



## D2.2 Pledger Requirements Analysis

Document Identification			
Status	Final	Due Date	31/08/2020
Version	2.0	Submission Date	24/08/2020

Related WP	WP2	Document Reference	D2.2
Related Deliverable(s)	D2.1 (Pledger Detailed Use Cases)	Dissemination Level (*)	PU
Lead Participant	ENG	Lead Author	Francesco Iadanza, Gabriele Giammatteo
Contributors	ATOS, ICCS, INTRA, INNOV, FILL, i2CAT, HOLO, IMI	Reviewers	August Betzler, i2CAT Orfefs Voutyras, Antonis Litke, ICCS

Keywords:
Stakeholders, User stories, State of the art, Requirements analysis, Minimal Viable Product

### Disclaimer for Deliverables with dissemination level PUBLIC

This document is issued within the frame and for the purpose of the PLEDGER project. This project has received funding from the European Union's Horizon2020 Framework Programme under Grant Agreement No. 871536. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

The dissemination of this document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains. **This deliverable is subject to final acceptance by the European Commission.**

This document and its content are the property of the PLEDGER Consortium. The content of all or parts of this document can be used and distributed provided that the PLEDGER project and the document are properly referenced.

Each PLEDGER Partner may use this document in conformity with the PLEDGER Consortium Grant Agreement provisions.

(\*) Dissemination level: **PU**: Public, fully open, e.g. web; **CO**: Confidential, restricted under conditions set out in Model Grant Agreement; **CI**: Classified EU RESTRICTED, EU CONFIDENTIAL, **Int** = Internal Working Document, information as referred to in Commission Decision 2001/844/EC.



This document is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871536.

## Document Information

List of Contributors	
Name	Partner
Roi Sucanas Font	ATOS
Antonio Castillo	ATOS
Francesco Iadanza	ENG
Gabriele Giammatteo	ENG
Stefan Murauer	FILL
Verena Stanzl	FILL
Alexander Werlberger	HOLO
Carina Pamminger	HOLO
August Betzler	i2CAT
Leonardo Ochoa-Aday	i2CAT
Orfevs Voutyras	ICCS
Gonzalo Cabezas	IMI
Nikolaos Kapsoulis	INNOV
Olga Segou	INTRA
Ioannis Sarris	INTRA

Document History			
Version	Date	Change editors	Changes
0.1	25/02/2020	ENG	Initial ToC
0.2	03/03/2020	ICCS	New version of the ToC
0.3	26/03/2020	ENG	Updated ToC and initial SotA
0.4	04/06/2020	ENG	Section 1-3 completed
0.5	17/07/2020	ENG	Sections 1-3 reviewed; section 4-5 updated
0.6	29/07/2020	ENG	Section 1: updated methodology description; section 4 and 5: added functional groups and updated diagrams
0.7	20/08/2020	i2CAT, ICCS	Final revision
1.0	21/08/2020	ENG	Version ready for submission
2.0	24/08/2020	ATOS	Clean version for submission

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Francesco Iadanza (ENG)	21/08/2020
Quality manager	Cesar Mediavilla (ATOS)	24/08/2020
Project Coordinator	Ana Juan (ATOS)	24/08/2020

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	2 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

## Disclaimer

---

This document may contain material that is copyright of certain Pledger project beneficiaries, and may not be reproduced or copied without permission. The commercial use of any information contained in this document may require a license from the proprietor of that information. The Pledger project is part of the European Community's Horizon 2020 Program for research and development and is as such funded by the European Commission. All information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability with respect to this document, which is merely representing the authors' view.

The Pledger Consortium consists of the following organisations:

Number	Participant organisation name	Short name	Country
1	ATOS SPAIN SA	ATOS	Spain
2	INSTITUTE OF COMMUNICATION AND COMPUTER SYSTEMS	ICCS	Greece
3	ENGINEERING - INGEGNERIA INFORMATICA SPA	ENG	Italy
4	FUNDACIO PRIVADA I2CAT, INTERNET I INNOVACIO DIGITAL A CATALUNYA	I2CAT	Spain
5	HOLO-INDUSTRIE 4.0 SOFTWARE GMBH	HOLO	Germany
6	INTRASOFT INTERNATIONAL SA	INTRA	Luxemburg
7	FILL GESELLSCHAFT MBH	FILL	Austria
8	INNOV-ACTS LIMITED	INNOV	Cyprus
9	INSTITUT MUNICIPAL D'INFORMATICA DE BARCELONA	IMI	Spain

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	3 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

## Table of Contents

---

Document Information .....	2
Disclaimer .....	3
Table of Contents .....	4
List of Tables.....	7
List of Figures .....	8
List of Acronyms.....	9
Executive Summary .....	12
1 Introduction.....	13
1.1 Purpose of the document.....	13
1.2 Relation to other project activities .....	13
1.3 Approach used for the requirements elicitation and analysis .....	13
1.4 Structure of the document.....	15
1.5 Terminology adopted in this document .....	15
2 Use Cases overview, Stakeholders & User Stories .....	16
2.1 Use cases overview .....	16
2.1.1 Mixed reality applications on the edge .....	16
2.1.2 Edge Infrastructure for enhancing safety of vulnerable road users.....	16
2.1.3 Manufacturing the data mining on edge.....	17
2.2 Stakeholders value chain .....	17
2.2.1 Infrastructure providers .....	18
2.2.2 System integrators .....	18
2.2.3 Service providers.....	18
2.2.4 End users .....	18
2.3 User Stories.....	19
3 Relevant Technologies for the Project.....	23
3.1 Slice Orchestration Engine for Cloud, Edge and Radio Infrastructures .....	23
3.1.1 Architecture and Way of Operation .....	24
3.1.2 Initial feasibility study.....	26
3.1.3 Expected benefits for the Pledger system .....	26
3.1.4 SWOT analysis.....	27
3.2 Containerisation on the edge.....	28
3.2.1 Edge computing requirements.....	28

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	4 of 101	
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	
	<b>Version:</b>	2.0	<b>Status:</b>	Final

3.2.2	Containers on the edge requirements .....	30
3.2.3	Open source container-based edge platforms and challenges .....	31
3.2.4	Initial feasibility study.....	34
3.2.5	Expected benefits for the Pledger system .....	34
3.2.6	SWOT analysis.....	35
3.3	Big Data .....	36
3.3.1	Overview of Big Data Technologies .....	36
3.3.2	The Streamhandler Big Data Platform .....	37
3.3.3	Streamhandler Integration & Interoperability .....	40
3.3.4	Expected benefits for the Pledger system .....	41
3.3.5	SWOT Analysis.....	41
3.4	Continuous Integration/Continuous Delivery .....	43
3.4.1	Overview of CI/CD .....	43
3.4.2	The Pledger CI/CD Environment .....	43
3.4.3	Expected benefits for the Pledger system .....	47
3.4.4	SWOT analysis.....	47
3.5	Blockchain .....	48
3.5.1	Microtransactions Framework for the Edge: Requirements .....	48
3.5.2	Overview of Blockchain Technologies .....	49
3.5.3	Expected benefits for the Pledger system .....	54
3.5.4	SWOT Analysis.....	54
3.6	Benchmarking .....	55
3.6.1	Cloud Benchmarking.....	55
3.6.2	Edge Benchmarking .....	57
3.6.3	Expected benefits for the Pledger system .....	58
3.6.4	SWOT Analysis.....	59
3.7	Cloud and edge SLA monitoring and evaluation.....	60
3.7.1	SLA/QoS monitoring and evaluation in cloud .....	60
3.7.2	SLA/QoS monitoring on edge.....	61
3.7.3	Expected benefits for the Pledger system .....	63
3.7.4	SWOT Analysis.....	63
3.8	Monitoring applications and infrastructures in Fog and Edge.....	65
3.8.1	Monitoring tools .....	65
3.8.2	Expected benefits for the Pledger system .....	66

3.8.3	SWOT Analysis.....	66
3.9	AR Streaming on the Edge .....	67
3.9.1	Overview .....	67
3.9.2	Fraunhofer Instant3Dhub Service .....	67
3.9.3	Microsoft Azure Remote Rendering Service .....	67
3.9.4	Holo-Light Multiplatform ISAR .....	67
3.9.5	Expected benefits for the Pledger system .....	68
3.9.6	SWOT analysis.....	68
4	Requirements' Consolidation and Coding.....	69
4.1	Approach.....	69
4.2	Functional Requirements .....	69
4.3	Non-Functional Requirements .....	82
4.4	Requirements report.....	90
5	Pledger Minimal Viable Product .....	93
5.1	Definition .....	93
5.2	MVP methodology in Pledger .....	93
6	Conclusions and next steps .....	96
7	Bibliography .....	97
Annexes	.....	100
	README tab, with instructions and rationale .....	100
	MVP_*** tabs .....	100
	MoSCoW_FR and MoSCoW_NFR tabs, with changed priorities .....	100

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	6 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

## List of Tables

---

<i>Table 1: User stories</i>	19
<i>Table 2: Slice Orchestration Framework for Cloud, Edge and Radio Infrastructures SWOT</i>	27
<i>Table 3: Kubernetes on the edge claimed production-readiness</i>	31
<i>Table 4: Containerisation on the edge SWOT table</i>	35
<i>Table 5: Big Data Platform SWOT analysis</i>	42
<i>Table 6: CI/CD Platform SWOT analysis</i>	47
<i>Table 7: Microtransactions Framework for the edge SWOT table</i>	54
<i>Table 8: Benchmarking SWOT analysis</i>	59
<i>Table 9: SLA and QoS evaluation SWOT table</i>	64
<i>Table 10: Monitoring applications and infrastructures in Fog and Edge SWOT</i>	66
<i>Table 11: Streaming on the Edge SWOT</i>	68
<i>Table 12: Functional requirements</i>	70
<i>Table 13: Non-functional requirements</i>	82
<i>Table 14: sample table for the MVP voting</i>	94
<i>Table 15: MoSCoW thresholds for MVP</i>	95

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	7 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

## List of Figures

---

Figure 1: Requirement management approach	13
Figure 2: Requirements management lifecycle	14
Figure 3: Pledger stakeholders value chain	18
Figure 4: Functional architecture of SOE and RAN Controller solutions	25
Figure 5: System architecture of SOE and RAN Controller solutions	25
Figure 6: Edge computing architectures (source: Canonical)	29
Figure 7: edge nodes vs latency (source: Canonical)	29
Figure 8: The 5V characteristics of Big Data & related architectural choices for Pledger	36
Figure 9: Big Data solution – Conceptual Architecture.	37
Figure 10: Streamhandler Platform - High level Architecture	38
Figure 11: Streamhandler Platform – Monitoring Dashboard	40
Figure 12: The Continuous Integration Lifecycle.	43
Figure 13: Continuous Integration & Continuous Delivery process	45
Figure 14: Flow from the docker registry to the deployment server	46
Figure 15: Code Changes Workflow	46
Figure 16: Multi-hop blockchain-empowered mobile edge computing	48
Figure 17: Integrated blockchain and edge computing systems	49
Figure 18: Satoshiwall platform.	50
Figure 19: Boltwall Flowchart & Explanation.	51
Figure 20: Visual representation of Smart Contract concept	52
Figure 21: Logo of Ethereum	52
Figure 22: Logos of Hyperledger, Quorum, R3 Corda	53
Figure 23: Basic paradigm of an edge computing framework (Taherizadeh)	62
Figure 24: RightScale 2019 cloud maturity in enterprises	63
Figure 25: Public and private cloud usage	64
Figure 26: Functional requirements' distribution categories	90
Figure 27: Functional requirements' groups	90
Figure 28: Non-functional requirements' distribution categories	91
Figure 29: Requirements' sources	91
Figure 30: Requirements' stakeholders	92
Figure 31: build-measure-learn cycle (source: Eric Ries)	93
Figure 32: MVP MoSCoW requirements' percentages	95
Figure 33: percentage of requirements that are part of the MVP	95

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	8 of 101	
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	
	<b>Version:</b>	2.0	<b>Status:</b>	Final

## List of Acronyms

Abbreviation / acronym	Description
3D	3 Dimensions
3GPP	Third Generation Partnership Project
5G	5 <sup>th</sup> Generation
6DoF	6 Degrees of Freedom
ACL	Access Control List
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
ARM	Advanced RISC Machine
AWS	Amazon Web Services
BCH	Bitcoin Cash
CAD	Computer-aided design
CAGR	Compound Annual Growth Rate
CD	Continuous Deployment
CI	Continuous Integration
CNCF	Cloud Native Computing Foundation
COTS	Commercial off-the-shelf
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
Dx.y	Deliverable number y belonging to WP x
DApps	Distributed Applications
DC	Data center
Dev	Development
DevOps	Development and Operations
DevSecOps	Secure Development and Operations
DL	Deep Learning
DLT	Distributed Ledgers Technology
DSS	Decision Support System
EC	European Commission
ETSI	European Telecommunications Standards Institute
FaaS	Function as a Service
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HMD	Head Mounted Display
HPC	High Performance Computing
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
HW	Hardware
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol

ISO	International Standards Organization
IT	Information Technology
I/O	Input/Output
JDBC	Java DataBase Connectivity
JMX	Java Management Extensions
K8S	Kubernetes
KPI	Key Performance Indicator
LTE	Long Term Evolution
Lx	ISO Network Level x
MAV	Micro Aerial Vehicles
MEC	Multi-Access Edge Computing
ML	Machine Learning
MoSCoW	Must, Should, Could, Would
MQTT	Message Queuing Telemetry Transport
MR	Mixed Reality
MTO	Multi-tier Orchestrator
MVP	Minimal Viable Product
Mx	Month x
NFV	Network Function Virtualisation
NFVO	Network Function Virtualisation Orchestration
OpenCV	Open Computer Vision
Ops	Operations
OS	Operating System
OSM	Open Source Mano
OvS	Open vSwitch
PLMNID	Public Land Mobile Network Identifiers
POS	Point Of Service
RPi	Raspberry Pi
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RDBMS	Relational Database Management System
REST	Representational state transfer
ROM	Read-Only Memory
SDK	Software Development Kit
SLA	Service Level Agreement
SSL	Secure Socket Layer
SM	Slice Manager
SOE	Slice Orchestration Engine
SotA	State of the Art
SPEC	Standard Performance Evaluation Corporation
SUT	System Under Test
SVM	Support Vector Machine
SWOT	Strengths, Weaknesses, Opportunities, Threats
TDD	Test-Driver Development
TLS	Transport Layer Security

UC	Use Case
UI	User Interface
UK	United Kingdom
US	United States
USB	Universal Serial Bus
USD	United States Dollar
VCS	Version Control System
VIM	Virtual Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VPN	Virtual Private Network
VR	Virtual Reality
VRU	Vulnerable Road User
WG	Working Group
WML	Workload Model Language
WP	Work Package

## Executive Summary

---

This deliverable presents the initial Pledger requirements based on the use cases described in D2.1 at M6. An updated list of the requirements will be maintained along the project lifecycle and will be reported in an updated version of the D2.2 at M24.

The scope of the current deliverable comprehends:

- ▶ documentation of the Stakeholders value chain and the User Stories;
- ▶ documentation of the relevant technologies for the project;
- ▶ codification of the functional and non-functional requirements;
- ▶ the definition of the Minimum Viable Product (MVP) to be developed during the project based on a prioritised list of requirements.

The main goal of this report is to produce a prioritized coded list of requirements to provide the basis on top of which the Pledger architecture is going to be defined in D2.3.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	12 of 101				
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	<b>Version:</b>	2.0	<b>Status:</b>	Final

# 1 Introduction

## 1.1 Purpose of the document

Pledger’s main goal is to define a new architectural paradigm and a set of tools to host the next generation of edge computing and couple the advantages of computation on the edge with the reliability of cloud infrastructures.

The purpose of this document is to elaborate on the Pledger use cases analysis, identify the different stakeholders’ value chain, formalize and prioritize functional and non-functional requirements and provide inputs to the architecture definition activities.

## 1.2 Relation to other project activities

This document represents a codification of the initial requirements analysis shown in the D2.1 “Pledger’s Detailed Use cases” at M6 and provides a coherent and prioritised list of requirements to be used by the demonstrators. It also provides the functional definitions that paves the way for the consolidation of the Pledger architecture in the WP2 and guides the scenario for the demonstrators in the WP3, WP4 and WP5. An updated list of the requirements will be maintained along the project lifetime and will be reported in an updated version of the D2.2 at M24.

## 1.3 Approach used for the requirements elicitation and analysis

Figure 1 gives an overview of the approach used for the requirements elicitation and analysis.

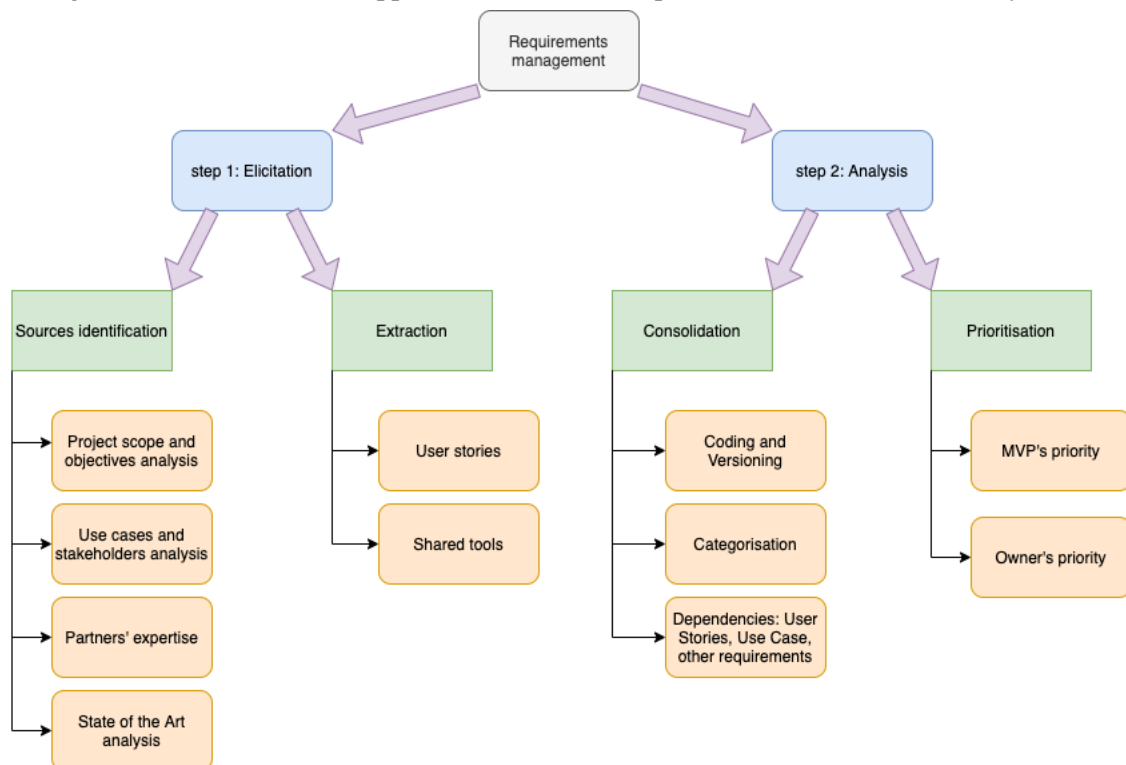


Figure 1: Requirement management approach

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	13 of 101	
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	
	<b>Version:</b>	2.0	<b>Status:</b>	Final

The first step is about the **elicitation**. It includes the **identification** of the sources and the **extraction** method. The identification is based on the analysis done on the project’s scope and objectives, the use cases and stakeholders described in D2.1, on the partners’ expertise on similar topics and from the analysis of the state-of-the-art relevant technologies. The extraction consists in the definition of the main user stories to identify the stakeholders, a desired feature and expected benefit, then on shared tools and iteration among the partners to define the description.

The second step is about **analysis**. It includes the **consolidation** of the requirements and the **prioritization**. The consolidation is achieved through the assignment of unique identifiers to the requirement, their categorization and the identification of the relation with the user stories, with the use cases and the dependencies with other requirements. The prioritization is achieved through the MVP process and the subsequent validation with the requirement’s author.

Figure 2 shows the requirements management lifecycle, which started at M3 with the analysis of the use cases and stakeholders description, consolidated in the D2.1 at M5, then with the iteration that produced the consolidated list of requirements for the MVP for the first year of activities that will be repeated for the second year and include the requirements fulfilment and produce an updated version of the D2.2.

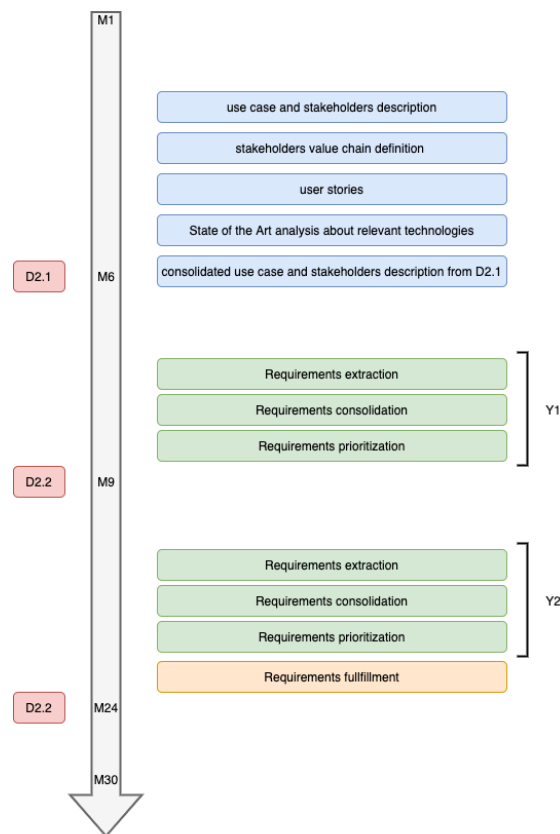


Figure 2: Requirements management lifecycle

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	14 of 101	
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	
	<b>Version:</b>	2.0	<b>Status:</b>	Final

## 1.4 Structure of the document

---

The stakeholders definition and overview of use cases are presented in Chapter 2 and represent the consolidation of the activities introduced in the D2.1. The partners background and State of the Art analysis are presented in Chapter 3 and are meant to elicit possible, additional requirements for the Pledger system and identify benefits and risks about through a SWOT analysis. The Requirement elicitation and consolidation activities are presented in Chapter 4 along with some key diagrams that highlight the distribution in different categories. The requirement prioritization is presented in Chapter 5 using the Minimal Viable Product (MVP) approach. The Annex contains the details about the spreadsheets used for the Pledger MVP.

## 1.5 Terminology adopted in this document

---

This subsection contains the list of terms used in this document in order to clarify its meaning to the readers

- ▶ **Asset.** Existing software/hardware artefact that is used in the Pledger system
- ▶ **Minimal Viable Product.** A minimal and coherent set of features to be implemented in an iterative development process.
- ▶ **Requirement.** A feature wanted or needed for the project.
- ▶ **Stakeholder.** Any individual or organization that is actively involved in the project, or whose interests may be affected as a result of project execution or successful project completion.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	15 of 101				
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	<b>Version:</b>	2.0	<b>Status:</b>	Final

## 2 Use Cases overview, Stakeholders & User Stories

The objective of this section is to present a bird-eye overview of the use cases described in the D2.1, describe the stakeholders value chain and the user stories to facilitate the requirements elicitation phase in the last sub-section.

### 2.1 Use cases overview

This section provides an overview of the three use cases that will be deployed on the Pledger system and the stakeholders involved.

#### 2.1.1 Mixed reality applications on the edge

The goal of this UC is to enhance the capabilities of Augmented Reality (AR), Mixed Reality (MR) and Virtual Reality (VR) solutions by coupling them with edge computing technologies and the corresponding load allocation and optimization tools, in order to provide high-level services in industrial environments. In particular, the UC will focus on three case studies integrating machine data into MR interfaces and optimizing them via edge computing technology:

- ▶ Collaborating in Fast Prototyping
- ▶ Assisting in Manufacturing and Service & Utility procedures
- ▶ Enhancing Training & Interaction

Holo-Light's Use Case is in general customer oriented. It will support engineers in the manufacturing hall. Therefore, departments like the Engineering Department and IT Department need to be in the loop.

Minimum requirements to deploy this use case are the existence of 3D CAD data, on-site availability of Wi-Fi and a 6DoF capable AR/VR Head Mounted Display (HMD), i.e. HoloLens.

Engineers will be able to visualize and work with industrial 3D CAD models in combined with reality (factory hall) in AR and to collaborate with colleagues to lower costs for product design and increase the quality of products.

#### 2.1.2 Edge Infrastructure for enhancing safety of vulnerable road users

Ensuring the safety of vulnerable road users (VRUs) in cities is a challenging objective that is tackled in this use case. By deploying Pledger on top of citywide infrastructure, a risk detection and notification service will be provided to enhance the security of VRUs. Using tracking mechanisms for VRUs and public transport (tram), potential risks will be identified and relayed to users (pedestrians) via the infrastructure, using both dedicated radio resources and computing resources that can be instantiated on-demand. The key stakeholders participating in this use case are the infrastructure providers, the platform providers and the end users, the latter being the ones whose security is to be enhanced.

The technologies used in this use case involve x86 CPU architectures (compute nodes), single board computers (x86/ARM) and common radio technologies, such as IEEE 802.11. The operating systems will mainly be based on Ubuntu across all nodes (computing, radio, user equipment). If personal devices (e.g. smartphones) are used, then the support of Android OS and an Android application may also be required for the use case. The detection of VRUs and public transport will be possible by using technologies like Galileo (for positioning) or sensors (such as radar).

The necessary infrastructure to host the hardware elements in the on-street deployment (location to be determined) will require public construction. The radio and sensor elements will be designed and chosen to integrate smoothly with the city landscape, while being able to satisfy the use case requirements.

The data generated by the end users (VRUs) will be transmitted in a secure manner to the applications running on top of the Pledger-enabled Barcelona infrastructure. It will be processed by a virtualized

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	16 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

service running in the computing nodes in order to detect situations of risk. Apart from processing the data, the applications will also check whether the data is trustable, and safety mechanisms will assure that it is not accessible by unauthorized third parties. If a risk is detected, the infrastructure will be used to relay a warning to VRUs, so that they are made aware of the risk and can avoid it.

### 2.1.3 Manufacturing the data mining on edge

As Fill edge computing devices face strict customer requirements (use low energy, harsh environments, low costs), Fill's Industrial IoT-Platform ("Cybernetics") is forced to outsource computationally intensive operations to a cloud service. Data transfer and transfer of computational power enables the generation of richer datasets and the computation of more complex machine learning models to improve predictions and analytics. These analyses by the data analysts enable the engineering experts to better evaluate the machine mechanics and to improve the engineering process. Furthermore, they provide valuable insights into the machine for the customer. These analyses are developed in Python3.

The data is collected and processed in dockerized applications on the edge device, which is based on an X86 architecture and running on Linux. Basic analysis is also performed there, e.g. evaluation of the cycletime and quality of a component or the average power of the machine. These analysis scripts are deployed on edge as Docker containers and the derived indicators can be visualized on a dashboard. For more complex analyses and the use of analysis services towards predictive maintenance, data must be transferred to the cloud, where the customer wants to decide which data streams to release for whom.

Customers' machine data transfers knowledge about the underlying processes and a special focus on security is required to ensure a secure data transmission and to prevent unauthorized access by third parties. The UC will be deployed in Fill's own premises, in the research & development area of the FILL Future Zone, where Wi-Fi is deployed for wireless data transfer within the building.

## 2.2 Stakeholders value chain

---

A key activity for the design of the Pledger system is the elicitation of requirements from the stakeholders that comprise its value chain.

Main stakeholders have been identified from D2.1 and from the high-level reference architecture proposed by Multi-access Edge Computing (MEC) by ETSI<sup>1</sup> about edge computing as will be described in Section 3 later on.

ETSI MEC initiative aims to unite the telco and IT-cloud worlds, identifies the main stakeholders of edge architecture and helps validate those identified in Pledger system using D2.1.

By comparing ETSI MEC stakeholders with those described in D2.1, it is observed that the most relevant beneficiaries of the Pledger system are the edge-computing providers (infrastructure providers), the edge computing adopters (both service providers and end users), the system integrators and the open-source community as Pledger aims to be mostly open-source.

