Ref. Ares(2025)11588396 - 24/12/2025

NebulOuS

A META OPERATING SYSTEM FOR BROKERING
HYPER-DISTRIBUTED APPLICATIONS ON
CLOUD COMPUTING CONTINUUMS

D 6.2
Final Release of the NebulOuS Integrated platform and Pilot Demonstrators Evaluation Report

24/12/2025

Grant Agreement No.	101070516
Project Acronym/ Name	NebulOuS - A META OPERATING SYSTEM FOR BROKERING HYPER DISTRIBUTED APPLICATIONS ON CLOUD COMPUTINGCONTINUUMS
Topic	HORIZON-CL4-2021-DATA-01-05
Type of action	HORIZON-RIA
Service	CNECT/E/04
Duration	40 months (starting date 1 September 2022)
Deliverable title	Final Release of the NebulOuS Integrated Platform and Pilot Demonstrators Evaluation Report
Deliverable number	D6.2
Deliverable version	1.1
Contractual date of delivery	31 December 2025
Actual date of delivery	24 December 2025
Nature of deliverable	Other
Dissemination level	Public
Work Package	WP6
Deliverable lead	7bulls.com
Author(s)	Joanna Chmielewska, Radosław Piliszek, Michał Soczewka, Ricard Cervera, Robert Sanfeliu Prat, Bruno Pereira, Nikos Papageorgopoulos, Azimina Tzana, Francisco Alvarez Terribas, Moritz von Stietencron
Abstract	This deliverable presents the outcomes of Work Package 6, which focused on validating, evaluating, and assuring the overall quality and readiness of the NebulOuS platform. This document summarizes the execution of pilot demonstrator activities.
Keywords	Integration, architecture, testing, use case evaluation

DISCLAIMER

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Directorate-General for Communications Networks, Content and Technology. Neither the European Union nor the granting authority can be held responsible for them.

COPYRIGHT

© NebulOuS Consortium, 2025

This document may not be copied, reproduced, or modified in whole or in part for any purpose without written permission from the NebulOuS Consortium. In addition to such written permission to copy, reproduce, or modify this document in whole or part, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.

All rights reserved.

CONTRIBUTORS

Name	Organization
Joanna Chmielewska	7bulls.com
Radosław Piliszek	7bulls.com
Michał Soczewka	TTA
Ricard Cervera	EUT
Robert Sanfeliu Prat	EUT
Bruno Pereira	Ubiwhere
Nikos Papageorgopoulos	Ubitech
Asimina Tzana	Augmenta
Francisco Alvarez Terribas	Telefonica Innovacion Digital
Moritz von Stietencron	BIBA
Ferran Diego Andilla	Telefonica Innovacion Digital

PEER REVIEWERS

Name	Organization
Geir Horn	UiO
Fotis Paraskevopoulos	EXZ

REVISION HISTORY

Version	Date	Owner	Author(s)	Comments
0.1	03/09/2025	7bulls	Joanna Chmielewska	First Draft
0.2	28/10/2025	7bulls	Joanna Chmielewska	Second Draft
0.3	07/11/2025	TTA	Michał Soczewka	Input for TTA use case
0.4	12/11/2025	EUT	Robert Sanfeliu Prat, Ricard Cervera	Input for EUT and Mercabarna use case
0.5	28/11/2025	UW	Bruno Pereira	Input for UW use case
0.6	30/11/2025	Ubitech	Nikos Papageorgopoulos	Input for Ubitech
0.7	02/12/2025	BIBA	Moritz von Stietencron	Input for FIRE use case
0.8	03/12/2025	Telefonica	Francisco Alvarez Terribas	Input for Telefonica
0.9	03/12/2025	Augmenta	Asimina Tzana	Input for Augmenta use case
0.10	05/12/2025	UiO	Geir Horn	1st review
0.11	12/12/2025	EXZ	Fotis Paraskevopoulos	2nd Review
0.12	22/12/2025	Augmenta	Asimina Tzana	Final input for Augmenta use case
1.1	23/12/2025	7bulls	Joanna Chmielewska	Final version
1.2	24/12/2025	EUT	Maria Navarro	FINAL

TABLE OF ABBREVIATIONS AND ACRONYMS

Abbreviation/Acronym	Open form
ActiveMQ	Message broker
AMPL	A Mathematical Programming Language
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
AWS	Amazon Web Services
CI/CD	Continuous integration/ Continuous Delivery
CFSB	Cloud Fog Service Broker
CNI	Container Network Interface
CPU	Central Processing Unit
CRD	Custom Resource Definition
CRI	Container Runtime Interface
EFK	Elasticsearch, Fluentd/Fluent Bit, Kibana
EMS	Event Management System
GB	GigaByte
GPU	Graphics Processing Unit
GUI	Graphical User Interface
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
K8s	Kubernetes
KPI	Key Performance Indicator
LLM	Large Language Model
Meta-OS	Meta-Operating System
MCDM	MultiCriteria Decision Making
MQTT	MQ Telemetry Transport
NUC	Next Unit of Computing

OAM	Open Application Model
OPA	Open Policy Agent
OS	Operating System
QoS	Quality of Service
PaaS	Platform-as-a-Service
RAM	Random Access Memory
RBAC	Role-Based Access Control
REST	Representational state transfer
RM	Resource Manager
SAL	Scheduling Abstraction Layer
SCE	Smart Contract Encapsulator
SLA	Service-Level Agreement
SLO	Service-Level Objective
SSD	Solid-State Drive
SSH	Secure Shell protocol
TC	Test case
UAV	Unmanned Aerial Vehicle
UC	Use Case
UI	User Interface
VM	Virtual Machine
VPN	Virtual Private Network
WP	Work Package
YAML	YAML Ain't Markup Language

TABLE OF CONTENTS

LIST OF FIGURES

Funded by the European Union

LIST OF TABLES

EXECUTIVE SUMMARY

This deliverable presents the final outcomes of Work Package 6 (WP6), which focused on the integration, validation, and evaluation of the NebulOuS Meta-Operating System across all pilot demonstrators. The primary objective of WP6 was to ensure that the platform reached the necessary level of functional stability, performance, and operational maturity required for real-world deployments across the cloud–edge continuum.

The NebulOuS platform architecture, including the control-plane and application-cluster environments, has been fully consolidated. Critical runtime intelligence—such as anomaly detection, forecasting, prediction orchestration, and simulation—was relocated to application clusters, ensuring lower latency and improved autonomy. All core components reached their target Technology Readiness Levels. Over the course of WP6 executed tests validated messaging workflows, deployment processes, SLA (Service-Level Agreement) monitoring, reconfiguration logic, and resource orchestration. The migration to GitHub improved development efficiency and transparency, supported by GitHub Actions and Flux-based continuous deployment (CD). The integration approach supports reproducible tests, early fault detection, and consistent deployment workflows.

Each pilot demonstrated that the NebulOuS platform could meet the required functional and non-functional requirements in realistic operational environments. Key Performance Indicators (KPIs) defined in Deliverable D2.1 were assessed, with most being achieved or exceeded.

Key achievements included:

- Near-real-time data processing capabilities.
- Predictive autoscaling mechanisms ensuring SLO compliance.
- Effective resource allocation and reconfiguration across heterogeneous cloud–edge infrastructures.
- Robustness against network fluctuations and device failures.

Overall, the results confirm that the NebulOuS platform has reached the intended maturity for stable deployment and further expansion.

1. INTRODUCTION

This deliverable documents the final release of the NebulOuS Integrated Platform and provides a comprehensive account of its validation and evaluation activities conducted through the project's pilot demonstrators. It represents the concluding outcome of the integration, testing, and assessment work carried out within Work Package 6 (WP6), consolidating the technical progress achieved throughout the project into a single, coherent overview.

The primary purpose of this document is to demonstrate the operational readiness, stability, and maturity of the NebulOuS platform at the end of the project lifecycle. It describes how the platform components have been integrated into a unified system, how their interactions have been verified, and how the platform has been evaluated against both functional and non-functional requirements in realistic usage scenarios. In this context, the deliverable serves as a reference for stakeholders, reviewers, and future adopters interested in understanding the platform's capabilities, design choices, and validation approach.

The scope of this deliverable encompasses the following key aspects:

- **Final platform architecture**: a detailed presentation of the NebulOuS architecture, including its modular structure, deployment model, and the distribution of intelligence between control-plane and application clusters across cloud and edge environments.
- **Final maturity assessment**: a summary of the final verification of component maturity, including references to Technology Readiness Levels (TRLs), key improvements introduced during the final development iterations, and an overall assessment of the platform's readiness for use beyond the project context.
- **Integration and deployment processes**: a description of the integration activities performed in WP6, covering the adopted CI/CD workflow, supporting tools, configuration management practices, and procedures used to ensure deployment consistency, traceability, and reliability.
- **Testing and validation framework**: an overview of the testing strategy applied to the platform, including both manual and automated testing approaches, the definition of test environments, and the methodologies used to validate end-to-end platform behaviour throughout the development lifecycle.
- **Pilot-based evaluation**: documentation of the evaluation activities carried out through the project's pilot demonstrators, presenting how the platform was assessed against functional and non-functional requirements, Key Performance Indicators (KPIs), and operational constraints in representative real-world scenarios.
- **Open Call–based evaluation:** summary of the evaluation carried out through two Open Calls, describing how external projects validated the platform across diverse real-world scenarios, and assessing its capabilities with respect to cloud–edge orchestration, scalability, monitoring, SLO-driven optimisation, and overall operational robustness beyond the core project pilots.

By consolidating architectural descriptions, integration practices, testing methodologies, and evaluation activities, this deliverable provides a complete and structured view of the NebulOuS platform at its final stage, serving both as evidence of project achievements and as a foundation for future exploitation and adoption.

2. NEBULOUS PLATFORM FINAL ARCHITECTURE

The conceptual architecture presented in Deliverable D6.1 remains valid and continues to represent the high-level structure of the NebulOuS Meta-OS. The final release maintains the overall design principles, the logical component groupings, and the user-role abstractions of the conceptual view.

However, the deployment architecture has evolved significantly through the completed integrated cloud-edge continuum prototype. The final distribution of components across the NebulOuS Control Plane and the dynamically created application clusters reflects the functional maturity of the system and the need to move key runtime intelligence closer to where applications execute.

The consolidated deployment view is illustrated in Figure 1, which depicts the full multi-cloud, edge, and fog landscape supported in the final release.

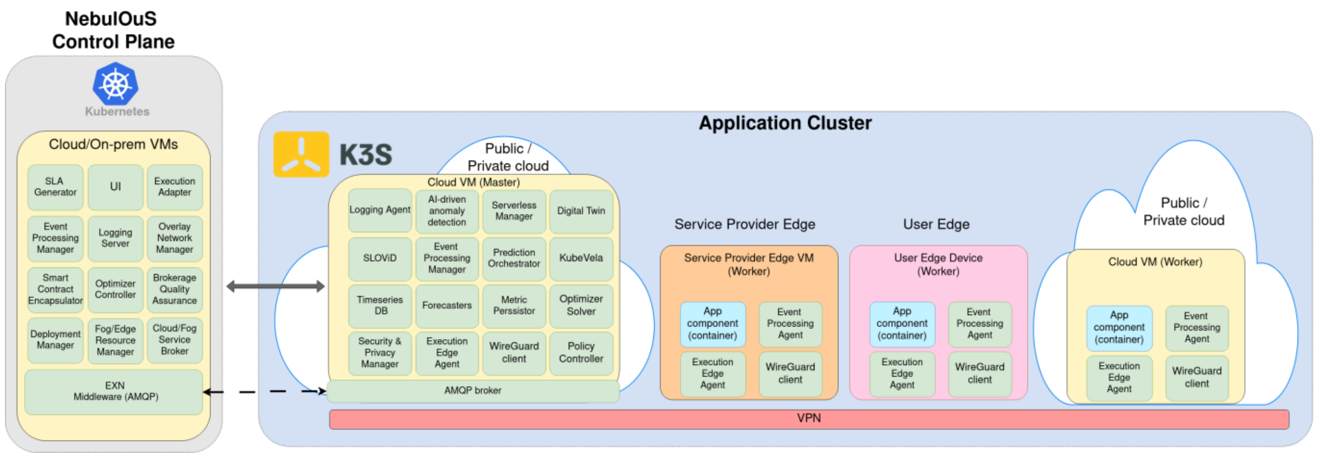


Figure 1. NebulOuS building blocks: Shared components run in the central control plane, while application-specific components run in the application clusters, with most execution logic placed on the Kubernetes master node.

2.1. DEPLOYMENT ARCHITECTURE OVERVIEW

NebulOuS operates through two connected environments, a Control Plane Cluster, which provides global management and orchestration and the Application Clusters, created on demand to host user applications across cloud, fog, and edge resources. Both environments rely on Kubernetes technologies and cooperate through the NebulOuS messaging middleware.

The NebulOuS control plane consists of all platform-wide coordination components common to all applications, which have been fully containerized and deployed on a dedicated Kubernetes cluster. This environment hosts the services that govern the brokerage of resources, reconfiguration processes, SLA handling, deployment orchestration, and user interaction. Control-plane components and NebulOuS components deployed within the application clusters communicate asynchronously via the NebulOuS EXN Middleware, implemented using an ActiveMQ broker exposed over AMQP (Advanced Message Queuing Protocol). Topics and message flows follow NebulOuS-defined conventions that allow loosely coupled interactions between orchestrators, brokers, monitoring services, and optimisation modules.

The control plane is responsible for providing the UI and API endpoints for resource registration, policy definition, and application specification. It orchestrates the resource brokerage and ranking mechanisms while initiating and overseeing the creation of application clusters. In addition, it generates SLAs and their associated smart contracts, coordinates application adaptation through the Optimizer Controller, and supervises the cluster lifecycle, including handling cloud-edge resource onboarding.

Once the control plane is deployed, NebulOuS enables users to deploy and execute containerized applications across the cloud–edge continuum. Eligible resources whether cloud VMs, service-provider edge nodes, or user-edge devices, are onboarded and provisioned by the platform and can then be used to form application clusters. These clusters are Kubernetes-based environments, with the option to be deployed using either standard Kubernetes distributions or the lightweight K3s distribution. They are automatically instantiated by the Execution Adapter and managed throughout their lifecycle by the Deployment Manager.

An application cluster may consist of cloud-only nodes, edge-only nodes, or mixed topologies combining both. The creation of distributed Kubernetes/K3s[1] clusters that federate heterogeneous resources across different administrative domains remains a key advancement of NebulOuS. All application cluster nodes are interconnected through an encrypted overlay network established by the Overlay Network Manager, ensuring secure and seamless communication.

During installation, NebulOuS deploys a CRI (Container Runtime Interface)-compatible container runtime together with Cilium, leveraging its WireGuard-based tunnels for intra-cluster encryption, with Kube-flannel offered as an alternative CNI (Container Network Interface) option. This is followed by the installation of Kubernetes/K3s itself, Event Management System agents, and the Overlay Networking agents. Finally, the system installs KubeVela and Policy Controllers, along with the newly relocated components, AI-driven Anomaly Detection, Security & Privacy Manager, Smart Contract Encapsulator runtime logic, Forecasters, Prediction Orchestrator, Metric Persistor and Timeseries Database.

2.2. RELOCATED AND NEWLY INTRODUCED COMPONENTS IN THE FINAL DEPLOYMENT ARCHITECTURE

In the final release of NebulOuS, a significant part of the execution intelligence has been moved from the control plane into the application-cluster master node, supported by several new components introduced. This relocation ensures faster evaluation of runtime conditions, reduces control-plane load, and enables decentralised responsiveness across hyper-distributed deployments.

[1] https://k3s.io/

The following describes both the components that have been relocated from the control plane to application clusters, and the new components added during the final development cycle.

2.2.1. Components Relocated to the Application Clusters

AI-driven Anomaly Detection: Operates fully within the cluster, analysing node-level, network-level, and application-level behaviour in near real-time. By evaluating metrics such as traffic flow, packet anomalies, and resource deviations directly at the source, it reduces latency and provides early detection of performance degradations or security threats.

Security & Privacy Manager: Although policy definition remains a control-plane function, enforcement, including Policy Controllers for validation the OPA (Open Policy Agent) Gatekeeper[2] and network policy interpretation the Cilium[3], is now performed inside the application cluster. This ensures that admission-control decisions and fine-grained security rules are applied directly to the application components are deployed.

Forecasters: Forecasting models for SLO-related metrics operate locally, predicting near-future behaviour based on cluster-level monitoring data. This improves the accuracy and timeliness of adaptation decisions delivered to the Optimizer Solver.

Prediction Orchestrator: Coordinates forecasting workflows inside the cluster. It aggregates metric streams from EMS agents, invokes forecasting models, interacts with anomaly detection outputs, and prepares prediction sets used by the Optimizer Solver during reconfiguration cycles.

Metrics Persistor: A lightweight metric-buffering layer that stores high-frequency monitoring data locally for consumption by anomaly detection, forecasting, and simulation tools. Where applicable, Netdata is used to collect node-level resource metrics that feed into the Metric Persistor and Timeseries DB.

Timeseries Database: A locally deployed InfluxDB that maintains short-term and medium-term historical metrics. This ensures low-latency data access for anomaly detection, prediction.

2.2.2 New Components Introduced in the Final Release

In addition to the relocated modules, the final release introduces several new components

Digital Twin: This provides a dynamic, data-driven representation of the deployed application, capturing runtime metrics, performance behaviour, and workload patterns. It simulates and estimates the performance of unseen application configurations based on real-time measurements and historical data into the Optimizer Solver to support more accurate, context-aware reconfiguration decisions.

[2] https://open-policy-agent.github.io/gatekeeper/website/

[3] https://cilium.io/

Serverless Manager: Integrating Knative[4] into application clusters, enabling the deployment and autoscaling of serverless functions alongside traditional microservices.

Smart Contract Encapsulator: At runtime, the SCE processes monitoring data originating within the application cluster. It evaluates SLA compliance by executing predefined contract functions and publishes SLA-related events back to AMQP.

Logging Server and Logging Agents: Providing observability through the adoption of an EFK-based logging pipeline. Elasticsearch[5] and Kibana[6] are deployed centrally in the control plane for log indexing, search, and visualisation. Fluentd[7] agents are deployed in each application cluster to collect, filter, and forward logs from application workloads, agents, and cluster services.

3. THE SCOPE OF THE FINAL NEBULOUS RELEASE

Below we present a summary of the final status of the different components.

User Interface: The final version of the User Interface (UI) enables users to perform actions associated with the different platform roles (DevOps, Organization Manager, and External Provider).

First, the UI supports resource management, including cloud accounts and edge resources. This functionality is implemented through a dedicated form and the integration of the Resource Management UI.

The UI also includes a Policy Editor, where organization managers can define organization-wide OPA Gatekeeper policies to be applied to the Kubernetes cluster (for example, restricting container images to trusted sources).
DevOps users can manage their applications by defining, deploying, and monitoring their status. Application definition is guided by a multi-step wizard, where users provide a KubeVela definition of the application and its components, and specify optimization variables. Users can also configure environment variables required for deployment and define application-specific policies enforced by OPA Gatekeeper.

To simplify KubeVela configuration, the UI provides a text-based input that allows users to describe their application in natural language. An LLM then generates the corresponding KubeVela definition based on the user input and NebulOuS-specific context injected into the prompt.

Users can select cloud resources and manage application-specific edge resources through an integrated GUI. They may also specify selection criteria used by Cloud Fog Service Broker CFSB when ranking node candidates for deployment.

[4] https://knative.dev/
[5] https://www.elastic.co/elasticsearch
[6] https://www.elastic.co/kibana
[7] https://www.fluentd.org/

For application monitoring, SLAs, and SLOs, the UI offers a form to capture all necessary information required by other NebulOuS subsystems (Monitoring, SLA, and Blockchain) to perform their respective tasks.

Additionally, the definition wizard enables users to specify the optimization problem to be later evaluated by the solver, from which deployment requirements are derived. Once the application is fully defined, users can request its deployment.

During deployment, the GUI displays the current deployment status (e.g., Pending, Running, etc.). After deployment, users can visualize metrics collected both from the application and from NebulOuS orchestration components, including SLO violations and AI-based anomaly detection results. Experimentally, the GUI offers the possibility to visualize said metrics using a Virtual Reality device.

Finally, users can terminate an application and release the allocated resources as needed.

Fog/Edge Resource Manager: The component offers a GUI, that permits the resource owner and platform administrator to cooperate and to register Service Provider Edge/User Edge nodes. The component manages the automatic installation of necessary software elements to assess the node capabilities and, if successful, register the device on NebulOuS for its brokerage. The system incorporates mechanisms to determine if an edge node has gone offline and handles its de-registration.

Security & Privacy Manager: The component manages the implementation of user-defined OPA Gatekeeper policies across all application clusters created by NebulOuS. Through this mechanism, organization managers can define fine-grained access control and security rules for Kubernetes resources. For example, restricting the sources of container images used within a cluster.

When a user submits a request to a Kubernetes cluster (such as deploying a pod or creating a service), Gatekeeper evaluates the request against all active constraints. If any policy violation is detected, the request is rejected before the change is applied, ensuring compliance and security consistency across the environment.
Users can also embed Cilium network policies within the KubeVela definition. These policies are parsed by the Security & Privacy Manager component and subsequently applied to the corresponding Kubernetes cluster.

Optimizer: Upon reception of an application deployment request from the UI, the Optimizer obtains a list of ranked node candidates from the Cloud Fog Service Broker, valid for deploying the user application. With this, an adapted version of the KubeVela application file is created. Once the application is deployed, the Optimizer Solver listens for SLO Violation messages and solves an AMPL formulated problem to search for a new deployment strategy. If found, this new deployment strategy is sent to the Optimizer Controller who enacts it through a series of interactions with the Deployment Manager and Execution Adapter. The system also monitors the health of the cluster, reactively re-creating any worker nodes that may disappear.

Deployment Manager and Execution Adapter: Together they offer necessary query methods to obtain node candidates that match certain criteria (CPU, RAM, OS, etc…) from Cloud providers (AWS[8] and OpenStack[9]) as well as Service Provider Edge/User Edge nodes. Also, they offer the methods necessary for instantiating nodes from Cloud providers and installing all the software elements to conform the application cluster (Overlay Network Manager agent, EMS agent, etc…).

Overlay Network Manager: The component handles the creation an on-demand VPN (Virtual Private Network) network that connect cluster nodes. This network provides secure node-level connectivity (i.e. VMs or bare metal devices) before the deployment of Kubernetes, ensuring that all intra-cluster traffic will be encrypted for each cluster.

Event Management System (EMS): Once EMS server and agents are installed in an application cluster, it receives a metric model generated by the UI and translates it to the internal structures used later by the EMS Agents. These agents permit to collect generic node metrics regarding resources utilization (CPU, RAM, disk, etc…). EMS Agents are visible by the application components running on each of the application cluster nodes. That allows for collecting application specific metrics either by exposing them using Prometheus endpoints or publishing them using the message broker internal to each EMS Agent. The EMS Agents receives metrics and processing configuration from the EMS Server and publish the processed metrics to the NebulOuS message broker (via EMS server), for other components to consume it. SLO violations are detected and reported to the message broker for further action. To minimise the amount of SLO violation messages sent, a Q-Learning based decision algorithm is employed.

Data Collection & Management System: The use of Apache Activemq for articulating an application pub-sub mechanism has been used. A plugin that collects relevant metrics regarding the utilization of the message broker by the application components has been implemented. These metrics include number of messages waiting in a queue, size of the messages, etc… The plugin also generates events throughout the lifecycle of a message (published, delivered and acknowledged). These metrics and events are published to the Event Management System and can be utilized for defining the SLO of the application and the optimization function. On top of the proposed publish-subscribe mechanism, a solution for orchestrating data processing pipelines has been offered. This solution is based on a YAML oriented mechanism for describing pipelines in terms of steps, inputs and outputs. To realize the pipelines, a plugin has been developed that a) creates necessary topics in the message broker and b) facilitates annotating each message appropriately based on its destination topic or content so it is routed to the correct instance of a step when multiple instances of a step are allowed (e.g.: all messages for the same vehicle must be processed by the same step instance).

Cloud Fog Service Broker: A mechanism has been implemented to allow users to rank the offering from different Cloud service providers as well as Edge nodes. The user can select from a wide set of attributes (related to usability, security, reputation, cost, etc…), the ones which are relevant to be used as criteria for the evaluation of available nodes. Also, the user can express his or her preferences and the component then generate a ranked list of available nodes. This ranking is considered by the

[8] https://aws.amazon.com/?nc2=h_home&refid=a783f7d4-42ee-494a-abcf-dff80bf02d1d
[9] https://www.openstack.org/

Optimizer to select the nodes to be used to deploy a user application. The algorithms used have been optimised to reduce the computational cost of the assessment and, thus, reducing the time required.

Brokerage Quality Assurance: The NebulOuS Brokerage Quality Assurance mechanism ensures that automatically generated SLAs are both internally consistent and compliant with meta-quality constraints governing service delivery across the Cloud-Edge Continuum. This mechanism supports a two-tier validation process—introspective and extrospective—implemented through ontology-driven reasoning.

SLA Generator: Service Levels (SLs) are used for modelling the fluctuating level of resources that is often provisioned in near-edge environments. They may also be used for defining compensating actions that transition between different SLs when certain performance-related threshold values are superseded or violated, thus implementing service degradation schemes. The component monitors Service Level Agreements (SLAs, a composition of smaller SLs) defined by the user on the GUI and enables automatic SLA transitions and settlements based on the metrics collected from the computing nodes.

