D3.1

FIRST VERSION OF THE COP-PILOT PLATFORM

Report on the initial version of the COP-PILOT platform with references to the software components, open-source communities, and related standards

Revision: v1.0

D3.1: First Version of the COP-PILOT Platform

Work package	WP3
Task	T3.1- T3.5
Due date	30/06/2026
Submission date	30/06/2026
Deliverable lead	Georgios P. Katsikas (UBITECH)
Version	1.0
Authors	Georgios P. Katsikas, Dimitrios Manolopoulos, Dimitrios Klonidis (UBITECH), Kostis Trantzas (UOP), Christos Tranoris (PNET), Konstantinos Fragkos, Panos Mantzakos (NETC), Philip Griffiths (TATA), Benjamin Ertl (AGE), Jesus Iglesias, Emilio Garrido (TID), Sergio Figueiredo (IPN), Dionysios Skordoulis (AXON), Spiros Kousouris (SUITE5), Epameinondas Koutavelis (KON), Luis Ferreira, Luis Rosa (ONE), Anastasis Tzoumpas (NOVA)
Reviewers	Luis Rosa (ONE)
Abstract	This deliverable presents the first version of the COP-PILOT platform, developed under Work Package 3 as part of the Horizon Europe COP-PILOT. It describes the design, implementation, and initial integration of the platform's five core components: the Business Management Portal, the End-to-End Service Orchestrator, the Domain Orchestrator, the Data Management platform, and the Secure Integration Fabric. Together, these form a hierarchical orchestration architecture for managing collaborative services across heterogeneous, geo-distributed IoT-edge-core environments, built on open-source technologies and standardised interfaces including TMForum APIs and ETSI NGSI-LD. The deliverable further documents the CI/CD stack supporting automated deployment across all components and piloting clusters, provides an overview of the closed-loop platform services identified during the first project phase, and assesses the platform's early impact in terms of open-source contributions and standards alignment.
Keywords	Edge computing, IoT orchestration, service orchestration, multi-domain management, compute continuum, ETSI OpenSlice, TMForum APIs, ETSI NGSI-LD, zero-trust networking, secure integration fabric, closed-loop automation, LLM-assisted service ordering, CI/CD automation, open-source platform, 5G, Kubernetes, cloud-native.

Co-funded by the European Union

Document Revision History

Version	Date	Description of change	List of contributors
V0.1	16/02/2026	Table of Contents	*Georgios P. Katsikas (UBITECH)*
V0.2	06/05/2026	Section 2 “Alignment with the final COP-PILOT architecture” Section 3.2 “End-to-end Service Orchestrator” Section 5 “Impact” structure and ESO part	Georgios P. Katsikas (UBITECH)
V0.3	14/05/2026	Section 4.1-4.3 “Platform Integration”	Konstantinos Fragkos, Panos Mantzakos (NETC)
V0.4	18/05/2026	Section 3.5 “Secure Integration Fabric” Section 5 “Impact” related to SIF	Philip Griffiths (TATA)
		Section 3.4 “Data Management” Section 5 “Impact” related to Data Mgmt.	Georgios P. Katsikas (UBITECH)
		Section 3.4 “Data Management” dashboards	Spiros Kousouris (SUITE5) Epameinondas Koutavelis (KON)
		Section 3.3 “Domain Orchestrator” Section 5 “Impact” related to DO	Kostis Trantzas (UOP), Christos Tranoris (PNET)
		Section 3.1 “Business Mgmt. Portal” Section 5 “Impact” related to BMP	Benjamin Ertl (AGE) Jesus Iglesias (TID) Sergio Figueiredo (IPN) Spiros Kousouris (SUITE5)
V0.5	21/05/2026	Section 4.4 “Vertical Services Integration”	Dionysios Skordoulis (AXON)
		Abstract, Executive Summary, Section 1 (Introduction), Section 6 (Conclusion)	Dimitrios Manolopoulos (UBITECH)
V0.6	12/06/2026	Final editing. Sent for internal review	Georgios P. Katsikas (UBITECH)
V0.7	24/06/2026	Internal review	Luis Rosa (ONE)
V1.0	30/06/2026	Addressed review comments	Georgios P. Katsikas (UBITECH)
		Submission	Ioanna Drigkopoulou (NETC)

Grant Agreement No: 101189819 | **Topic:** HORIZON-CL4-2024-DATA-01-03
Call: HORIZON-CL4-2024-DATA-01 | **Type of action:** HORIZON-IA

DISCLAIMER

Co-funded by the European Union

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI

Co-funded by the European Union (COP-PILOT, 101189819). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. This work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI).

COPYRIGHT NOTICE

Project Co-funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	R	
Dissemination Level		
PU	Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)	X
Classified R-UE/ EU-R	EU RESTRICTED under the Commission Decision No2015/ 444	
Classified C-UE/ EU-C	EU CONFIDENTIAL under the Commission Decision No2015/ 444	
Classified S-UE/ EU-S	EU SECRET under the Commission Decision No2015/ 444	

* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

DATA: Data sets, microdata, etc.

DMP: Data management plan

ETHICS: Deliverables related to ethics issues.

SECURITY: Deliverables related to security issues

OTHER: Software, technical diagram, algorithms, models, etc.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The COP-PILOT project aims to develop and validate an open, collaborative orchestration platform for deploying and managing services across heterogeneous IoT-edge-core computing environments on a European scale. This deliverable, D3.1, reports on the first version of the COP-PILOT platform as developed under Work Package 3, covering the period from project inception through Month 18.

The platform is structured around five tightly integrated core components, each addressing a distinct layer of the overall architecture. The **End-to-End Service Orchestrator (ESO)**, instantiated as ETSI HypO under the ETSI SDG OpenSlice community, provides centralised multi-domain service lifecycle management through standardised TMForum APIs, enabling stakeholders to design, order, and monitor services spanning multiple geo-distributed domains. The **Domain Orchestrator (DO)**, based on ETSI SDG OpenSlice and ColonyOS, operates at the domain level, exposing compute and network resources as-a-service to the ESO through a consistent set of northbound interfaces. The **Data Management (DM)** platform, built on the FIWARE Orion-LD Context Broker and Eclipse Arrowhead, handles IoT data ingestion, persistence, and federation across clusters using the ETSI NGSI-LD standard. The **Secure Integration Fabric (SIF)**, implemented using OpenZiti, provides identity-first, zero-trust connectivity across all platform components and cluster domains, eliminating the need for traditional VPN-based integration and enabling policy-driven, encrypted service exposure. Finally, the **Business Management Portal (BMP)** offers a unified product marketplace and an LLM-driven assistant that translates stakeholder intents into actionable orchestration operations, significantly lowering the barrier to platform adoption.

Co-funded by the European Union

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Term	Description
3GPP	3rd Generation Partnership Project
AI	Artificial Intelligence
API	Application Programming Interface
BM-L	Business Management Layer
BMP	Business Management Portal
BSS	Business Support System
CD	Continuous Delivery
CI	Continuous Integration
DC	Datacenter
DDO-L	Distributed Domain Orchestration Layer
DIS-L	Distributed Infrastructure Services Layer
DM	Data Management
DO	Domain Orchestrator
ESO	End-to-end Service Orchestrator
ESO-L	End-to-end Service Orchestration Layer
ETSI	European Telecommunications Standards Institute
INFRA-L	Infrastructure Layer
IoT	Internet of Things
MCP	Model Context Protocol
MDG	Module Development Group
NGSI-LD	Next Generation Service Interfaces - Linked Data
OSS	Operations Support System
SDG	Software Development Group
SIF	Secure Integration Fabric
SIF-L	Secure Integration Fabric Layer
TMF	TM Forum

1. INTRODUCTION

The COP-PILOT project addresses a fundamental challenge in modern digital infrastructure: the need for an open, interoperable, and intelligent platform capable of orchestrating services across the highly fragmented landscape of IoT devices, edge computing nodes, and core cloud environments that characterises today's industrial sectors. As edge computing ecosystems grow in scale and complexity, the absence of a unified orchestration framework capable of spanning multiple administrative domains, integrating heterogeneous technologies, and automating service management at scale remains a critical barrier to realising the full potential of edge intelligence.

Work Package 3 (WP3) of the COP-PILOT project directly responds to this challenge by designing and implementing the core platform that underpins the entire COP-PILOT ecosystem. This deliverable, D3.1, reports on the first version of that platform, presenting the outcomes of the initial eighteen months of platform development and integration. It documents how the architectural vision established in the preceding design deliverables has been translated into a working, deployed system, and provides the technical foundation upon which the project's large-scale piloting and validation activities are built.

1.1 OBJECTIVE

This document constitutes Deliverable D3.1 of the COP-PILOT project, titled "First Version of the COP-PILOT Platform". It reports on the implementation, and initial integration of the COP-PILOT platform as developed within Work Package 3 during the first eighteen months of the project.

The primary objective of this deliverable is to present a first functional version of the COP-PILOT platform, demonstrating its core architectural components, the open-source technologies that realise them, the standardised interfaces they expose, and the integration mechanisms that bind them into a coherent, operational system. Beyond describing individual components, the deliverable demonstrates how the platform has been deployed and validated and how it connects with the five piloting clusters distributed across Europe. Finally, the deliverable provides evidence on impact made so far, through open-source contributions, implemented standards, and dissemination material.

1.2 RELATION WITH OTHER DELIVERABLES

D3.1 sits at the intersection of the project's design and implementation streams. It builds directly upon D2.1 "Ecosystem Definition and Requirements" and D2.2 "COP-PILOT Architecture and Functionalities", which together establish the functional and non-functional requirements, the reference architecture, and the technical specifications that WP3 is tasked with realising. Where D2.2 defines the abstract architectural blocks and their expected interfaces, D3.1 maps each of those blocks to concrete software components, documents their internal design, and provides evidence of their initial deployment and integration.

Looking forward, D3.1 feeds directly into WP4 and WP5 activities. The platform components described here form the foundation upon which the piloting clusters are built, and the use case experiments are executed. The present deliverable will itself be superseded by D3.2 "Final Version of the COP-PILOT Platform", which will report on the complete platform incorporating all advanced features, further cluster integrations, and the outcomes of the full validation cycle.

Co-funded by the European Union

1.3 DOCUMENT STRUCTURE

This document is organized in five (5) sections as follows:

- **Section 2** stablishes the **alignment between WP3** activities **and** the final COP-PILOT architecture (**WP2**), mapping each abstract architectural block defined in D2.2 to the concrete software components developed within WP3.
- **Section 3** introduces **platform components in detail**, covering its internal design, the technologies that underpin it, and the interfaces it exposes.
- **Section 4** describes the **platform-level integration** work, including the Central Management Domain, the COP-PILOT CI/CD stack, and the automated provisioning and interconnection workflows developed to bind the platform components together and connect them with the piloting clusters.
- **Section 5** assesses the **impact of the first platform release** in terms of open-source contributions, implemented standards, and early dissemination activities.
- **Section 6 concludes** the document and **outlines the roadmap** towards the final platform release to be reported in D3.2.
- Appendix A – CI/CD Stack Services reports on technical **Continuous Integration and Continuous Delivery (CI/CD) services** implemented for the **COP-PILOT platform integration**.
- Appendix B – CI/CD Platform Workflows visualizes some key **CI/CD platform workflows.**

2 ALIGNMENT WITH THE FINAL COP-PILOT ARCHITECTURE

This section demonstrates the alignment of WP3 activities with the final COP-PILOT architecture (Section 2.1) as well as the mapping of logical architecture components introduced in D2.1 [1] and updated in D2.2 [2] with actual software components that are designed and developed in WP3 (see Section 2.2).

2.1 MAPPING OF THE FINAL COP-PILOT ARCHITECTURE WITH WP3

The final version of the COP-PILOT architecture is introduced in D2.2 [2]. This section realizes a mapping of WP3 activities (highlighted in purple) with key blocks of the final version of the COP-PILOT architecture as shown in Figure 2-1. Around the WP3 activities highlighted in purple, WP4 manages both the lower part in Figure 2-1 (i.e., the interaction of the platform with the Cluster infrastructures) and the upper part (i.e., the interaction of the platform with the vertical sectors' stakeholders and applications).

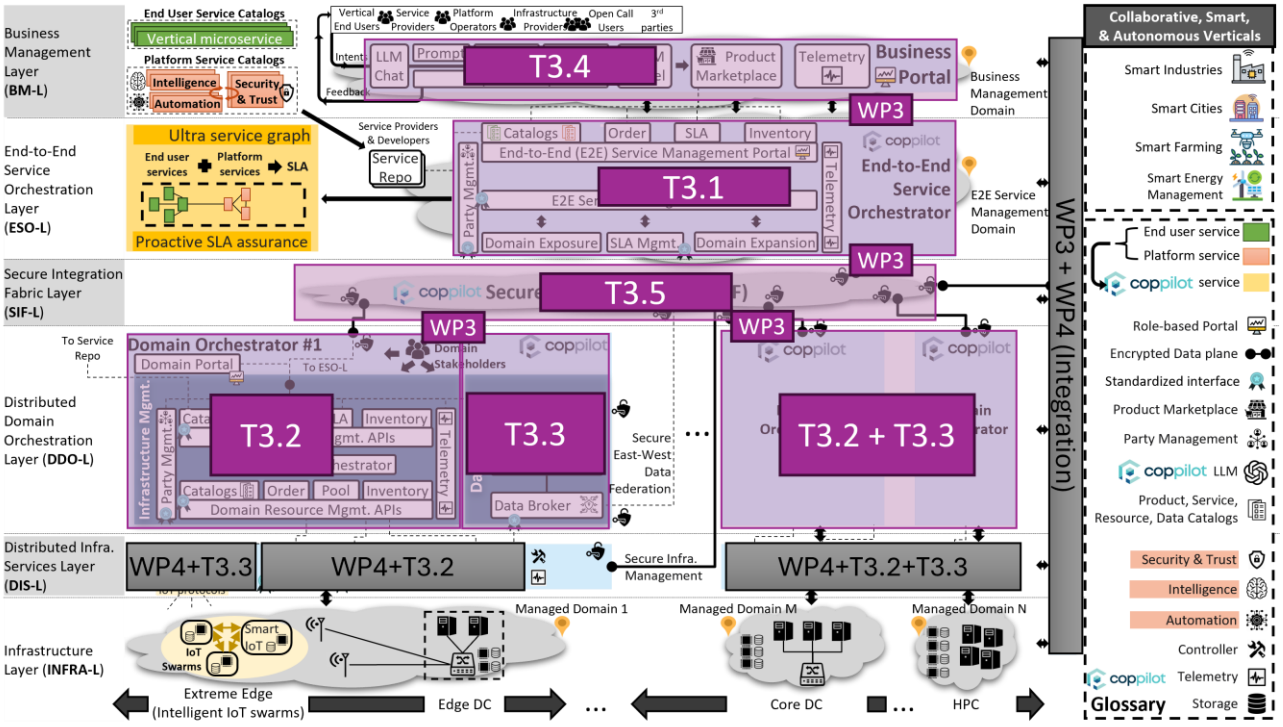

Figure 2-1: Mapping of WP3 activities to the final version of the COP-PILOT architecture.

The role of the WP3 tasks in the COP-PILOT ecosystem is outlined below on a per-task basis:

Task 3.1 is responsible for the COP-PILOT ESO-L, offering the end-to-end service orchestrator along with all its capabilities and APIs. This task ensures proper interfacing of the ESO with the DO (T3.2), SIF (T3.5), and Business Management Portal (T3.4) platform components for managing the entire COP-PILOT platform and its expansion to new domains.
Task 3.2 undertakes the resource and service management part of the COP-PILOT DDO-L, offering the domain-level orchestrator (i.e., the DO) and the integration of the DO with the underlying compute and network infrastructure services that reside in the DIS-L. This task ensures proper interfacing of edge-to-core compute resources, 5G, and transport network resources with the platform and exposure of these resources as-a-Service towards the ESO (T3.1).
Task 3.3 undertakes the data management part of the COP-PILOT DDO-L, offering a platform integrated with the underlying Data Connectors that reside in the DIS-L. This task ensures proper interfacing of all data spaces (e.g., sensors, legacy data sources) with the platform and exposure of data catalogues and services towards COP-PILOT applications.
Task 3.4 undertakes the development of the COP-PILOT Business Management Portal. The portal visualizes (i) the product marketplace (catalogues, categories, specifications) of COP-PILOT, (ii) allows ordering of certain products, and (iii) provides secure and explainable LLM assistants to facilitate product ordering towards the COP-PILOT Clusters. This task also ensures integration with both the ESO (T3.1) and SIF (T3.5) layers.
Task 3.5 is responsible for the COP-PILOT SIF-L, effectively developing the core part of the SIF and a highly distributed set of SIF clients across all COP-PILOT clusters and Open Call activities to create a secure ecosystem for collaborating COP-PILOT domains and services that belong to multiple stakeholders.

The integration among all these tasks as well as the COP-PILOT Clusters (managed by WP4) is a cross-cutting pillar (highlighted in grey) that consumes cycles across all WP3 and WP4 tasks. COP-PILOT materializes this pillar though a project-wide CI/CD platform.

2.2 MAPPING OF ABSTRACT BLOCKS TO SOFTWARE COMPONENTS

Table 2-1 displays a mapping of the abstract WP2 component names of the final COP-PILOT architecture - as defined in D2.1 [1] and finalized in D2.2 [2] - to concrete software names that realize the functional and non-functional specifications of these components in WP3.

Table 2-1: Mapping of COP-PILOT Architecture Blocks (WP2) to Platform Software Components (WP3).

COP-PILOT Abstract Architecture Block Name (WP2)	COP-PILOT Software Component Name (WP3)	Related open-source platform and community	Providers
Business Management Portal (BMP)	COP-PILOT Business Portal	ETSI SDG OpenSlice AG-UI	AGE, TID, SUITE5, IPN, ONE, UOP, UBI
End-to-End Service Orchestrator (ESO)	ETSI Hyper Orchestrator (HypO) as module under ETSI SDG OpenSlice	ETSI SDG OpenSlice	UBI (ETSI HypO MDG Leader), TATA (ETSI HypO participant), SDG OSL community
Domain Orchestrator (DO)	ETSI SDG OpenSlice ColonyOS	ETSI SDG OpenSlice ColonyOS	UOP (ETSI SDG OpenSlice Leader), SDG OSL community LTU (ColonyOS leader), ColonyOS community
Data Management (DM)	FIWARE Orion Context Broker FIWARE Scorpio Context Broker FIWARE Stellio Context Broker Eclipse Arrowhead	FIWARE Eclipse Foundation	CL2: UPV, TID CL3A: AGA, ILINK, AUA CL3E: ENIC CL4: ONE, TER, JIG CL1: LTU
Secure Integration Fabric (SIF)	OpenZiti	OpenZiti	TATA (OpenZiti leader), OpenZiti community
CI/CD Platform	COP-PILOT CI/CD Stack	Jenkins, Harbor, Keycloak, Portainer, GitHub, Docker, Kubernetes, Ansible, OpenTofu, NGINX	NETC, AXON

3 COP-PILOT'S HIERARCHICAL ORCHESTRATION PLATFORM (V1)

This section introduces an initial version of all COP-PILOT Platform components, covering the internal design of each component as well as the interfaces that each component implements. The following components are covered in the rest of this section:

- Section 3.1 introduces the COP-PILOT Business Management Portal (BMP) in the Business Management Layer (BM-L) of the COP-PILOT architecture.
- Section 3.2 describes the COP-PILOT End-to-End Service Orchestrator (ESO) in the End-to-end Service Orchestration Layer (ESO-L) of the COP-PILOT architecture.
- Sections 3.3 and 3.4 describe the COP-PILOT Domain Orchestrator (DO) and Data Management (DM) platform respectively, both residing in the Distributed Domain Orchestration Layer (DDO-L) of the COP-PILOT architecture.
- Section 3.5 introduces the COP-PILOT Secure Integration Fabric in the Secure Integration Fabric Layer (SIF-L) of the COP-PILOT architecture.
- Section 3.6 discusses how COP-PILOT approaches generic closed-loop services that extend the platform's capabilities with orchestratable actuation towards the vertical sectors' infrastructures.

3.1 BUSINESS MANAGEMENT PORTAL

This section introduces COP-PILOT's Business Management Portal (BMP). First, the initial design of the BMP is outlined in Section 3.1.1. Next, the technologies related to the implementation of the BMP are detailed in Section 3.1.2. Finally, the BMP interfaces are listed and briefly described in Section 3.1.3.

3.1.1 Design

The Business Management Portal (BMP) is the primary user-facing entry point of the COP-PILOT platform, providing a unified environment for managing services, applications, and infrastructure across multiple domains and clusters. It enables platform operators, service providers, developers, and end users to discover, order, monitor, and manage offerings through TM Forum-compliant interfaces while leveraging AI-assisted operations. By combining business management capabilities, AI-powered intent handling, observability dashboards, and integration with the COP-PILOT orchestration ecosystem, the BMP abstracts the complexity of the underlying distributed platform and offers a streamlined experience for operating cloud-edge services at scale. The BMP's abstract design is illustrated in Figure 3-1.

Co-funded by the European Union

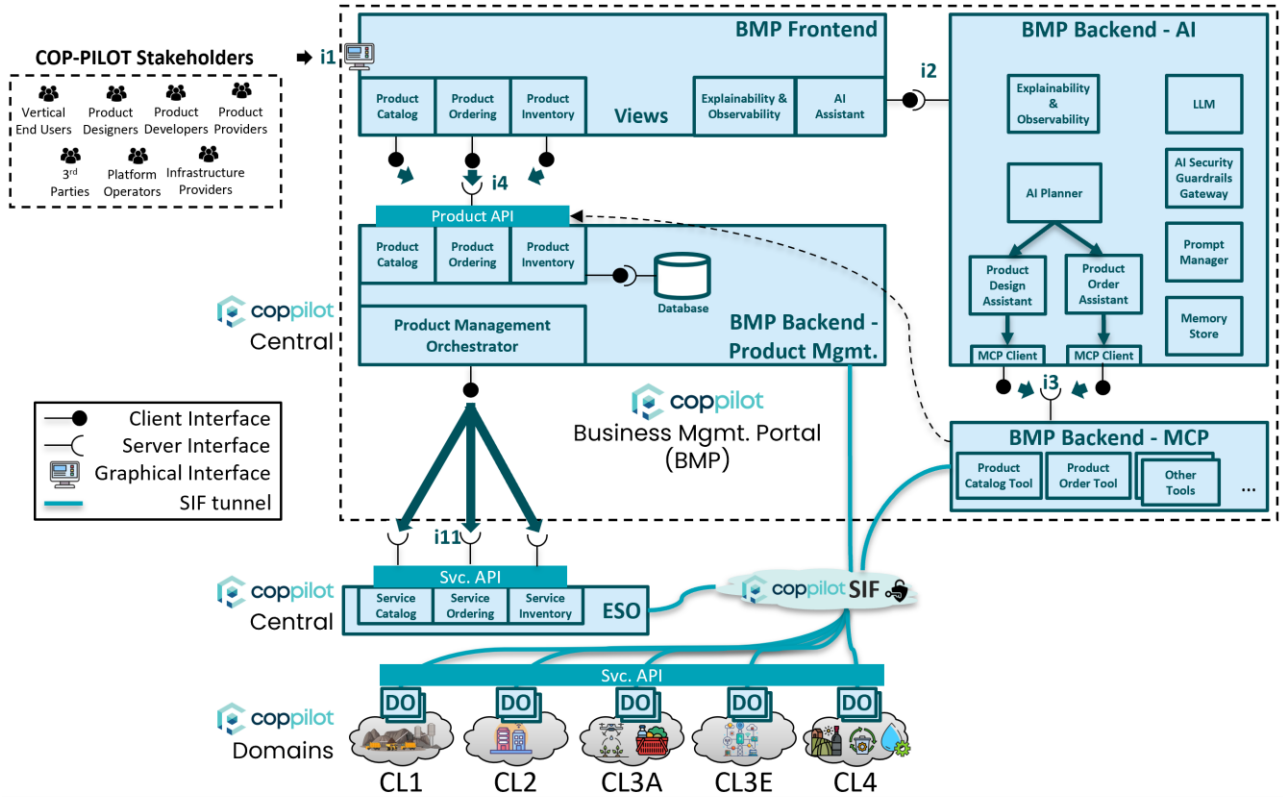

Figure 3-1: Abstract design of the COP-PILOT Business Management Portal.

The Business Management Portal (BMP) serves as the primary business-facing environment of the COP-PILOT platform, providing a unified interface through which stakeholders can discover, order, manage, and monitor products and services. The architecture depicted in Figure 3-1 is organised around four tightly integrated domains: the BMP Frontend and a disaggregated BMP Backend that comprises of the BMP Backend-AI, BMP Backend-MCP and BMP Backend-Product Management components.

From the perspective of end users, interaction with the platform begins in the BMP Frontend. This layer provides role-based dashboards and graphical user interfaces tailored to different stakeholder groups, including end users, product designers, platform operators, infrastructure providers, and third-party partners. Through a unified experience, users can browse available offerings, submit requests, manage existing subscriptions, and gain visibility into deployed products and services. The front-end combines traditional portal capabilities with conversational AI interactions, allowing users to engage with the platform either through standard graphical workflows or through natural language conversations with the AI Assistant. The AI Assistant in the BMP Frontend acts as the primary intelligent interaction point within the portal. Rather than requiring users to navigate complex catalogue structures or ordering processes manually, it enables stakeholders to discover products, request services, manage operational tasks, and obtain information through conversational exchanges. To ensure transparency and trust in these AI-supported interactions, the portal incorporates explainability and observability capabilities that provide visibility into AI decisions, orchestration activities, agent behaviour, tool usage, and operational outcomes. These capabilities allow users and operators to understand not only what actions were performed, but also why specific recommendations, decisions, or workflows were generated.

Co-funded by the European Union

Behind the user-facing layer, the BMP Backend components (i.e., AI, MCP, and Product Management) implement the business logic and management functions that support the portal. The BMP Backend-Product Management component maintains the product catalogue, manages product inventories, processes orders, and coordinates fulfilment activities. Product definitions and commercial offerings are managed through capabilities aligned with TM Forum Product Catalogue Management (TMF620), ensuring consistent representation and exposure of available services. Product orders are handled through TMF622-compliant interfaces, allowing business-level requests to be translated into fulfilment workflows while maintaining lifecycle visibility and governance. Inventory management capabilities aligned with TMF637 provide visibility into deployed products and active service instances, ensuring that both users and platform operators can track the current operational state of subscribed services.

A key responsibility of the BMP Backend-Product Management is the transformation of business requests into executable fulfilment actions. Once a product order has been accepted, the Product Management Orchestrator coordinates the fulfilment process by identifying the services required to realise the requested product and generating the corresponding service-level requests. These requests are then passed to the End-to-End Service Orchestrator (ESO) using standard TM Forum Service Ordering interfaces (TMF641), ensuring consistent communication between the business management layer and the underlying service orchestration environment. Throughout the lifecycle of an order, the BMP continuously monitors fulfilment progress and provides status updates to both users and AI services. Persistent information related to products, orders, operational metadata, user preferences, and AI interactions is maintained within the platform database layer.

Complementing these business management capabilities, the BMP Backend-AI introduces intelligent reasoning, planning, automation, and decision-support functions. At its core is the AI Planner, which interprets user requests, determines the required objectives, decomposes them into executable tasks, and coordinates the activities of specialised AI assistants. Acting as an orchestration intelligence layer, it selects the appropriate tools, identifies the most suitable assistants for each task, and consolidates results into coherent responses presented to the user.