The Pledger consortium includes all the relevant stakeholders of the Pledger value chain, in particular: infrastructure providers (e.g. ENG), system integrators (e.g. ATOS), service providers (e.g. FILL, HOLO).

Figure 3 gives an overview of the stakeholders' value chain:

- ▶ infrastructure providers
- ▶ system integrators
- ▶ service providers
- ▶ end users

<sup>1</sup> <https://www.etsi.org/technologies/multi-access-edge-computing/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	17 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final



Figure 3: Pledger stakeholders value chain

### 2.2.1 Infrastructure providers

The main role of the infrastructure providers in Pledger is to lease their infrastructure. They will also leverage the monitoring and benchmarking capabilities of the platform to verify the actual resource usage and dynamically adapt and reconfigure the network services across the infrastructure to guarantee the required QoE and reduce the operational costs.

### 2.2.2 System integrators

The role of system integrators in Pledger is related to the available API to enable the development of added-value services over those provided by the Pledger system. For example, it will be possible to rely on benchmarking to achieve resource efficient deployment of the services while achieving the same Quality of Experience (QoE), as well as leverage on the other main components such as Blockchain, SLA, AR and so on.

For example, the translator or bridge responsible for matching SLA terms to blockchain smart contracts in the context of a fast and efficiently integrated system generated by dissimilar technologies will be available through an API used from the “Microtransactions Framework for the Edge” asset.

Similarly, the StreamHandler big data platform API for data management will simplify the integration of multiple data providers and consumers adding features like encryption, authorization, authentication and fault-tolerance.

### 2.2.3 Service providers

The service providers in Pledger are the responsible for the deployment and operation of their services over the Pledger system. They leverage on the Pledger features to continuously monitor the QoE and resource allocation, so to dynamically choose the best placement of the functionalities with respect to cost efficiency and the SLA coupled to blockchain smart contracts.

### 2.2.4 End users

The end users in Pledger are the consumers of the main use cases identified by the D2.1, namely:

- ▶ UC1: Engineers as a general term. More specific departments and fields are Design and Planning, Production Support, Construction, Quality Engineering, Industrial Training, Industrial Maintenance, Simulation Engineers
- ▶ UC2: Pedestrians, bikes/electric scooters and other mobility users will benefit of increased safety features. City mobility departments will easily manage different VRU flows in alignment to the local regulations.
- ▶ UC3: Data Analysts, that benefit from richer datasets and more computational power to train machine learning models for predictive maintenance; Engineering experts, that benefit from these models to better estimate the durability of the machine for improving the engineering process; then Customers, that receive progress information about machine performance and the running process and therefore will benefit from better services and predictive maintenance to reduce downtimes and costs. They will rely on the secure data transmission and management across cloud and edge and gain their trust and involvement over time.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	18 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

## 2.3 User Stories

The objective of this subsection is to describe the user stories extracted from D2.1 along with an early list of requirements that will be formalized in Section 4.

The user stories are presented in the Table 1 using the “as a..”, “I want to..”, “so that..” Agile format<sup>2</sup> that should help identify the user’s roles, the desired features and the expected value to be achieved, along with the use case it relates to.

Table 1: User stories

ID	AS A(N)	I WANT TO	SO THAT	USE CASE
US.01	End user (Engineer)	upload 3D files and view them live in the room and interact with them	I can verify if the newly designed components, i.e. cylinder blocks, fits the rest of the model	UC1
US.02	End user (Engineer)	load my files through a remote rendering platform	I can overcome the device’s limitations.	UC1
US.03	End user (Engineer)	use world Anchors or Image Trackers as references point	I can interact in the future from the same place where I left from before.	UC1
US.04	End user (Engineer)	share the new design with my colleagues	I can collaborate with other engineers in real time.	UC1
US.05	End user (Engineer)	view my colleague as an Avatar during sessions	I can interact with other engineers more intuitively.	UC1
US.06	End user (Engineer)	see collisions highlighted in, i.e. red	I can be aware where the areas of improvement are.	UC1
US.07	End user (Engineer)	be able to save all my changes to the CAD repository	I have no development lost and the product development can progress faster.	UC1
US.08	End User (VRU)	register to the app/system that implements the safety features	I can benefit from its features and avoid risky situations.	UC2
US.09	End User (VRU)	be warned by an audiovisual signal in case I face a risky situation	I can detect the risky situation and I can avoid it.	UC2
US.10	End User (VRU)	share my position with the system in a trustable/secure way	I can preserve my privacy and security.	UC2
US.11	Infrastructure Provider	to be able to make computing and radio resources of the city-wide testbed available	I can enable the service provider to deploy services on top.	UC2

<sup>2</sup> <https://www.agilealliance.org/glossary/user-story-template/>

ID	AS A(N)	I WANT TO	SO THAT	USE CASE
US.12	Service Provider	to be able to dedicate a part of my resources in terms of computing and radio resources to a specific tenant	I can be sure that only the actually required resources are used by the specific tenant and that other services can run in parallel.	UC2
US.13	Service Provider	to have isolation between different services running in the infrastructure	I have privacy and security guaranteed.	UC2
US.14	Infrastructure Provider / Service Provider	to have dashboards/GUIs to manage the resources/services assigned to the different tenants	I can have both infrastructure owner and service provider easily manage the allocated resources.	UC2
US.15	Service Provider	the use case application needs to be able to process location and sensor inputs	I can determine risky situations.	UC2
US.16	Service Provider	receive a warning for a VRU issued in time	I have enough time to react in case of any risky situation.	UC2
US.17	Service Provider	receive/delete information about urban transport service location	I can determine, together with other inputs, a potential risky situation.	UC2
US.18	System integrator	use an easy verification tool	I can estimate whether the provided infrastructure is sufficient.	UC3
US.19	End user (Customer)	ensure that data transmission and release is secure	I have data that cannot be accessible by unauthorized third parties.	UC3
US.20	End user (Customer)	use an intuitive data privacy managing tool	I have an overview of which data is released to which service provider.	UC3
US.21	End user (Data Analyst)	create big datasets from different sources on the cloud	I can benefit from richer and diverse datasets for model training and prediction.	UC3
US.22	End user (Data Analyst)	process big datasets on the cloud	I can skip limitations in computation and storage of an edge devices.	UC3
US.23	End user (Data Analyst)	monitor the machine learning performance	I can take the right steps when performance drops.	UC3
US.24	Service Provider	track number of released streams/used applications	I can build a business model on e.g. pay-per-use approach.	UC3
US.25	End user (Data Analyst)	ensure that transferred data points are encrypted	I can prevent attacks against the machine learning model.	UC3

ID	AS A(N)	I WANT TO	SO THAT	USE CASE
US.26	Service Provider	easily update any developed application	I can have new versions deployed easily.	UC3
US.27	Service Provider	show results to a customer on a dashboard	I can directly transfer the outputs.	UC3
US.28	Service provider	deploy services on edge and cloud for different users	I can optimally use the resources available.	ALL
US.29	Service provider	create a service that can run on the cloud or on the edge without any change in the code	I can write the code and package the service just once.	ALL
US.30	Service provider	deploy a service on the edge and have it running in a few seconds	I can reduce the downtimes.	ALL
US.31	Service provider	reuse existing services from big open-source communities	I can improve the stability of the services deployed.	ALL
US.32	Service provider	deploy ML services on the edge using low cost HW	I can reduce the infrastructure costs and keep my models private.	ALL
US.33	Service provider	deploy a service on the edge and be sure it is resilient in case of major issues	I can reduce maintenance and operational costs.	ALL
US.34	Service Provider	configure radio resources to offer 80211p for vehicular connectivity	I can have my users with bicycles/electric scooters have the means to connect to the infrastructure and share their position.	UC2
US.35	Service provider	couple SLA terms to blockchain smart contracts	I can secure my SLAs privacy and immutability.	ALL
US.36	Service provider	use permissioned blockchain properties	I can assure privacy and security of my data.	ALL
US.37	Systems Integrator	handle data securely	I can guarantee data security without additional effort.	ALL
US.38	Systems Integrator	handle data effectively	I can assure timely delivery of data-to-data consumers.	ALL
US.39	Systems Integrator	ensure fault-tolerance in data management	I can protect data consumers from any critical data loss.	ALL
US.40	System integrator	authenticate and get access to the API	I can use the all services that I have been authorized to.	ALL

ID	AS A(N)	I WANT TO	SO THAT	USE CASE
US.41	Service provider	get performance metrics for my application on the different available Cloud and Edge infrastructures	I can select the one with the best performance/cost trade-off.	ALL
US.42	Service provider	run my own benchmarking tests, based on my application	I can get more accurate performance prediction results.	ALL
US.43	Service provider	be able to compare and analyse performance metrics for different time ranges	I can evaluate the stability of performance of the resources.	ALL
US.44	Infrastructure Provider	support benchmarking of my infrastructure resources without any change to my infrastructure	I can be compliant with the Pledger solution without additional effort.	ALL
US.45	Service provider	configure execution of private benchmarking tests	I can have available more data than the public benchmarking results.	ALL
US.46	System integrator	query benchmarking results using a standard API	I can process benchmarking data with existing tools/libraries/algorithms.	ALL
US.47	Infrastructure Provider	execute efficiently benchmarking tests on my infrastructure allocating the minimum necessary resources	I can optimize the usage of resources.	ALL
US.48	Service provider	use a dashboard to get a clear view of the infrastructure resource allocation and of the service deployment status	I can easily detect inefficiencies.	ALL
US.49	Service provider	use a dashboard to get a clear view of the critical issues about services and infrastructure	I can quickly spot the issues and possibly take an action to mitigate them.	ALL
US.50	Service provider	use a dashboard with a prioritized list of the options I have to change configuration and deployment of the services	I can choose the option that better fulfils SLA/QoS constraints.	ALL
US.51	Service provider	have a measurable estimation of the benefits I have if I choose a specific deployment option	I can simply compare the different options and take a decision.	ALL

## 3 Relevant Technologies for the Project

---

This section contains the state of the art of the relevant technologies for the Pledger system, with focus on the requirements and the possible risks and benefits, and an initial report about the possible integration activities in Pledger.

The technologies evaluated belong to the following topics:

- ▶ Slice Orchestration Engine for Cloud, Edge and Radio Infrastructures
- ▶ Containerisation on the edge
- ▶ Big Data
- ▶ Continuous Integration/Continuous Delivery
- ▶ Blockchain
- ▶ Benchmarking
- ▶ Monitoring applications and infrastructures in Fog and Edge
- ▶ Cloud and Edge SLA monitoring and evaluation
- ▶ AR Streaming on the Edge

Each topic is presented with a brief state of the art, the scenario explored in Pledger, the main advantages brought to the platform and the technical feasibility and risks of the integration in Pledger throughout a short SWOT analysis.

Eventually, this section aims at identifying the possible extensions to the use cases scenarios described in D2.1, elucidate new requirements and contribute to the Pledger architecture.

### 3.1 Slice Orchestration Engine for Cloud, Edge and Radio Infrastructures

---

Cornerstones of modern and flexible telecommunication architectures are orchestration and software elements that can effectively manage cloud, edge and radio infrastructures. Architectures that rely on these components not only provide cloud-like features close to the subscribers at the edge of the network, but it now includes control over a variety of wireless communication technologies. As a result, services can be deployed on-demand and by introducing concepts like slicing, the reutilization of the same infrastructure by multiple service providers is possible. This section provides an overview of current solutions that enables the development of such architectures.

- ▶ *Open Source MANO (OSM)*: Hosted at the ETSI repositories and under the umbrella of the Linux Foundation projects, this ETSI-NFV compliant orchestrator provides a production-quality MANO stack for commercial NFV deployments. Recent releases of this framework provides plugins not only to orchestrate resources in public and private clouds, but it also brings cloud-native applications to NFV deployments. As a result, OSM enables the onboarding of 20,000 pre-existing production-ready Kubernetes applications while preserving all the required advanced networking required to build a complex end to end services.
- ▶ *Slice Orchestration Engine (SOE)*: Developed in the context of the H2020 5GCity Project<sup>3</sup>, this framework provides connectors that dynamically enable the allocation of resources for services (i.e. connectivity and application VMs) in the cloud and edge infrastructures based on OpenStack. To that end, this engine leverages the current capabilities provided by OSM to allow also the pre-provisioning of resources in different types of infrastructure (e.g. compute, networking, and radio chunks). These operations are exposed via a RESTful API service that can be programmatically exploited by third-

---

<sup>3</sup> <https://www.5gcity.eu/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	23 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

parties to ensure the infrastructure couples with SLAs defined by the adopters (e.g. application developers, etc.).

The edge computing is standardized as Multi-access Edge Computing (MEC) by ETSI. The main feature introduced by MEC is the possibility of hosting services in a distributed manner close to the radio access network (RAN) to enable faster processing of data. Bringing services closer to the user, via e.g. video transcoding or edge caching reduces the latency of the services and also saves bandwidth in the backhaul that would be needed if the request is processed somewhere else. The computing resources provided by MEC are normally much more restricted than computing resources hosted in traditional data centres (DCs), given the restrictions imposed by the physical locations where MEC resources can be hosted: MEC compute nodes can be integrated anywhere from small DCs with a small amount of space in a building close to the RAN equipment, over on-street cabinets at street level, up to being collocated directly with radio equipment, e.g. in small boxes mounted on street furniture.

While the physical setup can vary, the logical architecture and the relationships between the RAN and compute components is well defined by ETSI<sup>4</sup>, specifying the requirements and how the different features of the architecture have to be implemented in order to be compliant.

The ETSI MEC standard serves as the main reference for the relationship of compute and RAN resources used in PLEDGER.

For the management and the integration of the RAN with the edge and cloud computes, a custom RAN controller solution will be used in PLEDGER. Similar to other solutions developed in European projects, like FlexRAN (Xenofon Foukas, 2016) and EmPOWER (E. Coronado, 2019), the RACOON RAN controller developed by i2CAT is capable to configure and manage radio devices (Wi-Fi, LTE). More importantly, it allows integrating RAN resources as part of an end-to-end slice with the SOE, which will be crucial for the use cases that combine both the radio and compute worlds. RACOON applies a YANG model management, using the NETCONF protocol (Enns, 2011) to remotely configure the nodes and deploy the required RAN slices. In the case of the LTE Small Cells, it is able to dynamically instantiate up to five different slices per Small Cell using custom Public Land Mobile Network Identifiers (PLMN-IDs) and their associated vEPCs. Regarding data paths, they are managed using OpenDayLight<sup>5</sup> open platform, using Layer-2 VLANs controlled by Open vSwitch<sup>6</sup> (OvS) to differentiate the traffic of the different slices through the network.

### 3.1.1 Architecture and Way of Operation

This section presents the high-level architecture, the main supported operations, and some implementation notes about the SOE solution for performing resource slicing and orchestration on cloud, edge and radio infrastructures.

One of the aspects that first needs to be identified when considering slicing and orchestration of cloud, edge and radio resources are the components which enable such operations. Figure 4 shows the positioning and corresponding interfaces of SOE along with the RAN Controller inside such a system, while it also illustrates the main internal components that a basic SOE solution would require.

As shown in Figure 4, the two main components of the SOE solution are the Slice Manager (SM) (H. Khalili, 2019) and the Multi-Tier Orchestrator (MTO) (M. P. Mena, 2020). On one hand, the SM enables the virtualized and non-virtualized network elements and functions to be logically segmented, configured and reused (isolated one from another) in order to meet various demands. Each slice has its own specific compute and network resource chunks that suit the SLAs of each network services offered by it.

<sup>4</sup> <https://www.etsi.org/committee/1425-mec/>

<sup>5</sup> <https://www.opendaylight.org/>

<sup>6</sup> <https://www.openvswitch.org/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	24 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

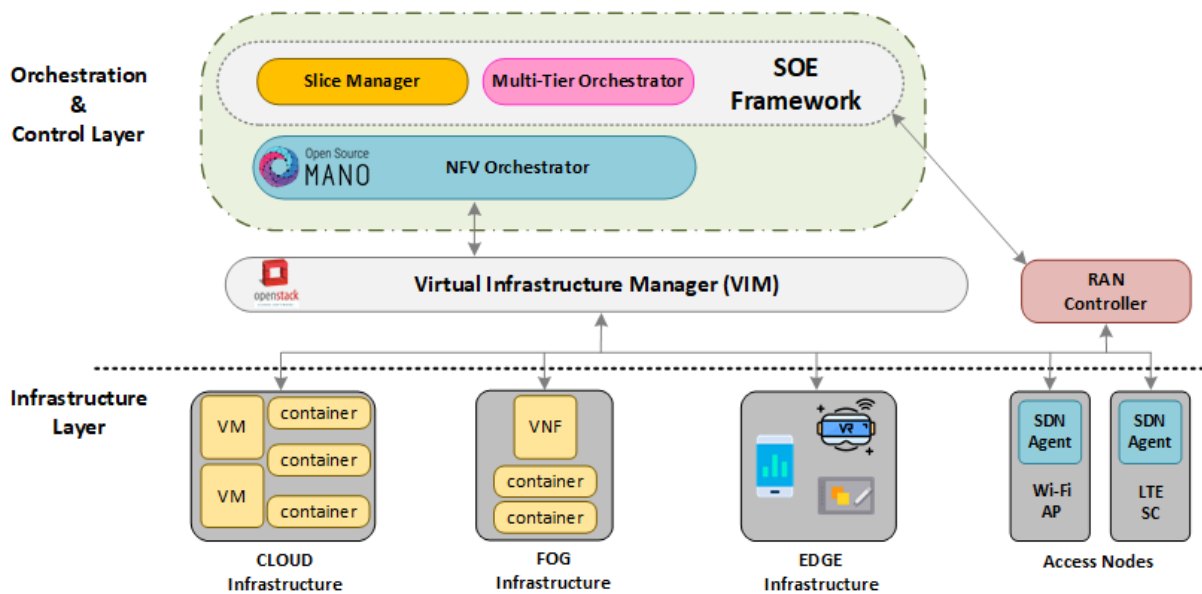


Figure 4: Functional architecture of SOE and RAN Controller solutions

On the other hand, the MTO lies between the SM and the NFV Orchestrator. It provides a single abstraction point between the SM and any other domain (e.g. MEC and cloud-native orchestrators) that needs to be integrated into the system. Moreover, it is responsible for the triggering of high-level actions such as onboarding, instantiation, and monitoring of (network) services and platform rules and configurations across the different orchestrators.

As a complement, in Figure 5 the overall system architecture of the SOE and RAN Controller solutions is revealed. Both, the SM and the MTO have a RESTful API, which is used to trigger the deployment of applications and services along with the corresponding API invocation needed on the NFV Orchestrator.

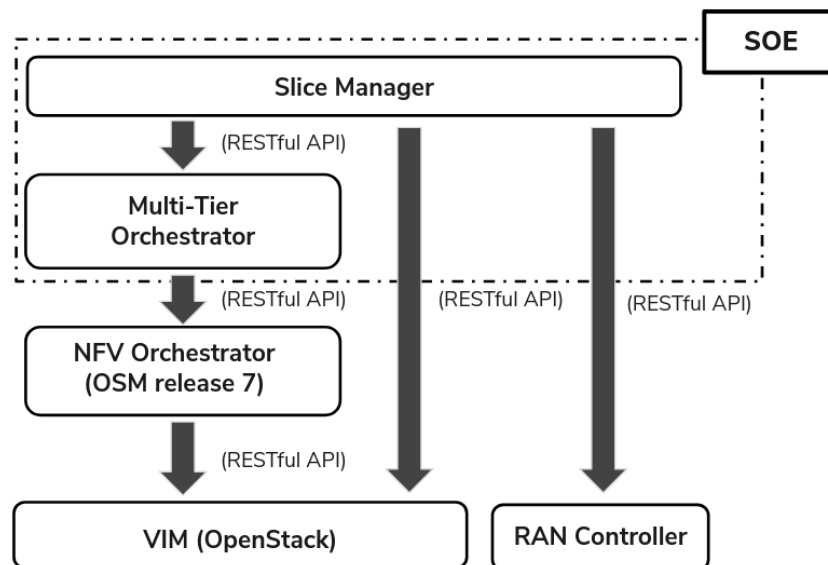


Figure 5: System architecture of SOE and RAN Controller solutions

### 3.1.2 Initial feasibility study

As an initial feasibility study, here we describe the used technologies and deployment techniques used to implement all the components presented in Figure 2. In this context, for the Virtualized Infrastructure Manager (VIM), we have used the latest stable release of OpenStack (i.e. Train<sup>7</sup>). The reason for selecting OpenStack is because this is one of the most widely deployed open-source cloud infrastructure software in the industry nowadays. To orchestrate the lifecycle of application and services we used as the NFV Orchestrator the OSM release 7, recently delivered under the umbrella of the Linux Foundation<sup>8</sup>.

After having prepared these components, we have proceeded to deploy the SOE Framework and the RAN Controller on the cloud following an automated approach. To that end, deployment tasks were divided in day 0 and day 1 configurations using the tools Terraform<sup>9</sup> and Ansible<sup>10</sup> respectively.

- ▶ Terraform, which is a cloud-agnostic management tool, was used for automating the creation of the VMs where the two components are going to be deployed. This approach has proven to be a reliable and flexible way to define the computing and networking requirements of these components as a blueprint that can be deployed at any moment.
- ▶ Ansible, which is a very efficient tool to configure, deploy, and orchestrate code, was used once the VMs are instantiated on the cloud infrastructure. This tool was used not only to effectively deploy the code of these components but also to ensure production-grade security configurations on the host where these components are deployed automatically.

This automated approach has been extensively tested in diverse and complex environments like the supported by different European projects like H2020 5GCity<sup>11</sup>, and the H2020 5GCroCo (on the MEC Pilot)<sup>12</sup>. Moreover, in scenarios where multiple installations (i.e. development environments) are required this approach saves efforts, time as well as significantly reduces the probability of error-prone deployments.

### 3.1.3 Expected benefits for the Pledger system

From the concepts listed in this section, in Pledger the following features are considered for the operation of the platform:

- ▶ The platform will have different compute tiers (at least two) including edge and cloud/main DC resources.
- ▶ The system will have an NFV orchestration (NFVO) system that allows to deploy and manage services during runtime.
- ▶ The system will have a RAN controller component that will allow to configure and manage radio services.
- ▶ The platform defines APIs for the different components (RAN, RAN controller, OSM, etc.) to talk to each other and defines the features necessary to combine the RAN and compute components.

<sup>7</sup> <https://www.openstack.org/software/train/>

<sup>8</sup> <https://osm.etsi.org/news-events/news/61-etsi-osm-unveils-release-seven>

<sup>9</sup> <https://www.terraform.io/>

<sup>10</sup> <https://www.ansible.com/>

<sup>11</sup> <https://www.5gcity.eu/>

<sup>12</sup> <https://www.5gcroco.eu/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	26 of 101	
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	
	<b>Version:</b>	2.0	<b>Status:</b>	Final

### 3.1.4 SWOT analysis

The main advantages and drawbacks, as well as opportunities and possible threats of the slice Orchestration Framework for Cloud, Edge and Radio Infrastructures known at the time of writing, are presented in Table 2.

Table 2: Slice Orchestration Framework for Cloud, Edge and Radio Infrastructures SWOT

Strengths	
<ul style="list-style-type: none"> <li>▶ end-to-end slicing enables to establish services all the way from the user access (at radio level) up to the application running in virtualized environments</li> <li>▶ easy configuration of services via network service descriptors and GUIs</li> <li>▶ support of the creation of customizable and user-specific slices with flexible and dynamically deployable resources, based on explicit user requests, existing slice templates, or implicit info about the planned services</li> <li>▶ unlocks dynamic provisioning and instantiation of end-to-end network services</li> <li>▶ support for multi-tenancy over different underlying technologies and resources used to compose the slices</li> <li>▶ validated and tested in city-wide testbeds, unlike other solutions that have been tested in virtual environments and/or lab testbeds</li> </ul>	
Weaknesses	
<ul style="list-style-type: none"> <li>▶ not yet production ready (though extensively tested)</li> <li>▶ requires integration work to support additional radio technologies; currently supported are plain Wi-Fi and a small cell (LTE) solution that is to be extended</li> <li>▶ at the current state not fully ETSI-compliant</li> </ul>	
Opportunities	
<ul style="list-style-type: none"> <li>▶ extensions, such as the support of containers and new radio technologies developed in the Pledger project will allow addressing even more use cases</li> </ul>	
Threats	
<ul style="list-style-type: none"> <li>▶ additional features have to be developed and integrated in order for UC2 to be supported</li> </ul>	

## 3.2 Containerisation on the edge

---

The goal of this section is to highlight the relevant requirements, with respect to the Pledger system, of some of the available open-source edge solutions, in particular, those supporting containers (aka “cloud-native”<sup>13</sup>) that, as it will be shown later on, contribute to improving the resiliency that is one of the key requirements nowadays requested on the edge platforms as well as in the cloud.

In particular, the following subsections will describe:

- ▶ edge computing requirements,
- ▶ containers on the edge requirements and how they fit into edge configurations,
- ▶ available open-source container-based edge platforms and challenges,
- ▶ expected benefits for the Pledger system,
- ▶ initial feasibility study.

### 3.2.1 Edge computing requirements

Cloud computing paradigm has brought major benefits to business efficiency (Weisong Shi, 2006) since its adoption around two decades ago. Some main issues have been raised (Shi, 2016) though, such as the loss of privacy, the complete delegation of the control from the users to the cloud and the missed chance of exploiting the available computational power of end-user devices.

In recent years, the interest for the edge computing paradigm, that brings the computation closer to the location needed, has increased. The so-called “edge-centric computing” (W. Shi, 2016) paradigm aims at improving trust, bringing the control back to the user, supporting distributed control to enable local autonomous decision-taking, saving bandwidth and keeping latency low.

Bandwidth and latency are the most intuitive topics affected by edge computing. A report from Canonical<sup>14</sup> summarizes their impact on control (Figure 6) and on the number of managed devices (Figure 7).

---

<sup>13</sup> <https://www.cncf.io/>

<sup>14</sup> [https://docs.google.com/presentation/d/1a\\_m3CC2L1Bm13VyN98NhWO5vkFN-lct2FmbsZhJ3Cr0/edit-slide=id.g5d07f1af87\\_0\\_90](https://docs.google.com/presentation/d/1a_m3CC2L1Bm13VyN98NhWO5vkFN-lct2FmbsZhJ3Cr0/edit-slide=id.g5d07f1af87_0_90)

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	28 of 101				
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	<b>Version:</b>	2.0	<b>Status:</b>	Final

## Edge Computing Architectures - Pyramids

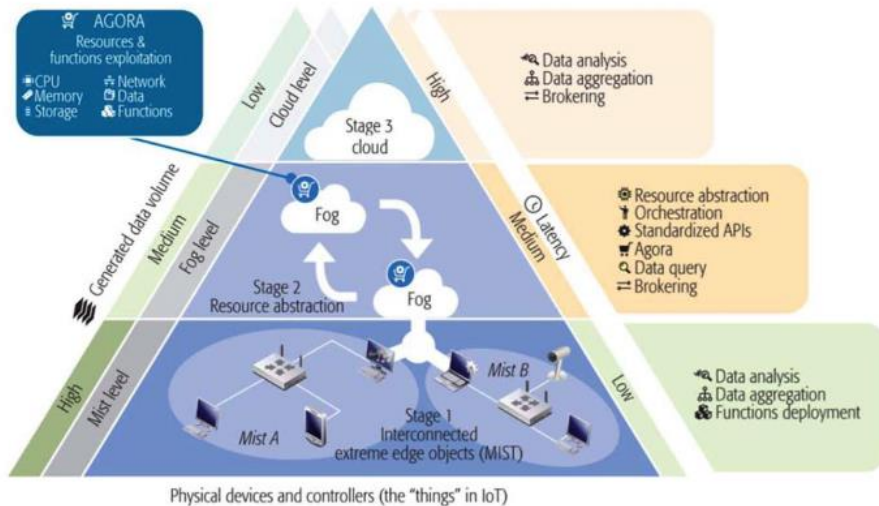


Figure 6: Edge computing architectures (source: Canonical)

### Themes

- 01 Billions of Data Generating Devices
- 02 Extreme **edge nodes** for **low latency**
- 03 Millions of **edge clouds** for **medium latency**
- 04 Thousands of **data centers** for **high latency**

Figure 7: edge nodes vs latency (source: Canonical)

To tackle less intuitive requirements, the recent report<sup>15</sup> from Gartner highlights the main benefits categories when working on the edge, reported below:

- ▶ data volume and bandwidth consumption
- ▶ autonomy
- ▶ privacy and security
- ▶ local interactivity
- ▶ latency

<sup>15</sup> <https://emtemp.gcom.cloud/ngw/globalassets/en/doc/documents/3889058-the-edge-completes-the-cloud-a-gartner-trend-insight-report.pdf>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	29 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

Another report from Canonical<sup>16</sup> identifies the same benefits from the industry perspective, in particular those about autonomy becoming crucial for some specific use-cases (e.g. autonomous driving).

