TRUSTWORTHY, COGNITIVE AND AI-DRIVEN COLLABORATIVE ASSOCIATIONS OF IOT DEVICES AND EDGE RESOURCES FOR DATA PROCESSING

Grant Agreement no. 101136024

Deliverable D2.3
Final EMPYREAN architecture, use cases analysis and KPIs

Programme:	HORIZON-CL4-2023-DATA-01-04
Project number:	101136024
Project acronym:	EMPYREAN
Start/End date:	01/02/2024 – 31/01/2027

Deliverable type:	Report
Related WP:	WP2
Responsible Editor:	RYAX
Due date:	31/01/2025
Actual submission date:	31/01/2025

Dissemination level:	Public
Revision:	FINAL

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101136024

Revision History

Date	Editor	Status	Version	Changes
17.11.24	RYAX	Draft	0.1	Initial ToC
20.12.24	RYAX	Draft	0.2	Integrate initial contributions in Sections 3, 4
09.01.25	RYAX	Draft	0.3	Integrate updated partners contributions in Sections 3, 4, 5
20.01.25	ICCS	Draft	0.4	Integrate final contributions in Sections 5, 6, 7, 8
23.01.25	RYAX	Draft	0.5	Complete version for internal review
29.01.25	RYAX	Draft	0.6	Revised version after internal review
31.01.25	RYAX	Final	1.0	

Author List

Organization	Author
RYAX	Yiannis Georgiou, Michael Mercier, Pedro Velho, Yuqiang Ma
ICCS	Aristotelis Kretsis, Panagiotis Kokkinos, Fotis Kouzinos, Emmanouel Varvarigos
CC	Márton Sipos, Marcell Fehér, Daniel E. Lucani
NVIDIA	Dimitris Syrivelis
UMU	Antonio Skarmeta, Eduardo Canovas
ZSCALE	Ivan Paez
NUBIS	Anastassios Nanos, Charalampos Mainas, Georgios Ntoutsos, Ilias Lagomatis, Konstantinos Papazafeiropoulos, Apostolos Giannousas
IDEKO	Aitor Fernández, Javier Martín
NEC	Jaime Fúster, Roberto González
EV ILVO	Jan Bauwens, Theodoros Chalazas
TRAC	Keshav Chintamani

Internal Reviewers

Eduardo Canovas, UMU

Ivan Paez, ZSCALE

Abstract: EMPYREAN introduces an innovative vision for an IoT-edge-cloud continuum, seamlessly integrating IoT devices, robots, and computational resources into dynamic, self-organizing collectives known as Associations. This deliverable presents the final outcomes of Task 2.2 “Concept, Use Cases and Requirements Analysis” and Task 2.3 “Architecture Specification”. It builds upon previous reports (D2.1 and D2.2), presenting key updates on EMPYREAN components, their interactions, and the finalized architecture. A significant contribution is the introduction of detailed system operation flows, providing insights into the internal workings of the platform and its innovative functionalities. Finally, it outlines the implementation and delivery plan, along with an analysis of requirements coverage, setting a clear path for the next phases of development and evaluation within the project.

Keywords: EMPYREAN Architecture, EMPYREAN Platform, EMPYREAN Components, System Operation Flows, Associations, Tracking KPIs, Edge-Cloud Continuum, Cognitive Orchestration, Trustworthy, AI-Driven Data Processing

Disclaimer: *The information, documentation and figures available in this deliverable are written by the EMPYREAN Consortium partners under EC co-financing (project HORIZON-CL4-2023-DATA-01-04-*101136024*) and do not necessarily reflect the view of the European Commission. The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The reader uses the information at his/her sole risk and liability.*

Table of Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
AMQP	Advanced Message Queuing Protocol
APC	Attribute-Based Credentials
API	Application Programming Interface
ATR	Autonomous Towing Robots
AWS	Amazon Web Services
CLI	Command Line Interface
CRI	Container Runtime Interface
CRUD	Create, Read, Update, Delete
CTA	Cyber Threat Alliance
CTI	Cyber Threat Intelligence
CV	Computer Vision
CVEs	Common Vulnerabilities and Exposures
D	Deliverable
DAG	Directed Acyclic Graph
DB	Database
DDS	Data Distribution Service
DID	Decentralized Identifier
DKMA	Distributed Key Management and Authentication
DLT	Distributed Ledger Technology
DoA	Description of Action
EAT	Entity Attestation Token
EC	European Commission
ETL	Extract, Transform, Load
EUs	End Users
FCS	Fleet Control System
FL	Federated Learning
FPGA	Field Programmable Gate Arrays
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HW	Hardware
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IIoT	Industrial Internet of Things
IoC	Indicators of Compromise
IoT	Internet of Things
JWT	JSON Web Tokens
K8s	Kubernetes
KMS	Key Management System
KPI	Key Performance Indicator
M	Month
ML	Machine Learning
MQTT	Message Queueing Telemetry Transport
MTTR	Mean Time to Repair
NBS	Nash Bargaining Solution

NIR	Near-Infrared Spectrum
OCI	Open Container Initiative
OF	Operation Flow
OOM	Out-of-Memory
OT	Operational Technology
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PMDS	Persistent Monitoring Data Storage
PoC	Proof of Concept
PPFL	Privacy-Preserving Federated Learning
PSM	Privacy and Security Manager
PVC	Persistent Volume Claim
QoS	Quality of Service
RAM	Random Access Memory
RDMA	Remote Direct Memory Access
REST	REpresentational State Transfer
RL	Reinforcement Learning
RLNC	Random Linear Network Coding
ROT	Resource Optimization Toolkit
SDK	Service Development Kit
SLA	Service Level Agreement
SOC	Soil Organic Carbon
SSI	Self-Sovereign Identity
SW	Software
TPU	Tensor Processing Unit
UAV	Unmanned Aerial Vehicles
UC	Use Case
UI	User Interface
URL	Uniform Resource Locator
VC	Verifiable Credentials
Vis-NIR	Visible and Near-Infrared Spectrum
VM	Virtual Machine
VP	Verifiable Presentations
VRAM	Video Random Access Memory
WAN	Wide Area Network
WP	Work Package
ZKP	Zero-Knowledge Proofs

1 Executive Summary

EMPYREAN introduces a groundbreaking vision for an IoT-edge-cloud continuum, seamlessly integrating IoT devices, robots, and computational resources into collaborative collectives termed "Associations." These dynamically formed Associations operate autonomously across diverse infrastructures, spanning multiple providers, geographical regions, and connectivity types, forming a unified and interconnected ecosystem. This Association-based continuum enables a harmonious blend of edge and cloud capabilities, fostering innovation in hyper-distributed, dynamic, and time-critical applications.

At the core of EMPYREAN's vision lies an AI-enabled control and management plane designed to autonomously balance computing tasks and data distribution. This distributed infrastructure empowers efficient and adaptive operations by optimizing resource utilization, performance, and resiliency within individual Associations and across federated ones. EMPYREAN's approach addresses the growing demands of modern applications, ensuring ubiquitous computing, storage, and communication capabilities across a highly dynamic IoT-edge-cloud ecosystem.

This deliverable continues the work reported in D2.1 (M6) and D2.2 (M7), introducing key updates on the various EMPYREAN components. In addition, it provides descriptions and details of some specific, noteworthy component interactions. A major contribution of this deliverable is the "System Operation Flows" section, which provides detailed presentations and diagrams of the platform's internals, bringing forward how the various EMPYREAN components interact to provide the innovative functionalities developed in the context of the EMPYREAN project.

Furthermore, this deliverable finalizes the EMPYREAN architecture and provides a structured approach for tracking Key Performance Indicators (KPIs), allowing consortium to prepare the terrain for further advancements in technical KPIs. Finally, it outlines the implementation and delivery plan, along with an analysis of requirements coverage, ensuring a clear roadmap for the project's next development and evaluation phases.

2 Introduction

2.1 Purpose of this document

This deliverable presents the final EMPYREAN architecture, detailing components' interactions, and system operation flows within the EMPYREAN platform. Additionally, it provides high-level information about the use cases, illustrating how they apply the proposed architecture and leverage the related functionalities.

In particular, the deliverable reports on the work done under tasks T2.2 "Use Case & Requirements Analysis" and T2.3 "Architecture Specification", which have been further enriched by the initial progress made in WP3 and WP4. The deliverables builds upon and advances the work previously presented in D2.1 (M6) and D2.2 (M7).

Section 3 offers updates on EMPYREAN components, providing detailed insights into noteworthy interactions between them. Section 4 presents high-level details on system operation flows, bringing new details on how different users of EMPYREAN interact with the platform. Both efforts upon the component interactions and system operation flows are then used as a base in Section 5, where the finalized architecture is presented. Furthermore, Section 5 presents the technical and use case KPIs, along with a methodology for tracking them during the project's lifecycle. Section 7 outlines the implementation and delivery plan, while the appendix includes a comprehensive analysis of requirements coverage.

2.2 Document Structure

The present deliverable is organized into six major chapters:

- EMPYREAN Components and Interactions
- System Operation flows
- Architecture Design and Tracking KPIs
- Use Case Analysis
- Implementation and Delivery Plan
- Requirements Coverage

2.3 Audience

This document is publicly available and is intended for anyone interested in gaining a high-level understanding of the EMPYREAN architecture and how it is applied within the project's use cases. In addition, this document can also help the general public grasp the system operation flows of the EMPYREAN project and how users can interact with the platform.

3 EMPYREAN Components

3.1 Overview

This section provides a detailed overview of the different EMPYREAN components. Initially, we present the latest updates of each component, related to their high-level functionality and their positioning in the final EMPYREAN architecture. Then, we describe some noteworthy interactions between components that play an important role in the way the different operation flows are defined in Section 4, as well as in the EMPYREAN architecture described in Section 5.

3.2 Components Updates

This subsection provides updates on the descriptions of EMPYREAN components, initially introduced in deliverable D2.2 (M7). These updates show the progress made in their development, highlighting changes in functionality, enhancements in design, and refinements in their roles within the platform. The complete list of all EMPYREAN components is available in Section 3 and Section 4 within deliverable D2.2.

3.2.1 Privacy and Security Manager

The Privacy and Security Manager (PSM) enforces privacy and security in decentralized ecosystems, particularly for IoT environments. It leverages Self-Sovereign Identity (SSI) systems and Decentralized Identifiers (DIDs) to offer secure, user-controlled identity solutions.

Key Features:

1. **Verifiable Credentials (VCs) & Verifiable Presentations (VPs):** PSM manages VCs and generates VPs using advanced cryptography, such as the p-ABC module, enabling Zero-Knowledge Proofs (ZKPs) and selective disclosure. This ensures that only necessary data is shared, protecting user privacy.
2. **JWT Signing with DIDs:** Enables secure and verifiable identity management by signing JSON Web Tokens (JWTs), facilitating fast and secure access to resources.
3. **Integration with Distributed Ledger Technologies (DLTs):** Verifies credentials with transparency and immutability while using smart contracts to automate DID retrieval and storage, enhancing security and reducing manual risks.

Due to emerging necessities and project requirements, the Privacy and Security Manager (PSM) will incorporate a mechanism for storing policies on the Distributed Ledger through smart contracts. This enhancement will enable dynamic policy installation across the EMPYREAN architecture (Figure 1), ensuring secure interactions and enhanced governance of resources, IoT devices, and associated access. By implementing this capability, the PSM will

facilitate the traceability of all interactions within the ecosystem, providing enhanced transparency, security, and accountability while supporting seamless policy enforcement across distributed components.

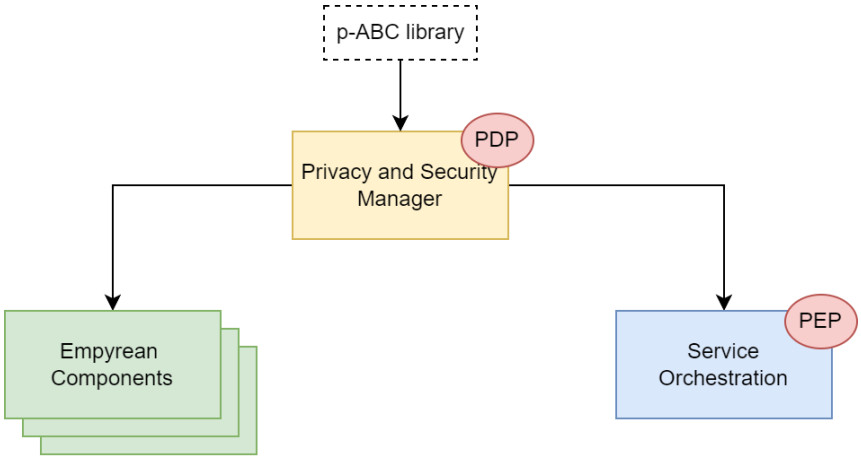

Figure 1: Privacy and Security Manager interaction with other EMPYREAN services

3.2.2 Edge Storage Gateway and Edge Storage

The main storage solution of the EMPYREAN platform is an enhanced extension of CC's SkyFlok S3 service. The service offers an S3-compatible interface, a de facto standard when it comes to (cloud-based) object storage solutions. This compatibility comes with the benefit of simple and quick integration with existing applications, courtesy of the large number of clients and SDKs spanning most programming languages built for Amazon's S3. The service supports all S3 CRUD operations related to buckets and objects, as well as advanced features such as multipart uploads and ranged downloads.

A special, unique feature of SkyFlok and SkyFlok S3 is its core storage architecture. Files are erasure-coded and distributed to different cloud locations. This approach offers several key benefits in terms of security, reliability, availability, and cost-effectiveness. In EMPYREAN, association-local storage locations will also be supported, enhancing the overall flexibility.

Moreover, users can customize exactly how and where their data is stored. They can define through a storage policy a list of storage locations, the redundancy level, the encryption scheme used, and so on. Each S3 bucket is then associated with a storage policy, making it possible to customize the storage to the specific requirements of each application. This approach enables users to benefit from cloud, edge, and hybrid storage solutions, offering adaptability and flexibility.

To support the temporary autonomous operation of a cluster when the Association becomes isolated from the outside world, basic S3 PUT and GET object functionality will be maintained for appropriately configured buckets.

The main components that provide the aforementioned functionality are depicted in Figure 2. Each Association will have an *On-premises Storage Gateway* that provides the S3-compatible API to platform applications. This component performs all data processing related to erasure coding, encryption, and compression as well as the uploading and downloading of file fragments to and from storage locations. It also plays an important part in providing autonomous operation, building on a local metadata database.

The metadata storage and management backbone is provided by the SkyFlok.com backend. It is deployed on Google Cloud Platform and is made up of a set of microservices. To facilitate the newly introduced features of the EMPYREAN project, CC is extending the SkyFlok S3 service with new microservices and new endpoints on existing microservices.

Two types of storage locations are supported: cloud backends (50+ covering the EU and US across major cloud providers) and edge storage. The *Edge Storage* component is an abstraction over association-local storage resources, provided through a containerized instance of MinIO. It is able to utilize any type of storage resource that can be mounted as a K8s volume.

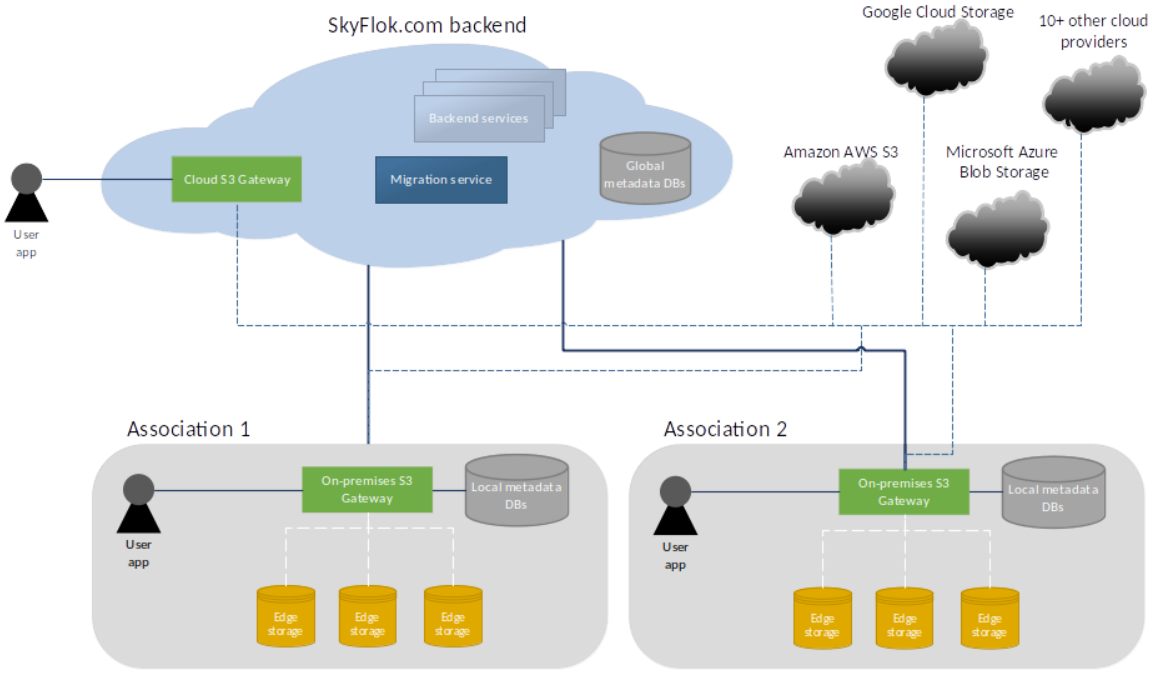

Figure 2: Overview of the main components of the EMPYREAN platform's storage solution.

3.2.3 Decentralized and Distributed Data Manager

EMPYREAN's decentralized and distributed data management framework is built on top of the Eclipse Zenoh[1] project. Zenoh is a Pub/Sub/Query protocol that provides a set of unified abstractions to seamlessly manage data in motion, data at rest, and computations across the Internet scale. Zenoh operates efficiently on server-grade hardware and networks as well as on microcontroller and constrained networks. This adaptability ensures that the framework meets the diverse demands of modern IoT and edge computing environments.

Additionally, Zenoh supports peer-to-peer, routed, and brokered communication models, enabling the selection of an optimal communication model at each stage of the system.

Eclipse Zenoh offers a two folds approach to facilitate seamless integration and operation in distributed systems. First, a networking protocol that is agnostic to the underlying technology, implementing a versatile networking layer capable of running above the Data Link, Network, or Transport layers of the OSI stack, as shown in Figure 3.

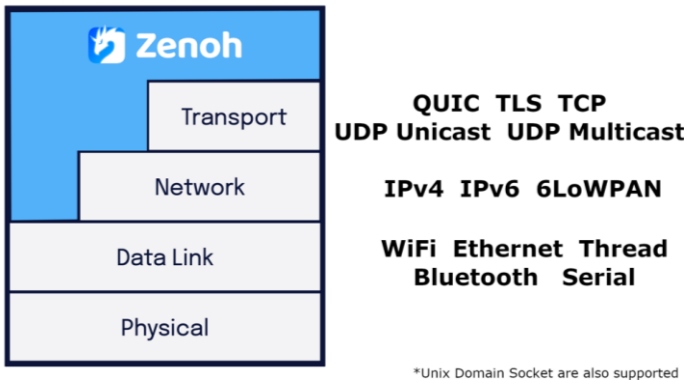

Figure 3: Eclipse Zenoh network stack

Moreover, Zenoh provides a set of APIs enabling EMPYREAN's application developers to build large-scale distributed systems. These APIs allow for the integration of Operational Technology (OT) protocols with the datacentre world and/or integrate IT protocols with the embedded world. Thus, it offers the capability to seamlessly bridge IT and OT protocols across different systems and networks.

Some of the main features of Eclipse Zenoh in EMPYREAN:

- ***Openness and Interoperability***: Zenoh enables diverse technologies to collaborate. For instance, the new Querier API supports efficient and optimized data retrieval. Enhanced support for ROS 2, a widely used framework in robotics, strengthens connections between Zenoh and other platforms, facilitating interoperability.
- ***Adaptability and Scalability:*** Zenoh's features are designed to adapt to varying technological requirements. The stabilization of Liveliness API support ensures real-time monitoring of active participants in the network. The inclusion of Zenoh-Pico[2],

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

which is made for small IoT devices, offers huge improvements in performance and extends its scope to devices like the Raspberry Pi Pico series.

- ***Security and Privacy***: Eclipse Zenoh protocol addresses critical issues such as fragmentation and message integrity, ensuring secure and reliable data transmission. These features safeguard interactions across the network, whether between IoT devices or cloud services.
- ***Performance Optimization:*** New advanced publisher/subscriber mechanisms complement the Querier API to improve data throughput and reliability. These optimizations enhance system performance while minimizing resource consumption, key to efficient IoT-to-cloud integration.
- **Technology Agnosticism**, EMPYREAN's components are committed to supporting diverse hardware and software platforms, including new features for QNX operating systems, which highlights its technology-agnostic approach. This openness reduces dependence on specific vendors and encourages widespread adoption.

3.2.4 Local Orchestration and Autoscaling Optimizations

This component is responsible for implementing advanced features to enhance local cluster orchestration in the edge-cloud continuum (Figure 4). The first one focuses on the development of AI/ML-driven vertical autoscaling mechanisms within Kubernetes-based clusters to enable autonomous and adaptive workload management in the continuum. While Kubernetes already supports horizontal autoscaling, vertical autoscaling offers the potential to optimize resource allocation by dynamically setting container limits based on telemetry data. This approach aims to (i) enhance container bin-packing efficiency, (ii) reduce the number of active nodes, and (iii) lower overall energy consumption. Additionally, the work explores extending vertical autoscaling techniques to GPU resources, addressing a growing need for hyper-distributed AI applications. This is particularly critical given the current lack of mature GPU fractioning technologies in Kubernetes.

Complementing this, the second advanced feature is related to optimizations upon Kubernetes to minimize cold start delays. To this end, this component introduces task place optimizations. By favouring the placement of tasks on nodes that have more container layers related to the task being deployed this approach further improves efficiency and responsiveness in the edge-cloud continuum.

The *AI-enabled Workload Autoscaling* that provides vertical pod autoscaling optimization will be developed as a microservice integrated with Kubernetes. To maximize effectiveness, it will be tightly integrated with the *Ryax Workflow Manager* in order to take advantage of the execution-related data stored within the Ryax's database. This data can be used to train the ML model used to inform autoscaling decisions. The *Container Layers Locality Scheduler* will provide the second feature that will be implemented directly within Kubernetes as a dedicated plugin.

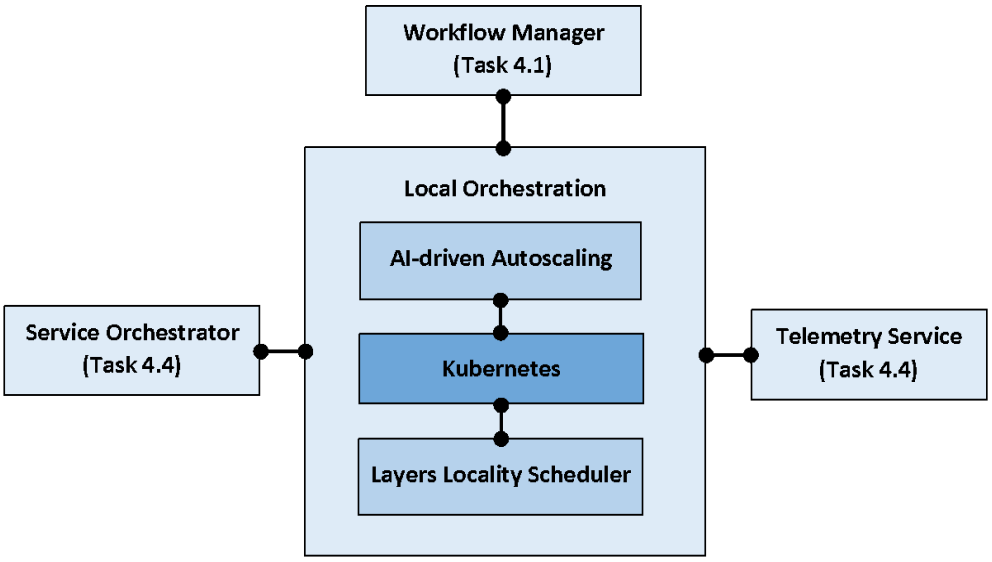

Figure 4: Local Orchestration and Autoscaling Optimization dependencies

3.2.5 Analytics Engine

Service assurance mechanisms are critical for enabling the self-driven adaptability of the IoT-edge-cloud continuum, ensuring optimal performance, reliability, and efficiency across this complex and dynamic infrastructure. To address this, EMPYREAN is developing a highly automated and intelligent IoT-edge-cloud continuum powered by AI-enabled distributed management through its Service Assurance service. This system will guarantee optimal application performance through autonomous adaptations operating within an infinite time horizon control loop.

EMPYREAN's approach integrates distributed service assurance mechanisms into each Association by utilizing real-time telemetry data and advanced algorithms within its *Analytics Engine*. The Analytics Engines employ continuous analysis techniques—such as machine learning, machine reasoning, swarm intelligence, and robust adaptive optimization—to drive orchestration mechanisms to (i) adapt resources within the Associations, (ii) provide dynamic load balancing of processing workloads, and data within and across Associations, (iii) migrate workloads to optimize energy efficiency, and (iv) mitigate resource fragmentation and connectivity issues.

Figure 5 illustrates the key building blocks of the EMPYREAN Analytics Engine and their interactions with other components of the EMPYREAN platform. The *Data Connector* processes collected monitoring data, applying various transformations to prepare them for the *Event Detection Engine*. The Data Manager provides local storage for processed data, trained models, and results, while also enabling the exchange of events and data with external components within the EMPYREAN platform. The *Event Detection Engine* implements the core functionality of EMPYREAN's distributed service assurance mechanisms. It facilitates the execution of developed data-driven algorithms designed to ensure that deployed applications and Associations consistently perform as intended.

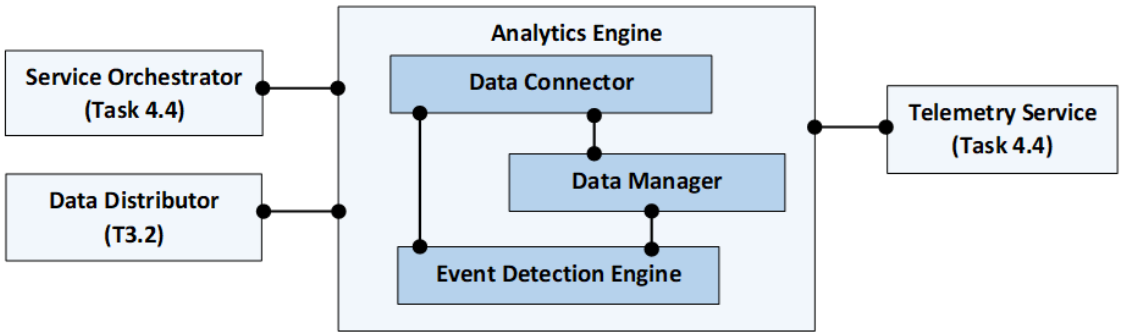

Figure 5: Analytics Engine core components and dependencies

3.2.6 Cyber Threat Intelligence Engine

Figure 6 illustrates the operational infrastructure of the Cyber Threat Intelligence (CTI) Engine. Data is collected from the CTA cloud and sent in JSON format to a MongoDB database, where it is stored and managed. Through the *Report Generation* module, this data, which includes detailed threat reports, indicators of compromise, and other essential metrics, is updated daily through automated scripts that connect to the CTA. The data will also be supplemented by information from UMU's MISP platform.

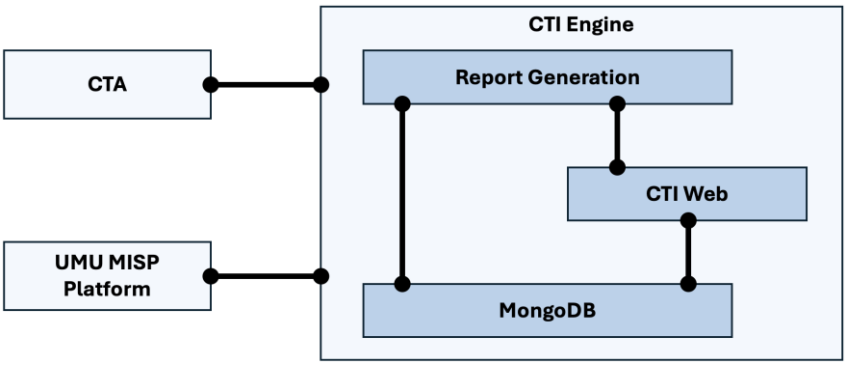

Figure 6: Architecture of the CTI Engine

The collected data undergoes rigorous analysis using advanced algorithms designed to detect patterns, trends, and anomalies. Before being stored in the MongoDB database, the data is thoroughly checked to ensure integrity and proper formatting. The database is optimized for fast queries and efficient data retrieval. It powers the *CTI Web* platform, a user-friendly interface that allows users to search, access detailed reports, and set up customized alerts. The platform also features data visualizations that facilitate the interpretation of threat trends and enable agile responses to security threats.

A recently introduced visualization feature enables users to identify trending elements within the database, providing insights into various data metrics. This feature categorizes and ranks trending and popular malware, malware families, attack patterns, and vulnerabilities, offering a comprehensive overview of the threat landscape. An example of this visualization is illustrated in Figure 7.

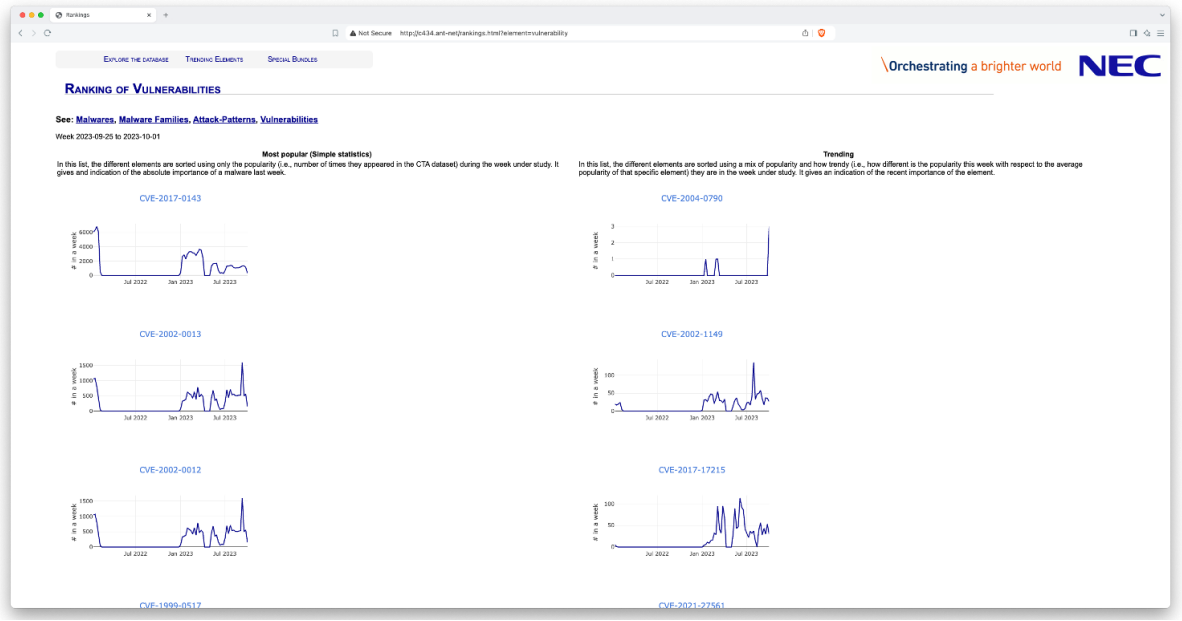

Figure 7: Example of the Trending Elements functionality

3.2.7 Decision Engine

The *Decision Engine* integrates distributed and multi-objective algorithms within the EMPYREAN platform, implementing the "decide" phase of the envisioned closed-loop control process based on the principles of observing, deciding, and acting. It will provide the EMPYREAN Aggregator and Service Orchestrator with the intelligence needed to (i) support the efficient operation of Associations, (ii) orchestrate hyper-distributed applications and allocate their workloads by considering local resource states and characteristics while meeting their objectives, and (iii) coordinate effective load-balancing of data and workload within and across Associations.

The Decision Engine will leverage the open-source Resource Optimization Toolkit (ROT) framework, initially developed by ICCS during the H2020 SERRANO[3] EU project. Within the EMPYREAN project, efforts will focus on enhancing the Decision Engine to enable distributed decision-making in a cloud-native environment. This will empower the Decision Engine to deliver robust, efficient decisions for dynamic resource allocation, workload balancing, and energy-efficient operations across the Association-based IoT-edge-cloud continuum.

Figure 8 depicts the key building blocks of the Decision Engine and its interactions with other EMPYREAN components. The *Decision Engine Controller* manages interactions with the available *Decision Engine Workers*, coordinating their operations. It also communicates with the Telemetry Service to retrieve necessary data and interfaces with other services within the platform, such as the Service Orchestrator.

[3] https://ict-serrano.eu

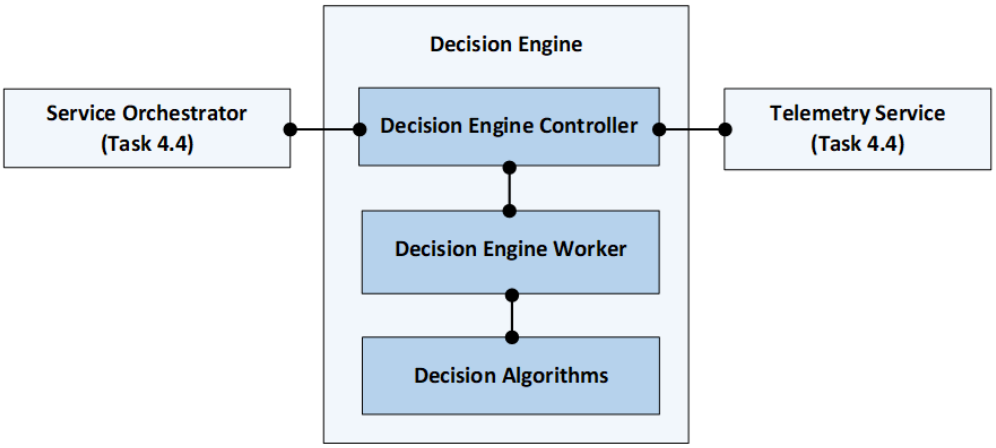

Figure 8: Decision Engine core components and dependencies

The *Decision Engine Worker* receives instructions from the Controller to start or stop the execution of algorithms, carries out the assigned tasks, and monitors their progress. It also reports relevant information back to the Controller. The architecture consists of a single Decision Engine Controller overseeing multiple Decision Engine Workers, which perform the computational tasks. Integrated within the workers are the Decision Algorithms that offer different trade-offs between optimality and complexity, ensuring efficient performance while meeting the diverse and strict applications requirements.

3.2.8 Workflow Manager

The *Workflow Manager* component (Figure 9) will be provided by the open-source Ryax workflow engine. Specific enhancements have been planned to be brought in EMPYREAN at the workflow management level. Initially, the goal is to enhance Ryax to efficiently support both long-running microservices and short-duration serverless functions for data analytics and AI applications in the edge-cloud continuum. Ryax leverages YAML-based abstractions which will be adopted in the project to facilitate the development of data analytics applications upon distributed systems.

Another important enhancement is to introduce multi-site workflow support, enabling workflows to execute across multiple clusters at both the edge and cloud. This capability requires particular networking and storage configurations to ensure seamless execution of workflows across distributed sites. Furthermore, the platform will support user-defined constraints and objectives to optimize workload placement. These features will be achieved through integrations with the Decision Engine and Service Orchestrator components. Additionally, Ryax will integrate with EMPYREAN Associations by interacting with the Aggregator and incorporating a native dataflow programming framework like Zenoh-Flow. This integration will provide fine-grained, real-time data communication capabilities, which is essential for IoT-based EMPYREAN use cases. To implement and support these capabilities, two new components will be introduced into the EMPYREAN architecture: the Ryax Runner, which will operate at the Association level, and the Ryax Worker that will function at the platform level.

The first feature related to the support of long-duration microservices will enable direct support of Docker containers and the definition of how they can be run individually or within a workflow. This will greatly benefit users who can bring their containers and run them directly without any changes or adaptations to the different underlying infrastructures. The second feature related to the support of multi-site workflow executions will utilize specialized networking abstractions to link a Kubernetes namespace with node pools spanning different sites. This innovation will enable the seamless execution of actions on different sites. The support of the dataflow programming framework will be enabled through the integration of the Zenoh-Flow open-source platform into Ryax. This will be done through an initial integration of Zenoh networking protocol on Ryax, enabling robust dataflow programming and efficient real-time communication, essential for IoT and edge applications.

