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

Grant Agreement no. 101136024

Deliverable D2.2
Initial Release of EMPYREAN Architecture

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:	ICCS
Due date:	31/08/2024
Actual submission date:	30/09/2024 (initial submission v. 1.0) 24/10/2025 (revised version)

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
18.03.24	Aristotelis Kretsis	Draft	0.1	Initial ToC
25.05.24	Aristotelis Kretsis	Draft	0.2	Merge initial contributions in Sections 3,4, and 7
10.06.24	Panagiotis Kokkinos	Draft	0.3	Update and finalize ToC
09.07.24	Panagiotis Kokkinos	Draft	0.4	Integrate contributions by ICCS, UMU, CC, NEC, NUBIS in Sections 3,4,5
23.08.24	Aristotelis Kretsis	Draft	0.5	Integrate final contributions by ICCS, NVIDIA, CC, UMU, ZSCALE, RYAX, NUBIS, NEC in Sections 3,4,5,7
10.09.24	Aristotelis Kretsis	Draft	0.6	Finalize sections 3,4,5,7,8
21.09.24	Panagiotis Kokkinos	Draft	0.7	Integrate final contributions by IDEKO, EV ILVO, TRAC in Section 6
26.09.24	Aristotelis Kretsis	Draft	0.8	Revised version after internal review
30.09.24	ICCS	Final	1.0	
24.10.25	ICCS	Final	1.1	Address first project review comments

Author List

Organization	Author
ICCS	Aristotelis Kretsis, Panagiotis Kokkinos, Emmanouel Varvarigos
NVIDIA	Dimitris Syrivelis
CC	Marton Sipos, Marcell Feher, Daniel E. Lucani
UMU	Eduardo Cánovas, Antonio Skarmeta
ZSCALE	Ivan Paez
RYAX	Pedro Velho, Yuqiang Ma, Michael, Mercier, Yiannis Georgiou
NUBIS	Anastassios Nanos, Charalampos Mainas, Georgios Ntoutsos, Ilias Lagomatis, Maria Goutha, Panagiotis Mavrikos, Anastassios Tsakas, Konstantinos Papazafeiropoulos
IDEKO	Aitor Fernández, Javier Martín
NEC	Jaime Fúster, Roberto González
EV ILVO	Jan Bauwens, Theodoros Chalazas, Panagiotis Ilias

Internal Reviewers

Dimitris Syrivelis, NVIDIA
Marton Sipos, CC

Abstract: This deliverable (D2.2) presents the outcomes of Task 2.3 “Architecture Specification”, Work Package 2 “Requirements and System Design” of the EMPYREAN project, during the first iteration of the incremental implementation plan. The deliverable presents the initial architecture specifications of the EMPYREAN platform. It also provides a breakdown of the platform’s key building blocks and describes their interactions. Moreover, the EMPYREAN implementation and delivery plan is introduced.

Keywords: EMPYREAN Architecture, EMPYREAN Platform, Associations, Edge-Cloud Continuum, Cognitive Orchestration, Trustworthy, AI-Driven Data Processing

Table of Contents

List of Figures

List of Tables

Abbreviations

ABC	Attribute-based Credentials
ABE	Attribute-Based-Encryption
AI	Artificial Intelligence
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BDVA	Big Data Value Association
ATR	Autonomous Towing Robot
CD	Continuous Delivery
CI	Continuous Integration
CNC	Computer Numerical Control
CP-ABE	Ciphertext-Policy Attribute-Based-Encryption
CPU	Central processing unit
CRI	Container Runtime Interface
CTA	Cyber Threat Alliance
CTI	Cyber Threat Intelligence
D	Deliverable
DAG	Directed Acyclic Graph
DevOps	Development and Operations
DDS	Data Distribution Service
DID	Decentralized Identifier
DLT	Distributed Ledger Technologies
DRS	Data Recording System
DSBA	Data Space Business Alliance
EC	Elliptic Curve
EDC	Eclipse Data Space Connector
FCS	Fingerprint Comparison System
FCS	Fleet Control System
FGS	Fingerprint Generation System
FPGA	Field Programmable Gate Arrays
FTC	FIWARE True Connector
GHCR	GitHub Container Registry
GPU	Graphics Processing Unit
HPC	High Performance Computing
HW	Hardware
I/O	Input/Output
IaaS	Infrastructure as a Service
IDSA	International Data Spaces Association
IIoT	Industrial Internet of Things
IoC	Indicators of Compromise
IoT	Internent of Things
IP	Intellectual Property
JSON	Javascript Object Notation
JWT	JSON Web Token
K8s	Kubernetes
KPI	Key Performance Indicator

LAN	Local Area Network
M	Month
MAN	Metropolitan Area Network
MISP	Malware Information Sharing Platform
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NDN	Named-Data Networking
NIC	Network Interface Card
OBU	Onboard Unit
OCI	Open Container Initiative
OPC-UA	OPC Unified Architecture
OTA	Over-the-Air
PaaS	Platform as a Service
PMDS	Persistent Monitoring Data Storage
PS-MS	Pointcheval–Sanders Multi-Signatures
PSM	Privacy and Security Manager
Pub/Sub	Publish/Subscribe
RDMA	Remote Direct Memory Access
REST	Representational State Transfer
RTL	Register Transfer Level
SaaS	Software as a Service
SDK	Service Development Kit
SO	Service Orchestrator
SSI	Self-Sovereign Identity
SW	Software
TCP	Transmission Control Protocol
TEE	Trusted Execution Environment
TF	Task Force
TPU	Tensor Processing Unit
TSN	Time-Sensitive Networking
UC	Use Case
URI	Unique Resource Identifier
VC	Verifiable Credentials
VCS	Version Control System
VM	Virtual Machine
VP	Verifiable Presentations
WAN	Wide Area Network
WASM	WebAssembly
WP	Work Package
XACML	Extensible Access Control Markup Language
ZKP	Zero-Knowledge Proofs

1 Executive Summary

EMPYREAN envisions an IoT-edge-cloud continuum composed of collaborative collectives of IoT devices, robots, and resources, seamlessly extending from the edge to the cloud. EMPYREAN refers to this concept as the Association-based continuum, where multiple Associations—each a collaborative collective of IoT devices, robots, and diverse resources—operate concurrently across heterogeneous infrastructures, spanning different providers, geographical locations, and connectivity types. These Associations collectively form a dynamic and interconnected IoT-edge-cloud ecosystem.

In this Association-based continuum, EMPYREAN aims to autonomously balance computing tasks and data both within individual Associations and between federated Associations, optimizing resource utilization, performance, and resiliency. To achieve this, EMPYREAN is developing a distributed and AI-enabled control and management plane across the IoT-edge-cloud continuum. This infrastructure will facilitate the creation, management, and operation of Associations, supporting the ubiquitous computing, storage, and communication needs of current and future hyper-distributed, dynamic, and time-critical applications.

This deliverable introduces the initial reference architecture of the EMPYREAN platform, derived from the consortium's efforts to design a system that meets the requirements identified in deliverable D2.1 - "State of the art, use cases analysis, platform requirements and KPIs" (M6). The architecture is designed to provide the functionalities needed to achieve the project's objectives and address the specific needs of the use cases. Key innovations and technological advancements are also highlighted, showcasing the novel ecosystem of technologies that the EMPYREAN employs and develops.

The EMPYREAN platform features a multi-layer modular architecture that integrates distinct functionalities and features necessary to enable the Association-based continuum. This approach not only facilitates future extensions but also supports the independent use and exploitation of platform components. The architecture is detailed at both the conceptual and logical levels in this deliverable, with the logical architecture elaborating on all the components developed within the project and illustrating their core interactions. Furthermore, this deliverable provides an early overview of the EMPYREAN platform deployments that will support the use cases (UCs), laying the groundwork for subsequent implementation and evaluation phases. It also outlines the adopted implementation and delivery plan, which includes the software engineering approach that will guide the EMPYREAN platform development. This information serves as a framework for the technical Work Packages (WPs), ensuring alignment and coherence across development activities.

As the project progresses, the architecture will be iteratively refined based on feedback from ongoing development efforts. The final version of the architecture, including comprehensive workflows and interface specifications, will be documented in Deliverable D2.3 – "Final EMPYREAN Architecture, Use Cases Analysis, and KPIs," scheduled for completion at M12. This final deliverable will present the fully matured architecture, providing a detailed and actionable blueprint for the platform's deployment and operation.

2 Introduction

2.1 Purpose of this document

This deliverable presents the preliminary outcomes of Task 2.3 – "Architecture Specification" within Work Package 2 – "Use Cases Analysis, System Requirements and Overall Architecture" during its initial phase (M1 – M7). Task 2.3 focuses on designing the overall EMPYREAN architecture and defining the workflows and interfaces between the EMPYREAN components, ensuring a structured and coherent system integration.

The primary objective of D2.2 is to build upon the initial contributions of Tasks 2.1 and 2.2, as described in deliverable D2.1 (M6). It aims to provide a comprehensive description of EMPYREAN's components, along with the associated technical specifications that will guide the project's next phases. In this context, D2.2 outlines the initial high-level architecture of the EMPYREAN platform, detailing the interactions among its key building blocks and offering a preliminary detailed architecture. Additionally, it includes an early description of the EMPYREAN platform deployments that will support the project's use cases (UCs), laying the groundwork for future implementation.

D2.2 will serve as a critical reference for the research and development activities within the technical work packages (WP3-4) and will also support the development, evaluation, and integration activities related to the project's use cases (WP5-6). Each technical work package will ensure that all project developments are in full alignment with the architecture and specifications outlined in this deliverable, maintaining consistency across the project's various activities. As the technical work progresses, the architecture will be iteratively refined and expanded, incorporating more detailed technical specifications for the components, interfaces, and workflows.

Furthermore, the forthcoming iteration of this deliverable will provide comprehensive interface specifications for EMPYREAN platform components, alongside detailed workflow descriptions that capture the interactions between these components, thereby ensuring a seamless and integrated approach throughout the project. The final output of Task 2.3 will be documented in D2.3 - "Final EMPYREAN architecture, use cases analysis and KPIs", which is scheduled for completion in M12 of the project, culminating the architectural development efforts with a refined and fully specified system blueprint.

2.2 Document structure

The present deliverable is split into six major chapters:

- EMPYREAN Technologies and Resources
- EMPYREAN Platform Components
- EMPYREAN Architecture
- EMPYREAN Platform Deployment View
- Implementation and Deliverable Plan
- Requirements Coverage

2.3 Audience

This document is publicly available and should be useful to anyone interested in the initial description of the EMPYREAN components and the specification of the initial release of the EMPYREAN architecture. Moreover, this document can also help the general public better understand the framework and scope of the EMPYREAN project.

3 EMPYREAN Technologies and Resources

EMPYREAN aims to develop an ecosystem of innovative technologies that enable a collaborative, trustful, and cognitive IoT-edge-cloud continuum. This continuum efficiently and autonomously integrates heterogeneous infrastructure resources from various technological domains with diverse characteristics and capabilities. The key technologies developed in EMPYREAN include: (i) lightweight, privacy-preserving solution for Attribute-Based Credentials (ABCs) in decentralized environments, (ii) secure and trusted execution environments with workload isolation across the IoT-edge-cloud continuum, (iii) software-defined IoT-edge interconnects, and (iv) hardware acceleration abstractions for the flexible execution of workloads that can benefit from hardware acceleration.

The following sections provide more information on the hardware resources considered within the EMPYREAN platform and associated software technologies developed within the EMPYREAN project.

3.1 EMPYREAN Hardware Resources

3.1.1 Hardware Accelerators

Hardware accelerators are specialized systems designed to enhance the efficiency of computations performed by general-purpose processors, such as CPUs. Many computational tasks that can be parallelized, while typically executable on a generic CPU, can also be offloaded to custom-designed hardware for significantly faster performance. Hardware accelerators are commonly employed in various domains, including machine learning (ML), computer vision, video editing/rendering, digital signal processing, and cryptography, However, the use of hardware acceleration is most beneficial when dealing with computationally intensive algorithms that demand high throughput and performance. For routine tasks or algorithms that are more serial in nature, a CPU may actually be more efficient. This is because CPUs generally operate at higher clock frequencies than accelerator cores and are optimized for handling serial tasks and branching logic.

Nevertheless, offloading performance-critical functions to specialized hardware can be a highly effective strategy to reduce execution time and improve energy efficiency. Hardware accelerators such as Graphics Processing Units (GPUs), Field-Programmable Gate Arrays (FPGAs), and Application-Specific Integrated Circuits (ASICs) are commonly used for this purpose. GPUs consist of a large number of specialized processing units designed to handle massive parallel data streams, making them ideal for tasks such as matrix calculations in deep learning and rendering in computer graphics. FPGAs and ASICs, on the other hand, implement fixed-function algorithms directly in hardware. FPGAs offer the flexibility to be reprogrammed for different tasks, while ASICs are highly efficient but custom-built for specific algorithms, providing optimal performance in their respective domains.

These accelerators are often deployed in environments such as edge computing, where they operate close to the data source in compact devices that integrate memory, CPU, I/O, and acceleration capabilities. Alternatively, they can be found in cloud environments, typically integrated into server systems where they handle large-scale, data-intensive workloads. By harnessing the power of hardware accelerators, applications can achieve faster processing speeds and greater energy efficiency, especially in scenarios requiring the parallel processing of large datasets.

3.1.2 Smart NICS and Data Processing Units

NVIDIA Mellanox ConnectX® SmartNICs utilize stateless offload engines, overlay networks, and native hardware support for RoCE and GPUDirect™ technologies that reduce I/O latency within and beyond the server boundaries and results in significantly improved application performance. Developers can use ConnectX custom packet processing technologies to accelerate server-based networking functions and offload datapath processing for compute-intensive workloads, including transport, network virtualization, security, and storage functionalities.

The NVIDIA® BlueField®-3 data processing unit (DPU) delivers a broad set of accelerated software defined networking, storage, security, and management services with the ability to offload, accelerate and isolate data center infrastructure. Sitting at the edge of every server, BlueField-3 empowers agile, secured and high-performance cloud and artificial intelligence (AI) workloads, all while reducing the total cost of ownership and increasing data center efficiency. The NVIDIA DOCA™ software framework enables developers to rapidly create applications and services for the DPU. NVIDIA DOCA makes it easy to leverage DPU hardware accelerators, providing breakthrough data center performance, efficiency and security.

3.1.3 Smart Boxes for CNC Machines

IDEKO collaborates closely with Savvy Data Systems, a technological start-up specializing in machine-monitoring and data analytics. Together, they developed the Smart Box, an industry-ready device to collect machine data. The Smart Box serves as both a data collection tool and a data gateway featuring an industrial PC setup. It is capable of connecting to the most common CNC models (machines) and other data sources and sensors.

There are several models of the Smart Box, each with different price ranges and computational power. Typically, the boxes are deployed to one-to-one basis, one box is deployed for each CNC machine. This arrangement can result in some machines having more computational power than others, depending on the model of the box attached to them. Despite these differences, each Smart Box comes with the full set of connectivity options required to gather data form the machine and send it to a private cloud for further analysis. It is designed to capture machine performance indicators such as axis positions, oil pressure, and running programs at a configurable frequency, usually every second. One of the key advantages of the Smart Box is its ease of use, as it is nearly plug-and-play, with support for remote configuration

and software upgrades. Additionally, it is compatible with Docker containers, enabling users to deploy edge computing solutions directly on the device. This is particularly useful for scenarios where customers prefer to process data locally rather than sending it to the cloud, either for privacy concerns or specific operational requirements.

3.2 EMPYREAN Software Components

3.2.1 p-ABC C Library

The p-ABC C Library is a lightweight, privacy-preserving solution designed to support Attribute-Based Credentials (ABCs) in decentralized environments. It is optimized for IoT and low-computation devices, enabling independent credential management. Utilizing advanced cryptographic techniques like Pointcheval–Sanders (PS) multi-signatures (MS) and Zero-Knowledge Proofs (ZKFs), the library ensures secure authentication. It enhances identity management and access control across the EMPYREAN ecosystem through its integration with the Privacy and Security Manager (Section 4.1).

Table 1: Description of p-ABC C Library components

Component ID	WP3.1.2
Name	p-ABC C Library
High level description	The p-ABC library is a cutting-edge solution designed to facilitate distributed, privacy-preserving Attribute-Based Credentials (ABCs) by leveraging PS Multi-Signatures (MS), with a robust foundation in Elliptic Curve (EC) arithmetic. It empowers developers to implement advanced cryptographic protocols, enabling secure and private authentication mechanisms. The p-ABC library enables the creation of derived, verifiable presentations that can be shared and validated without compromising user privacy. By employing ZKFs, the module ensures that sensitive information remains confidential, even in verification process. It is ideal for applications requiring high levels of privacy and security, such as decentralized identity systems, confidential transactions, and secure access control. Moreover, the p-ABC module is designed with ease of use in mind, offering an automated script to streamline the setup process. This feature simplifies its deployment and integration into existing systems, making it accessible even for those who may not have extensive expertise in cryptography. Overall, the p-ABC module library represents the latest in privacy-preserving technology, providing the tools necessary to build secure, privacy-centric solutions in a digital world that increasingly demands robust data protection. It enhances the Privacy and Security Manager by enabling the creation and management of privacy-preserving Verifiable Credentials and Presentations, crucial for secure authentication in IoT ecosystems. The library integrates advanced cryptographic techniques, such as zero-knowledge proofs and selective disclosure, ensuring that IoT devices can securely interact and authenticate without compromising user privacy.
Collaborators	• Privacy and Security Manager (WP3.1.1)
Interfaces	• C Library compiled as dependency

UCs	It will be used in all use cases as part of the secure and trust mechanisms within the EMPYREAN platform.

3.2.2 Secure Execution Environment

This component establishes a secure and trusted execution environment across the IoT-edge-cloud continuum, supporting deployment from resource-constrained edge devices to cloud infrastructures. It integrates (i) unikernel-based architecture, (ii) secure boot mechanisms, (iii) container image attestation, and (iv) secure over-the-air (OTA) updates, ensuring both security and scalability. It leverages unikernels, which are minimal, application-specific operating systems. By running only the necessary components for a single application, unikernels reduce the attack surface and increase efficiency, making them well-suited for IoT and edge environments with limited resources. Moreover, this component implements secure and measured boot mechanisms, ensuring that only trusted and verified software is loaded during the boot process. This tight security integration into the system boot process guarantees that both the hardware and software stacks are validated, providing a root of trust from system start-up.

To enhance security, the component includes attestation of all deployed workloads by simplifying deployment using container images. We attest container images through cryptographic signing, so before deploying or running applications across the IoT-edge-cloud continuum, each container image is verified for authenticity and integrity. This approach ensures that only trusted, tamper-free applications are deployed, minimizing risks of malicious software or altered images entering the system. This functionality enables fully secure OTA updates, allowing devices and systems to receive firmware and application updates remotely. These updates are cryptographically signed and verified before installation, ensuring they come from trusted sources. This capability is crucial for maintaining the security and reliability of distributed IoT and edge systems, especially in environments where physical access to devices is limited.

Additionally, applications can be deployed across micro and deep edge, far edge, and cloud environments without modifying their deployment descriptors or application logic via pure cloud-native for IoT devices. This transparent and scalable operation allows seamless adaptation across different hardware platforms.

The component’s approach to secure over-the-air (OTA) updates embraces a cloud-native architecture, ensuring that devices across the IoT-edge-cloud continuum can seamlessly receive updates from a central management system. By leveraging cloud-native technologies, updates are delivered in a highly automated and scalable manner, regardless of the number of connected devices or their geographical distribution. The OTA updates are cryptographically signed and verified as container images, ensuring that only trusted and authorized updates are installed, thus protecting the system from potential attacks. This cloud-native methodology allows updates to be managed dynamically, with minimal downtime, enhancing the overall resilience and reliability of the system. Moreover, integrating OTA updates within a cloud-native environment means that updates can be easily

customized and tailored based on specific device profiles or hardware capabilities. As new software versions or security patches are released, they can be securely distributed across various layers, from deep edge devices to central cloud servers, without manual intervention. The central control plane in the cloud can manage update rollouts, monitor device health, and initiate rollback mechanisms if needed, all while ensuring efficient use of resources and minimizing the impact on operational workloads. This allows deploying critical updates in real-time while maintaining a consistent and secure execution environment across the continuum.

Table 2: Description of Secure Execution Environment

Component ID	WP3.1.3
Name	Secure Execution Environment
High level description	This component is focused on establishing a secure and trusted execution environment across the IoT-edge-cloud continuum. This environment based on unikernels will support secure and measured boot mechanisms that are tightly coupled with the systems layer. This approach will ensure that applications can be deployed with varying levels of security and trustworthiness across different hardware, enabling scalable and transparent operation from the micro deep edge to far edge and to cloud environments without altering the deployment descriptor or application logic. Furthermore, it will address the trade-off between flexible workload deployment and the single-tenant use of computing resources, particularly in energy and resource-constrained edge platforms.
Collaborators	• Privacy and Security Manager (WP3.1.1) • EMPYREAN Controller (WP4.4.4) • Application Builder for Unikernels (WP4.3.2)
Interfaces	Internal interfaces with the platform-level container orchestration mechanisms.
UCs	UC2 and UC3 will use this component as part of the EMPYREAN Security, Privacy, and Trust layer.

3.2.3 Software-defined Edge Interconnect

This component aims to integrate Remote Direct Memory Access (RDMA) network transport technology into the communication and interconnection layer of selected EMPYREAN components. RDMA's key value proposition lies in its ability to offload network transport entirely to a dedicated hardware accelerator within the Network Interface Card (NIC). This offloading process (Figure 1) dramatically reduces network I/O software overheads for the communication endpoints, resulting in substantial improved performance in both latency and bandwidth.

At EMPYREAN, we have identified two main use cases for leveraging this RDMA-based support: (a) integration of hardware sensor equipment with wired network without any CPU involvement using the FlexDriver FPGA IP, which enables seamless data transfer between hardware sensors and the network, bypassing the CPU entirely for more efficient communication, and (b) integration of edge nodes with a central AI cluster using a FlexDriver-

based publish/subscribe (Pub/Sub) frontend software service, facilitating high-performance communication between distributed edge nodes and a central AI processing cluster.

The primary goal of RDMA support within EMPYREAN is to significantly accelerate small-sized message performance across the platform, addressing a well-known limitation of standard TCP-based Ethernet communication. TCP often struggles with the efficient handling of small message, which hinders scalability in high-performance environments.

The EMPYREAN FlexDriver service aims to overcome this challenge by offering a software-defined interface that allows a flexible definition of the communication descriptors. This enables their fine-tuning to improve performance, particularly for small-sized messages, thereby enhancing overall system efficiency and scalability.

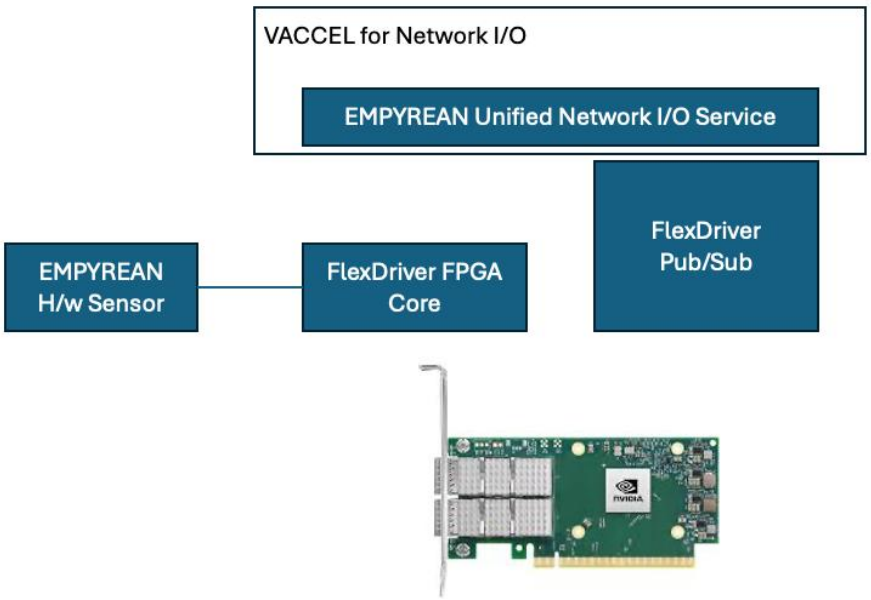

Figure 1: EMPYREAN edge node with wired backhaul and attached hardware sensors

Table 3: Description of software-defined edge interconnect components

Component ID	WP3.3.2
Name	H/W RDMA Transport Service
High level description	RDMA-based transport service in the form of Register Transfer Level (RTL) Intellectual Property (IP) that can be directly integrated into an FPGA-based design. The FPGA platform needs to be physically attached to a PCI-e bus of a server host (typically, it should have a PCI-e card form factor). Using Xilinx FPGA is also a hard requirement, as there are dependencies on vendor-specific RTL IPs. The host server needs an additional PCI-e slot where an NVIDIA ConnectX-6Dx NIC will be attached, which will offer the RDMA network function to the accelerator of the described hardware transport service component. The accelerator can use the transport service to directly communicate with another accelerator (Accelerator2Accelerator) or a centralized software service

	(Accelerator2Software) with very low latency and jitter without requiring the accelerator to run any software.
Collaborators	● Hardware Acceleration Abstraction (WP3.3.4)
Interfaces	Inputs: ● H/W accelerator description and available interfaces ● Netlist of the accelerator built by a Xilinx toolchain Outputs: ● Integrated FPGA design that allows an accelerator to directly communicate with other accelerators or the edge without the need for a host running any software.
UCs	UC1 will use this component as part of the EMPYREAN Data Management and Interconnection layer.