Another relevant use-case<sup>17</sup> for the edge is 5G, whose network needs a highly distributed and decentralized system. A quick view at the relation between edge and 5G contributes to identifying additional requirements that are relevant for the edge, below summarized as:

- ▶ standard infrastructure/vendor agnostic
- ▶ open interfaces/APIs
- ▶ low-cost COTS hardware
- ▶ resiliency (“single truck-roll”)

Also, on the 5G orchestration front, the transition to cloud-native is gaining huge interest from the open-source community. So, while from the industry<sup>18</sup> perspective, new flexible solutions have been proposed to manage edge through bare metal, plain virtual machines and containers, the recent support by ETSI OSM<sup>19</sup> orchestrator to Kubernetes aim to include around twenty thousands of pre-existing production-ready containerized applications within edge scenarios.

### 3.2.2 Containers on the edge requirements

Containers have brought a huge performance boost into virtualization, with faster instantiation time, around 1/1000 of plain VMs<sup>20</sup>. As a matter of fact, the DevOps community is witnessing increased requests from the industry about containers and edge solutions, with Kubernetes leading by far over alternative solutions for the reasons listed below, mainly thanks to the adoption of reliable tools and services to encompass developers, operators and customers’<sup>21</sup> needs:

- ▶ more stable than its competitors,
- ▶ not tied to one architecture or organization,
- ▶ well documented,
- ▶ a declarative application programming interface (API),
- ▶ large community

Referring again to 5G as one key use-case for the edge, Kubernetes based solutions have been proposed by Nokia and by Ericsson<sup>22</sup> and reflect the actual trend, promoted by 3GPP consortium, to move standardized 5G Core functions to cloud-native and container starting from 2019. Even the very latest trends from the academics about cloud-continuum (Bittencourt, 2018) include containers on the edge using the Function as a Service approach (Baresi, 2019), so this should be enough to justify the adoption of a cloud-native approach in Pledger.

Canonical started identifying the relevant requirements on the containers on the edge approach in a recent report<sup>23</sup>, stressing out those about:

<sup>16</sup> [https://docs.google.com/presentation/d/1ox-sAfej3MEQa8HFqExi58kqp8nbf2BdfrORnVg5M0/edit#slide=id.g5c6e5ef472\\_0\\_559](https://docs.google.com/presentation/d/1ox-sAfej3MEQa8HFqExi58kqp8nbf2BdfrORnVg5M0/edit#slide=id.g5c6e5ef472_0_559)

<sup>17</sup> <https://transition.fcc.gov/bureaus/oet/tac/tacdocs/reports/2018/5G-Edge-Computing-Whitepaper-v6-Final.pdf>

<sup>18</sup> [https://docs.google.com/presentation/d/1vSeztkHIUSaJx9pwGearrBD1d4Cxq0z-U\\_aSnMUlds/edit#slide=id.g5d1dfec3b2\\_0\\_575](https://docs.google.com/presentation/d/1vSeztkHIUSaJx9pwGearrBD1d4Cxq0z-U_aSnMUlds/edit#slide=id.g5d1dfec3b2_0_575)

<sup>19</sup> <https://osm.etsi.org/news-events/news/61-etsi-osm-unveils-release-seven>

<sup>20</sup> <https://www.backblaze.com/blog/vm-vs-containers/>

<sup>21</sup> <https://platform9.com/blog/six-kubernetes-takeaways-for-it-ops-teams-from-the-2019-gartner-infrastructure-operations-cloud-strategies-conference/>

<sup>22</sup> <https://platform9.com/blog/six-kubernetes-takeaways-for-it-ops-teams-from-the-2019-gartner-infrastructure-operations-cloud-strategies-conference/>

<sup>23</sup> [https://docs.google.com/presentation/d/1vSeztkHIUSaJx9pwGearrBD1d4Cxq0z-U\\_aSnMUlds/edit#slide=id.g5ed70af0b5\\_3\\_9](https://docs.google.com/presentation/d/1vSeztkHIUSaJx9pwGearrBD1d4Cxq0z-U_aSnMUlds/edit#slide=id.g5ed70af0b5_3_9)

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	30 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

- ▶ flexibility
- ▶ repeatability
- ▶ economics

The interest for the edge by the Kubernetes (K8S) community brought to the creation of the K8S IoT Edge Working Group<sup>24</sup>, that aims to identify emerging use cases and requirements<sup>25</sup> about IoT and edge architecture.

A first report shows how the Kubernetes requirements between cloud and edge have some similarities and differences as well, with the need to support specific network protocols, to add the ability to local data processing and support privacy or low-latency scenarios, introduce resiliency on the edge and autonomy to overcome<sup>26</sup> unattended and disconnected operations.

As a result, a first platform (KubeEdge) was presented at the KubeCon 2019 by one of the Working Group (WG) members which summarized the benefits of adopting Kubernetes on the edge, reported below as high-level requirements:

- ▶ Process a larger amount of data to reduce bandwidth consumption, increase responsiveness, decrease costs, improve data privacy
- ▶ Facilitate developers integration using regular HTTP or MQTT based containerised applications to be deployed on the cloud or on the edge seamlessly
- ▶ Facilitate the operation, providing tools to orchestrate, manage the devices and monitor the application like the traditional Kubernetes cluster in the cloud
- ▶ Leverage on the numerous already existing containerised applications available

### 3.2.3 Open source container-based edge platforms and challenges

Given the rising interest for container-based edge platforms and the number of solutions available, we start from those identified in a recent survey proposed by the Cloud Native Computing Foundation<sup>27</sup> (CNCF) and planned to be circulated<sup>28</sup> at the KubeCon 2020 that comprehends the most known solutions among the developers.

At the moment of writing this document (July 2020), those that are open-source (for sustainability) and that are backed by big developers' community are:

- ▶ Microk8s
- ▶ K3S
- ▶ KubeEdge

The next section highlights the similarities and differences among such solutions, the support to existing K8S applications and services ecosystem, as well as the technology readiness level according to the official statements from the project maintainers and the community opinions. As anticipated, such initial analysis is a picture of what is available at the time of writing this document, the topic is highly dynamic, and the final decisions could change along with the Pledger project lifespan.

A first comparison about the production readiness is shown in Table 3.

**Table 3: Kubernetes on the edge claimed production-readiness**

<sup>24</sup> <https://github.com/kubernetes/community/tree/master/wg-iot-edge>

<sup>25</sup> <https://github.com/kubernetes/community/tree/master/wg-iot-edge>

<sup>26</sup> <https://github.com/mrichman/kubecon-2018/blob/master/2018-12-12-intro-kubernetes-iot-edge-wg-cindy-xing-huawei-dejan-bosanac-red-hat-preston-holmes-google-steve-wong-vmware.pdf>

<sup>27</sup> <https://thenewstack.io/10-key-attributes-of-cloud-native-applications/>

<sup>28</sup> <https://docs.google.com/document/d/182thvKsieFttk925rIe9S1bZn8TpVDpPMrWt8PXvDBU/edit?ts=5e444465>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	31 of 101	
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	
	<b>Version:</b>	2.0	<b>Status:</b>	Final

Main contributor	Open-source solution	Production-ready
Canonical	MicroK8S	No officially stated, discordant opinions <sup>29</sup>
Rancher	K3S	Yes, stated <sup>30</sup> in the official website
Huawei	KubeEdge	Yes, comes from Huawei Cloud IEF <sup>31</sup>

An initial functional comparison<sup>32</sup> from the community highlights the benefits of K3S over MicroK8S in terms of the amount of resource used. No comparison is available with KubeEdge that, in turn, is more focused on IoT, with the integration of an MQTT broker to support inter-edge and edge-to-cloud communications, the support to some of the most common IoT protocols such as Bluetooth Low Energy and ModBus and the adoption of a pluggable architecture to support additional ones, as by some of the major requirements elicited by the K8S IoT WG<sup>33</sup>.

Additional requirements raised by the community are tools to support automated management and DevOps, that are already provisioned by some Kubernetes tools such as Istio.

Compatibility with the official Kubernetes release is another key point. While K3S and MicroK8S are CNCF certified<sup>34</sup>, KubeEdge seems still not to be but just “under CNCF”<sup>35</sup>. The benefits of CNCF certification are:

- ▶ consistency, to guarantee the compatibility with the official release of Kubernetes
- ▶ timely update, to guarantee the release is up-to-date wrt official Kubernetes
- ▶ confirmability, as the ability to allow anyone to replicate the compatibility verification process.

The approach followed by the solutions listed above is to cut unessential parts from the official Kubernetes to keep it light and (in some cases) have pre-packaged add-ons to get extra capabilities out of the box<sup>36</sup>, such as support to ML with KubeFlow<sup>37</sup>, Istio to manage container security and connectivity, Knative for FaaS and so on.

Another interesting aspect of the container-based edge platforms is the support for ARM boards that have a leading share<sup>38</sup> of the end point ecosystem. ARM is leading the Project Cassini<sup>39</sup> initiative to develop platform standards and bring cloud-native on secure edge infrastructures<sup>40</sup>; together with similar solutions (Azure IoT Edge, VMware Pulse IoT Center, AWS IoT Core, Cisco Kinetic, etc.), this confirms the stakes that major vendors have about containerisation on small, low-cost boards.

<sup>29</sup> <https://serverfault.com/questions/941857/is-microk8s-suitable-for-production-environments-or-is-it-just-for-development>

<sup>30</sup> [https://k3s.io/?\\_hstc=263286291.06022b42be8987226685c3c97bac490b.1583924575513.1583924575513.1583924575513.1&\\_hssc=263286291.1.1583924575514&\\_hsfp=2310034338](https://k3s.io/?_hstc=263286291.06022b42be8987226685c3c97bac490b.1583924575513.1583924575513.1583924575513.1&_hssc=263286291.1.1583924575514&_hsfp=2310034338)

<sup>31</sup> <https://medium.com/@gokulchandrpr/kubeedge-extending-kubernetes-to-edge-dcfedd91f5f9>

<sup>32</sup> [https://www.reddit.com/r/kubernetes/comments/be0415/k3s\\_minikube\\_or\\_microk8s/](https://www.reddit.com/r/kubernetes/comments/be0415/k3s_minikube_or_microk8s/)

<sup>33</sup> <https://docs.google.com/document/d/1We-pRDV9LDFo-vd9DURCPC5-Bum2FvjHUGZ1tacGmk8/edit#heading=h.n4tqqt3iuizs>

<sup>34</sup> <https://www.cncf.io/certification/software-conformance/>

<sup>35</sup> <https://landscape.cncf.io/selected=kube-edge>

<sup>36</sup> <https://microk8s.io/>

<sup>37</sup> <https://www.kubeflow.org/>

<sup>38</sup> <https://www.arm.com/>

<sup>38</sup> [https://media.global.company/investors/PDFs/Arm\\_SBG\\_Q1\\_2019\\_Roadshow\\_Slides\\_FINAL.pdf?revision=9e3e50c9-9b76-48f1-b02c-a30f5b4f2ee4&la=en&\\_ga=2.213120976.575359468.1584523399-183892849.1584523399](https://media.global.company/investors/PDFs/Arm_SBG_Q1_2019_Roadshow_Slides_FINAL.pdf?revision=9e3e50c9-9b76-48f1-b02c-a30f5b4f2ee4&la=en&_ga=2.213120976.575359468.1584523399-183892849.1584523399)

<sup>39</sup> <https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/taming-of-the-edge>

<sup>40</sup> <https://community.arm.com/developer/ip-products/security/b/security-ip-blog/posts/security-for-the-infrastructure-edge>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	32 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

A recent advancement has been the embedding of GPU in low-cost boards, supporting the execution of ML on the edge. For example, NVidia released the Jetson Nano<sup>41</sup> board which includes a CUDA GPU. NVidia also provided an SDK to allow the development of applications on low-cost boards on the edge as well as on more powerful and expensive ones on the cloud, starting from a specialised Docker image<sup>42</sup> to leverage on such board's capabilities. It is worth mentioning that other vendors are following the same approach, such as Intel<sup>43</sup> and Google<sup>44</sup>, and many others are likely to come.

While in the cloud the availability of GPUs has brought to the need to support sharing among different users, on the edge such requirement can be considered less compelling; nevertheless, support to GPU containers has been introduced in containers-based edge platforms to support both ML and not-ML scenarios such as:

- ▶ object recognition and classification (e.g. OpenCV and Support Vector Machine<sup>45</sup>)
- ▶ image processing and computer vision

Among the solutions listed above, K3S<sup>46</sup> and MicroK8S<sup>47</sup> officially support CUDA GPU, while KubeEdge should<sup>48</sup>, although not explicitly mentioned.

The availability of different form factors (e.g. RPi-style boards for Jetson Nano or USB keys for Intel Compute Stick<sup>49</sup>) should meet more constrained environment requirements on the edge.

Without entering the comparison among edge boards, a look at the performance chart<sup>50</sup> from NVidia shows how GPU enabled boards like the Jetson Nano to perform 20-30 times better than similar boards when using ML or image/video processing frameworks and gives an idea of the improved computing capacity now available to the edge.

So, in case of low cost (and low powered) general-purpose COTS programmable boards, the benefits of supporting GPU can be definitely considered a plus.

A side note is that all the solutions above only support Linux containers. To support Windows operating systems, apart from specific cloud vendors solutions (such as Azure IoT Edge), an option is to add a remote compute node based on Windows 2019 to a Kubernetes cluster<sup>51</sup>. Such a solution, like all others based on a centralized Kubernetes installation with remote compute nodes, does not fulfil the autonomous requirement highlighted above and can, of course, be used whereas this is not a strict constraint.

To conclude, here are reported some relevant open challenges that will be addressed by Pledger and are reported to identify the high-level requirements.

One is about security, where the Kubernetes IoT Edge Working Group listed the major security topics<sup>52</sup> about edge platforms, focused on the trust of:

- ▶ Hardware (e.g. trusted ROM and other programmable components)
- ▶ Connected devices (for IoT scenarios, e.g. to trust and accept new devices)

<sup>41</sup> <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>

<sup>42</sup> <https://github.com/NVIDIA/nvidia-docker/wiki/nvidia-docker-plugin>

<sup>43</sup> <https://software.intel.com/en-us/neural-compute-stick>

<sup>44</sup> <https://coral.ai/>

<sup>45</sup> <https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/>

<sup>46</sup> <https://github.com/rancher/k3s/blob/master/vendor/github.com/containerd/containerd/contrib/nvidia/nvidia.go>

<sup>47</sup> <https://microk8s.io/docs/addon-gpu>

<sup>48</sup> <https://github.com/kubeedge/kubeedge/blob/master/vendor/github.com/mindprince/gonvml/nvml.h>

<sup>49</sup> <https://www.intel.com/content/www/us/en/products/boards-kits/compute-stick.html>

<sup>50</sup> <https://devblogs.nvidia.com/jetson-nano-ai-computing/>

<sup>51</sup> <https://kubernetes.io/docs/setup/production-environment/windows/user-guide-windows-nodes/>

<sup>52</sup> <https://github.com/kubernetes/community/tree/master/wg-iot-edge/whitepapers/edge-security-challenges>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	33 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

- ▶ Operating system (trust of bios, firmware and basic o.s. running processes)
- ▶ Network (protocols to exchange data among nodes in the edge or to the cloud)
- ▶ Services (trust of the application layer)

Another open challenge is the network interconnection among different Kubernetes cloud and edge clusters. Apart from the more common L2 or L3 solutions based on VPN, it is worth mentioning the Skupper project<sup>53</sup> based on application-level communication (L7, messages between application endpoints using TLS instead of IP packet routing). It provides, among other features:

- ▶ easiness to configure
- ▶ dynamic load balancing capacity
- ▶ no need to setup any VPN
- ▶ secure inter-cluster and across cluster communication
- ▶ smart routing based on cost and location-awareness

Apart from the specific solution adoption itself, this project will be used as a reference to elicit connectivity requirements later on in this deliverable.

### 3.2.4 Initial feasibility study

Here the initial tests done to verify basic scenarios and estimate the risks of adopting their edge solutions. More specific tests will be documented during the project lifespan:

- ▶ simple cloud cluster for K3S and MicroK8 with 1 master and 2 workers;
- ▶ distributed cloud to edge cluster setup with K3S, using 1 master + 1 worker node on Openstack Ocata + 1 remote worker node on the edge on Raspberry PI3. Master and remote nodes are connected through site-to-site VPN (OpenVPN);
- ▶ edge cluster with K3S with GPU support for ML: 1 master (RPi 4) + 1 worker (RPi 3) + 1 worker (Jetson Nano);
- ▶ deployment of plain (Nginx) and ML (OpenCV + SVM) services as Docker containers.

This should lower the initial risks due to lack of documentation or poor reliability of the solutions identified. Further and specific tests will be carried on once the final architecture will be defined.

### 3.2.5 Expected benefits for the Pledger system

The main advantages of the containerization on the edge, brought to the Pledge platform, have been underlined in the previous subsection and are summarized below.

Some are in common with edge architecture, here briefly reported:

- ▶ low-latency
- ▶ less bandwidth
- ▶ improved privacy and security.

Some more generic, that could be valid also in other domains:

- ▶ standard infrastructure/vendor agnostic
- ▶ open interfaces/APIs
- ▶ low-cost hardware supporting GPU and ML
- ▶ autonomy and ability to overcome unattended and disconnected operations
- ▶ facilitate developer's integration using regular protocols such as HTTP or MQTT

---

<sup>53</sup> <https://skupper.io/index.html>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	34 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

- ▶ facilitate the operation, providing tools to orchestrate, manage the devices and monitor the application.

Some specific for the containerization domain

- ▶ much faster instantiation time compared to plain VMs
- ▶ improved resiliency due to faster and dynamic load balancing capacity
- ▶ optimized resource usage
- ▶ availability of numerous existing applications
- ▶ availability of well-documented services and tools supported by a large community
- ▶ smart routing based on cost and location-awareness.

A possible configuration for Pledger could multiple and autonomous containerized edge clusters that rely on GPU-enabled low-cost boards for ML and video processing.

### 3.2.6 SWOT analysis

Table 4 summarizes the Strengths, Weaknesses, Opportunities, and Threats of the Containerisation on the edge technology that have been identified so far.

Table 4: Containerisation on the edge SWOT table

Strengths	
<ul style="list-style-type: none"> <li>▶ faster deployment and instantiation compared to classic VM</li> <li>▶ support to "low budget" HW such as Jetson Nano with CUDA GPU</li> <li>▶ automatic load management and increased availability</li> <li>▶ recent support to K8S by OSM for orchestration (Nov. 2019)</li> </ul>	
Weaknesses	
<ul style="list-style-type: none"> <li>▶ few solutions available and all quite recent</li> <li>▶ not all are defined as production-ready</li> <li>▶ currently supporting only Linux containers</li> </ul>	
Opportunities	
<ul style="list-style-type: none"> <li>▶ support containers on the edge</li> </ul>	
Threats	
<ul style="list-style-type: none"> <li>▶ low TRL: official K8S with remote workers or small POSs should mitigate possible issues</li> </ul>	

## 3.3 Big Data

### 3.3.1 Overview of Big Data Technologies

Over the past years, the increased digitization and proliferation of digital devices and sensors has led to an explosive growth of data in almost every industry and area of human activity. This has led to the development of Big Data (Boyd & Crawford, 2011) as a field that refers to the technologies used to handle and systematically analyse extremely large data sets, to reveal patterns, trends and insights that might be otherwise unobtainable. The difference with traditional data approaches hinges on basic data characteristics, also known as “the 5 Vs” (Figure 8), namely the Volume, Velocity, Variety, Veracity and Value.

A large variety of technologies and heterogeneous architectures have since been applied in the implementation of big data use cases. Reports illustrating the value of Big Data approaches have also been published by major corporations such as LinkedIn (Sumbaly, Kreps, & Shah, 2013), Netflix (Amatriain, 2013) and others, focusing on the use of open source solutions used as the basis for complex recommendation systems, mining association rules and other uses.

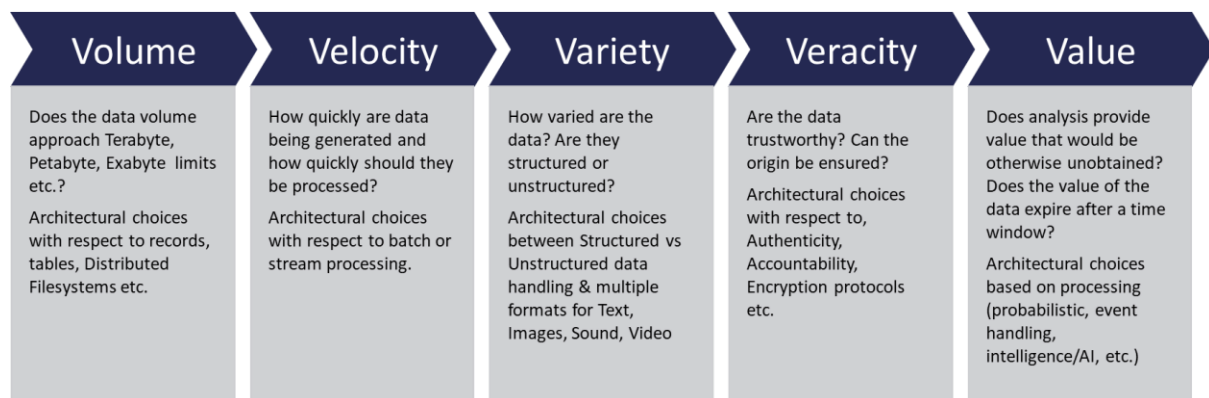


Figure 8: The 5V characteristics of Big Data & related architectural choices for Pledger

Managing Big Data, however, becomes a multifaceted problem that touches upon data storage, data processing, data integration and management. Multiple works (Khan, et al., 2014) (Xiaolong, W.Wah, Cheng, & Wang, 2015) focus on the basic technologies composing a fully-fledged Big Data implementation. In summary, some of the most common features required for Big Data deployments include:

- ▶ **Data Warehousing & Data Ingestion:** Information storage and the provision of data to the consumers’ needs to be able to withstand massive amounts of data.
- ▶ **Processing & Stream Computing:** The provision of data streams to consumers along with in-flight stream processing (e.g. anonymization)
- ▶ **Analytics and Machine Learning:** Extracting knowledge from massive amounts of data requires seamless integration with analytics and ML frameworks.
- ▶ **Integration:** A big data platform needs to be easy to integrate with modern data analytics, data warehousing, stream processing etc.
- ▶ **Data Governance:** The ability to create dedicated, encrypted streams of data which can be accessed by authorized applications is a key technology in modern edge scenarios.

Figure 9 provides a simple conceptual architecture of a generic Big Data solution which handles streaming data. In order to provide meaningful services, four main components are engaged: data

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	36 of 101	
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	
	<b>Version:</b>	2.0	<b>Status:</b>	Final

sources, data storage (databases), data streaming management and knowledge extraction (i.e. processing, analytics, visualisation and business intelligence).

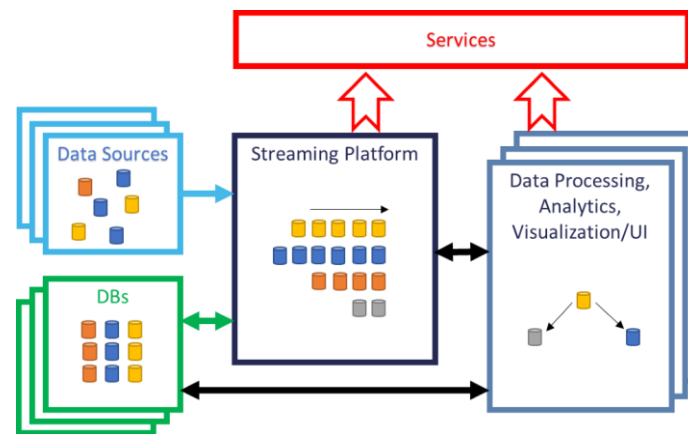


Figure 9: Big Data solution – Conceptual Architecture.

To that end, INTRA is developing and deploying the Streamhandler Platform, which provides the hooks for interconnecting, storing, transforming and processing data as well as training, validating and executing machine learning and Deep learning algorithms, resulting to a full Big Data solution with Artificial Intelligence Capabilities.

### 3.3.2 The Streamhandler Big Data Platform

INTRA’s Streamhandler is a high-performance distributed streaming platform for handling real-time data based on Apache Kafka<sup>54</sup> with demonstrated low latency and high throughput. It can efficiently ingest and handle massive amounts of data into processing pipelines, for both real-time and batch processing. The platform and its underlying technologies can support any type of data-intensive IT services (Artificial Intelligence, Business Intelligence, etc.) from cloud to edge.

The key capabilities and features offered by the platform are:

- ▶ Real-time monitoring and event-processing
- ▶ Interoperability with all modern data storage technologies and popular data sources
- ▶ Distributed messaging system
- ▶ High fault-tolerance - Resiliency to node failures and support of automatic recovery
- ▶ Elasticity - High scalability
- ▶ Security (encryption, authentication, authorization)

<sup>54</sup> <https://kafka.apache.org/>

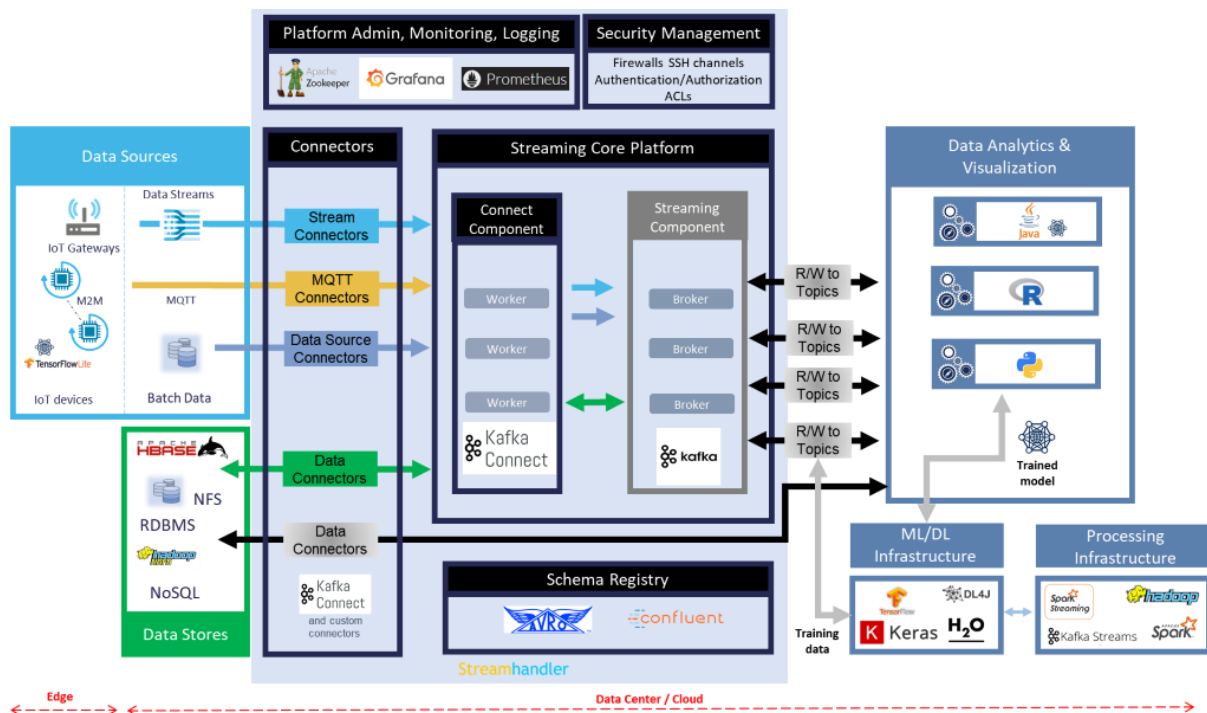


Figure 10: Streamhandler Platform - High level Architecture

As depicted in the Figure 9, Streamhandler consists of:

- ▶ Data sources and data stores
- ▶ Integrated Connectors,
- ▶ Streaming Core Platform,
- ▶ Schema Registry,
- ▶ Security Management and,
- ▶ a Platform Admin and Monitoring Dashboard.