Specialised assistants operate under the supervision of the AI Planner to support different business processes. The Product Design Assistant assists designers and administrators in creating and maintaining product offerings by generating specifications, recommending product structures, validating configurations, and supporting service-to-product mappings. The Product Order Assistant focuses on user-facing ordering workflows, guiding stakeholders through product selection, validating requests, generating service orders, and providing lifecycle tracking throughout fulfilment. These assistants rely on large language model (LLM) capabilities for natural language understanding, reasoning, knowledge retrieval, content generation, and workflow execution.

To ensure that AI-driven operations remain trustworthy and compliant, the BMP Backend-AI incorporates several governance and operational support functions. The AI Security Guardrails Gateway (AI-SGG) supports AI-assisted operational interactions, applying security controls to both inputs and outputs. Its responsibilities include prompt sanitisation, sensitive-data protection, validation of authorised operations, execution safeguards, confirmation verification for high-impact actions, output sanitisation, and security logging. In the current architecture, the AI Security Guardrails Gateway acts as a policy and validation control point for requests received from the AI assistant (crossing i2) and for tool invocations (prior to their triggering through i3). Its role is to reduce risks related to prompt injection, sensitive-data leakage, unauthorised operations, unsafe tool calls, malformed payloads, and absence of user confirmation in high-impact actions.

Co-funded by the European Union

Prompt management capabilities centralise reusable prompts, system instructions, workflow definitions, and configuration settings, ensuring consistent behaviour across all AI assistants. Persistent contextual information is maintained within a Memory Store that preserves conversation history, user preferences, prior actions, and relevant knowledge, enabling more personalised and context-aware interactions over time.

All AI assistants access platform functionality through a Model Context Protocol (MCP) integration layer in the BMP Backend-MCP. This component provides a standardised mechanism through which AI services can invoke platform capabilities exposed as reusable tools. Examples include Product Catalogue tools that support product discovery, recommendation, metadata retrieval, and catalogue navigation, as well as Product Order tools that enable order creation, modification, cancellation, lifecycle tracking, and status retrieval. Additional tools may be incorporated over time, including inventory management, monitoring, observability, orchestration, intent management, security, compliance, and domain-specific operational capabilities. This tool-based approach ensures modularity, extensibility, and interoperability across the evolving AI ecosystem.

To further strengthen operational transparency, the architecture includes an LLM Observability and Explainability Dashboard positioned alongside the AI services. This capability provides continuous oversight of AI-assisted workflows by collecting, analysing, and visualising quality and performance information related to LLM and agent execution. It captures agent execution timelines, tool invocations, AI-generated outputs, evaluation results, structured judgements, and user feedback, enabling platform operators and AI administrators to monitor behaviour, identify quality degradation, support auditing activities, and continuously improve AI performance. By combining explainability, observability, governance, and operational analytics, the dashboard contributes to the creation of trustworthy, accountable, and continuously optimised AI-enabled platform operations.

Together, these architectural layers establish a seamless connection between business users, AI-powered assistance, and the underlying orchestration infrastructure, enabling the COP-PILOT platform to deliver products and services through a consistent, scalable, and intelligent operational framework.

3.1.1.1 Product Marketplace

End user requests map to TMF products in a catalogue, while users can order these products from the underlying orchestrators. The portal's back-end maps intents to products to services offered by the ESO and issues the corresponding service orders towards the COP-PILOT orchestration platform.

3.1.1.2 Cluster Observability Portals

The following cluster specific portals will be linked with the BMP:

- **CL1:** Bemis UI, Nucleus UI, Iris DISCOVER UI, and Conveyor UI
- **CL2:** Smart City platform by TID could be used as the CL2 observability portal
- **CL3A:** AgroApps 360 Portal by AGA could be used as the CL3A observability portal
- **CL3E:** BioGas Pro or EV Chargers App
- **CL4:** RMLIoT UI, TheTorre UI, GAMAYA UI, and AIDGE UI

3.1.2 Technologies

The technology stack presented in Figure 3-2 represents a preliminary and illustrative implementation approach for the Business Management Portal (BMP) architecture. At this stage, the technologies shown should be considered as candidate solutions and recommendations rather than finalized implementation decisions. The proposed stack combines modern web technologies, AI frameworks, orchestration components, and observability platforms that align with the architectural requirements of COP-PILOT.

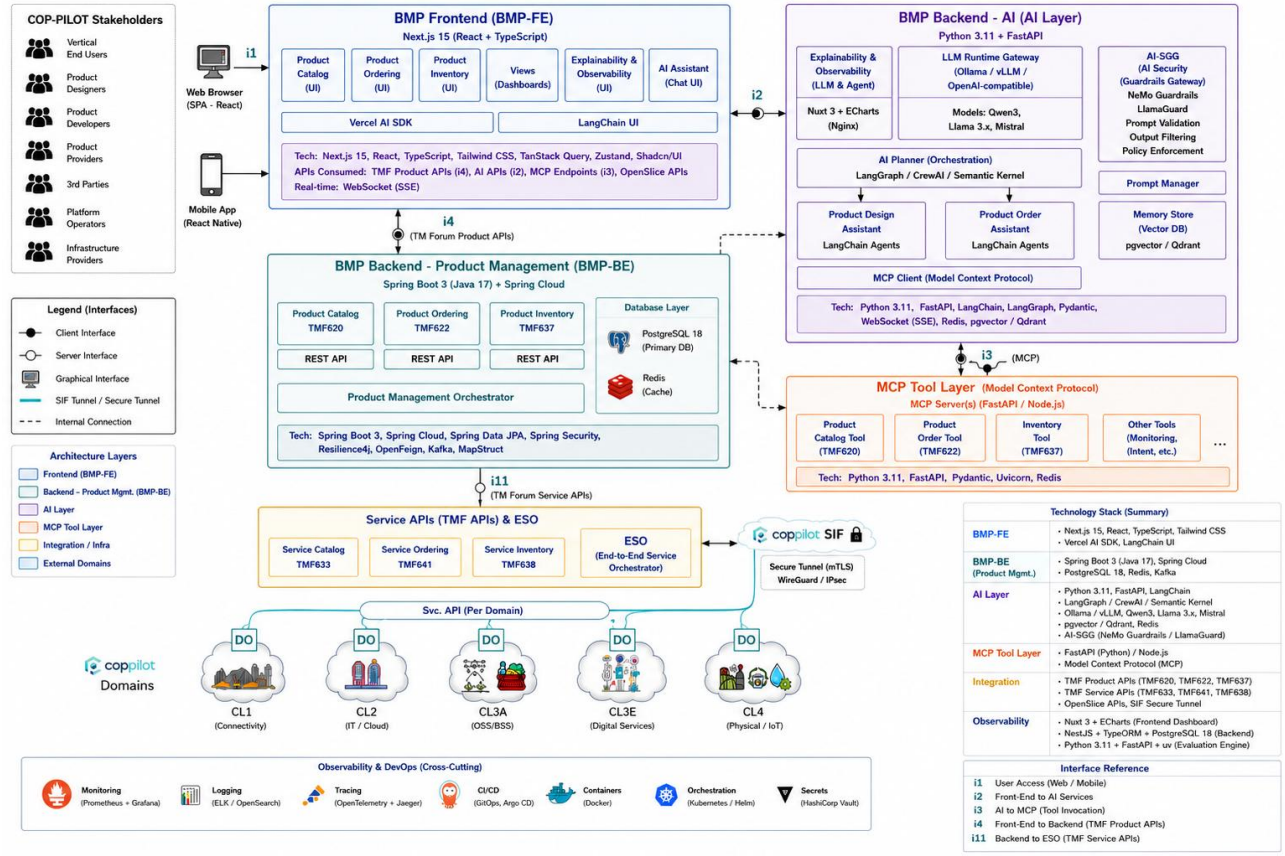

Figure 3-2: Technical design of the COP-PILOT Business Management Portal.

The technology stack presented in Figure 3-2 combines technologies that are currently implemented within the project, technologies planned for implementation in the scope of Deliverable D3.2, and candidate technologies that are being considered as part of the target architecture. The figure therefore serves both as a representation of the current solution landscape and as a reference architecture illustrating potential implementation options for future platform evolution.

For the BMP Frontend and Backend, technologies such as Next.js 15, React, TypeScript, Vercel AI SDK, LangChain, and TM Forum-compliant APIs are identified as candidate or planned technologies to support user interaction, business process management, and AI-assisted workflows. Within the AI Layer, OpenAI-compatible runtimes such as Ollama or vLLM, together with foundation models including Qwen3, Llama 3.x, and Mistral, are considered as potential implementation options. Similarly, orchestration frameworks such as LangGraph, CrewAI, and Semantic Kernel are being evaluated for supporting agent-based workflow execution and coordination.

Knowledge retrieval and contextual memory capabilities may be realised through vector database technologies such as pgvector or Qdrant, while AI governance and security functions may leverage solutions including Open Policy Agent (OPA), PromptGuard, NeMo Guardrails, or LlamaGuard. MCP-based integration is currently envisaged as the primary mechanism for exposing platform capabilities to AI agents and enabling standardised interaction between AI services and operational systems.

The observability and explainability capabilities depicted in the figure correspond to the architecture currently planned for D3.2. This includes a dedicated dashboard solution based on Nuxt 3 and ECharts for visualisation, NestJS and PostgreSQL for backend services and data management, and a Python/FastAPI-based evaluation engine for AI quality assessment, monitoring, and explainability functions.

Consequently, the technologies shown in the figure should not be interpreted as a definitive implementation stack. Rather, they represent a combination of existing project components, technologies planned within upcoming project activities, and reference technologies under evaluation. Final technology selections may evolve as implementation progresses and as functional requirements, performance considerations, deployment constraints, and operational needs are further refined.

3.1.3 Interfaces

The BMP architecture is interconnected through a set of standardized interfaces. The BMP Frontend communicates with the Product Management layer through TM Forum Product APIs (i4) and with the AI layer through conversational and service APIs (i2). AI assistants access platform capabilities through the Model Context Protocol (i3), while product fulfilment is delegated to the End-to-End Service Orchestrator through TM Forum Service APIs (i11). These interfaces provide a standards-based integration framework that ensures interoperability between business management, AI-driven automation, and service orchestration domains.

Table 3-1: Business Management Portal interfaces.

Interface	Source	Target	Technology/Standard	Description
i1	COP-PILOT Stakeholders	BMP Frontend	Web UI / HTTPS	User interaction interface through which business stakeholders access the Business Management Portal. It provides graphical dashboards, catalogue browsing, ordering functions, inventory management, AI-assisted interactions, and operational visibility.
i2	BMP Frontend	BMP Backend AI	REST APIs / WebSocket / AI Gateway	Interface between the portal user experience and the AI layer. User requests, prompts, and conversational interactions are forwarded to the AI services, while AI-generated responses, recommendations, and explanations are returned to the portal. This interface is protected by AI-SGG controls.
i3	BMP Backend AI	BMP Backend MCP	Model Context Protocol (MCP)	Standardized protocol interface used by AI assistants and planners to access platform capabilities exposed as tools. Through MCP, AI agents can invoke catalogue, ordering, inventory, orchestration, and other platform functions in a controlled and interoperable manner. Tool invocation should be governed

				through policy validation, allow-listing of authorised actions, and execution safeguards before triggering operational workflows.
i4	BMP Frontend	BMP Backend Product Management	TM Forum Product APIs (TMF620, TMF622, TMF637)	Business-facing product management interface used by the portal to access product catalogue information, create and manage product orders, and retrieve product inventory data. It exposes product-related capabilities according to TM Forum Open API standards.
i11	BMP Backend Product Management	ESO (End-to-End Service Orchestrator)	TM Forum Service APIs (TMF641, TMF638, TMF633, TMF640)	Service fulfilment interface used by the Product Management layer to interact with the orchestration domain. Product orders are decomposed into service requests and executed through standardized TM Forum service management APIs, enabling provisioning, activation, inventory management, and lifecycle control of services.

3.2 END-TO-END SERVICE ORCHESTRATOR

This section presents an initial version of the internal design of the COP-PILOT End-to-End Service Orchestrator (ESO). This abstract architecture block is linked to an actual open platform that brings features to address the ESO-L requirements defined in D2.1 Annex 3.

3.2.1 Design

Today's infrastructures expand towards the end users, where numerous Internet of Things (IoT) and/or user equipment (UE) devices require connectivity with local (extreme edge) or remote (far in the cloud) services. This connectivity may be often provided via cellular (5G and beyond) networks or low-power IoT networks integrated with multiple geo-distributed (edge and core) cloud infrastructures. Such complex ecosystems pose several challenges in the way service onboarding, deployment, and lifecycle management (LCM) is performed in an end-to-end fashion, mainly because modern services:

- are structured as collections of independently deployable and loosely coupled microservices.
- may span across multiple administrative domains - managed by different owners - who often do not trust each other.
- are often associated with service level agreements (SLAs), which pose strict compute (i.e., CPU, main memory, storage) and network (i.e., latency, throughput, packet loss) requirements.

Stakeholders: The COP-PILOT ESO is a holistic end-to-end service orchestration platform that aims to bridge these gaps by offering zero-trust multi-domain service orchestration abstractions to various stakeholders. Table 3-2 summarizes the various stakeholders that benefit from the COP-PILOT ESO.

Table 3-2: COP-PILOT ESO ecosystem and relevance with stakeholders.

Stakeholder	Why ETSI HypO is useful to a stakeholder
Infrastructure providers	manages services across multiple geo-distributed "non-trusted" domains acting as root of trust.
Service providers	• allows service providers to package distributed services as if they are centralized, thus moving complexity to the platform. • offers a single set of service management APIs, no matter how many domains your application expands to. • supports state-of-the-art service packaging tools (i.e., Kubernetes, helm, and docker-compose). • provides programmable connectivity services across clusters/domains (via the COP-PILOT SIF), giving a single-cluster illusion to the users. • provides a real-time view of the deployed service instances' state. provides knobs to change a service instance's runtime state via a service update API and/or real-time policies.
Service developers	• allows developers to declare their desired public/private services registries • offers the possibility to persist sensitive credentials towards private service registries into a secure secret management database, ensuring isolation between multiple tenants • dynamically authenticates against any service registry (e.g., Harbor [47], GitLab [48], GitHub [49], jFrog [50], etc.) to pull images during service deployment
Platform Operators	• decouples service and resource management via integration with operations support systems (OSS) – i.e., the COP-PILOT DO – using open standardized APIs. • deals with service management. • delegates resource management to OSS (i.e., the DO). • manages multiple OSS instances. • integrates with vanilla container orchestration platforms (i.e., Kubernetes).
Vertical end users and 3rd parties (e.g., Open Call users)	allows to browse the COP-PILOT ecosystem across Europe, explore the entire marketplace, and order services across Clusters of interest (role-based permissions are granted to access certain services)

Design: The COP-PILOT ESO is based on a service-based architecture of multiple loosely coupled microservices, as shown in Figure 3-3.

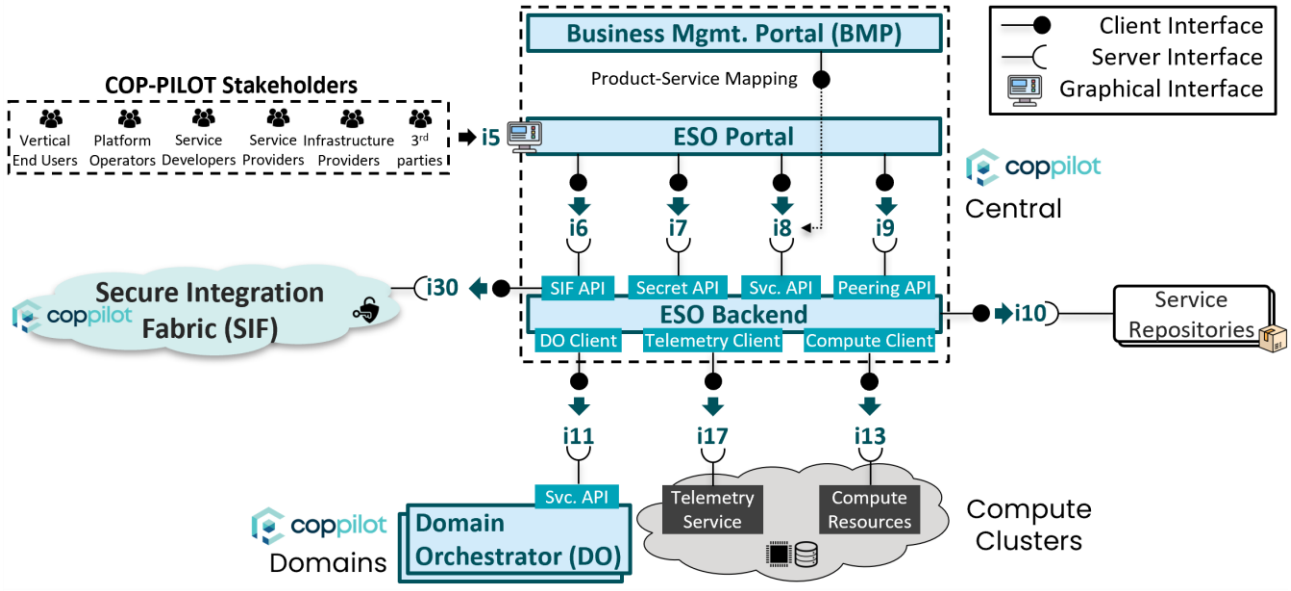

Figure 3-3: Abstract design of the COP-PILOT End-to-End Service Orchestrator.

3.2.2 Technologies

The COP-PILOT ESO is instantiated by the ETSI Hyper Orchestrator (HypO); a real open-source service orchestration platform under the ETSI OpenSlice Software Development Group (SDG). The architecture of ETSI HypO is depicted in Figure 3-4.

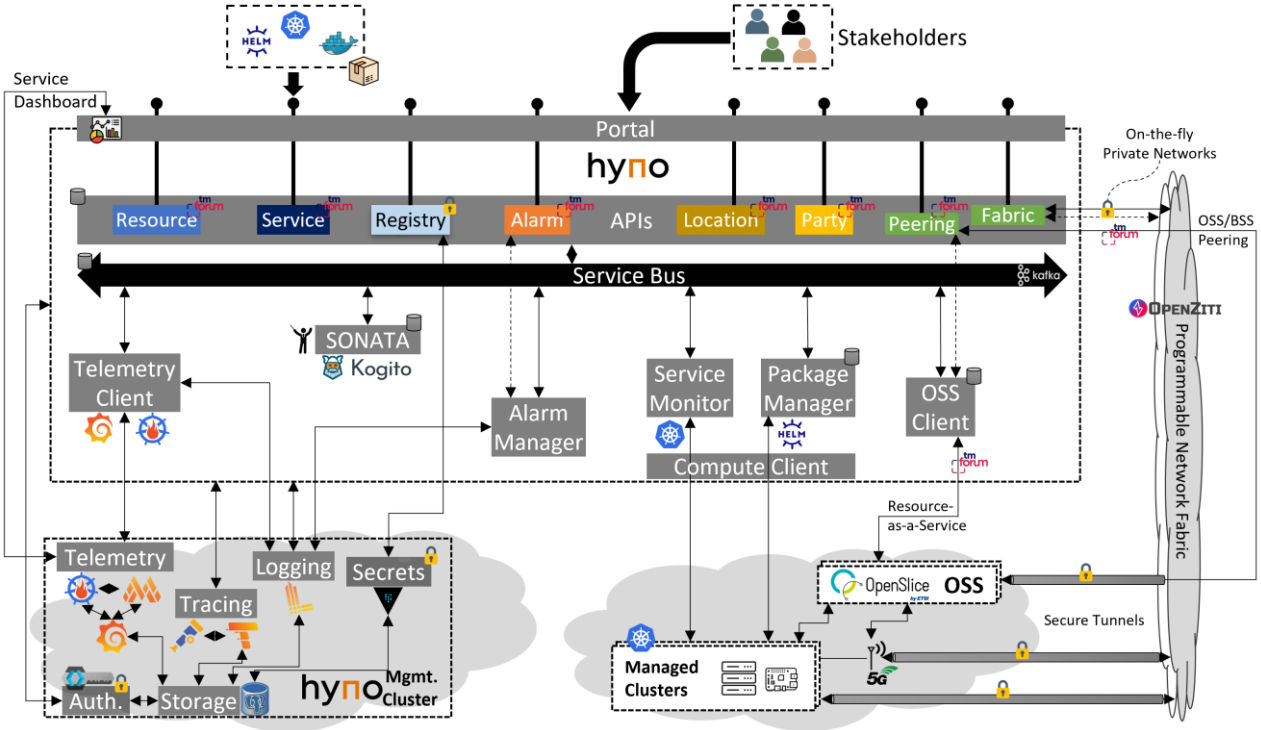

Figure 3-4: Technical design of the COP-PILOT End-to-End Service Orchestrator based on ETSI HypO.

In the rest of this sub-section, we classify HypO's services into two categories, i.e., core services summarized in Section 3.2.2.1 and auxiliary services listed in Section 3.2.2.2.

3.2.2.1 Core ETSI HypO Services

In its core, ETSI HypO puts together a set of services that co-operatively materialize and expose end-to-end service orchestration capabilities. These services are visualized in Figure 3-4, while their role is summarized in Table 3-3.

Table 3-3: Core ETSI HypO services.

Core ETSI HypO Service Name	Objective
Portal	offers a modern user interface to facilitate complex interactions with the underlying infrastructure as well as the overlay end-to-end services and their management.
TMF API	Exposes a set of standardized REST-based TMForum APIs for service, resource, location, party, and alarm management.
Peering API	Exposes a REST API for initiating a peering operation with a remote TMF-compliant system (e.g., an OSS or BSS) leveraging TMF 632 Party Management API.
Fabric API	Exposes a REST API for managing (create, read, update, delete) key COP-PILOT SIF entities, such as identities, encrypted connections, policies, etc.
Registry API	Exposes a REST API for managing secrets on top of a secret database (Vault).
SONATA	Leverages Business Process Model and Notation (BPMN) workflow diagrams to encode the lifecycle of a service (design, onboarding, ordering, runtime management) based on TMF standards.
Package Manager	Manages the full lifecycle of packaged service deployments using cloud-native best practises.
Service Monitor	Assesses the health and stability of deployed services running in a compute cluster.

OSS Client	Integrates with ETSI OpenSlice; a standardized Operations Support System (OSS) that provides compute and network resources as-a-service.
Telemetry Client	Gathers telemetry data across all managed clusters, where ETSI HypO deploys user services.
Compute Client	Binds to the compute API of a given cluster, issuing compute management commands to manage compute resources.
Service Bus	Allows asynchronous, event-driven communication across all ETSI HypO microservices. This service is based on Apache Kafka [26].

3.2.2.2 Auxiliary ETSI HypO Services

ETSI HypO is a cloud-native service that lives in a management Kubernetes cluster. In this cluster, popular open-source services are installed alongside ETSI HypO offering important features to the orchestrator. These services are visualized in and summarized in Table 3-4.

Table 3-4: Auxiliary ETSI HypO open-source services.

Auxiliary ETSI HypO Service Name	Objective
PostgreSQL [27]	A cloud-native relational database management system used to persist data across all ETSI HypO microservices (e.g., the TMF schema, Package Manager, OSS Client, etc.)
Keycloak [28]	Authenticates users against ETSI HypO, before allowing these users to access the ETSI HypO Northbound APIs. Corresponds to the Identity Manager service in Figure 3-3.
Hashicorp Vault [29]	Manages ETSI HypO stakeholders' secrets and protects their sensitive data.
Prometheus Operator [30] [31]	A cloud-native version of the Prometheus [30] monitoring and alerting system with a timeseries database backend.
Grafana [32]	An external dashboard that integrates with (i) Loki for visualizing service logs, (ii) Tempo for visualizing service traces, and (iii) Mimir for visualizing SLAs.
Grafana Mimir [33]	Enhances the Prometheus operator with long-term persistence and high-availability features.
Grafana Tempo [34]	A central component that integrates with popular open-source tracing protocols, such as OpenTelemetry [36], to collect and persist traces from every ETSI HypO microservice.
Grafana Loki [35]	A central component that aggregates, stores, and queries log entries from every ETSI HypO microservice. All logs are dispatched and converted to standardized TMF Alarm format.
OpenTelemetry [36]	A protocol used to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) for analysing a software's performance and behaviour.

3.2.3 Interfaces

Table 3-5 lists the interfaces exposed by the COP-PILOT ESO as well as their purpose. Most of these interfaces are standardized, based on TMForum.

Table 3-5: COP-PILOT ESO interfaces.

Endpoint	Version	Description
/tmf-api/serviceCatalogManagement/v4/	4.0.0	Provides catalogues, categories, candidates, and specifications of Services
/tmf-api/serviceOrderingManagement/v4/	4.1.1	Provides the ability to manage service orders that comprise of one or more service specifications
/tmf-api/serviceActivationManagement/v4/	4.0.1	Provides the ability to activate/deactivate and configure services in the service inventory
/tmf-api/serviceInventory/v4/	4.0.1	Provides the ability to query and manipulate active service instances
/tmf-api/resourceCatalogManagement/v4/	4.1.0	Provides catalogues, categories, candidates, and specifications of Resources
/tmf-api/resourceOrderingManagement/v4/	4.0.0	Provides the ability to manage resource orders that comprise of one or more resource specifications
/tmf-api/resourceInventoryManagement/v4/	4.0.0	Provides the ability to query and manipulate active resource instances
/tmf-api/party/individual/v5/	5.0.0	Manage individual parties

/tmf-api/party/organization/v5/	5.0.0	Manage corporate parties (i.e., organizations)
/tmf-api/party/role/v5/	5.0.0	Manage party roles
/tmf-api/geographicSiteManagement/v5/	5.0.0	Manage resource locations at abstract sites, where each site may contain a list of geographic addresses
/tmf-api/alarmManagement/v5/	5.0.0	Manage alarms of different severity levels withing the ETSI HypO ecosystem
/peering-api/peering	latest	Create, read, update, and delete peered organizations via TMF Party Mgmt. API.
/network/controller	latest	Create, read, update, and delete network controllers
/fabric-api/{netControllerId}/domain/identities	latest	Create, read, update, and delete identities for a given network controller
/fabric-api/{netControllerId}/domain/connections	latest	Create, read, update, and delete connections for a given network controller
/registry-api/image-pull	latest	Manage Docker image registry credentials
/registry-api/package-registry	latest	Manage Helm / artifact registry credentials
/registry-api/kube-conf	latest	List stored kubeconfigs and issue terminal access tokens
/registry-api/group	latest	Create, read, update, and delete Vault groups
/registry-api/policy	latest	Create, read, update, and delete Vault policies
/registry-api/policy-group-flow	latest	Atomically provision a group with its default policies
/registry-api/fabric	latest	Retrieve Fabric network controller credentials
/registry-api/admin/logs	latest	Dynamically change runtime log level

Figure 3-3 depicts the entire set of interfaces supported by the COP-PILOT ESO. As shown by the legend in Figure 3-3, the ESO interfaces are classified in 3 categories: (i) server-side interfaces, (ii) client-side interfaces, and (iii) user interfaces. The server-side interfaces are those exposed by the ESO towards other systems. Table 3-5 describes these interfaces in more detail, while a mapping between Table 3-5 and Figure 3-3 interfaces is presented below:

- All /fabric-api/ interfaces in Table 3-5 correspond to i6 interface in Figure 3-3.
- All /registry-api/ interfaces in Table 3-5 correspond to i7 interface in Figure 3-3.
- All /tmf-api/ interfaces in Table 3-5 correspond to i8 interface in Figure 3-3.
- All /peering-api/ interfaces in Table 3-5 correspond to i9 interface in Figure 3-3.