Smart Contract Encapsulator: Upon the reception of a new application SLA (as generated by the SLA Generator), a smart contract derived from the SLA is generated and recorded in a Hyperledger Fabric[10] blockchain. Then component listens to the necessary metric topics and starts monitoring the SL based on the metrics received and the SLA definition. Any degradation of the SLA (based on the rules defined in the SLA) is detected and reported to the blockchain.

AI-driven Anomaly Detection: The AI-driven anomaly detection mechanism monitors network-level behaviour to identify deviations and proactively anticipate potential SLO violations. It analyses network metrics in near real time, including traffic volume (received_packets, sent_packets, received_bytes, sent_bytes) and transmission errors (dropped_packets, errors, collisions, fifo_errors, frame_errors). The results are stored in InfluxDB and can be visualised through the NebulOuS UI, providing actionable insights for maintaining service reliability.

Serverless: Knative has been integrated into the NebulOuS, permitting users to define applications that include a serverless executor (the component responsible for executing the serverless functions) and multiple serverless functions. NebulOuS allows the user to define monitoring metrics on the serverless functions performance (most importantly, the number of concurrent serverless requests) and scale the serverless executor vertically to adapt to the workload.

Workflows: NebulOuS has designed an architecture that utilises Argo Flow[11] for orchestrating computing workflows. Argo Flow is a Kubernetes-native engine that executes complex, parallel, and event-driven workflows defined as Custom Resource Definitions (CRD) in YAML. The workflow executor, ProActive[12] scheduler, and Prometheus[13] agents are deployed to receive workflow requests as XML files, schedule to the appropriate worker nodes, and monitor and export the metrics to the Event Management System to drive application reconfiguration.

[10] https://hyperledger-fabric.readthedocs.io/en/latest/blockchain.html
[11] https://argoproj.github.io/workflows/
[12] https://github.com/ow2-proactive
[13] https://prometheus.io/

As a summary, the final version of the NebulOuS meta-OS permits users to manage resources to be brokered by NebulOuS. This involves the capability of registering Cloud providers and User Edge/Service Provider Edge resources and express the criteria used to rank them. Once the resources are registered, the user can define applications to be managed by NebulOuS. For this, the GUI guides the user on specifying the KubeVela application manifest file, its metric model, SLO, and the optimization criteria. Once an application is defined, the users can initiate deployment, whereby NebulOuS selects the appropriate deployment topology using the brokered resources based on user preferences. With this, NebulOuS components collaborate to automatically enact the decided deployment topology. This involves creating VMs on Cloud providers, installing the necessary software elements on all the nodes of the application cluster (Cloud and/or Edge), connecting the nodes through a VPN and finally deploying the application Kubevela. Once the app is deployed, the Event Management System captures relevant application metrics and aggregates them. EMS is also capable of detecting SLO violations. Upon a detection of SLO violation, the Optimizer Solver searches for a new deployment topology that mitigates this SLO violation and, if found, sends appropriate messages to Optimizer Controller. Then, the Optimizer Controller automatically enacts the application re-configuration. This involves removing nodes that are no longer part of the topology and/or adding new nodes to it. Again, the process of adding nodes to the topology requires NebulOuS to automatically create VMs on cloud providers (if necessary), installing the necessary software elements on the new nodes and joining these new nodes to the already existing application cluster VPN is repeated. User is also able to define service level agreements (SLA) agreed with the resource provider. These SLA are monitored by the Smart Contract Encapsulator and, any degradation of the optimal SL to lower SLs is stored in the blockchain.

To assess the maturity and deployment readiness of technologies, the project adopted the Technology Readiness Level (TRL) methodology, which allowed systematic measurement of progress throughout the development lifecycle. In the previous deliverable D6.1, we provided a detailed overview of the status of the platform components as they stood at the time of the first release. This initial assessment reflected the level of completeness, integration, and functional maturity that had been reached during the early deployment phase. In contrast, the present document offers an updated and comprehensive view of the final status of each component at the conclusion of the project.

The table below presents components together with their initial TRL values at the start of the project, their final TRLs at completion, and the maturity level targeted for the official release. Additionally, the table highlights the key enhancements and technical advancements achieved during the project's execution. This evaluation confirms that the components have reached the required level of robustness, integration capability, and operational reliability necessary for real-world deployment.

Table 1. Components, Authors, Initial TRL, Final TRLs and Maturity of the components

Component	Author	Initial TRL	Final TRL	Advancements
GUI	EXZ	-	6	Developed and dedicated for NebulOuS, based on requirements described in more detail in D 2.1
EXN Middleware	EXZ	-	6	This component, along with its adjacent libraries, has been developed to ensure abstraction and adaptability to any number of architectures, such as that of NebulOuS. The integration and implementation details have been worked to an extend that it makes it trivial for any developer to adapt to an asynchronous event driven communication pattern.
Optimizer	UiO	2	6	The architecture of the whole optimization flow developed in MELODIC was rethought and reimplemented to be fully distributed using AMQP messages and AMPL optimisation problem definition. See D3.1 and D3.2 for details.
EMS	ICCS	4[14]	6	Significant extensions and new features were introduced towards continuum monitoring capabilities, among which: a new EMS Translator for supporting the NebulOuS metric model, new Metric Model Validator, support for composite constants, support for plugins for pre-processing and post-processing of metric models, context-aware metric event propagation, k8s-based deployment, Prometheus / OpenMetrics[15] endpoints leveraging, support for retrieving Kubernetes pods and nodes lists from the Kubernetes API server, etc. Described in more detail in D 5.1 and D5.2.
SLOViD	ICCS	4[16]	5	Added multi-application handling functionality (requiring a major refactoring of the code) and elementary Spring support among others. Described in more detail in D 5.1 and D5.2.
CFSB	ICCS	2	6	Reused and extended a preference model for comparing different continuum resources, while the component was developed from scratch, integrating the Data Envelopment Analysis with multi-objective programming methods to cope with the NebulOuS requirements. Described in more detail in D 3.1 and D3.2.
Security and Privacy Manager	UBI	-	5	Developed from scratch for NebulOuS, based on requirements described in more detail in D 2.1. Implements the necessary functionality to create cluster-

[14] MORPHEMIC https://www.morphemic.cloud/
[15] https://openmetrics.io/
[16] MORPHEMIC https://www.morphemic.cloud/

				wide security policies and enforce them. Described in more detail in D4.1 and D4.2.
Deployment Manager	AE	4[17]	6	Extended with cloud continuum deployment capabilities Described in more detail in D 4.1 and D4.2.
Execution Adapter	AE	4[18]	6	Extended with cloud continuum deployment capabilities Described in more detail in D 4.1 and D4.2.
Overlay Network Manager	UBI	-	5	Developed from scratch for NebulOuS, based on requirements described in more detail in D 2.1. Implements the necessary functionality to automatically create and manage WireGuard-based VPNs between cluster nodes. Described in more detail in D4.1 and D4.2.
Data Collection and Management System	EUT	-	6	Developed and dedicated for NebulOuS based on requirements described in more detail in D 2.1
Smart Contract Encapsulator	UiO	-	5	Developed from scratch, stores SLA related data in a Hyperledger Fabric.
Brokerage QA	SEERC	-	5	Developed from scratch, based on preexisting semantic models.
SLA Generator	SEERC	-	5	Developed from scratch, based on preexisting semantic models.
AI-Driven Anomaly Detection	EUT	2 (Utilities)	5	PoC of a solution based on the selected algorithms with data from the Utilities environment, performing laboratory tests and confirming the defined hypotheses. Based on these results, the solution is formulated. It is designed to identify deviations in service behaviour and to proactively anticipate potential SLO violations through near real-time analysis of metrics derived from IoT devices and application workloads. Described in more detail in D5.1 and D5.2.

[17] MORPHEMIC https://www.morphemic.cloud/
[18] MORPHEMIC https://www.morphemic.cloud/

4. NEBULOUS PLATFORM FINAL INTEGRATION

Since the previous deliverable report on integration, the project has migrated from Gerrit- and Zuul-based OpenDev to GitHub (with their proprietary stack). The consortium decided that the lack of experience with the OpenDev platform is contributing to slower development and thus decided to move to a more widely known platform in spite of its proprietary aspects. This migration has affected the processes slightly and the new approach is described in more detail in the following sections. However, the main idea stayed the same and is reiterated in the next paragraph. The new home for NebulOuS development and integration is https://github.com/eu-nebulous

Achieving Technical Integration
To address the concern of the technical integration level, we have implemented several key practices:

- **Environment Consistency**: By maintaining distinct environments for production, testing, development, and continuous deployment (CD), we ensure that each stage of the development lifecycle is supported, and issues are identified early.
- **Comprehensive Documentation**: All components and their deployment processes are well-documented, providing clear guidelines for developers and maintainers.
- **Monitoring and Reporting**: Continuous monitoring of the continuous integration/continuous deployment (CI/CD) pipeline and production environment provides insights into system performance and health, allowing for proactive issue resolution.

4.1. Integration Flow of the NebulOuS Platform

1. Code Submission and Review

- **Developer Submits Code**: Developers submit their code changes to one of the NebulOuS repositories in GitHub as new Pull Requests (PRs).
- **GitHub PR Review**: The submitted code changes are reviewed in GitHub. Reviewers evaluate the code, leave comments, and approve or request modifications. Approval in GitHub is mandatory before further processing.

2. Automated Testing and Verification

- **Isolated Environments**: GitHub Actions (GitHub's CI implementation) executes the defined jobs in isolated environments (containers) to ensure consistency and reproducibility.
- **Testing**: The code change undergoes a series of automated tests. These tests include unit tests, integration tests, and end-to-end tests to verify the functionality and performance of the changes.
- **Feedback**: Test results are collected, and GitHub provides feedback to developers, indicating whether the tests pass or fail. Developers can address any issues identified during this phase.

3. Merging Approved Changes

- **Merging**: Once the change both passes the automated checks and is approved, it gets possible to be merged.

4. Continuous Deployment

- **Artifacts building and publishing**: Artifacts dependent on the repository (e.g., container images, language-specific libraries) are built and published in their respective public repositories (such as Quay.io19 for container images).
- **Flux Monitoring**: Flux continuously monitors the source code repositories for changes. When a change is detected in the main or release branch, Flux triggers the deployment process.
- **Kubernetes Deployment**: Flux integrates the changes into the Kubernetes cluster, deploying the updated components across the appropriate environments.

5. TESTING AND QUALITY ASSURANCE OVERVIEW

This chapter demonstrates that the testing strategy applied to the NebulOuS platform was both systematic and progressively comprehensive, covering the core functional paths, critical non-functional properties, and the most relevant failure and recovery scenarios of the platform. The defined set of manual and automated test cases focuses on end-to-end platform behaviour, including installation, resource onboarding, application deployment across cloud and edge resources, monitoring, SLO violation handling, optimisation-driven reconfiguration, and security enforcement. Taken together, these test cases provide strong coverage of the platform's primary operational features and validate the correct interaction between its major subsystems.

From a qualitative perspective, the test set can be considered functionally complete with respect to the project's scope and maturity level. The tests explicitly target the platform capabilities defined in earlier requirements deliverables, ensuring traceability between requirements, test cases, and validation outcomes. While additional edge cases could theoretically be tested—such as large-scale stress testing with higher numbers of clusters, or deeper fault injection inside application clusters—these scenarios were intentionally deprioritised, as they would require infrastructure and automation complexity beyond the project's timeframe and objectives. Importantly, no critical functional areas identified in the project scope remain untested.

Deliverable D6.1 distinguished two execution modes - manual testing, conducted by Telefónica, and automated testing, performed by Ubitech - both of which are essential for verifying software robustness and compliance with specifications.

Manual test cases proved particularly valuable for validating user-facing workflows, GUI behaviour, and scenarios that are difficult to automate reliably, such as partial infrastructure failures, node disconnections, or interactive resource onboarding. Although manual execution required close coordination with development teams due to frequent platform evolution, it enabled early

[19] https://quay.io/

identification of usability issues, missing feedback in the UI, and insufficient error reporting—several of which directly informed refinements in the final release.

Automated test cases played a decisive role in improving platform stability and integration quality. Their integration into the CI/CD pipeline ensured that core deployment, brokerage, optimisation, scaling, and security workflows were continuously validated as code evolved. In practice, these automated tests successfully detected integration issues at early CI stages, including misaligned message schemas, broken brokerage components where messages were not sent or received, the tests verify the presence of a response rather than its specific optimality, and regressions in deployment workflows. This significantly reduced the likelihood of broken releases and increased confidence that new versions of the platform could be deployed "out of the box" without manual intervention.

Overall, the combined manual and automated testing approach proved effective in balancing depth, realism, and automation. While some complex runtime scenarios remain candidates for future extensions—such as fully automated chaos testing or deep in-cluster fault injection—the current test coverage is adequate and well-aligned with the goals of WP6. The results presented in Chapter 5 therefore provide credible evidence of the platform's readiness, robustness, and suitability for real-world deployment within the validated use cases

To manage the bug reporting process effectively, we changed the platform from Launchpad to GitHub**,** for tracking the full lifecycle of test cases and related issues. All discovered defects were reported through GitHub, accompanied by detailed reproduction steps, logs, and supporting evidence. When reporting a new issue, contributors were required to identify the correct repository based on the scope and the affected elements. If the issue concerned a specific component, it had to be reported in that component's dedicated repository. If it affected multiple components or the platform as a whole, the issue was submitted to the main NebulOuS platform repository. This approach ensured that each reported problem reached the appropriate development team without unnecessary delays.

Test cases marked with an asterisk (*) have been assigned for both manual and automated execution in order to broaden the scope of testing.

5.1. MANUAL TEST CASES

Manual tests refer to the manual interaction of a NebulOuS end user from the installation of the platform to the deployment of applications in the cloud-edge continuum. These manual tests have been executed during the project aiming to stablish solid quality assurance procedures that guide platform component development.

5.1.1 Summary of Work Performed

In the beginning of the project the tests have been performed in an absolute manual way but, due to reproducibility matters the tests related to interactions with the NebulOuS GUI were implemented using Selenium[20]. This open-source project is a set of libraries specifically designed to simulate real

[20] https://www.selenium.dev/

Funded by the European Union

user actions across different web browsers and operative systems mainly used to perform efficient and scalable functional testing. This approach is very interesting because the speedup in running the tests was significant but, the biggest downside of it was that every time new GUI code was pushed most of the tests had to be checked. Also, it is important to mention that during the development of the platform new features were pushed constantly to production while other already implemented features were modified, and this made the testing challenging because the testing team needed to stay in constant contact with all the development teams.

Table 2. List of the test cases prepared for manual execution. Test cases marked with an asterisk (*) have been assigned for both manual and automated execution.

Test Case	SUMMARY	STATUS and REMARKS
TC_01	NebulOuS build	All resources necessary for installing NebulOuS core and agent can be generated.
TC_02	NebulOuS core installation	NebulOuS core can be installed.
TC_03	NebulOuS core installation fails when requirements are not met.	NebulOuS core will not initialize all components if it does not have sufficient resources.
TC_04	NebulOuS core recovers after failure of any of its component	NebulOuS core resumes normal operation of failed components.
TC_05	Uninstall NebulOuS core	NebulOuS core can be uninstalled.
TC_06	Login into NebulOuS web UI (invalid credentials)	NebulOuS web UI shows an alert indicating that the credentials are invalid.
TC_07	Login into NebulOuS web UI (valid credentials)	NebulOuS web UI logs-on user and redirects user to home screen.
TC_08	Logout from NebulOuS web UI	NebulOuS web UI logs-off user and redirects the user to login screen.
TC_09	Unauthenticated user can't interact with NebulOuS web UI	NebulOuS web UI shows an alert indicating that the user is not logged in and redirects the user to the login screen.
TC_10	Resource manager can onboard manually-managed nodes	Resource manager can onboard manually-managed nodes in NebulOuS
TC_11	Onboarding of a manually-managed node fails	User is properly informed in case of onboarding failure.
TC_12	Device owner can de-register his devices	Device owner can de-register a device.

TC_13	Resource manager can de-register devices	Device owner can de-register devices.
TC_14	Admin can install NebulOuS agent.	User can install NebulOuS agent; relevant information is provided to user and the new manually-managed node is visible from the administration panel.
TC_15	NebulOuS agent installer fails when requirements are not met.	User can see relevant logs related to agent installation failure.
TC_16	NebulOuS agent can be uninstalled.	NebulOuS Agent can be uninstalled from device.
TC_19	Admin can register cloud providers in NebulOuS.	Admin can register cloud providers in NebulOuS.
TC_20	Admin can de-register cloud providers in NebulOuS.	Admin can de-register cloud providers in NebulOuS.
TC_21 *	App deployment on manually-managed nodes	User cannot see application deployment logs as well as application execution logs from the web UI.
TC_22 *	App deployment on manually-managed node fails due to lack of resources	No sufficient information is provided to the user to identify the problem.
TC_23 *	App deployment using NebulOuS cloud providers	There are no application deployment logs provided to user from the web UI.
TC_24 *	App deployment on NebulOuS-managed node fails due to lack of resources	Failure during node registration, adequate information is provided to user.
TC_32	Validation of RBAC Policies in NebulOuS on a Kubernetes Cluster	Operations are allowed or denied in accordance with the defined RBAC policies, and relevant logs are generated.
TC_37 *	Cloud provider selected based on user preferences	Tested manually by setting user preferences in the CFSB UI while two cloud providers were registered and verifying the selected provider.

5.2. AUTOMATED TEST CASES

Throughout the project, an extensive automated testing framework was designed, implemented, and integrated into the NebulOuS CI/CD pipeline to validate the platform’s end-to-end behaviour during

Funded by the European Union

application deployment, optimisation, reconfiguration, and resource orchestration. The testing solution was built using the Citrus Framework[21], enabling the automation of complex integration tests that exercise NebulOuS components through AMQP messaging, REST APIs, and cloud/edge resource provisioning workflows.

5.2.1 Summary of Work Performed

The test suite interacts with the platform through: AMQP endpoints (ActiveMQ topics used by NebulOuS middleware), Scheduling Abstraction Layer (SAL) REST APIs, Resource Manager APIs, Utility APIs used for optimisation and metric evaluation.

All endpoints used in the tests are centrally defined via *NebulousEndpointConfig.java*, ensuring that the messaging configuration used in testing mirrors the production system.

For example, the *application creation topic* is defined as:

```
@Bean
public      JmsEndpoint      appCreationEndpoint(ConnectionFactory
connectionFactory) {
    String                appcreationDestination                =
"eu.nebulouscloud.ui.dsl.generic";
    return CitrusEndpoints.jms()
            .asynchronous()
            .connectionFactory(connectionFactory)
            .destination(appcreationDestination)
            .pubSubDomain(true)
            .autoStart(true)
            .build();
}
```

And the sending and the evaluation received message is defined as:

```
NebulousCoreMessage        appCreationMessage        =        new
NebulousCoreMessage(appCreationPayload, appCreationEndpoint);
messageSender.sendMessage(appCreationMessage);
$(receive(appCreationEndpoint)
        .message()
        .selector(selectorMap)
        .validate((message, context) -> {
            logger.debug("appCreationPayload payload received");
        }));
```

[21] https://citrusframework.org/

5.2.2 Core Testing Pipeline

The automated pipeline emulates the full lifecycle of an application deployment in NebulOuS. The full procedure described in this section is illustrated in Figure 2 in which presents the activity diagram corresponding to the end-to-end execution of the automated test workflow.

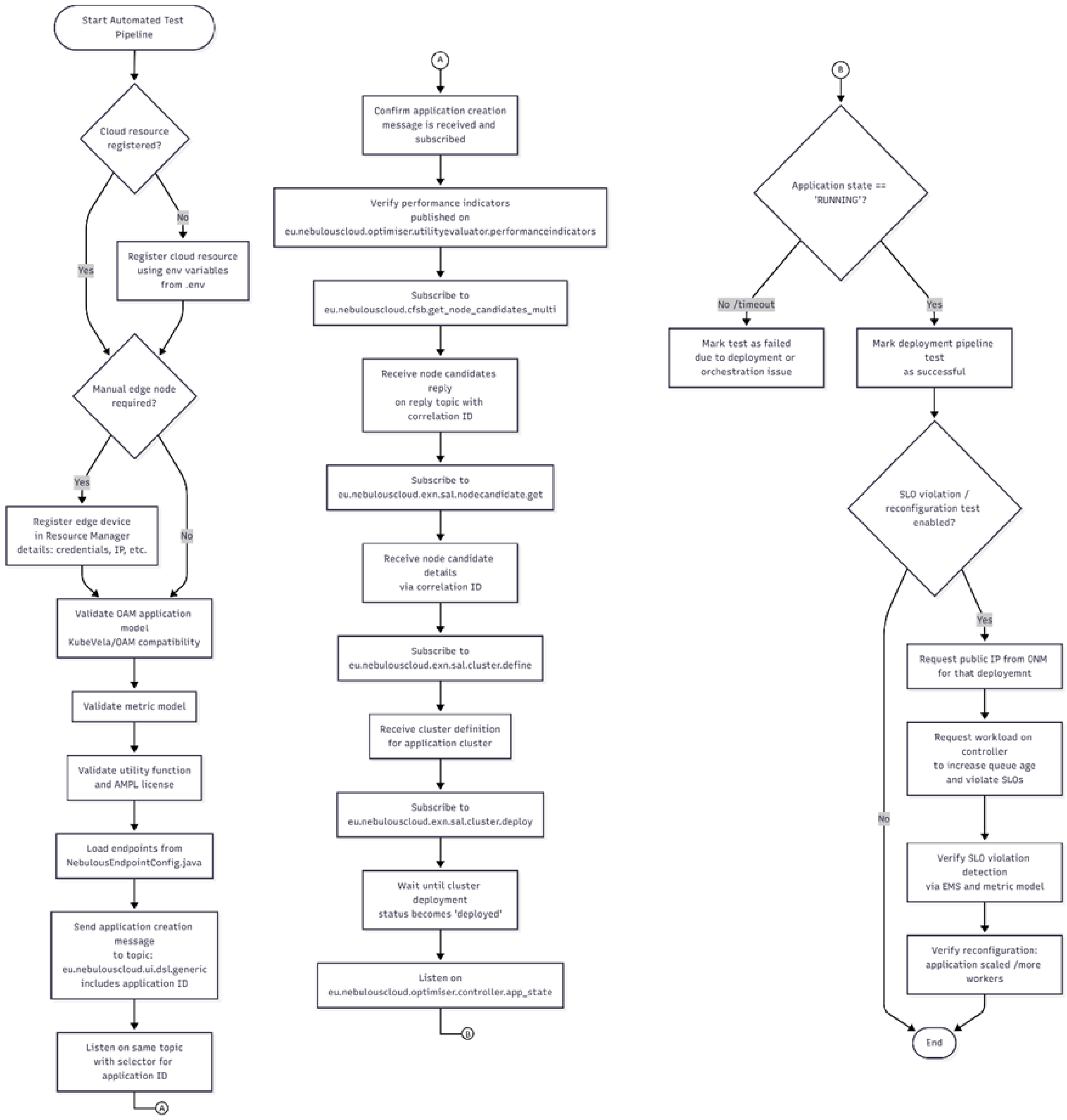
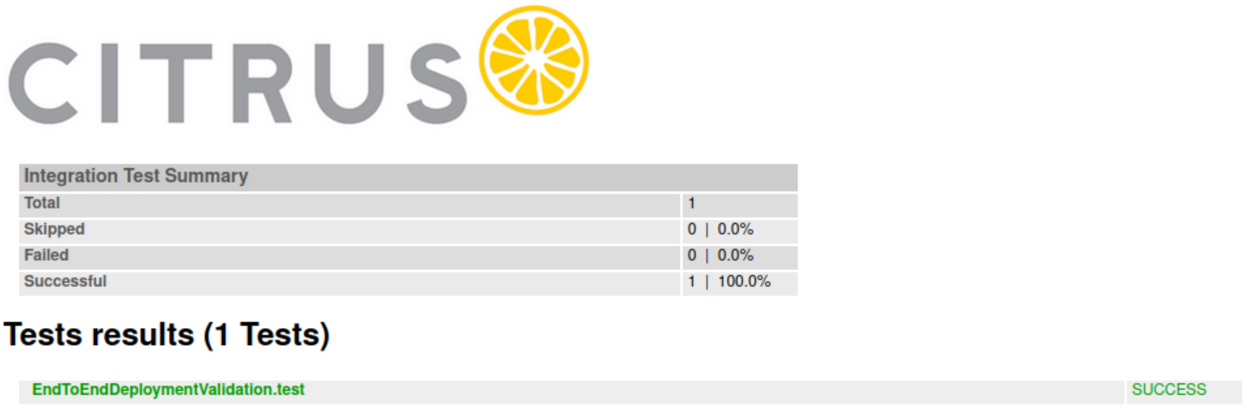

Start Automated Test Pipeline
Cloud resource registered?
No
Register cloud resource using env variables from .env
Yes
Manual edge node required?
Yes
Register edge device in Resource Manager details: credentials, IP, etc.
No
Validate OAM application model KubeVela/OAM compatibility
Validate metric model
Validate utility function and AMPL license
Load endpoints from NebulousEndpointConfig.java
Send application creation message to topic: eu.nebulouscloud.ui.dsl.generic includes application ID
Listen on same topic with selector for application ID
A
A
Confirm application creation message is received and subscribed
Verify performance indicators published on eu.nebulouscloud.optimiser.utilityevaluator.performanceindicators
Subscribe to eu.nebulouscloud.cfsb.get_node_candidates_multi
Receive node candidates reply on reply topic with correlation ID
Subscribe to eu.nebulouscloud.exn.sal.nodecandidate.get
Receive node candidate details via correlation ID
Subscribe to eu.nebulouscloud.exn.sal.cluster.define
Receive cluster definition for application cluster
Subscribe to eu.nebulouscloud.exn.sal.cluster.deploy
Wait until cluster deployment status becomes 'deployed'
Listen on eu.nebulouscloud.optimiser.controller.app_state
B
B
Application state == 'RUNNING'?
No /timeout
Mark test as failed due to deployment or orchestration issue
Yes
Mark deployment pipeline test as successful
SLO violation / reconfiguration test enabled?
Yes
Request public IP from ONM for that deployemnt
Request workload on controller to increase queue age and violate SLOs
Verify SLO violation detection via EMS and metric model
Verify reconfiguration: application scaled /more workers
No
End

Figure 2. End-to-end execution of the automated test workflow

An application creation message is initially sent under the topic *eu.nebulouscloud.ui.dsl.generic*. After sending an application creation message with the corresponding application ID embedded in its header, the test waits for NebulOuS to subscribe to that ID and confirm that the message has been correctly received. The test framework evaluates this by sending the message and then using Citrus to monitor the same topic, applying a selector that matches the application ID and confirming that the message was processed. The central testing pipeline validates the full deployment sequence of an application. The first step is to verify that the middleware has correctly received and propagated the application creation message. The pipeline then examines whether the Optimiser Utility Evaluator has published performance indicators on the topic *eu.nebulouscloud.optimiser.utilityevaluator.performanceindicators*. It next queries the Cloud Fog Service Broker by subscribing to *eu.nebulouscloud.cfsb.get_node_candidates_multi*, checking whether node candidates exist for the application, and validating the expected response on the associated reply *(_reply)* topic. The Execution Adapter is contacted in a similar manner by listening on *eu.nebulouscloud.exn.sal.nodecandidate.get*, where the test waits for the node candidate information required for cluster construction. Once the necessary resources have been identified, the test monitors the cluster definition topic *eu.nebulouscloud.exn.sal.cluster.define* and subsequently waits for the cluster deployment to progress to the "deployed" state through updates published on *eu.nebulouscloud.exn.sal.cluster.deploy*. The final stage of the pipeline confirms that the deployed application transitions to the "RUNNING" state, as reported on the topic *eu.nebulouscloud.optimiser.controller.app_state*.