Data sources and data stores represent the structured and unstructured data streams as well as the batch and interactive data sources that will be made available and will be connected to the platform; their choice and implementation are dependent on the Pledger use cases and scenarios.

Data Analytics and Visualisation applications as well as the supporting Processing Infrastructure and Machine learning and Deep Learning Infrastructure can complement the proposed solution; their choice and implementation are dependent on the Pledger use cases and scenarios.

Connectors connect external data sources and make them available to the Streamhandler. External data sources are connected and made available by employing the “Stream Connectors” and “Data Source Connectors”. Data connectors allow the interconnection of data stores for permanent storage of data. Well-known data stores can be integrated to the platform such as Hadoop<sup>55</sup>, Oracle JDBC<sup>56</sup>, Neo4j<sup>57</sup>,

<sup>55</sup> <https://hadoop.apache.org/>

<sup>56</sup> <https://www.oracle.com/database/technologies/appdev/jdbc.html>

<sup>57</sup> <https://neo4j.com/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	38 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

InfluxDB<sup>58</sup>, Cassandra<sup>59</sup>, MongoDB<sup>60</sup>, Elasticsearch<sup>61</sup> and more. The Confluent Hub<sup>62</sup> offers a huge variety of data connectors for both data stores and sources. In addition, the Kafka Connect API<sup>63</sup> simplifies the integration of a new data source or sink, enabling the development of custom-made connectors.

The Streaming Core component is implemented as an Apache Kafka cluster with multiple brokers to maintain load balancing and replication. The underlying technologies and capabilities of the Streaming Component and the multiple workers of the Connect Component allow the realisation of scalable and secure stream data pipelines. Apache Kafka allows to producers and consumers to publish and subscribe to streams of records (topics) similar to the functionality provided by a message queue. Producers push messages into a Kafka topic, while consumers pull messages off a Kafka topic. The streams of records are stored in a fault-tolerant durable way and consumers (Big Data Apps or AI Apps) can process them as they occur. In general, Kafka is suitable for building real-time streaming data pipelines to reliably get data between systems or applications and for building real-time streaming applications that transform or react to the streams of data. The flexibility of running the Communication Platform in a clustered manner allows for the horizontal scalability of the system achieving thus a scalable, fault-tolerant communication-efficient framework for cross-streaming data management and integration. Kafka Streams partitions the data within a topic using logical entities such as partitions and tasks to achieve data parallelism and enable elasticity, scalability, high performance, and fault tolerance.

The Schema Registry allows the definition and storage of data models describing the data. The Schema Registry is implemented through a Kafka add-on (Confluent Schema Registry) that exposes a RESTful interface for storing and retrieving schemas defined in Apache Avro serialization format. It stores a versioned history of all schemas, provides multiple compatibility settings and allows evolution of schemas according to the configured compatibility settings and expanded Avro support. It provides serializers that plug into Kafka clients that handle schema storage and retrieval for Kafka messages that are sent in the Avro format.

Security Management includes a set of mechanisms that enhance the security of the platform. Mainly, three components are included:

- ▶ Encryption of in-flight data using SSL / TLS: This allows data to be encrypted between producers and Kafka and between consumers and Kafka.
- ▶ Authentication using SSL or SASL: This allows Pledger data producers and data consumers to authenticate to the Kafka cluster, which verifies their identity.
- ▶ Authorization using Access Control Lists (ACLs): Once the individual clients are authenticated, the Kafka brokers can run them against access control lists (ACL) to determine whether or not a particular client would be authorised to write or read to a specific topic.

As expected, the encryption of data that are communicated between clients do incur an overhead which apart from the actual encryption and decryption algorithm execution costs, there exists a further performance implication to the platform due to the fact that the data now has to move to the Application Context and Zero Copy is not possible.

The Platform Admin Dashboard and Monitoring Tools include components that allow cluster configuration and provide constant overview of the cluster performance and health status, such as

---

<sup>58</sup> <https://www.influxdata.com/>

<sup>59</sup> <http://cassandra.apache.org/>

<sup>60</sup> <https://www.mongodb.com/>

<sup>61</sup> <https://www.elastic.co/elasticsearch/>

<sup>62</sup> <https://www.confluent.io/hub/>

<sup>63</sup> <https://kafka.apache.org/11/javadoc/org/apache/kafka/connect/connector/Connector.html>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	39 of 101	
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	
	<b>Version:</b>	2.0	<b>Status:</b>	Final

Prometheus<sup>64</sup>, Grafana<sup>65</sup> and Zookeeper<sup>66</sup> service etc. Metrics important to the health status of the platform can be scrapped through Prometheus and visualized through Grafana dashboards (Figure 11). The ZooKeeper service is a centralized service for maintaining cluster configuration information and naming and also for providing distributed synchronization among the Platform components. The platform components are capable of exporting Java Management Extensions (JMX) metrics which in turn are made available in customised Grafana dashboards. The available information includes the status of the Zookeeper chorus, the status of the Kafka Brokers together with insightful graphs about incoming and outgoing data throughputs, message rates per broker and per topic and other information regarding Central Processing Unit (CPU) and memory usage per broker. Similarly, healthy Schema Registry and Rest Proxy instances and their corresponding connections are monitored. The cluster overview is completed by Kafka Connect specific panels which present the user with running Connectors and tasks information in addition to data rates per worker node. The available dashboards allow not only a very good overview of the cluster performance and health status but also provide significant information when performance tuning is necessary.



Figure 11: Streamhandler Platform – Monitoring Dashboard

### 3.3.3 Streamhandler Integration & Interoperability

INTRA's Streamhandler Platform can interoperate with IoT and edge architectures and provide end-to-end integration with the Pledger system using for example the Message Queuing Telemetry Transport (MQTT) protocol. MQTT is a widely used ISO-standardised (ISO/IEC 20922:2016 (International Standards Organisation (ISO), 2016)) publish-subscribe-based messaging protocol. MQTT has many implementations such as Mosquitto<sup>67</sup> or HiveMQ<sup>68</sup>. MQTT and Apache Kafka are a perfect combination for end-to-end IoT integration from edge to datacenter with bidirectional messaging.

Data sources and Data stores can be made available and potentially be connected to the Big data platform, generated by any device and/or gateway on the edge. Similarly, and according to the requirements, appropriate persistent storage can be used, as depicted in the input/output data components (Figure 10). The described data sources will be seamlessly integrated with processing components by the means of integration connectors (Connectors). Streamhandler can efficiently interoperate with all the modern data storage technologies of a Big data ecosystem such as traditional RDBMS (e.g.

<sup>64</sup> <https://prometheus.io/>

<sup>65</sup> <https://grafana.com/>

<sup>66</sup> <https://zookeeper.apache.org/>

<sup>67</sup> <https://mosquitto.org/>

<sup>68</sup> <https://www.hivemq.com/blog/how-to-get-started-with-mqtt/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	40 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

MySQL<sup>69</sup>, JDBC<sup>70</sup>) other persistence approaches such as NoSQL solutions (e.g. MongoDB<sup>71</sup>, Apache HBASE<sup>72</sup>), distributed filesystems (e.g. Hadoop Distributed Filesystem HDFS<sup>73</sup>).

Data analytics and Data Visualization represent the applications that perform the data processing and analytics. These are dependent on the exact Pledger use case needs and can be implemented by the consortium partners in any programming language typically preferred for data science (such as Python, Java, R and Scala) or any native programming language (e.g. C/C++, Haskell, etc.).

Underlying Processing and Machine Learning (ML)/ Deep learning (DL) infrastructure may span multiple Virtual Machines in order to provide all the necessary functionalities. A distributed computing environment enables the interoperation of components that enable the storage and analysis of the data involved and further allows the usage of any technology agnostic algorithms. Streamhandler can integrate seamlessly with processing frameworks (e.g. Apache Spark<sup>74</sup>, Hadoop<sup>75</sup>, Kafka Streams<sup>76</sup>, Spark Streaming<sup>77</sup>) and with ML/DL infrastructure that provides all the necessary components for the analysis of the data in order to build analytics models (e.g. using open-source frameworks like TensorFlow<sup>78</sup>, DeepLearning4J<sup>79</sup>, or H2O.ai.<sup>80</sup>).

### 3.3.4 Expected benefits for the Pledger system

The selected technology brings significant benefits to Pledger including:

- ▶ Enterprise-level performance and availability,
- ▶ Low latency distributed messaging system,
- ▶ Strong security and authentication/authorization mechanisms,
- ▶ Elasticity and scalability,
- ▶ Interoperability with all modern data storage technologies and popular data sources,
- ▶ Seamless integration with modern processing and ML frameworks,
- ▶ High fault-tolerance - Resiliency to node failures and support of automatic recovery,
- ▶ Elasticity - High scalability.

In particular, the platform is a fully featured industrial grade solution:

- ▶ which is capable to scale out and accommodate various and from different domains Big Data, interoperating with all modern data storage technologies as well as other persistence approaches and
- ▶ can support all important Big Data languages including Python, Java, R and Scala as well as other traditional programming approaches.

### 3.3.5 SWOT Analysis

The relevance of this technology, both for the project and for edge deployments globally, is further highlighted by its fast growth. According to MarketsAndMarkets (MarketsAndMarkets), the Big Data streaming analytics market size is expected to grow from USD 10.3 billion in 2019 to USD 35.5 billion

<sup>69</sup> <https://www.mysql.com/>

<sup>70</sup> <https://www.oracle.com/database/technologies/appdev/jdbc.html>

<sup>71</sup> <https://www.mongodb.com/>

<sup>72</sup> <https://hbase.apache.org/>

<sup>73</sup> [https://hadoop.apache.org/docs/r1.2.1/hdfs\\_design.html](https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html)

<sup>74</sup> <https://spark.apache.org/>

<sup>75</sup> <https://hadoop.apache.org/>

<sup>76</sup> <https://kafka.apache.org/documentation/streams/>

<sup>77</sup> <https://spark.apache.org/docs/latest/streaming-programming-guide.html>

<sup>78</sup> <https://www.tensorflow.org/>

<sup>79</sup> <https://deeplearning4j.org/>

<sup>80</sup> <https://www.h2o.ai/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	41 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

by 2024, at a Compound Annual Growth Rate (CAGR) of 28.2% during this period. Driven by the emergence of edge Computing and the increasing IoT, smartphone and internet penetration, this expected increase in market size uncovers a business strategic shift toward real-time provisioning and analysis of data for faster decision making. Additional sources further support the same conclusions. For example, Mordor Intelligence estimates<sup>81</sup> market size at USD 7.08 billion in 2019 and expects to reach a value of USD 38.53 billion by 2025 at a CAGR of 32.67%, during the forecast period (2020-2025). Digitization and cloud Computing are considered vital catalysts in the big data digital transformation, and Pledger is in the position to greatly benefit from a Big Data approach.

Table 5 summarizes the Strengths, Weaknesses, Opportunities, and Threats for Big Data.

**Table 5: Big Data Platform SWOT analysis**

<b>Strengths</b>	
<ul style="list-style-type: none"> <li>▶ Faster content delivery to data consumers (e.g. devices, applications, ML frameworks etc.)</li> <li>▶ Authenticated and encrypted data access</li> <li>▶ Authorisation mechanisms to govern access of applications to the data steams</li> <li>▶ High availability &amp; fault-tolerance</li> <li>▶ Production-ready performance</li> <li>▶ Publish/Subscribe operation</li> <li>▶ Elastic operation, high scalability</li> <li>▶ Easy integration with cutting-edge AI and FaaS frameworks</li> </ul>	
<b>Weaknesses</b>	
<ul style="list-style-type: none"> <li>▶ Lack of integration with legacy systems: Integration is performed on a case-by-case basis by development of dedicated connectors</li> <li>▶ Quality of integrated analytics &amp; data quality/relevance is up to the data consumers/producers</li> </ul>	
<b>Opportunities</b>	
<ul style="list-style-type: none"> <li>▶ Emerging technologies drive the technological advancement of Big Data Streaming and create new market opportunities:               <ul style="list-style-type: none"> <li>- IoT and AI</li> <li>- Real-time recommendations, pattern detection and anomaly detection</li> <li>- 5G, edge computing and hybrid clouds</li> </ul> </li> <li>▶ Development of privacy preserving stream processing mechanisms</li> </ul>	
<b>Threats</b>	
<ul style="list-style-type: none"> <li>▶ Siloed data and lagging cooperation among providers of data producer and consumer applications.</li> <li>▶ Perceived threats in data protection.</li> </ul>	

<sup>81</sup> <https://www.mordorintelligence.com/industry-reports/streaming-analytics-market>

## 3.4 Continuous Integration/Continuous Delivery

### 3.4.1 Overview of CI/CD

A crucial part of complex software systems development and delivery lifecycle is Continuous Integration (CI) and Continuous Delivery/Deployment (CD) – jointly mentioned as CI/CD. CI, in software development, is a practice of building/integrating and testing all developer working code frequently in a shared code repository (Figure 12). A common practice in CI is to integrate the changed code at least daily. The frequent integration helps the contributors to notice any arising errors and correct them instantly.

In order to benefit from the CI process, we must perform extensive testing of each software component/module but also of the integrated platform as a whole. Automated per component tests and combined integration tests that are performed on each new build (version) of a component shall be executed and in case of success the integrated platform will be updated with the new version of the component. Otherwise the developers will be notified so as to take proper action to fix the problem that caused the failure.

CI/CD is the cornerstone of modern agile development and is closely related to the concept of DevOps (Dyck, Penners, & Lichter, 2015). DevOps is a concept that allows the combination of software development (Dev) practices with IT operations (Ops) in order to shorten system development cycles and provide high quality software. Deployment of services and management of their lifecycles is fast and flexible, in stark contrast with traditional waterfall approaches.

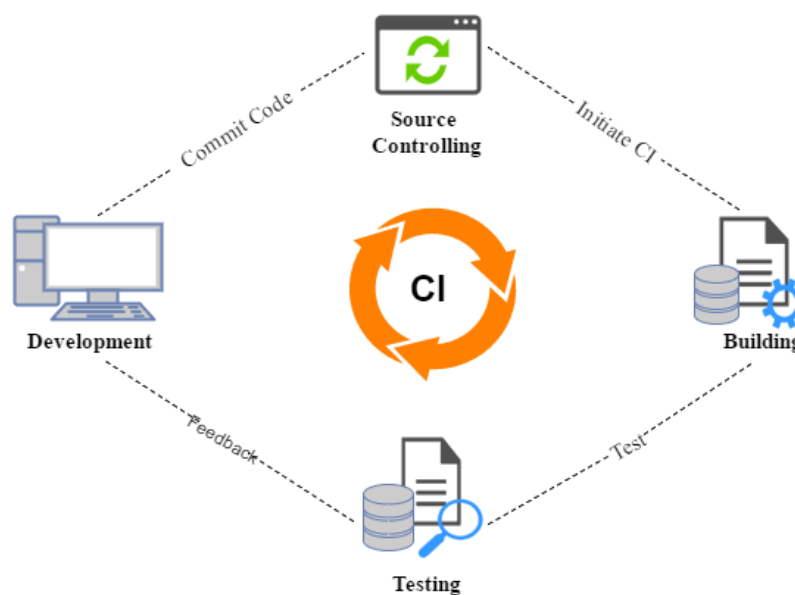


Figure 12: The Continuous Integration Lifecycle.

### 3.4.2 The Pledger CI/CD Environment

INTRA proposes a Test-Driven Development (TDD) (Kent, 2002) approach, starting from unit tests per software component, upon specified test cases for each component, proceeding to integration tests that validate the correct functionality of the integrated platform that involves two or more components in an automated manner with the use of a CI server, such as Jenkins<sup>82</sup>. Many teams find that this approach

<sup>82</sup> <https://jenkins.io/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	43 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.

In case that the resulting software system is composed of several software components/modules, developed by diverse development teams, a collaborative software development approach can be followed, during which it is important to ensure data integrity and availability. To support this, a distributed version control system (VCS) is necessary for the efficient system software delivery. For this purpose, GitLab<sup>83</sup> could be utilized, a source code management and VCS that aims mostly at data integrity and full version tracking. GitLab is an easy yet powerful and intuitive git VCS. Multiple developers can concurrently create, merge and delete parts of the code they are working on independently, at their local system before applying the changes to the shared GitLab repository.

To ensure quality of software development, build automation is additionally pursued. Build automation is the act of automating processes that are associated with software building. Such processes might include various parts like source code compiling into binary code, packaging binary code and automated test running but also the delivery/deployment and documentation parts. Maven<sup>84</sup> is a useful tool for build automation and project management for projects that are written in numerous programming languages (Java, Ruby, C# and other). This process can be applied for components software shared and residing in the software source code repository.

In collaborative software development it is also essential to distribute the software efficiently. Among others, Sonatype Nexus<sup>85</sup> is a popular repository manager that manages the required software artifacts. It allows developers to distribute their software easily but also to proxy, publish and collect the necessary project dependencies. Actually, the Nexus Repository offers a standardized way for cataloguing and storing developers' artifacts. Once a new library is developed, it is handed out to the repository manager. After that, other developers can efficiently access these software components by using a standardized procedure. Clearly, it is possible to control centrally the development of all artifacts and the access to them. Nexus can be used for pre-built software components, the source code of which is not available or shared – however, updated versions of software components need to be frequently built and released in new versions at the Nexus repository to support continuous integration and delivery. The CI server will monitor the repository and will initiate a new platform test and deployment cycle after each new version upload. An alternative highly scalable server-side application that may be used and let software teams to share and distribute docker images is Docker Registry V2<sup>86</sup>. This solution has an integrated REST API<sup>87</sup> that let developers to explore their images via a REST call. Together with a docker registry user interface solution<sup>88</sup> this may be a good alternative for the case of Pledger.

To further support automated delivery (Continuous Delivery - CD) in software development, Docker<sup>89</sup> has been chosen, an open-source software containerization platform, as a straightforward way to provide isolated running environments with pre-set configuration. Docker works with software containers in order to allow the software to run always the same, independently of the deployment environment. It wraps up the software in a complete file system, along with any necessary tools or software resources, such as libraries, code and runtime. This way, multiple docker containers can run on a single Linux instance, without any overhead for managing several virtual machines. Moreover, by using Docker, the software system deployment is simplified at a great extent, set up is minimal and uniform across all component projects. Each component is contained in a different Docker image ready to be executed in

---

<sup>83</sup> <https://about.gitlab.com/>

<sup>84</sup> <https://maven.apache.org/>

<sup>85</sup> <https://www.sonatype.com/product-nexus-repository>

<sup>86</sup> <https://docs.docker.com/registry/>

<sup>87</sup> <https://docs.docker.com/registry/spec/api/>

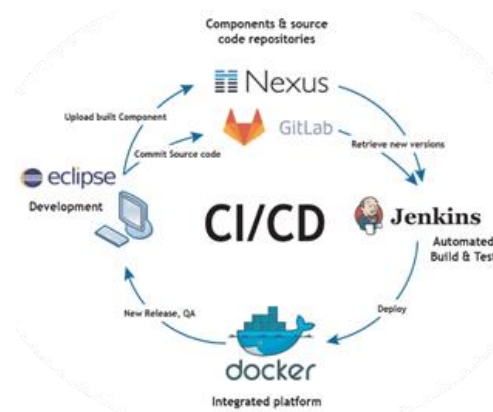
<sup>88</sup> <https://joxit.dev/docker-registry-ui/demo/?page=1#!>

<sup>89</sup> <https://www.docker.com/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	44 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

a Docker hosting machine as a separate container. These images can be published in a shared repository, such as the Docker registry, and through the Docker Compose functionality these images can be retrieved from the Docker registry and deployed together via a single configuration file. Containerization thus provides OS level virtualization. This means that multiple applications running in containers on a single host, access the same OS kernel. Hence, it is faster and more lightweight than isolating applications using VMs. Containers have an initial configuration which does not affect the configuration of other containers, even though they share the same host OS. This eliminates errors due to unexpected conflicts or missing dependencies, which are common when applications are installed on a single host without isolation. In more demanding installations due to increased load of the system, Docker is perfectly suitable to be configured with load balancing mechanisms that can scale up the performance of the system. Figure 13 depicts the CI/CD workflow with all the tools mentioned above, and summarized below:

- ▶ GitLab for source control, acting as code repository and allowing code versioning,
- ▶ Jenkins for automated build and testing,
- ▶ Docker for containerization of services and components,
- ▶ Docker Registry for easy deployment at different infrastructures.



**Figure 13: Continuous Integration & Continuous Delivery process**

Using the CI/CD environment and tools, when developers implement new component features or integration endpoints, they push their code to GitLab, the central source code repository in this environment, which is then compiled, built and tested using Jenkins, while Docker is finally creating a Docker image, that is pushed to the Docker registry. Once components have been built and their images have been pushed to the docker registry, they are available to be pulled from any server which has access to the docker registry. The deployment to Deployment servers can be carried out via a script which automates the entire process over a secure Virtual Private Network (VPN) connection or any other encrypted and authenticated way (i.e. HTTPS with user authentication), as shown in Figure 14.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	45 of 101	
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	
	<b>Version:</b>	2.0	<b>Status:</b>	Final

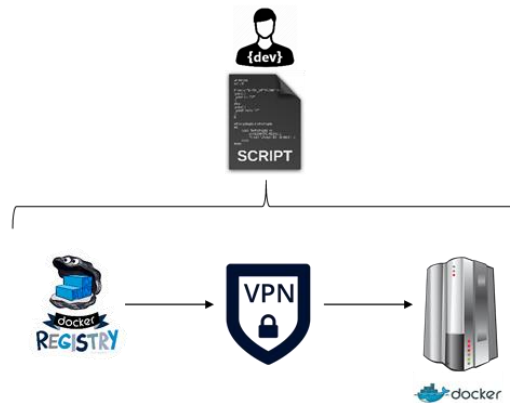


Figure 14: Flow from the docker registry to the deployment server

Whenever code changes are required, component developers push their source code to GitLab. Afterwards, Jenkins takes over from that point automatically performing the steps described above. Eventually, the component is built into an image and pushed into the Docker registry. Then, a new version of the system is specified, and the deployment script is rerun to install the updated software system at the Deployment server. Figure 15 illustrates the flow from code changes (e.g. bug fix, implementation of new features) to re-deployment on any server.

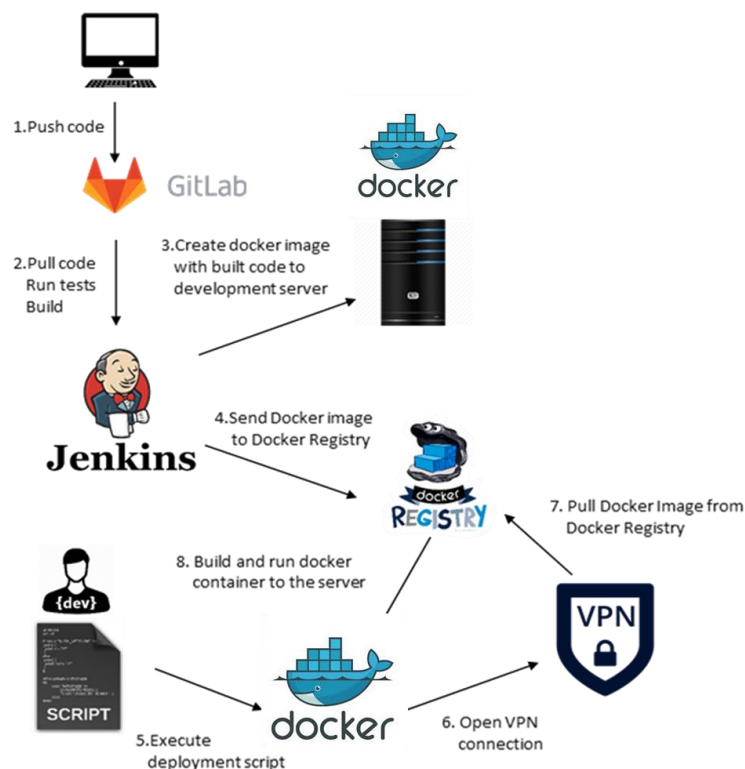


Figure 15: Code Changes Workflow

Docker containers are ephemeral. This means that when a container is removed, its state (and data) is also lost. To remedy this, docker offers named volumes on which containers mount. These containers exist independently from the containers that use them.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	46 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

INTRA will provide the CI/CD environment for the project and enhance it with much needed features to secure the pipeline such as HTTPS support, dedicated Firewall setup, hardening of Docker daemons etc. The exact selection and configuration of additional security measures will be selected in T4.1.

### 3.4.3 Expected benefits for the Pledger system

Continuous Integration offers significant advantages such as:

- ▶ reduction of risk in the development, since it reveals any possible incompatibility between software components early during the development phase
- ▶ facility of instant bug fixing
- ▶ availability of current product version at any moment, as the code is frequently integrated

### 3.4.4 SWOT analysis

The relevance of this technology for Pledger is also illustrated by the rapid growth of this market. According to MarketsAndMarkets (MarketsAndMarkets, 2019), the DevOps market size is expected to grow from USD 2.90 Billion in 2017 to USD 10.31 Billion by 2023, at a Compound Annual Growth Rate (CAGR) of 24.7% during the forecast period. The demand for DevOps solutions and services among enterprises is expected to gain huge traction, due to the increasing need for fast application delivery with high quality.

The major players in the DevOps market are CA Technologies (US), IBM (US), Atlassian (Australia), MicroFocus (UK), Puppet (US), Red Hat (US), AWS (US), Microsoft (US), Google (US), Oracle (US). It is evident that, at this point, the market is dominated by US vendors and it is crucial to enhance EU's capacities in this area. Table 6 summarizes the Strengths, Weaknesses, Opportunities, and Threats for secure CI/CD.

Table 6: CI/CD Platform SWOT analysis

Strengths	
<ul style="list-style-type: none"> <li>▶ Fast development and testing cycles following an agile approach</li> <li>▶ Chance to identify and fix bugs in early stages of software development</li> <li>▶ Creating a secure pipeline to the docker registry</li> </ul>	
Weaknesses	
<ul style="list-style-type: none"> <li>▶ Mixed developer talent pool</li> <li>▶ Difficult to refactor</li> </ul>	
Opportunities	
<ul style="list-style-type: none"> <li>▶ Development of new DevOps processes</li> <li>▶ Chance to build up European capacities in DevSecOps</li> <li>▶ Rapidly growing market</li> <li>▶ Stable pace of new framework and library releases</li> </ul>	
Threats	
<ul style="list-style-type: none"> <li>▶ Lack of cooperation among developers</li> <li>▶ High degree of evolutionary development</li> </ul>	

### 3.5 Blockchain

The following section will focus on the important requirements of combining the blockchain technology with the edge computing under the umbrella of Pledger. In particular, it is divided in these sub-sections:

- ▶ Microtransactions Framework for the Edge: Requirements
- ▶ Overview of Blockchain Technologies
- ▶ Benefits to Pledger
- ▶ SWOT Analysis

#### 3.5.1 Microtransactions Framework for the Edge: Requirements

Edge computing suffers from difficulties of security and decentralized management. In the context of resolving such issues by tackling any prompted research challenge, blockchain is the new convenient technology that can provide such protection and relief (e.g. Figure 16) Blockchain can be combined with edge computing into one system and assist with efficient and low-cost solutions.

