An Artificial Intelligent Aided Unified Network for Secure Beyond 5G Long Term Evolution [GA: 101096456]

Deliverable 6.2

NANCY Integrated System – Initial Version

Programme: HORIZON-JU-SNS-2022-STREAM-A-01-06

Start Date: 01 January 2023

Duration: 36 Months

NANCY project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101096456.

Document Control Page

Deliverable Name	NANCY Integrated System – Initial Version
Deliverable Number	D6.2
Work Package	WP6 NANCY System Integration, Validation & Demonstration
Associated Tasks	Task 6.1 - Integration plan and facilities, Task 6.2 - Continuous integration
Dissemination Level	Public
Due Date	31 January 2025
Completion Date	28 January 2025
Submission Date	30 January 2025
Deliverable Lead Partner	Netcompany-Intrasoft
Deliverable Author(s)	Panos Matzakos (INTRA), Olga Segou (INTRA), Themistoklis Anagnostopoulos (INTRA), Stylianos Trevlakis (INNO), Lamprini Mitsiou (INNO), Vasileios Kouvakis (INNO), Theodoros Tsiftsis (INNO), Eirini Gkarnetidou (INNO), Ilias Theodoropoulos (8BELLS), Stratos Vamvourellis (8BELLS), Alvise Rigo (VOS), Anna Panagopoulou (VOS), Daniel Raho (VOS), Dimitris Manolopoulos (UBI), Georgios P. Katsikas (UBI), Miguel Catalan-Cid (i2CAT), Hatim Chergui (i2CAT), Giorgos-Nektarios Panayotidis (CERTH), Theofanis Xifilidis (CERTH), Dimitris Kavallieros (CERTH), Shih-Kai Chou (JSI), Carolina Fortuna (IJS), Jean-Paul Truong (TDIS), Ramon Sanchez-Iborra (UMU), Rodrigo Asensio (UMU), Andrea Wrona (CRAT), Emanuele De Santis (CRAT), Simone Gentile (CRAT), Valentina Becchetti (CRAT), Grigorios Kalogiannis (DRAXIS), Cristina Regueiro (TECNALIA), Abir Yasser Barakat (TEI), Marco Tambasco (TEI), Giuseppe Celozzi (TEI), Antonella Clavenna (ITL), Ioannis Makris (MINDS), Nikolaos Ntampakis (MINDS), Konstantinos Kyranou (SID), Georgios Michoulis (SID), Wenting Li, Francisco Javier deVicente Gutierrez (NEC), Panagiotis Sarigiannidis (UOWM), Thomas Lagkas (UOWM), Athanasios Liatifis (UOWM), Dimitrios Pliatsios (UOWM), Sotirios Tegos (UOWM)
Version	1.0

Document History

Version	Date	Change History	Author(s)	Organisation
0.1	05/11/2024	Initial version	Panos Matzakos, Olga Segou, Themistoklis Anagnostopoulos	INTRA
0.2	20/11/2024	Updates in identified integration points and integration matrix (Section 6)	Panos Matzakos	INTRA
0.3	02/12/2024	Initial inputs in sections 5, 6.2 and 6.3	Ramon Sanchez-Iborra, Rodrigo Asensio, Cristina Regueiro, Ioannis Makris, Nikolaos Ntampakis, Abir	UMU, TECN, MINDS, TEI

			Yasser Barakat, Marco Tambasco, Giuseppe Celozzi	
0.4	15/12/2024	Inputs in section 3	Panos Matzakos, Dimitris Manolopoulos, Hatim Chergui, Francisco Javier deVicente Gutierrez, Cristina Regueiro, Stylianos Trevlakis, Grigorios Kalogiannis, Stratos Vamvourellis	INTRA, UBI, i2CAT, NEC, TECN, INNO, DRAXIS, 8Bells
0.5	20/12/2024	Inputs in section 4	Panos Matzakos	INTRA
0.6	10/01/2025	Final inputs and updates in sections 5, 6.2, 6.3 and Annex	Panos Matzakos, Olga Segou, Themistoklis Anagnostopoulos, Stylianos Trevlakis, Lamprini Mitsiou, Vasileios Kouvakis, Theodoros Tsiftsis, Eirini Gkarnetidou, Ilias Theodoropoulos, Stratos Vamvourellis, Alvise Rigo, Anna Panagopoulou, Daniel Raho, Dimitris Manolopoulos, Georgios P. Katsikas, Miguel Catalan-Cid, Hatim Chergui, Giorgos-Nektarios Panayotidis, Theofanis Xifilidis, Dimitris Kavallieros, Shih-Kai Chou, Carolina Fortuna, Jean-Paul Truong, Ramon Sanchez-Iborra, Rodrigo Asensio, Andrea Wrona, Emanuele De Santis, Simone Gentile, Valentina Becchetti, Grigorios Kalogiannis, Cristina Regueiro, Abir	INTRA, INNO, 8Bells, VOS, UBI, i2CAT, CERTH, IJS, TDIS, UMU, CRAT, DRAXIS, TECN, TEI, ITL, MINDS, SID, NEC

			Yasser Barakat, Marco Tambasco, Giuseppe Celozzi, Antonella Clavenna, Ioannis Makris, Nikolaos Ntampakis, Konstantinos Kyranou, Georgios Michoulis, Wenting Li	
0.7	13/01/2025	Inputs in introduction and section 2	Panos Matzakos, Olga Segou	INTRA
0.8	16/01/2025	Inputs in Executive summary, section 6.4 and Conclusions; Updates in sections 2.2 and 2.3, 3.2, 6.2, 6.3, A.22, A.23	Panos Matzakos, Dimitris Manolopoulos, Wenting Li, Francisco Javier deVicente Gutierrez	INTRA, UBI, NEC
0.9	20/01/2025	Deliverable version ready for internal review	Panos Matzakos	INTRA
1.0	27/01/2025	Deliverable version after addressing internal review comments and quality checks	Panos Matzakos, Olga Segou, Hatim Chergui, Grigorios Kalogiannis, Shih-Kai Chou, Cristina Regueiro, Francisco Javier deVicente Gutierrez, Panagiotis Sarigiannidis, Thomas Lagkas, Athanasios Liatifis, Dimitrios Pliatsios, Sotirios Tegos	INTRA, i2CAT, DRAXIS, IJS, TECN, NEC, UOWM

Internal Review History

Name	Organisation	Date
Giorgos-Nektarios Panayotidis, Theofanis Xifilidis, Dimitris Kavallieros	CERTH	23 January 2025
Marco Tambasco	TEI	22 January 2025

Quality Manager Revision

Name	Organisation	Date
Anna Triantafyllou, Dimitrios Pliatsios	UOWM	28 January 2025

Table of Contents

Table of Contents 6
List of Figures 9
List of Tables 10
List of Acronyms 11
Executive summary 13
1. Introduction 14
  1.1. Purpose of the Document 14
  1.2. Relation to other Tasks and Deliverables 14
  1.3. Structure of the Deliverable 15
2. Integration Methodology 16
  2.1. Continuous Integration/ Continuous Delivery 16
  2.2. Integration Points Identification, Specification and Testing 17
  2.3. Communication Channels and Management of Integration Activities 18
3. Instantiation of the NANCY Reference Architecture 19
  3.1. Overview of NANCY Architecture 19
  3.2. Deployment of NANCY Architecture 20
    3.2.1. Central Management Domain 21
    3.2.2. Inter-operator Domain 24
    3.2.3. NANCY Testbeds and Demonstrators 27
  3.3. Reusability of NANCY Solution Towards new Deployment Environments 29
4. NANCY Integration Environment 32
  4.1. Hetzner Cloud Infrastructure 32
  4.2. CI/CD system 33
    4.2.1. Overview of CI/CD Services 33
    4.2.2. Github Version Control System 33
    4.2.3. Jenkins Continuous Integration Server 34
    4.2.4. Harbor Private Docker Container Registry 39
  4.3. NANCY Central Kubernetes Cluster 41
    4.3.1. High Level Architecture 41
    4.3.2. Control Plane Master Nodes 42
    4.3.3. Worker Nodes 43
    4.3.4. Kubernetes Networking 43
    4.3.5. Kubernetes Namespaces 44
    4.3.6. External Access to the Kubernetes Cluster 44

4.4. Security Features of the NANCY Integration Environment 46
4.4.1. Secure Communication over HTTPS 46
4.4.2. User Authentication and Role-based Access Control 47
4.4.3. VPN Configuration 48
4.4.4. Firewall Protection of CI/CD and Central Management Infrastructure 48
4.4.5. SSH-key-based Authentication to the Central NANCY Infrastructure 49
5. Functional Testing of NANCY Components 50
6. Integration of NANCY Components and Services 55
6.1. Integration Timeline 55
6.2. Integration Points Identification 57
6.3. Integration Tests 63
6.4. Next Integration Steps 65
7. Conclusions 66
Bibliography 67
A. Annex: Integration Points Specifications 69
A.1 MRAT-NCP – ID Management 69
A.2 ID-Management – Wallet 69
A.3 MRAT-NCP – SemCom 70
A.4 MRAT-NCP – Models 71
A.5 ID Management – VoSySMonitor 72
A.6 DAC – Marketplace 73
A.7 BC – Wallet 74
A.8 BC – Marketplace 75
A.9 Wallet – Marketplace 75
A.10 AI Virt. - VoSySMonitor 76
A.11 AI Virt. - SEMR 77
A.12 Marketplace – SPP 78
A.13 Models - SEMR 79
A.14 Models - Elasticity 80
A.15 Models – TFS 81
A.16 SEMR – AINQM 81
A.17 SEMR – FL-IDS 82
A.18 PQC Sign – PQC SecCom 82
A.19 AINQM - NIF 83
A.20 AINQM - XAI 83
A.21 XAI – FL-IDS 84

A.22 Wallet – BSS 85
A.23 Wallet – Maestro 86

List of Figures

Figure 1 Relation of D6.2 to other NANCY tasks. 15
Figure 2 NANCY CI/CD infrastructure and execution environments........ 17
Figure 3 Functional and Deployment view of NANCY reference architecture........ 19
Figure 4 NANCY platform operation domains and deployment of the NANCY architecture........ 21
Figure 5 High-level view of integration among i2CAT's Slice Manager and SEMR hosted at IJS Kubernetes cluster 24
Figure 6 Service selection inside the offloading and caching workflow 25
Figure 7 Operators using their wallets to list services in the Marketplace........ 26
Figure 8 NANCY project and VMs allocation in Hetzner cloud........ 33
Figure 9 Demo (example) project repository under NANCY Github Organisation........ 34
Figure 10 NANCY component-specific workspaces in Jenkins CI server 35
Figure 11 Select Pipeline option........ 36
Figure 12 Jenkins pipeline creation: Link with specific Github repository........ 36
Figure 13 Retrieve GitHub repository's URL........ 37
Figure 14 Selection of Github hook trigger 37
Figure 15 Configuring consecutive Jenkins jobs........ 38
Figure 16 Jenkins pipeline creation: Further configurations for connecting to the target Github repository 38
Figure 17 Branches to Build........ 39
Figure 18 Wedhooks........ 39
Figure 19 NANCY component-specific projects in Harbor private registry........ 40
Figure 20 NANCY central Kubernetes cluster architecture 42
Figure 21 Instances of Kube-system components in central NANCY Kubernetes cluster........ 42
Figure 22 Setup for external access of the deployed services in the Kubernetes cluster........ 44
Figure 23 Ingress routing rules enabling external access to deployed services in the Kubernetes cluster........ 45
Figure 24 Example of Ingress resource configuration for name based virtual hosting 46
Figure 25 SSO access to NANCY CI/CD services via Keycloak........ 47
Figure 26 Access to central NANCY VPN 48
Figure 27 Hetzner firewalls to restrict access to Hetzner VMs 49

List of Tables

Table 1 NANCY interfaces and the related NANCY architectural components 19
Table 2 Listing of NANCY (WP2-WP5) components and their deployment locations 28
Table 3: NANCY functional testing 50
Table 4 NANCY integration timeline 55
Table 5 NANCY integration matrix 58
Table 6 Integration points specification summary 60
Table 7: NANCY integration testing 63

List of Acronyms

Acronym	Explanation
5G	5th Generation
AI	Artificial Intelligence
AI Virt	AI Virtualizer
API	Application Programming Interface
B5G	Beyond 5G
BSS	Business Support System
B-RAN	Blockchain RAN
CA	Certificate Authority
CI/CD	Continuous Integration/ Continuous Delivery
CNI	Container Network Interface
CSV	Comma-separated values
DAC	Digital Agreement Creator
DevOps	Development and Operations
DID	Decentralised Identifier
DSM	Design Structure Matrix
DT	Digital Twin
ETSI	European Telecommunications Standards Institute
FL	Federated Learning
FQDN	Fully Qualified Domain Name
GRPC	gRPC Remote Procedure Call
GSMA	GSM Association
GUI	Graphical User Interface
HA	Highly Available
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
K8s-aaS	Kubernetes-as-a-Service
KPI	Key Performance Indicator
LaaS	Localisation-as-a-Service
LLM	Large Language Model
MEC	Multi-Access Edge Computing
ML	Machine Learning
MLOps	Machine Learning Operations
NBI	Northbound Interface
NFVO	Network Functions Virtualisation Orchestrator
NI	NANCY Interface
NIS	NANCY Interface Set
NSaaS	Network Slice as a Service
OP-TEE	Open Trusted Execution Environment
OSS	Operations Support System
P2P	Point-to-Point
PQC	Post-Quantum Cryptography
RAN	Radio Access Network
RIC	RAN Intelligent Controller
RBAC	Role-Based Access Control
SCM	Source Code Management
SEMR	Self-Evolving Model Repository
SLA	Service Level Agreement

SM	Slice Manager
SO	Service Orchestrator
SP	Smart Pricing
SSI	Self-Sovereign Identity
SSL	Secure Sockets Layer
SSO	Single-Sign-On
TEE	Trusted Execution Environment
TLS	Transport Layer Security
UE	User Equipment
V2X	Vehicle-to-Everything
VNF	Virtual Network Function
VPN	Virtual Private Network
XAI	Explainable AI
YAML	YAML Ain't Markup Language

Executive summary

This deliverable presents in detail the intermediate results of WP6 tasks: *T6.1 - Integration plan and facilities* and *T6.2 - Continuous integration* associated with NANCY integration activities, which aim to incorporate the outcomes of the development tasks (WP2-WP5) into a unified NANCY platform. Building on top of the initial outcomes of T6.1 reported in [1], relative to the definition of the deployment view of NANCY architecture (initially produced in [2]), the current deliverable reports in detail on the association of the NANCY components developed in WP2-WP5 with the NANCY platform's deployment domains and interfaces defined in [1]. Following a well-defined integration methodology, it identifies and specifies the bilateral integration points of the platform as pairs of NANCY components which aim to interact with each other in the context of NANCY use cases, linking them to the NANCY platform interfaces. It describes in detail the first stage of NANCY functional and integration tests, performed to validate the intended functionalities per component and interoperability across the integration points, respectively. Furthermore, D6.2 elaborates on the integration framework guiding the production of the NANCY platform. It encompasses both methodological aspects and technical DevOps tools and best practices, aimed at standardizing, facilitating, and ultimately accelerating the software development and integration cycle.

An overview of the NANCY architecture and its deployment view produced in [2] and [1] is first presented, detailing the deployment domains defined in the context of NANCY: Central Management, Inter-Operator and NANCY testbeds/demonstrators, as well as their respective components. In this context, the associated deployment domain and physical deployment location for each NANCY component are provided. Reusability aspects of the NANCY platform across different deployment environments are also covered.

Then, the NANCY integration environment is presented in detail emphasizing different Continuous Integration/ Continuous Delivery (CI/CD) services that have been put in place, as well as the central Kubernetes cluster established for NANCY, with a focus on the methods used to isolate execution environments (e.g., development, staging and operational) through namespaces. Finally, an overview of the security features of the integration environment is provided.

The functional tests for each NANCY component are subsequently described, outlining the objectives of the various tests per component. This description is followed by the detailed specification of the integration plan, including the identification and comprehensive description of the NANCY bilateral integration points, along with the corresponding planned integration tests, as well as the next steps of the integration activities.

1. Introduction

1.1. Purpose of the Document

This document presents the preliminary version of the NANCY integrated system. Specifically, NANCY designs, implements, tests and validates a secure and intelligent architecture for Beyond 5G (B5G) wireless networks. Through the use of cutting-edge research in Artificial Intelligence, Blockchain, Orchestration etc. the project enables secure and intelligent resource management, flexible networking and orchestration. In terms of networking, novel architectures, namely point-to-point (P2P) connectivity for device-to-device communication, mesh networking, and relay-based communications, as well as protocols for medium access, mobility management, and resource allocation are designed and tested within the lifecycle of the project.

The project takes an ambitious approach to the provision of the necessary Beyond 5G functionalities. It brings together a multitude of disparate and complex technologies to create its core platform, on top of the three unique experimental testbeds. Multiple technical challenges are involved in deploying such a variety of complex functionalities; the project thus requires the definition of a methodical approach of integration, testing and validation. This document discusses the process utilised by the NANCY consortium towards achieving the first integrated system, up to M25. It covers the Continuous Integration methodology, the instantiation of the core platform, the integration infrastructure leveraged by the project and provides detailed information on the main integration points and the status of integration activities at M25.

1.2. Relation to other Tasks and Deliverables

The Integration task received inputs from the key technical work packages, namely WP2 “Usage Scenario and B-RAN Modelling, Network Requirements and Performance assessment” for requirements and modelling of B-RAN and Network Information Framework, WP3 “NANCY Architecture and Orchestration” for the core artificial intelligence and orchestration components, WP4 “Dynamic Resource Management and Smart Pricing” for the components regarding the computational off-loading, caching, resource elasticity, cell-free cooperative access, smart pricing and beyond-Shannon performance, and WP5 “Security, Privacy and Trust Mechanisms” for the quantum key distribution, blockchain-based security and privacy, self-healing/self-recovery and explainable AI components. The results of the integration and evaluation process that takes place in WP6 and is in part reflected in this deliverable (covering developments up to M25), are then provided as inputs to the technical work packages which in turn can refine and optimize the implementation. Specifically, this document covers the definition of integration points, the integration of the various services created in the span of the project and provides an overview of the first functional and bilateral integration tests.

This work is also strongly related to the work already performed in WP6 and reported in deliverable D6.1 “B-RAN and 5G End-to-End Facility Setup” [1] on the infrastructure fabric created for NANCY. D6.1 formulates a clear plan for the deployment of the core architectural components of NANCY, as well as the testbed platforms. This document presents technical work performed in Task 6.1 (B-RAN and 5G End-to-End Facilities Setup, Operations and Maintenance) and T6.2 (Continuous Integration) and outputs the integration timeline, test results and next integration steps to the technical work packages, the interoperability and joint-optimisation task (T6.3) and pilot tasks (T6.4-T6.9), as seen in Figure 1.

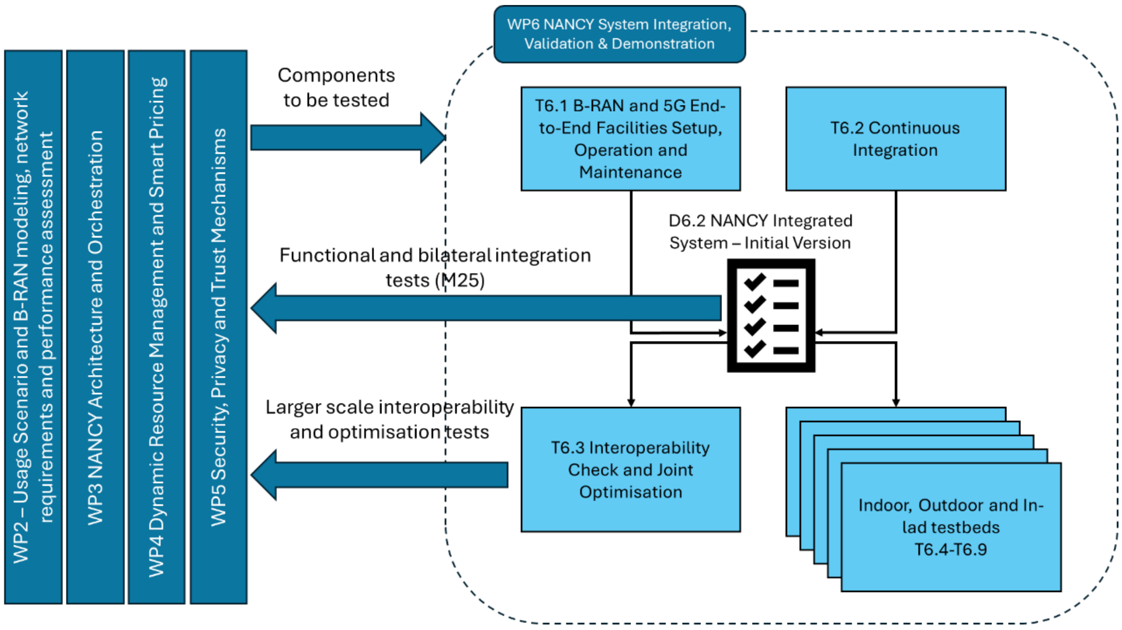

WP2 – Usage Scenario and B-RAN modeling, network requirements and performance assessment
WP3 NANCY Architecture and Orchestration
WP4 Dynamic Resource Management and Smart Pricing
WP5 Security, Privacy and Trust Mechanisms
Components to be tested
Functional and bilateral integration tests (M25)
Larger scale interoperability and optimisation tests
WP6 NANCY System Integration, Validation & Demonstration
T6.1 B-RAN and 5G End-to-End Facilities Setup, Operation and Maintenance
T6.2 Continuous Integration
D6.2 NANCY Integrated System – Initial Version
T6.3 Interoperability Check and Joint Optimisation
Indoor, Outdoor and In-lad testbeds T6.4-T6.9

Figure 1 Relation of D6.2 to other NANCY tasks.

1.3. Structure of the Deliverable

This document is structured as follows:

- **Chapter 1 – Introduction (current chapter)** defines the purpose and structure of this document and contextualises the integration work with respect to the project's technical work.
- **Chapter 2 – Integration Methodology** introduces the overall process followed by NANCY to deal with challenges in integrating the NANCY subsystems and achieving end-to-end functionalities. This chapter introduces the CI/CD platform used by NANCY and the process of integration points identification between the related subsystems.
- **Chapter 3 – Instantiation of the NANCY Reference Architecture** provides an overview of the NANCY architecture and delves into the deployment and reusability of its components.
- **Chapter 4 – NANCY Integration Environment** provides an analytical description of the CI/CD environment, the deployed services and the Version Control. Furthermore, it describes the NANCY Central Kubernetes cluster. Finally, it touches upon the security features that are utilised by the NANCY Integration Environment.
- **Chapter 5 – Functional testing of NANCY Components** provides an overview of the functional tests defined and executed by the NANCY partners, per NANCY component.
- **Chapter 6 – Integration of NANCY Components and Services** provides a detailed view of the integration process, starting with a timeline of activities. It also provides a listing of integration tests per integration point and discusses the status of the integration activities in terms of completion and future work.
- **Chapter 7 – Conclusions** concludes this document and provides a brief overview of the results and lessons learnt.
- **Annex Sections:** The full specifications of the NANCY integration points are included in Annex.

2. Integration Methodology

The objective of the NANCY integration framework is to incorporate the different NANCY components and services developed in the context of WP2-WP5 into a unified platform that bridges the Edge and Cloud environments and underlying technologies and instantiates the NANCY architecture in a versatile and secure manner. The challenges associated with integrating software from diverse technology pillars (AI, Blockchain, MEC and Orchestration) while aligning with 5G/6G standards and KPIs have been discussed in [2], and so have the technology-specific plans to overcome them. In the current section, we describe the basic processes of the NANCY integration methodology, which were also introduced in [2]. The methodology is based on breaking down the project into smaller, manageable pieces and employing a structured approach to testing and agile methodologies. This is to ensure that the process can be efficiently streamlined, while exploiting DevOps tools and best practices to facilitate, enhance and eventually accelerate the software development and integration cycle.

2.1. Continuous Integration/ Continuous Delivery

To manage the complexity of the NANCY platform in integrating the aforementioned technology components, a Continuous Integration/Continuous Delivery (CI/CD) strategy is employed. This approach, is in accordance with DevOps best practices and it fosters smooth collaboration among development teams, allowing for the efficient streamlining of code releases, the incorporation of updates, and the automation of testing and integration processes.

The basic principles of Continuous Integration and Continuous Delivery have already been described in [1]. In the context of NANCY, CI is related to the usage of CI/CD services (described in detail in section 4.2) connected to a central development and testing/staging environment (central Kubernetes cluster). The objective of this setup is to provide the developers/testers of the different NANCY components with a common environment that can be used to frequently and automatically test their code updates, covering different tests which range from unit and functional to integration with other components deployed at the same environment. This setup is addressed to the NANCY components which are containerized and thus can easily be deployed and tested in a Kubernetes cluster.

CD is related to centrally triggered deployment and validation of NANCY containerized components in an operational context. This context can either refer to direct deployment at the different execution sites (Kubernetes clusters) of the NANCY testbeds and demonstrators, or deployment at the NANCY central management domain (see section 3.2) which is also hosted in the central Kubernetes cluster, under dedicated and isolated resources. Furthermore, in the context of CD as described in [1] and further explained in section 3.2.1, the CI/CD services will also interoperate with the NANCY Service and Resource Orchestrator to manage deployment service orders towards the distinct Kubernetes clusters related to the target NANCY demonstrators/testbeds efficiently. Typically, CD takes place after any development updates at individual component-or integration-level have been validated in the CI phase.

The NANCY CI/CD system and central Kubernetes cluster are built on top of Linux VMs acquired from Hetzner public cloud provider [3] (Figure 2), as further explained in section 4.1. In this context, the specific hardware requirements for the various NANCY software components were initially assessed to ensure the infrastructure could support them within the CI/CD services and the central NANCY Kubernetes cluster. It is worth mentioning that the development and testing setup of NANCY aims to provide a deployment environment that is similar to the different NANCY testbeds and demonstrators.

In this context, the selection of NANCY software containerization (where possible) and Kubernetes container orchestration platform across the different environments of the project aims to facilitate the integration efforts.

Two technical workshops were conducted to acquaint the technical partners with the integration environment and framework created by the INTRA team for the project. Comprehensive documentation and sample projects were provided to assist the technical partners in developing their own deployment and testing pipelines for their containerized solutions on the NANCY Kubernetes cluster.