The automated tests are maintained in the dedicated repository at https://github.com/eu-nebulous/tests/tree/test-automated. The repository includes the *.env* file that drives the configuration of cloud and edge resources, the AMQP broker parameters and the integration URLs of the various NebulOuS components. This allows the test suite to be executed both locally and within the CI/CD environment using the same configuration.

During the development of the automated testing pipeline, several practical limitations were encountered. Certain test cases could not be fully automated because they required manual interventions, such as intentionally disconnecting nodes or removing resources during runtime, actions that cannot be reproduced reliably within a CI environment. In addition, some procedures depended on observing events that occurred exclusively inside the application clusters and were not exposed through the messaging middleware. Automating these checks would have required establishing SSH access into each cluster node, which was outside the scope of the automated framework and not compatible with its design principles.


Integration Test Summary

Total	1
Skipped	0 | 0.0%
Failed	0 | 0.0%
Successful	1 | 100.0%

Tests results (1 Tests)

EndToEndDeploymentValidation.test SUCCESS

Figure 3. The successful automated test execution illustrated in the Citrus test report

The successful execution of the automated integration tests is illustrated in the Citrus test report shown in Figure 3. The report confirms that the EndToEndDeploymentValidation test completed without errors, demonstrating that the full application-deployment workflow was validated as expected. Citrus produces these summaries automatically at the end of each test run, providing a clear indication of test success, failures, and overall platform integration stability within the CI/CD pipeline.

Table 3. List of the test cases prepared for automatic execution. Test cases marked with an asterisk (*) have been assigned for both manual and automated execution

Test Case	Summary	Status and Remarks
TC_17	NebulOuS agent recovers after failure with communication with NebulOuS core.	Tested manually by restarting the WireGuard agent on the cluster master node. After connectivity was restored, the EMS successfully reconnected to the middleware and resumed publishing metrics.
TC_18	NebulOuS agent recovers after one of its components fails.	Tested automatically by deleting the EMS pod and observing its recovery behaviour. The EMS reconnected through the *eu.nebulouscloud.ems.boot* topic as expected.
*TC_21 **	App deployment on manually-managed nodes	Tested automatically by registering an edge device through parameters provided in the .env file and executing the full deployment pipeline.
*TC_22 **	App deployment on manually-managed node fails due to lack of resources	Tested automatically by using a registered edge device with insufficient capacity and confirming that the CFSB returned a "no suitable node candidates" response.
*TC_23 **	App deployment using NebulOuS cloud providers	Tested automatically as part of the full deployment pipeline using cloud resources defined in the test environment.

*TC_24 **	App deployment on NebulOuS-managed node fails due to lack of resources	Tested automatically by requesting unrealistically high CPU and memory values and verifying that CFSB reported insufficient resources.
TC_25	NebulOuS reacts to node failure	Could not be automated because the test required live cluster-level details and manual intervention to simulate node loss.
TC_26	NebulOuS scales up and down applications to comply with SLO	Tested automatically using the example application that generates SLO violations, triggering the Optimizer to scale the worker components accordingly.
TC_28	NebulOuS manages the execution of IoT data processing pipelines	Tested automatically by deploying an IoT data-processing workflow and validating message flow through its dedicated broker topics.
TC_29	Secure Deployment of Applications via NebulOuS	Tested automatically by including a security policy in the application creation message and verifying successful enforcement.
TC_31	Network Isolation and Security Policy Compliance in Application Deployments	Tested manually by observing Overlay Network Manager logs and confirming isolation behaviour.
TC_35	Enforcement of Ingress and Egress Network Policies in Kubernetes	Tested automatically by embedding network policies in the application creation message and validating enforcement by Gatekeeper/Cilium.
*TC_37 **	Cloud provider selected based on user preferences	Tested manually by setting user preferences in the CFSB UI while two cloud providers were registered and verifying the selected provider.

6. PILOT DEMONSTRATORS' FINAL EVALUATION

This section provides a consolidated overview of how the NebulOuS platform components were adopted, deployed, and assessed across the six pilot use cases. While each use case was driven by specific domain requirements, a number of common patterns, challenges, and technological needs emerged. These insights provide important conclusions regarding the maturity, flexibility, and effectiveness of the NebulOuS platform in real operational environments.

Across all pilots, the objective of integrating NebulOuS was to validate its capability to support cloud–edge continuum deployments, ensure performance efficiency, optimise resource utilisation, and maintain service reliability under diverse operational conditions. The use cases span a wide range of scenarios – from wind turbine inspections and computer vision analytics through logistics and agriculture, to emergency response management – collectively providing a comprehensive evaluation spectrum for the platform.

Each pilot defined and applied utility functions to drive the optimisation mechanisms within NebulOuS. Although the criteria varied depending on business needs, there was a strong convergence around the goal of achieving optimal performance at the lowest possible resource and financial cost. In practice, this meant:

- Minimising processing time of critical tasks (e.g., image analysis, vehicle event processing, license plate recognition).
- Efficient use of resources such as CPU, GPU, RAM, and network.
- Deployment decision-making between local edge resources, service provider edge, and public cloud based on workload and conditions.
- Selecting deployment options to improve reliability or availability where relevant (e.g., FIRE use case).

The optimisation functions developed by each pilot capture the contextual trade-offs of their scenario (e.g., maintaining a stable image-processing pipeline vs. processing speed vs. cost). As a result, the platform demonstrated meaningful flexibility in adapting optimisation goals to scenario-specific requirements, proving the versatility of NebulOuS' optimisation and orchestration layer.

To enable autonomous optimisation and enforcement of SLOs, each use case defined relevant system-level and custom application-level metrics. System metrics primarily measured the utilisation of computational resources (CPU, GPU, RAM), while custom metrics reflected the functional KPIs of each pilot, including:

- Image queue size and average image processing time (Windmill Maintenance).
- Vehicle license plate processing time and Frames Per Second (FPS) for video analytics (Mercabarna use cases).
- Processing duration for individual events, routing requests, or agricultural analysis steps.
- Age of information and application UI response time (FIRE).

These metrics were used by the EMS/SLOV to monitor platform behaviour, detect performance degradation, and trigger adaptation policies when SLOs were violated. The inclusion of tailored metrics was key to ensuring that optimisation decisions were meaningful and aligned with the real needs of each use case.

When evaluating the platform using the Cloud Fog Service Broker (CFSB) perspective, cost reduction and performance efficiency emerged as the most important dimensions across nearly all pilots. The ability to dynamically adjust deployment according to cost–performance trade-offs was consistently reported as a core benefit of NebulOuS.

Some pilots additionally placed emphasis on:

- Proximity (processing closer to the data source to reduce latency and network costs).

- Availability and reliability (particularly important for emergency or time-critical applications such as FIRE).

A strong preference emerged for hybrid edge–cloud deployments, as opposed to edge-only or cloud-only setups. The pilots reported that the most effective configurations typically involved:

- Latency-sensitive or high-frequency preprocessing on local or service provider edge devices.
- Computationally heavy processing, post-processing, or long-term storage in the cloud.

This validated NebulOuS' role as an enabler of dynamic and distributed deployments, capable of leveraging heterogeneous infrastructure efficiently and transparently.

The pilots implemented diverse data flow strategies, demonstrating the flexibility of NebulOuS to support different computational workflows:

- Continuous streaming and local edge processing of imagery or video feeds.
- Event-based workflows triggered by data uploads (e.g., after a task completes).
- Use of IoT pub-sub communication for exchanging sensor readings and operational events.
- Hybrid processing pipelines where initial inference is done at the edge, followed by cloud-based enrichment.

These variations showcased the platform's ability to support both real-time and batch/event-driven processing models, confirming its adaptability across industry domains.

To verify the project requirements, each use case prepared specific files and models that were then loaded into the NebulOuS GUI and used during deployment. NebulOuS users define three main elements:

- **KubeVela application manifest files**, which describe hyper-distributed application components, their operational behaviours, deployment policies, and workflows across hybrid environments such as cloud, Kubernetes, or IoT devices.
- **Metrics and Requirements**, where metrics represent measurable system attributes collected as events, and requirements (currently expressed as Service-Level Objectives) define acceptable value ranges. Violations indicate degraded service performance and the needs for reconfiguration.
- **Optimisation Goals**, which express deployment and scaling preferences using an AMPL-based model. This model evaluates configuration utility, enforces constraints, and determines when alternative deployment configurations are better than the current configuration.

Throughout the duration of the project, regular online coordination meetings were held, taking place almost every Wednesday, to monitor progress and coordinate activities within this work package. In total, 93 such weekly meetings were conducted, providing a continuous forum for status updates, issue tracking, and alignment among the involved partners.

Additionally, a series of sessions were convened involving both the individual use case partners and the corresponding technical partners. These meetings were held on-site (Warsaw, Athens, Barcelona) or online, with the objective of facilitating knowledge exchange, clarifying implementation challenges, and providing targeted guidance. As a result, the sessions played a crucial role in enabling the use case partners to prepare the required documentation, architectural artefacts, and conceptual models necessary for the subsequent stages of development, deployment, and validation.

Diagrams and overview of the architectural deployment patterns adopted by each NebulOuS use case were presented in detail in Deliverable 6.1. For every pilot, it described how application and platform components are distributed across Edge (BYON) and Cloud environments, highlighting which elements are fixed, which can scale horizontally, and which must be deployed close to data sources to meet performance and latency requirements. The descriptions illustrated how NebulOuS enables flexible, hybrid deployments tailored to the operational characteristics of each use case, while maintaining a consistent control and management layer across the platform.

The list of KPIs and system requirements for all the use cases were presented in Deliverable 2.1 and Deliverable 6.1. In this document, we focus on providing a comprehensive summary of the outcomes achieved by each use case. The results of the validation process are presented, reflecting the extent to which the previously defined requirements and Key Performance Indicators (KPIs) have been fulfilled. The purpose of this summary is to highlight the effectiveness of the implemented solutions, assess their alignment with the original objectives, and demonstrate the overall performance and maturity of the system as observed through the lens of each individual use case.

UC 1.1 WINDMILL MAINTENANCE

Use-case Evaluation

The project achieved the successful validation of a semi-autonomous windmill maintenance pipeline, which was fully ported to and integrated with the NebulOuS platform. The application Kubevela specification outlined this to standardise deployment across distributed environments.

Deep integration with NebulOuS monitoring subsystems provided enhanced observability, enabling detailed analysis of internal metrics such as throughput, pipeline capacity, and performance bottlenecks. To meet the stringent operational requirements of the use case, the application leveraged the native autoscaling and optimisation capabilities provided by the NebulOuS platform. Through the combined use of monitoring, forecasting, and optimisation mechanisms, NebulOuS dynamically adjusted resource allocation to ensure that batches of photos were processed within the five-minute Service Level Objective (SLO). This outcome confirms the readiness of the integrated solution for real-world business deployments, without relying on a dedicated, use-case-specific autoscaling component.

The primary challenge was managing a complex, distributed data processing pipeline with components spanning the entire edge-cloud computing continuum. The NebulOuS platform addressed this challenge by abstracting away infrastructure-level complexity and eliminating the need for application-specific optimisation logic. In this use case, no bespoke optimisation or scheduling mechanisms were implemented within the application itself; instead, the application was described declaratively by defining its processing components, exposed metrics, and performance objectives in the form of Service Level Objectives and a utility function.

Based on this description, NebulOuS autonomously handled resource selection, deployment, scaling, and reconfiguration across heterogeneous environments. This significantly reduced the engineering effort required, avoiding days or weeks of manual infrastructure setup, testing, and performance tuning that would otherwise be necessary in a traditional DevOps-driven approach. Importantly, the optimisation process was fully managed by the platform and did not require in-depth DevOps expertise, allowing application developers to focus on domain logic rather than infrastructure concerns.

A key benefit of this approach was the seamless use of heterogeneous resources. NebulOuS transparently enabled dynamic transitions between edge and cloud resources, as well as between CPU- and GPU-enabled nodes, without requiring any explicit handling or conditional logic in the application code. This capability proved critical for maintaining high performance and resource efficiency under varying workload and infrastructure conditions, while preserving a simple and accessible development model.

This use case provided a conclusive validation of NebulOuS innovative technologies in a real-business setting. The implementation successfully transformed an inefficient, labour–intensive inspection workflow into a highly automated process, which significantly reduces the requirement for manual expert analysis. The principal lessons from this implementation highlight the strategic value of the NebulOuS platform abstraction in managing architectural complexity. By decoupling application logic from infrastructure-specific concerns, the platform enabled the windmill maintenance pipeline to be easily deployed, monitored, and reconfigured uniformly across heterogeneous edge and cloud resources. At the same time, the use case benefited from integrating metric feeds originating from IoT devices (UAVs), allowing the platform to react to device-level signals related to data generation rates, connectivity status, and operational conditions. This tight coupling between IoT-derived metrics and the NebulOuS monitoring stack proved essential for effective performance tuning, as it provided continuous, fine-grained visibility into pipeline throughput, resource utilisation, and SLO compliance. As a result, the platform's optimisation and scaling mechanisms were able to take timely, evidence-based adaptation decisions, ensuring stable performance under varying workload and connectivity conditions. Ultimately, this use case confirmed that a hybrid edge-cloud architecture, managed by an intelligent orchestration platform like NebulOuS, offers a powerful and efficient model for demanding, real-world industrial applications.

The validation of this use case combined execution on real infrastructure components with controlled simulation of adverse operational conditions. While the processing pipeline and its interaction with UAVs and edge/cloud resources were deployed on actual environments, scenarios involving limited infrastructure availability - such as constrained compute capacity, fluctuating connectivity, and bursty data ingestion - were deliberately exercised through controlled test conditions rather than uncontrolled real-world disruptions. This approach was necessary, as intentionally inducing resource starvation, network degradation, or hardware unavailability in a live operational setting would be impractical and non-reproducible.

To ensure that the observed behaviour remained representative of real-world conditions, the simulations were grounded in realistic workload patterns and device-level signals derived from UAV operations, including variations in image generation rates, intermittent connectivity, and constrained edge resources. Resource limits, node availability, and network characteristics were explicitly configured to reflect conditions commonly encountered in field deployments. This allowed the platform's monitoring, optimisation, and reconfiguration mechanisms to be evaluated under repeatable and well defined stress scenarios, providing reliable evidence of system behaviour while preserving experimental control and reproducibility.

Final functional and non-functional requirements assessment

Table 4. Final functional and non-functional requirements assessment for TTA use case

Requirements	Test Case	Status	Summary and Remarks
F_01- The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows	*UC1.1_F_01_TC1*	Passed	Under conditions of intermittent or absent network connectivity, including complete air-gap scenarios, the edge side of the system reliably buffers locally pre-processed image data (application layer functionality), subsequently transmitting it to the cloud-based processing pipeline upon network restoration. To ensure adherence to processing constraints, the NebulOuS controller dynamically scales computational components in response to high-volume image processing demands. Furthermore, should an edge-preferred, yet cloud migratable, component become unavailable due to network degradation or exceed its operational capacity, the platform autonomously initiates the deployment of its cloud-based equivalent, deallocates the compromised edge resource, and seamlessly reallocates the workload.
	UC1.1_F_01_TC2	Passed	
	UC1.1_F_01_TC3	Passed	
F_03- The system shall provide a graphic user interface for the administration of the deployed components	*UC1.1_F_03_TC1*	Passed	The GUI facilitates comprehensive application registration, enabling the provision of all requisite inputs, including the OAM file, metric model, SLO criteria, and optimisation function. Furthermore, the GUI provided clear and timely feedback concerning application deployment status. Post deployment, the

			interface successfully presented real-time operational details for each application component, alongside their associated SLOs, thereby validating its administrative and monitoring capabilities.
F_04- In addition to the Processing Capacity the system shall take into consideration the following factors when deciding on the deployment location of software components/containers: Bandwidth, Storage Capacity, Power Availability, Physical Location of edge device	*UC1.1_F_04_TC1*	Passed	The system effectively managed initial deployments by selecting appropriate User Edge devices according to varied component requirements and exhibited robust adaptability by responding intelligently to simulated changes in these environmental conditions, thereby ensuring sustained high performance and accessibility of deployed components.
F_06- The NebulOuS platform provides infrastructure monitoring capabilities	*UC1.1_F_06_TC1*	Passed	The system successfully detected and reflected various simulated operational states, including normal, stressed, and failure conditions, for infrastructure components. Crucially, the monitoring system accurately triggered necessary actions in accordance with predefined SLOs, and the platform demonstrated responsive and accurate redeployment initiation as required, thereby validating its reliability and efficacy in maintaining operational integrity.
F_07- The NebulOuS platform provides application components lifecycle management for all components of the application regardless of where	*UC1.1_F_07_TC1*	Passed	The platform successfully facilitated component deployment across the full spectrum of operational environments, including edge only, cloud only, and mixed configurations with specified edge or cloud preferences. Furthermore, the system demonstrated robust support for defining resource constraints for both cloud and edge deployments. Notably, for edge deployments, a more granular level

a component is deployed (cloud, edge, IoT device)			of constraint specification was available, encompassing geographical location, power limitations, storage capacity, and network characteristics.
	UC1.1_F_07_TC2	Passed	The platform successfully ceased the operation and accessibility of the target application and all its associated components within the NebulOuS environment. Crucially, all previously allocated resources were fully released, ensuring their availability for subsequent utilisation and contributing to efficient resource management.
F_08- The NebulOuS platform provides a deployment mechanism that makes it easy to deploy and manage data processing pipelines across various resource types (cloud/edge/fog)	*UC1.1_F_08_TC1*	Passed	The assessment confirmed the successful and operational deployment of data processing pipelines across chosen environments, including both Cloud and User Edge resources. The deployment mechanism effectively managed the orchestration of these pipelines, thereby validating its robust design for complex data workflows.
NF_01- Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions	*UC1.1_F_04_TC1*	Passed	NebulOuS capacity to ensure resource allocation efficiency was comprehensively monitored. This has been facilitated by GUI provided metrics, enabling the observation of resource provisioning and decommissioning speeds, alongside measured times for deployment, redeployment, and scaling operations. Comprehensive performance assessments, encompassing response times, throughput, and resource utilisation efficiency, have been conducted under varied workloads. Furthermore, stress tests were executed to challenge system limits. This rigorous process effectively identified areas of potential

			inefficiency or degradation, thereby informing the understanding of scalability limits and facilitating the validation and refinement of NebulOuS underlying resource allocation strategies. Ultimately, the evaluated pipeline proved sufficient for the specified use case requirements.
NF_11- The NebulOuS platform ensures data privacy when transferring data from one node to another	*UC1.1_NF_11_TC1*	Passed	Verification was performed on a running deployment of the application and encompassed breach-detection analysis, audit-trail inspection, and validation of encryption mechanisms applied to inter-node communication. The NebulOuS security layers were evaluated through a series of dedicated and controlled tests executed during normal application operation. Simulated external attack scenarios and firewall-efficacy checks were conducted by attempting unauthorised network connections and message exchanges against the active NebulOuS environment, verifying that data transferred between nodes and platform components could not be intercepted or modified. In parallel, audit-trail verification was carried out through systematic inspection of platform logs collected via the logging pipeline, confirming that all access attempts and message flows were properly authenticated and that no unauthorised access or data manipulation occurred. End-to-end encryption was further validated by simulating a malicious observer attempting packet inspection using tools such as tcpdump and Wireshark on selected nodes while the application was running. These tests confirmed that all captured traffic was encrypted at multiple layers and that no meaningful application data could be extracted.
	UC1.1_NF_11_TC2	Passed	
	UC1.1_NF_11_TC3	Passed	

Final status of the KPIs

Table 5. Final status of the KPIs for TTA use case.

KPI	Baseline	Target	Final Status and Remarks
KPI_UC1.1_1- Time to generate notification about one of data pipeline components failure or abnormal operation	n/a	<1 min	Notifications for data pipeline component failures are generated within **single seconds**, even with connectivity disruptions. Tests utilized Kubernetes (`kubectl`), InfluxDB CLI/UI, and raw logs.
KPI_UC1.1_2- Measures the time required to fully deploy the data processing pipeline within the NebulOuS platform	n/a	<15 min	Deployment time from start to fully running pipeline: **16 minutes, 22 seconds**.
KPI_UC1.1_3- Tracks the number of images processed during a single asset inspection	0	>300	Typical number of images processed per single asset inspection: **300–450 photos**.
KPI_UC1.1_4- Measures the response time of the quality assessment model, aiming for less than 2 seconds per image	n/a	< 2 sec	Image quality assessment response time: **1–2 seconds** (often <1 second).
KPI_UC1.1_5- Initial feedback on major damages: <5 min after inspection is completed	1 day	< 5min	Initial feedback on major damages: • Air-gap mode: within **5 minutes**, • fully connected mode: **live feedback**.
KPI_UC1.1_6- Reduction in data transferred to the Cloud	0% (no reduction)	30 % reduction	Data reduction (photos filtered on edge): approximately **40%**.
KPI_UC1.1_7- Time to restore normal	n/a	< 1min	Time to restore normal operation after connectivity is restored: **single**

operation after connectivity is restored			**seconds** (for short disruptions in online mode).

UC 1.2 COMPUTER VISION FOR CITY MAINTENANCE

Use-case Evaluation

The Computer Vision for City Maintenance use case successfully validated a distributed solution for automated detection of urban infrastructure issues, fully integrated within the NebulOuS platform. The implementation deployed a sophisticated video processing pipeline from live camera feeds. The system was designed to process video streams in near real-time while managing the inherent challenges of edge computing in urban environments.

Throughout the project, a flexible and resilient architecture was developed that leverages both edge and cloud resources. This architecture was driven by scalability necessities: the system needs to be able to adapt to variable workloads caused not only by fluctuating traffic density but also resource contention from other concurrent applications running on the same edge nodes, while also needing to efficiently manage the scaling of the deployment as more edge nodes are added for increased coverage of city maintenance systems. The NebulOuS platform orchestration capabilities enabled the deployment of computer vision models across heterogeneous hardware, while the integration with the monitoring and SLO subsystems provided visibility into system performance, particularly regarding object detection inference latency (the time elapsed between frame ingestion and generation of results) and resource utilization.

The platform's autoscaling capabilities proved essential for maintaining performance objectives. When detection inference times of the video processing increased due to higher workload or resource constraints, NebulOuS automatically provisioned additional resources or migrated processing tasks between edge and cloud environments. This dynamic resource management ensured that the system maintained consistent performance despite varying operational conditions.

The use case demonstrated significant practical value by automating infrastructure monitoring. As transmitting continuous raw footage—whether at 4K or even 1080p resolution—creates a heavy network load that easily saturates standard uplinks, the focus on integration of edge video processing reduced bandwidth requirements by performing initial video analysis locally and transmitting only relevant data. NebulOus allows the management of resources by only requiring heavier data transmission when cloud computing was required to ensure performance.

Final functional and non-functional requirements assessment

Table 6. Final functional and non-functional requirements assessment for Ubiwhere use case.