The rest of the interfaces (not listed above) correspond to client-side or user interfaces.

3.3 DOMAIN ORCHESTRATOR

This section presents an initial version of the internal design of the COP-PILOT DO, which realises the domain-level orchestration capabilities of the DDO-L, as defined in D2.1 Annex 3, providing a uniform way to model, expose, and manage compute and network resources within each COP-PILOT cluster domain. The following abstract architecture block is linked to an actual open platform that brings features to address the DDO-L requirements.

3.3.1 Design

The DO is deployed as a per-domain orchestration and OSS stack, providing a single authoritative view of the resources, resource offerings, and resource-level service instances available within a given COP-PILOT cluster. Conceptually, it sits between the ESO-L and the DIS-L, as depicted in Figure 3-5. It consumes service orders from the ESO and translates them into concrete lifecycle

actions on the underlying domain infrastructure. To support this role, the DO organises its functionality into three main planes:

- a northbound resource-as-a-service exposure plane
- an internal catalog and inventory management plane
- a southbound integration plane towards cluster-level controllers and orchestrators

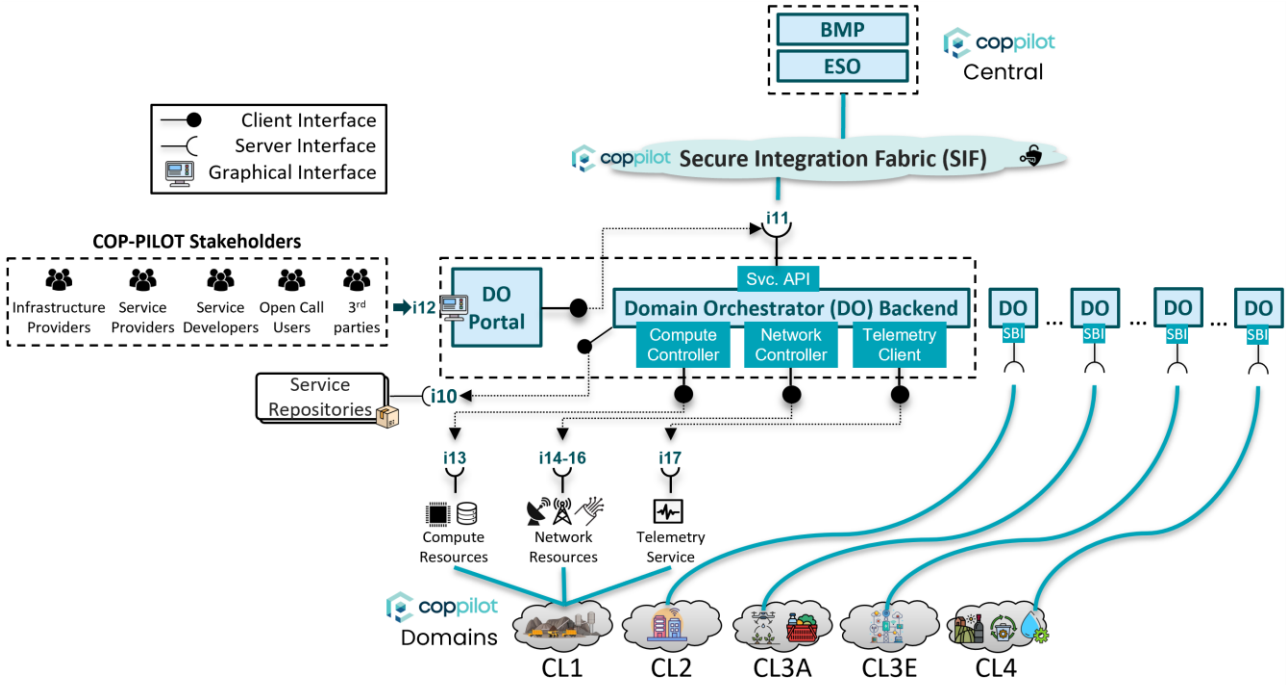

Figure 3-5: Abstract design of the COP-PILOT Domain Orchestrator.

On the northbound side, the DO exposes TM Forum-compliant catalogues, inventory, ordering, and monitoring APIs for resource-level service instances and domain-specific connectivity/computes/network resources. These APIs are used by HypO to extract domain service specifications, place resource orders, and monitor the lifecycle of resource-facing service instances associated with end-to-end services. Internally, OpenSlice maintains the domain's resource and service catalogues, the resource and service inventory, and the mapping between logical resource specifications and their realisations on local infrastructure, using a microservice architecture with an API gateway and an event bus. On the southbound side, the DO integrates with Kubernetes and other cluster controllers, as well as with ColonyOS, distributing workloads and placements across heterogeneous compute nodes in the domain.

In the COP-PILOT architecture, each piloting cluster instantiates its own DO instance following a common reference design, ensuring that all domains expose resources through a consistent set of TMF APIs and domain abstractions. The ESO peers with these DO instances using standard TMF party and site management APIs, allowing it to treat each domain as a federated provider from which it can request resource capabilities for composite services. This design decouples end-to-end service design and orchestration (handled centrally by HypO) from domain-specific service and resource design and orchestration (handled locally by OpenSlice), while still allowing the propagation of actions down to the infrastructure through the DO.

Ultimately, The COP-PILOT DO acts as a domain orchestration platform that bridges holistic end-to-end service orchestration with the complete lifecycle management and abstraction of the underlying domain infrastructure resources, essentially offering resource-as-a-Service. Table 3-6 summarizes the various stakeholders that benefit from the COP-PILOT DO.

Table 3-6: COP-PILOT DO ecosystem and relevance with stakeholders.

Stakeholder group	Role with respect to the DO	Typical DO interactions (via DO Portal / TMF APIs)
Infrastructure providers	Operate and expose cluster-level compute, storage, and network resources that are on-boarded into the DO domain.	• Publish and maintain resource specifications and sites • Validate inventory • Manage lifecycle policies
Service providers	Offer domain-specific services (e.g., vertical applications, NaaS/CaaS bundles) built on top of DO-managed resources.	• Define resource-backed service specifications • Request capacity • Track fulfilment and monitoring data for their services
Service developers	Design, integrate, and test new resource and service blueprints that will later be offered through the DO and ESO.	• On-board and refine service and resource specifications • Run test orders • Inspect telemetry and logs via the DO Portal
Open Call users	External innovators using COP-PILOT domains under Open Calls to experiment with new services and deployments.	• Browse available domain resources • Place controlled trial orders • Monitor usage within their projects
Third parties	External systems and tools (e.g., CI/CD pipelines, monitoring stacks, data platforms) integrated with the DO	• Invoke DO TMF APIs programmatically • Synchronise catalogues • Consume telemetry feeds

3.3.2 Technologies

The DO is instantiated using ETSI SDG OpenSlice as an open-source, intelligent OSS, based on TM Forum's Open Digital Architecture (ODA), extended and configured to expose COP-PILOT domain-specific Network-as-a-Service and Compute-as-a-Service offerings towards the ESO. The DO acts as a bridge between the ESO and local Cloud, 5G, and transport resources. The internal architecture of OpenSlice is captured in Figure 3-6.

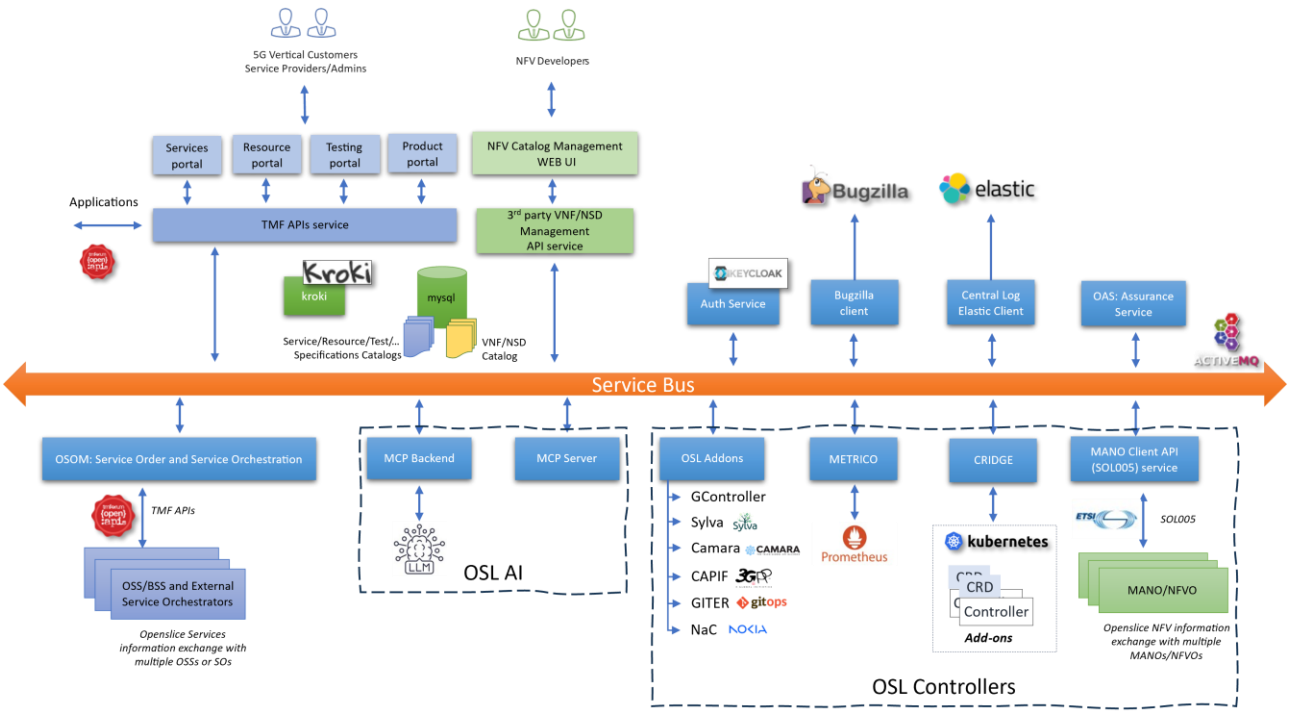

Figure 3-6: Technical design of the COP-PILOT Domain Orchestrator based on ETSI OpenSlice.

In the rest of this sub-section, we classify OpenSlice services into two categories, i.e., core services summarized in Section 3.3.2.1 and auxiliary services listed in Section 3.3.2.2, respectively.

3.3.2.1 Core ETSI OpenSlice Services

To materialize the DO functionality of the previously introduced Section 3.3.1, i.e., the complete lifecycle management and abstraction of the underlying domain infrastructure resources, ETSI OpenSlice composes the services presented in the following Table 3-7.

Table 3-7: Core ETSI OpenSlice services.

Core ETSI OpenSlice Service	Objective
Portal	Provides role-specific web portals (Services, NFV, Products, Testing, Resources) on top of OpenSlice APIs, enabling users to visually design, onboard, order, and monitor products, services, tests, and resources in a domain.
TMF API	Exposes REST-based TM Forum Open APIs that implement catalog, ordering, inventory, party, location, and alarm management functions, providing a standardised interface surface for domain-level OSS.
NFV API	Provides SOL-005-aligned [67] APIs for VNF/NS package onboarding, cataloguing, and lifecycle management towards external NFV MANO stacks, enabling DO-driven instantiation of virtualised network services.
OSOM	Implements OpenSlice Service Orchestration and Order Management, using the Flowable BPMN engine [57], combined with Google Blockly [59] to inject use-case dynamicity, to execute service and resource workflows that fulfil TMF service orders across infrastructure domains.
OSL AI	Embeds AI-driven capabilities into OpenSlice to analyse operational data and TMF-based models, enabling LLM/agent-assisted interactions and supporting closed-loop, autonomous optimisation of service and resource lifecycles.
Addons	Introduces an archive of reusable and replicable projects that leverage OpenSlice architecture to enable telco cloud scenarios and expose capabilities of a modern Operator Platform.
METRICO	Initiates customizable metrics-retrieval jobs and associates the data with services under scope.
CRIDGE	Create and Manage Custom Resources (CRs) based on Custom Resource Definitions (CRDs) installed on a Kubernetes cluster.

MANO Client	Acts as the mediation component between OpenSlice and external NFV MANO platforms, translating TMF and NFV API operations into concrete actions on target MANO providers.
Issue Management	Allows issue reporting and tracking via tickets to all relevant stakeholders.
Message Bus	Allows OpenSlice and 3rd party services to asynchronously exchange messages via queues and topics. This service is based on Apache ActiveMQ [68][26].

3.3.2.2 Auxiliary ETSI OpenSlice Services

Apart from the core services, ETSI OpenSlice comprises auxiliary services and open-source software for generic capabilities, as summarized in Table 3-8.

Table 3-8: Auxiliary ETSI OpenSlice open-source services.

Auxiliary ETSI HypO Service Name	Objective
MySQL [69]	A relational database management system used to persist data across all ETSI OpenSlice microservices (e.g., the TMF schema, etc.)
Keycloak [28]	An Authentication Server implementing OAuth2 authentication scheme that authenticates users against ETSI OpenSlice before allowing these users to access the Northbound APIs and UI.
Bugzilla [57]	A microservice capable of interfacing with an issue management system that raises issues to all related stakeholders, when actions such as new Service Orders are requested.
Prometheus Operator [30] [31]	A cloud-native version of the Prometheus [30] monitoring and alerting system with a timeseries database backend.
Grafana [32]	An external dashboard that integrates with Prometheus to offer usage metrics.
Elastic Stack [70]	A Central Logging microservice that logs all distributed actions into an Elasticsearch cluster, providing enhanced long-term persistence and high-availability features for OpenSlice operations.
Kroki [55]	A visualization server which enables an intuitive illustration of dependency graphs and interactions between services and components.

3.3.3 Interfaces

Table 3-9 lists the Northbound interfaces exposed by the COP-PILOT DO as well as their purpose.

Table 3-9: COP-PILOT DO interfaces.

Endpoint	Version	Description
/tmf-api/serviceCatalogManagement/v4/	4.0.0	Provides catalogues, categories, candidates, and specifications of Services
/tmf-api/resourceCatalogManagement/v4/	4.0.0	Provides catalogues, categories, candidates, and specifications of Resources
/tmf-api/serviceInventory/v4/	4.0.0	Provides a standardized mechanism to query and manipulate the Service inventory
/tmf-api/resourceInventoryManagement/v4/	4.0.0	Provides a standardized mechanism for managing Resources
/tmf-api/serviceOrdering/v4/	4.0.0	Provides a standardized mechanism for managing a Service Order
/tmf-api/partyRoleManagement/v4/	4.0.0	Provides a standardized mechanism to define Party Roles
/tmf-api/party/v4/organization	4.0.0	Provides a standardized mechanism to define Organizations
/tmf-api/serviceTestManagement/v4/	4.0.0	Provides a standardized mechanism for managing tests of provisioned Services
/tmf-api/geographicSiteManagement/v5/	5.0.0	Provides a standardized mechanism for managing sites that can be associated with entities
/tmf-api/performance/v5/	5.0.0	Provides a standardized mechanism for measurement collection
/tmf-api/lcmrulesmanagement/v1/	1.0.0	Custom API environment for LCM Rules

Figure 3-5 illustrates the overall interfaces supported by the DO. Table 3-9 summarizes the interfaces exposed by the DO towards external systems, including the ESO and third-party applications, that are actively utilized within the COP-PILOT use-cases. For completeness, all implemented standards, including those not utilized directly in COP-PILOT (e.g., business endpoints), are referenced in Section 5.2.

3.4 DATA MANAGEMENT

This section presents an initial version of the internal design of the COP-PILOT Data Management (DM) platform. This abstract architecture block is linked to an actual open platform that brings features to address data-related DDO-L requirements defined in D2.1 Annex 3.

3.4.1 Design

Stakeholders: The COP-PILOT DM is a generic data management platform that when obtained by a specific vertical sector stakeholder could be turned into a tailored Data Space solution for certain data types, data transactions, and data federation operations. Figure 3-7 shows an indicative list of stakeholders that could benefit from the COP-PILOT DM platform. These stakeholders include (i) IoT vendors and infrastructure providers who wish to onboard their (hardware or software-based) data sources on the DM, (ii) IoT platform operators that wish to integrate their existing IoT platforms with COP-PILOT's standardized DM, (iii) integrators that undertake this kind of integration activities in a business sector, (iv) IoT service developers and providers who wish to render their services capable of interacting with the dynamic storage, processing, and federation capabilities of the DM.

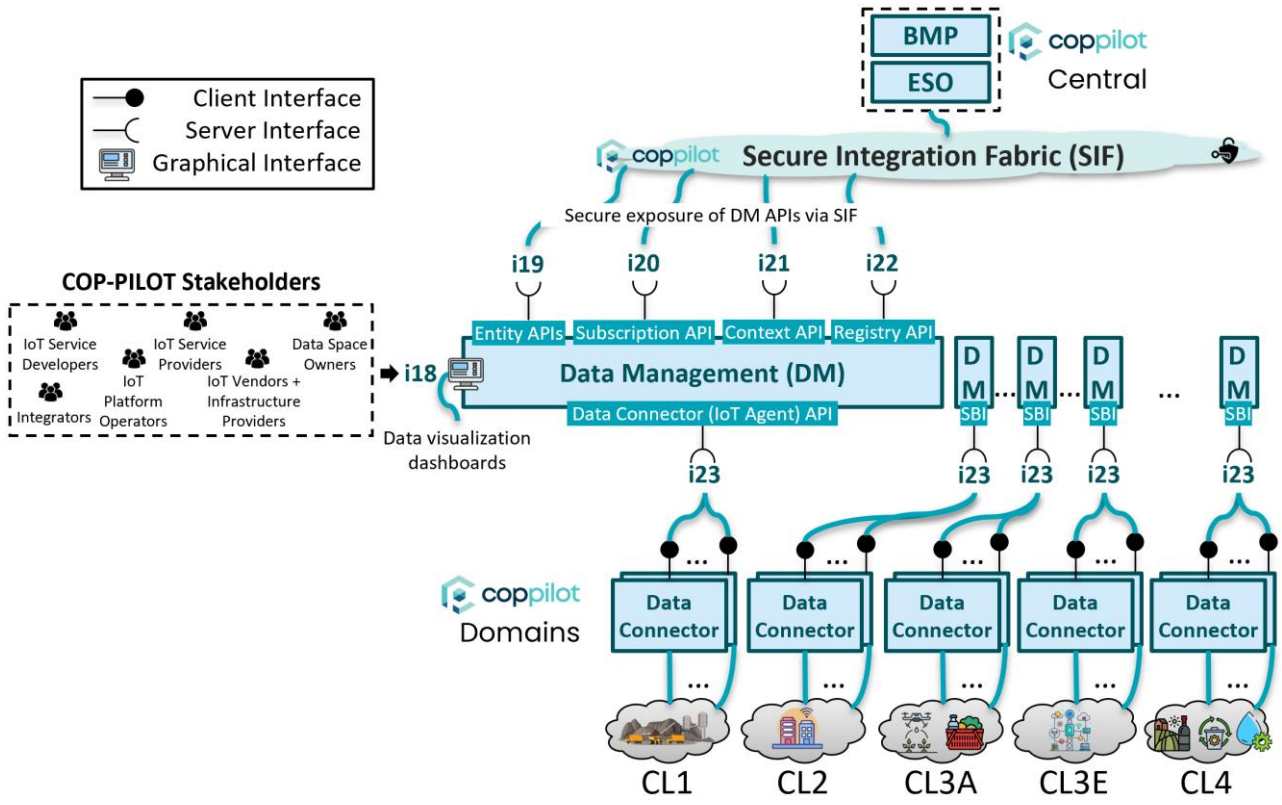

Figure 3-7: Abstract design of the COP-PILOT Data Management platform.

Design: The COP-PILOT DM is designed as shown in Figure 3-7. At the southbound side, the DM platform leverages distributed Data Connectors – employed by the COP-PILOT Clusters – to tame the vast cardinality and heterogeneity of IoT devices and data sources across multiple vertical sectors. This is also the reason that DM resides in the COP-PILOT Domain level, allowing domain owners to employ their own instance of the DM, thus managing their data. At the northbound side, the COP-PILOT DM exposes 4 sets of APIs to manage data in various ways, i.e., discovering and managing entities, managing context, subscribing to data sources, etc. These APIs may be securely exposed to relevant stakeholders (e.g., an IoT service provider), neighbouring intra-domain systems (e.g., the Domain Orchestrator), or even external consumers (e.g., a system in the central COP-PILOT domain) via the COP-PILOT SIF.

The next section provides more details about the COP-PILOT Data Management platform, both in terms of technologies and the details of the exposed interfaces.

3.4.2 Technologies

This section provides a high-level introduction to the internal design of the COP-PILOT Data Management platform, leveraging the Orion-LD Context Broker as an underlying technology stack. Orion-LD implements the ETSI ISG CIM standard specification API, also known as the ETSI Next Generation Service Interfaces - Linked Data (NGSI-LD) API [66]. For historical reasons, this component maintains backward compatibility with the previous version of the API (i.e., FIWARE NGSI v2). Figure 3-8 shows the main information flow and how program control passes from one module to another.

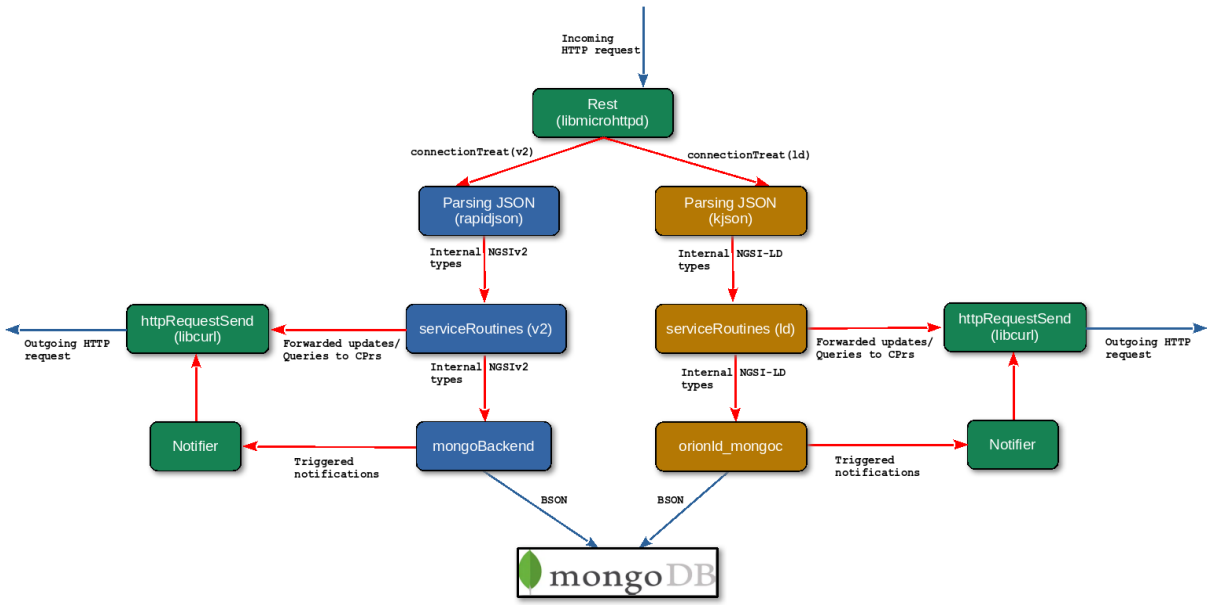

Figure 3-8: Technical design of the COP-PILOT DM based on Orion-LD.

Current Orion-LD internal design:

[1] At start-up, the Orion Context Broker starts an HTTP server listening for incoming requests. The rest library deals with these requests and the library is based on the **libmicrohttpd** external library, which spawns a new thread per request depending on the version of the API (NGSIv2 and NGSI-LD).

[2] The **connectionTreat** function is the entry point for new requests. Depending on the version of the NGSI API to which the request belongs (basically, depending on whether the request URL prefix /v2 or /ngsi-ld) the execution flow goes in one "branch" or another, of the execution logic as shown in Figure 3-8.

Flow for ETSI NGSIv2 requests

a. First, the **rapidjson** library takes the request payload as input and generates a set of objects. It supports both SAX and DOM style API.
b. Next, a request servicing function is invoked to process the request. Each request type (in terms of HTTP and URL pattern) has a separate function. We call these functions "service routines" and they reside in the library **serviceRoutinesV2** for NGSIv2 types.
c. At the end, the service routine calls the **mongoBackend** library to generate a Binary JSON to persist the data into the MongoDB database.

Flow for ETSI NGSI-LD requests

d. First, the **kjson** library takes the request payload as input and generates a tree structure of **KjNode** that is used in all Orion-LD with an increase of performance.
e. Next, like NGSI-LD, a service routine is called to process the request. Each request type (in terms of HTTP and URL pattern) has a service routine. These "NGSILD service routines" reside in the library **serviceRoutines LD**.
f. At the end, a new mongoDB library is implemented, **orionld_mongoc**, is implemented so it can be updated with the new versions of mongoDB and improve the performance of operations.

[3] Whenever a notification is triggered (e.g., because of updating an entity covered by an existing subscription), the **notifier** module (residing in the ngsiNotify library) is invoked from mongoBackend or orionld_mongoc to send such a notification.

[4] The **httpRequestSend** function oversees sending HTTP requests. It is based on the libcurl external library. These functions are called either by the notifier module (to send notifications) or by any serviceRoutines function capable of forwarding queries/updates to Context Providers under some conditions.

With respect to the Data Connectors that interact with the underlying heterogeneous IoT landscape, a design of this component can be seen in Figure 3-9. In this figure, the Data Connectors are seen as distributed IoT agents that provide a middleware between the Data Management platform (i.e., Orion-LD) and the various sensors/data sources located in the COP-PILOT Clusters.

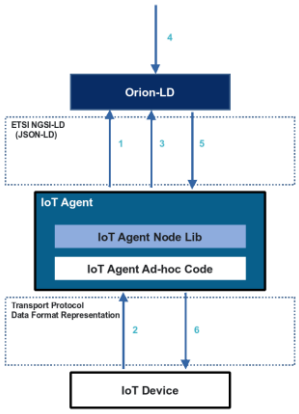

Figure 3-9: Design of IoT Agents interacting with Orion-LD via ETSI NGSI-LD/v2.

[5] The **IoT Agent Node Lib** is responsible of the implementation of the ETSI NGSI-LD API endpoints to communicate with any ETSI NGSI-LD broker. The operations include:
   a. Register a new IoT Agent in the ETSI NGSI-LD Broker to send IoT Device data (1).
   b. Send the IoT Device information to the ETSI NGSI-LD broker using the ETSI NGSI-LD API (3).
   c. Receive actuations from the ETSI NGSI-LD Broker to modify some state of the IoT Device (5).

[6] The **IoT Agent Ad-hoc Code** is responsible of the implementation of the corresponding parser for the IoT Device Transport Protocol and IoT Device data representation format. This is implemented using a callback operation in Node.js and links to the IoT Agent Node Lib. The operations include:
   a. The device sends the request in the corresponding transport protocol and representation format (2).
   b. The IoT Agents send the command to the IoT Device (6).