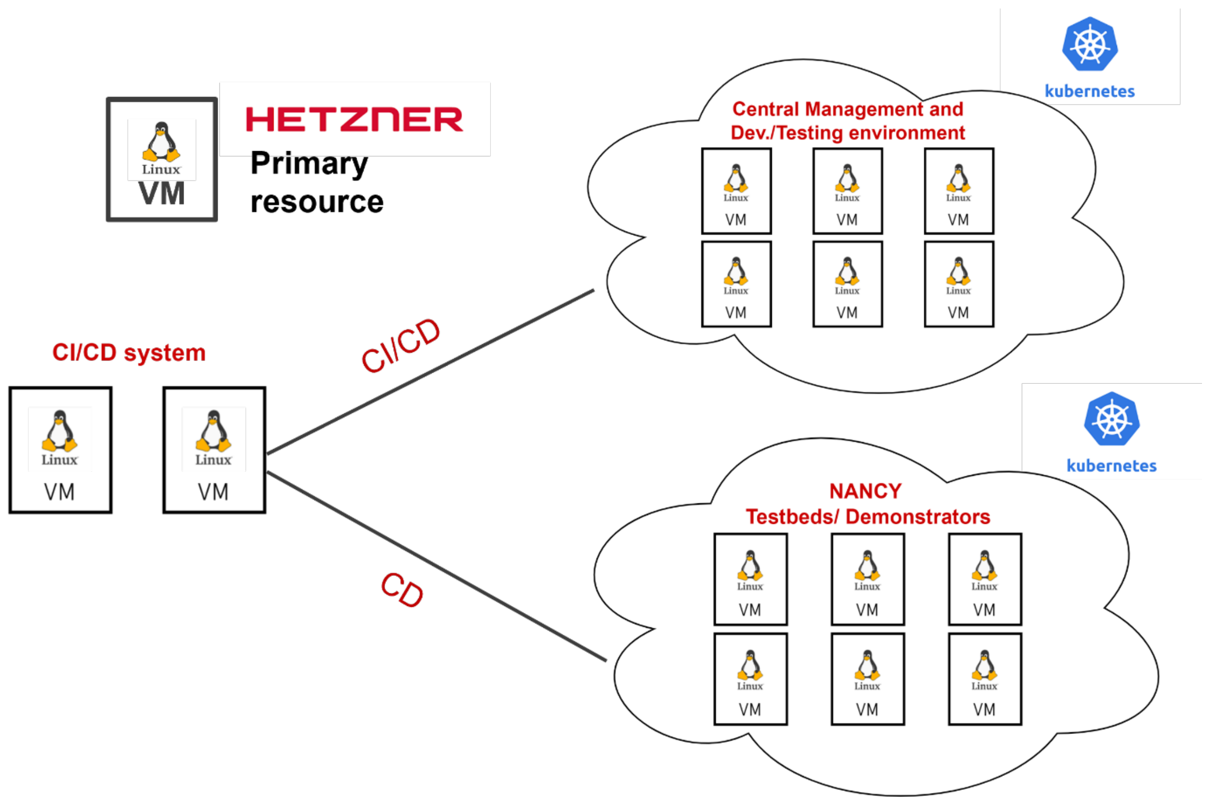

HETZNER
Primary resource
Linux VM
CI/CD system
CI/CD
CD
Central Management and Dev./Testing environment
NANCY Testbeds/ Demonstrators
kubernetes
kubernetes

Figure 2 NANCY CI/CD infrastructure and execution environments

2.2. Integration Points Identification, Specification and Testing

The initial steps of the integration activities included identifying the integration points (section 6.2) as a comprehensive set of bilateral NANCY components that will interface with each other across the various pilots. One of the objectives of this step was to update the NANCY architecture’s functional and deployment view presented in [1] (and provided also in section 3.1 for your convenience), where the NANCY platform’s interfaces were defined. Specifically, in the current document, each integration point (set of bilateral NANCY components) is linked to one or more NANCY platform interfaces (section 6.2).

After their identification, the integration points were specified in more detail, including the selection of suitable protocols/interface technologies to ensure interoperability and the creation of sequence diagrams to be implemented throughout the integration process (section A). Based on these specifications, precise testing plans have been executed starting from a NANCY component basis (functional tests reported in section 5). The objective of these tests is to first validate each component's functionality independently to make sure that individual components work as expected before integration efforts begin.

Following the validation of components' specific functionalities through functional tests, bilateral integration tests are taking place to verify and ensure that each pair of components previously defined as an integration point can work together seamlessly. These tests focus on the interaction and data exchange between the two components, ensuring that they communicate correctly, function as expected when integrated, and do not introduce any errors or issues. Bilateral integration tests help identify and resolve compatibility issues, interface mismatches, and integration-related bugs early in the development process, thereby improving the overall reliability and stability of the integrated system. An overview of the full set of bilateral integration tests defined in NANCY for the first release of the platform is provided in section 6.3.

In the context of the first release of the NANCY platform, most functional tests have been validated and the bilateral integration tests are ongoing. After these tests are finalised and updated, the plan is to gradually plan and implement larger-scale (end-to-end) tests. The objective of these tests will be to validate complete and integrated workflows (technical testing scenarios) of the NANCY platform from start to finish. These tests will ensure that the involved system components and subsystems interact correctly and operate together as intended for such technical workflows, providing feedback for platform refinements. The definition and results of these tests will be reported in D6.3 *"NANCY Integrated System – Final Version"*.

2.3. Communication Channels and Management of Integration Activities

To support the NANCY platform's integration activities, different communication channels have been established. Regular WP6 meetings are held weekly to monitor the status of ongoing activities, as well as ad-hoc meetings scheduled to address specific identified challenges, resolve technical issues etc. Moreover, shared project planning and integration monitoring tools based on Github [4] have been put in place and will be used extensively to support a common view on the status, blocking issues and progress of the integration/testing activities among the technical partners. Last but not least, dedicated NANCY communication channels on Slack platform [5] have been set up, facilitating quick and effective exchanges for brainstorming and issue resolution among NANCY component providers.

3. Instantiation of the NANCY Reference Architecture

3.1. Overview of NANCY Architecture

Figure 3 visualizes the functional and deployment view of the NANCY architecture. This figure provides both a horizontal split of the system across domains (from user equipment at the left-hand side to a core domain at the right-hand side) as well as a vertical split across layers (from the infrastructure layer at the bottom to the business layer at the top). These domains and layers are explained in detail in Deliverable D6.1 "B-RAN and 5G End-to-end Facilities Setup" submitted in M20.

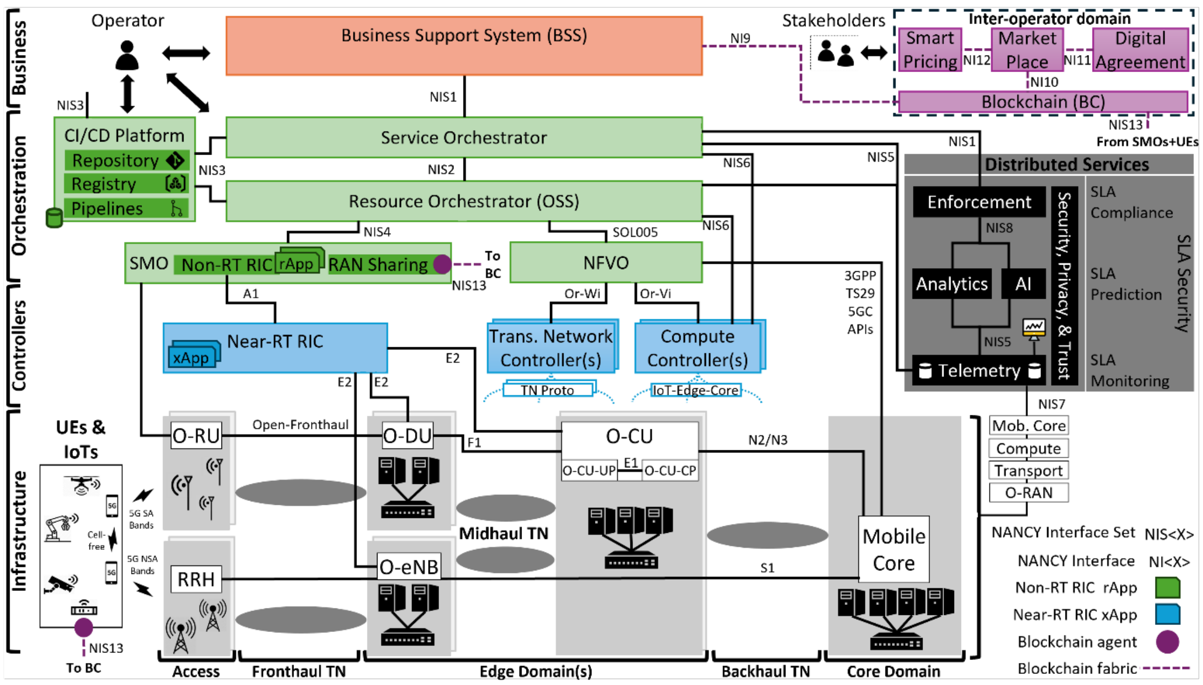


Figure 3 Functional and Deployment view of NANCY reference architecture

Figure 3 also shows all the interfaces among the NANCY platform components. Table 1 lists the NANCY interfaces (NI) and NANCY interface sets (NIS) visualised in Figure 3. In column 1, each interface is associated with a unique identifier. NANCY interfaces begin with NI followed by an increasing number, while NANCY interface sets begin with NIS followed by an increasing number. Column 2 states the high-level objective of each NI and/or NIS, while column 3 relates the interface with one or more logical components of the NANCY architecture as shown in Figure 3. Finally, column 4 states the partners responsible for the implementation of these components and interfaces.

Table 1 NANCY interfaces and the related NANCY architectural components

Interface ID	Interface Objective	Related Logical Architecture Component(s)	Responsible Partners
NIS1	End user Service Exposure set of APIs	Service Orchestrator (NBI)	UBI, I2CAT
NIS2	Resource Service Exposure set of APIs	Resource Orchestrator (NBI)	UBI, I2CAT
NIS3	Service Repository and Registry set of APIs	CI/CD Platform, Service/Resource Orchestrator	INTRA, UBI/ I2CAT
NIS4	O-RAN Orchestration set of APIs	SMO NBI	Testbed owners (T6.5, T6.6, T6.8, T6.9)

NIS5	Service and Resource Telemetry Exposure set of APIs	Telemetry Infrastructure Service (NBI)	Testbed owners (T6.5, T6.6, T6.8, T6.9)
NIS6	Compute slice management set of APIs	Compute Controller(s) (NBI)	Testbed owners (T6.5, T6.6, T6.7, T6.8, T6.9)
NIS7	Telemetry collection set of APIs from various domains, including (O-)RAN, Transport Network, Compute, and Mob. Core	Telemetry, Infrastructure (Testbeds)	Testbed owners (T6.5, T6.6, T6.7, T6.8, T6.9) and T2.3 partners
NIS8	(Smart) events/outputs exposure set of APIs (AI and/or Analytics services)	AI/Analytics, Enforcement	Partners in T2.3-T2.4, T3.2-T3.4, T4.4, T5.4-T5.5
NI9	Secure Product Exposure API	Blockchain, BSS	NEC, TEC
NI10	Smart Contract deployed on the blockchain	Marketplace, Blockchain	TEC, NEC
NI11	Blockchain oracle – server API	Marketplace, Digital Agreement	TEC, DRAXIS
NI12	Blockchain oracle – server API	Marketplace, Smart Pricing	TEC, 8BELLS
NIS13	Secure (gRPC) interface between blockchain and wallet owners (e.g., UEs, IoTs, etc.)	Blockchain, user's wallet (can involve PQC)	NEC, TDIS, WP6 partners

3.2. Deployment of NANCY Architecture

Following the definition of the functional and deployment view of the NANCY architecture presented in section 3.1, the deployment/operation domains of NANCY architecture, namely: *Central Management*, *Inter-operator* and *NANCY testbeds/demonstrators* are described here in detail. As shown in Figure 4, the secure connectivity across the various NANCY deployment domains and their associated services will be ensured through the establishment of an appropriate VPN fabric.

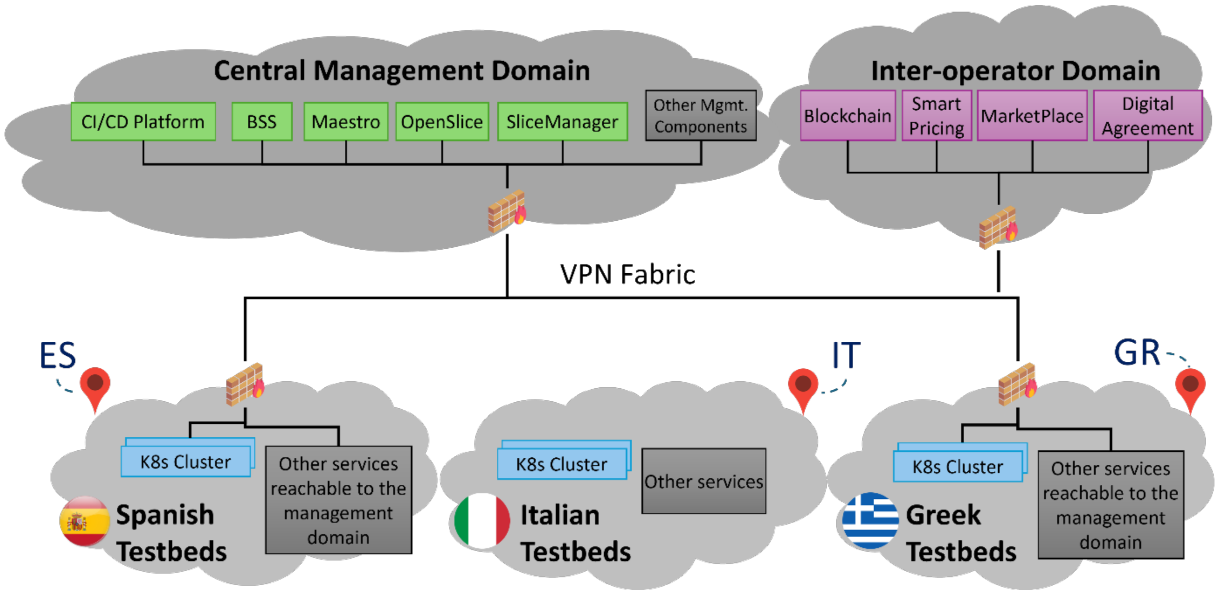


Figure 4 NANCY platform operation domains and deployment of the NANCY architecture

3.2.1. Central Management Domain

The NANCY Central Management Domain incorporates the end-to-end slicing, service and resource orchestration services of NANCY, already analysed in [1] and further described below. Moreover, the CI/CD system that has been put in place supports the efficient central management of the deployments and testing towards the different clusters/environments of NANCY, as it will be analysed in the following.

The Orchestration and CI/CD system services are hosted in the central NANCY Kubernetes cluster, which serves a dual purpose:

- To establish a common development and testing environment for containerized NANCY components, aiming to verify their functionalities and ensure integration among various components and services prior to deployment in operational settings (separate Kubernetes clusters at NANCY testbeds/demonstrators).
- To host NANCY Management domain services, which include as mentioned above orchestration services that are accessible across different operational environments via secure VPN tunnels.

The CI/CD Platform

The CI/CD Platform provides DevOps automation capabilities through the configuration of software build, testing and deployment pipelines for the different NANCY components and services. As it has been described in [1], it is composed of the ***NANCY Github organisation*** which is used as the code repository and version control system; the ***Jenkins CI server*** to configure and execute the software testing and delivery pipelines; the ***Harbor private Container registry*** managing the NANCY container images and Helm charts. The NANCY CI/CD system is connected to the central NANCY Kubernetes cluster, supporting the development and testing of NANCY components. The CI/CD system services are also securely connected to the Kubernetes environments of the different NANCY testbeds and demonstrators to control the deployments of NANCY containerized components and services either directly through the CI server or through the MAESTRO service orchestrator.

Detailed descriptions of the NANCY CI/CD system and integration environment are provided in section 4.

Business Support System (BSS):

The role of BSS in NANCY is to translate a product order made by a NANCY stakeholder through the Marketplace (Inter-operator domain) into a corresponding service order that the BSS will issue towards the operator's service orchestrator. To do so, the BSS must implement a secure interface with the Marketplace using a wallet agent. This agent authenticates against the blockchain that resides underneath the Marketplace to bridge the operator's domain with the inter-operator domain in a secure and trusted manner.

To realise the translation of a product order into a service order, the BSS must utilise the NBI of the service orchestrator i.e., TMF 641 service order API of Maestro, or fill in SliceManager NBI for service orders.

Service and Resource Orchestrator

Maestro

Maestro is a cloud-native service orchestrator designed to manage the lifecycle of end-to-end services across geo-distributed infrastructures. It facilitates automated deployment, scaling, and lifecycle management of microservices-based applications. Maestro integrates with Kubernetes and OpenStack environments to deploy services via containers or virtual machines. It connects with OpenSlice for resource provisioning and relies on Helm charts for service descriptor generation and deployment on Kubernetes clusters. Maestro operates in environments involving Kubernetes clusters and integrates with edge and core infrastructures. It uses Kubernetes-as-a-Service (K8s-aaS) for resource orchestration. For more details on Maestro see the online documentation [6] and [7]. Maestro supports Helm [8] and Docker Compose [9] for service packaging. Container images and Helm charts are managed in the NANCY Harbor container registry, ensuring smooth retrieval of artifacts during deployment. Service providers onboard containerized services, which are subsequently deployed via Kubernetes.

OpenSlice

OpenSlice, developed by the ETSI Software Development Group (SDG OSL), is an open-source, service-based Operations Support System (OSS) designed to deliver Network Slice as a Service (NSaaS). It aligns with specifications from major standards organisations, including ETSI, TM Forum, and GSMA [10]. OpenSlice is a resource orchestrator responsible for provisioning computational, memory, and storage resources. It works closely with Maestro by providing the necessary infrastructure-level resource allocation (e.g., Kubernetes namespace quotas). OpenSlice manages resources in Kubernetes clusters, ensuring that slices or namespaces are configured with appropriate compute, memory, and storage limits. OpenSlice's architecture supports the onboarding and management of NFV artifacts. This process involves managing containerized network functions and their deployment, which can be facilitated by container registries like Harbor [11]. Harbor could serve as a repository for storing and managing container images and Helm charts, essential for deploying services in Kubernetes environments. These components collaborate in the Central Management domain to enable service orchestration and resource elasticity. Harbor provides the Helm charts and container images that

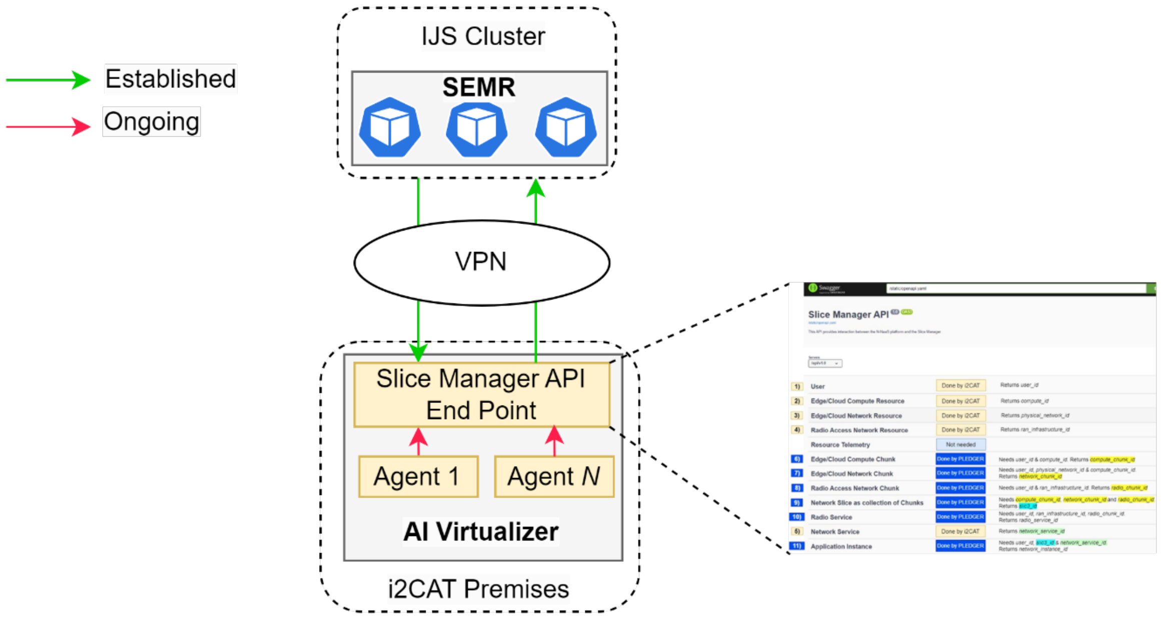
Maestro utilizes for service deployment, while OpenSlice ensures the underlying resources are adequately allocated and managed.

AI Virtualizer: The **AI Virtualizer (AI virt)** is designed to enable dynamic, efficient, and decentralised resource management. It consists of several interconnected components, including a telemetry server and agents deployed within each slice (i.e., Kubernetes namespaces). These agents manage CPU resources and communicate via a Kafka bus deployed in the default namespace, ensuring seamless coordination across the system.

A key interface of the Virtualizer is the **Slice Manager (SM)**, which operates through API endpoints to enforce resource allocation actions such as CPU, RAM, and storage assignment. To orchestrate the resources dedicated to Self-Evolving Model Repository (SEMR) in the distant Kubernetes cluster of IJS, the Slice Manager---hosted on i2CAT premises---is securely connected to the Kubernetes cluster hosted by IJS through a dedicated VPN channel using the kubeconfig file [12]. This secure communication pathway facilitates real-time monitoring and control of the Kubernetes cluster by the SM, ensuring consistent and reliable orchestration of resources.

Within this architecture, the SM plays a pivotal role by providing granular control over slices—logical partitions of shared physical and virtual resources. It enables optimal resource allocation based on the specific requirements of applications or tenants, ensuring resource isolation, scalability, and flexibility in multi-tenant environments. By governing slices through API calls, the SM supports high-level management tasks such as slice creation, scaling, and teardown while dynamically adjusting resource allocations to meet varying workloads and user demands.

Currently, integration between the Slice Manager and the Kubernetes cluster hosting the Self-Evolving Model Repository (SEMR) which is an AI model orchestrator responsible for managing model lifecycles and monitoring performance (as detailed in [13]), is achieved via the Slice Manager API, to actuate infrastructure-level resource management tasks within specific namespaces. The next step in this evolution involves leveraging AI agents to make autonomous decisions and trigger API calls, enhancing the intelligence and adaptability of the resource management process (Figure 5).



Slice Manager API

This API provides interaction between the N-NaaS platform and the Slice Manager

1)	User	Done by i2CAT	Returns *user_id*
2)	Edge/Cloud Compute Resource	Done by i2CAT	Returns *compute_id*
3)	Edge/Cloud Network Resource	Done by i2CAT	Returns *physical_network_id*
4)	Radio Access Network Resource	Done by i2CAT	Returns *ran_infrastructure_id*
	Resource Telemetry	Not needed	
6)	Edge/Cloud Compute Chunk	Done by PLEDGER	Needs *user_id* & *compute_id*. Returns *compute_chunk_id*
7)	Edge/Cloud Network Chunk	Done by PLEDGER	Needs *user_id*, *physical_network_id* & *compute_chunk_id*. Returns *network_chunk_id*
8)	Radio Access Network Chunk	Done by PLEDGER	Needs *user_id* & *ran_infrastructure_id*. Returns *radio_chunk_id*
9)	Network Slice as collection of Chunks	Done by PLEDGER	Needs *compute_chunk_id*, *network_chunk_id* and *radio_chunk_id* Returns *slic3_id*
10)	Radio Service	Done by PLEDGER	Needs *user_id*, *ran_infrastructure_id*, *radio_chunk_id*. Returns *radio_service_id*
5)	Network Service	Done by i2CAT	Returns *network_service_id*
11)	Application Instance	Done by PLEDGER	Needs *user_id*, *slic3_id* & *network_service_id*. Returns *network_instance_id*

Figure 5 High-level view of integration among i2CAT's Slice Manager and SEMR hosted at IJS Kubernetes cluster

3.2.2. Inter-operator Domain

As extensively described in [1], in the NANCY architecture, two differentiated vertical domains are shown: the intra-operator domain and the inter-operator domain. In addition to these, the architecture also proposes a set of distributed services to which the local service orchestrator can access.

Initially, a basic workflow for exclusively working inside the inter-operator domain was presented in M18. The workflow included three fundamental stages: (1) listing services in the marketplace, (2) service selection and (3) SLA signature, and it allowed a dummy service provider to effectively expose different services that could be granted to a dummy service consumer, who eventually – and to access said services – should sign a new SLA.

An extended workflow has been presented in M24 (see [14]). The main difference between this and the basic service selection and agreement workflow is that now the inter-operator domain works together with various WP3 and WP4 actors in the intra-operator domain, rendering a complete workflow for service handover. For the full workflow, please refer to Figure 6.

Blockchain and wallets

The NANCY wallet (see [14]) is a Kotlin GRPC server which exposes calls for working with the NANCY blockchain, the PQC component and the SSI infrastructure. The NANCY wallet (or blockchain adaptor, in certain contexts) also serves as a secure repository for the credentials and identities needed to access and interact with the Fabric-based NANCY blockchain network. In this sense, the NANCY wallet stores user identities, which can include (1) X.509 certificates as issued by the Fabric's Certificate Authority (CA) to authenticate a user or organisation's identity, (2) the user's self-generated DIDs, and (3) private keys used to sign transactions on behalf of the user. It is important to repeat that the NANCY network is permissioned, so users must have valid credentials in their wallets to interact with the blockchain. Once authenticated, the user can e.g. query the ledger or submit transactions by interacting with the chaincode (e.g. the NANCY Marketplace).

The NANCY Blockchain is a NEC-hosted, Hyperledger Fabric v2.2.0 based blockchain, with various security and privacy improvements (see [14]). To interact with a Hyperledger Fabric blockchain using a wallet, the process firstly involves setting up and registering identities, obtaining credentials, and configuring the wallet for interaction (see A.22 and A.23 in Annex).

Once connected to a blockchain, any user can submit transactions or query the ledger through the smart contracts APIs (see A.22 and A.23 in Annex). The main difference is that the originally (basic workflow) called "dummy user" and identified as the service provider is now (extended workflow) the Business Support System (BSS) of a local operator. Hence, it is this BSS the one equipped with a wallet and a valid identity. The NANCY Marketplace (see next section) is set up as a smart contract working in the NANCY Blockchain.

All these local operators, through their BSS, can publish services in the Marketplace by means of the *createService(JSON String)* method. Descriptions for these services include performance indicators that help the Marketplace chaincode to search the most suitable services and providers for a given request. In addition to this, the operators can add, update or delete themselves as providers and to add, update or delete services using GRPC calls to the Marketplace smart contract, effectively changing the ledger status.

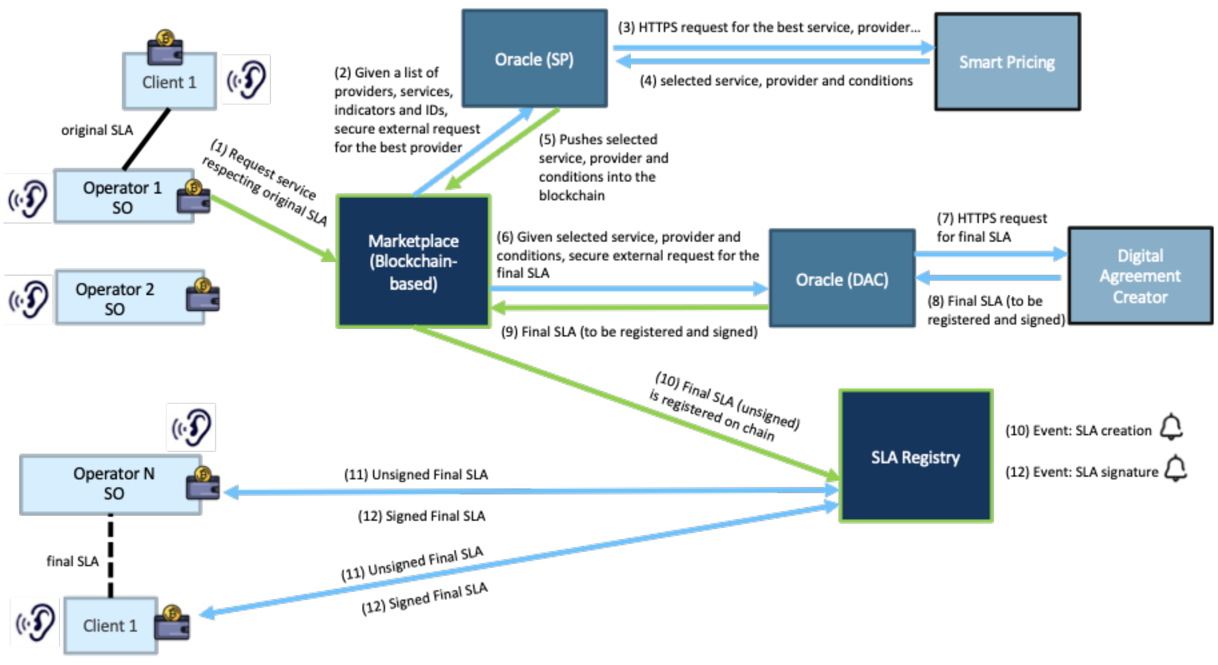

Client 1
original SLA
Operator 1 SO
Operator 2 SO
Operator N SO
final SLA
Client 1
(1) Request service respecting original SLA
(2) Given a list of providers, services, indicators and IDs, secure external request for the best provider
Oracle (SP)
(3) HTTPS request for the best service, provider...
(4) selected service, provider and conditions
Smart Pricing
(5) Pushes selected service, provider and conditions into the blockchain
Marketplace (Blockchain-based)
(6) Given selected service, provider and conditions, secure external request for the final SLA
Oracle (DAC)
(7) HTTPS request for final SLA
Digital Agreement Creator
(8) Final SLA (to be registered and signed)
(9) Final SLA (to be registered and signed)
(10) Final SLA (unsigned) is registered on chain
SLA Registry
(10) Event: SLA creation
(12) Event: SLA signature
(11) Unsigned Final SLA
(12) Signed Final SLA
(11) Unsigned Final SLA
(12) Signed Final SLA

Figure 6 Service selection inside the offloading and caching workflow

In order to trigger the second step in the extended inter-operator domain workflow, we assume that a service request from a client ("Client 1") has been received and analysed by the decision engines inside the offloading and caching workflow [15], and that there is an initial SLA created. This initial SLA has been forwarded to the service orchestrator (SO) of the local operator receiving the service request. In addition to this, we assume that the SO of the local operator is unable to fulfil this initial SLA, hence it needs to contact the inter-operator domain through its wallet. This contact is done using a *createSearch(JSON string)* method.

