A Cognitive Serverless Framework for the Cloud-Edge Continuum

D2.4 COGNIT Framework - Architecture - d

Version 1.0

7 November 2024

Abstract

COGNIT is an AI-Enabled Adaptive Serverless Framework for the Cognitive Cloud-Edge Continuum that enables the seamless, transparent, and trustworthy integration of data processing resources from providers and on-premises data centers in the cloud-edge continuum, and their automatic and intelligent adaptation to optimise where and how data is processed according to application requirements, changes in application demands and behaviour, and the operation of the infrastructure in terms of the main environmental sustainability metrics. The aim of this version of the COGNIT Framework Architecture report is to provide an overview of the Project's overall development status, offer a summary of the work done in the Third Research & Innovation Cycle (M15-M21), and identify the priorities for the Fourth Research & Innovation Cycle (M21-M27).

This project is funded by the European Union's Horizon Europe research and innovation programme under Grant Agreement 101092711 – SovereignEdge.Cognit

Deliverable Metadata

Project Title:	A Cognitive Serverless Framework for the Cloud-Edge Continuum
Project Acronym:	SovereignEdge.Cognit
Call:	HORIZON-CL4-2022-DATA-01-02
Grant Agreement:	101092711
WP number and Title:	WP2. Adaptive Cloud-Edge Serverless Framework Architecture
Nature:	R: Report
Dissemination Level:	PU: Public
Version:	1.0
Contractual Date of Delivery:	30/09/2024
Actual Date of Delivery:	07/11/2024
Lead Author:	Marco Mancini (OpenNebula) & Constantino Vázquez (OpenNebula)
Authors:	Monowar Bhuyan (UMU), Dominik Bocheński (Atende), Aritz Brosa (Ikerlan), Malik Bouhou (CETIC), Idoia de la Iglesia (Ikerlan), Christophe Ponsart (CETIC), Jean Lazarou(CETIC), Agnieszka Frąc (Atende), Grzegorz Gil (Atende), Torsten Hallmann (SUSE), Mateusz Kobak (Phoenix), Tomasz Korniluk (Phoenix), Johan Kristiansson (RISE), Antonio Lalaguna (ACISA), Juan Jose Ruiz (ACISA), Carlos Lopez (ACISA),Javad Forough (UMU), Yashwant Singh Patel (UMU), Mikel Irazola (Ikerlan), Alberto P. Martí (OpenNebula), Philippe Massonet (CETIC), Nikolaos Matskanis (CETIC), , Daniel Olsson (RISE), Álvaro Puente (Ikerlan), Holger Pfister (SUSE), Tomasz Piasecki (Atende), Francesco Renzi (Nature4.0), Kaja Swat (Phoenix), Paul Townend (UMU), Iván Valdés (Ikerlan), Thomas Ohlson Timoudas (RISE), Riccardo Valentini (Nature4.0), Ignacio M. Llorente (OpenNebula), Mirko Stojiljkovic (OpenNebula).
Status:	Submitted

Document History

Version	Issue Date	Status[1]	Content and changes
0.1	28/10/2024	Draft	Initial Draft
0.2	31/10/2024	Draft	Second Draft
0.2	06/11/2024	Peer-Reviewed	Reviewed Draft
1.0	07/11/2024	Submitted	Final Version

Peer Review History

Version	Peer Review Date	Reviewed By
0.1	30/10/2024	Johan Kristiansson (RISE)
0.1	30/10/2024	Antonio Álvarez (OpenNebula)
0.2	05/11/2024	Thomas Ohlson Timoudas (RISE), Joel Höglund (RISE)

Summary of Changes from Previous Versions

First Version of Deliverable D2.4

[1] A deliverable can be in one of these stages: Draft, Peer-Reviewed, Submitted, and Approved.

Executive Summary

Deliverable D2.4, released at the end of the Third Research & Innovation Cycle (M21), is a self-contained version of the COGNIT Framework Architecture report in WP2 “Adaptive Cloud-Edge Serverless Framework Architecture”. This report provides an overview of the new COGNIT Framework architecture, the Project's overall development status and offers a summary of the work done in the Third Research & Innovation Cycle (M16-M21).

In Deliverable D2.3, we have reported some limitations of the previous architecture. In particular, the main limitation is related to the management of the Serverless Runtime by the Device itself, an approach that has consequences in the optimization and orchestration of resources along the cloud-edge continuum. In the Third Research & Innovation Cycle (M16-M21), the Project has focused on the design of an improved COGNIT Architecture and the definition of a series of new components in order to address the limitations mentioned above. Therefore, this deliverable describes the design of the improved architecture, the specification, implementation and deployment of all necessary new components, and the update of the specifications, implementation and deployment of the existing components according to the updated Software Requirements.

In particular, the main components of the new COGNIT Architecture are the following:

- **Device Client.** It allows to offload function executions on the cloud-edge continuum using a Serverless paradigm and then perform different tasks such as function execution, transfer data from external sources and upload data from the device itself.
- **COGNIT Frontend.** It is the main entry point for the COGNIT service. It authenticates the device client that would like to use the service and returns a token that can be used by the Device for interacting with the COGNIT components. It provides different endpoints for uploading application requirements, storing functions and data, and it returns information about the most suitable resources where the device can offload the execution of the functions.
- **Edge Cluster.** It is a semi-autonomous complete environment with a dedicated software-defined resource set providing an abstraction layer enabling interoperability across the multi-provider cloud-edge infrastructure. Edge Clusters allow the execution of functions that are offloaded by the devices within a hardened and secure environment called **Serverless Runtime**.
- **Cloud-Edge Manager**. It is responsible for managing the highly distributed and heterogeneous resources of the cloud-edge continuum and the lifecycle of Serverless Runtimes within each Edge Cluster for the execution of the functions offloaded by the devices.
- **AI-Enabled Orchestrator**. It manages and optimises the lifecycle of Edge Clusters and the Serverless Runtimes within each Edge Clusters according to the application requirements, infrastructure and virtual resource usage, and energy-aware policies.

In connection with those components, the Project has delivered progress specifically in those software requirements needed to achieve the release 2.0 of the COGNIT Framework.

This document includes also a description of the methodology and the specific verification scenarios related to the new architecture to be used for the validation of the project's features and their applicability in each Use Case. This analysis represents the starting point for Work Packages WP3 and WP4, which will specify, design, and develop the components of the COGNIT Framework in order to satisfy the list of global and user requirements defined in this document. The present report also reports the overall development status of the project and a list of research and development priorities for the Fourth Research & Innovation Cycle (M21-M27).

Apart from the overview provided in this report, specific research and development activities performed in WP3 "Distributed FaaS Model for Edge Application Development" (related to the Device Client, the Edge Cluster Frontend, the Serverless Runtime, the COGNIT Frontend, and the Secure and Trusted Execution of Computing Environments) are described in detail in reports D3.3 "COGNIT FaaS Model - Scientific Report" and D3.8 "COGNIT FaaS Model - Software Source", whereas those performed in WP4 "AI-Enabled Distributed Serverless Platform and Workload Orchestration" (related to the Cloud-Edge Manager, the AI-Enabled Orchestrator, and the Energy Efficiency Optimization in the Multi-Provider Cloud-Edge Continuum) are described in reports D4.3 "COGNIT Serverless Platform - Scientific Report" and D4.8 "COGNIT Serverless Platform - Software Source"; details about the COGNIT software integration and the verification scenarios of each software requirements can be found in report D5.4 "Use Cases - Scientific Report".

This cycle has also witnessed the updated release of the **OpsForge** tool which allows the automated deployment of the release 2.0 of the COGNIT Platform in a target infrastructure, being a public cloud provider such as AWS or an on-premise datacenter.

This deliverable has been released at the end of the Third Research & Innovation Cycle (M21), and will be updated with incremental releases at the end of each research and innovation cycle (i.e. M27, and M33).

Table of Contents

Abbreviations and Acronyms

ACL	Access Control List
AI	Artificial Intelligence
API	Application Programming Interface
AUROC	Area Under ROC curve
AWS	Amazon Web Services
CC	Confidential Computing
CCA	Confidential Compute Architecture
CD	Continuous Delivery (Deployment)
CRA	Cyber Resilience Act
CVM	Confidential Virtual Machine
DaaS	Data as a Service
FaaS	Function as a Service
FFD	First-Fit Decreasing
IAM	Identity and Access Management system
IP	Internet Protocol
JSON	Javascript Object Notation
JWT	JSON Web Token
LDAP	Lightweight Directory Access Protocol
ML	Machine Learning
NIST	National Institute of Standards and Technology
NSGA	Non-Dominated Sorting Genetic Algorithm
OBS	Open Build Service
OIDC	Open ID Connect
RBAC	Role Based Access Control
REST	Representational State Transfer
ROC	Receiver Operating Characteristic
SDK	Software Development Kit
SEV	Secured Encrypted Virtualization
SGX	Software Guard eXtensions
TEE	Trusted Execution Environment
TDX	Trusted Domain eXtensions
VM	Virtual Machine
YAML	Yet Another Markup Language

1. Introduction

The initial version of the COGNIT Framework Architecture report (Deliverable D2.1), released in M3, included a summary of Use Case requirements, an analysis of sovereignty, sustainability, interoperability, and security requirements, the design and architecture specifications of the COGNIT Framework, and the methodology for verification. The incremental versions of Deliverable D2.2 and D2.3 provided an overview of the Project's overall development status, offered a summary of the work done in the First and Second Research & Innovation Cycle (M4-M9, M10-M15), and identified the priorities for the Third Research & Innovation Cycle (M16-M21). This self-contained version (Deliverable D2.4) is to provide a detailed description of the new version of the COGNIT Architecture and the software requirements, an overview of the Project's overall development status, offer a summary of the work done in the Third Research & Innovation Cycle (M16-M21), and identify the priorities for the Fourth Research & Innovation Cycle (M22-M27). An incremental version of this report will be released at the end of next research and innovation cycles (i.e. M27 and M33).

D2.4 is a living document that is composed of an introductory section and fourteen additional sections organised in four main blocks of content:

- **Part I** focuses on the definition of global requirements for the COGNIT Framework, as well as on the user requirements extracted from the project's Use Cases. Section 2 identifies the global sovereignty, sustainability, interoperability, and security requirements. Section 3, on the other hand, summarises the main user requirements of the Use Cases.
- **Part II** focuses on the definition of the software requirements of the COGNIT Framework. Section 4 defines the overall architecture of the COGNIT Framework, consisting of two main parts: the *Distributed Serverless Model for Edge Application Development* and the *Cognitive Cloud-Edge Module*, which are described in Section 5 and Section 6, respectively. Section 7 introduces and defines advanced mechanisms and techniques to build a secure and trusted execution environment in the cloud-edge continuum. Section 8 presents the functional gaps and software requirements grouped into the main building components of COGNIT. Section 9 provides the matching between software and user requirements.
- **Part III** focuses on verification and implementation. Section 10 presents the verification methodology and a list of verification scenarios. Section 11 identifies an initial set of technologies to be used in the instantiation of COGNIT Framework.
- **Part IV** reports on the overall progress, addressing in section 12 the progress in the implementation of the software requirements, as well as the status of the different KPIs of the project; in section 13 it reports the work carried out in the Third Research and Innovation Cycle with a breakdown that provides detailed information per component; while in section 14 it outlines the priorities for the starting Fourth Research Cycle (M22-M27).

The document ends with a conclusion section (Section 15).

PART I. Global and User Requirements

2. Sovereignty, Sustainability, Interoperability, and Security Requirements

This section summarises the results of an initial analysis of the European context in order to ensure that, by meeting a number of relevant, transversal sovereignty, sustainability, interoperability, and security requirements, the architecture of the COGNIT Framework is in line with EU digital policies and strategic priorities related to the Cognitive Cloud. This requirement analysis, which also incorporates sector-specific priorities from some of the Project's Use Cases, will be updated as part of each release of the Architecture report.

2.1. Sovereignty Requirements

Sovereignty, in the context of the development of a European cloud-edge continuum, involves the consolidation of the leadership and strategic autonomy of the EU in the digital world. The EU's New Industrial Strategy, for instance, links that global objective with the need to build solutions capable of leveraging the deployment of 5G and edge infrastructures, as well as competitive European alternatives for the multi-cloud, following an open source model.

It is also worth noting that the European Commission's Open Source Software Strategy 2020-2023 also states that *"open source impacts the digital autonomy of Europe. Against the hyperscalers in the cloud, it is likely that open source can give Europe a chance to create and maintain its own, independent digital approach and stay in control of its processes, its information and its technology"*. With all those priorities in mind, Table 2.1 below provides the list of sovereignty requirements to be met during the execution of the project:

Id	Description	Source
SOR0.1	The COGNIT Framework shall be able to leverage public, private, and self-hosted cloud and edge infrastructures hosted in the European Union.	All
SOR0.2	The implementation of the COGNIT Architecture shall maximise the use of European open source technologies and frameworks.	All
SOR0.3	The COGNIT Framework shall provide an abstraction layer that ensures workload portability seamlessly across different infrastructure providers.	All
SOR0.4	Data handling by the COGNIT Framework shall be compliant with the GDPR.	All

Table 2.1. Global Sovereignty Requirements.

