CLOUD·EDGE CONTINUUM

IPCEI-CIS Reference Architecture

Version 3.0

July 2026

Abstract

This document presents the third version of the IPCEI-CIS Reference Architecture, to be used as an architectural framework for the IPCEI-CIS Integrated Project, guidance for describing, implementing, and running integration pilots within the Project, as well as orientation for adopters and implementors.

TABLE OF CONTENTS

Contributors

IPCEI-CIS Partner	Co-author
SAP	Andreas Schlosser – *Coordinator, Editor*
	Florian Müller - *Editor*
TIM	Mauro Boldi
	Ignazio Di Natali
	Roberto Querio
	Roberto Procopio
Telefónica	Cristina Santana Casillas
	Antonio Medina Martín
	Alexandre Harmand
	Álvaro Fernández Garrido
REPLY	Giovanni Gregorelli
	Gabriele Lospoto - *Coordinator*
	Davide Scarano
	Paolo Fasano
WestfalenWIND IT	Marco Plaß
E-Group	Antal Kuthy
	Ákos Tényi
	Marcell Zoltay
OpenNebula Systems	Jordi Guijarro Olivares
	Alberto Picón
	Alfonso Carrillo Aspiazu
	Constantino Vázquez
NBIP	Hans Punt
Villanova.ai	Alessio Manca
	Patrizia Loi
BIT (ECOFED)	Cristiaan Brans - Jansen
	Stefan Kooman
Info Support (ECOFED)	Erwin Kersten
	Vincent van der Winkel
TNO (ECOFED/MISD)	Magiel Bruntink
	Edwin Harmsma
	Bart Kamphorst
	Erik Langius

	Bram van der Waaij
Lindner SE (BiGreen)	Ahmed Chebaane
Infobip	Tomislav Đuričić
	Ante Kapetanović
	Andro Merćep
	Emanuel Lacić
T-Systems	Werner Jost
Ericsson	Oliver Speks

0. Executive Summary

This document presents an updated version of the Reference Architecture for the IPCEI-CIS integrated project, describing the functional components that it will deliver. In the short term, it is guiding the definition, development, and delivery of first pilot and integration scenarios. These pilots help validate and further improve this *IPCEI-CIS Reference Architecture* (ICRA), adding aspects that may be missing in the initial versions. Looking forward, this document gives orientation to adopters and implementors of the ICRA, who are using the delivered components.

The model focuses on the integration and orchestration of infrastructure and workload lifecycle management across a hybrid edge-cloud environment, and includes other aspects like physical infrastructure management, connectivity management, and edge-cloud federation.

Section 1 defines the scope of this Reference Architecture. Section 2, “High Level Overview of the IPCEI-CIS Reference Architecture,” describes the layers and domains. Section 3, “Component Description,” provides detailed examination of each layer and domain, describing their functional components in depth. Section 4 “Federation,” offers a high-level overview of federation across multiple providers. Section 5, “Functional Interfaces,” covers the interfaces between layers and domains. Section 6, “Reference Architecture Roles,” explains the main personas considered in the ICRA. Section 7 summarizes the document and outlines possible next steps for future versions and adoption of the ICRA.

The Reference Architecture outlined in this document is designed to be robust and scalable, providing a flexible framework that covers the needs of the different verticals and can evolve to meet future demands. It is adapted, usually by streamlining it, to more efficiently fit specific sector needs in the shape of *blueprints*, or instantiations of this ICRA.

1. Scope

The **IPCEI-CIS Reference Architecture (ICRA)** is defined in the context of the *Important Project of Common European Interest on Next Generation Cloud Infrastructure and Services (IPCEI-CIS)*. The purpose of IPCEI-CIS is to perform research, development and innovation (R&D&I) and first industrial deployment (FID) of the software components necessary to establish and operate a **distributed, openly accessible and interoperable Multi-Provider Cloud-Edge Continuum**.

The IPCEI-CIS targets the creation of a cloud-edge environment that enables seamless data processing and service management across cloud and edge infrastructures operated by **multiple independent providers**. In contrast to current approaches based on proprietary, single-provider technology stacks, the IPCEI-CIS pursues a model in which infrastructure resources, platform capabilities, services, data, and applications can be **interoperable, portable, and composable** across provider boundaries, while respecting European requirements on data protection, security, and sustainability.[1]

A key objective of the IPCEI-CIS is the definition of a **common reference architecture** that provides a blueprint for how interoperable cloud and edge systems can be set up and operated. This reference architecture is intended to define a **shared understanding of layers, functional components, interfaces, and roles** within the Cloud-Edge Continuum, enabling federation, automation, and scalability in a multi-provider environment. The reference architecture is **technology-agnostic** and does not prescribe specific implementations; instead, it establishes a conceptual framework that can be instantiated by different providers and adapted to diverse technical and sector-specific contexts.

In addition to structuring the cloud-edge technology stack, the IPCEI-CIS requires that the reference architecture consistently addresses **cross-cutting concerns** applicable to all layers of the system. These include federation across providers, management of infrastructure and services, security and compliance, and sustainability. These aspects are considered integral architectural domains and are treated as first-class concerns alongside functional layers such as application, data, AI, service orchestration, platform, virtualization, and physical infrastructure.

Within this context, the **ICRA** serves as the **common architectural framework** for the IPCEI-CIS integrated project. It is used to describe and position the functional components delivered by the different workstreams, to guide the definition and execution of **pilot scenarios and first industrial deployments**, and to support the derivation of **blueprints** that tailor the reference architecture to specific sectors or use cases. The ICRA provides a consistent basis for integration, comparison, and evolution of IPCEI-CIS contributions, ensuring interoperability, portability, and compatibility across the European Multi-Provider Cloud-Edge Continuum.

This document is the foundation of the 8ra initiative[2], the strategic European endeavor dedicated to establishing a resilient, open and future-proof digital infrastructure. 8ra will be the umbrella of upcoming programs such as IPCEI AI and IPCEI CIC.

The objective of this document is to create an **IPCEI-CIS Reference Architecture** that:

[1] Decision of the European Commission on the IPCEI Next Generation Cloud Infrastructure and Services: https://competition-policy.ec.europa.eu/state-aid/ipcei/approved-ipceis/cloud_en

[2] https://www.8ra.com

1. integrates the main components with capabilities contributed by IPCEI-CIS partners into a unified structure, providing a high-level framework that effectively positions these components within the overall project, facilitating collaboration and maximizing the impact of each partner's contributions;
2. aligns with and endorses compliance with relevant EU regulations such as GDPR, Data Act and AI Act;
3. guides the positioning of IPCEI-CIS partners within a unified framework, driving the development and delivery of first Pilot Scenarios;
4. serves as baseline for IPCEI-AI and IPCEI-CIC, preventing further architectural documents which might lead to a proliferation of specifications, and including potential architectural extensions directly in this document or annexes, with the goal of having one single and consistent reference architecture covering multiple topics.

The Pilot Scenarios are proofs of concept to demonstrate the practical application of this Reference Architecture, showcasing the integrated management of infrastructure and workloads in a hybrid edge-cloud environment. They will help to validate the Reference Architecture and further improve it by identifying missing elements or aspects or optimizing others that are already considered.

1.1 Evolution of the document scope

Version 1 defined the basic layers, domains, and definitions of the overall ICRA architecture. This sets the framework for discussions among IPCEI-CIS participants on the various contributions in the different workstreams and individual projects.

In version 2, the ICRA was extended with a description of functional components based on current knowledge and status of the different workstreams. The content on federation was expanded based on evolution of the various projects.

With version 3 the document includes clearer references to the IPCEI-CIS mission and Chapeau text, a major overhaul of the federation and sustainability sections, and updates based on recent developments in the area of AI.

2. High Level Overview of the IPCEI-CIS Reference Architecture

2.1 ICRA Layers and Domains

The ICRA aims at clarifying and organizing responsibilities between the different IPCEI-CIS partners. For this purpose, it organizes the functional components in layers, that provide different levels of abstraction and help to manage the complexity of a Multi-Provider Cloud-Edge Continuum, and domains that represent cross-cutting aspects applicable to all layers (e.g. management or security).

Figure 2.1 presents a high-level view of the different layers and domains that compose the ICRA. The architecture includes the following **core layers**:

1. **Application Layer**: It provides the functionality required to design and develop applications and functions and to expose them for use. It includes end-user applications and function catalogs.

2. **Data Layer**: It contains functionalities for data collection, processing and exchange, making them simple, scalable, sustainable, trustable, and distributed over the Cloud-Edge Continuum. These data artifacts can be applied both in end-user applications, by third parties, and internally to optimize the edge-cloud system at different levels.

3. **Artificial Intelligence (AI) Layer**: Its aim is to create a fundamental set of advanced functionalities to a hybrid approach sets of data and processing resources distributed or centralized depending on the use cases. As with data artifacts, they are also applicable to end-user applications or to internal edge-cloud optimization.

4. **Service Orchestration Layer**: It addresses the integration and management of multiple cloud and edge services and applications in a unified and automated way across diverse multi-provider environments. In the context of multi-cloud and edge ecosystems, service orchestration is essential for handling the complexity of deploying and managing applications at scale, especially when dealing with distributed resources across cloud and edge infrastructures.

5. **Cloud-Edge Platform Layer**: It allows the allocation and lifecycle management of resources over the virtualized Cloud-Edge Continuum and the deployment and wiring of application components to deliver a service to end users in a certain geographical area. It includes components that facilitate and manage the complexity of computing environments that represent multiple cloud technologies and are offered by different providers. The components of this layer are designed to enable interoperability, compatibility, and portability across a Multi-Provider Cloud-Edge Continuum.

6. **Virtualization Layer**: It provides the necessary abstraction over the physical hardware resources (compute, storage, networking) to facilitate their dynamic allocation, usage, and sharing, hiding the heterogeneity of the hardware infrastructure. It provides a virtual runtime environment in which the applications are deployed and executed. This layer is

linked to the SDN (Software-Defined Network) or other management systems of public and private networks.

7. **Network Systems, SDN Controllers**: They provide the capabilities to manage physical and virtualized/cloudified networking elements to build network services in the geographically distributed Cloud-Edge Continuum. This layer provides connectivity towards user equipment, for example through telecommunication provider 5G or 6G networks.

8. **Physical Layer**: It includes all the physical hardware resources required to implement the Cloud-Edge Continuum (compute, storage, and networking). It is closely connected to the physical network infrastructure that supports communication among the computing nodes in the continuum and the connectivity of users to that continuum.

In addition to these layers, the architecture includes several **cross-cutting domains** that apply to all of them:

1. **Federation Domain**: It contains the components that provide functionality and mechanism to interconnect different providers in the Cloud-Edge Continuum, in such a way that the consumer may get access to and use resources and services from all the federated providers. Federations formalize collaboration among multiple providers. Federations may happen at virtualized infrastructure, platform, service, data, AI, and application layers, or over multiple layers simultaneously.

   In each layer, specific mechanisms and interfaces are defined to provide the means for collaboration. Federation will be a fundamental pillar for the European Multi-Provider Cloud-Edge Continuum, which will provide for interconnection and seamless operation of the cloud and edge components, even if they are operated by independent providers.

   The Multi-Provider Cloud-Edge Continuum is not static. Provider compositions for a service consumer may change during its life cycle. Consumers and providers may switch between data processing services (including cloud and edge services), in line with applicable regulatory requirements (for example, the EU Data Act's rules on switching and portability). Such events require support for porting workloads, data and digital assets, and reduction of technical, contractual and economic obstacles. The Federation domain facilitates these kinds of processes and the assigned responsibilities.

2. **Management Domain**: Effective management of the cloud stack necessitates a comprehensive suite of capabilities to ensure the required quality and performance at the different layers.

   Logging and monitoring are foundational, providing detailed records and continuous observation of system activities, which support troubleshooting, security analysis, and performance optimization. Coupled with an inventory of resources and services, as well as alerting mechanisms and fault management, these capabilities enable real-time detection

and swift resolution of anomalies and potential failures, to operate the different layers and ensure high availability and reliability.

Metering, accounting and charging systems are essential for tracking resource consumption and generating accurate invoices, fostering transparency and accountability. Meanwhile, the management of *Service Level Agreements* (SLAs) ensures that performance and availability commitments are consistently met, enhancing customer satisfaction and trust.

Additionally, fault management processes detect, diagnose, and resolve issues promptly, contributing to the resilience and robustness of the cloud environment. Automation for integration, delivery, verification, testing, and other processes in cloud-edge environments is crucial for efficient and reliable operations. Together, these management capabilities form an integrated framework that supports the scalability and efficiency of cloud services, driving both operational excellence and strategic value.

3. **Security and Compliance Domain**: It plays a crucial role in safeguarding data integrity, ensuring that access is strictly regulated, and maintaining adherence to industry standards and regulations. This domain is fundamental in designing a robust cloud stack that not only protects sensitive information but also enforces policies that prevent unauthorized access and misuse.

   By integrating advanced security protocols, real-time monitoring systems, and compliance frameworks, this domain mitigates risks, responds swiftly to potential threats, and ensures that the cloud environment remains secure and compliant with legal and regulatory requirements. This proactive approach to security and compliance is essential for fostering trust and reliability in cloud services.

4. **Sustainability Domain:** The sustainability domain focuses on improving the energy efficiency and flexibility of the cloud-edge continuum by embedding sustainability considerations into the operational behavior of all components. Enhancing energy efficiency and emission reduction in cloud and edge infrastructures is crucial and could be achieved at physical resource layer through energy-efficient cooling solutions, hardware optimization, and the integration of data processing with renewable energy sources. Smart placement of edge nodes ensures access to renewable energy and efficient cooling. As a side effect to efficient energy usage smart workload placement may reduce net congestion during times of an excess supply of green energy and times where demand of grey energy is high.

   Integrating sustainability into components does not fundamentally change their functionality but enhances them with awareness and optimization capabilities. Each component is expected to provide both transparency (through monitoring and metrics) and opportunities for improvement by reducing energy consumption and increasing efficiency. These improvements are driven by two main factors: economic incentives, where investments must yield a positive business case, and societal pressures, often translated into regulations that ensure a level playing field across providers.

Sustainability is further reinforced through both provider- and consumer-driven mechanisms. Providers can optimize their infrastructure, procure greener resources, and meet regulatory targets, while also enabling consumers to make more sustainable choices through transparency and service-level agreements. Consumers, in turn, can use this information to steer their consumption, compare providers, and support sustainability reporting.

These cross-cutting domains may use the capabilities provided by layers to optimize the different aspects they deal with.

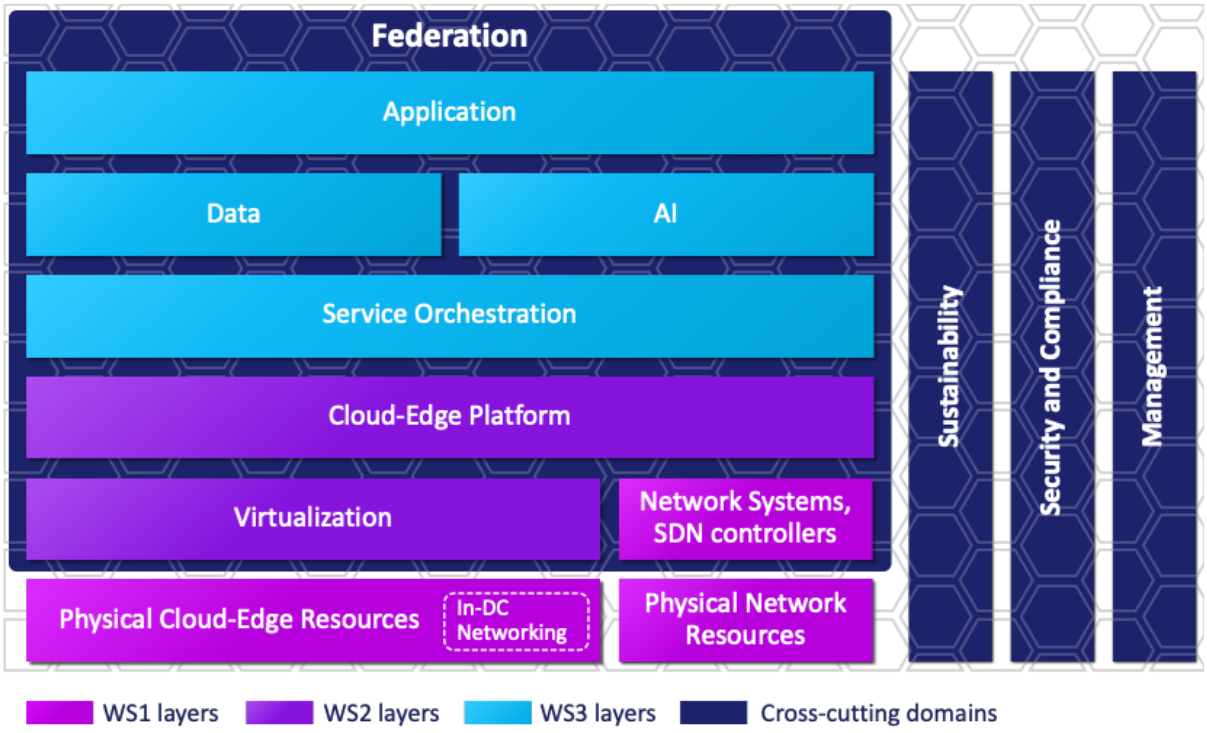

Figure 2.1: *Layers and Domains in the ICRA.*

The layers and domains of the ICRA manage different kinds of resources (physical, virtual, platform functions, and services and applications) and provide different kinds of cloud computing services (*Infrastructure as a Service* – IaaS, *Container as a Service* – CaaS, *Platform as a Service* – PaaS, *Software as a Service* – SaaS), as depicted in Figure 2.2. The combination of components providing different services is fundamental in the Cloud-Edge Continuum solutions that aim at being multi-provider, open, and disaggregated.