This starts a series of interactions between the Marketplace, its oracles, the Smart Pricing component and the Digital Agreement Creator (DAC) component. Eventually, a new SLA will be created and, potentially, signed between the interested parties.

When the provider (operator) proceeds, with the signature, its wallet issues a signing transaction to the blockchain. The transaction ID serves as a valid signature in the SLA. In the case of the service consumer, if it is a user equipment, its wallet will issue a transaction with a PQC signature, which can be automatically verified by the SLA Registry smart contract. Finally, both parties are informed of the signed SLA since they are subscribed to such events.

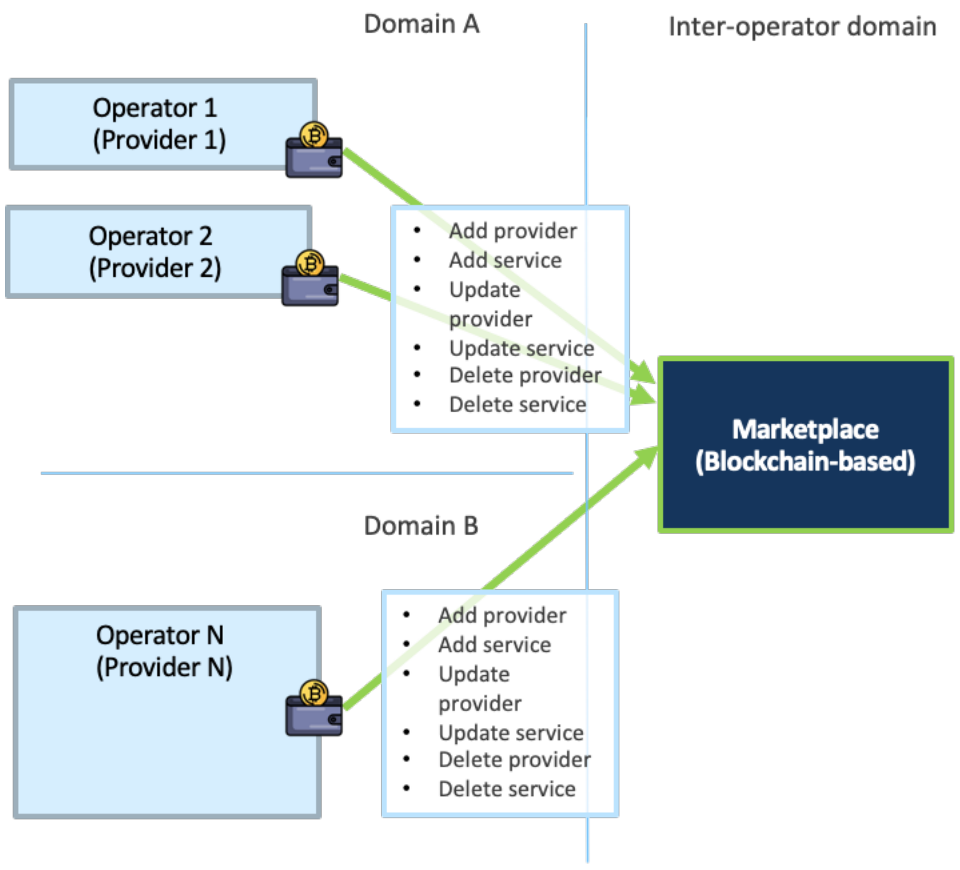


Figure 7 Operators using their wallets to list services in the Marketplace

Marketplace

The marketplace is a decentralized application operating on Smart Contracts. These Smart Contracts are instantiated on the NANCY Blockchain, replicating the marketplace's ledger across all nodes within the distributed network and improving its security in terms of trustworthiness, transparency and availability. The interaction with the marketplace happens through the Blockchain wallets that are deployed on the different users' premises. In this sense, the BSS interacts with the marketplace through the Blockchain wallet for the definition and searching of services and products fulfilling given features.

Additionally, the marketplace requires interaction with the DAC and the SP components for proper operation. These interactions happen using oracles deployed in combination with the Blockchain network.

Digital Agreement Creator

The Digital Agreement Creator (DAC) is a software solution that has been designed with the intention of creating smart contracts among NANCY stakeholders. DAC is based on the Java language and the Spring Boot Framework and is fully Dockerised. It can receive inputs via its RESTful interface concerning various parameters, such as providerId, consumerId, service, price, conditions, etc. DAC then creates ad-hoc containers that hold the smart contracts for relevant parties based on the provided input. These containers are assigned a unique identification number, generated by DAC, that works as a hash of the smart contract. The latter containers currently are being deployed to the same place where DAC is been deployed.

The DAC application has the capability of dynamically generating and managing these ad-hoc containers, which serve as holders of the smart contracts for the Hyperledger Fabric blockchain, via REST API requests. Its scalable design and modular architecture allow for concerns to be separated between the orchestration, generation, and API layers, while its RESTful APIs enable extensibility and integration with other systems. The component is hosted on DRAXIS premises, with communication handled via REST APIs and it is built using Java and Maven management and comprehension tools.

Smart Pricing

The Smart Pricing Component is designed to determine optimal pricing strategies between different providers, ensuring competitive pricing. It operates by receiving data for each provider in a given region, including information on minimum and maximum prices. Each provider is represented by an AI agent, and a reverse blind auction is conducted where the agent offering the best price wins. The Smart Pricing Component interacts with the Marketplace by receiving provider information and sending auction outcomes. Further details on the functionality can be found in [16].

The component is hosted on Eight Bells' premises, with communication handled via a REST API. It is built using Python and incorporates reinforcement learning libraries for AI decision-making.

3.2.3. NANCY Testbeds and Demonstrators

As shown in Figure 4, the Central Management and Inter-operator domains connect to the different testbeds and demonstrators through secure VPN connections. All demonstrators are connected to the NANCY Blockchain of the inter-operator domain described in section 3.2.2, while the Central Management domain's slicing and orchestration components manage the Greek and Spanish testbeds and demonstrators.

The detailed lists of services provisioned at the different testbeds have been provided in [1]. In the following, an updated list of the NANCY software components developed in the context of WP2-WP5 and classified as "NANCY Use Case Services" in [1] is provided. This list (Table 2) focuses on the components' deployment location, considering the NANCY operational environments. The term operational in this context refers to the environments which will be used for the end-to-end execution of the different NANCY use cases across the different testbeds and demonstrators, and it is differentiated from the testing/staging environments. Table 2 also provides the acronyms of the NANCY components referenced throughout the document.

In this context, each component is associated with one of the logical deployment domains of NANCY: Central Management, Inter-operator, Local (indicating deployments at the premises of the different NANCY testbeds and demonstrators) and Offline. The term Offline is used to describe components of the NANCY ecosystem which do not directly participate in any real-time operations of the platform but are rather used to provide offline results (e.g., BRAN model, simulators) based on real datasets, and possibly feed these results to other components of the NANCY platform.

Besides the logical deployment domain of each NANCY component, its physical deployment location is also provided in Table 2. It should be highlighted that according to the integration points specifications' requirements (section 6), the secure interconnection of different deployment locations and corresponding services is or will be performed (e.g., through VPN connections and/or REST API communications over HTTPS).

Finally, it is worth mentioning that NANCY's offloading service is not explicitly mentioned in this components list. The offloading service (comprehensively detailed in [15]) is understood as a series of mechanisms that permit to encompass the operation and alignment of multiple NANCY components that do appear in the list, e.g., Orchestrator, Network Information Framework (NIF), Marketplace, DAC, ID Management (ID_Mgnt), etc. Thus, the offloading functionality will foster the integration and coordination of such components to enable the end-to-end workflow needed for its realisation. To this end, the different interfaces and integration points described in the next sections become essential.

Table 2 Listing of NANCY (WP2-WP5) components and their deployment locations

NANCY Component	Provider/WP	NANCY Domain (Central Management, Inter-Operator, Local, Offline processing)	Deployment location
MultiRAT-Nomadic Connectivity Provider (MRAT-NCP)	UMU/WP4	Local	Testbed/Demonstrator
NANCY ID Management (ID_Mgnt)	UMU/WP4, WP5	Local	Testbed/Demonstrator
Digital Agreement Creator (DAC)	DRAXIS/WP5	Inter-Operator	DRAXIS premises
Blockchain (BC)	NEC/WP5	Inter-Operator	NEC premises
Wallet	NEC/WP5	Inter-Operator	Testbed/Demonstrator
AI Virtualizer (AI Virt)	i2CAT/WP3	Central Management	i2CAT premises
B-RAN Model	INNO/WP2	Offline processing	INNO premises
Semantic communications framework (SemCom)	INNO/WP4	Offline processing	INNO premises
QKD Simulator (QKDSim)	INNO/WP5	Offline processing	INTRA cloud
VOSySmonitor and vManager (VOSySmonitor)	VOS/WP3, WP4	Local	Testbed/Demonstrator
Marketplace/BSS	TECN/WP5	Inter-Operator	NEC premises
MAESTRO Service Orchestrator and ETSI Open-Slice Resource Orchestrator	UBI/WP3, WP4	Central Management	INTRA cloud
Models	IJS/WP3	Local	Testbed/Demonstrator

Self-Evolving Model Repo (SEMR)	IJS/WP3	Local	Testbed/Demonstrator
Elasticity	IJS/WP4	Local	Testbed/Demonstrator
PQC Signature (PQCSig)	TDIS/WP5	Local	Testbed/Demonstrator
Throughput Forecasting Service (TFS)	CERTH/WP3	Central Management and Local	For Greek demonstration: INTRA Cloud For Spanish demonstration: Testbed/Demonstrator
RIC Manager (RICMngr)	i2CAT/WP3	Central Management	i2CAT premises
AI-driven Network Quality Module (AINQM)	CERTH/WP2	Central Management and/or Local	For Greek demonstration: INTRA Cloud For Spanish demonstration: Testbed/Demonstrator
Network Information Framework (NIF)	8Bells/WP2	Offline	8Bells premises
Smart Pricing Policies (SPP)	8Bells/WP4	Inter-Operator	8Bells premises
eXplainable AI Toolkit (XAI)	MINDS/WP5	Local	Testbed/Demonstrator
Federated Learning-based Anomaly (Intrusion) Detection (FL-IDS)	MINDS/WP5	Local	Testbed/Demonstrator
Malicious Traffic Generation Application and Resource Monitoring (MTG-RM)	SSS/WP5	Local	Testbed/Demonstrator
PQC Secure Communication Library (PQC-SC)	TEI/WP5	Local	Testbed/Demonstrator
Distributed Anomaly Detection and Mitigation (D-ADM)	CRAT/WP5	Local	Testbed/Demonstrator

3.3. Reusability of NANCY Solution Towards new Deployment Environments

The NANCY integration framework strongly depends on cloud-native technologies, following integration by design techniques. Specifically, containerization has been used to a large extent for packaging many NANCY software components. This offers many advantages towards the reusability of NANCY software, leveraging straightforward deployments and testing both in the context of the NANCY project as well as considering its future use across different research and open-source groups as described in the following.

- **Portability**: Containers encapsulate software components, bundling their dependencies, libraries, and runtime into a unified, standalone unit. This guarantees consistent and reliable operation across various environments, whether on a developer's machine, a staging server, or a production environment. By enabling a "build once, deploy anywhere" approach, containers minimize issues related to environment-specific differences and incompatibilities.
- **Consistent runtime**: Each container operates within its own isolated runtime environment. This isolation guarantees that the software functions consistently, regardless of the underlying host system (e.g., different operating systems, cloud providers, or hardware configurations).
- **Dependency isolation**: Containers bundle all the dependencies necessary to execute the software, functioning independently of the host system. This separation allows multiple versions of the same component, such as different Python or Java versions for example, to operate simultaneously without conflicts within the same environment.
- **Simplified testing and debugging**: Containers ensure that developers, testers, and operational teams work within the same environment. Test scenarios can be replicated exactly across environments, ensuring that reusable components behave as expected. Debugging is more straightforward because the container's environment mirrors production environments.
- **Ease of Collaboration and Sharing**: Container registries (e.g., Docker Hub, or private registries like the one of NANCY described in section 4.2.4) allow teams to share pre-built container images easily. Developers can reuse these shared containers across different projects and environments without having to rebuild or reconfigure them.
- **Resource efficiency**: Containers are more lightweight than traditional virtual machines (VMs), enabling more efficient resource utilisation. A single host can support multiple containers, making them highly practical for reuse across a variety of workloads.

Moreover, the use of Kubernetes as target deployment and container orchestration environment across the NANCY central management and several testbed/demonstrator domains offers significant advantages in the sense of integrating the NANCY components and services in a powerful unified platform. As a powerful open-source container orchestration platform, Kubernetes natively supports:

- **Portability across environments:** Standardised configuration based on declarative YAML/JSON configuration files or Helm chart is used to define deployments, services, and other Kubernetes resources. Such blueprint files can be used across different execution environments (e.g., development, staging, production) with minimal modifications. Moreover, Kubernetes is platform-agnostic allowing workloads to be deployed on any cloud provider (AWS, Azure, GCP) or on-premises infrastructure (like in the case of NANCY) without changes.
- **Microservices-Oriented:** Each service in Kubernetes can be containerized and reused independently, allowing developers to share and redeploy modular components.
- **Scalability and Flexibility:** Kubernetes can automatically scale applications according to workload, ensuring that services are reused efficiently across environments with different demands.
- **CI/CD and automation:** Kubernetes easily integrates with CI/CD systems, facilitating the implementation of deployment pipelines and automation for the reuse of services and applications across separate deployment environments. As described in sections 2.1 and 3.2.1 and will be further explained in sections 4.1- 4.3, NANCY benefits from such an integration through its dedicated CI/CD system and its connection to the different Kubernetes clusters of the project.

In those configurations where containerization was not the ideal solution to offering a flexible deployment and orchestration environment, alternative solutions have been explored, having nevertheless in mind portability and, in general, the reusability of the technologies being used. This is the case of the edge/far edge environment, where low-power ARM devices are a more suitable choice compared to power-hungry systems embedding Intel or AMD architectures. In these cases, the partitioning of the hardware using the firmware-level VOSySmonitor [2] [15] was considered. To answer the need for portability, reusability and interoperability, the vManager system service was developed to open the resulting partitioning technology to the libvirt orchestration/virtualisation library [17], which in turn allows the orchestration of the partitions via solutions like OpenStack [18]. libvirt allows as well to wire this partition technology to CI/CD systems, allowing complex automated pipelines that seamlessly deploy tests to both virtual machines and partitions.

Finally, interoperability and reusability are core principles of NANCY's innovative approach that addresses the diverse needs of state-of-the-art networks. NANCY designs a security-oriented B-RAN architecture that is capable of operating across both O-RAN (facilitating its use by research groups and open-source communities) and commercial 5G network deployments; thus, achieving significant flexibility. This improves its interoperability and ensures that NANCY technologies can support a wide range of deployment scenarios; thus, enhancing cross-vendor collaboration and reducing vendor lock-in risks. Furthermore, NANCY reinforces reusability and flexibility by deploying 6 testbeds/demonstrators, specifically 3 focused on O-RAN (Greek and Italian testbeds, Spanish demonstrator) and 3 on commercial 5G (Greek and Italian demonstrators, extension of the Spanish demonstrator).

This approach offers numerous advantages. For network operators, interoperability guarantees compatibility with current infrastructure, facilitating seamless integration and minimizing downtime during implementation. The reusability of NANCY solutions facilitates technology developers in growing across many network types, hence augmenting market reach and adoption potential. Finally, NANCY fosters innovation and collaboration within the telecommunications sector by integrating open and commercial ecosystems, hence enhancing the security and resilience of the 5G and future 6G environments. The capacity to modify the same technologies for various deployment contexts reduces the necessity for expensive redevelopment and expedites the implementation process in new networks.

4. NANCY Integration Environment

The NANCY platform employs a diverse set of open-source DevOps tools, carefully chosen for the tasks of building, testing, and deploying its various software components. These tools are integrated through a robust Continuous Integration/Continuous Delivery (CI/CD) framework. This framework not only unifies the different components of the NANCY software but also sets up the necessary environments for development and release processes.

This chapter provides an in-depth examination of the stack components, detailing the installation and configuration procedures as implemented in the NANCY project. Each tool within the stack is discussed, emphasizing its role and contribution to the overall project workflow.

Additionally, the CI/CD pipeline of the NANCY platform has been successfully deployed in a cloud-based environment, transforming it into an automated build system. This automation greatly improves the efficiency of the development process, allowing developers and engineers working with the NANCY toolset to integrate and manage their services seamlessly. This seamless integration is crucial for maintaining consistency and reliability throughout the development lifecycle.

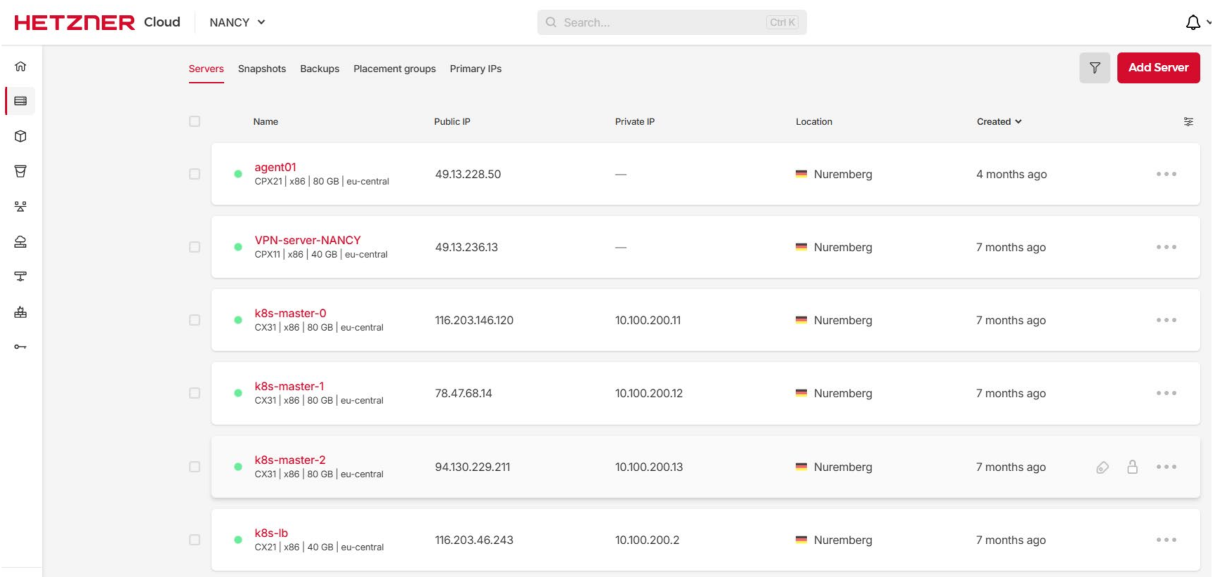
4.1. Hetzner Cloud Infrastructure

Cloud hosting is a service model where computing and storage capabilities are outsourced to an external provider. This provider offers its infrastructure on a flexible, pay-as-you-go basis, allowing clients to scale resources horizontally as needed while ensuring high reliability. For the NANCY project, the software components are hosted on virtual machines provided by Hetzner Cloud [3], a well-known public cloud service.

Hetzner Cloud operates its servers in data centers located in Germany. These servers are equipped with advanced technology, including AMD EPYC™ 2nd Gen and Intel® Xeon® Gold processors, along with fast NVMe SSDs for optimal performance. This setup ensures that the NANCY platform benefits from high-speed and reliable infrastructure.

The cloud infrastructure provided by Hetzner includes several valuable features, such as the ability to take snapshots, perform regular backups, and maintain strict data protection protocols. These features are essential for ensuring the integrity and availability of the NANCY software.

Figure 8 illustrates the specific project allocation for Hetzner Cloud's hosting services dedicated to the NANCY platform. It also shows some of the virtual servers that are integral to the Continuous Integration/Continuous Delivery (CI/CD) stack and the NANCY Kubernetes cluster. As the platform evolves and expands, additional virtual hosts might need to be added. The operating system chosen for these virtual machines is Ubuntu 22.04 LTS, an open-source enterprise operating system.



Name	Public IP	Private IP	Location	Created
agent01 CPX21 | x86 | 80 GB | eu-central	49.13.228.50	—	Nuremberg	4 months ago
VPN-server-NANCY CPX11 | x86 | 40 GB | eu-central	49.13.236.13	—	Nuremberg	7 months ago
k8s-master-0 CX31 | x86 | 80 GB | eu-central	116.203.146.120	10.100.200.11	Nuremberg	7 months ago
k8s-master-1 CX31 | x86 | 80 GB | eu-central	78.47.68.14	10.100.200.12	Nuremberg	7 months ago
k8s-master-2 CX31 | x86 | 80 GB | eu-central	94.130.229.211	10.100.200.13	Nuremberg	7 months ago
k8s-lb CX21 | x86 | 40 GB | eu-central	116.203.46.243	10.100.200.2	Nuremberg	7 months ago

Figure 8 NANCY project and VMs allocation in Hetzner cloud

4.2. CI/CD system

4.2.1. Overview of CI/CD Services

As previously mentioned, a Continuous Integration and Continuous Delivery (CI/CD) system, which includes a collection of tools designed to support the entire software lifecycle from development to the deployment of well-tested and fully functional systems, has been implemented within the NANCY software components. This system incorporates essential services such as GitHub, Jenkins, and Harbor.

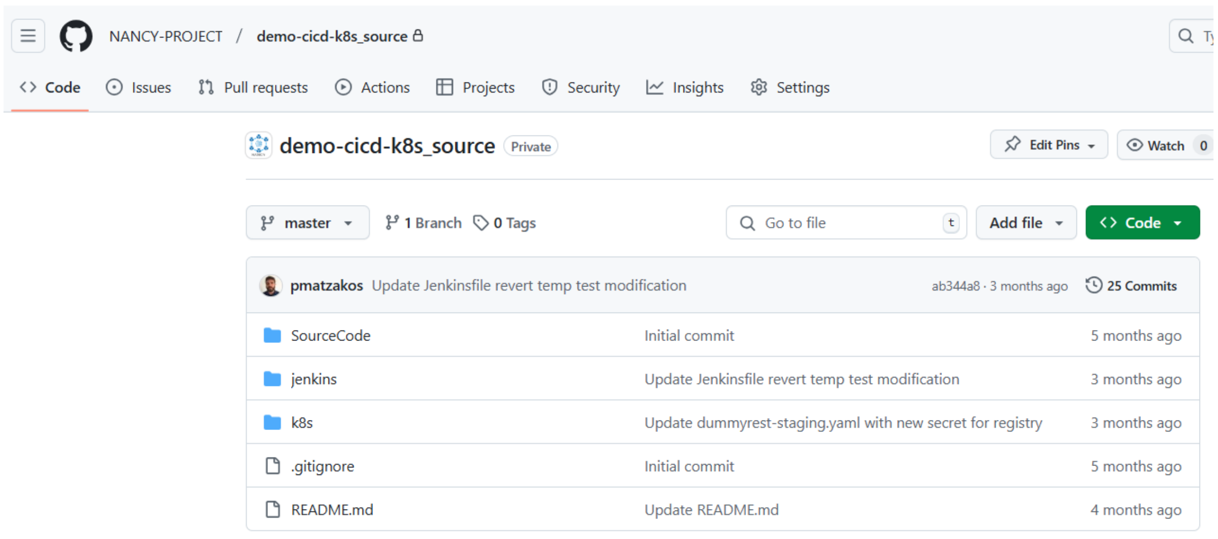
To facilitate better understanding and effective use of the NANCY Continuous Integration/Continuous Delivery (CI/CD) Platform and the Development and Integration Environment among project team members, a "CI/CD User Guide" has been distributed to the NANCY technical partners, and two training workshops have been organised. This guide provides in-depth information on managing both the development and production phases of the CI/CD pipeline. It contains detailed instructions on the overall system architecture, the specific software tools used, and their operational processes to ensure efficiency and best results. Detailed instructions are given through sample projects, covering the cases of full-development (targeting technical teams who are willing to share their code) and testing-only workflows (targeting technical teams who do not wish to share their codes and can only share container images of their components for deployment). Specifically, the guide covers the configuration and usage of GitHub for Source Code Management (SCM) (section 4.2.2), the Jenkins continuous integration server (section 4.2.3), the Harbor container registry (section 4.2.4), and the configuration of Kubernetes clusters and deployments (section 4.3), along with the relevant deployment protocols. Furthermore, the user guide provides some practical implemented examples that illustrate the step-by-step process of building, deploying, and testing.

4.2.2. Github Version Control System

GitHub is used as the version control system for the NANCY software, utilizing its robust, web-based platform to support software development and version control capabilities through Git. GitHub not only provides the essential distributed version control and Source Code Management (SCM) functionality of Git, but also improves these features with its own unique set of tools designed to streamline the development process. The platform's distributed version control system allows multiple developers to work on the NANCY software simultaneously, enabling them to clone the entire repository, make changes locally, and push their updates back to GitHub. This system ensures that there is always a complete version history and backup, which helps maintain the integrity and consistency of the project.

Furthermore, GitHub offers advanced access control mechanisms to enforce confidentiality. These enable NANCY project administrators to manage who can view or edit the project, ensuring that sensitive parts of the software are accessible only to authorised members. This is especially important in collaborative environments where multiple partners or contributors are involved in the project. Collaboration is further improved by GitHub's suite of features designed to support team collaboration and project management. Bug tracking allows team members to report, monitor, and resolve issues efficiently, ensuring that bugs are addressed promptly.

A GitHub NANCY organisation (see Figure 9) has been created and is accessible via: https://github.com/NANCY-PROJECT.


NANCY-PROJECT / demo-cicd-k8s_source

demo-cicd-k8s_source Private

master · 1 Branch · 0 Tags

pmatzakos Update Jenkinsfile revert temp test modification · ab344a8 · 3 months ago · 25 Commits

SourceCode	Initial commit	5 months ago
jenkins	Update Jenkinsfile revert temp test modification	3 months ago
k8s	Update dummyrest-staging.yaml with new secret for registry	3 months ago
.gitignore	Initial commit	5 months ago
README.md	Update README.md	4 months ago

Figure 9 Demo (example) project repository under NANCY Github Organisation

Every technical partner who aims to push source code to a NANCY code repository must have a GitHub account and be a member of the NANCY organisation. Additionally, to improve collaboration and streamline project management, separate GitHub teams have been created for each technical partner involved in the project. This organisational approach ensures that each partner has a dedicated space for their contributions, allowing for better management of permissions, more focused technical discussions, and access to relevant repositories and workflows.

