Ref. Ares(2024)4508951 - 21/06/2024

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

D6.1
1st Release of the NebulOuS Integrated Platform and Use Case Planning

21/06/2024

Funded by
the European Union

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	36 months (starting date 1 September 2022)
Deliverable title	1st Release of the NebulOuS Integrated Platform and Use Case Planning
Deliverable number	D6.1
Deliverable version	1.1
Contractual date of delivery	29 February 2024
Actual date of delivery	20 March 2024
Revised version date of delivery	21 June 2024
Nature of deliverable	Other
Dissemination level	Public
Work Package	WP6
Deliverable lead	7bulls.com
Author(s)	Joanna Chmielewska, Jan Marchel, Radosław Piliszek (7bulls), Małgorzata Jakubczyk (TTA), Robert Sanfeliu Prat (EUT), Christos-Alexandros Sarros (UBI), Amir Azimian (FIRE), Sofia Rosas (UW), Ferran Diego Andilla (Telefonica), Vasilis Zaridis (AUG), Fotis Paraskevopoulos (EXZ), Grigoris Savvas (EXZ), Geir Horn (UiO)
Abstract	This document summarizes the work that was carried out in preparing the first release of NebulOuS platform and it provides the strategic approach to ensure that the chosen platform and technology supports the consortium goals. It is a report on the implementation of the integration layer, description of the technical design and testing strategy. The document also provides the requirements for use cases evaluation and validation.
Keywords	Integration, architecture, testing, use case planning, 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

CONTRIBUTORS

Name	Organization
Joanna Chmielewska	7bulls.com
Jan Marchel	7bulls.com
Radosław Piliszek	7bulls.com
Robert Sanfeliu Prat	EUT
Christos-Alexandros Sarros	UBITECH
Ferran Diego Andilla	TELEFONICA
Małgorzata Jakubczyk	TTA
Amir Azimian	FIRE
Sofia Rosas	UW
Vasilis Zaridis	AUG
Fotis Paraskevopoulos	EXZ
Grigoris Savvas	EXZ
Geir Horn	UiO

PEER REVIEWERS

Name	Organization
Andreas Tsagkaropoulos	ICCS
Moritz von Stietencron	BIBA

REVISION HISTORY

Version	Date	Owner	Author(s)	Comments
0.1	08/01/2024	7bulls	Joanna Chmielewska	First Draft
0.2	22/01/2024	TTA	Małgorzata Jakubczyk	Input for TTA use case
0.3	30/01/2024	FIRE	Amir Azimian	Input for FIRE use case
0.4	31/01/2024	EUT UBI	Robert Sanfeliu Prat Christos-Alexandros Sarros	Input for Evaluation framework, architecture, interactions
0.5	02/02/2024	7bulls UW	Jan Marchel Sofia Rosas	Input on Integration chapter and Ubiwhere use case
0.6	14/02/2024	Telefonica AUG	Ferran Diego Andilla Vasilis Zaridis	Addition of manual TC Input for AUG use case
0.7	19/02/2024	EXZ	Fotis Paraskevopoulos Grigoris Savvas	Middleware Integration
0.8	21/02/2024	ICCS	Andreas Tsagkaropoulos	First review
0.9	23/02/2024	BIBA	Moritz von Stietencron	Second Review
1.1	21/06/2024	7bulls	Multiple authors	Revised version

TABLE OF ABBREVIATIONS AND ACRONYMS

Abbreviation/Acronym	Open form
ABAC	Attribute-based access control
AI	Artificial Intelligence
AMPL	A Mathematical Programming Language
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
ARM	Advanced RISC Machines
AWS	Amazon Web Services
BQA	Brokerage Quality Assurance
BYON	Bring Your Own Node
CCTV	Closed-circuit television
CFSB	Cloud & Fog Service Brokerage
CI/CD	Continuous integration/ Continuous Delivery
CPU	Central Processing Unit
CRI	Container Runtime Interface
DAG	Directed Acyclic Graph
DB	Database
EMS	Event Management System
ESB	Enterprise Service Bus
EXN	Existanze Integration Middleware
GB	GigaByte
GCP	Google Cloud Platform
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
K8s	Kubernetes
KPI	Key Performance Indicator

Meta-OS	Meta-Operating System
MCDM	MultiCriteria Decision Making
MQTT	MQ Telemetry Transport
OAM	Open Application Model
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
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

LIST OF TABLES

EXECUTIVE SUMMARY

NebulOuS[1] is a Meta Operating System designed to facilitate secure and efficient application provisioning and reconfiguration within the Cloud Computing Continuum. Its objective is to decide the optimal deployment of each application, perform the deployment, monitor the execution and search for a new optimal deployment of the application in an event of QoS violation. NebulOuS offers facilities for resource discovery, allocation and provisioning, component placement and scheduling. It exploits the concepts introduced in the section Terminology, which are Service Provider Edge nodes and User Edge nodes, in combination with multi-cloud resources, to help address the challenges associated with achieving low-latency performance. NebulOuS is fully open-source software. It adheres to open standards and protocols, promoting interoperability and compatibility with other systems and platforms. This interoperability enables seamless integration with existing infrastructure and tools, reducing barriers to adoption and facilitating smoother transitions.

This document provides an architectural concept of the NebulOuS platform and shows the interactions between the components, along with the details and guidelines for integration process. Additionally, this deliverable presents an approach to software testing, as it is a crucial phase in the software development cycle that involves systematic evaluation to identify and rectify defects, ensuring software quality and reliability.

Apart from the integration work, this document presents an overview of the prepared evaluation framework with functional and non-functional requirements, and KPIs to analyse the performance and impact of NebulOuS on each chosen use case. The in-depth descriptions of the use-cases and the deeper requirement analysis offers the possibility to review the architectural schema.

[1] https://nebulouscloud.eu/

1. INTRODUCTION

1.1. THE SCOPE

This deliverable describes the 1st release of NebulOuS. To this end, it presents two levels of architecture design. The first, conceptual design, includes the logical components and is closely tied to the software artifacts that are going to be implemented. It describes the basic flow of information between the components. The second, grounded architecture design, is a description of the actual implementation of our system, including all technical details and is related to the results of the technical tasks that were carried out during the first stage of the project.

Furthermore, this document describes the integration goals to streamline the development lifecycle, ensure software quality and enable rapid and reliable delivery of updates, as well as adaptations needed to integrate underpinning OSS mechanisms into the NebulOuS platform. It reports on the integration strategy for NebulOuS, the prepared tools to support code quality assurance, guidelines for writing code, the code workflow and CI/CD. Furthermore, it contains the list of prepared test cases for the first release of NebulOuS, with their Id and short summary.

In the final part, each of the use case partners presents the brief descriptions of their case and the benefits expected from using NebulOuS platform. They provide the means of verifications for the requirements identified in WP2. This deliverable provides feedback on the use of NebulOuS by every use case.

1.2. RELATIONS TO OTHER DELIVERABLES AND WP'S

Deliverable 6.1 summarises the work carried out mostly in Work Package 2 *Requirements Analysis, Architecture and Modelling for Ad-Hoc Cloud Computing Continuum* and Work Package 6 *Integrated Meta-OS & Pilot Demonstrators*. It's closely related to the Deliverable 2.1 *Requirements and Conceptual Architecture of the NebulOuS Meta-OS*[2] which is a point of reference for the implementation and integration work. Deliverable 2.1 presents the general architecture of the platform and contains the descriptions of the components and basic interactions between them that are expanded in this document. The overview of the use cases, that was presented in D2.1, is now refocused on evaluation and validation requirements and on prepared test cases. Being an integration deliverable, all technical WPs and previous deliverables are naturally related to this one.

1.3. DELIVERABLE STRUCTURE

This deliverable presents the following information contained in the following chapters:

- *Chapter 2*, NebulOuS Platform Architecture, contains the description of the terminology, architecture, the interactions between the components and sequence diagrams for the 1st release

[2] https://nebulouscloud.eu/wp-content/uploads/2023/04/D2.1-Requirements-and-Conceptual-Architecture-of-the-NebulOuS-Meta-OS-v1.0.pdf

- *Chapter 3*, The Scope of the NebulOuS 1st release, presents key components and functionalities of the platform for the 1st release.
- *Chapter 4*, NebulOuS Platform Integration, reports on implementation of the integration methods and tools for software quality assurance
- *Chapter 5*, Testing methodology, presents the testing approach, the process of the test case creation and the list of prepared tests.
- *Chapter 6*, Evaluation framework, outlines the methodology for defining and assessing the requirements and KPIs and for analysing the performance and impact of NebulOuS on each use case.
- *Chapter 7*, Use case planning, presents six cases with requirements, KPIs and means to validate them.
- *Chapter 8*, Conclusions.
- *Annex*, Instruction of the Manual Installation of the NebulOuS Core, Topics in communication between components, Manual Test Cases, Automated Test Cases, Test cases provided by Use Cases

2. NEBULOUS PLATFORM ARCHITECTURE

2.1. TERMINOLOGY

To better pinpoint the Cloud-to-Edge Computing Continuum resources that NebulOuS can exploit for deploying and reconfiguring applications, we need first to agree on a common terminology. Therefore, from now on, we will be using part of the glossary defined by the Linux Foundation Projects, the State of the Edge[3]. NebulOuS focuses on the following types of continuum resources that are mainly differentiated according to the resources provider and the level of latency and data transport costs their use entails, with respect to application data sources' location (e.g. IoT). Specifically, the terms and definitions as mentioned in the State of the Edge glossary (and adjusted to the NebulOuS context) are:

- **Cloud**: the "traditional" part of the Computing Continuum that provides on-demand access to a shared pool of computing resources, including network, storage, and computation services. Typically utilizes a small number of large, centralized data centers and regional data centers.
- **Service Provider Edge**: refers to Fog computing capability with sub-100ms latencies support, which is deployed on the operator side of the last mile network. Compute, data storage and network resources that allow for Cloud-like capabilities similar to those found in centralized data centers, but with lower latency and lower data transport costs due to a higher degree of proximity to application data sources, than with a centralized or regional data center. Examples include Nvidia Jetson Xavier[4] devices, Next Units of Computing (NUCs[5]) or even laptops near the antenna that application data sources are connected to. We note that further granularity of this continuum layer could be used, if needed, with the subcategories such as Access Edge and Edge Node.

[3] https://github.com/State-of-the-Edge/glossary
[4] https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/
[5] https://www.intel.com/content/www/us/en/support/products/98414/intel-nuc.html#197864

- **User Edge**: refers to Edge computing capability which is deployed on the user side of the last mile network. User Edge resources are adjacent to end-users and processes in the physical world and encompass a wide range of equipment types, including gateways, servers, and end user devices. Workloads on the User Edge often work in conjunction with resources on the Service Provider Edge but can achieve lower latencies and conserve broadband network bandwidth by processing data without requiring it to pass across the last mile networks. Compared to the Service Provider Edge, the User Edge represents a highly diverse mix of resources. Examples include Raspberry Pi[6] devices, laptops or NUCs near the application data source near data sources, Field Analyzer, etc.

2.2. CONCEPTUAL ARCHITECTURE

The NebulOuS reference architecture has been defined in D2.1 "Requirements and Conceptual Architecture of the NebulOuS Meta-OS" and is presented in Figure 1. This Figure includes some minor updates (with respect to D2.1) to be better aligned with the current status of the project. In particular, user roles are now more granular while the application deployment flow has been slightly altered based on the actual implementation.

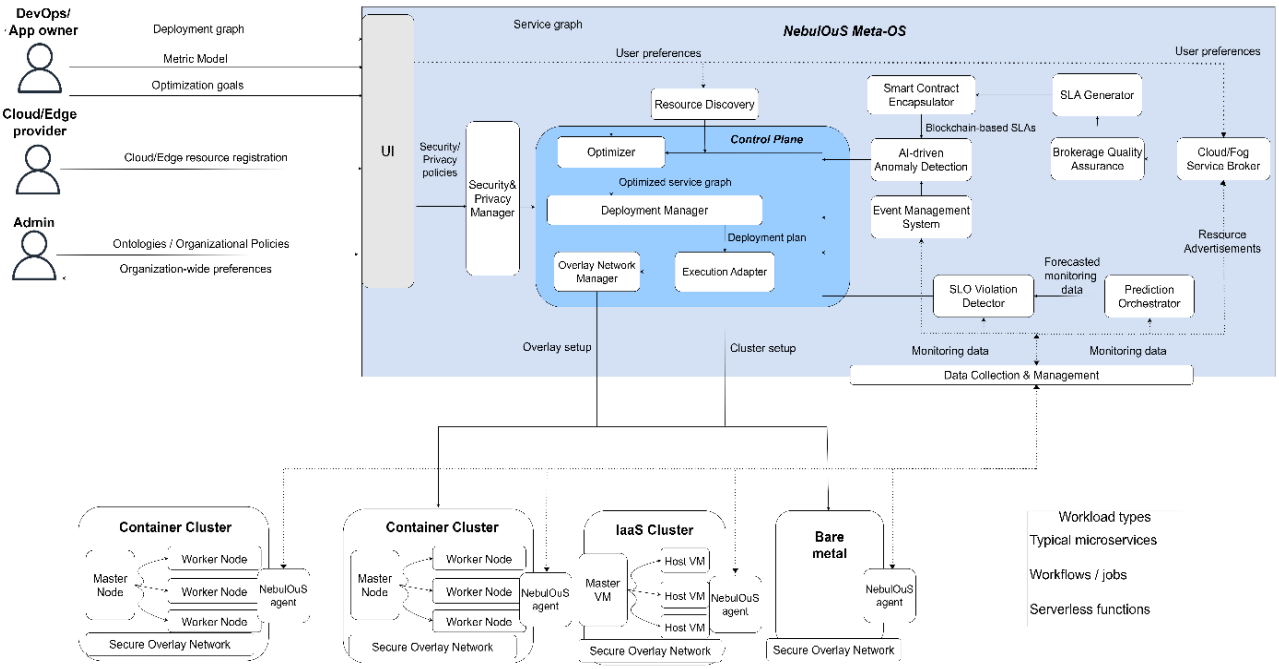


Figure 1. NebulOuS conceptual architecture.

The NebulOuS Meta-OS implementation can be considered as a PaaS for the Cloud-Edge Continuum. Its architecture follows a modular design, with discrete components implementing different functionality. The NebulOuS Meta-OS consists of:

[6] https://www.raspberrypi.com/

- **The User Interface:** This is the main point of entry for end users and system administrators to interact with the NebulOuS platform. It facilitates the definition and deployment of Cloud/Edge services along with their optimization requirements, along with security policies that can be enforced on the clusters. It includes a controller layer that exposes most of the logic as an API.
    - Repository:
        - https://opendev.org/nebulous/gui
        - https://opendev.org/nebulous/gui-controller
- **The Security & Privacy Manager:** It consumes user-defined security policies and deploys them in the NebulOuS-managed clusters. Currently supports access control policies, will additionally support network security policies in the next release.
    - Repository:
        - https://opendev.org/nebulous/security-manager
- **The Optimizer:** A typical NebulOuS application consists fundamentally of a set of containers pre-grouped into microservices. The role of the components collectively referred to as the Optimizer is: a) to decide on the placement of the microservices ranging from the core Cloud data centres to the Edge devices, b) to decide on the multiplicity of microservices and duplicate microservices that are bottlenecks in the application and c) to provide the application with more resources with the right capabilities in the right location to execute the application's microservices.
    - Repository:
        - https://opendev.org/nebulous/optimiser-controller
        - https://opendev.org/nebulous/optimiser-solver
        - https://opendev.org/nebulous/optimiser-utility-evaluator
- **The Deployment Manager**: The NebulOuS 'Deployment Manager' acts as an abstraction layer to handle the lifecycle of a cluster dedicated to a specific user application. It exposes the main REST endpoints used by other NebulOuS components to manage the cluster resources and communicates with the Execution Adapter that executes the required actions on the underlying infrastructure.
    - Repository:
        - https://github.com/ow2-proactive/scheduling-abstraction-layer
- **The Execution Adapter:** It is the NebulOuS "executionware". It provides on-demand access to computing resources such as servers, storage, and networking, offering more direct control over cloud-based systems. It enables to perform CRUD operations on different public or private clouds and offers REST endpoints to communicate with these infrastructure interfaces, to manage the virtual machines and Edge devices.
    - Repository:

- https://github.com/ow2-proactive/scheduling

- **The Overlay Network Manager:** The ONM automatically creates and manages a secure network overlay between the Cloud/Edge resources. This overlay takes the form of a VPN, which assumes two main functionalities: a) it provides connectivity between the NebulOuS compute resources (physical and/or virtual) and b) it secures the data in transit by encrypting them.
  - Repository:
    - https://opendev.org/nebulous/overlay-network-manager
- **The Data Collection & Management System**: The Data Collection and Management component is related to data management, both in the context of managing the monitoring data collected from the IoT/Edge/Cloud compute resources, as well as with respect to the data exchanged internally between the NebulOuS components.
  - Inter-component Communications Middleware repository:
    - https://opendev.org/nebulous/exn-middleware
    - https://opendev.org/nebulous/activemq
  - IoT data processing pipelines orchestration repository:
    - https://opendev.org/nebulous/iot-dpp-orchestrator
- **The Event Management System**: The role of EMS is to deploy and maintain the monitoring functionality of NebulOuS on infrastructure nodes, which will observe the necessary Quality of Service (QoS) metrics (both of the platform and of the deployed applications). Based on application description models, EMS will deploy and configure a number of monitoring agents that collect, filter, process and propagate the monitoring metrics values. The collected and computed monitoring metrics are then published in platform's pub/sub message broker hence any interested subscriber can use them. To allow NebulOuS to proactively respond to QoS variations, EMS will incorporate a metric predictor mechanism. This mechanism will allow to utilize a set of forecasting tools to anticipate future values of metrics collected by EMS.
  - EMS repository:
    - https://opendev.org/nebulous/monitoring
  - Prediction Orchestrator repository:
    - https://opendev.org/nebulous/prediction-orchestrator
- **The SLO Violation Detector (SLOViD)**: The role of SLOViD is to inform the NebulOuS platform about situations in which a reconfiguration is possibly needed, based on the predictions of forecasters, as well as information about one or more devices abandoning the processing topology.
  - SLOViD repository:

    - https://opendev.org/nebulous/slo-violation-detector

- **The Smart Contract Encapsulator**: The Smart Contract Encapsulator encapsulates SLAs in smart contract. It will provide a mechanism for infusing into smart contracts both the SLA rules and the monitoring algorithms used to determine whether these rules are satisfied by application execution. It will enable smart contracts to invoke other services and third parties for performing arbitrage when SLAs are violated, also encompassing a sensing/monitoring loop that regularly feeds the information into the smart contracts.
    - Repository:
        - Will be included in the 2nd release.
- **The Fog/Edge Resource Manager**: The component is the same tool mentioned in the D2.2 as the Resource Discovery mechanism. It's managing and monitoring Service Provider Edge and User Edge devices within the cloud computing continuum. The Resource Manager will execute appropriate scripts to detect/identify the capabilities of the device, and to install an appropriate monitoring agent for collecting health status data.
    - Repository:
        - https://opendev.org/nebulous/resource-manager
- **The Cloud/Fog Service Broker:** The Cloud/Fog Service Broker assists NebulOuS in coping with the massive number of resources that can be considered in Cloud Computing Continuum deployments. Its goal is to reduce the variability space when considering Cloud Computing Continuum resources for hosting application component instances.
    - Repository:
        - https://opendev.org/nebulous/cloud-fog-service-broker
- **The Brokerage Quality Assurance**: The BQA mechanism assures brokerage quality by ensuring abidance of application provisioning requirements and preferences with organisational policies: higher-level requirements that reflect an organisation's business and/or security standpoint with respect to application provisioning
    - Repository:
        - Will be included in the 2nd release.
- **The SLA Generator**: The SLA Generator is responsible for the automatic creation of SLA templates. It is based on the NebulOuS meta-ontology, which supports a) the description of entities and their properties in the cloud continuum ("resources") and b) the description of application provisioning requirements and preferences ("QoS").
    - Repository:
        - Will be included in the 2nd release.
- **The AI-driven Anomaly Detection**: The role of the AI-driven anomaly detection engine is to ensure QoS, being able to identify deviation from "normal" behaviour (i.e., low probability events, anomaly) of IoT devices.

- Repository:
  - Will be included in the 2nd release.

2.3. GROUNDED ARCHITECTURE

The above-mentioned components form the NebulOuS control plane, which is used to manage applications and services in the Cloud-Edge Continuum. During the project, the control plane services have been containerized and deployed in a compute cluster using the Kubernetes[7] container orchestration platform. The cluster hosting the NebulOuS control plane components will hereafter be referred to as the "control plane" cluster.

After installing the NebulOuS control plane, users can use the platform to deploy and manage containerized applications in Cloud-Edge Continuum resources. Those resources, once onboarded to NebulOuS, can be used by the Meta-OS to form compute clusters, on top of which applications are being deployed. Those clusters are referred to as the "application clusters". We use Kubernetes to orchestrate the containerized user applications, due to its open-source nature, rich ecosystem and wide industry adoption, which has made it the de-facto orchestrator for cloud-native containerized workloads.

The NebulOuS control plane components communicate asynchronously with each other, using an ActiveMQ[8] broker which acts as the communications middleware. AMQP messages are produced and consumed by components, using ActiveMQ topics that follow a certain naming convention. Details about the communication middleware is provided in Section 4.5, while the component interactions themselves are presented in the next subsection (Section 2.2).

A detailed description regarding the design of each Control Plane component can be found in D2.1, while their implementation status for the 1st release can be found in the Deliverables D3.1, D4.1 and D5.1.

[7] https://kubernetes.io/
[8] https://activemq.apache.org/

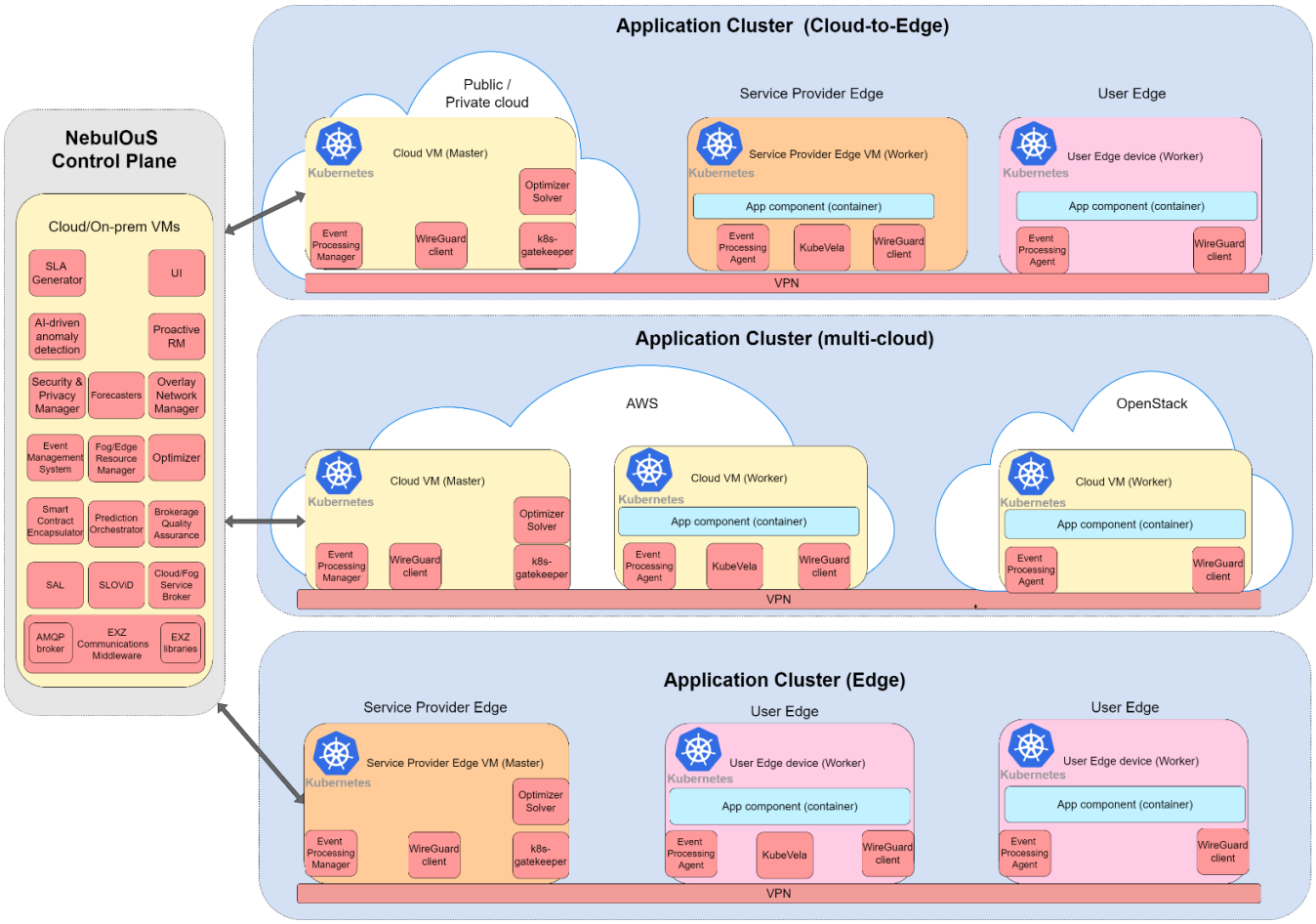


Figure 2. NebulOuS architecture (deployment view)

Next, we present a detailed view of the grounded NebulOuS architecture through three indicative application deployment scenarios (depicted in Figure 2). The purpose of this section is to clarify deployment details of all the critical components of the NebulOuS 1st integrated release, across continuum resources (as defined in the Terminology subsection):

a) **Cloud-to-Edge case**: NebulOuS uses infrastructure resources that span the entire Cloud-to-Edge Continuum to form an application cluster. This involves the use of a public or private Cloud to deploy one Kubernetes master and one or more Service Provider Edges that is/are offered by a Telecom Provider along with a User Edge, for deploying application components

b) **Multi-cloud case**: NebulOuS uses only Cloud infrastructure resources to form an application cluster, but this cluster can span different public Cloud providers (e.g. AWS and GCP) and on-prem infrastructure (thus, this scenario includes hybrid Cloud).

c) **Edge case:** NebulOuS uses only Edge resources to form an application cluster. In this case, no public or private Cloud resources are being used, but the Edge nodes may be of varying capabilities, including both powerful and resource-constrained resources that are part of either the Service Provider Edge or the User Edge (e.g. including Edge Node infrastructure provided by any third-party entity, as user-owned Laptops, NUCs, Raspberry Pi devices, etc.).

In Figure 2 we depict the NebulOuS system components in red colour and the application components in light blue colour. The control plane components are containerized microservices that can be

deployed in various setups; e.g. the deployment can be centralized or distributed, single-location or multi-location, on public Cloud or on-prem. Within the project, for now, we use k8s to deploy the control plane components in OpenStack VMs offered by UiO. Note that this is just "where the platform runs": this control plane k8s cluster is **not** part of the application clusters that are being formed and provisioned by NebulOuS on the Cloud-Edge Continuum.

The physical architecture of NebulOuS is presented with a focus on the execution model of its components in Cloud-Edge Continuum resources (alongside the application components). The internal details of the NebulOuS Control Plane are abstracted, since they have already been presented in Figure 1 and analysed in the aforementioned deliverables.

The NebulOuS system components that are deployed in the Cloud-Edge Continuum resources consist of i) NebulOuS host-level components and ii) cloud-native tools deployed on top of Kubernetes. There is a distinction between cluster Master and Worker nodes: some components are executed only on the Master nodes, while other ones only on the Worker nodes. Those components are executed in the application resources alongside containerized application components. The specific NebulOuS system components that are executed in the application cluster resources are the following:

- **Master nodes**: Event Processing Manager, Optimizer Controller, k8s-gatekeeper, WireGuard client.
- **Worker nodes:** Event Processing Agent, WireGuard client.

