6GSNS

Co-funded by the European Union

SAFE-6G

SAFE-6G

A Smart and Adaptive Framework for Enhancing Trust in 6G Networks

Deliverable D3.2: Cloud continuum, service mesh and resource orchestration

Date: 30/06/2026

Version: V1.0

DISCLAIMER

Grant Agreement	101139031
Document number	D3.2
Document title	Cloud continuum, service mesh and resource orchestration
Lead Beneficiary	UPV
Editor(s)	Alejandro Fornés (UPV)
Author(s)	Alejandro Fornés (UPV) Francisco Mahedero (UPV) Juan Gascón (UPV) Andreas Sakellaropoulos (NCSRD) Pablo Iglesias Sanuy (TID)
Dissemination level	Public
Contractual date of delivery	30/06/2026
Status	Final
File name	SAFE-6G_D3.2_V1.0.pdf

Revision History

Version	
V0.1	TOC proposal, guidelines and templates
V0.2	TOC refinement and first round of contributions
V0.9	Internal review (UNIWA, TID) comments addressed, sent to final edition
V1.0	Final version

GLOSSARY

Abbreviations/Acronym	Description
5G	5th Generation of Mobile Networks
6G	6th Generation of Mobile Networks
AF	Application Function
AI	Artificial Intelligence
API	Application Programming Interface
CAPIF	Common API Framework
CB	Context Broker
CD	Continuous Deployment
CNC	Cumucore Network Configuration
CNF	Cloud Native Network Function
CNI	Container Network Interface
CRI	Container Runtime Interface
CPU	Central Processing Unit
CRD	Custom Resource Definition
DataOps	Data Operations
gNB	Next Generation NodeB
gRPC	gRPC Remote Procedure Call
GUI	Graphical User Interface
HLO	High-Level Orchestrator
IdM	Identity Management
IE	Infrastructure Element
IoT	Internet of Things
JWK	JSON Web Key
K8s	Kubernetes
LLO	Low-Level Orchestrator
MANO	Management and Orchestration
MCS	Multi-Cluster Service
MLOps	Machine Learning Operations
mTLS	Mutual Transport Layer Security
NEF	Network Exposure Function
NF	Network Function
NWDAF	Network Data Analytics Function
OS	Operating System
OSM	Open-Source MANO
RAM	Random Access Memory
Rel	Release
REST	Representational State Transfer
SCP	Service Communication Proxy
SotA	State of the Art
SQL	Structured Query Language
TOSCA	Topology and Orchestration Specification for Cloud Applications
TF	Trust Function
VPN	Virtual Private Network
WP	Work Package
XR	Extended Reality

EXECUTIVE SUMMARY

This document represents the final software outcomes and architectural implementations of Task 3.1, reporting on the design, execution, and deployment of the Edge-Cloud continuum layer as well as the DataOps module for the SAFE-6G framework. The involved components compose the overall framework that manages distributed computing resources and orchestrates the lifecycle of microservices across heterogeneous domains, also monitoring their status and performance.

At the core of the continuum implementation is a Meta-Operating System based on ECLIPSE aeriOS. This system offers a programmable environment designed to integrate seamlessly with standard Cloud Native technologies. It successfully enables the federation of delocalized computing domains and provides continuous observability, identity management, and distributed service orchestration using Kubernetes as its foundation. The Meta-OS relies on critical components including a FIWARE Orion-LD Context Broker for sharing context data without replication, and an enhanced dual-level orchestration engine to automate the deployment, upgrade, and removal of services packaged as Helm charts or containers. The flexibility of its data model has allowed the extension of its default definition with SAFE-6G specific entities and attributes. Also, the exposure capabilities of the continuum have been secured by an API Gateway built on KrakenD, fully integrated with OpenCAPIF Release 3 to facilitate automated API discoverability, access management, and interaction with the broader SAFE-6G Trust Functions.

To complement the orchestration logic, a Multi-Domain Service Mesh has been successfully deployed utilizing Cilium. This module manages intra- and inter-domain connectivity, acting as a Cloud Native CNI plugin that enforces granular network policies. Thanks to the introduction of bidirectional VPNs and Cluster mesh capabilities, this layer guarantees secure service-to-service communications, comprehensive traffic observability, and multi-domain network security across the delocalized microservices.

Finally, the document details the DataOps Implementation, the module of the SAFE-6G framework responsible for the telemetry, monitoring, and long-term storage of infrastructure metrics. Specifically, the DataOps deploys a robust Prometheus stack, integrated with Loki for domain-level log aggregation and Grafana for visual dashboards. To implement a complete observability, widely-used metrics exporters have been complemented by a set of custom ones, such as the Data Fabric exporter, 5G Core exporter, Metaverse manager exporter, and gNB performance exporter. Together, these modules generate the historic and real-time data strictly required to train models in MLOps pipelines and empower the project's Trust Functions.

This report summarizes the requirements considered for their realization and the features implemented, pointing to relevant documentation and with extended details on those components explicitly developed or adapted for the needs of SAFE-6G; this is, including additional information such as developed/available APIs, container images and Helm packages' routes, required resources, etc.

KEYWORDS

Edge-Cloud Continuum, Meta-Operating System, Service Orchestration, Service Mesh, Cloud Native, DataOps, Prometheus.

TABLE OF CONTENTS

List of FIGURES

List of TABLES

1 INTRODUCTION

Figure 1 highlights in green and yellow the building blocks of the SAFE-6G architecture implemented in Work Package 3 (WP3). As a quick reminder, the SAFE-6G framework (whose core logic is developed in WP4) can modify the configuration or behaviour of the underlying planes (continuum, core, service/application or addition of supporting services) via Open Common Application Programming Interface (API) Framework (OpenCAPIF), based on the trust-related intents provided by users in combination with contextual, real-time data. To facilitate the interactions with users, an AI-powered Chatbot has been developed, embeddable in different user interfaces – Extended Reality (XR) headset, web browser, etc. Additionally, the AI models leveraged by WP4 components – i.e., cognitive coordinator and Trust Functions' (TFs) AI agents are trained using the MLOps and the DataOps platforms embedded in the framework (the former to orchestrate the training process and the latter to gather and collect the data leveraged in such training). Finally, the entire ecosystem has been tested and validated considering two Metaverse use cases.

More specifically, this deliverable presents the open-source implementation of the two main blocks: the **computing continuum**, which encompasses the orchestration of resources and services as well as their multi-domain connectivity by means of a service mesh; and the **DataOps**, which focuses on the collection and storage of the metrics from the 3 underlying planes. The involved blocks are highlighted in yellow, including the APIs to manage or get information from the continuum as well as the monitoring exporters deployed over it to get relevant metrics from the involved infrastructure and deployed services. The corresponding source code is publicly available at https://gitlab.com/safe-6g/development/continuum and https://gitlab.com/safe-6g/development/dataops, respectively. Rather than repeating the implementation and deployment details already provided in the repository documentation, this deliverable offers a high-level overview of the technical implementation of the specific component(s). It summarises the role of the component(s) within the SAFE-6G architecture, main final features, design choices changes and key implementation resources to facilitate their uptaking and reuse.

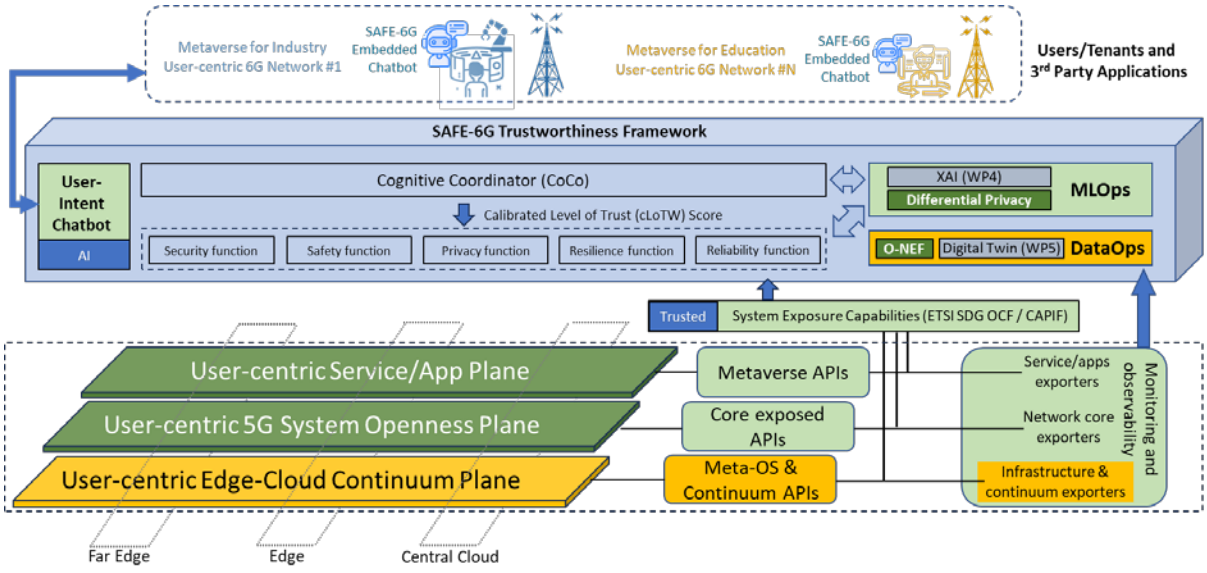

Figure 1: Building blocks and components of the SAFE-6G architecture

1.1 THE RATIONALE BEHIND THE STRUCTURE

Task 3.1 aimed at focusing on the following aspects: distributed computing continuum management; services orchestration and lifecycle management; service mesh considering 3GPP Release-16 Service Communication Proxy (SCP); observability and metrics gathering; network policies; and AI-based CNF chaining. The project adopted the novel Meta-Operating System (Meta-OS) paradigm for addressing most of these aspects, with specific developments to accommodate the requirements of the project and the 5G/6G ecosystem.