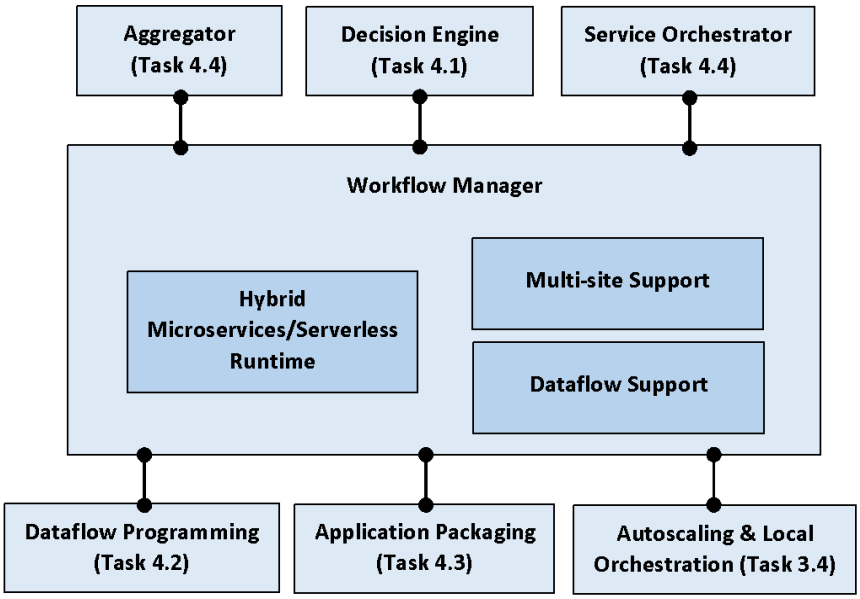

Figure 9: Workflow Manager components and dependencies

3.2.9 Dataflow Programming

EMPYREAN *Dataflow Programming* component will be built on top of the Eclipse Zenoh Flow open-source project to provide a framework for developing and deploying applications across the Cloud-to-Thing continuum. This framework aims to simplify and structure application creation while ensuring efficient deployment across distributed systems.

In this context, a dataflow application (Figure 10) is represented as a collection of nodes interconnected with links, forming a directed acyclic graph (DAG). These graphs are described using a human-readable descriptor file, referred to as a contract. The EMPYREAN Dataflow Programming component's primary objective is to enforce this contract. Starting from the descriptor file, Zenoh-Flow facilitates the instantiation of the application along with the placement of its components across the available infrastructures located at the thing, edge or cloud.

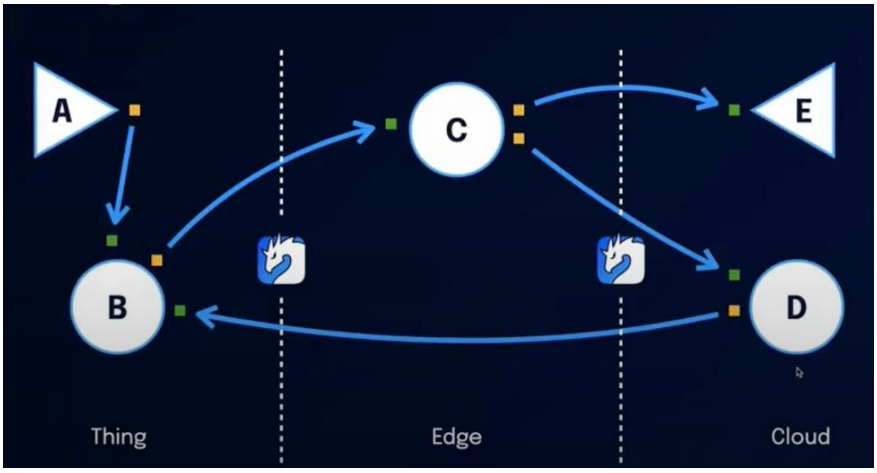

Figure 10: Example of dataflow application deployed across the continuum

Using EMPYREAN's Dataflow Programming component simplifies the role of the application developer to two primary tasks: (i) creating the different nodes that compose the application and (ii) describing the connections between them. Other steps are efficiently managed by the Zenoh-Flow framework and the other orchestration and deployment mechanisms within the EMPYREAN platform.

Key features of this component are:

- ***Declarative approach***: Through its descriptor file, application developers know precisely the structure of what will be deployed and how it will be connected. Its human-readable format lowers the entry barrier for non-technical application designers.
- ***Optimized communications:*** Being aware of the application's topology from the descriptor file, Zenoh-Flow will adapt communication channels for efficiency. Nodes located on the same Zenoh-Flow runtime instance will communicate through specific channels that add negligible overhead.
- ***Unified abstractions***: Regardless of where a node will be deployed, a developer only codes it once: Zenoh-Flow nodes implement a single interface. The result is a shared library that is later dynamically loaded by a Zenoh-Flow runtime.
- ***Location transparency***: As Zenoh-Flow uses Zenoh as its communication medium, application developers do not need to know ahead of deployment where their nodes will be running. Communications are based on key expressions that Zenoh routes transparently.
- ***Data isolation***: Zenoh-Flow associates each instance of an application with an identifier that is transparently leveraged in the communications. This ensures that if the same application is deployed twice on the same infrastructure or if several applications use the same "topics", no collision will occur. The same technique is used for each link, further allowing nodes to expose the same key expressions.

- ***High-performance***: Benchmarks show that Zenoh-Flow can achieve high throughput and low latency. A port of an Autonomous Driving System over to Zenoh-Flow further illustrated its capabilities, allowing for real-time control of a car in a simulated environment.

3.2.10 Action Packaging

Applications in the EMPYREAN platform are defined as a single workflow or combination of workflows. Each workflow is composed of one or more actions and each action is packaged as OCI-compatible images, making them ready for deployment using standard cloud-native tools (e.g., containerd, CRI-O, podman). These images can represent generic containers, lightweight application kernels (such as unikernels or libOS applications), or IoT firmware blobs. EMPYREAN offers a novel tool that can package any workload into an OCI-compatible image, enabling transparent storage, distribution, and deployment using standard cloud-native techniques. These capabilities are essential to the application design process, providing the system with the modularity and flexibility to support a wide range of application types.

By building on the NIX-based Environment Packaging tool, RYAX's workflow engine packaging, and NUBIS' unikernels builder, EMPYREAN introduces a unified packaging tool. This tool leverages the OCI specification to define crucial metadata for action binaries that can be used by the runtime environment to prepare and eventually implement the required execution environment.

This unified approach is a critical element for modular and flexible application design, empowering users to define microservices or serverless functions in any programming language while remaining agnostic to architectures (e.g., x86 or ARM). By leveraging the NIX functional package manager, it provides a declarative and reproducible process for building lightweight, OCI-compliant containers. This guarantees seamless polyglot workflows and supports diverse software and hardware infrastructures across the edge-cloud continuum.

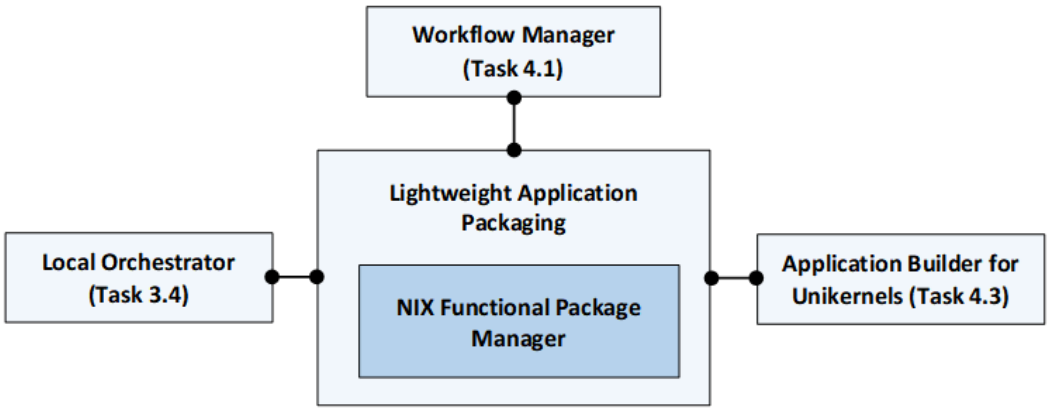

Figure 11: NIX-based Environment Packaging components and dependencies

3.2.11 Unikernels Builder

EMPYREAN aims to deliver an end-to-end software stack for application deployment based on unikernels. To achieve this, EMPYREAN introduces Bunny, a suite of system software components designed to enable the deployment of applications as unikernels in cloud-native environments.

Bunny is a libOS-based application building and packaging tool developed by NUBIS to streamline the creation and deployment of lightweight application images. By leveraging the powerful NIX package manager, Bunny simplifies the construction of slim, reproducible application images tailored for various unikernel frameworks, including Rumprun, Unikraft, OSv, MirageOS, and NanoVMs. These unikernel-based applications are optimized for performance and minimal resource usage, making them ideal for highly efficient and secure environments.

Bunny integrates seamlessly into the cloud-native ecosystem by packaging these lightweight application images into OCI-compliant images. This compatibility allows EMPYREAN to leverage standard container orchestration and deployment platforms, such as containerd and Kubernetes (K8s). When combined with the RYAX's workflow engine, Bunny provides a robust, efficient, and flexible solution for developers looking to harness the benefits of unikernels without compromising on compatibility or scalability. This integration ensures that developers can deploy unikernel-based applications with ease, leveraging the benefits of cloud-native technologies while achieving exceptional performance and resource efficiency.

Figure 12: Unikernels Builder - High-level overview of the Bunny workflow

3.2.12 Analytics-friendly Distributed Storage

Storing large volumes of IoT data reliably and cost-effectively is a significant challenge for current state-of-the-art systems. Given the need to efficiently query the data, replication has so far been seen as the only way to provide the required redundancy. Figure 14 illustrates the cost implications of using replication. To achieve the industry-standard requirement of tolerating the loss of two copies, 3x replication is typically used, causing a tripling of storage costs. In contrast, erasure coding provides a comparable level of reliability at exactly half the storage cost in this case. The cost difference becomes even greater when higher levels of reliability are required.

Unfortunately, time-series IoT data is only useful if it can be queried efficiently. In the case of cloud storage, providers charge a premium price for data egress. Thus, it is imperative to minimize the amount of data transferred when evaluating a query. Given that erasure coding mixes the original data, byte-level access has so far been impractical. The same problem arises with compression, most schemes make it impossible to know where each byte of the original data is stored and thus typically all data must be downloaded and decompressed. These are the two core challenges we address in EMPYREAN, aiming to go beyond the state of the art with this component.

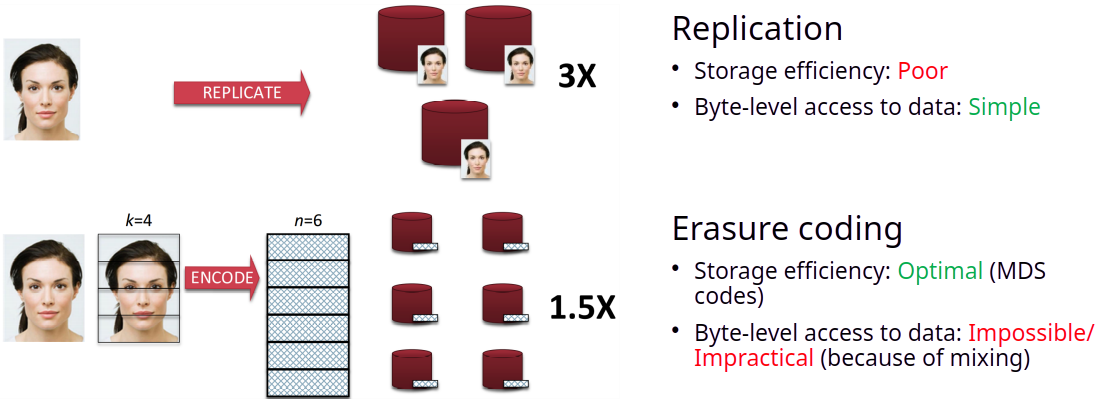

Figure 13: Comparing replication with erasure coding

The Analytics-Friendly Distributed Storage System will be implemented through the *IoT Query Engine* and will incorporate the following key features:

- Use of erasure coding and data distribution to multiple storage locations.
- Efficient erasure-coded analytics queries that transfer comparable amounts of data to replication-based queries.
- Use of a specialized form of deduplication to compress data both at rest and in-flight.

To achieve these objectives, the system will build upon the *Edge Storage Gateway* as well as a set of cloud-based components, deployed using cloud lambda functions to the three main cloud provider's infrastructures.

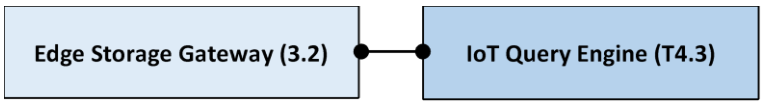

Figure 14: Analytics-friendly Distributed Storage

3.2.13 Service Orchestrator and EMPYREAN Controller

The orchestration process within EMPYREAN consists of two primary stages and involves two key components of the platform: the *Service Orchestrator* and the *EMPYREAN Controller* (Figure 15). At the Association level, multiple Service Orchestrators operate as high-level orchestrators, while various Local Orchestrators manage the individual edge and cloud platforms integrated within the EMPYREAN framework. This distributed orchestration model enables efficient and intelligent resource and task management across the entire IoT-edge-cloud continuum, ensuring seamless coordination and optimization of workloads throughout the platform.

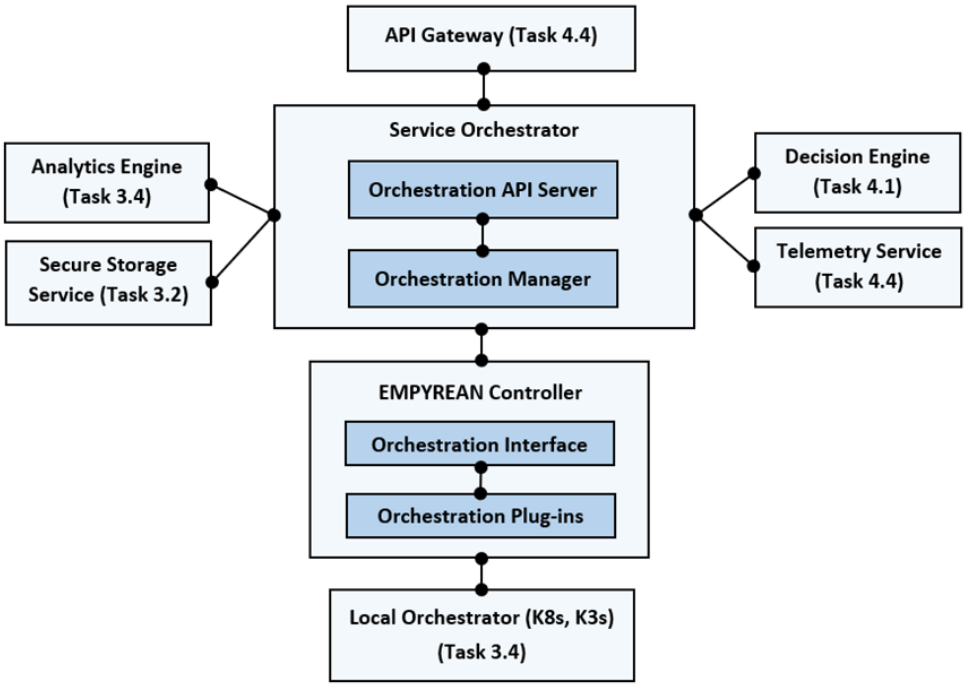

Figure 15: Service Orchestrator and EMPYREAN Controller components and dependencies

In the first stage, multiple Service Orchestrators act as cognitive agents, leveraging their local knowledge to compete for the efficient and rapid mapping of application workloads. In the second stage, each Service Orchestrator intelligently assigns its respective part of the overall workflow to the Associations it manages. EMPYREAN adopts a hierarchical orchestration approach to facilitate this process. High-level decisions are made by the Service Orchestrator at the Association layer, while low-level scheduling (i.e., the actual assignment of workloads to specific infrastructure resources) is handled by the native orchestration mechanisms (i.e., Local Orchestrator) of each platform (e.g., K8s, K3s).

These Local Orchestrators ensure compliance with platform specifications and provide fine-grained workload management. The Service Orchestrator consists of two primary services (i) the *Orchestration API Server* that serves as the single entry point for other components to access EMPYREAN's service orchestration functionalities and (ii) the *Orchestration Manager* that implements the application logic, oversees the operation of internal components, and coordinates resource allocation and application deployment.

The *EMPYREAN Controller* abstracts interactions with the specific edge and cloud orchestration mechanisms at each EMPYREAN location. It processes requests from the Service Orchestrator to deploy or adjust already deployed applications. The Controller comprises two main the *Orchestration Interface* and *Orchestration Plug-ins*. The former provides an infrastructure-agnostic interface for describing deployment descriptions and constraints to diverse local orchestration mechanisms, while the latter translates generic instructions from the Orchestration Interface into specific actions and procedures tailored to the selected local orchestration mechanisms.

3.2.14 Telemetry Service

The EMPYREAN Telemetry Service addresses the challenges of monitoring federated IoT-edge-cloud platforms by providing robust observability and telemetry capabilities within Associations. Traditional localized monitoring methods are insufficient for ensuring optimal performance, security, and efficiency across interconnected services. Observability enables gaining external insights into system behavior and performance, facilitating troubleshooting, and answering questions like, “Why is this happening?” Telemetry focuses on the real-time collection, measurement, and transmission of performance data, including CPU usage, memory consumption, storage capacity, and network traffic. Together, they form the foundation for effective monitoring, automation, and decision-making in the EMPYREAN platform.

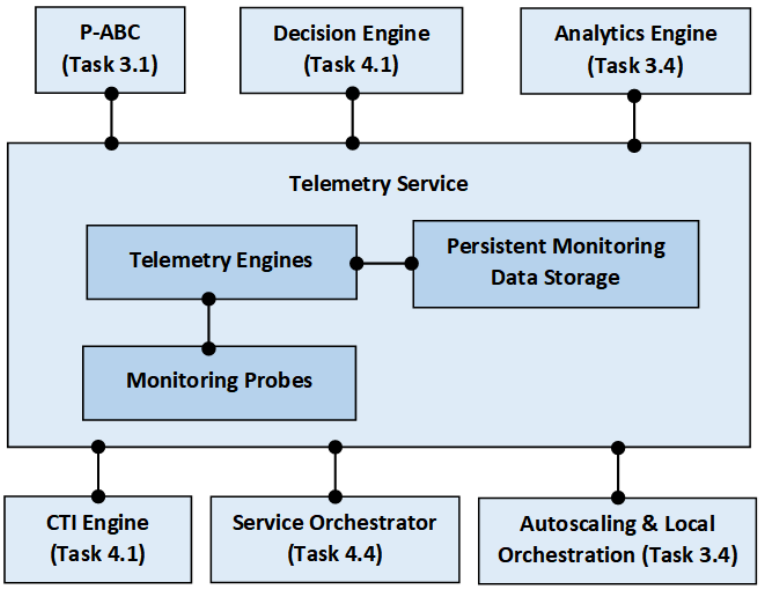

Figure 16: EMPYREAN Telemetry Service components and dependencies

The telemetry service is built on a distributed infrastructure composed of key components:

- *Telemetry Engines*: Independently manage, collect, and analyze telemetry data from various segments of the infrastructure, ensuring a unified view of system health.
- *Monitoring Probes*: Specialized components that collect real-time performance data from specific resource types.
- *Persistent Monitoring Data Storage (PMDS)*: A centralized repository for long-term storage of telemetry data, enabling historical analysis, trend identification, and resource optimization.

By integrating these components (Figure 16), the EMPYREAN telemetry service ensures end-to-end visibility, enhanced security, scalability, and dynamic resource utilization across the IoT-edge-cloud continuum, facilitating data-driven decision-making and advanced analytics.

3.2.15 EMPYREAN Aggregator

An EMPYREAN Aggregator (Figure 17) manages and coordinates the operation of an EMPYREAN Association. Each Aggregator is a logical component that integrates multiple core components and services to provide the intelligence and orchestration logic needed to operate an Association. Its key responsibilities include facilitating application deployment, ensuring secure and trusted workload execution, and overseeing data storage across the continuum. An Aggregator orchestrates its Associations, encompassing distinct or shared computational and storage resources.

Multiple self-managed and cooperating Aggregators form the management fabric of the EMPYREAN platform. Together, they transform the IoT-edge-cloud continuum into an autonomous, collaborative, composable, and self-organized ecosystem. Operating in a distributed and autonomous manner, Aggregators utilize an internal two-level hierarchical system to effectively manage resources within an Association.

The *API Gateway* facilitates seamless communication between Aggregators, general edge, or (multi-) cloud providers. Key components of the EMPYREAN Aggregator include:

- *Service Orchestrator:* Enables efficient resource and workload orchestration.
- *Decision Engine:* Provides intelligent decision-making for optimized operations.
- *Edge Storage Gateway:* Provides distributed, hybrid, and encrypted data storage.
- *Data Distributor*: Ensures decentralized interconnection and seamless data distribution.
- *Security and Privacy Manager:* Delivers distributed trust and identity management.
- *Telemetry Engine and Analytics Engine: S*upport the monitoring of heterogeneous resources and deployed applications while providing service assurance mechanisms.

In its updated version, the EMPYREAN Aggregator also integrates the *Ryax Runner*, which acts as the execution engine for user workflows within a specific Association. This component bridges the Workflow Manager with the computing resources of individual edge and cloud platforms, enabling the seamless execution of actions and workflows. The Ryax Runner operates in collaboration with the EMPYREAN Controller to deploy, execute, and manage the hyper-distributed application workflows across the EMPYREAN Associations, ensuring seamless deployment and efficient, coordinated operation within the IoT-edge-cloud continuum.

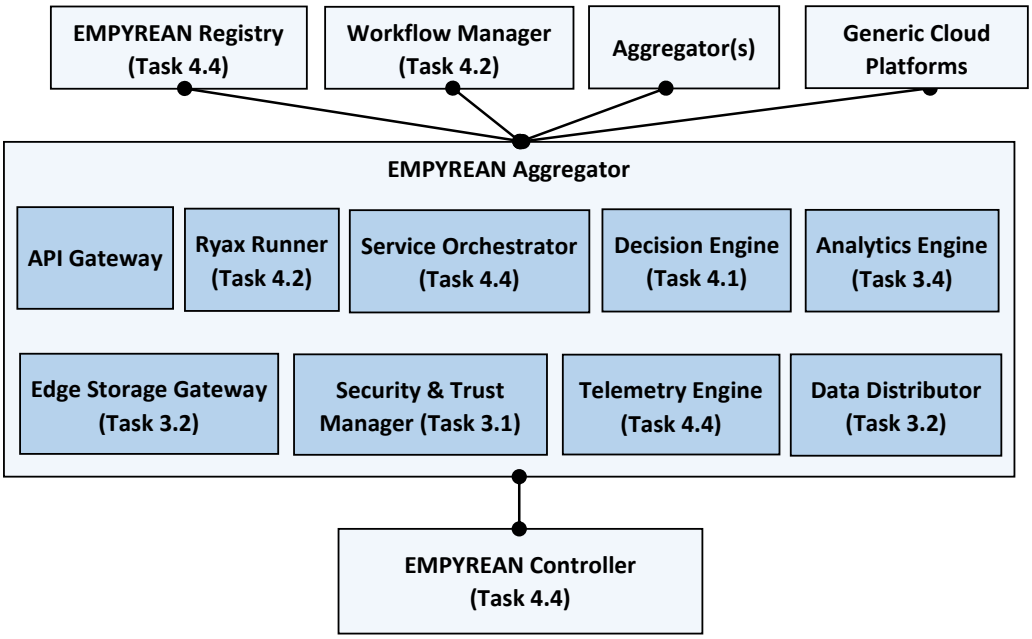

Figure 17: EMPYREAN Aggregator core components and dependencies

3.2.16 EMPYREAN Registry

The EMPYREAN Registry facilitates the registration and management of IoT devices, edge, and cloud resources within Associations. It serves as a unified entry point for both core platform services and third-party entities, enabling the discovery, cataloguing, and advertising of Associations and services across the Association-based continuum. The EMPYREAN registry keeps track of the available Associations and services, the mapping of infrastructure resources to Associations, and the relationships between users and Associations.

Figure 18 shows the updated version of the EMPYREAN Registry. The *API Gateway* facilitates seamless interaction and the exchange of events between the EMPYREAN services and core components of the Registry. The *Registry Manager* oversees the operation of the Registry and manages its interactions with the other services within the EMPYREAN platform. The *Service Catalogue* maintains critical information about software packages, container images, service descriptors, and other metadata essential for service deployment and management. It also stores and manages descriptors of hyper-distributed applications and services available for

deployment on the EMPYREAN infrastructure. The *Container Image Repository* stores OCI-compatible images of hyper-distributed applications, built and packaged using EMPYREAN's dedicated mechanisms.

The *Association Metadata Store* contains high-level metadata about the available Associations, including details on participating resources, their ownership, and sharing policies. This information supports orchestration and load-balancing decisions made by the platform's distributed decision-making mechanisms. The *Data Connectors* collect metadata and information from diverse systems, including data stores, external catalogues, data pipelines, and other relevant data sources. Moreover, the integration of Security and Trust Manager services establishes a trust anchor to support trust, identity, and credential management operations across the distributed Associations. This ensures secure interactions, enhances reliability and promotes seamless collaboration within the EMPYREAN ecosystem.

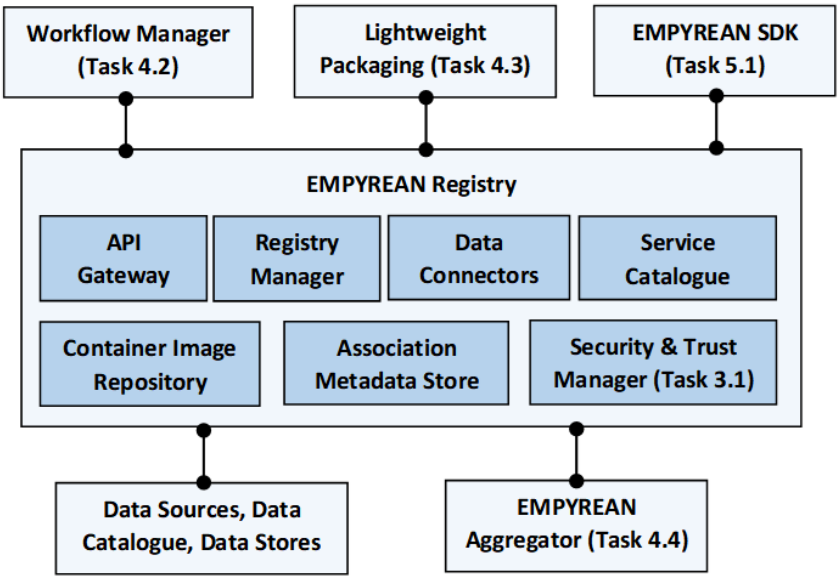

Figure 18: EMPYREAN Registry core components and dependencies

3.3 Components Interactions

3.3.1 Decision Engine with Telemetry Service

3.3.1.1 Functionality Unlocking

The Decision Engine integrates and executes several decision-making algorithms. To extend its capabilities, the Decision Engine will be enhanced to enable multi-agent speculative intelligent scheduling across EMPYREAN Associations. This enhancement also includes the ability to operate in a distributed manner, with multiple independent instances deployed across different Associations. Each Decision Engine instance will be designed to support multi-agent algorithmic solutions that accommodate various levels of cooperation, from full

collaboration to partial or competitive operation. This flexibility enables dynamic and intelligent decision-making tailored to the unique needs and constraints of the platform.

A key enabler of these capabilities is the interaction with the Telemetry Service, which provides the Decision Engine with the necessary data for real-time decision-making and adaptive resource management. The Telemetry Service will also be tightly coupled with observability mechanisms by integrating monitoring, metrics, distributed tracing, and logging. This integration will ensure that the Decision Engine receives high-fidelity data streams and diagnostic feedback for continuous improvement. The Telemetry Service will deliver a detailed, granular view of the continuum, allowing the Decision Engine to execute distributed algorithms that dynamically balance workloads and manage resources to optimize performance, energy efficiency, and fault tolerance across the Associations.

3.3.1.2 High-Level Design

The updated design of the Decision Engine will incorporate a lightweight and scalable communication layer (e.g., a message broker like Zenoh or gRPC). This middleware will serve as the backbone for inter-agent communication, facilitating the exchange of state information, coordination signals, and messages between Decision Engine instances. The real-time synchronization will ensure that instances can collaborate or compete effectively, even in highly dynamic environments.

There will also be a shared repository to store global states, shared objectives, and metadata that Decision Engine instances can access as needed, providing a centralized reference point for coordination across multiple Associations. Moreover, the Decision Engine Controller will be extended to support the dynamic instantiation of new Decision Engine instances. Instances can be deployed or decommissioned based on workload demand or new Association formations, enabling adaptive instance deployment. The Decision Engine will integrate closely with the Telemetry Service to receive real-time updates on resource states, workload metrics, and system performance. Finally, the Decision Engine instances will integrate a range of multi-agent algorithms, enabling instances to collaborate or compete depending on the scenario to achieve Association-specific goals.

By incorporating these features, the Decision Engine can evolve into a multi-agent system capable of cooperative and competitive operation, significantly enhancing its flexibility and performance in managing the hyper-distributed environments of the EMPYREAN platform.

3.3.1.3 Interfaces

Next, we provide a high-level description of the interfaces required for the communication between the involved components that will be developed and integrated within the technical work packages (WP 3-5). The interfaces are defined jointly between the interface implementer and interface user while being implemented in the context of the respective technical tasks.

Table 1: Decision Engine Interface

Interface ID	WP4T1DE-I
Description	The interface will enable orchestration entities to retrieve optimized deployment plans for workloads across Associations, support methods to manage decision-making processes and offer access to detailed log messages. It will also facilitate the operation between Decision Engine instances.
Component providing the interface	Decision Engine
Consumer components	Service Orchestrator, Decision Engine
Type of interface	REST and AMQP
State	Synchronous and Asynchronous
Responsibilities	ICCS

Table 2: Telemetry Service Interface

Interface ID	WP4T4TS-I
Description	This interface will provide real-time telemetry data, including resource utilization metrics, workload statuses, and resource characteristics, ensuring informed decision-making. It will support various query and filtering operations along with notifications.
Component providing the interface	Telemetry Service, Telemetry Engine
Consumer components	Decision Engine, Analytics Engine, Service Orchestrator, EMPYREAN Aggregator
Type of interface	REST and AMQP
State	Synchronous and Asynchronous
Responsibilities	ICCS, UMU

Table 3: PMDS Interface

Interface ID	WP4T4PMDS-I
Description	It will enable a storage service for the historical monitoring data using a time-series data store. It will expose methods to interact with the data store, supporting insert and various query and filtering operations.
Component providing the interface	Persistent Monitoring Data Storage
Consumer components	Decision Engine, Analytics Engine,
Type of interface	REST
State	Synchronous
Responsibilities	ICCS, UMU

3.3.2 Workflow Manager with Service Orchestrator

3.3.2.1 Functionality Unlocking

The Workflow Manager in EMPYREAN will be enhanced to support multi-site workflow executions, addressing the inherent complexities of the Association-based IoT-edge-cloud continuum. By default, the Workflow Manager utilizes first-fit policies for resource allocation and is limited to operating within a single Association. However, the integration of the Service Orchestrator will overcome these limitations by, enabling the adoption of intelligent best-fit policies capable of efficiently utilize resources across multiple Associations. Moreover, the Service Orchestrator will be extended and enhanced to facilitate decentralized and cooperative operations. This will empower the EMPYREAN platform to orchestrate edge-cloud resources efficiently, enabling dynamic application deployment across multiple, autonomous Associations.

These enhancements will ensure that workflows can be seamlessly and dynamically distributed across multiple edge-cloud sites and Associations, enhancing the platform's scalability, flexibility, and overall performance and resource efficiency. By collaborating with the Service Orchestrator, the Workflow Manager will be able to intelligently evaluate resource availability, operational constraints, and workload requirements to make optimized decisions on resource allocation.

3.3.2.2 High-Level Design

The integration design takes into account the architectures of the Workflow Manager and Service Orchestrator to enable communication between their internal components. The primary objective is to facilitate the exchange of necessary information, allowing the Service Orchestrator to execute the first stage of the EMPYREAN orchestration process, which involves intelligently distributing application workloads across the Associations.

By default, the Ryax Workflow Manager includes a single user interface, the Ryax Studio, which interacts with one Ryax Runner service to schedule and manage computing resources. To support multiple Associations, the user interface will be decoupled from a single Runner. Instead, the updated design will include one Runner service per Association, enabling each Runner to communicate independently with the user interface and its respective Service Orchestrator. This decoupling ensures that multiple Runners can operate concurrently, with each managing a distinct Association without interference.

When a deployment request is initiated from the user interface, it will first be routed to the EMPYREAN Aggregator of the user's default Association. Then, the Aggregator, in collaboration with the respective Service Orchestrator, will execute the multi-agent decision-making process to assign workloads across Associations. Once the high-level workload distribution is determined, these services will forward the deployment tasks to the appropriate Ryax Runner within the selected Associations.

To further enhance multi-site support, the Workflow Manager will introduce the concept of Ryax Worker services. Each Worker will be responsible for managing workflows on specific underlying resources, working in coordination with the EMPYREAN Controller. This approach assigns Workers per K8s/K3s cluster, ensuring efficient resource management and workflow execution across the infrastructure.

3.3.2.3 Interfaces

The following provides a high-level description of the exposed interfaces by the Workflow Manager and Service Orchestrator.

Table 4: Workflow Manager Interface

Interface ID	WP4T2WF-I
Description	The Workflow Manager interface will communicate with the Service Orchestrator through a set of well-defined APIs to facilitate seamless information exchange for workload distribution across multiple Associations. It will support bidirectional communication, allowing the Workflow Manager to send workload execution requests, resource availability queries, and operational constraints to the Service Orchestrator while receiving optimized resource allocation decisions and deployment instructions. The interface will also support asynchronous messaging and event-driven triggers to ensure dynamic adaptation to changing resource conditions.
Component providing the interface	Workflow Manager
Consumer components	EMPYREAN Aggregator, Service Orchestrator
Type of interface	combinations of REST, gRPC and RabbitMQ
State	Synchronous and Asynchronous
Responsibilities	RYAX, ICCS

Table 5: Service Orchestrator Interface

Interface ID	WP4T4SO-I
Description	This interface enables the deployment and management of cloud-native applications and abstracts the interaction of the Service Orchestrator with the EMPYREAN Controllers. Moreover, it allows the service assurance mechanisms to trigger dynamic re-configurations of the already deployed applications.
Component providing the interface	Service Orchestrator
Consumer components	Workflow Manager, EMPYREAN Aggregator, Analytics Engine, EMPYREAN Controller
Type of interface	REST and gRPC
State	Synchronous and Asynchronous
Responsibilities	ICCS, RYAX

3.3.3 Workflow Manager and Dataflow Programming

3.3.3.1 Functionality Unlocking

The integration of dataflow programming through Zenoh-Flow within Ryax introduces advanced functionalities that transform how workflows handle data communication and real-time processing. Users will be able to define precise dataflows between various actions within a workflow, ensuring seamless and efficient exchanges of information. This capability allows users to go beyond traditional input-output exchanges, providing fine-grained control over movement of data throughout complex workflows. Such precision is particularly valuable in IoT-based use cases, where the timing and accuracy of data exchanges across multi-infrastructure environments are critical to ensuring reliable and actionable outcomes.

Additionally, the integration enhances the platform's ability to handle real-time communication across diverse infrastructures, addressing the challenges posed by distributed systems in edge-cloud continuums. By leveraging Zenoh-Flow's inherent strengths in low-latency and high-throughput dataflow management capabilities, Ryax equips developers with tools to build workflows that are optimized for dynamic, time-sensitive applications. These advanced functionalities empower users to develop applications that are not only efficient but also inherently adaptable to the needs of modern IoT and AI-driven ecosystems, supporting scenarios where data must be processed and acted upon with minimal delays.

3.3.3.2 High-Level Design

The integration between Ryax and Zenoh-Flow focuses on seamlessly embedding a dataflow programming framework into workflow-based application development. At its core, this integration establishes a connection between Ryax workflows and Zenoh-Flow dataflows, allowing users to define the specific paths through which data moves between different actions in a workflow. Hence, the design allows the actions of the workflows to be further analyzed by how the data flows within actions. Ryax’s workflow abstraction is enhanced to include dataflow-specific configurations, enabling users to specify data dependencies, communication protocols, and the expected real-time behavior of their applications. This enhanced design ensures that workflows remain intuitive while gaining the flexibility and power of fine-grained dataflow control.

The integration leverages Zenoh-Flow’s capabilities to address the challenges of real-time communication across diverse and distributed infrastructures. By integrating Zenoh-Flow's low-latency communication primitives and its ability to support heterogeneous environments, Ryax workflows can efficiently manage data exchanges between edge, cloud, and hybrid setups in real-time. The design includes the deployment of Zenoh-Flow nodes as part of the Ryax runtime, ensuring that dataflow orchestration occurs transparently and efficiently. This approach allows the platform to handle time-sensitive data exchanges required by IoT and AI applications, without users requiring to configure complex networking setups manually.

To ensure scalability and robustness, the integration employs a modular architecture where dataflow components can be dynamically instantiated using high-level abstractions. This modularity allows developers to design workflows that are adaptable to varying workloads and infrastructures, while Zenoh-Flow's underlying engine ensures consistent and reliable data communication.

3.3.3.3 Interfaces

The following tables provide a high-level description of the exposed interfaces by the Workflow Manager and the Dataflow programming component.

Interface ID	WP4T2WFR-I
Description	This communication interface facilitates low-latency data transfer between Zenoh-Flow nodes and Ryax actions. It is a messaging protocol that leverages Zenoh's native real-time capabilities, ensuring efficient data transmission across distributed infrastructures. It serves as the backbone for real-time, low-latency data communication between Ryax and Zenoh-Flow nodes.
Component providing the interface	Zenoh-Flow Dataflow programming residing upon Zenoh communication protocol
Consumer components	Ryax Workflow Manager
Type of interface	Zenoh's native protocol
State	Synchronous
Responsibilities	ZSCALE, RYAX

Interface ID	WP4T2CM-I
Description	This Configuration and Monitoring Interface provides tools for configuring dataflow parameters, visualizing dataflows, and monitoring their performance within workflows, enabling users to fine-tune and debug integrated applications effectively. These interfaces collectively establish a cohesive integration, with Zenoh's communication layer ensuring robust real-time interactions.
Component providing the interface	Zenoh-Flow Dataflow programming residing upon Zenoh communication protocol
Consumer components	Ryax Workflow Manager
Type of interface	REST or GraphQL API
State	Synchronous and Asynchronous
Responsibilities	ZSCALE, RYAX

3.3.4 Application Packaging and Unikernels Builder

3.3.4.1 Functionality Unlocking

The integration of the Action Packaging with the Unikernels Builder components unlocks functionalities that streamline application development, deployment, and execution. By combining the modularity and flexibility of OCI-compatible action packaging with the efficiency and performance of unikernels, EMPYREAN allows developers to create powerful, secure, lightweight, reproducible, and architecture-agnostic application components. The unified packaging tool ensures that any application component, regardless of its programming language or runtime requirements, can be seamlessly packaged into OCI-compliant images. This approach guarantees compatibility with diverse deployment environments spanning the edge-cloud continuum.