For the 1st release, we do not differentiate between Cloud and Edge resources with respect to what system components are being installed; the same NebulOuS components are deployed in all resource types. We plan to investigate more fine-grained options at the second half of the project (e.g., using more lightweight implementations for the Edge, e.g. k3s[9] instead of k8s).

Since NebulOuS is used to deploy hyper-distributed applications on the Cloud-Edge Continuum, there needs to be a way for users to describe their application deployments. The application graphs are defined using the Open Application Model[10] and consist of containerized components. Those components are deployed in Kubernetes clusters using KubeVela.[11]

An application cluster is practically a Kubernetes cluster, which is formed by either a) Cloud resources, b) Edge resources, or c) both Cloud & Edge resources. Installing Kubernetes in either Cloud-only or Edge-only resources, especially when the resources are owned by a single provider, is more-or-less already supported by the Kubernetes ecosystem. However, the formation of distributed k8s clusters that pool resources scattered in the Cloud-Edge Continuum and owned by different providers, is a significant advancement of the NebulOuS project.

Clusters can be formed by provisioning virtual or physical resources, either by spinning up and provisioning VMs, or by onboarding and provisioning bare metal infrastructure - the latter is especially considered in Edge locations.

[9] https://k3s.io/
[10] https://oam.dev/
[11] The initial models used by NebulOuS for application deployment, monitoring, optimization and resource discovery are detailed in D2.2.

On top of this infrastructure, the NebulOuS runtime components are installed. This consists of the NebulOuS agents (Event Processing Agent, Overlay Network Agent), the containerd[12] runtime and Kubernetes. Moreover, after the installation of Kubernetes, two cloud-native tools also need to be installed in the existing k8s clusters: KubeVela[13] and Casbin's k8s-gatekeeper[14].

The Proactive Resource Manager (part of the Activeeon's Proactive[15] solution for workload automation) is the control-plane component that handles this installation as part of its resource management functionality. It successively onboards a resource (e.g. spinning up a VM in the case of Cloud nodes), installs the Overlay Network agent, the EPM agent, a container runtime and k8s, and finally KubeVela and Casbin k8s-gatekeeper.

The role of each NebulOuS component installed in a managed cluster node is the following:

- The **WireGuard client** is a VPN client that is installed by the Overlay Network Manager in the provisioned resources. WireGuard secures communication between the cluster nodes by creating tunnels at the host level, establishing an encrypted overlay network for each cluster. With the assistance of the bootstrapping scripts and *Overlay Network Manager* control-plane service, each node is onboarded into this secure overlay network.
- The **Event Processing Agent** collects monitoring metrics from the resources. It communicates with the **Event Processing Manager** to form monitoring clusters and announce relevant cluster information to the Event Processing Manager that is part of the NebulOuS control plane.
- **Kubernetes** and **containerd** allow for the execution and management of containerized applications.
- **KubeVela** allows for the deployment of Open Application Model-based service graphs in Kubernetes clusters.
- **Casbin k8s-gatekeeper** is a policy engine that acts as a Kubernetes admission controller. It allows for the deployment and enforcement of user-defined access control rules based on various models (ABAC, RBAC, etc.) in Kubernetes clusters. The policies are managed by the *Security and Privacy Manager* control-plane component.
- The **Optimizer Solver Component** is in charge of finding the optimal configuration for the application given the application's current execution context, i.e., the subset of the metric values used in the optimisation problem. The optimisation problem is unique to each application and its utility (function) to be maximised. The optimisation problem is received from the Optimizer Controller when the application cluster has been initialised, and reconfigurations will be triggered by SLO Violation messages from the SLO Violation Detector. The optimal configuration found by the Solver will make the Optimizer Controller reconfigure the application cluster and redeploy the application's components (pods).

[12] https://containerd.io/
[13] https:// kubevela.io
[14] https://github.com/casbin/k8s-gatekeeper
[15] https:// proactive.activeeon.com

Once a cluster has been set up, it can be managed by the NebulOuS control plane to deploy containerized applications on the brokered Cloud-Edge Continuum resources. NebulOuS automatically optimizes those deployments based on user-defined utility functions and re-configures the deployments to meet certain criteria.

2.4. COMPONENT INTERACTIONS AND SEQUENCE DIAGRAMS

We distinguish between three main user types, that will interact with the NebulOuS platform using GUI:

- DevOps/App owners: define the details of their applications that comprise hosting constraints and preferences, monitoring aspects, SLOs and SLAs, along with optimisation goals
- Cloud, Service Provider Edge and User Edge providers: announce the capabilities of their resource provider offerings
- Admins: define organisation-wide preferences and fine-tune the system

After providing all necessary input to the platform, based on the user type, the Meta-OS undertakes all necessary actions in the background (with respect to application deployment, monitoring & optimization, resource brokerage & management, security management, etc).

There are different information flows which take place during specific actions. Figure 3 provides the overall picture, with respect to the information exchange that takes place within the platform components that have been implemented for the 1st release (mainly using the dedicated ActiveMQ broker and respective AMQP messages, alongside some secondary REST calls). Communication via AMQP is facilitated using the NebulOuS Communications Middleware, as described in Section "Middleware integration". This Figure also depicts the main ways that end users use to interact with the platform.

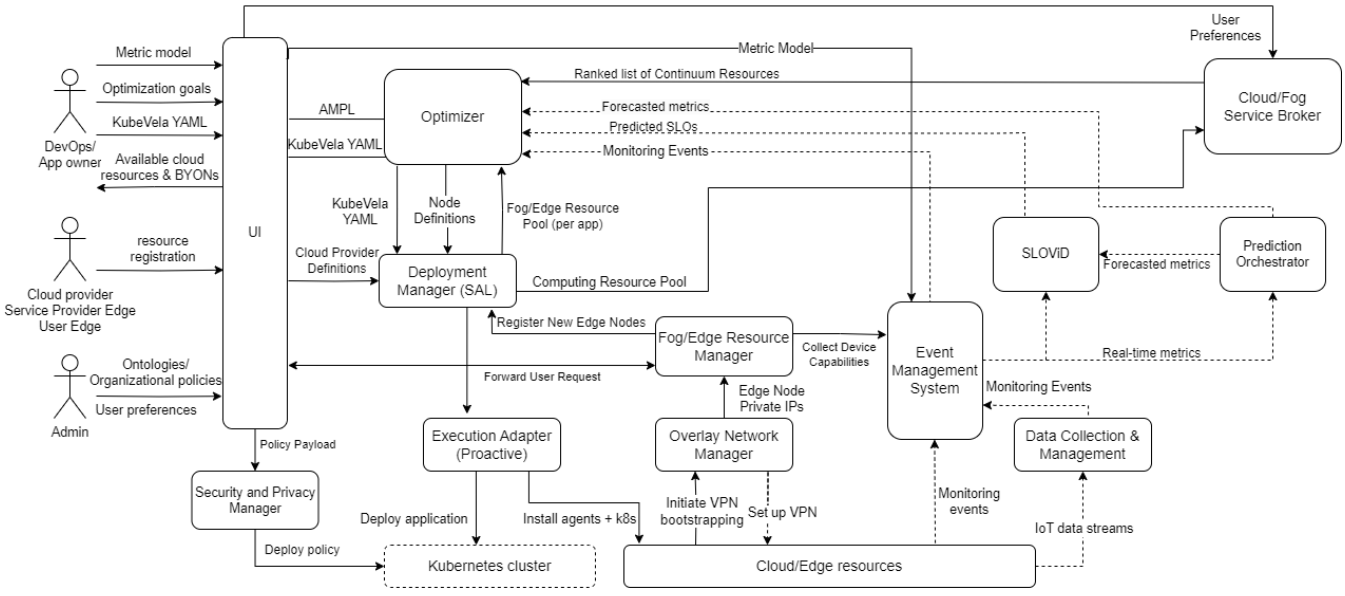


Figure 3. Summary of the NebulOuS component interactions

To provide a deeper view of the specific flows that take place within NebulOuS, we have created sequence diagrams that are presented below, as a graphic overview of the interactions between the components and the messages that they exchange during the interaction. List of the topics in communication between the components with their descriptions and usage context is provided in the Annex.

The **Cloud provider registration** sequence (Fig. 4) describes the process by which the Admin registers new Cloud providers by interacting with GUI, which will connect to SAL through the EXN-Middleware. Using the passed credentials, SAL will communicate with the Cloud providers using their respective API's such that SAL is capable of retrieving details related to the nodes that can be deployed on top of each Cloud provider. This process is done asynchronously, and the details collected are then saved in the database of SAL. In addition, GUI checks with SAL that node candidates are up to date. SAL is asynchronously updating and checking if there are new node candidates. Once the information is collected, the GUI is then capable of getting the information directly from SAL using a tailored endpoint.

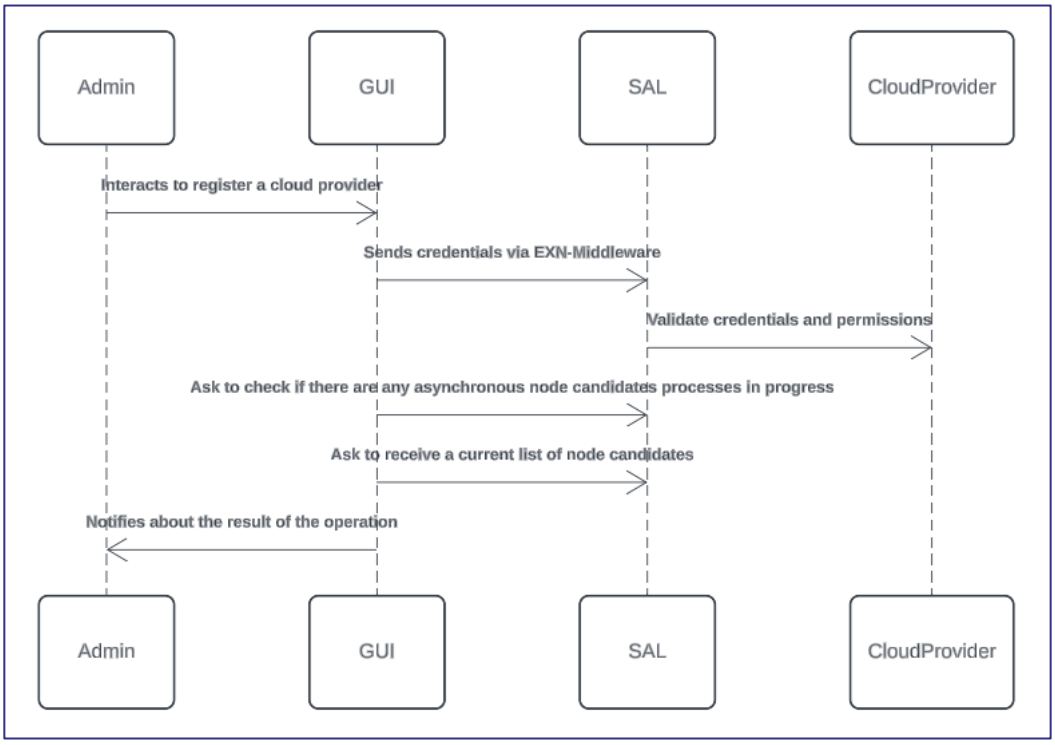


Figure 4. Cloud provider registration sequence

The **Service Provider Edge/User Edge device registration** sequence (Fig. 5) describes the process of registering Service Provider Edge/User Edge device by its owner using a GUI that contacts Resource Manager (RM) which stores the request in database. Next, the Fog/Edge Resource Manager asks EMS for the collection of device capabilities. EMS runs the process on Service Provider Edge/User Edge device and reports back to Fog/Edge Resource Manager, that registers the received capabilities in its NoSQL database. Afterwards, Fog/Edge RM contacts EMS for the device onboarding that installs its Agents on the device. Next, Fog/Edge RM sends a message to the database to register the device. Following, Fog/Edge Resource Manager will inform SAL about new device.

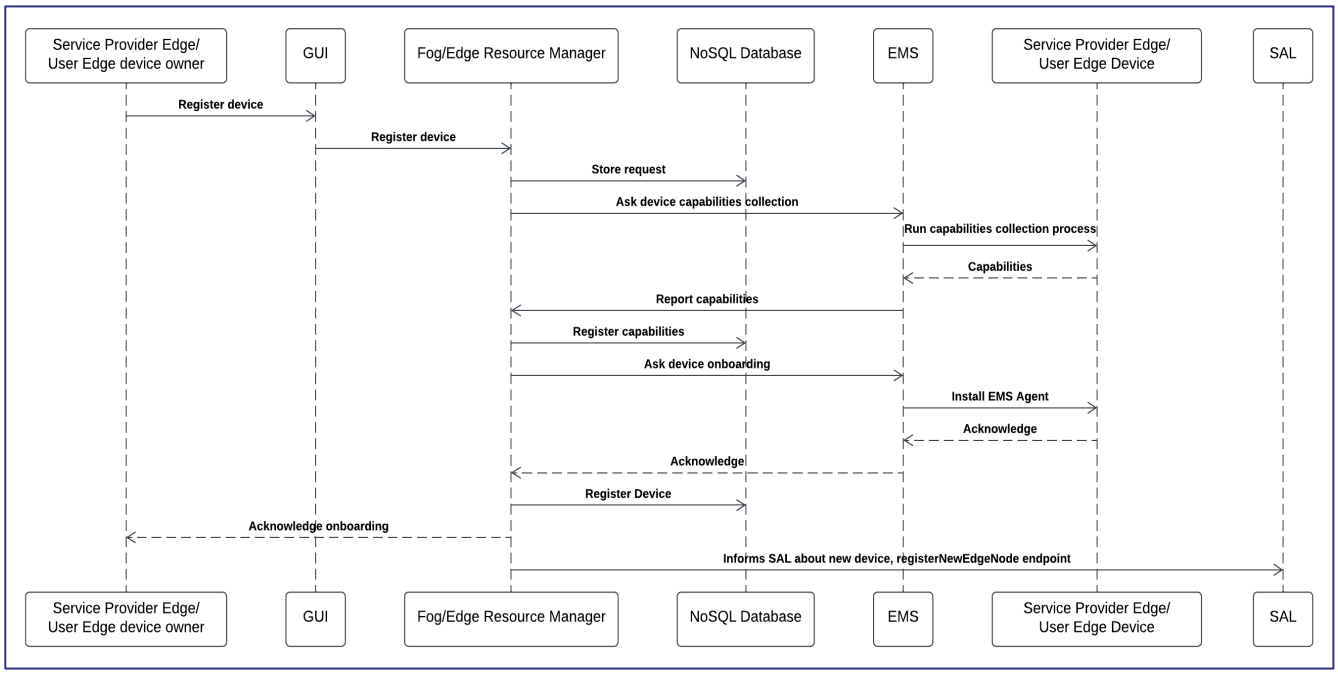


Figure 5. Service Provider Edge/User Edge device registration sequence

The **Application definition** sequence (Fig. 6) describes the steps followed to define an application in NebulOuS. User clicks on "Add application" button and fills the application details with the KubeVela manifest file and value ranges for the component variables, such as CPU or GPU, that will be used during the optimization process. Next, after moving to the "Resources" tab, the GUI requests from SAL a list of candidate computing nodes (Cloud, Service Provider Edge, User Edge) and then, the User can select preferred provider for instance types and devices. In the "Metrics" tab, the User defines templates, parameters, the metric model and SLO constraints which should not be violated. After moving to "Expression Editor" tab, the User defines the utility function for the application and selects the "Save" button.

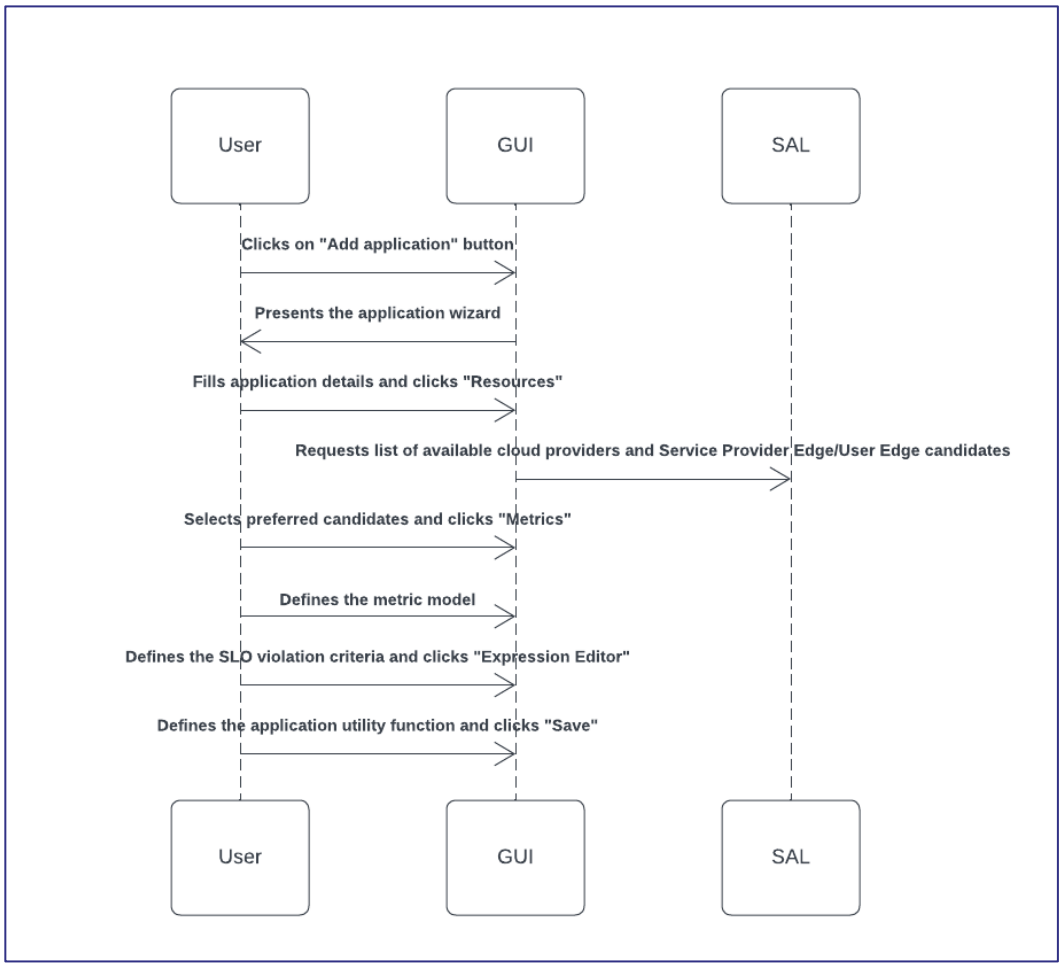


Figure 6. Application definition sequence

Initial application deployment sequence (Fig. 7) describes, how an application is deployed in NebulOuS after it has been defined by the user in the GUI. Once the user clicks on the "Deploy" button of the GUI, the app creation message and metric model messages are sent to the NebulOuS message broker. APP creation message is received by the Optimizer Utility Evaluator and it produces a message describing the relevant metrics for the deployment optimisation. These metrics, together with the original APP creation message are received by the Optimizer Controller. With this information, Optimizer Controller starts the process of deploying the app. First, it calls getRankedNodeCandidates for each component of the APP and master on CFSB (Cloud Fog Service Broker) to obtain a ranked list of node candidates for each cluster node. CFSB, in turn, calls SAL via EXN-Middleware, to get a list of CloudCandidates/EdgeCandidates, merges the lists and calculates the ranking for each one (synchronously). CFSB answers with a list of Edge nodes and Cloud instance types, with score and ranking attributes for each candidate. Afterwards, Optimizer calls defineCluster endpoint on SAL using the EXN-Middleware, with request: node_candidate_id for cluster master, node_candidate_ids and node names for workers, applicationID. In response to this it receives the clusterID. Afterwards, the Optimizer calls the deployCluster endpoint via the EXN middleware, which instructs SAL to create a new Kubernetes cluster by putting up new host VMs, running VPN setup scripts in the hosts (that interact with Overlay Network Manager running on the NebulOuS core) to create an overlay network

between the resources, running Kubernetes installation scripts, installing Cilium, KubeVela, the Optimizer-Solver component and k8s-Gatekeeper on the k8s cluster. Next, the Optimizer Controller instructs SAL to set appropriate labels to each cluster node and augments the KubeVela manifest file with node affinity information for all components using the same labels and calls deployApplication on SAL via EXN Middleware, sending the augmented KubeVela manifest file. Once Optimizer Solver running on the application cluster is initialized, the Optimizer Controller sends the AMPL file that describes the optimisation problem for the application. In parallel, EMS server running on the application cluster contacts the EMS server running on the NebulOuS core to obtain the metric model and SLO for the application (that was previously cached by the EMS server from the original messages sent by the GUI to trigger the app deployment). Also, in parallel, the metric Prediction Orchestrator sets up the infrastructure of metric predictors relevant to the application using the information contained in the messages sent by the GUI at the beginning of the deployment.

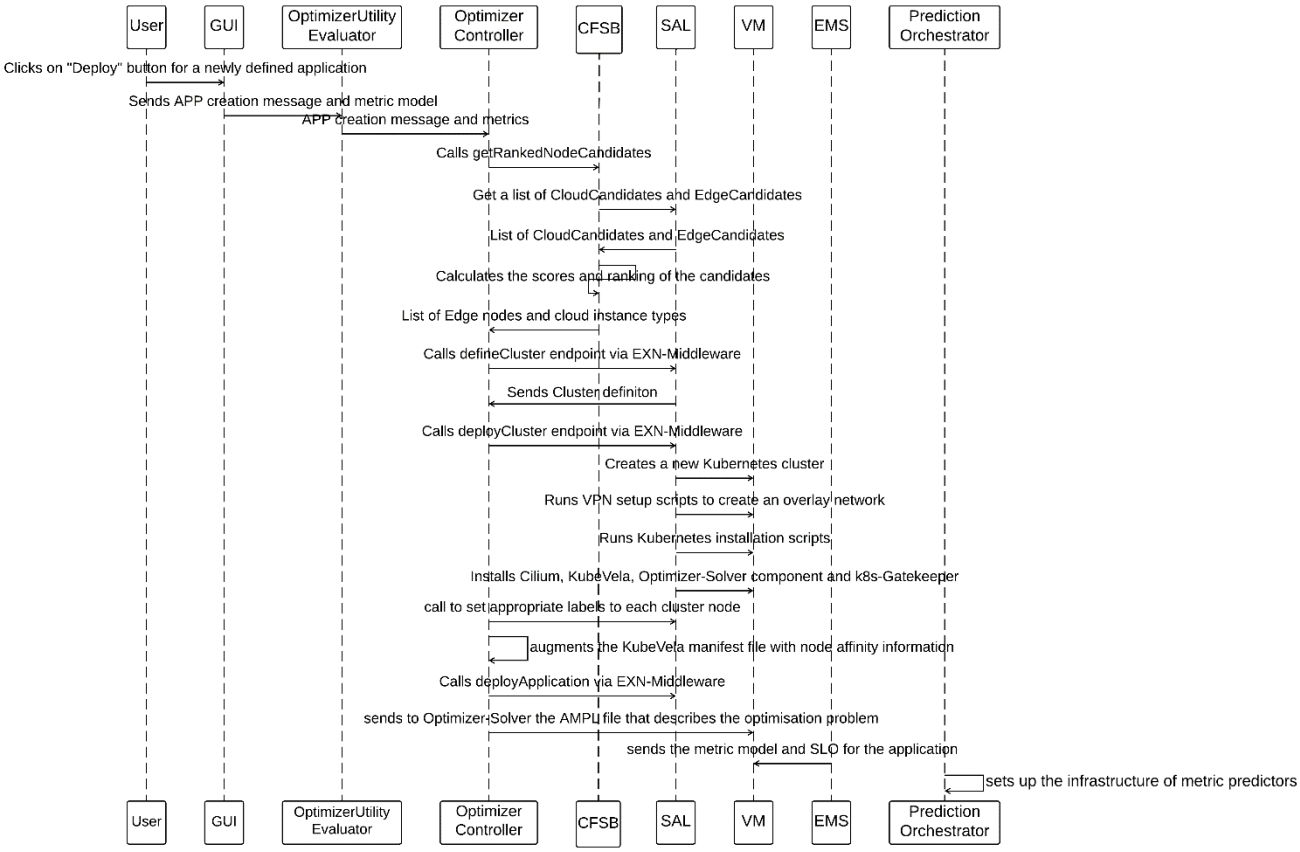


Figure 7. Initial application deployment

The **Deployment adaptation (reconfiguration)** sequence (Fig. 8) presents the process of application reconfiguration. After the initial deployment has been executed, the EMS collects events from the infrastructure and from the application sensors and sending real-time metrics to Prediction Orchestrator, SLOViD and SLO Violation indications to the Solver. EMS publishes metrics for Prediction Orchestrator which will send predicted metric values to the Solver of the Optimizer (metrics in SLOs and in the UF) and forecasted metrics to SLOViD. Solver also receives the SLO Violation indications from the EMS and SLOViD. The Solver subscribes to these events and starts

searching for a new configuration that will work better than the current for the set of cached predicted metric values. The Solver publishes the found solution. The Optimizer Controller subscribes to the solutions published by the Solver, and when a new solution arrives, the Optimizer Controller sends a request to the Cloud Fog Service Broker to obtain the best node candidates for the new solution. When the response has arrived, the Optimizer Controller calls the scaleOut endpoint on SAL through the EXN middleware to create and add new nodes to the application cluster. Once the new nodes are ready, the optimiser-controller adds updated nodeAffinity traits to the KubeVela file and calls the deployApplication endpoint with the updated KubeVela manifest file as payload via the EXN middleware to redeploy the Application. KubeVela redeploys the application's components following the updated node affinity traits. Finally, the Optimizer Controller calls the scaleIn endpoint to remove the nodes that are not in use by the application anymore.

