D2.2 Final Release of the CECC framework and CECCM

Document Summary Information

Project Identifier	HORIZON-CL4-2022-DATA-01. Project 101093129		
Project name	Agile and Cognitive Cloud-edge Continuum management		
Acronym	AC^3		
Start Date	January 1, 2023	**End Date**	December 31, 2025
Project URL	www.ac3-project.eu		
Deliverable	D2.2. Final Release of the CECC framework and CECCM		
Work Package	WP2		
Contractual due date	M24: 31st December 2024	**Actual submission date**	M25: 31 January 2025
Type	R- Document, report	**Dissemination Level**	PU – Public
Lead Beneficiary	ISI/ATH		
Responsible Author	Elias Dritsas (ISI/ATH)		
Contributors	Adlen Ksentini, Mohamed Mekki, Sofiane Messaoudi (EUR); Dimitrios Amaxilatis, Nikolaos Tsironis (SPA); Abdelhak Kadouma, Ibrahim Afolabi (FIN); Amadou Ba (IBM); Vrettos Moulos (UPR); Dimitris Klonidis (UBI), Souvik Sengupta (ION), Elias Dritsas (ISI/ATH), Ray Carol, Ryan Jenkins, Ben Capper (RHT).		
Peer reviewer(s)	John Beredimas (CSG), Sara Madariaga (ARS)		

Funded by the European Union

AC^3 project has received funding from European Union's Horizon Europe research and innovation programme under Grand Agreement No 101093129.

Revision history (including peer reviewing & quality control)

Version	Issue Date	% Complete	Changes	Contributor(s)
v1.6	10/10/2024	5%	Initial Deliverable Structure (ToC)	Elias Dritsas (ISI/ATH), Christos Verikoukis (ISI/ATH), Adlen Ksentini (EUR)
V1.7	21/10/2024	20%	Final version of the Architecture has been introduced	Adlen Ksentini (EUR)
V1.8	6/12/2024	70%	Provided inputs for Sections 3 and Section 4.	Souvik Sengupta (ION); Mohamed Mekki (EUR); Dimitrios Amaxilatis, Nikolaos Tsironis (SPA); Abdelhak Kadouma, Ibrahim Afolabi (FIN); Amadou Ba (IBM); Vrettos Moulos (UPR); Dimitris Klonidis (UBI), Ray Carol, Ryan Jenkins, Ben Capper (RHT)
V1.9	15/12/2024	92%	Prepared the final full draft for review	Elias Dritsas (ISI/ATH)
V2.0	20/12/2024	92%	Received internal reviewers' feedback	John Beredimas (CSG), Sara Madariaga (ARS)
V2.1	27/12/2024	95%	Addressing reviewers' comments	Elias Dritsas (ISI/ATH) ; Souvik Sengupta, (ION) ; Dimitrios Amaxilatis, Nikolaos Tsironis (SPA); Ibrahim Afolabi (FIN) ; Amadou Ba (IBM), Dimitrios Klonidis (UBI).
V2.2	7/01/2025	96%	Addressing reviewers' comments	Elias Dritsas (ISI/ATH); Ray Carol, Ryan Jenkins, Ben Capper (RHT).
V2.3	8/01/2025	96%	Technical Review	Adlen Ksentini, Mohamed Mekki (EUR)
V2.5	16/01/2025	98%	Addressing technical review comments	Mohamed Mekki (EUR) ; Dimitrios Amaxilatis, Nikolaos Tsironis (SPA); Ibrahim Afolabi (FIN); Amadou Ba (IBM) ; Dimitrios Klonidis (UBI).
V2.6	30/01/2025	98%	Final adjustments	Elias Dritsas (ISI/ATH) ; Souvik Sengupta, (ION) ; Dimitrios Klonidis (UBI).
V2.7	31/01/2025	100%	Final review document	Elias Dritsas (ISI/ATH), Christos Verikoukis (ISI/ATH); Adlen Ksentini (EUR)

Disclaimer

The content of this document reflects only the author's view. Neither the European Commission nor the HaDEA are responsible for any use that may be made of the information it contains.

While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the AC^3 consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the AC^3 consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the AC^3 Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

Table of Contents

List of Figures

List of Tables

Glossary of terms and abbreviations used

Abbreviation / Term	Description
AC³	Agile and Cognitive Cloud edge Continuum management
AI	Artificial Intelligence
API	Application Programming Interface
CEC	Cloud Edge Continuum
CECC	Cloud Edge Computing Continuum
CECCM	Cloud Edge Computing Continuum Manager
CPU	Central Processing Unit
CRDs	Custom Resource Definitions
CRUD	Create, Read, Update, and Delete
GUI	Graphical User Interface
IoT	Internet of Things
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
K3S	Kubernetes (Lightweight)
LCM	Life-Cycle Management
LMS	Local Management System
ML	Machine Learning
NBI	Northbound Interface
OCRD	Operators and Custom Resource Definitions
ORS	Orchestration Service
OSR	Ontology and Semantic aware Reasoner
OCM	Open Cluster Management
OWL	Web Ontology Language
PaaS	Platform as a Service
RDF	Resource Description Framework
QoS	Quality of Service
RL	Reinforcement Learning
RU	Read and Update

SD-WAN	Software-defined Wide Area Networking
SIIF	Standard for Intercloud Interoperability and Federation
SLA	Service Level Agreement
WP	Work Package
YAML	YAML ain't markup language

1 Executive Summary

The document is deliverable *"D2.2: the final release of the Cloud Edge Computing Continuum (CECC) framework and CECC Manager (CECCM)"* and describes the final version of the high-level functional architecture of the AC^3 framework, along with its key components and interfaces, that aims to facilitate the agile and cognitive management of the CECC. *This deliverable represents the final outcome of Task T2.2, entitled "Reference Architecture for CECC".*

The federation and orchestration of cloud, edge, and far-edge computing platforms are pivotal in shaping the modern computing ecosystem. This integration enables the construction of a resilient and efficient infrastructure by harnessing the unique strengths of each platform. The result is enhanced performance, scalability, real-time responsiveness, and cost-efficiency across a wide range of applications. Cloud computing provides on-demand access to shared resources via the Internet, ensuring scalability and flexibility. Edge computing processes data near its source, significantly reducing latency, while Far-Edge computing takes this a step further, catering to ultra-low latency scenarios for critical applications.

The integration of cloud, edge, and far-edge computing platforms offers numerous transformative benefits. Firstly, it reduces latency and enhances application responsiveness, essential for real-time analytics, video streaming, and industrial automation. Secondly, it optimizes resource allocation by minimizing data transfers and ensuring efficient utilization of available resources. Thirdly, it enhances bandwidth efficiency, alleviating network congestion while bolstering overall system resilience. Finally, this integration strengthens data privacy and security, making it indispensable for sensitive applications, while also providing the scalability and adaptability needed to handle dynamic workloads. In summary, the federation of these platforms creates a cohesive and dynamic computing ecosystem that optimizes performance, responsiveness, resource management, and data privacy. This comprehensive architecture enables the efficient deployment of applications across a broad range of use cases, driving operational excellence and securing a competitive advantage in an ever-evolving digital landscape.

Building on these considerations, this deliverable provides a detailed overview of the high-level architecture of the AC^3 framework. AC^3 introduces a cutting-edge architectural design tailored to address the dynamic and evolving requirements of the CECC. The architecture is structured into three distinct planes: the User plane, the Management plane, and the CECC plane. The User and Management planes collectively form the core of the CECCM, serving as essential components to facilitate seamless and intuitive interactions with application developers. These planes are specifically designed to support developers in adopting and optimizing cloud-native applications, streamlining the transition into the Cloud Edge Continuum (CEC).

The User Plane is meticulously designed to prioritize user-friendly interfaces, serving as the central hub for developers to interact with the CECCM. This plane is equipped with robust tools and components that streamline development processes and enhance usability. Key among these is the Application Gateway, which acts as a critical bridge, enabling developers to seamlessly access and engage with the CECCM. The Catalogue offers a comprehensive repository of application descriptors and crucial data source information, empowering developers with the resources needed to build and deploy applications effectively. Additionally, the Ontology and Semantic-Aware Reasoner (OSR) is a standout feature, providing advanced capabilities to interpret and manage complex policies defined by various CECCM stakeholders. Complementing these tools are versatile interfaces that facilitate essential operations related to computing, networking, and data management tasks. Together, these components make the User Plane an indispensable part of the AC^3 framework, enhancing accessibility, functionality, and the overall developer experience.

The Management plane serves as the cornerstone of the CECCM, embodying operational excellence and robust administrative capabilities. It is meticulously designed to integrate key functionalities, including Application Lifecycle Management (LCM) and Resource Orchestration, ensuring the seamless operation of applications while efficiently overseeing the CECC infrastructure.

Particular emphasis has been placed on critical factors such as energy efficiency and the strategic geographical deployment of the Management Plane to optimize performance and sustainability. Data

management also plays a pivotal role within AC^3, governed by stringent Gaia-X specifications [1], [2] to ensure compliance with the highest standards of data sovereignty, privacy, and interoperability.

Notably, the Abstraction and Federation Layer is a standout feature of the Management plane. By streamlining the diverse Create, Read, Update, Delete (CRUD) Application Programming Interfaces (APIs) of the infrastructure layer, this layer simplifies operations, making them more intuitive and accessible. This thoughtful design reinforces the Management Plane's role as the backbone of the CECCM, driving efficiency, scalability, and innovation in the CECC. The CECC plane is more than just a structural component; it embodies AC^3's forward-thinking vision for the future of the CECC. This plane forms the backbone of the infrastructure, encompassing critical elements such as data sources and computing nodes. Its design philosophy is deeply rooted in the principles of federation, drawing from established frameworks like NIST/IEEE standards for infrastructure design and the esteemed Gaia-X model for data and service federation.

A defining feature of the CECC plane is the empowerment it offers to CECCM owners, granting them unparalleled autonomy to oversee and manage their resources. This autonomy is especially significant in scenarios where cloud service providers assume control over edge infrastructure, ensuring that CECCM owners maintain authority and flexibility. By combining a robust foundation with forward-looking principles, the CECC Plane exemplifies AC^3's commitment to creating a resilient, scalable, and federated ecosystem that meets the evolving demands of modern computing.

AC^3's commitment to data excellence is epitomized in its Data Management Platform-as-a-Service (PaaS), a sophisticated framework designed to streamline and optimize data processes. This platform is built on a three-tier structural approach, each layer playing a pivotal role in ensuring seamless and efficient data management.

- Northbound API: Serving as the gateway to the Data Catalogue, Monitoring Module, and Data Federation Services, this layer acts as the custodian of AC^3's data ecosystem. It meticulously manages an extensive data inventory, enables real-time monitoring of data streams, and enforces robust data access controls, ensuring compliance with stringent security and governance requirements.
- Data Management Engine: At the heart of the PaaS, this engine integrates the Data Manipulator, Internal Data Broker, and Semantic Reasoner, working in harmony to transform, optimize, and intelligently manage data. It enables sophisticated data transformations, streamlines data flow, and employs semantic methodologies to support informed decision-making and advanced data reasoning.
- Southbound API: Dedicated to integration, this layer includes tools like Data Mappers and Data Source Connectors to facilitate seamless connectivity with external data repositories. It ensures data consistency through a unified format, enhancing interoperability and integration across diverse systems.

In summary, AC^3's architecture exemplifies a perfect blend of innovation and functionality, delivering a robust and user-centric data management framework. By simplifying complex data processes through its diverse modules and interfaces, AC^3 sets a new standard in architectural excellence, empowering organizations to harness the full potential of their data in an ever-evolving digital landscape.

This document represents the final deliverable of Task T2.2 and serves as a critical input for Task T2.3. In Task T2.3, all relevant technological tools have been carefully selected to implement and develop the various components of the CECCM and the Local Management System (LMS). These developments will align with the predefined architectural model and adhere to the target metrics established earlier in the project. By bridging the conceptual framework of T2.2 with the practical execution in T2.3, this document lays the foundation for the successful realization of AC^3's innovative vision.

2 Introduction

In the era of digital interconnectivity, the growing demand for real-time data processing, low-latency interactions, and optimized user experiences has underscored the need for seamless integration of cloud, edge, and far-edge technologies. This synergy is vital for establishing a federated computing continuum that maximizes the efficiency of centralized cloud resources while harnessing the immediacy and proximity of edge devices. Such integration ensures not only uninterrupted data flow and processing but also the capability to manage the vast influx of data from diverse sources effectively.

In this context, the AC^3 project is poised to revolutionize federated computing with its innovative architecture, which introduces a comprehensive CEC infrastructure that extends seamlessly to the far edge. At the core of this transformative vision lies the CECCM, which serves as the central pillar of the initiative. CECCM highlights the importance of resource federation, spanning centralized cloud hubs to the farthest edge nodes. This extensive integration is designed to enhance resource availability while embedding robust trust and security mechanisms, ensuring the system operates with superior resilience and adaptability. By addressing the challenges of modern data-driven ecosystems, AC^3 is redefining the boundaries of federated computing, delivering a future-ready solution that meets the evolving demands of real-time, secure, and scalable digital environments.

