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

D2.6 – aerOS architecture definition (1)

Deliverable No.	D2.6	Due Date	31-AUG-2023
Type	R – Document, Report	Dissemination Level	Public
Version	1.0	WP	WP2
Description	aerOS initial 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 SP ZOO	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
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
Dr. Harilaos Koumaras	P02 NCSRD	koumaras@iit.demokritos.gr
Vasilis Pitsilis	P02 NCSRD	vpitsilis@iit.demokritos.gr
George Makropoulos	P02 NCSRD	gmakropoulos@iit.demokritos.gr
Anastasios Gogos	P02 NCSRD	angogos@iit.demokritos.gr
Constantinos Vassilakis	P02 NCSRD	cvassilakis@iit.demokritos.gr
Thanos Papakyriakou	P02 NCSRD	thpap@iit.demokritos.gr
Dimitris Davazoglou	P02 NCSRD	d.davazoglou@iit.demokritos.gr
Eneco Rada	P03 INNOVALIA	erada@innovalia.org
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
Ken Zangelin	P06 FIWARE Foundation	ken.zangelin@fiware.org
Ignacio Dominguez Martinez-Casanueva	P07 TID	ignacio.dominguezmartinez@telefonica.com
Fofy Setaki	P08 COSM	fsetaki@cosmote.gr
Angelos Constantinides	P09 8Bells	angelos.constantinides@8bellsresearch.com
Ioannis Chouchoulis	P10 IQB	giannis.chouchoulis@inqbit.io
Panagiotis Bountakas	P10 IQB	panagiotis.bpountakas@inqbit.io
Giorgos Petihakis	P10 IQB	giorgos.petihakis@inqbit.io
Dimitris Tsolkas	P11 FOGUS	dtsolkas@fogus.gr
Chistos Milarokostas	P11 FOGUS	milarokostas@fogus.gr
Joseph McNamara	P12 L.M. ERICSSON LIMITED	Joseph.mcnamara@ericsson.com
Katarzyna Wasielewska-Michniewska	P13 IBSPAN	katarzyna.wasielewska@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
Nikolaos Gkatzios	P15 INF	ngkatzios@infolysis.gr
Vaios Koumaras	P15 INF	vkoumaras@infolysis.gr
Aggeliki Papaioannou	P15 INF	apapaioannou@infolysis.gr
Eduardo Garro Crevillen	P16 PRO	egarro@prodevelop.es
Alvaro Martínez Romero	P16 PRO	amromero@prodevelop.es
Michele Mondelli	P19 DST	m.mondelli@dstech.i
Jon Egaña	P20 S21SEC	jegana@s21sec.com
Oscar Lopez	P20 S21SEC	olopez@s21sec.com

History

Date	Version	Change
08 May 2023	0.0	Initial ToC and WP2 general meeting announced
15May–12June 2023	0.1-0.2	Sections' structure and ToC revisions based on architecture advancements
19 June 2023	0.3	Assignments according to ToC and WP2 discussions
10-24 July 2023	0.4-0.7	Content integration
24 July 2023	0.8	Internal review submitted
7 August 2023	0.9	Address internal reviewer comments
8 August 2023	1.0	Deliverable submitted

Key Data

Keywords	IoT, aerOS, meta operating system, continuum, network & compute fabric, service fabric, data fabric, aerOS distributed state repository, architecture, federation, orchestration, federated orchestration, aerOS Infrastructure Element, aerOS Domain,
Lead Editor	P02 NCSRD
Internal Reviewer(s)	Christos Milarokostas, P11 FOGUS Marcin Michel, P23 ELECTRUM

Executive Summary

This document is the first of two deliverables addressing aerOS architectural design with the goal to deliver a high-level prototype architecture of aerOS and specify all functional and technical concepts and components. By introducing aerOS technical framework and components' implementation principles, this deliverable introduces technical developments needed, to be executed within WP3 and WP4, and guides instantiation and deployment for all use cases that will take place within WP5 and open calls.

In order to produce the overall aerOS architecture, it is important to identify a solid methodology, which can introduce aspects of feasibility and efficiency. A methodology that can define a process, which receives requirements as input, identifies stakeholders, concerns and flows and finally specifies architectural concepts, implementing components and interaction needs.

Following the defined methodology, this deliverable specifies the rational for an IoT-Edge-Cloud continuum. It identifies the deficient, as of today, status which deprives the possibility from IoT developers to take advantage of distributed capabilities all along the continuum and to also take advantage of a common execution environment which can support IoT services deployment and reuse. The need to move from an isolated resources' usage to a compute and service fabric that spans all the way from edge to cloud and provides the underlay for a unified service orchestration and execution environment, aka service fabric, is analyzed, and the means to provide a most efficient and smart orchestration of resources on top of it, based on an innovative data fabric implementation, are thoroughly discussed. aerOS is identified as a management and orchestration system of the above-mentioned fabrics, and with the goal to expose a continuum of compute, network, and service resources to IoT developers. Based on this fact it becomes obvious that aerOS acts as a Meta-OS capable of operating all underlying fabrics and expose a coherent continuum, which should provide IoT developers with a seamless service deployment experience.

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 and over all domains enables the most efficient usage of all resources and the most efficient placement of IoT applications. This is well explained throughout this deliverable and the data fabric innovative mechanisms that make possible the seamless usage of information anywhere and anytime is discussed.

The actual infrastructural components that make feasible the implementation of these concepts are presented and the required process to render any available compute or network resource to an aerOS element is defined. Infrastructure elements and aerOS domains capabilities are presented and the set of basic services that should be running within every domain are discussed.

Thus, the process to render any legacy compute and network resource to an aerOS execution element is explored and the possibilities provided on top of this are presented. Therefore, having read this document all stakeholders should be aware of what should they expect and how can they benefit from aerOS, what is needed to render resources to aerOS elements and how to proceed with IoT services deployment once having registered with the aerOS ecosystem.

Table of contents

List of tables

List of figures

List of acronyms

Acronym	Explanation
AGV	Automated Guided Vehicles
aka	also known as
CB	Context Broker
CEI	CloudEdgeIoT
CNF	Cloud Native Function
CNCF	Cloud Native Computing Foundation
CR	Custom Resource
CRD	Custom Resource Definition
DNSB	Distributed State Network of Brokers
FaaS	Function-as-a-Service
HLO	High-Level Orchestrator
IE	Infrastructure Element
IaaS	Infrastructure-as-a-Service
IoT	Internet of Things
JSON-LD	JavaScript Object Notation for Linked Data
LCM	Life Cycle management
LLO	Low-Level Orchestrator
MANO	Management and Orchestration
NFV	Network Function Virtualization
OAuth	Open Authorization
OIDC	OpenID Connect
K8s	Kubernetes
MEC	Mobile Edge Computing
Meta-OS	Meta Operating System
NGSI-LD	NextGeneration Service Interface – Linked Data
PaaS	Platform-as-a-Service
RDF	Resource Description Framework
REST	Representational State Transfer
RPi	Raspberry Pi
SDN	Software Defined Network
SPARQL	SPARQL Protocol and RDF Query Language
URL	Uniform Resource Locator
VIM	Virtual Infrastructure Manager

VM	Virtual Machine
VNF	Virtual Network Function
VPN	Virtual Private Network

1. About this document

The main goal of this deliverable is to provide a comprehensive overview of the architecture that underpins aerOS as a Meta-OS instantiation. The document delves into the intricate details of the system's structure, design principles, components, and their interactions. It outlines the high-level vision and objectives, introduces basic architectural concepts and details how different elements work together to achieve the desired continuum functionality and performance. Additionally, the document encompasses the technology stack, data flow, communication protocols, and integration points, providing a clear roadmap for the development and deployment of the system. Capturing all essential architectural aspects, this document aims to serve as a reference for stakeholders, engineers, and decision-makers, ensuring a unified understanding of the system's design and subsequently guide its successful implementation and evolution.

It should be highlighted that this deliverable corresponds to the first out of two documents, and therefore its content will be expanded and adapted as the project evolves. Minimum Viable Products deployment, Pilots' architecture integration and Open Calls implementation feedback, as well as accurate components implementation details coming from WP3 and WP4 evolution will be integrated in the way to finalize an efficient and accurate architecture for aerOS.

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	D2.6 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. D2.6 defines WP5 process as it guides: • T5.2 which undertakes the implementation in vertical industry pilots based on the produced architecture blueprint. D2.6 should be in 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 in order 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 contributes to the realisation of *MS3 – Components defined*, that will be achieved in M12, It indirectly supports *MS4 – Use cases deployed*, that will be achieved in M18, it directly contributes to *MS5 – Final architecture defined*, that will be achieved in M21.
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 D2.1 (State-of-the-Art and market analysis report), D2.2 (Use cases manual, requirements, legal and regulatory analysis (1)). It is prepared in parallel to D3.1 (Initial distributed compute infrastructure) and D4.1 (Software for delivering intelligence at the edge preliminary release) and D5.1 (Integration, evaluation plan and KPIs definition) and "overviews" synchronization of their content to its output. It is expected to support next technical deliverables versions (D3.2, D3.3, D4.2, D4.3) and D5.3 (Use cases deployment and implementation).

1.2. The rationale behind the structure

The content of the deliverable is organised in five 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 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 4. This section introduces the basic concepts, and the implementing building blocks that aerOS employs towards continuum establishment and operation. Architectural decisions and their significance and the technologies implemented aiming 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 5. This section having introduced aerOS components, delves into different viewpoints that expose their roles but also highlight the functionality they should have and the interconnections they should provide in order to define their proper placement and interaction and ensure operational goals satisfaction.
- Appendix A. Finally, this section contains a brief presentation of main, commonly and frequently aerOS related, and for its purposes produced, terms usage.

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 at 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 a first version of 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.

aerOS deliverable D2.2 – Use cases manual, requirements, legal and regulatory analysis, provides the initial version of requirements elicitation, use cases and scenarios definition, and legal and regulatory analysis. It serves as a valuable input for this document, since in D2.2 aerOS stakeholders are identified, pilot use cases actors and scenarios are recognised, and technical, user, and system requirements are recorded in a structured way that allows us to extract the various Stakeholders Concerns and drive Architecture Decisions.

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. 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.).

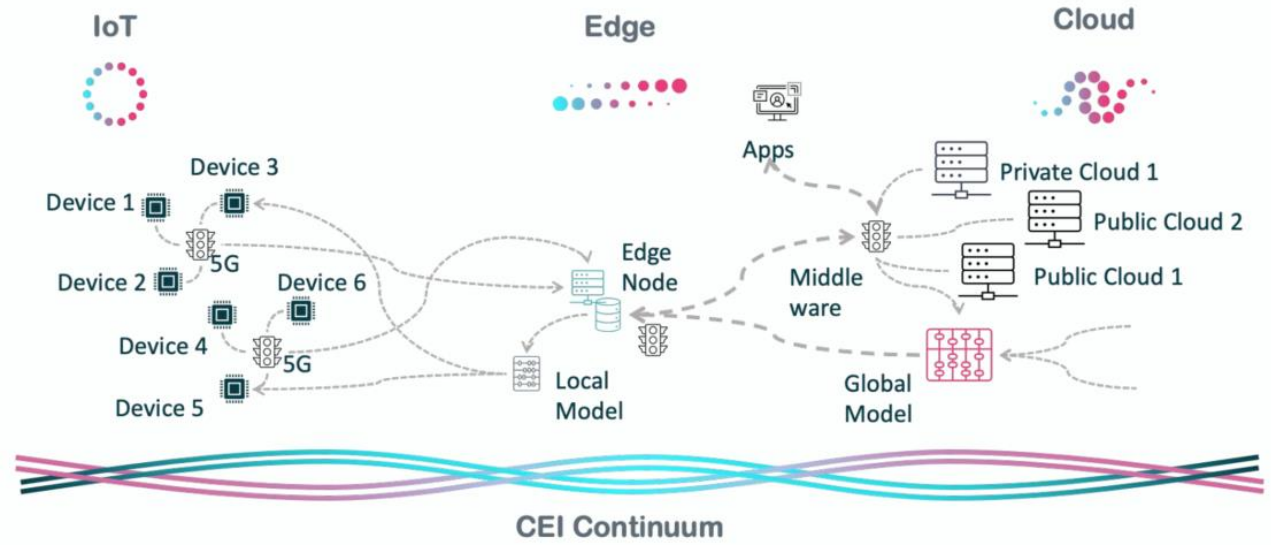


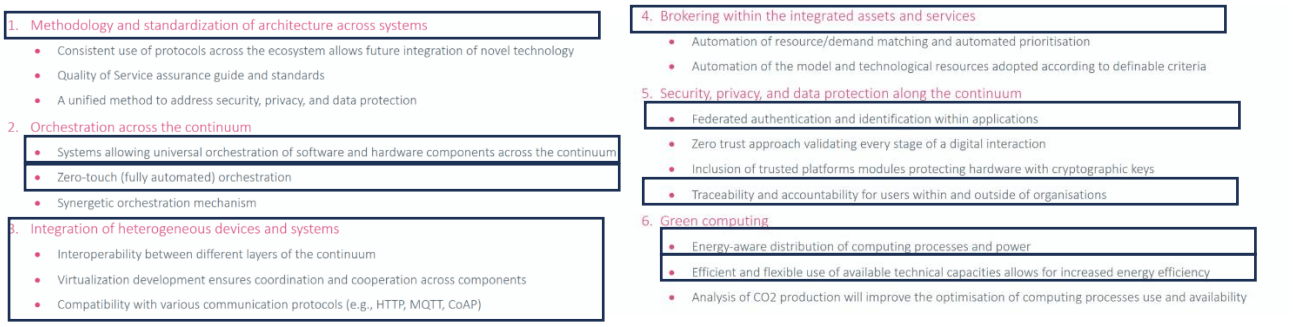
*Figure 1. Cloud-edge-IoT continuum perspective source: **EUCloudEdgeIoT***

Technologically, the continuum approach should achieve better management of the widespread and heterogenous computing resources stretching along the line from small devices to large cloud datacenters. 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.

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. 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. Ultimately, **the CloudEdgeIoT ecosystem** will work towards developing 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.

3.1. 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 Section 4.3.2), 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.

3.2. 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.

3.2.1. From heterogeneous IoT data to a unified data fabric

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

The notion of a Meta-OS that spans multiple technological domains that form the continuum -from the IoT sensors to edge locations, and up to the cloud- increases the complexity of data management. Data sources are not only located in some physical locations, such as IoT sensors, but they are also spread over multiple physical and virtual locations across the different domains. Besides, this is a highly changing environment, as new data sources may become available, move to other domains, and even disappear. Such an intricate and heterogenous data landscape introduces several challenges in two main aspects: i) data analysis and ii) data governance.

Vertical services, such as ML applications, that aim to consume and analyse these data to realize their business capabilities, 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 collecting, processing, and consuming the data of interest. Moreover, services shall correlate and combine data from multiple data sources. All these data analysis processes entail 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.

To truly deliver the feeling of a continuum, aerOS must provide a unified view of the data, regardless of their original location, format, or data source technology. To this end, aerOS relies on a homogenization layer -based on semantics- that abstracts data consumers from the underlying complexities of the continuum and exposes the data through a standard 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. 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.