Requirements	Test Case	Status	Summary and Remarks
F_01- The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows	*UC1.2_F_01_TC1*	Passed	The platform successfully orchestrated the video processing pipeline across edge and cloud resources. When network connectivity degraded or edge resources became constrained, NebulOuS automatically adapted the deployment topology, migrating video processing components to cloud instances while maintaining operational continuity.
NF_01- Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions	*UC1.2_F_01_TC1*	Passed	NebulOuS demonstrated responsive decision-making throughout testing. When SLO violations were detected due to increased detection latency, the platform identified appropriate remediation strategies within 1-2 minutes and initiated redeployment actions.
	UC1.2_NF_10_TC1	Passed	Upon detecting the workload surge, the NebulOuS optimizer calculated the necessary scaling actions and triggered the provisioning of cloud instances within acceptable time limits, minimizing the impact on service availability.
NF_10- The NebulOuS platform allows effective scaling of the Cloud resources (in the Edge or using Cloud Computing)	*UC1.2_NF_10_TC1*	Passed	The platform effectively scaled resources across the edge-cloud continuum in response to workload variations. When video processing workload increased beyond edge capacity, NebulOuS automatically provisioned cloud instances and migrated appropriate components. Conversely, during low-demand periods, the system scaled down cloud resources, returning processing to edge nodes. This elastic behaviour optimized both performance and resource costs.
NF_11- The NebulOuS platform	*UC1.2_NF_11_TC1*	Passed	Network traffic analysis using packet inspection tools (Wireshark[22])

[22] https://www.wireshark.org/

ensures data privacy when transferring data from one node to another			confirmed that data remained encrypted and could not be intercepted or decoded by unauthorized parties. Access to the messaging infrastructure required proper authentication, preventing external systems from accessing video data or detection results.

Final status of the KPIs

Table 7. Final status of the KPIs for Ubiwhere use case.

KPI	Baseline	Target	Final Status and Remarks
KPI_UC1.2_1 - Vertical scaling: reducing mean time to detect an object in a video frame by 50%	2 min	1 min	ACHIEVED: Mean detection time can be kept within the parameters defined as target through optimized model deployment and resource allocation. When the system detected increased latency, vertical scaling by allocating additional resources to detection components successfully maintained detection times within target thresholds.
KPI_UC1.2_2 - Horizontal scaling: decreasing the mean time to recover after a node corruption by 50%	n/a	1 min	ACHIEVED: NebulOuS monitoring detected node unavailability within seconds, and the platform automatically redeployed affected components on alternative resources. The combination of health monitoring and automated remediation ensured rapid service restoration.
KPI_UC1.2_3 - Reducing data transfer from the Edge to the Cloud by 70%	0	1 sec	ACHIEVED: Approximately 75% reduction in data transfer achieved through edge-based video processing. By performing initial object detection on edge devices and transmitting only metadata rather than continuous raw video streams. By focusing the video analysis on the edge and transmitting only relevant data, while using heavier data transmission only when cloud computing is required to ensure performance leads to even greater data transmission reductions.

KPI_UC1.2_4 - Reduce number of necessary Edge nodes by 10%	4	3	PARTIALLY: Through optimized resource utilization and intelligent workload distribution, the system can operate effectively with 3 edge nodes instead of the initially projected 4. NebulOuS resource management ensures efficient utilization of available edge capacity, and provisions cloud resources to ensure capacity. However, since NebulOuS does not currently support deploying multiple components on a single node, this requirement cannot be fully fulfilled.
KPI_UC1.2_5 - Reduce number of necessary Cloud resources by 50%	2	1	ACHIEVED: By maximizing edge processing capabilities and utilizing cloud resources only during peak demand or edge resource constraints, the system typically operated with a single cloud instance (master) instead of two or more. The platform's adaptive scaling ensured cloud resources were provisioned dynamically only when required, reducing overall cloud infrastructure costs.

UC 2.1 MERCABARNA INTRA-LOGISTICS

Use-case Evaluation

The use case successfully validated a distributed solution for traffic monitoring and analysis in Mercabarna, fully integrated and operational within the NebulOuS platform. During the implementation, a highly flexible and resilient video-processing pipeline was deployed, supported by the platform's orchestration, monitoring, and adaptation mechanisms. The NebulOuS application specification allowed standardized deployments across both heterogeneous edge and cloud nodes, including ARM-based environments such as Mercabarna's Geekom[23] devices and Raspberry Pi 4 units used in the laboratory. Integration with the SLO monitoring subsystem provided a detailed view of performance and the system's ability to recover from failures and performance drops.

One of the main challenges was managing a continuous data stream, including images, from multiple cameras distributed across different pavilions, while maintaining near-real-time communication between modules and robust operational continuity even when nodes failed. NebulOuS helped reduce this complexity by providing high-level abstractions to guide placement decisions based on factors such as geographical location and resource availability. Monitoring and proactive reconfiguration

[23] https://www.geekompc.com/

capabilities allowed components to be moved between edge and cloud nodes when latency increased, maintaining performance objectives and overall system efficiency.

The use case confirmed the capabilities of NebulOuS in real-world environments. By automating processes that previously relied on manual monitoring, Mercabarna obtained an effective traffic monitoring application that significantly improves incident detection, as well as a historical database that enables analysis and understanding of internal road usage patterns. The main lessons learned highlight the strategic value of intelligent orchestration in hybrid edge-cloud architectures, as well as the critical role of integrating metrics and SLOs to optimize performance in complex, high-demand scenarios.

Final functional and non-functional requirements assessment

Table 8. Final functional and non-functional requirements assessment for Mercabarna Intra-Logistics use case.

Requirements	Test Case	Status	Summary and Remarks
F_01- The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows	*UC2.1_F_01_TC1*	Passed	When a module becomes unavailable and stops sending metrics, the corresponding SLO is violated. This triggers a redeployment of the failing module on another node, ensuring continuity of the data-processing workflow.
F_02- The NebulOuS platform offers a lightweight resilient event pub/sub mechanism with persistence, access control and consumer-group capacities to orchestrate communication between the different modules of the solution	*UC2.1_F_02_TC1*	Passed	The modules successfully exchange information through the NebulOuS broker, and communication operates as expected. The solution also stores the events from all four cameras in the database correctly, confirming that the end-to-end data flow works as intended.

F_04- In addition to the Processing Capacity the system shall take into consideration the following factors when deciding on the deployment location of software components/containers: Bandwidth, Storage Capacity, Power Availability, Physical Location of edge device.	*UC2.1_F_04_TC1*	Passed	NebulOuS determines where the VehicleCropping modules are deployed by considering the configured GPS locations, placing each module on the closest available node. Multiple tests have been conducted on the Mercabarna edge nodes.
F_05- NebulOuS should constantly monitor the QoS required by the different application components (as specified by their SLA) and apply necessary actions to remediate QoS violations	*UC2.1_F_05_TC1*	Partially Passed	NebulOuS can redeploy components when a node lacks sufficient resources, detecting the issue through SLO monitoring and relocating the components as needed. However, since NebulOuS does not currently support deploying multiple components on a single node, this requirement cannot be fully fulfilled.
F_09- The NebulOuS platform provides near-real-time communication between application components	*UC2.1_F_02_TC1*	Passed	Communications between the VehicleCropping modules and the LicensePlate modules occur in near real-time, with events being stored in the database in less than five seconds.
NF_01- Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions	*UC2.1_F_01_TC1*	Passed	As mentioned above
	UC2.1_F_04_TC1	Passed	As mentioned above
	UC2.1_F_05_TC1	Partially Passed	As mentioned above
NF_03- The platform provides secure access to the resources	*UC2.1_NF_03_TC1*	Passed	Access to the web application where deployments are defined is protected. Users cannot log in without valid user credentials.
NF_04- NebulOuS platform should be able to deploy applications in a Raspberry Pi 3 Model B+ or similar ARM architecture	*UC2.1_F_01_TC1*	Passed	The NebulOuS platform successfully tested the deploy of the application in the laboratory environment using a Raspberry Pi 4, confirming support for ARM-based worker nodes. Also, despite some network instability and setbacks, Mercabarna's GEEKOM devices were successfully tested.
NF_05- Data transferred between edge nodes and	*UC2.1_NF_05_TC1*	Passed	Access to the message broker is restricted. Nodes cannot connect or

between edge nodes and clouds should be secured			exchange data without the appropriate credentials.
NF_09- The NebulOuS platform provides near-real-time processing of big data	*UC2.1_NF_09_TC1*	Passed	The platform was tested with real footage in Mercabarna. With all cameras active and the corresponding modules deployed across the Mercabarna nodes, the system achieved a processing delay of under 5 seconds, confirming near-real-time performance.
NF_10- The NebulOuS platform will allow effective scaling of the Cloud resources (in the Edge or using Cloud Computing)	*UC2.1_NF_10_TC1*	Passed	Because NebulOuS does not currently support deploying multiple components on a single node, the "Vehicle Detection and Cropping" module was manually deployed on W1. Under this setup, NebulOuS successfully deployed the "License Plate Detection and Reading" component on W1 and, when processing latency increased, automatically migrated it to a cloud node, demonstrating effective scaling across edge and cloud resources.
NF_11- The NebulOuS platform ensures data privacy when transferring data from one node to another	*UC2.1_NF_11_TC1*	Passed	The requirement was evaluated by deploying the application through NebulOuS and replacing the VehicleCropping module with one that periodically sends an image every second. Attempts to connect to the NebulOuS broker from outside the platform, even when using valid credentials, were rejected. This confirms that data exchanges between nodes remain protected and inaccessible to unauthorized external connections.

Final status of the KPIs

Table 9. Final status of the KPIs for Mercabarna Intra-Logistics use case.

KPI	Baseline	Target	Final Status and Remarks
KPI_UC2.1_1- Reducing data transfer from the Edge to the Cloud	448Mbps of video sent to the cloud (estimated)	0Mbps of video sent to the Cloud.	Completed, all video streams are processed directly by the VehicleCropping modules deployed on the Mercabarna edge devices. As a result, no raw video is transferred to the cloud, and only cropped images are sent when necessary.
KPI_UC2.1_2- Reduce the number of necessary cloud resources	n/a	Video streams processed on Edge	The requirement is fulfilled. Six NUC devices were deployed, one in each Mercabarna pavilion, providing sufficient capacity to run the entire solution at the edge and eliminating the need to process video streams using cloud resources.
KPI_UC2.1_3- Improve road usage patterns awareness	Qualitative knowledge from managers	Historical quantitative data available	Complete Mercabarna now has a historical database containing detailed quantitative data about road usage. In addition, the web application allows managers to query this data and compare usage patterns across different roads, significantly improving situational awareness.
KPI_UC2.1_4- Less than 5 minutes to identify potential traffic issues	15 min	5 min	The target has been met. The web application includes a feature that monitors traffic volume for the previous minute on each road. An alert is triggered when the configured threshold is exceeded in two consecutive time windows, [t-1 min, t-2 min] and [t-2 min, t-3 min], allowing traffic issues to be detected in four minutes.

UC 2.2 MERCABARNA- LAST MILE USE CASE

Use-case Evaluation

This use case focused on creating a distributed solution for real-time delivery route management within the NebulOuS platform. The deployed system continuously processes GPS updates from multiple vehicles, evaluates whether the current delivery route remains feasible, and triggers route recalculation whenever delays or deviations are detected. Throughout the implementation, NebulOuS supported the deployment of the solution on edge nodes and used cloud resources only when SLOs were not met, and a reconfiguration of the deployment was required. The integration with the SLO monitoring subsystem provided clear visibility into latency behavior and enabled the detection of performance degradation.

One of the main challenges involved handling the potentially high volume of GPS location updates while ensuring near-real-time reaction and the ability to recompute delivery routes quickly when needed. NebulOuS addressed this by enabling dynamic scaling and distributing workload across multiple DeliveryPlanManager instances. Thanks to this capability, messages were partitioned correctly, ensuring that each vehicle was consistently processed by the same worker and that the workload remained balanced as the number of vehicles or data streams increased.

This use case demonstrates that NebulOuS can effectively support continuous route monitoring and rapid route adjustments in operational delivery scenarios. Route recalculation is performed immediately after new location data is received, improving responsiveness and ensuring that delivery plans remain up to date. Executing routing tasks at the fog layer further reduces latency and avoids unnecessary dependence on cloud resources. Overall, UC2.2 demonstrates how adaptive behaviours and distributed processing contribute to more resilient and responsive delivery-management systems.

Final functional and non-functional requirements assessment

Table 10. Final functional and non-functional requirements assessment for Mercabarna Last-Mile use case.

Requirements	Test Case	Status	Summary and Remarks
F_01- The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows	*UC2.2_F_01_TC1*	Passed	Nebulous allow the deployment of the solution on the edge and it's only moved into cloud resources if the SLO are not meet and the solution structure is reconfigured.

F_02- The NebulOuS platform offers a lightweight resilient event pub/sub mechanism with persistence, access control and consumer-group capacities to orchestrate communication between the different modules of the solution	*UC2.2_F_01_TC1*	Passed	The modules are successfully exchanging information through the Nebulous broker, and the communication is functioning as expected. Furthermore, the verification procedures associated with F_01 and F_05 have been completed and confirmed.
F_04- In addition to the Processing Capacity the system shall take into consideration the following factors when deciding on the deployment location of software components/containers: Bandwidth, Storage Capacity, Power Availability, Physical Location of edge device		Due to the final proposed software architecture, this requirement is no longer relevant to the use case	Initially, the placement of edge devices was considered a key factor to ensure proximity to data sources. However, because the ultimate origin of the data is the cloud and cannot be geographically constrained, the physical location of computing nodes cannot be taken into consideration when selecting the most suitable node candidates.
F_05- NebulOuS should constantly monitor the QoS required by the different application components (as specified by their SLA) and apply necessary actions to remediate QoS violations	*UC2.2_F_01_TC1*	Passed	Map server exposes request processing latency values utilising a Prometheus endpoint. NebulOuS polls this API and, when the value of the latency goes above 5s, triggers an appropriate reconfiguration action to remediate the QoS violation.
F_09- The NebulOuS platform provides near-real-time communication between application components	*UC2.2_F_09_TC1*	Passed	A testing module was implemented that generates a simulated order and publishes messages with the delivery agent's location every 5 seconds. The messages were received by the DeliveryPlanManagerWorker module in less than 100 ms, as stipulated in the Means of Verification. Specifically, the observed latencies ranged between 10 and 20 ms, confirming near-real-time communication.

F_10- NebulOuS offers a mechanism to partition streaming analysis jobs between a variable number of workers	*UC2.2_F_10_TC1*	Passed	Utilizing the NebulOuS message broker, we defined an IoT pipeline that distributes messages among multiple instances of the DeliveryPlanManager host based on the topic in which the vehicle position is published (which contains the vehicle ID). The mechanism ensures that all messages belonging to a specific vehicle are processed by the same worker until a workload re-balancing occurs. In addition, by deploying 10 instances of the testing module simulating vehicles and 2 instances of the DeliveryPlanManager module, we observed that messages were correctly distributed between the two manager instances according to the partitioning mechanism.
F_11- NebulOuS should offer a mechanism to deploy and invoke serverless computation functions	*UC2.2_F_11_TC1*	Passed	NebulOuS is capable of allocating resources and executing serverless functions, such as the Route Optimization module, deployed using NebulOuS' serverless offering (Knative). Furthermore, metrics on the number of parallel requests can be collected, and an SLO along with a utility function can be defined. When the ratio of parallel requests to available CPUs exceeds a specified threshold, additional CPUs are automatically provisioned, ensuring that request processing times remain below 5 seconds on average.
NF_01- Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions	*UC2.2_F_01_TC1*	Passed	Under low NebulOuS workload, in test F_05_TC1, NebulOuS takes less than one minute from the moment an SLO violation is detected until the deployment re-adaptation is

			initiated and the new deployment topology is determined. The deployment process itself takes approximately 6 minutes when using devices with good connection speed and up-to-date base images (after apt update).
NF_03- The platform provides secure access to the resources (clouds, edge resources etc.)	*UC2.2_NF_03_TC1*	Passed	Access to the web application where deployments are defined is protected. Users cannot log in without valid user credentials.
NF_05- Data transferred between edge nodes and between edge nodes and clouds should be secured	*UC2.2_NF_05_TC1*	Passed	Access to the message broker is restricted. Nodes cannot connect or exchange data without the appropriate credentials.
NF_06- The system provides elasticity capabilities along with the cloud continuum, so HW resources are provisioned or freed depending on the workload	*UC2.2_F_05_TC1*	Passed	As demonstrated in test F_05_TC1, the Map Server component scales up/out when the workload increases. Once the test concludes and the request generator stops, NebulOuS automatically scales down/in the Map Server component, confirming the system's elasticity capabilities.
NF_09- The NebulOuS platform provides near-real-time processing of big data	*UC2.2_NF_09_TC1*	Passed	A testing module was developed to simulate 5,000 real vehicles. This module sends the orders for each vehicle and periodically transmits the vehicle information, just as a real vehicle would do. We ran the tests using the parameters defined in the test case: 5,000 vehicles sending messages every 5 seconds. The Nebulous Pub/Sub broker is able to handle this data volume, and the messages arrive in near real time, with delays of no more than 3 second.
NF_10- The NebulOuS platform will allow effective scaling of the Cloud resources (in the Edge or using Cloud Computing)	*UC2.2_F_01_TC1*	Passed	Successfully validated with test F_01, NebulOuS is capable of identifying and allocating new cloud hosts (e.g., Azure, Amazon/AWS, etc.) for deployed modules when the original

			resources are unable to handle the current workload.
NF_11- The NebulOuS platform ensures data privacy when transferring data from one node to another	*UC2.2_NF_11_TC1*	Passed	We captured the communication traffic using Wireshark between the dummy module used in test UC2.2_NF_09_TC1, which sends GPS positions every 5 seconds, and the DeliveryPlanManager component. During the capture, no readable information was observed; all intercepted packets contained encrypted payloads. This confirms that communication is transmitted using an appropriate encryption mechanism. The NebulOuS platform therefore ensures data confidentiality and privacy when transferring data between nodes.

Final status of the KPIs

Table 11. Final status of the KPIs for Mercabarna Last-Mile use case.

KPI	Baseline	Target	Final Status and Remarks
KPI_UC2.2_1 - Detect delays in delivery route and trigger a recalculation in less than 2 minutes	15 min	2 min	Each time a new GPS position is received, the DeliveryManager module checks whether the current delivery route is still feasible. If a delay or deviation is detected, it immediately triggers the route-reconfiguration module, which generates a new updated route.
KPI_UC2.2_2 - Reduce average delivery route duration by 10%	3 h	2 h 40 min	The reduction target in average delivery route duration was achieved according to the calculated routes. It is important to note, however, that this evaluation is theoretical: the estimated times are based on route calculations and have not been tested under real driving conditions. Actual delivery times may differ in practice, as drivers may not always follow the suggested route precisely.
KPI_UC2.2_3- Reduction of data	n/a	<5%	Based on the initial assumption that the data sources were located at servers in the Barcelona

transfer from the Edge to the Cloud			area, it was intended to do a pre-filtering of geolocation data at edge to remove any incorrect traces, thus, reducing data transfer to cloud. However, due to changes in the software architecture of the data provider, the ultimate origin of the data is the cloud. This makes the proposed KPI non achievable.
KPI_UC2.2_4- Increase the number of tasks executed in fog	None	Vehicle routing should happen at fog	Each vehicle routing task is executed at the fog layer, enabling low-latency decision-making and real-time responsiveness. This behavior is ensured by meeting the requirements F_10, F_11, and NF_10.

UC 3 PRECISION AGRICULTURE

Use-case Evaluation

Augmenta's Field Analyzer operates on tractors and agricultural machinery, capturing multispectral video and sensor data related to plant health and field conditions. As soon as the operation ends, the Field Analyzer sends all required data to the NebulOuS middleware through nearby 5G nodes provided by Ubitech. These datasets include high-resolution video logs, sensor readings, and CSV files containing application details.

Augmenta takes advantage of the NebulOuS outcomes by integrating the platform's orchestration, resource management, and edge–cloud deployment capabilities into our existing processing workflows. By using NebulOuS, we can distribute work intelligently between the edge and the cloud, preferring the edge whenever a suitable device is available and falling back to cloud resources when needed, so we can reduce delays during busy periods and better handle peaks in demand. This will improve our ability to process larger and more complex field operations and can directly enhance the services we offer to farmers and agronomists, such as quicker access to field results, more accurate insights, and smoother operation handling.

During the NebulOuS project we managed to integrate our Augmenta map-generation application with the platform's workflow engine: we containerised our map server, deployed it on the NebulOuS cluster, and used a small "workflow-runner" service which is part of the Nebulous Workflow Executor to submit batches of real Augmenta session IDs to Argo Workflows, which then scheduled and executed our map-generation containers and pushed the resulting maps, CSVs and shapefiles back to our cloud.

In practice this allowed us to replay many real operations, but we faced challenges such as preparing images for the correct architectures (ARM vs amd64), dealing with memory limits and *OOMKilled* pods when processing very large sessions (big fields in hectares or many sessions in parallel), and handling pending workflows when resources were exhausted. Overall, we have shown that NebulOuS can orchestrate and scale our map-generation workflows in realistic conditions and offers advantages in centralized orchestration and elasticity, but it also introduces additional complexity in debugging, resource management, and aligning the platform with real business constraints and very large agricultural workloads.

Despite initial hardware and stability challenges with physical Jetson devices, we successfully achieved edge deployment by onboarding our nodes and validating the system within virtualized environments. This confirmed that NebulOuS can fully integrate with real precision-agriculture applications. While the process highlighted the practical difficulties of cloud-to-edge orchestration, we have now established a stable setup that supports both cloud and edge-targeted workloads for final KPI validation.

Final functional and non-functional requirements assessment

Table 12. Final functional and non-functional requirements assessment for Augmenta use case.

Requirements	Test Case	Status	Summary and Remarks
F_01- The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows	*UC3_F_01_TC1*	Passed	NebulOuS successfully orchestrated our data-preparation and map-generation pipelines using Argo Workflows and the workflow-runner, allowing us to run flexible, containerised processing workflows on real Augmenta sessions.
	UC3_F_01_TC2	Passed	NebulOuS successfully orchestrated our data-preparation and map-generation pipelines using Argo Workflows and the workflow-runner, allowing us to run flexible, containerised processing workflows on real Augmenta sessions.
F_03- The system shall provide a graphic user interface for the administration of the deployed components	*UC3_F_03_TC1*	Passed	We used the NebulOuS / KubeVela-based graphical interface to deploy, configure and monitor our application components (runner, workers, workflow definitions).
F_04- In addition to the Processing Capacity the system shall take into consideration the following factors when deciding on the deployment location of		Due to the proposed software architecture, this functionality is no longer	The NebulOuS optimizer successfully incorporates these deployment factors. The Cloud Fog Service Broker (CFSB) uses attributes such as bandwidth, storage capacity, power availability, and physical location to filter

software components/containers: Bandwidth, Storage Capacity, Power Availability, Physical Location of edge device		required by the use case	potential node candidates. These criteria are then included in the Utility Function (UF), allowing the optimizer to select the best deployment location based on the complete set of operational requirements.
F_05- NebulOuS should constantly monitor the QoS required by the different application components (as specified by their SLA) and apply necessary actions to remediate QoS violations	*UC3_F_05_TC1*	Passed	NebulOuS (via EMS) continuously monitors our workflows and collects QoS-related metrics such as CPU usage and processing-time metrics for each workflow.
F_12- Monitor and predict downtime of registered resources to maximize time/cost ratio	*UC3_F_12_TC1*	Passed	NebulOuS does not predict the specific downtime of edge or cloud resources; rather, it is designed to react to failures when they occur. To optimize performance, the system selects nodes based on specific preference criteria configured in the Cloud Fog Service Broker. One of these key criteria is the probability of a node going offline, allowing the platform to proactively choose the most stable candidates for task deployment.
F_13- Handle paused or dropped tasks	*UC3_F_13_TC1*	Partially passed	Dropped tasks are handled by Argo's retry strategy, which automatically restarts them if they fail. However, we don't currently have a way to handle paused tasks.
NF_01- Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions	*UC3_F_05_TC1*	Passed	NebulOuS scheduled our workflows quickly, CPU usage scaled up as we submitted many sessions, and we were able to process large batches of real operations without noticeable delays, except when we intentionally pushed the nodes to their resource limits.
NF_02- The NebulOuS platform will be able to	*UC3_NF_02_TC1*	Passed	The platform prioritizes the edge node so that all work happens

operate even if available bandwidth between edge infrastructure and the cloud is limited or connection is temporarily lost			locally. This allows the system to finish its tasks even if the internet is very slow or the connection is lost.
NF_03- The platform provides secure access to the resources (clouds, edge resources etc.)	*UC3_NF_03_TC1*	Passed	NebulOuS provided secure access to cloud and edge resources through VPNs, credentials, and controlled endpoints, which worked smoothly and safely for us. The web application for managing deployments is also protected, requiring valid user credentials for any login. However, as a use case partner, we haven't performed a formal penetration test, so our validation of NF_03 is based on this daily operational experience rather than a dedicated security assessment.
NF_10- The NebulOuS platform will allow effective scaling of the Cloud resources (in the Edge or using Cloud Computing)	*UC3_NF_10_TC1*	Passed	When we sent many sessions, NebulOuS scaled our workloads across the worker nodes, ran many workflows in parallel, and we saw CPU usage and throughput increase until we hit the physical limits of the nodes.
NF_11- The NebulOuS platform ensures data privacy when transferring data from one node to another	*UC3_NF_11_TC1*	Passed	The platform uses WireGuard to create a secure "private tunnel" between nodes, making sure all information sent is automatically locked and encrypted. This keeps your data private and protected from unauthorized access as it moves across the network, fulfilling the NF_11 requirement.

Final status of the KPIs

Table 13. Final status of the KPIs for Augmenta use case.

KPI	Baseline	Target	Final Status and Remarks
KPI_UC3_1 - Processing time reduction	15 %	30 %	Lab tests showed execution times identical to a standard deployment without NebulOuS. This result was due to the limited resources of the lab machines, which restricted the system's ability to improve processing speed.
KPI_UC3_2 - Cloud cost reduction	5%	10%	By deploying our pipelines to the edge, we see a significant reduction in cloud compute costs as the processing is done on local hardware where the compute cost is zero.
KPI_UC3_3 - Increase capabilities of processing tasks	5%	10%	With the NebulOuS workflow setup, multiple map-generation tasks can run in parallel on the same worker node as long as there is enough capacity. If an SLO violation is detected (e.g., tasks taking too long), an additional worker is deployed, further increasing the overall processing capability

UC 4 CRISIS MANAGEMENT

Use-case Evaluation

Throughout the project, the primary objective was to validate and demonstrate the NebulOuS platform in real-business, high-demand operational environments. To this end, the Use case application was deployed and assessed across multiple large-scale training activities and full-scale disaster-response exercises, enabling the evaluation of the IoT–edge–cloud continuum under realistic field conditions.