A pivotal aspect of the AC^3 framework is its data management strategy, which serves as a foundation for streamlined application development and deployment. By embedding Data Management as a PaaS within the CECCM, AC^3 establishes a robust, user-centric approach to handling data. This strategy encompasses the entire data lifecycle, including retrieval, storage, and monitoring, ensuring seamless and efficient operations. AC^3 places significant emphasis on delivering an intuitive interface that simplifies data interactions. By leveraging web semantics and ontologies, the framework enhances usability, making data requests straightforward and accessible for developers. Complementing this user-oriented approach is AC^3's commitment to innovation, adopting zero-touch management and Application LCM methodology. Utilizing machine learning (ML), artificial intelligence (AI), and context-aware insights, AC^3 boosts predictive capabilities, enabling the CECCM to make informed strategic decisions, from microservice placements to lifecycle optimization. Sustainability and adaptability form the core elements of the AC^3 project. By championing a green-centric, zero-touch approach to infrastructure management, powered by AI/ML insights, AC^3 ensures energy-efficient operations while maintaining optimal application service levels. This sustainable direction strikes a balance between reducing energy consumption and delivering high performance. Additionally, AC^3 underscores the importance of network programmability within the CECC continuum. Leveraging advanced technologies like Software-Defined Wide Area Networking (SD-WAN), the framework delivers agile responses to network dynamics, ensuring resilience and adaptability. This holistic approach exemplifies AC^3's vision of a dynamic, responsive, and sustainable computing ecosystem, setting a new benchmark for the future of federated cloud-edge environments.

Building on a set of functional and non-functional requirements, including those derived from the project's use cases and outlined in deliverable D2.4, the initial version of the CECC component's specifications and interfaces within the AC^3 architecture was detailed in deliverable D2.1 [3]. However, this document aims to present the consortium's final iteration of the AC^3 architectural framework—an advanced and intricately structured system composed of 3 distinct planes: the User plane, Management plane, and Infrastructure plane. Each plane is meticulously designed with functional components that leverage state-of-the-art, well-defined interfaces, enabling seamless communication both within individual planes (intra-plane) and across different planes (inter-plane). The updates and modifications are dedicated to the Application Gateway and Management plane. In the former, KPI collection and exposure are newly introduced, while Service Catalogue and Graphical User Interface (GUI) sub-components were changed to Catalogue and Northbound API Engine, respectively. As for the latter, Data management was removed from the Management plane. Further details and justifications are provided in Table 2.

At the heart of this sophisticated architecture is the CECCM, envisioned as the cognitive core of the entire framework. The CECCM is imbued with dual functionalities: the precise management of application life cycles and the intelligent orchestration of CECC infrastructure resources. This dual functionality underscores its central role and is powered by an advanced fusion of ML, AI, and innovative semantic and context-aware algorithms. The CECCM's design ensures a balanced and efficient operation, delivering on two critical commitments: a steadfast focus on energy efficiency and unwavering adherence to the stringent standards of Service Level Agreement (SLA) benchmarks. By integrating these advanced capabilities, the CECCM exemplifies a transformative approach to managing and orchestrating the CECC, setting new standards for cognitive, sustainable, and efficient computing.

Finally, this document provides workflow schematics related to application deployments and AC^3's distinctive Data Management PaaS**.** These schematics, both comprehensive and meticulously detailed, offer invaluable insights into the operational mechanics of the AC^3 framework. They highlight its holistic and innovative approach to managing the CEC, demonstrating the seamless integration of its components and their collective contribution to efficient and adaptive system operations.

2.1 Mapping AC^3 Outputs

The purpose of this section is to map AC^3 Grant Agreement commitments, both within the formal Deliverable and Task description, against the project's respective outputs and work performed.

Table 1: Adherence to AC^3 GA Deliverable & Tasks Descriptions.

AC^3 GA Component Title	AC^3 GA Component Outline	Respective Document Chapter(s)	Justification
DELIVERABLE			
D2.2 Final Release of the CECC framework and CECCM			
TASKS			
CECC Component's specifications and interfaces	**Task T2.2:** This task will be dedicated to defining a detailed specification of the CECC architecture, key components composing the CECCM (building blocks), their roles, functions, and features, and the required communication interfaces and protocols to perform federation and interact with the different infrastructures in a secure and trusted manner. Also, it will select the federation model to be considered in AC^3, and accordingly define a unified API to be used by the infrastructure providers since AC^3 relies on a resource	Section 3	Presents the final functional architecture with detailed specification of functional blocks and the inter-plane and intra-plane interfaces

	federation architecture involving different infrastructure types such as the core cloud, edge, far edge, network equipment, and data sources, from different un-trusted domains.		
Defining the Data management Model	**Task 2.2: In** this task, the data management model relying on the concept of PaaS will be defined in terms of functions and interfaces with the other CECCM components, the data sources and the application developer.	Section 3.6, Section 4	Presented the final data management model and also the sequence diagrams for the corresponding operations of data management model.

2.2 Deliverable Overview and Report Structure

In this section, a brief description of the deliverable's structure is provided as follows.

Section 3 describes the components of the final version of the AC^3 architecture, including the User plane, Management plane, and Infrastructure plane. Also, this section analyses the Data Management components, which consists of Data Catalogue, Data Connectors, and Application Add-ons. Finally, it summarizes and explains the Interfaces.

Section 4 presents in different subsections the workflows of Application Deployment with/without hot data, with cold data, Run-time Management and Application Deletion.

3 AC³ High-level architecture

3.1 New architecture

Figure 1 illustrates the final high-level architecture of AC³. It builds upon the initial architecture presented in deliverable D2.1 [3], incorporating feedback and requirements from WP3—specifically from T3.1, T3.2 and T3.4—regarding interactions with application developers and data management.

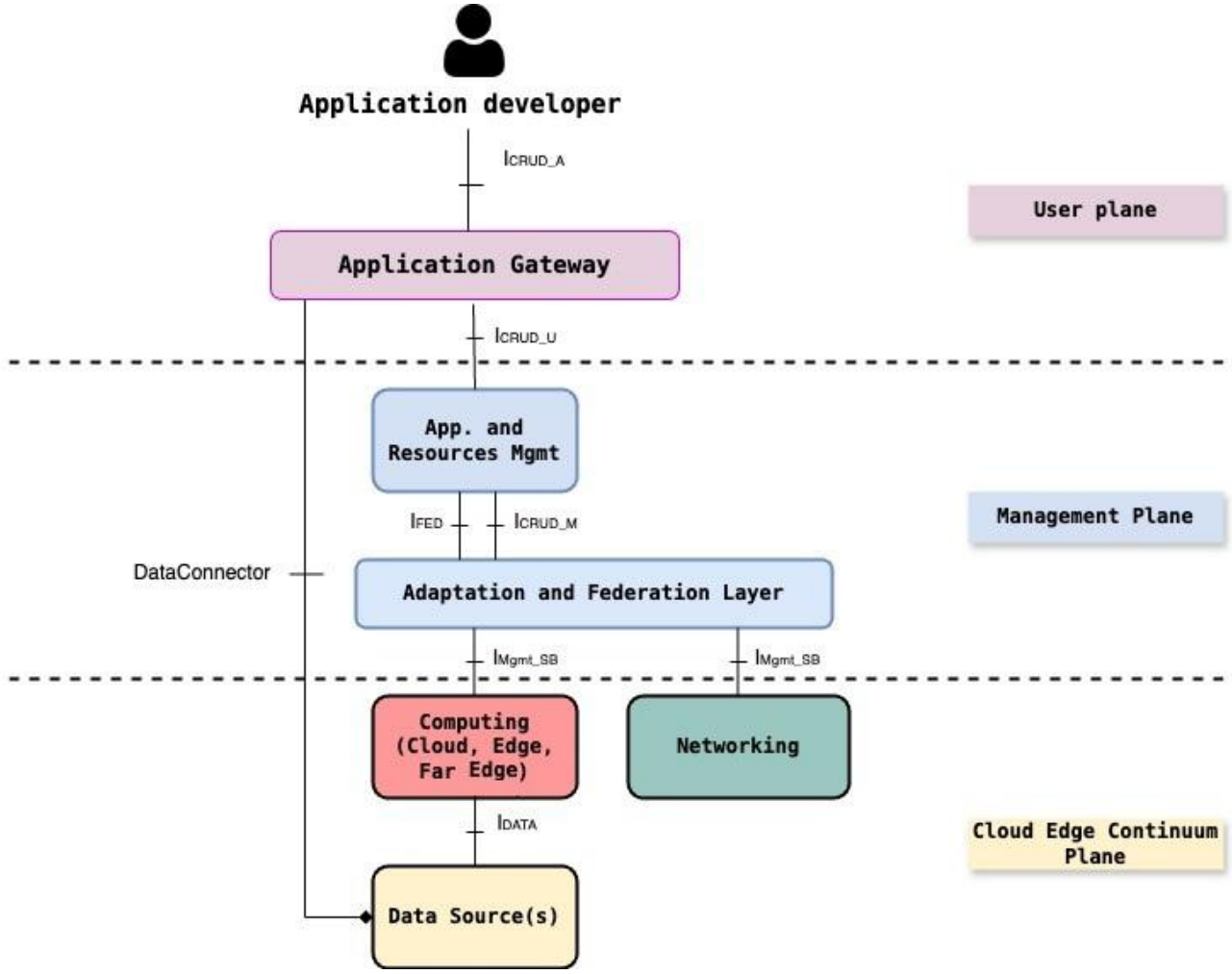

Figure 1. High-level architecture of AC³.

The primary objective of the AC³ architecture is to manage the lifecycle of service-based applications on a CECC infrastructure by utilizing advanced ML and AI algorithms. This ensures application Quality of Service (QoS) while optimizing CECC infrastructure resources. The unique features of the architecture include:

- Dynamic Application Definition: Enables application developers to define applications dynamically (including components and SLAs) through an Application Gateway that supports both a Northbound Interface (NBI) and a GUI.
- Microservice Composition and Reuse: Facilitates the composition of microservices to handle data sources and allows reuse of common microservices. This is achieved through the use of one composite catalogue, which contains the data description and service description.
- Application Descriptor Generation: Supports the generation of an application descriptor, which is enforced on the CECC infrastructure using AI-based LCM.
- Zero-Touch Orchestration and Management: Enables zero-touch orchestration and management of

the application lifecycle within the CECC architecture by leveraging advanced AI/ML models.

- Energy-Aware Resource Management: Optimizes resource management on the CECC infrastructure using AI/ML models to enhance energy efficiency.
- Dynamic Resource Composition: Allows for the dynamic composition of CECC resources (computational, networking, and storage) to meet application requirements effectively.
- Federation of CECC resources: This is achieved using the IEEE Standard for Intercloud Interoperability and Federation (SIIF) [4].

To support these features, we adopted a modular approach in defining the AC^3 architecture. This approach involves devising a set of independent modules that work together to form the CECCM, a key output of the AC^3 system. These modules are defined by the specific tasks they are designed to accomplish and are aligned with the objectives mentioned earlier.

Each module contains a set of independent components that interact with one another. For example, the Application Gateway module includes Catalogue, OSR, and Key Performance Indicator (KPI) collection and exposure. Figure 1 provides a high-level view of the AC^3 architecture, depicting all the modules and their corresponding components that collectively form the AC^3 CECCM. Meanwhile, Figure 2 illustrates all the interfaces and sub-components (detailed in Table 3 and Table 4) used by the components to: (1) Communicate within the CECCM; (2) Interact with external actors of the CECCM, such as the federated infrastructure and application developers.

The modules (and hence components) that make up the CECCM are grouped into two distinct planes: the user plane and the management plane. The figures also illustrate the CEC plane, which corresponds to the federated infrastructures leveraged by the CECCM to deploy application micro-services.