3.2.2. From a distributed cloud eco-system to a unified network-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.

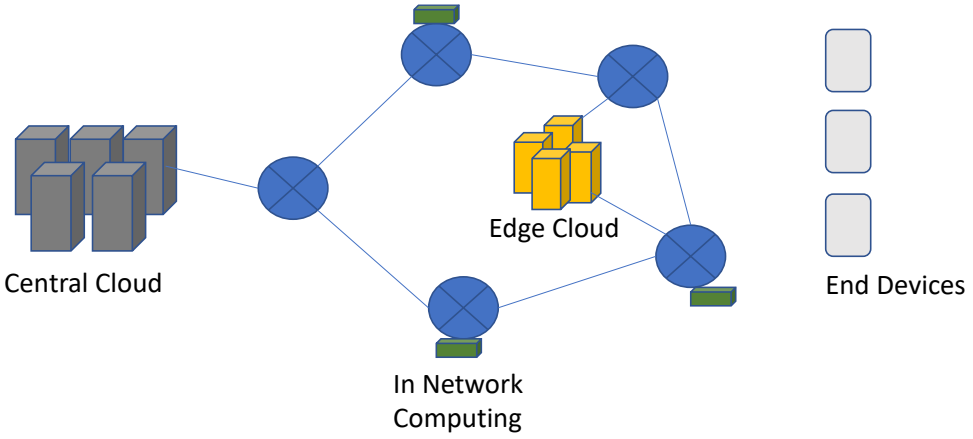


Figure 2. Schematic of aerOS Cloud Resource Types: Central Cloud, In-network Computing fabric, Edge

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 framework, namely 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.

3.2.3. 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.

The term "monolith" is frequently associated with something big and glacial, which is not far from the truth of a software design monolith architecture. 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.
- 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.
- Scalability - Individual components cannot be scaled.
- 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.

Moreover, the monolith application cannot take advantage of the heterogenous infrastructure on top of which can be deployed in a distributed way. The IoT-Edge-Cloud Continuum governed by aerOS will introduce an agile architecture, mechanisms that will allow 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 techniques -using explainable Artificial Intelligence (AI)- 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. 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 business logic and database, and they serve a specialised purpose. Each service undergoes 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.

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 micro-services 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 between microservices. 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, microservice orchestration works similarly. 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. The whole system is 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. More details on this intelligent distributed service management and placement is provided in Section 4.2.2.

3.3. Meta-OS approach and aerOS vision

A Meta-OS provides the services and functionalities that would be expected from an operating system while running over an environment where many distributed input/output resources are integrated in an information-driven manner. A schematic analogy between a legacy OS and a meta-OS, as envisioned and discussed in this document. The type and number of functionalities offegred may categorise it neither as an operating system nor as a framework. A good example of a Meta-OS approach is the popular in the robotics domain, Robot Operating System (ROS) [5]. 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 it can be used for a different combination of hardware implementation. Meta-OSs developed in various domains enable different levels of coordination among heterogeneous devices (physical or virtual), operating systems and services [6] [7]. aerOS builds on the concept of the Meta-OS approach to intelligently support hyper-distributed applications across the IoT–Edge–Cloud continuum.

aerOS aims to implement a Meta-OS which will unify and orchestrate 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 a significant diversity regarding their capabilities and the operating systems running on top of them. End users, i.e., IoT developers from a variety of industry verticals using aerOS, should have the impression of a seamless integration of these underlying resources and should be -transparently- in place to take advantage of all the smart, aerOS-enforced, orchestration and federation decisions, in their effort to deploy their services as close to the edge as possible, leveraging at the same time the most appropriate and efficient resources' capabilities combination, as exposed from all underlying infrastructure across all the path from edge to cloud.

This indicates the need for aerOS, as a Meta-OS, to implement, operate and manage the continuum in all the involved layers. In order to understand aerOS as a Meta-OS, it is necessary to explore the meaning of continuum as a derivative of layered fabrics supporting end-to-end and edge-to-cloud, orchestration, life cycle management, and federation of computing and networking nodes (physical or virtual), services, and data. Then aerOS can be perceived as the continuum(s) enabler and administrator, 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.

In a 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 administrative domains. The current status is that a vast number of non-connected, distinct, computing islands host applications with certain demands. Elements in those islands can expose processing, storage, or networking capabilities, only to a certain extent and are often forced to rely on big cloud providers, in their effort to execute intensive computing tasks. aerOS Network and Compute Fabric aims to expand these capabilities, by providing federated access to unused and available resources all the way from the edge to the cloud, acting as a network and compute fabric and establishing thus a "continuum" of resources' exposure.

Built upon the foundation of the "network and compute fabric", aerOS offers a comprehensive framework that serves two primary purposes. Firstly, it provides essential mechanisms for seamless and transparent resource sharing, orchestration, and federation. Secondly, it offers a higher-level abstraction for the development, deployment, and management and orchestration (a.k.a. MANO) of distributed applications and services. This abstraction empowers developers to create scalable, reliable, and highly available applications by abstracting the complexities of accessing underlying infrastructure. It also improves the whole continuum performance by selecting the most efficient placement of services on top of the most appropriate resources. This layer consists of a set of basic and auxiliary services, called aerOS services, running within each one of the connected domains and on top of their constituting network and compute elements, and is referred to as the **"aerOS Service Fabric"**. aerOS Service Fabric exhibits key characteristics that enable effective management of services deployed over the infrastructure, for the most efficient use of this infrastructure orchestration decisions, enabled within the aerOS service fabric, break down, if needed, IoT services deployment requests to a set of microservices, and provision the most efficient placement of each one of them according to the underlying resources availability and the required capabilities. Thus, it provides a comprehensive support for IoT developers to deploy applications across vertical domains, spanning from the far edge to cloud premises (whether public or private). This support encompasses a wide range of features, including life-cycle management of services, intelligent orchestration, resilience, scaling, discoverability, messaging, monitoring, and many other capabilities.

The above-mentioned Network & Compute and Service fabrics come with a lot of information regarding their capabilities, availability, and several context-related information (e.g., location). Also, each of the computing resources and all the services produce a lot of runtime data, that can provide substantial information about their current state and how their capabilities and availability change over time. A comprehensive layer is implemented to facilitate the efficient management, integration, and access of all these data across the distributed systems and diverse data sources and this is the **"aerOS Data Fabric"**. The data fabric provides a unified and consistent data layer that abstracts away the complexities of data storage, processing, integration, governance, and sharing; thus, enabling data aerOS producers and consumers, spanning across all discrete administrative domains, to transparently exchange data. This enables a seamless and interpretable integration of data in AI-based and cognitive services, by leveraging this information. aerOS Data Fabric employs knowledge graph models to store and represent aerOS integrated resources (hardware and services) and their properties. Under the light of this choice, all resources along with their state are depicted as entities with a connected set of properties and attributes. Relationships deployed within the knowledge graph can represent their physical, multilevel connections. Flexible relationship objects can visualise a potentially "mobile" world where the presence of computing resources and IoT services may be altered any time, and even triggered by unforeseen events.

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 any time among all connected domains and elements. These capabilities enable both the access to this information about all the aerOS entities deployed; thus, having a full overview of the whole ecosystem at any time for any consumer, and the exploitation of this information across the whole ecosystem, from the edge to the cloud, in order to develop mechanisms which can recursively adapt and reconfigure resources usage and orchestration. Here, adaptation to meet criteria expressed by the users or owners of the continuum will be paramount.

All the above highlight a system which can seamlessly integrate a wide spectrum of computing and network resources and services that span from the edge to the cloud. It represents the concept of a unified architecture, where computing capabilities are distributed across different administrative domains and physical locations and allow for the most efficient and optimized placement of workloads for processing, storage, and networking. This seamless integration and orchestration of resources, services, and data across multiple domains, locations, and regardless of the heterogeneity of underlying devices and operating systems, along with the data fabric capabilities to share and distribute data under interoperable mechanisms establish a continuum. A continuum that, even though it spans across distinct administrative domains, across a multitude of device capabilities and architectures, across different connectivity providing networks, works seamlessly and can leverage all underlying resources for the benefit of IoT developers. A continuum that at the end provides a common execution environment for these developers.

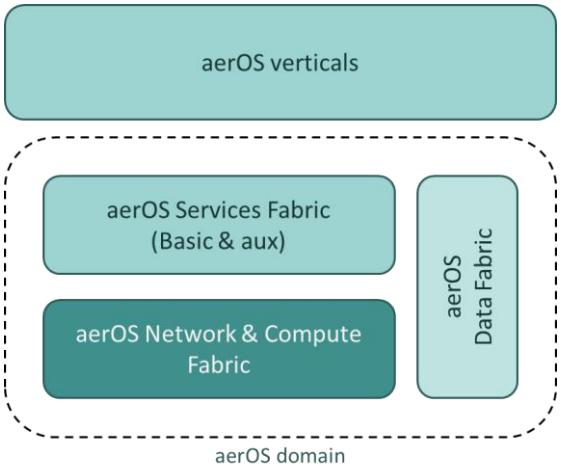


Figure 3. aerOS domain as part of the continuum

As a result, aerOS, as a Meta-OS, intelligently establishes and manages a seamless continuum that integrates resources and services, generating valuable data which are continuously monitored, processed, and then utilised by distributed AI services, with the aim to enhance continuum functionality. Insights derived from AI-based

data pipelines are provided as an input to the orchestration processes; thus, adapting, and reconfiguring resources and services, enabling a Meta-OS self-sustaining loop. This process operates in a zero-touch management paradigm, where automation and intelligence drive the continuous evolution and optimization of the continuum ecosystem. Conclusively, in pursuit of its goal, aerOS provides a comprehensive framework to establish a seamless continuum across computing, service, and data layers. As a Meta-OS, aerOS assumes the crucial role of supporting the establishment, sustainability, and efficiency of this continuum. By doing so, it strives to offer IoT service developers in industry vertical domains a unified execution environment built upon a transparent ecosystem of resources. This unified environment enables streamlined development, deployment, and management of IoT services, fostering efficiency and innovation within the industry.

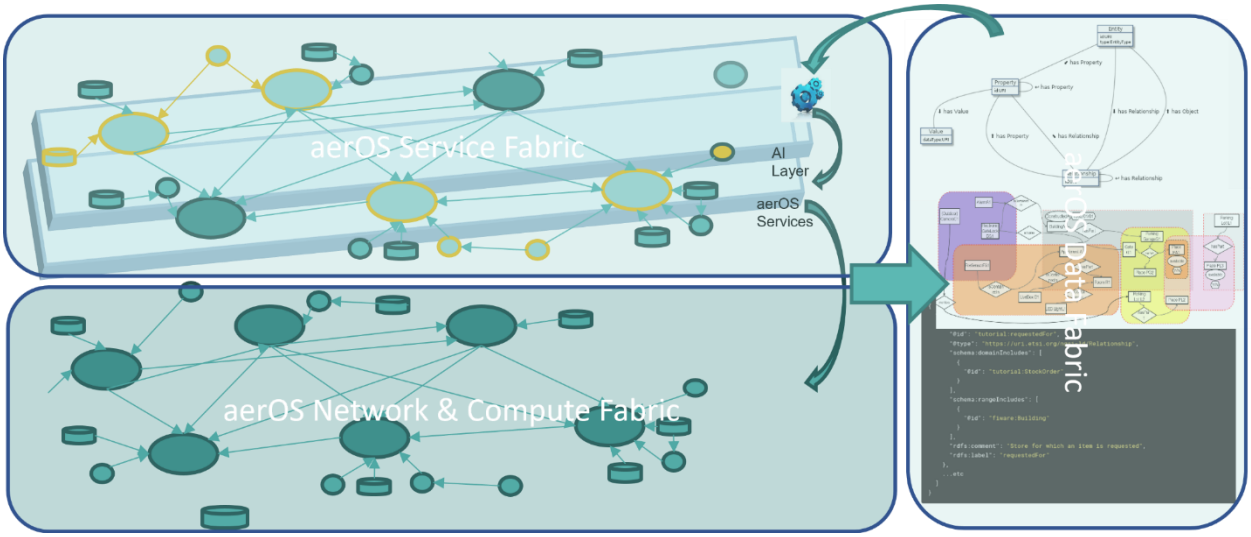


Figure 4. Compute, Service and Data Fabrics as aerOS continuum constituents

4. aerOS stack definition

To develop aerOS, following a Meta-OS approach that encompasses all continuum aspects discussed earlier and to address the project objectives, the system is structured around key building blocks and fundamental concepts. Section 4.1 offers 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.

4.1. aerOS stack

Our approach to describing the overall capabilities and components of aerOS is incremental, reflecting the evolving and expanding demands of moving from legacy IoT system approach to a Meta-OS for the IoT cloud to edge continuum. We gradually present the system's fundamental concepts, the implementing components, and the exposed capabilities, considering the dynamic nature of this transformation.

The **aerOS IoT cloud to edge system** comprises multiple **aerOS domains**, each consisting of a collection (at least one) of **Infrastructure Elements (IEs)**. Among these domains, there is a specialized aerOS domain known as the management portal, serving as the system's entry point. Among all aerOS domains, there exists a distinct and specialized one referred to as the "management portal," which functions as the primary entry point to the system and which despite the fact that it is uniquely substantiated in one domain, can easily be mitigated to some other while in runtime. These domains and IEs form the fundamental building blocks of aerOS, enabling the deployment of a comprehensive set of services that create a federated environment. This environment is designed to efficiently handle and fulfil requests from IoT developers, acting on behalf of various industry

verticals. To facilitate the seamless deployment and management of resources across all domains -from the far edge to the cloud- two key concepts play a prominent role: federation and orchestration. aerOS incorporates its own viewpoint, design, integration, and implementation of these concepts as functional enablers of the continuum, and their interdependent and inter-supported integration establishes an innovative federated orchestration process. Before delving into the tangible building blocks which make up aerOS stack, it is essential to explore these concepts from an aerOS perspective and as integrated in aerOS architecture.

Federation plays a vital role, by leveraging the collective capabilities and resources available across domains, in enabling collaboration, resource sharing, workload distribution, and interoperability across multiple aerOS domains within the IoT-Edge-Cloud environment. A basic feature, of the scoped continuum, is to build on the capabilities and resources available among domains in order to promote cooperation and synergy within the continuum, by sharing resources, exchanging data, and having them work together towards common goals. Federation establishment is meant to facilitate workload distribution and load balancing across domains. It acts as an enabler for the orchestration layer, the AI decision support, and the data fabric to assess resource availability in different domains and determine the optimal placement of workloads and additionally make possible the seamless migration and distribution of workloads to aerOS domains with available resources, ensuring balanced resource utilization and performance optimisation. Federation enabled, cross-domain resources provisioning and data exchange (under defined rules, protocols, and standards for access control, security, and interoperability) allows aerOS domains to expose their services and capabilities to other domains in a well-defined, interoperable, and controllable way. Thus, the overall system can leverage on resources and capabilities distributed across multiple administrative networks; thus, enabling the composition of end-to-end services which may be deployed across multiple domains. aerOS federation refers to the process of unifying and integrating all of the above-mentioned fabrics (compute and network, service, and data), allowing them to work together seamlessly as a cohesive and scalable system. It enables the system to efficiently utilize resources, share information, and provide a unified interface for users and administrators to interact with the entire system as a whole. aerOS federation is implemented based on a **"Federator"** (Figure 5) enabler (component) included within each aerOS domain and a central domain registry, where initially each new aerOS domain registers and publishes its capabilities and exposure location. This, centrally located information, can subsequently be pulled, and replicated across aerOS domains, based on the integrated federator enabler functionalities, avoiding thus centrally located, single point of failure and, run time dependencies. This idea is further discussed in section 4.3.8 and complementary presented in Figure 15.