It must be noted that the Reference Architecture is modular and that the different components and layers can be merged or segregated based on development approach and commercial need. As an example, not all components may be required for the commercial launch of an edge service, and several functional capabilities/layers may be consolidated in a single commercial product.

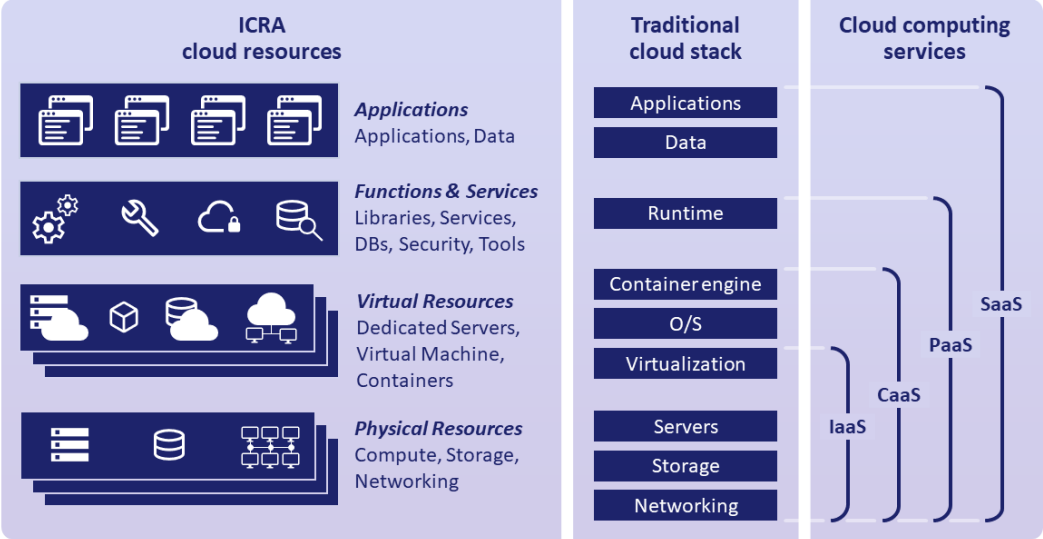

Figure 2.2: *ICRA Resources managed by every type of cloud computing service.*

2.2 ICRA Components

An ICRA Component is a conceptual and descriptive unit defined within the IPCEI-CIS context. It represents a functional capability within the ICRA framework and is not intended to be a software artifact. It enables consistent description, comparison, and mapping of capabilities across projects.

ICRA adopts a component-based perspective, where components are described as conceptual and functional building blocks. Components encapsulate both data and behavior, interacting through defined interfaces without exposing their internal workings. An ICRA Component is a conceptual and descriptive unit used to represent a specific functional capability. It is defined within the 2026 IPCEI-CIS context and abstracts from any concrete implementation, enabling consistent comparison, mapping, and analysis of capabilities across projects.

Applied to the European Multi-Provider Cloud-Edge Continuum, Component Based Design is not mere choice: it is the foundational design philosophy that makes the ICRA vision achievable. Without it, a multi-provider, multi-cloud, AI-composable ecosystem would require every integration to be custom-built, every migration to be hand-crafted, and every new capability to be re-invented per provider.

According to ICRA, a component is a modular functional building block within a layer or domain that provides a specific capability, exposes interfaces, and can be combined with other components to enable interoperable services across the Cloud-Edge Continuum.

The decomposition into components is intended to facilitate the description of the whole system functionality and the participation of specialized providers with innovative solutions in certain functional components of the cloud computing value chain.

For their interoperability, portability and compatibility within the Cloud-Edge Continuum, they will support the relevant interfaces to facilitate integration with other systems (via APIs, manifests, or other integration artifacts) and the consumption of services (via open standard service interfaces to customers).

An ICRA Component is a deployable unit of functionality that:

1. Aligns with ICRA layers (Application, Service Orchestration, Cloud-Edge Platform, etc.)
2. Exposes functional interfaces (preferably any of the 9 ICRA-defined interfaces)
3. Preferably supports federation (multi-provider, multi-cloud operation), where applicable
4. Preferably enables AI composition (machine-readable specs, agentic interfaces)
5. Is built based on sovereignty principles (e.g. GDPR, data/service/processing residency, compliance with EU regulations)

3. Component Descriptions

Each layer represents a certain level of abstraction and provides a set of functionalities that can be split into smaller components. The domains represent functional aspects that span across all layers, i.e., the domain components can be found in any of the architecture layers.

This section covers the different domains and layers, describing the components that implement them. In each layer description (subsections 3.5 to 3.11), some examples of domain components are given. The list of domain components in each layer will be more detailed and exhaustive in further releases of this architecture.

It must be noted that this architecture does not limit the possibility that the implementation of a certain layer groups certain functional components into bigger blocks or excludes some of them that may not be needed for a specific scenario. In some scenarios, even complete layers may be merged or skipped in the actual implementation.

The decomposition into components is intended to facilitate the description of the whole system functionality and the participation of specialized providers with innovative solutions in certain functional components of the cloud computing value chain.

It represents neither a reference implementation nor an implementation guideline, but more of a conceptual framework for understanding the integration and interaction of components within the Cloud-Edge Continuum. Industrial implementations may group components in a different way or exclude some of the components in case they are not required for the scenarios they address. For their interoperability, portability and compatibility within the Cloud-Edge Continuum, they will support the relevant interfaces to facilitate integration with other systems (via APIs, manifests, resource descriptors, or other integration artifacts) and the consumption of services (via open standard service interfaces to customers).

3.1 Management Domain

Effective management of cloud infrastructure necessitates a comprehensive suite of capabilities to ensure optimal performance, security, and compliance. Together, these management capabilities form an integrated framework that supports the scalability, efficiency, and security of cloud services, driving both operational excellence and strategic value.

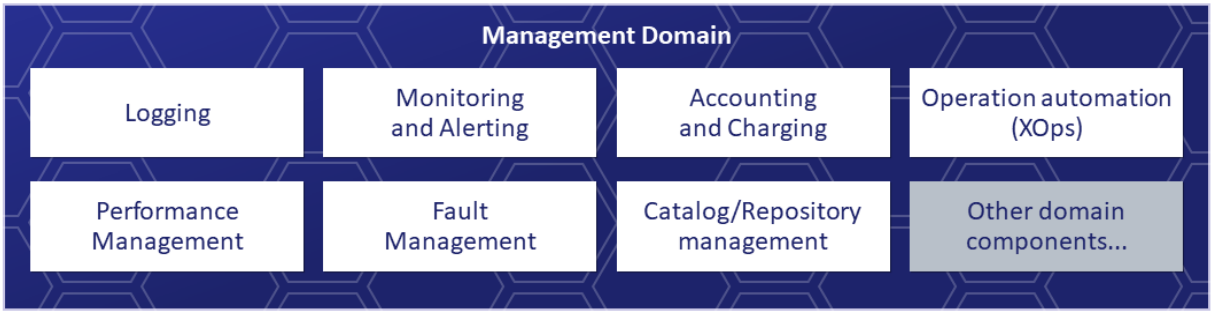

Figure 3.1: *Components in the Management domain.*

3.1.1 Logging

Logging in cloud infrastructures refers to the systematic recording of events, transactions, and activities within the cloud environment. This process captures detailed logs of user actions, system operations, and data interactions, creating a repository of information that supports monitoring, auditing, troubleshooting, and security analysis.

By maintaining comprehensive and accurate logs, cloud providers and users can trace system behaviors, detect anomalies, and swiftly respond to incidents. Effective logging not only aids in compliance with regulatory requirements but also enhances the overall transparency, reliability, and resilience of the cloud infrastructure.

3.1.2 Monitoring and Alerting

Monitoring and Alerting in cloud infrastructures involves continuous observation and real-time analysis of system performance, application behavior, and resource utilization. This process employs various tools and techniques to collect and analyze data from different components of the cloud environment, such as servers, networks, and applications (running on cloud, edge, or far edge devices).

By setting static or AI-supported thresholds and rules, monitoring systems can detect anomalies, performance bottlenecks, and potential failures. When these conditions are met, alerting mechanisms are triggered to notify administrators and stakeholders promptly, enabling swift resolution and minimizing downtime. Effective monitoring and alerting are essential for maintaining the reliability, availability, and overall health of cloud and edge services, ensuring optimal user experiences and adherence to SLAs.

Logging, monitoring, and alerting are foundational, providing detailed records and continuous observation of system activities, which support troubleshooting, security analysis, performance optimization, and pro-active intervention. Coupled with alerting mechanisms, they enable real-time detection and swift resolution of anomalies and potential failures, ensuring high availability and reliability.

Logging, monitoring, and alerting could use artifacts from the Data layer to implement its functionality.

3.1.3 Accounting and Charging

Accounting and Charging in cloud infrastructure and services refers to the systematic process of tracking and invoicing the usage of cloud services by users and applications. Accounting involves the continuous collection of data on resource consumption, such as CPU usage, memory allocation, storage, and network bandwidth. This data is then analyzed to generate detailed usage reports, which serve as the basis for customer billing. Charging systems apply predefined pricing models and rates to the metered data, ensuring accurate and transparent charges based on actual usage. This process not only provides customers with clear insights into their cloud expenditures but also enables providers to manage resources efficiently and optimize their service offerings.

Effective accounting and charging are essential for tracking resource consumption and generating accurate accounting, fostering transparency. This is critical for maintaining financial accountability, fostering trust, and supporting the scalable and on-demand nature of cloud services.

3.1.4 Performance Management

Performance Management in cloud stack refers to the comprehensive process of defining, monitoring, and enforcing SLAs between cloud service providers and their customers. This applies also to internal performance goals the providers may set for their services.

This involves setting clear expectations for service performance, availability, and support, and ensuring that these commitments are met consistently. SLA management includes tracking key performance indicators (KPIs), generating compliance reports, and addressing any deviations through corrective actions. Effective performance management not only enhances customer satisfaction and trust but also enables providers to maintain high standards of service quality and reliability, thereby fostering long-term business relationships and competitive advantage.

The management of SLAs ensures that performance and availability commitments are consistently met, enhancing customer satisfaction and trust.

3.1.5 Fault Management

Fault Management in cloud infrastructures involves systematic detection, isolation, and resolution of faults or issues within the cloud environment. This process includes identifying potential failures, diagnosing their root causes, and implementing corrective actions to restore normal operations. Fault management leverages workflow and ticketing systems combined with automated tools and techniques to monitor system components, analyze error logs, and trigger alerts for anomalies. By proactively managing faults, cloud providers can minimize downtime, enhance system reliability, and ensure continuous service delivery, ultimately contributing to the overall robustness and resilience of the cloud infrastructure.

Fault management processes detect, diagnose, and resolve issues promptly, contributing to the resilience and robustness of the cloud environment.

3.1.6 Catalog/Repository Management

It provides inventory information, in different layers, about resources used and available and services available and deployed to make decisions. For instance, list of *Kubernetes* (k8s) clusters available (with characteristics) and list of applications deployed (in which cluster). In addition, catalogs enable the implementation of a model-driven approach for services and resources.

3.1.7 Operation Automation

This component provides automation for integration, delivery, verification, testing, optimization, and other processes in cloud-edge environments. It allows us to manage distributed software and infrastructure in an automated way (by code, by software, scripting, declaratively) reducing human errors, making implementations more uniform and predictable and facilitating the reconciliation and

recovery to a working configuration after a misconfiguration, disaster, or failure. It is usually referred to as XOps (DevOps, MLOps, AIOps, FinOps...).

3.2 Security and Compliance Domain

The Security and Compliance Domain in a cloud environment plays a crucial role in safeguarding data integrity, ensuring that access is strictly regulated, and maintaining adherence to regulations.

In this document, the security and compliance domain is based on the *Collaborative Security Framework* (CSF)[3], developed by the IPCEI-CIS Security Workstream. The CSF aims to establish a unified approach to security across the IPCEI-CIS project by fostering collaboration among partners. This framework identifies and standardizes security components, promotes interoperability, and ensures baseline security compliance across all identified 8ra security topics.

This proactive approach to security and compliance is essential for fostering trust and reliability in cloud services.

Since the security and compliance domain requires an all-encompassing view, the CSF is based on 10 major topics listed below (**Figure 3.2:** Components in the Security and Compliance domain.). When applicable, subtopics are already listed.

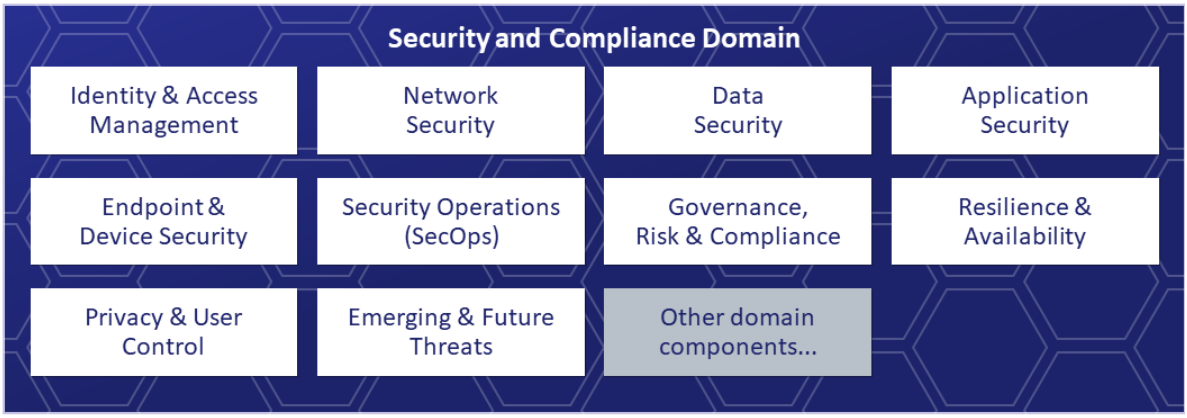

Figure 3.2: *Components in the Security and Compliance domain.*

3.2.1 Identity & Access Management (IAM)

- Decentralized Identity (DID)
- Role-Based Access Control (RBAC)
- Zero Trust Architecture (ZTA)
- Multi-Factor Authentication (MFA) and biometrics
- Identity federation (OAuth 2.0, OpenID Connect)

This domain focuses on controlling and verifying identities and their access to resources across the Cloud-Edge Continuum. It encompasses capabilities like decentralized identity, role-based access control, multi-factor authentication, and identity federation. The CSF delivers a unified approach for

[3] https://www.8ra.com/wp-content/uploads/IPCEI-CSF_2026.pdf

IAM by classifying all partner-contributed identity solutions and mapping them into the Reference Architecture. This ensures consistent application of Zero Trust principles for access, clear alignment of each IAM component to the overall security architecture, and a baseline set of identity services that all partners can leverage and trust.

3.2.2 Network Security

- Firewalls and Segmentation
- VPN and IPsec tunnels
- Secure DNS (DoH/DoT)
- Intrusion Detection/Prevention (IDS/IPS)
- Secure routing and path validation

Network Security covers the protection of data in transit and the defense of network infrastructure. This includes DDoS protection, firewalls, secure network segmentation, encrypted communications (e.g., VPN/IPsec tunnels), intrusion detection/prevention systems, and secure routing protocols. The CSF consolidates and classifies network security components provided by different partners, integrating them into a cohesive multi-provider network defense strategy. By mapping each solution from secure DNS to DDoS mitigation into the framework, the CSF ensures that the entire Cloud-Edge Continuum benefits from a coordinated network security posture, and it identifies any gaps where additional controls or partner contributions may be needed.

3.2.3 Data Security

- End-to-End Encryption (E2EE)
- At-rest and in-transit encryption
- Data loss prevention (DLP)
- Key management systems
- Post-quantum cryptography

Data Security is concerned with protecting data at rest, in transit, and in use. It spans end-to-end encryption, DLP measures, robust key management systems, and emerging techniques like post-quantum cryptography. The CSF delivers classifications of all data protection mechanisms across partners, establishing common standards for encryption and data handling throughout the platform. By mapping partner contributions (e.g., encryption services or key vaults) to this domain, the framework ensures consistent data confidentiality and integrity measures are applied project wide. It also helps in verifying that each partner's solutions meet compliance requirements for data privacy and that any critical gaps (such as missing encryption capabilities) are addressed collaboratively.

3.2.4 Application Security

- Secure coding practices
- Static and dynamic analysis
- Web Application Firewall (WAF)
- API security
- Software Bill of Materials (SBOM)

This domain addresses the security of software applications and services, including their development and runtime protection. It involves secure coding practices, code analysis (static and dynamic testing), application firewalls like WAF, API security, and the use of SBOM to track components. The CSF maps and classifies all application security tools and practices contributed by partners for example, cataloging code scanning tools or runtime protection modules to ensure that they align with the Reference Architecture's application layer. By doing so, the CSF facilitates a baseline for secure software development and deployment across the project. Partner contributions are integrated so that vulnerabilities are identified early, common security standards like OWASP Top 10 mitigations are followed, and all applications in the IPCEI-CIS ecosystem benefit from a robust and consistent security posture.

