



# D6.1

## Integration guidelines & initial IoT-NGIN components integration

**WORKPACKAGE** WP6

**DOCUMENT** D6.1

**REVISION** V1.0

**DELIVERY DATE** 31/01/2022

**PROGRAMME IDENTIFIER** H2020-ICT-  
2020-1

**GRANT AGREEMENT ID** 957246

**START DATE OF THE  
PROJECT** 01/10/2020

**DURATION** 3 YEARS

## DISCLAIMER

This document does not represent the opinion of the European Commission, and the European Commission is not responsible for any use that might be made of its content.

This document may contain material, which is the copyright of certain IoT-NGIN consortium parties, and may not be reproduced or copied without permission. All IoT-NGIN consortium parties have agreed to full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the IoT-NGIN consortium as a whole, nor a certain party of the IoT-NGIN consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, and does not accept any liability for loss or damage suffered using this information.

## ACKNOWLEDGEMENT

This document is a deliverable of IoT-NGIN project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 957246.

The opinions expressed in this document reflect only the author's view and in no way reflect the European Commission's opinions. The European Commission is not responsible for any use that may be made of the information it contains.

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

<b>PROJECT ACRONYM</b>	IoT-NGIN
<b>PROJECT TITLE</b>	Next Generation IoT as part of Next Generation Internet
<b>CALL ID</b>	H2020-ICT-2020-1
<b>CALL NAME</b>	Information and Communication Technologies
<b>TOPIC</b>	ICT-56-2020 - Next Generation Internet of Things
<b>TYPE OF ACTION</b>	Research and Innovation Action
<b>COORDINATOR</b>	Capgemini Technology Services (CAP)
<b>PRINCIPAL CONTRACTORS</b>	Atos Spain S.A. (ATOS), ERICSSON GmbH (EDD), ABB Oy (ABB), INTRASOFT International S.A. (INTRA), Engineering-Ingegneria Informatica SPA (ENG), Bosch Sistemas de Frenado S.L.U. (BOSCH), ASM Terni SpA (ASM), Forum Virium Helsinki (FVH), Optimum Technologies Piloforikis S.A. (OPT), eBOS Technologies Ltd (EBOS), Privanova SAS (PRI), Synelxis Solutions S.A. (SYN), CUMUCORE Oy (CMC), Emotion s.r.l. (EMOT), AALTO-Korkeakoulusaatio (AALTO), i2CAT Foundation (I2CAT), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Sorbonne Université (SU)
<b>WORKPACKAGE</b>	WP6
<b>DELIVERABLE TYPE</b>	REPORT
<b>DISSEMINATION LEVEL</b>	PUBLIC
<b>DELIVERABLE STATE</b>	FINAL
<b>CONTRACTUAL DATE OF DELIVERY</b>	31/01/2022
<b>ACTUAL DATE OF DELIVERY</b>	02/02/2022
<b>DOCUMENT TITLE</b>	Integration guidelines & initial IoT-NGIN components integration
<b>AUTHOR(S)</b>	D. Skias (INTRA), A. Zalonis (INTRA), T. Velivassaki (SYN), A. Voulkidis (SYN), F. Maier (EDD), R. Farac (EDD), Y. Quan (SU), M. Pitz (RWTH), J. Klimt (RWTH)
<b>REVIEWER(S)</b>	T. Velivassaki (SYN), A. Su (SU)
<b>ABSTRACT</b>	SEE EXECUTIVE SUMMARY
<b>HISTORY</b>	SEE DOCUMENT HISTORY
<b>KEYWORDS</b>	Integration, Continuous Integration, Continuous Delivery, interactions, architecture, validation

## Document History

Version	Date	Contributor(s)	Description
v0.1	17/12/2021	INTRA, EDD	Toc and first draft
v0.2	27/12/2022	INTRA, SYN	Initial content in sections 2, 3 and 4
v0.3	11/01/2022	EDD	Update of content in Section 4
v0.4	17/01/2022	INTRA, EDD, SYN	Updates of content in sections 3 and 4
v0.5	21/01/2022	INTRA, SYN	Updates of content in sections 1, 2
v0.6	25/01/2022	INTRA, SYN, SU	Updates of content in sections 2, 3 and 4
v0.7	28/01/2022	INTRA, SYN, RWTH	Updates in sections 2 and 4
v0.8	31/01/2022	INTRA, SYN	Peer-review version ready, including updates on sections 1, 2, 3 and 5.
v0.8.1	01/02/2022	SYN	Peer review comments
v0.8.2	01/02/2022	SU	Peer review comments
v0.9	02/02/2022	INTRA	Updates based on peer review comments; Overall enhancements
v1.0	02/02/2022	INTRA, CAP	Quality check; Final version

# Table of Contents

Document History .....	4
Table of Contents .....	5
List of Figures .....	7
List of Tables .....	9
List of Acronyms and Abbreviations.....	10
Executive Summary .....	13
1 Introduction.....	14
1.1 Intended Audience.....	14
1.2 Relations to other activities.....	14
1.3 Document overview .....	15
2 IoT-NGIN Platform Architecture .....	16
2.1 IoT-NGIN Component-level Architecture .....	17
2.2 Process diagrams .....	21
2.2.1 Model update and operation of malicious attack detection in IoT.....	21
2.2.2 IoT vulnerability scanning.....	22
2.2.3 IoT Device Access .....	23
3 IoT-NGIN Integration infrastructure and guidelines.....	25
3.1 Integration Guidelines.....	25
3.1.1 Interfaces and data models .....	27
3.1.2 Documentation .....	28
3.2 Integration Plan .....	29
3.2.1 Integration Methodology .....	29
3.2.2 Integration phases and time-plan .....	30
3.3 DevSecOps practices and tools .....	31
3.3.1 CI/CD .....	32
3.3.2 Source Code Management.....	35
3.3.3 CI/CD in IoT-NGIN.....	35
3.3.4 Containers.....	39
3.4 IoT-NGIN Integration, Testing and Validation Infrastructure .....	40
3.4.1 Integration and Testing environment .....	40
3.4.2 Laboratory testing environment.....	44
4 Initial integration and validation .....	48

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

4.1	Early integration activities .....	48
4.2	Early validation activities .....	52
4.2.1	Preparation of 5G performance tests of the IoT-NGIN use cases with synthetic data streams.....	53
5	Conclusions.....	56
6	References.....	57
Annex 1	IoT-NGIN Use Case Questionnaire .....	58

## List of Figures

Figure 1 - The high-level IoT-NGIN architecture.....	16
Figure 2 - Component-level architecture of IoT-NGIN.....	20
Figure 3 – Indicative process diagram for model update and operation of malicious attack detection in IoT.....	22
Figure 4 – Indicative process diagram for IoT vulnerability scanning.....	23
Figure 5 – Indicative process diagram for IoT Device Access. ....	24
Figure 6 - IoT-NGIN Cloud Native development approach.....	27
Figure 7 – Development, integration, validation lifecycle. ....	29
Figure 8 - IoT-NGIN phased approach.....	31
Figure 9 – IoT-NGIN DevSecOps. ....	32
Figure 10 – CI/CD pipeline – Source: about.gitlab.com. ....	33
Figure 11 - CI/CD pipeline steps - Source: docs.gitlab.....	33
Figure 12 - SAST/DAST in the development lifecycle. ....	34
Figure 13 - IoT-NGIN GitLab Home page.....	36
Figure 14 - IoT-NGIN Enhancing IoT Cybersecurity and Data Privacy Subgroup. ....	36
Figure 15 - IoT-NGIN Gitlab runners' configuration. ....	37
Figure 16 - Snapshot of the project docker image repository. ....	38
Figure 17 - IoT-NGIN GitLab Issue Tracking.....	39
Figure 18 - SU OneLab (FIT NITOS & FIT IoT-Lab) as IoT-NGIN Integration, Test and Validation Infrastructure.....	40
Figure 19 - Kubernetes Dashboard.....	42
Figure 20 - Name Based Virtual Hosting.....	43
Figure 21 - Ingress Resource.....	43
Figure 22 - Ceph OSD.....	44
Figure 23 - EDD lab infrastructure planned for IoT-NGIN project. ....	45
Figure 24 - RWTH Real time laboratory, Real Time Simulators.....	46
Figure 25 - RWTH Real time laboratory, Person working on the PMU rack. ....	47
Figure 26 - DevSecOps processes with the build, SAST, test and deployment stages enabled. ....	48
Figure 27 - Example of a SAST job output – related to code quality analysis. ....	49
Figure 28 - Kubernetes resources consumed by WP6 activities.....	49
Figure 29 - Ingress controllers configured to be used by WP6 components.....	50
Figure 30 - Details of certificate automatically issued by the cert-manager of the cluster. ..	50

Figure 31 - Web application (Quorum blockchain web UI) exposed with the help of cert-manager and an Ingress controller. .... 51

Figure 32 - Kubernetes resources consumed by WP4 activities..... 51

Figure 33 - Kubernetes resources consumed by WP5 activities..... 52



## List of Tables

Table 1 - Production Cluster Configuration. ....	41
Table 2 - Secondary Cluster Configuration. ....	41
Table 3 - Production Cluster Endpoints. ....	42
Table 4 - Secondary Cluster Endpoints. ....	42
Table 5 - Smart Agriculture communication characteristics.....	54
Table 6 - Smart Industry communication characteristics. ....	54
Table 7 - Smart Energy communication characteristics.....	55

## List of Acronyms and Abbreviations

Aml	Ambient Intelligence
AR	Augmented Reality
CI	Continuous Integration
CD	Continuous Deployment
D2D	Device-to-Device
DIB	Decentralised Interledger Bridge
CNFN	Cloud Native Foundation
DAST	Dynamic Application Security Testing
DevSecOps	Development, Security and Operations
DLT	Distributed Ledger Technology
DT	Digital Twin
FL	Federated Learning
FQDN	Fully Qualified Domain Name
GAN	Generative Adversarial Network
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HPC	High-performance computing
HTTP	Hyper Text Transfer Protocol
IAS	IoT/ ML/ AR service
IDA	IoT Device Actuation
IDAC	IoT Device Access Control
IDD	IoT Device Discovery
IDI	IoT Device Indexing
IoT	Internet of Things
IT	Information Technology
MAD	Malicious Attack Detector

MCM	Machine Cloud Machine
ML	Machine Learning
MLaaS	Machine Learning as a Service
MTD	Moving Target Defence
NSMS	Network Slice Management System
NSM	Network Slice Manager
NTP	Network Time Protocol
OAS	OpenApi Specification
OS	Operating System
OSM	OpenSource MANO
OSD	Object Storage Device
PaaS	Platform as a Service
PATE	Private Aggregation of Teacher Ensembles
PMU	Phasor Measurement Unit
Pub/Sub	Publish-Subscribe
QR	Quick Response
REST	Representational State Transfer
RTS	Real-Time Simulator
SAST	Static Application Security Testing
SCM	Source Code Management
SDLC	Software Development Life Cycle
SDR	Software Defined Radio
SSI	Self-Sovereign Identities
TDD	Test-Driven Development
TSN	Time-Sensitive Networking
UC	Use Case
USRP	Universal Software Radio Peripheral
UWB	Ultra-Wide Band
VCM	Version Control System

VNF                Virtual Network Function

WP                Work Package

## Executive Summary

The focus of this document is set on the provision of integration guidelines, adhering to the Continuous Integration and Continuous Delivery practise, and on the illustration of the initial version of the IoT-NGIN components. Through IoT-NGIN architecture instantiation presentation, this deliverable aims on highlighting the main functionality of the platform, but most importantly IoT-NGIN components' and services' interactions. Furthermore, D6.1 describes the integration and testing infrastructure that enables the integration activities that will be performed in the context of the IoT-NGIN project and presents their early integration and validation activities.

# 1 Introduction

Integration and testing methodology is of major importance for every system that aims to be successful and thus addressing effectively its purpose and role. A well thought solution that aims to be materialized in a system or a platform or even a service needs to be supported by proper integration and testing activities. These activities focus on bringing together all the development outcomes into a unified and effective system. Section 2 presents the high-level IoT-NGIN architecture, highlighting the main functionalities of the platform, as well as the components which deliver such functionalities. Based on this architecture and the integration guidelines that will be introduced later in this document, an effective integration (and testing) strategy is defined for IoT-NGIN. Furthermore, as suitable infrastructure, its tools and frameworks are key enablers for the integration and testing activities, the present document introduces those within the project's CI/CD lifecycle.

Initial integration and validation activities have been conducted within the first period of the project. Initial integration has been based on components' availability and the project's CI/CD tools, while the initial validation refers to preparatory activities for 5G performance tests.

This document (D6.1) is the first iteration of the three deliverables that will be generated in the context of WP6 and aim to describe the integration and validation activities of the IoT-NGIN project. The next deliverable D6.2 "Integrated IoT-NGIN platform & laboratory testing results" in the last quarter of 2022 and finally D6.3 "Interoperable IoT-NGIN meta-architecture & laboratory evaluation" will be produced in the second quarter of 2023.

## 1.1 Intended Audience

The intended audience of this document is primarily the consortium of the IoT-NGIN project, including the partners that will be later introduced into the consortium and will be part of it through the project's Open Calls (#1 and #2). More specifically, this deliverable presents to the technical partners of the consortium, the integration and testing environment of the IoT-NGIN platform, the integration guidelines that need to be adopted through the development, integration and testing lifecycle of the project and the IoT-NGIN architecture instantiation that is derived by the IoT-NGIN Meta-Architecture, as defined in deliverable document D1.2 "IoT meta-architecture, components and benchmarking" [1].

## 1.2 Relations to other activities

As already stated above, integration and testing activities act as the backbone of the system development process. Thus, this document is strongly connected with all the technical WPs (2-6). Furthermore, it is also related to WP7 that consists of the Living Lab validation work, dictated by the IoT-NGIN specific Use Cases. Finally, it is also related with WP8 and WP9 in the context of impact creation and Open Calls, respectively.

## 1.3 Document overview

The present document provides to the reader insights on the Integration guidelines, frameworks and tools, on the integration plan, and finally offers the IoT-NGIN platform architecture that is derived by the IoT-NGIN meta-architecture as defined within D1.2 and by the respective development activities of the Project's technical work packages.

Section 1 provides an introduction of the deliverable. It describes the intended audience of this document and provides an overview of its content.

Section 2 presents the IoT-NGIN architecture instantiation, through a component-level diagram that also incorporates components' interactions. Lastly it provides some indicative process diagrams that represent the flow of the information between the involved IoT-NGIN components.

Section 3 illustrates the integration and testing guidelines, frameworks and tools that are proposed to the IoT-NGIN technical partners and aim to facilitate the development and integration activities of the project. In addition, it contains information regarding the projected release plan of the IoT-NGIN implemented platform and presents the DevSecOps practices that will drive the development and integration activities of the project and on the relevant CI/CD pipeline. This section also elaborates on the integration and testing environment of the project, where the integration and validation testbeds that are available for the project are described. In addition, the specifications of the clusters that support IoT-NGIN integration activities are also depicted.

Section 4 presents the project's early integration and validation activities. The IoT-NGIN software development life cycle (SDLC) control is shown, implementing the DevSecOps processes. Moreover, the preparation of 5G performance tests in Ericsson Eurolab are presented.

Section 5 presents the conclusions and outcomes that are derived by the current document.

Finally, Annex 1, depicts the IoT-NGIN Use Case questionnaire descriptions that was circulated to the relevant Living Lab involved partners, in order to investigate and subsequently prepare and design the synthetic data streams.

## 2 IoT-NGIN Platform Architecture

As IoT-NGIN aims to drive the path to the next generation IoT, it provides tools and services to support the development, deployment and operation of next-generation internet and IoT services. The IoT-NGIN tools build upon the Meta-Architecture defined in D1.2, which, in the course of the project, is materialized in the high-level architecture depicted in Figure 1, as has been also presented in D1.2.

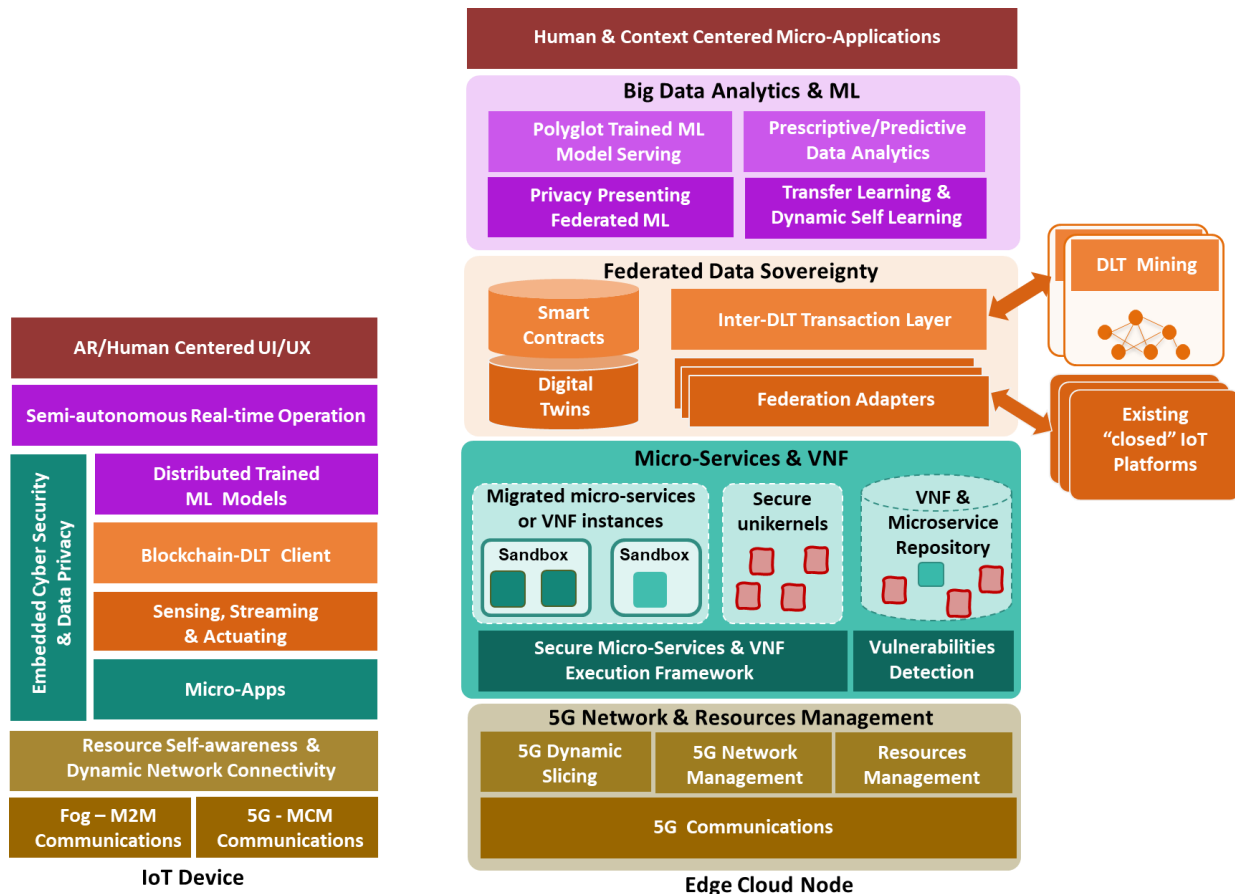


Figure 1 - The high-level IoT-NGIN architecture.

The high-level IoT-NGIN architecture is the basis for the integration of the IoT-NGIN platform components and incorporates five functional layers, each one illustrated by a distinct colour:

- **Federated Communications (brown colour)**, which includes communication and resources management functionalities, pertaining to IoT and 5G communication, as well as virtual networking management tools.
- **Micro-services and Virtual Network Functions (VNFs) (turquoise colour)**, which provides a “living” collection of functionalities, implemented as container-based micro-services or unikernels, which will be empowered by IoT-NGIN's novel Secure Micro-Services Execution Framework.
- **Federated Data Sovereignty (orange colour)**, which leverages some of the emerging Decentralised Identifier (DID), Self-Sovereign Identity (SSI) technologies and an inter-



Distributed Ledger Technologies (DLT) approach to enhance data sovereignty, privacy and security of data sharing.

- **Federation of Big Data Analytics & ML (purple colour)**, which relies on the IoT-NGIN ML as a Service (MLaaS) framework, offering ML/AI models or inference services to both the IoT-NGIN platform components and the hosted IoT or ML services, supporting polyglot model serving. Moreover, ML- and zero-knowledge proof-based cybersecurity components based on Generative Adversarial Networks (GANs, [2]) are developed to identify malicious IoT nodes and data poisoning attacks, as well as timely identify intrusion detection.
- **Human-Centred Augmented Reality Tactile IoT (dark red colour)**, which include the Ambient Intelligence components of IoT-NGIN responsible for identifying or recognizing an IoT device recognition, indexing, access control and actuation.

## 2.1 IoT-NGIN Component-level Architecture

The high-level architecture of IoT-NGIN is defined at a finer grade in a component-level architecture, the implementation of which will provide the instantiation of the IoT-NGIN platform. The IoT-NGIN components are briefly described in this section, based on the functional layer they belong to. More details about the specifications of the components can be found in D2.x, D3.x, D4.x and D5.x deliverables.

The **Federated Communications** layer includes the following components:

- *5G Coverage Extension*, which relies on Device-to-Device (D2D) communications to extend the cellular coverage. This component achieves coverage extension through one-hop relays to devices, considering link measurements and network dynamics.
- *Network Slice Manager (NSM) and Time-Sensitive Networking (TSN) Bridge*, which aim to enhance connectivity of Machine Cloud Machine (MCM) networks. NSM is provided as an implementation of 3GPP Network Slice Management System (NSMS) and allocates the required resources for creating end-to-end network slices. Moreover, the TSN Bridge is implemented as a TSN Application Function, which will transform 5G into TSN end-to-end system
- *5G resource management API*, which exposes 5G and IoT capabilities related to 5G Connectivity and Device Management, Network Slice Management, Microservice Management and Service Migration.

The **Micro-services and VNFs** layer include:

- The *Secure Edge Cloud Execution*, which supports micro-services offloading and service migration execution, properly integrating with Kubernetes and OpenStack, as well as OpenSource MANO (OSM) [3] APIs, for microservices creation and network services instantiation and lifecycle management, respectively. The secure execution framework will offer strong isolation based on virtual machines and reduced overhead based on Library Operating Systems (LibOS).
- The *IoT Vulnerability Crawler*, which is a distributed service which scans both IoT devices and services against a group of vulnerabilities organized in service-oriented plugins. This component, which is implemented as a stateless microservice, schedules vulnerability scans, either on demand, or following a schedule, orchestrates the spawning of vulnerability scanning jobs against the devices and performs a scan

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

against a device or service, against a particular grouped set of vulnerabilities, as managed by a scan plugin.

- The *Moving Target Defence (MTD) network of Honeypots*, which allows exploration of attackers' behaviour, exploiting IoT systems' vulnerabilities. A honeypot is a decoy computer system that appears attractive to an attacker and can be used to collect information on threat behaviour and vectors. MTD dynamically changes the attack surface to continuously increase complexity and confuse the attacker, thus preventing the system vulnerabilities from being exploited. In IoT-NGIN, it can be used to mimic vulnerabilities identified by the IoT vulnerabilities crawler.

The **Federated Data Sovereignty** layer is composed of:

- The *Decentralised Interledger Bridge (DIB)*, which enables a general-purpose atomic data transfer across heterogeneous distributed ledgers without requiring any changes to ledgers themselves, while providing support for coordination of distributed interledger nodes.
- The *Self-Sovereign Identities (SSI)*, which implements a privacy-preserving SSI solution, which uses Decentralized Identifiers (DIDs) for identification of the Resource Owner and the Client, as well as Verifiable Credentials (VCs), indicating the Client's right to access some Resource. The SSI component provides support for users to express that they have given consent to disclose certain information to a certain party.
- *Digital Twin(s) (DT)*, which act as a mediation layer to physical resources or devices. The Digital Twin may be used for monitoring operations, providing access to sensor data and allowing data analytics or ML operations on them, as well as for actuation services, channelling actuation control commands to physical devices, or for secure execution/deployment of microservices to physical devices, as instructed by the *Secure Edge Cloud Execution* framework.
- The *Meta-Level Digital Twin*, which acts as a semantic twin for adding metadata to digital twins. A Semantic Twin consists of a Twin ID and a twin description document, which constitutes a text-format file that actually describes a digital twin. The Semantic Twin uses SSI technologies that can either be built with DLTs or allow the use of DLTs as an additional feature.

The **Federation of Big Data Analytics & ML** layer consists of the following components.