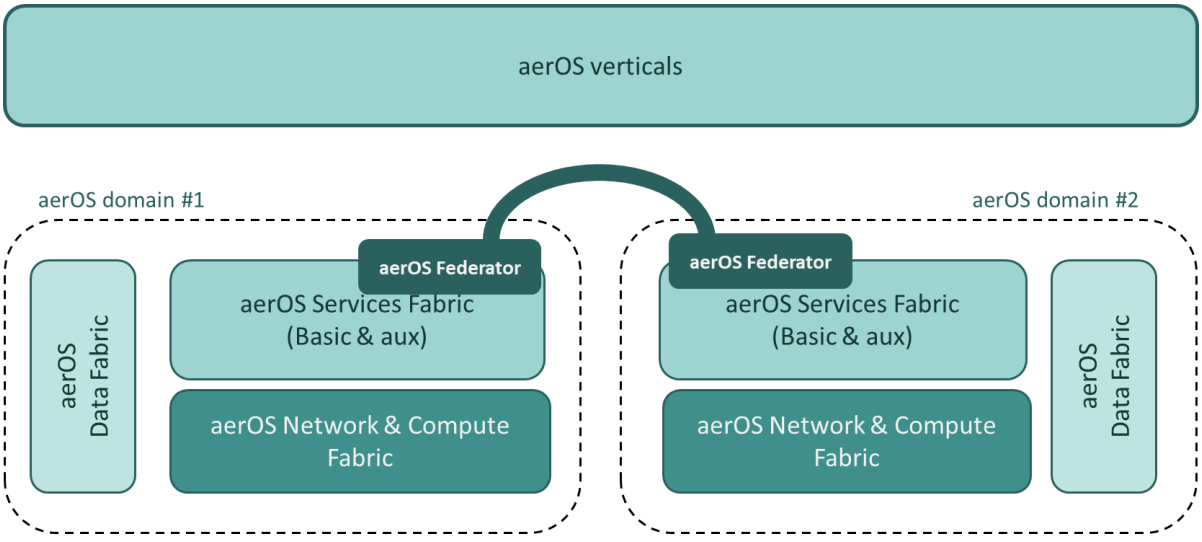


Figure 5. aerOS federator within and cross aerOS domains

Another concept, equally important for aerOS continuum establishment, is **Orchestration.** It is pivotal for managing and coordinating the deployment and execution of the containerised workloads across multiple domains from the edge to the cloud. aerOS orchestration "stands on the shoulders" of federation; thus, being able to take standard orchestration ideas - especially those of CNF- one step further, by integrating distributed AI smart decision support systems, trust services, and knowledge graph-based aerOS state models, to most

efficiently allocate workloads placement. Based on the federation provided capabilities, innovations in aerOS orchestration will now take advantage of cross domain communication facilities to coordinate activities and share resources located in numerous administrative domains all the way from the edge to the cloud. Additionally, taking advantage of the aerOS Data Fabric, enabled as part of the federation process, orchestrator components can have a complete overview of the resources' availability, and their capabilities, across all the continuum and during the time, having thus an extensive set of choices regarding workloads' placement. To take advantage of all this information, each aerOS domain encompasses an orchestrator component which can make the best decisions for the placement of the received requests not only in local managed resources but also across other aerOS domains. In fact, the architecture for the aerOS orchestrator involves a dual layered design where a **High-Level Orchestrator** (HLO) receives deployment requests, using a templated descriptive model, and with the support of AI and trust manager components, plus the Data Fabric provided privilege of having access to the information of current state across all domains, it can make the most appropriate decisions on where to place the workloads. In case local Infrastructure Elements can offer available resources then a suitable deployment *Decision Blueprint* is provided to the **Low-Level Orchestrator** (LLO) which knows how to access Infrastructure Elements' deployment facilities within the domain and deploy workloads. On the other hand, if a remote domain is chosen as more appropriate, the remote high-level orchestrator is addressed, and the request is transferred to this domain where a similar process is executed. As a takeaway regarding the aerOS orchestration, we can see that it is a totally distributed system where each orchestrator has jurisdiction within the domain and the infrastructure elements it lives in, but also thanks to aerOS federation and Data Fabric, has a direct, real time knowledge of all domains' status and can thus decide and access remote orchestrators to deploy jobs based on their specific demands and on resources' capabilities and availability. Finally, there is the innovative architectural design decision of implementing the orchestration, based on two layers, where the "lower" one has access to execution units, and the "higher" one has all the complexity and intelligence of receiving requests, revising them, and taking the decisions with the support of AI and trust agents. Thus, HLO integrates "pluggable" decision support systems, i.e., AI and trust components, which drive for the most efficient and secure choices.

The functionalities described above are realised through a cohesive set of interconnected functional components 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 an underlying framework 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.

Before proceeding to describe aerOS building blocks, which provide the actual implementation infrastructure, it is crucial to highlight here the role of **aerOS Data Fabric,** explained in Section 4.3.2, and the knowledge-graph based model representation choice for depicting aerOS resources state and their relationships across the continuum. Each aerOS domain integrates an aerOS Data Fabric enabler component which enables each interested consumer within the domain, either be the high-level orchestrator, the trust manager, or an AI decision support component, to ask for data and just receive data. Relaxing the need to access other domains or data producers across the continuum; a local Data Fabric agent takes care of all the underlying complexity. The technologies and protocols integrated and developed for the Data Fabric realisation which enables the aerOS distributed state repository (Section 4.2.3) make it possible to either poll for data or receive notifications about subscribed changes or even be updated based on established registrations, and all these across all the different administrative domains. The interesting part is not just the ability to get updated based on one of the previously mentioned mechanisms from the edge to the cloud but also the possibility, (the choice for a knowledge-graph model provides this), to represent and query infrastructure within a context of dynamically related resources and also characterised, and connected with a set of properties absolutely indicative of their most important features regarding their eligibility to be the "chosen ones" for a task execution or not.

aerOS stack and building blocks

Infrastructure Element is the fundamental building block of the aerOS system. It provides the computational infrastructure necessary to deploy and manage workloads and it can be any physical or virtual entity that

supports containerised workloads. IEs are deployed within or connected to an aerOS domain and play a crucial role in hosting and executing containerised applications or services. As a minimal execution unit, it is the base of aerOS stack, as it exposes the minimum capabilities needed to support workload execution, enhanced with a set of lightweight aerOS integration enablers, as thoroughly explained in Section 4.2.1. IEs are capable of supporting containerised workloads and are susceptible to be orchestrated by a primary element within each domain. Each IE can instantiate all the roles needed within a domain. It just executes workloads, and it is agnostic of the purpose of these workloads. These can be vertical oriented IoT applications, the domain enabler of the aerOS Data Fabric, a part of the orchestrator, or any other task. IEs just provide their capabilities to the network and the compute fabric and service fabric takes over from there to deploy and manage the most suitable and appropriate services.

An **aerOS Domain** constitutes a complete aerOS execution environment. It can be formed by (at least) one or more IEs. There are two prerequisites that make a set of connected compute and network resources an aerOS domain. The first is that these compute resources are integrated as IEs, as described above; thus, being capable to support workloads execution and provide a minimum set of aerOS integration capabilities, e.g., manageable network functionality. The second factor is that a set of core, basic, aerOS services are deployed on top of these IEs (or this IE as domain can be a single IE). So, in conclusion, an aerOS domain can be defined as a group of IEs which share the same set and instance of aerOS basic services. As to which are these basic aerOS services, it is distilled from the discussion above that those are:

- **Federation service**, which is provided by the federator component. Federation service is responsible to make the needed discovery subscriptions, and registrations to other aerOS domains and enable thus domains and IEs state propagation across the continuum, and orchestration requests forwarding. We must highlight here that federator requests and information can be propagated between domains thanks to the employed mechanisms implementations (NGSI-LD) which are discussed later, in Section 4.2.3.
- **Orchestration service,** which is responsible to receive user described intentions, regarding IoT services deployment, and encompasses decision support systems, trust management services, to translate to actual deployment requests and finally deployments on IEs. As mentioned above this is a two-level process, a high level receiving the, templated, user request and handling all the decision complexity and a low level to access resources. It is essential to mention here that although decision support systems are deployed locally, i.e., within the specific domain, they leverage information regarding the state of domains and IEs across all the continuum and this is provided "for free" by the Data Fabric services.
- **Data fabric services**, which are implemented within each aerOS domain and take care of the identification, collection and, in an interoperable way, integration of the data, and then enforce the required governance policies. Finally, they provide all the mechanisms to discover, connect, and retrieve data from other aerOS domains, as data fabric is a layer running across the continuum. As described before, in each domain, consumers should be enabled to just ask for data and they get them in a transparent way without knowing how or where they came from. These services are provided by the data fabric layer. Consumers can be other aerOS services (e.g., the orchestrator), IoT applications, or even external agents (e.g., portal).
- **Security services**, which integrate authentication, authorisation and access policies based on roles and identities.

Additionally, other aerOS services enable aerOS to act as an intelligent and secure Meta-OS able to establish and manage the continuum, reside possibly within each aerOS domain some of which we have already mentioned before.

- **Trust management service**s, which can exploit the aerOS data fabric provisions and calculate a trust score per aerOS domain or even node, to guide the domain orchestrator to the most appropriate choices.
- **AI decision support** services, which can again leverage on data retrieved by the data fabric, to provide input to the orchestrator regarding best placement, either locally or to another domain.
- **Analytics services**, which can provide insights into data, decision-making, and data processing for other components upon request.

Each aerOS domain, on top of these services, exposes a comprehensive and efficient Application Programming Interface (API) to communicate with stakeholders, agents, and other domains. Figure 6 displays an absolute minimal and schematic representation of an aerOS domain, where just the immediate identifiable continuum enablers are highlighted.

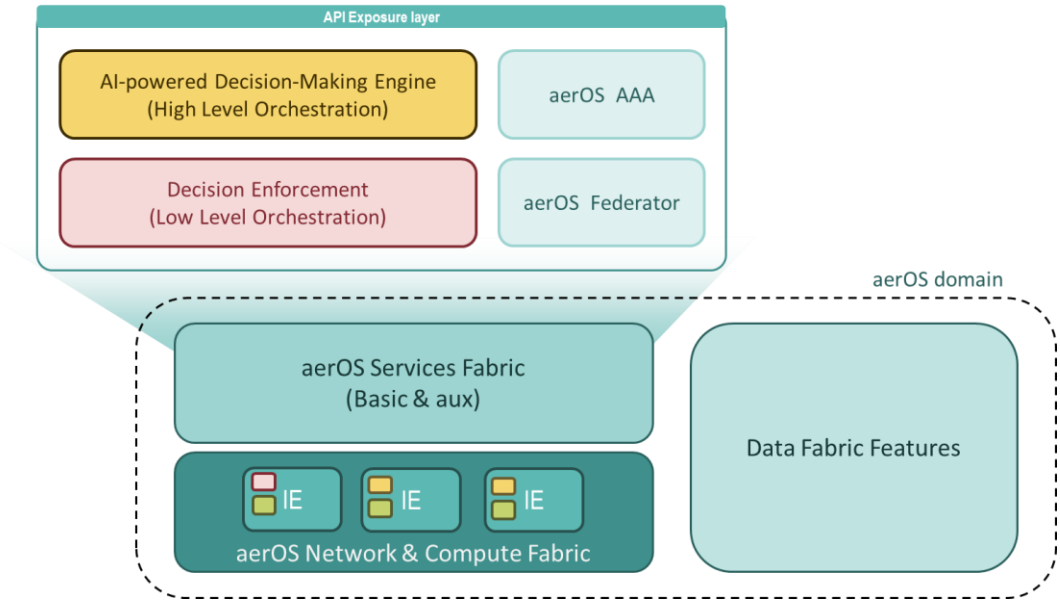


Figure 6. aerOS domain

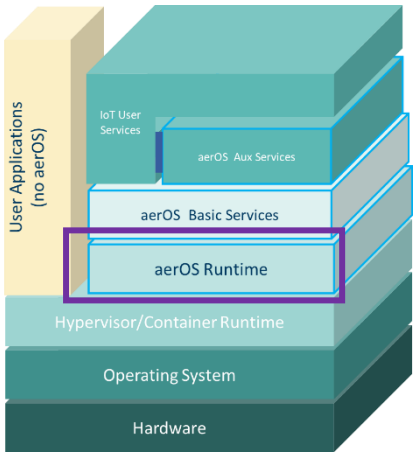
aerOS Management Portal is the entry point to the system. The Management portal will live in a certain aerOS domain, which will be considered the "entrypoint domain" and will be enriched with a set of additional components, which set this "unique" node as an aerOS point of presence offering Meta-OS management capabilities. It integrates the aerOS access dashboard, a registry of users and policies, a federation entry point for newly registered domains, and a mechanism to support a balanced distribution of user services deployment requests. Dashboard is the sole entry point to aerOS ecosystem for all stakeholders, acting as a single window for the management of the continuum (analogous to a terminal/shell in a classic OS). It maintains connectivity to the Identity and Authorisation Manager and exposes a friendly administrative interface for the User Registry to create/edit/delete users, change their roles, etc. Furthermore, the dashboard offers i) the space to register newly created aerOS domains as part of the aerOS continuum and ii) more functionalities related to management and visual representation of the "current" state of aerOS domains and the continuum. Although it is the sole point for the registration/management of domains (through the Management Portal dashboard), the exchange of information among the domains from there on is fully decentralized and federated. The fact that this "special" aerOS domain is unique does not mean that is not exchangeable at any time if a failure raises or administrator chooses to migrate it to another aerOS domain.

Finally, aerOS architecture introduces the concept of the aerOS Management Framework as a combination of Management Portal provisions and aerOS domains' located Federator components, which in synergy establish the framework for the creation and maintenance of the federation mechanisms between the multiple aerOS domains that build the continuum. This framework enables domains registration and discovery, aerOS distributed state federation and propagation, and access policy for consumers. By design **aerOS is not a centralised solution**, and it is this framework that architecturally makes all provisions for establishing appropriate mechanisms needed to achieve a fully federated and decentralised architecture among the aerOS domains.

The following sections delve into the details of aerOS building blocks and components and their role within aerOS, in the way to establish a smart orchestration of a federated resources' continuum from edge to cloud, having always in mind to provide a flexible, trusted, and common development and execution environment for IoT services developers across multiple industry verticals.

4.2. aerOS runtime

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".



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.

Figure 7. aerOS runtime component as part of aerOS stack

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.

The following sub-sections dig deeper in the particular mechanisms that conform the aerOS runtime. In the next version of this deliverable, more technical details on the chosen technologies and the recommended implementations will be provided.

4.2.1. aerOS Infrastructure Element

As mentioned, the essential functioning of the Meta-OS of aerOS is based on the establishment of IEs and 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 the continuum is, but share several characteristics in common. The aspects 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 board with a prepared OS (e.g., Raspberry Pi), iii) nodes in a KubeEdge deployment (*cloudnode* and *edgenodes* alike), and iv) virtual machines. Other examples might be envisioned.

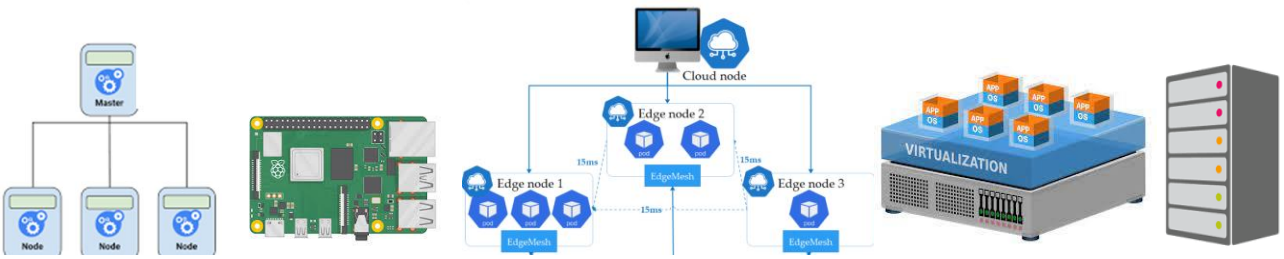


Figure 8. Possible Infrastructure Elements in a continuum

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. This will include total resources, ownership, list of peripherals available, and OS, among many others. Within these definitions, it is expected that enough descriptive fields will be created 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 may be used (at some stages, to be determined in the next iteration of the architecture) to indicate "flavours" of IEs. This way, IEs could be more easily "tagged" depending on the very spot in the continuum that they would be located in. 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 3.1, IoT devices are a crucial part of "continuum" deployments, as they can be considered both productive elements (providing context, monitoring, and actual exploitable data) and also potential receptor elements that could be orchestrated globally in the Meta-OS. In this regard, the characterisation of IoT devices within the continuum topology/architecture will be 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 would be 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). Specific examples and a thoroughly detailed description of the role of IoT sensors in the aerOS topology will be provided in the next deliverable (D2.7 – M21).