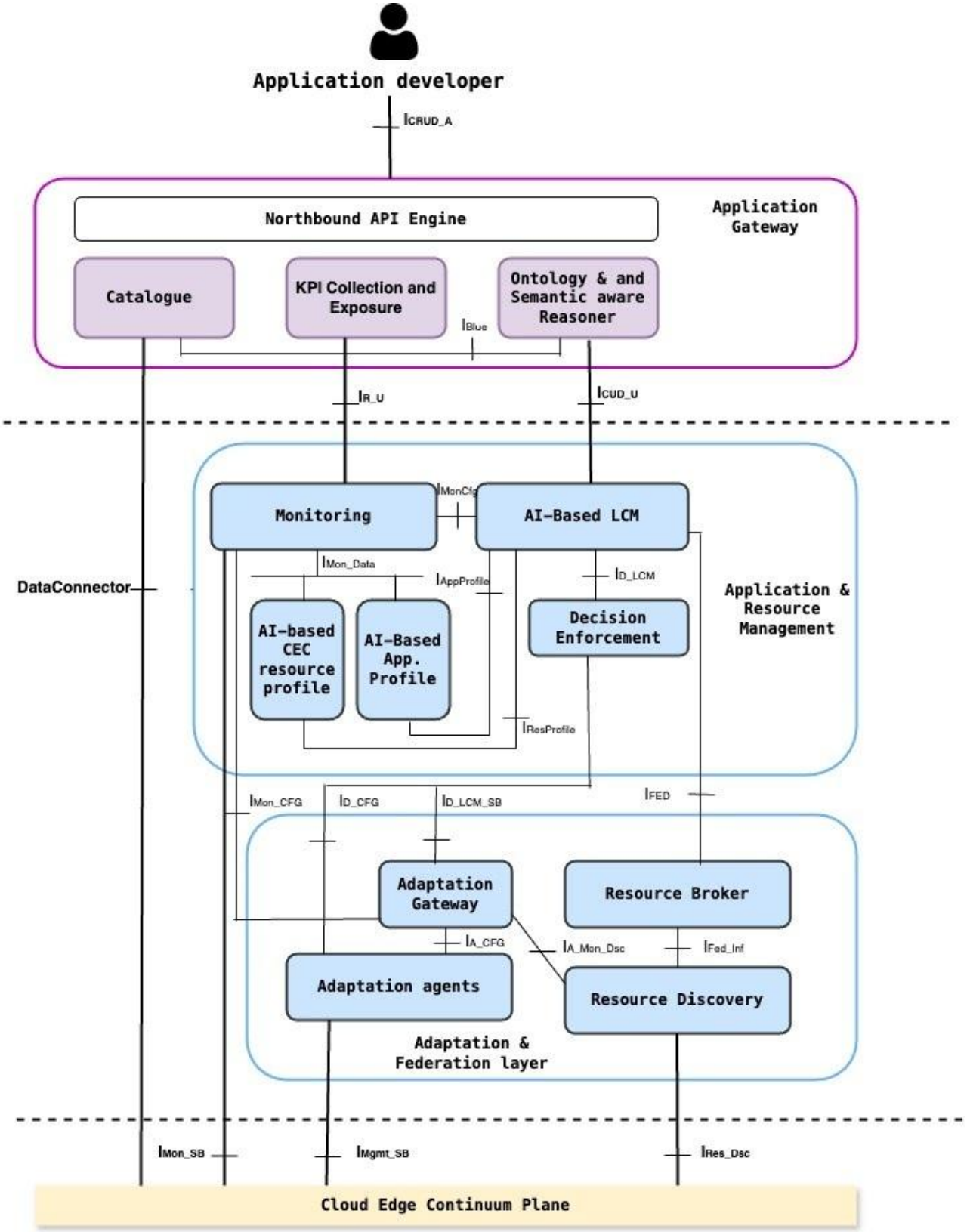

Figure 2. Holistic vision (all components) of the AC³ architecture.

All AC³ components communicate through well-defined interfaces (see Table 3 and Table 4), facilitating seamless integration and implementation. Furthermore, the modular design ensures that the CECCM architecture remains scalable and user-friendly for future enhancements. If additional functionalities or

components are introduced—or changes to existing modules are required—only the relevant components need to be updated, leaving the rest of the system unaffected.

This modularity allowed us to easily evolve the architecture from its initial version to this updated version, which required adaptations to data management and interaction processes.

The key components of the architecture include:

- **User plane (Application Gateway):** The role of the unique module of the User plane, namely the Application Gateway, is to be the interface with the end-user for CRUD operations, to define and compose the application, deploy the application, and monitor the application during their Life Cycle. The Application Gateway is constituted by:
    - **Northbound API Engine**: it interacts with the application developers through a GUI and a well-defined REST API for CRUD operations.
    - **Catalogue**: One catalogue is integrated as part of the AC^3 architecture where the data description of the catalogue allows to search and connect data sources supporting both cold data and hot data to the application. The service description of the Catalogue contains a repository of existing micro-service blueprints, which can be composed into applications. Examples include databases, security micro-services, and other reusable components.
    - **OSR**: It is a key element of the Application Gateway module. It enables developers to describe the micro-services that form an application by defining the associated software images and specifying SLAs, i.e., requirements related to computing resources, networking resources, and application availability. It interacts with the Catalogue to support the composition of micro-services allowing developers to integrate additional micro-services, such as add-on services that ensure data management when the application requires access to external data sources.
- **Management plane:** It is composed of two modules, the **Application and Resource Management** and **Adaptation and Federation Layer**.
    - **Application and Resource Management:** This component leverages advanced ML and AI algorithms to achieve two primary objectives. First, it manages the life cycle of micro-services—including deployment, run-time management, and deletion—while ensuring compliance with the SLAs defined by the application developer. Second, it optimizes resource utilization by balancing resource usage, energy consumption, and SLA guarantees, finding an effective trade-off between these often-competing factors.
        - **Monitoring**: It is a critical component as it is in charge of collecting data from the different CEC infrastructures on the computing usage of micro-services and metrics on the link as well as the resource exposed by the infrastructure (such as available computing resources, type of energy used, location, etc.). The collected data are formatted into a common data model and made available to the AI-based application and CEC resource profiles, while micro-service-related metrics are presented to the **Developer** through the **KPI Collection and Exposure** module of the Application Gateway.
        - **AI-based application profile**: it uses AI/ML algorithm to build application profiles that will be used by the AI-based LCM for the LCM decisions.
        - **AI-based CEC resource profile**: it uses AI/ML algorithm to build CEC resource profiles that will be used by the AI-based LCM for the LCM decisions.
        - **AI-based LCM**: It is the core component of the Application and Resource Management module. It is in charge of the LCM of the applications by combining the profiles built by the application and the CEC resource as well as the local AI/ML models (mainly based on Reinforcement Learning - RL) to derive optimal decisions (such as placement, migration, resource scaling, network overlay updates) to ensure optimal management of the application and CEC resources.
        - **Decision Enforcement**: The role of this component is to enforce the AI-based LCM decisions on the federated CEC infrastructure. Decision enforcement typically involves

horizontal scaling, vertical scaling or migration decisions and uses a common data model, regardless of the infrastructure used.

- **Adaptation and Federation Layer:** Its role is to abstract the access to the infrastructures to the Application and Resource Management components by defining a common API that will be translated to LMS-specific NBI to collect monitoring data or to enforce LCM decisions on the CEC infrastructure. Also, this layer is in charge of access to the federation management system and selecting the needed infrastructure components.
  - **Resource Broker**: It selects the needed infrastructure from the federation to be used to deploy micro-service. The broker uses specific algorithms from deliverables D2.5 [5] and D2.6 (the final version of D2.5) to select an infrastructure provider from the federation.
  - **Resource Discovery**: This component leverages the NBI endpoints to discover available resources exposed by the federated infrastructure.
  - **Adaptation Gateway**: Once an infrastructure is selected from the federation, the adaptation gateway selects the agent that needs to be used to translate the common deployment API exposed by the module to an LMS NBI (discovered by the resource discovery component).
  - **Adaptation Agent**: It contains a set of LMS-specific agents and translates the common API to an LMS NBI. We envision an agent for each LMS technology. For instance, one agent for Kubernetes, one for OpenShift, one for K3S, one for SD-WAN controller, etc.

3.2 Major changes by the report to the initial version of the D2.1

As mentioned earlier, the AC^3 architecture has been updated, with modifications made relative to the initial version. While the architectural approach remains modular, certain components have been updated or removed to reflect advancements achieved in WP3, particularly in T3.1, T3.2, and T3.4. These tasks are detailed in deliverables D3.1 [6] and D3.3 [7], respectively, and focus on enhancing interaction with application developers and improving the management of data and data sources associated with applications.

The changes specifically impact the Application Gateway module and the Data Management component within the Management plane. The updates were guided by the need to address i) the defined requirements in deliverable D2.1- 1st Release of the CECC framework and CECCM and ii) streamline interactions between components. Table 2 provides a summary of the changes and the rationale behind these updates.

Table 2: Changes between the initial and final version of the architecture.

Module	Component	Change	Reason
Application Gateway	Service Catalogue	Catalogue	The component has been renamed AC^3 Catalogue to reflect its updated functionality. It now encompasses two distinct descriptional parts: one dedicated to micro-service blueprints and another for registering and connecting data sources to applications through the data description of the Catalogue.
Application Gateway	GUI	Northbound API Engine	The CECCM will offer both a GUI and a REST API to allow application developers

			flexibility in defining applications. The REST API facilitates seamless integration with other orchestration or management systems, enabling them to utilize CECCM for running applications.
Application Gateway	KPI collection and exposure	Newly introduced	This newly introduced component enables application developers to visualize KPIs related to the micro-services composing their applications. The KPIs are specific to the application and include metrics such as CPU usage, memory consumption, storage utilization, and network traffic (inbound and outbound) for the micro-services. The KPIs can be presented in two formats: as raw data for detailed analysis or through a dashboard for user-friendly visualization.
Management plane	Data management	Removed	This component has been removed from the Management plane. Following a more detailed analysis of data management procedures and requirements in T3.4, it was determined that this module is no longer necessary within the management layer. Instead, the approach has been streamlined by enabling application developers to connect data sources directly to applications via the data catalogue. The OSR component then handles micro-service composition, automatically adding AC^3 data management micro-services (i.e., Data Consumer connectors, Data mappers, Message broker, and Data

			Manipulator) to the application deployment process. These micro-services ensure that data sources are seamlessly attached to the application upon deployment within the CEC infrastructure. This procedure applies to both cold data and hot data.

3.3 User plane components

The Application Gateway sits atop the CECC architecture, serving as a primary interface for developers to interact with and manage applications within the system. The Application Gateway provides a comprehensive GUI that streamlines the interaction with the CECCM, enabling developers with a way to execute the CRUD operations efficiently. Designed to simplify application lifecycle management, the Application Gateway focuses on delivering an intuitive and user-friendly experience, abstracting the underlying complexities of the CECC infrastructure and exposing key functionalities that facilitate application deployment, monitoring, and management.

3.3.1 Graphical User Interface (GUI) and Northbound API

The GUI serves as an intuitive entry point, complemented by a comprehensive Northbound API, enabling developers to interact with the CECC in a visually structured or programmatically accessible environment. This dual-interface approach ensures that all functionalities accessible through the GUI are also available via the API for automation and integration with other systems. The dashboard simplifies complex operations through guided workflows and organized data structures, making it easy to configure, monitor, and manage applications effectively.

The CRUD operations provided by the GUI and mirrored in the API are:

- **Create**: Users can create new application descriptors through either a multi-step form or by uploading a JSON file containing a pre-configured model. The multi-step form guides users to provide essential details across various sections, such as application metadata (name, version, description), microservices (with specifications on service type, dependencies, and resource requirements), deployment context (cloud platform, geographical deployment, containerization settings), network requirements (traffic patterns, load balancing strategies, rate-limiting), and security policies (access control, firewall settings, data encryption, SLA parameters). This approach ensures all necessary components are defined for successful deployment. Alternatively, users experienced with JSON can quickly upload a fully defined descriptor model, expediting setup and allowing greater customization flexibility.
- **Read**: The dashboard presents each application descriptor visually, making it easier for developers to review and understand application configurations. Users can explore descriptor details through collapsible sections, such as those for microservices, deployment, and security, which can be expanded and collapsed for smoother navigation. Additionally, the dashboard provides an option to view descriptors in a YAML format, chosen for its readability and suitability for configuration tasks. This YAML view not only enhances the user experience but also provides a straightforward, ready-to-use format for downstream deployment by other system components.
- **Update**: The GUI enables users to make updates to application descriptors before deployment, allowing seamless adjustments to various parameters, such as resource allocation and security settings, directly. For users who have external modifications, the system also supports re-uploading

a JSON file with the updated configurations, an efficient method when multiple descriptors share similar modifications. The update feature facilitates iterative development, helping users refine configurations and track changes.

- **Delete**: When application descriptors are no longer needed, users can remove them from the system, optimizing resource allocation and keeping the dashboard uncluttered. The descriptor is permanently removed from the database, releasing storage resources and allowing developers to maintain focus on active, relevant configurations.

Additionally, the GUI and API facilitate robust application monitoring, providing real-time insights into application performance, resource usage, and operational status. This monitoring capability is integral to the lifecycle management of applications, ensuring that developers have the tools necessary to track application health and make informed decisions based on comprehensive data analysis.

3.3.2 OSR

The OSR within the AC^3 framework's Application Gateway, serves as an intelligent bridge between user-defined application requirements and the technical configurations needed for cloud-edge deployment. By leveraging semantic web technologies, such as ontologies and reasoners, the OSR transforms these requirements into a machine-processable representation, capturing the cloud-edge computing domain, microservices, and application management rules in a structured format. This capability enables the OSR to deliver optimized, context-aware LCM tailored to the dynamic AC^3 infrastructure.

The OSR begins by processing inputs from a GUI/Northbound API where users specify essential application requirements, including metadata, consumers, microservices, data sources, deployment configurations, network, and SLAs. These inputs form a foundational application profile that allows the OSR to understand the domain's actors, such as data sources, applications, users, and the cloud-edge computing infrastructure. This profile serves as a guide for the OSR to generate an enriched application composition file, ensuring compatibility and optimal performance within the AC^3 ecosystem.

Following the establishment of user-defined application requirements, the OSR interfaces with the relevant Data and Service Catalogue (see Section 3.3.3 on Catalogue) to retrieve data and service descriptors that meet the specific needs of the application. These catalogues offer structured descriptions of data sources and microservices, detailing attributes like data formats, access protocols, service capabilities, dependencies, and resource requirements. By dynamically aligning user requirements with the catalogue descriptors, the OSR ensures that the resulting application composition is populated with accurate technical specifications, creating a robust and tailored application environment.