In addition, the platform ensures that unikernels can be effortlessly encapsulated within OCI-compliant images, allowing their seamless deployment via Kubernetes. This integration establishes a robust development pipeline that balances the performance and security benefits of unikernels with the scalability and standardization offered by cloud-native technologies. The result is a system that empowers developers to build and deploy highly efficient, secure, and portable applications.

3.3.4.2 High-Level Design

The integration of Action Packaging with the Unikernels Builder in the EMPYREAN platform will leverage the NIX-based Environment Packaging tool embedded within the Ryax Workflow Manager. This packaging tool will define and prepare action binaries in a declarative and reproducible manner while ensuring compatibility with the OCI specification. By incorporating Bunny, the Unikernels Builder, Ryax can generate lightweight, OCI-compliant images optimized for various unikernel frameworks like Unikraft and MirageOS. This integration ensures that actions within workflows are seamlessly packaged and reproducibly managed using the NIX functional package manager, enabling compatibility across diverse infrastructures.

To improve interoperability, Ryax will extend its capabilities to support HTTP protocol along with gRPC for the action wrapper. The action wrapper is a lightweight intermediary between the Ryax Workflow Engine and user code that creates a (gRPC) server with a simple interface for initializing action and running executions. The support of HTTP along with gRPC significantly facilitates the integration of Unikernels Builder with Ryax.

Moreover, Ryax will be upgraded to offer users the ability to specify the desired container runtime for executing actions, including support for the *urunc* runtime, which is designed for deploying unikernel-based workloads. This enhancement ensures that OCI-compliant images generated by the Unikernels Builder can be executed in optimal runtime environments, maximizing their performance and efficiency.

3.3.4.3 Interfaces

Interface ID	WP4T3PKG-I
Description	This interface ensures seamless communication for packaging actions into OCI-compliant unikernel images. It allows Ryax to submit build requests, including the action's NIX-based metadata, runtime specifications, and target unikernel framework.
Component providing the interface	Unikernels Builder
Consumer components	Ryax Workflow Manager
Type of interface	REST
State	Asynchronous
Responsibilities	NUBIS, RYAX

3.3.5 Workflow Manager and Edge Storage

3.3.5.1 Functionality Unlocking

Integrating the Edge Storage Gateway with the Ryax Workflow Manager will unlock advanced functionalities that significantly enhance the flexibility and efficiency of workflow execution in distributed, hybrid environments. One key functionality is the ability to dynamically allocate storage resources tailored to the specific requirements of each workflow. Each S3 bucket in Edge Storage is governed by a storage policy, enabling Ryax workflows to seamlessly direct data to edge resources for low-latency access, cloud storage for scalability, or a hybrid combination of both.

Another critical functionality is the enhanced adaptability of workflows to varying operational conditions. This integration enables Ryax workflows to align with the autonomous operation features of Edge Storage, ensuring uninterrupted execution even in isolated environments. By leveraging the metadata and erasure coding capabilities provided by the Edge Storage Gateway, workflows can reliably access, process, and store data on local edge resources during periods of disconnection from cloud storage.

3.3.5.2 High-Level Design

The Workflow Manager is a highly intuitive tool aimed at application developers. It empowers users to define data workflows for their applications, automating the assignment of computing and storage resources that serve the workflows. Through the fine integration of the Ryax Workflow Manager with the Edge Storage Gateway, users will gain enhanced control over the placement of ephemeral and persistent storage for their actions or services.

In the background, each S3 bucket accessed through the Edge Storage Gateway adheres to a storage policy that dictates how and where the data is stored. Data may be distributed to an Association’s Edge Storage resources, cloud locations, or a combination of the two. The

storage policy also defines other characteristics, such as redundancy level, encryption, and compression. On the Ryax side, the platform will be enhanced to support the configurable definition of ephemeral and persistent storage. This includes the ability to mount particular Persistent Volume Claims (PVCs) or storage volumes[4] mounting object store (S3 equivalent) buckets. As a result, storage requirements for each action can be closely matched, ensuring workflows benefit from optimal resource allocation and performance.

3.3.5.3 Interfaces

The fine integration between the Edge Storage Gateway and the Ryax Workflow Manager will make use of the following interfaces.

Interface ID	WP3T2ESG-I
Description	This interface enables Ryax to retrieve and interpret metadata and storage policies from the Edge Storage Gateway. It allows Ryax to query bucket configurations, including redundancy, encryption, and data distribution, ensuring workflows are aligned with predefined storage policies.
Component providing the interface	Edge Storage Gateway
Consumer components	Ryax Workflow Manager
Type of interface	REST
State	Asynchronous
Responsibilities	CC, RYAX

Interface ID	WP3T2ST-I
Description	This interface allows Ryax workflows to interact with the Edge Storage Gateway for CRUD operations on objects, enabling real-time data access and synchronization. It also supports runtime adjustments to storage policies and facilitates autonomous operation in network-isolated environments by utilizing local edge resources.
Component providing the interface	Edge Storage Gateway
Consumer components	Ryax Workflow Manager
Type of interface	REST
State	Asynchronous
Responsibilities	CC, RYAX

[4] Systems such as https://github.com/awslabs/mountpoint-s3-csi-driver will be explored and possible adaptation with MinIO may be proposed

3.3.6 EMPYREAN Aggregator with Security and Privacy Manager

3.3.6.1 Functionality Unlocking

The interaction between the EMPYREAN Aggregator and the Privacy and Security Manager (PSM) is pivotal for unlocking secure and seamless operations within the EMPYREAN platform. Each Aggregator within an Association hosts its own PSM instance, which collaborates with PSMs across other entities in the ecosystem. The PSM oversees the secure onboarding, authentication, and authorization of all entities, including users, devices, and workflows interacting with the Aggregator. By leveraging Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs), the PSM enforces robust identity management and enables privacy-preserving interactions. The Aggregator, in turn, orchestrates resource allocation, workload execution, and data management while relying on the PSM to validate and secure all operations. This collaboration ensures that only authenticated and authorized entities gain access to resources, data, and services within the Association.

A key functionality unlocked through this interaction is the dynamic policy management provided by the PSM. By using Distributed Ledger Technology (DLT) and smart contracts, the PSM facilitates real-time installation and enforcement across the architecture, ensuring compliance with access control rules. Within the Aggregator, the PSM serves as both a Policy Decision Point (PDP) and a Policy Enforcement Point (PEP), evaluating and enforcing access policies for the resources managed by the Aggregator. These policies are stored publicly in the DLT, ensuring secure and transparent access to resources between Associations across the EMPYREAN platform. This capability guarantees traceable, secure, and trusted operations across the IoT-edge-cloud continuum.

3.3.6.2 High-Level Design

The EMPYREAN Aggregator and PSM interact through a well-defined framework that integrates their functionalities into the broader architecture of the EMPYREAN platform. The design involves the following:

- **Authentication and Authorization:** When a user, device, or application interacts with the Aggregator, the PSM validates the entity's identity using DIDs and VCs. The PSM also generates Verifiable Presentations (VPs) that the Aggregator uses to verify access rights and trust levels.
- **Policy Enforcement:** The Aggregator relies on the PSM to enforce policies dynamically installed via smart contracts. These policies define access permissions, data usage constraints, and workload execution rules, and are stored in the DLT to ensure transparency.
- **Secure Data Exchange:** The PSM ensures the secure exchange of data between entities by signing and encrypting messages using advanced cryptographic techniques, such as Zero-Knowledge Proofs (ZKPs) and JSON Web Tokens (JWTs).

- **Traceability and Auditing:** The PSM records all interactions and policy changes in the DLT, enabling the Aggregator to maintain a transparent and auditable operation environment.

The Aggregator and PSM operate cohesively to provide a secure, scalable, and resilient Association capable of dynamic adaptation to evolving conditions.

3.3.6.3 Interfaces

Table 6: Privacy and Security Manager Interfaces

Interface ID	WP34PSMAGG
Description	Acts as a Policy Decision Point (PDP) and Policy Enforcement Point (PEP) for access authorization to resources managed by the Aggregator.
Component providing the interface	Privacy and Security Manager
Consumer components	Empyrean entities, Aggregator
Type of interface	Authorization
State	Synchronous
Responsibilities	UMU, ICCS

Interface ID	WP34PSMAGG-2
Description	Facilitates the verification of DIDs and VCs for all entities attempting to access the Aggregator.
Component providing the interface	Privacy and Security Manager
Consumer components	Aggregator
Type of interface	Authentication
State	Synchronous
Responsibilities	UMU, ICCS

Interface ID	WP34PSMAGG-3
Description	Enables entities to request Verifiable Credentials (VCs) from the Privacy and Security Manager.
Component providing the interface	Privacy and Security Manager
Consumer components	Aggregator, Empyrean entities
Type of interface	Credential Issuance
State	Synchronous
Responsibilities	UMU, ICCS

Table 7: EMPYREAN Aggregator Interface

Interface ID	WP4T4AGGR-I
Description	It supports scalable, efficient, and secure integration of distributed services and resources, ensuring the platform's interoperability and robust performance. The interface enables seamless interaction between the various components of the Aggregator, facilitates data exchange and workflows orchestration across the continuum.
Component providing the interface	EMPYREAN Aggregator
Consumer components	EMPYREAN Registry, Privacy and Security Manager, Service Orchestrator, Other Aggregators
Type of interface	REST and gRPC
State	Synchronous and Asynchronous
Responsibilities	ICCS, UMU

4 EMPYREAN System Operation Flows

4.1 Overview

This section presents the EMPYREAN system operation flows, organized into different sections based on the category to which the various operation flows belong, to ensure seamless integration of the platform's components. These flows detail the inter-components processes and methodologies enabling the system's functionality and supporting user interactions. The system operation flows provide a comprehensive description of the system's logic, the roles of individual components, and how data and actions are orchestrated to deliver the Association-based operation of the IoT-edge-cloud continuum. They form the backbone of the system's robustness, supporting its adaptability and efficiency across diverse applications.

4.2 EMPYREAN Ecosystem

The IoT-edge-cloud continuum integrates on-device, edge, and cloud resources to enhance application performance and address the limitations of centralized systems. This approach can significantly enhance application performance and infrastructure efficiency, effectively addressing the critical bottlenecks of data collection, transmission, and processing in centralized systems. However, many works oversimplify this continuum by treating it as a unified pipeline of universally accessible resources. In reality, the ecosystem is fragmented, with resources owned by different entities, leading to underutilization and limited integration of edge resources (e.g., on-device, on-premise, near-edge, far-edge, fog, etc.) with cloud infrastructures and markets.

EMPYREAN introduces a transformative approach to the IoT-edge-cloud continuum through the innovative concept of Associations. An Association represents a collaborative collective of IoT devices, robots, and resources spanning from the edge to the cloud. Each Association aggregates and shares computing and storage resources of diverse characteristics and capabilities, including both general-purpose and specialized units. In practice, an Association is operationally defined as one or more interconnected Kubernetes or K3s clusters, unified under shared administrative and security policies. EMPYREAN calls this novel framework the Association-based continuum (Figure 19), since it enables multiple Associations to operate simultaneously across space and time, creating a dynamic, scalable, and resilient continuum. These Associations are not static; they are dynamically formed, reconfigured, and updated based on resource owners' participation. Central cloud resources are integrated as needed to complement and enhance the capabilities of edge and IoT devices. This paradigm significantly enhances the flexibility, efficiency, and collaborative potential within the IoT-edge-cloud continuum, providing an ecosystem for innovative applications and services.

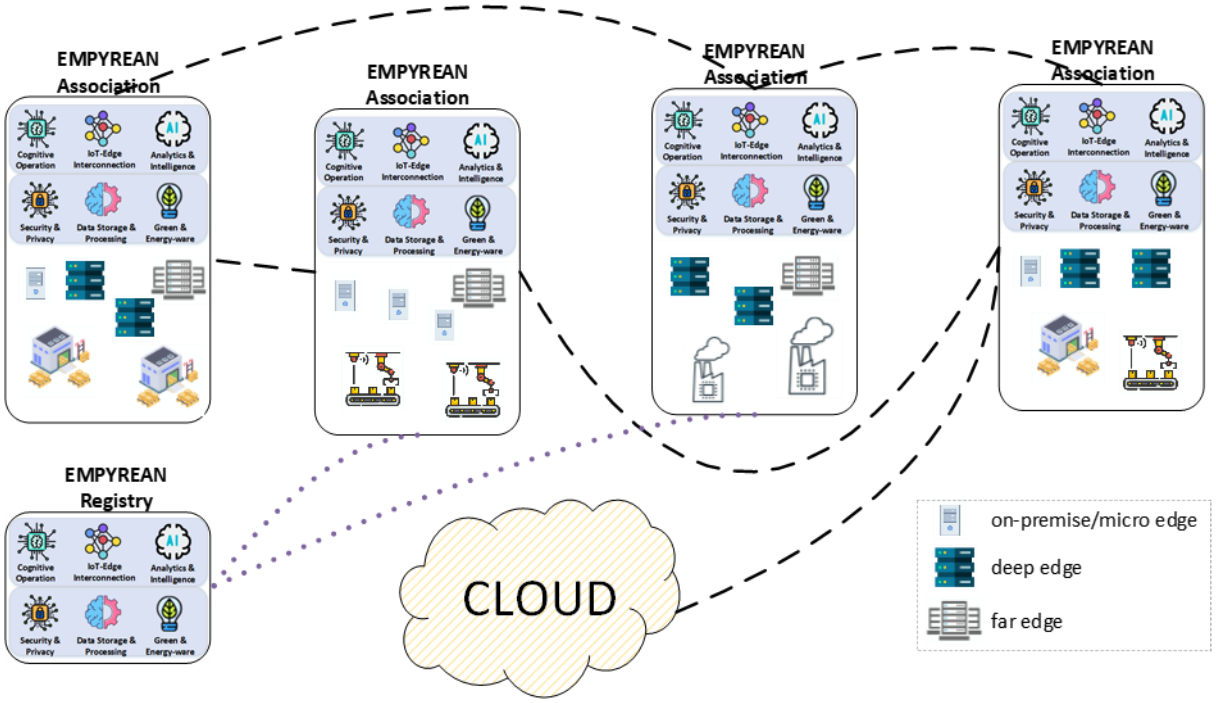

Figure 19: EMPYREAN Association-based IoT-Edge-Cloud continuum

In particular, EMPYREAN's Associations notion:

- **Supports Collaborative Continuum**: Empower organizations to build a collaborative IoT-edge-cloud continuum utilizing self-owned resources.
- **Enables Scalability**: Allows an Association to scale by involving multiple users/organizations and facilitating resource sharing between them.
- **Abstracts Complexity**: Simplifies the complexity and dynamicity of the underlying infrastructures which usually belong to different administrative domains.
- **Maximizes Resource Utilization**: Overcomes the isolation and underutilization of edge resources.
- **Promotes Self-Sufficiency**: Advocates for a self-sufficient IoT-edge continuum, acknowledging that the cloud may not always be available or its use may be prohibited for various reasons.
- **Facilitates Inter-Association Cooperation**: Supports cooperation between different Associations, enabling the use of resources under stricter access and security rules compared to resources belonging to an Association.

EMPYREAN's Association does not aim to introduce a disruptive greenfield change in the domain but instead provides a practical and viable way to organize existing and future resources within a brownfield context. By structuring the IoT-edge-cloud continuum as an Association-based continuum, EMPYREAN ensures that its software components, platform mechanisms, and decision-making algorithms are fully reusable and adaptable. This design enhances the EMPYREAN platform's flexibility and scalability, making it a robust solution for diverse and evolving resource management needs.

Figure 20 depicts the EMPYREAN ecosystem, highlighting the key stakeholders, their roles, and interactions. This ecosystem promotes the composability of infrastructures and services across the IoT-edge-cloud continuum. At its core are Associations, which enable the collaborative operation and management of virtual execution environments by pooling computational, storage, networking, and other resources. The ecosystem encompasses diverse stakeholders that interconnect seamlessly to maximize resource utilization, foster collaboration, and drive innovation across the IoT-edge-cloud continuum. These stakeholders are categorized based on their contributions and roles, including (i) *infrastructure providers*, (ii) *service providers*, (iii) *system administrators*, (iv) *application developers*, (v) *application operators*, and (vi) *end users*.

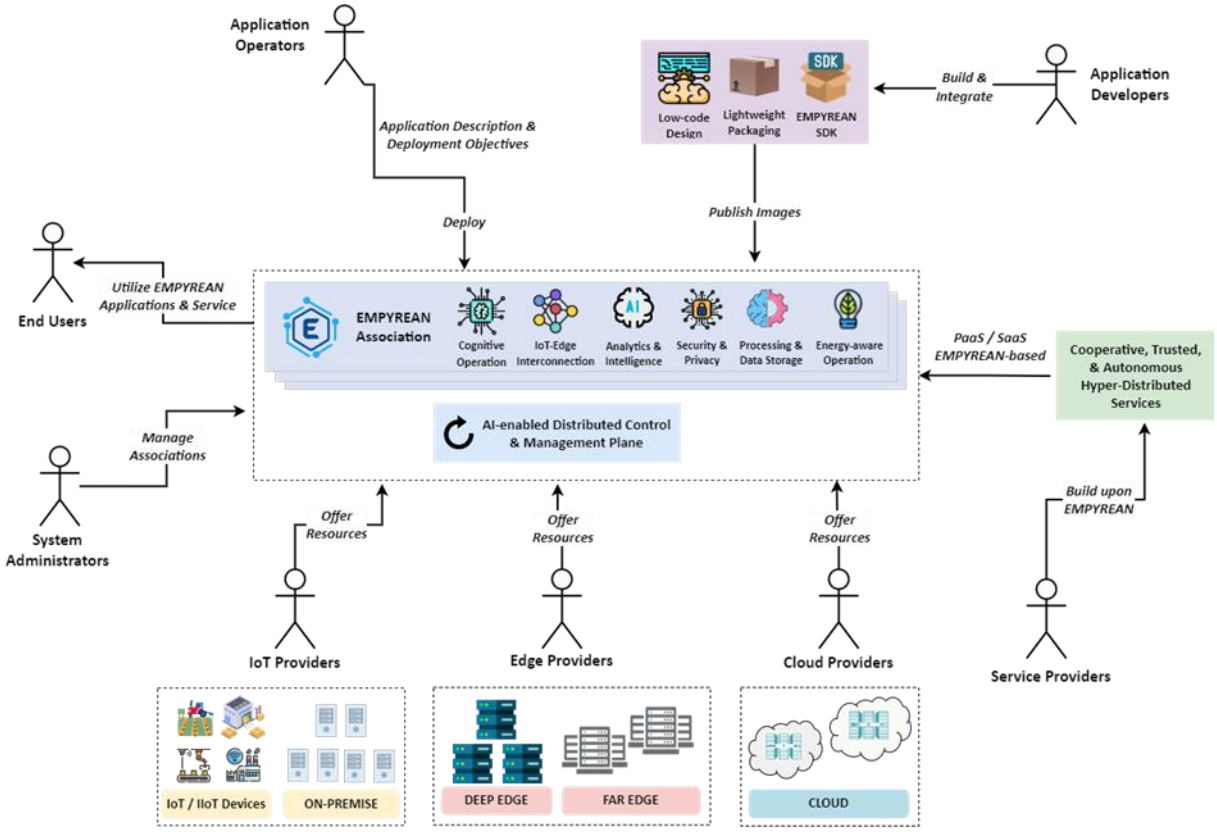

Figure 20: EMPYREAN ecosystem, key stakeholders, their roles, and interactions

The infrastructure providers include:

- IoT providers that supply IoT infrastructures distributed across the continuum, including IoT and Industry IoT (IIoT) devices, as well as on-premise low-capacity edge resources. These infrastructures serve as the primary source of data generation and the origin of service requests. Through the EMPYREAN platform, IoT providers can offer their resources to multiple vertical applications, enabling their reuse and optimizing infrastructure utilization.
- Edge providers that offer computational and storage resources located at the deep and far edge, close to or further away from the end-users and IoT devices. The offered resources can also be equipped with hardware acceleration capabilities, such as GPUs and TPUs, designed for demanding AI/ML workloads. These resources are critical for

time-sensitive operations and resource-demanding workloads, ensuring low latency and high throughput.

- Cloud providers that contribute centralized computational and storage resources to the continuum. These resources can be seamlessly integrated into the Association-based continuum to increase overall system robustness and reduce cost by efficiently handling computationally intensive and latency-tolerant workloads. They also can serve as a backbone for long-term data storage, replication, and large-scale analytics.

System administrators manage and operate the EMPYREAN platform and its underlying distributed systems. With specialized expertise in platform management and distributed systems, they ensure the seamless integration and operation of infrastructures and resources within the Association-based continuum. Their responsibilities include association management, resources onboarding and offboarding, system configuration, and user management. They ensure the integrity and reliability of the ecosystem.

Service providers deliver domain-specific, generic platforms and services built upon the Association-based continuum. Their platforms will enable the efficient deployment and autonomous adaptations of continuum-native applications, ensuring that extreme-scale distributed AI/ML workflows operate seamlessly over heterogeneous and trusted resources. By leveraging EMPYREAN's AI-enabled management mechanisms and trustworthy techniques, service providers can automate internal operations, optimize resource usage, and enable collaboration with infrastructure providers. This collaboration supports the seamless deployment of services and distributed data processing across the entire continuum, unlocking new opportunities for scalability and efficiency.

Application developers create hyper-distributed, continuum-native applications that fully leverage the potential of the EMPYREAN platform. They are skilled users with expertise in both coding and business, responsible for developing use cases and vertical application code, as well as building, upgrading, and maintaining them. They use EMPYREAN's workflow-based design tools, lightweight environment packaging, and low-code interfaces to develop infrastructure-agnostic applications. Additionally, developers oversee the deployment of these applications and debug any issues during execution. In cases of complex applications, developers may also serve as integrators, combining multiple components into fully functional solutions.

Application operators represent entities or organizations responsible for deploying hyper-distributed applications using the EMPYREAN platform. These users have business expertise and either deploy pre-existing applications or design custom application workflows based on packaged code already available within the platform. They also define deployment objectives in a generic manner, handle the deployment, manage execution, and collect results. These users benefit from the platform's advanced features, including trustworthy, autonomous, scalable, and collaborative data processing capabilities. By leveraging the Association-based continuum, they have access to a dynamic and flexible environment that supports their application needs across diverse resources within the continuum. This ensures optimal performance, adaptability, and scalability for their operational needs.

End users are individuals or entities that interact with the hyper-distributed applications deployed on the EMPYREAN platform. They use these applications to fulfil specific needs or tasks, remaining completely unaware of the underlying infrastructures, internal mechanisms, and complexities of the Association-based continuum. At the end, they benefit from advanced applications and services without requiring any technical expertise or knowledge of the platform's intricate operations.

The EMPYREAN ecosystem will offer tailored interfaces for each stakeholder role, aligning with their specific needs and expertise levels. Application developers will have access to (i) a user-friendly web-based interface for rapid prototyping, workflow management, and resource access, (ii) a command-line interface (CLI) for advanced users, enabling precise control over application development and debugging tasks, and (iii) the EMPYREAN Service Development Kit (SDK), which facilitates the implementation of infrastructure-agnostic applications by leveraging EMPYREAN's APIs and frameworks. Application operators will primarily use a streamlined web-based UI designed to allow application deployment, performance monitoring, execution management, and result collection without requiring technical expertise. System administrators will have high-level control over resource integration and platform customization via (i) a CLI for tasks such as Association management and node onboarding/offboarding, and (ii) configuration files and scripts for automating configurations and fine-tuning system parameters. The infrastructure providers and EMPYREAN service providers will perform their operations using (i) a CLI for direct interaction with EMPYREAN's core functionalities, (ii) the SDK for programmatic interaction with the platform, and (iii) configuration files and scripts for low-level setup, configuration, and management of infrastructure resources and services. Finally, the end users will interact directly with the applications deployed on the EMPYREAN platform, accessing them through their exposed interfaces.

The EMPYREAN ecosystem is inherently flexible, enabling stakeholders to take on multiple roles based on their needs and capabilities. For example, an organization can act as both an infrastructure provider and an application operator. In this dual role, it can contribute a portion of its infrastructure resources to the EMPYREAN platform through an Association, making those resources available to other EMPYREAN customers. At the same time, as an application operator, the organization can utilize the platform's decentralized intelligence and advanced application development and deployment capabilities to enhance its applications.

Serving as a bridge, EMPYREAN connects infrastructure and service providers (supply side) and application developers and end users (demand side) who require high-performing, low-latency, hyper-distributed applications. The platform aims to achieve an optimal balance by maximizing resource utilization and fostering collaboration among stakeholders, generating revenue opportunities for the supply side while ensuring the highest quality of service and user experience for the demand side.

4.3 Generic Operation Flow

The EMPYREAN platform's operational flow seamlessly integrates stakeholders, resources, and applications within the EMPYREAN platform, ensuring collaboration and enabling efficient deployment and management of Associations and applications across the continuum. This high-level flow (Figure 21) outlines the formation, management, and operation of the Association-based continuum, while subsequent sections provide specific platform-level flows.

Below is a step-by-step description of the generic operation flow:

- ***System Initialization***: The EMPYREAN system administrator initiates the platform by (i) creating the Registry, establishing the core infrastructure for the platform, (ii) bootstrapping the Identity and Authorization Engine, setting up mechanisms for managing stakeholder identities, and implementing policies for secure access and interactions, and (iii) onboarding initial stakeholders, adding the initial infrastructure and service providers by creating their identities, roles, and associated access policies.
- ***Association and Aggregator Initialization***: The administrators and service providers create Associations, establishing a collaborative environment. They set up and initialize the EMPYREAN Aggregators to manage and coordinate data workflow and application deployments across the resources contributed by the infrastructure providers.
- ***Resource Onboarding***: The infrastructure providers integrate their resources into the EMPYREAN platform by: (i) onboarding resources, adding physical and virtual resources, such as IoT devices, robots, computing units, storage, and networking infrastructures to the Association, (ii) providing resource descriptions, using templates to describe resource capabilities, configurations, and constraints, and (iii) authorization configuration, establishing policies and processes to control access to their resources.
- ***Application Development***: The system administrator adds application developers within the platform, who begin creating hyper-distributed applications by (i) designing workflows using EMPYREAN's workflow-based design tools, and (ii) packaging lightweight environments and defining infrastructure-agnostic deployment objectives using low-code interfaces.
- ***Application Deployment:*** The application operators handle the deployment and management of applications. Their responsibilities include (i) deploying applications created by developers across the Association-based continuum and (ii) managing application execution and monitoring their performance.
- ***System Automation and Optimization***: The EMPYREAN platform handles system-related operations automatically, including (i) monitoring, through continuous observation of resource and application performance across the continuum, (ii) autoscaling by dynamically adjusting resource allocation to meet application demands, and (iii) optimization by improving resource utilization, minimizing latency, and enhancing application performance through AI-enabled mechanisms.

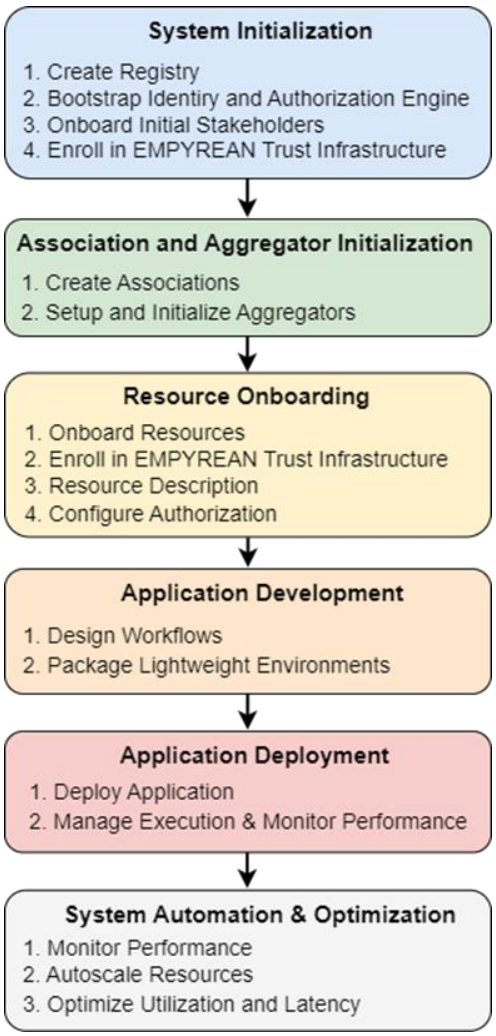

Figure 21: EMPYREAN generic operation flow

4.4 Associations Management

4.4.1 Association Setup

This operation flow is executed by EMPYREAN administrators and authorized infrastructure providers with the necessary permissions. It utilizes the core components of the EMPYREAN control and security planes already deployed within the platform. Dedicated operation flows (Section 4.5) manage the onboarding of users and the integration of IoT, edge, and cloud resources into each Association. Table 8 provides an overview of the operation flow, including its unique identifier, name, involved EMPYREAN components, associated interfaces, EMPYREAN platform requirements coverage, as well as the enabling project technologies that support its execution.

Table 8: Overview of Association setup operation flow

Op. Flow ID	OF 1.1
Name	Association Setup

Collaborators[5]	• EMPYREAN Registry (WP4.4.13)
	  ○ API Gateway (WP4.4.14)
	  ○ Service Catalogue (WP4.4.15)
	  ○ Container Image Repository (WP4.4.16)
	  ○ Association Metadata Store (WP4.4.17)
	• EMPYREAN Aggregator (WP4.4.11)
	• Privacy and Security Manager (WP3.1.1)
	• Telemetry Service (WP4.4.7)
	• EMPYREAN Controller (WP4.4.4)

Requirements Coverage[6]	F_GR.1, F_GR.2, F_GR.4, F_GR.5, F_ST.1, F_ST.2, F_SO.6, F_ASSOC.1, F_ASSOC.8, F_ASSOC.10
Enablers[7]	EN_1, EN_9, EN_10, EN_11

Figure 22 presents the operation flow, with the steps outlined as follows:

1. An administrator or infrastructure provider, enrolled with Verifiable Credentials (VCredentials) and JSON Web Token (JWT) access token generated with permissions to perform the operation, initiates the creation of an Association entity by specifying its core capabilities and embedding their access token in the request.
2. The *API Gateway* of the *EMPYREAN Registry* receives and processes the request, performing an initial validation to ensure the completeness of the information provided.
3. The *API Gateway* invokes its *Privacy and Security Manager* to authorize the requested operation, (in this case the Policy Decision Point (PDP) component that will validate through policies if the permissions to perform the operation are correct and validated), ensuring compliance with predefined policies and access control rules. After this, the Policy Enforcement Point (PEP) Proxy grant access to the request depending on the decisions of PDP.
4. Upon successful authorization, control is passed to the *Registry Manager,* which handles all subsequent interactions for setting up the Association.
5. The *Registry Manager* stores the Association's information, along with essential operational parameters, in the *Association Metadata Store*, enabling easy access for further interactions.
6. The *Registry Manager* registers the new Association in the *Service Catalogue*, making it discoverable and accessible to other services and stakeholders within the EMPYREAN platform.

[5] The identifiers refer to the EMPYREAN components descriptions as presented in deliverable D2.2 (M7).
[6] Requirements identifiers introduced in D2.1 (M6) and their descriptions are also available in the appendix.
[7] These identifiers refer to EMPYREAN enablers as described in deliverable D2.1 (M6).

7. The next step is to assign the new Association to an *EMPYREAN Aggregator* for management. The EMPYREAN platform supports two approaches for this operation, either using an existing *Aggregator* or automatically deploying a new *Aggregator* and assigning it to the newly created Association.
    - Scenario A (7.1.ii): An existing Aggregator is assigned:
        - The Aggregator's PSM validates the assignment request.
        - A Smart Contract installs policies in the Aggregator's PDP/PEP, and the DLT is updated with the new Association.
    - Scenario B (7.2.ii): A new Aggregator is provisioned:
        - The Registry Manager deploys a new Aggregator using blueprints and contacts the EMPYREAN Controller.
        - The Aggregator configures policies through its PSM, registers them in the DLT, and integrates with the new Association.
8. After assigning the Association to an *Aggregator*, the Privacy and Security Manager of the Aggregator setups the rules and policies about onboarding and resource usage within the Association through its Policy Administration Point (PAP) component.
9. Next, the *Registry Manager* updates the *Service Catalogue* to reflect the assignment.
10. The *Registry Manager* notifies the *Storage Service* about the new Association, enabling onboarding storage resources within the Association.
11. Finally, the *Registry Manager* informs the *Telemetry Service* about the new Association, enabling relevant data collection and monitoring processes to begin.
12. The Association is successfully created and available for use within the platform.

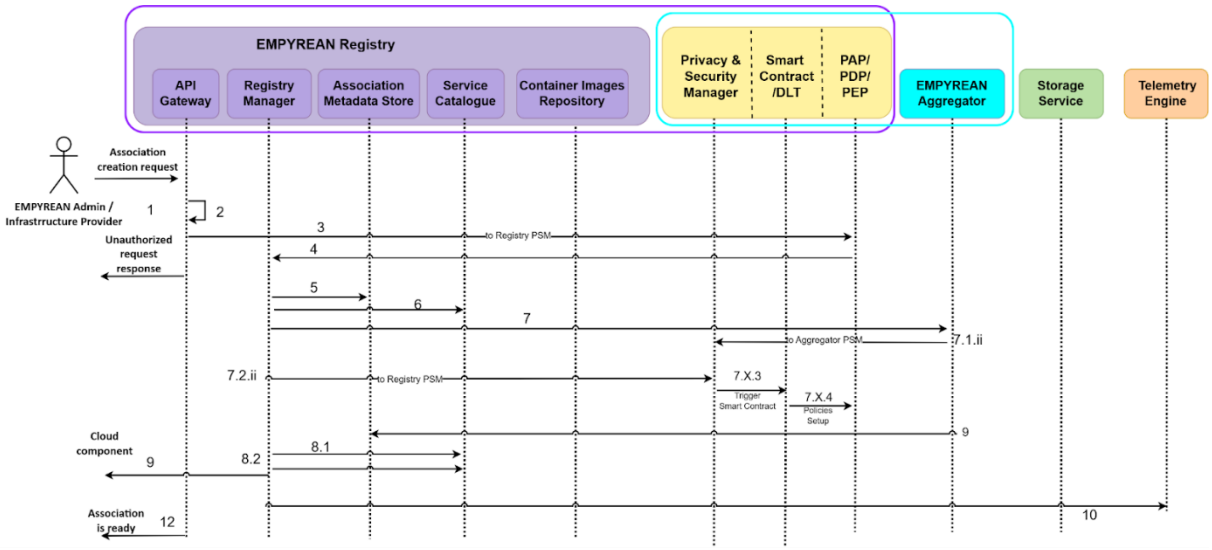

Figure 22: Association setup operation flow – Steps involved and interactions

4.4.2 Association Deletion

EMPYREAN administrators and authorized infrastructure providers with the necessary ownership permissions carry out this operation flow. The operation flow utilizes the platform's core control, management, and security components, which are already deployed and operational within EMPYREAN. Compared to OF1.1, this operation flow is more complex, as it must account for existing interconnections, active workloads, and stored data within the Association. It delivers a detailed and secure deletion process, ensuring all associated workloads, data, and resources are appropriately managed to prevent unauthorized access or data leaks. Table 9 outlines the operation flow and Figure 23 depicts the steps and interactions involved.

Table 9: Overview of Association deletion operation flow

Op. Flow ID	OF 1.2
Name	Association Deletion

Collaborators	• EMPYREAN Registry (WP4.4.13)
	  o API Gateway (WP4.4.14)
	  o Service Catalogue (WP4.4.15)
	  o Association Metadata Store (WP4.4.17)
	• EMPYREAN Aggregator (WP4.4.11)
	• Privacy and Security Manager (WP3.1.1)
	• Telemetry Service (WP4.4.7)
	• Service Orchestrator (WP4.4.1)
	• Decision Engine (WP4.1.3, WP4.1.4.)
	• EMPYREAN Controller (WP4.4.4)
	• Decentralized and Distributed Data Manager (WP3.2.3)
	• Edge Storage Gateway (WP3.2.1)

Requirements Coverage	F_GR.1, F_GR.2, F_GR.4, F_GR.5, F_ST.1, F_ST.2, F_SO.6, F_ASSOC.1, F_ASSOC.8, F_ASSOC.10
Enablers	EN_1, EN_9, EN_10, EN_11

Operation flow steps:

1. An authorized administrator or infrastructure provider initiates the deletion of an Association by specifying the Association's unique identifier.
2. The *API Gateway* of the *EMPYREAN Registry* receives and validates the deletion request, confirming among others the existence and current state of the specified Association.
3. The *API Gateway* interacts with the *Privacy and Security Manager* to authorize the deletion request, ensuring it complies with EMPYREAN's security policies and deletion control rules defined for the Association.
4. Upon successful authorization, the deletion process is handed over to the *Registry Manager*, which orchestrates the following steps to manage and coordinate the deletion process.