In addition, although the IE will be the most granular entity in aerOS, there will be no need of directly mapping one single "computing entity" to one aerOS IE (1:1 relation). It will be possible that several computing entities

are joined and seen (from the continuum perspective) as a single IE. These specific configurations will be the user's choice (system administrator); thus, enhancing the trait of flexibility of the aerOS architecture.

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. Pragmatically, domains serve for separating IEs that have certain aspects in common (for instance, the same containers deployment framework, the same geographical spot, the same local network environment, or the belonging of the same owner/stakeholder). This concept has been further explored in Section 4.1.

4.2.2. aerOS decentralised orchestration

At the very core of aerOS proposition is the capability of accomplishing smart, automatic, decentralised decision making in terms of 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 4.2.3), the Meta-OS is ready to unleash its orchestration power. The other two sub-sections (Section 4.2.1, Section 4.2.3) expose how to deal with the former (IE, Distributed State Network of Brokers (DSNB)). Here, the approach behind aerOS orchestration is portrayed.

As per the original conception of aerOS, an orchestration process should be 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 will include 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. As initially conceived, this module was called Federated Orchestration Module (FOM).

However, after the analysis of the state-of-the-art and the evolution of the technical design of aerOS architecture (depicted in this document), it has been decided that the decentralised decision-making – responsibility of that former FOM- is now 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 4.3.8).

According to the new structure, the decentralised decision-making, materialised in services orchestration decisions, is divided into a **HLO** and a **LLO** (as mentioned in Section 4.1). 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 will optimise parameters such as latency, it will 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 will allow 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 will lie.** 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 will be expressed through the selection and specification of a finite set of predefined requirements, which will be properly defined after a thorough process involving staff with previous experience in the subject (e.g., technical task leaders and use case scenarios leaders). After the definition of the deployment intention in the Management Portal, this component converts the request into an *Intention Blueprint*, which needs to be forwarded to the aerOS Federated Orchestrator module (former FOM). 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.

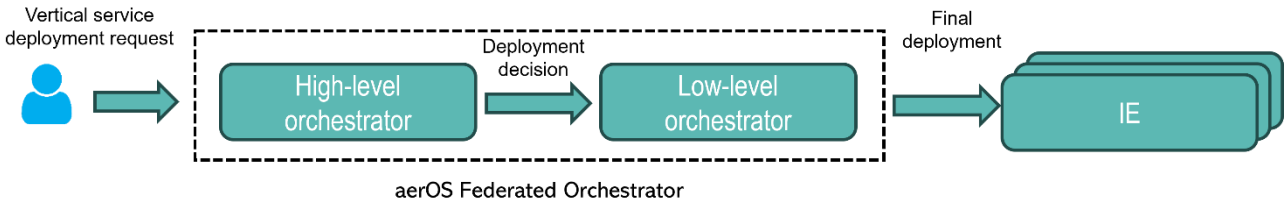


Figure 9. aerOS two-level structured orchestration for decentralised decision-making

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 will have 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 (templated) *Decision 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 orchestration technology of the domain. For example, it will rely on the APIs of K8s, Docker Swarm, an API on top of Docker, etc. Therefore, within a single domain, several LLOs may exist – as many as needed to properly ensure actual execution of workloads in all the IEs of that domain. In that sense, the *Decision Blueprints* coming from an HLO will be interpreted differently by each LLO in order to provide the expected (deployment) functionality. In aerOS, it is expected that LLOs will be custom created to cover the most common and state-of-the-art orchestration frameworks. For instance, a Custom Resource (CR) will be created in the case of a K8s cluster, to be interpreted by a controller to run the workloads (K8s operator pattern). Other examples might be found and will exist.

The first step of the process in Figure 10 (vertical service deployment request with specific requirements) is intended to be conducted in the aerOS Management Portal (described in depth in the section 4.3.8), which will provide 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 4.1).

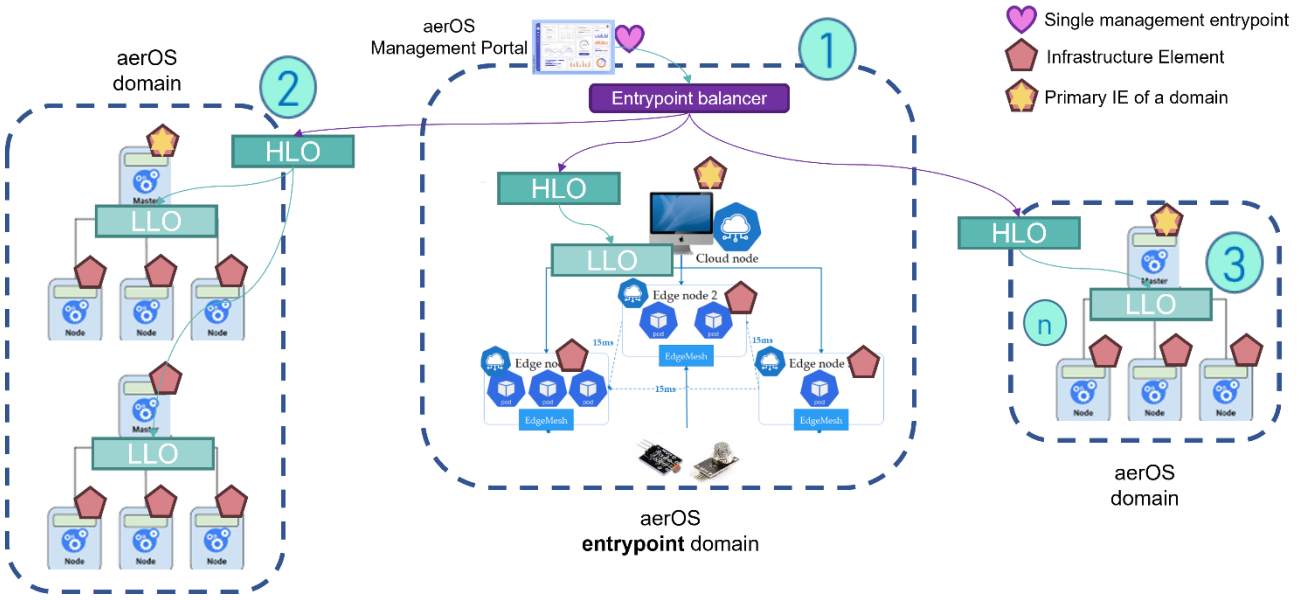


Figure 10. Entrypoint domains in decentralised decision-making of aerOS

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 centralisation 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 foresees the development of a specific component here, called *entrypoint balancer* – to be based on the state-of-the-art load balancing algorithms – that will 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 will interact with the appropriate HLO (as selected by the entrypoint balancer). The HLO will receive the *Intention Blueprint* and will execute the necessary logic to conduct the allocation decision. This logic will include a rules engine (to dismiss the IEs that do not meet the specific requirements) and the usage of judiciously selected frugal ML algorithms[1] that will optimise the allocation based on the current and forthcoming state of the continuum. For the latter to happen, the HLO will benefit from the federation of domains exerted by aerOS Federator. As conclusion, the HLO will translate the order down to any IE(s) within its domain (if they have the capacity) or will offload the first-level orchestration onto an HLO of another domain to repeat the process. Once the final deployment decision is made, a *Decision 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, will guarantee that other HLOs in the continuum will be requested first, thus distributing better the "needs of the first processing of the decision": In other words, the main differences between both are:

- 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 will always be as many HLOs as domains exist.
- The HLO will interpret the workload that must be executed, the *Intention Blueprint*, the state of the continuum, etc. and will take an allocation decision. This requires (likely, heavy) processing to be performed by the IE where the HLO lives. On the contrary, the *entrypoint balancer* will not analyse the information of the workload to be executed. It will just apply 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) will receive the *Decision Blueprint* (it will always have the same format) and will interpret it in a way that actual workload deployment can take place on its underlying, controlled IEs. Here, custom developments are expected.

[1] 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.

As a live system, aerOS will allow 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 will receive a reallocation request, then make a new allocation decision, and finally will send commands (*Infrastructure Blueprints*, similar to *Decision Blueprints*) 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. These mechanisms (the inner nuances of it) will be described in the next iteration of the deliverable.

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. At this stage of the project, 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 to 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 of the entrypoint balancer is emphasised, because without this component the same domain will always be used as the entrypoint for the orchestration request, risking unnecessary overloads Following the aerOS approach a new centralisation point will be 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 UI) will be able to specify only a subset of requirements, or give overall, vague instructions about the domains/IEs for prioritisation or selection purposes. The portal will forward 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" could 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. The granularity of this specification will be properly fine-tuned in the development of the orchestrator. In this scenario, the *Intention Blueprint* will be directly forwarded to the specific HLO, so the entrypoint balancer can be bypassed.

4.2.3. aerOS distributed state repository

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 will 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 IEs.

Concretely, the concept of a "distributed state repository" has been envisioned. 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 every domain. These elements, materialized as Context Brokers, have the capacity of exchanging contextual information based on established mechanisms following the widely adopted standard NGSI-LD[2]. To understand the usage of this mechanism in aerOS; it is useful to reflect on some of the principles of this standard, which is promoted by ETSI.

In NGSI-LD, the world is seen as consisting of entities. An NGSI-LD entity has:

- **an entity identifier** that uniquely identifies the entity

[2] https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.07.01_60/gs_CIM009v010701p.pdf

- **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 have values and sub-attributes, which are basically pieces of metadata that describe the attribute.

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, **IEs** will be the key entity to handle in the context of orchestration. Therefore, NGSI-LD offers a standard to perform the modelling of the resources to orchestrate throughout the whole aerOS continuum. With that background, the aerOS state will be stored as a number of entities –**IEs-**, having a determined number of attributes each, living in a distributed network of Context Brokers.

In practical terms, the distributed state of aerOS will be handled by instances of Orion-LD[3], the NGSI-LD compliant context broker, currently under continuous evolution. Specifically, the distributed operations feature of Orion-LD will be utilized for this purpose. More particularly, the distributed operation in NGSI-LD will be set up using context source registrations (from now on shorted to just "registrations"). A registration is a mechanism to inform an NGSI-LD broker about where to find more (non-local) entities. Upon queries, an entity is not only looked up in the local store of an NGSI-LD context broker, but also the registrations are consulted and for all matching registrations, a distributed request is sent to the broker behind the registration, and its information (its entities) is appended to the final response.

Using this distributed mechanism of NGSI-LD as implemented in Orion-LD, aerOS can query any of the brokers in the Distributed State Network of Brokers (DSNB)and get the same response, as long as the registrations are set up correctly.

For this to work, every broker needs to have (at least) one registration for each and every other broker of the DSNB (see Figure 11).

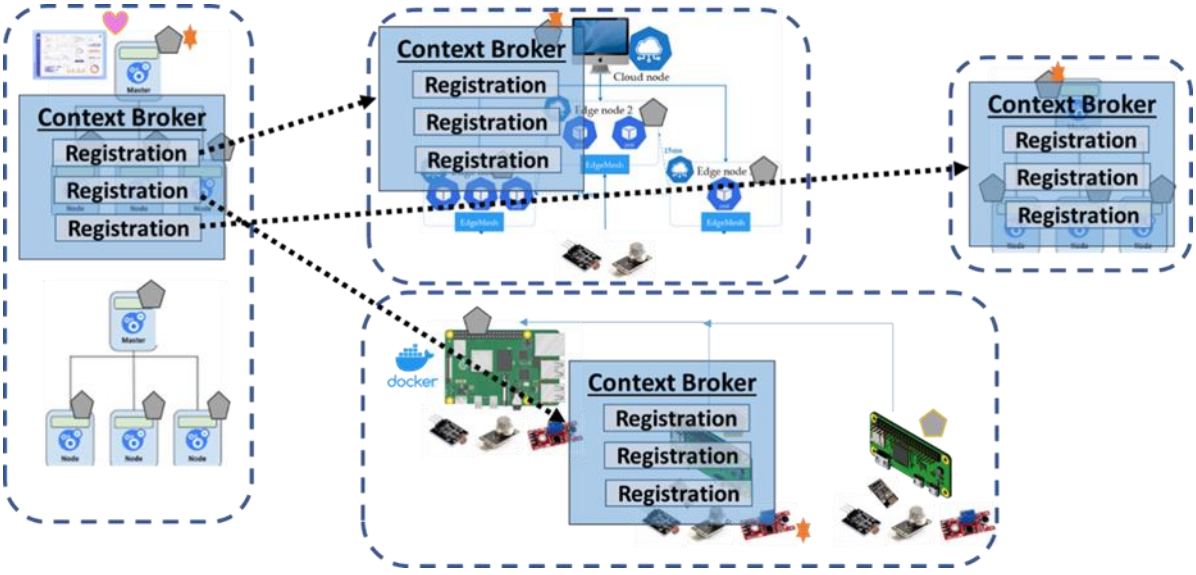


Figure 11. Example of a Distributed State Network of Brokers

NGSI-LD is quite versatile and offers several ways to configure the registrations for the distributed state. Specifically, the following two approaches of configuring the registrations are currently under analysis:

- The first approach considered in aerOS is to have every broker in the DSNB replicating the entire state of the whole continuum. This would be achieved by using "inclusive" registrations and have creation/modification of entities/attributes turned. Replication offers the advantage of faster queries and less vulnerability against network issues. The replication mechanism entails that any of the brokers in

[3] https://github.com/FIWARE/context.Orion-LD

the DSNB that receive a creation/modification request will forward the request to all the other brokers in the DSNB.

- The second potential approach proposes having each broker in the DSNB taking care of a part of the distributed state. Here, during queries, each broker should forward requests to the other brokers in the DSNB before giving the final response. This would be done using "exclusive" registrations and while the updates would be fast, queries would be slower (no forwarded requests on updates, only on queries).

Upon examination, it becomes clear that each approach has its own distinct set of advantages and disadvantages. In the coming months, a comprehensive analysis will be conducted to determine the most suitable solution for configuring the registrations for the distributed state, taking into consideration the specific performance requirements of aerOS. Subsequent iterations of this deliverable will describe the reasoning behind the selected approach and offer a more detailed description of it.

4.3. aerOS basic services

The aerOS basic services are defined as those that make the innovative computing continuum posed by aerOS Meta-OS work. They are a series of functionalities, and combination of those, that set up the whole ecosystem required to perform aerOS workload distribution and execution. Basic services are, also, run on top of IEs and rely on the aerOS runtime (see Section 4.2) to exist. According to the architecture design of aerOS, the basic services are: (i) network and compute fabric, (ii) Data Fabric, (iii) Service fabric, (iv) cybersecurity components, (v) self-* and monitoring capabilities, (vi) decentralised AI, (vii) common API and (viii) Management portal.

4.3.1. Network and compute fabric

Network and compute fabric refers to the interconnected infrastructure of network and computing resources within the IoT-Edge-Cloud continuum environment. It is a foundational architecture trait that provides the underlying fabric for data exchange, resource sharing, and workload distribution. The network fabric constitutes the backbone of communication and ensures that the data can be quickly and reliably transmitted between various components, while the compute fabric refers to the collection of computing resources (IEs) that provide the necessary processing power to run various containerised workloads. The binding of fast connectivity and ubiquitous computing supports services that are offered in the edge part of the network where the user equipment resides along with a plethora of IoT devices.