3.2.5 Endpoint & Device Security

- Secure OS and firmware
- Device identity and attestation
- Mobile device management (MDM)
- IoT security frameworks

Endpoint and Device Security focuses on protecting the multitude of devices and nodes from user devices to edge hardware and IoT sensors that connect to the cloud-edge environment. This includes secure operating systems and firmware, device identity and attestation mechanisms such as using TPM/HSM hardware roots of trust, MDM, and IoT security frameworks. The CSF classifies partner contributions in this area to build a comprehensive device security layer within the framework. It will deliver an integrated approach where solutions like secure boot processes, remote attestation services, and device management tools are mapped to the appropriate architecture layers. By aligning these contributions, the CSF ensures that all endpoints in the federation, regardless of provider, meet a common baseline of trustworthiness and that device-level threats are mitigated through coordinated controls and monitoring.

3.2.6 Security Operations (SecOps)

- Continuous monitoring and logging
- SIEM systems – threat detection and response
- Automated incident response
- Red team/blue team exercises

This domain covers the operational aspects of security, including continuous monitoring, threat detection, incident response, and security assurance activities. It involves SIEM systems for centralized logging, alerting, and analytics, automated threat response or SOAR playbooks, and regular exercises (red team/blue team) to test defenses. Through the CSF, partners' SecOps tools and services will be identified and integrated into a unified operational security framework. The CSF delivers a coordinated approach to monitoring and incident handling. Examples include ensuring that logs and alerts from all components feed into a common analysis platform and establishing joint incident response procedures. By mapping each partner's monitoring and response capabilities, the framework improves overall situational awareness and ensures that security events anywhere in the continuum can be detected and addressed quickly through collaborative efforts.

3.2.7 Governance, Risk & Compliance

- Security policy management
- NIST/ISO compliance
- Risk assessments
- Privacy regulations (e.g., GDPR, CCPA)
- Audit logging and forensic readiness

Governance, Risk and Compliance (GRC) encompasses the policies, processes, and oversight needed to manage security risks and ensure compliance with standards and regulations. This includes security policy management, risk assessment methodologies, audits and reporting, and adherence to frameworks like ISO 27001 or industry-specific standards. Within the CSF, all partner contributions related to governance and compliance, such as policy frameworks, audit tools, or compliance checkers, are cataloged and harmonized. The CSF delivers a baseline security policy framework developed in collaboration with all partners, aligning everyone with common compliance requirements, for example, GDPR for privacy or national cloud security guidelines. By classifying each component's compliance posture and mapping it to the relevant controls, the framework helps identify any regulatory gaps and ensures that risk management is a shared responsibility. This unified approach to GRC builds trust among partners and stakeholders by demonstrating that the entire platform adheres to high security standards and well-known best practices.

3.2.8 Resilience & Availability

- DDoS mitigation
- Redundancy and failover
- Disaster recovery planning
- Chaos engineering
- SLA for uptime

This domain focuses on keeping services reliable and available even under adverse conditions. Key aspects include DDoS mitigation strategies, redundancy and failover mechanisms, disaster recovery planning, and even techniques like chaos engineering to test system robustness. The CSF integrates and classifies all measures related to resilience contributed by partners, delivering a comprehensive continuity strategy for the project. This involves mapping out how each partner's components achieve high availability, for example, identifying which services have built-in failover or backup, and how they interconnect across providers. By coordinating these strategies, the CSF ensures that critical cloud-edge services maintain agreed SLAs for up-time and that recovery procedures are in place. The framework will highlight any weaknesses in continuity, such as single points of failure and facilitate joint improvements, thereby enhancing the overall reliability of the multi-provider environment.

3.2.9 Privacy & User Control

- Privacy-by-design principles
- Consent management
- Differential privacy
- Data minimization practices

Privacy and User Control is dedicated to protecting personal data and upholding user rights across the system. It includes enforcing privacy-by-design principles, managing user consent and data preferences, implementing data minimization, and techniques like differential privacy for data analysis. The CSF delivers a consolidated view of how each partner's solutions address privacy concerns, classifying these measures and ensuring they are embedded into the architecture from end to end. All partner components will be mapped against privacy requirements, for instance, whether they properly handle consent or anonymize user data, and the CSF identifies any gaps where additional controls are needed. By collaboratively developing a privacy questionnaire and a baseline aligned with regulations like GDPR, the framework guarantees that user data is handled transparently and that users maintain control over their information. This unified stance on privacy fosters user trust and compliance with legal mandates throughout the IPCEI-CIS platform.

3.2.10 Emerging & Future Threats

- Quantum-resistant encryption
- AI-driven threats
- Autonomous agents and smart contracts
- Satellite and mesh network security
- Biological and neuro-network interfaces
- Agent/tool interface security

This forward-looking domain addresses the need to anticipate and guard against evolving security challenges. It covers preparation for threats on the horizon such as quantum-computing attacks and the need for quantum-resistant encryption, AI-driven or autonomous attacks, vulnerabilities in novel technologies like smart contracts or satellite/mesh networks, and even potential bio/neuro-technology risks. In the Emerging and Future Threats domain, priority should be given to building federated threat intelligence capabilities that empower every IPCEI-CIS participant to proactively recognize and respond to existing and novel risks. Secure data sharing agreements and trust frameworks must enable partners to exchange timely threat information without compromising sovereignty. This approach will turn isolated insights into coordinated threat defense and establish a unified strategy to anticipate and counter new cyber threats across the European cloud-edge ecosystem. The CSF coordinates research and development efforts among partners to identify and classify solutions for these emerging threats, ensuring that the security architecture remains adaptive. Concretely, the CSF maps partner contributions, such as experimental quantum-resistant cryptography implementations or AI-based threat detection tools into the framework. By doing so, it delivers a strategy for continuous innovation in security: as new threats are identified, the framework can evolve by incorporating new controls or best practices. This collaborative approach means that the project is not just reacting to current threats but actively preparing for future challenges as a unified front.

3.3 Sustainability Domain

The sustainability components in the ICRA enable significant gains in energy efficiency and energy flexibility of the cloud-edge continuum.

Sustainability is a large topic. Within the ICRA scope, focus is on operational aspects of the ICRA: emission reduction and stretching energy savings. Other topics such as material usage, product life

cycle, water usage, social sustainability, etc. are left out of scope but might be included in further versions. In scope is 1) getting insights into and 2) control over the sustainability of each component in each layer in the ICRA, including heat reuse, green power and electricity net congestion mitigation.

Adding sustainability to a component can influence the component's design and operation. It does in principle not affect the function of the component, but it makes that component more sustainability aware by providing insight into its energy consumption and by making it more efficient to reduce its energy consumption.

There are two drivers to make sustainability-related investments for components: economic and societal. For the economic driver there must be a positive business case by itself: the benefits need to outweigh the costs of implementing the efficiency measures. The societal driver typically results in governmental (EU/national) regulations to create a level playing field amongst all providers of the same functionality. Regulations make organization change in the societal desired direction. They also create a level playing field for the necessary investments, allowing companies to create an acceptable economic environment.

Each component development in the whole ICRA should consider and, if possible, provide sustainability insights, and sustainability improvements for the component. Insight is important to facilitate awareness for its consumers. It can be done at the level of the whole component, but preferably on the level of each consumer of the component. For example, these insights can be at the whole component level of a hypervisor or at the consumer level for each VM hosted by the hypervisor. Improvement is important to lower overall energy consumption, CO2 emissions and efficiency enhancements, thereby increasing the sustainability of the component.

3.3.1 Steering on sustainability

The ICRA is defined via layers with components, and in this technical model it is intentionally unspecified who is offering (which provider) the component (or components) as a service to the market. From a provider perspective there are two drivers to make its components more sustainable: internally making the provider itself more sustainable, important for its own reporting. The second driver is to help its consumers make their IT usage more sustainable and provide them with insight and influence on that. In best case it uses resources more economically, so it is more cost efficient and effective. Therefore, it automatically reaches economic and sustainability at the same time.

Provider as a driver

Steering on sustainability can be an internal goal of a provider, reaching its own (possibly governmental enforced) sustainability goals (e.g. PUE and WUE) by:

a) gathering internal sustainability insight,
b) improving sustainability of all its components,
c) buying more sustainable resources (including green energy and components bought as-a-service from other providers), or
d) making its energy usage sustainable through buying certificates of origin.

Consumer as a driver

Components get related to a consumer when that component, or a combination of components, are offered as a service on the market. The provider can then choose to help its consumers in reaching their sustainability goals by:

a) providing insight into the sustainability impact (energy consumption and CO2 emissions) of the service usage of that consumer, or
b) providing specific sustainable SLAs under which this service can be offered in a more sustainable fashion.

With this information the consumer can steer on sustainability by:

a) understanding its sustainability situation due to the insight,
b) choosing its services under better sustainable SLA conditions,
c) switching to other providers who offer the same service under better sustainable SLA conditions, or
d) creating sustainability reports with insights underpinned by service usage data from its providers.

3.3.2 Sustainable component descriptions

To support implementing sustainability within and across the ICRA layers, several specific sustainability components are identified (as shown in Figure 3.3). They can be grouped based on the two in scope topics 1) getting insights into and 2) control over the sustainability:

Getting insight components:

- Benchmark, metrics and monitoring

Control over sustainability components:

- Energy consumption and CO2 emission optimization
- Sustainable workload placement and scheduling
- Cooling and heat management
- Renewable energy management

In the next sections each sustainability component is described in more detail.

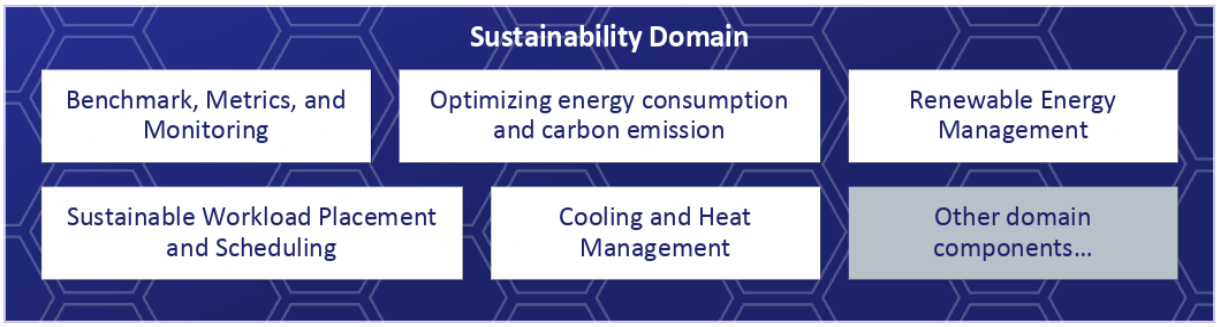

Figure 3.3: *Components in the Sustainability domain.*

3.3.3 Benchmark, Metrics, and Monitoring

This component sets the standards for monitoring services required to support environmental targets and energy cost reduction, as well as financial optimization of ICRA services and infrastructure.

It tracks in real-time (or at a desired aggregated time level) the *energy consumption* of workloads, hardware elements (servers, storage, networking), sites and services, and the energy generation of associated renewable energy sources. It also monitors the *resource consumption* per service or

application, the waste and heat generation, and other potential environmental protection measures. It is divided into three aspects: Monitoring, Metrics and Benchmarking.

The **Monitoring** component also tracks the *availability* and *cost of energy* in energy generation locations, identifying those where free or low-cost energy is available. These are energy producers or consumers with excess energy capacity that otherwise would be wasted if not consumed, like wind turbines, solar power plants and data centers.

Monitoring is important for:

1) getting insight into energy usage and CO2 emissions of the component as a whole and optionally at the component consumer's level,
2) reporting on service usage, and
3) providing (part of) the official sustainability reporting of the service provider (e.g. data center).

Monitoring is typically performed by:

1) Monitoring the physical electricity usage of the used hardware.
2) When possible, calculate/estimate the electricity usage of software resources and components, based on the hardware usage and the dependency tree of the resources. For example, Kubernetes-as-a-Service: On a physical server runs a hypervisor creating VMs. On these VMs are Kubernetes installed.
3) Calculate the CO2-emissions based on the overall energy consumption and attribute the corresponding CO2-emssions to the component as a whole and/or to the individual consumers of the component.
4) Generate reporting for customer service. This can be done on a yearly basis, but also on month, day, hour, 15min or even less, depending on the need of the component consumer.

Metrics are important to have uniform ways to express sustainability KPIs about components and services. These KPIs can be e.g.: energy usage, CO2 emission, water usage, heat production, heat reuse, etc.

Work is needed on standardization and linking to existing standards.

Benchmarking is the structured practice of measuring a process, product or service and comparing the resulting metrics against a meaningful reference, in order to understand performance, identify gaps and prioritize action.

Sustainability benchmarking is important for:

- Consumer: be able to compare the sustainability aspect of offerings amongst each other
- Provider:
    - be able to compare your sustainability performance with competitors.
    - to keep track of the development of own sustainability performance over a longer period of time.
- Government: Stimulate sustainability improvements in the market.

Applied to green software and cloud services, however, the classical formulation (compared against a market average), proves structurally hard:

1) the same digital service can carry very different footprints depending on hardware generation, regional energy mix, data-center efficiency and usage context;
2) embedded carbon and grid composition lie largely outside the operator's control;
3) even renewable, self-produced energy does not exempt a service from the duty to be efficient, because inefficiency still consumes resources that could be used elsewhere.

Although sustainability benchmarking as a comparison tool remains very important, it has not yet been possible to create a fair generic comparison model. What is possible is to use benchmarking as the structured comparison of a system against itself, over time and across scenarios, within a perimeter that is fully known, measured and governable. The result is a shift in posture, from sustainability as an occasional compliance exercise to sustainability as a permanent, measurable, defensible capability of the organization.

3.3.4 Optimizing energy consumption and carbon emission

This component can be used to implement mechanisms to optimize energy consumption and carbon emission of each component and each layer in the ICRA and its corresponding services. Energy consumption and carbon emissions can be optimized in each component in each level of the ICRA, as long as enough information (insight) is available at that level and there are enough sustainability options for the used services to adapt where necessary.

Such optimizations can be grouped along *efficiency* and *flexibility* measures. Efficiency is about reducing energy by using less, which is always the best option. This includes using optimized algorithms, reducing the quality of the workload with minimal impact on accuracy (e.g. through AI model optimization techniques), turning off (or sleeping) unused hardware, use more efficient hardware, etc. efficiency measures are less complex and therefore preferable. When those do not provide enough optimization, flexibility comes into play.

Flexibility steers along *time, speed and location*. Examples for each of these dimensions are:

- Time: Perhaps (part of) the component can be delayed or temporally suspended.
- Speed: Scaling down the resources allocated to a certain application, by lowering the number of nodes, clock speed and/or number of cores.
- Location: Perhaps the hosting location of the component can be changed, for instance to an edge data center close to green electricity.

Note that the flexibility measure partially overlaps with the “Sustainable Workload placement and scheduling” component.

This component provides recommendations for optimization of application and infrastructure setup, based on data from benchmarks and monitoring.

This component may also provide recommendations with respect to application placement to reduce data transfer, optimize energy consumption or balance occupation among sites (reduce risk of congestion).

For the Data layer, the component advises data fusion and data reuse options to minimize redundant data exchange.

3.3.5 Sustainable Workload Placement and Scheduling

This component generates recommendations (conventional or AI based) of temporal and spatial workload movement to balance energy costs, performance, and/or carbon emissions.

The scheduler plans and implements recommendations for moving workloads e.g. ai training, fine-tuning or inferencing temporarily to sustainable or low-cost energy generation locations and providing workload execution assurance. This component should balance the cost of moving workloads (e.g. data transfer or added latency) with the sustainability benefits.

It leverages data center properties (e.g. PUE) algorithms to maximize energy efficiency and resource optimization and to minimize environmental impact. Workload placement also considers the amount of data communication and latency, and the cost of migration into the selection of the optimum location.

The implementation is integrated into or used by a (de-)centralized Multi-Cloud Orchestrator, which is the component able to continuously move workloads between sites (cf. Section 3.9.1).

3.3.6 Renewable Energy Management

This component's purpose is to ensure that the optimal amount of renewable (green) energy is used for powering datacenters. There are many aspects that can play a role in this: CO2 emissions, Certificates, CO2 attribution, batteries, local production and net congestion mitigation.

Calculate for a provider its total CO2 emission. This can be based on LCA to have a full life cycle view but focused on operation based on energy contracts and energy usage.

If the calculated CO2 emissions are not satisfactory, additional CO2 certificates can be bought to reduce the CO2 emissions of the provider on paper. Currently EU certificates with a validation time of a year are used, which makes compensation from another country in another season possible. Upcoming alternatives are hourly based certificates with only local grid usage.

When CO2 emission data is also shared with the provider's consumer, the low CO2 emissions can be even attributed to those consumers with sustainable SLAs.

To optimize the availability of green (low carbon) electricity, batteries can be installed with a battery management system that optimizes on storing only green electricity. Note that these batteries are different than UPS batteries, which purpose is backup.

A provider can increase the availability of green (low carbon) electricity also by adding local green electricity production to its internal electricity net, such as solar and wind. He can be the owner of that but also commercial collaboration with solar or wind plants is possible.

Recommendations can be given for finding optimal locations of placement of (edge) datacenters, and integration of battery storage systems to balance grid capacity.

Net congestion mitigation measures, such as batteries, cooling temperatures, but also compute flexibility, can be installed to be able to react flexible on electricity availability.

3.3.7 Cooling and Heat Management

This component enhances energy and operational efficiency through:

1. Efficient next-generation (immersion and liquid-based) cooling technologies to increase cooling efficiency, to reduce energy consumption, and to support high-density computing.

