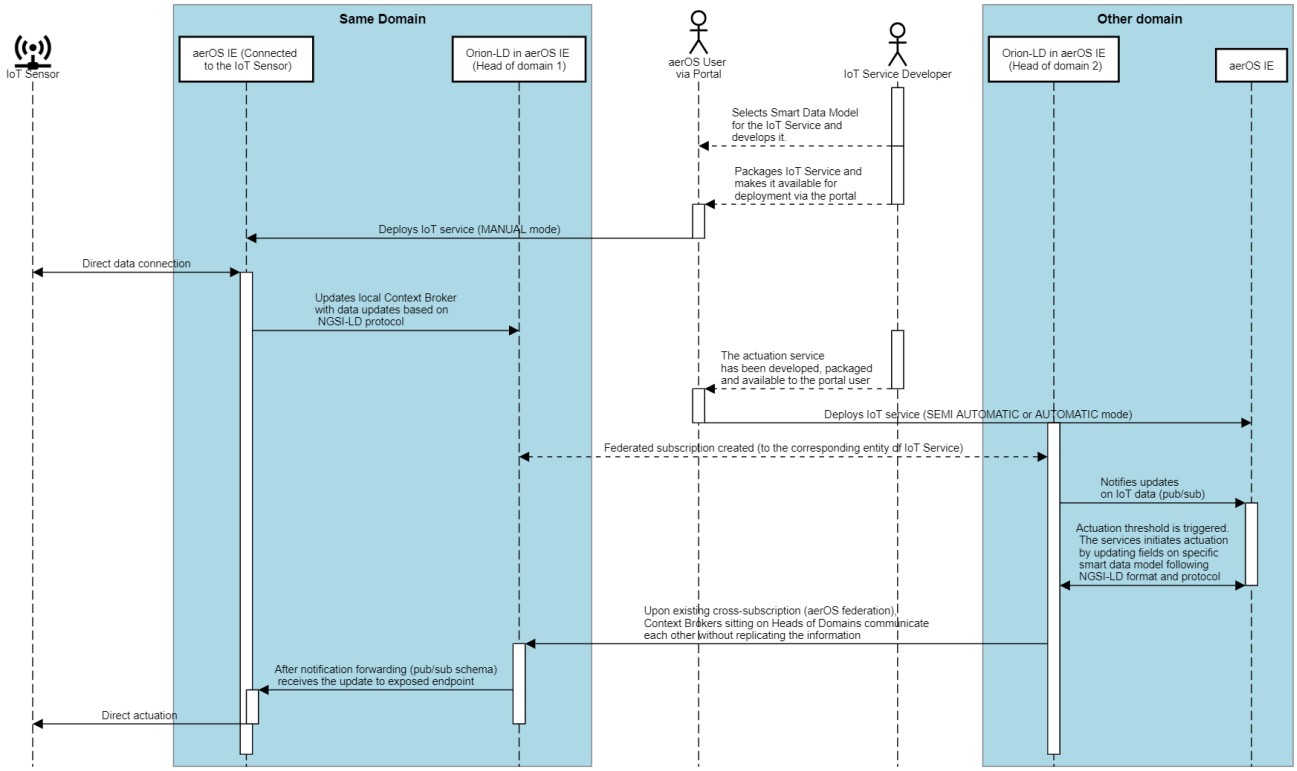


Figure 32. IoT Service (monitoring and actuation) deployment and functioning

Trustworthy exchange of immutable and irrepudiable messages across the continuum.

In the next process, a pilot user wishes to upload a new block into the Tangle with a message that needs to be immutable. It starts by making a petition to the IEs Hornet Node with the payload it wishes to upload, the block is built and sent to the node's neighbours, eventually making its way into the Coordinator. The Coordinator now validates and verifies the received block, if invalid it is discarded, otherwise it is stored and a reference to the block is attached to the most recent milestone. This milestone will confirm to all nodes the inclusion of all blocks attached to it into the Tangle itself. After this the Pilot can verify the inclusion of the uploaded block by retrieving the metadata of it, which will confirm it being referenced by a milestone.

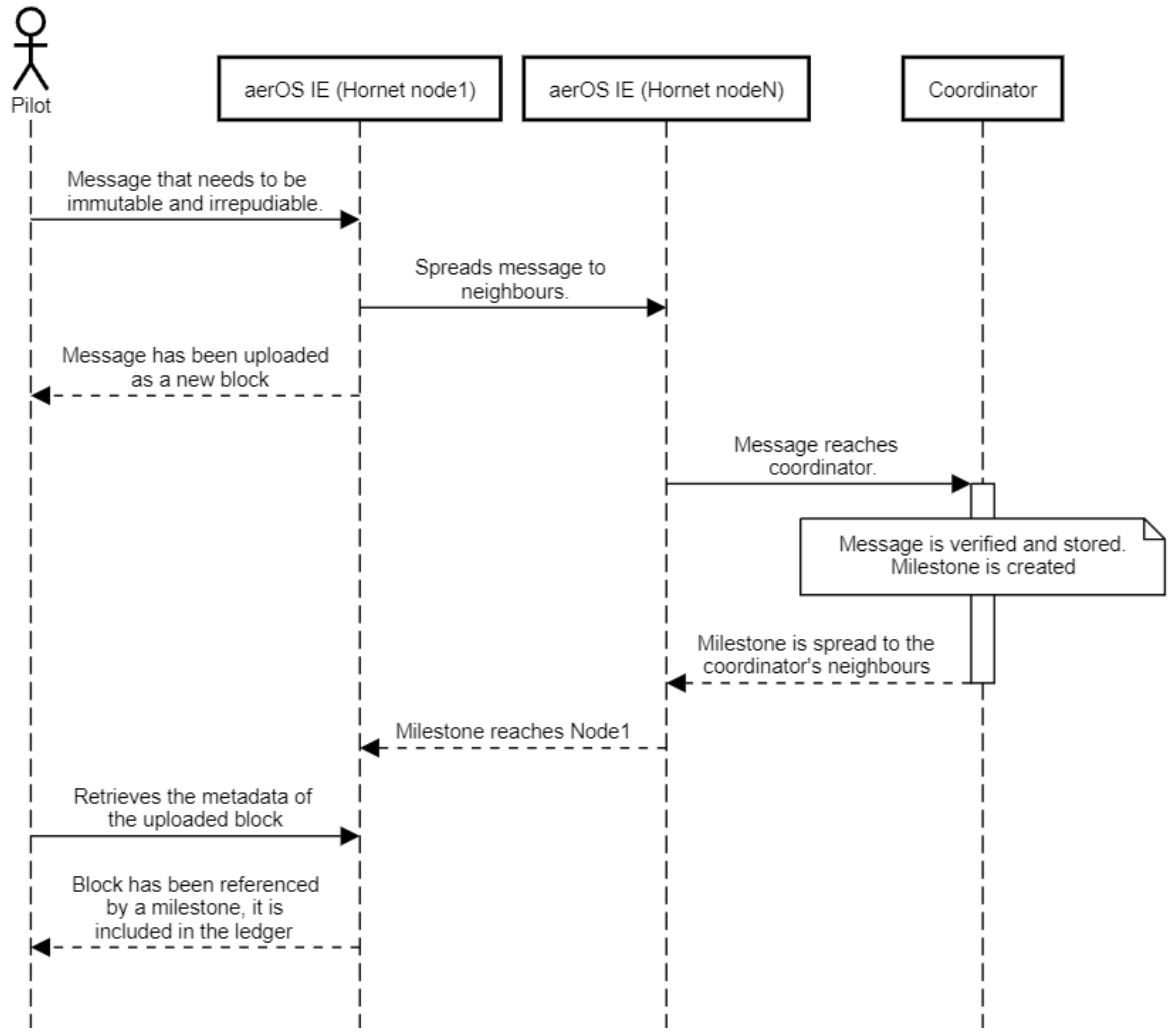


Figure 33. Trustworthy exchange of immutable and irrepudiable messages across the continuum

Decentralized AI task coordination and execution.

Figure 32 presents a sequence of steps required to run a decentralized AI task, specifically federated learning. First, aux AI services need to be deployed on the aerOS infrastructure. This is done with an *Intention Blueprint* that is declared and after processing, it is used by aerOS Orchestrator to select IEs on which services should be deployed. To run a decentralized AI task, one AI Task Controller service (with three components) and at least one AI Local Executor service need to be run. AI Local Executor services are responsible for running local sub-tasks of an AI task that's overall execution is controlled by AI Task Controller.

A specific task execution is triggered by providing configuration to the AI Task Controller (using service API). The required configuration is passed to AI Local Executors and task is initiated. AI Local Executors run rounds of training locally and send parameters update to AI Task Controller for aggregation. AI Task Controller checks if training is finished and if yes then a final model is stored and can be retrieved.

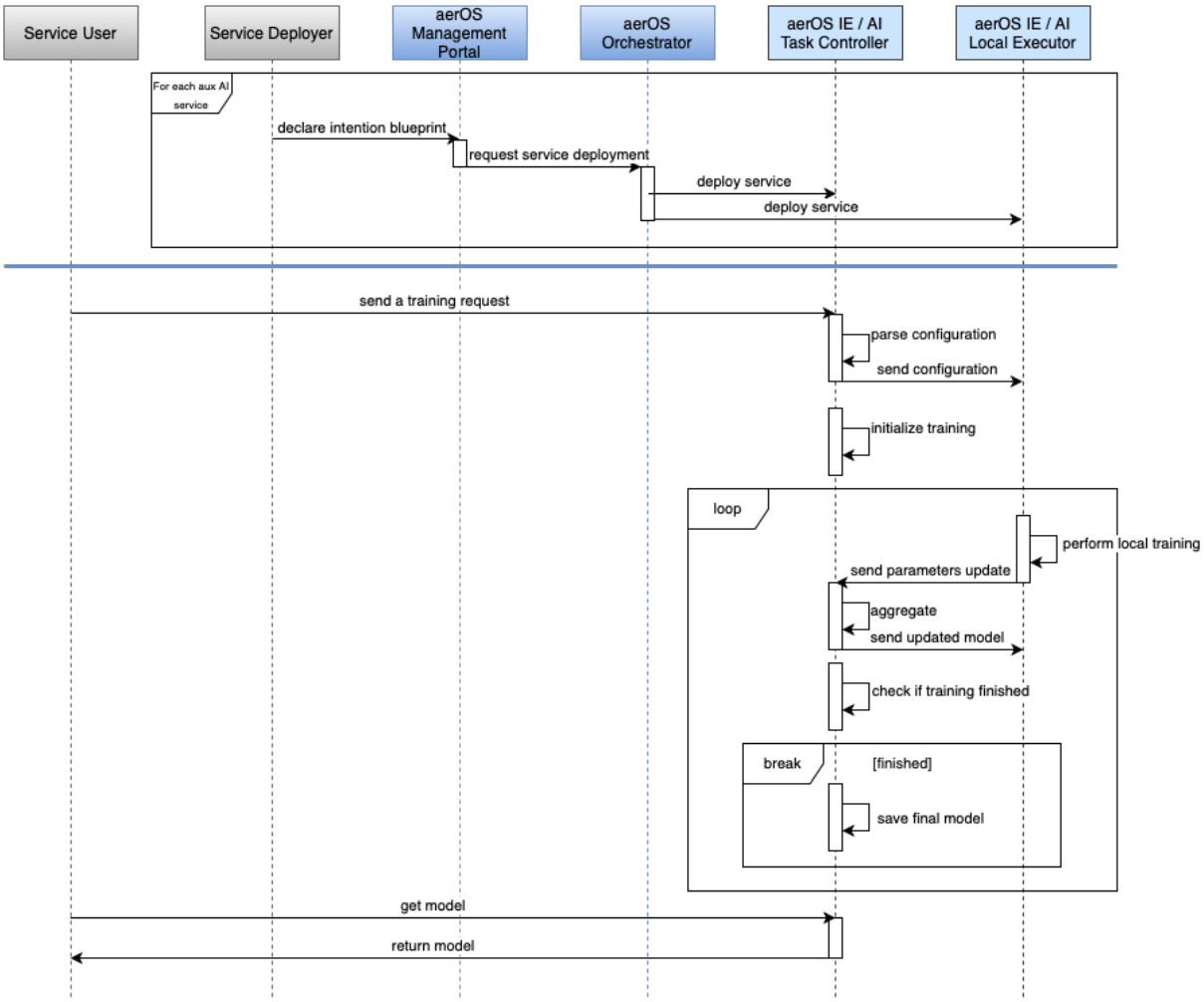


Figure 34. Decentralized AI task coordination and execution process

6.4. Data view

An aerOS domain may include multiple IEs, resulting in new data sources and data consumers dynamically becoming part of the continuum. To cope with this changing data landscape, while ensuring data governance policies are properly met, the aerOS Data Fabric of the aerOS domain provides data owners with a set of tools as introduced in Section 5.5.2.

The incorporation of new data sources and, therefore, the registration of new data products available within an aerOS domain is the responsibility the owners of the data. To this end, data owners must interact with the Data Product Manager of aerOS Data Fabric to onboard new data products, triggering the workflow depicted in **¡Error! No se encuentra el origen de la referencia.**. The following steps take place during the workflow:

- **Step 1:** The data product owner onboards the new data product via the REST interface exposed by the Data Product Manager. In this process, the data owner provides the metadata and artifacts that comprise a data product.
- **Step 2-3:** Based on the metadata and artifacts provided in the previous step, the Data Product Manager orchestration the Computing Resources of the IE (e.g., Kubernetes) to deploy a data product pipeline.
- **Steps 4-7:** Once the pipeline has been deployed, the data product is built and stored in NGSI-LD Context Broker, becoming part of the knowledge graph. The URL from which the data product can be accessed, in addition to the governance metadata provided by the data owner, are sent to the Data Catalogue component. The Data Catalogue process these metadata and integrates them into the knowledge graph.

- **Steps 8-11:** If the data product included a predefined access control policy, the policy information is sent to the Data Security component, which accordingly configures the Access Control component (e.g., Keycloak).
- **Step 12:** The data product owner is notified that the data product is now available in the aerOS Data Fabric.
- **After Step 12: At this point, data consumers can interact with the NGSI-LD Context Broker to discover in the knowledge graph the data product that has been created and access the containing data.**

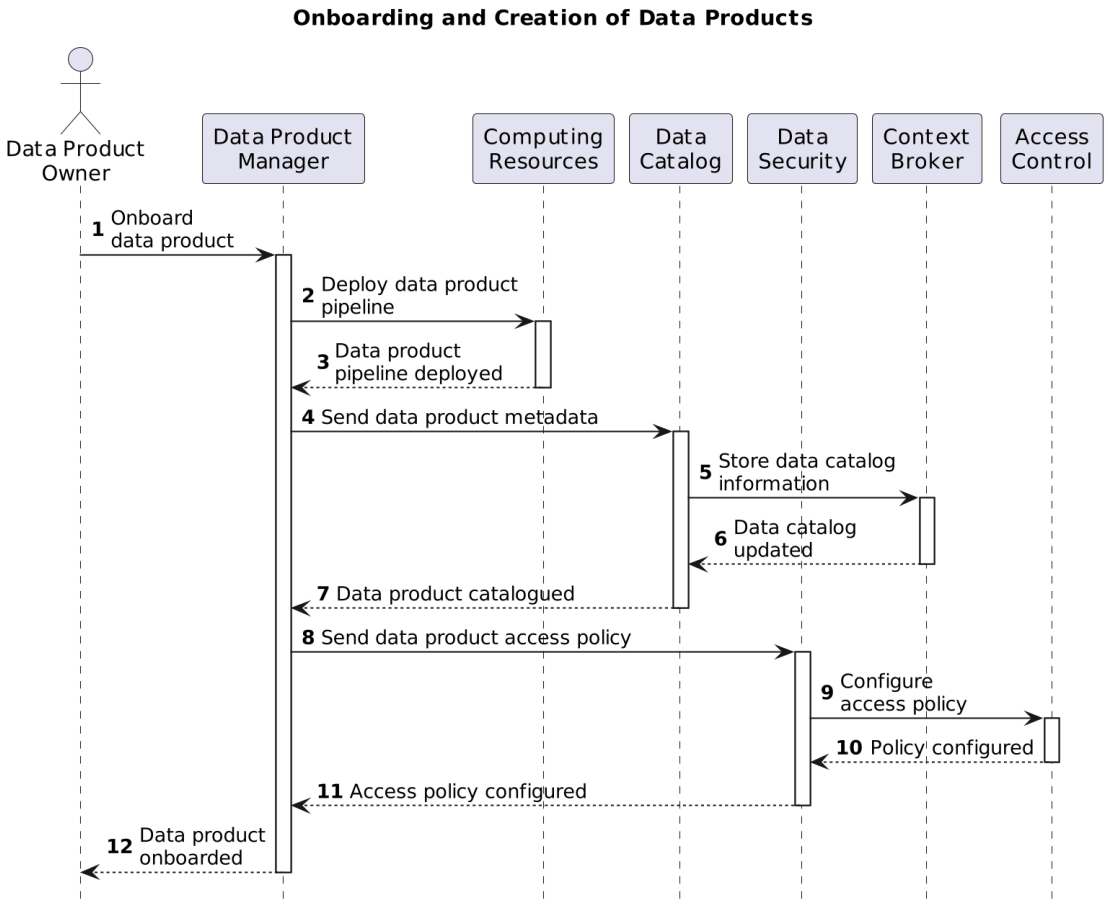


Figure 35. Workflow during onboarding and creation of data products in the aerOS Data Fabric.

However, as noted before, the IoT-Edge-Cloud continuum presents a highly dynamic data landscape, where new aerOS domains with one or several IEs might join or leave the continuum at any time. To enable the exchange of data flows among different IEs and different aerOS domains, aerOS implements a federated architecture comprising multiple Data Fabric instances, as depicted in **¡Error! No se encuentra el origen de la referencia.**.

In this architecture, the Context Brokers existing in an aerOS domain store data about the state of the IEs and what kind of data is available in them. They are also aware of the rest of Context Brokers in their domain and elsewhere as well (federation). IEs either implement Context Broker or Context Providers, depending on their computational resources and the spot in the continuum. Every time a data owner builds a new data product from a data providing domain, the local Context Broker registers the data product and informs the Context Brokers from other aerOS domains to update their Context Registries with this new data product.

As a result, when a data consuming domain from aerOS domain A requests a data product served by aerOS domain B, the local Context Broker knows the neighbour Context Broker from which the data product can be

retrieved. Based on this, the local Context Broker A interacts with the neighbour Context Broker and obtains the data product on behalf of the consumer.

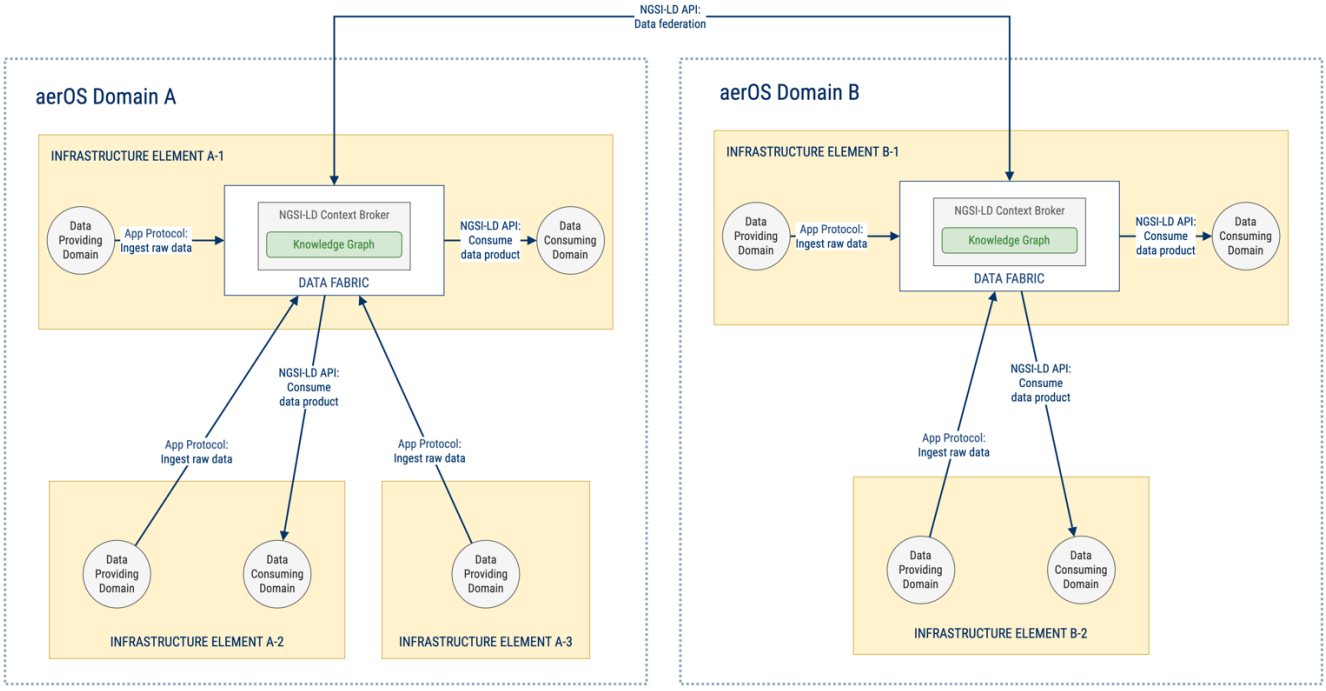


Figure 36. Federated architecture of the aerOS Data Fabric.

6.5. Deployment view

Deployment view refers to the process of positioning containerised software components on top of the physical layer and establishing the necessary topology needed to make aerOS systems and applications available and operational in industry verticals use cases. It presents the system from an engineer's point of view while deploying, positioning, configuring, and interconnecting all needed software components, on the physical layer, needed to ensure that aerOS capabilities are accessible and ready to operationally serve stakeholders according to their specified and designed intentions.

Although the goal and overall deployment process remains in line with what was described in first version, two activities that were initiated after the first version release, provided valuable input based on which the deployment view has been defined in more detail. The first activity was the deployment of the MVP, described in section 7.1.1, and the other source of feedback was the discussions curried out with WP5 regarding the scoped instantiation of aerOS reference architecture within project pilots.

The goal of aerOS deployment is to enable resource owners to register their assets as part of the aerOS continuum. On one hand this means to deploy aerOS stack, over available hardware, and render them to IEs collectively hosting, and supported by a common set of aerOS services and thus exposed and orchestrated as an aerOS domain. On the other hand, this means to automate the integration of any new registered domains with already enrolled resources and thus set them able to discover resources, services and data which could support their purposes. The overall process is fully documented and supported by both a deployment toolset and management portal registration activities.