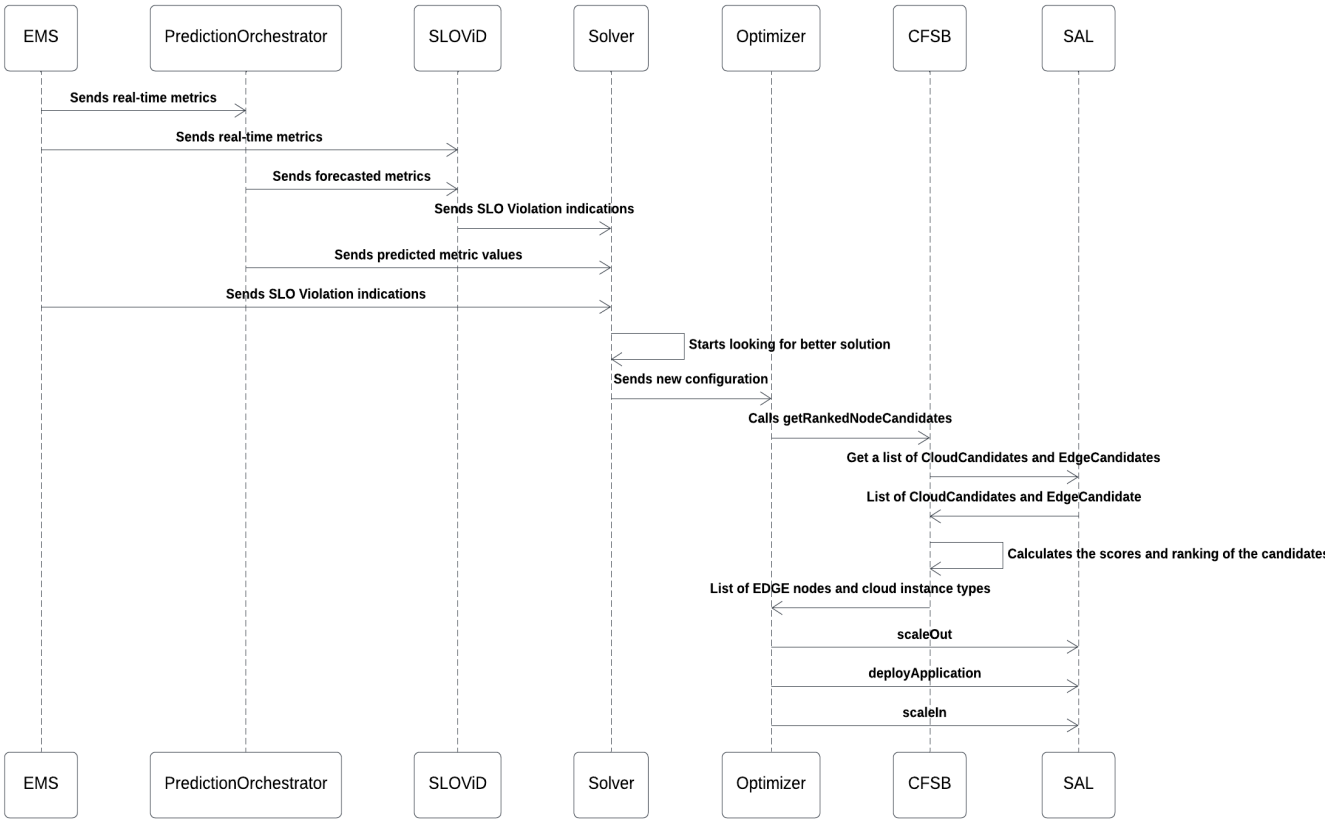


Figure 8. Deployment adaptation (reconfiguration) sequence

3. THE SCOPE OF THE 1ST NEBULOUS RELEASE

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

User Interface: The current version allows users to define their application by providing the KubeVela application manifest file, the metric model and SLO and the optimization goal and constraints. After that, the user can request the deployment of the application. The current version of the GUI offers sections for managing Cloud providers and user accounts. Also, a section exists where

an administrator can define Casbin policies to be delivered to the Security&Privacy Manager component.

Fog/Edge Resource Manager: The current version of 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.

Security & Privacy Manager: An access control mechanism for managing user application clusters has been designed and implemented. For this a Security and Privacy Manager offers the necessary methods to perform CRUD operations for cluster admission rules expressed in Casbin. These policies are then sent to the Casbin policy engine that acts as an admission controller for the Kubernetes cluster, effectively enforcing the policies.

Optimizer: Upon reception of an application deployment request from the UI, the current version of 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.

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 and OpenStack) 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 current version of the component handles the creation an on-demand VPN 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.

Data Collection & Management System: The use of Apache Artemis for articulating an application pub-sub mechanism has been proposed. A plugin that collects relevant metrics regarding the utilization of the message broker by the application components has been implemented. This 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 Messaging 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).

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 receive metrics collection 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.

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 Optimizer to select the nodes to be used to deploy a user application.

Brokerage Quality Assurance: The ontological models necessary to organization wide meta-constraints as well as application constraints has been defined. This permits to use semantic reasoning to automatically validate these constraints. In a future release of NebulOuS, this mechanism will be integrated with the UI to automatically validate the consistency of the application defined constraints.

AI-driven Anomaly Detection: A list of potential anomaly detection algorithms (k-nearest neighbours, logistic regression, random forest, decision tree classifier algorithm, etc…) have been investigated using the NSL-KDD dataset. An initial deployment of the metrics collection mechanism has been proposed and the interconnection with the rest of the system planned. After use cases are integrated using the first NebulOuS release, a case-by-case analysis will be conducted to select the most suitable anomaly detection algorithm to be used.

SLA Generator: The ontological Service-level Agreement (SLA) Generator models the SLAs formed by NebulOuS and its users, when an application gets registered. The Q-SLA ontology is used to capture the SLAs, which contain a set of Service-level Objectives (SLOs) that NebulOuS can fulfil along with other related information such as the involved parties, and compensation in case of violations. The SLA is formed using information taken from the metric model and KubeVela manifest files and after the SLA is formed; it is ontologically stored, and other components can query for information through a REST API. It is implemented in Java as a Springboot server and uses the OWL API for managing the ontology.

Serverless: Knative[16] has been selected as the preferred approach for offering serverless capabilities to NebulOuS users.

Smart Contract Encapsulator: Hyperledger Fabric V2.5 installation has been performed on a separated environment and efforts are being put on investigating how to translate user defined SLAs into smart contracts. It is envisioned that a generic smart contract template with configurable parameters will be implemented to simplify the generation process. Such a contract template will be used to call other contracts like management interface, performance monitoring, penalty schemes, etc.

[16] https://knative.dev/docs/

Workflows: Workflow executor is in active development. The first release includes the deployment of the workflow application as a KubeVela manifest file and a related metric model covering both the metrics of the executor and the metrics of the working nodes. The workflow executor, ProActive scheduler, and Prometheus 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.

As a summary, the current 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.

Some of the components were adapted from previous projects, but eight of them were designed and developed from scratch specifically for our platform. To estimate the maturity of the technology we used method of a Technology Readiness Levels (TRL), that helps us measure the progress of the project. The Table below presents the developed components with their authors, their TRLs at the beginning of the project, the current status and the assumed readiness for the final release, and in addition the advancements that were made during the life cycle of the project.

Table 1. Authors, TRLs and Maturity of the components

Component	Author	Initial TRL	Current TRL	Final TRL	Advancements
GUI	EXZ	-	4	6	Developed and dedicated for NebulOuS, based on requirements described in more detail in D 2.1

EXN Middleware	EXZ	-	4	6	This component, along with its adjacent libraries, has been developed in order 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	4	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 for details.
EMS	ICCS	4[17]	4	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, context-aware metric event propagation, k8s-based deployment, Prometheus / OpenMetrics endpoints leveraging, etc. Described in more detail in D 5.1.
SLOViD	ICCS	4[18]	4	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.
CFSB	ICCS	2	4	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.
Security and Privacy Manager	UBITECH	-	3	5	Developed from scratch for NebulOuS, based on requirements described in more detail in D 2.1. Implements the necessary functionality to create cluster-wide security policies and enforce them. Described in more detail in D4.1.
Deployment Manager	AE	4[19]	5	6	Extended with cloud continuum deployment capabilities Described in more detail in D 4.1

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

Execution Adapter	AE	4[20]	5	6	Extended with cloud continuum deployment capabilities Described in more detail in D 4.1
Overlay Network Manager	UBITECH	-	4	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.
Data Collection and Management System	EURECAT	-	4	6	Developed and dedicated for NebulOuS based on requirements described in more detail in D 2.1
Smart Contract Encapsulator	UiO	-	-	5	To be developed for the second release. Concept and baseline technologies identified.
Brokerage QA	SEERC	-	2	5	To be developed in the second release. All ontologically expressed semantic models that underpin this component have been defined.
SLA Generator	SEERC	-	3	5	To be developed in the second release. All ontologically expressed semantic models that underpin this component have already been defined.
AI-Driven Anomaly Detection	EURECAT	2 (Utilities)	4	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. Described in more detail in D5.1.

4. NEBULOUS PLATFORM INTEGRATION AND DELIVERY STRATEGY

The right choice of integration strategy is crucial for the successful implementation of a given project. NebulOuS is promised as an open-source software solution, licensed under the Mozilla Public License 2.0 and for such software, it makes sense to develop it in public. It uses OpenDev[21], project by the Open Infrastructure Foundation[22], which provides code hosting, code reviewing and CI/CD capabilities for open-source projects. OpenDev uses Gerrit for change management and Zuul for CI/CD.

[20] MORPHEMIC https://www.morphemic.cloud/
[21] https://opendev.org/
[22] https://openinfra.dev/

This decision shows our strong commitment to using powerful open-source technologies such as OpenStack and Gerrit. OpenStack lays a reliable foundation for our Cloud services, offering the scalability and adaptability we need, while Gerrit is essential for improving our code review and quality assurance efforts. This strategic use of these technologies highlights our dedication to maintaining transparency, fostering collaboration, and upholding the highest standards of code quality in the evolution of the NebulOuS platform.
This section outlines the Continuous Integration (CI) and Continuous Deployment (CD) framework for the NebulOuS platform, emphasizing the adoption of OpenDev's Gerrit and Zuul for comprehensive code review and management, alongside the utilization of Flux for streamlined, automated deployments. The main goal of this strategy is to refine the development workflow, ensuring a seamless and dependable transition from code inception to its deployment in live production settings. Integrated middleware acts as an intermediary layer that facilitates communication, data exchange, and interoperability between disparate systems, facilitates the transformation and mapping of data formats, structures, and protocols to ensure compatibility and consistency.

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

- **Unified Deployment Strategy**: Utilizing Kubernetes, we deploy all components in a coordinated manner, ensuring they work together seamlessly.
- **CI/CD Pipeline**: The integration of Zuul CI and Flux provides a continuous integration and deployment pipeline that ensures all code changes are tested, reviewed, and deployed efficiently.
- **Environment Consistency**: By maintaining distinct environments for production, testing, development, and continuous deployment, 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 CI/CD pipeline and production environment provides insights into system performance and health, allowing for proactive issue resolution.

4.1. CONTINUOUS INTEGRATION VIA ZUUL CI

Zuul CI[23], that was created by the OpenStack community, is an open-source platform tailored for project gating (this term refers to a process where changes submitted for integration into the main codebase are automatically tested and verified before they are allowed to merge), alongside the automated merging and testing of code. The gating process aims to hinder the merging of changes that introduce regressions. This ensures that the mainline of development remains accessible and functional for all developers, and only when a change is tested and verified to work seamlessly, it is merged. Project repositories contain a configuration file zuul.yaml, that defines the CI pipeline, including jobs, triggers, and dependencies. Zuul CI detects new code changes in the repository and

[23] https://zuul-ci.org/docs/zuul/latest/index.html

triggers a CI pipeline based on the configured workflow, and then executes the defined jobs in isolated environments (e.g., containers or virtual machines) to ensure consistency and reproducibility. It collects test results and provides feedback to developers, indicating whether the code changes pass or fail the defined tests and furthermore, generates reports and notifications, informing of the build status and any issues encountered during the CI process. If code changes pass all tests and meet the defined criteria (e.g., code review approval), Zuul CI places them in the merge queue. Once it reaches the front of the queue, Zuul CI merges them into the main branch of the repository. Developers receive continuous feedback on their code changes, enabling them to iterate quickly and address any issues identified during the CI process. Zuul continuously monitors the CI/CD pipeline and provides insights into the performance and health of the system.

Integration with Gerrit: The synergy between Zuul CI and Gerrit, a tool dedicated to code review and project oversight, facilitates a smooth code review process. Within this framework, code alterations require Gerrit's approval prior to undergoing testing and subsequent merging by Zuul CI. Gerrit triggers Zuul CI to run tests and verify the changes. Developers can view the status of their code changes directly within Gerrit, including information about passed or failed tests, and they can collaborate within it to review code changes and provide feedback. Each code change is submitted as a single unit known as a change or patch set that ensures that code changes are self-contained and can be reviewed, tested, and approved independently, simplifying the code review process and minimizing the risk of integration conflicts.
This integration not only upholds a superior code quality standard but also promotes a collaborative development environment.

Advantages Include:

- **Quality Assurance**: Guarantees comprehensive review and testing of all integrated code.
- **Flexibility**: Adapts to complex project dependencies, making it ideal for the multifaceted architecture of the NebulOuS platform.
- **Collaboration and Transparency**: Promotes an open review culture, fostering enhanced collaboration among developers. Reviewers can leave comments directly on specific lines of code, facilitating focused discussions and enabling efficient code review.

Disadvantages Include:

- **Impact on pace of development:** Delays in merging changes and the need for extensive revisions may frustrate contributors and impede the pace of development.
- **Long Wait Times for Merging Changes**: Gerrit's rigorous code review process, coupled with Zuul CI's testing requirements, can lead to significant delays in merging changes.

4.2. CONTINUOUS DEPLOYMENT WITH FLUX

Flux[24] stands as a pivotal tool in automating Kubernetes deployments, vigilantly tracking alterations within source code repositories and effortlessly integrating these changes into the cluster. This diligent monitoring guarantees that the production environment remains aligned with the most recent codebase.

[24] https://fluxcd.io/

Among its primary advantages, Flux simplifies the deployment process by significantly diminishing the need for manual oversight, thereby enhancing operational efficiency. Furthermore, it ensures a harmonious alignment between the deployment environment and the source code, bolstering the system's reliability. This alignment means that the deployed system accurately reflects the configurations and updates specified in the source code, reducing the risk of errors and downtime.

By embracing GitOps principles[25], Flux also elevates the security and auditability of the deployment process, instilling confidence in the integrity and oversight of the development lifecycle.

4.3. REPOSITORY FOR NEBULOUS

In crafting the NebulOuS platform, we've chosen to house our codebase on the OpenDev[26] collaborative platform. Development follows Gerrit's model of reviewed changes. Each change has to be reviewed by at least one maintainer of the code before being merged. It also needs to pass CI checks - there is no possibility of pushing directly to the main working branch.

For a component to be included in the NebulOuS platform, it must meet certain requirements to ensure consistency, reliability, and ease of deployment. Each component should provide its own Helm chart as it's the most popular method to install components on a Kubernetes cluster:

- **Health Checks**: The Helm chart must incorporate health checks to ensure the component's availability and stability. Health checks enable automated monitoring and help maintain the overall health of the platform.
- **Dependency Management**: Any dependencies required by the component should be clearly defined within the Helm chart. This ensures that all necessary resources are provisioned correctly during deployment and helps manage dependencies efficiently.
- **Chart Structure**: The Helm chart directory should contain all necessary files related to the deployment of the component, such as templates, values files, and metadata.

4.3.1. Integration Flow of the NebulOuS Platform

The integration flow of the NebulOuS platform involves several key stages to ensure that code changes are properly reviewed, tested, and deployed. This flow leverages continuous integration and continuous deployment (CI/CD) practices using Zuul CI, Gerrit, Flux, and a Kubernetes cluster to manage the environments. Below is a detailed breakdown of the integration flow.

1. Code Submission and Review

- **Developer Submits Code**: Developers submit their code changes to the NebulOuS repository. Each change is treated as a patch set.

[25] https://opengitops.dev/
[26] https://opendev.org/nebulous

- **Gerrit Code Review**: The submitted code change is reviewed in Gerrit. Reviewers evaluate the code, leave comments, and approve or request modifications. Approval in Gerrit is mandatory before further processing.

2. Triggering the CI Pipeline

- **Zuul CI Detection**: Once the code change is approved in Gerrit, it triggers Zuul CI to initiate the CI pipeline.
- **zuul.yaml Configuration**: Zuul CI reads the zuul.yaml configuration file from the repository, which defines the CI pipeline, including jobs, triggers, and dependencies.

3. Automated Testing and Verification

- **Isolated Environments**: Zuul CI executes the defined jobs in isolated environments (e.g., containers or virtual machines) 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 Zuul CI provides feedback to developers through Gerrit, indicating whether the tests pass or fail. Developers can address any issues identified during this phase.

4. Merging Approved Changes

- **Merge Queue**: If the code change passes all tests and meets the defined criteria (e.g., code review approval), Zuul CI places it in the merge queue.
- **Merging**: Once the change reaches the front of the queue, Zuul CI merges it into the master branch of the repository.

5. Continuous Deployment

- **Flux Monitoring**: Flux continuously monitors the source code repositories for changes. When a change is detected in the main branch, Flux triggers the deployment process.
- **Kubernetes Deployment**: Flux integrates the changes into the Kubernetes cluster, deploying the updated components across the appropriate environments.

4.4. LESSON LEARNT AND FUTURE PLANS FOR THE INTEGRATION AND DEVELOPMENT.

Despite the benefits of the current development and integration solution based on the OpenDev platform with Gerrit, Zuul and Launchpad tools, there have been efficiency problems for many in-consortium developers who found the platform too different from the tooling they are accustomed to. The very strict code review and automated checks policies were additionally slowing down the pace of this kind of "research development" (as it happens in this kind of research and innovation projects). Thus, the consortium has made a decision to move the effort for the development and integration of

the second and final release of NebulOuS to GitHub. Additionally, the policies around code review and automated checks will be relaxed.

4.5. SECURITY AND DATA SANITIZATION

The NebulOuS platform integrates Cloud, Service Provider Edge and User Edge computing technologies to ensure the Cloud Continuum across heterogenous resources. This necessitates rigorous privacy and security protocols, particularly during resource decommissioning or reconfiguration. In this section, we review data sanitization techniques. Following the analysis of data sanitization techniques, we reason about the most appropriate technique which will be adopted by the NebulOuS platform and be integrated with NebulOuS's brokerage, deployment and reconfiguration mechanisms.

4.5.1. Assessment of Data Sanitization Methods

Cryptographic Erasure (Crypto Erase):
Methodology: Secures the entire storage medium through encryption, effectively making data unrecoverable once the encryption key is eliminated.
Benefits: Facilitates the repurposing of devices, highly compliant with various data security and privacy standards and regulatory frameworks.
Limitations: Necessitates initial encryption setup and continuous encryption key management.

Data Erasure:
Methodology: A software-based method that overwrites data on storage devices with zeros, ones, or random patterns, making the original data unrecoverable.
Benefits: Permits device repurposing and complies with multiple regulatory frameworks.
Limitations: Less effective for solid-state drives (SSDs) and reliant on storage providers' cooperation after decommissioning. Does not protect against data exfiltration through medium theft.

Physical Destruction:
Methodology: Irreversibly damages the storage media, by shredding, crushing, incineration, and using degaussers (for magnetic storage) and making data retrieval impossible
Benefits: Ensures complete data elimination, maximally compliant with various data security and privacy standards and regulatory frameworks.
Limitations: Conflicts with environmental sustainability objectives and prevents device repurposing.

4.5.2. Recommended Sanitization Approach for NebulOuS

Cryptographic erasure stands out as the most suitable and efficient technique for data sanitization within the NebulOuS framework. This method meets the platform's demands for a scalable solution that secures data beyond decommissioning, while also ensuring compliance with pertinent regulations and standards.

Implementing cryptographic erasure in Edge devices, especially those shared with other applications, requires careful consideration of several factors to ensure effectiveness and security:

1. It's crucial to isolate the cryptographic erasure process from other applications running on the Edge device to prevent unauthorized access or interference. This can be achieved through containerization, virtualization, or hardware-based isolation mechanisms.

2. Cryptographic erasure relies on encryption keys to securely render data unrecoverable. Proper key management practices, such as storing keys securely and ensuring access control, are essential to prevent unauthorized access to sensitive data.
3. The cryptographic erasure process should employ strong encryption algorithms and secure deletion techniques to ensure that data cannot be reconstructed or recovered after erasure.
4. Edge environments are dynamic, with devices frequently connecting and disconnecting from the network. Cryptographic erasure mechanisms should be resilient to network interruptions and capable of initiating erasure processes automatically when devices are decommissioned or taken offline.

4.6. MIDDLEWARE INTEGRATION

Middleware components, such as message brokers, ESBs (Enterprise Service Buses), or integration platforms, facilitate interactions between disparate components by providing communication and integration capabilities. The EXN Middleware routes messages and events to the appropriate destinations based on predefined rules and criteria, enabling dynamic routing, content-based routing, and message filtering. As described in D5.1, it is used as a critical architectural element within our technological framework, embodying a comprehensive solution for fostering service integration and smooth communication across varied systems and protocols. Along with client-side libraries, this component ensures dependable asynchronous communication the AMQP protocol.

By using the EXN Middleware during the integration process we ensured a uniform way to oversee communication, data exchange, and interoperability between different application components or across various applications. The EXN Middleware, specifically, offers considerable benefits by abstracting the complexities involved in integrating system components which differ in protocols used for communication, as well as a message payloads. This flexibility and scalability enable the integration of different technologies and protocols, allowing systems to evolve and adapt to new requirements seamlessly. It also ensures reliable message delivery and efficient resource utilization, which are crucial for minimizing data loss risks and optimizing performance.

As part of the EXN Middleware, a set of libraries have been developed, abstracting the boilerplate code required to integrate and use the AMQP protocol efficiently. These are client-side libraries are used by components in order to homogenize topic names and message payloads across communication channels. This standardization facilitates easier discovery and subscription for consumers, who can navigate and subscribe to topics without the burden of understanding each producer's naming conventions. It also minimizes the risk of misconfiguration and errors in message routing, thereby ensuring that messages are accurately delivered to their intended destinations. As part of the integration effort, libraries were provided in the Java(exn-java-library) and Python (exn-python-library) programming language.

The middleware's role extends to facilitating service integration and proxying between various protocols and AMQP, significantly enabling seamless communication between disparate systems. By acting as a conduit, it allows these systems to communicate smoothly despite inherent differences in their underlying protocols. The middleware's ability to convert messages from one protocol to another, such as from HTTP to AMQP, is instrumental in integrating systems that employ different communication standards. This proxying capability enhances the system's flexibility, making it easier to integrate with external services and adapt to new integration challenges.

Through the EXN Middleware we integrated two core services with the rest of the components by exposing their HTTP interfaces to the rest of the components through asynchronous communication, SAL and the Cloud Fog Service Broker (CFSB).

The HTTP abstraction layer in the EXN Middleware follows a set of conventions:

- URL endpoints after the base URL of the HTTP interface, are converted to dot notation
- HTTP Method names are appended to the end of the topic name following the mapping in Table 1.
- Replies are sent using the calling topic name and appending **.reply**

This request-reply paradigm, was achieved by using the standard AMQP header property **correlation-id.** Components using the client-side are able to call methods on any endpoint in a *pseudo-sync* manner, given that the client-side libraries follow the Enterprise Integration Pattern for request-reply using the **correlation-id.**

The following table provides an example of how the HTTP endpoints for **/node/register** of the NebulOuS component SAL is mapped.

Table 2. Mapped HTTP endpoints

HTTP Method	**Topic Convention**
GET (default action)	eu.nebulouscloud.**sal.***node.register* eu.nebulouscloud.**sal.***node.register***.reply**
POST	eu.nebulouscloud.**sal.***node.register***.post** eu.nebulouscloud.**sal.***node.register***.post.reply**
PUT	eu.nebulouscloud.**sal.***node.register***.put** eu.nebulouscloud.**sal.***node.register***.put.reply**
DELETE	eu.nebulouscloud.**sal.***node.register***.delete** eu.nebulouscloud.**sal.***node.register***.delete.reply**

A pivotal focus of the EXN Middleware is its support for asynchronous communication, assisting in the management of this communication paradigm to ensure robust and efficient interactions.

5. TESTING METHODOLOGY

Testing is an essential part of every project created by the cooperation of many groups and people. Tests are needed to verify interactions between system components and to check if all requirements have been correctly implemented. System behaviour needs to be tested under various conditions and different workloads to verify its stability, resource utilization and error handling. To ensure quality and reliability, it is crucial to follow all guidelines and procedures of testing such as preparing and executing test cases and reporting any issues that occurred. In an iterative development process, test cases may be executed continuously throughout the development lifecycle, as new features are added or modifications are made, relevant test cases should be executed to maintain product quality.
To execute a test case:

- Ensure that the test environment is set up properly to mimic the production environment as closely as possible.

- Prepare the necessary test data required for the test cases. This may involve creating sample data or configuring databases.
- Choose the test case to be executed.
- Run the selected test case on the target system or software, following the steps outlined in each test case, including inputs, expected outcomes, and any specific actions to be performed.
- Record the actual outcome of the test case and document any discrepancies between the expected and actual results.
- After making corrections, if necessary, execute any regression tests to ensure that the changes do not negatively impact existing functionalities.
- Update the test case based on the results of test execution.

We decided to use the Launchpad[27] platform to track the life cycle of the test cases and reported issues.

This deliverable contains prepared test cases divided into two separate lists, based on the difference in their execution. The manual tests that will be performed by Telefonica are presented in section 5.4 and automated by Ubitech, in 5.5 section. The whole content of the prepared test cases for the first release, can be found in the Annex.
In the following sections, we will present the quality assurance guidelines and test cases management, including test case definition and test case creation.

5.1. QUALITY ASSURANCE GUIDELINES

Testing ensures that software meets specified quality standards, reducing the likelihood of defects in production. Identifying and fixing defects early in the development process is more cost-effective than addressing issues in the later stages or post-release. Apart from that, rigorous testing contributes to a more stable and user-friendly software product.

The core components of good Quality Assurance approach are:

1. **Unit tests**: to test the building blocks of an application, typically in isolation from the application's other units and components, code level testing prepared and executed by the development team.
2. **Integration tests**: to ensure that all interacting components are operating correctly together, and each feature/functionality of the system can be successfully executed.
3. **System tests**: to validate the complete system, including interactions between hardware and software components.
4. **Compatibility tests**: to ensure compatibility with different hardware configurations and software environments.
5. **Performance tests**: to evaluate the OS's performance under different workloads and stress conditions.
6. **Security tests**: to identify and address security vulnerabilities and weaknesses.
7. **Reliability and Stability Tests**: to ensure the OS remains stable over extended periods of use.
8. **Usability Test**: to evaluate the user interface and overall user experience.
9. **Regression tests**: to test previous features of the system (not delivered in current release), to ensure that they're still working properly after a software update

[27] https://bugs.launchpad.net/nebulous

By incorporating these testing strategies into the development and maintenance processes, OS developers and QA teams can identify and address issues early, resulting in a more stable, secure, and reliable operating system.

If any defects or issues are identified during test execution, these should be reported to the development team by using a bug tracking system Launchpad. Developers are requested to provide detailed information, including steps to reproduce the issue, screenshots, and logs, to assist developers in understanding and fixing the problem.

5.2. TEST CASE DEFINITION

A test case is a detailed set of test data, pre-conditions, expected results and post-conditions for the tested implementations, developed to verify compliance with a specific requirement. It is written to verify that the software behaves as intended and to identify any defects or error in its functionality. Each test case represents a specific scenario or condition that the software should be able to handle correctly. A test case describes how to perform a specific test. Some of the tests requires specific applications to be prepared for them:

- **Dummy application**, a simple application composed of a single component that each second prints the System time to console. No SLO is configured and exactly one instance of the service can run at any given time.
- **Scaling application**, an application composed of two elements: Controller and Worker
- **IoT application,** an application for processing IoT data streams
- **Serverless dummy application**, a simple application composed of two elements: Factorial and Controller.

5.3. TEST CASE CREATION

Since the Launchpad platform does not have a dedicated place to store test cases, we report them as bugs and label them with a release-specific name.
After logging in into Launchpad https://launchpad.net/nebulous, the Bugs tab should be selected and "Report a bug" clicked. In the next steps, number of the test case before the short description should be added. In the window for further information, preconditions, steps to complete and expected result are to be added, as presented on Figure 9.

Figure 9. Creating a Test Case in Launchpad

In the Extra options, as it shown on Figure 10, it's important for a Tag named *test-case-r.1.0.* to be added, as it will be used to filter them out from the reported bugs.

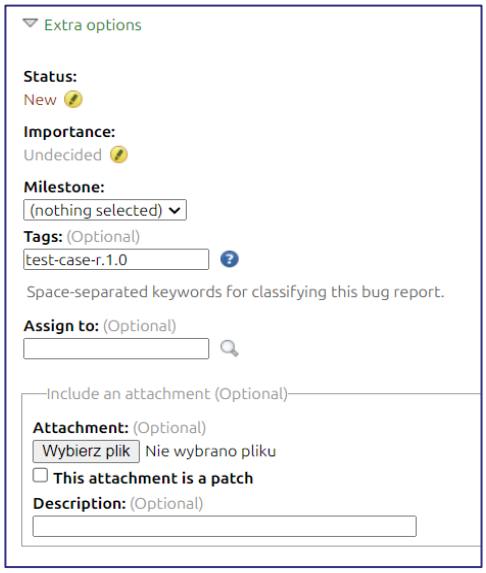


Figure 10. Extra options for created test case

Next, "Report a bug" should be selected by the user and the test case is created (Figure 11).