Component ID	WP3.3.3
Name	S/W RDMA-based Pub/Sub Transport Service
High level description	It is a software-based Pub/Sub transport service that is brokerless and offers all the benefits of RDMA (low jitter and low latency) to the applications that need to communicate at the edge. The communication interface offered will be similar to Pub/Sub but with reduced functionality with regards to what would have been offered if a broker acted as a mediator. The broker functionality is traded for the significantly lower latency and jitter, which is the foundation for guaranteeing near real-time operation where required. The bring up of Pub/Sub communications is offered by a software-defined rendezvous server.
Collaborators	● Decentralized and Distributed Data Manager (WP3.2.3)
Interfaces	Inputs: ● Software application transport performance requirements and current network I/O interface design Outputs: ● Applications using the described transport for communication
UCs	UC1 will use this component as part of the EMPYREAN Data Management and Interconnection layer.

3.2.4 Hardware Acceleration Abstractions

The vAccel[1] framework is designed to accelerate workloads, particularly those that involve AI/ML, using hardware accelerators like GPUs, TPUs, or FPGAs. It aims to abstract the complexity of hardware accelerators from application developers, allowing them to easily offload compute-heavy tasks to more powerful hardware. At its core, vAccel provides a standardized API and abstraction layer, making hardware-accelerated computing more

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

accessible, especially in cloud-native and edge environments. Figure 2 shows the architecture of the vAccel software stack.

The framework abstracts the specifics of different hardware accelerators, providing a unified interface for developers. This design makes it easier to deploy applications that require hardware acceleration without knowing the details of how the underlying accelerators work. vAccel can interface with multiple hardware backends (e.g., GPUs, TPUs, FPGAs) and cloud environments, allowing it to run on diverse platforms, including IoT edge devices, cloud servers, and data centers. Backends (*plugins* in vAccel terminology) can also be extended to support new accelerators or customized for specific applications. vAccel is designed with security in mind. It can integrate with security frameworks like Confidential Computing (e.g., using Trusted Execution Environments—TEEs), allowing for secure execution of workloads, even in shared environments like public clouds. The framework aligns well with containerized applications and Kubernetes environments, offering cloud-native support.

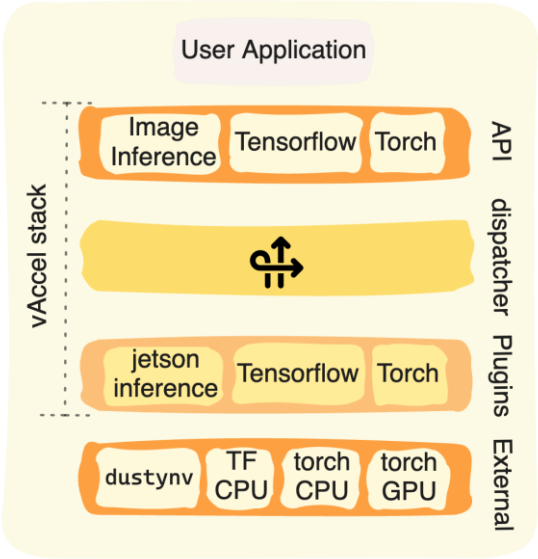

Figure 2: vAccel software stack

Just like generic (hardware) plugins, the vAccel transport plugin enables the forwarding of an API operation to a vAccel library instance that is executing on a different context (Host instead of Guest VM, or even a remote Host). In particular, the vsock plugin allows for efficient and secure communication between isolated environments (e.g., virtual machines, containers) and the host system via a generic RPC protocol. It leverages virtio-vsock, which is a para-virtualized socket that enables communication without needing traditional networking stacks (e.g., TCP/IP). This makes vsock ideal for environments where you want low-overhead communication between a guest and host, while maintaining isolation.

Additionally, the vsock plugin is compatible with generic sockets, making it ideal for remote execution over the network. IoT and edge devices often have limited computational resources

but require real-time AI/ML inference for tasks like object detection, anomaly detection, or predictive maintenance. The vAccel framework, with the vsock plugin, enables IoT and lightweight edge devices to offload computationally expensive tasks to more powerful edge nodes or nearby virtualized environments, which may host accelerators.

In EMPYREAN, we will enable this functionality, porting the existing gRPC protocol to an IoT software framework (e.g., esp-idf) and, combined with the pure cloud-native support, already existing for the vAccel framework, enable end-to-end, hardware-acceleration-enabled execution across the whole continuum (IoT, Edge, Cloud nodes).

Table 4: Description of vAccel component

Component ID	WP3.3.4
Name	vAccel
High level description	vAccel is an open-source framework designed to enable flexible execution by mapping hardware-accelerate-able workloads to relevant hardware functions, thus decoupling applications from hardware-specific code. It aims at enhancing security by ensuring that consecutive executions on a hardware-accelerated platform do not leak sensitive data. This framework is part of the EMPYREAN project's initiative to facilitate the development and deployment of compute-intensive functions across IoT devices and edge nodes, leveraging the concept of remote hardware accelerators. IoT devices can use the vAccel API to request compute-intensive tasks to be executed by an available neighbouring node within the Association, integrating with established open-source solutions at the systems level (e.g., Kubernetes, K3s, OpenFaaS) and including their high-level APIs in the EMPYREAN SDK to simplify application development and deployment.
Collaborators	● Software-defined Edge Interconnect (WP3.3.1) ● EMPYREAN Controller (WP4.4.4) ● Decentralized and Distributed Data Manager (WP3.2.3)
Interfaces	It provides for popular languages, such as C, Python, and Rust.
UCs	UC1 and UC2 will use this component as part of the EMPYREAN Resource Management layer.

4 EMPYREAN Platform Components

EMPYREAN aims to deliver trustworthy, cognitive, and AI-driven collaborative associations of IoT devices and edge resources for efficient data processing across the continuum. To this end, EMPYREAN introduces and develops innovative technologies that bridge existing technological gaps, enabling a cognitive computing continuum. The Association-based continuum seamlessly integrates intelligence and automation, resulting in more efficient, adaptive, and scalable data processing capabilities. EMPYREAN's contributions focus on several key factors for realising this continuum, including intelligence and automation, trustworthiness and security, heterogeneous IoT-edge mesh connectivity, interoperability, elasticity and energy efficiency.

In the sections that follow, we present an overview of the main components of the EMPYREAN platform.

4.1 Privacy and Security Manager

The Privacy and Security Manager is a core component of the EMPYREAN architecture, designed to ensure advanced privacy and security features across decentralized environments, especially in IoT ecosystems. By leveraging Decentralized Identifiers (DIDs) and Self-Sovereign Identity (SSI) systems, it manages Verifiable Credentials (VCs) and Verifiable Presentations (VPs) using cryptographic techniques such as Zero-Knowledge Proofs (ZKPs) and selective disclosure. This design enables privacy-preserving, secure interactions, while also ensuring the integrity and authenticity of identities within the ecosystem.

Figure 3 and Table 5 present the Privacy and Security Manager components and their dependencies with other components within the EMPYREAN platform.

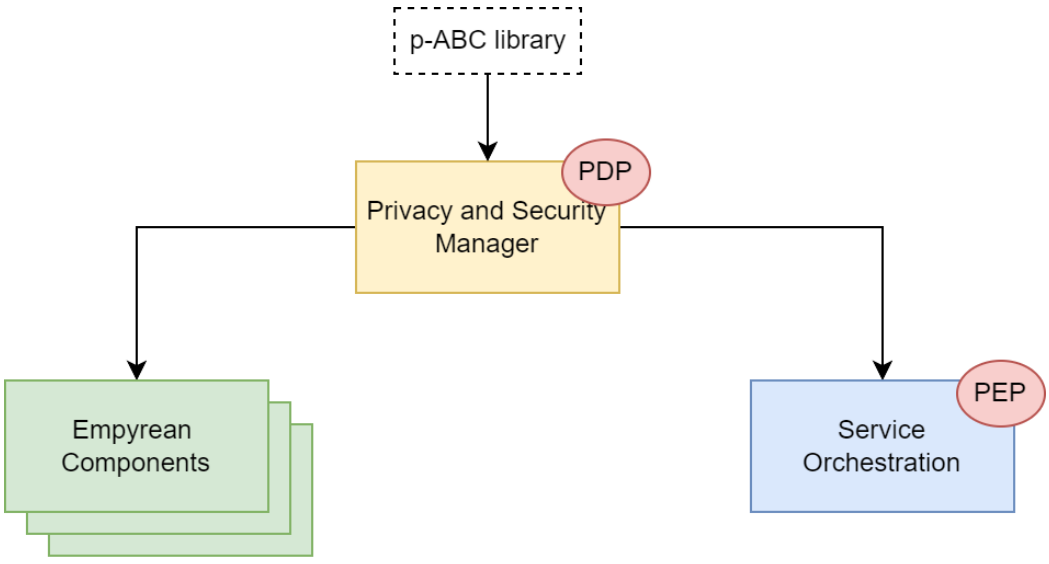

Figure 3: Privacy and Security Manager interaction with other EMPYREN services

Table 5: Description of Privacy and Security Manager core components

Component ID	WP3.1.1
Name	Privacy and Security Manager
High level description	This component enables and enforces privacy and security features within decentralized ecosystems, particularly in environments involving IoT devices. It provides comprehensive management of Self-Sovereign Identity (SSI) systems, leveraging Decentralized Identifiers (DIDs) to enable secure, user-controlled identity solutions. Key functionalities include the management and authorization of Verifiable Credentials (VCs) and the generation of Verifiable Presentations (VPs). Utilizing the advanced cryptography of the p-ABC module (Section 3.2.1), the Privacy and Security Manager supports the creation of verifiable presentations that employ Zero-Knowledge Proofs (ZKPs) and selective disclosure. This ensures that only the necessary attributes are revealed during interactions, preserving user privacy without compromising the integrity of the data. Additionally, the Privacy and Security Manager facilitates signing JSON Web Tokens (JWTs) using DIDs, enabling secure and verifiable identity management. This capability is particularly useful for creating fast access tokens for any entity within the ecosystem, allowing swift and secure access to resources while ensuring the authenticity and integrity of the identities involved. To further enhance security, the Privacy and Security Manager integrates with Distributed Ledger Technologies (DLTs) to verify credentials, ensuring transparency and immutability in the authentication processes. It also employs smart contracts to securely retrieve and store DIDs, automating these processes to reduce manual intervention and potential security risks. In summary, the Privacy and Security Manager is a powerful tool for enabling secure, privacy-preserving interactions within decentralized environments. It offers the EMPYREAN ecosystem advanced identity management, credential verification, and transaction security, all while ensuring that user privacy is maintained through state-of-the-art cryptographic techniques.
Collaborators	• P-ABC (WP3.1.1) • Service Orchestrator (WP4.4.1)

Interfaces	• REST interface with methods to provide security
	  • /empyrean/psm/generateDID
	  • /empyrean/psm/doEnrolment
	  • /empyrean/psm/generateVP
	  • /empyrean/psm/verifyCredential
	  • /empyrean/psm/signJWTContent
	  • /empyrean/psm/verifyJWT

UCs	In UC2, Proximal Sensing in Agriculture Fields, it will provide DID's, VCreds and framework for verification through a Decentralized Ledger for edge devices. In UC4, Security in Smart factories with S. Korea International Collaboration, it will provide authN/authZ to enable a security level.

4.2 Edge Storage and Edge Storage Gateway

EMPYREAN will develop an S3-compatible distributed secure storage service for the Associations that stretches across the edge-cloud continuum, using erasure coding to provide redundancy. Figure 4 shows the key building blocks of this service and their interactions with other EMPYREAN components, while Table 6 provides a detailed description of each component.

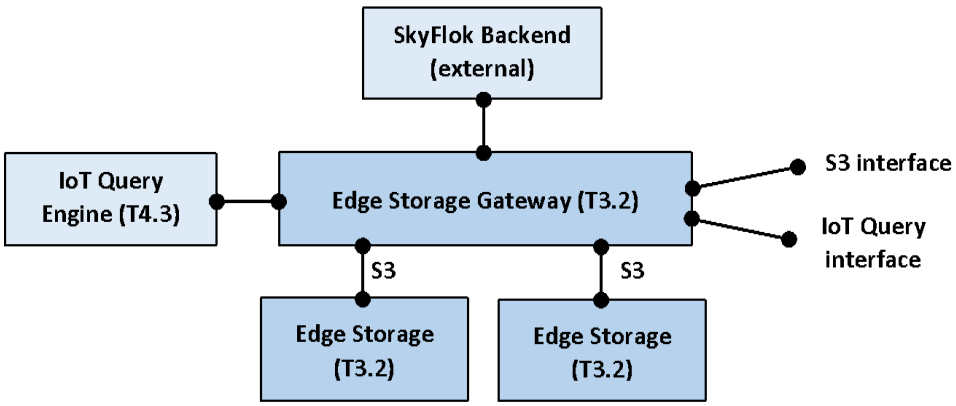

Figure 4: Edge Storage and Edge Storage Gateway components and dependencies

Table 6: Description of Edge Storage and Edge Storage Gateway components

Component ID	WP3.2.1
Name	Edge Storage Gateway
High level description	It provides access to SkyFlok Object Storage, a secure distributed data storage SaaS. EMPYREAN users can access their data through an industry-standard S3 interface. To improve performance, the Gateway should be deployed to the edge, close to the users as well as to edge storage resources. Users are given a choice of storage locations to which their data is distributed in an erasure coded manner. They can select both cloud (through SkyFlok) and edge (through Edge Storage) locations, as well as the level of redundancy needed. Beyond the standard S3 interface, the Gateway also provides a specialized interface and storage schema for IoT time series data. This feature is implemented in the IoT Query Engine (T4.3), the Gateway proxies requests to this component. Internally, the Gateway utilizes the SkyFlok backend for metadata storage, user and team management as well as authentication. It stores and retrieves data fragments to/from cloud storage providers and Edge Storages directly, performs the erasure coding, encryption and compression. Potentially, it could be extended to include an object cache as well.
Collaborators	• Edge Storage (WP3.2.2) • IoT Query engine (W4.3.5) • SkyFlok Backend (outside the scope of EMPYREAN)

Interfaces	• It exposes an S3-compatible object storage interface • It exposes an SQL-like IoT Query interface (REST)
UCs	An S3-compatible storage service will be provided to all three use cases, tailored using storage policies to the applications' requirements. An indirect approach is also possible, where this component acts as a data source and sink in a large scale data flow. The UC applications would then interact indirectly with this component through the data flow.

Component ID	WP3.2.2
Name	Edge Storage
High level description	Manages an Edge Storage device, it is used by the Edge Storage Gateway. This is a containerized application based on Min.io[2]. It makes it possible to take advantage of edge storage resources, making them accessible to the platform's storage system. Any storage resource that can be mounted as a K8s or Docker volume can thus be integrated into the storage system. It is an internal component that, once set up, only offers its S3 interface to the Gateway. It is not accessible as a storage location directly to platform users.
Collaborators	• Edge Storage Gateway (WP3.2.1)
Interfaces	• It exposes an S3 interface for the Edge Storage Gateway. • It exposes a Prometheus-compatible telemetry interface.
UCs	Potentially all three UCs, depending on whether they have storage resources at the edge and the need to utilize them.

4.3 Decentralized and Distributed Data Manager

EMPYREAN Decentralised and Distributed Data Manager component is provided by ZSCALE and it is based on the Eclipse Zenoh[3] open source project. Eclipse Zenoh is a Pub/Sub/Query protocol that provides a set of unified abstractions to deal with data in motion, data at rest, and computations at the Internet scale. EMPYREAN leverages the capabilities of Zenoh, aiming to enhance real-time safety data exchange and communication between vehicle's Onboard Units (OBUs), robotics, and basically any component deployed across the edge and cloud infrastructure. Zenoh protocol is based on Named-Data Networking (NDN), where the data is represented using Unique Resource Identifiers (URIs), where a name and a value identify each resource, also known as the data model. Besides interoperability, resource naming has a key role in facilitating and optimizing, routing, and querying data. Thus, it is very simple to submit a distributed query that indicates the desired data.

[2] MinIO – High performance object storage: https://github.com/minio/minio

[3] https://github.com/eclipse-zenoh/zenoh

Figure 5 shows a data flow path for solving the distributed query use case in an automotive context by submitting a get("desired_data") function, with the desired data passed as a parameter. Zenoh supports the use of wildcards, single-start (i.e. *) to represent any given string char within the same level structure or double-start (i.e. **) to navigate within the same level and deeper levels of the data hierarchy.

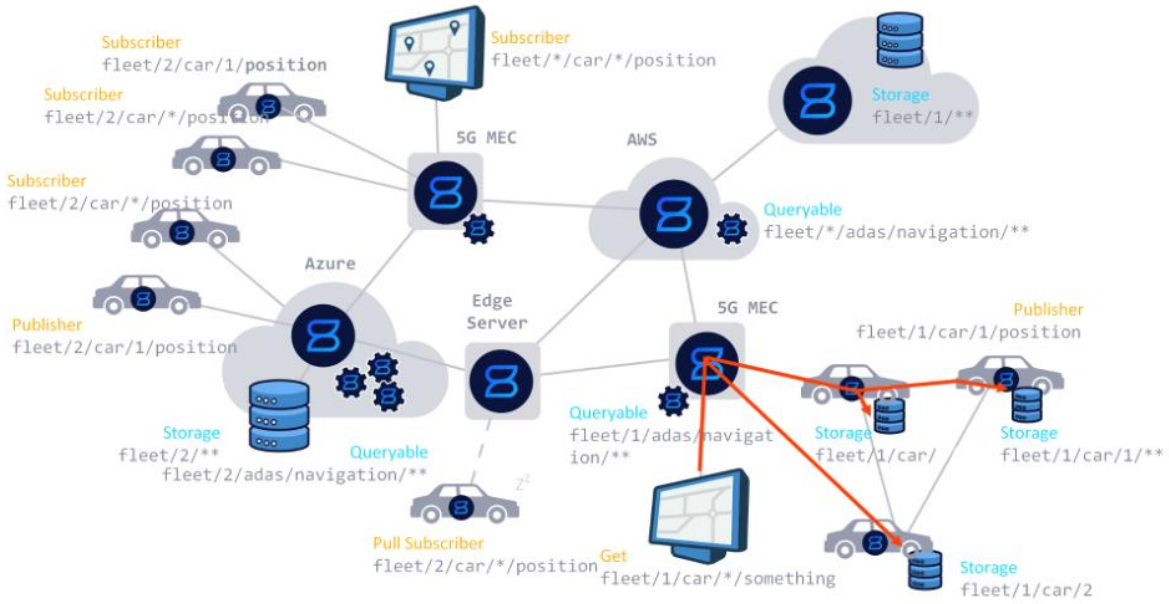

Figure 5: Distributed data query in automotive context

EMPYREAN Decentralised and Distributed Data Manager can run on the same virtual or physical computer, a separate computer on a heterogeneous network, or a remote computer at the edge or in the cloud. Zenoh's routing determines how to route such a request to where the data is. This is a very powerful mechanism when the user must collect data from multiple sources.

Adopting this communication middleware will enable EMPYREAN's components and user applications to exchange data across different communication technologies such as Ethernet, time-sensitive networking (TSN), Wi-Fi, and 4G/5G. It can also be used at different geographical locations, such as a Local Area Network (LAN), a metropolitan area network (MAN), and a Wide Area Network (WAN), and in various topology configurations such as peer-to-peer, mesh, brokered, and routed. Another benefit is Zenoh's integration with other messaging protocols such as Data Distribution Service (DDS), Message Queuing Telemetry Transport (MQTT), OPC Unified Architecture (OPC-UA), and its support for different database storages such as traditional SQL-based, non-SQL-based, time series storages (i.e. influxDB), in memory storage, filesystem storage, or cloud storages (i.e. S3).

Table 7: Description of Decentralized and Distributed Data Manager components

Component ID	WP3.2.3
Name	Decentralised and Distributed Data Manager
High level description	It manages data exchanges between any device in the network. It supports Pub/Sub, Push/Pull, distributed queries, and computations. This component also supports the creation of storage resources that can be mounted backend volumes and can thus be integrated into the storage system and queried on demand. It is an internal component that should be installed and configured by the application developers, and it can operate under different models, allowing it to run over any topology and anywhere across the continuum. It can run in (i) a peer-to-peer fashion way, allowing the creation of clique or mesh topologies; (ii) a brokered fashion way, where nodes have only a limited set of functionalities and rely on the network to provide the full Zenoh capabilities; (iii) routed fashion way, where nodes act as software routers that forward messages between nodes
Collaborators	● Dataflow Programming Component (WP4.2.5) ● EMPYREAN Aggregator (WP4.4.11)
Interfaces	Zenoh APIs are available for the most popular programming languages like RUST, Python, and REST. It also includes Zenoh-Pico, which was developed in C to support microcontrollers and embedded devices. The API instructions are fairly simple and support efficient publish/subscribe primitives, supporting multiple levels of reliability, dynamic discovery, fragmentation, and wire-level batching.
UCs	Potentially all three UCs, our technology is available and ready to be used. At the time of writing the deliverable, it has been adopted by the UC3, Robotic Semi-autonomous and Lights Out Logistics Order Picking.

4.4 Local Orchestration and Autoscaling Optimizations

Kubernetes (K8s) is the de-facto industry standard for cloud infrastructure resource management and orchestration, and it has also been adopted as the main low-level orchestration software for the edge-cloud continuum.

To bring intelligence to the low-level orchestrator in the edge-cloud continuum, we will develop AI/ML-driven mechanisms to enable the autonomous and adaptive workload autoscaling on the low-level Kubernetes platforms (Figure 6). A common workload autoscaling technique is horizontal autoscaling, which already exists in Kubernetes and allows applications to scale out or scale in the number of replicas. This powerful feature enables the system to automatically adapt its resource allocation based on actual traffic. However, if the limits are not set correctly, the average utilisation might grow the application in a non-optimal way. Instead, we could keep more resources powered down and gain a lot in the system's energy

consumption. Hence another technique to address the adaptation of workload is vertical auto scaling, which enables the automated setting of limits for each replica[4].

These EMPYREAN components focus on applying AI/ML techniques to vertical autoscaling (i.e., to implement an ML-based vertical autoscaler) in the Kubernetes cluster within the edge-cloud continuum, based on telemetry data collected from the Kubernetes platform. The goal is to recommend suitable container sizes for workloads, thus allowing better container bin-packing on nodes and the saving on total number of nodes. This work will also allow us to power off more resources, thus improving energy use.

In the edge-cloud continuum, besides traditional workloads that use CPU and RAM resources, the emerging hyper-distributed AI applications also use GPU resources. However, there is currently no mature technology to fraction GPU instances as the containers for CPU and RAM in Kubernetes. Therefore, a possible extension of this work is to research state of-the-art GPU fractioning methods, and apply vertical auto-scaling techniques to GPUs in the edge-cloud continuum.

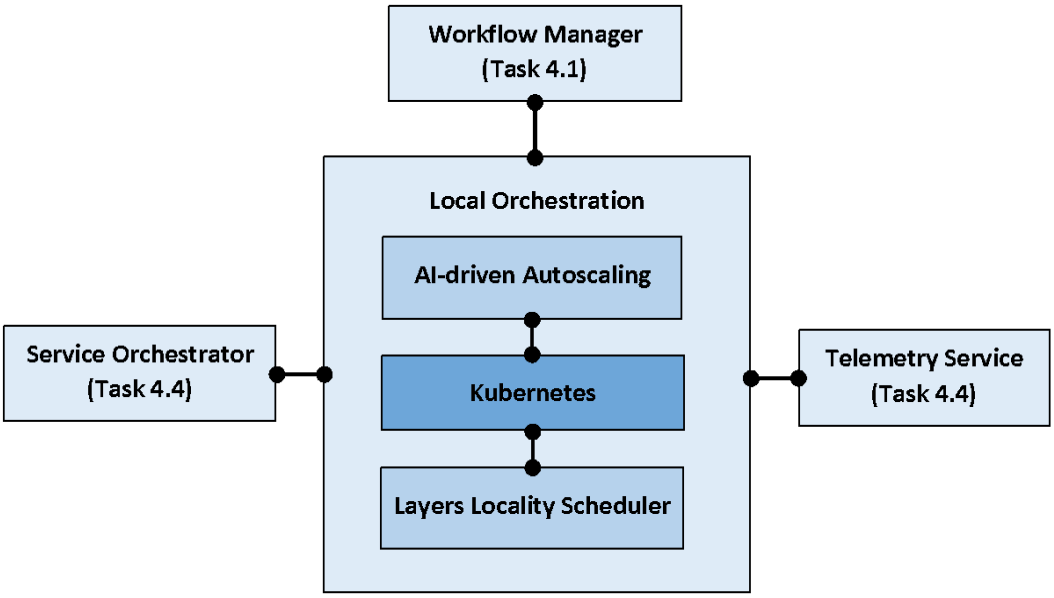

Figure 6: Local Orchestration and Autoscaling Optimizations dependencies

[4] Minh-Ngoc Tran, Dinh-Dai Vu, and Younghan Kim. A Survey of Autoscaling in Kuber netes. In 2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN), pages 263–265, Barcelona, Spain, July 2022. IEEE

Table 8: Description of Local Orchestration and Autoscaling Optimizations components

Component ID	WP3.4.1
Name	Autoscaling Optimizations
High level description	AI-enabled workload autoscaling mechanism that will be implemented based on Kubernetes orchestrator enhanced with AI/ML techniques for intelligent resource requests and limits allocation. In more detail, we will use AI/ML to perform optimal workload autoscaling by setting the most adequate resource limits configuration and performing dynamic adaptation based on historical data of previous executions.
Collaborators	• Workflow Manager (WP4.2.1) • Service Orchestrator (WP4.4.1) • Telemetry Service (WP4.4.7) • Kubernetes
Interfaces	The main interface to exchange information between the Autoscaling optimizations engine and the local orchestrator, the telemetry engine and the Workflow Manager will be based upon the standard Kubernetes API.
UCs	The autoscaling optimizations will be implemented to be lightweight and adaptable to different workloads. They will optimize the resources usage of local clusters. Hence, all 3 use cases will be possible to use them even those which enable Kubernetes clusters with low-power edge hardware.