Initially the process requires that stakeholders, resource owners, register themselves within the aerOS continuum. This provides the required identity for them to be able to expose and supervise their resources. Then the design provides a one stop process towards rendering available computing and network resources to an aerOS domain. This process is automated with the use of, aerOS provided, tools which undertake the deployment of all that is needed to expose enrolled computing resources as IEs and on top of them to deploy all aerOS services which unify them under the umbrella of an aerOS domain. This includes the deployment of self-* packages and LLOs to target the variety of integrated computing resources as IEs and then all the services'

suite (section 5.5) which will unify, and transparently manage and orchestrate integrated resources as one administrative entity, i.e. an aerOS domain. Once this process is completed, the new domain will host all aerOS basic -and selected auxiliary- services and this means that it will be able to securely provide selected resources and consume resources hosted in other domains. Having transformed all legacy hardware into aerOS IEs, the aerOS runtime (section 5.4), acts as an abstraction layer on top of them and seamlessly provides the basis for aerOS services execution which in turn provide connectivity, orchestration, Data Fabric integration and interface to all other integrated domains. Figure 1Figure 35 provides an example of aerOS deployment over stakeholder resources on top of, most probably, heterogeneous hardware and operating systems. Guided, and tools-supported, aerOS runtime deployment will abstract underlying diversity and will provide a common runtime, on top of which aerOS basic and auxiliary services are running and transparently taking care of all underlying resources -like a legacy OS would do- providing a common and federated execution environment for IoT developers to deploy their applications. These applications will now receive, from aerOS service fabric, all Life Cycle Management (LCM) support needed to run smoothly over a secure environment and take advantage of federated capabilities.

This whole process, as explained, is designed to be supported and automated by artifacts publicly available to interested parties. These artifacts are packages tackling all the above activities, and container images, built for a variety of architectures, needed for the deployments. In fact, these packages are modular components, requiring minimal configuration, which address a) IE rendering for each integrated computing element, b) basic services deployment for domains setup and c) domain's automated registration in the continuum. Then the aerOS stack deployment and registration is complete and stakeholders can further control and modify access to resources and proceed to services deployment on top of these resources or on top of a federated computing underlay partly composed of their resources and partly from other resources which aerOS Meta-OS might transparently provide as more suitable candidates.

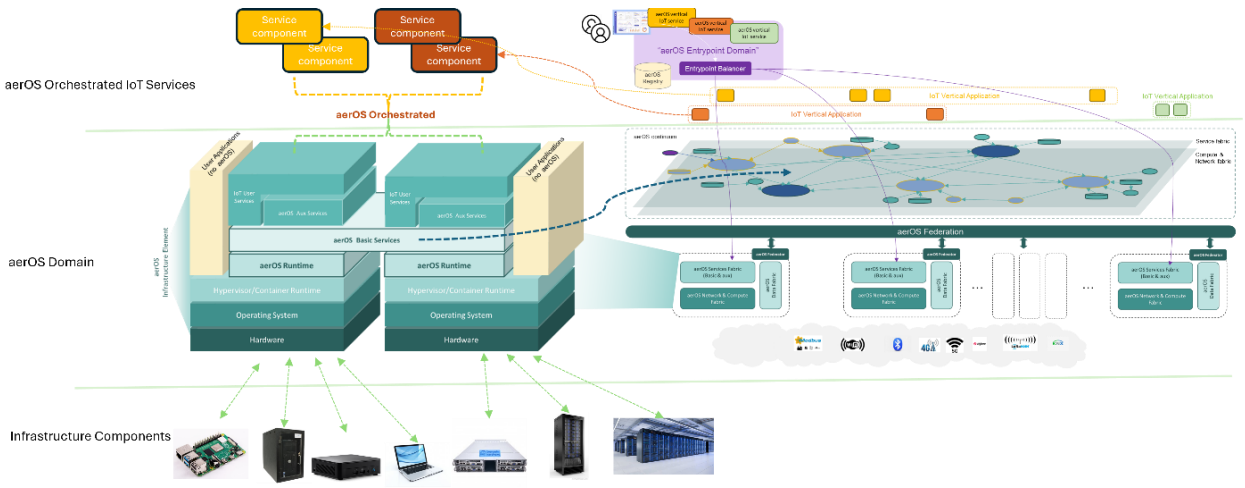


Figure 37. Deploying an aerOS domain and rendering resources as part of the continuum

It is obvious that aerOS stack deployment on top of existing resources and subsequent registration in the aerOS continuum is a smooth and straightforward process. What is yet important to explain is that aerOS, as an open system which does not enforce any kind of hierarchy or dependencies, with the same flexibility offers the possibility, and the means to initiate and set up the whole Meta-OS from scratch. If a collaboration or federation or any other kind of interested party would like to build their stand-alone ecosystem, they should be able to do so. The only additional thing they should address is the Entrypoint domain set-up, as a first step and before anything else. Entrypoint domain requires some more components to be deployed additionally to what is already described for an aerOS domain. These include the AAA subsystem and the management framework, which mainly integrates the previously mentioned portal. But this should not be any difficult as the modular design foresees the "Entrypoint domain package" which will take care of this. From there on, for the subsequent resources' integration in the continuum under aerOS Meta-OS, the procedure is the same as described above.

The following table depicts the basic components expected to be deployed within an aerOS continuum, which consists of N domains and X IEs. This list is not exhaustive but aims to provide a comprehensive picture of the continuum's components and their distribution. The overall concept is to correlate the multitude of components across the continuum, within each domain, and on top of each IE. Moreover, this helps to understand what is required for each unit to operate as an aerOS entity.

Table 2. Deployment Level of components in aerOS continuum

Component/service	In the whole continuum	In each domain	In each IE
aerOS Management Portal	Single Instance (1)	-	-
aerOS Federator	N (as many as domains)	1	-
HLO	N (as many as domains)	1	-
LLO	$\sum_{i=0}^{N} D^i$ Total number of LLOs in the continuum (N domains)	D (the multitude of IE container management framework types integrated in the domain)	1 associated LLO
Keycloak	Single Instance (1)	-	-
OpenLDAP	Single Instance (1)	-	-
KrakenD	N (as many as domains)	1	-
Data Fabric enabler (Orion-LD)	N (as many as domains)	1	
Self-*	X (one per IE)	X (one per IEs)	1

The table highlights the essential components that make up the aerOS continuum, illustrating how they are distributed across it within domains and IEs. Some of them have a single instance in the continuum, some need to have presence within each domain and some need to be hosted in each IE. Notice that for LLOs there might be small number of distinct types, but each of them should be replicated in each domain which integrates this type of IE.

Although the whole deployment view is quite close to what was already presented in first version, there are some updates that are related to careful decisions and feedback received during this period both from technical partners but from pilot engaged people too. The integration of LDAP which provides a more granular and efficient access control and the wish to make the whole procedure less divided among on-site deployments and via dashboard metadata and information provisions, instructed the creation of packages, as mentioned above, which only require the existence of an associated and registered aerOS user and then take over the rest that are needed for both deploying and registering. On the other hand, the decision that deployment process for aerOS does not include machines setup, e.g. OS or VMs provisioning, set a clear borderline as to what is a prerequisite and what is an aerOS deployment view. This moved away the initially declared possibility, in version 1, to integrate into the management dashboard tools and processes to perform this. Such an approach would introduce complexity and raise security concerns, potentially making the system more susceptible to vulnerabilities and functional instability. Consequently, this could lead to increased reluctance among users to adopt the system.

6.6. Business view

The business view describes the functionality of the aerOS Meta-OS from the perspective of external actors (e.g., end-users not directly involved on aerOS related administrative actions). Furthermore, this view contributes to the development of a suitable business model for the aerOS exploitation context and helps on understanding the main activities and interconnections among the aerOS services. Consequently, this view

strongly relies on the previously introduced ones, as it relates the concepts described in them. Figure 32 presents the type of this relationship:

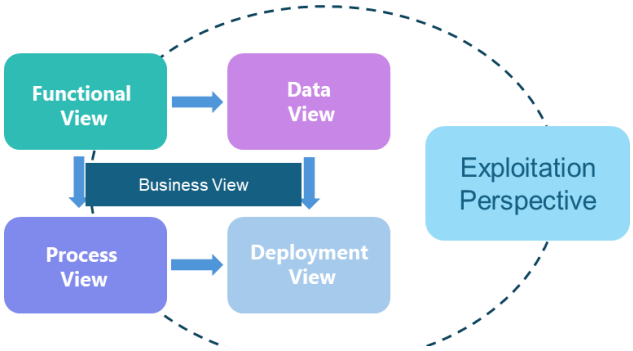


Figure 38.Business view interactions

According to the first design of this view (available in "D2.6 aerOS architecture definition (1)") External actors in the aerOS context mainly include (i) **end-users which may develop services** built on top of the aerOS infrastructure and (ii) **end users which consume aerOS services**. The main difference between them is that while the developer has control over some physical components of aerOS (i.e., aerOS is seen as a Platform-as-a-Service (PaaS)), the consumers only make use of aerOS services and produced data (i.e., aerOS is seen as a Software-as-a-Service - SaaS).

In "D2.1 State-of-the-Art and market analysis report", a comprehensive survey among several stakeholders that form part of these potential end-users was conducted and published. The top-ten challenges envisioned for a platform such as aerOS shall be addressed were in that priority order: (i) Integration complexity, (ii) Data collection & Analytics processes, (iii) Privacy, (iv) Securing network, devices or data, (v) Scalability, (vi) Cost of maintenance and management, (vii) Connectivity of devices edge-to-cloud, (viii) End-to-end IoT solution monitoring, (ix) Vendor Lock-In, and (x) Regulatory and safety certification.

However, in this document, reflections from the reality of aerOS architecture design and adoption have driven the team to include the relevant perspective of **exploitation route**.

aerOS has been conceived as a modular system where several components (basic and auxiliary services) can be selected, always running on top of the aerOS runtime (see 5.4). The decisions to be taken about which components to select, or where to deploy them upon will differ based on the **exploitation viewpoint** of the adopter entity.

Therefore, a specific flow can be envisioned as follows:

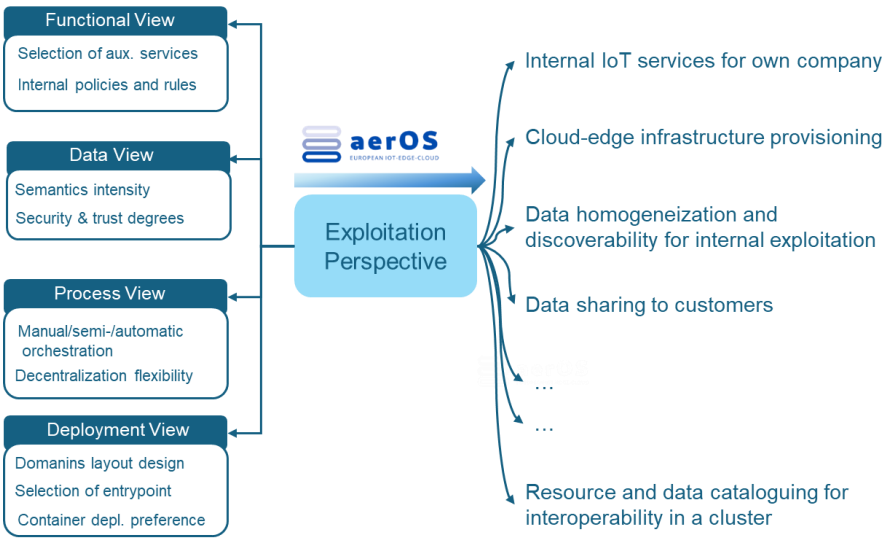


Figure 39.Exploitation perspective intertwining aerOS architecture views and decisions

As it is illustrated above, different exploitation routes directly affect the decisions taken across all aerOS architecture views. While the functional, process, data and deployment view follow the principles explained in 6.2, 6.3 and 6.4 sections correspondingly, the way they will be approached/focused may vary.

This way, the exploitation perspective acquires a relevant dimension Meta-OS-wise.

If a company aims at exploiting aerOS for increasing their internal IoT services capacity (more sensing, actuation, real-time monitoring of real life enterprise "things"…), they might select auxiliary services such as self-configuration, self-healing, advanced networking component, or other rather than powerful serverless cloud capacity, semantic annotation and translation, or , among others.

Similarly, cloud providers would opt for cloud-native container management frameworks (i.e., K8s-based), thus aligning their deployment options to those such as self-scaling, unified low-level orchestration, higher centralization of services, etc., while still keeping aerOS continuum traits.

Other exploitation cases might pivot around data sharing, either for internal interoperability or for potential commercialization of the information. There, Data Fabric additional tools (LOT-based results, semantics expression…) and relevant annotation and translation tools will come handy. Also, trustworthiness components provided by aerOS might be given preference, and might be installed in more computationally powerful nodes, etc.

Many examples could be found, and will need to be analysed carefully by the adopting entity how to approach the fine-tuning of aerOS uptaking based on their own exploitation path.

Once the exploitation viewpoint has been settled to drive the decisions on how to follow aerOS architecture adoption and materialisation, in the following sections, a short explanation on how aerOS and its corresponding services **map the business perspective of the adopting entity** is presented:

Scalability: Scalability is one of the main design drivers in aerOS. The core service providing this feature is the Network & Compute Fabric (see section 5.5.1), which allows expanding aerOS infrastructure seamlessly, without sacrificing performance, as new servers, devices or workloads are introduced in the aerOS ecosystem.

Cost of maintenance and management: This challenge has not yet been prioritised, although continuously borne in mind, considering the initial stages of the project, without a clear business model identified yet. aerOS will tackle this challenge more thoroughly in future activities related to exploitation. In the next architecture definition deliverable further details will be provided.

Connectivity of devices edge-to-cloud: aerOS envisions the connectivity of devices as set of services running on specific IEs, which provide physical access/connectivity to the device (e.g., a device connected using a Bluetooth connection). aerOS will facilitate the establishment and maintainability of such connectivity through the self-* features (described in deliverable "D3.1 Initial distributed compute infrastructure specification and implementation") and the network & compute fabric (Section 5.5.1), potentially integrating technologies such as Liqo and Netmaker to allow resource sharing (i.e., access to a physical device) between IEs and domains.

End-to-end IoT solution monitoring: aerOS will provide and end-to-end monitoring. To do so, multiple metrics from the different aerOS core and auxiliary services will be gathered and available to either just visualisation purposes, or for platform malfunctioning alerts, through the embedded analytics service (Section 5.6.2). Furthermore, tools and guidelines for also allowing user to publish customized metrics from their own user applications will be provided. These monitoring metrics will be as well propagated to the rest of IEs in the domain.

Vendor Lock-in: aerOS is a Meta–OS, which provides a set of open-source based core services. Therefore, aerOS does not impose any vendor lock-in on the deployment infrastructure, meaning that the running host system (either a hardware device or a virtualized one) can be selected by the final user. This does not preclude that any service/application designed on top of aerOS can impose some "vendor lock-in" in the future.

Regulatory and safety certification: Following the preliminary analysis carried out in deliverable "D2.1 State-of-the-Art and market analysis report", all aerOS components are GDPR compliant by default. Furthermore, they will also address specific European and national regulations regarding data privacy.

7. aerOS Reference Architecture instantiations and evaluation

Within this section the practical applicability and partial validation of the aerOS architecture design is demonstrated. The deployment of the architecture in real-world scenarios, ensuring that the theoretical concepts are translated into tangible implementations is highlighted. , The foundational capabilities and functionalities of the aerOS architecture in a controlled environment are showcased by detailing the development of a Minimum Viable Product (MVP) based demonstrator. Further, it is illustrated how the aerOS Reference Architecture maps to various project pilots, providing concrete references of pilots' functionalities and concerned aerOS concepts. Finally, it is described how the aerOS Reference Architecture is aligned with the EUCEI continuum (see 7.3, highlighting the commitment in aerOS to meet European standards and contribute to wider technological frameworks). These activities collectively underscore the robustness, versatility, and relevance of the aerOS architecture, reinforcing its potential to drive innovation and efficiency over multiple domains related to European Meta-OS vision.

7.1. aerOS demonstrator development

With a focus on the instantiation and validation of the aerOS reference architecture, a demonstrator was designed and executed to serve as a practical proof of concept, showcasing the entire deployment flow of an IoT application supported by the aerOS Meta-OS. This demonstrator is a simulated real use case that was run on the aerOS instantiation that offers the MVP., which indeed is a continuously running deployment and validation space for aerOS concepts and design. The demonstrator, making use of many aerOS components, illustrates the continuum setup and orchestration capabilities of aerOS. Not only validates it the core functionalities of the aerOS architecture, but also highlights its potential in real-world IoT applications, demonstrating seamless integration and effective orchestration of diverse components within the IoT-Edge-Cloud continuum.

7.1.1. aerOS MVP

aerOS architecture integrated and performed research and implementation beyond current state of the art in the fields of compute and network fabric, service fabric and data fabric. These activities introduced development and integration complexities, which are reflected in the great variety of technologies and tools that must coexist, cooperate, and seamlessly provide Meta-OS functionalities for the IoT-Edge-Cloud continuum (see 7.3). aerOS MVP was deployed to serve as a practical paradigm where all complexities could be addressed, and concepts validated.

The aerOS MVP that hosted the demonstrator, an IoT application detailed in the next section, integrated a topology that initially included two registered aerOS domains. Later, during the demonstrator's runtime, an additional plug-and-play mobile domain was registered to showcase the flexible extendibility of the aerOS continuum and service migration. The first two domains were constantly acting as the MVP baseline, providing a permanent integration and validation environment for the aerOS architecture.

The goal was an initial implementation of all the capabilities that aerOS can offer. Thus, all basic aerOS services and some auxiliaries too were integrated in the two domains which permanently support MVP purposes, and later these were also deployed in the, ad-hoc integrated, mobile domain. The Entrypoint domain (see section5.3.2) is deployed in CloudFerro, a commercial cloud provider and project partner, premises and the "plain" domain resides in the premises of the Technical Coordinator - NCSRD. In addition, the mobile domain consisted of a legacy laptop and a Raspberry Pi that was brought physically to the review room in Brussels to support the heterogeneity of computing resources and execution environments that can hanlde the aerOS Meta-OS. All domains were exposing their access endpoints in the public internet and each of them owned a FQDN registry under the domain *aeros-project.eu*.

The Entrypoint domain, located in a public cloud K8s cluster, integrated all management aerOS services. These services, which have a singleton presence in the continuum, include the aerOS federator, management portal, and AAA components such as IdM and LDAP. These components provide a single point of GUI-supported entry for visualized management of the aerOS continuum and based on the implemented user and role-based

access control registry, serve as a single point of truth for authentication and authorization permissions for all resource access.

Beyond these Entrypoint-specific services, which are just deployed in one domain, all other services, described in section 5.5, were deployed in all domains, thereby exposing APIs for resources federation and orchestration on top of the underlying integrated IEs. Local agents for security enforcement were also deployed. Without going into detail, it can be stated that all networking capabilities, resources' monitoring, federation, and orchestration capabilities were integrated in this demo. As a result, the domains could interact among them and were accordingly represented in detail within aerOS dashboard. A short visual representation of the MVP as shaped for the demonstrator use case is presented in Figure 36Figure 37.

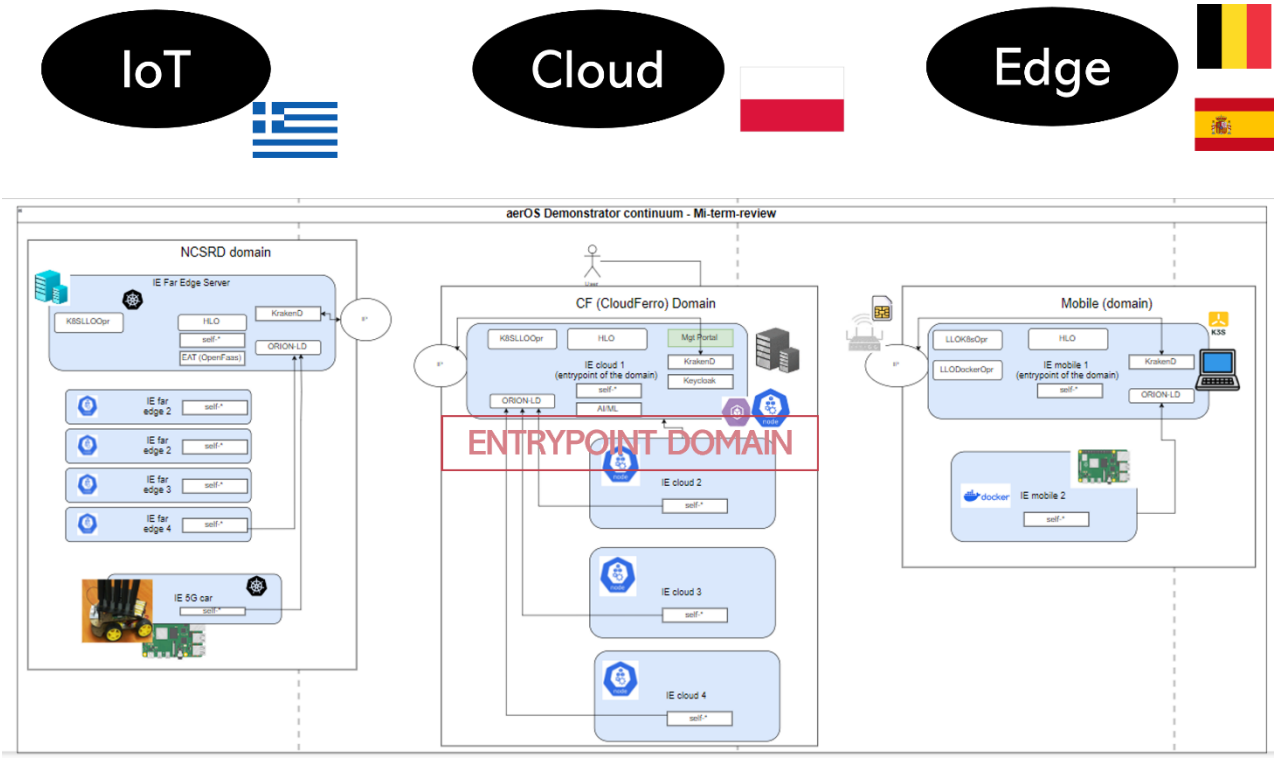


Figure 40. MVP topology

Three domains were part of the aerOS orchestrated continuum, one in Poland which acted as the Entrypoint, one in Greece and one mobile coming from Spain and functionally integrated in Belgium. While the domain hosted in CloudFerro has an essence of cloud, the domain in NCSRD is used as a far-edge IoT domain hosting the "things" and the mobile domain is another domain interested to host applications which can interact with the IoT devices in the IoT domain. All integrated IEs in each domain hosted self-* package to report their state, a federated Orion-LD CB were hosted in each domain to share IEs statues in real time, networking components (in the form of CNFs) enforced secure and private exposure are deployed in each domain, HLO and LLO to support decisions and enforce onto the IEs are deployed as well and all resources and functionalities are exposed based on aerOS (Open)API . More aerOS components were deployed but it should be clear that the overall topology is based on the aerOS architecture blueprint.

The kind of computational resources integrated in the MVP for the demonstrator use case, as well as the connectivity networks expose some variations. This supports the validation of a Meta-OS requirement to be able to run on top of and orchestrate diverse resources. Entrypoint domain integrated a K8s cluster managed by a cloud provider (K8s as a Service), , while NCSRD based domain integrated both VMs and an RPi, which is translated into different underlying processor architectures. Additionally, beyond wired and virtual overlayed networking used for the VMs, the RPi based IE is connected over 5G network connectivity. This is enabled with the integration of a Waveshare 5G-HAT and under the control of SA (standalone) NCSRD hosted 5G core on

top of an Amarisoft Callbox and Open5gs equipment. Only the user (data) plane[10] was integrated as part of the aerOS continuum but the control plane could have also been orchestrated as part of aerOS continuum. Finally, the mobile domain included a laptop and an RPi model 2b. The public connectivity was provided over 4G commercial network with the use of an LTE router and resources were networked with the use of Wi-Fi providing access to the router's internal network.