5. The *Registry Manager* marks the Association as "non-schedulable" in *the Association Metadata Store*, preventing the service assurance mechanisms at the upcoming steps to redeploy any affecting workloads in the specific Association.
6. The next steps focus on handling the deployed workloads and data within the Association. The *Registry Manager* requests the *EMPYREAN Aggregator* to terminate or migrate the deployed workloads that use resources from this Association. OF5.1 details this process.
7. The *EMPYREAN Aggregator* triggers the corresponding *Decentralized and Distributed Data Manager* and *Edge Storage Gateway* to manage Association's data according to established data retention policies.
8. The *EMPYREAN Aggregator* disassociates and releases all resources connected to the Association, ensuring devices are securely removed and made available for reallocation. Operation flows OF2.2 and OF2.3 detail these processes.
9. The *EMPYREAN Aggregator* requests from the Privacy and Security Manager to remove any policies, control rules, and access permissions linked to the Association.
10. The *EMPYREAN Aggregator* informs the *Telemetry Service* to stop monitoring and collecting metrics for the deleted Association.
11. The *Registry Manager* updates the *Service Catalogue*, removing the Association's entry to ensure it is no longer discoverable or accessible within the platform.
12. The *Registry Manager* deletes all metadata associated with the Association from the *Association Metadata Store*, finalizing the entity removal from the platform.
13. The Association is successfully deleted and the user is notified.

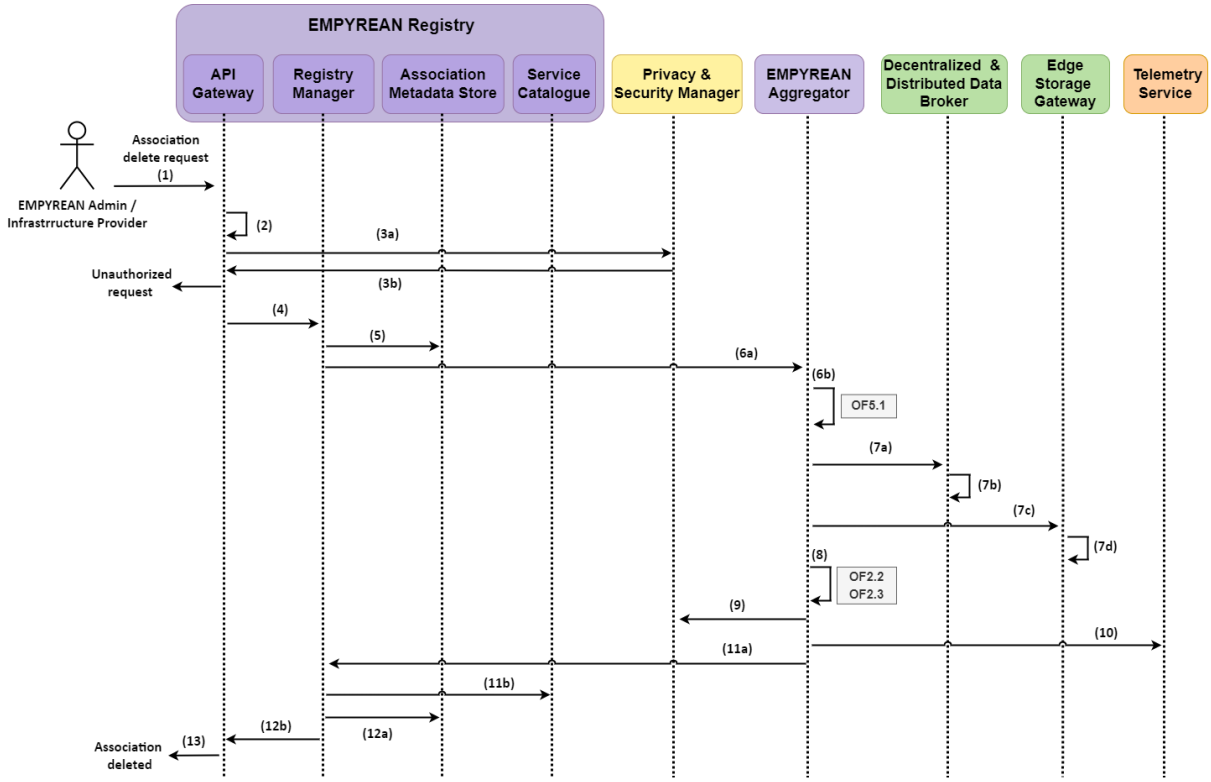

Figure 23: Association deletion operation flow – Steps involved and interactions

4.4.3 Association Update

This operation flow outlines the process of updating participation policies within an existing Association. It is carried out by EMPYREAN administrators and authorized infrastructure providers. The operation flow is structured to manage complex operations, including the eviction of resources, workload, and data migration to ensure seamless integration of updated policies. Updates to available resources and participating users within an Association are also possible. For these updates, the EMPYREAN platform provides supplementary, detailed operation flows (Section 4.5) tailored to specific operations. Table 10 provides an overview of the operation flow and Figure 24 illustrates the steps and interactions involved.

Table 10: Overview of Association update operation flow

Op. Flow ID	OF 1.3
Name	Association Update
Collaborators	● EMPYREAN Registry (WP4.4.13) ○ API Gateway (WP4.4.14) ○ Service Catalogue (WP4.4.15) ○ Association Metadata Store (WP4.4.17) ● EMPYREAN Aggregator (WP4.4.11) ● Privacy and Security Manager (WP3.1.1) ● Service Orchestrator (WP4.4.1) ● Decentralized and Distributed Data Manager (WP3.2.3) ● Edge Storage Gateway (WP3.2.1)
Requirements Coverage	F_GR.1, F_GR.2, F_GR.3, F_GR.4, F_GR.5, F_GR.6, F_ST.1, F_ST.2, F_DCM.1, F_DI.1, F_DI.2, F_DI.4, F_SO.6, F_SO.7, F_SO.8, F_ASSOC.1, F_ASSOC.2, F_ASSOC.3, F_ASSOC.4, F_ASSOC.6, F_ASSOC.7, F_ASSOC.8, F_ASSOC.10
Enablers	EN_1, EN_2, EN_4, EN_9, EN_10, EN_11

Operation flow steps:

1. An authorized administrator or infrastructure provider initiates the update request, specifying the unique identifier of the target Association and detailing the desired changes. These changes may include updating access policies, modifying resource-sharing rules, and modifying configurations.
2. The *API Gateway* of the *EMPYREAN Registry* receives the update request and performs a preliminary validation to verify the existence and current status of the specified Association.
3. The *API Gateway* interacts with the *Privacy and Security Manager* to authorize the update request, ensuring compliance with security policies and access control rules.
4. Upon successful authorization, the *Registry Manager* takes control, coordinating the necessary steps to update the Association according to the specified changes.

5. The *Registry Manager* requests from the respective *EMPYREAN Aggregator* to evaluate whether requested updates will disrupt or require adjustments to currently running applications and available resources within the Association.
6. If the analysis identifies workloads and data that need relocation due to the changes, the workflow triggers the workload and data migration process as outlined in operation flows OF4.2 and OF5.1. This ensures a smooth transition, maintaining workload availability and data integrity across the platform.
7. If any resources or nodes have to be removed from the Association, the *Registry Manager* initiates the resource offboarding process (operation flows OF2.2 and OF2.3).
8. The *Registry Manager* is notified upon the completion of the migration and offload.
9. Upon the successful handling of affected workloads and resources, the *Privacy and Security Manager* enforces the updated control and access policies to align with the new Association setup.
10. The *EMPYREAN Aggregator* is informed for the update in the Association configuration. This ensures that the *Aggregator* can adapt its resource management strategies to accommodate the changes.
11. The *Registry Manager* updates the *Service Catalogue* to reflect the new configuration, making the changes discoverable and accessible to other components and stakeholders within the EMPYREAN platform.
12. The *Registry Manager* revises the *Association Metadata Store* to record the modifications, ensuring an accurate, up-to-date repository of all Association configurations and operational parameters.
13. Finally, the request initiator is informed of the successful completion of the process.

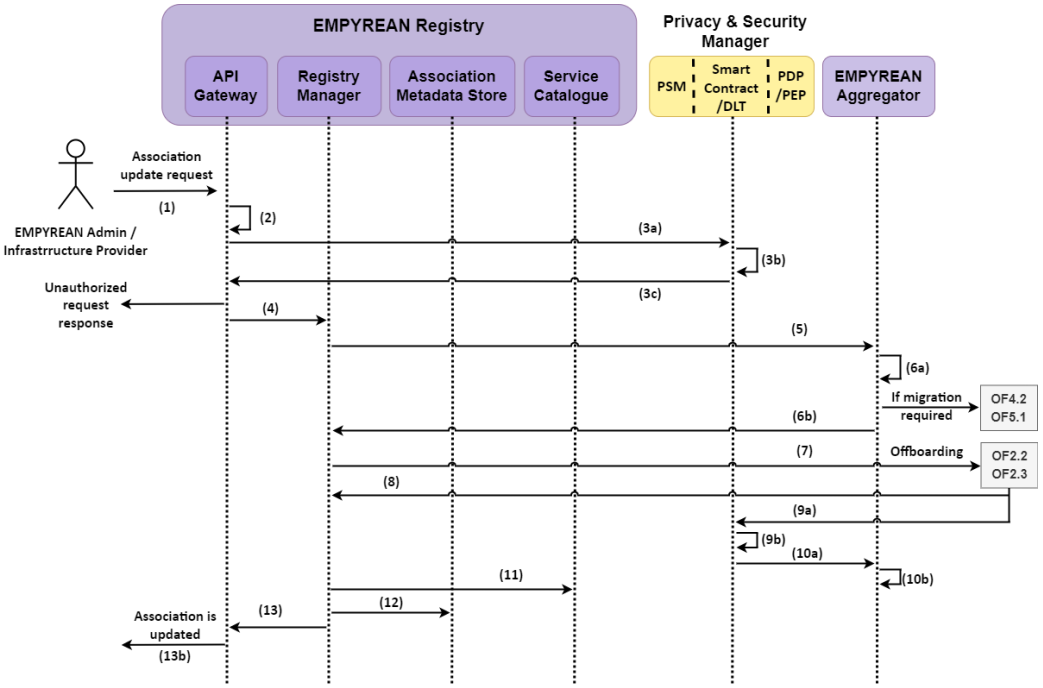

Figure 24: Association update operation flow – Steps involved and interactions

4.5 Infrastructure Resources and Users Management

4.5.1 Computing Resources Onboarding

This operation flow enables the integration of computing resources into Associations, enabling their utilization by the EMPYREAN platform for the deployment of applications. An Association will have to be populated with at least one cluster composed by at least one computing resource. The onboarding process can be initiated either manually by a designated user or performed automatically after a periodic resource discovery phase. Components such as the Privacy and Security Manager, Telemetry Service, and Service Orchestrator are involved in order to verify user authorization and ensure the eligibility of computing resources.

Table 11: Overview of computing resources onboarding operation flow

Op. Flow ID	OF 2.1
Name	Computing Resource Onboarding
Collaborators	• Privacy and Security Manager (WP3.1.1) • Aggregator API Gateway (WP4.4.12) • Service Orchestrator (WP4.4.1) • Telemetry Service (WP4.4.7) • Containers Layer Locality Scheduler (WP3.4.2)
Requirements Coverage	F_GR.1, F_GR.6, F_ASSOC.1, F_ASSOC.2, F_ASSOC.8, F_ASSOC.9, F_ST.1, F_ST.2, F_DI.6
Enablers	En_1, EN_2, EN_10, EN_11

The following steps detail the process for onboarding an entire cluster or computing resources (from the far or deep edge or the cloud) to a particular Association:

1. Initially, the resource owner must be identified and validated through the *Security and Privacy Manager* to ensure it has the necessary rights to onboard statically or dynamically resources or a cluster into the particular Association. If the owner is not already part of the Association, it has to go first through the Entity Enrolment operation flow (OF2.4).
2. The onboarding phase then begins which is performed through a mechanism or script to install the necessary packages and initiate the setup process. Resource onboarding can be performed either statically or dynamically. Moreover, the Privacy and Security Manager performs necessary validations during this phase.
    a. Static Onboarding: the designated user manually registers specific resources to a particular Association and triggers the onboarding process.
    b. Dynamic Onboarding: The resource uses a token for self-registration into one or multiple Associations. Once establishing network communication, it can query the *EMPYREAN Registry* to identify suitable Association(s) to register, based on parameters such as capacity, available data, HW accelerators, IoT devices, energy consumption, latency, and security requirements.

3. During registration, the resource owner defines the percentage of the resource to be shared within the Association. Different parameters such as energy consumption may play a role in the allocation. For dynamic onboarding, a default low percentage will be initially assigned, which the owner can adjust later.
4. The *Privacy and Security Manager* is involved during the onboarding to ensure the integrity and authenticity of resource-related data and verify the entity's credentials.
5. *The Telemetry Service* in the Association level is updated accordingly.
6. The *Aggregator*, *Service Orchestrator,* and *Local Orchestrators* are also informed to integrate the resource into their operational frameworks and workload distribution mechanisms.
7. Individual resources can be connected either as additional nodes to existing K8s/K3s clusters or participate as a new K8s/K3s cluster. The platform will also enable to some extent IoT resources to connect under the control of existing Association resources, enhancing flexibility.

Once the onboarding process is complete, the resources are fully integrated into the Association and available for workload assignments, contributing to the overall computational and operational capacity of the ecosystem.

4.5.2 Computing Resources Offboarding

This operation flow manages the removal of computing resources from the Associations. The offboarding process is initiated manually by an authorized user.

Table 12: Overview of computing resources offboarding operation flow

Op. Flow ID	OF 2.2
Name	Computing Resources Offboarding
Collaborators	• Privacy and Security Manager (WP3.1.1) • EMPYREAN Registry (WP4.4.13) • Aggregator API Gateway (WP4.4.12) • Service Orchestrator (WP4.4.1) • Telemetry Service (WP4.4.7) • EMPYREAN Controller (WP4.4.4) • Containers Layer Locality Scheduler (WP3.4.2)
Requirements Coverage	F_GR.1, F_GR.6, F_ASSOC.1, F_ASSOC.2, F_ASSOC.8, F_ASSOC.9, F_ST.1, F_ST.2, F_DI.6
Enablers	EN_1, EN_2, EN_10, EN_11

The offboarding process in EMPYREAN ensures a smooth and controlled removal of resources or clusters from an Association. The following steps outline this operation:

1. A computing resource or cluster offboarding from an Association when the resource's owner requests it from the respective *EMPYREAN Aggregator*, either after a certain period of time (lease time), or after an event is detected (e.g., the resource does not provide the requested capacity).
2. The *Privacy and Security Manager* is then notified for this explicit request or triggered event. It will then validate the integrity of the message and allow the offboarding to take place.
3. Active tasks or workloads on the affected resources are allowed to be completed before the offboarding proceeds. For tasks not yet started, the system ensures their migration to other resources within the same or different Association. More details are available in operation flows OF4.2 and OF5.1.
4. The *Service Orchestrator*, *Local Orchestrator*, *EMPYREAN Controller*, *Telemetry Service*, and *Privacy and Security Manager* are informed of the offboarding event. This ensures system-wide awareness and preparation for the resource's removal.
5. After all tasks are resolved and components updated, the resource or cluster is officially unregistered from the Association and the *EMPYREAN Aggregator* is notified.
6. The *EMPYREAN Registry* removes the relevant metadata and configurations to ensure consistency.

Once the offboarding is complete, the resource is successfully detached from the Association and is no longer available for workload assignment or participation.

4.5.3 Storage Resources Onboarding / Offboarding

To enable the use of storage resources at the Association level, these resources must first be registered with the CC's storage service. This operation is performed through a dedicated web application. Once a resource is successfully onboarded, it becomes available for use as a storage location within a storage policy. These policies define how the storage resources will be utilized and managed. Users can create S3 buckets and associate them with specific storage policies.

The Edge Storage Gateway works in conjunction with the Privacy and Security Manager to ensure the integrity and authenticity of storage resource identities. If a quota needs to be defined (e.g., to restrict the percentage of a resource that a "3rd party" user can utilize), this can be enforced through a distributed ledger.

The onboarding and offboarding processes are designed to ensure secure and seamless integration of storage resources into the EMPYREAN platform.

Table 13: Overview of storage resources onboarding and offboarding operation flow

Op. Flow ID	OF 2.3
Name	Storage Resources Onboarding and Offboarding
Collaborators	• Edge Storage (WP3.2.2) • Privacy and Security Manager (WP3.1.1) • EMPYREAN Registry (WP4.4.13) • EMPYREAN Aggregator (WP4.4.12) • Telemetry Service (WP4.4.7)
Requirements Coverage	F_GR.1, F_GR.4, F_ASSOC.1, F_ASSOC.2, F_ASSOC.7, F_ASSOC.8, F_ASSOC.9, F_ASSOC.10, F_DCM.1, F_DI.4, F_SO.4, F_SO.8
Enablers	EN_1, EN_5, EN_9, EN_10

Onboarding process:

1. The storage resource is configured to run inside the Association. For example, a helm chart that defines the containerized service that exposes the storage resource is installed in a K8s cluster. The service (a Min.IO instance) is provided by CC with a basic configuration that can be changed as needed (e.g. credentials should be changed).
2. A member of the association (user) that owns the resource logs into a web application provided by CC.
3. The member specifies the URL and credentials needed to access the resource.
4. The user can optionally set a quota on resource usage for other users. This policy is stored and managed via the *Privacy and Security Manager*.
5. The Skyflok.com backend stores this information as an *EMPYREAN Edge Storage* device, making it a part of the appropriate association.
6. The *EMPYREAN Aggregator* is updated to include the new storage resource in its available resources, while the *Telemetry Service* is updated for monitoring purposes.
7. The *EMPYREAN Registry* records the storage resource as active and accessible within the Association.

Offboarding process:

1. A member of the Association with the appropriate authorization logs into the web application provided by CC.
2. The member schedules the edge device for removal.
3. Affected storage policies are disabled, S3 buckets with these policies must also be disabled for writing until a new, valid storage policy is specified for them.
4. The actual removal should only take place once data has been migrated from the edge device.
5. The *EMPYREAN Aggregator* and *Service Orchestrator* within the Association, and the *EMPYREAN Registry* are informed for the resource offboarding.

4.5.4 Entity Enrollment

The entity enrolment operation flow facilitates the seamless onboarding and lifecycle management of entities, including users, IoT devices, robots, and resources, within the EMPYREAN ecosystem. Central to this process is the Privacy and Security Manager (PSM), which supports the issuance of Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs), enabling secure and privacy-preserving interactions.

Entities can join the trusted participant list maintained by the Distributed Ledger Technology (DLT) by providing sufficient identity proofs, ensuring compliance with the required trust standards for participation. In alignment with a Zero Trust model, guest users can be enrolled with self-generated DIDs and assigned the lowest trust score, allowing limited interaction capabilities with the platform. Their access and permissions are dynamically managed through policies defined by smart contracts and tailored to their roles and trust levels.

The operation flow also includes robust offboarding mechanisms, ensuring secure and reliable decommissioning of entities. This includes the revocation of Verifiable Credentials and removal of DIDs from the trusted participant list in the DLT, guaranteeing that access rights are fully terminated and the integrity of the system is preserved.

This approach enables comprehensive traceability, secure interactions, and adherence to the platform's access control policies throughout the lifecycle of all entities.

Table 14: Overview of entity enrollment operation flow

Op. Flow ID	OF 2.4
Name	Entity Enrollment
Collaborators	• EMPYREAN Registry (WP4.4.13) • EMPYREAN Aggregator (WP4.4.11) • Privacy and Security Manager (WP3.1.1)
Requirements Coverage	F_GR.1, F_ASSOC.2, F_ASSOC.3, F_ST.1, F_ST.2
Enablers	EN_1, EN_10, EN_11

Figure 25 illustrates the steps and interactions that take place during the entity enrollment operation.

1. An entity begins the process by generating a Decentralized Identifier (DID).
2. The generated DID is automatically stored in the *Verifiable Data Registry* (VDR) through a smart contract.
3. The entity requests a Verifiable Credential (VC) from the *Privacy and Security Manager* (PSM), including attributes necessary for its intended role.
4. The PSM verifies the request, processes the attributes, and issues the requested Verifiable Credential.

5. The entity presents the Verifiable Credential as a Verifiable Presentation (VP) to the *EMPYREAN Registry* for authentication and enrollment.
6. The *EMPYREAN Registry* interacts with the PSM to validate the VP, ensuring that the attributes and credentials comply with the platform's trust policies.
7. Upon successful validation, the PSM returns an "Authorized" status, along with a DID-signed session JWT token.

Example of Authorized Flow:

- The entity submits an Association setup request, using the session token to authenticate and authorize its access.
- The association setup request is authorized by validating it against policies stored in the Distributed Ledger Technology (DLT).
- Smart contracts in the DLT are triggered to create and enforce policies that govern the entity's interaction within the association.
- The association setup is completed, and the entity is successfully enrolled and ready to operate within the EMPYREAN platform.

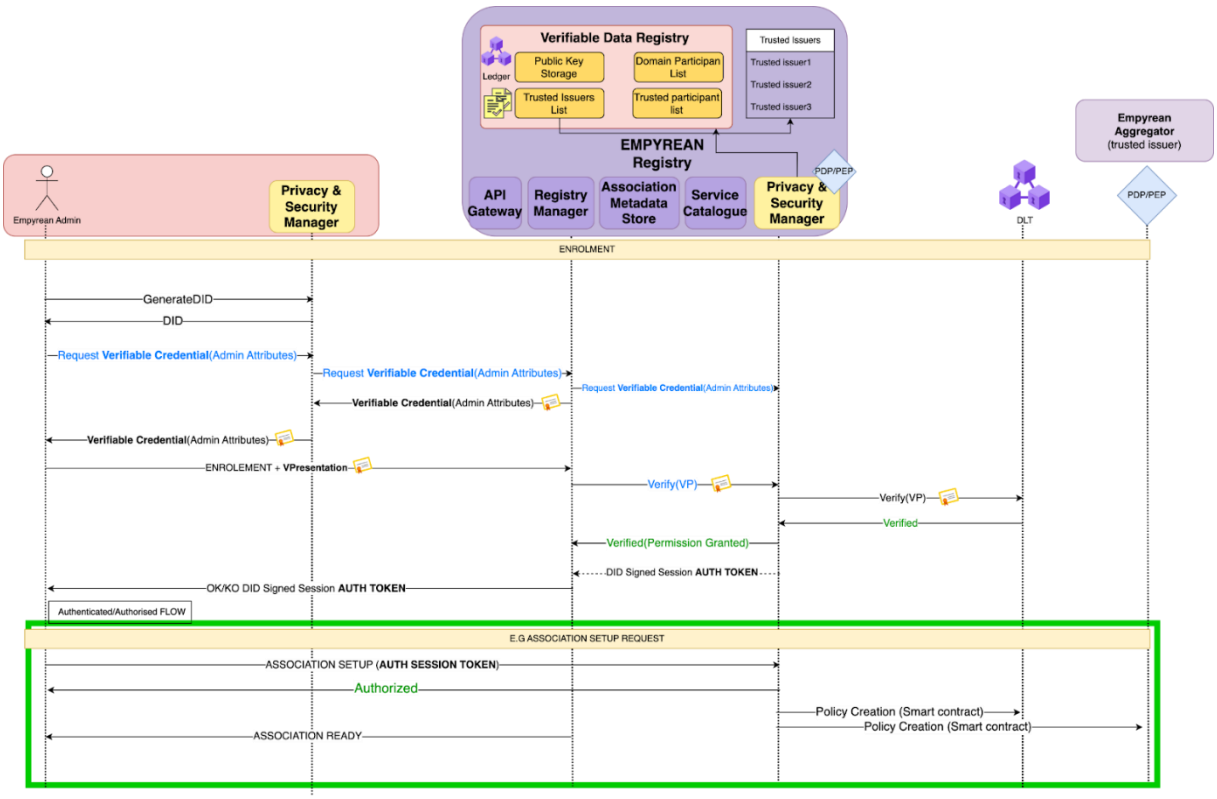

Figure 25: Entity enrollment operation flow

4.6 Application Development

4.6.1 Low-Code Application Development

The low-code application development process enables users to create and deploy applications through a user-friendly Web UI provided by the EMPYREAN Workflow Manager component. This operation flow supports different designated users with specific roles (e.g., application developers, operators, service providers) by offering high-level abstractions and pre-built tools for efficient application design and deployment.

Users log into the Web UI through the Privacy and Security Manager, ensuring secure authentication and access control. Then, they can make use of the Workflow Manager's high-level abstractions to build hyper-distributed applications by selecting pre-built actions and connecting them sequentially to define the application's global logic. Connections between actions are made by linking the outputs of one action to the inputs of the successive action, creating a clear and logical workflow. Furthermore, users can define dataflows for the actions, specifying how data moves from one part of the application to another. Expert users will have the ability to directly define the required computing and storage resources for their application through the Web UI. The non-expert users can opt for the system to automatically allocate the most suitable resources during the deployment phase, ensuring ease of use and optimal performance.

The Decentralized and Distributed Data Manager component is utilized to configure the necessary edge and cloud storage resources for the application. If resources are not explicitly defined, the system dynamically selects the most appropriate resources during the deployment phase, leveraging EMPYREAN's intelligent decision-making capabilities. The application development lifecycle also contains the phase of tracking the logs and debugging during the application execution. These functionalities are accessible directly within the Workflow Manager Web UI, enabled through integration with the Telemetry Service.

Table 15: Overview of low-code application development operation flow

Op. Flow ID	OF 3.1
Name	Low-code Application Development
Collaborators	• Workflow Engine (WP4.2.1) • Privacy and Security Manager (WP3.1.1) • Decentralized and Distributed Data Manager (WP3.2.3) • Edge Storage Gateway (WP3.2.1) • NIX-based Environment Packaging (WP4.3.1) & Application Packaging (WP4.3.3) • Telemetry Service (WP4.4.7) • Dataflow Programming Component (WP4.2.5)
Requirements Coverage	F_GR.4, F_GR.5, F_SO.1, F_SO.2, F_SO.3, F_SO.6, F_SO.13, F_SO.14, F_SO.15
Enablers	EN_4, EN_6, EN_9, EN_11, EN_14

The following operation flow steps will take place for the low-code application development:

1. The designated EMPYREAN application developer connects to the platform. The *Privacy and Security Manager* validates the user's permissions and access policies, granting access to specific Associations, features, and capabilities. These application development permissions include user rights and agreed-upon usage percentages.
2. The user defines an application in the form of one or multiple workflows that may include extract, transform, load (ETL) processes, AI/ML tasks (training or inference), data storage, and data movement between generation, processing, and storage units. This can be accomplished using either the Web UI or the CLI.
3. The user (as part of an Association or as a guest) has the ability to create a workflow, a microservice, or a batch container and deploy an application either through the UI, the CLI or the SDK/API. The workflow can be created from existing "actions", which are available in the repository. The creation of new "actions" is done during the action packaging (OF 3.2).
4. Besides defining the workflows, the actions composing the workflows, and expressing which inputs and outputs should be exchanged; the user has the ability to define how data should flow between the different actions of the workflow.
5. During the application design phase, the user can define the characteristics needed for application execution. For this, the characteristics of Associations can be retrieved to adapt the application to available resources effectively. These deployment definitions can include hard constraints or objective-based requirements for each action. These features allow users to define application needs during both development and deployment phases.
6. The platform supports storage operations (based on S3), including independent storage and retrieval tasks or storage-related operations as part of a workflow.

4.6.2 Action Packaging

The action packaging operation flow within the EMPYREAN platform provides a structured process for building, packaging, and deploying hyper-distributed applications. Applications in EMPYREAN are composed by one or multiple workflows, each comprising one or more actions designed to perform specialized tasks. The actions can be either pre-available in the platform or custom-made by the user, allowing for flexibility and adaptability across diverse use cases.

In the case of custom-made actions, users can develop actions tailored to specific requirements, preparing the code following predefined specifications and using any programming language of their choice. This operation flow leverages advanced EMPYREAN tooling, including the Action Packaging, NIX-based Environment Packaging, and Unikernels Builder components, to prepare and containerize actions for deployment, ensuring compatibility with modern cloud-native infrastructure.

Table 16: Overview of application and action packaging operation flow

Op. Flow ID	OF 3.2
Name	Action Packaging
Collaborators	• Workflow Engine (WP4.2.1) • EMPYREAN Registry (WP4.4.13) • EMPYREAN Aggregator (WP4.4.11) • Privacy and Security Manager (WP3.1.1) • NIX-based Environment Packaging (WP4.3.1) • Unikernels Builder (WP4.3.2)
Requirements Coverage	F_GR.4, F_GR.5, F_SO.1, F_SO.6, F_SO.13, F_SO.14, F_SO.15
Enablers	EN_6, EN_9, EN_13, EN_14, EN_15

The following operation flow steps will take place for the action packaging:

1. The user prepares its code following specifications related to the action packaging component providing: a YAML descriptor defining inputs, outputs, the programming language that the action is developed and other information related to the action. It also provides the action code and the dependencies of the code that will all be used to build a container.
2. The user pushes the YAML descriptor, code, and dependency definitions to an online Git repository (e.g, GitLab, GitHub, Bitbucket).
3. The user authenticates through the *Privacy and Security Manager* to validate its identity and permissions. The action packaging tool scans the repository, which after an eligibility test will start building the action.
4. Once validated, the action goes through the *NIX-based Environment Packaging* tool for typical containers and through the *Unikernels Builder* if the action is targeted for unikernels-based deployment. Both processes result in the creation of OCI-compliant containers, ensuring seamless integration into EMPYREAN's cloud-native ecosystem.
5. Once the OCI container is built, it is uploaded in the *EMPYREAN Registry* and both the *Workflow Engine* and the *EMPYREAN Aggregator* are updated and can use this new image. In particular the *Workflow Engine* will offer a specific view to provide high-level details about the action built.

4.6.3 Integration of Data Spaces

The EMPYREAN platform integrates Data Spaces to enable secure, federated data sharing and collaboration across Associations. This workflow describes the process of integrating and managing Data Spaces within the EMPYREAN architecture, ensuring compliance with standards such as Gaia-X while supporting interoperability, data sovereignty, and privacy. The EMPYREAN trust framework, built on verifiable credentials and advanced trust mechanisms, guarantees privacy and secure in interactions, maintaining transparency and accountability among stakeholders.

Table 17: Overview of data spaces integration operation flow

Op. Flow ID	OF 3.3
Name	Data spaces
Collaborators	● EMPYREAN Registry (WP4.4.13) o API Gateway (WP4.4.14) o Registry Manager (WP4.4.19) o Service Catalogue (WP4.4.15) o Data Connectors (WP4.4.18) o Association Metadata Store (WP4.4.17) ● EMPYREAN Aggregator (WP4.4.11) ● Privacy and Security Manager (WP3.1.1) ● Telemetry Service (WP4.4.7) ● Decentralized and Distributed Data Manager (WP3.2.3)
Requirements Coverage / UCs	F_GR.1, F_ASSOC.1, F_ASSOC.2, F_ASSOC.3, F_ASSOC.7, F_ASSOC.8, F_ST.1, F_ST.2, F_ST3, F_DCM.1, F_DI.4, F_SO.8
Enablers	EN_1, EN_2, EN_3, EN_4, EN_5, EN_10, EN_11

Operation flow steps:

1. An EMPYREAN administrator submits a request to integrate a new Data Space into the EMPYREAN platform via the *API Gateway* component of the *EMPYREAN Registry.* The request also includes the definition of data-sharing policies, specifying access control and usage terms.
2. The *API Gateway* invokes the *Privacy and Security Manager* to authorize the requested operation (through PDP/PEP proxy), ensuring compliance with predefined policies and access control rules.
3. Upon successful authorization, the *Registry Manager* takes control of the operation to manage all subsequent interactions for registering the Data Space.
4. The *Registry Manager* utilizes the *Privacy and Security Manager* triggering a smart contract to create and enforce the specified data-sharing and access policies. Data consumers are also added to the Data Space, their credentials and permissions for accessing data are issued.
5. The *Registry Manager* registers the new Data Space in the *Service Catalogue*, making it discoverable and accessible to services and stakeholders within the EMPYREAN platform. It also updates accordingly the *Association Metadata Store.*
6. The *Data Connectors* component is then engaged to verify the availability of the new Data Space by interacting with the respective Data Source Connector.
7. The *Registry Manager* notifies the *Telemetry Service* to monitor the availability of the Data Space at predefined intervals. Feedback by the *Data Connectors* component is also relayed to the *Telemetry Service* to ensure updated information about the Data Space's status.

8. Storage resources and data are seamlessly integrated via the *Decentralized and Distributed Data Manager* services across the EMPYREAN Associations, enabling collaborative processing and analysis while preserving data sovereignty.

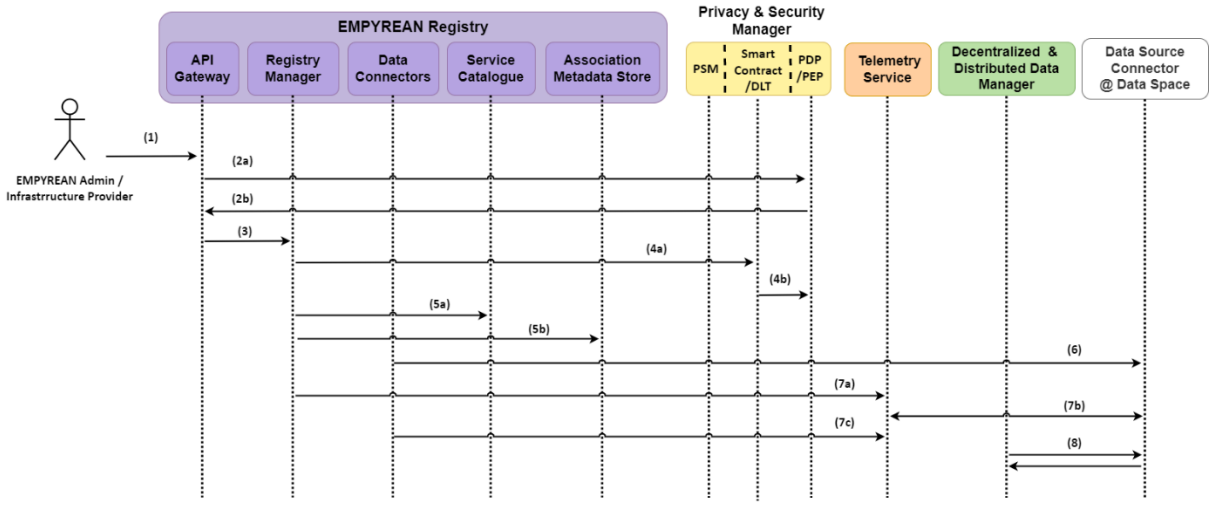

Figure 26: Data Spaces integration within the EMPYREAN platform

4.7 Application Deployment

4.7.1 Cloud-Native Application Deployment

The deployment of cloud-native applications within the EMPYREAN platform is structured into three distinct phases. The first phase involves decentralized and speculative resource orchestration, where application workloads are strategically distributed to specific Associations according to deployment objectives (OF4.1.1). The second phase encompasses hierarchical cognitive resource orchestration within the selected Associations (OF4.1.2), further refining the deployment plan based on platform-specific deployment objectives. The final phase, executed by the local orchestrator and optimized by the Containers Layer Locality Scheduler component, selects specific worker nodes across the utilized K8s/K3s clusters and also executes the actual deployment procedures as outlined in the declarative descriptions from the previous phases (OF4.1.3). Furthermore, the Workload Autoscaling component can perform continuous optimizations by adjusting further the resources allocation.

The first two phases facilitate collaborative and intelligent resource orchestration, while the third phase ensures seamless deployment across the EMPYREAN platform. Throughout these phases, the orchestration mechanisms consider various key criteria, including latency constraints, performance objectives, energy efficiency, and security requirements, guiding the optimal allocation of workloads to Associations, clusters, and underlying infrastructure resources.

This operation flow is initiated and managed by an EMPYREAN application operator who is a member of at least one Association. Before the execution of the operation flow, the application developer must provide an application description as described in OF3.1. This description includes manifests that describe the application components and their specific requirements, such as resource characteristics, network configurations, performance needs, required data, provided security, and hardware acceleration.

Table 18: Overview of cloud-native application deployment operation flow

Op. Flow ID	OF 4.1
Name	Cloud-Native Application Deployment
Collaborators	• Workflow Engine (WP4.2.1) • EMPYREAN Registry (WP4.4.13) • EMPYREAN Aggregator (WP4.4.11) • Privacy and Security Manager (WP3.1.1) • Service Orchestrator (WP4.4.1) • Decision Engine (WP4.1.3, WP4.1.4) • EMPYREAN Controller (WP4.4.4) • Telemetry Service (WP4.4.7) • Analytics Engine (WP3.4.3) • Containers Layer Locality Scheduler (WP3.4.2) • Decentralized and Distributed Data Manager (WP3.2.3) • Edge Storage Gateway (WP3.2.1) • Container Runtime (WP4.3.4) • NIX-based Environment Packaging (WP4.3.1) & Application Packaging (WP4.3.3)
Requirements Coverage	F_GR.1, F_GR.2, F_GR.3, F_GR.4, F_GR.5, F_GR.6, F_ASSOC.1, F_ASSOC.5, F_ASSOC.8, F_ASSOC.9, F_ASSOC.10, F_ST.1, F_ST.2, F_ST.3, F_ST.6, F_DI.1, F_DI.2, F_DI.3, F_DI.5, F_DI.9, F_SO.1, F_SO.2, F_SO.3, F_SO.4, F_SO.5, F_SO.6, F_SO.9, F_SO.10, F_SO.13, F_SO.14, F_SO.15
Enablers	EN_1, EN_2, EN_4, EN_6, EN_9, EN_10, EN_11, EN_14, EN_15, EN_17

Initial assignment of cloud-native application's microservices to EMPYREAN Associations (OF4.1.1):

1. The application operator initiates a deployment request through the *Workflow Manager*, providing the application descriptor along with deployment objectives and requirements.
2. The *Workflow Manager* retrieves the user's default *EMPYREAN Aggregator* from the *EMPYREAN Registry*. This Aggregator corresponds to one of the Associations in which the user participates and is authorized to utilize their resources (or part of them).
3. The *EMPYREAN Aggregator* receives the request and invokes its *Privacy and Security Manager* to validate it against the security policies and compliance rules of the involved Associations.