The first field test took place during the @fire Simulation USAR exercise in Vilejust (Paris), where approximately 60 participants were involved. This event provided the initial opportunity to deploy early versions of our application and test the available NebulOuS components in an operational urban-search-and-rescue setting.

A significant milestone was the participation in FullEx "Achilles" (May 2024) in Epeisses/Geneva, involving around 350 participants. This crisis-management exercise simulated a complex, multi-agency response in a dense urban environment operating under a layered incident-command structure. The event offered a controlled but highly dynamic context in which we could observe how

NebulOuS supports situational-awareness processes and adaptive coordination across emergency services.

The platform was further validated during FullEx “Magnitude” (October 2024), an EU Civil Protection Mechanism full-scale earthquake-response simulation in southwest Germany. The exercise involved approximately 950 specialists, including international USAR (Urban Search and Rescue) teams, CBRNE (Chemical, Biological, Radioactive, Nuclear, Explosive) units, local authorities, and regional fire departments. These operational conditions underscored the critical importance of robust synchronisation mechanisms linking IoT sensors, edge devices, and cloud-based services—particularly in a multinational, resource-intensive environment.

Additional insights were gained during the large-scale evacuation in Osnabrück (November 2024), where roughly 14,000 residents were evacuated and over 500 responders were deployed during the disposal of seven WWII bombs. In this context, NebulOuS enabled the visualisation of tracking devices installed in transport vehicles through the use-case dashboard service. The exercise exposed information-flow bottlenecks between coordination centres and field units, highlighting the relevance of integrated, real-time data pipelines for evacuation management.

The evaluation continued during the NATO-led FullEx “Bulgaria” (September 2025), co-organised by the EADRCC (Euro-Atlantic Disaster Response Coordination Centre) and the Bulgarian Ministry of Interior, with more than 1,000 participants. This civil–military exercise enabled an assessment of secure and resilient communication infrastructures in a complex multinational operational environment. Challenges encountered—particularly those arising from differing national procedures and command structures—reinforced the need for common standards and harmonised digital tools to support joint decision-making.

Finally, RETTmobil 2025 provided an important dissemination and demonstration setting. As one of Europe’s leading rescue and mobility fairs, it enabled the presentation of our integrated communication and data-processing architecture, as well as the broader achievements of the project. The event emphasised the operational necessity of resilient cloud-continuum solutions for disaster-response organisations.

Final functional and non-functional requirements assessment

Table 14. Final functional and non-functional requirements assessment for FIRE use case.

Requirements	Test Case	Status	Summary and Remarks
F_01- The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing	*UC4_F_01_TC1*	Passed	NebulOuS deploys the application and automatically starts a redeployment process when individual nodes are removed from the cluster

flexible data processing workflows			
F_02- The NebulOuS platform offers a lightweight resilient event pub/sub mechanism with persistence, access control and consumer-group capacities to orchestrate communication between the different modules of the solution	*UC4_F_02_TC1*	Passed	NebulOuS allows the transfer of IoT data between application components through a Pub/Sub mechanism.
F_03- The system shall provide a graphic user interface for the administration of the deployed components	*UC4_F_03_TC1*	Partially Passed	Change in optimization criteria is realised by either redeployment of application, or publication of application specific metric.
F_04- In addition to the Processing Capacity the system shall take into consideration the following factors when deciding on the deployment location of software components/containers: Bandwidth, Storage Capacity, Power Availability, Physical Location of edge device	*UC4_F_04_TC1*	Not possible to execute the TC	Application component can only be attached with geographic proximity to worker nodes. A selection of one specific node in a location with multiple nodes is not possible.
	UC4_F_04_TC2	Partially Passed	Non-native system metrics like “power availability” have to be virtualised as application metric.
F_14- The system shall be able to be started and operated during run-time with minimal prior training	*UC4_F_14_TC1*	Passed	The deployment of a preconfigured application is feasible with minimal training.
NF_01- Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions	*UC4_NF_01_TC1*	Passed	NebulOuS scales application components based on application specific metrics and the corresponding utility function.

NF_02- The NebulOuS platform will be able to operate even if available bandwidth between edge infrastructure and the cloud is limited or connection is temporarily lost	*UC4_NF_02_TC1*	Passed	NebulOuS redeploys failed application components to the available resources.
NF_03- The platform provides secure access to the resources (clouds, edge resources etc.)	*UC4_NF_03_TC1*	Passed	Standard Access Controls are implemented and working.
NF_07- The system shall be able to cope with a sudden loss of connection to individual computational units	*UC4_NF_07_TC1*	Passed	NebulOuS redeploys failed application components after loss of connectivity to individual nodes to the available resources.
NF_08- The system shall be able to adapt the maximum load of the edge devices to the available power supply	*UC4_F_04_TC2*	Partially Passed	see above under F_04
NF_10- The NebulOuS platform will allow effective scaling of the Cloud resources (in the Edge or using Cloud Computing)	*UC4_NF_10_TC1*	Passed	NebulOuS can be configured to deploy applications with best effort, also allowing to partially deploy applications. Once more suitable nodes are available the remaining application components are deployed.

Final status of the KPIs

Table 15. Final status of the KPIs for FIRE use case.

KPI	Baseline	Target	Final Status and Remarks
KPI_UC4_1 - Increase of total information processed digitally in disaster management	2 %	10 %	In the experimentations different types and numbers of field devices (iot) were used. The calculated target of 200 data points per day was far exceded in all deployments. *ACHILLES: 18033 / 4 days / 14 devices* *MAGNITUDE: 50157 / 4 days / 15 devices*

			EVACUATION: 6752 / 1 day / 21 devices RETTmobil: 2106 / 3 days / 6 devices BULGARIA: 277 / 2 days / 8 devices Average data points per day*: ~5523 (2716% of target)*
KPI_UC4_2 - Increase Local data communication coverage during the early stages of disaster situations	5 %	60 %	In the experimentations a mission area was defined as map polgon. During the exercise the coverage of the area was assessed through the distributed sensors and iot connected devices as well as dedicated range test devices. The coverage is highly dependent on size and topology of the mission area. ACHILLES coverage ~44% (120147/270894m^2) MAGNITUDE coverage ~68% (144524/211998m^2) RETTmobil coverage ~71% (98501/138951m^2) Average coverage: ~61% (102% of target)
KPI_UC4_3 - The Acquisition of individual situational information (e.g., weather info), shall be automated and personnel effort shall be eliminated which leads to the reduction of time to acquire data points	15-60 min	1 min	In the ACHILLES and MAGNITUDE missions location tracking as well as environmental information was collected and processed. The data was collected in varying intervals from 10s to several minutes. The target of a 1 min acquisition time could be reached with all devices.
KPI_UC4_4- Increase the capacity of Average number of individual situational information points available (e.g., water level)	30	1000	The capacity of the system was both experimented in the field missions (see comment of KPI_UC4_1 for numbers) as well as in simulated load tests with up to 10.000 parallel sensors.
KPI_UC4_5- Complementary information automatically derived in disaster management	0	100	During the ACHILLES mission real-time thermal drone imagery generated by a fellow research team of DLR was integrated into the use case application and could be directly viewed. Likewise in the ACHILLES and MAGNITUDE mission the ASIGN data source was integrated

			into the use case application. The tracking, text and images from the ASIGN service were processed and made available through the use case application.

6.1. VALIDATION OF THE NEBULOUS PLATFORM BY USE CASES-SUMMARY

Use case KPIs make it possible to evaluate the use cases themselves, and by extension, assess the NebulOuS platform from the viewpoint of those use cases, rather than only from the platform developers' perspective, as is done with the objectives KPIs defined in the DoA. Because of this difference, there is no direct one-to-one relationship between objectives KPIs and use case KPIs, although some of them naturally align. Both sets ultimately aim to confirm that the developed software is valuable for all stakeholders.

The tables below illustrate this subtle relationship between both KPI types. Table 16 links relevant objectives KPIs to use case KPIs based on groups of related functionalities. The second Table 17 restates the objectives KPI descriptions and assigns them to these groups where appropriate (or provides commentary when no clear assignment exists).

Table 16. Grouping Use Cases KPIs and Objective KPIs in terms of functionality.

Group	Description	Related Objective KPIs	Related Use Cases KPIs
K1	Monitoring (including QoS, SLA, eventing, anomaly detection)	KPI 2.1 KPI 2.2 KPI 4.1 KPI 4.2 KPI 4.3 KPI 5.1 KPI 5.2 KPI 5.3	KPI_UC1.1_1, KPI_UC1.1_3, KPI_UC1.1_4, KPI_UC1.1_5, KPI_UC1.2_1, KPI_UC2.1_4, KPI_UC2.2_1, KPI_UC3_1, KPI_UC3_2, KPI_UC3_3, KPI_UC4_1, KPI_UC4_4
K2	Secure networking across environments	KPI 2.3 KPI 2.4	KPI_UC1.1_6, KPI_UC1.2_3, KPI_UC2.1_1, KPI_UC2.2_3, KPI_UC4_2
K3	Deployment lifecycle management	KPI 1.3 KPI 3.1 KPI 3.2 KPI 3.3	KPI_UC1.1_2, KPI_UC1.1_6, KPI_UC1.1_7, KPI_UC1.2_2, KPI_UC1.2_3, KPI_UC2.1_1, KPI_UC2.2_3, KPI_UC4_2
K4	Deployment reconfiguration (adaptation and optimisation)	KPI 1.2 KPI 1.4 KPI 3.4	KPI_UC1.1_3, KPI_UC1.1_4, KPI_UC1.1_5, KPI_UC1.2_1, KPI_UC1.2_2, KPI_UC2.2_1

Table 17. Mapping KPIs groups to the Objectives KPIs.

Objectives KPIs	Description	Related Use Cases KPIs Group(s)
KPI 1.1	design of the NebulOuS Platform Architecture capturing the complete NebulOuS ecosystem	N/A - this is not validated directly at the use cases level. Instead, this is a prerequisite to develop the NebulOuS platform and enable any other validations.
KPI 1.2	design and implementation of the MCDM-based cloud and fog brokerage service	K4
KPI 1.3	design and implementation of all the appropriate mechanisms for supporting cross-cloud and Fog applications deployment	K3
KPI 1.4	implementation of all the appropriate mechanisms for autonomous reconfigurations in ad-hoc cloud computing continuums	K4
KPI 2.1	design of a tool for automatically deriving SLAs based on an awareness of the application provisioning capabilities and requirements	K1
KPI 2.2	design of a dedicated tool for translating SLAs to Smart Contracts	K1
KPI 2.3	design of the mechanism that automatically installs secure network overlays over ad-hoc cross-clouds and fog resource	K3
KPI 2.4	design of the NebulOuS security and privacy-by design in data streams propagation	K3
KPI 3.1	design of mechanisms that undertake automatic deployment and orchestration of processing jobs and function-as-a-service applications in ad-hoc cloud computing continuums	K3
KPI 3.2	design of mechanisms that undertake and real-time scheduling of functions and workflow tasks	K3
KPI 3.3	design of lifecycle management support mechanisms for efficient management of interoperable IOT/Fog data streams	K3
KPI 3.4	design of self-adaptive and proactive mechanisms for reconfiguration enactment and workflow adaptation	K3
KPI 4.1	semantic models design for managing applications and brokerage services in fog computing	K1
KPI 4.2	design SLAs driven from the semantic models	K2
KPI 4.3	design and implement a model-driven quality assurance mechanism that will guide and control the platform's brokerage capabilities	K3

KPI 5.1	design of the NebulOuS distributed event management system	K1
KPI 5.2	design of the NebulOuS AI-driven anomaly detector	K1
KPI 5.3	design of the appropriate interfaces to feed with distributed monitoring data the autonomous reconfigurations mechanism	K1
KPI 6.1	design and specification of four pilots	N/A - this is validated by the contents of deliverable D6.1
KPI 6.2	use cases demonstration through the deployment of the platform	N/A - this is validated by the contents of deliverable D6.1
KPI 6.3	validation of NebulOuS innovative technologies in real-business settings	N/A - this is validated by the contents of deliverable D6.1 and in this one D6.2

The Table 18 outlined the final status of the functional and non-functional requirements defined in Deliverable 2.1 and presented in Deliverable 6.1. The evaluation results demonstrate that the NebulOuS platform successfully satisfies the majority of functional and non-functional requirements across the assessed use cases, confirming its maturity as a cloud–edge orchestration and data-driven platform.

From a **functional perspective**, core platform capabilities—such as orchestration of distributed data processing workflows, deployment and lifecycle management of application components across cloud and edge environments, infrastructure and application monitoring, near-real-time communication, and scalable deployment of data pipelines—were consistently validated. These features were successfully exercised across multiple use cases, indicating strong coverage of the platform's primary design objectives. Advanced mechanisms, including serverless function execution, workload partitioning, and QoS-aware deployment decisions, were also validated in selected use cases, reflecting the platform's flexibility in supporting diverse application paradigms.

From a **non-functional standpoint**, the results show strong compliance with performance, scalability, resilience, and security requirements. The platform demonstrated the ability to make fast resource allocation decisions, operate under limited or disrupted connectivity, securely transfer data across nodes, and scale resources dynamically along the cloud–edge continuum. Support for deployment on resource-constrained edge devices, including ARM-based platforms, was also successfully validated.

A small number of requirements were partially validated, most notably the handling of paused or dropped tasks and power-aware adaptation of edge workloads. These outcomes reflect the complexity of such scenarios and the fact that they were exercised under constrained or use-case-specific conditions, rather than indicating fundamental platform limitations.

In two instances, specific requirements were deemed obsolete due to the final software architecture.

Table 18. Functional and Non-functional requirements validated by the use cases.

	Description	UC1.1	UC1.2	UC2.1	UC2.2	UC3	UC4
F_01	The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED
F_02	The NebulOuS platform offers a lightweight resilient event pub/sub mechanism with persistence, access control and consumer-group capacities to orchestrate communication between the different modules of the solution.			PASSED	PASSED		PASSED
F_03	The system shall provide a graphic user interface for the administration of the deployed components	PASSED				PASSED	PARTIALLY PASSED
F_04	In addition to the Processing Capacity the system shall take into consideration the following factors when deciding on the deployment location of software components/containers: Bandwidth, Storage Capacity, Power Availability, Physical Location of edge device	PASSED		PASSED	No longer relevant to the UC	No longer relevant to the UC	PARTIALLY PASSED
F_05	NebulOuS should constantly monitor the QoS required by the different application components (as specified by their SLA) and apply necessary actions to remediate QoS violations			PARTIALLY PASSED	PASSED	PASSED	
F_06	The NebulOuS platform provides infrastructure monitoring capabilities	PASSED					

F_07	The NebulOuS platform provides application components lifecycle management for all components of the application regardless of where a component is deployed (cloud, edge, IoT device)	PASSED					
F_08	The NebulOuS platform provides a deployment mechanism that makes it easy to deploy and manage data processing pipelines across various resource types (cloud/edge/fog).	PASSED					
F_09	The NebulOuS platform provides near-real-time communication between application components.			PASSED	PASSED		
F_10	NebulOuS offers a mechanism to partition streaming analysis jobs between a variable number of workers (similar to Spark partitioning)				PASSED		
F_11	NebulOuS should offer a mechanism to deploy and invoke serverless computation functions.				PASSED		
F_12	Monitor and predict downtime of registered resources to maximize time/cost ratio. NebulOuS should be able to accommodate applications that prioritize resources with lower down time, in comparison with the opposite					PASSED	
F_13	Handle paused or dropped tasks					PARTIALLY PASSED	
F_14	The system shall be able to be started and operated during run-time with minimal prior training						PASSED
NF_01	Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions	PASSED	PASSED	PARTIALLY PASSED	PASSED	PASSED	PASSED

NF_02	The NebulOuS platform will be able to operate even if available bandwidth between edge infrastructure and the cloud is limited or connection is temporarily lost	PASSED				PASSED	PASSED
NF_03	The platform provides secure access to the resources			PASSED	PASSED	PASSED	PASSED
NF_04	NebulOuS platform should be able to deploy applications in a Raspberry Pi 3 Model B+ or similar ARM architecture			PASSED			
NF_05	Data transferred between edge nodes and between edge nodes and clouds should be secured			PASSED	PASSED		
NF_06	The system provides elasticity capabilities along with the cloud continuum, so HW resources are provisioned or freed depending on the workload				PASSED		
NF_07	The system shall be able to cope with a sudden loss of connection to individual computational units						PASSED
NF_08	The system shall be able to adapt the maximum load of the edge devices to the available power supply						PARTIALLY PASSED
NF_09	The NebulOuS platform provides near-real-time processing of big data			PASSED	PASSED		
NF_10	The NebulOuS platform will allow effective scaling of the cloud resources (in the edge/cloud computing)	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED
NF_11	The NebulOuS platform ensures data privacy when transferring data from one node to another	PASSED	PASSED	PASSED	PASSED	PASSED	

7. OPENCALLS' EVALUATION

In addition to the internal pilot demonstrators, the NebulOuS platform was further validated through two consecutive Open Calls, which engaged external projects in adopting, integrating, and evaluating the platform in diverse real-world application scenarios. Open Call 1 and Open Call 2 targeted complementary use cases spanning energy management, industrial safety, environmental monitoring, smart cities, digital twins, and immersive applications. Together, these Open Calls provided an independent and technology-agnostic validation of the NebulOuS Meta-Operating System, demonstrating its applicability beyond the original project scope and its ability to support heterogeneous workloads, infrastructures, and operational constraints across the cloud–edge continuum.

The detailed descriptions of each Open Call project and their individual activities are presented in Deliverable WP7. In this deliverable, the focus is placed on summarising the final outcomes achieved by these projects, highlighting their contributions to the overall validation of the NebulOuS platform.

Across both Open Calls, a broad and consistent set of NebulOuS core features was exercised and validated. All projects relied on the NebulOuS application modelling and deployment mechanisms to define multi-component applications and orchestrate them across edge, fog, and cloud resources. Dynamic resource brokering and automated deployment were extensively used to adapt applications to changing workloads, infrastructure availability, and latency constraints.

Monitoring and observability capabilities were a central aspect of validation in both Open Calls. Projects actively used metric collection, time-series storage, and real-time monitoring to track application performance, resource utilisation, and operational health. These metrics were further leveraged to define and evaluate Service Level Objectives (SLOs), particularly in scenarios requiring low latency, high availability, or energy efficiency.

Several projects in both Open Calls validated NebulOuS support for adaptive behaviour, including scaling decisions, workload redistribution, and resilience to edge or network failures. Open Call 1 primarily focused on establishing stable edge–cloud pipelines and validating end-to-end deployments in industrial and environmental domains, while Open Call 2 placed stronger emphasis on resilience, scalability, and responsiveness under highly dynamic and resource-constrained conditions.

The Open Calls also confirmed the platform's flexibility in supporting AI-driven analytics, IoT data pipelines, and event-driven processing, as well as its ability to integrate heterogeneous sensors, edge devices, and third-party services.

Funded by the European Union

Table 19. Summary of Opencalls evaluation

Acronym	Short summary	Outcome
HeonICT	An IoT-based energy management solution for homes and small office buildings, aiming to optimise the balance between harvested photovoltaic energy and the consumption of high-demand devices such as heat pumps and electric vehicle chargers.	NebulOuS was used to deploy and manage scalable AI and machine-learning components across the cloud–fog–edge continuum. The platform enabled dynamic optimisation and reconfiguration of computing resources based on changing energy supply and demand conditions, supporting continuous and adaptive energy management. NebulOuS supported the large-scale deployment of the workflow application, ensured that as it expands to new facilities and increases in complexity, the required computing resources are efficiently managed and available.
GUARDIAN	A real-time Health, Safety, and Environment (HSE) monitoring system for industrial environments, combining ultra-wideband positioning, air quality sensors, edge devices, and computer vision to detect risks, monitor worker safety, and improve compliance with safety protocols.	NebulOuS supported the integration and orchestration of heterogeneous sensing and processing components, enabling scalable data processing and real-time monitoring across distributed environments. The platform facilitated reliable deployment and operation of the monitoring services while supporting future extensibility of the solution. Optimized Computer Vision model achieved over 90% detection accuracy, with an inference latency of less than 300ms, ensured real-time monitoring of safety compliance violations.
PAUSA	A serverless, data-driven platform for urban sustainability, focused on environmental monitoring, predictive air-quality modelling, and decision support for urban planners and public health authorities. The application processes real-time pollution data to generate actionable insights for sustainable city management	NebulOuS provided the meta-operating system for deploying a serverless architecture across distributed infrastructures. The platform enabled elastic resource allocation, real-time data processing, and predictive analytics, supporting scalable and efficient handling of urban environmental data.
DYEMAC	An IoT-based predictive maintenance and condition monitoring solution for the dosage phase of high-temperature textile dyeing machines, aiming to improve efficiency, reduce environmental impact, and support multi-factory industrial deployments.	NebulOuS was used to model, deploy, and operate a containerised application across the edge–cloud continuum. Core platform features such as monitoring, SLOs, utility functions, and automated orchestration enabled adaptive scaling, reliable edge–cloud coordination, and continuous industrial operation under varying workloads. Successful NebulOuS-based application modelling and deployment, analysis of completed dosage phases in under 90 seconds, reduction of pump performance analysis time to under 10 minutes, daily energy consumption below 720 Wh per factory, and a data analysis and storage success rate exceeding 99%
Smart4Boxing	A collaborative gamified physical exercise application deployed in	NebulOuS was used to dynamically decide where computation should be executed (local edge devices

	public spaces such as coffee shops, where users interact through real physical movements captured by wearable sensors. The solution aims to deliver an engaging, low-latency user experience while minimising reliance on costly cloud resources and maintaining high-quality service under variable workloads.	or remote cloud resources) based on context, utility-based evaluation, workload characteristics, monitored system metrics, and predefined SLO thresholds, and user-experience requirements. Monitoring rules and meta-monitoring policies enabled adaptive deployment decisions, optimising resource usage while ensuring smooth and responsive gameplay in real-world environments.
AquaSense	A real-time water quality monitoring system that collects data from distributed sensors, performs preprocessing and AI-based pollution detection, and delivers timely alerts to authorities and industrial operators. The application addresses latency, scalability, and bandwidth limitations typical of centralised IoT platforms.	NebulOuS enabled the modelling and deployment of a multi-stage data processing pipeline across edge and cloud infrastructures. Geographic-aware orchestration and dynamic resource allocation reduced latency and bandwidth consumption while allowing the system to scale efficiently under increased data volumes, validating NebulOuS for large-scale IoT monitoring scenarios. NebulOuS's orchestration reduce latency to under 2 seconds, scale to handle 10x data spikes, and cut costs by 30% through dynamic resource allocation. The system processed data from 100+ sensors across urban and rural sites.
URBAN-EYE	A smart-city environmental monitoring solution combining UAV-based thermal sensing, satellite data, and AI analytics to detect and predict Urban Heat Island (UHI) effects. The system produces near real-time heat maps, alerts, and decision-support insights for urban planners and authorities.	NebulOuS provided edge-to-cloud orchestration and automated deployment of data ingestion, AI processing, and visualisation components. The platform supported low-latency analytics, scalable processing of geospatial data, and seamless integration of heterogeneous sensing sources, enabling real-time urban monitoring and decision support. The main KPIs achieved include a 40% reduction in data processing time, hotspot alert latency below 5 seconds, dashboard refresh rates of 5 seconds or less and scalable processing validated for larger urban areas through NebulOuS-enabled orchestration
BUILD-DT	A real-time digital twin for construction sites focused on occupational safety and health, integrating PPE detection, fall detection, and worker health monitoring under highly dynamic and resource-constrained conditions. The system must remain responsive despite fluctuating workloads and potential edge failures.	NebulOuS enabled SLO-aware orchestration, adaptive scaling, and resilient deployment across edge and cloud resources. Core platform components supported monitoring, anomaly detection, and dynamic workload reallocation, ensuring continuous operation and safety-critical responsiveness in demanding construction site environments. Key KPIs achieved include integration of geographically distributed sites, deployment on select edge devices and execution of all planned workload scenarios, demonstrating clear gains in resilience, scalability, and operational robustness enabled by NebulOuS
AQUAGUARD	An end-to-end aquaculture monitoring and decision-support system combining IoT sensors, edge-based video acquisition, and	NebulOuS was used to orchestrate cloud-to-edge deployments, manage data exchange via MQTT, and scale analytics pipelines transparently from single tanks to large installations. The platform enabled

Funded by the European Union

	AI analytics to improve fish welfare, feeding efficiency, and environmental sustainability. The solution targets scalable operation across multiple tanks and sites.	modular deployment, reliable data processing, and adaptive scaling while maintaining consistent system behaviour across increasing operational complexity

8. CONCLUSIONS

The work carried out under WP6 successfully fulfilled its objectives, demonstrating that the NebulOuS platform can support hyper-distributed applications across heterogeneous environments. The evaluation and adaptation of the integration strategy ensured that all software components—whether newly developed or adapted—were seamlessly incorporated into a unified platform. Comprehensive testing confirmed the functional correctness and stability of the system, while continuous validation cycles enabled early detection and resolution of defects. Functional validations verified that all major system components operate as intended and fulfil the defined requirements.

