SovereignEDGE.EU COGNIT

A Cognitive Serverless Framework for the Cloud-Edge Continuum

D2.2 COGNIT Framework - Architecture - b

Version 1.0

31 October 2023

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 incremental 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 First Research & Innovation Cycle (M4-M9), and identify the priorities for the Second Research & Innovation Cycle (M10-M15).

Copyright © 2023 SovereignEdge.Cognit. All rights reserved.

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

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

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:	30/09/2023
Actual Date of Delivery:	31/10/2023
Lead Author:	Marco Mancini (OpenNebula) & Constantino Vázquez (OpenNebula)
Authors:	Michael Abdou (OpenNebula), Monowar Bhuyan (UMU), Dominik Bocheński (Atende), Aritz Brosa (Ikerlan), Malik Bouhou (CETIC), Idoia de la Iglesia (Ikerlan), Sébastien Dupont (CETIC), Agnieszka Frąc (Atende), Grzegorz Gil (Atende), Torsten Hallmann (SUSE), Joan Iglesias (ACISA), Tomasz Korniluk (Phoenix), Johan Kristiansson (RISE), Antonio Lalaguna (ACISA), Martxel Lasa (Ikerlan), Ignacio M. Llorente (OpenNebula), Alberto P. Martí (OpenNebula), Philippe Massonet (CETIC), Nikolaos Matskanis (CETIC), Behnam Ojaghi (ACISA), Daniel Olsson (RISE), Goiuri Peralta (Ikerlan), Samuel Pérez (Ikerlan), Holger Pfister (SUSE), Tomasz Piasecki (Atende), Francesco Renzi (Nature4.0), Juan José Ruiz (ACISA), Kaja Swat (Phoenix), Paul Townend (UMU), Iván Valdés (Ikerlan), Thomas Ohlson Timoudas (RISE), Riccardo Valentini (Nature4.0).
Status:	Submitted

Document History

Version	Issue Date	Status[1]	Content and changes
0.1	20/10/2023	Draft	Initial Draft
0.2	27/10/2023	Peer-Reviewed	Reviewed Draft
1.0	31/10/2023	Submitted	Final Version

Peer Review History

Version	Peer Review Date	Reviewed By
0.1	27/10/2023	Ignacio M. Llorente (OpenNebula)
0.1	27/10/2023	Paul Townend (UMU)

Summary of Changes from Previous Versions

First Version of Deliverable D2.2

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

Executive Summary

Deliverable D2.2, released at the end of the First Research & Innovation Cycle (M9), is the first incremental version of the COGNIT Framework Architecture report in WP2 “Adaptive Cloud-Edge Serverless Framework Architecture”. This report provides an overview of the Project's overall development status and offers a summary of the work done in the First Research & Innovation Cycle (M4-M9).

During the First Research & Innovation Cycle (M4-M9), and in line with the expected contribution of each Software Requirement towards meeting the Project’s Milestones and global KPIs (as defined in Deliverable D2.1), the Project has focused its efforts on launching initial research and development activities across all the main components of the COGNIT Architecture, namely:

- The **Device Client**, which allows the device to offload functions to the COGNIT Framework based on a series of requirements.
- The **Serverless Runtime**, in charge of executing the functions offloaded by the device and storing data uploaded by the device or coming from an external storage system.
- The **Provisioning Engine**, responsible for managing the lifecycle of the Serverless Runtimes.
- The **Cloud-Edge Manager**, responsible for managing the Serverless Runtime according to the deployment plan provided by the AI-Enabled Orchestrator.
- The **AI-Enabled Orchestrator**, the component that, according to the device’s requirements and available infrastructure, schedules the Serverless Runtime across the cloud-edge continuum.

In connection with those components, the Project has delivered progress specifically in those software requirements needed to achieve Milestone 2 in M15, and to provide a basic set of functionalities for the Use Cases to be able to launch their own research and development activities in the next cycle (M10-M15).

The features of the first internal release of the COGNIT Framework include:

- First version of the Device Client Python SDK that provides the interface to the Provisioning Engine to create and manage Serverless Runtimes by the device and the interface to the Serverless Runtime to offload from the device the execution of Python functions.
- First version of the Serverless Runtime Appliance that contains the FaaS runtime component needed for the execution of Python functions.
- First version of the Provisioning Engine that provides a REST API to create, read and delete Serverless Runtimes.
- First version of the AI-Enabled Orchestrator for the smart placement of Serverless Runtimes across the resources provisioned and managed by the Cloud-Edge Manager, based on a scheduler able to delegate placement decisions to an external

AI-enabled module implementing algorithms according to specific use cases or business logics.