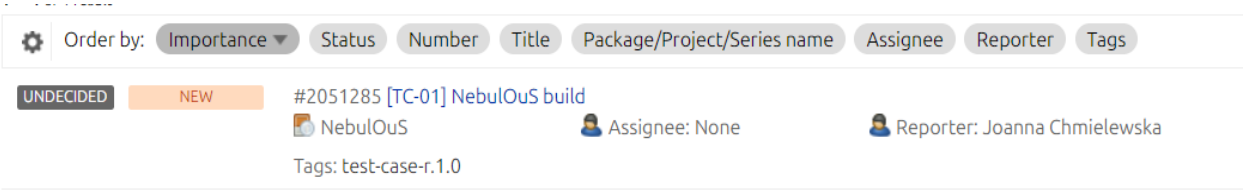


Figure 11. Created test case

5.4. MANUAL TEST CASES

Table 3 . List of the test cases prepared for manual execution.

ID	SUMMARY
TC_01/ 2051285	NebulOuS build
TC_02 / 2053066	NebulOuS core installation
TC_03 / 2053075	NebulOuS core installation fails when requirements are not met.
TC_04 / 2053076	NebulOuS core recovers after failure of any of its component
TC_05 / 2053077	Uninstall NebulOuS core

TC_06 / 2053078	Login into NebulOuS web UI (invalid credentials)
TC_07 / 2053079	Login into NebulOuS web UI (valid credentials)
TC_08 / 2053080	Logout from NebulOuS web UI
TC_09 / 2053081	Unauthenticated user can't interact with NebulOuS web UI
TC_10 / 2053082	Resource manager can onboard manually-managed nodes
TC_11 / 2053083	Onboarding of a manually-managed node fails
TC_12 / 2053084	Device owner can de-register his devices
TC_13 / 2053085	Resource manager can de-register devices
TC_19 / 2053086	Admin can register cloud providers in NebulOuS.
TC_20 / 2053087	Admin can de-register cloud providers in NebulOuS.
TC_21 / 2053088	App deployment on manually-managed nodes
TC_22 / 2053089	App deployment on manually-managed node fails due to lack of resources
TC_23 / 2053090	App deployment using NebulOuS cloud providers
TC_24 / 2053091	App deployment on NebulOuS-managed node fails due to lack of resources
TC_37 / 2053092	Cloud provider selected based on user preferences

Test cases highlighted in green will be executed both manually and automatically to expedite the testing process.

5.5. AUTOMATED TEST CASES

Table 4. List of the test cases prepared for automatic execution.

ID	SUMMARY
TC_17/2054105	NebulOuS agent recovers after failure with communication with NebulOuS core.
TC_18/2054106	NebulOuS agent recovers after one of its components fails.
TC_21/2053088	App deployment on manually-managed nodes

TC_22/2053089	App deployment on manually-managed node fails due to lack of resources
TC_23/2053090	App deployment using NebulOuS cloud providers
TC_24/2053091	App deployment on NebulOuS-managed node fails due to lack of resources
TC_25/2054110	NebulOuS reacts to node failure
TC_26/2054112	NebulOuS scales up and down applications to comply with SLO
TC_28/2054113	NebulOuS manages the execution of IoT data processing pipelines
TC_29/2054114	Secure Deployment of Applications via NebulOuS
TC_31/2054115	Network Isolation and Security Policy Compliance in Application Deployments
TC_35/2054117	Enforcement of Ingress and Egress Network Policies in Kubernetes
TC_37/2053092	Cloud provider selected based on user preferences

6. EVALUATION FRAMEWORK

The goal of Task 6.5: Evaluation framework and KPIs was to develop metrics and protocols to assess the performance of NebulOuS on each use case. To reach this goal, a document with the Guidelines for creating the evaluation framework was prepared and shared with all the use case partners. In this document the creation of an evaluation framework document was proposed for each use case, with the following structure:

- **Introduction:** Explanation of the use case and provide an overview of the purpose of the application deployed in the UC.
- **Software architecture:** In-depth description of the underlying software architecture of the application/applications running in the use case. It should include a description of the major components of the software system, their functionality, and their relationship with other components and systems. Diagrams or flowcharts to illustrate the relationships between components should be provided. The communication protocols used by the software components to communicate with each other should be highlighted, as well as what data is sent. Stakeholders should be identified and their interactions with the systems should be clearly described using diagrams or flowcharts. Finally, the deployment strategy should be also outlined.
- **Usage scenarios:** Quantifiable description of the expected usage scenarios for the solution in terms of number of user, number of requests/second, number of GB/second and similar metrics. A normal usage scenario and a high-demand usage scenario should be described.
- **Functional requirements assessment:** List of functional requirements identified for the UC and their means of verification.
- **Non-functional requirements assessment:** List of non-functional requirements identified for the UC and their means of verification.
- **Key Performance Indicators assessment:** List of Key Performance Indicators identified for the UC and their means of verification.

For each requirement identified in the sections devoted at functional and non-functional requirements assessment, as well as each of the KPIs identified, the guidelines proposed to adhere to the following structure:

- **Description:** An in-depth description of the requirement. Starting from the definition of the requirement provided in D2.1, it should be further refined to indicate how the requirement translates to the particular UC. The definition of the requirement should include the stakeholders involved and their expectations. To better understand the requirement, it was advised to provide a description on how this requirement is tackled in the current system.

- **Acceptance criteria:** A set of specific conditions that must be met for the requirement to be considered complete (acceptance criteria). These criteria should be measurable, meaning they should be clearly defined and quantifiable. Should not be open to interpretation. All involved stakeholders should agree with the provided acceptance criteria.

- **Means of verification:** A description of the procedure to be followed to verify that the acceptance criteria are met. This could include manual testing, automated testing, surveys, or other means. If testing was to be performed (either manual or automatic), a set of one or more test cases had to be developed for each requirement. Test cases had to be documented, clear, concise, and easy to understand. They had to cover all the required scenarios and be written in a way that can be easily executed and measured. Each test case had to clearly indicate how the tests will be performed and what steps need to be taken to execute the test cases.

Following the described guidelines, in collaboration with the different use case partners, an individual evaluation framework was prepared for each of the use cases.

7. USE CASES PLANNING

This section focuses on the six use cases that were selected to evaluate NebulOuS platform as they cover a wide spectrum of usage possibilities and provides a reliable indicator of what users demand and how a product like NebulOuS can be useful.

1. UC 1.1. Windmill Maintenance by Transformation Technology for Analysis (TTA). TTA deals with the inspection of wind turbines and the development of the platform for the automation of the damage detection and classification using AI mechanisms. Their UAV acquires photos and videos and implements AI-enabled processing to automatically detect turbine damage.
2. UC 1.2. Computer Vision for city maintenance by Ubiwhere. The company was founded in Portugal and is focused on research, development and innovation of software-based solutions in the areas of Smart Cities, Telco and Future Internet, and new technologies. Computer Vision uses photos captured from CCTV cameras, to detect and identify problems in the city, like damage in public buildings and infrastructures.
3. UC 2.1. Intra Logistics by MercaBarna. MercaBarna is the food hub of Barcelona that operates 24 hours a day with the aim of guaranteeing the supply of fresh food to the public. In this case a real-time traffic monitoring system based on a network of strategically located CCTV cameras recording the movements of vehicles is being developed.
4. UC 2.2. Last Mile by MercaBarna. Capturing and management of the data on trucks destination (city areas) to improve the last-mile goods distribution in the city. The system will be responsible for planning the optimal delivery route for each of the delivery vehicles and constantly monitor the fulfilment of the delivery route to detect any deviation.

5. UC 3. Precision Agriculture by Augmenta. Augmenta is a company that created a real-time VRA system that can fully automate farming agrochemical applications for farmers. In it, the execution of data preparation and map generation tasks aims at giving complete control to farmers over their agrochemical applications and help them to operate their farms sustainably while securing or even increasing their yield.
6. UC 4. Crisis Management by @FIRE. This nongovernmental organization is funding itself through donations and contributions. Their software solution allows management of the response crews, materials and equipment from a different teams present on the scene of the disaster. The data collected from the field by the sensor devices, creates a dynamic map of the situation and works as a base for AI-driven analysis.

In order to test and validate the requirements, all of the use cases needed to prepare certain files and models, that were next entered into the GUI and used in the deployment process. NebulOuS users define:

• **KubeVela** application manifest file for hyper-distributed application components and their deployments. KubeVela helps with deploying and managing applications across hybrid environments including Kubernetes, cloud or IoT devices. Every application deployment plan can be composed by multiple components with attachable operational behaviors (traits), deployment policy and workflow. KubeVela applications files uses four main abstractions: *Component*, that defines the delivery artefact or cloud service; *Trait*, a characteristic defined on a single component; *Policy*, that defines a strategy for a certain aspect of an application; *Workflow*, that provides tailored control flow and flexibility based on the standard Kubernetes delivery model. KubeVela provides automation for custom capabilities that may be attached to an application. More details and an example are presented in Deliverable 2.2.

• **Metrics and Requirements**- Metrics are any attributes for which a source of values (i.e., a sensor) can be specified. Metric is a measure that quantifies an application/system/environment attribute, which can be used to characterise the progress, performance, or status of an application aspect. Metric values (either raw or computed) are recorded as events and exchanged through event streams. Requirements specify the admissible values or value ranges for a service level that is measured by a service-level metric. Currently, a single kind of requirements is provided, Service-Level Objectives (SLOs). When the metric values fall within the defined SLO Violation range, the corresponding SLO is triggered and that indicates that the service is not functioning at an adequate performance level. This is described in detail inside Deliverable 2.2.

• **Optimisation goals**- These are preferences related to application deployment and scaling, through the use of an optimisation model based on AMPL[28]. The information in the AMPL file is used in expressions to calculate the utility value in the unit interval giving a value between zero and unity. The constraints define when a deployment configuration is valid, and if a new configuration should be found when the current configuration is infeasible.

Use case KPIs enable to validate the use cases themselves and effectively the underlying NebulOuS platform from the perspective of the use cases – instead of the platform creators as is the case with objectives KPIs defined in the DoA. Thus, there does not exist a clear mapping between objectives KPIs and use cases KPIs but some naturally correlate as the goal is to ensure the usefulness of the created software – in the eyes of all the stakeholders. The following tables help with understanding this subtle relation between the two kinds of KPIs. The first table relates appropriate objectives KPIs with use cases KPIs using groups of common relevant functionalities. The second recalls objectives KPIs' descriptions and assigns their groups if applicable (giving a relevant comment if not).

[28] https://ampl.com/

Table 5. 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 6. 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	K3

	and fog resource	
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 this very deliverable and D6.2 in the future.
KPI 6.2	use cases demonstration through the deployment of the platform	N/A - this is validated by the contents of this very deliverable and D6.2 in the future.
KPI 6.3	validation of NebulOuS innovative technologies in real-business settings	N/A - this is validated by the contents of this very deliverable and D6.2 in the future.

The list of KPIs and system requirements for all the use cases were presented in Deliverable 2.1. In this document we want to focus on each UC separately, with prepared functional and non-functional requirements and KPIs for each, and processes to verify them.
During the project, workshops between individual use case partners and technical partners were organized several times, to help them prepare necessary documentation and models.

UC 1.1 WINDMILL MAINTENANCE

Introduction

TTA has developed a windmill inspection solution that uses drones and AI algorithms to help detect the damages, assess their severity and, if necessary, execute maintenance operations to remediate them. With TTA solution, an operator moves to the base of a windmill and deploys a drone equipped with a high-resolution DSLR (Digital Single Lens Reflex) camera. The drone is connected to an on-board computing unit capable of acquiring, processing, and transmitting images to a computer located in the operator's vehicle (ground station) for further analysis. During the flight (limited to 30 minutes), the operator needs to cover all the wind blade faces to be able to quickly assess any major defects. The operation is repeated for all the blades. The objective is that the windmill can be restored to working condition promptly if no significant issues are found.

Every picture goes through initial processing on-board the drone (IoT step) and after that is send further to the Edge server for filtering, cropping etc. Finally, the data goes to Cloud for final processing. TTA solution can offload some of the workload to the Cloud, however, windmill farms are often situated in areas where cellular networks used for communication between the ground station and the cloud are unreliable or have limited bandwidth. To overcome these challenges, NebulOuS will allow establishing a Cloud-Edge computing continuum for deploying the data processing pipeline transparently, utilizing User Edge resources whenever possible while relying on the Cloud when necessary/possible. This will ensure a timely analysis of windmill data whilst limiting the amount of data sent to the cloud. Automated infrastructure scalability will allow the system to manage different workloads without manual input, optimizing both cost and performance, and ensure that resources are utilized efficiently, balancing quick processing needs with cost-effectiveness.
Sequence diagram below (Figure 12) shows sequence of events spanning over IoT, Edge and Cloud processing phase.

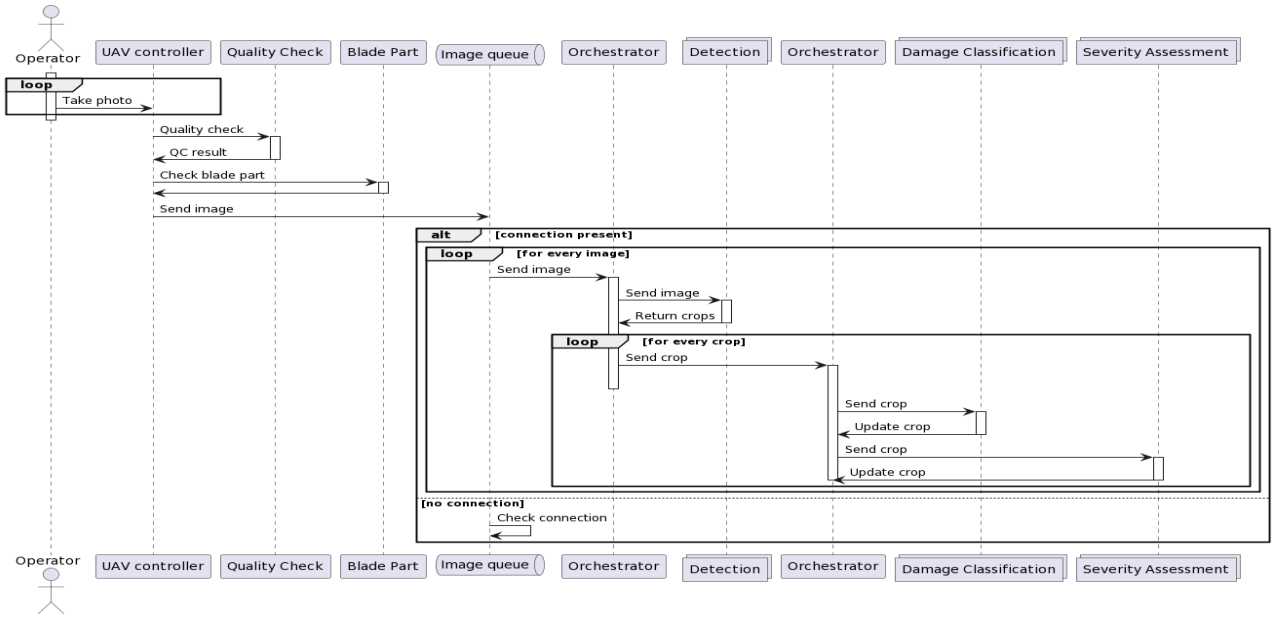


Figure 12. UC 1.1- Sequence of events split into IoT, Edge and Cloud.

The diagram below (Figure 13) shows how particular components are deployed within NebulOuS infrastructure (ignoring UAV/IoT deployment). Damage classification and severity assessment modules can be scaled infinitely. They are independent, stateless modules. The same is true for

damage detection, however in this case we consider 2 scaling options: either at both Edge and Cloud, or at the Edge only. Finally, orchestration modules have exactly one instance running at the Edge and one in the Cloud.

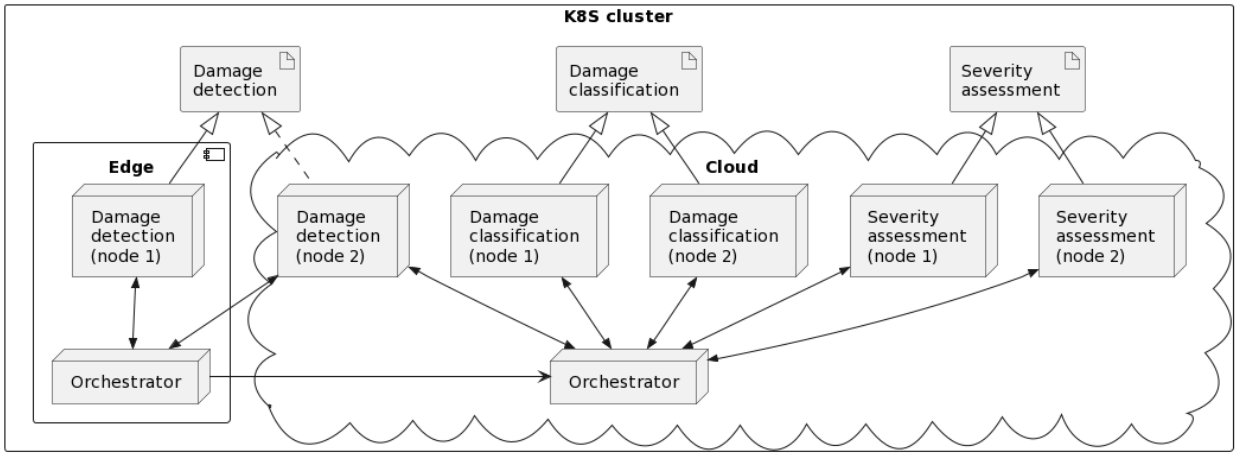


Figure 13. UC 1.1- Components deployed in NebulOuS.

Use case 1.1 architecture overview

The Windmill Maintenance pilot application architecture spans across Cloud and User Edge environments, utilizing various components and services for efficient data processing and analysis. Here is a detailed description of the architecture:

Cloud (AWS) Infrastructure

Kubernetes Cluster (Cloud)

1. **VM (Master)**
   - **Event Processing Manager:** Handles the initial processing of events in the cloud.
   - **WireGuard Client:** Provides secure VPN connectivity to ensure data transmission security between the Cloud and Edge environments.
   - **Optimizer Solver:** Optimizes resource allocation to improve system efficiency.
   - **k8s-gatekeeper:** Ensures policy enforcement and compliance within the Kubernetes cluster.
2. **VM (Worker)**
   - **Orchestrator:** Manages task orchestration, ensuring efficient workflow execution across components.
   - **MinIO:** Implements high-performance, object storage for data management.
   - **Severity Classification:** Classifies events based on their severity to prioritize responses.
   - **Damage Classification:** Analyzes and classifies damage-related data to aid in decision-making.
   - **WireGuard Client:** Maintains secure VPN connections for data transmission.
   - **Event Processing Manager**: Handles the processing of events within the worker nodes, ensuring efficient data handling and coordination.

- **KubeVela:** allows for the deployment of Open Application Model-based service graphs

User Edge Infrastructure

Kubernetes Cluster (Edge)

1. **Laptop (Worker)**
   - **Damage Detection:** Detects damage and anomalies in data received from the drone.
   - **Orchestrator:** Manages local orchestration tasks at the User Edge to ensure timely processing.
   - **MinIO:** Provides object storage at the edge for efficient data handling.
   - **Event Processing Agent:** Processes events locally to reduce latency and improve response times.
   - **WireGuard Client:** Ensures secure communication between the User Edge device and the Cloud infrastructure.

Network Connectivity

- **VPN:** The architecture uses a secure Virtual Private Network (VPN) facilitated by WireGuard clients to ensure safe and encrypted data transmission between the Cloud (AWS) and Edge environments.

Table 7. UC 1.1- Execution of the components of TTA application.

UC 1.1 (TTA)	Component	Execution (Cloud/Service Provider Edge/User Edge)	Cloud	User Edge
#1	Damage detection	Cloud/User Edge	AWS	Laptop(s)
#2	Damage classification	Cloud	AWS	
#3	Severity classification	Cloud	AWS	
#4	Orchestrator	Coud/User Edge	AWS	Laptop(s)
#5	MinIO	Coud/User Edge	AWS	Laptop(s)

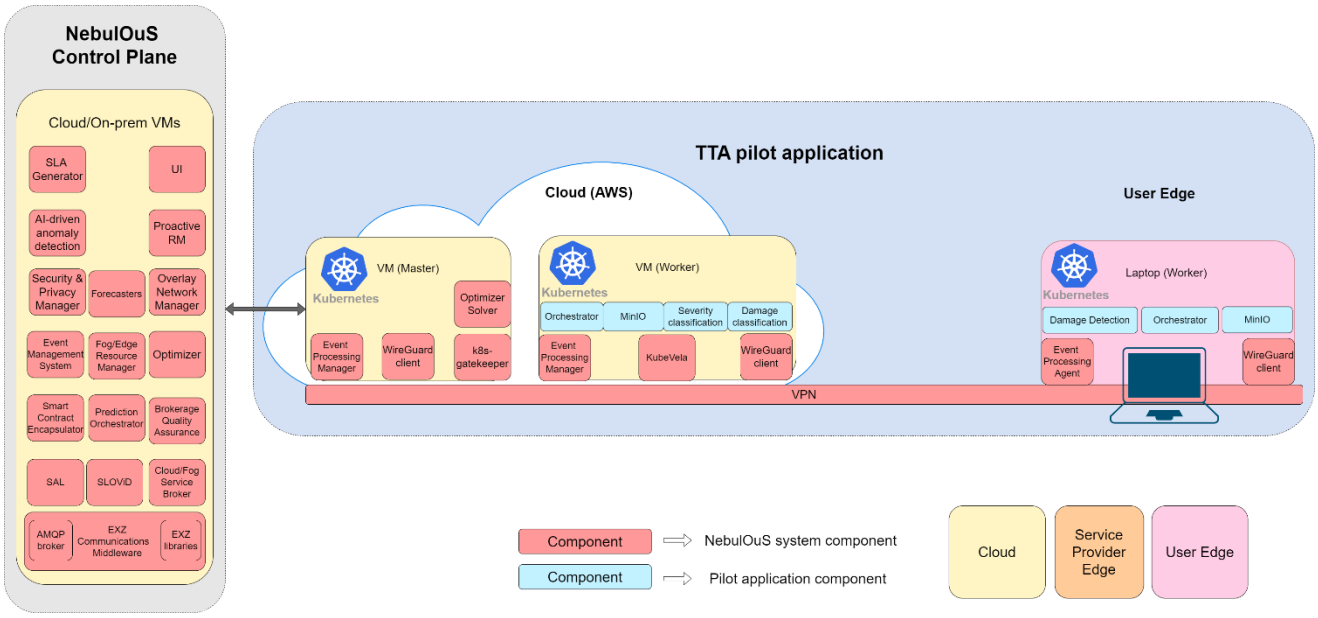


Figure 14. UC 1.1- Execution of the components of TTA application

Functional requirements assessment

- **F_01**

Description.

The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows. It ensures workloads are distributed on the appropriate computing continuum (just edge, just cloud, edge + cloud).

Acceptance criteria

- **Resilient Processing**: NebulOuS should start adjusting within 3-5 minutes, minimizing any delays.
- **Data Integrity:** In scenarios of connectivity loss, the system should ensure that no critical data is lost and that it can be processed effectively once the connection is re-established.
- **Adaptive Workload Management**: NebulOuS must effectively balance processing loads between edge and cloud resources, especially under high-demand conditions.
- **Responsive to Node Changes**: NebulOuS should dynamically adjust within 20 minutes of computing resource changes (node/device loss), ensuring seamless task continuation without interruption.

Means of verification

Conduct test *UC_1.1_F_01_TC1, UC1.1_F_01_TC2* and *UC1.1_F_01_TC3* as described in the Annex.

- **F_03**

Description

The NebulOuS platform will feature a graphical user interface (GUI) designed for efficient administration of deployed application components. This GUI will enable users to visually manage and monitor various aspects of the application, including the configuration, deployment across the computing continuum (edge, cloud). In the GUI we should be able to declare deployments with ease.

Acceptance criteria

- **Intuitive Navigation**: The GUI should be user-friendly, allowing easy navigation and helping the user to define all components, utility functions and metrics for application monitoring.
- **Real-Time Component Status**: The interface should display real-time statuses of all deployed components, including their health, activity, and any alerts or warnings.
- **Configuration Management**: Users should be able to configure and deploy components directly from the GUI.

Means of verification

Conduct test *UC1.1_F_03_TC1* as described in the Annex.

- **F_04**

Description

When determining where to deploy software components or containers, the system should factor in aspects such as network bandwidth, available storage space, power supply, and the physical positioning of the edge device.

Acceptance criteria

- **Efficient Resource Utilization**: The system must optimize the use of network bandwidth, storage, and power in deploying software components to ensure efficient operation.
- **Strategic Placement**: Software components should be deployed in locations that enhance performance and accessibility, based on factors such as proximity to end-users, network latency, and available resources.
- **Adaptive Deployment Strategy**: NebulOuS should dynamically adjust its deployment strategies based on real-time analysis of network conditions, resource availability, and workload requirements.

Means of verification

Conduct test *UC1.1_F_04_TC1* as described in the Annex.

- **F_06**

Description

The NebulOuS platform is equipped with advanced infrastructure monitoring capabilities. This feature allows for comprehensive oversight of the platform's operational components, including edge and cloud resources, network status, and the health of various deployed systems. It is designed to provide real-time insights into the performance and status of the infrastructure.

Acceptance criteria

- **Accuracy in Monitoring**: The monitoring system must accurately reflect the real-time status of all infrastructure components and applications, ensuring all metrics are consistently tracked without oversight, including Memory, CPU, GPU, Storage, Image quality, UAV response time, Model throughput, Model response time, Damages per image, Processing errors, Number of images in a queue, E2E throughput, Network throughput and Object detection response time.
- **Adaptive Reconfigurations**: Based on monitoring insights and anomaly detection, the platform should recommend and execute reconfigurations to enhance system performance and resilience, demonstrating its ability to adapt to changing conditions and demands

Means of verification
Conduct test *UC1.1_F_06_TC1, UC1.1_F_01_TC1* and *UC1.1_F_01_TC2* as described in the Annex.

- **F_07**

Description
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). It is able to start, stop components and react to monitoring alerts (if a component requires restarting for example).

Acceptance criteria
- **Lifecycle Management Efficiency**: The system must effectively manage the lifecycle of all application components, including the ability to start, stop, and restart components as required.
- **Cross-Environment Operation**: The lifecycle management should function seamlessly across different deployment environments (cloud, edge, IoT device).

Means of verification
Conduct test *UC1.1_F_07_TC1* and *UC1.1_F_07_TC2* as described in the Annex.

- **F_08**

Description
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).

Acceptance criteria
- **Support for multiple computing environments** (Cloud, User Edge, Service Provider Edge).
- **Integration** with Cloud, User Edge, Service Provider Edge computing resources for optimized deployment.

Means of verification
Conduct test *UC1.1_F_08_TC1* as described in the Annex.

Non-functional requirements assessment

- **NF_01**

Description
Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions (fast communication, internal processing to take decisions etc.).

Acceptance criteria
- **Speed of Decision Making**: The system must analyse and decide on resource allocation at a pace much faster than standard operational norms, ideally in a matter of seconds.
- **Execution Efficiency**: Once a decision is made, the execution of these resource allocation decisions should be equally swift, minimizing lag between decision and action.
- **Real-Time Responsiveness**: In dynamic environments where rapid changes occur, NebulOuS should adapt and respond immediately to new data or conditions, altering resource allocations as needed without delay.