The Pilots and the applicants from the open calls tested and validated the NebulOuS platform according to their technological vision. The pilot evaluations demonstrated that NebulOuS successfully addressed the core needs of all six use cases by enabling decisions relating to cost-effectiveness, performance, and deployment flexibility across the cloud–edge continuum. The pilots confirmed that:

- NebulOuS supports diverse deployment environments and application types.
- Optimisation and self-adaptation mechanisms are beneficial across domains.
- Custom metrics and SLO-driven monitoring add significant value for real-world operations.
- Hybrid execution strategies consistently achieve the best balance between performance and cost.

Overall, the final assessment confirms that the NebulOuS platform reached a high level of technical maturity and readiness by the end of the project. The achieved results validate both the architectural vision and the practical applicability of the solution, paving the way for extended evaluation, further adaptations, and potential market adoption.

9. ANNEX

DETAILS OF PREPARED TEST CASES

The term *manually-managed node* refers to a computing node manually allocated by a person. This kind of node requires the NebulOuS agent to be installed on it and registered in NebulOuS for it to be usable. In contrast, the term *NebulOuS-managed* node refers to a computing node that has been automatically allocated by NebulOuS from a cloud provider registered on it.

ID	**Title**
TC_01	*NebulOuS build*
Objective	
Test that the NebulOuS (core and agent) can be built from source.	
Preconditions	
1. *User has a computer with necessary hardware and software requirements to build the NebulOuS source.* 2. *The server can download NebulOuS source from the online repository.* 3. *User has an open terminal in the server.*	
Steps	
Step 1 **Action** *User executes the NebulOuS build process.* **Expected results** *All resources necessary for installing NebulOuS core and agent are generated (binaries, default configuration files, etc…). Execute test case "TC_02 NebulOuS core installation" to verify it.*	

ID	**Title**
TC_02	*NebulOuS core installation*
Objective	
Test that the NebulOuS core installation process can be executed.	
Preconditions	
1. *A server with NebulOuS core minimal requirements is available.* 2. *User has necessary permissions to install new software on the server.* 3. *The user has access to NebulOuS core binaries (either from a public repository or locally compiled).* 4. *User has an open terminal in the server.*	
Steps	

Step 1 **Action** *User executes the NebulOuS core installation process.* **Expected results** *NebulOuS core is correctly installed.* *All its core components are up and running.*

ID	**Title**
TC_03	*NebulOuS core installation fails when requirements are not met.*
Objective	
If an admin tries to install NebulOuS core on a machine without necessary requirements the installation process fails and user can easily identify the source of the problem.	
Preconditions	
1. *A machine that doesn't comply with minimum requirements (hardware and/or user privileges) for NebulOuS core is available.*	
Steps	
Step 1 **Action** *User logs in to the machine and starts installation of NebulOuS core.* **Expected results** *NebulOuS core installation fails. User can see relevant logs that helps identify the problem.*	

ID	**Title**
TC_04	*NebulOuS core recovers after failure of any of its components.*
Objective	
NebulOuS core tries to recover from failure of any of its components. Meanwhile, sufficient logs to identify the root of the problem are produced.	
Preconditions	
1. *NebulOuS core is correctly installed.* 2. *All its core components are up and running. This test is to be repeated by each component.*	
Steps	
Step 1 **Action** *The NebulOuS core component under test fails (its process is killed).* **Expected results** *User can see relevant logs.*	

NebulOuS core actively tries to resume normal operation

ID	**Title**
TC_05	*Uninstall NebulOuS core.*
Objective	
NebulOuS core can be uninstalled, and all resources removed.	
Preconditions	
1. *NebulOuS core is installed on a server and running correctly.* 2. *A manually-managed node has NebulOuS agent installed and it is connected to the NebulOuS core.* 3. *A NebulOuS cloud provider is registered.* 4. *Two versions of "Dummy APP" are registered, one that requires the APP to be deployed on the manually-managed node and one that requires to be deployed using resources from the NebulOuS cloud provider. An instance of each application is running (one using the manually-managed node and another using a NebulOuS managed node from the cloud provider).* 5. *User is logged in the server and has appropriate rights uninstalling NebulOuS core.*	
Steps	
Step 1 **Action** *User uninstall NebulOuS core.* **Expected results** *All NebulOuS core binaries, services, logs are removed from the server dedicated to NebulOuS core.* *All binaries, data and logs pertaining to applications managed by NebulOuS are removed from the server dedicated to NebulOuS core.* *All application components deployed by NebulOuS in nodes are removed (logs, binaries, services, etc…).* *NebulOuS-managed node is destroyed.*	

ID	**Title**
TC_06	*Login into NebulOuS web UI (invalid credentials)*
Objective	
Test user cannot log-in into NebulOuS web UI using invalid credentials	
Preconditions	
1. *NebulOuS core is installed on a server and running correctly.* 2. *User can reach the NebulOuS web UI.*	
Steps	

Step 1 **Action** *User tries to log-in in the NebulOuS web UI using invalid credentials* **Expected results** *NebulOuS web UI shows an alert indicating that the credentials are invalid* **Step 2** **Action** *Execute TC_09 Unauthenticated user can't interact with NebulOuS web UI* **Expected results** *Test passes*

ID	**Title**
TC_07	*Login into NebulOuS web UI (valid credentials)*
Objective	
Test user can log-in into NebulOuS web UI	
Preconditions	
1. *NebulOuS core is installed on a server and running correctly.* 2. *User can reach the NebulOuS web UI.*	
Steps	
Step 1 **Action** *User tries to log-in in the NebulOuS web UI using valid credentials* **Expected results** *NebulOuS web UI redirects user to home screen.*	

ID	**Title**
TC_08	*Logout from NebulOuS web UI*
Objective	
Test user can log-out from NebulOuS web UI	
Preconditions	
1. *NebulOuS core is installed on a server and running correctly.* 2. *User is logged in the web UI.*	
Steps	
Step 1 **Action** *User logs-off NebulOuS web UI.* **Expected results**	

NebulOuS web UI redirects user to login screen. **Step 2** **Action** *Execute TC_09 Unauthenticated user can't interact with NebulOuS web UI* **Expected results** *Test passes*

ID	**Title**
TC_09	*Unauthenticated user can't interact with NebulOuS web UI*
Objective	
If a user is not authenticated in the web UI, all of the web UI deep URLs direct the user to the login page.	
Preconditions	
1. *NebulOuS core is installed on a server and running correctly.* 2. *User can reach the NebulOuS web UI.*	
Steps	
Step 1 **Action** *User manually navigates to the following urls:* - *{nebulous_base_url}/applications* - *{nebulous_base_url}/users* - *{nebulous_base_url}/applications/resources* **Expected results** *NebulOuS web UI shows an alert indicating that the user is not logged in and redirects the user to the login screen.*	

ID	**Title**
TC_10	*Resource manager can onboard manually-managed nodes*
Objective	
Validate that a resource manager can onboard manually-managed nodes in NebulOuS	
Preconditions	
1. *A NebulOuS core installation is up and running. It's IP is known.* 2. *Two users exist:* a. *Device owner: A user with "device owner" role in NebulOuS.* b. *Resource manager: A user with "resource manager" role in NebulOuS.* 3. *A machine with minimum requirements for NebulOuS agent is available. NebulOuS has network visibility over it. A pair of SSH keys of the machine linked with a user account with appropriate rights to install new software are known by the user "device owner".*	
Steps	
Step 1	

Action *User "device owner" logs in to NebulOuS and fills in a device registration request. The user provides a device Id, the public IP and the SSH keys.* **Expected results** *A device registration request is created.* *NebulOuS automatically collects the device capabilities.* **Step 2** **Action** *User "resource manager" logs in to NebulOuS and performs the "ask device-onboarding" step of the resource discovery mechanism for the recently created registration request.* **Expected results** *NebulOuS installs the NebulOuS agent in the device and the device registration process is completed successfully.* *Users Device owner and resource manager are informed about the status of the process.* *The device is made available to NebulOuS for deploying payloads.*

ID	**Title**
TC_11	*Onboarding of a manually-managed node fails.*
Objective	
Validate that sufficient feedback is given to the user when the process of onboarding a manually-managed node fails.	
Preconditions	
1. *A NebulOuS core installation is up and running. It's IP is known.* 2. *Two users exists:* a. *Device owner: A user with "device owner" role in NebulOuS.* b. *Resource manager: A user with "resource manager" role in NebulOuS.* 3. *A machine **WITHOUT** the minimum requirements for NebulOuS agent is available. NebulOuS has network visibility over it. A pair of SSH keys of the machine linked with a user account with appropriate rights to install new software are known by the user "device owner".*	
Steps	
Step 1 **Action** *User "device owner" logs in to NebulOuS and fills in a device registration request. The user provides a device Id, the public IP and the SSH keys. Some of the provided values are intentionally invalid.* **Expected results** *A device registration request is created.* *NebulOuS tries to collect the device capabilities but fails due to the invalid parameters provided.* *Device owner and resource manager are informed about the status of the process. Relevant error logs are collected and made can be seen by these users.*	

Step 2 **Action** *User "device owner" logs in to NebulOuS and fills in a device registration request. He provides valid values for device Id, the public IP and the SSH keys.* **Expected results** *A device registration request is created.* *NebulOuS tries to collect the device capabilities but fails due to the device not meeting the requirements for installing the NebulOuS agent.* *Device owner and resource manager are informed about the status of the process. Relevant error logs are collected and made can be seen by these users.*

ID	Title
TC_12	*Device owner can de-register his devices*
Objective	
Validate that a device owner can de-register a device.	
Preconditions	
1. A NebulOuS core installation is up and running. It's IP is known. *2. Two users exist:* *a. Device owner: A user with "device owner" role in NebulOuS.* *b. Resource manager: A user with "resource manager" role in NebulOuS.* *3. A node owned by the "device owner" has been previously registered in NebulOuS.* *4. "Dummy application" is running on said node.*	
Steps	
Step 1 **Action** *User "device owner" logs in to NebulOuS and requests the de-registration of its device.* **Expected results** *NebulOuS de-registers the node.* *All NebulOuS agent binaries, services, logs are removed from the manually-managed node.* *Application running on the node are terminated.* *All binaries, data, logs pertaining to user applications deployed with NebulOuS in the agent are removed.* *Device owner is informed about the status of the process.*	

ID	Title
TC_13	*Resource manager can de-register devices*
Objective	
Validate that a device owner can de-register devices.	
Preconditions	
1. A NebulOuS core installation is up and running. It's IP is known.	

2. *Two users exist:*   a. *Device owner: A user with "device owner" role in NebulOuS.*   b. *Resource manager: A user with "resource manager" role in NebulOuS.* 3. *A node owned by the "device owner" has been previously registered in NebulOuS.* 4. *"Dummy application" is running on said node.*
Steps
Step 1 **Action** *User "Resource manager" logs in to NebulOuS, lists all active devices and requests the de-registration of the device under test.* **Expected results** *NebulOuS de-registers the node.* *All NebulOuS agent binaries, services, logs are removed from the manually-managed node.* *Application running on the node are terminated.* *All binaries, data, logs pertaining to user applications deployed with NebulOuS in the agent are removed.* *Resource manager is informed about the status of the process.*

ID	**Title**
TC_14	Admin can install NebulOuS agent.
Objective	
Validate that admin can install NebulOuS agent if requirements are meet.	
Preconditions	
• *A NebulOuS core installation is up and running. It's IP is known.* • *A machine with minimum requirements for NebulOuS agent is available. User has enough permissions to install software on the machine.*	
Steps	
Step 1 / **Action**: *User logs in to the machine and starts installation [following the provided instructions](#).* / **Expected results**: *NebulOuS agent is successfully installed. User can see relevant logs.* *The new NebulOuS manually-managed node is visible from the NebulOuS administration panel and can be used.*	

ID	**Title**
TC_15	NebulOuS agent installer fails when requirements are not met.

Objective
If an admin tries to install NebulOuS agent on a machine without necessary requirements the installation process fails and user can easily identify the source of the problem.
Preconditions
• *A machine that doesn't comply with minimum requirements for NebulOuS agent is available. User has enough permissions to install software on the machine.*
Steps

Step 1
Action
User logs in to the machine and starts installation following the provided instructions.
Expected results
NebulOuS agent installation fails. User can see relevant logs that helps identify the problem.

ID	**Title**
TC_16	*NebulOuS agent can be uninstalled.*
Objective	
NebulOuS agent can be uninstalled.	
Preconditions	
1. *NebulOuS agent is installed on a manually-managed node and running correctly.* 2. *User is logged in the machine and has appropriate rights uninstalling NebulOuS agent.* 3. *"Dummy APP" is registered in NebulOuS and an instance is running on the node.*	
Steps	

Step 1
Action
User follows the instructions provided on NebulOuS documentation to uninstall NebulOuS agent .
Expected results
All NebulOuS agent binaries, services, logs are removed from the manually-managed node. *Application running on the node is terminated.* *All binaries, data, logs pertaining to user applications deployed with NebulOuS in the agent are removed.*

ID	**Title**
TC_17	*NebulOuS agent recovers after failure with communication with NebulOuS core.*
Objective	
NebulOuS agent tries to recover from failure in communications with NebulOuS core and provides sufficient logs to identify the root of the problem.	

Preconditions
1. *NebulOuS agent is installed on a manually-managed node and running correctly.* 2. *User is logged in the machine and has appropriate rights for accessing NebulOuS agent logs.* 3. *"Dummy APP" is registered in NebulOuS and an instance is running on the node.*
Steps
Step 1 **Action** *NebulOuS agent can no longer contact NebulOuS core due to some network issues (e.g.: LAN cable disconnected, NebulOuS core goes down, etc....).* **Expected results** *User can see relevant logs.* *NebulOuS agent actively tries to resume normal operation.* *Application running on the node is terminated.* **Step 2** **Action** *Link with NebulOuS core is recovered.* **Expected results** *NebulOuS agent resumes normal operation briefly.* *User can see relevant logs.*

ID	**Title**
TC_18	*NebulOuS agent recovers after one of its components fails.*
Objective	
NebulOuS agent tries to recover from failure of any of its components and provides sufficient logs to identify the root of the problem.	
Preconditions	
1. *NebulOuS agent is installed on a manually-managed node and running correctly.* 2. *User is logged in the machine and has appropriate rights for accessing NebulOuS agent logs.* 3. *"Dummy APP" is registered in NebulOuS and an instance is running on the node.*	
Steps	
Step 1 **Action** *A component of NebulOuS agent fails (e.g.: the process is manually killed).* **Expected results** *User can see relevant logs.* *NebulOuS agent actively tries to resume normal operation.* *Application running on the node is terminated.*	

Step 2
Action
NebulOuS agent detects that one of its sub-components has failed and remediates the situation to make it again available.
Expected results
NebulOuS agent resumes normal operation briefly.
User can see relevant logs.

ID	**Title**
TC_19	*Admin can register cloud providers in NebulOuS.*
Objective	
Validate that admin can register cloud providers in NebulOuS using the administrative web UI.	
Preconditions	
A NebulOuS core installation is up and running. *User has appropriate rights to manage NebulOuS cloud providers.* *User is logged in NebulOuS web UI.* *User has administrative access to a cloud provider (AWS, OpenStack, etc..)* Test with each one.*	
Steps	
Step 1 **Action** *User navigates to the cloud provider management section of the NebulOuS web UI and registers a new cloud provider.* **Expected results** *The cloud provider under test is available to NebulOuS.*	

ID	**Title**
TC_20	*Admin can de-register cloud providers in NebulOuS.*
Objective	
Validate that admin can de-register cloud providers in NebulOuS using the administrative web UI.	
Preconditions	
1. *A NebulOuS core installation is up and running.* 2. *User has appropriate rights to manage NebulOuS cloud providers.* 3. *User is logged in NebulOuS web UI.* 4. *A cloud provider is registered in NebulOuS.* 5. *"Dummy APP" is registered in NebulOuS.* 6. *A NebulOuS-managed node is active and running the application.* 7. *User has administrative access to a cloud provider (AWS, OpenStack, etc..) * Test with each one.*	

Steps
Step 1 **Action** *User navigates to the cloud provider management section of the NebulOuS web UI and de-registers the cloud provider.* **Expected results** *The cloud provider under test is no longer available to NebulOuS.* *NebulOuS-managed nodes are deleted, and resources freed. (Check via the Cloud provider UI).* *NebulOuS cannot deploy the "Dummy APP".*

ID	**Title**
TC_21	*App deployment on manually-managed nodes*
Objective	
Test if user can deploy an APP using NebulOuS on manually-managed nodes	
Preconditions	
1. *NebulOuS core is running.* 2. *A NebulOuS node is active and registered in NebulOuS.* 3. *User has appropriate access rights to deploy APPs in NebulOuS.*	
Steps	
Step 1 **Action** *User registers an application that can be deployed in the existing NebulOuS node. For this, it executes the following steps:* - *Logs in the NebulOuS web UI.* - *Defines application using the web UI.* - *Launches the application.* **Expected result** *NebulOuS selects a node to execute the application service and it is deployed. Application starts operation. User can see application deployment logs as well as application execution logs from the web UI.*	

ID	**Title**
TC_22	*App deployment on manually-managed node fails due to lack of resources*
Objective	
Test if sufficient debug information is provided when an APP deployment fails due to lack of resources to deploy it.	
Preconditions	

1. *NebulOuS core is running.* 2. *A manually-managed node is active and registered in NebulOuS.* 3. *User has appropriate access rights to deploy APPs in NebulOuS.*
Steps
Step 1 **Action** *User registers an application that can't be deployed on the existing NebulOuS manually-managed for one of the following reasons:* 1. *User doesn't have permissions to deploy APPs on the node.* 2. *Application requirements and the node offerings don't match.* *User launches the application.* **Expected result** *NebulOuS tries to find a node to deploy the application but none is available. Sufficient information is provided to the user to identify the problem.*

ID	**Title**
TC_23	*App deployment using NebulOuS cloud providers*
Objective	
Test if user can deploy an APP using NebulOuS on a NebulOuS cloud provider.	
Preconditions	
1. *NebulOuS core is running.* 2. *A NebulOuS cloud provider is registered in NebulOuS.* 3. *User has appropriate access rights to deploy APPs in NebulOuS.*	
Steps	
Step 1 **Action** *User registers an application that can be deployed in a node provided by the registered NebulOuS cloud provider. For this, it executes the following steps:* - *Logs in the NebulOuS web UI..* - *Defines application using the web UI.* - *Launches the application.* **Expected result** *NebulOuS creates a new NebulOuS-managed node on the registered NebulOuS cloud provider. The application is deployed in the new node. Application starts operation. User can see application deployment logs as well as application execution logs from the web UI.*	

ID	**Title**
TC_24	*App deployment on NebulOuS-managed node fails due to lack of resources*
Objective	
Test if sufficient debug information is provided when an APP deployment fails due to lack of resources to deploy it.	
Preconditions	
1. *NebulOuS core is running.* 2. *A NebulOuS cloud provider is registered in NebulOuS.* 3. *User has appropriate access rights to deploy APPs in NebulOuS.*	
Steps	
Step 1 **Action** *User registers an application that can't be deployed using resources from the registered NebulOuS cloud provider for one of the following reasons:* 3. *User doesn't have permissions to deploy APPs using resources from the cloud provider.* 4. *The cloud provider can't fulfil the hardware requirements of the application.* *User launches the application.* **Expected result** *NebulOuS tries to allocate new NebulOuS managed node but fails. Sufficient information is provided to the user to identify the problem.*	

ID	**Title**
TC_25	*NebulOuS reacts to NebulOuS* node *failure*
Objective	
Test if NebulOuS can detect that a NebulOuS node crashes and re-assigns applications components to other available workers.	
Preconditions	
1. *NebulOuS core is running.* 2. *Two NebulOuS nodes are active and registered in NebulOuS.* 3. *An instance of "Dummy APP" is registered in NebulOuS. The application component is correctly deployed at one of the NebulOuS nodes.*	
Steps	
Step 1 **Action** *Manually kill (e.g.: power off) the NebulOuS node executing the application component.* **Expected result**	

NebulOuS detects that the application component is not being executed by any node. *NebulOuS selects a new node to execute the application component and it is deployed. Application resumes operation. User can see relevant information of the process through the web UI.*

ID	**Title**
TC_26	*NebulOuS scales up and down applications to comply with SLO*
Objective	
Test if NebulOuS can detect that the SLO for an application component that allows scaling up is not being met and deploys new instances of the component.	
Preconditions	
1. *NebulOuS core is running.* 2. *Three NebulOuS nodes are active and registered in NebulOuS.* 3. *Application of "Scaling APP" is registered in NebulOuS. The APP SLO is configured such as max_age can't be bigger than 30 seconds. Number of worker component instances can range from 0 to infinite. The objective function is set to minimize the number of instances of worker component.*	
Steps	
Step 1 **Action** *User logs in into NebulOuS web UI and launches an instance of the application "Scaling APP".* **Expected result** *NebulOuS deploys an instance of the APP component "controller". No instance of worker component is created.* **Step 2** **Action** *User interacts with the REST API offered by the controller component and sends a new job with a value of t = 30* **Expected result** *After 30 seconds, NebulOuS detects that the SLO of the application has been violated and creates a new instance of worker component.* **Step 3** **Action** *User interacts with the REST API offered by the controller component and sends 6 new jobs with a value of t = 30* **Expected result** *After 30 seconds, NebulOuS detects that the SLO of the application has been violated and creates a new instance of worker component. This happens multiple times more without the user interacting with the APP.* **Step 4**	

Action *None* **Expected result** *All requests are completed and associated worker components terminated.* *Resources are freed.*

ID	**Title**
TC_28	*NebulOuS manages the execution of IoT data processing pipelines*
Objective	
Test if user can express an IoT data processing pipeline as a NebulOuS application and NebulOuS efficiently manages the execution of the pipeline.	
Preconditions	
1. *NebulOuS core is running.* 2. *Three NebulOuS nodes are active and registered in NebulOuS.* 3. *Application of "IoT APP" is registered in NebulOuS. The APP SLO is configured such as max_age for "step1_output" can't be bigger than 30 seconds. The objective function is set to minimize the number of instances of step 2.*	
Steps	
Step 1 **Action** *User logs in into NebulOuS web UI and launches an instance of the application "Scaling APP".* **Expected result** *NebulOuS deploys an instance of the APP step "step1". No instance of step2 is created.* **Step 2** **Action** *User publishes a message to the pub/sub queue "stream_input" of the application, with the body {"group":"g1","t":10}* **Expected result** *NebulOuS detects that the max_age of "step1_output" queue is bigger than 30 and creates an instance of "step2". Pending messages are processed and max_age for the queue goes to 0. NebulOuS stops the execution of "Step 2" instance.* **Step 2** **Action** *User publishes a burst of 500 messages to the pub/sub queue "stream_input" of the application, with a fixed value for "t" (10) and different values for "group" ("g1","g2","g3")* **Expected result** *NebulOuS detects that the max_age of "step1_output" queue is bigger than 30 and creates an instance of "step2". After a short period of time, NebulOuS reevaluates the SLO and confirms that the max_age is still bigger than 30 seconds and creates more instances of the "step2". This goes on until all the pending messages are processed and max_age for the queue goes to 0. NebulOuS stops the execution of "Step 2" instances.*	

ID	**Title**
TC_29	*Secure Deployment of Applications via NebulOuS*
Objective	
Test the secure deployment of applications through NebulOuS, ensuring that deployment configurations adhere to common security practices.	
Preconditions	
1. *NebulOuS core is running.* 2. *A NebulOuS cloud provider is registered.* 3. *User has appropriate access rights to deploy apps in NebulOuS.*	
Steps	

Step 1
Action
User logs into the NebulOuS web UI, uploads application binaries, defines application using the web UI with security configurations (such as network policies, resource limits, and security contexts), and launches the application
Expected result
NebulOuS deploys the application securely. The application's deployment configurations adhere to predefined security policies, and the user can view deployment and security logs through the web UI.

ID	**Title**
TC_31	*Network Isolation and Security Policy Compliance in Application Deployments*
Objective	
To ensure that network isolation and security policies are correctly applied to applications deployed through NebulOuS.	
Preconditions	
1. *NebulOuS core is running.* 2. *A NebulOuS cloud provider is registered.*	
Steps	

Step 1
Action
User deploys an application via NebulOuS web UI, specifying network isolation and security policies as part of the deployment process.
Expected result
The application is deployed with the specified network isolation. Attempts to access the application from unauthorized networks are denied, and the compliance with the specified security policies is logged.

ID	**Title**
TC_32	*Validation of RBAC Policies in NebulOuS on a Kubernetes Cluster*
Objective	

To ensure that Role-Based Access Control (RBAC) policies are correctly enforced in the Kubernetes cluster managed by NebulOuS.
Preconditions
1. *NebulOuS core is running.* 2. *A NebulOuS cloud provider is registered.* 3. *RBAC policies are defined in the Kubernetes cluster.*
Steps

Step 1
Action
User attempts to perform various operations (create, read, update, delete) on Kubernetes resources based on their RBAC roles.
Expected result
Operations are allowed or denied in accordance with the defined RBAC policies, and relevant logs are generated.