2. Functionality to enable recovery, upgrade, and reuse of heat energy from cooling systems within data centers, and the subsequent storage and exploitation of the recovered heat.
3. Functionality to support the selection of data center locations to benefit from natural environments factors such as cool climates and other natural cooling opportunities.

3.4 Federation Domain

Federation in cloud-edge infrastructures refers to collaboration between distinct provider, organizational, or administrative domains that interoperate through shared interfaces and conventions while retaining autonomy over their own environments. In the ICRA, the Federation Domain introduces this cross-domain perspective as a first-class architectural concern: it enables participants to share resources and services, extend access, and support workload or data mobility across the Multi-Provider Cloud-Edge Continuum.

Section 4 explains this concept in detail. It first clarifies the main intents of federation, including service composition, resource sharing, access extension, and mobility. It then sets out the federation principles that should guide any implementation. The section also describes the capabilities and technical components typically needed to establish and operate a federation. The detailed component descriptions are therefore not repeated here; readers should refer to Section 4 for the full federation model.

3.5 Application Layer

It provides the functionality required to design and develop applications and functions and to expose them for use. It includes end-user applications and function catalogs.

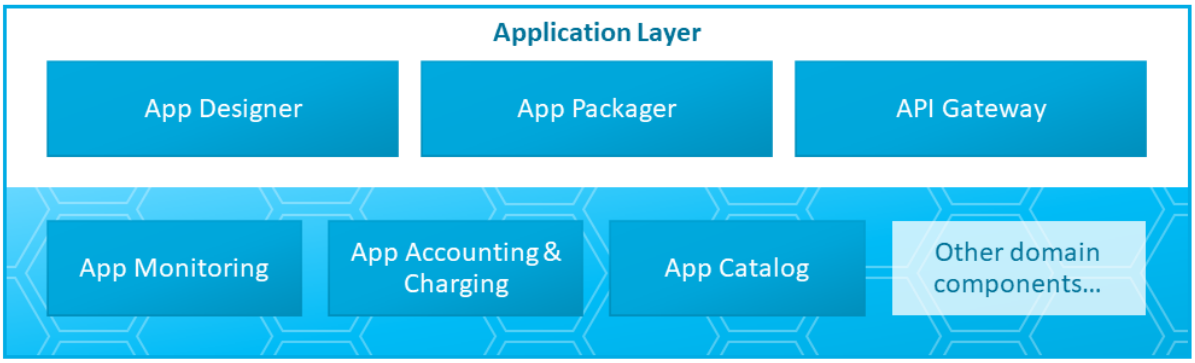

Figure 3.4: *Components in the Application Layer. In blue background, examples of components from management domains associated to that layer.*

3.5.1 Application Designer

This component enables developers to design, create, and customize applications using intuitive interfaces or predefined templates. It facilitates rapid development, integration, and delivery of tailored applications using automated CI/CD practices (DevOps).

The Application Designer facilitates the description of the application in terms of: set of application components it is made of and how they are connected (service function chain), runtime environment that each of the application components will require, including the set of

functions/services to support its execution, the attributes that may allow the selection of the computing node to host it (hardware requirements, latency, privacy, etc.).

It also provides tools for automatic verification and validation (CV/CT) of the application and its supply chain before its final packaging.

3.5.2 Application Packager

The Application Packager supports the packaging of applications for their deployment in the Cloud-Edge Continuum.

It facilitates the automation of application deployment and update (DevOps, both traditional and AI-assisted), providing an integrated toolkit that enables quick, secure and innovative ways to deploy cloud-aware applications.

3.5.3 API Gateway

This component provides the interface to invoke and use the applications contained in the catalog. It checks the identity and authenticates the user and checks his authorization to use the application before providing access to it.

3.5.4 Application Monitoring

It tracks application usage and execution, monitors the performance and identifies abnormal behavior and suboptimal use of resources.

3.5.5 Application Catalog

It implements a directory of applications and functions that the providers have made available. contain the characteristics of the application and the environment it requires for its execution (runtime, services, hardware characteristics).

3.5.6 Application Accounting and Charging

This component implements the accounting of application usage and provides online charging information for the customer to track application expenditure in real-time.

3.6 Data Layer

This layer offers simple, scalable, sustainable, and trustable mechanisms for collection, processing, and exchange of data distributed over the Cloud-Edge Continuum.

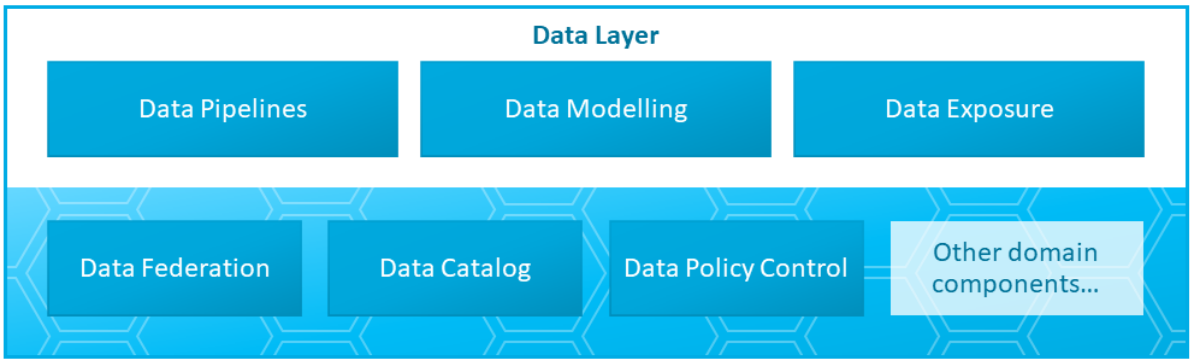

Figure 3.5: *Components in the Data Layer. In blue background, examples of components from federation and management domains associated to that layer.*

3.6.1 Data Pipelines

This component provides the functionality for data collection, including the connectors to integrate with the data sources and the capabilities for data curation and pre-processing that ensure its quality and readiness for analytics, insight generation, training, modelling, or inferencing phases.

3.6.2 Data Modelling

This component enables data cataloguing to enable exposure and discovery at scale to easily search, find, and browse data, over a distributed environment.

3.6.3 Data Exposure

The *Data Exposure* component provides customers with standard mechanisms and interfaces for safe and controlled access to data. It includes capabilities for making data offers and contracting data acquisition, identity checking, and data access authentication and authorization.

3.6.4 Data Policy Control

Data Policy Control sets the required policies for data sharing, providing a safe, controlled and regulation-compliant environment for data exchange. It allows the data owner to manage the permissions to access its data: who can make it, at which conditions and for which purposes.

3.6.5 Data Catalog

Data Catalog provides efficient storage and indexing of data to facilitate browsing, searching and finding data over a distributed environment.

3.6.6 Data Federation

Data Federation enables standard mechanisms and interfaces (connectors) for partnering in the provision of datasets, providing a unified view of data catalogs and databases from multiple data providers.

This component enables real-time data exchange across companies using data mesh principles, connecting distributed and heterogeneous actors over the Cloud-Edge Continuum, keeping data owners in full control of their data.

In order to create and maintain a coherent federated Multi-Provider Cloud-Edge Continuum, data federation capabilities should be designed consistently with the other federation capabilities described in this document.

3.7 AI Layer

The AI layer provides a set of advanced functionalities to apply *Artificial Intelligence* (AI) natively across the Cloud-Edge Continuum, with distribution of data and processing resources as the default rather than an add-on to centralized cloud.

This layer facilitates the seamless integration of AI model lifecycle management into the Cloud-Edge Continuum, enabling efficient operations across distributed environments. Its capabilities are tailored to the specific characteristics of the Cloud-Edge Continuum, ensuring optimal performance and resource utilization. It provides:

- end-to-end solutions for the AI lifecycle, including preprocessing, model training, and evaluation
- inference with deployment, monitoring, operations across cloud and edge, and
- agents with tools, context interfaces, and workflows for coordinated AI execution, including governed access to external tools, resources, and data sources through agent-facing interfaces such as MCP.

The distribution of AI workloads across the cloud-edge continuum is driven by several rationales that often combine in practice:

- latency requirements for human, robotic, or industrial interaction (standard edge definition, https://digital-strategy.ec.europa.eu/en/policies/edge-observatory),
- data locality, where data must remain at its source for privacy, regulatory, or sustainability reasons,
- privacy-aware operation, where data and models stay within an edge governance perimeter unless explicit policy controls authorize their movement, and
- sovereignty over computing, where existing edge resources are orchestrated and parallelized to perform distributed training or inference without dependence on external infrastructure.

These rationales widen the operational definition of "edge" in the AI Layer beyond network-distance measures: from the AI Layer's perspective, an edge is any environment where placement is justified by latency, data locality, privacy governance, or sovereignty.

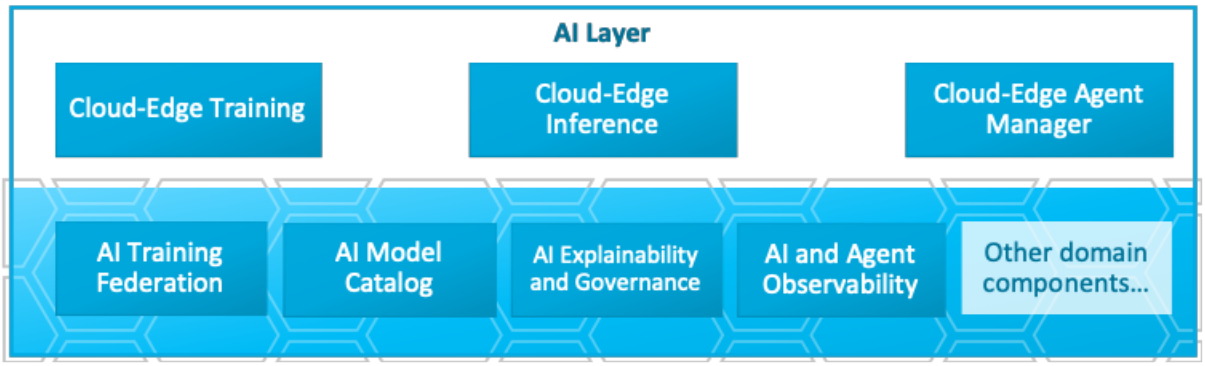

Figure 3.6: *Components in the Intelligence Layer. In blue background, examples of components from federation, management and security domains associated to that layer.*

3.7.1 Cloud-Edge Training

This component facilitates the dynamic and adjustable training of AI models across cloud and edge environments, ensuring scalability, reduced latency, and optimized resource utilization. It covers training within a single provider's infrastructure, including distributed training where workloads are split across multiple nodes under central orchestration, and the placement of training jobs according to resource, cost, and sustainability constraints. Training that spans multiple independent providers is addressed by the AI Training Federation component.

3.7.2 Cloud-Edge Inference

The inference components facilitate real-time deployment and execution of trained AI models on across the cloud-edge continuum, with placement chosen to match resource availability, latency requirements, and data-governance constraints. Models run

- at the edge when local infrastructure is sufficient and data must remain in place,
- on sovereign cloud nodes operated by a federated provider when local resources are insufficient (for example, for large language models), or
- in hybrid configurations where parts of an inference workflow are split between edge and sovereign cloud for resource optimization.

Edge deployments may apply techniques such as model quantization, pruning, or distillation to fit resource-constrained environments. Sovereign cloud and hybrid configurations may apply additional security and compliance components to fit data-governance constraints, for example ephemeral storage and secure channels protecting data in transit.

The inference plane is synchronized with the cloud for updates, monitoring, and lifecycle management. This distinguishes federated cloud-edge inference from inference delegated to AI providers outside the federation, where the model and runtime fall outside its sovereignty and governance guarantees.

3.7.3 AI and Agent Observability

This component provides AI-specific monitoring that complements the generic capabilities in the Management Domain. It covers model performance and drift detection, output quality signals such

as hallucination indicators for generative models, inference latency, and token usage. For agentic workloads, it captures full execution traces including model calls, tool invocations, MCP and A2A interactions, handoffs, approvals, and policy decisions. Replicable traces support audit, evaluation, and post-market monitoring obligations under the AI Act.

3.7.4 Cloud-Edge Agent Manager

The Cloud-Edge Agent Manager enables the deployment and management of agents and agentic workflows on edge and hybrid edge-cloud deployments. It manages agent identity, exposed tools, MCP endpoints, transport, authentication requirements, tool schemas, owner, version, risk level, persistent state, policy enforcement, auditability, and observability hooks, creating an agentic mesh that can be operated across providers.

Federated discovery of agents and MCP endpoints across providers is enabled in coordination with the Federation Domain, preserving endpoint identity, authorization, versioning, data residency, and audit requirements.

3.7.5 AI Training Federation

AI workloads can be split across multiple nodes with central orchestration for scalability and operational efficiency (distributed AI). AI federation enables autonomous nodes to collaborate securely, ensuring privacy and sovereignty. Together, they balance task-sharing efficiency with autonomy.

In distributed AI training, the AI model is generated at a central point based on the combination of models produced by different training agents distributed across an ecosystem of federated AI service providers or owners. The distributed training agents work locally on local datasets, reducing the need to transfer data to a central location for training.

This component allows users to use and orchestrate AI resources across multiple providers to collaboratively perform a specific machine learning training task. It leverages a federated network of AI capabilities geographically distributed across the Multi-Provider Cloud-Edge Continuum, enabling accessible resource sharing and scaling while maintaining sovereignty and compliance. It ensures efficient distribution of AI computational workloads, minimizes data movement, and facilitates parallel model training without requiring centralized data aggregation, thus preserving data privacy and autonomy while enhancing overall system performance.

In order to create and maintain a coherent federated Multi-Provider Cloud-Edge Continuum, federated learning capabilities should be designed consistently with the other federation capabilities described in this document.

3.7.6 AI Model Catalog

The AI Model Catalog holds trained models available for deployment across the Cloud-Edge Continuum, spanning the full range of model types: classical machine learning models (such as regression and tree-based models), task-specific deep learning models (such as vision and speech models), and foundation models (LLMs, SLMs, and multimodal models). For each model it maintains

metadata supporting discovery, versioning, provenance, licensing, and intended use, enabling applications and orchestration components to select appropriate models for a given task and deployment context.

Models can be fine-tuned and adapted to specific use cases using techniques such as RAG, quantization, pruning, or distillation. Model selection often balances capability against resource footprint, which is particularly relevant at the edge, where smaller and specialized models may be preferred.

As one notable example, the catalog includes multilingual and multimodal foundation models tailored to diverse EU languages and data types. These address the scarcity of generative AI solutions in non-English languages, supporting semantic precision and completeness, with metadata that supports AI Act compliance.

3.7.7 AI Explainability and Governance

This component ensures transparency, accountability, and regulatory alignment of AI systems across the Cloud-Edge Continuum. It provides interpretable insights into AI decision-making, maintains model cards and technical documentation, and supports risk classification and compliance with the AI Act. Decision logging and audit trails enable human oversight and post-market monitoring, while respecting privacy and data sovereignty.

3.8 Service Orchestration Layer

The service orchestration refers to the integration and management of multiple cloud and edge services and applications in a unified and automated way across diverse multi-provider environments. In the context of multi-cloud and edge ecosystems, service orchestration is essential for handling the complexities of deploying and managing applications at scale, especially when dealing with distributed resources across cloud and edge infrastructures.

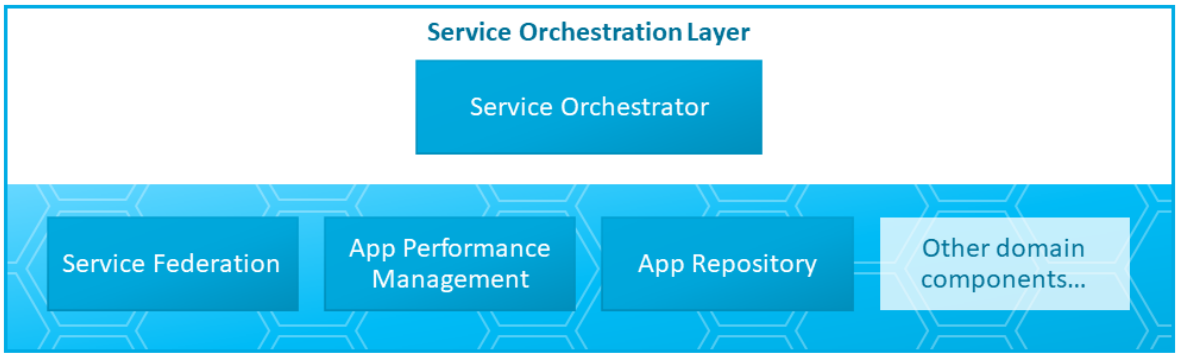

Figure 3.7: *Components in the Service Orchestration Layer. In blue background, examples of components from federation, management and security domains associated to that layer.*

3.8.1 Service Orchestrator

Service orchestration assures efficient tasks execution, load balancing, and real-time operations. For example, it could communicate with the Multi-Cloud Orchestrator (cf. Section 3.9.1) that manages the virtualized infrastructure layer offering a single unified environment for application development

and monitoring. This allows applications and services to be deployed seamlessly across multiple platforms, optimizing resource allocation and reducing operational complexity. Alternatively, the Service Orchestrator may directly or indirectly interact with the underlying capabilities of the cloud platform or virtualization management layer to orchestrate workload execution.

The Service Orchestrator automates application and tenant deployment, and lifecycle management processes. By automating these workflows (or service function chains), orchestration ensures that services communicate efficiently across the Cloud-Edge Continuum.

3.8.2 Application Performance Management

It monitors the performance and resource consumption of the application or service and communicates deviations from set thresholds or SLAs to the Service Orchestrator for this to take actions to recover a state that meets application requirements.