Component ID	WP3.4.2
Name	Local Kubernetes Scheduler - Container Layers Locality Scheduler
High level description	Kubernetes scheduling algorithm to minimize the cold start delays favouring the placement of tasks on nodes that have more Layers of Containers related to the task to be deployed.
Collaborators	• Workflow Manager (WP4.2.1) • Kubernetes • Container Runtime Interface (CRI)
Interfaces	This scheduling plugin will mainly communicate with Kubernetes and the related Container Runtime Interface (CRI-O or Containerd) through the Kubernetes API.
UCs	All 3 use cases will have the ability to use the Layers Locality Scheduler.

4.5 Analytics Engine

Service assurance mechanisms for the self-driven adaptability of the IoT-edge-cloud continuum are essential for maintaining optimal performance, reliability, and efficiency across this complex and dynamic infrastructure. To achieve this, EMPYREAN aims to develop a highly automated and intelligent IoT-edge-cloud continuum, empowered by AI-enabled distributed management through its Service Assurance service. This will ensure optimal performance for deployed applications through autonomous adaptations over an infinite time horizon control loop.

EMPYREAN will integrate distributed service assurance mechanisms within each Association, utilizing real-time telemetry data and a robust set of algorithms within the Analytics Engine. This approach ensures that applications perform as intended while proactively or reactively triggering any necessary re-optimizations. The algorithms within the Analytics Engine will leverage continuous analysis techniques, such as machine learning, machine reasoning, swarm intelligence, and robust adaptive optimization, that will 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 7 shows the key building blocks of the EMPYREAN Analytics Engine and their interactions with other EMPYREAN components, while Table 9 provides an initial description of each component.

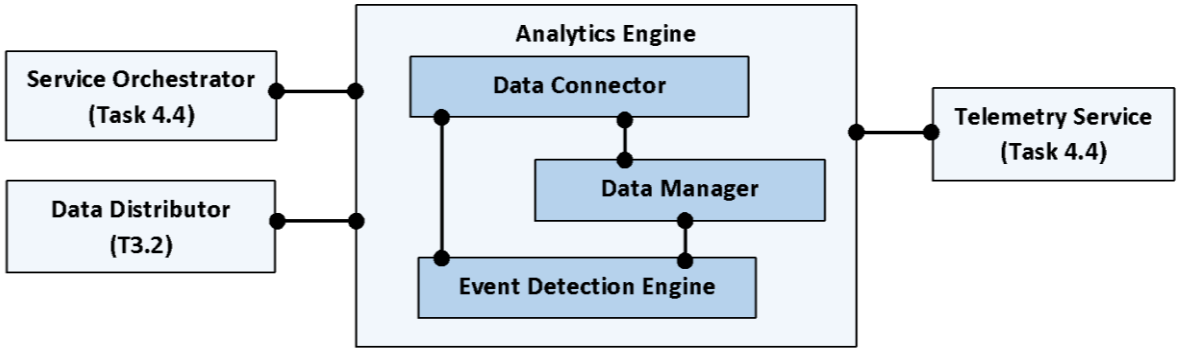

Figure 7: Analytics Engine core components and dependencies

Table 9: Description of Analytics Engine core components

Component ID	WP3.4.3
Name	Analytics Engine
High level description	It implements the service assurance mechanisms within the EMPYREAN platform to detect issues with the operation of the infrastructure resources and Associations along with the performance of the deployed applications. It will be designed as a distributed and scalable service to analyze the collected telemetry data in order to trigger pro-actively and re-actively dynamic adjustments.
Collaborators	• Service Orchestrator (WP4.4.1) • Telemetry Service (WP4.4.7) • Decentralized and Distributed Data Manager (WP3.2.3)
Interfaces	• REST • Asynchronous message-based
UCs	It will ensure that the applications perform as intended, while it will dynamically trigger the necessary adjustments if the current deployments do not comply with the requested SLA guarantees.

Component ID	WP3.4.4
Name	Data Connector
High level description	It will handle the collected monitoring data and apply several transformations to prepare them for the Event Detection Engine. It will support various data formats (e.g., JSON, CSV, raw) and will be able to fetch data directly by querying the monitoring solution (using a predefined interface) or consuming a stream directly from a queuing service.
Collaborators	• Event Detection Engine (WP3.4.6) • Data Manager (W3.4.5) • Telemetry Service (WP4.4.7) • Decentralized and Distributed Data Manager (WP3.2.3)
Interfaces	REST, MQTT, AMQP, and other types supported by open-source connectors.
UCs	It will facilitate the operation of the Analytics Engine and the provision of service assurance mechanisms within the EMPYREAN platform for all deployed applications.

Component ID	WP3.4.5
Name	Data Manager
High level description	It will provide local storage for process data, trained models, and results. It will also facilitate the exchange of events and data with other external components, such as the Data Distributor.
Collaborators	• Event Detection Engine (WP3.4.6) • Data Connector (WP3.4.4)
Interfaces	• REST • S3-compatible • Asynchronous message-based
UCs	It is an internal component that will facilitate the operation of the Analytics Engine.

Component ID	WP3.4.6
Name	Event Detection Engine
High level description	It will implement the core functionality of the EMPYREAN distributed service assurance mechanisms based on provided real-time telemetry data and appropriate machine reasoning techniques. It will facilitate the execution of the developed data-driven algorithms that will safeguard that deployed applications and available Associations perform as intended.
Collaborators	• Data Connector (WP3.4.4) • Data Manager (WP3.4.5) • Decentralized and Distributed Data Manager (WP3.2.3) • Service Orchestrator (WP4.4.1)
Interfaces	It will support both synchronous and asynchronous communication interfaces to facilitate the interaction with the internal components (e.g., Data Manager) and other EMPYREAN services (e.g., Data Distributor, Service Orchestrator).

UCs	It will provide service assurance and anomalous events detection capabilities, including but not limited to hardware resources, performance, applications. The provided analysis will be made available to service orchestration and deployment mechanisms to assist the autonomous and efficient operation of the overall EMPYREAN platform.

4.6 Cyber Threat Intelligence Engine

The increasing frequency and sophistication of cyber threats in recent years has kindled a dramatic surge in the amount of Cyber Threat Intelligence (CTI) available to the general public and companies. However, the sheer volume and diversity of this information have made comprehensive analysis of CTI a daunting task. The Cyber Threat Intelligence Engine (Figure 8) will try to address these issues by compiling and analysing CTI from various sources, including the Cyber Threat Alliance (CTA) and the Malware Information Sharing Platform (MISP), providing a user-friendly interface for security experts, and a REST API for integration with the EMPYREAN orchestration mechanisms.

In addition, the CTI Engine will leverage advanced machine learning and data mining algorithms to identify trends and patterns within the CTI data. These algorithms will be two-fold: On the one hand, algorithms based on large language models will be used to extract information from unstructured data sources. On the other hand, AI-based algorithms will be developed to learn relevant facts from the complex CTI knowledge graph database. By identifying emerging threats and trends, our system will enable proactive measures and informed decision-making. This analytical capability will be crucial in understanding the evolving tactics, techniques, and procedures of cyber adversaries, thereby enhancing the overall cybersecurity posture of organizations.

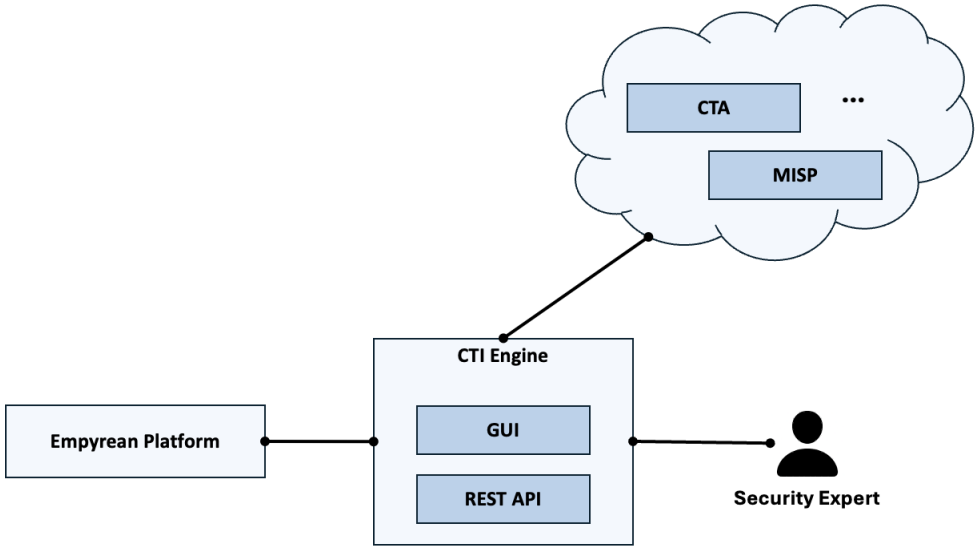

Figure 8: EMPYREAN Cyber Threat Intelligence Engine

Table 10: Description of Cyber Threat Intelligence

Component ID	WP4.1.1
Name	CTI Engine
High level description	Module that collects and analyses Cyber Threat Intelligence (CTI) from the Cyber Threat Alliance (CTA) repository to extract trends and important information. By integrating data from these prominent sources, the engine will compile a comprehensive repository of Indicators of Compromise (IoCs), 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, allowing security professionals to quickly search, filter, and visualise relevant threat information. The CTI Engine will also feature a REST API, enabling integration with existing systems and tools, facilitating the automation of threat intelligence workflows.
Collaborators	• Service Orchestrator (WP4.4.1) • Telemetry Service (WP4.4.7)
Interfaces	• GUI for security experts • REST API
UCs	The engine will contribute to Anomaly Detection in Robotic Machining Cells (UC1).

4.7 Decision Engine

The proliferation of hyper-distributed applications and the integration of distributed data processing across the IoT-edge-cloud continuum pose significant challenges in resource allocation, task placement, load balancing, and data management within the computing infrastructure. These challenges are further exacerbated by the dynamic and unpredictable nature of IoT, edge and cloud multi-technology environments, as well as the variability of workloads, resource availability, and infrastructure and application failures. Managing these complexities requires decentralized decision-making and autonomous adaptations to ensure efficient and reliable operation across this heterogeneous and evolving landscape.

EMPYREAN adopts a distributed speculative intelligent approach to orchestrate hyper-distributed applications, balancing centralized and decentralized solutions by developing novel algorithms. EMPYREAN will exploit multi-objective optimization, game theory, AI/ML techniques, and heuristics to design a set of algorithms aiming to provide different trade-offs between optimality and complexity. EMPYREAN will also incorporate environmental considerations to support intelligent, energy-aware workload and data distribution. Specifically, the developed algorithms will prioritize energy efficiency and adhere to relevant limits and thresholds, such as consumption, carbon emissions, and energy budgets, in the Associations' operations. EMPYREAN will also take into account the energy sources powering devices and resources (e.g., battery or renewable energy sources), their charge states, and their overall sustainability (e.g., quality of energy-saving technology), focusing on prioritizing green resources.

The Decision Engine will integrate these algorithms in the EMPYREAN platform, implementing the decide part at the envisioned closed-loop control based on the principles of observe, decide and act. It will provide to EMPYREAN Aggregator and Service Orchestrator the required intelligence (i) to support the efficient operation of Associations, (ii) to orchestrate the hyper-distributed applications and allocate their workloads considering the local resource state and characteristics while trying to fulfil their own objectives, and (iii) to coordinate the efficient load-balancing of data and workload within and across the available Associations.

The Decision Engine will be built upon the open-source Resource Optimization Toolkit (ROT) framework, initially developed by ICCS during the H2020 SERRANO[5] EU project. ROT facilitates the integration and execution of various decision-making algorithms. Within the EMPYREAN project, efforts will focus on extending the ROT into the cloud-native Decision Engine component, which includes the development and evaluation of distributed decision-making algorithms. In addition, EMPYREAN will enhance the Decision Engine to support sophisticated resource optimization in a cloud-native environment. By doing so, the Decision Engine will provide robust and efficient decisions for dynamic resource allocation, workload balancing, and energy-efficient operations across the Association-based IoT-edge-cloud continuum.

Figure 9 illustrates the key building blocks of the Decision Engine and its interactions with other EMPYREAN components, while Table 11 provides a high-level description of these components. Task 4.1 will provide the development of distributed and AI-enable decision-making algorithms, as well as the detailed design and implementation of the Decision Engine.

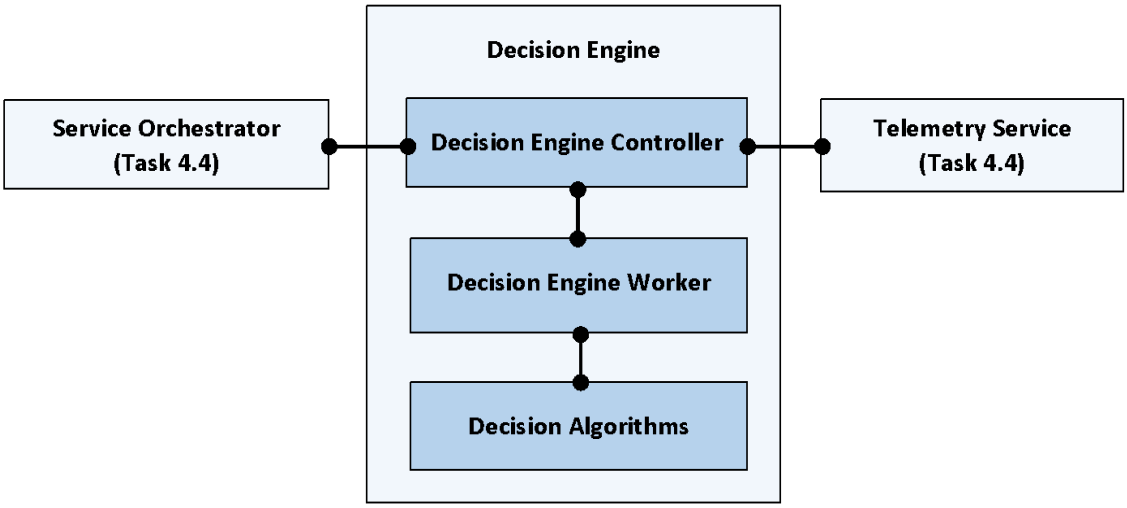

Figure 9: Decision Engine core components and dependencies

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

Table 11: Description of Decision Engine core components

Component ID	WP4.1.2
Name	Decision Engine
High level description	This component provides the decision-making functionality to the Service Orchestration. It prepares and manages the requested algorithm's execution to provide suggestions for application deployment across the available Associations.
Collaborators	• Decision Engine Controller (WP4.1.3) • Decision Engine Worker (WP4.1.4) • Decision Algorithms (WP4.1.5) • Service Orchestrator (WP4.4.1) • Telemetry Service (WP4.4.7)
Interfaces	• REST interface to request execution of algorithms and retrieve the results. • Asynchronous internal interface for interacting with the Decision Engine Workers based on the AMQP protocol. • Internal interface based on predefined JSON schema descriptions
UCs	It will provide the required high-level orchestration decisions to EMPYREAN Service to assign and re-optimize the UC applications' workloads.

Component ID	WP4.1.3
Name	Decision Engine Controller
High level description	It receives execution requests from the Service Orchestrator handles the interaction with the multiple Decision Engine Workers. It also interacts with the Telemetry Service to retrieve the required information.
Collaborators	• Service Orchestrator that requests the execution of some specific algorithms. • Telemetry Service that provides details for the state of infrastructure resources and available workloads. • Decision Engine Worker that serves the requests.
Interfaces	• REST interface to request execution of algorithms and retrieve the results. • Asynchronous internal interface for interacting with the Decision Engine Workers based on the AMQP protocol.
UCs	It will provide the required high-level orchestration decisions to EMPYREAN Service to assign and re-optimize the UC applications' workloads.

Component ID	WP4.1.4
Name	Decision Engine Worker
High level description	It implements the developed multi-objective optimization and orchestration algorithms. It receives requests, for starting or terminating algorithm execution, from the Decision Engine Controller and performs all the required actions, including preparing the execution environment, monitoring progress, and forwarding the results to Controller.
Collaborators	• Decision Engine Controller that orchestrates and handles the execution requests

Interfaces	• Asynchronous internal interface for interacting with the Decision Engine Controller based on the AMQP protocol.
UCs	It will provide the required high-level orchestration decisions to EMPYREAN Service to assign and re-optimize the UC applications' workloads.

Component ID	WP4.1.5
Name	Decision Algorithms
High level description	The library of multi-objective optimization and orchestration algorithms. The integrated algorithms will achieve different trade-offs between optimality and complexity to efficiently satisfy the heterogeneous and strict applications' requirements. There will be also algorithms to provide energy-aware workload and data balancing, both within and between Associations.
Collaborators	• Decision Engine Worker
Interfaces	• Internal interface based on predefined JSON schema descriptions
UCs	It will provide the required high-level orchestration decisions to EMPYREAN Service to assign and re-optimize the UC applications' workloads.

4.8 Workflow Manager

The Workflow Manager component enables the user to design and execute a data analytics application. In particular, it provides the means to create, deploy, update, execute, and monitor the execution of data processing applications in the form of workflows upon hybrid cloud, edge, and on-premises computing infrastructures. It allows users to create their data automations and expose them with APIs through fully customizable workflows using a low-code UI. The Workflow Manager in EMPYREAN (Figure 10) is based upon the open-source platform Ryax, which will be sufficiently enhanced to cover the needs of project. It uses a powerful custom runtime, abstracting completely the complexity of building and deploying containers with their dependencies upon edge-cloud infrastructures.

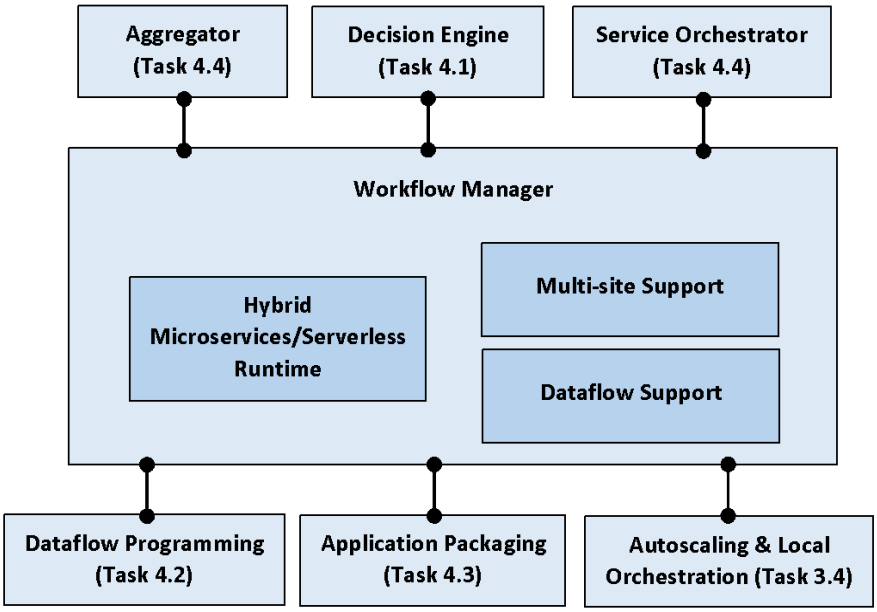

Figure 10: Workflow Manager components and dependencies

The engine will be enhanced to provide the right abstractions and internal mechanisms to efficiently support both long-duration microservices and short-duration serverless functions in such a way in order to cover the requirements of most data analytics and AI applications. It will use the Ryax abstractions, particularly the default YAML-based representation to define actions and workflows.

One of the principal features that is needed in a workflow manager for the edge-cloud continuum and which will be added on Ryax is the support of multiple sites, meaning the possibility to execute parts of a workflow on one cluster at the edge and parts on another cluster in the cloud. Besides the networking and storage, which need to be configured to enable the exchanges in a multi-infrastructure setting, the platform needs to provide the necessary hooks and abstractions to enable the user to add their hard constraints related to their choice of infrastructure to use for each part of the workflow, connect to the decision engine to perform the orchestration and control the selection by proposing the recommended choice to the Local orchestrator. In this context the workflow manager Ryax will be also enhanced to efficiently handle EMPYREAN Associations through its communication with the Aggregator.

Finally, another important optimization integrated within the Ryax Workflow Manager, is its support for a native dataflow programming framework such as ZenohFlow (Section 4.9). The idea is to provide fine-grain support for how data communications take place, which is not handled by default by most workflow managers like Ryax. The support of dataflow within Ryax will enable users with increased real-time capabilities, which is ideal for the IoT-based use cases of EMPYREAN.

Table 12: Description of Workflow Manager components

Component ID	WP4.2.1
Name	Ryax Workflow Manager Engine
High level description	Ryax open-source workflow engine enables the design, deployment and monitoring of workflows of data analytics upon Cloud, Edge, HPC infrastructures. It makes use of Kubernetes orchestration, and it provides a custom runtime environment for the deployment of components upon the related infrastructure. This will be enhanced to provide a tight integration with EMPYREAN, among others, through the support of associations and aggregators.
Collaborators	• Decision Engine (WP4.1.2) / Service Orchestrator (WP4.4.1) • Autoscaling Optimizations (WP3.4.1) & Local Orchestration (WP3.4.2) • Application Packaging (WP4.3.3) • Dataflow Programming Component (WP4.2.5)
Interfaces	Different interfaces are available to exchange with Ryax: a CLI and a Web Interface can be used by users while a REST-API can be used by the different services.
UCs	All 3 use cases will have the ability to use and benefit from the usage of the Ryax workflow engine.

Component ID	WP4.2.2
Name	Ryax Workflow Engine's Hybrid Microservices/Serverless Runtime
High level description	Ryax hybrid microservices/serverless runtime is developed in a way to allow users to specifically define through the YAML programming language and specific hooks and abstractions the definition of actions to be participating in a workflow as microservices or serverless functions. Currently in Ryax microservices can only be defined as triggers of a workflow while only serverless functions can be continuing in the middle or the end of the workflow logic. This will be enhanced to allow microservices and serverless functions to be executed in all levels of the workflows with no exceptions. This will provide a better support of the real-time nature of IoT-based applications.
Collaborators	• Autoscaling Optimizations (WP3.4.1) & Local Orchestration (WP3.4.2) • EMPYREAN Aggregator (WP4.4.11) • Application Packaging (WP4.3.3)
Interfaces	The Ryax API will be the main communication interface between the runtime and the other components.
UCs	All 3 use cases will have the ability to use and benefit from the usage of the Ryax hybrid microservices/serverless runtime.

Component ID	WP4.2.3
Name	Ryax workflow engine's multi-site support
High level description	Ryax multi-site support will allow users to execute their workflows seamlessly upon different infrastructures (e.g., cloud, edge, on-premise) by allowing them to select during deployment time the site, site characteristics or specific node-pool they need. The component will also provide the alternative to set execution objectives, letting the system to choose the optimal resource matching based on

	the application needs. This will be performed through the integration with the Decision Engine and the Service Orchestrator components. The component will provide a tight integration with EMPYREAN's associations and aggregator in order to enable the system to enable executions on one or multiple associations addressing the needs of EMPYREAN's standards through the aggregator.
Collaborators	• EMPYREAN Aggregator (WP4.4.11) • Decision Engine (WP4.1.2) / Service Orchestration (WP4.4.1) • Autoscaling Optimizations (WP3.4.1) / Local Orchestration (WP3.4.2)
Interfaces	The Ryax API will be the main communication interface between the multi-site sub-component and the other components and services.
UCs	All 3 use cases will have the ability to use and benefit from the usage of the Ryax workflow engine's multi-site support.

Component ID	WP4.2.4
Name	Ryax workflow engine's dataflow support
High level description	Ryax dataflow support will be brought through a tight integration with the Dataflow programming component based on Zenohflow. In particular the user will get the ability to express the ways data should flow between the different actions of the workflows, besides expressing which inputs and outputs should be exchanged. In particular, the real-time communication from the different multi-infrastructures will be handled by the dataflow programming component.
Collaborators	• Dataflow Programming Component (WP4.2.5) • Networking and Storage components
Interfaces	The Ryax API will be the main communication interface between the dataflow sub-component and the other components and services.
UCs	All 3 use cases will have the ability to use and benefit from the usage of the Ryax workflow engine's dataflow support.

4.9 Dataflow Programming Component

EMPYREAN Dataflow Programming Component is provided by ZSCALE and it is based on the Eclipse ZenohFlow[6] open-source project. ZenohFlow component facilitates the development and deployment of data-intensive applications across the IoT-edge-cloud continuum. It consists of a set of nodes interconnected with links. Thus, an application can be represented as a Directed Acyclic Graph (DAG)[7] . These graphs are described in human-readable descriptor files that enforce by contract all the communications and possible data exchanges. Starting from the base descriptor file, ZenohFlow instantiates the application components' placement across the available infrastructure. This declarative approach simplifies the development of complex and constraint applications, as the developer simply needs to (a) create the different

[6] https://github.com/eclipse-zenoh-flow/zenoh-flow

[7] https://en.wikipedia.org/wiki/Directed_acyclic_graph

nodes (i.e., source, operators, sinks) that compose the application, and (b) describe the connection between them (i.e., data-type, timeout).