3.4.2.1 Data Management Platform Dashboards

Context broker data Observability: The Context Broker Data Observability dashboard is a web-based monitoring interface that provides real-time monitoring of context data quality, latency, and schema compliance that flow through an NGSI-LD Context Broker. It enables data operators/platform administrators to track attribute distributions, detect data drift, monitor ingestion latency, and verify that entity attributes conform to expected behaviour; supporting trust in the underlying IoT data layer across COP-PILOT piloting clusters. The component is implemented as a three-service monorepo, consisting of the frontend dashboard (Nuxt 4/Vue 3), a NestJS/TypeORM backend that ingests NGSI-LD entity update notifications and stores context history in PostgreSQL, and a Python FastAPI data analytics engine that computes observability metrics including distributions, drift, and latency.

Cluster Operational Intelligence Dashboard: The Cluster Operational Intelligence Dashboard is a centralised, user-centric web interface designed to translate real-time NGSI-LD data from the COP-PILOT Orion Context Broker into actionable, domain-specific insights. While infrastructure tools monitor data health, this solution focuses on high-level operational visualisation, allowing cluster operators to seamlessly track core KPIs, dynamic sensor states, and critical threshold alerts. To

minimise local administrative overhead, the dashboard operates via a central cloud deployment, securely ingesting remote cluster data through OpenZiti network tunnels. Built upon a scalable modern web stack, it provides dynamic visualisations that empower cluster decision-making without requiring complex, localised technical setup or maintenance.

3.4.3 Interfaces

The complete list of ETSI NGSI-LD API features implemented by Orion-LD is described in Table 3-10, Table 3-11, Table 3-12, Table 3-13, Table 3-14, and Table 3-15.

Table 3-10: COP-PILOT Data Management interfaces – Entities.

Endpoint	Version	Description
POST /entities	V1.12.0	Create an entity, i.e., real-world or virtual objects (e.g., a sensor, a weather station, a car, etc.)
POST /entities/{entityId}/attrs/		Append attributes to an entity
GET /entities		Query an entity by several characteristics (e.g., id, type, idPattern, attrs, georel, geometry, coordinates, etc.)
GET /entities/{entityId}		Query a specific entity by ID
GET /entities/{entityId}/attrs		Query a specific entity's attributes
GET /entities/{entityId} with options		Query a specific entity by ID with options
PUT /entities/{entityId}		Replace entity
PUT /entities/{entityId}/attrs/{attrId}		Replace entity attribute
PATCH /entities/{entityId}		Merge entity
PATCH /entities/{entityId}/attrs/{attrId}		Update entity attribute
DELETE /entities/{entityId}		Delete entity
DELETE /entities/{entityId}/attrs/{attrId}		Partially delete entity attribute

Table 3-11: COP-PILOT Data Management interfaces – Entities operations.

Endpoint	Version	Description
POST /entityOperations/create	V1.12.0	Batch entity creation (more than one instance of the same entity)
POST /entityOperations/upsert		Batch entity creation/update
POST /entityOperations/update		Batch entity update
POST /entityOperations/delete		Batch entity delete
POST /entityOperations/query		Batch entity query

Table 3-12: COP-PILOT Data Management interfaces – Entity discovery.

Endpoint	Version	Description
GET /types	V1.12.0	Query available entity types
GET /types/{type}		Query details of available entity types
GET /attributes		Query available attributes
GET /attributes/{attrId}		Query details of available attributes

Table 3-13: COP-PILOT Data Management interfaces – Subscriptions.

Endpoint	Version	Description
POST /subscriptions	V1.12.0	Create a subscription
GET /subscriptions		Query subscriptions
GET /subscriptions/{subscriptionId}		Retrieve specific subscription
PATCH /subscriptions/{subscriptionId}		Update subscription
DELETE /subscriptions/{subscriptionId}		Delete subscription

Table 3-14: COP-PILOT Data Management interfaces – Registry.

Endpoint	Version	Description
POST /csourceRegistrations	V1.12.0	Create a context source registration
GET /csourceRegistrations		Query context source registrations
GET /csourceRegistrations/{registrationId}		Retrieve context source registration
PATCH /csourceRegistrations/{registrationId}		Update context source registration
DELETE /csourceRegistrations/{registrationId}		Delete context source registration

Table 3-15: COP-PILOT Data Management interfaces – JSON-LD context.

Endpoint	Version	Description
POST /jsonldContexts	V1.12.0	Add context
GET /jsonldcontexts		List contexts
GET /jsonldcontexts/{contextId}		Retrieve a specific context
DELETE /jsonldcontexts/{contextId}		Delete context

At the level of the various IoT and data connectors, Table 3-16 describes the endpoints implemented by the IoTAgent Node library. This is the part of the Context Broker relevant to COP-PILOT Clusters as each Cluster may possess its own IoT agents for their own types of devices, therefore WP4 may find this API useful for onboarding IoTs and other data sources onto the COP-PILOT Data Management platform.

Table 3-16: COP-PILOT Data Management interfaces – IoT agent configuration.

Endpoint	Version	Description
GET /services	V1.12.0	Return the service description
POST /services		Create a service
PUT /services		Update a service
DELETE /services		Remove a service
GET /devices		Retrieve all devices
POST /devices		Create a device
GET /devices/{device_id}		Retrieve a specific device
PUT /devices/{device_id}		Update a device
DELETE /devices/{device_id}		Remove a device

Figure 3-7 depicts the entire set of interfaces supported by the COP-PILOT DM. As shown by the legend in Figure 3-7, the DM interfaces are classified in 2 categories: (i) server-side interfaces and (ii) client-side interfaces. The server-side interfaces are those exposed by the DM towards other systems. Table 3-10, Table 3-11, Table 3-12, Table 3-13, Table 3-14, and Table 3-15 describe these interfaces in more detail, while a mapping between these tables and Figure 3-7 interfaces is presented below:

- All /entities/ interfaces in Table 3-10, /entityOperations/ in Table 3-11, and /types/ in Table 3-12 correspond to i19 interface in Figure 3-7.
- All /subscriptions/ interfaces in Table 3-13 correspond to i20 interface in Figure 3-7.
- All /csourceRegistrations/ interfaces in Table 3-14 correspond to i22 interface in Figure 3-7.
- All /jsonldContexts/ interfaces in Table 3-15 correspond to i21 interface in Figure 3-7.
- The IoT Agent API in Table 3-16 corresponds to i23 interface in Figure 3-7.

3.5 SECURE INTEGRATION FABRIC

This section presents the initial version of the COP-PILOT Secure Integration Fabric (SIF). The SIF provides the secure communication layer of the COP-PILOT platform, enabling controlled, identity-based and policy-driven connectivity between central platform components, distributed domain components, cluster services, and authorised users or systems.

The SIF is instantiated using **NetFoundry**/**OpenZiti**, an open-source programmable zero-trust networking platform, which provides identity-first reachability. Within COP-PILOT, SIF provides a secure overlay fabric that allows services to communicate across heterogeneous administrative domains without requiring direct exposure of the underlying infrastructure. It supports the secure interconnection of central platform components such as the Business Management Portal and End-to-End Service Orchestrator with Domain Orchestrators, Data Management services, compute clusters, service repositories, and vertical applications deployed across the COP-PILOT clusters.

The key objective of SIF is to provide secure integration across domains while reducing the operational complexity of traditional network integration. Instead of relying on static network reachability, public IP exposure, VPN tunnels, or manually managed firewall rules, SIF uses identities, services, policies, and encrypted overlay connections to ensure that only authorised entities can discover, reach, or consume specific COP-PILOT services.

3.5.1 Design

The COP-PILOT SIF follows an identity-first secure overlay design. Every participating component, service, client, or gateway is represented by an identity. Connectivity is then governed through explicit service definitions and access policies. This enables COP-PILOT to create private, service-specific communication paths between authorised entities, while keeping the underlying services unreachable to unauthorised actors.

At a high level, the SIF is composed of the logical elements listed in Table 3-17.

Table 3-17: Logical SIF elements.

SIF Element	Description
SIF Controller	Provides the control plane for identity management, service definition, policy creation, and fabric configuration. This interface is used by the ESO to request creation, update, and deletion of SIF entities such as identities, services, connections, and policies.
SIF Edge Routers	Provide the distributed data-plane fabric used to route encrypted overlay connections between authorised endpoints.
SIF Clients	Software clients installed on platform or cluster nodes to connect them securely to the COP-PILOT SIF.
SIF Frontdoor	A controlled exposure pattern used to publish (via a public URL) selected web interfaces, such as the Business Portal and ESO UI, without directly exposing their infrastructure endpoints.
SIF Services	Logical service definitions representing applications, APIs, portals, orchestrators, data services, or cluster resources exposed through the fabric.
SIF Policies	Policy objects defining which identities are authorised to dial or bind specific SIF services.

The SIF design supports both **central platform integration** and **cluster-domain integration**.

In the central platform, SIF is used to securely expose and interconnect COP-PILOT management services, including the Business Management Portal, ESO, and other central components. Selected graphical interfaces can be exposed through the SIF Frontdoor so that authorised users can access them through controlled (and authorised) entry points.

Across Clusters, SIF clients are deployed on virtual machines, Kubernetes nodes, service gateways, or other suitable endpoints. These clients enrol into the COP-PILOT SIF using issued identities and then host or consume authorised services. This allows each cluster domain to participate in the COP-PILOT platform without requiring direct inbound exposure of local infrastructure.

The SIF also supports programmable private connectivity for orchestrated services. When a COP-PILOT service order requires communication across domains, the orchestrators can coordinate with the SIF to create the corresponding secure service connectivity. This gives service providers and vertical users a simplified “single platform” experience even when services are deployed across different compute clusters, domains, and ownership boundaries.

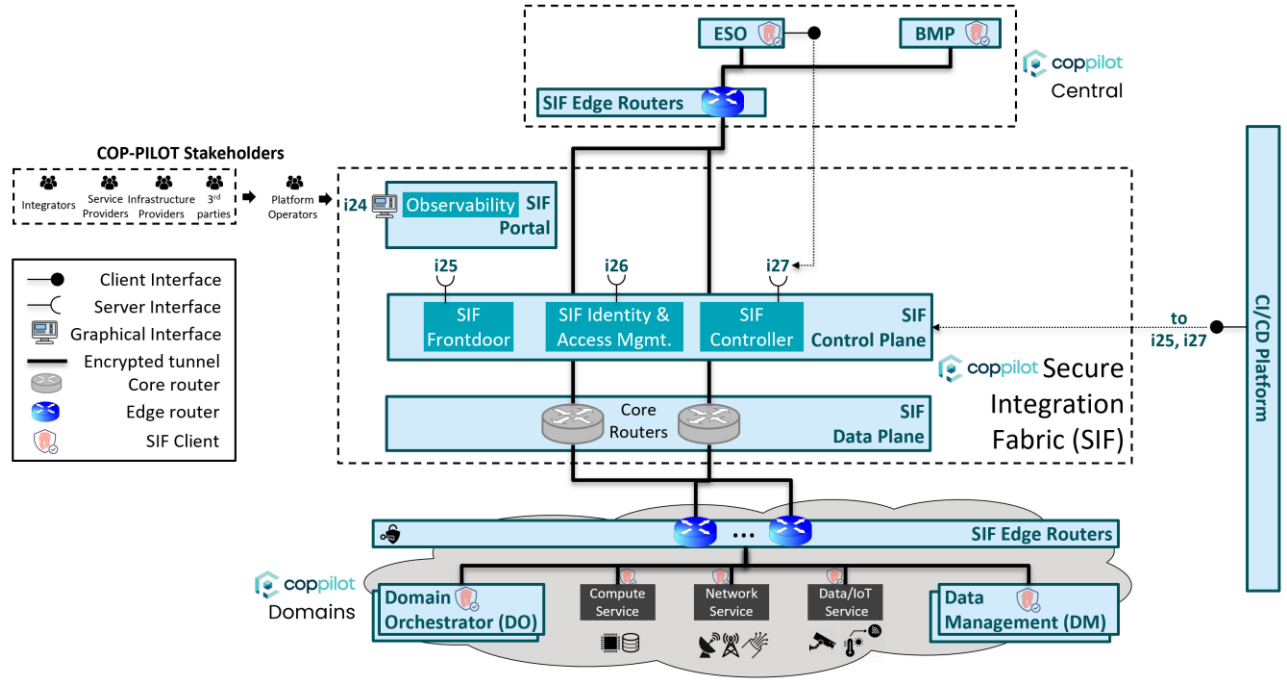

Figure 3-10: Abstract design of the COP-PILOT Secure Integration Fabric.

3.5.2 Technologies

The COP-PILOT SIF is implemented using **NetFoundry/OpenZiti**, an open-source zero-trust networking platform. OpenZiti enables applications, APIs, devices, and users to be connected through private, encrypted, identity-based overlay networks.

The main technologies used by the SIF are summarised in Table 3-18.

Table 3-18: Technologies used by SIF.

Technology	Role in COP-PILOT SIF
OpenZiti Controller	SIF control plane for identities, services, policies, posture configuration, and network control.
OpenZiti Edge Routers	Distributed overlay data-plane routers used to establish secure paths between authorised endpoints.
Ziti Edge Tunnel	Endpoint tunnelling client used to connect VMs, services, and cluster nodes to the SIF.
OpenZiti LLM Gateway	Provides an OpenAI-compatible gateway for controlled LLM access across model providers and self-hosted model backends, using OpenZiti-based identity, encrypted connectivity, and policy-governed access.
OpenZiti MCP Gateway	Provides secure access from AI assistants and MCP-compatible clients to internal MCP tool servers without exposing public endpoints; supports tool aggregation, namespacing, and client-specific tool visibility.

MCP Bridge / MCP-compatible clients	Enables AI assistants, developer tools, and operational agents to access authorised COP-PILOT tools and APIs through the MCP Gateway.
OpenAI-compatible LLM APIs	Supported interface pattern for routing model requests through the LLM Gateway to commercial or self-hosted model providers.
Ziti CLI	Command-line tool used by CI/CD pipelines and administrators to automate identity enrolment, service creation, and policy management.
OpenZiti SDKs	Optional application-embedded connectivity model for services that require direct programmatic integration with the fabric.
Ziti Frontdoor	Web exposure pattern used to provide controlled access to selected COP-PILOT user interfaces.
Jenkins	Automates SIF onboarding, service exposure, and configuration workflows.
Keycloak	Provides identity and access management for COP-PILOT platform users and CI/CD access control.
Harbor / GitHub	Store SIF-related artefacts, automation scripts, and integration pipelines.
Kubernetes / Docker	Runtime environments for platform and cluster services that may be exposed through the SIF.

The use of OpenZiti allows the COP-PILOT platform to implement secure connectivity without depending on the addressing, routing, or firewall configuration of the underlying infrastructure. Services can remain private and unreachable by default, while authorised clients consume them through policy-controlled SIF services.

The SIF supports the technical capabilities listed and explained in Table 3-19.

Table 3-19: Technical capabilities supported by SIF.

Capability	Description
Identity-based access	Every SIF participant is assigned an identity used for authentication and policy enforcement.
Service-level connectivity	Access is granted to named services rather than broad network segments.
Encrypted overlay communication	Communication between endpoints is protected through encrypted overlay sessions.
No inbound exposure requirement	Services can be hosted behind NAT, firewalls, private networks, or cluster boundaries without opening inbound ports.
Policy-driven access control	Dial and bind policies define which identities can consume or host each service.
Programmable fabric management	SIF entities can be created and managed through APIs and automation pipelines.
Multi-domain integration	Services can be securely connected across central and distributed COP-PILOT domains.
Controlled web exposure	Selected UIs can be exposed through SIF Frontdoor while keeping backend services private.

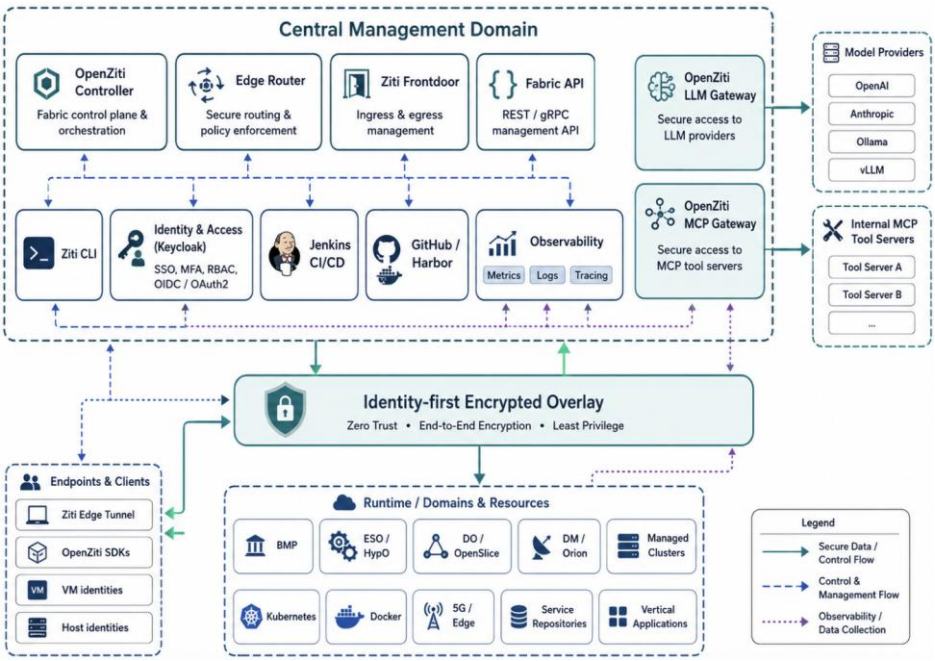

Figure 3-11: Technical design of the COP-PILOT SIF based on OpenZiti.

This figure should show the SIF Controller and Edge Router as central fabric components, with distributed SIF clients installed in central and cluster domains.

3.5.3 Interfaces

The COP-PILOT SIF exposes and consumes several interfaces, covering fabric management, client onboarding, service hosting, service consumption, and graphical access to selected platform services.

The SIF interfaces are grouped into four categories as follows:

1. **Management interfaces**, used by the ESO and automation pipelines to configure the fabric.
2. **Client interfaces**, used by SIF clients to enrol and connect to the fabric.
3. **Service interfaces**, used by applications and platform components to host or consume services.
4. **Graphical interfaces**, used to expose selected COP-PILOT UIs through the SIF Frontdoor.

A typical SIF service exposure workflow is as follows:

1. A COP-PILOT platform or cluster service is selected for secure exposure.
2. An identity is created for the hosting endpoint.
3. A SIF service definition is created, including intercept and host configuration.
4. Bind policies are created to authorise the hosting identity.
5. Dial policies are created to authorise consuming identities.
6. The host endpoint enrols into the SIF using the SIF client.
7. Authorised clients can consume the service through the fabric.
8. The underlying service remains private and is not directly exposed to the public network.

Table 3-20 summarises the main SIF interfaces.

Table 3-20: SIF interfaces.

Endpoint	Version	Description
OpenZiti Controller API	v1.6.17	Native OpenZiti API for managing identities, services, edge routers, policies, and posture-related configuration. Corresponds to interface i27 in Figure 3-10.
Ziti enrolment JWT	v1.6.17	Used by SIF clients to enrol into the COP-PILOT SIF as authorised identities. Corresponds to interface i26 in Figure 3-10.
Ziti Edge Tunnel	v1.6.17	Local client interface that intercepts authorised traffic and forwards it through the SIF overlay. This is a client interface not visualized in Figure 3-10.
SIF Frontdoor HTTPS endpoint	v1.6.17	Provides controlled web access to selected COP-PILOT platform UIs. Corresponds to interface i25 in Figure 3-10.

The SIF interfaces support both manual administration and automated orchestration. In the first platform release, the primary integration pattern is automation through CI/CD pipelines and ESO-facing Fabric APIs. This provides a repeatable method for onboarding platform components and cluster resources into the COP-PILOT secure integration environment.

3.6 FIRST VERSION OF PLATFORM INTERFACES

Table 3-21 summarizes both server and client interfaces of the COP-PILOT platform mappings the components they belong to, their specific scope within a component, as well as their relationship with standards. It is evident that the COP-PILOT platform implements a broad range of standards in the Cloud-Edge-IoT areas, while most importantly the entire platform provides these implementations as open-source.

Table 3-21: Summary of platform interfaces and relation with standards.

COP-PILOT Platform Component	Interface number	Interface role	Scope	Relation with standards
Business Management Portal	i1	UI	BMP Frontend	Consumes TMF620 Product Catalogue [7], TMF622 Product Ordering [8], and TMF637 Product Inventory [9] Management APIs
	i2	API	BMP Backend AI	N/A
	i3	API	BMP Backend MCP	Open Model Context Protocol (MCP) [71] standard for connecting AI applications to external systems
	i4	API	BMP Backend Prod. Mgmt.	Standardized TMF620 Product Catalogue [7], TMF622 Product Ordering [8], and TMF637 Product Inventory [9] Management APIs Standardized TMF633 Service Catalogue [10], TMF641 Service Ordering [11], and TMF638 Service Inventory [12] Management APIs
End-to-end Service Orchestrator	i5	UI	ESO Portal	Consumes all i8 TMF APIs
	i6	API	ESO Backend – SIF	Consumes the SIF Fabric API / OpenZiti Controller API to provision secure connectivity constructs, including identities, services, service policies, dial/bind policies and service exposure. Uses REST/HTTPS interfaces protected with TLS 1.2 / TLS 1.3 and X.509 certificates. This is a SIF-specific integration interface rather than a TMF interface.
	i7	API	ESO Backend – Secret	N/A. Wraps the industry-grade Vault API
	i8	API	ESO Backend – Service	TMF633 Service Catalogue [10], TMF641 Service Ordering [11], TMF638 Service Inventory [12], and TMF640 Service Activation [13] Management APIs TMF632 Party [17] and TMF669 Party Role [18] Management APIs TMF642 Alarm [22] Management API
	i9	API	ESO Backend – Peering	TMF632 Party [17] Management API
	i10	API	ESO Backend – Service Repositories/Registries	Open Container Initiative (OCI) distribution specification [72]
	i11	API	ESO Backend – Client interface to DO Service API	Consumes standardized set of TMF APIs by the DO (i11 belongs to DO)
	i13	API	ESO Backend – Client interface to Compute Clusters	ETSI GS NFV-SOL 018 V5.2.1 [73] ETSI GS NFV-SOL 020 V5.2.1 [74]
	i17	API	ESO Backend – Client interface to Telemetry Service	Consumes the Prometheus API [30]: the de facto industry standard for cloud-native monitoring and metrics
	i30	API	ESO Backend – Client interface to SIF Controller	REST/HTTPS API used by the ESO / HypO Fabric API to manage COP-PILOT SIF entities, including OpenZiti network controllers, identities, services, service policies, dial/bind policies, and secure service exposure. Uses TLS 1.2 / TLS 1.3, X.509 certificates, JWT-based enrolment, and OpenZiti identity/service/policy constructs. This is a SIF-specific secure connectivity interface rather than a TMF interface.
Domain Orchestrator	i10	API	DO Backend – Service Repositories/Registries	Open Container Initiative (OCI) distribution specification [72]
	i11	API	DO Backend – Service API (covers also	TMF633 Service Catalogue [10], TMF641 Service Ordering [11], TMF638 Service Inventory [12], and TMF640 Service Activation [13] Management APIs

			resource API, but it is internal to domains)	TMF634 Resource Catalogue [14], TMF652 Resource Ordering [15], and TMF639 Resource Inventory [16] Management APIs TMF632 Party [17] and TMF669 Party Role [18] Management APIs TMF642 Alarm [22] Management API
	i12	UI	DO Portal	Consumes all i11 TMF APIs
	i13	API	DO Backend – Client interface to Compute Clusters	ETSI GS NFV-SOL 018 V5.2.1 [73] ETSI GS NFV-SOL 020 V5.2.1 [74]
	i14	API	DO Backend – Client interface to satellite transport network controller	N/A. The COP-PILOT DO does not implement such an API, as it is not needed in the project
	i15	API	DO Backend – Client interface to cellular transport network controller	N/A. Consumes proprietary, cluster-tailor-made operations and maintenance APIs Additional support is expected for the O-RAN O1 Interface [77], if needed
	i16	API	DO Backend – Client interface to optical transport network controller	IETF Network Slice API as per RFC9543 [75]
	i17	API	DO Backend – Client interface to Telemetry Service	Consumes the Prometheus API [30]: the de facto industry standard for cloud-native monitoring and metrics Wraps this API with TMF628 Performance Management API [63]
Data Management	i18	UI	DM Observability Dashboards	Consumes ETSI NGSI-LD-based i19-i22 APIs
	i19	API	DM Backend – Entity	ETSI NGSI-LD specification [76]
	i20	API	DM Backend – Subscription	
	i21	API	DM Backend – Context	
	i22	API	DM Backend – Registry	
	i23	API	DM Backend – SBI to IoT agents	
Secure Integration Fabric	i24	UI	SIF Portal	Provides administrative and operational access to the SIF management plane. Access is provided over HTTPS/TLS and may integrate with OAuth 2.0 / OpenID Connect-based platform IAM where deployed.
	i25	API	SIF Control Plane – Frontdoor	Provides controlled web exposure for selected COP-PILOT platform UIs and services through SIF Frontdoor. Uses HTTPS/TLS, X.509 certificates and OpenZiti service definitions/policies to expose services without directly exposing the underlying infrastructure endpoints.
	i26	API	SIF Control Plane – IAM	Supports SIF identity enrolment and client onboarding using JWT-based enrolment, X.509 identity certificates and OpenZiti identity constructs. Where integrated with the wider COP-PILOT IAM stack, this can be combined with OAuth 2.0 / OpenID Connect-based authentication and authorisation workflows.
	i27	API	SIF Control Plane – Controller	Native OpenZiti Controller API used to create, read, update and delete SIF entities such as identities, services, edge routers, service policies, dial/bind policies and posture-related configuration.

				Uses REST/HTTPS interfaces protected with TLS 1.2 / TLS 1.3 and X.509 certificates.
CI/CD Platform	i28	UI	CI/CD services	The CI/CD stack leverages Keycloak as a centralized Identity Provider to ensure secure and standardized access across all tools. By implementing OAuth 2.0 (RFC 6749) and OpenID Connect Core 1.0 (ISO/IEC 26131:2024) Access is governed by a strict Role-Based Access Control (RBAC) model, ensuring that users are granted permissions to specific resources, folders, and environments based on their assigned team and cluster membership.
	i29	API	BMP auto-provisioning	N/A
	i31	API	ESO APIs	All exposed ESO TMF APIs
	i32	API	Cluster components and services auto-provisioning	N/A

4 PLATFORM-LEVEL INTEGRATION

This section presents the platform-level integration activities for the first version of the COP-PILOT platform. Building on the individual components introduced in Section 3, it focuses on how these components are provisioned, interconnected, validated, and prepared for use across the COP-PILOT ecosystem. The scope covers the WP3 integration of the Central Management Domain, the CI/CD stack, automated provisioning, secure exposure through SIF, and the mechanisms for interacting with Domain Orchestrators and vertical applications.

The section is organized as follows:

- **Section 4.1:** Introduction to the Platform Integration Methodology, Central Management Domain, the CI/CD stack, and the automation used for development, testing, and deployment.
- **Section 4.2:** Details on the integration of the Business Portal, End-to-End Service Orchestrator (ESO), Secure Integration Fabric (SIF), and the Domain Orchestrator (DO).
- **Section 4.3:** Presentation of common cluster-targeting integration mechanisms for DO/DM auto-provisioning and SIF client onboarding.
- **Section 4.4:** Description of workflows for ESO–DO peering, application testing, release generation, and vertical service ordering through ESO.