- The *ML as a service (MLaaS) framework*, which supports the complete ML lifecycle. It provides support for handling the data from the data sources, including loading the data according to templates, performing data processing and storing the data in edge nodes. It also provides functionalities to deal with AI models, including training, deploying, optimizing, and sharing. Moreover, the *MLaaS framework* provides a Development Environment as the entry point or any user where, through a set of APIs and/or a graphical user interface (GUI), it will grant usage of the rest of the platform components and functionalities from a common place. The data sources of the *MLaaS framework* could be either IoT devices or Digital Twins, which provide data in batches or in a stream. The *MLaaS framework* integrates with edge and cloud resources, supporting the edge computing paradigm.

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

- The *Polyglot Model Sharing* component, which is part of the MLaaS framework, is responsible for sharing the model with third parties or other external users in different AI frameworks (polyglot).
- The *Deep Learning, Reinforcement Learning & Transfer Learning framework* provides enhancements in the model's training processes and automatic improvements in the resulting models. It will execute incremental learning, online learning and/or automatic data labelling algorithms on selected pilots suitable for them. Moreover, it supports acceleration of the operations executed by the models, especially when the operational environment is at the edge of the network.
- The *Privacy-preserving Federated Machine Learning*, which enables ML developers to use an appropriate FL technique to train their models, without having to delve into the technique's specifics, while supporting different privacy-preserving techniques, leveraging on Flower, PySyft and PyGrid, TensorFlow Federated and Private Aggregation of Teacher Ensembles (PATE).
- The *Generative Adversarial Network (GAN)-based IoT attack dataset generator*, which generates high-value synthetic datasets of attacks, using a small portion of real data and preserving the utility and fidelity of real datasets. In IoT-NGIN, GANs are exploited in generating datasets implying data and model poisoning attacks, as well as network level attacks for IoT systems participating in FL configurations. Such synthetic datasets can be used to compensate for the unavailability of appropriate training datasets for ML anomaly detection models.
- The *Malicious Attack Detector (MAD)* component, which is an ML-based anomaly detection module able to identify attacks in on-device FL. The types of attacks or anomalies identified depend on the training dataset, which will rely on both real and synthetic data. Considering an FL system, a MAD instance can be placed in each FL node, either on the edge nodes participating in FL as contributors of their local ML model updates or on the Aggregator node, which calculates the global (aggregated) model and resides at edge or cloud resources.

The **Human-Centred Augmented Reality Tactile IoT** layer includes the following components:

- *IoT Device Discovery (IDD)*, which is responsible for identifying or recognizing an IoT device, based on conventional methods, such as Quick Response (QR) scanning, or advanced visual methods such as combining computer vision based recognition with Ultra-Wide Band (UWB) based localization.
- *IoT Device Indexing (IDI)*, which provides information related to IoT devices' status and characteristics. IDI can be used to collect device information by the physical device and appropriately update its digital twin.
- *IoT Device Access Control (IDAC)*, which provides controlled access to the IoT devices, considering a permission- and role-based mechanism applied before any activity on the device by any user.
- *IoT Device Actuation (IDA)*, which enables the actuation controls to be applied on the IoT devices automatically or as a result of (advanced) human interaction.
- *IoT/ ML/ AR service (IAR)*, which refers to services built on top of the IoT-NGIN tools. They could expose IoT, ML or Augmented Reality (AR) functionality. For the AR case, it will exploit the *IoT AR assets' repository*, which will be implemented within IoT-NGIN and will include software components from widely accepted AR and gaming platforms.

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

The high-level interactions of the IoT-NGIN components as described above are depicted in Figure 2, which will guide the integration activities.

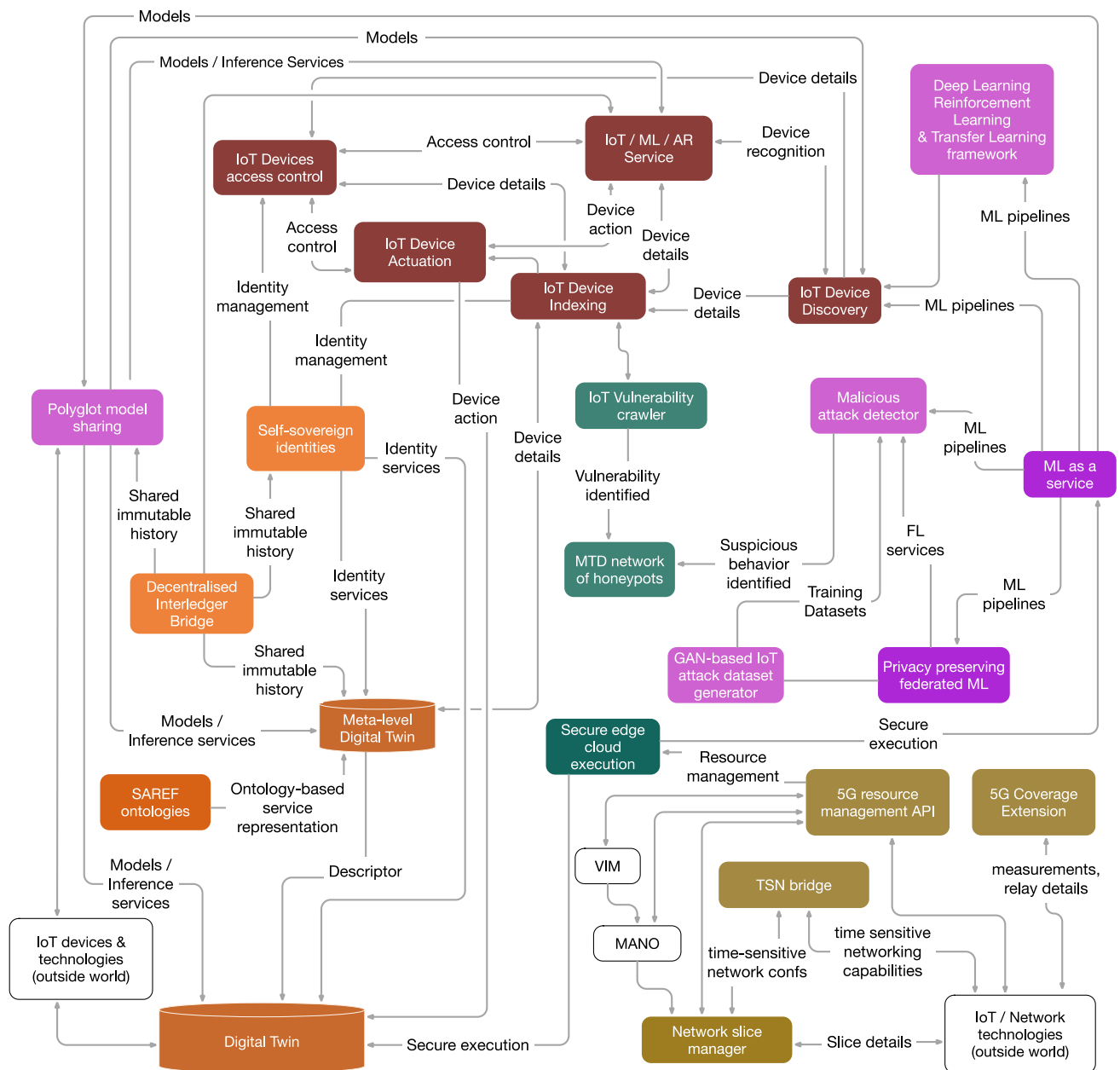


Figure 2 - Component-level architecture of IoT-NGIN.

As shown in the figure, there is a strong interconnection among the IoT-NGIN components, which allow to support the project's main objectives, summarized as:

- Optimization of IoT/M2M and 5G/MCM communications, including secure-by-design micro-services to extend the edge cloud paradigm
- Enabling user and self-aware, autonomous IoT systems, supported by Artificial Intelligence, including privacy-preserving federated ML, and ambient intelligence (Aml), with AR support for humans
- Ensuring distributed IoT cybersecurity and privacy, leveraging on SSIs, interconnected DLTs and Meta-Level Digital Twins.

Hence, the component-level architecture provides the basis for the development of infrastructure and tools to support the next-generation of IoT.

## 2.2 Process diagrams

In this section, indicative processes enabled by the IoT-NGIN architecture are analysed, highlighting interactions among components belonging to the same or different functional layers. Specifically, the following indicative processes are analysed:

- Model update and operation of malicious attack detection in IoT
- IoT vulnerability scanning
- IoT device access

The selected processes demonstrate the presence of cybersecurity by design in different aspects of IoT systems, while involving ML based operations and AML support. The list of processes is indicative, yet not exhaustive, since sequence diagrams for processes specific to the functional layers have been developed in the scope of WP-level deliverables for WP2-WP5 and WP7.

### 2.2.1 Model update and operation of malicious attack detection in IoT

IoT devices, residing at the edge of potentially complex networks and systems, often attract malicious actors and lead to cybersecurity threats or attacks. IoT-NGIN considers cybersecurity at the edge, providing tools for:

- IoT device vulnerability scanning, which may proactively tackle potential attacks,
- Attack monitoring through MTD networks of honeypots
- IoT attack detection, based on ML models for anomaly detection
- Building and updating ML models (applied here for the IoT attack detection models), which are trained cooperatively by distributed nodes, ensuring security and privacy of both data and models
- Generating much valued training datasets, which are not available otherwise.

Figure 3 presents the basic operations for training and providing the ML model for attack detection and the creation of honeypots based on identified vulnerabilities and detected attacks. Indicatively, let assume that an ML developer requests data generation from the *GAN-based IoT attack dataset generator*, in order to use it as a training dataset for attack detection. The generated data are provided to the *MLaaS* platform and the ML developer asks *MLaaS* to retrain their attack detection model, using these data and employing federated learning. Thus, training is performed by *Privacy-preserving federated ML*. As soon as the model is trained, the *ML Developer* requests to provide the model itself or the inference services to the *Malicious Attack Detector*, which is done via the *Ployglot Model Sharing* component. The *Malicious Attack Detector* performs inference, and, upon detection of malicious detection, it notifies the *MTD Network of Honeypots* accordingly. The latter may also receive information about IoT vulnerabilities identified by the *IoT Vulnerability Crawler*. It finally uses both types of information to expose relevant vulnerabilities in a new honeypot.

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

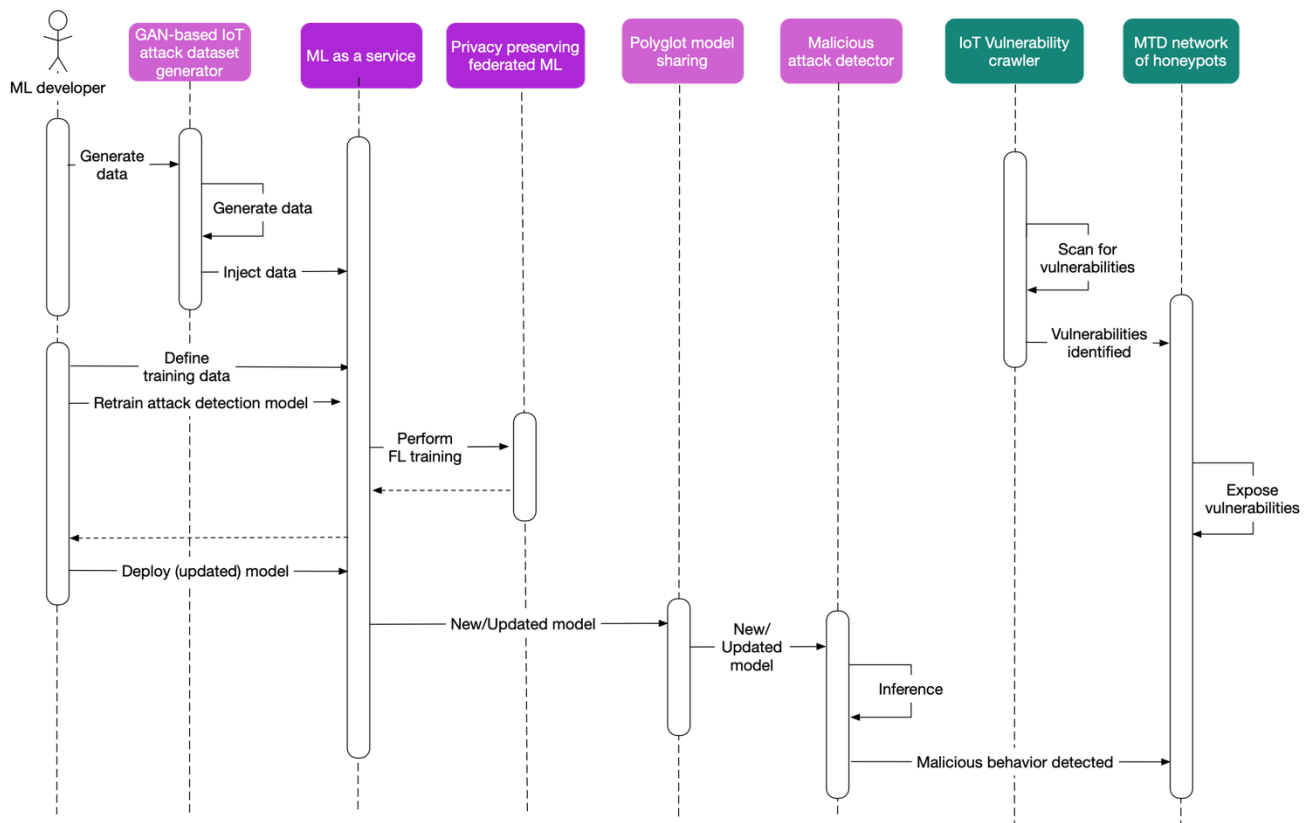


Figure 3 – Indicative process diagram for model update and operation of malicious attack detection in IoT.

With this process, it is possible to both identify potential attacks, using an updated model, but also exploit the collected information about cyberthreats in order to monitor attackers' behaviour in low-secured network zones of honeypots. This information is useful as feedback to cyber-threat management and risk assessment, which might be applied in any business having deployed IoT systems.

## 2.2.2 IoT vulnerability scanning

In this process, a closer look to the vulnerability scanning functionality and its role within the IoT-NGIN system is presented. As shown in Figure 4, the *IoT Device Indexing* module provides device details for newly registered devices in the IoT-NGIN platform to the *IoT Vulnerability Crawler*. The crawler considers this information while generating scan schedules. Then, it spawns specific scanning jobs, based on the vulnerabilities and the available devices, and performs the scheduled scans. Upon detection of vulnerabilities, the crawler outputs its results to the *MTD network of Honeypots*. As a next step, a honeypot will be created, being configured to have the identified vulnerabilities open.



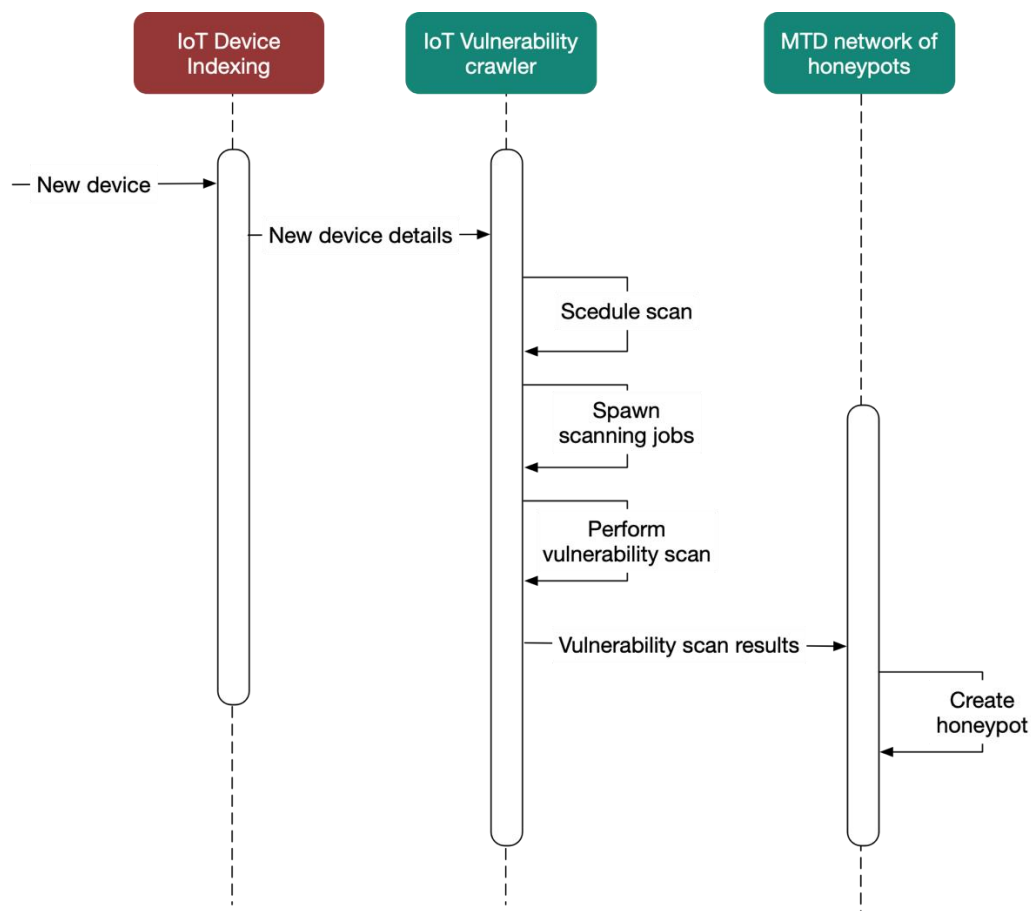


Figure 4 – Indicative process diagram for IoT vulnerability scanning.

This process indicates that any device participating in the IoT-NGIN platform is monitored on a scheduled or on-demand basis for potential vulnerabilities, while the integration of new vulnerabilities in the Honeypot network allows for increasing the preparedness against cyberattacks in systems exposing such vulnerabilities.

## 2.2.3 IoT Device Access

The access to information or to control actions in IoT devices must be duly secured in the next generation of IoT systems. The process presented in Figure 5 describes the IoT-NGIN operations for allowing an *End User* to perform a defined action to a specific device, such as receiving monitored data or applying some control or configuration action. The *End User* requests to access a certain device through an *IoT/ML/AR Service* and this request is forwarded to the *IoT Device Indexing* module, including information about the user, the device and the desired action. The request is then forwarded to the *IoT Device Access Control* module, in order to ensure that access is possible only after proper authentication and authorization. The request includes also positioning information, added by the *IoT Device Indexing* module, since e.g. access for users requesting AR services should be granted only to users in close proximity to the device. Based on the authentication scheme considered, the *IoT Device Access Control* module may check the provided credentials' validity or ask

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

the *Self-Sovereign Identities* component to check it. In case the validity check fails, the result is communicated to the *End User* via the *IoT Device Indexing* and the *IoT/ML/AR Service*. In case the authentication is successful, the request is resolved either by the *IoT Device Indexing* module or via the *Digital Twin*, in case access to the devices is done through it. Finally, the user is presented with the results via the *IoT/ML/AR Service*.

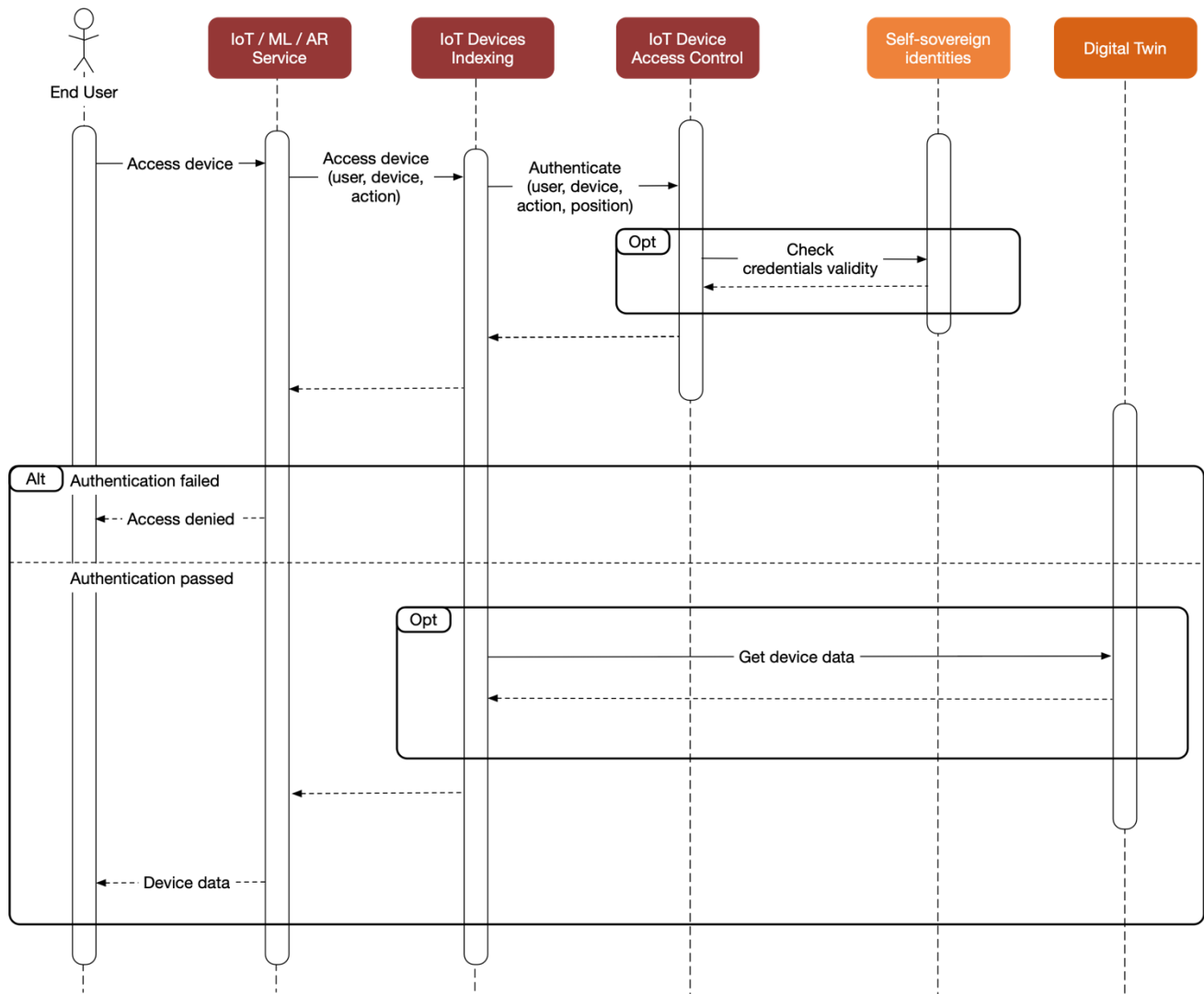


Figure 5 – Indicative process diagram for IoT Device Access.

Through this process, IoT-NGIN is flexible to support different configurations of IoT systems, having incorporated DT solutions or not, integrating different authentication systems and allowing various operations on different types of devices. Through this, IoT-NGIN may support a multitude of applications in different domains.



## 3 IoT-NGIN Integration infrastructure and guidelines

This section analyzes the integration guidelines, the integration plan, the underlying infrastructure, the integration framework, platforms and tools utilized and, last but not least, the IoT-NGIN integration and testing environment. The following subsections will provide a deeper analysis on each of the integration and testing environment elements.

Software development, integration and testing, given its significance and inherent complexity, can be a very cumbersome task if not approached in the right way from the beginning. The task becomes even more challenging when the developing teams participate in these activities remotely. Thus, a proper integration guide must be produced and adopted by the all the technical partners before even the start of any development activity.

Prior to introducing the IoT-NGIN platform integration guidelines, we need to describe the core elements upon which the integration activities will be performed. The IoT-NGIN platform consists of purpose-specific frameworks and platforms (e.g. Big Data Analytics and ML) that also include their own modules, as components and services. In addition, the IoT-NGIN platform contains micro-services that will be situated in the secure edge cloud environment and Use Case specific tailored services that will be developed in the context of T6.2 and aim to enhance the application logic of the IoT-NGIN Use Cases. The latter will be documented in detail in D6.2 "Integrated IoT-NGIN platform & laboratory testing results", due on the last quarter of 2022.

It should be highlighted that although IoT devices are of significant essence in the context of the IoT-NGIN project and a significant number of IoT devices will be participating in the integration activities for the validation of the functionality of various IoT-NGIN components, the part of integration that adheres to hardware IoT devices will not be addressed by this deliverable. Therefore, any necessary integration activity (between an IoT device and an IoT-NGIN module) shall be conducted by the technical party that is responsible for the production of a service that utilizes IoT devices. Then, the service or the component itself as will be integrated into the IoT-NGIN platform will be the one that will be monitored and subsequently tested, integrated and evaluated.