The OSR's final output is an enriched application composition file, a structured document that captures all essential configurations, resource allocations, dependencies, and network details. Figure 3 provides an Application Descriptor Template that consolidates these parameters—including microservice configurations, resource requirements, and SLA constraints—into a single, coherent YAML blueprint. This file is then sent to the LCM component, which orchestrates deployment across the CECC. By transforming static requirements into dynamic, machine-processable deployment instructions, the OSR introduces an intelligent semantic layer between high-level user policies and the actual application runtime environment. This allows AC^3 to optimize resource allocation, application performance, and scalability across diverse cloud-edge environments, delivering a seamless pathway from user-defined requirements to concrete technical deployment.

```
ApplicationName: "[ApplicationName]"
Version: "[Version]"

Microservices_configuration:
  - MicroserviceName: "[MicroserviceName]"
    Version: "[Version]"
    Image: "[Image]"
    ID: "[ID]"
    Dependencies: ["[Dependency1]", "[Dependency2]"]
    ResourceRequirements:
      Cpu: "[Cpu]"
      Memory: "[Memory]"
      Storage: "[Storage]"
      Gpu: "[Gpu]"
    MicroservicesSLAs:
      ServiceAvailability: "[ServiceAvailability]"
      MaxResponseTime: "[MaxResponseTime]"
      DataThroughput: "[DataThroughput]"
    ReplicaCount: "[ReplicaCount]"
    EnvironmentVariables:
      - Name: "[EnvVarName]"
        Value: "[EnvVarValue]"
    apiEndpoint: "[apiEndpoint]"
    apiPort: "[apiPort]"
    Protocol: "[Protocol]"
    InternetAccess: "[InternetAccess]"
    GeographicalArea:
      Region: "[Region]"
      LocationType: "[LocationType]"

Networking_graph:
  - Source: "[Source]"
    Destination: "[Destination]"
    Protocol: "[Protocol]"
    Port: "[Port]"
    ConnectionSLAs:
      Latency: "[Latency]"
      Availability: "[Availability]"
      Bandwidth: "[Bandwidth]"
      ErrorRate: "[ErrorRate]"

Global_SLA:
  ServiceAvailability: "[ServiceAvailability]"
  MaxLatency: "[MaxLatency]"
  MaxResponseTime: "[MaxResponseTime]"
  DataThroughput: "[DataThroughput]"
```

Figure 3. Application Descriptor Template.

3.3.3 Catalogue: Data and services

Data and Services Catalogue component serves as an important framework for the systematic management and accessibility of resources within the AC³ CECC. Leveraging the advanced Gaia-X conformant catalogue framework (e.g., Piveau [9]), this component underpins a robust infrastructure, facilitating the efficient indexing, organization, and dissemination of data and services across multifaceted, distributed computational environments. Features and functional capabilities of the "Catalogue" are explained in the subsections below.

3.3.3.1 Service Discovery and Metadata Management

- **Comprehensive Metadata Indexing:** For CECC, each service within the AC³ Catalogue is meticulously indexed with comprehensive metadata, which encompasses functionality, dependency information,

performance metrics, and compatible data formats. This detailed indexing facilitates accurate discovery and selection of services to match specific user requirements efficiently.

- **Semantic Metadata Enrichment:** Incorporates metadata enrichment technologies employing semantic web standards such as RDF (Resource Description Framework) and OWL (Web Ontology Language). This ensures uniformity in metadata interpretation and reuse across diverse CECC infrastructures, significantly enhancing interoperability.

3.3.3.2 Integration with Ontology and Semantic-aware Reasoner (OSR)

- **Semantic Alignment**: Catalogues interact directly with the OSR to align semantically the high-level requirements articulated by users with the underlying technical efficacy of services. This ensures a precise mapping of user intentions to available strategic resources.
- **Enhanced Discoverability**: Utilizes ontological schemas to heighten service discoverability, optimizing the utilization of resources within the CECC through efficient search and retrieval mechanisms.

3.3.3.3 APIs for Catalogue Management

- **RESTful API Support**: Offers RESTful APIs enabling the execution of CRUD operations on catalogue entries. This ensures a flexible and programmatic approach to catalog management.
- **Standards Compliance**: Aligns API endpoints with Gaia-X conformant, facilitating compatibility with external catalogue systems and enhancing interoperability.
- **Advanced Querying Capabilities**: Provides sophisticated querying options based on metadata attributes or specific ontologies, allowing precise retrieval of desired services.

3.3.3.4 Performance and KPI Integration

- **Real-time Monitoring**: The AC^3 framework enables real-time monitoring of services across the CECC by capturing critical performance metrics, including CPU usage, memory utilization, network throughput, and latency. These metrics are gathered through a unified data model, ensuring consistency and eliminating the need for manual data transformation. By delivering instantaneous insights into system behavior, real-time monitoring facilitates the swift detection of bottlenecks and performance issues, thereby ensuring services operate optimally while maintaining reliability and responsiveness.
- **Informed Deployment Decisions**: The AC^3 framework empowers application developers to make data-driven deployment decisions by utilizing both real-time and historical performance data. The LCM module incorporates KPI metrics into its decision-making processes, optimizing resource allocation, microservice placement, and scaling strategies. By analyzing performance trends against SLA requirements, the system ensures applications are deployed on the most suitable resources. This approach not only enhances application performance but also promotes energy efficiency and operational scalability within the CECC.
- **Integration with KPI Collection Layer**: The KPI collection layer serves as a central repository for performance data, aggregating metrics from monitoring modules across the CECC. This layer standardizes data outputs, enabling seamless integration with other components of the AC^3 framework. Developers and operators can access these insights through intuitive dashboards and APIs, which offer both detailed technical analysis and high-level performance visualization. By directly interfacing with this layer, the AC^3 framework provides a continuous feedback loop for performance evaluations, facilitating proactive adjustments and ensuring sustained operational excellence.

3.3.3.5 Self-Description Support

- **Self-Describing Metadata**: Utilizes standardized self-describing metadata, ensuring resources are interpretable by any system component seamlessly. Figure 4 is an example of the self-describing metadata.

```
@prefix dcat:   <http://www.w3.org/ns/dcat#> .
@prefix dct:    <http://purl.org/dc/terms/> .
@prefix foaf:   <http://xmlns.com/foaf/0.1/> .
@prefix locn:   <http://www.w3.org/ns/locn#> .
@prefix owl:    <http://www.w3.org/2002/07/owl#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix skos:   <http://www.w3.org/2004/02/skos/core#> .
@prefix time:   <http://www.w3.org/2006/time> .
@prefix vcard:  <http://www.w3.org/2006/vcard/ns#> .
@prefix xml:    <http://www.w3.org/XML/1998/namespace> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .

<https://piveau.io/id/catalogue/test-catalog>
    a               dcat:Catalog ;
    dct:type        "dcat-ap" ;
    dct:title       "AC3 Test Data Catalogue"@en ;
    dct:description "This is a testing ground Catalog for the AC3 project"@en ;
    dct:language    <http://publications.europa.eu/resource/authority/language/ENG> ;
    dct:spatial     <http://publications.europa.eu/resource/authority/country/EUR> ;
    dcat:dataset    <https://piveau.io/set/data/test-dataset> ;
    dcat:record     <https://piveau.io/set/record/test-dataset> ;
    dct:publisher   [ a             foaf:Agent ;
                      foaf:homepage <https://ac3-project.eu/> ;
                      foaf:mbox     <mailto:info@ac3-project.eu> ;
                      foaf:name     "AC3 Project" ] .
```

Figure 4. An example of Self-description metadata of the AC³ data-service catalogue (based on modified Piveau catalogue).

- **Interoperability and Scalability**: Advocates for the use of standardized metadata to bolster system interoperability and scalability across the CECC landscape.

3.4 Management plane components

The management plane of AC³ CECC architecture is the core part that implements all the logic related with the deployment and runtime management of the requested applications over the targeted infrastructures and according to the application requirements and policies as well as the available resources.

A more detailed view of the management plane is depicted in Figure 5 [6], showing the logical flow of information as well as the main processes that are supported. The management plane structure is also mapped to the overall CECC framework architecture. The management plane can be split into two layers:

a. The **Application and Resource Management layer** contains the orchestration functions related to the deployment and runtime management of the micro-services that are associated to end user application requests received from the user plane (Section 3.3).

b. The **Adaptation and Federation layer** provides the mechanisms for interacting with the underlaying infrastructures and their resources and includes the discovery and update of the of the available resources as well as the management of the resources following the orchestration decisions.

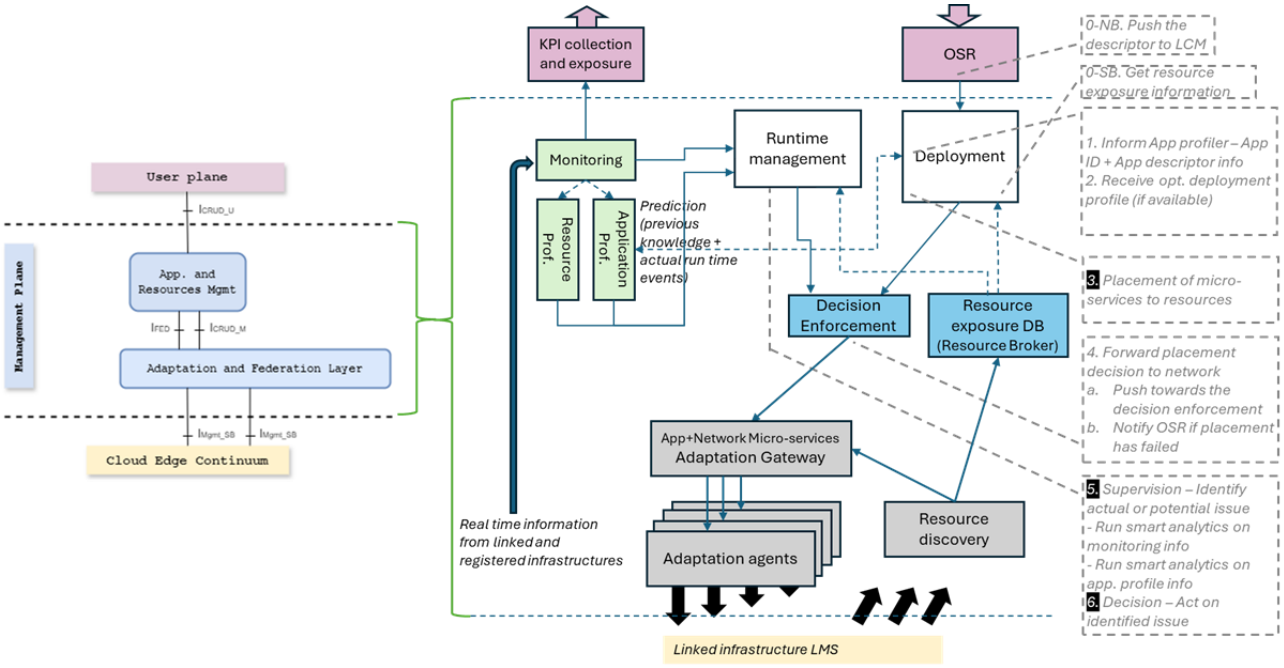

Figure 5. Detailed view of the management plane functionalities and processes and mapping with the overall CECC framework architecture.

In its Northbound Interface, the management plane is fed by the OSR's application composition file, which provides the application request in a structured model, encapsulating all micro-services and the way they are linked into a service chain, as well as all the necessary configurations, resource requirements, dependencies, and security access parameters per micro-service. In its Southbound Interface, the management plane interacts with the registered infrastructures providing the deployment and reconfiguration commands for the allocated resource environments. Moreover, it updates a resource exposure database with the exposed resources from each infrastructure domain, required for the decision mechanisms.

In the following subsections we proceed in further detail with each of the management plane components highlighting the key functionalities within the CECC framework and the interfacing with the other modules.

3.4.1 Application and Resource Management (through LCM)

The application and resource management functionalities are implemented through the LCM module. Its initial purpose is to collect the end user requests and convert them into deployment requests towards the infrastructure management entities. Its purpose during runtime is to manage the reconfiguration of the running applications through the generation of updated requests following the application profile, the targeted infrastructure resource profile and the application SLAs. By separating the two mechanisms, the LCM can be divided into two key modules as presented in the following paragraphs.

3.4.1.1 LCM Deployment mechanism

Upon receiving and parsing the application composition file, the deployment mechanism validates the request for completeness and correctness, ensuring that all necessary components are included and properly specified.

Next, the optimum placement of micro-services across the available infrastructure resources is calculated. At its input, the module is fed with the list of micro-services and the requested parameters. The optimization algorithm is triggered with the goal to calculate a) the deployment location for each micro-service taking into account factors such as latency, throughput, resource availability, and fault tolerance, ensuring that the deployed application performs optimally while adhering to the specified SLAs, and b) the order at which the micro-services are deployed, initialised and interconnected (i.e., according to their dependencies). It is noted that any type of optimisation algorithm can be used as long as it is aligned with input and targeted outcomes defined. Such algorithms may include bin packing algorithms, which enhance resource utilization by minimizing the number of nodes required for deployment, or genetic algorithms to explore a broad solution space, optimizing multiple objectives such as latency, throughput, and resource consumption. Also, heuristic-based approaches can be used to provide quick, near-optimal placement decisions, crucial for maintaining performance in dynamic environments.