The key features of the Network and compute fabric, which is a critical element of the aerOS, are:

- **Scalability**: The fabric should have the ability to expand seamlessly, without sacrificing performance, as new servers, devices, or workloads are introduced to the aerOS ecosystem.
- **Reliability**: The fabric should be resilient, robust and ensure 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**: The fabric should demonstrate high throughput and low latency in order to ensure efficient communication between the different components.

aerOS aims at transforming 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 main goal here, in aerOS, 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. With aerOS, the network and compute fabric will abstract the connections via name (or other handy commodity) with established mechanisms, underlying, that will be in charge of performing the proper connection. The most prevalent design and implementation concepts considered for the aerOS compute and network fabric were the following:

- **Containerization runtime**, for all workloads deployment, management and orchestration services of course included, as it is a critical concept for managing and deploying services in a consistent and isolated manner across all administrative domains' infrastructure.
- **Container orchestration framework**, including all K8s flavors, which provides a coherent and common layer for managing containerized workloads and resources, which is a key aspect of the Compute Fabric.
- **Service Discovery** and transparent service communication, as a service mesh provision, to facilitate seamless communication and connection between services within the fabric.
- **API Gateway** to serve as the single-entry point for external clients to access the services within the fabric, handling authentication, routing, and other features (Section xxx).
- **Network Virtualization** abstracting and managing network resources within each domain to ensure programmable communication capabilities and efficient data transmission.
- **Edge computing nodes** (integrated as IEs) which play a crucial role in bringing computing capabilities closer to the data sources and users.
- **Load Balancing and Traffic Management**, provided by API gateway ingress component, to ensure incoming requests distribution across multiple instances of services.
- **Resilience and Redundancy**, provided by orchestrator frameworks, to ensure continuous operation and fault tolerance in case of component failures.
- **Monitoring and analytics** to provide insights into the performance and health of the management and orchestration services.
- **DevOps and continuous deployment** to facilitate rapid and reliable updates to the management and orchestration services.
- **Resource management and scaling policies**, as provided by orchestrator framework, providing dynamic resource allocation and scaling of computing resources.

For the implementation of most of the above features aerOS builds on top of compute and network orchestrators as is K8s (among others foreseen), when appropriate extends capabilities of containerization platforms like Docker, and exploits VIM flavors (e.g., OpenStack.) and their virtual networking capabilities to deploy IEs on top of them. **S**ecure gateways or virtual private networks (VPNs) between the administrative networks of different domains are foreseen to allow for encrypted communication and secure connectivity between the data fabric nodes and deployment request endpoints across domains.

4.3.2. Data Fabric

The IoT-Edge-Cloud continuum introduces a highly distributed and dynamic data landscape. With the goal of providing a holistic view of the data available in the IoT-Edge-Cloud continuum, while enabling data governance policies that can ensure a responsible use of data, aerOS aligns with the two recent data management approaches, 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. In this sense, aerOS embraces the **data as a product** thinking, as proposed by the data mesh paradigm. Owners of data providing domains are responsible for turning their raw data into high-quality data products that data consuming domains can easily discover, understand, trust, and access. However, building data products requires data engineering skills, as well as following standard data models and standard interfaces for enabling interoperability with data consuming domains. To help data providing domains create data products, aerOS proposes a self-serve data infrastructure that follows the architecture defined by the Data Fabric paradigm.

The **Data Fabric** is a metadata-driven architecture that automates the integration of data from heterogenous sources and exposes the data through a standard interface. Aligning this architecture with the management of data as a product means: i) the Data Fabric transforms the raw data of the providing domain into a data product that follows a standard data model; and ii) the resulting data product can be shared with consuming domains through the standard interface of the data fabric.

The **knowledge graph**, as described in [9], represents a promising technology for the realisation of the Data Fabric architecture. Knowledge graphs enable representing concepts that relate to other concepts, by semantically annotating data through ontologies. In this regard, aerOS understands a data product as the conceptual representation of the physical (raw) data, as depicted in Figure 12. 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 data fabric.

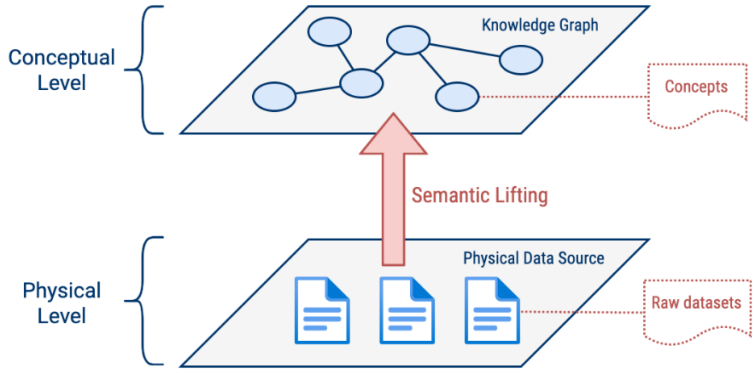


Figure 12. Lifting physical data into concepts.

Besides 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 these data governance services. For example, when it comes to cataloguing data, the knowledge graph would contain metadata that describe the owner of a data product and the domain that the data product belongs to. 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.

In short, the knowledge graph acts as the heart of the aerOS Data Fabric as depicted in Figure 13, integrating both metadata and data from the continuum.

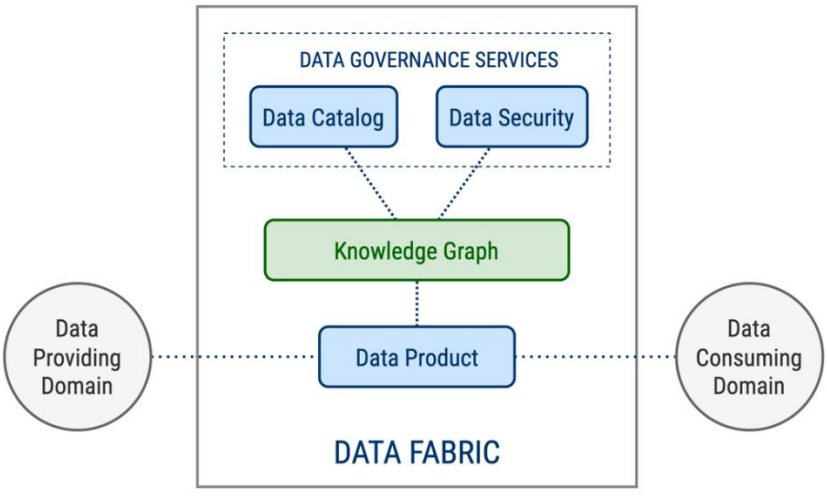


Figure 13. Logical architecture of aerOS data fabric

To realise the knowledge graph, aerOS has adopted the ETSI NGSI-LD standard [10]. NGSI-LD 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, i.e., knowledge graph. 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 yet powerful REST API, makes NGSI-LD a promising standard for implementing knowledge graphs, compared to other traditional graph standards, such as Resource Description Framework (RDF) and SPARQL Protocol and RDF Query Language (SPARQL), which are verbose and have a steep learning curve.

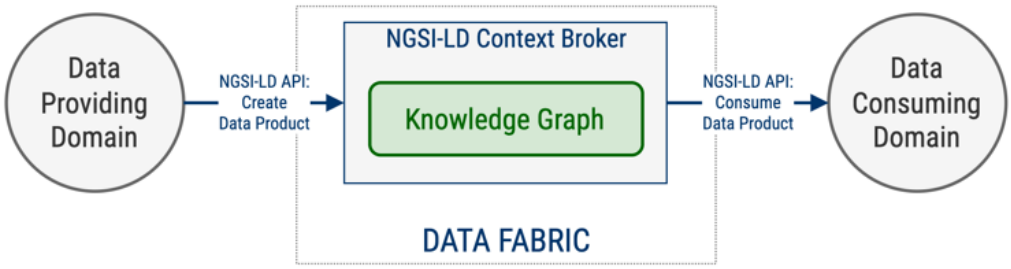


Figure 14. Creation and consumption of data products through the NGSI-LD Context Broker

In terms of the NGSI-LD standard, the NGSI-LD Context Broker is the component that stores and exposes the graph data. Therefore, the data products built in the Data Fabric are ingested and served through the NGSI-LD Context Broker, making them part of the knowledge graph (Figure 14).

4.3.3. Service fabric

aerOS service fabric is designed around a microservices architecture, breaking down services into independent and loosely coupled microservices. It abstracts the underlying infrastructure complexities, providing a unified interface for developers to create, deploy, and manage distributed IoT applications and services seamlessly across diverse aerOS domains over a continuum consisted of heterogenous compute resources integrated as aerOS IEs. All aerOS, basic and auxiliary, services are defined with clear boundaries and responsibilities, enabling specialised functionalities and independent deployment. The designed service fabric includes robust lifecycle management tools and APIs, facilitating dynamic scaling based on workload demands to ensure best workloads placement and optimal resource utilisation.

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 double 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 computing resources are available. Service fabric in one hand receives users' deployments requests, as submitted in the form of templated submissions compiled in the Entrypoint dashboard (called Intention Blueprint), and on the other hand has an overall knowledge of the status of all IEs and domains across the continuum based on the federator provided services, as implemented within aerOS data fabric. HLO, which acts as an AI-powered Decision-Making Engine as it interfaces with AI/ML services, produces the allocation decisions which are forwarded, in the form of Decision Blueprints, to a local or remote (in other aerOS domain) LLO which will proceed with the actual decision enforcement, i.e., workloads placement, on the underlying IEs. aerOS embedded analytics tool, implemented as FaaS within aerOS domains, further support LCM of IoT applications. Triggered over HTTP API calls and integrating data queried from the Data Fabric, they are utilised to provide LCM decision functionalities to the aerOS system.

Even more aerOS service fabric design foresees for additional features integration which enhance the overall provisioning of a safe and robust environment for secure IoT services deployment. Inter-service communication is facilitated through well-defined APIs and protocols, supporting various patterns like request-response, pub-

sub, and message queues. Integration with an API gateway (Section 4.3.7), which serves as the single-entry point for external clients per aerOS domain, handles authentication, routing, and request/response transformations. Security measures, including authentication and authorization mechanisms, protect services and data, controlling access based on user roles and permissions. Monitoring and logging capabilities provide real-time insights into service performance and health. Finally, by interfacing with DevOps practices and aerOS DevOps suite, automated testing, continuous integration, and deployment, are enabled and ensure rapid and reliable updates.

4.3.4. aerOS cyber security components

aerOS provides high-level cybersecurity capabilities that contribute to the protection of the whole aerOS ecosystem protecting its resources from unauthorised access and cyberattacks. The main aerOS cybersecurity functions focus on three axes: a) authentication, b) authorisation, and c) trust management. These functions will be supported in aerOS by implementing two cybersecurity components, the Identity Management and the Trust Management. The Identity Management will be responsible for the authentication and authorization procedures, while the Trust Management will establish and maintain trustworthiness in the IE and aerOS domain.

More specifically, the Identity Management component will be based on the Keycloak IdM[4] that offers authentication and authorization functions and can be easily incorporated in aerOS to enhance its cybersecurity capabilities. Elaborating more on the authentication, its main activities focus on users' authentication by employing innovative authentication mechanisms to protect aerOS from unauthorised access. aerOS portal will be the main gateway between users and aerOS; thus, the authentication module will be linked with the aerOS portal to authenticate the users. Authentication will be based on OpenID connect[5], which is an authentication protocol that is built on top of the Open Authorization (OAuth) 2.0 framework and provides a secure token-based authentication and Single-Sign-On (SSO) capabilities; namely, the users' credentials (e.g., obtained from their organization) can be used to access also the aerOS portal. The authorization in aerOS is based on Role-based Access Control (RBAC) mechanisms, where access policies are implemented to allow users accessing only specific parts of aerOS based on their roles. For instance, an aerOS user with the role "aerOS Application User" will have access only to the parts of aerOS related to the application that is allowed to use (further information about authentication and authorisation in aerOS can be found in D3.1).

The Trust Manager component is responsible for measuring the trust of the IEs of an aerOS domain. This happens by retrieving some attributes from the IE (e.g., security events, health score, active services, firmware version, etc.) that will be later parsed to a weighted function, namely different weights will be assigned to every attribute based on their significancy (e.g., a security event will have a higher weight than firmware version) to calculate the trust score of each IE. Furthermore, the Trust Manager will also calculate the trust score of the whole aerOS domain based on the IEs that belong to the domain. The trust scores will be shared to the aerOS administrator in order to take the necessary actions (e.g., suspend or remove an IE).

To sum up, aerOS main cybersecurity capabilities are based on the implementation of an innovative token-based authentication along with RBAC authorization policies to avoid unauthorized access, and trust calculation and management to establish trustworthiness in aerOS domains.

4.3.5. aerOS self-* and monitoring

Explanation of the service:

This basic service is materialised in a suite of automated self-* features (microservices) that an IE will be continuously applying to itself. In a widely varied environment where a large number of IEs will co-live, each of those should have a set of capabilities to modify their behaviour / status in such a continuum. Those capabilities, in combination with the global orchestration and data management, will allow the IEs to still be considered empowered entities, playing a crucial role in a large environment, thus reinforcing the decentralisation principle of aerOS.

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

Functionally, this service supports the monitoring of inner parameters (some that are exposed to the whole continuum and some that are not), 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). This service also includes the management of the IoT devices that will be attached to the IEs, offloading the need of central management of configuration, control, or healing. It will also involve certain cybersecurity traits, to relax the demands of centralised security/privacy control. Besides, it will implement active attempting to recovery after abnormal activities, dependability, and long-term, up-to-date synchronisation with its custom configuration and horizontally scaling of resources. All the functionalities of this basic service will have in common the automation degree (will be automated) and the dynamicity and capacity of parameterisation by the user of the continuum.

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 will run as a suite of microservices, each of them tackling a specific functionality, that will be installed in the IEs (depending on their flavour or role in the continuum). These microservices will be 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 D3.1.

4.3.6. aerOS decentralised AI

In aerOS, AI can be analysed from an internal and external perspective. External AI support should fulfil 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 on a prediction model (e.g., air quality prediction model). In particular, external AI is a specific task that may have corresponding workflow and can be commissioned to be executed on the aerOS infrastructure.

Internal AI will support aerOS continuum internal management by providing intelligent decision-making that spans across different base services and respective components. Examples of possible internal AI can be self-* or intelligent network reconfiguration. As part of these services, AI-related functionalities can be deployed in different locations in the continuum. AI-related functionality can be “built-in” in specific services being part of an internal component (e.g., already trained model 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). In the former case, these will be design considerations for specific base services developed in aerOS. In the latter case and in the case of external AI, aerOS aux AI services (see Section 4.4.1) can be used to provide execution environment for AI-related functionalities (e.g., deploying trained model for inferencing, training a model) and support for additional features such as frugality and/or explainability/interpretability. Aux AI services are not obligatory in the aerOS deployment but can be included depending on the characteristics of the use case.

4.3.7. 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 updates 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.

4.3.8. aerOS management portal

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. The project foresees the creation of 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 will be deployed in a unique domain, named as **"entrypoint domain"**. However, the portal will provide migration capabilities to be moved to another domain or IE due to a user requirement or an unexpected failure. Therefore, this will 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 4.2.2.

This user-friendly dashboard, developed as a modern web application, will act as a frontend for performing operations regarding the Meta-OS, which will finally be performed by the aerOS Basic Services in the background. For example, an administrator will be able to add a new domain to the continuum using the portal, but the inclusion of this domain will be managed by the aerOS Federator component. The interaction between the User Interface (UI) of the Management Portal and those basic services will be undertaken by the backend of the aerOS Management Portal. Furthermore, this dashboard will display 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 will, however, serve as the single window access for interacting with the continuum.

