A Cognitive Serverless Framework for the Cloud-Edge Continuum

D2.3 COGNIT Framework - Architecture - c

Version 1.0

30 April 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 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 Second Research & Innovation Cycle (M10-M15), and identify the priorities for the Third Research & Innovation Cycle (M16-M21).

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:	31/03/2024
Actual Date of Delivery:	30/04/2024
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), 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), 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), Ignacio M. Llorente (OpenNebula), Mirko Stojiljkovic (OpenNebula).
Status:	Submitted

Document History

Version	Issue Date	Status[1]	Content and changes
0.1	29/04/2024	Draft	Initial Draft
0.2	29/04/2024	Peer-Reviewed	Reviewed Draft
1.0	30/04/2024	Submitted	Final Version

Peer Review History

Version	Peer Review Date	Reviewed By
0.1	29/04/2024	Johan Kristiansson (RISE)
0.1	29/04/2024	Alberto P. Martí (OpenNebula)

Summary of Changes from Previous Versions

First Version of Deliverable D2.3

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

Executive Summary

Deliverable D2.3, released at the end of the Second Research & Innovation Cycle (M15), is the second 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 Second Research & Innovation Cycle (M10-M15).

During the Second Research & Innovation Cycle (M10-M15), and in line with the expected contribution of each Software Requirement towards meeting the Project’s Milestones and global KPIs, the Project has focused its efforts on releasing the first version (release 1.0) of the COGNIT Framework, which is formed by these main components:

- The **Device Client**, which through its SDK allows the devices to create serverless runtime environments for offloading function executions on the COGNIT Framework and then perform tasks such as function execution, based on a set of requirements defined by the device itself.
- 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 the first release of the COGNIT Framework, and to allow the Use Cases to perform their own research and development activities and implement the first version of their own applications in this cycle (M10-M15), using the Device Client SDK.

The features of the release 1.0 of the COGNIT Framework include:

- Release 1.0 of the Device Client SDK (both for Python and C) that provides the interface to the Provisioning Engine to create, manage, and update Serverless Runtimes by the device and the interface to the Serverless Runtime to offload from the device the execution of Python and C functions.
- Release 1.0 of the Serverless Runtime Appliance that contains the FaaS runtime component needed for the execution of Python and C functions.
- Release 1.0 of the Provisioning Engine that provides a REST API to create, read, update and delete Serverless Runtimes.
- Release 1.0 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 energy-aware algorithms for the placement and workload optimization of the Serverless Runtimes.

The present incremental report (Deliverable D2.3) includes a list of research and development priorities for the Third Research & Innovation Cycle (M16-M21).

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 Serverless Runtime, the Provisioning Engine, and the Secure and Trusted Execution of Computing Environments) are described in detail in reports D3.2 "COGNIT FaaS Model - Scientific Report" and D3.7 "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 D4.2 "COGNIT Serverless Platform - Scientific Report" and D4.7 "COGNIT Serverless Platform - Software Source"; details about the COGNIT software integration and the verification scenarios of each software requirements can be found in report D5.3 "Use Cases - Scientific Report".

This cycle has also witnessed the release of the **OpsForge** tool (see Deliverable D5.7 for more details), which allows the automated deployment of the COGNIT Platform in a target infrastructure, being a public cloud provider such as AWS or an on-premise datacenter. On the other hand, Deliverable D5.10 shows how to deploy the COGNIT Framework on a target infrastructure, and includes a demonstration of some of the capabilities of the COGNIT Framework using the COGNIT testbed.

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

Table of Contents

Abbreviations and Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
CD	Continuous Delivery (Deployment)
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
ML	Machine Learning
OBS	Open Build Service
REST	Representational State Transfer
SDK	Software Development Kit
VM	Virtual Machine

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 incremental version of Deliverable D2.2 provided an overview of the Project's overall development status, offered a summary of the work done in the First Research & Innovation Cycle (M4-M9), and identified the priorities for the Second Research & Innovation Cycle (M10-M15).