2.2. Sustainability Requirements

These global requirements aim at reducing the ecological footprint of the COGNIT Framework (reported in Table 2.2) in line with the European Green Deal's objective to create a climate neutral continent and assuming that, in line with the Digital Decade objectives, our solution will have to be able to leverage at some point the *"10,000 climate neutral highly secure edge nodes"* that are expected to be deployed throughout the EU by 2023. Sustainability is the focus of the Project's Research Challenge 6—*"Optimization of energy efficiency and adaptation to variable (local) green energy supply throughout the cloud-edge continuum"* stated in the Grant Agreement; this challenge will be studied in particular as part of the Energy Use Case (UC3), where the COGNIT Framework will have to leverage AI/ML techniques to reduce energy usage in a household, consequently reducing its energy footprint.

Id	Description	Source
SUR0.1	Sustainability performance needs to be measurable (e.g. energy profiles should be queryable and updatable for every feature/component within the framework), including energy sources (e.g. renewable, non-renewable) and energy consumption profiles (e.g. estimated power consumption).	UC3
SUR0.2	Sustainability needs to be maximised to reduce environmental footprint by leveraging edge characteristics (e.g. by increasing the share of renewables, minimising battery use/size, using energy otherwise wasted, or scaling down active Runtimes).	UC3
SUR0.3	The whole energy lifecycle should be taken into account in order to implement a circular economy, including e.g. energy availability and cost and hardware degradation.	All

Table 2.2. Global Sustainability Requirements.

2.3. Interoperability Requirements

The EU's New Industrial Strategy also identifies the need to achieve a fairer and more competitive business environment between business users and cloud providers, enhancing access to fair, competitive, and trustworthy cloud solutions. Those objectives are also in line with the technological priorities being defined as part of the updated EU industry roadmap by the European Alliance for Industrial Data, Edge and Cloud. Interoperability is also the focus of the Project's Research Challenge 4—*"Portable and adaptive execution of serverless workloads across a multi-provider cloud-edge continuum"* stated in the Grant Agreement.

The interoperability requirements, reported in Table 2.3, target the ability of the COGNIT Framework to connect and integrate with existing infrastructures, services, hardware. This means that the framework should be aware of and use existing frameworks, technologies and standards, generic enough to be deployed on common infrastructures, and its usage instructions should be properly documented:

Id	Description	Source
IR0.1	Deployment of the COGNIT Framework and of its components should be as portable as possible across heterogeneous infrastructures or cloud/edge service providers (e.g. by using broadly-adopted virtualisation and container technologies).	All
IR0.2	Preference should be given to expanding existing frameworks, tools, and open standards.	All
IR0.3	The interfaces of the COGNIT Framework shall be documented in order to facilitate discovery of its features by third-parties.	All

Table 2.3. Global Interoperability Requirements.

2.4. Security Requirements

Security requirements (reported in Table 2.4) cover the ability of the COGNIT Framework to protect data and services at rest and in transit, in particular regarding confidentiality, integrity, and availability (CIA). In line with the recommendations defined by the Cybersecurity Research Directions for the EU's Digital Strategic Autonomy, they follow these three dimensions: protect the personal data from Internet giants, ensure resilience, and retain the ability to make informed and independent decisions. The COGNIT Framework should implement a security-by-design approach by adopting a number of good practices, including a risk-based approach based on the NIST CyberSecurity framework that structures security controls (identify, detect, protect, respond, recover), and a secure software lifecycle management based on DevSecOps (see Section 10).

In addition, special care will be taken to adapt these high level security requirements to make sure that they are aligned with the latest EU cybersecurity priorities, such as the NIS2 Directive and the future CRA. Security is the focus of the Project's Research Challenge 7—*"Secure and trusted execution of computing environments on multi-provider cloud-edge continuum"* stated in the Grant Agreement; and will be expanded through the CyberSecurity Use Case by incorporating advanced anomaly detection mechanisms, privacy respecting techniques, and security remediation orchestration:

Id	Description	Source
SER0.1	Communications inside COGNIT, and between the COGNIT environment and the outside (e.g. IoT devices) should be encrypted and signed using security mechanisms such as SSLv3.	All
SER0.2	The COGNIT Framework should be built following security-by-design and Zero Trust practices.	UC4
SER0.3	The implementation of the COGNIT Framework should be aligned with the latest legislative frameworks, such as the NIS2 Directive, the GDPR, and the future Cyber Resilience Act (CRA).	All
SER0.4	Runtimes should be protected against threats by the enforcement of security controls such as secure defaults, vulnerability scans, intrusion and anomaly detection and continuous security assessment (the specific controls to be implemented will be determined by a risk analysis).	All
SER0.5	Resources should be protected by an Identity and Access Management (IAM) system, implementing role based access control (RBAC), security zones, and support for a multi-tenant security model.	All
SER0.6	Integrity of the offloaded functions needs to be guaranteed, including the function inputs and outputs	All

Table 2.4. Global Security Requirements.

3. Use Case Requirements

This section summarises the user requirements extracted from the Use Cases:

Id	Description	Source
UR0.1	Device applications should be able to offload any function written in C or Python languages.	All
UR0.2	Device applications should be able to upload data from the device ensuring data locality with respect to where the offloaded function is executed.	All
UR0.3	Device applications should be able to upload data from external backend storages ensuring data locality with respect to where the offloaded function is executed.	All
UR0.4	Execution of functions such as ML inference engines should be able to load machine learning models stored ensuring data locality with respect to where the function is executed.	All
UR0.5	Function execution can be executed in different tiers of the Cloud-Edge continuum according to network latency requirements.	All
UR0.6	Device application shall have the ability to define maximum execution time of the offloaded function upon offloading.	All
UR0.7	Device application shall have the ability to specify and enforce runtime maximum provisioning time and runtime shall be provisioned within the previously specified time.	All
UR0.8	Device applications must be able to request and obtain authorization prior to establishing any further interaction with COGNIT.	All
UR0.9	IAM system integration for high granularity authentication and user management for device clients and the COGNIT Framework.	All
UR0.10	Push mechanism to inform about status/events from the COGNIT Framework back to the requestor device client.	All

Table 3.1. Common user requirements.

Id	Description	Source
UR1.1	Function execution shall be supported in shared, multi-provider environments (with different access and authorization procedures), and the execution must be isolated from other processes on the host system.	UC1
UR1.2	Device applications shall be able to use scalable resources for offloading function execution to maximise exploitation of resources in shared environments, while avoiding saturation or resource kidnapping.	UC1
UR1.3	Function execution should exploit data locality and prioritise edge nodes where the required data is already stored.	UC1
UR1.4	The whole life cycle of either function execution or code offloading should be auditable and non repudiable.	UC1
UR1.5	Device applications should be able to request execution over GPUs.	UC1

Table 3.2. User requirements for UC1 (Smart Cities).

Id	Description	Source
UR2.1	It shall be possible to obtain both a-priori estimates of expected, and actual measurements of, energy consumption of the execution of functions.	UC2
UR2.2	COGNIT Framework should be able to adapt to events with sudden peaks of function requests, in which the offloaded functions require much heavier computations and more frequent execution than usual.	UC2
UR2.3	Possibility for devices to have access to GPUs, when available, during high-alert mode.	UC2

Table 3.3. User requirements for UC2 (Wildfire Detection).

Id	Description	Source
UR3.1	Device Client and user applications shall share a maximum of 500 kB of available RAM in total.	UC3
UR3.2	It shall be possible for the user application to dynamically use scalable resources for function execution due to changes in the application demand.	UC3
UR3.3	The COGNIT Framework shall have support for the C programming language.	UC3

Table 3.4. User requirements for UC3 (Energy).

Id	Description	Source
UR4.1	The COGNIT Framework should have the ability to dynamically set the permissible edge nodes for executing the function based on policy (e.g. geographic security zones, distance to edge node).	UC4
UR4.2	The COGNIT Framework should have the ability to make data available between different (edge) clusters transparently to the user application.	UC4
UR4.3	The Device should be able to request the execution of a function as close as possible to the Device's location, satisfying latency requirements.	UC4

Table 3.5. User requirements for UC4 (Cybersecurity).

PART II. Architecture Definition

4. COGNIT Framework

In Deliverable D2.3, we have reported some limitations of the previous architecture. In particular, the main limitation is related to the management of the Serverless Runtime by the Device itself, an approach that has consequences in the optimization and orchestration of resources along the cloud-edge continuum. In particular, this limitation led to:

- A public IP being required for each Serverless Runtime, since the Device communicates directly with it. This requires the usage of IPv6 in order to scale up to a large number of Devices.
- Serverless Runtimes not being able to migrate between edge clusters on different cloud providers since the public IP cannot be moved from one provider to another.
- Creation of Serverless Runtimes cannot be anticipated by the AI-Enabled Orchestrator in order to reduce the cold start for low-latency applications.
- Applications whose execution of functions are triggered by events create Serverless Runtimes that are not used until events happen. This results in a suboptimal use of infrastructure resources.
- Developers need to directly manage the lifecycle of the Serverless Runtime, by creating, updating and releasing it according to the application requirements.

In the Third Research & Innovation Cycle (M16-M21), the Project has focused on the design of an improved COGNIT Architecture and the definition of a series of new components in order to address the limitations mentioned above. Therefore, this deliverable describes the design of the improved architecture, the specification, implementation and deployment of all necessary new components, and the update of the specifications, implementation and deployment of the existing components according to the updated Software Requirements.

4.1. COGNIT Application Profile

COGNIT addresses emerging mobile and IoT edge device applications that need to augment their capabilities for computationally intensive data processing, using fast computational units, special-purpose resources and services (e.g. GPUs, HPC, DB), and low-latency connections (i.e. 5G). Using code offloading allows computationally intensive data processing to be executed outside the edge devices, sensors, and actuators; this translates to more efficient power management, fewer storage requirements, and better application performance where devices may not provide the hardware acceleration capabilities for increasing the performance of the critical regions of computations.

Although code offloading has been widely considered to save energy and increase responsiveness of mobile devices, the technique still faces many challenges pertaining to practical usage, namely the complexity of its integration with the cloud management environment, the dynamic configuration of the system, its scalability, and the lack of Offloading-as-a-Service implementations. COGNIT addresses existing code offloading limitations by developing a new distributed Serverless model that, integrated with a

Cognitive cloud-edge management platform, facilitates the development of elastic and scalable edge-based applications.

In order to provide end-users with seamless access to a continuous data processing environment, COGNIT allows application developers to easily integrate cloud-edge processing in their coding logic since it makes feasible to offload data processing code fragments and tasks from the end-user system, device, sensor, or actuator to the cognitive continuum in order to speed up computation, save energy, save bandwidth, or provide low latency.

Serverless computing is a cloud computing execution model in which the cloud provider allocates machine resources on demand, taking care of the servers on behalf of their customers. It has been rapidly adopted by developers because it relieves them of the burden of provisioning, scaling, and operating the underlying infrastructure. Different FaaS services (AWS Lambda, Azure Functions, Google Cloud Functions), and open source frameworks (Apache OpenWhisk, Iron Functions, Fission, Kubeless, OpenFaaS) are available in the market, but they assume that the cloud platform runs on large, highly homogeneous datacenters with commodity infrastructure, and the functions have a small footprint and short execution duration.

Table 4.1 summarises the main differences between the COGNIT model and the traditional Serverless/FaaS model implemented by existing cloud offerings and technologies:

	Serverless/FaaS Cloud Model	**COGNIT Serverless Cloud-Edge Model**
PROGRAMMING		
Programming model	Interconnected functions (code) defined at a Cloud provider	Single program source code on device that follows an asynchronous model
Where the function runs	On Cloud provider	On cloud-edge location
When the function runs	On event, cloud event-driven	On demand, application logic-driven
Application profile	Low footprint	Compute-intensive data processing
Program state	Stateless	Stateless / Stateful
Maximum runtime	Short (e.g. <900 seconds)	None
Maximum capacity	Limited (e.g. < 3GiB memory)	None
Communication patterns	Workflows and state machines	Results forwarding to other functions
Deployment requirements	Basic capacity (e.g. memory) & quotas	Performance, cost, security, and energy
OPERATION		
Scaling	Cloud provider responsible	Cloud provider responsible
Deployment	Cloud provider responsible	Cloud provider responsible
Fault Tolerance	Cloud provider responsible	Cloud provider responsible

INFRASTRUCTURE		
Infrastructure	Single centralised cloud	Dynamic distributed cloud/edge
Location	Single centralised cloud	Developer selects (cloud-edge continuum)
Special-purpose devices	None	Hardware (GPU), services (AI)

Table 4.1. COGNIT Serverless Cloud-Edge Model vs traditional Serverless/FaaS Cloud Model.