It provides a unified view of states, including logging, monitoring, and alerting, for effective real-time application management and validation at runtime.

3.8.3 Application Repository

This component tracks the applications and services that have been deployed and their configuration, the locations where the application and service components are installed, and the resources they are consuming.

3.8.4 Service Federation

This component interconnects the Service Orchestrator with those of other federated providers, enabling the deployment and execution of applications (service function chains) across multiple providers in a seamless way, interacting with a single provider.

In order to create and maintain a coherent federated Multi-Provider Cloud-Edge Continuum, Service Federation capabilities should be designed consistently with the other federation capabilities described in this document.

3.9 Cloud-Edge Platform Layer

The Cloud-Edge Platform Layer hosts components that manage and orchestrate runtime environments (virtual machines, containers, serverless) and platform resources (libraries, functions, services, security, databases, tools).

The platform services (PaaS) it provides support the deployment of applications and the corresponding runtime environments and platform resources.

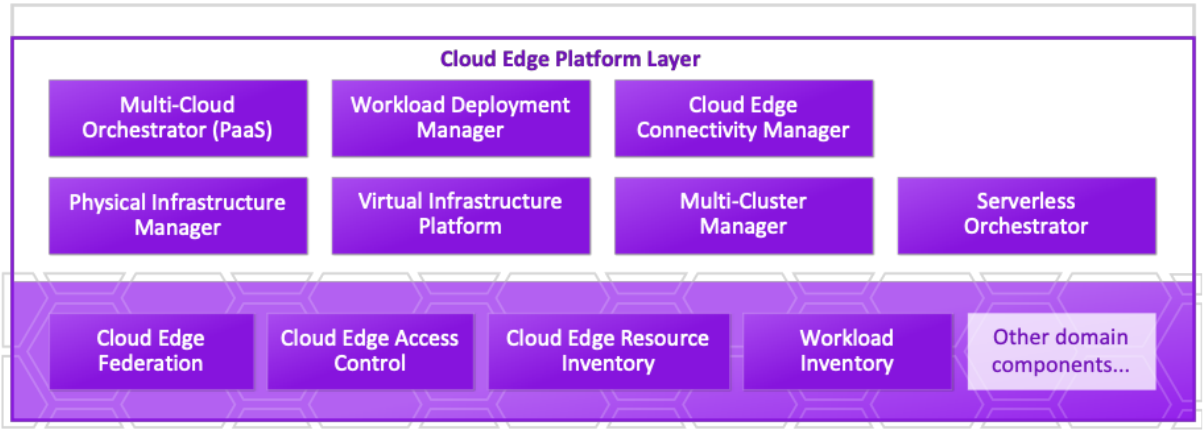

Figure 3.8: *Components in Cloud-Edge Platform Layer. In purple background, examples of components from federation, management and security domains associated to that layer.*

3.9.1 Multi-Cloud Orchestrator

MCO delivers a PaaS service. A PaaS provides a complete application development and deployment environment in the cloud. With PaaS, customers can build, test, deploy, manage, and update applications quickly and efficiently, without worrying about the underlying infrastructure.

It receives from the Service Orchestrator (cf. Section 3.8.1) a request to deploy (or manage the lifecycle of) a certain application together with a descriptor (resource model) that defines the state the application needs for its execution (including runtime environment, services, data, application image and other attributes like area of service, performance...). The MCO processes the state and takes actions to set it up and preserve it, by updating, upgrading, or removing workloads and services, or rescaling or releasing resources.

MCO works in close relationship with other components (*Physical Infrastructure Manager* – PIM, *Virtual Infrastructure Platform Manager* – VIP, *Multi-Cluster Manager* – MCM, Serverless orchestrator) to provide the virtual runtime environment defined for the application, the specific combination of bare metal, virtual machine, containers, and serverless mechanisms, it has been developed to run on, using the technologies over which it has been tested and certified.

MCO also deploys and manages the lifecycle of essential tools and services such as middleware, development frameworks, databases, and business analytics, enabling organizations to streamline application development and drive innovation.

A PaaS, managed by the MCO, offers scalability, high availability, and reduced time-to-market, allowing developers to focus on coding and application functionality while the MCO supports with infrastructure, security, and operational aspects.

Based on certain attributes, like area of service and performance, the MCO may select the location(s) where to deploy the workload and the resources (physical and virtual) required at those location(s) to meet the desired state. This decision on application placement can also follow sustainability and privacy requirements.

The MCO deploys the workload once the necessary resources are available, using the *Workload Deployment Manager* (WDM). The MCO also updates and removes workloads, rescaling or releasing the corresponding resources.

This MCO description shows a decomposition of the functionality of a Cloud-Edge Continuum workload management solution that may be implemented in many ways, combining or excluding some of its components in order to fit specific sector needs.

3.9.2 Cloud-Edge Connectivity Manager

The *Cloud-Edge Connectivity Manager* (CEC) implements and modifies the service function chain, or removes it, totally or partially, following the requests from a Service Orchestrator, to guarantee the connectivity between workloads that will enable the service delivery and the connectivity from the service user to the workloads implementing the service front-end.

Connectivity is usually based on both overlay and underlay components in each domain crossed by the traffic (e.g. WAN, data centers, etc.). The CEC manages the networking in the data center domain through the virtualization managers (*Virtual Infrastructure Manager* – VIM, *Container Infrastructure Service Manager* – CISM) or via specific *Network as a Service* (NaaS) interfaces. It manages the WAN connectivity using cloud networking services (via transport SDN Controllers) for the connection of different computing nodes.

In addition, the CEC manages the complexity deriving from the need to ensure consistency between overlay and underlay networking solutions (for example adapting the networking between the data center fabric and the WAN connectivity).

3.9.3 Physical Infrastructure Manager

The PIM monitors and manages a pool of physical resources (CPUs, storage, networking) and selects and prepares them (with the corresponding OS and necessary software) to allocate these resources to a virtual machine or container cluster.

The PIM provides multiple physical infrastructure management functions, including physical resource provisioning and lifecycle management, physical resource inventory management, or physical resource performance management.

3.9.4 Multi-Cluster Manager

The MCM creates and configures container clusters both over bare metal and over virtual machines upon request from the MCO, offering a single interface to manage infrastructure from multiple providers and with multiple k8s distributions.

The MCM provides open connectors/APIs to interact with the resources and k8s distributions offered by different providers (private and public) for cluster creation, configuration, and monitoring, and keep track of their evolution.

The MCM may create a k8s cluster on bare metal (cluster nodes are servers) or on the virtualization stack (cluster nodes are VMs), interacting with PIM or VIP respectively.

3.9.5 Virtual Infrastructure Platform Manager

The *Virtual Infrastructure Platform Manager* (VIP) creates virtual machine clusters across several locations using the resources allocated by the PIM.

The VIP is required when the service component to be deployed is a virtualized application or a containerized application, which runs over container clusters that make use of VMs (virtual machines).

This component works on infrastructure and technology from different providers, enabling the Cloud-Edge Continuum to run on a diverse set of different virtualization solutions (VIMs, CISMs, or any other future virtualization technology).

3.9.6 Workload Deployment Manager

The WDM deploys software package(s) on top of an existing cluster, following the request of the MCO. It exposes a single interface to deploy software packages (i.e., via a helm chart or resource model declaration) on any k8s cluster (or alike) based on any distribution.

The WDM provides the connectors/APIs to interact with existing clusters in different locations and technologies (k8s distributions) for application deployment and lifecycle management.

This component can also deploy software packages directly on virtual machines (IaaS).

3.9.7 Cloud-Edge Federation

This component interconnects the MCO with the ones of other federated providers, enabling the consumer to use cloud-edge computing services (IaaS, CaaS, PaaS, Serverless, NaaS, ...) across multiple providers seamlessly, interacting with a single provider.

This platform federation provides seamless integration and collaboration between multiple cloud platform providers, enabling interoperability, resource sharing, and unified lifecycle management. Shared resources may exist on all layers of cloud architecture.

By adopting standardized protocols and interfaces, platform federation facilitates enhanced scalability, efficiency, and innovation across different cloud environments while maintaining autonomy and security for each participating entity.

In order to create and maintain a coherent federated Multi-Provider Cloud-Edge Continuum, cloud-edge federation capabilities should be designed consistently with the other federation capabilities described in this document.

3.9.8 Cloud-Edge Access Control

This component implements a key aspect in terms of security in the management of cloud-edge infrastructure, a role-based access control that ensures proper access rights and security across the infrastructure.

3.9.9 Cloud-Edge Resource Inventory

This component keeps a record of the resources available in each of the edge locations, the virtualization platforms available, and the configuration. The information in this repository helps the MCO to select the right location(s) to deploy workloads.

3.9.10 Workload Inventory

This component keeps record of the workloads that have been deployed and their configuration, as well as information about the location and cluster where they have been deployed and the resources they are consuming.

3.9.11 Serverless Orchestrator

The Serverless Orchestrator provides serverless capabilities, also known as *Function as a Service* (FaaS), which is a cloud computing model that allows developers to build and deploy applications in the form of individual functions, that are executed in response to specific events or triggers. This model eliminates the need to manage server infrastructure, enabling developers to focus solely on writing code. Each function runs in a stateless container, automatically scaling with demand and only consuming resources when invoked, leading to cost savings and efficient resource utilization.

3.10 Virtualization Management Layer

The Virtualization Layer in cloud infrastructures acts as a crucial abstraction layer that enables the efficient utilization of physical resources by creating multiple virtual instances of servers, storage, and network resources. This layer leverages hypervisors or related technologies to decouple hardware from the operating system, allowing for the dynamic allocation and scaling of resources based on demand. By providing a flexible and scalable environment, the virtualization layer enhances resource optimization, simplifies management, and supports the seamless deployment of various cloud services and applications. This foundational component is essential for delivering IaaS and other cloud service models, ensuring agility, cost-effectiveness, and high availability.

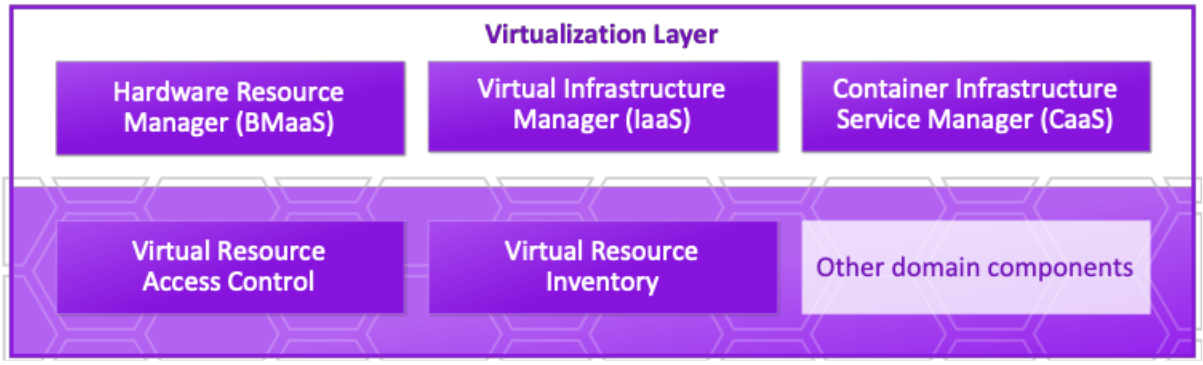

***Figure 3.9**: Components in the Virtualization Layer. In purple background, examples of components from management and security domains associated to that layer.*

3.10.1 Hardware Resource Manager

The Hardware Resource Manager component delivers a *Bare Metal as a Service* (BMaaS) service. BMaaS is an abstraction that provides physical, non-virtualized hardware resources directly to users, offering dedicated servers, storage, and networking components without any virtualization layer. This service allows users to harness the full power of the hardware for applications, resulting in higher performance, predictable latency, and complete control over the environment. BMaaS is particularly beneficial for workloads that require intensive computation, low-latency networking, or compliance with specific hardware configurations.

3.10.2 Virtual Infrastructure Manager

The VIM component provides IaaS service. IaaS is a cloud computing model that provides virtualized computing resources. IaaS delivers essential services such as virtual machines, storage, and networks. Users can provision, scale, and manage the resources dynamically according to their needs, while the cloud provider takes care of maintaining the underlying hardware, networking, and security. This model offers high flexibility, enabling organizations to quickly deploy and run applications and services, test new solutions, and handle varying workloads with ease, ultimately driving innovation and operational efficiency.

3.10.3 Container Infrastructure Service Manager

The CISM component provides a CaaS service. CaaS is a cloud service model that provides a platform allowing users to manage and deploy containerized applications and workloads. By leveraging container orchestration tools such as Kubernetes, CaaS facilitates the automation of container deployment, scaling, and operations, ensuring high availability and performance. This model abstracts the underlying infrastructure complexities, enabling developers and IT teams to focus on application and service development and deployment without worrying about the maintenance of the physical or virtual infrastructure.

3.10.4 Virtual Resource Access Control

As in the cloud-edge platform layer, this access control component implements virtual infrastructure management security, a role-based access control that ensures proper access rights and security for virtual resource management.

3.10.5 Virtual Resource Inventory

This component keeps records of cloud-edge sites and the configuration and availability of virtual resources in each one of them (for instance, number of k8s clusters available per site, CPU/memory available per k8s cluster, number of virtual CPUs that are available to setup new k8s clusters, ...) in order to help make decisions on workload placement.

3.11 Physical Cloud-Edge Resource Layer

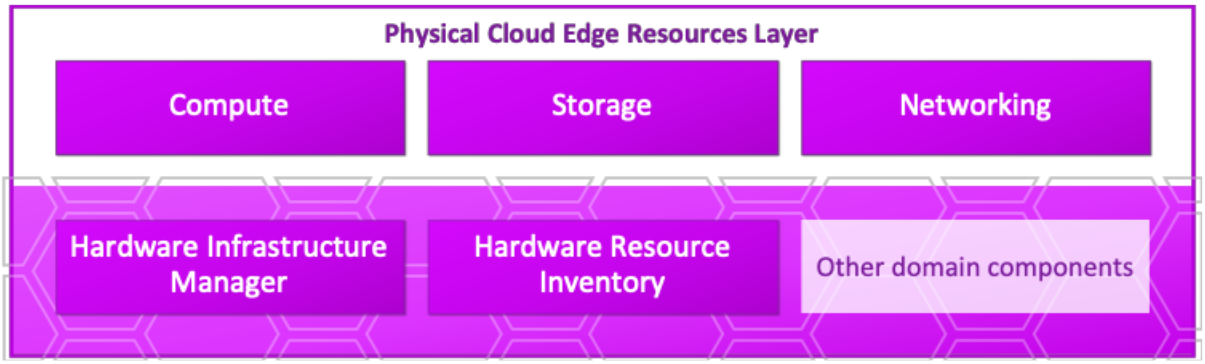

Figure 3.10: *Components in the Physical Cloud-Edge Resource Layer. In pink background, examples of components from management domain associated to that layer.*

3.11.1 Compute

Compute resources are fundamental to cloud infrastructure, delivering the computational power required for running applications and services. They facilitate scalable and efficient environments that dynamically adjust to varying workloads, thus enhancing resource utilization and performance while minimizing costs.

3.11.2 Storage

Storage is essential in cloud infrastructure, providing data persistence, management, and accessibility. It includes block storage for databases, object storage for unstructured data, and file storage for shared access applications. Advanced technologies like SSDs and distributed file systems ensure scalability, reliability, and performance.

3.11.3 Networking

Hardware networking resources in a cloud-edge location include routers, switches, load balancers, and firewalls. These components form the backbone of data center connectivity and inter-server communication. *Network Interface Cards* (NIC) in servers enable high-throughput connections to the virtual network. WAN gateways and edge routers extend connectivity to external networks, supporting hybrid cloud and remote access scenarios. All hardware is managed centrally through SDN controllers and scaled dynamically to support edge-cloud service demands.

3.11.4 Hardware Infrastructure Manager

A *Hardware Infrastructure Manager* (also known as *Data Center Infrastructure Management* system, DCIM[4]) is a management component designed to monitor, measure, and manage the IT equipment and infrastructure within a cloud-edge data center. It encompasses the following key aspects:

[4] Relevant standards: 1) ISO/IEC 18598, which focuses on automated infrastructure management (AIM) and includes aspects relevant to DCIM, 2) ITU-T L.1305 outlines technical specifications for DCIM systems, including principles, management objects, and operational function requirements

- Monitoring and Management: It provides real-time monitoring of data center operations, including power usage, cooling efficiency, and physical security. This helps in optimizing the performance and efficiency of the data center.
- Documentation and Planning: It maintains detailed documentation of the data center's physical and virtual assets. This includes layout planning, capacity management, and future expansion plans.
- Risk Management: By continuously monitoring environmental conditions and equipment status, it helps in identifying potential risks and mitigating them before they lead to failures.
- Integration with IT Systems: It integrates with other IT management systems to provide a holistic view of the data center's operations, facilitating better decision-making and resource allocation.
- Sustainability and Compliance: It supports sustainability goals by optimizing energy usage and ensuring compliance with industry standards and regulations.

3.11.5 Hardware Resource Inventory

This component keeps records of cloud-edge locations and the configuration and availability of physical hardware resources in each one of them (for instance, number of servers per location, type of servers, type of NIC cards available per location, cost of resources, energy consumption of resources, ...) in order to help make decisions on workload placement and resource lifecycle management.