This set of capabilities or actions which can be performed by users will be properly defined based on previously specified requirements by technical developers and potential end users of aerOS (e.g., use cases leaders). Thus, aerOS envisions the definition of a set of roles and permissions to be applied to end users. Using a more practical

example, the content that will be displayed in the dashboard will change based on the role of the logged users, e.g., an administrator will only be able to deploy certain services in a set of domains on which they have rights and permissions, as an example. In addition, a common list of users must be implemented for all the domains that compose the continuum, as described in the aerOS AAA block of the architecture (Section 4.3.4).

However, the aerOS Management Framework reaches beyond the capabilities offered only by the Management Portal. It also includes other management tools that will 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 & 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 4.2.3 (aerOS distributed state repository), this block will be 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), will play a major role in this architectural block.

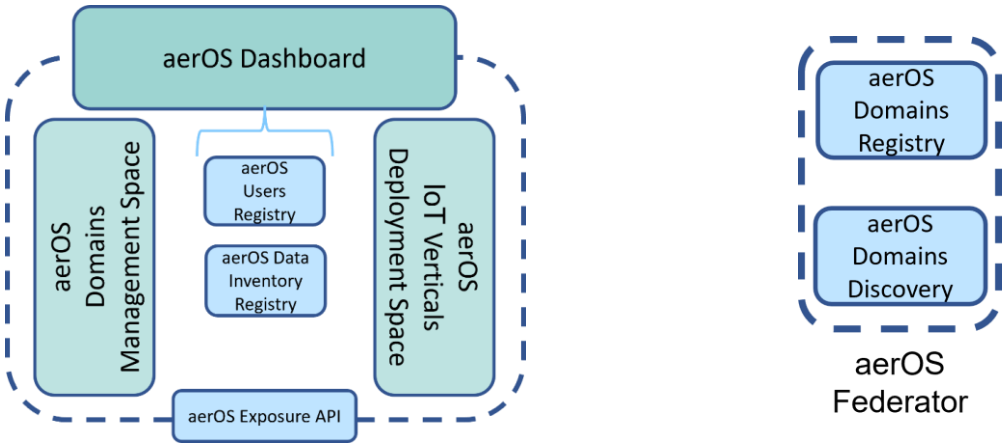


Figure 15. aerOS Management Framework (left: aerOS Management Portal, right: aerOS Federator)

4.4. 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 (control of workflows in the continuum and use cases deployments) and (ii) Embedded Analytics.

4.4.1. 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 decentralised fashion without moving the data to a central location (e.g., cloud). 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. This may require application of additional mechanisms

to support frugality, i.e., techniques that allow to train and later use AI models in resource-restricted conditions (limited processing power or memory, low network bandwidth). Finally, to provide accountable and trustworthy AI-driven solutions explainability/interpretability of AI models should be considered.

Here, the auxiliary AI to be deployed in aerOS can be divided in two main blocks:

4.4.1.1. Control of AI workflows in the continuum

aerOS aux AI services will cover different functionalities required to execute AI tasks using aerOS infrastructure. aerOS will allow to execute AI tasks described as workflows. AI tasks can be decomposed into sub-tasks that may have dedicated requirements and their execution can span over several IEs. In the simplest case AI task may have only one sub-task, i.e., workflow consisting of one step. Note that AI task is a specialisation of a general task that can be executed using aerOS infrastructure.

Within aerOS dedicated services will be prepared to: (1) interpret task description / workflow definition (AI Service Controller), (2) monitor specific task execution (Task [n] Controller), and (3) execute an AI sub-task (AI Task Executor).

Frugality support in aerOS will cover mechanisms such as: one/few-shot learning, heterogenous FL (FL with consideration of clients' characteristics and processing capabilities), model reduction and availability of TinyML and compliant services deployment. These techniques will be realised as: (1) preparation of "minimised" AI-related services to be deployed on aerOS, and (2) preparation of services that will expose functionality to reduce an AI model.

The objective for explainability support in aerOS is to prepare mechanisms to optionally "plug-in" such functionality in the AI workflow for an AI task execution. Popular and widely used metrics will be judiciously selected to provide respective functions and to study trade-offs between explainability, cost in the edge (explainability tools vs IE configuration), and acceptable model accuracy (which is proved to be lower for better explainable methods).

Note that frugality mechanisms and explainability/interpretability may have influence on AI model accuracy (and other metrics) so their inclusion in AI task workflow will be optional.

4.4.1.1. Explainable AI for pilots

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 is 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.

aerOS will explore the usage of XAI in the pilots for validating the technology. Here, although not directly related to the architecture, it is likely that successful validations will lead to a set of operational and technological recommendations.

4.4.2. Embedded analytics

The aerOS embedded analytics tool can be compartmentalised into three roles; these are the analytics framework, function authoring and visualisation. This section will detail each of these roles with reference to the aerOS embedded analytics tool architecture and integration with cooperative aerOS services.

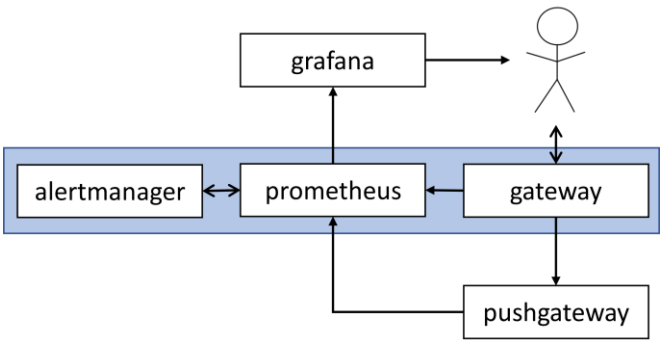


Figure 16. Embedded Analytics Tool Architecture

The aerOS embedded analytic tool (Figure 16) provides a framework for users to design, implement and deploy functions to generate insights into data and provide decision-making and data processing for other components upon request. The aerOS embedded analytics tool functions are triggered over an HTTP API call, where information from the call and data queried from the Data Fabric (section 4.3.2) are utilised to provide defined functionality to the aerOS system. This functionality is directed towards several project tasks such as T3.3 to provide insights for service orchestration (Section 4.2.2), T4.3 to provide sampling for Frugal AI (Section 4.4.1) and T4.2 as a data source provider for the Data Fabric (Section 4.3.2). A collection of additional functions will be provided to address those listed in the project proposal such as data stream statistics, advanced real-time composite indicators on performance and anomaly detection.

The aerOS embedded analytics tool also provides visualisation capabilities for in-function metrics to allow function authors to expose internal information directed towards non-technical users.

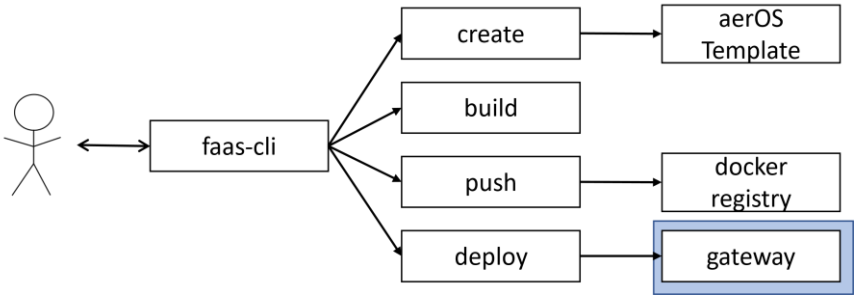


Figure 17. Function Authoring

The aerOS embedded analytics tool utilises the *faas-cli* tool for the creation, building, and deployment of the aerOS embedded analytics tool functions. This process is shown in Figure 17, highlighting the four primary functions of *faas-cli*. The *create* function utilises an aerOS specific template for the creation of functions integrated with the aerOS embedded analytics tool features, such as the exposing of in-function metrics for visualisation. The push function uploads a build function to a Docker registry, which allows for public/private access. This allows for functions to be deployed remotely with appropriate access permissions. The deploy function instantiates a new function on the aerOS embedded analytics tool. This may be a build function on the user's local machine or a function hosted on the Docker registry.

4.5. 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 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 should be 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**

This service would allow aerOS to have a kernel core that is mandatory, and all other functionalities 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 product-service 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 product-service 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 product-service 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 product-service 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 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 should enable the update of devices and systems in the product-service system. This can fix vulnerabilities and bugs and improve performance and functionality.
- *Authentication:* aerOS should enable the authentication of devices, systems, and users in the product-service system by using passwords, biometrics, certificates or tokens. This can prevent unauthorized access and ensure data integrity to the pilots' assets.
- *Encryption:* aerOS should enable the encryption of data and information in transit and at rest. This can protect data confidentiality and privacy coming from pilots' premises.

- Firewall: aerOS should enable 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.

5. aerOS Reference Architecture

Having introduced the key concepts and components of the aerOS architecture in the previous sections (Section 3 and Section 0) and adhering to the design guidelines outlined in Section 2, we now present the aerOS Reference Architecture (RA) through a set of viewpoints as identified within the described methodology. Firstly, we provide an overview of the entire architecture, from a high-level viewpoint, including an example IoT service deployment instantiation, and delve into different instantiations of aerOS Entities in subsection 5.1. Following that, we present the functional viewpoint (in subsection 5.2), which describes the role of the main components of the aerOS systems. Moving on, subsection 5.3 focuses on the process view of the RA, outlining how components communicate to accomplish fundamental functionalities. We provide activity and sequence diagrams that detail the domain and IE registration, IoT service deployment, and aerOS data discovery and usage. Subsequently, subsection 5.4 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 enabling the federation across aerOS domains which of importance in aerOS ecosystem is described. Next, subsection 5.5 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 5.6, we discuss the business viewpoint of the architecture, which serves as a guide for developing application components and supporting the decision-making process of stakeholders involved.

5.1. aerOS architectural views

A selection of architectural viewpoints that address aerOS stakeholders' concerns, capturing their requirements to provide for a consistent aerOS reference architecture description, have been selected to describe aerOS Meta OS. The selected viewpoints that are described in the section are the following: **(1) High-level view**, that describes interactions, relationships, and dependencies between the system and its environment, **(2) Functional view**, that describes the main functional elements of the architecture, interfaces and interactions, **(3) Process view**, that 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**, that describes data models, data flows, and how this data is manipulated and stored, **(5) Deployment view**, that provides a consistent mapping across the existing and emerging technologies and the functional components specified in the Functional View, and the **(6) Business view**. It addresses the business processes, organizational structures, roles, responsibilities, and strategic objectives that the system supports.

5.1.1. High-level view

aerOS initiates from the fact that there is a plethora of isolated processing units and private computing islands with restricted resources and lack of available services to implement and deploy holistic solutions for their rising IoT needs, either in the industry domain or for more personalized use. A large volume of computing units and private networks act as plain data concentrators and forwarding equipment in order to store and process a large volume of data in central commercial cloud infrastructures operated by a limited set of services providers. 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 serve similar needs. The proposed approach describes a safe, based on privacy and security enforcing mechanisms and technologies, federation of individual computing resources which render a network and computing continuum. This federation exposes an environment made up of a combined pool of resources which can transparently host parts (or the

entirety of) the intended IoT tasks as near as possible to data sources and with total control over the availability and governance of their data. Industry verticals, or individual users may provide "off the shelf" resources, even with restricted capabilities as long as they can support virtualised containerisation environments, which can be enhanced, based on a well-documented and easy procedure, as aerOS IEs or domains and be part of a great common execution ecosystem where their services can take advantage of existing available resources or services across the aerOS continuum. The following core objectives need to be fulfilled by the proposed solutions:

- 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.
- 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.

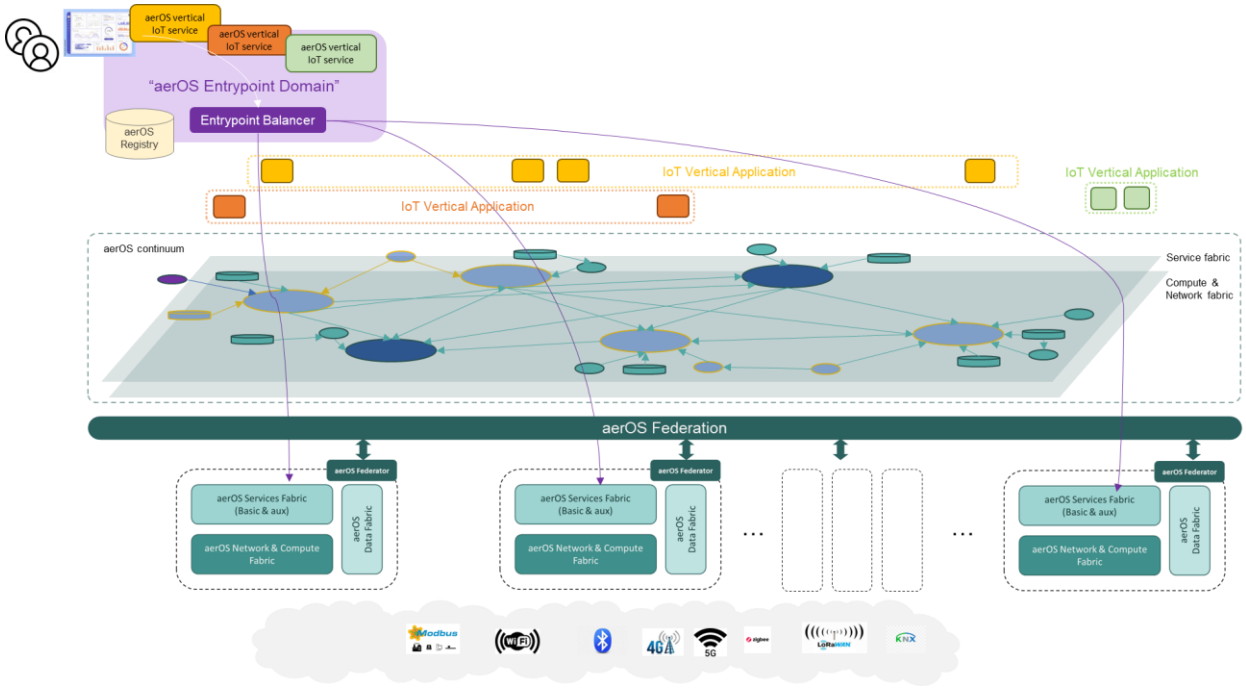


Figure 18. aerOS high-level View

The suggested architecture introduces a guided procedure for the deployment of an aerOS domain on top of the available compute 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 18 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 orchestrate IoT services deployment according to users' requests. Users' IoT services deployments requests are submitted in the "aerOS Entrypoint" which provides a guided, and based on a multitude of templates, IoT application deployment process. In some cases, when the requested workload can be expressed as an AI workflow, the submitted application, based on smart orchestration, is deployed within a user selected domain but can also be further broken down for which a more efficient placement might be needed based on their requirements on computing resources or data consumption. Parts of the AI workflow, that may need direct access to data, are placed in the edge aerOS domain

and parts of it might be deployed in a domain, in some public or private cloud, which can support more intensive processing tasks, or which exposes specific AI capabilities. The aerOS orchestrator takes care of this by leveraging all continuum federated information available and does this in a way completely transparent to the users. The result is that a pool of resources is hosting the entire application under a more efficient placement strategy and verticals can just receive requested services without knowing 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 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 as aerOS AI enabled federated orchestration takes care of hiding all the details related to the most efficient placement of services. 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 a component deployed within each aerOS domain. Data interoperability, a provision of aerOS data fabric, is important within aerOS architecture. Existing smart models will enforce this interoperability for industry verticals and an aerOS management information model, which is currently under development, will offer this homogenized status exchange among all infrastructure elements across the continuum.

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 hosting control. 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 jobs 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.

5.1.2. Functional view

Having briefly explained the overall process of registering resources within aerOS continuum and taking advantage of this continuum to deploy IoT services, this sub-seciton discusses the most basic functional blocks engaged in these processes and their placement within aerOS ecosystem.