This is the exact topology deployed for the demonstrator. It is partly integrating what is already there for the MVP and what the mobile domain brought in. Although the demonstrator use case has been run several times the description that follows is based on the demonstrator execution in Brussels for the midterm review of the project.

7.1.2. IoT application over the aerOS continuum

In the above section, it was described the topology deployed for the MVP instantiation based on the aerOS RA. Now, it is time to validate that the deployed solution can indeed operate as a Meta-OS, providing the functionalities necessary for establishing a continuum and the required mechanisms for orchestrating services across it from edge to cloud. To do so, a real application scenario has been designed and implemented with the support of aerOS.

Initially and before the IoT application deployment, aerOS administrator navigates within the aerOS Management Portal to visually query the topology and overview all integrated resources. So does a plain user to demonstrate that resource access control is strongly enforced as registered users can access integrated resources and proceed to deployment actions under a strict security schema and role-based set of privileges. The topology is displayed in a user-friendly way and information about resources capabilities are visualized.

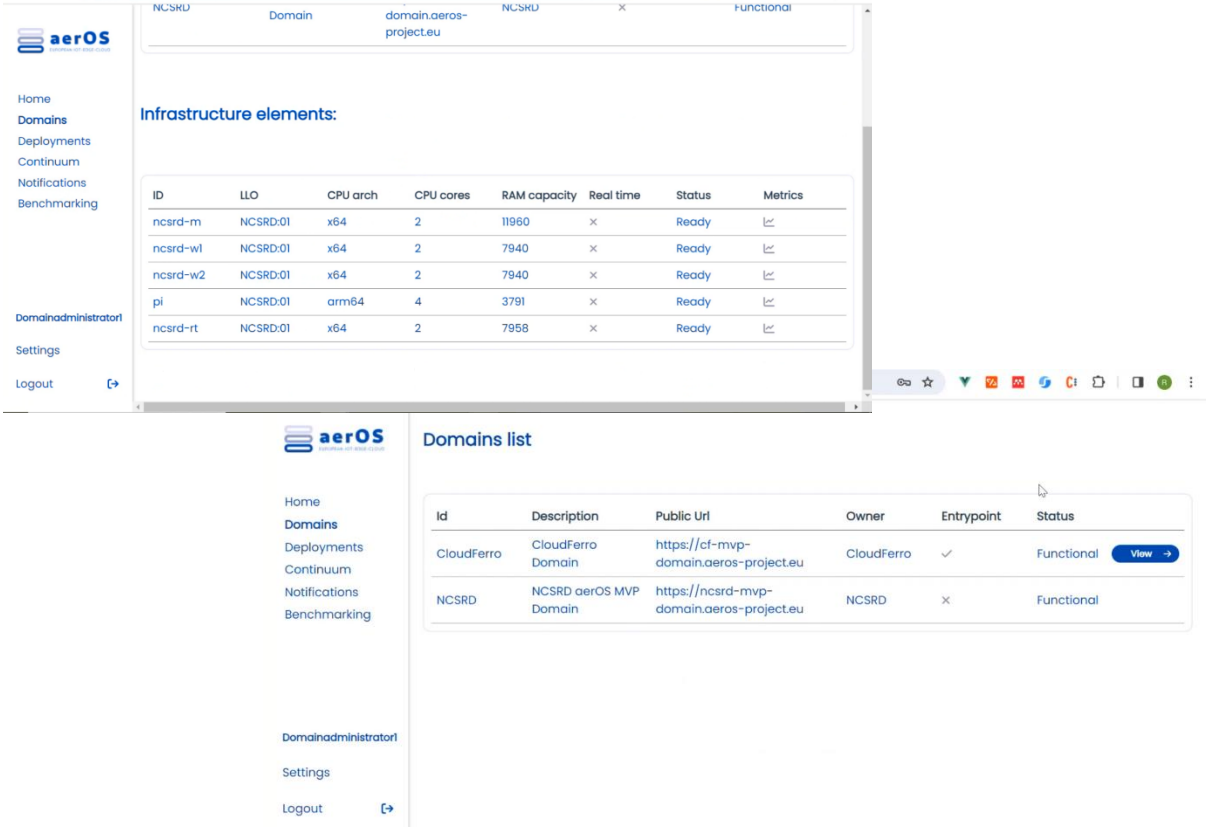


Figure 41. aerOS domains and IEs in management dashboard

Further, the actual scenario involves a user from the NCSRD domain requesting the deployment of an IoT application via the aerOS portal. The user specifies the application requirements, and the aerOS Meta-OS

[10] https://emblasoft.com/blog/exploring-the-3gpp-upf-user-plane-function

handles the rest. The IoT application integrates a 5G-car service component, built on top of the previously referenced 5G integrated IE, and a controlling unit as a separate service component. The 5G-car, in addition to the 5G-hat, includes an IoT sonar sensor for collision distance recognition. All data regarding 5G network QoS, 5G-car location, and collision distance are retrieved, modelled, and federated using aerOS Meta-OS components. The data modelling is based on the transportation FIWARE Smart Data Model, specifically the vehicle entity.

Figure 42. aerOS 5G-car IE

This vehicle entity not only exposes data but also provides a mechanism to receive navigation directives (start/stop/turn) through aerOS federation mechanisms (NGSI-LD Context Source Registrations) and subsequently report its conformance and updated current state. The navigation directives are part of the controlling service component unit of the application, which subscribes to vehicle data, updated at a per-second frequency, and processes this data to enforce navigation updates to the car using rule-based decisions or AI techniques. Additionally, a third service component builds on this exposed data and the current driving status to create comprehensive analytics dashboards, providing insights into the current location and status of the vehicle.

Thus, the IoT application will run on top of the 5G-car and simultaneously receive assisted navigation based on the federated metrics. This scenario extends beyond the deployment of two aerOS orchestrated services and requires a comprehensive application dashboard, thus employing the integration of aerOS analytics capabilities. While the service component of the IoT application, which retrieves the 5G-car status and exposes all run-time metrics, must be located on the aerOS 5G-enabled IE, all other service components can run anywhere in the continuum. aerOS takes control to select the most appropriate candidates and orchestrate them efficiently.

As a first step for the IoT application deployment, the user employs the dashboard procedure to compile a GUI-guided application deployment request. The user defines the number of service components: one is the 5G-car component, and the other is the navigation control component. Subsequently, the service component constraints are submitted for each one. For the former, the user specifies the exact IE that should host it (5G-car), but for the latter, only the IE requirements are provided, and aerOS takes over from there.

Upon deployment request aerOS, with the support of the Entrypoint balancer, see section 5.5.8, submits this request to the HLO exposed endpoints of the selected domain. In this case the local domain's HLO was selected and based on the service components constraints and the IE availabilities and capabilities across the continuum the HLO allocates the most suitable resources. This means that for the 5G-car service component the NCSRD

located IE is selected. Accordingly, the NCSRD HLO allocation engine endpoints (described in "D3.2 Intermediate distributed compute infrastructure implementation"), are accessed (a cross domain HLO communication), and the process proceeds until the LLO, responsible for the RPi IE, undertakes the service component deployment. For the other service component aerOS proceeds to querying the IEs available across the continuum, run a double optimization process and finally allocates the navigation control service component on top of an IE located in the entrypoint domain.

For the deployment of workloads, the internal container registry of the project's Gilab (see section 3.1.2 of "D5.2 Integration, evaluation plan and KPIs definition (2)")is used. Application specific images are retrieved from there. Finally, the whole deployment process is monitored, and all APIs access and information exchange are validated, with the use of K8s monitoring tools. K9s is the monitoring tool used, and this provides the capability to watch all the information and requests propagation along all the aerOS fabric components deployed in each domain.

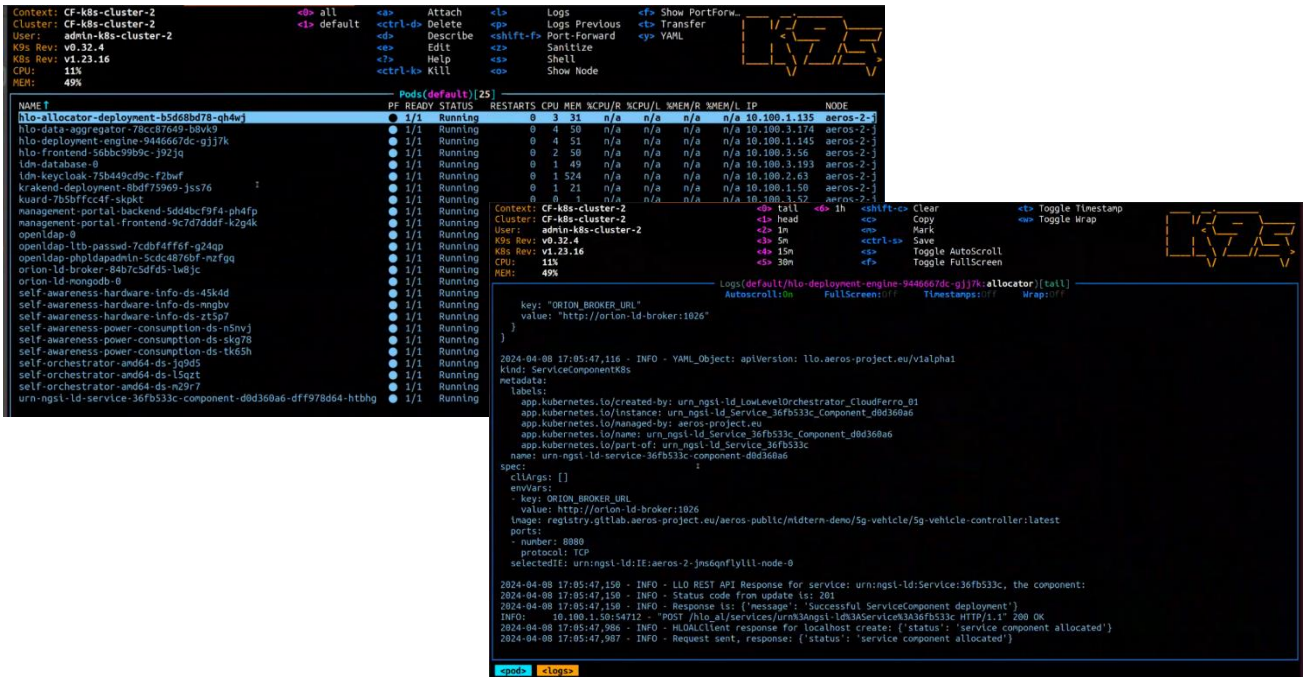


Figure 43. Monitoring IoT service deployment process with K9s

Once the two components are in a running state the 5G-car component publishes its state and receives navigation commands while the counterpart service component consumes this information, processes and feeds back any navigation commands. Additionally, one of the aerOS EAT components is integrated to provide visual insights. The result is depicted in the next figures where the continuum deployment status is presented in the dashboard, and the application data are presented in the Grafana (as part of the EAT) dashboard. For the last part an automated configuration is provided for Grafana dashboard creation.

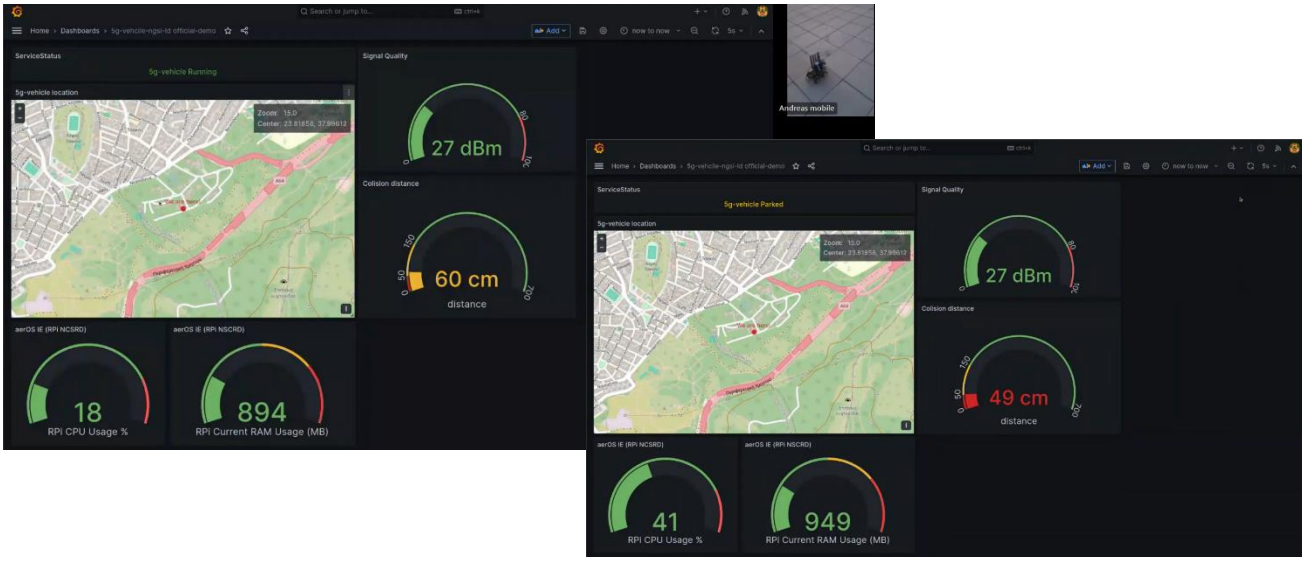


Figure 44. aerOS EAT integrated in IoT application.

As a final stage a new domain is registered in the continuum. This is the above-mentioned mobile domain. All the procedure to enrol a new domain in the aerOS continuum is demonstrated. The required aerOS packages, as described in section 6.5, are deployed, and all information is propagated along the continuum based on the aerOS federator capabilities. This is also clear in the aerOS portal where a new domain has appeared, a new domain which hosts two new IEs.

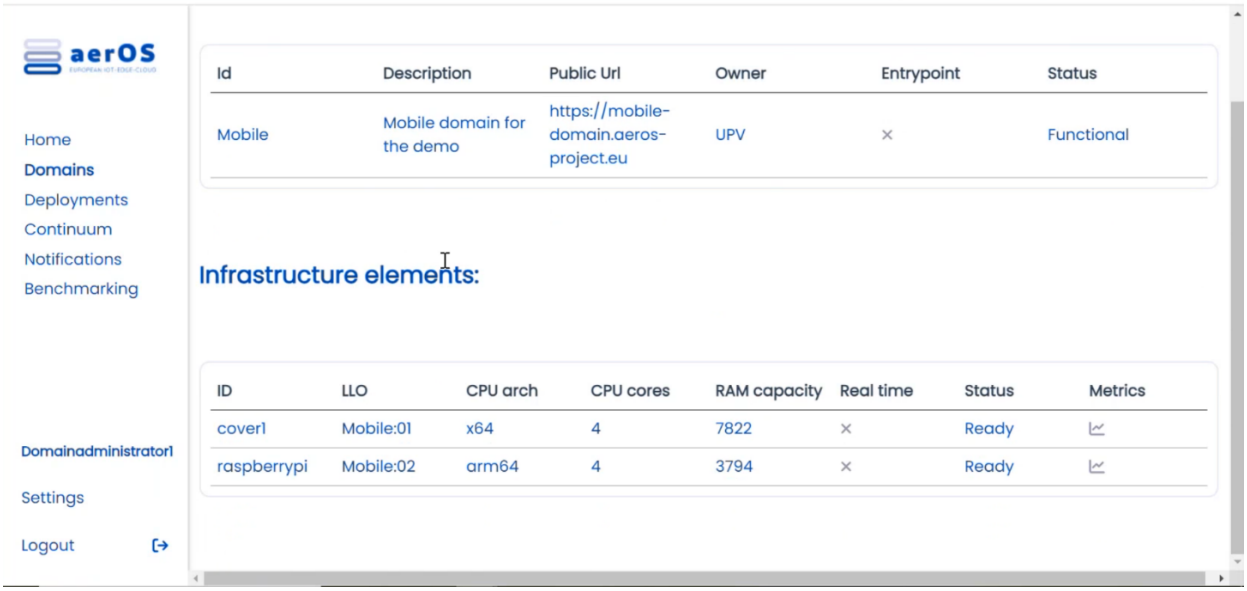


Figure 45. Mobile domain integration.

But this is not all, as it is also important to demonstrate aerOS automated migration capabilities. To do so, the IE hosting the navigation service component is artificially overloaded by running a CPU-intensive workload in form of a K8s Job. As a result, when the CPU overload exceeds the set threshold (80% of RAM usage), the self-orchestration package triggers an OVERLOAD event and initiates a re-orchestration process for this service component. This process includes accessing an internal HLO (High-Level Orchestrator) endpoint submission, including the service component ID and event type (OVERLOAD). Subsequently, the HLO queries the federated information regarding candidate IEs across the continuum that fulfil the set constraints and makes a new selection. In this case, the selected IE was in the mobile domain, and the migration process began. The local LLO (Low-Level Orchestrator) is requested to remove the service component from the hosting IE, and the

remote allocation engine HLO endpoint is called to submit a new allocation request. When the process is complete, the 5G-car continues to be navigated, but with the difference that the controlling component is located in a new IE belonging to another domain without interrupting the IoT application's functionality.

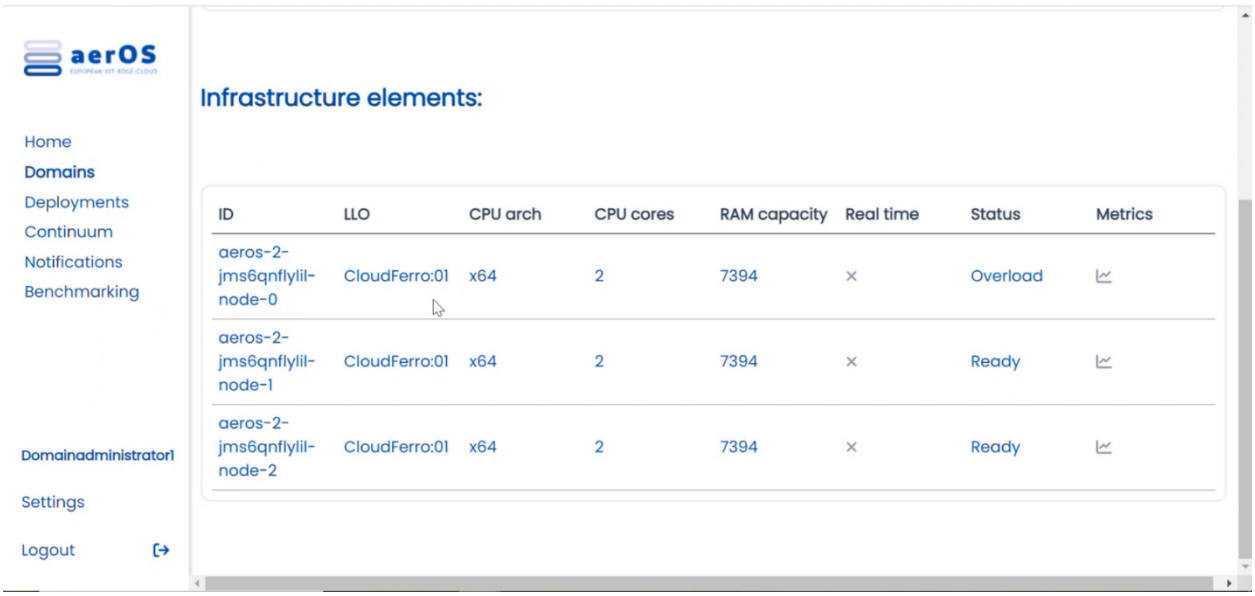


Figure 46. IEs status in management dashboard, IE Overloaded..

Last but not least, when the developer needs to stop the IoT application, it can always be requested via the management portal. Although the service components are deallocated from the hosting IEs, all the application constraints and requirements are stored in the aerOS continuum knowledge graph, allowing the application to be restarted at any time. Any HLO in any domain can receive this request, proceed with service component placement as needed, and update the service component status and connected allocated resources. This implies that the aerOS Meta-OS can support full life-cycle management (LCM) for applications running within it. The current state of applications is always registered, and their requirements too, enabling thus them to be started, stopped, or even migrated, while always registering their connection with the hosting IEs. This process is fully supported by the aerOS continuum ontology described in 5.4.3.3

7.1.3. Achievements and Conclusions

The deployment and operation of the MVP-based demonstrator successfully initiated a “real-life” use case, validating the aerOS architecture's capability to establish a functional continuum and operate as a Meta-OS on top of it. This deployment integrated various access media networks and resources, abstracting underlying details to provide IoT developers with a common execution environment for hosting synergetic applications. This demonstrated how aerOS can effectively integrate diverse resources and manage service orchestration, proving its potential as a Meta-OS.

Key architectural concepts were instantiated, and technical components developed within WP3 and WP4 under the aerOS architecture blueprint were deployed to support the goals set for aerOS. Concepts of established aerOS fabrics were validated. A diverse set of resources are integrated and access to them is abstracted while a common execution environment enabled services to run on top of each element integrated, exhibiting thus network and compute fabric. An aerOS service fabric, based on a set of microservices integrated and interacting under the reference architecture vision, provided seamless and efficient orchestration and life-cycle management (LCM) of IoT services across the underlying pool of resources. AI-supported allocation algorithms, integrating IE capabilities and energy efficiency criteria, provisioned the best placement of services and facilitated proper migration when needed. aerOS data fabric demonstrated its capabilities to establish mechanisms for sharing both aerOS resources and IoT data across the continuum. The extendable aerOS knowledge graph, as part of the data fabric, was validated to semantically enrich and link aerOS entities efficiently, seamlessly connecting and integrating diverse resource datasets, enabling context-rich insights and

decisions. All these capabilities were demonstrated under the constraints of a robust security framework and granular privacy access to integrated data.

Finally, it is important to say that while the MVP and the demonstrator deployed on top of it have successfully validated key aspects of the aerOS architecture, they do not encompass the full spectrum of functionalities developed throughout the project. Several additional components, such as Serverless functionalities, EAT, IOTA for trust establishment, low-code development tools, and more, are integral to the complete aerOS solution. These components will be rigorously validated in subsequent deployments, each tailored to the specific needs of various pilot scenarios. These future deployments will provide a more comprehensive evaluation of aerOS capabilities, ensuring that all project-developed functionalities are thoroughly tested and optimized for real-world applications across different domains. This iterative approach will ensure that aerOS can deliver robust, scalable, and versatile solutions to meet the diverse requirements of IoT service developers and stakeholders.

In conclusion, the demonstrator's success underscores the robustness and flexibility of the aerOS architecture, setting the stage for further advancements and real-world applications.