4.2. COGNIT Execution Model

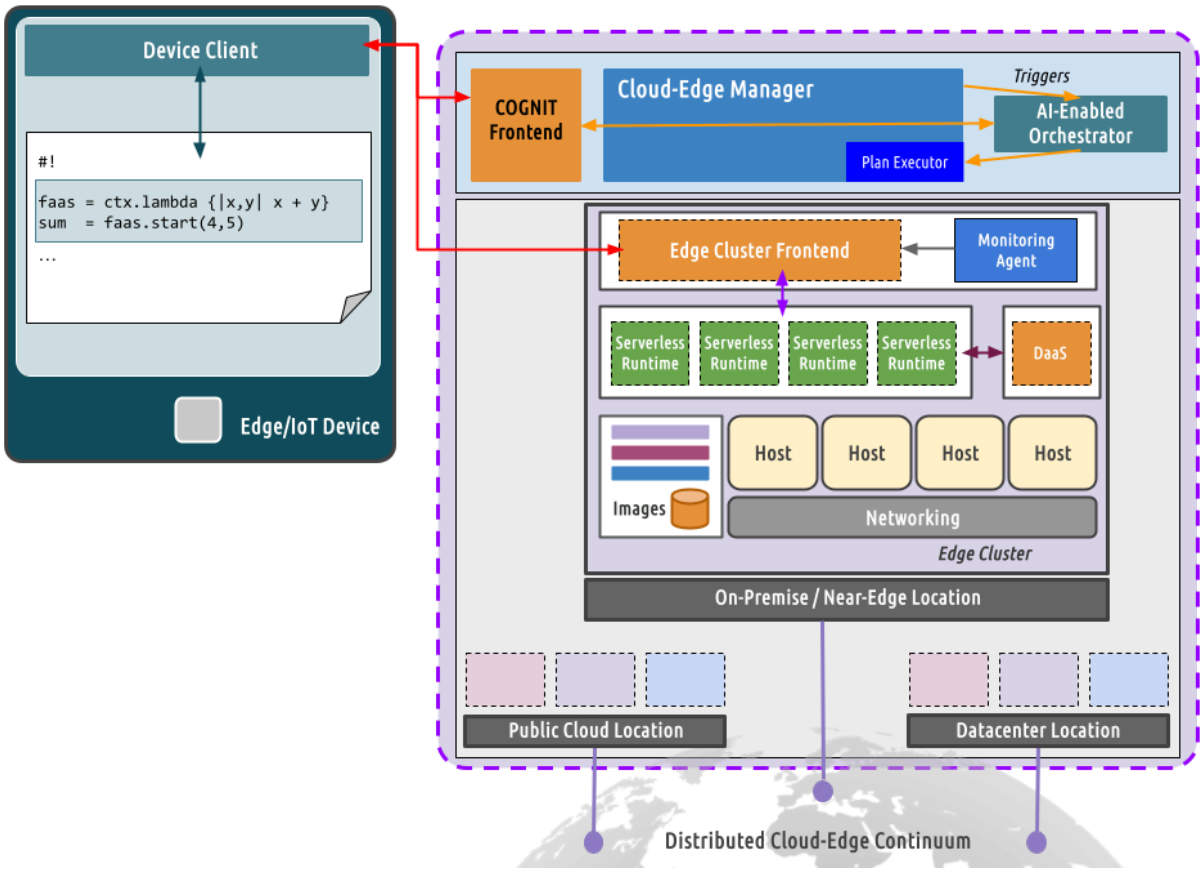

Figure 4.1. General view of the modified COGNIT Architecture.

The modified architecture of the COGNIT Framework consists of different components as depicted in Figure 4.1.

The first group of components, described in detail in Section 5 (Distributed Serverless Model for Edge Application Development), consists of the following:

- **Device Client.** It allows to offload function executions on the cloud-edge continuum using a Serverless paradigm and then perform different tasks such as function execution, transfer data from external sources and upload data from the device itself.

- **COGNIT Frontend.** It is the main entry point for the COGNIT service. It authenticates the device client that would like to use the service and returns a token that can be used by the Device for interacting with the COGNIT components. It provides different endpoints for uploading application requirements, storing functions and data, and it returns information about the most suitable resources where the device can offload the execution of the functions.
- **Edge Cluster.** It is a semi-autonomous complete environment with a dedicated software-defined resource set providing an abstraction layer enabling interoperability across the multi-provider cloud-edge infrastructure. Edge Clusters allow the execution of functions that are offloaded by the devices within a hardened and secure environment called **Serverless Runtime**.

The other architectural components, described in detail in Section 6 (Cognitive Cloud-Edge Module), allow the management of the cloud-edge continuum resources in an intelligent and adaptive way:

- **Cloud-Edge Manager**. It is responsible for managing the highly distributed and heterogeneous resources of the cloud-edge continuum and the lifecycle of Serverless Runtimes within each Edge Cluster for the execution of the functions offloaded by the devices.
- **AI-Enabled Orchestrator**. It manages and optimises the lifecycle of Edge Clusters and the Serverless Runtimes within each Edge Clusters according to the application requirements, infrastructure and virtual resource usage, and energy-aware policies.

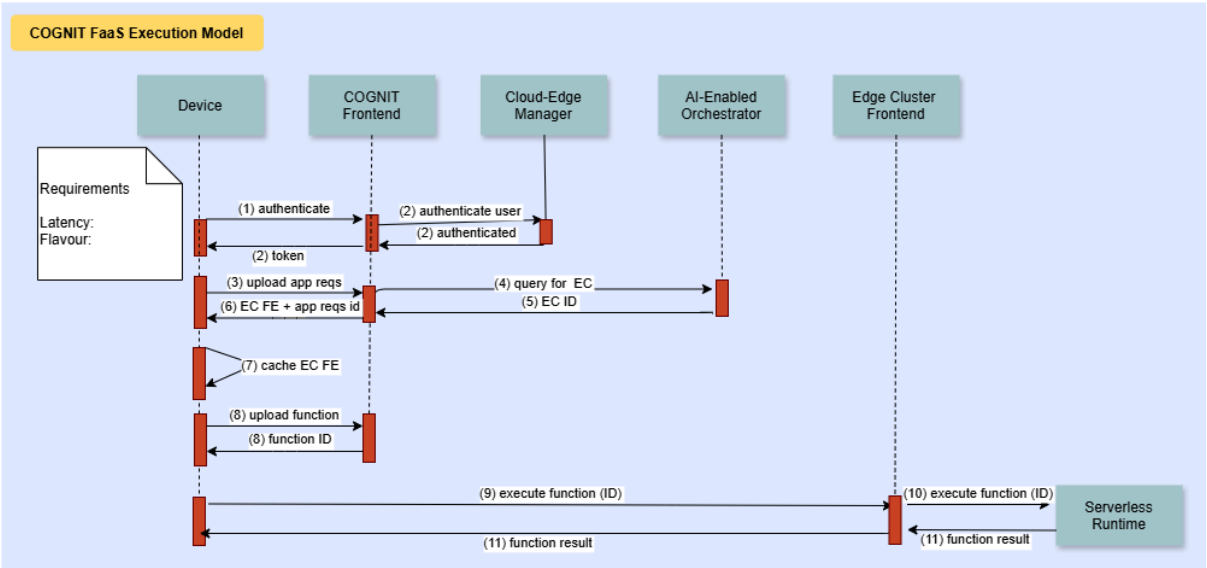

Figure 4.2. COGNIT Execution Model sequence diagram.

In Figure 4.2 a sequence diagram shows the communications between the different components. Different phases can be identified in the diagram:

1) The device will authenticate with the COGNIT Frontend sending username and password using a secure channel communication.
2) The COGNIT Frontend authenticates the device through the Cloud-Edge Manager and if it is successful will return an authorization token that can be used by the device in the subsequent interaction with the COGNIT Framework components.
3) The device sends to the COGNIT Frontend the application requirements.
4) The COGNIT Frontend stores the application requirements of the device and sends a request to the AI-Enabled Orchestrator for the most suitable Edge Cluster for the device.
5) The AI-Enabled Orchestrator selects the best Edge Cluster for the application and sends the information to the COGNIT Frontend.
6) The COGNIT Frontend sends back to the device a context related to the most suitable Edge Cluster.
7) The device caches the information about the Edge Cluster that will be used when the application offloads functions to the COGNIT Framework.
8) The device client uploads the function to be offloaded to the COGNIT Frontend if it is not already there.
9) The device client sends the function id and the input parameters to the Edge Cluster Frontend and waits for the results related to the function execution.
10) The Edge Cluster Frontend, when receiving a request from a device, will select a Serverless Runtime to execute the function.
11) The Edge Cluster Frontend, when receiving the result from the Serverless Runtime, sends the result back to the device client.

5. Distributed Serverless Model for Edge Application Development

In this Section we detail the model for an application running on the Cloud-Edge continuum using a distributed Serverless model. The main components are a) the Device Client, b) the COGNIT Frontend and c) Edge Cluster Frontend:

5.1. Device Client

The Device Client allows devices to offload the execution of functions using resources managed by a deployment of the COGNIT Framework. The Device Client first needs to communicate with the COGNIT Frontend for authentication; once it is authenticated it will receive a token that can be used in subsequent interactions with the COGNIT Frontend and the Edge Cluster Frontend.

The Device Client allows uploading application requirements that are needed by the AI-Enabled Orchestrator to make decisions in order to optimise the performance applications and the overall Cloud-Edge resource; the device can update the requirements during the application execution.

The Device Client can upload functions to be executed in the COGNIT Framework using the COGNIT Frontend. This allows executing functions in any Edge Cluster of the cloud-edge continuum without uploading them every time. Each function has a unique identifier assigned by the AI-Enabled Orchestrator. The ID will be used by the Device Client when requesting the execution of a function to the Edge Cluster.

Finally the Device client can upload data to COGNIT using the COGNIT Frontend; in this case data can be accessed by the function from any Edge Cluster provisioned within COGNIT. The Device Client can also upload data directly to the Edge Cluster; in this case the data will be visible only to functions executed on the same Edge Cluster.

The Device Client can also measure latency against the different Edge Clusters and push metrics to the COGNIT Framework; those metrics are used by the AI-Enabled Orchestrator to make decisions about the provisioning of Edge Clusters that have to satisfy latency requirements from the application.

5.2. COGNIT Frontend

The COGNIT Frontend is the main entry point for the COGNIT service. It authenticates the device that would like to use the service; once the device has been authenticated it generates a token that can be used by the device to interact with the different components of the COGNIT framework (i.e. COGNIT Frontend and Edge Cluster) to offload functions.

When it is requested by the device, the COGNIT Frontend queries the AI-Enabled Orchestrator for the most suitable Edge Cluster for the device to offload the execution of the function and returns an Edge Cluster context to the device.

Furthermore, the COGNIT Frontend provides an endpoint for the device to upload application requirements that will be stored and used subsequently by the AI-Enabled Orchestrator for optimising the resources needed by the devices to offload functions related to the application.

Finally the COGNIT Frontend allows the devices to upload data from the devices themselves or from external storages; this capability is used by the device client for two main reasons:

- To upload functions that will be used subsequently by the Serverless Runtime for their execution.
- To upload data that can be used by the device functions when they are executed in any Edge Cluster of the cloud-edge continuum.

5.3. Edge Cluster

The **Edge Cluster** is a semi-autonomous complete environment with a dedicated **software-defined resource set** (compute, network and storage) providing an abstraction layer enabling interoperability across the multi-provider cloud-edge infrastructure. Edge Clusters can be provisioned anywhere in the cloud-edge continuum (on-premises, on public cloud and edge providers).

The **Edge Cluster Frontend** is responsible for executing functions that are sent by the Device Client and providing the results to the Device Client itself. According to the application requirements, it selects the appropriate **Serverless Runtime** (i.e. a hardened and secure environment where to execute device functions) to run the functions.

In particular, the Edge Cluster Frontend implements the following capabilities:

- Load balances the function requests from the different devices according to the application requirements and Edge Cluster resource availability.
- Executes the function on the Serverless Runtime.
- Returns the function result to the Device Runtime.
- Allows devices to upload data locally to the Edge Cluster.
- Provides an endpoint for latency measurements from the device.

6. Cognitive Cloud-Edge Module

The Cognitive Cloud-Edge Module, in line with the Project's Research Challenge 4—*"Portable and adaptive execution of serverless workloads across a multi-provider cloud-edge continuum"* and Research Challenge 5—*"Automatic and intelligent adaptation of the cloud-edge continuum to the changing demands of the applications"*—will decide about the placement, mobility and elasticity of Serverless Runtimes across the cloud-edge continuum and about the elasticity measures to be applied to the cloud-edge infrastructure, all that in order to respond to the application requirements sent by the edge devices and the need to proactively optimise placements, costs, and other utility functions (e.g. energy consumption) according to the dynamic changes in the conditions and requirements of the application or to those unexpected incidents affecting the underlying infrastructure.