4. The request is forwarded to the *Service Orchestrator* which analyses the application descriptor and requests the *Decision Engine,* located at the same Aggregator, to initiate the multi-agent decision-making process for assigning the application's workloads to Associations within the platform.
5. The *Decision Engine* queries the *EMPYREAN Registry* to identify Associations with particular characteristics that meet the requirements for deploying application workloads. It then forwards the deployment request to the Associations' Decision Engines.
6. Each *Decision Engine* retrieves updated information on available resources from the *Telemetry Service* across its Associations. By utilizing multi-agent speculative algorithms, the *Decision Engines* provide high-level workload placement decisions that meet user requirements while optimizing resource utilization.
7. The default *EMPYREAN Aggregator* is informed of these decisions and, if necessary, coordinates with the other *Aggregators*. During this step, the *EMPYREAN Aggregator* also generates the execution blueprint for the application's microservices, augmenting and segmenting the initial application descriptor with infrastructure-specific instructions for each selected Association.
8. The default *Aggregator* updates the *EMPYREAN Registry* with the high-level assignment for tracking and compliance.
9. The successful completion of these steps triggers workflow WF4.1.2, which is executed concurrently across all selected Associations.

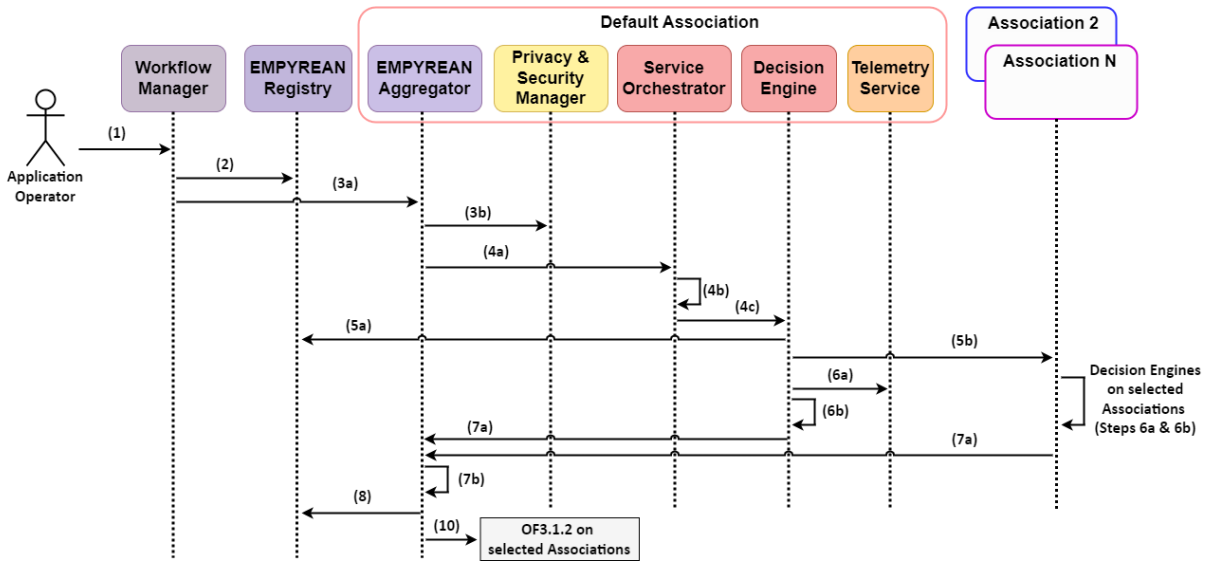

Figure 27: Initial assignment of cloud-native application's microservices to Associations

Hierarchical and cognitive orchestration at the Association level (OF4.1.2):

1. The *Service Orchestrator* receives the application description, including infrastructure-specific deployment requirements and constraints, from its *EMPYREAN Aggregator*. At this step, the application description may encompass the entire application or specific microservices, depending on the outcome of decision-making mechanisms in OF4.1.1.
2. It then requests its *Decision Engine* to determine the optimal assignment of the application workloads to individual platforms within the Association where the user is authorized to access.
3. The *Decision Engine* retrieves detailed monitoring data for the candidate edge and cloud platforms from the *Telemetry Service.*
4. Utilizing multi-objective resource allocation algorithms, the *Decision Engine* identifies the best allocation of application workloads to specific K8s and K3s clusters within the Association. This includes infrastructure-specific deployment requirements for the final phase.
5. The *Service Orchestrator* informs the *EMPYREAN Aggregator* with the application's platform-specific assignments. The *EMPYREAN Aggregator* updates accordingly its *Ryax Runner, Orchestration Drivers,* and the *EMPYREAN Registry.*
6. The *Ryax Runner* then notifies the *Ryax Worker* at the selected clusters to deploy the relevant components of the overall cloud-native application, including also any low-level deployment objectives specified by the Decision Engine.
7. Upon successful completion of these steps, operation flow OF4.1.3 is triggered and executed across all selected K8s and K3s clusters within the Association.

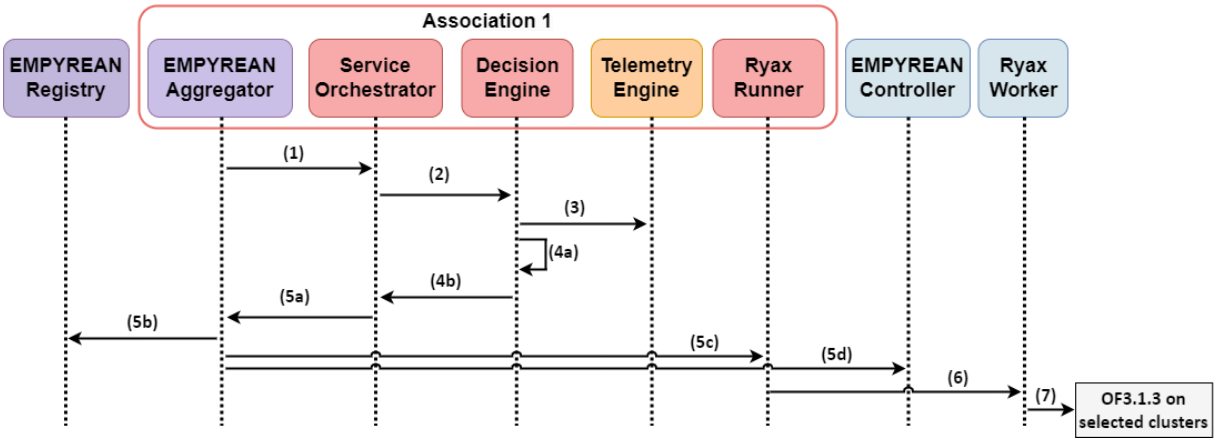

Figure 28: Hierarchical and cognitive orchestration at the Association level

Selection of worker nodes and seamless application deployment (OF4.1.3):

1. Each *RYAX Worker* dynamically generates the necessary descriptors for local-level orchestration mechanisms, specifying also the use of the *Container Layers Locality Scheduler* for assigning workloads to specific worker nodes.
2. The *RYAX Worker* interfaces with the K8s or K3s API server of the underlying cluster to deploy the containerized application components.
3. The Local Orchestrator along with the *Container Layers Locality Scheduler* makes the final assignment of application workloads to worker nodes within the specific cluster, ensuring that pods are scheduled based on the resource plans developed by the *Decision Engine*, including node affinity, tolerations, and resource limits.
4. Within EMPYREAN, each application component is deployed as a container, unifying software delivery across the IoT-edge-cloud continuum. The *Container Runtime* component is involved in this process. Additional details are covered in operation flows OF4.4 and OF4.6.
5. The *Decentralized and Distributed Data Manager* configures data pathways to provide application components with the necessary access to storage resources. This process is further detailed in operation flow OF4.3.
6. After deployment, the *EMPYREAN Controller* informs the *Service Orchestrator* regarding the application's microservices assignment to worker nodes.
7. The *Service Orchestrator* updates its *EMPYREAN Aggregator* and the *EMPYREAN Registry.* It also notifies the *Telemetry Service and Analytics Engine* to automatically monitor and analyze the performance of the specific application's microservices.
8. During application execution, the *Workload Autoscaling* component dynamically engages as needed to optimize resource allocation and ensure application performance.

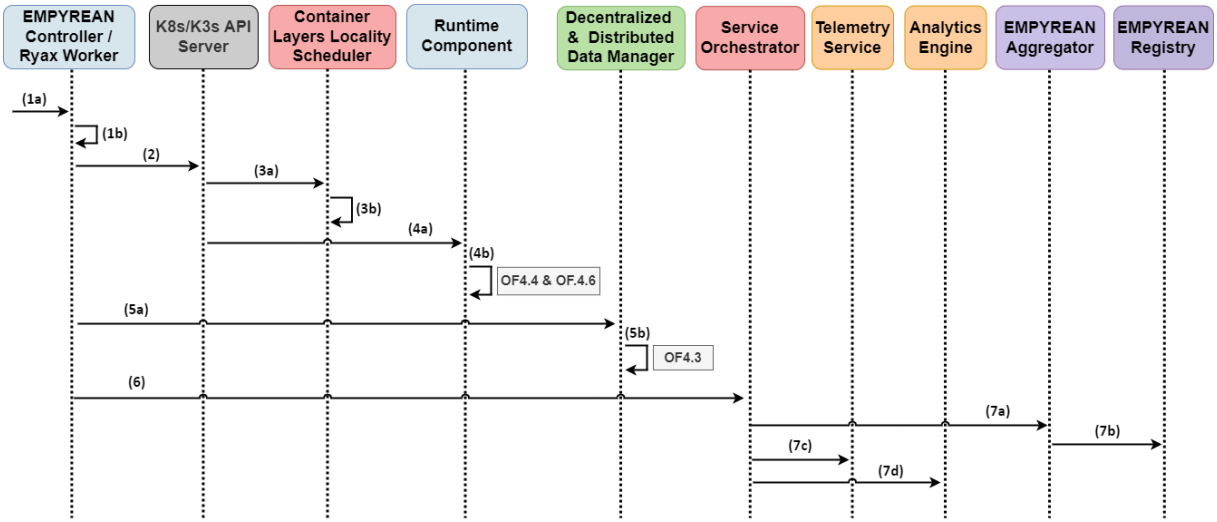

Figure 29: Selection of worker nodes and seamless application deployment

4.7.2 Intra-Association Workload and Data Migration

This operation flow is initiated by service assurance mechanisms at the Association level, either to automatically balance workloads and data within the Association or in response to events that may affect the security and performance of deployed workloads and associated data. It ensures that workloads and data are redistributed in an automated and intelligent manner to maintain performance, availability, and security within an Association.

Additionally, this operation flow can also be triggered by explicit user requests to update their deployed applications. In such cases, operation flow execution begins at step 3 and continues through to completion.

Table 19: Overview of intra-association workload and data migration operation flow

Op. Flow ID	OF 4.2
Name	Intra-Association Workload and Data Migration
Collaborators	• Privacy and Security Manager (WP3.1.1) • Analytics Engine (WP3.4.3) • EMPYREAN Aggregator (WP4.4.11) • Telemetry Service (WP4.4.7) • Service Orchestrator (WP4.4.1) • Decision Engine (WP4.1.3, WP4.1.4) • EMPYREAN Controller (WP4.4.4) • Edge Storage Gateway (WP3.2.1) • Decentralized and Distributed Data Manager (WP3.2.3) • EMPYREAN Registry (WP4.4.13)
Requirements Coverage	F_GR.3, F_GR.4, F_GR.6, F_GR.7, F_ST.1, F_ST.2, F_ST.3, F_ST.5, F_ST.6, F_ASSOC.1, F_ASSOC.4, F_ASSOC.8, F_ASSOC.10, F_DCM.1, F_DI.1, F_DI.3, F_DI.4, F_DI.5, F_DI.6, F_DI.7, F_DI.8, F_SO.3, F_SO.5, F_SO.6, F_SO.7, F_SO.8, F_SO.9, F_SO.10, F_SO.13, F_SO.14, F_SO.15
Enablers	EN_1, EN_2, EN_3, EN_4, EN_9, EN_14, EN_15, EN_17

Operation flow steps:

1. Each Association's *Analytics Engine* continuously monitors the state of available resources and deployed workloads to identify potential issues or events. It also receives notifications from the *Service Orchestrator* regarding newly deployed workloads.
2. If performance optimization is required or event detected, the *Analytics Engine* notifies the *EMPYREAN Aggregator* to trigger appropriate remediation actions. Application operators are also informed of any impending migrations to keep them updated on system changes.
3. The *EMPYREAN Aggregator* instructs the corresponding *Service Orchestrator* to adjust the deployment of affected workloads to address the detected issues. It also shares feedback from the Analytics Engines and generates a unique identifier to track the migration process.

4. The *Privacy and Security Manager* is contacted to ensure compliance with data governance policies and access control rules for the candidate infrastructure resources.
5. The *Service Orchestrator* requests the *Decision Engine* to devise an optimal migration plan that minimizes downtime and maximizes resource utilization. The *Decision Engine* is also directed to exclude any affected infrastructure resources from consideration. This workflow corresponds to cases where the *Decision Engine* can manage the request using resources solely within the same Association. A more complex workflow with inter-Association migration is outlined in operation flow OF5.1.
6. If data migration is necessary, the *Edge Storage Gateway* updates the secure storage policies.
7. The *Service Orchestrator* coordinates with the appropriate *EMPYREAN Controllers* to execute workload migration, terminating the affected workloads and redeploying them on the newly selected resources.
8. The *Decentralized and Distributed Data Manager* reconfigures the interconnection between migrating workloads to redirect traffic and maintain service availability after the migration.
9. The *Service Orchestrator* updates the related *Telemetry Engine and Analytics Engine* to adjust their configurations to continue monitoring and analysing the migrated workloads.
10. The *EMPYREAN Aggregator* receives updates from the *Service Orchestrator* regarding the migration progress and updates its internal information accordingly.
11. The *EMPYREAN Aggregator* updates the *EMPYREAN Registry* with the latest information related to the migration.

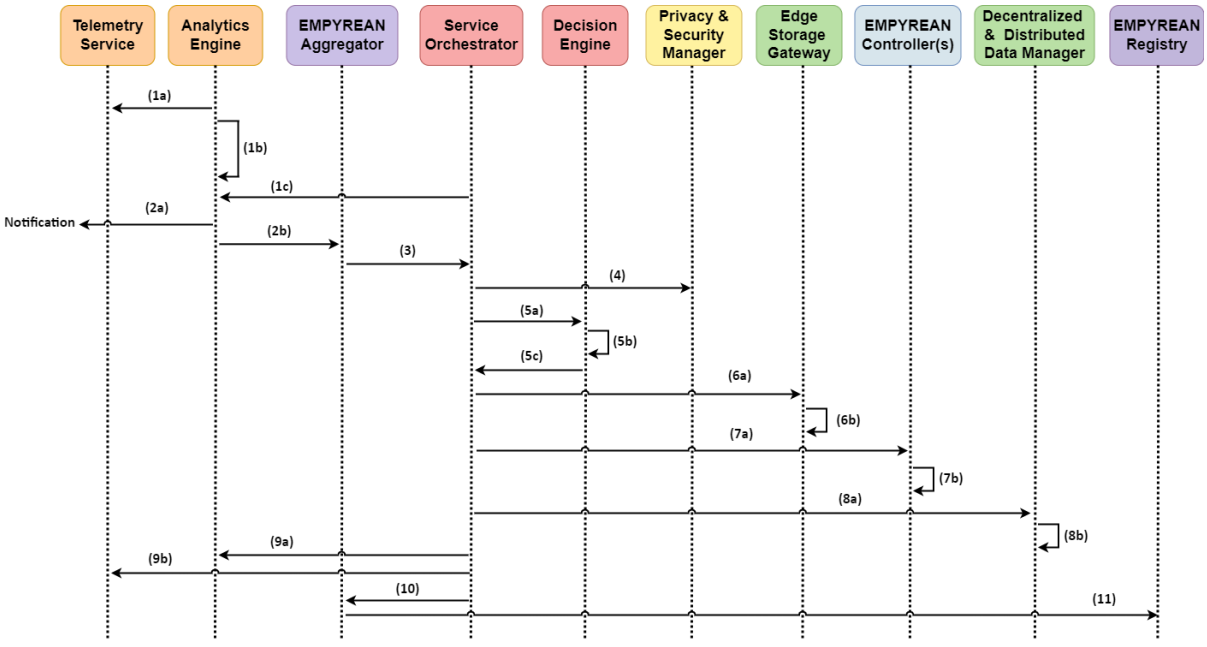

Figure 30: Intra-Association workload and data migration

4.7.3 Data Flows and Data Access

This operation flow illustrates the edge storage capabilities of the EMPYREAN platform emphasizing its integration with Associations, data management, authentication and authorization. By integrating secure storage capabilities through the Edge Storage Gateways with advanced communication frameworks such as Zenoh, EMPYREAN provides an efficient and flexible approach to data management in modern distributed environments.

Table 20: Overview of intra-association workload and data migration operation flow

Op. Flow ID	OF 4.3
Name	Data Flows and Data Access
Collaborators	• Decentralized & Distributed Data Manager (WP3.2.3) • Edge Storage Gateway (WP3.2.1) • Edge Storage (WP3.2.2) • Workflow Manager (WP4.2.1) • Dataflow Programming (WP4.2.5) • EMPYREAN Aggregator (WP4.4.11) • Service Orchestrator & Decision Engine (WP4.4.1, WP4.1.3)
Requirements Coverage	F_GR.1, F_GR.4, F_GR.5, F_ASSOC.1, F_ASSOC.5, F_ASSOC.6, F_ST.1, F_ST.2, F_ST.3, F_DCM.1, F_DI.4, F_SO.1, F_SO.6
Enablers	EN_4, EN_5, EN_6, EN_9

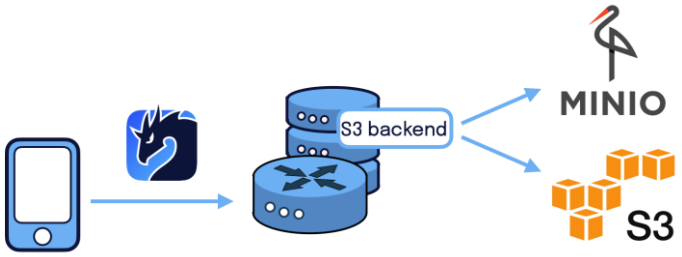

Figure 31: Seamless access of object-based storage resources through Zenoh

The Edge Storage Gateway offers an S3-compatible API to the platform's applications, enabling flexible data storage and retrieval. Applications can either directly interact with this API or access it indirectly through dataflows defined within the Workflow Manager. This flexibility allows developers to choose the most suitable interaction model based on their application requirements. In both cases, the authentication and authorization are managed by the Privacy and Security Manager, ensuring secure access. When applications utilize the storage service through a dataflow, the Zenoh and Zenoh-Flow frameworks enable seamless communication between processing nodes and S3-based data sources and sinks. The Zenoh S3-based backend[8] storage is also fully compatible with MinIO object storage (Figure 31), enabling seamless integration with the EMPYREAN Edge Storage components built on MinIO.

[8] https://github.com/eclipse-zenoh/zenoh-backend-s3

The integration of the Edge Storage Gateway and Edge Storage components with the Zenoh and Zenoh-Flow frameworks through the Decentralized and Distributed Data Manager and Dataflow Programming components implement EMPYREAN's data management capabilities, showcasing its ability to support decentralized and distributed interconnection, secure data distribution, and reliable storage. Through its S3-compatible API, the platform offers versatile storage solutions that cater to a wide range of application needs, whether for immediate processing, forwarding results, or long-term storage.

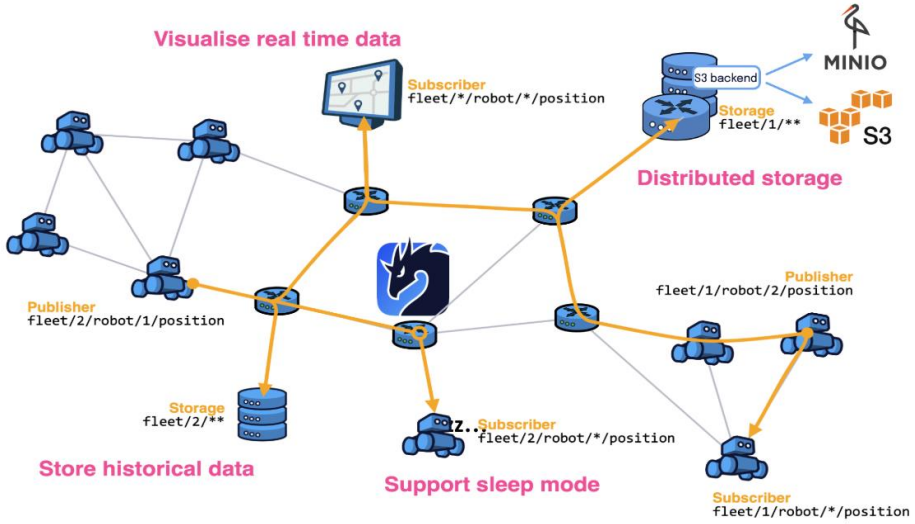

Figure 32: Decentralized and distributed interaction and data distribution with secure storage across Associations

As illustrated in Figure 32, these integrations enable seamless integration of IoT devices within a specific Association with other devices and edge resources across the EMPYREAN platform. These devices can generate data and send output to designated locations, which in turn can trigger specific operations. This dynamic interaction allows the system to process and manage data efficiently while ensuring secure and efficient data handling. Different data categories can be managed by this operation flow including (i) application data that generated by deployed application during runtime, (ii) results that need to be forwarded to other services or application components for further processing or integration, and (iii) data that is intended for later use, which can be securely stored in the S3-compatible storage locations.

Next, we provide additional details regarding the operations required for distributed interconnection and data management across multiple Associations. These operations (Figure 33) are an essential and additional step in generating the application deployment and execution blueprint, as outlined in OF4.1.1. Their objective is to automatically generate (i) the topics (referred to as "key expressions" in the Eclipse Zenoh terminology) and (ii) the encryption keys that will be used to exchange data while the application is running.

The topics are constructed based on the application descriptor and the unique identifier returned by the Privacy and Security Manager (if the user is authorized to deploy the application). The default EMPYREAN Aggregator, after received inputs from other involved Aggregators, crafts a global view of the application and determines interconnection

requirements across Associations and infrastructure platforms. For each identified interconnection (whether at the Association or platform level), it generates an encryption key and a topic name. To enable data to cross the process boundary effectively, a serialisation schema is also required (e.g. Protocol Buffers (Protobuf)) enabling seamless communication between components, regardless of their location within the distributed environment.

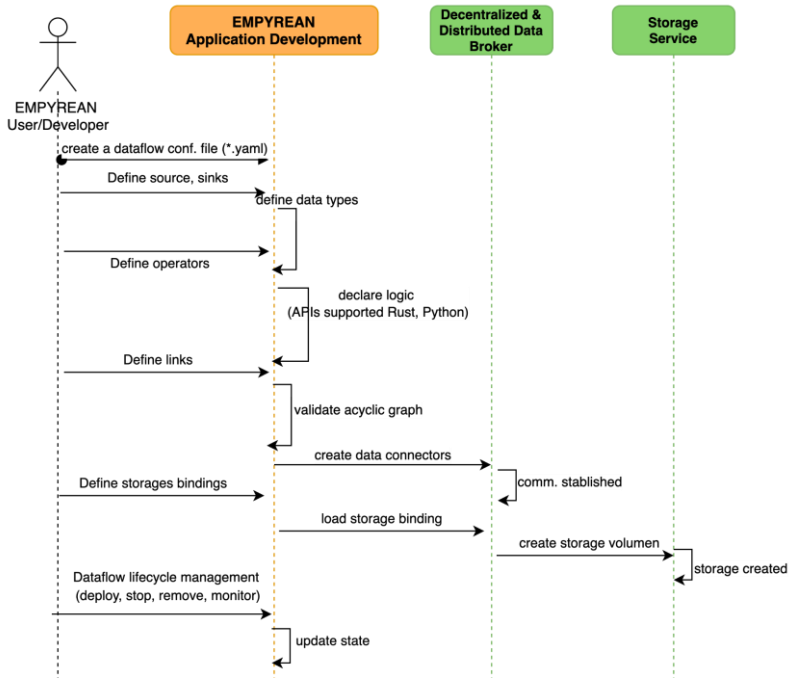

Figure 33: Workflow for creating a dataflow descriptor file using Eclipse zenoh-flow

4.7.4 Isolated and Trusted Execution

To address the increasing demand for secure and isolated execution environments, EMPYREAN employs a sandboxed deployment approach for applications across the cloud-edge continuum. This mode ensures that applications operate within highly isolated and trusted execution environments, leveraging advanced hardware and software techniques to guarantee confidentiality, integrity, and controlled resource access.

This operation flow includes three key features: (i) cloud-native sandbox deployment, (ii) unikernel-based optimization, and (iii) attestation framework.

EMPYREAN utilizes sandboxed container runtimes, such as gVisor[9] and Kata Containers[10], to enable sandboxed execution environments. These technologies isolate workloads from the host and other applications, providing enhanced security while maintaining efficient operation. To further optimize system utilization and reduce the overall system overhead from additional abstraction layers, EMPYREAN introduces a pure unikernel container runtime (urunc[11]), further eliminating the additional overhead associated with sandboxed enclaves.

[9] https://gvisor.dev
[10] https://katacontainers.io
[11] https://github.com/nubificus/urunc

This pure unikernel approach eliminates the overhead of traditional sandboxing mechanisms while preserving isolation, offering a highly optimized solution for secure execution.

In addition to the cloud-native deployment approaches, EMPYREAN leverages common practices for application/container attestation, extending their functionality to IoT devices. By employing cryptographic container signatures, Entity Attestation Tokens (EATs) and open-DICE, EMPYREAN provides a unified attestation base for all workloads and ensures trusted execution across the entire EMPYREAN platform, whether it is a simple container, a unikernel or an IoT firmware blob.

Through this operation flow, EMPYREAN provides a scalable, secure, and efficient solution for isolated and trusted application execution, addressing the challenges of modern IoT-edge-cloud ecosystems while maintaining high performance and adaptability.

Table 21: Overview of isolated and trusted execution operation flow

Op. Flow ID	OF 4.4
Name	Isolated and Trusted Execution
Collaborators	• Application Builder for Unikernels (WP4.3.2) • Container Runtime (WP4.3.4) • Privacy and Security Manager (WP3.1.1) • EMPYREAN Controller (WP4.4.4)
Requirements Coverage	F_GR.4, F_GR.5, F_ASSOC.3, F_ST.6, F_SO.13, F_SO.14
Enablers	EN_9, EN_13, EN_15, EN_16

4.7.5 Software-Defined Interconnect over RDMA and Hardware Accelerated Workloads

This operation flow in the EMPYREAN platform introduces a transformative approach to integrating hardware accelerators across the IoT-cloud-edge continuum. By leveraging NVIDIA's new disaggregated circular-buffer communication primitive over Remote Direct Memory Access (RDMA) and versatile vAccel[12] framework developed by NUBIS, this operation flow optimizes the use of specialized hardware accelerators across disaggregated environments.

The first key feature is the utilization of NVIDIA's disaggregated circular buffer primitive over RDMA that enables efficient communication for small message transfers across disaggregated systems. This subsystem facilitates the seamless integration of hardware accelerators located beyond server boundaries, enabling them to function as if they were local to the host system. The primitive significantly advances the aggregation and performance of small message transfers and reduces latency for data-intensive workloads in tightly coupled systems.

[12] https://docs.vaccel.org

The second key feature is the integration with the vAccel, a hardware-agnostic framework designed to unify the use of hardware accelerators such as GPUs, TPUs, FPGAs, and other specialized hardware across cloud, edge, and IoT infrastructures. By abstracting hardware-specific APIs, vAccel provides developers with a consistent interface to offload computationally intensive tasks to specialized accelerators without being tied to specific hardware or vendors, ensuring portability and performance optimization regardless of the underlying infrastructure. The integration of NVIDIA's circular-buffer primitive allows vAccel to extend its capabilities, enabling the use of remote accelerators over RDMA without compromising performance.

The VirtIO backend currently used by vAccel will be the integration point for this operation flow. NVIDIA's disaggregated circular-buffer primitive will replace the existing VirtIO backend, bridging VirtIO calls over the network and enabling the disaggregation of vAccel. This replacement ensures that workloads can access remote accelerators as efficiently as local ones, enhancing system flexibility and scalability.

By leveraging RDMA, the operation flow minimizes data transfer overhead between nodes and accelerators, enabling low-latency communication. This approach is particularly beneficial for workloads requiring high throughput and low latency, such as AI/ML training, inference, and real-time data processing.

Table 22: Overview of software-defined interconnect over RDMA and hardware accelerated workloads operation flow

Op. Flow ID	OF 4.5
Name	Software-Defined Interconnect over RDMA
Collaborators	• Software-defined Edge Interconnect (WP3.3.1) • EMPYREAN Controller (WP4.4.4) • Decentralized and Distributed Data Manager (WP3.2.3) • vAccel (WP3.3.4) • Container Runtime (WP4.3.4)
Requirements Coverage	F_GR.4, F_GR.5, F_ASSOC.1, F_ASSOC.5, F_SO.6, F_SO.10, F_SO.11, F_SO.12
Enablers	EN_3, EN_9, EN_12, EN_15

4.7.6 Cloud-Native Unikernels Execution

This operation flow introduces a seamless method for deploying and managing unikernel-based applications within the EMPYREAN platform, leveraging the Kubernetes-native Container Runtime, *urunc*, that is developed in the context of the project along with other core platform components. This approach bridges the gap between traditional unikernels and containerized environments, providing flexibility, security, and performance optimization.

The operation flow considers building and registering the appropriate unikernel-based images in the EMPYREAN platform (OF3.2). Once built, these images are made available in the Container Image Repository at the EMPYREAN Registry or other container image repositories, allowing seamless access for deployment. During the deployment, the EMPYREAN Controller orchestrates the execution of unikernel-based workloads, leveraging the container runtime urunc. The urunc provides seamless compatibility with Kubernetes workflows and practices, enabling operators to seamlessly integrate unikernels alongside traditional containers. It serves as the container runtime for unikernel-based applications, offering full compatibility with Kubernetes' Container Runtime Interface (CRI) and leveraging the container semantics and benefits from the OCI tools and methodology. Thus, urunc embeds unikernel images into K8s/K3s clusters as generic containers, enabling developers to leverage the robust orchestration, scaling, and monitoring capabilities of these platforms. Unlike traditional container runtimes, urunc is optimized for unikernels, reducing overhead while maintaining the lightweight and secure properties inherent to unikernel architectures.

This operation flow enhances the adoption of unikernels within the IoT-edge-cloud continuum, providing a cutting-edge, cloud-native approach to hyper-distributed application deployment and management.

Table 23: Overview of cloud-native unikernels execution operation flow

Op. Flow ID	OF 4.6
Name	Cloud-Native Unikernels Execution
Collaborators	• Application Builder for Unikernels (WP4.3.2) • Application Packaging (WP4.3.3) • Container Runtime (WP4.3.4) • Container Image Repository (WP4.4.16) • EMPYREAN Controller (WP4.4.4)
Requirements Coverage	F_GR.4, F_GR.5, F_SO.3, F_SO.6, F_SO.13, F_SO.14, F_SO.15
Enablers	EN_9, EN_13, EN_14, EN_15

4.7.7 Analytics-Friendly Data Storage and Query

This operation flow showcases a novel storage schema specifically designed for IoT time series data, optimizing storage, retrieval, and analysis processes to meet the demands of hyper-distributed environments like the EMPYREAN platform.

IoT time series data is stored using erasure coded approach, which balances data redundancy, fault tolerance, and storage efficiency. It is particularly advantageous for edge environments where storage resources are constrained. The Edge Storage and Edge Storage Gateway components provide efficient data storage with erasure coding. Next, the IoT Query Engine enables the query execution for time series analysis. These queries, similar to SQL SELECT statements, are run against the stored datasets. The efficiency in terms of data transfers is evaluated during query execution, enhancing performance particularly in edge environments.

The operation flow interactions will be enabled through a custom REST API exposed by the Edge Storage Gateway, enabling seamless storage and query operations.

The integration of this operation flow in the EMPYREAN platform provides a novel approach to managing IoT time-series data, enabling efficient storage and analytics while adhering to the high standards of performance and security expected within the EMPYREAN ecosystem.

Table 24: Overview of analytics-friendly data storage and query operation flow

Op. Flow ID	OF 4.7
Name	Analytics-Friendly Data Storage and Query
Collaborators	• IoT Query Engine (WP4.3.5) • Edge Storage Gateway (WP3.2.1) • Edge Storage (WP3.2.2)
Requirements Coverage	F_GR.1, F_GR.5, F_GR.6, F_ASSOC.1, F_DCM.1, F_DCM.2
Enablers	EN_1, EN_5

4.7.8 Workload Autoscaling

This operation flow shows the dynamic resource allocation updates that take place during the lifecycle and execution of applications, particularly focusing on the individual actions that compose them. More specifically, after the initial placement of actions on cluster nodes and the allocation of necessary resources, the Workload Autoscaling component ensures optimal resource utilization by right-sizing allocations to meet the exact needs of the executions.

The Workload Autoscaling functions in accordance with the EMPYREAN Aggregator, which provides the rules for how the workload autoscaling of the K8s/K3s clusters should perform optimizations. These rules define parameters such as the number of retries, the incremental amount RAM allocation in case of Out-Of-Memory events, timeout durations, and other critical configurations.

At the management level, the Workflow Manager orchestrates operations by controlling the Ryax Runner, while Ryax Workers (one per K8S/K3S cluster) perform the necessary adjustments on each cluster's resources. For each workflow, the Workload Autoscaling dynamically provides resource recommendations for each action based on the automatically collected utilization metrics from the past executions. Details such as the action image, hardware type being executed, and input specifications are some of the references that will be used for applying these recommendations to future occurrences. These details are provided through the collaboration between the Workflow Manager, Service Orchestrator, and EMPYREAN Aggregator.

The power of the Workload Autoscaling component comes from the internal ML-based algorithms. These algorithms analyze past executions data to identify resource utilization patterns for each action and adapt dynamically to execution variations. For past executions' insights, the component connects directly to the Telemetry Service to retrieve data related to the actual usage of resources, such as CPU, RAM, GPU and VRAM. For dynamic adaptations, the follow-up by the Workflow Engine and in particular by services such as the Ryax Runner and Worker is necessary to allow the service of retrying of failed actions to be efficient.

Table 25: Overview of workload autoscaling operation flow

Op. Flow ID	OF 4.8
Name	Workload Autoscaling
Collaborators	• EMPYREAN Aggregator (WP4.4.11) • Workflow Engine (WP4.2.1) • Telemetry Service (WP4.4.7) • Service Orchestrator (WP4.4.1)
Requirements Coverage	F_GR.2, F_GR.3, F_GR.6, F_GR.7, F_ASSOC.4, F_DI.5, F_DI.6, F_DI.7, F_DI.8, F_DI.9, F_SO.5
Enablers	EN_2, EN_6, EN_7, EN_9, EN_17

4.8 Inter-Association Operations

4.8.1 Inter-Association Workload and Data Migration

This operation flow is executed when workloads and data require migration between different Associations. This migration process ensures that performance, security, and data integrity are maintained across diverse infrastructures. Migration may be prompted by performance optimization, resource balancing, or responses to events affecting workload performance or security. Additionally, migration can be explicitly initiated by EMPYREAN users seeking to modify their deployed applications. In this case, the operation flow begins at step 3 and proceeds to completion.

Key operational scenarios related to this operation flow include (i) autonomous offloading computations by migrating non-sensitive or time-critical computations to reduce load or energy consumption on some specific Association, (ii) transient workload and data storage where a temporary user, such as a drone or moving robot, uses an Association under specific conditions to submit workload and store data, and (iii) distributed erasure-coded storage provisioning, by moving and storing data fragments across multiple Associations to enhance data availability, security, and redundancy.