### 3.1 Integration Guidelines

Software engineering offers a multitude of design techniques that a developer might prefer to adopt. While no technique can be considered as the best one for any case, taking into account specific requirements, limitations and prerequisites that are introduced on each project, some are more advantageous than others in specific cases. As already stated, the essence of the IoT-NGIN platform from a technical perspective lies in its core components and services. Those core elements can be described as modules that adhere to a specific functional design, driven by the IoT-NGIN architecture. Such modules are being implemented by disperse and possible diverse software developing teams. Once developed, they should interoperate with each other in order to deliver pieces of the required functionality that IoT-NGIN envisages to deliver. Consequently, modular design is the design principle of choice that is strongly suggested to the IoT-NGIN developers. Modular

software design practice dictates that each implemented module shall be an independent, self-contained building block that exposes its functionality through a set of well-defined and documented interfaces. Following this approach, each technical team of the project would only need to be aware of the external interfaces exposed by each developed module, thus abstracting arbitrary complexity that otherwise, inevitably would be introduced. In summary, modular design advocates:

- The division of a system into modules that are independent from each other
- The division of each module into two parts:
  - Interface, which in an abstract level consists of anything that must be known to other modules. This partition, apart from the offered functionality, includes also some relevant documentation and comments.
  - Implementation, which consists of the source code that delivers the functionality promised by the interface.

In summary, the ultimate goal of modular software design is to define isolated and simple abstractions where the interface should be much simpler than the implementation and at the same time guaranteeing that if a change affects only a module's implementation but not its interface, then required fix or update of a module source will not affect any other modules of the system.

Modular design practice combined with DevSecOps and automation tools will enable the IoT-NGIN project to take full advantage of the cloud computing model through cloud-native technologies.

As defined by Cloud Native Foundation (CNFN), the "Cloud Native" definition<sup>1</sup> is expressed as:

*"Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach."*

*These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil."*

This definition is considered to be deeply aligned with the IoT-NGIN project objectives with respect both to the resulting platform, but also to the supporting cloud infrastructure that enables the development, integration and validation activities.

---

<sup>1</sup> <https://github.com/cncf/toc/blob/main/DEFINITION.md>

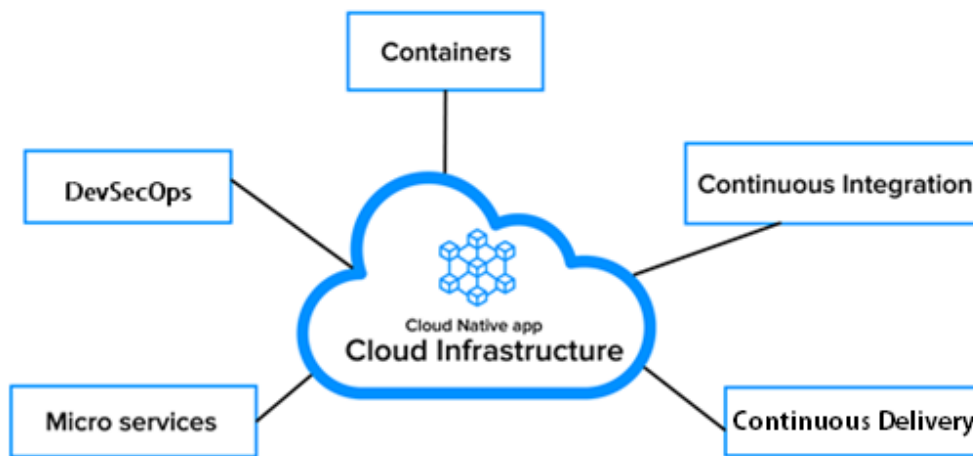


Figure 6 - IoT-NGIN Cloud Native development approach.

The aforementioned design approach is further complemented with an additional set of key recommendations, as guidelines, which are proposed to the consortium technical teams aiming to greatly ease the integration process, rendering it a lot faster and at the same time more effective.

### 3.1.1 Interfaces and data models

Apart from the modular design practices which are particularly well suited to cloud technologies and automations promoting the smooth integration of a large platform or system, components' interfaces and data models are also of major importance. Data models describe the format and the structure of the information that is being exchanged between two or more components through their interfaces. Data models are visual representations of data elements and the connections between them. Therefore, a data model describes not only the structure of the data, such as types or classes of data, but also any relationships between the data that are handled by the system. It is also suggested, where applicable, to adhere to standardized data models (e.g. FIWARE NGSI-LD [4]). This will further facilitate the open exchange and sharing of structured information between the IoT-NGIN stakeholders, enabling interoperability of the produced solutions.

#### 3.1.1.1 RESTful architecture

With respect to the topic of interfaces, standardized interfaces (e.g. RESTful API) shall also be adopted where applicable. REST stands for REpresentational State Transfer and is a stateless client-server software architectural style that relies on rules that describe how to define and access resources. The communication between the client and the server is conducted via the Hyper Text Transfer Protocol (HTTP). REST uses the HTTP protocol's request types (POST, GET, PUT, and DELETE) to allow users to Create, Read, Update, and Delete via a RESTful API.

The adoption of the RESTful architectural paradigm provides a number of significant benefits which can be summarized below:

**Lightweight and language independent:** Relies on the HTTP standard, which means it is format-agnostic and developers can use XML, JSON, HTML, etc., making the implementation of a RESTful API really fast.

**Scalable and flexible:** Thanks to the separation between client and server, a module may be scaled by a development team without much difficulty. In addition, developers can also easily integrate REST APIs without much added work. Provided that the data from one of the requests are properly sent, it is possible to perform a migration from one server to another or carry out changes on the database at any time without much difficulty. For example, it is easy to host the front-end and the back-end of an application on different servers.

### 3.1.1.2 Publish/Subscribe pattern

In addition to the REST architectural software style, the publish-subscribe messaging pattern, so called Pub/Sub, is also proposed to be utilized by the IoT-NGIN developers. Pub/Sub is an architectural design pattern that provides a framework for exchanging messages between publishers and subscribers. This pattern involves the publisher and the subscriber relying on a message broker that relays messages from the publisher to the subscribers. The major benefits of the Pub/Sub paradigm can be summarized as follows.

**Asynchronous:** Pub/Sub is asynchronous, therefore, once a request is sent by an application there is little risk of performance degradation due to a process getting stalled (time-out) in a long-lasting data exchanging communication.

**Scalable and Flexible:** Adding or removing subscribers to a topic is a matter of configuration. No complex programming is required. Thus, Pub-Sub systems provide a great deal of scalability and flexibility.

Finally, adhering to common data models through standardized interfaces, strongly promotes project's openness towards the Open Source community, addressing a significant requirement of the IoT-NGIN platform. On one hand, IoT-NGIN aims to leverage on software solutions or modules that are developed and distributed under an open-source model, including open source solutions produced by H2020 projects accelerating the developments of the consortium, using already available and reliable solutions and on the other, IoT-NGIN project to provide its developed solutions as open source by-design, and thus greatly contribute to the European IoT community.

## 3.1.2 Documentation

Proper documentation, in general, facilitates greatly the development and integration activities. It can be approached as a vertical necessity for any project, especially in a complex one such IoT-NGIN, and should be part of every development and integration step. Starting with source code documentation, it which greatly helps to comprehend a piece of software that is written even by the same developer at some earlier time and especially if it has been written by someone else. In addition, source code documentation, significantly eases a future update or correction activity within module's code, where source code "comments" will help the developer to comprehend the code's functionality and avoid mis-

conception mistakes that could easily occur. In general, it is a very good practice to be adopted by the project's developers.

Furthermore, API documentation is also considered of significant importance, especially for integration purposes. API documentation allows a component user to understand how to use the exposed services. Adhering to the common approaches described previously, OpenApi Specification (OAS), defines a standard, programming language-agnostic interface description for RESTful APIs, which allows to discover and understand the capabilities of a service, without requiring access to source code, additional documentation or inspection of network traffic [5]. In IoT-NGIN, developers are strongly recommended to utilize online documentation tools (e.g. Swagger) for the documentation of the developed APIs.

### Open Source

Finally, it worth stressing that the IoT-NGIN platform is envisaged to be offered as Open Source, thus the proper adoption of standardized practices and the proper documentation of the project's technical outputs is vital. Therefore, IoT-NGIN implemented components shall be clearly described via detailed documentation. Documentation shall describe each component's goals, its provided functionality, technical specification of the programming interfaces (APIs) utilized, deployment information and example-based instructions that pertain to the actual usage of the component.

## 3.2 Integration Plan

### 3.2.1 Integration Methodology

In IoT-NGIN, the integration methodology aims at guaranteeing that the delivered components from the technical work packages are successfully integrated, thus inter-communicating and inter-operating with each other.

#### 3.2.1.1 Integration Cycle

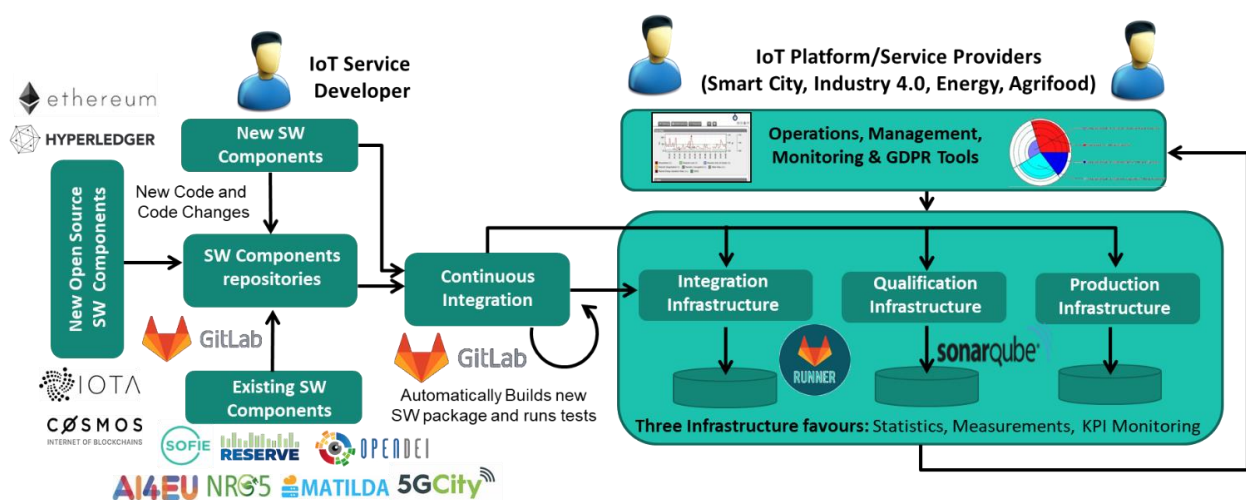


Figure 7 – Development, integration, validation lifecycle.

As illustrated in the Figure 7, IoT-NGIN aims to incorporate components that will be developed by the consortium partners, but also, existing components from open source projects (e.g. Ethereum, Hyperledger Fabric), and possibly components produced by past H2020 projects (e.g. SOFIE).

DevSecOps software development practices (e.g. Continuous Integration) will be incorporated into the loop to ensure agile and seamless integration and testing. Gitlab is used for software components versioning and as the core continuous integration driver (Gitlab-CI). DevSecOps practices applied in IoT-NGIN are presented in section 3.3 in more detail.

As extensive penetration and other testing is essential for hassle-free integration, component (unit) and service-level tests will be given strong focus and will comprise simple and scripts-based functionality and operational testing. More specifically, IoT-NGIN will perform testing as follows.

**Component testing** (also known as unit testing) searches for defects in, and verifies the function of, software modules programs. It may be done in isolation from the rest of the system, depending on the context of the development life cycle. In IoT-NGIN, separate component tests will be planned and executed in each technical work package.

**Integration testing** examines the interfaces between components, interactions with different parts of a system and interfaces between systems. Systematic integration strategies may be applied as appropriate testing integrated components' functional tasks, transaction processing sequences or some other aspect of the system or components. In order to ease fault isolation and detect defects early, integration should normally be incremental rather than "big bang". In IoT-NGIN, integration testing should be scheduled prior of each release cycle, in order to ensure the expected way of operation of the platform.

**System testing** is concerned with the behaviour of a whole product. In system testing, the test environment should correspond to the final target or production environment as much as possible in order to minimize the risk of environment-specific failures not being found in testing. System testing may include tests based on risks and/or on requirements specifications, business processes, use cases, or other high-level text descriptions or models of system behaviour, interactions with the operating system, and system resources. In IoT-NGIN this level of testing should be scheduled during product releases and is expected to be facilitated by the IoT-NGIN demonstrators.

Before each major or minor release cycle, the qualification infrastructure (a subset of each living lab) can be potentially used to test the upgrade from the previous development version to the new one, allowing the identification and elimination of compatibility issues. The qualification infrastructure serves as an entry point to test bug fixes without interfering with the development of new releases.

Finally, the bug free and integrated version of the IoT-NGIN platform will be deployed to each Living Lab infrastructure in order to validate the relevant Use Cases.

### 3.2.2 Integration phases and time-plan

IoT-NGIN implementation will be established in 3 phases. These phases are briefly described below. As depicted in the figure, IoT-NGIN components' implementation and integration



## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

activities are primarily contained within all three phases. Each IoT-NGIN integrated platform release takes place prior to the end of each phase and is tested and validated by the end of each phase. Thus, the initial integration of the IoT-NGIN platform (ver.0) will be released within the first phase, the first integrated version of the IoT-NGIN platform (ver.1) will be released within the second phase and the second integrated version of the IoT-NGIN (ver.2) will be released within the third phase. Although the projected major releases are two (2), additional integration sub-versions may also be introduced if necessary.

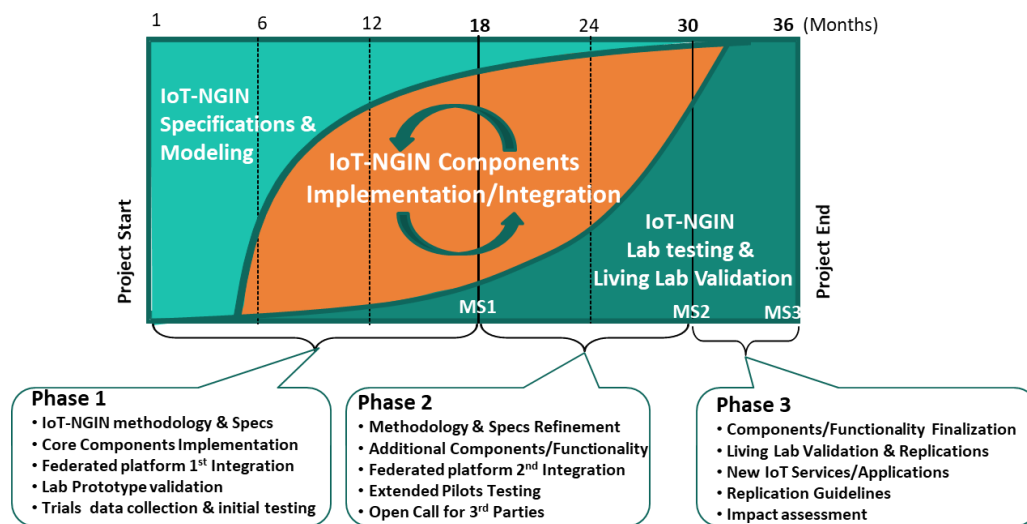


Figure 8 - IoT-NGIN phased approach.

The first phase concerns the specifications & modelling generation. These processes have already been undertaken by the specific technical work packages of the project, namely WP2-WP6. Within each WP, the specifications of each solution are defined and subsequently the proposed solution is modelled. The resulted developed components will then be integrated forming the 1<sup>st</sup> integration of the IoT-NGIN platform. Finally, the 1<sup>st</sup> version of the platform will be initially tested and validated. The outcomes of these tests will fuel the second phase of the IoT-NGIN platform development.

The second phase aims to refine the 1<sup>st</sup> integrated version of the IoT-NGIN platform. This will result in the 2<sup>nd</sup> integrated version of the IoT-NGIN platform. The 2<sup>nd</sup> integrated version will also include the integration activities that pertain to the Open Call for 3<sup>rd</sup> parties (1 and 2).

Finally, the third phase of the IoT-NGIN platform development focus on the Use Case oriented Living Lab validation of the IoT-NGIN platform.

### 3.3 DevSecOps practices and tools

IoT-NGIN will adopt DevSecOps practices to support the platform's integration activities and the management of the foreseen IoT-NGIN platform releases. DevSecOps in plain words is defined as security enhancements over the DevOps cycle. DevOps is a set of practices that complement Agile Software development, and which combines software development and Information Technology (IT) operations. More specifically, it aims to accelerate the systems development life cycle and ultimately provide continuous delivery with high software quality. DevSecOps incorporates the element of security by design into DevOps, meaning that infrastructure security is incorporated into the loop from the beginning and by-design.

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

DevSecOps, short for Development, Security, and Operations, automates the integration of security at every phase of the software development lifecycle, from initial design through integration, testing, deployment, and software delivery [6]. It implies that security becomes an integral part of the agile DevOps practices and methods such as continuous delivery, continuous integration, and collaboration.

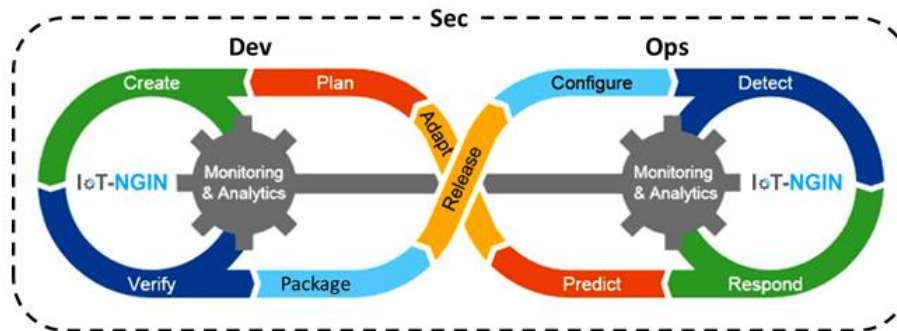


Figure 9 – IoT-NGIN DevSecOps.

The best practices for DevSecOps, as identified by IBM in July 2020 [6] may be summarized as follows:

- **Shift left:** It encourages moving security from the right, i.e., the end of the DevSecOps process, to the left and involving security to all processes from software development to delivery, from the beginning. Specifically, cybersecurity experts involved early in the design, development, validation processes can contribute to implementing security as software pieces are built. Then, security risks and vulnerabilities can be identified in the early software lifecycle phases and addressed properly.
- **Security education:** Training software engineers on security is essential to achieving the intersection point on engineering and compliance to organization's security measures. Developers should be familiar with terms, such as threat models, compliance checks, vulnerability tests and implementing security controls.
- **Culture:** A culture of security within organizations or even teams is critical to implementing DevSecOps, as it will help individuals understand and undertake their responsibilities in the DevSecOps lifecycle. Indeed, the culture of security will involve people, communication, processes and technology, so that software security requirements are integrated into organizations' processes, with the selected technological tools.
- **Traceability, auditability, and visibility:** These principles should guide organizations adopting DevSecOps, in order to ensure deeper insight into a more secure environment, where tracking configuration items, well-documented and with appropriate dissemination level within the organization will facilitate the operation of DevSecOps within the organization.

### 3.3.1 CI/CD

Continuous Integration (CI) is a developer practice to keep a working system by small changes growing the system by integrating frequently (usually at least daily) on the mainline by means of appropriate tools supporting automation with lots of automated tests. This enables teams to work on shared code and increases the visibility into the development and



## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

quality of the system. By referring to a developer practice, Continuous Integration (CI) typically expects developers to implement Test-Driven Development (TDD) with constant refactoring practice. When a developer is unit-test-driving his code, he ensures that his local copy is always working.

Continuous Deployment (CD) refers to the automated deployment of new -release- versions of a system to the production environment. Following the continuous integration process, as described above, when a system reaches a maturity level (as indicated by specific, predefined criteria), CD takes care of updating an existing running version of the system automatically, minimizing downtime.

Combined, CI/CD is a pipeline that gets new developments and provides an updated running version of a system hosted in a predefined environment. In IoT-NGIN a CI/CD environment is already established in GitLab. Figure 10 presents in high level GitLab's CI/CD pipeline.

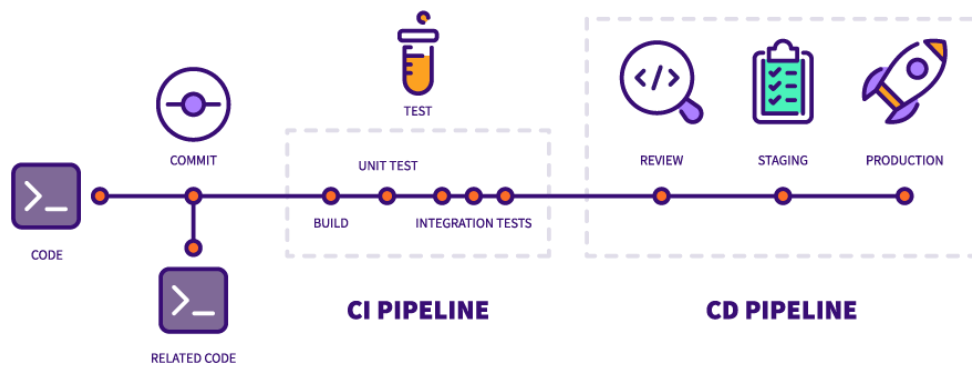


Figure 10 – CI/CD pipeline – Source: [about.gitlab.com](https://about.gitlab.com).

The proposed CI/CD pipeline is illustrated in more detail in the following figure.

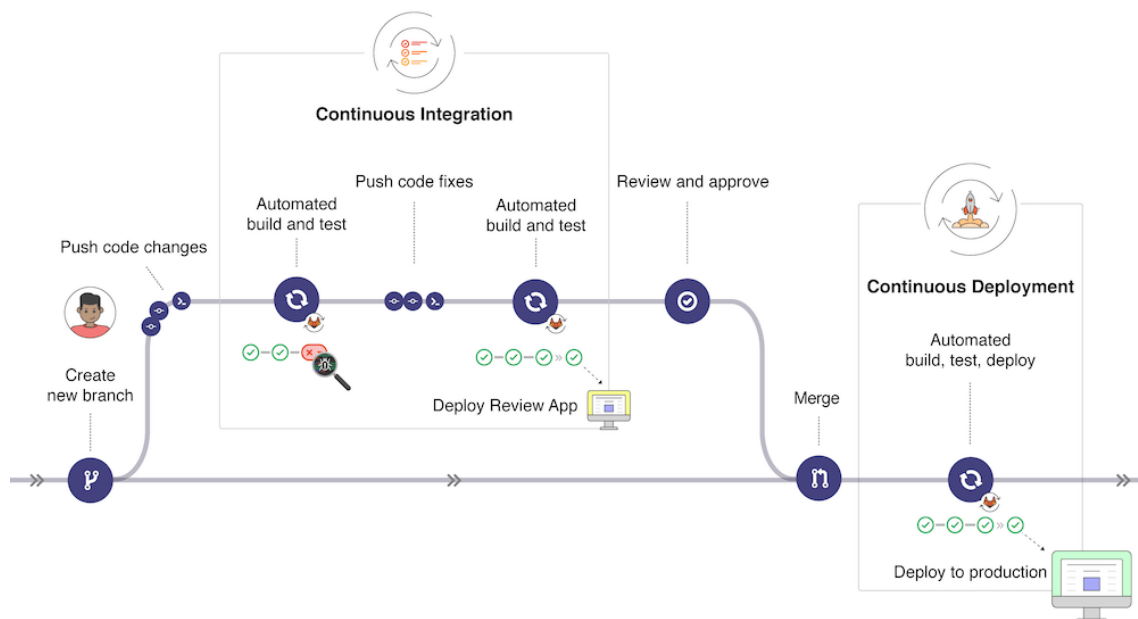


Figure 11 - CI/CD pipeline steps - Source: [docs.gitlab.com](https://docs.gitlab.com).

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

In Figure 11, the major steps of the GitLab's CI/CD process are presented in more detail, which can be described as follows.

- The developer implements a certain piece of software that wishes to integrate to a module functionality.
- The developer introduces unit tests for the relevant piece of code evaluating the modules outcomes consistency.
- The developer commits and pushes code developments on their local repository to a branch of remote GitLab repository that is a dedicated development branch defined in the CI infrastructure.
- The developer requests code merging.
- The CI/CD pipeline set for the specific project is triggered. The CI platform performs the determined unit test(s);
- If unit testing is successful, the code is subject to further integrated system testing.
- If the integrated system testing is successful
  - the merge request is accepted, and the newly committed source code gets part of the code master branch.
- Otherwise, the merge request is rejected
  - In this case, the developer should update the code to satisfy the unit testing procedures, or should update the unit testing per se.
- The source code management platform moves to the CD part, building the package and deploying it;
- After successful CD, the new code is running on the deployment infrastructure and is subject to user testing/production.

Furthermore, CI/CD pipeline can be further enhanced via security processes in order to comply with DevSecOps practices incorporating automated security testing such as Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) which can be performed at critical points of the development cycle as illustrated in Figure 12.