3.4.1.2 LCM Runtime management mechanism

The runtime management mechanism of the LCM is primarily responsible for analysing the application and resource profile predictions, exploiting real-time monitoring information, and making decisions on the actions that should apply to already deployed micro-services, such as autoscaling, load balancing, and migration. By continuously monitoring the performance and status of both applications and infrastructure, the runtime management mechanism ensures that the system adapts to changing conditions and maintains optimal operation.

At the heart of this mechanism is the decision-making process. It begins with the analysis of real-time data provided by the monitoring module and predictive insights based on historical and current performance metrics. The runtime management mechanism evaluates these inputs to determine the most appropriate actions to maintain optimal performance and resource utilization. The respective capabilities of monitoring, SLA violation, predictive analysis, and decision-making are articulated in further detail in deliverable D4.1 [8] of WP4.

Once a decision is made, it is forwarded to the adaptation gateway module. The adaptation gateway acts as an intermediary, translating the high-level management decisions into actionable instructions for the LMS. It identifies the specific adaptation agent associated with the relevant LMS, as configured with the NBI endpoint during the setup phase.

3.4.2 Resource discovery/exposure

The resource discovery module is essential for collecting and providing infrastructure-related data from the linked infrastructures. The module connects to the LMS and gathers information on federated infrastructure resources, such as compute, storage, and network capacities, and standardizes this data into a common model. This formatted information is then exposed to the Resource Exposure module (Resource Broker) and the Adaptation Gateway module. The Resource Exposure uses this data to feed the optimized resource allocation and scheduling processes, while the Adaptation Gateway adjusts resource configurations dynamically in response to changing conditions. By continuously monitoring and updating resource information, the module ensures that the orchestrator framework can make informed decisions, maintain optimal performance, and efficiently utilize resources across diverse environments. Details on the resource exposure and collection is studied in WP4 and the contribution of AC^3 will be detailed in D4.2.

3.4.3 Monitoring

The monitoring component serves as the centralized layer where all metrics related to application performance and infrastructure resources are collected. It acts as the primary source for any AC^3 service requiring insights into the health and performance of an application. This data is provided in a standardized format based on a unified data model, eliminating the need for adapters, wrappers, or schema converters to translate or align information.

Specifically, the monitoring service is informed by the runtime management layer of the infrastructure about events such as changes or the instantiation of microservices, which are part of the overall application. This data is collected through dedicated processes and is augmented with additional insights from the infrastructure provider. Together, this forms a comprehensive view of the application's status, resource utilization, and allocation. These insights can then be utilized by AI-driven profiling mechanisms to analyze trends, detect anomalies, and support decision-making throughout the application's lifecycle.

Moreover, this monitoring capability extends its value to the Application Developers who receive detailed health and performance data about their deployed applications, enabling them to extract/check KPIs. These KPIs provide actionable insights into how an application is performing in real-world scenarios. By analyzing this data, developers can identify potential mistakes, optimize their workflows, and adopt best practices, leveraging feedback from real deployments to improve future versions of the application. This creates a continuous feedback loop, enhancing both application quality and developer efficiency.

It is worth mentioning that the monitoring framework embraced by the AC^3 project is detailed in D3.1.

3.4.4 Application Profile

The Application Profile in the AC^3 system enables efficient LCM of microservice-based applications across the CECC. It is built and linked to certain monitoring parameters for providing AI-based predictions during runtime. Upon registration, the profile is used by the application profile management module to facilitate application-specific reconfigurations based on monitoring information. In practice, the profile dynamically constructs and updates by integrating static application descriptors (provided by developers) with real-time monitoring data. These profiles encapsulate critical application elements, including resource requirements, performance metrics, traffic patterns, and SLA constraints.

By utilizing AI/ML models, the Application Profile predicts future application behaviors, such as resource utilization trends, traffic spikes, and SLA compliance, thus enabling proactive decision-making. This approach is studied and devised in T3.2, with initial findings in D3.1 and further details forthcoming in D3.2 on how AC^3 refines these models. The profiles are seamlessly integrated with the AI-based LCM module, serving as the bridge for exposing them to consuming services. This integration provides actionable insights to support resource scaling, optimal microservice placement, and load balancing, ensuring efficient resource utilization and SLA compliance. Adaptive by design, the Application Profile continuously evolves by leveraging real-time monitoring data, making it responsive to the dynamic nature of cloud-edge applications.

3.4.5 Resource Profile

In AC^3, a resource profile is a detailed specification and description of the characteristics of a particular resource used by an application developer. The resource profile provides information about a resource, such as its type, capacity, performance, to facilitate the allocation, management, and optimization of the resources that will be used. The type of resources can be a compute instance, storage and/or networking. The capacity specifies the metrics used which can be, for example, the CPU, the memory or the network throughput. The resource profile allows to meet the workload demands by making sure that the resources are appropriately sized and allocated. It also facilitates auto-provisioning and autoscaling resources. In AC^3, the resource profiling in the context of the CECCM can help define the CPU and memory requirements enabling efficient resource scheduling and preventing SLA violations. The resource profile will be built using the collected data on LMS using the resource and exposure module detailed in D4.2. As for the application profile, AI/ML algorithms will be used to build resource profiles and predict their availability for future deployment of micro-services. Clearly, the resource profile follows a similar approach as the application profile, but for CEC resources.

3.4.6 Adaptation gateway and agents

The adaptation gateway module has a crucial role in linking the LCM decisions to the LMS of various infrastructures. It associates an adaptation agent with each LMS, facilitating the enforcement of decisions made by the deployment and runtime management mechanisms of the LCM. Each adaptation agent is configured with the NBI endpoint of the LMS, as discovered by the resource discovery module. To achieve this, the adaptation gateway is designed to house a set of LMS-specific agents, each tailored to a particular LMS technology. These agents act as translators, converting the common API used by the management systems to the LMS-specific NBI. For instance, there could be dedicated agents for technologies such as Kubernetes, OpenShift, K3S, and SD-WAN controllers, among others. Upon receiving a request from the decision enforcement module indicating a specific LMS, the adaptation gateway identifies and provides the corresponding adaptation agent's address for execution. Furthermore, the adaptation gateway exposes the NBI endpoints discovered via the resource discovery module to the monitoring module. This allows the monitoring module to access the necessary endpoints, ensuring effective infrastructure oversight and management. In essence, the adaptation gateway module ensures seamless integration and communication between the management layer and the diverse local infrastructure management systems.

3.5 Infrastructure plane components

As previously mentioned, and detailed in D2.1, AC^3 builds upon the concept of CEC federation to constitute the CECC infrastructure. This design ensures that the CECCM does not directly manage the lifecycle of CEC resources. Instead, resources are selected from the federation by interfacing with the LMS of the respective infrastructure providers. AC^3 leverages NIST and IEEE SIF standards to establish agreements and select resource providers, enabling the CECCM to deploy applications on the CECC.

The CECCM interacts with the CEC federation across three levels:

- Business Level: This level is focused on selecting infrastructure providers. Detailed in D2.6, AC^3 defines a trust architecture that leverages blockchain technology and smart contracts to build infrastructure provider reputations. This mechanism assists the CECCM's federation layer broker in selecting appropriate infrastructure providers for application deployment.
- Control Plane Level: At this level, the CECCM interacts with the Local Management System (LMS) of an infrastructure provider. These interactions are necessary to deploy the microservices comprising the application or initiate a monitoring process via the local LMS APIs.
- Data Plane Level: Here, the CECCM collects monitoring data from the infrastructure provider's LMS through the local LMS APIs.

The control and data plane interactions occur sequentially after the business-level interaction. Initially, the CECCM Broker selects infrastructure providers. When an application needs to be deployed, the CECCM initiates interaction with the LMS.

To address the heterogeneity of LMS technologies (as noted in D2.3), the CECCM employs the adaptation agent component of the federation and adaptation layer (see Figure 2) to interact with the LMS. For each LMS technology, a corresponding adaptation agent is used, providing a unified northbound interface (NBI) to the Application and Resource Management Module for lifecycle management decisions. The adaptation agent translates the NBI into LMS-specific APIs. This approach enables AC^3 to abstract the use of federation resources from the application management components, ensuring seamless interaction and resource management.

To recall, the LMS can be defined as a localized orchestration framework that supports the deployment, scaling, and operation of infrastructure resources such as computing, storage, and networking. The next sub-sections describe the main type of LMS considered in AC^3.

3.5.1 Compute LMS

While there are numerous instantiations that meet the definition of a Compute LMS, one of the most comprehensive, and hence dominant across both industry and research, is Kubernetes. As such, we will discuss the capabilities of Kubernetes as it pertains to the function of an LMS. Kubernetes serves as a comprehensive LMS by orchestrating resources and managing application workloads across a cluster of physical machines. It ensures efficient use of CPU, memory, and storage by scheduling applications based on resource availability and defined constraints. Kubernetes abstracts the complexity of infrastructure, providing a unified platform for deploying, scaling, and maintaining containerized applications in a local environment. Kubernetes also provides a level of self-management behavior that ensures smooth workload operation through application restarts and autoscaling monitoring, as well as supporting vertical resource scaling. This makes Kubernetes a powerful tool for deploying and maintaining resilient, scalable applications within a localized infrastructure.

Kubernetes also offers a powerful API, allowing external systems to orchestrate application and infrastructure management, as well as extensibility mechanisms for integrating custom resources and functionality tools for logging, monitoring, and networking tailored to local needs. Kubernetes distributions, such as OpenShift, further enhance these capabilities by adding enterprise-grade features like integrated CI/CD pipelines, role-based access control, and a web-based management console, providing a more complete LMS experience. For environments requiring reduced resource footprints or far-edge deployment, lightweight Kubernetes distributions like K3S play a vital role. These distributions maintain core Kubernetes APIs while minimizing overhead, making them suitable for the edge. Their ability to scale down while preserving compatibility with centralized Kubernetes management tools is critical and ensures resource efficiency without sacrificing orchestration capabilities.

Beyond single clusters, Open Cluster Management (OCM) extends Kubernetes and OpenShift with multi-cluster federation capabilities. OCM enables workload placement, unified logging, monitoring, and policy enforcement across multiple clusters, ensuring seamless coordination of resources at scale, making it ideal for hybrid or multi-cloud environments.

While Kubernetes and its extensions provide significant capabilities for managing local and federated resources, the mechanisms for doing so are relatively simplistic and not well suited to the challenges of highly federated, heterogeneous and transient infrastructures. The ability of the AC^3 framework to leverage an LMS like Kubernetes, in order to provide intelligent and optimized management of workloads, data and resources is significant.

3.5.2 Network LMS

Similar to the Compute LMS, the role of the Network LMS is to provide an interface to AC^3 management plane that allows the configuration and management of network resources. This enables AC^3 to carry out advanced network management, facilitating connectivity across a federated infrastructure, as well as aiming to deliver robust, performant and secure communications.

The AC^3 network operator transforms Kubernetes into a specialized Network LMS, automating and streamlining networking operations within and across clusters. By leveraging Kubernetes' extensibility, the operator provides a high-level framework for managing complex networking tasks, reducing manual effort and ensuring efficient operation of networked applications. At its core, the AC^3 network operator dynamically manages networking resources, such as Secrets, Config Maps, and inter-cluster links. It automates the creation and maintenance of these resources, ensuring secure communication between namespaces and clusters. For example, it synchronizes Secrets like authentication tokens and copies them between clusters to enable seamless integration without manual intervention.

Additionally, the operator adapts to changing traffic patterns by dynamically creating and scaling links as needed. This ensures that the network remains responsive to demand while maintaining efficient use of

resources. Its design supports multi-tenancy by handling isolated namespaces and avoiding conflicts in shared environments, further enhancing its role as a local network management solution.

SD-WAN offers another Network LMS option to manage and optimize wide-area network connections. SD-WAN enhances connectivity across distributed sites by providing flexible, policy-driven traffic routing and link optimization. It enables consistent performance through dynamic failover and intelligent path selection, addressing key challenges like latency and bandwidth constraints in federated environments.

While SD-WAN can operate independently as a Network LMS, it can also be integrated with the AC^3 network operator to further streamline cross-cluster and inter-site communication. Together, these technologies create a versatile framework for managing local and federated network resources to ensure scalable connectivity across AC^3's distributed infrastructure.

3.6 Data Management

The AC^3 framework integrates a robust data management system that is essential for orchestrating and optimizing the deployment of applications across the cloud-edge continuum. This comprehensive integration supports a wide array of functionalities, crucial for managing the complexities of modern data-driven environments. Data management within the AC^3 framework facilitates efficient, secure, and interoperable interactions among diverse data sources and services. This is accomplished through an advanced framework that incorporates data handling, processing, and exposure functionalities seamlessly with the application deployment and management processes. By ensuring data is accessible and actionable throughout the service lifecycle, the framework adapts dynamically to application-specific requirements and infrastructure statuses. Figure 6 showcases how AC^3 Data Management is integrated into the project's architecture, with the core components to be presented in the rest of this section. Retrieval and processing of the data selected by application developers takes place inside the deployed application using what we define as application add-ons with Data Connectors performing the required handshakes to initiate this exchange. These components are deployed dynamically as part of the AC^3 application with their operation terminated when the application is destroyed.