Apart from the overview provided in this document (D2.2), specific research and development activities performed in WP3 “Distributed FaaS Model for Edge Application Development” (related to the Device Client, the Serverless Runtime, the Provisioning Engine, and the Secure and Trusted Execution of Computing Environments) are described in detail in reports D3.1 “COGNIT FaaS Model - Scientific Report” and D3.6 “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 D4.1 “COGNIT Serverless Platform - Scientific Report” and D4.6 “COGNIT Serverless Platform - Software Source”; details about the COGNIT software integration and the verification scenarios of each software requirements can be found in report D5.2 “Use Cases - Scientific Report”.

The present incremental report (Deliverable D2.2) includes a list of research and development priorities for the Second Research & Innovation Cycle (M10-M15).

This deliverable has been released at the end of the First Research & Innovation Cycle (M9), and will be updated with incremental releases at the end of each research and innovation cycle (i.e. M15, M21, M27, M33).

Table of Contents

Abbreviations and Acronyms

ACL	Access Control List
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
CC	Confidential Computing
CCA	Confidential Computing Architecture
CD	Continuous Delivery (Deployment)
CI	Continuous Integration
CIA	Confidentiality, Integrity and Availability
CRA	Cyber Resilient Act
DaaS	Data as a Service
DB	Database
FaaS	Function as a Service
FLOPS	FLoating point Operations Per Second
GPU	Graphics Processing Unit
GDPR	General Data Protection Regulation
HTTP	Hypertext Transfer Protocol
IAM	Identity and Access Management system
IOPS	I/O Operations Per Second
IP	Internet Protocol
IoT	Internet of Things
JSON	Javascript Object Notation
LDAP	Lightweight Directory Access Protocol
ML	Machine Learning
NIS	Network and Information Security
NIST	National Institute of Standards and Technology
OIDC	OpenID Connect
OS	Operating System
QoS	Quality of Service
REST	Representational State Transfer
RBAC	Role-Based Access Control
S3	Simple Storage Service
SDK	Software Development Kit
SEV	Secure Encrypted Virtualization

SGX	Software Guard eXtension
SLA	Service Level Agreement
SQL	Structured Query Language
TEE	Trusted Execution Environments
TLS	Transport Layer Security
TPM	Trusted Platform Module
TPU	Tensor Processing Unit
VM	Virtual Machine
YAML	Yaml Ain’t a markup language

1. Introduction

The initial version of the COGNIT Framework Architecture report (Deliverable D2.1), released in M3, included a summary of Use Cases 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 aim of this incremental version (Deliverable D2.2) is to provide an overview of the Project's overall development status, offer a summary of the work done in the First Research & Innovation Cycle (M4-M9), and identify the priorities for the Second Research & Innovation Cycle (M10-M15). An incremental version of this report will be released at the end of each research and innovation cycle (i.e. M15, M21, M27, M33).

D2.2 is a living document that is composed of an introduction and three main sections:

- Section 1 provides an overview of the Project's overall development status, including a list of new user requirements identified during the First Research & Innovation Cycle (M4-M9), an update on software requirements (e.g. minor amendments to existing software requirements, those whose scope might have been expanded, or new ones), the current status of each Software Requirement towards completion, and an update on the expected contribution of each of them towards meeting the Project's Milestones and global KPIs.
- Section 2 provides an up-to-date overview of the readiness and maturity level of each component of the COGNIT Framework Architecture, and describes which (and how) specific software requirements have been addressed during the First Innovation Cycle (M4-M9).
- Section 3 provides a brief summary of research and development priorities for the Second Research & Innovation Cycle (M10-M15).

The document ends with a conclusion section.

2. Overall Development Status

This section provides an overview of the Project's overall development status.

2.1. Update of User Requirements

New User Requirements

As defined in Deliverable D5.2—and as an extension to Deliverable D5.1—a number of additional user requirements have been identified during the First Research & Innovation Cycle (M4-M9):

Id	Description	Source
UR0.9	IAM system integration for high granularity authentication and user management for device clients, provisioning engine and serverless runtime.	All
UR0.10	Push mechanism to inform about status or events from Provisioning Engine and Serverless runtime back to the requestor device client.	All

Table 2.1. New common user requirements identified in M4-M9.