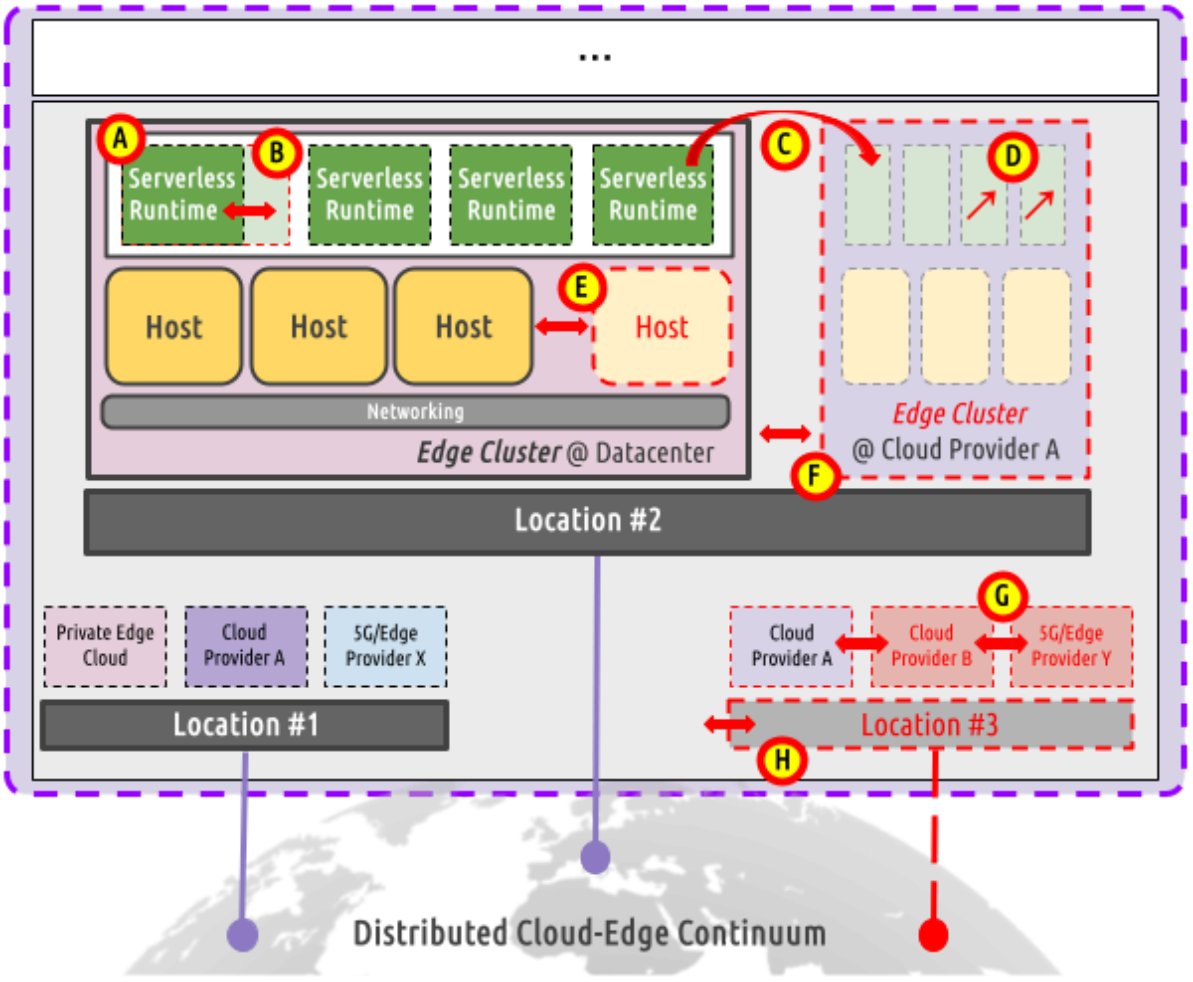

Figure 6.1. Workload and cloud-edge infrastructure adaptability supported by COGNIT.

The following table describes in more detail how the COGNIT Framework will deal with the adaptability required for the automatic placement and mobility of the Serverless Runtimes

and for leveraging in an intelligent and optimal way the elasticity capabilities that the infrastructure across the cloud-edge continuum provides:

Feature	Description	Activation
(A) Optimal Serverless Runtime Placement	Instantiate a Serverless Runtime on an Edge Cluster able to meet its current requirements.	✔ **Reactive**: edge applications trigger the instantiation or update of Serverless Runtimes
(B) Serverless Runtime Vertical Elasticity	Scale up/down the resources associated with the workloads (e.g. VMs) hosting the Serverless Runtime.	✔ **Reactive**: due to event-driven elasticity rules or changes in application requirements ✔ **Proactive**: due to AI-driven forecasting techniques.
(C) Serverless Runtime Mobility	Live migrate the Serverless Runtime to a different Edge Cluster.	✔ **Reactive**: due to event-driven mobility rules or changes in edge application requirements ✔ **Proactive**: due to AI-driven forecasting techniques.
(D) Optimal Global Workload Placement	Optimise the placement of existing Serverless Runtimes.	✔ **Proactive**: due to AI-driven optimisation techniques (e.g. reduce overall energy consumption and/or carbon emission footprint).
(E) Infrastructure Cluster Elasticity	Scale up/down existing Edge Clusters by activating/deactivating hosts (i.e. servers).	✔ **Reactive**: due to event-driven elasticity rules or changes in edge application requirements ✔ **Proactive**: due to AI-driven forecasting techniques.
(F) Infrastructure Horizontal Elasticity	Increase the number of available Edge Clusters in a given location.	✔ **Reactive**: due to event-driven elasticity rules or changes in edge application requirements ✔ **Proactive**: due to AI-driven forecasting techniques.
(G) Additional Cloud/Edge Provider	Increase the number of available cloud/edge service providers offering infrastructure resources.	✔ **Reactive**: due to event-driven provisioning rules or changes in edge application requirements

(H) Additional Cloud/Edge Locations	Incorporate new geographical locations to the cloud-edge continuum.	✔ **Reactive**: due to event-driven provisioning rules or changes in edge application requirements

Table 6.1. COGNIT reactive and proactive adaptability features.

In order to respond to those adaptability requirements, the Cognitive Cloud-Edge Module will be based on two main components, which will be presented in detail in the following sections:

- The **Cloud-Edge Manager** is in charge of managing the cloud-edge continuum infrastructure and of performing actions to manage the lifecycle of the Edge Clusters and Serverless Rutimes, collecting their metrics and monitoring the infrastructure resources used by them.
- The **AI-Enabled Orchestrator** manages and optimises the lifecycle of Edge Clusters and the Serverless Runtimes within each Edge Cluster according to the application requirements, infrastructure and virtual resource usage, and energy-aware policies. It uses AI-driven approaches based on the metrics collected by the Cloud-Edge-Manager and produces deployment plans for the Cloud-Edge Manager to manage both Edge Clusters (i.e. provisioning/deprovisioning, horizontal scaling) and Serverless Runtimes within an Edge Cluster based on dynamic conditions in the infrastructure, application requirements (sent by the Devices), energy usage policies.

6.1. Cloud-Edge Manager

The main responsibilities of the Cloud-Edge Manager are:

- Managing the cloud-edge continuum infrastructure (i.e. physical computational hosts, networks and storages across multi-cloud providers and edge locations) that is used by the AI-Enabled Orchestrator to optimise the execution of the applications based on the requirements sent by the device.
- Executing deployment plans produced by the AI-Enabled Orchestrator to optimise the usage of Edge Cluster resources and for the optimal placement of Serverless Runtimes to offload functions from the devices.
- Monitoring the cloud-edge infrastructure and the application workloads to provide the AI-Enabled Orchestrator with information to implement automatic and intelligent adaptation for the execution of the application.
- Providing authentication and authorization mechanisms for accessing and securing resources such as physical hosts, virtual machines, networks, services, etc.

The Cloud-Edge Manager provisions resources across the cloud-edge continuum and offers a simplified, efficient way to deploy Serverless Runtimes as virtualized workloads. The Cloud-Edge Manager is responsible for coordinating the use of a set of highly heterogeneous and distributed locations, providing a uniform view of the underlying

resources, in such a way that the application functions can be executed by the devices anywhere in the cloud-edge infrastructure.

The Cloud-Edge Manager has the following features, that are described in details in the following section:

- Exposes an API (Provider Catalogue) used by AI-Enabled Orchestrator to access information about the resource providers available in the cloud-edge continuum.
- Responsible for the management of the Serverless Runtimes lifecycle within each Edge Cluster.
- Provides a monitoring service aggregating information from the different Edge Clusters, with a highly distributed and reliable persistence storage used for building AI/ML models and for real-time predictions.
- Exposes an API (Plan Executor) used by the AI-Enabled Orchestrator to submit deployment plans for updating the resources within the cloud-edge continuum.
- Responsible for the management of the Edge Clusters lifecycle.
- Provides support for the decentralised authentication mechanism based on authorization tokens used in order to to enforce access control to requests from Device Clients to COGNIT.

Provider Catalogue

The Cloud-Edge Manager maintains a catalogue with a list of resource providers available in the cloud-edge continuum. The catalogue allows the AI-Enabled Orchestrator to select which providers/locations are better suited according to the device requirements (in terms of cost, capacity, energy, latency, bandwidth, etc).

Each provider has an entry in the catalogue with:

- Unique ID.
- List of locations, each location with a list of instance types.
- Each instance type has
    - Name.
    - Capacity.
    - Price per hour.
    - Price per megabyte (outbound and inbound).
    - Energy consumption per hour.
- Link to the Provider Driver (custom adapter that uses the provider API).
- Additional characteristics:
    - Public IPs.
    - GPUs.
    - Backend storage types.

The Provider Catalogue will contain entries related to public, private cloud or self hosted infrastructures hosted in the European Union. A set of filters (i.e. for latency, cost, energy, specific characteristics) are available to the AI-Enabled Orchestrator to select the appropriate provider according to the requirements provided by the device.

Edge Cluster Management

The main management unit of the Cloud-Edge Manager where device applications are executed is the **Edge Cluster**. An Edge Cluster is a semi-autonomous complete environment with a dedicated software-defined resource set (compute, network and storage) able to work on unreliable infrastructures. Given the heterogeneity of cloud-edge systems, the Edge Cluster creates an abstraction layer enabling interoperability across the multi-provider cloud-edge infrastructure. Edge Clusters can be provisioned anywhere in the cloud-edge (on-premises, on public cloud and edge providers), but we assume that each Edge Cluster is defined only for a specific provider of the cloud-edge – a provider can be a public cloud provider, an edge local provider, a set of far edge devices, etc.

The Cloud-Edge Manager exposes an internal API that provides the following operations to the Plan Executor for the implementation of the plans related to the management of Edge Clusters resources:

Operation	Definition
`Create`	Creates a new Edge Cluster on one of the locations available in the Catalogue.
`Delete`	Deletes an Edge Cluster.
`List`	Lists existing Edge Clusters and returns details for each of them.
`Configure`	Configures services on the Edge Cluster's hosts.
`Scale`	Scales up/down the size of an existing Edge Cluster (i.e. add or remove hosts).

Table 6.2. Edge Cluster management operations provided by the Cloud-Edge Manager.

The Cloud-Edge Manager will use a **Provider Driver** (a custom adapter for the provider API) that will automatically create the necessary resources for the Edge Cluster and will configure the different created entities (physical hosts, networks and storage) to provide software-defined components:

- Hypervisors for the deployment of workloads (i.e. Serverless Runtime) as VMs, microVMs or containers.
- Software-defined storage for storing the Serverless Runtimes images and for providing data blocks to the DaaS.
- Software-defined networks (i.e. virtual/overlay networks) for providing the communication of the Serverless Runtimes within the Edge Cluster and for the communication with the public Internet (i.e. the edge devices).
- Edge Cluster Frontend to interface with the devices for offloading functions.
- Monitoring agent to monitor the infrastructure and virtual resources, and Serverless Runtimes within the Edge Cluster.

Serverless Runtime Management

The Cloud-Edge Manager exposes an API that allows the following operations related to the deployment of Serverless Runtime on Edge Clusters:

Operation	Definition
`Create`	Creates a new Serverless Runtime on an Edge Cluster.
`Check`	Checks the status of the Serverless Runtime (i.e. PENDING, DEPLOYING, RUNNING, SCALING, …).
`Update`	Updates the Serverless Runtime deployment (e.g. scheduling on another Edge Cluster).
`Scale`	Scales the resources of the Serverless Runtime (e.g. increase number of CPUs).
`Delete`	Deletes a Serverless Runtime.

Table 6.3. Methods for Serverless Runtime management provided by the Cloud-Edge Manager.

Plan Executor

The Plan Executor is responsible for translating deployment plans provided by the AI-Enabled Orchestrator into a sequence of actions for the Cloud-Edge Manager related to management of Edge Clusters and/or management of Serverless Runtime. The Plan Executor employs a plugin mechanism that accommodates different drivers tailored to interact with specific APIs of the Cloud Edge Manager, facilitating actions such as creating a Serverless Runtime in an Edge Cluster, scaling resources of a Serverless Runtime, adding a new host to an existing Edge Cluster.

The Plan Executor exposes an API that allows the AI-Enabled Orchestrator to submit deployment plans.

Operation	Definition
`Apply`	Submit a deployment plan

Table 6.4. Methods for the Plan Executor.