Means of verification
NebulOuS aims to make decisions within 1 minute for *UC1.1_F_04_TC1*. However, it's crucial to note that while the decision-making process is within NebulOuS's control, the execution of these decisions may not be fully dependent on NebulOuS alone.

- **NF_02**

Description
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. Minimal downtime.

Acceptance criteria
- **Resilient Operational Capacity**: The platform should demonstrate the ability to continue key operational functions, such as data processing (ingestion, transformation, analysis) and communication tasks (transmission, routing), even when bandwidth is reduced to a specified threshold, such as 50% of typical capacity, without significant degradation in performance.
- **Rapid Recovery Post-Connectivity Restoration**: Upon the re-establishment of connectivity, the platform should quickly return to normal operational capacity, minimizing the recovery time.

Means of verification
Conduct tests *UC1.1_F_01_TC1* as described in the Annex.

- **NF_10**

Description
The NebulOuS platform will allow effective scaling of the cloud resources (in the edge/cloud computing).
This scaling feature is crucial for applications with variable processing needs, ensuring they always have access to the necessary computing power without overburdening the system.

Acceptance criteria
- **Responsiveness to Workload Variations**: NebulOuS should reacts to SLO violation events and trigger a new reconfiguration for each SLO violation and use the optimisation algorithms to determine appropriate VM instances/edge devices that will be used and also perform the ranking of the resources. This will be done in response to real-time changes in workload demand, acknowledging that the precise adjustment time may vary based on the capabilities and policies of the cloud provider. This includes both scaling up during increased demand and scaling down when demand decreases.
- **Resource Allocation Efficiency**: The system must improve resource usage, ensuring no underutilization or overallocation of resources after scaling, or triggering a new reconfiguration if necessary.

Means of verification
Conduct test *UC1.1_NF_10_TC1* as described in the Annex, where workload demands are varied and assess the platform's response in scaling cloud resources.

- **NF_11**

Description
The NebulOuS platform ensures data privacy when transferring data from one node to another. It incorporates standard security measures to prevent unauthorized access and ensure the confidentiality of data in transit.

Acceptance criteria
- **Integrity of Encryption:** All data transferred between nodes and NebulOuS system must be encrypted using industry-standard protocols, ensuring that the data cannot be read or altered during transit.
- **Verification of Secure Transmission**: Each data transfer must be accompanied by a verification process to confirm that the data has not been intercepted or tampered with.

Means of verification
Conduct tests *UC1.1_NF_11_TC1, UC1.1_NF_11_TC2, UC1.1_NF_11_TC3* as described in the Annex.

KPI's

- **KPI_UC1.1_1**

Description
Time to generate notification about one of data pipeline components failure or abnormal operation: <1 minute.

Baseline
n/a
Target
Less than 1 minute
Acceptance criteria
- The system must detect and notify any failure or abnormal behaviour in the data pipeline components within one minute of occurrence

Means of verification
Conduct a test by simulating failure or abnormal operation in a data pipeline component and measure the time taken for the system to generate an alert notification.

- **KPI_UC1.1_2**

Description
This KPI measures the time required to fully deploy the data processing pipeline within the NebulOuS platform. The goal is to achieve complete deployment in less than 15 minutes, ensuring rapid setup and initiation of data processing tasks.

Baseline
n/a
Target
Less than 15 minutes
Acceptance criteria
- The data processing pipeline must be fully deployed and operational within a 15-minute window.

Means of verification
Perform a timed test of the pipeline deployment process from initiation to full operational status, ensuring the total time does not exceed the 15-minute target.

- **KPI_UC1.1_3**

Description
Number of images processed during single asset inspection: >300. This KPI tracks the number of images processed during a single asset inspection. The target is to process over 300 images per inspection, ensuring thorough coverage and analysis.

Baseline
0 images processed
Target
300 images processed
Acceptance criteria
- Successfully process over 300 images in a single asset inspection session.

Means of verification
Conduct an asset inspection and count the number of images processed to verify if it exceeds 300.

- **KPI_UC1.1_4**

Description
Response time for quality assessment model: <2s. Measures the response time of the quality assessment model, aiming for less than 2 seconds per image. This ensures rapid feedback, essential for timely decision-making.

Baseline
n/a
Target
Less than 2 seconds (2s)
Acceptance criteria
- Each image's quality is assessed in under 2 seconds.

Means of verification:
Time the response of the quality assessment model across a set of images

- **KPI_UC1.1_5**

Description
Initial feedback on major damages: <5 min after inspection is completed. This KPI focuses on the time taken to provide initial feedback on major damages post-inspection, with a target of less than 5 minutes.

Baseline
1 day
Target
Less than 5 minutes (5 min)
Acceptance Criteria
- Initial damage report generated within 5 minutes after completing the inspection.

Means of Verification
Simulate an inspection and measure the time taken to generate the initial damage report.

- **KPI_UC1.1_6**

Description
Reduction in data transferred to the Cloud (defined as the ratio of the difference between the size of gathered data set and data sent to the Cloud, to the size of the full data set): 30%

Baseline
0% (No reduction)
Target
30% reduction
Acceptance criteria
- Achieve a 30% reduction in the volume of data transferred to the Cloud compared to the total data set size.

Means of verification
Compare the volume of data collected and the volume actually transferred to the Cloud over several inspection cycles.

- **KPI_UC1.1_7**

Description
Time to restore normal operation after connectivity is restored (in case of temporarily lost connection): <1 minute
Measures the time NebulOuS requires returning to its regular service performance following a brief interruption in connectivity, aiming for a recovery time of less than one minute.

Baseline
n/a
Target
Less than 1 minute (1 min)
Acceptance criteria
- System resumes normal operational state within 1 minute after connectivity is restored.

Means of verification
Simulate temporary loss of connectivity and measure the time taken for the system to resume normal operations.

UC 1.2 COMPUTER VISION FOR CITY MAINTENANCE

Introduction

Ubiwhere has developed a city maintenance solution that uses CCTVs, Smart Lampposts and computer vision algorithms to support the city livingness, by detecting infrastructure damages and traffic incidents. The CCTVs mounted on the Smart lampposts gather the images, and each camera sends the live video stream to a local processing node (Edge node), (an Nvidia Jetson Xavier) or to the cloud. The Service Provider Edge server, undertakes initial processing (detection, faces and car plates anonymisation - blurring, …), to convert images into metadata. After this, further processing and storage will occur either fully on the Edge or fully on Cloud resources.

To validate NebulOuS, this use case aims to provide a solution to automatic decision-making based on the seriousness and urgency of the detection. The goal is to use a full-Edge approach whenever the data to be processed is time-sensitive, i.e. when it relates to detections of an urgent nature. The full Cloud approach will be the default one to be used whenever there are no indications of urgency in data processing from a given detection, also to consider security restrictions.

Another scenario of the use case which uses to the autonomous decision-making capabilities of NebulOuS and relates to a possibility of a node being compromised (or any suspicion of that exists), involves the migration of the processing and storage to the nearby available node, and the first one is temporarily shut down.

Exact models or techniques used in every processing stage should always be the same since it makes no sense to change the algorithm in the middle of the process.

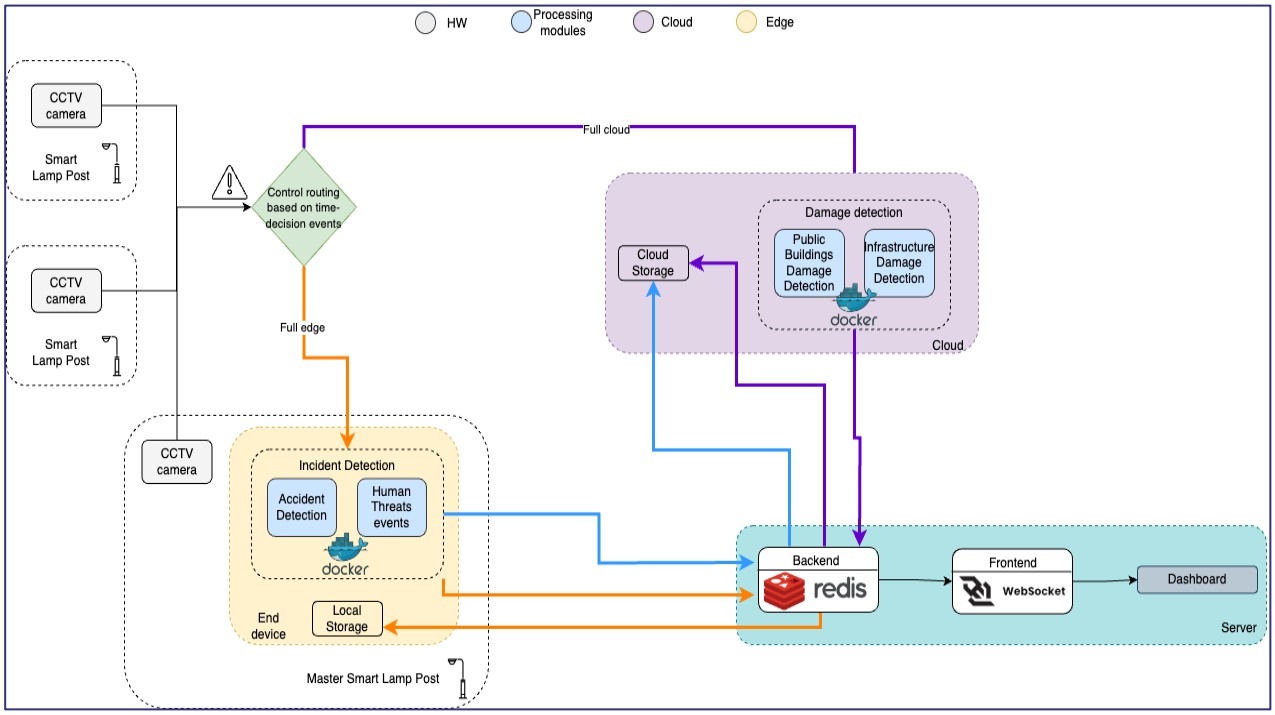


Figure 15. UC 1.2- Diagram of processes in Ubiwhere use case.

Use case 1.2 architecture overview

The Ubiwhere pilot application architecture is designed to leverage both Cloud and Service Provider Edge resources to enhance city maintenance through real-time data processing and decision-making. This architecture includes various components organized into Kubernetes Master Node, Kubernetes Worker Node, and Service Provider Edge Devices.

Cloud Infrastructure

1. **Kubernetes Master Node**:
    - **Event Processing Manager**: Manages and processes events triggered by different components in the system.
    - **Wireguard Client**: Provides secure VPN connectivity between different components and nodes.
    - **Optimiser Solver**: Optimizes the allocation of resources and processing tasks.
    - **K8s Gatekeeper**: Ensures policy enforcement and compliance within the Kubernetes cluster.

2. **Kubernetes Worker Node**:
    - **Public Buildings Damage Detection**: This component is designed to identify damages to public buildings using video streams and other sensory data. It can function in both Edge and Cloud environments, depending on the nature of the data. Urgent data is processed on the Service Provider Edge to facilitate immediate action, while non-urgent data can be sent to the Cloud for further analysis and storage.
    - **Infrastructure Detection**: Identifies and processes data related to city infrastructure issues. Similar to the Public Buildings Damage Detection, it operates in both Service Provider Edge and Cloud environments, optimizing processing based on the urgency of the data.
    - **Event Processing Manager**: Similar to the master node, it manages and processes events at the worker level.
    - **Wireguard Client**: Ensures secure VPN connections for data transmission.
    - **KubeVela:** allows for the deployment of Open Application Model-based service graphs

Service Provider Edge Infrastructure

1. **Nvidia Jetson Xavier**:
    - **Human Threats Events Detection**: This Service Provider Edge component processes data related to human threats and ensures a rapid response by processing it locally.
    - **Incident Detection**: This component detects various types of incidents, ranging from traffic accidents to public disturbances. By processing data locally on Nvidia Jetson Xavier devices, it ensures that actions can be taken immediately without the latency associated with cloud processing.
    - **Event Processing Agent**: Manages and processes events locally on the Service Provider Edge device.
    - **Wireguard Client**: Provides secure VPN connectivity for transmitting data securely.

Network Connectivity

- **VPN:** The architecture uses a secure Virtual Private Network (VPN) facilitated by WireGuard clients to ensure safe and encrypted data transmission between the Cloud (AWS) and Edge environments.

Table 8. UC 1.2- Execution of the components of Ubiwhere application.

UC 1.2 (Ubiwhere)	Component	Execution (Cloud/Service Provider Edge/User Edge)	Cloud	Service Provider Edge
#1	Incident Detection	Service Provider Edge		Nvidia Jetson xavier
#2	Human Threats events	Service Provider Edge		Nvidia Jetson xavier

#3	Public Buildings Damage detection	Service Provider Edge /Cloud	Ubiwhere data center	Nvidia Jetson xavier
#4	Infrastructure detection	Service Provider Edge /Cloud	Ubiwhere data center	Nvidia Jetson xavier

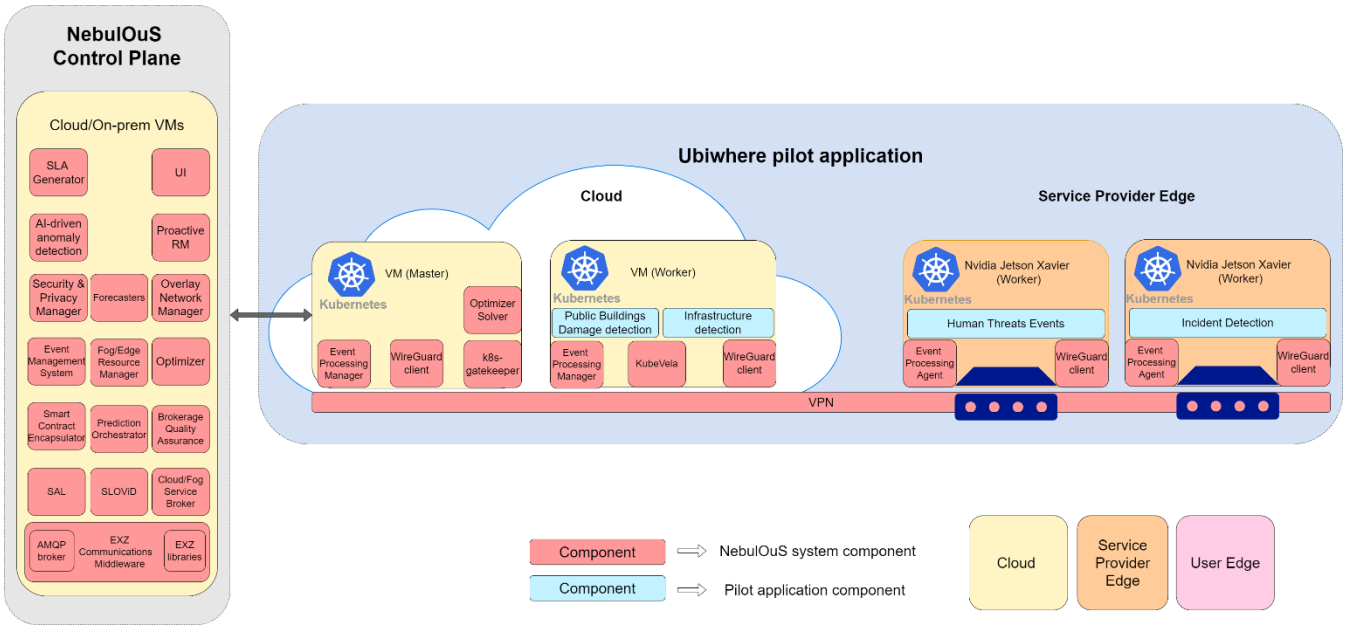


Figure 16. UC 1.2- Execution of the components of Ubiwhere application.

Functional requirements assessment

- **F_01**

Description

The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows. It ensures workloads are distributed on the appropriate Computing Continuum (just edge, just cloud, edge + cloud).
The Service Provider Edge node, a Jetson Xavier, receives the video stream from every CCTV camera. It processes the Computer Vision algorithm and stores the output in the Cloud. NebulOuS should, however, launch the application on the Cloud if the Edge node is compromised (e.g. lacks resources or has been corrupted).

Acceptance criteria

- NebulOuS will be notified when a node fails, and the CCTV camera video streams will be transmitted to the Cloud for processing.

Means of verification

Conduct the test *UC1.2_F_01_TC1* as described in the Annex.

Non-functional requirements assessment

- **NF_01**

Description

Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions (fast communication, internal processing to take decisions etc.).
When the damage detection process needs to run on the cloud, NebulOuS should detect it and take appropriate actions.

Acceptance criteria

- NebulOuS detects that the Edge node has failed and re-assigns the damage detection to the Cloud.

Means of verification

Conduct the test *UC1.2_F_01_TC1* and *UC1.2_NF_10_TC1* as described in the Annex.

- **NF_10**

Description

The NebulOuS platform will allow effective scaling of the Cloud resources (in the Edge or using Cloud Computing).

Acceptance criteria

- NebulOuS can detect when the workload demands an increase of the processing capacity, dynamically deciding how many of the available Edge resources (max 3) are needed to distribute the workload.

Means of verification

Conduct *UC1.2_NF_10_TC1* as described in the Annex.

- **NF_11**

Description

The NebulOuS platform ensures data privacy when transferring data from one node to another.
Since the CCTV cameras are filming roads, sensitive data can be recorded such as people or license plates NebulOuS should guarantee that any communication between the Edge devices and the Cloud infrastructure is secure.

Acceptance criteria

- Communication between application components should be encrypted.

Means of verification

Conduct *UC1.2_NF_11_TC1* as described in the Annex.

KPI's

- **KPI_UC1.2_1**

Description

Vertical scaling: reducing mean time to detect an object in a video frame by 50%.
The system should be able to detect that more computation resources are needed to keep with the application SLOs.

Baseline

2 min

Target

1 min

Acceptance criteria

- NebulOuS decides on a new deployment topology that uses Cloud resources, when metrics that indicate the need for vertical scaling are generated.

Means of verification

Conduct *KPI_UC1.2_1_TC1* as described in the Annex.

- **KPI_UC1.2_2**

Description

Horizontal scaling: decreasing the mean time to recover after a node corruption by 50%
The system should be able to detect that a node has been compromised (e.g.: the node disappears) and re-configure the application to correct the problem.

Baseline

n/a

Target

1 min

Acceptance criteria

- NebulOuS decide on an application re-configuration when a node is compromised.

Means of verification

This KPI will be met as long as requirement *NF_01* passes.

- **KPI_UC1.2_3**

Description

Reducing data transfer from the Edge to the Cloud by 70%

Baseline

0

Target

1 s

Acceptance criteria

- Reducing data transfer from the Edge to the Cloud by 70%

Means of verification

Comparison between the data transferred to the Cloud in a Cloud only-setup and the data transferred in a hybrid Edge/Cloud setup.

- **KPI_UC1.2_4**

Description

Reduce number of necessary Edge nodes by 10%

Baseline

4

Target

3

Acceptance criteria

- Reduce number of necessary Edge nodes by 10%

Means of verification

To reduce the number of nodes deployed, a test will be carried out to validate if the number of Edge nodes diminished implies the malfunction of the use case. The test to be carried out will be to deploy the use case with the initial number of Edge nodes and then perform a reduction of 10% in the number of available Edge nodes.

- **KPI_UC1.2_5**

Description

Reduce number of necessary Cloud resources by 50%

Baseline
2
Target
1
Acceptance criteria

- Reduce number of necessary Cloud resources by 50%

Means of verification
To reduce the number of Cloud resources deployed, a test will be carried out to validate if a reduction in Cloud resources is associated to poor performance from the side of the use case. The test to be carried out will initially deploy the use case with the higher number of Cloud resources and afterwards perform the reduction of 50%.

UC 2.1 MERCABARNA INTRA-LOGISTICS

Introduction

Mercabarna (MrB) is a small living city, where many vehicles of all types (from big trucks to vans, cars or forklifts) work together to carry on Mercabarna's customer's operations. An effective management of internal traffic is of utmost importance to enable smooth operations and prevent jams or bottlenecks that would delay the lifecycle of the entire Mercabarna. With this objective, a real-time traffic monitoring system for intra-logistics operations will be developed to identify, in real time, the vehicles entering/leaving each one of the streets, and their typology (van, truck, etc....). On top of this system, a Decision Support System will provide recommendations for optimal roads management, and a what-if tool will enable the ability to test and validate different scenarios.

To monitor traffic on Mercabarna's roads, a network of CCTV cameras will be installed (Figure 17). These cameras will be placed in the road intersections (two for intersection) on the roof of each building. Those cameras will record the traffic in both directions and stream the video in the internal network (RSTVP).

Figure 17. UC 2.1- Camera network top view (left), street view (right)

From a software point of view, the solution can be divided into two interlaced applications (Figure 18).

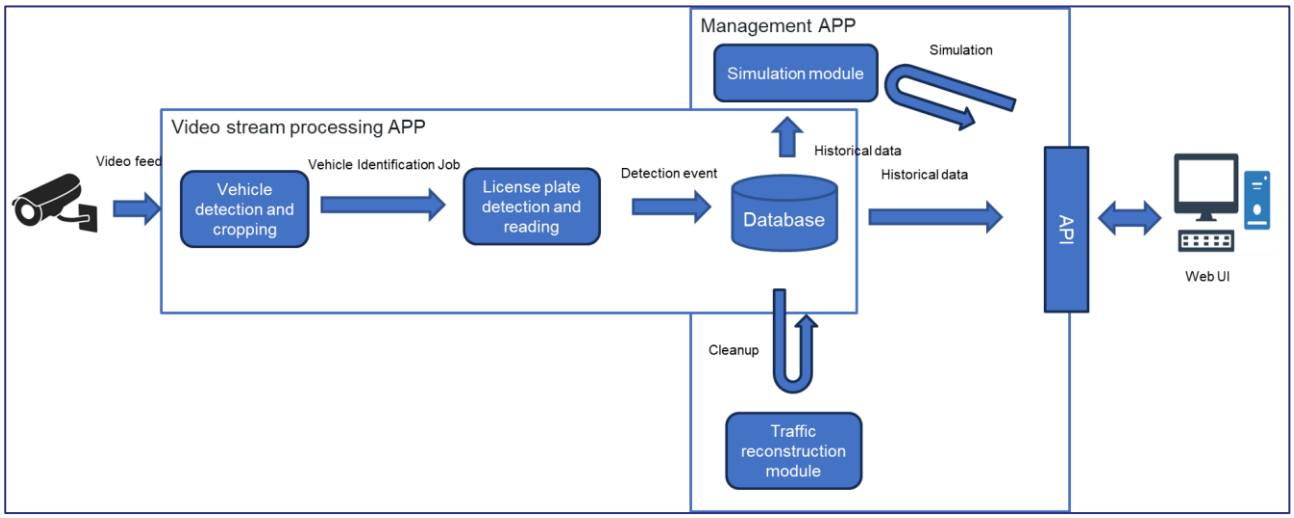


Figure 18. UC 2.1- Software architecture overview

- **The video stream processing APP:** Responsible for analyzing video feed from the installed cameras to detect passing vehicles, read their license plate and their direction.
- **The management APP:** Responsible for consolidating events generated by the "Video stream processing APP", presenting the results via a web UI and permitting the user to execute simulations on management policies over historical data.

Use case 2.1 architecture overview

The architecture of the application includes components operating at the User Edge and in the Cloud.

Cloud Infrastructure- AWS

1. **Master Node (Cloud)**
   - **Master Optimizer Solver**: Central component responsible for generating optimal traffic management strategies based on real-time and historical data.
   - **Event Processing Manager**: Manages and processes events generated by the video stream processing app and other system components.
   - **WireGuard Client**: Ensures secure communication between various system components and the master node.
   - **K8s Gatekeeper**: Ensures policy enforcement and compliance within the Kubernetes cluster.

2. **VM (Worker) Nodes (Cloud)**
   - **Web UI**: User interface for presenting data and managing the system, allowing users to access real-time analysis results, view traffic history, and run traffic management simulations.
   - **NebulOuS IoT message broker**: Manages communication between different parts of the system, ensuring reliable message exchange.

- **FileServer**: Stores cropped images and other data generated by the system, allowing access for further analysis and reporting.
- **Database**: Stores all collected and processed information about vehicles, their movement, license plates, and video analysis results, providing quick access to both historical and real-time data.
- **Traffic Reconstruction Module**: Analyzes data from cameras and reconstructs traffic flow, enabling better understanding of traffic patterns and identifying potential issues.
- **Event Processing Manager**: Handles the processing of events within the worker nodes, ensuring efficient data handling and coordination.
- **WireGuard Client**: Ensures secure communication between the worker nodes and other components of the system.
- **KubeVela:** allows for the deployment of Open Application Model-based service graphs

User Edge Infrastructure

1. **Raspberry Pi (Edge) Nodes** There are three main functionalities that can be deployed on the Raspberry Pi devices:
   - **Vehicle Detection and Cropping**: Detects vehicles in the video stream and crops images of the vehicles.
   - **License Plate Detection and Reading**: After detecting vehicles, reads and identifies license plates using advanced algorithms.
   - **Event Processing Agent**: Manages the initial processing of video data and events at the User Edge, sending relevant information to cloud components.
   - **Simulation Module (planned)**: Will enable testing of different traffic management policies based on historical and real-time data, supporting decisions for traffic optimization.
   - **Wireguard Client**: Provides secure VPN connectivity for transmitting data securely.

Network Connectivity

- **VPN:** The architecture uses a secure Virtual Private Network (VPN) facilitated by WireGuard clients to ensure safe and encrypted data transmission between the Cloud (AWS) and Edge environments.

Table 9. UC 2.1- Execution of the components of Mercabarna Intralogistics application.

UC 2.1 (Mercabarna) Intra logistic	Component	Execution (Cloud/Service Provider Edge/User Edge)	Cloud	User Edge
#1	Vehicle detection and cropping	User Edge		Raspberry Pi devices (one for camera)
#2	NebulOuS IoT message broker	Cloud	AWS	

#3	License plate detection and reading	Cloud/ User Edge	AWS	Raspberry Pi devices
#4	Database	Cloud	AWS	
#5	Traffic reconstruction module	Cloud	AWS	
#6	Web UI	Cloud	AWS	
#7	Simulation module (not developed yet)	Cloud/User Edge	AWS	Raspberry Pi devices

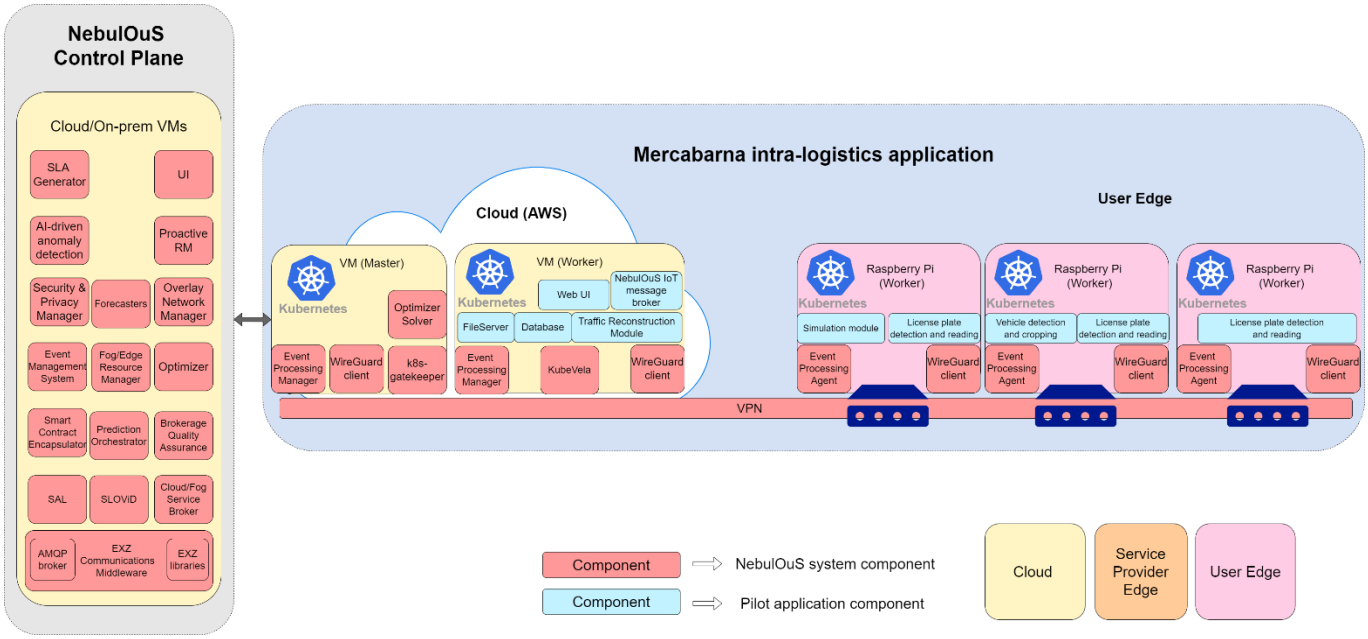


Figure 19. UC 2.1- Execution of the components of Mercabarna Intralogistics application.