4. Federation

In cloud-edge infrastructures, federation refers to a form of collaboration in which cloud environments from distinct domains, organizations, or providers interoperate through shared interfaces and conventions, while each participant retains autonomy over its own environment. Federation relies on shared interoperability mechanisms to enable consistent interaction (e.g., standard APIs and shared information models, or semantic interoperability across heterogeneous models). It preserves the security, governance, and policy boundaries of each participating entity, while enabling cross-domain interoperability, resource sharing, and service exposure via standardized protocols and interfaces. Examples of such interfaces are outlined in the next section, “Functional Interfaces”.

Federation can serve different intents. In some scenarios, it can support *service composition*, where services from different providers are selected and combined into a single solution. In other scenarios, it supports *resource sharing and access extension*, where a consumer or a provider acting on the consumer’s behalf can use resources in another domain under agreed policies. Examples include seamless cross-domain access (roaming-like experiences in telco contexts), capacity overflow, and geographic reach. Across both intents, federation decisions are typically driven by technical constraints and non-technical constraints. Technical constraints include interoperability, performance objectives such as latency, throughput, availability, resilience, and security baselines. Non-technical constraints include regulatory and sovereignty obligations, sustainability targets, and commercial/ contractual conditions.

From a consumer perspective, federation commonly enables two primary approaches:

- Service Composition & Integration: Consuming external provider services and integrating them either from scratch or with existing distributed workloads.
- Resource Sharing, Access Extension & Mobility: Using resources across provider domains under agreed policies. This may include migrating or relocating workloads and/or data when needed while maintaining functional and quality requirements.

Across both approaches, resource allocations can remain flexible throughout the workload lifecycle. Both the services consumed by workload and the workload’s execution location can be dynamically reassigned or relocated in response to evolving business requirements, performance demands, cost optimization objectives, or sustainability imperatives. Depending on the service characteristics and operational scenario, this reallocation may be consumer-initiated or executed through inter-provider coordination.

4.1 Federation Principles

The core principles described here are designed to be universally applicable across diverse deployment scenarios: whether working with many providers, a few providers, or even a single provider with multiple locations. These principles should seamlessly support all connectivity patterns, including data center to data center, data center to edge, and edge-to-edge deployments.

A fundamental principle is to avoid technological lock-in. Any federation should rely on open, well-specified, and technology-agnostic intermediate abstractions, such as interfaces and data models. Provider-specific technologies should be mediated through those abstractions. The abstractions expose capabilities and policies in a technology-neutral way while allowing providers to implement them within their preferred stacks. The Reference Architecture provides a common and standardized foundation by defining roles, responsibilities, and interaction patterns across federated domains. It does not prescribe specific technologies, but it ensures that different federations can coexist as long as they comply with these shared principles. By aligning on a common model, new federation participants can integrate more easily without negotiating bespoke integrations. Different federation frameworks, including those developed in specific IPCEI-CIS projects, can coexist as long as they adhere to those responsibilities. This supports flexibility and interoperability across different vendor solutions.

Similarly, a lock-in to a single federation participant is generally undesirable. The architecture encourages flexible federation models where federation roles and functions may be operated by different participants that want to provide service and/or resources. The role of a participant is not fixed; the same entity may act as a provider in one federation relationship and a consumer in another. The architecture should not rely on any single privileged operator or specific technology. It strongly stimulates multiple, independently operated implementations to coexist and to be plug-replaceable, so that participants can change providers or implementations without disrupting the overall federation.

The architecture supports collaboration between many independent participants at the same time, for example, multiple cloud and edge providers, federation middleware operators, marketplaces, trust/credential authorities. Several federations may coexist and interconnect. No single participant is assumed or required to act as a global or central federation operator.

To work effectively across various cloud-edge providers, network interconnection and transport providers, service marketplaces, and consumer organizations, a common understanding of the objects of interest is essential for consistent discovery, comparison, and composition across providers. As such, it is strongly recommended to describe services, their compositions, policies, and SLAs in a machine-readable or even machine-understandable, shared information model (e.g., an ontology-based model using semantic web standards). This standardized approach should be applied to all layers within the cloud-edge federation.

Trust, governance, and compliance are foundational elements in federations. All organizations involved in a federation must respect regulatory, contractual, and sector-specific governance requirements. This entails implementing cross-provider identity and access management and establishing mechanisms for expressing and enforcing policies across domains. It also requires machine-readable (or preferably machine-understandable), verifiable representations of organizational attributes, such as a provider’s headquarters location, certifications (e.g., ISO), and regulatory compliance. Trust models may incorporate external trust frameworks and credential systems to deliver cryptographically verifiable evidence of identity, attributes, and compliance.

Finally, in line with the regulations on switching Data Processing Services into the Data Act[5], federation components and mechanisms should not introduce additional switching barriers beyond those of the underlying services and should, where possible, help to remove existing obstacles rather than create new forms of lock-in.

Building on the principles above, the following section summarizes the concrete capabilities a federation typically needs to support. These capabilities span both initial onboarding and day-to-day operation across multiple provider domains.[6]

4.2 Federation Capabilities

This section lists the capabilities that a cloud federation can support for establishment and for ongoing operation. These capabilities are technology-agnostic and apply to the whole cloud-edge continuum.

The list is neither exhaustive nor prescriptive. Additional capabilities may exist beyond those described here, and specific federations may choose to implement only a subset depending on their scope, maturity, and use cases. However, the capabilities outlined in this section represent a comprehensive set of aspects that should typically be considered when designing, evaluating, or implementing a cloud federation.

For readability, the operational capabilities in this section are grouped to follow a typical workload journey: discover and select offers, order and configure access, run and observe workloads, and (when needed) relocate data and workloads or switch providers.

4.2.1 Establishment (onboarding into a federation)

4.2.1.1 Governance & Participation

- Define participation rules (eligibility, conformance, auditability, dispute handling).
- Contracting and commercial agreements within the federation are established between participants. This includes terms, pricing constructs, liability, SLA templates, and settlement models. The architecture supports both upfront contracting models and dynamic trust/policy agreement models without favoring either. Commercial relationships between a participant and its own end clients fall outside the federation scope and remain governed independently by each participant.
- Register participants and bind them to verifiable organizational identities and attributes.

[5] See Chapter VI: “Switching between data processing services” in the Data Act.

[6] Editorial note: The exact scope of "Federation Participant", explicitly including the participation of consumers to the federation, is under active discussion. The next version of the ICRA may introduce changes in that aspect.

4.2.1.2 Trust Establishment

NB: these capabilities relate to Section 3.2 "Security and Compliance Domain".

- Establish federation identity and trust anchors. This includes credential issuance and/ or verification, key material, and endpoint metadata.
- Align on security profiles and minimum baselines for transport, authentication, authorization, and incident handling.

4.2.1.3 Interoperability Setup

- Adopt shared service description conventions: machine-readable/ machine-understandable descriptions including functional, non-functional, and commercial compatibility information.
- Provide conformance and interoperability testing.

4.2.1.4 Interconnection Readiness

- Ensure control-plane reachability between participants. This includes secure transport, routing constraints, and (where applicable) federation context identifiers.

4.2.2 Operation (running workloads across the federation)

4.2.2.1 Discovery & Selection

- Discover and compare offers across providers using shared information models.
- Assess technical compatibility for composing multi-provider workloads, based on declared functional and non-functional properties such as interfaces, data formats, resource requirements, network characteristics, and supported protocols.
- Perform policy-aware selection (e.g., sovereignty, certifications, sustainability constraints).

4.2.2.2 Ordering, Provisioning & Configuration

- Initiate orders/subscriptions and track order state.
- Perform cross-provider technical onboarding and configuration required to realize an order, i.e. establishing the necessary accounts, identities, policy bindings, and connectivity between the provider's domains involved.

4.2.2.3 Cross-domain Access Enablement

NB: these capabilities relate to Section 3.2 "Security and Compliance Domain".

- Enable federated authentication and authorization for control-plane actions, with auditability and delegation semantics.
- Represent "who acts" and "who is accountable" across domains for logging, audits, and UI.
- Lifecycle & Operations

NB: the second capability relates to Section 3.1 "Management Domain".

- Provision/deploy/scale/update/terminate resources and services across providers through federated abstractions.
- Support monitoring, events/notifications, and operational coordination across domains.

4.2.2.4 Portability & Switching

- Support workload and data relocation across domains when required, subject to provider policies. Typical triggers include latency, availability, capacity overflow, or regulatory constraints.
- Reduce switching friction by standardizing discovery, trust, and control-plane interactions across providers.
- Require services to declare data take-in/take-out processes, responsibilities, and constraints.

4.2.2.5 Metering, Accounting & Settlement

NB: these capabilities relate to Sections 3.1 "Management Domain" and 3.3 "Sustainability Domain".

- Provide standardized usage metering records and accounting hooks to support various business models, sustainability, etc.
- Allow implementation by providers, intermediaries, or consumer tooling, while avoiding metering and accounting approaches that introduce new switching barriers.

4.3 Federation Components

- Section 4.2 describes *what* a federation should be able to do. This section describes *how* those capabilities are realized through a set of technical components. **Figure 4.1** presents an overview of those component overview, whereas **Table 1** provides a quick mapping from capability areas to the components that typically implement them. The presented mapping is not binding; alternative mappings are possible.

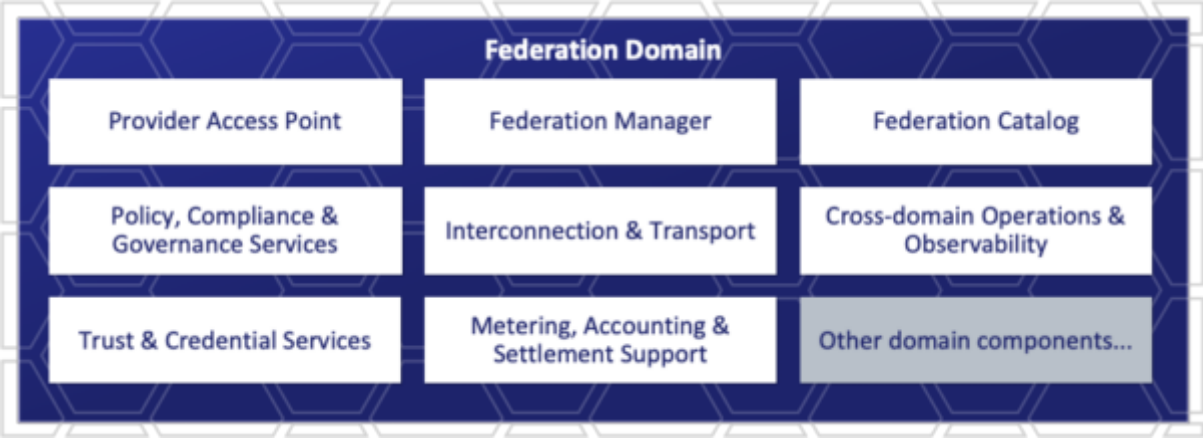

Figure 4.1: *Components in the Federation domain.*

Capability area	Typical primary technical component(s)
Governance & Participation	Policy, Compliance & Governance Services; Provider Access Point (exposure of rules and constraints)
Trust Establishment	Trust & Credential Services; Federation Manager
Interoperability Setup	Federation Manager; Federation Catalog (optional, e.g. through service description conventions)
Interconnection Readiness	Interconnection & Transport; Federation Manager
Discovery & Selection	Federation Catalog; Policy, Compliance & Governance Services
Ordering, Provisioning & Configuration	Provider Access Point; Federation Catalog (optional ordering); Federation Manager
Cross-domain Access Enablement	Federation Manager; Trust & Credential Services
Lifecycle & Operations	Federation Manager; Cross-domain Operations & Observability
Portability & Switching	Federation Manager; Provider Access Point; Policy, Compliance & Governance Services
Metering, Accounting & Settlement	Metering, Accounting & Settlement Support; Federation Manager

Table 1: *Mapping of federation capabilities to federation components.*

The remainder of this section defines each component's purpose, responsibilities, and common variants.

At a high level, a federated cloud ecosystem consists of multiple providers that collaborate while maintaining independent environments. Each provider owns and manages an environment with local infrastructure, services, and policies, and can expose both its own services and services sourced from other providers. Depending on the federation pattern, consumers access services either through a provider-managed access point (that typically hides federation-specific complexity), via consumer-operated federation components such as a Federation Manager, or directly.

A typical setup with the three primary components, Provider Access Point, Federation Manager, and Federation Catalogue, is visualized in **Figure 4.2.**

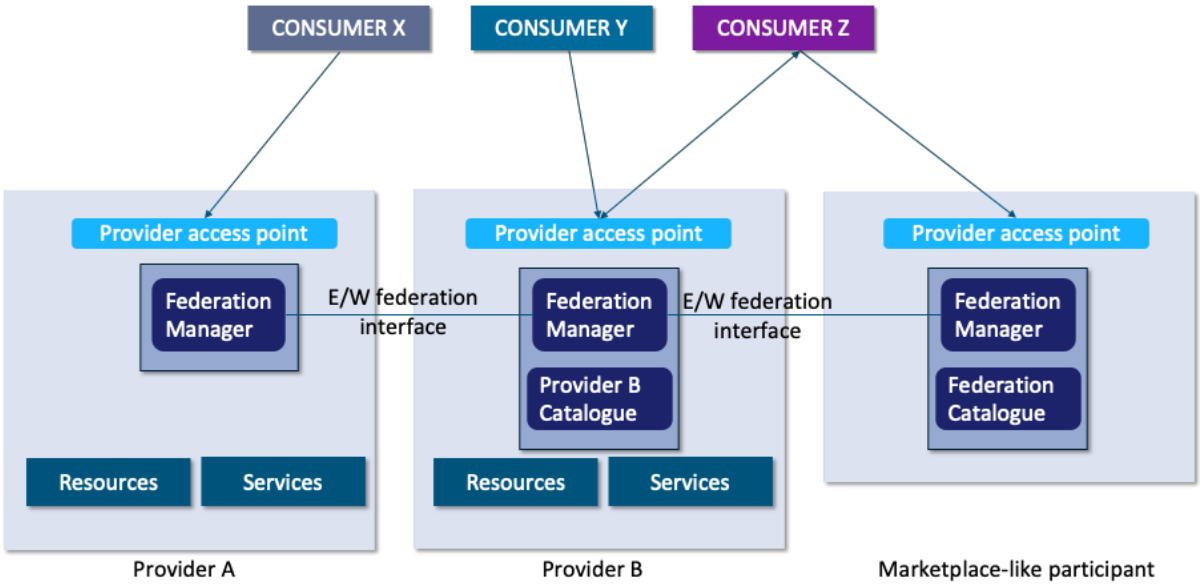

Figure 4.2: *Example set-up of a federation with three providers. Their resources and services are discoverable through the Federation Catalogues. Consumers acquire resources and services through the Provider Access Point. The Federation Managers communicate with each other through an East/West (E/W) interface to set up the federation and take care of ordering and provisioning. Consumer Y can access all deployed resources through provider B (or even directly), regardless of which provider actually delivers those resources.*

4.3.1 Provider Access Point

Purpose: Consumer-facing entry point (API/portal/control plane) that exposes provider services and permitted other providers' services.

Responsibilities:

- Expose services to consumers (provider-local and permitted other providers' services).
- Enforce consumer scoping: show only what is necessary and only resources belonging to the consumer.
- Translate consumer intent into service ordering and lifecycle actions.
- Present portability and switching options and constraints in a consistent way.

Notes / Variants:

- May completely hide federation complexity from the consumer.
- May be implemented as an API gateway, developer portal, marketplace front-end for a specific provider, or an administrative control plane.
- May delegate ordering to a Federation Catalog checkout flow or may accept orders directly and trigger federation execution behind the scenes.

4.3.2 Federation Manager

Purpose: Standard component that enables communication and coordinated operations across environments via an East-West (E/W) federation interface.

Responsibilities:

- Maintain peer registry and federation context; support technical federation onboarding between participants.
- Expose and consume the E/W federation interface for interoperability between environments.
- Execute cross-domain onboarding/configuration required to realize orders (e.g., accounts/tenants, policy bindings, credentials, required integrations).
- Perform cross-domain lifecycle actions (allocate/scale/release) on resources and services through standardized abstractions.
- Support federated authentication/authorization for control-plane actions and preserve audit trails across domains.
- Provide accounting hooks and usage records where required (optional; commercial settlement may be handled elsewhere).

Notes / Variants:

- May be operated by a provider, a neutral intermediary, a consortium, or (optionally) by a consumer.

4.3.3 Federation Catalog

Purpose: Discovery and (optionally) ordering components that aggregates or brokers access to offerings from multiple providers.

Responsibilities:

- Enable publication, discovery, search, and comparison of offerings using standardized service descriptions.
- Support compatibility-aware discovery for composing multi-provider workloads.
- Support either or both a Service Marketplace approach, where providers publish pre-defined offers in a shared catalog, and an On-Demand Bundle approach, where customers request specific capability combinations and providers respond with tailored, composite offers.
- Optionally provide ordering/checkout capabilities, acting as a front-end for order initiation, while the actual provisioning and execution is performed by the relevant Provider Access Point(s) and/or Federation Manager(s).

Notes / Variants:

- May be operated by a provider, a consortium, or a neutral intermediary; the architecture does not require a single global catalog.
- If Catalog provides ordering, it should still allow provider-side enforcement of policies, consumer scoping, and execution responsibility.

4.3.4 Trust & Credential Services

Purpose: Provide cryptographically verifiable trust foundations and evidence across domains.

Responsibilities:

- Issue and verify organizational identities and attributes (e.g., certifications, location, compliance evidence).
- Enable mutual authentication between federation components and participants.
- Optionally support conformance attestations and certification artifacts.