4.2.3. Jenkins Continuous Integration Server

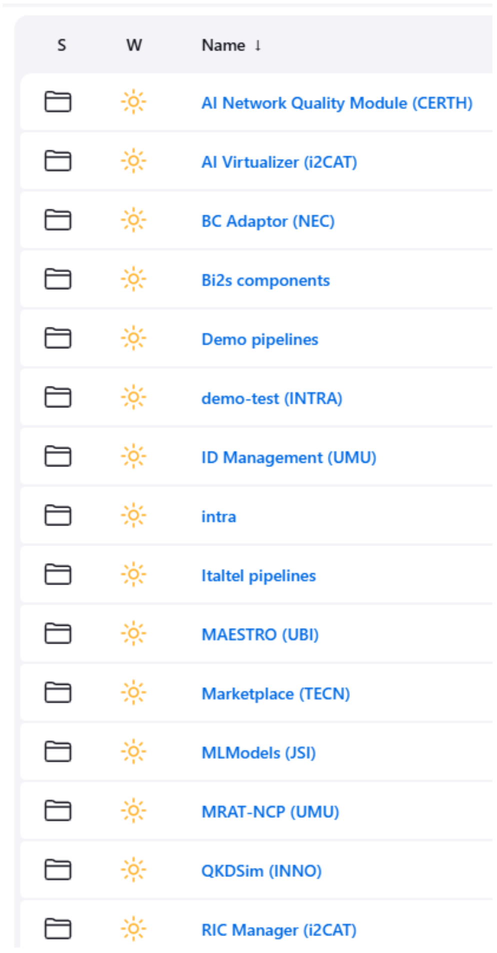
Jenkins has been selected as the Continuous Integration (CI) server within the CI/CD stack for the NANCY platform. The Jenkins server for NANCY is hosted as a Docker container on one of the virtual hosts provided by Hetzner Cloud, accessible via the URL: https://jenkins.nancy.rid-intrasoft.eu/. Additionally, Let's Encrypt [19] has been used to serve as a trusted certificate authority for the CI server, allowing secure access to Jenkins over HTTPS.

The Continuous Integration process on this platform is structured as follows:

- Local Development: Software engineers modify the source code in their local repositories.

- Commit Changes: After making changes locally, they commit these modifications to the shared repository.
- Trigger Build: Upon these commits, a notification is sent to the Jenkins CI server.
- Automated Build and Test: The CI server then pulls the latest source code, builds the application, and performs a list of tests (e.g., unit, functional, integration) which have been configured by the developer/tester of each component.
- Artifact Generation: Following successful tests, the CI server generates deployable, testable artifacts.
- Tagging Builds: The server assigns a build tag to the version of the code it has just built.
- Feedback and Reporting: Jenkins provides the development team with detailed reports on the outcomes of build and test pipelines (notifying them also if a build or test fails).
- Issue Resolution: The corresponding development team resolves any issues as fast as possible.
- Ongoing Integration and Testing: Throughout the project's lifecycle, the server continuously integrates any code updates and executes tests to verify the intended functionality and stability.

This systematic approach leveraging Jenkins ensures a robust, efficient, and continuous integration environment, significantly enhancing and accelerating the development process for the NANCY platform. Specific workspaces (folders) have been created which are mapped to the different NANCY components (Figure 10).



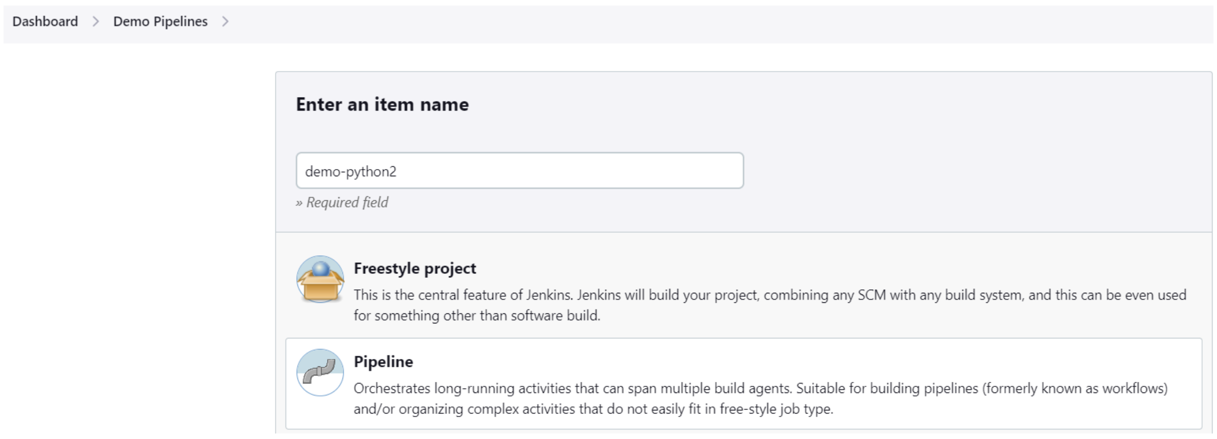
S	W	Name ↓
🗀	☼	AI Network Quality Module (CERTH)
🗀	☼	AI Virtualizer (i2CAT)
🗀	☼	BC Adaptor (NEC)
🗀	☼	Bi2s components
🗀	☼	Demo pipelines
🗀	☼	demo-test (INTRA)
🗀	☼	ID Management (UMU)
🗀	☼	intra
🗀	☼	Italtel pipelines
🗀	☼	MAESTRO (UBI)
🗀	☼	Marketplace (TECN)
🗀	☼	MLModels (JSI)
🗀	☼	MRAT-NCP (UMU)
🗀	☼	QKDSim (INNO)
🗀	☼	RIC Manager (i2CAT)

Figure 10 NANCY component-specific workspaces in Jenkins CI server

A Jenkins account has been provided to each partner with write permissions on specific workspaces. In the following, the steps to create a Jenkins pipeline are described:

1. Create a **New Item** in Jenkins under the folder that corresponds to your component.

- Enter an item name. It is recommended to give a name that matches the GitHub project and the associated branch (e.g., /master or /development).
- Select Pipeline option as shown below in Figure 11.


Dashboard > Demo Pipelines >

Enter an item name

demo-python2

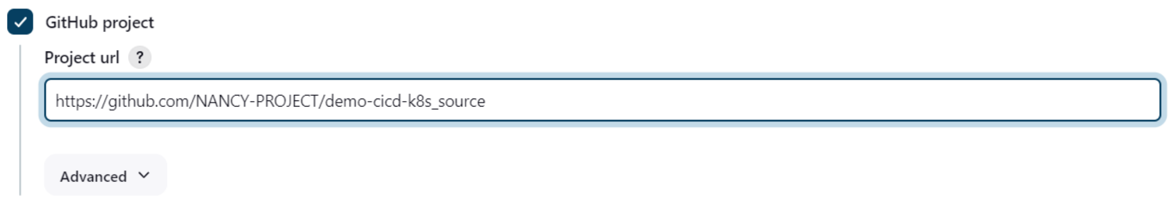
» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Figure 11 Select Pipeline option

2. Select GitHub project and add the URL of your GitHub repository (see Figure 12).

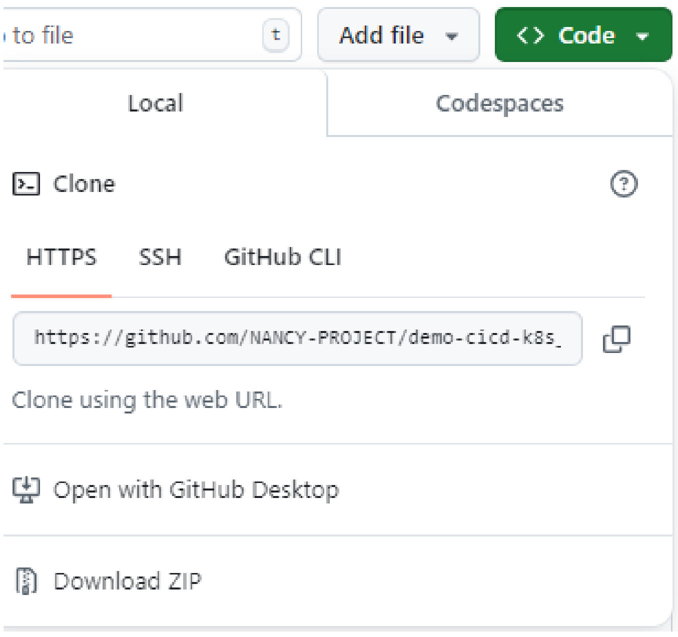

✓ GitHub project

Project url ?

https://github.com/NANCY-PROJECT/demo-cicd-k8s_source

Figure 12 Jenkins pipeline creation: Link with specific Github repository

- You can easily retrieve your Github repository's URL as follows (see Figure 13):

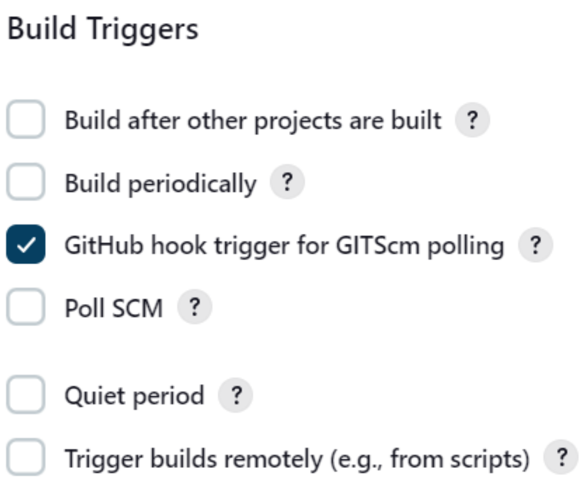


```
https://github.com/NANCY-PROJECT/demo-cicd-k8s_
```

Clone using the web URL.

Figure 13 Retrieve GitHub repository's URL

3. Select the GitHub hook trigger option from the list of Build triggers for Jenkins jobs that you want to be triggered after pushing to the respective GitHub repository (see Figure 14).



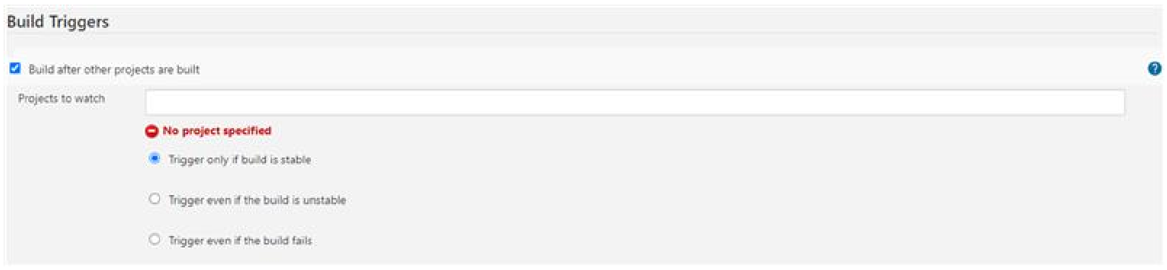
Build Triggers

- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [✓] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?
- [ ] Quiet period ?
- [ ] Trigger builds remotely (e.g., from scripts) ?

Figure 14 Selection of Github hook trigger

- There is also the option to launch a job after another project is finished. In such a scenario, where the job needs to be launched after another job is finished (ex:

functional/integration test-jobs, acceptance test jobs, etc.), the following option needs to be checked, referencing the job that precedes (see Figure 15):


Build Triggers

☑ Build after other projects are built

Projects to watch

No project specified

● Trigger only if build is stable

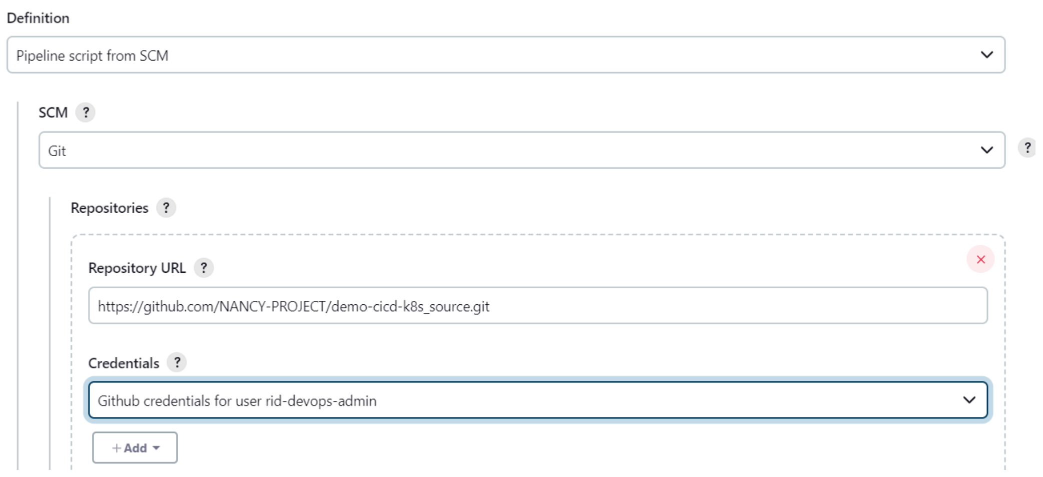
○ Trigger even if the build is unstable

○ Trigger even if the build fails

Figure 15 Configuring consecutive Jenkins jobs

4. In the pipeline section select (see Figure 16):

- Definition: Pipeline script from SCM
- SCM->Git
- Insert the Github repository URL as in step 3.
- Select “Github credentials for user rid-devops-admin”


Definition: Pipeline script from SCM

SCM: Git

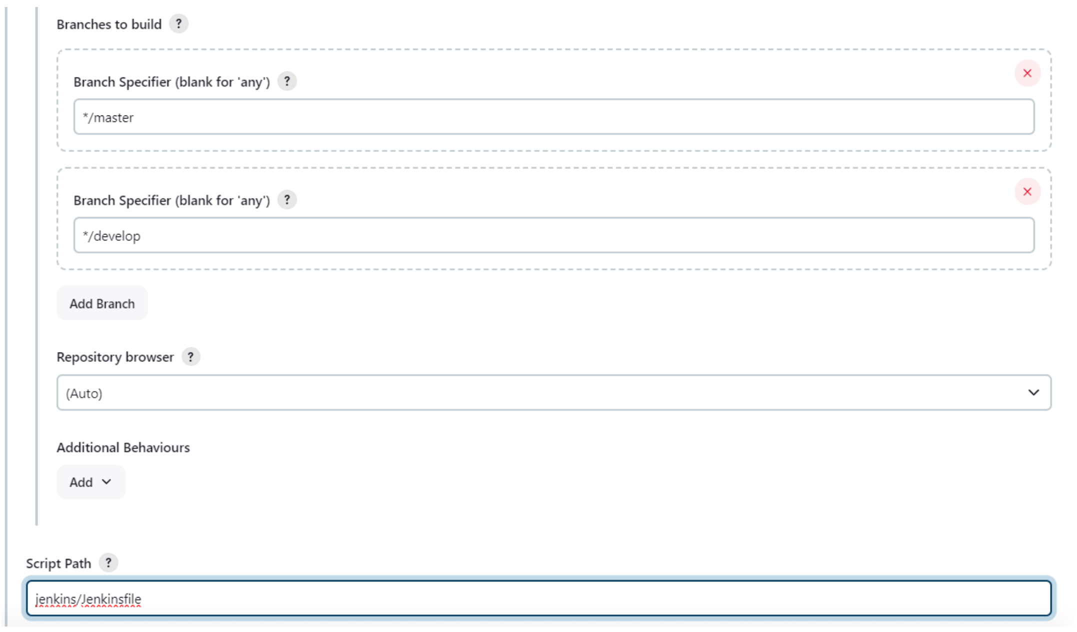
Repositories

Repository URL: https://github.com/NANCY-PROJECT/demo-cicd-k8s_source.git

Credentials: Github credentials for user rid-devops-admin

Figure 16 Jenkins pipeline creation: Further configurations for connecting to the target Github repository

- In "Branches to build" (see Figure 17), add the target branch(es) (e.g., */master, */develop, or */* for any branch).
- In the script path, add the path of your jenkinsfile relative to the repository’s parent directory (e.g., jenkins/Jenkinsfile). The jenkinsfile is used to define the execution steps of a pipeline.


Branches to build ?

Branch Specifier (blank for 'any') ?
*/master

Branch Specifier (blank for 'any') ?
*/develop

Repository browser ?
(Auto)

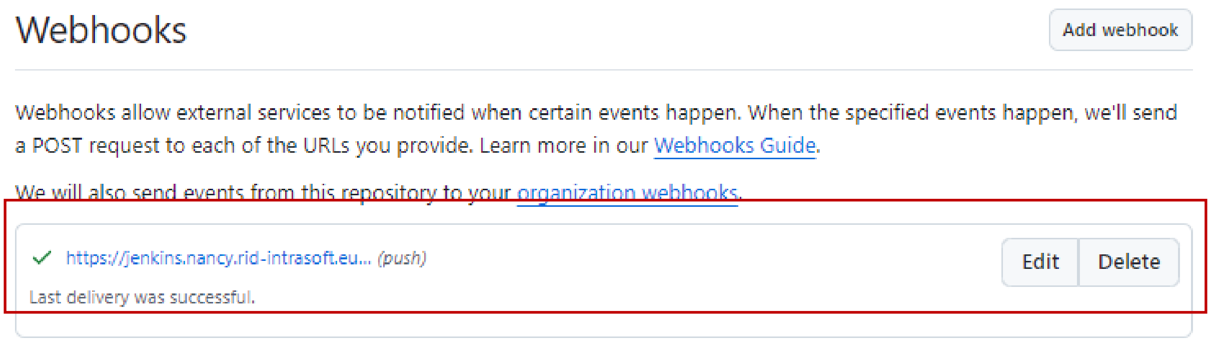
Additional Behaviours

Script Path ?
jenkins/Jenkinsfile

Figure 17 Branches to Build

5. Check if the webhook is registered in Github to automatically trigger the Jenkins pipeline after GitHub push events:

- On your Github repository go to Settings -> Webhooks and verify that the Webhook is registered with your pipeline as shown below (see Figure 18):


Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

We will also send events from this repository to your organization webhooks.

✓ https://jenkins.nancy.rid-intrasoft.eu... (push)

Last delivery was successful.

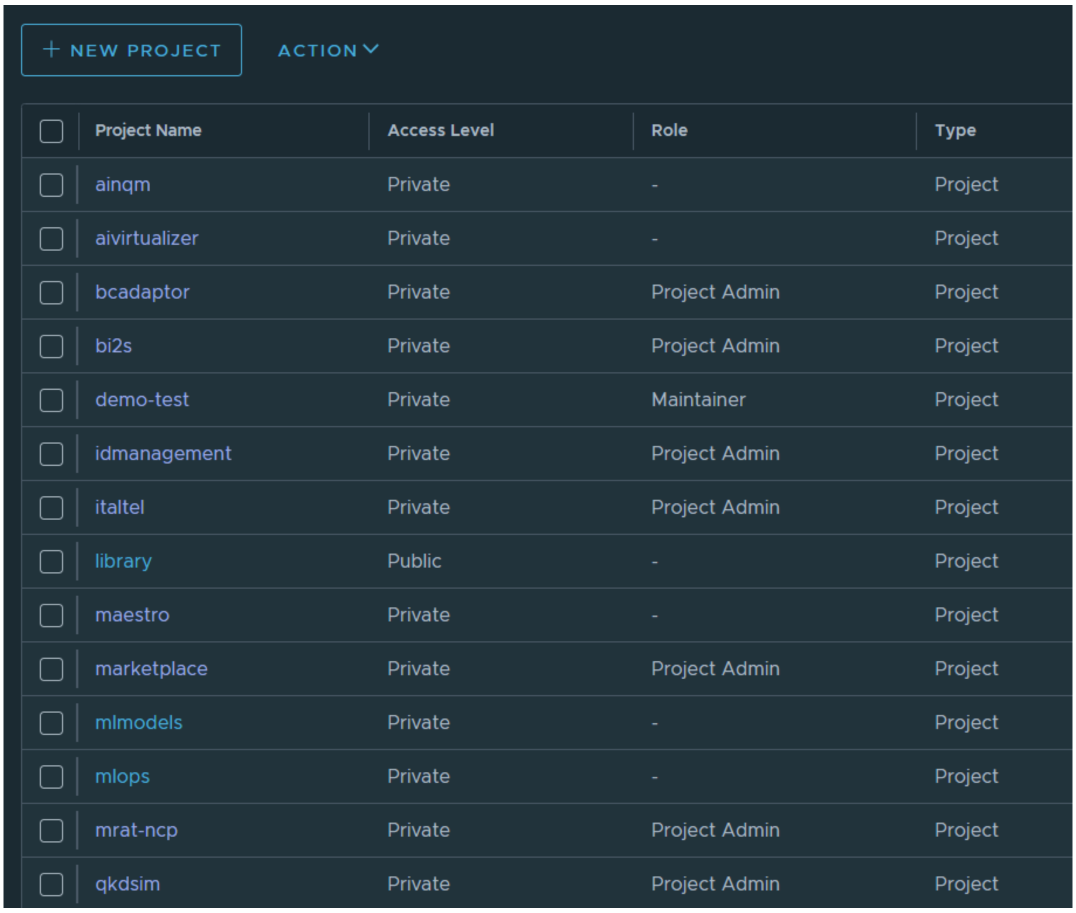
Figure 18 Wedhooks

4.2.4. Harbor Private Docker Container Registry

Harbor is an open-source project that serves as a container image registry, offering advanced management and security features for Docker images. It builds upon the fundamental functionalities of a container registry by incorporating elements such as role-based access control (RBAC), vulnerability scanning, and image signing and verification. Moreover, Harbor's architecture is paired with an intuitive web interface, simplifying the management and tracking of container images.

A private Docker Registry has been set up, allowing users to push and pull Docker images. It employs SSL encryption and user authentication. The URL of the Docker Registry is the following: https://harbor.nancy.rid-intrasoft.eu/

Different projects have been created in NANCY Harbor registry to associate with the different NANCY dockerised components (similarly to Jenkins CI server as described in section 4.2.3) (see Figure 19).



Project Name	Access Level	Role	Type
ainqm	Private	-	Project
aivirtualizer	Private	-	Project
bcadaptor	Private	Project Admin	Project
bi2s	Private	Project Admin	Project
demo-test	Private	Maintainer	Project
idmanagement	Private	Project Admin	Project
italtel	Private	Project Admin	Project
library	Public	-	Project
maestro	Private	-	Project
marketplace	Private	Project Admin	Project
mlmodels	Private	-	Project
mlops	Private	-	Project
mrat-ncp	Private	Project Admin	Project
qkdsim	Private	Project Admin	Project

Figure 19 NANCY component-specific projects in Harbor private registry

Harbor accounts have been provided to the technical partners with access to the GUI and permissions to push and pull images for specific projects related to their components and developments. Each partner can create different registry repositories within their assigned projects to host their Docker images.

The Harbor GUI provides the following functionalities:

- Securely view different versions of Docker images.
- View the history of each image.
- Delete unnecessary tags.

A retention policy has been implemented to keep the latest 5 Docker image tags per project, per repository. This policy is automatically executed every hour. It is strongly recommended that partners push their Docker images to the private Docker registry before deployment. The stored images can then be pulled and deployed in either the development or operational environments of NANCY (Kubernetes clusters at testbeds/demonstrators). Users can push and pull images either through Jenkins (automated pipeline) or from their own remote hosts.

4.3. NANCY Central Kubernetes Cluster

Kubernetes has become the predominant standard for deploying containerized applications at scale across private, public, and hybrid cloud environments. This open-source platform automates the deployment, scaling, and management of containerized applications. Its features, including pod utilisation, clustering, load balancing, and horizontal scaling, enable dynamic scaling for software components, microservices, and containers.

Within the NANCY project, Kubernetes was chosen as the cloud-native infrastructure to support the development, staging, and operational environments of the platform. This section will detail the architecture of the central NANCY Kubernetes cluster specifically used for development, staging, and for hosting part of the central management services of the platform (Section 3.2.1), along with an in-depth explanation of its software components and functionalities.

4.3.1. High Level Architecture

A Kubernetes cluster has been established to host the NANCY software components and manage the deployment, scaling, and orchestration of the NANCY containerized applications within the development/testing and part of the operational central management execution environments envisioned for NANCY. This is a highly available (HA) and fault-tolerant cluster consisting of:

- Three master nodes managing the entire control plane of the cluster.
- Three worker nodes hosting the containerized applications of the NANCY Platform.
- One load balancer node to efficiently distribute the control-plane load among the master nodes using round-robin scheduling. An HA Proxy has been deployed in front of the master nodes in the NANCY cluster for this purpose.