7.2. aerOS Reference Architecture in project pilots

aerOS has clearly stated its goal to establish a common Meta-OS which will support a collaborative IoT-Edge-Cloud execution runtime to support flexible deployments. It has also strongly correlated this task with the ability to integrate all possible industry domains, across the continuum, and to bring substantial benefits to them both operational and financial. To this end all pilots initially provided their requirements, to support the process of establishing a robust architecture blueprint and this is roughly presented in section 2. While the instantiation of aerOS Reference Architecture within each pilot, is a currently running process, all pilots have already proceeded to designing the deployment of aerOS Meta-OS and the migration of their existing services, which support their industry business cases, within the aerOS continuum. This in one hand will validate their requirements on aerOS as a technically enhanced orchestrating environment but will also validate their expectations to enhance their business processes with enabling and integrating capabilities with synergies across the continuum. So, since aerOS architectural blueprint is compiled, all pilots are in the process of mapping aerOS reference architecture concepts and components on top of existing infrastructure, computing environment, and business processes.

7.2.1. Data-driven Cognitive Production Lines

7.2.1.1. Gren manufacturing (zero net-energy) and CO2 footprint monitoring

The first scenario in the "Data-Driven Cognitive Production Lines" pilot focuses on green manufacturing and CO2 footprint monitoring. This scenario is hosted by the Switzerland Innovation Park in Biel/Bienne, using the Swiss Smart Factory's test and demonstration platform, called Lighthouse Project Industry 4.0. Consisting of a modular production line for customised drones, this facility encompasses all stages of production, from IT and software, parts production, product assembly, packaging, right through to remanufacturing activities.

aerOS is deployed on selected assets in the production line to collect data locally (at the edge) and process and analyse it remotely (in the cloud), seamlessly. As a result, accurate feedback can be sent directly to users, i.e. customers and manufacturers, who can in turn make informed decisions about the choice of product ordered or the layout of the production line, thereby reducing their environmental impact. This AI-supported feedback would be represented by a dashboard for production, and integrated into the drone configurator web platform which is linked to the ERP system on the customer’s side, while enabling the implementation of a Digital Product Passport (DPP).

aerOS provides a clear added value for such a case, by providing a reliable IoT-Edge-Cloud Meta-OS that will swiftly handle the data transfer among the edge and cloud locations, and will allow a smart allocation of different processing needs to satisfy the requirements of the end user.

Figure 47 below represents the technical architecture of aerOS elements in the existing Swiss production line of drones, with the different aerOS domains deployed to implement selected aerOS services.

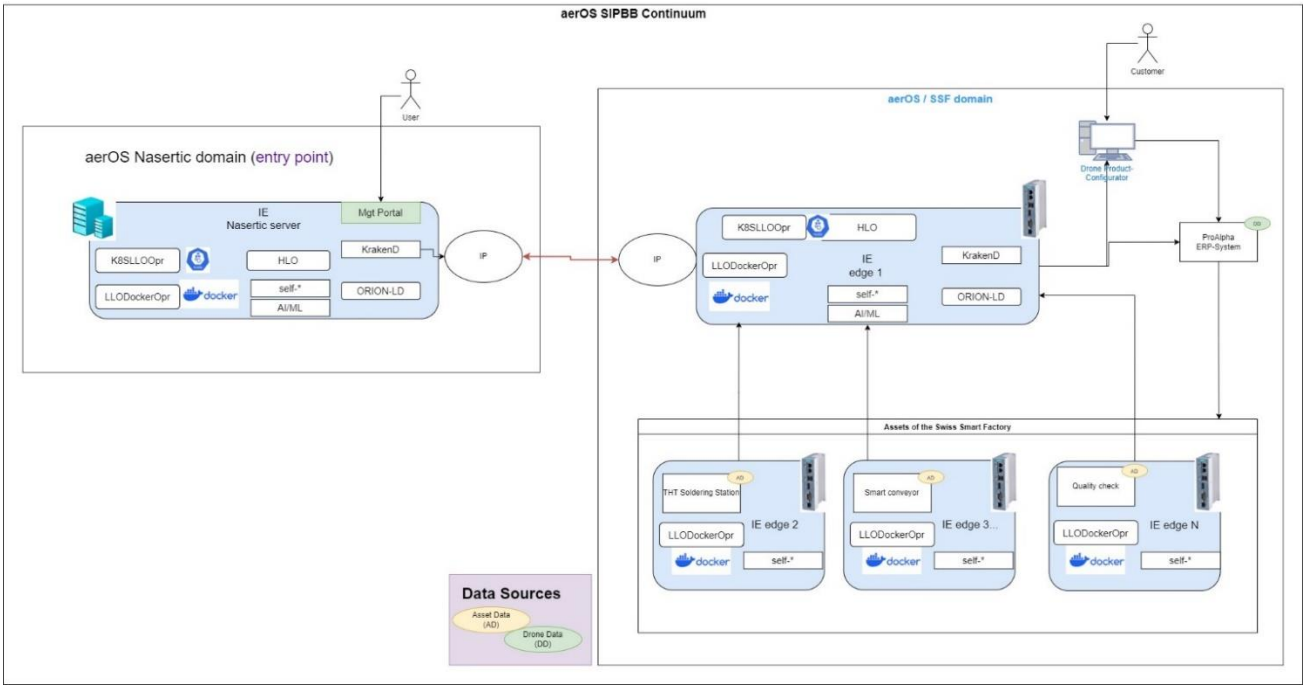


Figure 47. aerOS compliant high-level diagram for pilot 1 SIPBB" scenario.

The following table provides a granular enumeration of the aerOS components involved, associating them with the relevant pilot functions and modules, while explaining the reasoning behind the available options and final deployment decisions

Table 3. Granular mapping of aerOS components within pilot 1 "UPDATE ME" scenario.

Item of aerOS RA	Mapping to pilot/vertical relevance and significance
Domains	Since this scenario involves collaborative efforts between distinct geo-separated entities, this scenario is comprised of 2 domains which include: • Nasertic domain: Nasertic acts as a common compute infrastructure provider across all scenarios this use case. • SSF domain: the Swiss Smart Factory (SSF), located in the Switzerland Innovation Park Biel/Bienne (SIPBB), encompasses the entire drone production chain, i.e. all the assets on which the aerOS components will be deployed.
Infrastructure Elements	The infrastructure elements integrated in the aerOS domains are a collection of computing resources and fall under 3 different categories. • Nasertic servers. • SSF's Edge 1 Entrypoint of the domain (HLO). • SSF's Edge 2, 3, …n (LLOs), each containing an OPC-UA and/or Rest- API server.
IoT services definition	IoT services part of the SIPBB use case can be divided in 3 main categories: • Product configuration: customers can configure their personalised drone on a web-based platform linked to the production line's Enterprise Resource Planning (ERP) system, and this data is then automatically sent to the MES and plant assets to start production. • Data collection: various sensors are used in the use case to collect information on the condition of products (RFID asset tracking, location) and

	machines, the environment (temperature, humidity, vibrations, atmospheric pressure) and energy-related data (power, CO2 footprint). • Data visualisation: this data is then pushed to the ERP/MES or pulled from the assets to be visualised in dashboards via different software, displaying graphs on asset productivity, the plant's environmental values, the location and carbon footprint of products. aerOS will harmonise this data flow across assets and enable the deployment of new services, such as integrated analytics and personalised AI models.
Self-* tool suite	Self-awareness to expose the state and availability of IEs and self-orchestration components are selected for deployment in each IE.
Two-level orchestrator	HLO, as part of aerOS core services, is deployed in both domains to support efficient and optimized service placement on the available IEs. In the SIPBB domain, it will be deployed on top of IE Edge 1, which also hosts the production line's Manufacturing Execution System (MES). Since the underlying container management framework for the application domain is Docker, the Docker LLO will also be deployed and used.
Data Fabric	An instance of FIWARE Orion-LD context broker will be deployed in each domain to host all pilot data encompassing information regarding the machine status, environmental and energy related data (in JSON format). Data fabric components are expected to enable seamless data exchange between the different assets to achieve a level of transparency and availability required to certify accurate carbon footprint monitoring and optimise the production line accordingly.
Semantic translation and annotation	The pilot is based on processing of the following types of data, and ensures that the appropriate notation and handling is performed based on the aerOS data principles: 1) <u>SSF domain</u> • Product configuration set by the customer (drone type, color etc.) and sent to the ERP system. • RFID asset tracking of the products. • Machines status: name, status, order ID, timestamp. • Environmental data: temperature, humidity, vibrations, atmospheric pressure. • Energy monitoring data: current and cumulated power, current and cumulated CO2 footprint. • Production process data related to the assets. 2) <u>Infrastructure data</u> Beyond IoT applications' data, there is a constant data flow related to network and computing resources usage. This is foreseen, to be handled, by the overall aerOS architecture design and the self-* components discussed elsewhere in this table.
Networking and service mesh	Rest API is currently established to ensure communication between the assets.
Low-code programming tools	SIPBB uses Node-RED (self-configured) in the Swiss Smart Factory for several assets and tasks. The purpose of Node-RED is data forwarding, data processing and dashboarding.
Cybersecurity components	KrakenD, OpenLDAP and KeyCloak will be used for cybersecurity.
Explainable and Frugal AI	AI will mainly be used to support production optimisation, which requires an explanation so that the user, i.e. the manufacturer, can make better decisions by taking into account all the factors involved (external to production, for example).

EAT	As this scenario is already serverless, the use of the serverless function will enable the aerOS services in the use case to be deployed and used more easily, without disrupting the current production system.
Trustworthiness	Trust must be ensured in the use case, particularly between users, so that each user can only access specific data relating to their own assets.
Portal	aerOS management portal will be deployed in the "aerOS Nasertic domain" to manage the services deployed in the pilot from a centralised UI.

7.2.1.2. Automotive Smart Factory Zero Defect Manufacturing

The 'Automotive Smart Factory Zero Defect Manufacturing' is the second of the scenarios which falls under the 'Data-Driven Cognitive Production Lines.'

The Scenario is executed by Innovalia at the Automotive Intelligence Center (AIC) in Bilbao, and is also depending on Nasertic's Infrastructure, which provides the entry domain and allows the virtualization of the M3 Metrology platform.

The goals are fundamentally two. First, to achieve remote tactile operations by low latency communication protocols, which drive towards autonomous operations and the delocalization of physical resources. And second, to place different IoT sensors on the CMM to promote real time data acquisition and foster the previously mention remote operations.

From a quantitative standpoint, this scenario aims to reach the target of three key performance indicators, like Production process accuracy (increase a minimum of 10% over baseline), Digital service programming time (from 2 weeks to 2 days), or dimensional control quality control productivity (from 3 parts/hr to 5 parts/hr). And from a qualitative point of view this scenario will boost trust, security, and sustainability, between others.

The image below (Figure 48) depicts the first conceptual layer of the architecture continuum. The first stage is the metrology project engineering where all the preliminary simulations take place, and where aerOS is still not involved.

The next stage and where aerOS starts to contribute is production metrology. Here M3 executes the orders given to the machines through the control layer.

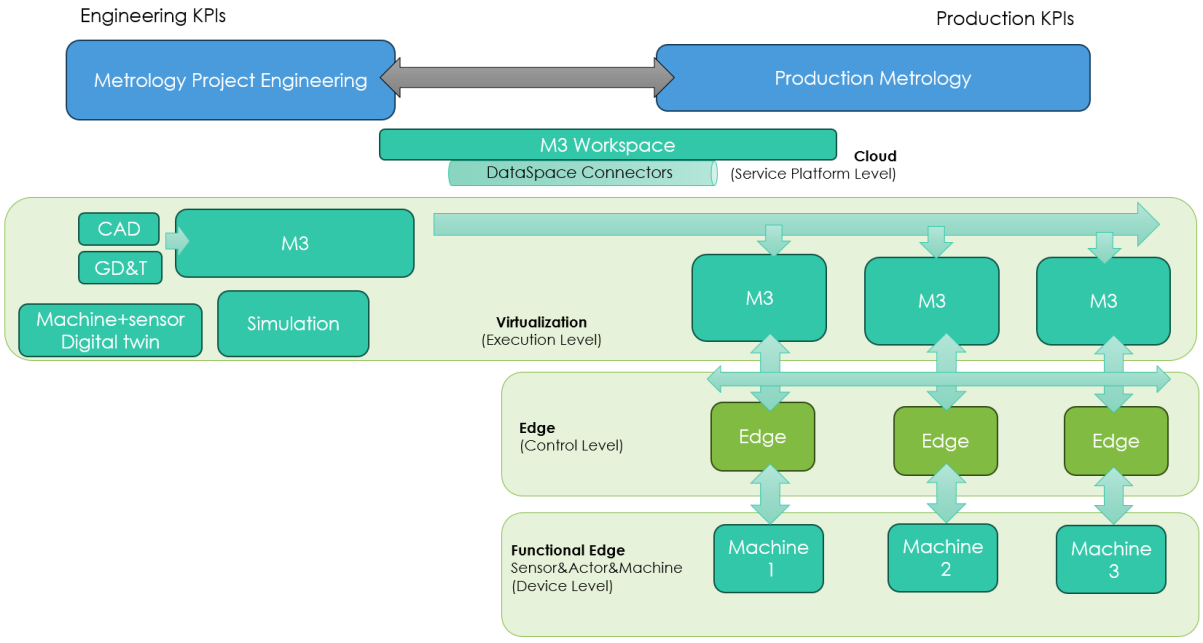


Figure 48. ZDM dimensional metrology continuum part of aerOS Trial.

Within Production metrology and zooming in from the perspective of the services involved, there are two main branches, the left one with robolink (RL), controller and machine, handling the physical movement of the machines, and the right focus on monitoring and automation, which is where aerOS shows its potential.

RL environment communicates via OPC UA to the sensors and different devices, and together with OPC UA Codesys interface, it ensures automation and real-time monitoring.

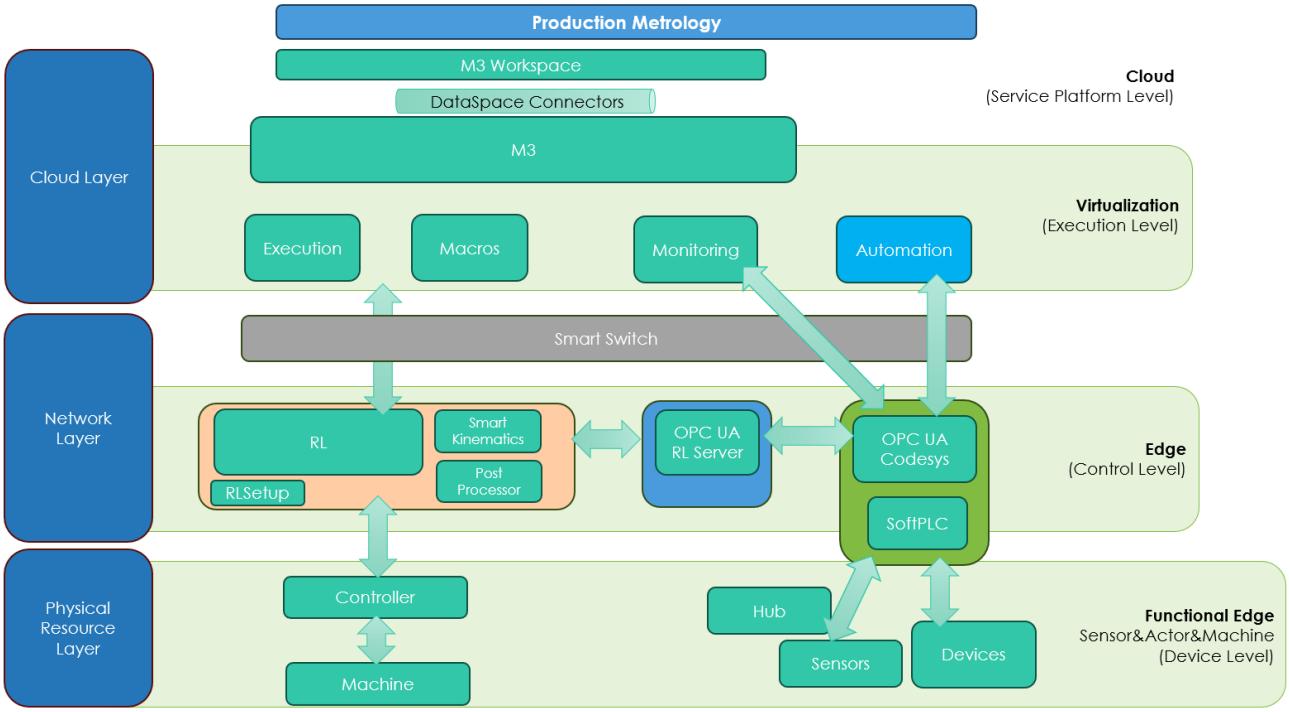


Figure 49. ZDM dimensional metrology service suite deployable in the aerOS continuum.

The diagram below (Figure 50) shows a high-level architectural view of the aerOS elements on top of the existing computational resources available across each aerOS domain that will be a part of this use-case scenario. And each IE possess an OPC UA RL server that allows communication with the virtualization layer.

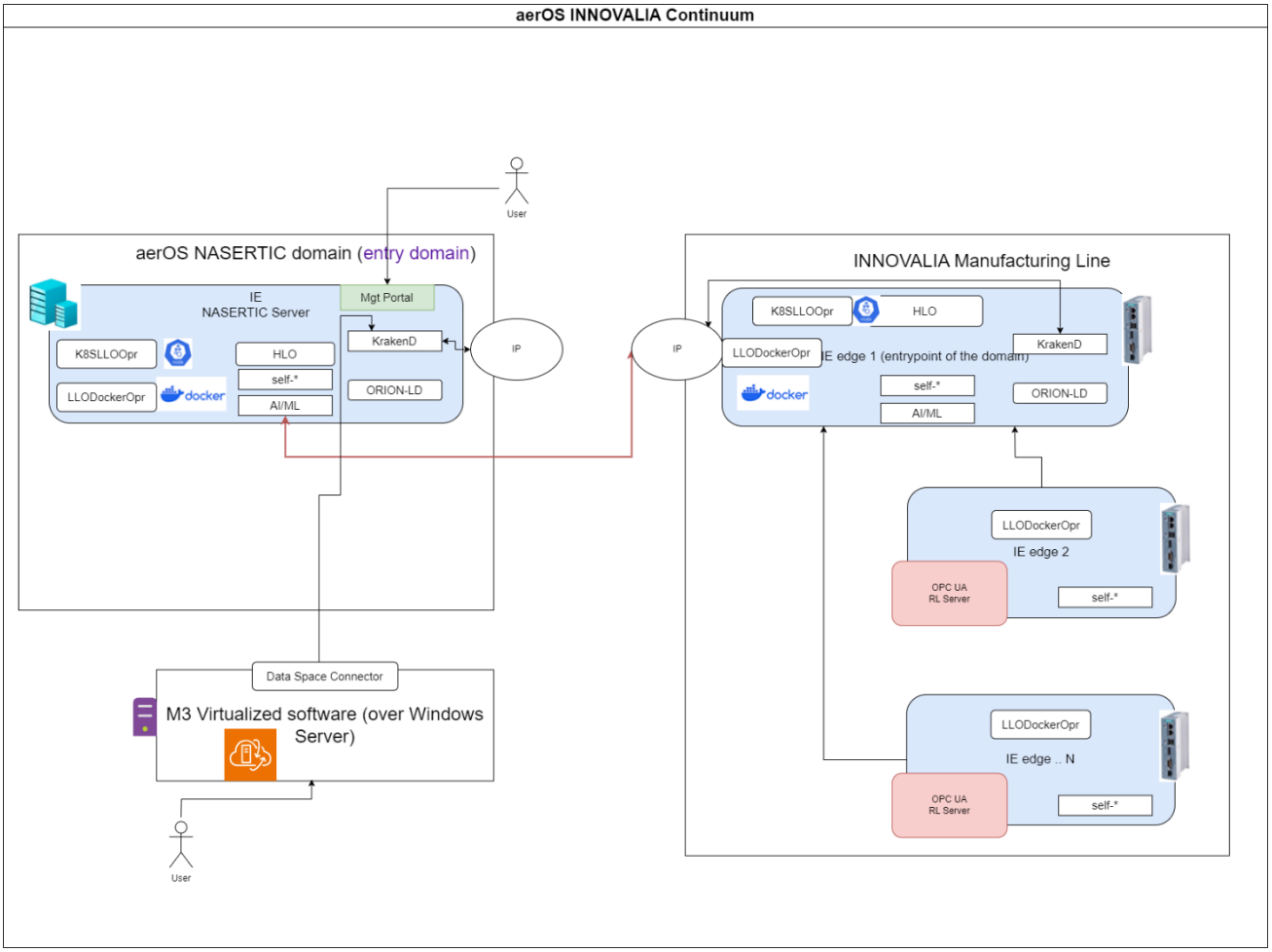


Figure 50. aerOS compliant high-level diagram for pilot 1 "'Automotive Smart Factory Zero Defect Manufacturing'" scenario.

The table below provides a granular enumeration of the aerOS components involved and depicted on the high-level view (Figure 50), associating them with the relevant scenario functions and modules, while explaining the reasoning behind the available options and final deployment decisions.

Table 4. Granular mapping of aerOS components within pilot 1 "Automotive Smart Factory ZDM" scenario.

Item of aerOS RA	Mapping to pilot/vertical relevance and significance
Domains	Since this scenario involves collaborative efforts between distinct geo-separated entities, this scenario is comprised of 3 domains which include: • Nasertic domain: Nasertic acts as a common compute infrastructure provider across all scenarios this use case. • Innovalia manufacturing line domain • M3Virtualised software (over windows server) domain
Infrastructure Elements	The infrastructure elements integrated in three aerOS domains are a collection of computing resources and fall under 3 different categories. • Nasertic servers. • Innovalia's Edge 1 Entrypoint of the domain (HLO) • Innovalia's Edge 2,3,4…n (LLO), each containing a OPC UA RL Server

IoT services definition	• Configured for direct CNC controlled CMMs. Communications link with: PC: Gigabit Ethernet. Direct Software integration. EAGLE driver. RL: CMM middleware for metrology process work with OPC UA communication. • RL connection with SoftPLC through OPC UA (RL Server) • M3 Execution for monitoring and automation of sensors and machines (Resource Layer)
Self-* tool suite	Self-awareness: To expose the state and availability of IEs Self-orchestration: Fed by Self-awareness, helps to manage each IE. Self-Optimisation and adaptation: To optimise the process and detect potential anomalies.
Two-level orchestrator	HLO is deployed in Nasertic and Innovalia's domains to support efficient and optimized service placement on the available IEs. Since the underlying container management framework for the application domain is Docker, the Docker LLO will also be deployed and used.
Data Fabric	For the remote tactile operation, data must be recorded and transmitted over a secure channel that allows uninterrupted operations, and sensors and devices will circulate data continuously through a SoftPLC (via OPC UA) in order to achieve the right level of automation.
Semantic translation and annotation	The scenario is based on processing of the following types of data, and ensures that the appropriate notation and handling is performed based on the aerOS data principles: 1) Innovalia Domain The data within the Innovalia domain can be classified into 3 broad categories: • Production process data related to CMM. Machine → Controller → RL → M3 execution level. • Incoming orders from and to the M3 Workspace. Monitoring and automation communication with sensors and devices. • Energy consumption data from the machines, which will feed the aforementioned services. 2) Infrastructure Data Beyond IoT applications' data, there is a constant data flow related to network and computing resources usage. This is foreseen, to be handled, by the overall aerOS architecture design and the self-* components discussed elsewhere in this table.
Networking and service mesh	The first and biggest concern of the pilot is to achieve the necessary virtualization level of the M3 platform and/or OPC UA RL Server in order to seamlessly integrate aerOS functionalities. To achieve this, substantial efforts are being allocated. And on other hand the communication between the cloud and edge layer from the automation and monitoring side, since is critical for aerOS to successfully work. Thanks to the reliability of OPC UA protocols, this shouldn't be an issue.
Low-code programming tools	M3 platform includes visual and importing capabilities that allow little to no coding to fully deploy different metrology operations. With aerOS, these options could increase and/or improve.
Cybersecurity components	The Entrypoint domain will host the Identity Manager which is implemented integrating based on Keycloak, connected with LDAP. KrakenD will be deployed in each domain for providing authentication and authorization.