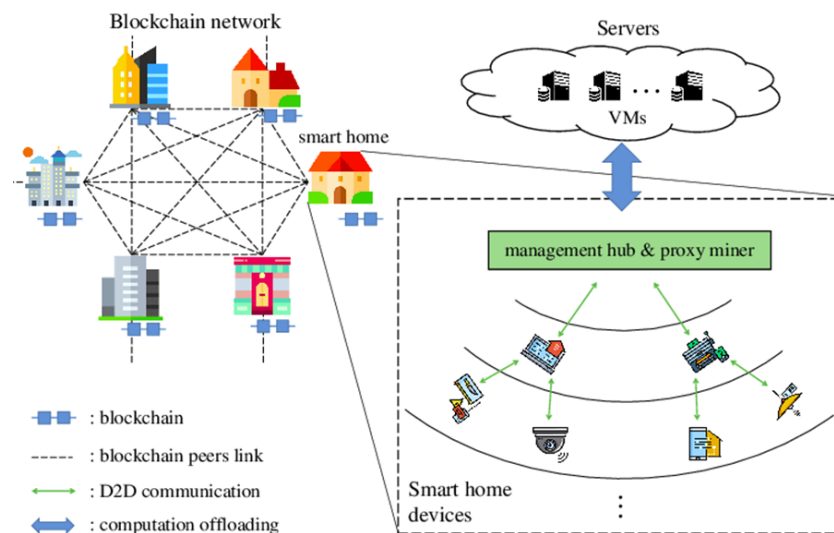


Figure 16: Multi-hop blockchain-empowered mobile edge computing

In the concepts of this interoperability and integration, both edge computing and blockchain technologies could co-exist when certain requirements are met properly, according to recent research (R. Yang, 2019). Such requirements are simplified and listed below (Figure 17):

- ▶ Authentication: edge computing environments that consist of numerous entities, such as services, providers and infrastructures, need a validation of their authenticity, hence authentication process is crucial. All entities' data and privileges can be tracked and documented through blockchain smart contracts. Thus, secure communication channels between the participants of the edge can be structured.
- ▶ Adaptability: while edge-node devices and applications complexity increases, an integrated blockchain-based system on the edge should be capable of adapting with ease and flexibility at any fluctuation of i.e. the connected users and devices quantity, the bandwidth, etc.
- ▶ Network security: heterogeneous attacks could offend the security of the edge network. The integration of blockchain into edge computing networks will essentially improve communications management and monitoring of edge simple nodes and distributed servers; thus, effective prevention of malicious network events.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	48 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

- ▶ Data integrity: more reliable data accuracy and consistency over the entire network path is required. By exploiting and combining the plethora of distributed resources of edge computing and decentralized features of blockchain, data integrity can be improved.
- ▶ Verifiable computation: “verifiable computation” of edge computing means to offer offloaded computation to specific clients. Due to vulnerability of edge nodes and devices, the results of this computation could be tampered. Blockchain co-operation provides the offload results to be guaranteed for their correction and validity.

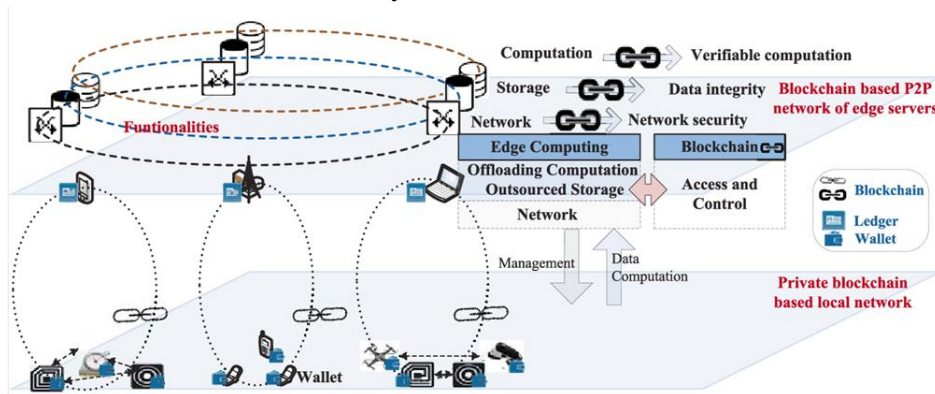


Figure 17: Integrated blockchain and edge computing systems

In the sub-section below the different blockchain technologies (most with direct interest to Pledger) are described.

### 3.5.2 Overview of Blockchain Technologies

In this sub-section the different necessary state-of-the-art blockchain technologies are described. They are grouped in these categories:

- ▶ Paywalls
- ▶ Smart Contracts
- ▶ Hyperledger (specific platform)

#### 3.5.2.1 Paywall

Popular news providing websites are constructed in a way that a user must sign up or subscribe (e.g. monthly, yearly plans, etc) in order to access the content. In the internet world, the paywall approach appears in large volumes the last decade.

The concept of blockchain-based paywall service for websites or blog articles monetization was firstly attempted in an enterprise-interesting way in 2013<sup>90</sup>, with the open-source experiment known as “Bitmonet”<sup>91</sup>. From then on, more blockchain-based monetization practices were trying to innovate in this domain. In this sub-section, some of these contemporary blockchain-based paywall solutions are simply explained and elaborated below.

<sup>90</sup> <https://www.coindesk.com/bitmonet-unveils-free-bitcoin-paywall-function-for-publishers>

<sup>91</sup> <http://bitmonet.com/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	49 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

### 3.5.2.2 Satoshiwall

A modern blockchain-based paywall is known as Satoshiwall<sup>92</sup>. It is a bitcoin cash-based (BCH<sup>93</sup>) paywall service. Through the platform (Figure 18), a new user is offered a customizable paywall that accepts BCH for payments<sup>94</sup>.

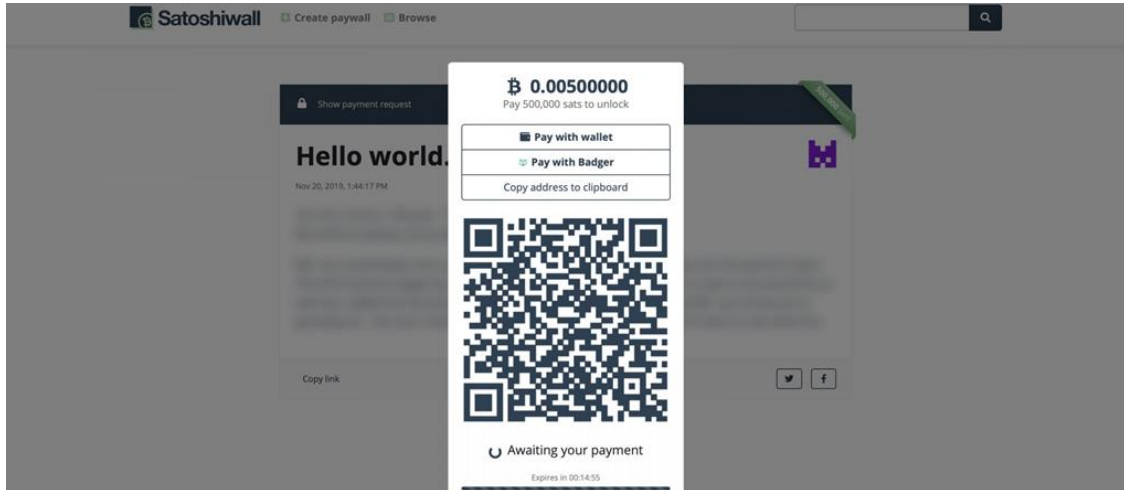


Figure 18: Satoshiwall platform.

### 3.5.2.3 Boltwall

Another widely-known developer paywall service for blockchain technology is “Boltwall”<sup>95</sup>. Boltwall is a paywall project build for the Bitcoin Lightning Network<sup>96</sup>. With Boltwall<sup>97</sup>, by simply setting up a lightning node<sup>98, 99</sup>, a developer can monetize access to their API and more without requiring user accounts, API keys, credit cards, or storing any user data (Figure 19).

<sup>92</sup> <https://satoshiwall.cash/>

<sup>93</sup> <https://coinmarketcap.com/currencies/bitcoin-cash/>

<sup>94</sup> <https://news.bitcoin.com/developer-launches-bch-powered-paywall-service/>

<sup>95</sup> <https://tierion.github.io/boltwall/>

<sup>96</sup> <https://www.coindesk.com/learn/bitcoin-101/what-is-the-lightning-network>

<sup>97</sup> <https://github.com/Tierion/boltwall>

<sup>98</sup> <https://www.bitcoinmarketjournal.com/bitcoin-lightning-network-node/>

<sup>99</sup> <https://techexpert.tips/bitcoin/bitcoin-lightning-node-linux/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	50 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

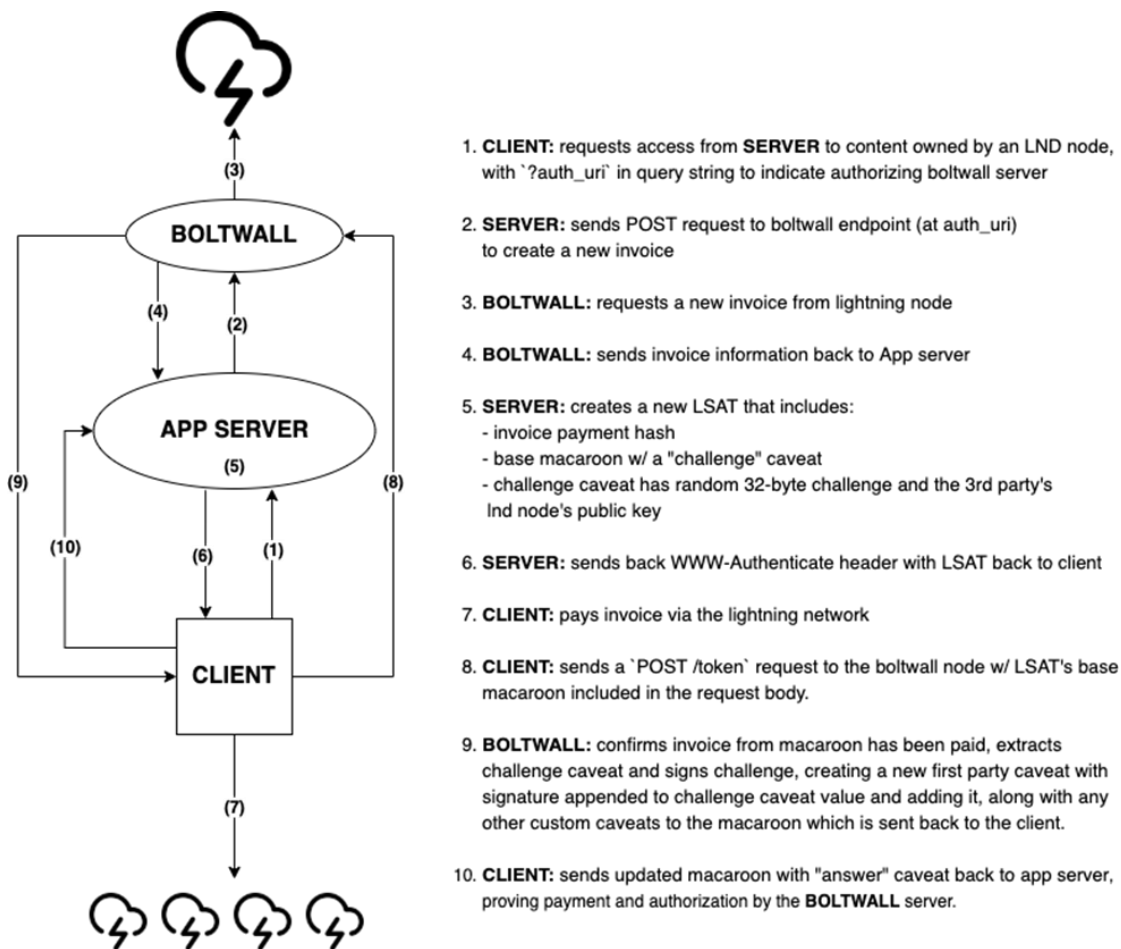


Figure 19: Boltwall Flowchart & Explanation.

### 3.5.2.4 Smart Contracts

Public blockchains generated the new technology called “Smart Contracts”<sup>100</sup>. Firstly perceived in 1993, cryptographer and computer scientist Nick Szabo mentioned the notion of smart contracts as a type of vending machine in the digital world<sup>101</sup>. While in the beginning it may be challenging to grasp the concept of smart contracts, in a simple way they can be described like this: while e.g. a law contract forms the conditions of an agreement through paper, a smart contract forms an agreement through cryptographic code (Figure 20). In the broader sense, smart contracts are programmed code that executes exactly when arranged by their creators (developers). For instance, an Ethereum<sup>102</sup> user can send one (1) ether<sup>103</sup> to another on a specific date using a smart contract. The first user would first create the smart contract and then put the data into the smart contract so that it can execute at the programmed date and time.

<sup>100</sup> <https://www.coindesk.com/learn/ethereum-101/ethereum-smart-contracts-work>

<sup>101</sup> <http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/sza-bo.best.vwh.net/idea.html>

<sup>102</sup> <https://ethereum.org/>

<sup>103</sup> <https://coinmarketcap.com/currencies/ethereum/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	51 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

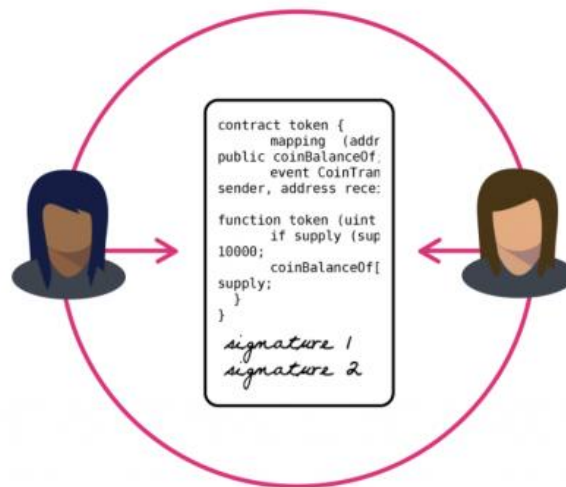


Figure 20: Visual representation of Smart Contract concept

Ethereum is a public blockchain network<sup>104</sup>, whereas Quorum is a permissioned one<sup>105</sup>. In a public blockchain network, anyone can join in the sense that they can read and write on the blockchain ledger or execute smart contracts. They offer immutability of submitted transactions and executed smart contracts while no one has control over the network activity. In permissioned blockchains, a candidate user must fulfil the requirements or simply be accepted from an already member or admin of the network.

Different type of networks host smart contracts of different nature. From the project's perspective, the different natures of smart contracts are described below.

#### 3.5.2.5 Public network (Ethereum Smart Contracts)

In public blockchain of Ethereum (Figure 21), several use cases can be implemented, such as money transfer in the form of currencies, betting platform transactions, video games monetization (manually or automated i.e. smart contracts). Thus, the corresponding smart contracts can be seen by the whole network (all participants) and any user can organize, program and deploy smart contracts on the Ethereum network.



Figure 21: Logo of Ethereum

<sup>104</sup> <https://en.wikipedia.org/wiki/Blockchain>

<sup>105</sup> <https://101blockchains.com/permissioned-blockchain/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	52 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

### 3.5.2.6 Permissioned network (Hyperledger<sup>106</sup>, Quorum<sup>107</sup>, R3 Corda<sup>108</sup>)

As mentioned above, in permissioned blockchains (Figure 22), only selected participants are able to join the network. Of these members, and depending on the platform, they are able to execute smart contracts on the corresponding private ledger. For instance, in Hyperledger (see 3.5.2.3 below), private channels are keeping transaction and smart contract data separately from user that are not authorized in the channel.



Figure 22: Logos of Hyperledger, Quorum, R3 Corda

### 3.5.2.7 Hyperledger

Hyperledger is an open source collaborative effort to advance cross-industry blockchain technologies, hosted by The Linux Foundation. It is mainly an umbrella project supported by industry giants<sup>109</sup> while it consists at the time of writing of various blockchain platforms (distributed ledgers or DLT<sup>110</sup>), libraries and developer tools<sup>111</sup>:

- ▶ Hyperledger Aries (library)
- ▶ Hyperledger Avalon (tool)
- ▶ Hyperledger Besu (DLT)
- ▶ Hyperledger Burrow (DLT)
- ▶ Hyperledger Caliper (tool)
- ▶ Hyperledger Cello (tool)
- ▶ Hyperledger Explorer (tool)
- ▶ Hyperledger Fabric (DLT)
- ▶ Hyperledger Grid (domain-specific)
- ▶ Hyperledger Indy (DLT)
- ▶ Hyperledger Iroha (DLT)
- ▶ Hyperledger Quilt (library)
- ▶ Hyperledger Sawtooth (DLT)
- ▶ Hyperledger Transact (library)
- ▶ Hyperledger Ursa (library)

In Pledger, the Hyperledger Fabric<sup>112</sup> project can bring all of its security, data-privacy and protection and decentralization in permissioned networks features which are better elaborated below:

<sup>106</sup> <https://www.hyperledger.org/>

<sup>107</sup> <https://www.goquorum.com/>

<sup>108</sup> <https://www.r3.com/corda-platform/>

<sup>109</sup> <https://www.hyperledger.org/members>

<sup>110</sup> [https://en.wikipedia.org/wiki/Distributed\\_ledger](https://en.wikipedia.org/wiki/Distributed_ledger)

<sup>111</sup> <https://www.hyperledger.org/projects>

<sup>112</sup> <https://www.hyperledger.org/projects/fabric>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	53 of 101	
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	
	<b>Version:</b>	2.0	<b>Status:</b>	Final

- ▶ Private transactions and private smart contracts in enterprise blockchain networks on the edge. Fabric offers a permissioned blockchain network that by definition emphasizes on user privacy and blockchain data immutability.
- ▶ Channels for confidential transaction execution and smart contract triggering only by privileged members. Each channel has its own participants and activity inside the channel is not available to the other network members (Fabric network).
- ▶ Data partitioning for sensitive data management simplifies enterprise data and user account information administration, organization and maintenance.
- ▶ Privacy enhancing mechanisms sustained by anonymous personalization services. Pledger can exploit the advantages of such technology while equipped with concurrent protection of the edge user data and personalized services based on their secured information.
- ▶ Decentralized applications (DApps). Ability to create, aggregate, modify and deploy DApps inside the Fabric’s blockchain network.
- ▶ Computation-free consensus algorithms. Blockchain ledger is structured by modern permissioned network consensus algorithms that agree on the ledger state without wasting energy resources<sup>113</sup>; thus, there is no computational overhead from mining.

Hyperledger Fabric already applies to an abundance of use cases i.e. in Finance, Banking, Healthcare, IT, Supply chain management while empowered by huge consortiums and communities worldwide as mentioned above.

### 3.5.3 Expected benefits for the Pledger system

Blockchain technology delivers important and meaningful benefits to the Pledger project:

- ▶ exploitation of standard and reliable blockchain platforms
- ▶ coupling of SLA contracts and blockchain smart contracts
- ▶ automatically arranged, programmed and deployed smart contracts
- ▶ user data privacy and security through permissioned blockchain properties
- ▶ fast and efficiently automated edge node communications

In particular, the new technological framework will:

- ▶ explore and face the research challenges arising from supporting the edge and cloud environments and services with blockchain technology capabilities
- ▶ realize a library of methods for creating and distributing smart contracts and decentralized applications
- ▶ aid edge nodes to arrange the resource user’s service delivery level
- ▶ efficiently contribute to necessary pairing of SLA terms and smart contracts in order to for the latter to be deployed and executed on the Pledger blockchain

### 3.5.4 SWOT Analysis

Table 7 summarizes the Strength, Weakness, Opportunities, Threats of the Microtransactions Framework for the edge technology that have been identified so far.

**Table 7: Microtransactions Framework for the edge SWOT table**

Strengths	
▶ compatibility with SLA contracts	▶ rapid communication with automated transactions execution instead of user-managed edge nodes

<sup>113</sup> <https://media.consensys.net/scaling-consensus-for-enterprise-explaining-the-ibft-algorithm-ba86182ea668>

<ul style="list-style-type: none"> <li>▶ secure edge node authentication (identity validation through immutable blockchain transactions)</li> <li>▶ integration with cloud frameworks</li> </ul>
<b>Weaknesses</b>
<ul style="list-style-type: none"> <li>▶ new research for new custom solution</li> <li>▶ complex technology from beginner developers' point of view</li> </ul>
<b>Opportunities</b>
<ul style="list-style-type: none"> <li>▶ custom technological extension:</li> <li>▶ data protection on the edge with blockchain on the back-end</li> <li>▶ automated solution with machine-managed edge nodes</li> <li>▶ cumulative research field</li> </ul>
<b>Threats</b>
<ul style="list-style-type: none"> <li>▶ promising infrastructure; fear of under-delivery</li> </ul>

## 3.6 Benchmarking

Benchmarking is a widely adopted technique to assess performance of computing systems since the beginning of computer age. The essence of benchmarking is to stress a System Under Test (SUT) with a known workload and observe the response of the system measuring one (or more) quality metrics of interest (e.g. I/O throughput, computation latency) that allow to understand "how well" the system can execute the workload. In contrast with other approaches like prediction and simulation, benchmarking allows to test the real system, taking into account all the factors that have an impact on the performance of the workload. Data acquired by benchmarking multiple systems can be extremely useful to rank them by performance and to take informed decisions on the most suitable system for a given workload.

Benchmarks are very specialized tools and their evolution followed closely the evolution of computing architectures: from the first benchmarks developed in 1970s to assess the performance of a CPU in floating-point operations, to Grid and HPC benchmarks, to distributed, application-driven benchmarks for the cloud services and, in the last years, for the edge infrastructures. A recent survey (Hong, 2019) (Blesson Varghese, 2020) provides a good timeline of benchmarking development.

### 3.6.1 Cloud Benchmarking

In cloud computing, benchmarking is an important tool for assessing the performance of cloud services. Given the high diversity of cloud providers' offering and the complexity of the applications, it is not trivial to select the best resource for each application component in order to maximize its performance (Borhani, 2014) or obtain the best performance/cost trade-off (Kousiouris, 2017). Moreover, in the cloud, the performance of an application can be highly affected by factors like the underlying physical hardware, co-location, and scheduling that are not disclosed to the final user by the cloud provider. However, these factors can be taken into consideration in the benchmarking process, because they affect it in the same way they do for the real applications.

In the cloud era, benchmarks evolved to adapt to the increasing complexity and distribution of the applications. The benchmarks that are able to test only a single resource (e.g. CPU, network), often referred to as micro, synthetic or system-level benchmarks, have been replaced by application-driven (or macro) benchmarks that are able to generate workloads similar to the ones of the real applications (Muller, 2014) (Luo C. Z., 2012). This approach guarantees more realistic predictions of the real application performance.

A (not exhaustive) list of application-driven benchmarking tools for the cloud are:

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	55 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

- ▶ YCSB<sup>114</sup> (BF Cooper, 2010) is a benchmark released by Yahoo! In 2010 for testing database performance. It has been extended by YCSB++ (Patil, 2011) to test specific features of scalable table stores;
- ▶ FileBench<sup>115</sup> is an extremely flexible benchmarking tool for the file-system that allows to define custom workloads through the Workload Model Language (WML). It comes with a set of predefined workloads to simulate mail servers, video streaming servers, web servers;
- ▶ TechEmpower's FrameworkBenchmarks<sup>116</sup> is a framework to test web frameworks through seven operations (JSON serialization, single database query, multiple database queries, database updates, caching, plain text processing, html rendering). A public website<sup>117</sup> is available with results of more than 350 different frameworks tested;
- ▶ CloudRank-D (Luo C. Z., 2012) is a suite of benchmarks used to evaluate private cloud systems for running data processing applications. It proposes two complementary metrics for evaluating the cloud systems: data processed per second and data processed per Joule;
- ▶ CloudSuite<sup>118</sup> (Ferdman, 2012) is a suite that collects some of the most popular cloud workloads. The latest release (i.e., 3.0) consists of eight applications based on real-world software stacks and setups (e.g. media streaming, data caching, web search). It uses Docker to deploy the distributed workloads on multiple VMs and execute the tests;
- ▶ OLTP-Bench<sup>119</sup> (Difallah, 2013) is a suite of benchmarks to test JDBC-enabled relational databases on the cloud. It (a) integrates workloads from other benchmarks like YCSB, TCP-C<sup>120</sup>, LinkBench (Armstrong, 2013) and (b) define some new workloads based on popular applications (e.g. Twitter, Wikipedia);
- ▶ PALMScloud (Hao Wu, 2016) is a suite of benchmarks that simulates workloads for the most common type of cloud servers (e.g. Web Server, Mail Server, Streaming Server);
- ▶ SPEC Cloud IaaS 2018<sup>121</sup> is the latest version of the benchmark developed by The Standard Performance Evaluation Corporation (SPEC) to test cloud resources;
- ▶ DeathStarBench<sup>122</sup> (Gan, 2019) is a benchmark suite for cloud microservices. It provides six end-to-end microservice-based benchmarks: Social Network, Media Service, Hotel Reservation, E-commerce site, Banking System and Drone coordination system.

Analysing the characteristics of the most recent benchmarks available, the tendency is to build benchmarks that try to replicate as much as possible real applications: simulating end-to-end applications (e.g. DeathStarBench) and using real datasets (e.g. Twitter, Wikipedia). This, of course, has an impact on the complexity and portability of the benchmark. It also increases its configuration effort and running time and cost. This is, partially, mitigated by the tendency of using containerization technologies (e.g. DeathStarBench, CloudSuite).

---

<sup>114</sup> <https://github.com/brianfrankcooper/YCSB>

<sup>115</sup> <https://github.com/filebench/filebench/wiki>

<sup>116</sup> <https://github.com/TechEmpower/FrameworkBenchmarks/wiki/Project-Information-Framework-Tests-Overview>

<sup>117</sup> <https://www.techempower.com/benchmarks/>

<sup>118</sup> <https://github.com/parsa-epfl/cloudsuite>

<sup>119</sup> <https://github.com/oltpbenchmark/oltpbench>

<sup>120</sup> <http://www.tpc.org/tpcc/default5.asp>

<sup>121</sup> [https://www.spec.org/cloud\\_iaas2018/docs/UserGuide.html](https://www.spec.org/cloud_iaas2018/docs/UserGuide.html)

<sup>122</sup> <https://github.com/delimitrou/DeathStarBench>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	56 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

Finally, on top of the benchmarking tools, few orchestrator solutions have been developed like Perfkit Benchmark<sup>123</sup> and the Benchmarking Suite<sup>124</sup> to automate the scheduling and execution of tests and the storage of the results that would be a time consuming and error prone activity if done manually.

### 3.6.2 Edge Benchmarking

When shifting from the cloud to the edge computing, benchmarking becomes even more valuable because of the high diversity and variability of edge nodes in terms of hardware, energy consumption, connectivity capabilities, software stacks and configuration. All these factors can highly affect the performance of the workloads. In order to optimize the deployment of an application in this harsh environment, very precise and updated information on the resources capabilities and performance must be provided.

The knowledge and techniques acquired in cloud computing benchmarking can be efficiently extended to the edge (and to the cloud-Edge connection), but new challenges arises from the diversity of the two paradigms (Hong, 2019).

Firstly, the limited capabilities of edge nodes need more lightweight benchmarking tools that can run without congesting the resources.

Secondly, the range of hardware is much wider than in the cloud: Internet-Of-Things sensors, smartphones, wearable devices, routers, switches, gateways, single board computers (e.g. Raspberry Pi). Also specialized hardware accelerators to run deep neural networks at the Edge exists like the Google Edge TPU and the NVIDIA Maxwell (Reuther, 2019). This diversity of the hardware makes it difficult to create generic and comparable benchmarks.

Finally, the workloads usually adopted in the cloud benchmarking (e.g. data analysis, data searching, video streaming) do not represent the emerging use cases for the edge computing (e.g. virtual reality, image processing). Also, new metrics should be defined that are relevant for the edge workloads. For instance, since many IoT devices run on battery, power consumed by a workload is a relevant metric in this context (Xiao, 2017).

These challenges are being tackled by different initiatives with ongoing works to (a) characterize the most common edge use cases and their workloads (K. Toczé, 2019) (b) select metrics to define their performance (K. Toczé, 2019) (Xiao, 2017) and (c) implement new suites of benchmarking tools designed to run in the Edge.