As illustrated in Figure 11, an application can consist of a source node (A), three operator nodes (B, C, and D), and a sink node (E). ZenohFlow utilizes Eclipse Zenoh (as described in Section 4.3) as its communication framework. This allows application developers using ZenohFlow to deploy their application without needing prior knowledge of where the individual nodes will be running. Zenoh seamlessly handles data routing based on key expressions, ensuring efficient communication between nodes.

EMPYREAN will benefit from this technology. ZenohFlow assigns each instance of an application a unique identifier, which is automatically incorporated into the communication process. This ensures that even if the same application is deployed multiple times on the same infrastructure, or if several applications share the same resources (i.e., topics), there are no collisions or conflicts. Additionally, the same technique is used for each link, allowing multiple nodes to safely expose the same key expressions without interference.

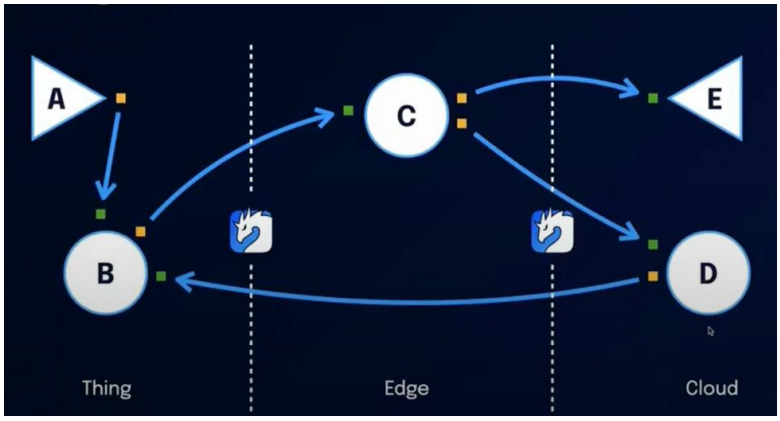

Figure 11: Representation of a distributed dataflow programming using ZenohFlow

Table 13: Description of Dataflow programming component

Component ID	WP4.2.5
Name	Dataflow Programming Component
High level description	It provides a declarative approach for developers to describe the application's structure precisely, what will be deployed, and how it will be connected. Its human-readable format has the advantage of lowering the entry barrier for purely technical application designers.
Collaborators	• Decentralised and Distributed Data Manager (WP3.2.3) • EMPYREAN Aggregator (WP4.4.11)
Interfaces	ZenohFlow is built on top of Eclipse Zenoh, it supports all of Zenoh interfaces and pub/sub mechanisms.
UCs	Potentially all three UCs, as previous usage in an Autonomous Driving System[8] illustrates its capabilities, allowing for real-time control.

[8] https://www.youtube.com/watch?v=QajVWYshaHk&t=11s

4.10 Lightweight Application Packaging

This particular component is essential for the application design process. It allows the system to be modular and flexible to cover the definition of various types of applications since it enables users to build their logic in small microservices or serverless functions in whichever programming language they prefer while being agnostic to different types of architectures (x86, ARM). Since each environment is built as a standard OCI container, this allows workflows to be completely polyglot in a completely seamless and infrastructure-agnostic way.

The NIX-based Environment Packaging component is based on the NIX functional package manager, providing a declarative way to build reproducible and lightweight environments. The technique used to build the containers takes particular care in packaging the environments in different structured layers, which can be an essential advantage during deployment. In the context of EMPYREAN, this component will be enhanced to support unikernels and Web Assembly in order to improve the security and overhead of typical container creation upon different types of architectures.

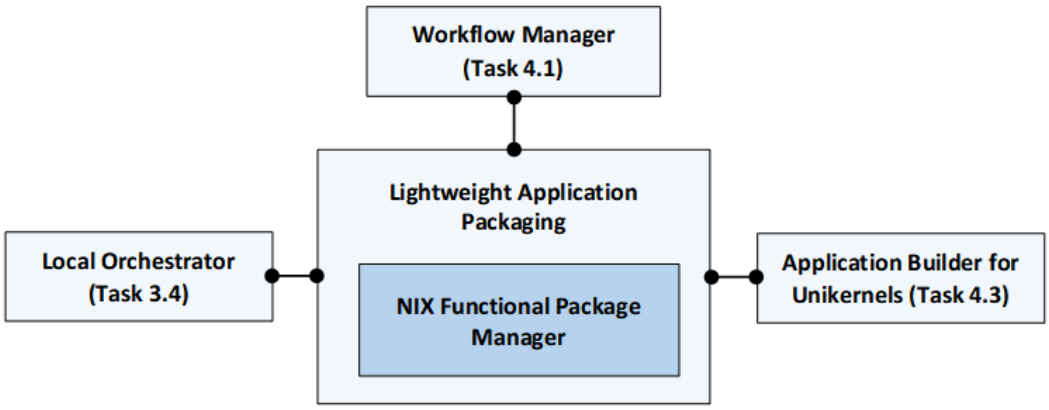

Figure 12: NIX-based Environment Packaging components and dependencies

Table 14: Description of NIX-based Environment Packaging

Component ID	WP4.3.1
Name	NIX-based Environment Packaging
High level description	Ryax workflow engine offers internally the mechanism of performing multi-arch polyglot environment packaging to build the components to be used within the workflows. This will be enhanced to support Web Assembly and unikernels.
Collaborators	• Workflow Manager (WP4.2.1) • Application Builder for Unikernels (WP4.3.2) • Local Orchestrator (WP3.4.2)
Interfaces	The main interface for this component will be its REST API. The communication with the Local Orchestrator will be performed through the Kubernetes API.
UCs	All 3 use cases will have the ability to use and benefit from the usage of the Lightweight Application Packaging.

4.11 Application Builder for Unikernels

EMPYREAN targets an end-to-end software stack for application deployment based on unikernels. EMPYREAN develops Bunny, a set of systems software components that enable deploying applications as unikernels in cloud-native environments. Bunny is designed to streamline the process of creating, packaging, and deploying custom-made, single-tenant-based applications. This toolchain (Figure 13) simplifies the development of highly efficient and secure applications by automating the creation of unikernels or simple binaries and the packaging process. Unikernels, being lightweight and specialized operating systems that run a single application, offer improved performance and a reduced attack surface, making them particularly suitable for IoT, edge, and cloud environments.

Figure 13: High-level overview of the Bunny workflow

Using Bunny as the underlying framework, the Application Builder can seamlessly integrate the compilation of application code into a self-contained binary, or a unikernel, abstracting away the complexity of manually configuring the operating system components. It provides a cloud-native approach to packaging applications, ensuring that the binaries generated are portable and can be deployed across the IoT-edge-cloud continuum. This packaging tool (Figure 13) automates dependencies, builds, and configurations, allowing developers to focus on application logic while the system ensures that the application runs efficiently within a unikernel. This approach enhances security and performance, particularly in environments where resource constraints and security are critical concerns.

Coupled with a custom container runtime (i.e., urunc[9]), able to spawn unikernels (or pre-compiled, static binaries) packaged as OCI images, EMPYREAN fully embraces the cloud-native concept.

[9]urunc - a container runtime for unikernels: https://github.com/nubificus/urunc

Table 15: Description of Application Builder for Unikernels components

Component ID	WP4.3.2
Name	Application Builder for Unikernels
High level description	This component addresses the deployment of applications in cloud-native environments using unikernels. It aims to tackle two major challenges associated with unikernels: (i) simplifying the building and deployment process and (ii) minimizing the engineering overhead to resolve external software dependencies.
Collaborators	● Application Packaging (WP4.3.3) ● Container Runtime (WP4.3.4) ● EMPYREAN Registry (WP4.4.13) ● Workflow Manager (WP4.2.1)
Interfaces	Inputs: ● application description ● source/binary repository of the application Outputs: ● binary artifact (unikernel, bootable using a hypervisor, or on bare metal)
UCs	At least one use case will use this component as part of the EMPYREAN development and deployment stack.

Component ID	WP4.3.3
Name	Application Packaging
High level description	This component is designed to streamline the application packaging process across diverse computing environments, focusing on creating OCI-compatible container images. It aims to bundle binary artifacts along with their descriptors into OCI container images, facilitating deployment across EMPYREAN's supported execution modes, including containers, sandboxed containers, WebAssembly (WASM), unikernels, and binary blobs for IoT devices. This development is crucial for EMPYREAN's overarching goal of enabling seamless application deployment and execution across heterogeneous hardware architectures and environments, enhancing interoperability, and ensuring efficient, cloud-native deployment methodologies.
Collaborators	● Container Runtime (WP4.3.4) ● NIX-based Environment Packaging (WP4.3.1)
Interfaces	Inputs: ● application description (Dockerfile-like) ● source/binary repository of the application (combined with Application Builder component), or unikernel binary Outputs: ● OCI artifact bootable
UCs	At least one use case will use this component as part of the EMPYREAN development and deployment stack.

Component ID	WP4.3.4
Name	Container Runtime
High level description	The component within the EMPYREAN project aims at facilitating the deployment of applications across various execution environments, including unikernels and IoT devices. This component is based on urunc, a runtime capable of spawning unikernels and seamlessly integrating them with generic container runtimes compatible with Kubernetes and serverless architectures. This component allows for the execution of applications built with the "Application Builder" component within the existing container orchestration ecosystems, providing the benefits of diverse building systems (e.g., unikernels for improved security and performance) while maintaining compatibility with widespread deployment models.
Collaborators	• Application Packaging (WP4.3.4) • NIX-based Environment Packaging (WP4.3.1) • EMPYREAN Controller (WP4.4.4)
Interfaces	Inputs: • OCI artifact • metadata Outputs: • successful execution of the binary artifact
UCs	At least one use case will use this component as part of the EMPYREAN development and deployment stack.

4.12 Analytics-friendly Distributed Storage

EMPYREAN will offer a novel way to store and retrieve IoT data, distributed across edge and cloud locations. By employing erasure coding and compression, storage costs are minimized. By structuring data in a highly organized schema that provides different alignment characteristics, queries can be evaluated directly on the erasure coded, compressed representation. Without this approach, network egress costs charged by cloud providers would make such a system unfeasible.

The IoT Query Engine will work either in close collaboration or as a submodule of the Edge Storage Gateway. It will hide the complexities of data storage and retrieval behind simple, easy to use interfaces.

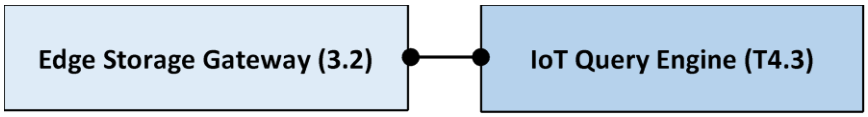

Figure 14: Analytics-friendly Distributed Storage

Table 16: Description of Analytics-friendly Distributed Storage components

Component ID	WP4.3.5
Name	IoT Query Engine
High level description	This component is in charge or providing an IoT analytics-friendly distributed storage solution. IoT time series data is ingested, processed and stored in an erasure-coded manner, distributed to both cloud and edge locations. This is key to offering cost-efficient, highly reliable and available data storage. To be able to run analytics workloads efficiently, the component uses a novel data rearrangement and erasure coding schema. The goal is to avoid having to reconstruct complete data fragments when evaluating queries. Our solution will make it possible to access individual bytes of data files with very little to no overhead, while maintaining the benefits of erasure coding and allowing the use of the whole continuum for distributed data storage. This feature will be accessed through the Edge Storage Gateway. Data ingest will include a description of the time series data next to the actual values. Data egress will happen through the IoT Query interface. Users will be able specify using an SQL-like syntax the parts of the time series data they are interested in (e.g. SELECT 'temperature' FROM 'robot3_telemetry' WHERE 'cell_voltage_1' < 3.0 AND `timestamp` > 1712828555). To further improve storage costs, we might be able to also support compression through a novel technique called Generalized Deduplication.
Collaborators	• Edge Storage Gateway (WP3.2.1)
Interfaces	It likely exposes some binary interface (e.g. direct TCP connection with protobuf, ZeroMQ) to the Edge Storage Gateway.
UCs	Hopefully at least one or two UCs.

4.13 Service Orchestrator and EMPYREAN Controller

EMPYREAN is dedicated to achieving the autonomous operation of the IoT-edge-cloud continuum through an advanced cognitive platform that integrates decentralized decision-making and self-adaptive capabilities. To realize this vision, EMPYREAN utilizes a distributed speculative intelligence approach for orchestrating hyper-distributed applications, striking an optimal balance between centralized and decentralized solutions. This methodology empowers more localized decision-making while maintaining a collective logic that ensures overall system-wide optimality. Consequently, the continuum will be well-equipped to support the future of hyper-distributed, dynamic applications and the rapid expansion of IoT.

The orchestration process within EMPYREAN involves two primary stages, engaging two key components of the platform: the Service Orchestrator and the EMPYREAN Controller. The platform features multiple high-level orchestrators, the Service Orchestrators, operating at the Association level, in conjunction with various Local Orchestrators managing the individual edge and cloud platforms unified within the EMPYREAN framework. This orchestration model ensures efficient and intelligent management of resources and tasks across the entire IoT-edge-cloud continuum.

In the first stage, multiple Service Orchestrators acts as cognitive agents competing based on their local knowledge for the efficient and rapid mapping of applications' workloads. Thus, individual parts of the overall application can be assigned to resources in an Association or in different Associations according to the deployment requirements and available infrastructure resources. In the second stage, each Service Orchestrator intelligently assigns the part of the overall workflow that is responsible to the Associations it manages. For this operation, EMPYREAN adopts a hierarchical orchestration approach, with high-level decisions taken, by the Service Orchestrator, on the Association layer, and low-level scheduling (i.e., actual assignment of workload to specific infrastructure resources) performed by each platform's (e.g., K8s, K3s) orchestration mechanisms (i.e., Local Orchestrator) to provide guarantees to the platform specifications. This approach provides several degrees of freedom to Local Orchestrator for serving in an optimal manner a job request, satisfying both the Aggregator and the resource's objectives. Moreover, the EMPYREAN Controller abstracts the management and interaction with the individual devices, resources, and services. Its modular design allows for seamless integration with resource-specific services, effectively handling orchestration, deployment, and management requests from the Service Orchestrator.

Both the Service Orchestrator and the EMPYREAN Controller will build upon the Resource Orchestrator service developed by ICCS during the H2020 SERRANO EU project. This high-level orchestrator operates seamlessly across diverse cloud and HPC platforms. In EMPYREAN, the original design and implementation of the Resource Orchestrator will be extended and enhanced to support decentralized and cooperative operations. This will support the efficient orchestration of edge-cloud resources and dynamic application deployment across the Association-based continuum.

The core building blocks of the EMPYREAN Service Orchestrator and Controller along with their interactions with other EMPYREAN components are presented in Figure 15, and Table 17 provides their high-level description.

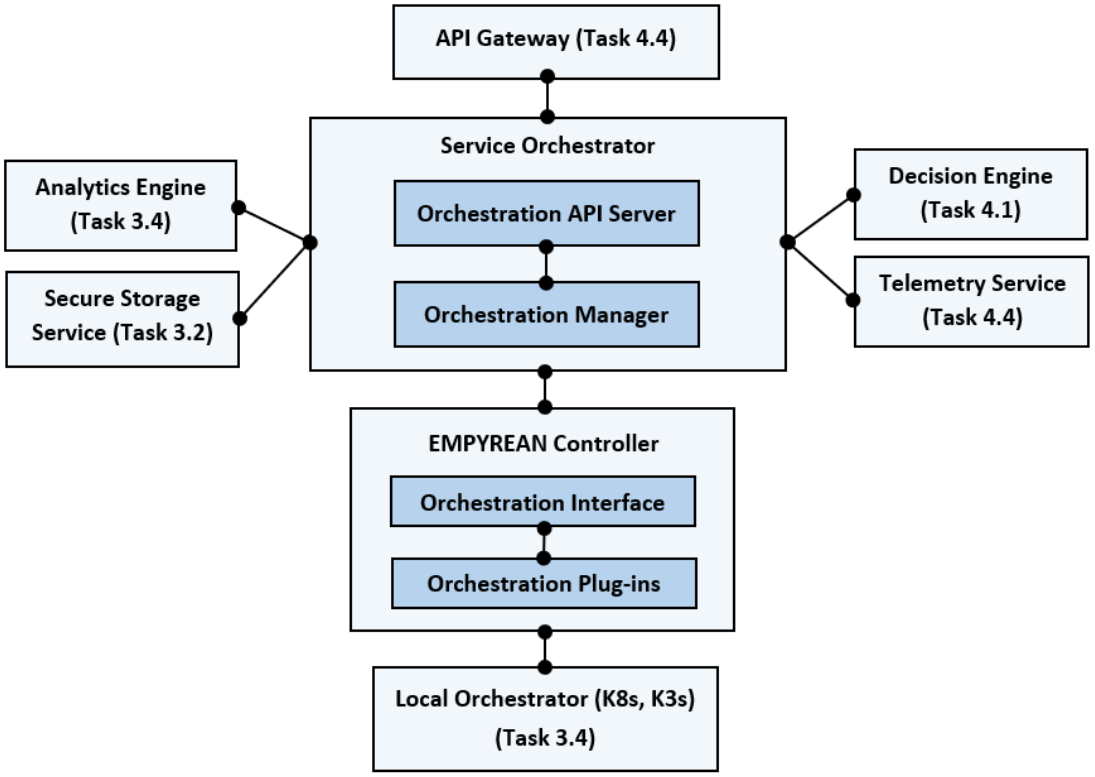

Figure 15: Service Orchestrator and EMPYREAN Controller components and dependencies

Table 17: Description of Service Orchestrator and EMPYREAN Controller core components

Component ID	WP4.4.1
Name	Service Orchestrator
High level description	It provides efficient service orchestration and resource management in the disaggregated and heterogeneous EMPYREAN infrastructure. It initiates the application deployment and automatically coordinates the necessary supplemental actions (e.g., transfer of required data).
Collaborators	• API Gateway (WP4.4.12) • Decision Engine (WP4.1.2) • Telemetry Service (WP4.4.7) • EMPYREAN Controller (WP4.4.4) • Analytics Engine (WP3.4.3)
Interfaces	• REST interface • Asynchronous message-based interface for notifications
UCs	It will coordinate the resource allocation and workload deployment procedures derived by analysing the use case preferences.

Component ID	WP4.4.2
Name	Orchestration API Server
High level description	It acts as the single-entry point for the other components to EMPYREAN service orchestration functionalities. Initially it validates the requests and then forwards them to the Orchestration Manager
Collaborators	• Orchestration Manager (WP4.4.3) • API Gateway (WP4.4.12) • Analytics Engine (WP3.4.3)
Interfaces	• REST interface • Asynchronous message-based interface for notifications
UCs	It will trigger the necessary internal procedures for the appropriate resource allocation and initial deployment of workloads based on UCs preferences.

Component ID	WP4.4.3
Name	Orchestration Manager
High level description	It implements the application logic, oversees the operation of the other internal components and coordinates the resource allocation and application deployment operations. It also interacts with the Decision Engine and Telemetry Service components.
Collaborators	• EMPYREAN Controller/Orchestration Driver (WP4.4.4) • Decision Engine (WP4.1.2) • Telemetry Service (WP4.4.7)
Interfaces	• REST interface
UCs	It will setup and coordinate the application deployment at the selected individual edge and cloud platforms.

Component ID	WP4.4.4
Name	EMPYREAN Controller / Orchestration Driver
High level description	It provides an abstraction layer for interacting with the specific edge and cloud orchestration mechanisms at each EMPYREAN location. It receives by the Service Orchestrator requests for deploying or adjusting already deployed applications.
Collaborators	• Orchestration Manager (WP4.4.3)
Interfaces	• REST interface
UCs	It will handle the requests from the Service Orchestrator regarding the initial deployment and re-optimization of submitted applications.

Component ID	WP4.4.5
Name	Orchestration Interface
High level description	It will provide at the Orchestration Manager and Orchestration Driver an infrastructure agnostic interface for descripting the deployment description and constraints to the heterogeneous local orchestration mechanisms.
Collaborators	• Orchestration Manager (WP4.4.3)

Interfaces	• REST interface
UCs	It will manage the seamless deployment and execution of submitted applications' workloads at the specific part of the EMPYREAN infrastructure.

Component ID	WP4.4.6
Name	Orchestration Plug-ins
High level description	It will map the generic instructions from the Orchestration Interface to specific actions and procedures for the selected local orchestration mechanisms. The interaction will be based on the APIs exposed by each local orchestration component. There will be a specific plug-in for each supported local orchestration mechanism.
Collaborators	• Orchestration Interface (WP4.4.5) • Local Orchestrator (e.g., K8s, K3s)
Interfaces	• REST interface
UCs	It will handle the actual seamless deployment and execution of applications' workloads at the specific part of the EMPYREAN infrastructure.

4.14 Telemetry Service

As machine learning continues to drive advancements in continuum automation, the significance of telemetry and observability across federated IoT-edge-cloud platforms has become critical. Traditional localized monitoring methods are no longer sufficient to provide the essential data required to ensure that all platform services operate in harmony, delivering optimal performance, security, and efficiency throughout the entire system.

The EMPYREAN telemetry service is designed to address these challenges by incorporating components that deliver robust observability and telemetry capabilities within the Associations. Observability allows us to gain insights into a system from the outside, enabling us to ask questions about its behaviour and performance without needing detailed knowledge of its internal workings. This capability is crucial for effective troubleshooting and resolving new, unforeseen issues, helping to answer the fundamental question, "Why is this happening?". Telemetry, on the other hand, involves the real-time collection, measurement, and transmission of data related to a system's performance, status, and behaviour. This data encompasses a wide range of metrics, including CPU usage, memory consumption, storage capacity, and network traffic, providing a comprehensive view of the system's health and operational state.

Together, observability and telemetry form the backbone of effective monitoring and management in the EMPYREAN platform, ensuring that the complex, interconnected services across IoT, edge, and cloud environments function seamlessly and respond dynamically to evolving conditions. Furthermore, they facilitate additional key operations such as data-driven decision making, enhanced security, automation and scalability, optimized resources utilization, and end-to-end visibility.

The EMPYREAN platform employs a robust and distributed telemetry infrastructure that is built around several key components, including several *Telemetry Engines*, *Monitoring Probes*, and a *Persistent Monitoring Data Storage* (PMDS) system. Telemetry Engines serve as the backbone of the EMPYREAN telemetry infrastructure. Each Telemetry Engine represents a generic monitoring entity tasked with managing, collecting, and analyzing telemetry data from a specific segment of the infrastructure. These engines operate independently yet cohesively to provide a detailed and unified view of system performance and health. The Monitoring Probes are specialized components, each dedicated to collecting telemetry data from a specific resource type. They collect real-time performance data related to various aspects of the infrastructure and deployed applications. To support long-term analysis and decision-making, the telemetry service integrates the Persistent Monitoring Data Storage (PMDS) system. The PMDS serves as a centralized repository for all collected telemetry data, storing it in a timestamped format that facilitates historical analysis. This long-term data storage is crucial for enabling advanced data analytics and supporting the EMPYREAN decision-making processes, as it provides the necessary historical context to identify trends, predict future issues, and optimize resource allocation.

Figure 16 illustrates the main building blocks of the EMPYREAN Telemetry Service, highlighting how these components interact with other elements of the EMPYREAN platform. Table 18 provides a high-level description of each component, detailing their roles to the overall telemetry infrastructure.

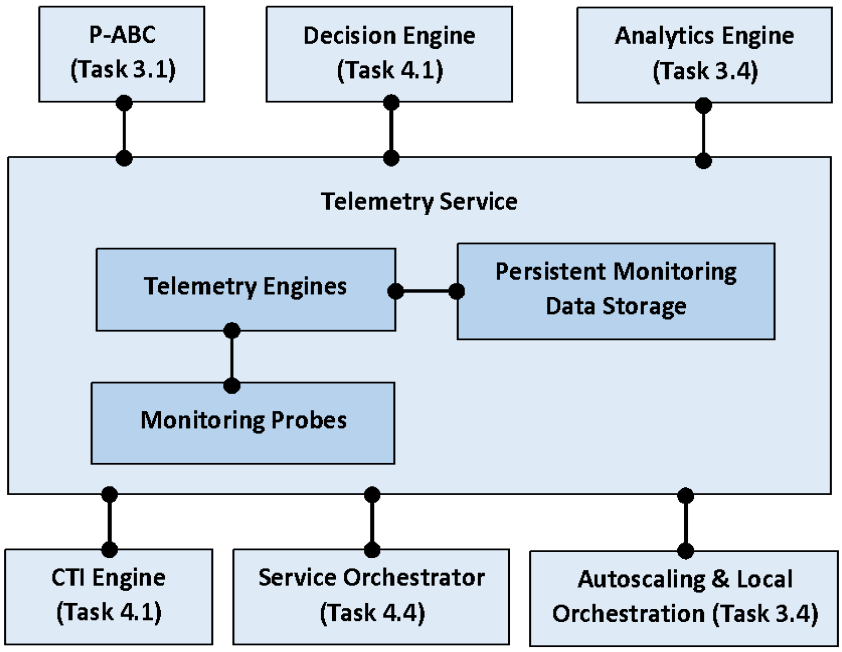

Figure 16: EMPYREAN Telemetry Service components and dependencies

Table 18: Description of Telemetry Service core components

Component ID	WP4.4.7
Name	Telemetry Service
High level description	It provides the EMPYREAN distributed telemetry service, maintaining an overall view of the state of the infrastructure resources and deployed applications. The collected data will be supplied to the EMPYREAN orchestration, service assurance, and security analysis mechanisms to enable efficient orchestration, improve resilience, and detect threats and anomalies. It will also provide alerts and notifications whenever abnormal values are detected to trigger the appropriate actions. Finally, a dashboard based on web-based visualization tools will enable quick access to real-time updates on the platform status.
Collaborators	• Telemetry Engine (WP4.4.8) • Monitoring Probes (WP4.4.10) • Persistent Monitoring Data Storage (WP4.4.9) • p-ABC (WP3.1.2) • Analytics Engine (WP3.4.3) • CTI Engine (WP4.1.1) • Decision Engine (WP4.1.2) • Service Orchestrator (WP4.4.1) • Autoscaling Optimizations (WP3.4.1) / Local Orchestrator (WP3.4.2)
Interfaces	• REST interface
UCs	It will provide the required information for the EMPYREAN orchestration and decision-making mechanisms. Thus, it will contribute to UC applications' cognitive orchestration and deployment in the EMPYREAN platform.