4.1 GENERAL INTRODUCTION

The current subsection introduces the main COP-PILOT integration aspects at the central platform level. In this context, the integration methodology is described first, focusing on CI/CD automation as one of the main tools to ensure efficiency and scalability in deployment, configuration and testing/validation workflows as the COP-PILOT platform evolves. The role of the Central Management Domain is also described, as well as the COP-PILOT components that are centrally deployed to support the different clusters at an operational level. Finally, the different services comprising the CI/CD system are described in detail.

4.1.1 COP-PILOT Platform Integration and Methodology

Integration responsibilities are divided between WP3 and WP4: WP3 handles system-level platform integration (deployment, exposure, CI/CD, registry management, and orchestration peering), while WP4 focuses on cluster-level integration (testbed preparation, connectivity, and use-case validation). Accordingly, this section describes the mechanisms enabling cluster onboarding, while WP4 deliverables detail their application in piloting environments.

The integration approach utilizes the Central Management Domain and automated CI/CD pipelines to ensure controlled deployment, configuration, and traceability from source code to artefacts. Using an Agile-driven methodology, components are developed in dedicated repositories, packaged as artefacts, and validated through automated pipelines before promotion to the shared integration environment. This incremental approach minimizes the risk of manual, non-repeatable deployments. NETC, acting as Scrum Master, coordinates these activities to ensure alignment across component teams and maintains transparency regarding pipeline performance and integration blockers.

4.1.2 Central Management Domain

The Central Management Domain serves a threefold role:

- **Hosting core components:** It provides the runtime environment for platform services such as the Business Management Portal, End-to-End Service Orchestrator (ESO), and Secure Integration Fabric (SIF).
- **Integration testing:** It hosts a multi-server environment, including a dedicated testing instance of the Domain Orchestrator, to validate components and updates before cluster exposure.
- **Workflow automation:** It supports automated operational workflows across the Central Domain - Cluster continuum, including SIF interconnection and ESO - DO peering orchestration.

To support these functions, Hetzner Cloud was selected as the European public-cloud provider, offering scalable resources (VMs, storage, networking, and backups) within European data centers and ensuring GDPR-compliant operations and recovery. Detailed information concerning the deployment locations and resource configurations of the initially provisioned resources is provided in Appendix 7.1.

Building on the interface definitions in **Table 3.21**, **Figure 4.1** illustrates the interaction between COP-PILOT stakeholders and the platform. Stakeholders interact via the CI/CD service frontend interfaces (**i28**). Meanwhile, automated deployment and operational workflows towards the Central and Domain-specific clusters are handled via backend interfaces (**i29, i31, i32**), which typically route through the **Secure Integration Fabric (SIF)**.

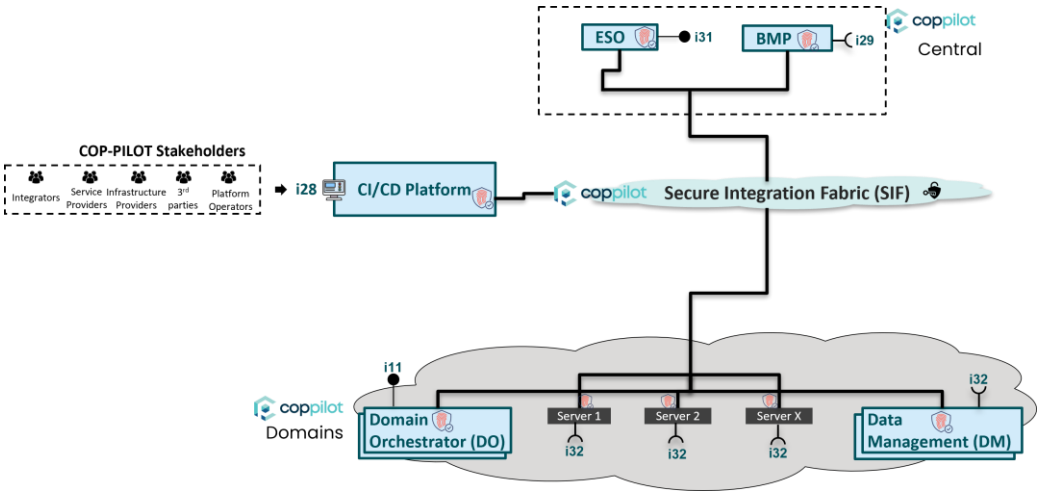

Figure 4-1 CI/CD platform and interactions within the Central Management Domain.

4.1.3 COP-PILOT CI/CD Stack Design

This Section provides detailed information about the COP-PILOT CI/CD Stack that has been put in place to support the development, integration, testing, and deployment of the Platform components. Table 4-1 provides a list of the CI/CD Stack tools which are available to all COP-PILOT component providers. The detailed descriptions of these services are provided in Appendix A (section 7). Besides the tools provided in Table 4-1, the stack also makes available OpenTofu, Ansible, and Nginx - open-source tools for infrastructure provisioning, configuration automation, and web traffic routing and serving.

Table 4-1 Main tools of the COP-PILOT CI/CD Stack.

Tool	Purpose in the integration workflow	Description
Keycloak	Centralised user management, authentication, and role-based access control across the CI/CD stack.	https://keycloak.cop-pilot.rid-intrasoft.eu/
Jenkins	Automation server for build, test, packaging, release, and deployment pipelines.	https://jenkins.cop-pilot.rid-intrasoft.eu
Harbor	Container and artefact registry for versioned images and deployment packages.	https://harbor.cop-pilot.rid-intrasoft.eu
Portainer	Operational interface for monitoring and managing containerised runtime environments.	https://portainer.cop-pilot.rid-intrasoft.eu
GitHub Organization	Source-code management for COP-PILOT platform components and deployment artefacts.	https://github.com/cop-pilot-eu

4.2 CENTRAL PLATFORM INTEGRATION

4.2.1 Business Portal auto-provisioning and integration

The COP-PILOT Business Portal (Section 3.1) is a core component whose delivery is integrated into the CI/CD stack to ensure repeatable updates, validation, and deployment. Once the workflow is triggered (see Appendix 7.4), the pipeline automatically executes the following seven stages:

1. **Lint validation:** Detects code-quality and formatting issues.
2. **Production image build:** Creates the Docker image via Docker Compose.
3. **Functional smoke testing:** Verifies portal responsiveness on port 3000.
4. **Image publication:** Pushes the validated image to the COP-PILOT Harbor registry.
5. **Deployment:** Deploys the latest image to the Business Portal server.
6. **Admin provisioning:** Optionally seeds an initial administrator user.
7. **SIF exposure:** Exposes the portal via the SIF Frontdoor (Section 4.2.3) at: https://business.portal.cop-pilot.rid-intrasoft.eu/.

The full workflow and execution interactions are depicted in Appendix 8.1.

4.2.2 ESO auto-provisioning and integration

The End-to-End Service Orchestrator (ESO, Section 3.2) is a core platform component integrated into the COP-PILOT CI/CD stack to ensure controlled, repeatable deployment. A dedicated pipeline automates the deployment of the HypO orchestrator on Kubernetes by applying required manifests and performing post-deployment validation checks. Whenever updates are introduced to the HypO, the pipeline is triggered to deploy the latest validated version. This process follows the same principles as the DO deployment workflow (Appendix 8.3), ensuring automated deployment and secure exposure. As part of this integration, HypO is connected to the SIF for secure communication with platform components and cluster domains, while the HypO user interface is exposed via the Ziti Frontdoor for authorized access.

4.2.3 SIF auto-provisioning and integration

The Secure Integration Fabric (SIF in Section 3.5) serves as the secure communication layer for the COP-PILOT Central Management Platform, providing the connectivity foundation for component

exposure, service access, and client onboarding. To support this, dedicated CI/CD pipelines automate SIF infrastructure deployment, OpenZiti service exposure, and the onboarding of authorized clients. This approach enables service exposure without opening platform components to the public network. Instead, access is strictly governed by OpenZiti identities and policies. These pipelines also form the basis for the workflows described in Section 4.3. Additionally, SIF provides the Ziti Frontdoor capability for the secure public exposure of services like the HypO and Business Management Portal interfaces (see Figure 4-2).

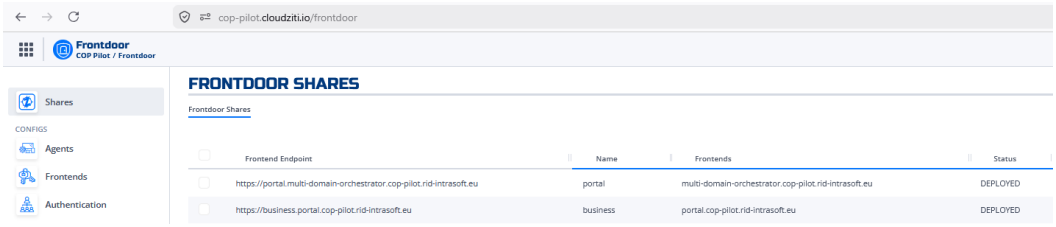

Figure 4-2 Ziti Frontdoor service.

4.2.4 Test DO integration

A dedicated test Domain Orchestrator (DO) instance is deployed within the Central Management Domain to support platform-level integration testing. Rather than representing a specific piloting cluster, this controlled environment allows the consortium to validate DO-related pipelines, service deployment procedures, and ESO–DO interactions. By providing a stable target for testing vertical applications, service orders, and peering procedures, the test DO enables early detection of deployment issues and reduces integration risks before workflows are transferred to operational cluster environments.

4.3 COMMON PLATFORM COMPONENTS INTEGRATED ACROSS ANY COP-PILOT CLUSTER

4.3.1 DO auto-provisioning and integration

To ensure consistent deployment across all COP-PILOT cluster domains, the Domain Orchestrator (Section 3.3) is managed via dedicated CI/CD pipelines. These pipelines automate both the instantiation of the DO and its secure exposure to the End-to-End Service Orchestrator (ESO) via SIF.

Pipeline 1 - OpenSlice Deployment: This pipeline automates the deployment of a functional OpenSlice stack on Kubernetes, including:

1. **OpenSlice Services:** Deployment of the full-service set.
2. **NGINX Ingress Controller:** Configured as a NodePort service to enable SIF exposure.
3. **CRIDGE Component:** Enables Kubernetes Custom Resource management.
4. **TMF API Smoke Testing:** Verifies reachability and core API functionality.

Pipeline 2 - Secure Exposure: A second pipeline securely exposes the deployed DO to the ESO through OpenZiti. Detailed workflows for both pipelines are provided in Appendices 8.3 and 8.4.

4.3.2 DM auto-provisioning and integration

To ensure consistent deployment across COP-PILOT environments, dedicated CI/CD pipelines automate the instantiation and validation of the Data Management (DM) component (Section 3.4), based on the FIWARE Orion Context Broker and MongoDB.

Pipeline 1: DM Deployment & Validation This pipeline automates the deployment of the Orion/MongoDB stack through four stages:

1. **Preparation:** Dynamic generation of a target-specific Docker Compose file.
2. **Deployment:** Provisioning of the Orion Context Broker and MongoDB.
3. **Health Validation:** Verification of service start-up and operational status.
4. **API Smoke Testing:** Live validation of the Orion API reachability. *Execution details are available on GitHub and in Appendix 8.5.*

Pipeline 2: SIF Exposure This pipeline automates the OpenZiti configuration required to securely expose the Orion Context Broker through the SIF. It manages the creation of the following elements:

- **Intercept & Host Configs:** Defining client dialing parameters and traffic forwarding to the local Orion endpoint.
- **OpenZiti Service:** Representing the exposed Orion Context Broker.
- **Service Policies:** Implementing access control via **Dial Policies** (authorizing client identities) and **Bind Policies** (authorizing host identities).

Execution details are provided in GitHub, together with a README file describing the required configuration, execution steps, and expected outputs. The full workflow is provided in Appendix 8.4.

4.3.3 SIF Client auto-provisioning and integration

To enable secure communication between COP-PILOT clusters and the central platform, relevant servers must be enrolled in the COP-PILOT SIF (accessible via: https://cop-pilot.cloudziti.io/). This enrollment is automated via a dedicated CI/CD pipeline and script, ensuring consistent integration across all environments.

The automation pipeline executes the following stages:

1. **Dependency Installation:** Installs required OS packages and runtimes.
2. **Ziti CLI & Edge Tunnel Installation:** Deploys the latest Ziti command-line interface and the ziti-edge tunnel for network connectivity.
3. **VM Identity Enrollment:** Registers the VM as an authorized endpoint using a JWT token.
4. **Tunnel Activation & Verification:** Starts the Ziti tunnel service and validates the successful connection to the SIF.

The pipeline and automation scripts are available in the platform-integration-pipelines GitHub repository. The detailed workflows are provided in Appendix 8.2.

4.4 VERTICAL SERVICES INTEGRATION

4.4.1 ESO-DO Peering for Marketplace Synchronization

The ESO–DO peering pipeline automates the onboarding of cluster-deployed Domain Orchestrators (DOs) into the central HypO instance. The process creates a TMF Organization in HypO and initiates peering with an OpenSlice, enriching the HypO marketplace with service specifications, catalogues, and categories retrieved from the peered DOs.

The pipeline executes the following stages:

1. **HypO Authentication:** Performs authentication via OAuth2.
2. **Payload Rendering:** Prepares the required TMF Organization payload.
3. **Organization Creation:** Creates the Organization within HypO.
4. **API Validation:** Validates access to the Peering API.
5. **Peering Initiation:** Starts the peering process with the external OpenSlice domain.
6. **Catalogue Import:** Imports service specifications and categories into the HypO marketplace.

Current peering status and marketplace visibility are shown in Figure 4-3, with a detailed UML workflow provided in Appendix 8.6.

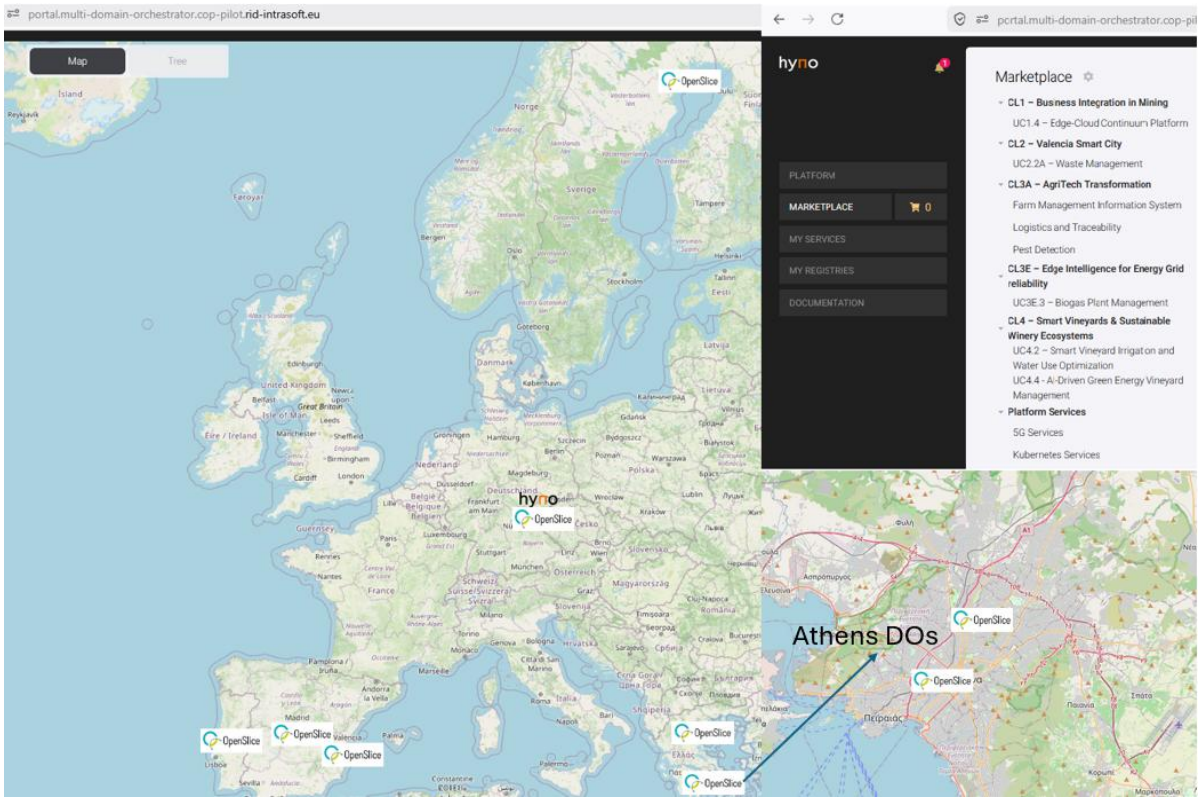

Figure 4-3 Peered DOs under the ESO and the ESO Marketplace.

4.4.2 Cluster Applications' Build and Validation

The COP-PILOT continuous integration pipeline provides a standardised build and validation process for all cluster applications. Each application is maintained in a dedicated repository connected to the central CI/CD platform, and any commit or new release triggers the pipeline via a webhook. The pipeline retrieves the source code and deployment descriptors, then executes dependency resolution, compilation, static code analysis, unit testing, and container image construction. Where applicable, Helm charts and deployment manifests are validated against the target Kubernetes environments. If any validation step fails, the pipeline terminates and reports the outcome to the Service Provider, preventing faulty software from progressing further in the release lifecycle. It is assumed that the corresponding service definitions have already been onboarded into the orchestrator catalogue by the platform operator during an initial one-off onboarding process.

The complete process is organised into three sequential stages, illustrated end-to-end in Figure 4-4. Stage 1 corresponds to the build and validation process described in this section, Stage 2 covers release management (Section 4.4.3), and Stage 3 covers deployment validation using the COP-PILOT CD plugin (Section 4.4.4).

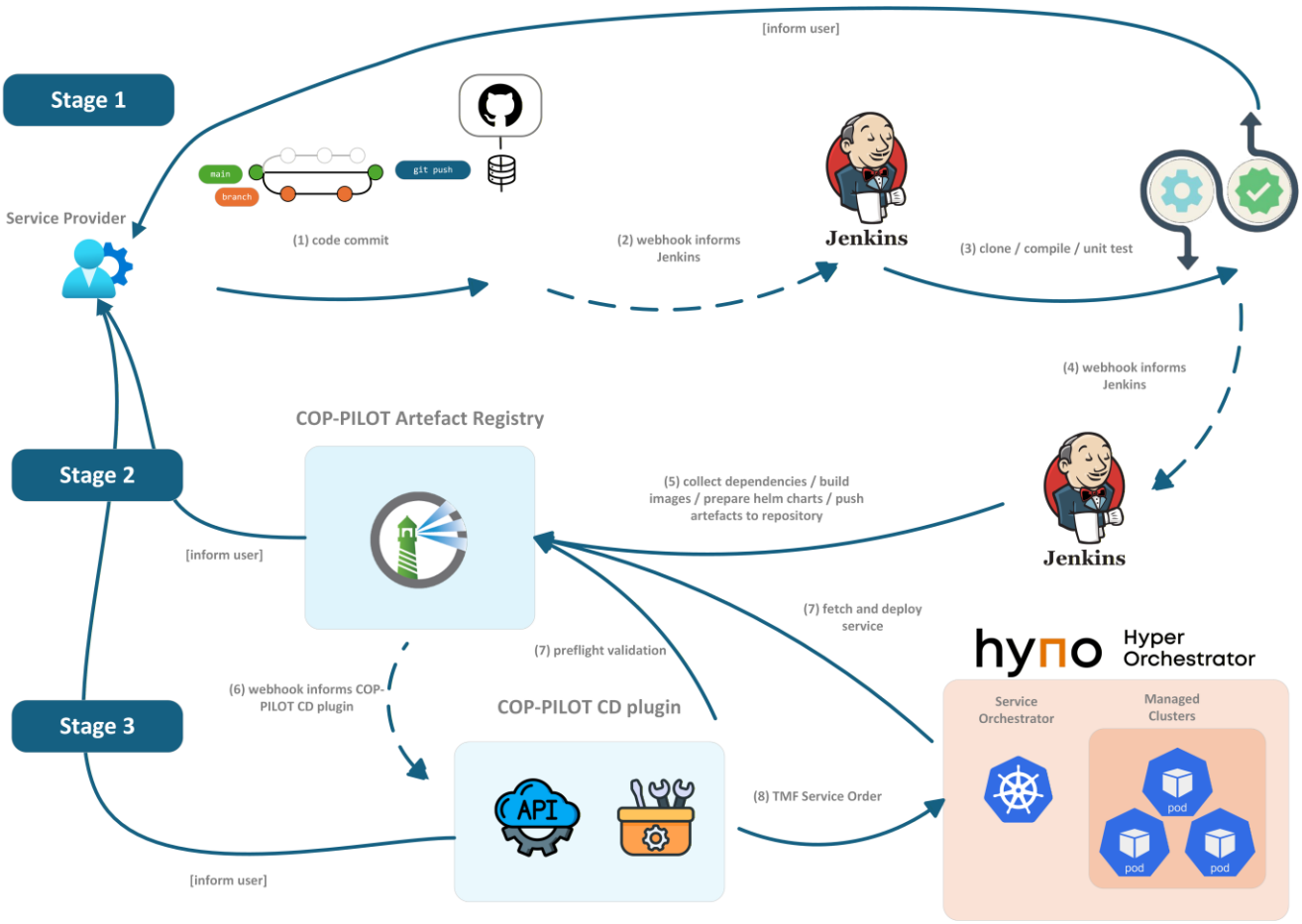

Figure 4-4 End-to-end COP-PILOT CI/CD flow for vertical Cluster applications.

4.4.3 Cluster Applications' Release Management

This section describes Stage 2 of the workflow shown in Figure 4-4. Following successful validation, the CI/CD platform automatically creates a new application release. The release comprises immutable deployment artefacts, including the container image and the corresponding Helm chart, which are versioned and published to the central COP-PILOT Artefact Registry.

Versioning ensures that every software release is uniquely identifiable, reproducible, and traceable throughout its lifecycle. The Artefact Registry serves as the single trusted repository from which all cluster deployments retrieve validated application packages, guaranteeing that all clusters deploy identical, validated software versions.

Once the release is published, the CI/CD platform notifies the COP-PILOT CD Plugin that a new deployable version is available. This notification triggers the deployment validation workflow described in the following section.

4.4.4 Cluster Applications' Deployment Validation through the Orchestrators

This section describes Stage 3 of the workflow shown in Figure 4-4. The final stage verifies that the released application can be successfully deployed before it is promoted to production. Upon notification of a new release, the COP-PILOT CD Plugin retrieves the published artefacts and performs a set of pre-deployment validation checks, including manifest linting and validation of OCI references and deployment descriptors. The outcome of these preflight checks is reported to the Service Provider before any deployment is initiated.

The plugin then collects the deployment parameters required to redeploy the application in the test environment. It first checks whether a test service instance is already running via the TMF638 Service Inventory API. If so, the returned inventory record provides the configuration needed to redeploy the service. If no test instance is running, the plugin retrieves the corresponding service specification from the production environment via the TMF633 Service Specification API.

Using these parameters, the plugin validates the release by invoking the TM Forum Service Test Management API (TMF653) against a dedicated test environment. The test orchestrator retrieves the application image from the Artefact Registry and executes automated smoke tests that cover service startup, connectivity, and interface availability, while the release remains isolated from end users. The validation results are subsequently reported back to the Service Provider.

Once validation completes successfully, the release is promoted to production. The CD Plugin submits a TM Forum Service Order (TMF641) to the HypO End-to-End Service Orchestrator (ESO), which coordinates the release deployment. The ESO then retrieves the service artefacts from the Artefact Registry, provisions the required resources, and deploys the application. It reports deployment progress back to the CI/CD platform until the service reaches a terminal state, thereby completing the automated release workflow. The full workflow can also be found in Appendix B-8.7.

Co-funded by the European Union

5 IMPACT

This section highlights the impact of the COP-PILOT platform, focusing mostly on the relation of the core platform components with open-source communities and software (Section 5.1) as well as the various standards that COP-PILOT implements in its core (Section 5.2) to ensure interoperability with market solutions and long-term viability of the platform post the project duration. In Section 5.3 we provide evidence on the use of the platform for early demonstration activities of certain UCs.

5.1 OPEN-SOURCE CONTRIBUTIONS AND COMMUNITIES' BUILD-UP

Table 5-1 summarizes the COP-PILOT Platform components (WP3) that are released as open-source projects, either under the umbrella of broader open-source communities or as new COP-PILOT open-source projects. The released versions of all these components are also provided along with the partners behind these contributions.

Table 5-1: Release of COP-PILOT Platform Components as open-source.

COP-PILOT Platform Service Name	Link to open-source repository	Version (M18)	Contributing Partner(s)
BMP - Frontend (T3.4)	**New** ETSI BSS MDG under ETSI SDG OpenSlice • Product Catalogue view • Product Ordering view • Product Inventory view Currently under a private COP-PILOT repo: https://github.com/cop-pilot-eu/cop-pilot-portal Soon a dedicated MDG-BSS/ sub-group will host the code under https://labs.etsi.org/rep/osl/	Not versioned yet	AGE Contributing Clusters: - CL1 (LTU), - CL2 (TID), - CL3A (AGA), - CL3E (UOP), - CL4 (ONE),
BMP - Backend Prod. Management (T3.4)	**New** ETSI BSS MDG under ETSI SDG OpenSlice • Product Catalogue/Ordering/Inventory APIs • Product Catalogue/Ordering/Inventory models • Product Management Orchestrator Soon a dedicated MDG-BSS/ sub-group will host the code under https://labs.etsi.org/rep/osl/	2026Q2 (Currently part of main OSL)	TID, UOP UOP UOP TID
BMP – Backend AI (T3.4)	• AI agents: https://github.com/cop-pilot-eu/cop-pilot-llm-layer • Explainable LLM: https://github.com/cop-pilot-eu/cop-pilot-llm-observability • AI security guardrails gateway: https://github.com/cop-pilot-eu/ai-security-guardrails-gateway	v0.1.0 Not versioned yet	ONE SUITE5 IPN
BMP – Backend MCP (T3.4)	**New** ETSI BSS MDG under ETSI SDG OpenSlice • Product Catalogue MCP tool • Product Ordering MCP tool • Product Inventory MCP tool Soon a dedicated MDG-BSS/ sub-group will host the code under https://labs.etsi.org/rep/osl/	2026Q2 (Currently part of main OSL)	TID
ESO Portal (T3.1)	ETSI HypO MDG Portal under ETSI SDG OpenSlice	2026-06	UBI
ESO Backend (T3.1)	ETSI HypO MDG Backend under ETSI SDG OpenSlice: - API - Core - Clients - DevOps	2026-06	UBI
DO Portal (T3.2)	ETSI SDG OpenSlice Portal	2026Q2	UOP, PNET

DO Backend (T3.2)	ETSI SDG OpenSlice:
	- OSOM
	- TMF API
	- OSL MCP server and backend
	- Addons:
	  - Giter
	  - NaC
	
	ColonyOS
		2026Q2
		
		v1.9.13-beta
			UOP, PNET
			
			LTU

DM Platform (T3.3)	FIWARE Context Brokers: - Orion-LD (NGSI-LD) - Scorpio (NGSI-LD) - Stellio (NGSI-LD) Eclipse Arrowhead Data observability dashboard	v1.12 v6.0.1 v2.32.0 v5.2.0 v1.0	FIWARE, UBI LTU SUITE5, KON
SIF Portal and Backend (T3.5)	• OpenZiti • OpenZiti Helm Charts • OpenZiti LLM Gateway • OpenZiti MCP Gateway	v2.0.0 v2.0.2	TATA
CI/CD platform (across all WP3 tasks)	Platform Integration and automation pipelines: - https://github.com/cop-pilot-eu/platform-integration-pipelines - https://github.com/cop-pilot-eu/platform-integration-pipelines/cop-pilot-cd-plugin	v1.0.0 v1.0.0	NETC AXON

Note that in Table 5-2 we refer to the terms Software Development Group (SDG) and Module Development Group (MDG) using the abbreviated acronyms.