With respect to the structure of the deliverable, Section 2 provides a report of the Continuum implementation, with two subsections – one for the Meta-OS itself and one for the Multi-Domain service mesh implementation, which simplifies the communication of services deployed on different computing domains. For each of them, a short review of the requirements addressed, the features of the final release and the architecture updates (if any) are presented. Additionally, for specific components tailored or evolved during the project implementation, a summary table including links to the source code, container images, Helm charts and API documentation is provided. The Docker images currently available in SAFE-6G's GitLab registry will be progressively shifted to the project's Docker-Hub account: https://hub.docker.com/u/safe6g. Section 3 performs the same exercise but for the DataOps module, with dedicated subsections for the exporters adopted and developed as well as for the Grafana dashboards prepared. While the Meta-OS provides some inherent monitoring capabilities, advanced Cloud Native enablers are incorporated to enhance the observability features across the continuum. Section 4 briefly walks through the realization of the continuum and the DataOps, pointing to the relevant documentation, while Section 5 presents the conclusions of the deliverable.

2 EDGE-CLOUD CONTINUUM IMPLEMENTATION

The computing continuum refers to the effective integration of Internet of Things (IoT), edge and Cloud computing layers, facilitating the use of the available distributed resources by unifying their management. SAFE-6G adopts the Meta-OS model to manage such resources and orchestrate the lifecycle of services on top of them, as an alternative of State of the Art (SotA) alternatives like Management and Orchestration (MANO) due to the additional features it incorporates. While a Meta-OS can integrate components of different building blocks (e.g., accountability and logging mechanisms, cybersecurity, distributed AI inference, etc.), SAFE-6G focuses on its **orchestration, data federation and monitoring** capabilities. Apart from Meta-OS features, the continuum integrates *de facto* Cloud Native standards as complementary enhancements, as being a paradigm embraced by the 5G/6G ecosystem. In summary, the main continuum functionalities considered for SAFE-6G include:

- Federation of delocalised computing domains, running on top of Kubernetes (K8s). Seamless exchange of context data among the registered ones.
- Lifecycle management of services (deploy, update, remove) on top of the infrastructure, packaged in containers or Helm charts. Manual and automatic allocation enabled.
- Cloud Native load balancing integrating native support to core network functions (i.e., SCP from 3GPP release 16).
- Infrastructure and service monitoring and autonomous (self-*) operations.
- Container Network Interface (CNI) for Network policy management (L3, L4).
- Intra and Multi-domain service mesh (L7).

The Meta-OS components, described in D3.1, are depicted in Figure 2 (no significant changes). The Meta-OS itself is deployed in all Infrastructure Elements (IEs) of the managed domains, being some of its services running in all IEs – i.e., Low-Level Orchestrator (LLO), Self-*; while others only required once per domain – API Gateway, Context Broker (CB), High-Level Orchestrator (HLO), Federator, Load Balancer); or per continuum (Portal, HLO Balancer). Complementary to the Meta-OS, additional continuum features are provided by external components:

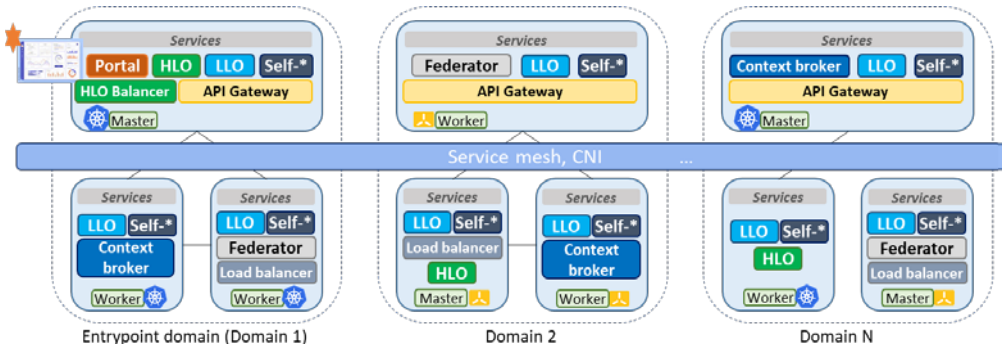

Figure 2: High-level architecture of Meta-OS inter-domain orchestration and context data sharing

2.1 META-OPERATING SYSTEM

The selection and developments of the Meta-OS are related to the following project-level requirements (from D2.1):

- **REQ-COM-ETH-NF-M-03:** Compliance with 5G and industry standards.

- **REQ-COM-ETH-NF-M-04:** Following Cloud Native paradigm.
- **REQ-COM-DES-F-M-08:** APIs interaction through Open CAPIF.
- **REQ-COM-DES-F-M-20:** Access to performance metrics from the cloud continuum.
- **REQ-COM-DES-F-R-24:** Fault Tolerance.
- **REQ-COG-SCHED-F-M-11:** Management of Cloud Computing continuum resources.
- **REQ-INFRA-RES-NF-M-01:** Distributed, heterogeneous ecosystem of computing resources.
- **REQ-INFRA-RES-NF-M-02:** Support of Cloud Native technologies.
- **REQ-INFRA-META-NF-M-03:** Meta-operating system for the IoT edge-cloud continuum.
- **REQ-INFRA-META-NF-M-04:** Exposure capabilities of the meta-operating system.
- **REQ-INFRA-META-NF-M-05:** Data accessibility across the computing continuum.
- **REQ-INFRA-NET-NF-M-09:** Reallocation capabilities for ensuring low latency requirements.
- **REQ-INFRA-NET-NF-M-10:** Secure network connectivity.
- **REQ-INFRA-SEC-NF-M-11:** Management of identity and access to the meta-operating system.
- **REQ-INFRA-NET-NF-R-12:** Load balancing adapted to 6G needs.

The Meta-OS tailored for the project has been based on ECLIPSE aeriOS[1], delving into specific requirements posed by the SAFE-6G framework and the Cloud Native paradigm of 5G and beyond ecosystem. The final release enhances the robustness of the Meta-OS, expands its offered APIs and its overall robustness and monitoring capabilities. The full list of key **features** is described here, highlighting those implemented for Release (Rel)-B:

- Baseline federation, orchestration, self-* and security features from aeriOS:
  - Federation of computing domains.
  - Dedicated continuum ontology.
  - Continuous observability of computing elements.
  - Lifecycle management (deploy, delete) of containerised workloads.
  - Identity management (Keycloak-based) and encrypted communications.
  - Unified management portal / Graphical User Interface (GUI) and decentralised API-based management.
- Rel-A: Extended support for services packaged as Helm charts (updated HLO components, dedicated LLO).
- Rel-A: implementation of service upgrade capability.
- Rel-A: Updated GUI and API to manage the previously-mentioned Rel-A features.
- Rel-A: CD features via FluxCD.
- Rel-A: Replacement of MetalLB load balancing technology to LoxiLB, optimized for 5G core support.
- Rel-B: Updated continuum ontology (service entity) to support SAFE-6G framework-specific attributes (see Section 2.1.1 below).
- Rel-B: Use of Topology and Orchestration Specification for Cloud Applications (TOSCA) of the HLO API endpoints simplified, especially for introducing Helm values and target IE.
- Rel-B: New GET, PUT, PATCH and DELETE methods implemented for more granular lifecycle management options (see Table 1 below).

[1] ECLIPSE aeriOS. Website: https://projects.eclipse.org/proposals/eclipse-aerios. Documentation: https://github.com/eclipse-aerios/resources

- Rel-B: Support for private Helm chart repositories apart from public ones.
- Rel-B: New internal monitoring component "stage manager" to extend the traceability of deployments through the HLO-LLO-K8s status.
- Rel-B: Updated Management Portal graphical interfaces, supporting the updated features of the final release.
- Rel-B: API Gateway integration with OpenCAPIF Release 3 for API discoverability and usage.