Explainable and Frugal AI	With consideration of the diversity in the computational capability of IEs of the continuum the use of Frugal AI could be potentially explored especially considering the Edge devices (gages and sensors) for supporting the KeVs such as environmental and economical sustainability.
Portal	aerOS management portal will be deployed in the "aerOS Nasertic Domain." It will interface with KrakenD to provide access control to registered users.

7.2.1.3. Zero Ramp-up Safe PLC Reconfiguration for Lot-Size-1 Production

The goal of this scenario is to establish a versatile production system distinguished by modularity, efficiency, and adaptability to dynamic manufacturing conditions. A key focus is placed on the development of a cyber-physical system that synergizes the capabilities of automated guided vehicles (AGVs) and robotic arms. This integration is facilitated through aerOS decentralized intelligence and communication technology, complemented by Siemens cloud systems services known as SIMATIC Industrial EDGE. This use case scenario delves into the exploration of flexible adaptation within a production line. This involves the modification of a particular process step to address a specific task, seamlessly facilitated through the utilization of a decentralized intelligent cloud system. This innovative approach permits a dynamic adjustment of the manufacturing process and ensures a smooth transition back to the previous production line configuration. Fundamentally, this scenario seeks to offer manufacturing solutions characterized by flexibility and modularity, departing from the limitations associated with contemporary static production systems. The effectiveness of this innovative system will be rigorously assessed for feasibility and robustness during a trial phase, poised to pave the way for the widespread adoption of this novel cyber-physical system and enhance the overall efficiency of the manufacturing process.

A high-level view of the design and an overall projection of aerOS architecture on top of pilot's integrated resources is presented in the following diagram. This diagram presents the correspondence of the existing application modules and computational resources to the aerOS concepts and components that implement aerOS core functionalities.

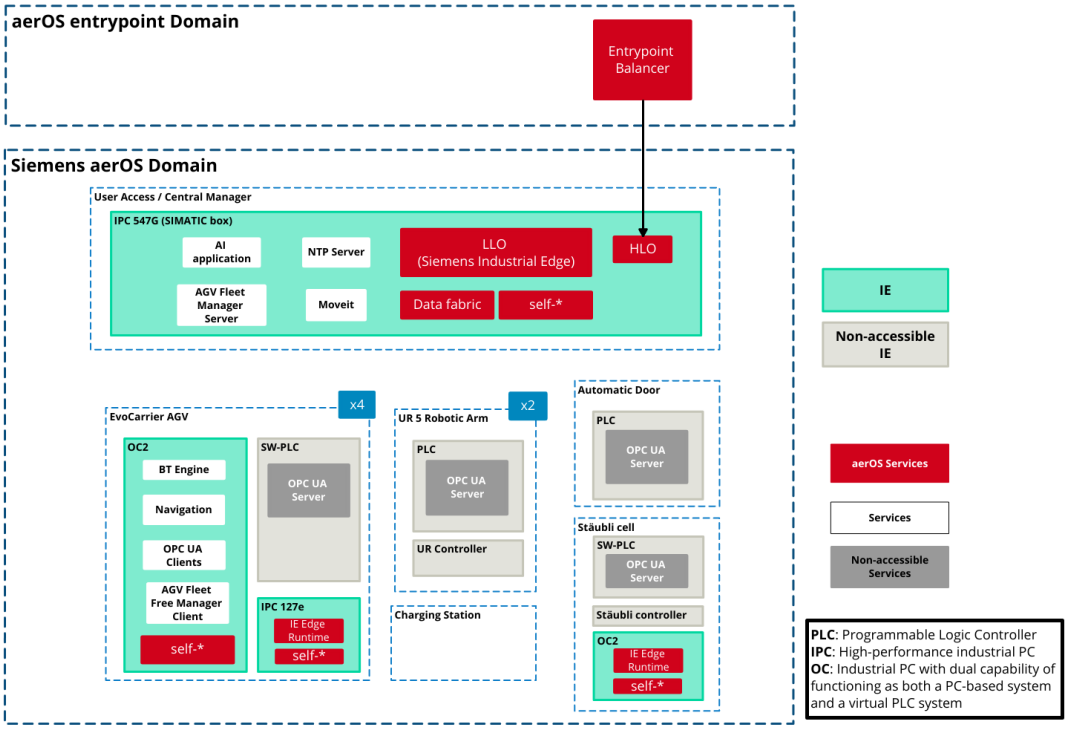


Figure 51. aerOS compliant high-level diagram for "Zero Ramp-up Safe PLC Reconfiguration for Lot-Size-1 Production" use case scenario.

The following table provides a granular enumeration of the aerOS components involved, associating them with the relevant pilot functions and modules, while explaining the reasoning behind the available options and final deployment decisions.

Table 5. Granular mapping of aerOS components within pilot 1 "Zero Ramp-up Safe PLC Reconfiguration for Lot-Size-1 Production" scenario.

Item of aerOS RA	Mapping to pilot/vertical relevance and significance
Infrastructure Elements	The infrastructure elements fall under two main categories, accessible IE, and non-accessible IE (Siemens PLCs in this case). • **Accessible IE:** These devices feature a Debian-based ecosystem (Siemens Industrial OS) that supports self-* and the Siemens LLO (Siemens Industrial Edge Runtime). **Devices**: High-performance industrial PCs for server purposes (Siemens IPC). Lower-performance industrial PCs for AGVs and robotic arms (Siemens Open Controller) The advantage of these Siemens devices over standard industrial PCs is their dual capability of functioning as both a PC-based system and a virtual PLC system. All these computing resources are integrated as aerOS Infrastructure Elements using the self-* package. • **Non-accessible IE**: These include PLC devices that communicate with Siemens devices and services, as well as aerOS services, using industrial protocols like OPC UA. **Devices**: PLC of the AGVs, robotic arms, PLC of the staübli cell and automatic door. All non-accessible devices are communicated and integrated in aerOS by using communication tools as defined it their proper AsyncAPI (MQTT, OPC UA, ROS2, etc.).
Domains	This scenario utilizes two domains: the primary "SIEMENS aerOS Domain" and the "aerOS Entrypoint Domain". Siemens domain's Data Fabric will have a prominent role as strict policies specify that data must not leave the premises of the company. However, the management portal, including the Entrypoint balancer and some auxiliary services, will be hosted on an external server provided by Nasertic, referred to as the "aerOS Entrypoint Domain."
IoT services definition	AGVs and robotic arms are IT/OT machines running various services on our infrastructure. **AGVs:** • Navigation: Using the SIMOVE ANS+ toolset developed by Siemens for precise navigation. • Sensor Data Reading: An application to read and process data from various sensors. • Lift Movement: An application to control the movement of the AGV's lift. **Robotic Arms:** • Arm Movement: An application to control the robotic arm, allowing it to move to specified positions. **Behaviour Trees:**

	• An application that triggers different services and establishes connections with the control part (OT part) through industrial protocols like OPC UA, making the control accessible from the IT part. The benefit of using aerOS is its ability to decide where and when to run these different services based on available resources. This capability enhances the performance and optimization of our flexible production line by dynamically managing resources and services.
Self-* tool suite	Self-* capabilities will monitor and collect essential information for the High-Level Orchestrator (HLO) to make informed decisions. This capability can even facilitate the recovery or reconfiguration of services within the AGVs or robotic modules in the event of failure or anomalous behaviour.
Two-level orchestrator	HLO, as part of aerOS core services, is deployed in "Siemens aerOS Domain" to support efficient and optimized service placement on the available use case IEs. Siemens will employ its proprietary low-level orchestrator (LLO) known as Siemens Industrial EDGE, offering functionalities comparable to the widely recognized Kubernetes. This orchestrator provides additional advantages, such as the ability to download applications directly from the Siemens Cloud Store. Focus is on exploring the integration of this framework with external aerOS services, such as the High-Level Orchestrator (HLO).
Data Fabric	An instance of FIWARE Orion-LD context broker will be deployed in the server of the "Siemens aerOS Domain" to host all use case data encompassing information regarding the availability of resources in each IE. The information is stored in the Orion-LD broker with NGSI-LD specification. aerOS components, provided as part of the data fabric, will be deployed to consume data published by devices and applications.
Semantic translation and annotation	The use case is based on processing of the following types of data, and ensures that the appropriate notation and handlings is performed based on the aerOS data principles: 1) <u>AGV, Universal Robot Arm UR5e and staübli robotic arm cell data</u> One of the main components in this use case are the AGVs and robotic arms. It is important to now their status at any time. Recorder data are machine generated and are initially stored in timeseries format. This information will use and industrial protocol such as PROFINET, OPC UA or MQTT. 2) <u>Pilot Application & Context Data</u> The pilot assumes a set of application components that dynamically interact and exchange processed information. Beyond the obvious fact that all these data need to be "recognized" (syntactically and semantically) by all existing and future applications, they also need to be correlated (linked) to benefit from linked-data advantages. All the above will be addressed and automated with the integration of the product manager tools, presented in data fabric. 3) <u>Infrastructure Data</u> Beyond IoT applications' data, there is a constant data flow related to network and computing resources usage. This is foreseen, to be handled, by the overall aerOS architecture design and the self-* components.

Networking and service mesh	Due to internal policy, Siemens information needs to be in the local network (not connection to the outside). The management portal with the Entrypoint balancer and some auxiliary services will be deployed in an external domain provided by Nasertic ("aerOS Entrypoint Domain"). Networking-wise, the main concern is to securely expose aerOS API on each domain. The ingress to provide a single and controlled entry-point for the "aerOS Entrypoint Domain" APIs which will integrate certificates for secure and private communication (SSL/TLS), the API gateway which will translate and route requests to underlying aerOS services (e.g. federation, HLO), a certificate manager to manage (even self-signed) certificates and the load-balancer to advertise a routable IP are selected for deployment.
Low-code programming tools	Behaviour trees, functioning as graphical low-code interfaces, enable users to define triggers and adjust parameters interactively. Behaviour trees trigger functionalities within already running applications, activating specific service functionalities such as specific orders for the AGV (move, lift, etc.) Additionally, AsyncAPI, is used to model interfaces of existing asynchronous and event-driven communication based on MQTT, OPC UA and ROS2 messaging protocol.
Cybersecurity components	Primarily the "Siemens aerOS domain" will host the IdM which is implemented integrating Keycloak and LDAP for providing authentication and authorization. It could be optionally deployed in the "aerOS Entrypoint Domain" provided by Nasertic. Roles with related permissions will be defined through OpenLDAP according to aerOS architecture.
Explainable and Frugal AI	The Explainable AI is used in the HLO allocator. The aerOS continuum consists of heterogenous, networked Infrastructure Elements with resource constraints and hardware capabilities which allow on-demand service execution. To execute a service on an IE, the allocator needs to adhere to computational and hardware constraints while, for example, minimizing the overall power consumption of the continuum.
EAT	Embedded analytics are pivotal in collecting pertinent information for the HLO to make informed decisions. Monitoring CPU resources is essential for the HLO to analyse which devices possess sufficient resources for executing resource-intensive or complex operations, such as AI services. The EAT functions will be useful to detect anomalies and to generate samples of data (Stratified Sampling and Anomaly detection) to be able to decide where and when to run the different services depending on the available recurses.
Trustworthiness	In this use case, trustworthiness plays a crucial role in ensuring the availability of each IE. For instance, understanding whether an AGV has a low trust score (possibly due to connection issues) allows us to prioritize other AGVs with higher trust scores. This approach ensures the smooth operation of the flexible production line.
Portal	aerOS management portal will be deployed in the "Siemens aerOS Domain". It could be optionally deployed in the "aerOS Entrypoint Domain" provided by Nasretic. This is a relevant case of multistakeholder preference for Entrypoint domain selection (see 5.3.2).

7.2.1.4. AGV zero break-down logistics at pre-industrial level

The 'AGV zero break-down logistics at pre-industrial level' is one of the scenarios which falls under the 'Data-Driven Cognitive Production Lines.' The test bed for this scenario is jointly hosted in MADE Competence Centre and POLIMI's Industry4.0 lab, while also relying on computational infrastructure hosted by Nasertic. The scenario is developed on top of an existing valve production line at MADE cc. which uses an AGV-based internal logistics solution for transport of raw, semi-finished and finished products during different stages of production. In its current form AGV travels are sub optimal owing to the limitations posed by the existing centralized production informative system. The scenario aims to demonstrate the integration of aerOS architecture alongside existing manufacturing systems. The scenario also intends to reap the benefits in terms of improving some key performance indicators like reduction of optimization of AGV travels and improvements in CO2 emissions by leveraging AI services offered by the aerOS platform. The scenario also intends to simulate demand/order backlog based automated order outsourcing scenario wherein POLIMI's Didactic Factory and AGV will be used for playing the role of a remote outsourcing location.

Figure 52 shows a high-level map of the aerOS architecture elements on top of the existing computational resources available across each aerOS domain that will be a part of this use-case scenario. Additionally, the figure also tries to present preliminary attempt to mapping of some of the pre-existing and under development application services to be used in this scenario.

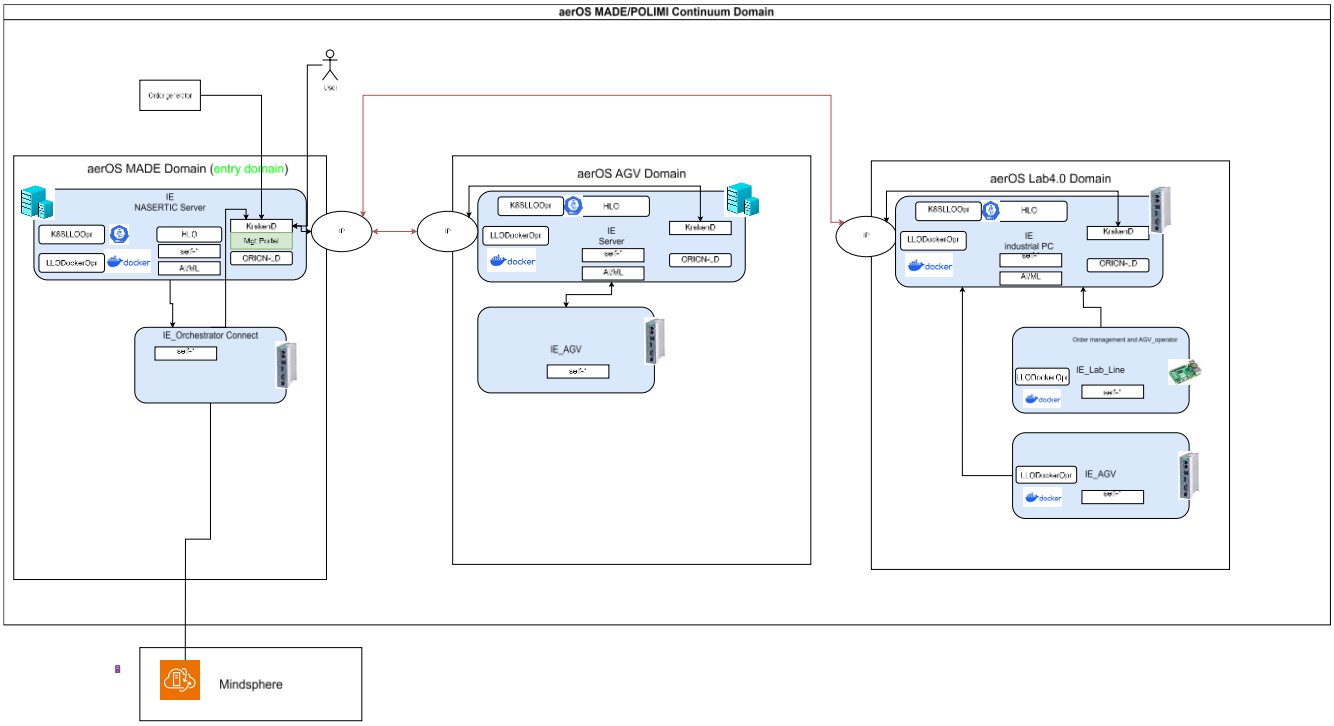


Figure 52. aerOS compliant high-level diagram for pilot 1 "AGV zero break-down logistics at pre-industrial level" scenario.

Table below provides a granular enumeration of the aerOS components involved and depicted in Figure 52, associating them with the relevant pilot functions and modules, while explaining the reasoning behind the available options and final deployment decisions.

Table 6. Granular mapping of aerOS components within pilot 1 "AGV zero break-down logistics at pre-industrial level" scenario

Item of aerOS RA	Mapping to pilot/vertical relevance and significance
Domains	Since this scenario involves collaborative efforts between distinct geo-separated entities, this scenario is comprised of 3 domains which include:

	• Nasertic domain: Nasertic acts as a common compute infrastructure provider across all scenarios this use case. • MADE AGV domain: MADE cc acts as the primary testbed for this use case and hosts a variety of computational equipment ranging from on premise servers to industrial edge devices. • POLIMI Industry4.0 Lab domain: POLIMI plays the role of a remote production out-sourcing location. and offers an added variety in terms of the infrastructure element pool.
Infrastructure Elements	The infrastructure elements integrated in three aerOS domains are a collection of computing resources and fall under 5 different categories. • Cloud computational resources in the form of 2 virtual machines hosted on top of Nasertic servers. • On Premise Server VM hosted in MADE cc. • Industrial Edge devices one each hosted in MADE and POLIMI. • Raspberry Pi 5- 8GB version hosted in POLIMI. • On board computer, hosted within POLIMI AGV.
IoT services definition	• POLIMI will use an existing toolset developed for AGV actuation under the purview of L4MS, BEinCPPS and FASTEN EU (doi: 10.1016/j.promfg.2020.02.055) projects. The existing suite relies on the ROS1 interfaces to FIWARE Orion-V2 broker to actuate the AGV. • The current suit will be expanded by adding a scenario specific functionalities like addition of a scenario specific interactive front-end web app. The app will provide an interactive interface to the Line operator for retrieval and dispatch of raw materials and processed components, respectively. Additionally, APIs exposed by MADE cc's Order management systems will be explored for potentially sending automated updates related to orders completed/dispatched from POLIMI premises. • Potential exploration of migration to NGSI-LD can also be explored under the conditional availability of a stable FIWARE agent for ROS1
Self-* tool suite	Self-awareness: To expose the state and availability of IEs Self-orchestration: Fed by Self-awareness, helps to manage each IE. Self-Optimisation and adaptation: To optimise the process and detect potential anomalies.
Two-level orchestrator	HLO, as part of aerOS core services, is deployed in all 3 domains to support efficient and optimized service placement on the available IEs. Since the underlying container management framework for the application domain is Docker, the Docker LLO will also be deployed and used. The AGV actuation application in POLIMI domain will be packaged as a docker compose and thus Docker LLO will be used for orchestration decisions enforcement.
Data Fabric	An instance of FIWARE Orion-LD context broker will be deployed in each domain to host all pilot data encompassing information regarding AGV and Application related data from both POLIMI and MADE domains. Data fabric components are expected to facilitate cross domain data sharing between the two entities to achieve a level of synchronisation required to achieve an acceptable level of automation in the smart outsourcing scenario.
Semantic translation and annotation	The pilot is based on processing of the following types of data, and ensures that the appropriate notation and handling is performed based on the aerOS data principles: 1) MADE Domain

	The data within the MADE domain can be classified into these broad categories: • Production process data related to AGV actuation and Station data. • Incoming orders from the order generator. • Energy consumption data from the machines. • Application data related to order outsourcing sent to and received from POLIMI line. 2 POLIMI Domain In context to the scenario there are three main data types that originate from various sources within the POLIMI domain these include: • AGV data which is in the from of commands, static descriptors, and dynamic state of the AGV. Using 7 JSON payload structures • Application related event data generated from the operator interaction with order management cum AGV operator application. • Data coming from the production line. 3. Infrastructure Data Beyond IoT applications' data, there is a constant data flow related to network and computing resources usage. This is foreseen, to be handled, by the overall aerOS architecture design and the self-* components discussed elsewhere in this table.
Networking and service mesh	The first and biggest concern of the pilot is to achieve the necessary virtualization to seamlessly integrate aerOS functionalities. To achieve this, substantial efforts are being allocated. And on other hand the communication between the cloud and edge layer from the automation and monitoring side, since is critical for aerOS to successfully work. Thanks to the reliability of OPC UA protocols, this shouldn't be an issue.
Low-code programming tools	Currently the use case does not rely on any low code programming tools, however case for potential utilization of tools like Node-Red could be made.
Cybersecurity components	The Entrypoint domain will host the Identity Manager, which is implemented integrating based on Keycloak, connected with LDAP. KrakenD will be deployed in each domain for providing authentication and authorization.
Explainable and Frugal AI	With consideration of the diversity in the computational capability of IEs of the continuum the use of Frugal AI could be potentially explored especially considering the use of Low power devices like Raspberry and Industrial Edge devices for supporting with the CO2 emission related KPIs.
Portal	aerOS management portal will be deployed in the "aerOS MADE Domain." It will interface with KrakenD to provide access control to registered users.

7.2.2. Containerized Edge Computing near Renewable Energy Sources

The architecture of Containerized Edge Computing near Renewable Energy Sources contains the central site hosted in CloudFerro Cloud. It hosts common services such as authorization, shared storage and central aerOS components (1 per continuum). Each physical container is exposed as one aerOS domain based on a stand-alone K8s cluster with all required aerOS domain components integrated. Domains are connected via WAN network; the network architecture is documented at the end of this paragraph.