The different execution environments of NANCY (e.g., development/testing, central management) are isolated through the use of dedicated Kubernetes namespaces. Communication is secured over HTTPS within the cluster, and all Kubernetes cluster data is stored in the ETCD distributed reliable key-value store. ETCD ensures that the cluster data is consistent across all nodes. The following figure illustrates the stacked ETCD architecture topology implemented for NANCY:

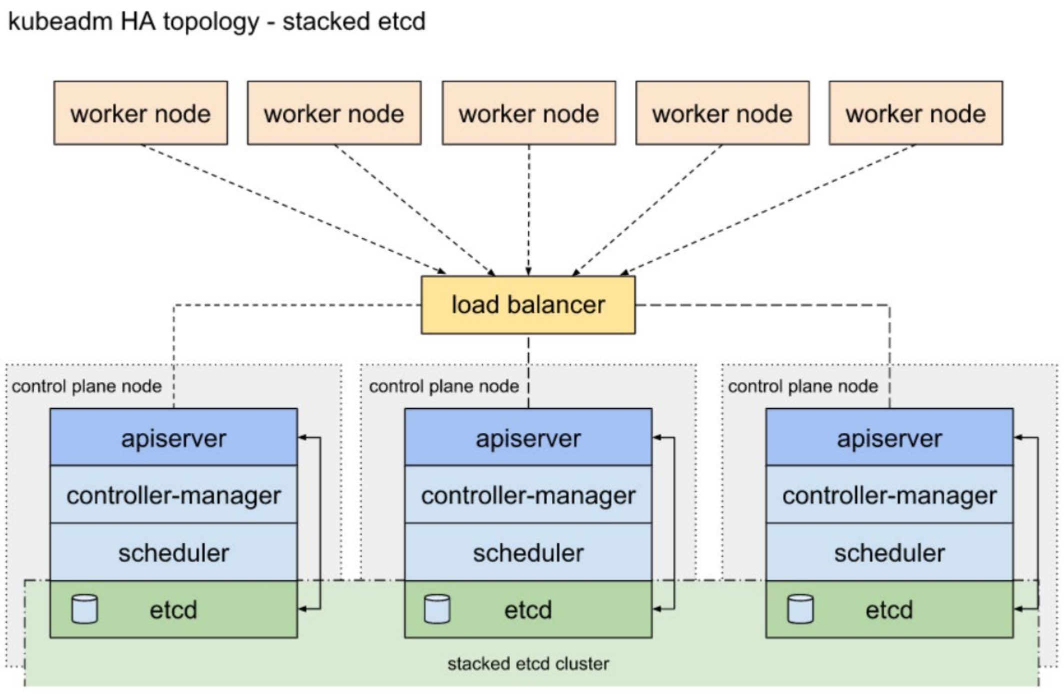
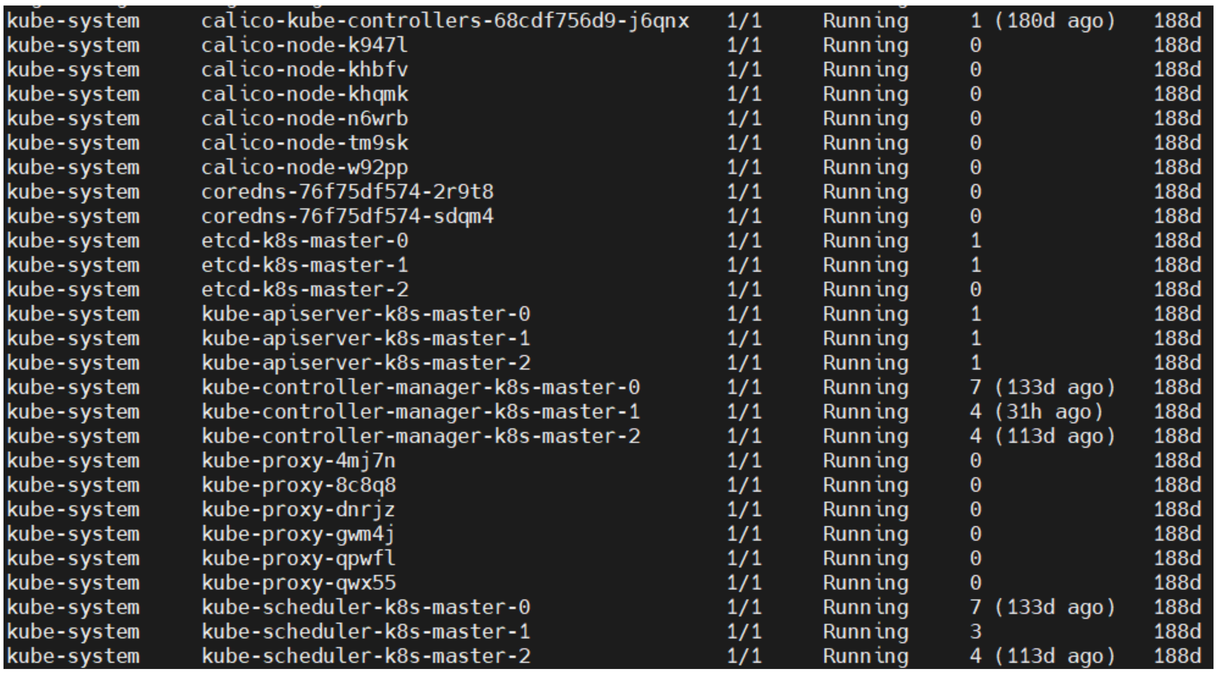


Figure 20 NANCY central Kubernetes cluster architecture[1]

In sections 4.3.2 and 4.3.3 some of the main Kubernetes system components running in the master and worker nodes (as shown in Figure 21) are presented.



```
kube-system     calico-kube-controllers-68cdf756d9-j6qnx   1/1     Running     1 (180d ago)   188d
kube-system     calico-node-k947l                          1/1     Running     0              188d
kube-system     calico-node-khbfv                          1/1     Running     0              188d
kube-system     calico-node-khqmk                          1/1     Running     0              188d
kube-system     calico-node-n6wrb                          1/1     Running     0              188d
kube-system     calico-node-tm9sk                          1/1     Running     0              188d
kube-system     calico-node-w92pp                          1/1     Running     0              188d
kube-system     coredns-76f75df574-2r9t8                   1/1     Running     0              188d
kube-system     coredns-76f75df574-sdqm4                   1/1     Running     0              188d
kube-system     etcd-k8s-master-0                          1/1     Running     1              188d
kube-system     etcd-k8s-master-1                          1/1     Running     1              188d
kube-system     etcd-k8s-master-2                          1/1     Running     0              188d
kube-system     kube-apiserver-k8s-master-0                1/1     Running     1              188d
kube-system     kube-apiserver-k8s-master-1                1/1     Running     1              188d
kube-system     kube-apiserver-k8s-master-2                1/1     Running     1              188d
kube-system     kube-controller-manager-k8s-master-0       1/1     Running     7 (133d ago)   188d
kube-system     kube-controller-manager-k8s-master-1       1/1     Running     4 (31h ago)    188d
kube-system     kube-controller-manager-k8s-master-2       1/1     Running     4 (113d ago)   188d
kube-system     kube-proxy-4mj7n                           1/1     Running     0              188d
kube-system     kube-proxy-8c8q8                           1/1     Running     0              188d
kube-system     kube-proxy-dnrjz                           1/1     Running     0              188d
kube-system     kube-proxy-gwm4j                           1/1     Running     0              188d
kube-system     kube-proxy-qpwfl                           1/1     Running     0              188d
kube-system     kube-proxy-qwx55                           1/1     Running     0              188d
kube-system     kube-scheduler-k8s-master-0                1/1     Running     7 (133d ago)   188d
kube-system     kube-scheduler-k8s-master-1                1/1     Running     3              188d
kube-system     kube-scheduler-k8s-master-2                1/1     Running     4 (113d ago)   188d
```

Figure 21 Instances of Kube-system components in central NANCY Kubernetes cluster

4.3.2. Control Plane Master Nodes

The control plane components within a Kubernetes cluster are assigned to making overarching decisions, including scheduling tasks and managing cluster events. These components, which function on the master nodes, are detailed as follows:

[1] Figure from https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/ha-topology/

API Server

The API Server serves as the gateway to the Kubernetes API and functions as the interface for the Kubernetes control plane. Its primary roles include managing incoming requests, processing them, and providing cluster information. This component can scale horizontally. For instance, as shown in Figure 17, the API Server is distributed across three instances.

With three active API Server instances, a load balancer is necessary to distribute traffic among the master nodes. The kubectl utility is then configured to route requests to this load balancer.

ETCD distributed reliable key-value store

ETCD is a distributed and reliable key-value store that maintains the data for the Kubernetes cluster. For the NANCY platform, a highly available configuration has been implemented, featuring three ETCD instances, each associated with one of the three master nodes. Each ETCD instance communicates exclusively with the API Server, Controller Manager, and Scheduler on its corresponding node. ETCD guarantees data consistency across all master nodes by synchronizing information among all instances. This synchronisation is managed by a leader node, which is elected to handle replication tasks for the other nodes in the cluster.

Kubernetes controller manager

The Kubernetes Controller Manager is a daemon that operates as an ongoing control loop within a Kubernetes cluster. It checks the cluster's current state by communicating with the API Server and works to match this state with the intended configuration outlined in the cluster's declarative setup. It manages multiple controllers, each dedicated to specific functions, such as the replication controller, endpoints controller, namespace controller, and service accounts controller.

Scheduler

The Scheduler is a control plane component that oversees newly added Pods and assigns them to the most appropriate node for execution. Several factors can influence the Scheduler's decisions, including affinity rules, hardware and software limitations, and other constraints.

4.3.3. Worker Nodes

Kubelet

Kubelet is a daemon that runs on each node (whether master or worker) in the cluster. Its role is to ensure that the containers within a Pod are running as expected and to report any issues to the API server and scheduler if any intervention is needed.

Kube-proxy

Kube-proxy is a network proxy running on every node (master or worker) in the cluster. It handles network rules on worker nodes, facilitating communication with Pods from both internal and external network connections.

Containerd

The containerd daemon is employed on each worker node to manage the entire container lifecycle on the worker host.

4.3.4. Kubernetes Networking

Kubernetes supports the Container Network Interface (CNI) specification for handling network resources within a cluster. For NANCY, Calico [20] was selected as the CNI. Kubernetes employs the Calico CNI plugin to manage the details of pod connectivity to the underlying network infrastructure.

Calico connects pods to the host network using L3 routing, eliminating the need for an L2 bridge. This method is simpler and more efficient compared to many other common alternatives.

4.3.5. Kubernetes Namespaces

In Kubernetes, namespaces offer a way to isolate groups of resources within a single cluster. Resource names must be unique within a namespace but can be duplicated across different namespaces. Namespaces are designed for environments with numerous users distributed across multiple teams or projects. In the context of NANCY, namespaces are also utilised to ensure resource isolation among development, staging, and operational (relevant to NANCY demonstrators) environments within the project's central Kubernetes cluster.

4.3.6. External Access to the Kubernetes Cluster

To facilitate remote access to the deployed services in the NANCY Kubernetes cluster, an ingress controller and an external load balancer from Hetzner have been configured, as illustrated in Figure 22.

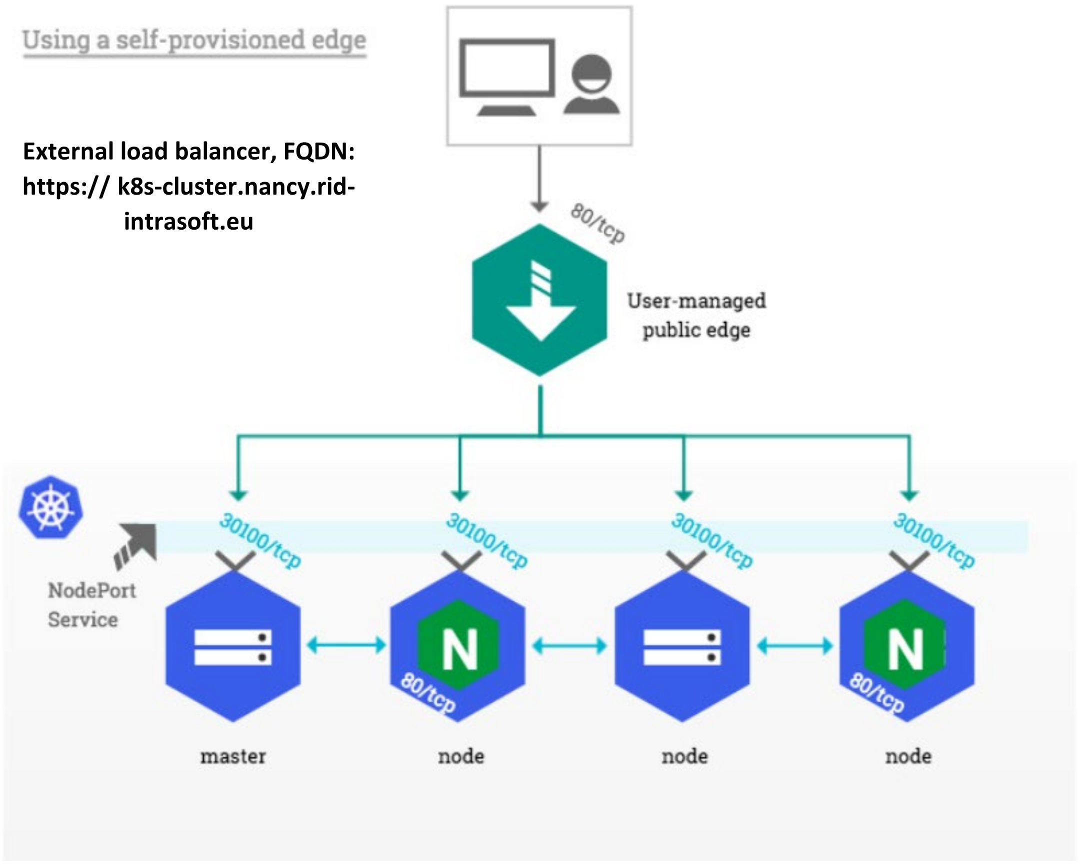


Figure 22 Setup for external access of the deployed services in the Kubernetes cluster[2]

The load balancer is responsible for evenly distributing traffic across the ingress controller instances (pods) deployed on the three different worker nodes. It also maps port 80 (HTTP) and port 443 (HTTPS) to the NodePort where the ingress controller service is accessible. Ingress exposes HTTP and HTTPS routes from outside the cluster to NANCY services within the cluster using Fully Qualified Domain Names (FQDN). Specific ingress resources can be configured for each deployed application to define routing rules (Figure 23), providing the corresponding application services with externally reachable

[2] Figure from: https://kubernetes.github.io/ingress-nginx/deploy/baremetal/#using-a-self-provisioned-edge

URLs under any NANCY-specific subdomain (e.g., k8s-cluster.nancy.rid-intrasoft.eu for applications deployed in the development/testing and operational central management environment of the Kubernetes cluster).

A Kubernetes cluster does not have an ingress controller enabled by default. In the NANCY setup, the NGINX Ingress controller [21] was enabled.

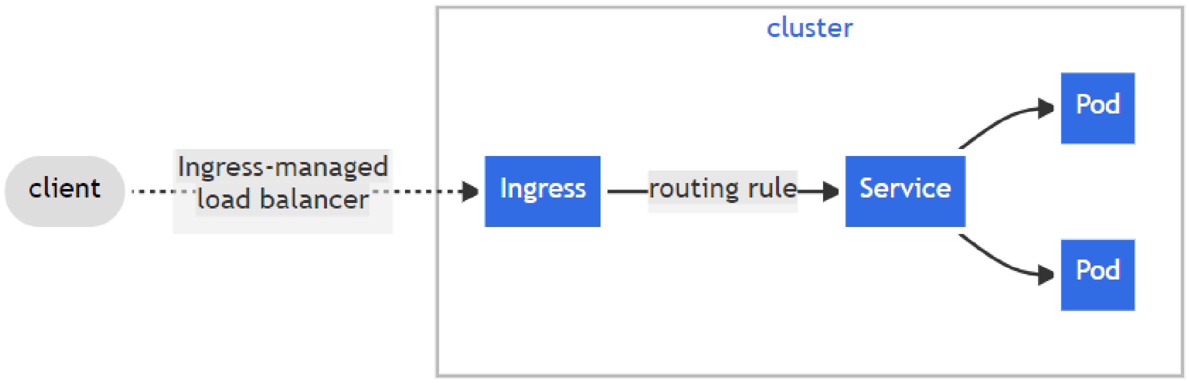
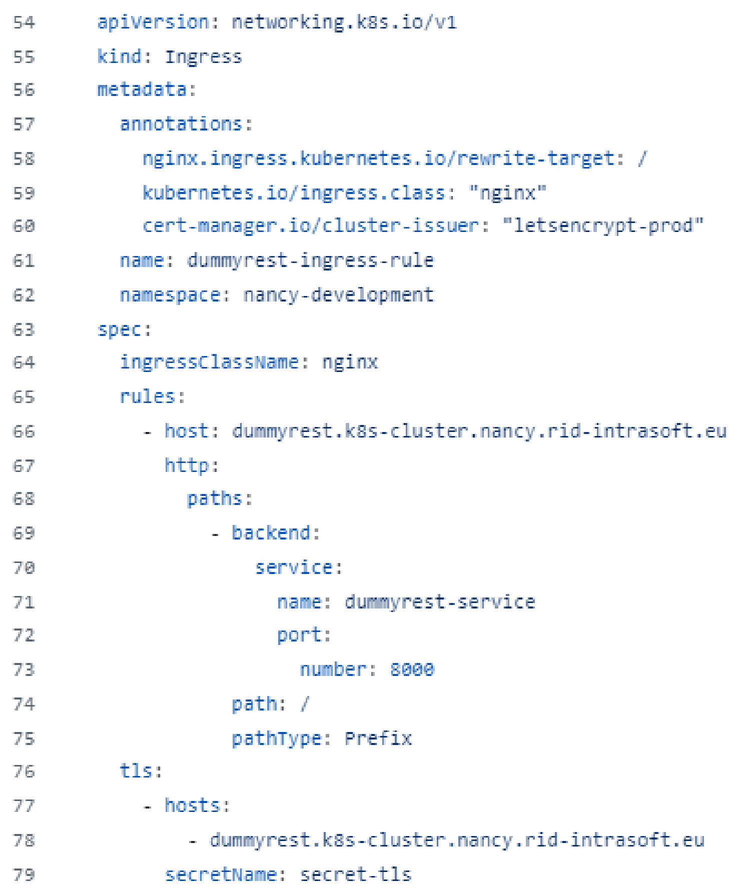


Figure 23 Ingress routing rules enabling external access to deployed services in the Kubernetes cluster[3]

Ingress resources configuration

To establish specific routing rules that provide the application services with externally accessible URLs under the k8s-cluster.nancy.rid-intrasoft.eu domain, an ingress resource using a YAML file needs to be configured. An example of such an ingress resource is provided in Figure 24. In this example, the ingress resource is set up to route HTTP traffic based on the hostname (name-based virtual hosting). According to this configuration, our demo application service is accessible at: https://dummyrest.k8s-cluster.nancy.rid-intrasoft.eu.