Table 5-2: Open-source contributions of COP-PILOT Platform Components.

COP-PILOT Platform Service Name	Open-source project	Summary of COP-PILOT contributions to the project/community	Reference to committed code (issues)
BMP - Frontend (T3.4)	COP-PILOT Business Management Portal	The entire project is contributed by COP-PILOT	
BMP - Backend Product Management (T3.4)	ETSI SDG OpenSlice	The project inherits some primitive building blocks from OpenSlice (i.e., the TMF APIs and data models) but aims to drastically extend the Product Management Orchestrator to meet the COP-PILOT BMP objectives. Actual COP-PILOT contributions will be in place in D3.2.	
BMP – Backend AI (T3.4)	Currently under COP-PILOT's GitHub: • LLM • LLM Observability • AI Guardrails Soon under ETSI SDG OpenSlice	The entire project is contributed by COP-PILOT	
BMP – Backend MCP (T3.4)	ETSI SDG OpenSlice	The project inherits some primitive building blocks from OpenSlice, such as MCP tools for Product Catalogue and Product Ordering but aims to extend the MCP backend to meet the COP-PILOT BMP objectives by e.g., covering LCM aspects through a Product Inventory MCP tool. Actual COP-PILOT contributions will be in place in D3.2.	
BMP – Secure agentic	OpenZiti LLM Gateway and OpenZiti MCP Gateway	COP-PILOT evaluates and integrates OpenZiti AI gateway components to support secure agentic access to models, tools, and internal APIs. The	Repository 1 Repository 2

communication service (T3.4)		OpenZiti LLM Gateway provides an OpenAI-compatible interface for routing authorised requests to LLM providers and self-hosted model backends, while the OpenZiti MCP Gateway enables AI assistants and MCP clients to access internal MCP tool servers without exposing public endpoints. This extends the SIF model from platform-to-platform connectivity to AI-to-model and AI-to-tool connectivity.	
ESO Portal (T3.1)	ETSI HypO MDG under ETSI SDG OpenSlice	Portal view improvements and stability fixes	Issue #1
ESO Backend (T3.1)	ETSI HypO MDG under ETSI SDG OpenSlice	- Improved integration with Keycloak and Vault platforms - Improved integration with OpenZiti through updated Fabric API - Improved Helm installer with configurable env. - Alert Management through TMF Alarm API integration with Grafana Loki - Backend stability fixes - New UML workflows - Updated documentation	Issue #1 Issue #2 Issue #3 Issue #4 Issue #5 Issue #6 Issue #7 Issue #8 Issue #9 Issue #10 Issue #11
DO Portal (T3.2)	ETSI SDG OpenSlice	Portal extensions to support the COP-PILOT stakeholders.	TMF Web Issue #40
DO Backend (T3.2)	ETSI SDG OpenSlice ColonyOS	COP-PILOT contributes a consistent and location-aware service catalogue synchronization. It also extends the addons repository with new controllers, i.e., Giter, NaC, and its orchestrator to utilize them.	Issue #31 Issue #32 NaC Issue #1
DM Platform (T3.3)	FIWARE Context Brokers - Orion-LD - Scorpio - Stellio	New NGSI-LD data models for integrating sensors and other types of data sources for the 4 COP-PILOT Clusters (CL2-CL3A, CL3E, CL4) to the Orion Context Broker	COP-PILOT NGSI Models (Currently a private COP-PILOT GitHub repository. To be open sourced at a later stage of the project)
SIF Portal and Backend (T3.5)	OpenZiti	COP-PILOT uses OpenZiti as the open-source basis for the Secure Integration Fabric. The contribution focuses on applying OpenZiti to the COP-PILOT multi-domain integration model, including identity-based service exposure, client onboarding, policy-driven connectivity, and automation through CI/CD pipelines. The work also validates OpenZiti as a secure integration layer for heterogeneous platform components, clusters, domain orchestrators, repositories, and vertical services.	OpenZiti repo COP-PILOT SIF pipeline repository Commit references to be added.
CI/CD platform (across all WP3 tasks)	COP-PILOT CI/CD platform	- Automation of infrastructure, networking, and platform-integration processes through reusable CI/CD pipelines. - Deployment and lifecycle management of OpenZiti components - Automated provisioning and validation of COP-PILOT platform components, such as the Orion-LD Context Broker, ESO, DO - Integration workflows, including ESO–DO peering between HypO and OpenSlice.	Platform integration pipelines (Entire repository created under COP-PILOT)

5.2 IMPLEMENTED STANDARDS

Table 5-3 maps the COP-PILOT platform components with relevant standards that are implemented by these components.

Table 5-3: COP-PILOT Platform Components and implemented standards.

COP-PILOT Platform Service Name	Implemented Standard and Version
BMP - Frontend (T3.4)	TMF 620 Product Catalogue Management API v4.1.0 TMF 622 Product Ordering Management API v4.0.0 TMF 637 Product Inventory Management API v4.0.0 TMF 633 Service Catalogue Management v4.0 TMF 641 Service Ordering Management v4.1.1 TMF 638 Service Inventory Management v4.0.1 Model Context Protocol (MCP)
BMP – Backend Product Management (T3.4)	
BMP – Backend AI (T3.4)	**AI agents** Model Context Protocol (MCP) **Explainable LLM** TMF: 915 AI Management API v4.0 ISO/IEC 42001:2023 The AI Management System (AIMS) ISO/IEC 23894 and Risk-Based Transparency ISO/IEC TR 24028:2020 IEEE 7001-2021: Standards for Autonomous System Transparency EU AI Act NIST AI Risk Management Framework (AI RMF) NIST AI 600-1 (Generative AI Profile) OpenTelemetry (OTel) **AI security guardrails gateway** ETSI, Securing artificial intelligence (SAI); Understanding and preventing harm from generative AI, ETSI TR 104 159 V1.1.1, Jan. 2026. ETSI, "Securing Artificial Intelligence (SAI); Security aspects of using AI/ML techniques in telecom sector," ETSI Technical Report 104 051 V1.1.1, Jun. 2025 ETSI, Securing Artificial Intelligence (SAI); Guide to Cyber Security for AI Models and Systems, ETSI TR 104 128 V1.1.1.1, May 2025 **Secure agentic communication service** IETF ZTCPP Model Context Protocol (MCP) for AI-to-tool integration, implemented through OpenZiti MCP Gateway; OpenAI-compatible API pattern for LLM/model access, implemented through OpenZiti LLM Gateway; A2A / agent-to-agent secure agent collaboration, implemented through OpenZiti Agora; TLS 1.2 / TLS 1.3, X.509 certificates, OpenZiti identities, service policies, and end-to-end encrypted overlay connectivity. Aligned with the NetFoundry AI Accelerator Program
BMP – Backend MCP	Model Context Protocol (MCP)
End-to-End Service Orchestrator Portal (T3.1)	TMF 632 Party Management v5.0 TMF 669 Party Role Management v5.0 TMF 673 Geographic Address Management v4.0 TMF 674 Geographic Site Management v4.0 TMF 675 Geographic Location Management v4.0 TMF 633 Service Catalogue Management v4.0 TMF 641 Service Ordering Management v4.1.1 TMF 638 Service Inventory Management v4.0.1 TMF 640 Service Activation and Configuration v4.0.1 TMF 634 Resource Catalogue Management v4.1.0 TMF 652 Resource Ordering Management v4.0.0 TMF 639 Resource Inventory Management v4.0.0 TMF 642 Alarm Management API v5.0
End-to-End Service Orchestrator Backend (T3.1)	

Domain Orchestrator Portal (T3.2)	All ESO TMF APIs above +
Domain Orchestrator Backend (T3.2)	TMF 702 Resource Activation Management v4.0 TMF 628 Performance Management API v5.0 TMF 620 Product Catalogue Management API v4.1.0 TMF 622 Product Ordering Management API v4.0.0 TMF 637 Product Inventory Management API v4.0.0 TMF 653 Service Test Management v.4.0.0
Data Management Platform (T3.3)	ETSI NGSI-LD ETSI NGSI-v2 Both platforms comply with GAIA-X and the EU Data Spaces policy
Secure Integration Fabric Portal and Backend (T3.5)	TLS 1.2 / TLS 1.3; X.509 certificates; OAuth 2.0 / OpenID Connect; JWT-based enrolment; REST APIs; OpenAPI-described APIs where exposed; Kubernetes APIs; Helm packaging; HTTP/HTTPS and TCP service exposure through OpenZiti service definitions; OpenZiti identity-based overlay services. **Standards alignment:** ongoing IETF ZTCPP work for Zero Trust control and policy interoperability - https://github.com/ietf-ztcpp/Charter
CI/CD platform	OAuth 2.0 / RFC 6749 OpenID Connect Core 1.0 / ISO/IEC 26131:2024 ACME (RFC 8555) RBAC

The joint outcome of Table 5-1, Table 5-2, and Table 5-3 results in a visual mapping of the final high-level COP-PILOT architecture as presented in D2.2 to open-source software, related communities, and implemented standards as depicted in Figure 5-1.

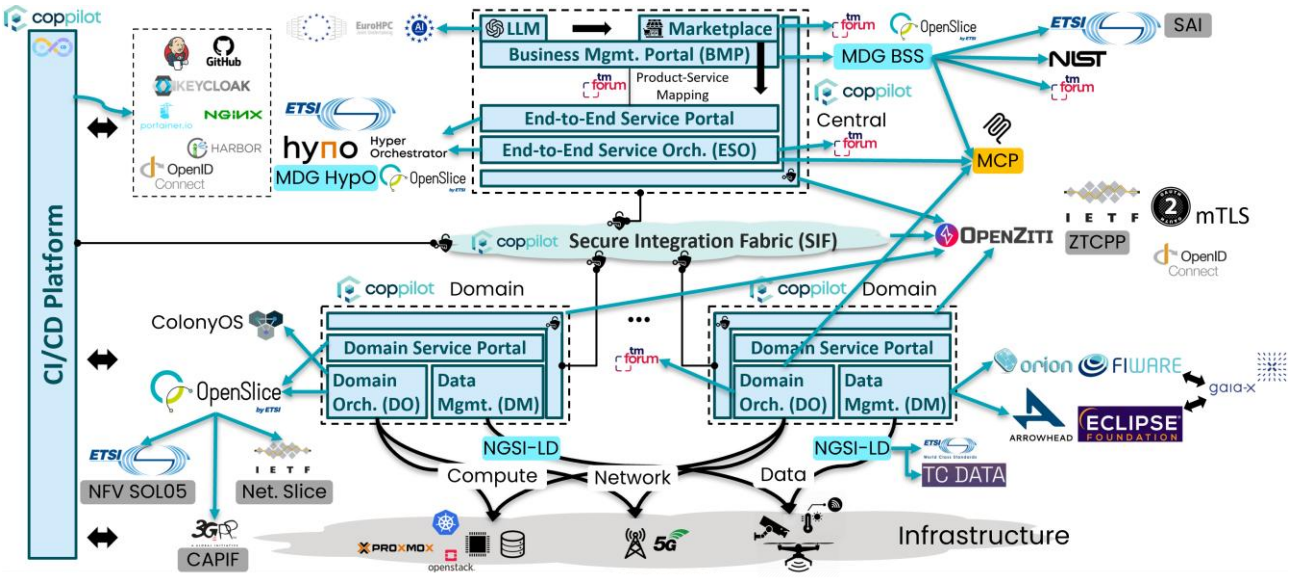

Figure 5-1: Mapping of the final COP-PILOT architecture to open-source software, communities, and implemented standards.

5.3 DISSEMINATED MATERIAL (UP TO M18)

The COP-PILOT ESO exposes a TMF-compliant marketplace of services around the 5 COP-PILOT Clusters, each pertaining to a different vertical sector. The status of this marketplace up until M18 is reported in D7.2. Table 5-4 lists recorded videos that demonstrate the COP-PILOT platform in action along with the stablished service marketplace.

Table 5-4: COP-PILOT Platform demonstration activities by M18.

Material	Involved COP-PILOT Platform Components	Purpose	Link
Orchestrators (ESO-to-DO) Peering	ESO, DO, SIF	Demonstrates the ESO's ability to peer with a DO instance and obtain the desired services exposed by this DO through standardized TMF APIs.	Video on the COP-PILOT YouTube channel
Federated ESO Service Marketplace	ESO	After the ESO has peered with multiple DO instances spread across all COP-PILOT Clusters in Europe, it obtains and promotes a central Pan-European COP-PILOT service marketplace through standardized TMF APIs.	

6 CONCLUSION

This deliverable has presented the first version of the COP-PILOT platform, marking a significant milestone in the project's development trajectory. Between M4 and M18, WP3 has successfully translated the architectural vision established in D2.1 and D2.2 into a working, deployed platform, demonstrating that the foundational components of the COP-PILOT ecosystem are operational, integrated, and ready to support the piloting and validation activities that follow.

The five core platform components (the Business Management Portal, the End-to-End Service Orchestrator, the Domain Orchestrator, the Data Management platform, and the Secure Integration Fabric) have each been designed, implemented, and integrated within a Central Management Domain hosted on European cloud infrastructure. Together, they realise the hierarchical orchestration model at the heart of the COP-PILOT architecture, enabling secure, standards-based service management across multiple geo-distributed domains. The automated CI/CD stack established during this period ensures that the platform can be provisioned, updated, and validated in a repeatable and controlled manner, providing a reliable foundation for the increasingly complex integration activities that will characterise the second phase of the project.

A notable achievement of this first release is the successful peering of the ESO with Domain Orchestrators deployed across all five COP-PILOT piloting clusters, resulting in a functioning Pan-European service marketplace populated with cluster-specific services. This demonstrates not only the technical correctness of the platform's inter-component interfaces but also its readiness to support real vertical sector applications.

From an impact perspective, the first platform release establishes COP-PILOT as an active contributor to several major open-source communities, including ETSI SDG OpenSlice, FIWARE, and OpenZiti, and demonstrates alignment with more than twenty TMF APIs and key ETSI standards. This commitment to openness and standardisation is fundamental to the platform's long-term viability and market relevance beyond the project's duration.

6.1 PLAN TOWARDS FINAL PLATFORM VERSION

While the first release confirms the soundness of the platform's core architecture and integration approach, several important capabilities remain to be delivered in the second development cycle, to be reported in D3.2. The planned extensions are summarised in Table 6-1 on a per-component basis.

Table 6-1: Planned features for the final COP-PILOT Platform release (D3.2).

COP-PILOT Platform Component	Features Planned towards final release (D3.2)
Business Management Portal (BMP)	Product-level Orchestration • Orchestration workflow for Product LCM (Product Inventory) • Dynamic synchronization between BMP Product Marketplace and ESO Service Marketplace
End-to-end Service Orchestrator (ESO)	LLM-enhanced ESO Marketplace • LLM chatbot portal extension • Native ESO LLM (potential agent-to-agent integration with BMP and DO LLMs) • LLM-assisted service catalogue browsing • LLM-assisted service ordering Multi-domain compute-as-a-service • Provisioning of Hub compute clusters as a service • Provisioning of Managed compute clusters as a service and dynamic registration to the Hub • Provisioning of compute scheduler as a service with configurable labels according to business needs Improved orchestration workflows for complex multi-domain services • Composite service bundles for collaborative COP-PILOT services • Joint provisioning of composite service bundles with multiple DOs • Joint LCM of composite services with multiple DOs Improved end-to-end telemetry of complex multi-domain services • Cross-domain telemetry federation • End-to-end performance dashboards
Domain Orchestrator (DO)	5G Core controllers • OTE's network slice management (Cluster 3E) flow integration • NOKIA's Network as Code (NaC – Cluster 4) flow integration • Tailor-made controllers' development Secret Management • Sensitive information management in UI view • Sensitive information management during orchestration Service Specification characteristic validation • Characteristics type/value validation during design • Characteristics value validation during ordering • Characteristics updated value validation during operation Improved IETF Network Slice controller • Evaluate 2026Q2 release functionality • SLO/SLA templates discovery enhancements Improved AI-native functionalities • Evaluate 2025Q4 release agent-based orchestration • Integrate agent skills support in the 2025Q2 release's MCP server • MCP server tools enhancements

Data Management (DM)	Bug fixes and features from FIWARE community. No contribution by COP-PILOT is expected in the second half of the project due to FIWARE not being in the project anymore.
Secure Integration Fabric (SIF)	During the second project period, TATA / NetFoundry will evolve the COP-PILOT Secure Integration Fabric from the first-release SIF deployment into a more productised, observable and AI-ready secure integration layer. Planned work will focus on reusable SIF capabilities that can benefit COP-PILOT and wider NetFoundry / OpenZiti users, including improved SIF onboarding workflows, reusable service-exposure templates, enhanced documentation, and clearer API-driven integration patterns for COP-PILOT components such as DO, DM, BMP, vertical applications, service repositories, gateways and selected platform services. TATA / NetFoundry will also extend Frontdoor / Universal Exchange capabilities for controlled access to authorised COP-PILOT UIs, APIs and partner-facing services, including collaborators or external systems that cannot deploy Ziti endpoints. This will support approved access patterns such as HTTPS, mTLS, IPsec or other controlled protocols, while preserving Ziti-based identity, policy enforcement and private service reachability on the COP-PILOT side. A further work track will productise and integrate the emerging OpenZiti agentic AI portfolio into the SIF roadmap. This includes releasing and validating NetFoundry-supported functionality for OpenZiti LLM Gateway, OpenZiti MCP Gateway, and A2A / OpenZiti Agora-style secure agent-to-agent communication, then evaluating their use within COP-PILOT BMP, ESO and agentic workflows where relevant. The target is to support controlled AI-to-model, AI-to-tool and agent-to-agent communication using a common identity-first, policy-driven security model. TATA / NetFoundry will also continue planned work on SIF observability, visibility and micro segmentation. This includes richer control-plane and data-plane telemetry, service exposure status, policy decision evidence, session and path context, and exportable events for COP-PILOT observability and security operations. TATA /NetFoudry will investigate and, where appropriate, validate zLAN / eBPF / Zeek-based local enforcement and visibility patterns, including process-, socket- and container-level telemetry, local allow/deny decisions, and selective inspection or telemetry export for designated services. Finally, TATA / NetFoundry will continue operational hardening of the SIF platform for the final release, including improved self-hosted deployment options, HA validation, documentation, guided onboarding workflows, reusable templates, and service-aware overlay behaviour for constrained or unreliable cluster connectivity. This work will support COP-PILOT's objective of simplifying secure multi-domain integration while reducing dependence on manual firewall, VPN, routing and underlay reconfiguration.
CI/CD Platform	Evolution of the COP-PILOT CI/CD stack into a mature and reusable automation framework for the final platform release. Planned work includes consolidating pipelines for BMP, ESO, DO, DM, SIF, SIF clients, DO-ESO peering, and vertical applications, introducing reusable pipeline templates, improving traceability from GitHub commits to deployed services, and integrating automated quality, security, and validation activities, including linting, smoke tests, deployment validation, artefact scanning, dependency checks. The final CI/CD platform will support repeatable, auditable, and secure deployment of COP-PILOT components across the central management domain and all clusters, with improved visibility of pipeline execution, deployment status, artefact versions, and integration health.

REFERENCES

[1] COP-PILOT D2.1 “Ecosystem Definition and Requirements”, Available: link

[2] COP-PILOT D2.2 “COP-PILOT architecture and functionalities”, Available: link

[3] COP-PILOT D7.2 “Exploitation, innovation and IPR management report – Interim“, Available: link

[4] ETSI SDG OpenSlice, “HypO Multi-domain Service Orchestrator”, Available: https://labs.etsi.org/rep/groups/osl/hypo

[5] ETSI SDG OpenSlice, “Network as a Service, made easy”, Available: https://osl.etsi.org

[6] OpenZiti, “Cloak Your Network. Secure Services not IPs”, Available: https://netfoundry.io/docs/openziti/

[7] TMForum, “TMF620 Product Catalog Management API v4.1.0”, 2021, Available: https://www.tmforum.org/open-digital-architecture/open-apis/product-catalog-management-api-TMF620/v4.1

[8] TMForum, “TMF622 Product Ordering Management API v4.0.0”, 2021, Available: https://www.tmforum.org/open-digital-architecture/open-apis/product-ordering-management-api-TMF622/v4.0

[9] TMForum, “TMF637 Product Inventory Management API v4.0.0”, 2021, Available: https://www.tmforum.org/open-digital-architecture/open-apis/product-inventory-management-api-TMF637/v4.0

[10] TMForum, “TMF633 Service Catalog Management API v4.0.0”, 2021, Available: https://www.tmforum.org/open-digital-architecture/open-apis/service-catalog-management-api-TMF633/v4.0

[11] TMForum, “TMF641 Service Ordering Management API v4.1.1”, 2021, Available: https://www.tmforum.org/open-digital-architecture/open-apis/service-ordering-management-api-TMF641/v4.1

[12] TMForum, “TMF638 Service Inventory Management API v4.0.1”, 2020, Available: https://www.tmforum.org/open-digital-architecture/open-apis/service-inventory-management-api-TMF638/v4.0

[13] TMForum, “TMF640 Service Activation and Configuration API v4.0.1”, 2020, Available: https://www.tmforum.org/open-digital-architecture/open-apis/service-activation-management-api-TMF640/v4.0

[14] TMForum, “TMF634 Resource Catalog Management API v4.1.0”, 2021, Available: https://www.tmforum.org/open-digital-architecture/open-apis/resource-catalog-management-api-TMF634/v4.1

[15] TMForum, “TMF652 Resource Ordering Management API v4.0.0”, 2020, Available: https://www.tmforum.org/open-digital-architecture/open-apis/resource-order-management-api-TMF652/v4.0

[16] TMForum, “TMF639 Resource Inventory Management API v4.0.0”, 2020, Available: https://www.tmforum.org/open-digital-architecture/open-apis/resource-inventory-management-api-TMF639/v4.0

[17] TMForum, “TMF632 Party Management API v5.0.0”, 2023, Available: https://www.tmforum.org/open-digital-architecture/open-apis/party-management-api-TMF632/v5.0

[18] TMForum, “TMF669 Party Role Management API v5.0.0”, 2023, Available: https://www.tmforum.org/open-digital-architecture/open-apis/party-role-management-api-TMF669/v5.0

[19] TMForum, “TMF675 Geographic Location Management API v4.0.0”, Available: https://www.tmforum.org/open-digital-architecture/open-apis/geographic-location-management-api-TMF675/v4.0

[20] TMForum, “TMF674 Geographic Site Management API v4.0.1”, 2020, Available: https://www.tmforum.org/open-digital-architecture/open-apis/geographic-site-management-api-TMF674/v4.0

[21] TMForum, “TMF673 Geographic Address Management API v4.0.0”, 2020, Available: https://www.tmforum.org/open-digital-architecture/open-apis/geographic-address-management-api-TMF673/v4.0

[22] TMForum, “TMF642 Alarm Management API v5.0”, 2025, Available: https://www.tmforum.org/open-digital-architecture/open-apis/alarm-management-api-TMF642/v5.0

[23] Cloud Native Computing Foundation, Kubernetes, “Production-Grade Container Orchestration”, Available: https://kubernetes.io/

[24] Docker Inc., “Docker-compose: Define and run multi-container applications with Docker” Available: https://docs.docker.com/compose/

[25] Cloud Native Computing Foundation, “Helm: The package manager for Kubernetes”, Available: https://helm.sh/

[26] Apache, “Kafka: Open-source distributed event streaming platform”, Available: https://kafka.apache.org/

[27] PostgreSQL, “The World's Most Advanced Open-source Relational Database”, Available: https://www.postgresql.org/

[28] Cloud Native Computing Foundation, “Keycloak: Open-source Identity and Access Management”, Available: https://www.keycloak.org/

[29] Hashicorp Vault: “Manage secrets and protect sensitive data”: Available: https://developer.hashicorp.com/vault

[30] Cloud Native Computing Foundation, “Prometheus: From metrics to insight “, Available: https://prometheus.io/

[31] Prometheus operator, Available: https://prometheus-operator.dev/

[32] Grafana Labs, “Grafana: Visualize your data, optimize your performance”, Available: https://grafana.com/oss/grafana/

[33] Grafana Labs, “Grafana Mimir: Open-source, horizontally scalable, highly available, multi-tenant TSDB for long-term storage for Prometheus”, Available: https://grafana.com/oss/mimir/

[34] Grafana Labs, “Grafana Tempo: Distributed tracing system for better application performance”, Available: https://grafana.com/oss/tempo/

[35] Grafana Labs, “Grafana Loki: Log monitoring for faster troubleshooting at scale”, Available: https://grafana.com/oss/loki/

[36] Cloud Native Computing Foundation, “OpenTelemetry: High-quality, ubiquitous, and portable telemetry to enable effective observability”, Available: https://opentelemetry.io/

[37] RedHat, “Kogito Cloud-Native Business Automation”, Available: https://kogito.kie.org/

[38] NetFoundry, “OpenZiti: Open-Source Zero-Trust Platform”, Available: https://openziti.io/

[39] Go Helm client library, Available: https://github.com/mittwald/go-helm-client

[40] Native Go Helm library, Available: https://github.com/helm/helm

[41] Kompose library, Available: https://github.com/kubernetes/kompose

[42] Cloud Native Computing Foundation, “Istio”, Available: https://istio.io/

[43] ETSI, “OpenSlice (OSL) Software Development Group (SDG)”, Available: https://osl.etsi.org/

[44] ETSI, “TeraFlowSDN (TFS) Software Development Group (SDG)”, Available: https://tfs.etsi.org/

[45] The Linux Foundation, Available: https://www.linuxfoundation.org/

[46] The Linux Foundation, “Open Container Initiative”, Available: https://opencontainers.org/

[47] CNCF Harbor, Available: https://goharbor.io/

[48] GitLab, Available: https://gitlab.com/

[49] GitHub, Available: https://github.com/

[50] jFrog, Available: https://jfrog.com/

[51] 3GPP, "TR 28.801; Telecommunication management; Study on management and orchestration of network slicing for next generation network (Release 15)," 2018.

[52] TM Forum, "TMF 909A - Network as a Service (NaaS) API Component Suite Profile," 2019

[53] GSMA, “The Ecosystem for Open Gateway NaaS API Development,” 2023.

[54] GSMA, "Generic Network Slice Template v10.0," 2024. [Online]. Available: https://www.gsma.com/newsroom/wp-content/uploads//NG.116-v10.0.pdf

[55] Kroki [online] Available at: https://kroki.io/

[56] IETF, “The OAuth 2.0 Authorization Framework,” 2012.

[57] Bugzilla, Available: https://www.bugzilla.org/

[58] Flowable, Available: https://flowable.com/products/flowable-orchestrate/

[59] Google, “Blockly,” Available: https://developers.google.com/blockly

[60] Elastic Stack, Available: https://www.elastic.co/what-is/elk-stack

[61] IETF, “YANG - A Data Modelling Language for the Network Configuration Protocol (NETCONF),” 2010.

[62] ETSI, "GS NFV-SOL 005; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point," 2020.