Notes / Variants:

- May integrate external trust frameworks and credential systems.
- May be implemented centrally per federation or federated itself.

4.3.5 Policy, Compliance & Governance Services

Purpose: Support policy definition and evaluation for compliance, sovereignty, and operational constraints.

Responsibilities:

- Express and evaluate policies for eligibility, data residency, workload placement constraints, and access decisions.
- Support auditability by linking decisions to verifiable evidence.
- Support portability/switching declarations and process descriptions to reduce switching friction.

Notes / Variants:

- Links directly to components in the “Security and Compliance” domain.
- May be operated by providers, consumers, intermediaries, or consortia.
- May be realized via policy-as-code approaches or semantic policy models.

4.3.6 Interconnection & Transport

Purpose: Provide secure network connectivity for federation control-plane (and optionally data-plane) interactions.

Responsibilities:

- Secure transport between federation components (e.g., Federation Managers, Catalogs, Trust services).
- Support connectivity patterns needed by the workload (service-to-service connectivity) where applicable.
- Avoid exposing internal topology and configuration beyond what is necessary.

4.3.7 Cross-domain Operations & Observability

Purpose: Enable operational coordination across provider boundaries without sacrificing autonomy.

Responsibilities:

- Exchange events/notifications relevant to federated workloads and services.

- Support metrics and SLO monitoring, fault and incident coordination, and operational reporting.
- Provide minimal necessary information exchange aligned with governance and privacy constraints.

Notes / Variants:

- Links directly to components in the “Management” domain.

4.3.8 Metering, Accounting & Settlement Support

Purpose: Provide consistent usage records and settlement support for commercial models where needed.

Responsibilities:

- Collect and expose usage metering records in a standardized way.
- Provide accounting hooks for invoicing/settlement, cost allocation, and chargeback.
- Allow multiple commercial models (reseller, marketplace, direct billing, bundled offers) without requiring a single approach.

Notes / Variants:

- Links directly to components in the “Management” domain.

4.4 Operating variants

This section illustrates how the components and interfaces defined in Section 4.3 can come together in practice. It outlines operating model variants, focusing on who operates the key functions (discovery, ordering, and cross-domain execution) and how those responsibilities are distributed across providers, catalogs/marketplaces, and consumers. The presented variants are intended as illustrative examples; other variants may exist.

1) Provider-mediated federation: the consumer interacts only with a Provider Access Point, while the provider’s Federation Manager coordinates with other providers’ Federation Managers behind the scenes. The consumer can be oblivious of the federation.
2) Marketplace-mediated ordering: a Federation Catalog provides discovery and ordering, and hands off execution to Provider Access Points and Federation Managers.

5. Functional Interfaces

The functional interfaces used in actual implementations are presented in the **Figure 5.1**. These are high-level abstractions and represent essential interfaces in all possible architecture implementations. Lower-level interfaces, despite being present, are not reported in this view.

The interfaces in this view could/should be labeled as interfaces increasing European independence to different degrees. The potential identification of the interfaces that will enforce European independence could be a matter for further studies and improvements.

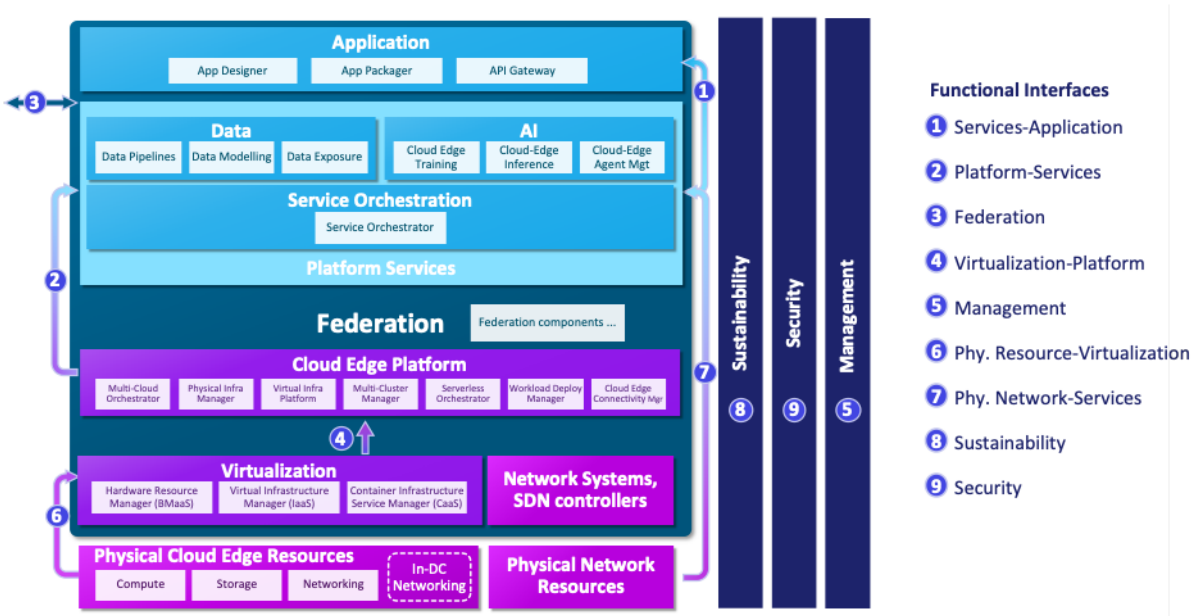

Figure 5.1: *Functional Interfaces.*

Based on the comprehensive analysis of partner responses to the functional interface definitions for the Reference Architecture, here is the synthesis of the nine interfaces that were selected among all possible interfaces in the Reference Architecture. For each of the interfaces, the key technologies on which they are based in current implementations are shown, with a synthesis to describe which are used in the projects, and a preliminary assessment of the strong points, where an alignment with state-of-the-art implementations is reached, and areas where further enhancements could be important in the near future.

5.1 Platform Services to Application Layer

Key Technologies:

- Kubernetes API (https://kubernetes.io/docs/reference/)
- Kubernetes Resource Model as used in NeoNephos Platform Mesh (https://platform-mesh.io/)
- REST APIs (https://restfulapi.net/)
- JSON(-LD) (https://www.json.org/json-en.html)
- Model Context Protocol (MCP, https://modelcontextprotocol.io/)

Synthesis: This interface enables the platform services layer to deploy and manage application components across the Cloud-Edge Continuum. It ensures lifecycle control, connectivity, and policy enforcement for applications, allowing seamless integration of workloads with orchestration workflows. The industry converges on the Kubernetes API as the dominant technology. Solutions promoted or developed as part of the IPCEI-CIS include NeoNephos Platform Mesh for service instance management, the CAMARA EAM API (https://github.com/camaraproject) for application lifecycle management, and TMForum Business API Ecosystem (https://github.com/FIWARE-TMForum/) for monetization. The emerging **Model Context Protocol (MCP)** can be used as an agent-facing interface for tool and context invocation, complementary to deployment and lifecycle APIs such as Kubernetes, CAMARA, and TMForum. The interface heavily relies on **RESTful APIs** with JSON(-LD) payloads and is consistently identified as a core interface that cannot be easily substituted.

The emphasis on this interface is primarily on:

- **Kubernetes API** dominance aligns with ETSI NFV Release 5+ evolution toward cloud-native orchestration
- **CAMARA APIs** for application lifecycle management reflects the Linux Foundation's major telco API standardization initiative
- **Model Context Protocol (MCP)** shows forward-thinking AI integration, consistent with emerging cloud-native requirements.

Missing considerations: Intent-based orchestration patterns like those in Nephio could strengthen this interface.

5.2 Cloud Edge Platform to Platform Services

Key Technologies:

- Kubernetes API (https://kubernetes.io/docs/reference/)
- OpenStack API (https://docs.openstack.org/api-ref/)
- OpenNebula API (https://docs.opennebula.io/7.0/product/integration_references/system_interfaces/)
- GitOps (https://opengitops.dev/)
- OpenTelemetry (https://opentelemetry.io/docs/)
- NeoNephos CobaltCore (https://cobaltcore-dev.github.io/docs/)
- NeoNephos IronCore (https://ironcore.dev)
- NeoNephos Gardener (https://gardener.cloud)
- NeoNephos Platform Mesh (https://platform-mesh.io)
- Sylva (https://sylvaproject.org)

Synthesis: This interface connects platform services layers with cloud edge platform services, translating high-level deployment intents into executable actions on virtualized and containerized environments. It provides observability and automation for resource allocation and service chaining. This interface represents the most mature and standardized layer, with general adoption of **Kubernetes API** as the core technology. The NeoNephos foundation (https://neonephos.org) provides comprehensive coverage through Platform Mesh, Gardener, CobaltCore (OpenStack), and

IronCore APIs. For deployment and operation, **GitOps-based service provisioning** and **OpenTelemetry for observability** can be used. The interface supports both traditional VM management through OpenStack/OpenNebula/CobaltCore and cloud-native workloads through Kubernetes/Gardener/IronCore/Sylva, making it the foundational layer for hybrid-cloud operations.

This interface represents the industry's best practice with:

- **Kubernetes API adoption** matches ETSI NFV's shift toward declarative, cloud-native MANO.
- **GitOps-based service provisioning** aligns with Nephio's configuration-as-data approach.
- **OpenTelemetry for observability** follows CNCF graduated project standards.
- Interfaces heavily rely on **RESTful APIs** with JSON payloads.

State of the art validation: ETSI NFV specifically calls for simplified, declarative orchestration interfaces exactly as the partners propose.

5.3 Federation (Multi-Cloud-Edge Coordination)

Key Technologies:

- Kubernetes API (https://kubernetes.io/docs/reference/)
- GSMA EWBI API (https://www.gsma.com/solutions-and-impact/gsma-open-gateway/)
- NeoNephos Platform Mesh (https://platform-mesh.io/)
- NeoNephos Katalis (https://github.com/NeoNephos-Katalis)
- ECOFED API

Synthesis: This interface supports interoperability between multiple providers, enabling resource sharing and workload migration across federated clouds. It ensures consistent APIs for cross-domain orchestration and facilitates collaborative use cases like distributed AI and data exchange. On the federation interface enabling multi-operator and multi-cloud coordination, the **GSMA EWBI API** provides a solution for secure resource sharing between operators. Other options include the NeoNephos Platform Mesh and **Kubernetes federation via Liqo** (https://liqo.io/docs/). Open-source projects like NeoNephos Katalis are potential references for federation in multi-cloud-edge environments.

Strong points:

- The definition of a federation interface helps greatly towards the implementation of a common European Multi-Provider Cloud-Edge Continuum.
- **GSMA EWBI API** for operator federation is one of the current industry standards and it is well present in IPCEI implementations.
- **Kubernetes federation via Liqo** represents a further multi-cloud orchestration option.
- Possible synergies and interactions with open-source projects on the topic.

Areas needing attention:

- Frameworks for European sovereignty to be further elaborated, considering all the potential technologies which will strengthen sovereignty through the federation environment

- Federated agent and tool discovery should be addressed, including trusted discovery of MCP endpoints and peer agents across providers while preserving data residency, endpoint identity, authorization, versioning, and audit requirements

5.4 Virtualization to Cloud Edge Platform

Key Technologies:

- Kubernetes API (https://kubernetes.io/docs/reference/)
- NeoNephos Gardener (https://gardener.cloud)
- NeoNephos CobaltCore (https://cobaltcore-dev.github.io/docs/)
- NeoNephos IronCore (https://ironcore.dev/)
- OpenStack API (https://docs.openstack.org/)
- OpenNebula API (https://docs.opennebula.io/7.0/product/integration_references/system_interfaces/)
- TOSCA (https://www.oasis-open.org/committees/tosca/)
- Cluster API (CAPI, https://cluster-api.sigs.k8s.io/)
- Sylva (https://sylvaproject.org/)

Synthesis: This interface bridges physical infrastructure and platform services by exposing virtualized resources through standardized APIs. It enables dynamic provisioning of VMs, containers, and clusters, ensuring portability and scalability across heterogeneous environments. It bridges physical infrastructure with platform services through Kubernetes Cluster API (CAPI), NeoNephos Gardener, NeoNephos CobaltCore, NeoNephos IronCore, OpenStack, and OpenNebula. A Sylva Cluster API is another alternative for Kubernetes cluster creation and management. TOSCA provides an Infrastructure-as-Code language for topology definition, providing expressiveness for translation into target orchestration languages.

This interface is considered to match adequately the current virtualization evolution:

- **Kubernetes Cluster API (CAPI)** is the de facto standard for cloud-native cluster lifecycle management
- **TOSCA 2.0 for IaC** remains relevant for telco workload description, though cloud-native extensions are needed
- **OpenStack, CobaltCore, and OpenNebula** integration interfaces reflect hybrid cloud realities in telco environments
- **IronCore** can be used for modern cloud-native environments
- Interfaces heavily rely on **RESTful APIs** with JSON payloads

Enhancement opportunity: From a telco perspective, integration with O-RAN cloud-native architecture could strengthen edge virtualization aspects.

5.5 Management

Key Technologies:

- Kubernetes API (https://kubernetes.io/docs/reference/)
- OpenTelemetry (https://opentelemetry.io/docs/)

- Agent Composition Platform
- AI/Agent Health and Tool Observability APIs for monitoring inference latency, resource utilization, service outages, model calls, tool calls, MCP/A2A interactions, handoffs, approvals, policy violations, token usage, and replicable traces for audit and evaluation

Synthesis: This interface provides monitoring, logging, and SLA enforcement for platform services. It integrates observability frameworks and automation tools to maintain performance, reliability, and compliance across distributed cloud-edge deployments. It focuses on comprehensive platform monitoring and management through **OpenTelemetry** for observability and logging. A partner provides App Deployment Monitoring APIs with Kafka Bus integration and OTEL collectors, emphasizing performance SLA compliance. Multiple partners contribute **Agent Composition Platform APIs** for low-code/no-code application development and monitoring. The interface includes specialized **AI Model Health APIs** for monitoring inference latency, resource utilization, and service outages in AI workloads. Sylva is mentioned by a partner too.

This interface aligns well with industry evolution:

- **OpenTelemetry-centric** observability follows CNCF best practices
- **AI Model Health APIs could** anticipate the AI-driven telco cloud future
- **Agent Composition Platform APIs** reflect the trend toward low-code/no-code telco service creation

State-of-the-art match: The multi-partner focus on AI monitoring aligns with emerging requirements in cloud-native operations.

5.6 Physical Cloud Edge Resources to Virtualization

Key Technologies:

- Redfish API (https://www.dmtf.org/standards/redfish)
- OpenStack API (https://docs.openstack.org/)
- NeoNephos IronCore Bare Metal API (https://ironcore.dev/baremetal/)
- libvirt (https://libvirt.org/)
- Ceph (https://docs.ceph.com/en/latest/)
- WASM (https://webassembly.org/)

Synthesis: This interface abstracts hardware resources into virtualized entities using standard protocols. It manages computing, storage, and networking allocation, ensuring efficient utilization and enabling advanced features like bare-metal provisioning and hardware-specific optimizations. It handles the lowest-level infrastructure abstraction using industry standards where possible. **Redfish API** is a common API for hardware management, with proprietary drivers for GPUs and specialized networking hardware. The interface utilizes **libvirt and Ceph** for virtualization and storage management. A partner adds **WASM resource management capabilities**, directly interfacing with physical resources for containerized workload optimization, considering also Cubbit. NeoNephos Bare Metal API leverages the Redfish protocol for full automation based on Kubernetes mechanisms.

Industry-standard approach with:

- **Redfish API** for hardware abstraction follows DMTF standards
- **libvirt and Ceph** represent mature open-source virtualization stacks
- **WASM resource management** shows innovation in edge computing optimization

Potential steps forward: From a telco perspective, **O-RAN specifications** for RAN-specific hardware abstraction could be a valuable addition.

5.7 Physical Network Resources to Platform Services

Key Technologies:

- 3GPP 5GS (https://www.3gpp.org/specifications)
- 3GPP LTE (https://www.3gpp.org/specifications)
- GSMA OP (https://www.gsma.com/solutions-and-impact/technologies/networks/operator-platform-hp/)
- REST APIs (https://restfulapi.net/)
- JSON (https://www.json.org/json-en.html)
- IPX

Synthesis: This interface connects the Service Orchestrator with the specific telco network capabilities with the purpose of collecting user equipment location information or the serving gateway, while also performing actions from the SO. The concept of edge is closely related to the proximity of the edge server to the subscriber, furthermore when the terminal connected to the edge is moving across the telco network, the optimal edge server may change and consequently the telco gateway that supports the telco service. GSMA standards support this interface called Operator Platform's SBI-NR based on 3GPP specifications and elements as NEF/SCEF or NWDAF. Partners are working on a network integration model with the goal of end-to-end SLA guarantee in all the edge scenarios. The key technologies are based on 3GPP specifications (architecture, elements involved, services, *Multi-Access Edge Computing* (MEC), procedures, policy and charging, specialized mediation elements, APIs) and GSMA architecture and APIs, also ETSI ISG MEC is working on edge.