Figure 19 presents the building blocks, taking part and interacting in the process of deploying resources and services within the continuum:

- **aerOS domain,** consisted of one or more aerOS **IEs,** which provides the connection point with the industry vertical. A common execution environment with aerOS federation and orchestration capabilities, enhanced with many more aerOS provided services, which provides the binding with the edge IoT applications, services or generally things. Computing and networking resources are provided by vertical users, but their applications are not restricted to be executed there and, the other way around, if available resources exist at some point they can be exposed and shared for other domains' services execution. Domains are integrated in the ecosystem and insights and management capabilities are incorporated into aerOS Entrypoint domain.
- **aerOS Entrypoint Domain,** one "more" aerOS domain but enhanced with capabilities related to ecosystem management. Migration, at run time also, of management services to another domain will not break the ecosystem functionality but will provide a new domain as an Entrypoint. aerOS domains state information is not centralised and will thus continue to propagate from edge to cloud and support appropriate resources federation. IoT developers and domain operators can have an operational access and overview of their services and resources.

- **aerOS Federated Orchestration and aerOS Management Framework,** are not tangible entities, but abstract concepts representing a set of combined aerOS services and information, which can provide at each aerOS domain a complete view of all the resources and the capabilities dispersed from the edge to the cloud and thus, it can provide the ability for its orchestrator to request services deployment on more efficient placements. aerOS Management Framework is mainly concerned with core entities registration (e.g., aerOS Users, Domains) and discovery, in order to form a federated environment. aerOS Management Framework is placed partly in the Entrypoint domain and within each aerOS domain federator component. aerOS Federated Orchestration is the -AI supported- orchestration of resources, both compute and network fabric and service fabric related resources, all over and across the aerOS federated from the edge to the cloud.

In Figure 19 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 enter aerOS 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.

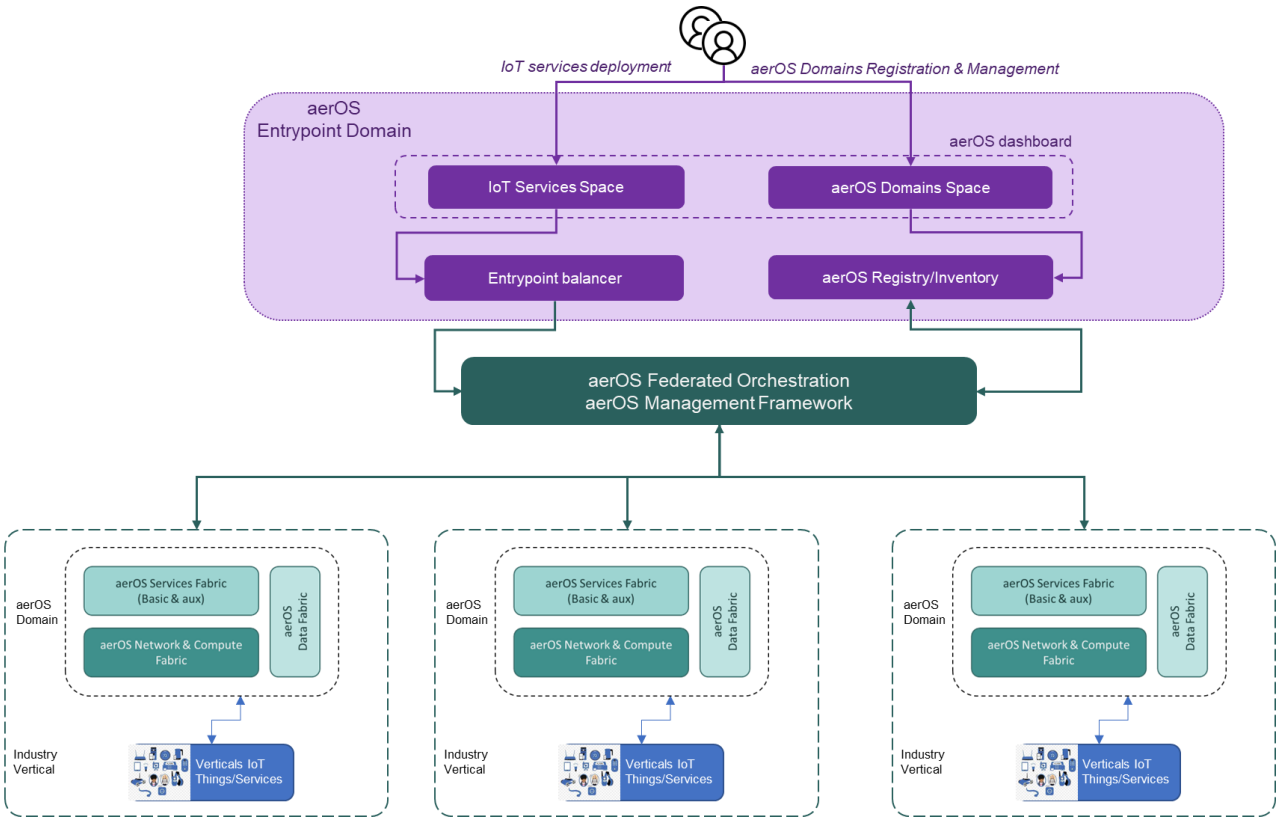


Figure 19. aerOS entities and actors overview

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 (this is sustained via two different processes in the Process View – see Section 5.1.3).

A more focused view on the functional blocks which implement these aerOS features in a domain level is depicted in Figure 20. 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 (authentication, authorization, access) component.

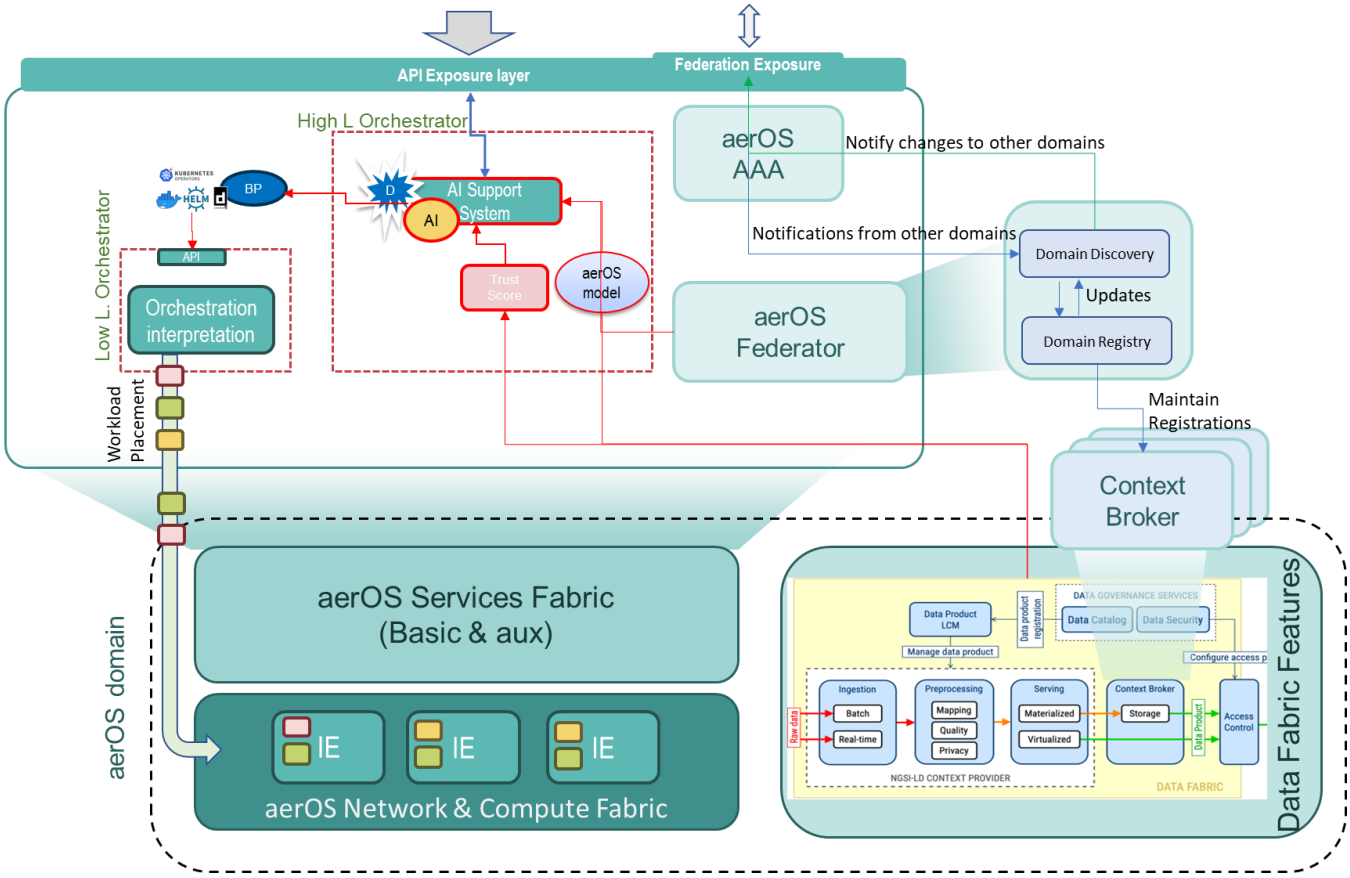


Figure 20. aerOS domain functional blocks

Figure 20 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.

5.1.3. Process view

The process view provides an overview of how the different aerOS processes (outlined in the previous sub-section) 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. At this stage, the in-depth work of process view in aerOS has focused on describing three main aerOS processes:

- Installation and aggregation of computing resources to the continuum ("*aerOS Domains Registration & Management*").
- Optimally deploy a service by leveraging the aerOS continuum orchestration processes ("*IoT services deployment*").
- Access data within the continuum, regardless of location.

These three processes provide an initial overview aimed at facilitating the understanding of Reference Architecture. However, subsequent iterations of this deliverable will offer more detailed descriptions of the aerOS processes and sub-processes that define this Process View.

Figure 21 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. After the successful completion of the installation, the SysAdmin proceeds to configure the domain and the IEs using the aerOS Management Portal. The SysAdmin has the following capabilities:

- Register a domain, ensuring its proper inclusion.
- Add IEs to an existent domain in the aerOS continuum.

In both cases, the SysAdmin receives confirmation that either the domain or IEs have been successfully added and are now available within the continuum.

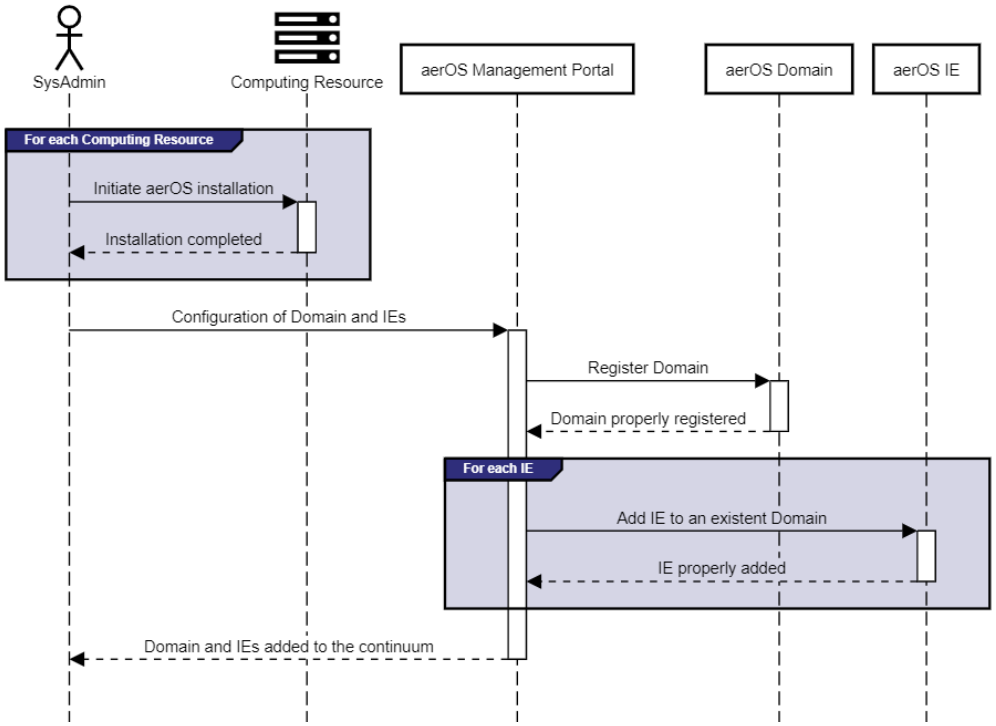


Figure 21. Installation and aggregation of computing resources to the continuum.

The next main process that has been detailed, represented as a sequence diagram Figure 22, 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.

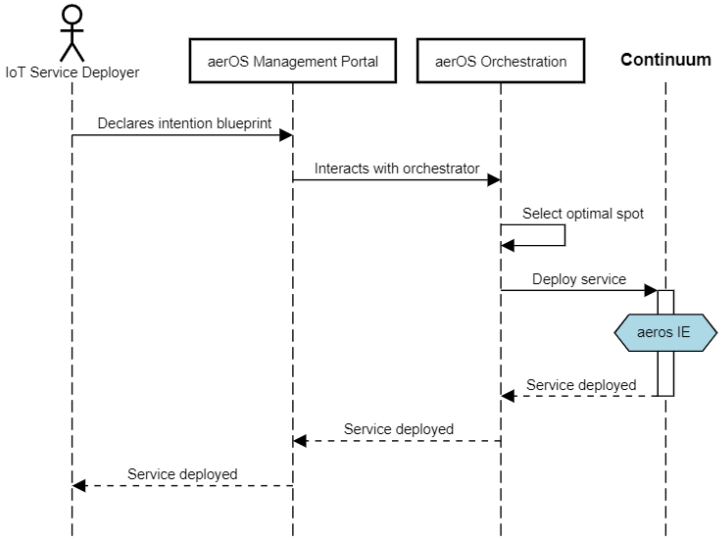


Figure 22. Optimally deploy a service by leveraging the aerOS continuum orchestration processes.

About the third main process mentioned for the Process View, Figure 23 describes a sequence diagram that exemplifies how to access data available within the continuum, regardless of its location. This process begins with a Vertical User requesting access to a specific piece of data within the continuum. Initially, aerOS AAA verifies the user's permissions and, if authorised, grants access to the requested information. Once access is granted, the user can interact with or retrieve the data from its closest Data Fabric. Then, Data Fabrics will, if needed, operate in a federated manner within the continuum, allowing data retrieval from either a Data Fabric located in the same domain as the request or from a Data Fabric in a different domain. Finally, the data is sent back to the Vertical User, completing the process.

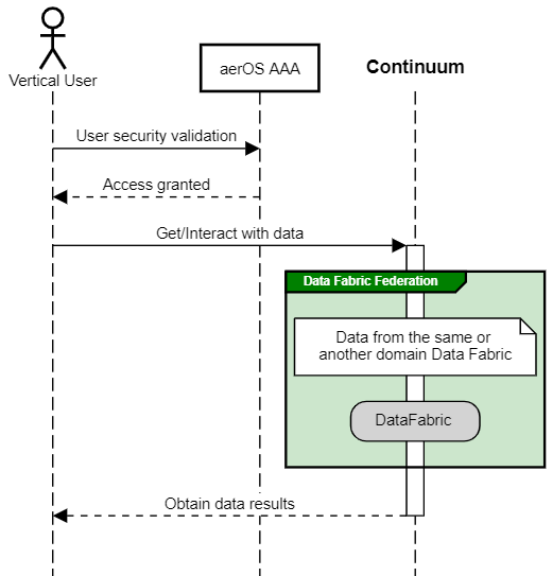


Figure 23. Access data within the continuum, regardless of location.