Table 26: Overview of inter-Association workload and data migration operation flow

Op. Flow ID	OF 5.1
Name	Inter-Association Workload and Data Migration
Collaborators	• EMPYREAN Registry (WP4.4.13) • EMPYREAN Aggregator (WP4.4.11) • Privacy and Security Manager (WP3.1.1) • Analytics Engine (WP3.4.3) • Telemetry Service (WP4.4.7) • Service Orchestrator (WP4.4.1) • Decision Engine (WP4.1.3, WP4.1.4) • Edge Storage Gateway (WP3.2.1) • Decentralized and Distributed Data Manager (WP3.2.3) • EMPYREAN Controller (WP4.4.4)
Requirements Coverage	F_GR.1, F_GR.2, F_GR.3, F_GR.4, F_GR.5, F_GR.6, F_GR.7, F_ST.1, F_ST.2, F_ST.5, F_ST.6, F_ASSOC.1, F_ASSOC.3, F_ASSOC.4, F_ASSOC.6, F_ASSOC.7, F_ASSOC.8, F_ASSOC.9, F_ASSOC.10, F_DCM.1, F_DI.1, F_DI.2, F_DI.3, F_DI.4, F_DI.5, F_DI.6, F_DI.7, F_DI.8, F_SO.3, F_SO.4, F_SO.5, F_SO.6, F_SO.7, F_SO.8, F_SO.9, F_SO.10, F_SO.13, F_SO.14, F_SO.15
Enablers	EN_1, EN_2, EN_3, EN_4, EN_5, EN_9, EN_11, EN_14, EN_15, EN_17

Operation flow steps:

1. Each Association's *Analytics Engine* continuously monitors the state of available resources and deployed workloads to identify potential issues or events.
2. If performance optimization is required or an event is detected, the *Analytics Engine* notifies the *EMPYREAN Aggregator* to initiate appropriate remediation actions.
3. The *Privacy and Security Manager* in the source Association validates the migration request to ensure compliance with data governance policies and access control rules.
4. The *EMPYREAN Aggregator* instructs its *Service Orchestrator* to adjust the deployment of affected workloads to address the identified issues. It also shares feedback from the *Analytics Engines* and generates a unique identifier to track the migration process.
5. The *Service Orchestrator* requests the *Decision Engine* to generate an optimal migration plan, explicitly directing it to exclude any affected infrastructure resources from consideration. This operation flow corresponds to scenarios where resources within the Association are insufficient and *Decision Engine* responds without a valid migration plan.
6. The *Service Orchestrator* then instructs its *Decision Engine* to initiate the multi-agent decision-making process, aiming to reassign the affected workloads to other Associations within the platform.
7. The *Decision Engine* in the source Association queries the *EMPYREAN Registry* to identify Associations with specific characteristics, such as geographic location, hardware type, or other criteria. The orchestration mechanisms leverage the information by the *Association Metadata Store* within the *EMPYREAN Registry.*

Deployment requests are then forwarded to the *Decision Engines* of the identified Associations.

8. The *Decision Engines* across the Associations collaboratively evaluate resource availability, current workload distribution, and inter-Association network constraints to determine the most efficient allocation of resources and migration paths.
9. The source Association's *EMPYREAN Aggregator* is informed of the decisions resulting from the multi-agent orchestration.
10. If data migration is necessary, the Edge Storage Gateway updates accordingly the secure storage policies, and the *Decentralized and Distributed Data Manager* oversees the secure transfer of data to new locations.
11. At the source Association, the *Service Orchestrator* coordinates with the respective *EMPYREAN Controllers* to terminate the affected workloads. It also updates the *EMPYREAN Registry* to reflect the revised high-level assignment for tracking and compliance purposes.
12. The *Service Orchestrator* informs the relevant *Telemetry Engine* and *Analytics Engine* to stop monitoring and analysing the migrated workloads.
13. The initial *EMPYREAN Aggregator* coordinates with other selected resources. Upon successful completion of the above steps, operation flows OF3.1.2 and OF3.1.3 are triggered and executed across all selected Associations.

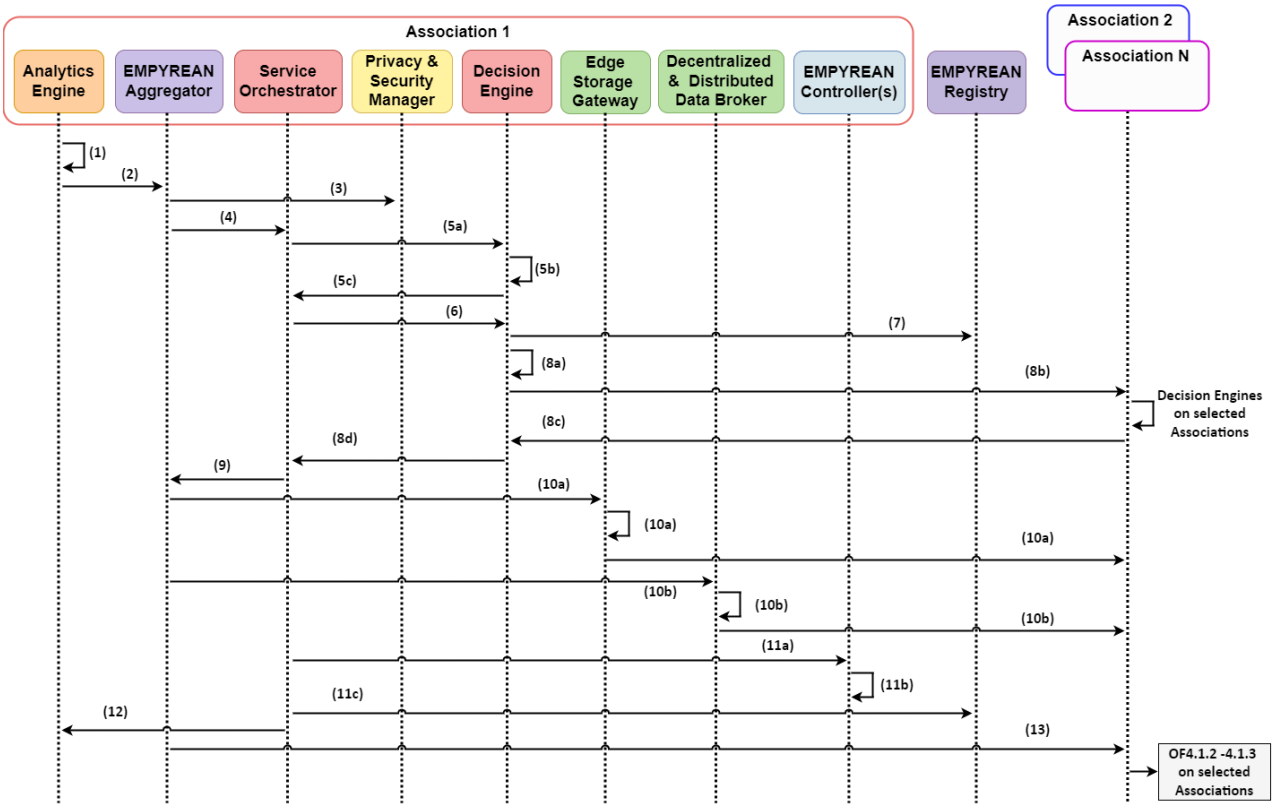

Figure 34: Inter-Association workload and data migration operation flow – Steps involved and interactions

4.9 Telemetry and Service Assurance

4.9.1 Telemetry and Observability

This operation flow outlines the telemetry and observability processes across Associations in the EMPYREAN platform, enabling seamless real-time monitoring, comprehensive data collection, and the generation of actionable insights to optimize platform performance.

By leveraging EMPYREAN's distributed telemetry infrastructure, it continuously and automatically discovers and monitors resources, including infrastructure components, robots, IoT devices, and deployed workloads. Additionally, it tracks and collects energy consumption metrics for all platform resources, contributing to energy efficiency insights. The collected telemetry data is a critical input to the distributed decision-making mechanisms within the EMPYREAN platform, enabling dynamic optimization and intelligent resource management.

Table 27: Overview of telemetry and observability operation flow

Op. Flow ID	OF6.1
Name	Telemetry and Observability
Collaborators	• Telemetry Service (WP4.4.7) • Telemetry Engine (WP4.4.8) • Monitoring Probes (WP4.4.10) • Persistent Monitoring Data Storage (WP4.4.9) • EMPYREAN Registry (WP4.4.13) • EMPYREAN Aggregator (WP4.4.11) • Service Orchestrator (WP4.4.1) • Analytics Engine (WP3.4.3) • CTI Engine (WP4.1.1)
Requirements Coverage / UCs	F_GR.2, F_GR.3, F_GR.6, F_GR.7, F_ASSOC.4, F_ASSOC.5, F_ASSOC.7, F_ASSOC.9, F_DI.1, F_DI.2, F_DI.5, F_DI.6, F_DI.7, F_DI.8, F_DI.9, F_SO.2, F_SO.4, F_SO.6, F_SO.7, F_SO.8
Enablers	EN_2, EN_4, EN_5, EN_7, EN_8, EN_9, EN_10, EN_11, EN_17

Operation flow steps:

1. *Monitoring Probes* are deployed across the Association-based continuum to continuously discover and collect real-time telemetry data, including metrics, logs, and events. Each probe is tailored to monitor specific platform resources.
2. Multiple *Telemetry Engines* within each Association continuously gather raw telemetry data from the *Monitoring Probes*. These engines pre-process the data, filtering and structuring it to ensure only relevant information is available through the telemetry infrastructure.

3. Pre-processed telemetry data is then stored in the *Persistent Monitoring Data Storage*, providing a repository for long-term reference and historical analysis.
4. The *EMPYREAN Aggregator* consolidates telemetry insights across all the Association it manages, providing a unified view of system health, resource utilization, and performance.
5. The *EMPYREAN Aggregator* also forwards high-level information to the *EMPYREAN Registry*. The *EMPYREAN Registry* maintains metadata and configuration information about monitored resources, applications, and their associations.
6. The *Service Orchestrator* within the same Association notifies the *Telemetry Engine* for changes such as workload deployments, migrations, or terminations. The engine coordinates with the respective *Monitoring Probes* to automatically adjust monitoring configurations.
7. The *Analytics Engine* in each Association and the *CTI Engine* subscribe to telemetry streams via exposed interfaces of *Telemetry Engine*. These engines analyze the collected data, detecting anomalies and security issues, observing trends, and identifying performance bottlenecks. Operation flows OF 6.2 and OF 6.3 provide detailed descriptions of these functionalities.

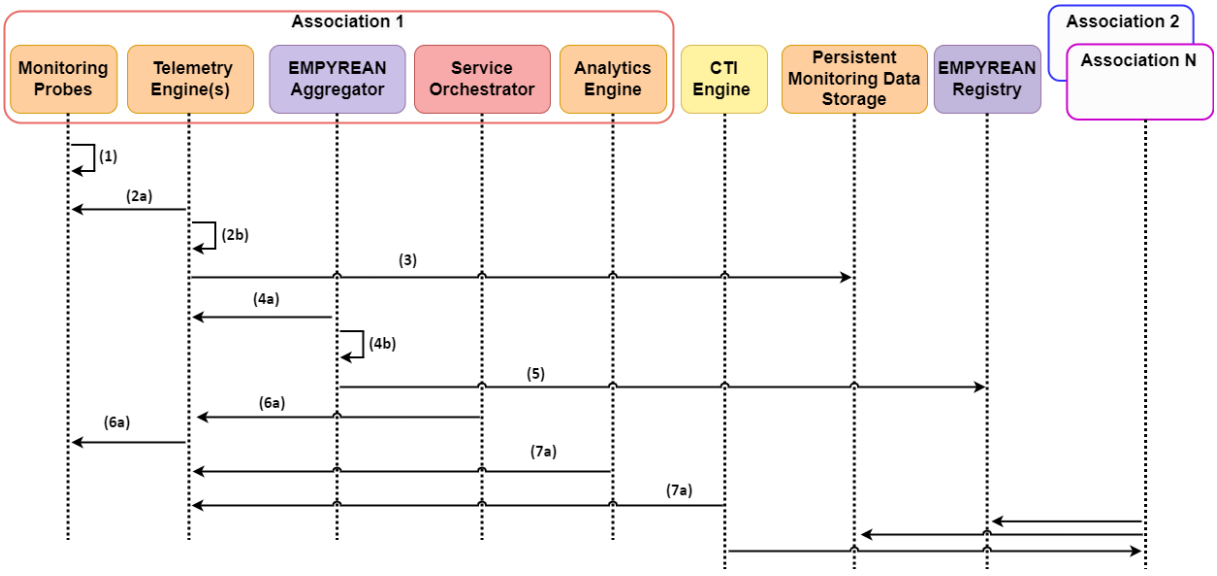

Figure 35: Telemetry and observability operation flow – Steps involved and interactions

4.9.2 Service Assurance

This operation flow represents the final step in the distributed closed-loop control system designed to maintain the desired state across the EMPYREAN Associations through self-driven, continuous adaptations. It performs AI/ML-assisted continuous analysis to detect critical situations (i.e., events, anomalies, malicious behaviours) and triggers reactively or proactively re-optimization actions within the EMPYREAN platform.

Leveraging real-time telemetry data, AI-based decision-making mechanisms, and advanced analytics, the system ensures performance, reliability, and resilience across the IoT-edge-cloud continuum. By integrating monitoring probes, telemetry engines, and automated orchestration workflows, it guarantees adherence to predefined SLAs, detecting anomalies and triggering appropriate corrective actions dynamically.

Table 28 provides an overview of the operation flow and Figure 36 illustrates the steps involved and interactions.

Table 28: Overview of service assurance operation flow

Op. Flow ID	OF 6.2
Name	Service Assurance
Collaborators	• Analytics Engine (WP3.4.3) • Telemetry Engine (WP4.4.8) • CTI Engine (WP4.1.1) • EMPYREAN Aggregator (WP4.4.11) • Service Orchestrator (WP4.4.1) • Persistent Monitoring Data Storage (WP4.4.9) • Autoscaling Optimizations (WP3.4.1) / Local Orchestrator (WP3.4.2)
Requirements Coverage	F_GR.3, F_GR.6, F_ASSOC.4, F_ST.4, F_ST.5, F_ST.6, F_DI.7, F_DI.8, F_DI.9
Enablers	EN_2, EN_4, EN_8, EN_9

Operation flow steps:

1. The *Analytics Engine* within each Association subscribes to available *Telemetry Engine(s)* and *CTI Engine* via the *Data Connector* component. This enables continuous retrieval of real-time telemetry and cyber threat intelligence data, which is pre-processed and ingested into the internal data bus for further analysis.
2. The *Analytics Engine* obtains the application execution plan by its *EMPYREAN Aggregator.* This plan includes details about the committed resources and mappings of service components to resources.
3. If necessary, the *Analytics Engine* can also query the *Persistent Monitoring Data Storage* service for historical telemetry data to support its analysis.
4. The *Analytics Engine* continuously analyzes the current state of available resources and deployed applications through the *Event Detection Engine* component.

5. Upon detecting a performance issue, the *Analytics Engine* issues alerts and forwards them to the Association's resource orchestration mechanisms via the *EMPYREAN Aggregator,* triggering remediation actions.
6. The *Service Orchestrator* contacts the *Local Orchestrator* at the affected platform. Using the current state of resources and feedback from the *Analytics Engine*, based on telemetry data the *Local Orchestrator* readjusts the initial deployment of the affected application.
7. Operation flows OF 3.1.2 and OF 3.1.3 detail the specific steps for mitigating the detected issue.

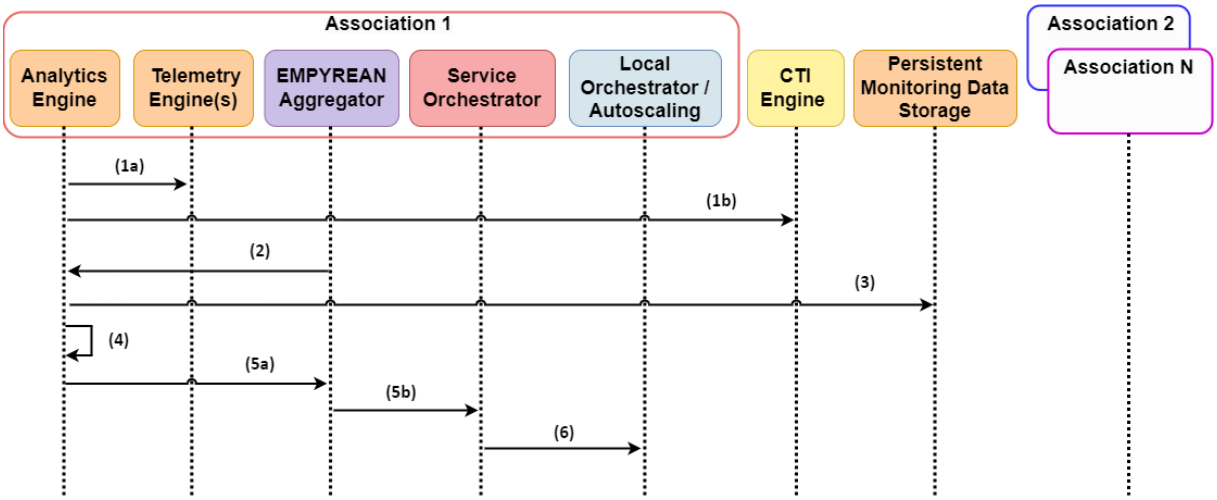

Figure 36: Service assurance operation flow – Steps involved and interactions

4.9.3 Cyber-Security Aspects

This operation flow is dedicating to enhancing the security of the EMPYREAN platform by utilizing advanced Cyber Threat Intelligence (CTI). It ensures that EMPYREAN remains resilient, adaptive, and well-prepared to defend against the sophisticated threats within the IoT-edge-cloud ecosystem.

The EMPYREAN CTI Engine collects and analyzes Cyber Threat Intelligence (CTI) from trusted sources such as the Cyber Threat Alliance (CTA) and UMU's MISP repositories to extract trends and critical information. Additionally, it integrates with the Telemetry Service to gather monitoring data across the entire EMPYREAN platform. By integrating data from these prominent sources, the engine will compile a comprehensive repository of Indicators of Compromise (IoCs), including, malicious IP addresses, domain names, file hashes, URLs, and more. This extensive dataset will serve as a foundation for thorough threat analysis and proactive defence strategies.

A user-friendly interface will streamline information retrieval, enabling security professionals to quickly search, filter, and visualize relevant threat information and intelligence. Furthermore, the CTI Engine will feature a REST API, enabling integration with orchestration and analysis tools, such as the EMPYREAN Aggregator and Analytics Engine. This integration supports the automation of threat intelligence operation flows, enabling seamless monitoring, analytics, and response.

Table 29: Overview of cyber-security operation

Op. Flow ID	OF 6.3
Name	Cyber-Security Aspects
Collaborators	• CTI Engine (WP4.1.1) • Privacy and Security Manager (WP3.1.1) • Telemetry Service (WP4.4.7) • EMPYREAN Aggregator (WP4.4.11) • Analytics Engine (WP3.4.3)
Requirements Coverage	F_ASSOC.3, F_ASSOC.4, F_ST.4, F_ST.5, F_DI.8
Enablers	EN_2, EN_8, EN_9

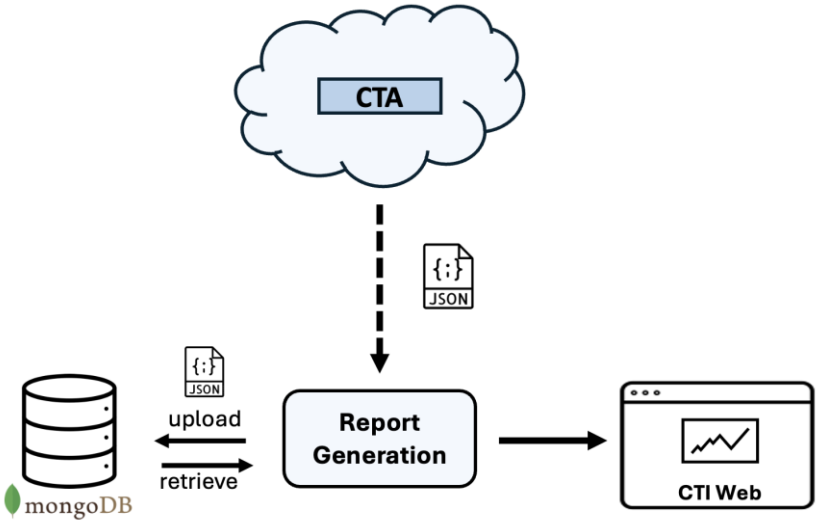

Figure 37: CTI Engine core components and dependencies.

5 EMPYREAN Architecture Design

5.1 EMPYREAN Final Architecture

EMPYREAN adopts a layered architecture, where each layer consists of discrete components that interact using well-defined and open interfaces. These interactions occur both horizontally, within the same layer, and vertically, between layers, forming the EMPYREAN platform. The final version of the high-level architecture is illustrated in Figure 38, providing a clear and structured view of the platform's design.

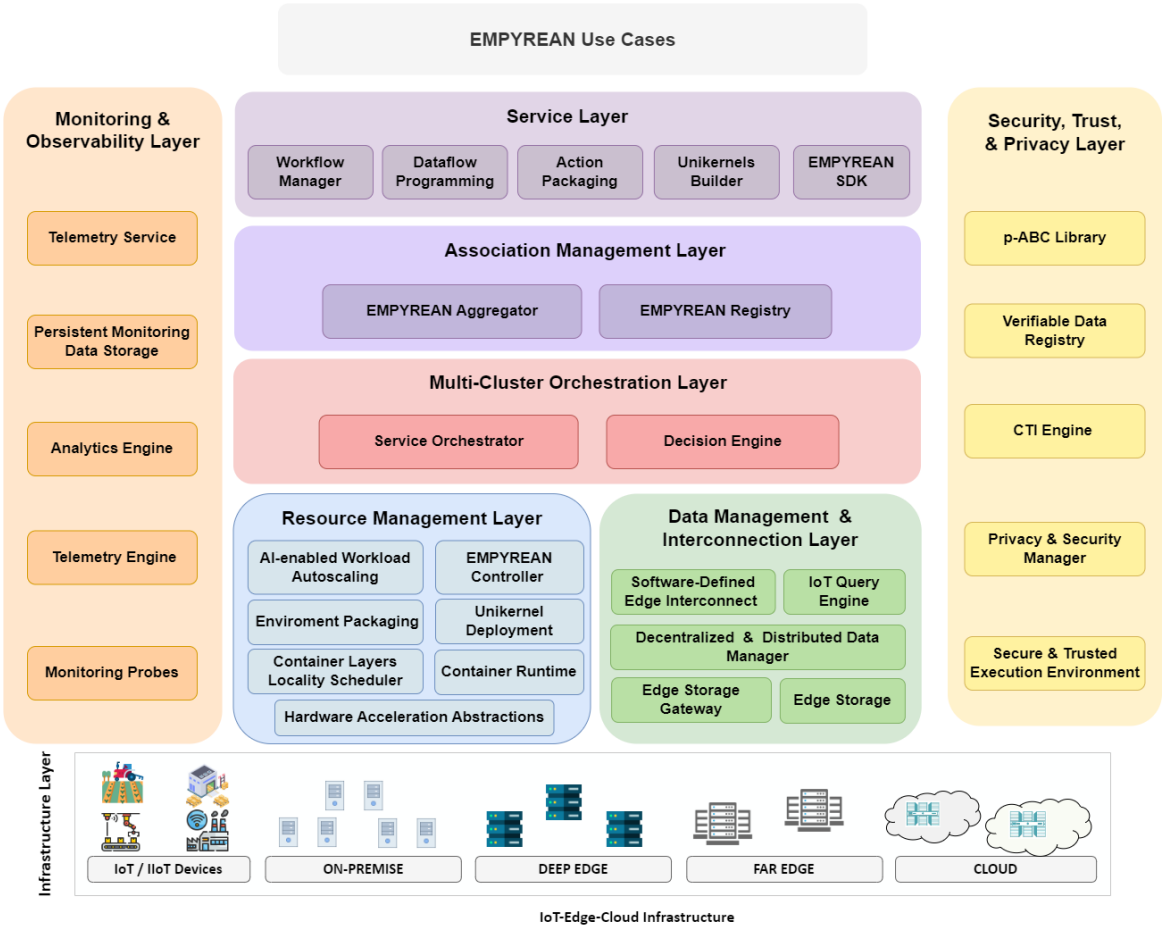

Figure 38: EMPYREAN high-level architecture

The consortium followed a top-down, iterative approach to define the EMPYREAN architecture, starting with the development of the high-level architecture that offers a conceptual overview of the platform. This high-level design identifies the key components and their functionalities without focusing on the implementation specifics. Building upon this foundation, the consortium created the logical architecture, detailed in Section 5.2, which elaborates on the logical components of the platform and aligns them with the technological solutions developed during the project.

This iterative design process incorporated the analysis and functional requirements gathered through two key tasks “T2.1: State-of-the-Art Analysis”, which provided a comprehensive evaluation of existing technologies and identified gaps to address, and task “T2.2: Concept, Use Cases and Requirements Analysis”, which ensured alignment of the architecture with the platform’s objectives and use case scenarios. This work is documented in deliverable D2.1 (M6), with the initial architecture outlined in deliverable D2.2 (M7). The process concluded with a refined version of the initial EMPYREAN architecture, as presented in this deliverable. The resulting design not only encapsulates the project’s conceptual and logical components but also establishes a robust framework for the implementation and scaling of the EMPYREAN platform.

The ***Service Layer*** encompasses components that facilitate the development of Association-native applications, offering robust support for application-level adaptations, interoperability, elasticity, and scalability across the IoT-edge-cloud continuum. This layer focuses on key aspects such as: (a) workflow design and management, simplifying the creation and orchestration of hyper-distributed applications, (b) cloud-native unikernel application development, supporting lightweight, secure, and efficient deployment models, and (c) dataflow description, enabling precise and scalable data management within applications.

The ***Association Management Layer*** dynamically manages Associations within the IoT-edge-cloud continuum. By forming resource federations, it enables seamless collaboration, resource sharing, and data distribution across various segments within the continuum. Together with the Multi-Cluster Orchestration Layer, it is central to EMPYREAN’s distributed and autonomous management, establishing a resilient Association-based continuum.

The ***Multi-Cluster Orchestration Layer*** handles service orchestration and resource management across EMPYREAN's disaggregated infrastructure. Using autonomous, distributed decision-making mechanisms, it orchestrates dynamic, hyper-distributed applications while enabling self-driven adaptations. Multiple instances of this layer’s components provide decentralized operation, optimize resource utilization, and ensure scalability, resiliency, energy efficiency, and high service quality.

The ***Resource Management Layer*** unifies the management of IoT, edge, and cloud platforms under the EMPYREAN platform. It integrates software mechanisms for both platform-level scheduling (e.g., EMPYREAN Controller, AI-enabled Workload Autoscaling) and low-level mechanisms (e.g., Unikernel Deployment). This layer operates within Kubernetes or K3s clusters and offers modularity, simplifying the integration of new hardware and software.

The ***Data Management and Interconnection Layer*** ensures dynamic communication and secure data storage between IoT devices and computing resources. Operating at both cluster and Association levels, it provides flexible and scalable data management and seamless integration of IoT, edge, and cloud resources. It also supports distributed operation, facilitating efficient operation in complex, distributed environments.

The ***Security, Trust, and Privacy Layer*** ensures secure access, privacy, and trusted execution across the EMPYREAN platform. Operating at both the cluster and Association levels, it delivers distributed trust services, enables secure and trusted execution environments, and provides controlled data access for guaranteeing data confidentiality and continuous validation of trust among entities.

5.2 EMPYREAN Detailed Architecture

Following the second iteration of the requirements analysis and design (M8-M12), a more detailed and comprehensive design of the EMPYREAN platform has been developed, as presented in Figure 39. This refined architecture incorporates insights from initial implementation efforts and further feedback from stakeholders, ensuring alignment with the platform's objectives and technical requirements.

The architecture integrates all components to be developed by the project's technical work packages (WP3–WP5), as initially introduced in deliverable D2.2 (M7). These components form the backbone of the EMPYREAN platform, enabling capabilities such as dynamic resource management, intelligent orchestration, and federated data sharing. Updated descriptions of these components, including their roles and functionalities, are provided in Section 3.

The architecture diagram in Figure 39 highlights the core interactions and essential information exchanges between components. These interactions outline the operational flows necessary for the platform's seamless functionality, which are further elaborated in Section 4. These system operation flows ensure efficient coordination across the IoT-edge-cloud continuum, supporting advanced features such as seamless deployment, enhanced security and trust, autonomous decision-making, service assurance, and data sovereignty.

The detailed architecture outlines a sophisticated system comprising three distinct platforms, each tailored to address specific operational needs across the IoT-edge-cloud continuum. A K3s cluster integrating IoT devices and on-premises resources, designed for lightweight and resource-constrained environments. This platform focuses on real-time data collection and localized processing. Two K8s clusters encompassing deep edge, far edge, and cloud resources. These platforms support scalable computing and storage capabilities, enabling the execution of complex workflows and resource-intensive tasks.

The architecture includes three Associations, dynamically built upon the resources provided by these platforms. Each Association represents a logical grouping of resources, designed to facilitate collaborative operations and workload optimization across the distributed environment. The operation of the three Associations is managed by two EMPYREAN Aggregators, which form the platform's core coordination layer. These Aggregators are responsible for orchestrating data flows, managing resource allocation, and ensuring seamless interaction between components within each Association. Additionally, the Aggregators interface with higher-level components to guarantee adherence to service-level agreements (SLAs) and alignment with overall platform objectives.

This detailed architecture underscores the EMPYREAN platform's capability to integrate diverse resources into a unified framework while maintaining:

- ***Flexibility***: tailored support for varied operational needs.
- ***Scalability***: adaptability to changing workloads and infrastructure demands.
- ***Operational Efficiency***: optimized use of resources and seamless coordination.

By integrating these components, operation flows, and the coordination capabilities of the EMPYREAN Aggregators, the platform provides a modular, scalable, collaborative, and resilient framework. This design effectively addresses the challenges of managing hyper-distributed environments.

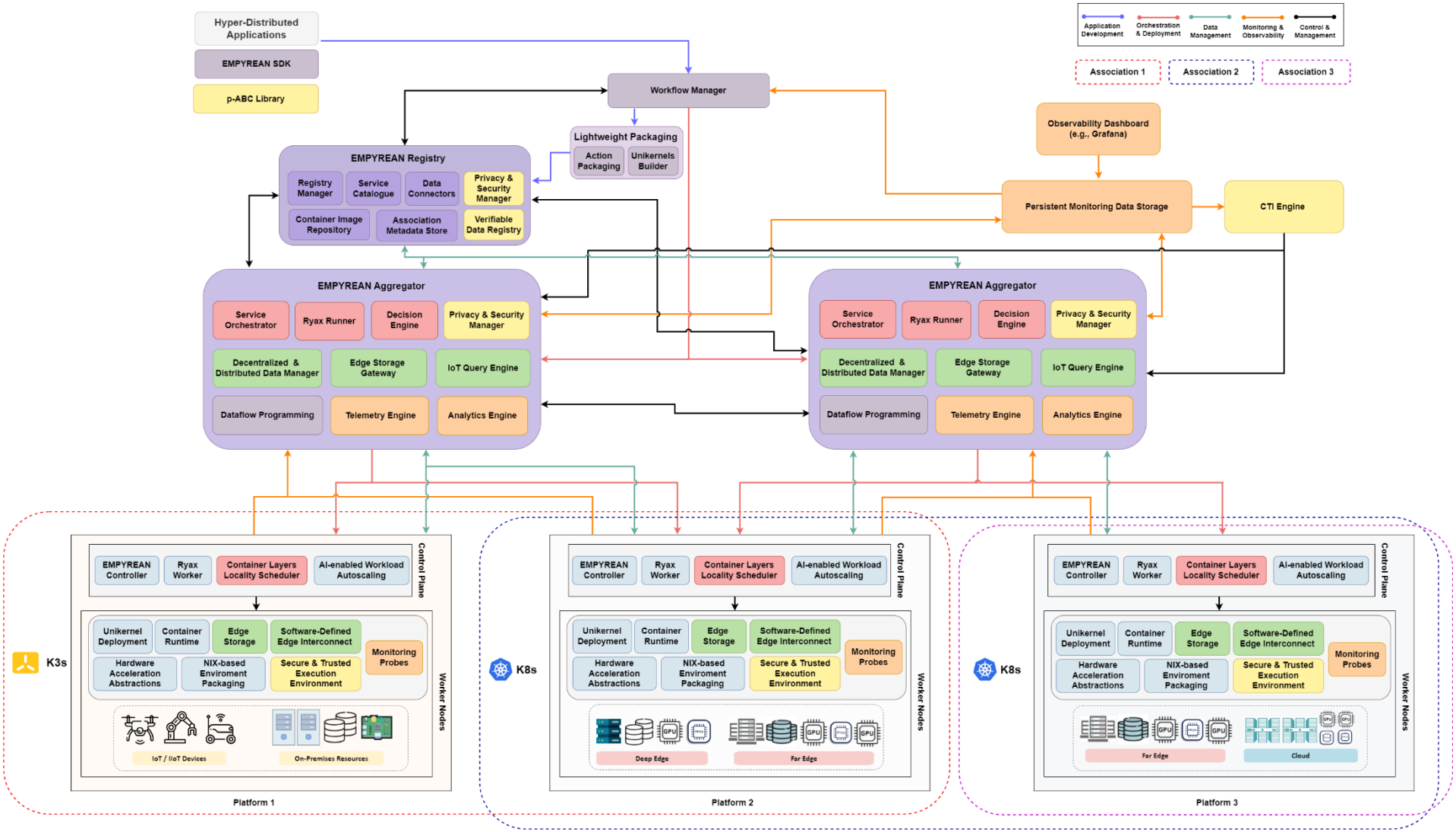

Figure 39: EMPYREAN detailed architecture, final version

5.3 Tracking KPIs

5.3.1 Methodology for Tracking KPIs

Tracking the Key Performance Indicators (KPIs) for the EMPYREAN project involves a structured systematic approach to measure, analyze, share, and optimize performance metrics. The proposed methodology ensures KPIs are monitored in a consistent and organized way. The following steps outline this methodology:

1. ***Centralize KPIs Tracking:*** The consortium will use a centralized repository for storing and managing KPI data. A shared spreadsheet will be created and used for centralizing KPIs tracking among the partners of the consortium. The initial version of the KPI tables, provided in the following sections (5.3.2 Technical KPIs and 5.3.3 Use case KPIs), will act as a starting point. These KPIs will be extended with additional data to capture the analysis, evaluation and different milestones of KPIs tracking in the context of work packages 5 and 6.
2. ***Monitor and Analyze Performance***: The shared spreadsheets will provide additional data to express the ways to monitor and analyze performance in order to evaluate whether the success criteria are being met. Lead partners assigned to each KPI will define the evaluation methods and, if needed, establish additional metrics. Once this is done, then regular reviews of the collected data will be conducted to monitor performance against predefined targets.
3. ***Communicate Results***: Following the specific milestones, the consortium will share insights and progress, both internally among the partners of the consortium and externally with stakeholders through reports and presentations. During these results we will highlight key achievements, risks, mitigation plans and areas for improvement.
4. ***Implement Corrective Actions***: Based on performance analysis, the consortium will identify and implement corrective actions to address any challenges or problems. If evaluations indicate that progress is not going as expected, strategies will be adjusted, and different evaluation procedures will be adopted to optimize performance and ensure KPI objectives are achieved.

5.3.2 Technical KPIs

The following table provides a grouping of the technical KPIs, featuring their success criteria and the Lead Partners responsible for tracking, coordinating evaluation, and validation efforts. These KPIs form the foundation for assessing the technical achievements and milestones of the EMPYREAN project.

Table 30: EMPYREAN Technical KPIs

ID	Indicator	Success Criteria	Lead Partners
T1.1	Reduce cloud and increase edge utilization via workload balancing optimization.	50% reduction in core cloud	ICCS, RYAX
T1.2	Increase reliability in the edge.	>50% increase compared to SotA	UMU, NEC
T1.3	Increase statistical multiplexing gains through associations.	x2 compared to standard execution	NUBIS, CC
T1.4	Provide low and predictable latency for hyper-distributed applications.	<1 ms for delay-sensitive apps	ZSCALE, NVIDIA
T2.1	Improve overall performance compared to SotA.	by 40%	RYAX, ICCS
T2.2	Reduce energy consumption on Associations compared to standard execution.	>25%	RYAX, ICCS
T2.3	React fast to rapid changes in computational and data demands so as to maximize the number of demands served.	between x2 and x10 increase	ICCS, RYAX
T2.4	Boost AI-driven decision-making accuracy.	>25% compared to SotA	ICCS, NEC
T2.5	Increase the robustness of the algorithms, ensuring consistent performance even under uncertain or noisy conditions.	>25% compared to SotA	ICCS, UMU
T3.1	Number of trustworthy identity and trust management processes enabled by smart contracts.	>=3	UMU, ICCS
T3.2	Accuracy of user and device verification and authentication.	> 99%;	NUBIS, UMU
T3.3	Reduction of privacy violation incidents in data sharing.	> 50%;	NEC, UMU
T3.4	Time reduction to read/write data when storing data purely on the edge compared to storage on the cloud.	by 40%	CC, ZSCALE
T3.5	Ability to access data stored on the edge when the link to the cloud is severed.	-	ZSCALE, CC
T4.1	Increase small-message transfer performance measured at the application level.	by 3x	NVIDIA, ICCS
T4.2	Improve the RDMA programming efficiency of edge applications.	-	NVIDIA, NUBIS
T4.3	Decrease the wired overhead over today's protocols like MQTT and Kafka.	by 50%	ZSCALE, ICCS

T4.4	Ensure that the amount of erasure-coded data retrieved for a query scales linearly.	-	CC, ZSCALE
T4.5	Provide an upper limit on the overhead incurred, that is either constant or a linear function.	-	CC, ZSCALE
T5.1	Reduce the development time of continuum-native applications	>20% decrease compared to SotA	RYAX, NUBIS
T5.2	Number of supported hardware architectures for seamless deployment of an application.	>3	NUBIS, RYAX
T5.3	Reduce memory and space required for deploying applications in resource-constrained IoT/Edge devices.	>70% decrease of footprint	NUBIS, RYAX
T5.4	Offload acceleration functionality to nearby devices.	>1 IoT device, >3 Edge devices	NVIDIA, NUBIS

5.3.3 Use case KPIs

The following table provides a grouping of the use case KPIs featuring success criteria and the Lead Partner responsible to track them down while coordinating their validation.