Results of these works can be summarized (Hong, 2019) in the following (not exhaustive) list of State-Of-The-Art tools:

- ▶ EdgeBench<sup>125</sup> (A. Das, 2018) is a benchmark for serverless edge computing targeting AWS Greengrass and Azure IoT edge platforms. It contains of six different workloads in the domain of speech decoding, image recognition and face detection;
- ▶ AIBench<sup>126</sup> (Gao, 2018) is a benchmarking framework from the BenchCouncil that covers sixteen problem domains: image classification, image generation, text-to-text translation, image-to-text, image-to-image, speech-to-text, face embedding, 3D face recognition, object detection, video prediction, image compression, recommendation, 3D object reconstruction, text summarization, spatial transformer, and learning to rank;

<sup>123</sup> <https://github.com/GoogleCloudPlatform/PerfKitBenchmarker>

<sup>124</sup> <https://benchmarking-suite.readthedocs.io/en/latest/>

<sup>125</sup> <https://github.com/akaanirban/edgebench>

<sup>126</sup> <http://www.benchcouncil.org/AIBench/index.html>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	57 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

- ▶ AIoTBench<sup>127</sup> (Luo C. &, 2019) is a suite of benchmarks that includes image classification, speech recognition and language translation workloads. It also includes some micro workloads for basic operations on neural networks;
- ▶ EdgeAIBench<sup>128</sup> (Tianshu Hao, 2019) is a benchmark from the BenchCouncil for end-to-end edge computing applications including four typical application scenarios: ICU Patient Monitor, Surveillance Camera, Smart Home and Autonomous Vehicle;
- ▶ IoTBench (C. Lee, 2019) is a benchmarking suite targeting at IoT edge-device applications. It includes seven representative benchmarks (Video summarization, Stereo image matching, Image recognition, Scan matching, Voice feature extraction, Signals enhancement, Data compression). It analyses the computational demand and execution efficiency of the executions;
- ▶ MAVBench (Behzad Boroujerdian, 2019) is benchmark suite, the first of its kind, consisting of a variety of Micro Aerial Vehicles (MAV) applications like Scanning, Aerial photography, Package delivery, 3D mapping, Search and rescue;
- ▶ EdgeDroid<sup>129</sup> (Olguín, 2019) is a benchmarking framework to evaluate human-in-the-loop applications;
- ▶ MLPerf<sup>130</sup> (Vijay Janapa Reddi, 2019) is an effort from several organizations (e.g. Google, NVIDIA, Cisco, ARM, Berkeley Lab) to provides fair and useful benchmarks for measuring machine learning training and inference performance.

Edge computing is a fast-growing research topic and the number of research works and proposed benchmarks is rapidly increasing to cover new hardware, software and use cases. However, most of benchmarks available at the moment are still at research stage and not easily portable to execution environments different from the one they have been created for. Also, scalability to real-world infrastructures and standardization of workloads and metrics are to be addressed by future research works (Blesson Varghese, 2020).

### 3.6.3 Expected benefits for the Pledger system

Pledger aims at offering a benchmarking service for cloud and edge resources with the objective of supporting decisions for resources selection at deploy time and monitoring of the resource's performance.

Pledger benefits from the benchmarking are mainly related to the optimization of the deployment: benchmarking will be able to provide accurate predictions on the performance of application for different deployment plans. This will make it possible to select the deployment plan that provides the best performance (or the best cost/performance trade-off) for the application. In addition, benchmarking will provide useful hints for potential re-deployments of the applications.

In order for this to be possible, the benchmarking service in Pledger should:

- ▶ provide metrics meaningful and immediately relatable to the UCs applications;
- ▶ provide metrics that can predict the QoS and QoE levels of the UCs applications;
- ▶ provide an interface to access and query metrics in a standard format from other Pledger components (e.g. QoE Assessment, Decision Support System);
- ▶ provide always fresh and updated metrics due to the frequency with which resources (mostly at the Edge) can change;

---

<sup>127</sup> [www.benchcouncil.org/AIoT/Bench](http://www.benchcouncil.org/AIoT/Bench)

<sup>128</sup> <http://www.benchcouncil.org/EdgeAIBench/index.html>

<sup>129</sup> <https://github.com/molguin92/EdgeDroid>

<sup>130</sup> <https://mlperf.org/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	58 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

- ▶ provide metrics for security of cloud/edge resources in order to support the security mechanisms in Pledger.

In order to satisfy all these requirements, we foresee in Pledger some possible improvements to the current State Of The Art of benchmarking in the cloud and edge.

Concerning (1) and (2), we start from the statement that the more the benchmark is close (in terms of generated workload) to the real application, the more the metrics provided are relatable to the real application and usable to predict its performance. The Pledger use cases (at the current stage of development) will use applications in the domain of the augmented reality, object recognition and data streaming/pipelining. A good set of benchmarks for these domains already exist at the State Of The Art and we will employ them, making improvements and adaptations to fit the use case applications needs, if necessary. In case no tests are suitable to simulate a use case component, we will develop completely new tests to evaluate its performance.

Concerning (5), we did not find a reasonable set of security benchmarks in the State Of The Art, especially for edge resources. Considering the specificity and the innovation in the security solution foreseen in Pledger, we expect to develop specific benchmarks shaped around the security mechanisms that will be developed by Pledger.

Finally, to satisfy requirements (3) and (4) we will advance a tool at the State Of The Art, the Benchmarking Suite (asset brought by ENG in Pledger). The Benchmarking Suite already provides a benchmarking scheduling functionality and the storage of metrics in a database. In Pledger, the scheduling functionality will be revised to fits the project needs and an interface to access collected metrics exposing its data in a standard way will be developed (timeseries databases commonly used for metrics will be considered - e.g. Prometheus).

### 3.6.4 SWOT Analysis

Table 8 represents a SWOT analysis to evaluate the current plan for providing the benchmarking service in Pledger evolving the current Benchmarking Suite asset.

Table 8: Benchmarking SWOT analysis

Strengths	
<ul style="list-style-type: none"> <li>▶ easily extensible integrating new benchmarking tools</li> <li>▶ automated management of the entire benchmarking process</li> <li>▶ declarative configuration</li> </ul>	
Weaknesses	
<ul style="list-style-type: none"> <li>▶ benchmarks and metrics must be carefully selected to be representative of a given application</li> <li>▶ no edge specific benchmarks are integrated at the moment</li> </ul>	
Opportunities	
<ul style="list-style-type: none"> <li>▶ Use Case partners can support in the development of new benchmarks and in the definition of new metrics relevant for their domain</li> <li>▶ Edge benchmarking is a very active research topic and we can effectively contribute to its advancement</li> </ul>	
Threats	
<ul style="list-style-type: none"> <li>▶ Use case applications could be difficult to decompose and classify in benchmarking categories</li> <li>▶ Edge benchmarking is still a recent research topic and most of the technology produced has low maturity, portability and robustness</li> </ul>	

## 3.7 Cloud and edge SLA monitoring and evaluation

This section is dedicated for the analysis of the state-of-the-art landscape for the SLA and QoS research and solutions, both for cloud and edge computing. SLA monitoring is a process that requires to extract the SLA parameters from the contract and monitor them throughout a specific period of time and evaluate if the contract is not breached by the service provider. Although the parameters of a contract may be similar there are different approaches taken in the SLA monitoring of the cloud and Edge. In this section both cloud and edge solutions are presented.

SLA evaluations as well as tools that evaluate the services in different way (i.e. benchmarking) are the main input of the quality of service evaluation. In this manner this section also analyses the QoS solutions offered.

### 3.7.1 SLA/QoS monitoring and evaluation in cloud

In general, there is a variety of monitoring tools available<sup>131, 132</sup>, but the main issue is that, in all these cases, there is a lack of adaptation to the specific and diverse SLA definitions (in terms of preconditions, calculation formulas, etc.) of the public and more importantly private cloud and edge SLA realm.

The SeICSP framework is presented in (Ghosh, Ghosh, & Das, 2015), which highlights differences in cloud SLAs, including for the same type of services, and assists users in selecting providers that are best suited to their needs. Rating is extracted on the basis of user feedback from past experience and the transparency of the respective SLAs. SMICloud is presented in (Garg, Versteeg, & Buyya, 2011) to rank cloud services from a variety of perspectives. This is not directly related to SLA monitoring, but this aspect is taken into account in the ranking process, mainly in the form of the provider's probability of maintaining the SLA.

One very interesting approach to monitoring information on availability of services in real time is Cloudsleuth<sup>133</sup>. Cloudsleuth has an extensive network of applications deployed in a number of providers and locations and continuously monitors their performance in terms of response times and availability metrics. The measurement method of Cloudsleuth is not adapted to the SLA definition of each provider, so it cannot be used to claim compensation. It also checks the response of a web server (return type status 200 on a GET request). Thus, it cannot distinguish between a failure due to an unavailable VM (provider liability) or a crash of an application server (customer liability in the case of IaaS deployment, provider liability in the case of PaaS/SaaS) or a pure application fault (customer liability). On the other hand, it follows a normal definition of availability that makes it possible to compare services from different providers, a process that cannot be carried out in accordance with their specific SLA definitions, since they differ in the way they define availability. In addition, it appears that CloudSleuth has not been publicly available in recent months.

CloudHarmony<sup>134</sup> is another effort to provide availability measurements and to report unavailability intervals, but it is not clear how the overall level is measured (depending on the definition of each provider or not) and whether the deployment of the measurement service is consistent with the SLA of the specific provider. Furthermore, it cannot be used for claims made by a single user against a provider, since it refers to and monitors a specific test service on the website and not an arbitrary user account. In fact, therefore, it monitors only a very specific subset of the overall data center.

<sup>131</sup> <https://www.softwaretestinghelp.com/cloud-monitoring-tools/>

<sup>132</sup> <http://www.cloudbook.net/resources/stories/accurately-monitoring-cloud-slAs>

<sup>133</sup> <https://cloudsleuth.net/>

<sup>134</sup> <http://cloudharmony.com>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	60 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

Site24x7<sup>135</sup> provides SLA management tools as an arbitration between Service Providers and Application Owners, but it tends again to be focused on high-level availability of software, which are not suited to the different interpretations of a cloud service. Furthermore, it is not very clear what happens to more complex models, such as the common case in many public cloud resources, such as GAE or Azure, where the error rate is calculated, e.g. in hourly intervals, where the error must be over a given time period and with different response time limits per type of operation. Bitnami<sup>136</sup> offers tools for cloud management and monitoring, but applies mainly to applications for the use of existing provider services, such as AWS Cloudwatch. Uptime Cloud Monitor<sup>137</sup> is a commercial cloud monitoring company, providing customizable dashboards and a fascinating set of indicators used to produce warnings and updates for users. CloudMonix<sup>138</sup> is a related service. Support seems to be limited to Microsoft Azure at this point. Netcraft<sup>139</sup> is a company that monitors the output of hosting environments from a particular point of view (e.g. networking tests every 15 minutes per host) at a fixed annual price. Again, this sampling technique can be used to compare different services, but does not conform to specific definitions of SLAs in public cloud environments.

Akamai's Cloud Monitor<sup>140</sup> is a real-time, push API service that delivers essential transaction data from Akamai's Intelligent Platform. Cloud Monitor provides real-time visibility into a wide range of data captured-including basic transaction details, network and client performance metrics, security alerts, network statistics, location data, and cookie information. Although fascinating as a source of information, the data acquired would still have to be analyzed and measured on the basis of the specific SLA concept of each provider. Zabbix<sup>141</sup> is an open source platform that can be used to track IT infrastructures. It comes with a set of interesting features, such as linking to specific services and nodes, building chains of resource dependence, etc. The most interesting aspect is that the definition of SLA can be more precise than the other cases of publicly available tools (including more detailed conditions on the failed study, although without the complete versatility for the consumer to specify a specific formula), but it seems more fitting for the provider to self-monitor the services.

### 3.7.2 SLA/QoS monitoring on edge

Edge monitoring is tightly coupled with the network layer as depicted in Figure 23, resulting in most of the tools for edge QoS being network monitoring tools. There is a plethora of monitoring tools for the network. An edge computing platform itself is an application agnostic technology in that it is not dedicated to a single type of software system or function. Although a significant number of research works consider the stability of the underlying cloud infrastructure, there is still a lack of effective application-level control techniques to identify and quantify QoS and by extend SLA breaches of contract (Taherizadeh, 2018).

---

<sup>135</sup> <https://www.site24x7.com/sla-management.html>

<sup>136</sup> <https://bitnami.com/tools/cloud>

<sup>137</sup> <https://www.idera.com/infrastructure-monitoring-As-A-Service>

<sup>138</sup> <http://cloudmonix.com/>

<sup>139</sup> <http://uptime.netcraft.com/perf/reports/Hosters>

<sup>140</sup> <https://www.akamai.com/>

<sup>141</sup> <https://www.zabbix.com>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	61 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

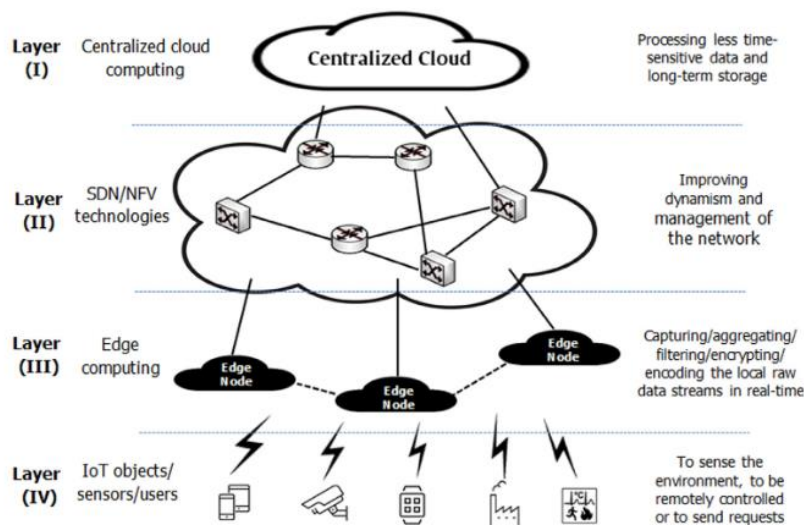


Figure 23: Basic paradigm of an edge computing framework (Taherizadeh)

Although there are no tools dedicated only for the edge QoS monitoring there are several Cloud based and network monitoring tools capable of working with edge-based solutions.

CloudScale (Shen, 2011) which measures the distributed application's performance at runtime and also adopts user-specified scaling policies for provisioning and de-provisioning of virtual resources. Their proposed event-based approach models the workload behaviour supports multi-dimensional analysis, and defines the adaptation action.

CASViD (Emeakaroha, 2011) is a general purpose that facilitates the calculation of low-level device metrics such as CPU and memory consumption, as well as high-level application metrics that depend on the application type and performance. The results suggest that CASViD, which is built on a non-intrusive architecture, can establish an appropriate measurement interval for measuring different metrics. It can give efficient intervals for the calculation of different metrics for specific workloads.

VPerfGuard (Xiong, 2013) is a model-driven system designed to achieve automatic adaptive application performance. This methodology, which is able to identify performance inefficiencies, has three modules:

- ▶ A sensor element for the selection of runtime system metrics along with application-level metrics.
- ▶ A model construction factor that allows the development of a personalized performance.
- ▶ A model that demonstrates the connection between system metrics and application efficiency.

An effort to monitor the performance guaranties based on a Runtime Model for Cloud Monitoring (RMCM) has been also established (Shao, 2011). In this research, a performance model is built on runtime tracked data using a linear regression algorithm. Such related measurements include the resource assigned to the Server where the application lives, the amount of co-existing applications on the same VM, the real resource role of the client, the workload etc. The results indicate that the performance model can be successful in guiding the provisioning approach to the achievement of defined performance targets.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	62 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
		<b>Version:</b>	2.0
		<b>Status:</b>	Final

### 3.7.3 Expected benefits for the Pledger system

#### 3.7.3.1 SLA Monitoring and Evaluation

One of the tools that are introduced in the Pledger for the SLA monitoring and evaluation is 3ALib<sup>142</sup>. One of the main features of 3ALib, which allows it to differentiate itself from competition, is that it completely adapts to various provider requirements, including the required implementation preconditions for testing or overall considerations when evaluating the operating cloud service. For example, if a service consists of several VMs or components, a traditional typical monitoring tool will provide statistics for each component. However, from an SLA point of view, even if all the cloud services used are not accessible at the same time can a particular sample be considered inaccessible (at the level of the cloud region). Thus, the logs kept by 3ALib may in effect be used for reimbursement, while the respective logs of the conventional tools may not be used.

#### 3.7.3.2 QoS Monitoring and Evaluation

QoE Entity is a complete web-based UI with back-end storage to enfold all the features required to construct a full cloud-based analysis service. QoE introduces all the data obtained from the software (benchmarking package, 3ALib SLA monitoring) to users. Presenting information is one of the two key features of QoE, the other being the ability to monitor tools from the QoE UI. Using Docker Secrets and REST Full Services, QoE is able to control (Start, Stop, Create) the Benchmarking Suite and 3ALib. One of the main features of the QoE is the uniform metric it generates to evaluate the SLAs of the various IaaS providers and to evaluate the performance of the Cloud Services.

### 3.7.4 SWOT Analysis

In the latest years small and bigger enterprises have been adapting rapidly cloud services for their business. 2019 RightScale cloud report<sup>143</sup> shows an increase on the maturity of cloud adoption (Figure 24) by the enterprises. This is no surprise given the maturity level and the stability cloud services have accomplished.

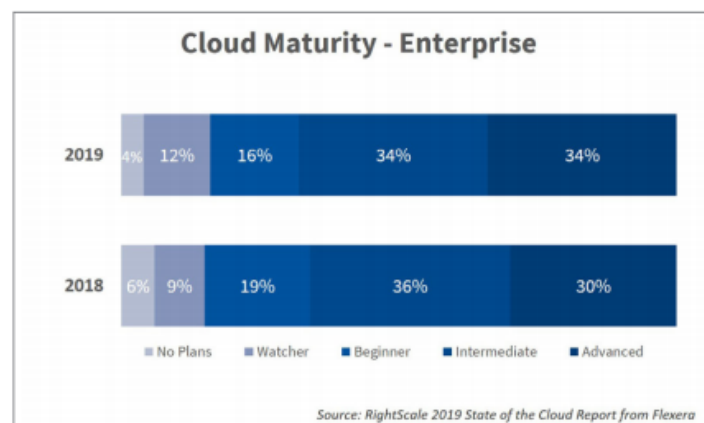


Figure 24: RightScale 2019 cloud maturity in enterprises

Given the increase on the level of maturity of cloud adaptation, enterprises have been shifting from public to private cloud solutions (Figure 25). Private cloud can provide a better quality of service and also the services can have more advanced and stricter SLA's. Private cloud adaptation creates new opportunities for the tools that monitor and evaluate cloud SLAs. Given the complexity of the private

<sup>142</sup> <https://github.com/cloudperfect-project/CloudPerfect/wiki/3ALib-Agent-Usage>

<sup>143</sup> <https://resources.flexera.com/web/media/documents/rightscale-2019-state-of-the-cloud-report-from-flexera.pdf>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	63 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

contracts tools that are able to monitor and evaluate accordingly the appropriate QoS parameters can be of a great importance for the business continuity of all the enterprise types.

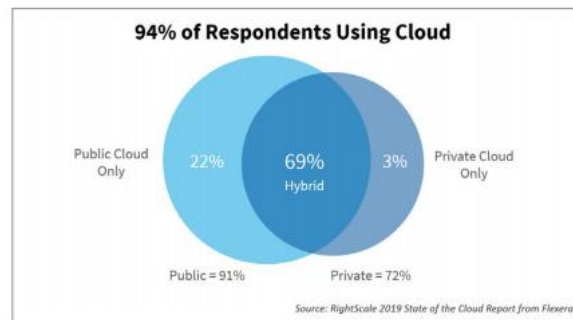


Figure 25: Public and private cloud usage

Table 9 encapsulates the Strengths, Weaknesses, Opportunities, and Threats of the SLA and QoS monitoring and evaluation technologies that have been identified so far.

Table 9: SLA and QoS evaluation SWOT table

Strengths
<ul style="list-style-type: none"> <li>▶ Adaptation to various cloud provider’s requirements</li> <li>▶ Creation of custom SLA for private Clouds</li> <li>▶ Abstraction Libraries for monitoring different type of public and private cloud</li> <li>▶ Standardized Public Cloud provider SLA strictness assessment</li> <li>▶ Highly scalable and easy to deploy due to containerized form</li> </ul>
Weaknesses
<ul style="list-style-type: none"> <li>▶ Difficult to operate as a standalone tool</li> <li>▶ Not automated way of introducing new agents for new public or private providers</li> </ul>
Opportunities
<ul style="list-style-type: none"> <li>▶ Emerging private cloud services can create new opportunities in the SLA development and monitoring</li> <li>▶ Create SLA contracts in form of smart contracts, in order to create a unified commonly accessible platform for SLA negotiation and monitoring</li> </ul>
Threats
<ul style="list-style-type: none"> <li>▶ Great heterogeneity on public and private cloud SLA definition</li> <li>▶ New emerging software and solutions</li> </ul>

## 3.8 Monitoring applications and infrastructures in Fog and Edge

Monitoring multiple applications distributed in the heterogeneous ecosystem proposed in Pledger will pose a great challenge. To ensure that applications Quality of Service (QoS) requirements are met during runtime, it will be necessary to use an adaptable and comprehensive monitoring system, able to be used in multiple and different Cloud, Fog and edge devices to monitor the infrastructures and the applications running on them. Thus, it will be required the use of a monitoring and SLA management tools stack, that will collect the information needed for the proper functioning of the SLAs managed by the Pledger framework.

This section describes the current State of the Art technologies for SLAs and QoS monitoring of services and infrastructures in Cloud, Fog and Edge, and the advantages they can bring to Pledger.

### 3.8.1 Monitoring tools

There are many available monitoring solutions, mainly in the Cloud and Fog ecosystem. We can find mature applications used by the different main cloud providers, which have been improved during the last years and are now offered as integrated features by these cloud providers. This includes the Service Level Agreements (SLA) management and QoS monitoring of the applications running in these environments. One inconvenient is that most of these providers offer their cloud-specific and proprietary monitoring services, such as Amazon CloudWatch<sup>144</sup>, Microsoft Azure Monitor<sup>145</sup>, Alibaba CloudMonitor<sup>146</sup>, and (Google partner) RackSpace Monitor<sup>147</sup>. This is something that also applies to other commercial platforms like VMWare vCloud Suite<sup>148</sup> and Stratoscale<sup>149</sup>, and to opensource cloud computing platforms like Openstack, which offers their own stack of tools (Telemetry services<sup>150</sup>) to monitor the services.

Furthermore, in the case of the cloud providers, like Remote Monitoring Azure IoT solution<sup>151</sup> offered by Azure platform, or AWS IoT<sup>152</sup> offered by Amazon, they usually include IoT management in their offerings. And this includes the monitoring of the applications running in these edge devices. But as stated before, this kind of solutions present the same problem, they are proprietary and specific for each provider.

Apart from the tools offered by cloud providers or platforms, we can also find commercial monitoring solutions like Datadog<sup>153</sup>. This solution offers a unified monitoring system for metrics, traces and logs, including the SLAs and their automatization, which is compatible with many different cloud providers like AWS and Azure, and other solutions like Kubernetes. In addition to Datadog, we can find opensource monitoring solutions like JCatascopia<sup>154</sup>, which is able to monitor adaptive cloud applications running in different environments.

Focusing on the required Fog and edge scenarios, there are widely used standalone monitoring solutions like Prometheus, Scout, Ganglia<sup>155</sup>, Zabbix, Nagios, Collectd, and cAdvisor, among many others.

<sup>144</sup> [https://aws.amazon.com/cloudwatch/?nc1=h\\_ls](https://aws.amazon.com/cloudwatch/?nc1=h_ls)

<sup>145</sup> <https://azure.microsoft.com/en-ca/services/monitor/#product-overview>

<sup>146</sup> <https://www.alibabacloud.com/product/cloud-monitor>

<sup>147</sup> <https://www.rackspace.com/managed-google-cloud>

<sup>148</sup> <https://www.vmware.com/products/vcloud-suite.html>

<sup>149</sup> <https://www.stratoscale.com/products-services/>

<sup>150</sup> <https://wiki.openstack.org/wiki/Telemetry>

<sup>151</sup> <https://docs.microsoft.com/en-us/azure/iot-accelerators/quickstart-remote-monitoring-deploy>

<sup>152</sup> <https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html>

<sup>153</sup> <https://www.datadoghq.com/>

<sup>154</sup> <http://linc.ucy.ac.cy/CELAR/jcatascopia/>

<sup>155</sup> <http://ganglia.sourceforge.net/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	65 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

Prometheus<sup>156</sup> is a free monitoring application which collects, stores and manages metrics coming from different targets or other Prometheus instances. This tool is easy to extend and integrate with other tools like Grafana or platforms like Kubernetes, but it presents some problems in large deployments. In fact, most of these tools present the same problems in large-scale deployments. Nagios Core<sup>157</sup> is another open source monitoring tool that can monitor infrastructures, networks and applications. Configuration and a lack of some of the features already presented in the commercial version are some of its drawbacks. One more example of the wide range of monitoring tools available is Collectd<sup>158</sup>, which is a daemon application responsible for periodically gathering performance metrics from applications and systems. Collect can work together with tools like InfluxDB or other external systems to store these gathered metric values. On one hand Collectd can be a particular problem when configuring it, but on the other hand Collectd comes together with a set of plugins that can be enabled and configured, and it also allows the use of custom plugins which enhance the monitoring capabilities of this tool. These plugins can be written in programming languages like Java, C or Python, among many others.

In summary, although many of these tools offer a lot of features and capabilities that can be very useful for the SLA and QoS monitoring in Fog and Edge, they generally present some drawbacks, like the automatic configuration, the integration with other tools, the use of a central server or the adaptability to dynamic contexts.

### 3.8.2 Expected benefits for the Pledger system

Pledger aims to integrate and expand existing opensource monitoring and SLA management tools into a unified framework responsible for monitoring QoS metrics of the infrastructures and applications targeted by Pledger applications. This will provide the following benefits:

- ▶ One common stack of tools responsible for monitoring QoS metrics of infrastructures and applications in Cloud, Fog and Edge.
- ▶ The SLA Lite application, one of the SLA tools introduced in Pledger, relies on this kind of monitoring tools, and is prepared to be adapted to external tools like the ones described.

### 3.8.3 SWOT Analysis

This section summarizes the identified Strengths, Weaknesses, Opportunities and Threats of the proposed monitoring framework.

Table 10: Monitoring applications and infrastructures in Fog and Edge SWOT

Strengths
<ul style="list-style-type: none"> <li>▶ Use of open source tools supported by a big community</li> <li>▶ Production-ready monitoring tools</li> <li>▶ Easy integration (REST APIs) with applications introduced to Pledger (SLA Lite)</li> <li>▶ Monitoring of infrastructures and applications in Cloud, Fog and Edge</li> </ul>
Weaknesses
<ul style="list-style-type: none"> <li>▶ Automatic deployment and configuration</li> <li>▶ Scalability in large deployments</li> <li>▶ Integration of multiple tools in one common stack</li> </ul>
Opportunities
<ul style="list-style-type: none"> <li>▶ Custom solution for Pledger requirements</li> </ul>

<sup>156</sup> <https://prometheus.io/>

<sup>157</sup> <https://www.nagios.com/>

<sup>158</sup> <https://collectd.org/>

▶ Use of state-of-the-art tools
<b>Threats</b>
▶ Heterogeneous environment with different O.S. and capabilities

## 3.9 AR Streaming on the Edge

### 3.9.1 Overview

In the Pledger project, we will investigate new modalities for MR and AR applications as part of a multilevel offloading scheme that will include infrastructure edge nodes and more powerful cloud nodes. While resources on the infrastructure edge are further away (physically and logically) from users than resources on the device edge, they are still close enough to provide low round-trip latencies (5-10 ms) to most devices and they are also capable of housing equipment that is orders of magnitude more powerful than what exists on the device edge. The infrastructure edge can be viewed as a mid-point between the device edge and the traditional centralised cloud, aiming to combine the advantages of both. Moreover, the MR application deployed in the respective use case of Pledger will adopt an innovative privacy-preserving mechanism of visualising content on the edge while filtering out any content that should be no further transmitted on the backend edge/cloud infrastructure.