2.2. Update of Software Requirements

On the basis of those software requirements already defined in Deliverable D2.1, a number of changes have taken place during the First Research & Innovation Cycle (M4-M9):

Minor amendments to existing Software Requirements

These changes are intended to correct typos or errata in the original versions:

SR4.2 Edge Cluster Provisioning

Description: Provision Edge Clusters as a set of software-defined compute, network, storage on any cloud/edge location available in the Provider Catalogue.

- A provisioning template for Edge Cluster shall be defined and provided as an input by the AI-Enabled Orchestrator.
- The **Cloud-Edge Manager** ~~Provisioning Engine~~ shall implement an API to provision/update/scale/migrate Edge Clusters.

SR4.3 Serverless Runtime Deployment

Description: Deploy Serverless Runtime as Virtualized Workloads (e.g. Containers or VMs/microVMs) on the cloud-edge infrastructure.

- A deployment template specifying the different components of the Serverless Runtime (i.e. FaaS Runtime & DaaS Runtimes) and their dependencies shall be defined and provided as an input by the Provisioning Engine.
- The **Cloud-Edge Manager** ~~Provisioning Engine~~ shall provide an API to deploy/update/scale/migrate Serverless Runtimes.

Existing Software Requirements whose scope has been modified

The scope of these software requirements has been expanded in order to cover the new User Requirement UR0.9:

SR3.1 Provisioning Interface for the Device to manage Serverless Runtimes

Description: Provide an interface to the Device asking for a Serverless Runtime to offload functions and data transfer on any resource of the cloud-edge continuum.

- A document with requirements and attributes for the Provisioning Interface shall be defined and provided as input by the Device.
- Provisioning interface shall implement a REST API to create/read/update/delete Serverless Runtimes.
- Provisioning interface shall provide means of secure communication with the Device.
- Provisioning interface shall provide means of secure communication with the Cloud-Edge Manager and the AI-Enabled Orchestrator.
- **[NEW]** An Identity and Access Management (IAM) mechanism shall be integrated.

SR4.3 Serverless Runtime Deployment

Description: Deploy Serverless Runtime as Virtualized Workloads (e.g. Containers or VMs/microVMs) on the cloud-edge infrastructure.

- A deployment template specifying the different components of the Serverless Runtime (i.e. FaaS Runtime & DaaS Runtimes) and their dependencies shall be defined and provided as an input by the Provisioning Engine.
- The Cloud-Edge Manager shall provide an API to deploy/update/scale/migrate Serverless Runtimes.
- **[NEW]** An Identity and Access Management (IAM) mechanism shall be integrated.

SR4.5 Authentication & Authorization

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

- Provisioning Engine shall be able to use mechanisms for delegation of authentication and authorization.
- The device shall be able to use x509 certificates for requests to the Cloud-Edge Manager through the Provisioning Engine.
- **[NEW]** An Identity and Access Management (IAM) mechanism shall be integrated.

The scope of these software requirements has been expanded in order to cover the new User Requirement UR0.10:

SR1.1 Interface with Provisioning Engine

Description: Implementation of the communication with the Provisioning Engine.

- The Device Client shall be able to ask for the creation of a Serverless Runtime with specific requirements to the Provisioning Engine.
- The Device Client shall be able to ask for information about a Serverless Runtime (read) to the Provisioning Engine.
- The Device Client shall be able to ask for deletion of a Serverless Runtime to the Provisioning Engine.
- The Device Client shall be able to ask for an update of the requirements of a Serverless Runtime to the Provisioning Engine.
- **[NEW]** The Python version of the Device Client shall implement an async client (e.g. based on WebSocket) to receive events from the Provisioning Engine.

SR1.2 Interface with Serverless Runtime

Description: Implementation of the communication of with the Serverless Runtime

- The Device Client shall be able to upload data from the device to the Serverless Runtime.
- The Device Client shall be able to upload a function to the Serverless Runtime.
- The Device Client shall be able to request for executing a function to the Serverless Runtime.
- The Device Client shall be able to request to transfer data from external resources to the Serverless Runtime.
- **[NEW]** The Python version of the Device Client shall implement an async client (e.g. based on WebSocket) to receive events from the Serverless Runtime.

SR2.1 Secure and Trusted FaaS Runtimes

Description: Automated building of secure and trusted images (vulnerability scans, security assessment) related to different flavours of FaaS Runtimes.

- Cloud-Edge Manager shall provide a base FaaS image that exposes a REST API interface to the device for building and executing functions, provides a secure channel for the communication with the Device and pushes metrics to the Cloud-Edge Manager (for monitoring and auditing).