This incremental version (Deliverable D2.3) is to provide an overview of the Project's overall development status, offer a summary of the work done in the Second Research & Innovation Cycle (M10-M15), and identify the priorities for the Third Research & Innovation Cycle (M16-M21). An incremental version of this report will be released at the end of each research and innovation cycle (i.e. M15, M21, M27, M33).

D2.3 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 Second Research & Innovation Cycle (M10-M15), 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 Second Innovation Cycle (M10-M15).
- Section 3 provides a brief summary of research and development priorities for the Third Research & Innovation Cycle (M16-M21).

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

No additional user requirements have been identified during the Second Research & Innovation Cycle (M10-M15).

2.2. Update of Software Requirements

Minor amendments to existing Software Requirements

Existing Software Requirements have not suffered any minor amendment during the Second Research & Innovation Cycle (M10-M15).

Existing Software Requirements whose scope has been modified

Existing Software Requirements have not suffered any modification during the Second Research & Innovation Cycle (M10-M15).

New Software Requirements

No additional Software Requirements have been defined during the Second Research & Innovation Cycle (M10-M15).

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

2.4. Global KPIs Progress

The table below shows in which Milestone each Software Requirement is expected to contribute to meet the Project's global KPIs. Here we are marking in green activities actually completed by Milestone 2, 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					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	**[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.									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	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.	2–3	2–3	2	3				2–3											
Developers can change the deployment requirements of the FaaS Runtime according to execution flow.	**[KPI2.3]** Migration across edge PoPs to meet application needs.	2–3							2–3											
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	2–3	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		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.										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					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 2.2. Expected contribution of each Software Requirement towards meeting the Project's global KPIs, by Milestone.

3. Work Done in Second Research & Innovation Cycle (M10-M15)

During the Second Research & Innovation Cycle (M10-M15) the Project has mostly focused on those software requirements needed to achieve Milestone 2 by M15 and to release the first version of the COGNIT Framework.

During this cycle, the tool **OpsForge** has been released in order to allow the user to deploy the main components of the COGNIT Framework: Provisioning Engine, Cloud Edge-Manager, AI-Orchestrator in order to setup a Cloud-Edge infrastructure to manage Serverless Runtimes needed to offload the function execution from the devices. Furthemore, a new version of the Device Client SDK has been released in order to support both Python and C languages and the update of the application requirements.

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

- The **Device Client** SDK for both Python and C provides the interface to the Provisioning Engine to create and manage Serverless Runtimes by the device, to update the application requirements and the interface to the Serverless Runtime to offload from the device the execution of Python and C functions.
- The **Provisioning Engine** provides a REST API to create, read, update and delete Serverless Runtimes.
- The **Serverless Runtime** Appliance contains the FaaS runtime component needed for the execution of Python and C functions. The Serverless Runtime has been instrumented to push metrics related to the function execution to the Monitoring System. Finally, for each use case a Serverless Runtime appliance has been built with specific libraries needed for executing the Use Case application functions.
- The **Monitoring** system has been enhanced with energy measurements related both to the infrastructure (hosts) and virtual resources (Serverless Runtimes) and with information related to the geolocation of the infrastructure.
- Some **ML/AI models** have been evaluated, trained and validated for two main purposes: workload characterization of Serverless Runtimes and interference-aware scheduling.
- The ML/AI models have been integrated in the **AI-Enabled Orchestrator** for the smart placement of Serverless Runtimes across the resources provisioned and managed by the **Cloud-Edge Manager** in order to optimise the energy consumption

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 D5.3.

The following section summarises, per component, the work that has been done as part of the Second Research & Innovation Cycle (M10-M15), 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. The Device Client supports the update of the requirements of an existing Serverless Runtime.
Pending Tasks: 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 Python and C clients. 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: COMPLETED
Completed Tasks: The Device Client supports both C and the Python programming languages. The Device Client is implemented as a Python module and C library. Both implementations enable 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. The Device Client (C and Python) also supports the update of the requirements of an existing Serverless Runtime.

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. FaaS 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.
Pending Tasks: Implementation of mechanisms for the secure communication between the Device and the FaaS Runtime.