It should be mentioned that the CNF chaining feature was discarded. Its requirement was identified as optional as most TFs implement Application Functions (AFs) rather than Network Functions (NFs), thus not applicable. Also, chaining requires manifests to be specifically prepared for that aim, as happens with ETSI's Open-Source MANO (OSM), which would overcomplicate the onboarding of TFs.

Relevant process flows related to the Meta-OS are in essence those stated in D3.1 (related to the deployment, update and termination of services as well as the retrieval of data from the context broker). New API endpoints have been included based on requirements received during implementation. While the "/ngsi-ld/v1" endpoints remain the same, the "/hlo_fe/" services endpoints related to service lifecycle orchestration have been extended as depicted in Table 1:

API prefix: /hlo_fe/services		
Endpoint	***Method/s***	***Description***
/{service_id}	POST/PATCH (PUT in Rel-A)/PUT/DELETE/GET	Deploys, upgrades, reallocates, removes or provides information related to a managed service. Reallocation (PUT) deploys/reuses a previously deallocated (DELETE) service
/{service_id}/component/{service_component_id}	PATCH	Used for updating a single service component (e.g., if a service is composed by 2 Helm charts, it only updates one of them)
/{service_id}/purge	DELETE	Completely removes the service metadata from the CB (only enabled if not running)

Table 1: Updated HLO API

The main enhancements/adaptations of Meta-OS components during SAFE-6G execution are now described (modules and components such as Load Balancer, Self-*, Federator or its native Identity management system are used but not commented, please check the Meta-OS documentation[1]).

2.1.1 Context Broker

The Meta-OS CB manages contextual information modelled as entities by recording the most recent values of their attributes. Data can be obtained either via subscriptions or REST API requests. In SAFE-6G, a CB instance is deployed on a computing node of each domain, which is then federated, so that the managed information can be accessed from any point of the continuum without requiring data to be replicated. Therefore, it plays a crucial role for the optimal performance of distributed Meta-OS operations such as service orchestration, as it facilitates the sharing of data (REQ-INFRA-META-NF-M-05). The component is based on a forked version of the Open Source tool FIWARE **Orion-LD** (which works with a MongoDB) having focused on the project mostly in the adaptation of the Continuum's Smart data model[2] for SAFE-6G, namely in the following entities (highlighted in green in Figure 3):

[2] DataModel.IT Smart Data Model: https://github.com/smart-data-models/dataModel.IT

- Service entity (attributes extended)
  - belongsToTrustFunction [Boolean: True/False]
  - trustFunctionDimension [String: “Security” / “Safety” / “Privacy” / “Resilience” / “Reliability”]
- ServiceComponent entity (attributes extended):
  - trustFunctionElement [String: “aiAgent” / “nApp” / “vApp”]
  - interactsWithContinuum [Boolean: True/False]
  - interactsWithCore [Boolean: True/False]
  - interactsWithService [Boolean: True/False]
- Service Lifecycle (new entity and attributes)

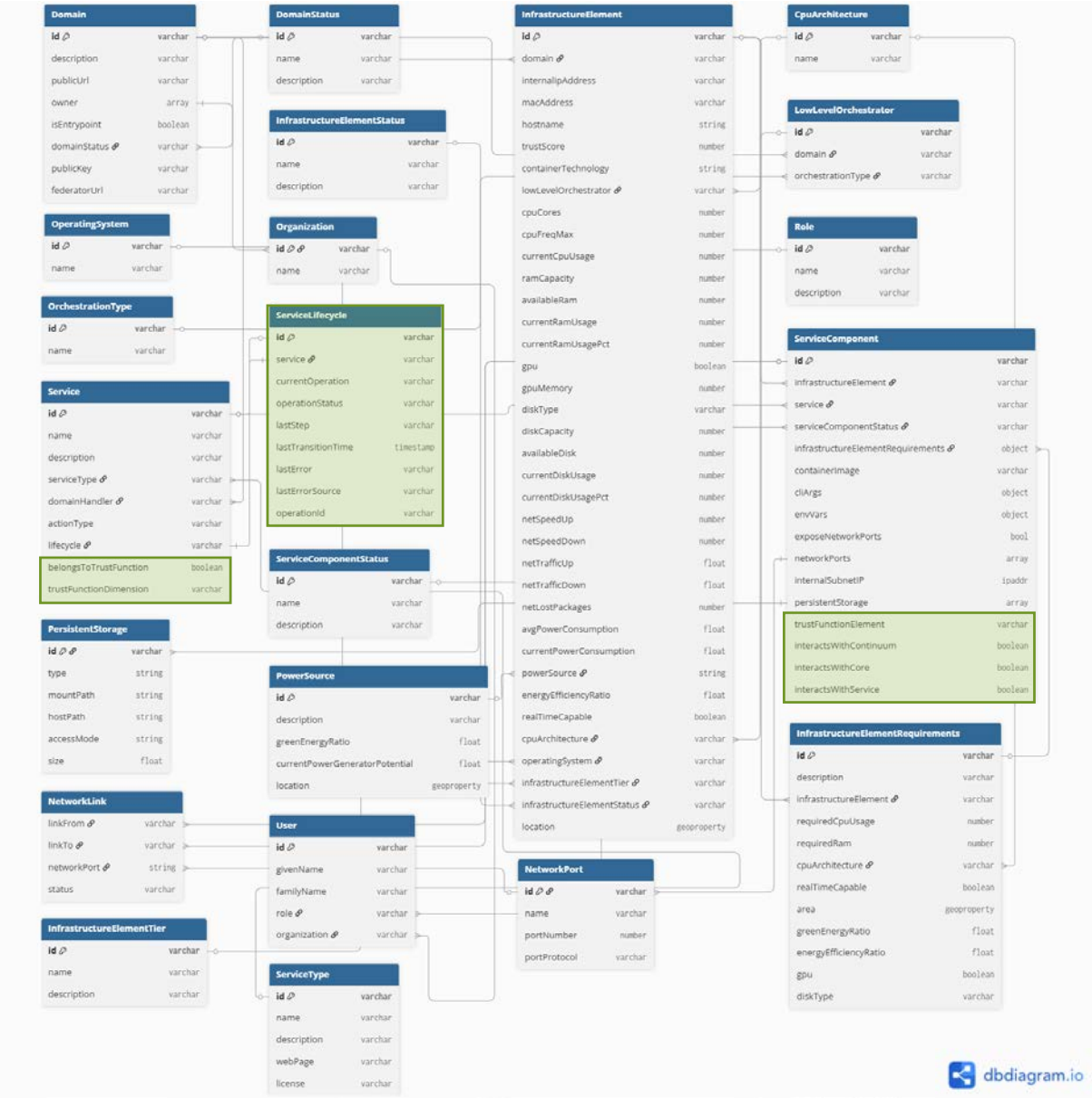

Figure 3: SAFE-6G extended continuum data model

FIWARE's Orion-LD is the CB used, which federation capabilities were enhanced during the HE aerOS[3] project implementation. It must be deployed once per domain, consisting of an API and a MongoDB and requiring at least 1 vCPU and 1 GB RAM.

2.1.2 ORCHESTRATION (HLO, LLO)

The Orchestration module translates service lifecycle management requests (REQ-INFRA-META-NF-M-03) into concrete instructions, which are executed on computing nodes of the managed continuum. The orchestration flow has two levels:

1. High-Level Orchestrator (HLO): Available per domain, it receives the deployment instructions (via TOSCA descriptor) and, if not manually-decided by a user, selected their positioning in the continuum thanks to its decision support system. It also channels updates and deallocation.
2. Low-Level Orchestrator (LLO): Available at computing IE level, it translates HLO commands to lifecycle management actions tailored to the available deployment engine/s (with services packaged as Helm charts or as plain containers).