[3] Figure from https://kubernetes.io/docs/concepts/services-networking/ingress/



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    kubernetes.io/ingress.class: "nginx"
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
  name: dummyrest-ingress-rule
  namespace: nancy-development
spec:
  ingressClassName: nginx
  rules:
    - host: dummyrest.k8s-cluster.nancy.rid-intrasoft.eu
      http:
        paths:
          - backend:
              service:
                name: dummyrest-service
                port:
                  number: 8000
            path: /
            pathType: Prefix
  tls:
    - hosts:
        - dummyrest.k8s-cluster.nancy.rid-intrasoft.eu
      secretName: secret-tls
```

Figure 24 Example of Ingress resource configuration for name based virtual hosting

Securing ingress

To ensure secure access to the services via ingress over HTTPS, the following components have been deployed and configured in the NANCY cluster:

- A cert-manager [22] to request TLS certificates from the Let’s Encrypt Certificate Authority (CA), automatically renew them, and manage their usage.
- A Let’s Encrypt cluster issuer (operating across the namespaces of the cluster) that defines how the cert-manager requests TLS certificates from the CA.
- Properly configured TLS ingress resources for each deployed application requiring external access to initiate the request for a TLS certificate (as shown in Figure 24).

4.4. Security Features of the NANCY Integration Environment

A collection of security features has been integrated into the CI/CD solution to protect the CI/CD infrastructure and services, along with the deployed project's artifacts. The subsequent sections offer a comprehensive overview of these security measures.

4.4.1. Secure Communication over HTTPS

Hypertext Transfer Protocol Secure (HTTPS) is used to ensure secure user connections to the deployed applications. Specifically, the continuous integration server (Jenkins) and the container image registry (Harbor) are secured with HTTPS. Data transmitted via HTTPS is safeguarded by the Transport Layer Security (TLS) protocol, which provides three essential layers of protection:

- Encryption: Exchanged data is encrypted to prevent eavesdropping. This ensures that when users access an HTTPS-secured service, their communications cannot be intercepted, their activities cannot be tracked, and their information remains secure.
- Data Integrity: It is guaranteed that data cannot be altered or corrupted during transfer, whether accidentally or maliciously, without detection.
- Authentication: It is verified that users are communicating with the intended service, protecting against man-in-the-middle (MitM) attacks and fostering user trust.

Let’s Encrypt CA has been utilised for the issuance of X.509 certificates, providing TLS encryption to secure the CI/CD services. Additionally, the certificates for the CI/CD services are configured for automatic renewal through a cron job.

As detailed in section 4.3.6, HTTPS is also used to secure the access to the deployed NANCY services within the Kubernetes cluster. Similarly to the CI/CD services, Let’s Encrypt is used as the CA for the automatic issuance and renewal of TLS certificates, facilitated by the relevant Kubernetes resources described in section 4.3.6.

4.4.2. User Authentication and Role-based Access Control

The services offered in the NANCY CI/CD environment, namely the source code Github organisation, the continuous integration server (Jenkins) and the container image registry (Harbor) are secured using user authentication and role-based access control (RBAC). More specifically Keycloak open-source identity and access management solution [23] has been installed to access the CI/CD services, providing Single-Sign-On (SSO) (see Figure 25). It enhances security for Jenkins and Harbor by centralizing authentication, reducing credential exposure, and enforcing consistent security policies like role-based access control (RBAC) towards the CI/CD services. RBAC is configured across Keycloak and the CI/CD services and it is used to ensure that all the technical component providers have access only to their own and any agreed collaborative development and testing projects, where different access levels can be configured.

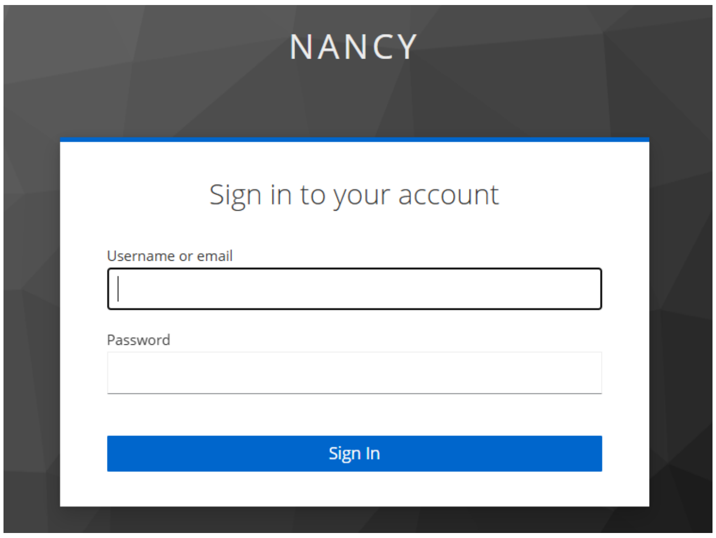
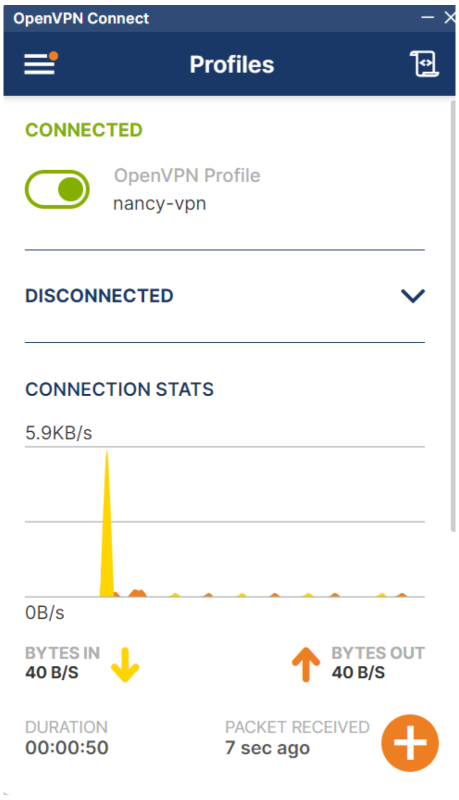


Figure 25 SSO access to NANCY CI/CD services via Keycloak

4.4.3. VPN Configuration

An OpenVPN server has been put in place enabling NANCY component providers to access the deployed applications and services in the Kubernetes cluster hosted on the central Hetzner cloud infrastructure. This VPN setup uses a PFSense software firewall [24]. By utilizing the OpenVPN server, external partners can connect via a private and encrypted VPN tunnel to reach these services.

For each user requiring access to this infrastructure, a corresponding SSL/TLS certificate is generated and distributed. With this certificate and their credentials, users can authenticate to the NANCY VPN through the OpenVPN client application (Figure 26).



OpenVPN Profile
nancy-vpn

BYTES IN
40 B/S

BYTES OUT
40 B/S

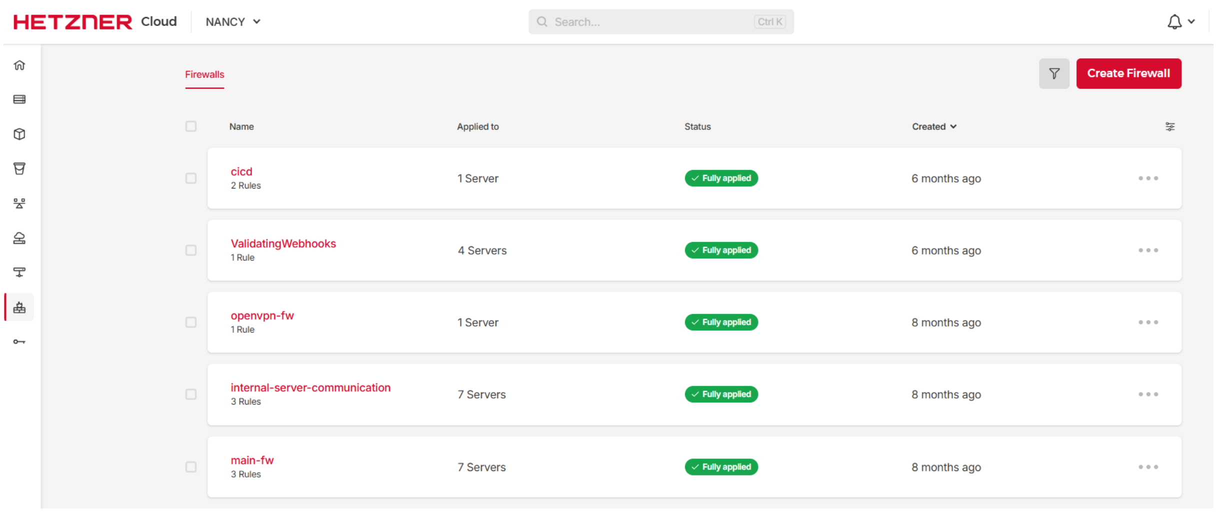
DURATION
00:00:50

PACKET RECEIVED
7 sec ago

Figure 26 Access to central NANCY VPN

4.4.4. Firewall Protection of CI/CD and Central Management Infrastructure

The central NANCY VM infrastructure is protected from a set of firewalls under the Hetzner cloud (Figure 27) to restrict access to authorised external IPs/subnets and ports only, and through the NANCY VPN, while allowing internal server communication among the NANCY servers.



Name	Applied to	Status	Created
cicd 2 Rules	1 Server	✓ Fully applied	6 months ago
ValidatingWebhooks 1 Rule	4 Servers	✓ Fully applied	6 months ago
openvpn-fw 1 Rule	1 Server	✓ Fully applied	8 months ago
internal-server-communication 3 Rules	7 Servers	✓ Fully applied	8 months ago
main-fw 3 Rules	7 Servers	✓ Fully applied	8 months ago

Figure 27 Hetzner firewalls to restrict access to Hetzner VMs

4.4.5. SSH-key-based Authentication to the Central NANCY Infrastructure

For administering and managing the NANCY VM infrastructure servers, SSH access is configured to use key-based authentication only. This is a more secure approach than the traditional password authentication approach. Although passwords are securely transmitted to the server, they can often lack sufficient complexity or length, making them vulnerable to persistent attacks. Instead, SSH public key authentication involves generating and storing a pair of cryptographic keys and configuring the servers to accept these keys. Consequently, it provides a more robust security solution.

5. Functional Testing of NANCY Components

Table 3 summarizes the functional tests for all the NANCY software components. The naming convention for the test identifiers is as follows: *COMPONENTX_F00x*, where *COMPONENTX* represents the acronym of the respective component and *F00x* is the sequential number of functional tests for that component. The Objective column provides a brief description of each test and its purpose.

Table 3: NANCY functional testing

Component	Test Identifier	Test Objective
MRAT-NCP	MRAT-NCP_F001	Test PC5 link reliability and requirements.
	MRAT-NCP_F002	Test tandem connection for PC5 and 5G connectivity.
	MRAT-NCP_F003	Test intra-network connectivity to compute nodes.
ID_Mgnt	ID_Mgnt _F001	Test key derivation and generation of credentials.
	ID_Mgnt _F002	Test Configuration of the wallet.
	ID_Mgnt _F00x	Test verification of credentials.
DAC	DAC _F001	Test API endpoint for creating smart contracts with valid input.
	DAC _F002	Test API endpoint with invalid or missing input data.
	DAC_F003	Test API response time and ensure it meets performance criteria.
BC	BC_F001	Adding new user: The network administrator uses their admin identity to register a new user with the CA. This step associates the new user with a specific role and organisation.
	BC _F002	Smart contract unit tests: Test the functionalities in each smart contract
Wallet	Wallet_F001	SSI capabilities: User DID creation and registration: Start the wallet gateway provided with a specified uid. A unique and anonymous DID is created for this uid and a certificate corresponding to the created DID is acquired from the CA of this organisation. The public DID document is registered to the DIDRegistry smart contract in the blockchain, so that other users are able to look up for the verification methods of this DID.
	Wallet _F002	SSI capabilities II: acquire and verify verifiable credential (Provided Wallet_F001) Start a wallet gateway on an ID holder, a credential issuer, and a service verifier respectively. The holder acquires a verifiable credential from the issuer by means of their wallets, then the holder wallet generates a verifiable presentation from the acquired verifiable credential and delivers it to the verifier's wallet service for verification. The verification returns successfully.
	Wallet _F003	SSI capabilities III: revoke the verifiable credential

Component	Test Identifier	Test Objective
		Same setup as in Wallet_F002, but the issuer revokes the previous verifiable credential issued to the holder. The last verification step then fails.
	Wallet _F004	Integration with the smart contract marketplace: (Provided Wallet_F001) Test all wallet interfaces related to marketplace operations to manage providers, services and searches.
	Wallet _F005	Integration with PQC: (Provided Wallet_F001) Unit test for the correctness of the wallet PQC signing function and the verification function on SLARegistry chaincode.
	Wallet _F006	Integration with the smart contract SLA: (Provided Wallet_F001 and Wallet_F005) Test wallet *signSLA* interface to see if the SLA is correctly signed and updated (also with PQC signature) and if the subscribers for SLASigning events get notified. Test wallet *getSLA* and *getSLAByConsumerId* to look up SLAs in the blockchain.
	Wallet _F007	Integration with the oracle: (Provided Wallet_F001, Wallet_F004 and Wallet_F006) Test wallet createSearch interface to see if the oracle triggers SLA creation correctly and if the subscriber for SLASigning events get notified.
AI-virtualizer	AI-virtualizer _F001	Test the Kafka bus communication between agents.
	AI-virtualizer _F002	Test the Slice Manager API response to IJS remote calls.
	AI-virtualizer _F003	Test Grafana visualisation.
BRAN-model	BRAN-model _F001	Test the B-RAN model estimated performance for various system configurations.
	BRAN-model _F002	Test the functionality of the B-RAN model for various extreme cases.
SemCom	SemCom_F0 01	Test SemCom performance for V2x and DT creation.
	SemCom _F002	Test SemCom energy and data efficiency improvement for ASL transmission.
QKDSim	QKDSim _F001	Test the deployment and communication with the dockerised QKDSim.
	QKDSim _F002	Test the functionality of QKDSim for various extreme cases
VOSySmonito r	VOSySmonito r _F001	Test vManager partitions operations (create,deploy, destroy etc.).
	VOSySmonito r _F002	Test availability of virtio devices inside the partitions for deploying VNFs.
	VOSySmonito r _F003	Test OPTEE Secure Storage service functions (store, load sensitive content).
Marketplace	Marketplace _F001	Test connection with SP.
	Marketplace _F002	Test connection with DAC.

Component	Test Identifier	Test Objective
	Marketplace_F003	Test connection with Blockchain wallet (receive requests).
	Marketplace_F004	Test connection with other smart contracts.
Maestro	Maestro_F001	Test connection with Resource Orchestrator (NIS2).
	Maestro_F002	Test connection with Compute Controller (NIS6).
	Maestro_F003	Test connection with service Telemetry (NIS5).
	Maestro_F004	Test compute enforcement APIs via Compute Controller (NIS1).
	Maestro_F005	Test connection with Service Repository/Registry (NIS3).
	Maestro_F006	Test Connection with BSS/Marketplace (NIS1).
Models	Models_F001	Testing ML algorithms to make prediction on the location of the users, providing Localisation as a Service (LaaS).
	Models_F002	Testing ML algorithms to detect anomaly in the signals with key network performance indicators.
	Models_F003	To assess the spectrum occupancy with ML-based Radio spectrum sensing techniques.
SEMR	SEMR_F001	Testing on NAOMI [25] for AI workflow orchestration.
Elasticity	Elasticity_F001	Testing the developed computing resource elasticity techniques.
PQCSig	TC_NE_C_SignInit::testCKM_ML_DSA	Initialisation of Signature sequence.
	TC_NE_C_Sign::testCKM_ML_DSA	Signature creation and verification using ML_DSA keys using the following PKCS#11 functions: C_SignInit C_Sign C_VerifyInit C_Verify Returns CKR_MECHANISM_INVALID if mechanism is not supported.
	TC_NE_C_SignUpdate::testCKM_ML_DSA	Signature update and verification of multiple-part signature operation using ML_DSA keys using the following PKCS#11 functions: C_SignInit C_SignUpdate C_SignFinal C_VerifyInit C_Verify Returns CKR_MECHANISM_INVALID if mechanism is not supported.

Component	Test Identifier	Test Objective
	TC_NE_C_SignFinal::testCKM_ML_DSA	Signature finalisation of a multiple-part signature operation using ML_DSA keys. see TC_NE_C_SignUpdate
	TC_NE_C_VerifyInit::testCKM_ML_DSA	Initialisation of Verification sequence using C_VerifyInit function using ML_DSA keys. Returns CKR_MECHANISM_INVALID if CKM_ML_DSA is not supported.
	TC_NE_C_Verify::testCKM_ML_DSA	Signature single-part data verification for CKM_ML_DSA keys. See TC_NE_C_Sign
	TC_NE_C_VerifyUpdate::testCKM_ML_DSA	Signature multiple-part verification using ML_DSA keys using the following PKCS#11 functions: C_SignInit C_Sign C_VerifyInit C_VerifyUpdate C_VerifyFinal Returns CKR_MECHANISM_INVALID if mechanism is not supported.
	TC_NE_C_VerifyFinal::testCKM_ML_DSA	Signature verification finalisation for a multiple-part verification operation using ML_DSA. See TC_NE_C_VerifyUpdate.
TFS	TFS_F001	Throughput forecasting intended for network analytics purposes.
	TFS_F002	Throughput forecasting for assessment of upcoming network performance and mitigation of anticipated degradation
RICMngr	RICMngr_F001	Successful communication with the Slice Manager via the defined interface.
	RICMngr_F002	Successful communication with the near-RT RIC via A1 interface.
	RICMngr_F003	Successful implementation of the required control-loop workflow.
AINQM	AINQM_F001	Prediction of network outage probability for analytics.
	AINQM_F002	Prediction of network outage probability to anticipate network events and support decision-making.
NIF	NIF_F001	AI model testing with synthetic data to ensure reasonable predictions.
	NIF_F002	Ensure the AI model's loss function converges without overfitting.
	NIF_F003	Successful creation of users and data upload process.
SPP	SPP_F001	Simulation environment testing (Boundary conditions, auction rules etc) with synthetic data.
	SPP_F002	Ensure the neural network loss function converges without overfitting.
	SPP_F003	Performance testing under load to measure and improve latency and throughput.
	SPP_F004	Reinforcement learning model testing with synthetic data to ensure reasonable behavior.
XAI	XAI_F001	Discrete Validation of XAI methods in the context of anomaly detection and cyber-attack identification.
	XAI_F002	Testing of XAI Visualisation Dashboard.

Component	Test Identifier	Test Objective
	XAI_F003	Testing XAI in a Greek in-lab testbed simulates real-world conditions to ensure explanations are clear, relevant, and trustworthy for users and administrators.
FL-IDS	FL-IDS_F001	Successful Communication between distributed clients.
	FL-IDS_F002	FL Training for Intrusion Detection in Greek In-lab Testbed utilising 3 clients and a server.
MTG-RM	MTG-RM_F001	Demonstrate capability of generating intense memory traffic to simulate malicious behavior.
	MTG-RM_F002	Monitor the system resource usage at the CPU level as provided by Performance Measurement Units.
	MTG-RM_F00x	Limit the CPU resources assigned to malicious application.
PQC-SC	PQC-SC_F001	Successful integration of PQC algorithms into OpenSSL library.
	PQC-SC_F002	Successful communication using TLS of a specific application (e.g. MQTT).
D-ADM	D-ADM_F001	Application and validation of the automatic anomaly detection methodology within a testbed with data from malicious server applications.
	D-ADM_F002	Testing of anomaly compensation techniques in a testbed for load balancing purposes.
ETSI OpenSlice	OSL_F001	Test connection with Service Orchestrator (NIS2).
	OSL_F002	Test connection with Service Repository/Registry (NIS3).
	OSL_F003	Test connection with resource Telemetry (NIS5).
	OSL_F004	Test connection with Compute Controller (NIS6).
	OSL_F005	Test connection with NFVO (SOL005).

6. Integration of NANCY Components and Services

6.1. Integration Timeline

The development of the NANCY integrated platform follows an iterative process, leading to two major releases as specified in the integration timeline provided in Table 4 (bold indicates a large period of integration activities which is split into smaller phases/milestones explained in the following lines). This timeline closely follows the integration methodology specified in section 2. It follows CI/CD best practices to minimize errors during the integration and deployment stages of NANCY software components and to facilitate software release for operational purposes (availability for NANCY testbeds and demonstrators).

Table 4 NANCY integration timeline

Iteration	Integration activities	Partners	Date
Init	Collection of HW/SW component-specific requirements to be considered for the NANCY integration environment	All WP2-WP5 tech. component providers	M16-M17
	Main development phase of NANCY software components including their unit, functional and initial bilateral integration tests	**All WP2-WP5 tech. component providers**	**M07-M25**
	Initial integration points identification	All WP2-WP5 tech. component providers	M17-M18
	Definition of NANCY architecture's functional and deployment view (incl. NANCY interfaces definition).	UBI, INNO, IJS, UMU, NEC, INTRA, WP6 testbed leaders	M18-M19
	Installation of CI/CD platform and central Kubernetes cluster supporting development, staging and (partially) operational environments for NANCY. Provision of training workshop and accompanying material to the NANCY tech. component providers on the usage of the CI/CD platform	INTRA	M20-M21
	Updates in integration points identification; First release of integration points specifications,	All WP2-WP5 tech. component providers	M22-M24

	functional and integration testing activities		
First, early release of NANCY Integrated Framework	Initial release of NANCY integrated platform incl. the CI/CD integration framework, NANCY integration points and testing specifications	INTRA, UBI, All WP2-WP5 tech. component providers	M25