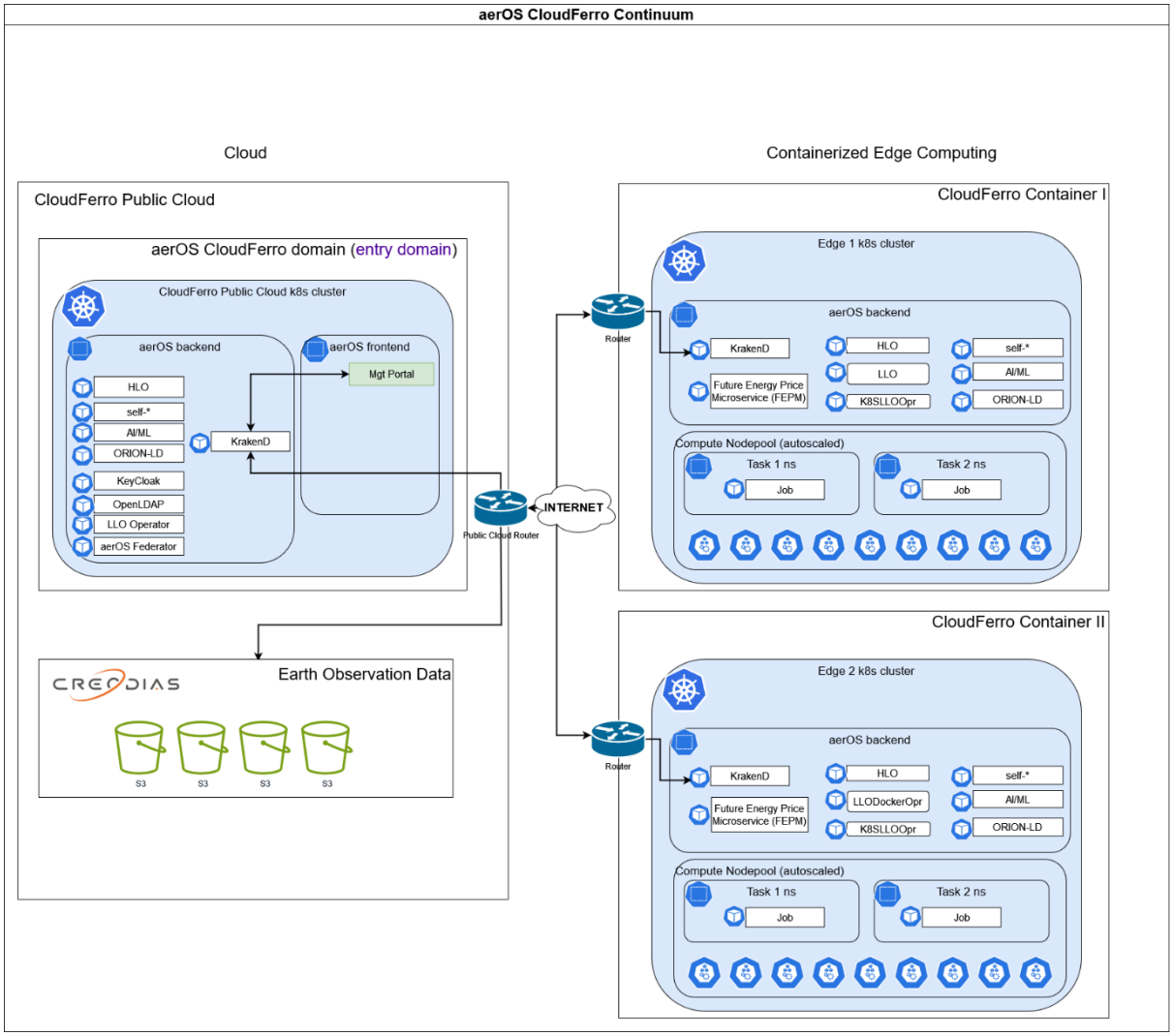


Figure 53. aerOS compliant high-level diagram for "Containerized Edge Computing near Renewable Energy Sources".

The following table enumerates aspects of aerOS that are used in pilots and their relation specifically to use cases of pilot 2. It also explains rationale of some decisions made that are related to these parts of aerOS architecture.

Table 7. Granular mapping of aerOS components within "Containerized Edge Computing near Renewable Energy Sources" pilot.

Item of aerOS RA	Mapping to pilot/vertical relevance and significance
Infrastructure Elements	In pilot 2 each IE will be a bare metal server exposed to aerOS as a Kubernetes node. These servers will be in containers placed near renewable energy sources.
Domains	Main considerations when considering domains for pilot 2 were location of compute and its availability. Pilot 2 includes two types of domains. Cloud domain of which there will be only one will be a Kubernetes cluster in a cloud environment (hosted by CF cloud). Due to its high availability and easily configurable public access, it will host management portal and be user's point of contact with aerOS (Entrypoint domain). Another type of domain will be a containerized edge node. Pilot 2 will include two domains of this type. These clusters will consist of bare metal servers configured as

	Kubernetes nodes and powered by renewable energy sources located near their container. These servers are better suited to running aerOS user workloads than VMs in the cloud due to usage of renewable energy and using less layers of virtualization. This is why user workloads will be run only in this type of domain, cloud domain will not have any IE to run tasks.
Self-* tool suite	Self-awareness tools will be used to monitor the state of the environments and discovery any problems with it. Self-orchestration will allow distribution of workloads across compute resources located in different sites. Edge Kubernetes clusters have Kepler add-on to measure energy consumption per workload.
Two-level orchestrator	HLO will be deployed both in the cloud domain to distribute workloads to other domains and in containerized edge nodes to select best IE in the domain to execute given workload. Since each containerized edge node will be a Kubernetes cluster K8S LLO will be used to execute workloads scheduled there by HLO.
Data Fabric	An instance of Orion-LD context broker will be deployed in each domain to facilitate usage of aerOS's inter-domain orchestration mechanisms. Information about compute resources available in computerized edge nodes will be accessible in cloud domain where it will be used to improve scheduling. Scheduling decisions will then be accessible to aerOS components in containerized edge nodes which will dictate what workloads will be ran on them. Energy availability information will be a crucial part of orchestration enabling data exposed by containerized edge nodes. Thanks to this HLO in Cloud domain will be able to prefer scheduling on nodes that have enough power available.
Semantic translation and annotation	Pilot must consider constant stream of metrics about network and resources usage gathered by Prometheus-exporters and aggregated by Prometheus instances both in containerized edge nodes and in cloud domain. Making use of this information will improve system's observability and orchestration abilities.
Networking and service mesh	Due to the nature of workloads that are expected to run in Pilot 2 inter-workload communication is not expected to be required. Regardless, to ensure proper functioning of aerOS in pilot 2 some networking considerations are made. Each domain's HTTP services is exposed though nginx-based ingresses. In cloud domain those ingresses are exposed through load balancer managed by the cloud provider are available to all users. In containerized edge nodes load balancer exposing ingresses is managed by MetalLB and are exposed only to intranet of CF. All services exposed on ingresses are secured with TLS using internal certificate. Each domain is connected to CF's intranet, containerized edge nodes fully through VPN and cloud domain only in a way that allows it to access other pilot 2 domains. Cilium is used as CNI solution for Kubernetes clusters acting as domains in pilot 2. By making use of network policies access to CF's intranet resources is restricted.
Cybersecurity components	Cloud domain will host IdM restricting access to aerOS endpoints. It will be based on Keycloak integrated with OpenLDAP will properly configured roles and permissions.

Explainable and Frugal AI	AI will be used to improve decision-making process of HLO.
EAT	Pilot 2 will make use of Grafana to expose the state of the system.
Trustworthiness	Second use case of pilot 2 focuses on multi-tenancy so trustworthiness is a crucial aspect here. Standard mechanisms of isolation (*runc* containers based on *cgroups*) used in Kubernetes clusters are not enough when dealing with untrusted workloads. Kata Containers project allows using lightweight VMs as an alternative to typical *cgroups*-based containers. By configuring them in *containerd* daemon used by Kubernetes to start containers and specifying appropriate runtime class, user's workloads can be started in an isolated and secure way in a same way as one would start any other workload on Kubernetes. Isolation of user workloads on network level will also need to be considered. To achieve that goal proper network policies will have to be configured. aerOS design and tools will help as necessary (network, IOTA, etc.) to make this possible, in synch with specific solutions proposed in pilot 2 (see above).
Portal	aerOS Management Portal will be deployed in the cloud domain. It will be secured using IdM solution based on Keycloak mentioned in Cybersecurity components.

7.2.3. High Performance Computing Platform for Connected and Cooperative Mobile Machinery

Pilot "High Performance Computing Platform for Connected and Cooperative Mobile Machinery" effectively harnesses the innovative aerOS architecture to significantly enhance the functionality and efficiency of agricultural machinery, focusing primarily on tractors and other field equipment, and can be extended to other application domains such as construction or forestry machinery as well. By transforming traditional farming machinery into interconnected, intelligent systems, the pilot not only enhances operational efficiency but also supports broader initiatives towards CO2 neutral farming. The comprehensive integration of advanced analytics and AI-driven insights ensures that each component of the pilot is perfectly aligned with overarching business processes, thereby enhancing both the performance and sustainability of agricultural operations. This pilot serves as a model for future agricultural innovations, highlighting the potential of integrated edge-cloud architectures in enhancing the efficiency and environmental sustainability of farming practices.

This pilot is designed to implement a sophisticated multi-domain approach utilizing the aerOS system across three distinct domains: the field domain, the on-premise computing domain, and the cloud computing domain. Each domain plays an important role in the overall integration strategy, as detailed in the accompanying diagram below. This diagram illustrates how the existing application modules and computational resources align with the aerOS concepts and components that provide the core functionalities of aerOS. The key components of the aerOS stack are depicted as blocks atop the hosting resources, serving as essential middleware. This middleware provides a critical abstraction layer that effectively bridges the heterogeneity in data and platforms. As a result, sensors, systems, and analytics can be seamlessly orchestrated across the edge-cloud continuum, and new network and computing resources can be readily integrated and federated.

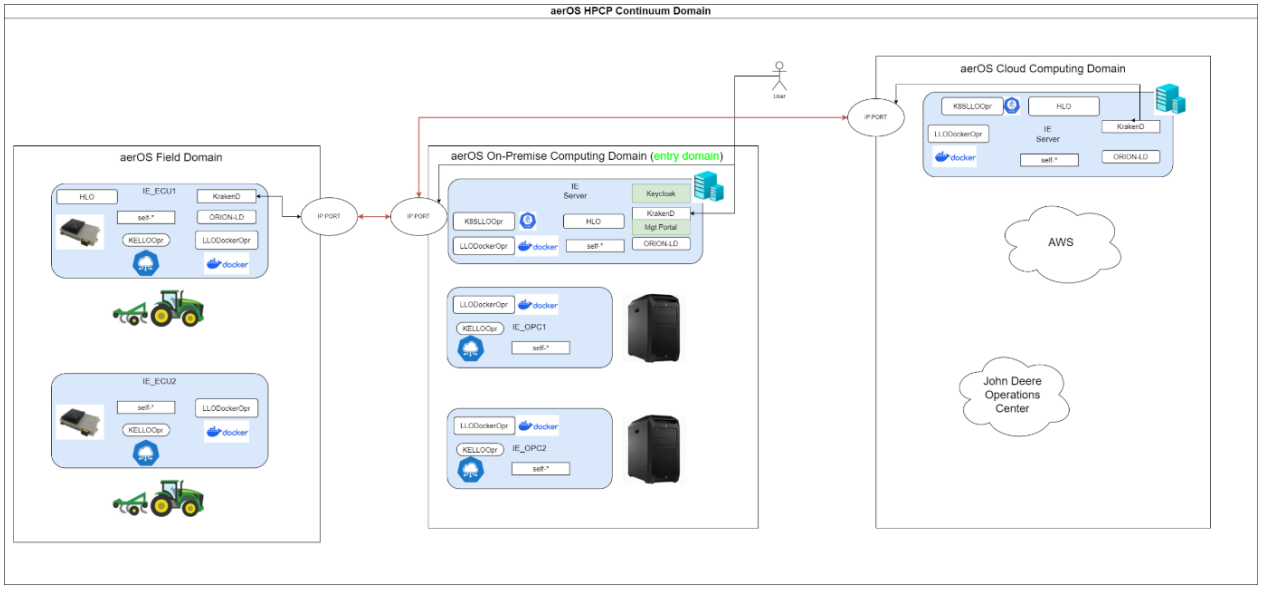


Figure 54. aerOS compliant high-level diagram for "High Performance Computing Platform for Connected and Cooperative Mobile Machinery".

The table below offers a detailed breakdown of the aerOS components involved, linking them to the corresponding pilot functions and modules. It also elucidates the rationale behind the available choices and the decisions made for final deployment.

Table 8. Granular mapping of aerOS components within "High Performance Computing Platform for Connected and Cooperative Mobile Machinery" pilot.

Item of aerOS RA	Mapping to pilot/vertical relevance and significance
Domains	This pilot employs a three-domain approach, incorporating the field domain, on-premise computing domain, and cloud computing domain, each chosen to leverage its specific strengths and functionalities, thereby enhancing the overall efficiency and effectiveness of the system. The "aerOS field domain" is directly interfaced with agricultural machinery, crucial for real-time data collection and immediate response capabilities essential in dynamic agricultural environments. It is equipped with high-performance ECUs and essential aerOS components, enabling swift data processing and decision-making directly at the source. Adjacent to this, the "aerOS on-premise computing domain" acts as the operational control center (i.e., Entrypoint domain). It hosts the aerOS Management Portal and core orchestration services, providing robust data security and higher data processing capabilities. This setup ensures efficient local processing and storage of data, crucial for managing operations efficiently and securely within the farmer's control, adhering to stringent data governance and privacy standards. The "aerOS cloud computing domain" serves as the scalable processing and storage backbone. It offers extensive computational resources and storage capacities necessary for heavy-duty data processing tasks and long-term data analytics. This domain facilitates advanced analytics, AI model training and execution, and integrates broader data sets from multiple sources, supporting comprehensive decision-making processes. The separation of this domain from the on-field and on-premise systems helps to align with regulations and compliance requirements, ensuring that sensitive data is handled appropriately while still benefiting from cloud-based capabilities.

	By distributing functionalities across these tailored domains, the system maximizes responsiveness, scalability, and security. The multi-domain approach not only enhances operational agility and precision in agricultural practices but also ensures that data governance is maintained across different levels of data control and ownership—from the farmer-controlled field and on-premise domains to the service provider or manufacturer-managed cloud domain. This strategic distribution of resources and responsibilities helps in complying with regulatory requirements while optimizing the agricultural operations.
Infrastructure Elements	The infrastructure elements deployed across the three aerOS domains are strategically configured to harness the unique strengths of each domain, enhancing the overall computational efficiency and responsiveness required for modern agricultural operations. In the field domain, tractors equipped with high-performance Electronic Control Units (ECUs) based on NVIDIA Jetson platforms form the core hardware that interfaces directly with agricultural machinery, enabling advanced computing capabilities right in the field. These ECUs provide the necessary power for sophisticated algorithm processing and real-time data handling, essential for immediate operational adjustments and decision-making at machine level. Adjacent to the field is the aerOS on-premise computing domain, which serves as the primary entrypoint to the aerOS architecture. This domain features powerful Linux workstations equipped with GPUs, handling the computational demands of on-premise operations. These workstations provide robust support for the management and coordination of field data, facilitating local data processing and analytics. Extending operational capabilities, the aerOS cloud computing domain includes virtual machines deployed in a cloud environment. These cloud-based virtual machines offer scalable processing power and storage solutions, crucial for managing and analyzing the vast data volumes generated from field operations. They support extensive data analytics that drive decision-making processes in modern agricultural practices, ensuring that data-driven insights are both timely and actionable. This configuration across the three domains – field, on-premise, and cloud – leverages specific hardware setups tailored to meet the unique computing requirements of each domain, enhancing overall system performance and efficiency. The deployment of NVIDIA Jetson-based ECUs, GPU-equipped Linux workstations, and scalable virtual machines in the cloud collectively ensure a resilient, efficient, and high-performing architecture. All these systems – ECUs in the field, workstations on-premise, and virtual machines in the cloud – are transformed and integrated as aerOS Infrastructure Elements through the deployment of the self-* package and core aerOS components, discussed in the next sections of this table.
IoT services definition	The "High Performance Computing Platform for Connected and Cooperative Mobile" pilot hosts the following services as part of the business case: • Data Processing and Analytics: Advanced services that process and analyze vast amounts of data collected from various sensors installed on mobile machinery. These services are crucial for extracting actionable insights, aiding in decision-making processes such as predicting maintenance needs or optimizing machine performance in real-time. • Control and Automation: Services that enable the remote control and automation of mobile machinery. This includes automating routine tasks and precisely

	controlling machinery operations, enhancing efficiency and reducing the need for manual intervention. The transition to aerOS is expected to significantly enhance these capabilities through several key benefits: • Flexible and Expandable Environment: aerOS's unique architectural framework allows for easy addition of new vertical or legacy applications, ensuring their full integration with existing data and services. • Integrated Data Privacy by Design: aerOS ensures that all user data is collected and managed securely, maintaining privacy and building trust in the system's operations. This is particularly important for handling sensitive information related to machine operators and farm management. • Dynamic Resource Scaling and Optimal Utilization: aerOS supports dynamic scaling of resources to meet changing operational demands without compromising system performance. This capability is essential for adapting to varied agricultural environments and operational scales, from small farms to large agribusinesses. • Service Resilience and Fault Tolerance: Leveraging aerOS's optimal resource utilization and dynamic service deployment, the pilot can maintain high service reliability and quick recovery in case of incidents, ensuring continuous operation and minimal downtime. • Efficient and Energy-Saving Resource Use: aerOS's orchestration services maximize the efficiency and energy preservation of resources by utilizing data monitored from IEs to make intelligent adjustments to operations, significantly reducing unnecessary energy expenditure and enhancing overall system sustainability. These enhancements position the aerOS architecture as a transformative solution for the "High Performance Computing Platform for Connected and Cooperative Mobile" pilot, driving improvements in agricultural productivity and operational efficiency.
Self-* tool suite	The aerOS architecture is set to incorporate a series of self-* components across its domains to significantly enhance the efficiency and automation of Infrastructure Elements (IEs). Planned components include self-awareness for continuous operational monitoring, self-orchestration to autonomously manage workloads, self-realtimeness for instantaneous data processing, and self-optimization to continuously refine performance while minimizing energy use. Additionally, there are prospective plans to integrate self-diagnose for proactive health monitoring, self-security to safeguard against cyber threats, and self-scaling to dynamically adjust resource allocation based on operational demands. These enhancements will not only improve system resilience but also ensure that the aerOS infrastructure can efficiently manage the complex and dynamic needs of modern agricultural operations with enhanced autonomy and scalability.
Two-level orchestrator	HLOs are deployed across the domains within the aerOS pilot continuum to enhance service placement and optimization. An HLO operates within an IE server in both the on-premise and cloud computing domains, and another is integrated within an IE in the field domain. This strategic placement across domains ensures efficient management and coordination of pilot IEs. KubeEdge enables K8s to function as the LLO across these domains, managing the container framework crucial for deploying and operating applications. This integration enhances the system's functionality and responsiveness.

	Furthermore, Docker is also utilized as an LLO alongside KubeEdge across the aerOS domains. Docker complements Kubernetes by providing a lightweight container platform that simplifies the packaging, deployment, and management of applications. This dual LLO setup with both Docker and Kubernetes ensures robust, flexible management of containerized workloads, enhancing the system's operational efficiency and scalability.
Data Fabric	In this pilot, a data fabric utilizing the FIWARE Orion-LD context broker is implemented across each domain to manage and store all relevant pilot data. The Orion-LD brokers ensure that information is consistently available and easily retrievable across domains. Additionally, aerOS components within the Data Dabric are tasked with ingesting data from various devices and applications, processing and transforming this data in alignment with a predefined model. This structured approach facilitates efficient data integration and utilization throughout the system. Information about the data is detailed in the next table row.
Semantic translation and annotation	The pilot leverages a comprehensive data management strategy aligned with aerOS data principles, encompassing various types of data crucial for operational and analytical processes: 1. Pilot Sensor Data Data Generation: The core sensor data includes streams from forward and backward cameras, along with detailed machine configuration and operation data. This rich data set is essential for driving real-time decision-making and performance optimizations. This primarily machine-generated data is provided in formats that facilitate rapid access and analysis, supporting immediate and actionable insights. Metrics and Metadata: Metrics include operational parameters such as speed, temperature, implement depth, energy consumption and battery state of charge (SoC) levels. Metadata encompasses the name of each component or sensor, application-specific data like application names and broker topics, availability, and other critical diagnostics that provide a comprehensive view of machine health and operational status. 2. Pilot Application & Context Data Dynamic Interaction: AI algorithms process real-time sensor data to generate predictions for steering angles, velocity changes, adjustments in implement depth, and further machine configuration adaptations. These predictions are dynamically integrated into the operational processes to enhance machine efficiency and effectiveness. Data Recognition and Correlation: It is crucial for all data, especially AI-generated predictions and real-time operational settings, to be syntactically and semantically recognized by the system. This ensures that AI insights are effectively integrated and utilized to optimize performance and operational efficiency. 3. Infrastructure Data Continuous Data Flow: Data related to the usage of network and computing resources supports the broader aerOS architecture, facilitating smooth and efficient system operations. Management and Automation: This infrastructure data is managed and automated through the overall aerOS architecture and self-* components, ensuring optimal resource utilization and system performance. Through this structured data management approach, the system effectively processes and utilizes a wide array of data types – from high-resolution images and sophisticated AI predictions to real-time machine telemetry and operational

	metadata. The integration of advanced data management tools within the data fabric ensures the system's ability to intelligently process, act upon, and leverage diverse datasets, supporting comprehensive decision-making and enhancing agricultural productivity.
Networking and service mesh	Each domain within the aerOS architecture of this pilot is meticulously equipped with a comprehensive suite of core services that form an integrated service fabric. This fabric encompasses essential functionalities such as orchestration, federation, and AAA, all seamlessly coordinated through the LLOs, which manage lower-level operations. Moreover, robust VPN connections are integrated to facilitate effective communication within each domain and secure the exposure of the aerOS API across domains, addressing network stability challenges typical in remote agricultural settings. These connections, supported by failover solutions, ensure continuous and secure data flow. Furthermore, Container Network Interface (CNI) plugins like Cilium enhance intra-domain communication, while several Containerized Network Function (CNF) components maintain secure API functionality across the system. The orchestration services play a pivotal role by enabling dynamic resource allocation and scaling across the continuum, which significantly enhances system performance and resource utilization. This setup not only ensures operational efficiency but also maintains high security and connectivity standards, critical for the seamless integration and management of distributed resources in the agricultural domain.
Cybersecurity components	The cybersecurity framework within the aerOS architecture utilizes robust IdM systems, including Keycloak and LDAP, hosted in the entrypoint domain. These systems manage authentication and authorization, defining user roles and permissions to ensure secure access to system resources. Roles with related permissions are being specified.
Explainable and Frugal AI	Frugal AI is employed to optimize resource utilization and operational efficiency within the aerOS architecture. By implementing frugal AI techniques, the system can effectively perform data analysis and decision-making processes with minimal computational resources. This approach not only conserves energy but also reduces costs, making it ideal for the scalable and sustainable management of agricultural operations. Frugal AI ensures that even with limited bandwidth and processing power, the pilot can maintain high levels of accuracy and functionality in real-time data processing and analytics.
EAT	The aerOS cloud computing domain supports scalability and robust data handling capabilities, essential for the extensive analytics that drive decision-making processes in modern agricultural practices.
Portal	aerOS management portal will be deployed in the “on-premise computing domain”. It will interface with Keycloak to provide access control to registered users.