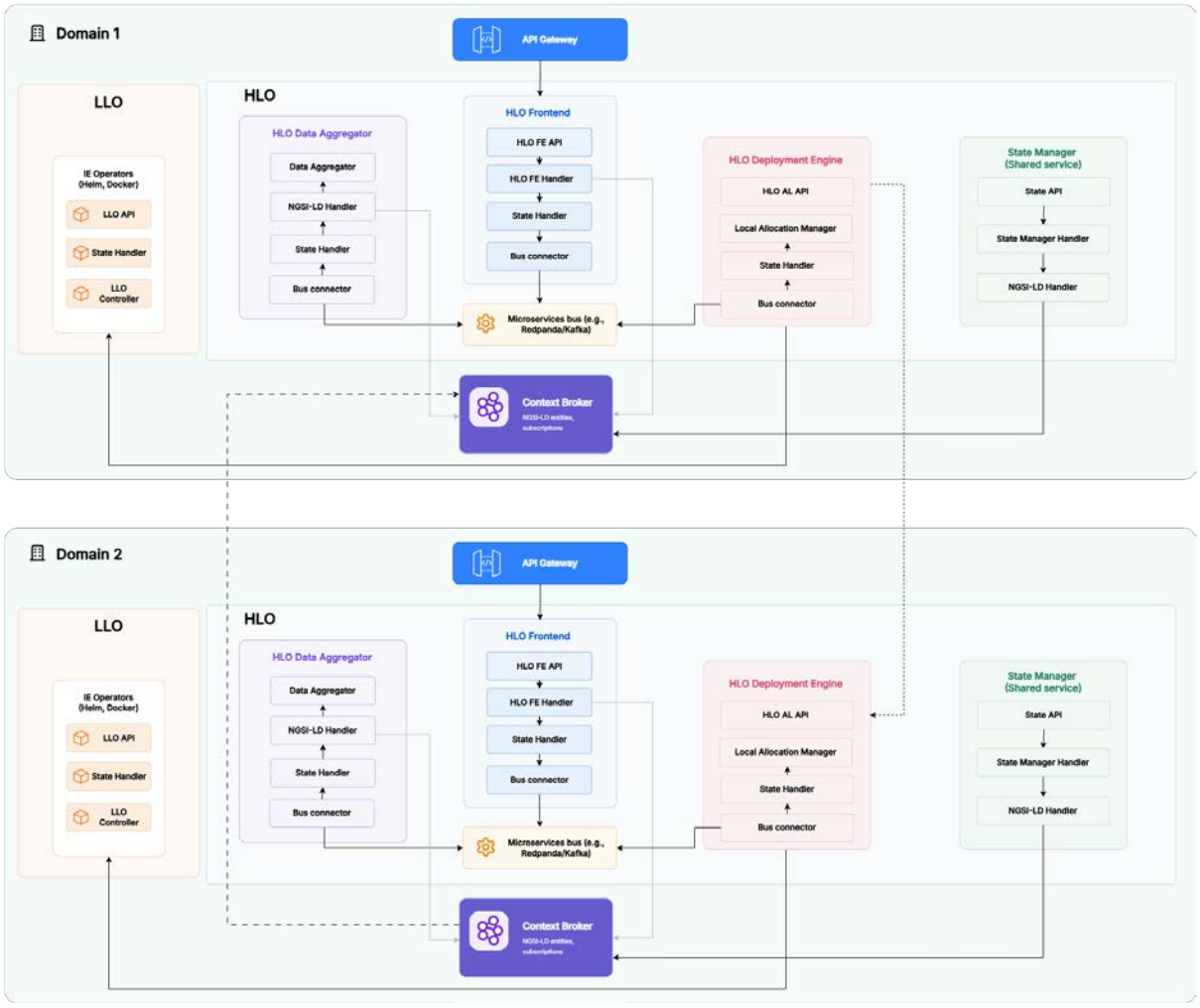

Figure 4: Orchestration components breakdown and communication (internal and with Context Broker)

[3] Horizon Europe CL4-2021-DATA-01-05 aerOS project. Website: https://aeros-project.eu/. Documentation: https://github.com/fiware/context.orion-ld

Thanks to the CB (see *Infrastructure Element* entity in Figure 3) and the federation mechanisms, the HLO can obtain the state of IEs captured by the deployed Self-*awareness components across the continuum and can perform informed allocation decisions. In the project, the Consortium has worked on the one hand to provide support for services and applications packaged as Helm charts, as it is the de facto *Cloud Native* packaging and deployment technology (REQ-INFRA-RES-NF-M-02); and second, to implement new methods and flow to smooth the lifecycle management of services, as presented in Table 1.

The release summary of the orchestration components can be found in the following tables. The HLO is composed of four components, three of which have been modified during the project execution:

Category	***Status***
Version	v2.0.0
Documentation	https://gitlab.com/safe-6g/development/continuum/-/blob/main/services/hlo/README.md?ref_type=heads
Source code	The HLO is composed of four main components, three of which have been adapted and tailored for the SAFE-6G needs (support for Helm charts, new methods – including upgrade, etc.): • hlo-frontend • hlo-data-aggregator • hlo-local-allocation-manager https://gitlab.com/safe-6g/development/continuum/-/tree/main/services/hlo?ref_type=heads The fourth component (HLO allocator) source code is hosted in ECLIPSE aeriOS repository, not modified in SAFE-6G: https://github.com/eclipse-aerios/hlo-allocator/tree/main
Docker images	The four related container images are the following: • registry.gitlab.com/safe-6g/development/continuum/hlo/local-fe-engine:v2.0.0 • registry.gitlab.com/safe-6g/development/continuum/hlo/data-aggregator:v2.0.0 • registry.gitlab.com/safe-6g/development/continuum/hlo/local-allocation-manager:v2.0.0 • registry.gitlab.com/safe-6g/development/continuum/hlo/hlo-allocator:v2.0.0
Helm charts	And packaged in the following respective charts: • local-fe-engine-1.1.1.tgz • data-aggregator-1.1.1.tgz • local-allocation-manager-1.1.1.tgz • hlo-allocator-3.0.0.tgz
Open API Swagger	https://gitlab.com/safe-6g/development/continuum/-/blob/main/documentation/api/openapi/hlo-fe-engine.openapi.yaml?ref_type=heads
Demo	NA
Deployment requirements	0.1 vCPU, 400 MB RAM
Dependencies	aeriOS dependencies are documented here: https://gitlab.com/safe-6g/development/continuum/-/tree/main/deployment/infrastructure?ref_

	type=heads, with a dedicated script to prepare the computing nodes. Other components of aeriOS have to be installed in advance for it to work. See deployment manual: https://gitlab.com/safe-6g/development/continuum/-/blob/main/deployment/README.md?ref_type=heads
License	Apache 2.0

Table 2: HLO module final release summary

Category	***Status***
Version	v2.0.0
Documentation	https://gitlab.com/safe-6g/development/continuum/-/blob/main/services/llo/README.md?ref_type=heads
Source code	The LLO is composed of two main components, which have been developed for the SAFE-6G needs (dedicated LLO for managing Helm charts, and their lifecycle management): • llo-cr-api • llo-k8s-operator-sdk https://gitlab.com/safe-6g/development/continuum/-/tree/main/services/llo?ref_type=heads
Docker images	The two related container images are the following: • registry.gitlab.com/safe-6g/development/continuum/llo/cr-api:v2.0.0 • registry.gitlab.com/safe-6g/development/continuum/k8shelm-operator:v2.1.0
Helm charts	And packaged in the following respective charts: • llo-api-1.0.0.tgz • llo-k8s-helm-1.1.0.tgz
Open API Swagger	NA (Internal module, interacted by the HLO module)
Demo	NA
Deployment requirements	0.5 vCPU, 500 MB RAM
Dependencies	aeriOS dependencies are documented here: https://gitlab.com/safe-6g/development/continuum/-/tree/main/deployment/infrastructure?ref_type=heads, with a dedicated script to prepare the computing nodes. Other components of aeriOS have to be installed in advance for it to work, including the HLO stack. See deployment manual: https://gitlab.com/safe-6g/development/continuum/-/blob/main/deployment/README.md?ref_%20type=heads
License	Apache 2.0

Table 3: LLO module final release summary

2.1.3 Management Portal

The Portal provides a set of user dashboards that allows them to manage their computing continuum and services in a user-friendly way (alternatively, the API endpoints can be consumed). The following interfaces are available: Landing page, Domains & continuum (views providing details including IEs), Deployments, Benchmarking, Users management, Notifications, Settings.

In SAFE-6G, main changes have been in the Deployment view, working on: i) specific visualizations for Helm chart-based deployments (as the main version focuses on containers), ii) update option, and (iii) TOSCA file preparation and downloading to facilitate the preparation of REST API calls. The Portal's release summary can be found in the following table:

Category	***Status***
Version	v2.0.0
Documentation	https://gitlab.com/safe-6g/development/continuum/-/blob/main/services/portal/README.md?ref_type=heads
Source code	The Management portal is composed by two services: frontend and a backend: https://gitlab.com/safe-6g/development/continuum/-/tree/main /services/portal?ref_type=heads
Docker images	The two implemented container images are the following: • registry.gitlab.com/safe-6g/development/continuum/management-portal/frontend:v2.0.0 • registry.gitlab.com/safe-6g/development/continuum/management-portal/backend:v2.0.0
Helm charts	With the following packages: • management-portal-1.2.0.tgz (external package)
Open API Swagger	NA
Demo	NA
Deployment requirements	0.5 vCPU, 500 MB RAM
Dependencies	For the management portal to work, the Context broker and orchestration components should perform properly, including thus their respective dependencies.
License	Apache 2.0