ID	**Title**
TC_35	*Enforcement of Ingress and Egress Network Policies in Kubernetes*
Objective	
To validate the enforcement of ingress and egress network policies for pods in the Kubernetes environment.	
Preconditions	
1. *NebulOuS core is running.* 2. *Kubernetes cluster with network policies is integrated with NebulOuS.*	
Steps	

Step 1
Action
*Define and apply an ingress network policy for a pod (e.g., **nginx-pod**) that allows traffic only from a certain namespace (e.g., **internal**) on a specific port (e.g., 80).*
Expected result
*The ingress policy is successfully applied. **nginx-pod** should only accept traffic on port 80 from pods within the **internal** namespace.*

Step 2
Action
*Test the ingress policy by sending traffic to **nginx-pod** from a pod within the **internal** namespace and then from a pod outside this namespace.*
Expected result
*Traffic from the pod within the **internal** namespace reaches **nginx-pod**. Traffic from the pod outside the **internal** namespace is blocked, and an attempt is logged.*

Step 3
Action

Define and apply an egress network policy for another pod (e.g., ***backend-pod****) that restricts outbound traffic to a specific external IP address range.*
Expected result
The egress policy is successfully applied. ***backend-pod*** *can only initiate outbound traffic to the specified IP address range.*

Step 4
Action
Test the egress policy by attempting to connect from ***backend-pod*** *to an allowed external IP address and then to a disallowed IP address.*
Expected result
Connections to the allowed IP address are successful. Attempts to connect to disallowed IP addresses are blocked and logged.

ID	**Title**
TC_37	*Cloud provider selected based on user preferences*
Objective	
Test if NebulOuS takes into account user preferences when deploying APPs in cloud providers.	
Preconditions	
1. *NebulOuS core is running.* 2. *Two NebulOuS cloud providers are registered in NebulOuS (CloudA and CloudB).* 3. *User has appropriate access rights to deploy APPs in NebulOuS.*	
Steps	
Step 1 **Action** *User logs in the NebulOuS UI and specifies cloud provider preferences in a way that instances from Cloud A are clearly more preferable than instances from Cloud B.* **Expected result** *NebulOuS registers the user cloud preferences* **Step 2** **Action** *User registers the "dummy application" and launches it.* **Expected result** *NebulOuS allocates all application components to instances in Cloud A.* **Step 3** **Action** *User de-registers the application.* **Expected result**	

All resources are freed. **Step 4** **Action** *User changes the cloud provider preferences in a way that instances from Cloud B are clearly more preferable than instances from Cloud A.* **Expected result** *NebulOuS registers the user cloud preferences* **Step 5** **Action** *User registers the "dummy application" and launches it.* **Expected result** *NebulOuS allocates all application components to instances in Cloud B.*

VERIFICATION OF FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Windmill Maintenance use case

ID	**Title**
UC1.1_F_01_TC1	*Evaluate NebulOuS's response in scenarios of limited or no connectivity.*
Objective	
This test aims to assess the robustness of NebulOuS in handling image processing tasks under scenarios where connectivity is limited or completely unavailable.	
Preconditions	
1. *Multiple UAVs are active and capturing images.* 2. *NebulOuS is operational, managing Edge and Cloud nodes.* 3. *An image processing pipeline is established for damage detection and classification.* 4. *Establish a controlled environment where connectivity can be varied. This could involve a network simulator or a real-world setup where network conditions can be altered.*	
Steps	
Execution: • *Begin with normal connectivity and monitor application performance.* • *Gradually reduce the bandwidth to simulate limited connectivity.* • *Finally, cut off connectivity entirely to simulate a no-connection scenario.* ***Monitoring:*** • *Record any delays or failures in processing.* • *Monitor application for any data loss during these transitions.*	
Expected Result	
• *NebulOuS should be able to manage application without losing data.* • *Minimal operational interruption should be observed.*	

ID	**Title**
UC1.1_F_01_TC2	*High-Demand Workload Distribution Test*
Objective	
To evaluate NebulOuS's efficiency in distributing and managing high workloads across both Edge and Cloud Computing resources.	
Preconditions	
1. *Multiple UAVs are active and capturing images.* 2. *NebulOuS is operational, managing Edge and Cloud nodes.* 3. *An image processing pipeline is established for damage detection and classification.* 4. *Prepare multiple UAVs equipped with cameras to capture and send images simultaneously, creating a high-demand scenario.*	
Steps	
Execution: • *Activate all UAVs, ensuring a surge in incoming data.* • *Monitor how NebulOuS scale instances of application.* ***Data Analysis:*** • *Analyse the latency and efficiency of data processing.*	
Expected Result	
• *The system should efficiently utilize Edge Computing for real-time processing and Cloud resources for more computationally intensive tasks.* • *Overall operational efficiency should not degrade despite the increased load.*	

ID	**Title**
UC1.1_F_01_TC3	*Node Deactivation Response Test*
Objective	
To test NebulOuS's resilience and adaptability in reallocating tasks when an Edge node becomes unavailable during operation.	
Preconditions	
1. *Multiple UAVs are active and capturing images.* 2. *NebulOuS is operational, managing Edge and Cloud nodes.* 3. *An image processing pipeline is established for damage detection and classification.*	
Steps	
Execution: • *During peak operation, manually deactivate a pre-selected Edge node.* • *Closely monitor how NebulOuS detects this change and reallocates the application.* ***Response Analysis:*** • *Observe the time taken for the system to adjust.* • *Analyse the efficiency of application reallocation and the impact on overall system performance.*	
Expected Result	
• *NebulOuS should quickly recognize the deactivation of the node and redistribute the tasks among remaining nodes.*	

• *The system should maintain operational continuity, with minimal impact on performance and no data loss.* • *The reallocation should be smooth, without causing bottlenecks or overloading other nodes.*

ID	**Title**
UC1.1_F_03_TC1	*Evaluation of GUI for Component Administration*
Objective	
To assess the effectiveness of the NebulOuS GUI in managing and administering deployed components.	
Preconditions	
1. *The NebulOuS platform is operational, managing Edge and Cloud nodes.* 2. *The GUI for NebulOuS is accessible to the tester.*	
Steps	
Step 1*: Navigate through the GUI to the application registration section.* ***Step 2****: Use the GUI to register the use case application OAM file, metric model, SLO criteria and optimisation function.* ***Step 3****: Request NebulOuS to deploy the newly created application.* ***Step 4****: Navigate through the GUI to the application monitoring section of the newly created application.*	
Expected Result	
• *During the application registration process, the GUI should allow to provide all the necessary input required by NebulOuS to manage the application.* • *The GUI should provide feedback regarding the app deployment status.* • *Once the application is deployed, the GUI should provide real-time details on the status of each component of the application as well as their associated SLOs.*	

ID	**Title**
UC1.1_F_04_TC1	*Deployment decisions based on network speed, storage availability, power supply, and device location.*
Objective	
To assess NebulOuS's ability to make intelligent deployment decisions based on network speed, storage availability, power supply, and device location	
Preconditions	
1. *NebulOuS platform operational.* 2. *Initial deployment of software components completed on multiple User Edge devices with varying network, storage, and power conditions.*	
Steps	
1. *Record initial network, storage, power, and location data for all User Edge devices.* 2. *Deploy software components with varied network, storage, power, and location requirements.* 3. *Note the rationale behind device selection for each component based on network, storage, power, and location.* 4. *Track network, storage, and power changes during and after the deployment.* 5. *Ensure components are appropriately placed for performance and accessibility.*	

6. *Simulate changes in network, storage, and power conditions and observe deployment response.* 7. *Confirm deployed components meet performance and accessibility standards.*
Expected Result
• *NebulOuS intelligently and efficiently manages the deployment of software components, optimizing for current network conditions, resource availability, and physical constraints while maintaining high performance and adaptability to changing conditions.*

ID	**Title**
UC1.1_F_06_TC1	*Testing Infrastructure Monitoring Capabilities*
Objective	
To verify the effectiveness and accuracy of the infrastructure application metrics monitoring system in the NebulOuS platform	
Preconditions	
1. *NebulOuS platform operational with various components deployed.* 2. *Infrastructure monitoring system activated.*	
Steps	
Step 1: *Simulate various operational states (normal, stressed, failure) of the infrastructure components.* ***Step 2***: *Check if the application events monitoring system accurately reflects these states and triggers the necessary actions according to the SLO.* ***Step 3***: *Evaluate the responsiveness and accuracy of the next deployment initiated by the platform in response to the triggered SLO.*	
Expected Result	
• *The NebulOuS platform's components monitoring system successfully detects and reports the various metrics of the infrastructure in real time, proving its reliability and effectiveness and then the platform performs a redeployment as necessary.*	
ID	**Title**
UC1.1_F_07_TC1	*Deployment of the components across different environments.*
Objective	
Thoroughly verify NebulOuS's capability to initiate various application components across different environments.	
Preconditions	
1. *NebulOuS platform operational with application components deployed across Cloud and User Edge devices.* 2. *Lifecycle management system activated.*	
Steps	

***Step 1**: Identify and list specific application components for testing.* ***Step 2**: Initiate the start command for each component.* ***Step 3**: Monitor the startup process, noting the time taken and any anomalies.* ***Step 4**: Validate the operational status of each component post-startup.*
Expected Result
• *The system should demonstrate efficient control over the lifecycle of all application components.* • *Each component should start within 2 minutes, without errors, and transition to an operational state. If an application component requires additional time beyond this threshold for startup due to factors beyond our control: Complexity of the application, Operating system constraints etc.*

ID	**Title**
UC1.1_F_07_TC2	*Stopping the deployed application and all components.*
Objective	
Assess NebulOuS's effectiveness in stopping an application.	
Preconditions	
1. NebulOuS platform operational with the use case application components deployed.	
Steps	
***Step 1**: Instruct NebulOuS to stop the use case application.*	
Expected Result	
• *The application and all its components should no longer be operational or accessible within the NebulOuS environment.* • *Any resources previously allocated to the application should be released and available for other uses.*	

ID	**Title**
UC1.1_F_08_TC1	*Data processing pipeline managing the deployments across different environments.*
Objective	
To evaluate the efficiency, ease of use, and flexibility of the NebulOuS platform's deployment mechanism for data processing pipelines	
Preconditions	
1. Availability of various computing resources (Cloud, User Edge). *2. Data processing pipeline ready for deployment*	
Steps	
***Step 1**: Deploy a data processing pipeline using the NebulOuS platform across different environments.* ***Step 2**: Assess the ease and time taken for the deployment process.* ***Step 3**: Evaluate the operational efficiency of the deployed pipeline in each environment.*	
Expected Result	

• *The data processing pipeline is successfully deployed and operational across the chosen environments.* • *The deployment mechanism proves to be user-friendly, flexible, and efficient, effectively managing deployments across Cloud and Edge environments*

ID	**Title**
UC1.1_NF_10_TC1	*Resource Allocation Monitoring*
Objective	
• *The NebulOuS platform will support resource allocation efficiency by highlighting potential scalability limits.*	
Preconditions	
1. *NebulOuS platform operational with the use case application components deployed.*	
Steps	
Step 1*: Implement metrics to observe the speed of resource provisioning and decommissioning* ***Step 2:*** *Conduct performance assessments focusing on response times, throughput, and efficiency of resource utilization under various workloads.* ***Step 3:*** *Execute stress tests to challenge the system's limits, monitoring for stability and performance degradation.*	
Expected Result	
• *Bottlenecks or inefficiencies in the scaling process are identified.* • *Resource allocation strategies are validated and refined.*	

ID	**Title**
UC1.1_NF_11_TC1	*Detailed Encryption Validation*
Objective	
• *The NebulOuS platform ensures data privacy when transferring data between the nodes.*	
Preconditions	
1. *NebulOuS platform operational with the use case application components deployed.*	
Steps	
Step 1*: Verify the implementation of encryption algorithms.* ***Step 2:*** *Conduct end-to-end encryption testing on a sample data set.* ***Step 3:*** *Assess decryption integrity at the receiving end.*	
Expected Result	
• *All data transferred between nodes and NebulOuS system are encrypted.*	

ID	**Title**
UC1.1_NF_11_TC2	*Comprehensive Audit Trail Review*
Objective	

• *The NebulOuS platform runs verification process to confirm that the data has not been intercepted or tampered with.*
Preconditions
1. *NebulOuS platform operational with the use case application components deployed.*
Steps
***Step 1**: Inspect logs for entry and exit points of data.* ***Step 2:** Validate the timestamps and sequence of events in the transfer process.* ***Step 3:** Check for any unauthorized access attempts logged during the transfer.*
Expected Result
• *No unauthorized users had read or altered the data.*

ID	**Title**
UC1.1_NF_11_TC3	*In-Depth Security Breach Detection Analysis*
Objective	
• *The NebulOuS platform runs verification process to confirm that the data has not been intercepted or tampered with.*	
Preconditions	
1. *NebulOuS platform operational with the use case application components deployed.*	
Steps	
***Step 1**: Perform intrusion detection system (IDS) checks during data transfer.* ***Step 2:** Simulate external attacks and monitor for breaches or vulnerabilities.* ***Step 3:** Test the efficacy of firewall and other security measures in place during data transfers*	
Expected Result	
• *The data transferred between nodes and NebulOuS system has not been intercepted or tampered with.*	

Ubiwhere use case

ID	**Title**
UC1.2_F_01_TC1	*NebulOuS reacts to Service Provider Edge node failure*
Objective	
• *Test if NebulOuS can detect that the Service Provider Edge node is compromised and re-assigns the damage detection to the Cloud.*	
Preconditions	

1. *An Edge node is active.* 2. *NebulOuS is running and an Edge node is registered to it.* 3. *3 cameras are generating video streams.* 4. *Damage detection process is being executed on the Edge node.*
Steps
Step 1
Action
Manually kill (e.g: power off) the Edge node processing the camera video streams
Postconditions
NebulOuS detects that the camera video streams are not being processed by the Edge node.
Expected Result
• *NebulOuS detects that the Service Provider Edge node is not processing the video streams.* • *NebulOuS moves the damage detection process to the Cloud, through an MQTT feed of what the camera streams.*

ID	**Title**
UC1.2_NF_10_TC1	*The NebulOuS platform will allow effective scaling of the Service Provider Edge resources*
Objective	
• *Test if NebulOuS can detect when the workload demands an increase of the processing capacity, dynamically deciding whether to increase the number of the Edge nodes used.*	
Preconditions	
1. *Three Edge nodes are active.* 2. *NebulOuS is running and an Edge node is registered to it.* 3. *Three simulated cameras are generating video streams.* 4. *Damage detection process for the three cameras is being executed on a single Edge node.*	
Steps	
Step 1	
Action	
Force an increase of the workload for the damage detection process. Make the simulated cameras to feed video streams with higher processing complexity (e.g.: more buildings on the frame).	
Postconditions	
SLO for the damage detection process can no longer be satisfied. NebulOuS detects that condition and starts a re-configuration process that utilises at least one more of the available Service Provider Edge resources. When the deployment is completed, the SLO can be met.	

Expected Result
• *NebulOuS detects an SLO violation due to a workload spike and re-configures the application to use more resources.*

ID	**Title**
UC1.2_NF_11_TC1	*Security of data transfer.*
Objective	
• *NebulOuS should guarantee that any communication between nodes is secure.*	
Preconditions	
1. *NebulOuS platform operational* 2. *Application components deployed across Cloud and Edge devices*	
Steps	
Step 1	
Action	
Use a packet sniffing software (Wireshark) or intercept the communication between Edge and Cloud nodes	
Postconditions	
Verify that the communications are encrypted.	
Expected Result	
• *The connection should be secure and any of the interceptions blocked.*	

ID	**Title**
KPI_UC1.2_1 TC1	*The NebulOuS platform will allow effective scaling of the Cloud resources*
Objective	
Test if NebulOuS can detect when the workload demands an increase of the processing capacity, dynamically deciding whether to increase the number of Cloud nodes.	
Preconditions	
1. *An Edge node is active.* 2. *NebulOuS is running and an Edge node is registered to it.* 3. *Three simulated cameras are generating video streams.* 4. *Damage detection process for the three cameras is being executed on a single Edge node.*	
Steps	
Step 1	

Action
Force an increase of the workload for the damage detection process. Make the simulated cameras to feed video streams with higher processing complexity (e.g.: more buildings on the frame).
Postconditions
SLO for the damage detection process can no longer be satisfied. NebulOuS detects that condition and starts a re-configuration process. Given that no more Edge resources are available, NebulOuS must instantiate nodes in the Cloud provider. When the deployment is completed, the SLO can be met.
Expected Result
NebulOuS detects an SLO violation due to a workload spike and re-configures the application to use more Cloud resources.

Mercabarna Intra Logistic use case

ID	**Title**
UC2.1_F_01_TC1	*NebulOuS reacts to worker* node *failure*
Objective	
• *Test if NebulOuS can detect that a worker node crashes and re-assigns license plate reading processes to other available workers.*	
Preconditions	
1. *Two ARM worker nodes are active.* 2. *NebulOuS is running and is connected to the worker nodes.* 3. *A camera is generating a video stream.* 4. *Video stream processing APP is correctly configured in NebulOuS.* 5. *One of the worker nodes is executing the "Vehicle detection and cropping" processes that analyses the video stream of the camera.*	
Steps	
Step 1 **Action** *Manually kill (e.g.: power off) the worker node processing the camera video stream* **Postconditions** *NebulOuS detects a camera video stream is not being processed by any node.*	
Expected Result	
• *NebulOuS detected that the worker node processing the video stream has disappeared.* • *NebulOuS moves the "Vehicle detection and cropping" process to a new worker node.*	

ID	**Title**
UC2.1_F_02_TC1	*Communication between "Vehicle detection and cropping", "License plate detection and reading" and "Management APP".*
Objective	

• *Nebulous can provide correct operation of the Pub/Sub mechanism with two instances of "Vehicle detection and cropping" four instances of "License plate detection and reading" and the Management APP.*
Preconditions
1. *Five worker nodes are active.* 2. *NebulOuS is running and is connected to the worker nodes.* 3. *Two cameras are generating a video stream.* 4. *Video stream processing APP is correctly configured in NebulOuS.* 5. *Management APP is correctly configured in NebulOuS.* 6. *An instance of "Vehicle detection and cropping" is deployed for each camera. Four instances of "License plate detection and reading" are running.* 7. *All components of "Management APP" are running.*
Steps
Step 1 **Action** *Generate traffic over the active cameras in the industrial area* **Postconditions** *"Vehicle detection and cropping" process generates "Vehicle identification jobs" that are received by a "License plate detection and reading" instance and processed and detection events are generated. These detection events are received by the Management APP that stores them on the database.*
Expected Result
• *All the information generated is correctly sent across the Pub/Sub mechanism and received by "License plate detection and reading" and the Management APP.* • *Events are finally stored in the database*

ID	**Title**
UC2.1_F_04_TC1	*Deploy camera "Vehicle detection and cropping" processes to the closest worker node.*
Objective	
• *NebulOuS deploy the "Vehicle detection and cropping" processes to the closest node*	
Preconditions	
1. *Two worker nodes are active.* 2. *A camera is streaming video.* 3. *NebulOuS is running and the worker nodes are connected to it.*	
Steps	
Step 1 **Action** *Instruct NebulOuS to deploy the Video stream processing APP (without forcing a worker node selection).*	
Expected Result	

• *NebulOuS deploys the "Vehicle detection and cropping" process for the video stream to the closest node.*

ID	**Title**
UC2.1_F_05_TC1	*Relocate "Vehicle detection and cropping" processes depending on performance*
Objective	
• *Test if NebulOuS reacts to violation of QoS requirement thresholds and re-configures the deployment to mitigate it.*	
Preconditions	
1. *Three cameras are deployed (C1, C2, C3)* 2. *Two edge worker nodes are available to NebulOuS (W1, W2)* 3. *NebulOuS is up and running and is connected to the worker nodes.* 4. *Video stream processing APP is correctly configured in NebulOuS.* 5. *"Vehicle detection and cropping" for the cameras C1 and C2 are deployed in W1. "Vehicle detection and cropping" for C3 is deployed in W2.*	
Steps	
Step 1 **Action** *Stress W1 to the point that it can no longer process both streams (C1, C2) at 10fps. (Use "stress" command line utility or similar).* **Postconditions** *Camera's video is processed at less than 10fps*	
Expected Result	
• *NebulOuS detects that the video streams are not being processed at less than 10fps.* • *NebulOuS detects that there is a worker node (W2) with lower CPU usage.* • *NebulOuS moves the "Vehicle detection and cropping" job for any of the cameras (C1 or C2) from W1 to W2.*	

ID	**Title**
UC2.1_NF_03_TC1	*Prevent Non-Admin User from Defining New Devices for Mercabarna Intralogistics UC*
Objective	
• *Verify that NebulOuS prevents a non-admin user from creating new resources for the Mercabarna Intralogistics UC project.*	
Preconditions	
1. *A non admin Nebulous user is already created.* 2. *The Mercabarna Intralogistics UC project is already created.*	
Steps	
Step 1	

Action *Log in with the non admin user.* **Postconditions** *The user should be successfully logged in and see the user dashboard.* **Step 2** **Action** *Create any kind of resource for Mercabarna Intralogistics UC project.* **Postconditions** -
Expected Result
• *The non-admin user should remain logged in, but the attempt to create a resource should fail with an appropriate error message.*

ID	**Title**
UC2.1_NF_05_TC1	*Prevent Unauthorized Subscription to the NebulOuS IoT pub/sub broker*
Objective	
• *Verify that NebulOuS can detect and block a subscription to the NebulOuS IoT pub/sub broker initiated from outside the NebulOuS environment, even with correct credentials.*	
Preconditions	
1. *The application is deployed.* 2. *The License Plate Detection and Reading module is deployed*	
Steps	
Step 1 **Action** *Create a constant data flow in the "License Plate Detection and Reading" module* **Postconditions** *Messages are being passed between application components using the NebulOuS IoT pub/sub.* **Step 2** **Action** *Using an MQTT client from outside NebulOuS, attempt to connect to the NebulOuS IoT pub/sub broker with the correct credentials.* **Postconditions** -	
Expected Result	
• *Even with the correct credentials the connection cannot be stablished.*	

ID	**Title**
UC2.1_NF_09_TC1	*NebulOuS is capable of store all the messages generated by the License Plate Detection and Reading module without losing any of the messages*
Objective	
• *Verify that NebulOuS can manage and store a large volume of messages sent in a short period using the NebulOuS IoT pub/sub broker.*	
Preconditions	
1. *Application is deployed.* 2. *NebulOuS IoT pub/sub broker is deployed in a cloud resource.*	
Steps	
Step 1 **Action** *Create a constant data flow of 1000 messages per second to the " NebulOuS IoT pub/sub broker " for 1 minute. Each message should contain an ID with a numerical order number reflecting the sequence of messages sent.* **Postconditions** -	
Expected Result	
• *All messages should be stored correctly, and the numerical sequence of IDs should be intact, with no missing messages.*	

ID	**Title**
UC2.1_NF_10_TC1	*Scale up-down "License plate detection and reading" processes depending on performance*
Objective	
• *Test if NebulOuS reacts to violation of QoS requirement thresholds and re-configures the deployment of "License plate detection and reading" workers to mitigate it.*	
Preconditions	
1. *One camera is deployed (C1)* 2. *A worker node is available to NebulOuS (W1)* 3. *A cloud provider is registered in NebulOuS.* 4. *NebulOuS is up and running and is connected to the worker nodes.* 5. *Video stream processing APP is correctly configured in NebulOuS.* 6. *An instance of "Vehicle detection and cropping" for camera (C1) is running on W1. An instance of "License plate detection and reading" is deployed on W1.* 7. *The "Vehicle identification jobs" generated by "Vehicle detection and cropping" process are completed in less than 10 seconds.*	
Steps	
Step 1	

Action

Stress W1 to the point that it can no longer process "Vehicle identification jobs" in less than 10 seconds. (Use "stress" command line utility or similar).

Postconditions

"Vehicle identification jobs" are processed in more than 10 seconds.

Expected Result

- *NebulOuS concludes that more resources are needed for deploying the application.*
- *NebulOuS instantiates a new worker node (W2) in the cloud provider.*
- *NebulOuS moves the instance of "License plate detection and reading" from W1 to W2 or deploys a new instance in W2.*
- *The time elapsed since the postcondition of step 1 is met and the app is re-configured*

ID	Title
UC2.1_NF_11_TC1	*Protect data transferred from "Vehicle Detection and Cropping module" placed on edge, to NebulOuS IoT pub/sub broker deployed on cloud.*

Objective

- *Verify a subscription to the NebulOuS IoT pub/sub broker initiated from outside the application cluster is not possible even with correct credentials.*

Preconditions

1. *The application is deployed.*
2. *The Vehicle Detection and Cropping module is deployed in an edge device.*
3. *The NebulOuS IoT pub/sub broker is deployed in a cloud resource.*

Steps

Step 1

Action

Create a constant data flow of messages from "Vehicle Detection and Cropping module" to "NebulOuS IoT pub/sub broker"

Postconditions

The NebulOuS IoT pub/sub broker is receiving messages periodically

Step 2

Action

From outside NebulOuS, using a MQTT broker client, attempt to connect to the module "NebulOuS IoT pub/sub broker " with the correct credentials.