7.2.4. Smart Edge Services for the Port Continuum

The Smart edge services for the Port Continuum pilot is led by EUROGATE Container Terminal Limassol (ECTL), the largest terminal in Cyprus, handling more than 90% of the island’s gateway container cargo. Technically, ECTL is supported by partner Prodevelop (PRO) and the Cyprus University of Technology (CUT). The terminal equipment consists of 36 Straddle Carriers (SC), 5 Ship-To-Shore (STS) cranes, and other types of container handling equipment. Within this context, ECTL is in the process of digitizing and automating more operations processes. aerOS is the starting point for creating scalable IoT infrastructure that the Terminal can build and expand in the future. It is expected that more than two hundred devices will be connected in the next 5 years. The pilot is divided in 2 use case scenarios that, jointly, pursue the achievement of four objectives: (i)

to provide a digital platform for the terminal which can improve the traceability of its assets in the yard, (ii) to support a heterogeneous set of Infrastructure Elements compliant with aerOS, from the lightweight IoT gateways up to the most-capable cloud servers, (iii) to provide a very accurate predictive maintenance service based on frugal AI models that are embedded in the aerOS stack, and (iv) to provide a computer vision solution that can be inferred from the edge without requiring very high bandwidths.

Even having enough data, the inference process of the most suitable predictive maintenance and video object detection services required the clear understanding of the computing capabilities of the Infrastructure Elements (IEs) of the pilot. The below diagram sketches the expected architectural deployment of Pilot 4 use case scenarios. As it can be seen from the Figure 55, the pilot plans to differentiate up to 4 different aerOS domains.

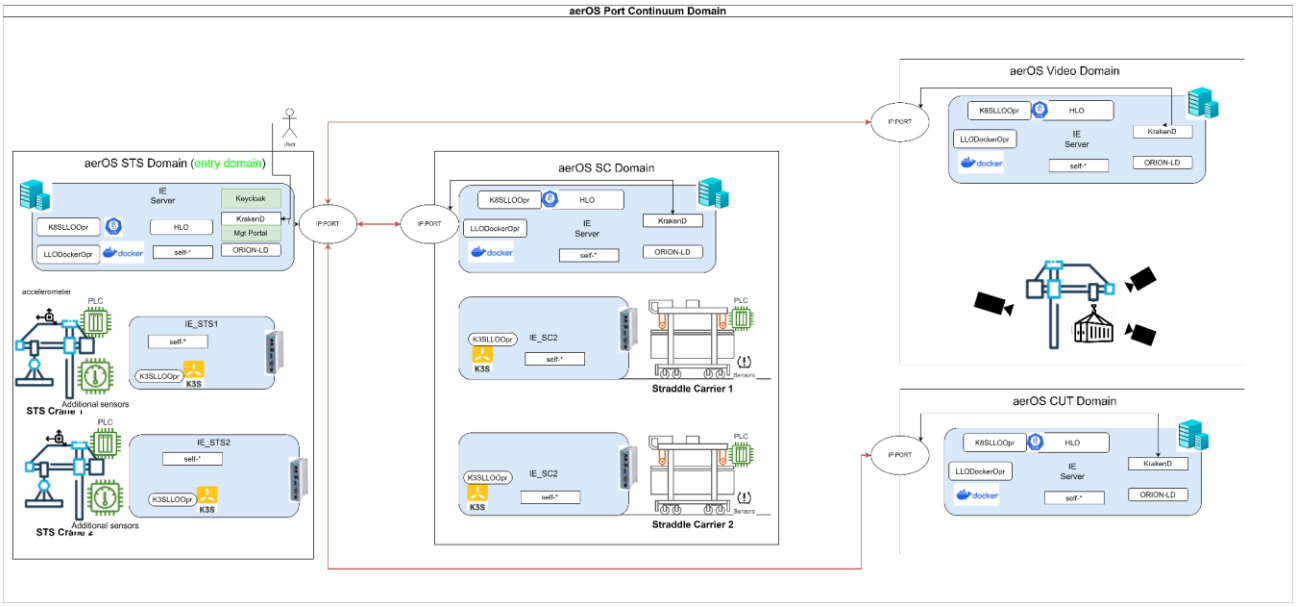


Figure 55. Port Continuum use case scenarios of aerOS.

The table below offers a detailed breakdown of the aerOS components involved, linking them to the corresponding pilot functions and modules. It also elucidates the rationale behind the available choices and the decisions made for final deployment

Table 9. Granular mapping of aerOS components within pilot 5 "Smart Edge Services for the Port Continuum"

Item of aerOS RA	Mapping to pilot/vertical relevance and significance
Infrastructure Elements	The pilot will be formed by multiple heterogeneous IEs. They can be split into the three tiers of IoT-Edge-Cloud continuum, that redound in the establishment of four domains in the pilot. First, sensors and PLCs from the cranes will be connected to far-edge IEs, in this case Siemens IoT2050 gateways. There will be 6 IEs of this kind, deployed across the monitored 4 Straddle Carriers and the 2 STS cranes. The main characteristics of these IEs are the following: Intel 1.1GHz CPU, RAM 2GB, 16GB eMMC, 2xRJ45 Ports, 1xSerial Port. In addition, the video streams captured through the 3 IP cameras per crane will be inferenced at the edge side by the other two far-edge IEs. In this case, a couple of Jetson Orin Nano Developer Kits, with the following specifications: NVIDIA Ampere architecture with 1024 NVIDIA® CUDA® cores with 32 tensor cores, 6-core Arm Cortex-A78AE v8.2 64-bit CPU, 8GB 128-bit LPDDR5 68 GB/s, and supports for external NVMe. Then, two IEs as virtual machines managed at EUROGATE premises are being installed. The first one will oversee data collection and model training for the

	predictive maintenance use case scenario. The second one will store at first the video pre-processed from the IP cameras for further ML training through computer vision. Finally, a final IE located at the aerOS CUT domain (see the next row below) with distributed and parallel GPU processing capabilities will be oversee the heavy training processes of the computer vision models. A Dell PowerEdge R6525 server with 48 AMD physical cores 2.3GHz-3.2GHz turbo, 256MB L3 Cache, 128GB DDR4 RAM, 6 HDDs (10K SAS 1.2 TB each), and an NVIDIA Tesla T4 16GB GPU constitutes this IE.
Domains	The pilot differentiates up to 4 different aerOS domains: For use case scenario 1, both aerOS STS domain, as well as aerOS SC domain are expected to collect data from sensors and the machine's PLC to perform basic pre-processing. In addition, the aerOS STS domain will act as the entry domain. Due to making use of a high-end IE that manages all the distribution orchestration services from aerOS. For use case scenario 2, two additional aerOS domains will be connected. The aerOS Video domain will oversee video capture and collection, while aerOS CUT domain will be in charge of development of ML models for predictive maintenance and computer vision models.
IoT services definition	Different IoT services are currently deployed among the on-site IEs, including time-series data acquisition, data collection, and data analysis. All these services are manually deployed, without considering any intelligent and distributed orchestration. Furthermore, the IP cameras are recording the video streams in an embedded storage disk, which will be replaced by a real-streaming service deployed through the aerOS continuum. This will permit the transfer of those videos directly to the connected IEs of the continuum. All these data will be correlated with the API service that will retrieve operational data from EUROGATE's TOS, and CMMS. Finally, two different data analytics or AI/ML services are needed for data training and further inference (for both use case scenarios of the pilot).
Self-* tool suite	In principle, it is envisioned that all the IEs of the port continuum will have the aerOS self-* core components, i.e., the self-awareness (who collects and sends data on the current state of the IEs), the self-diagnose (who send alerts when it detects health problems on the IE), the self-orchestrator (who determines whether to generate and launch reorchestration warnings due to IE state), and the self-* API (who exposes a single point of connection to the self-* suite of each IE, being able to retrieve certain aspects of management, such as dynamic rules to be parametrised in the self-orchestrator module).
Two-level orchestrator	The IoT services described above are expected to be intelligently distributed by means of the two-level orchestrator. On the one hand, the core components of the HLO will be used at the entry domain (i.e., the STS domain). This will analyse the most optimum allocation of the IoT services and associated service components along the different IEs and domains, thanks to IE state information collected in the HLO Data aggregation system, who in turn, gets that information from the self-awareness module. Next, once the HLO allocation engine has defined the optimum allocation of the IoT services (who will include some clauses regarding the domain under scope), the HLO deployment engine will connect either to its underlying LLO from the STS domain (for predictive maintenance use case scenario associated services), or to the HLOs of the other domains (for any other associated service of both use case scenarios in pilot 4).
Data Fabric	The most relevant application of Data Fabric in pilot 4 will be the transfer and distribution of data about the continuum (IE state tobe recorded and circulated through the Data Fabric).

Semantic translation and annotation	Pilot data are streaming data, which are already formatted in the standard TIC 4.0 data model. Semantic annotation and translation is not considered an essential service for this pilot.
Networking and service mesh	The aerOS network capabilities support self-contained functionality for the different Pilot 4 domains, flexibly adapting the connectivity requirements among these domains. Hence, Ingress controller and MetalLB components are used for providing incoming traffic to the port continuum domains. Cilium (and Cilium cluster mesh) is also used for allowing the deployed IoT services to communicate seamlessly on top of existing network infrastructure from the different IEs of the pilot. This will also allow to seamlessly integrate new IEs in any domain, so that new STS cranes or straddle carriers are easily registered and monitored as part of the continuum. For the networking security, the selection of either WireGuard or the actual VPN managed by EUROGATE is still under consideration.
Low-code programming tools	Node-red is already integrated in pilot.
Cybersecurity components	Since the pilot collects information that directly or indirectly affect the business operations of EUROGATE, security is critical. Hence, the cybersecurity components of aerOS (both Keycloak authentication, and KrakenD authorization) are needed to control the access to the deployed services along the IEs of the port continuum, as well as to the data product associated to IE state metadata. In addition, enforcing data privacy is also paramount for the use case scenario 2, in which EUROGATE workers can be observed through the video streams captured by the IP cameras set at STS cranes.
Explainable and Frugal AI	Both, pilot 4 use case scenarios rely on its high-end need over accurate ML models. On the one hand, regression models are being trained for the predictive maintenance use case scenario, while object detection classification models are being trained for the computer vision use case scenario. From both use case scenarios, the predictive maintenance might be approached from a frugal and explainable AI approach. On the one hand, considering that an initial trained model has been validated, the newer datasets for further reinforcement would only require around 7 KB/s per crane, which would lead to newer training datasets of less than 1 GB. On the other hand, there are several parameters that can motivate the identification of CHE failures in advance. Consequently, the inferenced model should not only alert about the failure, but also provide explanations about the reasons/features that led to those predictions.
EAT	Although the IE performance is well monitored by means of the self-awareness, self-orchestrators, self-diagnose, benchmark tool from the management portal, they all refer to the HW processing resources. Pilot 4 demands several processes related to data acquisition, data collection, and data storage, which requires the deployment of heterogenous software services. The Embedded Analytics Tool will be used to deploy serverless functions aiming at monitoring the proper functioning of all the data processing services of the pilot to allow maintainers to be aware of potential malfunctions along the data pipeline. To do so, performance metrics will be scraped by Prometheus, visualised in Grafana, and alerts will be set up in the alert manager.
Portal	The aerOS management portal is conceived as the graphical entrypoint to the port continuum domains. Hence, all the currently supported functionalities for the portal will be needed: welcome page connected to the Keycloak authentication service, and all the connected orchestrations tabs, such as the domain and continuum tabs for creating and connecting all the pilot 4 domains, the deployment tab for enabling through the HLO the deployment of pilot-specific IoT services and its surrounding service components among the multiple IEs of the pilot, and the benchmarking tab for being informed about the IE states in a more visual way. It should also be noticed that the pilot will also have its own UIs related to the real-time movement of

	EUROGATE CHEs across the terminal, the low-code tool used for data acquisition from CHEs, or the graph-engine tool for exposing operational and maintenance advanced analytics.

7.2.5. Energy Efficient, Health Safe and Sustainable Smart Buildings

The goal of "Energy efficient, Health Safe and Sustainable smart buildings" pilot is to validate, and demonstrate, gains of the aerOS architecture in an edge deployment for energy efficient, sustainable, flexible, and health-safe smart buildings. The extension and instantiation of existing IoT services, according to aerOS reference architecture, is expected to automate operational efficiency, which is necessary to reduce costs as energy consumption is put in the spotlight, and workspace and rooms management towards a human health-safe environment. The existing IoT solution, prototyped by COSMOTE R&D and which is based on open-source software and commercial hardware, is being extended and transformed, as part of aerOS, with the aim to build an IoT, vendor-agnostic, solution which can dynamically adopt variety of IoT technologies and adapt to changes in infrastructures using automated orchestration solutions. The aerOS Meta-OS deployment is meant to provide a solution which can be, on demand, rapidly expanded on additional premises' workspaces and be deployed on top of existing infrastructure, enforcing autonomous and decentralized decision-making at the edge.

A high-level view of the design and an overall projection of aerOS architecture on top of pilot's integrated resources is presented in the following diagram. This diagram presents the correspondence of existing resources to aerOS concepts and the set of components that are mapped on top of these resources to implement aerOS core functionalities. The most prominent aerOS stack components are represented as blocks on top of hosting resources. This makes evident how existing resources finally transform to aerOS elements. The overall goal is to employ aerOS architecture to stand as a unique abstraction layer and offer an adaptable solution that can bridge heterogeneity (data and platforms), so that sensors, systems, and analytics can be orchestrated in the IoT edge-cloud continuum, and new network and computing resources can be easily federated.

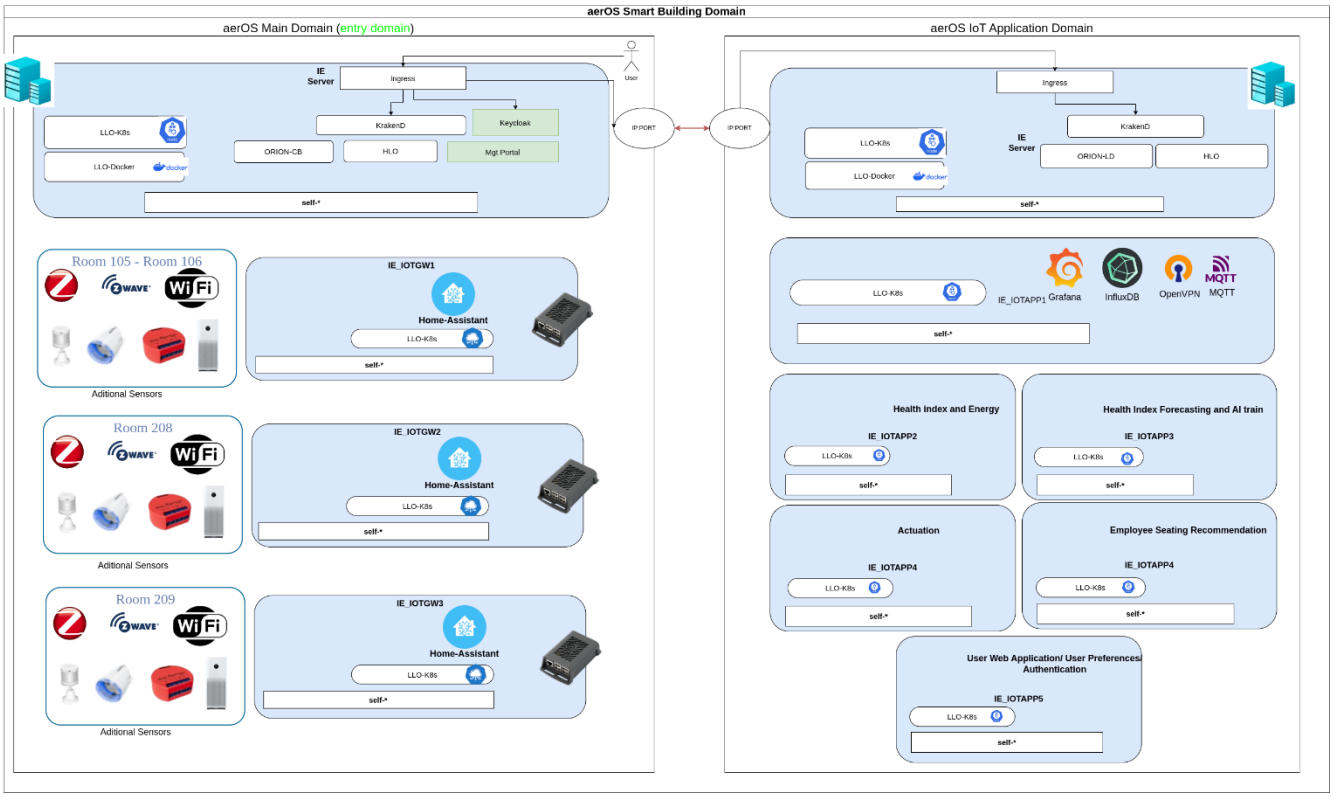


Figure 56. aerOS compliant high-level diagram for "Energy efficient, Health Safe and Sustainable smart buildings" pilot at COSMOTE.

The following table provides a more granular enumeration of components and offers a more complete overview for the reasoning behind choices and mapping of aerOS concepts for deployment within the pilot.

Table 10. Granular mapping of aerOS components within pilot 5 "Energy Efficient, Health Safe and Sustainable Smart Buildings"

Item of aerOS RA	Mapping to pilot/vertical relevance and significance
Domains	The acknowledgement that the placement as well as the life cycle, management, and configuration of the components in each domain may differ and should adhere to distinct operational and service-level standards has led to the decision for the provision of two distinct domains, one for IoT and another for Applications. Nonetheless, the flexibility and plasticity that is provided based on the HLO-LLO placement heuristics, makes it easy to incorporate everything within a single domain, the Pilot aerOS Main Domain, if performance analysis dictate such a transition. The "aerOS Main Domain" integrates the IoT components that need to directly interact with IoT devices, either gathering data or instructing actuation events, and the "aerOS Application Domain" which hosts applications more closely related with storage, representation, analytics, user access and AI and recommendation applications.
Infrastructure Elements	The infrastructure elements integrated in each of the two aerOS domains are a collection of computing resources and fall under two main categories. • The IoT Gateways which are based on the robust and affordable "UP" board series hardware platform (https://up-board.org/up/specifications/), which is a credit-card sized board with high performance and low power consumption. These are all included in the "aerOS Main Domain" and are closely engaged with the devices' integration task. They follow the Raspberry Pi2/4 GPIO communication specification, ensuring full compatibility with Pi shields/sensors. They are equipped with a highly customized Debian based OS -loaded with only bare minimum services and applications to minimize CPU and RAM footprint. They are easily integrated in a K8s cluster ensuring the containerized nature and requirements of aerOS. • The back end virtual machines which are deployed over a virtualized/cloud infrastructure, constituting an affordable and robust platform. These are deployed within the virtual infrastructure manager (VIM) running in the NOC of Cosmote Academy. All computing resources of both categories are "transformed" and integrated as aerOS Infrastructure Elements based on the self-* package which is deployed on top of them and is discussed in next section of this table.
IoT services definition	"Energy efficient, Health Safe and Sustainable smart buildings" pilot already hosts the following IoT services as part of the business case: • Sensor Management: Services responsible for managing and interacting with various sensors deployed throughout the building. • Device Integration: Services that facilitate the integration of diverse IoT devices and systems within the building ecosystem. This involves ensuring compatibility, communication protocols, and data interoperability between different devices from various vendors. • Data Processing and Analytics: Services that process, analyze, and derive insights from the data collected by IoT sensors.

	• Control and Automation: Services that enable remote control and automation of IoT devices and systems within the building. The benefits, related to technical expanded capabilities, that come by delegating their operation to aerOS, are the following. • Flexible and Expandable environment ready to host new vertical IoT applications and provide full integration with existing data and services. It is expected to bypass obstacles that prevent technical integration which can expand and seamlessly integrate a variety of resources. Multiple IoT vendors and solutions tend to make broad alignment and exploitation almost impossible. Applying the aerOS architecture, that stands as a unique abstraction layer, the end-to-end integration can be achieved easily embracing the strategic IoT sourcing of each enterprise to offer an adaptable solution that can bridge heterogeneity and thus enable so that sensors, systems, analytics work in sync. • Enable data produced by an extensive number of IoT sensors, deployed within the Smart Buildings ecosystem, to be valid and reusable beyond narrow environment (application and geographically wise) set by their connected vendor. Render the generation and processing of huge amount of data as a valid and asset which can support the distinctive infrastructure characteristics of each building and rationalize the autonomous and decentralized decision-making at the edge with the use of the aerOS nodes intelligence. • Integrate by design privacy capabilities integration regarding data management and exploitation. Based on these capabilities introduced by aerOS, user data will be collected and manipulated in a secure manner that protects personal information and secure applications which enable smart decision making and recommendations based on that data will be developed. • As automation and expandability are key to any IoT solution especially when targeting reliable autonomous operation, it is important that aerOS provisions for a deployment can which can dynamically scale-in and out (e.g., add/remove rooms, floors, and buildings) and integrate in a bigger ecosystem (e.g. federate with other smart buildings solutions). With the aerOS architecture, the most optimum deployment and utilization of resources, as well as recovery in case of incidents is assured. • Most efficient and energy preserving use of resources as supported by aerOS orchestration services which will consider monitored IEs data.
Self-* tool suite	Self-awareness and self-orchestration components are mapped for deployment in each IE.
Two-level orchestrator	HLO, as part of aerOS core services, is mapped for deployment on both domains to support efficient optimization of service placement on top of integrated IEs. The underlying container management framework for all VMs, which are part of the application domain is K8s and thus the K8s LLO will be deployed and used. The “UP boards” are integrated with the support of KubeEdge and thus K8s LLO is also used for orchestration decisions enforcement. Of course, since LLO is an (extendable) component, designs for further system extension and additional rooms integration, have the freedom to integrate existing lower-capabilities RPIs with the support of Docker LLO which has been also deployed although not used.

Data Fabric	An instance of FIWARE Orion-LD context broker will be deployed in each domain to host all pilot data encompassing information regarding IoT sensors metrics and activation status, AI-generated insights, seating recommendations for employees, premises information. These are detailed in the next table row and are stored, in semantically annotated format according to NGSI-LD specification, in the Orion-LD broker. A knowledge graph which can host all information and correlate entities is designed based on the NGSI-LD standard. This will encompass information and relationships among all above mentioned pilot data. aerOS components, provided as part of the data fabric, will be deployed to consume data published by devices and applications and transform them according to the above-mentioned designed model.

Semantic translation and annotation	Within the pilot applications there is a production of a large volume of data, stream data, originating form a variety of installed IoT devices. Data coming from these devices are both metrics and devices metadata, which can also be valuable for decision algorithms. Recorder data are machine generated and are initially stored in timeseries format.
	
	Metrics coming from sensors are:
	
	• Temperature
	• Luminosity
	• Air Quality, CO2, MQ135 gases
	• Motion sensing
	• Door open/closed events
	• Event triggering (via sensor's onboard button)
	Related metadata include:
	
	• Protocol-specific parameters (LoRaWAN)
	• Link quality parameters.
	  ○ RSSI
	  ○ SNR
	• Transmission's Time on Air
	• Name of the sensor that transmitted and name of the serving gateway.
	• Application-specific data (application name, broker topic)
	• Availability, battery, device movement
	Data collected can have different semantics of represented information (e.g. units and scaling), have different naming conventions based on the providing device and are provided in various formats (ascii text, raw/hex format needing decoding).
	