[63] TMForum, “TMF628 Performance Management API v5.0.0”, 2024, Available: https://www.tmforum.org/oda/open-

apis/directory/performance-management-api-TMF628/v5.0

[64] Patras5G Swagger, Available: https://patras5g.eu/tmf-api/swagger-ui/index.html

[65] TMForum, "TMF702 Resource Activation Management API v4.0.1", 2020, Available: https://www.tmforum.org/resources/specification/tmf702-resource-activation-api-user-guide-v4-0-0/

[66] ETSI Context Information Management (CIM), NGSI-LD API, v1.7.1, June 2023, Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.07.01_60/gs_cim009v010701p.pdf

[67] ETSI, "ETSI GS NFV-SOL 005; Network Functions Virtualisation (NFV); Protocols and Data Models; RESTful Protocols Specification for the Os-Ma-nfvo Reference Point, v3.5.1 October. 2021, Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/03.05.01_60/gs_nfv-sol005v030501p.pdf

[68] Apache, "ActiveMQ: Flexible & Powerful Open Source Multi-Protocol Messaging", Available: https://kafka.apache.org/

[69] Oracle Corporation, MySQL, Available: https://www.mysql.com

[70] Elastic, Elasticsearch, Available: https://www.elastic.co/elasticsearch

[71] Model Context Protocol, Available: https://modelcontextprotocol.io/docs/getting-started/intro

[72] Open Container Initiative (OCI), Available: https://opencontainers.org/

[73] ETSI, "ETSI GS NFV-SOL 018 V5.2.1 (2025-05)", Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/018/05.02.01_60/gs_NFV-SOL018v050201p.pdf

[74] ETSI, "ETSI GS NFV-SOL 020 V5.2.1 (2025-05)", Available: https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/020/05.02.01_60/gs_NFV-SOL020v050201p.pdf

[75] IETF, "A Framework for Network Slices in Networks Built from IETF Technologies", RFC9543, Available: https://datatracker.ietf.org/doc/rfc9543/

[76] ETSI, "ETSI GS CIM 009 V1.9.1 (2025-07): Context Information Management (CIM); NGSI-LD API", Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.09.01_60/gs_cim009v010901p.pdf

[77] O-RAN Alliance, "O-RAN.WG1.O1-Interface.0-v04.00: O-RAN Operations and Maintenance Interface Specification," Technical Specification, ver. 4.0, 2024

7 APPENDIX A – CI/CD STACK SERVICES

The current Appendix provides the detailed descriptions of the infrastructure and services composing the CI/CD stack, namely for: User Management, Artefact Registry, CI/CD Server and Container Management.

7.1 CLOUD INFRASTRUCTURE

The initial central COP-PILOT deployments are in Falkenstein, Germany, and Helsinki, Finland. Figure 7-1 summarizes the Hetzner Cloud resources allocated to these locations.

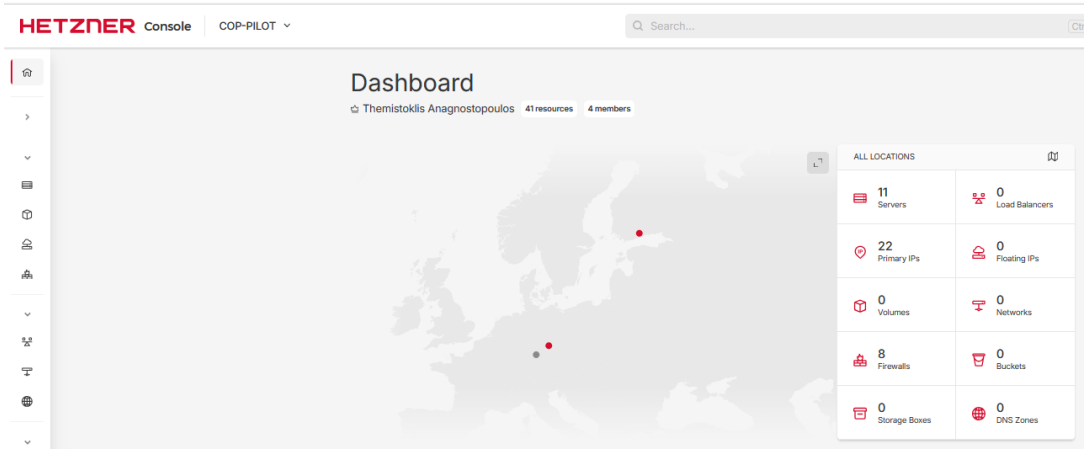

Figure 7-1 Summary of COP-PILOT resources and their locations in Hetzner Cloud.

Table 7-1 provides an overview of the centrally deployed VMs, their roles and associated HW resources. These resources can be up scaled depending on the demands. All VMs run Ubuntu Server 24.04 as their operating system.

Table 7-1 Central Management Domain resources used for the 1st COP-PILOT platform release.

VM name	Role in platform integration	vCPUs	RAM /storage
cop-pilot-bml-comoponents	Business Management Layer Components	16	32 GB / 320 GB
cop-pilot-ocm	K8s Cluster for testing	16	32 GB / 320 GB
cop-pilot-cicd	Runtime environment for CI/CD tools and automation services.	4	8 GB / 160 GB
cop-pilot-OS	OpenSlice instance for integration tests support.	16	32 GB / 320 GB
dev-node-0	Development and staging node for integration experiments.	4	8 GB / 160 GB

dev-node-1	Lightweight development node for auxiliary tests and support workloads.	2	4 GB / 40 GB
cop-pilot-bml	Business Management Portal runtime environment.	8	16 GB / 240 GB
cop-pilot-sif	SIF Edge Router	4	8 GB / 80 GB
cop-pilot-test-cluster	K8s Cluster for testing	16	32 GB / 320 GB
cop-pilot-eso	HypO Orchestrator	16	32 GB / 320 GB
vpn-server	OpenVPN Server	2	2 GB / 40 GB

7.2 USER MANAGEMENT

Keycloak is the tool that was selected, installed, and configured to act as the centralised user-management service of the COP-PILOT CI/CD Stack. Dedicated accounts were created for all platform developers. These are added to groups corresponding to different technical teams of the consortium. The Keycloak groups have different access permissions on the CI/CD Stack resources, following the RBAC model.

User accounts are used for user authentication across the multiple CI/CD Stack tools. When users try to log into any of these tools, they are redirected to the Keycloak log-in page to enter their credentials, as presented in Figure 7-2.

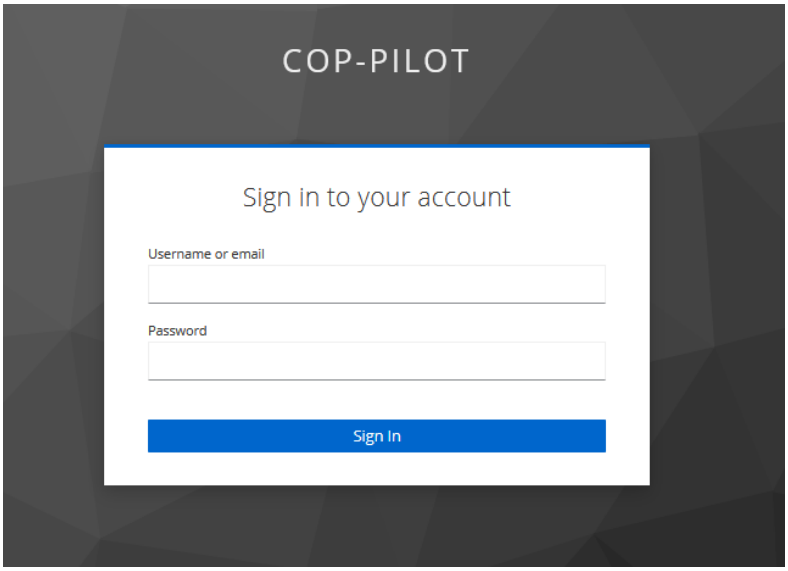

Figure 7-2 Keycloak sign-in page.

7.3 COP-PILOT GITHUB ORGANIZATION

COP-PILOT utilizes GitHub as its primary Source Code Management (SCM) and version control platform. By leveraging Git's distributed architecture, the project enables simultaneous development across multiple teams while maintaining a complete, traceable version history. To ensure project integrity and confidentiality, GitHub's advanced access controls are used to restrict sensitive code to authorized contributors. Additionally, the platform's integrated issue-tracking tools are employed to streamline bug reporting and collaborative project management.

The COP-PILOT GitHub organization can be accessed at: https://github.com/cop-pilot-eu

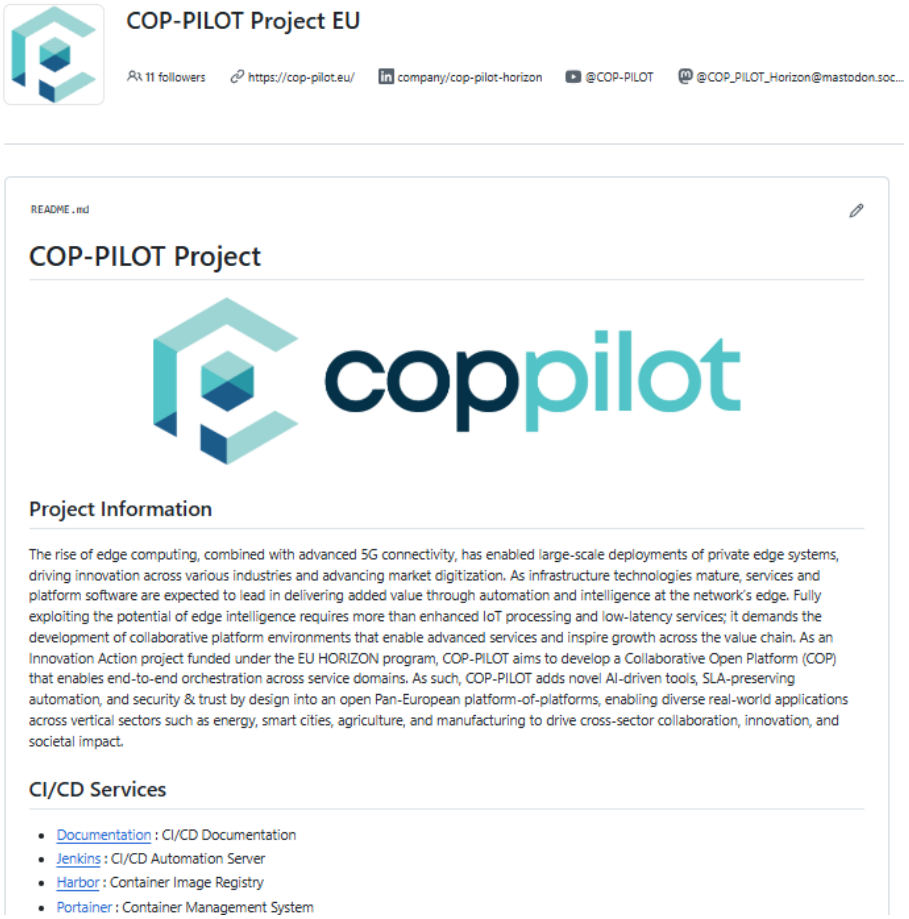

Figure 7-3 COP-PILOT GitHub Organization.

To contribute their source code, all technical partners must maintain a GitHub account and be enrolled in the COP-PILOT organization. For streamlined management, dedicated GitHub teams have been established for each partner. This structure facilitates granular permission management, targeted technical discussions, and efficient access to partner-specific repositories and workflows.

7.4 COP-PILOT ARTEFACT REGISTRY

Harbor (image) is an open-source container image registry developed as a Cloud Native Computing Foundation (CNCF) project. It stores and secures artefacts with policies and role-based access control, emphasising security, compliance, and performance. It supports storing, signing, scanning, and distributing artefacts. Harbor is integrated with the other tools of the COP-PILOT CI/CD stack.

Different Harbor projects (registries) are created per Cluster and per platform component and are mapped to the platform building blocks. Each project acts as a registry used to store one or more container repositories. Users with a COP-PILOT Keycloak account have image push/pull permissions to specific projects (only associated with their developments). Additionally, they can access the web GUI and create repositories under the projects they manage to host their container images.

Figure 7-4below presents a snapshot of the Harbor Web UI and the different projects created to host the artefacts. Through this UI, users can view the different tags (i.e., versions) of the docker images, pull a specific tag, upload new images, and delete tags that are not needed anymore. Moreover, they can initiate security scans on their stored images to detect vulnerabilities in the bundled code.

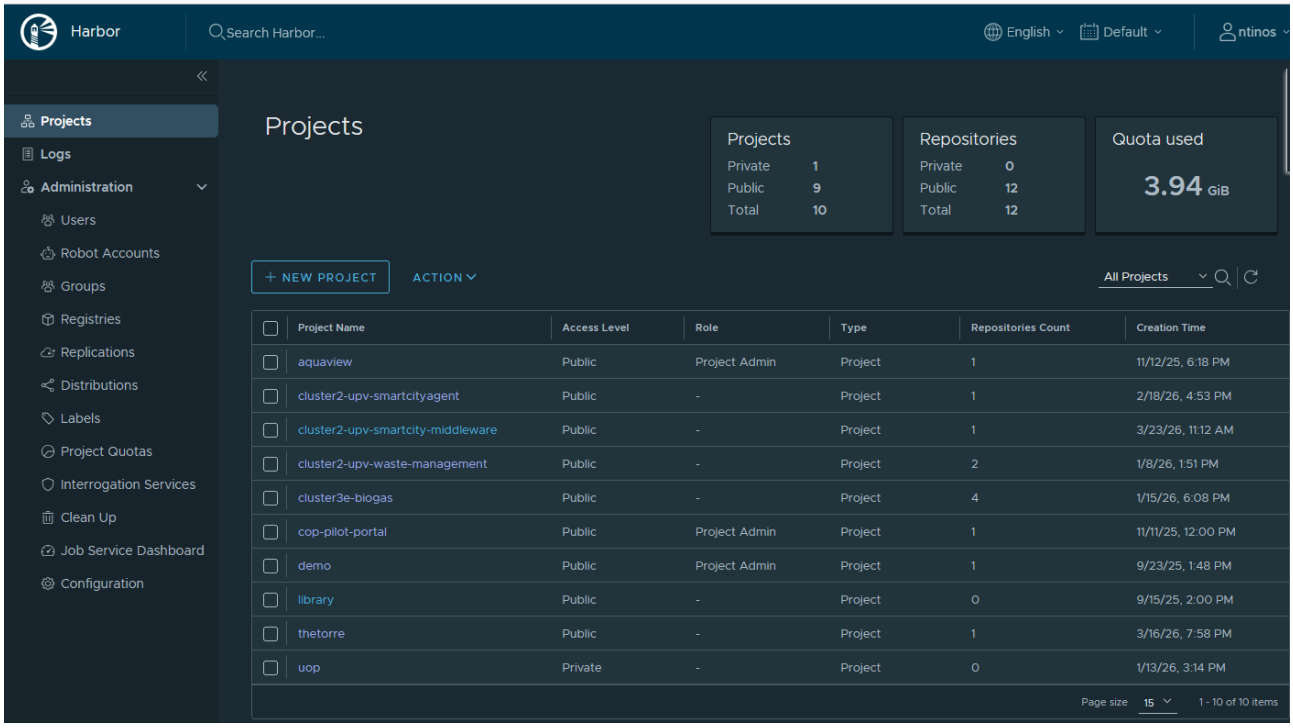

Figure 7-4 COP-PILOT projects hosted in the central Harbor container image registry.

A retention policy for keeping the latest ten artefacts per project per repository has been applied for all the projects. This policy is automatically executed every hour.

7.5 COP-PILOT CI/CD SERVER

Jenkins implements the role of the centralised CI/CD server. Jenkins is an automation server that facilitates CI/CD pipelines, which include a sequence of actions and tasks. These actions typically include pulling the most recent version of the code, building the Artefacts, checking for any issues, running unit and integration tests, and reporting any potential problems. After these checks, Jenkins releases the artefact for deployment and deploys the software component for further testing. Beyond the aforementioned role, Jenkins' automation capabilities are also leveraged to orchestrate operational/configuration processes among the Central management and the different Cluster domains of COP-PILOT, such as the automated peering between centrally located ESO and DOs of the different clusters. Figure 7-5 demonstrates the status view of an example pipeline that includes multiple events and actions called stages.

Co-funded by the European Union

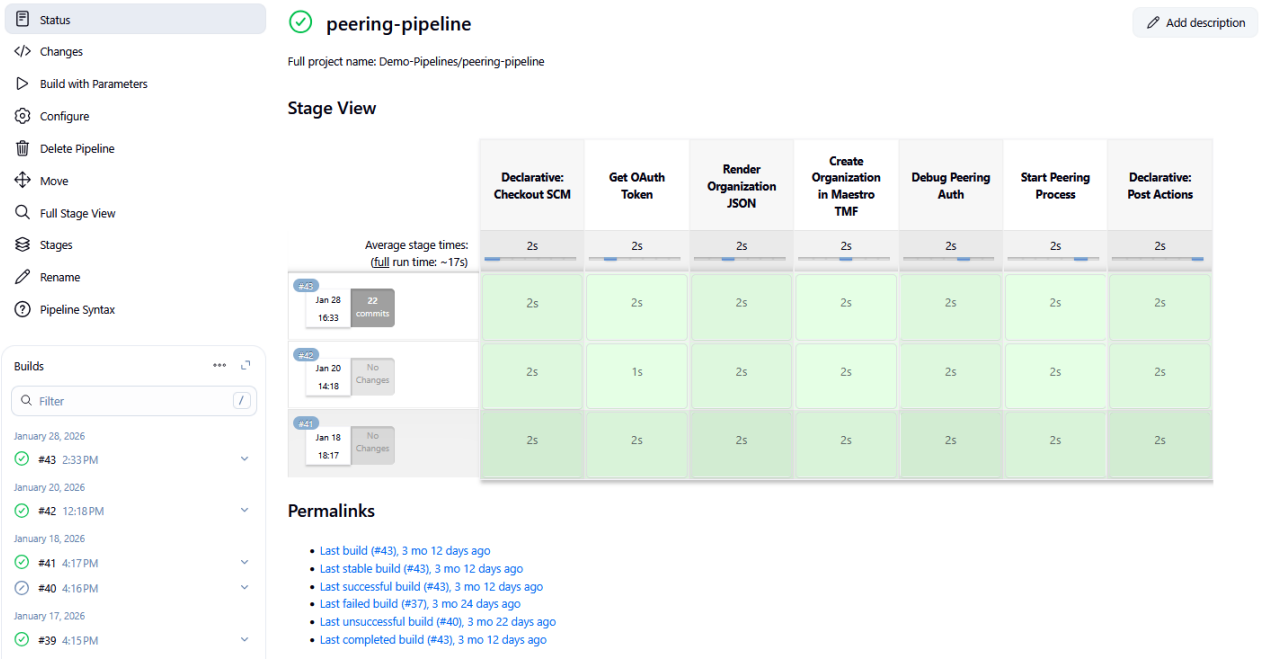

Figure 7-5 Jenkins pipeline status view.

The CI/CD pipelines can be triggered in two ways:

- **Automatically**: In this scenario, developers make a local clone or copy of the component's source code from a remote repository and perform updates. Once the changes are ready, the developers commit and push the updated code to the centralised GitHub repository. The Jenkins server is automatically notified of these incoming changes through a webhook mechanism which is enabled in GitHub for the corresponding Jenkins pipeline (example in Figure 7-6) and the execution of the pipeline is triggered.

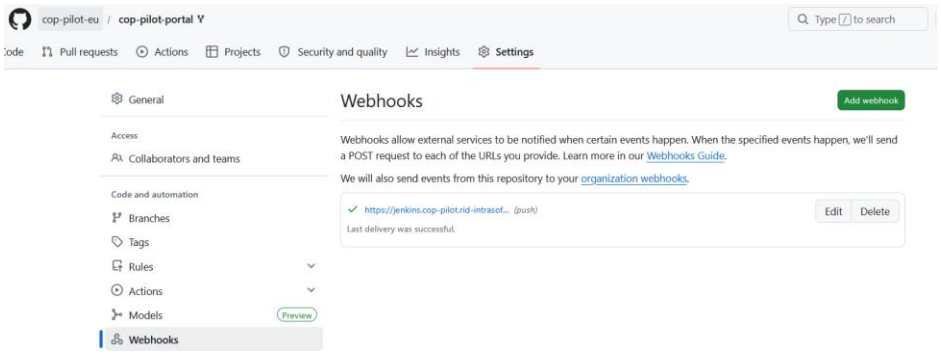

Figure 7-6 Webhook example for the Business Portal pipeline

- **Manually**: In this scenario, a built artefact is already pushed to the Harbor registry. Component developers or administrators log into the Jenkins Dashboard and manually initiate the execution of a pipeline that uses the said container image.

Jenkins provides an extensible toolset for modelling user-defined delivery pipelines "as code" using the Pipeline Domain-Specific Language (DSL) syntax in a configuration file called “Jenkinsfile”. The Jenkinsfile is committed and stored in the component’s GitHub repository, making the CI/CD process an integral part of the component.

Dedicated workspaces in Jenkins have been created to host various pipelines for each of the COP-PILOT Clusters and for the different platform components. These workspaces are accessible only by Cluster and/or component development teams, following the RBAC model applied to the whole CI/CD Stack. Figure below captures a snapshot of the available workspaces for COP-PILOT. Each workspace contains one or more pipelines associated with the internal modules of each Cluster and component.

S	W	Name ↓	Last Success	Last Failure	Last Duration
		Business Portal	N/A	N/A	N/A
		CL1 – Business Integration in Mining	N/A	N/A	N/A
		CL2 - Valencia Waste Chatbot	N/A	N/A	N/A
		CL2 – Valencia Smart City	N/A	N/A	N/A
		CL3A – AgriTech Transformation	N/A	N/A	N/A
		CL3E – Edge Intelligence for Energy Grid reliability	N/A	N/A	N/A
		CL4 – IoT-enhanced Agriculture, Recycling, & Manufacturing	N/A	N/A	N/A
		Demo-Pipelines	N/A	N/A	N/A
		SIF Layer	N/A	N/A	N/A

Figure 7-7 Jenkins workspaces following RBAC.

7.6 CONTAINER MANAGEMENT

Portainer is a lightweight management UI designed to simplify the management of environments. It offers a user-friendly web interface, making it accessible to both beginners and experienced professionals. With Portainer, users can manage individual clusters, deploy and configure applications, and handle containers, images, networks, and volumes with ease. It provides comprehensive monitoring tools and access to logs, facilitating performance monitoring and troubleshooting. Additionally, Portainer includes role-based access control for security in multi-user environments and supports deployment on various platforms, including Linux, Windows, and macOS. Extensible through support for extensions, Portainer allows users to customise and enhance its capabilities to meet specific needs. Overall, Portainer is a powerful tool for deploying, monitoring, and managing containerised applications.

A snapshot of the Portainer dashboard is shown in Figure below. Users can see the list of the hosts comprising the deployment environments. By clicking on a specific host, they can see and investigate the Docker resources running on that host.

Co-funded by the European Union

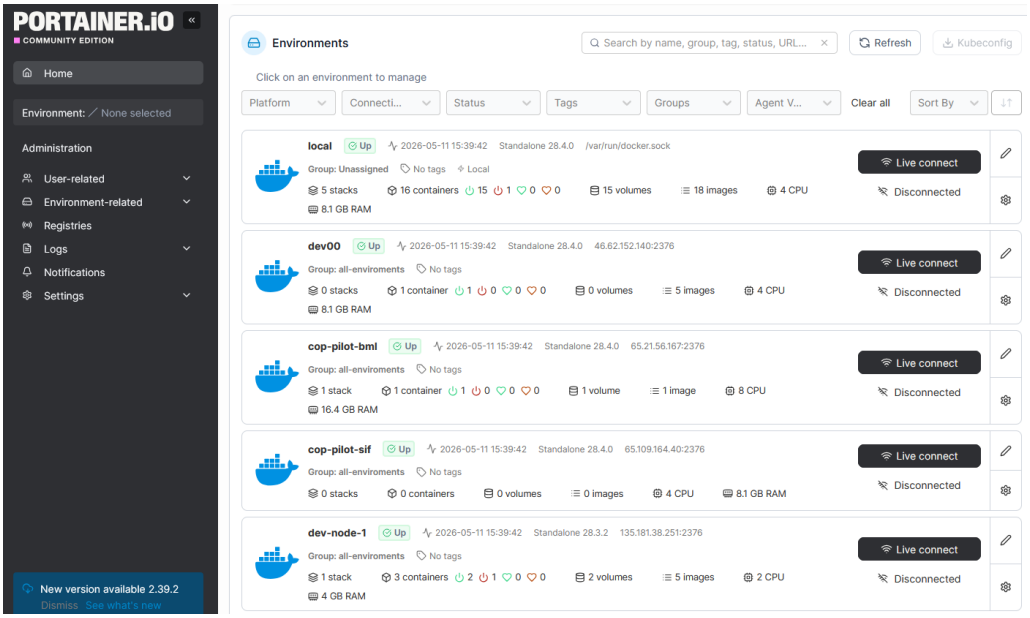

Figure 7-8 Snapshot of COP-PILOT VMs (hosts) in Portainer.

8 APPENDIX B – CI/CD PLATFORM WORKFLOWS

This appendix complements Section 4 with additional information on the actual set-up of the CI/CD platform. Specifically, it provides formal notation diagrams that visualize the way CI/CD pipelines attempt to automate crucial platform operations across the entire ecosystem in COP-PILOT.

8.1 PIPELINE FOR AUTOMATED DEPLOYMENT OF THE BUSINESS MANAGEMENT PORTAL

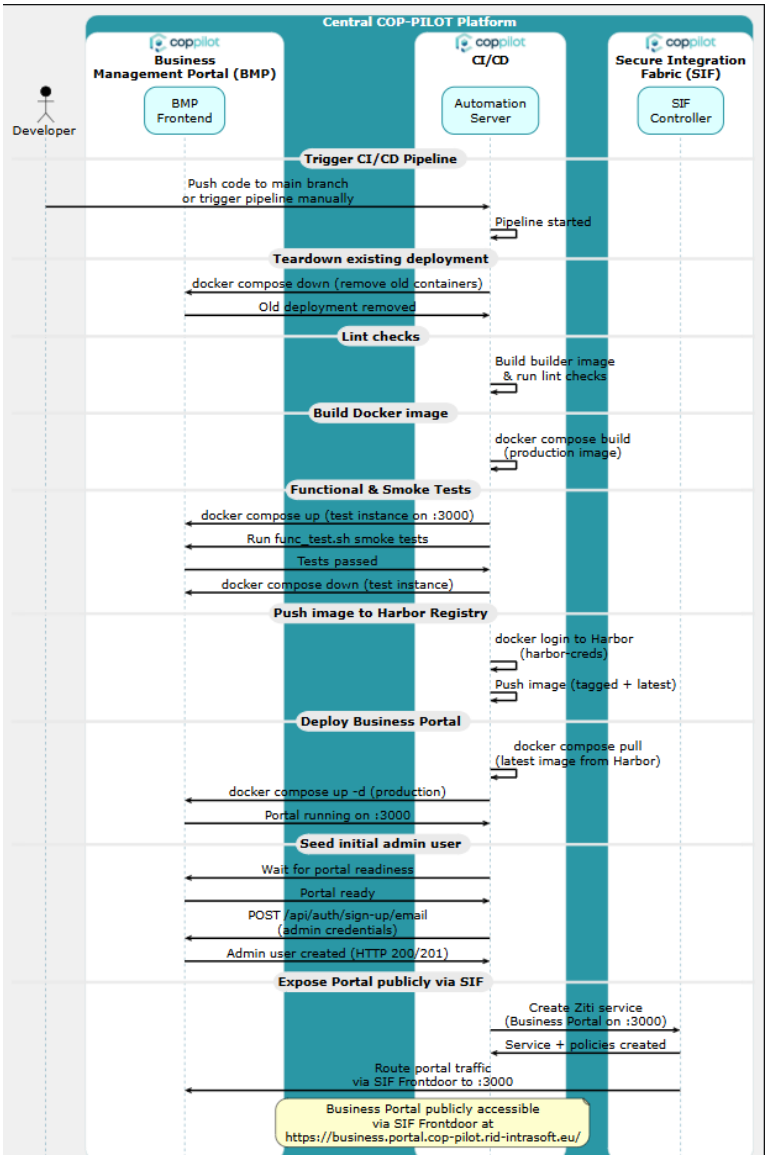

Figure 8-1 Workflow for Business Management Portal testing and deployment.