Monitoring, Scaling, and Migration

The provision of Edge Clusters and placement of Serverless Runtime can be updated by the AI-Enabled Orchestrator according to the metrics that are collected by the Cloud-Edge Manager. The metrics are related to both the Edge Clusters (i.e. hosts, network, storage, software-defined components) and the Serverless Runtimes (application-defined metrics, e.g. average execution time of a function). According to the information collected, the AI-Enabled Orchestrator will produce a deployment plan for the Cloud-Edge Manager to

update the Edge Clusters and the placement of the Serverless Runtime according to the following scenarios:

- Provisioning and deprovisioning of Edge Cluster.
- Scaling up/down the Edge Cluster resources.
- Scaling up/down the Serverless Runtime.
- Migrating the Serverless Runtime from one host to another, or even from one Edge Cluster to another.

In addition, the Cloud-Edge Manager should provide mechanisms to ensure a secure and trusted environment for the Serverless Runtimes, such as intrusion and anomaly detection. More details about secure and trusted execution are reported in Section 7.

Authentication and Authorization

The Cloud-Edge Manager provides an authentication system based on username and password, where information and secrets are stored in the Cloud-Edge Manager itself. Dedicated external user authentication drivers can be used to leverage additional authentication mechanisms or sources of information about the users (e.g. LDAP, OIDC). Users can belong to groups that are authorization boundaries for the users.

The Cloud-Edge Manager provides an ACL authorization system that enables fine-tuning of the allowed operations for any user, or group of users. Each operation generates an authorization request that is checked against the registered set of ACL rules. The Cloud-Edge Manager then can grant permission, or reject the request. Therefore the user roles can be tailored according to their cloud-edge infrastructure needs.

Each user entity can represent single or multiple devices that need to use cloud-edge resources to offload workloads. The devices are authenticated by the COGNIT Frontend through a special mechanism as described in the following.

The COGNIT Frontend uses a mechanism to delegate the authentication process to the Cloud-Edge Manager that can be used for the initial authentication of the device. Advanced mechanisms for device authentication and authorization are described in more detail in Section 7.2.

6.2. AI-Enabled Orchestrator

The AI-Enabled Orchestrator is responsible for managing and optimising the lifecycle of Edge Clusters (and the Serverless Runtimes within each Edge Cluster) according to the application requirements (sent by the Devices), infrastructure and virtual resource usage, and energy-aware policies.

The AI-Enable Orchestrator **produces deployment plans** for the Cloud-Edge Manager based on dynamic conditions in the infrastructure, the infrastructure policies and requirements sent by the different devices. In particular the deployment plans are related to the:

- Optimisation of a single Edge Cluster resource (Serverless Runtimes placement, resizing, migration, ...) taking into account the monitoring of the resources (e.g., virtual machine resource availability, networks, etc.) used by the Serverless Runtime and the application metrics (e.g., application workload) sent by the Serverless Runtime.
- Optimisation of cloud-edge continuum resources (create new Edge Clusters, scale existing Edge Clusters, delete Edge Clusters, …) according to the multi-criteria policies: application demands (e.g., function request frequency, latency, GPU access, etc.) and environmental policies (e.g., cost, energy efficiency, green-energy availability, etc.).
- The decision of optimising Edge Clusters and Serverless Runtimes for diverse tasks (e.g., placement, migration, predicting function arrival time, etc.) will be made with AI-driven models and optimizations techniques built based on collected metrics.

The AI-Enabled Orchestrator is a proactive component that will dynamically optimise Serverless Runtime placement, taking into account changes in requirements sent by devices, the monitoring of resources (e.g., virtual machines resources such as CPU, GPU, memory, storage, networks, etc.) used by the Serverless Runtime, and application metrics sent by the Serverless Runtime related to the function execution.

The AI-Enabled Orchestrator will utilise these metrics for building learning models (e.g., self-supervised learning, contrastive learning, transformers) that adapt to changes in application requests from devices and changes in Edge Cluster resources for smart placement across the distributed cloud-edge continuum. Furthermore, the orchestrator will optimise the application workloads using multiobjective optimization (e.g., NSGA-III, Bayesian optimization) techniques to mitigate uncertainty and complexity in multiple objectives.

According to the deployment plan produced by the AI-Enabled Orchestrator for each task, the Cloud-Edge Manager will execute actions against each task in order to reach the declarative state in the plan creating, deleting and/or updating resources (Edge Clusters, Serverless Runtime) across the cloud-edge continuum.

7. Secure and Trusted Execution of Computing Environments on the Multi-Provider Cloud-Edge Continuum

The Multi-Provider Cloud-Edge Continuum presents an increased attack surface, increasing the security risk. Visibility and control of the edge devices and nodes, for example, is not always guaranteed, and may be difficult to achieve, especially in highly distributed and heterogeneous edge cloud environments. In order to secure and provide a level of trust in COGNIT, we will research state-of-the-art security mechanisms, i.e. advanced access control based on policies and attributes, trusted/confidential techniques, and the application of Federated Learning for privacy-preserving model training.

7.1. Risk analysis

As the COGNIT Architecture and components have changed, the risk analysis is being adapted to the updated list of components and interactions between them. This work is following a well defined risk analysis methodology relying on threat modelling that is described in further detail in D3.3. Thanks to the new architecture and the threat modelling methodology the updated risk analysis will be more precise and complete and we will be able to identify more relevant counter-measures.

7.2. Advanced access control

The complexity of FaaS workloads necessitates a very granular access control: who can run what function, what data can be accessed, when a workload can run and how much resources can be consumed. In order to implement these requirements, the COGNIT Framework can rely on the Cloud-Edge Manager authorization services that secures resources such as hosts, virtual machines, networks, services, … using for example access control lists (ACLs), certificates or tokens.

The Cybersecurity Use Case brings additional access control requirements, with the need for security policies on geographic zones that define a security level for specific geographic areas. We have identified for example Open Policy Agent as a solution to implement context-aware security policies in a declarative and portable way.

Decentralised Authentication

Biscuit authentication has been identified as a suitable tool for providing critical security and authorization capabilities that enhance COGNIT. Here are the key advantages:

- The COGNIT Frontend acts as the central token issuer for device clients.
- When a device client needs to make a request, it obtains a Biscuit token from the COGNIT Frontend.
- The token can then be used to authenticate against any component in the COGNIT Framework.

- Each component (e.g. Edge Cluster Frontend) independently validates tokens using the COGNIT Frontend's public key.
- No central authentication server is required for validation.

Function Execution Control

While OpenNebula provides basic authorization, Biscuit extends these capabilities with granular access control specifically designed for COGNIT's requirements:

Biscuit tokens can encode specific permissions for
- Execution modes (e.g., synchronous, asynchronous).
- Time-based limitations (execution duration, validity periods).
- Execution quotas and rate limits.
- Resource-specific constraints.

7.3. Confidential computing

Computing at the Edge is much more vulnerable with respect to confidentiality aspects of data because either the operation is in environments outside datacenter perimeters or the infrastructure is managed by 3rd party and not in your own control. In order to keep the confidentiality of data, especially in cases that process personal data (e.g. Energy Use Case or Cybersecurity Use Case), confidential (or trusted) computing techniques can be applied. Confidential Computing (CC) relies on Trusted Execution Environments (TEEs) which are secure processing areas in modern CPUs, that guarantee that the code and data are protected with respect to confidentiality and integrity against access and tampering by unauthorised actors. This way, data “in use” is protected while in RAM, which is an appreciable protection improvement beyond already common methods for data in transit and at rest.

There are several technologies available in the market. Intel® Software Guard Extensions (SGX) was the first noticeable development in this area, but has not reached broader adoption in the software industry. It requires adaptations to the code of an application and by that solely takes care of the processing of a specific application. Current x86-64 hardware implementations are Intel TDX and AMD-SEV, which are transparent to the application code and protect the whole virtual machine operation (CVM – Confidential Virtual Machine). ARM has also introduced their Confidential Compute Architecture (CCA) in their Armv9-A family. Based on the advanced availability of Open Source code, the attestation approach[2] and the availability of hardware, we decided to proceed in the COGNIT project with AMD-SEV technology, to run the serverless runtime in a CVM, in case this level of confidentiality is required.

In order to expose the confidential computing capabilities of the testbed infrastructure to the COGNIT Framework, some adaptations are required. The Cloud-Edge Manager needs

[2] “Enabling Security and Performance Enhancements in the Confidential Computing Stack” by Vijay Varadharajan and Udaya Tupakula, *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2020

to provide in the Provider Catalogue specific filters to identify CC-capable hosts, and the AI-Enabled Orchestrator should support this deployment constraint for edge nodes.

7.4. Federated Learning

The privacy of data produced by devices and stored on each edge node is particularly important. Those devices can for example record personal information of individuals that is covered by GDPR, or handle information that needs to stay locally and cannot be shared with other devices or systems. This poses a challenge for machine learning approaches such as AI-based anomaly detection algorithms that rely on data collected by various devices to train detection models, and then need to redistribute those models to the anomaly detection function. Federated Learning is the main state-of-the-art approach to solving this problem. Federated learning can use local datasets in the edge to train machine learning algorithms without exchanging the sensitive edge data samples.

This approach provides the added benefit of reducing the volume of exchanged data for training, which is something particularly interesting in edge contexts where bandwidth and connectivity can be limited and unreliable. The ecosystem for Federated Learning frameworks and libraries is very dynamic, from plugins of established machine learning platforms such as TensorFlow Federated[3] to interoperable frameworks like Flower[4] which focuses on edge use cases. The applicability of federated learning techniques in a FaaS edge environment will be studied in the Cybersecurity Use Case, where anomaly detection will be performed on vehicles that have confidentiality requirements preventing the exchange of information for training.

[3] https://www.tensorflow.org/federated
[4] https://flower.dev

8. Software Requirements

This section provides an overview of the Software Requirements.

8.1. Device Client

SR1.1 Interface with COGNIT Frontend

Description: Implementation of the communication of the Device Client with the COGNIT Frontend.

- The Device Client shall be able to ask for the most suitable (or a set of) Edge Cluster to which it should offload functions.
- The Device Client shall be able to upload application requirements to the COGNIT Frontend and update them.
- The Device Client shall be able to upload application functions to the COGNIT Frontend.
- The Device Client shall be able to upload data from the device to the COGNIT Frontend.
- The Device Client shall be able to request to transfer data from external resources to the COGNIT Frontend.

SR1.2 Interface with Edge Cluster

Description: Implementation of the communication between a Device Client and an Edge Cluster.

- The Device Client shall be able to invoke an execution of a function at the assigned Edge Cluster.
- The Device Client shall be able to upload data to the assigned Edge Cluster.

SR1.3 Programming languages

Description: Support for different programming languages.

- The Device Client shall support the C programming language.
- The Device Client shall support the Python programming language.

SR1.4 Low memory footprint for constrained devices

Description: Low memory footprint for constrained devices.

- The Device Client supporting the C programming language shall have a memory footprint lower than 500 kB.
- In order to be able to use the Device Client, the Device Application shall be able to implement an HTTP client with TLS capabilities.

SR1.5 Security
Description: The Device Client must be secured. ● All the communications between the Device Runtime and the COGNIT Frontend shall use JSON Web Tokens (JWT) for authentication and authorization. ● All the communications between the Device Runtime and the Edge Cluster shall use JSON Web Tokens (JWT) for authentication and authorization. ● Device Runtime shall take into account the latest legislative frameworks, such as the NIS2 directive, the GDPR, and the CRA.

SR1.6 Collecting Latency Measurements
Description: Latency measurements against Edge Clusters should be acquired by the Device Client. ● The Device Client shall implement a mechanism for latency measurement towards the assigned Edge Cluster.

8.2. COGNIT Frontend

SR2.1 COGNIT Frontend
Description: Provides an entry point for devices to communicate with the COGNIT Framework for offloading the execution of functions and uploading global data. The COGNIT Frontend: ● Must be able to authenticate the Device Client and return a token once the device is authenticated. ● Must interface with the Cloud-Edge Manager to delegate the authentication of devices. ● Must interface with the AI-Enabled Orchestrator for querying the most suitable Edge Clusters for devices. ● Must be able to provide the Device Client with information about Edge Clusters that can be used to offload functions. ● Must provide an endpoint for storing application requirements and other metadata (for instance, Device Client current location) from the Device Client, in a way that is accessible by the AI-Enabled Orchestrator. ● Must provide an endpoint for uploading files related to the execution of functions and global data.

8.3. Edge Cluster

SR3.1 Edge Cluster Frontend

Description: The Edge Cluster must provide an interface (Edge Cluster Frontend) for the Device Client to offload the execution of functions and to upload local data that is needed to execute the function. The Edge Cluster Frontend:

- Must provide an endpoint for the execution of functions previously uploaded through the COGNIT Frontend.
- Must provide an endpoint for uploading data from the Device Client to the Edge Cluster, to be stored locally at the Edge Cluster.
- Must interface with Serverless Runtimes deployed within the Edge Cluster in order to execute functions as requested by the Device Client.
- Must provide an endpoint for measuring latency from the Device Client.

SR3.2 Secure and Trusted Serverless Runtimes

Description: The Serverless Runtime is the minimal execution unit for the execution of functions offloaded by Device Clients.