5.1.4. Data view

An aerOS domain may include multiple IEs, resulting in new data sources and data consumers dynamically becoming part of the distributed knowledge graph. To enable the exchange of data flows among different IEs, aerOS envisions a distributed architecture for the Data Fabric, as depicted in Figure 24.

In this architecture, the Context Brokers existing in a domain will store information about the state of the IEs and also what kind of data is available in them. They will also be aware of the rest of Context Brokers in their domain and elsewhere as well (federation). IEs will either implement Context Broker or Context Providers, depending on their computational resources and the spot in the continuum. Every time a data providing domain registers a new data product in its local Data Fabric, the rest of the Context Brokers update their Context Registries with this new information. As a result, when a data consuming domain requests a data product served by a others Context Brokers, the local Context Broker knows the neighbour Context Broker from which the data product can be retrieved. Based on this information, the local Context Broker interacts with the neighbour Context Broker and obtains the data product on behalf of the consumer. As mentioned in Section 4.2.3, there are two strategies to address such updates/registrations (inclusive or exclusive, mainly) and the selection of the particular one for usage in aerOS will be done in months to come.

Compared to archetypical distributed architectures in NGSI-LD, where there is a root Context Broker that centralises the exchange of data flows, the proposed architecture aims to be fully distributed. Each Context Broker knows who to ask for each registered data product in the aerOS domain; thus, the data flows are optimised and there is no longer a central component that can become a bottleneck. Additionally, this architecture adds more flexibility and scalability as there is no first class Context Broker and second-class Context Brokers. In the proposed architecture, all Context Brokers play the same role, thus, new Context Broker can easily join or leave the continuum.

Nevertheless, the envisioned architecture requires the implementation of mechanisms responsible for advertising the registration of a data product among the rest of the Context Brokers within the aerOS domain. These mechanisms could be seen as a control plane for the distributed Data Fabric. Technologies such as service mesh, from microservice-based architectures, will serve as reference for the implementation of such mechanisms.

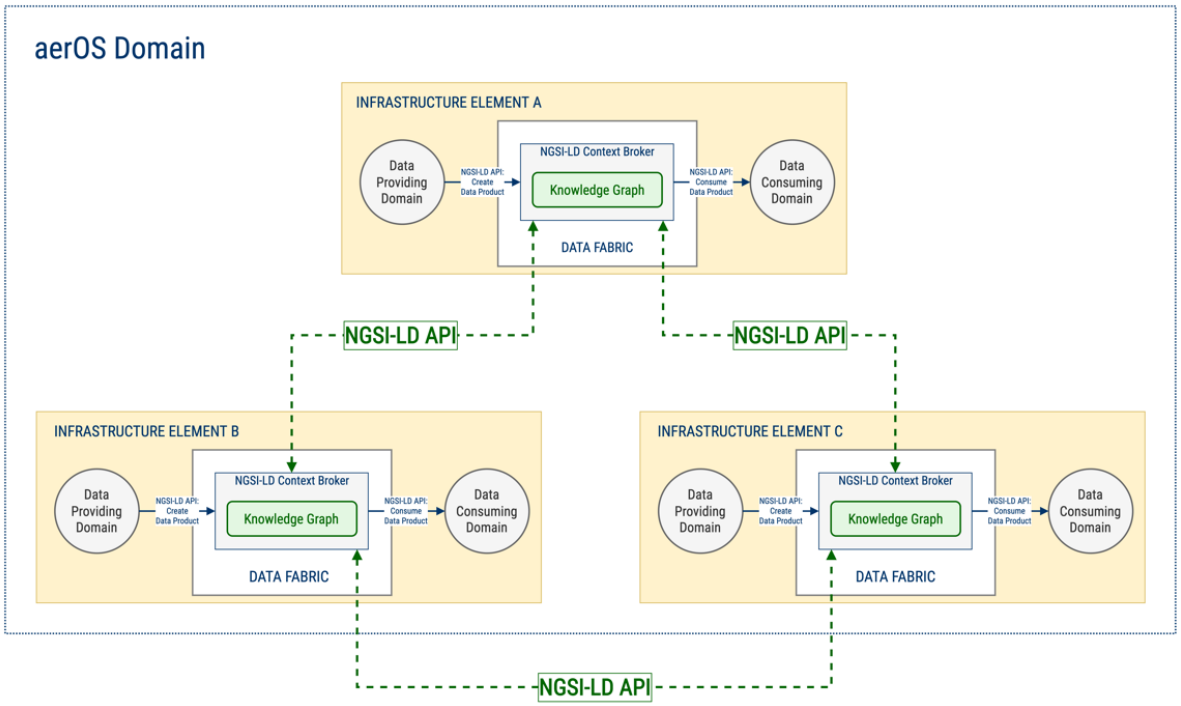


Figure 24. Data Fabric distributed architecture

On the other hand, an important characteristic of aerOS is the combination of multiple aerOS domains by means of federation. In this sense, the proposed architecture for the aerOS Data Fabric follows a similar approach. Figure 25 shows an example federation scenario with two aerOS domains, each domain running an IE with federation capabilities. A Context Broker running on an IE with federation capabilities for the first aerOS domain, registers its counterpart Context Broker from the second aerOS domain as the source for all data products available in the domain.

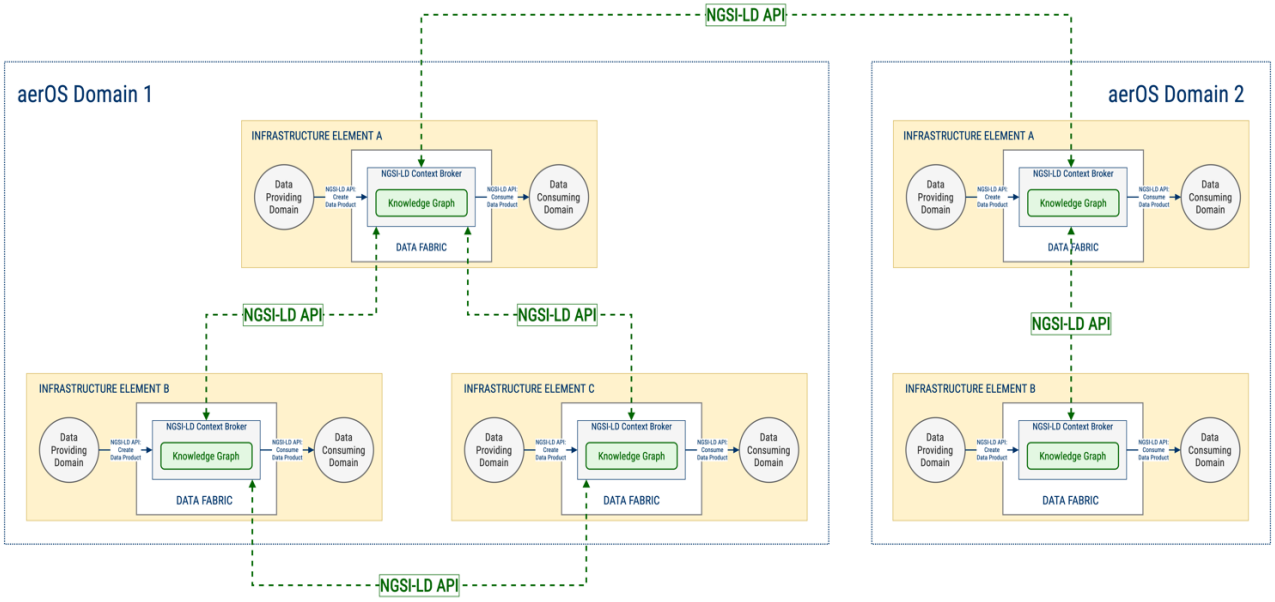


Figure 25. Data Fabric architecture in a federated scenario

5.1.5. Deployment view

Deployment view refers to the process of making aerOS systems and applications available and operational as in industry verticals use cases. It presents the system from an engineer's point of view while deploying, placing, 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.

The goal of aerOS deployment over available hardware is to render it to a set of IEs under an aerOS domain. This, fully documented, process will be supported by provided tools and scripts and will be a one stop process in rendering available computing and network resources to an aerOS domain running all aerOS basic services, and selected auxiliaries also, which will provide the entire API for connecting to the rest of the ecosystem. Having transformed all legacy hardware into aerOS IEs, a thin layer, the aerOS runtime (see Section 4.2), will seamlessly provide the basis for aerOS services to provide connectivity, orchestration, Data Fabric integration and interface to all other integrated domains. Figure 26 provides an example of aerOS deployment on top of stakeholder resources on top of probably heterogeneous hardware and operating systems. Guided and tools-supported aerOS runtime deployment will abstract underlying differences 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, which should also benefit by Life Cycle Management (LCM) support.

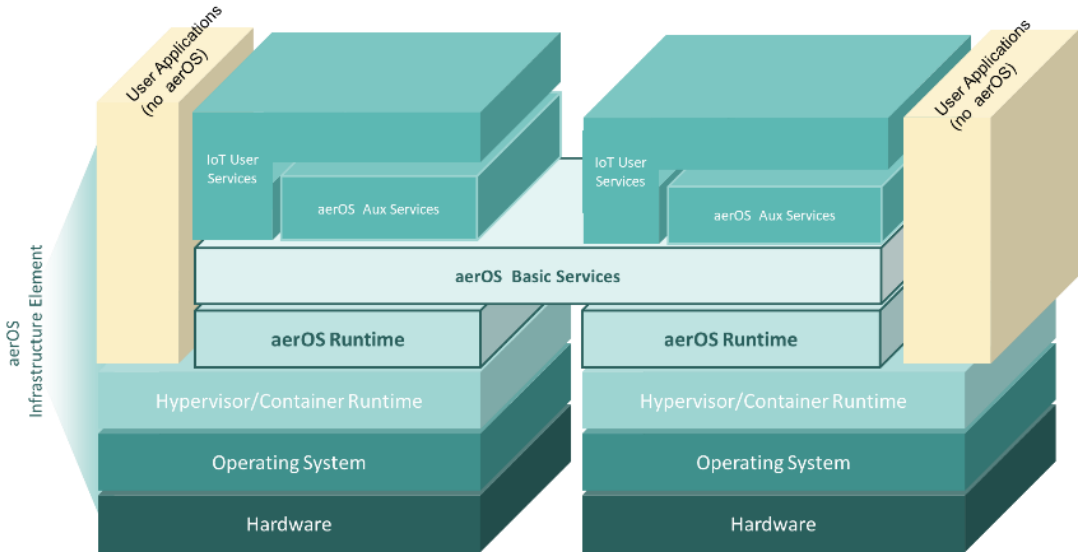


Figure 26. Deploying aerOS within a domain

Having rendered physical and virtual computing and network resources to aerOS IEs and domain, the next step in the deployment phase is to register and enrol newly created domain to the aerOS continuum. aerOS domain registration is part of the process presented in Figure 19. The right sequence of actions shows the sequence of stakeholder activities while registering their resources. Thus, aerOS Management Portal provides infrastructure providers the possibility to register their aerOS rendered resources to the continuum; thus, making them ready to take advantage of, and offer to, all aerOS functionalities. As an evolution to the whole process, for the future project iterations, it is designed to also offer the possibility to automatically deploy aerOS runtime and services on top of the provided resources directly through the portal. This will require an advanced set of deployment capabilities integrated in the aerOS Management Portal and of course the permissions and credentials to access stakeholder's cloud premisses or own managed resources. For this deployment capabilities should be exposed as per example VIMs API. In terms of connectivity, interconnection of aerOS IEs and domains is based on IP/TCP layer but employs a variety of backhauls as for example 5G, 4G, internet, WiFi, etc. VPN and tunnelling overlays are used when required.

The complementary sequence of actions depicted in Figure 19 represent the IoT services deployment process. IoT developers interested to seamlessly deploy their services are granted access to aerOS Dashboard and from a dedicated portal space, they submit their deployment requests based on a guided and template-framed process. In case their services integrate IoT devices deployed at their industry, home, etc, these devices should be located in the environment and aerOS basic services and IoT applications and services running on top of them should render them accessible to the continuum.

5.1.6. Business view

The business view describes the functionality of the aerOS system 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 27 presents the type of this relationship:

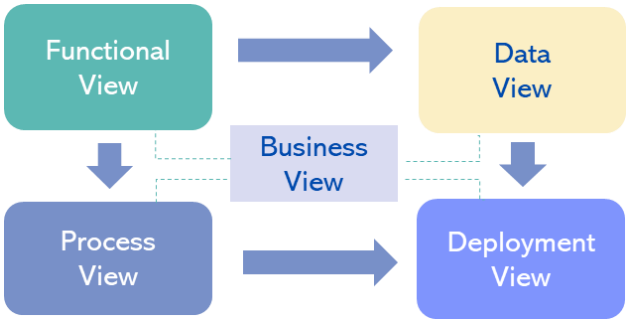


Figure 27. Business View interactions

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 Deliverable D2.1, 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. In the following sections, a short explanation on how aerOS and its corresponding core and/or auxiliary services will tackle them from a business perspective is presented:

Integration complexity

aerOS aims to provide a set of well-defined APIs to provide access to core architectural services covering security and authentication as well as data access and sharing. Integration complexity is reduced by delegating certain core functions to the different fabrics (network & compute, service and data fabrics as described in section 4.3) which often rely on the NGSI-LD API (refer to section 4.3.7) to facilitate a seamless interaction across services spanning the entire compute continuum.

Data collection and Analytics processes

aerOS provides a Data Fabric to model and store data. The use of semantic data models will facilitate the mapping of customer specific data formats to the aerOS data model and ease the process of data retrieval and analysis.

Privacy

aerOS architecture is built with data privacy by design. In fact, privacy is embedded in all the development stages by defining a novel methodology called DevSecPrivOps, which is defined in detail in deliverable D2.4 [11] . Nevertheless, privacy in aerOS will be considered as a shared responsibility between the aerOS system itself and the final users. Thus, while aerOS will take some actions such as securing physical access to the databases, users providing services on top of aerOS should take care of the logical access and disclosure of the stored data. To do so, aerOS introduces cybersecurity components (see section 4.3.4) which provides functionality for authentication, authorization, and IE trust management.

Securing network, devices, or data

Security in aerOS is delivered at several levels:

- The first level corresponds to the underlying operative system and running environment (e.g., Docker engine running over a Linux system), which provides physical access and isolation (e.g., OS level firewall and docker subnets).

- The second level corresponds to the aerOS cybersecurity services which introduce logical access to resources and data governance.

Additionally, aerOS provides the self-security feature (see deliverable D3.1) which enable detection of malfunctions and vulnerabilities at every IE. The assessment done by this self-feature is used by the orchestrations services to disallow untrusted IEs that may compromise the overall security of the aerOS domain.

Scalability

Scalability is one of the main design drivers in aerOS and has been properly discussed in section 3.2.2. The core service providing this feature is the Network & Compute Fabric (see section 4.3.1), 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) and the network & compute fabric (Section 4.3.1), 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 4.4.2). 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, all aerOS components will be GDPR compliant by default. Furthermore, they will also address specific European and national regulations regarding data privacy.

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] 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.

[6] 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.

[7] P. Trakadas, X. Masip-Bruin, F. Facca, S. Spantideas, A. Giannopoulos, N. Kapsalis, R. Martins, E. Bosani, J. Ramon and R. P. e. al., "A Reference Architecture for Cloud–Edge Meta-Operating Systems Enabling Cross-Domain, Data-Intensive, ML-Assisted Applications," *Sensors,* vol. 22, no. 22, 2022.

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

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

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

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

Appendix A. aerOS Terminology

Table 1. 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, 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 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 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 Context provider	Component exposing underlying IE status, or IoT data, in appropriate and compatible format (smart models), to be pulled/pushed and integrated to context brokers for further sharing across the continuum.
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 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 a *Decision 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 *Decision 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.