This interface is being standardized and developed in:

- **ETSI:** Edge concept was introduced by ETSI and then developed by 3GPP in 2014, now this concept is renamed to MEC, https://www.etsi.org/technologies/multi-access-edge-computing). The most important ETSI specifications are ETSI GS MEC 002 MEC requirements and ETSI GR MEC 031 guidance on MEC integration with 3GPP 5G systems.
- **3GPP:** 4G and 5G architecture are entirely described in the TS documents with all the procedures for establishing communication and its lifecycle management. MEC is standardized in TS 23.558. The architecture where MEC is included is described in TS 23.501 and procedures in TS 23.502, policy and charging control in TS 23.503. NEF is the 5G function which can expose the network capabilities to 3rd parties, NBI specified in TS 29.522. NWDAF is also important to collect statistical data and network information specified in TS 29.520.
- **GSMA PRDs:** develop the concept of edge and introduce the Service Orchestrator called Operator Platform (https://www.gsma.com/solutions-and-impact/technologies/networks/gsma_resources/gsma-operator-platform-group-july-2025-

specifications/). There are two distinct groups: OPG for OP and EPG for edge. The OP architecture and SBI-NR are defined in GSMA: OPG.02 defines the OP architecture where SBI-NR is an essential part, OPG.11 is about edge services requirements and OPG.03 describes SBI-NR APIs.

Strong aspects: This set of standards and best practices becomes the basis for integrating edge into the telco network.

Missing considerations: Network integration depends on the maturity of the technologies. There are important open issues about mobility, UPF/SMF reselection and collecting end-to-end delay information or transport mechanisms to provide the optimal path. There are distinct groups working on these tasks.

5.8 Sustainability (Environmental Optimization)

Key Technologies:

- OpenLCA (https://www.openlca.org/)
- Carbon Footprint Assessment
- Cloud-Edge Platforms
- Kubernetes APIs
- Telemetry APIs
- Orchestration tools

Synthesis: This interface introduces mechanisms for energy-aware orchestration and carbon footprint monitoring. It enables workload placement strategies that optimize resource consumption for sustainable cloud-edge operations. It addresses environmental impact through **energy consumption monitoring** and **carbon footprint calculation**. For example, outcomes are foreseen to be generated with OpenLCA, a tool for lifecycle assessment and resource consumption tracking (water, energy, CO2). Another option is doing ex-ante carbon footprint evaluation and runtime data acquisition for energy-saving policies. The functional interface will also support operational optimization of (federated and secure) digital services infrastructure by providing incentives, doing optimization, and providing reporting.

The following key technologies are currently being involved, among others:

- **Platform APIs** to minimize redundant data creation and transfer, promoting optimized application deployment for sustainability benefits. Standards are still being defined, while appropriate metering and monitoring solutions are still being explored. Among others, Sylva, EFN, OpenNebula, and Camara are being considered.
- **Telemetry APIs** enables standardized energy measurement collection, aggregation, and reporting. Examples include many open-source technologies such as Prometheus, Kepler, Alumet, but also SNMP, Redfish, and IPMI, and others.
- **Kubernetes APIs** allow applications and/or platform controllers to label resources and schedule workloads across clusters according to sustainability parameters. Examples of relevant technologies here are *Kubernetes Control Plane* (KCP), IronCore, Gardener, and *Kubernetes Resource Model* (KRM), among others.

- **Orchestration APIs** enables sustainability management by linking services and resources and automating workflows for administrative processes and supervising service-impacting changes. Examples of open-source technologies that provide interfaces in this domain are OpenNebula, Surf Workflow Orchestrator, Netbox, and others.

5.9 Security

This interface enforces security and compliance policies across all layers, integrating identity management, encryption, and threat detection into operational workflows. It ensures consistent application of security controls during orchestration and runtime. The operational execution of security and compliance, as described in the CSF[7], will be further detailed following the progress in the IPCEI-CIS project.

[7] https://www.8ra.com/wp-content/uploads/IPCEI-CSF_2026.pdf

6. Reference Architecture Roles

The Reference Architecture is designed to support unified pan-European cloud-edge infrastructure and services capable of executing diverse use cases by using different, yet compatible and interoperable software stacks. This flexibility is essential for accommodating the varied requirements of different sectors and applications across the IPCEI-CIS project, ensuring that the infrastructure can adapt to a wide range of operational demands while maintaining a consistent and integrated framework. The openness and standardization of interfaces facilitate the necessary portability and scale in a multi-provider environment.

This model is **technology-agnostic** and allows the deployment, when required, of customized software stacks for specific use cases.

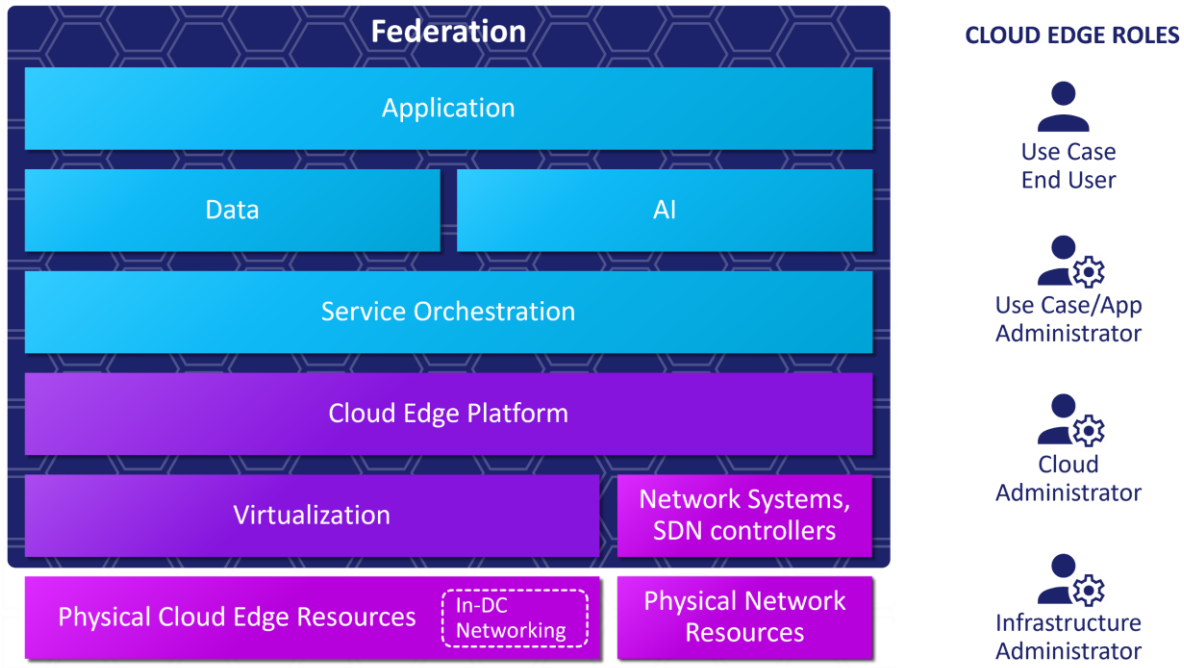

Figure 6.1: *Different roles in the ICRA.*

The different roles in this architecture model are described in **Figure 6.1**. The infrastructure resources are managed by **Infrastructure Administrators**, ensuring their availability through well-defined APIs to manage bare metal resources through the Physical Infrastructure Manager (in a multi-provider scenario), or directly through specific hardware resource managers.

The **Cloud-Edge Administrator** is responsible for the management of the container, serverless and virtual infrastructure resources, interacting with the corresponding orchestrators (VIP, MCM, serverless orchestrator) or directly with the specific managers (VIM, CISM, etc.) in the case of single-provider scenarios.

The **Use Case Administrator** works with the Application Layer components to design and package the use case application to prepare it for its deployment, defining the state it must reach for a proper execution (resource model). The Use Case Administrators hand this resource model over to the Service Orchestrator for its deployment and lifecycle management. By using the cloud-edge

platform layer components, it deploys the cloud-edge runtime environment with the necessary software stack to reach the desired state. This runtime environment is tailored to meet the specific infrastructure and functionality needs of the use case, providing the necessary resources for the execution of the use case applications. The automation of this deployment process ensures that the environment is rapidly configured and aligned with the predefined requirements, reducing the time and effort needed to set up the infrastructure.

Once the cloud-edge runtime environment is deployed, the Use Case Administrator manages the operation of the environment through the orchestrators, that provide the flexibility to handle workloads efficiently, whether they are running in containers or on virtual machines.

Finally, the **End Users** of the specific **Use Case** are responsible for developing and deploying the workloads within this environment, ensuring that the applications and services required for the use case are designed and packaged properly.

7. Summary and next steps

This document describes the Reference Architecture, which serves as the common framework for the IPCEI-CIS integrated project. It is used to allow the description and comparison of solutions, workstream activities, integration clusters, and pilots run in the project, and to facilitate the integration of the diverse technological components the IPCEI-CIS is generating.

This version of the ICRA describes functional components, organized into layers to help clarify the complexity of systems needed to support the European Multi-Provider Cloud-Edge Continuum. However, it is not intended as an implementation guideline. Compatibility, interoperability, and portability among different IPCEI-CIS implementations are ensured through standard open interfaces that provide access to various services and enable service federation.

Future incremental versions will include additional details and use cases. A comprehensive definition of ICRA component readiness and a collection of real components produced in the IPCEI-CIS project will also be addressed in subsequent document versions. Recent developments in AI, security, and other fast-moving topics and their implications on architecture will continue to be added into the relevant sections. For practical purposes, this document includes only selected examples of domain components per layer, with more detailed information available through documents produced by specific IPCEI-CIS workstreams.

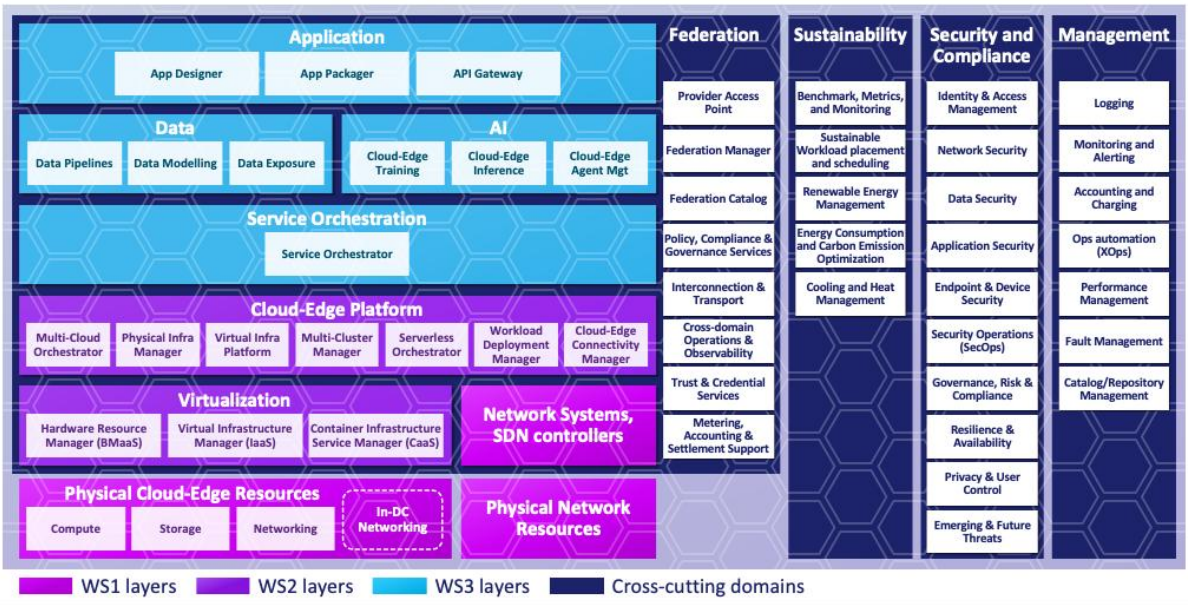

Figure 7.1: *Layers, Domains and Components at the ICRA.*

A Terminology

This appendix provides definitions for terminology used in this IPCEI-CIS Reference Architecture document. The terms below are intended to support consistent interpretation across partners, workstreams, pilots, and future blueprint implementations.

Term	Definition
A2A (Agent-to-Agent)	A pattern and emerging set of interfaces enabling AI agents to communicate, delegate, and coordinate with each other across systems and providers.
Agent	An autonomous or semi-autonomous AI system that perceives context, plans actions, and executes them by invoking tools, models, or other agents, typically operating within defined policies and oversight mechanisms.
AI Layer	The layer that supports AI lifecycle capabilities such as training, inference, explainability, model cataloging, and agent-oriented execution across cloud and edge environments.
Application Layer	The layer that supports designing, packaging, exposing, monitoring, and accounting for end-user applications and functions.
Blueprint	A sector-specific or use-case-specific instantiation of the reference architecture, typically created by tailoring or streamlining the broader framework.
Cloud-Edge Administrator	A role responsible for managing virtualized, containerized, and serverless cloud-edge resources through the relevant infrastructure managers and orchestrators.
Cloud-Edge Connectivity Manager (CEC)	A component that establishes and manages connectivity between workloads, services, users, and sites across data center and wide-area network domains.
Cloud-Edge Continuum	A distributed computing environment spanning cloud and edge infrastructures, enabling data processing and service execution across geographically and operationally diverse resources.
Cloud-Edge Platform Layer	The layer that manages runtime environments, platform services, workload placement, connectivity, and lifecycle operations over the virtualized continuum.
Collaborative Security Framework (CSF)	The project security framework is used to classify, align, and integrate security capabilities and controls contributed by partners across reference architecture.
Composable	Capable of being combined with other services, capabilities, or resources in a modular way to create higher-level solutions.
Data Layer	This layer provides capabilities for data collection, processing, cataloging, policy control, exposure, and exchange across the continuum.

Domain	A cross-cutting architectural concern that applies across multiple layers, such as management, security and compliance, federation, or sustainability.
End User	The consumer of a deployed application or service within a use case, operating within the environment prepared following the reference architecture.
EWBI	East-West-Bound Interface; an interoperability interface exposed by federation components to support communication between federated environments.
Federation	The structured collaboration between providers or environments, which enables interoperability, resource sharing, service composition, and workload portability while preserving autonomy and governance.
Federation Catalog	A federation component that aggregates or exposes offerings from multiple providers, for example through marketplace-style discovery or on-demand bundle creation.
Federation Manager	A federation component that enables inter-provider communication, negotiation, onboarding, runtime coordination, and access to federated resources and services.
IPCEI-CIS	Important Project of Common European Interest on Next Generation Cloud Infrastructure and Services, within which the reference architecture is defined.
IPCEI-CIS Reference Architecture (ICRA)	The common technology-agnostic architectural framework used to describe, position, and integrate functional components, roles, interfaces, and pilot-related implementations in the IPCEI-CIS project.
Identity and Access Management (IAM)	The set of capabilities used to establish identities, authenticate entities, and authorize access to resources and services.
Infrastructure Administrator	A role responsible for managing physical infrastructure resources and ensuring their availability through appropriate management interfaces.
Interoperability	The ability of components, providers, or systems to work together through compatible interfaces, data models, and operational semantics.
Layer	A horizontal architectural level that groups functionality by abstraction level, from applications down to physical resources.
Management Domain	The domain that provides operational capabilities such as logging, monitoring, alerting, accounting, charging, performance management, inventory, and automation.
Model Context Protocol (MCP)	An emerging agent-facing interface mentioned in the document for governed access to tools, context, resources, and external systems in AI and agent-based execution scenarios.

Multi-Cloud Orchestrator (MCO)	A platform-layer component that interprets desired application state and coordinates deployment, scaling, updating, and removal of workloads and supporting platform services across multiple sites or providers.
Multi-Cluster Manager (MCM)	A component that creates, configures, and manages container clusters across multiple providers or technologies through a unified interface.
Multi-Provider Cloud-Edge Continuum	A cloud-edge continuum operated across multiple independent providers, with emphasis on interoperability, portability, composability, and coordinated service delivery.
Observability	The ability to understand system behavior and health through telemetry such as logs, metrics, traces, alerts, and related operational signals.
Physical Infrastructure Manager (PIM)	A component that provisions, inventories, and manages physical resources such as servers, storage, and networking for allocation into higher layers.
Physical Layer	The layer comprising the underlying hardware resources, such as compute, storage, and networking, that support the continuum.
Pilot Scenario	A proof-of-concept or early implementation used to validate, refine, and extend the reference architecture in practical settings.
Portability	The ability to move workloads, services, or data between providers or environments with limited rework or dependency on one specific platform.
RAG (Retrieval-Augmented-Generation)	A technique in which a generative AI model retrieves relevant information from an external knowledge source at inference time and incorporates it into its response, improving factual grounding and enabling use of data outside the model's training set.
Reference Architecture	A conceptual architectural model that defines structure, responsibilities, interfaces, and relationships without prescribing one specific implementation.
Security and Compliance Domain	The domain that provides controls and capabilities for identity, access, data protection, threat management, governance, resilience, privacy, and regulatory alignment.
Service Function Chain	A defined sequence of connected application or service components that together deliver a required service behavior.
Service Level Agreement (SLA)	A defined commitment regarding service quality, availability, performance, or support that is monitored and managed operationally.
Service Orchestration Layer	The layer responsible for coordinating deployment, lifecycle, and operational behavior of services and applications across distributed environments.
Sustainability Domain	The domain that focuses on energy efficiency, carbon reduction, renewable energy usage, sustainable workload placement, and environmental monitoring and optimization.

Technology-agnostic	Describing an architectural approach that defines required functions and relationships without mandating one vendor, stack, or implementation technology.
Use Case Administrator	A role responsible for preparing use case applications for deployment, defining required state, and managing the resulting runtime environment through orchestration capabilities.
Virtual Infrastructure Platform Manager (VIP)	A component that creates and manages virtual machine clusters and related virtual infrastructure resources across locations.
Virtualization Layer	The layer that abstracts physical resources into virtualized or containerized resources and exposes them to dynamic allocation and management.
Workload Deployment Manager (WDM)	A component that deploys software packages or workload definitions onto clusters or virtual environments in response to orchestration requests.