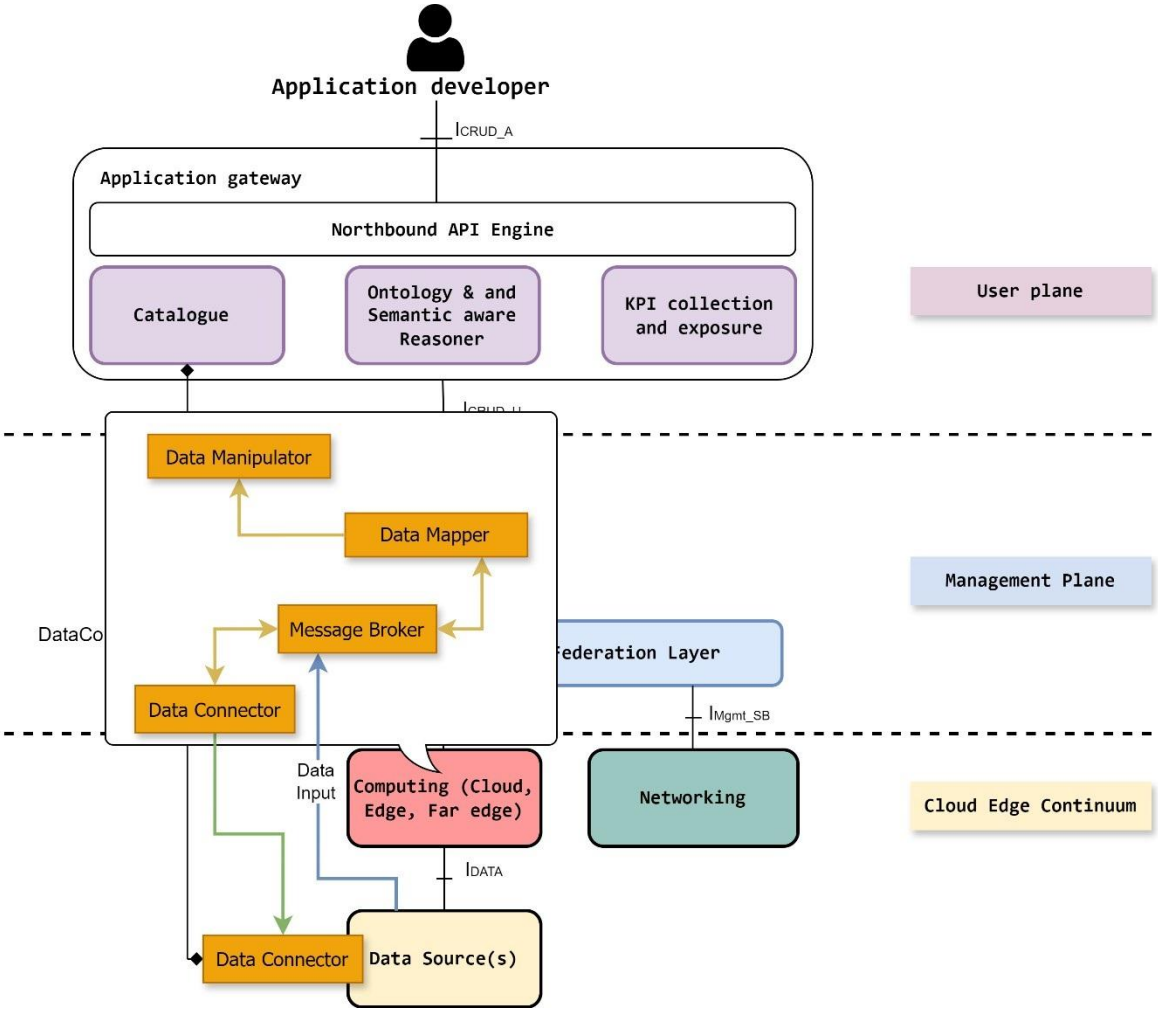

Figure 6. Data management as part of the High-level architecture of AC³.

3.6.1 Data Catalogue

The Data Catalogue within the AC³ framework represents a pivotal component in the domain of data management, encapsulating an exhaustive repository of metadata pertinent to data assets throughout the entire ecosystem. It is an integral part of the AC³ Catalogue. Architecturally sophisticated, this catalogue is engineered to precisely capture, manage, and mobilize data to accommodate the multifaceted demands of varied applications and services operating within the framework. Far surpassing the functionality of a mere static repository, the catalogue dynamically engages with and contributes to the orchestration and deployment of services across the AC³ architecture.

At the core of its functionality, the Data Catalogue meticulously maintains a full spectrum of information concerning each data element, encompassing specifications such as format, type, storage location, and the requisite protocols for access. This comprehensive metadata management fosters enhanced levels of data discoverability and operability within the system, streamlining the integration and execution of services. By facilitating the automated alignment of data requirements with existing resources, the catalogue significantly mitigates the complexities traditionally associated with managing extensive data repositories in distributed environments.

In addition to these capabilities, the Data Catalogue exhibits profound integration with service descriptors, enabling the agile and dynamic composition of services leveraging the data catalogued within. This flexibility is instrumental in responding to evolving conditions and requirements, empowering the AC³ system to

autonomously recalibrate and optimize service alignments with the most congruous data resources. Such a mechanism not only maximizes resource efficiency but also elevates the overall system responsiveness and agility.

Critical to its utility is the robust security framework embedded within the Data Catalogue, a response to the sensitive nature of data processed within the AC^3 ecosystem. The catalogue implements stringent access controls and sophisticated authentication protocols to rigorously regulate data access, ensuring that only duly authorized actors interact with specific data sets. These security measures are paramount not only for safeguarding data confidentiality and integrity but also for ensuring adherence to regulatory mandates pertaining to data protection and security standards.

3.6.2 Data Connectors

Data Connectors within the AC^3 framework serve as critical interfaces that seamlessly bridge diverse data sources, ensuring streamlined and secure data flow between these sources and the central system. Architecturally, these connectors form a unified data access layer that cohesively integrates various data endpoints, including cloud services, edge repositories, and far-edge devices, under a comprehensive data access schema operating within the cloud-edge continuum. The architecture of data connectors is beyond the scope of this document and articulated in deliverable D3.3 [7].

Meticulously designed, these connectors are engineered to accommodate a wide range of data interactions, integrating effectively with distributed storage infrastructures to ensure secure, efficient, and scalable data access throughout the continuum. They are equipped to handle a variety of data types, formats, and communication protocols, thereby enabling flexible and efficient data transfers. The architectural design incorporates high availability and fault tolerance, utilizing advanced protocols for establishing robust and resilient connections capable of withstanding potential failures.

Each Data Connector is specifically tailored to support multiple protocols, thereby addressing the unique requirements inherent in diverse data environments. For example, connectors tasked with real-time data streams from IoT devices are likely to utilize protocols optimized for rapid and reliable data transmission, while connectors responsible for handling extensive volumes of stored data might deploy protocols that facilitate efficient management of bulk data transfers. This versatility is crucial, as it empowers the AC^3 framework to be adaptive and responsive to the diverse demands imposed by the applications and services it supports.

Integration with the Adaptation and Federation Layer further enhances the functionality of Data Connectors, ensuring compliance with data access policies and federation agreements. This integration facilitates secure and compliant data migration and sharing across the continuum, thereby supporting dynamic and distributed workload management. By minimizing data transfer delays and maximizing throughput, these connectors significantly enhance overall system performance, which is particularly vital for applications that depend on real-time data processing capabilities.

Advanced data handling capabilities are a core feature of the Data Connectors, incorporating strategies to effectively manage both hot (real-time) and cold (archived) data, thus optimizing storage efficiency. They provide a robust suite of tools for data transformation and enrichment prior to integration, thereby enhancing data quality for downstream processes.

From an architectural perspective, Data Connectors are instrumental in upholding data governance and privacy by ensuring compliance with regulatory requirements and enforcing metadata-driven policy adherence. They ensure that data handling practices are consistent with regulatory frameworks such as GDPR, leveraging metadata frameworks to enforce data usage policies that align with organizational and regulatory directives.

The framework is inherently scalable and extensible, supporting modular extensions that allow for the integration of additional storage and communication protocols, thereby facilitating adaptation to technological advancements. Engineered for efficient management of substantial data transfers, the Data

Connectors align with the rigorous performance demands of the AC³ framework, ensuring seamless operation and high-performance data management capabilities.

3.6.3 Application Add-ons

Data Management Application Add-ons within the AC³ framework provide essential functionalities that significantly enhance the system's capability to handle, process, and manipulate data across various applications. These add-ons are specifically designed to ensure that data is not only accessible but also in a format and structure that can be effectively utilized by applications for various operational needs. Below is an in-depth exploration of each add-on, their roles, and how they integrate with the AC³ framework. Each of these Data Management Application Add-ons is deeply integrated with the AC³ framework, playing a specific role in data lifecycle management. From initial data ingestion and standardization by Data Mappers to detailed processing by Data Manipulators and efficient distribution by the Message Broker, these add-ons work in concert to provide a robust, scalable, and secure data management infrastructure. This integration ensures that the AC³ framework can support complex, data-driven applications across distributed computing environments, facilitating advanced data operations and real-time decision-making processes.

3.6.3.1 Data Mappers

Data Mappers play a crucial role in ensuring that data from various sources can be seamlessly integrated and utilized across the AC³ framework. The primary function of Data Mappers is to transform incoming data into a standardized format, thus facilitating interoperability and reducing complexities associated with managing heterogeneous data structures. This transformation process involves converting data from disparate formats and structures into a common model that aligns with the system's requirements.

In the AC³ architecture, Data Mappers are integrated at points where data is ingested into the system, ensuring that all incoming data is immediately processed into a usable form. This is essential for enabling different components of the system to communicate and function cohesively, regardless of the original data source or format. By standardizing data at the entry point, Data Mappers help maintain consistency and reliability throughout data processing workflows, enhancing the overall efficiency and effectiveness of the system.

3.6.3.2 Data Manipulators

Data Manipulators refine raw data to enhance its utility for further use within the AC³ framework. These components perform a variety of operations that improve data quality and prepare it for in-depth analysis and decision-making processes. Key operations include:

- **Data Cleaning**: This process involves removing inaccuracies, redundancies, and inconsistencies from data. Data cleaning is vital for ensuring the accuracy and reliability of data analyses performed within the system.
- **Data Transformation**: Data Manipulators convert data into formats that are suitable for analysis. This may involve normalization, scaling, or other transformations that facilitate easier and more effective data processing.
- **Data Aggregation**: Combining data from multiple sources or streams into a single dataset is crucial for comprehensive analysis. Aggregation helps in providing a consolidated view of data, which is essential for applications that require a holistic understanding of information from various inputs.

Data Manipulators are integrated within the data processing pipelines of the AC³ framework, interacting closely with Data Mappers and other system components. By refining data post-mapping, these add-ons ensure that data is not only uniform but also primed for the specific analytical needs of applications, thereby enhancing the value and applicability of information within the ecosystem.

3.6.3.3 Message Broker

The Message Broker serves as the central communication hub within the AC³ framework, managing and optimizing the flow of data among various applications and components. Its primary role is to facilitate

efficient and secure data exchange, ensuring that data requests and deliveries are conducted promptly and accurately. The Message Broker is strategically placed within the network infrastructure to intercept and route data as needed. It uses sophisticated routing algorithms to determine the most efficient paths for data delivery, taking into account factors such as network latency, data priority, and system load. By dynamically managing data flows, the Message Broker enhances the responsiveness and reliability of the AC^3 system, supporting real-time data processing and decision-making capabilities.

3.7 Interfaces summary and explanation

We summarize all the inter-plane and intra-plane interfaces of the AC^3 architecture in Table 3 and Table 4, respectively.

Table 3: Inter-plane interfaces and their functionalities.

Interface Name	Involved modules	Sub-Interface(s)	Description
I_{CRUD_A}	Application Developer, Application Gateway	-	Interface allowing the application developer to request CRUD procedures (i.e., deploy, instantiate, monitor, and delete micro-services composing an application)
I_{CRUD_U}	Application Gateway, Application & Resources Mgmt	I_{R_U}	Monitoring Management and measured KPI presentation
		I_{CUD_U}	Deployment, instantiation, and deletion of micro-services
I_{OSR}	Application Gateway and OSR	-	Provides the OSR with the semantic models and policy rules, and the OSR produces ontology instances and validated/verified semantic rules
$I_{DataMgmt}$	Application Gateway, Data Mgmt	-	Provides the Application Gateway with access to the available data sources as well as with access to their management operations
$I_{DataAccess}$	Data Federation Services, Data Source Connector	-	Provides access and permissions management for the available data sources.
$I_{DataTransfer}$	Data Mappers, Data Source Connector	-	Provides access and data transfer operations to the actual data stored in the data spaces connected to the AC^3 platform.
I_{FED}	Application & Resources Mgmt, Adaptation & Federation layer	-	Select an infrastructure provider from the Federation
I_{CRUD_M}	Application & Resources Mgmt,	$I_{D\_LCM\_SB}$	Further divided into two sub-interfaces: LD_CFG and LD_SB. The first one is to discover the adaptation agent handling an

	Adaptation & Federation layer		infrastructure. The second one is to enforce LCM decisions using the agent discovered by LD_CFG
		$I_{Mon\_M\_SB}$	Further divided into two sub-interfaces: Mon_CFG and Mon_SB. The first one is needed to discover the NBI of a LMS to collect monitoring. The second one is to collect monitoring data from the LMS discovered using Mon_CFG.
$I_{DataMgmt\_SB}$	Application & Resources Mgmt, Data infrastructure	$I_{DataAccess}$, $I_{DataTransfer}$	Provides interaction with federated data sources and data transfer operations.
I_{Mgmt_SB}	Adaptation & Federation layer, CECC infrastructure	I_{RES_DSC}	Discover resources and NBI exposed by LMS participating to the federation
		I_{Mgmt_SB}	Enforce LCM decisions using NBI exposed by LMS

Table 4: Intra-plane interfaces and their functionalities.