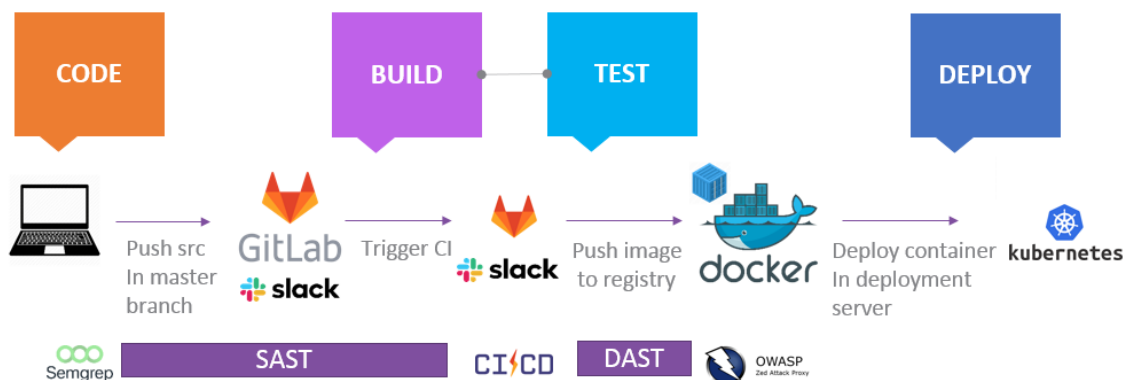


Figure 12 - SAST/DAST in the development lifecycle.

### 3.3.1.1 CI/CD security enhancements

**Static Application Security Testing (SAST)**, also known as "white box testing", allows developers to find security vulnerabilities in the application source code earlier in the software development life cycle. SAST is utilized to check the code without executing it. It also ensures conformance to coding guidelines and standards without actually executing the underlying code. Incorporating a static analyser into the CI/CD loop helps forestall programming bugs from the early stages of the development before getting to a higher level.

**Dynamic Application Security Testing (DAST)**, also known as "black box testing", investigates for security vulnerabilities and weaknesses in a running application. It is performed later in the development lifecycle, as it requires a built and tested application. The tester has no knowledge of the source code of the application or the technologies or frameworks the application is built on. DAST, in a nutshell, tests the security of developed software by feeding it with malicious data trying to detect security vulnerabilities and to determine security vulnerabilities that are linked to the operational deployment of an application.

## 3.3.2 Source Code Management

Source code management (SCM) is the practice of tracking modifications to source code. Keeping a running history of the changes made to a codebase helps programmers, developers and testers ensure that they are always working with accurate and up-to-date code and helps resolve conflicts when merging code from multiple sources. It is also known as Version Control System (VCM). The most popular VCM is GIT which is an open-source distributed software for tracking changes in any set of files.

GitLab [7] and GitHub [8], are open-source GIT management software that offer GIT repository management and additionally, code reviews, issue tracking, wikis etc. Both software management solutions are offered as a single stand-alone web platform that span the entire software development lifecycle.

In the context of IoT-NGIN, GitLab is set to be the CI/CD platform that will be utilized by the consortium. GitLab DevOps software combines the ability to develop, secure and operate software in a single application.

## 3.3.3 CI/CD in IoT-NGIN

For the IoT-NGIN project, the GitLab CI/CD framework has been set up and organized in the public instance of Gitlab, the official GitLab group of IoT-NGIN is entitled "H2020 IoT-NGIN" and is accessible publicly at <https://gitlab.com/h2020-iot-ngin>. The group hosts the source code that is related to each thematic entity-specific development as dictated by the IoT-NGIN architecture. Each thematic entity is organized as a subgroup of the IoT-NGIN GitLab group.

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

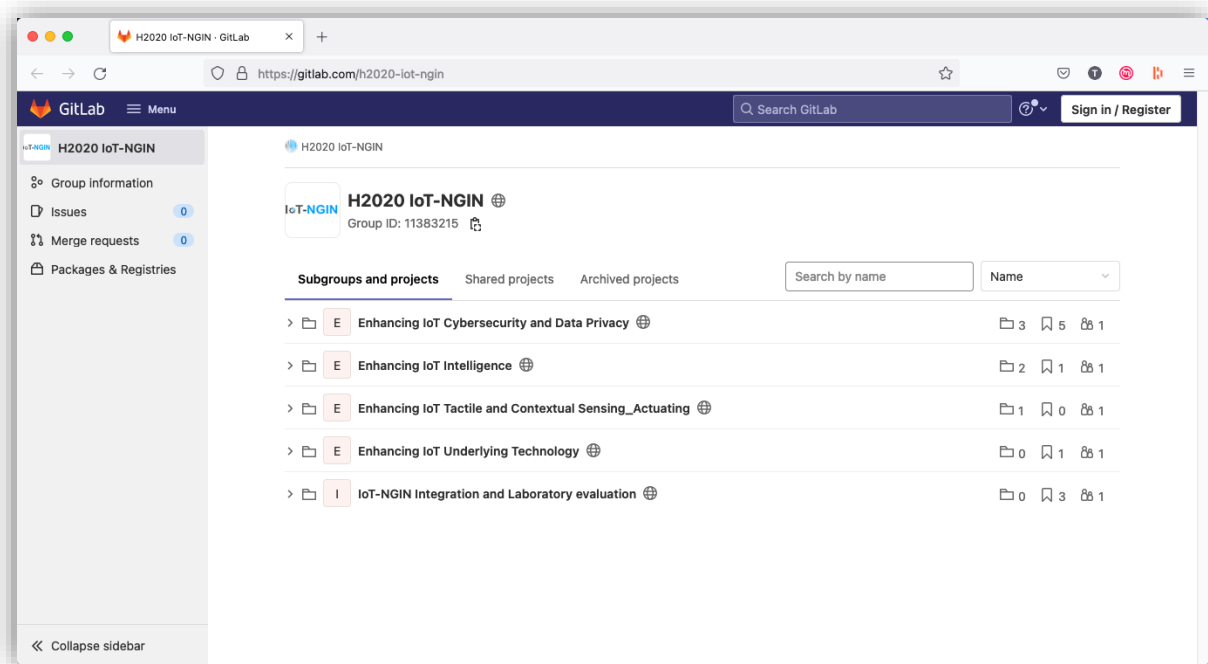


Figure 13 - IoT-NGIN GitLab Home page.

Within each subgroup, the development activities are organized based on the implemented outcomes the relevant tasks. Indicatively, Figure 14, depicts the content of the subgroup “Enhancing IoT Cybersecurity and Data Privacy”.

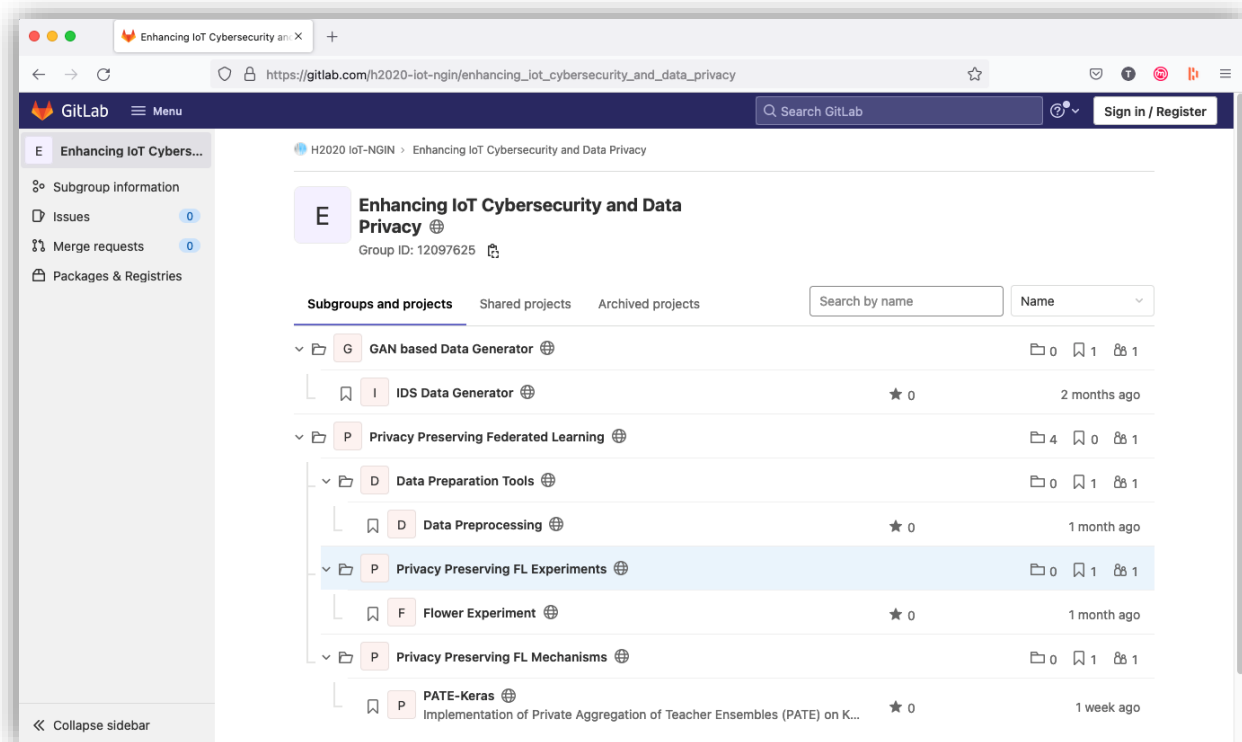
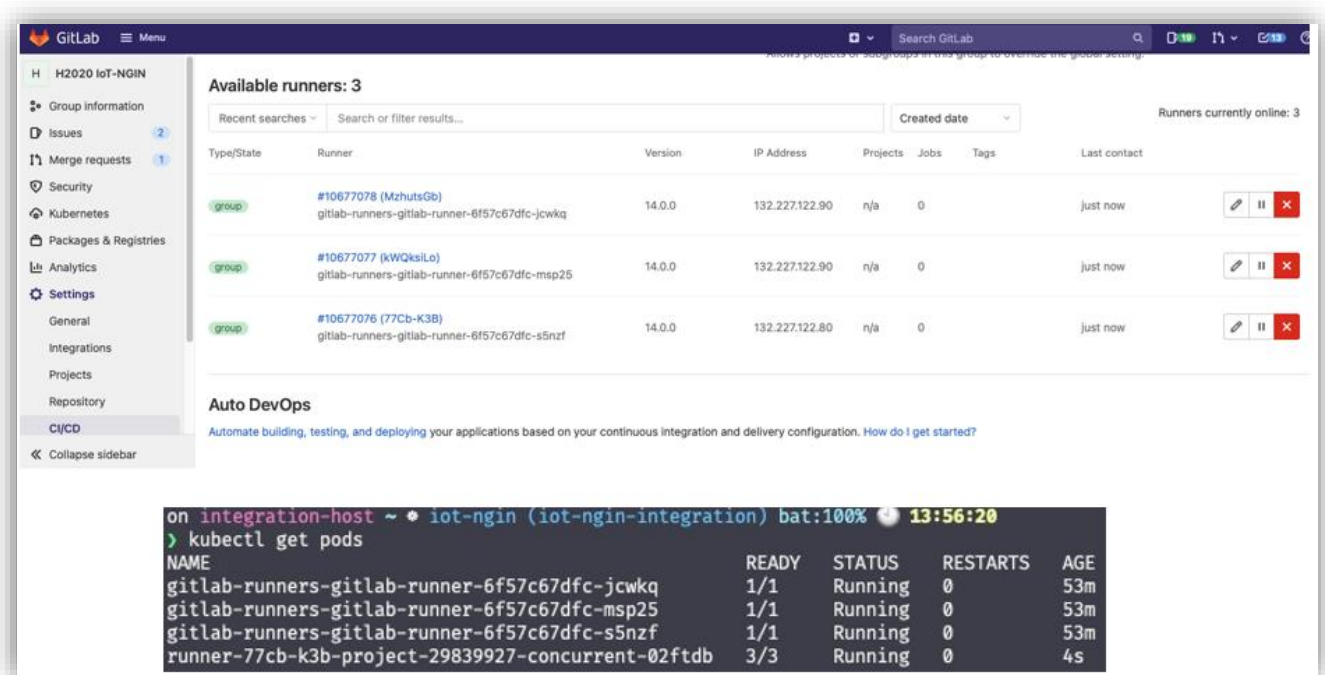


Figure 14 - IoT-NGIN Enhancing IoT Cybersecurity and Data Privacy Subgroup.

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

Within the GitLab CI/CD platform, IoT-NGIN has configured the CI/CD pipeline that will enable the automation for the integration and deployment of the project technical outcomes. The CI/CD pipeline is driven by three GitLab Runners that have been deployed and integrated within the Kubernetes cluster of the OneLab facilities of the Sorbonne University. Figure 15 depicts the GitLab runners of IoT-NGIN and their configuration. More information on the specification of the cluster is presented in section 3.4. In addition, IoT-NGIN, adhering to the cloud native approach, provides automated delivery procedures. More specifically, IoT-NGIN docker images are built based on the uploaded source code and in parallel the images are uploaded to the IoT-NGIN public docker repository, that has been established for the needs of the project. The IoT-NGIN profile in the docker image repository is available at <https://hub.docker.com/u/iotngin>. Figure 16 provides an indicative snapshot of this profile.



**Available runners: 3**

Recent searches ~ Search or filter results... Created date Runners currently online: 3

Type/State	Runner	Version	IP Address	Projects	Jobs	Tags	Last contact
group	#10677078 (MzhutsGb) gitlab-runners-gitlab-runner-6f57c67dfc-jcwqk	14.0.0	132.227.122.90	n/a	0		just now
group	#10677077 (kVQksLo) gitlab-runners-gitlab-runner-6f57c67dfc-msp25	14.0.0	132.227.122.90	n/a	0		just now
group	#10677076 (77Cb-K3B) gitlab-runners-gitlab-runner-6f57c67dfc-s5nzf	14.0.0	132.227.122.80	n/a	0		just now

**Auto DevOps**  
Automate building, testing, and deploying your applications based on your continuous integration and delivery configuration. [How do I get started?](#)

```
on integration-host ~ • iot-ngin (iot-ngin-integration) bat:100% 13:56:20
> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
gitlab-runners-gitlab-runner-6f57c67dfc-jcwqk   1/1     Running   0           53m
gitlab-runners-gitlab-runner-6f57c67dfc-msp25   1/1     Running   0           53m
gitlab-runners-gitlab-runner-6f57c67dfc-s5nzf   1/1     Running   0           53m
runner-77cb-k3b-project-29839927-concurrent-02ftdb 3/3     Running   0           4s
```

Figure 15 - IoT-NGIN Gitlab runners' configuration.

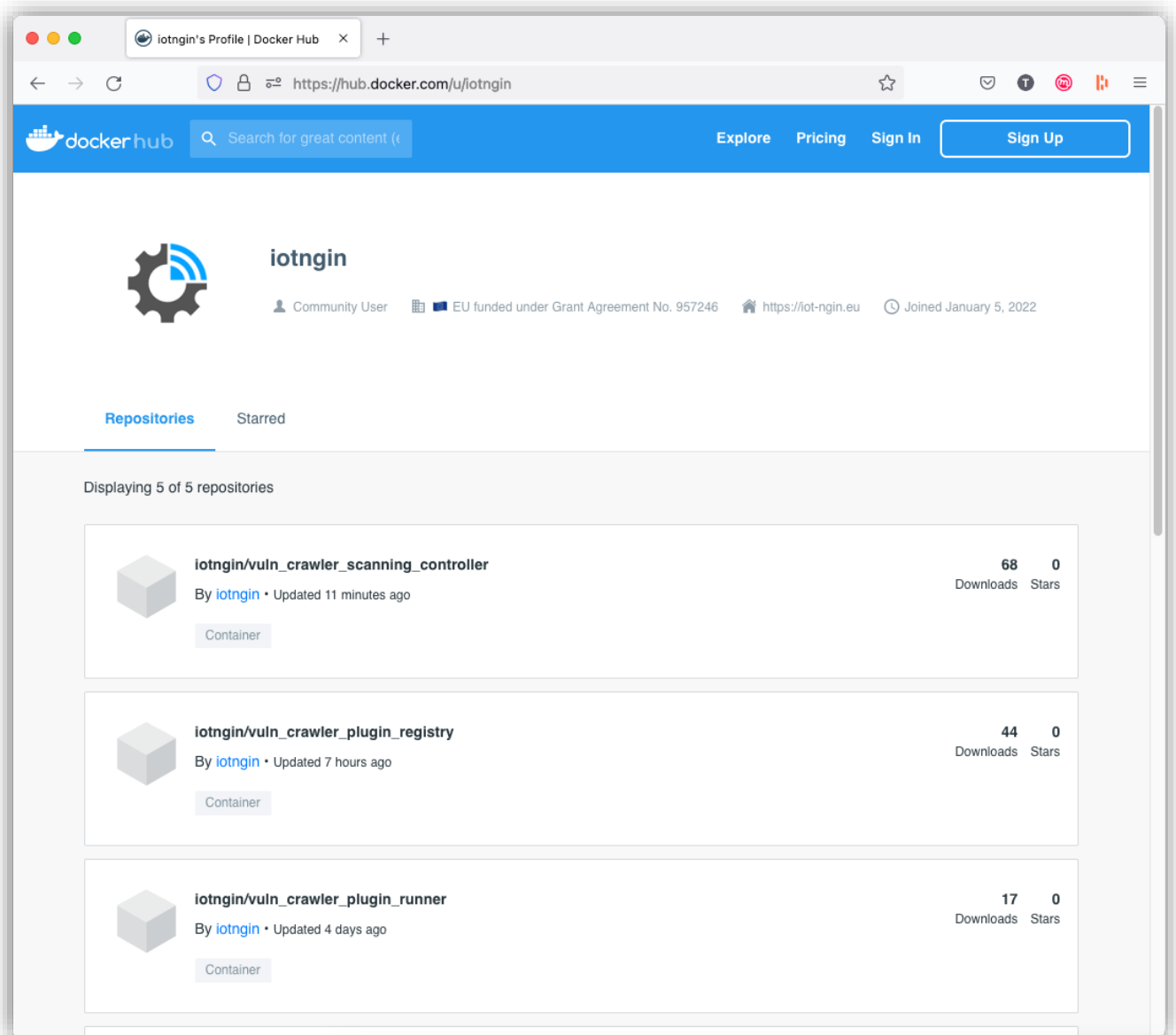


Figure 16 - Snapshot of the project docker image repository.

The issue tracking capabilities of the GitLab web platform are also utilized for the purposes of the IoT-NGIN project. Figure 17 illustrates the relevant issue tracking webpage that represents open and closed issues that have been introduced by the project's technical teams.

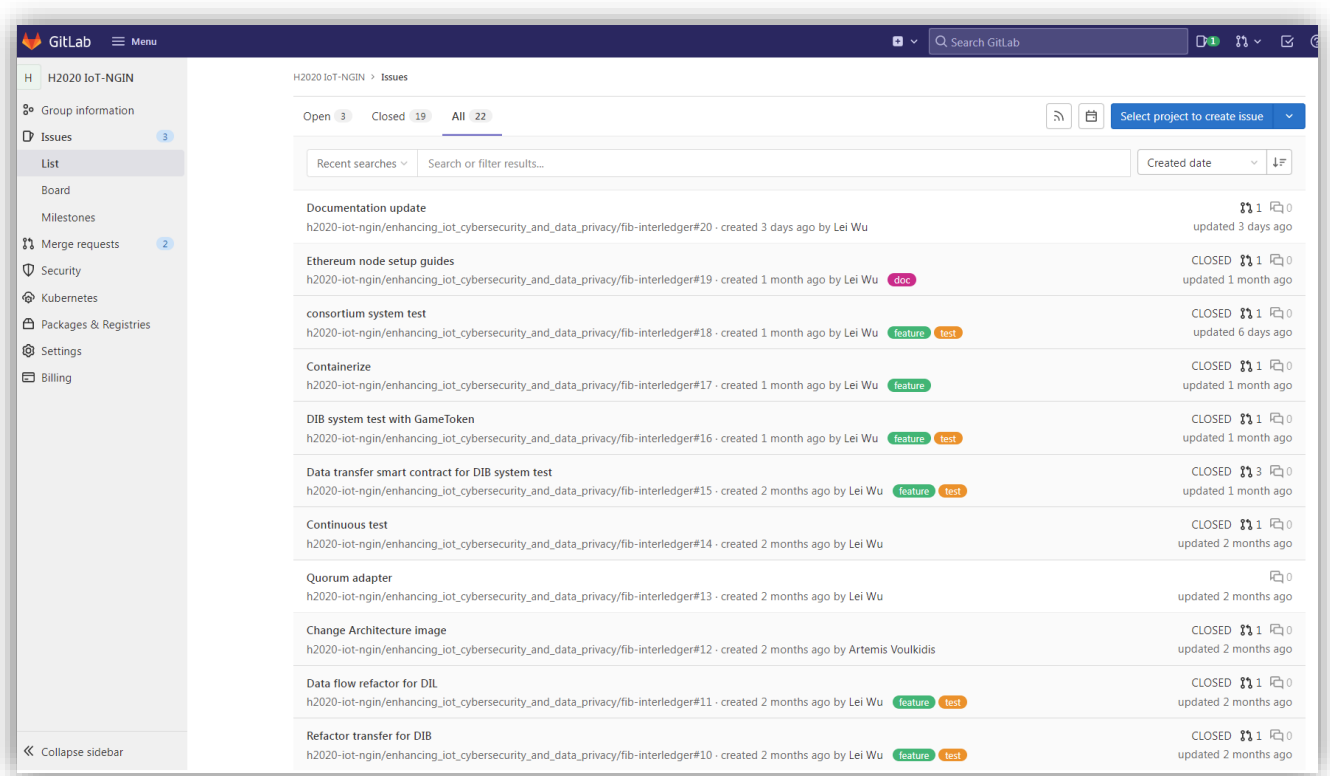


Figure 17 - IoT-NGIN GitLab Issue Tracking.

### 3.3.4 Containers

Containerization refers to Operating System (OS)-level virtualization for deploying and running distributed applications. Containerization is a lightweight alternative to a virtual machine that involves encapsulating an application in a container with its own operating system. A container takes its meaning from the logistics term, *packaging container*.

The most popular containerization technology application is Docker [9]. Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings [10]. Moreover, Kubernetes (K8s) [11] is an open-source system for automating deployment, scaling, and management of containerized applications and is the most common automated container deployment, scaling, and management platform, appropriate for working with Docker containers,

IoT-NGIN adhering to the cloud native approach and to the modular architecture principles as stated above, adopts Docker as a containerization framework and thus, strongly encourages the technical developers of the project to wrap their implemented technical outcomes in docker containers. Ultimately, the containerized technical outcomes of the projects will be deployed within the IoT-NGIN integration and testing infrastructure hosted by OneLab, which utilizes Kubernetes as a container orchestration platform. More details on the specifications of the IoT-NGIN integration and testing cluster are presented in the following sections.



### 3.4 IoT-NGIN Integration, Testing and Validation Infrastructure

### 3.4.1 Integration and Testing environment

### 3.4.1.1 The OneLab Facility

The OneLab facility is a state-of-the-art test platform for exploring the design of digital infrastructures. It provides control and remote access over a large and diverse set of virtualized and programmable resources from IoT to the Cloud. OneLab federates multiple facilities among which the NITOS and the FIT (Figure 18) test platforms and offers the possibility to run large-scale experiments combining multiple heterogeneous resources through one single portal.

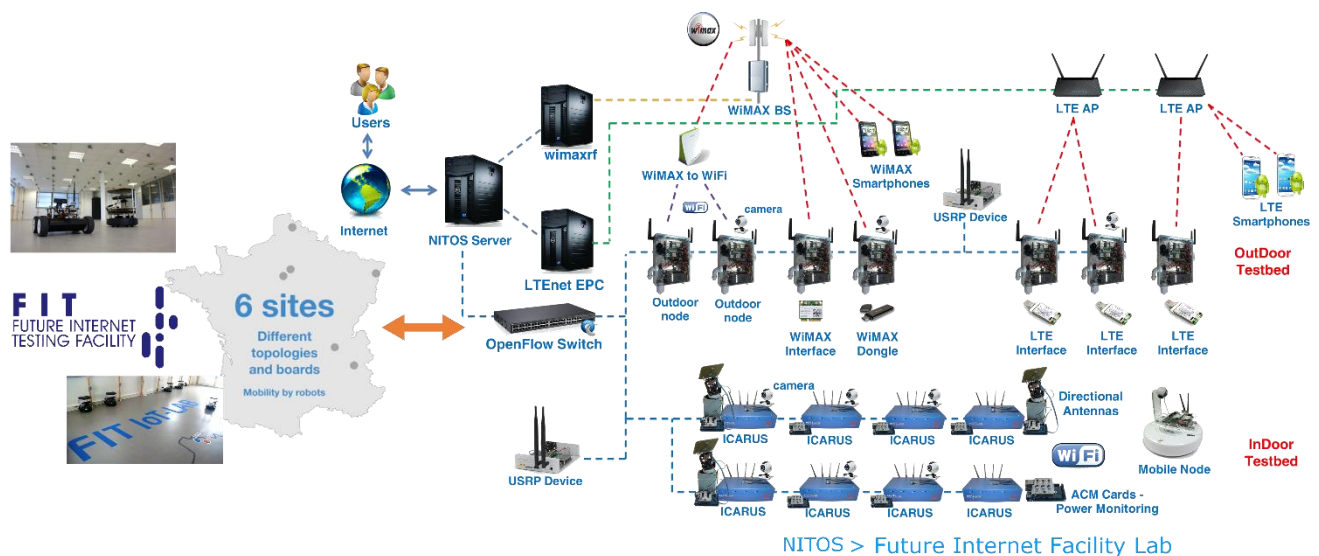


Figure 18 - SU OneLab (FIT NITOS & FIT IoT-Lab) as IoT-NGIN Integration, Test and Validation Infrastructure.