	There is also data concerning entities that do not exhibit great variances over time, as for example buildings, rooms, and employes.
	
	Additionally, there are the expected outcomes of the AI algorithms and the recommenders.
	
	Beyond the obvious fact that all these data need to be "recognized" by all existing and future applications, they also need to be correlated (linked) to benefit from linked-data advantages.
	
	All the above will be addressed and automated with the integration of the product manager tools, presented in data fabric.
	
	Of course, beyond IoT applications' data, there will be a constant data flow related to network and computing resources usage. This is foreseen, to be handled, by the

	overall aerOS architecture design and the self-* components discussed elsewhere in this table.
Networking and service mesh	The main concerns of the pilot regarding networking capabilities are, on one hand, to enable IE communication within each domain and on the other hand to securely expose aerOS API on each domain. The first part of the above statement is handed to the CNI plugins for each underlying K8s cluster. Cilium is a definite selection to go for IEs based on VMs, but Calico is also tested as it might offer a possible better integration with KubeEdge. For the exposure of each domain several CNF components are required. The ingress to provide a single and controlled entrypoint for the domain's APIs which will integrate certificates for secure and private communication (SSL/TLS), the API gateway which will translate and route requests to underlying aerOS services (e.g. Data Fabric, HLO), a certificate manager to manage (even self-signed) certificates and the load-balancer to advertise a routable IP are selected for deployment. Finaly network policies will be defined and used to restrict broad access to privacy and operational critical services.
Low-code programming tools	The IoT solution, which is responsible for IoT devices integration, and is being migrated to perform as an aerOS orchestrated service, is already providing low-coding functionalities. It provides a graphical interface capable to relate, with the support of a flexible and intuitive GUI, triggered events and sensors input and in response produce and dictate devices actuations. Beyond this, aerOS auxiliary tools, specifically the Node-Red flow wiring tool is part of the discussion related with the interaction of the energy efficient AI algorithm and personnel placement with the core automation system. Additionally, AsyncAPI, is used to model interfaces of existing asynchronous and event-driven communication based on MQTT messaging protocol.
Cybersecurity components	The Entrypoint domain will host the IdM which is implemented integrating Keycloak and LDAP for providing authentication and authorization. Roles with related permissions are being specified.
Explainable and Frugal AI	Explainable AI has a prominent role on the aerOS supported application that is being developed. As part of the application functionality is to recommend to users where they should be sited, it is significant to be able to present to users the motivations and logic beyond decisions.
EAT	From EAT, Grafana is of interest for the pilot as it is already a functional component of the smart building management system.
Portal	aerOS management portal will be deployed in the "aerOS Main Domain". It will interface with Keycloak to provide access control to registered users.

7.3. Mapping and alignment of aerOS RA to the European CEI continuum

It is relevant to note that aerOS is not alone in the quest for an architecture to cover the cloud-edge-IoT continuum. Two Coordination and Support Actions (CSAs) were funded in 2021 to cluster the communication and collaboration of the RIA projects coming out of the DATA-01-02 (cognitive cloud), DATA-01-03 (swarm computing) and DATA-01-05 (meta operating systems). There, several projects collaborate to solve the issues

and come up with a joint approach for handling orchestration, data, AI, cybersecurity, trust, network and monitoring in those environments.

In that regard, as it was succinctly described in deliverable D6.2., aerOS plays a fundamental role, being the leader project in defining the architecture or service orchestration in EUCEI continuums. While it contributes to all technological Working Groups, WG5 is moved forward rooting on aerOS ideas.

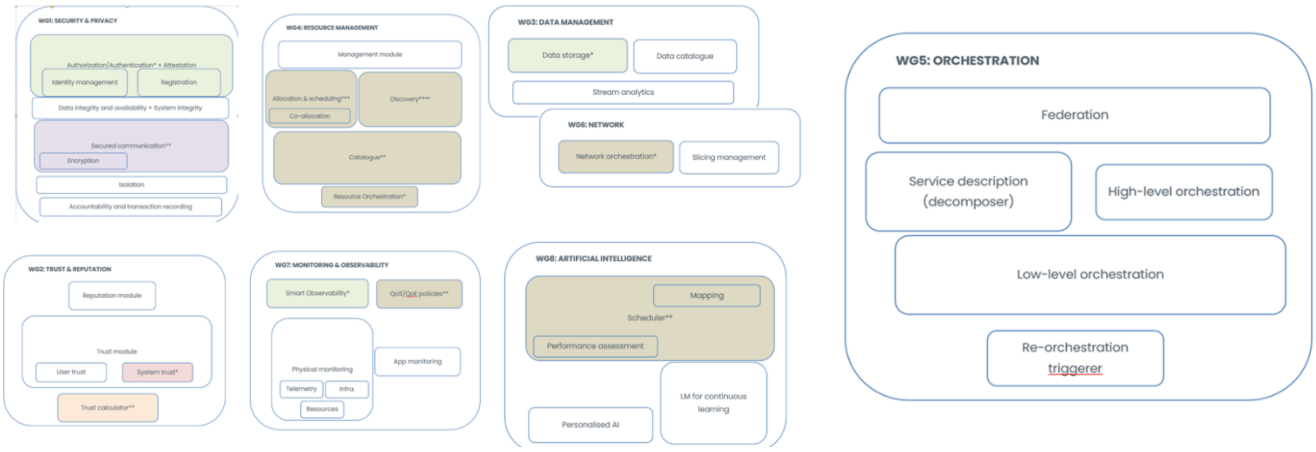


Figure 57. EUCloudEdgeIoT Working Groups – at the right, WG5 aerOS leadership.

aerOS is innovating in the computing continuum field, providing its own perspective to diverse areas, as it has been demonstrated along this document. Notwithstanding, the proposed architecture complies with the minimum features being promoted by the prominent pre-standardisation activity moved forward by the mentioned CSAs. The table below indicates how aerOS addresses the relevant concerns in the field (expressed by an equilibrium of demand-side and supply-side entities).

Table 11. aerOS architecture mapping and beyond EUCEI common blocks

Feature	Minimum coverage	aerOS approach
WG1 – Security block		
Authentication / Authorization / Accessibility	Allowing the access to the system to users according to a predefined role, plus ensuring all agents are allowed to be part of the system	aerOS catalogue based on OpenLDAP for registering all users and their roles, enabling RBAC. aerOS IAM based on Keycloak. aerOS Secure API Gateway based on KrakenD.
Data security, integrity and availability	Secure persistence of data, including proper, selected access to the catalogue of data sources.	aerOS Data Fabric implements access control policies based on roles expressed in *FOAF* and usage of OpenLDAP.
Secured communication	Needed to ensure the security and safeness of all communications between the different components (across all WGs).	aerOS employs TLS encryption for its exposed communications based on cert-manager. VPN connections based on WireGuard tool. aerOS Secure Gateway based on KrakenD.
Isolation	Kernel namespaces for workloads isolation, within each domain. based on their access policies defined	aerOS makes use of the intrinsics mechanisms provided by cloud-native technologies such as Cilium, KubeEdge or Kubernetes.
Accountability	Tracking and traceability of actions (aspect proposed by aerOS).	IOTA tangle is customized to introduce certain exchanges into an immutable DLT, using a Directed Acyclic Graph.

How aerOS goes beyond	aerOS establishes certain elements in specific positions in the architecture to ensure decentralization. Additionally, a methodology for inserting security and privacy in CI/CD procedures, together with tools and manuals to guarantee DevPrivSecOps	
WG2 – Trust and reputation		
Trust module	Element monitoring different elements regarding trustworthiness in the network	aerOS Trust Agent collects attributes from IEs, leveraging self-awareness module.
Trust calculator	Algorithm for establishing a quantitative index of trustworthiness of a node (proposed by aerOS)	aerOS Trust Manager collects all data submitted from above mentioned agents and employs the trust score algorithm to calculate a health and trust score of every node.
How aerOS goes beyond	The IOTA tangle introduction in the architecture provides a new level of scalability, being more robust as new IEs join the continuum, and also allows the incorporation of messages or actions that would qualify for "traceable" or "immutability worthy".	
WG3 – Data Management		
Data storage	Persistence of IoT data for usage in the continuum	aerOS DF integrates Orion-LD Context Broker, which in turn, in aerOS, can integrate with Mintaka
Data Catalogue	Exposure of the catalogue of data available to be accessed in a continuum, including associated metadata.	aerOS employs DCAT under a sophisticated LOT methodology. OpenLDAP is connected to NGSI-LD, which follows standardized API and data formats.
Stream Analytics	Capacity to gather and process data in streams	SemAnn and SemTrans handle annotation and translation in stream on-the-fly data circulation. aerOS EAT (based on OpenFaaS) allows swift handling of stream data.
How aerOS goes beyond	*aerOS includes a knowledge graph to model all aerOS entities and their relationships and is based on linked data technology as modelled over NGSI-LD concepts*	
WG4 – Resource management		
Management module	Resource management, including addition, removal, configuration options	aerOS Portal includes a service for handling the Infrastructure Elements and domains
Resource catalogue and discovery	Capacity to monitor status of the resources that form the continuum, being able to retrieve their name, characteristics, etc.	aerOS knowledge graph, built on top of NGSI-LD (ETSI GS CIM 006) and persisted and federated with domains' connected Orion-CBs
Resource allocation & scheduling	Services scheduling and resources allocation based on continuum availabilities	aerOS self-* capabilities introduce resources usage thresholds which orchestrator considers most efficient placement.
How aerOS goes beyond	*Continuous resource monitoring enforces services migration before resources failure due to overload conditions.*	
WG5 – Orchestration		
Federation	(proposed by aerOS) Capacity of different domains, or grouping of nodes, to be part of a continuum although belonging to different entities, and keeping their separation.	Leverage aerOS data fabric provisions. Continuum status, encompassing IEs capabilities and availabilities and hosted services deployment, rooting on NGSI-LD cross registrations (see 5.4.3.3).
Service description	(proposed by aerOS) Describing -in a homogeneised format- services of different nature (servers, one-shot…)	aerOS employs a customized version of TOSCA to express the services to be deployed. aerOS proposed a novel decomposition in service components following a defined format.

High Level Orchestration	(proposed by aerOS) Element that holds the allocation intelligence rooting on the continuum state.	aerOS HLO.
Low Level Orchestration	(proposed by aerOS) Element that performs orders towards executing workloads in a targeted node.	aerOS LLOs corresponding to different container management frameworks.
Re-orchestration trigger	(proposed by aerOS) Capacity of automatically, intelligently re-allocate a workload based on dynamically configurable rules.	Self-orchestrator module within aerOS self-* tool suite.
How aerOS goes beyond	aerOS proposes the novel capacity of dual decentralization: (i) decentralized execution of workloads and (ii) decentralized decision – that can take place where the trigger occurs.	
WG6 – Network		
Network orchestration	Capacity to handle via software certain network configuration, controlling the access and being able to route traffic among components in the continuum.	aerOS Secure API Gateway based on KrakenD. Service components interconnection using cloud-native network solutions. Components such as Cilium and MetalLB.
Slicing management	NFV deployment and software-controlled components in the distributed system.	Performed under request in aerOS, embedding network functions (NFV) as CNIs
How aerOS goes beyond	Implementing virtual network functions (VNF) as programmable cloud native functions (CNF). Integrate networking requirements in aerOS services representation in aerOS continuum knowledge graph, to support overlay networks programmability via LLOs.	
WG7 – Monitoring and observability		
Smart Observability	Monitoring of selected information, taking advantage of edge computing in order to reduce network bandwidth usage, among other.	aerOS self-* suite, specially self-awareness, self-security and self-healing provide monitoring, and are supported by AI in self-optimisation
QoS/QoE policies	Establishment of quality of service or quality of experience policies for the services, the network and/or the user	aerOS portal and continuum ontology includes SLA incorporation, which is interpreted by HLO Allocation Engine
Physical monitoring	Monitoring of HW and SW related parameters of the continuum	aerOS Self-awareness and self-realtimeness
App monitoring	Service tracking, including status of the applications.	Services and service components registered and accessible under pub/sub schema following own aerOS' designed data ontology. aerOS EAT offers network (and templated, customized) analytics in the form or serverless functions
How aerOS goes beyond	Benchmark, decentralization in the observability of monitoring data!	
WG8 – Artificial Intelligence		
Scheduler **Personalised AI**	Capacity of scheduling workloads based on ML/DL algorithms, departing from the smart management of the continuum.	aerOS HLO Allocation Engine uses at least two approaches for allocating and scheduling tasks: (i) reinforcement learning, (ii) educated multi-parameter optimisation.
LM for cont. learning	aerOS does not address language model in its basic delivery, however the capacity of including LM models via EAT (OpenFaaS) is direct.	
How aerOS goes beyond	aerOS provides explainability methods (XAI) and frugality (FAI) for both internal intelligence (scheduler, smart allocation) but also globally to other models in the continuum.	

8. Conclusions and next steps

In conclusion, the journey of designing and refining the aerOS Meta-OS has been both challenging and rewarding. The advancements presented in this final version of the architecture deliverable demonstrate significant progress from our initial vision. Through rigorous research, continuous feedback, and iterative development, we have developed a robust and scalable framework that successfully integrates diverse and distributed resources into a cohesive IoT-Edge-Cloud continuum.

One of the most significant achievements has been the seamless integration of heterogeneous resources and systems. By leveraging standardized interfaces and advanced orchestration mechanisms, we have created a flexible and interoperable architecture that can adapt to various industry verticals. Enhancing security and privacy controls was a critical focus area, and the integration of advanced security protocols and the establishment of a centralized AAA control point in the Entrypoint domain ensure that our architecture meets stringent security requirements.

The architecture's ability to scale dynamically in response to workload demands, while maintaining optimal resource utilization, guarantees for the robustness of aerOS design. This flexibility is crucial for supporting diverse IoT applications and services across multiple domains. Continuous feedback from ongoing development and MVP integration has been invaluable, allowing us to refine our approach, address emerging challenges, and incorporate practical insights into the final design. Additionally, the introduction aerOS Federated Orchestration and the aerOS Management Framework concepts has provided a solid foundation for efficient resource management and service deployment across the continuum.

As we move forward, several key areas and steps have been identified to ensure the continued success and evolution of the aerOS architecture. Conducting extensive real-world testing and deployment of the architecture is essential to validate its performance, security, and scalability across different environments and use cases. This will help identify any remaining gaps and areas for improvement.

Fostering a vibrant ecosystem of developers, stakeholders, and industry partners is also crucial. Thus, the provisioned comprehensive documentation, development tools, and support will facilitate the adoption and extension of the aerOS architecture. In parallel, the established feedback loop to continuously gather insights from deployed instances and stakeholder experiences drives ongoing enhancements and ensures the architecture evolves to meet emerging needs and technologies.

Exploring the integration of advanced features such as AI-driven orchestration, enhanced analytics, and support for new types of IoT devices and edge computing paradigms will keep aerOS at the forefront of technological advancements. Ensuring that aerOS continues to adhere to industry standards and compliance requirements is also important. Engaging with standardization bodies will help us contribute to and stay aligned with evolving best practices and regulations.

In summary, the aerOS Meta-OS architecture has made significant strides toward realizing a Meta-OS which drives unified, scalable, and secure IoT-Edge-Cloud continuum. By building on the solid foundation established in this document and by pursuing the outlined next steps, aerOS is well-positioned to drive further innovation and achieve the vision of a seamless and integrated digital ecosystem covering all the distance form IoT to Edge to Cloud.

References

[1] ISO/IEC/IEEE, "Systems and software engineering -- Architecture description," *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000,* pp. 1-46, 2011.

[2] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob and M. Imran, "The Role of Edge Computing in Internet of Things," *IEEE Communications Magazine,* vol. 56, no. 11, pp. 110-115, 2018.

[3] Z. Sharif, L. T. Jung, I. Alazab and M. Razzak, "Adaptive and Priority-Based Resource Allocation for Efficient Resources Utilization in Mobile-Edge Computing," *IEEE Internet of Things Journal,* vol. 10, no. 4, pp. 3079-3093, 2023.

[4] M. Zhang and T. Chiang, "Fog and IoT: An Overview of Research Opportunities," *IEEE Internet of Things Journal,* vol. 3, no. 6, pp. 854-864, 2016.

[5] R. Vaño, I. Lacalle, P. Sowiński, R. S-Julián and C. Palau, "Cloud-Native Workload Orchestration at the Edge: A Deployment Review and Future Directions.," *Sensors,* vol. 23, no. 2215, 2023.

[6] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger and R. Wheeler, "ROS: An Open-Source Robot Operating System.," in *ICRA Workshop on Open Source Software,* 2009.

[7] R. Debab and W.-K. Hidouci, "Boosting the Cloud Meta-Operating System with Heterogeneous Kernels. A Novel Approach Based on Containers and Microservices," *J. Eng. Sci. Technol,* vol. 11, pp. 103-108, 2018.

[8] M. Hamdan, E. Hassan, A. Abdelaziz, A. Elhigazi, B. Mohammed, S. Khan, A. Vasilakos and M. M. , "A comprehensive survey of load balancing techniques in software-defined network,," *Journal of Network and Computer Applications,,* vol. 174, 2021.

[9] aerOS, "D2.1 State of the art and market analysis report," EC, https://aeros-project.eu/wp-content/uploads/2023/05/aerOS_D2.1_State-of-the-Art-and-market-analysis-report_v1.1.pdf, 2023.

[10] ETSI, "Context Information Management (CIM) NGSI-LD API," ETSI, 2021.

[11] "Lico," [Online]. Available: https://liqo.io/. [Accessed 20 7 2023].

[12] "NetMaker," [Online]. Available: https://www.netmaker.io/. [Accessed 18 7 2023].

[13] "Smart Data Model," [Online]. Available: https://smartdatamodels.org/. [Accessed 7 8 2023].

[14] aerOS, "DevPrivSecOps Methodology Specification," EC, 2023.

A. aerOS Terminology

Table 12. aerOS Terminology table

aerOS Term	Definition
aerOS Infrastructure Element (IE)	The fundamental building block within aerOS Meta-OS. A physical or virtual computing resource providing the necessary processing power, storage capacity, and network connectivity to support containerised workloads and services. Exposes aerOS runtime on top of provided capabilities being thus the minimum execution unit within the IoT-Edge-Cloud continuum.
aerOS Domain	A set of one or more IEs, functionally connected and sharing a common instance of aerOS basic services among them, constituting an administrative domain able to be managed and orchestrated by aerOS Meta-OS and thus be part of the IoT-Edge-Cloud continuum.
aerOS Runtime	The operational environment, running on top of each IE or domain, where the containerised workloads, services, and applications are executed and managed. Builds on top of container runtime, enhanced with standard capabilities needed per IE, to be integrated as part of aerOS domain, regarding its state exposure and self-monitoring.
aerOS Data Fabric	Architectural component, indispensable part of each aerOS domain, that automates the integration of data from heterogenous sources and exposes them through a standard interface. Enables the transparent exchange of information across the continuum using common interpretable information models and thus making data accessible any time, from anywhere. It serves the needs of a "consumer" to either support an internal Meta-OS procedure or an IoT vertical application execution.
aerOS Network and Compute Fabric	Architectural component, indispensable part of each aerOS domain, which establishes the underlying framework that abstracts this variety of heterogeneous computing resources and exposes them as a homogenized IoT-Edge-Cloud continuum environment. Therefore, it creates a unified layer for managing connectivity, communication, and computational resources, ensuring efficient integration and secure network interactions.
aerOS Service Fabric	Architectural component, indispensable part of each aerOS domain, which complements the Network and Compute Fabric by establishing a common "runtime environment". This runtime environment offers a robust framework for managing and orchestrating IoT applications as microservices in a standardized and unified manner. The aerOS Service fabric is built as a set of dedicated microservices capable of managing IoT service deployment, orchestration, and lifecycle management (LCM).
aerOS Federator	Component, implemented as set of services, included within each aerOS domain, which oversees the establishment of the appropriate mechanisms to achieve a fully federated and decentralised architecture among the aerOS domains. Federator component provides functionalities that, in conjunction with entrypoint domain management space, implement the mechanism that enables domain registration and discovery of other domains, and their capabilities, across the continuum.
aerOS Management framework	aerOS UI and framework responsible for the registration, management, and federated integration of aerOS entities across all domains along the Edge to Cloud path. It is a combination of management capabilities, regarding users,

	domains and IoT services implemented in aerOS entrypoint domain, and of management mechanisms that will oversee the creation and maintenance of the federation among the multiple aerOS domains that build the continuum.
aerOS Entrypoint domain	An aerOS domain enriched with capabilities related to Meta-OS management. Although it is a singleton presence within the continuum, it can be migrated at any time to another aerOS domain. It provides the Management Portal where stakeholders perform role-based access to capabilities such as domains registration and management, IoT services deployment etc. Maintains central registries as User, Policies, Data Catalog and in synergy with aerOS domains federator establish the framework for the creation and maintenance of the federation mechanisms between the multiple aerOS domains that build the continuum.
aerOS Context Broker (CB)	Component existing within each domain to store information about the state of the IEs and the kind of data available. CB implements NGSI-LD standard and enables the transparent and real-time update and exchange of information among aerOS domains, targeting real-time aerOS continuum state exposure to all "consumers", and establishing thus state federation across aerOS domains.
aerOS Distributed State Network of Brokers	In aerOS, a distributed state repository is a decentralised storage system responsible for maintaining the state information among different elements or components present in the continuum, fragmentarily corresponding to local domains. This repository is handled by one instance of Orion-LD CB in each aerOS domain, building a Distributed State Network of Brokers (DSNB).
aerOS Federated Orchestration	The process of orchestrating aerOS resources, both infrastructural and service resources, supported by AI/ML decision support services. This orchestration process spans beyond the legacy administrative domain borders as federator components, integrated within each aerOS domain, permit the knowledge of all aerOS integrated domains all over the established continuum. Thus AI/ML components take advantage of real time-information of the continuum status and solve constraint based double optimisation, placement related, problems regarding application requirements and resources availability.
aerOS Basic services	Services running within each aerOS domain, relying on aerOS runtime on top of IEs, realising functionalities (such as authentication and authorisation, among others) that enable integration of domain in the aerOS ecosystem, as part of the IoT-Edge-Cloud continuum, providing smart decisions and orchestration of aerOS workloads execution, targeting most efficient placement within the domain or to other domains across the federated aerOS network.
aerOS Auxiliary services	Services that provide complementary functionality across the continuum, without having an explicit core functionality.
aerOS High Level Orchestrator (HLO)	First step of the Federated Orchestration in aerOS is realised by the HLO. This element analyses the state of the continuum leveraging the Data Fabric (and the aerOS Federator) and takes an allocation decision (service deployment spot). This action takes an *Intention Blueprint* as an input and delivers an *Implementation Blueprint* as output to the HLO.
aerOS Low Level Orchestrator (LLO)	Second step of the Federated Orchestration in aerOS is realised by the LLO. This element interprets the *Implementation Blueprint* coming from the HLO and oversees the actual deployment of workloads in the selected IE(s). Being aware of the underlying container management frameworks, is able to convert the allocation order into proper deployment. Several LLOs may live in the same domain.