	Development updates, functional, integration and system (end-to-end) level tests, using the NANCY CI/CD system, central development/staging and remote operational environments (testbeds and demonstrators)	**All WP2-WP5 tech. component providers**	**M25-M34**
	Secure connection of Central Management (incl. CI/CD, Service & Resource orchestrators etc.) and Inter-Operator domains (incl. Blockchain fabric) with remote NANCY testbeds/ demonstrators and initial testing/verifications	INTRA, UBI, NEC, WP6 testbed / demonstrator leaders	M26-M28
	Main testing/verification phase at NANCY testbeds/ demonstrators, including the integration of NANCY dockerised components; Provision of feedback/recommendations to NANCY tech. component providers for improvements	INTRA, UBI, WP2-WP5 tech. component providers and WP6 testbed/ demonstrator leaders	M26-M31
	Bilateral and small-scale integration testing	WP2-WP5, WP6 testbed/ demonstrator leaders	M26-M29
	Support for Larger scale (system-level) testing, relevant to NANCY operational use cases for each demonstrator	WP2-WP5, WP6 testbed/ demonstrator leaders	M28-M31
Second release of NANCY Integrated Framework	Final release of NANCY integrated platform, incl. updates in integration points specifications and testing validations	INTRA, All WP2-WP5 tech. component providers	M33

6.2. Integration Points Identification

Integration points consist of pairs of NANCY components that interact with each other via well-defined interfaces, exchanging data and information according to predefined sequence diagrams.

To document and identify the integration points for the NANCY platform, an integration matrix using a format similar to a Design Structure Matrix (DSM [26]) has been created. DSM is used extensively in systems integration for modelling the interactions among complex system structures. It is a square matrix that visually displays connections between system elements, typically indicated on the rows to the left and the columns above the matrix. These elements might include, for example, the components that make up a system. Similar to an adjacency matrix in graph theory, the DSM is employed in systems engineering and project management for modelling complex systems or processes, conducting system analysis, and planning and organizing projects.

In this context, the integration matrix (Table 5) includes the software components of the NANCY platform developed in the context of WP2-WP5 and indicates the bilateral communication among them. Each software component from the NANCY platform is represented by a row and a column. This is an upper triangular square matrix, where each entry indicates an integration point. The naming convention applied to each integration point is X.Y, where X and Y denote the respective row and column of the software components involved.

Table 5 NANCY integration matrix

Integration Points	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1. MRAT-NCP		1,2						1,8					1,13														
2. ID-Mngt					2,5					2,10																	
3. DAC											3,11																
4. BC					4,5						4,11																
5. Wallet											5,11	5,12															5,27
6. AI Virtualizer										6,10				6,14													
7. BRAN-model																											
8. SemCom																											
9. QKDSim																											
10. VOSyS Monitor																											
11. Marketplace																					11,21						
12. Maestro																											
13. Models														13,14	13,15		13,17										
14. SEMR																			14,19				14,23				
15. Elasticity																											
16. PQC Sign.																									16,25		
17. TFS																											
18. RIC-mngr																											
19. AINQM																				19,20		19,22					
20. NIF																											
21. SPP																											
22. XAI																							22,23				

Integration Points	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
23. FL-IDS																											
24. MTG-RM																											
25. PQC-SecCom																											
26. D-ADM																											
27. BSS																											

The identified integration points illustrated in Table 5 are specified in detail in Annex A. Table 6 provides the summaries of these specifications for each identified integration point, covering their *objectives*, *protocol/interface type* used for their exchanges and *implementation status.*

Table 6 Integration points specification summary

NANCY Integration Points	NANCY Platform Interface	Objective	Protocol	Integration in testbeds/demonstrators	Status
A.1 MRAT-NCP – ID Management(1,2)	NIS13, Uu, PC5	Allow access for 5G network usage for a remote non-5G subscriber.		Spanish expansion	Ongoing
MRAT-NCP – SemCom (1,8)	Uu, S1, NIS7, NIS5, NIS8	Send encoded video and decode at the edge	Offline data transfer	Spanish expansion data	Not started
MRAT-NCP – Models (1,13)	Uu, S1, NIS7, NIS5, NIS8	Infer location depending on radio metrics extracted from the UE. Throughput forecast based on radio metrics extracted from the UE.		Spanish expansion	Ongoing
ID Mngnt – Wallet (2,5)	N/A	Configure the wallet with NANCY ID credentials, use credential to authenticate towards NANCY services.	REST API	Spanish inlab testbed	Ongoing
ID Mngnt – VoSyS Monitor (2,10)	N/A	Instantiate OP-TEE solution[4] for caching mechanisms on ARM-based far-edge device included in the MRAT-NCP. This will be used to securely store the access token of NANCY subscriber to accelerate authorisation times.	API based on Global Platform TEE Client API	Spanish inlab testbed	Ongoing
DAC – Marketplace (3,11)	NI11	• Provide the required information for creating the digital agreement. • Receive the generated digital agreement to be signed	REST API over HTTPS	All	Completed
BC – Wallet (4,5)	NIS13	Once the identity is in the wallet, the user connects to the Fabric network using a connection profile (YAML or JSON file). Once connected, the user can submit	GRPC	All	Completed

[4] To be introduced in D4.3: Trustworthy Grant/cell-free Cooperative Access Mechanisms

NANCY Integration Points	NANCY Platform Interface	Objective	Protocol	Integration in testbeds/demonstrators	Status
		transactions or query the ledger through the smart contract API.			
BC – Marketplace (4,11)	NI10	Deploy the Smart Contracts of the marketplace on the Blockchain network	N/A	All	Completed
Wallet – Marketplace (5,11)	TBD	Send the required transactions to the Blockchain based marketplace for any operation (both read and write). All the interactions with the marketplace happen through the wallet (or the oracles in specific cases)	gRPC	All	Completed
Wallet – Maestro (5, 12)	TBD	Query the blockchain-based marketplace for potential service providers and sign SLA transactions. All the interactions with the blockchain happen through the wallet (which acts as a client app).	gRPC	All	Ongoing
Wallet – BSS (5,27)	TBD	Publish services in the blockchain-based marketplace and become aware of signed SLAs involving current clients. All the interactions with the blockchain happen through the wallet (which acts as a client app).	gRPC	All	Ongoing
AI_Virt – VoSyS Monitor (6,10)	N/A	Allow to deploy VNFs from OpenStack Nova into vManager partitions	Libvirt API	Italian InLab testbed	Completed
AI_Virt – SEMR (6,14)	NIS5	Integrate NAOMI in Slice Manager and further provide MLOps to models that are deployed on NAOMI.	REST API	TBD	Ongoing
Marketplace – SPP (11,21)	NI12	• Provide the information used for calculating the most suitable price for the better service. • Receive the most suitable price for the most suitable service fulfilling the given features.	REST API over HTTPS	Greek in-lab / outdoor testbed	Completed
Models – SEMR (13,14)	TBD	Network AI Workflow Democratisation (NAOMI) can	REST API	TBD	Ongoing

NANCY Integration Points	NANCY Platform Interface	Objective	Protocol	Integration in testbeds/demonstrators	Status
		provide MLOps for the models that are going to be deployed in repo			
Models – Elasticity (13,15)	NIS5	In-place resource elasticity technique allocating computing resources within a slice	REST API	Spanish	Ongoing
Models – TFS (13,17)	NIS8	Integration of the Throughput forecasting service will assist in predicting upcoming throughput optimizing AI-based network functionalities for different scenarios measuring speed and latency of model serving.	REST API	TBD	Not started
SEMR – FL-IDS (14,23)	NIS5	SEMR will provide a set of functionalities that cover the full ML pipeline through NAOMI tool in order to streamline the deployment of the FL-IDS model to the Greek In-lab Testbed	REST API	Greek In-lab Testbed	Ongoing
SEMR – AINQM (14,19)	NIS8	The SEMR will be continuously monitoring the performance of the AINQM module, thus triggering and overall optimizing retraining processes in changing network conditions.	REST API	TBD	Not started
PQC Sign – PQC SecCom (16,25)	Internal UE interface	Integration of the HW signature token into secure communication infrastructure to use PQC HW signing capabilities	API calls	Italian Massive IoT testbed	Ongoing
AINQM – NIF (19,20)	NIS8	Integration of Outage Probability Model in order to predict network outages for different scenarios	direct code integration	None	Completed
AINQM– XAI (19,22)	NIS8	AINQM will be utilised to calculate the outage probability, and XAI will provide the rationale behind the decisions of the respective AI model.	REST API	Greek In-lab Testbed	Ongoing
XAI – FL-IDS (22,23)	NIS8	The output of the distributed FL Training will be the AI-enabled Intrusion Detection System that the Greek in-lab testbed will utilize to identify different attacks	TBD	Greek In-lab Testbed	Completed

6.3. Integration Tests

In Table 7, the scheduled integration tests for all the identified NANCY integration points (section 6.2) are outlined. The test identifiers follow this naming convention: COMPONENTX_COMPONENTY_I00x, where COMPONENTX and COMPONENTY represent the components involved, and I00x is the sequential number of integration tests for each integration point. Similarly to Table 3, the Objective column provides a brief description of each test and its purpose.

Table 7: NANCY integration testing

Integration Point	Test Identifier	Test Objective
MRAT-NCP - ID-Mngnt	MRAT-NCP_ID-Mngnt_I001	Test Issuer to generate credentials
	MRAT-NCP_ID-Mngnt_I002	Test p-abc cryptographic operations with retrieved keys for verifier
	MRAT-NCP_ID-Mngnt_I00x	Test p-abc cryptographic operations with retrieved keys for pseudonym generation
MRAT-NCP – SemCom	MRAT-NCP_SemCom_I001	Test the provisioning of the data from the MRAT-NCP to the SemCom
	MRAT-NCP_SemCom_I002	Test the transmission of the extracted semantic information towards the MRAT-NCP
	MRAT-NCP_SemCom_I00x	Test the accuracy of the DT created at the edge server
MRAT-NCP – Models	MRAT-NCP_Models_I001	Test reconfiguration of data path depending on network status
ID Mngnt – Wallet	ID Mngnt_Wallet_I001	Test configuration of the wallet with ID credentials
	ID Mngnt_Wallet_I002	Test verification of credentials provided by a wallet
ID Mngnt – VoSySMonitor	ID Mngnt_VoSySMonitor_I001	Test read and write access of ID system on OP-TEE-based cache
DAC_Marketplace	DAC_ Marketplace_I001	Test the correct generation and reception of a digital agreement based on the information from the marketplace.
BC_Wallet	BC_Wallet_I001	Test correct user connection to the Fabric network using a connection profile (YAML or JSON file).
	BC_Wallet_I002	Test correct submission of transactions or queries to the ledger through the smart contract API.
BC_Marketplace	BC_ Marketplace_I001	Test the correct deployment of the smart contracts on the Blockchain network.
Wallet_Marketplace	Wallet_Marketplace_I001	Test the correct execution of functions in the marketplace from the wallet.
Wallet_BSS	Wallet_BSS_I001	Test correct submission of transactions to the ledger through the smart contract API. Specifically

Integration Point	Test Identifier	Test Objective
		publishing a new service in the marketplace (chaincode)
Wallet_Maestro	Wallet_Maestro_I001	Test correct submission of transactions to the ledger through the smart contract API. Specifically searching for a provider-service pair in the marketplace (chaincode)
AIVirt_VoSySMonitor	AIVirt_VoSySMonitor_I001	Test response of VoSySMonitor to Libvirt commands (virsh define, virsh start, virsh create, virsh shutdown, virsh destroy, virsh reboot, virsh suspend, virsh resume) triggered from AIVirt through OpenStack
AIVirt_SEMR	AIVirt_SEMR_I001	Test connectivity between the AI Virt and JSI SEMR cluster
	AIVirt_SEMR_I002	Test triggering the AI Virt (through the Slice Manager API end point) after provisioning it with the JSI SEMR kubeconfig file
	AIVirt_SEMR_I00x	Test namespace reconfiguration
Marketplace_SPP	Marketplace_SPP_I001	Test the correct reception of the most suitable price of the most suitable service from those available in the marketplace
Models_SEMR	Models_SEMR_I001	Testing the speed and latency of model serving
Models_Elasticity	Models_Elasticity_I001	Testing the response time and latency among different elasticity techniques
Models_TFS	Models_TFS_I001	Testing the speed and latency of model serving
SEMR_AINQM	SEMR_AINQM_I001	Testing outage prediction model performance along with model update and retraining via SEMR.
SEMR_FL-IDS	SEMR_FL-IDS_I001	Validation of the deployment of NAOMI in the Greek In-lab testbed and the consumption of network traffic in a csv format (network flows)
	SEMR_FL-IDS_I002	Utilisation of certain NAOMI functionalities (load model, load data, etc.) for the identification and classification of network flows between benign traffic and attack type
PQCSign_PQCSecCom	PQCSign_PQCSecCom _I001	Test the integration of the PQC Signature Token and the PQC secure communication component for dilithium algorithm sign in hardware
AINQM_NIF	AINQM_NIF_I001	Test the correct format for the input and output of the AI model
AINQM_XAI	AINQM_XAI_I001	AINQM will be part of a demonstration scenario that will provide the outage probability utilizing an XGBoost model, while XAI will explain the predictions.
XAI_FL-IDS	XAI_FL-IDS_I001	Integration of trained FL model to the pipeline of cyber-attack identification
	XAI_FL-IDS_I002	Integration of trained FL model to the pipeline of NANCY XAI Toolkit

6.4. Next Integration Steps

As described in section 6.1 (Table 4), the forthcoming steps towards the final release of the NANCY platform involve the ongoing development and testing updates for each component, as well as the secure interconnection of the different NANCY deployment domains, as per the operational requirements of the different NANCY use cases. Following the validation of the integration points through the bilateral integration tests, more extensive integration testing and validation activities will take place. Specific end-to-end testing scenarios will be defined to ensure the readiness of the integrated platform to support the technical data flows for the various operational scenarios (T6.4-T6.9). These tests will be manually implemented with the involvement of the respective component providers and will be documented in D6.3.

During the various testing phases, feedback for improvements and fixes will be provided to the technical component providers. The goal is to deliver the final version of the NANCY integrated platform in M33, which will be used for the execution and assessment of NANCY operational scenarios in the context of the different testbeds and demonstrators.

7. Conclusions

NANCY aims to provide a secure and advanced framework for B5G wireless networks, leveraging state-of-the-art technologies in Artificial Intelligence, Blockchain, MEC and Orchestration to facilitate secure and sophisticated resource management, adaptable networking, and improve orchestration capabilities. This deliverable utilizes the results of the components being developed in WP2-WP5, as well as NANCY architecture's functional and deployment view reported in [1] to document the ongoing integration efforts aimed at creating the NANCY integrated platform. Following a well-defined integration methodology, it also establishes the groundwork for its adaptable instantiation, customised to meet the specific needs of various NANCY operational scenarios (within the NANCY testbeds and demonstrators).

The deployment domains of the NANCY system are covered extensively, describing the high-level view of their core functionalities (described in more detail in the corresponding deliverables of WP2-WP5), envisioned interoperability among their subsequent components, as well as the physical deployment location of the full set of NANCY components in the context of NANCY operational scenarios. We also focus on the reusability aspects of the platform, highlighting the integration by design choices that have been made to facilitate the flexible instantiation of the platform across different deployment setups, as well as the integration efforts to support/interoperate with both commercial and O-RAN solutions.

This document also provided a detailed overview of the NANCY integration environment, focusing on the CI/CD system and Kubernetes cluster implemented to support developers in their development and testing activities and offer a central point of triggering/control for deployments towards the NANCY testbed and demonstrator setups, in conjunction with NANCY Service and Resource Orchestrators. The integration plan for the full duration of the relevant WP6 integration tasks is outlined, followed by the identification and specification of NANCY integration points and the reporting of the ongoing functional and integration testing activities of the individual NANCY components for the first release of the NANCY integrated platform.

D6.3 "NANCY Integrated System – Final Version" will offer comprehensive and up-to-date information about the specification and validation activities of the final release of the NANCY integrated platform, which will support the NANCY demonstrators.

Bibliography

[1] NANCY Consortium, “D6.1: B-RAN and 5G End-to-end Facilities Setup,” 2024.

[2] NANCY Consortium, “D3.1: NANCY Architecture Design,” 2024.

[3] Hetzner, “Hetzner cloud infrastructure,” [Online]. Available: https://www.hetzner.com/. [Accessed 20 01 2025].

[4] “Github projects,” [Online]. Available: https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects. [Accessed 20 01 2025].

[5] “Slack team communication platform:,” [Online]. Available: https://slack.com/. [Accessed 20 01 2025].

[6] “MAESTRO Service Orchestrator,” [Online]. Available: https://maestro-mkdocs.readthedocs.io/en/latest/. [Accessed 20 01 2025].

[7] NANCY Consortium, “D4.2: Resource Elasticity Techniques,” 2024.

[8] “Helm charts,” [Online]. Available: https://helm.sh/. [Accessed 20 01 2025].

[9] “Docker compose,” [Online]. Available: https://docs.docker.com/compose/. [Accessed 20 01 2025].