- Cloud-Edge Manager shall provide FaaS images (from the base one) adding specific libraries (e.g. python libraries for image segmentation) needed for the execution of functions according to the need of the different Use Cases.
- **[NEW]** The Serverless Runtime shall implement a mechanism (e.g. based on WebSocket) to send async events to the Device Client.

SR3.1 Provisioning Interface for the Device to manage Serverless Runtimes

Description: Provide an interface to the Device asking for a Serverless Runtime to offload functions and data transfer on any resource of the cloud-edge continuum.

- A document with requirements and attributes for the Provisioning Interface shall be defined and provided as input by the Device.
- Provisioning interface shall implement a REST API to create/read/update/delete Serverless Runtimes.
- Provisioning interface shall provide means of secure communication with the Device.
- Provisioning interface shall provide means of secure communication with the Cloud-Edge Manager and the AI-Enabled Orchestrator.
- **[NEW]** The Provisioning Engine shall implement a mechanism (e.g. based on WebSocket) to send async events to the Device Client.

New Software Requirements

New Software Requirement aimed at clarifying the different tasks involved in implementing the scheduler subcomponent of the AI-Enabled Orchestrator:

SR5.3 Scheduling Mechanisms

Description: Implement a scheduler that will place the Serverless Runtimes on the Edge-Clusters resources according to the deployment plan provided by the AI-Enabled Orchestrator.

- The scheduler will provide the AI-Enabled Orchestrator with a filtered list of suitable resources (hosts) for the Serverless Runtimes that need to be deployed
- The scheduler will interact with the REST API provided by the AI-Enabled Orchestrator to get placement plans for Serverless Runtime that are in a pending status
- The scheduler will interact with the REST API provided by the AI-Enabled Orchestrator to get updated placement plans for Serverless Runtime that are already scheduled/running

2.3. 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, ↻ for activities in progress, and ✓ for completed activities:

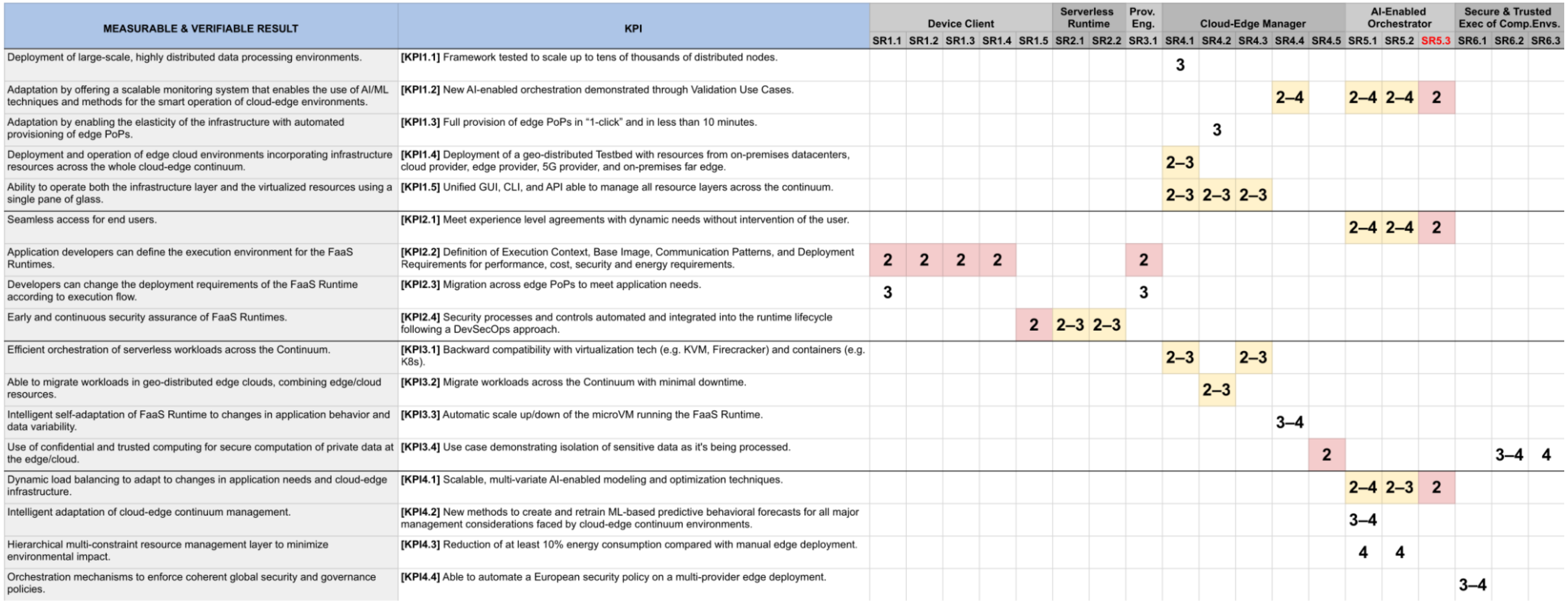
Software Requirements	**Cycle 1** (M4-M9)	**Cycle 2** (M10-M15)	**Cycle 3** (M16-M21)	**Cycle 4** (M22-M27)	**Cycle 5** (M28-M33)
Device Client					
SR1.1 Interface with Provisioning Engine	↻	–	–	–	–
SR1.2 Interface with Serverless Runtime	↻	–	–	–	–
SR1.3 Programming languages	↻	–	–	–	–
SR1.4 Low memory footprint for constrained devices	–	–	–	–	–
SR1.5 Security	–	–	–	–	–
Serverless Runtime					
SR2.1 Secure and Trusted FaaS Runtimes	↻	–	–	–	–
SR2.2 Secure and Trusted DaaS Runtimes	–	–	–	–	–
Provisioning Engine					
SR3.1 Provisioning Interface for the Device to manage 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	↻	–	–	–	–
AI-Enabled Orchestrator					
SR5.1 Building Learning Model	↻	–	–	–	–
SR5.2 Smart Deployment of Serverless Runtimes	↻	–	–	–	–
SR5.3 Scheduling Mechanisms	↻	–	–	–	–
Secure and Trusted Execution of Computing Environments					
SR6.1 Advanced Access Control	↻	–	–	–	–
SR6.2 Confidential Computing	↻	–	–	–	–
SR6.3 Federated Learning	–	–	–	–	–