Component ID	WP4.4.8
Name	Telemetry Engine
High level description	It coordinates the operation of a specific set of Monitoring Probes that are responsible for monitoring the underlying platform resources and deployed applications. The implementation will be based on well-established open-source solutions such as Prometheus and Grafana. Prometheus will facilitate the data collection through its reconfigurable pull-based model, data storage as time series, provision of alerts based on specific rules, and automatic service discovery within K8s and K3s platforms. Grafana will be used to visualize and analyze the available telemetry data. It will also enable querying, visualizing, alerting, and exploring metrics, logs, and traces. EMPYREAN will implement the required extensions and improvements to glue together the different components.
Collaborators	• Telemetry Service (WP4.4.7) • Persistent Monitoring Data Storage (WP4.4.9) • Monitoring Probes (WP4.4.10) • p-ABC (WP3.1.2)
Interfaces	• REST interface • Web application, based on open-source Grafana

UCs	It will collect, store, and forward to Persistent Monitoring Data details about the characteristics and current status of available resources and deployed services under its administration domain.

Component ID	WP4.4.9
Name	Persistent Monitoring Data Storage
High level description	It provides a central repository to retain historical telemetry data for the current state of the heterogeneous resources and deployed applications. It is a fundamental piece of EMPYREAN's effective orchestration pipeline that feeds the various data-driven decision mechanisms within the EMPYREAN platform. The implementation will be based on InfluxDB[10], an open-source time-series database that provides fast, highly available storage for time-series data and can also be used as a data source for many other solutions, such as the Grafana.
Collaborators	• Telemetry Service (WP4.4.7) • Telemetry Engine (WP4.4.8)
Interfaces	• REST interface
UCs	It will provide historical telemetry data to enable their cognitive orchestration and re-optimization.

Component ID	WP4.4.10
Name	Monitoring Probes
High level description	This component is responsible for the actual monitoring of resources and deployed applications. It will provide the necessary telemetry data to the telemetry framework. Some probes will be out of the shelf, while others will be implemented in the context of EMPYREAN. By focusing on individual resource types, Monitoring Probes ensure that telemetry data is both precise and relevant, allowing for targeted monitoring and analysis.
Collaborators	• Telemetry Engine (WP4.4.8)
Interfaces	• REST interface
UCs	It will collect and forward to telemetry components information about the characteristics and current status of available resources and deployed services.

4.15 EMPYREAN Aggregator

The management fabric of the EMPYREAN continuum will be implemented through EMPYREAN Aggregators. Multiple self-managed and interacting Aggregators, that cooperate in a multi-agent manner, transforms the IoT-edge-cloud continuum into an autonomous, collaborative, composable, and self-organized ecosystem. These Aggregators are pivotal entities responsible for the seamless and dynamic creation, maintenance, and management of Associations. Operating autonomously and in a distributed manner, Aggregators employ an internal hierarchical two-level system to effectively oversee the resources within an

[10] InfluxDB: Open Source Time Series Database: https://www.influxdata.com/developers/

Association. To establish a robust and collaborative management fabric, Aggregators will communicate not only among themselves but also with edge computing infrastructures and (multi) cloud providers. This interconnectivity ensures a comprehensive and cohesive management framework across the continuum.

Moreover, the EMPYREAN Aggregator will facilitate the Associations capability to function independently. In scenarios where connectivity to remote cloud resources is either unfeasible or undesirable, Associations can maintain operational continuity without relying on external cloud connectivity. This independence underscores the flexibility and resilience of the EMPYREAN continuum's management architecture, enabling it to adapt to a variety of network conditions and operational requirements.

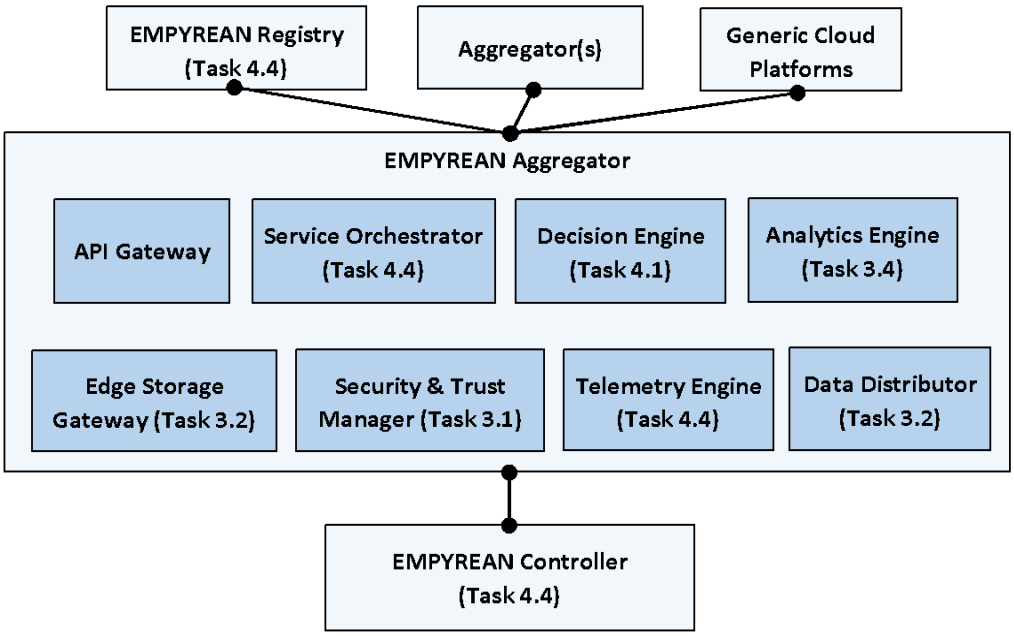

Figure 17: EMPYREAN Aggregator core components and dependencies

An Aggregator is a logical component that consolidates multiple components and services to deliver the necessary intelligence and orchestration logic for managing an Association. It also facilitates the deployment of applications, ensures secure and trusted execution of workloads, and oversees data storage across Associations within the continuum. These include functionalities, such as resource and workload orchestration (*Service Orchestrator*), intelligent decision-making (*Decision Engine)*, distributed, hybrid and encrypted data storage (*Edge Storage Gateway*), decentralized interconnection and seamless data distribution (*Data Distributor*), distributed trust and identity management (*Security and Privacy Manager*), monitoring of heterogeneous resources and deployed applications (*Telemetry Engine*), and service assurance mechanisms (*Analytics Engine*). These components are platform-agnostic, leveraging open and standardized APIs and frameworks to operate independently of the underlying platforms in the lower layers.

Figure 17 illustrates the key building blocks of the EMPYREAN Aggregator and its interactions with other EMPYREAN components. Table 19 provides a high-level description of its

components that are not described in the previous sections. The EMPYREAN Aggregator design promotes the composability of infrastructures and services across the continuum.

Table 19: Description of EMPYREAN Aggregator core components

Component ID	WP4.4.11
Name	EMPYREAN Aggregator
High level description	It manages and coordinates the operation of an EMPYREAN Association. Each Aggregator includes several core services that provide the required intelligence and orchestration logic to operation an Association, deploy workloads, and manage data access and storage. An Aggregator orchestrates its own Associations that include separate or shared computational and storage resources.
Collaborators	• API Gateway (WP4.4.12) • EMPYREAN Registry (WP4.4.13) • Service Orchestrator (WP4.4.1) • Security and Trust Manager (W3.1.1) • Decentralize and Distributed Data Manager (WP3.2.3) • Telemetry Engine (WP4.4.8) • Decision Engine (WP4.1.2) • Edge Storage Gateway (WP3.2.1) • Analytics Engine (WP3.4.3) • EMPYREAN Controller (WP4.4.4) • Other EMPYREAN Aggregators • Generic cloud platforms
Interfaces	• REST interface • Asynchronous interface for notifications and events
UCs	It will provide and manage the Association-based IoT-edge-cloud continuum to ensure the performance, security and energy efficiency for the use cases workloads.

Component ID	WP4.4.12
Name	API Gateway
High level description	It enables communication between Aggregators, general edge or (multi-) cloud providers. It abstracts through its open and well-defined interfaces the interactions between users and EMPYREAN's core services.
Collaborators	• EMPYREAN Registry (WP4.4.13) • Service Orchestrator (WP4.4.1) • Security and Trust Manager (WP3.1.1) • Decentralized and Distributed Data Manager (WP3.2.3) • Telemetry Service (WP4.4.7) • Other EMPYREAN Aggregators
Interfaces	• REST interface • Asynchronous interface for notifications and events

UCs	It will abstract the inter-Association interactions and facilitate their collaborative operation.

4.16 EMPYREAN Registry

Within the Association-based continuum, the EMPYREAN Registry (Figure 18) serves as a crucial component for discovering, cataloging, and advertising Associations and services across diverse environments, including IoT devices, edge devices and nodes, and central cloud platforms. It provides a unified entry point for both core platform services and third-party entities, enabling access to the functionalities offered by the EMPYREAN components.

The EMPYREAN Registry supports application developers in managing, maintaining, and updating application and deployment blueprints. It offers a centralized repository for exploring and accessing all available software, container images, services, and datasets, thereby enhancing the composability of deployed services and applications within the EMPYREAN platform. The Registry will be dynamically updated with new information as infrastructure resources are registered or new applications and services are published.

Furthermore, the Registry will store information and metadata about the available Associations, including their resources, the available services, and deployed applications. It will list all the available services, applications, and EMPYREAN-compliant artifacts along with their capabilities, requirements, and dependencies. All metadata and datasets from the individual platforms and data sources will be shared across multiple Associations, according to the selected privacy settings. The Registry will also interact also with other catalogues, providing a starting point for the interconnection with the Gaia-X service catalogue[11] composition model and architecture to ensure compliance and broader usage. The orchestration and deployment mechanisms of the EMPYREAN control and management plane will utilize this data to facilitate the efficient management of Associations and the deployment of hyper-distributed applications within and across them.

[11]Gaia-X Architecture Document - 24.04 Release - https://docs.gaia-x.eu/technical-committee/architecture-document/latest/enabling_services/#federated-catalogues

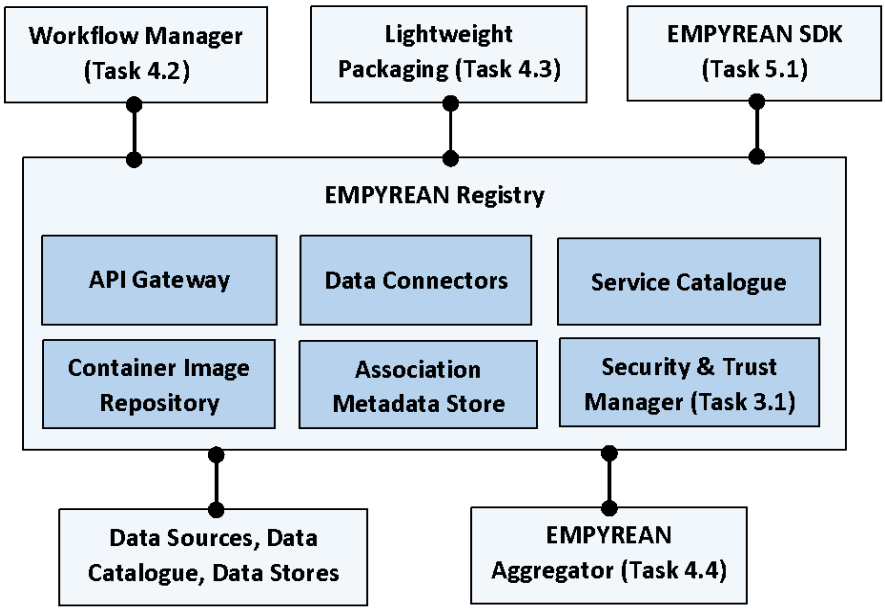

Figure 18: EMPYREAN Registry core components and dependencies

Table 20: Description of EMPYREAN Registry core components

Component ID	WP4.4.13
Name	EMPYREAN Registry
High level description	It manages the registration of IoT devices, edge, and cloud resources in Associations. It also abstracts to the Workflow Manager the interaction with the available Associations. The EMPYREAN registry will keep track of the available Associations and services, the mapping of the infrastructure resources to Associations, and the relation between users and Associations.
Collaborators	• 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) • Workflow Manager (WP4.2.1) • Application Packaging (WP4.3.3) • EMPYREAN SDK
Interfaces	• REST interface • Asynchronous interface for notifications and events
UCs	It will facilitate the seamless deployment of EMPYREAN use case applications across an Association-based IoT-edge-cloud continuum. It will handle along with the Workflow Manager the initial steps of the applications' lifecycle workflow within the EMPYREAN platform.

Component ID	WP4.4.14
Name	API Gateway
High level description	It is a lightweight version of the corresponding component in the EMPYREAN Aggregator. It facilitates the interaction and exchange of events between the EMPYREAN services and the Registry core components.
Collaborators	• Association Metadata Store (WP4.4.17) • Service Catalogue (WP4.4.15) • Container Image Repository (WP4.4.16) • Workflow Manager (WP4.2.1) • Application Packaging (WP4.3.3) • EMPYREAN Aggregator (WP4.4.11) • EMPYREAN SDK
Interfaces	• REST interface • Asynchronous interface for notifications and events
UCs	It will be used in all use cases as a core component of the EMPYREAN Registry that is essential for the operation of the EMPYREAN platform.

Component ID	WP4.4.15
Name	Service Catalogue
High level description	It will keep track of ownership and metadata for all the available software in the EMPYREAN platform such as, services, containers, and data pipelines. The Service Catalogue will store and manage the descriptors of hyper-distributed applications and services that are available for deployment on the EMPYREAN infrastructure. The catalogue will provide information about the software packages, container images, service descriptors, and other metadata that are required for service deployment and management.
Collaborators	• API Gateway (WP4.4.14) • Association Metadata Store (WP4.4.17)
Interfaces	• REST interface • Asynchronous interface for notifications and events
UCs	It will be used in all use cases as a core component of the EMPYREAN Registry that is essential for the operation of the EMPYREAN platform.

Component ID	WP4.4.16
Name	Container Image Repository
High level description	This component will be part of the EMPYREAN services that focus on building and storing EMPYREAN-enhanced lightweight container images. Once a hyper-distributed application is created through the EMPYREAN building and packaging mechanisms, its OCI-compatible images contain will be stored in the Container Image Repository.
Collaborators	• API Gateway (WP4.4.14) • Service Catalogue (WP4.4.15)

Interfaces	• REST interface • Asynchronous interface for notifications and events
UCs	It will be used in all use cases as a core component of the EMPYREAN Registry that is essential for the operation of the EMPYREAN platform.

Component ID	WP4.4.17
Name	Association Metadata Store
High level description	It will store high-level information and metadata concerning the formation of available Associations, including details about participating resources, their owners, and sharing policies. This information will be continuously updated during the registration of new resources and their decommissioning. Furthermore, the stored information will support the orchestration and load-balancing decisions of distributed decision-making mechanisms, ensuring efficient and balanced resource management across the continuum.
Collaborators	• API Gateway (WP4.4.14) • Service Catalogue (WP4.4.15) • EMPYREAN Aggregator (WP4.4.11)
Interfaces	• REST interface • Asynchronous interface for notifications and events
UCs	It will be used in all use cases as a core component of the EMPYREAN Registry that is essential for the operation of the EMPYREAN platform.

Component ID	WP4.4.18
Name	Data Connectors
High level description	It will collect metadata and information from various systems, including data stores, external catalogues, data pipelines, any other relevant data source. It will rely on connector interfaces for each specific data. By adding new connectors or expanding the graph model, the system can be extended to collect any required metadata.
Collaborators	• API Gateway (WP4.4.14) • Service Catalogue (WP4.4.15) • Association Metadata Store (WP4.4.17) • Data Catalogues, Data Sources, Data Stores
Interfaces	• REST interface • Asynchronous interface for notifications and events • Data-specific interfaces
UCs	It will be used in all use cases as a core component of the EMPYREAN Registry that is essential for the operation of the EMPYREAN platform.

5 EMPYREAN Architecture

5.1 High-Level Architecture

EMPYREAN envisions an IoT-edge-cloud continuum composed of collaborative collectives of IoT devices, robots, and resources that extend from the edge to the cloud. EMPYREAN refers to this concept as the Association-based continuum (Figure 19). In this paradigm, multiple Associations—each a collaborative collective of IoT devices, robots, and resources—operate simultaneously across different locations, collectively forming the IoT-edge-cloud continuum.

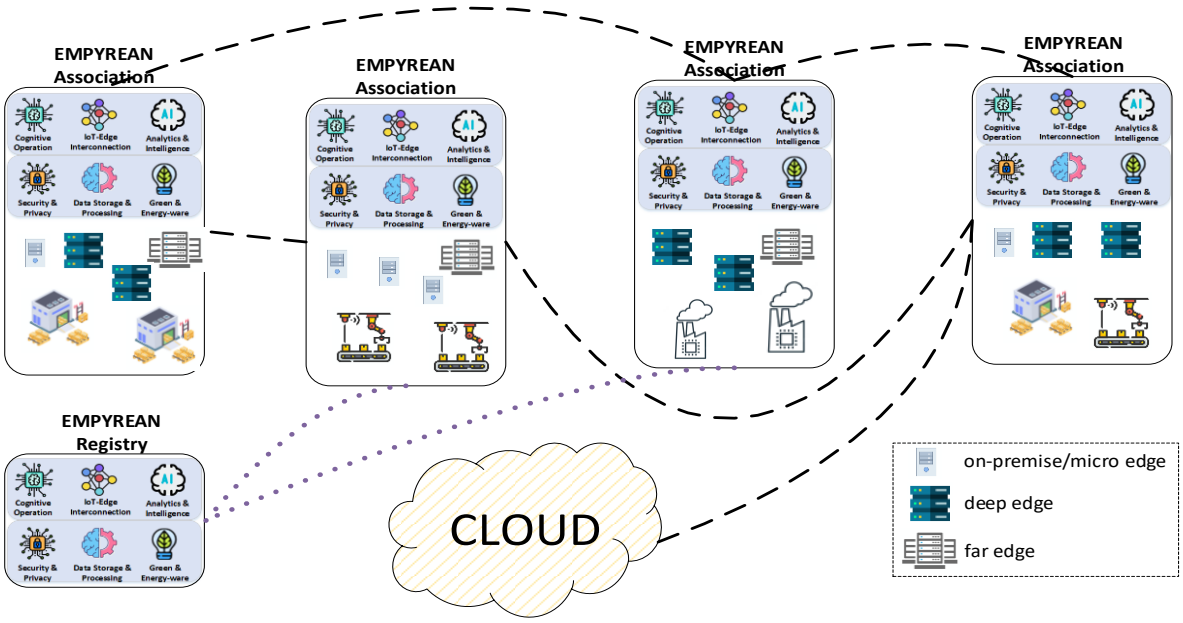

Figure 19: EMPYREAN Association-based IoT-Edge-Cloud continuum

An Association consists of a diverse array of IoT devices, robots, edge computing and storage resources, which can vary in size and purpose, encompassing both general-purpose and specialized units. Through an Association, resources from different owners are shared between the participant users and seamlessly and cognitively combined to create a unified virtual execution environment. These Associations are dynamically formed and updated, depending on the participation of resource owners. While the system primarily relies on edge resources, it can leverage central cloud resources whenever necessary.

EMPYREAN platform promotes the composability of infrastructures and services across the IoT-edge-cloud continuum. Associations facilitate the collaborative operation and management of virtual execution environments by pooling computational, storage, networking, and other infrastructure and service resources. This flexible and adaptive structure ensures efficient resource utilization and enhances the overall functionality of the IoT-edge-cloud ecosystem. Moreover, the EMPYREAN AI-enabled distributed control and management plane provides robust anomaly mitigation, adaptability, and self-driven recovery, ensuring resilient and efficient operations in the face of unforeseen issues across

the infrastructure. The EMPYREAN platform provides a loosely coupled continuum implementation that enables more local decisions and a collective logic that leads to system-wide welfare optimality to serve dynamic and hyper-distributed applications more efficiently.

We employed a top-down, iterative approach to define the EMPYREAN architecture. We began by drafting the high-level architecture, which outlines the conceptual design of the EMPYREAN platform, emphasizing the main components and functionalities without delving into implementation details. Following this, we developed the logical architecture, detailed in Section 5.3, which describes the logical components comprising the EMPYREAN platform and aligns closely with the developed technological solutions. This comprehensive design process integrates the analysis and functional requirements from tasks T2.1 “State-of-the-Art Analysis” and T2.2 “Concept, Use Cases and Requirements Analysis,” as documented in deliverable D2.1 (M6).

EMPYREAN adopts a layered architecture, where each layer consists of a set of discrete components that use well-defined and open interfaces to interact horizontally and vertically to form the EMPYREAN platform. Figure 20 illustrates the EMPYREAN high-level architecture.

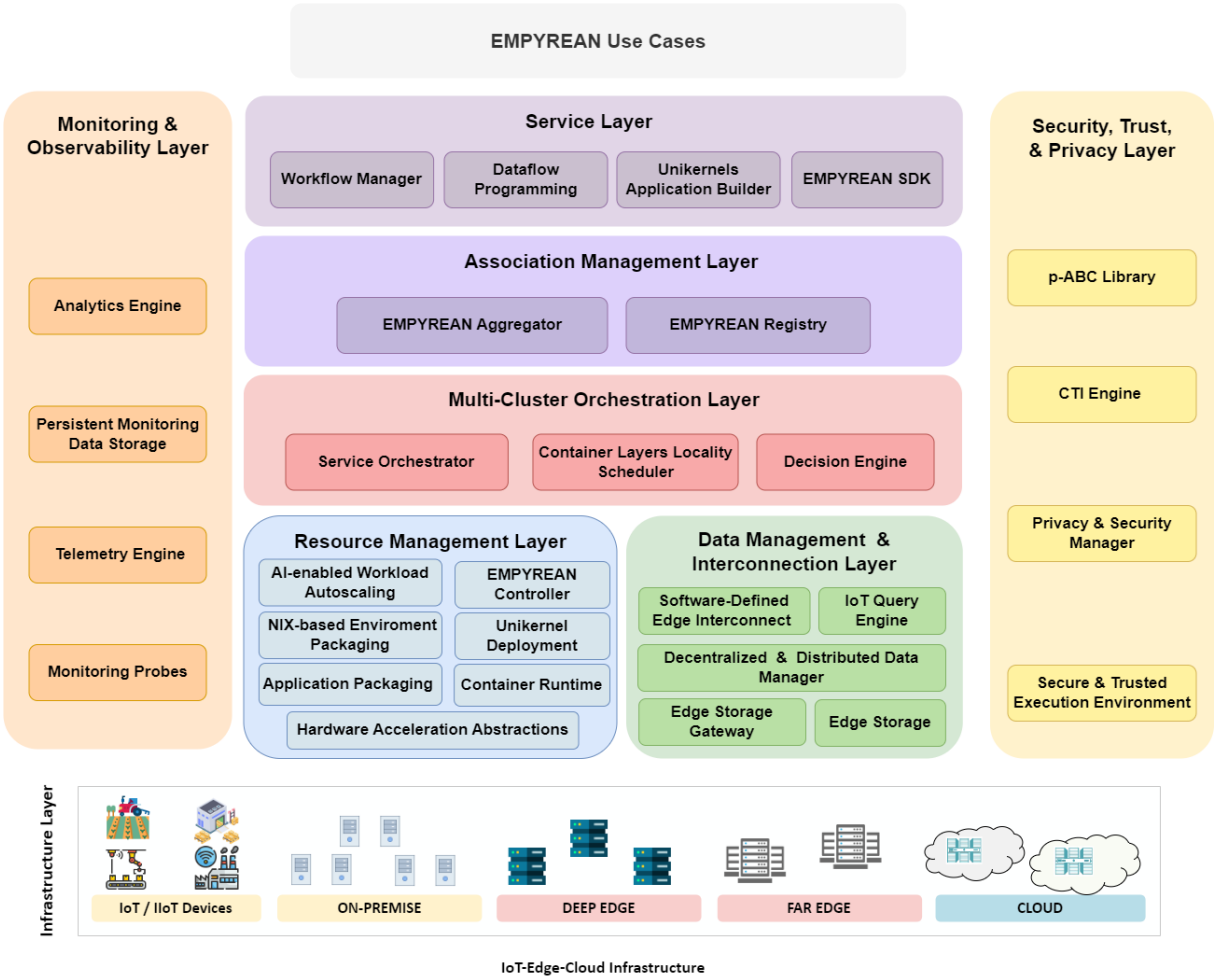

Figure 20: EMPYREAN high-level architecture

Moreover, the architecture is aligned with the current version of the European reference architecture for the continuum, provided by the Task Force (TF) 3 "Architecture" of the EUCloudEdgeIoT[12] initiative. The reference architecture[13] defines eight main categories of building blocks corresponding to the technical processes to operate applications along the continuum: Security & Privacy, Trust & Reputation, Data Management, Resource Management, Orchestration, Network, Monitoring & Observability, and Artificial Intelligence.