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. The Provisioning Engine has been implemented as a REST API to create/read/update/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. 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 lifecycle 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. 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: ON HOLD
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: Once the initial research tasks associated to SR6.1 “Advanced Access Control” have been

completed, 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 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 Cloud-Edge Continuum according to application and resource requirements. Several models for the Orchestrator have been trained and validated.
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. Workload characterization and interference-aware scheduling for energy requirements have been integrated in the AI-Enabled Orchestrator. 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.

SR5.3 Scheduling Mechanisms
Status: COMPLETED
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. 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 Third Research & Innovation Cycle (M16-M21)

During the Second Research & Innovation Cycle (M10-M15), and as part of the research and development tasks undertaken by the Use Cases, a few limitations have been identified in the original COGNIT Architecture. 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 leads to:

1. A public IP being required for each Serverless Runtime, since the Device communicates directly with the Serverless Runtime. This requires the usage of IPv6 in order to scale up to a large number of Devices.
2. 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.
3. Creation of Serverless Runtimes cannot be anticipated by the AI-Enabled Orchestrator in order to reduce the cold start for low-latency applications.
4. 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.
5. Developers need to directly manage the lifecycle of the Serverless Runtime, by creating, updating and releasing it according to the application requirements.

During the Third Research & Innovation Cycle (M16-M21), the Project will focus 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, the next cycle will be devoted to the design of that 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.

Regarding the **Device Client**, the priority for the next cycle is to increase the user experience by improving the user API; in particular, the developer shall define the requirements of the application, the definition of the functions to be offloaded and the workflow for the execution of the functions. The Serverless Runtimes will be managed by the COGNIT Platform and not by the device anymore. Furthermore, the Device Client will support the uploading of data and will establish secure communications with the COGNIT Platform. Finally, the Device Runtime will collect and push latency measurements from the device to the Serverless Runtime that is doing the actual offloading instead of the latency to the Provisioning Engine. The main purpose of these changes is to move towards a fully synchronous communication with the COGNIT Platform from this component.

From the **Serverless Runtime** standpoint, the main focus in the following cycle should be the addition of the DaaS component, which is paramount for many Use Cases in order to manage the lifecycle of data within the cloud-edge continuum. Moreover, in order to support the Use Cases to build their own Serverless Runtime images, it has been identified the need of an automated image building system able 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 Open Build Service (OBS). As part of this solutions, images are defined by XML descriptions and allow for automating the build process of an image. Most Use Cases require specific adjustments regarding additional packages, files, and other preparation tasks, in order to speed up the provisioning and execution process of the Serverless Runtime. Those specific Use Case customisations can be achieved through small extensions to the XML descriptions and thus ease the creation of specific images.

Furthermore, work to be carried out in the **Provisioning Engine** includes the integration with a new Identity and Access Management (IAM) mechanism implementing a secure channel to communicate within the COGNIT Framework. Moreover, work will be done to implement a mechanism (e.g. based on WebSocket) to send asynchronous events to the Device Client.

Regarding the **Cloud-Edge Manager** component, several functionalities will be addressed and developed. The implementation of that IAM mechanism will also impact this component given the need 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 Third Research & Innovation Cycle, this component will also exhibit a generic mechanism to create FaaS appliances so they can be used in the COGNIT Framework. Most of the work in this cycle will be focused on the ability to dynamically create new edge locations automatically to better satisfy scheduling policies and Serverless Runtime 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 Third Research & Innovation Cycle, the **AI-Enabled Orchestrator** will focus on the development of multiple features that enable smart and optimised downstream orchestration tasks, including energy and latency-aware placements, in order to achieve this select model from the model repository according to the specific task. The AI-Enabled Orchestrator is decomposed into multiple subcomponents, including the database, the environment server, and the ML server. 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 and energy-aware resource optimization, 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.

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 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 incremental version of this report (Deliverable D2.2), the new 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.

This second incremental version of the report (Deliverable D2.3) provides an overview of the Project's overall development status, offers a summary of the work done in the Second Research & Innovation Cycle (M10-M15), and identifies the priorities for the Third Research & Innovation Cycle (M16-M21). Additional incremental versions of this report will be released at the end of each research and innovation cycle (i.e. M21, M27, and M33).