No	Indicator	Success Criteria	Lead Partners
U1.1	Transition from offline operation analysis to real-time operation fingerprint analysis.	-	IDEKO
U1.2	Ability to process real-time data	3 robots / 200 operations	IDEKO
U1.3	Ability to alert an abnormal operation	max 2sec after it occurs	IDEKO
U2.1	Development of processes that support the transition from subjective to objective, accurate and harmonised soil health data sets	-	EL ILVO
U2.2	Transition to a real- or near real-time assessment of soil, crop, and water parameters, allowing cooperated integrated farm management;	-	EL ILVO
U2.3	Reduce the time and effort needed to develop soil data-driven models.	by 25%	EL ILVO
U3.1	Fleet of heterogeneous robots working collaboratively on specific warehouse tasks with the aim of achieving human parity performance	(i.e. payback in less than 1 year).	TRAC
U3.2	Uptime of fleet despite variable network conditions	(i.e. packet loss 1%,lat> 500ms).	TRAC
U3.3	Real-time detection of network threats and efficient storage and retrieval of data.	-	TRAC

6 Use Cases Analysis

6.1 Anomaly Detection in Robotic Machining Cells (UC1)

6.1.1 Overview

In the manufacturing sector, the adoption of robots for machining tasks offers significant advantages by enhancing flexibility and reducing costs compared to traditional machine tools. Robots enable rapid adjustments to production processes and product designs, providing a high level of adaptability. This flexibility is particularly valuable in dynamic markets, where manufacturers must quickly respond to evolving consumer demands and technological advancements while maintaining precision and efficiency. However, integrating robots into machining operations necessitates rigorous process monitoring to address challenges such as precision loss, tool breakage, and other common defects encountered when machining composite materials. Effectively managing these issues is critical to improving operational efficiency and maintaining quality standards in the manufacturing industry.

The use case "Process Monitoring and Anomaly Detection in Robotic Machining Cells" aims to develop a system capable of real-time monitoring within robotic machining cells performing composite manufacturing operations using high-frequency data. These operations include turning, milling, and drilling. The system focuses on real-time signal monitoring and the detection of abnormal machining activities, enabling rapid identification and response to deviations in the machining process. This approach aims to enhance production efficiency and minimize potential losses.

6.1.2 Development and Deployment Updates

The development and deployment strategy for this use case is grounded in the typical architecture utilized by machine tool clients (Figure 40). The proposed architecture is structured into two main layers: deep-edge devices integrated with the robots and far-edge resources hosted on the client's premises. This layered design ensures an efficient allocation of computational workloads, assigning tasks to the layers best suited to handle them.

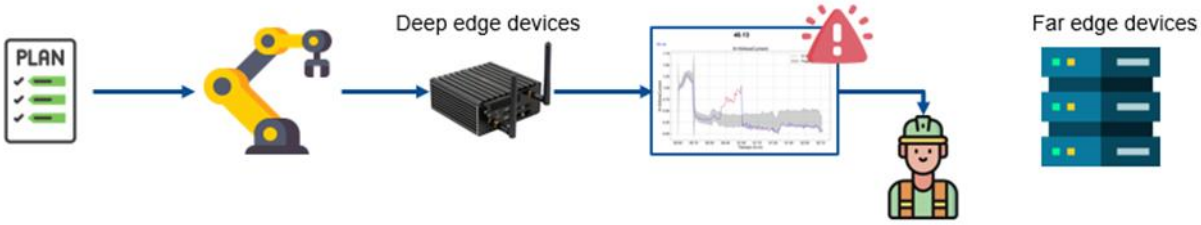

Figure 40: The typical architecture employed by machine tool clients, consisting of deep-edge devices integrated with the robots and far-edge resources hosted on-premise by the client.

In the initial deployment phase, the focus will be on setting up a single EMPYREAN Association for one robot, managed by an Aggregator. This phased approach is intended to validate the fundamental architecture and its components, providing a strong foundation for future scalability. The robots will be equipped with deep-edge devices offering various CPU, RAM, and storage configurations. Although these devices do not include GPU support, they can run containerized applications, enabling the efficient and flexible deployment of EMPYREAN components.

The far-edge devices, located on the client's premises, will manage more computationally intensive tasks. These resources will support advanced processing activities, such as real-time analytics, ensuring the seamless execution of high-demand workloads. EMPYREAN's distributed architecture features, including multi-clustering support and real-time dataflow management, will enable efficient coordination and communication between deep and far-edge layers.

During Task 5.2, "Anomaly Detection in Robotic Machining Cells Technological Developments," the current production workflows will be tailored to align with the EMPYREAN distributed architecture, as detailed in Deliverable D2.1 (Section 4.1.3, Figure 9). This alignment will take place as part of WP5 activities (M13-M28). The current production workflows will be restructured into distinct Ryax workflows (WF), as described in D2.2 (Section 6.1, Figure 25), to fully harness EMPYREAN's advanced capabilities (Figure 41, Figure 42).

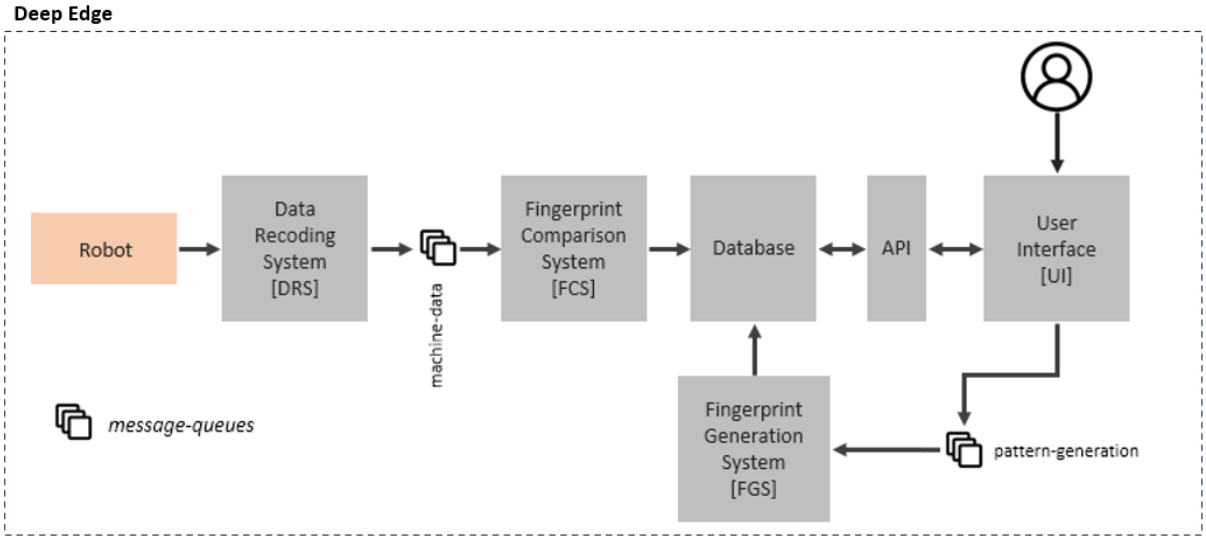

Figure 41: Current production workflow.

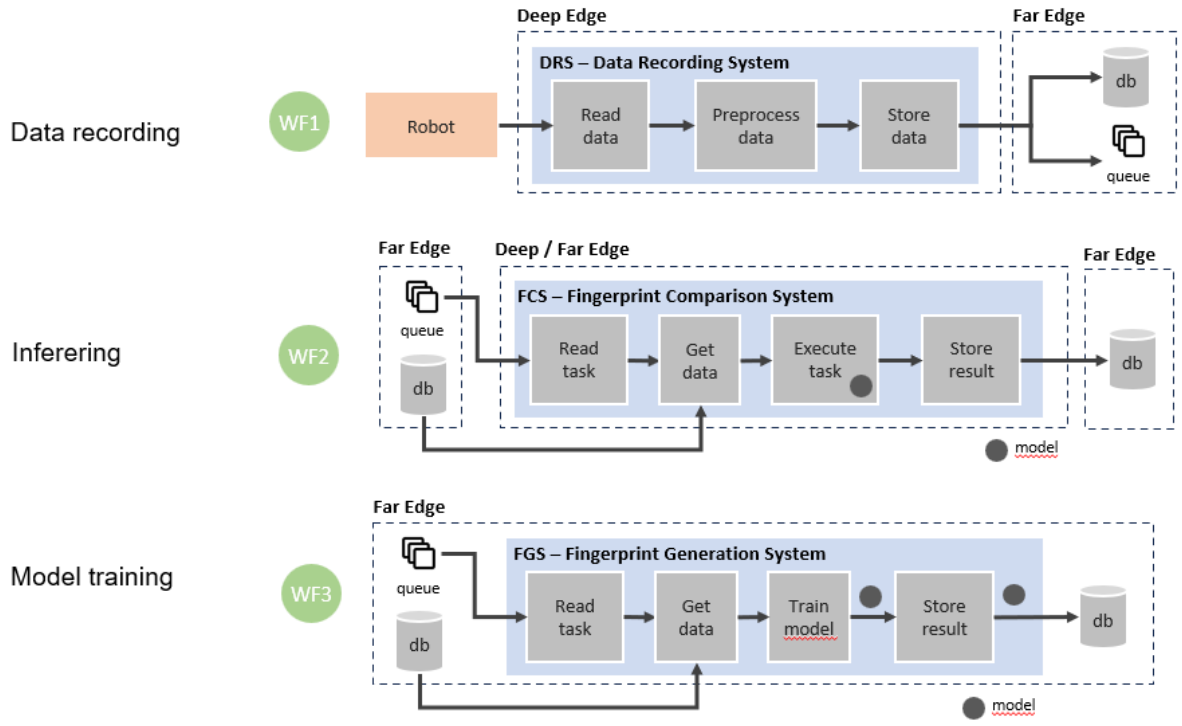

Figure 42: Possible breakdown in EMPYREAN of current behavior into three workflows

This incremental deployment strategy, beginning with a single robot, ensures a comprehensive validation and optimization process before scaling to scenarios involving multiple robots and associations. Development and deployment updates will systematically build upon previous steps, guaranteeing a smooth transition to a fully operational monitoring solution. By leveraging EMPYREAN's extensive feature set (including support for resilience, scalability, and operational efficiency) the deployment will address the demands of this highly distributed, on-premise use case effectively.

6.1.3 Leveraging EMPYREAN Components and Features

The deployment of this use case will leverage the EMPYREAN architecture to establish a robust and efficient system tailored to the requirements of advanced manufacturing operations. EMPYREAN's distributed components will play a pivotal role in addressing the unique characteristics of this scenario, including the fully on-premise setup (without Cloud interaction), the need for online, real-time operations, and the limited computational capacity at the deep edge. Processing will be divided across layers, with complex, high-computation tasks performed at the far and deep edge focused on data collection and minimal pre-processing before forwarding data for further analysis.

In this deployment, multiple Kubernetes clusters, potentially using different distributions to accommodate the constraints of lower compute power, will orchestrate containerized workloads across infrastructure layers. EMPYREAN's multi-cluster scheduling capabilities will be implemented to ensure seamless coordination and optimal resource utilization across the different edge layers.

The deep edge layer will employ Edge Smart Boxes equipped with 8GB of RAM. These devices will run lightweight Kubernetes distributions and host EMPYREAN components such as the Ryax Workflow Engine Worker and the Zeno-Flow daemon. These components will enable real-time dataflow management and facilitate communication between the deep and far edge layers. While some basic pre-processing may occur here, the primary function of the deep edge will be to collect sensor data and forward it to the far edge for advanced processing. EMPYREAN's telemetry service will be crucial in ensuring consistent monitoring and management of this distributed layer.

At the far edge, the deployment will rely on high-performance nodes equipped with 16 to 32GB of RAM, potentially complemented by GPUs for compute-intensive tasks such as training machine learning models for the Fingerprint Generation System. EMPYREAN's analytics-friendly distributed storage will ensure efficient data management during these operations, while the far edge will also host the majority of EMPYREAN's computational components, enabling scalable and advanced real-time analytics. This layer will act as the system's backbone, executing processes that demand high computational power and supporting complex real-time decision-making.

EMPYREAN's multi-clustering support within the Ryax Workflow Engine will be a key feature, enabling workload orchestration across different infrastructure layers. This will include optimized scheduling at both the local and multi-cluster levels to ensure efficient use of resources while maintaining strict real-time operational standards. The integration of the Dataflow programming framework within the Ryax Workflow Engine will enable real-time data communication and processing across the system, ensuring responsiveness to the high-frequency demands of this use case.

To ensure security and privacy in this highly distributed architecture, EMPYREAN's Privacy and Security Manager will be deployed to safeguard interactions between layers. Additionally, the application builder for unikernels will enable the deployment of lightweight binaries for IoT devices, allowing remote execution and simplified management at the deep-edge and on-premises layers. At the on-premises edge, lightweight devices such as Edge Smart Boxes with 2 to 4GB of RAM will handle the initial data collection from robotic sensors, running lightweight executables orchestrated by EMPYREAN's deep-edge microservices.

EMPYREAN's comprehensive feature set, including telemetry services, distributed storage, and real-time dataflow management will ensure that the architecture can scale to meet these challenges while maintaining efficiency and operational reliability.

6.2 Proximal Sensing in Agriculture Fields (UC2)

6.2.1 Overview

This use case focuses on the dynamic assessment of Soil Organic Carbon (SOC) to evaluate and manage soil conditions in agricultural fields. By combining proximal sensing technology with edge computing, the system enables real-time SOC assessment without relying on centralized data processing. This innovative approach supports integrative farm management and sustainable agricultural practices by providing timely and actionable insights for soil health.

The EMPYREAN platform revolutionizes soil health assessment workflows, enabling real-time Soil Organic Carbon (SOC) evaluation through advanced sensing and edge computing technologies. The system facilitates efficient and dynamic SOC analysis directly in the field, by integrating UAVs and robots equipped with cutting-edge sensors, including spectrometers and soil moisture devices. High-resolution data is processed locally using distributed AI and edge computing, ensuring accurate results with reduced latency, enhanced data privacy, and minimal data transfer. This approach supports sustainable agricultural practices by optimizing resource use, increasing yields, and minimizing environmental impacts. Through a decision support platform, farmers receive actionable insights and strategies for SOC improvement, empowering informed, adaptive farm management.

The use case begins with a drone capturing high-resolution multispectral images of the field, forming the foundation of the SOC assessment. These images are processed through advanced workflows, including image stitching and the creation of management zones, which enable precise SOC predictions either locally or on remote infrastructure, depending on resource availability. Secondly, a robot performs complementary ground operations, focusing on detailed SOC analysis at specific points of interest. Equipped with a portable spectrometer and a moisture sensor, it collects precise soil data to generate a comprehensive SOC map. This map provides a granular view of soil health, guiding key decisions such as fertilization, irrigation, and weed management. At the heart of the system lies a decision support platform that integrates all data from drones, robots, optional satellite imagery, and farmer inputs. Through a user-friendly interface, the platform offers actionable recommendations and SOC improvement strategies. This seamless integration of technologies empowers farmers with real-time monitoring and informed decision-making, transforming agricultural practices to prioritize both productivity and sustainability. One or more Associations will be set up for the realization of the use case.

6.2.2 Development and Deployment Updates

This section provides further details on the development of the use case. A preliminary selection of the edge hardware has been made. Additionally, some adjustments have been made to the initial plans described in D2.1, specifically regarding the overall development of the use case. Lastly, the different workflows within the use case are described, as well as the

decision support tool that will be presented to the farmer and their advisors to facilitate informed decision-making.

6.2.2.1 Devices, equipment and communications

Device	Component	Key features	Communication
UAV	DJI M350 RTK drone	- RTK GPS - 5880 mAh battery: flight time up to 55 minutes - Max speed 23m/s - IP55 - Power: 44.76V/5.88A	Wi-Fi 6 Bluetooth 5.1
	Raspberry Pi 5	- CPU: 2.4GHz 64-bit Arm Cortex-A76 - RAM: 8GB - Memory: SD card - Power: 5V/5A	Wi-Fi 5 Bluetooth 5.0
	Hailo 8L AI accelerator	- 13 TOPS - Power: 1.5W	Not applicable
	Micasense RedEdge Multispectral camera	- 10 multispectral bands - 1.6mp/band - Up to 3fps - Memory: SD card - Power: 7V/2.85A	Wi-Fi
Robot	ILVO Cimat robot	- RTK GPS - IMU, camera, lidar, sonar - GPU - FPGA	Wi-Fi 4G
	Spectrometer	To be selected	
	Moisture sensor	To be selected	

Table 31: The different hardware components and their key features of UC2.

6.2.2.2 Adjustments from D2.1

This list contains some alterations and updates to the initial planning provided in D2.1.

- ***Cover crops***: Initially, we planned to use RGB and multispectral imaging to estimate cover crop biomass and identify management zones. However, this plan has been revised to focus on a different model that assesses Soil Organic Carbon (SOC). The new model utilizes multispectral images of fields with bare soil to perform the SOC assessment, providing valuable insights into soil health and composition.
- ***Decision support***: After discussions with stakeholders, we have decided to broaden the scope of the use case. Rather than focusing solely on providing farmers with a prescription map based on SOC evaluations, we aim to develop a more complete farmer advisor platform. This platform will enable farmers and their advisors to gain valuable insights about the land using a variety of data sources, including SOC management zones, yield maps, and NDVI/NDWI data from satellite imaging, among others. By integrating these diverse data sources, the platform will guide users through a decision support tool to make well-informed management decisions. Outputs may include the aforementioned SOC-based prescription maps, as well as recommendations for fertilization, weed control, irrigation activities, and more.
- ***Platform output***: The original plan for the final stage of the use case involved guiding fertilization through prescription maps generated from SOC assessments and other collected data. To accommodate this, the last step of the use case will involve the generation and download of the prescription map in an industry-standard format. The farmer can then upload this file to the proprietary tractor's system to carry out the fertilization process.

6.2.2.3 AI models training/development

The development of AI models for this use case focuses on two distinct but complementary approaches to assessing SOC levels, leveraging data from the drone-mounted multispectral sensor and the robot's portable spectrometer. These models ensure a comprehensive and dynamic assessment of SOC across the agricultural field.

The SOC Classification Model (Drone) will use multispectral images captured by the UAVs camera during field surveys, which provide ten spectral bands with high spatial resolution. Its objective is to classify SOC levels across the field into categories, such as low, medium, and high, and generate management zones based on these classifications. The training process involves collecting soil samples from ILVO fields that represent diverse conditions and SOC levels. These samples will be analysed in the laboratory to determine their precise SOC content. The resulting SOC data is then paired with the corresponding multispectral images to create the training dataset. Once trained, the classification model will be deployed on edge devices, allowing in-field SOC level predictions during drone flights. These outputs guide subsequent workflows, including identifying points of interest for detailed analysis by the robot.

The SOC Prediction Model (Robot) relies on data from a portable spectrometer and a soil moisture sensor mounted on the robot. These devices capture precise measurements at specific points of interest. The model's objective is to provide highly accurate SOC values for particular locations within the field, offering granular insights and serving as a benchmark for validating and refining the classification model. For training, soil samples will be collected from ILVO fields and undergo detailed spectrometer analysis and laboratory testing to determine their SOC levels and other relevant properties. The spectrometer readings are correlated with the lab-analysed SOC values to train the model, with environmental factors such as soil moisture levels included to ensure robustness under varying field conditions. After training, the prediction model will operate on the robot's computing infrastructure at the deep edge, enabling real-time SOC predictions during field operations.

6.2.2.4 Workflows identified within the use case

The workflows for this use case are categorized into two groups: those executed on the drone and those executed on the robot. This section provides a detailed overview of these workflows, describing their inputs, processing steps, and outputs. By outlining these workflows, we aim to highlight how the various components interact to achieve the objectives of the use case. Additionally, some information is given regarding a decision support tool, which can be used to make informed farming decisions based on data acquired in the aforementioned workflows and other data sources.

Drone Workflows:

- ***1st workflow [DroneWF1] - Multispectral image collection***: During flight, the UAV collects data using a multispectral camera, capturing ten distinct spectral bands. Data from the relevant bands is compressed and stored, either locally or on remote storage. This workflow is deployed on the drone's computing infrastructure, allowing for real-time processing and efficient collection of multispectral images at the deep edge. By performing these tasks locally on the drone, the workflow minimizes latency and optimizes the data pipeline for further analysis.
- ***2nd workflow [DroneWF2] - Monitoring of drone characteristics*:** Throughout the flight, the UAV provides detailed performance metrics about its operation, processing unit, and multispectral sensor. These include flight data such as speed, altitude, and position, as well as battery capacity, CPU/memory usage of its computing unit, and more. This workflow could for example track the battery consumption, providing different soft and hard thresholds and triggering alerts when necessary. Alerts may need human intervention or trigger other specific actions, such as stopping data collection or offloading computation to nearby computing units.
- ***3rd workflow [DroneWF3] - Multispectral image stitching***: In the DroneWF1 workflow, the UAV captures multispectral images of the field, following a flight path designed with specific image overlap in mind. This overlap is crucial for stitching the images from different spectral bands into a single, multi-layered composite image that provides a comprehensive overview of the entire field. Given the potential size of the dataset,

which depends on the field's dimensions, this process is computationally intensive and demands significant resources for effective processing.

- ***4th workflow [DroneWF4] - SOC assessment and creation of management zones based on SOC assessment:*** The multi-layered stitched image serves as input for evaluating SOC levels across the field. This evaluation is performed using a pre-trained AI model, which is periodically updated with more precise data collected by the robot. The SOC levels are then classified into management zones, categorized into different levels (e.g., low, medium, high). These management zones represent the key output of the UAV flight and provide critical insights for informed decision-making in field management.

Robot Workflows:

- ***1st workflow [RobotWF1] - Identification of points of interest for detailed SOC assessment*:** Based on the management zone map, which is the primary output of the UAV flight, specific points of interest are selected for more detailed analysis. These points are primarily chosen to calibrate the SOC measurements captured by the multispectral camera and to enhance the accuracy of the AI model. This targeted approach ensures continuous improvement of the model and more precise SOC evaluations in future workflows.
- ***2nd workflow [RobotWF2] - Data collection*:** The ILVO Cimat robot conducts complementary ground operations, focusing on detailed SOC analysis at designated points of interest. Equipped with a portable spectrometer and a moisture sensor, the robot gathers highly precise soil data. This data will feed into subsequent workflows, contributing to the generation of a comprehensive and accurate SOC map.
- ***3rd workflow [RobotWF3] - Monitoring of robot characteristics*:** Similar to the UAV, the robot provided data that allows for monitoring its performance through metrics, such as position, speed, battery status, and the resource usage of its onboard computer. These data points help ensure the efficient operation of the robot during its tasks.
- ***4th workflow [RobotWF4] - Creation of SOC map of the field*:** This workflow will be deployed on the robots computing infrastructure, using data from the robot's spectrometer and moisture-related sensors at the deep edge. The prediction phase will primarily take place at the far edge, although lightweight ML models could be executed at the deep edge. The processed data will then be stored at the far edge, and the soil organic carbon assessments will be used as inputs for creating prescription maps. These maps provide a granular view of soil health, guiding key decisions such as fertilization, irrigation, and weed management.

Decision Support Platform: At the heart of the system lies a decision support platform that integrates all data from drones, robots, optional satellite imagery, and farmer inputs. Through a user-friendly interface, the platform offers actionable recommendations and SOC improvement strategies. The platform allows uploading data sources, in addition to the SOC maps, such as satellite images, yield maps, index, and others.

6.2.3 Leveraging EMPYREAN Components and Features

With the implementation of the EMPYREAN platform, the future state of soil health assessment will see a significant transformation in workflows. Integrating various sensors and data types, combined with edge computing, will enable dynamic and efficient SOC assessment, allowing for near real-time analysis of soil conditions. This will create a more responsive and adaptive agricultural management system. In the future state envisioned with EMPYREAN, UAVs equipped with advanced sensors will play a crucial role. Robotic sensors will further enhance soil assessment by performing detailed analyzes in the identified management zones. The robot, equipped with visible and near-infrared (Vis-NIR) spectrometers, RTK GPS, and soil moisture sensors, will conduct dynamic and efficient SOC assessments directly in the field. The data collected will be processed on-site using edge computing, ensuring timely and accurate results.

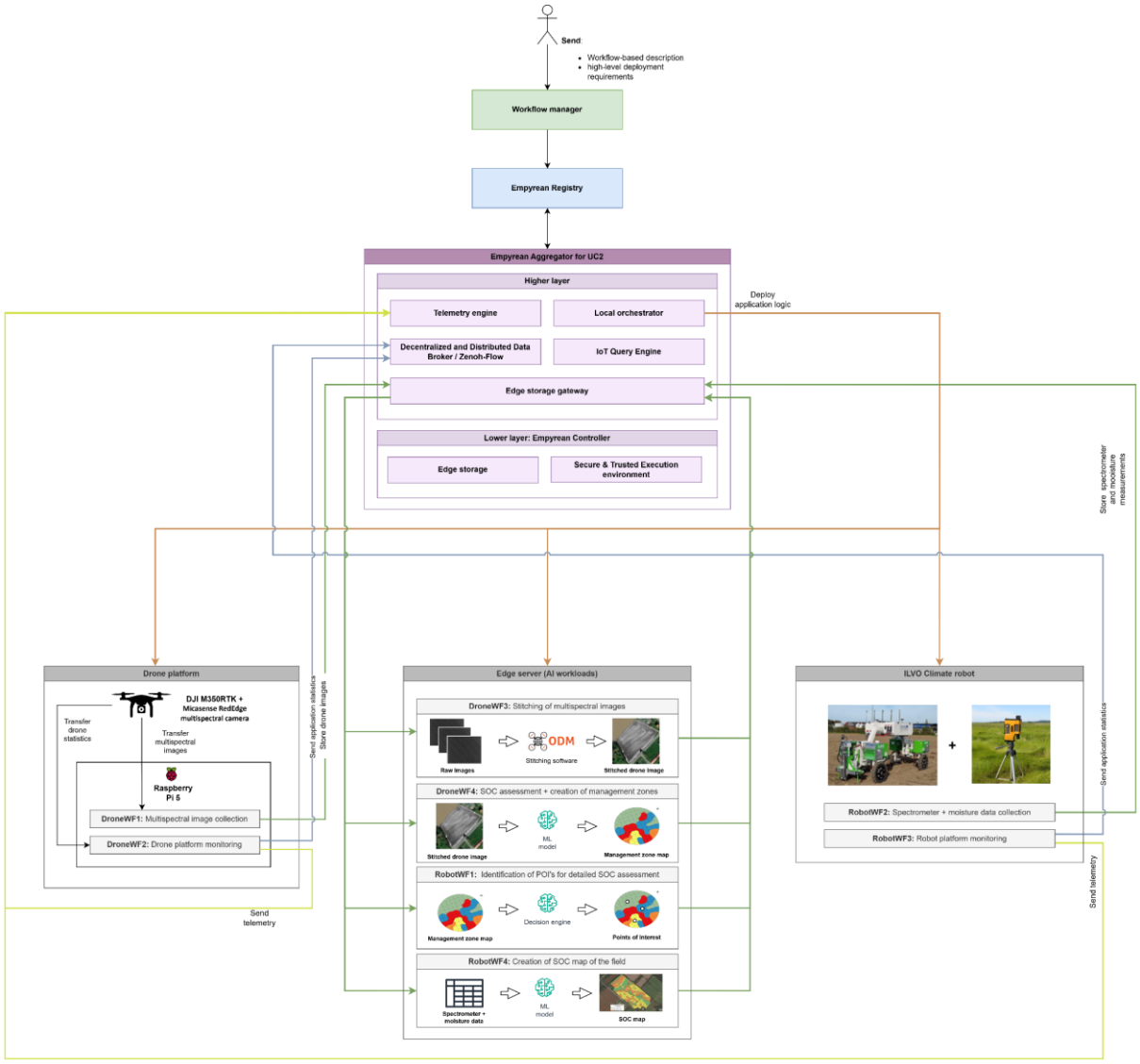

Figure 43: Planned UC2 architecture with the EMPYREAN components

By leveraging the EMPYREAN components, we can efficiently achieve the targets outlined in this use case (Figure 43). Below is a list of the key components and features that will be utilized during the use case development. Please note that this list is not exhaustive and additional components, such as the Privacy and Security Manager, the Lightweight Application Packaging, the Application Builder for Unikernels, etc are going to be included indirectly.

Workflow Manager: The end result of the use case is a decision support platform that allows the farmer (advisors) to make informed decisions on the farm management based on data presented to him. For this, the large amounts of raw data collected by the UAV, robot, satellites, etc should be analyzed and converted to a format that can be interpreted by the users. For this purpose, the EMPYREAN Workflow Manager will be used as it enables efficient design and execution of data analytics applications. In section 6.2.2 we have identified a number of workflows that could potentially be developed in the context of a Ryax workflow, such as parsing of the multispectral drone images or parsing of the spectrometer/moisture measurements.

Dataflow Programming: The Dataflow Programming component will play a key role in defining and managing the various data flows within the system. This includes data exchanges between edge devices, such as UAVs and the Cimat robot, the edge server, and the cloud infrastructure. By orchestrating these data flows, the component ensures that raw data collected at the edge is efficiently processed, transferred, and made available in the cloud for visualisation to the end user.

Edge Storage (Gateway): Both the multispectral sensor and the spectrometer generate a high volume of data during operation. Initially, most of this data is stored on edge storage, so it can be easily processed on the edge. This limits the amount of data to be transferred to the cloud. Once the analysis is performed, the resulting output data from the workflows is transferred to the cloud storage. This processed data can be visualized and presented to the user in a clear and accessible format, ensuring that insights are readily available for decision-making purposes.

Decentralized and Distributed Data Manager: The use case relies on the data manager as the primary mechanism for data exchange between edge devices and the cloud infrastructure. This layer facilitates the efficient transfer of data collected from edge devices. Additionally, the communication layer can trigger specific workflows to execute within the Workflow Manager, enabling automated data processing and analysis.

Telemetry Engine: The edge devices provide a range of parameters that can be monitored to ensure its optimal operation. For example, the UAV collects detailed flight data, including speed, altitude, and position, as well as information about battery capacity and the CPU/memory usage of its computing unit (Raspberry Pi). Similarly, the robot offers comparable metrics, such as position, speed, battery status, and the resource usage of its onboard computer.

6.3 Advanced Inference and Coordinated Behaviors for Warehouse Automation Robots (UC3)

6.3.1 Overview

The particular use case centers around a robotic warehouse automation application, where a fleet of Autonomous Towing Robots (ATRs) performs order-picking based on incoming orders. Warehouse operators utilize a specialized software named the Fleet Control System (FCS), which enables them to submit order-picking tasks to the ATRs. In this use case, the ATRs will be equipped with on-board computational capabilities along with a variety of sensors (such as lidars and radars), enabling them to perform the needed computations directly upon the robots at the deep edge. The Fleet Control System is installed and operates mainly at the far edge, where more substantial computational resources are available, including GPU node pools that can offload ML inference tasks and potentially perform ML training. Additionally, public or private cloud resources can be also used to offload even more compute-intensive operations.

A key challenge of this use case is the collaboration and data exchange between robots, which besides the adaptations needed on the Fleet Control System it requires efficient data transfer among robots and with the FCS. This setup also demands increased computational power across the different layers of the continuum. Moreover, the possible intermittent connectivity of the ATRs within the warehouse needs also to be taken into account. To address these needs, the use case leverages novel functionalities from the EMPYREAN platform. EMPYREAN's ability to enable seamless executions on different layers of the edge-cloud computing infrastructure to facilitate the operation of the robots, while ensuring secure data transfers even under intermittent networking connectivity, will significantly enhance the robot fleet's operation.

6.3.2 Development and Deployment updates

This section provides further details on the development of the use case.

6.3.2.1 Devices, equipment and communications

The initial computing infrastructure setup for this use case will include: a) the ATR robots equipped with onboard industrial PC units featuring Intel i5 CPUs and 16GB of RAM, representing the deep-edge part. The robots will have various onboard sensors (such as lidars and radars) to facilitate navigation and order-picking within the warehouse; b) intermediate servers located in the warehouse office, consisting of PC's, laptops, or GPU-equipped units with around 32GB of RAM, representing the far-edge; and optionally c) a distant data-center or cloud resources for offloading more demanding computations, offering more powerful computation resources and diverse hardware sources along with access to GPU node pools.

6.3.2.2 Workflows identified within the use case

At this initial stage of the project, the following use case workflows are anticipated:

- ***1st workflow***: The FCS is connected to the EMPYREAN platform, and when a certain order is submitted on the FCS, the workflow demands the collection of data at the deep edge on the computing cluster of the ATR while performing an initial pre-processing onboard and transferring data to the far edge for further data treatment while enabling storage of insights and results on the FCS triggering specific alerts or observability functions. This workflow can be duplicated and used for each different ATR.
- ***2nd workflow***: The ATR retrieves data, and depending on the task's computational demands, it may offload the more compute-intensive tasks to the far edge (or the cloud) featuring a GPU. The processed data is then transferred back to the ATR to use it accordingly.
- ***3rd workflow***: The FCS operator submits a request for robot collaboration between two ATR robots, such as jointly picking up a particular (heavier) cart. Based on that, the two robots will receive the operation order, and data will be transferred to both for pre-processing. This particular pre-processing process will be started on both robots simultaneously. This may be managed by a single workflow executing across both robots and waiting to aggregate insights from both or by two workflows coordinating with each other.

6.3.2.3 Developments

The Tractonomy development team has been preparing the core robot technology for the above integrations with the EMPYREAN platform. Tractonomy's autonomous towing robot (ATR) is a commercially ready robot designed for towing all sorts of existing carts. It has a unique omnidirectional platform with a rotating gripping system for grabbing and pulling all sorts of industrial carts. A number of areas had to be prepared within the hardware architecture of the existing platform. A demo robot is used as a development platform and showcase for the EMPYREAN project.

Distributed Zenoh Databases: To support the first workflow, the team has updated the robot to the latest Zenoh 1.0.0 "Firesong" release. We have been experimenting with how data-in-motion and data-at-rest geo-distributed storage can work. As there exist many ways for Zenoh nodes to store values it may need to serve later, we have integrated the storage manager plugin relying on dynamically loaded "backends" to provide this functionality. Typically, a backend will leverage some third-party technology, such as databases, to handle storage. A possibly convenient side effect of using databases as backends is that they may also be used as an interface between the robot's Zenoh infrastructure and the EMPYREAN infrastructure and may interact independently with the database.

Point cloud compression: A key enabler for the second workflow is to stream real-time point clouds from the on-board Intel Realsense D455 depth camera to the far edge where machine learning and computer vision (CV) pipelines can process and return inferences from the data the robot sees (Figure 44). In this case, we developed a containerized CV pipeline that can be dynamically configured and deployed by the RYAX engine based on the relevant associations for GPU resources. Currently, the latency has been tested on local loopback networks without point-cloud compression at 1Hz. The next goal is to implement MPEG-PCC and evaluate the benefits of point cloud compression in 5 GHz wireless networks, which are representative of Tractonomy's production deployment environments.

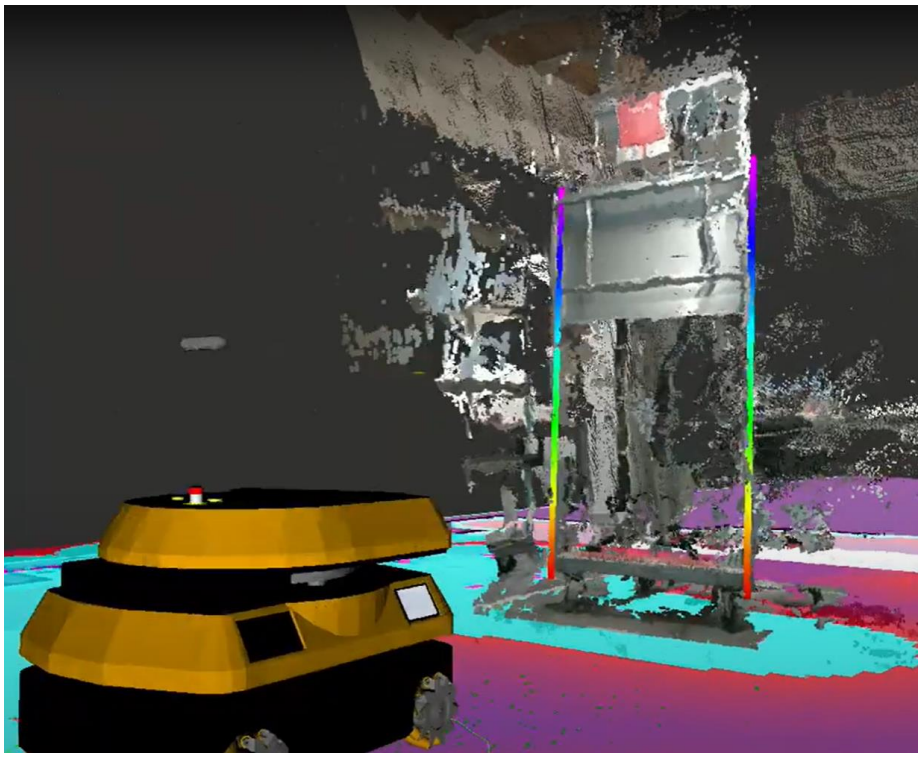

Figure 44: A Tractonomy's autonomous towing robot (ATR) collecting point cloud data.

Automatic cart locking system: A key concept is the ATR's unique cart docking system. The cart docking system is a rotating turret enabled by a patented gripping system. The turret allows Tractonomy to handle two types of popular carts:

1. Carts with fixed wheels in front and castor wheels in the rear.
2. Carts with castor wheels on all sides.

This property is already being exploited in commercial applications. We are now developing a prototype to extend this property to manipulate very long carts with castor wheels on all sides using two robots (Figure 45). Initially, a shared motion model has been partially developed in simulation to establish a smooth and seamless motion control demonstrator with two robots in simulation. A realization of a final demonstrator with two physical robots will also be evaluated.