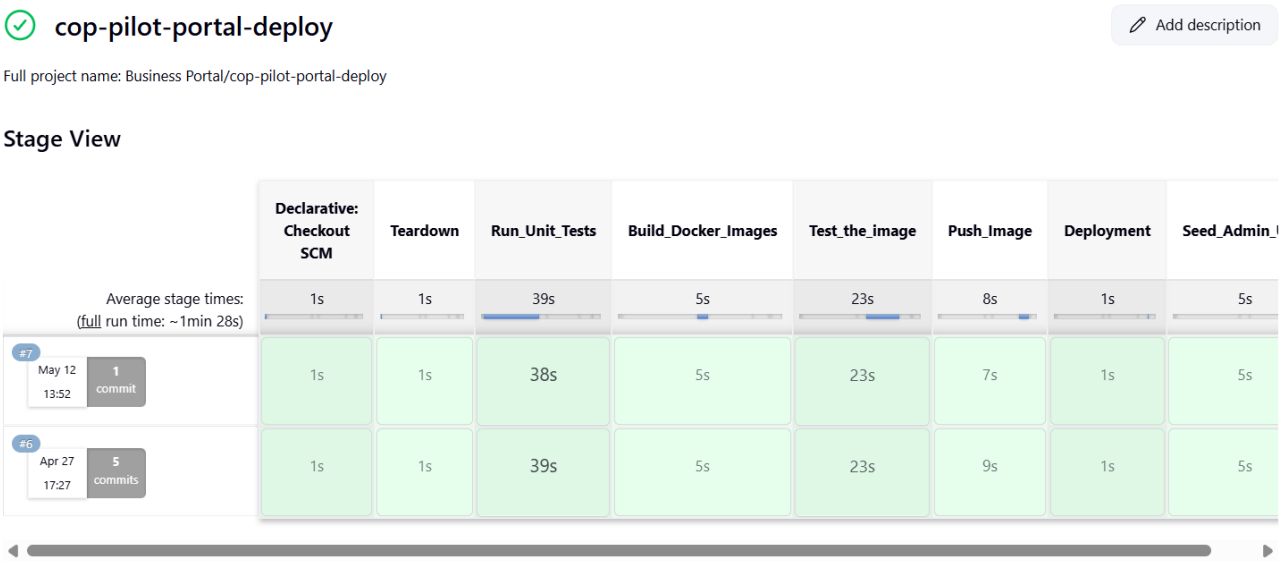

Figure 8-2 Jenkins pipeline for Business Management Portal

8.2 PIPELINE FOR AUTOMATED SIF CLIENT PROVISIONING AND INTEGRATION

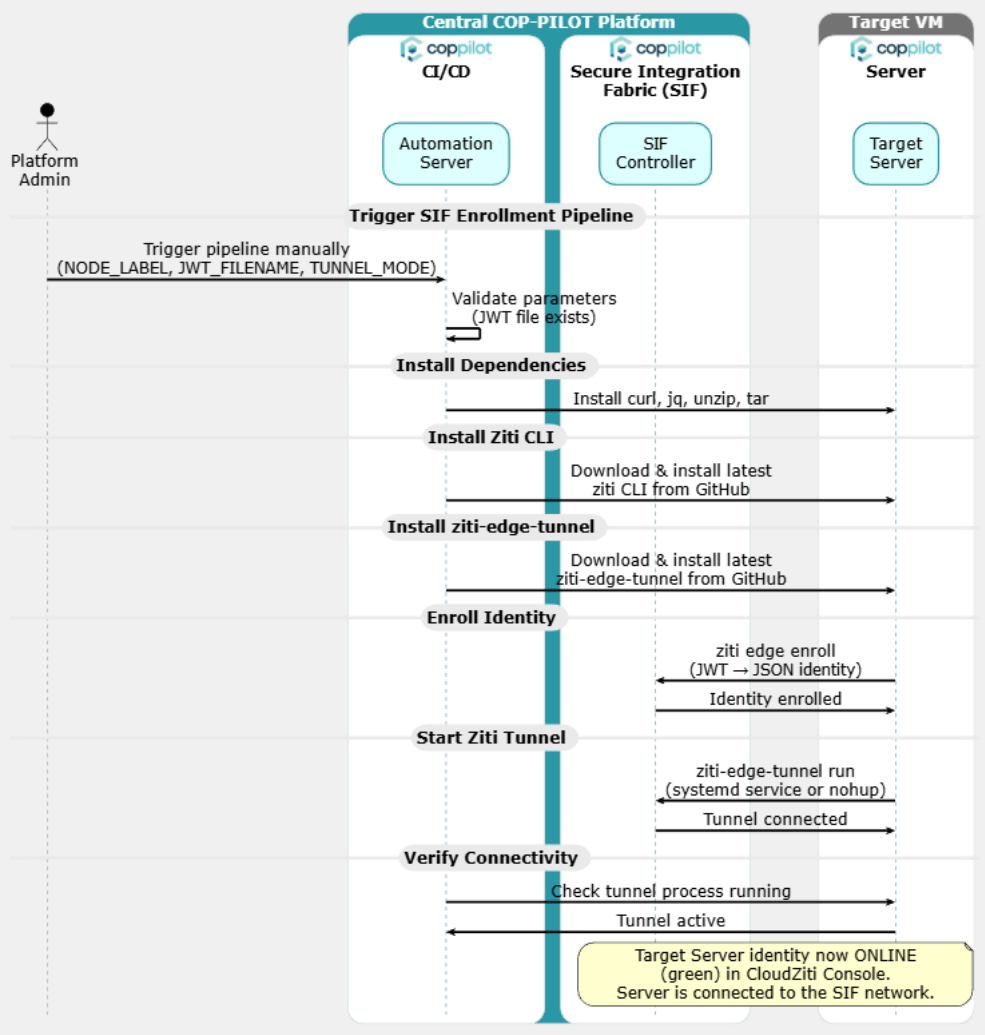

Figure 8-3 Workflow for SIF installation, identity enrolment and connectivity verification.

Full project name: SIF Layer/bml-components-ziti-enroll-tunnel

Stage View

	Declarative: Checkout SCM	Validate parameters	Install dependencies	Install Ziti CLI	Install ziti-edge-tunnel	Enroll identity	Start ziti-edge-tunnel	Verify tunnel connectivity	Declarative: Post Actions
Average stage times: (full run time: ~10s)	3s	765ms	1s	1s	657ms	539ms	247ms	170ms	49ms
#9 May 18 15:01 1 commit	872ms	849ms	673ms	2s	1s	1s	1s	707ms	36ms

Figure 8-4 Jenkins pipeline for SIF.

8.3 PIPELINE FOR AUTOMATED DEPLOYMENT OF THE DOMAIN ORCHESTRATOR

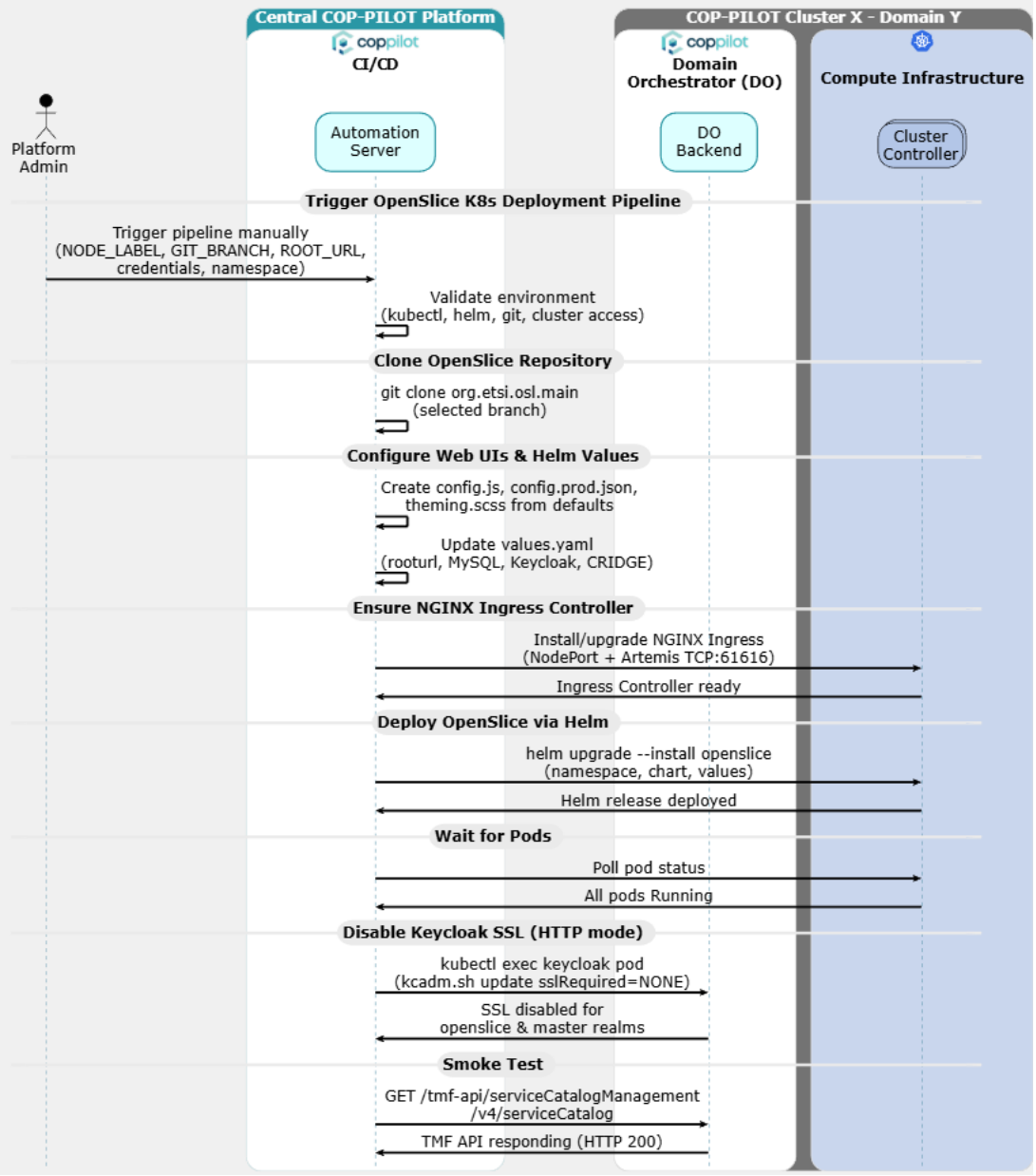

Figure 8-5 Workflow for deployment of the Domain Orchestrator.

8.4 PIPELINE FOR AUTOMATED SERVICE EXPOSURE OF THE DOMAIN ORCHESTRATOR VIA SIF

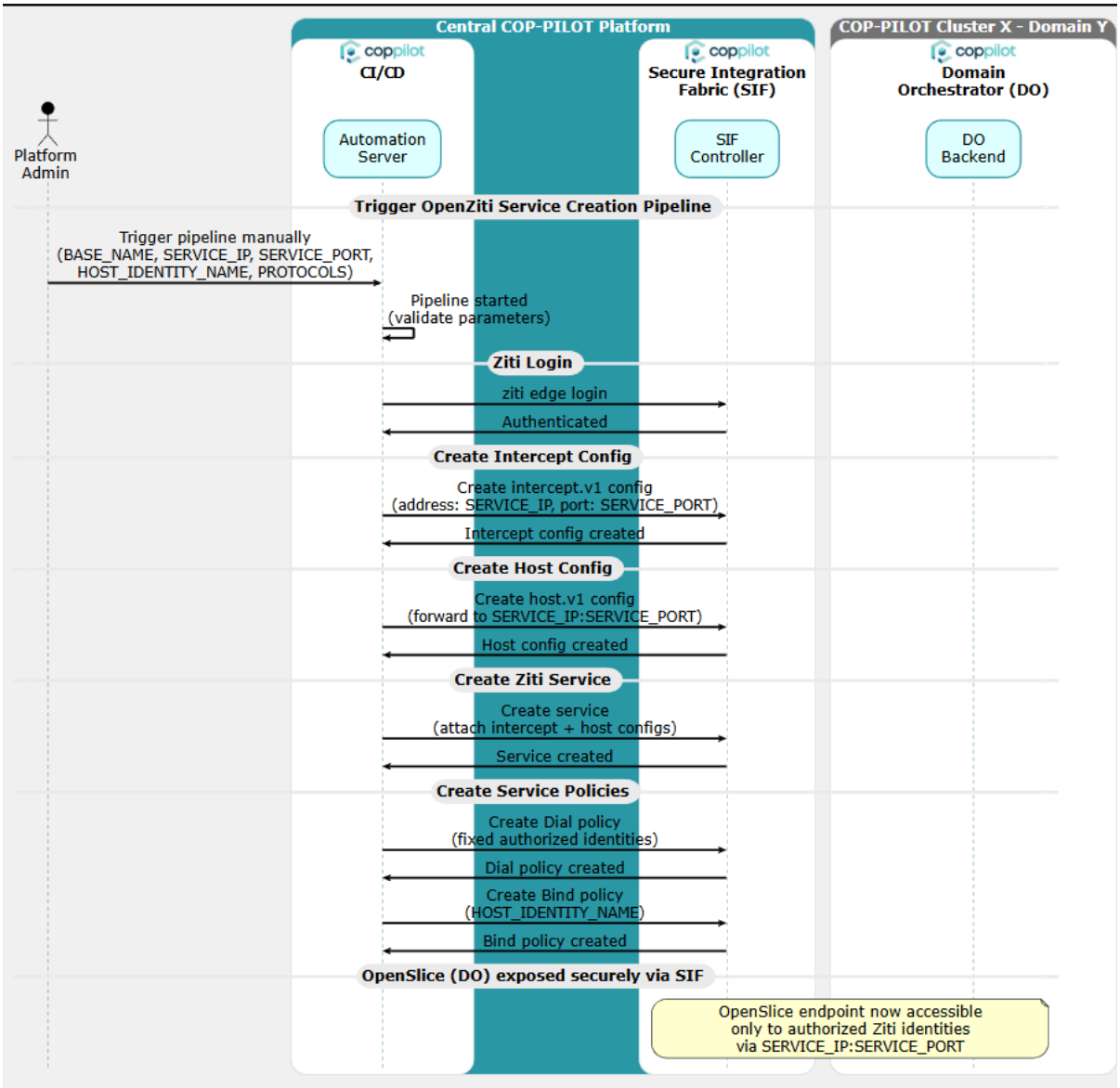

Figure 8-6 Workflow for secure service exposure of the Domain Orchestrator.

8.5 PIPELINE FOR AUTOMATED DEPLOYMENT OF THE DATA MANAGEMENT PLATFORM

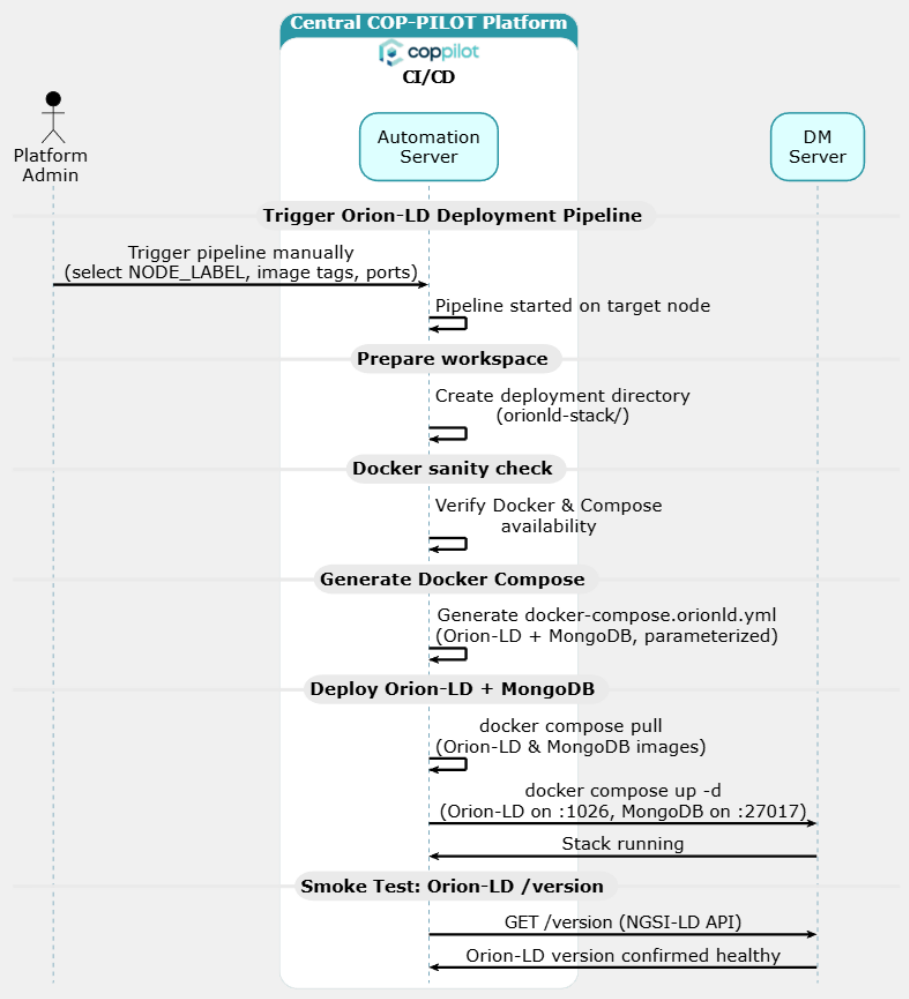

Figure 8-7 Workflow for deployment of the Data Management platform (Orion-LD).

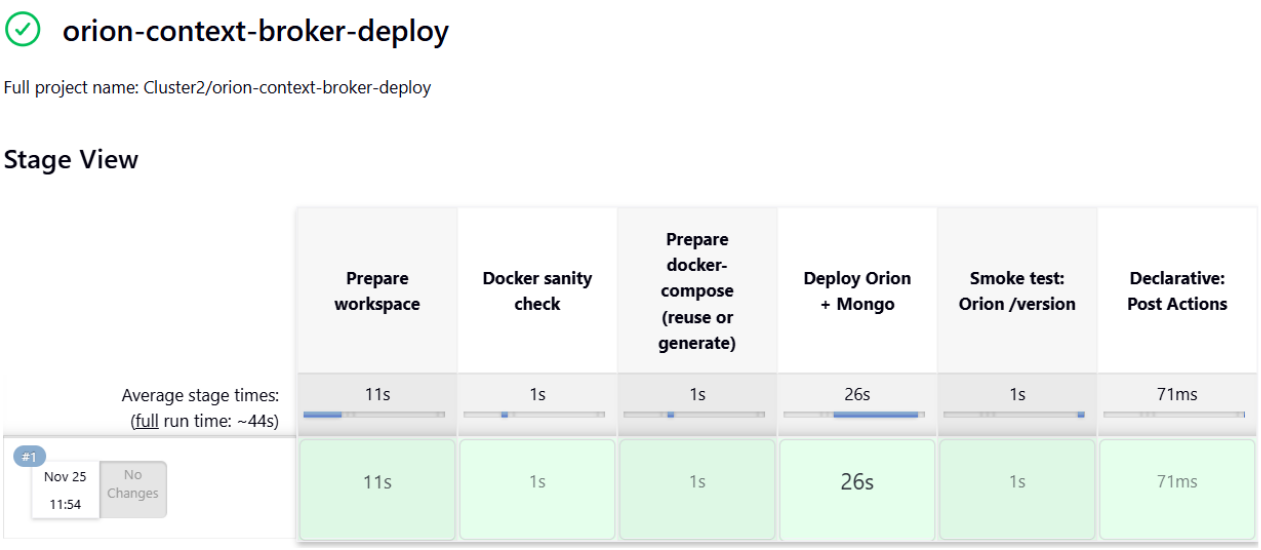

Figure 8-8 Reference Jenkins pipeline for Data Management Platform deployment in Cluster 2.

8.6 PIPELINE FOR AUTOMATED DO-ESO PEERING

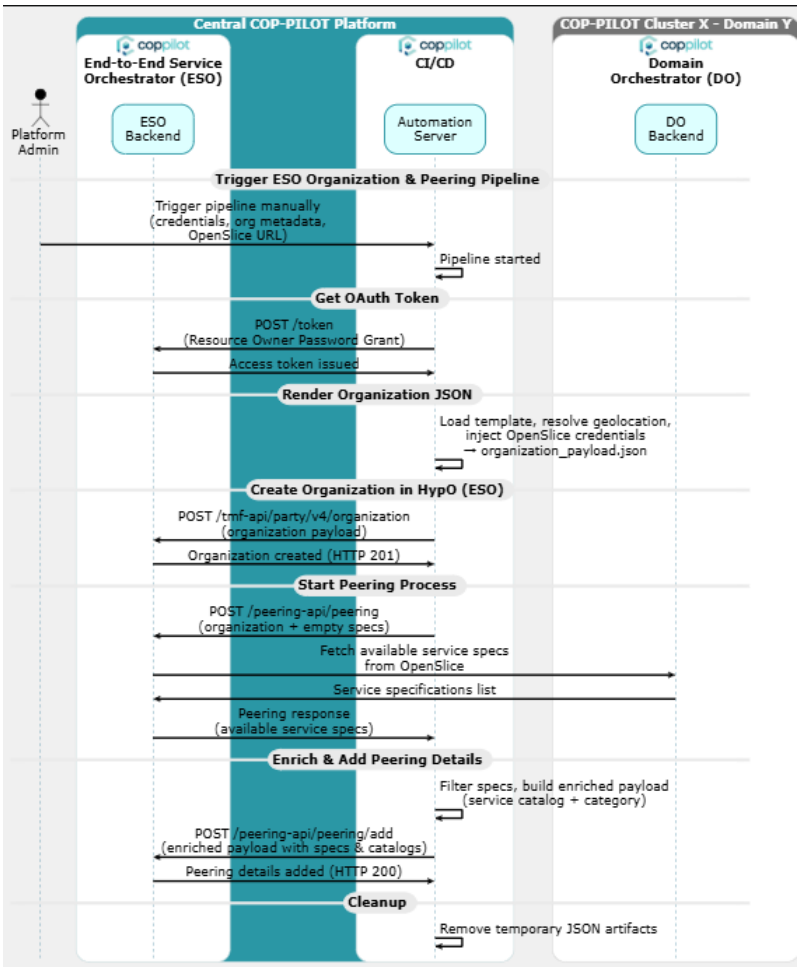

Figure 8-9 Workflow for DO-ESO peering.

8.7 PIPELINE FOR AUTOMATED DEPLOYMENT OF CLUSTER VERTICAL APPS UPON EVERY UPDATE

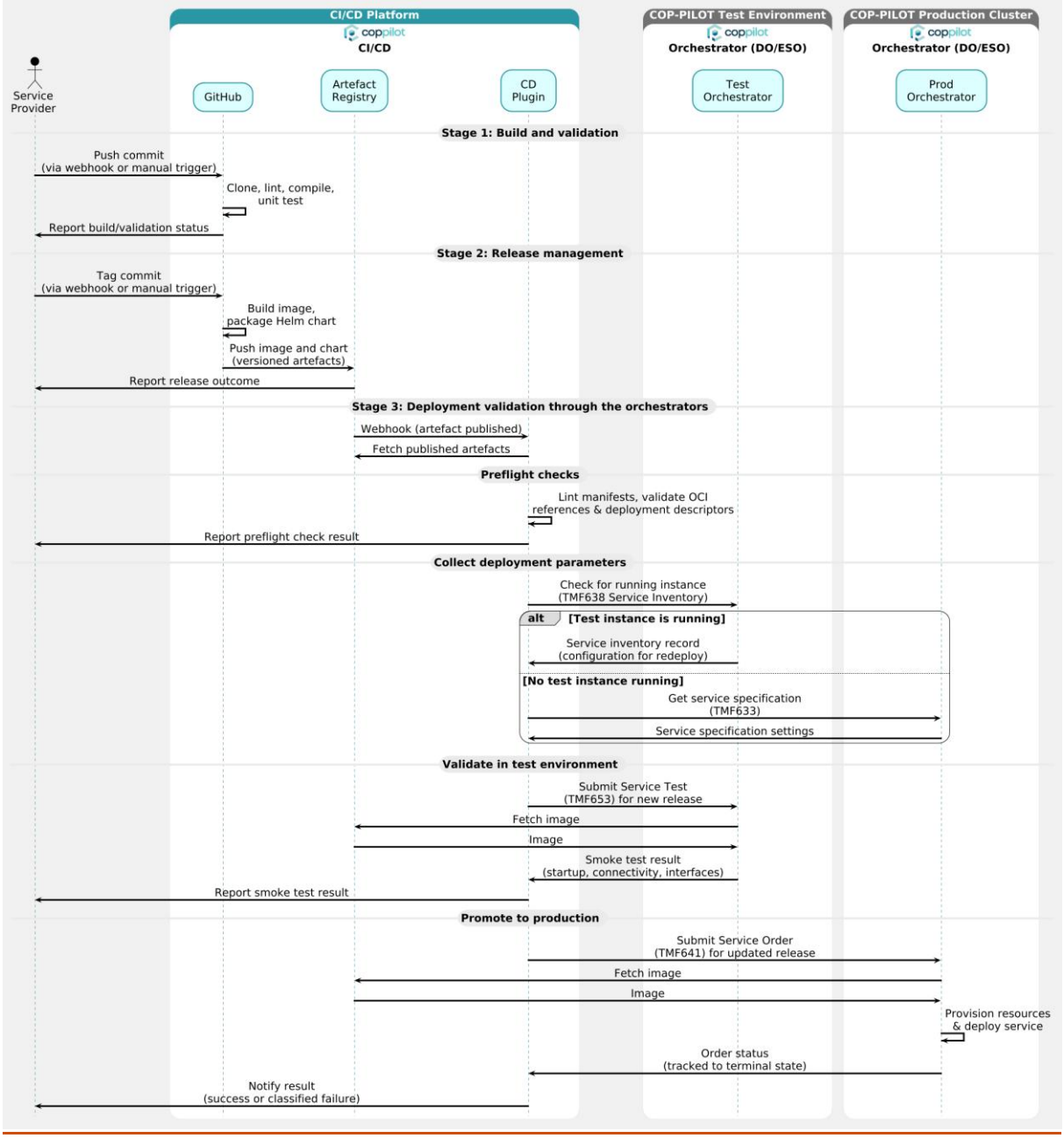

Figure 8-10 Workflow for the deployment of cluster Vertical apps upon every update.