### 3.9.2 Fraunhofer Instant3Dhub Service

Instant3Dhub<sup>159</sup> is a new, intelligent software platform for using “visual computing as a service” solutions in customer environments. Its focus is on end-to-end applications that can be implemented on site, in the cloud and in hybrid environments.

Instant3Dhub uses a hybrid approach where 3d content which is in your field of view is streamed to the local AR device and gets rendered. This means, a user still has its own UI locally running at the AR Device but can additionally stream the 3D content.

### 3.9.3 Microsoft Azure Remote Rendering Service

Microsoft Azure Remote Rendering Service<sup>160</sup> brings remote rendered high quality 3D content and interactive experiences to mixed reality devices, such as HoloLens 2. This service uses a hybrid approach where an additional layer of texture is added to the local device. This means, a user still has its own UI locally running at Microsoft HoloLens but can additionally display an 3D content on top.

### 3.9.4 Holo-Light Multiplatform ISAR

Holo-Light’s remote rendering is based on a 100% cloud approach. This means the complete app is running on an external machine. The local AR Device only acts as a mobile SLAM upstream machine and smart display the downstream.

Holo-Light’s ISAR is divided in two parts, process and stream of "Rendered" views of a 3D Scene generated on a powerful GPU based on a user applications "down" to the AR Device. The Remote Rendering Service access this Data at GPU Level. The rendered views are 32\_Bit Data (Red, Green, Blue, Alpha) per Pixel per Image. (e.g. two times 720p Images for HoloLens per Frame ~ 5 - 10 MBit/s)

<sup>159</sup> <https://www.igd.fraunhofer.de/en/projects/instant3dhub>

<sup>160</sup> <https://azure.microsoft.com/en-us/services/remote-rendering/>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	67 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

The AR Device is streaming spatial Data. This means Position & Rotation from the AR Device in reference to the Room Frame of Reference. This set of Data (~5 KBit/s) is sent from the AR Device to the GPU, in order to compute the View Matrix for the next View to render the next pair of images.

### 3.9.5 Expected benefits for the Pledger system

Remote Rendering in general shifts the complexity from the end-user away to a more controllable cloud/edge environment. End-users only have smart displays, where they can interact, and the necessary computing performance comes from the backend.

### 3.9.6 SWOT analysis

This section summarizes the identified Strengths, Weaknesses, Opportunities and Threats.

Table 11: Streaming on the Edge SWOT

Strengths
<ul style="list-style-type: none"> <li>▶ High-Performance Rendering on mobile AR Devices</li> <li>▶ Ability to render high amount of Data</li> <li>▶ Unified Experience across all supported devices; one App development for multiple devices</li> <li>▶ high availability</li> <li>▶ high scalability</li> </ul>
Weaknesses
<ul style="list-style-type: none"> <li>▶ Not much support for GPUs</li> <li>▶ latency sensitive</li> <li>▶ bandwidth sensitive</li> <li>▶ Only Windows 10 Support</li> </ul>
Opportunities
<ul style="list-style-type: none"> <li>▶ Support Pledger UC scenarios</li> </ul>
Threats
<ul style="list-style-type: none"> <li>▶ Weak support for Windows 10 VM with render GPUs</li> <li>▶ Availability of low latency edge Infrastructure</li> </ul>

## 4 Requirements' Consolidation and Coding

---

The objective of this section is to provide a consolidated view of the Pledger requirements and provide a formal coding to facilitate the tracking of updates up to the project lifespan.

### 4.1 Approach

---

In Section 2, user stories have been used as the initial step to elicit requirements from the perspective of the stakeholders initially identified in the D2.1. They also synthesize the interviews led by the partners' consortium to their customers and to internal domain experts. Additional sources for the requirements that have been considered are from the State of the Art (including the tools and best practices from the open-source communities), then from the expertise each partner brings to the consortium as solution provider, and finally, from the assets the consortium provides to the project.

To summarize, the **sources** used for the requirements, initially described by the user stories in Section 2, include:

- ▶ the State-of-the-Art analysis in Section 3;
- ▶ the technical expertise from the partners acting as domain experts/solution providers, including interviews from stakeholders and internal business units;
- ▶ the technical expertise related to the assets used in Pledger.

A further categorization is done between functional and non-functional requirements, the former related to the specific behaviour or function expected to be provided by the Pledger system, the latter related to some criteria to be used to judge the operation of the platform. Both have been categorized in order to group requirements by similar properties.

A final subsection is dedicated to the reports about the distribution of the requirements over the different sources and categories identified so far and is meant to help balance the priorities during the MVP voting step (for prioritization) that will be described in Section 5.

### 4.2 Functional Requirements

---

This subsection contains the list of Pledger functional requirements, using the following **categories**:

- ▶ Core ("FU\_CORE"): the relevant functions that are part of the core Pledger system;
- ▶ Application ("FU\_APPL"): the relevant functions that are specific for the Pledger UCs.

The requirements are also labelled with **groups** to facilitate the architecture definition activities in D2.3; the list is the following:

- ▶ use cases ("G\_UCS");
- ▶ platform orchestrator ("G\_ORC");
- ▶ sla ("G\_SLA");
- ▶ blockchain ("G\_BLO");
- ▶ benchmarking ("B\_BEN");
- ▶ security ("G\_SEC");
- ▶ dashboard ("G\_DAS").

The Table 12 reports the list of the functional requirements with references to the related user story, the stakeholders involved, the category and the source where the requirement was extracted.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis			<b>Page:</b>	69 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	<b>Version:</b>	2.0
				<b>Status:</b>	Final

Table 12: Functional requirements

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.01	The Pledger system should allow the update or rollback of a new service with little or no interruption for the users (new version, bug fixed, etc.).		Service Provider, Infrastructure Provider	FU_CORE	G_SLA	expertise from existing asset
FR.02	The Pledger system would schedule automatic scaling of services based on SLA violation as far as the tenant has available resources.		Service Provider	FU_CORE	G_SLA	expertise from existing asset
FR.03	The Pledger system must allow to remove or update part of the offering to service provider based on infrastructure catalogue: new hardware, decommissions, etc.		Service Provider	FU_CORE	G_ORC	expertise from existing asset
FR.04	The Pledger system should allow to choose in which region, geolocation and specific subset of resources (e.g. availability zones) the data can be uploaded from edge to cloud.		Infrastructure Provider	FU_CORE	G_ORC	expertise from existing asset
FR.05	The Pledger system must allow dynamic configurations before and after service deployment.		Infrastructure Provider	FU_CORE	G_ORC	expertise from existing asset
FR.06	The Pledger system must visualize performance statistics about each service in a dashboard.		Service Provider, Infrastructure Provider	FU_CORE	G_SLA	expertise from existing asset
FR.07	The Pledger system must allow to manage service lifecycle (e.g. starting/stopping) via a dashboard.		Service Provider	FU_CORE	G_SLA	expertise from existing asset

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.08	The Pledger system must be able to share selected datastreams to third-party service providers.		Service Provider	FU_CORE	G_SLA	expertise from existing asset
FR.09	The Pledger system should provide an API to share selected datastreams to third parties.		Infrastructure Provider	FU_CORE	G_SLA	expertise from existing asset
FR.10	The Pledger system should be maintained by a CI/CD devops development pipeline where bug fixes and features are automatically deployed frequently without disruption.		Service Provider	FU_CORE	G_SLA, G_ORC	expertise from existing asset
FR.11	The Pledger system could allow user end devices to be registered on the services that implement the VRU safety features.		Service Provider	FU_CORE	G_ORC	expertise as domain expert/solution provider
FR.12	The Pledger system must allow the users to be warned by an audiovisual signal in case a risk is detected.		Service Provider	FU_CORE	G_ORC	expertise as domain expert/solution provider
FR.13	The Pledger system should allow users to share their position with the system.		Service Provider	FU_CORE	G_ORC	expertise as domain expert/solution provider
FR.14	The Pledger system must provide dashboards/GUIs to manage the resources/services assigned to the tenant.	US.19, US.21	System Integrator	FU_CORE	G_ORC	expertise as domain expert/solution provider
FR.15	The Pledger system must be able to process location and sensor inputs to determine risky situations.	US.19, US.20, US.21	End user, Service Provider	FU_CORE	G_SLA, G_BLO	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.16	The Pledger system could be able to detect the urban transport (TRAM) presence by reading its shared location or by detecting it via sensors to identify potential risk situations.	US.20, US.24	End user	FU_CORE	G_BLO	expertise as domain expert/solution provider
FR.17	The Pledger system must allow the configuration of the SLA offering by the infrastructure provider.	US.21	End user	FU_CORE	G_ORC, G_SLA	expertise as domain expert/solution provider
FR.18	The Pledger system could show SLAs details and historical status of SLA violations or fulfilment.		Service Provider	FU_APPL	G_BEN	expertise as domain expert/solution provider
FR.19	The Pledger system could be able to support node communication in privileged networks that are able to exchange important value inside the Pledger blockchain network.	US.11, US.13	Service Provider	FU_CORE	G_BEN	state of the art / best practices
FR.20	The Pledger system must allow the initiation of smart contract deployment from a user-friendly interface, hiding unnecessary details to the developers.		Service Provider	FU_CORE	G_ORC	state of the art / best practices
FR.21	The Pledger system must have user-friendly UI to interact with data stored in the ledger.	US.11, US.12	Service Provider	FU_CORE	G_BEN, G_DAS	expertise from existing asset
FR.22	The Pledger system would detect and identify anomalies, prioritise risks and provide prompt alerts about abnormal security behaviours.		Infrastructure Provider	FU_CORE	G_BEN	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.23	The Pledger system must provide an infrastructure test suite in order to measure and gather performance metrics to classify the infrastructure.		Service Provider	FU_CORE	G_BEN, G_SEC	expertise as domain expert/solution provider
FR.24	The Pledger system must provide customizable dashboards with benchmarking results.		Service Provider, System Integrator	FU_CORE	G_BEN, G_SEC	expertise as domain expert/solution provider
FR.25	The Pledger system should execute a benchmarking test provided by the user.		System Integrator	FU_CORE	G_BEN	expertise from existing asset
FR.26	The Pledger system must provide a dashboard to highlight the different infrastructure nodes and their availability in terms of computation and network resources quotas.		System Integrator	FU_CORE	G_BEN	expertise from existing asset
FR.27	The Pledger system would provide a dashboard to highlight the different infrastructure nodes and their usage in terms of computation and network.		System Integrator	FU_CORE	G_BEN	expertise from existing asset
FR.28	The Pledger system must provide a dashboard to highlight the critical resource allocations.	US.33	Infrastructure Provider	FU_CORE	G_BEN	state of the art / best practices
FR.29	The Pledger system must provide a list of possible activity options with respect to different metrics.	US.32	Service Provider	FU_CORE	G_BEN, G_DAS	state of the art / best practices
FR.30	The Pledger system should have an SLA template repository.		Service Provider	FU_APPL	G_DAS	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.31	The Pledger system could allow the cancelation of SLAs because of some changes, like end of product support, product improvements, discovery of infrastructure incompatibility, etc.		Service Provider	FU_CORE	G_DAS	expertise as domain expert/solution provider
FR.32	The Pledger system would allow the automatic renewal of SLAs in case SLA has begun and end date.		Service Provider	FU_CORE	G_DAS	expertise as domain expert/solution provider
FR.33	The Pledger system could send notifications regarding SLA violations.		Service Provider	FU_APPL	G_DAS	expertise as domain expert/solution provider
FR.34	The Pledger system must allow the sign up of smart contracts of the SLA agreements.		Service Provider	FU_APPL	G_DAS	expertise as domain expert/solution provider
FR.35	The Pledger system must allow the check of smart contract transactions and violations.		Service Provider	FU_APPL	G_DAS	expertise as domain expert/solution provider
FR.36	The Pledger system could be able to provide a trustful network for edge nodes in the context of a permissioned blockchain network where each participant can have enhanced guarantee that the co-participant nodes operate under legal and well-intentioned motives.		Service Provider	FU_APPL	G_DAS	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.37	The Pledger system should be able to execute blockchain network transactions on the edge between participant edge nodes that maintain and are engaged in the blockchain network.		Service Provider	FU_APPL	G_DAS	expertise as domain expert/solution provider
FR.38	The Pledger system must support crypto-token transactions (i.e. Pledger token).		End User	FU_APPL	G_ORC	expertise as domain expert/solution provider
FR.39	The Pledger system must support private smart contracts execution.		End User	FU_APPL	G_ORC, G_DAS	expertise as domain expert/solution provider
FR.40	The Pledger system must provide a tamper-proof and secure ledger to track the exchange of data among edge nodes.		Service Provider	FU_APPL	G_ORC, G_DAS	expertise as domain expert/solution provider
FR.41	The Pledger system should provide a test suite for workload (applications) in order to measure and gather metrics of performance to classify the services .		Service Provider	FU_CORE	G_ORC, G_SEC	expertise as domain expert/solution provider
FR.42	The Pledger system must integrate reports about benchmarking in a dashboard (e.g. charts).		Service Provider	FU_APPL	G_ORC	expertise as domain expert/solution provider
FR.43	The Pledger system must schedule the execution of benchmarking tests on an infrastructure with a custom frequency.		Service Provider	FU_APPL	G_ORC, G_SEC	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.44	The Pledger system should gather metrics from infrastructure and services, including response time, latency and other metrics collected from the monitoring tools.		Infrastructure Provider	FU_CORE	G_ORC, G_SEC	expertise from existing asset
FR.45	The Pledger system would provide advice about the best infrastructure for a specific service based on profiling and classification processes for infrastructure and services (automatic or manual).		System Integrator	FU_CORE	G_UCS	expertise from existing asset
FR.46	The Pledger system must show historical performance about services and infrastructure.		Service Provider, Infrastructure Provider	FU_CORE	G_ORC	expertise from existing asset
FR.47	The Pledger system must allow the definition of the execution plan of services: max/min instances, schedules.	US.11, US.12	Service Provider	FU_CORE	G_UCS, G_BLO	expertise from existing asset
FR.48	The Pledger system should deploy event-driven apps with zero instances (FaaS).		Service Provider	FU_CORE	G_UCS	expertise from existing asset
FR.49	The Pledger system must implement the following recovery actions after an SLA breach: apps scalability, apps migration to a Cloud or Edge.		Service Provider, Infrastructure Provider	FU_CORE	G_UCS	expertise from existing asset
FR.50	The Pledger system must trace data usage/transfer to enable a business model based on pay-per-use approach.		Service Provider	FU_CORE	G_UCS	expertise from existing asset

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.51	The Pledger system should provide a dashboard to check which data has been released to whom.		Service Provider	FU_CORE	G_UCS	expertise from existing asset
FR.52	The Pledger system should configure private execution on private infrastructure and/or with private benchmarking tests.		Service Provider	FU_CORE	G_UCS	expertise as domain expert/solution provider
FR.53	The Pledger system must allow users to keep benchmarking results private.		Service Provider	FU_CORE	G_UCS	expertise as domain expert/solution provider
FR.54	The Pledger system must provide a library of general-purpose micro- and application-level benchmarking tests.		Service Provider	FU_CORE	G_ORC	expertise as domain expert/solution provider
FR.55	The Pledger system must provide an interface to query benchmarking results.		System Integrator	FU_CORE	G_ORC	expertise as domain expert/solution provider
FR.56	The Pledger system must filter, aggregate and group benchmarking results.		System Integrator	FU_CORE	G_UCS, G_ORC	expertise as domain expert/solution provider
FR.57	The Pledger system must provide, for each possible activity option, an efficiency function that weights costs and benefits to help the user pick the best.		Service Provider	FU_CORE	G_UCS	expertise as domain expert/solution provider
FR.58	The Pledger system must allow the user to customize his preferences with respect to different metrics about the best activity option.		System Integrator	FU_CORE	G_UCS	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.59	The Pledger system must provide, for each possible activity option, a suggestion about the best choice based on the user preferences, as a notification or a highlighted option in the dashboard.		Service Provider	FU_CORE	G_UCS	expertise as domain expert/solution provider
FR.60	The Pledger system would support suggestions based on ML algorithms.	US.37, US.33	All	FU_CORE	G_UCS, G_ORC	expertise as domain expert/solution provider
FR.61	The Pledger system could be able to offload computation to remote rendering services in case data is too complex, in order to overcome the device limitations and ensure functionality.	US.41, US.42, US.44	System Integrator	FU_CORE	G_UCS, G_ORC	expertise from existing asset
FR.62	The Pledger system should be able to configure the networking between apps deployed on the cloud, the edge, and the end-user devices.		System Integrator	FU_CORE	G_UCS, G_DAS	expertise from existing asset
FR.63	The Pledger system must be able to register cloud, edge, and radio infrastructures resources while reflecting the available capabilities (e.g. GPU, etc.) of the registered infrastructure.	US.41, US.43	Service Provider	FU_CORE	G_UCS	expertise from existing asset
FR.64	The Pledger system could be able to configure and use the computation and radio resources of the citywide testbed that the infrastructure provider makes available.	US.41	Service Provider	FU_CORE	G_UCS	expertise from existing asset

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.65	The Pledger system could allow to dedicate a part of the testbed-wide resources to a specific tenant, i.e., offer a dedicated slice for computation and radio resources.	US.45	Service Provider	FU_CORE	G_BLO	expertise from existing asset
FR.66	The Pledger system could support multiple connectors to stream big data.	US.45	Service Provider	FU_CORE	G_BLO	expertise from existing asset
FR.67	The Pledger system could deploy workload in edge and cloud infrastructure.	US.44	Infrastructure Provider	FU_CORE	G_BLO	expertise from existing asset
FR.68	The Pledger system must import metrics from external sources (monitoring tools, system logs, etc.).	US.41, US.43, US.46	System Integrator	FU_CORE	G_BLO	expertise from existing asset
FR.69	The Pledger system should allow the configuration of different notification channels (webhook, external api call, email, etc.).	US.41, US.43	Service Provider	FU_CORE	G_DAS, G_BLO	expertise from existing asset
FR.70	The Pledger system must support the optimal selection of the least-cost path towards each edge location.	US.41, US.43	Service Provider	FU_CORE	G_ORC	expertise from existing asset
FR.71	The Pledger system must support deployment of apps on edge node with ML capabilities.	US.41, US.42	Service Provider	FU_CORE	G_ORC	expertise from existing asset
FR.72	The Pledger system should be able to load 3D CAD files which can be viewed live in a room through a HoloLens or a similar device.	US.48	Service Provider	FU_CORE	G_ORC	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.73	The Pledger system should support blockchain for ID management. Possible input can be user credentials, but also eye tracking, cloud-based voice recognition, SLAM based room recognition.	US.48	Service Provider	FU_CORE	G_ORC	expertise as domain expert/solution provider
FR.74	The Pledger system would support world anchors or image tracking as reference points, so that AR devices can remember and load the CAD file at the same location for the next iteration.	US.49	Service Provider	FU_CORE	G_ORC	expertise as domain expert/solution provider
FR.75	The Pledger system must allow collaboration among users through shared experience functionality.	US.50	Service Provider	FU_CORE	G_ORC	expertise as domain expert/solution provider
FR.76	The Pledger system must allow, during the shared experience sessions, all off-site members to be displayed as avatars to allow a more intuitive interaction.	US.51	Service Provider	FU_CORE	G_BLO	expertise as domain expert/solution provider
FR.77	The Pledger system must allow, when having more than one CAD file loaded, or when displaying a file 1x1 in a real life location, areas of overlapping or collision to be highlighted in red.	US.51	Service Provider	FU_CORE	G_BLO	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Groups	Source
FR.78	The Pledger system should allow the addition of annotations, notes, etc, as well as measuring distances or drawing of basic shapes to the CAD file.	US.51	Service Provider	FU_CORE	G_BLO	expertise as domain expert/solution provider
FR.79	The Pledger system must export CAD files changes to CAD repositories, so that development improvements are saved and can be used for the next iteration of the design process.	US.51	Service Provider	FU_CORE	G_BLO	expertise as domain expert/solution provider

### 4.3 Non-Functional Requirements

This subsection contains the list of Pledger non-functional requirements, using the following **categories**:

- ▶ Scalability (“NF\_SCA”): the ability to easily upscale or downscale when required;
- ▶ Performance (“NF\_PER”): related to specific expected performance indicators;
- ▶ Reliability and availability (“NF\_AVA”): related to the resiliency of the platform w.r.t. possible service or network interruptions;
- ▶ Manageability (“NF\_MAN”): related to ability to easily manage the platform;
- ▶ Modularity (“NF\_MOD”): about the separation of different components into independent yet interrelated pieces;
- ▶ Security (“NF\_SEC”): about security related features;
- ▶ Hardware and Software constraints (“NF\_CTR”): about hardware and software constraints due to the integration of existing assets;
- ▶ Openness and Extensibility (“NF\_EXT”), the ability to easily extend and integrate the platform with other components, e.g. with compliance to a specific standard.

The Table 13 reports the list of the non-functional requirements, with the same structure as above.

Table 13: Non-functional requirements

Code	Description	User stories	Stakeholders	Category	Source
NFR.01	The Pledger system should expose REST APIs for control/management plane with the HTTPS protocol.		System Integrator	NF_SEC	state of the art / best practices
NFR.02	The Pledger system should be designed for high availability and fault tolerance of the solution avoiding SPF (single point of failure) as far as possible.		System Integrator	NF_AVA	expertise from existing asset
NFR.03	The Pledger system must be tested to support at least a few Cloud Management Toolkits (Kubernetes, OpenStack, etc.)		System Integrator	NF_EXT	state of the art / best practices
NFR.04	The Pledger system should define SLA contracts in JSON format following some standards (predicates, operators, etc.) like JSONPath or similar.		System Integrator	NF_EXT	expertise from existing asset

Code	Description	User stories	Stakeholders	Category	Source
NFR.05	The Pledger system should allow infrastructure provisioning (IaC) in declarative format (JSON, YAML, etc.).		System Integrator	NF_MAN	state of the art / best practices
NFR.06	The Pledger system could allow to scale-up/down benchmarking executors.	US.47	Infrastructure Provider	NF_SCA	state of the art / best practices
NFR.07	The Pledger system should expose a standard querying API for benchmarking results (e.g. PromQL).	US.46	System Integrator	NF_EXT	expertise from existing asset
NFR.08	The Pledger system would provide a standard format to include new benchmarking tests (e.g. Docker images).	US.42	Service Provider	NF_EXT	expertise from existing asset
NFR.09	The Pledger system must have credentials encrypted in the benchmarking backend.	US.45	Service Provider	NF_SEC	expertise from existing asset
NFR.10	The Pledger system would provide an extensible way to exchange metrics and actions for the DSS computation.		Service Provider	NF_EXT	expertise from existing asset
NFR.11	The Pledger system should support a packaging format that allows the building of new artefacts in few seconds.	US.30	Service Provider	NF_PER	state of the art / best practices
NFR.12	The Pledger system could support a packaging format that allows the instantiation of new artefacts in few seconds.	US.30	Service Provider	NF_PER	state of the art / best practices

Code	Description	User stories	Stakeholders	Category	Source
NFR.13	The Pledger system could support the deployment on edge nodes where apps can run autonomously in case of disconnection from the cloud.	US.33	Service Provider	NF_AVA	state of the art / best practices
NFR.14	The Pledger system would support edge orchestrators that are CNCF compatible with Kubernetes.	US.33	Service Provider	NF_EXT	state of the art / best practices
NFR.15	The Pledger system must support a packaging format with huge availability of artefacts from the open-source community.	US.33	Service Provider	NF_EXT	state of the art / best practices
NFR.16	The Pledger system must support redundant connection paths for resiliency.	US.33	Infrastructure Provider	NF_AVA	state of the art / best practices
NFR.17	The Pledger system could support the easy configuration of multi-cloud connections.		Infrastructure Provider	NF_EXT	state of the art / best practices
NFR.18	The Pledger system could support multi tenancy both on cloud and edge resources.		All	NF_SEC	state of the art / best practices
NFR.19	The Pledger system must support connections among large numbers of edge locations having overlapping IP sub-networks.	US.33	Infrastructure Provider	NF_CTR	state of the art / best practices
NFR.20	The Pledger system would integrate with the most popular infrastructures for executing benchmarking tests (e.g. Openstack, Kubernetes, VMWare).		Service Provider	NF_EXT	state of the art / best practices

Code	Description	User stories	Stakeholders	Category	Source
NFR.21	The Pledger system must support edge nodes that can monitor and restart apps automatically in case of major issues, track and report the failure.	US.33	Service Provider	NF_AVA	state of the art / best practices
NFR.22	The Pledger system could support a packaging format that supports different GPUs and related SDK (e.g. NVidia).		Service Provider	NF_EXT	state of the art / best practices
NFR.23	The Pledger system could support deployment of apps on edge nodes using low cost HW with possible support to GPUs	US.32	Service Provider	NF_EXT	state of the art / best practices
NFR.24	The Pledger system should synchronize all data from edge to cloud every 5 seconds		Service Provider	NF_PER	expertise as domain expert/solution provider
NFR.25	The Pledger system could enable the deployment of applications as Docker containers on edge		System Integrator	NF_CTR	expertise as domain expert/solution provider
NFR.26	The Pledger system must support training of ML/DL models on the cloud through specialised hardware (e.g. GPU)	US.22	Service Provider	NF_CTR	expertise as domain expert/solution provider
NFR.27	The Pledger system must enable the transfer of big data from edge to the cloud (15 GB/day)	US.22	System Integrator	NF_PER	expertise as domain expert/solution provider
NFR.28	The Pledger system could enable Python to be used as development language for the applications		Service Provider	NF_CTR	expertise as domain expert/solution provider
NFR.29	The Pledger system could be able to recover from disconnections, work autonomously in the meanwhile and resync data afterwards		System Integrator	NF_AVA	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Source
NFR.30	The Pledger system must transfer data in a secure way to prevent unwanted third-party access.	US.25	End user	NF_SEC	expertise as domain expert/solution provider
NFR.31	The Pledger system must support a packaging format that supports heterogenous architectures for both cloud and edge, such as X86 and ARM.		Service Provider, Infrastructure Provider	NF_EXT	expertise from existing asset
NFR.32	The Pledger system must produce a warning for a VRU needs to be issued in time to avoid any risky situation giving enough time to react (5-10s).		Service Provider	NF_PER	expertise as domain expert/solution provider
NFR.33	The Pledger system must assure isolation between different services running in the infrastructure with different tenants.		Service Provider	NF_SEC	expertise as domain expert/solution provider
NFR.34	The Pledger system must require minimum intervention over end user devices.		Service Provider	NF_AVA	expertise as domain expert/solution provider
NFR.35	The Pledger system must support simple and secure administration for pluggable images (e.g. Docker images).		System Integrator	NF_PER	expertise as domain expert/solution provider
NFR.36	The Pledger system must support decentralized applications which are supported by grouped smart contracts that are programmed to execute autonomously on the blockchain network and deliver different application services when triggered.		Service Provider	NF_MOD	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Source
NFR.37	The Pledger system would be able to ensure data integrity of edge node and user data while keeping specific information on the Pledger blockchain and retrieving the requested data with secure queries.		System Integrators	NF_SEC	expertise as domain expert/solution provider
NFR.38	The Pledger system must allow the development of applications running on the Pledger blockchain that enable a modular architecture while providing performance at scale with preserving privacy.		All	NF_MOD	expertise as domain expert/solution provider
NFR.39	The Pledger system must manage big data in compliance with the European legislative framework with respect to data protection.	US.33, US.37, US.38, US.39	All	NF_SEC	expertise as domain expert/solution provider
NFR.40	The Pledger system must have encryption applied to the big data streams.	US.34, US.37, US.38, US.39	System Integrators	NF_SEC	expertise as domain expert/solution provider
NFR.41	The Pledger system must have authorisation and authentication management for the big data so that only specified connections can access to pub/sub data streams.	US.37, US.38, US.39	System Integrators	NF_SEC	expertise as domain expert/solution provider
NFR.42	The Pledger system must have secure connections among cluster components and also among communications clients and clusters.	US.37, US.38, US.39	System Integrators	NF_SEC	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Source
NFR.43	The Pledger system must have authenticated access to an easy-to-use interface for the administration of the big data platform.	US.37, US.38, US.39	System Integrators	NF_MAN	expertise as domain expert/solution provider
NFR.44	The Pledger system must have multiple brokers to maintain replication and load balancing, contributing to the overall elasticity and fault tolerance of Pledger.	US.37, US.38, US.39	System Integrators	NF_AVA	expertise as domain expert/solution provider
NFR.45	The Pledger system must have the capacity to manage data exchanged between core, cloud and edge.	US.37, US.38, US.39	System Integrators	NF_PER	expertise as domain expert/solution provider
NFR.46	The Pledger system must have the capacity to handle both real time data and batch processing pipelines.	US.37, US.38, US.39	System Integrators	NF_PER	expertise as domain expert/solution provider
NFR.47	The Pledger system would support the capacity to interface with different storages (e.g. DBMS, and so on).	US.37, US.38, US.39	System Integrators	NF_EXT	expertise as domain expert/solution provider
NFR.48	The Pledger system must enforce security among the edge nodes and among single services at IP level (e.g. with firewalls).	US.37, US.38, US.39	System Integrators	NF_PER	expertise as domain expert/solution provider
NFR.49	The Pledger system must provide a security and privacy risk assessment framework to list the trusted nodes that have already proved their ability to comply with the requirements provided by the users.	US.37, US.33	All	NF_SEC	expertise as domain expert/solution provider

Code	Description	User stories	Stakeholders	Category	Source
NFR.50	The Pledger system must use encryption to enforce storage security.	US.37, US.33	All	NF_SEC	expertise as domain expert/solution provider
NFR.51	The Pledger system must enforce secure communication and authorisation management on the entire CI pipeline.	US.33, US.37, US.39, US.40	All	NF_SEC	expertise as domain expert/solution provider

## 4.4 Requirements report

This subsection aims at highlighting the requirements relations to the user stories, the possible dependencies and the distribution among the different sources and categories.