Table 2.2. Current status of each Software Requirement towards completion, per research & innovation cycle (new SRs in red).

2.3. Global KPIs Progress

The table below shows in which Milestone each Software Requirement is expected to meet the Project's global KPIs, including activities expected to be completed by Milestone 2 and those that are already active but will be completed later on during the Project:



MEASURABLE & VERIFIABLE RESULT	KPI	Device Client					Serverless Runtime		Prov. Eng.	Cloud-Edge Manager					AI-Enabled Orchestrator			Secure & Trusted Exec of Comp.Envs.		
		SR1.1	SR1.2	SR1.3	SR1.4	SR1.5	SR2.1	SR2.2	SR3.1	SR4.1	SR4.2	SR4.3	SR4.4	SR4.5	SR5.1	SR5.2	SR5.3	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.												2–4		2–4	2–4	2			
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.									2–3										
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.									2–3	2–3	2–3								
Seamless access for end users.	**[KPI2.1]** Meet experience level agreements with dynamic needs without intervention of the user.														2–4	2–4	2			
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.	2	2	2	2				2											
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							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.					2	2–3	2–3												
Efficient orchestration of serverless workloads across the Continuum.	**[KPI3.1]** Backward compatibility with virtualization tech (e.g. KVM, Firecracker) and containers (e.g. K8s).									2–3		2–3								
Able to migrate workloads in geo-distributed edge clouds, combining edge/cloud resources.	**[KPI3.2]** Migrate workloads across the Continuum with minimal downtime.										2–3									
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.												3–4							
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.													2					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.														2–4	2–3	2			
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.														3–4					
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.																	3–4		

Table 2.3. Updated expected contribution of each Software Requirement towards meeting the Project's Milestones and global KPIs (new SRs in red).

3. Work Done in First Research & Innovation Cycle (M4-M9)

During the First Research & Innovation Cycle (M4-M9) the Project has mostly focused on those software requirements needed to achieve Milestone 2 by M15 and to provide a basic set of functionalities for the Use Cases to be able to launch their own research and development activities in the next cycle (M10-M15), especially the base functionality needed for the deployment of a Serverless Runtime so that the Device Client is able to offload computational tasks (i.e. Python functions) to the Cloud-Edge Continuum.