The **Service Layer** encompasses components designed to enhance 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 addresses, among others, the following key aspects: (a) workflow design and management of hyper-distributed applications, (b) cloud-native unikernel application development, and (c) data-flow description.

The *Workflow Manager* provides tools for the high-level design, development, and remote debugging of cloud-native applications. It allows these applications to be seamlessly deployed across the Association-based IoT-edge-cloud continuum. The *Unikernels Application Builder* facilitates the development and deployment of applications as unikernels in cloud-native environments. It reduces engineering overhead by simplifying the building process. Unikernels are highly efficient, lightweight, and secure, making them ideal for edge and cloud environments where performance and security are critical. The *Dataflow Programming* complements the workflow-based application management by focusing on a data-centric, decentralized, and highly dynamic data interconnection. It supports the declarative definition of data flow requirements through unified abstractions and location-transparent descriptions. This enables more responsive and adaptable data management, particularly in highly distributed and heterogeneous environments. Additionally, the *EMPYREAN SDK* empowers developers by providing a comprehensive toolkit for the development and deployment of hyper-distributed applications that fully leverage the EMPYREAN platform functionalities.

The **Association Management Layer** is responsible for the dynamic and transparent creation and management of Associations that integrate heterogeneous resources across multiple providers, connectivity types, and segments of the IoT-edge-cloud continuum. It incorporates components that intelligently and dynamically form resource federations, facilitating collaboration, resource sharing, workload and data distribution, and interoperability across diverse administrative domains within the IoT and edge-cloud environment. Alongside the Multi-Cluster Orchestration Layer, it forms the core of EMPYREAN's novel, distributed, and autonomous control and management plane, realizing the concept of an Association-based continuum.

[12] https://eucloudedgeiot.eu

[13] Task Force 3: Architecture. (2023). Developing a Reference Architecture for the Continuum - Concept, Taxonomy and Building Blocks. Zenodo. https://doi.org/10.5281/zenodo.8403593

At the heart of this layer is the EMPYREAN Aggregator, a key component that enables the formation, coordination, and management of Associations, while also facilitating the discovery of available resources. An Aggregator (Figure 21) can manage multiple Associations, within each it enables IoT devices and computational resources, potentially spanning multiple clusters, to cooperate dynamically and autonomously. Moreover, it interfaces with other Aggregators and edge or (multi) cloud providers, enabling services and data to be processed seamlessly across various providers. Multiple self-managed and interacting Aggregators constitute the distributed and data-driven management plane for the EMPYREAN platform.

This setup abstracts the underlying complexity and heterogeneity of diverse IoT devices and edge resources, while providing resiliency, fault-tolerance, and elasticity for users and applications. Additionally, the *EMPYREAN Registry* plays a critical role in managing the registration of IoT devices, edge, and cloud resources within Associations, while also tracking available services and resources. It registers all data within the system, making it accessible and providing EMPYREAN developers and platform services with a unified view of all available software, services, machine learning models, and datasets. This unified view enhances operational efficiency and simplifies the development and deployment of applications within the EMPYREAN platform.

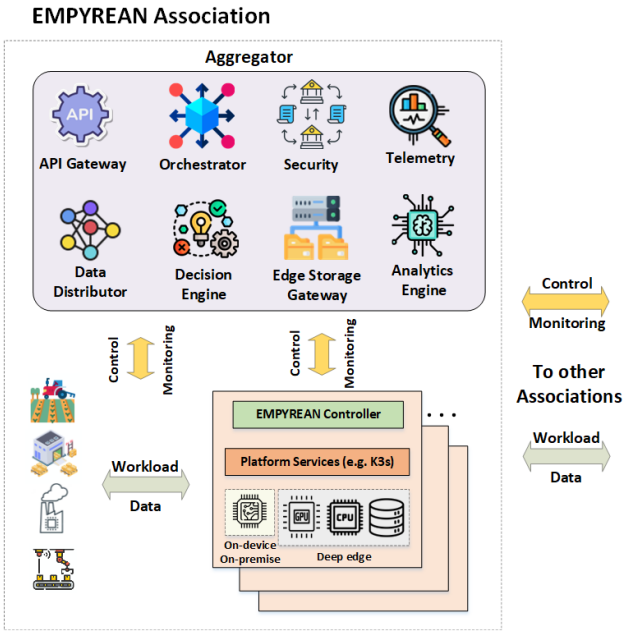

Figure 21: EMPYREAN Aggregator and Associations' management

The **Multi-Cluster Orchestration Layer** provides efficient service orchestration and resource management across the disaggregated and heterogeneous EMPYREAN infrastructure. This layer includes distributed, cognitive, and autonomous decision-making mechanisms designed to efficiently orchestrate emerging, highly dynamic, hyper-distributed applications, while also supporting their autonomous and self-driven adaptations. Multiple instances of this layer components underpin the decentralized and multi-agent operation of the EMPYREAN control and management plane, optimizing resource utilization and providing scalability, resiliency, energy efficiency, and high quality of service.

The *EMPYREAN Orchestrator and Decision Engine* are key components responsible for service orchestration and resource management. The EMPYREAN Orchestrator oversees the efficient deployment of applications and coordinates the necessary actions for resource management. Workload assignment decisions are delegated to the Decision Engine, which enables decentralized, speculative, and multi-objective resource orchestration. The Decision Engine integrates various distributed optimization and orchestration algorithms to balance computing tasks and data both locally within an Association and across federated Associations. It also considers environmental factors, enabling intelligent and energy-efficient workload and data distribution. These components operate at the Association level, managing multiple clusters within the EMPYREAN platform. Additionally, the *Container Layers Locality Scheduler,* implemented as a scheduling plugin for the local orchestrator in each platform, optimizes workload scheduling at the cluster level. It enhances the native decision-making capabilities of Kubernetes and K3s by implementing advanced scheduling algorithms that minimize cold start delays and optimize workload placement based on container layer locality.

The **Resource Management Layer** abstracts the complexity of managing and interacting with orchestration and deployment mechanisms across IoT, edge, and cloud platforms, all of which are unified under the EMPYREAN platform. To this end, it integrates a wide range of novel software mechanisms that span the entire system stack from platform-level scheduling mechanisms (e.g., EMPYREAN Controller, AI-enabled Workload Autoscaling) to low-level mechanisms (e.g., Unikernel Deployment, Container Runtime). The components within this layer operate within a specific Kubernetes or K3s cluster, providing targeted management and optimization of resources. Additionally, the layer's modular design offers flexibility to the EMPYREAN platform, allowing for seamless extension and the quick integration of new hardware and software platforms at the Infrastructure Layer. This modularity ensures that the platform remains adaptable and scalable, capable of evolving with emerging technologies and diverse deployment environments.

The *EMPYREAN Controller* serves as a bridge for integrating individual IoT, edge, and cloud platforms. It incorporates platform-specific logic and interfaces to enable efficient and seamless deployment of cloud-native applications and serverless workloads. It translates assignment decisions from the multi-cluster orchestration layer's decision-making mechanisms into platform-specific deployment objectives. These objectives are then conveyed as declarative descriptions to low-level deployment mechanisms, ensuring smooth and consistent deployment across heterogeneous environments. The *AI-enabled Workload Autoscaling* component enhances the Kubernetes orchestrator by incorporating AI/ML techniques for intelligent workload autoscaling. By analysing historical data, this component ensures optimized resource allocation, dynamic application-level adaptations, and efficient utilization of resources, providing a more responsive and adaptive environment for workloads.

The *Application Packaging* and *NIX-based Environment Packaging* components support multi-environment and multi-architecture packaging for cloud-native applications, improving the interoperability and adaptability of workloads within the EMPYREAN platform. They streamline the packaging process by creating OCI-compatible container images, supporting multiple architectures and programming languages, and ensuring deployment flexibility

across different execution environments. The *Hardware Acceleration Abstractions* component, based on the open-source vAccel[14] framework, enables the offloading of compute-intensive tasks to hardware accelerators on neighbouring nodes. This offloading is performed while ensuring data security and integrity, thereby enhancing performance for resource-heavy workloads without compromising on safety. Additionally, the *Unikernel Deployment and Container Runtime* components provide a versatile container runtime integration for the deployment of cloud-native applications across various execution environments. These components facilitate the spawning of unikernels and their integration with generic container runtimes that are compatible with Kubernetes and serverless architectures.

The **Data Management and Interconnection Layer** is responsible for managing secure data storage and ensuring dynamic interconnection and communication between IoT devices and computing and storage resources. This layer's components operate both at cluster and Association levels, enabling flexible and scalable data management across the EMPYREAN platform. They facilitate the seamless integration of IoT devices with edge and cloud resources, enabling data-driven applications to operate effectively in highly distributed and heterogeneous environments.

The *Edge Storage*, *Edge Storage Gateway*, and *IoT Query Engine* are the core elements of EMPYREAN's secure and hybrid cloud-edge storage and efficient time series data storage management. They are designed to manage storage resources across cloud and edge environments, supporting hybrid policies for data distribution, redundancy, and security. They also enhance the storage and retrieval of IoT time series data, employing erasure coding techniques to ensure secure and reliable data management. This approach provides robust support for efficient and scalable time series data storage, which is critical for applications requiring high-performance data access and analytics. Additionally, the *Decentralized and Distributed Data Manager* implements decentralized and distributed communication mechanisms with efficient publish/subscribe and data querying capabilities. It facilitates communication between IoT devices and edge computing and storage resources across various providers/administrative domains, connectivity types (e.g., extremely constrained networks), technologies, and network zones. Moreover, the *Software-Defined Edge Interconnect* provides a high-performance data transport service that integrates remote I/O operations into large computational pipelines, such as AI training workflows. It optimally overlaps computation with network I/O, improving the efficiency of data-intensive tasks across distributed environments and thereby supporting real-time processing and analytics.

The **Infrastructure Layer** consists of heterogeneous resources from various administrative and technology domains including, (i) IoT/IIoT devices, robots, and on-premise, micro edge resources, where data is produced and service requests are generated, (ii) deep and far edges, close and further from the end users/devices, for real-time processing and as aggregator nodes, and (iii) multiple clouds (federated operation) to increase robustness and reduce cost and dependencies on a single cloud provider, for data storage and replication. This layer also

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

includes the EMPYREAN-enhanced software resources (Section 3.2). Moreover, EMPYREAN will not only accommodate container-based application development and deployment, but also the serverless paradigm allowing users to optimize the execution of their functions. That exceptional unification of highly diverse resources and deployment modes provides the EMPYREAN platform the ability to cater for application and user constraints, while calibrating the configuration of available resources.

The architecture is complemented by the Security, Trust, and Privacy Layer and the Monitoring and Observability Layer, which are across the other layers, providing critical functionalities for the overall platform.

The **Security, Trust, and Privacy Layer** integrates components that are designed to ensure secure access to resources while addressing privacy concerns and enabling trusted execution. These components are distributed across EMPYREAN platform, functioning at both cluster and Association levels. They ensure that Associations operate as secure and trusted execution environments, where trust between data-generating and data-processing entities is continuously validated using distributed trust services. In parallel, identity and data access management components ensure controlled access and data confidentiality among different entities.

The *Privacy and Security Manager* component and *P-ABC* library provide robust identity and access management alongside attribute-based credential management. The Privacy and Security Manager ensures secure and private identity management, data verification, and a strong cryptographic foundation for managing privacy-preserving attribute-based credentials across the platform. The P-ABC library complements this by offering a distributed privacy-preserving attribute-based credential system based on PS multi-signatures. The *Secure and Trusted Execution Environment* establishes secure and trusted execution across the IoT-edge-cloud continuum, supporting secure and measured boot mechanisms. It enables applications to be deployed seamlessly with varying levels of security and trustworthiness across different hardware platforms. This allows for scalable and transparent operation, from micro deep edge devices to the far edge and cloud environments, ensuring that security requirements are met at all levels. Furthermore, the *Cyber Threat Intelligence (CTI) Engine* facilitates automated cyber threat analysis, providing valuable insights into past cyber threat events observed globally. By quantifying system risks, it enables proactive adaptations within and across EMPYREAN Associations, significantly enhancing overall platform security.

Finally, the **Monitoring and Observability Layer** integrates real-time monitoring, observability, and service assurance components to provide comprehensive visibility and control over the EMPYREAN platform. This layer incorporates distributed and automated telemetry mechanisms that dynamically collect a wide range of metrics from heterogeneous infrastructures and deployed applications. These mechanisms continuously track the health, performance, and availability of IoT devices, edge/cloud infrastructures, platform services, and applications, facilitating data-driven decision-making and enabling advanced automation capabilities.

At the core of this telemetry infrastructure are the *Telemetry Engine*, *Monitoring Probes*, and *Persistent Monitoring Data Storage (PMDS).* The Telemetry Engine also provides smart observability, offering an initial analysis of telemetry data to deliver real-time insights into system performance and security. This enables rapid anomaly detection, efficient resource utilization, and prompt response to emerging issues. Moreover, the *Analytics Engine* provides service assurance by using AI-driven analytics on top of monitoring and observability data. This approach ensures that applications perform as intended by dynamically adjusting deployments based on changing conditions and requirements. Components within this layer operate both at cluster and at the Association level, the latter covering multiple clusters to ensure a cohesive and scalable monitoring and observability framework across the entire EMPYREAN platform.

5.2 Data Spaces and Architecture

The increasing importance of data has led to the creation and development of Data Spaces, which are ecosystems where organizations can share data from different sources and collaborate to achieve a goal. Data Spaces are essential in today's data-driven world, promoting innovation, collaboration, and value creation by enabling trusted and secure data sharing across industries and sectors. They are designed to break down traditional data silos by providing a common framework where participants can maintain control over their own data while sharing it with others under mutually agreed-upon terms. Data spaces bring together relevant data infrastructures and governance frameworks to facilitate data pooling and sharing. The emergence of Data Spaces entails a shift in the way that companies and organizations share and manage data.

From a technical point of view, Data Spaces are a concept of data management: they put technology systems and rules in place to integrate and exchange data. What emerges is a federated data ecosystem based on shared policies and rules. Data is distributed across storage points and integrated on the basis of what is needed. Tools are provided to discover, access, and analyse data that is distributed across industries, companies and entities.

Data Spaces are designed with a focus on robust architecture, emphasizing interoperability and security. To this end, Data Spaces incorporate several key roles:

- **Data Owners***:* Entities that hold the rights to access and use the data, maintaining control over how their data is shared and utilized within the Data Space.
- **Data Providers**: Entities that make data available within the Data Space, managing data offerings through a catalogue that ensures data discoverability and accessibility.
- **Data Consumers**: Participants who access and utilize the data offered in the Data Space, leveraging shared data to drive insights, innovation, and decision-making.

- **Data Intermediaries:** Entities that provide services that facilitate data access and sharing within the Data Space. They preserve the catalogue of organized data, enforce data quality, and ensure operational control through various application tools.
- **Technology Providers**: Entities that supply the necessary technologies to enable the functionality and utility of the Data Space, including infrastructure, software, and platform services.
- **Operators**: Responsible for the definition, management, and maintenance of the Data Space, ensuring that it operates smoothly, securely, and in accordance with agreed-upon standards and protocols.

A standard reference technological framework is employed for implementing the Data Spaces, incorporating key elements such as the *Data Space Registry*, *federated services* of the Data Space, and the *connector*. The connector serves as the primary hardware and software agent that enable secure and interoperable data sharing within Data Spaces ecosystems. It allows participants in the Data Space to function as data providers, consumers, or both, ensuring interoperability and compliance to predefined standards for each transaction within the Data Space. There is available a reference implementation supported by the Data Space Business Alliance[15] (DSBA), a coalition that promotes a robust data economy and include several key organizations such as the International Data Spaces Association[16] (IDSA), FIWARE[17], Gaia-X[18], and Big Data Value Association[19] (BDVA). Additional implementations include the Eclipse Data Space Connector[20] (EDC), an open-source and extensible connector developed under the Eclipse Foundation, and the FIWARE True Connector[21] (FTC) for the IDS ecosystem that promotes a standardized approach to connecting and interacting within Data Spaces, facilitating trusted data sharing and compliance with the IDS reference architecture.

The EMPYREAN architecture is designed to facilitate seamless operation within multi-instance, multi-domain configurations in federated frameworks, supporting advanced edge/cloud services and Data Spaces. EMPYREAN aligns with the concepts defined by Gaia-X, which emphasizes secure, cross-border data sharing and the establishment of a federated data infrastructure. In the Gaia-X framework, various infrastructure ecosystems and data ecosystems are interconnected to foster the data economy. This framework defines three pillars (Figure 22) that must be addressed to integrate infrastructure and data ecosystems effectively:

- **Compliance**: Ensuring that all participants adhere to the common set of rules and standards defined by the ecosystem. This includes data protection regulations, security

[15] Data Space Business Alliance: https://data-spaces-business-alliance.eu

[16] International Data Spaces Association: https://internationaldataspaces.org/

[17] FIWARE Foundation: https://www.fiware.org/

[18] Gaia-X: https://www.gaia-x.eu/

[19] Big Data Value Association: https://www.bdva.eu/

[20] Eclipse Dataspace Components: https://projects.eclipse.org/projects/technology.edc

[21] FIWARE TRUsted Engineering Connector: https://fiware-true-connector.readthedocs.io/en/latest/

protocols, and usage policies that govern how data can be shared and used within the Data Space.

- **Federation**: Facilitating the interconnection of different data and service providers through a federated model. This allows for the seamless integration of resources from multiple domains, enhancing the scalability and flexibility of the Data Space.
- **Data Exchange**: Enabling the efficient and secure exchange of data between participants, supported by a standardized set of protocols and interfaces. This pillar focuses on maintaining data integrity, ensuring traceability, and providing mechanisms for consent and data usage management.

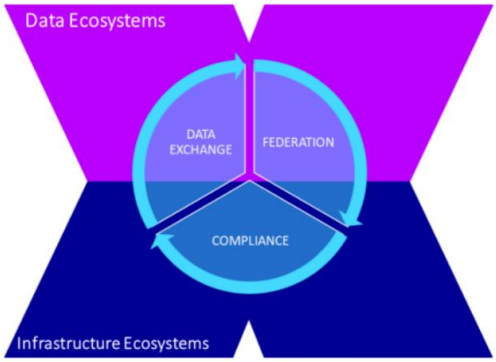

Figure 22: Gaia-X Connecting Data & Infrastructure Ecosystems[18]

The EMPYREAN architecture is inherently designed to address the federation pillar, supporting a hyper-distributed computing paradigm that relies on federations of collaborative and heterogeneous IoT devices, multilayer edge, and cloud resources across different providers and networks. This architecture promotes a highly adaptable and scalable framework, facilitating the seamless integration and cooperation of diverse resources, regardless of their geographical or administrative boundaries. EMPYREAN incorporates distributed and AI-enabled decision-making mechanisms that dynamically balance computing tasks and data both within individual Associations and across multiple federated Associations. This approach brings services closer to the edge, leveraging trustworthy IoT devices and edge resources for efficient data processing. By doing so, EMPYREAN ensures that data sovereignty is maintained, providing stakeholders with control over their data.

Additionally, the EMPYREAN platform can function as a Data Provider. Data generated from IoT resources is collected, stored, and analysed through developed extreme scale analytics mechanisms. This processed data can then be offered as valuable data services, enhancing the platform's role within data-centric ecosystems. The platform's ability to provide data services makes it an integral part of the data economy, where data becomes a key asset for innovation and decision-making. Through components like the EMPYREAN Registry and EMPYREAN Aggregator, the platform operates effectively as both an Infrastructure Service Provider and Data Provider within the Gaia-X ecosystem. These components enable seamless registration,

management, and discovery of resources, facilitating interoperability and promoting a federated model where data and resources are shared across multiple domains.

Moreover, EMPYREAN fosters the development of a dynamic open-edge ecosystem by enabling a marketplace where infrastructure and service providers, developers, and end-users can interact. This collaborative platform integrates diverse stakeholders from across the value chain, encouraging innovation and the development of new services and applications. EMPYREAN's components support the advertisement of data, edge resources, and services to third-party marketplaces through standardized APIs, enhancing market accessibility and fostering a competitive environment.

Another key concept in defining a Data Space is the trust framework (Figure 23), which includes all agreements and decisions necessary to establish a functional Data Economy within the ecosystem. This framework integrates elements that ensure interoperability, trust, data sovereignty, and empowerment among participants. It also employs mechanisms that protect and preserve privacy, utilizing privacy-preserving enablers to keep data secure. Enhanced privacy and confidentiality are crucial in identity (trust) management within data connectors, as compromising sensitive identity information can result to significant data breaches. The Gaia-X Trust Framework[22] envisions the use of verifiable credentials and linked data representations. EMPYREAN's architecture aligns with Gaia-X principles, emphasizing secure, federated data sharing and infrastructure cooperation, thereby contributing to a robust and scalable data economy.

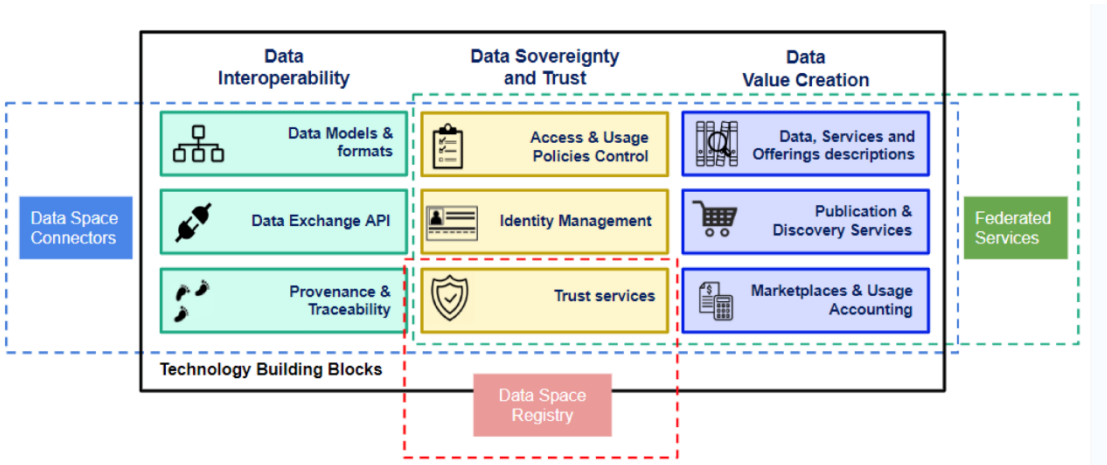

Figure 23: Design principles for Data Spaces[23]

EMPYREAN will offer secure identity and access management, alongside attribute-based credential management mechanisms, to support privacy-preserving attribute-based credentials across the Association-based continuum. These mechanisms will enhance data sovereignty by establishing trust between stakeholders, making data and services searchable, discoverable, and consumable in a secure manner.

[22] Gaia-X Trust Framework: https://docs.gaia-x.eu/policy-rules-committee/trust-framework/22.10/

[23] Data Spaces Business Alliance, Technical Convergence: https://data-spaces-business-alliance.eu/wp-content/uploads/dlm_uploads/Data-Spaces-Business-Alliance-Technical-Convergence-V2.pdf

A privacy-preserving Self-Sovereign Identity (SSI) approach empowers end-users with control over their personal data without relying on a central authority, involving three main participants: the holder of Verifiable Credentials (VCs) with a digital wallet, the issuer of VCs, and the verifier of VCs, supported by a decentralized Data Registry like Blockchain. To strengthen the basic SSI model, EMPYREAN incorporates a cryptographic module that includes distributed attribute-based credentials leveraging Pointcheval-Sanders multisignatures (dp-ABC) and Zero-Knowledge Proofs (ZKP). This robust and comprehensive approach to privacy-preserving identity management allows users to disclose only the necessary attributes required for specific transactions through Selective Disclosure, ensuring sensitive information remains protected and that users have greater control over their personal data. ZKP further enhance security by allowing the prover to demonstrate to the verifier that specific attributes are met without revealing the underlying data. This cryptographic module significantly enhances the security and privacy of the SSI mechanisms, granting users greater control over their personal information.

Access control is another crucial mechanism that ensures data is accessible only to authorized entities while maintaining privacy and compliance with security policies. Sticky policies serve as a form of data policy management that persists with the data as it moves across different systems and environments, offering fine-grained access control via Ciphertext-Policy Attribute-Based-Encryption (CP-ABE). These policies ensure data privacy by restricting access to authorized members. Data producers encrypt their data based on attribute-based policies (CP-ABE), and once access is granted, the consumer must decrypt the data. The decryption process relies on a key associated with the consumer's identity attributes, ensuring that data can only be decrypted if the attributes align with the sticky policy.

Additionally, a policy enforcement model based on the XACML framework will be employed. This model, combined with sticky policies and ABE, creates a secure resource access authorization layer, integrating identity attributes with user preferences, permissions, and consent choices. Beyond basic access control, EMPYREAN also addresses usage control by enabling more dynamic and context-aware control over the usage of the assets. Policies are defined as a set of rules consisting of conditions and associated decisions (e.g., permit, deny), allowing for refined control over how data and resources are utilized within the ecosystem.

5.3 Logical Architecture

A more detailed and elaborate design for the overall architecture of the EMPYREAN platform after the first iteration of the requirements analysis and design (M01-M07) is presented in Figure 24. It includes all the components that will be developed by the project technical work packages (WP3-5), previously presented in Sections 3 and 4. Additionally, the diagram illustrates the core interactions and required information exchanges. According to the project implementation plan, the final version of the EMPYREAN architecture will be reported in deliverable D2.3 “Final EMPYREAN architecture, use cases analysis and KPI” (M12), including all the required revisions, detailed workflows, and interfaces specifications.