- The Serverless Runtime must expose a REST API for all execution variants for the Python and C Executor used by the Edge Cluster Frontend.
- The Cloud-Edge Manager must provide a base Serverless Runtime image that contains the following minimal capabilities: a REST API interface for executing functions, a secure channel for the communication within the Edge Cluster and a mechanism to push function-related metrics to the monitoring service.
- The COGNIT Framework must provide an automated procedure for building secure and trusted images (vulnerability scans, security assessment) related to different flavours[5] of Serverless Runtimes, according to the need of the different Use Cases.
- The Serverless Runtime must provide a mechanism to report function execution metrics in a way that can be consumed by the AI-Enabled Orchestrator.

8.4. Cloud-Edge Manager

SR4.1 Provider Catalogue

Description: Implement a backend to persist information about the available providers that can be used to extend the capacity of the COGNIT infrastructure.

- Provider Catalogue data model must be persistent.

[5] With “flavours” we are referring to different VM templates available for creating specific Serverless Runtimes.

- Provider Catalogue must implement an API to manage providers entries.
- Provider Catalogue must be able to filter providers according to latency, costs, energy consumption and/or specific characteristics.

SR4.2 Edge Cluster Provisioning

Description: The Cloud-Edge Manager must be able to provision Edge Clusters as a set of software-defined compute, network, storage on any cloud/edge location available in the Provider Catalogue.

- Cloud-Edge Manager must provide an API to manage the lifecycle (provision/update/scale/deprovision) of Edge Clusters.

SR4.3 Serverless Runtime Deployment

Description: The Cloud-Edge Manager must be able to deploy Serverless Runtimes as Virtualized Workloads within an Edge Cluster.

- Cloud-Edge Manager shall provide an API to manage the life cycle (deploy/update/scale/migrate) of Serverless Runtimes.

SR4.4 Metrics, Monitoring, Auditing

Description: Edge-Clusters monitoring, Serverless Runtimes metrics collection and continuous security assessment.

- A distributed system shall be used for monitoring Edge Cluster entities (hypervisor, virtual/overlay networks and datastores).
- A distributed system shall be used for collecting metrics from Serverless Runtimes deployed across the cloud-edge infrastructure.
- Intrusion and anomaly detection of the different Edge Cluster entities and Serverless Runtimes shall be available.
- Provisioning of an alerting system to trigger events and notifications to be routed to the AI-Enabled Orchestrator.

SR4.5 Authentication & Authorization

Description: Authentication and authorization mechanisms for accessing cloud-edge infrastructure resources by the devices for offloading workloads.

- Cloud-Edge Manager must provide mechanisms allowing the COGNIT Frontend to delegate authentication and authorization on behalf of the Device Client.
- Devices will use JWT tokens for the authorization to use resources provided by the Cloud-Edge Manager.

SR4.6 Plan Executor
Description: The Plan Executor is responsible for converting plans provided by the AI-Enabled Orchestrator in Cloud-Edge Manager actions for the life cycle management of Edge Clusters and Serverless Runtimes. It provides the following capabilities: ● A driver for executing Cloud-Edge Manager actions related to the life cycle management of Serverless Runtimes (create, update, migration, delete). ● A driver for executing Cloud-Edge Manager actions for the provisioning, deprovisioning and horizontal elasticity of Edge Clusters.

8.5. AI-Enabled Orchestrator

SR5.1 Building Learning Models
Description: Provide AI/ML models trained with input from collected metrics from the Cloud-Edge Manager monitoring service related to Edge Clusters and Serverless Runtimes deployed across the distributed cloud-edge continuum. Main features provided are: ● Workflows for training, testing and validation of AI/ML models for diverse downstream tasks: workload characterization, metrics prediction (such as CPU usage, function arrival time, workload, resource requirements), energy-aware scheduling. ● AI/ML Model repository for tracking and versioning ML model artefacts. ● Provide mechanisms for retraining and model updates according to the workload and infrastructure dynamics. ● Integration with the Cloud-Edge Manager monitoring system for both ML model training and inference.

SR5.2 Smart Management of Cloud-Edge Resources
Description: The AI-Enabled Orchestrator is responsible for the automated management of cloud-edge continuum resources in order to optimize the performance of the applications that are offloading functions to the COGNIT Framework. The AI-Enabled Orchestrator includes the following capabilities. ● Leverages AI/ML models and energy-aware optimization algorithms for the smart (re)allocation of Serverless Runtimes across the Edge Clusters. ● Dynamic workload and resource optimization in response to system-changing conditions of cloud-edge infrastructure using horizontal elasticity (i.e., creation, scaling and deletion of Edge Clusters).

- Exposes a REST API interface enabling orchestrating tasks to be triggered by the Cloud-Edge Manager.
- Interfaces with the Plan Executor for the submission of deployment plans.

8.6. Secure and Trusted Execution of Computing Environments

SR6.1 Advanced Access Control

Description: Implement policy-based access control to support security policies on geographic zones that define a security level for specific areas.

- Cloud-Edge Manager shall install and configure a policy-based access control service to enforce and manage security policies.
- Cloud-Edge Manager shall define appropriate security policies, based on various attributes (location, device type, etc.).

SR6.2 Confidential Computing

Description: Enable privacy protection for the application workloads at the hardware level using Confidential Computing (CC) techniques.

- The resource definition model shall be expanded for CC-capable hosts to be tagged as such.
- CC-capable constraint shall be included in the orchestrator to deploy workloads requiring CC to appropriate devices and hosts.

SR6.3 Federated Learning

Description: Enhance privacy of AI workloads that have confidentiality requirements preventing the exchange of information for training. Federated Learning techniques enable confidential or private data processing under the control of the data owner or controller, with only learned models shared.

- Federated Learning Frameworks to implement such approach of AI model training while preserving data privacy need to be surveyed
- Integration of such FL frameworks with the COGNIT framework to be investigated taking into account COGNIT Framework architectural restrictions and requirements.

9. User to Software Requirements Matching

This section provides the results of the requirement engineering process, which derives system requirements (functional and non-functional) from functional gaps in order to implement a system that fulfils both user requirements and sovereignty, sustainability, interoperability and security requirements. The following tables (Tables 9.1 to 9.6) summarise the resulting system requirements:

Id	Description	Source
SR1.1	Interface with COGNIT Frontend	UR0.1 UR0.2 UR0.3 UR0.4 UR0.8 UR0.9 UR1.2 UR1.4 UR1.5 UR2.3 UR3.1 UR3.2 UR3.3 UR4.1 UR4.2 UR4.3 IR0.3 SER0.1 SER0.3
SR1.2	Interface with Edge Cluster	UR0.2 UR0.3 UR0.4 UR1.3 IR0.3
SR1.3	Programming Languages	UR0.1 UR3.3 IR0.3
SR1.4	Low memory footprint for constrained devices	UR3.1 SER0.1 SER0.5 SER0.6

SR1.5	Security	SER0.1 SER0.2
SR1.6	Collecting Latency Measurements	SER0.1 SER0.2

Table 9.1. System requirements for the Device Client

Id	Description	Source
SR2.1	COGNIT Frontend	UR0.2 UR0.3 UR0.4 UR0.6 UR0.7 UR0.8 UR0.9 UR2.1 UR4.1 UR4.2 UR4.3 IR0.1 IR0.2 IR0.3 SER0.1 SER0.2 SER0.3 SER0.5

Table 9.2. System requirements for the COGNIT Frontend.

Id	Description	Source
SR3.1	Edge Cluster Frontend	UR0.1 UR0.2 UR0.3 UR0.4 UR0.10 UR1.1 UR1.2 UR1.3 UR1.4

		UR2.1 UR2.3 SUR0.1 IR0.1 IR0.2 IR0.3 SER0.1 SER0.2 SER0.6 SOR0.4
SR3.2	Secure and Trusted Serverless Runtimes	UR0.1 UR1.2 UR2.3 IR0.1 IR0.2 SER0.4 SER0.6

Table 9.3. System requirements for the Edge Cluster

Id	Description	Source
SR4.1	Provider Catalogue	UR0.5 UR4.1 SOR0.1 SUR0.1 IR0.1 IR0.2 IR0.3
SR4.2	Edge Cluster Provisioning	UR0.5 UR1.2 SOR0.3 SUR0.1 IR0.1 IR0.2 IR0.3 SER0.1 SER0.2
SR4.3	Serverless Runtime Deployment	UR1.1 UR2.3

		IR0.1 IR0.2 IR0.3 SER0.1 SER0.2 SER0.4
SR4.4	Metrics, Monitoring, Auditing	UR1.4 UR2.1 SUR0.1 IR0.1 IR0.2 IR0.3
SR4.5	Authentication & Authorization	UR0.8 UR1.1 IR0.1 IR0.2 IR0.3
SR4.6	Plan Executor	UR4.1 UR4.2 UR4.3 IR0.1 IR0.2 IR0.3

Table 9.4. System requirements for the Cloud-Edge Manager.

Id	Description	Source
SR5.1	Building Learning Models	IR0.2 SUR0.2 SUR0.3 UR1.2 UR2.1 UR2.2
SR5.2	Smart Management of Cloud-Edge Resources	UR0.6 UR0.7 UR1.2 UR1.3 UR2.2 UR3.2 UR4.1

		UR4.2 UR4.3

Table 9.5. System requirements for the AI-Enabled Orchestrator.

Id	Description	Source
SR6.1	Advanced Access Control	UR0.8 UR1.1 UR4.1 SER0.2 SER0.5
SR6.2	Confidential Computing	UR1.1 SER0.1 SER0.2 SER0.4
SR6.3	Federated Learning	UR1.1 IR0.2 SER0.2 SER0.4

Table 9.6. System requirements for Secure & Trusted Execution of Computing Environments.

PART III. Verification and Implementation Plan

10. Software Build and Verification

The COGNIT software development model will ensure that components are delivered securely, timely, and with a high-quality level. Figure 10.1 shows a DevSecOps approach where sample security controls are associated with each phase of the development. The phases cover the entire lifecycle of the platform, from the planning phase where requirements are gathered and specifications drafted to the operations and monitoring phases where the platform is available to end-users. This method relies heavily on automation to orchestrate the platform life cycle, using techniques such as continuous integration/deployment (CI/CD), infrastructure as code (IaaC), and observability.

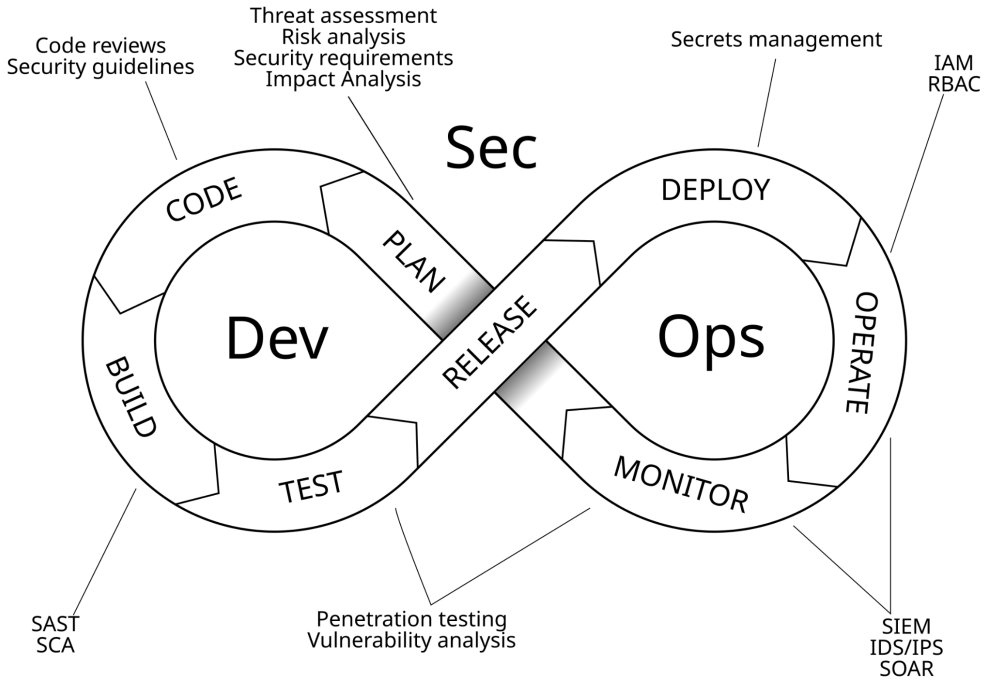

Figure 10.1. DevSecOps integrates development, operations, and security activities.

10.1. Verification Methodology

The goal of the verification process is to assess that the functional components of the software platform conforms to the Software Requirements identified in Section 8. This, in turn, will validate that the COGNIT Framework is feature-complete and able to achieve the objectives of the Projects' Use Cases.

In order to support the agile development adopted in this project, the verification process is integrated with the software development procedure mentioned before. The methodology is structured as follows:

- **Verification scenario**. Describes a simple user story that captures one or more functional requirements of a software requirement of a component (see Section 9).
- **Verification test**. An automated testing program that exercises the functional aspects of the scenario. Each verification test is then integrated into the certification platform to certificate and test software releases.

10.2. Verification Scenarios

This section presents a list of verification scenarios for verifying the initial set of Software Requirements defined in Section 8. Each COGNIT platform component is presented in a separate table that includes a brief description of each scenario.