Functional requirements assessment

- **F_01**

Description

The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows. It ensures workloads are distributed on the appropriate Computing Continuum (just edge, just cloud, edge + cloud).

For each of the video streams (one per camera) the application requires a running "Vehicle detection and cropping" that connects to the camera video stream and processes the frames. These processes can reside in any worker nodes inside Mercabarna's facilities. In principle it should be processed by the closest computing node (installed in the same building as the camera) but, if a node fails, NebulOuS should deploy it in any other node.

Acceptance criteria

- NebulOuS reacts to a worker node failure (suddenly disappears) and diverts the workload to other workers to guarantee that the video streams of each camera are processed by one of the nodes.

Means of verification
Conduct test *UC2.1_F_01_TC1* as described in the Annex.

- **F_02**

Description
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. The system shall provide events publish and subscribe-based communication "bus" for the interaction of components.
This case study uses the NebulOuS pub/sub mechanism between the modules "Vehicle detection and cropping" and "License plate detection and reading" to communicate *"Vehicle identification jobs"*.

Acceptance criteria
- NebulOuS can orchestrate the communication between two instances of "Vehicle detection and cropping" four instances of "License plate detection and reading" module and the Management APP.

Means of verification
Conduct test *UC2.1_F_02_TC1* as described in the Annex.

- **F_04**

Description
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.

In this use case, the location of the "Vehicle detection and cropping" processes that analyze camera video streams should be as close as possible (network wise) to the source camera to reduce as much as possible the network congestion.

Acceptance criteria
- If the closest node to a camera has enough free computation resources to allocate the "Vehicle detection and cropping" processes for the camera, NebulOuS should use that node.

Means of verification
Conduct test *UC2.1_F_04_TC1* as described in the Annex.

- **F_05**

Description
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.

NebulOuS should constantly check the status of the "Vehicle detection and cropping" processes deployed and guarantee that each is processing its assigned stream at 10fps. If this requirement is not meet, NebulOuS should try to find a better deploy organization.

Acceptance criteria
If a "Vehicle detection and cropping" processes is processing the video stream at less than 10fps because many instances of the process are running in the same worker node and there are other

worker nodes that can assume the workload (less resource clogged), NebulOuS should re-deploy some workload to other worker nodes to warrantee that the video streams are processed at 10fps.

Means of verification
Conduct test *UC2.1_F_05_TC1* as described in the Annex.

- **F_09**

Description
The NebulOuS platform provides near-real-time communication between application components.

Acceptance criteria
- Events generated by the "License plate detection and reading" component are received by the *Management APP* in less than 1 second.

Means of verification
Conduct test *UC2.1_F_02_TC1*, validate that event generated by "License plate detection and reading" component are received by the *Management APP* in less than 1 second.

Non-functional requirements assessment

- **NF_01**

Description
Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions (fast communication, internal processing to take decisions etc.).
In case any of the license plate reading processes needs to be allocated/re-allocated. NebulOuS should detect it and take appropriate actions.

Acceptance criteria
- If a video stream is not being processed by a license plate reading process (it was not configured or the worker executing the instance fails), NebulOuS should detect it and find a new deployment topology using any of the available worker nodes.
- If a license plate reading process is processing frames slower than 5fps due to the worker node being saturated by many processes, NebulOuS should detect it and find a new deployment topology.

Means of verification
Conduct test *UC2.1_F_01_TC1, UC2.1_F_04_TC1* and *UC2.1_F_05_TC1* as described in the Annex.

- **NF_03**

Description
The platform provides secure access to the resources (clouds, edge resources etc.) (i.e. provides user Identification/Authentication, User/application manager roles etc.)
Only admin can login to define computation resources managed by Nebulous.

Acceptance criteria
- Only registered users can onboard new Edge devices to be used as part of the application deployment.

Means of verification
Conduct test *UC2.1_NF_03_TC1* as described in the Annex.

- **NF_04**

Description

NebulOuS platform should be able to deploy applications in a Raspberry Pi 3 Model B+ or similar ARM architecture.

NebulOuS should be able to use Raspberry Pi's as worker nodes to deploy the different solution modules.

Acceptance criteria

- Mercabarna's intra-logistic management application can be deployed by NebulOuS in provisioned Raspberry Pi 3'.

Means of verification

Raspberry Pi 3 nodes can be registered in NebulOuS.

After configuring the application using NebulOuS, NebulOuS can deploy the license plate reading process components in the RPIs. Since the RPIs are the only computing units available inside the Mercabarna network for executing the license plate reading process, this will be as long as *UC2.1_F_01_TC1* passes.

- **NF_05**

Description

Data transferred between edge nodes and between edge nodes and clouds should be secured.

Acceptance criteria

- NebulOuS pub/sub mechanism cannot be accessible from outside the applications components.

Means of verification

Conduct test *UC2.1_NF_05_TC1* as described in the Annex.

- **NF_09**

Description

The NebulOuS platform provides near-real-time processing of big data.

Since Mercabarna's case study will continuously generate information over the video streaming coming from the cameras, NebulOuS should be prepared to manage/store big volumes of data.

Acceptance criteria

- NebulOuS should guarantee that it is capable of managing the resources generate by 20 cameras deployed over the industrial area of Mercabarna's and maintain an historical database for at least 4 years.

Means of verification

Conduct test *UC2.1_NF_09_TC1* as described in the Annex.

- **NF_10**

Description

The NebulOuS platform will allow effective scaling of the cloud resources (in the edge/cloud computing).

If vehicle identification jobs are not processes in less than 10 seconds because computing nodes running "License plate detection and reading" instances are congested, NebulOuS should deploy

"License plate detection and reading" instances in Cloud nodes to guarantee that the instances of Vehicle identification jobs are processed in less than 10 seconds.

Acceptance criteria

- NebulOuS should guarantee that a host reinforcement/substitute is found after the problem is detected.

Means of verification

Conduct the following test: *UC2.1_NF_10_TC1* as described in the Annex.

- **NF_11**

Description

The NebulOuS platform ensures data privacy when transferring data from one node to another.

Acceptance criteria

- NebulOuS application internal pub/sub mechanism cannot be accessible from outside the applications components.

Means of verification

Conduct test *UC2.1_NF_11_TC1* as described in the Annex.

KPI's

- **KPI_UC2.1_1**

Description

Reducing data transfer from the Edge to the Cloud.
In this case study the data transfer is reduced in the first step of the process and all the video generated by the cameras is processed locally to avoid transferring it to any Cloud device. Furthermore, the cameras stream is processed by the closest local node.

Baseline

448Mbps of video sent to the cloud (estimated)

Target

0Mbps of video sent to the Cloud.

Acceptance criteria

- The "Vehicle identification jobs" are processed locally by Edge devices and is not sent to any Cloud device unless Edge infrastructure is not capable of meeting the SLO requirement.

Means of verification

This KPI will be met as long as requirement *NF_10* passes.

- **KPI_UC2.1_2**

Description

Reduce the number of necessary cloud resources.
In this case of study, the cloud services usage is reduced by using local working nodes to run the "License plate detection and reading" module.

Baseline

n/a

Target

Video streams processed on Edge.

Acceptance criteria

- The solution processes all "Vehicle identification jobs" on Edge devices.

Means of verification
This KPI will be met as long as requirement *NF_10* passes.

- **KPI_UC2.1_3**

Description
Improve road usage patterns awareness.
Baseline
Qualitative knowledge from managers
Target
Historical quantitative data available
Acceptance criteria
- The solution stores historical data from the traffic on Mercabarna and provide the user heatmaps of the traffic patterns.

Means of verification
User is able to visualize historical traffic data from Mercabarna using the provided GUI.

- **KPI_UC2.1_4**

Description
Less than 5 minutes to identify potential traffic issues.
Baseline
15 min
Target
5 min
Acceptance criteria
- The application generates a heatmap of the traffic location every minute, these heatmaps are stored in the database and can be consulted from the interface by the user.
- If a section is marked with potential traffic issues, the application checks if it is the second consecutive time that happens and generates an alert if it's necessary.

Means of verification
Manually recreating a scenario where one road is congested for more than 2 minutes the application should identify it and notify the user about the anomalous behaviour of the road in less than 5 minutes.

UC 2.2 MERCABARNA- LAST MILE USE CASE

Introduction
More than 7,000 small trucks leave MrB's facilities every day and go to the city and its premises to distribute goods. This puts stress on the over-congested streets of Barcelona. This case study will focus on developing a real-time system to monitor transport vehicles and propose optimal routes for efficient goods delivery. Routing will be based on real time data from traffic status. The proposed plan will be updated on-line to adapt to changes.

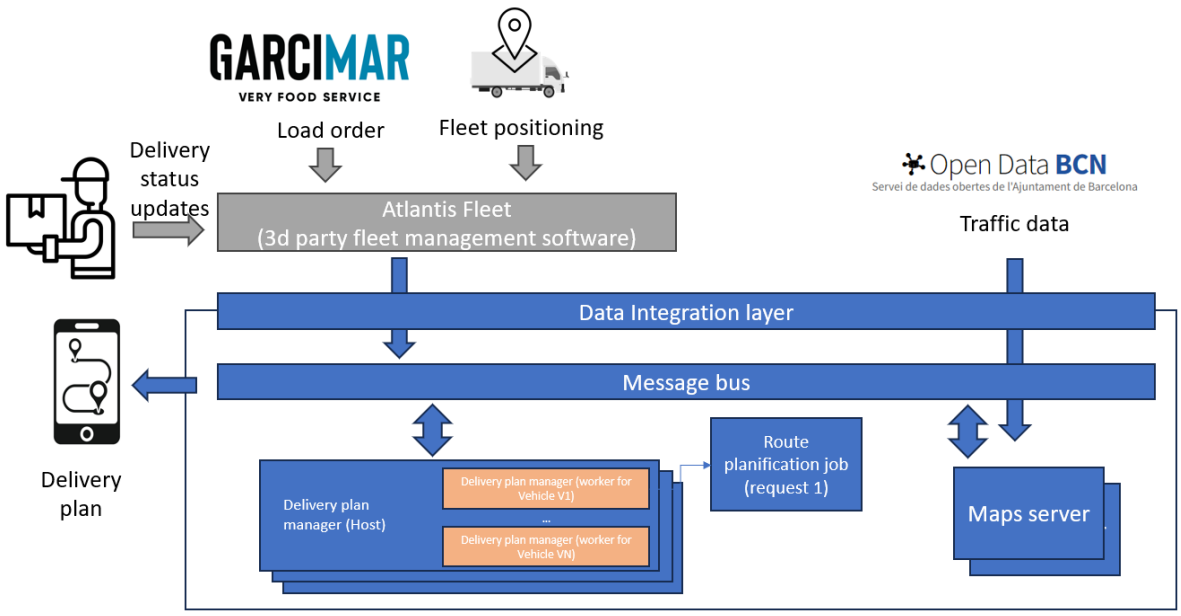


Figure 20. UC 2.2- Application components view

As shown in Figure 20, the proposed software architecture is composed of the following components:

- **Data integration layer:** Responsible for connecting the different information sources.
- **Message bus:** For articulating the communication between application components.
- **App server:** Serves the web application files (HTML+CSS+JS).
- **Maps server:** Given a list of pairs of geographic coordinates, calculates the drive time between each of these pairs based on the most up to data available traffic data.
- **Delivery plan manager (host):** Responsible for hosting the execution of many delivery plan manager (workers) as separated threads.
- **Delivery plan manager (worker):** Responsible for watching for changes in the status of the delivery plan for a vehicle and its position and triggering a route recalculation whenever necessary. There are three reasons to tigger such a recalculation:
  - There isn't a delivery plan for the vehicle.
  - The list of load orders for a vehicle has changed.
  - The current delivery plan is not feasible.

  Under any of these conditions, the "delivery plan manager" module will gather all necessary information (vehicle position, load orders and routes between locations) and invoke the route calculation function.
- **Route planification:** A serverless function that accepts a route calculation request containing the delivery plan of a vehicle, its current position and driving times between possible route points. With all this data, the route calculation job is responsible for providing an optimal sorting of the load orders.

Use case 2.2 architecture overview

The Mercabarna Last Mile application architecture leverages a combination of Cloud and User Edge computing resources to optimize the delivery of goods within Barcelona. Below is a detailed description of the various components involved in the system, including the two edge workers.

Cloud Infrastructure– AWS

1. **VM Master**
   - **Optimiser Solver:** Responsible for calculating the optimal routes for delivery vehicles based on real-time traffic data and delivery requirements.
   - **Event Processing Manager:** Manages and processes events such as changes in traffic conditions, delivery orders, and vehicle statuses to trigger necessary actions like route recalculations.
   - **WireGuard Client:** Ensures secure VPN connections between Cloud resources and User Edge devices.
   - **K8s Gatekeeper**: Ensures policy enforcement and compliance within the Kubernetes cluster.

2. **VM Worker**
   - **UI:** Serves the user interface of the application, providing a web-based dashboard for monitoring and managing delivery operations.
   - **Maps Server:** Provides map services, including drive time calculations and route mapping, based on real-time traffic data.
   - **Data Integration Layer:** Integrates data from various sources such as traffic updates, vehicle locations, and delivery orders.
   - **Message Bus (NebulOuS IoT message broker)**: Provides communication between application components.
   - **Event Processing Manager:** Similar to the master node, it processes events at the worker level to ensure local operations are optimized.
   - **WireGuard Client:** Ensures secure VPN connections between Cloud resources and User Edge devices.
   - **KubeVela:** allows for the deployment of Open Application Model-based service graphs

User Edge Infrastructure

1. **Raspberry Pi (Worker)**
   - **Delivery Plan Manager:** Monitors the status of delivery plans and vehicle positions, triggering route recalculations when necessary.
   - **Message Bus (NebulOuS IoT message broker):** Provides communication between application components.
   - **Event Processing Agent:** Handles local events and communicates with the central event processing manager to synchronize operations.
   - **WireGuard Client:** Maintains secure VPN connections between User Edge devices and Cloud resources.
2. **Raspberry Pi (Worker)**
   - **Route Planification Module:** Performs the actual route calculation, determining the optimal order of delivery stops based on current conditions.

- **Event Processing Agent:**Handles local events and communicates with the central event processing manager to synchronize operations.
- **WireGuard Client:** Maintains secure VPN connections between User Edge devices and Cloud resources.

Network Connectivity

- **VPN:** The architecture uses a secure Virtual Private Network (VPN) facilitated by WireGuard clients to ensure safe and encrypted data transmission between the Cloud (AWS) and Edge environments.

Table 10. UC 2.2- Execution of the components of Mercabarna Last Mile application.

UC 2.2 (Mercabarna) Last mile	Component	Execution (Cloud/Service Provider Edge/User Edge)	Cloud	User Edge
#1	Data integration layer	Cloud	AWS	
#2	UI	Cloud	AWS	
#3	Maps server	Cloud	AWS	
#4	Delivery plan manager	Cloud/User Edge	AWS	Raspberry Pi devices
#5	Route planification module	Cloud/User Edge	AWS	Raspberry Pi devices
#6	Message bus (NebulOuS IoT message broker)	Cloud/User Edge	AWS	Raspberry Pi devices

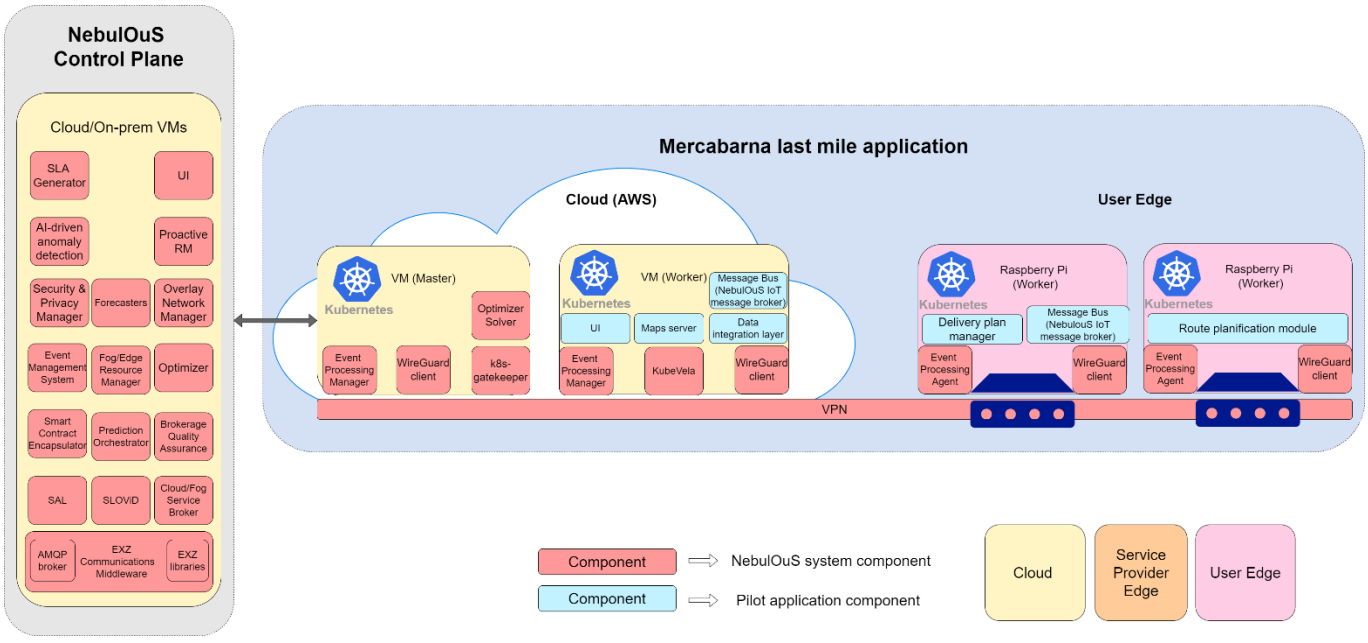


Figure 21. UC 2.2- Execution of the components of Mercabarna Last Mile application.

Functional requirements assessment

- **F_01**

Description
The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows. It ensures workloads are distributed on the appropriate computing continuum (just edge, just cloud, edge + cloud)

Acceptance criteria
- If computation resources are available on the edge nodes, NebulOuS will use them. Under high load scenarios, workload can be executed in the cloud.

Means of verification
Conduct the test *UC2.2_F_01_TC1* as described in the Annex.

- **F_02**

Description
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. The system shall provide events publish and subscribe based communication "bus" for the interaction of components.
The proposed application architecture makes use of the NebulOuS pub/sub system to share information between components. One example is the map server. Delivery plan manager (host) sends routing requests to the map server using the pub/sub mechanism offered by NebulOuS.

Acceptance criteria
- The application works correctly when more than one node is necessary to cope with the workload (at least two).

Means of verification
Conduct the test *UC2.2_F_01_TC1* as described in the Annex.

- **F_04**

Description
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.

- **F_05**

Description
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.
It is the responsibility of NebulOuS to scale in/out – up/down the map server to guarantee that (on average) a request does not take more than 5 seconds to be processed.

Acceptance criteria
- 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.

Means of verification
Conduct the test: *UC2.2_F_01_TC1* as described in the Annex.

- **F_09**

Description
The NebulOuS platform provides near-real-time communication between application components.

Acceptance criteria
- NebulOuS should ensure that GPS updates published to the internal application message broker are available for being consumed by the delivery plan manager (worker) instance in less than 3s.

Means of verification
Conduct the test: *UC2.2_F_09_TC1* as described in the Annex.

- **F_10**

Description
NebulOuS offers a mechanism to partition streaming analysis jobs between a variable number of workers (similar to Spark partitioning)

Acceptance criteria
- At least two instances of delivery plan manager (host) can exist, and the workload (vehicles to monitor) divided among them.

Means of verification
Conduct the test *UC2.2_F_10_TC1* as described in the Annex.

- **F_11**

Description
NebulOuS should offer a mechanism to deploy and invoke serverless computation functions.

Route optimization will be implemented as a serverless computation function. For each route optimization request, a new instance of this function will be triggered and NebulOuS should handle its deployment.

Acceptance criteria
- NebulOuS is capable of allocating resources and executing the serverless function that implements the route optimization module.

Means of verification
Conduct the test: *UC2.2_F_11_TC1* as described in the Annex.

Non-functional requirements assessment

- **NF_01**

Description
Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions (fast communication, internal processing to take decisions etc.).

Acceptance criteria

- In case of the map server needing to scale in/out up/down, NebulOuS should detect it and take appropriate actions.

Means of verification

In test *UC2.2_F_01_TC1*, NebulOuS quickly started the deployment re-adaptation and decided on the new deployment topology, after the SLO violation is detected.

- **NF_03**

Description

The platform provides secure access to the resources (clouds, edge resources etc.) (i.e. provides user Identification/Authentication, User/application manager roles etc.)

Only admin can login to define computation resources managed by Nebulous.

Acceptance criteria

- For this case study, only a Mercabarna admin can configure and manage the application.

Means of verification

Conduct the test: *UC2.2_NF_03_TC1* as described in the Annex.

- **NF_05**

Description

Data transferred between edge nodes and between edge nodes and clouds should be secured.

Acceptance criteria

- Connecting with the IoT pub/sub mechanism should not be possible without appropriate credentials.

Means of verification

Conduct test *UC2.2_NF_05_TC1* as described in the Annex.

- **NF_06**

Description

The system provides elasticity capabilities along with the cloud continuum, so HW resources are provisioned or freed depending on the workload

Acceptance criteria

- The map server component is scaled up/out when workload spikes. It is scaled down/in when workload goes down.

Means of verification

Execute test *UC2.2_F_05_TC1*. During the test, map server component should scale up/out. When the test concludes and the generator of requests stops, NebulOuS should scale down/in the map server component.

- **NF_09**

Description

The NebulOuS platform provides near-real-time processing of big data.
Since Mercabarna's case study will continuously be generating information over the delivery routes generated for each vehicle, NebulOuS should be prepared to manage/store big volumes of data.

Acceptance criteria

- NebulOuS PUB/SUB system should be able to 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.

Means of verification

Conduct test *UC2.2_NF_09_TC1* as described in the Annex.

- **NF_10**

Description

The NebulOuS platform will allow effective scaling of the cloud resources (in the edge/cloud computing).

If necessary, NebulOuS should be able to find a new host (Azure/Amazon/AWS/etc.) for the modules deployed on the solution if the original ones are not capable of processing the current workload.

Acceptance criteria

- This requirement will be considered met if requirement "F_01" is met.

Means of verification

Conduct test *UC2.2_F_01_TC1* as described in the Annex.

- **NF_11**

Description

The NebulOuS platform ensures data privacy when transferring data from one node to another.

Acceptance criteria

Using WireShark or similar software to listen to the communications between components using the IoT pub/sub mechanism, we should be able to verify that messages are encrypted.

Means of verification

Conduct test *UC2.2_NF_11_TC1* as described in the Annex.

KPI's

- **KPI_UC2.2_1**

Description

Detect delays in delivery route and trigger a recalculation in less than 2 minutes.

Baseline

15 min

Target

2 min

Acceptance criteria

- Delays in delivery route are detected under 2 minutes and a recalculation is triggered.

Means of verification

Conduct the test *KPI_UC2.2_1_TC1* as described in the Annex.

- **KPI_UC2.2_2**

Description
Reduce average delivery route duration by 10%
Baseline
3h
Target
2h40min
Acceptance criteria
- Average delivery route duration is reduced.

Means of verification
A dataset of historical delivery routes together with their duration will be collected. The route identification for these routes will be executed and the resulting theoretical time compared.

- **KPI_UC2.2_3**

Description
Reduction of data transfer from the Edge to the Cloud. Less than 5% of inconsistent data traces are sent.
Baseline
n/a
Target
<5%
Due to the final proposed software architecture, this requirement is no longer relevant to the use case.

- **KPI_UC2.2_4**

Description
Increase the number of tasks executed in fog.
Baseline
None
Target
Vehicle routing should happen at fog.
Acceptance criteria
- Route optimization requests are executed at edge or at cloud depending on the overall performance of the application.

Means of verification
This KPI will be met as long as requirements F_10, F_11 and NF_10 are met.

UC 3 PRECISION AGRICULTURE

Introduction
Augmenta Edge devices improve agricultural practices by optimizing the application of chemicals on farms. Through advanced technology, Augmenta devices enable farmers to achieve higher crop yields while minimizing chemical usage. Augmenta's applications integrate state-of-the-art AI algorithms and intuitive visualization tools, empowering farmers to gain deeper insights into their field operations. By leveraging these solutions, farmers can efficiently acquire understanding of their chemical applications, track savings on chemical usage, and monitor their final yields.

Augmenta's Field Analyzers collect images from multispectrum cameras and other sensor data regarding to the plant health and environmental conditions. Upon completion of the re-collection of

the information, the Field Analyzer transmits the collected data to the computing platform and starts its processing. The processing of the collected data is divided in two stages:

i) A data preparation phase where data is transformed, prepared and some relevant metrics for the map generation phase are generated. The data preparation phase is a relatively simple serial process with low computing requirements that are mostly invariant of the size of the land plot being processed and the type of maps requested.

ii) A map generation phase that takes the outputs from the data preparation phase and generates images (maps), csv files with statistical analyses and shapefiles. The map generation phase is an intensive computation process (both in terms of CPU and RAM) that is highly affected by the size of the land plot being processed and the type of maps requested.

Once the processing of the data is completed, the resulting outputs are uploaded to remote storage systems for further analysis by the users through specialized applications.

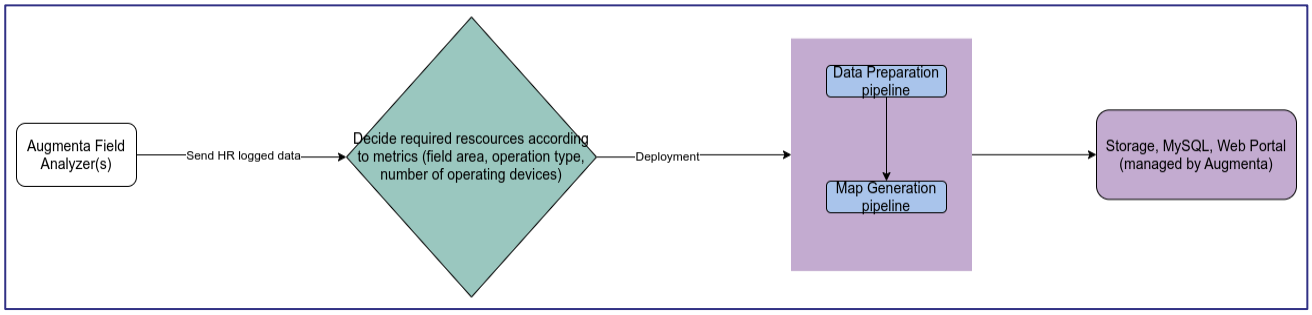


Figure 22. UC 3- Sequence of deployment processes within Augmenta use case.