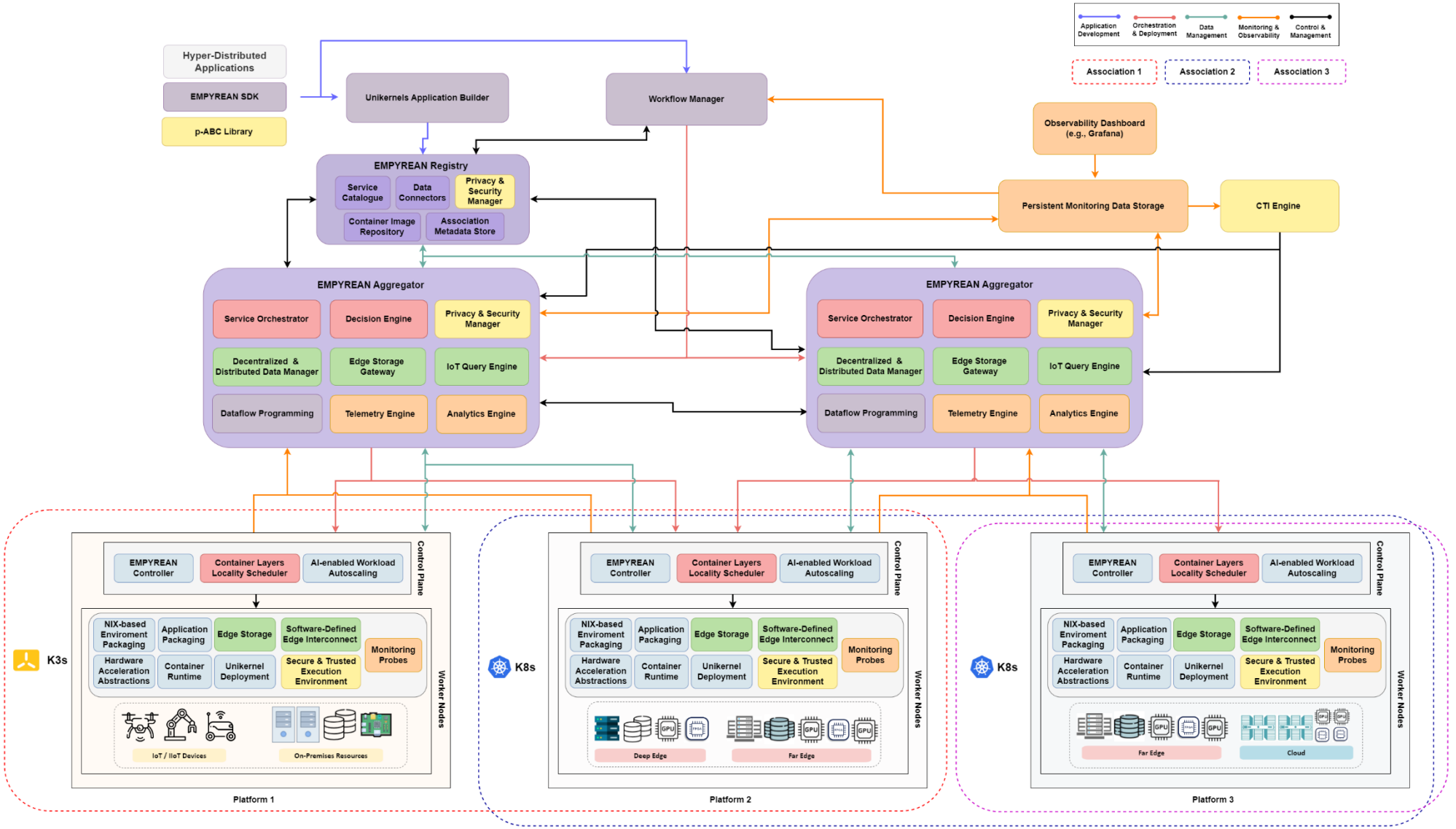

Figure 24: EMPYREAN logical architecture

6 EMPYREAN Platform Deployment View

EMPYREAN seeks to demonstrate the innovative capabilities of its platform by showcasing how it can enable trustworthy, cognitive, and AI-driven collaborative Associations of IoT devices and edge resources for efficient data processing. This will be achieved through three carefully selected use cases (UCs) that span diverse domains characterized by device- and data-intensive applications. These use cases correspond to diverse, dynamic, hyper-distributed, safety-critical, and high-performance demanding applications that impose diverse and heterogeneous requirements.

The EMPYREAN UCs correspond to high-impact applications in (i) advanced manufacturing with a focus on enabling real-time anomaly detection in robotic machining cells, (ii) smart agriculture with a focus on enabling proximal sensing in agriculture fields, and (iii) warehouse automation with a focus on robotic semi-autonomous and lights out logistics order picking. Each of these UCs involves robots and other devices equipped with cameras and sensors, operating in diverse environments such as industrial settings, warehouses, and agricultural fields. These environments demand significant computational power and intelligence at the edge due to the continuous data streams, the unpredictable and ad hoc nature of computational loads, and the distinct processing, data, and latency requirements. Deliverable D2.1 (M6) provides a comprehensive specification of the EMPYREAN use cases, detailing their current and envisioned future states with the implementation of EMPYREAN, identifying key challenges, and outlining an initial validation and testing methodology.

Next, we provide an initial high-level overview of the individual deployments that will be established for the evaluation of each UC. In the second and third years of the projects, activities within WP6 “Use Cases Demonstrator and EMPYREAN Evaluation” will refine this description by detailing the evaluation methodology, defining a comprehensive set of evaluation Key Performance Indicators (KPIs), setting up testbeds for demonstrators, and specifying detailed demonstration and evaluation scenarios for each use case. These efforts will ensure that the EMPYREAN platform is rigorously tested and validated against the diverse and demanding requirements of its targeted applications.

6.1 Anomaly Detection in Robotic Machining Cells (UC1)

Based on the EMPYREAN architecture design (Section 5) and the analysis of this use case, as elaborated in deliverable D2.1 (M6), we present an initial deployment architecture. The specificities of this advanced manufacturing use case, namely the purely on-premise setup (no Cloud interaction), the online, real-time operations and relatively low compute power at the deep-edge dictate a particular setup. In this setup, complex and compute-intensive processing occurs at the far-edge, while the deep-edge primarily collects data from sensors and forwards it to the far-edge for processing. The deep-edge may also perform some basic pre-processing tasks.

Multiple Kubernetes clusters (with possibly different distributions to adjust to the lower compute power) will enable the orchestration of containerized workloads across the respective layers (far-edge, deep edge, etc.). This will be then combined with a higher multi-cluster scheduling, offered by EMPYREAN components, enabling efficient execution across the different infrastructure layers based on application requirements.

The far-edge, which comprising the most computationally powerful hardware, will be equipped with at least 16 or 32GB of RAM to host the majority of EMPYREAN components. It may also include a GPU pool of nodes to execute the training of models needed for the "Fingerprint Generation System" process (see D2.1 - section 4.1.3). The deep-edge, composed mainly by the "Edge Smart Boxes - 8GB of RAM" (see D2.1 - section 4.1.5), will host a lightweight Kubernetes distribution along with distributed EMPYREAN components, such as the Ryax Workflow Engine Worker to control executions and connect with the main Ryax server on the main cluster along with ZenoFlow daemon to perform the necessary dataflow and real-time communication medium with the far-edge. While, some pre-processing may occur at the deep-edge, this layer's primary role is to transfer data to the far-edge for online operations. Finally, the low-power hardware at the on-premises edge, such as the "Edge Smart Boxes - 2/4GB of RAM" (see D2.1 - section 4.1.5) will probably only run light executables orchestrated by the deep-edge microservices. These will primarily handle data collection and transfer from the robots' sensors.

Following the previous setup, the project will first explore the deployment of a single EMPYREAN association for one robot that will be managed by one Aggregator. Towards the end of the project, more complex scenarios will be investigated, such as a robot participating in multiple associations and having multiple robots per association. Key EMPYREAN features essential for the success of this deployment include (non-exhaustive list): (i) the multi-clustering support within Ryax Workflow Engine, along with the scheduling optimizations at both the multi-cluster and local cluster levels, (ii) the integration of Dataflow programming framework within the Ryax Workflow Engine to support real-time data communication, (iii) the application builder for unikernels, which will enable the deployment of binaries remote controlled execution for IoT devices, (iv) the EMPYREAN telemetry service, (v) the analytics-friendly distributed storage, (vi) and the privacy and security manager to ensure the secure interactions in the highly distributed environment.

Concerning the workflow architecture, the initial approach will involve adapting the existing production workflows (see D2.1 - section 4.1.3, figure 9) to the EMPYREAN distributed architecture by leveraging the different components and added functionalities brought by EMPYREAN. Hence, a possible direction is to decompose the current processes into the following Ryax workflows (Figure 25):

1st workflow: This workflow will operate between the deep-edge and the far-edge, involving the on-premises edge layer. The Data Recording System (DRS) will be executed at the deep-edge, retrieving high-frequency data from on-premises edge devices and IoT sensors, and compute some indicators for a first-level processing. It will then feed the data to a MongoDB-based message queue system at the far-edge.

2nd workflow: Executed primarily at the far-edge, this workflow will be triggered by the data stored in the MongoDB-based message queue. It will execute the Fingerprint Comparison System (FCS) (initially using statistical simulations, later involving to ML inference), trigger online alerts, and store necessary data on the database.

3rd workflow: Periodically executed on the most powerful computational units at the far-edge, this workflow will retrieve data from the database and execute the Fingerprint Generation System (FGS) to automatically generate patterns (ML training). These models will be stored in the database for use in the 2nd workflow's inference processes.

The first two workflows will be our focus during the first years of the project while the 3d one will be developed towards the end.

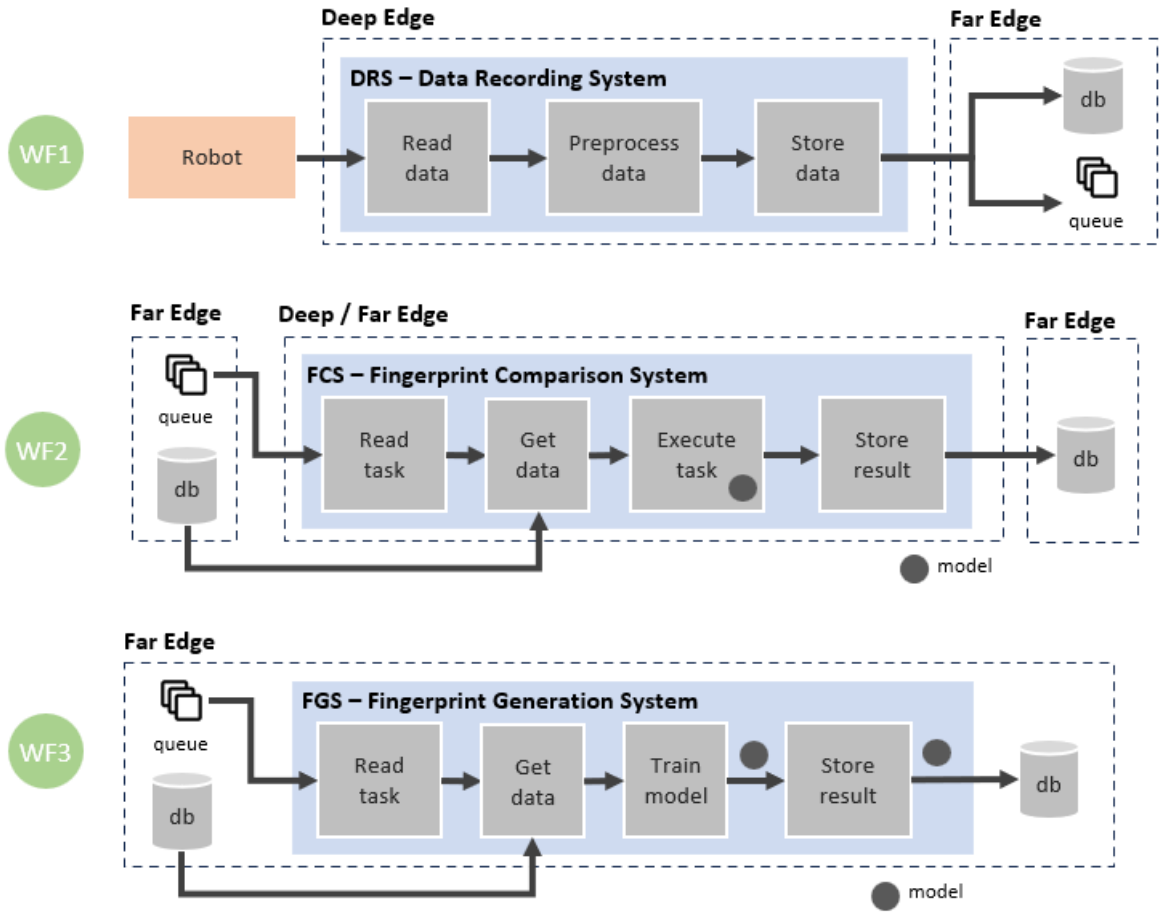

Figure 25: Possible breakdown of the current behavior into three workflows (WF)

6.2 Proximal Sensing in Agriculture Fields (UC2)

This section presents an initial analysis of the smart agriculture use case in combination with the EMPYREAN architecture, to propose a preliminary deployment setup. The use case is characterised by remote, battery-powered field units (drones, robots) with 4G network connection. The deep-edge infrastructure consists of drones or robots, collecting data from different on-board sensors, offering basic computational capabilities. The far-edge infrastructure has a more typical power supply, with WiFi or cabled network connections and more powerful computational units, potentially including GPUs for ML inference.

A key objective of this use case is to perform data processing as close as possible to the sources in order to transition to a near real-time assessment of soil health. To achieve this efficiently, besides the real-time data processing nature of operations, spanning from edge to cloud, we also need to manage the power consumption of the battery-powered field units (drones, robots), ensuring their tasks are executed optimally within the limited time before they need recharging. EMPYREAN components will provide key features to facilitate this operation.

The initial computing infrastructure setup will be composed by: (a) drones and robots equipped with on-board computational units with around 8GB of RAM (such as Raspberry Pi 5), representing the deep-edge. The drones and robots will also feature various on-board sensors, such as RGB or multispectral/hyperspectral cameras, (b) intermediate servers near the fields, consisting of PC's and laptops or GPU units (such as Nvidia Jetson) with around 32GB of RAM, representing the far-edge, and (c) connection to the cloud, which will offer more powerful computation, utilizing a variety of hardware resources and access to GPU node pools.

Initially, the deployment will explore a single EMPYREAN association and aggregator to enable the execution of tasks from the deep-edge to the far-edge up to the cloud. In the second half of the project, we will investigate more complex scenarios involving multiple associations, where each association may be dedicated to different soil fields and different organisations, featuring different deep-edge setups but sharing far-edge and cloud resources.

Several key EMPYREAN features critical to the success of this deployment include: (i) power consumption monitoring and its integration with the telemetry service, (ii) the privacy and security manager along with the cyber-threat intelligence engine to guarantee that operation will remain secure even in highly vulnerable contexts (such in 4G networks), (iii) the support for intermittent connectivity, (iv) the provision and control of secure and distributed edge storage, (v) the multi-clustering support, and (vi) the energy-efficient orchestration, among the others.

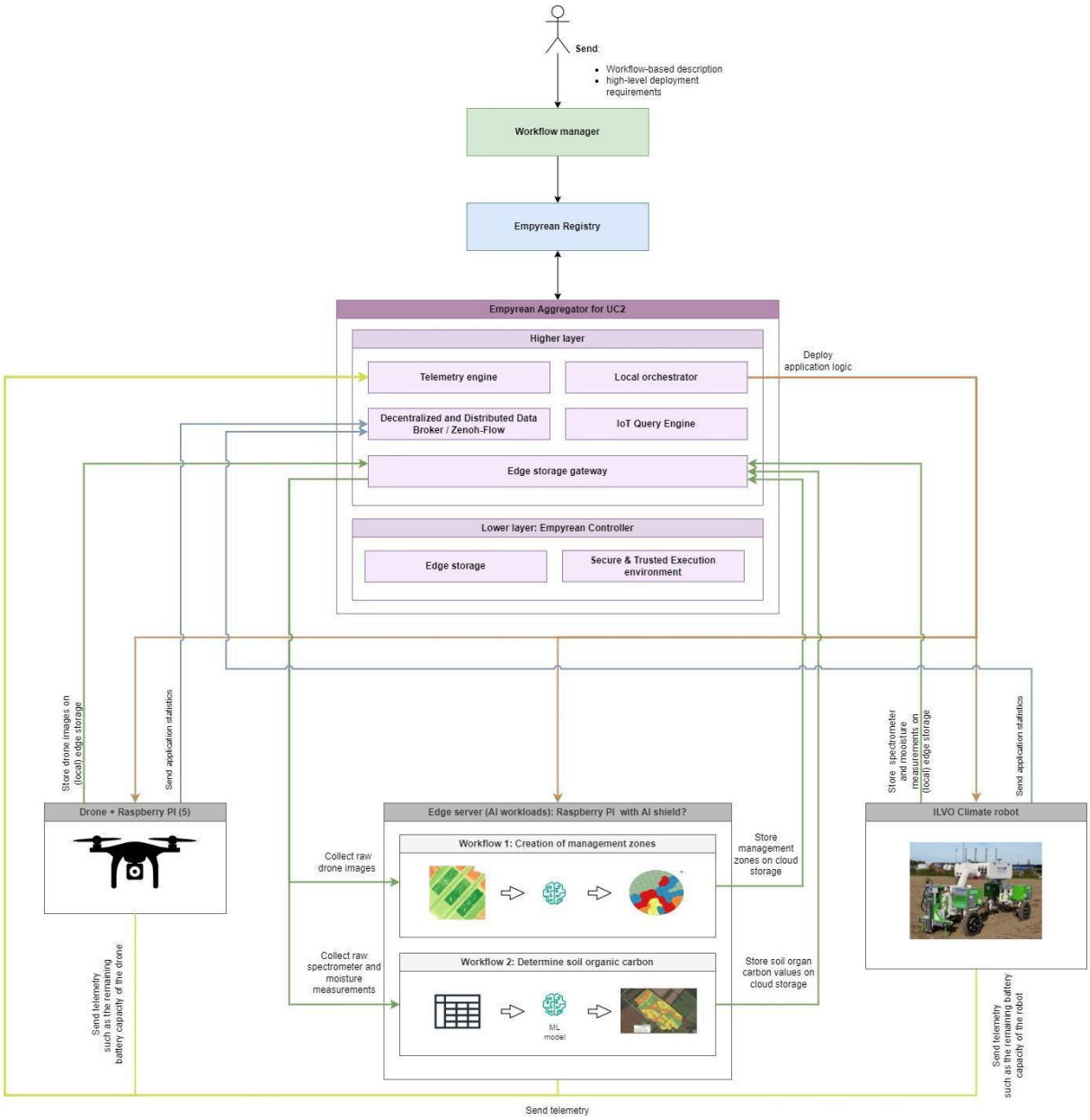

Figure 26: Proximal sensing in agriculture fields use case deployment overview

Regarding the workflow architecture, Figure 26 presents an initial design of the workflows to be explored. This design not only highlights the task separation and the way they will be deployed on the hybrid computing infrastructure but also illustrates the use of various key EMPYREAN components and how they will be utilised to efficiently address the needs of the workflows. In particular, the following workflows are anticipated:

1st workflow – Management Zone Creation: This workflow will be deployed on the drones computing infrastructure, enabling the collection of images at the deep-edge. The prediction phase will take place either at the far-edge or directly at the deep-edge, depending on the resource availability. The collected data will then be stored at the far-edge for insights collection and future reference or training purposes.

2nd workflow – Soil Organic Carbon Inference: 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.

3rd workflow - Battery Consumption Monitoring: This workflow tracks the battery consumption of drones and robots, 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.

6.3 5G-Enabled Vehicle-Assisted Services (UC3)

This section presents an initial analysis of the vehicle-assisted services use case, focused on 5G-enabled, mobility-aware orchestration and secure edge-to-cloud data management, within the EMPYREAN architecture. The use case addresses dynamic vehicular environments where computational tasks (e.g., sensor fusion, camera stream processing, or telemetry analysis) are offloaded in real time to nearby edge infrastructure to ensure continuity and safety in autonomous or connected driving scenarios.

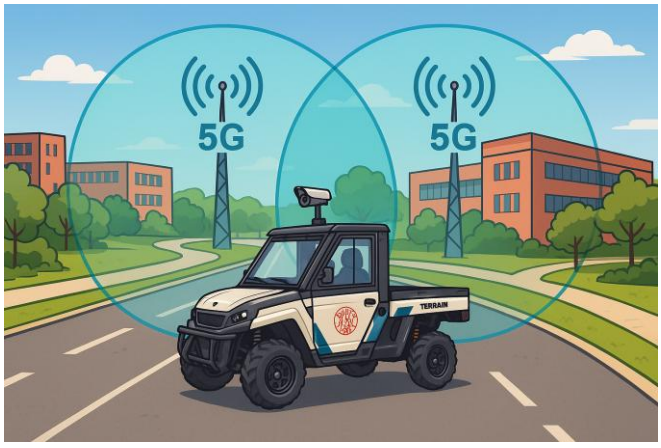

Figure 27: Overall technical development strategy and methodology

The experimentation will be carried out in the GAIA Lab testbed at the University of Murcia (UMU), which provides a private 5G and edge-cloud continuum supporting the deployment and validation of EMPYREAN components such as the Decision Engine, Telemetry Engine, CTI Module, Workflow Manager, and Skyfolk Edge Storage Gateway.

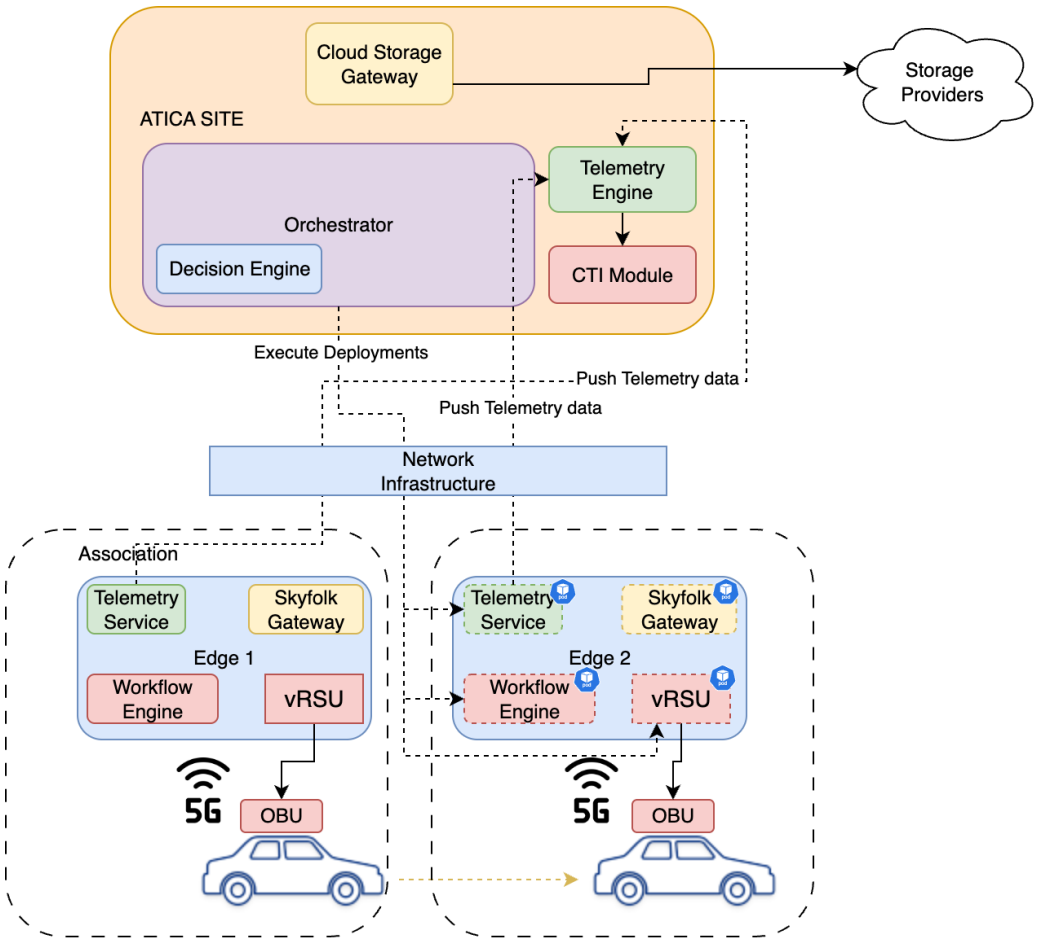

Figure 28: 5G-Enabled Vehicle-Assisted Services – EMPYREAN architecture and workflow overview

The **GAIA Lab** environment features multiple distributed 5G-connected edge nodes (e.g., ATICA, Luis Vives, GAIA, and Bellas Artes sites), interconnected through SDN and NFV-enabled infrastructure. Each site hosts an instance of the EMPYREAN orchestration stack and can operate independently or as part of a federation of Associations.

The architecture follows a hierarchical model:

- **Orchestrator layer (ATICA site)**: Hosts the **Decision Engine**, **Telemetry Engine**, **CTI Module**, and **Cloud Storage Gateway**. This layer is responsible for making global orchestration and security decisions, managing workload placement, and storing aggregated telemetry and threat intelligence data.