Table 4: Management Portal final release summary

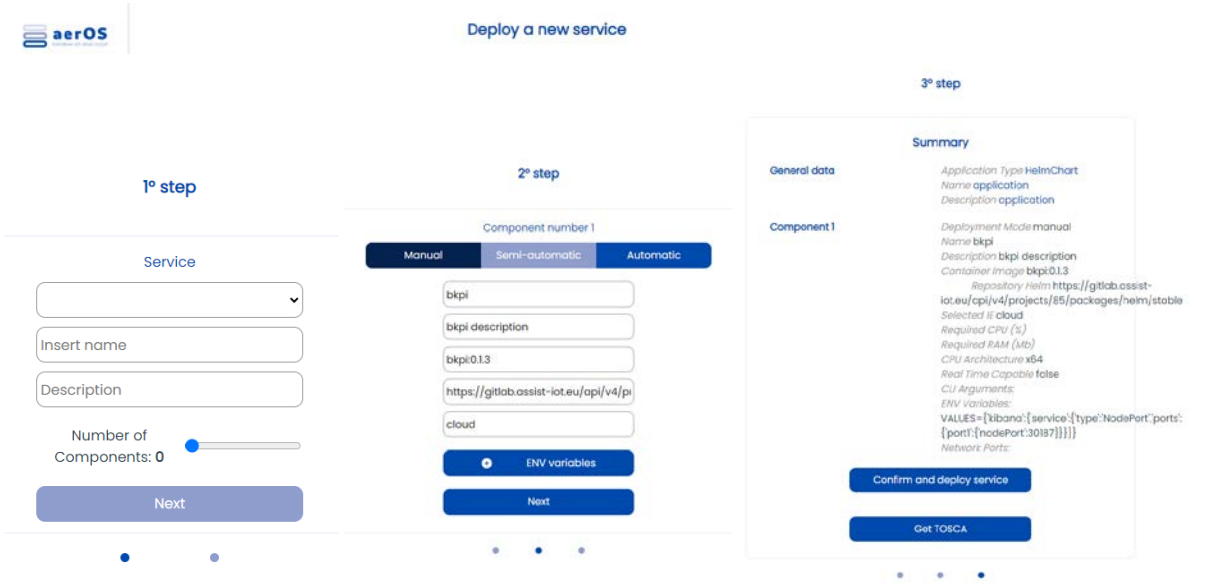

Figure 5: Deployment assisted by the portal

Figure 6: Deployments menu

2.1.4 API GATEWAY

The API Gateway serves two purposes: (i) entrypoint for Meta-OS services so they can communicate among them regardless of their location in the continuum – HLO API, Federator, CB, Portal, etc.; and (ii) interface for external systems, such as the TFs of the SAFE-6G framework. In the Meta-OS implementation, an Identity Management system (IdM, **Keycloak** in this case) grants or denies access to the exposed endpoints. Apart from updating it to expose the methods implemented during Rel-B (Table 1), this component (implemented with **KrakenD**) has been integrated with OpenCAPIF so that the API endpoints can be discovered and consumed by authorized invokers using this technology (REQ-COM-DES-F-M-08, see D3.4), thus replacing the role of the IdM. Specifically, the Meta-OS gateway was split in two as seen in Figure 7 below: one integrated with CAPIF, and another for Meta-OS internal purposes, needed for having an effective federation of its internal features.

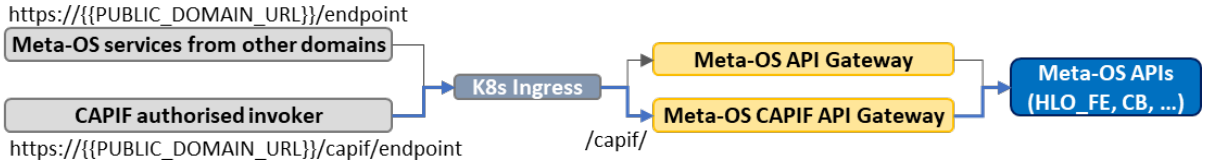

Figure 7: Meta-OS Gateway and CAPIF integration

Focusing on the Gateway integrated with the CAPIF flow, the following actions were performed:

1. Meta-OS Open API specification sent to OpenCAPIF manager (NCSRD) to register the API Gateway as CAPIF provider, taking into account that the base path is "/capif/".
2. OpenCAPIF manager returns a PEM certificate. This certificate cannot be used directly by KrakenD to validate the tokens; it must be converted to JSON Web Key (JWK).
3. Once obtained, a K8s secret is created in the hosting cluster with it.
4. KrakenD configuration endpoints are modified to use the local JWKs file instead of Keycloak, (minor change in extra_config). For instance, given one of the exposed endpoints:

```
"extra_config": {
  "auth/validator": {
    "alg": "RS256",
    "jwk_local_path": "/etc/krakend/jwks.json",
    "disable_jwk_security": true
  }
}
```

As mentioned, the API Gateway is based on KrakenD technology (v2.6.3, available: https://github.com/krakend/krakend-ce/releases/tag/v2.6.3), and its source code has not been modified in the project. Still, its deployment manifests for K8s and Helm charts have been modified to expose the specific Meta-OS services as well as doing it though OpenCAPIF, and which specific resources can be observed in the following table:

Category	*Status*
Version	V2.6.3
Documentation	https://gitlab.com/safe-6g/development/continuum/-/tree/main/deployment/aeros/krakend?ref_type=heads
Source code	NA (official code)
Docker images	devopsfaith/krakend:2.6.3 (external image)
Helm charts	api-gateway-1.7.0.tgz
Open API Swagger	It exposes the methods for the HLO and the CB: https://gitlab.com/safe-6g/development/continuum/-/tree/main/documentation/api/openapi?ref_type=heads
Demo	NA
Deployment requirements	0.25 vCPU, 128 MB RAM
Dependencies	As with the Management Portal, the CB and orchestration components should perform properly, including thus their respective dependencies.
License	Apache 2.0

Table 5: API Gateway final release summary

2.2 Multi-Domain Service Mesh

Service mesh technologies enhance the traceability, connectivity and security of distributed applications, providing a set of functionalities that are offloaded from the application logic, so that they can focus on the latter. Implementations vary, but service meshes typically incorporate Layer-7 traffic management features (e.g., load balancing, rate limiting), automatic encryption and authentication – e.g., Mutual Transport Layer Security (mTLS), observability in the form of tracing and metrics, and additional ones. In the case of multiple domains, apart from extending the previous features, service mesh technologies facilitate service-to-service communication across boundaries (i.e., among different clouds, clusters and premises and in the case of the Meta-OS-managed continuum, among different domains). The selection and implementation of the Service Mesh (and CNI) technologies have been based on the following project-level requirements (from D2.1):

- **REQ-COM-ETH-NF-M-04:** Following Cloud Native paradigm.
- **REQ-COM-DES-F-M-20**: Access to performance metrics from the cloud continuum.
- **REQ-INFRA-NET-NF-M-08**: Service visibility.
- **REQ-INFRA-NET-NF-M-10**: Secure network connectivity.

The multi-domain service mesh implementation complements the operation of the Meta-OS, providing a curated set of Cloud Native features yet keeping them decoupled to smooth their maintenance. This module roots on **Cilium**, which serves both as a K8s CNI plugin (L3, L4) and service-mesh technology. The following features have been implemented for the continuum:

- Rel-A: Cilium as essential intra-domain CNI and service mesh technology (including Hubble).
- Rel-B: Bidirectional VPN to establish direct connectivity among the managed IEs.
- Rel-B: Cluster mesh and Multi-Cluster Service (MCS) API capabilities to allow reaching delocalized services by name, independently of their location.
- Rel-B: Enablement of multi-domain network security rules. Essential ones pre-implemented:
    - No inter/multi-domain service-to-service by default (otherwise, several security breaches would be opened).
    - Meta-OS orchestration reachable from SAFE-6G services (OpenCAPIF, TFs, MLOps) regardless of their position in the continuum.
    - Metaverse manager & Network Core APIs – Network Exposure Function (NEF), Network Data Analytics Function (NWDAF), Cumucore core Network Configuration (CNC) – also reachable by TFs.

Cilium[4] (version >=1.19.1) was adopted for the service mesh implementation. This tool has an Open source license (Apache 2.0), and requires at least 0.2 vCPU and 400 MB RAM to operate on each computing node (Cilium agent with Hubble), plus one cluster-level service of 0.1 vCPU and 100 MB (Cilium operator). If clusters host several services and heavy traffic, additional resources might be required. Apart from intra-cluster capabilities, it enables service-to-service communication among services instantiated in different clusters/domain. In SAFE-6G, this feature has been implemented for now in UPV & NCSRD domains.

The multi-domain service manifests implemented in SAFE-6G take advantage of the enhanced continuum data model presented in Section 2.1.1 to simplify its implementation, as TFs are labelled by the Meta-OS at deployment time (if properly indicated). Specifically, the Meta-OS (through its LLO) creates by default a *ServiceExport* that enables services belonging to a specific TF to be accessed by the rest of its components (e.g., the AI Agent could access its related vApp or nApp despite these latter being deployed on a different domain; or vice versa). A *ServiceExport* is a simple manifest defined as follows:

```
apiVersion: multicluster.x-k8s.io/v1beta1
kind: ServiceExport
metadata:
   name: {{aerios-service-component-name}}
   namespace: {{service-namespace}}
```

It should be reminded that an aeriOS service component might be a Helm chart, and thus it might be composed of more than one service. The LLO must resolve their K8s service name and then deploy the needed manifests.