Interface Name	Concerned module	Description
$I_{Mon\_CFG\_M}$	Application & Resources Mgmt.	Configure the monitoring module (KPI to measure) per application
I_{Mon_Data}	Application & Resources Mgmt.	Expose measured KPI by application and used-infrastructure.
$I_{AppProfile}$	Application & Resources Mgmt.	Expose learned Application profile
$I_{ResProfile}$	Application & Resources Mgmt.	Expose resources profile including current resource availability
I_{A_CFG}	Adaptation & Federation layer	Configuration (instantiation) of an agent per infrastructure LMS.
I_{A_Disc}	Adaptation & Federation layer	Discover NBI exposed by LMS for monitoring (R) and CUD.
I_{FED_Inf}	Adaptation & Federation layer	Discover LMS exposed resources.

4 Workflows

4.1 Application deployment without data

We first start with the Application deployment workflow (see Figure 7) which is considered a user-driven workflow.

The application creation process begins at the Application Gateway, specifically through the Northbound API, where the Application developer submits a request using an application descriptor. The Northbound API forwards this request to the service catalog to check if a suitable blueprint exists. If available, it forwards the request to the orchestration service (ORS) to generate a ready-to-use application descriptor.

The Application Gateway then passes the application creation request to the App and Resources Management module, where the AI-based LCM processes it. The LCM retrieves the application profile from the AI-based Application Profile service and the resources profile from the AI-based CEC Resource Profile service. Using this information, the AI-based LCM determines the initial placement and resource allocation for the application within the target infrastructure.

Once the decision is made, the AI-Based LCM initiates the application deployment process, which consists of three key steps:

1. **Onboarding**: Each microservice of the application is onboarded by loading its respective container image.
2. **Instantiation**: The application is instantiated by starting it with the allocated resources in the selected infrastructure.
3. **Network Configuration**: Traffic rules are created to configure the application's network, enabling it to handle Read and Update (RU) operations.

For each step, the AI-Based LCM sends a request to the Decision Enforcement components, which then forward it to the Adaptation Agent within the Adaptation and Federation Layer, which in turn, communicates the request to the infrastructure-specific Local Management System (not presented in the workflow).

After the application is deployed, the AI-Based LCM sends a monitoring request to the Monitoring component to begin tracking the performance of the application's microservices and network.

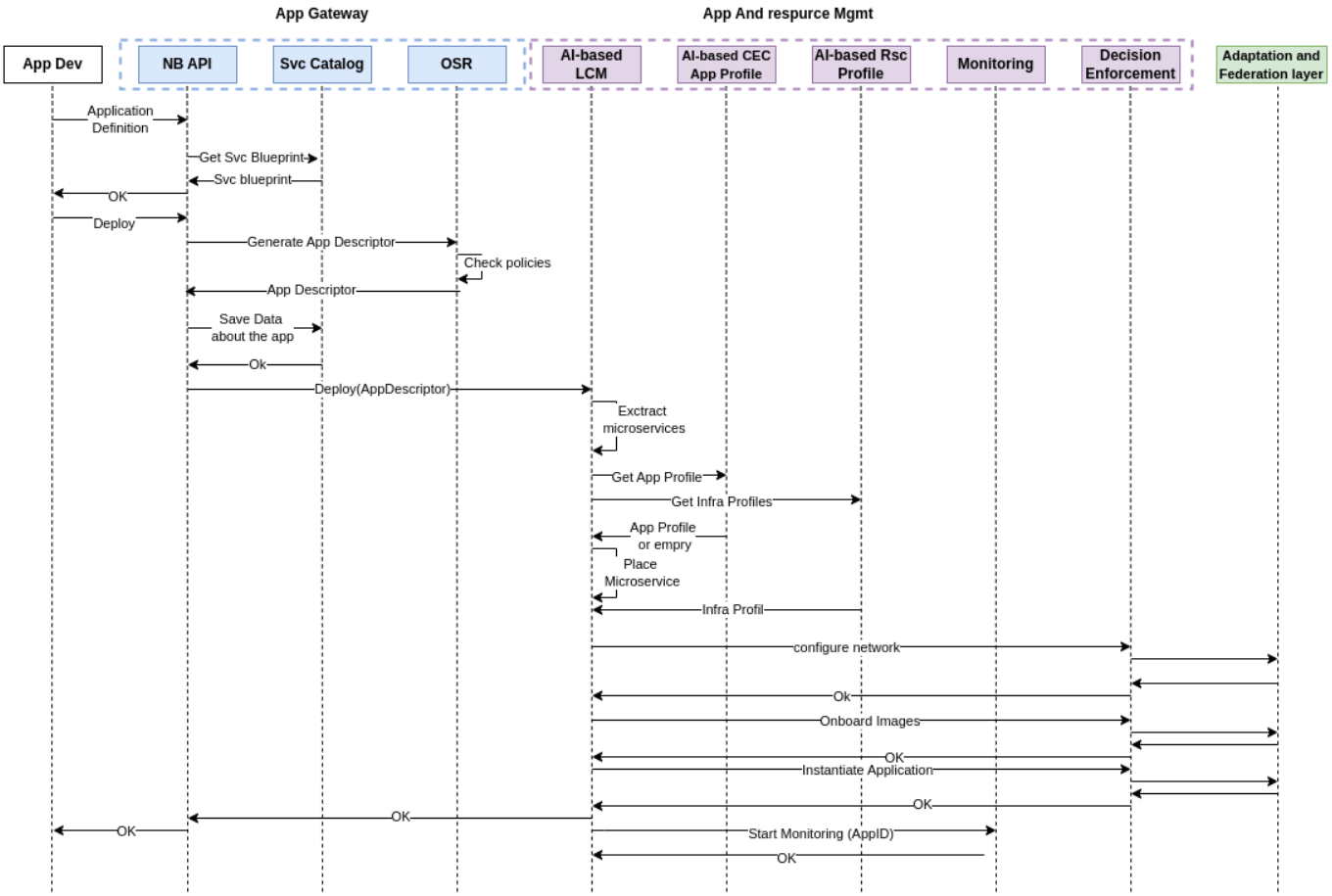

Figure 7. Application deployment without data.

4.2 Application deployment with hot data

The hot data application deployment workflow in Figure 8 provides a depiction of the dynamic data retrieval process within AC^3.

The application creation process begins at the Application Gateway, specifically through the Northbound API, where the application developer submits a request using an application descriptor. The Northbound API forwards this request to the data catalogue to check for the appropriate data sources that fit the user's request. Based on these data sources selected, a set of data management addons are added to the needed service blueprints. Next, Northbound API forwards this request to the service catalogue to check if a suitable blueprint exists. After these two requests, it forwards the request to the ORS to generate a ready-to-use application descriptor that also contains the data source descriptions and the required data management application add-ons.

The Application Gateway, then passes the application creation request to the Application and Resources Management module, where the AI-based LCM processes it. The LCM retrieves the application profile from the AI-based Application Profile service and the resources profile from the AI-based CEC Resource Profile service. Using this information, the AI-based LCM determines the initial placement and resource allocation for the application within the target infrastructure.

Once the decision is made, the AI-Based LCM initiates the application deployment process, which consists of three key steps:

1. **Onboarding**: Each microservice of the application is onboarded by loading its respective container image.
2. **Instantiation**: The application is instantiated by starting it with the allocated resources in the selected infrastructure.
3. **Network Configuration**: Traffic rules are created to configure the application's network, enabling it to handle RU operations.

For each step, the AI-Based LCM sends a request to the Decision Enforcement components, which then forward it to the Adaptation Agent within the Adaptation and Federation Layer, which in turn, communicates the request to the infrastructure-specific LMS to deploy the services of the application. In this workflow we depict only the instantiation of the two data management application addons, the Consumer Connector and the Message Broker.

The Consumer Connector then starts the process of requesting the hot data from the data source via the Data Source Connector. This process has the following steps:

1. **Asset Request:** The Consumer Connector requests the asset description from the Data Source Connector. This description contains also the policy information for using the data source.
2. **Contract Negotiation:** If the policy is accepted (i.e., data usage restrictions are acceptable) the Consumer Connector negotiates a contract with the Data Source Connector.
3. **Data Subscription:** Once the Data Source Connector accepts the contract request, the Consumer Connector subscribes for data updates providing the endpoint of the Message Broker where the Data Source Connector should forward updates (i.e., new readings from the IoT deployment).
4. **Data Forwarding:** The Data Source Connector streams the new data to the Message Broker using the data provided.

After the application is deployed, the AI-Based LCM sends a monitoring request to the Monitoring component to begin tracking the performance of the application's microservices and network.

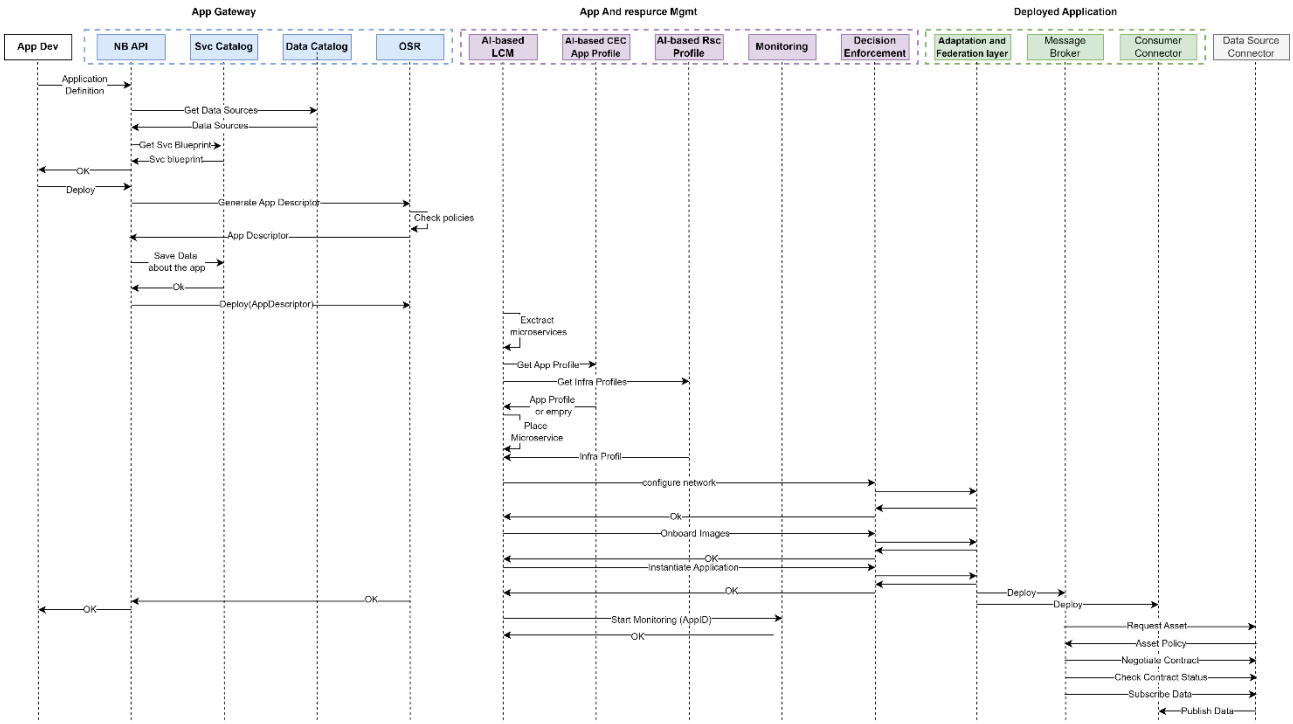

Figure 8. Application Deployment with hot data.

4.3 Application deployment with cold data

Correspondingly, this section focuses on the cold data application deployment workflow which is illustrated in Figure 9. It provides a depiction of the stored data retrieval process within AC^3.

The application creation process begins at the Application Gateway, specifically through the Northbound API, where the application developer submits a request using an application descriptor. The Northbound API forwards this request to the data catalogue to check for the appropriate data sources that fit the user's request. Based on these data sources selected, a set of data management add-ons are added to the needed service blueprints. Next, Northbound API forwards this request to the service catalog to check if a suitable blueprint exists. After these two requests, it forwards the request to the OSR to generate a ready-to-use application descriptor that also contains the data source descriptions and the required data management application add-ons.

The Application Gateway then passes the application creation request to the App and Resources Management module, where the AI-based LCM processes it. The LCM retrieves the application profile from the AI-based Application Profile service and the resources profile from the AI-based CEC Resource Profile service. Using this information, the AI-based LCM determines the initial placement and resource allocation for the application within the target infrastructure.

Once the decision is made, the AI-Based LCM initiates the application deployment process, which consists of three key steps:

1. **Onboarding**: Each microservice of the application is onboarded by loading its respective container image.
2. **Instantiation**: The application is instantiated by starting it with the allocated resources in the selected infrastructure.
3. **Network Configuration**: Traffic rules are created to configure the application's network, enabling it to handle RU operations.