SW Req.	Verification Scenario
SR1.1	VS1.1.1 The Device Client is able to get authorization from COGNIT and is able to send App valid requirements to the COGNIT Frontend. VS1.1.2 The Device Client is able to receive a valid Edge Cluster (effectively a valid Edge Cluster Frontend IP address). VS1.1.3 The Device Client is able to update the App requirements at any moment. VS1.1.4 The Device Client is able to receive a changed Edge Cluster seamlessly from a COGNIT's proactive decision making action. VS1.1.5 The Device Client is able to handle (upload/read) data on the COGNIT global layer.
SR1.2	VS 1.2.1 The Device Client is able to execute functions (either preloaded or by uploading it at the moment of execution) on the assigned Edge Cluster. VS1.2.2 The Device Client is able to handle (upload/read) data privately in the assigned Edge Cluster.
SR1.3	VS1.3.1 Test previously described validation scenarios implemented in C language. VS1.3.2 Test previously described validation scenarios implemented in Python language.
SR1.4	VS1.4.1 Test validation scenarios described above on a device with less than 500kB of RAM.

SR1.5	VS1.5.1 The Device Client is able to perform secure communications against the COGNIT Frontend and the assigned Edge Cluster Frontend with the acceptance of the authorization mechanism. VS1.5.2 The Device Client is not permitted any unauthorised action towards the COGNIT Frontend or the assigned Edge Cluster.
SR1.6	VS1.6.1 The Device Client is able to measure latencies to different Edge Clusters concurrently with other activities of the Client, to be able to monitor and make COGNIT aware of the effective latency of the potential Edge Clusters to be used.

Table 10.1. Verification scenarios for Device Client.

SW Req.	Verification Scenario
SR2.1	VS2.1.1 Authenticate a Device against the COGNIT Frontend and verify that an authorization token is returned. VS2.1.2 Upload application requirements to the COGNIT Frontend and verify that a unique ID is returned for the application requirements. VS2.1.3 Upload application requirements and query the COGNIT Frontend for an Edge Cluster and verify that it meets the application requirements. VS2.1.4 Upload a function to the COGNIT Frontend and verify that a unique ID is returned for that function. VS2.1.5 Test uploading and downloading data by the device to and from the COGNIT Frontend.

Table 10.2. Verification scenarios for the COGNIT Frontend.

SW Req.	Verification Scenario
SR3.1	VS3.1.1 Instantiate a Serverless Runtime and verify that a device can request the execution of a function to the Edge Cluster Frontend and assert the result of the function. VS3.1.2 Test uploading and downloading data by the device to and from the Edge Cluster using a secure communication channel.

SR3.2	VS3.2.1 Build a Serverless Runtime image, customized for each Use Case, in an automated way.

Table 10.3. Verification scenarios for the Edge Cluster.

SW Req.	Verification Scenario
SR4.1	VS4.1.1 Listing the providers belonging to the Provider Catalogue. VS4.1.2 Filtering the providers according to a desired latency threshold on a geographic area. VS4.1.3 Filtering the providers according to a cost per hour threshold. VS4.1.4 Filtering the providers according to energy consumption per hour threshold. VS4.1.5 Filtering the providers according to some specific hardware characteristics (e.g. GPUs, Trusted Execution Environments).
SR4.2	VS4.2.1 A YAML file containing the information about the provision is provided to the Cloud-Edge Manager that creates a new Edge Cluster. VS4.2.2 Query the Cloud-Edge Manager to return the status of an Edge Cluster identified by its ID. VS4.2.3 Query the Cloud-Edge Manager to scale up/down the number of hosts of an Edge Cluster identified by its ID. VS4.2.4 Query the Cloud-Edge Manager to delete an Edge Cluster identified by its ID.
SR4.3	VS4.3.1 A YAML file containing the information about the deployment is provided to the Cloud-Edge Manager that creates a new Serverless Runtime. VS4.3.2 Query the Cloud-Edge Manager to return the status of a Serverless Runtime identified by its ID. VS4.3.3 Query the Cloud-Edge Manager to scale up/down the resources (CPU, memory and disks) of a Serverless Runtime identified by its ID. VS4.3.4 Query the Cloud-Edge Manager to update the deployment of the Serverless Runtime identified by its ID.

	VS4.3.5 Query the Cloud-Edge Manager to delete a Serverless Runtime identified by its ID.
SR4.4	VS4.4.1 Create an Edge Cluster and deploy a Serverless Runtime and check the metrics collected for a certain period of time.
SR4.5	VS4.5.1 Test the creation of new users and groups. VS4.5.2 Assign ACLs to designated users and test the creation of new Edge Clusters and Serverless Runtimes. VS4.5.3 Communicate with the COGNIT Frontend and the Edge Cluster Frontend using tokens.
SR4.6	VS4.6.1 Submit a plan to the Plan Executor for creating new Serverless Runtimes and verify the deployment of the Serverless Runtimes. VS4.6.2 Submit a plan to the Plan Executor for creating a new Edge Cluster and verify that the Edge Cluster is created correctly. VS4.6.3 Submit a plan to the Plan Executor for resizing and migrating a Serverless Runtime. VS4.6.4 Submit a plan to the Plan Executor for increasing the number of hosts (horizontal scaling) of an existing Edge Cluster.

Table 10.4. Verification scenarios for the Cloud-Edge Manager.

SW Req.	Verification Scenario
SR5.1	VS5.1.1 List instances from Devices to Applications to System for metrics to be collected. VS5.1.2 Correlate and represent features that are ready to take as input to the Model. VS5.1.3 Feedback-aware performance check when training the model on represented features. VS5.1.4 Assess the ability in terms of AUROC score for each task (e.g., scheduling).
SR5.2	VS5.2.1 Assess the ability of workload and resource optimization in terms of cost and performance trade-off.

Table 10.5. Verification scenarios for the AI-Enabled Orchestrator.

SW Req.	Verification Scenario
SR6.1	VS6.1.1 Define a security policy that is based on a geographic zone attribute. VS6.1.2 Check enforcement of new security policy when edge device moves closer from one edge node to another.
SR6.2	VS6.2.1 Deploy a function on a host that provides confidential computing capability. VS 6.2.2 Check that the function is executed inside the host trusted execution environment (TEE).
SR6.3	VS6.3.1 Perform training of the ML algorithm without exchanging local data. VS6.3.2 Check that the redistributed models for inference do not contain private data.

Table 10.6. Verification scenarios for the Secure & Trusted Execution of Computing Environments.

11. Instantiation of the COGNIT Architecture

This section identifies the gaps and the initial set of technologies that will be used to instantiate the COGNIT Architecture through the implementation of the Software Requirements described in Section 8. Based on the analysis in Section 2 with regards to sovereignty, sustainability, interoperability, and security requirements, the choice of technologies involved in the implementation plan of the different software requirements has prioritised the use of European open source alternatives:

SR	Description	Implementation
SR1.1	Interface with COGNIT Frontend	New component (HTTPS client plus certificate management tool).
SR1.2	Interface with Edge Cluster	New component (HTTPS client plus certificate management tool).
SR2.1	COGNIT Frontend	New component.
SR3.1	Edge Cluster Frontend	New component.
SR3.2	Secure and Trusted Serverless Runtimes	New component developed as an OpenNebula Virtual Appliance (e.g. based on openSUSE and KVM), and incorporating specific applications such as a relational database (e.g. MariaDB) or object storage (e.g. MinIO).
SR4.1	Provider Catalogue	Expansion of the OpenNebula OneProvision Edge Catalogue.
SR4.2	Edge Cluster Provisioning	Expansion of the OpenNebula OneProvision component.
SR4.3	Serverless Runtime Deployment	Expansion of the OpenNebula *OneFlow*[6] component, with Serverless Runtimes defined as *OneFlow* Services.
SR4.4	Metrics, Monitoring, Auditing	Enhancement of OpenNebula *OneGate*[7] component, plus Prometheus integration, and Edge Cluster monitoring enhanced with auditing (e.g. for intrusion and anomaly detection).
SR4.5	Authentication & Authorization	Expansion of the OpenNebula Authentication component with added support for ACLs.

[6] https://docs.opennebula.io/6.10/management_and_operations/multivm_service_management/index.html
[7] https://docs.opennebula.io/6.10/management_and_operations/multivm_service_management/onegate_usage.html

SR4.6	Plan Executor	Expansion of OpenNebula *AIOps* Plan Executor component.
SR5.1	Building Learning Models	New component developed using ML/AI frameworks like PyTorch or TensorFlow.
SR5.2	Smart Operations/Management of Cloud-Edge Resources	Enhancement of the OpenNebula *AIOps* Orchestrator through the integration of new ML/AI models and algorithms for Cloud-Edge resource optimization.
SR6.1	Advanced Access Control	Enhancement of the OpenNebula support to ACLs and integration of Biscuit tokens.
SR6.2	Confidential Computing	Enhancement of the OpenNebula drivers that expose CC hardware capabilities.
SR6.3	Federated Learning	New component developed using Federating Learning frameworks such as Flower.

Table 11.1. Implementation plan per Software Requirement.

PART IV. Progress Report

12. Overall Development Status

12.1. Software Requirement Progress

The table below shows the status of each Software Requirement towards completion, following a simple colour code: – for activities that have not started yet or are on hold because of dependencies, ↻ for activities in progress, and ✓ for completed activities:

Software Requirements	**Cycle 1** (M4-M9)	**Cycle 2** (M10-M15)	**Cycle 3** (M16-M21)	**Cycle 4** (M22-M27)	**Cycle 5** (M28-M33)
Device Client					
SR1.1 Interface with COGNIT Frontend			↻	–	–
SR1.2 Interface with Edge Cluster			↻	–	–
SR1.3 Programming languages	↻	↻	↻	–	–
SR1.4 Low memory footprint for constrained devices	–	–	–	–	–
SR1.5 Security	–	–	↻	–	–
SR1.6 Collecting Latency Measurements			–	–	–
COGNIT Frontend					
SR2.1 COGNIT Frontend			↻	–	–
Edge Cluster					
SR3.1 Edge Cluster Frontend			↻	–	–

SR3.2 Secure and Trusted Serverless Runtimes	↻	↻	✓	–	–
Cloud-Edge Manager					
SR4.1 Provider Catalogue	–	–	–	–	–
SR4.2 Edge Cluster Provisioning	–	–	–	–	–
SR4.3 Serverless Runtime Deployment	↻	↻	✓	–	–
SR4.4 Metrics, Monitoring, Auditing	↻	↻	↻	–	–
SR4.5 Authentication & Authorization	↻	–	↻	–	–
SR4.6 Plan Executor			–	–	–
AI-Enabled Orchestrator					
SR5.1 Building Learning Models	↻	↻	↻	–	–
SR5.2 Smart Management of Cloud-Edge Resources	↻	↻	↻	–	–
Secure and Trusted Execution of Computing Environments					
SR6.1 Advanced Access Control	↻	↻	↻	–	–
SR6.2 Confidential Computing	↻	↻	↻	–	–
SR6.3 Federated Learning	–	–	–	–	–

Table 12.1. Current status of each Software Requirement towards completion, per research & innovation cycle.

12.2. Global KPIs Progress

The table below shows in which Milestone each Software Requirement is expected to contribute to meet the Project's global KPIs, showing activities already completed, those that have already started but not complete yet, and those to be started later on during the Project:

MEASURABLE & VERIFIABLE RESULT	KPI	Device Client						COGNIT Frontend		Edge Cluster		Cloud-Edge Manager						AI-Enabled Orchestrator		Secure & Trusted Execution of Computing Environments		
		SR1.1	SR1.2	SR1.3	SR1.4	SR1.5	SR1.6	SR2.1	SR3.1	SR3.2	SR4.1	SR4.2	SR4.3	SR4.4	SR4.5	SR4.6	SR5.1	SR5.2	SR6.1	SR6.2	SR6.3
Deployment of large-scale, highly distributed data processing environments.	**[KPI1.1]** Framework tested to scale up to tens of thousands of distributed nodes.										3										
Adaptation by offering a scalable monitoring system that enables the use of AI/ML techniques and methods for the smart operation of cloud-edge environments.	**[KPI1.2]** New AI-enabled orchestration demonstrated through Validation Use Cases.													4			4	4			
Adaptation by enabling the elasticity of the infrastructure with automated provisioning of edge PoPs.	**[KPI1.3]** Full provision of edge PoPs in "1-click" and in less than 10 minutes.											3									
Deployment and operation of edge cloud environments incorporating infrastructure resources across the whole cloud-edge continuum.	**[KPI1.4]** Deployment of a geo-distributed Testbed with resources from on-premises datacenters, cloud provider, edge provider, 5G provider, and on-premises far edge.										3										
Ability to operate both the infrastructure layer and the virtualized resources using a single pane of glass.	**[KPI1.5]** Unified GUI, CLI, and API able to manage all resource layers across the continuum.										3	3									
Seamless access for end users.	**[KPI2.1]** Meet experience level agreements with dynamic needs without intervention of the user.						3									3	4	4			
Application developers can define the execution environment for the FaaS Runtimes.	**[KPI2.2]** Definition of Execution Context, Base Image, Communication Patterns, and Deployment Requirements for performance, cost, security and energy requirements.	3	3	4	4			3	3												
Developers can change the deployment requirements of the FaaS Runtime according to execution flow.	**[KPI2.3]** Migration across edge PoPs to meet application needs.															3					
Early and continuous security assurance of FaaS Runtimes.	**[KPI2.4]** Security processes and controls automated and integrated into the runtime lifecycle following a DevSecOps approach.					3		3	3												
Efficient orchestration of serverless workloads across the Continuum.	**[KPI3.1]** Backward compatibility with virtualization tech (e.g. KVM, Firecracker) and containers (e.g. K8s).										3										
Able to migrate workloads in geo-distributed edge clouds, combining edge/cloud resources.	**[KPI3.2]** Migrate workloads across the Continuum with minimal downtime.											3				3					
Intelligent self-adaptation of FaaS Runtime to changes in application behavior and data variability.	**[KPI3.3]** Automatic scale up/down of the microVM running the FaaS Runtime.													4		3					
Use of confidential and trusted computing for secure computation of private data at the edge/cloud.	**[KPI3.4]** Use case demonstrating isolation of sensitive data as it's being processed.														3					4	4
Dynamic load balancing to adapt to changes in application needs and cloud-edge infrastructure.	**[KPI4.1]** Scalable, multi-variate AI-enabled modeling and optimization techniques.																3	3			
Intelligent adaptation of cloud-edge continuum management.	**[KPI4.2]** New methods to create and retrain ML-based predictive behavioral forecasts for all major management considerations faced by cloud-edge continuum environments.																4				
Hierarchical multi-constraint resource management layer to minimize environmental impact.	**[KPI4.3]** Reduction of at least 10% energy consumption compared with manual edge deployment.																4	4			
Orchestration mechanisms to enforce coherent global security and governance policies.	**[KPI4.4]** Able to automate a European security policy on a multi-provider edge deployment.																		4		

Table 12.2. Expected contribution of each Software Requirement towards meeting the Project's global KPIs, by Milestone.