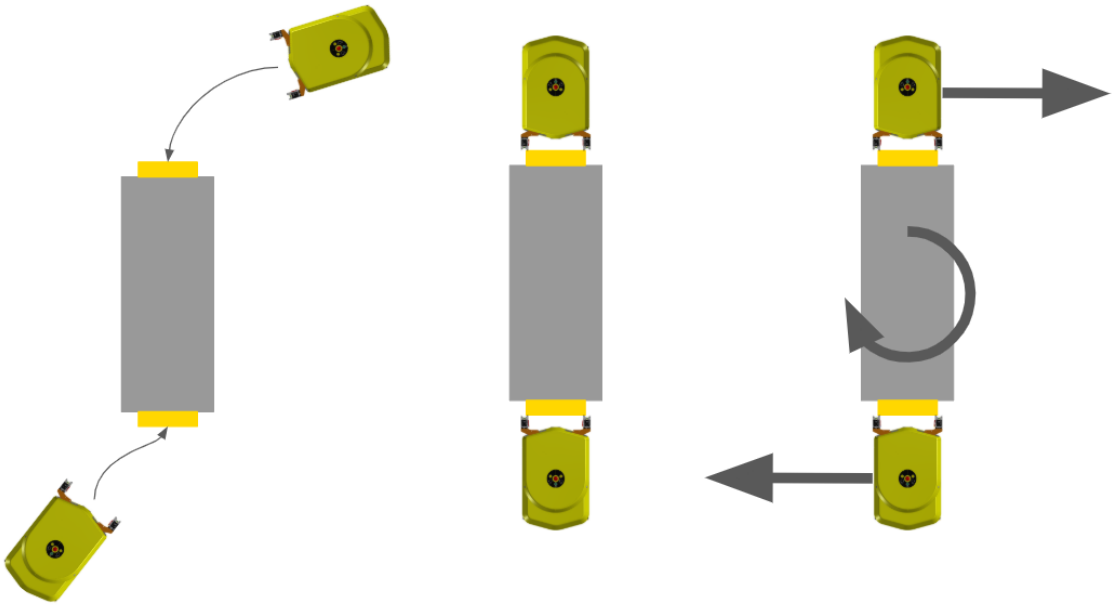

Figure 45: A two carts automatic locking system.

6.3.3 Leveraging EMPYREAN Components and Features

Based on this infrastructure setup, one EMPYREAN Association and Aggregator will be explored initially to enable the execution of tasks from deep to far edge up to the cloud. Each ATR robot will be an independent cluster in the Association, possibly using lightweight Kubernetes distributions such as K3S. Initially, one robot will participate in the Association, while afterwards, the participation of multiple robots will be realized. By the end of the project, we will validate the functionality of defining multiple Associations for different aspects where robots may participate in more than one Association.

Critical EMPYREAN features for the successful execution of this UC include: (i) the integration of the dataflow programming (Zenoh-Flow) within the workflow management system (Ryax), (ii) the cyber-threat intelligence engine and the privacy and security manager to guarantee that operation will remain secure even in highly vulnerable contexts (such in 4G networks), (iii) the support of unikernels for lightweight, secure, and reproducible deployment at the edge, (iv) the support for intermittent connectivity, (v) the edge distributed storage, (vi) the multi-clustering execution, and (vii) optimized offloading for GPU-based ML inference.

6.4 Security in Smart Factories - S. Korea International Collaboration (UC4)

6.4.1 Overview

Smart factories are becoming more prevalent, and their operations rely heavily on network connectivity. Private 5G networks are being used to provide the high-speed, low-latency connectivity required by smart factories. However, as with any network, security is a critical concern. This use case considers a situation-aware security orchestration model that can effectively address security threats to smart factories that use private 5G networks. Towards this end, situation-aware security and other applications are required, including but not limited to Intrusion Detection Systems (IDS) and Firewall and Cyber Thread Intelligence (CTI) modules. These provide an effective means of ensuring the security of the factory's operations/services and physical assets.

This UC focuses on addressing these challenges by automating the deployment of security applications that run inside smart factories Association(s). EMPYREAN's advanced orchestration and autoscaling mechanisms will ensure that the applications are executed efficiently under any circumstances while utilizing both edge resources (inside and between Association) and cloud resources. EMPYREAN's employed security and trust functionalities will ensure that the security application operates securely. The utilized AI-based procedures for accurately recognizing and responding to security threats will take advantage of EMPYREAN's privacy-preserving Federated Learning (PPFL) mechanisms that support a fully privacy-preserving and federated anomaly detection system.

6.4.2 Development and Deployment Updates

As indicated in D2.1, this use case was agreed originally with Prof. Ilsun You of the Department of Information Security, Cryptology, and Mathematics, Kookmin University, Seoul, Korea under the conditions of being funded by the South Korean government. As it was not the case, it was decided to continue the collaboration in various ways. In this context, a joint workshop was organized linked to the special session Secure and Cognitive Continuum (SECON) that was held at the 8th International Conference on Mobile Internet Security (MobiSec 2024) Sapporo, Japan, on December 17, 2024 (https://manuscriptlink-society-file.s3.ap-northeast-1.amazonaws.com/kiisc/conference/mobisec2024/secon24-2.htm).

During this workshop, the EMPYREAN architecture and the different modules envisioned were discussed among EMPYREAN and Korean partners. The Korean partners expressed their interest in the work done and the possibility of collaboration, taking however into account their funding limitations. It was decided to continue collaboration during 2025 (Y2) so as to evaluate the use of key EMPYREAN components that could fit the needs of the Korean partner threat management scenarios. Also, the University of Murcia's 5G testbed can be used to replicate these scenarios as an initial Proof of Concept (PoC), demonstrating EMPYREAN's

capabilities. Towards this direction, another workshop was scheduled for Q2 of 2025 to provide concrete planning based on a proposal provided by EMPYREAN on the components ready to be tested during Y2.

6.4.3 Leveraging EMPYREAN Components and Features

Based on the above discussion, the focus of the use case will be centred around three major achievements and features:

- Orchestration and deployment of secure enablers at the edge level to provide threat identification and abnormal behaviours based on privacy-preserving Federated Learning (PPFL).
- Use the telemetry component to aggregate information coming from the PPFL to trigger events in the CTI components.
- Integrate the CTI component either to:
    - identify if possible identified threats related to the existing components deployed in order to deploy the most suitable solution for the PPFL;
    - or to provide support to detect the misbehaviours identified and if needed to exchange information with the privacy preserving MISP component to share this situation awareness with the rest of the nodes.

7 Implementation and Delivery Plan

The overall implementation and delivery of the EMPYREAN project (Figure 46) are structured into a series of well-defined and complementary phases, ensuring a systematic and iterative approach to achieving the project's objectives. These phases enable seamless progression from requirement analysis to the delivery of a fully functional platform, incorporating feedback and refinements at every stage. Deliverable D2.2 (M7) provides a more detailed description of these plans and phases.

The present deliverable marks the successful completion of the first two phases: Requirements Analysis (Phase 1) and Architecture Definition (Phase 2). Moving into the second year, the consortium's focus will shift towards the research and development activities of the individual technological advancements (Phase 3). Initial outcomes from these activities will be reported in M15 through the first set of technical deliverables from WP3 and WP4. Simultaneously, WP5 will commence activities to support the continuous integration and testing activities (Phase 4), preparing the initial platform release by M18. Moreover, efforts will also focus on initiating the implementation of the project use cases.

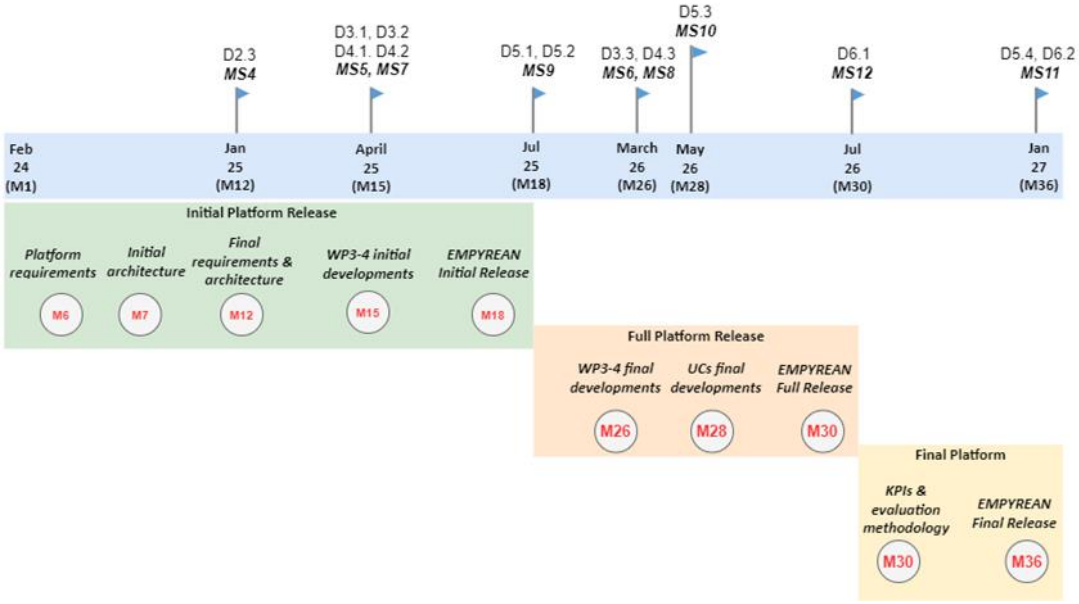

Figure 46: EMPYREAN development roadmap

The initial platform release, resulting from the first project development iteration (M4-M15), will provide a partial implementation of EMPYREAN components. This version will feature a subset of the envisioned features and primary inter-component communication interfaces, forming a functional prototype to support the core platform's objectives. This prototype will offer critical feedback to guide and enhance the second development iteration (M18–M36) that will provide the full platform release (M30) and final platform release (M36).

By adopting this structured and iterative approach, the EMPYREAN consortium ensures a systematic transition from concept to a fully functional and exploitable platform, aligning technological developments with the project's objectives and stakeholder expectations.

8 Conclusions

This deliverable presents the final version of the EMPYREAN architecture, emphasizing the critical technical aspects of component interactions and system operation flows. The comprehensive analysis of how components interact within the IoT-edge-cloud continuum has been pivotal in identifying and addressing key operational challenges. These interactions are designed to enable seamless communication, efficient task distribution, and dynamic resource allocation across Associations, ensuring the platform's ability to support diverse and time-critical applications.

System operation flows have been meticulously designed to align with the needs of various users and application scenarios. These flows outline the procedural dynamics of how tasks are initiated, managed, and executed across the platform, providing clarity on the operational alignment of system components. The iterative refinement of these flows informed necessary adjustments to the architecture, ensuring it accurately reflects EMPYREAN operational demands, particularly related to its use cases, while optimizing platform efficiency.

As a result, the finalized EMPYREAN architecture integrates these technical insights to serve as the basis for measuring and achieving the project's technical KPIs. By incorporating detailed component interactions and well-defined system operation flows, the architecture sets a robust foundation for tracking the platform's performance and effectiveness throughout its lifecycle.

Furthermore, the deliverable provides key updates on the use cases, illustrating how they will leverage EMPYREAN's new features to enhance their functionality. Additionally, the implementation and delivery plan, along with the requirements coverage analysis provide more context on the platform's development roadmap and its alignment with project objectives.

Finally, this document will also serve as a guideline during the iterative execution of the implementation and evaluation phases supporting technical activities in WPs 3-4, UCs development and platform integration activities in WP5, and project demonstrations in WP6. By ensuring that technological developments remain relevant and aligned with EMPYREAN's ambitions, this deliverable reinforces the project's vision of introducing a novel ecosystem of trustworthy, cognitive, and AI-driven collaborative Associations of IoT devices and edge resources for intelligent data processing.

9 Appendix - Requirements Coverage

This section outlines how the final set of functional requirements, collected, described, and categorized within WP2, are addressed by the components of the EMPYREAN architecture and the presented operation flows. Deliverable D2.2 (M7) introduced the initial mapping of requirements in the context of the preliminary architecture release. This deliverable builds upon that by expanding the initial mapping to include updates, detailing the operation flows that implement these requirements, and providing an overall analysis of non-functional requirements.

Throughout the architecture design process, all partners closely collaborated to analyze each requirement and map it to the platform's components. This iterative and cooperative interaction between the technical and use case partners ensured that the EMPYREAN platform is capable of realizing the desired functionalities, aligning the architecture with both the project's objectives and the specific needs of its use cases.

Table 32: Functional requirements coverage in the final EMPYREAN architecture and operation flows

ID	Short Description	Components	Operation Flows
F_GR.1	Federate heterogeneous and distributed IoT, edge, and cloud resources.	EMPYREAN Aggregator, EMPYREAN Registry, Service Orchestrator, Privacy and Security Manager, Telemetry Engine, EMPYREAN Controller, Decentralized and Distributed Data Manager, Edge Storage Gateway	OF1.1, OF1.2, OF1.3, OF2.1, OF2.2, OF2.3, OF2.4, OF3.3, OF4.1, OF4.3, OF4.7, OF5.1
F_GR.2	Enable collaborative autonomy in the IoT-edge-cloud continuum.	EMPYREAN Aggregator, EMPYREAN Registry, Service Orchestrator, Decision Engine, Telemetry Engine, Decentralized and Distributed Data Manager, Privacy and Security Manager	OF1.1, OF1.2, OF1.3, OF4.1, OF4.8, OF5.1, OF6.1
F_GR.3	Encompass autonomous and continuous control loops.	Service Orchestrator, Decision Engine, Analytics Engine, Telemetry Engine, Persistent Monitoring Data Storage, EMPYREAN Controller	OF1.3, OF4.1, OF4.2, OF5.1, OF6.1, OF6.2
F_GR.4	Provide seamless deployment of hyper-distributed cloud-native applications across a collaborative IoT-edge-cloud continuum.	Workflow Manager, Dataflow Programming Component, Service Orchestrator, NIX-based Environment Packaging, Application Packaging, Container Runtime, Edge Storage Gateway, Decentralized and Distributed Data Manager	OF1.1, OF1.2, OF1.3, OF2.3, OF3.1, OF3.2, OF4.1, OF4.2, OF4.3, OF4.4, OF4.5, OF4.6, OF4.8, OF5.1
F_GR.5	Support hyper-distributed, highly-demanding, and dynamic applications from diverse domains.	Workflow Manager, Dataflow Programming Component, Software-Defined Edge Interconnect, Decentralized & Distributed Data Manager, Workload Autoscaling, Hardware Acceleration Abstractions, Application Packaging	OF1.1, OF1.2, OF1.3, OF3.1, OF3.2, OF4.1, OF4.3, OF4.4, OF4.5, OF4.6, OF4.7, OF5.1
F_GR.6	Provide monitoring for cloud-native applications and heterogeneous infrastructure resources.	Monitoring Probes, Telemetry Engine, Persistent Monitoring Data Storage, Container Runtime	OF1.3, OF2.1, OF2.2, OF4.1, OF4.2, OF4.7, OF4.8, OF5.1, OF6.1, OF6.2

F_GR.7	Energy and power-aware operation for optimal power management, energy efficiency and ecological sustainability.	Monitoring Probes, Telemetry Engine, Decision Engine, Workload Autoscaling, Container Layers Locality Scheduler	OF4.2, OF4.8, OF5.1, OF6.1
F_ASSOC.1	Combine heterogeneous computational and storage resources and different connectivity resources.	EMPYREAN Registry, EMPYREAN Aggregator, EMPYREAN Controller, Edge Storage Gateway, Decentralized and Distributed Data Manager	OF1.1, OF1.2, OF1.3, OF2.1, OF2.2, OF2.3, OF3.3, OF4.1, OF4.2, OF4.3, OF4.5, OF4.7, OF5.1
F_ASSOC.2	Facilitate secure onboarding of IoT devices, robots, and edge/cloud resources within the EMPYREAN control and management plane.	EMPYREAN Registry, Secure and Trusted Execution Environment, Privacy and Security Manager	OF1.3, OF2.1, OF2.2, OF2.3, OF2.4, OF3.3
F_ASSOC.3	Constitute a secure and trustworthy execution environment.	Privacy and Security Manager, CTI Engine, Secure and Trusted Execution Environment, Edge Storage Gateway	OF1.3, OF2.4, OF3.3, OF4.4, OF5.1, OF6.3
F_ASSOC.4	Support autonomous operation and enhance resiliency across the continuum.	Workflow Manager, Analytics Engine, Decentralized and Distributed Data Manager, Decision Engine, Workload Autoscaling, EMPYREAN Controller, Software-Defined Edge Interconnect	OF1.3, OF4.2, OF4.8, OF5.1, OF6.1, OF6.2, OF6.3
F_ASSOC.5	Provide low and predictable latency for hyper-distributed applications.	EMPYREAN Aggregator, Software-Defined Edge Interconnect, Decentralized and Distributed Data Manager	OF4.1, OF4.3, OF4.5, OF6.1
F_ASSOC.6	Provide inter-Association communication and exchange of events.	EMPYREAN Aggregator, Decentralized and Distributed Data Manager	OF1.3, OF4.3, OF5.1
F_ASSOC.7	Data-driven seamless workload and data migration across the Associations.	Telemetry Engine, Monitoring Probes, Analytics Engine, Service Orchestrator, Edge Storage Gateway	OF1.3, OF2.3, OF3.3, OF5.1, OF6.1
F_ASSOC.8	Aggregators must maintain a catalogue of the Association resources.	EMPYREAN Registry, EMPYREAN Controller, Telemetry Engine	OF1.1, OF1.2, OF1.3, OF2.1, OF2.2, OF2.3, OF3.3, OF4.1, OF4.2, OF5.1
F_ASSOC.9	Aggregators must dynamically discover resources within the registered infrastructures and detect events.	EMPYREAN Registry, EMPYREAN Aggregator, EMPYREAN Controller Telemetry Engine, Monitoring Probes	OF2.1, OF2.2, OF2.3, OF4.1, OF5.1, OF6.1
F_ASSOC.10	Aggregators must maintain the state of the Association.	EMPYREAN Registry, EMPYREAN Aggregator, Telemetry Engine, Persistent Monitoring Data Storage	OF1.1, OF1.2, OF1.3, OF2.3, OF4.1, OF4.2, OF5.1
F_ST.1	Decentralized identity management.	p-ABC Library, Privacy and Security Manager	OF1.1, OF1.2, OF1.3, OF2.1, OF2.2, OF2.4, OF4.1, OF4.2, OF4.3, OF5.1
F_ST.2	Privacy-Preserving authentication and authorization.	p-ABC Library, Privacy and Security Manager, Secure and Trusted Execution Environment	OF1.1, OF1.2, OF1.3, OF2.1, OF2.2, OF2.4,

			OF3.3, OF4.1, OF4.2, OF4.3, OF5.1
F_ST.3	Policy-Based Encryption.	p-ABC Library, Privacy and Security Manager	OF3.3, OF4.1, OF4.3
F_ST.4	Automated Cyber Threat Analysis.	CTI Engine, Telemetry Engine, Persistent Monitoring Data Storage	OF6.2, OF6.3
F_ST.5	ML for Anomaly Detection and Cybersecurity.	CTI Engine, Telemetry Engine, Persistent Monitoring Data Storage	OF4.2, OF5.1, OF6.2, OF6.3
F_ST.6	Secure and Trusted Execution.	Secure and Trusted Execution Environment, Privacy and Security Manager, Container Runtime, Unikernel Deployment	OF4.1, OF4.2, OF4.4, OF5.1, OF6.2
F_DCM.1	Provide S3-compatible storage service that encompasses edge-cloud continuum.	Edge Storage, Edge Storage Gateway	OF1.3, OF2.3, OF3.3, OF4.2, OF4.3, OF4.7, OF5.1
F_DCM.2	Provide an analytics-friendly erasure-coded IoT storage platform.	IoT Query Engine, Edge Storage, Edge Storage Gateway	OF4.7
F_DI.1	Decentralized decision-making, speculative and multi-objective resource orchestration.	EMPYREAN Registry, EMPYREAN Aggregator, Decision Engine, Service Orchestrator, EMPYREAN Controller	OF1.3, OF4.1, OF4.2, OF5.1, OF6.1
F_DI.2	Multi-agent speculative intelligent resource orchestration across EMPYREAN Associations.	Decision Engine, Service Orchestrator, EMPYREAN Controller, Container Layers Locality Scheduler	OF1.3, OF4.1, OF5.1, OF6.1
F_DI.3	Hierarchical orchestration and multi-objective optimization for cognitive resource orchestration within Associations.	Decision Engine, Service Orchestrator, EMPYREAN Controller, Container Layers Locality Scheduler	OF4.1, OF4.2, OF5.1
F_DI.4	AI-enhanced data orchestration and storage resource management within and across Associations.	Decision Engine, Service Orchestrator, Edge Storage Gateway, Edge Storage, Decentralized and Distributed Data Manager	OF1.3, OF2.3, OF3.3, OF4.2, OF4.3, OF5.1
F_DI.5	Energy-aware workload and data distribution mechanisms.	Decision Engine, Service Orchestrator, EMPYREAN Controller, Container Layers Locality Scheduler	OF4.1, OF4.2, OF4.8, OF5.1, OF6.1
F_DI.6	Monitoring and managing power and energy consumption in IoT devices and edge nodes.	Monitoring Probes, Telemetry Engine, Analytics Engine	OF2.1, OF2.2, OF4.2, OF4.8, OF5.1, OF6.1
F_DI.7	Decentralized and AI-enabled service assurance mechanisms.	Analytics Engine, Service Orchestrator, Telemetry Engine, Persistent Monitoring Data Storage	OF4.2, OF4.8, OF5.1, OF6.1, OF6.2
F_DI.8	AI-enhanced self-healing for enhanced resiliency, adaptability, and autonomous operation.	Workload Autoscaling, Analytics Engine, CTI Engine, Service Orchestrator, EMPYREAN Controller, Telemetry Engine, Persistent Monitoring Data Storage	OF4.2, OF4.8, OF5.1, OF6.1, OF6.2, OF6.3

F_DI.9	Autonomous and adaptive workload autoscaling.	Workload Autoscaling, Analytics Engine, Container Layers Locality Scheduling, EMPYREAN Controller, Telemetry Engine, Persistent Monitoring Data Storage	OF4.1, OF4.8, OF6.1, OF6.2
F_SO.1	Continuum-native workflow-based application design considering dataflow programming and low-code techniques.	Workflow Manager, Dataflow Programming Component, NIX-based Environment Packaging, Application Packaging	OF3.1, OF3.2, OF4.1, OF4.3
F_SO.2	Deployment objectives (SLOs) definition for continuum-native applications	Workflow Manager, EMPYREAN Registry, Service Orchestrator, Decision Engine	OF3.1, OF4.1, OF6.1
F_SO.3	Seamless and declarative orchestration of self-organized distributed orchestration systems.	EMPYREAN Registry, Service Orchestrator, Decision Engine, EMPYREAN Controller, Container Layers Locality Scheduler	OF3.1, OF4.1, OF4.2, OF4.6, OF5.1
F_SO.4	Policy-based orchestration and efficient resource allocation.	Workflow Manager, EMPYREAN Registry, Service Orchestrator, Decision Engine, EMPYREAN Controller, Telemetry Engine	OF2.3, OF4.1, OF5.1, OF6.1
F_SO.5	Context awareness and autonomous adaptive response.	Workflow Manager, EMPYREAN Registry, Analytics Engine, Workload Autoscaling	OF4.1, OF4.2, OF4.8, OF5.1
F_SO.6	Transparent lifecycle management of hyper-distributed application components.	Workflow Manager, Dataflow Programming, Service Orchestrator, EMPYREAN Controller, Container Runtime	OF1.1, OF1.2, OF1.3, OF3.1, OF3.2, OF4.1, OF4.2, OF4.3, OF4.5, OF4.6, OF5.1, OF6.1
F_SO.7	Coordinate workload migration within and across Associations.	EMPYREAN Aggregator, Service Orchestrator, Decision Engine, Analytics Engine, EMPYREAN Controller	OF1.3, OF4.2, OF5.1, OF6.1
F_SO.8	Support automatic data migration operations within and across Associations.	EMPYREAN Aggregator, Service Orchestrator, Decision Engine, Analytics Engine, EMPYREAN Controller, Edge Storage Gateway, Decentralized & Distributed Data Manager	OF1.3, OF2.3, OF3.3, OF4.2, OF5.1, OF6.1
F_SO.9	Implementation and integration of custom scheduling policies.	Decision Engine, Container Layers Locality Scheduler	OF4.1, OF4.2, OF5.1
F_SO.10	Seamless orchestration and management of both container-based and serverless workloads.	Workflow Manager, Service Orchestrator, EMPYREAN Controller, NIX-based Environment Packaging, Container Runtime	OF4.1, OF4.2, OF4.5, OF5.1
F_SO.11	Flexible Hardware-accelerated execution.	Hardware Acceleration Abstractions, Container Runtime, Application Packaging, EMPYREAN Controller	OF4.5
F_SO.12	Offload acceleration to nearby devices.	Hardware Acceleration Abstractions, Container Runtime, EMPYREAN Controller, Software-Defined Edge Interconnect	OF4.5
F_SO.13	OCI-compatible container images.	Unikernel Application Builder, NIX-based Environment Packaging, Unikernel Deployment, Container Runtime	OF3.1, OF3.2, OF4.1, OF4.2, OF4.4, OF4.6, OF5.1

F_SO.14	Support diverse execution environments.	Unikernel Application Builder, NIX-based Environment Packaging, Unikernel Deployment, Container Runtime, EMPYREAN Controller	OF3.1, OF3.2, OF4.1, OF4.2, OF4.4, OF4.6, OF5.1
F_SO.15	Reproducible Environment Packaging	Unikernel Application Builder, NIX-based Environment Packaging, Unikernel Deployment	OF3.1, OF3.2, OF4.1, OF4.2, OF4.6, OF5.1

The following table presents the relationship between the non-functional requirement categories of ISO/IEC 25010 and all the functional requirements.

Table 33: Analysis of overall non-functional requirements

Requirement ID:	NF_GR.1	**Stakeholders Involved:**	All
Title:	Performance Efficiency		
Description:	This characteristic represents the performance relative to the number of resources used under stated conditions. This characteristic is composed of the following sub-characteristics: • **Time behaviour** - Degree to which the response, processing times and throughput rates of a product or system, when performing its functions, meet the requirements. • **Resource utilization** - Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet the requirements. • **Capacity** - Degree to which the maximum limits of a product or system parameter meet requirements.		
Related Functional Requirements	***F_GR.4*** Provide seamless deployment of hyper-distributed cloud-native applications across a collaborative IoT-edge-cloud continuum. ***F_GR.5*** Support hyper-distributed, highly-demanding, and dynamic applications from diverse domains. ***F_GR.6*** Provide monitoring for cloud-native applications and heterogeneous infrastructure resources. ***F_GR.7*** Energy and power aware operation for optimal power management, energy efficiency and ecological sustainability. ***F_ASSOC.5*** Provide low and predictable latency for hyper-distributed applications. ***F_ASSOC.8*** Aggregators must maintain a catalogue of the Association resources. ***F_ASSOC.9*** Aggregators must dynamically discover resources within the registered infrastructures and detect events. ***F_ASSOC.10*** Aggregators must maintain the state of the Association. ***F_ST.4*** Automated Cyber Threat Analysis. ***F_ST.5*** ML for Anomaly Detection and Cybersecurity. ***F_DCM.1*** Provide S3-compatible storage service that encompasses edge-cloud continuum. ***F_DI.4*** AI-enhanced data orchestration and storage resource management within and across Associations. ***F_DI.5*** Energy-aware workload and data distribution mechanisms.		

	F_DI.6 Monitoring and managing power and energy consumption in IoT devices and edge nodes. ***F_SO.4*** Policy-based orchestration and efficient resource allocation. ***F_SO.6*** Transparent lifecycle management of hyper-distributed application components. ***F_SO.7*** Coordinate workload migration within and across Associations. ***F_SO.10*** Seamless orchestration and management of both container-based and serverless workloads. ***F_SO.11*** Flexible Hardware-accelerated execution. ***F_SO.12*** Offload acceleration to nearby devices.

Requirement ID:	NF_GR.2	**Stakeholders Involved:**	All
Title:	Functional Suitability		
Description:	This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. This characteristic is composed of the following sub-characteristics: • **Functional completeness** - Degree to which the set of functions covers all the specified tasks and user objectives. • **Functional correctness** - Degree to which a product or system provides the correct results with the needed degree of precision. • **Functional appropriateness** - Degree to which the functions facilitate the accomplishment of specified tasks and objectives**.**		
Related Functional Requirements	***F_GR.4*** Provide seamless deployment of hyper-distributed cloud-native applications across a collaborative IoT-edge-cloud continuum. ***F_GR.5*** Support hyper-distributed, highly-demanding, and dynamic applications from diverse domains. ***F_SO.1*** Continuum-native workflow-based application design considering dataflow programming and low-code techniques. ***F_SO.2*** Deployment objectives (SLOs) definition for continuum-native applications. ***F_IPDR.1*** Expose well-defined APIs through EMPYREAN SDK. ***F_IPDR.2*** Build upon well-established open-source platforms and consortium existing solutions. ***F_IPDR.6*** CI/CD guidelines.		

Requirement ID:	NF_GR.3	**Stakeholders Involved:**	All
Title:	Compatibility		
Description:	Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions while sharing the same hardware or software environment. This characteristic is composed of the following sub-characteristics: • **Co-existence -** Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.		

	• **Interoperability** - Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.
Related Functional Requirements	***F_GR.1*** Federate heterogeneous, distributed IoT, edge and cloud resources. ***F_GR.2*** Enable collaborative autonomy in the IoT-edge-cloud continuum. ***F_ASSOC.1*** Combine heterogeneous computational, storage resources, different connectivity resources. ***F_ASSOC.2*** Facilitate secure onboarding of new IoT devices, robots, edge/cloud resources within the EMPYREAN control and management plane. ***F_ASSOC.6*** Provide inter-Association communication and exchange of events. ***F_SO.6*** Transparent lifecycle management of hyper-distributed application components. ***F_SO.7*** Coordinate workload migration within and across Associations. ***F_SO.8*** Support automatic data migration operations within and across Associations. ***F_SO.13*** OCI-compatible container images. ***F_SO.14*** Support diverse execution environments. ***F_SO.15*** Reproducible Environment Packaging.

Requirement ID:	NF_GR.4	**Stakeholders Involved:**	All
Title:	Usability		
Description:	Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness and efficiency in a specified context of use. This characteristic is composed of the following sub-characteristics: • **Appropriateness recognizability** - Degree to which users can recognize whether a product or system is appropriate to their needs. • **Learnability -** Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use. • **Operability** - Degree to which a product or system has attributes that make it easy to operate and control. • **User error protection** - Degree to which a system protects users against making errors.		
Related Functional Requirements	***F_GR.4*** Provide seamless deployment of hyper-distributed cloud-native applications across a collaborative IoT-edge-cloud continuum. ***F_ASSOC.1*** Combine heterogeneous computational and storage resources and different connectivity resources. ***F_ASSOC.2*** Facilitate secure onboarding of new IoT devices, robots and edge/cloud resources within the EMPYREAN control and management plane. ***F_DCM.2*** Provide an analytics-friendly erasure-coded IoT storage platform. ***F_SO.1*** Continuum-native workflow-based application design considering dataflow programming and low-code techniques. ***F_SO.4*** Policy-based orchestration and efficient resource allocation. ***F_SO.6*** Transparent lifecycle management of hyper-distributed application components.		

	F_SO.10 Seamless orchestration and management of both container-based and serverless workloads. ***F_SO.15*** Reproducible Environment Packaging. ***F_IPDR.1*** Expose well-defined APIs through EMPYREAN SDK. ***F_IPDR.2*** Build upon well-established open-source platforms and consortium existing solutions. ***F_IPDR.3*** Documentation of all integration points.

Requirement ID:	NF_GR.5	**Stakeholders Involved:**	All
Title:	Reliability		
Description:	Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. This characteristic is composed of the following sub-characteristics: • **Maturity** - Degree to which a system, product or component meets needs for reliability under normal operation. • **Availability** - Degree to which a system, product or component is operational and accessible when required for use. • **Fault tolerance** - Degree to which a system, product or component operates as intended despite the presence of hardware or software faults. • **Recoverability** - Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.		
Related Functional Requirements	***F_GR.2*** Enable collaborative autonomy in the IoT-edge-cloud continuum. ***F_GR.3*** Encompass autonomous and continuous control loops. ***F_GR.6*** Provide monitoring for cloud-native applications and heterogeneous infrastructure resources. ***F_ASSOC.4*** Support autonomous operation and enhance resiliency across the continuum. ***F_ASSOC.7*** Data-driven seamless workload and data migration across the Associations. ***F_DI.1*** Decentralized decision-making, speculative and multi-objective resource orchestration. ***F_DI.7*** Decentralized and AI-enabled service assurance mechanisms. ***F_DI.8*** AI-enhanced self-healing for enhanced resiliency, adaptability, and autonomous operation. ***F_DI.9*** Autonomous and adaptive workload autoscaling. ***F_SO.6*** Transparent lifecycle management of hyper-distributed application components. ***F_SO.7*** Coordinate workload migration within and across Associations. ***F_SO.8*** Support automatic data migration operations within and across Associations.		

Requirement ID:	NF_GR.6	**Stakeholders Involved:**	All
Title:	Security		
Description:	Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access		

	appropriate to their types and levels of authorization. This characteristic is composed of the following sub-characteristics: • **Confidentiality** - Degree to which a product or system ensures that data are accessible only to those authorized to have access. • **Integrity** - Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data. • **Non-repudiation** - Degree to which actions or events can be proven to have taken place so that the events or actions cannot be repudiated later. • **Accountability** - Degree to which the actions of an entity can be traced uniquely to the entity. • **Authenticity** - Degree to which the identity of a subject or resource can be proved to be the one claimed.
Related Functional Requirements	***F_GR.5*** Support hyper-distributed, highly-demanding, and dynamic applications from diverse domains. ***F_ASSOC.3*** Constitute a secure and trustworthy execution environment. ***F_ASSOC.4*** Support autonomous operation and enhance resiliency across the continuum. ***F_ST.1*** Decentralized identity management. ***F_ST.2*** Privacy-Preserving authentication and authorization. ***F_ST.3*** Policy-Based Encryption. ***F_ST.4*** Automated Cyber Threat Analysis. ***F_ST.5*** ML for Anomaly Detection and Cybersecurity. ***F_ST.6*** Secure and Trusted Execution. ***F_DI.7*** Decentralized and AI-enabled service assurance mechanisms. ***F_DI.8*** AI-enhanced self-healing for enhanced resiliency, adaptability, and autonomous operation. ***F_IPDR.6*** CI/CD guidelines.

Requirement ID:	NF_GR.7	**Stakeholders Involved:**	All
Title:	Maintainability		
Description:	This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements. This characteristic is composed of the following sub-characteristics: • **Modularity** - Degree to which a system or computer program is composed of discrete components, such that a change to one component has minimal impact on other components. • **Reusability** - Degree to which an asset can be used in more than one system, or in building other assets. • **Analysability** - Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified**.** • **Modifiability** - Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.		

	• **Testability** - Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.
Related Functional Requirements	***F_GR.2*** Enable collaborative autonomy in the IoT-edge-cloud continuum. ***F_GR.3*** Encompass autonomous and continuous control loops. ***F_GR.6*** Provide monitoring for cloud-native applications and heterogeneous infrastructure resources. ***F_ASSOC.8*** Aggregators must maintain a catalogue of the Association resources. ***F_ASSOC.10*** Aggregators must maintain the state of the Association. ***F_DI.6*** Monitoring and managing power and energy consumption in IoT devices and edge nodes. ***F_SO.9*** Implementation and integration of custom scheduling policies. ***F_SO.10*** Seamless orchestration and management of both container-based and serverless workloads. ***F_IPDR.1*** Expose well-defined APIs through EMPYREAN SDK. ***F_IPDR.2*** Build upon well-established open-source platforms and consortium existing solutions. ***F_IPDR.3*** Documentation of all integration points. ***F_IPDR.4*** Docker image of developed components for creating containers. ***F_IPDR.5*** EMPYREAN Git-based code repository. ***F_IPDR.6*** CI/CD guidelines.

Requirement ID:	NF_GR.8	**Stakeholders Involved:**	All
Title:	Portability		
Description:	Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. This characteristic is composed of the following sub-characteristics: • **Adaptability** - Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments. • **Installability** - Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment. • **Replaceability** - Degree to which a product can replace another specified software product for the same purpose in the same environment.		
Related Functional Requirements	***F_GR.1*** Federate heterogeneous and distributed IoT, edge and cloud resources. ***F_GR.2*** Enable collaborative autonomy in the IoT-edge-cloud continuum. ***F_GR.4*** Provide seamless deployment of hyper-distributed cloud-native applications across a collaborative IoT-edge-cloud continuum. ***F_ASSOC.4*** Support autonomous operation and enhance resiliency across the continuum. ***F_ASSOC.9*** Aggregators must dynamically discover resources within the registered infrastructures and detect events. ***F_DI.7*** Decentralized and AI-enabled service assurance mechanisms.		

	F_DI.8 AI-enhanced self-healing for enhanced resiliency, adaptability, and autonomous operation. ***F_DI.9*** Autonomous and adaptive workload autoscaling. ***F_SO.1*** Continuum-native workflow-based application design considering dataflow programming and low-code techniques. ***F_SO.3*** Seamless and declarative orchestration of self-organized distributed orchestration systems. ***F_SO.5*** Context awareness and autonomous adaptive response. ***F_SO.13*** OCI-compatible container images. ***F_SO.14*** Support diverse execution environments. ***F_SO.15*** Reproducible Environment Packaging ***F_IPDR.1*** Expose well-defined APIs through EMPYREAN SDK. ***F_IPDR.2*** Build upon well-established open-source platforms and consortium existing solutions. ***F_IPDR.3*** Documentation of all integration points. ***F_IPDR.4*** Docker image of developed components for creating containers. ***F_IPDR.5*** EMPYREAN Git-based code repository. ***F_IPDR.6*** CI/CD guidelines.