Apart from working on an initial implementation of functionalities in all COGNIT components (i.e. Device Client, Serverless Runtime, Provisioning Engine, Cloud-Edge Manager, and AI-Enabled Orchestrator), other tasks have involved:

- A preliminary analysis of requirements for the secure and trusted execution of computing environments and an initial **threat assessment** on the COGNIT Framework's architecture, including external dependencies, access points, and definition of trust levels.
- Research on energy efficiency optimisation in the multi-provider Cloud-Edge Continuum and development of a **first prototype** exploring the integration aspects of the COGNIT Framework when it comes to prioritising the deployment of Serverless Runtimes on edge nodes or virtualization hosts using renewable energy.

The features of the first internal release of the COGNIT Framework include:

- First version of the **Device Client** Python SDK that provides the interface to the Provisioning Engine to create and manage Serverless Runtimes by the device and the interface to the Serverless Runtime to offload from the device the execution of Python functions.
- First version of the **Provisioning Engine** that provides a REST API to create, read and delete Serverless Runtimes.
- First version of the **Serverless Runtime** Appliance that contains the FaaS runtime component needed for the execution of Python functions.
- First version of the **AI-Enabled Orchestrator** for the smart placement of Serverless Runtimes across the resources provisioned and managed by the **Cloud-Edge Manager**, based on a scheduler able to delegate placement decisions to an external AI-enabled module implementing algorithms according to specific use cases or business logics.

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 been verified and demonstrated within WP5 and reported in Deliverable D5.2.

The following section summarises, per component, the work that has been done as part of the First Research & Innovation Cycle (M4-M9), including the completed and pending tasks associated with each of the software requirements that have been active during the cycle:

3.1. Device Client

SR1.1 Interface with Provisioning Engine
Status: IN PROGRESS
Completed Tasks: The current implementation of the Device Client enables the user to communicate with the Provisioning Engine to request, retrieve and delete a Serverless Runtime instance. Currently the client only supports requesting Serverless Runtime using the green energy usage parameter value for the scheduling.
Pending Tasks: Add support on the Device Client to update the requirements of the Serverless Runtime. Add support on the Device Client to configure additional scheduling policies. Upgrade the current polling based communication to an event based (e.g. based on WebSocket) on the Python version of the client.

SR1.2 Interface with Serverless Runtime
Status: IN PROGRESS
Completed Tasks: The current Device Client implementation supports uploading and executing functions on the Serverless Runtime through the client python module methods. From the user's perspective, the result of the execution is retrieved in the same way as if it were executed locally.
Pending Tasks: Secure communications with the Serverless Runtime. Add support on Device client to upload data from the device to the Serverless Runtime and to transfer data from external resources to the Serverless Runtime. Upgrade the current polling based communication to an event based (e.g. based on WebSocket) on the Python version of the client.

SR1.3 Programming languages
Status: IN PROGRESS
Completed Tasks: Currently the Device Client supports the Python programming language. The Device Client is currently implemented as a Python module. This implementation enables the users to request and delete Serverless Runtimes based on a “green energy usage”

parameter. Once the Serverless Runtime is ready, the Device Client can offload the execution of Python functions to the Serverless Runtime at runtime.

Pending Tasks:
Add support to enable the clients to update the requirements of an existing Serverless Runtime.
Implement a Device Client library in C, adding support for uploading and executing C functions.

3.2. Serverless Runtime

SR2.1 Secure and Trusted FaaS Runtime

Status: IN PROGRESS

Completed Tasks:
A first version of the FaaS Runtime component of the Serverless Runtime has been implemented; it exposes a REST API interface for executing Python functions uploaded by the Device Client.
A FaaS image has been built in the Cloud-Manager for the deployment of the Serverless Runtime.

Pending Tasks:
Implementation of mechanisms for the secure communication between the Device and the FaaS Runtime.
Build FaaS images (from the base one) adding specific libraries (e.g. Python libraries for image segmentation) needed for the execution of functions according to the need of the different Use Cases.

3.3. Provisioning Engine

SR3.1 Provisioning Interface for the Device to manage Serverless Runtimes

Status: IN PROGRESS

Completed Tasks:
A JSON document with requirements and attributes for the Provisioning Interface has been defined that can be used as input from the Device Client. A first version of the Provisioning Engine has been implemented as a REST API to create/read/delete Serverless Runtimes.

Pending Tasks:
Implementation of secure mechanisms for the communication between the Provisioning Engine and the Device, and between the Provisioning and the Cloud-Edge

Manager. Implementation of the “update” REST API operation to modify the requirements of an existing Serverless Runtime. Integrate with Identity and Access Management (IAM) mechanism The Provisioning Engine implements a mechanism (e.g. based on WebSocket) to send async events to the Device Client.