13. Work Done in Third Research & Innovation Cycle (M16-M21)

During the Fourth Research & Innovation Cycle (M16-M21) the Project has mostly focused on designing the new architecture for the COGNIT framework due to some limitations identified in the previous innovation cycle, and on those software requirements needed to achieve a first prototype of the new framework.

The work covered by the first release of the new COGNIT Framework include:

- The **Device Client SDK** provides the interface to the COGNIT Frontend for the authentication of the devices, the upload of the application requirements, the query to get a suitable Edge Cluster and the interface to the Edge Cluster Frontend to offload from the device the execution of Python functions.
- The **COGNIT Frontend** provides a REST API to communicate with the Device Client providing the following capabilities: authentication and authorization token generation, storing application requirements, uploading function from the devices and interface with the AI-Enabled Orchestrator for querying the most suitable Edge Cluster for the devices.
- An automated image building system for the **Serverless Runtimes**, based on SUSE Open Build Service, has been set up in order to handle all the workflow of the image generation and deployment within the COGNIT Platform. SUSE will contribute to this task with its open source image building tool
- The **Edge Cluster Frontend** provides a REST API that allows devices to offload execution of functions.
- **ML/AI models** have been evaluated, trained and validated for two main purposes: application workload characterization and energy-aware scheduling.
- The ML/AI models have been integrated in the **AI-Enabled Orchestrator** for the orchestration of Serverless Runtimes within an Edge Cluster in order to optimise the energy consumption.

During this cycle, the tool **OpsForge** has been upgraded in order to allow the user to deploy the main components of the new COGNIT Framework: COGNIT Frontend, Cloud Edge-Manager and Edge Cluster Frontend in order to setup a Cloud-Edge infrastructure to manage Serverless Runtimes needed to offload the function execution from the devices.

These features have been developed in a coordinated way between WP3 and WP4. The new software components and extensions to meet the software requirements have been specified, developed, and tested within the work package WP3 and WP4. The integration of new functionalities has to be verified and demonstrated within WP5 and reported in Deliverable D5.4.

The following section summarises, per component, the work that has been done as part of the Third Research & Innovation Cycle (M16-M21), including the completed and pending tasks associated with each of the software requirements that have been active during the cycle.

13.1. Device Client

SR1.1 Interface with COGNIT Frontend
Status: IN PROGRESS
Completed Tasks: The current implementation of the Device Client enables the device to communicate with the COGNIT Frontend for authentication, for uploading application requirements, for uploading functions, and for requesting the Edge Cluster where to offload the execution of functions.
Pending Tasks: Add method for uploading data to the COGNIT Framework.

SR1.2 Interface with Edge Cluster
Status: IN PROGRESS
Completed Tasks: The current Device Client implementation supports executing functions on the Edge Cluster through the client python module methods.
Pending Tasks: Add a method for uploading data to the Edge Cluster.

SR1.3 Programming languages
Status: IN PROGRESS
Completed Tasks: The Device Client supports Python programming languages. The Device Client is implemented as a Python module.
Pending Tasks: Add C support.

SR1.4 Low memory footprint for constrained devices
Status: NOT STARTED YET

SR1.5 Security
Status: IN PROGRESS
Completed Tasks: The device client authenticates with the COGNIT Frontend and then it uses the token returned by the COGNIT Frontend for subsequent requests to the COGNIT Frontend and to the Edge Cluster Frontend.
Pending Tasks: Add support for secure channel communication with the COGNIT Frontend and Edge Cluster Frontend.

SR1.6 Collecting Latency Measurements
Status: NOT STARTED YET

13.2. COGNIT Frontend

SR2.1 COGNIT Frontend
Status: IN PROGRESS
Completed Tasks: The COGNIT Frontend allows devices to authenticate and returns a Biscuit token once the device is authenticated. It allows the device to upload application requirements, to upload functions, and to query for the optimal Edge Cluster to offload the execution of functions.
Pending Tasks: Add an endpoint for uploading data from the device.

13.3. Edge Cluster

SR3.1 Edge Cluster Frontend
Status: IN PROGRESS
Completed Tasks:

The Edge Cluster Frontend allows devices to offload the execution of the function in one of the Serverless Runtimes available in the Edge Cluster.

Pending Tasks:
Add an endpoint for uploading data from the device to the Edge Cluster.

SR3.2 Secure and Trusted Serverless Runtimes

Status: COMPLETED

Completed Tasks:
The Serverless Runtime exposes a REST API interface for executing Python functions uploaded to the COGNIT Frontend by the Device Client.
An image has been built in the Cloud-Manager for the deployment of the Serverless Runtime. The images (from the base one) with specific libraries (e.g. Python libraries for image segmentation) have been built according to the need of the different Use Cases. The images are ready to incorporate the Trusted Computing Execution Environment once it is available in the COGNIT Framework.

13.4. Cloud-Edge Manager

SR4.1 Provider Catalogue

Status: NOT STARTED YET

SR4.2 Edge Cluster Provisioning

Status: NOT STARTED YET

SR4.3 Serverless Runtime Deployment

Status: COMPLETED

Completed Tasks:
A template has been defined for the deployment of the Serverless Runtime. The Cloud-Edge Manager API to manage the lifecycle of Serverless Runtimes is based on OpenNebula Oneflow.

SR4.4 Metrics, Monitoring, Auditing

Status: IN PROGRESS

Completed Tasks:
The OpenNebula monitoring system has been enhanced with a component for collecting energy metrics from Hosts and Serverless Runtimes based on the Scaphandre tool. Metrics are pushed to the Prometheus server integrated in OpenNebula. Hosts and Device locations are collected and integrated.

Pending Tasks:
Gather latency metrics from the devices.
Implementation of intrusion and anomaly detection of the different Edge Cluster entities and Serverless Runtimes.

SR4.5 Authentication & Authorization

Status: IN PROGRESS

Completed Tasks:
Implementation of delegation mechanisms in the COGNIT Frontend for authentication and authorization. The COGNIT Frontend delegates the credentials provided by the Device Client to the Cloud Edge Manager.

Pending Tasks:
Implementation of an IAM mechanism based on Biscuit tokens that allows for fine-grained access control of FaaS functions.
Implementation of secure communication between the Device Client and the COGNIT Frontend and the Edge Cluster Frontend.

SR4.6 Plan Executor

Status: NOT STARTED YET

13.5. AI-Enabled Orchestrator

SR5.1 Building Learning Models

Status: IN PROGRESS

Completed Tasks:
An analysis of the state-of-the-art of ML/AI approaches for modelling the Cloud-Edge Continuum has been carried out. Different approaches have been selected and considered as potential candidates for providing the capability to automate the "optimal" placement of Serverless Runtimes on the distributed Cloud-Edge Continuum according to application and resource availability. Several models for the AI-Enabled Orchestrator have been trained and validated.

Pending Tasks:
Create and validate multiple AI/ML models under different categories such as regression, classification, prediction, and federated learning as well as evaluate them for diverse orchestration tasks.

SR5.2 Smart Management of Cloud-Edge Resources

Status: IN PROGRESS

Completed Tasks:
Algorithms for interference and energy-aware optimization.

Pending Tasks:
Features such as smart scheduling, scaling, migration of serverless runtimes, and orchestration of Edge Clusters in order to meet the workload requirements
Integration with Cloud-Edge Manager and Plan Executor.

13.6. Secure and Trusted Execution of Computing Environments

SR6.1 Advanced Access Control

Status: IN PROGRESS

Completed Tasks:
Architecture analysis and technology survey for integrating a policy-based access control service in the COGNIT Framework
Integration of an advanced authorization mechanism based on tokens.

Pending Tasks: Implementation of security policies based on the authentication mechanism**.**

SR6.2 Confidential Computing

Status: IN PROGRESS

Completed Tasks:
Architecture analysis and technology survey on using Confidential computing approach in the COGNIT Framework.

Pending Tasks:
Integration and deployment of the confidential computing capabilities within the COGNIT Framework.

SR6.3 Federated Learning

Status: NOT STARTED YET

14. Priorities for Forth Research & Innovation Cycle (M22-M27)

During the Third Research & Innovation Cycle (M16-M21), and as part of the research and development tasks undertaken by the Use Cases, we have focused on the design and implementation of an improved version of the COGNIT Framework due to a few limitations that have been identified in the previous COGNIT Architecture.

During the Fourth Research & Innovation Cycle (M21-M27), the Project will focus on continuing the development of the new software requirements that have been identified in this cycle.

Regarding the **Device Client**, the priority for the next cycle is to increase the user experience by improving the user API. Furthermore, the Device Client will support the uploading of data to the COGNIT Frontend and to the Edge Cluster Frontend. Finally, the Device Runtime will collect and push latency measurements from the device to the Edge Cluster that is doing the actual offloading.

The **Edge Cluster Frontend** will be extended to support uploading data to the Edge Cluster that can be used by the functions that are executed within the same Edge Cluster.

Regarding the **Cloud-Edge Manager** component, in the Fourth Research & Innovation Cycle, most of the work will be focused on the ability to dynamically create new edge locations automatically to better satisfy scheduling policies and application requirements. On the monitoring side, metrics related to the latency with respect to the Device Client will be stored in order to be used by the AI-Enabled Orchestrator to anticipate the creation of Serverless Runtime or Edge Clusters in order to minimise the cold start and to satisfy latency requirements.

In the Fourth Research & Innovation Cycle, the **AI-Enabled Orchestrator** will focus on the development of multiple features that enable smart and optimised downstream orchestration tasks. This includes interference and energy-aware placements, energy and latency-aware placements. A model is selected from the model repository according to the specific task. An ML server will hold the model repository, which is being developed and able to include more models according to the requirements of tasks (e.g. fine-grained classification and prediction of workloads, on-demand resource prediction, multi-objective resource optimization, such as, maximising green energy usage and minimising interference, and energy availability prediction). These tasks will utilise the metrics and resources provided by the Cloud-Edge Manager in order to improve learning and smart decisions.

15. Conclusions and Next Steps

The initial version of the COGNIT Framework Architecture (Deliverable D2.1), released in M3, identified and analysed the main sovereignty, sustainability, interoperability, and security requirements, as well as the user requirements, derived from the European context and from the Project's specific Use Cases. They are expected to guide the research and development of the project, having played a central role in the initial definition of the original COGNIT Architecture. From those global and user requirements, a list of software requirements and functional gaps to be implemented by the components of the COGNIT Framework were identified, followed by a definition of the methodology and scenarios required to verify their fulfilment and applicability in the Project's Use Cases.

In the first and second incremental versions of D2.1 (Deliverable D2.2, Deliverable 2.3) the open source software components and extensions needed to meet the Software Requirements were specified and developed within the Work Packages WP3 and WP4, with these new functionalities being tested, verified, and demonstrated as part of the Use Cases in their respective associated tasks in WP5.

Deliverable D2.4 starts from a few limitations that have been identified in the previous Deliverable D2.3, that drove us to review and modify the software requirements of the COGNIT Framework. Therefore, Deliverable D2.4 provides not an incremental report but rather a comprehensive overview of the Project's global development status, offers a summary of the work done in the Third Research & Innovation Cycle (M16-M21), and identifies the priorities for the Fourth Research & Innovation Cycle (M22-M27). Additional incremental versions of this report will be released at the end of the next research and innovation cycles (i.e. M27 and M33).