Augmenta's aims to leverage the NebulOuS workflow execution mechanism to orchestrate the execution of the described processing pipeline and optimally managing the computation resources needed to execute parallel instances of such pipeline. The overarching objective of this orchestration is to optimize processing capabilities (minimize processing time) while concurrently reducing cloud expenses through the integration of Edge Computing.

Use case 3 architecture overview

The architecture of the Augmenta pilot application involves several components and resources distributed across Cloud, Service Provider Edge, and User Edge layers. The NebulOuS Control Plane orchestrates the deployment and management of these components. Below is a detailed description of each layer and the components involved.

Cloud Infrastructure (GCP)

1. **VM (Master)**:
   - **Optimizer Solver**: Manages and optimizes resource allocation and task scheduling.
   - **Event Processing Manager**: Handles events and notifications within the system.
   - **WireGuard Client**: Ensures secure VPN connections between different layers.
   - **k8s-gatekkeper**: Manages Kubernetes policies and ensures security compliance.
2. **VM (Worker)**:

- **Map Generation**: Performs the intensive computation required for generating maps from the prepared data.
- **Event Processing Manager**: Similar to the master VM, it handles events specific to the worker VM.
- **WireGuard Client**: Maintains secure connections for data transmission.
- **KubeVela:** allows for the deployment of Open Application Model-based service graphs

Service Provider Edge Infrastructure

1. **5G Node (Worker)**:
   - **Data Preparation**: Prepares data collected from the User Edge devices for further processing.
   - **Event Processing Agent**: Manages events and notifications specific to the Service Provider Edge layer.
   - **WireGuard Client**: Ensures secure connections within the Service Provider Edge layer.

User Edge Infrastructure

1. **Aug Field Analyzer (Worker)**:
   - **Data Preparation**: Collects and prepares data from multispectral cameras and sensors.
   - **Map generation**: Performs the intensive computation required for generating maps from the prepared data
   - **Event Processing Agent**: Manages and processes events at the User Edge level.
   - **WireGuard Client**: Maintains secure connections for data transmission to Service Provider Edge and Cloud layers.

Network Connectivity

- **VPN:** The architecture uses a secure Virtual Private Network (VPN) facilitated by WireGuard clients to ensure safe and encrypted data transmission between the Cloud (AWS) and Edge environments

Table 11. UC 3- Execution of the components of Augmenta application.

UC 3 (Augmenta)	Component	Execution (Cloud/Service Provider Edge/User Edge)	Cloud	Service Provider Edge	User Edge
#1	Data Preparation	Cloud/Service Provider Edge/ User Edge	GCP	5G Node	Augmenta's Field Analyzer
#2	Map generation	Cloud/User Edge	GCP		Augmenta's Field Analyzer

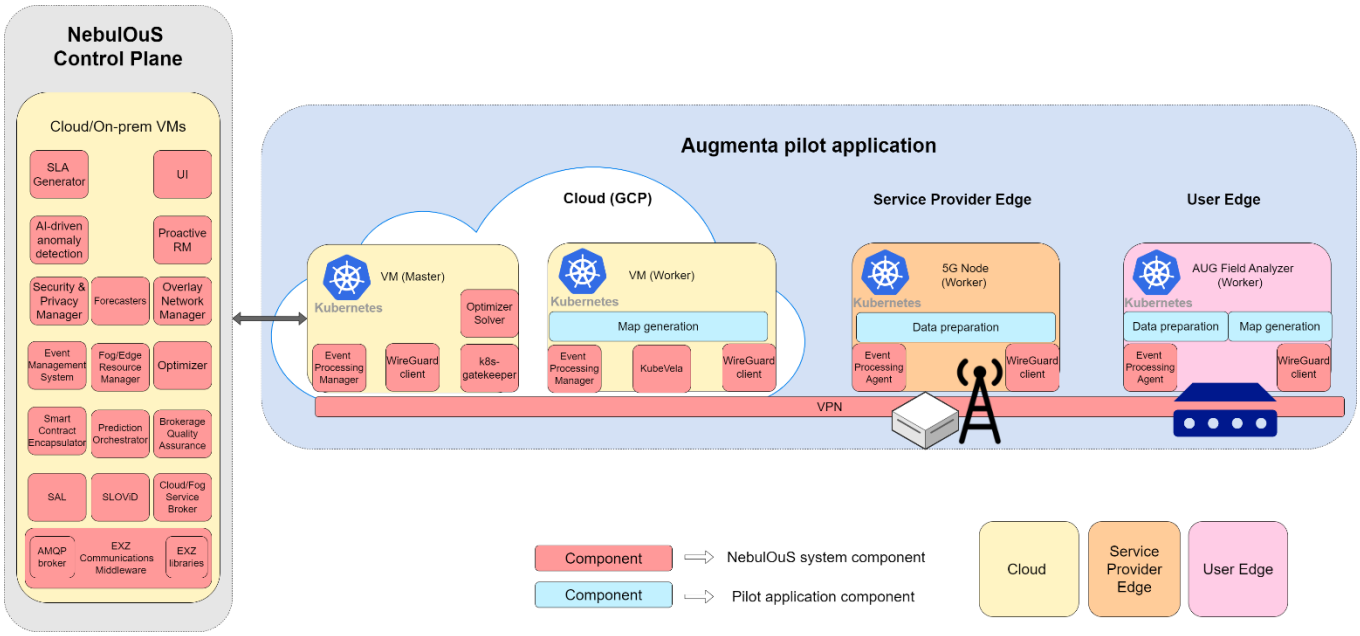


Figure 23. UC 3- Execution of the components of Augmenta application.

Functional requirements assessment

- **F_01**

Description

The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows. It ensures workloads are distributed on the appropriate computing continuum (just edge, just cloud, edge + cloud).

NebulOuS should orchestrate the described data processing workflow (data preparation, map generation), allocating the necessary resources for each step.

Acceptance criteria

- NebulOuS selects the appropriate computing node/s to execute the data processing workflow depending on the size of the field (ha).

Means of verification

Conduct the tests *UC3_F_01_TC1* and *UC3_F_01_TC2* as described in the Annex.

- **F_03**

Description

The system shall provide a graphic user interface for the administration of the deployed components

Acceptance criteria

- User-friendly Interface
- The user can adjust the optimization criteria and computational resources utilized by NebulOuS
- User can monitor the deployed components in GUI

Means of verification
Conduct the test *UC3_F_03_TC1* as described in the Annex.

- **F_04**

Description
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 proposed software architecture, this functionality is no longer required by the use case.

- **F_05**

Description
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.

The expected duration of each of the workflow phases (data preparation, map generation) can be estimated beforehand based on the size of the land plot and the amount of computation resources dedicated. However, the estimate is not 100% accurate and some variations can happen due to unexpected complexities on the processing of the data. This results in blocking of the computing nodes for longer periods. In such situations, NebulOuS should re-schedule any workflow steps that are waiting for the resources to be freed.

Acceptance criteria
- When there is a workflow execution that is taking longer than expected, NebulOuS should re-allocate any workflow executions that are waiting for the resources dedicated to the prolonged workflow execution.

Means of verification
Conduct the test *UC3_F_05_TC1* as described in the Annex.

- **F_12**

Description
Monitor and predict downtime of registered resources to maximize time/cost ratio. NebulOuS should be able to accommodate applications that prioritize resources with lower downtime, in comparison with the opposite.

Acceptance criteria
- NebulOuS prioritizes the use of cloud resources that are considered to be more reliable.

Means of verification
Conduct the test *UC3_F_12_TC1* as described in the Annex.

- **F_13**

Description
Handle paused or dropped tasks.

Acceptance criteria

- In the event of dropped or killed jobs for any reason, Nebulous is responsible for re-scheduling the jobs to other nodes.

Means of verification
Conduct the test *UC3_F_13_TC1* as described in the Annex.

Non-functional requirements assessment

- **NF_01**

Description
Fast decisions from NebulOuS, regarding the resources to allocate, and fast rapid execution of these decisions (fast communication, internal processing to take decisions etc.).

Acceptance criteria
- Once a new workflow execution request is submitted, NebulOuS decides where the computation will be allocated.

Means of verification
Conduct the test *UC3_F_05_TC1* as described in the Annex.

- **NF_02**

Description
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.

Acceptance criteria
- NebulOuS continues operation even when the bandwidth of the edge nodes and the cloud is limited and ignores poor connectivity for limited time.

Means of verification
Conduct the test *UC3_NF_02_TC1* as described in the Annex.

- **NF_03**

Description
The platform provides secure access to the resources (clouds, edge resources etc.) (i.e provides user Identification/Authentication, User/application manager roles etc.)

Acceptance criteria
- Only registered users can onboard new edge devices to be used as part of the application deployment.

Means of verification
Conduct the test *UC3_NF_03_TC1* as described in the Annex.

- **NF_10**

Description
The NebulOuS platform will allow effective scaling of the cloud resources (using edge/cloud computing).

The expected duration of each of the workflow phases (data preparation, map generation) can be estimated beforehand based on the size of the land plot and the amount of computation resources dedicated. NebulOuS should use this information to allocate necessary computation resources to cope with the workload at any given time.

Acceptance criteria

- When there is a spike in the number of workflows to be executed, NebulOuS should allocate more computing resources. When the number of workflows to be executed is reduced, NebulOuS should de-allocate unnecessary resources.

Means of verification
Conduct the test *UC3_NF_10_TC1* as described in the Annex.

- **NF_11**

Description
The NebulOuS platform ensures data privacy when transferring data from one node to another.

Acceptance criteria

- Information exchanged as part of the workflow execution is encrypted.

Means of verification
Conduct the test *UC3_NF_11_TC1* as described in the Annex.

KPI's

- **KPI_UC3_1**

Description
Processing time reduction
Baseline
15 %
Target
30%
Acceptance criteria

- The total processing time using NebulOuS orchestration mechanism is reduced.

Means of verification
A test dataset with a 50 workflow execution requests of different sizes will be created. A computing environment with a mix of edge nodes and a cloud provider with limitations on the available number of nodes. The dataset will be processed with the current orchestration mechanism and the NebulOuS orchestration mechanism. The total processing time will be computed and compared.

- **KPI_UC3_2**

Description
Cloud cost reduction
Baseline
5%
Target
10%
Acceptance criteria

- The cost of cloud allocated resources using the NebulOuS orchestration mechanism is reduced.

Means of verification
After the test procedure described for KPI_01, the cost of the cloud resources used with the current orchestration mechanism and the NebulOuS orchestration mechanism will be compared.

- **KPI_UC3_3**

Description
Increase capabilities of processing tasks
Baseline
5%
Target
10%
Acceptance criteria
- The total count of tasks completed in a specific amount of time using the NebulOuS orchestration mechanism is increased.

Means of verification
The test procedure described for KPI_01 will be executed again, limiting the amount of time available to run the test. The number of tasks completed with the current orchestration mechanism and the NebulOuS orchestration mechanism will be compared.

UC 4 CRISIS MANAGEMENT

Introduction
In an event of large-scale disasters like floods, earthquakes or wildfires, the @-Fire International Disaster Response Germany (FIRE) crisis response team needs to deploy, in the early stages of the incident, a software solution that facilitates the management of response crews, material and equipment from different stakeholders intervening in the situation. This software solution integrates information from on field sensors, information gathered through an APP, and information from Cloud (if available) to create a dynamic situational map that can then be used for coordination and by the response teams for direct assistance.
However, often, the established structures for communication in the area are disrupted, limiting, or prohibiting access to Cloud Computing. Moreover, computation resources are scarce as response crews have the possibility to deploy a very limited amount of hardware in the first phases of the disaster. On top of that, energy consumed by these devices is limited, as it often comes from generators running on fuel. This creates a scenario with un-reliable computation and communication resources with very tight constraints regarding computing power and energy consumption.

The goal of NebulOuS is to allow the creation of a flexible Edge Computing network composed of multiple computing units that are constrained in terms of computing power, communication,
and energy. Using this Edge NebulOuS shall orchestrate the execution of the different software components that compose the FIRE software solution, reacting to sudden loses of part of the edge computing network (due to hardware resources being undeployed, energy or communication outages, etc...). NebulOuS shall also allow offloading some computation requirements to the cloud when it is available. Finally, due to the sensitivity of data, security aspects and level of access in all the process are critical elements.

As described in Figure 24 the FIRE software solution is divided in three levels: i) field; ii) Edge and iii) Cloud.

The **field level** is mainly constituted of a network of sensors reporting relevant field data, e.g., water levels, structural changes, weather conditions, and Personal/equipment locations. They are connected via LORAWAN to a LORAWAN Server (The things stack) in the Edge level (#7).
The **Edge level** (physically located at FIRE base camp) is where the crisis management software solution components are deployed. This solution is composed of a use case persistence APP (#1). The map application (#2) and its accompanying database (#4) that offers a dynamic situational map augmented with all the data collected from the different sources. The MQTT broker (#3) that serves as a communication channel for IoT data (coming from field) with the Things Stack (IoT platform) and other components of the solution. The tile server (#5) responsible for providing tiles for the map visualization. The PostgreSQL DB (#6) that serves as database, The Redis DB (#8) that serves as Backup Storage for IoT data.
Finally, the Cloud level, refers to all the services available outside the disaster zone. It includes an ArcGIS web application that allows personnel in the headquarters consume the in-field data and OpenStreetMap as a Basemap provider of up-to-date maps to Mapserver in the field.

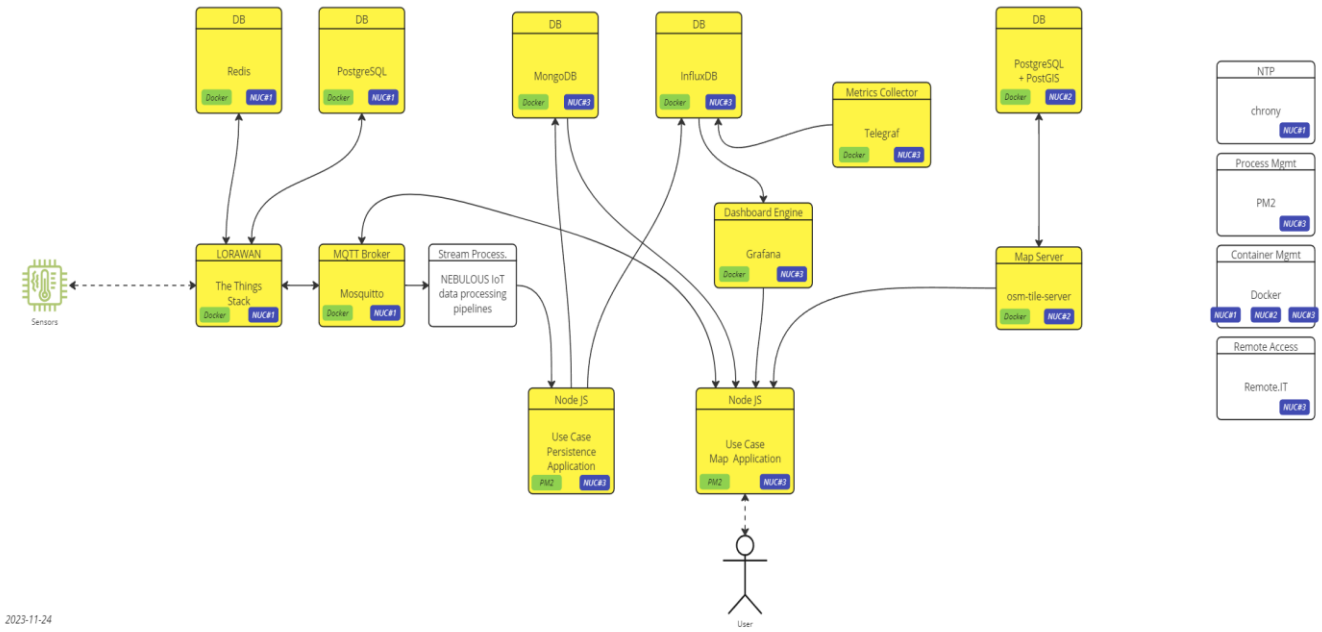


Figure 24. -UC4- Component Diagram

Use case 4 architecture overview

The FIRE architecture adapts to the availability of Cloud connectivity by leveraging Cloud resources when available for better performance and scalability. When Cloud connectivity is lost, it relies entirely on User Edge resources, maintaining critical functionalities through a robust and secure local setup. The use of components such as the Optimiser Solver, Event Processing Manager, and WireGuard ensures efficient and secure operations across both scenarios.

With Cloud Connectivity

Cloud Infrastructure (AWS/Azure)

1. **VM (Master)**:

- **Optimiser Solver**: Manages the optimization tasks within the system.
- **Event Processing Manager**: Handles the processing of events from various sources.
- **WireGuard Client**: Ensures secure communication through a VPN.
- **K8s Gatekeeper**: Provides security and policy enforcement in the Kubernetes cluster.

2. **VM (Worker)**:

- **Persistence Webservice**: Manages the storage and retrieval of data.
- **MQTT Broker**: Facilitates communication between IoT devices.
- **Event Processing Manager**: Similar to the master, processes events.
- **WireGuard Client**: Ensures secure communication through a VPN.
- **KubeVela:** allows for the deployment of Open Application Model-based service graphs

User Edge Infrastructure

1. **Intel NUC (Worker) (x3)**:

- **Map Webservice**: Similar to the Edge server, provides map-related services.
- **Postgres DB**: Database for local data storage.
- **Event Processing Agent**: Processes events locally.
- **WireGuard Client**: Ensures secure communication through a VPN.

Without Cloud Connectivity

In scenarios where cloud connectivity is not available, the architecture relies entirely on User Edge resources, and the NebulOuS control plane is hosted on the User Edge server.

User Edge Infrastructure

1. **Intel NUC (Master):**
   - **Optimiser Solver**: Manages the optimization tasks within the system.
   - **K8s Gatekeeper**: Provides security and policy enforcement in the Kubernetes cluster
   - **WireGuard Client**: Ensures secure communication through a VPN.
   - **Event Processing Manager**: Handles the processing of events.
2. **Intel NUC (Worker) (x1)**:

- **Persistence Webservice**: Manages the storage and retrieval of data.
- **MQTT Broker**: Facilitates communication between IoT devices.
- **WireGuard Client**: Ensures secure communication through a VPN.
- **Event Processing Agent**: Processes events.
- **KubeVela:** allows for the deployment of Open Application Model-based service graphs

3. **Intel NUC (Worker) (x2)**:
    - **Map Webservice**: Provides map-related services and data.
    - **Postgres DB**: Database for local data storage.
    - **Event Processing Agent**: Processes events.
    - **WireGuard Client**: Ensures secure communication through a VPN.

Network Connectivity

- **VPN:** The architecture uses a secure Virtual Private Network (VPN) facilitated by WireGuard clients to ensure safe and encrypted data transmission between the cloud (AWS) and edge environments.

Table 12. UC 4- Execution of the components of FIRE application.

UC 4 (FIRE)	Component	Execution (Cloud/Service Provider Edge/User Edge)	Cloud	User Edge
#1	Edge Postgres Database	User Edge	-	3 Intel NUC
#2	Edge Map Webservice	User Edge	-	3 Intel NUC
#3	Mqtt Broker	Cloud/ User Edge	AWS (in future: AZURE)	3 Intel NUC
#4	Persistence Webservice	Cloud/ User Edge	AWS (in future: AZURE)	3 Intel NUC
#5	Tileserver IMPORT	Cloud/ User Edge	AWS (in future: AZURE)	3 Intel NUC
#6	Tileserver SERVE	User Edge	-	3 Intel NUC
#7	Cloud Postgres Database	Cloud	AWS (in future: AZURE)	
#8	Cloud Map Webservice	Cloud	AWS (in future: AZURE)	

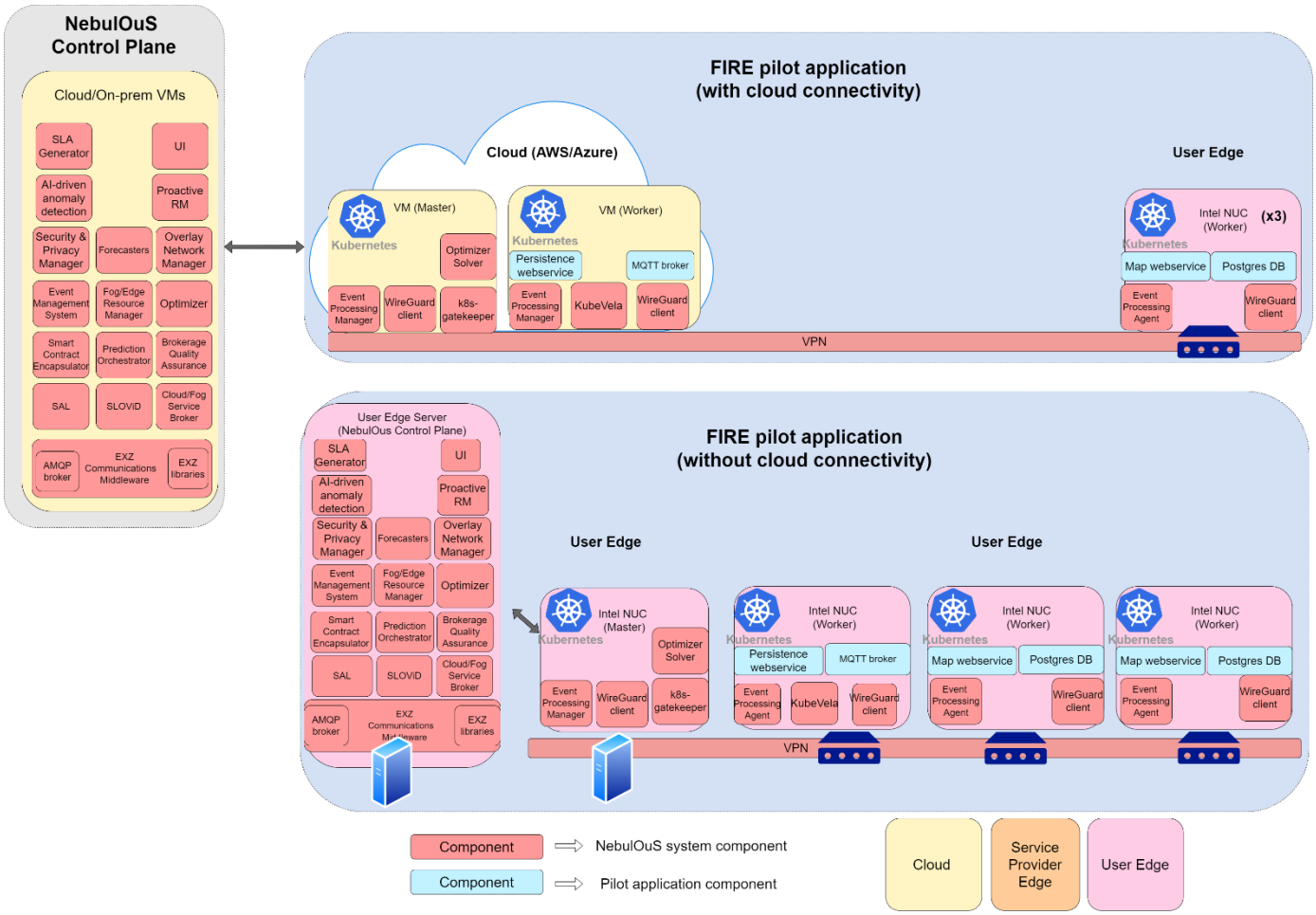


Figure 25. UC 4- Execution of the components of FIRE application.

Functional requirements assessment

- **F_01**

Description

The NebulOuS platform provides orchestration mechanisms and data streaming capabilities that allow implementing flexible data processing workflows. It ensures workloads are distributed on the appropriate computing continuum (just edge, just cloud, edge + cloud).
The Workload can be distributed to available edge capacities in the case that there is no cloud capacity available. The NebulOuS shall be able to orchestrate the distribution.

Acceptance criteria

- NebulOuS reacts to a failure of a software component (e.g. if it suddenly disappears) and reorganizes the deployment on the available resources brokered by NebulOuS at the moment.

Means of verification

Conduct the test *UC4_F_01_TC1* as described in the Annex.

- **F_02**

Description

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. The system shall provide an event-based publish/subscribe communication "bus" for the interaction of components.
This application uses the MQTT based pub/sub for communication between several application components.

Acceptance criteria

- NebulOuS can orchestrate communication between multiple application components.

Means of verification
Conduct test *UC4_F_02_TC1* as described in the Annex.

- **F_03**

Description
The system shall provide a graphic user interface for the administration of the deployed components. The users of this application shall be able to configure the optimization criteria and computational resources to be used by NebulOuS via a GUI.

Acceptance criteria

- The user is able to adapt the optimization criteria and computational resources to be used by NebulOuS via a GUI

Means of verification
Conduct the test *UC4_F_03_TC1* as described in the Annex.

- **F_04**

Description
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.

The edge computing network created in the first stages of the emergency response is based on small computing units (NUCs) powered by unreliable and limited energy sources (e.g. generators running on fuel). NebulOuS shall permit to define deployment policies that take into consideration power availability.
Furthermore, NebulOuS shall allow to define specific deployment locations for individual software components.

Acceptance criteria

- When creating the application service graph, we can define a policy that states that a certain component can only run on certain devices.

Means of verification
Conduct the test *UC4_F_04_TC1* as described in the Annex.

- **F_14**

Description
The system shall be able to be started and operated during run-time with minimal prior training.

Acceptance criteria

- A non-IT expert should that undertakes a crash course of 4 hours should be available to deploy the described software solution in a private cloud composed of 3 computation nodes in less than 1 hour using a node with an existing installation of NebulOuS core and the application already registered.

Means of verification
Conduct the test *UC4_F_14_TC1* as described in the Annex.

Non-functional requirements assessment

- **NF_01**

Description
Fast decisions from NebulOuS, regarding the resources to allocate, and fast rapid execution of these decisions (fast communication, internal processing to take decisions etc.).

Acceptance criteria

- The deployment of the application is handled in a timely manner, so that operations of the deployed application are not interrupted.

Means of verification
Conduct the test *UC4_NF_01_TC1* as described in the Annex.

- **NF_02**

Description
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.
FIRE software solution must run regardless of the availability of connectivity with the remote cloud.

Acceptance criteria

- NebulOuS is able to deploy the application components on mixed edge-cloud environments. If cloud connection is lost the application must be redeployed with all components on the edge.

Means of verification
Conduct the test *UC4_NF_02_TC1* as described in the Annex.

- **NF_03**

Description
The platform provides secure access to the resources (clouds, edge resources etc.) (i.e. provides user Identification/Authentication, User/application manager roles etc.)
Only admin can login to define computation resources managed by NebulOuS.

Acceptance criteria

- For this case study, only an admin user can configure and decide which working nodes can be used to deploy the application.

Means of verification
Conduct the test *UC4_NF_03_TC1* as described in the Annex.

- **NF_07**

Description
The system shall be able to cope with a sudden loss of connection to individual computational units. The edge computing network created in the first stages of the emergency response is based on diverse computing units (PCs) powered by unreliable and limited energy sources (e.g. generators running on fuel). NebulOuS shall use these computing units to deploy FIRE software solution components, and, in case of a sudden loss of any of the nodes, re-deploy the missing components to other nodes.