3.4. Cloud-Edge Manager

SR4.3 Serverless Runtime Deployment
Status: IN PROGRESS
Completed Tasks: A template has been defined for the deployment of the FaaS component of the Serverless Runtime. Provisioning Engine communicates with the Cloud-Edge Manager to create, read, and terminate Serverless Runtimes. The Cloud-Edge Manager API to manage the life-cycle of Serverless Runtimes is based on OpenNebula Oneflow.
Pending Tasks: Definition of the deployment template specifying the different components of the Serverless Runtime (i.e. FaaS Runtime & DaaS Runtimes) and their dependencies. Cloud-Edge Manager REST API to scale and migrate Serverless Runtimes. Integrate with Identity and Access Management (IAM) mechanism

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
Pending Tasks: Gather location and latency metrics. 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 Provisioning Engine for authentication and authorization. The Provisioning Engine delegates the credentials provided by the Device Client to the Cloud Edge Manager.
Pending Tasks: Implement an IAM mechanism based on JWT that allows for fine-grained access control of FaaS and DaaS functions. Implementation of secure communication between the Device Client and the Provisioning Engine and the Serverless Runtime.

3.5. AI-Enabled Orchestrator

SR5.1 Building Learning Model
Status: IN PROGRESS
Completed Tasks: An analysis of the state-of-the-art of ML/AI approaches for modeling 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 Cloud-Edge Continuum according to application and resource requirements.
Pending Tasks: Create a repository with AI/ML models trained/validated for different tasks such as intelligent orchestration, workload characterization, etc.

SR5.2 Smart Deployment of Serverless Runtimes
Status: IN PROGRESS
Completed Tasks: A first cycle implementation of the AI-Enabled Orchestrator has been completed by enabling an improved FFD (First-Fit Decreasing) algorithm that provides recommended VMs with associated hosts. It exposes a REST API to interact with OpenNebula extended scheduler that accepts requests and responds to the scheduler for performing the placement task.
Pending Tasks: Integrating AI/ML models trained and validated in SR5.1 within the AI-Enabled Orchestrator to provide the Cloud-Edge Manager with the capability to do the optimal placements of Serverless Runtimes according to the dynamic application and resource requirements.

[NEW] SR5.3 Scheduling Mechanisms
Status: IN PROGRESS
Completed Tasks: OpenNebula Scheduler has been refactored to support an external module for the placement of pending VMs (related to the Serverless Runtimes). A REST API has been defined to be implemented by the AI-Enabled Orchestrator, and a mockup of said orchestrator has been developed (using a basic round-robin implementation of scheduling) to properly test it in the Q&A process of OpenNebula.
Pending Tasks: Implementation of on-demand updates of deployment policies of Serverless Runtimes from the Cloud-Edge Manager that receives requests from Device Clients. This includes the ability to not only reach out to the AI-Enabled Orchestrator to request initial placement policies, but also implement the needed triggers to reschedule existing workloads to optimise a given scheduling policy.

3.6. Secure and Trusted Execution of Computing Environments

SR6.1 Advanced Access Control
Status: IN PROGRESS
Completed Tasks: Threat analysis has been done on the Cloud Edge Manager. Methods to exploit, detect and remediate overly permissive namespace access defaults in a multi-tenant context have been identified.
Pending Tasks: Implementation of detection and remediation controls, validated by the implementation of the exploit.

SR6.2 Confidential Computing
Status: IN PROGRESS
Completed Tasks: A first threat analysis has been done on the Serverless Runtime, highlighting the threat of an attacker inspecting the device memory for confidential information. Confidential computing methods to protect the device against such attacks have been identified, as well as methods and tools to perform the exploit.
Pending Tasks:

Implementation of the confidential computing control, validated by the implementation of the exploit.

4. Priorities for Second Research & Innovation Cycle (M10-M15)

During the Second Research & Innovation Cycle (M10-M15), the Project will focus on those software requirements needed to achieve Milestone 2 by M15, which will lead to the first public release of the COGNIT Framework.

Regarding the **Device Client**, the priority for the next cycle is to have a C SDK (or C version) for the use cases needing it for their integration endeavours. This C SDK will be as similar as possible to the Python SDK (putting aside the language intrinsic constraints). Another major priority for the next version of the Device Client is having in place the methods for being able to upload data from the device (or from a remote endpoint) to the DaaS, which will be added to the COGNIT Framework during the next cycle. Moreover, in this new version the user will be able to update the requirements of the Serverless Runtime, at least through the Python SDK.