Besides, Cilium offers an extensive API (not REST) to interact with the offered features, such as the instantiation of network security rules. Particularly, Cilium exposes *CiliumNetworkPolicy* and *CiliumClusterwideNetworkPolicy* as Custom Resource Definitions (CRDs) for L3-L7 policy enforcement. TFs can therefore reinforce such policies, either including dedicated manifests at deployment time (static) or by integrating a controller for K8s that manage them based on specific needs (dynamic).

[4] Cilium documentation: https://docs.cilium.io/en/stable/

The following manifest provides an example of label-based network security for ingress traffic (egress could be set as well), allowing the Chatbot API to be consumed only by the Metaverse Manager, the Cognitive Coordinator and Prometheus (in case it had an exporter). They are not implemented by default, but could be reinforced by a SAFE-6G TF.

```
apiVersion: cilium.io/v2
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: chatbot-api-ingress-allow-metaverse-manager
spec:
  description: >
    Cluster-wide rule: Chatbot API pods only accept HTTPS traffic from
    the Metaverse Manager, while preserving monitoring/observability access.

  endpointSelector:
    matchLabels:
      app.kubernetes.io/name: {{aerios-service-component-name-for-chatbot-api}}

  ingress:
    - fromEndpoints:
        - matchLabels:
            io.kubernetes.pod.namespace: metaverse
            app.kubernetes.io/name: {{aerios-service-component-name-for-metaverse-
            -manager}}
      toPorts:
        - ports:
            - port: "443"
              protocol: TCP

    - fromEndpoints:
        - matchLabels:
            io.kubernetes.pod.namespace: default
            app.kubernetes.io/name: {{aerios-service-component-name-for-CoCo-api}}
      toPorts:
        - ports:
            - port: "443"
              protocol: TCP

    - fromEndpoints:
        - matchLabels:
            io.kubernetes.pod.namespace: monitoring
            app.kubernetes.io/name: prometheus
      toPorts:
        - ports:
            - port: "9090"
              protocol: TCP
```

3 DataOps Implementation

DataOps is the module of the SAFE-6G framework that **collects, stores and exposes** real and simulated data from the underlying architecture layers (i.e., infrastructure/continuum, core network and application) so that other components of the framework such as MLOps or TFs' components can consume them. The DataOps provides:

- Standard conventions to capture metrics and logs in Cloud Native ecosystems.
- Dedicated exporters tailored to the needs of the SAFE-6G ecosystem, including Meta-OS, Metaverse manager, core and Next Generation NodeB (gNB) exporters (plus commonly-used ones).
- Dedicated simulation capabilities to generate synthetic datasets of the network for specific scenarios (Digital Twin to be reported in D5.2).

The updated high-level view of the DataOps module, which components are described in D3.1, is depicted in Figure 8. Three updates are worth to be commented: (i) A dedicated logging service ("Log collector") is now part of all domains, to smooth debugging of deployments and running services; (ii) the No Structured Query Language (NoSQL) components (Long-term NoSQL DB and distributed NoSQL DBs) have been discarded as a unified time-series-based architecture has been agreed as more optimal and easier to maintain, and (iii) the O-NEF component and role has been taken over by the Digital Twin, formally part of this module despite being implemented in WP5 – thus not reported in this deliverable.

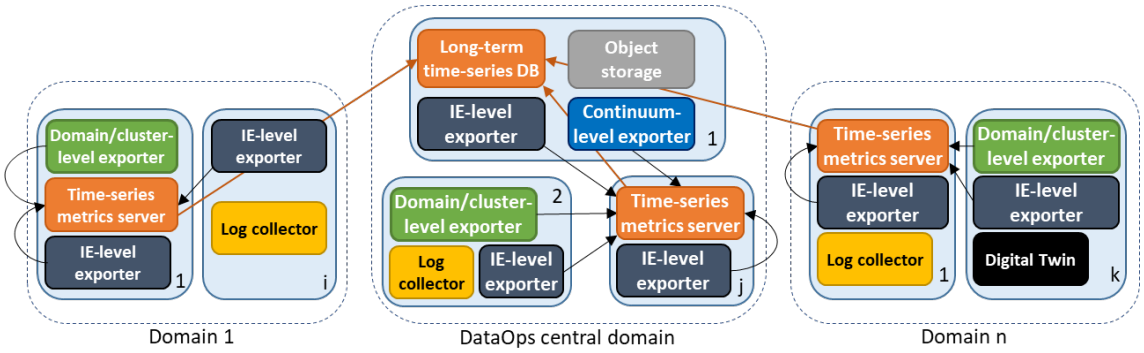

Figure 8: High-level view of SAFE-6G DataOps[5]

In this architecture, metrics and telemetry exporters can capture data at IE, domain, cluster or continuum levels, forwarding them to their respective domain's time-series metrics server. The latter can be configured to pull exporters' metrics with specific frequency, as well as retaining them for a specific time period, therefore it should be designed carefully depending on the variability of the managed data, its importance for consumer services or applications, and the available storage. Domain-level metrics are exposed over secured inter-domain gRPC Remote Procedure Call (gRPC) interfaces and persisted in a long-term metrics store, so that key data is available for advanced analytics or to be used by the MLOps pipelines for training models. Besides, logs are collected and managed at domain level, not centralised in this case to minimize network consumption.

[5] A few notes on the high-level DataOps view: (i) the digital twin could be instantiated in any domain with enough resources; (ii) domain or cluster-level exporters as well as log collectors could be deployed in principle on any computing node/IE of their respective domains, as the diagram just aims at depicting that one instance is available.

The DataOps infrastructure is based on the Prometheus stack and widely-used tools of the Cloud Native ecosystem. The main **features** are described here:

- Rel-A: Collection and local storage of metrics from Prometheus-compatible exporters (see Section 3.1).
- Rel-A: Centralised, long-term gathering of metrics from the distributed domains (varying retention times depending on the managed metrics).
- Rel-A: Implementation of a custom Prometheus-compatible Meta-OS exporter.
- Rel-A: implementation of a 5G core exporter for NoSQL/tabular database (discarded in Rel-B).
- Rel-A: Metrics accessible through the distributed and central Grafana dashboards.
- Rel-B: Collection and aggregation of Logs at domain level.
- Rel-B: Logs accessible through the distributed Grafana dashboards.
- Rel-B: Implementation of a custom Prometheus-compatible gNB exporter for Baicells integrated base station gNB considered for the trials.
- Rel-B: Implementation of a custom Prometheus-compatible Metaverse manager exporter.
- Rel-B: Metrics exposed and accessible via Meta-OS API Gateway, with valid credentials.

The selection and developments of the DataOps components are related to the following project-level requirements (from D2.1):

- **REQ-COM-DES-F-M-04:** Following Cloud Native paradigm.
- **REQ-COM-RES-F-M-07**: Data availability.
- **REQ-COM-DES-F-M-19**: Access to performance metrics from 5G/6G core infrastructure.
- **REQ-COM-DES-F-M-20**: Access to performance metrics from the cloud continuum.
- **REQ-COM-DES-F-M-21**: Access to performance metrics from user application.
- **REQ-COM-RES-NF-M-26**: Data persistence.
- **REQ-INFRA-META-NF-M-05**: Data accessibility across the computing continuum.

The basic components include:

- A **Prometheus server**[6] on each domain, acting as time-series metrics server.
- One central **Thanos**[7] **server** on top of Ceph S3 object storage, as long-term time-series database (with the data collected by the Prometheus servers). Additionally, a **Thanos sidecar** connected to each Prometheus, facilitating the shipping of metrics to the central server.
- A **Loki**[8] log aggregation system, deployed on each domain to collect all relevant logs.
- While all metrics can be retrieved from the distributed Prometheus servers or Thanos server via PromQL API calls (and logs managed by Loki through LogQL, based on the previous), a **Grafana**[9] dashboard is deployed on each domain, enabling their graphical representation for smooth user experience.
- Plus a set of metrics exporters on each IE/domain or continuum, presented in Section 3.1.

[6] Prometheus documentation: https://prometheus.io/docs/introduction/overview/
[7] Thanos documentation: https://thanos.io/tip/thanos/getting-started.md/
[8] Loki documentation: https://grafana.com/docs/loki/latest/
[9] Grafana documentation: https://grafana.com/docs/grafana/latest/

These tools have an Open-source license (Apache 2.0). The minimum resources for the basic monitoring and logging stack to work (Prometheus, Loki, Grafana and Thanos sidecar) is around 1.2-1.5 CPU and 1.5-2GB of RAM per domain, extended by 1 CPU and 1 GB of RAM in the domain where the central Thanos components are deployed. The API contains the same four endpoints stated in D3.1, now including the **/thanos** prefix, being now exposed through the Meta-OS API Gateway for external consumption.