OneLab provides a large set of IoT resources including mobile and wireless IoT devices equipped with various sensors such as ambient light, temperature, atmospheric pressure and temperature sensor, tri-axis accelerometer, tri-axis magnetometer, tri-axis gyrometer and also cloud and bare-metal resources allowing experimentation with cloud-based technologies with over a hundred of computing cores available to experimenters.

OneLab offers, through the federated NITOS Lab facility, an SDR (Software Defined Radio) testbed consisting of wireless nodes attached with Universal Software Radio Peripheral (USRP) devices and a SDN testbed equipped with multiple OpenFlow enabled switches to run experiments with network switching and routing protocols.



### 3.4.1.2 OneLab cluster for IoT-NGIN

An Integration Infrastructure is crucial to ensure proper integration of the IoT-NGIN components developed by the consortium partners as a whole product. In scope of the project, The Sorbonne University provides full access to the OneLab facility as an Integration Infrastructure for integration, tests, and validation of the different IoT-NGIN components.

The integration infrastructure consists, at present, of two different Kubernetes Clusters which serve different purposes. A primary cluster as the production cluster and a secondary cluster for KubeFlow and for all components that require a Graphics Processing Unit (GPU) workload.

Kubernetes is a container orchestration tool to deploy containerized applications. The containers are more lightweight and flexible than virtual machines and easy to scale up. This will facilitate the deployment and integration of the IoT-NGIN components. Moreover, a Kubernetes cluster is composed of a set of nodes: a master node to handle the scheduling and scaling of applications, and also the management of the cluster, and a worker node to perform tasks assigned by the master node, such as deploying the container and hosting the applications. In the production setting, multiple worker nodes are used to offer redundancy and to ensure the service availability.

The cluster setup is specified in Table 1 and Table 2.

Table 1 - Production Cluster Configuration.

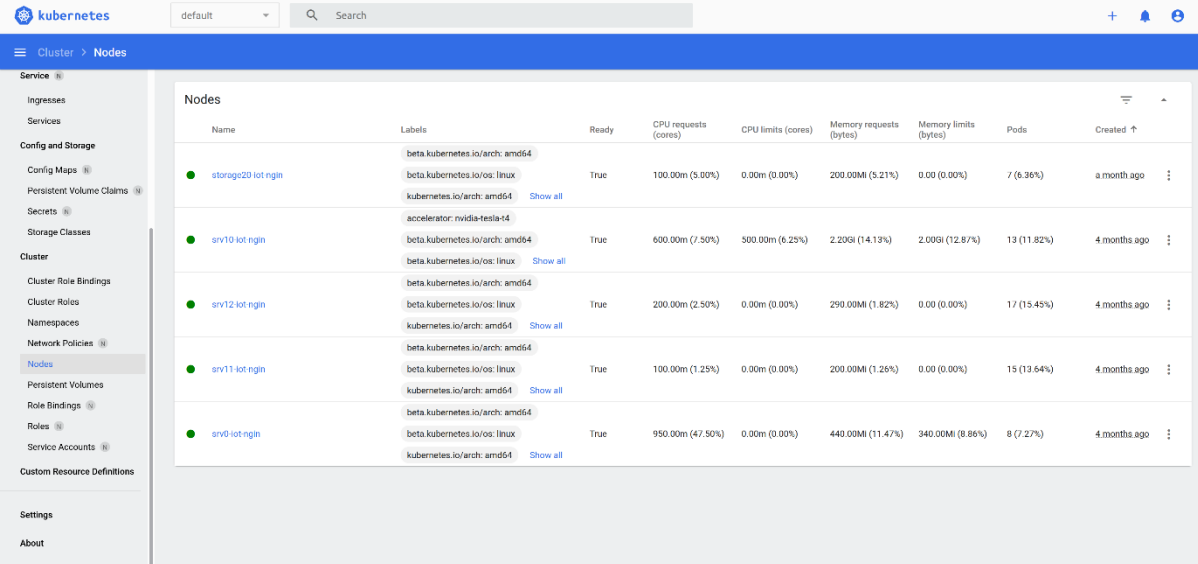
Node ID	Node Type	Specification	Ceph OSD Capacity
srv0-iot-ngin	Master	2 CPU Cores 4Go RAM	-
srv10-iot-ngin	Worker and Storage	8 CPU Cores 16Go RAM	100Go
srv11-iot-ngin	Worker and Storage	8 CPU Cores 16Go RAM	100Go
srv12-iot-ngin	Worker and Storage	8 CPU Cores 16Go RAM	100Go
storage20-iot-ngin	Storage Node	2 CPU Cores 4Go RAM	100Go

Table 2 - Secondary Cluster Configuration.

Node ID	Node Type	Specification	Ceph OSD Capacity
srv100-iot-ngin	Master	2 CPU Cores 4Go RAM	-
srv110-iot-ngin	Worker and Storage	8 CPU Cores 16Go RAM	100Go
srv111-iot-ngin	Worker and Storage	8 CPU Cores 16Go RAM & GPU NVIDIA Tesla T4	100Go
storage120-iot-ngin	Storage Node	2 CPU Cores 4Go RAM	100Go
storage121-iot-ngin	Storage Node	2 CPU Cores 4Go RAM	100Go

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

Kubernetes cluster configures by default, an API server that serves as the front end of the control plane. The REST API is consumed by the CLI client (kubectl) to manage the cluster and to deploy applications. A dashboard is deployed to facilitate the overview and management of the cluster resources.



Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Pods	Created
storage20-iot-nginx	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/arch: amd64	True	100.00m (5.00%)	0.00m (0.00%)	200.00Mi (5.21%)	0.00 (0.00%)	7 (6.36%)	a month ago
srv10-iot-nginx	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux accelerator: nvidia-tesla-t4	True	600.00m (7.50%)	500.00m (6.25%)	2.20Gi (14.13%)	2.00Gi (12.87%)	13 (11.82%)	4 months ago
srv12-iot-nginx	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/arch: amd64	True	200.00m (2.50%)	0.00m (0.00%)	290.00Mi (1.82%)	0.00 (0.00%)	17 (15.45%)	4 months ago
srv11-iot-nginx	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/arch: amd64	True	100.00m (1.25%)	0.00m (0.00%)	200.00Mi (1.26%)	0.00 (0.00%)	15 (13.64%)	4 months ago
srv0-iot-nginx	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/arch: amd64	True	950.00m (47.50%)	0.00m (0.00%)	440.00Mi (11.47%)	340.00Mi (8.86%)	8 (7.27%)	4 months ago

Figure 19 - Kubernetes Dashboard.

The current exposed endpoints are described in Table 3 and Table 4.

Table 3 - Production Cluster Endpoints.

Endpoints	Ports	Services
api.iot-ngin.onelab.eu	6443	API server
apps.iot-ngin.onelab.eu	30080 (HTTP), 30443 (HTTPS)	Ingress Controller
dashboard.iot-ngin.onelab.eu	30443	Dashboard

Table 4 - Secondary Cluster Endpoints.

Endpoints	Ports	Services
api.kube2.iot-ngin.onelab.eu	6443	API server
apps.kube2.iot-ngin.onelab.eu	30080 (HTTP), 30443 (HTTPS)	Ingress Controller
dashboard.kube2.iot-ngin.onelab.eu	30443	Dashboard

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

The clusters are configured with an Ingress Controller in order to provide external access to the services and IoT-NGIN components. The Ingress Controller uses a set of rules defined by Ingress Resources to manage the traffic routing. The Ingress Controller provides also load balancing and SSL termination capabilities.

The clusters integrate the NGINX Ingress Controller which uses NGINX as the reverse proxy and load balancer. The Ingress Controller is also configured using the name based virtual hosting which allows traffic routing to multiple host names at the same IP address. Figure 20 illustrates the traffic routing from the client to the pod in this configuration.

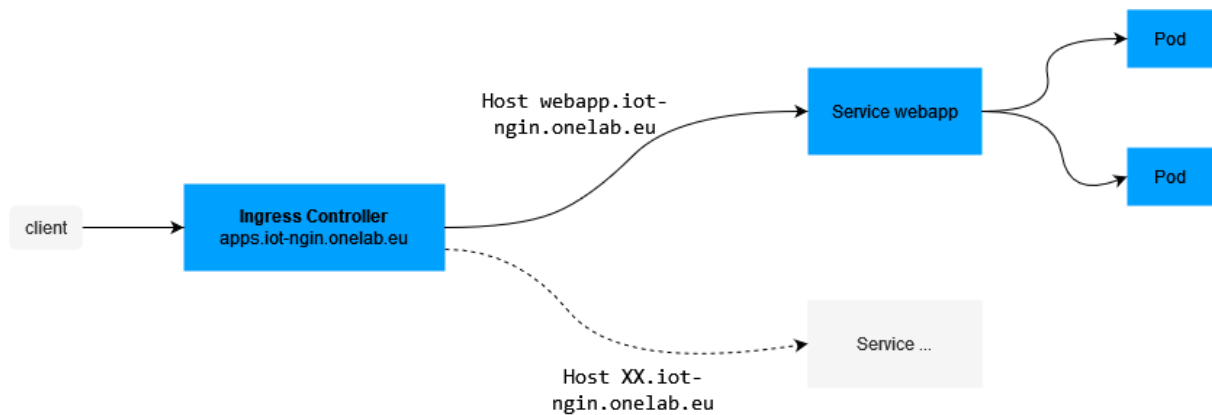


Figure 20 - Name Based Virtual Hosting.

Figure 21 is an Ingress Resource which correlates with the example illustrated in Figure 20. The Ingress Resource incorporates the traffic routing rules, the configuration for SSL termination and the issuer for the certificate to be issued.

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: default-ingress-resource
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    kubernetes.io/ingress.class: nginx
    cert-manager.io/cluster-issuer: letsencrypt-production
  namespace: default
spec:
  tls:
    - hosts:
        - apps.iot-ngin.onelab.eu
      secretName: iot-ngin-webapp-tls
  rules:
    - host: webapp.iot-ngin.onelab.eu
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: webapp
                port:
                  number: 8080
  
```

Figure 21 - Ingress Resource.

The SSL certificates for the host names defined in the Ingress Resources are managed by Cert-manager, an automation tool which simplifies the process of obtaining and renewing

the certificates. It uses well-known and trusted sources to issue the certificate such as Let's Encrypt.

Furthermore, the clusters use Ceph to deploy a reliable and scalable distributed storage infrastructure. The data is stored and replicated in the pool of OSD (Object Storage Device) in order to guarantee data availability. Each of the worker nodes has a partition configured as an OSD to ensure the data is replicated across different hosts, thus, to prevent data loss in case of a host failure.

ID	CLASS	WEIGHT	REWEIGHT	SIZE	RAW USE	DATA	OMAP	META	AVAIL	%USE	VAR	PGS	STATUS
2	hdd	0.09769	1.00000	100 GiB	4.7 GiB	4.1 GiB	15 KiB	706 MiB	95 GiB	4.75	0.99	24	up
0	hdd	0.09769	1.00000	100 GiB	4.4 GiB	4.1 GiB	15 KiB	312 MiB	96 GiB	4.38	0.91	23	up
1	hdd	0.09769	1.00000	100 GiB	5.4 GiB	4.2 GiB	18 KiB	1.1 GiB	95 GiB	5.37	1.12	25	up
3	hdd	0.09769	1.00000	100 GiB	4.7 GiB	4.5 GiB	15 KiB	222 MiB	95 GiB	4.74	0.99	27	up
TOTAL				400 GiB	19 GiB	17 GiB	65 KiB	2.3 GiB	381 GiB	4.81			
--- RAW STORAGE ---													
CLASS		SIZE		AVAIL		USED		%RAW USED					
hdd		400 GiB		381 GiB		19 GiB		19 GiB		4.81			
TOTAL		400 GiB		381 GiB		19 GiB		19 GiB		4.81			

Figure 22 - Ceph OSD.

## 3.4.2 Laboratory testing environment

### 3.4.2.1 5G test infrastructure in Eurolab

The Ericsson laboratory infrastructure is located in Ericsson Eurolab in Aachen, Germany. It will be configured to test the performance of 5G mobile networks by transmitting synthetic data streams of the IoT-NGIN living lab trials.

Figure 23 depicts a schematic diagram of the Ericsson mobile network laboratory test infrastructure which is typical of the type of mobile network infrastructure to provide 5G services. The infrastructure includes a 5G live, standalone, non-public mobile network. Moreover, a use case data generator and the 5G modem is used to perform the tests. The data streams of the IoT-NGIN use cases are generated by the data simulator according to the communication characteristics described in Section 4.2.1.1. Then, these data streams are transmitted via the 5G radio modem to the edge cloud hosted by the 5G mobile network.

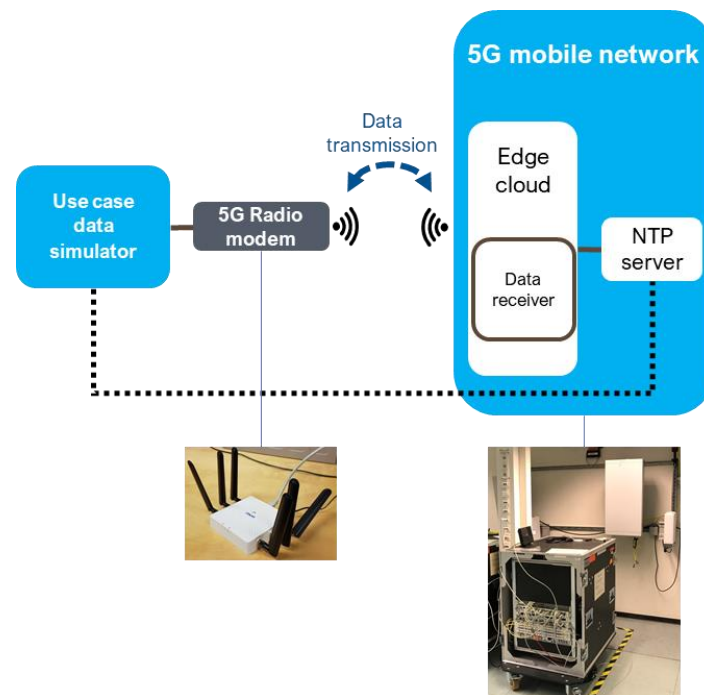


Figure 23 - EDD lab infrastructure planned for IoT-NGIN project.

When transmitting the synthetic data streams, the performance of the 5G network will be evaluated in terms of throughput, packet loss, and one-way latency. Latency is the difference between the time when sending a message from the source and the time of arrival at the receiver. One-way latencies can be measured in uplink or downlink. Uplink is the direction from the 5G radio modem towards the edge cloud, and downlink vice versa.

For precise one-way latency measurements, the system clocks of the data simulator and the edge cloud must be synchronized. Time synchronization is performed using Network Time Protocol (NTP). It is executed on both data simulator and edge cloud such that the resulting time difference between the two entities is minimized.

The results of the tests are expected to show that the 5G performance is significantly higher than previous technologies and that 5G can meet the requirements of the IoT-NGIN use cases.

### 3.4.2.2 Realtime Laboratory

The Real-time Laboratory is located at RWTH Aachen University. With the available equipment, a unique view of the interaction between the power grid, communication infrastructure and control systems in energy systems is possible. Specifically, the Real-Time Simulators (RTS), as shown in Figure 24, provide the ability to connect real control infrastructure and power hardware like converters to a simulated power grid. This enables us to understand the impact of different grid conditions on the device under test.



Figure 24 - RWTH Real time laboratory, Real Time Simulators.

In addition, the Real-time Laboratory also hosts commercial and in house developed Phasor Measurement Units (PMUs). They are used to compare the results of the phasor of voltage or current in power grids, in terms of accuracy and phasor rate. For this comparison, the RTS system also plays an important role, as it is used to generate Global Positioning System (GPS) synchronized test signals that simulate the steady state as well as the dynamic behavior of the power grid. Based on these signals, a qualification of the PMUs can be done. Figure 25 shows the PMU rack in the laboratory. To explore the interface of power grid dynamics with cutting edge IT technologies, the RWTH Real-time laboratory hosts a High-performance computing (HPC) cluster and multiple servers that run test instances of edge-clouds, unikernel applications and agent based power grid optimizations. This enables a holistic test environment for the development of future energy systems.





Figure 25 - RWTH Real time laboratory, Person working on the PMU rack.

The above-mentioned capabilities will be used to build a hardware in the loop setup consisting of a grid simulation to generate the measurements for the edge PMU and the phasor estimation running in the emulated edge cloud. This will provide the means for investigating the behavior of the whole chain of components. Based on the setup, different parameters, for example the network quality, impact of migration, dynamic grid changes as well as the means of communication with new technologies like 5G, will be evaluated.

## 4 Initial integration and validation

### 4.1 Early integration activities

The guidelines presented in the previous sections have been actively applied and have been integrated into the development processes of the consortium members. Indicatively, in Figure 26, an example of IoT-NGIN software development life cycle (SDLC) control implementing the DevSecOps processes of the project is presented, having four relevant stages properly defined and executed as follows:

- **Build:** employed in order to build a docker image of the intended module.
- **SAST:** employed in order to check the source code for known security issues e.g. related to known bad coding practices that can lead to security breaches, improper hashing algorithms etc.
- **Test:** employed to test the source code functionality (in Figure 26, only generic tests have been defined, not unit/component tests).
- **Deploy:** employed in order to automatically deploy updated codebases on the integration testbed presented in paragraph 3.4.

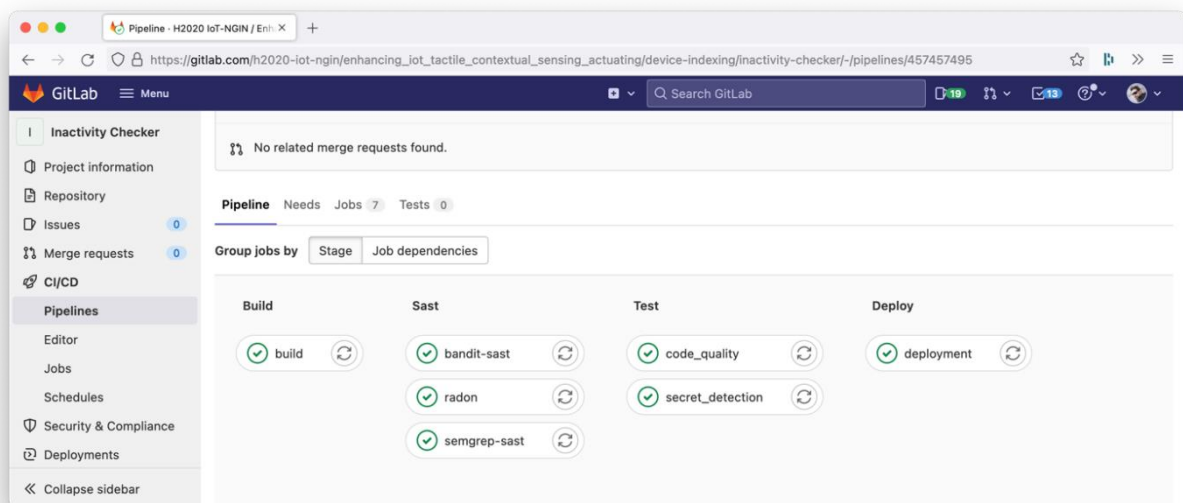


Figure 26 - DevSecOps processes with the build, SAST, test and deployment stages enabled.

In the same context, in Figure 27, an example of a successful SAST-stage job is depicted.



## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

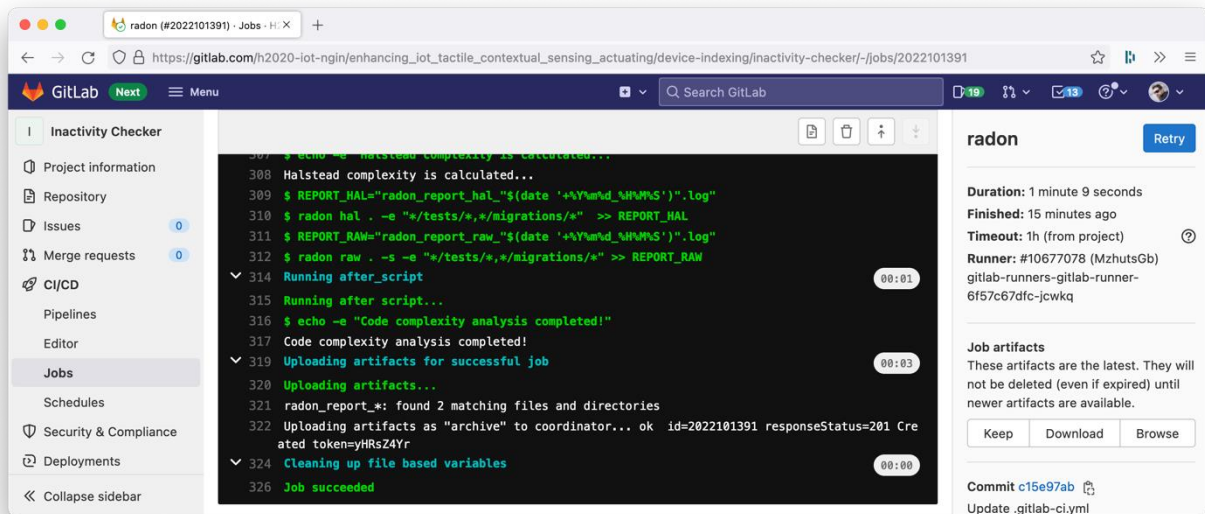


Figure 27 - Example of a SAST job output – related to code quality analysis.

Regarding the deployment of the various project components, as these have been defined and presented in section 2.1, many of them have already been deployed on the project Kubernetes cluster (see §3.4). Indicatively, Figure 28 depicts the Kubernetes resources (pods, services, deployments and replica sets) dedicated to a blockchain (Quorum) network provided to the rest of the project components (e.g. towards mediating the zero knowledge model verification processes performed in the context of the MLaaS framework).