With respect to the **Serverless Runtime**, the API will need to be updated to implement the endpoints needed for supporting the creation of DaaS enabling Serverless Runtimes, and additionally will need to provide the means for the Device Client to be able to upload data and copy (or transfer) data to the DaaS from a remote endpoint. Furthermore, it will need to enable the offload of functions from a C version of the Device Client the same way as it does for the Python version.

Work to be carried out in the **Provisioning Engine** includes implementation of the update operation on an existing Serverless Runtime, to modify its requirements in terms of capacity, scheduling policies, presence of DaaS, etc. This component will also need to be integrated with the new Identity and Access Management (IAM) mechanism, and implement a secure channel to communicate with the Device Client.

Regarding the **Cloud-Edge Manager** component, several functionalities will be addressed and developed. The implementation of an IAM mechanism, even if it affects almost all of the other components, will be centralised in this component to correctly define a multi-tenancy environment in the future. This mechanism will most likely be based on JSON Web Tokens (JWT) technology. In the Second Research & Innovation Cycle, this component will exhibit a generic mechanism to create FaaS and DaaS appliances so they can be used in the COGNIT Framework, which will be able to create a new DaaS appliance as well as to refresh the version of the existing FaaS appliance. On the monitoring side, the Scaphandre integration developed in the First Research & Innovation Cycle as part of the Cloud-Edge Manager will be expanded to also implement energy consumption metrics of running Virtual Machines, which then will be conveyed to the AI-Enabled Orchestrator to aid in workload rescheduling decisions. Other metrics may have to be gathered after this second cycle, including the geographic location of the Serverless Runtime and its latency with respect to the Device Client.

In the Second Research & Innovation Cycle, the **AI-Enabled Orchestrator**'s priority-based scheduling algorithm will be replaced with AI/ML models, where multiple machine learning models (e.g. autoencoder based models, LSTM with self-attention head) will be implemented and verified the performance in terms of placement, prediction of energy, and load. Furthermore, features of smart placement will be improved by adding refined and coarse-grained energy and sustainability metrics. This module may utilise the developed ML models in VM placement task and also add more ML models suitable for prioritising energy and sustainability factors for prediction of geo-distributed green powered hosts. Additionally, there will be further enhancements to the OpenNebula scheduler, enabling on-demand updates of deployment policies for Serverless Runtimes from the Cloud-Edge Manager, which receives requests from Device Clients. This improvement encompasses the capability to not only contact the AI-Enabled Orchestrator for initial placement policy requests but also to implement the necessary triggers for rescheduling existing workloads in order to optimise a specified scheduling policy.

Orthogonal to the above tasks, and also in the context of the upcoming development cycle, work will be performed with the focus on transitioning from a component-centric approach to a unified product within the system architecture. Currently, the system comprises separate modules that function independently and integrate primarily at the infrastructure level. To address this, the Second Research & Innovation Cycle aims to implement essential changes. These include the establishment of a streamlined deployment process, the creation of comprehensive documentation covering all system aspects, automation through scripts and Ansible recipes, and centralisation of configuration management. This transition anticipates improved system efficiency and user experience, laying the groundwork for a more user-friendly and adaptable solution in various environments. It is also indispensable to correctly serve the different Use Cases if they want to deploy the full COGNIT stack on-prem.

This transition will enable integration testing, a critical aspect in ensuring the correctness of the COGNIT software stack. To achieve this, a Continuous Integration framework will be established, similar to the existing mechanism for the Provisioning Engine. This framework will involve automated processes that validate the system's integrity at each stage of development. Commits to the master branch will trigger containerised testing, encompassing repository checkout, dependency installation, installation, configuration, launching, executing smoke and security tests, shutting down, and uninstalling. This comprehensive testing strategy will be extended to encompass all integrated components of the COGNIT software stack, ensuring robustness and reliability throughout the development cycle (for more details, please see Deliverable D5.2).

5. 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 this initial definition of the architecture of the COGNIT Framework. 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.

The new open source software components and extensions needed to meet the software requirements are 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.

This first incremental version of the COGNIT Framework Architecture report (Deliverable D2.2) provides an overview of the Project's overall development status, offers a summary of the work done in the First Research & Innovation Cycle (M4-M9), and identifies the priorities for the Second Research & Innovation Cycle (M10-M15). Additional incremental versions of this report will be released at the end of each research and innovation cycle (i.e. M15, M21, M27, M33).