The specific resources created for the project are documented in this repository: https://gitlab.com/safe-6g/development/dataops/monitoring, focusing on the overall architecture, deployment and configuration of the previous components.

3.1 EXPORTERS

The Consortium considered a balanced approach of exporters, leveraging both existing exporters for widely-adopted technologies and *de facto* standards as well as custom-made ones for particular SAFE-6G components or underlying modules when no exporters where available. The current list of Prometheus targets deployed by default can be found in this link: https://gitlab.com/safe-6g/development/dataops/monitoring/-/blob/main/docs/targets.md?ref_type=heads; with some of the implemented exporters (those from the next subsection) only instantiated where the monitored target is available.

3.1.1 IMPLEMENTED

The following exporters have been implemented during the action:

- **Data Fabric exporter** (one for the continuum): Prometheus-compatible exporter to retrieve continuum-level data managed by the Meta-Operating system.

Category	***Status***
Version	1.0.0
Documentation	Deployment, configuration and use instructions can be found here: https://gitlab.com/safe-6g/development/dataops/data-fabric-exporter/-/blob/main/README.md?ref_type=heads
Source code	https://gitlab.com/safe-6g/development/dataops/data-fabric-exporter
Docker images	registry.gitlab.com/safe-6g/development/continuum/datafabric-exporter:1.0.0
Helm charts	datafabric-exporter-1.0.0.tgz
Open API Swagger	NA
Demo	NA
Deployment requirements	0.050 vCPU, 64 MB RAM
Dependencies	It requires at least a Prometheus server in the domain to receive the data, in turn collected from the Meta-OS' Context broker that must be working
License	Apache 2.0

Table 6: Data fabric final release summary

- **CMC 5G Core exporter** (per network core): NoSQL/tabular exporter to retrieve metrics from a Cumucore core[10] implementation.

Category	*Status*
Version	1.0.1
Documentation	Configuration and use instructions can be found here: https://gitlab.com/safe-6g/development/dataops/cnc-exporter/-/blob/main/README.md?ref_type=heads
Source code	https://gitlab.com/safe-6g/development/dataops/cnc-exporter/-/tree/main/applications/cnc-exporter?ref_type=heads
Docker images	registry.gitlab.com/safe-6g/development/dataops/cnc-exporter/exporter:1.0.1
Helm charts	NA
Open API Swagger	NA
Demo	NA
Deployment requirements	0.100 vCPU, 128 MB RAM
Dependencies	It requires at least a MongoDB to receive the data, in turn collected from a core network that must be working (implemented for CMC core models)
License	Apache 2.0

Table 7: CMC Core exporter final release summary

- **Metaverse manager exporter** (per Metaverse manager service): Prometheus-compatible exporter to capture metrics from the target use cases, through its Metaverse manager API.

Category	*Status*
Version	1.0.1
Documentation	Deployment, configuration and use instructions can be found here: https://gitlab.com/safe-6g/development/dataops/metaverse-exporter/-/blob/main/README.md?ref_type=heads
Source code	https://gitlab.com/safe-6g/development/dataops/metaverse-exporter
Docker images	registry.gitlab.com/safe-6g/development/dataops/monitoring/metaverse-manager
Helm charts	metaverse-exporter-1.0.0.tgz
Open API Swagger	NA
Demo	NA
Deployment requirements	0.050 vCPU, 64 MB RAM
Dependencies	It requires at least a Prometheus server in the domain to receive the data, in turn collected from a Metaverse manager (API) that must be working
License	Apache 2.0

Table 8: Metaverse manager exporter final release summary

- **gNB performance exporter** (per gNB): Prometheus-compatible exporter to retrieve metrics from the radio nodes leveraged in the experiments.

[10] Cumucore core documentation: https://cumucore.com/mobile-private-network/#5g-lte-network

Category	*Status*
Version	2.0.0
Documentation	Deployment, configuration and use instructions can be found here: https://gitlab.com/safe-6g/development/resilience-function/gnb-performance-exporter/-/blob/main/README.md?ref_type=heads
Source code	https://gitlab.com/safe-6g/development/resilience-function/gnb-performance-exporter
Docker images	safe6g/gnb-performance-exporter:v2.0.0
Helm charts	NA, but can be built based on this code: https://gitlab.com/safe-6g/development/resilience-function/gnb-performance-exporter/-/tree/main/helm/gnb-performance-exporter?ref_type=heads
Open API Swagger	NA
Demo	NA
Deployment requirements	0.050 vCPU, 64 MB RAM
Dependencies	It requires at least a Prometheus server in the domain to receive the data, in turn collected from a gNB that must be working (implemented for Baicells models)
License	Apache 2.0

Table 9: gNB performance exporter final release summary

3.1.2 ADOPTED

In addition, the following widely-used exporters have been adopted:

- **Cilium Hubble exporter**[11] (per IE): Built-in component of Cilium, converts Hubble's real-time network flow data into Prometheus-compatible formats.
- **Kepler**[12] (per IE): It probes energy-related system stats and exports as Prometheus metrics.
- **Rook Ceph exporter**[13] (one for the continuum / Thanos host): It scrapes metadata related to the long-term object storage hosting the Thanos server.
- **Kubelet exporter**[14] (per IE): It translates information of kubelet summary API into Prometheus format, providing resource usage metrics such CPU or memory.
- **Kube State Metrics**[15] (per cluster): It generates and exposes cluster-level metrics from the Kubernetes API server, focusing on the health and state of Kubernetes objects themselves.
- **Node exporter**[16] (per IE): It exposes a wide variety of hardware- and kernel-related metrics.

For deploying these IE-level exporters, a minimum footprint of around 0.100 vCPU and 300-400 MB of RAM per IE/computing node is sufficient, being Kepler the most intensive one. The cluster-level kube-state-metrics exporter requires very low resources from the hosted device (just 0.010 vCPU and

[11] Hubble exporter documentation: https://docs.cilium.io/en/stable/observability/metrics/#metrics-reference
[12] Kepler documentation: https://sustainable-computing.io/kepler/design/metrics/
[13] Rook Ceph exporter documentation: https://rook.io/docs/rook/latest/Storage-Configuration/Monitoring/ceph-monitoring/#updates-and-upgrades
[14] Kubelet exporter documentation: https://grafana.com/docs/alloy/latest/reference/components/discovery/discovery.kubelet/
[15] Kube state metrics documentation: https://github.com/kubernetes/kube-state-metrics
[16] Node exporter documentation: https://github.com/prometheus/node_exporter

32 MB RAM), while the Rook Ceph exporter requires a bit more (0.050 vCPU and 50 MB RAM) but just one for the entire continuum is needed. These figures are the minimum required resources, which can increase depending on the clusters size and number of managed services.

3.2 GRAFANA DASHBOARDS

Several Grafana dashboards have been prepared for providing SAFE-6G adopters specific graphical information related to the managed infrastructure, services and applications. Some of the prepared graphical interfaces can be imported from this repository: https://gitlab.com/safe-6g/development/dataops/monitoring/-/tree/main/dashboard/grafana?ref_type=heads.