```
$ kubectl get all -n iot-ngin-wp6
```

NAME	READY	STATUS	RESTARTS	AGE
pod/cakeshop-d5b6b89c9-7lz8n	1/1	Running	0	8d
pod/quorum-node1-deployment-7f47894f57-lsf4b	2/2	Running	0	12d
pod/quorum-node2-deployment-7b9798ddc-7dtbk	2/2	Running	0	12d
pod/quorum-node3-deployment-6bf7fcd8fd-74gj5	2/2	Running	0	12d

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/cakeshop	NodePort	10.108.205.141	<none>	8080:30504/TCP	8d
service/quorum-node1	NodePort	10.96.41.65	<none>	9001:32253/TCP, 9080:30115/TCP, 8545:32496/TCP, 8546:31527/TCP, 30303:30578/TCP	12d
service/quorum-node2	NodePort	10.104.98.205	<none>	9001:31634/TCP, 9080:31718/TCP, 8545:30096/TCP, 8546:30460/TCP, 30303:31507/TCP	12d
service/quorum-node3	NodePort	10.98.244.164	<none>	9001:30985/TCP, 9080:31977/TCP, 8545:32513/TCP, 8546:32587/TCP, 30303:31294/TCP	12d

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/cakeshop	1/1	1	1	8d
deployment.apps/quorum-node1-deployment	1/1	1	1	12d
deployment.apps/quorum-node2-deployment	1/1	1	1	12d
deployment.apps/quorum-node3-deployment	1/1	1	1	12d

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/cakeshop-d5b6b89c9	1	1	1	8d
replicaset.apps/quorum-node1-deployment-7f47894f57	1	1	1	12d
replicaset.apps/quorum-node2-deployment-7b9798ddc	1	1	1	12d
replicaset.apps/quorum-node3-deployment-6bf7fcd8fd	1	1	1	12d

Figure 28 - Kubernetes resources consumed by WP6 activities.

Similarly, in Figure 29, an overview of the already applied Ingress controllers (implemented as dynamic NGINX configurations) is provided, featuring four fully qualified domain names (FQDN) pointing to the services depicted in Figure 28.

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

```
$ kubectl get ingress -n iot-ngin-wp6
NAME          CLASS          HOSTS          ADDRESS          PORTS          AGE
dlt-explorer  <none>         explorer.dlt.apps.iot-ngin.onelab.eu  10.109.2.245    80, 443       8d
tls-quorum-node1 <none>         quorum-node-1.dlt.apps.iot-ngin.onelab.eu  10.109.2.245    80, 443       8d
tls-quorum-node2 <none>         quorum-node-2.dlt.apps.iot-ngin.onelab.eu  10.109.2.245    80, 443       8d
tls-quorum-node3 <none>         quorum-node-3.dlt.apps.iot-ngin.onelab.eu  10.109.2.245    80, 443       8d
$
```

Figure 29 - Ingress controllers configured to be used by WP6 components.

For each one of the Ingress controllers, a dedicated TLS v1.3 certificate has been generated, encrypting the data that get exchanged through that particular Ingress (Figure 30).

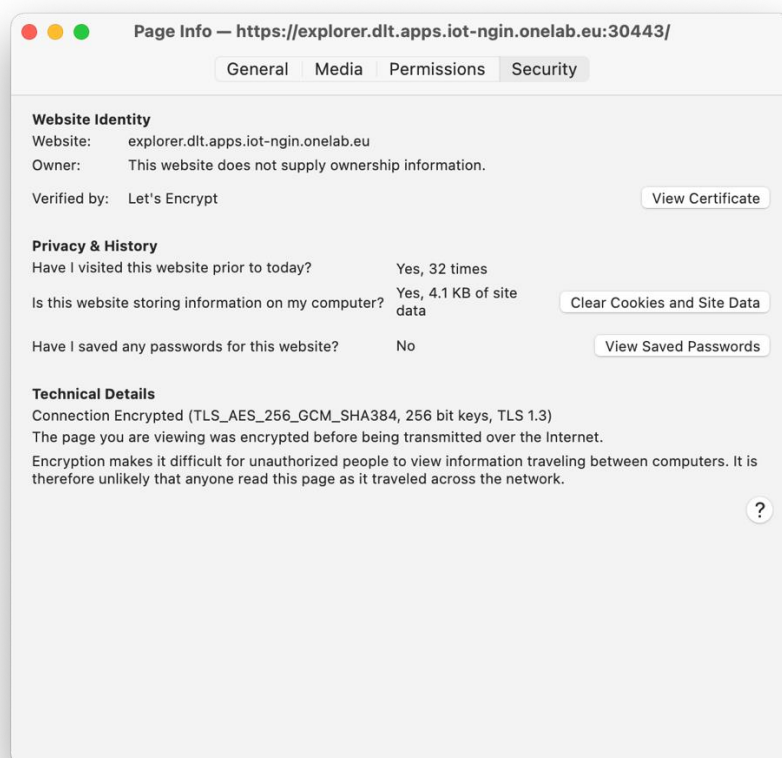


Figure 30 - Details of certificate automatically issued by the cert-manager of the cluster.

Figure 31, below, showcases the web UI of the deployed Quorum network, publicly reachable through the FQDN of the respective Ingress controller and secured via the certificate of Figure 30.

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

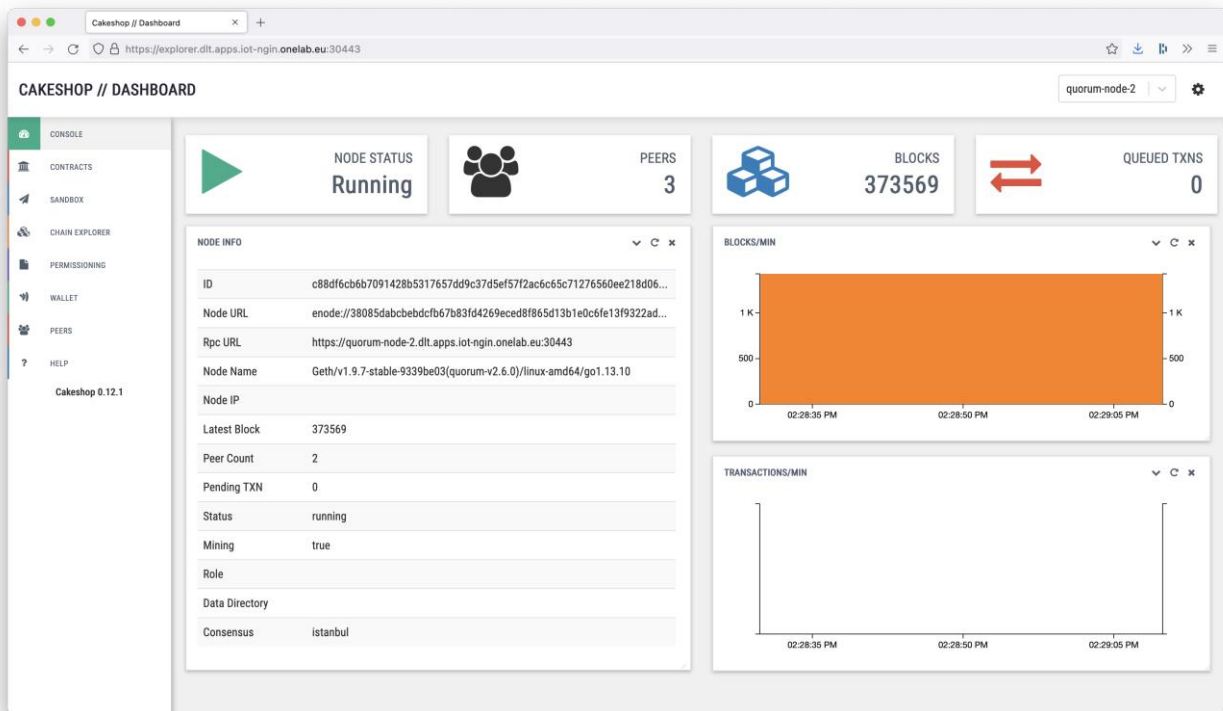


Figure 31 - Web application (Quorum blockchain web UI) exposed with the help of cert-manager and an Ingress controller.

Last, Figure 32 and Figure 33 showcase relevant resource listings for the cases of the project work related to the AR enablement and device indexing services and to the cybersecurity vulnerability crawler framework, respectively.

```
$ kubectl get all -n iot-ngin-wp4
```

NAME	READY	STATUS	RESTARTS	AGE
pod/device-inactivity-checker-27388101--1-cgdx9	0/1	Completed	0	22s
pod/orion-6dfccc5cc4-5blbr	1/1	Running	0	2d3h
pod/orion-mongodb-8c855765b-6f8p2	1/1	Running	0	2d3h

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/orion	ClusterIP	10.102.216.225	<none>	1026/TCP	2d3h
service/orion-mongodb	ClusterIP	10.111.83.89	<none>	27017/TCP	2d3h

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/orion	1/1	1	1	2d3h
deployment.apps/orion-mongodb	1/1	1	1	2d3h

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/orion-6dfccc5cc4	1	1	1	2d3h
replicaset.apps/orion-mongodb-8c855765b	1	1	1	2d3h

NAME	SCHEDULE	SUSPEND	ACTIVE	LAST SCHEDULE	AGE
cronjob.batch/device-inactivity-checker	*1 * * * *	False	0	23s	4h30m

NAME	COMPLETIONS	DURATION	AGE
job.batch/device-inactivity-checker-27388101	1/1	3s	23s

Figure 32 - Kubernetes resources consumed by WP4 activities.

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

```
$ kubectl get all -n iot-ngin-wp5
```

NAME	READY	STATUS	RESTARTS	AGE
pod/argocd-application-controller-7c564c89cf-46psc	1/1	Running	1 (5h32m ago)	5h33m
pod/argocd-dex-server-78696d64cb-gwq78	1/1	Running	2 (5h32m ago)	5h33m
pod/argocd-redis-5f95f98455-7wd25	1/1	Running	0	5h33m
pod/argocd-repo-server-7b6768b4df-wcd8m	1/1	Running	0	5h33m
pod/argocd-server-745784c9b-j5dnw	1/1	Running	1 (5h32m ago)	5h33m
pod/scanning-controller-d7c65c9dd-p9bpp	1/1	Running	0	5h29m
pod/vuln-crawler-device-manager-69fc68dc9c-6hvn7	1/1	Running	0	5h19m
pod/vuln-crawler-plugin-registry-59459f7986-fj5rv	1/1	Running	2 (5h5m ago)	5h5m
pod/vuln-crawler-plugin-registry-db-9db9f467b-698nq	1/1	Running	0	5h5m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/argocd-application-controller	ClusterIP	10.105.32.23	<none>	8082/TCP	5h33m
service/argocd-dex-server	ClusterIP	10.104.205.62	<none>	5556/TCP, 5557/TCP	5h33m
service/argocd-redis	ClusterIP	10.96.88.80	<none>	6379/TCP	5h33m
service/argocd-repo-server	ClusterIP	10.103.62.35	<none>	8081/TCP	5h33m
service/argocd-server	ClusterIP	10.110.181.87	<none>	80/TCP, 443/TCP	5h33m
service/scanning-controller-svc	ClusterIP	10.99.53.166	<none>	5000/TCP	26h
service/vuln-crawler-device-manager	NodePort	10.96.244.49	<none>	8080:32057/TCP	5h19m
service/vuln-crawler-plugin-registry	NodePort	10.100.246.82	<none>	80:31322/TCP	5h5m
service/vuln-crawler-plugin-registry-db	ClusterIP	10.103.112.120	<none>	5432/TCP	5h5m

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/argocd-application-controller	1/1	1	1	5h33m
deployment.apps/argocd-dex-server	1/1	1	1	5h33m
deployment.apps/argocd-redis	1/1	1	1	5h33m
deployment.apps/argocd-repo-server	1/1	1	1	5h33m
deployment.apps/argocd-server	1/1	1	1	5h33m
deployment.apps/scanning-controller	1/1	1	1	26h
deployment.apps/vuln-crawler-device-manager	1/1	1	1	5h19m
deployment.apps/vuln-crawler-plugin-registry	1/1	1	1	5h5m
deployment.apps/vuln-crawler-plugin-registry-db	1/1	1	1	5h5m

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/argocd-application-controller-7c564c89cf	1	1	1	5h33m
replicaset.apps/argocd-dex-server-78696d64cb	1	1	1	5h33m
replicaset.apps/argocd-redis-5f95f98455	1	1	1	5h33m
replicaset.apps/argocd-repo-server-7b6768b4df	1	1	1	5h33m
replicaset.apps/argocd-server-745784c9b	1	1	1	5h33m
replicaset.apps/scanning-controller-54fbbd876f	0	0	0	26h
replicaset.apps/scanning-controller-564fddcd9	0	0	0	26h
replicaset.apps/scanning-controller-65d597c595	0	0	0	26h
replicaset.apps/scanning-controller-665c8dc74	0	0	0	5h29m
replicaset.apps/scanning-controller-68fc6b7d98	0	0	0	26h
replicaset.apps/scanning-controller-84559d46b4	0	0	0	25h
replicaset.apps/scanning-controller-d7c65c9dd	1	1	1	5h29m
replicaset.apps/vuln-crawler-device-manager-69fc68dc9c	1	1	1	5h19m
replicaset.apps/vuln-crawler-plugin-registry-59459f7986	1	1	1	5h5m
replicaset.apps/vuln-crawler-plugin-registry-db-9db9f467b	1	1	1	5h5m

```
$
```

Figure 33 - Kubernetes resources consumed by WP5 activities.

## 4.2 Early validation activities

5G forms an important component of the IoT-NGIN meta-architecture underpinning providing reliable, secure connectivity of data and edge cloud hosting of IoT-NGIN components and applications. 5G is available to use as a live network for some of the Living Lab experiments but is very unlikely to be available for use in all Living Labs for validation of integrated components. In this context, laboratory tests of use cases typical of those which will be used in Living Labs have been prepared in the Ericsson Eurolab 5G live network laboratory.

As live data streams are not yet available from Living Labs, synthetic data streams typical of those which will be communicated in the Living Lab use cases were defined. In this section,

we describe our preparation work for the 5G live tests of IoT-NGIN use case utilizing typical data streams.

The data streams will be transmitted over a live 5G network to evaluate the latency performance of the 5G network in the laboratory at low power in an indoor setting and the results of the tests will be described in future deliverables such as D6.2. The results will describe how well 5G can support the requirements of the use cases.

## 4.2.1 Preparation of 5G performance tests of the IoT-NGIN use cases with synthetic data streams

This section describes the preparation of 5G performance tests of IoT-NGIN use cases with synthetic data streams and their typical communication characteristics. The aim of the tests is to simulate data streams typical of the project use cases as they will be trialed in the Living Labs of WP7, and the performance evaluation of the 5G system when transmitting those data streams. The data streams will be transmitted in a live 5G network in Ericsson Eurolab laboratories.

In order to prepare and design the synthetic data streams, a questionnaire was filled by the project partners involved in the Living Lab trials. The questionnaire asked for information about the use cases and the data stream of the living lab trial which will be the basis for the 5G tests in the EDD laboratory. Furthermore, it includes questions about the communication requirements of the project and future use cases as well as communication characteristics to provide background of the use cases (preferably in future scenarios of large-scale deployments in commercial networks). Annex 1 lists the completed questionnaires.

The following project use cases were investigated:

- Smart agriculture use cases:
  - Crop disease prediction, smart irrigation and precise spraying (UC4)
  - Sensor aided harvesting (UC5)
- Smart industry use cases:
  - Human-centered augmented reality assisted build-to-order assembly (UC6)
  - Digital powertrain and condition monitoring (UC8)
- Smart energy use cases:
  - Move from reacting to acting in smart grid monitoring and control (UC9)
  - Driver-friendly dispatchable electric vehicle charging (UC10)

### 4.2.1.1 Communication characteristics of the IoT-NGIN use cases

The communication characteristics of the IoT-NGIN use cases are described by the communication environment (e.g., urban or rural), communication direction (uplink and/or downlink), distance between end points (e.g., distance between user and base station), number and density of endpoints (number of communicating devices) as well as time synchronization.

In the following, the communication characteristics of the use cases are summarized of which information was available for use in this series of 5G tests. Of the 10 use cases defined

by the IoT-NGIN project, we have been able to define synthetic data streams and sets of tests for 5 of the 10 use cases.

**Smart Agriculture** use cases are applied in outdoor and rural environments with static obstacles like trees. The communication link is bidirectional, i.e. that the sensor nodes (SynField nodes) and a drone send their data to the server.

Table 5 - Smart Agriculture communication characteristics.

Characteristics	UC4	UC5
Communication protocol	MQTT, HTTP	MQTT, HTTP
Communication direction	Bidirectional	Bidirectional
Message size	20KB – 5MB	100KB – 5MB
Message frequency	0.02 - 1 message per second	5 messages per second
Message content	jpg, timestamp, battery level, localisation	timestamp, battery level, position of the robot, speed of the robot, image data (jpg)

The **Smart Industry** use cases are characterized by indoor activities, e.g., in a factory hall. Uplink only or both communication directions can be used. Around 6 endpoints are set up with a maximal distance of 1 to 2km.

Table 6 - Smart Industry communication characteristics.

Characteristics	UC8
Communication protocol	MQTT, OPC
Communication direction	Uplink
Message size	1KB – 400KB
Message frequency	0.02 - 0.2 message per second
Message content	Payload, message id, node id, datatype, browse Name



## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

**Smart Energy** is in outdoor, urban environments without relevant obstacles. Nevertheless, indoor and urban characteristics can appear. The number of endpoints and the distance between them might change. Table 7 - Smart Energy communication characteristics.

Characteristics	UC9	UC10
Communication protocol	MQTT, HTTP	MQTT
Communication direction	Uplink	Bidirectional
Message size	100MB/day	10B, 50B, 100B

## 5 Conclusions

Effective project activities leverage on well-structured and rigorous planning. Consortium-based projects, like IoT-NGIN, which comprise a set of subsystems developed by geographically dispersed and distributed teams, are facilitated in accomplishing complex project development tasks via detailed and concrete guidelines, in order to coordinate sporadic developments and facilitate their integration into a common platform..

This document presents the IoT-NGIN platform architecture at component level, drawing the main interactions among them and specifying indicative generic processes that are available through the IoT-NGIN platform. Moreover, the integration guidelines for the IoT-NGIN project are presented, including recommendations for the interfaces' technologies and data models and proper code and API documentation, which support to the open-source nature of the project. Moreover, DevSecOps practices are presented for IoT-NGIN, in order to maximize the level of automation and thus minimize the time, effort and risks implied for later integrating individual pieces of code, while injecting security practices within the integration and deployment lifecycle. The integration and testing infrastructure is also defined, comprising resources of the OneLab Facility of the Sorbonne University, the 5G testbed in Eurolab, Ericsson and the Real-time Laboratory of RWTH. Last, but not least, the first integration activities among components which are currently available, according to the IoT-NGIN plan, have been presented. Also, preparatory validation activities have been conducted, to support 5G performance tests in Eurolab.

The next round of integration of the IoT-NGIN components, along with testing and evaluation results will be provided in D6.2 "Integrated IoT-NGIN platform & laboratory testing results", which is due on the last quarter of 2022.



## 6 References

- [1] IoT-NGIN, "D1.2 - IoT meta-architecture, components, and benchmarking," H2020 957246 - IoT-NGIN Deliverable Report, 2021.
- [2] M. . Salvaris, D. . Dean and W. H. Tok, "Generative Adversarial Networks," *arXiv: Machine Learning*, vol. , no. , pp. 187-208, 2018.
- [3] "Open Source MANO Website," ETSI, [Online]. Available: <https://osm.etsi.org/>. [Accessed 30 11 2021].
- [4] EC - CEF Digital, "Context Broker Services," 2020. [Online]. Available: <https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/EU+Standards>. [Accessed 31 01 2022].
- [5] OpenAPI, "OpenAPI Specificaiton," OpenAPI, [Online]. Available: <https://swagger.io/specification/>.
- [6] IBM Cloud Education, "DevSecOps," IBM, 30 07 2020. [Online]. Available: <https://www.ibm.com/cloud/learn/devsecops>. [Accessed 17 12 2020].
- [7] GitLab, [Online]. Available: <https://about.gitlab.com>.
- [8] GitHub, [Online]. Available: <https://github.com>.
- [9] Docker, 2022. [Online]. Available: <https://www.docker.com>.
- [10] docker, "What is a Container?," Docker, 2020. [Online]. Available: <https://www.docker.com/resources/what-container>. [Accessed 12 2020].
- [11] kubernetes, "Production-Grade Container Orchestration," The Linux Foundation , 2020. [Online]. Available: <https://kubernetes.io>. [Accessed 12 2020].
- [12] IoT-NGIN, "D9.1 - Project Handbook," H2020-957246 IoT-NGIN Deliverable Report, 2020.
- [13] The Linux Foundation, "Rook: Open-Source, Cloud-Native Storage for Kubernetes," [Online]. Available: <https://rook.io/>. [Accessed Sep. 2021].
- [14] Ceph Foundation, "Ceph: The Future of Storage," [Online]. Available: <https://ceph.io/en/>. [Accessed Sep. 2021].
- [15] ETSI, "Open Source MANO (OSM)," [Online]. Available: <https://www.etsi.org/technologies/open-source-mano>. [Accessed Sep. 2021].

## Annex 1 IoT-NGIN Use Case Questionnaire

### Use Case 4 - Crop diseases prediction. Smart irrigation and precision aerial spraying

<b>Short description of the use case</b>
This use case aims to improve the prediction of crop diseases and, based on that, optimize watering and precision aerial spraying processes. The predictions will leverage Federated Learning models, based on images acquired via drones flying over the orchard and sensor measurements acquired via SYN <a href="#">SynField</a> IoT nodes.
<b>Data streams of the living lab trial (relevant for the 5G test in EDD laboratory)</b>
<b>End points:</b> Who is communicating with whom? Example: a device deployed in the field sends sensor data to a central server.
<p>SynField node sends sensor data to edge server</p> <p>Drone sends images to the Edge server</p> <p>Edge or cloud server sends control commands to the SynField node</p> <p>Edge server communicates with cloud server</p> <p>Mobile phone receives data from the edge or the cloud server (application)</p> <p>Mobile phone sends control commands to the edge or the cloud server</p>
<b>Protocol:</b> What communication protocol is used? For example: MQTT, UDP
<p>The SynField API is HTTP based (REST)</p> <p>It may be extended to support MQTT</p>
<b>Message size:</b> What is the size of messages? Example: each sensor data has 20 KB
<p>Drone message: image message size ~ 5MB (JPEG, JPG) or video message ~150MBs per minute</p> <p>For 1 SynField device: message size ~20KB</p> <p>For a network of SynField devices: message size ~40KB</p>
<b>Message frequency and periodicity:</b> How many messages are sent per second? Is the transmission periodical or not? Example: the field device sends 10 messages per second periodically, so that message frequency is 10 messages per second.

Drone: Configurable by the user; up to 30 messages per second if it is a video stream; up to 1 message per second if the drone collects images
SynField: Configurable by the user; 1 message can be sent every 5 min up to 24 hours. In case of power supply, 1 message / minute
<b>Content:</b> What data is inside of a message? Example: In each message of an AGV, there is a timestamp, battery level and position of the AGV.
Drone: 1) DATA messages include an image or a video stream, timestamp, battery level, coordinate of the capture point SynField: 2) DATA messages include measurements, timestamp, battery level and, optionally, coordinates of the SynField node 3) ACK: do not include timestamp, battery level and coordinates
<b>Sample data:</b> Is there sample data available (for example in json format) or data generator available from previous tests of the use case?
<b>Data aggregation:</b> Are multiple messages aggregated in one packet or are messages sent one by one?
Drone: No. SynField: Every message encapsulates a list of json objects
Requirements on communications (of project use cases and future use cases)
<b>Latency:</b> What is the maximum transmission time that the use case expects from the communication system? How much latency could the use case tolerate? Example: If the application does not receive a message within 20ms, then the data is outdated and cannot be used anymore.
No special latency requirements
<b>Throughput:</b> What is the minimum/maximum throughput required from the communication system? Example: Each field device is capable of sending 100 Mbps of data.
No special throughput requirements
<b>Reliability:</b> How reliable should be the communication system? What happens in case of a downtime/packet loss? Example: If the application does not receive consecutive X number of messages, then the AGV must break in order not to harm people.
Drone: 1 message loss could be harmful

<p>If consecutive SynField messages do not get acquired, then the disease prediction could be false.</p> <p>However, the application could tolerate several messages being lost, which could be e.g. of several hours (depending on the disease)</p>
<p><b>Security:</b> What security measures are required to secure the application from unauthorized access/cyber-attacks? Example: In a city, sensors for the traffic prediction should be physically protected against vandalism or weather conditions.</p>
<p>SynField nodes are installed on the fields. As such, the SynField electronic circuit is located in a waterproof box (IP-67), which protects the system from rain, snow and hail.</p> <p>Also, drones are manufactured for outdoor use.</p> <p>Apart from that, smart agriculture is vulnerable to cybersecurity attacks, which could result in control system intrusion.</p>
<p>Communication characteristics (background of the use cases preferably in the use case of a large-scale deployment in commercial networks in the future)</p>
<p><b>Communication environment:</b> Example: indoor/outdoor? Rural/urban? Obstacles expected?</p>
<p>Rural; obstacles are expected (static – e.g. trees, moving e.g. human workers)</p>
<p><b>Communication directions:</b> one-way (from device to application or vice versa) or two-way (between device and application in both ways). Example: a device sends sensor data to the application in a one-way manner.</p>
<p>SynField – Edge: bidirectional</p> <p>Drone – Edge: Bidirectional</p> <p>Mobile phone – Cloud: Bidirectional</p>
<p><b>Wireless distance between end points:</b> Example: The distance between the field device and the nearest 4G/5G base station.</p> <p>Not relevant for the test, but for the general requirements of the use case.</p>
<p>N/A</p>
<p><b>Number of end points:</b> What is the total number of communicating devices deployed in the field? Example: In a smart factory, there are 40 robot arms equipped with wireless communication.</p> <p>Not relevant for the test, but for the general requirements of the use case.</p>
<p>In a field, 3 SynField devices and 1 drone</p>
<p><b>Density of endpoints per km2:</b> How many communicating devices are deployed on an area? Example: in a smart factory, the density of devices is 80 endpoints per km2.</p>

Not relevant for the test, but for the general requirements of the use case.
N/A (1 drone, 3 SynField nodes for the whole orchard)
<b>Time synchronization:</b> Do the devices require to be synchronized? Example: two or more robot arms have to be synchronized in order to be able to interact with each other. Not relevant for the test, but for the general requirements of the use case.
No.
5G in the field trial
<b>Are you planning to use 5G in the field trial?</b> If not, what is the reason? What would be the alternative communication technology that you will be using in the trial?
No, there is no coverage in the field area Alternatively we will use 4G or 3G, depending on the coverage in the field area.
<b>5G coverage:</b> Is public 5G available at the trial location?
No
<b>Cooperation with mobile network operator:</b> Are you planning to cooperate with an operator?
Not within the project scope

## Use Case 5 - Sensor aided crop harvesting

Short description of the use case
IoT-NGIN Use Case #5 "Sensor aided crop harvesting" will experiment with a hybrid, semi-mechanical crop harvesting use case, in which Automated Guided Land Vehicles (AGLV) will support human-workers by autonomously carrying the crates to the loading point. We will experiment with agriculture AGLV serving as carrier machines, by enabling them to locate and avoid workers (for safety reasons) and trees (for operating reasons)
Data streams of the living lab trial (relevant for the 5G test in EDD laboratory)
<b>End points:</b> Who is communicating with whom? Example: a device deployed in the field sends sensor data to a central server.
AGLV sends images and sensor measurements to edge device Edge device sends data to the mobile device Mobile device sends control commands to Edge device (application) Edge or cloud server sends control commands to AGLV Edge server communicates with cloud server
<b>Protocol:</b> What communication protocol is used? For example: MQTT, UDP
HTTP, MQTT
<b>Message size:</b> What is the size of messages? Example: each sensor data has 20 KB
AGLV message: image message size ~ 5MB (JPEG, JPG) AGLV message: location info message size ~ 100KB
<b>Message frequency and periodicity:</b> How many messages are sent per second? Is the transmission periodical or not? Example: the field device sends 10 messages per second periodically, so that message frequency is 10 messages per second.
AGLV: Robot sends 5 messages per second
<b>Content:</b> What data is inside of a message? Example: In each message of an AGV, there is a timestamp, battery level and position of the AGV.
In each message of the robot there is a timestamp, battery level, position of the robot, speed of the robot, image data captured
<b>Sample data:</b> Is there sample data available (for example in json format) or data generator available from previous tests of the use case?

<p><b>Data aggregation:</b> Are multiple messages aggregated in one packet or are messages sent one by one?</p>
<p>1 message is sent in 1 packet</p> <p>Every message contains a set of parameters. The distinction is done based on the content, if it is image or general data to be send.</p>
Requirements on communications (of project use cases and future use cases)
<p><b>Latency:</b> What is the maximum transmission time that the use case expects from the communication system? How much latency could the use case tolerate? Example: If the application does not receive a message within 20ms, then the data is outdated and cannot be used anymore.</p>
<p>If inference takes place on the AGLV, then no special requirements for latency.</p> <p>It would be interesting, though, to see if inference taking place at the edge is feasible for obstacle avoidance. Even in this case, it is hard to estimate the transmission time requirements, given that the overall tolerated latency (from sensing to reaction) depends on a number of factors, including the camera/sensors, the AGLV velocity and accelerator, the use of GPUs/CPU's, the distance from which the obstacle is first identified, etc. With rough estimations for our use case, we could consider the transmission time tolerated at 15ms.</p>
<p><b>Throughput:</b> What is the minimum/maximum throughput required from the communication system? Example: Each field device is capable of sending 100 Mbps of data.</p>
Each AGLV should be capable of sending 150Kbps of data
<p><b>Reliability:</b> How reliable should be the communication system? What happens in case of a downtime/packet loss? Example: If the application does not receive consecutive X number of messages, then the AGV must break in order not to harm people.</p>
It could mean that the AGLV would not avoid an obstacle or a human
<p><b>Security:</b> What security measures are required to secure the application from unauthorized access/cyber-attacks? Example: In a city, sensors for the traffic prediction should be physically protected against vandalism or weather conditions.</p>
<p>The AGLV is constructed for outdoor use.</p> <p>Also, cybersecurity should be ensured, as it could result to control intrusion</p>
Communication characteristics (background of the use cases preferably in the use case of a large-scale deployment in commercial networks in the future)
<p><b>Communication environment:</b> Example: indoor/outdoor? Rural/urban? Obstacles expected?</p>

Outdoor / rural / Obstacles (trees, humans) are expected
<b>Communication directions:</b> one-way (from device to application or vice versa) or two-way (between device and application in both ways). Example: a device sends sensor data to the application in a one-way manner.
Bi-directional in any case
<b>Wireless distance between end points:</b> Example: The distance between the field device and the nearest 4G/5G base station.  Not relevant for the test, but for the general requirements of the use case.
N/A
<b>Number of end points:</b> What is the total number of communicating devices deployed in the field? Example: In a smart factory, there are 40 robot arms equipped with wireless communication.  Not relevant for the test, but for the general requirements of the use case.
1 AGLV  1 Edge server
<b>Density of endpoints per km2:</b> How many communicating devices are deployed on an area? Example: in a smart factory, the density of devices is 80 endpoints per km2.  Not relevant for the test, but for the general requirements of the use case.
1 AGLV
<b>Time synchronization:</b> Do the devices require to be synchronized? Example: two or more robot arms have to be synchronized in order to be able to interact with each other.  Not relevant for the test, but for the general requirements of the use case.
No
5G in the field trial
<b>Are you planning to use 5G in the field trial?</b> If not, what is the reason? What would be the alternative communication technology that you will be using in the trial?
No, there is no coverage in the field area  Alternatively, we will use 4G or 3G, depending on the coverage in the field area.
<b>5G coverage:</b> Is public 5G available at the trial location?
No
<b>Cooperation with mobile network operator:</b> Are you planning to cooperate with an operator?



Not within the project scope

## Use Case 7 - Human-centered augmented reality assisted build-to-order assembly

<b>Short description of the use case</b>
Utilising existing CAD-models to create AR assisted build-to-order assembly of cabinet drive units at the Pitäjänmäki factory site.
<b>Data streams of the living lab trial (relevant for the 5G test in EDD laboratory)</b>
<b>End points:</b> Who is communicating with whom? Example: a device deployed in the field sends sensor data to a central server.
Hololens, smart tools (e.g. screw driver ending torque data) communicating with an edge server containing the AR application and other software components.
<b>Protocol:</b> What communication protocol is used? For example: MQTT, UDP
-
<b>Message size:</b> What is the size of messages? Example: each sensor data has 20 KB
-
<b>Message frequency and periodicity:</b> How many messages are sent per second? Is the transmission periodical or not? Example: the field device sends 10 messages per second periodically, so that message frequency is 10 messages per second.
-
<b>Content:</b> What data is inside of a message? Example: In each message of an AGV, there is a timestamp, battery level and position of the AGV.
-
<b>Sample data:</b> Is there sample data available (for example in json format) or data generator available from previous tests of the use case?
No
<b>Data aggregation:</b> Are multiple messages aggregated in one packet or are messages sent one by one?
-

Requirements on communications (of project use cases and future use cases)
<b>Latency:</b> What is the maximum transmission time that the use case expects from the communication system? How much latency could the use case tolerate? Example: If the application does not receive a message within 20ms, then the data is outdated and cannot be used anymore.
The latency should be small enough to enable a smooth AR experience.
<b>Throughput:</b> What is the minimum/maximum throughput required from the communication system? Example: Each field device is capable of sending 100 Mbps of data.
-
<b>Reliability:</b> How reliable should be the communication system? What happens in case of a downtime/packet loss? Example: If the application does not receive consecutive X number of messages, then the AGV must break in order not to harm people.
User should be informed that system is dropping packets. Assumedly Hololens and related AR software can handle some packet loss. Smart tools need to be handled separately, however they are not the main focus of the use case.
<b>Security:</b> What security measures are required to secure the application from unauthorized access/cyber-attacks? Example: In a city, sensors for the traffic prediction should be physically protected against vandalism or weather conditions.
Ideally the setup would utilise a private 5G network. There is no need for outside access.
Communication characteristics (background of the use cases preferably in the use case of a large-scale deployment in commercial networks in the future)
<b>Communication environment:</b> Example: indoor/outdoor? Rural/urban? Obstacles expected?
Indoor, factory hall
<b>Communication directions:</b> one-way (from device to application or vice versa) or two-way (between device and application in both ways). Example: a device sends sensor data to the application in a one-way manner.
two-way most likely

<p><b>Wireless distance between end points:</b> Example: The distance between the field device and the nearest 4G/5G base station.</p> <p>Not relevant for the test, but for the general requirements of the use case.</p>
N/A
<p><b>Number of end points:</b> What is the total number of communicating devices deployed in the field? Example: In a smart factory, there are 40 robot arms equipped with wireless communication.</p> <p>Not relevant for the test, but for the general requirements of the use case.</p>
N/A
<p><b>Density of endpoints per km2:</b> How many communicating devices are deployed on an area? Example: in a smart factory, the density of devices is 80 endpoints per km2.</p> <p>Not relevant for the test, but for the general requirements of the use case.</p>
N/A
<p><b>Time synchronization:</b> Do the devices require to be synchronized? Example: two or more robot arms have to be synchronized in order to be able to interact with each other.</p> <p>Not relevant for the test, but for the general requirements of the use case.</p>
N/A
5G in the field trial
<p><b>Are you planning to use 5G in the field trial?</b> If not, what is the reason? What would be the alternative communication technology that you will be using in the trial?</p> <p>Yes, there should be an existing 5G setup available from a previous pilot project.</p>
<p><b>5G coverage:</b> Is public 5G available at the trial location?</p> <p>Yes</p>
<p><b>Cooperation with mobile network operator:</b> Are you planning to cooperate with an operator?</p> <p>Yes</p>

## Use Case 8 - Digital powertrain and condition monitoring

Short description of the use case
Gathering data from powertrain (Drive – Motor – Working machine / load) setups for condition monitoring and analytics.
Data streams of the living lab trial (relevant for the 5G test in EDD laboratory)
<b>End points:</b> Who is communicating with whom? Example: a device deployed in the field sends sensor data to a central server.
Various sensors (temperature, the drive unit itself, smart sensors, thermal camera...) are used to gather information about the powertrain. These are collected to a gateway device (Raspberry PI), which will be used for higher level IoT communications e.g. an edge or cloud server containing NGIN components.
<b>Protocol:</b> What communication protocol is used? For example: MQTT, UDP
The data processing on the gateway is currently done using node-red. The endpoints for NGIN components and higher level IoT operations will be made available on the gateway most likely using MQTT. The sensors themselves use various protocols (OPC DA/UA, MQTT, Plain TCP Byte Stream, FTP).
<b>Message size:</b> What is the size of messages? Example: each sensor data has 20 KB
Thermal Camera – Sends a simple array of values using MQTT. <1 KB?
Smart Sensor – OPC UA items, not sure of exact size.
Drive Unit – Depends on signal, some are sampled more often. Probably <1 KB per message?
PLC (temperature & acceleration data) – Temperature is read using OPC DA (not sure of size). Accelerometer data is fetched via FTP, 1 acceleration measurement file is 400 KB
<b>Message frequency and periodicity:</b> How many messages are sent per second? Is the transmission periodical or not? Example: the field device sends 10 messages per second periodically, so that message frequency is 10 messages per second.
Thermal Camera – Sends a new array of values every 5 seconds
Smart Sensor – Values are fetched using OPC UA every minute.

Drive Unit – Sends messages as bursts/batches every minute. Motor speed is sampled at a 1s interval and sent in as one batch while the next batch buffers. Some signals are sampled and sent less frequently e.g. temperature (1 min interval).

PLC – Temperature is read every 5 seconds. Acceleration files are fetched every 30 minutes if a new measurement is taken.

**Content:** What data is inside of a message? Example: In each message of an AGV, there is a timestamp, battery level and position of the AGV.

Thermal Cam – An array of bytes containing mostly data (a value indicating the temperature of the that specific element in the resulting 2d heatmap) but also some control words.

Smart Sensor – Temperature reading from smart sensor results in following json gateway side:

payload: 28.67614

topic: "ns=2;g=00007987-0003-d1dd-27fc-000000000004"

\_msgid: "59c53b60.b5fd34"

datatype: "Double"

browseName: "Skin Temperature"

Same format for rest of the measurements (acceleration and KPI values).

Drive unit – Unpacked result in json on gateway side for one of the signals:

```
{
  "objectId": "5806180602648987F",
  "model" : "abb.ability.device",
  "variable" : "01_11",
  "timestamp" : "2021-10-24T21:10:12+03:00",
  "value" : 1009.21,
  "unit" : "V"
},
{
  "objectId": "5806180602648987F",
  "model" : "abb.ability.device",
  "variable" : "01_11",
```

<pre>         "timestamp" : "2021-10-24T21:10:09+03:00",         "value" : 1010.25,         "unit" : "V"       },       ...     ]   } </pre>
PLC – OPC DA temperature readings and acceleration in WAV-file format
<b>Sample data:</b> Is there sample data available (for example in json format) or data generator available from previous tests of the use case?
<b>Data aggregation:</b> Are multiple messages aggregated in one packet or are messages sent one by one?
<p>Messages are sent one by one, however multiple samples are aggregated in one message.</p>
Requirements on communications (of project use cases and future use cases)
<b>Latency:</b> What is the maximum transmission time that the use case expects from the communication system? How much latency could the use case tolerate? Example: If the application does not receive a message within 20ms, then the data is outdated and cannot be used anymore.
<p>There are no strict latency requirements as data is used for analytics and monitoring purposes. High frequency data (accelerometer data) are sampled locally and sent in batches or made available for fetching via FTP, at which point the transmission time is not that crucial.</p>
<b>Throughput:</b> What is the minimum/maximum throughput required from the communication system? Example: Each field device is capable of sending 100 Mbps of data.
-
<b>Reliability:</b> How reliable should be the communication system? What happens in case of a downtime/packet loss? Example: If the application does not receive consecutive X number of messages, then the AGV must break in order not to harm people.

The application is not mission critical, currently nothing is done if messages stop reaching the gateway.
<b>Security:</b> What security measures are required to secure the application from unauthorized access/cyber-attacks? Example: In a city, sensors for the traffic prediction should be physically protected against vandalism or weather conditions.
Currently the setup uses private 4G (private IPs and SIMs provided by ISP). Gateway could possibly be opened to public networks. Ideally, the actual drive device responsible for motor control should always be out of reach of outside access.
Communication characteristics (background of the use cases preferably in the use case of a large-scale deployment in commercial networks in the future)
<b>Communication environment:</b> Example: indoor/outdoor? Rural/urban? Obstacles expected?
Indoor, factory hall.
<b>Communication directions:</b> one-way (from device to application or vice versa) or two-way (between device and application in both ways). Example: a device sends sensor data to the application in a one-way manner.
one-way, devices send sensor data to gateway.
<b>Wireless distance between end points:</b> Example: The distance between the field device and the nearest 4G/5G base station.
Not relevant for the test, but for the general requirements of the use case.
1-2 km
<b>Number of end points:</b> What is the total number of communicating devices deployed in the field? Example: In a smart factory, there are 40 robot arms equipped with wireless communication.
Not relevant for the test, but for the general requirements of the use case.
Currently around 6.
<b>Density of endpoints per km2:</b> How many communicating devices are deployed on an area? Example: in a smart factory, the density of devices is 80 endpoints per km2.
Not relevant for the test, but for the general requirements of the use case.
-



<b>Time synchronization:</b> Do the devices require to be synchronized? Example: two or more robot arms have to be synchronized in order to be able to interact with each other. Not relevant for the test, but for the general requirements of the use case.
Time synchronization could be useful for analytics, as it could allow combining data gathered from different sensors.
5G in the field trial
<b>Are you planning to use 5G in the field trial?</b> If not, what is the reason? What would be the alternative communication technology that you will be using in the trial?
Possibly, currently 4G is used. The connection requirements for this use case are not that extensive, so there has been no need to use 5G for demo purposes.
<b>5G coverage:</b> Is public 5G available at the trial location?
Yes
<b>Cooperation with mobile network operator:</b> Are you planning to cooperate with an operator?
Yes we are cooperating with an operator.

## Use Case 9 - Move from reacting to acting in smart grid monitoring and control

Short description of the use case
<p>The purpose of UC 9 is to monitor electrical quantities in real time and detect when the power quality is no longer adequate, so that it can be restored.</p> <p>The infrastructure to be used in this UC consists of 2 phasor measurement units (PMU), 6 power quality analysers (PQA) and 150 smart meters.</p>
Data streams of the living lab trial (relevant for the 5G test in EDD laboratory)
<p><b>End points:</b> Who is communicating with whom? Example: a device deployed in the field sends sensor data to a central server.</p> <p>Each of the sensors communicates with the LV SCADA where data are stored and shared, via authentication, with the outside world.</p>
<p><b>Protocol:</b> What communication protocol is used? For example: MQTT, UDP</p> <p>MQTT for PMU and Smart Meters Http for PQA</p>
<p><b>Message size:</b> What is the size of messages? Example: each sensor data has 20 KB</p> <p>PMU: 100 MB/day Smart meters: 100 MB/day PQA: 100 MB/day</p>
<p><b>Message frequency and periodicity:</b> How many messages are sent per second? Is the transmission periodical or not? Example: the field device sends 10 messages per second periodically, so that message frequency is 10 messages per second.</p> <p>PMU: 1 update per second Smart meters: 1 message per 5 seconds periodically PQA: 1 update per second</p>
<p><b>Content:</b> What data is inside of a message? Example: In each message of an AGV, there is a timestamp, battery level and position of the AGV.</p>

**PMU:**

"node_id":	node	identificative	number	
"stn":	station		number	
"vm1":	voltage	magnitude	phase	1
"vph1":	voltage	angle	phase	1
"vm2":	voltage	magnitude	phase	2
"vph2":	voltage	angle	phase	2
"vm3":	voltage	magnitude	phase	3
"vph3":	voltage	angle	phase	3
"vm4":	voltage	magnitude	phase	4
"vph4":	voltage	angle	phase	4
"im1":	current	magnitude	phase	1
"iph1":	current	angle	phase	1
"im2":	current	magnitude	phase	2
"iph2":	current	angle	phase	2
"im3":	current	magnitude	phase	3
"iph3":	current	angle	phase	3
"im4":	current	magnitude	phase	4
"iph4":	current	angle	phase	4
"f": frequency				

**SMART METERS:**

Voltage

MeterPoint

MeterType

Actor\_type

City

Country

GPS

Project

RBAC

Service

User\_name

SMXtimestamp

wallya1

Apparent Power 3-Phase

Apparent Power Phase L1

Apparent Power Phase L2

Apparent Power Phase L3  
Flicker Inst Max L1-N  
Flicker Inst Max L2-N  
Flicker Inst Max L3-N  
Flicker Plt L1-N  
Flicker Plt L2-N  
Flicker Plt L3-N  
Flicker Pst L1-N  
Flicker Pst L2-N  
Flicker Pst L3-N  
Fundamental PF 3-Phase  
Non-Active Power 3-Phase  
Non-Active Power Phase L1  
Non-Active Power Phase L2  
Non-Active Power Phase L3  
Non-Fundamental Power 3-Phase  
Non-Fundamental Power Phase L1  
Non-Fundamental Power Phase L2  
Non-Fundamental Power Phase L3  
Q1 Reactive Energy  
Q2 Reactive Energy  
Q3 Reactive Energy  
Q4 Reactive Energy  
Rms Current L4-N  
Rms Voltage L1-L2  
Rms Voltage L2-L3  
Rms Voltage L3-L1  
Rms Voltage L4-N  
THDI L1  
THDI L2  
THDI L3  
THDI L4  
THDV L1-N  
THDV L2-N

THDV L3-N

THDV L4-N

time

### PQA:

t1: initial measuring time and data

t2: end measuring time and data

p\_TOTAL\_avg: average active power over the time interval

q\_TOTAL\_avg: average reactive power over the time interval

v\_AB\_avg: average voltage between the first and second phase over the time interval

v\_BC\_avg: average voltage between the second and third phase over the time interval

v\_CA\_avg: average voltage between the third and first phase over the time interval

freq\_avg: average frequency over the time interval

a\_AN\_avg: average current of the first phase over the time interval

a\_BN\_avg: average current of the second phase over the time interval

a\_CN\_avg: average current of the third phase over the time interval

**Sample data:** Is there sample data available (for example in json format) or data generator available from previous tests of the use case?

### PMU:

```
{
  "node_id": "",
  "stn": "Station-0",
  "soc": 1602078510,
  "fracsec": 0,
  "vm1": 230.31439208984375,
  "vph1": -47.657395631859266,
  "vm2": 230.9148406982422,
  "vph2": 72.33367040491599,
  "vm3": 231.7793426513672,
  "vph3": -167.5164955836807,
  "vm4": 230.86802673339844,
  "vph4": -167.50331331858257,
  "im1": 0.000057650599046610296,
  "iph1": 52.590786627846114,
  "im2": 0.00003093302439083345,
  "iph2": -111.98963261279414,
```

```

"im3": 0.000042565585317788646,
"iph3": 76.26046227197222,
"im4": 0.000029644323149113916,
"iph4": 149.5296841780724,
"f": 49.98057174682617,
"df": 0.06389617919921875,
"digita": 0,
"analog": 0,
"vzsm": 0.3782951235771179,
"vzsph": -164.22327889422402,
"izsm": 0.00002246705480501987,
"izsph": 60.14759522742692
}

```

## SMART METERS

```

{"LD01":{"1-1-1-8-0-255":{"2":"1709647.307","5":"05/05/2021
10:40:20","Description":"Energy A+","Unit":"kWh"},"1-1-14-7-0-255":{"2":"050.017","5":"05/05/2021
10:40:20","Description":"Frequency","Unit":"Hz"},"1-1-16-7-0-255":{"2":"55.052","5":"05/05/2021
10:40:20","9":"053.259k","Description":"P","Unit":"W"},"1-1-2-8-0-255":{"2":"103756.6345","5":"05/05/2021
10:40:20","Description":"Energy A-","Unit":"kWh"},"1-1-23-7-0-255":{"2":"-3.641","5":"05/05/2021
10:40:20","9":"-03.641k","Description":"Q1","Unit":"W"},"1-1-3-7-0-255":{"2":"-12.581","5":"05/05/2021
10:40:20","9":"-12.581k","Description":"Q","Unit":"var"},"1-1-3-8-0-255":{"Description":"Energy R+","Unit":"kWh"},"1-1-31-7-0-255":{"2":"79.89","5":"05/05/2021
10:40:20","9":"079.890","Description":"I1","Unit":"A"},"1-1-32-7-0-255":{"2":"235.036","5":"05/05/2021
10:40:20","9":"235.236","Description":"U(1-0)","Unit":"V"},"1-1-33-7-0-255":{"Description":"K1","Unit":"NoUnit"},"1-1-36-7-0-255":{"2":"18.121","5":"05/05/2021
10:40:20","9":"018.162k","Description":"P1","Unit":"W"},"1-1-4-8-0-255":{"Description":"Energy R-","Unit":"kWh"},"1-1-43-7-0-255":{"2":"-4.755","5":"05/05/2021
10:40:20","9":"-04.755k","Description":"Q2","Unit":"W"},"1-1-51-7-0-255":{"2":"77.036","5":"05/05/2021
10:40:20","9":"077.036","Description":"I2","Unit":"A"},"1-1-52-7-0-255":{"2":"234.785","5":"05/05/2021
10:40:20","9":"234.785","Description":"U(2-0)","Unit":"V"},"1-1-53-7-0-255":{"Description":"K2","Unit":"NoUnit"},"1-1-56-7-0-255":{"2":"17.234","5":"05/05/2021
10:40:20","9":"017.257k","Description":"P2","Unit":"W"},"1-1-63-7-0-255":{"2":"-4.186","5":"05/05/2021
10:40:20","9":"-04.186k","Description":"Q3","Unit":"W"},"1-1-71-7-0-255":{"2":"86.378","5":"05/05/2021
10:40:20","9":"078.342","Description":"I3","Unit":"A"},"1-1-72-7-0-255":{"2":"234.894","5":"05/05/2021
10:40:20","9":"234.894","Description":"U(3-0)","Unit":"V"},"1-1-73-7-0-255":{"Description":"K3","Unit":"NoUnit"},"1-1-76-7-0-255":{"2":"19.698","5":"05/05/2021
10:40:20","9":"017.839k","Description":"P3","Unit":"W"},"Voltage":{"Unit":"V"},"_MeterPoint":"Grid
MV","_MeterType":"Wally-A:
172.16.1.111"},"_Actor_type":"Network","_City":"Terni","_Country":"Italy","_GPS":"42.569944,
12.651323","_Project":"H2020
Nobel

```