Acceptance criteria
- If the connection to a worker node is lost NebulOuS must redeploy the components which were deployed on this node, to the remaining available nodes.

Means of verification
Conduct the test *UC4_NF_07_TC1* as described in the Annex.

- **NF_08**

Description
The system shall be able to adapt the maximum load of the edge devices to the available power supply.

Acceptance criteria
- The NebulOuS optimization criteria can be informed with external sensors (e.g. power availability) and the optimization goal can be set to optimize by these criteria.

Means of verification
Conduct the test *UC4_F_04_TC2* as described in the Annex.

- **NF_10**

Description
The NebulOuS platform will allow effective scaling of the cloud resources (in the edge/cloud computing).
If new hardware resources are deployed to the edge and registered in the Cloud Fog Service Broker, NebulOuS should be able to make use of them.
If remote cloud connectivity is available, NebulOuS should offload some computation there.

Acceptance criteria
- NebulOuS will reevaluate the deployment according to available computational resources and redeploy the application if a better deployment is possible.

Means of verification
Conduct the test *UC4_NF_10_TC1* as described in the Annex.

KPI's

- **KPI_UC4_1**

Description
Increase of total information processed digitally in disaster management.
Baseline
2%
The collection of data from different resources commonly happens in an analog fashion.

For a usual crisis response scenario, it can be assumed, that 20 personnel are deployed for at least 5 days (120 hours) excluding travel time.
Personnel are paired and communication is done per pair. This means, that 10 pairs are tracked / observed by the coordination cell. On average personnel is working 20 hours per day, resulting in 100 active hours per pair and status / location updates are collected twice per hour per team. This results in a total of approx. 2000 communication events per deployment.
Since communication infrastructure is usually limited, only a small number of these events are process automatically, through systems like WhatsApp location sharing, UN ASIGN or Garmin Explore / inReach.

Additionally, environmental data (e.g. seismic activity, water levels, temperature and humidity readings) are not processed digitally.

Information Category	**Average volume (per deployment)**	**Digitally processed (per deployment)**
Status / Location Updates	2000	20
Point of Interest (e.g. Infrastructure) Update	120	3
Environmental / Ambient Information	60	20
Total	**2180**	**43 (~2%)**

Target

10%

Through the application of the solution, it is expected, that the amount of data regarding points of interest and environmental information can be increased significantly.
Likewise, a significant number of location updates are expected to be automated. This number can likely even be increased in a full adoption of the system.

Information Category	**Average volume (per deployment)**	**Digitally processed (per deployment)**
Status / Location Updates	2000	180
Point of Interest (e.g. Infrastructure) Update	120	10
Environmental / Ambient Information	60	30
Total	**2180**	**220 (~10%)**

Acceptance criteria

- The KPI will be monitored through evaluation in field exercises throughout the project and documented. The expected increase in digitally processed data is achieved during field exercises.

Means of verification

The KPI will be monitored through evaluation in field exercises throughout the project and documented.

- **KPI_UC4_2**

Description

Increase Local data communication coverage during the early stages of disaster situations.

Baseline

5%

Target

60%

Acceptance criteria

- In current deployments a digital communication infrastructure is usually only available in the Base of Operations (through WiFi) while there is limited mobile network coverage. Thanks to the flexible edge low power computing network enabled by NebulOuS and the deployment of FIRE software solution on that network, it is estimated that the digital data transfers will be available in a much larger area. While this heavily depends on the topology of the deployment area, it is estimated that about 60% of a typical deployment area can be covered.

Means of verification

The KPI will be monitored through evaluation in field exercises throughout the project and documented.

- **KPI_UC4_3**

Description

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.

Baseline

15-60 mins

Target

1 min

Acceptance criteria

- The automatization of collecting, storing, and visualizing the data from the field through the different types of sensors or external resources.

Means of verification

During field exercises, the user can visualize the collected information from field and other information sources (e.g. from field directly, verbal Information), using the provided application GUI.

- **KPI_UC4_4**

Description

Increase the capacity of Average number of individual situational information points available (e.g., water level)

Baseline

30

Target

1000

Acceptance criteria

- The number of sensors (without considering the type) provided to the application should reach the target value, and the information they provide shall be appropriately processed.

Means of verification

A performance test of the FIRE software solution will be conducted. For this, a deployment of the solution using NebulOuS will be done utilizing resources that are comparable to the computation resources available on the first stage of the crisis response. With this deployment in place, 1000 readings from water level sensors will be simulated in a period of 1 day.

- **KPI_UC4_5**

Description

Complementary information automatically derived in disaster management.

Baseline

0

Target

100

Acceptance criteria

- The crowd/individual analysis from existing collected information or external information resources has to be considered as an extra layer of situational awareness (complementary information).
- This includes information, which is currently not observed, as the effort is too high (e.g. detailed environmental data, detailed PoI data) but would positively impact the coordination of the deployment and the decision making process.

Means of verification

This KPI will compare the availability of multiple information sources and will be met if during the participation of the use-case in NebulOuS at least one more source of information is gathered.

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

All selected use cases demonstrated ways to leverage the potential of NebulOuS and outlined plans for verifying the requirements of the platform. They will enable us to verify the NebulOuS approach and implementation.

The Table below presents how the provided in D 2.1 functional and non-functional requirements will be tested and validated by the pilots. In two cases, a requirement was considered no longer relevant due to the final proposed software architecture. Detailed descriptions of the test cases are added in the Annex.

Table 13. 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	UC1.1_F_01_TC1 UC1.1_F_01_TC2 UC1.1_F_01_TC3	UC1.2_F_01_TC1	UC2.1_F_01_TC1	UC2.1_F_01_TC1	UC3_F_01_TC1 UC3_F_01_TC2	UC4_F_01_TC1
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	UC2.2_F_01_TC1		UC4_F_02_TC1
F_03	The system shall provide a graphic user interface for the administration of the deployed components	UC1.1_F_03_TC1				UC3_F_03_TC1	UC4_F_03_TC1
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		UC2.1_F_04_TC1	No longer relevant to the UC	No longer relevant to the UC	UC4_F_04_TC1
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	UC2.2_F_01_TC1	UC3_F_05_TC1	

F_06	The NebulOuS platform provides infrastructure monitoring capabilities	UC1.1_F_06_TC1 UC1.1_F_01_TC1 UC1.1_F_01_TC2					
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)	UC1.1_F_07_TC1 UC1.1_F_07_TC2					
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					
F_09	The NebulOuS platform provides near-real-time communication between application components.			UC2.1_F_02_TC1	UC2.2_F_09_TC1		
F_10	NebulOuS offers a mechanism to partition streaming analysis jobs between a variable number of workers (similar to Spark partitioning)				UC2.2_F_10_TC1		
F_11	NebulOuS should offer a mechanism to deploy and invoke serverless computation functions.				UC2.2_F_11_TC1		
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					UC3_F_12_TC1	
F_13	Handle paused or dropped tasks					UC3_F_13_TC	

F_14	The system shall be able to be started and operated during run-time with minimal prior training						UC4_F_14_TC1
NF_01	Fast decisions from NebulOuS, regarding the resources to allocate and fast execution of these decisions	UC1.1_F_04_TC1	UC1.2_F_01_TC1 UC1.2_NF_10_TC1	UC2.1_F_01_TC1 UC2.1_F_04_TC1 UC2.1_F_05_TC1	UC2.2_F_05_TC1	UC3_F_05_TC1	UC4_NF_01_TC1
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	UC1.1_F_01_TC1				UC3_NF_02_TC1	UC4_NF_02_TC1
NF_03	The platform provides secure access to the resources			UC2.1_NF_03_TC1	UC2.2_NF_03_TC1	UC3_NF_03_TC1	UC4_NF_03_TC1
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			
NF_05	Data transferred between edge nodes and between edge nodes and clouds should be secured			UC2.1_NF_05_TC1	UC2.2_NF_05_TC1		
NF_06	The system provides elasticity capabilities along with the cloud continuum, so HW resources are provisioned or freed depending on the workload				UC2.1_F_05_TC1		
NF_07	The system shall be able to cope with a sudden loss of connection to individual computational units						UC4_NF_07_TC1
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
NF_09	The NebulOuS platform provides near-real-time processing of big data			UC2.1_NF_09_TC1	UC2.2_NF_09_TC1		
NF_10	The NebulOuS platform will allow effective scaling of the cloud resources (in the edge/cloud computing)	UC1.1_NF_10_TC1	UC1.2_NF_10_TC1	UC2.1_NF_10_TC1	UC2.2_F_01_TC1	UC3_NF_10_TC1	UC4_NF_10_TC1

NF_11	The NebulOuS platform ensures data privacy when transferring data from one node to another	UC1.1_NF_11_TC1 UC1.1_NF_11_TC2 UC1.1_NF_11_TC3	UC1.2_NF_11_TC1	UC2.1_NF_11_TC1	UC2.2_NF_11_TC1	UC3_NF_11_TC1	

In Table 14 we demonstrate how the pilots evaluate and use the capabilities provided by components of NebulOuS platform.

Table 14. Usability of given components.

	UC1.1 Windmill Maintenance	UC1.2 Computer Vision	UC2.1 Mercabarna Intralogistics	UC2.2 Mercabarna Last Mile	UC3 Augmenta	UC4 FIRE
GUI	✓	✓	✓	✓	✓	✓
Optimizer	Utility function that will describe image queue length in comparison to image processing time. The goal is to reach processing time limit with minimal resource costs.	Utility function that will decide when to migrate from full cloud, full edge or edge + cloud scenarios.	Keep the license plate reading job execution time low while limiting necessary computing resources. Maintain 10fps for the vehicle detection and cropping module while limiting necessary computing resources.	Keep the time taken to process a new vehicle even low and the time taken to process a routing request low while limiting necessary computing resources.	Utility function that will describe the state of cpu and ram using metrics according to field area	1. find the most reliable (i.e. highest availability) deployment option 2. reduce the energy usage of edge resources
EMS/SLOV	*System metrics*: (cpu, gpu) - Image queue depth - Average processing time of every module	*System metrics:* (cpu, gpu, ram) *Custom metrics:* -Processing time -Detection time	*Custom metrics*: -Time for taken to process a license plate reading job. -Frames per second consumed by the vehicle detection and cropping module.	*Custom metrics*: -Time taken to process a new vehicle event. -Time taken to process a routing request.	*Custom metrics*: - field area - processing time minus field area - processing time minus field volume - no of parallel users/tasks	*System metrics*: - CPU utilisation - RAM utilisation *Custom metrics sent to EMS*: - age of information - app ui response time
CFSB	- cost reduction - processing time	-processing time	- proximity - cost reduction	- cost reduction	-	- reliability / availability - cost reduction

Security and Privacy Manager	None	None	None	None	None	None
Deployment Manager	✓	✓	✓	✓	✓	✓
Execution Adapter	Cloud /User Edge	Cloud/Service Provider Edge	Cloud/User Edge	Cloud/User Edge	Cloud/Service Provider Edge/User Edge	Cloud/User Edge
Overlay Network Menager	✓	✓	✓	✓	✓	✓
Data Collection and Management System	IoT device (UAV) sends image data to be process by object detection component (primarily User Edge but can be offloaded to cloud). Output of object detections is processed in the Cloud.	CCTV collects data and sends it to be processed either by the Service Provider Edge device (nvidea Jetson Xavier) or to the Cloud. The processing detects damage, incidents and animals on the street.	User Edge device process the camera stream and upload the data to Cloud/User Edge where the related metadata is generated and stored in a database	IoTpub-sub mechanism used to communicate between components of the application and collect relevant metrics.	Service Provider Edge/User Edge device after the end of the operation upload the data to Cloud that triggers a scheduler which initiates the processing workflow	Sensor data is received and validated against geofences and thresholds
Smart Contract Encapsulator	Components will be developed in the 2nd release and their evaluation from the UC is yet to be defined.					
Brokerage QA						
SLA Generator						
AI-Driven Anomaly Detection						

The Pilots and the applicants from the open calls will test and validate the NebulOuS platform according to their technological vision. The chosen projects will test the features of developed components and evaluate the innovation of the NebulOuS. To benefit from NebulOuS directly, the applications should be Cloud native encapsulating its component as containers that can be deployed under Kubernetes. In the Table below we present the list of features that NebulOuS platform providing for the users.

Table 15. Features of NebulOuS platform.

Features in components	Description
Cloud resource managing	A user interface that allows adding/removing public and private cloud providers to be brokered by NebulOuS.
Edge resource managing	A user interface that allows adding/removing edge nodes to be brokered by NebulOuS.
Application registration	A user interface that allows the definition of applications by specifying: the components that constitute the application, resource restrictions (such as the amount of CPU, RAM, and number of instances) for these components, resource brokerage preferences, service level objectives, and optimization goals.
Application deployment	Mechanisms for finding optimal deployment strategies for the user defined applications using the resources brokered by the system, enacting the deployment, monitoring the runtime of the application and the fulfilment of the SLO.
Secure communication	Encrypted communications between components of the application clusters.
IoT data processing	Tools for facilitating the definition of IoT data processing-oriented applications in NebulOuS and leverage the meta-OS capacity of adapting the application deployment topology based on the varying workload.
Serverless functions	Tools for using serverless functions as part of applications in NebulOuS and leverage the meta-OS capacity of adapting the application deployment topology based on the varying workload.
Computing resources brokerage	A multi-criteria decision mechanism that selects the most appropriate computing resources based on factors such as computing capacity, network bandwidth, storage capacity, power availability, and physical location of edge devices.
Organisation wide resources preferences.	Mechanisms to ensure abidance of application provisioning requirements and preferences at organisation level.
Workflows	Tools for facilitating the definition of workflow-oriented applications in NebulOuS and leverage the meta-OS capacity of adapting the application deployment topology based on the varying workload.
SLA definition and monitoring	Capability to define SLAs with computing resource providers, monitor their fulfilment, and identify any violations. Utilize blockchain technology to record the data securely.
Application adaptation	Continuously monitor application metrics to both reactively and proactively detect SLO violations, and make necessary adjustments to the application deployment topology to mitigate such violations.
Anomaly detection	AI-driven techniques for detecting compromised computing nodes part of the application cluster and automatically removing them.

8. CONCLUSIONS

In this document, we have outlined all of the components presented in the architecture scope of the platform, developed for the initial release, as well as those slated for further development in subsequent iterations. We have presented the interactions between components and the connections are depicted in the diagrams. Dependencies between components evolved dynamically, so the final sequences for all created components will be presented in the next version of the document.

Software testing as an integral part of the software development process is playing pivotal role in delivering high-quality, reliable, secure software products to end-users. Through collaboration with our use case partners, we endeavoured to tailor the platform to meet the users' needs through numerous meetings and sessions. Looking ahead, our future plans include optimizing the platform extensively for usability and aligning with the requirements of businesses.

Deliverable 6.2 *Final Release of the NebulOuS Integrated Platform & Pilot Demonstrators Evaluation Report* will be the report on the final integrated platform to be officially released. It will consider the pilots demonstrators key findings that will be available on M34. The results in terms of technical merits and functional performance and lessons learnt will also be reported, while the end-user's satisfaction will be discussed for further testing adjustments.

9. ANNEX

NEBULOUS CORE MANUAL INSTALLATION INSTRUCTION

A. Prerequisites

- *Operating system*

Ubuntu 22.04

- *Firewall*

Allow incoming TCP traffic to the following ports:

Port	Component	Comments
30111	EMS	EMS web GUI (protocol HTTPS)
31617	EMS	EMS ActiveMQ broker (protocol openwire over TLS)
32222	EMS	EMS baguette server (protocol SSH)

Note: *EMS Helm chart will deploy* `nebulous-monitoring-public` *service with fixed port bindings:* `8111:30111/TCP,61617:31617/TCP,2222:32222/TCP`

B. Installation steps

1. *Update System and Install Dependencies*

```
sudo apt update && sudo apt upgrade -y
sudo apt install curl apt-transport-https -y
```

2. *Install Kubernetes Command Line Tool (kubectl)*

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
```

3. *Install Docker*

```
sudo apt install docker.io -y
sudo systemctl start docker
sudo systemctl enable docker
```

4. *Install local cluster (example Minikube)*

```
curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-
amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
minikube start --driver=docker
```

5. *Download Nebulous manifests*

https://drive.google.com/drive/folders/1Uo0HlO-FLO6kY99fGLYb1vrgdZOs6DZt

6. *Install Helm*

```
curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
chmod 700 get_helm.sh
./get_helm.sh
```

7. *Install and Configure Flux*

```
curl -s https://fluxcd.io/install.sh | sudo bash
. <(flux completion bash)
flux install
flux --version
```

8. *Apply Kubernetes Configurations*

for example:
`kubectl apply -f nebulous/kustomization.yaml`

TOPICS IN COMMUNICATION BETWEEN THE COMPONENTS

Name of the topic	Description and Usage context
eu.nebulouscloud.monitoring.metric_list	EMS server deployed at application clusters, publish to this topic in order to let other components know which metrics are collected in the context of an application deployment. Prediction Orchestrator and Forecasters may subscribe to this topic in order to receive the list of metrics provided by EMS.
eu.nebulouscloud.monitoring.slo.new	EMS servers deployed at application clusters, publish to this topic in order to let other components (esp. SLOViD) know how SLO constraints are calculated in the context of an application deployment. SLO Violation Detector subscribes to this topic in order to receive the list of SLO constraints and the calculations needed for evaluating them, in order to properly initialize. Evaluating if SLO constraints are met.
eu.nebulouscloud.monitoring.slo.severity_value	SLO Violation Detector and EMS server deployed at application clusters, publish to this topic in order to let other components (esp. Optimiser) know that an SLO violation is about to happen or has already occurred. The distinction between predicted SLO violation and SLO violation that have already occurred is based on the value of 'probability'; for SLO violations that have already occurred it is always 1.0, while for predicted SLO violations it is <= 1.0. Optimiser subscribes to this topic in order to receive the SLO violation announcements Announcing predicted or occurred (realtime) SLO violations they detect.
eu.nebulouscloud.monitoring.realtime.{METRIC_NAME}	This topic provides information on the values of realtime metrics, used by various components

eu.nebulouscloud.monitoring.predicted.{METRIC_NAME}	This topic provides the values of predicted metrics, used to deliver appropriate reconfiguration actions
eu.nebulouscloud.monitoring.device_lost	Events sent to this topic include the id of the edge device which has been non-responsive or lost, in order to trigger a reconfiguration (in case it hosted a component). This topic informs SLOViD.
sloviolationdetector.debug	This is a private topic for debugging, with a format that should be free to change. For now, it is an empty json. This topic informs SLOViD to publish information on its internal state to sloviolationdetector.debug_output
sloviolationdetector.debug_output	This is a private topic for debugging, the messages sent to it following a format that should be free to change. This topic is used by SLOViD to publish information on its internal state.
eu.nebulouscloud.exn.sal.cloud.get	Transform into SAL endpoint Get All Clouds
eu.nebulouscloud.exn.sal.cloud.create	Transform into SAL endpoint Create new cloud
eu.nebulouscloud.exn.sal.cloud.delete	Transform into SAL endpoint Delete list of clouds provided in body
eu.nebulouscloud.exn.sal.node.get	Transform into SAL endpoint Get node definition
eu.nebulouscloud.exn.sal.node.create	Transform into SAL endpoint Create new node definition
eu.nebulouscloud.exn.sal.node.delete	Transform into SAL endpoint Delete existing node
eu.nebulouscloud.exn.sal.node.assign	Transform into SAL endpoint Assign tasks to nodes
eu.nebulouscloud.exn.sal.job.get	Transform into SAL endpoint Get existing deployment jobs
eu.nebulouscloud.exn.sal.job.create	Transform into SAL endpoint Create a new deployment job
eu.nebulouscloud.exn.sal.job.delete	Transform into SAL endpoint Delete existing deployment jobs
eu.nebulouscloud.exn.sal.job.kill	Transform into SAL endpoint Kill existing deployment job
eu.nebulouscloud.exn.sal.job.stop	Transform into SAL endpoint Stop existing deployment job
eu.nebulouscloud.exn.sal.job.submit	Transform into SAL endpoint Submit an existing deployment job
eu.nebulouscloud.exn.sal.nodecandidate.get	Transform into SAL endpoint Get node candidates based on search conditions. CFSB sends messages to SAL using this topic
eu.nebulouscloud.exn.sal.cluster.get	Transform into SAL endpoint Get info of existing cluster
eu.nebulouscloud.exn.sal.cluster.delete	Transform into SAL endpoint Delete existing cluster
eu.nebulouscloud.exn.sal.cluster.define	Transform into SAL endpoint Define a new cluster
eu.nebulouscloud.exn.sal.cluster.deploy	Transform into SAL endpoint Deploy defined cluster

eu.nebulouscloud.exn.sal.cluster.label	Transform into SAL endpoint Label existing cluster
eu.nebulouscloud.exn.sal.cluster.deployapplication	Transform into SAL endpoint Deploy application in existing cluster
eu.nebulouscloud.exn.sal.cluster.scaleout	Transform into SAL endpoint Scale out existing cluster
eu.nebulouscloud.exn.sal.cluster.scalein	Transform into SAL endpoint Scale in existing cluster
eu.nebulouscloud.optimiser.controller.app_state	The Optimiser Controller component is sending the application status message to indicate the status of the managed application. The Metric Updater will ignore SLO violation messages unless the application status indicate that the application is running
eu.nebulouscloud.optimiser.controller.metric_list	This message is sent from the Optimizer Controller component when a new application is created. It contains the names of the metric that jointly constitutes the application execution context. The Metric Updater actor will subscribe to the predicted versions of all of these metrics and cache the last value received for each of these metrics. Define the name of the metrics to be used in the *application execution context* and the naming conventions in both the metric model and the optimization problem AMPL model
eu.nebulouscloud.optimiser.controller.model	The definition of the constraint optimisation problem is sent as an AMPL file from the Optimizer Controller component to all the running AMPL Solver actors
eu.nebulouscloud.optimiser.controller.app_delete	The controller listens on this channel for a message containing an app ID, and will delete an application with that ID if it has been started before. Deleting an application entails sending a deleteCluster message to SAL (via the exn middleware) that will terminate the running app, and then unregistering the application object from the optimiser-controller itself. After this call, it is possible to deploy an application with the same ID.
eu.nebulouscloud.solver.state	The Solver Component will post the state as "starting" when the code starts running, and "ready" when it is ready to receive the messages from external components. When the solver is shut down it will send the "stopped" message.
eu.nebulouscloud.optimiser.solver.context	This message conveys the cached metric values by the Metric Updater to the Solver Manager when an SLO Violation message is received from the SLO Violation Detector. It contains the time stamp of the latest metric recorded and the map of all metric names and their values
eu.nebulouscloud.optimiser.solver.solution	The solution message is sent by the Execution Manger when the solver working on a particular application execution context returns the new configuration with values assigned to all variables of the problem
eu.nebulouscloud.cfsb.get_node_candidates	CFSB listens to this topic to receive a message from Optimizer
eu.nebulouscloud.exn.sal.nodecandidate.get.reply	Sent from SAL to the Cloud & Fog Service Broker (CFSB). Such a message contains a list of available resources filtered by the requirements set by the by Optimizer
eu.nebulouscloud.cfsb.get_node_candidates.reply	Sent from the Cloud & Fog Service Broker (CFSB) to the Optimizer to provide a list with characteristics of the resources as well as a score and ranking for each of them
eu.nebulouscloud.ui.application.new	This message is sent when the application is created. It contains the Applications' UUID which will be used as a reference for the entire lifecycle of the application

eu.nebulouscloud.ui.application.updated	This message is sent when the application is created. It contains the Applications' UUID which will be used as a reference for the entire lifecycle of the application
eu.nebulouscloud.ui.application.deploy	This message is sent when the user determines that the application should be deployed or re-deployed. It contains the Applications' UUID which will be used as a reference for the entire lifecycle of the application
eu.nebulouscloud.ui.dsl.generic	This message is sent in sequence with the *.new and the *.updated application, in order to provide the application description in JSON format.
eu.nebulouscloud.ui.dsl.metric_model	This message is sent in sequence with the *.new and the *.updated application. This communicated the metric model specified by the user in the agreed upon YAML format
eu.nebulouscloud.ui.policies.rule.upsert	This message is sent each time the policy rules have been updated in the UI.
eu.nebulouscloud.ui.policies.model.upsert	This message is sent each time the policy rules have been inserted in the UI
eu.nebulouscloud.infrastructure.edge_node_capacity.{DEVICE_ID}.{METRIC_NAME}	Inform the Brokerage Quality Assurance with data on the edge devices current context which will be used for instantiating the relevant ontology

DETAILS OF PREPARED TEST CASES

Some tests require the following APPs:

- **Dummy APP**: A simple application composed of a single component that each second prints the System time to console. No SLO is configured. Exactly one instance of the service can run at any given time.
- **Serverless dummy APP**: A simple application composed of two elements:
  - Factorial: A function as a service component that accepts a request with the parameter "n" of type numeric and returns the factorial of the input parameter. The computation is done using naïve number multiplication. Each instance of the function requires 1CPU and a variable amount of RAM depending on the input value.
  - Controller: A component that offers a REST API with one endpoint "GET /compute" with two URL parameters "I" and "K" being positive integer values. When the endpoint is called, the controller requests the execution of "I" instances of the "factorial" function to be executed, each instance "i" is given as an input "i+K". (First instance is given 1+K, second 2+K, third 3+K, etc…). The return of the request is a text with a line for each factorial execution, with the "i+K" value of that execution and the time it took to complete (in seconds).
- **Scaling APP**: An application composed of two elements:
  - Controller: A component that offers a REST API with one endpoint "POST /" with one URL parameters "t" being a positive integer value. When the endpoint is called, the controller adds a new request to the queue of pending requests. The component periodically publishes to the EMS the age in seconds of the oldest request waiting in the queue (metric "max_age"). The component offers an internal REST API to be consumed by the "worker" component. This API offers a "POST /accept" endpoint that

removes the first element from the list of pending requests (the oldest request) and returns it as an answer. If the list is empty returns null. The component offers a "GET /" that returns the list of pending requests and their age in seconds.
    - Worker: A component that, on start-up, queries the "POST /accept" endpoint of the "controller" component and sleeps during the time specified by the obtained value. After completion, it terminates immediately. If the value obtained is "null", it terminates immediately.
- **IoT APP**: An application for processing IoT data streams. It is composed of two steps:
    - Step 1: This step consumes messages published in the topic "stream_input" of the NebulOuS IoT message. The message payload is JSON with the structure {"group":string, "t":int}. When a message is received, the step sleeps the amount of time specified by "t". After that, publishes the same message in the topic "step1_output". Exactly one instance of this step must exist at any given time.
    - Step 2: This step consumes messages published in topic "step1_output". When a message is received, the step sleeps the amount of time specified by "t". There can exist as many instances of this step as necessary, all messages with the same value in "group" must be processed by the same instance of step 2.

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_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_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*

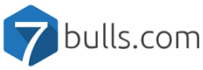
UC4_NF_07_TC1	*Redeploy Application Component after Edge worker node connection loss*
Objective	
• *NebulOuS shall redeploy an application component to the remaining Edge worker nodes once connection to one worker node is lost.*	
Preconditions	
1. *One or more worker nodes are active.* 2. *NebulOuS is 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


UBITECH
ubiquitous solutions


SEERC
SOUTH-EAST
EUROPEAN
RESEARCH
CENTRE


7bulls.com


Activeeon
SCALE BEYOND LIMITS


AUGMENTA
PRECISION AGRICULTURE · REDEFINED


ubiwhere
SUITING THE FUTURE


eurecat


mercabarna


ICCS
ΕΠΙΣΕΥ


existanze
#connectingdots


F6S


TTAnalysis


@fire


Telefónica


BIBA

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.