Postconditions

-

Expected Result

- *The connection from the external client should be blocked, ensuring the integrity and security of the NebulOuS environment.*

Mercabarna Last Mile use case

ID	**Title**
UC2.2_F_01_TC1	*Only use Cloud resources when necessary*
Objective	
• *Assure that NebulOuS utilizes cloud resources only in saturation scenarios.*	
Preconditions	
1. *Necessary User Edge nodes for running the application under low workload are active.* 2. *NebulOuS is running and is connected to the User Edge nodes.* 3. *Cloud provider is configured in NebulOuS.* 4. *The application is deployed only using User Edge nodes.*	
Steps	
Step 1 **Action** *Manually generate data for some vehicles (GPS position and load orders updates). This will trigger the creation of delivery plan manager (worker) jobs that will query the routing engine and execute route optimization jobs.* *Ensure that the generated workload can be handled by the User Edge resources (maintain the same workload for 10 minutes and check that the component SLAs are met).* **Postconditions** *Application is running as expected using only User Edge resources.* **Step 2** **Action** *Saturate the User Edge resources until the point that SLAs are no longer met (Increase the number of vehicles simulated).* **Postconditions** *SLAs are no longer met.*	
Expected Result	
• *NebulOuS re-adapts the application, making use of Cloud resources, to cope with the new workload.*	

ID	**Title**
UC2.2_F_05_TC1	*Map server load balancing*
Objective	
• *Ensure that NebulOuS scales in/out – up/down the map server to guarantee that (on average) a request does not take more than 5 seconds to be processed.*	
Preconditions	
1. *Two User Edge nodes exist.* 2. *NebulOuS is running and is connected to the User Edge nodes.* 3. *A subset of the application with only the map server is deployed using only one of the nodes.*	
Steps	

Step 1 **Action** *Manually generate requests for the maps server. Ensure that the generated workload can be handled by a single node (maintain the same workload for 10 minutes and check that the component SLAs are met).* **Postconditions** *Application is running as expected using only one node.* **Step 2** **Action** *Increase the number of requests to the map server.* **Postconditions** *SLAs are no longer met.*
Expected Result
• *NebulOuS re-adapts the application, scaling up or out the map server component.*

ID	**Title**
UC2.2_F_09_TC1	*Test data transmission time*
Objective	
• *Test that GPS updates are available for being consumed by the delivery plan manager (worker) instance in less than 3s.*	
Preconditions	
1. At least one worker node is active. *2. NebulOuS has a successful connection to the node.* *3. A process that generates fake vehicle (V1) updates each second exists.* *4. The delivery plan manager (worker) for V1 is being executed.*	
Steps	
None	
Expected Result	
• *The vehicle updates are received by the corresponding delivery plan manager (worker).* • *The reception time is not bigger than the generation time + 3s (as seen in the logs of the delivery plan manager (worker)).*	

ID	**Title**
UC2.2_F_10_TC1	*Delivery plan manager (host) load balancing*
Objective	
• *Ensure that several instances of delivery plan manager (host) can exist, and the workload (vehicles to monitor) divided among them.*	
Preconditions	
1. An adapted version of the application is deployed using NebulOuS. This adaptation defines a fixed number of "Delivery plan manager (host)" instances.	

Steps
Step 1 **Action** *Manually generate data for 10 vehicles (GPS position and load orders updates).* **Postconditions** -
Expected Result
• *The created Delivery plan manager (host) instances distribute the workload, hosting several Delivery plan manager (worker) instances each.*

ID	**Title**
UC2.2_F_11_TC1	*Route optimization as serverless function*
Objective	
• *The route optimization logic implemented as a serverless function can be executed by NebulOuS*	
Preconditions	
1. *A worker node is active.* 2. *NebulOuS has a successful connection to the node.* 3. *The source code for the route optimization logic is implemented as a serverless function and is uploaded in NebulOuS.* 4. *The serverless framework for executing instances of the serverless function in the worker node is up and running.*	
Steps	
Step 1 **Action** *Manually trigger the execution of the route optimization logic providing a dummy payload.* **Postconditions** *An instance of the route optimization logic executes as a serverless function in the worker node.*	
Expected Result	
• *After the completion of the route optimization process as a serverless function, the optimized route is available for being consumed.*	

ID	**Title**
UC2.2_NF_03_TC1	*The platform provides secure access to the resources*
Objective	
• *Validate that the platform provides secure access to the resources (clouds, edge resources etc.) (i.e. provides user Identification/Authentication, User/application manager roles etc.)*	
Preconditions	
1. *NebulOuS is deployed* 2. *A user with resource administrations permissions on NebulOuS exists (admin)*	

3. *A user without resource administration permissions exists. (editor)* 4. *An edge node is available to be registered.*
Steps
Step 1 **Action** *Login in the UI with admin user* **Postconditions** *User is logged in as admin* **Step 2** **Action** *Add a cloud account* **Postconditions** *Cloud account is registered* **Step 3** **Action** *Register an edge node* **Postconditions** *Edge node is correctly registered* **Step 4** **Action** *Login in the UI with editor user* **Postconditions** *User is logged in as editor* **Step 2** **Action** *Try to add a cloud account* **Postconditions** *Process can't be completed since user doesn't have permissions* **Step 3** **Action** *Try to add an edge node* **Postconditions** *Process can't be completed since user doesn't have permissions*

ID	**Title**
UC2.2_NF_05_TC1	*Secure NebulOuS IoT pub/sub mechanism*
Objective	

• *Verify that connecting with the IoT pub/sub mechanism should not be possible without appropriate credentials.*
Preconditions
1. *NebulOuS is running and a Cloud provider is registered.* 2. *The application is deployed and is not monitoring any delivery route.*
Steps
Step 1 **Action** *Try to connect to the exposed port of the NebulOuS IoT pub/sub broker using **invalid** credentials* **Postconditions** *Connection fails* **Step 2** **Action** *Try to connect to the exposed port of the NebulOuS IoT pub/sub broker using **valid** credentials* **Postconditions** *Connection succeeds*

ID	**Title**
UC2.2_NF_09_TC1	*IoT pub/sub system handles high volumes of data*
Objective	
• *Validate that APP PUB/SUB system can handle the distribution of data for 5000 vehicles with an update of their GPS position every 5 seconds and their load order status every 10 minutes*	
Preconditions	
1. *NebulOuS is running and a Cloud provider is registered.* 2. *The application is deployed and is not monitoring any delivery route.*	
Steps	
Step 1 **Action** *Start a process that generates data for 5000 vehicles (GPS position update each 5 seconds and load order status update each 10 minutes). Deploy a dummy consumer of the data subscribed to the NebulOuS IoT PUB/SUB system (reading from NebulOuS IoT pub/sub and printing the reception date). Publish data for 1h*	
Expected Result	
• *The difference between the time of the event generation and the time of the event consumption is not bigger than 3 seconds*	

ID	**Title**
UC2.2_NF_11_TC1	*Data transferred between application components using NebulOuS IoT pub/sub is encrypted.*
Objective	
• *Validate that data transferred using NebulOuS IoT PUB/SUB system is encrypted.*	
Preconditions	
1. *NebulOuS is running and a Cloud provider is registered.* 2. *The application is deployed and is not monitoring a delivery route.*	
Steps	
Step 1 **Action** *Using WireShark or similar software to listen to the communications between the data integration layer and delivery plan manager through the NebulOuS IoT pub/sub mechanism.*	
Expected Result	
• *Data transferred is encrypted.*	

ID	**Title**
KPI_U2.2_1_TC1	*Detect delays*
Objective	
• *Detect delays in delivery route and trigger a recalculation in less than 2 minutes.*	
Preconditions	
1. *NebulOuS is running and a Cloud provider is registered.* 2. *The application is deployed and is not monitoring any delivery route.*	
Steps	
Step 1 **Action** *Manually generate a new load order for a vehicle with only one delivery with status* "STARTED"*.* *This will force the creation of a delivery plan manager (worker) for the vehicle and execute route optimization.* *Start a process that generates GPS updates for the vehicle. This will keep the delivery plan manager (worker) running. The GPS coordinates shall make the delivery plan manager (worker) understand that the route is being fulfilled.* **Postconditions** *Application is running and resources are allocated.* *Delivery plan manager (worker) doesn't detect any anomaly that requires a recalculation of the route.* **Step 2** **Action**	

Stop the GPS data generator. **Postconditions** *Delivery plan manager (worker) concludes that the route can no longer be met and creates a route optimization request.*
Expected Result
• *A new route is obtained.*

Augmenta use case

ID	**Title**
UC3_F_01_TC1	*NebulOuS adapts resources depending on field size*
Objective	
• *Test if NebulOuS selects appropriate node for allocating the map generation step of the workflow based on the size of the map to be generated.*	
Preconditions	
1. *NebulOuS is up and running.* 2. *An Edge node with limited resources is registered.* 3. *A Cloud provider registered.* 4. *The precision agriculture is deployed in NebulOuS. No work is being carried by the application.*	
Steps	

Step 1
Action
Manually submit a workflow execution request for a ***small*** *map.*
Postconditions
NebulOuS utilizes the Edge node with limited resources to process the workflow. *The workflow is completed.*

Step 2
Action
Manually submit a workflow execution request for a ***big*** *map.*
Postconditions
1. *NebulOuS allocates a new node with greater processing capacity in the Cloud provider to process the workflow.*

Expected Result
• *NebulOuS allocates computing resources with a different capacity depending on the job size (ha of the field).*

ID	**Title**

UC3_F_01_TC2	*NebulOuS utilizes Service Provider Edge and User Edge resources whenever possible*
Objective	
• *Test if NebulOuS utilizes Edge resources whenever the situation permits it.*	
Preconditions	
1. *NebulOuS is up and running.* 2. *A Cloud provider registered.* 3. *An Edge device is registered.* 4. *The precision agriculture is deployed in NebulOuS. No work is being carried by the application.*	
Steps	
Step 1 **Action** *Manually submit a workflow execution request for a* ***small*** *map.* **Postconditions** *NebulOuS uses the Edge node to process the request.* *The request is completed.* **Step 2** **Action** *Manually submit two workflow execution requests for a* ***small*** *map.* **Postconditions** 1. *NebulOuS uses the Edge node to process one request. NebulOuS allocates a node with necessary resources in the Cloud to process the remaining request.* 2. *Both jobs are processed in parallel.*	
Expected Result	
• *When Edge resource is free, NebulOuS allocates computation on it.* • *When it is occupied, NebulOuS offloads some computation to the Cloud.*	

ID	**Title**
UC3_F_03_TC1	*Ensure the graphic user interface (GUI) provides functionality for administration of deployed components on the NebulOuS platform.*
Objective	
• *Test if the NebulOuS platform provides a user-friendly GUI that allows users to adjust optimization criteria, manage computational resources, and monitor deployed components.*	
Preconditions	
1. *NebulOuS platform is up and running.* 2. *User account with necessary permissions to register providers and use case applications is available.*	
Steps	

Step 1
Action
Log in to the NebulOuS platform GUI using a registered user account.
Postconditions
User is successfully logged in and navigated to the main dashboard.

Step 2
Action
1. *Navigate to the section for registering providers and use case applications.* 2. *Register a new provider and use case application by uploading the required KubeVela file, metric model, SLO criteria, and optimization function.* 3. *Navigate to the optimization criteria section in the GUI.* 4. *Adjust the optimization criteria and computational resources as needed.*
Postconditions
New provider and use case application are successfully registered in the system and system saves the new optimization criteria and computational resource settings

Expected Result
• *Successfully register providers and use case applications.* • *Adjust optimization criteria and computational resources.* • *Monitor the deployed components in real-time.*

ID	**Title**
UC3_F_05_TC1	*NebulOuS re-schedules workflows to keep SLO.*
Objective	
• *Test if NebulOuS detects that a workflow execution is taking longer than expected and re-schedule any other workflow execution that might be waiting on the resources used by the first one.*	
Preconditions	
1. *NebulOuS is up and running.* 2. *A cloud provider registered.* 3. *The precision agriculture is deployed in NebulOuS. No work is being carried by the application.*	
Steps	

Step 1
Action
Manually submit a workflow execution request for a ***big*** *map but provide a duration estimation that is clearly lower than the expected duration.*
Postconditions
1. *NebulOuS allocates a computing node in the Cloud to process the workflow.*

Step 2
Action
Manually submit a new workflow execution request moments before the first workflow execution request was supposed to be completed according to the provided duration estimation.
Postconditions
1. *Initially, NebulOuS plans to execute the new request using the resources utilized by the first request (waiting for its imminent completion), after some time, it detects that the initial workflow is taking longer than expected and decides to allocate a new computing node to execute the new request.*

Expected Result
• *NebulOuS reacts to a workflow execution taking longer than expected by allocating more resources to execute pending workflows.*

ID	**Title**
UC3_F_12_TC1	*NebulOuS utilizes reliable Cloud providers*
Objective	
• *Test if NebulOuS takes into account the use of Cloud providers that are more reliable.*	
Preconditions	
NebulOuS is up and running. 1. *Two Cloud providers are registered (Cloud A and Cloud B).* 2. *User has specified that Cloud A is much more reliable than Cloud B via the resource brokerage UI.* 3. *The precision agriculture is deployed in NebulOuS. No work is being carried by the application.*	
Steps	

Step 1
Action
Manually submit a workflow execution request.
Postconditions
1. *NebulOuS allocates a node in Cloud A to execute the workflow.* 2. *The workflow finalizes.*

Step 2
Action
Modify the ranking criteria for Cloud providers. Now Cloud B is considered to be much more reliable.
Postconditions
-

Step 3
Action
Manually submit a workflow execution request.
Postconditions
1. *NebulOuS allocates a node in Cloud B to execute the workflow.* 2. *The workflow finalizes.*

Expected Result
• *NebulOuS takes into account the user preferences when selecting the Cloud provider.*

ID	**Title**
UC3_F_13_TC1	*NebulOuS resumes workflows*
Objective	
• *Test if NebulOuS re-executes flow steps whenever the machine running the flow step fails.*	
Preconditions	
1. *NebulOuS is up and running.* 2. *A cloud provider registered.* 3. *The precision agriculture is deployed in NebulOuS. No work is being carried by the application.*	
Steps	

Step 1
Action
Manually submit a workflow execution request.
Postconditions
NebulOuS starts the execution of the workflow.

Step 2
Action
Manually kill the node executing the workflow step.
Postconditions
NebulOuS should detect that the node executing the workflow step has stopped, allocate a new computing node and execute the workflow step on that node.

Expected Result
• *When a node executing a workflow step fails, NebulOuS finds a new node to deploy the workflow step and re-submits the job.*

ID	**Title**
UC3_NF_02_TC1	*NebulOuS continues operation on limited connection*
Objective	

• *Test if NebulOuS can continue to utilize Edge resources even when the connection with the NebulOuS core running on cloud is poor.*
Preconditions
1. *NebulOuS is up and running.* 2. *Two Edge nodes are registered.* 3. *The precision agriculture is deployed in NebulOuS. No work is being carried by the application.*
Steps

Step 1
Action
Manually submit a workflow execution request for a ***small*** *map.*
Postconditions
NebulOuS starts the execution of the workflow using the Edge node.

Step 2
Action
While the execution of the workflow is ongoing in the Edge node, simulate connectivity problems of the Edge node (packet loss, intermittent dis-connections for no longer than 2 seconds) with the NebulOuS core running on Cloud.
Postconditions
The execution of the workflow is continued.

Expected Result
• *NebulOuS doesn't decide to terminate the execution of the workflow in the Edge node.*

ID	**Title**
UC3_NF_03_TC1	*Ensure secure access for onboarding new edge devices on the NebulOuS platform.*
Objective	
• *Test if only registered users with necessary permissions can onboard new edge devices for application deployment.*	
Preconditions	
1. *NebulOuS platform is up and running.* 2. *User accounts with different roles (e.g., registered user with permissions, and a user without permissions) are set up.* 3. *The necessary credentials for both user types are available.*	
Steps	

Step 1
Action
Attempt to onboard a new edge device using the credentials of a user without the necessary permissions.
Postconditions
NebulOuS platform denies the attempt to onboard the new edge device.

Step 2
Action
Attempt to onboard a new edge device using the credentials of a registered user with the necessary permissions.
Postconditions
NebulOuS platform allows the onboarding of the new edge device successfully.

Expected Result
• *Only registered users with the necessary permissions can onboard new edge devices to the NebulOuS platform.*   • *Users without the necessary permissions are denied access to onboard new edge devices.*

ID	**Title**
UC3_NF_10_TC1	*NebulOuS adapts to the variations of workload.*
Objective	
• *Test if NebulOuS allocates more computing resources when the number of workflow execution requests grows. Test if NebulOuS de-allocates the resources once they are no longer necessary.*	
Preconditions	
1. *NebulOuS is up and running.* 2. *A Cloud provider is registered.* 3. *The precision agriculture is deployed in NebulOuS. No work is being carried by the application.*	
Steps	

Step 1
Action
Submit 5 workflow processing requests per minute for 30 minutes. Each request has an expected duration of 2 minutes.
Postconditions
NebulOuS allocates necessary computing resources in the Cloud provider to cope with the workload. After 15 minutes, the number of allocated resources stabilizes.

Step 2
Action

After 30 minutes, submit 10 workflow processing requests per minute for 30 minutes. Each request has an expected duration of 2 minutes.
Postconditions
NebulOuS increases the amount of allocated resources in the Cloud provider to cope with the workload. After 15 minutes, the number of allocated resources stabilizes.

Step 3
Action
After 30 minutes, submit 1 workflow processing requests per minute for 30 minutes. Each request has an expected duration of 2 minutes.
Postconditions
NebulOuS decreases the amount of allocated resources in the Cloud provider to cope with the workload. After 15 minutes, the number of allocated resources stabilizes.
Expected Result
• *NebulOuS reacts to the changes in the workload and allocates/de-allocates resources accordingly.*

ID	**Title**
UC3_NF_11_TC1	*Ensure data privacy during data transfer on the NebulOuS platform.*
Objective	
• *Test if the information exchanged as part of the workflow execution is encrypted to ensure data privacy.*	
Preconditions	
1. *NebulOuS platform is up and running.* 2. *A workflow execution request process is prepared and ready for testing.* 3. *WireShark or similar network protocol analysis software is installed and configured to capture network traffic.*	
Steps	

Step 1
Action
Start WireShark or similar software and set it to capture traffic on the network interface used by NebulOuS. *Initiate a workflow execution request on the NebulOuS platform.*
Postconditions
WireShark captures network traffic for the workflow execution request.

Step 2
Action
Analyze the captured network traffic using WireShark. *Inspect the packets related to the workflow execution request to determine if the data is encrypted.*
Postconditions
All information exchanged as part of the workflow execution request appears encrypted in the network traffic.

Expected Result
• *The information exchanged during the workflow execution request is encrypted, ensuring data privacy during transfer.*

Crisis Management- @FIRE use case

ID	**Title**
UC4_F_01_TC1	*NebulOuS reacts to component failure*
Objective	
• *Test if NebulOuS registers a component failure / drop out and redeploys the component on the remaining infrastructure.*	
Preconditions	
1. *Two or more worker nodes are active.* 2. *NebulOuS is running and is connected to the worker nodes.* 3. *Application components are up and running.*	
Steps	
Step 1 **Action** *Manually kill (e.g.: power off) one of the nodes that is executing application components* **Postconditions** *NebulOuS detects that some components of the application have disappeared*	
Expected Result	
• *NebulOuS redeploys missing application components in a different available device/Cloud VM.*	

ID	**Title**
UC4_F_02_TC1	*Inter-Component Communication*
Objective	
• *NebulOuS can provide a correct functioning of the NebulOuS IoT pub/sub mechanism for the application components*	
Preconditions	
1. *Two or more worker nodes are active.* 2. *NebulOuS is running and is connected to the worker nodes.* 3. *Application components are correctly configured to use the NebulOuS IoT pub/sub mechanism.*	
Steps	
Step 1 **Action** *Generate sensor data and publish to the NebulOuS IoT pub/sub mechanism* **Postconditions** *Application components receive and process sensor data*	
Expected Result	
• *All sensor data generated is correctly sent across the NebulOuS IoT pub/sub mechanism and received by the appropriate application components*	

ID	**Title**
UC4_F_03_TC1	*NebulOuS Optimization Customization*
Objective	
• *NebulOuS provides a GUI which enables the user to change the optimization criteria*	
Preconditions	
1. *Two or more worker nodes* are active. 2. *NebulOuS is running and is connected to the worker nodes.* 3. *The NebulOuS GUI is available.* 4. *The application is deployed and running*	
Steps	
Step 1 **Action** *User changes the optimization criteria drastically via the GUI.* **Postconditions** *NebulOuS detects that the current deployment is sub-optimal.*	
Expected Result	
• *NebulOuS detects that the current deployment is sub-optimal and redeploys the application in optimized configuration.*	

ID	**Title**
UC4_F_04_TC1	*Deploy Application Component to specific worker node*

Objective
• *NebulOuS shall allow to deploy an application component to a specific worker node.*
Preconditions
1. *Two worker nodes are active (node A and B).* 2. *NebulOuS is running and is connected to the worker nodes.* 3. *One application component (C1) is configured to be deployed on node A.*
Steps
Step 1 **Action** *Instruct NebulOuS to deploy the* application **Postcondition** *NebulOuS deploys the application with component C1 deployed to worker node A.* **Step 2** **Action** *Remove Node A.* **Postcondition** *NebulOuS can no longer deploy the application as Node A is not available.*
Expected Result
• *NebulOuS honours the node placement restrictions defined in the KubeVela file.*

ID	**Title**
UC4_F_04_TC2	*Deploy Application Component to worker node only if power availability is sufficient*
Objective	
• *NebulOuS shall allow to deploy an application component to a specific worker node only if the power availability of the node is sufficient.*	
Preconditions	
1. *Two worker nodes are active (node A and B).* 2. *NebulOuS is running and is connected to the worker nodes.* 3. *One application component (C1) is configured to be deployed only on worker node A and only if power availability is sufficient.* 4. *Power availability for node A is sufficient to deploy component C1.*	
Steps	
Step 1 **Action** *Instruct NebulOuS to deploy the* application **Postconditions** *NebulOuS deploys the application, instantiating the target component C1 in node A.* **Step 2**	

Action *Alter the readings of power availability provided by node A to signal insufficient power availability.* **Postconditions** *NebulOuS detects that the power availability is insufficient and re-configures the application.* *The target component C1 is removed from node A.* *The target component C1 is no longer deployed.*
Expected Result
• *NebulOuS deploys a power-constrained component only when power availability is sufficient.*

ID	**Title**
UC4_F_14_TC1	*System is started after minimal training provided.*
Objective	
• *The Non-IT Personal (newly trained users) should be able to deploy the application using NebulOuS.*	
Preconditions	
1. *Crew members from FIRE selected* 2. *Minimal training Crash course (ca. 4 hours) for selected crew* 3. *NebulOuS is up and running and is connected to the worker nodes.* 4. *Fire use case application is correctly configured in NebulOuS*	
Steps	
Step 1 **Action** *The test personnel start the deployment through Nebulous and operate the system independently.*	
Expected Result	
• *The deployment of the application through NebulOuS is achieved without any support or issues.*	

ID	**Title**
UC4_NF_01_TC1	*Relocate "Persistence Component" depending on performance*
Objective	
• *Test if NebulOuS reacts to violation of SLO and re-configures the deployment to mitigate it.*	
Preconditions	
1. *Sensor data is being generated from 10 sensors with 1 Hz frequency* 2. *Two Edge worker nodes are available to NebulOuS* 3. *NebulOuS is up and running and is connected to the worker nodes.* 4. *Fire use case app and persistence component is correctly configured in NebulOuS.*	
Steps	
Step 1	

Action *Stress application by increasing number of sensors to 2500 with 1 Hz frequency* **Postconditions** *Persistence application is failing to process data stream within 10 sec.*
Expected Result
• *NebulOuS detects that the data streams are not being processed within at most 10 sec.* • *NebulOuS detects that there is a worker node with lower CPU usage.* • *NebulOuS scales the persistence component to the second worker node.*

ID	**Title**
UC4_NF_02_TC1	*Redeploy Application Component after Cloud connection loss*
Objective	
• *NebulOuS shall redeploy an application component to an Edge worker node once cloud connection is lost.*	
Preconditions	
1. *One or more Cloud worker* nodes *are active.* 2. *One or more Edge worker nodes are active.* 3. *NebulOuS is running in the Edge and is connected to the worker nodes.* 4. *One or more application components are deployed to the Cloud.*	
Steps	
Step 1 **Action** *Cut the connection between Edge and Cloud. Cloud is no longer reachable by NebulOuS running on the Edge node.*	
Expected Result	
• *NebulOuS redeploys the application with all components deployed on the available Service Provider Edge worker nodes.*	

ID	**Title**
UC4_NF_03_TC1	*The platform provides secure access to the resources*
Objective	
• *Validate that the platform provides secure access to the resources (clouds, edge resources etc.) (i.e. provides user Identification/Authentication, User/application manager roles etc.)*	
Preconditions	
1. *NebulOuS is deployed* 2. *A user with administrative permissions on NebulOuS exists (admin)* 3. *A second user without administrative privileges exists (user)*	
Steps	
Step 1 **Action**	

The user without admin privileges tries to log in to the Admin GUI.
Expected Result
• *The access to the admin GUI is rejected*

UC4_NF_07_TC1	*Redeploy Application Component after Edge worker node connection loss*
Objective	
• *NebulOuS shall redeploy an application component to the remaining Edge worker nodes once connection to one worker node is lost.*	
Preconditions	
1. *One or more worker nodes are active.* 2. *NebulOuS is running and is connected to the worker nodes.* 3. *Two or more application components are deployed to the worker nodes*	
Steps	
Step 1 **Action** *Disconnect one worker node from Edge network*	
Expected Result	
• *NebulOuS redeploys the application with all components deployed on the available worker nodes.*	

UC4_NF_10_TC1	*Redeploy Application Component after additional worker nodes are available*
Objective	
• *NebulOuS shall redeploy an application component once additional worker nodes are available.*	
Preconditions	
1. *One or more worker nodes are active.* 2. *NebulOuS is running and is connected to the worker nodes.* 3. *Two or more application components are deployed to the worker nodes but their SLO is being constantly violated. NebulOuS cannot find a better deployment given that there are no more available resources.*	
Steps	
Step 1 **Action** *Register additional worker nodes to the NebulOuS network*	
Expected Result	
• *NebulOuS redeploys the application across all available worker nodes.*	

CONSORTIUM

Funded by the European Union

NebulOuS

A META OPERATING SYSTEM FOR BROKERING
HYPER-DISTRIBUTED APPLICATIONS ON
CLOUD COMPUTING CONTINUUMS

Funded by
the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Directorate-General for Communications Networks, Content and Technology. Neither the European Union nor the granting authority can be held responsible for them.