For each step, the AI-Based LCM sends a request to the Decision Enforcement components, which then forward it to the Adaptation Agent within the Adaptation and Federation Layer, which in turn, communicates the request to the infrastructure-specific LMS to deploy the services of the application. In this workflow, we depict only the instantiation of the two data management application add-ons, the Consumer Connector and the Message Broker.

The Consumer Connector then starts the process of requesting the cold data from the data source via the Data Source Connector. This process has the following steps:

1. **Asset Request:** The Consumer Connector requests the asset description from the Data Source Connector. This description contains also the policy information for using the data source.
2. **Contract Negotiation:** If the policy is accepted (i.e., data usage restrictions are acceptable), the Consumer Connector negotiates a contract with the Data Source Connector.
3. **Data Subscription:** Once the Data Source Connector accepts the contract request, the Consumer Connector initiates the transfer of data providing the location where the data should be transferred.
4. **File Transfer:** The Data Source Connector transfers the data to the location provided.

After the application is deployed, the AI-Based LCM sends a monitoring request to the Monitoring component to begin tracking the performance of the application's microservices and network.

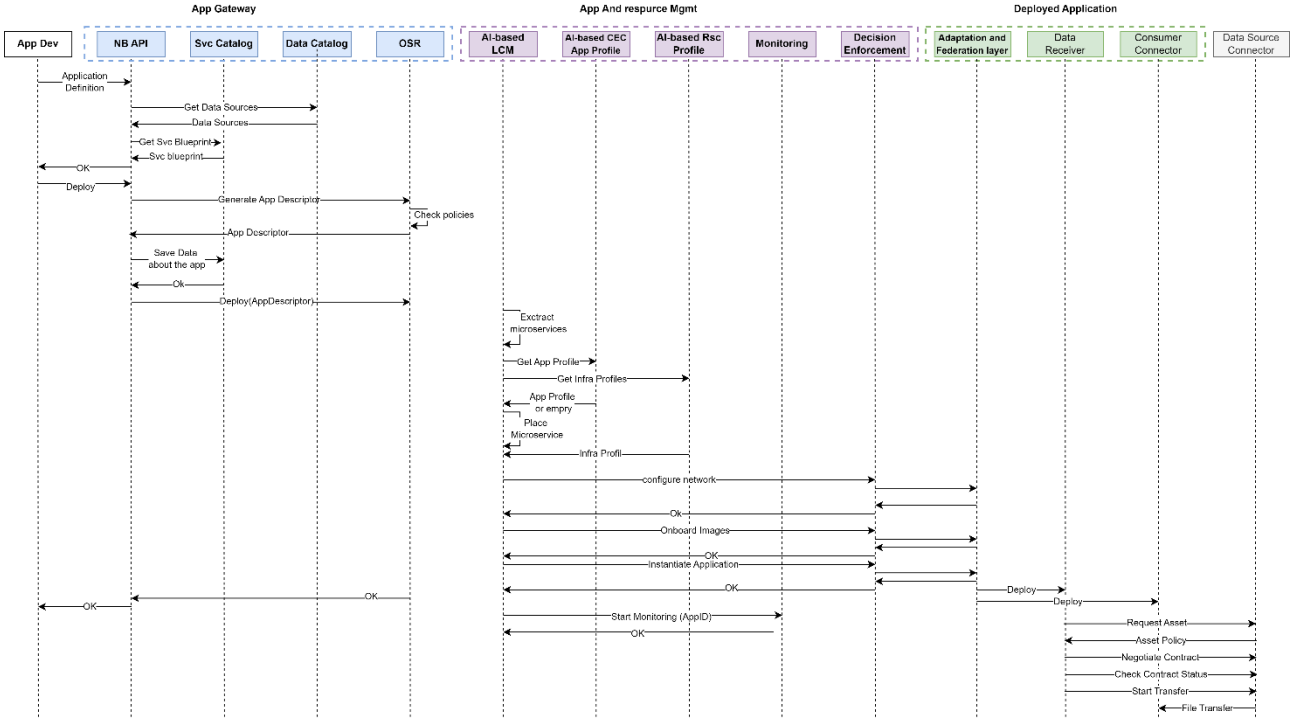

Figure 9. Application Deployment with cold data.

4.4 Run-time management: micro-service update and migration

The runtime management workflow is depicted in Figure 10. This process relies on continuous monitoring and validation during the application run-time phase, and therefore, it assumes that the deployment and monitoring establishment processes have been successfully completed.

The monitoring mechanisms feed the AI-based Application and Resource profile modules with the declared (during application deployment) application-specific metrics. Both modules run validation algorithms in parallel and provide prediction metrics that are related to the application SLA parameters, as well as the application micro-services and the infrastructure where the micro-services are deployed. When a potential update is identified, (e.g., as a predicted violation due to an increased rate of a certain metric or a metric exceeding a threshold), then an update recommendation is created and sent to the LCM module. The LCM is then responsible for validating the recommendation and making the optimum decision which may include a simple upscale of resources or migration of a micro-service. The decision is next communicated to the targeted infrastructure management layer through the decision enforcement element that provides the communication interface. The first action requires the configuration and update of the cluster. If this configuration requires migration, then the micro-service image is onboarded to the target LMS and following this, the traffic is redirected, before the old image is removed and resources are released. Finally, the monitoring is also updated accordingly.

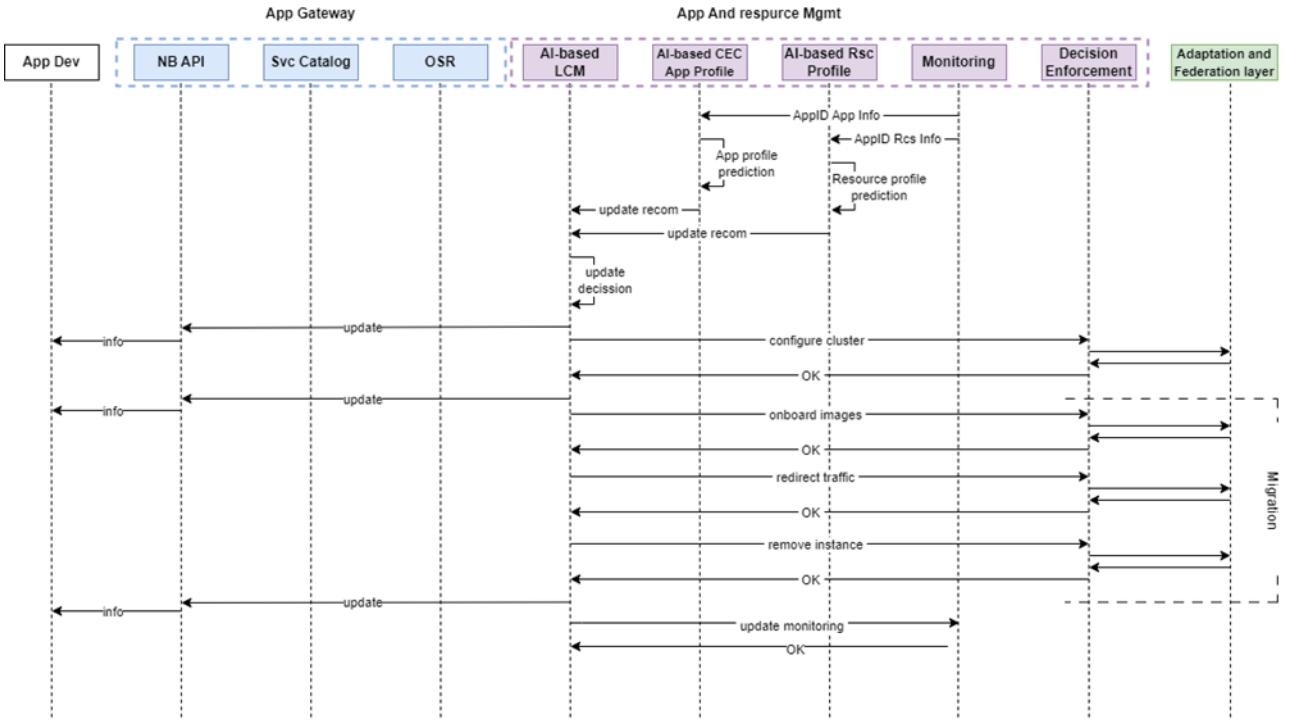

Figure 10. Run-time management workflow including the migration process.

4.5 Application deletion

The application deletion workflow is presented in Figure 11. Upon reception of an application deletion request by the end-user the NB API of the App gateway associates the request with the appropriate AppID (obtained at the final step of the deployment workflow once the deployment is confirmed) and posts a deletion request to the LCM. The request proceeds towards two paths. One internal to the resource manager that updates the profile modules and the monitoring to terminate their processes and one external towards the decision enforcement interface. The latter terminates the whole application deployment (extended potentially to the workspace level) which in turn removes the deployed micro-service pods. A confirmation message propagates back to the LCM and then to the end user (App Dev).

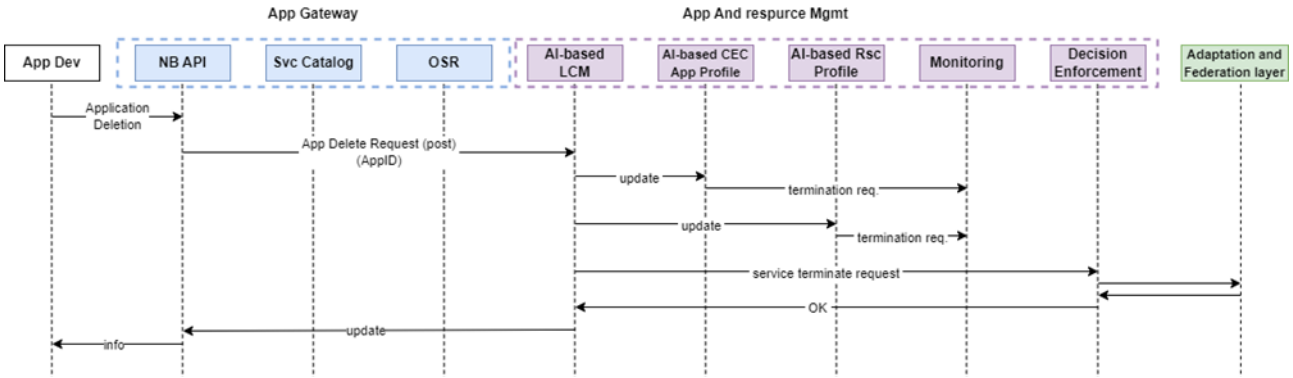

Figure 11. Application deletion workflow.

Here, we assume that the end-user is the one that initiates the deletion request. This is the most generic case. However, a deletion may be triggered as a result of a corrupted application that must be terminated or in general due to policy enforcement reasons. What changes, in this case, is the triggering point in the workflow which may be a process within the LCM.

5 Conclusions

In this deliverable, we presented the final version of the AC^3 architecture, which adopts a modular approach to support the features and requirements of managing the lifecycle of microservice-based applications over a federated CEC infrastructure. This modular design consists of independent modules that collaboratively form the CECCM, a central output of the AC^3 system. Each module is defined by its specific tasks and is grouped into one of three distinct planes: the user plane, management plane, and infrastructure plane. These modules contain independent components that interact both within their respective planes and across planes through well-defined interfaces.

The finalized architecture consolidates the outcomes of T2.2 and builds upon the initial version introduced in D2.1. Significant updates were made to the user plane components, incorporating feedback from WP3 and WP4 activities. This final version enables seamless integration of various algorithms and components developed in WP3 and WP4, specifically: (1) AI/ML Algorithms that ensure cognitive management of the lifecycle of microservice-based applications within the federated CECC architecture; (2) Data Management that includes mechanisms for managing hot and cold data efficiently.

Key components of the architecture will be demonstrated in WP5 through AC^3 use cases, showcasing the system's capabilities and validating its design.

6 References

1. https://gaia-x.eu/
2. https://gaia-x.eu/services-deliverables/deliverables/
3. D2.1: 1st release of the CECC framework and CECCM, 2024. [Online]. Available: https://ac3-project.eu/wp-content/uploads/2024/02/AC3_D2.1.pdf
4. 2302-2021 - IEEE Standard for Intercloud Interoperability and Federation (SIIF). [Online]. Available: 10.1109/IEEESTD.2022.9732072
5. D2.5: Security and trust management for CECC, 2024. [Online]. Available: https://ac3-project.eu/wp-content/uploads/2024/07/AC3_D2.5_v1.6_version_to_be_submitted.pdf
6. D3.1: Report on the application LCM in the CEC – Initial, 2024. [Online]. Available: https://ac3-project.eu/wp-content/uploads/2024/11/AC3_D3.1_v7.pdf
7. D3.3: Report on data management for applications in CECC – Initial, 2024. [Online]. Available: https://ac3-project.eu/wp-content/uploads/2024/11/AC3_D3.3_20240222_v1.0.pdf
8. D4.1: Initial report on mechanisms that enable green-oriented zero touch management of CECC resources–Initial, 2024. [Online]. Available:

   https://ac3-project.eu/wp-content/uploads/2024/11/AC3_D4.1_to_be_submitted_annexes.pdf

9. https://doc.piveau.eu/guides/create-custom-metadata-model/