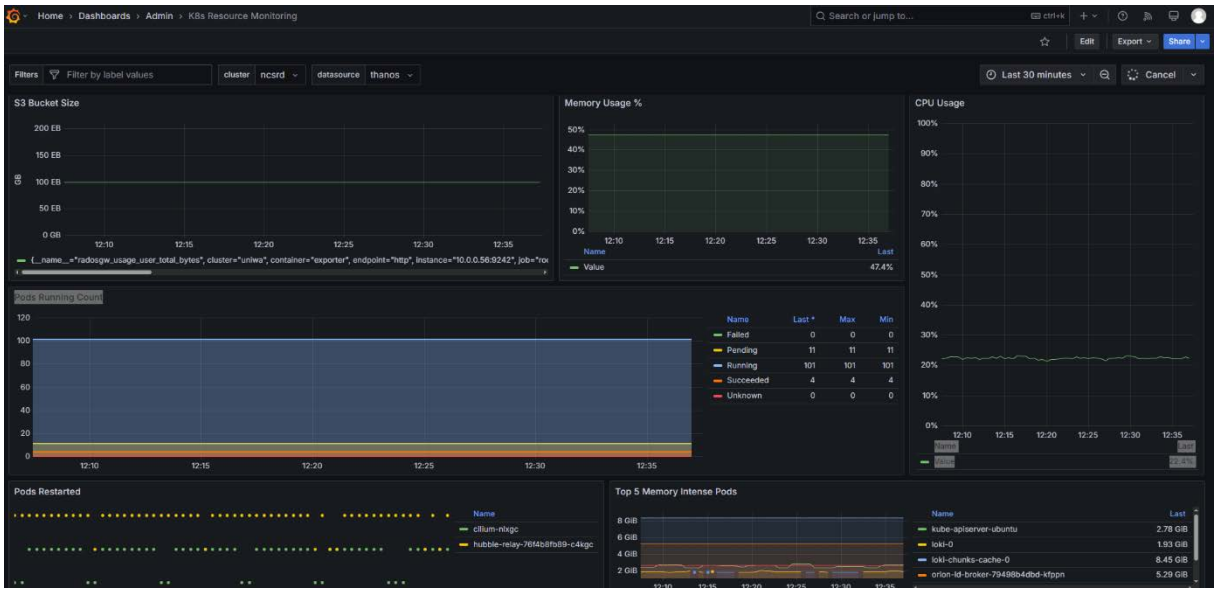

Figure 9: Kubernetes resources monitoring dashboard

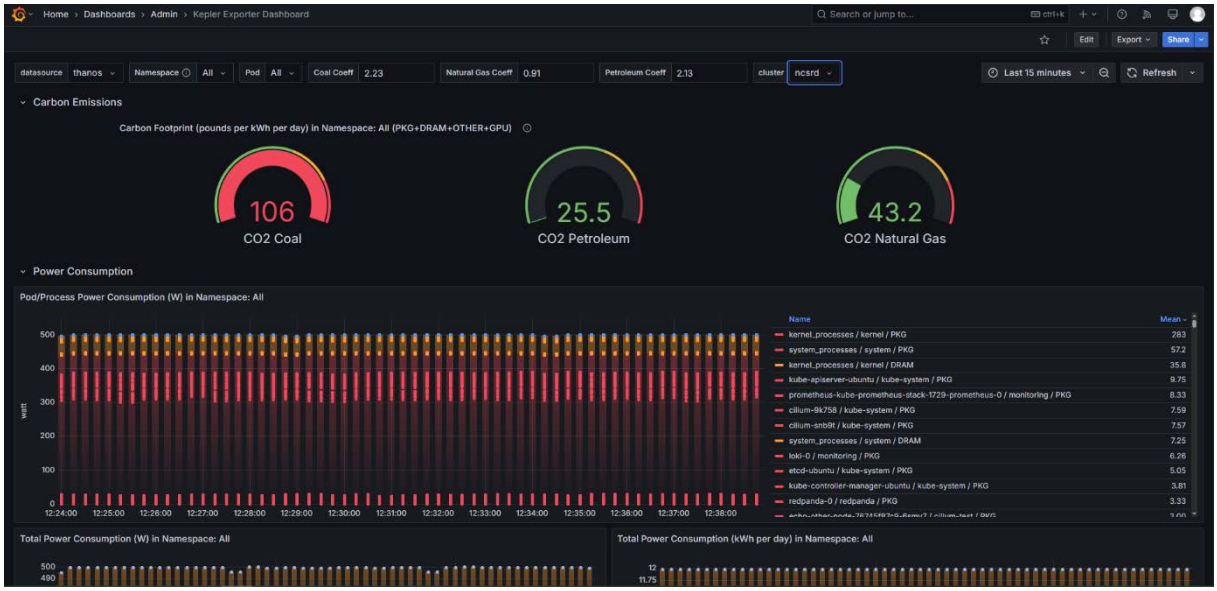

Figure 10: Power consumption monitoring dashboard

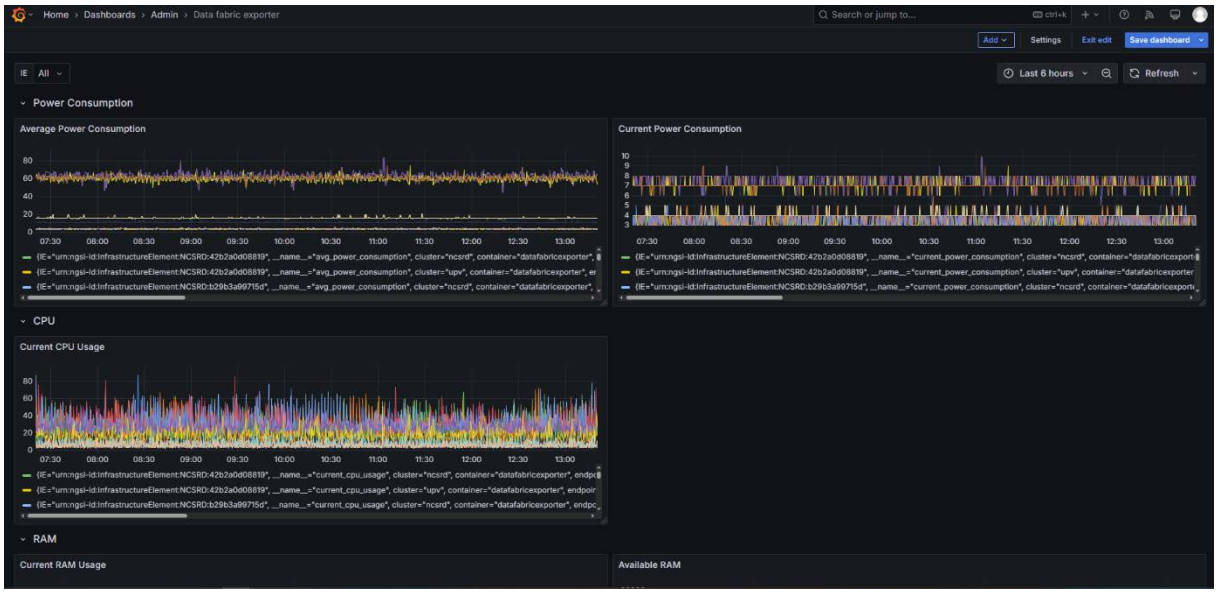

Figure 11: Meta-OS Data Fabric monitoring dashboard

4 CONTINUUM INSTALLATION PROCESS

In order to install all the software modules that are part of the Continuum ecosystem, supporting scripts and manifests have been prepared. This section depicts the main assets prepared to smooth the process, which consists of 5 steps:

1. **Environment preparation** (script: https://gitlab.com/safe-6g/development/continuum/-/tree/main/deployment/infrastructure?ref_type=heads). Before deploying the modules presented in the previous sections, the domains have to be prepared. They will consider:
    - Kubeadm for K8s cluster creation and configuration – with different PodCIDR range per domain.
    - Docker with containerd as Container Runtime Interface (CRI).
    - Helm as deployment manager.
    - OpenEBS as storage class.
    - Cilium as CNI – not configured for multi-cluster.
    - Hubble as virtualized network observability.
2. **Meta-OS deployment** (guided process: https://gitlab.com/safe-6g/development/continuum/-/blob/main/deployment/README.md?ref_type=heads). Guides the user through the installation and configuration of the Meta-OS, which considers deployments per IE/computing node, cluster/domain and continuum.
    - In the entrypoint domain:
        i. CB (Orion-LD) + configurations.
        ii. Federator (aerOS Federator).
        iii. Identity management (Keycloak and OpenLDAP) + configurations.
        iv. OpenAPI Gateway (KrakenD).
        v. Self-* modules.
        vi. Management portal.
        vii. HLO and microservices bus (Kafka).
        viii. LLO/s.
    - In other domains, the identity management and the management portal are not required.
3. **DataOps deployment** (guided process: https://gitlab.com/safe-6g/development/dataops/monitoring/-/blob/main/README.md?ref_type=heads). Simplifies the implementation of the monitoring, logging components, plus key exporters along the managed computing continuum. It includes:
    - The Prometheus stack (including Prometheus server, Prometheus Alert Manager, Grafana, plus kubelet, kube-state-metrics and node exporters).
    - Thanos main server in one domain, and sidecars in the others.
    - Logging stack (Loki and Promtail).
    - Default Grafana dashboards.
    - Exporters: Cilium Hubble, Data Fabric exporter, Rook Ceph exporter, Kepler.
    - Remaining exporters (i.e., for the Metaverse manager, core and gNB) have to be manually installed in a domain with access to the related devices/services.

4. **Meta-OS' APIs registration in OpenCAPIF**, by sending the API Gateway file to the CAPIF management team. Some DataOps APIs are also exposed via Meta-OS API Gateway.
5. **Cluster-mesh preparation and configuration**. Depends on the networking configuration and conditions, for SAFE-6G the provisioning actions (https://gitlab.com/safe-6g/development/continuum/-/blob/main/deployment/infrastructure/multi_domain_service_mesh.md) have to be performed.

5 CONCLUSION

This document reported the outcomes of T3.1, mostly related to the final release of the software assets developed, tailored or updated, related to the cloud-edge continuum and the DataOps. The final set of main features and considered requirements have been summarised, having included also a curated set of external links pointing out to relevant documentation, code and packages required to facilitate the replication of the modules in other distributed computing environments. While with this report the activities of the task can be considered completed, some refinements and bug fixing might be required during the final integration and testing phases executed in WP5.