Figure 26 shows the functional requirements' categories distribution, with core requirements having higher percentage.

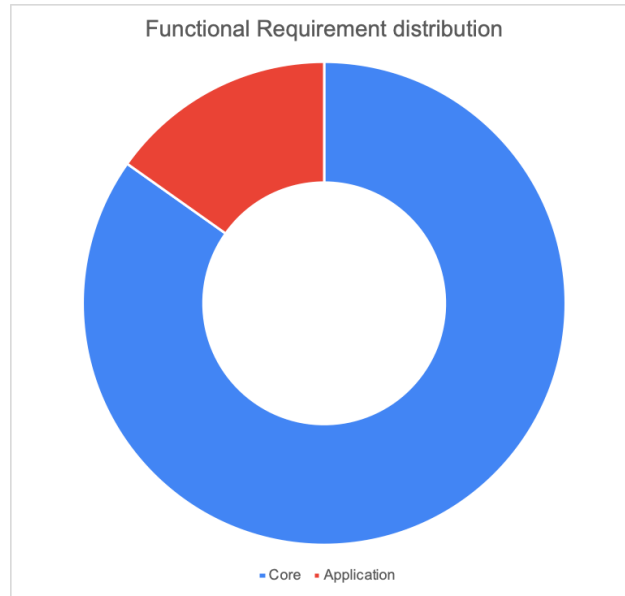


Figure 26: Functional requirements' distribution categories

Figure 27 shows the functional requirements' groups, with orchestration and use cases features having the higher shares.

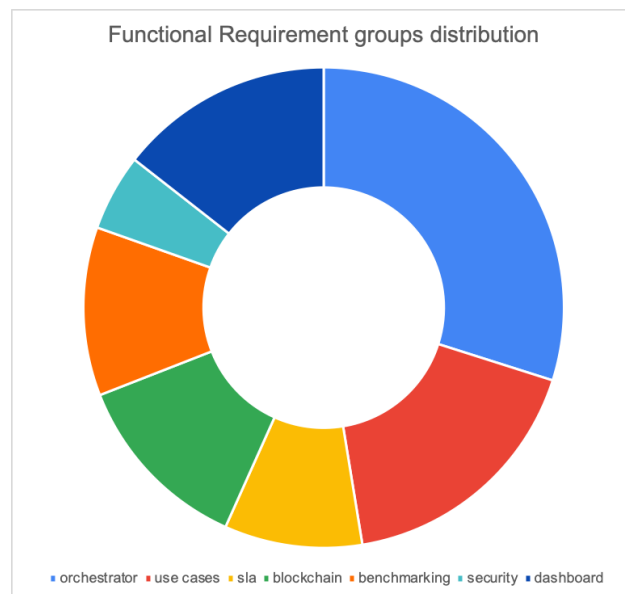


Figure 27: Functional requirements' groups

Figure 28 shows the non-functional requirements' categories distribution, with scalability and performance higher percentages.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	90 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

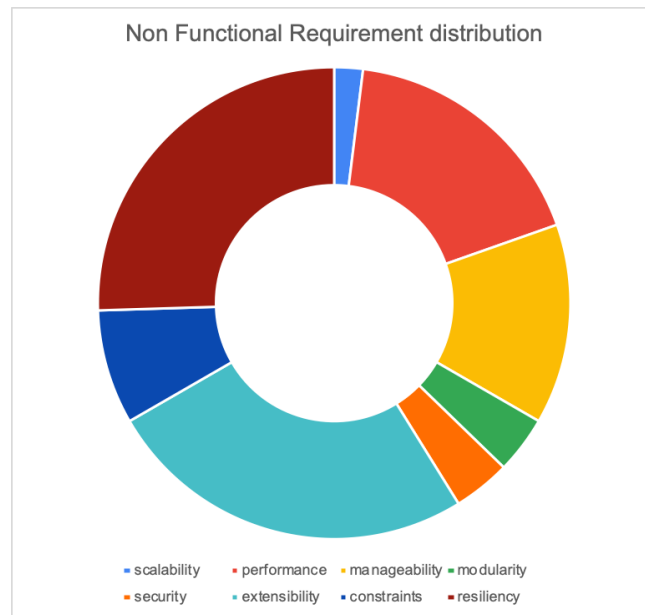


Figure 28: Non-functional requirements' distribution categories

Figure 29 shows the sources used to extract the requirements, with a higher share about expertise as domain expert or solution provider.

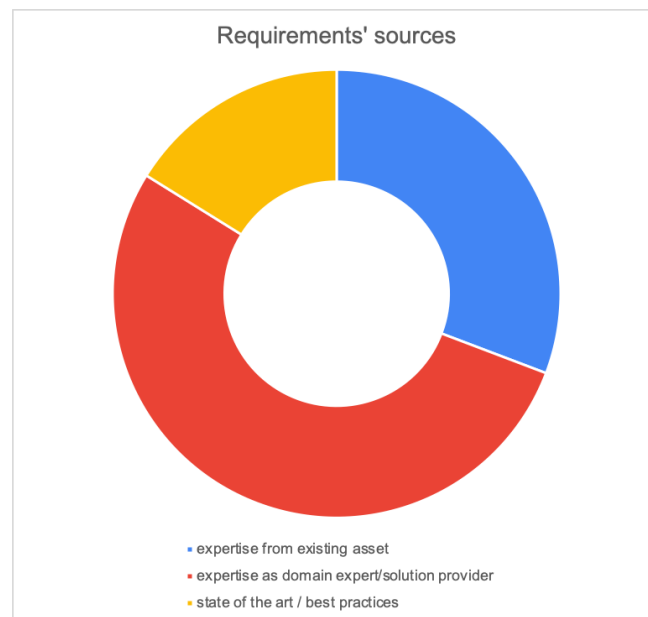


Figure 29: Requirements' sources

The last report, shown in Figure 30, represents the involved stakeholders distribution, with a clear predominance of the service providers, followed by the system integrators and the others stakeholders.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	91 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

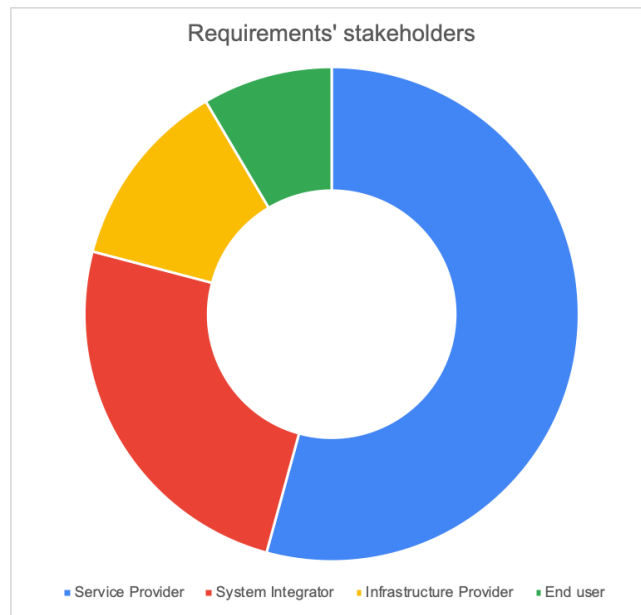


Figure 30: Requirements' stakeholders

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	92 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
		<b>Version:</b>	2.0
		<b>Status:</b>	Final

## 5 Pledger Minimal Viable Product

The objective of this section is to explain the Minimal Viable Product (MVP) concept and how it is used in Pledger to derive the architecture that will be part of deliverable D2.3 and the development of the prototype. The result is a list of requirement priorities, agreed by all the partners' project, that aims at improving the commitment within the consortium, cut the less relevant features and focus on the most valued features to be presented by the demonstrators at the end of the project.

### 5.1 Definition

The use of a Minimum Viable Product (MVP) for Pledger aims at improving the value produced by the platform via strong involvement of stakeholders from the beginning of the project. This approach is considered the basis of the "Lean Startup"<sup>161</sup> method, that aims at *building* artefacts as efficiently as possible through the continuous delivery of artifacts, *measuring* the impact on the expected business value, *learning* and addressing the changes for the next development iteration. Figure 31 shows the MVP build-measure-learn cycle and summarize the iterations that will take place after the D2.2 release in order to track the actual progress and include possible variations and new requirements that could be elicited in the next period.

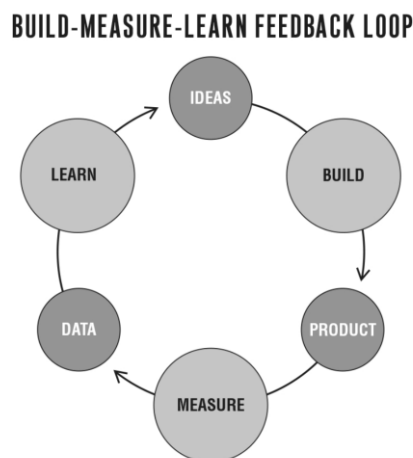


Figure 31: build-measure-learn cycle (source: Eric Ries)

As a result, the MVP is used to identify the first prototype's features and continuously verify the outputs against the stakeholders' perspective to address the Pledger incremental evolution.

### 5.2 MVP methodology in Pledger

The approach used for the requirements prioritization is based on the identification of three specific attributes valued from 1 to 5:

- ▶ business value
- ▶ urgency
- ▶ technical feasibility

<sup>161</sup> <http://theleanstartup.com/principles>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	93 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b>
			Final

The **business value** is a qualitative estimation of the need to have a specific feature in the product; at this stage, it is mostly related to the frequency and the number of expected users involved and it will be better addressed in the WP6 activities.

The **urgency** represents the need to have an homogeneous product with some basic core and initial vertical features. It is impacted by dependencies with other requirements and from key missing pieces from the core that would not allow specific verticals (e.g. no blockchain at all).

The **technical feasibility** is the estimation of the technical difficulties to implement a specific requirement, 5 means it is very easy to be implemented.

The most relevant aspect of this approach is the need for all the consortium partners to have a common understanding of these three points in order to vote, while the individual perspective is still preserved even on those requirements that are more technical and less intuitive for some partners.

Eventually, the votes aggregation and final discussion about the minimal threshold to adopt leads to and strengthen the necessary commitment to move on to the next step, about development and integration, with a complete set of features of the Pledger demonstrators.

A spreadsheet with all the requirements has been shared within the consortium to evaluate each attribute with a score from 1 to 5; for each requirement, the score values have been averaged and normalized to 100 for better readability  $\left(\frac{\text{business value} + \text{urgency} + \text{feasibility}}{5+5+5} * 100\right)$ , then used to determine the priority of each requirement computing the average among all the partners  $\left(\frac{\sum_{j=1}^{\text{Number of partners}} \text{req\#i.score}(j)}{\text{Number of partners}}\right)$ .

Table 14 shows the table used by the Pledger consortium to decide whether to include or not a requirement into the MVP. The Annex contains more details about the spreadsheet used.

Table 14: sample table for the MVP voting

Req. ID	Partner	Score (1-5) for			Total Score
		Business Value, Urgency, Feasibility			
Req#1	Partner#A	3	4	4	38
Req#1	Partner#B	4	4	5	64

According to the voting results, the requirements descriptions have been adjusted using the MoSCoW method<sup>162</sup> to prioritize and share a common understanding of the importance of each requirement among the consortium's partners.

The term MoSCoW is an acronym derived from the first letter of each of four prioritization categories (**M**ust have, **S**hould have, **C**ould have, and **W**ould have). “Must” requirements are considered as those with the highest priority that must drive the next development phase; those with “Should” are still very relevant although scheduled right after the “Must” ones; “Could” represents some highly desirable features that would increase the final value of Pledger and “Would” are requirements that are identified but considered not crucial for this first iteration and that will be evaluated along with possible new requirements in the next year of activities.

A final check has been done with the requirement's owner to either accept the new priority got from the MVP voting or change it, so finally consolidate the values and define of the MVP thresholds.

Table 15 shows the thresholds used for MoSCoW prioritization, agreed among the consortium, with the requirements belonging to each category.

<sup>162</sup> [https://www.agilebusiness.org/page/ProjectFramework\\_10\\_MoSCoWPrioritisation](https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation)

Table 15: MoSCoW thresholds for MVP

MoSCoW	Thresholds	Requirements count	Percentage
Must	MVP $\geq$ 75	69	53
Should	70 $\leq$ MVP < 75	34	26
Could	65 $\leq$ MVP < 70	19	15
Would	MVP < 65	8	6

Figure 32 shows the percentage of the MoSCoW priorities for all the requirements, with higher percentages for “must” and “should”.

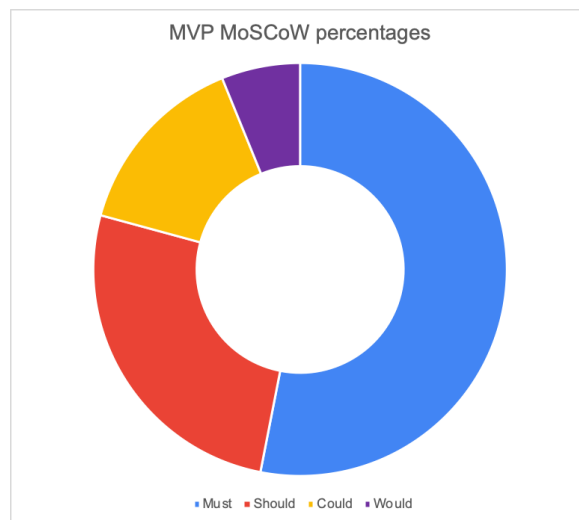


Figure 32: MVP MoSCoW requirements' percentages

For the first development iteration, the consortium agreed to include in the MVP the requirements with **score  $\geq$  70**, which includes (at least) the MoSCoW priorities « must » and « should ». Figure 33 shows the percentage of functional and non-functional requirements that have been considered as part of the MVP (in blue) and those that are not (in red).

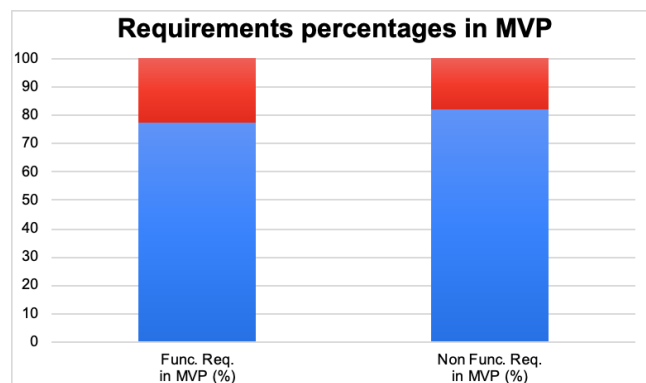


Figure 33: percentage of requirements that are part of the MVP

The selection of such thresholds along with the priorities assigned to each requirement, confirmed by its owner, have been agreed among all partners in order to improve the communication and common understanding of the next activities to be carried out in the next period.

## 6 Conclusions and next steps

---

This deliverable contains the list of requirements of the Pledger system at M9.

The methodology used to elicit and consolidate the requirements is described in Section 1, along with the time plan for the next iterations. The user's stories in Section 2 are used to identify the relevant stakeholders before proceeding to the requirements' formalization. Section 3 describes the risks and benefits of adopting some specific technologies that are considered relevant by the consortium expertise and include a short SWOT analysis to summarize them. Section 4 lists the requirements and groups them together by similar properties, while Section 5 focuses on their prioritization using MVP through a shared spreadsheet to be used as a common reference for the next activities about the architecture of the Pledger system.

Such spreadsheet will be also used to update the requirements through regular checks that will take place within the consortium in order to trace, review and approve them along the project lifecycle. This process will lead to an updated release of D2.2 at M24.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	96 of 101				
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	<b>Version:</b>	2.0	<b>Status:</b>	Final

## 7 Bibliography

- A. Das, S. P. (2018). Edgebench: Benchmarking edge computing platforms.
- Amatriain, X. (2013). Big & Personal: data and models behind Netflix recommendations. *The 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. Chicago, Illinois, USA.
- Armstrong, T. P. (2013). LinkBench: a database benchmark based on the Facebook social graph. ., (pp. pp. 1185-1196).
- Baresi, L. &. (2019). A Unified Model for the Mobile-Edge-Cloud Continuum.
- Behzad Boroujerdian, H. G. (2019). MAVBench: Micro Aerial Vehicle Benchmarking.
- BF Cooper, A. S. (2010). Benchmarking Cloud Serving Systems with YCSB in Proceedings of the 1st ACM Symposium on Cloud Computing. p. 143–154 (2010).
- Bittencourt, L. F. (2018). The Internet of Things, Fog and Cloud Continuum: Integration and Challenges.
- Blesson Varghese, N. W.-H. (2020). A Survey on Edge Benchmarking.
- Borhani, A. L. (2014). WPress: An Application-Driven Performance Benchmark for Cloud-Based Virtual Machines. ., (pp. pp. 101-109).
- Boyd, D., & Crawford, K. (2011). Six Provocations for Big Data. *Social Science Research Network: A Decade in Internet time: Symposium on the Dynamics of the Internet and Society*. doi:<https://doi.org/10.2139%2Fssrn.1926431>
- C. Lee, M. L. (2019). Iotbench: A Benchmark Suite for Intelligent Internet of Things Edge Devices," 2019 IEEE International Conference on Image Processing (ICIP)., (pp. pp. 170-174).
- Difallah, D. P.-M. (2013). OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases. Proc. VLDB Endow., 7(4), ., (pp. p.277–288).
- Dyck, A., Penners, R., & Lichter, H. (2015). Towards Definitions for Release Engineering and DevOps. *Proceedings of the 2015 IEEE/ACM 3rd International Workshop on Release Engineering*.
- E. Coronado, S. N. (2019). 5G-EmPOWER: A Software-Defined Networking Platform for 5G Radio Access Networks., (pp. vol. 16, no. 2, pp. 715-728).
- Emeakaroha, V. C. (2011). Casvid: Application level monitoring for sla violation detection in clouds.
- Enns, R. E. (2011). Network Configuration Protocol (NETCONF), RFC 6241,.
- Ferdman, M. A. (2012). Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware. In Proceedings of the Seventeenth .
- Gan, Y. J. (2019). An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems.
- Gao, W. L. (2018). AIBench: towards scalable and comprehensive datacenter AI benchmarking.
- Garg, S., Versteeg, S., & Buyya, R. (2011). Smicloud: A Framework For Comparing And Ranking Cloud Services.
- Ghosh, N., Ghosh, S., & Das, S. (2015). Selcsp: A Framework To Facilitate Selection Of Cloud Service Providers., (pp. Vol.3, No.1, Pp.66,79).
- H. Khalili, e. a. (2019). Network Slicing-aware NFV Orchestration for 5G Service Platforms," 2019 European Conference on Networks and Communications (EuCNC). Valencia, Spain.
- Hao Wu, F. L. (2016). Cloud Server Benchmarks for Performance Evaluation of New Hardware Architecture.
- Hong, C. a. (2019). Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms., (pp. ACM Computing Surveys, 52, p.1-37).
- International Standards Organisation (ISO). (2016). *ISO/IEC 20922:2016 Information Technology - Message Queuing Telemetry Transport (MQTT) v3.1.1*. Retrieved May 2020, from <https://www.iso.org/standard/69466.html>

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	97 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

- K. Toczé, N. S.-T. (2019). Towards Edge Benchmarking: A Methodology for Characterizing Edge Workloads," 2019 IEEE 4th International Workshops on Foundations and Applications of Self\* Systems (FAS\*W).
- Kent, B. (2002). *Test-Driven Development by Example*. Addison Wesley.
- Khan, N., Yaqoob, I., Hashem, I., Inayat, Z., Kamaleldin, W., Alam, M., . . . Gani, A. (2014). Big Data: Survey, Technologies, Opportunities, and Challenges. (18.10.1155/2014/712826).
- Kousiouris, G. (2017). A Toolkit Based Architecture for Optimizing Cloud Management, Performance Evaluation and Provider Selection Processes," 2017 International Conference on High Performance Computing & Simulation (HPCS)., (pp. pp. 224-232).
- Luo, C. &. (2019). AIoT Bench: Towards Comprehensive Benchmarking Mobile and Embedded Device Intelligence. .
- Luo, C. Z. (2012). CloudRank-D: Benchmarking and ranking cloud computing systems for data processing applications. *Frontiers of Computer Science*.
- M. P. Mena, A. P.-A. (2020). Enhancing the Performance of 5G Slicing Operations via Multi-tier Orchestration.
- MarketsAndMarkets. (2019). *DevOps Market by Type (Solutions and Services), Deployment Model (Public, Private, and Hybrid), Organization Size, Vertical (BFSI, Healthcare, Telecommunications & ITES, Manufacturing), and Region - Global Forecast to 2023* .
- MarketsAndMarkets. (n.d.). *Streaming Analytics Market by Component, Application (Predictive Asset Management, Risk Management, Location Intelligence, Sales and Marketing, Supply Chain Management), Industry Vertical, Deployment Model, and Region - Global Forecast to 2024*. Retrieved from <https://www.marketsandmarkets.com/Market-Reports/streaming-analytics-market-64196229.html>
- Muller, S. B. (2014). Benchmarking the Performance Impact of Transport Layer Security in Cloud Database Systems., (pp. pp. 27-36).
- Olguín, M. &. (2019). EdgeDroid: An Experimental Approach to Benchmarking Human-in-the-Loop Applications.
- Patil, S. &. (2011). YCSB++: Benchmarking and Performance Debugging Advanced Features in Scalable Table Stores. .
- Qiu, X. &. (2019). Online Deep Reinforcement Learning for Computation Offloading in Blockchain-Empowered Mobile Edge Computing.
- R. Yang, F. R. (2019). Integrated Blockchain and Edge Computing Systems: A Survey, Some Research Issues and Challenges., (pp. vol. 21, no. 2, pp. 1508-1532).
- Reuther, A. M. (2019). Survey and Benchmarking of Machine Learning Accelerators. 2019 IEEE High Performance Extreme Computing Conference (HPEC).
- Shao, J. a. (2011). A performance guarantee approach for cloud applications based on monitoring.
- Shen, Z. e. (2011). Cloudscale: elastic resource scaling for multi-tenant cloud systems.
- Shi, W. &. (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*. 3. 1-1. 10.1109/JIOT.2016.2579198. .
- Sumbaly, R., Kreps, J., & Shah, S. (2013). The "Big Data" Ecosystem at LinkedIn. *2013 ACM SIGMOD International Conference on Management of Data*. New York, USA.
- Taherizadeh, S. e. (2018). Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review.
- Taherizadeh, S. e. (2018). Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review.
- Tianshu Hao, Y. H. (2019). Edge AIBench: Towards Comprehensive End-to-end Edge Computing Benchmarking.
- Vijay Janapa Reddi, C. C.-J. (2019). MLPerf Inference Benchmark.
- W. Shi, J. C. (2016). "Edge Computing: Vision and Challenges,". (pp. pp. 637-646). *IEEE Internet of Things Journal*, vol. 3.
- Weisong Shi, J. C. (2006). Edge Computing: Vision and Challenges., (pp. pagg. 2-3).

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	98 of 101	
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	
	<b>Version:</b>	2.0	<b>Status:</b>	Final

- Xenofon Foukas, N. N. (2016). FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks. In Proceedings of the 12th International on Conference on emerg.
- Xiao, Y. a. (2017). QoE and power efficiency tradeoff for fog computing networks with fog node cooperation. .
- Xiaolong, J., W.Wah, B., Cheng, X., & Wang, Y. (2015). Significance and Challenges of Big Data Research. *Big Data Research*, 2(2), 59-64. doi:<https://doi.org/10.1016/j.bdr.2015.01.006>
- Xiong, P. e. (2013). vPerfGuard: an automated model-driven framework for application performance diagnosis in consolidated cloud environments.

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	99 of 101				
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU	<b>Version:</b>	2.0	<b>Status:</b>	Final

## Annexes

This section contains the description of the shared spreadsheet used to keep track of the user stories, the requirements and the MVP voting activities along the project lifecycle.

### README tab, with instructions and rationale

The table below reports the “README” tab content that contains the basic instructions and about how to use the spreadsheet.

### MVP\_\*\*\* tabs

The “MVP\_\*\*\*” tabs contains the MVP votes from each partner. The “MVP\_ALL” tab contains the average MVP results computed from the MVP\_\*\*\* tabs from each partner, with the score normalized to 100. The table below shows an extract from such table.

Req.ID	Mean Business Value (1-5)	Mean Urgency (1-5)	Mean Technical feasibility (1-5)	Total mean score normalised to 100	MVP (yes/no)
FR.01	3.67	4.44	3,00	74	yes
FR.02	3.00	3.67	4,00	71	yes
FR.03	3.44	4.78	4.00	81	yes
FR.04	3.11	4.67	2.89	71	yes
FR.05	3.67	4.44	2.89	73	yes
FR.06	3.67	3.56	3.22	70	no

Along with the MVP results, the “MVP\_ALL” tab also contains a table, reported below, that allows to change the MVP thresholds and automatically computes the percentages of the requirements falling into each MoSCoW category. This help to identify more easily the actual weight received by each partner and the actual amount of requirements that should be prioritized or cut out from the MVP.

MoSCoW	Thresholds	Requirements count	Percentage	Threshold
<b>Must</b>	MVP >= 75	69	53	75
<b>Should</b>	70 <= MVP < 75	34	26	70
<b>Could</b>	65 <= MVP < 70	19	15	65
<b>Would</b>	MVP < 65	8	6	-
Total requirements		130		

### MoSCoW\_FR and MoSCoW\_NFR tabs, with changed priorities

The MoSCoW\_FR and MoSCoW\_NFR tabs contains the list of the functional and non-functional requirements with an additional column to highlight if the requirement has changed its MoSCoW priority after the MVP voting activities. This is used to get feedback from the requirement owner and

<b>Document name:</b>	D2.2 Pledger Requirements Analysis	<b>Page:</b>	100 of 101
<b>Reference:</b>	D2.2	<b>Dissemination:</b>	PU
	<b>Version:</b>	2.0	<b>Status:</b> Final

possibly adjust the voting afterwards. The table below shows an extract of the table used to change the MoSCoW priorities.

Original description from Requirement tab	New MoSCoW from MVP voting	New Description with MoSCoW from MVP	Did MoSCoW change with MVP?	New MoSCoW from the req. owner	Final Description
The Pledger system should do...	could	The Pledger system could do...	TRUE	would	The Pledger system would do...