```
Grid","_RBAC":"SMXcore.2018.04.08.001","_Service":"Providing real-time values for G3M-
SCADA service","_User_name":"TERNI-MV","__SMXtimestamp":"05/05/2021
10:40:20","wallya1":{"Apparent Energy":{"value":"1843146.1218"},"Apparent Power 3-
Phase":{"value":"057.147k"},"Apparent Power Phase L1":{"value":"018.775k"},"Apparent
Power Phase L2":{"value":"018.119k"},"Apparent Power Phase
L3":{"value":"020.288k"},"Flicker Inst Max L1-N":{"value":"000.013"},"Flicker Inst Max L2-
N":{"value":"000.011"},"Flicker Inst Max L3-N":{"value":"000.018"},"Flicker Plt L1-
N":{"value":"000.349"},"Flicker Plt L2-N":{"value":"000.356"},"Flicker Plt L3-
N":{"value":"000.391"},"Flicker Pst L1-N":{"value":"000.070"},"Flicker Pst L2-
N":{"value":"000.064"},"Flicker Pst L3-N":{"value":"000.066"},"Fundamental PF 3-
Phase":{"value":"0.974"},"Non-Active Power 3-Phase":{"value":"015.253k"},"Non-Active
Power Phase L1":{"value":"004.907k"},"Non-Active Power Phase
L2":{"value":"005.521k"},"Non-Active Power Phase L3":{"value":"004.589k"},"Non-
Fundamental Power 3-Phase":{"value":"008.949k"},"Non-Fundamental Power Phase
L1":{"value":"003.445k"},"Non-Fundamental Power Phase L2":{"value":"002.868k"},"Non-
Fundamental Power Phase L3":{"value":"002.616k"},"Q1 Reactive
Energy":{"value":"104876.6044"},"Q2 Reactive Energy":{"value":"000498.5516"},"Q3
Reactive Energy":{"value":"011564.3003"},"Q4 Reactive
Energy":{"value":"084917.0184"},"Rms Current L4-N":{"value":"000.000"},"Rms Voltage L1-
L2":{"value":"406.746"},"Rms Voltage L2-L3":{"value":"406.685"},"Rms Voltage L3-
L1":{"value":"407.076"},"Rms Voltage L4-N":{"value":"000.000"},"THDI
L1":{"value":"018.316"},"THDI L2":{"value":"015.616"},"THDI L3":{"value":"012.370"},"THDI
L4":{"value":"n/a"},"THDV L1-N":{"value":"001.593"},"THDV L2-N":{"value":"001.654"},"THDV
L3-N":{"value":"001.626"},"THDV L4-N":{"value":"n/a"},"time":"05/05/2021 10:40:20"}}
```

## PQA

t1	t2	p_TOTAL_avg	q_TOTAL_avg	v_AB_avg	v_BC_avg	v_CA_avg	freq_avg	a_AN_avg	a_BN_avg	a_CN_avg
2018-05-22T08:30:00+1	2018-05-22T08:40:00+1	-0.64945	-0.71045	412.125	409.396	410.966	+0.00000	+0.00932	+0.00815	+0.01012
2018-05-22T08:40:00+1	2018-05-22T08:50:00+1	-0.64308	-0.70517	410.027	407.418	408.881	498.047	+0.00931	+0.00821	+0.01006
2018-05-22T08:50:00+1	2018-05-22T09:00:00+1	-0.64393	-0.70882	410.800	408.378	409.442	500.010	+0.00929	+0.00820	+0.01009
2018-05-22T09:00:00+1	2018-05-22T09:10:00+1	-0.63962	-0.70917	409.555	407.147	408.240	499.965	+0.00927	+0.00817	+0.01006
2018-05-22T09:10:00+1	2018-05-22T09:20:00+1	-0.64700	-0.71642	411.895	409.531	410.825	499.751	+0.00928	+0.00817	+0.01009
2018-05-22T09:20:00+1	2018-05-22T09:30:00+1	-0.64624	-0.72066	411.894	409.525	410.859	499.915	+0.00938	+0.00817	+0.01006
2018-05-22T09:30:00+1	2018-05-22T09:40:00+1	-0.64559	-0.72179	411.956	409.537	410.907	499.966	+0.00934	+0.00822	+0.01003
2018-05-22T09:40:00+1	2018-05-22T09:50:00+1	-0.64417	-0.71811	411.314	408.996	410.418	499.934	+0.00940	+0.00815	+0.01001

2018-05-22T09:50:00+1 2018-05-22T10:00:00+1 -0.64939 -0.71475 412.437 410.085 411.475 499.925 +0.00937 +0.00819 +0.01003
<b>Data aggregation:</b> Are multiple messages aggregated in one packet or are messages sent one by one?
No
Requirements on communications (of project use cases and future use cases)
<b>Latency:</b> What is the maximum transmission time that the use case expects from the communication system? How much latency could the use case tolerate? Example: If the application does not receive a message within 20ms, then the data is outdated and cannot be used anymore.
This UC does not require low latency
<b>Throughput:</b> What is the minimum/maximum throughput required from the communication system? Example: Each field device is capable of sending 100 Mbps of data.
This UC does not require throughput
<b>Reliability:</b> How reliable should be the communication system? What happens in case of a downtime/packet loss? Example: If the application does not receive consecutive X number of messages, then the AGV must break in order not to harm people.
No particular problems in case of downtime, except the loss of analysis data
<b>Security:</b> What security measures are required to secure the application from unauthorized access/cyber-attacks? Example: In a city, sensors for the traffic prediction should be physically protected against vandalism or weather conditions.
The data to be protected are in the LV SCADA, Sensors are protected as they are in private and not accessible locations.
Communication characteristics (background of the use cases preferably in the use case of a large-scale deployment in commercial networks in the future)
<b>Communication environment:</b> Example: indoor/outdoor? Rural/urban? Obstacles expected?
Outdoor,



Urban, No relevant obstacles
<b>Communication directions:</b> one-way (from device to application or vice versa) or two-way (between device and application in both ways). Example: a device sends sensor data to the application in a one-way manner.
One way
<b>Wireless distance between end points:</b> Example: The distance between the field device and the nearest 4G/5G base station. Not relevant for the test, but for the general requirements of the use case.
The distance depends on the substations' locations.
<b>Number of end points:</b> What is the total number of communicating devices deployed in the field? Example: In a smart factory, there are 40 robot arms equipped with wireless communication. Not relevant for the test, but for the general requirements of the use case.
There are 150 smart meters, 2 PMUs and 6 PQA, so a total of 158 end points
<b>Density of endpoints per km2:</b> How many communicating devices are deployed on an area? Example: in a smart factory, the density of devices is 80 endpoints per km2. Not relevant for the test, but for the general requirements of the use case.
The density of endpoints is about 10 endpoints/km <sup>2</sup>
<b>Time synchronization:</b> Do the devices require to be synchronized? Example: two or more robot arms have to be synchronized in order to be able to interact with each other. Not relevant for the test, but for the general requirements of the use case.
The synchronisation time is not relevant to the functioning of the components, except in the case of large deviations
5G in the field trial
<b>Are you planning to use 5G in the field trial?</b> If not, what is the reason? What would be the alternative communication technology that you will be using in the trial?

<b>yes</b>
<b>5G coverage:</b> Is public 5G available at the trial location?
<b>yes</b>
<b>Cooperation with mobile network operator:</b> Are you planning to cooperate with an operator?
<b>No</b>

## Use Case 10 - Driver-friendly dispatchable EV charging

Short description of the use case
<p>In UC10, electric vehicles (EV) and charging stations in the area of Terni (Italy) will be involved to realise driver-friendly scenarios for dispatchable charging of EVs based on energy demand-response due to renewable energy sources along with human-centred smart micro-contracts and micro-payments. The Terni pilot will utilize 4 substations and focus on the Medium Voltage/Low Voltage network branch managed by the SCOV secondary substation, which serves a 200 kW Photovoltaic local generation plant, which often has an electricity surplus, generated from fluctuating renewable energy sources (RES). A fleet of six EVs will be part of the pilot infrastructure together with 3 smart charging stations. IoT-NGIN will provide optimized grid management via EV charging smart scheduling using Digital Twin concept and advanced AI/ML-based analytics, along with the latest requirements on cybersecurity; furthermore charging stations discovery, recognition and indexing and eMobility augmented reality interaction will be tested in UC10.</p>
Data streams of the living lab trial (relevant for the 5G test in EDD laboratory)
<p><b>End points:</b> Who is communicating with whom? Example: a device deployed in the field sends sensor data to a central server.</p>
<p>Following device deployed in the field send data to a central server:</p> <ul style="list-style-type: none"> <li>• Smart meters (substations, PV plants)</li> <li>• Charging stations</li> <li>• Electric vehicles</li> </ul>
<p><b>Protocol:</b> What communication protocol is used? For example: MQTT, UDP</p>
<p>MQTT</p>
<p><b>Message size:</b> What is the size of messages? Example: each sensor data has 20 KB</p>
<ul style="list-style-type: none"> <li>• Smart meters: 100 B</li> <li>• Charging stations: 10 B</li> <li>• Electric vehicles: 50 B</li> </ul>
<p><b>Message frequency and periodicity:</b> How many messages are sent per second? Is the transmission periodical or not? Example: the field device sends 10 messages per second periodically, so that message frequency is 10 messages per second.</p>
<ul style="list-style-type: none"> <li>• Smart meters: 1 message per 5 seconds periodically</li> <li>• Charging stations: 1 message per 2 seconds periodically</li> <li>• Electric vehicles: 1 message per 5 seconds periodically</li> </ul>
<p><b>Content:</b> What data is inside of a message? Example: In each message of an AGV, there is a timestamp, battery level and position of the AGV.</p>
<ul style="list-style-type: none"> <li>• Smart meters: <ul style="list-style-type: none"> <li>◦ Voltage</li> </ul> </li> </ul>

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

- MeterPoint
- MeterType
- Actor\_type
- City
- Country
- GPS
- Project
- RBAC
- Service
- User\_name
- SMXtimestamp
- wallya1
- Apparent Power 3-Phase
- Apparent Power Phase L1
- Apparent Power Phase L2
- Apparent Power Phase L3
- Flicker Inst Max L1-N
- Flicker Inst Max L2-N
- Flicker Inst Max L3-N
- Flicker Plt L1-N
- Flicker Plt L2-N
- Flicker Plt L3-N
- Flicker Pst L1-N
- Flicker Pst L2-N
- Flicker Pst L3-N
- Fundamental PF 3-Phase
- Non-Active Power 3-Phase
- Non-Active Power Phase L1
- Non-Active Power Phase L2
- Non-Active Power Phase L3
- Non-Fundamental Power 3-Phase
- Non-Fundamental Power Phase L1
- Non-Fundamental Power Phase L2
- Non-Fundamental Power Phase L3
- Q1 Reactive Energy
- Q2 Reactive Energy
- Q3 Reactive Energy
- Q4 Reactive Energy
- Rms Current L4-N
- Rms Voltage L1-L2
- Rms Voltage L2-L3
- Rms Voltage L3-L1
- Rms Voltage L4-N
- THDI L1
- THDI L2
- THDI L3
- THDI L4
- THDV L1-N
- THDV L2-N

- THDV L3-N
  - THDV L4-N
  - time
- Charging stations:
  - Charging Session ID: unique identifier of a specific charging session;
  - dataStart: start time of a specific charging session;
  - dataStop: end time of a specific charging session;
  - Energy (Wh): amount of energy provided in a specific charging session;
  - Socket ID: unique identifier of a specific charging station socket;
  - Charging Station ID: unique identifier of a specific charging station;
  - Voltage: real-time charging station voltage value (V);
  - Electric Current: real-time charging station electric current value (A);
  - Power: real-time charging station power value (W).
- Electric vehicles:
  - Measure ID: unique identifier of a specific measurement;
  - Vehicle ID: unique identifier of a specific electric vehicle;
  - Brand: EV manufacturer name;
  - Model: EV model name;
  - Battery Power (kW): maximum EV charging power value;
  - Battery Capacity (kWh): maximum EV battery energy capacity value;
  - Connector Type: EV charging connector type name;
  - Autonomy (km): real-time EV kilometers autonomy value;
  - Odometer (km): real-time EV odometer value;
  - SoC (%): real-time EV State of Charge percentage value;
  - Timestamp: record of the time of measurement event.

**Sample data:** Is there sample data available (for example in json format) or data generator available from previous tests of the use case?

- Smart meters:

```
{
  "LD01": {
    "1-1-1-8-0-255": {
      "-2": "1709647.307",
      "-5": "05/05/2021 10:40:20",
      "Description": "Energy A+",
      "Unit": "kWh"
    },
    "1-1-14-7-0-255": {
      "-2": "050.017",
      "-5": "05/05/2021 10:40:20",
      "Description": "Frequency",
      "Unit": "Hz"
    },
    "1-1-16-7-0-255": {
      "-2": "55.052",
      "-5": "05/05/2021 10:40:20",
      "-9": "053.259k",
      "Description": "P",
      "Unit": "W"
    },
    "1-1-28-0-255": {
      "-2": "103756.6345",
      "-5": "05/05/2021 10:40:20",
      "Description": "Energy A-",
      "Unit": "kWh"
    },
    "1-1-23-7-0-255": {
      "-2": "-3.641",
      "-5": "05/05/2021 10:40:20",
      "-9": "-03.641k",
      "Description": "Q1",
      "Unit": "W"
    },
    "1-1-3-7-0-255": {
      "-2": "-12.581",
      "-5": "05/05/2021 10:40:20",
      "-9": "-12.581k",
      "Description": "Q",
      "Unit": "var"
    },
    "1-1-3-8-0-255": {
      "Description": "Energy R+",
      "Unit": "kWh"
    },
    "1-1-31-7-0-255": {
      "-2": "79.89",
      "-5": "05/05/2021 10:40:20",
      "-9": "079.890",
      "Description": "I1",
      "Unit": "A"
    },
    "1-1-32-7-0-255": {
      "-2": "235.036",
      "-5": "05/05/2021 10:40:20",
      "-9": "235.236",
      "Description": "U(1-0)",
      "Unit": "V"
    },
    "1-1-33-7-0-255": {
      "Description": "K1",
      "Unit": "NoUnit"
    },
    "1-1-36-7-0-255": {
      "-2": "18.121",
      "-5": "05/05/2021 10:40:20",
      "-9": "018.162k",
      "Description": "P1",
      "Unit": "W"
    },
    "1-1-4-8-0-255": {
      "Description": "Energy R-",
      "Unit": "kWh"
    },
    "1-1-43-7-0-255": {
      "-2": "-4.755",
      "-5": "05/05/2021 10:40:20",
      "-9": "-04.755k",
      "Description": "Q2",
      "Unit": "W"
    },
    "1-1-51-7-0-255": {
      "-2": "77.036",
      "-5": "05/05/2021 10:40:20",
      "-9": "077.036",
      "Description": "I2",
      "Unit": "A"
    },
    "1-1-52-7-0-255": {
      "-2": "234.785",
      "-5": "05/05/2021 10:40:20",
      "-9": "234.785",
      "Description": "U(2-0)",
      "Unit": "V"
    },
    "1-1-53-7-0-255": {
      "Description": "K2",
      "Unit": "NoUnit"
    },
    "1-1-56-7-0-255": {
      "-2": "17.234",
      "-5": "05/05/2021"
    }
  }
}
```

```

10:40:20",-9":"017.257k","Description":"P2","Unit":"W"},"1-1-63-7-0-255":{"-2":"-4.186","-
5":"05/05/2021      10:40:20",-9":"-04.186k","Description":"Q3","Unit":"W"},"1-1-71-7-0-
255":{"-2":"86.378","-5":"05/05/2021      10:40:20",-
9":"078.342","Description":"I3","Unit":"A"},"1-1-72-7-0-255":{"-2":"234.894","-
5":"05/05/2021      10:40:20",-9":"234.894","Description":"U(3-0)","Unit":"V"},"1-1-73-7-0-
255":{"Description":"K3","Unit":"NoUnit"},"1-1-76-7-0-255":{"-2":"19.698","-5":"05/05/2021
10:40:20",-
9":"017.839k","Description":"P3","Unit":"W"},"Voltage":{"Unit":"V"},"_MeterPoint":"Grid
MV","_MeterType":"Wally-A:
172.16.1.111"},"_Actor_type":"Network","_City":"Terni","_Country":"Italy","_GPS":"42.569
944,      12.651323","_Project":"H2020      Nobel
Grid","_RBAC":"SMXcore.2018.04.08.001","_Service":"Providing real-time values for
G3M-SCADA      service","_User_name":"TERNI-MV","__SMXtimestamp":"05/05/2021
10:40:20","wallya1":{"Apparent Energy":{"value":"1843146.1218"},"Apparent Power 3-
Phase":{"value":"057.147k"},"Apparent      Power      Phase
L1":{"value":"018.775k"},"Apparent Power Phase L2":{"value":"018.119k"},"Apparent
Power Phase L3":{"value":"020.288k"},"Flicker Inst Max L1-N":{"value":"000.013"},"Flicker
Inst Max L2-N":{"value":"000.011"},"Flicker Inst Max L3-N":{"value":"000.018"},"Flicker Plt
L1-N":{"value":"000.349"},"Flicker Plt L2-N":{"value":"000.356"},"Flicker Plt L3-
N":{"value":"000.391"},"Flicker Pst L1-N":{"value":"000.070"},"Flicker Pst L2-
N":{"value":"000.064"},"Flicker Pst L3-N":{"value":"000.066"},"Fundamental PF 3-
Phase":{"value":"0.974"},"Non-Active Power 3-Phase":{"value":"015.253k"},"Non-
Active Power Phase L1":{"value":"004.907k"},"Non-Active Power Phase
L2":{"value":"005.521k"},"Non-Active Power Phase L3":{"value":"004.589k"},"Non-
Fundamental Power 3-Phase":{"value":"008.949k"},"Non-Fundamental Power Phase
L1":{"value":"003.445k"},"Non-Fundamental      Power      Phase
L2":{"value":"002.868k"},"Non-Fundamental Power Phase L3":{"value":"002.616k"},"Q1
Reactive      Energy":{"value":"104876.6044"},"Q2      Reactive
Energy":{"value":"000498.5516"},"Q3 Reactive Energy":{"value":"011564.3003"},"Q4
Reactive Energy":{"value":"084917.0184"},"Rms Current L4-N":{"value":"000.000"},"Rms
Voltage L1-L2":{"value":"406.746"},"Rms Voltage L2-L3":{"value":"406.685"},"Rms
Voltage L3-L1":{"value":"407.076"},"Rms Voltage L4-N":{"value":"000.000"},"THDI
L1":{"value":"018.316"},"THDI L2":{"value":"015.616"},"THDI L3":{"value":"012.370"},"THDI
L4":{"value":"n/a"},"THDV      L1-N":{"value":"001.593"},"THDV      L2-
N":{"value":"001.654"},"THDV      L3-N":{"value":"001.626"},"THDV      L4-
N":{"value":"n/a"},"time":"05/05/2021 10:40:20"}}

```

- Charging stations:

```

{
  "chargeboxID": "24",
  "address": "ASM Terni, Strada di Maratta Bassa, TR - Parcheggio",
  "latitude": "42.5673558",
  "longitude": "12.6070454",
  "maxPwrAC": "64",
  "maxPwrDC": "0",
  "drStatus": "1",
  "idSocketA": "37",
  "tSocketA": "type 2",

```

## D6.1 - Integration guidelines &amp; initial IoT-NGIN components integration

```
"stSocketA": "charging",
"idSocketB": "38",
"tSocketB": "type 2",
"stSocketB": "waiting"
}

{
  "numCharge": "686",
  "recharges": [
    {
      "chargeID": "12523",
      "dataStart": "2020-12-18 15:19:03",
      "dataStop": "notEnded",
      "kWh": 0,
      "importoTot": "0",
      "address": "ASM Terni, Strada di Maratta Bassa, TR - Parcheggio",
      "idUser": "1403",
      "socketID": "37",
      "chargeboxID": "24",
      "nomePresa": "presa A"
    },
    {
      "chargeID": "12477",
      "dataStart": "2020-12-16 11:30:21",
      "dataStop": "2020-12-16 13:32:04",
      "kWh": 23,
      "importoTot": "8.28",
      "address": "ASM Terni, Strada di Maratta Bassa, TR - Parcheggio",
      "idUser": "1403",
      "socketID": "37",
      "chargeboxID": "24",
      "nomePresa": "presa A"
    }
  ]
}
```

```
{
  "powerMeasurement": {
    "description": "P",
    "value": "8180",
    "timestamp": "12/01/2020, 09:36:16",
    "unit": "kW"
  }
}
{
  "powerMeasurement": {
    "description": "P",
    "value": "8180",
    "timestamp": "12/01/2020, 09:36:18",
    "unit": "kW"
  }
}
{
  "powerMeasurement": {
    "description": "P",
    "value": "8000",
    "timestamp": "12/01/2020, 09:36:22",
    "unit": "kW"
  }
}
```

- Electric vehicles:

```
[{"characteristic":{"vehicleID":"10","model":"Renault
ZOE","connector":"type2","batteryKw":"41","batteryPower":"22","license
Plate":"FE132DG"},"status":{"status":"disconnected","timestamp":"2020-
11-16
10:07:06","autonomyKm":"261","speed":"0","batteryPerc":"100","latitude
":"43.0720388833333","longitude":"12.3419610333333","ready":false,"doo
rsLocked":"yes","frontDX":"close","frontSX":"close","rearDX":"close","
rearSX":"close","carTrunk":"close"}}]
```

**Data aggregation:** Are multiple messages aggregated in one packet or are messages sent one by one?

- Smart meters: messages sent one by one
- Charging stations: messages sent one by one
- Electric vehicles: messages sent one by one

Requirements on communications (of project use cases and future use cases)

**Latency:** What is the maximum transmission time that the use case expects from the communication system? How much latency could the use case tolerate? Example: If the application does not receive a message within 20ms, then the data is outdated and cannot be used anymore.

1 second

**Throughput:** What is the minimum/maximum throughput required from the communication system? Example: Each field device is capable of sending 100 Mbps of data.

Each field device is capable of sending 30 Mbps of data



<p><b>Reliability:</b> How reliable should be the communication system? What happens in case of a downtime/packet loss? Example: If the application does not receive consecutive X number of messages, then the AGV must break in order not to harm people.</p>
<p>If the application does not receive consecutive X number of messages, then the effectiveness of the smart charging system is reduced, losing the real-time balance between energy consumption and local production.</p>
<p><b>Security:</b> What security measures are required to secure the application from unauthorized access/cyber-attacks? Example: In a city, sensors for the traffic prediction should be physically protected against vandalism or weather conditions.</p>
<p>In UC10, sensors are physically protected against vandalism or weather conditions; furthermore, different measures for preventing/mitigate cyber-attacks are under investigation for field implementation.</p>
<p>Communication characteristics (background of the use cases preferably in the use case of a large-scale deployment in commercial networks in the future)</p>
<p><b>Communication environment:</b> Example: indoor/outdoor? Rural/urban? Obstacles expected?</p>
<p>UC10 devices are located both indoor and outdoor and are located both in rural and urban environment.</p>
<p><b>Communication directions:</b> one-way (from device to application or vice versa) or two-way (between device and application in both ways). Example: a device sends sensor data to the application in a one-way manner.</p>
<p>Two-way communication</p>
<p><b>Wireless distance between end points:</b> Example: The distance between the field device and the nearest 4G/5G base station.</p>
<p>Not relevant for the test, but for the general requirements of the use case.</p>
<p>Less than one kilometer for smart meters and charging stations.</p>
<p>For electric vehicles the distance is dynamic.</p>
<p><b>Number of end points:</b> What is the total number of communicating devices deployed in the field? Example: In a smart factory, there are 40 robot arms equipped with wireless communication.</p>
<p>Not relevant for the test, but for the general requirements of the use case.</p>
<p>Terni Pilot:</p> <ul style="list-style-type: none"> <li>• 4 substations</li> <li>• 2 PV plants</li> <li>• 3 charging stations</li> <li>• 6 electric vehicles</li> </ul> <p>Terni city:</p> <ul style="list-style-type: none"> <li>• 700 substations</li> </ul>

<ul style="list-style-type: none"> <li>• 690 PV plants</li> <li>• 50 charging stations</li> <li>• 100 electric vehicles</li> </ul>
<b>Density of endpoints per km2:</b> How many communicating devices are deployed on an area? Example: in a smart factory, the density of devices is 80 endpoints per km2. Not relevant for the test, but for the general requirements of the use case.
10 endpoints per km2
<b>Time synchronization:</b> Do the devices require to be synchronized? Example: two or more robot arms have to be synchronized in order to be able to interact with each other. Not relevant for the test, but for the general requirements of the use case.
Smart charging system is based on real-time consumption via EV charging of locally produced renewable energy surplus
5G in the field trial
<b>Are you planning to use 5G in the field trial?</b> If not, what is the reason? What would be the alternative communication technology that you will be using in the trial?
yes
<b>5G coverage:</b> Is public 5G available at the trial location?
yes
<b>Cooperation with mobile network operator:</b> Are you planning to cooperate with an operator?
No