- **Edge layer (Luis Vives, GAIA, Bellas Artes)**: Hosts **Telemetry Services**, **Workflow Engines (RYAX)**, **Skyfolk Edge Storage Gateways**, and **virtual RSUs (vRSUs)** responsible for processing vehicular data, executing AI models, and maintaining local state.

- **Vehicle layer (OBUs)**: Onboard units connect via 5G to the nearest Association, sending telemetry and sensor data, receiving processed outputs, and seamlessly migrating between Associations as they move.

The orchestration workflow dynamically places and migrates vRSUs and telemetry modules to maintain ultra-low latency and high reliability during handovers. EMPYREAN's security components ensure that these transitions are verified (DICE + DID/VC) and protected against anomalies or attacks.

1st Workflow – vRSU Deployment and Migration

This workflow focuses on the dynamic deployment and migration of vehicular services between edge nodes as a vehicle moves through the 5G-connected campus.

Steps involved:

1. The OBU connects via 5G to Edge 1 (Luis Vives).
2. The Orchestrator deploys the vRSU and Telemetry Engine on Edge 1.
3. As the vehicle moves toward Edge 2 (GAIA or Bellas Artes), telemetry data is used by the Decision Engine to predict the upcoming handover.
4. The Workflow Manager (RYAX) triggers migration to Edge 2.
5. The Secure Boot and Digital Identity Verification (DICE + DID/VC) mechanisms ensure integrity during transition.
6. The Telemetry Service validates service continuity.
7. The CTI Engine checks for anomalies or threats during the migration process.

2nd Workflow – Edge-to-Cloud Data Storage and Management

This workflow demonstrates policy-driven storage and data management using Skyfolk's Edge Storage Gateway and Chocolate Cloud's multi-location storage system, tightly integrated with the Workflow Manager.

Key mechanisms:

- Each Association includes a pre-deployed Edge Storage Gateway and Edge Storage location configured for the vehicle's route.
- The Orchestrator can either activate gateways dynamically as vehicles enter a new Association or maintain them permanently active.
- Telemetry data (from OBD sensors) is stored in an S3 bucket using a cloud-based policy, automatically pushing data to remote cloud storage through the Association's gateway.
- Image and perception data (used for object recognition or scene understanding) are stored in edge-based S3 buckets, ensuring locality and reduced latency.
- The Workflow Manager (RYAX) coordinates the interaction between orchestration, data analytics, and storage, ensuring consistent data flows between edge and cloud environments.

7 Implementation and Delivery Plan

7.1 Software Engineering Approach

7.1.1 EMPYREAN Platform CI/CD

This section outlines the components and software tools chosen for the EMPYREAN project, detailing their roles in the Continuous Integration (CI) and Continuous Delivery (CD) process. The EMPYREAN environment comprises various software elements designed to work cohesively for integrating, testing, and deploying the platform efficiently.

In Task 5.1, our focus is on integrating, testing, and refining the EMPYREAN platform using a streamlined, lightweight approach that promotes quick deployment and ease of use across diverse environments. Given the collaborative nature of the project, involving numerous partners with varying methodologies, this approach ensures a unified CI/CD framework that can be readily adopted by all. This scenario minimizes installation and maintenance needs, allowing partners to focus on development and innovation.

7.1.1.1 CI/CD Components

To satisfy the project's need for a CI/CD procedure that guarantees all components are interconnected and tested, four key components are essential.

Version Control System (VCS)

Version control, also known as source control, is the practice of tracking and managing changes to the software code. Version Control Systems (VCS) are software tools that help software teams manage changes to source code over time. VCS keeps track of every modification to the code in a special kind of database. It consists of a remote repository of files that comprise the source code of a software application. If a mistake is made, developers can compare earlier versions of the code to fix the mistake while minimizing disruption to all team members. This component must be integrated with CI tools to monitor the VCS and trigger automated builds, tests, and deployments when it detects changes.

Continuous Integration / Continuous Delivery (CI/CD)

Continuous integration (CI) is a primary DevOps practice that allows for automation of the integration of code changes from multiple contributors to a single software project. Developers regularly commit code into a centralized repository where builds and tests then run, thus asserting the new code's correctness before integration.

The steps in a CI procedure include:

- Developers get copies of the source code and apply changes to their local system.
- The changes are then committed to the centralized, common repository.
- The server is immediately notified upon any incoming change.

- The server initiates the following actions:
  - Pulls the latest code version, including the new changes.
  - Builds the application and reports any potential problems.
  - Runs unit and integration tests, reporting any issues if they exist.
  - Releases any artifacts to be deployed for testing.
  - Assigns a build tag to the software version that was built.

Test Logs and Report

A mechanism to display results related to unitary and integration tests performed after producing testing builds of the software. A build refers to the process of compiling the source code, linking it with libraries and dependencies, and generating an executable or deployable artifact. Using this mechanism, the different log files generated during program execution and the information about the program's status will be shown and easily checked. These reports are fundamental in the CI/CD procedure as they provide insight into the status and results of different tests executed during the development cycle. The reports generated are structured in legible files containing relevant information, such as software dependencies.

Container Registry

A stateless, highly scalable central space for storing and distributing container images. Container registries provide secure image management and a fast way to pull and push images with the right permissions. The images stored in these containers are preconfigured snapshots of applications and are configured to run in various environments. Container registries are essential in CI/CD because they allow the DevOps team to manage different versions efficiently and simultaneously.

7.1.1.2 CI/CD Workflow

Once all the components needed for the deployment of the CI/CD are revisited, a CI/CD preliminary scenario composed of different tools is proposed, and the suggested workflow is described in Figure 30 and detailed next:

1. Developers commit their code to the project code repository in GitHub.
2. A webhook is triggered (Commit trigger).
3. A Build is prepared in a GitHub Actions Runner.
4. Unit testing is conducted and results are displayed directly in GitHub with GA Test Reporter.
5. Integration testing is performed, if necessary (in this case example of 2 components for concept), involving components such as the Privacy and Security Manager (PSM) and Service Orchestration (SO)
6. Based on the results:
   a. If the Build and tests pass successfully, it will be published in the GitHub Container Registry (GHCR).
   b. If the Build or any of the tests fail, the artifact is not ready to publish, and the issues must be resolved before proceeding.

This streamlined approach ensures that all partners can efficiently work together, leveraging a common CI/CD framework that supports the collective goals of the EMPYREAN project.

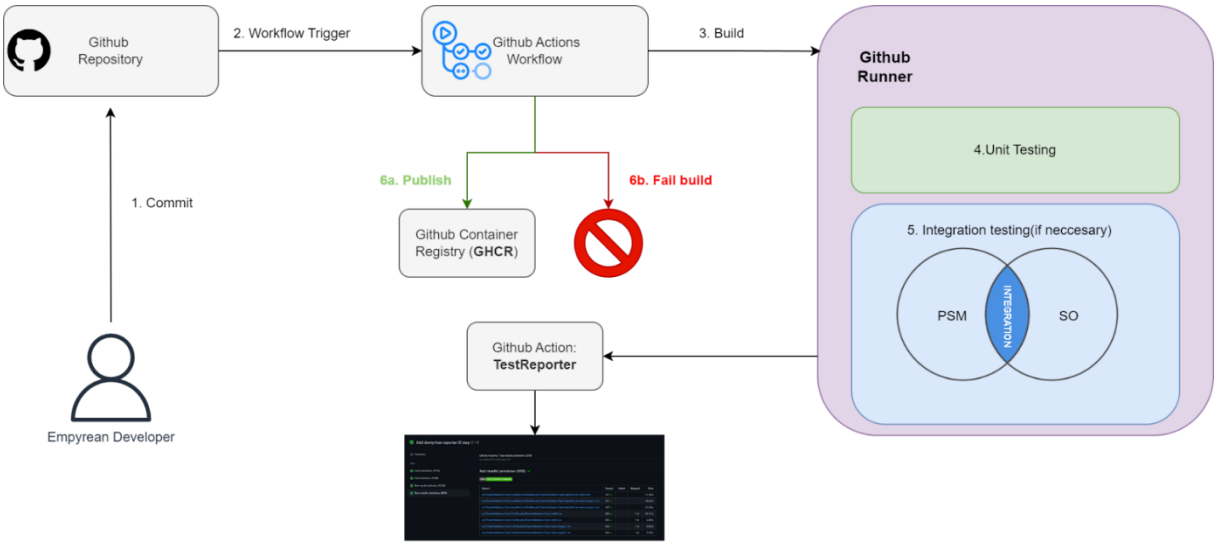

Figure 29: CI/CD workflow

7.1.1.3 CI/CD Possible tool selection

These components were selected due to their lightweight and agile nature, which eliminates the need for installation on our own servers and additional infrastructure. This plug-and-play scenario effectively meets the CI/CD requirements of the EMPYREAN project, ensuring a seamless and scalable integration process.

This streamlined approach ensures that all partners can efficiently work together, leveraging a common CI/CD framework that supports the collective goals of the EMPYREAN project.

Version Control System: GitHub

GitHub could be selected as the VCS. GitHub is an ideal fit due to its multiple advantages, such as security, integration tools, and cost-effectiveness. GitHub incorporates security features like vulnerability scanning, dependency tracking, and code scanning, helping teams identify and address security issues in their code. Additionally, GitHub natively integrates with various development tools, CI/CD platforms, and third-party services, offering a wide range of variety. Finally, GitHub offers a free plan that allows users access to a variety of resources and functionalities, including unlimited collaborators for public repositories.

Continuous Integration: GitHub Actions

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows the automation of building, testing, and deployment of pipelines. Workflows can be created to build and test every pull request to a repository or deploy merged pull requests to

production. Beyond DevOps, GitHub Actions allows workflows to be triggered when events happen in other repositories. GitHub provides Linux, Windows, and macOS virtual machines to run workflows, or can be self-hosted runners in an own data center or cloud infrastructure (e.g. runners in a own k8s cluster). GitHub has no restrictions for free Organizations that have public repositories and offers 2000 free minutes of build time per month. If these limits are exceeded, hosting running is advisable to prevent any decrease in efficiency.

Logs and Report: Test Reporter (GitHub Action)

Test Reporter is a GitHub Action that displays test results from testing frameworks directly in GitHub. It is used mainly for:

- Parsing test results in XML or JSON format to generate a visually appealing report within a GitHub Check Run.
- Automatically annotating code sections where failures occur, leveraging captured messages and stack traces from test executions.
- Offering a comprehensive evaluation of test results, including counts for passed, failed, and skipped tests.

Container Registry: Docker Hub (Public Registry)

Docker Hub is the standard registry for Docker and Kubernetes. It is a highly scalable central space for storing and distributing container images, providing secure image management and a fast way to pull and push images with the right permissions and without either administration overhead or resource costs. The only limitation with public registries is the lack of full control over their actions and the potential expense if multiple private images are needed.

7.2 Implementation Schedule

The overall work within the EMPYREAN project is organized based on a series of well-defined and complementary phases (Figure 31) that start with the definition of the use cases and the requirement analysis (Phase 1). Next, EMPYREAN adopts an iterative approach for the definition of the architecture (Phase 2), the implementation and evaluation of the individual technological developments (Phase 3), and the overall platform integration (Phase 4). The design and implementation activities will be implemented according to a spiral model with two iterations (M01-M18, M18-M36), each of which will include a series of activities that bridge the gap between requirement analysis, technology, and innovation. The developed components and services will be continuously integrated, according to the integration plan (Section 7.1), with the defined interfaces and communication protocols as set in the EMPYREAN architecture specification. The project will conclude with the demonstration of the UCs (Phase 5) and the exploitation of the results.

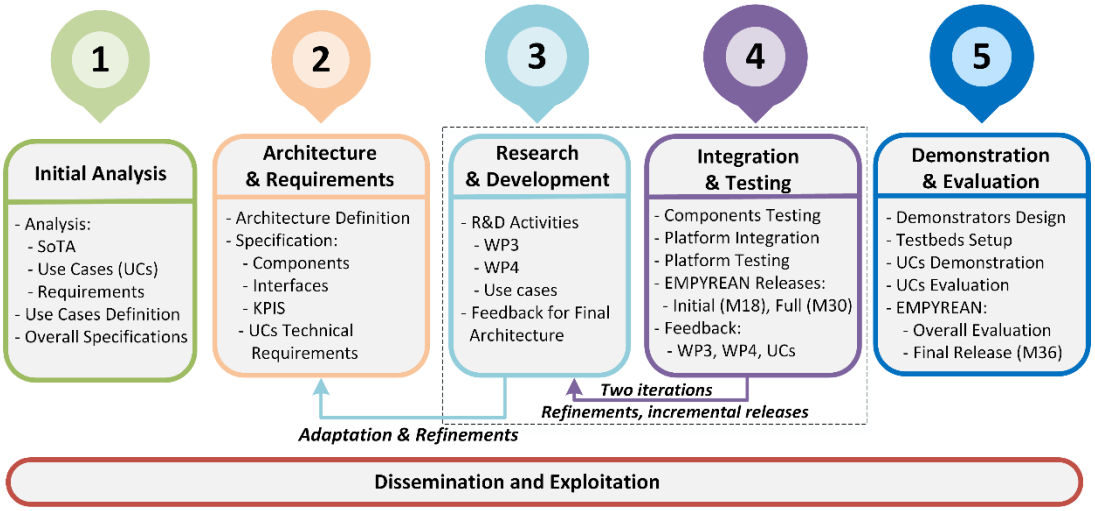

Figure 30: Overall technical development strategy and methodology

Based on the above development strategy, the three main releases (Figure 32) for the integrated EMPYREAN platform are:

- Initial platform release in M18
- Full platform release in M30
- Final platform release in M36

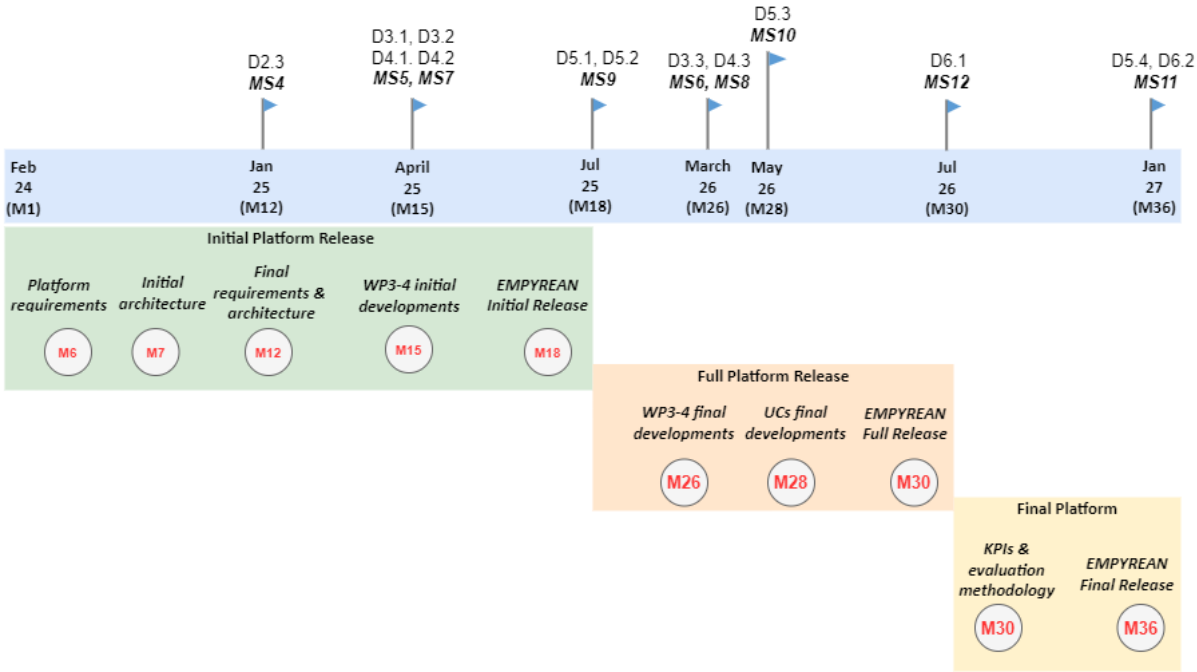

Figure 31: EMPYREAN development roadmap

The initial platform release will be the outcome of the first project development iteration (M4-M15) and will provide the partial implementation of EMPYREAN components. In this version, each component will implement a subset of the envisioned features along with the primary inter-component communication interfaces. The initial release aims to provide a basic functional prototype that will support the core functionalities of the project use cases. Deliverable D5.2 “Initial release of EMPYREAN integrated platform” (M18) will provide its description and documentation. Moreover, the initial platform prototype will provide critical feedback for the second development iteration (M18-M36).

The EMPYREAN full platform release will be based on the initial release, and will provide the remaining functionality, which has not been included in the early prototype. As the development in technical work packages (WP3, WP4) conclude by M26, this release will be provided in M30. The intention is to achieve a fully functional platform that integrates all project components and provides a prototype suitable for the pilots’ experimentation.

For the final release of the EMPYREAN platform, delivered at the end of the project, the consortium will focus on implementing the feedback from the final evaluation of the EMPYREAN platform (Phase 5) through the demonstration of the three project use cases. The outcomes from the demonstrators will be collected and documented, and videos will be prepared and uploaded to project communication and dissemination channels. This version will be fully integrated and documented as part of deliverables D5.4 “Final release of EMPYREAN integrated platform” (M36) and D6.2 “Demonstrators’ deployment and EMPYREAN evaluation” (M36). Moreover, it will be used to identify the critical and high-impact components to create the final exploitation plans.

8 Requirements Coverage

Next, we outline how the initial set of functional requirements, which were meticulously gathered, described, and categorized in deliverable D2.1 “State of the art, use cases analysis, platform requirements and KPIs” (M6), are addressed by the various components of the EMPYREAN architecture. Throughout the design phase of the initial release version of the EMPYREAN architecture, all partners closely collaborated. This collaborative effort involved iterative discussions and reviews among the technical and UC partners of the project, with the primary objective of ensuring that the EMPYREAN platform would be fully capable of delivering the specified functionalities.

The forthcoming deliverable D2.3 “Final EMPYREAN architecture, use cases analysis and KPIs” (M12) will provide a comprehensive update, including the refined analysis of use cases and platform requirements. The updated insights and evaluations will be incorporated into the final version of the EMPYREAN architecture, also detailed in D2.3, to ensure the platform’s robustness and alignment with project objectives.

Table 21: Functional requirements covered by the initial release of the EMPYREAN architecture

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

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

F_ST.4	Automated Cyber Threat Analysis.	Must-have	CTI Engine, Telemetry Engine, Persistent Monitoring Data Storage
F_ST.5	ML for Anomaly Detection and Cybersecurity.	Must-have	CTI Engine, Telemetry Engine, Persistent Monitoring Data Storage
F_ST.6	Secure and Trusted Execution.	Must-have	Secure and Trusted Execution Environment, Privacy and Security Manager, Container Runtime, Unikernel Deployment
F_DCM.1	Provide S3-compatible storage service that encompasses edge-cloud continuum.	Must-have	Edge Storage, Edge Storage Gateway
F_DCM.2	Provide an analytics-friendly erasure-coded IoT storage platform.	Must-have	IoT Query Engine, Edge Storage, Edge Storage Gateway
F_DI.1	Decentralized decision-making, speculative and multi-objective resource orchestration.	Must-have	EMPYREAN Registry, EMPYREAN Aggregator, Decision Engine, Service Orchestrator, EMPYREAN Controller
F_DI.2	Multi-agent speculative intelligent resource orchestration across EMPYREAN Associations.	Must-have	Decision Engine, Service Orchestrator, EMPYREAN Controller, Container Layers Locality Scheduler
F_DI.3	Hierarchical orchestration and multi-objective optimization for cognitive resource orchestration within Associations.	Must-have	Decision Engine, Service Orchestrator, EMPYREAN Controller, Container Layers Locality Scheduler
F_DI.4	AI-enhanced data orchestration and storage resource management within and across Associations.	Must-have	Decision Engine, Service Orchestrator, Edge Storage Gateway, Edge Storage, Decentralized and Distributed Data Manager
F_DI.5	Energy-aware workload and data distribution mechanisms.	Must-have	Decision Engine, Service Orchestrator, EMPYREAN Controller, Container Layers Locality Scheduler
F_DI.6	Monitoring and managing power and energy consumption in IoT devices and edge nodes.	Must-have	Monitoring Probes, Telemetry Engine, Analytics Engine
F_DI.7	Decentralized and AI-enabled service assurance mechanisms.	Must-have	Analytics Engine, Service Orchestrator, Telemetry Engine, Persistent Monitoring Data Storage
F_DI.8	AI-enhanced self-healing for enhanced resiliency, adaptability, and autonomous operation.	Must-have	Workload Autoscaling, Analytics Engine, CTI Engine, Service Orchestrator, EMPYREAN Controller, Telemetry Engine, Persistent Monitoring Data Storage
F_DI.9	Autonomous and adaptive workload autoscaling.	Must-have	Workload Autoscaling, Analytics Engine, Container Layers Locality Scheduling, EMPYREAN Controller, Telemetry Engine, Persistent Monitoring Data Storage
F_SO.1	Continuum-native workflow-based application design considering dataflow programming and low-code techniques.	Must-have	Workflow Manager, Dataflow Programming Component, NIX-based Environment Packaging, Application Packaging

F_SO.2	Deployment objectives (SLOs) definition for continuum-native applications	Must-have	Workflow Manager, EMPYREAN Registry, Service Orchestrator, Decision Engine
F_SO.3	Seamless and declarative orchestration of self-organized distributed orchestration systems.	Must-have	EMPYREAN Registry, Service Orchestrator, Decision Engine, EMPYREAN Controller, Container Layers Locality Scheduler
F_SO.4	Policy-based orchestration and efficient resource allocation.	Must-have	Workflow Manager, EMPYREAN Registry, Service Orchestrator, Decision Engine, EMPYREAN Controller, Telemetry Engine
F_SO.5	Context awareness and autonomous adaptive response.	Must-have	Workflow Manager, EMPYREAN Registry, Analytics Engine, Workload Autoscaling
F_SO.6	Transparent lifecycle management of hyper-distributed application components.	Must-have	Workflow Manager, Dataflow Programming, Service Orchestrator, EMPYREAN Controller, Container Runtime
F_SO.7	Coordinate workload migration within and across Associations.	Must-have	EMPYREAN Aggregator, Service Orchestrator, Decision Engine, Analytics Engine, EMPYREAN Controller
F_SO.8	Support automatic data migration operations within and across Associations.	Must-have	EMPYREAN Aggregator, Service Orchestrator, Decision Engine, Analytics Engine, EMPYREAN Controller, Edge Storage Gateway, Decentralized & Distributed Data Manager
F_SO.9	Implementation and integration of custom scheduling policies.	Must-have	Decision Engine, Container Layers Locality Scheduler
F_SO.10	Seamless orchestration and management of both container-based and serverless workloads.	Must-have	Workflow Manager, Service Orchestrator, EMPYREAN Controller, NIX-based Environment Packaging, Container Runtime
F_SO.11	Flexible Hardware-accelerated execution.	Must-have	Hardware Acceleration Abstractions, Container Runtime, Application Packaging, EMPYREAN Controller
F_SO.12	Offload acceleration to nearby devices.	Must-have	Hardware Acceleration Abstractions, Container Runtime, EMPYREAN Controller, Software-Defined Edge Interconnect
F_SO.13	OCI-compatible container images.	Must-have	Unikernel Application Builder, NIX-based Environment Packaging, Unikernel Deployment, Container Runtime
F_SO.14	Support diverse execution environments.	Must-have	Unikernel Application Builder, NIX-based Environment Packaging, Unikernel Deployment, Container Runtime, EMPYREAN Controller
F_SO.15	Reproducible Environment Packaging	Must-have	Unikernel Application Builder, NIX-based Environment Packaging, Unikernel Deployment

9 Conclusions

This deliverable provides a detailed report on the work performed in WP2, specifically within Task 2.3, which focuses on defining the initial architecture of the EMPYREAN platform. The architecture design was a collaborative effort involving all project partners, following a well-defined methodology. The goal of the architecture is to establish an IoT-edge-cloud continuum consisting of collaborative collectives of IoT devices, robots, and resources, extending seamlessly from the edge to the cloud.

The deliverable presents both the conceptual and logical architecture of the multi-layered EMPYREAN platform. Each layer is thoroughly analysed, detailing its internal decomposition into functional components and their interrelationships. This early identification of the main integration points between components is crucial, as it facilitates the overall design process and sets the foundation for subsequent implementation phases. The document also elaborates on the core functional components and provides an overview of the EMPYREAN platform deployments that will support the project's use cases. Furthermore, it introduces the development roadmap for the EMPYREAN platform, and presents how the proposed architecture addresses the challenging technical and use cases requirements outlined in D2.1 (M6).

This deliverable will serve as a guiding framework for the technical activities in WP3 and WP4, which involve the development of the key building blocks of the EMPYREAN platform. It will also support WP5 and WP6 in the development, integration, and evaluation of the project use cases, ensuring that the outcomes of WP3-6 remain relevant and aligned with the EMPYREAN objectives.

The architecture detailed in this deliverable is intended to be a living document that will evolve throughout the project. It will be refined and enhanced based on insights from ongoing WP2 activities, as well as feedback from the technological developments in WP3 and WP4. The final version of the EMPYREAN platform architecture, including detailed workflows between its components and comprehensive interface descriptions, will be reported in D2.3 "Final EMPYREAN Architecture, Use Cases Analysis, and KPIs" (M12). This final deliverable will encapsulate the complete architectural vision of EMPYREAN, ensuring a cohesive and robust framework that supports the project's ambitious goals and provides clear guidance for future developments.