[10] “OpenSlice,” [Online]. Available: https://osl.etsi.org/. [Accessed 20 01 2025].

[11] “Harbor container registry,” [Online]. Available: https://goharbor.io/. [Accessed 20 01 2025].

[12] “Kubernetes kubeconfig files,” [Online]. Available: https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/. [Accessed 24 01 2025].

[13] NANCY Consortium, “D3.3: NANCY AI-based B-RAN Orchestration,” 2024.

[14] NANCY Consortium, “D5.2: NANCY Security and Privacy Distributed,” 2024.

[15] NANCY Consortium, “D4.1: Computational Offloading and User-centric Caching,” 2024.

[16] NANCY Consortium, “D4.5: Smart Pricing Policies,” 2024.

[17] “LibVirt,” [Online]. Available: https://libvirt.org/. [Accessed 20 01 2025].

[18] “OpenStack,” [Online]. Available: https://www.openstack.org/. [Accessed 20 01 2025].

[19] “Let's Encrypt CA:,” [Online]. Available: https://letsencrypt.org/. [Accessed 20 01 2025].

[20] “Calico CNI,” [Online]. Available: https://docs.tigera.io/calico/latest/about/. [Accessed 27 01 2025].

[21] "NGINX Ingress Controller," [Online]. Available: https://docs.nginx.com/nginx-ingress-controller/. [Accessed 20 01 2025].

[22] "Kubernetes cert manager," [Online]. Available: https://cert-manager.io/. [Accessed 27 01 2025].

[23] "Keycloak," [Online]. Available: https://www.keycloak.org/. [Accessed 20 01 2025].

[24] "Pfsense open-source firewall," [Online]. Available: https://www.pfsense.org/. [Accessed 20 01 2025].

[25] "NAOMI," [Online]. Available: https://github.com/sensorlab/NAOMI. [Accessed 24 01 2025].

[26] "Design Structure Matrix," [Online]. Available: https://dsmweb.org/introduction-to-dsm/. [Accessed 20 01 2025].

[27] T. Chen and C. Guestrin, "XGBoost," 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.

A. Annex: Integration Points Specifications

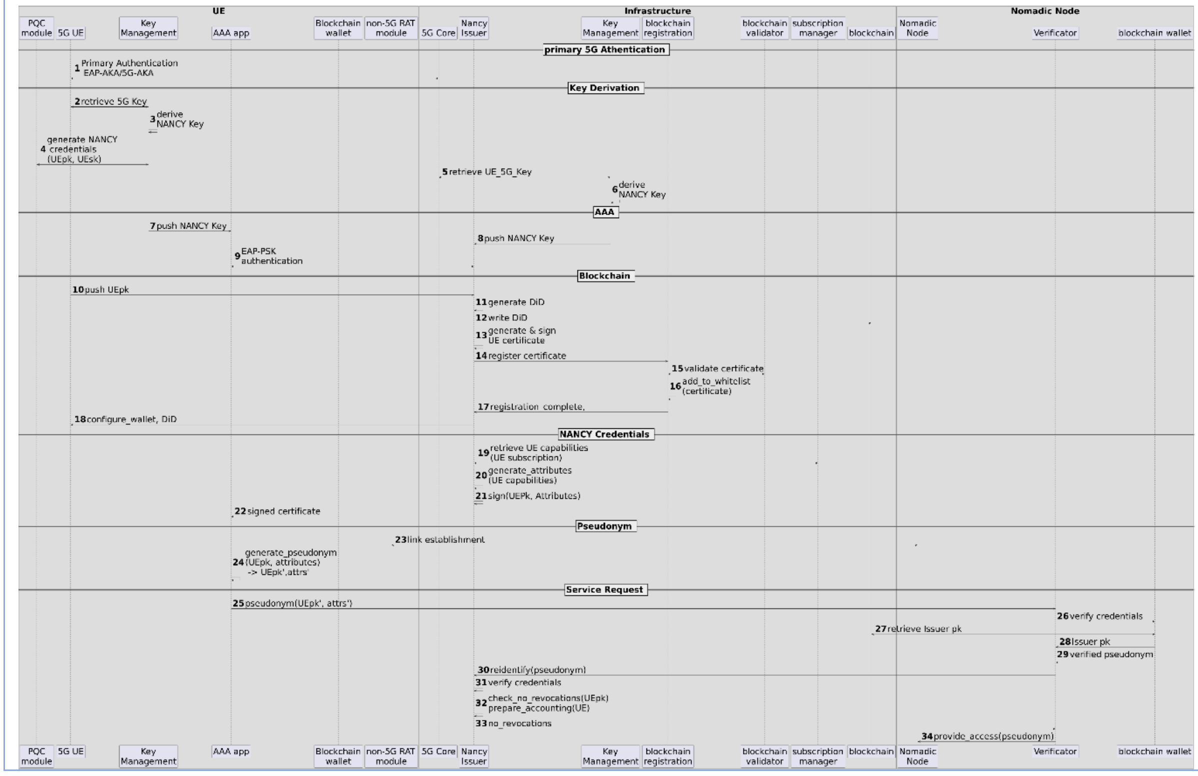
Following the identification and specification summary of the integration points presented in Table 5 and Table 6 respectively, this Annex provides the detailed specifications for the corresponding integration points, including sequence diagrams that describe the different data/information exchange flows.

A.1 MRAT-NCP – ID Management

Integration Point Description	
Identifier	1.2 – MRAT-NCP – ID Mngnt
Involved organisations	UMU
Integration Point Purpose	*Allow access for 5G network usage for a remote non-5G subscriber but NANCY subscriber.*
NANCY platform interface	-
Interface/ Protocol technology	*PC5, REST API over HTTPS*
Status	*Ongoing*
Sequence diagram	
Sequence diagram is presented joined with A.2 section	

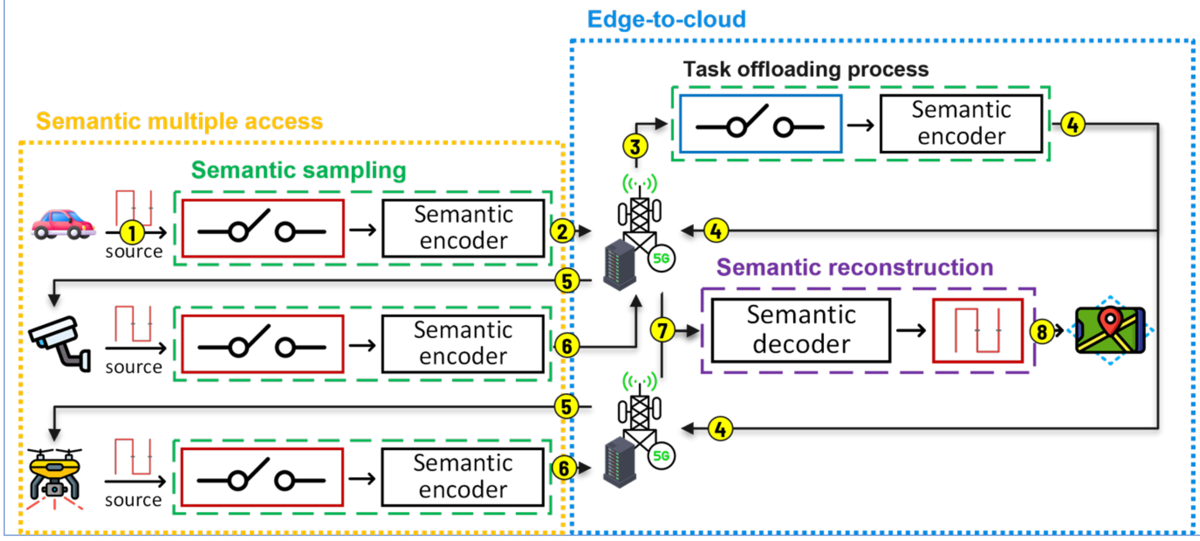
A.2 ID-Management – Wallet

Integration Point Description	
Identifier	1.5 – MRAT-NCP – Wallet
Involved organisations	UMU, NEC
Integration Point Purpose	*Use the wallet for ID management purposes such as Authentication and Authorisation. Main steps include:* *Primary 5G authentication: regular (non-)3GPP access to the 5G network* *Key Derivation: use 5G keys to derivate NANCY keys for PSK purposes* *AAA: Authentication between UE and NANCY issuer with derived Keys* *Blockchain: subscribing NANCY subscriber to the blockchain* *NANCY credential: Generation and deliver of NANCY credentials to NANCY subscriber* *Pseudonym: Generation of privacy preserving pseudonyms for attribute-based authentication*

	Service Request: NANCY subscriber request a service using the generated credentials.
NANCY platform interface	*NIS13*
Interface/ Protocol technology	*REST API over HTTPS, oracles*
Status	*Ongoing*
Sequence diagram	
	

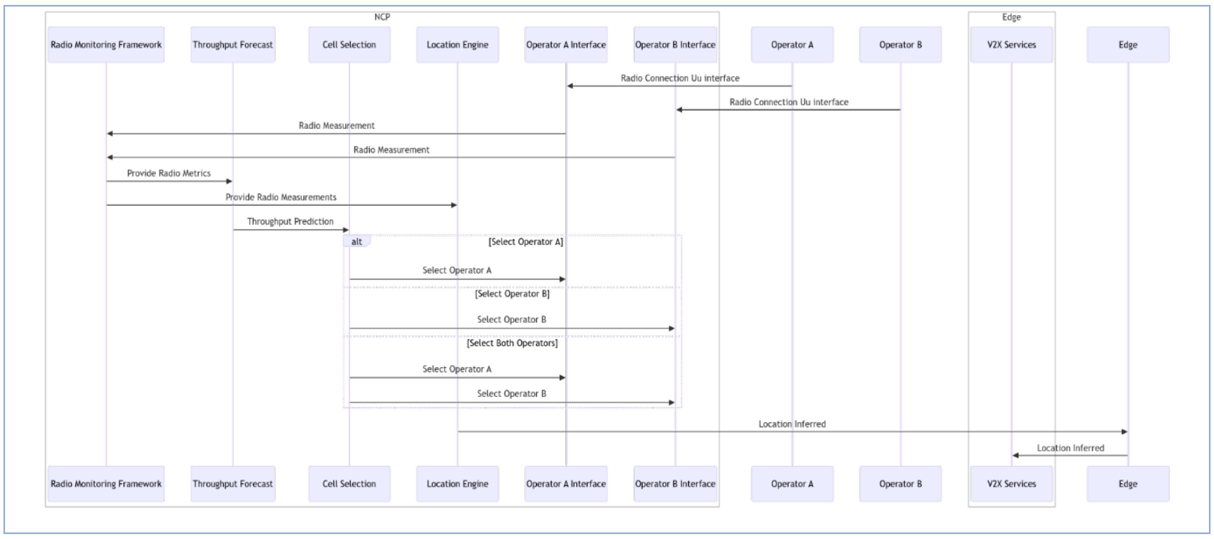
A.3 MRAT-NCP – SemCom

Integration Point Description	
Identifier	1.8 – MRAT-NCP – SemCom
Involved organisations	UMU, INNO
Integration Point Purpose	*The goal of this integration point is to showcase the energy and data efficiency achieved by semantic communications. The SemCom component is comprised of two entities, namely the semantic encoder and the semantic decoder. In the simulated setup, the former is deployed on the network nodes (cars, RSUs, drones, etc.) capturing video of the observed location. The extracted semantic information will be transmitted through the network to the edge server, where the semantic decoder will be deployed in order to create a DT.*

NANCY platform interface	*Uu, S1, NIS7, NIS5, NIS8*
Interface/ Protocol technology	*N/A*
Status	*Ongoing*
Sequence diagram	
 Edge-to-cloud Task offloading process Semantic encoder Semantic multiple access Semantic sampling source Semantic encoder Semantic reconstruction Semantic decoder 1 2 3 4 5 6 7 8 5G	

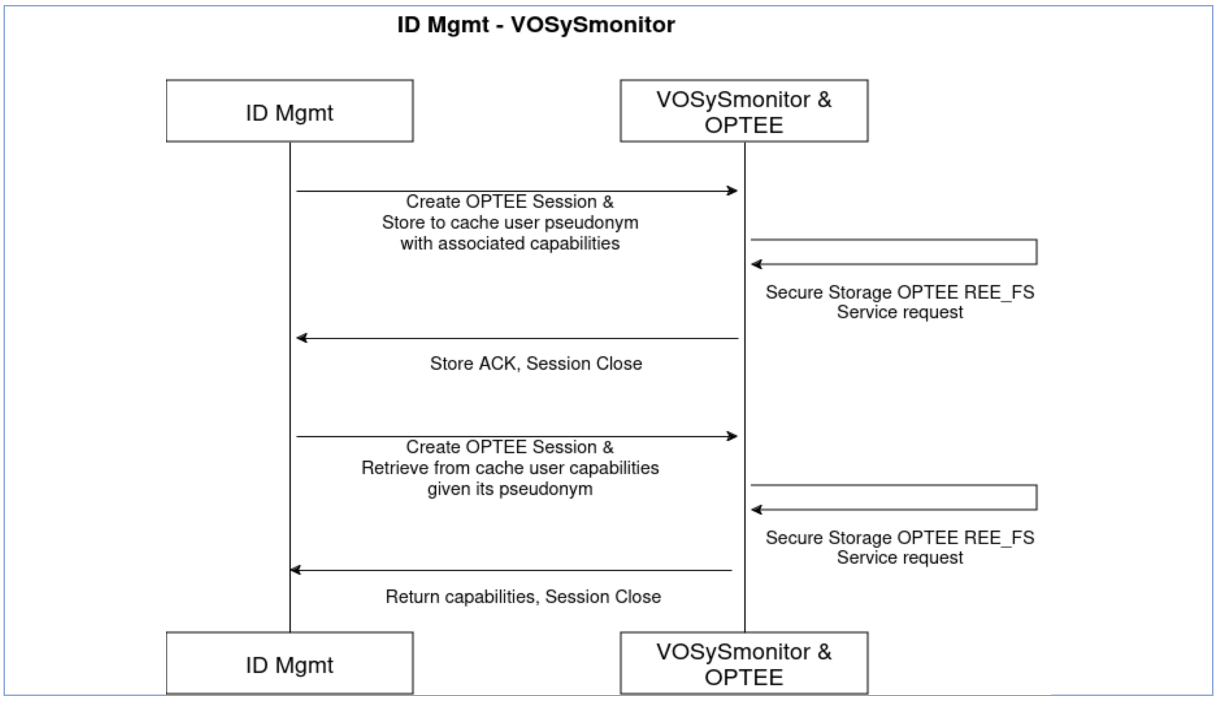
A.4 MRAT-NCP – Models

Integration Point Description	
Identifier	1.13 – MRAT-NCP – Models
Involved organisations	UMU, JSI, CERTH
Integration Point Purpose	*To feed embedded AI models in the Nomadic Connectivity Provider (NCP) with monitored data to rule the behaviour of the connectivity and V2X location services of Remote vehicles.*
NANCY platform interface	*NIS7, NIS5, NIS8*
Interface/ Protocol technology	*REST API over HTTPS*
Status	*Ongoing*
Sequence diagram	



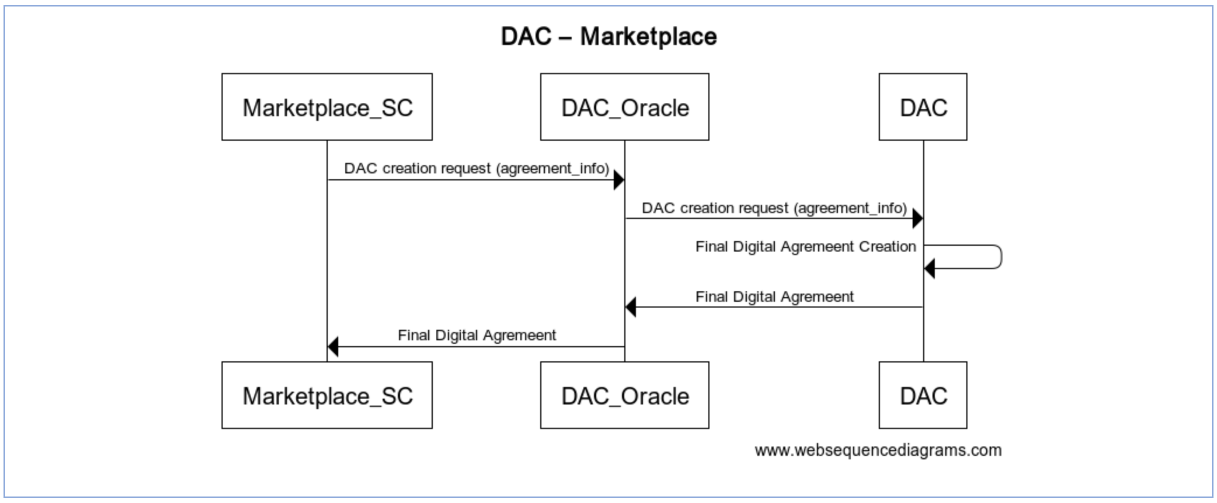
A.5 ID Management – VoSySMonitor

Integration Point Description	
Identifier	2.10 – ID Mgnt – VoSySMonitor
Involved organisations	UMU, VOS
Integration Point Purpose	*Instantiate Open Trusted Execution Environment (OPTEE) solution for caching mechanisms on ARM-based far-edge device included in the MRAT-NCP.*
NANCY platform interface	*N/A* *Integration through CLI tool to store and retrieve secure assets from the OPTEE Secure Storage cache*
Interface/ Protocol technology	*API based on Global Platform TEE Client API*
Status	*Ongoing*
Sequence diagram	


ID Mgmt - VOSySmonitor
ID Mgmt
VOSySmonitor & OPTEE
Create OPTEE Session & Store to cache user pseudonym with associated capabilities
Secure Storage OPTEE REE_FS Service request
Store ACK, Session Close
Create OPTEE Session & Retrieve from cache user capabilities given its pseudonym
Secure Storage OPTEE REE_FS Service request
Return capabilities, Session Close
ID Mgmt
VOSySmonitor & OPTEE

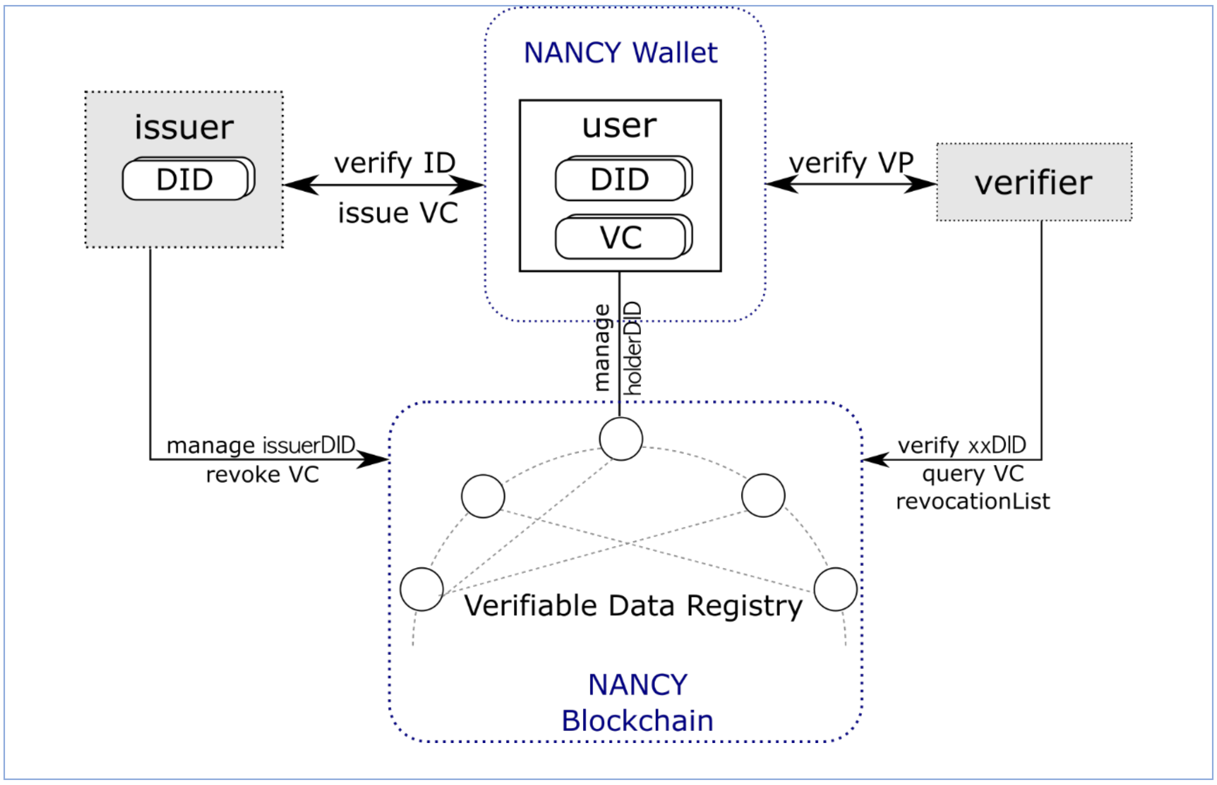
A.6 DAC – Marketplace

Integration Point Description	
Identifier	3,11 – DAC – Marketplace
Involved organisations	DRAXIS, TECN
Integration Point Purpose	• *Provide the required information for creating the digital agreement.* • *Receive the generated digital agreement to be signed*
NANCY platform interface	*NI11*
Interface/ Protocol technology	*REST API over HTTPS*
Status	*Completed*
Sequence diagram	


DAC – Marketplace
Marketplace_SC
DAC_Oracle
DAC
DAC creation request (agreement_info)
DAC creation request (agreement_info)
Final Digital Agremeent Creation
Final Digital Agremeent
Final Digital Agremeent
Marketplace_SC
DAC_Oracle
DAC
www.websequencediagrams.com

A.7 BC – Wallet

Integration Point Description	
Identifier	4,5 – BC – Wallet
Involved organisations	NEC
Integration Point Purpose	*To provide the means to securely connect to the NANCY Blockchain by using the NANCY wallet, and to query and alter the ledger.*
NANCY platform interface	*NIS13*
Interface/ Protocol technology	*GRPC calls*
Status	*Completed*
Sequence diagram	

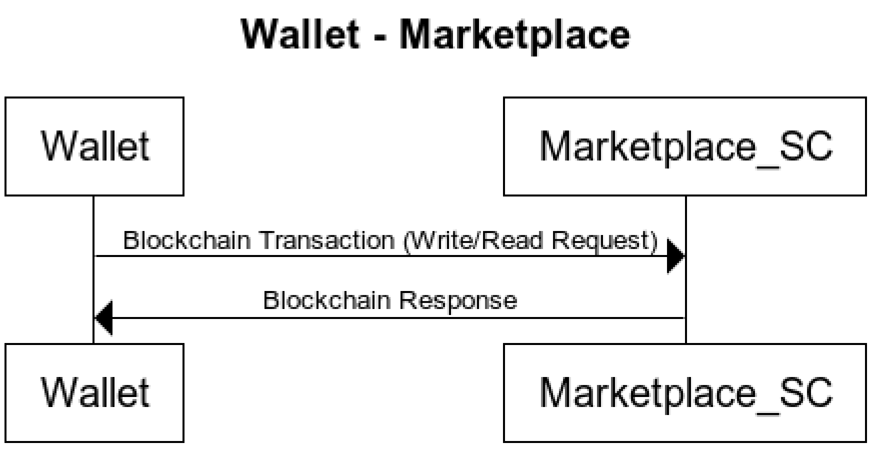


A.8 BC – Marketplace

Integration Point Description	
Identifier	4,11 – BC – Marketplace
Involved organisations	NEC, TECN
Integration Point Purpose	*Deploy the Smart Contracts of the marketplace on the Blockchain network.*
NANCY platform interface	*NI10*
Interface/ Protocol technology	*N/A*
Status	*Completed*
Sequence diagram	
N/A: *The marketplace SC are deployed in the Blockchain (Blockchain is its infrastructure).*	

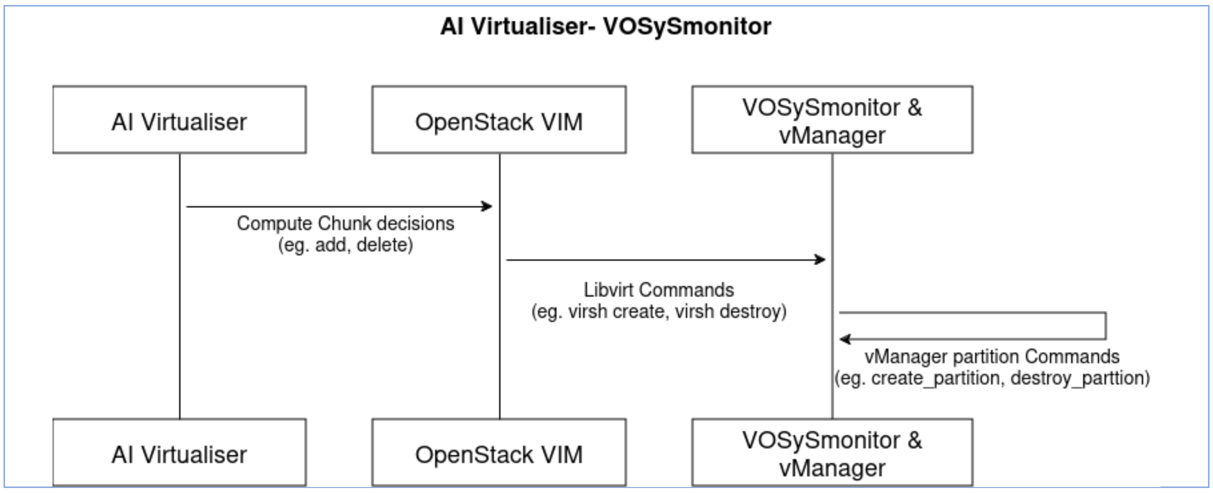
A.9 Wallet – Marketplace

Integration Point Description	
Identifier	5,11 – Wallet – Marketplace
Involved organisations	NEC, TECN

Integration Point Purpose	*Send the required transactions to the Blockchain based marketplace for any operation (both read and write). All the interactions with the marketplace happen through the wallet (or the oracles in specific cases)*
NANCY platform interface	*NI10*
Interface/ Protocol technology	*gRPC*
Status	*Completed*
Sequence diagram	
 Wallet - Marketplace Wallet Marketplace_SC Blockchain Transaction (Write/Read Request) Blockchain Response Wallet Marketplace_SC	

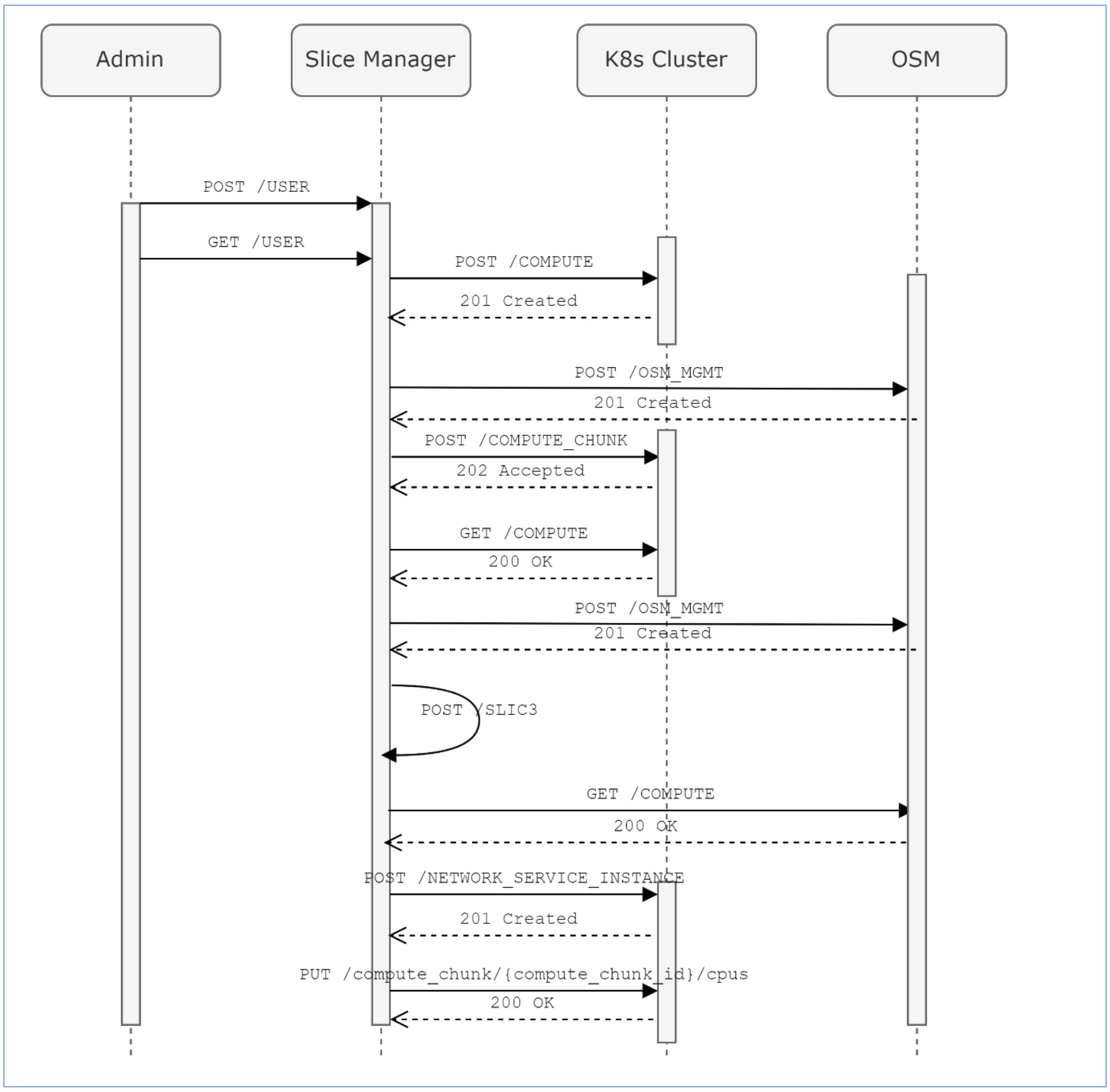
A.10 AI Virt. - VoSySMonitor

Integration Point Description	
Identifier	6,10 – AI Virt. - VoSySMonitor
Involved organisations	i2CAT, VOS
Integration Point Purpose	*Allow to deploy VNFs from OpenStack Nova into vManager partitions*
NANCY platform interface	*N/A* *The two components integrate through the VIM (OpenStack) to the Virtualisation Layer (vManager) interface, which is placed below the Or-Vi NANCY interface at the edge domain.*
Interface/ Protocol technology	*Libvirt Management API*
Status	*Completed*
Sequence diagram	


AI Virtualiser- VOSySmonitor
AI Virtualiser
OpenStack VIM
VOSySmonitor & vManager
Compute Chunk decisions (eg. add, delete)
Libvirt Commands (eg. virsh create, virsh destroy)
vManager partition Commands (eg. create_partition, destroy_partion)
AI Virtualiser
OpenStack VIM
VOSySmonitor & vManager

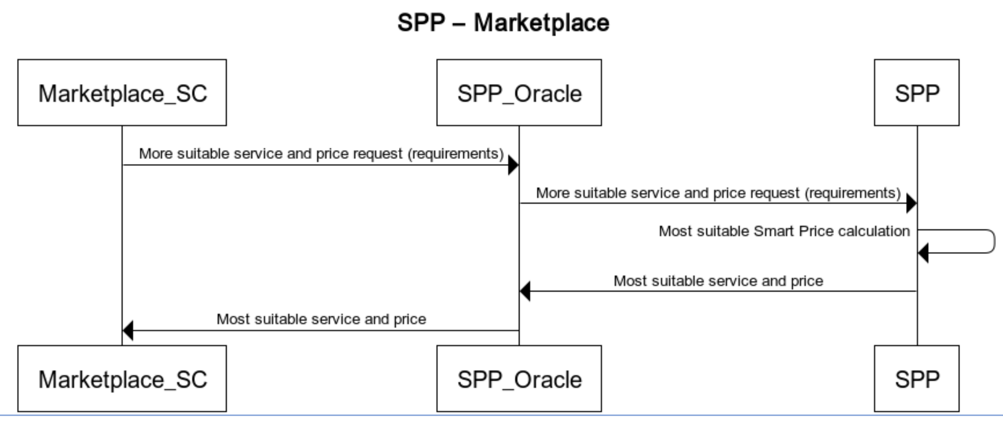
A.11 AI Virt. - SEMR

Integration Point Description	
Identifier	6,14 – AI Virt. - SEMR
Involved organisations	i2CAT, JSI
Integration Point Purpose	*Reconfigure resources of the JSI SEMR namespace via the Slice Manager API end point, which is the external interface of the AI Virt.*
NANCY platform interface	*NIS1, NIS2*
Interface/ Protocol technology	*REST API over HTTPS*
Status	*Ongoing*
Sequence diagram	



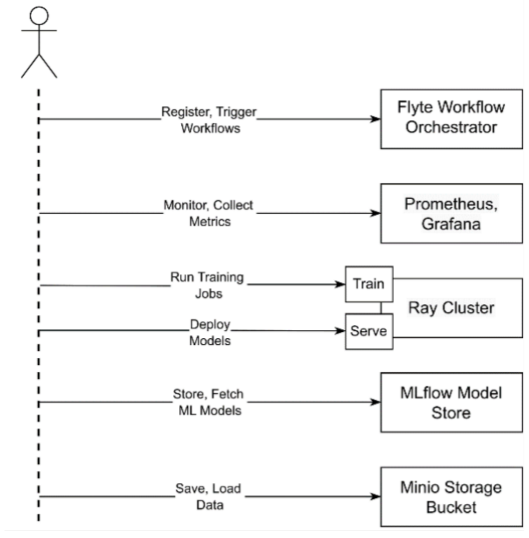
A.12 Marketplace – SPP

Integration Point Description	
Identifier	11,21 – Marketplace - SPP
Involved organisations	TECN, 8Bells
Integration Point Purpose	• *Provide the information used for calculating the most suitable price for the better service.* • *Receive the most suitable price of the most suitable service fulfilling given features.*
NANCY platform interface	*NI12*
Interface/ Protocol technology	*REST API over HTTPS*

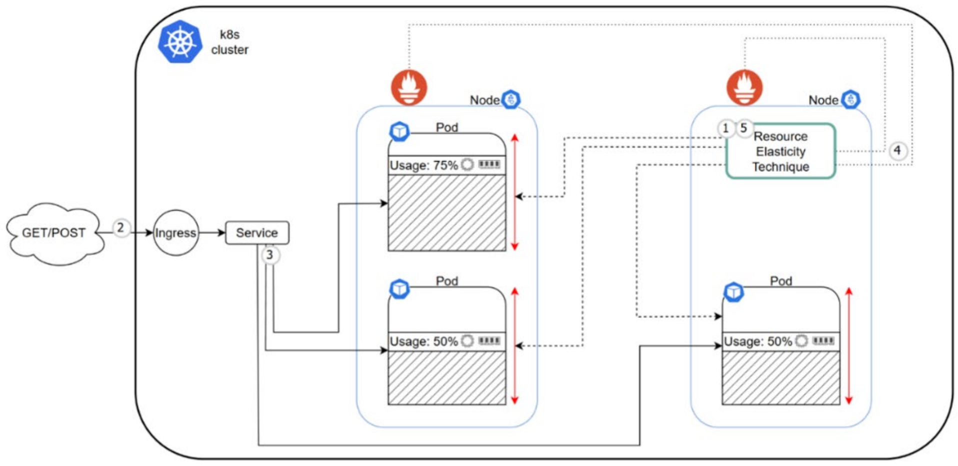
Status	*Ongoing*
Sequence diagram	
 SPP – Marketplace Marketplace_SC SPP_Oracle SPP More suitable service and price request (requirements) More suitable service and price request (requirements) Most suitable Smart Price calculation Most suitable service and price Most suitable service and price Marketplace_SC SPP_Oracle SPP	

A.13 Models - SEMR

Integration Point Description	
Identifier	13,14– Models - SEMR
Involved organisations	JSI
Integration Point Purpose	*The integration will evaluate the performance of the orchestration of the lifecycle of AI-based models and their serving.*
NANCY platform interface	*NIS5*
Interface/ Protocol technology	*REST API*
Status	*Ongoing*
Sequence diagram	



A.14 Models - Elasticity

Integration Point Description	
Identifier	13,15 – Models -Elasticity
Involved organisations	JSI
Integration Point Purpose	*The integration will evaluate the performance of the elasticity module in a more dynamic environment.*
NANCY platform interface	*<e.g., NIS1, NIS8, NI9>NIS5*
Interface/ Protocol technology	*REST API*
Status	*Ongoing*
Sequence diagram	
 k8s cluster Node Pod Usage: 75% Pod Usage: 50% Node Resource Elasticity Technique Pod Usage: 50% GET/POST 2 Ingress Service 3 1 5 4 *The system operates by hosting containers (pods) on virtual machines (VMs), and managing the execution of these pods as they handle service requests. These service requests utilize HTTP protocol and are further routed through the Ingress controller, which directs them to*	

the appropriate services. These services then distribute the requests to the relevant pods based on internal routing and load-balancing mechanisms. By continuously adjusting resources based on demand, the system ensures seamless performance under varying conditions, optimizing both resource utilisation and service reliability.

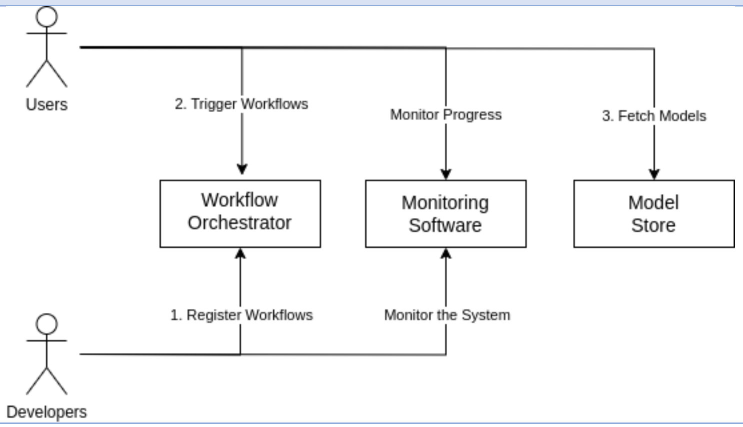
A.15 Models – TFS

Integration Point Description	
Identifier	13,17 – Models -TFS
Involved organisations	JSI, CERTH
Integration Point Purpose	*The integration will evaluate the performance of the Throughput forecasting service, which will be assisted by other AI Models, in order to enhance network functionalities and ultimately support decision making.*
NANCY platform interface	*NIS8*
Interface/ Protocol technology	*REST API*
Status	*Not started*
Sequence diagram	
Seq. diagram for this integration point will be provided in D6.3.	

A.16 SEMR – AINQM

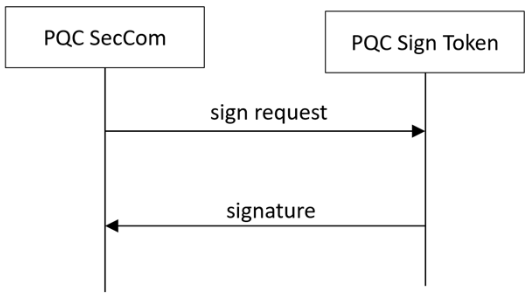
Integration Point Description	
Identifier	14,19 – SEMR – AINQM
Involved organisations	JSI, CERTH
Integration Point Purpose	*The integration will evaluate outage probability prediction performance, supported by SEMR for model deployment, inference and re-training if needed.*
NANCY platform interface	*NIS8*
Interface/ Protocol technology	*REST API*
Status	*Not started*
Sequence diagram	
Seq. diagram for this integration point will be provided in D6.3.	

A.17 SEMR – FL-IDS

Integration Point Description	
Identifier	14,23 – SEMR – FL-IDS
Involved organisations	JSI, MINDS
Integration Point Purpose	*The Self-Evolving Model Repository (NAOMI) will be integrated in the Greek In-lab testbed, aiming to support the Federated Learning-based training of an AI Intrusion Detection System (IDS) along with its deployment, inference, and re-training if needed. The NAOMI framework will offer several functionalities (train, load model, load data) to the FL-IDS, aiming to streamline the Federated Learning Operations (FLOps).*
NANCY platform interface	*NIS8*
Interface/ Protocol technology	*REST API*
Status	*Ongoing*
Sequence diagram	
 Users 2. Trigger Workflows Monitor Progress 3. Fetch Models Workflow Orchestrator Monitoring Software Model Store 1. Register Workflows Monitor the System Developers	

A.18 PQC Sign – PQC SecCom

Integration Point Description	
Identifier	16,25 – PQC Sign – PQC SecCom
Involved organisations	TDIS, TEI
Integration Point Purpose	*This integration point is between PQC signature token (provided by TDIS) and PQC Secure Communication Library (developed by TEI). The purpose is to perform the signing operation using PQC algorithms in hardware (in the TDIS token). This approach improves the security of the overall solution.*

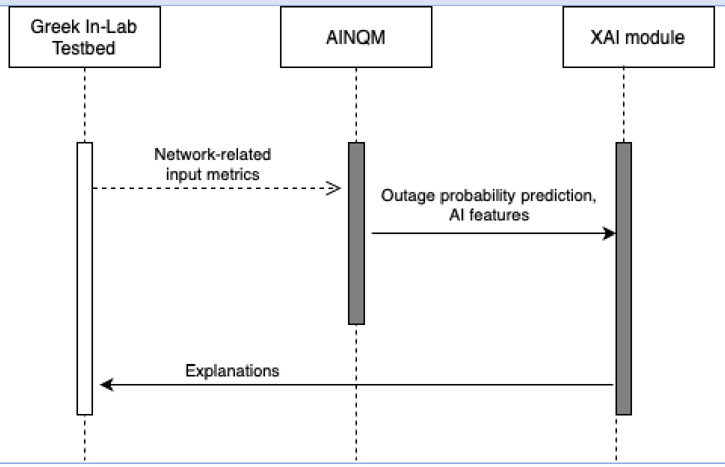
NANCY platform interface	*N/A - It is an internal integration point inside the UE.*
Interface/ Protocol technology	*API calls (defined by TDIS)*
Status	*Ongoing (to be demonstrated in Italian Massive IoT testbed)*
Sequence diagram	
	

A.19 AINQM - NIF

Integration Point Description	
Identifier	19,20 – AINQM - NIF
Involved organisations	CERTH, 8Bells
Integration Point Purpose	*The purpose of this synergy point is to provide the NIF with the capability of predicting network outages using an AI model.*
NANCY platform interface	*NIS8*
Interface/ Protocol technology	*Direct Code Integration*
Status	*Completed*
Sequence diagram	
N/A	

A.20 AINQM - XAI

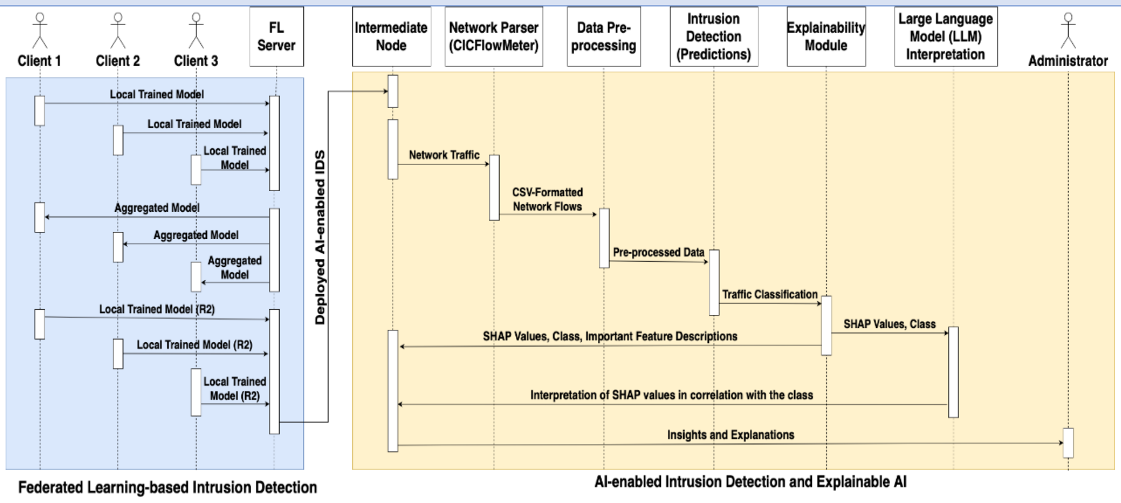
Integration Point Description	
Identifier	19,22 – AINQM - XAI
Involved organisations	CERTH, MINDS
Integration Point Purpose	*The AINQM will receive inputs regarding network status by the Greek In-lab Testbed in order to predict the outage probability utilizing an*

	XGBoost AI model [27]. *After the predictions, the XAI module will provide explanations on the decisions by showing which features contributed the most the final prediction.*
NANCY platform interface	*NIS8*
Interface/ Protocol technology	*REST API*
Status	*Ongoing*
Sequence diagram	
	

Greek In-Lab Testbed
AINQM
XAI module
Network-related input metrics
Outage probability prediction, AI features
Explanations

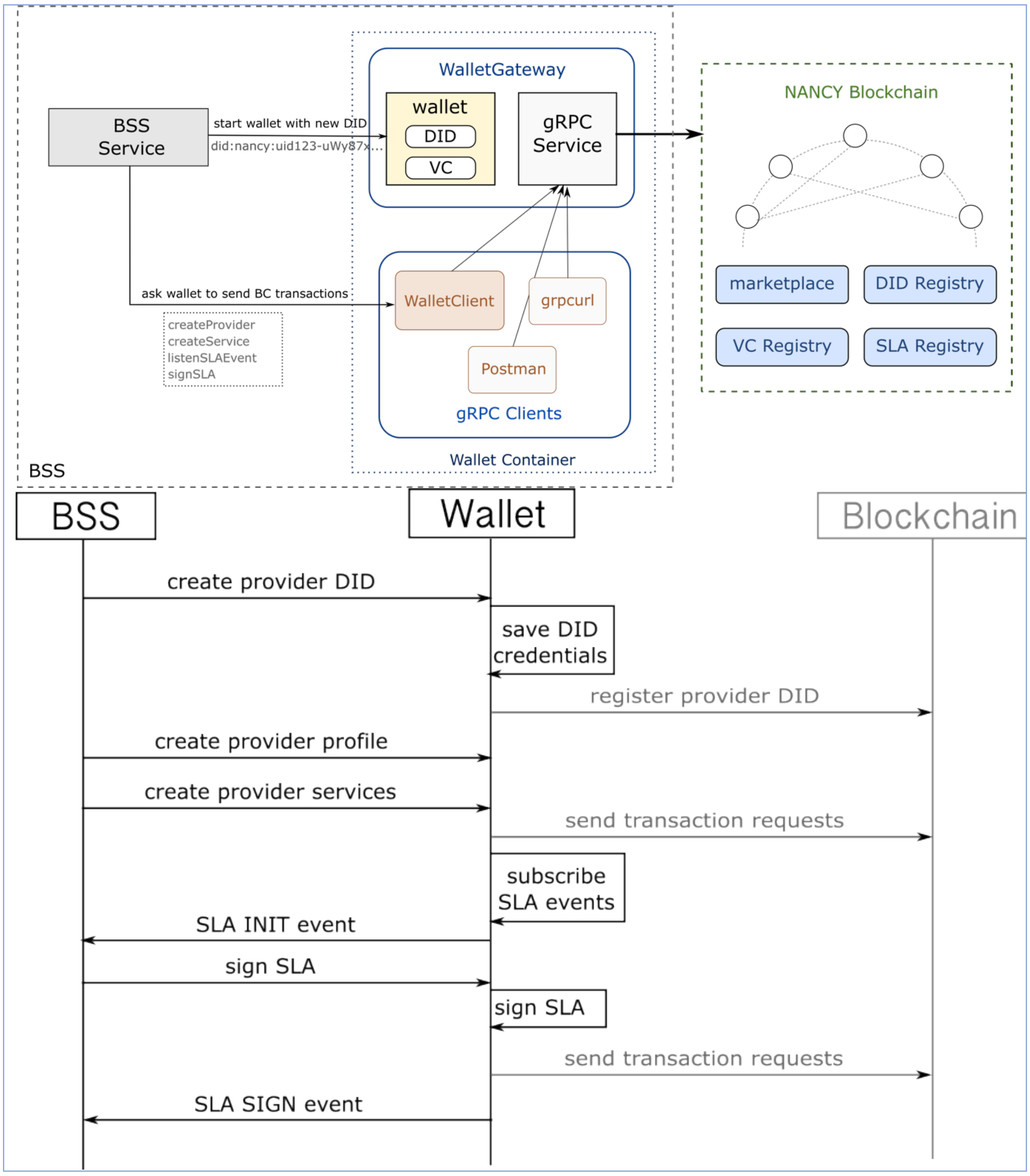
A.21 XAI – FL-IDS

Integration Point Description	
Identifier	22,23 – XAI -FL-IDS
Involved organisations	MINDS
Integration Point Purpose	*This integration point involves a Federated Learning (FL) framework for training an AI model across three clients with local network traffic data, ensuring privacy by keeping data decentralised. The trained global model is deployed on an intermediate node, which collects raw network traffic, parses it into CSV-formatted network flows, and preprocesses it for classification. The deployed model classifies the traffic into normal or various attack types. An integrated Explainable AI (XAI) module identifies key features influencing predictions, calculates SHAP values, and sends these along with the model's outputs to a Large Language Model (LLM). The LLM translates these insights into human-understandable explanations, enabling administrators to interpret model decisions effectively. This system ensures privacy-preserving training, automated traffic analysis, and actionable interpretability.*
NANCY platform interface	*NIS8*
Interface/ Protocol technology	*REST APIs*
Status	*Ongoing*

Sequence diagram


A.22 Wallet – BSS

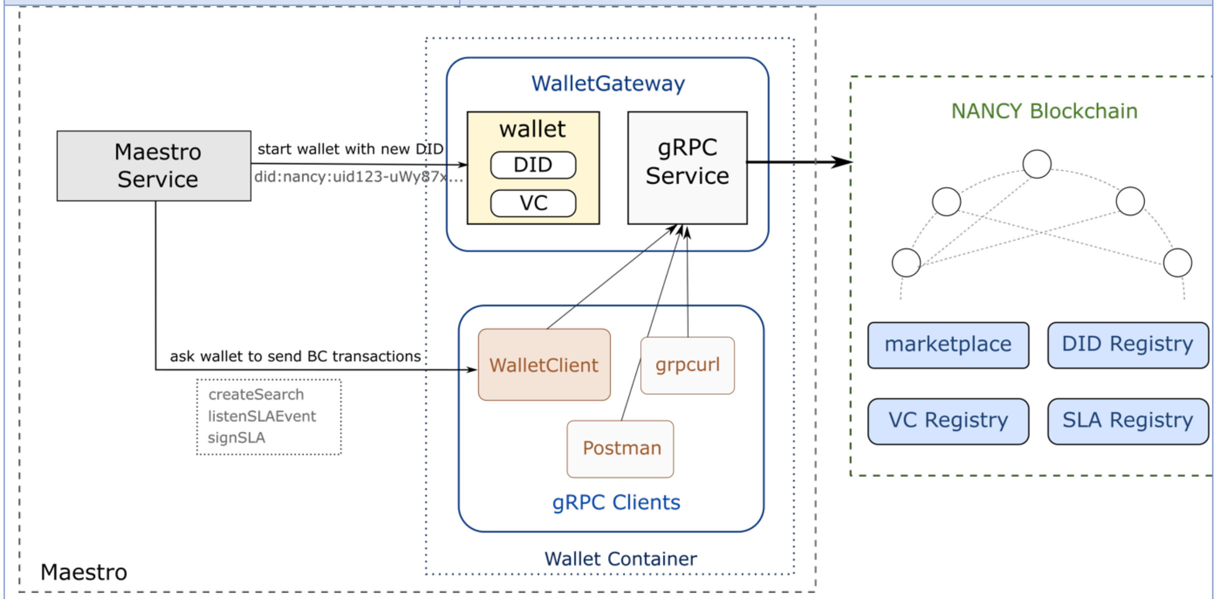
Integration Point Description	
Identifier	5,27 Wallet – BSS
Involved organisations	NEC, UBI
Integration Point Purpose	*The wallet creates and registers DIDs to represent providers in BSS.* *Furthermore, the wallet provides the means to securely connect to the NANCY Blockchain, so that BSS can post provider services on the blockchain-based marketplace, listen to SLA creation and signing events and sign the SLA when agreement is reached with consumers.*
NANCY platform interface	*TBD*
Interface/ Protocol technology	*Command-line calls to docker container*
Status	*Ongoing*
Sequence diagram	

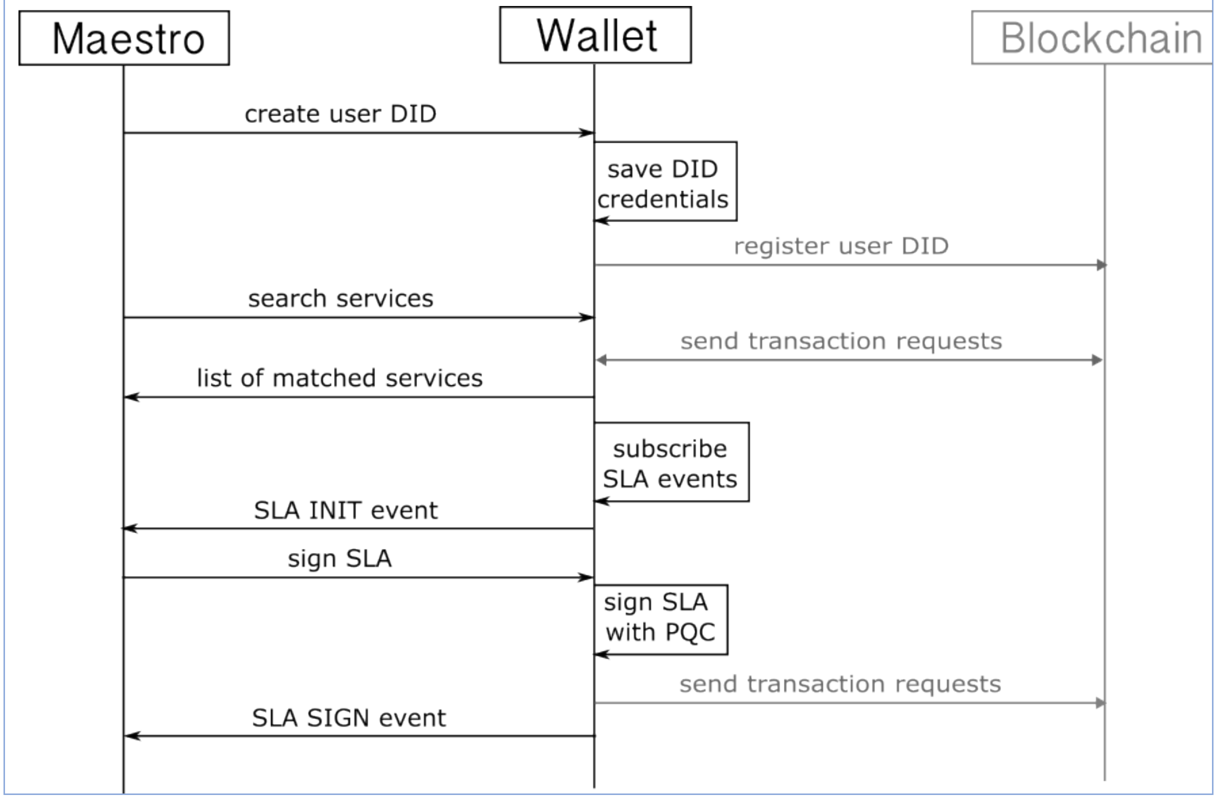


A.23 Wallet – Maestro

Integration Point Description	
Identifier	5,12 Wallet – Maestro
Involved organisations	NEC, UBI
Integration Point Purpose	*The wallet creates and registers DIDs to represent users in Maestro. Furthermore, the wallet provides the means to securely connect to the NANCY Blockchain, so that users in Maestro can search suitable services on the blockchain-based marketplace, listen to SLA creation and signing events and sign the SLA when agreement is reached with providers.*

NANCY platform interface	*TBD*
Interface/ Protocol technology	*Command-line calls to docker container*
Status	*Ongoing*
Sequence diagram	


WalletGateway
Maestro Service
start wallet with new DID
did:nancy:uid123-uWy87x...
wallet
DID
VC
gRPC Service
NANCY Blockchain
ask wallet to send BC transactions
createSearch
listenSLAEvent
signSLA
WalletClient
grpcurl
Postman
gRPC Clients
marketplace
DID Registry
VC Registry
SLA Registry
Wallet Container
Maestro


Maestro
Wallet
Blockchain
create user DID
save DID credentials
register user DID
search services
send transaction requests
list of matched services
subscribe SLA events
SLA INIT event
sign SLA
sign SLA with PQC
send transaction requests
SLA SIGN event